# NAG Library Function Document

# nag_zggbak (f08wwc)

## 1 Purpose

nag_zggbak (f08wwc) forms the right or left eigenvectors of the real generalized eigenvalue problem $Ax = \lambda Bx$, by backward transformation on the computed eigenvectors given by nag_ztgevc (f08yxc). It is necessary to call this function only if the optional balancing function nag_zggbal (f08wvc) was previously called to balance the matrix pair $(A, B)$.

## 2 Specification

```
#include <nag.h>
#include <nagf08.h>
void nag_zggbak (Nag_OrderType order, Nag_JobType job, Nag_SideType side,
     Integer n, Integer ilo, Integer ihi, const double lscale[],
     const double rscale[], Integer m, Complex v[], Integer pdv,
     NagError *fail)
```

## 3 Description

If the matrix pair has been previously balanced using the function nag_zggbal (f08wvc) then nag_zggbak (f08wwc) backtransforms the eigenvector solution given by nag_ztgevc (f08yxc). This is usually the sixth and last step in the solution of the generalized eigenvalue problem.

For a description of balancing, see the document for nag_zggbal (f08wvc).

## 4 References

Ward R C (1981) Balancing the generalized eigenvalue problem *SIAM J. Sci. Stat. Comp.* **2** 141–152

## 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*: specifies the backtransformation step required.

**job** = Nag_DoNothing
No transformations are done.

**job** = Nag_Permute
Only do backward transformations based on permutations.

**job** = Nag_Scale
Only do backward transformations based on scaling.

**job** = Nag_DoBoth
Do backward transformations for both permutations and scaling.

**Note:** this must be identical to the argument **job** as supplied to nag_dggbal (f08whc).

*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 order of the matrices $A$ and $B$ of the generalized eigenvalue problem.

*Constraint*: $\mathbf{n} \geq 0$.

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

*On entry*: $i_{\mathrm{lo}}$ and $i_{\mathrm{hi}}$ as determined by a previous call to nag_zggbal (f08wvc).

*Constraints*:

if $\mathbf{n} > 0$, $1 \leq \mathbf{ilo} \leq \mathbf{ihi} \leq \mathbf{n}$;
if $\mathbf{n} = 0$, $\mathbf{ilo} = 1$ and $\mathbf{ihi} = 0$.

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

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

*On entry*: details of the permutations and scaling factors applied to the left side of the matrices $A$ and $B$, as returned by a previous call to nag_zggbal (f08wvc).

8: **rscale**[$dim$] – const double *Input*

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

*On entry*: details of the permutations and scaling factors applied to the right side of the matrices $A$ and $B$, as returned by a previous call to nag_zggbal (f08wvc).

9: **m** – Integer *Input*

*On entry*: $m$, the required number of left or right eigenvectors.

*Constraint*: $0 \leq \mathbf{m} \leq \mathbf{n}$.

10: **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;
$\max(1, \mathbf{n} \times \mathbf{pdv})$ when **order** = Nag_RowMajor.

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

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

*On entry*: the matrix of right or left eigenvectors, as returned by nag_zggbal (f08wvc).

*On exit*: the transformed right or left eigenvectors.

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

12: **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_INT**

On entry, **n** = ⟨*value*⟩.
Constraint: **n** $\geq 0$.

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

**NE_INT_2**

On entry, **m** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: $0 \leq \mathbf{m} \leq \mathbf{n}$.

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

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

**NE_INT_3**

On entry, **n** = ⟨*value*⟩, **ilo** = ⟨*value*⟩ and **ihi** = ⟨*value*⟩.
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 errors are negligible.

## 8 Parallelism and Performance

Not applicable.

## 9 Further Comments

The number of operations is proportional to $n^2$.

The real analogue of this function is nag_dggbak (f08wjc).

## 10 Example

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