NAG Library Function Document
nag_dsyevd (f08fcc)

1 Purpose
nag_dsyevd (f08fcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric 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_dsyevd (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
                 Integer n, double a[], Integer pda, double w[], NagError *fail)
```

3 Description
nag_dsyevd (f08fcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric 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 Arguments
1: \textbf{order} – Nag_OrderType  \hspace{1cm} \textit{Input}
   \textit{On entry:} the \textbf{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 \textbf{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.
   \textit{Constraint:} \textbf{order} = Nag_RowMajor or Nag_ColMajor.

2: \textbf{job} – Nag_JobType  \hspace{1cm} \textit{Input}
   \textit{On entry:} indicates whether eigenvectors are computed.
   \textbf{job} = Nag_DoNothing\hfill Only eigenvalues are computed.
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.

uplo = Nag_Upper
   The upper triangular part of A is stored.

uplo = Nag_Lower
   The lower triangular part of A is stored.

Constraint: uplo = Nag_Upper or Nag_Lower.

4: n – Integer
   Input

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

Constraint: n ≥ 0.

5: a[dim] – double
   Input/Output

Note: the dimension, dim, of the array a must be at least max(1, pda × n).

On entry: the n by n symmetric matrix A.

If order = Nag_ColMajor, A_{ij} is stored in a((j - 1) × pda + i - 1).

If order = Nag_RowMajor, A_{ij} is stored in a((i - 1) × pda + j - 1).

If uplo = Nag_Upper, the upper triangular part of A must be stored and the elements of the array below the diagonal are not referenced.

If uplo = Nag_Lower, the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: if job = Nag_EigVecs, a is overwritten by the orthogonal matrix Z which contains the eigenvectors of A.

6: pda – Integer
   Input

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

Constraint: pda ≥ max(1, n).

7: 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.

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

NE_INT_2
On entry, \( pda = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).
Constraint: \( pda \geq \max(1, n) \).

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 \textit{machine precision}. See Section 4.7 of Anderson \textit{et al.} (1999) for further details.

8 Parallelism and Performance
\texttt{nag_dsyevd (f08fcc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
\texttt{nag_dsyevd (f08fcc)} 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 \texttt{nag_zheevd (f08fqc)}. 
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_dsyevd (f08fcc) 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, pda, w_len;
    Integer exit_status = 0;
    NagError fail;
    Nag_JobType job;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char nag_enum_arg[40];
    double *a = 0, *w = 0;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J - 1) * pda + I - 1]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I - 1) * pda + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);
    printf("nag_dsyevd (f08fcc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n"]);
    #else
    scanf("%*[\n"]);
    #endif
    #ifdef _WIN32
    scanf("%"NAG_IFMT "%*[\n"]", &n);
    #else
    scanf("%"NAG_IFMT "%*[\n"]", &n);
    #endif
    pda = n;
    w_len = n;

    /* Allocate memory */
    if (!(a = NAG_ALLOC(n * n, double)) ||
        !(w = NAG_ALLOC(w_len, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
    }
/* 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(i, j));
            #else
                scanf("%lf", &A(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(i, j));
            #else
                scanf("%lf", &A(i, j));
            #endif
        }
    #ifdef _WIN32
        scanf_s("%\n ");
    #else
        scanf("%\n ");
    #endif
}
/* Read type of job to be performed */
#ifdef _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_dsyevd (f08fcc).
* All eigenvalues and optionally all eigenvectors of real
* symmetric matrix (divide-and-conquer) */
    nag_dsyevd(order, job, uplo, n, a, pda, w, &fail);
if (fail.code != NE_NOERROR)
    { printf("Error from nag_dsyevd (f08fcc)\n", fail.message);
        exit_status = 1;
    goto END;
    }
/* Normalize the eigenvectors */
for(j=1; j<=n; j++)
    { for(i=n; i>=1; i--)
        { /* Normalization code */
        }
\( A(i, j) = A(i, j) / A(1,j); \)

} /* 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, a,
pda, "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(a);
NAG_FREE(w);
return exit_status;

10.2 Program Data
nag_dsyevd (f08fcc) Example Program Data
4 :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_dsyevd (f08fcc) Example Program Results

Eigenvalues
-2.0531 -0.5146 -0.2943 12.8621
Eigenvectors
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.5129</td>
<td>-0.9431</td>
<td>-2.3976</td>
</tr>
<tr>
<td>3</td>
<td>-0.2240</td>
<td>-1.0537</td>
<td>2.3508</td>
</tr>
<tr>
<td>4</td>
<td>-0.8518</td>
<td>0.8831</td>
<td>-0.8879</td>
</tr>
</tbody>
</table>