NAG Library Function Document

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, $B = LL^{\mathrm{T}}$, where L is lower triangular. Then $Ax = \lambda Bx$ implies $\left(L^{-1}AL^{-T}\right)(L^{\mathrm{T}}x) = \lambda(L^{\mathrm{T}}x)$; 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 References

Wilkinson J H and Reinsch C (1971) *Handbook for Automatic Computation II, Linear Algebra* Springer–Verlag

5 Arguments

1: **n** – Integer *Input*

On entry: n, the order of the matrices A and B.

Constraint: **n** ≥ 1.

2: **a**[**n** $\times$ **tda**] – double *Input/Output*

Note: the (i,j)th element of the matrix A is stored in **a**$[(i-1) \times$ **tda** $+ j - 1]$.

On entry: the upper triangle of the n by n symmetric matrix A. The elements of the array below the diagonal need not be set.

On exit: the lower triangle of the array is overwritten. The rest of the array is unchanged.

3: **tda** – Integer *Input*

On entry: the stride separating matrix column elements in the array **a**.

Constraint: **tda** $\geq$ **n**.

4: **b**[**n** $\times$ **tdb**] – double *Input/Output*

Note: the (i,j)th element of the matrix B is stored in **b**$[(i-1) \times$ **tdb** $+ j - 1]$.

On entry: 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.

On exit: the elements below the diagonal are overwritten. The rest of the array is unchanged.

5: **tdb** – Integer *Input*

On entry: the stride separating matrix column elements in the array **b**.

Constraint: **tdb** $\geq$ **n**.

6: **r**[**n**] – double *Output*

On exit: the eigenvalues in ascending order.

7: **fail** – NagError * *Input/Output*

The NAG error argument (see Section 3.6 in the Essential Introduction).

6 Error Indicators and Warnings

NE_2_INT_ARG_LT

On entry, **tda** $= \langle value \rangle$ while **n** $= \langle value \rangle$. These arguments must satisfy **tda** $\geq$ **n**.

On entry, **tdb** $= \langle value \rangle$ while **n** $= \langle value \rangle$. These arguments must satisfy **tdb** $\geq$ **n**.

NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_INT_ARG_LT

On entry, **n** $= \langle value \rangle$.
Constraint: **n** ≥ 1.

NE_NOT_POS_DEF

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

NE_TOO_MANY_ITERATIONS

More than $\langle value \rangle$ iterations are required to isolate all the eigenvalues.

7 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 pages 310, 222 and 235 Wilkinson and Reinsch (1971).

8 Parallelism and Performance

Not applicable.

9 Further Comments

The time taken by nag_real_symm_general_eigenvalues (f02adc) is approximately proportional to n^3.

10 Example

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

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

and B is the symmetric positive definite matrix

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

10.1 Program Text

```
/* nag_real_symm_general_eigenvalues (f02adc) Example Program.
 *
 * Copyright 1990 Numerical Algorithms Group.
 *
 * Mark 1, 1990.
 * Mark 8 revised, 2004.
 */

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

#define A(I, J) a[(I) *tda + J]
#define B(I, J) b[(I) *tdb + J]
int main(void)
{

  Integer  exit_status = 0, i, j, n, tda, tdb;
  NagError fail;
  double   *a = 0, *b = 0, *r = 0;

  INIT_FAIL(fail);

  printf("nag_real_symm_general_eigenvalues (f02adc) Example Program"
          " Results\n");
  /* Skip heading in data file */
  scanf("%*[^\n]");
  scanf("%ld", &n);
  if (n >= 1)
    {
      if (!(a = NAG_ALLOC(n*n, double)) ||
          !(b = NAG_ALLOC(n*n, double)) ||
          !(r = NAG_ALLOC(n, double)))
        {
          printf("Allocation failure\n");
          exit_status = -1;
          goto END;
        }
      tda = n;
      tdb = n;
    }
  for (i = 0; i < n; i++)
    {
      for (j = 0; j < n; j++)
        scanf("%lf", &A(i, j));
      for (j = 0; j < n; j++)
        scanf("%lf", &B(i, j));
    }
  /* nag_real_symm_general_eigenvalues (f02adc).
```

```
   * All eigenvalues of generalized real symmetric-definite
   * eigenproblem
   */
  nag_real_symm_general_eigenvalues(n, a, tda, b, tdb, r, &fail);
  if (fail.code != NE_NOERROR)
    {
      printf(
              "Error from nag_real_symm_general_eigenvalues (f02adc).\n%s\n",
              fail.message);
      exit_status = 1;
      goto END;
    }

  printf("Eigenvalues\n");
  for (i = 0; i < n; i++)
    printf("%9.4f%s", r[i], (i%8 == 7 || i == n-1)?"\n":" ");
 END:
  NAG_FREE(a);
  NAG_FREE(b);
  NAG_FREE(r);
  return exit_status;
}
```

10.2 Program Data

```
nag_real_symm_general_eigenvalues (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
```

10.3 Program Results

```
nag_real_symm_general_eigenvalues (f02adc) Example Program Results
Eigenvalues
  -3.0000   -1.0000    2.0000    4.0000
```