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

```c
#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

$$A = Z \Lambda Z^T,$$

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

$$Az_i = \lambda_i z_i, \quad i = 1, 2, \ldots, n.$$
Eigenvalues and eigenvectors are computed.

**Constraint:** \( \text{job} = \text{Nag\_DoNothing} \) or \( \text{Nag\_EigVecs} \).

3: \( \text{uplo} = \text{Nag\_UploType} \)

*Input*

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

- \( \text{uplo} = \text{Nag\_Upper} \)
  
  The upper triangular part of \( A \) is stored.

- \( \text{uplo} = \text{Nag\_Lower} \)
  
  The lower triangular part of \( A \) is stored.

**Constraint:** \( \text{uplo} = \text{Nag\_Upper} \) or \( \text{Nag\_Lower} \).

4: \( n \) \( -\) **Integer**

*Input*

*On entry:* \( n \), the order of the matrix \( A \).

**Constraint:** \( n \geq 0 \).

5: \( kd \) \( -\) **Integer**

*Input*

*On entry:* if \( \text{uplo} = \text{Nag\_Upper} \), the number of superdiagonals, \( k_d \), of the matrix \( A \).

If \( \text{uplo} = \text{Nag\_Lower} \), 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 \( \max(1, pdab \times n) \).

*On entry:* the upper or lower triangle of the \( n \) by \( n \) symmetric band matrix \( A \) 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 \( \text{order} \) and \( \text{uplo} \) arguments as follows:

- \( \text{order} = \text{Nag\_ColMajor} \) and \( \text{uplo} = \text{Nag\_Upper} \),
  
  \( A_{ij} \) is stored in \( ab[k_d + i - j + (j - 1) \times pdab] \), for \( j = 1, \ldots, n \) and \( i = \max(1, j - k_d), \ldots, j \);

- \( \text{order} = \text{Nag\_ColMajor} \) and \( \text{uplo} = \text{Nag\_Lower} \),
  
  \( A_{ij} \) is stored in \( ab[i - j + (j - 1) \times pdab] \), for \( j = 1, \ldots, n \) and \( i = j, \ldots, \min(n, j + k_d) \);

- \( \text{order} = \text{Nag\_RowMajor} \) and \( \text{uplo} = \text{Nag\_Upper} \),
  
  \( A_{ij} \) is stored in \( ab[j - i + (i - 1) \times pdab] \), for \( i = 1, \ldots, n \) and \( j = i, \ldots, \min(n, i + k_d) \);

- \( \text{order} = \text{Nag\_RowMajor} \) and \( \text{uplo} = \text{Nag\_Lower} \),
  
  \( A_{ij} \) is stored in \( ab[k_d + j - i + (i - 1) \times pdab] \), for \( i = 1, \ldots, n \) and \( j = \max(1, i - k_d), \ldots, i \).

*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 \( \text{order} \)) of the matrix \( A \) in the array \( ab \).

**Constraint:** \( pdab \geq kd + 1 \).
8: \( \mathbf{w}[\text{dim}] \) – double  
   \textbf{Output}  
   \textbf{Note}: the dimension, \( \text{dim} \), of the array \( \mathbf{w} \) must be at least \( \max(1, n) \).
   \textit{On exit}: the eigenvalues of the matrix \( A \) in ascending order.

9: \( \mathbf{z}[\text{dim}] \) – double  
   \textbf{Output}  
   \textbf{Note}: the dimension, \( \text{dim} \), of the array \( \mathbf{z} \) must be at least 
   \[ \max(1, \text{pdz} \times n) \] when \( \text{job} = \text{Nag_EigVecs} \); 
   \[ 1 \] when \( \text{job} = \text{Nag_DoNothing} \).

   The \( (i, j) \)th element of the matrix \( \mathbf{Z} \) is stored in 
   \[ \mathbf{z}[(j - 1) \times \text{pdz} + i - 1] \] when \( \text{order} = \text{Nag_ColMajor} \); 
   \[ \mathbf{z}[(i - 1) \times \text{pdz} + j - 1] \] when \( \text{order} = \text{Nag_RowMajor} \).

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

   If \( \text{job} = \text{Nag_DoNothing} \), \( \mathbf{z} \) is not referenced.

10: \text{pdz} – Integer  
    \textbf{Input}  
    \textit{On entry}: the stride separating row or column elements (depending on the value of \text{order}) in the 
    array \( \mathbf{z} \).

    \textbf{Constraints}:
    \begin{align*}
    \text{if } \text{job} &= \text{Nag_EigVecs}, \quad \text{pdz} \geq \max(1, n); \\
    \text{if } \text{job} &= \text{Nag_DoNothing}, \quad \text{pdz} \geq 1.
    \end{align*}

11: \text{fail} – NagError *  
    \textbf{Input/Output}  
    The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_ALLOC_FAIL}  
Dynamic memory allocation failed.
See Section 3.2.1.2 in the Essential Introduction for further information.

\textbf{NE_BAD_PARAM}  
On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

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

\textbf{NE_ENUM_INT_2}  
On entry, \( \text{job} = \langle \text{value} \rangle \), \( \text{pdz} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: if \( \text{job} = \text{Nag_EigVecs} \), \( \text{pdz} \geq \max(1, n) \); 
if \( \text{job} = \text{Nag_DoNothing} \), \( \text{pdz} \geq 1 \).

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

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, \( pdab = \langle \text{value} \rangle \) and \( kd = \langle \text{value} \rangle \).
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 \( \epsilon \) 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

\[
A = \begin{pmatrix}
1 & 2 & 3 & 0 & 0 \\
2 & 2 & 3 & 4 & 0 \\
3 & 3 & 3 & 4 & 5 \\
0 & 4 & 4 & 5 & 5 \\
0 & 0 & 5 & 5 & 5
\end{pmatrix}.
\]
10.1 Program Text

/* nag_dsbevd (f08hcc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 */
#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));
    #endif

    return 0;
}

Mark 25

f08hcc.5
```c
#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
    }
#endif
#ifdef _WIN32
    scanf_s("%*\n ");
#else
    scanf("%*\n ");
#endif
} /* Read type of job to be performed */
#else _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)
 */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsbevd (f08hcc).\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_dsbvdx (f08hcc) Example Program Data
5 2 :Values of n and kd
Nag_Lower :Value of uplo
1.0
2.0
3.0
4.0
5.0
2.0
3.0
4.0
5.0
3.0
4.0
5.0
5.0 :End of matrix A
Nag_EigVecs :Value of job

10.3 Program Results

nag_dsbvdx (f08hcc) Example Program Results

Eigenvalues
-3.2474 -2.6633 1.7511 4.1599 14.9997

Eigenvalues
-3.2474 -2.6633 1.7511 4.1599 14.9997

Eigenvectors

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