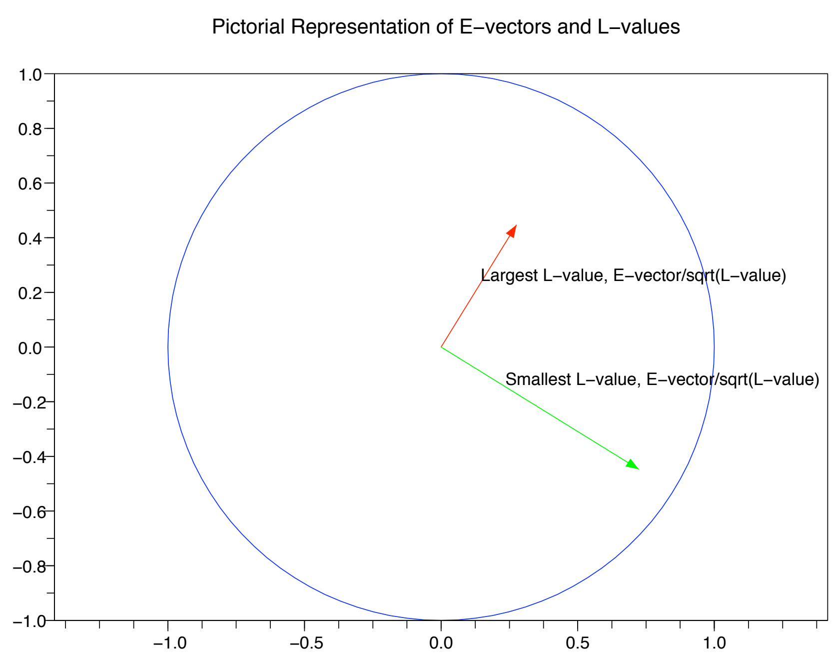
L-VALUES AND E-VECTORS

STEVEN F. BELLENOT

ABSTRACT. We describe iterative methods to get L-values and E-vectors. These are geometric properties of square matrices. We will limit the discussion to symmetric matrices. (General matrices are more complex than we need for Lab 1.) Later on we will that L-values and E-vectors will be (excellent) approximations to eigenvalues and eigenvectors.



CONTENTS

1. Introduction 2
2. Finding $L_{\max}$ and $E_{\max}$, for max Largest in Absolute Value 2
3. Finding $L_{\min}$ and $E_{\min}$, min for Smallest in Absolute Value 4
4. Finding other L_i and E_i in the Mid-Range 5

1. Introduction

Our goal is to find geometric knowledge of a symmetric matrix A like

$$A = \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

by iteration. We are doing special cases of a more general theory, but without any theory, all we are doing is looking at the output of computer programs.

2. Finding $L_{\max}$ and $E_{\max}$, for max Largest in Absolute Value

Consider the following iteration. Pick a non-zero unit column vector X at random. (A unit vector has length one. If Y is a non-zero vector, then $X = Y/|Y|$ is a unit vector in the same direction as Y. Here $|Y| = \sqrt{y_1^2 + y_2^2 + ... + y_n^2}$ is the norm or length of Y.) And then execute this for loop until in converges. (This loop is designed for 2×2 matrices A and 2×1 vectors X, it is in the scilab directory named iterate2d.in

```
for i = 1:10,
    X = A*X;
    N = norm(X);
    X = X/N;
printf("N = %f, x = [ %f; %f], i = %d\n", N, X(1,1), X(2,1), i);
end;
```

The output of the above code (which is in iterate2d.in, a file in our scilab directory) when

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \qquad X = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

So we start by typing:

```
A = [2,1;1,3]
X = [1;0]
exec('iterate2d.in')
```

then 'exec('iterate2d.in')'will produce the following output:

```
N = 2.236068, X = [0.894427; 0.447214], i = 1
N = 3.162278, X = [0.707107; 0.707107], i = 2
N = 3.535534, X = [0.600000; 0.800000], i = 3
N = 3.605551, X = [0.554700; 0.832050], i = 4
N = 3.616203, X = [0.536875; 0.843661], i = 5
N = 3.617767, X = [0.529999; 0.847998], i = 6
N = 3.617995, X = [0.527363; 0.849640], i = 7
N = 3.618028, X = [0.526355; 0.850265], i = 8
N = 3.618033, X = [0.525969; 0.850504], i = 9
N = 3.618034, X = [0.525822; 0.850595], i = 10
```

Clearly the values are close to converging but haven't yet converged. We do the 'exec('iterate2d.in')' another time. We do this by using the up-arrow key and then enter key. This starts the calculation where we last left off. And now we are doing i form 11 to 20, even though it says 1 to 10 again. Here is the next output:

```
N = 3.618034, X = [0.525766; 0.850629], i = 1
N = 3.618034, X = [0.525744; 0.850643], i = 2
N = 3.618034, X = [0.525736; 0.850648], i = 3
```

```
N = 3.618034, X = [0.525733; 0.850650], i = 4
N = 3.618034, X = [0.525732; 0.850650], i = 5
N = 3.618034, X = [0.525731; 0.850651], i = 6
N = 3.618034, X = [0.525731; 0.850651], i = 7
N = 3.618034, X = [0.525731; 0.850651], i = 8
N = 3.618034, X = [0.525731; 0.850651], i = 9
N = 3.618034, X = [0.525731; 0.850651], i = 10
```

Both N and X have settled. This is what happens when the largest value is positive. We find L using the next command, note the period before the slash:

```
A*X ./ X
ans  =
    3.618034
    3.618034
```

We would say $L_{\max} = 3.618034$ and $E_{\max} = [0.525731; 0.850651]$. Notice that

$$A * E_{\max} = L_{\max} * E_{\max}$$

to the limits of the numerical system.

It is not always this easy. There are two gotcha's. One's choice of X can be unlucky. To be sure, you need to test at least n independent vectors for a $n \times n$ matrix. In the 2×2 case, the only way it can go wrong is if X is the E-vector for a smaller L-value. If this is true, the convergence is immediate which isn't the case here.

The second problem, is that L might be negative. Replace A by -A and the (double) output becomes

```
N = 2.236068, X = [-0.894427; -0.447214], i = 1
N = 3.162278, X = [0.707107; 0.707107], i = 2
N = 3.535534, X = [-0.600000; -0.800000], i = 3
N = 3.605551, X = [0.554700; 0.832050], i = 4
N = 3.616203, X = [-0.536875; -0.843661], i = 5
N = 3.617767, X = [0.529999; 0.847998], i = 6
N = 3.617995, X = [-0.527363; -0.849640], i = 7
N = 3.618028, X = [0.526355; 0.850265], i = 8
N = 3.618033, X = [-0.525969; -0.850504], i = 9
N = 3.618034, X = [0.525822; 0.850595], i = 10

N = 3.618034, X = [-0.525766; -0.850629], i = 1
N = 3.618034, X = [0.525744; 0.850643], i = 2
N = 3.618034, X = [-0.525736; -0.850648], i = 3
N = 3.618034, X = [0.525733; 0.850650], i = 4
N = 3.618034, X = [-0.525732; -0.850650], i = 5
N = 3.618034, X = [0.525731; 0.850651], i = 6
N = 3.618034, X = [-0.525731; -0.850651], i = 7
N = 3.618034, X = [0.525731; 0.850651], i = 8
N = 3.618034, X = [-0.525731; -0.850651], i = 9
N = 3.618034, X = [0.525731; 0.850651], i = 10
```

where the N converges but not the X's. But note that the X's are alternating between two vectors which are negatives of each other.

```
A*X ./ X
ans  =
  - 3.618034
  - 3.618034
```

When this happens we make L negative, $L_{\max} = -3.618034$ and either X or $-X$ say the $E_{\max} = [0.525731; 0.850651]$. Both X and $-X$ can be the E-vector. In all cases

$$A * E_{\max} = L_{\max} * E_{\max}$$

If A was not symmetric, then there would more cases to handle. (L can be complex.)

3. Finding $L_{\min}$ and $E_{\min}$, min for Smallest in Absolute Value

There are two cases. If A has an inverse (which can be found in scilab by inv(A). Then compute $L_{\max}$ and $E_{\max}$ for the inverse by the same method, and $L_{\min} = 1/L_{\max}$ and $E_{\min} = E_{\max}$. For first the A above, we start again with A = inv(A), X = [1;0] and then the output of iterate2d.in twice is

```
N = 0.632456, X = [0.948683; -0.316228], i = 1
N = 0.707107, X = [0.894427; -0.447214], i = 2
N = 0.721110, X = [0.868243; -0.496139], i = 3
N = 0.723241, X = [0.857493; -0.514496], i = 4
N = 0.723553, X = [0.853282; -0.521450], i = 5
N = 0.723599, X = [0.851658; -0.524097], i = 6
N = 0.723606, X = [0.851036; -0.525107], i = 7
N = 0.723607, X = [0.850798; -0.525493], i = 8
N = 0.723607, X = [0.850707; -0.525640], i = 9
N = 0.723607, X = [0.850672; -0.525696], i = 10

N = 0.723607, X = [0.850659; -0.525718], i = 1
N = 0.723607, X = [0.850654; -0.525726], i = 2
N = 0.723607, X = [0.850652; -0.525729], i = 3
N = 0.723607, X = [0.850651; -0.525730], i = 4
N = 0.723607, X = [0.850651; -0.525731], i = 5
N = 0.723607, X = [0.850651; -0.525731], i = 6
N = 0.723607, X = [0.850651; -0.525731], i = 7
N = 0.723607, X = [0.850651; -0.525731], i = 8
N = 0.723607, X = [0.850651; -0.525731], i = 9
N = 0.723607, X = [0.850651; -0.525731], i = 10

A*X ./X
 ans  =
    0.7236068
    0.7236068
```

So $L_{\min} = 1/0.7236068 = 1.381966$ and $E_{\min} = [0.850651; -0.525731]$ Note that $E_{\max}$ and $E_{\min}$ are orthogonal.

The second case is when A does not have an inverse. Then $L_{\min} = 0$ and $E_{\min}$ is one of the two unit vectors for which $A * E_{\min} = 0$.

For our symmetric matrices the dot product of $E_{\max}$ and $E_{\min}$ is always zero when $L_{\max} \neq L_{\min}$.

4. Finding other L_i and E_i in the Mid-Range

If A is 2×2 then there are at most two L-values, but an $n \times n$ matrix can n of the them. Here is how to find the L nearest to number s. Replace A by inverse(A-s*I), find $E_{\max}$ and $L_{\max}$ for this new matrix. then $L_{\text{mid}} = s + 1/L_{\max}$ and $E_{\text{mid}} = E_{\max}$

For example consider

$$A = \begin{bmatrix} 4 & 1 & 1 \\ 1 & 3 & 1 \\ 1 & 1 & 5 \end{bmatrix}$$

We use iterate3d.in instead of iterate.

```
N = 6.214320, X = [0.520657; 0.397113; 0.755789], i = 10
```

So $L_{\max} = 6.214320, E_{\max} = [0.520657; 0.397113; 0.755789]$

```
N = 0.430132, X = [0.427132; -0.887650; 0.172148], i = 10
```

So $L_{\min} = 2.3248691, E_{\min} = [0.427132; -0.887650; 0.172148]$

pick $s = 4.2695946$ (midway between the two L values)

```
N = -1.236425, X = [0.739239; 0.233192; -0.631781], i = 10
```

So $L_{\text{mid}} = 3.460811, E_{\text{mid}} = [0.739239; 0.233192; -0.631781]$

The three vectors are only nearly orthogonal, apparently I stopped the iteration early. Rats.