

# NAG 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

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

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  $A'' = DPAP^T D^{-1}$ , 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  $A''$ ,  $P^T D^{-1}x$  is a right eigenvector of  $A$ ;

if  $y$  is a left eigenvector of  $A''$ ,  $P^T D y$  is a left eigenvector of  $A$ .

### 4 References

None.

### 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: **job** – Nag\_JobType *Input*  
*On entry:* this **must** be the same argument **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.  
**side** = Nag\_LeftSide  
The left eigenvectors are transformed.

- side** = Nag\_RightSide  
The 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:*  $n \geq 0$ .
- 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  $\max(1, n)$ .  
*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:*  $m \geq 0$ .
- 9: **v**[*dim*] – Complex *Input/Output*  
**Note:** the dimension, *dim*, of the array **v** must be at least  
 $\max(1, pdv \times m)$  when **order** = Nag\_ColMajor;  
 $\max(1, n \times pdv)$  when **order** = Nag\_RowMajor.  
The ( $i, j$ )th element of the matrix  $V$  is stored in  
 $v[(j - 1) \times pdv + i - 1]$  when **order** = Nag\_ColMajor;  
 $v[(i - 1) \times pdv + j - 1]$  when **order** = Nag\_RowMajor.  
*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 row or column elements (depending on the value of **order**) in the array **v**.  
*Constraints:*  
if **order** = Nag\_ColMajor,  $pdv \geq \max(1, n)$ ;  
if **order** = Nag\_RowMajor,  $pdv \geq \max(1, m)$ .
- 11: **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.  
See Section 3.2.1.2 in the Essential Introduction for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_INT

On entry,  $\mathbf{m} = \langle value \rangle$ .  
Constraint:  $\mathbf{m} \geq 0$ .

On entry,  $\mathbf{n} = \langle value \rangle$ .  
Constraint:  $\mathbf{n} \geq 0$ .

On entry,  $\mathbf{pdv} = \langle value \rangle$ .  
Constraint:  $\mathbf{pdv} > 0$ .

### NE\_INT\_2

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

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

### NE\_INT\_3

On entry,  $\mathbf{n} = \langle value \rangle$ ,  $\mathbf{ilo} = \langle value \rangle$  and  $\mathbf{ihi} = \langle value \rangle$ .  
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\_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 the Essential Introduction 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 the Essential Introduction for further information.

## 7 Accuracy

The errors are negligible.

## 8 Parallelism and Performance

nag\_zgebak (f08nwc) is not threaded by NAG in any implementation.

nag\_zgebak (f08nwc) 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

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

The real analogue of this function is nag\_dgebak (f08njc).

## 10 Example

See Section 10 in nag\_zgebal (f08nvc).

---