NAG Library Function Document

nag_dstev (f08jac)

1 Purpose

nag_dstev (f08jac) computes all the eigenvalues and, optionally, all the eigenvectors of a real n by n symmetric tridiagonal matrix A.

2 Specification

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

3 Description

nag_dstev (f08jac) computes all the eigenvalues and, optionally, all the eigenvectors of A using a combination of the QR and QL algorithms, with an implicit shift.

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

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_EigVals
Only eigenvalues are computed.

job = Nag_DoBoth
Eigenvalues and eigenvectors are computed.

Constraint: **job** = Nag_EigVals or Nag_DoBoth.

3: **n** – Integer *Input*

On entry: n, the order of the matrix.

Constraint: **n** ≥ 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 n diagonal elements of the tridiagonal matrix A.

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 $\max(1, \mathbf{n} - 1)$.

On entry: the $(n - 1)$ subdiagonal elements of the tridiagonal matrix A.

On exit: the contents of **e** are destroyed.

6: **z**[dim] – double *Output*

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

$\max(1, \mathbf{pdz} \times \mathbf{n})$ when **job** = Nag_DoBoth;
1 otherwise.

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_DoBoth, then if **fail**.**code** = NE_NOERROR, **z** contains the orthonormal eigenvectors of the matrix A, with the ith column of Z holding the eigenvector associated with **d**$[i - 1]$.

If **job** = Nag_EigVals, **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_DoBoth, **pdz** $\geq \max(1, \mathbf{n})$;
otherwise **pdz** ≥ 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 3.2.1.2 in How to Use the NAG Library and its Documentation for further information.

NE_BAD_PARAM

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

NE_CONVERGENCE

The algorithm failed to converge; ⟨*value*⟩ off-diagonal elements of **e** did not converge to zero.

NE_ENUM_INT_2

On entry, **job** = ⟨*value*⟩, **pdz** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **job** = Nag_DoBoth, **pdz** ≥ max(1, **n**);
otherwise **pdz** ≥ 1.

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 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 3.6.5 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 $(A + E)$, where

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

and ϵ is the ***machine precision***. See Section 4.7 of Anderson *et al.* (1999) for further details.

8 Parallelism and Performance

nag_dstev (f08jac) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dstev (f08jac) 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' Notefor your implementation for any additional implementation-specific information.

9 Further Comments

The total number of floating-point operations is proportional to n^2 if **job** = Nag_EigVals and is proportional to n^3 if **job** = Nag_DoBoth.

10 Example

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

$$A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 4 & 2 & 0 \\ 0 & 2 & 9 & 3 \\ 0 & 0 & 3 & 16 \end{pmatrix},$$

together with approximate error bounds for the computed eigenvalues and eigenvectors.

10.1 Program Text

```
/* nag_dstev (f08jac) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <math.h>
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx02.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  double eerrbd, eps;
  Integer exit_status = 0, i, j, n, pdz;
  /* Arrays */
  double *d = 0, *e = 0, *rcondz = 0, *z = 0, *zerrbd = 0;
  /* Nag Types */
  Nag_OrderType order;
  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_dstev (f08jac) 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

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

  pdz = n;
  /* Read the diagonal and off-diagonal elements of the matrix A
   * from data file.
   */
  for (i = 0; i < n; ++i)
#ifdef _WIN32
```

```
    scanf_s("%lf", &d[i]);
#else
    scanf("%lf", &d[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  for (i = 0; i < n - 1; ++i)
#ifdef _WIN32
    scanf_s("%lf", &e[i]);
#else
    scanf("%lf", &e[i]);
#endif
#ifdef _WIN32
  scanf_s("%*[^\n]");
#else
  scanf("%*[^\n]");
#endif

  /* nag_dstev (f08jac).
   * Solve the symmetric tridiagonal eigenvalue problem.
   */
  nag_dstev(order, Nag_DoBoth, n, d, e, z, pdz, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_dstev (f08jac).\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 solution */
  printf("Eigenvalues\n");
  for (i = 0; i < n; ++i)
    printf("%8.4f%s", d[i], (i + 1) % 8 == 0 ? "\n" : " ");
  printf("\n");

  /* nag_gen_real_mat_print (x04cac).
   * Print eigenvectors.
   */
  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;
  }

  /* Get the machine precision, eps, using nag_machine_precision (X02AJC)
   * and compute the approximate error bound for the computed eigenvalues.
   * Note that for the 2-norm, ||A|| = max {|d[i]|, i=0..n-1}, and since
   * the eigenvalues are in ascending order ||A|| = max( |d[0]|, |d[n-1]|).
   */
  eps = nag_machine_precision;
  eerrbd = eps * MAX(fabs(d[0]), fabs(d[n - 1]));

  /* nag_ddisna (f08flc).
   * Estimate reciprocal condition numbers for the eigenvectors.
   */
  nag_ddisna(Nag_EigVecs, n, n, d, rcondz, &fail);
  if (fail.code != NE_NOERROR) {
    printf("Error from nag_ddisna (f08flc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
```

```
  }

  /* Compute the error estimates for the eigenvectors */
  for (i = 0; i < n; ++i)
    zerrbd[i] = eerrbd / rcondz[i];

  /* Print the approximate error bounds for the eigenvalues and vectors */
  printf("\nError estimate for the eigenvalues\n");
  printf("%11.1e\n", eerrbd);
  printf("\nError estimates for the eigenvectors\n");
  for (i = 0; i < n; ++i)
    printf("%11.1e%s", zerrbd[i], (i + 1) % 6 == 0 ? "\n" : " ");

END:
  NAG_FREE(d);
  NAG_FREE(e);
  NAG_FREE(rcondz);
  NAG_FREE(z);
  NAG_FREE(zerrbd);

  return exit_status;
}

#undef Z
```

10.2 Program Data

```
nag_dstev (f08jac) Example Program Data

  4                    :Value of n

  1.0  4.0  9.0  16.0  :End of diagonal elements
  1.0  2.0  3.0        :End of off-diagonal elements
```

10.3 Program Results

```
nag_dstev (f08jac) 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

Error estimate for the eigenvalues
    1.9e-15

Error estimates for the eigenvectors
    6.6e-16     6.6e-16     3.7e-16     2.2e-16
```