NAG Library Function Document

nag_zhpev (f08gnc)

1 Purpose

nag_zhpev (f08gnc) computes all the eigenvalues and, optionally, all the eigenvectors of a complex \( n \) by \( n \) Hermitian matrix \( A \) in packed storage.

2 Specification

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

void nag_zhpev (Nag_OrderType order, Nag_JobType job, Nag_UploType uplo,
                Integer n, Complex ap[], double w[], Complex z[], Integer pdz,
                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


5 Arguments

1: \( \text{order} \) – Nag_OrderType

_**Input**_

_On entry:_ the \( \text{order} \) 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 \( \text{order} = \text{Nag_RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

_Constraint:_ \( \text{order} = \text{Nag_RowMajor} \) or \( \text{Nag_ColMajor} \).

2: \( \text{job} \) – Nag_JobType

_**Input**_

_On entry:_ indicates whether eigenvectors are computed.

\( \text{job} = \text{Nag_EigVals} \)

Only eigenvalues are computed.

\( \text{job} = \text{Nag_DoBoth} \)

Eigenvalues and eigenvectors are computed.

_Constraint:_ \( \text{job} = \text{Nag_EigVals} \) or \( \text{Nag_DoBoth} \).

3: \( \text{uplo} \) – Nag_UploType

_**Input**_

_On entry:_ if \( \text{uplo} = \text{Nag_Upper} \), the upper triangular part of \( A \) is stored.
If \( \text{uplo} = \text{Nag}_\text{Lower} \), the lower triangular part of \( A \) is stored.

\text{Constraint: } \text{uplo} \text{=} \text{Nag}_\text{Upper} \text{ or } \text{Nag}_\text{Lower}.

4: \quad n \rightarrow \text{Integer} \quad \text{Input}

\text{On entry: } n, \text{ the order of the matrix } A.

\text{Constraint: } n \geq 0.

5: \quad \text{ap}[\text{dim}] \rightarrow \text{Complex} \quad \text{Input/Output}

\text{Note: } the \text{ dimension, } \text{dim}, \text{ of the array } \text{ap} \text{ must be at least } \max(1, n \times (n + 1)/2).

\text{On entry: } the \text{ upper or lower triangle of the } n \text{ by } n \text{ Hermitian matrix } A, \text{ packed by rows or columns.}

The storage of elements \( A_{ij} \) depends on the \text{order} and \text{uplo} arguments as follows:

- if \( \text{order} = \text{Nag}_\text{ColMajor} \text{ and } \text{uplo} = \text{Nag}_\text{Upper}, \)
  \( A_{ij} \) is stored in \( \text{ap}[(j - 1) \times j/2 + i - 1], \text{ for } i \leq j; \)

- if \( \text{order} = \text{Nag}_\text{ColMajor} \text{ and } \text{uplo} = \text{Nag}_\text{Lower}, \)
  \( A_{ij} \) is stored in \( \text{ap}[(2n - j) \times (j - 1)/2 + i - 1], \text{ for } i \geq j; \)

- if \( \text{order} = \text{Nag}_\text{RowMajor} \text{ and } \text{uplo} = \text{Nag}_\text{Upper}, \)
  \( A_{ij} \) is stored in \( \text{ap}[(2n - i) \times (i - 1)/2 + j - 1], \text{ for } i \leq j; \)

- if \( \text{order} = \text{Nag}_\text{RowMajor} \text{ and } \text{uplo} = \text{Nag}_\text{Lower}, \)
  \( A_{ij} \) is stored in \( \text{ap}[(i - 1) \times i/2 + j - 1], \text{ for } i \geq j. \)

\text{On exit: } \text{ap} \text{ is overwritten by the values generated during the reduction to tridiagonal form. The elements of the diagonal and the off-diagonal of the tridiagonal matrix overwrite the corresponding elements of } A.

6: \quad \text{w}[n] \rightarrow \text{double} \quad \text{Output}

\text{On exit: } \text{the eigenvalues in ascending order.}

7: \quad \text{z}[\text{dim}] \rightarrow \text{Complex} \quad \text{Output}

\text{Note: } the \text{ dimension, } \text{dim}, \text{ of the array } \text{z} \text{ must be at least}

\max(1, \text{pdz} \times n) \text{ when } \text{job} = \text{Nag}_\text{DoBoth};

1 \text{ otherwise.}

The \((i, j)\)th element of the matrix \( Z \) is stored in

\( z[(j - 1) \times \text{pdz} + i - 1] \text{ when } \text{order} = \text{Nag}_\text{ColMajor}; \)

\( z[(i - 1) \times \text{pdz} + j - 1] \text{ when } \text{order} = \text{Nag}_\text{RowMajor}. \)

\text{On exit: if } \text{job} = \text{Nag}_\text{DoBoth}, \text{ z contains the orthonormal eigenvectors of the matrix } A, \text{ with the}

\text{i}th \text{ column of } Z \text{ holding the eigenvector associated with } \text{w}[i - 1].

\text{If } \text{job} = \text{Nag}_\text{EigVals}, \text{ z is not referenced.}

8: \quad \text{pdz} \rightarrow \text{Integer} \quad \text{Input}

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

\text{Constraints:}

- if \( \text{job} = \text{Nag}_\text{DoBoth}, \text{ pdz} \geq \max(1, n); \)

- otherwise \( \text{pdz} \geq 1. \)

9: \quad \text{fail} \rightarrow \text{NagError} \quad \text{Input/Output}

\text{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**
The algorithm failed to converge; ⟨value⟩ off-diagonal elements of an intermediate tridiagonal form did not converge to zero.

**NE_ENUM_INT_2**
On entry, job = ⟨value⟩, pdz = ⟨value⟩ and n = ⟨value⟩.
Constraint: if job = Nag_DoBoth, pdz ≥ max(1, n); otherwise pdz ≥ 1.

**NE_INT**
On entry, n = ⟨value⟩.
Constraint: n ≥ 0.
On entry, pdz = ⟨value⟩.
Constraint: pdz > 0.

**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_zhpev (f08gnc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_zhpev (f08gnc) 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 and positive.

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

The real analogue of this function is nag_dspev (f08gac).

10 Example

This example finds all the eigenvalues 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.

10.1 Program Text

/* nag_zhpev (f08gnc) Example Program. */
/* Copyright 2014 Numerical Algorithms Group. */
/* * Mark 23, 2011. */

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

int main(void)
{
    /* Scalars */
    double eerrbd, eps;
    Integer exit_status = 0, i, j, n;
    /* Arrays */
    char nag_enum_arg[40];
    Complex *ap = 0, *dummy = 0;
    double *w = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_UploType uplo;
    NagError fail;

    #ifdef NAG_COLUMN_MAJOR
    #define AP_UPPER(I, J) ap[J * (J - 1) /2+I-1 ]
    #define AP_LOWER(I, J) ap[(2 *n-J)*( J-1 )/2+I-1 ]
    #else
    #define AP_LOWER(I, J) ap[I * (I - 1)/2+J-1 ]
    #define AP_UPPER(I, J) ap[(2 *n-I)*( I-1 )/2+J-1 ]
    #endif

    INIT_FAIL(fail);
    printf("nag_zhpev (f08gnc) Example Program Results\n\n");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s("%*[\n]");
    #else
    scanf("%*[\n]");
    
"}
/* Read uplo */
#ifdef _WIN32
    scanf_s("%39s%[\n]", nag_enum_arg, _countof(nag_enum_arg));
#else
    scanf("%39s%[\n]", nag_enum_arg);
#endif
/* nag_enum_name_to_value (x04nac).
 * Converts NAG enum member name to value */
uplo = (Nag_UploType) nag_enum_name_to_value(nag_enum_arg);
/* Allocate memory */
if (!(ap = NAG_ALLOC(n*(n+1)/2, Complex)) ||
    !(dummy = NAG_ALLOC(1, Complex)) ||
    !(w = NAG_ALLOC(n, double)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}
/* Read the upper or lower triangular part of the matrix A from data file */
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
#ifdef _WIN32
            scanf_s( " ( %lf , %lf )", &AP_UPPER(i, j).re, &AP_UPPER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &AP_UPPER(i, j).re, &AP_UPPER(i, j).im);
#endif
#ifdef _WIN32
            scanf_s("%[\n]");
#else
            scanf("%*[\n]"Fl);} 
#endif
#else if (uplo == Nag_Lower) {
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
#ifdef _WIN32
            scanf_s( " ( %lf , %lf )", &AP_LOWER(i, j).re, &AP_LOWER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &AP_LOWER(i, j).re, &AP_LOWER(i, j).im);
#endif
#ifdef _WIN32
            scanf_s("%[\n]"Fl);} 
#else
            scanf("%*[\n]"Fl);} 
#endif
/* nag_zhpev (f08gnc).
 * Solve the Hermitian eigenvalue problem.
 */
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhpev (f08gnc).\n", fail.message);
    exit_status = 1;
    goto END;
}
/* Print solution */
printf("Eigenvalues\n");
for (j = 0; j < n; ++j)
    printf("%8.4f%s", w[j], (j+1)%8 == 0?"\n":" ");
printf("\n");

/* 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]));

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

END:
    NAG_FREE(ap);
    NAG_FREE(dummy);
    NAG_FREE(w);

    return exit_status;
}
#endif AP_UPPER
#endif AP_LOWER

10.2 Program Data
nag_zhpev (f08gnc) Example Program Data

    4 :Value of n
    Nag_Upper :Value of uplo

    (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_zhpev (f08gnc) Example Program Results

    Eigenvalues
    -4.2443  -0.6886   1.1412  13.7916

    Error estimate for the eigenvalues
    1.5e-15