nag_real_symm_general_eigenvalues (f02adc)

1. Purpose

nag_real_symm_general_eigenvalues (f02adc) calculates all the eigenvalues of $Ax = \lambda Bx$, where A is a real symmetric matrix and B is a real symmetric positive-definite matrix.

2. Specification

```
#include <nag.h>
#include <nagf02.h>

void nag_real_symm_general_eigenvalues(Integer n, double a[],
     Integer tda, double b[], Integer tdb, double r[], NagError *fail)
```

3. Description

The problem is reduced to the standard symmetric eigenproblem using Cholesky's method to decompose B into triangular matrices, $B = LL^T$, where L is lower triangular. Then $Ax = \lambda Bx$ implies $(L^{-1}AL^{-T})(L^T x) = \lambda(L^T x)$; hence the eigenvalues of $Ax = \lambda Bx$ are those of $Py = \lambda y$ where P is the symmetric matrix $L^{-1}AL^{-T}$. Householder's method is used to tridiagonalise the matrix P and the eigenvalues are then found using the QL algorithm.

4. Parameters

n

Input: n, the order of the matrices A and B.
Constraint: **n** ≥ 1.

a[n][tda]

Input: the upper triangle of the n by n symmetric matrix A. The elements of the array below the diagonal need not be set.
Output: the lower triangle of the array is overwritten. The rest of the array is unchanged.

tda

Input: the second dimension of the array **a** as declared in the function from which nag_real_symm_general_eigenvalues is called.
Constraint: **tda** $\geq$ **n**.

b[n][tdb]

Input: the upper triangle of the n by n symmetric positive-definite matrix B. The elements of the array below the diagonal need not be set.
Output: the elements below the diagonal are overwritten. The rest of the array is unchanged.

tdb

Input: the second dimension of the array **b** as declared in the function from which nag_real_symm_general_eigenvalues is called.
Constraint: **tdb** $\geq$ **n**.

r[n]

Output: the eigenvalues in ascending order.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

5. Error Indications and Warnings

NE_NOT_POS_DEF

The matrix B is not positive-definite, possibly due to rounding errors.

NE_TOO_MANY_ITERATIONS

More than ⟨*value*⟩ iterations are required to isolate all the eigenvalues.

NE_INT_ARG_LT
On entry, **n** must not be less than 1: **n** = ⟨*value*⟩.

NE_2_INT_ARG_LT
On entry, **tda** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These parameters must satisfy **tda** ≥ **n**.
On entry, **tdb** = ⟨*value*⟩ while **n** = ⟨*value*⟩. These parameters must satisfy **tdb** ≥ **n**.

NE_ALLOC_FAIL
Memory allocation failed.

6. Further Comments

The time taken by the function is approximately proportional to n^3.

6.1. Accuracy

In general this function is very accurate. However, if B is ill-conditioned with respect to inversion, the eigenvalues could be inaccurately determined. For a detailed error analysis see Wilkinson and Reinsch (1971) pp 310, 222 and 235.

6.2. References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation (Vol II, Linear Algebra)* Springer-Verlag pp 303–314, 212–226 and 227–240.

7. See Also

None.

8. Example

To calculate all the eigenvalues of the general symmetric eigenproblem $Ax = \lambda Bx$ where A is the symmetric matrix

$$\begin{pmatrix} 0.5 & 1.5 & 6.6 & 4.8 \\ 1.5 & 6.5 & 16.2 & 8.6 \\ 6.6 & 16.2 & 37.6 & 9.8 \\ 4.8 & 8.6 & 9.8 & -17.1 \end{pmatrix}$$

and B is the symmetric positive-definite matrix

$$\begin{pmatrix} 1 & 3 & 4 & 1 \\ 3 & 13 & 16 & 11 \\ 4 & 16 & 24 & 18 \\ 1 & 11 & 18 & 27 \end{pmatrix}.$$

8.1. Program Text

```
/* nag_real_symm_general_eigenvalues(f02adc) Example Program
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <nagf02.h>

#define NMAX 8
#define TDA NMAX
#define TDB NMAX

main()
{
```

```
  Integer  i, j, n;
  double a[NMAX][TDA], b[NMAX][TDB], r[NMAX];

  Vprintf("f02adc Example Program Results\n");
  /* Skip heading in data file */
  Vscanf("%*[^\n]");
  Vscanf("%ld",&n);
  if (n<1 || n>NMAX)
    {
      Vfprintf(stderr, "N is out of range: N = %5ld\n", n);
      exit(EXIT_FAILURE);
    }
  for (i=0; i<n; i++)
    {
      for (j=0; j<n; j++)
        Vscanf("%lf",&a[i][j]);
      for (j=0; j<n; j++)
        Vscanf("%lf",&b[i][j]);
    }
  f02adc(n, (double *)a, (Integer)TDA, (double *)b, (Integer)TDB, r,
         NAGERR_DEFAULT);
  Vprintf("Eigenvalues\n");
  for (i=0; i<n; i++)
    Vprintf("%9.4f%s",r[i],(i%8==7 || i==n-1) ? "\n" : " ");
  exit(EXIT_SUCCESS);
}
```

8.2. Program Data

```
f02adc Example Program Data
  4
   0.5   1.5   6.6   4.8     1.0   3.0   4.0   1.0
   1.5   6.5  16.2   8.6     3.0  13.0  16.0  11.0
   6.6  16.2  37.6   9.8     4.0  16.0  24.0  18.0
   4.8   8.6   9.8 -17.1     1.0  11.0  18.0  27.0
```

8.3. Program Results

```
f02adc Example Program Results
Eigenvalues
  -3.0000   -1.0000    2.0000    4.0000
```