NAG C Library Function Document

nag_zheevd (f08fqc)

1 Purpose

nag_zheevd (f08fqc) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian 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_zheevd (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
                Integer n, Complex a[], Integer pda, double w[], NagError *fail)
```

3 Description

nag_zheevd (f08fqc) computes all the eigenvalues, and optionally all the eigenvectors, of a complex Hermitian 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: \( \text{order} \) – Nag_OrderType

*Input*

*On entry*: the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 2.2.1.4 of the Essential Introduction for a more detailed explanation of the use of this parameter.

*Constraint*: \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( \text{job} \) – Nag_JobType

*Input*

*On entry*: indicates whether eigenvectors are computed as follows:

- if \( \text{job} = \text{Nag_DoNothing} \), only eigenvalues are computed;
- if \( \text{job} = \text{Nag_EigVecs} \), eigenvalues and eigenvectors are computed.

*Constraint*: \( \text{job} = \text{Nag_DoNothing} \) or \( \text{Nag_EigVecs} \).

3: \( \text{uplo} \) – Nag_UploType

*Input*

*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: \( \text{n} \) – Integer
   \( \text{Input} \)
   \( \text{On entry:} \, n \), the order of the matrix \( A \).
   \( \text{Constraint:} \, n \geq 0. \)

5: \( a[dim] \) – Complex
   \( \text{Input/Output} \)
   \( \text{Note:} \) the dimension, \( dim \), of the array \( a \) must be at least \( \max(1, \text{pda} \times n) \).
   
   \( \text{If order = Nag_ColMajor,} \) the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(j-1) \times \text{pda} + i - 1] \) and
   \( \text{if order = Nag_RowMajor,} \) the \((i, j)\)th element of the matrix \( A \) is stored in \( a[(i-1) \times \text{pda} + j - 1] \).
   
   \( \text{On entry:} \) the \( n \) by \( n \) Hermitian matrix \( A \). \( \text{If uplo = Nag_Upper,} \) the upper triangular part of \( A \) must be stored and the elements of the array below the diagonal are not referenced; if \( \text{uplo = Nag_Lower,} \) the lower triangular part of \( A \) must be stored and the elements of the array above the diagonal are not referenced.
   
   \( \text{On exit:} \) if \( \text{job = Nag_EigVecs,} \) this is overwritten by the unitary matrix \( Z \) which contains the eigenvectors of \( A \).

6: \( \text{pda} \) – Integer
   \( \text{Input} \)
   \( \text{On entry:} \) the stride separating row or column elements (depending on the value of \( \text{order} \)) of the matrix \( A \) in the array \( a \).
   \( \text{Constraint:} \, \text{pda} \geq \max(1, n). \)

7: \( w[dim] \) – double
   \( \text{Output} \)
   \( \text{Note:} \) the dimension, \( dim \), of the array \( w \) must be at least \( \max(1, n) \).
   
   \( \text{On exit:} \) the eigenvalues of the matrix \( A \) in ascending order.

8: \( \text{fail} \) – NagError *
   \( \text{Output} \)
   The NAG error parameter (see the Essential Introduction).

6 \ Error Indicators and Warnings

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

\textbf{NE_INT_2}
\( \text{On entry,} \, \text{pda} = \langle \text{value} \rangle, \, \text{n} = \langle \text{value} \rangle. \)
\( \text{Constraint:} \, \text{pda} \geq \max(1, n). \)

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

\textbf{NE_ALLOC_FAIL}
\( \text{Memory allocation failed.} \)

\textbf{NE_BAD_PARAM}
\( \text{On entry, parameter} \, \langle \text{value} \rangle \text{had an illegal value.} \)
**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_dsyevd (f08fcc).

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

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

9.1 Program Text
/* nag_zheevd (f08fqc) Example Program. */
/* Copyright 2001 Numerical Algorithms Group. */
/* Mark 7, 2001. */

```c
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
  /* Scalars */
  Integer i, j, n, pda, 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];
  double *w=0;
  Complex *a=0;

  INIT_FAIL(fail);
  Vprintf("f08fqc Example Program Results\n\n");

  /* Skip heading in data file */
```
```
Vscanf("%*[\n \n ] ");
Vscanf("%ld%*[\n \n ] ", &n);
pda = n;
w_len = n;

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

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

/* Read A from data file */
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
    {
        for (j = i; j <= n; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
    Vscanf("%*[\n \n ] ");
}
else
{
    for (i = 1; i <= n; ++i)
    {
        for (j = 1; j <= i; ++j)
            Vscanf(" ( %lf , %lf )", &A(i,j).re, &A(i,j).im);
    }
    Vscanf("%*[\n \n ] ");
}

/* Read type of job to be performed */
Vscanf(" %1s '%*[\n \n ] ", job_char);
if (*(unsigned char *)job_char == 'V')
    job = Nag_EigVecs;
else
    job = Nag_DoNothing;

/* Calculate all the eigenvalues and eigenvectors of A */
f08fqc(order, job, uplo, n, a, pda, w, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from f08fqc.\n%s\n", fail.message);
    exit_status = 1;
    goto 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, a, pda,
       Nag_AboveForm, "%7.4f", "Eigenvectors", Nag_IntegerLabels,
       0, Nag_IntegerLabels, 0, 0, 0, &fail);
if (fail.code != NE_NOERROR)
{
    Vprintf("Error from x04dbc.\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}
if (a) NAG_FREE(a);
if (w) NAG_FREE(w);
return exit_status;
}

9.2 Program Data

f08fqc Example Program Data
4 :Value of N
' L' :Value of UPLO
(1.0, 0.0) (2.0, 1.0) (3.0, 1.0) (4.0, 1.0) :End of matrix A
(2.0, 0.0) (2.0, 0.0) (3.0, 2.0) (4.0, 2.0) :Value of JOB
(3.0, 0.0) (3.0, 0.0) (4.0, 3.0) (4.0, 3.0) :End of matrix A
(4.0, 0.0) (4.0, 0.0) (4.0, 0.0) :End of matrix A

9.3 Program Results

f08fqc Example Program Results

Eigenvalues
1  -4.2443
2  -0.6886
3   1.1412
4  13.7916

Eigenvectors
1  2   3   4
1  0.4836 0.6470 -0.4456 -0.3859
   0.0000 0.0000 0.0000 -0.0000
2  0.2912 -0.4984 -0.0230 -0.4441
   -0.3618 -0.1130 -0.5702 0.0156
3 -0.3163 0.2949 0.5331 -0.5173
   -0.3696 0.3165 0.1317 -0.0844
4 -0.4447 -0.2241 -0.3510 -0.5277
   0.3406 -0.2878 0.2261 -0.3168