nag_real_eigensystem_sel (f02ecc)

1. Purpose

nag_real_eigensystem_sel (f02ecc) computes selected eigenvalues and eigenvectors of a real general matrix.

2. Specification

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

void nag_real_eigensystem_sel(Nag_Select_Eigenvalues crit, Integer n,
         double a[], Integer tda, double wl, double wu, Integer mest,
         Integer *m, Complex w[], Complex v[], Integer tdv, NagError *fail)
```

3. Description

This routine computes selected eigenvalues and the corresponding right eigenvectors of a real general matrix A:

$$Ax_i = \lambda_i x_i.$$

Eigenvalues λ_i may be selected either by *modulus*, satisfying:

$$w_l \leq |\lambda_i| \leq w_u,$$

or by *real part*, satisfying:

$$w_l \leq \mathrm{Re}(\lambda_i) \leq w_u.$$

Note that even though A is real, λ_i and x_i may be complex. If x_i is an eigenvector corresponding to a complex eigenvalue λ_i, then the complex conjugate vector $\bar{x}_i$ is the eigenvector corresponding to the complex conjugate eigenvalue $\bar{\lambda}_i$. The eigenvalues in a complex conjugate pair λ_i and $\bar{\lambda}_i$ are either both selected or both not selected.

4. Parameters

crit

Input: indicates the criterion for selecting eigenvalues:

if **crit** = **Nag_Select_Modulus,** then eigenvalues are selected according to their moduli: $w_l \leq |\lambda_i| \leq w_u$.

if **crit** = **Nag_Select_RealPart**, then eigenvalues are selected according to their real parts: $w_l \leq \mathrm{Re}(\lambda_i) \leq w_u$.

Constraint: **crit** = **Nag_Select_Modulus** or **Nag_Select_RealPart**.

n

Input: n, the order of the matrix A.

Constraint: $\mathbf{n} \geq 0$.

a[n][tda]

Note: The first dimension of **a** must be at least 1.
Input: the n by n general matrix A.
Output: **a** contains the Hessenberg form of the balanced input matrix A' (see Section 6).

tda

Input: the last dimension of the array **a** as declared in the calling program.
Constraint: **tda** $\geq$ max (1,**n**).

wl
wu
Input: w_l and w_u, the lower and upper bounds on the criterion for the selected eigenvalues.
Constraint: **wu** $>$ **wl**.

mest
Input: **mest** must be an upper bound on m, the number of eigenvalues and eigenvectors selected. No eigenvectors are computed if **mest** $< m$.
Constraint: **mest** $\geq \max(1, m)$.

m
Output: m, the number of eigenvalues actually selected.

w[n]
Note: The dimension of **w** must be at least 1.
Output: the first **m** elements of **w** hold the values of the selected eigenvalues; elements from the index **m** to **n**-1 contain the other eigenvalues. Complex conjugate pairs of eigenvalues are stored in consecutive elements of the array, with the eigenvalue having the positive imaginary part first.

v[n][tdv]
Note: The first dimension of **v** must be at least 1.
Output: **v** contains the selected eigenvectors, with the ith column holding the eigenvector associated with the eigenvalue λ_i (stored in **w**$[i-1]$).

tdv
Input: the second dimension of the array **v** as declared in the calling program.
Constraint: **tdv** $\geq$ **mest**.

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

5. Error Indications and Warnings

NE_BAD_PARAM
On entry, parameter **crit** had an illegal value.

NE_INT_ARG_LT
On entry, **n** must not be less than 0: **n** = ⟨*value*⟩.
On entry, **mest** must not be less than 1: **mest** = ⟨*value*⟩.

NE_INT_2
On entry, **tda** = ⟨*value*⟩ while **n** = ⟨*value*⟩.
Constraint: **tda** $\geq$ max(1,**n**).
On entry, **tdv** = ⟨*value*⟩ while **mest** = ⟨*value*⟩.
Constraint: **tdv** $\geq$ max(1,**mest**).

NE_2_REAL_ARG_LE
On entry, **wu** = ⟨*value*⟩ while **wl** = ⟨*value*⟩.
These parameters must satisfy **wu** $>$ **wl**.

NE_QR_FAIL
The QR algorithm failed to compute all the eigenvalues. No eigenvectors have been computed.

NE_REQD_EIGVAL
There are more than **mest** eigenvalues in the specified range. The actual number of eigenvalues in the range is returned in **m**. No eigenvectors have been computed.
Rerun with the second dimension of **v** = **mest** $\geq$ **m**.

NE_EIGVEC
Inverse iteration failed to compute all the specified eigenvectors. If an eigenvector failed to converge, the corresponding column of **v** is set to zero.

NE_ALLOC_FAIL
Memory allocation failed.

6. Further Comments

The function first balances the matrix, using a diagonal similarity transformation to reduce its norm; and then reduces the balanced matrix A' to upper Hessenberg form H, using an orthogonal similarity transformation: $A' = QHQ^T$. The routine uses the Hessenberg QR algorithm to compute all the eigenvalues of H, which are the same as the eigenvalues of A. It computes the eigenvectors of H which correspond to the selected eigenvalues, using inverse iteration. It premultiplies the eigenvectors by Q to form the eigenvectors of A'; and finally transforms the eigenvectors to those of the original matrix A.

Each eigenvector x (real or complex) is normalized so that $\|x\|_2 = 1$, and the element of largest absolute value is real and positive.

The inverse iteration routine may make a small perturbation to the real parts of close eigenvalues, and this may shift their moduli just outside the specified bounds. If you are relying on eigenvalues being within the bounds, you should test them on return from nag_real_eigensystem_sel.

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

The routine can be used to compute *all* eigenvalues and eigenvectors, by setting **wl** large and negative, and **wu** large and positive.

6.1. Accuracy

If λ_i is an exact eigenvalue, and $\tilde{\lambda}_i$ is the corresponding computed value, then

$$|\tilde{\lambda}_i - \lambda_i| \leq \frac{c(n)\epsilon\|A'\|_2}{s_i},$$

where $c(n)$ is a modestly increasing function of n, ϵ is the ***machine precision***, and s_i is the reciprocal condition number of λ_i; A' is the balanced form of the original matrix A, and $\|A'\| \leq \|A\|$.

If x_i is the corresponding exact eigenvector, and $\tilde{x}_i$ is the corresponding computed eigenvector, then the angle $\theta(\tilde{x}_i, x_i)$ between them is bounded as follows:

$$\theta(\tilde{x}_i, x_i) \leq \frac{c(n)\epsilon\|A'\|_2}{sep_i}$$

where sep_i is the reciprocal condition number of x_i.

6.2. References

Golub G H and Van Loan C F (1989) *Matrix Computations* Johns Hopkins University Press (2nd Edition), Baltimore.

7. See Also

nag_real_eigenvalues (f02afc)
nag_real_eigensystem (f02agc)

8. Example

To compute those eigenvalues of the matrix A whose moduli lie in the range [0.2,0.5], and their corresponding eigenvectors, where

$$\mathrm{A} = \begin{pmatrix} 0.35 & 0.45 & -0.14 & -0.17 \\ 0.09 & 0.07 & -0.54 & 0.35 \\ -0.44 & -0.33 & -0.03 & 0.17 \\ 0.25 & -0.32 & -0.13 & 0.11 \end{pmatrix}$$

8.1. Program Text

```
/* nag_real_eigensystem_sel(f02ecc) Example Program.
 *
 * Copyright 1998 Numerical Algorithms Group.
 *
 * Mark 5, 1998.
 *
 */
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nagf02.h>

#define NMAX 8
#define MMAX 3
#define TDA NMAX
#define TDV MMAX

main()
{

  double a[NMAX][NMAX];
  double wl, wu;

  Integer i, j, m, n;
  Integer mest = MMAX;
  Integer tda=TDA, tdv=TDV;

  Complex w[NMAX], v[NMAX][MMAX];

  Vprintf("f02ecc Example Program Results\n\n");

  /*  Skip heading in data file */
  Vscanf("%*[^\n]");

  Vscanf("%ld%lf%lf%*[^\n]",&n, &wl, &wu);

  if (n <= NMAX)
    {
      /*  Read a from data file */
      for (i = 0; i < n; ++i)
        for (j = 0; j < n; ++j)
          Vscanf("%lf",&a[i][j]);
      Vscanf("%*[^\n]");

      /*  Compute selected eigenvalues and eigenvectors of A */

      f02ecc(Nag_Select_Modulus, n, (double *)a, tda, wl, wu, mest, &m, w,
             (Complex *)v, tdv, NAGERR_DEFAULT);

      Vprintf("\n\nEigenvalues\n");
      for (i = 0; i < m; ++i)
        Vprintf("(%7.4f, %7.4f)  ", w[i].re, w[i].im);
      Vprintf("\n");


      Vprintf(" Eigenvectors\n    ");
      for (i = 1; i <= m; i++)
        Vprintf("%15ld%s", i, i%m == 0 ?"\n" : "");
      for (i = 0; i < n; i++)
        {
          Vprintf("%ld  ", i + 1);
          for (j = 0; j < m; j++)
            Vprintf("(%8.4f, %8.4f)%s", v[i][j].re,
                    v[i][j].im, (j + 1) % m == 0 ? "\n":"  ");
        }
      exit(EXIT_SUCCESS);
    }
  else
```

```
    {
      Vprintf( "Incorrect input value of n.\n");
      exit(EXIT_FAILURE);
    }
}
```

8.2. Program Data

```
f02ecc Example Program Data
  4  0.2  0.5                :Values of n, wl, wu
  0.35   0.45  -0.14  -0.17
  0.09   0.07  -0.54   0.35
 -0.44  -0.33  -0.03   0.17
  0.25  -0.32  -0.13   0.11   :End of matrix a
```

8.3. Program Results

```
f02ecc Example Program Results



Eigenvalues
(-0.0994,  0.4008)  (-0.0994, -0.4008)
 Eigenvectors
                  1             2
1  ( -0.1933,   0.2546)  ( -0.1933,  -0.2546)
2  (  0.2519,  -0.5224)  (  0.2519,   0.5224)
3  (  0.0972,  -0.3084)  (  0.0972,   0.3084)
4  (  0.6760,   0.0000)  (  0.6760,   0.0000)
```