

# NAG Library Function Document

## nag\_zheev (f08fnc)

### 1 Purpose

nag\_zheev (f08fnc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex  $n$  by  $n$  Hermitian matrix  $A$ .

### 2 Specification

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

void nag_zheev (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
               Integer n, Complex a[], Integer pda, double w[], NagError *fail)
```

### 3 Description

The Hermitian matrix  $A$  is first reduced to real tridiagonal form, using unitary similarity transformations, and then the  $QR$  algorithm is applied to the tridiagonal matrix to compute the eigenvalues and (optionally) the eigenvectors.

### 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 2.3.1.3 in How to Use the NAG Library and its Documentation 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\_EigVals  
     Only eigenvalues are computed.  
**job** = Nag\_DoBoth  
     Eigenvalues and eigenvectors are computed.  
*Constraint:* **job** = Nag\_EigVals or Nag\_DoBoth.
- 3: **uplo** – Nag\_UploType *Input*  
*On entry:* if **uplo** = Nag\_Upper, the upper triangular part of  $A$  is stored.

If **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:*  $n \geq 0$ .

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

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

*On entry:* the  $n$  by  $n$  Hermitian matrix  $A$ .

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

If **order** = Nag\_RowMajor,  $A_{ij}$  is stored in **a**[( $i - 1$ )  $\times$  **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\_DoBoth, then **a** contains the orthonormal eigenvectors of the matrix  $A$ .

If **job** = Nag\_EigVals, then on exit the lower triangle (if **uplo** = Nag\_Lower) or the upper triangle (if **uplo** = Nag\_Upper) of **a**, including the diagonal, is overwritten.

6: **pda** – Integer *Input*

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

*Constraint:*  $pda \geq \max(1, n)$ .

7: **w**[**n**] – double *Output*

*On exit:* the eigenvalues in ascending order.

8: **fail** – NagError \* *Input/Output*

The NAG error argument (see Section 2.7 in How to Use the NAG Library and its Documentation).

## 6 Error Indicators and Warnings

### NE\_ALLOC\_FAIL

Dynamic memory allocation failed.

See Section 2.3.1.2 in How to Use the NAG Library and its Documentation for further information.

### NE\_BAD\_PARAM

On entry, argument  $\langle value \rangle$  had an illegal value.

### NE\_CONVERGENCE

The algorithm failed to converge;  $\langle value \rangle$  off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

**NE\_INT**

On entry, **n** =  $\langle value \rangle$ .

Constraint: **n**  $\geq 0$ .

On entry, **pda** =  $\langle value \rangle$ .

Constraint: **pda**  $> 0$ .

**NE\_INT\_2**

On entry, **pda** =  $\langle value \rangle$  and **n** =  $\langle value \rangle$ .

Constraint: **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.

An unexpected error has been triggered by this function. Please contact NAG.

See Section 2.7.6 in How to Use the NAG Library and its Documentation for further information.

**NE\_NO\_LICENCE**

Your licence key may have expired or may not have been installed correctly.

See Section 2.7.5 in How to Use the NAG Library and its Documentation 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\_zheev (f08fnc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag\_zheev (f08fnc) 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**

Each eigenvector is normalized so that the element of largest absolute value is real.

The total number of floating-point operations is proportional to  $n^3$ .

The real analogue of this function is nag\_dsyev (f08fac).

## 10 Example

This example finds all the eigenvalues and eigenvectors of the Hermitian matrix

$$A = \begin{pmatrix} 1 & 2-i & 3-i & 4-i \\ 2+i & 2 & 3-2i & 4-2i \\ 3+i & 3+2i & 3 & 4-3i \\ 4+i & 4+2i & 4+3i & 4 \end{pmatrix},$$

together with approximate error bounds for the computed eigenvalues and eigenvectors.

### 10.1 Program Text

```

/* nag_zheev (f08fnc) Example Program.
 *
 * NAGPRODCODE Version.
 *
 * Copyright 2016 Numerical Algorithms Group.
 *
 * Mark 26, 2016.
 */

#include <math.h>
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <naga02.h>
#include <nagf08.h>
#include <nagx02.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double eerrbd, eps;
    Integer exit_status = 0, i, j, n, pda;
    /* Arrays */
    Complex *a = 0;
    double *rcondz = 0, *w = 0, *zerrbd = 0;
    /* Nag Types */
    Nag_OrderType order;
    NagError fail;

#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_zheev (f08fnc) 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

    /* Allocate memory */
    if (!(a = NAG_ALLOC(n * n, Complex)) ||

```

```

        !(rcondz = NAG_ALLOC(n, double)) ||
        !(w = NAG_ALLOC(n, double)) || !(zerrbd = NAG_ALLOC(n, double)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }

#ifdef NAG_COLUMN_MAJOR
    pda = n;
#else
    pda = n;
#endif

    /* Read the upper triangular part of the matrix A from data file */
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
#ifdef _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
#ifdef _WIN32
        scanf_s("%*[\n]");
#else
        scanf("%*[\n]");
#endif

    /* nag_zheev (f08fnc).
     * Solve the Hermitian eigenvalue problem.
     */
    nag_zheev(order, Nag_DoBoth, Nag_Upper, n, a, pda, w, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_zheev (f08fnc).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

    /* nag_complex_divide (a02cdc).
     * 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 solution */
    printf("Eigenvalues\n");
    for (j = 0; j < n; ++j)
        printf("%8.4f%s", w[j], (j + 1) % 8 == 0 ? "\n" : " ");
    printf("\n\n");

    /* nag_gen_complx_mat_print (x04dac).
     * Print eigenvectors.
     */
    fflush(stdout);
    nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n,
                             n, a, pda, "Eigenvectors", 0, &fail);
    if (fail.code != NE_NOERROR) {
        printf("Error from nag_gen_complx_mat_print (x04dac).\n%s\n",
              fail.message);
        exit_status = 1;
        goto END;
    }

    /* Get the machine precision, eps, using nag_machine_precision (X02AJC)
     * and compute the approximate error bound for the computed eigenvalues.
     * Note that for the 2-norm, ||A|| = max { |w[i]|, i=0..n-1 }, and since
     * the eigenvalues are in ascending order ||A|| = max( |w[0]|, |w[n-1]| ).
     */
    eps = nag_machine_precision;
    eerrbd = eps * MAX(fabs(w[0]), fabs(w[n - 1]));

```

```

/* nag_ddisna (f08flc).
 * Estimate reciprocal condition numbers for the eigenvectors.
 */
nag_ddisna(Nag_EigVecs, n, n, w, rcondz, &fail);
if (fail.code != NE_NOERROR) {
    printf("Error from nag_ddisna (f08flc).\n%s\n", fail.message);
    exit_status = 1;
    goto END;
}

/* Compute the error estimates for the eigenvectors */
for (i = 0; i < n; ++i)
    zerrbd[i] = eerrbd / rcondz[i];

/* Print the approximate error bounds for the eigenvalues and vectors */
printf("\nError estimate for the eigenvalues\n");
printf("%11.1e\n\n", eerrbd);

printf("Error estimates for the eigenvectors\n");
for (i = 0; i < n; ++i)
    printf("%11.1e%s", zerrbd[i], (i + 1) % 6 == 0 ? "\n" : " ");

END:
    NAG_FREE(a);
    NAG_FREE(rcondz);
    NAG_FREE(w);
    NAG_FREE(zerrbd);

    return exit_status;
}

#undef A

```

## 10.2 Program Data

nag\_zheev (f08fnc) Example Program Data

```

4                                     :Value of n

(1.0, 0.0) (2.0, -1.0) (3.0, -1.0) (4.0, -1.0)
          (2.0, 0.0) (3.0, -2.0) (4.0, -2.0)
                    (3.0, 0.0) (4.0, -3.0)
                              (4.0, 0.0) :End of matrix A

```

## 10.3 Program Results

nag\_zheev (f08fnc) Example Program Results

```

Eigenvalues
-4.2443 -0.6886  1.1412 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

```

Error estimate for the eigenvalues  
1.5e-15

Error estimates for the eigenvectors  
4.3e-16      8.4e-16      8.4e-16      1.2e-16

---