

# NAG Library Function Document

## nag\_dstevd (f08jcc)

### 1 Purpose

nag\_dstevd (f08jcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric tridiagonal 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

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

void nag_dstevd (Nag_OrderType order, Nag_JobType job, Integer n, double d[],
                 double e[], double z[], Integer pdz, NagError *fail)
```

### 3 Description

nag\_dstevd (f08jcc) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric tridiagonal matrix  $T$ . In other words, it can compute the spectral factorization of  $T$  as

$$T = Z\Lambda Z^T,$$

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

$$Tz_i = \lambda_i z_i, \quad i = 1, 2, \dots, n.$$

### 4 References

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

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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\_DoNothing  
     Only eigenvalues are computed.  
**job** = Nag\_EigVecs  
     Eigenvalues and eigenvectors are computed.  
*Constraint:* **job** = Nag\_DoNothing or Nag\_EigVecs.

- 3:     **n** – Integer *Input*  
       *On entry:*  $n$ , the order of the matrix  $T$ .  
       *Constraint:*  $n \geq 0$ .
- 4:     **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:* the eigenvalues of the matrix  $T$  in ascending order.
- 5:     **e**[ $dim$ ] – double *Input/Output*  
       **Note:** the dimension,  $dim$ , of the array **e** must be at least  $\max(1, n)$ .  
       *On entry:* the  $n - 1$  off-diagonal elements of the tridiagonal matrix  $T$ . The  $n$ th element of this array is used as workspace.  
       *On exit:* **e** is overwritten with intermediate results.
- 6:     **z**[ $dim$ ] – double *Output*  
       **Note:** the dimension,  $dim$ , of the array **z** must be at least  
            $\max(1, \mathbf{pdz} \times n)$  when **job** = Nag\_EigVecs;  
           1 when **job** = Nag\_DoNothing.  
       The  $(i, j)$ th element of the matrix  $Z$  is stored in  
           **z**[ $(j - 1) \times \mathbf{pdz} + i - 1$ ] when **order** = Nag\_ColMajor;  
           **z**[ $(i - 1) \times \mathbf{pdz} + j - 1$ ] when **order** = Nag\_RowMajor.  
       *On exit:* if **job** = Nag\_EigVecs, **z** is overwritten by the orthogonal matrix  $Z$  which contains the eigenvectors of  $T$ .  
       If **job** = Nag\_DoNothing, **z** is not referenced.
- 7:     **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\_EigVecs,  $\mathbf{pdz} \geq \max(1, n)$ ;  
           if **job** = Nag\_DoNothing,  $\mathbf{pdz} \geq 1$ .
- 8:     **fail** – NagError \* *Input/Output*  
       The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

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

**NE\_CONVERGENCE**

The algorithm failed to converge;  $\langle value \rangle$  eigenvectors did not converge.

**NE\_ENUM\_INT\_2**

On entry, **job** =  $\langle value \rangle$ , **pdz** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: if **job** = Nag\_EigVecs, **pdz**  $\geq \max(1, \mathbf{n})$ ;

if **job** = Nag\_DoNothing, **pdz**  $\geq 1$ .

**NE\_INT**

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq 0$ .

On entry, **pdz** =  $\langle value \rangle$ .

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 unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation for further information.

**7 Accuracy**

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

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

and  $\epsilon$  is the *machine precision*.

If  $\lambda_i$  is an exact eigenvalue and  $\tilde{\lambda}_i$  is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\epsilon\|T\|_2,$$

where  $c(n)$  is a modestly increasing function of  $n$ .

If  $z_i$  is the corresponding exact eigenvector, and  $\tilde{z}_i$  is the corresponding computed eigenvector, then the angle  $\theta(\tilde{z}_i, z_i)$  between them is bounded as follows:

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\epsilon\|T\|_2}{\min_{i \neq j} |\lambda_i - \lambda_j|}.$$

Thus the accuracy of a computed eigenvector depends on the gap between its eigenvalue and all the other eigenvalues.

**8 Parallelism and Performance**

nag\_dstevd (f08jcc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag\_dstevd (f08jcc) 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

There is no complex analogue of this function.

## 10 Example

This example computes all the eigenvalues and eigenvectors of the symmetric tridiagonal matrix  $T$ , where

$$T = \begin{pmatrix} 1.0 & 1.0 & 0.0 & 0.0 \\ 1.0 & 4.0 & 2.0 & 0.0 \\ 0.0 & 2.0 & 9.0 & 3.0 \\ 0.0 & 0.0 & 3.0 & 16.0 \end{pmatrix}.$$

### 10.1 Program Text

```
/* nag_dstevd (f08jcc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    Integer i, j, n, pdz, d_len, e_len;
    Integer exit_status = 0;
    NagError fail;
    Nag_JobType job;
    Nag_OrderType order;
    /* Arrays */
    char nag_job_arg[40];
    double *z = 0, *d = 0, *e = 0;

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

    INIT_FAIL(fail);

    printf("nag_dstevd (f08jcc) 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
    pdz = n;
    d_len = n;
    e_len = n - 1;

    /* Allocate memory */
    if (!(z = NAG_ALLOC(n * n, double)) ||
        !(d = NAG_ALLOC(d_len, double)) || !(e = NAG_ALLOC(e_len, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    /* Read T from data file */
    for (i = 0; i < d_len; ++i)
#ifdef _WIN32
        scanf_s("%lf", &d[i]);
#else
        scanf("%lf", &d[i]);
#endif
    for (i = 0; i < e_len; ++i)
#ifdef _WIN32
        scanf_s("%lf", &e[i]);
#else
        scanf("%lf", &e[i]);
#endif
    /* Read type of job to be performed */
#ifdef _WIN32
        scanf_s("%*[\n] ");
#else
        scanf("%*[\n] ");
#endif
#ifdef _WIN32
        scanf_s(" %39s%*[\n] ", nag_job_arg, (unsigned)_countof(nag_job_arg));
#else
        scanf(" %39s%*[\n] ", nag_job_arg);
#endif
    /* nag_enum_name_to_value (x04nac).
     * Converts NAG enum member name to value
     */
    job = (Nag_JobType) nag_enum_name_to_value(nag_job_arg);

    /* Calculate all the eigenvalues and eigenvectors of T using using */
    /* nag_dstevd (f08jcc) */
    nag_dstevd(order, job, n, d, e, z, pdz, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_dstevd (f08jcc).\n%s\n", 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(" %7.4lf", d[i]);
    printf("\n\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).\n%s\n", fail.message);
    }

```

```

        exit_status = 1;
        goto END;
    }
END:
    NAG_FREE(d);
    NAG_FREE(e);
    NAG_FREE(z);
    return exit_status;
}

```

## 10.2 Program Data

```

nag_dstevd (f08jcc) Example Program Data
  4                               :Value of n
  1.0  4.0  9.0  16.0
  1.0  2.0  3.0                  :End of t
  Nag_EigVecs                    :Value of job

```

## 10.3 Program Results

```

nag_dstevd (f08jcc) Example Program Results

```

```

Eigenvalues
  0.6476   3.5470   8.6578  17.1477

```

```

Eigenvectors
           1           2           3           4
1         1.0000         1.0000         1.0000         1.0000
2        -0.3524         2.5470         7.6578        16.1477
3         0.0908        -1.0769        17.3340       105.6521
4        -0.0177         0.2594        -7.0826       276.1742

```

---