NAG Library Function Document
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
nag_real_eigensystem_sel (f02ecc) computes selected eigenvalues and the corresponding right eigenvectors of a real general matrix $A$:

$$Ax_i = \lambda_i x_i.$$ 

Eigenvalues $\lambda_i$ may be selected either by modulus, satisfying:

$$wl \leq |\lambda_i| \leq wu,$$

or by real part, satisfying:

$$wl \leq \text{Re}(\lambda_i) \leq wu.$$ 

Note that even though $A$ is real, $\lambda_i$ and $x_i$ may be complex. If $x_i$ is an eigenvector corresponding to a complex eigenvalue $\lambda_i$, 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 $\lambda_i$ and $\bar{\lambda}_i$ are either both selected or both not selected.

4 References

5 Arguments
1: crit – Nag_Select_Eigenvalues
   Input
   On entry: indicates the criterion for selecting eigenvalues:
   - if crit = Nag_Select_Modulus, then eigenvalues are selected according to their moduli:
     $wl \leq |\lambda_i| \leq wu$.
   - if crit = Nag_Select_RealPart, then eigenvalues are selected according to their real parts:
     $wl \leq \text{Re}(\lambda_i) \leq wu$.
   Constraint: crit = Nag_Select_Modulus or Nag_Select_RealPart.
2:  \textbf{n} – Integer \hspace{1cm} Input

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

3:  \textbf{a[\textbf{n} \times \textbf{tda}]} – double \hspace{1cm} Input/Output

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

\textit{On entry:} the \(n\times n\) general matrix \(A\).
\textit{On exit:} \(a\) contains the Hessenberg form of the balanced input matrix \(A'\) (see Section 9).

4:  \textbf{tda} – Integer \hspace{1cm} Input

\textit{On entry:} the stride separating matrix column elements in the array \(a\).
\textit{Constraint:} \(\text{tda} \geq \max(1, n)\).

5:  \textbf{wl} – double \hspace{1cm} Input

6:  \textbf{wu} – double \hspace{1cm} Input

\textit{On entry:} \(w_l\) and \(w_u\), the lower and upper bounds on the criterion for the selected eigenvalues.
\textit{Constraint:} \(w_u > w_l\).

7:  \textbf{mest} – Integer \hspace{1cm} Input

\textit{On entry:} \(mest\) must be an upper bound on \(m\), the number of eigenvalues and eigenvectors selected. No eigenvectors are computed if \(mest < m\).
\textit{Constraint:} \(mest \geq \max(1, m)\).

8:  \textbf{m} – Integer * \hspace{1cm} Output

\textit{On exit:} \(m\), the number of eigenvalues actually selected.

9:  \textbf{w[\max(1, n)]} – Complex \hspace{1cm} Output

\textit{On exit:} 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.

10: \textbf{v[\textbf{n} \times \textbf{tdv}]} – Complex \hspace{1cm} Output

\textit{Note:} the \((i,j)\)th element of the matrix \(V\) is stored in \(v[(i-1) \times \textbf{tdv} + j - 1]\).

\textit{On exit:} \(v\) contains the selected eigenvectors, with the \(i\)th column holding the eigenvector associated with the eigenvalue \(\lambda_i\) (stored in \(w[i - 1]\)).

11: \textbf{tdv} – Integer \hspace{1cm} Input

\textit{On entry:} the stride separating matrix column elements in the array \(v\).
\textit{Constraint:} \(\text{tdv} \geq \text{mest}\).

12: \textbf{fail} – NagError * \hspace{1cm} Input/Output

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

6 \hspace{1cm} \textbf{Error Indicators and Warnings}

\textbf{NE_2_REAL_ARG_LE}

\textit{On entry,} \(w_u = (\text{value})\) while \(w_l = (\text{value})\). These arguments must satisfy \(w_u > w_l\).
NE_ALLOC_FAIL

Dynamic memory allocation failed.

NE_BAD_PARAM

On entry, argument crit had an illegal value.

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_INT_2

On entry, tda = ⟨value⟩ while n = ⟨value⟩.
Constraint: tda ≥ max(1, n).

On entry, tdv = ⟨value⟩ while mest = ⟨value⟩.
Constraint: tdv ≥ max(1, mest).

NE_INT_ARG_LT

On entry, mest = ⟨value⟩.
Constraint: mest ≥ 1.

On entry, n = ⟨value⟩.
Constraint: n ≥ 0.

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 ≥ m.

7 Accuracy

If λᵢ is an exact eigenvalue, and ̂λᵢ is the corresponding computed value, then

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

where c(n) is a modestly increasing function of n, ε is the machine precision, and sᵢ is the reciprocal condition number of λᵢ; A' is the balanced form of the original matrix A, and \(\|A^\prime\| \leq \|A\|\).

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

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

where \(sep_i\) is the reciprocal condition number of \(x_i\).

8 Parallelism and Performance

Not applicable.
9 Further Comments

nag_real_eigensystem_sel (f02ecc) 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 function 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 function 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 (f02ecc).

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

The function can be used to compute all eigenvalues and eigenvectors, by setting $w_l$ large and negative, and $w_u$ large and positive.

10 Example

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

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

10.1 Program Text

/* nag_real_eigensystem_sel (f02ecc) Example Program. *
 * Copyright 2014 Numerical Algorithms Group.
 * * Mark 5, 1998.
 * */
#include <nag.h>
#include <nag_stdlib.h>
#include <stdio.h>
#include <nagf02.h>
#define MMAX 3
#define A(I, J) a[(I) *tda + J]
#define V(I, J) v[(I) *tdv + J]
int main(void)
{
  Complex *v = 0, *w = 0;
  Integer exit_status = 0, i, j, m, mest, n, tda, tdv;
  double *a = 0, w1, wu;
  NagError fail;
  INIT_FAIL(fail);

  printf(
    "nag_real_eigensystem_sel (f02ecc) Example Program Results\n\n"
  );

  /* Skip heading in data file */
  #ifdef _WIN32
  f02ecc
  NAG Library Manual
  f02ecc.4 Mark 25
  */
if (n >= 0)
{
    if (!(a = NAG_ALLOC(n*n, double)) ||
         !(w = NAG_ALLOC(n, Complex)) ||
         !(v = NAG_ALLOC(n*mest, Complex)))
    {
        printf("Allocation failure\n");
        exit_status = -1;
        goto END;
    }
    tda = n;
    tdv = mest;
}
else
{
    printf("Invalid n.\n");
    exit_status = 1;
    return exit_status;
}
/* Read a from data file */
for (i = 0; i < n; ++i)
    for (j = 0; 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
/* Compute selected eigenvalues and eigenvectors of A */
/* nag_real_eigensystem_sel (f02ecc).  
* Computes selected eigenvalues and eigenvectors of a real  
* general matrix  
*/
nag_real_eigensystem_sel(Nag_Select_Modulus, n, a, tda, w[0], w[u], mest, &m, w,  
v, tdv, &fail);
if (fail.code != NE_NOERROR)
{
    printf("Error from nag_real_eigensystem_sel (f02ecc).\n%\n",  
        fail.message);
    exit_status = 1;
    goto END;
}
printf("\n\nEigenvalues\n");
for (i = 0; i < m; ++i)
    printf("(%7.4f, %7.4f) ", w[i].re, w[i].im);
printf("\n");

printf(" Eigenvectors\n  ");
for (i = 0; i < n; i++)
    printf("%15"NAG_IFMT"%s", i, i%%m == 0?"\n":"");
for (i = 0; i < n; i++)
    {  

Mark 25

f02ecc.5
printf("%"NAG_IFMT" ", i + 1);
for (j = 0; j < m; j++)
    printf("(%8.4f, %8.4f)%s", V(i, j).re, V(i, j).im, (j + 1) % m == 0?"\n":" ");
}
END:
NAG_FREE(a);
NAG_FREE(w);
NAG_FREE(v);
return exit_status;
}

10.2 Program Data

nag_real_eigensystem_sel (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

10.3 Program Results

nag_real_eigensystem_sel (f02ecc) Example Program Results

Eigenvalues
(-0.0994, 0.4008) (-0.0994, -0.4008)
Eigenvectors
1
   1 (-0.1933, 0.2546) ( -0.1933, -0.2546)
   2
   3
   4

2
   2
   3
   4

3
   3
   4

4
   4
   4