NAG Library Function Document

nag_dsbevd (f08hcc)

1 Purpose

nag_dsbevd (f08hcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric band matrix. If the eigenvectors are requested, then it uses a divide-and-conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

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

```
void nag_dsbevd (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
     Integer n, Integer kd, double ab[], Integer pdab, double w[],
     double z[], Integer pdz, NagError *fail)
```

3 Description

nag_dsbevd (f08hcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric band matrix A. In other words, it can compute the spectral factorization of A as

$$A = Z\Lambda Z^{\mathrm{T}},$$

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

$$Az_i = \lambda_i z_i, \quad i = 1, 2, \ldots, n.$$

4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users' Guide* (3rd Edition) SIAM, Philadelphia http://www.netlib.org/lapack/lug

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

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: indicates whether eigenvectors are computed.

job = Nag_DoNothing
Only eigenvalues are computed.

job = Nag_EigVecs
Eigenvalues and eigenvectors are computed.

Constraint: **job** = Nag_DoNothing or Nag_EigVecs.

3: **uplo** – Nag_UploType *Input*

On entry: indicates whether the upper or lower triangular part of A is stored.

uplo = Nag_Upper
The upper triangular part of A is stored.

uplo = Nag_Lower
The lower triangular part of A is stored.

Constraint: **uplo** = Nag_Upper or Nag_Lower.

4: **n** – Integer *Input*

On entry: n, the order of the matrix A.

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

5: **kd** – Integer *Input*

On entry: if **uplo** = Nag_Upper, the number of superdiagonals, k_d, of the matrix A.

If **uplo** = Nag_Lower, the number of subdiagonals, k_d, of the matrix A.

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

6: **ab**[*dim*] – double *Input/Output*

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

On entry: the upper or lower triangle of the n by n symmetric band matrix A.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of A_{ij}, depends on the **order** and **uplo** arguments as follows:

if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ab}[k_d + i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \dots, n$ and $i = \max(1, j - k_d), \dots, j$;
if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ab}[i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \dots, n$ and $i = j, \dots, \min(n, j + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ab}[j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \dots, n$ and $j = i, \dots, \min(n, i + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ab}[k_d + j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \dots, n$ and $j = \max(1, i - k_d), \dots, i$.

On exit: **ab** is overwritten by values generated during the reduction to tridiagonal form.

The first superdiagonal or subdiagonal and the diagonal of the tridiagonal matrix T are returned in **ab** using the same storage format as described above.

7: **pdab** – Integer *Input*

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

Constraint: $\mathbf{pdab} \geq \mathbf{kd} + 1$.

8: **w**[dim] – double *Output*

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

On exit: the eigenvalues of the matrix A in ascending order.

9: **z**[dim] – double *Output*

Note: the dimension, *dim*, of the array **z** must be at least

$\max(1, \mathbf{pdz} \times \mathbf{n})$ when **job** = Nag_EigVecs;
1 when **job** = Nag_DoNothing.

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

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

On exit: if **job** = Nag_EigVecs, **z** is overwritten by the orthogonal matrix Z which contains the eigenvectors of A. The ith column of Z contains the eigenvector which corresponds to the eigenvalue $\mathbf{w}[i-1]$.

If **job** = Nag_DoNothing, **z** is not referenced.

10: **pdz** – Integer *Input*

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

Constraints:

if **job** = Nag_EigVecs, $\mathbf{pdz} \geq \max(1, \mathbf{n})$;
if **job** = Nag_DoNothing, $\mathbf{pdz} \geq 1$.

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 ⟨*value*⟩ had an illegal value.

NE_CONVERGENCE

If **fail**.**errnum** = ⟨*value*⟩ and **job** = Nag_DoNothing, the algorithm failed to converge; ⟨*value*⟩ elements of an intermediate tridiagonal form did not converge to zero; if **fail**.**errnum** = ⟨*value*⟩ and **job** = Nag_EigVecs, then the algorithm failed to compute an eigenvalue while working on the submatrix lying in rows and column ⟨*value*⟩/$(\mathbf{n}+1)$ through ⟨*value*⟩ mod $(\mathbf{n}+1)$.

NE_ENUM_INT_2

On entry, **job** = ⟨*value*⟩, **pdz** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **job** = Nag_EigVecs, $\mathbf{pdz} \geq \max(1, \mathbf{n})$;
if **job** = Nag_DoNothing, $\mathbf{pdz} \geq 1$.

NE_INT

On entry, **kd** = ⟨*value*⟩.
Constraint: $\mathbf{kd} \geq 0$.

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 0.

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

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

NE_INT_2

On entry, **pdab** $= \langle value \rangle$ and **kd** $= \langle value \rangle$.
Constraint: **pdab** $\geq$ **kd** $+ 1$.

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 computed eigenvalues and eigenvectors are exact for a nearby matrix $(A + E)$, where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the ***machine precision***. See Section 4.7 of Anderson *et al.* (1999) for further details.

8 Parallelism and Performance

nag_dsbevd (f08hcc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dsbevd (f08hcc) 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 complex analogue of this function is nag_zhbevd (f08hqc).

10 Example

This example computes all the eigenvalues and eigenvectors of the symmetric band matrix A, where

$$A = \begin{pmatrix} 1 & 2 & 3 & 0 & 0 \\ 2 & 2 & 3 & 4 & 0 \\ 3 & 3 & 3 & 4 & 5 \\ 0 & 4 & 4 & 4 & 5 \\ 0 & 0 & 5 & 5 & 5 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_dsbevd (f08hcc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer       i, j, k, kd, n, pdab, pdz, w_len;
  Integer       exit_status = 0;
  NagError      fail;
  Nag_JobType   job;
  Nag_UploType  uplo;
  Nag_OrderType order;
  /* Arrays */
  char          nag_enum_arg[40];
  double        *ab = 0, *w = 0, *z = 0;

#ifdef NAG_COLUMN_MAJOR
#define AB_UPPER(I, J) ab[(J - 1) * pdab + k + I - J - 1]
#define AB_LOWER(I, J) ab[(J - 1) * pdab + I - J]
#define Z(I, J) z[(J - 1) * pdz + I - 1]
  order = Nag_ColMajor;
#else
#define AB_UPPER(I, J) ab[(I - 1) * pdab + J - I]
#define AB_LOWER(I, J) ab[(I - 1) * pdab + k + J - I - 1]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_dsbevd (f08hcc) Example Program Results\n\n");

  /* Skip heading in data file */
#ifdef _WIN32
  scanf_s("%*[^\n] ");
#else
  scanf("%*[^\n] ");
#endif
#ifdef _WIN32
  scanf_s("%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &n, &kd);
#else
  scanf("%"NAG_IFMT"%"NAG_IFMT"%*[^\n] ", &n, &kd);
#endif
  pdab = kd + 1;
  pdz = n;
  w_len = n;

  /* Allocate memory */
  if (!(ab = NAG_ALLOC(pdab * n, double)) ||
      !(w = NAG_ALLOC(w_len, double)) ||
      !(z = NAG_ALLOC(n * n, double)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  /* Read whether Upper or Lower part of A is stored */
#ifdef _WIN32
  scanf_s("%39s%*[^\n] ", nag_enum_arg, _countof(nag_enum_arg));
```

```
#else
  scanf("%39s%*[^\n] ", nag_enum_arg);
#endif
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value
   */
  uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
  /* Read A from data file */
  k = kd + 1;
  if (uplo == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = i; j <= MIN(i + kd, n); ++j)
#ifdef _WIN32
            scanf_s("%lf", &AB_UPPER(i, j));
#else
            scanf("%lf", &AB_UPPER(i, j));
#endif
        }
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }
  else
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = MAX(1, i - kd); j <= i; ++j)
#ifdef _WIN32
            scanf_s("%lf", &AB_LOWER(i, j));
#else
            scanf("%lf", &AB_LOWER(i, j));
#endif
        }
#ifdef _WIN32
      scanf_s("%*[^\n] ");
#else
      scanf("%*[^\n] ");
#endif
    }
  /* Read type of job to be performed */
#ifdef _WIN32
  scanf_s("%39s%*[^\n] ", nag_enum_arg, _countof(nag_enum_arg));
#else
  scanf("%39s%*[^\n] ", nag_enum_arg);
#endif
  job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);
  /* Calculate all the eigenvalues and eigenvectors of A */
  /* nag_dsbevd (f08hcc).
   * All eigenvalues and optionally all eigenvectors of real
   * symmetric band matrix (divide-and-conquer)
   */
  nag_dsbevd(order, job, uplo, n, kd, ab, pdab, w, z, pdz, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dsbevd (f08hcc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Normalize the eigenvectors */
  for(j=1; j<=n; j++)
    {
      for(i=n; i>=1; i--)
        {
          Z(i, j) = Z(i, j) / Z(1,j);
        }
    }
  /* Print eigenvalues and eigenvectors */
  printf(" Eigenvalues\n");
  for (i = 0; i < n; ++i)
    printf(" %8.4lf", w[i]);
  printf("\n\n");
  /* nag_gen_real_mat_print (x04cac).
   * Print real general matrix (easy-to-use)
   */
  fflush(stdout);
  nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
                         z, pdz, "Eigenvectors", 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  NAG_FREE(ab);
  NAG_FREE(w);
  NAG_FREE(z);
  return exit_status;
}
```

10.2 Program Data

```
nag_dsbevd (f08hcc) Example Program Data
  5  2                        :Values of n and kd
  Nag_Lower                   :Value of uplo
  1.0
  2.0  2.0
  3.0  3.0  3.0
       4.0  4.0  4.0
            5.0  5.0  5.0     :End of matrix A
  Nag_EigVecs                 :Value of job
```

10.3 Program Results

```
nag_dsbevd (f08hcc) Example Program Results

 Eigenvalues
  -3.2474  -2.6633   1.7511   4.1599  14.9997

 Eigenvectors
            1          2          3          4          5
 1      1.0000     1.0000     1.0000     1.0000     1.0000
 2     14.5267    -0.4128    -0.6915     1.1530     1.9975
 3    -11.1002    -0.9459     0.7113     0.2847     3.3349
 4    -11.2315     0.6907    -0.9905    -0.0909     3.4904
 5     13.5387     0.1665     0.4296    -1.1530     3.4128
```