NAG Library Function Document

nag_dspevd (f08gcc)

1 Purpose

nag_dspevd (f08gcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric matrix held in packed storage. 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_dspevd (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
                   Integer n, double ap[], double w[], double z[], Integer pdz,
                   NagError *fail)
```

3 Description

nag_dspevd (f08gcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric matrix $A$ (held in packed storage). 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 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: \( \text{job} \) – Nag_JobType 

*Input*

*On entry:* indicates whether eigenvectors are computed.

- \( \text{job} = \text{Nag_DoNothing} \)
  - Only eigenvalues are computed.
- \( \text{job} = \text{Nag_EigVecs} \)
  - Eigenvalues and eigenvectors are computed.

*Constraint:* \( \text{job} = \text{Nag_DoNothing} \) or \( \text{Nag_EigVecs} \).

3: \( \text{uplo} \) – 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: \( \text{ap}[\text{dim}] \) – double 

*Input/Output*

*Note:* the dimension, \( \text{dim} \), of the array \( \text{ap} \) 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 \( \text{order} \) and \( \text{uplo} \) arguments as follows:

- if \( \text{order} = \text{Nag_ColMajor} \) and \( \text{uplo} = \text{Nag_Upper} \),
  - \( A_{ij} \) is stored in \( \text{ap}[(j - 1) \times j/2 + i - 1] \), for \( i \leq j \);
- if \( \text{order} = \text{Nag_ColMajor} \) and \( \text{uplo} = \text{Nag_Lower} \),
  - \( A_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j - 1)/2 + i - 1] \), for \( i \geq j \);
- if \( \text{order} = \text{Nag_RowMajor} \) and \( \text{uplo} = \text{Nag_Upper} \),
  - \( A_{ij} \) is stored in \( \text{ap}[(2n - i) \times (i - 1)/2 + j - 1] \), for \( i \leq j \);
- if \( \text{order} = \text{Nag_RowMajor} \) and \( \text{uplo} = \text{Nag_Lower} \),
  - \( A_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1] \), for \( i \geq j \).

*On exit:* \( \text{ap} \) 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 \( A \).

6: \( \text{w}[\text{dim}] \) – double 

*Output*

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

*On exit:* the eigenvalues of the matrix \( A \) in ascending order.

7: \( \text{z}[\text{dim}] \) – double 

*Output*

*Note:* the dimension, \( \text{dim} \), of the array \( \text{z} \) 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 \( Z \) is stored in

- \( \text{z}[(j - 1) \times \text{pdz} + i - 1] \) when \( \text{order} = \text{Nag_ColMajor} \);
- \( \text{z}[(i - 1) \times \text{pdz} + j - 1] \) when \( \text{order} = \text{Nag_RowMajor} \).
On exit: if \( \text{job} = \text{Nag EigVecs} \), \( z \) is overwritten by the orthogonal matrix \( Z \) which contains the eigenvectors of \( A \).

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

8: \[ \text{pdz} \rightarrow \text{Integer} \]

\[ \text{Input} \]

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

\[ \text{Constraints:} \]

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

9: \[ \text{fail} \rightarrow \text{NagError *} \]

\( \text{Input/Output} \)

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 \( \langle \text{value} \rangle \) had an illegal value.

**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 \( \langle \text{value} \rangle \mod(n + 1) \).

**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 \( \text{job} = \text{Nag EigVecs} \), \( \text{pdz} \geq \text{max}(1, n) \); if \( \text{job} = \text{Nag DoNothing} \), \( \text{pdz} \geq 1 \).

**NE_INT**

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{pdz} = \langle \text{value} \rangle \).

Constraint: \( \text{pdz} > 0 \).

**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_dspevd (f08gcc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dspevd (f08gcc) 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_zhpevd (f08gqc).

10 Example
This example computes all the eigenvalues and eigenvectors of the symmetric matrix \( A \), where
\[
A = \begin{pmatrix}
1.0 & 2.0 & 3.0 & 4.0 \\
2.0 & 2.0 & 3.0 & 4.0 \\
3.0 & 3.0 & 3.0 & 4.0 \\
4.0 & 4.0 & 4.0 & 4.0
\end{pmatrix}.
\]

10.1 Program Text
/* nag_dspevd (f08gcc) Example Program. */
/* Copyright 2014 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, n, ap_len, pdz, w_len;
    Integer exit_status = 0;
    NagError fail;
    NagJobType job;
    NagUploType uplo;
    NagOrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    double *ap = 0, *w = 0, *z = 0;

    #ifdef NAG_COLUMN_MAJOR
    #define A_UPPER(I, J) ap[J * (J - 1) / 2 + I - 1]
    #define A_LOWER(I, J) ap[(2 * n - J) * (J - 1) / 2 + I - 1]
    #define Z(I, J) z[(J - 1) * pdz + I - 1]
    #else
    #define A_UPPER(I, J) ap[(J - 1) * (J - 1) / 2 + I - 1]
    #define A_LOWER(I, J) ap[(J - 1) * (J - 1) / 2 + I - 1]
    #define Z(I, J) z[(J - 1) * pdz + I - 1]
    #endif

    /* ...
    */
order = Nag_ColMajor;
#endif
#define A_LOWER(I, J) ap[I * (I - 1) / 2 + J - 1]
#define A_UPPER(I, J) ap[(2 * n - I) * (I - 1) / 2 + J - 1]
order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_dspevd (f08gcc) 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"%*[\n"] , &n);
#else
scanf("%"NAG_IFMT"%*[\n"] , &n);
#endif
ap_len = n*(n+1)/2;
w_len = n;
pdz = n;

/* Allocate memory */
if(!(ap = NAG_ALLOC(ap_len, double)) ||
 !(z = NAG_ALLOC(n * n, double)) ||
 !(w = NAG_ALLOC(w_len, 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));
#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 */
if (uplo == Nag_Upper)
{
 for (i = 1; i <= n; ++i)
 {
  for (j = i; j <= n; ++j)
  #ifdef _WIN32
   scanf_s("%lf", &A_UPPER(i, j));
  #else
   scanf("%lf", &A_UPPER(i, j));
  #endif
  }
  #ifdef _WIN32
  scanf_s("%*[\n"]);
  #else
  scanf("%*[\n"]);
  #endif
}
else
{
 for (i = 1; i <= n; ++i)
 {
  for (j = 1; j <= i; ++j)
  #ifdef _WIN32
   scanf_s("%lf", &A_LOWER(i, j));
  #else
   scanf("%lf", &A_LOWER(i, j));
  #endif
  }
}
```c
#if defined _WIN32
    scanf_s("%*\[\n "");
#else
    scanf("%*\[\n ");
#endif

} /* Read type of job to be performed */
#elif defined _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_dspevd (f08gcc).
   * All eigenvalues and optionally all eigenvectors of real
   * symmetric matrix, packed storage (divide-and-conquer)
   */
    nag_dspevd(order, job, uplo, n, ap, w, z, pdz, &fail);
    if (fail.code != NE_NOERROR)
    {
        printf("Error from nag_dspevd (f08gcc).
%s
", 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");
    /* 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).
%s
", fail.message);
        exit_status = 1;
        goto END;
    }
END:
    NAG_FREE(ap);
    NAG_FREE(w);
    NAG_FREE(z);
    return exit_status;
```
10.2 Program Data

nag_dspevd (f08gcc) Example Program Data

- Value of n
- Nag_Lower : Value of uplo
- 1.0
- 2.0 2.0
- 3.0 3.0 3.0
- 4.0 4.0 4.0 4.0 : End of matrix A
- Nag_EigVecs : Value of job

10.3 Program Results

nag_dspevd (f08gcc) Example Program Results

Eigenvalues
- -2.0531 -0.5146 -0.2943 12.8621

Eigenvectors
- 1 2 3 4
- 1 1.0000 1.0000 1.0000 1.0000
- 2 0.5129 -0.9431 -2.3976 1.0777
- 3 -0.2240 -1.0537 2.3508 1.2393
- 4 -0.8518 0.8831 -0.8879 1.4972