NAG Library Function Document

nag_zhbevd (f08hqc)

1 Purpose

nag_zhbevd (f08hqc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex Hermitian band matrix. If the eigenvectors are requested, then it uses a divide-and-conquer algorithm to compute eigenvalues and eigenvectors. However, if only eigenvalues are required, then it uses the Pal–Walker–Kahan variant of the QL or QR algorithm.

2 Specification

```c
#include <nag.h>
#include <nagf08.h>
void nag_zhbevd (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

nag_zhbevd (f08hqc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex Hermitian band matrix $A$. In other words, it can compute the spectral factorization of $A$ as

$$A = Z \Lambda Z^H,$$

where $\Lambda$ is a real diagonal matrix whose diagonal elements are the eigenvalues $\lambda_i$, and $Z$ is the (complex) unitary matrix whose columns are the eigenvectors $z_i$. Thus

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

4 References


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_DoNothing`

   Only eigenvalues are computed.

Mark 25
Eigenvalues and eigenvectors are computed.

Constraint: $\text{job} = \text{Nag\_DoNothing}$ or $\text{Nag\_EigVecs}$.

3: $\text{uplo} \rightarrow \text{Nag\_UploType}$  

*Input*

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

- $\text{uplo} = \text{Nag\_Upper}$
  - The upper triangular part of $A$ is stored.
- $\text{uplo} = \text{Nag\_Lower}$
  - The lower triangular part of $A$ is stored.

Constraint: $\text{uplo} = \text{Nag\_Upper}$ or $\text{Nag\_Lower}$.

4: $n \rightarrow \text{Integer}$  

*Input*

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

Constraint: $n \geq 0$.

5: $kd \rightarrow \text{Integer}$  

*Input*

On entry: if $\text{uplo} = \text{Nag\_Upper}$, the number of superdiagonals, $k_d$, of the matrix $A$.

If $\text{uplo} = \text{Nag\_Lower}$, the number of subdiagonals, $k_d$, of the matrix $A$.

Constraint: $kd \geq 0$.

6: $\text{ab}[\text{dim}] \rightarrow \text{Complex}$  

*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 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 $\text{order}$ and $\text{uplo}$ arguments as follows:

- if $\text{order} = \text{Nag\_ColMajor}$ and $\text{uplo} = \text{Nag\_Upper}$,
  - $A_{ij}$ is stored in $\text{ab}[k_d + i - j + (j - 1) \times \text{pdab}]$, for $j = 1, \ldots, n$ and $i = \max(1, j - k_d), \ldots, j$;
- 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 $j = 1, \ldots, n$ and $i = j, \ldots, \min(n, j + k_d)$;
- 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 $j = i, \ldots, \min(n, i + k_d)$;
- if $\text{order} = \text{Nag\_RowMajor}$ and $\text{uplo} = \text{Nag\_Lower}$,
  - $A_{ij}$ is stored in $\text{ab}[k_d + j - i + (i - 1) \times \text{pdab}]$, for $i = 1, \ldots, n$ and $j = \max(1, i - k_d), \ldots, i$.

On exit: $\text{ab}$ 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 $\text{ab}$ using the same storage format as described above.

7: $\text{pdab} \rightarrow \text{Integer}$  

*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: $\text{pdab} \geq kd + 1$. 

---

The given text describes the parameters and constraints for computing eigenvalues and eigenvectors using the NAG Library function f08hqc. It details the data types, input and output requirements, and the storage formats for the matrix $A$. The text also explains the constraints on the parameters $\text{job}$, $\text{uplo}$, $n$, $kd$, and $\text{dim}$, and the output $\text{ab}$ and $\text{pdab}$. The storage of the matrix elements in the array $\text{ab}$ is described based on the order and uplo arguments, and the values are overwritten during the reduction to tridiagonal form.
8: \( \mathbf{w}[\text{dim}] \) – double

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

On exit: the eigenvalues of the matrix \( \mathbf{A} \) in ascending order.

9: \( \mathbf{z}[\text{dim}] \) – Complex

Note: the dimension, \( \text{dim} \), of the array \( \mathbf{z} \) must be at least

\[ \max(1, \text{pdz} \times n) \] when \( \text{job} = \text{Nag\_EigVecs} \);
1 when \( \text{job} = \text{Nag\_DoNothing} \).

The \((i,j)\)th element of the matrix \( \mathbf{Z} \) is stored in

\[ \mathbf{z}[ (j-1) \times \text{pdz} + i - 1] \] when \( \text{order} = \text{Nag\_ColMajor} \);
\[ \mathbf{z}[ (i-1) \times \text{pdz} + j - 1] \] when \( \text{order} = \text{Nag\_RowMajor} \).

On exit: if \( \text{job} = \text{Nag\_EigVecs} \), \( \mathbf{z} \) is overwritten by the unitary matrix \( \mathbf{Z} \) which contains the eigenvectors of \( \mathbf{A} \). The \( i \)th column of \( \mathbf{Z} \) contains the eigenvector which corresponds to the eigenvalue \( \mathbf{w}[i-1] \).

If \( \text{job} = \text{Nag\_DoNothing} \), \( \mathbf{z} \) is not referenced.

10: \( \text{pdz} \) – Integer

Input

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

Constraints:

\[ \text{if} \ \text{job} = \text{Nag\_EigVecs}, \ \text{pdz} \geq \max(1,n); \]
\[ \text{if} \ \text{job} = \text{Nag\_DoNothing}, \ \text{pdz} \geq 1. \]

11: \( \text{fail} \) – 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 \(<\text{value}>\) had an illegal value.

NE_CONVERGENCE

If fail.errnum = \(<\text{value}>\) and job = Nag\_DoNothing, the algorithm failed to converge; \(<\text{value}>\) elements of an intermediate tridiagonal form did not converge to zero; if fail.errnum = \(<\text{value}>\) and job = Nag\_EigVecs, then the algorithm failed to compute an eigenvalue while working on the submatrix lying in rows and column \(<\text{value}>\)/(\(n + 1\)) through \(<\text{value}>\) mod (\(n + 1\)).

NE_ENUM_INT_2

On entry, job = \(<\text{value}>\), pdz = \(<\text{value}>\) and n = \(<\text{value}>\).
Constraint: if job = Nag\_EigVecs, pdz \(\geq\) max(1,n);
if job = Nag\_DoNothing, pdz \(\geq\) 1.

NE_INT

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

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

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

**NE_INT_2**
On entry, $pdab = \langle value \rangle$ and $kd = \langle value \rangle$.
Constraint: $pdab \geq kd + 1$.

**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 eigenvalues and eigenvectors are exact for a nearby matrix $(A + E)$, where

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

and $\epsilon$ is the *machine precision*. See Section 4.7 of Anderson *et al.* (1999) for further details.

### 8 Parallelism and Performance

`nag_zhbevd (f08hqc)` is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

`nag_zhbevd (f08hqc)` 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 real analogue of this function is `nag_dsbevd (f08hcc)`.

### 10 Example

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

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

/* nag_zhbevd (f08hqc) Example Program.  
* Copyright 2014 Numerical Algorithms Group.  
*/

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

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

    INIT_FAIL(fail);
    printf("nag_zhbevd (f08hqc) Example Program Results\n\n");

    /* Read whether Upper or Lower part of A is stored */
    #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_zhbevd (f08hqc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n] ");
    #else
    scanf("%*[\n] ");
    #endif
    #ifdef _WIN32
    scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &kd);
    #else
    scanf("%"NAG_IFMT"%"NAG_IFMT"%*[\n] ", &n, &kd);
    #endif
    pdab = kd + 1;
    pdz = n;
    w_len = n;

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

    /* Read whether Upper or Lower part of A is stored */

    END:
    exit_status;
    return exit_status;
}
```c
#ifdef _WIN32
    scanf_s("%39s%\n\n", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s%\n\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);

/* Read A from data file */
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\n");
    #else
        scanf("%\n\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\n");
    #else
        scanf("%\n\n");
    #endif
}

/* Read type of job to be performed */
#ifdef _WIN32
    scanf_s("%39s%\n\n", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s%\n\n", nag_enum_arg);
#endif

job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);

/* Calculate all the eigenvalues and eigenvectors of A */
/* nag_zhbevd (f08hqc).
* All eigenvalues and optionally all eigenvectors of
* complex Hermitian band matrix (divide-and-conquer)
*/
if (fail.code != NE_NOERROR)
    printf("Error from nag_zhbevd (f08hqc).
%s
", 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 = 0; i < n; ++i)
    printf(" %5"NAG_IFMT" %8.4f\n", i + 1, w[i]);
printf("\n");
/* nag_gen_complex_mat_print_comp (x04dbc).
 * Print complex general matrix (comprehensive)
 */
fflush(stdout);
nag_gen_complex_mat_print_comp(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, 
n, z, pdz, Nag_AboveForm, "%7.4f", 
"Eigenvectors", Nag_IntegerLabels, 0, 
Nag_IntegerLabels, 0, 80, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf(
    "Error from nag_gen_complex_mat_print_comp (x04dbc).\n%s\n", 
    fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(ab);
NAG_FREE(w);
NAG_FREE(z);
return exit_status;

10.2 Program Data
nag_zhbevd (f08hqc) Example Program Data
5 2
Nag_Lower
(1.0, 0.0) (2.0, 1.0) (2.0, 0.0)
(3.0, 1.0) (3.0, 2.0) (3.0, 0.0)
(4.0, 2.0) (4.0, 3.0) (4.0, 0.0)
(5.0, 3.0) (5.0, 4.0) (5.0, 0.0)
Nag_EigVecs
:Values of n and kd
:Value of uplo
:Value of job
:End of matrix A
### 10.3 Program Results

nag_zhbevd (f08hqc) Example Program Results

**Eigenvalues**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-6.4185</td>
</tr>
<tr>
<td>2</td>
<td>-1.4094</td>
</tr>
<tr>
<td>3</td>
<td>1.4421</td>
</tr>
<tr>
<td>4</td>
<td>4.4856</td>
</tr>
<tr>
<td>5</td>
<td>16.9002</td>
</tr>
</tbody>
</table>

**Eigenvectors**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>-0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>-0.0946</td>
<td>-0.4049</td>
<td>-0.6707</td>
<td>0.8697</td>
<td>2.1263</td>
</tr>
<tr>
<td></td>
<td>-1.6770</td>
<td>0.3789</td>
<td>-0.9748</td>
<td>0.5014</td>
<td>0.2858</td>
</tr>
<tr>
<td>3</td>
<td>-1.9916</td>
<td>-0.4773</td>
<td>0.6996</td>
<td>0.3868</td>
<td>3.2531</td>
</tr>
<tr>
<td></td>
<td>0.4226</td>
<td>-0.5467</td>
<td>0.6595</td>
<td>0.0846</td>
<td>1.6026</td>
</tr>
<tr>
<td>4</td>
<td>-0.0014</td>
<td>0.5418</td>
<td>-0.9052</td>
<td>-0.3102</td>
<td>2.8478</td>
</tr>
<tr>
<td></td>
<td>1.9659</td>
<td>-0.1307</td>
<td>-0.7115</td>
<td>0.0364</td>
<td>2.6633</td>
</tr>
<tr>
<td>5</td>
<td>1.6725</td>
<td>-0.3878</td>
<td>0.0451</td>
<td>0.0318</td>
<td>1.2641</td>
</tr>
<tr>
<td></td>
<td>-0.5221</td>
<td>0.4137</td>
<td>0.5008</td>
<td>-1.0197</td>
<td>3.5697</td>
</tr>
</tbody>
</table>