

NAG Library Function Document

nag_dstevr (f08jdc)

1 Purpose

nag_dstevr (f08jdc) computes selected eigenvalues and, optionally, eigenvectors of a real n by n symmetric tridiagonal matrix T . 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_dstevr (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
                Integer n, double d[], double e[], double vl, double vu, Integer il,
                Integer iu, double abstol, Integer *m, double w[], double z[],
                Integer pdz, Integer isuppz[], NagError *fail)
```

3 Description

Whenever possible nag_dstevr (f08jdc) computes the eigenspectrum using Relatively Robust Representations. nag_dstevr (f08jdc) 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, λ_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’ σ_i close to the cluster, and go to (a),
- given the approximate eigenvalue λ_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.
Constraint: **range** = Nag_AllValues, Nag_Interval or Nag_Indices.
- 4: **n** – Integer *Input*
On entry: *n*, the order of the matrix.
Constraint: $n \geq 0$.
- 5: **d**[*dim*] – double *Input/Output*
Note: the dimension, *dim*, of the array **d** must be at least $\max(1, n)$.
On entry: the *n* diagonal elements of the tridiagonal matrix *T*.
On exit: may be multiplied by a constant factor chosen to avoid over/underflow in computing the eigenvalues.
- 6: **e**[*dim*] – double *Input/Output*
Note: the dimension, *dim*, of the array **e** must be at least $\max(1, n - 1)$.
On entry: the (*n* - 1) subdiagonal elements of the tridiagonal matrix *T*.
On exit: may be multiplied by a constant factor chosen to avoid over/underflow in computing the eigenvalues.
- 7: **vl** – double *Input*
 8: **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**.

- 9: **il** – Integer *Input*
 10: **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, $1 \leq \mathbf{il} \leq \mathbf{iu} \leq \mathbf{n}$.

- 11: **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 ϵ is the *machine precision*. If **abstol** is less than or equal to zero, then $\epsilon \|T\|_1$ will be used in its place. 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.

- 12: **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.

- 13: **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.

- 14: **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

- $\mathbf{z}[(j - 1) \times \mathbf{pdz} + i - 1]$ when **order** = Nag_ColMajor;
- $\mathbf{z}[(i - 1) \times \mathbf{pdz} + j - 1]$ when **order** = Nag_RowMajor.

On exit: if **job** = Nag_DoBoth, 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 **job** = Nag_EigVals, **z** is not referenced.

- 15: **pdz** – Integer *Input*

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

Constraints:

- if **job** = Nag_DoBoth, $\mathbf{pdz} \geq \max(1, \mathbf{n})$;
- otherwise $\mathbf{pdz} \geq 1$.

16: **isuppz**[*dim*] – Integer *Output*

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

On exit: the support of the eigenvectors in **z**, i.e., the indices indicating the nonzero elements in **z**. The *i*th eigenvector is nonzero only in elements **isuppz**[$2 \times i - 2$] through **isuppz**[$2 \times i - 1$]. Implemented only for **range** = Nag_AllValues or Nag_Indices and **iu** – **il** = **n** – 1.

17: **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 *<value>* had an illegal value.

NE_ENUM_INT_2

On entry, **job** = *<value>*, **pdz** = *<value>* and **n** = *<value>*.
Constraint: if **job** = Nag_DoBoth, **pdz** \geq $\max(1, \mathbf{n})$;
otherwise **pdz** \geq 1.

NE_ENUM_INT_3

On entry, **range** = *<value>*, **il** = *<value>*, **iu** = *<value>* and **n** = *<value>*.
Constraint: if **range** = Nag_Indices and **n** = 0, **il** = 1 and **iu** = 0;
if **range** = Nag_Indices and **n** > 0, $1 \leq \mathbf{il} \leq \mathbf{iu} \leq \mathbf{n}$.

NE_ENUM_REAL_2

On entry, **range** = *<value>*, **vl** = *<value>* and **vu** = *<value>*.
Constraint: if **range** = Nag_Interval, **vl** < **vu**.

NE_INT

On entry, **n** = *<value>*.
Constraint: **n** \geq 0.

On entry, **pdz** = *<value>*.
Constraint: **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 internal error has occurred in this function. Please refer to **fail** in nag_dstebz (f08jjc).

7 Accuracy

The computed eigenvalues and eigenvectors are exact for a nearby matrix ($A + E$), where

$$\|E\|_2 = O(\epsilon)\|A\|_2,$$

and ϵ is the *machine precision*. See Section 4.7 of Anderson *et al.* (1999) for further details.

8 Parallelism and Performance

nag_dstevr (f08jdc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dstevr (f08jdc) 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^2 if **job** = Nag_EigVals and is proportional to n^3 if **job** = Nag_DoBoth and **range** = Nag_AllValues, otherwise the number of floating-point operations will depend upon the number of computed eigenvectors.

10 Example

This example finds the eigenvalues with indices in the range [2, 3], and the corresponding eigenvectors, of the symmetric tridiagonal matrix

$$T = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 4 & 2 & 0 \\ 0 & 2 & 9 & 3 \\ 0 & 0 & 3 & 16 \end{pmatrix}.$$

10.1 Program Text

```

/* nag_dstevr (f08jdc) 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, pdz;
    /* Arrays */
    double      *d = 0, *e = 0, *w = 0, *z = 0;
    Integer     *isuppz = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError    fail;

#ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
#else
    order = Nag_RowMajor;
#endif

    INIT_FAIL(fail);

    printf("nag_dstevr (f08jdc) Example Program Results\n\n");

    /* Skip heading in data file and read n and the lower and upper

```

```

    * indices of the eigenvalues to be found.
    */
scanf("%*[\n]");
scanf("%ld%ld%ld%*[\n]", &n, &il, &iu);

m = iu - il + 1;
pdz = n;

/* Allocate memory */
if (!(d = NAG_ALLOC(n, double)) ||
    !(e = NAG_ALLOC(n-1, double)) ||
    !(w = NAG_ALLOC(n, double)) ||
    !(z = NAG_ALLOC(n * n, double)) ||
    !(isuppz = NAG_ALLOC(n, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read the diagonal and off-diagonal elements of the matrix A
 * from data file.
 */
for (i = 0; i < n; ++i)
    scanf("%lf", &d[i]);
scanf("%*[\n]");

for (i = 0; i < n - 1; ++i)
    scanf("%lf", &e[i]);
scanf("%*[\n]");

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

/* nag_dstevr (f08jdc).
 * Solve the symmetric tridiagonal eigenvalue problem.
 */
nag_dstevr(order, Nag_DoBoth, Nag_Indices, n, d, e, vl, vu, il, iu, abstol,
           &m, w, z, pdz, isuppz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dstevr (f08jdc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* 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\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;
    goto END;
}

END:
NAG_FREE(d);
NAG_FREE(e);

```

```
NAG_FREE(w);  
NAG_FREE(z);  
NAG_FREE(isuppz);  
  
return exit_status;  
}
```

10.2 Program Data

nag_dstevr (f08jdc) Example Program Data

```
4      2      3           :Values of n, il and iu  
  
1.0  4.0  9.0  16.0  :End of diagonal elements  
1.0  2.0  3.0           :End of off-diagonal elements
```

10.3 Program Results

nag_dstevr (f08jdc) Example Program Results

Selected eigenvalues
3.5470 8.6578

Selected eigenvectors

	1	2
1	0.3388	0.0494
2	0.8628	0.3781
3	-0.3648	0.8558
4	0.0879	-0.3497
