Reinforcement Learning and Optimal Control
Project 2 - Q Learning

Yash Patel(yp2285)

1. **Write a function get_cost(x,u) that returns the current cost g(x,u) as a function of the current state and control.**
    - Check Code

2. **What is the dimension of the Q-table that you will need to implement (as a numpy array)? Why?**
    - Q Table dimensions - (50, 50, 3).
    - The Q-table has q value assigned to each state and action pairs. In this case, 50 discretized states for θ and 50 for ω, and 3 values for the action u. There exists a Q value for each state-action pair. In our case we have 50 theta values (θ), 50 omega values(ω) and 3 action values (u).

3. **How can you compute the optimal policy from the Q-table? And the optimal value function? Write a function "'get policy and value function(q table)"' that computes both given a Q-table.**
    - To value function, we need to compute the min values from the q table for all the states and action pairs. Therefore I perform indexing through the entire Q table and compute the minimum to calculate the value function.
    - To compute the optimal policy, the indexing operation on the Q table is performed such that it returns the indices that have the minimum value, and these indices are used to get the actions hence giving the optimal policy.

4. **Write a function q_learning(q_table) that implements the tabular Q-learning algorithm (use episodes of 100 timesteps and an epsilon greedy policy with ϵ=0.1). The function should get as an input an initial Q-table and return a learned Q-table of similar size. Use the function get_next_state from the pendulum package to generate the episode . During learning, store the cost per episode to track learning progress.**
    - Check Code

5. **How many episodes (approximately) does it take for Q-learning to learn how to invert the pendulum when u∈{−4,0,4}? (use a learning rate of 0.1). Show the learning progress in a plot.**

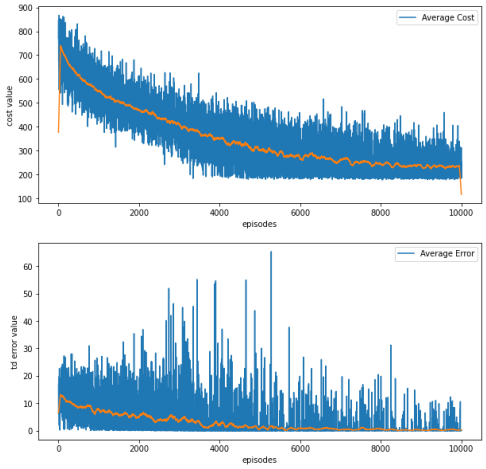

Fig 1: Case {-4,0,4} Learning Progress

- The results in the above fig 1 shows that the algorithm was able to achieve this goal and converge after approximately 8000 episodes.The cost and TD error after 7500 episodes indicates that the algorithm had learned an effective policy and was no longer making significant progress.

6. **Using the simulate / animate functions (cf. below) how many back and forth of the pendulum are necessary to go from x=[0,0] to the fully inverted position? Plot the time evolution of θ and ω.**
    - The pendulum takes around 2 back-and-forth motions when starting from the state (θ = 0, ω = 0) to reach the target position.

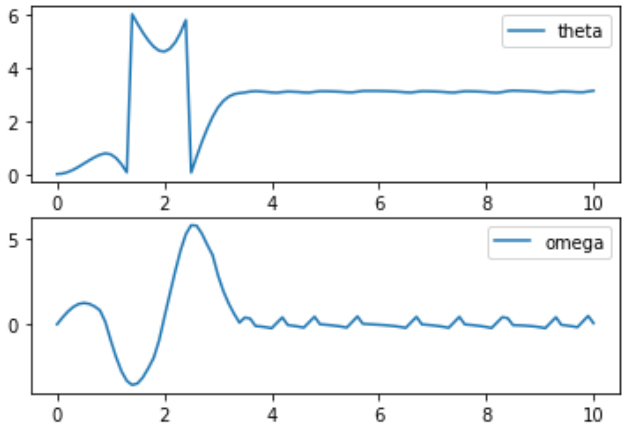

Fig 2: Case {-4,0,4}- θ, ω plot

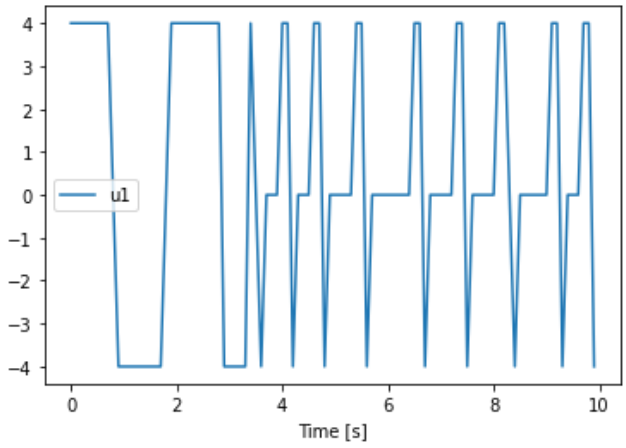

Fig 3: Case {-4,0,4}- actions plot

7. **Plot the found policy and value function as 2D images.**

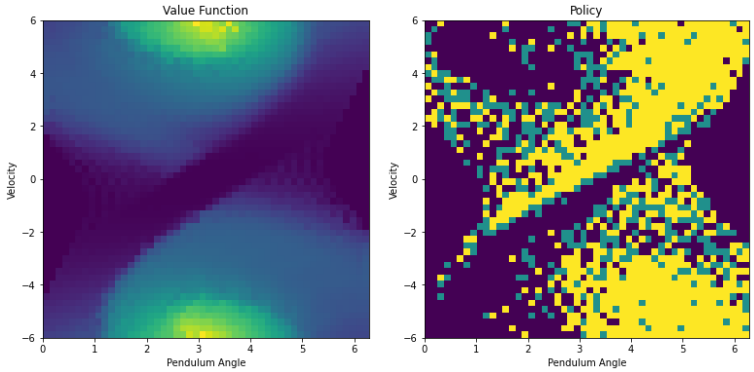

Fig 4: Case {-4,0,4} Value Function and Optimal Policy

8. **Answer questions 5 to 7 when using u∈{−5,0,5} What quantitative differences do you see between the computed policies in 5. and 8.? Can you explain?**

- The plot for the states is shown in fig 5. The fig 6 represents the actions plot, and fig 7 represents the optimal policy and value function.
- In this case we see that the pendulum only made one swing to reach the desired target position. In the above case the pendulum had to make 2 back and forth motions to achieve the target position.

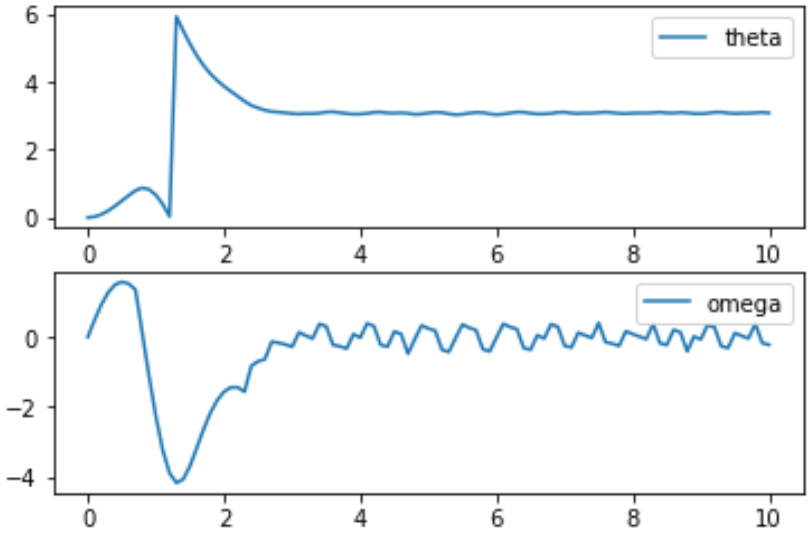

Fig 5:Case {-5,0,5}- θ, ω plot

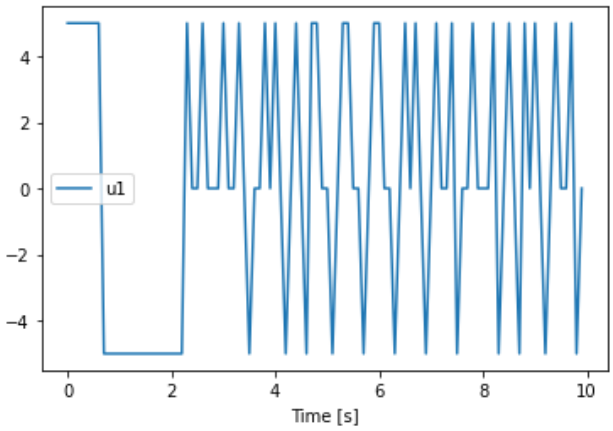

Fig 6: Case {-5,0,5}- actions plot

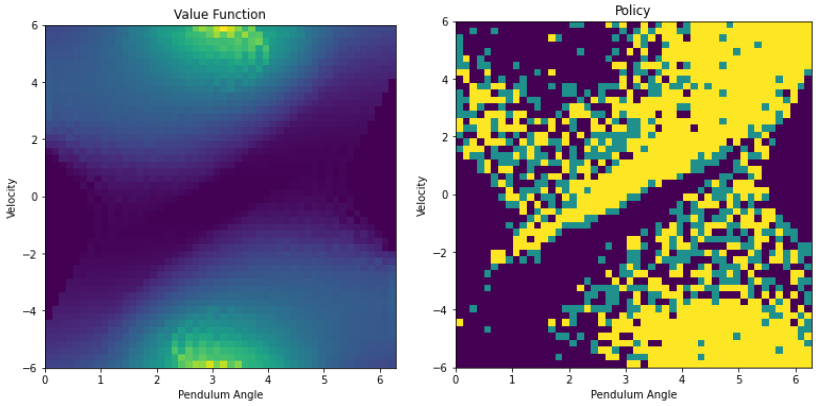

Fig 7: Case {-5,0,5} Value Function and Optimal Policy

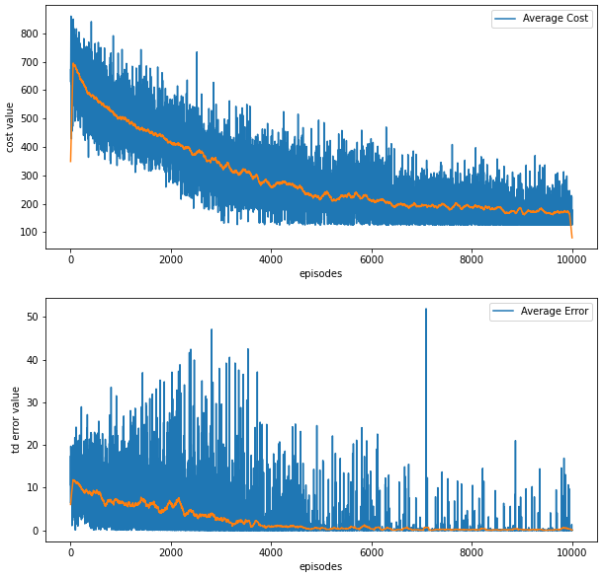

Fig 8: Case {-5,0,5} Learning Progress

9. How is learning affected when changing ϵ and the learning rate? Why?

- Learning rate is a hyperparameter that has to be tuned. Higher the learning rate may lead to faster learning but the agent can become unstable and diverge from the optimal solution. And if the learning rate is small the learning will be slow making the agent less sensitive to new experiences and maintaining the current behavior for longer steps. Therefore it is important to have a balanced learning rate or else it can affect the learning and performance of the agent.
- The epsilon ϵ addresses the exploration-exploitation tradeoff. If the ϵ is high it means the agent is exploring the environment. But this is not good for a long run since the agent won't be able to find an optimal policy if the agent just keeps on exploring. To benefit, the agent should also exploit at certain stages to maximize its reward , and exploit it to learn an optimal policy. If the ϵ is low, the agent will directly exploit the reward which will limit the exploring capability hence leading to a sub optimal policy or a bad policy. Therefore the value of ϵ should be such that it takes into account exploration-exploitation tradeoff.