NAG C Library Function Document

nag_dpteqr (f08jgc)

1 Purpose

nag_dpteqr (f08jgc) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric positive-definite tridiagonal matrix, or of a real symmetric positive-definite matrix which has been reduced to tridiagonal form.

2 Specification

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

3 Description

nag_dpteqr (f08jgc) computes all the eigenvalues, and optionally all the eigenvectors, of a real symmetric positive-definite tridiagonal matrix T. In other words, it can compute the spectral factorization of T as

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

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 positive-definite matrix A which has been reduced to tridiagonal form T:

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

In this case, the matrix Q must be formed explicitly and passed to nag_dpteqr (f08jgc), which is 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) + nag_dorgtr (f08ffc)
full matrix, packed storage	nag_dsptrd (f08gec) + nag_dopgtr (f08gfc)
band matrix	nag_dsbtrd (f08hec) with **vect** = **Nag_FormQ**.

nag_dpteqr (f08jgc) first factorizes T as LDL^T where L is unit lower bidiagonal and D is diagonal. It forms the bidiagonal matrix $B = LD^{\frac{1}{2}}$, and then calls nag_dbdsqr (f08mec) to compute the singular values of B which are the same as the eigenvalues of T. The method used by the function allows high relative accuracy to be achieved in the small eigenvalues of T. The eigenvectors are normalized so that $\|z_i\|_2 = 1$, but are determined only to within a factor ± 1.

4 References

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

5 Parameters

1: **order** – Nag_OrderType *Input*

On entry: the **order** parameter 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 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

Constraint: **order** = **Nag_RowMajor** or **Nag_ColMajor**.

2: **compz** – Nag_ComputeZType *Input*

On entry: indicates whether the eigenvectors are to be computed as follows:

if **compz** = **Nag_NotZ**, only the eigenvalues are computed (and the array **z** is not referenced);

if **compz** = **Nag_InitZ**, the eigenvalues and eigenvectors of T are computed (and the array **z** is initialised by the routine);

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

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 descending order, unless **fail** > 0, in which case the array is overwritten.

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: the array is overwritten.

6: **z**[dim] – double *Input/Output*

Note: the dimension, dim, of the array **z** must be at least
$\max(1, \mathbf{pdz} \times \mathbf{n})$ when **compz** = **Nag_UpdateZ** or **Nag_InitZ**;
1 when **compz** = **Nag_NotZ**.

If **order** = **Nag_ColMajor**, the (i, j)th element of the matrix Z is stored in **z**$[(j-1) \times$ **pdz** $+ i - 1]$ and if **order** = **Nag_RowMajor**, the (i, j)th element of the matrix Z is stored in **z**$[(i-1) \times$ **pdz** $+ j - 1]$.

On entry: if **compz** = **Nag_UpdateZ**, **z** must contain the orthogonal matrix Q from the reduction to tridiagonal form. If **compz** = **Nag_InitZ**, **z** need not be set.

On exit: if **compz** = **Nag_InitZ** or **Nag_UpdateZ**, 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** > 0.

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

7: **pdz** – Integer *Input*

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

Constraints:

if **compz** = **Nag_UpdateZ** or **Nag_InitZ**, **pdz** $\geq \max(1, \mathbf{n})$;

if **compz** = **Nag_NotZ**, **pdz** ≥ 1.

8: **fail** – NagError * *Output*

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

NE_INT

On entry, **n** = ⟨*value*⟩.
Constraint: **n** ≥ 0.

On entry, **pdz** = ⟨*value*⟩.
Constraint: **pdz** > 0.

NE_ENUM_INT_2

On entry, **compz** = ⟨*value*⟩, **n** = ⟨*value*⟩, **pdz** = ⟨*value*⟩.
Constraint: if **compz** = **Nag_UpdateZ** or **Nag_InitZ**, **pdz** $\geq \max(1, \mathbf{n})$;
if **compz** = **Nag_NotZ**, **pdz** ≥ 1.

On entry, **n** = ⟨*value*⟩, **compz** = ⟨*value*⟩, **pdz** = ⟨*value*⟩.
Constraint: if **compz** = **Nag_UpdateZ** or **Nag_InitZ**, **pdz** $\geq \max(1, \mathbf{n})$;
if **compz** = **Nag_NotZ**, **pdz** ≥ 1.

NE_CONVERGENCE

The leading minor of order ⟨*value*⟩ is not positive-definite and the Cholesky factorization of T could not be completed. Hence T itself is not positive-definite.

The algorithm to compute the singular values of the Cholesky factor B failed to converge; ⟨*value*⟩ off-diagonal elements did not converge to zero.

NE_ALLOC_FAIL

Memory allocation failed.

NE_BAD_PARAM

On entry, parameter ⟨*value*⟩ had an illegal value.

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 consult NAG for assistance.

7 Accuracy

The eigenvalues and eigenvectors of T are computed to high relative accuracy which means that if they vary widely in magnitude, then any small eigenvalues (and corresponding eigenvectors) will be computed more accurately than, for example, with the standard QR method. However, the reduction to tridiagonal form (prior to calling the function) may exclude the possibility of obtaining high relative accuracy in the small eigenvalues of the original matrix if its eigenvalues vary widely in magnitude.

To be more precise, let H be the tridiagonal matrix defined by $H = DTD$, where D is diagonal with $d_{ii} = t_{ii}^{-\frac{1}{2}}$, and $h_{ii} = 1$ for all i. If λ_i is an exact eigenvalue of T and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq c(n)\epsilon\kappa_2(H)\lambda_i$$

where $c(n)$ is a modestly increasing function of n, ϵ is the ***machine precision***, and $\kappa_2(H)$ is the condition number of H with respect to inversion defined by: $\kappa_2(H) = \|H\| \cdot \|H^{-1}\|$.

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

$$\theta(\tilde{z}_i, z_i) \leq \frac{c(n)\epsilon\kappa_2(H)}{relgap_i}$$

where $relgap_i$ is the relative gap between λ_i and the other eigenvalues, defined by

$$relgap_i = \min_{i \neq j} \frac{|\lambda_i - \lambda_j|}{(\lambda_i + \lambda_j)}.$$

8 Further Comments

The total number of floating-point operations is typically about $30n^2$ if **compz** = **Nag_NotZ** and about $6n^3$ 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_zpteqr (f08juc).

9 Example

To compute all the eigenvalues and eigenvectors of the symmetric positive-definite tridiagonal matrix T, where

$$T = \begin{pmatrix} 4.16 & 3.17 & 0.00 & 0.00 \\ 3.17 & 5.25 & -0.97 & 0.00 \\ 0.00 & -0.97 & 1.09 & 0.55 \\ 0.00 & 0.00 & 0.55 & 0.62 \end{pmatrix}.$$

9.1 Program Text

```
/* nag_dpteqr (f08jgc) Example Program.
 *
 * Copyright 2001 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

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

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

  INIT_FAIL(fail);
  Vprintf("f08jgc Example Program Results\n\n");

  /* Skip heading in data file */
```

```
  Vscanf("%*[^\n] ");
  Vscanf("%ld%*[^\n] ", &n);
  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)) )
    {
      Vprintf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }
  /* Read T from data file */
  for (i = 0; i < d_len; ++i)
    Vscanf("%lf", &d[i]);
  for (i = 0; i < e_len; ++i)
    Vscanf("%lf", &e[i]);
  /* Calculate all the eigenvalues and eigenvectors of T */
  f08jgc(order, Nag_InitZ, n, d, e, z, pdz, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from f08jgc.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Print eigenvalues and eigenvectors */
  Vprintf(" Eigenvalues\n");
  for (i = 0; i < n; ++i)
    Vprintf("  %7.4lf",d[i]);
  Vprintf("\n\n");
  x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
         z, pdz, "Eigenvectors", 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      Vprintf("Error from x04cac.\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
 END:
  if (d) NAG_FREE(d);
  if (e) NAG_FREE(e);
  if (z) NAG_FREE(z);
  return exit_status;
}
```

9.2 Program Data

```
f08jgc Example Program Data
  4                              :Value of N
  4.16   5.25   1.09   0.62
  3.17  -0.97   0.55           :End of matrix T
```

9.3 Program Results

```
f08jgc Example Program Results

 Eigenvalues
   8.0023   1.9926   1.0014   0.1237

 Eigenvectors
          1        2        3        4
 1   0.6326   0.6245  -0.4191   0.1847
 2   0.7668  -0.4270   0.4176  -0.2352
 3  -0.1082   0.6071   0.4594  -0.6393
 4  -0.0081   0.2432   0.6625   0.7084
```