NAG C 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

```c
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

\[
A = Z \Lambda Z^T,
\]

where \( \Lambda \) is a diagonal matrix whose diagonal elements are the eigenvalues \( \lambda_i \), and \( Z \) is the orthogonal matrix whose columns are the eigenvectors \( z_i \). Thus

\[
A z_i = \lambda_i z_i, \quad i = 1, 2, \ldots, n.
\]

4 References


5 Parameters

1: order – Nag_OrderType

*Input*

*On entry:* the `order` parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

*Constraint:* `order = Nag_RowMajor` or `Nag_ColMajor`.

2: job – Nag_JobType

*Input*

*On entry:* indicates whether eigenvectors are computed as follows:

- if `job = Nag_DoNothing`, only eigenvalues are computed;
- if `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 as follows:
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
On entry: n, the order of the matrix A.
Constraint: n ≥ 0.

5: kd – Integer
On entry: k, the number of super-diagonals of the matrix A if uplo = Nag_Upper, or the number of
sub-diagonals if uplo = Nag_Lower.
Constraint: kd ≥ 0.

6: ab[dim] – double
Input/Output
Note: the dimension, dim, of the array ab must be at least max(1, pdab × n).
On entry: 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. Just the upper or lower triangular part
of the array is held depending on the value of uplo. The storage of elements a_ij depends on the
order and uplo parameters as follows:

if order = Nag_ColMajor and uplo = Nag_Upper,
a_{ij} is stored in ab[k + i − j + (j − 1) × pdab], for i = 1, . . . , n and
j = i, . . . , min(n, i + k);
if order = Nag_ColMajor and uplo = Nag_Lower,
a_{ij} is stored in ab[i − j + (j − 1) × pdab], for i = 1, . . . , n and
j = max(1, i − k), . . . , i;
if order = Nag_RowMajor and uplo = Nag_Upper,
a_{ij} is stored in ab[j − i + (i − 1) × pdab], for i = 1, . . . , n and
j = i, . . . , min(n, i + k);
if order = Nag_RowMajor and uplo = Nag_Lower,
a_{ij} is stored in ab[k + j − i + (i − 1) × pdab], for i = 1, . . . , n and
j = max(1, i − k), . . . , i.

On exit: A is overwritten by the values generated during the reduction to tridiagonal form. The
storage details depend on the input values of the parameters order and uplo.

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: pdab ≥ kd + 1.

8: w[dim] – double
Output
Note: the dimension, dim, of the array w must be at least max(1, 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, pdz × n) when job = Nag_EigVecs;
1 when job = Nag_DoNothing.
If \( \text{order} = \text{Nag}_\text{ColMajor} \), the \((i, j)\)th element of the matrix \( Z \) is stored in \( z[j-1] \times pdz + i - 1 \) and if \( \text{order} = \text{Nag}_\text{RowMajor} \), the \((i, j)\)th element of the matrix \( Z \) is stored in \( z[(i-1) \times pdz + j - 1] \).

On exit: if \( \text{job} = \text{Nag}_\text{EigVecs} \), \( z \) is overwritten by the orthogonal matrix \( Z \) which contains the eigenvectors of \( A \). The \( r \)th column of \( Z \) contains the eigenvector which corresponds to the eigenvalue \( w[i] \).

If \( \text{job} = \text{Nag}_\text{DoNothing} \), \( z \) is not referenced.

10: \( \text{pdz} \) – Integer \hspace{1cm} Input

On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( z \).

Constraints:

\[
\begin{align*}
\text{if } \text{job} = \text{Nag}_\text{EigVecs}, \text{ pdz } &\geq \max(1, n) ; \\
\text{if } \text{job} = \text{Nag}_\text{DoNothing}, \text{ pdz } &\geq 1 .
\end{align*}
\]

11: \( \text{fail} \) – NagError * \hspace{1cm} Output

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).

On entry, \( kd = \langle \text{value} \rangle \).
Constraint: \( kd \geq 0 \).

On entry, \( pdab = \langle \text{value} \rangle \).
Constraint: \( pdab > 0 \).

On entry, \( pdz = \langle \text{value} \rangle \).
Constraint: \( pdz > 0 \).

NE_INT_2

On entry, \( pdab = \langle \text{value} \rangle , \text{ kd } = \langle \text{value} \rangle \).
Constraint: \( pdab \geq kd + 1 \).

NE_ENUM_INT_2

On entry, \( \text{job} = \langle \text{value} \rangle , \text{ n } = \langle \text{value} \rangle , \text{ pdz } = \langle \text{value} \rangle \).
Constraint: if \( \text{job} = \text{Nag}_\text{EigVecs} \), \( \text{ pdz } \geq \max(1, n) \); if \( \text{job} = \text{Nag}_\text{DoNothing} \), \( \text{ pdz } \geq 1 \).

NE_CONVERGENCE

The algorithm failed to converge, \( \langle \text{value} \rangle \) elements of an intermediate tridiagonal form did not converge to zero.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.
An internal error has occurred in this function. Check the function call and any array sizes. If the call is correct then please consult 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.

8 Further Comments

The complex analogue of this function is nag_zhbevd (f08hqc).

9 Example

To compute all the eigenvalues and eigenvectors of the symmetric band matrix $A$, where

$$A = \begin{pmatrix}
1.0 & 2.0 & 3.0 & 0.0 & 0.0 \\
2.0 & 2.0 & 3.0 & 4.0 & 0.0 \\
3.0 & 3.0 & 3.0 & 4.0 & 5.0 \\
0.0 & 4.0 & 4.0 & 4.0 & 5.0 \\
0.0 & 0.0 & 5.0 & 5.0 & 5.0
\end{pmatrix},$$

9.1 Program Text

```c
/* nag_dsbevd (f08hcc) Example Program. */
/* Copyright 2001 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 uplo_char[2], job_char[2];
    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]
    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]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
```
Vprintf("f08hcc Example Program Results\n\n");
/* Skip heading in data file */
Vscanf("%*[\n] ");
Vscanf("%ld%ld%*[\n] ", &n, &kd);
pdag = 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)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read whether Upper or Lower part of A is stored */
Vscanf("' %1s ' *[\n] ", uplo_char);
if (*(unsigned char *)uplo_char == 'L')
    uplo = Nag_Lower;
else if (*(unsigned char *)uplo_char == 'U')
    uplo = Nag_Upper;
else
{
    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
}
/* 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)
            Vscanf("%lf", &AB_UPPER(i,j));
    }
    Vscanf("%*[\n] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = MAX(1,i-kd); j <= i; ++j)
            Vscanf("%lf", &AB_LOWER(i,j));
    }
    Vscanf("%*[\n] ");
}
/* Read type of job to be performed */
Vscanf("' %ls ' *[\n] ", job_char);
if (*(unsigned char *)job_char == 'V')
    job = Nag_EigVecs;
else
    job = Nag_DoNothing;
/* Calculate all the eigenvalues and eigenvectors of A */
f08hcc(order, job, uplo, n, kd, ab, pdab, w, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08hcc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print eigenvalues and eigenvectors */
Vprintf(" Eigenvalues\n");
for (i = 0; i < n; ++i)
    Vprintf(" %8.4lf", w[i]);
Vprintf("\n\n");
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
z, pdz, "Eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\n\n", fail.message);
    exit_status = 1;
    goto END;
}

END:
if (ab) NAG_FREE(ab);
if (w) NAG_FREE(w);
if (z) NAG_FREE(z);
return exit_status;

9.2 Program Data

f08hcc Example Program Data
5  2 :Values of N and KD
'L' :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
'V' :Value of JOB

9.3 Program Results

f08hcc Example Program Results

Eigenvalues
-3.2474 -2.6633  1.7511  4.1599  14.9997

Eigenvectors
1  2  3  4  5
1  0.0394 -0.6238 -0.5635  0.5165  0.1582
2  0.5721  0.2575  0.3896  0.5955  0.3161
3 -0.4372  0.5900 -0.4008  0.1470  0.5277
4 -0.4424 -0.4308  0.5581 -0.0470  0.5523
5  0.5332 -0.1039 -0.2421 -0.5956  0.5400