Name - Naveen Kumar
N Number - N19068326
Net ID - nk3090

Course Name - ROB-GY 6323 Reinforcement Learning and Optimal Control for Robotics
Instructor - Prof. Ludovic Righetti

Summary of the task

In this project, I am supposed to write a Q Learning Algorithm which is a reinforcement learning algorithm to invert the pendulum. There are 3 control inputs are given and 50 state of both angular position and angular velocity are here and according to that we make a Q table.

Answer of Question 2

We need the Q table of numpy array of 50$\times$50$\times$3 dimension. Both he angular position and angular velocity range is descritized in 50 equally spaced points. And every particular iteration having a angular position and an angular position can take 3 possible actions according to the question. So the dimension of the q table is 50$\times$50$\times$3

Answer of Question 3

Both optimal value function and optimal policy are 50$\times$50 dimension.
In order to calculate the value function from the trained Q table, I am taking the minimum Q value from the corresponding 3 given values in the Q table and optimal policy is the respective input value (-u,0,u) where the minimum Q value is present in the Q table.

Answer of Question 4

In the Q learning algorithm, I have considered 10,000 episodes of 100 iterations. Here learning rate i.e. γ = 0.1, epsilon(ϵ) = 0.1, the initial point that I have considered is 0 radian with 0 angular velocity. Here the discount factor I am considering is 0.99. Initially the Every number in the Q table is zero.
I get initial u(input) from epsilon greedy policy.

$$u_t = \left\{arg\ min_u Q(x_t, u)\ with\ probability\ 1 - \epsilon\ or\ random\ possible\ action\ with\ probability\ \epsilon\right\}$$

In order to create a normal distribution of policy I used numpy.random.choice() function. So many a time the policy is chosen from the Q table because it has probability 1-ϵ which is equal to 0.9.
In the algorithm according to the given policy, I calculate the next state and I also calculate current cost using the function mentioned in the Jupyter notebook and that I have created during Answering question 1.
In order to update q value, I am calculating updated Q in the following manner.

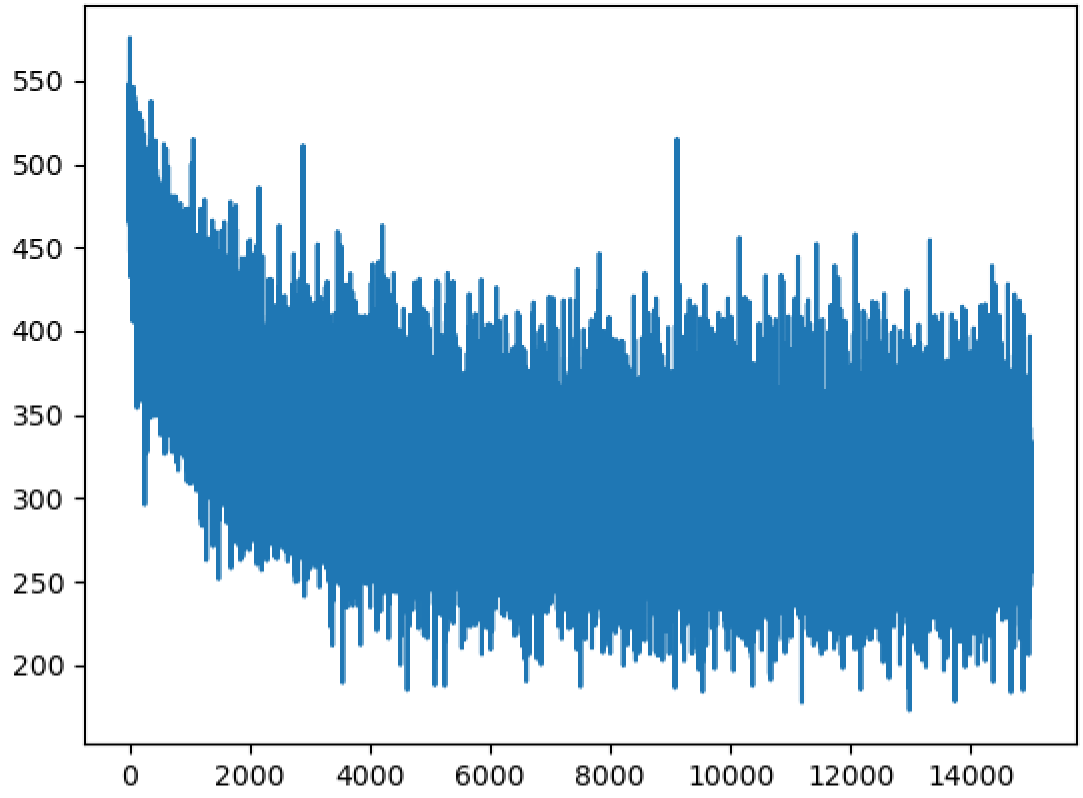
$\delta_t = g(x_t, u_t) + \alpha\, min_u Q(x_{t+1}, u_t) - Q(x_t, u_t)$

$Q(x_t, u_t) = Q(x_t, u_t) + \gamma\delta_t$

I iterate it 100 times for one episode and there are 10000 episodes which creates quite well trained Q table that can be used to keep the pendulum inverted.

Answer of Question 5

The following is the plot between the cost and the episode number.

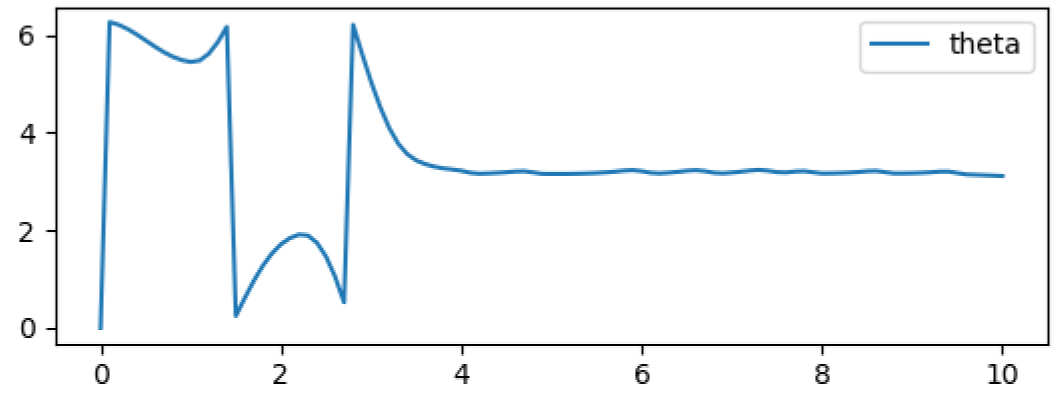


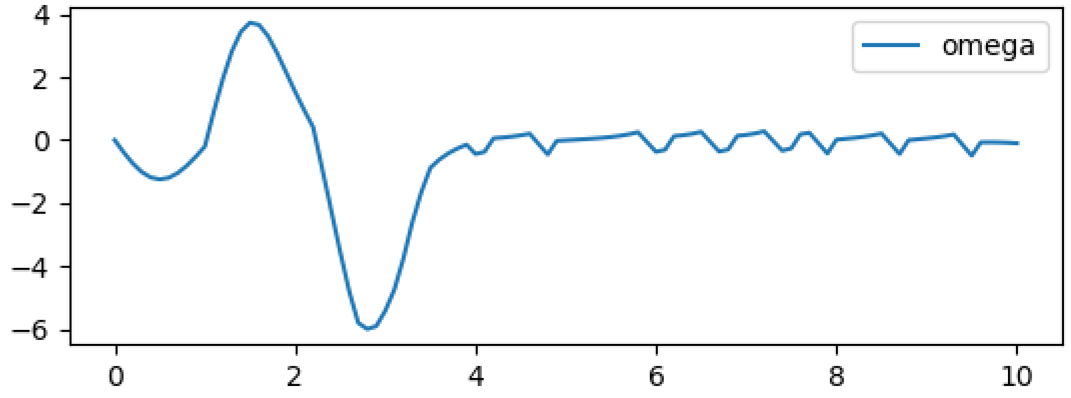
From above plot it seems that the around 7000th to 9000th iteration, the average trend starts coming as a straight line. So, I can say that around 8000 iterations are enough for the model to be trained to make pendulum invert in the given case.

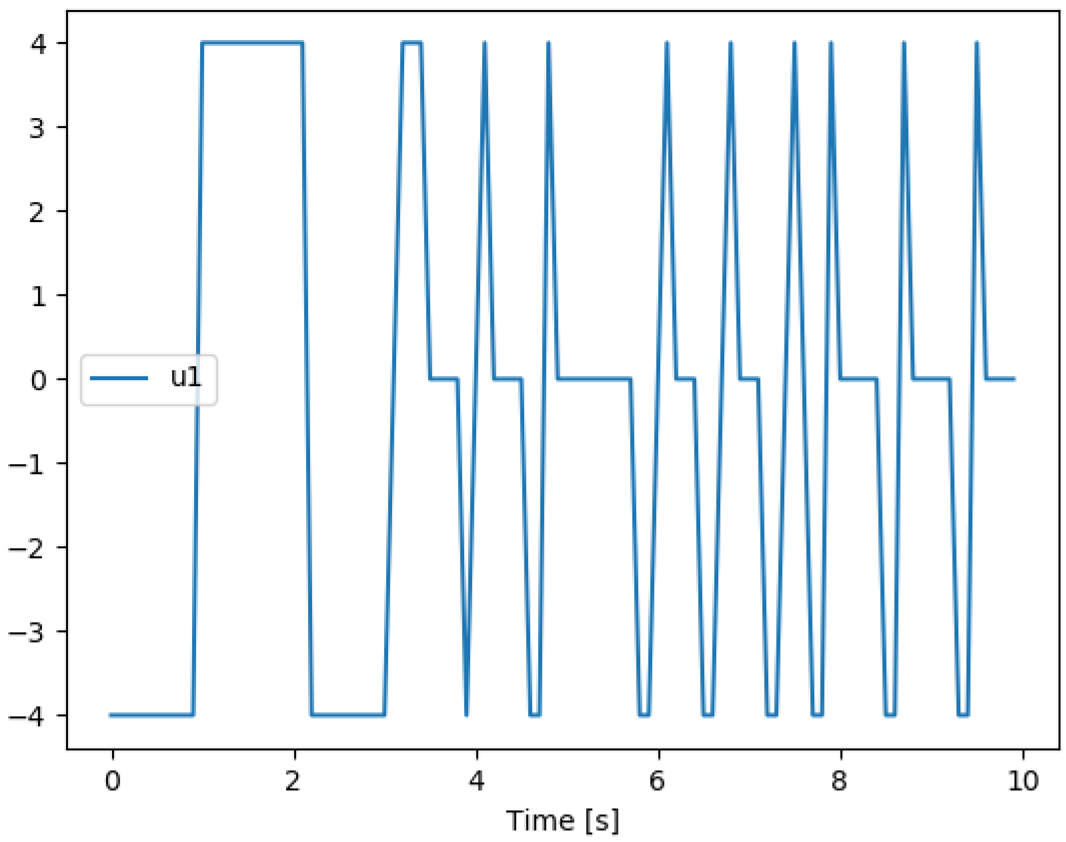
Answer of Question 6

When u = {-4,0,4}

I am seeing these values of theta, omega and u.
I am seeing these plots when the starting condition is angular position = 0 and angular velocity is 0.

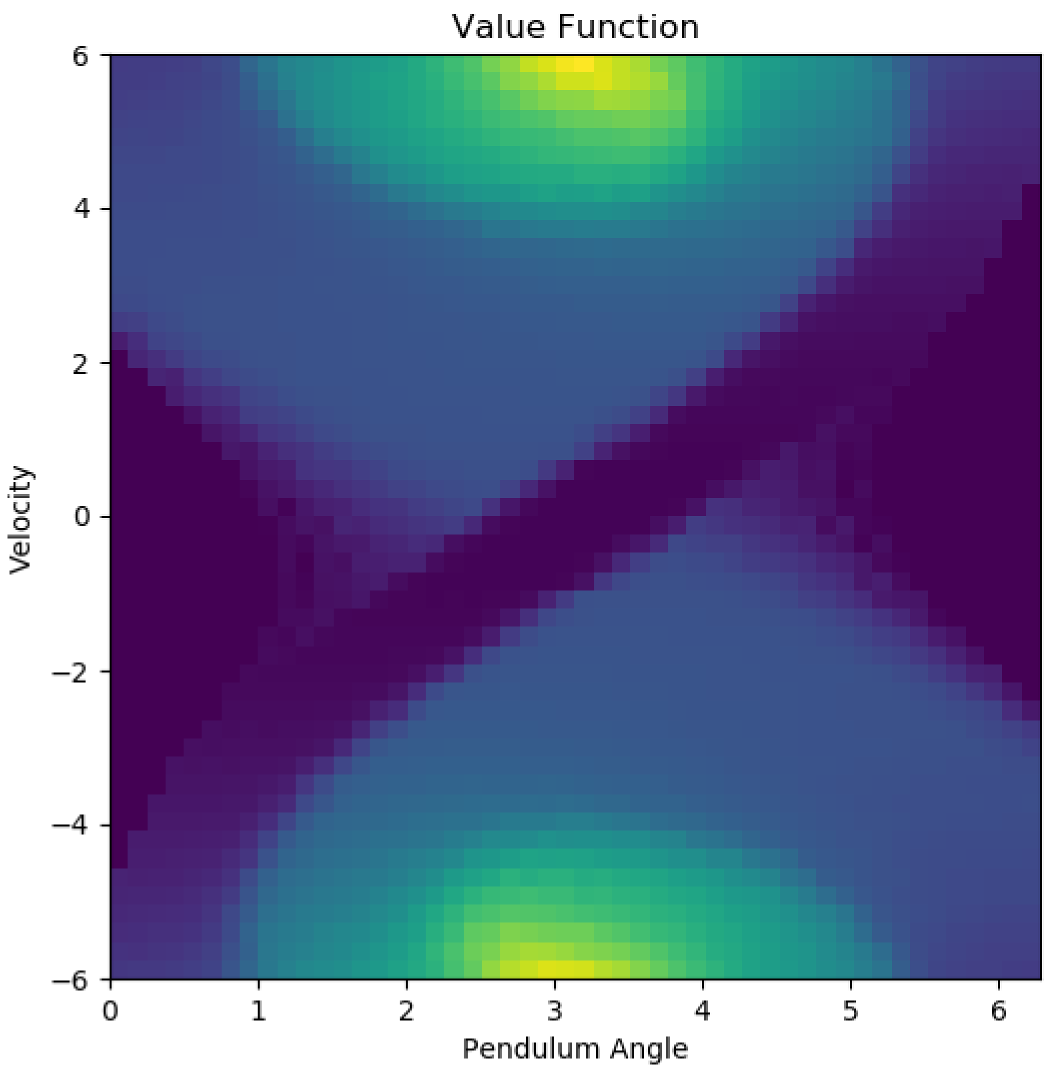

theta
0
2
4
6
0
2
4
6
8
10


omega
4
2
0
−2
−4
−6
0
2
4
6
8
10

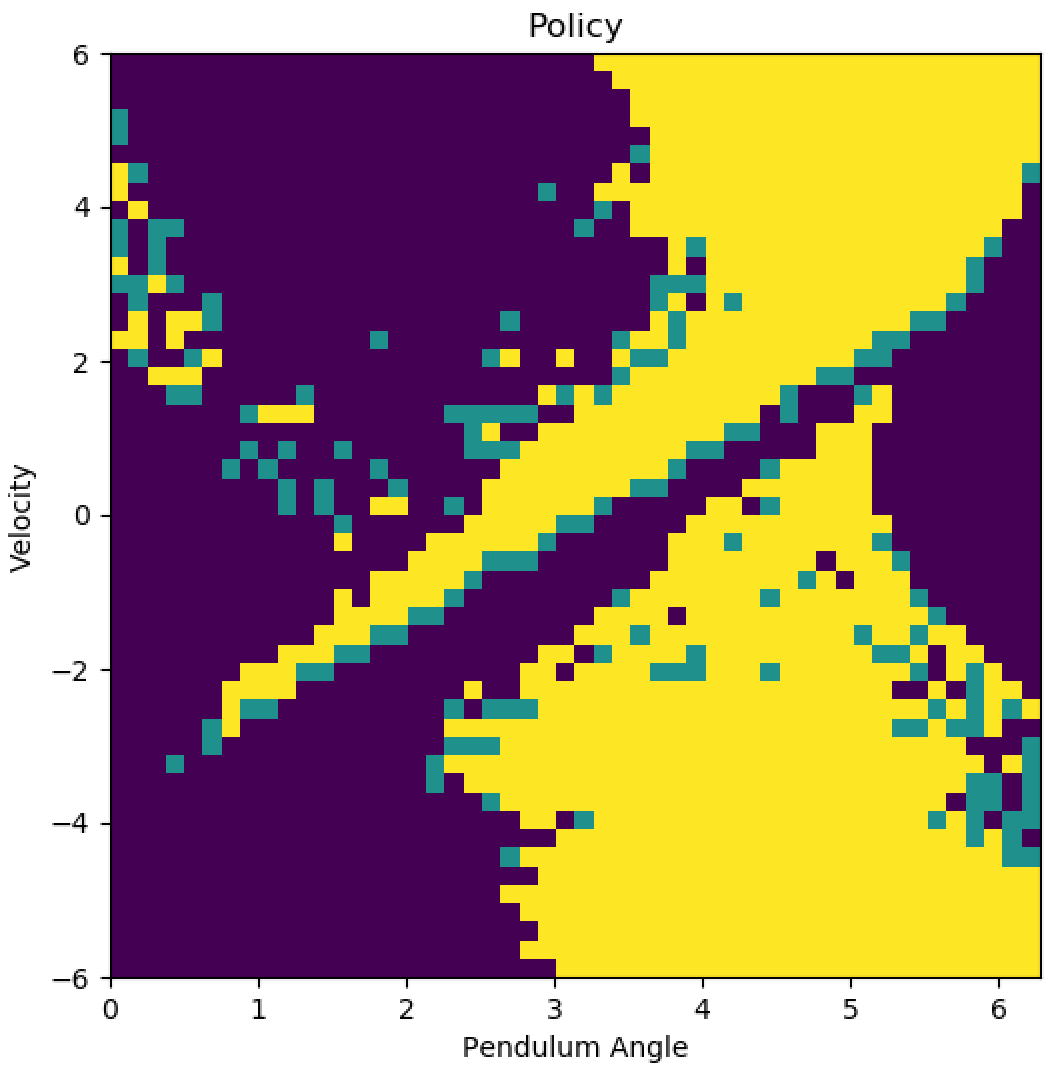


So from the plots and animation around **2 back and forth** of the pendulum are necessary to go from x = [0,0] to fully inverted position.

Answer of Question 7

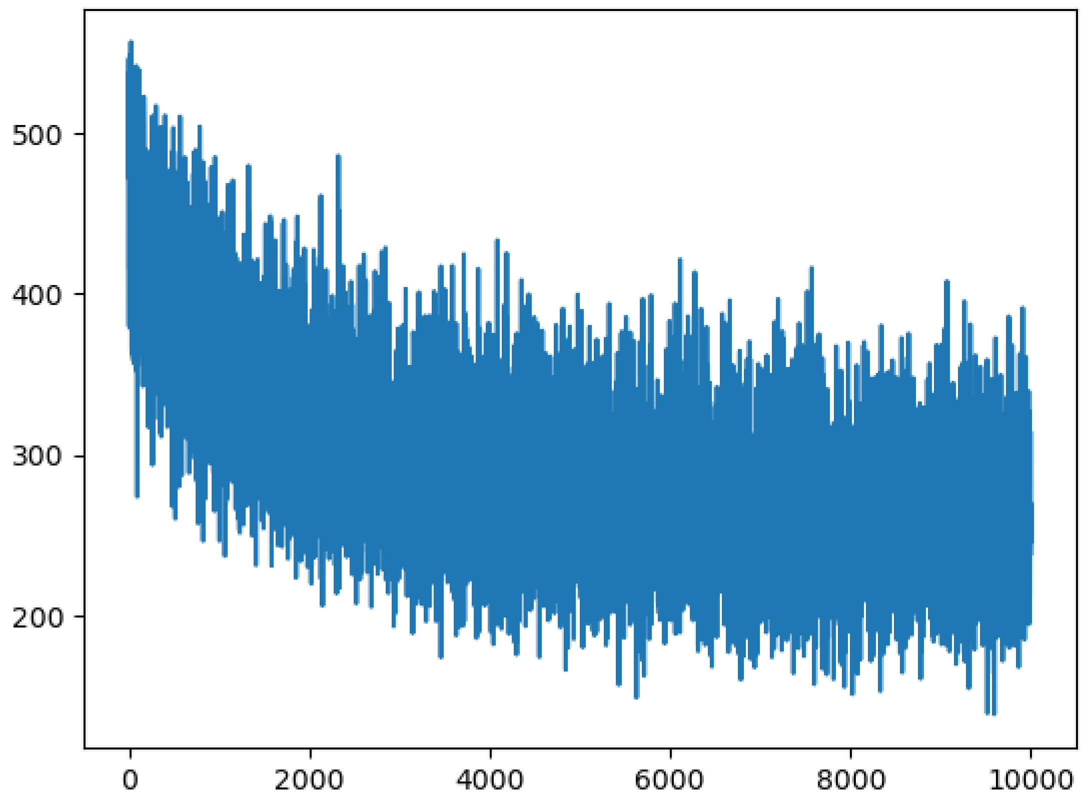


When we look at these plots, it clearly shows that at π and 0 angular velocities it is becoming stable however at higher angular velocities like -6 or +6 the value function has some bigger values of u to bring the speed at zero.



Here green shows u = 0, blue = -u and yellow = +u so the model tries to balance the pendulum that's why yellow at right side and blue at left side.

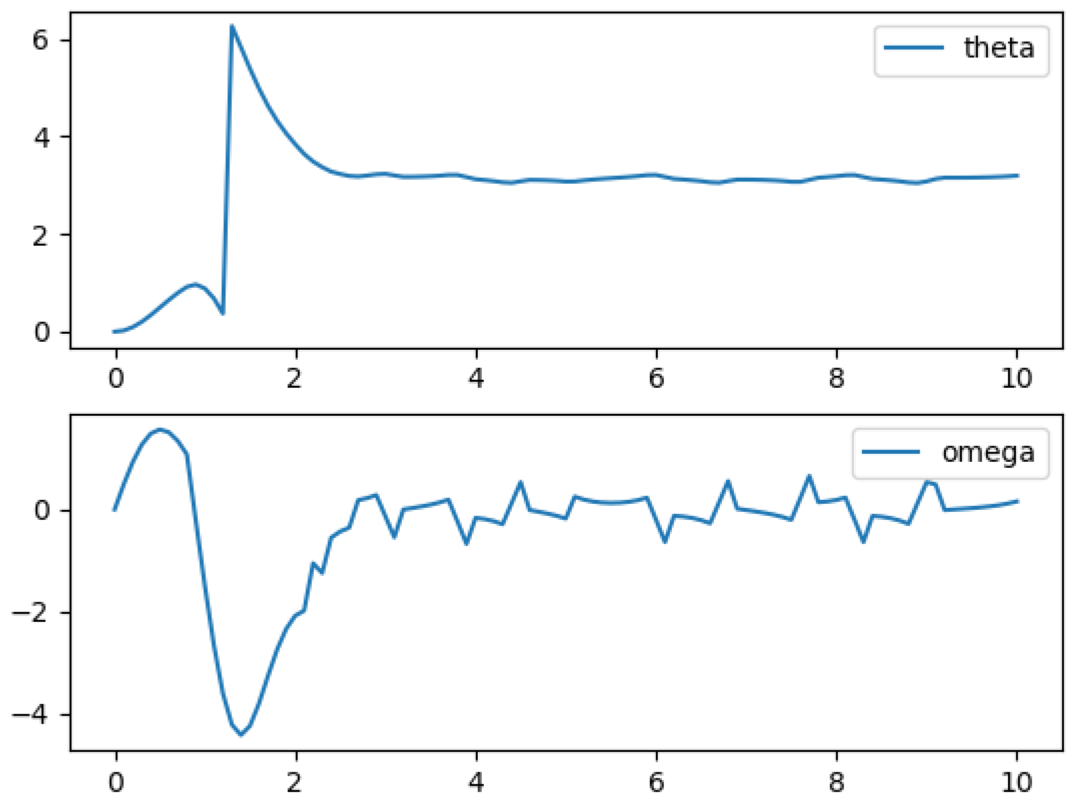
Answer of Question 8 (5 to 7) when u$\epsilon\{-5, 0, 5\}$

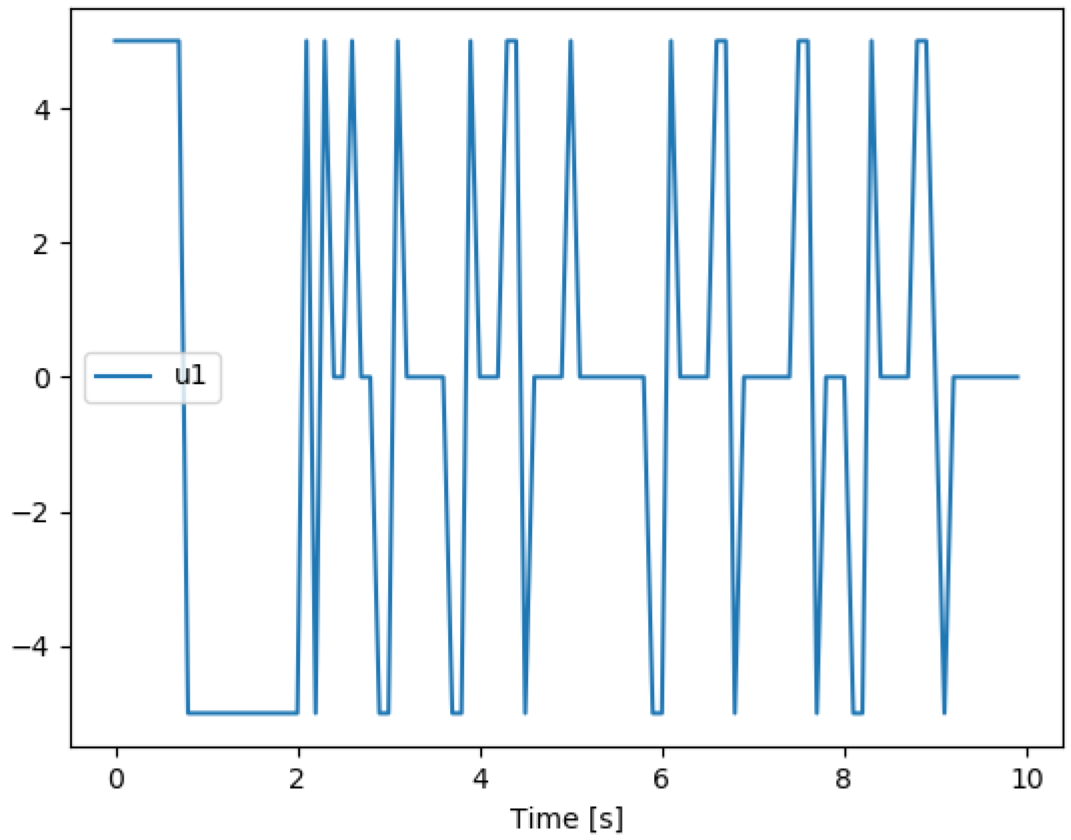


This is when uϵ\{-5,0,5\}
It seems like the number of episodes when the average curve start seeming like an straight line start when 7000th episode completes. So, there is not much difference from the previous cost.

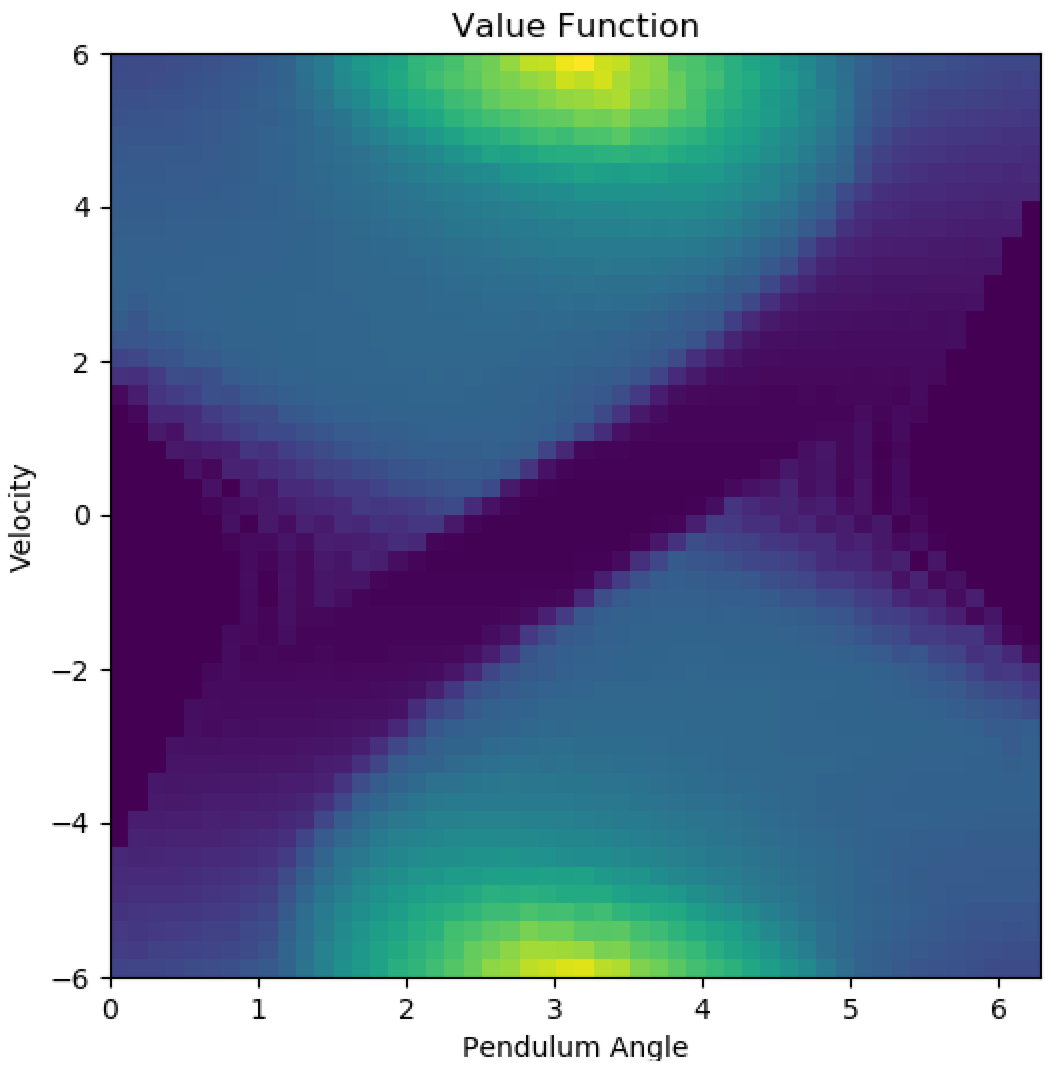
When u = {-5,0,5}

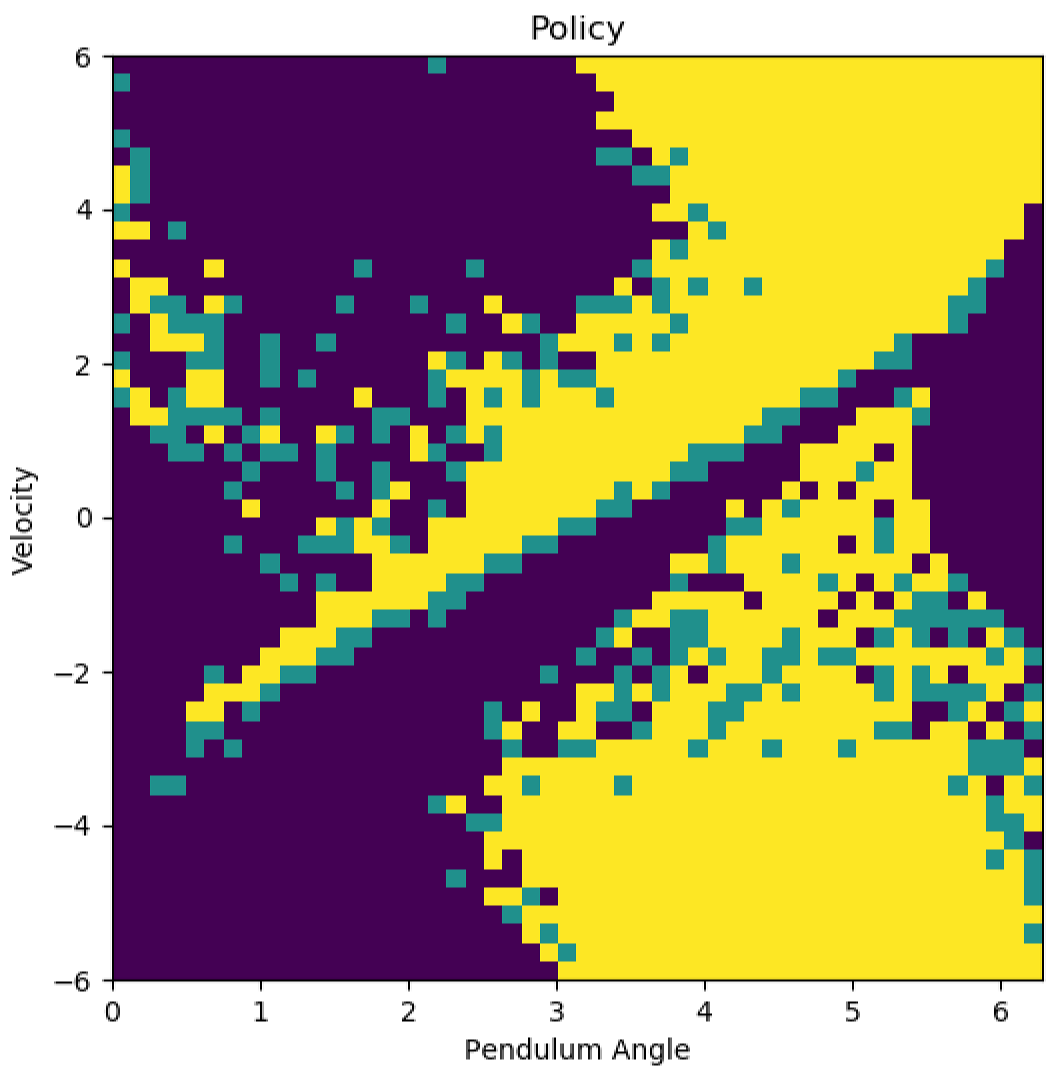
At this instance, I see the following curve of theta and omega


theta
omega
0
2
4
6
8
10
−2
−4



From the above curves, it seems because of more value of u the the amplitude of oscillation after stablization in both theta and omega more in case of u = 5 then u = 4.
There was only one back and forth motion required to make it inverted.


Value Function
Velocity
Pendulum Angle
6
4
2
0
−2
−4
−6
0
1
2
3
4
5
6



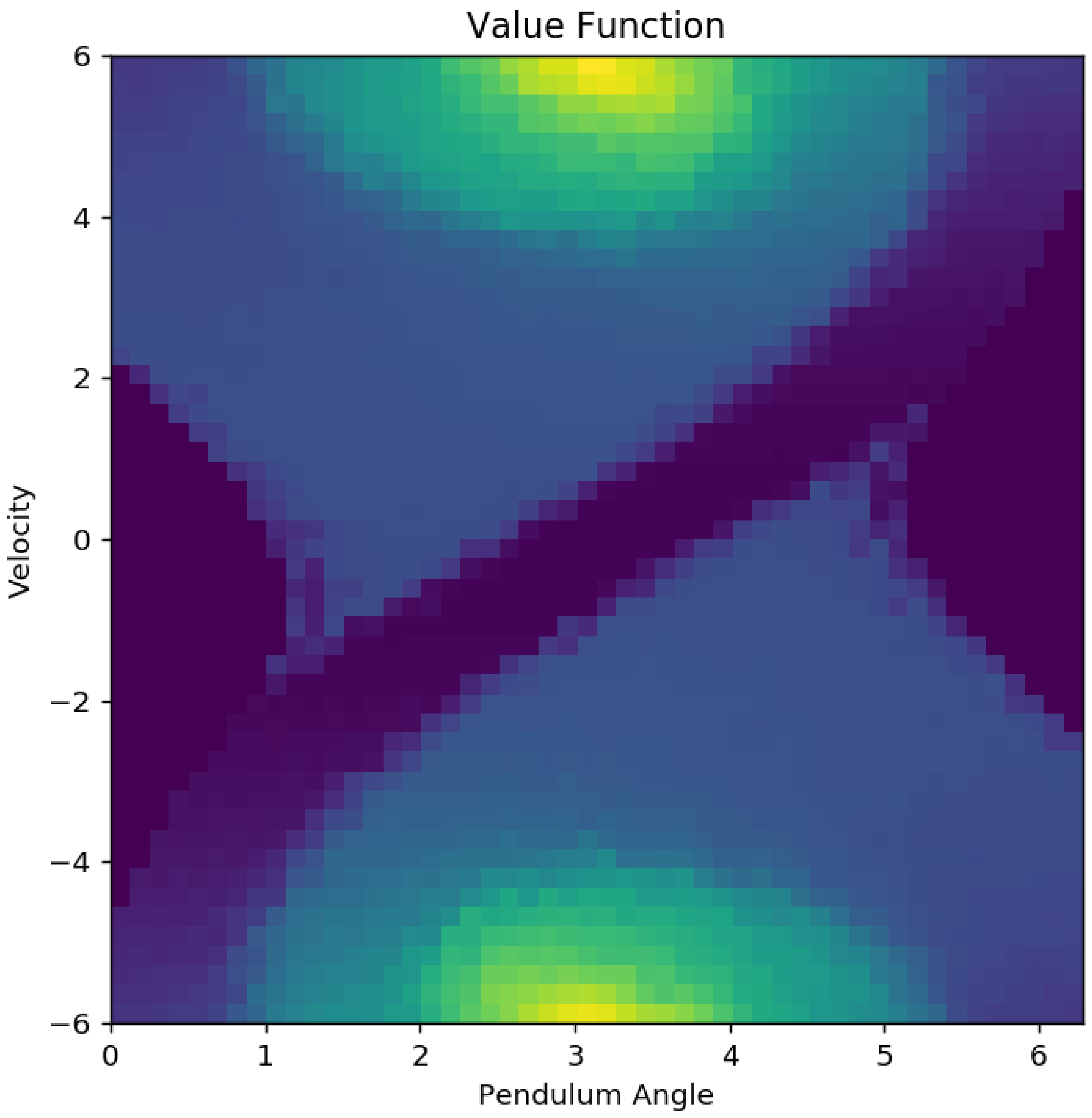
The policy and value function looks quite similar in both of the cases however there are some differences.

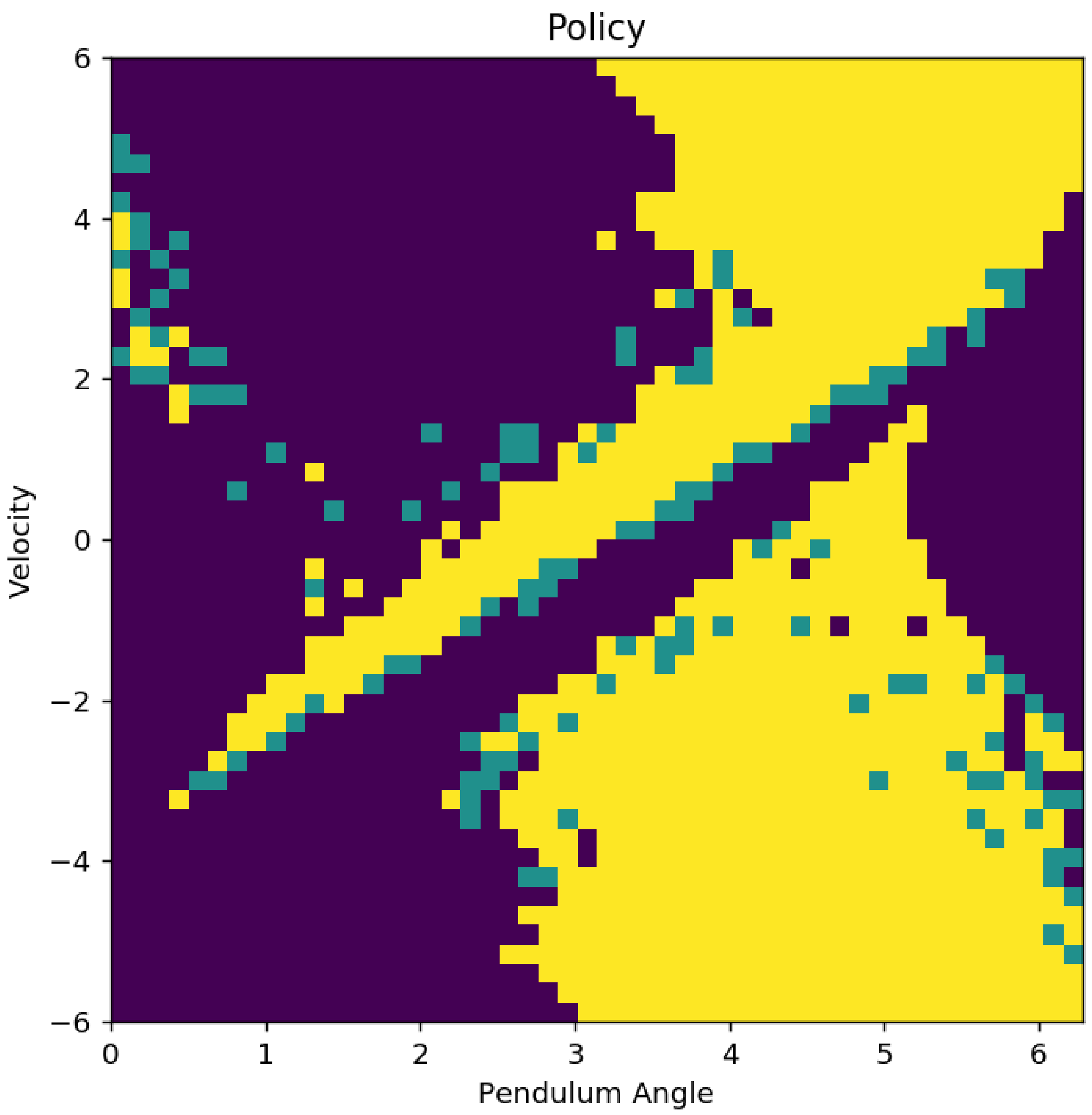
At u = -4 and 4 the blue corners at 0 velocity are more sharp than at u = 5 and -5. Which shows that at u = 5 the value_function has more values than zero.
The same applies for the Policy, it is more smoother at u = 5 and sharper at u = 4.

Answer of question 9

The value of epsilon controls the action if it is random or suggested by q_table. When epsilon is 0.1, it prefers more action form q_table in which all the actions are zero initially which causes the action at q_table[i,j,0] to execute first and cause more time to make the pendulum invert however increased value of epsilon can cause to increase learning time as there are more random actions. So, the value of epsilon should be balanced appropriately.

Learning rate, when the learning rate is quite less like 0.1 the model will learn slowly but optimally. Learning rate contributes towards the updated q value. If it’s more then model learn fastly but there will be oscillations during stabilized conditions because of higher Q values. We can see the results of higher learning rate on Value function and policy plots, With higher values both plots will not give a smooth plot and we observe big square and rectangular brackets.




Policy
6
4
2
0
−2
−4
−6
Velocity
0
1
2
3
4
5
6
Pendulum Angle