NAG Library Function Document

nag_zhpevx (f08gpc)

1 Purpose

nag_zhpevx (f08gpc) computes selected eigenvalues and, optionally, eigenvectors of a complex $n$ by $n$ Hermitian matrix $A$ in packed storage. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

2 Specification

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

void nag_zhpevx (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
     Nag_UploType uplo, Integer n, Complex ap[ ], double vl, double vu,
     Integer il, Integer iu, double abstol, Integer *m, double w[ ],
     Complex z[ ], Integer pdz, Integer jfail[ ], NagError *fail)
```

3 Description

The Hermitian matrix $A$ is first reduced to real tridiagonal form, using unitary similarity transformations. The required eigenvalues and eigenvectors are then computed from the tridiagonal matrix; the method used depends upon whether all, or selected, eigenvalues and eigenvectors are required.

4 References


5 Arguments

1:  
   **order** – Nag_OrderType

   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

   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:  **range** – Nag.RangeType  
*Input*  
*On entry:* if *range* = Nag.AllValues, all eigenvalues will be found.  
If *range* = Nag.Interval, all eigenvalues in the half-open interval \( [vl, vu] \) will be found.  
If *range* = Nag.Indices, the \( il \)th to \( iu \)th eigenvalues will be found.  
*Constraint:* *range* = Nag.AllValues, Nag.Interval or Nag.Indices.

4:  **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.

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

6:  **ap[dim]** – Complex  
*Input/Output*  
*Note:* the dimension, \( dim \), of the array \( ap \) must be at least \( \max(1, n \times (n + 1)/2) \).  
*On entry:* the upper or lower triangle of the \( n \) by \( n \) Hermitian matrix \( A \), packed by rows or columns.  
The storage of elements \( A_{ij} \) depends on the *order* and *uplo* arguments as follows:  
if *order* = Nag.ColMajor and *uplo* = Nag.Upper,  
\( A_{ij} \) is stored in \( ap[(j-1) \times j/2 + i - 1] \), for \( i \leq j \);  
if *order* = Nag.ColMajor and *uplo* = Nag.Lower,  
\( A_{ij} \) is stored in \( ap[(2n-j) \times (j-1)/2 + i - 1] \), for \( i \geq j \);  
if *order* = Nag.RowMajor and *uplo* = Nag.Upper,  
\( A_{ij} \) is stored in \( ap[(2n-i) \times (i-1)/2 + j - 1] \), for \( i \leq j \);  
if *order* = Nag.RowMajor and *uplo* = Nag.Lower,  
\( A_{ij} \) is stored in \( ap[(i-1) \times i/2 + j - 1] \), for \( i \geq j \).  
*On exit:* \( ap \) 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 \( A \).

7:  **vl** – double  
*Input*  
*On entry:* if *range* = Nag.Interval, the lower and upper bounds of the interval to be searched for eigenvalues.  
If *range* = Nag.AllValues or Nag.Indices, \( vl \) and \( vu \) are not referenced.  
*Constraint:* if *range* = Nag.Interval, \( vl < vu \).

8:  **vu** – double  
*Input*  
*On entry:* if *range* = Nag.Interval, the lower and upper bounds of the interval to be searched for eigenvalues.  
If *range* = Nag.AllValues or Nag.Indices, \( vl \) and \( vu \) are not referenced.  
*Constraint:* if *range* = Nag.Interval, \( vl < vu \).

9:  **il** – Integer  
*Input*  
*On entry:* if *range* = Nag.Indices, the indices (in ascending order) of the smallest and largest eigenvalues to be returned.  
If *range* = Nag.AllValues or Nag.Interval, \( il \) and \( iu \) are not referenced.  
*Constraints:*  
if *range* = Nag.Indices and \( n = 0 \), \( il = 1 \) and \( iu = 0 \);  
if *range* = Nag.Indices and \( n > 0 \), \( 1 \leq il \leq iu \leq n \).
11:  abstol – double

   Input

   On entry: the absolute error tolerance for the eigenvalues. An approximate eigenvalue is accepted as converged when it is determined to lie in an interval \([a, b]\) of width less than or equal to

   \(\text{abstol} + \epsilon \max(|a|, |b|)\),

   where \(\epsilon\) is the \textit{machine precision}. If \text{abstol} is less than or equal to zero, then \(\epsilon \|T\|_1\) will be used in its place, where \(T\) is the tridiagonal matrix obtained by reducing \(A\) to tridiagonal form. Eigenvalues will be computed most accurately when \text{abstol} is set to twice the underflow threshold \(2 \times \text{nag\_real\_safe\_small\_number}\), not zero. If this function returns with \text{fail\_code} = \text{NE\_CONVERGENCE}, indicating that some eigenvectors did not converge, try setting \text{abstol} to \(2 \times \text{nag\_real\_safe\_small\_number}\). See Demmel and Kahan (1990).

12:  m – Integer

   Output

   On exit: the total number of eigenvalues found. \(0 \leq m \leq n\).

   If \text{range} = \text{Nag\_AllValues}, \(m = n\).

   If \text{range} = \text{Nag\_Indices}, \(m = \text{iu} - \text{il} + 1\).

13:  w[n] – double

   Output

   On exit: the selected eigenvalues in ascending order.

14:  z[dim] – Complex

   Output

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

   \[\max(1, \text{pdz} \times \text{n})\] when \text{job} = \text{Nag\_DoBoth};

   \[1\] otherwise.

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

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

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

   On exit: if \text{job} = \text{Nag\_DoBoth}, then

   if \text{fail\_code} = \text{NE\_NOERROR}, the first \(m\) columns of \(Z\) contain the orthonormal eigenvectors of the matrix \(A\) corresponding to the selected eigenvalues, with the \(i\)th column of \(Z\) holding the eigenvector associated with \(w[i - 1]\);

   if an eigenvector fails to converge (\text{fail\_code} = \text{NE\_CONVERGENCE}), then that column of \(Z\) contains the latest approximation to the eigenvector, and the index of the eigenvector is returned in \text{jffail}.

   If \text{job} = \text{Nag\_EigVals}, \(z\) is not referenced.

15:  pdz – Integer

   Input

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

   Constraints:

   \[
   \begin{align*}
   \text{if} \quad & \text{job} = \text{Nag\_DoBoth}, \quad \text{pdz} \geq \max(1, \text{n});
   \\
   \text{otherwise} \quad & \text{pdz} \geq 1.
   \end{align*}
   \]

16:  jfail[dim] – Integer

   Output

   Note: the dimension, \(\text{dim}\), of the array \(\text{jffail}\) must be at least \(\max(1, \text{n})\).

   On exit: if \text{job} = \text{Nag\_DoBoth}, then
if fail.code = NE_NOERROR, the first m elements of jfail are zero;
if fail.code = NE_CONVERGENCE, jfail contains the indices of the eigenvectors that failed to converge.

If job = Nag_EigVals, jfail is not referenced.

17: fail – NagError *

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> eigenvectors did not converge. Their indices are stored in array jfail.

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_ENUM_INT_3
On entry, range = <value>, il = <value>, iu = <value> and n = <value>.
Constraint: if range = Nag_Indices and n = 0, il = 1 and iu = 0;
if range = Nag_Indices and n > 0, 1 ≤ il ≤ iu ≤ n.

NE_ENUM_REAL_2
On entry, range = <value>, vl = <value> and vu = <value>.
Constraint: if range = Nag_Interval, vl < vu.

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

\texttt{nag\_zhpevx (f08gpc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\texttt{nag\_zhpevx (f08gpc)} 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 total number of floating-point operations is proportional to \( n^3 \).

The real analogue of this function is \texttt{nag\_dspevx (f08gbc)}.

10 Example

This example finds the eigenvalues in the half-open interval \((-2, 2]\), and the corresponding eigenvectors, of the Hermitian matrix

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

10.1 Program Text

/* \texttt{nag\_zhpevx (f08gpc)} Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011.
*/
#include <stdio.h>
#include <nag.h>
#include <naq_stdlib.h>
#include <naga02.h>
#include <nagf08.h>
#include <nagx04.h>

int main(void)
{
    /* Scalars */
    double abstol, vl, vu;
    Integer exit_status = 0, i, il = 0, iu = 0, j, m, n, pdz;
    /* Arrays */
    char nag_enum_arg[40];
    Complex *ap = 0, *z = 0;
    double *w = 0;
    Integer *index = 0;
    /* Nag Types */
    Nag_OrderType order;
Nag_UploType uplo;
NagError fail, fail_print;

#ifndef 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]
#define Z(I, J) z[(J - 1) * pdz + I - 1]

order = Nag_ColMajor;
#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]
#define Z(I, J) z[(I - 1) * pdz + J - 1]

order = Nag_RowMajor;
#endif

INIT_FAIL(fail);

printf("nag_zhpevx (f08gpc) 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

/* Read uplo */
#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);

/* Allocate memory */
if (!(ap = NAG_ALLOC(n*(n+1)/2, Complex)) ||
    !(z = NAG_ALLOC(n*n, Complex)) ||
    !(w = NAG_ALLOC(n, double)) ||
    !(index = NAG_ALLOC(n, Integer)))
{
    printf("Allocation failure\n");
    exit_status = -1;
    goto END;
}

pdz = n;

/* Read the lower and upper bounds of the interval to be searched, and *
* read the upper or lower triangular part of the matrix A from data file. */
#ifdef _WIN32
scanf_s("%lf%lf%*[\n"]", &vl, &vu);
#else
scanf("%lf%lf%*[\n"]", &vl, &vu);
#endif

if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= n; ++j)
            scanf(" ( %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
f08gpc
NAG Library Manual
f08gpc.6 Mark 25
```c
#ifndef _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
#endif
```
printf("%8"NAG_IFMT"%s", index[j],(j+1)%8 == 0?"\n":" ");

} END:
NAG_FREE(ap);
NAG_FREE(z);
NAG_FREE(w);
NAG_FREE(index);
return exit_status;
}

#define AP_UPPER
#define AP_LOWER
#define Z

10.2 Program Data

nag_zhpevx (f08gpc) Example Program Data

<table>
<thead>
<tr>
<th>Value of n</th>
<th>Value of uplo</th>
<th>Values of vl and vu</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Nag_Lower</td>
<td>(1.0, 0.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(2.0, 1.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(3.0, 1.0)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(4.0, 1.0)</td>
</tr>
</tbody>
</table>

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

10.3 Program Results

nag_zhpevx (f08gpc) Example Program Results

Number of eigenvalues found = 2

Eigenvalues
-0.6886  1.1412

Selected eigenvectors

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.0000</td>
</tr>
<tr>
<td>2</td>
<td>-0.7703</td>
</tr>
<tr>
<td></td>
<td>-0.1746</td>
</tr>
<tr>
<td>3</td>
<td>0.4559</td>
</tr>
<tr>
<td></td>
<td>0.4892</td>
</tr>
<tr>
<td>4</td>
<td>-0.3464</td>
</tr>
<tr>
<td></td>
<td>-0.4448</td>
</tr>
</tbody>
</table>