NAG Library Function Document

nag_zhbgvx (f08upc)

1 Purpose

nag_zhbgvx (f08upc) computes selected the eigenvalues and, optionally, the eigenvectors of a complex generalised Hermitian-definite banded eigenproblem, of the form

\[ Az = \lambda Bz, \]

where \( A \) and \( B \) are Hermitian and banded, and \( B \) is also positive definite. Eigenvalues and eigenvectors can be selected by specifying either all eigenvalues, a range of values or a range of indices for the desired eigenvalues.

2 Specification

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

void nag_zhbgvx (Nag_OrderType order, Nag_JobType job, Nag_RangeType range,
                Nag_UploType uplo, Integer n, Integer ka, Integer kb, Complex ab[ ],
                Integer pdab, Complex bb[ ], Integer pdbb, Complex q[ ], Integer pdq,
                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 generalised Hermitian-definite band problem

\[ Az = \lambda Bz \]

is first reduced to a standard band Hermitian problem

\[ Cx = \lambda x, \]

where \( C \) is a Hermitian band matrix, using Wilkinson’s modification to Crawford’s algorithm (see Crawford (1973) and Wilkinson (1977)). The Hermitian eigenvalue problem is then solved for the required eigenvalues and eigenvectors, and the eigenvectors are then backtransformed to the eigenvectors of the original problem.

The eigenvectors are normalised so that

\[ z^H A z = \lambda \quad \text{and} \quad z^H B z = 1. \]

4 References


Crawford C R (1973) Reduction of a band-symmetric generalised eigenvalue problem Comm. ACM 16 41-44


5 Arguments

1:  order – Nag_OrderType  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \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 = 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 = Nag_ROWMajor or Nag_COLMajor.}
    \end{array} \]

2:  job – Nag_JobType  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \text{On entry: indicates whether eigenvectors are computed.}
    
    job = Nag_EigVals & \Rightarrow & \text{Only eigenvalues are computed.}
    
    job = Nag_DoBoth & \Rightarrow & \text{Eigenvalues and eigenvectors are computed.}
    
    \text{Constraint: job = Nag_EigVals or Nag_DoBoth.}
    \end{array} \]

3:  range – Nag_RangeType  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \text{On entry: if range = Nag_AllValues, all eigenvalues will be found.}
    
    \text{If range = Nag_Interval, all eigenvalues in the half-open interval } (vl, vu) \text{ will be found.}
    
    \text{If range = Nag_Indices, the ith to jth eigenvalues will be found.}
    
    \text{Constraint: range = Nag_AllValues, Nag_Interval or Nag_Indices.}
    \end{array} \]

4:  uplo – Nag_UploType  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \text{On entry: if uplo = Nag_Upper, the upper triangles of } A \text{ and } B \text{ are stored.}
    
    \text{If uplo = Nag_Lower, the lower triangles of } A \text{ and } B \text{ are stored.}
    
    \text{Constraint: uplo = Nag_Upper or Nag_Lower.}
    \end{array} \]

5:  n – Integer  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \text{On entry: } n, \text{ the order of the matrices } A \text{ and } B.
    
    \text{Constraint: n } \geq 0.
    \end{array} \]

6:  ka – Integer  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \text{On entry: if uplo = Nag_Upper, the number of superdiagonals, } k_a, \text{ of the matrix } A.
    
    \text{If uplo = Nag_Lower, the number of subdiagonals, } k_a, \text{ of the matrix } A.
    
    \text{Constraint: ka } \geq 0.
    \end{array} \]

7:  kb – Integer  
    \[ \begin{array}{rcl}
    \text{Input} & \quad & \text{On entry: if uplo = Nag_Upper, the number of superdiagonals, } k_b, \text{ of the matrix } B.
    
    \text{If uplo = Nag_Lower, the number of subdiagonals, } k_b, \text{ of the matrix } B.
    
    \text{Constraint: ka } \geq kb \geq 0.
    \end{array} \]

8:  ab[dim] – Complex  
    \[ \begin{array}{rcl}
    \text{Input/Output} & \quad & \text{On entry: the upper or lower triangle of the } n \text{ by } n \text{ Hermitian band matrix } A.
    
    \text{Note: the dimension, dim, of the array } ab \text{ must be at least max}(1, pdab \times n).
    \end{array} \]
This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of $A_{ij}$, depends on the order and uplo arguments as follows:

- If order = Nag_ColMajor and uplo = Nag_Upper,
  \[ A_{ij} \text{ is stored in } ab[k_a + i - j (j - 1) \times \text{pdab}], \text{ for } j = 1, \ldots, n \text{ and } i = \max(1, j - k_a), \ldots, j; \]
- If order = Nag_ColMajor and uplo = Nag_Lower,
  \[ A_{ij} \text{ is stored in } ab[i - j + (j - 1) \times \text{pdab}], \text{ for } j = 1, \ldots, n \text{ and } i = j, \ldots, \min(n, j + k_a); \]
- If order = Nag_RowMajor and uplo = Nag_Upper,
  \[ A_{ij} \text{ is stored in } ab[j - i + (i - 1) \times \text{pdab}], \text{ for } i = 1, \ldots, n \text{ and } j = i, \ldots, \min(n, i + k_a); \]
- If order = Nag_RowMajor and uplo = Nag_Lower,
  \[ A_{ij} \text{ is stored in } ab[k_a + j - i + (i - 1) \times \text{pdab}], \text{ for } i = 1, \ldots, n \text{ and } j = \max(1, i - k_a), \ldots, i. \]

On exit: the contents of ab are overwritten.

9: pdab – Integer

On entry: the stride separating row or column elements (depending on the value of order) of the matrix $A$ in the array ab.

Constraint: pdab $\geq$ ka + 1.

10: bb[dim] – Complex

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

On entry: the upper or lower triangle of the n by n Hermitian positive definite band matrix B.

This is stored as a notional two-dimensional array with row elements or column elements stored contiguously. The storage of elements of $B_{ij}$, depends on the order and uplo arguments as follows:

- If order = Nag_ColMajor and uplo = Nag_Upper,
  \[ B_{ij} \text{ is stored in } bb[k_b + i - j + (j - 1) \times \text{pdbb}], \text{ for } j = 1, \ldots, n \text{ and } i = \max(1, j - k_b), \ldots, j; \]
- If order = Nag_ColMajor and uplo = Nag_Lower,
  \[ B_{ij} \text{ is stored in } bb[i - j + (j - 1) \times \text{pdbb}], \text{ for } j = 1, \ldots, n \text{ and } i = j, \ldots, \min(n, j + k_b); \]
- If order = Nag_RowMajor and uplo = Nag_Upper,
  \[ B_{ij} \text{ is stored in } bb[j - i + (i - 1) \times \text{pdbb}], \text{ for } i = 1, \ldots, n \text{ and } j = i, \ldots, \min(n, i + k_b); \]
- If order = Nag_RowMajor and uplo = Nag_Lower,
  \[ B_{ij} \text{ is stored in } bb[k_b + j - i + (i - 1) \times \text{pdbb}], \text{ for } i = 1, \ldots, n \text{ and } j = \max(1, i - k_b), \ldots, i. \]

On exit: the factor $S$ from the split Cholesky factorization $B = S^H S$, as returned by nag_zpbstf (f08utc).

11: pdbb – Integer

On entry: the stride separating row or column elements (depending on the value of order) of the matrix $B$ in the array bb.

Constraint: pdbb $\geq$ kb + 1.
12: q[dim] – Complex

Output

Note: the dimension, dim, of the array q must be at least

\[ \max(1, \text{pdq} \times n) \] when \( \text{job} = \text{Nag}_{-}\text{DoBoth} \);

1 otherwise.

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

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

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

On exit: if \( \text{job} = \text{Nag}_{-}\text{DoBoth} \), the \( n \times n \) matrix, \( Q \) used in the reduction of the standard form, i.e., \( Cx = \lambda x \), from symmetric banded to tridiagonal form.

If \( \text{job} = \text{Nag}_{-}\text{EigVals} \), \( q \) is not referenced.

13: pdq – Integer

Input

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

Constraints:

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

14: vl – double

Input

15: vu – double

Input

On entry: if \( \text{range} = \text{Nag}_{-}\text{Interval} \), the lower and upper bounds of the interval to be searched for eigenvalues.

If \( \text{range} = \text{Nag}_{-}\text{AllValues} \) or \( \text{Nag}_{-}\text{Indices} \), \( vl \) and \( vu \) are not referenced.

Constraint: if \( \text{range} = \text{Nag}_{-}\text{Interval} \), \( vl < vu \).

16: il – Integer

Input

17: iu – Integer

Input

On entry: if \( \text{range} = \text{Nag}_{-}\text{Indices} \), the indices (in ascending order) of the smallest and largest eigenvalues to be returned.

If \( \text{range} = \text{Nag}_{-}\text{AllValues} \) or \( \text{Nag}_{-}\text{Interval} \), \( il \) and \( iu \) are not referenced.

Constraints:

\[
\begin{align*}
\text{if } \text{range} = \text{Nag}_{-}\text{Indices} \text{ and } n = 0, & \quad il = 1 \text{ and } iu = 0; \\
\text{if } \text{range} = \text{Nag}_{-}\text{Indices} \text{ and } n > 0, & \quad 1 \leq il \leq iu \leq n.
\end{align*}
\]

18: 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 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 \( C \) to tridiagonal form. Eigenvalues will be computed most accurately when \( \text{abstol} \) is set to twice the underflow threshold \( 2 \times \text{nag}_{-}\text{real}_{-}\text{safe}_{-}\text{small}_{-}\text{number}( ) \), not zero. If this function returns with \( \text{fail.code} = \text{NE}_{-}\text{CONVERGENCE} \), indicating that some eigenvectors did not converge, try setting \( \text{abstol} \) to \( 2 \times \text{nag}_{-}\text{real}_{-}\text{safe}_{-}\text{small}_{-}\text{number}( ) \). See Demmel and Kahan (1990).

19: m – Integer *

Output

On exit: the total number of eigenvalues found. \( 0 \leq m \leq n \).
If range = Nag_AllValues, m = n.
If range = Nag_Indices, m = iu - il + 1.

20: \( \mathbf{w}[\mathbf{n}] \) – double  
\textit{Output}
\textit{On exit:} the eigenvalues in ascending order.

21: \( \mathbf{z}[\dim] \) – Complex  
\textit{Output}
\textit{Note:} the dimension, \( \dim \), of the array \( \mathbf{z} \) must be at least 
\text{max}(1, \mathbf{pdz} \times \mathbf{n}) \text{ when } \text{job} = \text{Nag_DoBoth};
1 otherwise.

The \((i, j)\text{th element of the matrix } \mathbf{Z} \text{ is stored in}
\begin{align*}
\mathbf{z}[(j - 1) \times \mathbf{pdz} + i - 1] \text{ when } \text{order} = \text{Nag_ColMajor};
\mathbf{z}[(i - 1) \times \mathbf{pdz} + j - 1] \text{ when } \text{order} = \text{Nag_RowMajor}.
\end{align*}

\textit{On exit:} if \text{job} = \text{Nag_DoBoth}, \( \mathbf{z} \text{ contains the matrix } \mathbf{Z} \text{ of eigenvectors, with the } i \text{th column of } \mathbf{Z} \text{ holding the eigenvector associated with } \mathbf{w}[i - 1]. \text{ The eigenvectors are normalized so that } \mathbf{Z}^H \mathbf{B} \mathbf{Z} = \mathbf{I}.

If \text{job} = \text{Nag_EigVals}, \( \mathbf{z} \) is not referenced.

22: \( \mathbf{pdz} \) – Integer  
\textit{Input}
\textit{On entry:} the stride separating row or column elements (depending on the value of \text{order}) in the array \( \mathbf{z}. \)
\textit{Constraints:}
\begin{align*}
\text{if } \text{job} = \text{Nag_DoBoth}, & \mathbf{pdz} \geq \text{max}(1, \mathbf{n}); \\
\text{otherwise } & \mathbf{pdz} \geq 1.
\end{align*}

23: \( \mathbf{jfail}[\dim] \) – Integer  
\textit{Output}
\textit{Note:} the dimension, \( \dim \), of the array \( \mathbf{jfail} \) must be at least \text{max}(1, \mathbf{n}).
\textit{On exit:} if \text{job} = \text{Nag_DoBoth}, then
\begin{align*}
\text{if } \text{fail} \text{.code} = \text{NE_NOERROR}, & \text{ the first } \mathbf{m} \text{ elements of } \mathbf{jfail} \text{ are zero}; \\
\text{if } \text{fail} \text{.code} = \text{NE_CONVERGENCE}, & \mathbf{jfail} \text{ contains the indices of the eigenvectors that failed to converge.}
\end{align*}

If \text{job} = \text{Nag_EigVals}, \( \mathbf{jfail} \) is not referenced.

24: \( \text{fail} \) – NagError *  
\textit{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.
See Section 3.2.1.2 in the Essential Introduction for further information.

NE_BAD_PARAM
On entry, argument \( \langle \text{value} \rangle \) had an illegal value.
**NE_CONVERGENCE**

The algorithm failed to converge; \(\text{value}\) eigenvectors did not converge. Their indices are stored in array \(jfail\).

**NE_ENUM_INT_2**

On entry, \(\text{job} = \langle \text{value} \rangle\), \(\text{pdq} = \langle \text{value} \rangle\) and \(n = \langle \text{value} \rangle\).
Constraint: if \(\text{job} = \text{Nag}_{\text{DoBoth}}\), \(\text{pdq} \geq \max(1,n)\); otherwise \(\text{pdq} \geq 1\).

On entry, \(\text{job} = \langle \text{value} \rangle\), \(\text{pdz} = \langle \text{value} \rangle\) and \(n = \langle \text{value} \rangle\).
Constraint: if \(\text{job} = \text{Nag}_{\text{DoBoth}}\), \(\text{pdz} \geq \max(1,n)\); otherwise \(\text{pdz} \geq 1\).

**NE_ENUM_INT_3**

On entry, \(\text{range} = \langle \text{value} \rangle\), \(\text{il} = \langle \text{value} \rangle\), \(\text{iu} = \langle \text{value} \rangle\) and \(n = \langle \text{value} \rangle\).
Constraint: if \(\text{range} = \text{Nag}_{\text{Indices}}\) and \(n = 0\), \(\text{il} = 1\) and \(\text{iu} = 0\); if \(\text{range} = \text{Nag}_{\text{Indices}}\) and \(n > 0\), \(1 \leq \text{il} \leq \text{iu} \leq n\).

**NE_ENUM_REAL_2**

On entry, \(\text{range} = \langle \text{value} \rangle\), \(\text{vl} = \langle \text{value} \rangle\) and \(\text{vu} = \langle \text{value} \rangle\).
Constraint: if \(\text{range} = \text{Nag}_{\text{Interval}}\), \(\text{vl} < \text{vu}\).

**NE_INT**

On entry, \(\text{ka} = \langle \text{value} \rangle\).
Constraint: \(\text{ka} \geq 0\).

On entry, \(n = \langle \text{value} \rangle\).
Constraint: \(n \geq 0\).

On entry, \(\text{pdab} = \langle \text{value} \rangle\).
Constraint: \(\text{pdab} > 0\).

On entry, \(\text{pdbb} = \langle \text{value} \rangle\).
Constraint: \(\text{pdbb} > 0\).

On entry, \(\text{pdq} = \langle \text{value} \rangle\).
Constraint: \(\text{pdq} > 0\).

On entry, \(\text{pdz} = \langle \text{value} \rangle\).
Constraint: \(\text{pdz} > 0\).

**NE_INT_2**

On entry, \(\text{ka} = \langle \text{value} \rangle\) and \(\text{kb} = \langle \text{value} \rangle\).
Constraint: \(\text{ka} \geq \text{kb} \geq 0\).

On entry, \(\text{pdab} = \langle \text{value} \rangle\) and \(\text{ka} = \langle \text{value} \rangle\).
Constraint: \(\text{pdab} \geq \text{ka} + 1\).

On entry, \(\text{pdbb} = \langle \text{value} \rangle\) and \(\text{kb} = \langle \text{value} \rangle\).
Constraint: \(\text{pdbb} \geq \text{kb} + 1\).

**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 fail.errnum = n + (value), for 1 ≤ (value) ≤ n, then nag_zpbstf (f08utc) returned
fail.errnum = (value): 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
nag_zhbgvx (f08upc) is threaded by NAG for parallel execution in multithreaded implementations of the
NAG Library.

nag_zhbgvx (f08upc) 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 n³ if job = Nag_DoBoth and
range = Nag_AllValues, and assuming that n ≫ k_a, is approximately proportional to n²k_a if
job = Nag_EigVals. Otherwise the number of floating-point operations depends upon the number of
eigenvectors computed.

The real analogue of this function is nag_zsbgvx (f08sbc).

10 Example
This example finds the eigenvalues in the half-open interval (0.0, 2.0], and corresponding eigenvectors,
of the generalized band Hermitian eigenproblem Az = λBz, where

A = \begin{pmatrix}
-1.13 & 1.94 - 2.10i & -1.40 + 0.25i & 0 \\
1.94 + 2.10i & -1.91 & -0.82 - 0.89i & -0.67 + 0.34i \\
-1.40 - 0.25i & -0.82 + 0.89i & -1.87 & -1.10 - 0.16i \\
0 & -0.67 - 0.34i & -1.10 + 0.16i & 0.50
\end{pmatrix}

and

B = \begin{pmatrix}
9.89 & 1.08 - 1.73i & 0 & 0 \\
1.08 + 1.73i & 1.69 & -0.04 + 0.29i & 0 \\
0 & -0.04 - 0.29i & 2.65 & -0.33 + 2.24i \\
0 & 0 & -0.33 - 2.24i & 2.17
\end{pmatrix}.
10.1 Program Text

/* nag_zhbgvx (f08upc) Example Program. */
* Copyright 2014 Numerical Algorithms Group.
* Mark 23, 2011. */

#include <stdio.h>
#include <nag.h>
#include <nagf08.h>
#include <nagx04.h>
#include <nagstdlib.h>

int main(void)
{

    /* Scalars */
    double abstol, vl, vu;
    Integer exit_status = 0, il = 1, iu = 1;
    Integer i, j, ka, kb, m, n, pdab, pdbb, pdq, pdz, zsize;

    /* Arrays */
    Complex *ab = 0, *bb = 0, *q = 0, *z = 0;
    double *w = 0;
    Integer *index = 0;
    char nag_enum_arg[40];

    /* Nag Types */
    NagError fail, failp;
    Nag_OrderType order;
    Nag_UploType uplo;
    Nag_JobType job;

    #ifdef NAG_COLUMN_MAJOR
    #define AB_UPPER(I, J) ab[(J-1)*pdab + ka + I - J]
    #define AB_LOWER(I, J) ab[(J-1)*pdab + I - J]
    #define BB_UPPER(I, J) bb[(J-1)*pdbb + kb + I - J]
    #define BB_LOWER(I, J) bb[(J-1)*pdbb + I - J]
    order = Nag_ColMajor;
    #else
    #define AB_UPPER(I, J) ab[(I-1)*pdab + J - I]
    #define AB_LOWER(I, J) ab[(I-1)*pdab + ka + J - I]
    #define BB_UPPER(I, J) bb[(I-1)*pdbb + kb + J - I]
    #define BB_LOWER(I, J) bb[(I-1)*pdbb + J - I]
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_zhbgvx (f08upc) Example Program Results\n\n");

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

    if (n < 0 || ka < kb || kb < 0)
    {
        printf("Invalid n, ka or kb\n");
        exit_status = 1;
        goto END;
    }
    #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);
#ifdef _WIN32
scanf_s(" %39s[\n]", nag_enum_arg, _countof(nag_enum_arg));
#else
scanf(" %39s[\n]", nag_enum_arg);
#endif
job = (Nag_JobType) nag_enum_name_to_value(nag_enum_arg);
if (job==Nag_EigVals) {
    zsize = 1;
pdz = 1;
} else {
    zsize = n*n;
pdz = n;
}
pdab = ka + 1;
pdbb = kb + 1;
m = n;
pdq = n;
/* Allocate memory */
if (!(ab = NAG_ALLOC((ka+1) * n, Complex)) ||
! (bb = NAG_ALLOC((kb+1) * n, Complex)) ||
! (q = NAG_ALLOC(n * n, Complex)) ||
! (z = NAG_ALLOC(zsize, 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. */
#ifdef _WIN32
scanf_s("%lf%lf[\n]", &vl, &vu);
#else
scanf("%lf%lf[\n]", &vl, &vu);
#endif
/* Read the triangular parts of the matrices A and B from data file */
if (uplo == Nag_Upper) {
    for (i = 1; i <= n; ++i)
        for (j = i; j <= MIN(i+ka, n); ++j)
#ifdef _WIN32
            scanf_s(" ( %lf , %lf )", &AB_UPPER(i, j).re, &AB_UPPER(i, j).im);
#else
            scanf(" ( %lf , %lf )", &AB_UPPER(i, j).re, &AB_UPPER(i, j).im);
#endif
#else
    scanf("%*[\n]");
#endif
#ifdef _WIN32
    scanf_s("%*[\n]");
#else
    scanf("%*[\n]");
#endif
for (i = 1; i <= n; ++i)
    for (j = i; j <= MIN(i+kb, n); ++j)
#ifdef _WIN32
        scanf_s(" ( %lf , %lf )", &BB_UPPER(i, j).re, &BB_UPPER(i, j).im);
#else
        scanf(" ( %lf , %lf )", &BB_UPPER(i, j).re, &BB_UPPER(i, j).im);
#endif
#else
    scanf("%*[\n]");
#endif
     */

Mark 25
for (i = 1; i <= n; ++i)
  for (j = MAX(1, i-ka); j <= i; ++j)
    #ifdef _WIN32
      scanf_s(" ( %lf , %lf )", &AB_LOWER(i, j).re, &AB_LOWER(i, j).im);
    #else
      scanf(" ( %lf , %lf )", &AB_LOWER(i, j).re, &AB_LOWER(i, j).im);
    #endif
  #ifdef _WIN32
    scanf_s("%*[\n"]);
  #else
    scanf("%*[\n"]);
  #endif
  for (i = 1; i <= n; ++i)
    for (j = MAX(1, i-kb); j <= i; ++j)
      #ifdef _WIN32
        scanf_s(" ( %lf , %lf )", &BB_LOWER(i, j).re, &BB_LOWER(i, j).im);
      #else
        scanf(" ( %lf , %lf )", &BB_LOWER(i, j).re, &BB_LOWER(i, j).im);
      #endif
  #ifdef _WIN32
    scanf_s("%*[\n"]);
  #else
    scanf("%*[\n"]);
  #endif
/* Use the default absolute error tolerance for eigenvalues. */
abstol = 0.;
/* Solve the generalized symmetric eigenvalue problem A*x = lambda*B*x
 * using nag_zhbgvx (f08upc).
 */
nag_zhbgvx(order, job, Nag_Interval, uplo, n, ka, kb, ab, pdab, bb, pdbb, q,
             pdq, vl, vu, il, iu, abstol, &m, w, z, pdz, index, &fail);
  if (fail.code != NE_NOERROR && fail.code != NE_CONVERGENCE)
    {
      printf("Error from nag_zhbgvx (f08upc).
%s
", fail.message);
      exit_status = 1;
      goto END;
    }
/* Print eigensolution */
printf("Number of eigenvalues found =%5"NAG_IFMT"

", m);
printf(" Eigenvalues
 ");
for (j = 0; j < m; ++j) printf(" %10.4f%s", w[j], j%6 == 5?"\n":" ");
printf("\n");
if (job==Nag_DoBoth) {
    /* nag_gen_complx_mat_print (x04dac): Print Matrix of eigenvectors Z. */
    printf("\n");
    fflush(stdout);
    INIT_FAIL(failp);
    nag_gen_complx_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m,
                z, pdz, "Eigenvectors", 0, &failp);
    if (failp.code != NE_NOERROR)
      {
        printf("Error from nag_gen_complx_mat_print (x04dac).
%s
", failp.message);
        exit_status = 1;
      }
    if (fail.code == NE_CONVERGENCE)
      {
        printf("eigenvectors failed to converge
");
        printf("Indices of eigenvectors that did not converge
 ");
        for (j = 0; j < m; ++j)
          printf("%8"NAG_IFMT"
", index[j], j%6 == 5?"\n":"");
        printf("\n");
      }
END:
NAG_FREE(ab);
NAG_FREE(bb);
NAG_FREE(q);
NAG_FREE(z);
NAG_FREE(w);
NAG_FREE(index);

return exit_status;
}

10.2 Program Data

nag_zhbgvx (f08upc) Example Program Data

<table>
<thead>
<tr>
<th>n</th>
<th>ka</th>
<th>kb</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

: n, ka and kb

Nag_Upper : uplo
Nag_DoBoth : job

<table>
<thead>
<tr>
<th>vl</th>
<th>vu</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>2.0</td>
</tr>
</tbody>
</table>

: vl and vu

(-1.13, 0.00) (1.94, -2.10) (-1.40, 0.25) (-0.82, -0.89) (-0.67, 0.34) (-1.10, -0.16) (0.50, 0.00)

: matrix A

(9.89, 0.00) (1.08, -1.73) (1.69, 0.00) (-0.04, 0.29) (2.65, 0.00) (-0.33, 2.24) (2.17, 0.00)

: matrix B

10.3 Program Results

nag_zhbgvx (f08upc) Example Program Results

Number of eigenvalues found = 2

Eigenvalues

| 0.1603 | 1.7712 |

Eigenvectors

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1908</td>
<td>0.0494</td>
</tr>
<tr>
<td>0.0137</td>
<td>-0.0045</td>
</tr>
<tr>
<td>0.1413</td>
<td>0.2505</td>
</tr>
<tr>
<td>0.1012</td>
<td>0.4427</td>
</tr>
<tr>
<td>-0.0437</td>
<td>-0.9705</td>
</tr>
<tr>
<td>-0.0905</td>
<td>0.0679</td>
</tr>
<tr>
<td>-0.2135</td>
<td>0.0606</td>
</tr>
<tr>
<td>0.2880</td>
<td>-1.3227</td>
</tr>
</tbody>
</table>