NAG Library Function Document

nag_dsygvx (f08sbc)

1 Purpose

nag_dsygvx (f08sbc) computes selected eigenvalues and, optionally, eigenvectors of a real generalized symmetric-definite eigenproblem, of the form

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

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

3 Description

nag_dsygvx (f08sbc) first performs a Cholesky factorization of the matrix \( B \) as \( B = U^T U \), when \( \text{uplo} = \text{Nag Upper} \) or \( B = LL^T \), 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

\[ Z^T Az = \Lambda \quad \text{and} \quad Z^T Bz = I, \]

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

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

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

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

4 References


5 Arguments

1: **order** – Nag_OrderType \hspace{1cm} \textit{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 \( \text{order} = \text{Nag} \_\text{RowMajor} \). See Section 3.2.1.3 in the Essential Introduction for a more detailed explanation of the use of this argument.

\textit{Constraint:} \( \text{order} = \text{Nag} \_\text{RowMajor} \) or \( \text{Nag} \_\text{ColMajor} \).

2: **itype** – Integer \hspace{1cm} \textit{Input}

*On entry:* specifies the problem type to be solved.

\( \text{itype} = 1 \)

\[ Az = \lambda Bz. \]

\( \text{itype} = 2 \)

\[ ABz = \lambda z. \]

\( \text{itype} = 3 \)

\[ BAz = \lambda z. \]

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

3: **job** – Nag_JobType \hspace{1cm} \textit{Input}

*On entry:* indicates whether eigenvectors are computed.

\( \text{job} = \text{Nag} \_\text{EigVals} \)

Only eigenvalues are computed.

\( \text{job} = \text{Nag} \_\text{DoBoth} \)

Eigenvalues and eigenvectors are computed.

\textit{Constraint:} \( \text{job} = \text{Nag} \_\text{EigVals} \) or \( \text{Nag} \_\text{DoBoth} \).

4: **range** – Nag_RangeType \hspace{1cm} \textit{Input}

*On entry:* if \( \text{range} = \text{Nag} \_\text{AllValues} \), all eigenvalues will be found.

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

If \( \text{range} = \text{Nag} \_\text{Indices} \), the \( il \)th to \( iu \)th eigenvalues will be found.

\textit{Constraint:} \( \text{range} = \text{Nag} \_\text{AllValues} \), \( \text{Nag} \_\text{Interval} \) or \( \text{Nag} \_\text{Indices} \).

5: **uplo** – Nag_UploType \hspace{1cm} \textit{Input}

*On entry:* if \( \text{uplo} = \text{Nag} \_\text{Upper} \), the upper triangles of \( A \) and \( B \) are stored.

If \( \text{uplo} = \text{Nag} \_\text{Lower} \), the lower triangles of \( A \) and \( B \) are stored.

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

6: **n** – Integer \hspace{1cm} \textit{Input}

*On entry:* \( n \), the order of the matrices \( A \) and \( B \).

\textit{Constraint:} \( n \geq 0 \).

7: **a[dim]** – double \hspace{1cm} \textit{Input/Output}

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

*On entry:* the \( n \) by \( n \) symmetric matrix \( A \).

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

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

If \( \text{uplo} = \text{Nag} \_\text{Lower} \), 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 \( \text{uplo} = \text{Nag} \_\text{Lower} \)) or the upper triangle (if \( \text{uplo} = \text{Nag} \_\text{Upper} \)) 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: \( b[\text{dim}] \) – double

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 \( n \) symmetric matrix \( B \).

If \( \text{order} = \text{Nag} \_\text{ColMajor} \), \( B_{ij} \) is stored in \( b[(j-1) \times \text{pdb} + i-1] \).

If \( \text{order} = \text{Nag} \_\text{RowMajor} \), \( B_{ij} \) is stored in \( b[(i-1) \times \text{pdb} + j-1] \).

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

If \( \text{uplo} = \text{Nag} \_\text{Lower} \), 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 \( B = U^T U \) or \( B = LL^T \).

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

12: \( \text{vu} \) – 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 \( \text{Nag} \_\text{Indices} \), \( \text{vl} \) and \( \text{vu} \) are not referenced.

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

13: \( \text{il} \) – Integer

Input

14: \( \text{iu} \) – 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 \( \text{Nag} \_\text{Interval} \), \( \text{il} \) and \( \text{iu} \) are not referenced.

Constraints:

- 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 \( n > 0 \), \( 1 \leq \text{il} \leq \text{iu} \leq n \).
15: 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

\[
abstol + \epsilon \max(|a|, |b|),
\]

where \(\epsilon\) is the machine precision. If abstol is less than or equal to zero, then 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 abstol is set to twice the underflow threshold \(2 \times \text{nag\_real\_safe\_small\_number}()\), not zero. If this function returns with fail.code = NE_CONVERGENCE, indicating that some eigenvectors did not converge, try setting abstol to \(2 \times \text{nag\_real\_safe\_small\_number}()\). See Demmel and Kahan (1990).

16: 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, \(m = iu – il + 1\).

17: w[n] – double

Output

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

18: z[dim] – double

Output

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

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

\[1\] otherwise.

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

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

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

On exit: if job = Nag_DoBoth, then

if fail.code = 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 \(w[i - 1]\). The eigenvectors are normalized as follows:

\[
\text{if itype = 1 or 2, } Z^T B Z = I;
\]

\[
\text{if itype = 3, } Z^T B^{-1} Z = I;
\]

if an eigenvector fails to converge (fail.code = NE_CONVERGENCE), then that column of \(Z\) contains the latest approximation to the eigenvector, and the index of the eigenvector is returned in jfail.

If job = Nag_EigVals, z is not referenced.

19: pdz – Integer

Input

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

Constraints:

\[
\text{if job = Nag_DoBoth, pdz} \geq \max(1, n);
\]

otherwise pdz \(\geq 1\).

20: jfail[dim] – Integer

Output

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

- if \( \text{fail\_code} = \text{NE\_NOERROR} \), the first \( m \) elements of \( \text{jfail} \) are zero;
- if \( \text{fail\_code} = \text{NE\_CONVERGENCE} \), \( \text{jfail} \) contains the indices of the eigenvectors that failed to converge.

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

21: \( \text{fail} - \text{NagError}^* \)

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 \( \langle \text{value} \rangle \) had an illegal value.

**NE\_CONVERGENCE**

The algorithm failed to converge; \( \langle \text{value} \rangle \) eigenvectors failed to converge.

**NE\_ENUM\_INT\_2**

On entry, \( \text{job} = \langle \text{value} \rangle \), \( \text{pdz} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

Constraint: if \( \text{job} = \text{Nag\_DoBoth} \), \( \text{pdz} \geq \max(1, n) \);
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 \( \text{range} = \text{Nag\_Indices} \) and \( n = 0 \), \( \text{il} = 1 \) and \( \text{iu} = 0 \); if \( \text{range} = \text{Nag\_Indices} \) and \( n > 0 \), \( 1 \leq \text{il} \leq \text{iu} \leq n \).

**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 \( \text{range} = \text{Nag\_Interval} \), \( \text{vl} < \text{vu} \).

**NE\_INT**

On entry, \( \text{itype} = \langle \text{value} \rangle \).

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

On entry, \( n = \langle \text{value} \rangle \).

Constraint: \( n \geq 0 \).

On entry, \( \text{pda} = \langle \text{value} \rangle \).

Constraint: \( \text{pda} > 0 \).

On entry, \( \text{pdb} = \langle \text{value} \rangle \).

Constraint: \( \text{pdb} > 0 \).

On entry, \( \text{pdz} = \langle \text{value} \rangle \).

Constraint: \( \text{pdz} > 0 \).

**NE\_INT\_2**

On entry, \( \text{pda} = \langle \text{value} \rangle \) and \( n = \langle \text{value} \rangle \).

Constraint: \( \text{pda} \geq \max(1, n) \).
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 \( \text{fail.errnum} = \text{n} + \langle \text{value} \rangle \), 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
\text{nag_dsygvx} (f08sbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.
\text{nag_dsygvx} (f08sbc) 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 complex analogue of this function is \text{nag_zhegvx} (f08spc).

### 10 Example
This example finds the eigenvalues in the half-open interval \((-1.0, 1.0]\), and corresponding eigenvectors, of the generalized symmetric eigenproblem \( Az = \lambda Bz \), where

\[
A = \begin{pmatrix}
0.24 & 0.39 & 0.42 & -0.16 \\
0.39 & -0.11 & 0.79 & 0.63 \\
0.42 & 0.79 & -0.25 & 0.48 \\
-0.16 & 0.63 & 0.48 & -0.03
\end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix}
4.16 & -3.12 & 0.56 & -0.10 \\
-3.12 & 5.03 & -0.83 & 1.09 \\
0.56 & -0.83 & 0.76 & 0.34 \\
-0.10 & 1.09 & 0.34 & 1.18
\end{pmatrix}.
\]

The example program for \text{nag_dsygvd} (f08scc) illustrates solving a generalized symmetric eigenproblem of the form \( ABz = \lambda z \).
10.1 Program Text

/* nag_dsygvx (f08sbc) Example Program.
 * Copyright 2014 Numerical Algorithms Group.
 * Mark 23, 2011.
 */

#include <stdio.h>
#include <nag.h>
#include <nagx04.h>
#include <nag_stdlib.h>
#include <nagf08.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 */
    double *a = 0, *b = 0, *w = 0, *z = 0;
    Integer *index = 0;
    char nag_enum_arg[40];
    /* Nag Types */
    NagError fail;
    Nag_OrderType order;
    Nag_UploType uplo;

#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 ]
    #ifdef NAG_COLUMN_MAJOR
    order = Nag_ColMajor;
    #else
    order = Nag_RowMajor;
    #endif

    INIT_FAIL(fail);

    printf("nag_dsygvx (f08sbc) 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
    if (n < 0)
    {
        printf("Invalid n\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

END: // Terminal marker

Mark 25

f08sbc.7
/* 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, double)) ||
! (b = NAG_ALLOC(n * n, double)) ||
! (w = NAG_ALLOC(n, double)) ||
! (z = NAG_ALLOC(n * m, 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. */
if (uplo == Nag_Upper)
    {
        for (i = 1; i <= n; ++i)
            {
                #ifdef _WIN32
                    for (j = i; j <= n; ++j) scanf_s("%lf", &A(i, j));
                #else
                    for (j = i; j <= n; ++j) scanf("%lf", &A(i, j));
                #endif
                #ifdef _WIN32
                    scanf_s("%*[\n]");
                #else
                    scanf("%*[\n]");
                #endif
            }
    }
else
    {
        for (i = 1; i <= n; ++i)
            {
                #ifdef _WIN32
                    for (j = 1; j <= i; ++j) scanf_s("%lf", &A(i, j));
                #else
                    for (j = 1; j <= i; ++j) scanf("%lf", &A(i, j));
                #endif
                #ifdef _WIN32
                    scanf_s("%*[\n]");
                #else
                    scanf("%*[\n]");
                #endif
            }
    }
#endif

f08sbc.8 Mark 25
#else
    scanf("%*[\n ] ");
#endif

/* Use default value for the absolute error tolerance for eigenvalues. */
abstol = 0.0;

/* Solve the generalized symmetric eigenvalue problem A*x = lambda*B*x */
/* using nag_dsygvx (f08sbc). */
nag_dsygvx(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_dsygvx (f08sbc).\n%s\n", 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) = Z(i, j) / Z(1,j);

/* 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%8 == 7?"\n":"");
printf("\n\n");
/* Print eigenvalues using nag_gen_real_mat_print (x04cac). */
fflush(stdout);
nag_gen_real_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_real_mat_print (x04cac).\n%s\n", fail.message);
        exit_status = 1;
        goto END;
    }

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

10.2 Program Data

nag_dsygvx (f08sbc) Example Program Data

4
Nag_Upper
-1.0  1.0
0.24  0.39  0.42  -0.16
-0.11  0.79  0.63
-0.25  0.48
-0.03 : matrix A
4.16 -3.12  0.56  -0.10
  5.03 -0.83  1.09
  0.76  0.34
  1.18 : matrix B
10.3 Program Results

nag_dsygvx (f08sbc) Example Program Results

Number of eigenvalues found = 2

Eigenvalues

-0.4548  0.1001

Selected eigenvectors

1 2

1  1.0000  1.0000
2  1.7303  0.0830
3 -1.1354 -0.1129
4 -2.0169 -1.0611