NAG Library Function Document

nag_dsyevx (f08fbc)

1 Purpose

nag_dsyevx (f08fbc) computes selected eigenvalues and, optionally, eigenvectors of a real $n$ by $n$ symmetric matrix $A$. Eigenvalues and eigenvectors can be selected by specifying either a range of values or a range of indices for the desired eigenvalues.

2 Specification

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

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

3 Description

The symmetric matrix $A$ is first reduced to tridiagonal form, using orthogonal 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

   **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 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

   **Input**

   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.
range – Nag_RangeType

On entry: if range = Nag_AllValues, all eigenvalues will be found.
If range = Nag_Interval, all eigenvalues in the half-open interval \((v_l, v_u]\) will be found.
If range = Nag_Indices, the \(i_l\)th to \(i_u\)th eigenvalues will be found.
Constraint: range = Nag_AllValues, Nag_Interval or Nag_Indices.

uplo – Nag_UploType

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.

n – Integer

On entry: \(n\), the order of the matrix \(A\).
Constraint: \(n \geq 0\).

a[\(dim\)] – double

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 order = Nag_ColMajor, \(A_{ij}\) is stored in \(a[(j - 1) \times pda + i - 1]\).
If order = Nag_RowMajor, \(A_{ij}\) is stored in \(a[(i - 1) \times pda + j - 1]\).
If uplo = Nag_Upper, the upper triangular part of \(A\) must be stored and the elements of the array below the diagonal are not referenced.
If uplo = 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 uplo = Nag_Lower) or the upper triangle (if uplo = Nag_Upper) of a, including the diagonal, is overwritten.

pda – Integer

On entry: the stride separating row or column elements (depending on the value of order) in the array a.
Constraint: \(pda \geq \max(1, n)\).

vl – double

vu – double

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, \(vl < vu\).

il – Integer

iu – Integer

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 ≤ il ≤ iu ≤ n.

12: 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 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 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 fail.code = NE_CONVERGENCE, indicating that some eigenvectors did not converge, try setting abstol to \(2 \times \text{nag}_\text{real}_\text{safe}_\text{small}_\text{number}\). See Demmel and Kahan (1990).

13: m – Integer *
   Output
   On exit: the total number of eigenvalues found. 0 ≤ m ≤ n.
   If range = Nag_AllValues, m = n.
   If range = Nag_Indices, m = iu – il + 1.

14: w[dim] – double
   Output
   Note: the dimension, dim, of the array w must be at least max(1, n).
   On exit: the first m elements contain the selected eigenvalues in ascending order.

15: z[dim] – double
   Output
   Note: the dimension, dim, of the array z must be at least
   \[
   \max(1, \text{pdz} \times n) \quad \text{when job = Nag_DoBoth};
   1 \quad \text{otherwise}.
   \]
   The \(i\)th element of the \(j\)th vector \(Z\) 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 \(w[i - 1]\);
   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.

16: pdz – Integer
   Input
   On entry: the stride used in the array z.
   Constraints:
   if job = Nag_DoBoth, pdz ≥ max(1, n);
   otherwise pdz ≥ 1.
Note: the dimension, \( \text{dim} \), 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 \( jfail \) are zero;
- if \( \text{fail.code} = \text{NE_CONVERGENCE} \), \( jfail \) contains the indices of the eigenvectors that failed to converge.

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

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 did not converge. Their indices are stored in array \( jfail \).

**NE_ENUM_INT_2**
On entry, \( \text{job} = \langle \text{value} \rangle \), \( \text{pdz} = \langle \text{value} \rangle \) and \( \text{n} = \langle \text{value} \rangle \).
Constraint: if \( \text{job} = \text{Nag_DoBoth} \), \( \text{pdz} \geq \max(1, \text{n}) \); 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 \( \text{n} = \langle \text{value} \rangle \).
Constraint: if \( \text{range} = \text{Nag_Indices} \) and \( \text{n} = 0 \), \( \text{il} = 1 \) and \( \text{iu} = 0 \);
if \( \text{range} = \text{Nag_Indices} \) and \( \text{n} > 0 \), \( 1 \leq \text{il} \leq \text{iu} \leq \text{n} \).

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

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

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

**NE_INT_2**
On entry, \( \text{pda} = \langle \text{value} \rangle \) and \( \text{n} = \langle \text{value} \rangle \).
Constraint: \( \text{pda} \geq \max(1, \text{n}) \).
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_dsyevx (f08fbc) is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

nag_dsyevx (f08fbc) 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 nag_zheevx (f08fpc).

10 Example
This example finds the eigenvalues in the half-open interval \((-1, 1]\), and the corresponding eigenvectors, of the symmetric matrix
\[
A = \begin{pmatrix}
1 & 2 & 3 & 4 \\
2 & 2 & 3 & 4 \\
3 & 3 & 3 & 4 \\
4 & 4 & 4 & 4
\end{pmatrix}.
\]

10.1 Program Text
/* nag_dsyevx (f08fbc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * *
 * Mark 23, 2011.
 */
#include <stdio.h>
#include <nag.h>
#include <nag_stdlib.h>
#include <nagf08.h>
#include <nagx04.h>
```c
int main(void)
{
    /* Scalars */
    double abstol, vl, vu;
    Integer i, il = 0, iu = 0, j, m, n, pda, pdz;
    Integer exit_status = 0;
    /* Arrays */
    char nag_enum_arg[40];
    double *a = 0, *w = 0, *z = 0;
    Integer *index = 0;
    /* Nag Types */
    Nag_OrderType order;
    Nag_RangeType range;
    Nag_UploType uplo;
    Nag_JobType job;
    NagError fail, fail_print;

    #ifdef NAG_COLUMN_MAJOR
    #define A(I, J) a[(J - 1) * pda + 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 Z(I, J) z[(I - 1) * pdz + J - 1]
    order = Nag_RowMajor;
    #endif
    INIT_FAIL(fail);
    printf("nag_dsyevx (f08fbc) 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, range and job */
    #ifdef _WIN32
    scanf_s("%39s%\n", nag_enum_arg, _countof(nag_enum_arg));
    #else
    scanf("%39s%\n", nag_enum_arg);
    #endif
    range = (Nag_RangeType) nag_enum_name_to_value(nag_enum_arg);
    /* 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

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

pda = n;
pdz = n;

/* Read the lower and upper bounds of the interval to be searched, */
/* and read the upper 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
for (i = 1; i <= n; ++i)
  for (j = i; j <= n; ++j)
#ifdef _WIN32
    scanf_s("%lf", &A(i, j));
#else
    scanf("%lf", &A(i, j));
#endif
#ifdef _WIN32
  scanf_s("%*[\n]" );
#else
  scanf("%*[\n]" );
#endif

/* Set the absolute error tolerance for eigenvalues. With abstol */
/* set to zero, the default value is used instead. */
abstol = 0.0;

/* nag_dsyevx (f08fbc). */
/* Solve the symmetric eigenvalue problem. */
/*
   nag_dsyevx(order, job, range, uplo, n, a, pda, vl, vu, il, iu, abstol, &m, 
   w, z, pdz, index, &fail);
if (fail.code != NE_NOERROR && fail.code != NE_CONVERGENCE)
{ printf("Error from nag_dsyevx (f08fbc).\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 solution */
printf("Number of eigenvalues found =%5"NAG_IFMT"\n", m);
printf("\nEigenvalues\n");
for (j = 0; j < m; ++j)
  printf("%8.4f%s", w[j], (j+1)%8 == 0?"\n": "");
printf("\n");

/* nag_gen_real_mat_print (x04cac). */
/* Print selected eigenvectors. */
INIT_FAIL(fail_print);
fflush(stdout);
nag_gen_real_mat_print(order, Nag_GeneralMatrix, Nag_NonUnitDiag, n, m, z,
pdz, "Selected eigenvectors", 0, &fail_print);
if (fail_print.code != NE_NOERROR)
{    
    printf("Error from nag_gen_real_mat_print (x04cac).\n%s\n", \
        fail_print.message); 
    exit_status = 1; 
    goto END;  
}  
if (fail.code == NE_CONVERGENCE)  
{    
    printf("eigenvectors failed to converge\n");    
    printf("Indices of eigenvectors that did not converge\n");    
    for (j = 0; j < m; ++j)    
        printf("%8"NAG_IFMT%s", index[j], (j+1)%8 == 0?\n": " );    
}  
END:  
NAG_FREE(a);  
NAG_FREE(w);  
NAG_FREE(z);  
NAG_FREE(index);  
return exit_status;  
}  
#undef A  
#undef Z

10.2 Program Data
nag_dsyevx (f08fbc) Example Program Data  
4 :Value of n  
Nag_Upper :Value of uplo  
Nag_Interval :Value of range  
Nag_DoBoth :Value of job  
-1.0 1.0 :Values of vl and vu  
1.0 2.0 3.0 4.0  
2.0 3.0 4.0  
3.0 4.0  
4.0 :End of matrix A

10.3 Program Results
nag_dsyevx (f08fbc) Example Program Results
Number of eigenvalues found = 2

Eigenvalues  
n-0.5146  -0.2943

Selected eigenvectors  
1  2  
1 1.0000  1.0000  
2 -0.9431  -2.3976  
3 -1.0537  2.3508  
4 0.8831  -0.8879