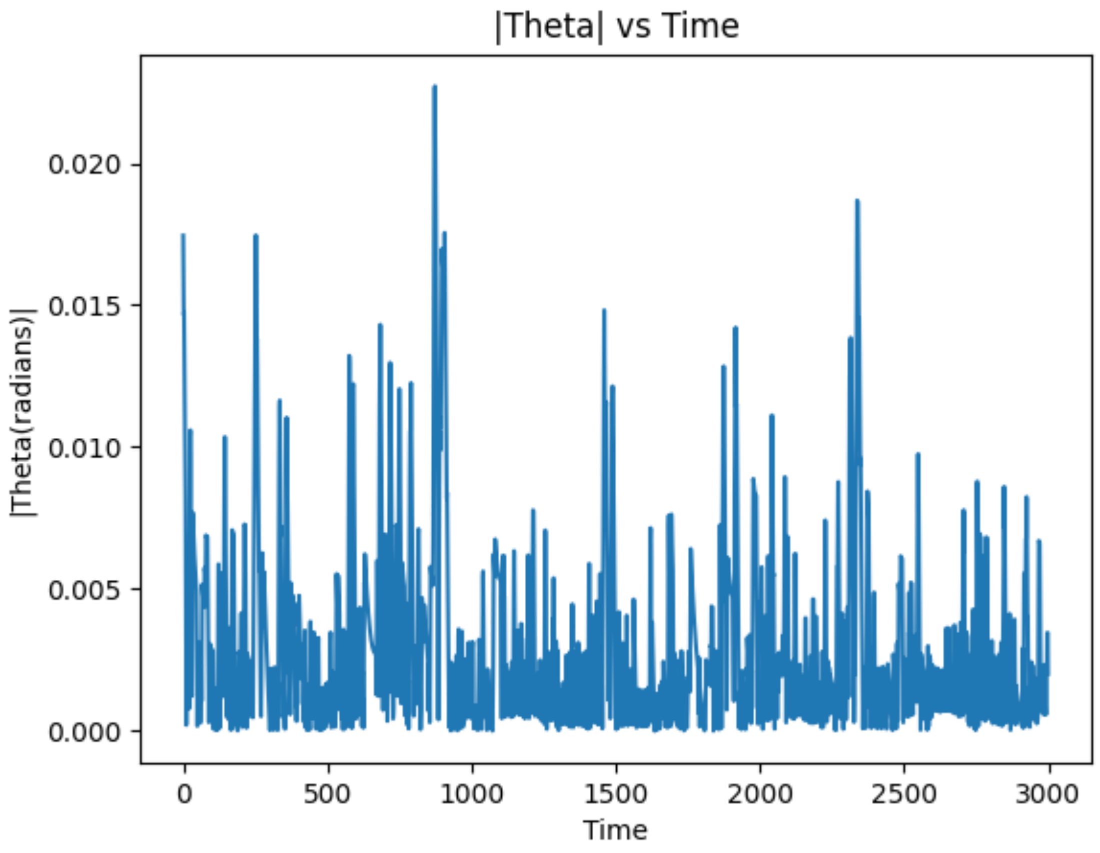
Ben Hepditch – COMP 417 Assignment 3

Part A)

The pole was stabilized at episode 1100. Using the reward function provided in the code, I was unable to reach a satisfactory level of stability without running several thousands of training episode. Alternatively, I used a discontinuous reward function.[1]



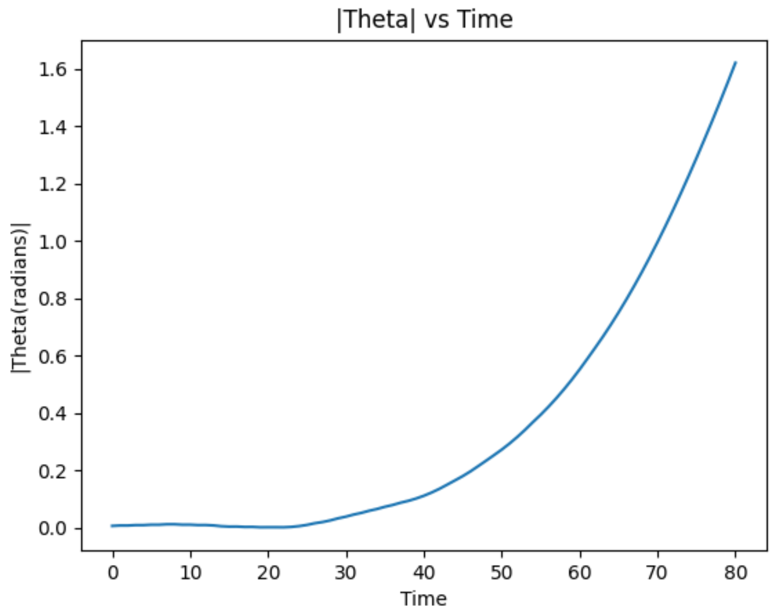
$$R(x) = \begin{cases} 1 & x \in [r-1, r-0.1] \cup [r+0.1, r+1] \\ 5 & x \in (r-0.1, r+0.1) \\ 0 & \text{Otherwise} \end{cases}$$

[1] Mukherjee, A. (2021). *A Comparison of Reward Functions in Q-Learning Applied to a Cart Position Problem*. Arxiv.org. https://arxiv.org/pdf/2105.11617.pdf

The reward function I chose to implement is similar to the above from , with the difference being that in their function, *r* refers to the distance of the cart from zero, whereas in my implementation, r refers to the number of radians between the pendulum and the cart. Additionally, I modified the learning rate to be 0.05 instead of the default 0.001.

Part B)

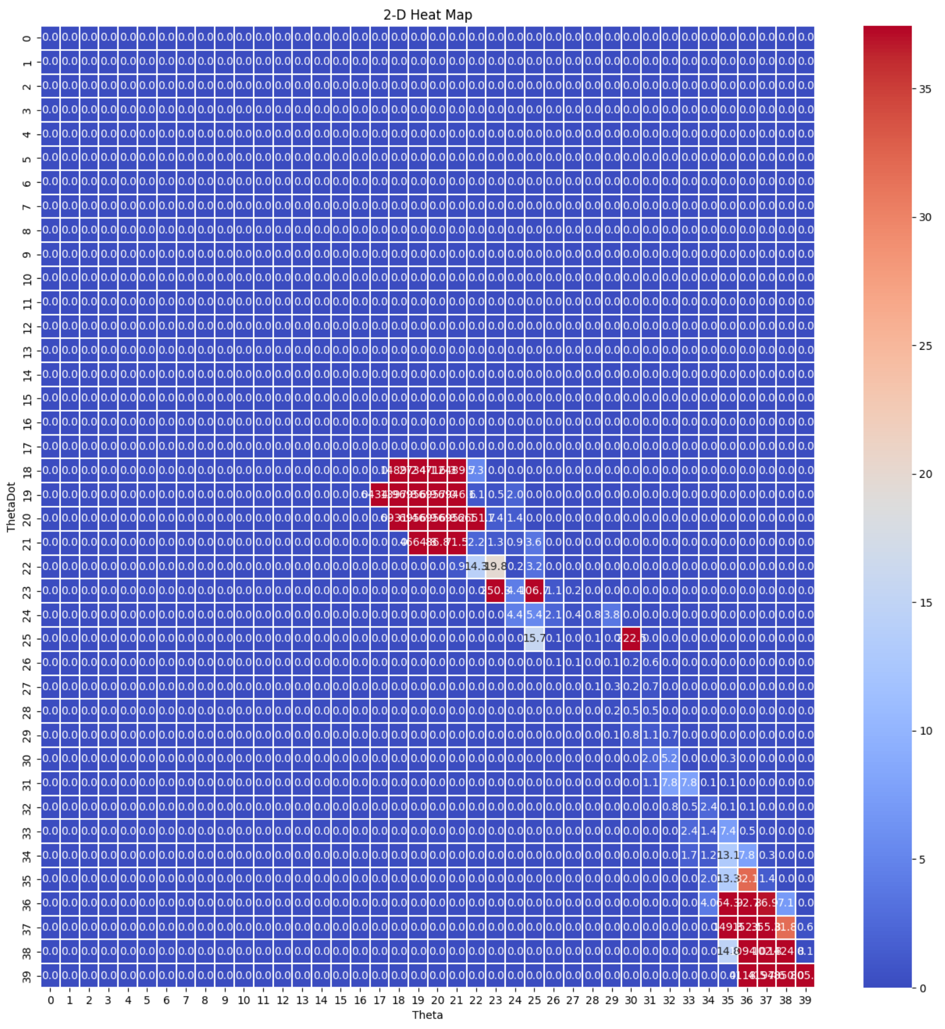
Diminishing the threshold for introducing random actions causes the pole to get stuck at local minima for too long. As a result, the pole never properly stabilized because not enough variability was introduced, I hypothesize that if thousands of more training episodes are used with the same rate of randomness, the cart would eventually converge on a more stable value of theta.



Part C)

By default, all Q-values are initialized to zero, which is also the action that corresponds to "move left". When all values of the matrix are equal, *numpy.argmax* will return the first index of the matrix, which is zero. This results in the cart having a tendency to shift left any time a state hasn't been explored. Since the Q-values are compounded over the duration of the trials, the cart maintains its tendency to move left when it's in a state, unless it is drastically un-rewarding to do so. As a solution, the arrays could be filled with random numbers very close to zero, or, a value other than zero could be used to represent "move left".

Part D)

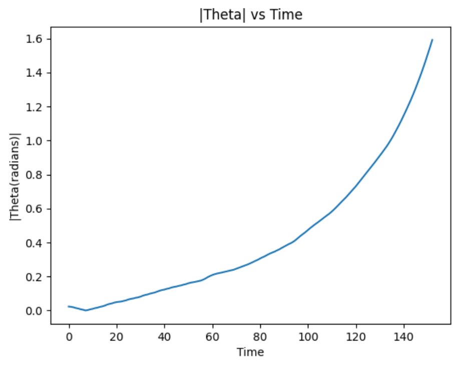


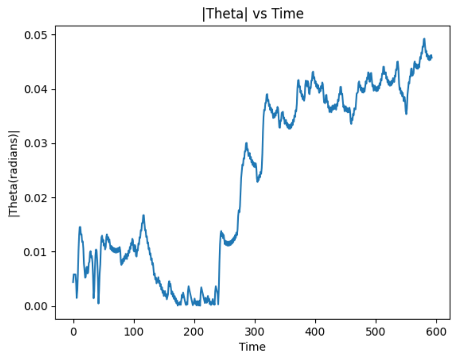
Part E)

To test the impacts of modifying the various hyperparameters, I performed the following experiments:

1. Learning Rate

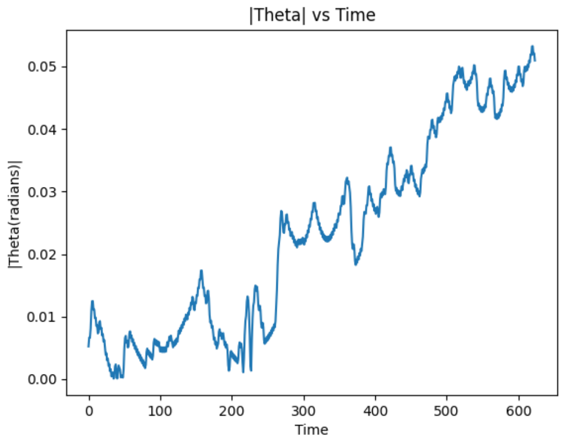
   Initially, I used a learning rate of 0.05 to best replicate the outcomes of the study from which I obtained the discontinuous reward function. I found that by significantly increasing the learning rate (up to 0.75 to be precise), I was able to achieve relative stability throughout the duration of each trial. However, the performance of the cart was far more inconsistent from trial-to-trial. The following charts are of two trials which were 20 episodes apart, but had drastically different performances. They are of episodes 819 and 839 respectively.

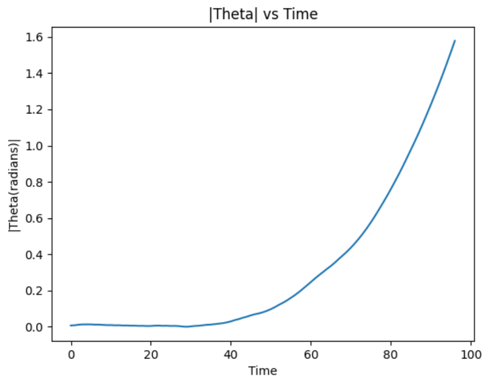




2. Discount factor (gamma)

When I used a lower discount rate (gamma), specifically of 0.8, I achieved much slower progress towards stability than I did using one of 0.99. For example, this was the theta vs time graph at episode 100 with 0.99, compared to the theta vs time of the same number of episodes, but gamma of 0.8 (left to right).


|Theta| vs Time
|Theta(radians)|
Time
0.05
0.04
0.03
0.02
0.01
0.00
0
100
200
300
400
500
600


|Theta| vs Time
|Theta(radians)|
Time
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100