NAG C Library Function Document

nag_zhbtrd (f08hsc)

1 Purpose

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

2 Specification

```c
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

The Hermitian band matrix $A$ is reduced to real symmetric tridiagonal form $T$ by a unitary similarity transformation: $T = Q^H A Q$. 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 vectorisable form of the reduction, due to Kaufman (1984).

4 References


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: `vect` – Nag_VectType

*Input*

On entry: indicates whether $Q$ is to be returned as follows:

- if vect = Nag_FormQ, $Q$ is returned (and the array q must contain a matrix on entry);
- if vect = Nag_UpdateQ, $Q$ is updated (and the array q must contain a matrix on entry);
- if 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 as follows:

- if uplo = Nag_Upper, then the upper triangular part of $A$ is stored;
- if uplo = Nag_Lower, then the lower triangular part of $A$ is stored.

*Constraint:* uplo = Nag_Upper or Nag_Lower.
On entry: \( n \), the order of the matrix \( A \).

Constraint: \( n \geq 0 \).

5: \( k \) – Integer

On entry: \( k \), the number of super-diagonals of the matrix \( A \) if \( \text{uplo} = \text{Nag_Upper} \), or the number of sub-diagonals if \( \text{uplo} = \text{Nag_Lower} \).

Constraint: \( k \geq 0 \).

6: \( ab[\text{dim}] \) – Complex

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

On entry: the \( n \) by \( n \) Hermitian band matrix \( A \) with \( k \) sub or super-diagonals. This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. Just the upper or lower triangular part of the array is held depending on the value of \( \text{uplo} \). The storage of elements \( a_{ij} \) depends on the \( \text{order} \) and \( \text{uplo} \) parameters as follows:

- if \( \text{order} = \text{Nag_ColMajor} \) and \( \text{uplo} = \text{Nag_Upper} \),
  \( a_{ij} \) is stored in \( ab[k + i - j + (j - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and \( j = i, \ldots, \min(n, i + k) \);
- if \( \text{order} = \text{Nag_ColMajor} \) and \( \text{uplo} = \text{Nag_Lower} \),
  \( a_{ij} \) is stored in \( ab[i - j + (j - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and \( j = \max(1,i-k), \ldots, i \);
- if \( \text{order} = \text{Nag_RowMajor} \) and \( \text{uplo} = \text{Nag_Upper} \),
  \( a_{ij} \) is stored in \( ab[j - i + (i - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and \( j = i, \ldots, \min(n, i + k) \);
- if \( \text{order} = \text{Nag_RowMajor} \) and \( \text{uplo} = \text{Nag_Lower} \),
  \( a_{ij} \) is stored in \( ab[k + j - i + (i - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and \( j = \max(1,i-k), \ldots, i \).

On exit: \( A \) is overwritten.

7: \( \text{pdab} \) – Integer

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

Constraint: \( \text{pdab} \geq \max(1, k + 1) \).

8: \( d[\text{dim}] \) – double

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

On exit: the diagonal elements of the tridiagonal matrix \( T \).

9: \( e[\text{dim}] \) – double

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

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

10: \( q[\text{dim}] \) – Complex

Note: the dimension, \( \text{dim} \), of the array \( q \) must be at least \( \max(1, \text{pdq} \times n) \) when \( \text{vect} = \text{Nag_FormQ} \) or \( \text{Nag_UpdateQ} \); 1 when \( \text{vect} = \text{Nag_DoNotForm} \).

If \( \text{order} = \text{Nag_ColMajor} \), the \((i,j)\)th element of the matrix \( Q \) is stored in \( q[(j - 1) \times \text{pdq} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \((i,j)\)th element of the matrix \( Q \) is stored in \( q[(i - 1) \times \text{pdq} + j - 1] \).
On entry: if \( \text{vect} = \text{Nag\_UpdateQ} \), \( q \) 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 \( \text{vect} = \text{Nag\_FormQ} \) or \( \text{Nag\_UpdateQ} \), the \( n \) by \( n \) matrix \( Q \).

\( q \) is not referenced if \( \text{vect} = \text{Nag\_DoNotForm} \).

11: \( pdq \) – Integer  
Input

On entry: the stride separating matrix row or column elements (depending on the value of \( \text{order} \)) in the array \( q \).

Constraints:
- if \( \text{vect} = \text{Nag\_FormQ} \) or \( \text{Nag\_UpdateQ} \), \( pdq \geq \max(1, n) \);
- if \( \text{vect} = \text{Nag\_DoNotForm} \), \( pdq \geq 1 \).

12: \( \text{fail} \) – NagError *  
Output

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE\_INT}

On entry, \( n = \langle \text{value} \rangle \).
Constraint: \( n \geq 0 \).

On entry, \( kd = \langle \text{value} \rangle \).
Constraint: \( kd \geq 0 \).

On entry, \( pdab = \langle \text{value} \rangle \).
Constraint: \( pdab > 0 \).

On entry, \( pdq = \langle \text{value} \rangle \).
Constraint: \( pdq > 0 \).

\textbf{NE\_INT\_2}

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

\textbf{NE\_ENUM\_INT\_2}

On entry, \( \text{vect} = \langle \text{value} \rangle \), \( n = \langle \text{value} \rangle \), \( pdq = \langle \text{value} \rangle \).
Constraint: if \( \text{vect} = \text{Nag\_FormQ} \) or \( \text{Nag\_UpdateQ} \), \( pdq \geq \max(1, n) \);
if \( \text{vect} = \text{Nag\_DoNotForm} \), \( pdq \geq 1 \).

\textbf{NE\_ALLOC\_FAIL}

Memory allocation failed.

\textbf{NE\_BAD\_PARAM}

On entry, parameter \( \langle \text{value} \rangle \) had an illegal value.

\textbf{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 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 \( \epsilon \) 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 \( \epsilon \) is the machine precision.

8 Further Comments

The total number of real floating-point operations is approximately \( 20n^2k \) if \( \text{vect} = \text{Nag_DoNotForm} \) with \( 10n^3(k - 1)/k \) additional operations if \( \text{vect} = \text{Nag_FormQ} \).

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

9 Example

To compute 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 \( A \).

9.1 Program Text

/* nag_zhbtrd (f08hsc) Example Program.
 * Copyright 2001 Numerical Algorithms Group.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.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 uplo_char[2];
    Complex *ab=0, *z=0;
    double *d=0, *e=0;
    #ifdef NAG_COLUMN_MAJOR
    #define AR_UPPER(I,J) ab[(J-1)*pdab + k + I - J - 1]
    #else
    #define AR_UPPER(I,J) ab[(J-1)*pdab + k + I - J - 1]
    #endif
    /* ...
*/
#define AB_LOWER(I,J) ab[(J-1)*pdab + I - J]
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]
#endif

INIT_FAIL(fail);
Vprintf("f08hsc Example Program Results\n\n");
/* Skip heading in data file */
Vscanf("%*[\n \n]");
Vscanf("%ld%ld%*[\n \n] \n", &n, &kd);
pdab = kd + 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)) ) 
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read A from data file */
Vscanf(" ' %1s '%*[\n \n]", uplo_char);
if (*(unsigned char *)uplo_char == 'L')
    uplo = Nag_Lower;
else if (*(unsigned char *)uplo_char == 'U')
    uplo = Nag_Upper;
else 
{
    Vprintf("Unrecognised character for Nag_UploType type\n");
    exit_status = -1;
    goto END;
}
k = kd + 1;
/* Reduce A to tridiagonal form */
f08hsc(order, Nag_FormQ, uplo, n, kd, ab, pdab, d, e, 
z, pdz, &fail);
if (fail.code != NE_NOERROR) 
{
    Vprintf("Error from f08hsc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Calculate all the eigenvalues and eigenvectors of A */
fo8jsc(order, Nag_UpdateZ, n, d, e, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08jsc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print eigenvalues and eigenvectors */
Vprintf(" Eigenvalues\n");
for (i = 1; i <= n; ++i)
    Vprintf("%8.4f%s", d[i-1], i%8==0 ?"\n":" ");
Vprintf("\n\n");
x04dbc(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)
{
    Vprintf("Error from x04dbc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
END:
if (ab) NAG_FREE(ab);
if (d) NAG_FREE(d);
if (e) NAG_FREE(e);
if (z) NAG_FREE(z);
return exit_status;
}

9.2 Program Data

f08hsc Example Program Data

4 2
[L]
(-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

9.3 Program Results

f08hsc Example Program Results

Eigenvalues
-7.0042 -4.0038 0.5968 3.0012

Eigenvectors

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 0.7293, 0.0000) (-0.2128, 0.1511) (-0.3354,-0.1604) (-0.5114,-0.0163)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(-0.1654,-0.2046) ( 0.7316, 0.0000) (-0.2804,-0.3413) (-0.2374,-0.3796)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>( 0.6081, 0.0301) ( 0.3910,-0.3843) (-0.0144, 0.1532) ( 0.5523, 0.0000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>( 0.1653,-0.0303) ( 0.2775,-0.1378) ( 0.8019, 0.0000) (-0.4517, 0.1693)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>