NAG Library Function Document
nag_dsbevx (f08hbc)

1 Purpose
nag_dsbevx (f08hbc) computes selected eigenvalues and, optionally, eigenvectors of a real $n$ by $n$ symmetric band matrix $A$ of bandwidth $(2k_d + 1)$. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

2 Specification

```c
#include <nag.h>
#include <nagf08.h>
void nag_dsbevx (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
    Nag_UploType uplo, Integer n, Integer kd, double ab[], Integer pdab,
    double q[], Integer pdq, double vl, double vu, Integer il, Integer iu,
    double abstol, Integer *m, double w[], double z[], Integer pdz,
    Integer jfail[], NagError *fail)
```

3 Description

The symmetric band matrix $A$ is first reduced to tridiagonal form, using orthogonal similarity transformations. The required eigenvalues and eigenvectors are then computed from the tridiagonal matrix; the method used depends upon whether all, or selected, eigenvalues and eigenvectors are required.

4 References


5 Arguments

1: order – Nag_OrderType

   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

   On entry: indicates whether eigenvectors are computed.

   job = Nag_EigVals
   Only eigenvalues are computed.
Eigenvalues and eigenvectors are computed.

Constraint: $job = \text{Nag\_EigVals}$ or $\text{Nag\_DoBoth}.$

3: $range \rightarrow \text{Nag\_RangeType}$  
\text{Input}

On entry: if $range = \text{Nag\_AllValues},$ all eigenvalues will be found.
If $range = \text{Nag\_Interval},$ all eigenvalues in the half-open interval $[vl, vu)$ will be found.
If $range = \text{Nag\_Indices},$ the $i_{th}$ to $u_{th}$ eigenvalues will be found.

Constraint: $range = \text{Nag\_AllValues}, \text{Nag\_Interval}$ or $\text{Nag\_Indices}.$

4: $uplo \rightarrow \text{Nag\_UploType}$  
\text{Input}

On entry: if $uplo = \text{Nag\_Upper},$ the upper triangular part of $A$ is stored.
If $uplo = \text{Nag\_Lower},$ the lower triangular part of $A$ is stored.

Constraint: $uplo = \text{Nag\_Upper}$ or $\text{Nag\_Lower}.$

5: $n \rightarrow \text{Integer}$  
\text{Input}

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

Constraint: $n \geq 0.$

6: $kd \rightarrow \text{Integer}$  
\text{Input}

On entry: if $uplo = \text{Nag\_Upper},$ the number of superdiagonals, $k_d,$ of the matrix $A.$
If $uplo = \text{Nag\_Lower},$ the number of subdiagonals, $k_d,$ of the matrix $A.$

Constraint: $kd \geq 0.$

7: $ab[dim] \rightarrow \text{double}$  
\text{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 $A.$

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

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

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

if $order = \text{Nag\_RowMajor}$ and $uplo = \text{Nag\_Lower},$
$A_{ij}$ 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.
8: \( \text{pdab} \) – Integer
   \textit{Input}
   
   On entry: the stride separating row or column elements (depending on the value of \texttt{order}) of the matrix \( A \) in the array \( ab \).
   
   Constraint: \( \text{pdab} \geq kd + 1 \).

9: \( q[\text{dim}] \) – double
   \textit{Output}
   
   Note: the dimension, \( \text{dim} \), of the array \( q \) must be at least
   
   \[
   \max(1, \text{pdq} \times n) \quad \text{when} \quad \text{job} = \text{Nag}_\text{DoBoth};
   \]
   
   \[
   1 \quad \text{otherwise}.
   \]
   
   The \( (i, j) \)th element of the matrix \( Q \) is stored in
   
   \[
   q[(j - 1) \times \text{pdq} + i - 1] \quad \text{when} \quad \text{order} = \text{Nag}_\text{ColMajor};
   \]
   
   \[
   q[(i - 1) \times \text{pdq} + j - 1] \quad \text{when} \quad \text{order} = \text{Nag}_\text{RowMajor}.
   \]
   
   On exit: if \( \text{job} = \text{Nag}_\text{DoBoth} \), the \( n \) by \( n \) orthogonal matrix used in the reduction to tridiagonal form.
   
   If \( \text{job} = \text{Nag}_\text{EigVals} \), \( q \) is not referenced.

10: \( \text{pdq} \) – Integer
    \textit{Input}
    
    On entry: the stride separating row or column elements (depending on the value of \texttt{order}) in the array \( q \).
    
    Constraints:
    
    \[
    \begin{align*}
    &\text{if} \quad \text{job} = \text{Nag}_\text{DoBoth}, \quad \text{pdq} \geq \max(1, n); \\
    &\text{otherwise} \quad \text{pdq} \geq 1.
    \end{align*}
    \]

11: \( \text{vl} \) – double
    \textit{Input}
    
    On entry: if \texttt{range} = \texttt{Nag}_\texttt{Interval}, the lower and upper bounds of the interval to be searched for eigenvalues.
    
    If \( \texttt{range} = \texttt{Nag}_\texttt{AllValues} \) or \( \texttt{Nag}_\texttt{Indices} \), \( \text{vl} \) and \( \text{vu} \) are not referenced.
    
    Constraint: if \( \texttt{range} = \texttt{Nag}_\texttt{Interval} \), \( \text{vl} < \text{vu} \).

12: \( \text{vu} \) – double
    \textit{Input}
    
    On entry: if \texttt{range} = \texttt{Nag}_\texttt{Interval}, the lower and upper bounds of the interval to be searched for eigenvalues.
    
    If \( \texttt{range} = \texttt{Nag}_\texttt{AllValues} \) or \( \texttt{Nag}_\texttt{Indices} \), \( \text{vl} \) and \( \text{vu} \) are not referenced.
    
    Constraint: if \( \texttt{range} = \texttt{Nag}_\texttt{Interval} \), \( \text{vl} < \text{vu} \).

13: \( \text{il} \) – Integer
    \textit{Input}
    
    On entry: if \texttt{range} = \texttt{Nag}_\texttt{Indices}, the indices (in ascending order) of the smallest and largest eigenvalues to be returned.
    
    If \( \texttt{range} = \texttt{Nag}_\texttt{AllValues} \) or \( \texttt{Nag}_\texttt{Interval} \), \( \text{il} \) and \( \text{iu} \) are not referenced.
    
    Constraints:
    
    \[
    \begin{align*}
    &\text{if} \quad \texttt{range} = \texttt{Nag}_\texttt{Indices} \text{ and } n = 0, \quad \text{il} = 1 \quad \text{and} \quad \text{iu} = 0; \\
    &\text{if} \quad \texttt{range} = \texttt{Nag}_\texttt{Indices} \text{ and } n > 0, \quad 1 \leq \text{il} \leq \text{iu} \leq n.
    \end{align*}
    \]

14: \( \text{iu} \) – Integer
    \textit{Input}
    
    On entry: if \texttt{range} = \texttt{Nag}_\texttt{Indices}, the indices (in ascending order) of the smallest and largest eigenvalues to be returned.
    
    If \( \texttt{range} = \texttt{Nag}_\texttt{AllValues} \) or \( \texttt{Nag}_\texttt{Interval} \), \( \text{il} \) and \( \text{iu} \) are not referenced.
    
    Constraints:
    
    \[
    \begin{align*}
    &\text{if} \quad \texttt{range} = \texttt{Nag}_\texttt{Indices} \text{ and } n = 0, \quad \text{il} = 1 \quad \text{and} \quad \text{iu} = 0; \\
    &\text{if} \quad \texttt{range} = \texttt{Nag}_\texttt{Indices} \text{ and } n > 0, \quad 1 \leq \text{il} \leq \text{iu} \leq n.
    \end{align*}
    \]

15: \( \text{abstol} \) – double
    \textit{Input}
    
    On entry: the absolute error tolerance for the eigenvalues. An approximate eigenvalue is accepted as converged when it is determined to lie in an interval \( [a, b] \) of width less than or equal to
    
    \[
    \text{abstol} + \epsilon \max(|a|, |b|),
    \]
    
    where \( \epsilon \) is the \textit{machine precision}. If \( \text{abstol} \) is less than or equal to zero, then \( \epsilon \|T\|_1 \) will be used in its place, where \( T \) is the tridiagonal matrix obtained by reducing \( A \) to tridiagonal form.
    
    Eigenvalues will be computed most accurately when \( \text{abstol} \) is set to twice the underflow threshold \( 2 \times \text{nag}_\text{real}_\text{safe}_\text{small}_\text{number} \), not zero. If this function returns with \( \text{fail} = \text{code} = \)
NE_CONVERGENCE, indicating that some eigenvectors did not converge, try setting abstol to $2 \times \text{nag\_real\_safe\_small\_number}$. See Demmel and Kahan (1990).

16: $m$ – Integer *  
Output

On exit: the total number of eigenvalues found. $0 \leq m \leq n$.

If range = Nag_AllValues, $m = n$.
If range = Nag_Indices, $m = iu - il + 1$.

17: $w[n]$ – double  
Output

On exit: the first $m$ elements contain the selected eigenvalues in ascending order.

18: $z[dim]$ – double  
Output

Note: the dimension, $dim$, of the array $z$ must be at least
$$\text{max}(1, \text{pdz} \times n) \text{ when } \text{job} = \text{Nag\_DoBoth};$$
1 otherwise.

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

On exit: if job = Nag_DoBoth, then
if fail.code = NE_NOERROR, the first $m$ columns of $Z$ contain the orthonormal eigenvectors of the matrix $A$ corresponding to the selected eigenvalues, with the $i$th column of $Z$ holding the eigenvector associated with $w[i - 1]$;
if an eigenvector fails to converge (fail.code = NE_CONVERGENCE), then that column of $Z$ contains the latest approximation to the eigenvector, and the index of the eigenvector is returned in $jfail$.

If job = Nag_EigVals, $z$ is not referenced.

19: 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 $\geq \text{max}(1, n)$;
otherwise pdz $\geq 1$.

20: jfail[dim] – Integer  
Output

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

On exit: if job = Nag_DoBoth, then
if fail.code = NE_NOERROR, the first $m$ elements of jfail are zero;
if fail.code = NE_CONVERGENCE, jfail contains the indices of the eigenvectors that failed to converge.

If job = Nag_EigVals, jfail is not referenced.

21: 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 \(\text{value}\) had an illegal value.

**NE_CONVERGENCE**
The algorithm failed to converge; \(\text{value}\) eigenvectors did not converge. Their indices are stored in array jfail.

**NE_ENUM_INT_2**
On entry, job = \(\text{value}\), pdq = \(\text{value}\) and n = \(\text{value}\).
Constraint: if job = Nag_DoBoth, pdq \(\geq\) \(\max(1, n)\);
otherwise pdq \(\geq\) 1.

On entry, job = \(\text{value}\), pdz = \(\text{value}\) and n = \(\text{value}\).
Constraint: if job = Nag_DoBoth, pdz \(\geq\) \(\max(1, n)\);
otherwise pdz \(\geq\) 1.

**NE_ENUM_INT_3**
On entry, range = \(\text{value}\), il = \(\text{value}\), iu = \(\text{value}\) and n = \(\text{value}\).
Constraint: if range = Nag_Indices and n = 0, il = 1 and iu = 0;
if range = Nag_Indices and n > 0, 1 \(\leq\) il \(\leq\) iu \(\leq\) n.

**NE_ENUM_REAL_2**
On entry, range = \(\text{value}\), vl = \(\text{value}\) and vu = \(\text{value}\).
Constraint: if range = Nag_Interval, vl < vu.

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

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

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

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

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

**NE_INT_2**
On entry, pdab = \(\text{value}\) and kd = \(\text{value}\).
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.
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 \textit{machine precision}. See Section 4.7 of Anderson \textit{et al.} (1999) for further details.

8  Parallelism and Performance

\texttt{nag_dsbevx (f08hbc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\texttt{nag_dsbevx (f08hbc)} 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 total number of floating-point operations is proportional to \( k_d n^2 \) if \( \text{job} = \text{Nag EigVals} \), and is proportional to \( n^3 \) if \( \text{job} = \text{Nag_DoBoth} \) and \( \text{range} = \text{Nag_AllValues} \), otherwise the number of floating-point operations will depend upon the number of computed eigenvectors.

The complex analogue of this function is \texttt{nag_zhbevx (f08hpc)}.

10  Example

This example finds the eigenvalues in the half-open interval \((-3,3]\), and the corresponding eigenvectors, of the symmetric band matrix
\[
A = \begin{pmatrix}
1 & 2 & 3 & 0 & 0 \\
2 & 2 & 3 & 4 & 0 \\
3 & 3 & 3 & 4 & 5 \\
0 & 4 & 4 & 4 & 5 \\
0 & 0 & 5 & 5 & 5
\end{pmatrix}.
\]

10.1  Program Text

/* nag_dsbevx (f08hbc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* * Mark 23, 2011. */
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double abstol, vl, vu;

Integer exit_status = 0, i, il = 0, iu = 0, j, kd, m, n, pdab, pdq, pdz;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 0, *q = 0, *w = 0, *z = 0;
    Integer *index = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_UploType uplo;
    NagError fail, fail_print;

#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]
#else
#define AB_UPPER(I, J) ab[(I - 1) * pdab + J - I]
#define AB_LOWER(I, J) ab[(I - 1) * pdab + kd + J - I]
#endif
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]
#endif

INIT_FAIL(fail);
printf("nag_dsbevx (f08hbc) Example Program Results\n\n");

/* Skip heading in data file */
#ifdef _WIN32
scanf_s("%*[\n]");
#else
scanf("%*[\n]");
#endif
#ifdef _WIN32
scanf_s("%lf%lf%*\n", &vl, &vu);
#else
scanf("%lf%lf%*[\n]", &vl, &vu);
#endif

uplo = Nag_RowMajor;

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

pdab = kd+1;
pdq = n;
pdz = n;

/* Read the lower and upper bounds of the interval to be searched,*/
/* and read the upper or lower triangular part of the matrix A */
/* from data file. */
#ifdef _WIN32
scanf_s("%lf%lf%*[\n]", &vl, &vu);
#else
scanf("%*\n");
#endif

Mark 25
scanf("%lf%lf%*[\n]", &vl, &vu);
#endif
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= MIN(n, i + kd); ++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 if (uplo == Nag_Lower) {
    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
} /* Set the absolute error tolerance for eigenvalues.
    * With abstol set to zero, the default value is used instead.
    */
    abstol = 0.0;

    /* nag_dsbevx (f08hbc).
    * Solve the band symmetric eigenvalue problem.
    */
    nag_dsbevx(order, Nag_DoBoth, Nag_Interval, uplo, n, kd, ab, pdab, q,
               pdq, vl, vu, il, iu, abstol, &m, w, z, pdz, index, &fail);
    if (fail.code != NE_NOERROR && fail.code != NE_CONVERGENCE)
        {
            printf("Error from nag_dsbevx (f08hbc).\n%s\n", fail.message);
            exit_status = 1;
            goto END;
        }
    /* Print solution */
    printf("Number of eigenvalues found =%5"NAG_IFMT"\n", m);
    printf("\nEigenvalues\n" );
    for (j = 0; j < m; ++j)
        printf("%8.4f%" , w[j], (j+1)&8 == 0?"\n": "");
    printf("\n\n" );

    /* nag_gen_real_mat_print (x04cac).
    * Print selected eigenvectors.
    */
    INIT_FAIL(fail_print);
    fflush(stdout);
    nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, z,
                           pdz, "Selected eigenvectors", 0, &fail_print);
    if (fail_print.code != NE_NOERROR)
        {
            printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", fail_print.message);
            exit_status = 1;
            goto END;
        }
    if (fail.code == NE_CONVERGENCE)


```c
{    
    printf("eigenvectors failed to converge\n");
    printf("Indices of eigenvectors that did not converge\n");
    for (j = 0; j < m; ++j)
        printf("%8\n", index[j], (j+1)%8 == 0?\n":" ");
}

END:
NAG_FREE(ab);
NAG_FREE(q);
NAG_FREE(w);
NAG_FREE(z);
NAG_FREE(index);

    return exit_status;
}

#undef AB_UPPER
#undef AB_LOWER

10.2 Program Data

nag_dsbevx (f08hbc) Example Program Data

5 2 :Values of n and kd
Nag_Upper :Value of uplo
-3.0 3.0 :Values of vl and vu
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

10.3 Program Results

nag_dsbevx (f08hbc) Example Program Results

Number of eigenvalues found = 2

Eigenvalues
-2.6633 1.7511

Selected eigenvectors
  1 2
1 -0.6238 -0.5635
2 0.2575 0.3896
3 0.5900 -0.4008
4 -0.4308 0.5581
5 -0.1039 -0.2421
```