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

```
#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

$$Q^{\mathrm{H}}AZ = H$$
$$Q^{\mathrm{H}}BZ = T$$

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

$$Q_1AZ_1^{\mathrm{H}} = (Q_1Q)H(Z_1Z)^{\mathrm{H}},$$
$$Q_1BZ_1^{\mathrm{H}} = (Q_1Q)T(Z_1Z)^{\mathrm{H}}.$$

4 References

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd 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 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: **compq** – Nag_ComputeQType *Input*

On entry: specifies the form of the computed unitary matrix Q.

compq = Nag_NotQ
Do not compute Q.

compq = Nag_InitQ
The unitary matrix Q is returned.

compq = 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: **compz** – Nag_ComputeZType *Input*

On entry: specifies the form of the computed unitary matrix Z.

compz = Nag_NotZ
Do not compute Z.

compz = Nag_InitZ
The unitary matrix Z is returned.

compz = Nag_UpdateZ
z must contain a unitary matrix Z_1, and the product Z_1Z is returned.

Constraint: **compz** = Nag_NotZ, Nag_InitZ or Nag_UpdateZ.

4: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

5: **ilo** – Integer *Input*
6: **ihi** – Integer *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, $1 \leq$ **ilo** $\leq$ **ihi** $\leq$ **n**;
if **n** $= 0$, **ilo** $= 1$ and **ihi** $= 0$.

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

Note: the dimension, *dim*, of the array **a** must be at least max(1, **pda** × **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_zunmqr (f08auc).

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*] – Complex *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

$\mathbf{b}[(j-1) \times \mathbf{pdb} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{b}[(i-1) \times \mathbf{pdb} + j - 1]$ when **order** = 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_zgeqrf (f08asc).

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*] – Complex *Input/Output*

Note: the dimension, *dim*, of the array **q** must be at least

$\max(1, \mathbf{pdq} \times \mathbf{n})$ when **compq** = Nag_InitQ or Nag_UpdateSchur;
1 when **compq** = Nag_NotQ.

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

$\mathbf{q}[(j-1) \times \mathbf{pdq} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{q}[(i-1) \times \mathbf{pdq} + j - 1]$ when **order** = Nag_RowMajor.

On entry: if **compq** = Nag_UpdateSchur, **q** must contain a unitary matrix Q_1.

If **compq** = Nag_NotQ, **q** is not referenced.

On exit: if **compq** = Nag_InitQ, **q** contains the unitary matrix Q.

Iif **compq** = Nag_UpdateSchur, **q** is overwritten by $Q_1 Q$.

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** ≥ 1.

13: **z**[*dim*] – Complex *Input/Output*

Note: the dimension, *dim*, of the array **z** must be at least

$\max(1, \mathbf{pdz} \times \mathbf{n})$ when **compz** = Nag_InitZ or Nag_UpdateZ;
1 when **compz** = Nag_NotZ.

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

$\mathbf{z}[(j-1) \times \mathbf{pdz} + i - 1]$ when **order** = Nag_ColMajor;
$\mathbf{z}[(i-1) \times \mathbf{pdz} + j - 1]$ when **order** = Nag_RowMajor.

On entry: if **compz** = Nag_UpdateZ, **z** must contain a unitary matrix Z_1.

If **compz** = Nag_NotZ, **z** is not referenced.

On exit: if **compz** = Nag_InitZ, **z** contains the unitary matrix Z.

If **compz** = Nag_UpdateZ, **z** is overwritten by $Z_1 Z$.

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_InitZ or Nag_UpdateZ, **pdz** $\geq \max(1, \mathbf{n})$;
if **compz** = Nag_NotZ, **pdz** ≥ 1.

15: **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.

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** $\geq \max(1, \mathbf{n})$;
if **compq** = Nag_NotQ, **pdq** ≥ 1.

On entry, **compz** = ⟨*value*⟩, **pdz** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **compz** = Nag_InitZ or Nag_UpdateZ, **pdz** $\geq \max(1, \mathbf{n})$;
if **compz** = Nag_NotZ, **pdz** ≥ 1.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 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** $\geq \max(1, \mathbf{n})$.

On entry, **pdb** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
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, $1 \leq$ **ilo** $\leq$ **ihi** $\leq$ **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.

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 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).

10 Example

See Section 10 in nag_zhgeqz (f08xsc) and nag_ztgevc (f08yxc).