NAG Library Function Document

nag_zstedc (f08jvc)

1 Purpose

nag_zstedc (f08jvc) computes all the eigenvalues and, optionally, all the eigenvectors of a real n by n symmetric tridiagonal matrix, or of a complex full or banded Hermitian matrix which has been reduced to tridiagonal form.

2 Specification

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

3 Description

nag_zstedc (f08jvc) 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

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

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

$$A = QTQ^{\mathrm{H}},$$

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

$$A = (QZ)\Lambda(QZ)^{\mathrm{H}}.$$

In this case Q must be formed explicitly and passed to nag_zstedc (f08jvc) 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	nag_zhetrd (f08fsc) and nag_zungtr (f08ftc)
full matrix, packed storage	nag_zhptrd (f08gsc) and nag_zupgtr (f08gtc)
band matrix	nag_zhbtrd (f08hsc), 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_zsteqr (f08jsc), although more storage is required.

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

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.

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 subdiagonal elements of the tridiagonal matrix.

On exit: **e** is overwritten.

6: **z**[dim] – Complex *Input/Output*

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

$\max(1, \mathbf{pdz} \times \mathbf{n})$ 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$[(j-1) \times \mathbf{pdz} + i - 1]$ when **order** = Nag_ColMajor;
z$[(i-1) \times \mathbf{pdz} + j - 1]$ when **order** = Nag_RowMajor.

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

On exit: if **compz** = Nag_OrigEigVecs, **z** contains the orthonormal eigenvectors of the original Hermitian matrix A, and if **compz** = Nag_TridiagEigVecs, **z** contains the orthonormal eigenvectors of the symmetric tridiagonal matrix T.

If **compz** = Nag_NotEigVecs, **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 **compz** = Nag_OrigEigVecs or Nag_TridiagEigVecs, **pdz** $\geq \max(1, \mathbf{n})$;
otherwise **pdz** ≥ 1.

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 failed to compute an eigenvalue while working on the submatrix lying in rows and columns ⟨*value*⟩/(**n** + 1) through ⟨*value*⟩ mod (**n** + 1).

NE_ENUM_INT_2

On entry, **compz** = ⟨*value*⟩, **pdz** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **compz** = Nag_OrigEigVecs or Nag_TridiagEigVecs, **pdz** $\geq \max(1, \mathbf{n})$;
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 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.

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

8 Parallelism and Performance

nag_zstedc (f08jvc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zstedc (f08jvc) 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_zsteqr (f08jsc), but for large matrices nag_zstedc (f08jvc) is usually much faster.

The real analogue of this function is nag_dstedc (f08jhc).

10 Example

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

$$A = \begin{pmatrix} -3.13 & 1.94 - 2.10i & -3.40 + 0.25i & 0 \\ 1.94 + 2.10i & -1.91 & -0.82 - 0.89i & -0.67 + 0.34i \\ -3.40 - 0.25i & -0.82 + 0.89i & -2.87 & -2.10 - 0.16i \\ 0 & -0.67 - 0.34i & -2.10 + 0.16i & 0.50 \end{pmatrix}.$$

A is first reduced to tridiagonal form by a call to nag_zhbtrd (f08hsc).

10.1 Program Text

```
/* nag_zstedc (f08jvc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer       i, j, k, kd, n;
  Integer       exit_status = 0;
  Integer       pdab, pdq;
  /* Arrays */
  char          nag_enum_arg[40];
  Complex       *ab = 0, *q = 0;
  double        *d = 0, *e = 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_zstedc (f08jvc) 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;
```

```
  pdq = n;

  /* Allocate memory */
  if (!(ab = NAG_ALLOC((kd+1)*n, Complex)) ||
      !(q = NAG_ALLOC(n*n, Complex)) ||
      !(d = NAG_ALLOC(n, double)) ||
      !(e = NAG_ALLOC(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 = kd + 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 , %lf )", &AB_UPPER(i, j).re, &AB_UPPER(i, j).im);
#else
        scanf(" ( %lf , %lf )", &AB_UPPER(i, j).re, &AB_UPPER(i, j).im);
#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 , %lf )", &AB_LOWER(i, j).re, &AB_LOWER(i, j).im);
#else
          scanf(" ( %lf , %lf )", &AB_LOWER(i, j).re, &AB_LOWER(i, j).im);
#endif
#ifdef _WIN32
      scanf_s("%*[^\n]");
#else
      scanf("%*[^\n]");
#endif
    }

  /* nag_zhbtrd (f08hsc).
   * Reduce A to tridiagonal form T = (Q**T)*A*Q, and form Q.
   */
  nag_zhbtrd(order, Nag_FormQ, uplo, n, kd, ab, pdab, d, e, q, pdq, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zhbtrd (f08hsc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_zstedc (f08jvc).
   * Calculate all the eigenvalues and eigenvectors of A,
   * from T and Q.
   */
  nag_zstedc(order, Nag_OrigEigVecs, n, d, e, q, pdq, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zstedc (f08jvc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }

  /* nag_complex_divide (a02cdc).
   * Normalize the eigenvectors.
```

```
   */
  for(j=1; j<=n; j++)
    for(i=n; i>=1; i--)
      Q(i, j) = nag_complex_divide(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)%4 == 0?"\n":" ");
  printf("\n");

  /* nag_gen_complx_mat_print_comp (x04dbc).
   * Print eigenvectors.
   */
  fflush(stdout);
  nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag,
                                n, n, q, pdq, Nag_BracketForm, "%7.4f",
                                "Eigenvectors", Nag_IntegerLabels, 0,
                                Nag_IntegerLabels, 0, 80, 0, 0, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_gen_complx_mat_print_comp (x04dbc).\n%s\n",
             fail.message);
      exit_status = 1;
      goto END;
    }

 END:
  NAG_FREE(ab);
  NAG_FREE(q);
  NAG_FREE(d);
  NAG_FREE(e);

  return exit_status;
}

#undef AB_UPPER
#undef AB_LOWER
#undef Q
```

10.2 Program Data

```
nag_zstedc (f08jvc) Example Program Data

    4              2                                        :Values of n and kd
    Nag_Upper                                               :Value of uplo

 ( -3.13 , 0.00) (  1.94, -2.10) ( -3.40,  0.25)
                 ( -1.91,  0.00) ( -0.82, -0.89) ( -0.67,  0.34)
                                 ( -2.87,  0.00) ( -2.10, -0.16)
                                                 (  0.50,  0.00) :End matrix A
```

10.3 Program Results

```
nag_zstedc (f08jvc) Example Program Results

Eigenvalues
 -7.0042  -4.0038   0.5968   3.0012

 Eigenvectors
                     1                  2                  3                  4
 1  ( 1.0000, 0.0000)  ( 1.0000,-0.0000)  ( 1.0000,-0.0000)  ( 1.0000,-0.0000)
 2  (-0.2268,-0.2805)  (-2.2857,-1.6226)  ( 1.0765, 0.5028)  ( 0.4873, 0.7267)
 3  ( 0.8338, 0.0413)  (-2.0739, 0.3334)  (-0.1427,-0.3885)  (-1.0790, 0.0343)
 4  ( 0.2267,-0.0415)  (-1.1727,-0.1848)  (-1.9460, 0.9305)  ( 0.8719,-0.3587)
```