

# NAG Library Function Document

## nag\_dsbev (f08hac)

### 1 Purpose

nag\_dsbev (f08hac) computes all the eigenvalues and, optionally, all the eigenvectors of a real  $n$  by  $n$  symmetric band matrix  $A$  of bandwidth  $(2k_d + 1)$ .

### 2 Specification

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

void nag_dsbev (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

The symmetric band matrix  $A$  is first reduced to tridiagonal form, using orthogonal similarity transformations, and then the  $QR$  algorithm is applied to the tridiagonal matrix to compute the eigenvalues and (optionally) the eigenvectors.

### 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\_EigVals  
 Only eigenvalues are computed.  
**job** = Nag\_DoBoth  
 Eigenvalues and eigenvectors are computed.  
*Constraint:* **job** = Nag\_EigVals or Nag\_DoBoth.
- 3: **uplo** – Nag\_UploType *Input*  
*On entry:* if **uplo** = Nag\_Upper, the upper triangular part of  $A$  is stored.  
 If **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:*  $n \geq 0$ .
- 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:*  $kd \geq 0$ .
- 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 **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 **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 **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 **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**[**n**] – double Output  
*On exit:* the eigenvalues 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\_DoBoth;  
 1 otherwise.  
 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\_DoBoth, **z** contains the orthonormal eigenvectors of the matrix  $A$ , with the  $i$ th column of  $Z$  holding the eigenvector associated with **w**[ $i - 1$ ].  
 If **job** = Nag\_EigVals, **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\_DoBoth, **pdz**  $\geq$  max(1, **n**);  
 otherwise **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.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_CONVERGENCE

The algorithm failed to converge;  $\langle value \rangle$  off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

### NE\_ENUM\_INT\_2

On entry, **job** =  $\langle value \rangle$ , **pdz** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .  
 Constraint: if **job** = Nag\_DoBoth, **pdz**  $\geq$  max(1, **n**);  
 otherwise **pdz**  $\geq$  1.

### NE\_INT

On entry, **kd** =  $\langle value \rangle$ .  
 Constraint: **kd**  $\geq$  0.

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

## 7 Accuracy

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

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

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

## 8 Further Comments

The total number of floating point operations is proportional to  $n^3$  if `job = Nag_DoBoth` and is proportional to  $k_d n^2$  otherwise.

The complex analogue of this function is `nag_zhbev (f08hnc)`.

## 9 Example

This example finds all the eigenvalues and eigenvectors of the symmetric band matrix

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

together with approximate error bounds for the computed eigenvalues and eigenvectors.

### 9.1 Program Text

```

/* nag_dsbev (f08hac) Example Program.
 *
 * Copyright 2011 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <math.h>
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx02.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double      eerrbd, eps;
    Integer     exit_status = 0, i, j, kd, n, pdab, pdz;
    /* Arrays */
    char        nag_enum_arg[40];
    double      *ab = 0, *rcondz = 0, *w = 0, *z = 0, *zerrbd = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_UploType  uplo;
    NagError      fail;

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

    INIT_FAIL(fail);

    printf("nag_dsbev (f08hac) Example Program Results\n\n");

    /* Skip heading in data file */
    scanf("%*[\n]");

```

```

scanf("%ld%ld%*[\n]", &n, &kd);

/* Read uplo */
scanf("%s%*[\n]", nag_enum_arg);
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

/* Allocate memory */
if (!(ab = NAG_ALLOC((kd+1)*n, double)) ||
    !(rcondz = NAG_ALLOC(n, double)) ||
    !(w = NAG_ALLOC(n, double)) ||
    !(z = NAG_ALLOC(n*n, double)) ||
    !(zerrbd = NAG_ALLOC(n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

pdab = kd+1;
pdz = n;

/* Read the upper or lower triangular part of the symmetric band
 * matrix A from data file.
 */
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= MIN(n, i + kd); ++j)
            scanf("%lf", &AB_UPPER(i, j));
    scanf("%*[\n]");
}
else if (uplo == Nag_Lower) {
    for (i = 1; i <= n; ++i)
        for (j = MAX(1, i - kd); j <= i; ++j)
            scanf("%lf", &AB_LOWER(i, j));
    scanf("%*[\n]");
}

/* nag_dsbev (f08hac).
 * Solve the band symmetric eigenvalue problem.
 */
nag_dsbev(order, Nag_DoBoth, uplo, n, kd, ab, pdab, w, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsbev (f08hac).\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 solution */
printf("Eigenvalues\n");
for (j = 0; j < n; ++j)
    printf("%8.4f%s", w[j], (j+1)%8 == 0?"\n":" ");
printf("\n");

/* nag_gen_real_mat_print (x04cac).
 * Print eigenvectors.
 */
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;
}

/* Get the machine precision, eps, using nag_machine_precision (X02AJC)
 * and compute the approximate error bound for the computed eigenvalues.
 * Note that for the 2-norm, ||A|| = max {|w[i]|, i=0..n-1}, and since
 * the eigenvalues are in ascending order ||A|| = max( |w[0]|, |w[n-1]|).
 */
eps = nag_machine_precision;
eerrbd = eps * MAX(fabs(w[0]), fabs(w[n-1]));

/* nag_ddisna (f08flc).
 * Estimate reciprocal condition numbers for eigenvectors.
 */
nag_ddisna(Nag_EigVecs, n, n, w, rcondz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ddisna (f08flc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute the error estimates for the eigenvectors. */
for (i = 0; i < n; ++i)
    zerrbd[i] = eerrbd / rcondz[i];

/* Print the approximate error bounds for the eigenvalues and vectors. */
printf("\nError estimate for the eigenvalues\n");
printf("%11.1e\n", eerrbd);

printf("\nError estimates for the eigenvectors\n");
for (i = 0; i < n; ++i)
    printf("%11.1e%s", zerrbd[i], (i+1)%6 == 0 ? "\n": " ");

END:
if (ab) NAG_FREE(ab);
if (rcondz) NAG_FREE(rcondz);
if (w) NAG_FREE(w);
if (z) NAG_FREE(z);
if (zerrbd) NAG_FREE(zerrbd);

return exit_status;
}

#undef AB_UPPER
#undef AB_LOWER
#undef Z

```

## 9.2 Program Data

nag\_dsbev (f08hac) Example Program Data

```

5      2                      :Values of n and kd
Nag_Upper                      :Value of uplo

1.0  2.0  3.0
      2.0  3.0  4.0
           3.0  4.0  5.0
                4.0  5.0
                     5.0 :End of matrix A

```

### 9.3 Program Results

nag\_dsbev (f08hac) 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

Error estimate for the eigenvalues

1.7e-15

Error estimates for the eigenvectors

2.9e-15 2.9e-15 6.9e-16 6.9e-16 1.5e-16

---