NAG Library Function Document

nag_dstedc (f08jhc)

1 Purpose

nag_dstedc (f08jhc) computes all the eigenvalues and, optionally, all the eigenvectors of a real $n$ by $n$ symmetric tridiagonal matrix, or of a real full or banded symmetric matrix which has been reduced to tridiagonal form.

2 Specification

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

void nag_dstedc (Nag_OrderType order, Nag_ComputeEigVecsType compz,
                Integer n, double d[], double e[], double z[], Integer pdz,
                NagError *fail)
```

3 Description

nag_dstedc (f08jhc) computes all the eigenvalues and, optionally, the eigenvectors of a real symmetric tridiagonal matrix $T$. That is, the function computes the spectral factorization of $T$ given by

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

where $\Lambda$ is a diagonal matrix whose diagonal elements are the eigenvalues, $\lambda_i$, of $T$ and $Z$ is an orthogonal matrix whose columns are the eigenvectors, $z_i$, of $T$. Thus

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

The function may also be used to compute all the eigenvalues and vectors of a real full, or banded, symmetric matrix $A$ which has been reduced to tridiagonal form $T$ as

$$ A = QTQ^T, $$

where $Q$ is orthogonal. The spectral factorization of $A$ is then given by

$$ A = (QZ)\Lambda(QZ)^T. $$

In this case $Q$ must be formed explicitly and passed to nag_dstedc (f08jhc) in the array $z$, and the function called with compz = Nag_OrigEigVecs. Functions which may be called to form $T$ and $Q$ are

- full matrix
- full matrix, packed storage
- band matrix
- nag_dsytrd (f08fec) and nag_dorgtr (f08fbc)
- nag_dsyrq (f08gec) and nag_dopgtr (f08gfc)
- nag_dsbrd (f08hec), with vect = Nag_FormQ

When only eigenvalues are required then this function calls nag_dsterf (f08jfc) to compute the eigenvalues of the tridiagonal matrix $T$, but when eigenvectors of $T$ are also required and the matrix is not too small, then a divide and conquer method is used, which can be much faster than nag_dsteqr (f08jec), although more storage is required.

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: compz – Nag_ComputeEigVecsType
   
   **Input**
   
   On entry: indicates whether the eigenvectors are to be computed.
   
   compz = Nag_NotEigVecs
   
   Only the eigenvalues are computed (and the array z is not referenced).
   
   compz = Nag_OrigEigVecs
   
   The eigenvalues and eigenvectors of A are computed (and the array z must contain the matrix Q on entry).
   
   compz = Nag_TridiagEigVecs
   
   The eigenvalues and eigenvectors of T are computed (and the array z is initialized by the function).
   
   **Constraint:** compz = Nag_NotEigVecs, Nag_OrigEigVecs or Nag_TridiagEigVecs.

3: n – Integer
   
   **Input**
   
   On entry: n, the order of the symmetric tridiagonal matrix T.
   
   **Constraint:** n ≥ 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 diagonal elements of the tridiagonal matrix.
   
   On exit: if fail.code = NE_NOERROR, the eigenvalues in ascending order.

5: e[dim] – double
   
   **Input/Output**
   
   Note: the dimension, dim, of the array e must be at least max(1, n − 1).
   
   On entry: the subdiagonal elements of the tridiagonal matrix.
   
   On exit: e is overwritten.

6: z[dim] – double
   
   **Input/Output**
   
   Note: the dimension, dim, of the array z must be at least max(1, pdz × n) when compz = Nag_OrigEigVecs or Nag_TridiagEigVecs; 1 otherwise.

   If compz = Nag_OrigEigVecs then the(i, j)th element of the matrix Q is stored in
   
   \[ z[(j - 1) \times pdz + i - 1] \] when order = Nag_ColMajor;
   
   \[ z[(i - 1) \times pdz + j - 1] \] when order = Nag_RowMajor.

   On entry: if compz = Nag_OrigEigVecs, z must contain the orthogonal matrix Q used in the reduction to tridiagonal form.

   On exit: if compz = Nag_OrigEigVecs, z contains the orthonormal eigenvectors of the original symmetric matrix A, and if compz = Nag_TridiagEigVecs, z contains the orthonormal eigenvectors of the symmetric tridiagonal matrix T.
If \( \text{compz} = \text{Nag}_\text{NotEigVecs} \), \( z \) is not referenced.

7: \( \text{pdz} \) – Integer

\( \text{Input} \)

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

\( \text{Constraints:} \)

\( \text{if} \ \text{compz} = \text{Nag}_\text{OrigEigVecs} \text{ or Nag}_\text{TridiagEigVecs}, \ \text{pdz} \geq \text{max}(1, n); \)

\( \text{otherwise} \ \text{pdz} \geq 1. \)

8: \( \text{fail} \) – NagError *

\( \text{Input/Output} \)

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 \ Error Indicators and Warnings

\( \text{NE}_\text{ALLOC}_\text{FAIL} \)

Dynamic memory allocation failed.

See Section 3.2.1.2 in the Essential Introduction for further information.

\( \text{NE}_\text{BAD}_\text{PARAM} \)

On entry, argument \( \langle \text{value} \rangle \) had an illegal value.

\( \text{NE}_\text{CONVERGENCE} \)

The algorithm failed to compute an eigenvalue while working on the submatrix lying in rows and columns \( \langle \text{value} \rangle / (n + 1) \) through \( \langle \text{value} \rangle \text{ mod } (n + 1) \).

\( \text{NE}_\text{ENUM}_\text{INT}_2 \)

On entry, \( \text{compz} = \langle \text{value} \rangle, \ \text{pdz} = \langle \text{value} \rangle \text{ and } n = \langle \text{value} \rangle. \)

Constraint: if \( \text{compz} = \text{Nag}_\text{OrigEigVecs} \text{ or Nag}_\text{TridiagEigVecs}, \ \text{pdz} \geq \text{max}(1, n); \)

\( \text{otherwise} \ \text{pdz} \geq 1. \)

\( \text{NE}_\text{INT} \)

On entry, \( n = \langle \text{value} \rangle. \)

Constraint: \( n \geq 0. \)

On entry, \( \text{pdz} = \langle \text{value} \rangle. \)

Constraint: \( \text{pdz} > 0. \)

\( \text{NE}_\text{INTERNAL}_\text{ERROR} \)

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.

\( \text{NE}_\text{NO}_\text{LICENCE} \)

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 \((T + E)\), 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.

See Section 4.7 of Anderson et al. (1999) for further details. See also nag_ddisna (f08flc).

8 Parallelism and Performance

nag_dstedc (f08jhc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dstedc (f08jhc) 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

If only eigenvalues are required, the total number of floating-point operations is approximately proportional to \(n^2\). When eigenvectors are required the number of operations is bounded above by approximately the same number of operations as nag_dsteqf (f08jec), but for large matrices nag_dstedc (f08jhc) is usually much faster.

The complex analogue of this function is nag_zstedc (f08jvc).

10 Example

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

\[
A = \begin{pmatrix}
4.99 & 0.04 & 0.22 & 0 \\
0.04 & 1.05 & -0.79 & 1.04 \\
0.22 & -0.79 & -2.31 & -1.30 \\
0 & 1.04 & -1.30 & -0.43 \\
\end{pmatrix}.
\]

\(A\) is first reduced to tridiagonal form by a call to nag_dsbdtrd (f08hec).
10.1 Program Text

/* nag_dstedc (f08jhc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011.
*/

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

int main(void)
{
    /* Scalars */
    Integer i, j, k, kd, n, pdab, pdq;
    Integer exit_status = 0;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 0, *d = 0, *e = 0, *q = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_UploType uplo;
    NagError fail;

    #ifdef NAG_COLUMN_MAJOR
    #define AB_UPPER(I, J) ab[(J - 1) * pdab + k + I - J - 1]
    #define AB_LOWER(I, J) ab[(J - 1) * pdab + I - J]
    #define Q(I, J) q[(J - 1) * pdq + I - 1]
    order = Nag_ColMajor;
    #else
    #define AB_UPPER(I, J) ab[(I - 1) * pdab + J - I]
    #define AB_LOWER(I, J) ab[(I - 1) * pdab + k + J - I - 1]
    #define Q(I, J) q[(I - 1) * pdq + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dstedc (f08jhc) 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"%"NAG_IFMT"%[\n]", &n, &kd);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%[\n]", &n, &kd);
    #endif

    /* Read uplo */
    #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);

    pdab = kd + 1;
    #ifdef NAG_COLUMN_MAJOR
    pdq = n;
    
    #endif

    /* Nag Types */
    /* Arrays */
    /* Scalars */
}

/* Skip heading in data file */

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

int main(void)
{
    /* Scalars */
    Integer i, j, k, kd, n, pdab, pdq;
    Integer exit_status = 0;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 0, *d = 0, *e = 0, *q = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_UploType uplo;
    NagError fail;

    #ifdef NAG_COLUMN_MAJOR
    #define AB_UPPER(I, J) ab[(J - 1) * pdab + k + I - J - 1]
    #define AB_LOWER(I, J) ab[(J - 1) * pdab + I - J]
    #define Q(I, J) q[(J - 1) * pdq + I - 1]
    order = Nag_ColMajor;
    #else
    #define AB_UPPER(I, J) ab[(I - 1) * pdab + J - I]
    #define AB_LOWER(I, J) ab[(I - 1) * pdab + k + J - I - 1]
    #define Q(I, J) q[(I - 1) * pdq + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dstedc (f08jhc) 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"%"NAG_IFMT"%[\n]", &n, &kd);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%[\n]", &n, &kd);
    #endif

    /* Read uplo */
    #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);

    pdab = kd + 1;
    #ifdef NAG_COLUMN_MAJOR
    pdq = n;
    
    #endif

    /* Nag Types */
    /* Arrays */
    /* Scalars */
}

/* Skip heading in data file */

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

int main(void)
{
    /* Scalars */
    Integer i, j, k, kd, n, pdab, pdq;
    Integer exit_status = 0;
    /* Arrays */
    char nag_enum_arg[40];
    double *ab = 0, *d = 0, *e = 0, *q = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_UploType uplo;
    NagError fail;

    #ifdef NAG_COLUMN_MAJOR
    #define AB_UPPER(I, J) ab[(J - 1) * pdab + k + I - J - 1]
    #define AB_LOWER(I, J) ab[(J - 1) * pdab + I - J]
    #define Q(I, J) q[(J - 1) * pdq + I - 1]
    order = Nag_ColMajor;
    #else
    #define AB_UPPER(I, J) ab[(I - 1) * pdab + J - I]
    #define AB_LOWER(I, J) ab[(I - 1) * pdab + k + J - I - 1]
    #define Q(I, J) q[(I - 1) * pdq + J - 1]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dstedc (f08jhc) 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"%"NAG_IFMT"%[\n]", &n, &kd);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%[\n]", &n, &kd);
    #endif

    /* Read uplo */
    #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);

    pdab = kd + 1;
    #ifdef NAG_COLUMN_MAJOR
    pdq = n;
    
    #endif

    /* Nag Types */
    /* Arrays */
    /* Scalars */
}
else

pdq = n;
#endif

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

/* Read the upper or lower triangular part of the band matrix A
* from data file.
*/
k = k*d + 1;
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= MIN(n, i + kd); ++j)
#ifdef _WIN32
            scanf_s("%lf", &AB_UPPER(i, j));
#else
            scanf("%lf", &AB_UPPER(i, j));
#endif
#ifdef _WIN32
            scanf_s("%*[\n");
#else
            scanf("%*[\n");
#endif
#else if (uplo == Nag_Lower) {
    for (i = 1; i <= n; ++i)
        for (j = MAX(1, i - kd); j <= i; ++j)
#ifdef _WIN32
            scanf_s("%lf", &AB_LOWER(i, j));
#else
            scanf("%lf", &AB_LOWER(i, j));
#endif
#ifdef _WIN32
            scanf_s("%*[\n");
#else
            scanf("%*[\n");
#endif
#endif

/* nag_dsbtrd (f08hec). 
* Reduce A to tridiagonal form T = (Q**T)*A*Q, and form Q. 
*/
nag_dsbtrd(order, Nag_FormQ, uplo, n, kd, ab, pdab, d, e, q, pdq, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dsbtrd (f08hec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* nag_dstedc (f08jhc) 
* Calculate all the eigenvalues and eigenvectors of A, 
* from T and Q. 
*/
nag_dstedc(order, Nag_OrigEigVecs, n, d, e, q, pdq, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_dstedc (f08jhc).\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--)
        Q(i, j) = Q(i, j) / Q(1,j);

/* Print eigenvalues and eigenvectors */
printf("%s\n", "Eigenvalues");
for (i = 0; i < n; ++i)
    printf("%8.4f%s", d[i], (i+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, q, pdq, "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(ab);
NAG_FREE(d);
NAG_FREE(e);
NAG_FREE(q);
return exit_status;

#undef AB_UPPER
#undef AB_LOWER
#undef Q

10.2 Program Data

nag_dstedc (f08jhc) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
<th>kd</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

:Values of n and kd

<table>
<thead>
<tr>
<th>uplo</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nag_Upper</td>
</tr>
</tbody>
</table>

:Value of uplo

4.99  0.04  0.22
     1.05 -0.79  1.04
     -2.31 -1.30
     -0.43 :End of matrix A

10.3 Program Results

nag_dstedc (f08jhc) Example Program Results

Eigenvalues
-2.9943  -0.7000   1.9974   4.9969

Eigenvalues

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>-2.6092</td>
<td>-36.0739</td>
<td>71.4695</td>
<td>0.0020</td>
</tr>
<tr>
<td>-35.8180</td>
<td>-19.3048</td>
<td>-26.5971</td>
<td>0.0311</td>
</tr>
<tr>
<td>-17.1000</td>
<td>45.9991</td>
<td>44.8645</td>
<td>-0.0071</td>
</tr>
</tbody>
</table>