NAG C Library Function Document

nag_zgebak (f08nwc)

1 Purpose

nag_zgebak (f08nwc) transforms eigenvectors of a balanced matrix to those of the original complex general matrix.

2 Specification

```
void nag_zgebak (Nag_OrderType order, Nag_JobType job, Nag_SideType side, Integer n,
    Integer ilo, Integer ihi, const double scale[], Integer m, Complex v[],
    Integer pdv, NagError *fail)
```

3 Description

nag_zgebak (f08nwc) is intended to be used after a complex general matrix A has been balanced by nag_zgebal (f08nvc), and eigenvectors of the balanced matrix A''_{22} have subsequently been computed.

For a description of balancing, see the document for nag_zgebal (f08nvc). The balanced matrix A'' is obtained as $A'' = DPAP^TD^{-1}$, where P is a permutation matrix and D is a diagonal scaling matrix. This function transforms left or right eigenvectors as follows:

if x is a right eigenvector of A'', $P^TD^{-1}x$ is a right eigenvector of A;

if y is a left eigenvector of A'', P^TDy is a left eigenvector of A;

4 References

None.

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: **job** – Nag_JobType *Input*

On entry: this **must** be the same parameter **job** as supplied to nag_zgebal (f08nvc).

Constraint: **job** = **Nag_DoNothing**, **Nag_Permute**, **Nag_Scale** or **Nag_DoBoth**.

3: **side** – Nag_SideType *Input*

On entry: indicates whether left or right eigenvectors are to be transformed, as follows:

if **side** = **Nag_LeftSide**, left eigenvectors are transformed;

if **side** = **Nag_RightSide**, right eigenvectors are transformed.

Constraint: **side** = **Nag_LeftSide** or **Nag_RightSide**.

4: **n** – Integer *Input*

On entry: n, the number of rows of the matrix of eigenvectors.

Constraint: **n** ≥ 0.

5: **ilo** – Integer *Input*
6: **ihi** – Integer *Input*

On entry: the values i_{lo} and i_{hi}, as returned by nag_zgebal (f08nvc).

Constraints:

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

7: **scale**$[dim]$ – const double *Input*

Note: the dimension, dim, of the array **scale** must be at least $\max(1, \mathbf{n})$.

On entry: details of the permutations and/or the scaling factors used to balance the original complex general matrix, as returned by nag_zgebal (f08nvc).

8: **m** – Integer *Input*

On entry: m, the number of columns of the matrix of eigenvectors.

Constraint: **m** ≥ 0.

9: **v**$[dim]$ – Complex *Input/Output*

Note: the dimension, dim, of the array **v** must be at least $\max(1, \mathbf{pdv} \times \mathbf{m})$ when **order** $=$ **Nag_ColMajor** and at least $\max(1, \mathbf{pdv} \times \mathbf{n})$ when **order** $=$ **Nag_RowMajor**.

If **order** $=$ **Nag_ColMajor**, the (i, j)th element of the matrix V is stored in **v**$[(j-1) \times \mathbf{pdv} + i - 1]$ and if **order** $=$ **Nag_RowMajor**, the (i, j)th element of the matrix V is stored in **v**$[(i-1) \times \mathbf{pdv} + j - 1]$.

On entry: the matrix of left or right eigenvectors to be transformed.

On exit: the transformed eigenvectors.

10: **pdv** – Integer *Input*

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

Constraints:

if **order** $=$ **Nag_ColMajor**, **pdv** $\geq \max(1, \mathbf{n})$;
if **order** $=$ **Nag_RowMajor**, **pdv** $\geq \max(1, \mathbf{m})$.

11: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **m** = ⟨*value*⟩.
Constraint: **m** ≥ 0.

On entry, **pdv** = ⟨*value*⟩.
Constraint: **pdv** > 0.

NE_INT_2

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

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

NE_INT_3

On entry, **n** = ⟨*value*⟩, **ilo** = ⟨*value*⟩, **ihi** = ⟨*value*⟩.
Constraint: if $\mathbf{n} > 0$, $1 \leq \mathbf{ilo} \leq \mathbf{ihi} \leq \mathbf{n}$;
if $\mathbf{n} = 0$, $\mathbf{ilo} = 1$ and $\mathbf{ihi} = 0$.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ 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 errors are negligible.

8 Further Comments

The total number of real floating-point operations is approximately proportional to nm.

The real analogue of this function is nag_dgebak (f08njc).

9 Example

See Section 9 of the document for nag_zgebal (f08nvc).