NAG Library Function Document

nag_dstegr (f08jlc)

1 Purpose

nag_dstegr (f08jlc) computes all the eigenvalues and, optionally, all the eigenvectors of a real n by n symmetric tridiagonal matrix.

2 Specification

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

```
void nag_dstegr (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
     Integer n, double d[], double e[], double vl, double vu, Integer il,
     Integer iu, Integer *m, double w[], double z[], Integer pdz,
     Integer isuppz[], NagError *fail)
```

3 Description

nag_dstegr (f08jlc) 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

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

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

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

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 explicitly applied to the output matrix Z. The functions which must be called to perform the reduction to tridiagonal form and apply Q are:

full matrix	nag_dsytrd (f08fec) and nag_dormtr (f08fgc)
full matrix, packed storage	nag_dsptrd (f08gec) and nag_dopmtr (f08ggc)
band matrix	nag_dsbtrd (f08hec) with **vect** = Nag_FormQ and nag_dgemm (f16yac).

This function uses the dqds and the Relatively Robust Representation algorithms to compute the eigenvalues and eigenvectors respectively; see for example Parlett and Dhillon (2000) and Dhillon and Parlett (2004) for further details. nag_dstegr (f08jlc) can usually compute all the eigenvalues and eigenvectors in $O(n^2)$ floating-point operations and so, for large matrices, is often considerably faster than the other symmetric tridiagonal functions in this chapter when all the eigenvectors are required, particularly so compared to those functions that are based on the QR algorithm.

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

Dhillon I S and Parlett B N (2004) Orthogonal eigenvectors and relative gaps. *SIAM J. Appl. Math.* **25** 858–899

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: indicates which eigenvalues should be returned.

range = Nag_AllValues
All eigenvalues will be found.

range = Nag_Interval
All eigenvalues in the half-open interval (**vl**, **vu**] will be found.

range = Nag_Indices
The **il**th through **iu**th eigenvectors will be found.

Constraint: **range** = Nag_AllValues, Nag_Interval or Nag_Indices.

4: **n** – Integer *Input*

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

Constraint: **n** ≥ 0.

5: **d**[*dim*] – double *Input/Output*

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

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

On exit: **d** is overwritten.

6: **e**[*dim*] – double *Input/Output*

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

On entry: **e**[0] to **e**[**n** − 2] are the subdiagonal elements of the tridiagonal matrix T. **e**[**n** − 1] need not be set.

On exit: **e** is overwritten.

7: **vl** – double *Input*

8: **vu** – double *Input*

On entry: if **range** = Nag_Interval, **vl** and **vu** contain the lower and upper bounds respectively 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, **il** and **iu** contains the indices (in ascending order) of the smallest and largest eigenvalues to be returned, respectively.

If **range** = Nag_AllValues or Nag_Interval, **il** and **iu** are not referenced.

Constraints:

if **range** = Nag_Indices and **n** > 0, $1 \leq$ **il** $\leq$ **iu** $\leq$ **n**;
if **range** = Nag_Indices and **n** = 0, **il** = 1 and **iu** = 0.

11: **m** – Integer * *Output*

On exit: the total number of eigenvalues found. $0 \leq$ **m** $\leq$ **n**.

If **range** = Nag_AllValues, **m** = **n**.

If **range** = Nag_Indices, **m** = **iu** − **il** + 1.

12: **w**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **w** must be at least max(1, **n**).

On exit: the eigenvalues in ascending order.

13: **z**[*dim*] – double *Output*

Note: the dimension, *dim*, of the array **z** must be at least

max(1, **pdz** × **n**) when **job** = Nag_DoBoth;
1 otherwise.

The (i, j)th element of the matrix Z is stored in

z[$(j-1)$ × **pdz** + $i - 1$] when **order** = Nag_ColMajor;
z[$(i-1)$ × **pdz** + $j - 1$] when **order** = Nag_RowMajor.

On exit: if **job** = Nag_DoBoth, then if **fail.code** = NE_NOERROR, the columns of **z** contain the orthonormal eigenvectors of the matrix T, with the ith column of Z holding the eigenvector associated with **w**[$i - 1$].

If **job** = Nag_EigVals, **z** is not referenced.

14: **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, **pdz** $\geq$ max(1, **n**);
otherwise **pdz** ≥ 1.

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

Note: the dimension, *dim*, of the array **isuppz** must be at least max(1, 2 × **m**).

On exit: the support of the eigenvectors in Z, i.e., the indices indicating the nonzero elements in Z. The ith eigenvector is nonzero only in elements **isuppz**$[2 \times i - 2]$ through **isuppz**$[2 \times i - 1]$.

16: **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_CONVERGENCE

Inverse iteration failed to converge.

The dqds algorithm failed to converge.

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** ≥ 1.

NE_ENUM_INT_3

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

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** ≥ 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.

7 Accuracy

See Section 4.7 of Anderson *et al.* (1999) and Barlow and Demmel (1990) for further details.

8 Parallelism and Performance

nag_dstegr (f08jlc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dstegr (f08jlc) 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 required to compute all the eigenvalues and eigenvectors is approximately proportional to n^2.

The complex analogue of this function is nag_zstegr (f08jyc).

10 Example

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

$$T = \begin{pmatrix} 1.0 & 1.0 & 0 & 0 \\ 1.0 & 4.0 & 2.0 & 0 \\ 0 & 2.0 & 9.0 & 3.0 \\ 0 & 0 & 3.0 & 16.0 \end{pmatrix}.$$

10.1 Program Text

```
/* nag_dstegr (f08jlc) 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        vl = 0.0, vu = 0.0;
  Integer       i, j, m, n, pdz;
  Integer       exit_status = 0;
  /* Arrays */
  char          nag_enum_arg[40];
  double        *d = 0, *e = 0, *w = 0, *z = 0;
  Integer       *isuppz = 0;
  /* Nag Types */
  Nag_OrderType order;
  Nag_JobType   job;
  Nag_RangeType range;
  NagError      fail;

#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_dstegr (f08jlc) Example Program Results\n\n");

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

```
  scanf("%*[^\n]");
  scanf("%ld%*[^\n]", &n);
  m = n;

  /* Read job and range*/
  scanf("%39s%*[^\n]", nag_enum_arg);
  /* nag_enum_name_to_value (x04nac).
   * Converts NAG enum member name to value.
   */
  job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);
  scanf("%39s%*[^\n]", nag_enum_arg);
  range = (Nag_RangeType) nag_enum_name_to_value(nag_enum_arg);

#ifdef NAG_COLUMN_MAJOR
  pdz = n;
#else
  pdz = m;
#endif

  /* Allocate memory */
  if (!(d = NAG_ALLOC(n, double)) ||
      !(e = NAG_ALLOC(n, double)) ||
      !(w = NAG_ALLOC(n, double)) ||
      !(z = NAG_ALLOC(n*m, double)) ||
      !(isuppz = NAG_ALLOC(2*m, Integer)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read the symmetric tridiagonal matrix T from data file, first
   * the diagonal elements, then the off diagonal elements.
   */
  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]");

  /* nag_dstegr (f08jlc).
   * Calculate all the eigenvalues of T.
   */
  nag_dstegr(order, job, range, n, d, e, vl, vu, 0, 0, &m, w, z,
             pdz, isuppz, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_dstegr (f08jlc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* Normalize the eigenvectors */
  for(j=1; j<=m; j++)
    for(i=n; i>=1; i--)
      Z(i, j) = Z(i, j) / Z(1,j);

  /* Print eigenvalues and eigenvectors */
  printf(" %s\n", "Eigenvalues");
  for (i = 0; i < m; ++i)
    printf("%8.4f%s", w[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, m,
                         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(w);
  NAG_FREE(z);
  NAG_FREE(isuppz);

  return exit_status;
}

#undef Z
```

10.2 Program Data

```
nag_dstegr (f08jlc) Example Program Data

  4                    :Value of n
  Nag_DoBoth           :Value of job
  Nag_AllValues        :Value of range

  1.0  4.0  9.0  16.0  :End of d
  1.0  2.0  3.0        :End of e
```

10.3 Program Results

```
nag_dstegr (f08jlc) 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
```