NAG C 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
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 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: job – Nag_JobType

   On entry: indicates whether eigenvectors are computed as follows:
   - if job = Nag_DoNothing, only eigenvalues are computed;
   - if job = Nag_EigVecs, eigenvalues and eigenvectors are computed.

   Constraint: job = Nag_DoNothing or Nag_EigVecs.

3: uplo – Nag_UploType

   On entry: indicates whether the upper or lower triangular part of $A$ is stored as follows:
if \( \text{uplo} = \text{Nag\_Upper} \), the upper triangular part of \( A \) is stored;

if \( \text{uplo} = \text{Nag\_Lower} \), the lower triangular part of \( A \) is stored.

**Constraint:** \( \text{uplo} = \text{Nag\_Upper} \) or \( \text{Nag\_Lower} \).

4: \n- **n** – Integer  \( \text{Input} \)

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

**Constraint:** \( n \geq 0 \).

5: \n- **kd** – Integer  \( \text{Input} \)

**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: \n- **ab[dim]** – Complex  \( \text{Input/Output} \)

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

**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 \( ab[k + i - j + (j - 1) \times pdab] \), 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 \( ab[i - j + (j - 1) \times pdab] \), 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 \( ab[j - i + (i - 1) \times pdab] \), 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 \( ab[k + j - i + (i - 1) \times pdab] \), for \( i = 1, \ldots, n \) and \( j = \max(1, i - k), \ldots, i \).

**On exit:** \( A \) is overwritten by the values generated during the reduction to tridiagonal form. If \( \text{uplo} = \text{Nag\_Upper} \), the first superdiagonal and the diagonal of the tridiagonal matrix are returned in rows \( kd \) and \( kd + 1 \) of the array \( ab \), respectively, and if \( \text{uplo} = \text{Nag\_Lower} \) then the diagonal and the first subdiagonal of the tridiagonal matrix are returned in the first two rows of the array \( ab \).

7: \n- **pdab** – Integer  \( \text{Input} \)

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

**Constraint:** \( pdab \geq kd + 1 \).

8: \n- **w[dim]** – double  \( \text{Output} \)

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

**On exit:** the eigenvalues of the matrix \( A \) in ascending order.

9: \n- **z[dim]** – Complex  \( \text{Output} \)

**Note:** the dimension, \( dim \), of the array \( z \) must be at least \( \max(1, pdz \times n) \) when \( \text{job} = \text{Nag\_EigVecs} \);

1 when \( \text{job} = \text{Nag\_DoNothing} \).
If order = Nag_ColMajor, the \((i, j)\)th element of the matrix \(Z\) is stored in \(z[(j - 1) \times pdz + i - 1]\) and if order = Nag_RowMajor, the \((i, j)\)th element of the matrix \(Z\) is stored in \(z[(i - 1) \times pdz + j - 1]\).

On exit: if job = Nag_EigVecs, \(Z\) is overwritten by the unitary matrix which contains the eigenvectors of \(A\). The \(i\)th column of \(Z\) contains the eigenvector which corresponds to the eigenvalue \(w[i]\).

If job = Nag_DoNothing, \(Z\) is not referenced.

10: \(pdz\) – Integer

\(pdz\) is the stride separating matrix row or column elements (depending on the value of order) in the array \(z\).

Constraints:
- if job = Nag_EigVecs, \(pdz \geq \max(1, n)\);
- if job = Nag_DoNothing, \(pdz \geq 1\).

11: \(fail\) – NagError *

The NAG error parameter (see the Essential Introduction).

6 Error Indicators and Warnings

\textbf{NE_INT}

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

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

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

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

\textbf{NE_INT_2}

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

\textbf{NE_ENUM_INT_2}

On entry, \(job = \langle value\rangle, n = \langle value\rangle, pdz = \langle value\rangle\).
Constraint: if job = Nag_EigVecs, \(pdz \geq \max(1, n)\);
if job = Nag_DoNothing, \(pdz \geq 1\).

\textbf{NE_CONVERGENCE}

The algorithm failed to converge, \(\langle value\rangle\) elements of an intermediate tridiagonal form did not converge to zero.

\textbf{NE_ALLOC_FAIL}

Memory allocation failed.

\textbf{NE_BAD_PARAM}

On entry, parameter \(\langle value\rangle\) had an illegal value.
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 eigenvalues and eigenvectors are exact for a nearby matrix $A + E$, where

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

and $\epsilon$ is the machine precision.

8 Further Comments

The real analogue of this function is nag_dsbevd (f08hcc).

9 Example

To compute all the eigenvalues and eigenvectors of the Hermitian band matrix $A$, where

$$
A = \begin{pmatrix}
1.0 + 0.0i & 2.0 - 1.0i & 3.0 - 1.0i & 0.0 + 0.0i & 0.0 + 0.0i \\
2.0 + 1.0i & 2.0 + 0.0i & 3.0 - 2.0i & 4.0 - 2.0i & 0.0 + 0.0i \\
3.0 + 1.0i & 3.0 + 2.0i & 3.0 + 0.0i & 4.0 - 3.0i & 5.0 - 3.0i \\
0.0 + 0.0i & 4.0 + 2.0i & 4.0 + 3.0i & 4.0 + 0.0i & 5.0 - 4.0i \\
0.0 + 0.0i & 0.0 + 0.0i & 5.0 + 3.0i & 5.0 + 4.0i & 5.0 + 0.0i \\
\end{pmatrix}
$$

9.1 Program Text

/* nag_zhbevd (f08hqc) 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, w_len;
    Integer exit_status=0;
    NagError fail;
    Nag_JobType job;
    Nag_UploType uplo;
    Nag_OrderType order;
    /* Arrays */
    char uplo_char[2], job_char[2];
    Complex *ab=0, *z=0;
    double *w=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]
    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]
    order = Nag_RowMajor;
    #endif

    return 0;
}
INIT_FAIL(fail);
Vprintf("f08hqc Example Program Results\n\n");

/* Skip heading in data file */
Vscanf("%*[\n] ");
Vscanf("%ld%ld%*[\n] ", &n, &kd);
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)))
{
    Vprintf("Allocation failure\n");
    exit_status = -1;
go END;
}

/* Read whether Upper or Lower part of A is stored */
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;
go END;
}

/* 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)
        {
            Vscanf(" ( %lf , %lf )", &AB_UPPER(i,j).re,
                    &AB_UPPER(i,j).im);
        }
    }
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = MAX(1,i-kd); j <= i; ++j)
        {
            Vscanf(" ( %lf , %lf )", &AB_LOWER(i,j).re,
                    &AB_LOWER(i,j).im);
        }
    }
}
/* Read type of job to be performed */
Vscanf(" ' %1s '%*[\n] ", job_char);
if (*(unsigned char *)job_char == 'V')
    job = Nag_EigVecs;
else
    job = Nag_DoNothing;
/* Calculate all the eigenvalues and eigenvectors of A */
f08hqc(order, job, uplo, n, kd, ab, pdab, w, pdz, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08hqc.\n\n");
    exit_status = 1;
go END;
}
/* Print eigenvalues and eigenvectors */
Vprintf(" Eigenvalues\n");
for (i = 0; i < n; ++i)
    Vprintf(" %5ld %8.4f\n", i+1, w[i]);
Vprintf("\n");
x04dbc(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)
    {
    Vprintf("Error from x04dbc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
    }
END:
if (ab) NAG_FREE(ab);
if (w) NAG_FREE(w);
if (z) NAG_FREE(z);
return exit_status;

9.2 Program Data
f08hqc Example Program Data
5 2
'L'
(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) :End of matrix A
'V'
:Value of JOB

9.3 Program Results
f08hqc Example Program Results

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

Eigenvalues
1  -0.2591  0.6367  0.4516  0.5503  0.1439
-0.0000 -0.0000 -0.0000 -0.0000 -0.0000
2   0.0245 -0.2578 -0.3029  0.4785  0.3060
  0.4344  0.2413 -0.4402  0.2759  0.0411
3   0.5159 -0.3039  0.3160  0.2128  0.4681
-0.1095 -0.3481  0.2978  0.0465  0.2306
4   0.0004  0.3450 -0.4088 -0.1707  0.4098
-0.5093 -0.0832 -0.3213  0.0200  0.3832
5  -0.4333 -0.2469  0.0204  0.0175  0.1819
  0.1353  0.2634  0.2262 -0.5611  0.5136