NAG Library Function Document

nag_dsyev (f08fac)

1 Purpose

nag_dsyev (f08fac) computes all the eigenvalues and, optionally, all the eigenvectors of a real $n$ by $n$ symmetric matrix $A$.

2 Specification

```c
#include <nag.h>
#include <nagf08.h>
void nag_dsyev (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
    Integer n, double a[], Integer pda, double w[], NagError *fail)
```

3 Description

The symmetric matrix $A$ is first reduced to tridiagonal form, using orthogonal similarity transformations, and then the QR algorithm is applied to the tridiagonal matrix to compute the eigenvalues and (optionally) the eigenvectors.

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:  
```
job – Nag_JobType
```

*Input*

On entry: indicates whether eigenvectors are computed.

job = Nag_EigVals

Only eigenvalues are computed.

job = Nag_DoBoth

Eigenvalues and eigenvectors are computed.

Constraint: job = Nag_EigVals or Nag_DoBoth.

3:  
```
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.

4: n – Integer
   Input
   On entry: $n$, the order of the matrix $A$.
   Constraint: $n \geq 0$.

5: a[dim] – double
   Input/Output
   Note: the dimension, dim, of the array a must be at least max$(1, pda \times n)$.
   On entry: the $n$ by $n$ symmetric matrix $A$.
   If order = Nag_ColMajor, $A_{ij}$ is stored in $a[(j - 1) \times pda + i - 1]$.
   If order = Nag_RowMajor, $A_{ij}$ is stored in $a[(i - 1) \times pda + j - 1]$.
   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_DoBoth, then a contains the orthonormal eigenvectors of the matrix $A$.
   If job = Nag_EigVals, then on exit the lower triangle (if uplo = Nag_Lower) or the upper triangle
   (if uplo = Nag_Upper) of a, including the diagonal, is overwritten.

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 \geq \max(1, n).

7: w[n] – double
   Output
   On exit: the eigenvalues 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 $\langle value\rangle$ had an illegal value.

NE_CONVERGENCE
   The algorithm failed to converge; $\langle value\rangle$ off-diagonal elements of an intermediate tridiagonal
   form did not converge to zero.

NE_INT
   On entry, $n = \langle value\rangle$.
   Constraint: $n \geq 0$. 

f08fac.2
On entry, $pda = \langle value \rangle$.
Constraint: $pda > 0$.

**NE_INT_2**

On entry, $pda = \langle value \rangle$ and $n = \langle value \rangle$.
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 *machine precision*. See Section 4.7 of Anderson *et al.* (1999) for further details.

8 **Parallelism and Performance**

nag_dsyev (f08fac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dsyev (f08fac) 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_zheev (f08fnc).

10 **Example**

This example finds all the eigenvalues and eigenvectors of the symmetric matrix

$$
A = \begin{pmatrix}
1 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 \\
3 & 3 & 3 & 4 \\
4 & 4 & 4 & 4
\end{pmatrix},
$$

together with approximate error bounds for the computed eigenvalues and eigenvectors.
#include <math.h>
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx02.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double eerrbd, eps;
    Integer i, j, n, pda;
    Integer exit_status = 0;
    /* Arrays */
    double *a = 0, *rcondz = 0, *w = 0, *zerrbd = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;

    #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_dsyev (f08fac) Example Program Results\n\n");

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

    /* Allocate memory */
    if (!(a = NAG_ALLOC(n*n, double)) ||
        !(rcondz = NAG_ALLOC(n, double)) ||
        !(w = NAG_ALLOC(n, double)) ||
        !(zerrbd = NAG_ALLOC(n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    pda = n;

    /* Read the upper triangular part of the matrix A from data file */
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
            scanf_s("%lf", &A(i, j));
    #else

scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
scanf_s("%*\[\n\n");
#else
scanf("%*\[\n\n");
#endif

/* nag_dsyev (f08fac).
 * Solve the symmetric eigenvalue problem.
 */
nag_dsyev(order, Nag_DoBoth, Nag_Upper, n, a, pda, w, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsyev (f08fac).\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Normalize the eigenvectors */
for(j=1; j<=n; j++)
    for(i=n; i>=1; i--)
        A(i, j) = A(i, j) / A(1,j);
/* Print solution */
printf("Eigenvalues\n");
for (j = 0; j < n; ++j)
    printf("%8.4f%s", w[j], (j+1)%8 == 0?"\n":" ");
printf("\n\n");
/* nag_gen_real_mat_print (x04cac).
 * Print eigenvectors.
 */
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\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Get the machine precision, eps, using nag_machine_precision (X02AJC)
 * and compute the approximate error bound for the computed eigenvalues.
 * Note that for the 2-norm, ||A|| = max (|w[i]|, i=0..n-1), and since
 * the eigenvalues are in ascending order ||A|| = max( |w[0]|, |w[n-1]|).
 */
eps = nag_machine_precision;
eerrbd = eps * MAX(fabs(w[0]), fabs(w[n-1]));
/* nag_ddisna (f08flc).
 * Estimate reciprocal condition numbers for the eigenvectors.
 */
nag_ddisna(Nag_EigVecs, n, n, w, rcondz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_ddisna (f08flc).\n\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Compute the error estimates for the eigenvectors */
for (i = 0; i < n; ++i)
    zerrbd[i] = ierrbd / rcondz[i];
/* Print the approximate error bounds for the eigenvalues and vectors */
printf("Error estimate for the eigenvalues\n");
printf("%11.1e\n", ierrbd);
printf("Error estimates for the eigenvectors\n");
for (i = 0; i < n; ++i)
    printf("%11.1e%s", zerrbd[i], (i+1)%6 == 0?"\n":" ");

END:
    NAG_FREE(a);
    NAG_FREE(rcondz);
    NAG_FREE(w);
    NAG_FREE(zerrbd);

    return exit_status;
}

#undef A

## 10.2 Program Data

**nag_dsyev (f08fac) Example Program Data**

4 :Value of n

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_dsyev (f08fac) Example Program Results**

Eigenvalues

-2.0531  -0.5146  -0.2943  12.8621

Eigenvectors

<table>
<thead>
<tr>
<th></th>
<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>
<td>1.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.5129</td>
<td>-0.9431</td>
<td>-2.3976</td>
<td>1.0777</td>
</tr>
<tr>
<td>3</td>
<td>-0.2240</td>
<td>-1.0537</td>
<td>2.3508</td>
<td>1.2393</td>
</tr>
<tr>
<td>4</td>
<td>-0.8518</td>
<td>0.8831</td>
<td>-0.8879</td>
<td>1.4972</td>
</tr>
</tbody>
</table>

Error estimate for the eigenvalues

1.4e-15

Error estimates for the eigenvectors

9.3e-16  6.5e-15  6.5e-15  1.1e-16