1 Purpose

nag_dspevx (f08gbc) computes selected eigenvalues and, optionally, eigenvectors of a real \( n \) by \( n \) symmetric matrix \( A \) in packed storage. 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_dspevx (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
                 Nag_UploType uplo, Integer n, double ap[], 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 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  

   **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 \( \text{order} = \text{Nag_RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

   **Constraint:** \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2:  **job**  – Nag_JobType  

   **Input**

   *On entry:* indicates whether eigenvectors are computed.

   \( \text{job} = \text{Nag_EigVals} \)
   
   Only eigenvalues are computed.

   \( \text{job} = \text{Nag_DoBoth} \)
   
   Eigenvalues and eigenvectors are computed.

   **Constraint:** \( \text{job} = \text{Nag_EigVals} \) or \( \text{Nag_DoBoth} \).
1. **range** – Nag.RangeType
   
   *Input*
   
   On entry: if **range** = Nag.AllValues, all eigenvalues will be found.
   
   If **range** = Nag.Interval, all eigenvalues in the half-open interval \( vl, vu \) will be found.
   
   If **range** = Nag.Indices, the \( il \)th to \( iu \)th eigenvalues will be found.
   
   Constraint: **range** = Nag.AllValues, Nag.Interval or Nag.Indices.

2. **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.

3. **n** – Integer
   
   *Input*
   
   On entry: \( n \), the order of the matrix \( A \).
   
   Constraint: \( n \geq 0 \).

4. **ap[\text{dim}]** – double
   
   *Input/Output*
   
   Note: the dimension, \( \text{dim} \), of the array \( ap \) must be at least \( \max(1, n \times (n + 1)/2) \).

   On entry: the upper or lower triangle of the \( n \) by \( n \) symmetric matrix \( A \), packed by rows or columns.

   The storage of elements \( A_{ij} \) depends on the **order** and **uplo** arguments as follows:
   
   if **order** = Nag.ColMajor and **uplo** = Nag.Upper,
   
   \( A_{ij} \) is stored in \( ap[(j - 1) \times j/2 + i - 1] \), for \( i \leq j \);
   
   if **order** = Nag.ColMajor and **uplo** = Nag.Lower,
   
   \( A_{ij} \) is stored in \( ap[(2n - j) \times (j - 1)/2 + i - 1] \), for \( i \geq j \);
   
   if **order** = Nag.RowMajor and **uplo** = Nag.Upper,
   
   \( A_{ij} \) is stored in \( ap[(2n - i) \times (i - 1)/2 + j - 1] \), for \( i \leq j \);
   
   if **order** = Nag.RowMajor and **uplo** = Nag.Lower,
   
   \( A_{ij} \) is stored in \( ap[(i - 1) \times i/2 + j - 1] \), for \( i \geq j \).

   On exit: \( ap \) is overwritten by the values generated during the reduction to tridiagonal form. The elements of the diagonal and the off-diagonal of the tridiagonal matrix overwrite the corresponding elements of \( A \).

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

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

7. **il** – Integer
   
   *Input*
   
   On entry: if **range** = Nag.Indices, the indices (in ascending order) of the smallest and largest eigenvalues to be returned.
   
   If **range** = Nag.AllValues or Nag.Interval, \( il \) and \( iu \) are not referenced.
   
   Constraints:
   
   if **range** = Nag.Indices and \( n = 0 \), \( il = 1 \) and \( iu = 0 \);
   
   if **range** = Nag.Indices and \( n > 0 \), \( 1 \leq il \leq iu \leq n \).
abstol – double

**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 **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 \(2 \times \text{nag\_real\_safe\_small\_number}\), not zero. If this function returns with \(\text{fail\_code} = \text{NE\_CONVERGENCE}\), indicating that some eigenvectors did not converge, try setting \(\text{abstol}\) to \(2 \times \text{nag\_real\_safe\_small\_number}\). See Demmel and Kahan (1990).

m – Integer*

**Output**

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

If \(\text{range} = \text{Nag\_AllValues}\), \(m = n\).

If \(\text{range} = \text{Nag\_Indices}\), \(m = iu - il + 1\).

w[n] – double

**Output**

On exit: the selected eigenvalues in ascending order.

z[dim] – double

**Output**

Note: the dimension, \(dim\), of the array \(z\) must be at least

\[
\max(1, pdz \times n) \quad \text{when} \quad \text{job} = \text{Nag\_DoBoth};
\]

\[
1 \quad \text{otherwise}.
\]

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

\[
\begin{align*}
  z[(j - 1) \times pdz + i - 1] \quad \text{when} \quad order = \text{Nag\_ColMajor}; \\
  z[(i - 1) \times pdz + j - 1] \quad \text{when} \quad order = \text{Nag\_RowMajor}.
\end{align*}
\]

On exit: if \(\text{job} = \text{Nag\_DoBoth}\), then

if \(\text{fail\_code} = \text{NE\_NOERROR}\), 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 (\(\text{fail\_code} = \text{NE\_CONVERGENCE}\)), then that column of \(Z\) contains the latest approximation to the eigenvector, and the index of the eigenvector is returned in \(jfail\).

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

pdz – Integer

**Input**

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

Constraints:

- if \(\text{job} = \text{Nag\_DoBoth}\), \(pdz \geq \max(1, n)\);
- otherwise \(pdz \geq 1\).

jfail[dim] – Integer

**Output**

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

On exit: if \(\text{job} = \text{Nag\_DoBoth}\), then
if \texttt{fail.code} = \texttt{NE_NOERROR}, the first \texttt{m} elements of \texttt{jfail} are zero;

if \texttt{fail.code} = \texttt{NE_CONVERGENCE}, \texttt{jfail} contains the indices of the eigenvectors that failed to converge.

If \texttt{job} = \texttt{Nag_EigVals}, \texttt{jfail} is not referenced.

17: \texttt{fail} \texttt{- NAGError *} \hspace{1cm} \texttt{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 \langle \textit{value} \rangle had an illegal value.

**NE_CONVERGENCE**

The algorithm failed to converge; \langle \textit{value} \rangle eigenvectors did not converge. Their indices are stored in array \texttt{jfail}.

**NE_ENUM_INT_2**

On entry, \texttt{job} = \langle \textit{value} \rangle, \texttt{pdz} = \langle \textit{value} \rangle and \texttt{n} = \langle \textit{value} \rangle.

Constraint: if \texttt{job} = \texttt{Nag_DoBoth}, \texttt{pdz} \geq \max(1, \texttt{n});

otherwise \texttt{pdz} \geq 1.

**NE_ENUM_INT_3**

On entry, \texttt{range} = \langle \textit{value} \rangle, \texttt{il} = \langle \textit{value} \rangle, \texttt{iu} = \langle \textit{value} \rangle and \texttt{n} = \langle \textit{value} \rangle.

Constraint: if \texttt{range} = \texttt{Nag_Indices} and \texttt{n} = 0, \texttt{il} = 1 and \texttt{iu} = 0;

if \texttt{range} = \texttt{Nag_Indices} and \texttt{n} > 0, 1 \leq \texttt{il} \leq \texttt{iu} \leq \texttt{n}.

**NE_ENUM_REAL_2**

On entry, \texttt{range} = \langle \textit{value} \rangle, \texttt{vl} = \langle \textit{value} \rangle and \texttt{vu} = \langle \textit{value} \rangle.

Constraint: if \texttt{range} = \texttt{Nag_Interval}, \texttt{vl} < \texttt{vu}.

**NE_INT**

On entry, \texttt{n} = \langle \textit{value} \rangle.

Constraint: \texttt{n} \geq 0.

On entry, \texttt{pdz} = \langle \textit{value} \rangle.

Constraint: \texttt{pdz} \geq 0.

**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_dspevx (f08gbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dspevx (f08gbc) 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 \(n^3\).

The complex analogue of this function is nag_zhpevx (f08gpc).

10 Example

This example finds the eigenvalues in the half-open interval \((-1, 1]\), and the corresponding eigenvectors, of the symmetric matrix

\[
A = \begin{pmatrix}
1 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 \\
3 & 3 & 3 & 4 \\
4 & 4 & 4 & 4
\end{pmatrix}.
\]

10.1 Program Text

/* nag_dspevx (f08gbc) 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, m, n, pdz;
  /* Arrays */
  char nag_enum_arg[40];
  double *ap = 0, *w = 0, *z = 0;
  Integer *index = 0;
  /* Nag Types */
  Nag_OrderType order;
  Nag_UpperType uplo;
  NagError fail, fail_print;
#ifdef NAG_COLUMN_MAJOR
#define AP_UPPER(I, J) ap[J * (J - 1) / 2 + I - 1]
#define AP_LOWER(I, J) ap[(2 * n - J) * (J - 1) / 2 + I - 1]
#define Z(I, J) z[(J - 1) * pdz + I - 1]
#define order = Nag_ColMajor;
#else
#define AP_LOWER(I, J) ap[I * (I - 1)/2+J-1]
#define AP_UPPER(I, J) ap[(2 * n - I) * (I - 1) / 2 + J - 1]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
#define order = Nag_RowMajor;
#endif

INIT_FAIL(fail);
printf("nag_dspevx (f08gbc) Example Program Results

");

/* Skip heading in data file */
#include WIN32
scanf_s("%*[\n]");
#else
scanf("%*[\n]");
#endif

/* Read uplo */
#include 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). */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

pdz = n;
m = n;

/* Allocate memory */
if (!ap || !w || !z || !index)
{
  printf("Allocation failure\n")
  exit_status = -1;
  goto END;
}

/* 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. */
#include WIN32
scanf_s("%lf%lf%*[\n]", &vl, &vu);
#else
scanf("%lf%lf%*[\n]", &vl, &vu);
#endif
if (uplo == Nag_Upper) {
  for (i = 1; i <= n; ++i)
    for (j = i; j <= n; ++j)
      scanf("%lf", &AP_UPPER(i, j));
}
else
  scanf("%lf", &AP_UPPER(i, j));
#endif

#include WIN32
```c
scanf_s("%*[\n"];
#else
    scanf("%*[\n"];
#endif
}
else if (uplo == Nag_Lower) {
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
            #ifdef _WIN32
                scanf_s("%lf", &AP_LOWER(i, j));
            #else
                scanf("%lf", &AP_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_dspevx (f08gbc).
*Solve the symmetric eigenvalue problem.
*/
if (fail.code != NE_NOERROR && fail.code != NE_CONVERGENCE)
    { printf("Error from nag_dspevx (f08gbc).%s", fail.message); 
      exit_status = 1;
      goto END;
    }
/* Normalize the eigenvectors */
for(j=1; j<=m; j++)
    for(i=n; i>=1; i--)
        Z(i, j) = Z(i, j) / Z(1,j);
/* Print solution */
printf("Number of eigenvalues found =%5"NAG_IFMT"\n", m);

f08 – Least-squares and Eigenvalue Problems (LAPACK)

Mark 25

f08gbc.7
END:
NAG_FREE(ap);
NAG_FREE(w);
NAG_FREE(z);
NAG_FREE(index);

    return exit_status;
}

#undef AP_UPPER
#undef AP_LOWER
#undef Z

10.2 Program Data
nag_dspevx (f08gbc) Example Program Data

4 :Value of n
Nag_Upper :Value of uplo
-1.0 1.0 :Values of vl and vu

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

10.3 Program Results
nag_dspevx (f08gbc) Example Program Results

Number of eigenvalues found = 2

Eigenvalues
-0.5146 -0.2943

Selected eigenvectors

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>-0.9431</td>
<td>-2.3976</td>
</tr>
<tr>
<td>3</td>
<td>-1.0537</td>
<td>2.3508</td>
</tr>
<tr>
<td>4</td>
<td>0.8831</td>
<td>-0.8879</td>
</tr>
</tbody>
</table>