NAG Library Function Document

nag_dsteqr (f08jec)

1 Purpose

nag_dsteqr (f08jec) computes all the eigenvalues and, optionally, all the eigenvectors of a real symmetric tridiagonal matrix, or of a real symmetric matrix which has been reduced to tridiagonal form.

2 Specification

```
#include <nag.h>
#include <nagf08.h>
void nag_dsteqr (Nag_OrderType order, Nag_ComputeZType compz, Integer n,
     double d[], double e[], double z[], Integer pdz, NagError *fail)
```

3 Description

nag_dsteqr (f08jec) 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

$$T = Z\Lambda Z^{\mathrm{T}},$$

where Λ is a diagonal matrix whose diagonal elements are the eigenvalues λ_i, and Z is the orthogonal matrix whose columns are the eigenvectors z_i. Thus

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

The function may also be used to compute all the eigenvalues and eigenvectors of a real symmetric matrix A which has been reduced to tridiagonal form T:

$$\begin{aligned} A &= QTQ^{\mathrm{T}}, \text{ where } Q \text{ is orthogonal} \\ &= (QZ)\Lambda(QZ)^{\mathrm{T}}. \end{aligned}$$

In this case, the matrix Q must be formed explicitly and passed to nag_dsteqr (f08jec), which must be called with **compz** = Nag_UpdateZ. The functions which must be called to perform the reduction to tridiagonal form and form Q are:

full matrix	nag_dsytrd (f08fec) and nag_dorgtr (f08ffc)
full matrix, packed storage	nag_dsptrd (f08gec) and nag_dopgtr (f08gfc)
band matrix	nag_dsbtrd (f08hec) with **vect** = Nag_FormQ.

nag_dsteqr (f08jec) uses the implicitly shifted QR algorithm, switching between the QR and QL variants in order to handle graded matrices effectively (see Greenbaum and Dongarra (1980)). The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a factor ± 1.

If only the eigenvalues of T are required, it is more efficient to call nag_dsterf (f08jfc) instead. If T is positive definite, small eigenvalues can be computed more accurately by nag_dpteqr (f08jgc).

4 References

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

Greenbaum A and Dongarra J J (1980) Experiments with QR/QL methods for the symmetric triangular eigenproblem *LAPACK Working Note No. 17 (Technical Report CS-89-92)* University of Tennessee, Knoxville http://www.netlib.org/lapack/lawnspdf/lawn17.pdf

Parlett B N (1998) *The Symmetric Eigenvalue Problem* SIAM, Philadelphia

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_ComputeZType *Input*

On entry: indicates whether the eigenvectors are to be computed.

compz = Nag_NotZ
Only the eigenvalues are computed (and the array **z** is not referenced).

compz = Nag_UpdateZ
The eigenvalues and eigenvectors of A are computed (and the array **z** must contain the matrix Q on entry).

compz = Nag_InitZ
The eigenvalues and eigenvectors of T are computed (and the array **z** is initialized by the function).

Constraint: **compz** = Nag_NotZ, Nag_UpdateZ or Nag_InitZ.

3: **n** – Integer *Input*

On entry: n, the order of the matrix T.

Constraint: $\mathbf{n} \geq 0$.

4: **d**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **d** must be at least $\max(1, \mathbf{n})$.

On entry: the diagonal elements of the tridiagonal matrix T.

On exit: the n eigenvalues in ascending order, unless **fail.code** = NE_CONVERGENCE (in which case see Section 6).

5: **e**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **e** must be at least $\max(1, \mathbf{n} - 1)$.

On entry: the off-diagonal elements of the tridiagonal matrix T.

On exit: **e** is overwritten.

6: **z**$[dim]$ – double *Input/Output*

Note: the dimension, *dim*, of the array **z** must be at least **pdz** $\times$ **n** when **compz** = Nag_UpdateZ or Nag_InitZ.

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 entry: if **compz** = Nag_UpdateZ, **z** must contain the orthogonal matrix Q from the reduction to tridiagonal form.

If **compz** = Nag_InitZ, **z** must be allocated, but its contents need not be set.

If **compz** = Nag_NotZ, **z** is not referenced and may be **NULL**.

On exit: if **compz** = Nag_UpdateZ or Nag_InitZ, the n required orthonormal eigenvectors stored as columns of Z; the ith column corresponds to the ith eigenvalue, where $i = 1, 2, \ldots, n$, unless **fail.errnum** > 0.

z is not changed if **compz** = Nag_NotZ.

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 **compz** = Nag_UpdateZ or Nag_InitZ, **pdz** $\geq$ **n**;
if **compz** = Nag_NotZ, **z** may be **NULL**.

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 ⟨*value*⟩ had an illegal value.

NE_CONVERGENCE

The algorithm has failed to find all the eigenvalues after a total of $30 \times$ **n** iterations. In this case, **d** and **e** contain on exit the diagonal and off-diagonal elements, respectively, of a tridiagonal matrix orthogonally similar to T. ⟨*value*⟩ off-diagonal elements have not converged to zero.

NE_ENUM_INT_2

On entry, **compz** = ⟨*value*⟩, **pdz** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **compz** = Nag_UpdateZ or Nag_InitZ, **pdz** $\geq$ **n**.

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 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 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 $(T + E)$, where

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

and ϵ is the ***machine precision***.

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

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

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:

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

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

8 Parallelism and Performance

nag_dsteqr (f08jec) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dsteqr (f08jec) 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 typically about $24n^2$ if **compz** = Nag_NotZ and about $7n^3$ if **compz** = Nag_UpdateZ or Nag_InitZ, but depends on how rapidly the algorithm converges. When **compz** = Nag_NotZ, the operations are all performed in scalar mode; the additional operations to compute the eigenvectors when **compz** = Nag_UpdateZ or Nag_InitZ can be vectorized and on some machines may be performed much faster.

The complex analogue of this function is nag_zsteqr (f08jsc).

10 Example

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

$$T = \begin{pmatrix} -6.99 & -0.44 & 0.00 & 0.00 \\ -0.44 & 7.92 & -2.63 & 0.00 \\ 0.00 & -2.63 & 2.34 & -1.18 \\ 0.00 & 0.00 & -1.18 & 0.32 \end{pmatrix}.$$

See also the examples for nag_dorgtr (f08ffc), nag_dopgtr (f08gfc) or nag_dsbtrd (f08hec), which illustrate the use of this function to compute the eigenvalues and eigenvectors of a full or band symmetric matrix.

10.1 Program Text

```
/* nag_dsteqr (f08jec) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagf16.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer       i, j, k, n, pdz, d_len, e_len, rowinc;
  Integer       exit_status = 0;
  double        r;
  /* Arrays */
  double        *z = 0, *d = 0, *e = 0;
  /* Nag Types */
  NagError      fail;
  Nag_OrderType order;

#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_dsteqr (f08jec) 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
#ifdef NAG_COLUMN_MAJOR
  rowinc = 1;
#else
  rowinc = 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
  /* Calculate all the eigenvalues and eigenvectors of tridiagonal matrix T,
   * reduced from real symmetric matrix using nag_dsteqr (f08jec).
   */
  nag_dsteqr(order, Nag_InitZ, n, d, e, z, pdz, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dsteqr (f08jec).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
  }

  /* Normalize the eigenvectors */
  for(j=1; j<=n; j++) {
    nag_damax_val(n, &Z(1,j), rowinc, &k, &r, &fail);
    for (i=1; i<=n; i++)
      Z(i,j) = Z(i,j)/r;
  }

  /* Print eigenvalues and eigenvectors */
  printf(" Eigenvalues\n");
  for (i = 0; i < n; ++i)
    printf("  %7.4lf", d[i]);
  printf("\n\n");

  /* Print real general matrix Z using easy-to-use
   * nag_gen_real_mat_print (x04cac).
   */
  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;
  }

 END:
  NAG_FREE(d);
  NAG_FREE(e);
  NAG_FREE(z);
  return exit_status;
}
```

10.2 Program Data

```
nag_dsteqr (f08jec) Example Program Data
  4                              :Value of N
 -6.99   7.92   2.34   0.32
 -0.44  -2.63  -1.18           :End of matrix T
```

10.3 Program Results

```
nag_dsteqr (f08jec) Example Program Results

 Eigenvalues
  -7.0037  -0.4059   2.0028   8.9968

 Eigenvectors
         1       2       3       4
 1   1.0000 -0.0129 -0.0217 -0.0275
 2   0.0311  0.1936  0.4429  1.0000
 3   0.0089  0.6152  1.0000 -0.4048
 4   0.0014  1.0000 -0.7012  0.0551
```