

NAG Library Function Document

nag_zhbev (f08hnc)

1 Purpose

nag_zhbev (f08hnc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex n by n Hermitian band matrix A of bandwidth $(2k_d + 1)$.

2 Specification

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

void nag_zhbev (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
               Integer n, Integer kd, Complex ab[], Integer pdab, double w[],
               Complex z[], Integer pdz, NagError *fail)
```

3 Description

The Hermitian band matrix A is first reduced to real tridiagonal form, using unitary similarity transformations, and then the QR algorithm is applied to the tridiagonal matrix to compute the eigenvalues and (optionally) the eigenvectors.

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: **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: **uplo** – Nag_UploType *Input*

On entry: 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 *Input*

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

Constraint: $n \geq 0$.

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: $kd \geq 0$.

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 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 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 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 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: $\mathbf{pdab} \geq \mathbf{kd} + 1$.

8: **w**[**n**] – double *Output*

On exit: the eigenvalues in ascending order.

9: **z**[*dim*] – Complex *Output*

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

```

max(1, pdz × n) when job = Nag_DoBoth;
1 otherwise.

```

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

$$\begin{aligned} & \mathbf{z}[(j-1) \times \mathbf{pdz} + i - 1] \text{ when } \mathbf{order} = \text{Nag_ColMajor}; \\ & \mathbf{z}[(i-1) \times \mathbf{pdz} + j - 1] \text{ when } \mathbf{order} = \text{Nag_RowMajor}. \end{aligned}$$

On exit: if $\mathbf{job} = \text{Nag_DoBoth}$, \mathbf{z} contains the orthonormal eigenvectors of the matrix A , with the i th column of Z holding the eigenvector associated with $\mathbf{w}[i-1]$.

If $\mathbf{job} = \text{Nag_EigVals}$, \mathbf{z} is not referenced.

10: **pdz** – Integer *Input*

On entry: the stride separating row or column elements (depending on the value of **order**) in the array \mathbf{z} .

Constraints:

$$\begin{aligned} & \text{if } \mathbf{job} = \text{Nag_DoBoth}, \mathbf{pdz} \geq \max(1, \mathbf{n}); \\ & \text{otherwise } \mathbf{pdz} \geq 1. \end{aligned}$$

11: **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.

NE_BAD_PARAM

On entry, argument $\langle value \rangle$ had an illegal value.

NE_CONVERGENCE

The algorithm failed to converge; $\langle value \rangle$ off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

NE_ENUM_INT_2

On entry, $\mathbf{job} = \langle value \rangle$, $\mathbf{pdz} = \langle value \rangle$ and $\mathbf{n} = \langle value \rangle$.
Constraint: if $\mathbf{job} = \text{Nag_DoBoth}$, $\mathbf{pdz} \geq \max(1, \mathbf{n})$;
otherwise $\mathbf{pdz} \geq 1$.

NE_INT

On entry, $\mathbf{kd} = \langle value \rangle$.
Constraint: $\mathbf{kd} \geq 0$.

On entry, $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{n} \geq 0$.

On entry, $\mathbf{pdab} = \langle value \rangle$.
Constraint: $\mathbf{pdab} > 0$.

On entry, $\mathbf{pdz} = \langle value \rangle$.
Constraint: $\mathbf{pdz} > 0$.

NE_INT_2

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

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.

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_zhbev (f08hnc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zhbev (f08hnc) 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 Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

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

The real analogue of this function is nag_dsbev (f08hac).

10 Example

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

$$A = \begin{pmatrix} 1 & 2 - i & 3 - i & 0 & 0 \\ 2 + i & 2 & 3 - 2i & 4 - 2i & 0 \\ 3 + i & 3 + 2i & 3 & 4 - 3i & 5 - 3i \\ 0 & 4 + 2i & 4 + 3i & 4 & 5 - 4i \\ 0 & 0 & 5 + 3i & 5 + 4i & 5 \end{pmatrix},$$

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

10.1 Program Text

```
/* nag_zhbev (f08hnc) Example Program.
 *
 * Copyright 2011 Numerical Algorithms Group.
 *
 * Mark 23, 2011.
 */
```

```
#include <math.h>
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf08.h>
#include <nagx02.h>
#include <nagx04.h>
```

```
int main(void)
{
```

```

/* Scalars */
double      eerrbd, eps;
Integer     exit_status = 0, i, j, kd, n, pdab, pdz;
/* Arrays */
char        nag_enum_arg[40];
Complex     *ab = 0, *z = 0;
double      *rcondz = 0, *w = 0, *zerrbd = 0;
/* Nag Types */
Nag_OrderType order;
Nag_UploType  uplo;
Nag_Error     fail;

#ifdef NAG_COLUMN_MAJOR
#define AB_UPPER(I, J) ab[(J - 1) * pdab + kd + I - J]
#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 + kd + J - I]
#define Z(I, J) z[(I - 1) * pdz + J - 1]
    order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_zhbev (f08hnc) Example Program Results\n\n");

/* Skip heading in data file */
scanf("%*[\n]");
scanf("%ld%ld%*[\n]", &n, &kd);

/* Read uplo */
scanf("%39s%*[\n]", nag_enum_arg);
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value
 */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

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

pdab = kd+1;
pdz = n;

/* Read the upper or lower triangular part of the symmetric band
 * matrix A from data file.
 */
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= MIN(n, i + kd); ++j)
            scanf(" ( %lf , %lf )", &AB_UPPER(i, j).re, &AB_UPPER(i, j).im);
    scanf("%*[\n]");
}
else if (uplo == Nag_Lower) {
    for (i = 1; i <= n; ++i)
        for (j = MAX(1, i - kd); j <= i; ++j)
            scanf(" ( %lf , %lf )", &AB_LOWER(i, j).re, &AB_LOWER(i, j).im);
    scanf("%*[\n]");
}

/* nag_zhbev (f08hnc).

```

```

    * Solve the band Hermitian eigenvalue problem.
    */
nag_zhbev(order, Nag_DoBoth, uplo, n, kd, ab, pdab, w, z, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhbev (f08hnc).\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--)
        Z(i, j) = nag_complex_divide(Z(i, j),Z(1, j));

/* Print solution */
printf("Eigenvalues\n");
for (j = 0; j < n; ++j)
    printf("%8.4f%s", w[j], (j+1)%8 == 0?"\n":" ");
printf("\n");

/* nag_gen_complx_mat_print (x04dac).
 * Print eigenvectors.
 */
fflush(stdout);
nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, n, z,
                        pdz, "Eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print (x04dac).\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 {|w[i]|, i=0..n-1}, and since
 * the eigenvalues are in ascending order ||A|| = max( |w[0]|, |w[n-1]|).
 */
eps = nag_machine_precision;
eerrbd = eps * MAX(fabs(w[0]), fabs(w[n-1]));

/* nag_ddisna (f08flc).
 * Estimate reciprocal condition numbers for the eigenvectors.
 */
nag_ddisna(Nag_EigVecs, n, n, w, 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("Error 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(ab);
NAG_FREE(z);

```

```

NAG_FREE(rcondz);
NAG_FREE(w);
NAG_FREE(zerrbd);

return exit_status;
}

#undef AB_UPPER
#undef AB_LOWER
#undef Z

```

10.2 Program Data

nag_zhbev (f08hnc) Example Program Data

```

5          2                               :Values of n and kd
Nag_Upper                               :Value of uplo

(1.0, 0.0) (2.0,-1.0) (3.0,-1.0)
          (2.0, 0.0) (3.0,-2.0) (4.0,-2.0)
          (3.0, 0.0) (4.0,-3.0) (5.0,-3.0)
          (4.0, 0.0) (5.0,-4.0)
          (5.0, 0.0) :End of matrix A

```

10.3 Program Results

nag_zhbev (f08hnc) Example Program Results

```

Eigenvalues
-6.4185  -1.4094   1.4421   4.4856  16.9002
Eigenvectors
      1      2      3      4      5
1      1.0000   1.0000   1.0000   1.0000   1.0000
      -0.0000   0.0000   0.0000   0.0000   0.0000

2      -0.0946  -0.4049  -0.6707   0.8697   2.1263
      -1.6770   0.3789  -0.9748   0.5014   0.2858

3      -1.9916  -0.4773   0.6996   0.3868   3.2531
      0.4226   -0.5467   0.6595   0.0846   1.6026

4      -0.0014   0.5418  -0.9052  -0.3102   2.8478
      1.9659  -0.1307  -0.7115   0.0364   2.6633

5      1.6725  -0.3878   0.0451   0.0318   1.2641
      -0.5221   0.4137   0.5008  -1.0197   3.5697

```

Error estimate for the eigenvalues

1.9e-15

Error estimates for the eigenvectors

3.7e-16 6.6e-16 6.6e-16 6.2e-16 1.5e-16
