NAG C Library Function Document
nag_dsbtrd (f08hec)

1 Purpose
nag_dsbtrd (f08hec) reduces a real symmetric band matrix to tridiagonal form.

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

3 Description
The symmetric band matrix $A$ is reduced to symmetric tridiagonal form $T$ by an orthogonal similarity transformation: $T = Q^T A Q$. The orthogonal 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

    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

    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

    On entry: indicates whether the upper or lower triangular part of $A$ is stored as follows:
    if uplo = Nag_Upper, the upper triangular part of $A$ is stored;
    if uplo = Nag_Lower, the lower triangular part of $A$ is stored.

    Constraint: uplo = Nag_Upper or Nag_Lower.
4: \( n \) – Integer \hspace{1cm} \text{Input}

On entry: \( n \), the order of the matrix \( A \).

Constraint: \( n \geq 0 \).

5: \( kd \) – Integer \hspace{1cm} \text{Input}

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: \( kd \geq 0 \).

6: \( \text{ab}[\text{dim}] \) – double \hspace{1cm} \text{Input/Output}

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

On entry: the \( n \) by \( n \) symmetric band matrix \( A \). 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 \( \text{ab}[k + i - j + (j - 1) \times \text{pdab}] \), 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 \( \text{ab}[i - j + (j - 1) \times \text{pdab}] \), 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 \( \text{ab}[j - i + (i - 1) \times \text{pdab}] \), 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 \( \text{ab}[k + j - i + (i - 1) \times \text{pdab}] \), for \( i = 1, \ldots, n \) and \( j = \max(1, i - k), \ldots, i \).

On exit: \( A \) is overwritten.

7: \( \text{pdab} \) – Integer \hspace{1cm} \text{Input}

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

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

8: \( \text{d}[\text{dim}] \) – double \hspace{1cm} \text{Output}

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

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

9: \( \text{e}[\text{dim}] \) – double \hspace{1cm} \text{Output}

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

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

10: \( \text{q}[\text{dim}] \) – double \hspace{1cm} \text{Input/Output}

Note: the dimension, \( \text{dim} \), of the array \( \text{q} \) 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 \( \text{q}[(j - 1) \times \text{pdq} + i - 1] \) and if \( \text{order} = \text{Nag_RowMajor} \), the \( (i,j) \)th element of the matrix \( Q \) is stored in \( \text{q}[(i - 1) \times \text{pdq} + j - 1] \).
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 symmetric-definite generalized eigenproblem); otherwise q need not be set.

On exit: if vect = Nag_FormQ or Nag_UpdateQ, the n by n matrix Q.
q is not referenced if vect = Nag_DoNotForm.

11: pdq – Integer

Input

On entry: the stride separating matrix 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 *

Output

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

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

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

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

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

NE_INT_2
On entry, pdab = ⟨value⟩, kd = ⟨value⟩.
Constraint: pdab ≥ max(1, kd + 1).

NE_ENUM_INT_2
On entry, vect = ⟨value⟩, n = ⟨value⟩, pdq = ⟨value⟩.
Constraint: if vect = Nag_FormQ or Nag_UpdateQ, pdq ≥ max(1, n);
if vect = Nag_DoNotForm, pdq ≥ 1.

NE_ALLOC_FAIL
Memory allocation failed.

NE_BAD_PARAM
On entry, parameter ⟨value⟩ had an illegal value.

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 orthogonal matrix by a matrix $E$ such that

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

where $\epsilon$ is the machine precision.

8 Further Comments
The total number of floating-point operations is approximately $6n^2k$ if vect = Nag_DoNotForm with $3n^3(k - 1)/k$ additional operations if vect = Nag_FormQ.

The complex analogue of this function is nag_zhbtrd (f08hsc).

9 Example
To compute all the eigenvalues and eigenvectors of the matrix $A$, where

$$A = \begin{pmatrix} 4.99 & 0.04 & 0.22 & 0.00 \\ 0.04 & 1.05 & -0.79 & 1.04 \\ 0.22 & -0.79 & -2.31 & -1.30 \\ 0.00 & 1.04 & -1.30 & -0.43 \end{pmatrix}.$$  

Here $A$ is symmetric and is treated as a band matrix. The program first calls nag_dsbtrd (f08hec) to reduce $A$ to tridiagonal form $T$, and to form the orthogonal matrix $Q$; the results are then passed to nag_dsteqr (f08jec) which computes the eigenvalues and eigenvectors of $A$.

9.1 Program Text
/* nag_dsbtrd (f08hec) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */
#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];
    double *ab=0, *d=0, *e=0, *z=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]
    #endif

    //...
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
 order = Nag_RowMajor;
#endif
INIT_FAIL(fail);
Vprintf("f08hec Example Program Results\n\n");
/* Skip heading in data file */
Vscanf("%*[\n\] ");
Vscanf("%d%d%*[\n\] ", &n, &kd);
pdab = kd + 1;
pdz = n;
d_len = n;
e_len = n-1;
/* Allocate memory */
if ( !(ab = NAG_ALLOC(pdab * n, double)) ||
 ! (d = NAG_ALLOC(d_len, double)) ||
 ! (e = NAG_ALLOC(e_len, double)) ||
 !(z = NAG_ALLOC(pdz * n, double)) )
{
 Vprintf("Allocation failure\n");
 exit_status = -1;
 goto END;
}
/* Read A from data file */
Vscanf( " ' %1s '%*[\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;
if ( uplo == Nag_Upper )
{
     for ( i = 1; i <= n; ++i )
     {
     for ( j = i; j <= MIN(i+kd,n); ++j )
 Vscanf("%lf", &AB_UPPER(i,j));
    }
 Vscanf("%*[\n\] ");
     }
else
{
     for ( i = 1; i <= n; ++i )
     {
 for ( j = MAX(1,i-kd); j <= i; ++j )
 Vscanf("%lf", &AB_LOWER(i,j));
       }
 Vscanf("%*[\n\] ");
   }

/* Reduce A to tridiagonal form */
f08hec(order, Nag_FormQ, uplo, n, kd, ab, pdab, d, e, 
z, pdz, &fail);
if ( fail.code != NE_NOERROR )
{
 Vprintf("Error from f08hec.\n%\s\n", fail.message);
 exit_status = 1;
 goto END;
}
/* Calculate all the eigenvalues and eigenvectors of A */
f08jec(order, Nag_UpdateZ, n, d, e, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08jec.\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");
x04cac(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n,
z, pdz, "Eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04cac.\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

f08hec Example Program Data
4 2 :Values of N and KD
'L' :Value of UPLO
4.99
0.04 1.05
0.22 -0.79 -2.31
1.04 -1.30 -0.43 :End of matrix A

9.3 Program Results

f08hec Example Program Results

Eigenvalues
-2.9943 -0.7000 1.9974 4.9969

Eigenvalues

9.2 Program Data

f08hec Example Program Data
4 2 :Values of N and KD
'L' :Value of UPLO
4.99
0.04 1.05
0.22 -0.79 -2.31
1.04 -1.30 -0.43 :End of matrix A

9.3 Program Results

f08hec Example Program Results

Eigenvalues
-2.9943 -0.7000 1.9974 4.9969

Eigenvalues