NAG Library Function Document

nag_zhbtrd (f08hsc)

1 Purpose

nag_zhbtrd (f08hsc) reduces a complex Hermitian band matrix to tridiagonal form.

2 Specification

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

```
void nag_zhbtrd (Nag_OrderType order, Nag_VectType vect, Nag_UploType uplo,
     Integer n, Integer kd, Complex ab[], Integer pdab, double d[],
     double e[], Complex q[], Integer pdq, NagError *fail)
```

3 Description

nag_zhbtrd (f08hsc) reduces a Hermitian band matrix A to real symmetric tridiagonal form T by a unitary similarity transformation:

$$T = Q^{\mathrm{H}} AQ.$$

The unitary matrix Q is determined as a product of Givens rotation matrices, and may be formed explicitly by the function if required.

The function uses a vectorizable form of the reduction, due to Kaufman (1984).

4 References

Kaufman L (1984) Banded eigenvalue solvers on vector machines *ACM Trans. Math. Software* **10** 73–86

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: **vect** – Nag_VectType *Input*

On entry: indicates whether Q is to be returned.

vect = Nag_FormQ
Q is returned.

vect = Nag_UpdateQ
Q is updated (and the array **q** must contain a matrix on entry).

vect = Nag_DoNotForm
Q is not required.

Constraint: **vect** = Nag_FormQ, Nag_UpdateQ or Nag_DoNotForm.

3: **uplo** – Nag_UploType *Input*

On entry: indicates whether the upper or lower triangular part of A is stored.

uplo = Nag_Upper
The upper triangular part of A is stored.

uplo = Nag_Lower
The lower triangular part of A is stored.

Constraint: **uplo** = Nag_Upper or Nag_Lower.

4: **n** – Integer *Input*

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

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

5: **kd** – Integer *Input*

On entry: if **uplo** = Nag_Upper, the number of superdiagonals, k_d, of the matrix A.

If **uplo** = Nag_Lower, the number of subdiagonals, k_d, of the matrix A.

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

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

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

On entry: the upper or lower triangle of the n by n Hermitian band matrix A.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of A_{ij}, depends on the **order** and **uplo** arguments as follows:

if **order** = Nag_ColMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ab}[k_d + i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \dots, n$ and $i = \max(1, j - k_d), \dots, j$;
if **order** = Nag_ColMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ab}[i - j + (j-1) \times \mathbf{pdab}]$, for $j = 1, \dots, n$ and $i = j, \dots, \min(n, j + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Upper,
A_{ij} is stored in $\mathbf{ab}[j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \dots, n$ and $j = i, \dots, \min(n, i + k_d)$;
if **order** = Nag_RowMajor and **uplo** = Nag_Lower,
A_{ij} is stored in $\mathbf{ab}[k_d + j - i + (i-1) \times \mathbf{pdab}]$, for $i = 1, \dots, n$ and $j = \max(1, i - k_d), \dots, i$.

On exit: **ab** is overwritten by values generated during the reduction to tridiagonal form.

The first superdiagonal or subdiagonal and the diagonal of the tridiagonal matrix T are returned in **ab** using the same storage format as described above.

7: **pdab** – Integer *Input*

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

Constraint: $\mathbf{pdab} \geq \max(1, \mathbf{kd} + 1)$.

8: **d**[**n**] – double *Output*

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

9: **e**[**n** − **1**] – double *Output*

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

10: **q**[*dim*] – Complex *Input/Output*

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

max(1, **pdq** × **n**) when **vect** = Nag_FormQ or Nag_UpdateQ;
1 when **vect** = Nag_DoNotForm.

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

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

On entry: if **vect** = Nag_UpdateQ, **q** must contain the matrix formed in a previous stage of the reduction (for example, the reduction of a banded Hermitian-definite generalized eigenproblem); otherwise **q** need not be set.

On exit: if **vect** = Nag_FormQ or Nag_UpdateQ, the n by n matrix Q.

If **vect** = Nag_DoNotForm, **q** is not referenced.

11: **pdq** – Integer *Input*

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

Constraints:

if **vect** = Nag_FormQ or Nag_UpdateQ, **pdq** ≥ max(1, **n**);
if **vect** = Nag_DoNotForm, **pdq** ≥ 1.

12: **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_ENUM_INT_2

On entry, **vect** = ⟨*value*⟩, **pdq** = ⟨*value*⟩ and **n** = ⟨*value*⟩.
Constraint: if **vect** = Nag_FormQ or Nag_UpdateQ, **pdq** ≥ max(1, **n**);
if **vect** = Nag_DoNotForm, **pdq** ≥ 1.

NE_INT

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

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

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

On entry, **pdq** $= \langle value \rangle$.
Constraint: **pdq** > 0.

NE_INT_2

On entry, **pdab** $= \langle value \rangle$ and **kd** $= \langle value \rangle$.
Constraint: **pdab** $\geq \max(1, \mathbf{kd} + 1)$.

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 tridiagonal matrix T is exactly similar to a nearby matrix $(A + E)$, where

$$\|E\|_2 \leq c(n)\epsilon\|A\|_2,$$

$c(n)$ is a modestly increasing function of n, and ϵ is the ***machine precision***.

The elements of T themselves may be sensitive to small perturbations in A or to rounding errors in the computation, but this does not affect the stability of the eigenvalues and eigenvectors.

The computed matrix Q differs from an exactly unitary matrix by a matrix E such that

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

where ϵ is the ***machine precision***.

8 Parallelism and Performance

nag_zhbtrd (f08hsc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zhbtrd (f08hsc) 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 real floating-point operations is approximately $20n^2k$ if **vect** = Nag_DoNotForm with $10n^3(k-1)/k$ additional operations if **vect** = Nag_FormQ.

The real analogue of this function is nag_dsbtrd (f08hec).

10 Example

This example computes all the eigenvalues and eigenvectors of the matrix A, where

$$A = \begin{pmatrix} -3.13 + 0.00i & 1.94 - 2.10i & -3.40 + 0.25i & 0.00 + 0.00i \\ 1.94 + 2.10i & -1.91 + 0.00i & -0.82 - 0.89i & -0.67 + 0.34i \\ -3.40 - 0.25i & -0.82 + 0.89i & -2.87 + 0.00i & -2.10 - 0.16i \\ 0.00 + 0.00i & -0.67 - 0.34i & -2.10 + 0.16i & 0.50 + 0.00i \end{pmatrix}.$$

Here A is Hermitian and is treated as a band matrix. The program first calls nag_zhbtrd (f08hsc) to reduce A to tridiagonal form T, and to form the unitary matrix Q; the results are then passed to nag_zsteqr (f08jsc) which computes the eigenvalues and eigenvectors of A.

10.1 Program Text

```
/* nag_zhbtrd (f08hsc) Example Program.
 *
 * Copyright 2014 Numerical Algorithms Group.
 *
 * Mark 7, 2001.
 */

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

int main(void)
{
  /* Scalars */
  Integer       i, j, k, kd, n, pdab, pdz, d_len, e_len;
  Integer       exit_status = 0;
  NagError      fail;
  Nag_UploType  uplo;
  Nag_OrderType order;
  /* Arrays */
  char          nag_enum_arg[40];
  Complex       *ab = 0, *z = 0;
  double        *d = 0, *e = 0;

#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 Z(I, J) z[(J - 1) * pdz + 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 Z(I, J) z[(I - 1) * pdz + J - 1]
  order = Nag_RowMajor;
#endif

  INIT_FAIL(fail);

  printf("nag_zhbtrd (f08hsc) 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
```

```
  pdab = kd + 1;
  pdz = n;
  d_len = n;
  e_len = n - 1;

  /* Allocate memory */
  if (!(ab = NAG_ALLOC(pdab * n, Complex)) ||
      !(d = NAG_ALLOC(d_len, double)) ||
      !(e = NAG_ALLOC(e_len, double)) ||
      !(z = NAG_ALLOC(pdz * n, Complex)))
    {
      printf("Allocation failure\n");
      exit_status = -1;
      goto END;
    }

  /* Read A from data file */
#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);
  k = kd + 1;
  if (uplo == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
        {
          for (j = i; j <= MIN(i + kd, n); ++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
    {
      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
    }

  /* Reduce A to tridiagonal form */
  /* nag_zhbtrd (f08hsc).
   * Unitary reduction of complex Hermitian band matrix to
   * real symmetric tridiagonal form
   */
```

```
  nag_zhbtrd(order, Nag_FormQ, uplo, n, kd, ab, pdab, d, e,
             z, pdz, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zhbtrd (f08hsc).\n%s\n", fail.message);
      exit_status = 1;
      goto END;
    }
  /* Calculate all the eigenvalues and eigenvectors of A */
  /* nag_zsteqr (f08jsc).
   * All eigenvalues and eigenvectors of real symmetric
   * tridiagonal matrix, reduced from complex Hermitian
   * matrix, using implicit QL or QR
   */
  nag_zsteqr(order, Nag_UpdateZ, n, d, e, z, pdz, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zsteqr (f08jsc).\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) = nag_complex_divide(Z(i, j),Z(1, j));
        }
    }
  /* Print eigenvalues and eigenvectors */
  printf(" Eigenvalues\n");
  for (i = 1; i <= n; ++i)
    printf("%8.4f%s", d[i-1], i%8 == 0?"\n":"           ");
  printf("\n\n");
  /* nag_gen_complx_mat_print_comp (x04dbc).
   * Print complex general matrix (comprehensive)
   */
  fflush(stdout);
  nag_gen_complx_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                                n, z, pdz, 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(d);
  NAG_FREE(e);
  NAG_FREE(z);

  return exit_status;
}
```

10.2 Program Data

```
nag_zhbtrd (f08hsc) Example Program Data
  4  2                                                      :Values of n and kd
  Nag_Lower                                                 :Value of uplo
 (-3.13, 0.00)
 ( 1.94, 2.10) (-1.91, 0.00)
 (-3.40,-0.25) (-0.82, 0.89) (-2.87, 0.00)
               (-0.67,-0.34) (-2.10, 0.16) ( 0.50, 0.00)   :End of matrix A
```

10.3 Program Results

```
nag_zhbtrd (f08hsc) 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)
```