NAG Library Function Document

nag_zggbal (f08wvc)

1 Purpose

nag_zggbal (f08wvc) balances a pair of complex square matrices (A, B) of order n. Balancing usually improves the accuracy of computed generalized eigenvalues and eigenvectors.

2 Specification

```
#include <nag.h>
#include <nagf08.h>
void nag_zggbal (Nag_OrderType order, Nag_JobType job, Integer n,
     Complex a[], Integer pda, Complex b[], Integer pdb, Integer *ilo,
     Integer *ihi, double lscale[], double rscale[], NagError *fail)
```

3 Description

Balancing may reduce the 1-norm of the matrices and improve the accuracy of the computed eigenvalues and eigenvectors in the complex generalized eigenvalue problem

$$Ax = \lambda Bx.$$

nag_zggbal (f08wvc) is usually the first step in the solution of the above generalized eigenvalue problem. Balancing is optional but it is highly recommended.

The term 'balancing' covers two steps, each of which involves similarity transformations on A and B. The function can perform either or both of these steps. Both steps are optional.

1. The function first attempts to permute A and B to block upper triangular form by a similarity transformation:

$$PAP^{\mathrm{T}} = F = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ & F_{22} & F_{23} \\ & & F_{33} \end{pmatrix}$$

$$PBP^{\mathrm{T}} = G = \begin{pmatrix} G_{11} & G_{12} & G_{13} \\ & G_{22} & G_{23} \\ & & G_{33} \end{pmatrix}$$

   where P is a permutation matrix, F_{11}, F_{33}, G_{11} and G_{33} are upper triangular. Then the diagonal elements of the matrix pairs (F_{11}, G_{11}) and (F_{33}, G_{33}) are generalized eigenvalues of (A, B). The rest of the generalized eigenvalues are given by the matrix pair (F_{22}, G_{22}) which are in rows and columns i_{lo} to i_{hi}. Subsequent operations to compute the generalized eigenvalues of (A, B) need only be applied to the matrix pair (F_{22}, G_{22}); this can save a significant amount of work if $i_{\mathrm{lo}} > 1$ and $i_{\mathrm{hi}} < n$. If no suitable permutation exists (as is often the case), the function sets $i_{\mathrm{lo}} = 1$ and $i_{\mathrm{hi}} = n$.

2. The function applies a diagonal similarity transformation to (F, G), to make the rows and columns of (F_{22}, G_{22}) as close in norm as possible:

$$DF\hat{D} = \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ & F_{22} & F_{23} \\ & & F_{33} \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & \hat{D}_{22} & 0 \\ 0 & 0 & I \end{pmatrix}$$

$$DGD^{-1} = \begin{pmatrix} I & 0 & 0 \\ 0 & D_{22} & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} G_{11} & G_{12} & G_{13} \\ & G_{22} & G_{23} \\ & & G_{33} \end{pmatrix} \begin{pmatrix} I & 0 & 0 \\ 0 & \hat{D}_{22} & 0 \\ 0 & 0 & I \end{pmatrix}$$

This transformation usually improves the accuracy of computed generalized eigenvalues and eigenvectors.

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 operations to be performed on matrices A and B.

job = Nag_DoNothing
No balancing is done. Initialize **ilo** = 1, **ihi** = **n**, **lscale**$[i-1] = 1.0$ and **rscale**$[i-1] = 1.0$, for $i = 1, 2, \ldots, n$.

job = Nag_Permute
Only permutations are used in balancing.

job = Nag_Scale
Only scalings are are used in balancing.

job = Nag_DoBoth
Both permutations and scalings are used in balancing.

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

3: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

4: **a**$[dim]$ – Complex *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least $\max(1, \textbf{pda} \times \textbf{n})$.

Where $\mathbf{A}(i, j)$ appears in this document, it refers to the array element

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 n by n matrix A.

On exit: **a** is overwritten by the balanced matrix. If **job** = Nag_DoNothing, **a** is not referenced.

5: **pda** – Integer *Input*

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

Constraint: **pda** $\geq \max(1, \mathbf{n})$.

6: **b**[dim] – Complex *Input/Output*

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

Where $\mathbf{B}(i, j)$ appears in this document, it refers to the array element

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

On entry: the n by n matrix B.

On exit: **b** is overwritten by the balanced matrix. If **job** = Nag_DoNothing, **b** is not referenced.

7: **pdb** – Integer *Input*

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

Constraint: **pdb** $\geq \max(1, \mathbf{n})$.

8: **ilo** – Integer * *Output*
9: **ihi** – Integer * *Output*

On exit: i_{lo} and i_{hi} are set such that $\mathbf{A}(i, j) = 0$ and $\mathbf{B}(i, j) = 0$ if $i > j$ and $1 \leq j < i_{\mathrm{lo}}$ or $i_{\mathrm{hi}} < i \leq n$.

If **job** = Nag_DoNothing or Nag_Scale, $i_{\mathrm{lo}} = 1$ and $i_{\mathrm{hi}} = n$.

10: **lscale**[**n**] – double *Output*

On exit: details of the permutations and scaling factors applied to the left side of the matrices A and B. If P_i is the index of the row interchanged with row i and d_i is the scaling factor applied to row i, then

lscale$[i-1] = P_i$, for $i = 1, 2, \ldots, i_{\mathrm{lo}} - 1$;

lscale$[i-1] = d_i$, for $i = i_{\mathrm{lo}}, \ldots, i_{\mathrm{hi}}$;

lscale$[i-1] = P_i$, for $i = i_{\mathrm{hi}} + 1, \ldots, n$.

The order in which the interchanges are made is n to $i_{\mathrm{hi}} + 1$, then 1 to $i_{\mathrm{lo}} - 1$.

11: **rscale**[**n**] – double *Output*

On exit: details of the permutations and scaling factors applied to the right side of the matrices A and B.

If P_j is the index of the column interchanged with column j and $\hat{d}_j$ is the scaling factor applied to column j, then

rscale$[j-1] = P_j$, for $j = 1, 2, \ldots, i_{\mathrm{lo}} - 1$;

rscale$[j-1] = \hat{d}_j$, for $j = i_{\mathrm{lo}}, \ldots, i_{\mathrm{hi}}$;

rscale$[j-1] = P_j$, for $j = i_{\mathrm{hi}} + 1, \ldots, n$.

The order in which the interchanges are made is n to $i_{\mathrm{hi}} + 1$, then 1 to $i_{\mathrm{lo}} - 1$.

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** ≥ 0.

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

On entry, **pdb** = ⟨*value*⟩.
Constraint: **pdb** > 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_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, compared to those in subsequent computations.

8 Parallelism and Performance

nag_zggbal (f08wvc) is not threaded by NAG in any implementation.

nag_zggbal (f08wvc) 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 Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

nag_zggbal (f08wvc) is usually the first step in computing the complex generalized eigenvalue problem but it is an optional step. The matrix B is reduced to the triangular form using the QR factorization function nag_zgeqrf (f08asc) and the unitary transformation Q is applied to the matrix A by calling nag_zunmqr (f08auc). This is followed by nag_zgghrd (f08wsc) which reduces the matrix pair into the generalized Hessenberg form.

If the matrix pair (A, B) is balanced by this function, then any generalized eigenvectors computed subsequently are eigenvectors of the balanced matrix pair. In that case, to compute the generalized eigenvectors of the original matrix, nag_zggbak (f08wwc) must be called.

The total number of floating-point operations is approximately proportional to n^2.

The real analogue of this function is nag_dggbal (f08whc).

10 Example

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