

# NAG Library Function Document

## nag\_dsyevr (f08fdc)

### 1 Purpose

nag\_dsyevr (f08fdc) computes selected eigenvalues and, optionally, eigenvectors of a real  $n$  by  $n$  symmetric matrix  $A$ . Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

### 2 Specification

```
#include <nag.h>
#include <nagf08.h>

void nag_dsyevr (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
                Nag_UploType uplo, Integer n, double a[], Integer pda, double vl,
                double vu, Integer il, Integer iu, double abstol, Integer *m,
                double w[], double z[], Integer pdz, Integer isuppz[], NagError *fail)
```

### 3 Description

The symmetric matrix is first reduced to a tridiagonal matrix  $T$ , using orthogonal similarity transformations. Then whenever possible, nag\_dsyevr (f08fdc) computes the eigenspectrum using Relatively Robust Representations. nag\_dsyevr (f08fdc) computes eigenvalues by the dqds algorithm, while orthogonal eigenvectors are computed from various ‘good’  $LDL^T$  representations (also known as Relatively Robust Representations). Gram–Schmidt orthogonalization is avoided as far as possible. More specifically, the various steps of the algorithm are as follows. For the  $i$ th unreduced block of  $T$ :

- compute  $T - \sigma_i I = L_i D_i L_i^T$ , such that  $L_i D_i L_i^T$  is a relatively robust representation,
- compute the eigenvalues,  $\lambda_j$ , of  $L_i D_i L_i^T$  to high relative accuracy by the dqds algorithm,
- if there is a cluster of close eigenvalues, ‘choose’  $\sigma_i$  close to the cluster, and go to (a),
- given the approximate eigenvalue  $\lambda_j$  of  $L_i D_i L_i^T$ , compute the corresponding eigenvector by forming a rank-revealing twisted factorization.

The desired accuracy of the output can be specified by the argument **abstol**. For more details, see Dhillon (1997) and Parlett and Dhillon (2000).

### 4 References

Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J J, Du Croz J J, Greenbaum A, Hammarling S, McKenney A and Sorensen D (1999) *LAPACK Users’ Guide* (3rd Edition) SIAM, Philadelphia <http://www.netlib.org/lapack/lug>

Barlow J and Demmel J W (1990) Computing accurate eigensystems of scaled diagonally dominant matrices *SIAM J. Numer. Anal.* **27** 762–791

Demmel J W and Kahan W (1990) Accurate singular values of bidiagonal matrices *SIAM J. Sci. Statist. Comput.* **11** 873–912

Dhillon I (1997) A new  $O(n^2)$  algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem *Computer Science Division Technical Report No. UCB//CSD-97-971* UC Berkeley

Golub G H and Van Loan C F (1996) *Matrix Computations* (3rd Edition) Johns Hopkins University Press, Baltimore

Parlett B N and Dhillon I S (2000) Relatively robust representations of symmetric tridiagonals *Linear Algebra Appl.* **309** 121–151

## 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: **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 **ilth** to **iuth** eigenvalues will be found.  
 For **range** = Nag\_Interval or Nag\_Indices and **iu** – **il** < **n** – 1, nag\_dstebz (f08jjc) and nag\_dstein (f08jkc) are called.  
*Constraint:* **range** = Nag\_AllValues, Nag\_Interval or Nag\_Indices.
- 4: **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.
- 5: **n** – Integer *Input*  
*On entry:* *n*, the order of the matrix *A*.  
*Constraint:* **n** ≥ 0.
- 6: **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:* the lower triangle (if **uplo** = Nag\_Lower) or the upper triangle (if **uplo** = Nag\_Upper) of **a**, including the diagonal, is overwritten.

- 7: **pda** – Integer *Input*  
*On entry:* the stride separating row or column elements (depending on the value of **order**) in the array **a**.  
*Constraint:* **pda**  $\geq$   $\max(1, \mathbf{n})$ .
- 8: **vl** – double *Input*  
9: **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**.
- 10: **il** – Integer *Input*  
11: **iu** – 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,  $1 \leq \mathbf{il} \leq \mathbf{iu} \leq \mathbf{n}$ .
- 12: **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  

$$\mathbf{abstol} + \epsilon \max(|a|, |b|),$$
where  $\epsilon$  is the *machine precision*. If **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. See Demmel and Kahan (1990).  
If high relative accuracy is important, set **abstol** to `nag_real_safe_small_number( )`, although doing so does not currently guarantee that eigenvalues are computed to high relative accuracy. See Barlow and Demmel (1990) for a discussion of which matrices can define their eigenvalues to high relative accuracy.
- 13: **m** – Integer \* *Output*  
*On exit:* the total number of eigenvalues found.  $0 \leq \mathbf{m} \leq \mathbf{n}$ .  
If **range** = Nag\_AllValues, **m** = **n**.  
If **range** = Nag\_Indices, **m** = **iu** – **il** + 1.
- 14: **w**[*dim*] – double *Output*  
**Note:** the dimension, *dim*, of the array **w** must be at least  $\max(1, \mathbf{n})$ .  
*On exit:* the first **m** elements contain the selected eigenvalues in ascending order.
- 15: **z**[*dim*] – double *Output*  
**Note:** the dimension, *dim*, of the array **z** must be at least  
 $\max(1, \mathbf{pdz} \times \mathbf{n})$  when **job** = Nag\_DoBoth;  
1 otherwise.

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

$$\begin{aligned} & \mathbf{z}[(j-1) \times \mathbf{pdz} + i - 1] \text{ when } \mathbf{order} = \text{Nag\_ColMajor}; \\ & \mathbf{z}[(i-1) \times \mathbf{pdz} + j - 1] \text{ when } \mathbf{order} = \text{Nag\_RowMajor}. \end{aligned}$$

*On exit:* if  $\mathbf{job} = \text{Nag\_DoBoth}$ , the first  $\mathbf{m}$  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  $\mathbf{w}[i-1]$ .

If  $\mathbf{job} = \text{Nag\_EigVals}$ ,  $\mathbf{z}$  is not referenced.

16: **pdz** – Integer *Input*

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

*Constraints:*

$$\begin{aligned} & \text{if } \mathbf{job} = \text{Nag\_DoBoth}, \mathbf{pdz} \geq \max(1, \mathbf{n}); \\ & \text{otherwise } \mathbf{pdz} \geq 1. \end{aligned}$$

17: **isuppz** $[\mathit{dim}]$  – Integer *Output*

**Note:** the dimension,  $\mathit{dim}$ , of the array **isuppz** must be at least  $\max(1, 2 \times \mathbf{m})$ .

*On exit:* the support of the eigenvectors in  $\mathbf{z}$ , i.e., the indices indicating the nonzero elements in  $\mathbf{z}$ . The  $i$ th eigenvector is nonzero only in elements **isuppz** $[2 \times i - 2]$  through **isuppz** $[2 \times i - 1]$ . Implemented only for  $\mathbf{range} = \text{Nag\_AllValues}$  or  $\text{Nag\_Indices}$  and  $\mathbf{i}u - \mathbf{i}l = \mathbf{n} - 1$ .

18: **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.

### NE\_BAD\_PARAM

On entry, argument  $\langle \mathit{value} \rangle$  had an illegal value.

### NE\_CONVERGENCE

nag\_dsyevr (f08fdc) failed to converge.

### NE\_ENUM\_INT\_2

On entry,  $\mathbf{job} = \langle \mathit{value} \rangle$ ,  $\mathbf{pdz} = \langle \mathit{value} \rangle$  and  $\mathbf{n} = \langle \mathit{value} \rangle$ .  
Constraint: if  $\mathbf{job} = \text{Nag\_DoBoth}$ ,  $\mathbf{pdz} \geq \max(1, \mathbf{n})$ ;  
otherwise  $\mathbf{pdz} \geq 1$ .

### NE\_ENUM\_INT\_3

On entry,  $\mathbf{range} = \langle \mathit{value} \rangle$ ,  $\mathbf{i}l = \langle \mathit{value} \rangle$ ,  $\mathbf{i}u = \langle \mathit{value} \rangle$  and  $\mathbf{n} = \langle \mathit{value} \rangle$ .  
Constraint: if  $\mathbf{range} = \text{Nag\_Indices}$  and  $\mathbf{n} = 0$ ,  $\mathbf{i}l = 1$  and  $\mathbf{i}u = 0$ ;  
if  $\mathbf{range} = \text{Nag\_Indices}$  and  $\mathbf{n} > 0$ ,  $1 \leq \mathbf{i}l \leq \mathbf{i}u \leq \mathbf{n}$ .

### NE\_ENUM\_REAL\_2

On entry,  $\mathbf{range} = \langle \mathit{value} \rangle$ ,  $\mathbf{v}l = \langle \mathit{value} \rangle$  and  $\mathbf{v}u = \langle \mathit{value} \rangle$ .  
Constraint: if  $\mathbf{range} = \text{Nag\_Interval}$ ,  $\mathbf{v}l < \mathbf{v}u$ .

**NE\_INT**

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

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

On entry, **pdz** =  $\langle value \rangle$ .  
 Constraint: **pdz**  $> 0$ .

**NE\_INT\_2**

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

**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.

**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\_dsyevr (f08fdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag\_dsyevr (f08fdc) 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 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\_zheevr (f08frc).

**10 Example**

This example finds the eigenvalues with indices in the range  $[2, 3]$ , 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_dsyevr (f08fdc) Example Program.
 *
 * Copyright 2011 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 = 0.0, vu = 0.0;
    Integer         exit_status = 0, i, il, iu, j, m, n;
    Integer         pda, pdz;
    /* Arrays */
    double          *a = 0, *w = 0, *z = 0;
    Integer         *isuppz = 0;
    /* Nag Types */
    Nag_OrderType  order;
    NagError       fail;

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

    INIT_FAIL(fail);

    printf("nag_dsyevr (f08fdc) Example Program Results\n\n");

    /* Skip heading in data file and read n and the lower and upper
     * indices of the smallest and largest eigenvalues to be found
     */
    scanf("%*[\n]");
    scanf("%ld%ld%ld%*[\n]", &n, &il, &iu);

    m = iu - il + 1;
    /* Allocate memory */
    if (!(a = NAG_ALLOC(n*n, double)) ||
        !(w = NAG_ALLOC(n, double)) ||
        !(z = NAG_ALLOC(n*n, double)) ||
        !(isuppz = NAG_ALLOC(m*2, Integer)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

    pda = n;
    pdz = 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("%lf", &A(i, j));
    scanf("%*[\n]");

    /* Set the absolute error tolerance for eigenvalues. With abstol
     * set to zero, the default value is used instead.
     */
    abstol = 0.0;

    /* nag_dsyevr (f08fdc).
     * Solve the symmetric eigenvalue problem.
     */
    nag_dsyevr(order, Nag_DoBoth, Nag_Indices, Nag_Upper, n, a, pda, vl, vu, il,

```

```

        iu, abstol, &m, w, z, pdz, isuppz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsyevr (f08fdc).\n%s\n", 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("Selected eigenvalues\n");
for (j = 0; j < m; ++j)
    printf("%8.4f%s", w[j], (j+1)%8 == 0?"\n": " ");
printf("\n");

/* nag_gen_real_mat_print (x04cac).
 * Print selected eigenvectors.
 */
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, z,
                        pdz, "Selected 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;
}

END:
NAG_FREE(a);
NAG_FREE(w);
NAG_FREE(z);
NAG_FREE(isuppz);

return exit_status;
}

#undef A
#undef Z

```

## 10.2 Program Data

nag\_dsyevr (f08fdc) Example Program Data

```

4      2      3          :Values of n, il and iu

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\_dsyevr (f08fdc) Example Program Results

```

Selected eigenvalues
-0.5146  -0.2943
Selected eigenvectors
          1          2
1         1.0000         1.0000
2        -0.9431        -2.3976
3        -1.0537         2.3508
4         0.8831        -0.8879

```

---