NAG Library Function Document

nag_dgebak (f08njc)

1 Purpose

nag_dgebak (f08njc) transforms eigenvectors of a balanced matrix to those of the original real nonsymmetric matrix.

2 Specification

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

3 Description

nag_dgebak (f08njc) is intended to be used after a real nonsymmetric matrix A has been balanced by nag_dgebal (f08nhc), and eigenvectors of the balanced matrix A''_{22} have subsequently been computed.

For a description of balancing, see the document for nag_dgebal (f08nhc). The balanced matrix A'' is obtained as $A'' = DPAP^{\mathrm{T}}D^{-1}$, 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'', $P^{\mathrm{T}}D^{-1}x$ is a right eigenvector of A;

if y is a left eigenvector of A'', $P^{\mathrm{T}}Dy$ 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_dgebal (f08nhc).

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

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

On entry: the values i_{lo} and i_{hi}, as returned by nag_dgebal (f08nhc).

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: **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 real nonsymmetric matrix, as returned by nag_dgebal (f08nhc).

8: **m** – Integer *Input*

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

Constraint: $\mathbf{m} \geq 0$.

9: **v**$[dim]$ – double *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 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, $\mathbf{pdv} \geq \max(1, \mathbf{n})$;
if **order** = Nag_RowMajor, $\mathbf{pdv} \geq \max(1, \mathbf{m})$.

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.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_INT

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

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

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

NE_INT_2

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

nag_dgebak (f08njc) is not threaded by NAG in any implementation.

nag_dgebak (f08njc) 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

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

The complex analogue of this function is nag_zgebak (f08nwc).

10 Example

See Section 10 in nag_dgebal (f08nhc).