NAG Library Function Document

**nag_zhegvx (f08spc)**

### 1 Purpose

`nag_zhegvx (f08spc)` computes selected eigenvalues and, optionally, eigenvectors of a complex generalized Hermitian-definite eigenproblem, of the form

\[ Az = \lambda Bz, \quad ABz = \lambda z \quad \text{or} \quad BAz = \lambda z, \]

where \( A \) and \( B \) are Hermitian and \( B \) is also positive definite. 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_zhegvx (Nag_OrderType order, Integer itype, Nag_JobType job, 
                Nag_RangeType range, Nag_UploType uplo, Integer n, Complex a[], 
                Integer pda, Complex b[], Integer pdb, double vl, double vu, Integer il, 
                Integer iu, double abstol, Integer *m, double w[], Complex z[], 
                Integer pdz, Integer *jfail, NagError *fail)
```

### 3 Description

`nag_zhegvx (f08spc)` first performs a Cholesky factorization of the matrix \( B \) as \( B = U^H U \), when \( \text{uplo} = \text{Nag\_Upper} \) or \( B = LL^H \), when \( \text{uplo} = \text{Nag\_Lower} \). The generalized problem is then reduced to a standard symmetric eigenvalue problem

\[ Cx = \lambda x, \]

which is solved for the desired eigenvalues and eigenvectors; the eigenvectors are then backtransformed to give the eigenvectors of the original problem.

For the problem \( Az = \lambda Bz \), the eigenvectors are normalized so that the matrix of eigenvectors, \( Z \), satisfies

\[ Z^H AZ = \Lambda \quad \text{and} \quad Z^H BZ = I, \]

where \( \Lambda \) is the diagonal matrix whose diagonal elements are the eigenvalues. For the problem \( ABz = \lambda z \) we correspondingly have

\[ Z^{-1} A Z^{-H} = \Lambda \quad \text{and} \quad Z^H B Z = I, \]

and for \( BAz = \lambda z \) we have

\[ Z^H A Z = \Lambda \quad \text{and} \quad Z^H B^{-1} Z = I. \]

### 4 References


5 Arguments

1: \( \text{order} \) – Nag_OrderType \hspace{1cm} \text{Input}

\text{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 = \text{Nag\_RowMajor}. \ See \ Section \ 3.2.1.3 \ in \ the \ Essential \ Introduction \ for \ a \ more \ detailed \ explanation \ of \ the \ use \ of \ this \ argument.

\text{Constraint:} \ order = \text{Nag\_RowMajor} \ or \ \text{Nag\_ColMajor}.

2: \( \text{itype} \) – Integer \hspace{1cm} \text{Input}

\text{On entry:} \ specifies \ the \ problem \ type \ to \ be \ solved.

\( \text{itype} = 1 \)
\( A z = \lambda Bz. \)

\( \text{itype} = 2 \)
\( ABz = \lambda z. \)

\( \text{itype} = 3 \)
\( BAz = \lambda z. \)

\text{Constraint:} \ itype = 1, 2 \ or \ 3.

3: \( \text{job} \) – Nag_JobType \hspace{1cm} \text{Input}

\text{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.

\text{Constraint:} \ job = \text{Nag\_EigVals} \ or \ \text{Nag\_DoBoth}.

4: \( \text{range} \) – Nag_RangeType \hspace{1cm} \text{Input}

\text{On entry:} \ if \ range = \text{Nag\_AllValues}, \ all \ eigenvalues \ will \ be \ found.

\text{If} \ range = \text{Nag\_Interval}, \ all \ eigenvalues \ in \ the \ half-open \ interval \ (vl, vu] \ will \ be \ found.

\text{If} \ range = \text{Nag\_Indices}, \ the \ ith \ to \ jth \ eigenvalues \ will \ be \ found.

\text{Constraint:} \ range = \text{Nag\_AllValues}, \ \text{Nag\_Interval} \ or \ \text{Nag\_Indices}.

5: \( \text{uplo} \) – Nag_UploType \hspace{1cm} \text{Input}

\text{On entry:} \ if \ uplo = \text{Nag\_Upper}, \ the \ upper \ triangles \ of \ A \ and \ B \ are \ stored.

\text{If} \ uplo = \text{Nag\_Lower}, \ the \ lower \ triangles \ of \ A \ and \ B \ are \ stored.

\text{Constraint:} \ uplo = \text{Nag\_Upper} \ or \ \text{Nag\_Lower}.

6: \( n \) – Integer \hspace{1cm} \text{Input}

\text{On entry:} \ n, \ the \ order \ of \ the \ matrices \ A \ and \ B.

\text{Constraint:} \ n \geq 0.

7: \( a[dim] \) – Complex \hspace{1cm} \text{Input/Output}

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

\text{On entry:} \ the \ n \ by \ n \ Hermitian \ matrix \ A.

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

If \( \text{uplo} = \text{Nag\_Upper} \), the upper triangular part of \( A \) must be stored and the elements of the array below the diagonal are not referenced.

If \( \text{uplo} = \text{Nag\_Lower} \), the lower triangular part of \( A \) must be stored and the elements of the array above the diagonal are not referenced.

On exit: the lower triangle (if \( \text{uplo} = \text{Nag\_Lower} \)) or the upper triangle (if \( \text{uplo} = \text{Nag\_Upper} \)) of \( A \), including the diagonal, is overwritten.

8: \( \text{pda} \) – Integer
   
   Input
   
   On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( A \).
   
   Constraint: \( \text{pda} \geq \max(1,n) \).

9: \( b[\text{dim}] \) – Complex
   
   Input/Output
   
   Note: the dimension, \( \text{dim} \), of the array \( b \) must be at least \( \max(1,\text{pdb} \times n) \).
   
   On entry: the \( n \) by \( n \) Hermitian matrix \( B \).
   
   If \( \text{order} = \text{Nag\_ColMajor} \), \( B_{ij} \) is stored in \( b[(j - 1) \times \text{pdb} + i - 1] \).
   
   If \( \text{order} = \text{Nag\_RowMajor} \), \( B_{ij} \) is stored in \( b[(i - 1) \times \text{pdb} + j - 1] \).
   
   If \( \text{uplo} = \text{Nag\_Upper} \), the upper triangular part of \( B \) must be stored and the elements of the array below the diagonal are not referenced.
   
   If \( \text{uplo} = \text{Nag\_Lower} \), the lower triangular part of \( B \) must be stored and the elements of the array above the diagonal are not referenced.
   
   On exit: the triangular factor \( U \) or \( L \) from the Cholesky factorization \( B = U^H U \) or \( B = LL^H \).

10: \( \text{pdb} \) – Integer
    
    Input
    
    On entry: the stride separating row or column elements (depending on the value of \( \text{order} \)) in the array \( B \).
    
    Constraint: \( \text{pdb} \geq \max(1,n) \).

11: \( \text{vl} \) – double
    
    Input
    
    On entry: if \( \text{range} = \text{Nag\_Interval} \), the lower and upper bounds of the interval to be searched for eigenvalues.
    
    If \( \text{range} = \text{Nag\_AllValues} \) or \( \text{Nag\_Indices} \), \( \text{vl} \) and \( \text{vu} \) are not referenced.
    
    Constraint: if \( \text{range} = \text{Nag\_Interval} \), \( \text{vl} < \text{vu} \).

12: \( \text{vu} \) – double
    
    Input
    
    On entry: if \( \text{range} = \text{Nag\_Interval} \), the lower and upper bounds of the interval to be searched for eigenvalues.
    
    If \( \text{range} = \text{Nag\_AllValues} \) or \( \text{Nag\_Indices} \), \( \text{vl} \) and \( \text{vu} \) are not referenced.
    
    Constraint: if \( \text{range} = \text{Nag\_Interval} \), \( \text{vl} < \text{vu} \).

13: \( \text{il} \) – Integer
    
    Input
    
    On entry: if \( \text{range} = \text{Nag\_Indices} \), the indices (in ascending order) of the smallest and largest eigenvalues to be returned.
    
    If \( \text{range} = \text{Nag\_AllValues} \) or \( \text{Nag\_Interval} \), \( \text{il} \) and \( \text{iu} \) are not referenced.
    
    Constraints:
    
    if \( \text{range} = \text{Nag\_Indices} \) and \( n = 0 \), \( \text{il} = 1 \) and \( \text{iu} = 0 \);
    
    if \( \text{range} = \text{Nag\_Indices} \) and \( n > 0 \), \( 1 \leq \text{il} \leq \text{iu} \leq n \).
15: \texttt{abstol} – double 
\textit{Input}

\textit{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 \texttt{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 \(C\) to tridiagonal form. Eigenvalues will be computed most accurately when \texttt{abstol} is set to twice the underflow threshold \(2 \times \text{nag\_real\_safe\_small\_number( )}\), not zero. If this function returns with \texttt{fail\_code} = \text{NE\_CONVERGENCE}, indicating that some eigenvectors did not converge, try setting \texttt{abstol} to \(2 \times \text{nag\_real\_safe\_small\_number( )}\). See Demmel and Kahan (1990).

16: \texttt{m} – Integer * 
\textit{Output}

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

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

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

17: \texttt{w[n]} – double 
\textit{Output}

\textit{On exit}: the first \(m\) elements contain the selected eigenvalues in ascending order.

18: \texttt{z[dim]} – Complex 
\textit{Output}

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

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

\[
1 \quad \text{otherwise}.
\]

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

\[
\begin{align*}
Z[(j - 1) \times \texttt{pdz} + i - 1] \quad &\text{when} \quad \texttt{order} = \text{Nag\_ColMajor}; \\
Z[(i - 1) \times \texttt{pdz} + j - 1] \quad &\text{when} \quad \texttt{order} = \text{Nag\_RowMajor}.
\end{align*}
\]

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

if \texttt{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 \texttt{w[i - 1]}. The eigenvectors are normalized as follows:

\[
\begin{align*}
\text{if} \quad \texttt{itype} = 1 \text{ or } 2, \quad Z^\text{H}BZ = I; \\
\text{if} \quad \texttt{itype} = 3, \quad Z^\text{H}B^{-1}Z = I;
\end{align*}
\]

if an eigenvector fails to converge (\texttt{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 \texttt{jfail}.

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

19: \texttt{pdz} – Integer 
\textit{Input}

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

\textit{Constraints}:

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

20: \texttt{jfail[dim]} – Integer 
\textit{Output}

\textit{Note}: the dimension, \texttt{dim}, of the array \texttt{jfail} must be at least \(\max(1, n)\).
On exit: if job = 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.

21: 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 failed to converge.

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, itype = ⟨value⟩.

Constraint: itype = 1, 2 or 3.

On entry, n = ⟨value⟩.

Constraint: n ≥ 0.

On entry, pda = ⟨value⟩.

Constraint: pda > 0.

On entry, pdb = ⟨value⟩.

Constraint: pdb > 0.

On entry, pdz = ⟨value⟩.

Constraint: pdz > 0.

NE_INT_2

On entry, pda = ⟨value⟩ and n = ⟨value⟩.

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

**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_MAT_NOT_POS_DEF**

If \( \text{fail}\_\text{errno} = \text{n} + \langle \text{value} \rangle \), for \( 1 \leq \langle \text{value} \rangle \leq \text{n} \), then the leading minor of order \( \langle \text{value} \rangle \) of \( B \) is not positive definite. The factorization of \( B \) could not be completed and no eigenvalues or eigenvectors were computed.

**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**

If \( B \) is ill-conditioned with respect to inversion, then the error bounds for the computed eigenvalues and vectors may be large, although when the diagonal elements of \( B \) differ widely in magnitude the eigenvalues and eigenvectors may be less sensitive than the condition of \( B \) would suggest. See Section 4.10 of Anderson *et al.* (1999) for details of the error bounds.

8   **Parallelism and Performance**

\texttt{nag\_zhegvx (f08spc)} is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

\texttt{nag\_zhegvx (f08spc)} 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\_dsygvx (f08sbc)}.

10   **Example**

This example finds the eigenvalues in the half-open interval \( (-3, 3] \), and corresponding eigenvectors, of the generalized Hermitian eigenproblem \( Az = \lambda Bz \), where

\[
A = \begin{pmatrix}
-7.36 & 0.77 - 0.43i & -0.64 - 0.92i & 3.01 - 6.97i \\
0.77 + 0.43i & 3.49 & 2.19 + 4.45i & 1.90 + 3.73i \\
-0.64 + 0.92i & 2.19 - 4.45i & 0.12 & 2.88 - 3.17i \\
3.01 + 6.97i & 1.90 - 3.73i & 2.88 + 3.17i & -2.54
\end{pmatrix}
\]

and
$B = \begin{pmatrix} 3.23 & 1.51 - 1.92i & 1.90 + 0.84i & 0.42 + 2.50i \\ 1.51 + 1.92i & 3.58 & -0.23 + 1.11i & -1.18 + 1.37i \\ 1.90 - 0.84i & -0.23 - 1.11i & 4.09 & 2.33 - 0.14i \\ 0.42 - 2.50i & -1.18 - 1.37i & 2.33 + 0.14i & 4.29 \end{pmatrix}. $

The example program for nag_zhegvd (f08sqc) illustrates solving a generalized Hermitian eigenproblem of the form $ABz = \lambda z$.

### 10.1 Program Text

/* nag_zhegvx (f08spc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 23, 2011.
 */

```c
#include <stdio.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <naga02.h>

int main(void)
{
    /* Scalars */
    double abstol, vl, vu;
    Integer i, il = 0, iu = 0, j, m, n, pda, pdb, pdz;
    Integer exit_status = 0;

    /* Arrays */
    Complex *a = 0, *b = 0, *z = 0;
    double *w = 0;
    Integer *index = 0;
    char nag_enum_arg[40];

    /* Nag Types */
    NagError fail;
    Nag_OrderType order;
    Nag_UploType uplo;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J-1)*pda + I-1 ]
    #define B(I, J) b[(J-1)*pdb + I-1 ]
    #define Z(I, J) z[(J-1)*pdz + I-1 ]
    order = Nag_ColMajor;
    #else
    #define A(I, J) a[(I-1)*pda+J-1 ]
    #define B(I, J) b[(I-1)*pdb + J-1 ]
    #define Z(I, J) z[(I-1)*pdz + J-1 ]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_zhegvx (f08spc) Example Program Results

    ");

    /* Skip heading in data file */
    #ifdef _WIN32
    scanf_s(%*[\n];
    #else
    scanf("%*[\n];
    #endif
    #ifdef _WIN32
    scanf("%"NAG_IFMT"%*[\n], &n);
    #else

Mark 25

f08spc.7
```c
scanf("%NAG_IFMT"%*[\n]", &n);
#endif
if (n < 0)
{
    printf("Invalid n\n");
    exit_status = 1;
    goto END;;
}
#endif
#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);
m = n;
pda = n;
pdb = n;
pdz = n;

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

/* Read the lower and upper bounds of the interval to be searched. */
#endif
#ifdef _WIN32
scanf_s("%lf%lf%*[\n]", &vl, &vu);
#else
scanf("%lf%lf%*[\n]", &vl, &vu);
#endif
/* Read the upper triangular parts of the matrices A and B */
if (uplo == Nag_Upper)
{
    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
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
            #ifdef _WIN32
                scanf_s(" ( %lf , %lf ) ", &B(i, j).re, &B(i, j).im);
            #else
                scanf(" ( %lf , %lf ) ", &B(i, j).re, &B(i, j).im);
            #endif
    else
    {
        for (i = 1; i <= n; ++i)
            for (j = 1; j <= i; ++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
```
```c
else
    scanf(" ( %lf , %lf ) ", &A(i, j).re, &A(i, j).im);
#endif
#endif _WIN32
    scanf_s("%*[\n"]);
#else
    scanf("%*[\n"]);
#endif
    for (i = 1; i <= n; ++i)
        for (j = 1; j <= i; ++j)
#ifdef _WIN32
    scanf_s(" ( %lf , %lf ) ", &B(i, j).re, &B(i, j).im);
#else
    scanf(" ( %lf , %lf ) ", &B(i, j).re, &B(i, j).im);
#endif
#endif _WIN32
    scanf_s("%*[\n"]);
#else
    scanf("%*[\n"]);
#endif
/* Use default value for the absolute error tolerance for eigenvalues. */
abstol = 0.0;
/* Solve the generalized Hermitian eigenvalue problem A*x = lambda*B*x
using nag_zhegvx (f08spc).
*/
nag_zhegvx(order, 1, Nag_DoBoth, Nag_Interval, uplo, n, a, pda,
b, pdb, vl, vu, il, iu, abstol, &m, w, z, pdz, index, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_zhegvx (f08spc).
%s
", fail.message);
    exit_status = 1;
    goto END;
}
/* Normalize the eigenvectors */
for(j=1; j<=m; j++)
    for(i=n; i>=1; i--)
        Z(i, j) = nag_complex_divide(Z(i, j), Z(1, j));
/* Print eigensolution */
print("Number of eigenvalues found =%5"NAG_IFMT"

", m);
print("Eigenvalues\n ");
for (j = 0; j < m; ++j) printf(" %7.4f%s", w[j], j%8 == 7?"\n":"");
print("\n\n");
/* Print eigenvalues using nag_gen_complx_mat_print (x04dac). */
fflush(stdout);
nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m,
z, pdz, "Selected eigenvectors", 0, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_gen_complx_mat_print (x04dac).
"
fail.message);
    exit_status = 1;
    goto END;
}
END:
NAG_FREE(a);
NAG_FREE(b);
NAG_FREE(z);
NAG_FREE(w);
NAG_FREE(index);
return exit_status;
}
10.2 Program Data

nag_zhegvx (f08spc) Example Program Data

4 : n
Nag_Upper : uplo
-3.0 3.0 : VL and VU

(-7.36, 0.00) ( 0.77, -0.43) (-0.64, -0.92) ( 3.01, -6.97)
 ( 3.49, 0.00) ( 2.19, 4.45) ( 1.90, 3.73)
 ( 0.12, 0.00) ( 2.88, -3.17)
 (-2.54, 0.00) : matrix A

( 3.23, 0.00) ( 1.51, -1.92) ( 1.90, 0.84) ( 0.42, 2.50)
 ( 3.58, 0.00) (-0.23, 1.11) (-1.18, 1.37)
 ( 4.09, 0.00) ( 2.33, -0.14)
 ( 4.29, 0.00) : matrix B

10.3 Program Results

nag_zhegvx (f08spc) Example Program Results

Number of eigenvalues found = 2

Eigenvalues
-2.9936  0.5047

Selected eigenvectors

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td></td>
<td>0.0000</td>
<td>-0.0000</td>
</tr>
<tr>
<td>2</td>
<td>0.1491</td>
<td>0.1882</td>
</tr>
<tr>
<td></td>
<td>0.0777</td>
<td>-0.7410</td>
</tr>
<tr>
<td>3</td>
<td>-1.2303</td>
<td>-0.2080</td>
</tr>
<tr>
<td></td>
<td>-0.4192</td>
<td>-0.4733</td>
</tr>
<tr>
<td>4</td>
<td>0.5811</td>
<td>0.4524</td>
</tr>
<tr>
<td></td>
<td>1.0051</td>
<td>0.9265</td>
</tr>
</tbody>
</table>