

# NAG Library Function Document

## nag\_dgghd3 (f08wfc)

### 1 Purpose

nag\_dgghd3 (f08wfc) reduces a pair of real matrices  $(A, B)$ , where  $B$  is upper triangular, to the generalized upper Hessenberg form using orthogonal transformations.

### 2 Specification

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

void nag_dgghd3 (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\_dgghd3 (f08wfc) 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\_dggbal (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). The driver, nag\_dggev3 (f08wcc), solves the real generalized eigenvalue problem by combining all the required steps including those just listed.

nag\_dgghd3 (f08wfc) 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

$$\begin{aligned} Q^T A Z &= H, \\ Q^T B Z &= T \end{aligned}$$

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

$$\begin{aligned} Q_1 A Z_1^T &= (Q_1 Q) H (Z_1 Z)^T, \\ Q_1 B Z_1^T &= (Q_1 Q) T (Z_1 Z)^T. \end{aligned}$$

### 4 References

Golub G H and Van Loan C F (2012) *Matrix Computations* (4th Edition) Johns Hopkins University Press, Baltimore

Moler C B and Stewart G W (1973) An algorithm for generalized matrix eigenproblems *SIAM J. Numer. Anal.* **10** 241–256

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation for a more detailed explanation of the use of this argument.

*Constraint:* **order** = Nag\_RowMajor or Nag\_ColMajor.

2: **compq** – Nag\_ComputeQType *Input*

*On entry:* specifies the form of the computed orthogonal matrix  $Q$ .

**compq** = Nag\_NotQ  
Do not compute  $Q$ .

**compq** = Nag\_InitQ  
The orthogonal matrix  $Q$  is returned.

**compq** = Nag\_UpdateSchur  
**q** must contain an orthogonal matrix  $Q_1$ , and the product  $Q_1Q$  is returned.

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

3: **compz** – Nag\_ComputeZType *Input*

*On entry:* specifies the form of the computed orthogonal matrix  $Z$ .

**compz** = Nag\_NotZ  
Do not compute  $Z$ .

**compz** = Nag\_InitZ  
The orthogonal matrix  $Z$  is returned.

**compz** = Nag\_UpdateZ  
**z** must contain an orthogonal matrix  $Z_1$ , and the product  $Z_1Z$  is returned.

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

4: **n** – Integer *Input*

*On entry:*  $n$ , the order of the matrices  $A$  and  $B$ .

*Constraint:*  $n \geq 0$ .

5: **ilo** – Integer *Input*

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

7: **a**[*dim*] – double *Input/Output*

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

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

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

*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: **pda** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **a**.  
*Constraint:* **pda**  $\geq$   $\max(1, \mathbf{n})$ .
- 9: **b**[*dim*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **b** must be at least  $\max(1, \mathbf{pdb} \times \mathbf{n})$ .  
The (*i*, *j*)th element of the matrix *B* is stored in  

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

$$\mathbf{b}[(i-1) \times \mathbf{pdb} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
*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:* **b** is overwritten by the upper triangular matrix *T*.
- 10: **pdb** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **b**.  
*Constraint:* **pdb**  $\geq$   $\max(1, \mathbf{n})$ .
- 11: **q**[*dim*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **q** must be at least  

$$\max(1, \mathbf{pdq} \times \mathbf{n}) \text{ when } \mathbf{compq} = \text{Nag\_InitQ or Nag\_UpdateSchur};$$

$$1 \text{ when } \mathbf{compq} = \text{Nag\_NotQ}.$$
The (*i*, *j*)th element of the matrix *Q* is stored in  

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

$$\mathbf{q}[(i-1) \times \mathbf{pdq} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
*On entry:* if **compq** = Nag\_UpdateSchur, **q** must contain an orthogonal matrix  $Q_1$ .  
If **compq** = Nag\_NotQ, **q** is not referenced.  
*On exit:* if **compq** = Nag\_InitQ, **q** contains the orthogonal matrix *Q*.  
If **compq** = Nag\_UpdateSchur, **q** is overwritten by  $Q_1Q$ .
- 12: **pdq** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **q**.  
*Constraints:*  
if **compq** = Nag\_InitQ or Nag\_UpdateSchur, **pdq**  $\geq$   $\max(1, \mathbf{n})$ ;  
if **compq** = Nag\_NotQ, **pdq**  $\geq$  1.
- 13: **z**[*dim*] – double *Input/Output*  
**Note:** the dimension, *dim*, of the array **z** must be at least  

$$\max(1, \mathbf{pdz} \times \mathbf{n}) \text{ when } \mathbf{compz} = \text{Nag\_UpdateZ or Nag\_InitZ};$$

$$1 \text{ when } \mathbf{compz} = \text{Nag\_NotZ}.$$
The (*i*, *j*)th element of the matrix *Z* is stored in  

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

$$\mathbf{z}[(i-1) \times \mathbf{pdz} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}.$$
*On entry:* if **compz** = Nag\_UpdateZ, **z** must contain an orthogonal matrix  $Z_1$ .

If **compz** = Nag\_NotZ, **z** is not referenced.

On exit: if **compz** = Nag\_InitZ, **z** contains the orthogonal matrix *Z*.

If **compz** = Nag\_UpdateZ, **z** is overwritten by  $Z_1Z$ .

14: **pdz** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array **z**.

Constraints:

if **compz** = Nag\_UpdateZ or Nag\_InitZ, **pdz**  $\geq$  max(1, **n**);  
if **compz** = Nag\_NotZ, **pdz**  $\geq$  1.

15: **fail** – NagError \* *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument *⟨value⟩* had an illegal value.

### NE\_ENUM\_INT\_2

On entry, **compq** = *⟨value⟩*, **pdq** = *⟨value⟩* and **n** = *⟨value⟩*.

Constraint: if **compq** = Nag\_InitQ or Nag\_UpdateSchur, **pdq**  $\geq$  max(1, **n**);  
if **compq** = Nag\_NotQ, **pdq**  $\geq$  1.

On entry, **compz** = *⟨value⟩*, **pdz** = *⟨value⟩* and **n** = *⟨value⟩*.

Constraint: if **compz** = Nag\_UpdateZ or Nag\_InitZ, **pdz**  $\geq$  max(1, **n**);  
if **compz** = Nag\_NotZ, **pdz**  $\geq$  1.

### NE\_INT

On entry, **n** = *⟨value⟩*.

Constraint: **n**  $\geq$  0.

On entry, **pda** = *⟨value⟩*.

Constraint: **pda**  $>$  0.

On entry, **pdb** = *⟨value⟩*.

Constraint: **pdb**  $>$  0.

On entry, **pdq** = *⟨value⟩*.

Constraint: **pdq**  $>$  0.

On entry, **pdz** = *⟨value⟩*.

Constraint: **pdz**  $>$  0.

### NE\_INT\_2

On entry, **pda** = *⟨value⟩* and **n** = *⟨value⟩*.

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

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

### NE\_INT\_3

On entry, **n** =  $\langle value \rangle$ , **ilo** =  $\langle value \rangle$  and **ihi** =  $\langle value \rangle$ .  
 Constraint: if **n** > 0,  $1 \leq \mathbf{ilo} \leq \mathbf{ihi} \leq \mathbf{n}$ ;  
 if **n** = 0, **ilo** = 1 and **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 How to Use the NAG Library and its Documentation 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 How to Use the NAG Library and its Documentation for further information.

## 7 Accuracy

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

## 8 Parallelism and Performance

nag\_dgghd3 (f08wfc) 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

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\_zgghd3 (f08wtc).

## 10 Example

See Section 10 in nag\_dhgeqz (f08xec) and nag\_dtgevc (f08ykc).

---