

# 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:*  $n \geq 0$ .
- 5: **ilo** – Integer *Input*  
 6: **ihi** – Integer *Input*  
*On entry:*  $i_{lo}$  and  $i_{hi}$  as determined by a previous call to nag\_zggbal (f08wvc).  
*Constraints:*  
 if  $n > 0$ ,  $1 \leq ilo \leq ihi \leq n$ ;  
 if  $n = 0$ ,  $ilo = 1$  and  $ihi = 0$ .
- 7: **lscale** $[dim]$  – const double *Input*  
**Note:** the dimension,  $dim$ , of the array **lscale** must be at least  $\max(1, n)$ .  
*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  $\max(1, n)$ .  
*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:*  $0 \leq m \leq n$ .
- 10: **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 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.  
 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, **n** =  $\langle value \rangle$ .  
 Constraint: **n**  $\geq$  0.

On entry, **pdv** =  $\langle value \rangle$ .  
 Constraint: **pdv**  $>$  0.

### NE\_INT\_2

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

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

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

### 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 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\_zggbak (f08wwc) is not threaded by NAG in any implementation.

nag\_zggbak (f08wwc) 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 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).

---