eigensystemselect( ) — Compute selected eigenvectors and eigenvalues

Description   Syntax   Remarks and examples   Conformability
Diagnostics   Also see

Description

`eigensystemselectr(`*A*`,` *range*`,` *X*`,` *L*`)` computes selected right eigenvectors of a square, numeric matrix *A* along with their corresponding eigenvalues. Only the eigenvectors corresponding to selected eigenvalues are computed. Eigenvalues that lie in a range are selected. The selected eigenvectors are returned in *X*, and their corresponding eigenvalues are returned in *L*.

> *range* is a vector of length 2. All eigenvalues with absolute value in the half-open interval (*range*`[1]`, *range*`[2]`] are selected.

`lefteigensystemselectr(`*A*`,` *range*`,` *X*`,` *L*`)` mirrors `eigensystemselectr()`, the difference being that it computes selected left eigenvectors instead of selected right eigenvectors.

`eigensystemselecti(`*A*`,` *index*`,` *X*`,` *L*`)` computes selected right eigenvectors of a square, numeric matrix, *A*, along with their corresponding eigenvalues. Only the eigenvectors corresponding to selected eigenvalues are computed. Eigenvalues are selected by an index. The selected eigenvectors are returned in *X*, and the selected eigenvalues are returned in *L*.

> *index* is a vector of length 2. The eigenvalues are sorted by their absolute values, in descending order. The eigenvalues whose rank is *index*`[1]` through *index*`[2]`, inclusive, are selected.

`lefteigensystemselecti(`*A*`,` *index*`,` *X*`,` *L*`)` mirrors `eigensystemselecti()`, the difference being that it computes selected left eigenvectors instead of selected right eigenvectors.

`eigensystemselectf(`*A*`,` *f*`,` *X*`,` *L*`)` computes selected right eigenvectors of a square, numeric matrix, *A*, along with their corresponding eigenvalues. Only the eigenvectors corresponding to selected eigenvalues are computed. Eigenvalues are selected by a user-written function described below. The selected eigenvectors are returned in *X*, and the selected eigenvalues are returned in *L*.

`lefteigensystemselectf(`*A*`,` *f*`,` *X*`,` *L*`)` mirrors `eigensystemselectf()`, the difference being that it computes selected left eigenvectors instead of selected right eigenvectors.

`symeigensystemselectr(`*A*`,` *range*`,` *X*`,` *L*`)` computes selected eigenvectors of a symmetric (Hermitian) matrix, *A*, along with their corresponding eigenvalues. Only the eigenvectors corresponding to selected eigenvalues are computed. Eigenvalues that lie in a range are selected. The selected eigenvectors are returned in *X*, and their corresponding eigenvalues are returned in *L*.

`symeigensystemselecti(`*A*`,` *index*`,` *X*`,` *L*`)` computes selected eigenvectors of a symmetric (Hermitian) matrix, *A*, along with their corresponding eigenvalues. Only the eigenvectors corresponding to selected eigenvalues are computed. Eigenvalues are selected by an index. The selected eigenvectors are returned in *X*, and the selected eigenvalues are returned in *L*.

`_eigenselectr_la()`, `_eigenselecti_la()`, `_eigenselectf_la()`, `_eigenselect_la()`, and `_symeigenselect_la()` are the interfaces into the [M-1] **LAPACK** routines used to implement the above functions. Their direct use is not recommended.

Syntax

void `eigensystemselectr(`*A*, *range*, *X*, *L*`)`

void `lefteigensystemselectr(`*A*, *range*, *X*, *L*`)`

void `eigensystemselecti(`*A*, *index*, *X*, *L*`)`

void `lefteigensystemselecti(`*A*, *index*, *X*, *L*`)`

void `eigensystemselectf(`*A*, *f*, *X*, *L*`)`

void `lefteigensystemselectf(`*A*, *f*, *X*, *L*`)`

void `symeigensystemselectr(`*A*, *range*, *X*, *L*`)`

void `symeigensystemselecti(`*A*, *index*, *X*, *L*`)`

where inputs are

A:	*numeric matrix*	
range:	*real vector*	(range of eigenvalues to be selected)
index:	*real vector*	(indices of eigenvalues to be selected)
f:	*pointer scalar*	(points to a function used to select eigenvalue)

and outputs are

X:	*numeric matrix* of eigenvectors
L:	*numeric vector* of eigenvalues

The following routines are used in implementing the above routines:

void `_eigenselecti_la(`*numeric matrix A*, *XL*, *XR*, *L*,
string scalar side, *real vector index*`)`

void `_eigenselectr_la(`*numeric matrix A*, *XL*, *XR*, *L*,
string scalar side, *real vector range*`)`

void `_eigenselectf_la(`*numeric matrix A*, *XL*, *XR*, *L*,
string scalar side, *pointer scalar f*`)`

real scalar `_eigenselect_la(`*numeric matrix A*, *XL*, *XR*, *L*, *select*,
string scalar side, *real scalar noflopin*`)`

real scalar `_symeigenselect_la(`*numeric matrix A*, *X*, *L*, *ifail*,
real scalar type, *lower*, *upper*, *abstol*`)`

Remarks and examples

Remarks are presented under the following headings:

Introduction

These routines compute subsets of the available eigenvectors. This computation can be much faster than computing all the eigenvectors. (See [M-5] **eigensystem( )** for routines to compute all the eigenvectors and an introduction to the eigensystem problem.)

There are three methods for selecting which eigenvectors to compute; all of them are based on the corresponding eigenvalues. First, we can select only those eigenvectors whose eigenvalues have absolute values that fall in a half-open interval. Second, we can select only those eigenvectors whose eigenvalues have certain indices, after sorting the eigenvalues by their absolute values in descending order. Third, we can select only those eigenvectors whose eigenvalues meet a criterion encoded in a function.

Below we illustrate each of these methods. For comparison purposes, we begin by computing all the eigenvectors of the matrix

```
: A
         1     2     3     4
    +-------------------------+
  1 |  .31   .69   .13   .56  |
  2 |  .31    .5   .72   .42  |
  3 |  .68   .37   .71    .8  |
  4 |  .09   .16   .83    .9  |
    +-------------------------+
```

We perform the computation with `eigensystem()`:

```
: eigensystem(A, X=., L=.)
```

The absolute values of the eigenvalues are

```
: abs(L)
                 1              2              3              4
    +-------------------------------------------------------------+
  1 |  2.10742167    .4658398402    .4005757984    .4005757984    |
    +-------------------------------------------------------------+
```

The corresponding eigenvectors are

```
: X
                 1              2                              3
    +-----------------------------------------------------------+
  1 |  .385302069   -.394945842                      .672770333 |
  2 |  .477773165   -.597299386   -.292386384 - .171958335i     |
  3 |  .604617181   -.192938403   -.102481414 + .519705293i     |
  4 |  .50765459     .670839771    -.08043663 - .381122722i     |
    +-----------------------------------------------------------+

                              4
    +-------------------------------+
  1 |                  .672770333   |
  2 | -.292386384 + .171958335i     |
  3 | -.102481414 - .519705293i     |
  4 | -.08043663  + .381122722i     |
    +-------------------------------+
```

Range selection

In applications, an eigenvalue whose absolute value is greater than 1 frequently corresponds to an explosive solution, whereas an eigenvalue whose absolute value is less than 1 corresponds to a stable solution. We frequently want to use only the eigenvectors from the stable solutions, which we can do using `eigensystemselectr()`. We begin by specifying

```
: range = (-1, .999999999)
```

which starts from −1 to include 0 and stops at .999999999 to exclude 1. (The half-open interval in `range` is open on the left and closed on the right.)

Using this `range` in `eigensystemselectr()` requests each eigenvector for which the absolute value of the corresponding eigenvalue falls in the interval $(-1, .999999999]$. For the example at hand, we have

```
: eigensystemselectr(A, range, X=., L=.)
: X
                 1                            2                            3
    +-------------------------------------------------------------------------+
  1 |  -.442004357    .201218963 - .875384534i    .201218963 + .875384534i    |
  2 |  -.668468693    .136296114 + .431873675i    .136296114 - .431873675i    |
  3 |  -.215927364   -.706872994 - .022093594i   -.706872994 + .022093594i    |
  4 |   .750771548    .471845361 + .218651289i    .471845361 - .218651289i    |
    +-------------------------------------------------------------------------+
: L
                1                            2                            3
    +-------------------------------------------------------------------------+
  1 |   .46583984   -.076630755 + .393177692i   -.076630755 - .393177692i     |
    +-------------------------------------------------------------------------+
: abs(L)
                  1              2              3
    +--------------------------------------------+
  1 |   .4658398402    .4005757984    .4005757984 |
    +--------------------------------------------+
```

The above output illustrates that `eigensystemselectr()` has not included the results for the eigenvalue whose absolute value is greater than 1, as desired.

Index selection

In many statistical applications, an eigenvalue measures the importance of an eigenvector factor. In these applications, we want only to compute several of the largest eigenvectors. Here we use `eigensystemselecti()` to compute the eigenvectors corresponding to the two largest eigenvalues:

```
: index = (1, 2)
: eigensystemselecti(A, index, X=., L=.)
: L
                 1             2
    +----------------------------+
  1 |   2.10742167     .46583984 |
    +----------------------------+
```

```
: X
                  1             2
    +-----------------------------+
  1 |   .385302069   -.442004357  |
  2 |   .477773165   -.668468693  |
  3 |   .604617181   -.215927364  |
  4 |   .50765459     .750771548  |
    +-----------------------------+
```

Criterion selection

In some applications, we want to compute only those eigenvectors whose corresponding eigenvalues satisfy a more complicated criterion. We can use `eigensystemselectf()` to solve these problems.

We must pass `eigensystemselectf()` a pointer to a function that implements our criterion. The function must accept a complex scalar argument so that it can receive an eigenvalue, and it must return the real value 0 to indicate rejection and a nonzero real value to indicate selection.

In the example below, we consider the common criterion of whether the eigenvalue is real. We want only to compute the eigenvectors corresponding to real eigenvalues. After deciding that anything smaller than 1e–15 is zero, we define our function to be

```
: real scalar onlyreal(complex scalar ev)
> {
>         return( (abs(Im(ev))<1e-15) )
> }
```

We compute only the eigenvectors corresponding to the real eigenvalues by typing

```
: eigensystemselectf(A, &onlyreal(), X=., L=.)
```

The eigenvalues that satisfy this criterion and their corresponding eigenvectors are

```
: L
                  1            2
    +--------------------------+
  1 |   2.10742167    .46583984 |
    +--------------------------+

: X
                  1             2
    +-----------------------------+
  1 |   .385302069   -.442004357  |
  2 |   .477773165   -.668468693  |
  3 |   .604617181   -.215927364  |
  4 |   .50765459     .750771548  |
    +-----------------------------+
```

Other functions

`lefteigensystemselectr()` and `symeigensystemselectr()` use a *range* like `eigensystemselectr()`.

`lefteigensystemselecti()` and `symeigensystemselecti()` use an *index* like `eigensystemselecti()`.

`lefteigensystemselectf()` uses a pointer to a function like `eigensystemselectf()`.

Conformability

`eigensystemselectr(`A, *range*, X, L`)`:

input:

A:	$n \times n$
range:	1×2 or 2×1

output:

X:	$n \times m$
L:	$1 \times m$

`lefteigensystemselectr(`A, *range*, X, L`)`:

input:

A:	$n \times n$
range:	1×2 or 2×1

output:

X:	$m \times n$
L:	$1 \times m$

`eigensystemselecti(`A, *index*, X, L`)`:

input:

A:	$n \times n$
index:	1×2 or 2×1

output:

X:	$n \times m$
L:	$1 \times m$

`lefteigensystemselecti(`A, *index*, X, L`)`:

input:

A:	$n \times n$
index:	1×2 or 2×1

output:

X:	$m \times n$
L:	$1 \times m$

`eigensystemselectf(`A, f, X, L`)`:

input:

A:	$n \times n$
f:	1×1

output:

X:	$n \times m$
L:	$1 \times m$

`lefteigensystemselectf(`A`,` f`,` X`,` L`)`:

input:

A:	$n \times n$
f:	1×1

output:

X:	$m \times n$
L:	$1 \times m$

`symeigensystemselectr(`A`,` *range*`,` X`,` L`)`:

input:

A:	$n \times n$
range:	1×2 or 2×1

output:

X:	$n \times m$
L:	$1 \times m$

`symeigensystemselecti(`A`,` *index*`,` X`,` L`)`:

input:

A:	$n \times n$
index:	1×2 or 2×1

output:

X:	$n \times m$
L:	$1 \times m$

Diagnostics

All functions return missing-value results if A has missing values.

`symeigensystemselectr()` and `symeigensystemselecti()` use the lower triangle of A without checking for symmetry. When A is complex, only the real part of the diagonal is used.

If the ith eigenvector failed to converge, `symeigensystemselectr()` and `symeigensystemselecti()` insert a vector of missing values into the ith column of the returned eigenvector matrix.

Also see

[M-1] **LAPACK** — Linear algebra package (LAPACK) routines

[M-5] **eigensystem( )** — Eigenvectors and eigenvalues

[M-5] **matexpsym( )** — Exponentiation and logarithms of symmetric matrices

[M-5] **matpowersym( )** — Powers of a symmetric matrix

[M-4] **Matrix** — Matrix functions

For suggested citations, see the FAQ on citing Stata documentation.