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

$$A = Z \Lambda Z^{\mathrm{H}},$$

where Λ is a real diagonal matrix whose diagonal elements are the eigenvalues λ_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

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_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: $\mathbf{n} \geq 0$.

5: **a**[dim] – Complex *Input/Output*

Note: the dimension, *dim*, of the array **a** must be at least $\max(1, \mathbf{pda} \times \mathbf{n})$.

On entry: the n by n Hermitian matrix A.

If **order** = 'Nag_ColMajor', A_{ij} is stored in $\mathbf{a}[(j-1) \times \mathbf{pda} + i - 1]$.

If **order** = 'Nag_RowMajor', A_{ij} is stored in $\mathbf{a}[(i-1) \times \mathbf{pda} + j - 1]$.

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: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

7: **w**[dim] – double *Output*

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

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.

NE_BAD_PARAM

On entry, argument ⟨*value*⟩ had an illegal value.

NE_CONVERGENCE

If **fail.errnum** $= \langle value \rangle$ and **job** $=$ Nag_DoNothing, the algorithm failed to converge; $\langle value \rangle$ elements of an intermediate tridiagonal form did not converge to zero; if **fail.errnum** $= \langle value \rangle$ and **job** $=$ Nag_EigVecs, then the algorithm failed to compute an eigenvalue while working on the submatrix lying in rows and column $\langle value \rangle/(\mathbf{n}+1)$ through $\langle value \rangle \bmod (\mathbf{n}+1)$.

NE_INT

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

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

NE_INT_2

On entry, $\mathbf{pda} = \langle value \rangle$ and $\mathbf{n} = \langle value \rangle$.
Constraint: $\mathbf{pda} \geq \max(1, \mathbf{n})$.

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.

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_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 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 2001 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 */
  scanf("%*[^\n]");
  scanf("%ld%*[^\n]", &n);
  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 */
  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);
  /* Read A from data file */
  if (uplo == Nag_Upper)
    {
      for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
          scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
      scanf("%*[^\n] ");
    }
  else
```

```
    {
      for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
          scanf(" ( %lf , %lf )", &A(i, j).re, &A(i, j).im);
      scanf("%*[^\n] ");
    }

  /* Read type of job to be performed */
  scanf("%39s%*[^\n]", nag_enum_arg);
  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)
   */
  nag_zheevd(order, job, uplo, n, a, pda, w, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf("Error from nag_zheevd (f08fqc).\n%s\n", fail.message);
      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("   %5ld     %8.4f\n", i + 1, w[i]);
  printf("\n");
  /* nag_gen_complx_mat_print_comp (x04dbc).
   * Print complex general matrix (comprehensive)
   */
  fflush(stdout);
  nag_gen_complx_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_complx_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)  (2.0, 0.0)
  (3.0, 1.0)  (3.0, 2.0)  (3.0, 0.0)
  (4.0, 1.0)  (4.0, 2.0)  (4.0, 3.0)  (4.0, 0.0)       :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

 Eigenvectors
         1       2       3       4
 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
```