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

\[ Az_i = \lambda_i z_i, \quad i = 1, 2, \ldots, n. \]

4 References

5 Arguments
1: \textit{order} – Nag_OrderType \hspace{1cm} \textit{Input}
   \textit{On entry}: the \textit{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 \textit{order} = Nag_RowMajor. See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.
   \textit{Constraint}: \textit{order} = Nag_RowMajor or Nag_ColMajor.

2: \textit{job} – Nag_JobType \hspace{1cm} \textit{Input}
   \textit{On entry}: indicates whether eigenvectors are computed.
   \textit{job} = Nag_DoNothing
       Only eigenvalues are computed.
job = Nag_EigVecs
     Eigenvalues and eigenvectors are computed.

Constraint: job = Nag_DoNothing or Nag_EigVecs.

3:  uplo = Nag_UploType
     Input

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

uplo = Nag_Upper
     The upper triangular part of A is stored.

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 ≥ 0.

5:  a[dim] = Complex
     Input/Output

Note: the dimension, dim, of the array a must be at least max(1, pda × n).

On entry: the n by n Hermitian matrix A.

If order = Nag_ColMajor, A_{ij} is stored in a[(j - 1) × pda + i - 1].

If order = Nag_RowMajor, A_{ij} is stored in a[(i - 1) × pda + j - 1].

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 uplo = Nag_Lower, the lower triangular part of A must be stored and the elements of the array above the diagonal are not referenced.

On exit: if job = Nag_EigVecs, a is overwritten by the unitary matrix Z which contains the eigenvectors of A.

6:  pda = Integer
     Input

On entry: the stride separating row or column elements (depending on the value of order) of the matrix A in the array a.

Constraint: pda ≥ max(1, n).

7:  w[dim] = double
     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.

8:  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 <value> had an illegal value.

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

NE_INT
On entry, n = <value>.
Constraint: n ≥ 0.
On entry, pda = <value>.
Constraint: pda > 0.

NE_INT_2
On entry, pda = <value> and n = <value>.
Constraint: pda ≥ max(1, n).

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_zheevd (f08fqc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zheevd (f08fqc) 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_dsyevd (f08fcc).
10  Example

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

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

The example program for nag_zheevd (f08fqc) illustrates the computation of error bounds for the eigenvalues and eigenvectors.

10.1  Program Text

/* nag_zheevd (f08fqc) Example Program. *
* * Copyright 2014 Numerical Algorithms Group. *
* * Mark 7, 2001.
*/
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>
#include <naga02.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 nag_enum_arg[40];
    double *w = 0;
    Complex *a = 0;
    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J - 1) * pda + I - 1]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I - 1) * pda + J - 1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    printf("nag_zheevd (f08fqc) Example Program Results\n\n");
    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n]");
    #else
    scanf("%*[\n]");
    #endif
    #ifdef _WIN32
    scanf_s("%"NAG_IFMT"%*[\n]", &n);
    #else
    scanf("%"NAG_IFMT"%*[\n]", &n);
    #endif
    pda = n;
    w_len = n;
    /* Allocate memory */
if (!(a = NAG_ALLOC(n * n, Complex)) ||
!(w = NAG_ALLOC(w_len, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

/* Read whether Upper or Lower part of A is stored */
#ifdef _WIN32
    scanf_s("%39s\n", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s\n", nag_enum_arg);
#endif

uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);

/* Read A from data file */
if (uplo == Nag_Upper)
{
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
#endif _WIN32
        scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#else
        scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif _WIN32
#endif _WIN32
    scanf_s("%*\n");
#else
    scanf("%*\n");
#endif _WIN32
else
{
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
#endif _WIN32
        scanf_s(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#else
        scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
#endif _WIN32
#endif _WIN32
    scanf_s("%*\n");
#else
    scanf("%*\n");
#endif _WIN32

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

job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);

/* Calculate all the eigenvalues and eigenvectors of A using
* nag_zheevd (f08fqc).  * All eigenvalues and optionally all eigenvectors of
* complex Hermitian matrix (divide-and-conquer)
*/

if (f08fqc("f08fqc", order, job, uplo, n, a, pda, w, &fail))
{
    printf("Error from nag_zheevd (f08fqc).\n");
    exit_status = 1;
    goto END;
}

/* Normalize the eigenvectors */
for(j=1; j<=n; j++)
{
  for(i=n; i>=1; i--)
  {
    A(i, j) = nag_complex_divide(A(i, j),A(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, a, pda, 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 = 2;
  goto END;
}
END:
NAG_FREE(a);
NAG_FREE(w);
return exit_status;

10.2 Program Data
nag_zheevd (f08fqc) Example Program Data
  4 : Value of n
  Nag_Lower : Value of uplo
 (1.0, 0.0) (2.0, 1.0) (3.0, 1.0) (4.0, 1.0) : End of matrix A
  Nag_EigVecs : Value of job

10.3 Program Results
nag_zheevd (f08fqc) Example Program Results

Eigenvalues
  1 -4.2443
  2 -0.6886
  3 1.1412
  4 13.7916

Eigenvalues
  1  1.0000  1.0000  1.0000  1.0000
     0.0000  0.0000 -0.0000 -0.0000
  2  0.6022 -0.7703  0.0516  1.1508
     -0.7483 -0.1746  1.2795 -0.0404
  3 -0.6540  0.4559 -1.1962  1.3404
     -0.7642  0.4892 -0.2954  0.2188
  4 -0.9197 -0.3464  0.7876  1.3674
     0.7044 -0.4448 -0.5075  0.8207