Semih Akbayrak
2010401168
04.05.2015

EE550
HW4 Report

In this project, it is asked from us to design a Continuous Hopfield Model with two neuron and given weight matrix T=[0 1;1 0].

In the lectures, the dynamic equation for Continuos Hopfield Model is given to us. According to this dynamic equation, the current states of the outputs determine the potential of cells. Then these cell potentials give the new state of the outputs of neurons.

Dynamic equation is

Ci*dui/dt + ui/R = Summation-wrt-j(Tij*vj) + Ii with g(ui) = vi

External input Ii is given as 0. Also we can think the derivative as the change in cell potential u.

First of all I need to determine u, then I will utilize from it to find the output vi = g(ui). For the first neuron cell

v2*t21 = C1.delta(u1) + u1/R1

delta(u1) = v2*t21/C1 - u1/R1

where t21=1 the weight from second neuron's output to first neuron. Then

u1 = u1 + delta(u1)

by this formula I can compute u1 and v1new = g(u1). So these equations will form my algorithm. Scaling factor lambda given as 1.4 and for simplicity I took all the resistor and capacitors as 1

```
lambda = 1.4;
R1 = 1;
R2 = 1;
C1 = 1;
C2 = 1;
t12 = 1;
t21 = 1;
```

Then I wrote the activation function g(.)

```
function y = activate(x)
    y = 2/pi * atan(pi*x*lambda/2);
end
```

Because energy function is wanted from us to find, I wrote another two function both to take the inverse of activation and to calculate Energy function

```
revactivate= @(x) 2/(pi*lambda) * tan(pi*x/2);
function E = Energy(v1,v2)
    E = -1/2*(v1*v2+v2*v1) + integral(revactivate,0,v1)/R1 + integral(revactivate,0,v2)/R2;
end
```

Then I specified the initial potentials of neurons as

```
u = [0.1 -0.05];
```

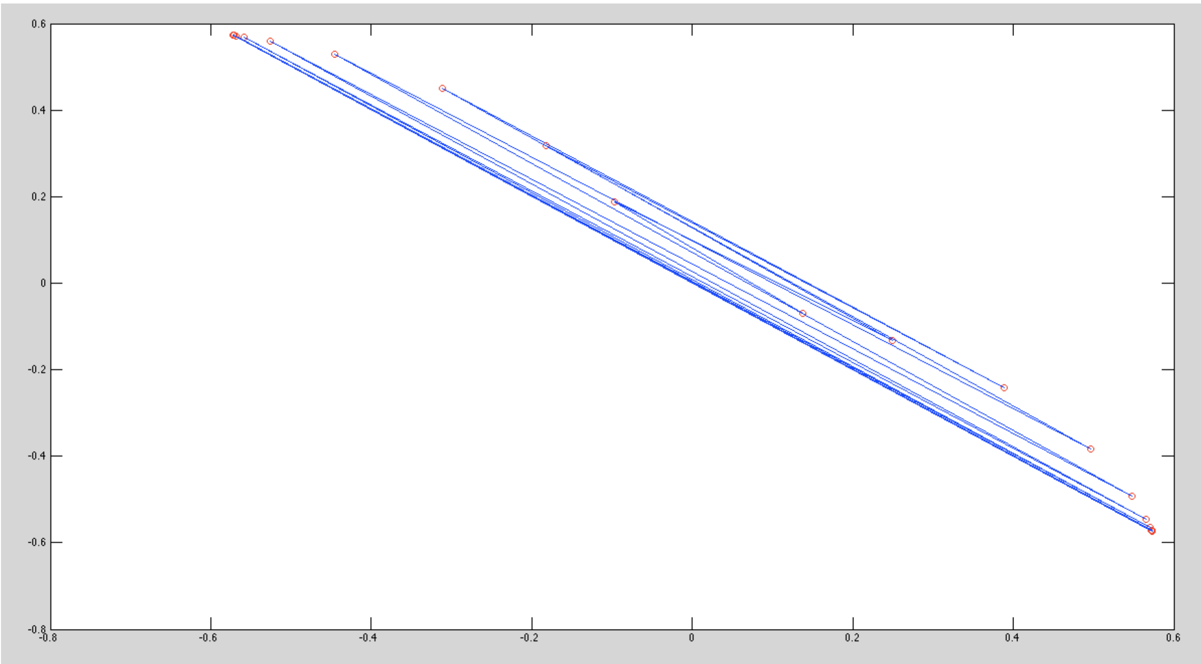
Below you can find the code blocks which were used to create matrix structures etc. both to use in plotting and computing.

```
v = activate(u);
delta = [0 0];
delta(1) = v(2)*t21/C1 - u(1)/R1;
delta(2) = v(1)*t12/C2 - u(2)/R2;
vone = [];
vtwo = [];
vone(1) = v(1);
vtwo(1) = v(2);
E = Energy(v(1),v(2));
Ematrix = [];
Ematrix(1) = E;
figure(1)
plot(v(1),v(2),'ro');
hold on;
```

In the first page, I mentioned about how I used the equations to create algorithm and below you can find the code block which the main algorithm for this project.

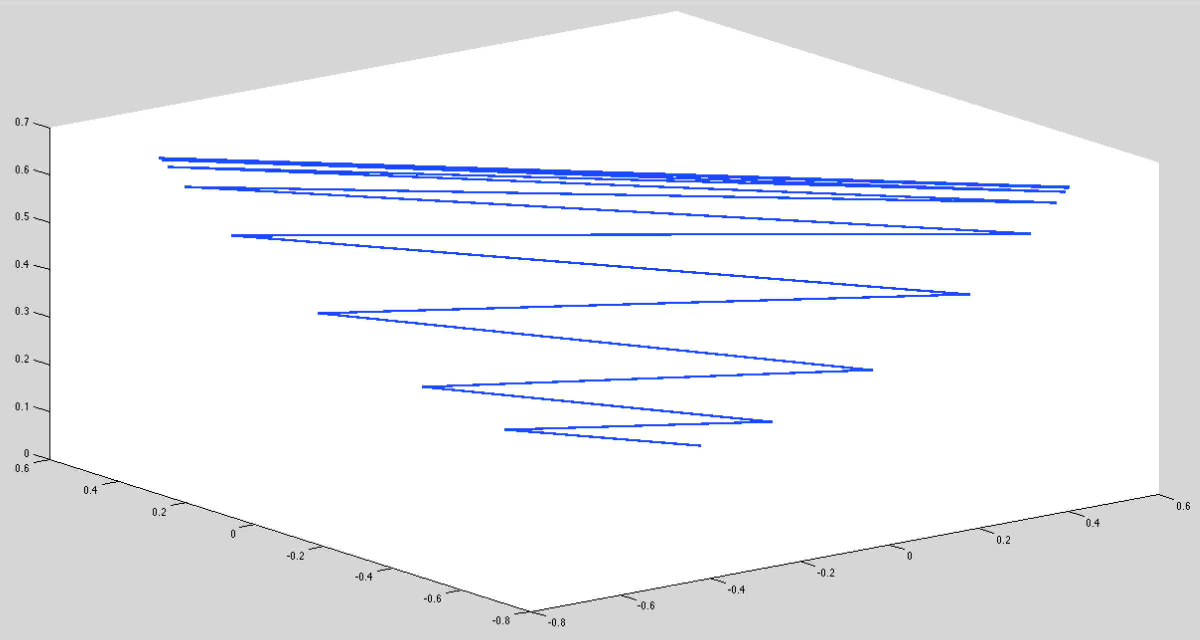
```
i = 0;
while 1
   i = i+1;
   u1 = u(1) + delta(1);
   u2 = u(2) + delta(2);
```

```
    u = [u1 u2];
    v = activate(u)
    delta(1) = v(2)/C1 - u(1)/R1;
    delta(2) = v(1)/C2 - u(2)/R2;
    E = Energy(v(1),v(2))
    vone(i+1) = v(1);
    vtwo(i+1) = v(2);
    Ematrix(i+1) = E;
    plot(v(1),v(2),'ro');
    hold on;
    if (Ematrix(i+1)-Ematrix(i))<0.0001
        break
    end
end
```

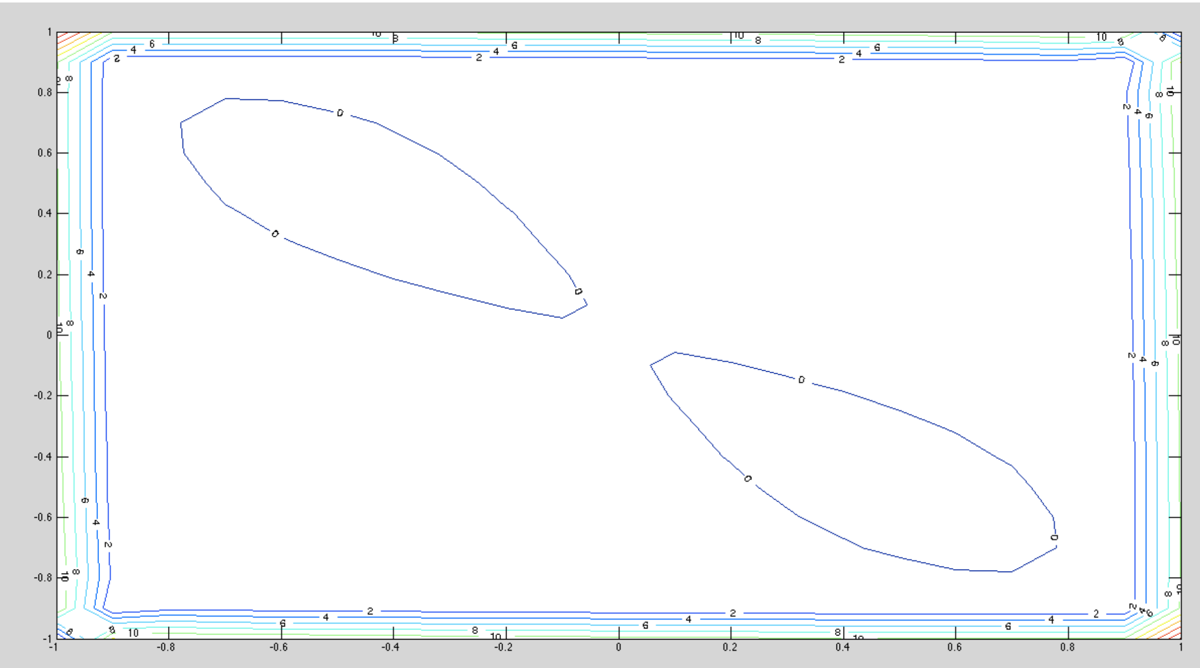


This plot shows the output trajectory of the outputs. As it can be seen the equation points for the output are around (-0.6,0.6) and (0.6,-0.6) for lambda=1.4

Below you can find the output trajectory versus energy function in 3D.



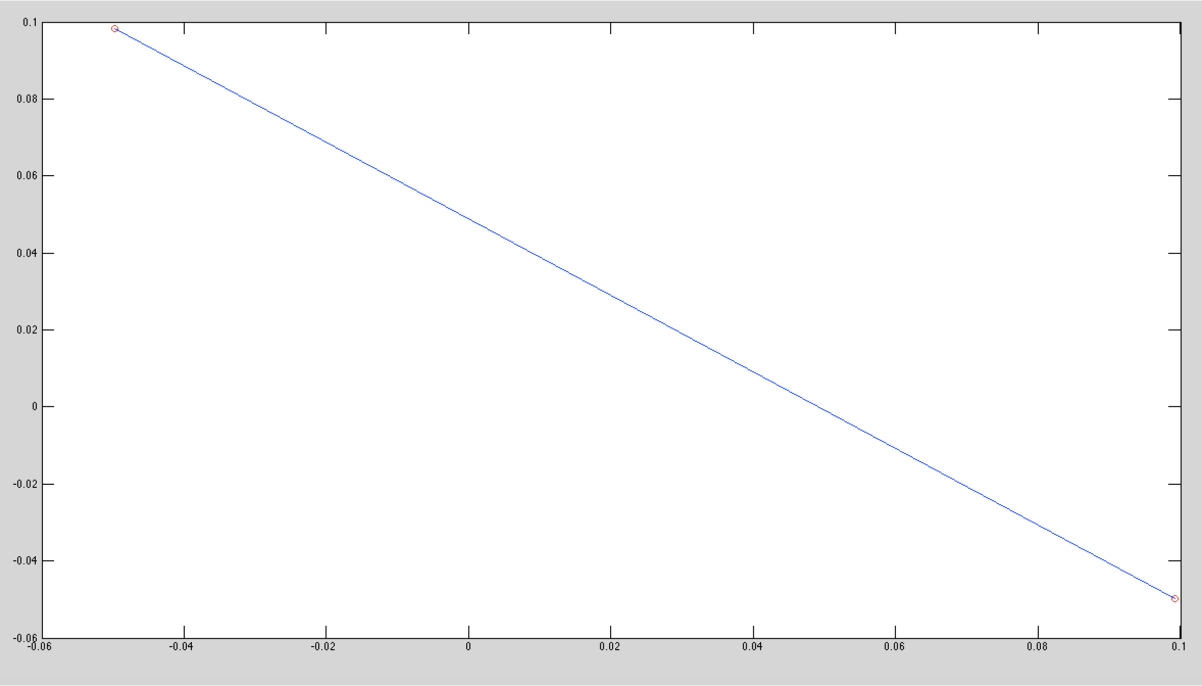
And the energy contour map is



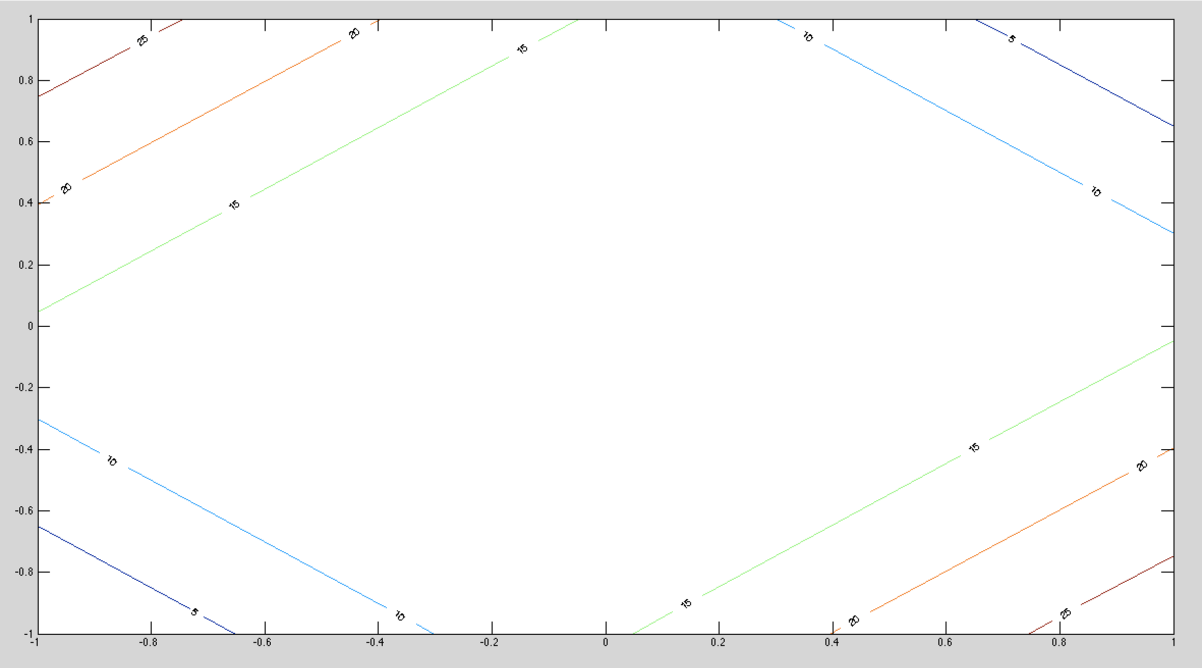
The red area represents the points with higher energy level. This contour map looks like topographic maps. Energy function is higher for the points (1,-1) and (-1,1) as expected. And energy is 0.6033.

So far I evaluated the results for lambda=1.4, now I will look at the results for lambda=1, 2, 5

lambda=1

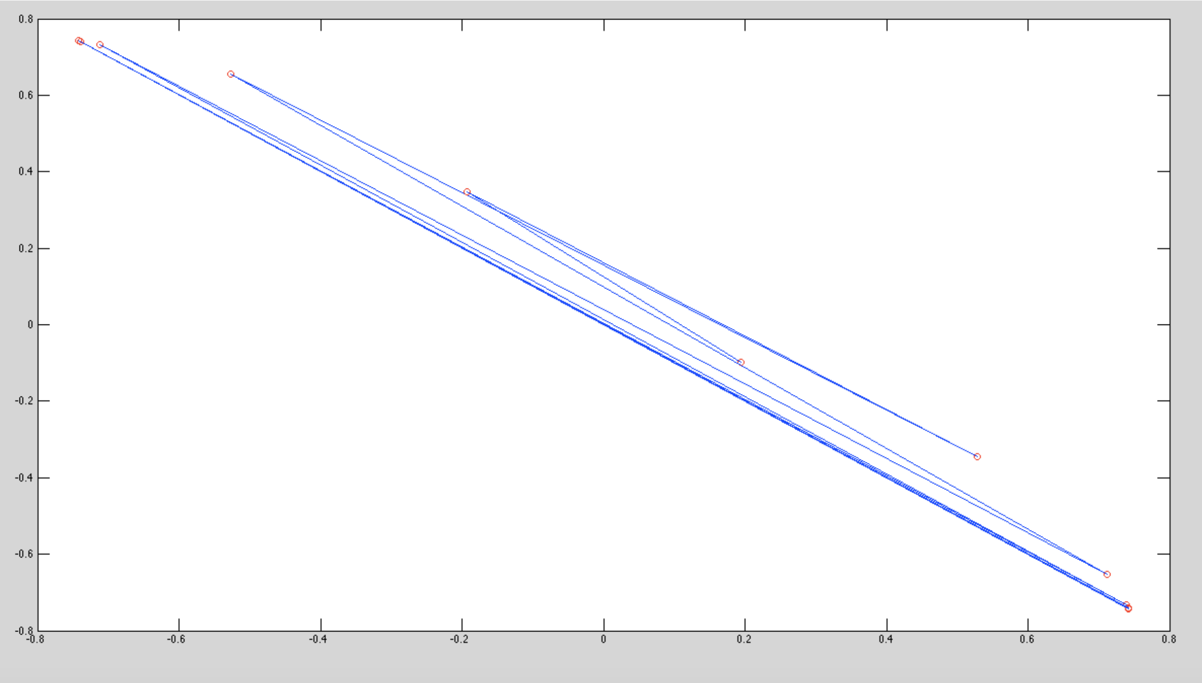


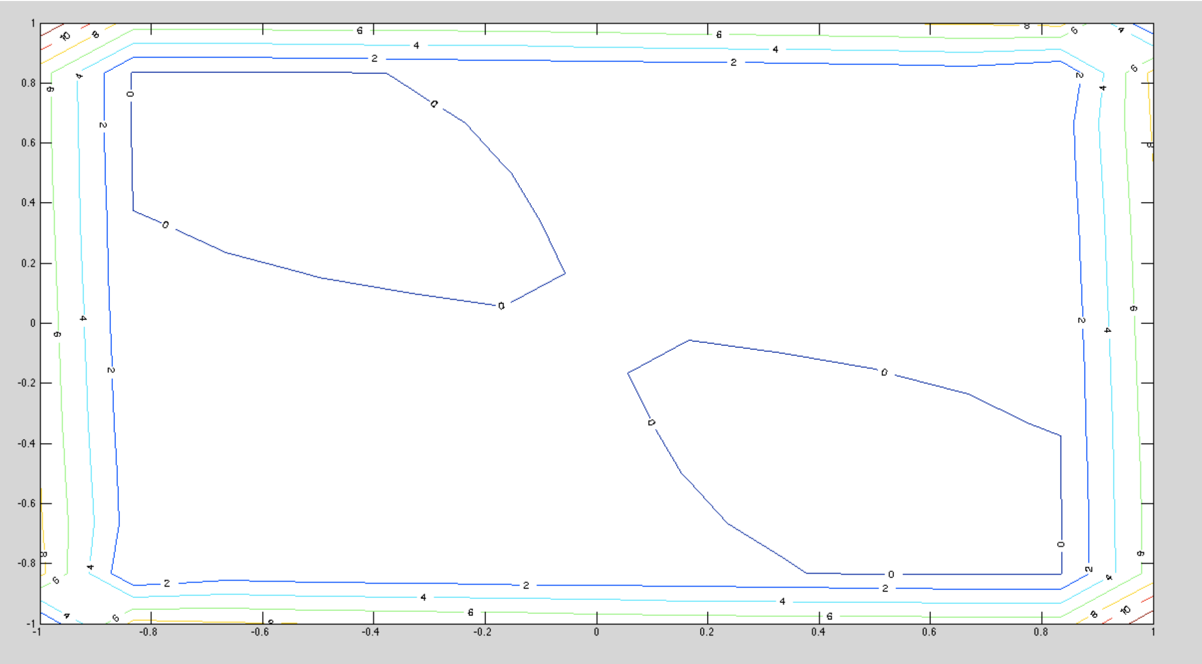
0.1
0.08
0.06
0.04
0.02
0
-0.02
-0.04
-0.06
-0.06
-0.04
-0.02
0
0.02
0.04
0.06
0.08
0.1



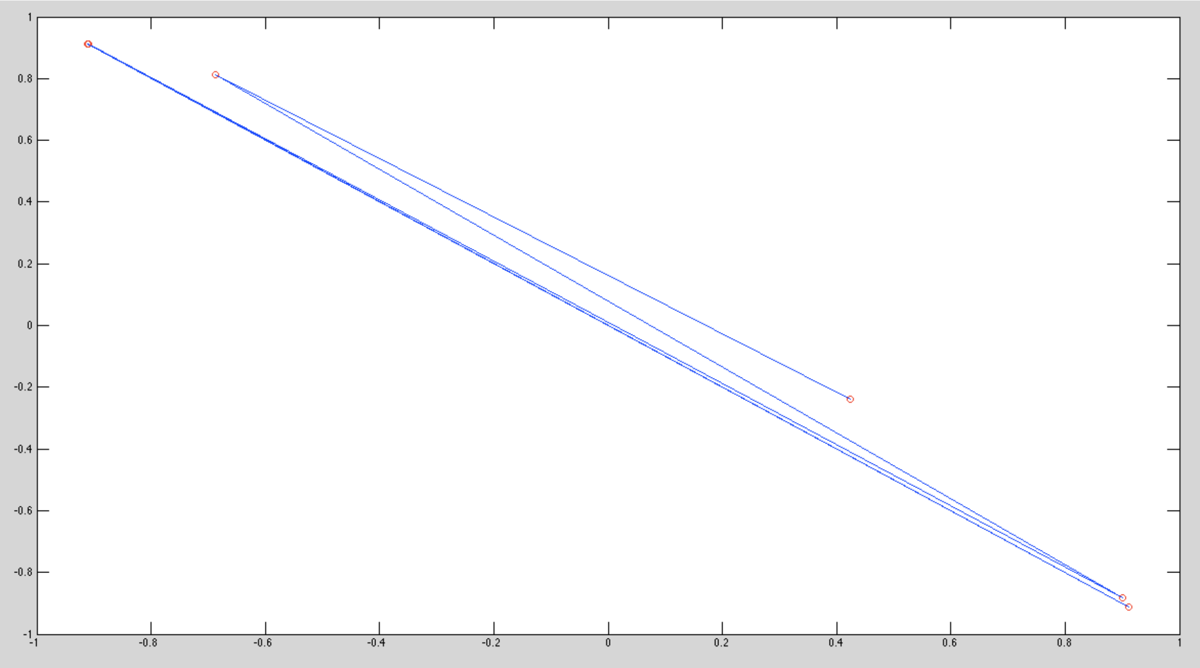
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
1
25
20
15
10
5

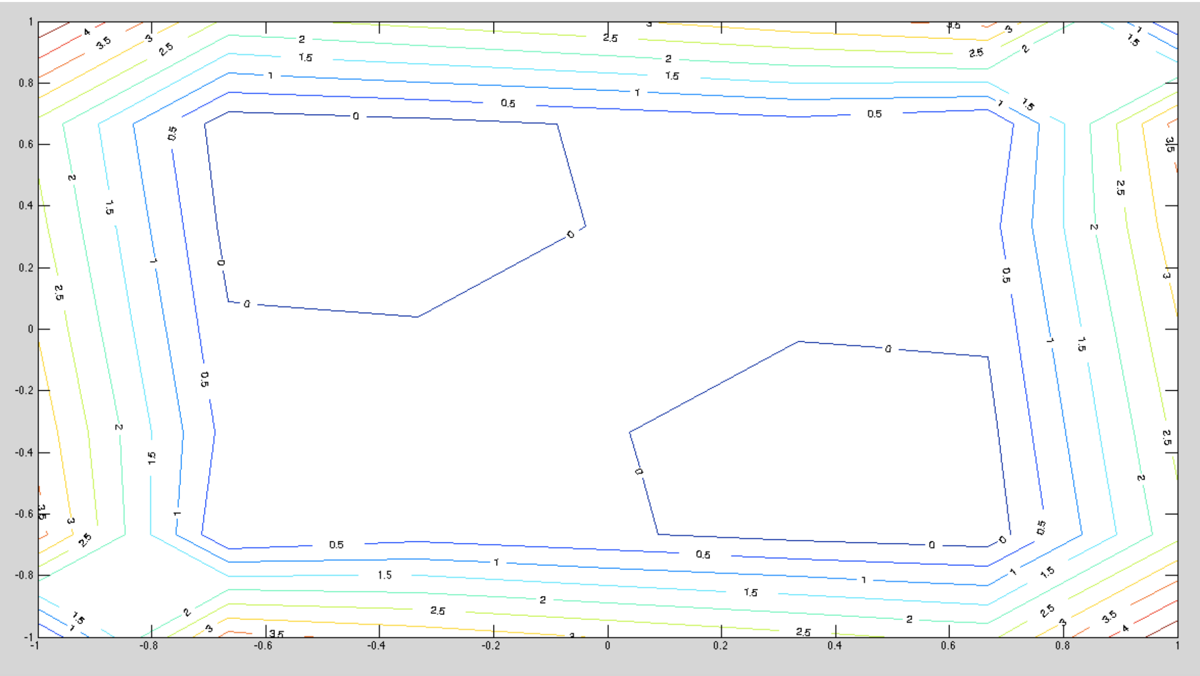
lambda=2





lambda=5





These plots show that, as lambda increasing which means as it converge to discrete case, and equilibrium points getting closer to (1,-1) and (-1,1).