Luke Davidson
CS 5180
Ex4

1.) a.) If every-visit MC was to be used instead of first-visit, the results would be extremely similar because of the way the states, rewards and episodes are built. Every episode terminates at the end of a game, and the states are essentially reset. The states are defined as the sum of the players two cards (12-21), and the dealers showing card (ace-10). The action set is stay or hit. If a player were to hit, the most likely resulting state would be to increase their sum, with the only exception being an ace switching between the values of 1 and 11. It is highly unlikely for a player to decrease their sum of cards by hitting (the ace situation), therefore it is very unlikely to return to the same state twice. In other words, every time a state is entered in this scenario, it will likely be the first and only time it will be entered, and therefore first-visit MC will lead to the same results as every-visit MC since they will essentially be the same thing.

1b.)

Luke Davidson
CS 5180
Ex 4

1.) b) terminal = 1 - p

for 10 steps, return = 10, the simulation must have gone

step #	0	1	2	3	4	5	6	7	8	9	10
state	nt	nt	nt	nt	nt	nt	nt	nt	nt	nt	t
action	p	p	p	p	p	p	p	p	p	1-p	
state	~~—~~	nt	nt	nt	nt	nt	nt	nt	nt	nt	
reward		+1	+1	+1	+1	+1	+1	+1	+1	+1	+1

episode = $S_0, A_0, R_1, S_1, A_1, R_2 \ldots, S_9, A_9, R_{10}$, terminate

```
G = 0
for t = 9 → 0:
    G = γG + R_{t+1}
    if S_t in [S_0 → S_{t-1}]:
        break
    else
        Returns(S_t).append(G)
        V(S_t) = avg(Returns(S_t))
```

following the above build, G will be calculated as

step #	0	1	2	3	4	5	6	7	8	9	10
G	10	9	8	7	6	5	4	3	2	1	—

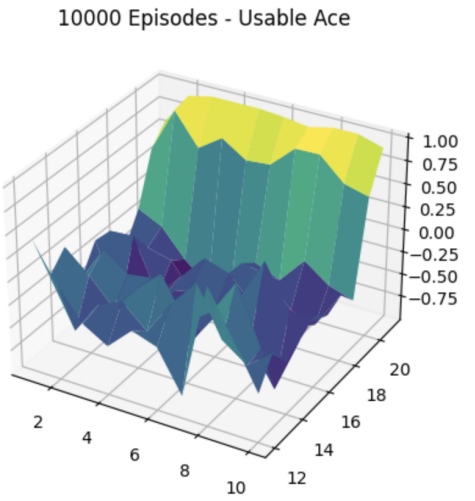
so for first visit (step 0),

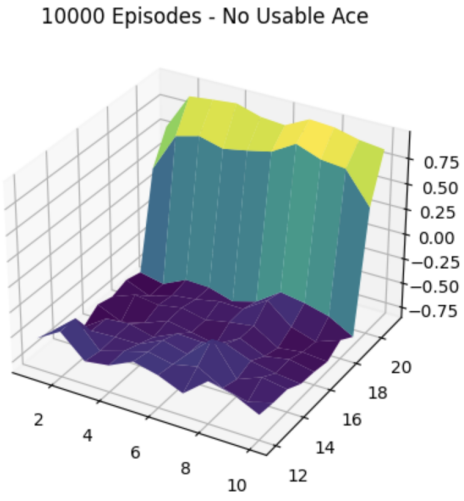
$$V(nt) = \frac{10}{1} = 10$$

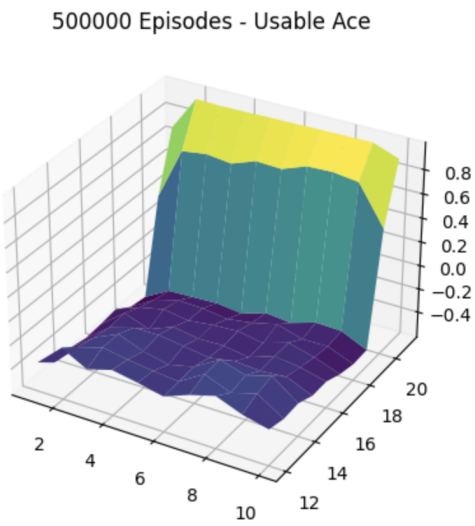
for every first visit, there will be no if S_t statement, so

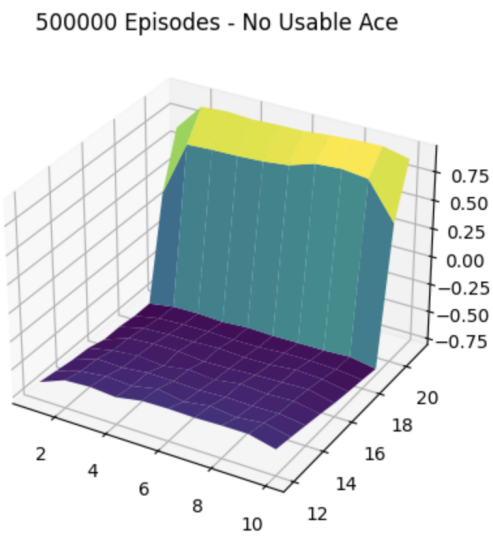
$$V(nt) = \frac{1+2+3+4+5+6+7+8+9+10}{10} = 5.5$$

2.) a.) Plots found for part a, reproduction Fig 5.1 are shown below:



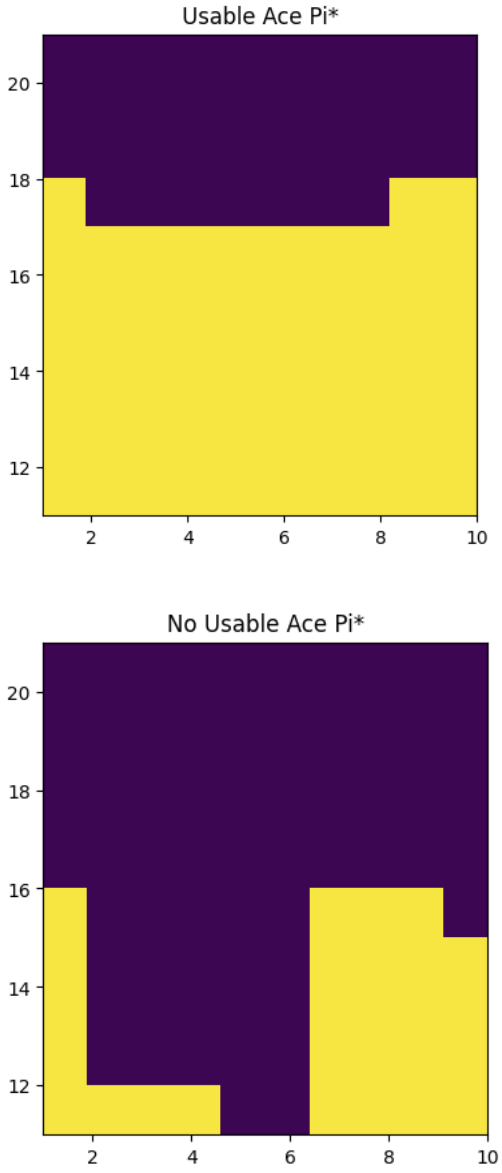




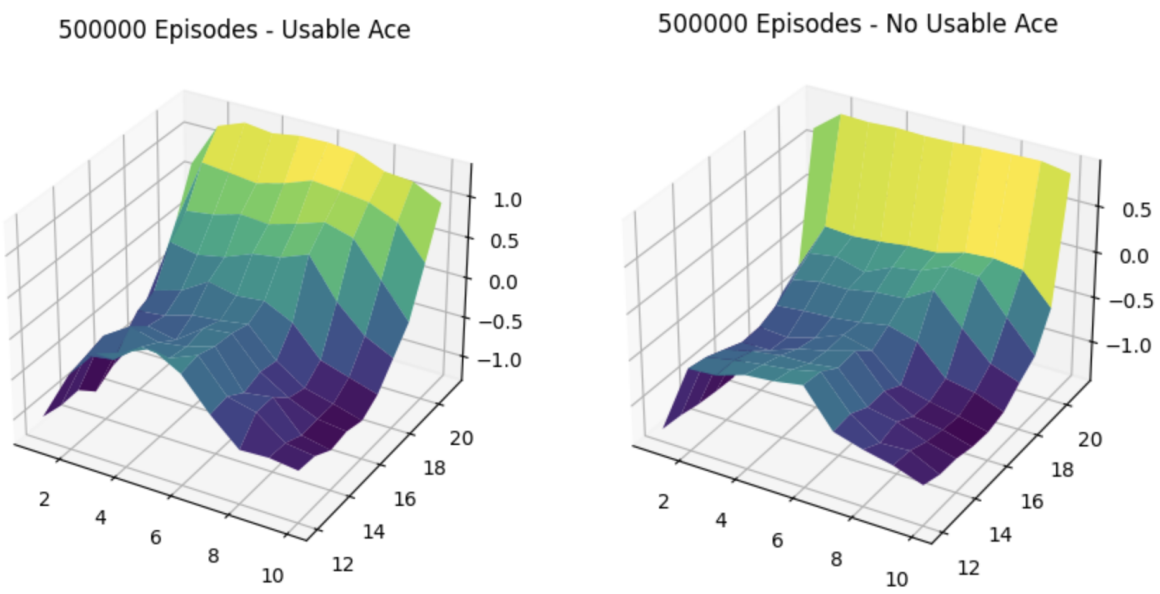


As you can see, after 10,000 episodes the value functions are pretty inconsistent compared to those of 500,000 episodes, especially the Usable Ace condition (less state occurrences). This shows the same results as the textbook.

2b.)

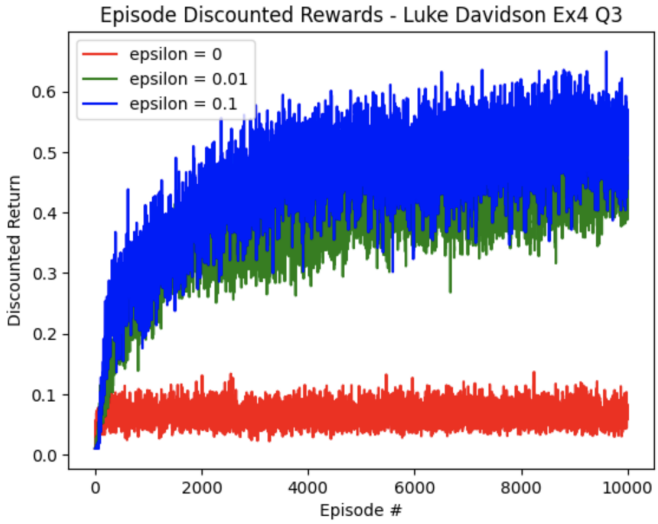


Optimal policies for the Ace and No Ace conditions. Yellow = hit, Purple = stay, y-axis = sum of player cards, x-axis = dealer showing card. These match those of the textbook.

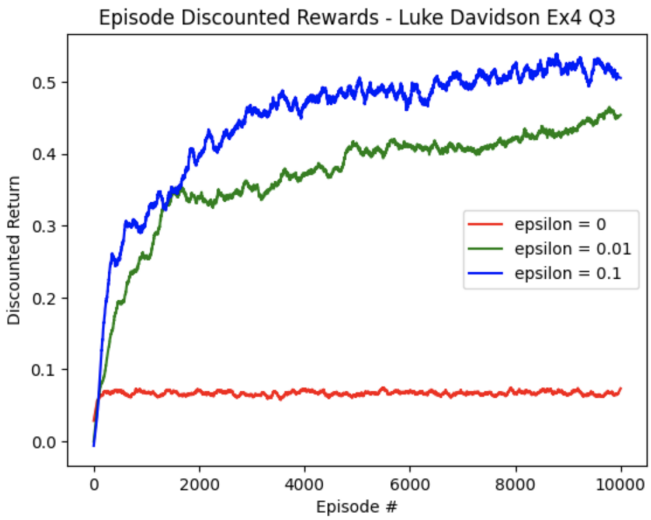


Optimal values for the Ace and No Ace situations, matching those of the textbook

3.) a.) Below is the graph I received when averaging episode discount rewards from 10 trials, each 10,000 episodes, with epsilon values of 0, 0.01, and 0.1.



As you can see, the data came out with a lot of noise. To better visualize the data trends, I applied a Savitsky-Golay filter to the data, using a window size of 101 points (episodes) and a polyfit value of 1. The results came out as seen below:



As one can see, all three learning curves came out in similar shapes to those of ex1. With epsilon = 0.1, the policy reaches higher levels of return faster than epsilon = 0.01, however near the end of the graph it is evident that the slope of the blue line (e = 0.1) is less than that of the green line (e = 0.01). This suggests that as the number of episodes increases (in the long run), the epsilon = 0.01 case will reach a higher level of discounted rewards, and therefore a better policy, than that of the epsilon = 0.1 case.

3b.) With epsilon = 0 (red line), the policy acts in a 100% greedy way and does not explore at all. This displays the importance of having exploring starts in Monte-Carlo ES because exploring starts represents exploring. In the limit of infinity episodes, exploring starts guarantees that every state-action pair is exhibited an infinite amount of times. Without exploring starts, the agent will act greedy and will not explore, causing a much lower potential discounted return because it is not learning the extents of the environment nearly as well, which is exactly what is shown above in the case of epsilon = 0.

4.)

$$58 = V_{n+1} = V_n + \frac{W_n}{C_n}\left[G_n - V_n\right] = V_n + \frac{W_n G_n}{C_n} - \frac{W_n V_n}{C_n} = \frac{W_n G_n}{C_n} + V_n\left[1 - \frac{W_n}{C_n}\right]$$

4.) a) $$5.7 = V_n = \frac{\sum_{k=1}^{n-1} W_k G_k}{\sum_{k=1}^{n-1} W_k} \quad \text{so} \quad V_{n+1} = \frac{\sum_{k=1}^{n} W_k G_k}{\sum_{k=1}^{n} W_k}$$

$$= \frac{W_n G_n + \sum_{k=1}^{n-1} W_k G_k}{C_n} = \frac{W_n G_n}{C_n} + \frac{(C_{n-1})\sum_{k=1}^{n-1} W_k G_n}{C_n (C_{n-1})} =$$

$$= V_n$$

$$\text{so} \quad \frac{W_n G_n}{C_n} + \frac{V_n (C_{n-1})}{C_n} = \frac{W_n G_n}{C_n} + \frac{V_n (C_{n-1})}{C_{n-1} + W_n}$$

b/c $C_{n+1} = C_n + W_{n+1}, \implies C_n = C_{n-1} + W_n$

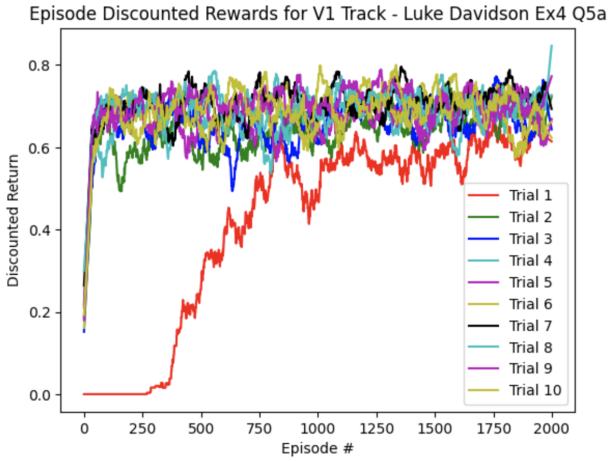
want a 1, so add and subtract W_n to match above

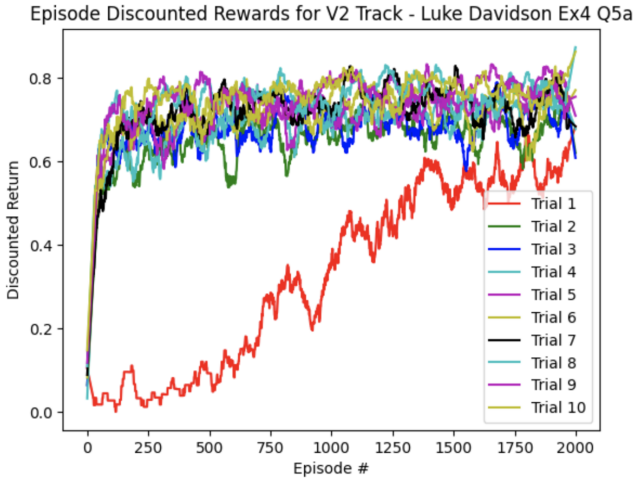
$$= \frac{W_n G_n}{C_n} + V_n\left(\frac{C_{n-1} + W_n}{C_{n-1} + W_n} - \frac{W_n}{C_{n-1} + W_n}\right) = \frac{W_n G_n}{C_n} + V_n\left(1 - \frac{W_n}{C_n}\right)$$

$$\frac{W_n G_n}{C_n} + V_n - \frac{V_n W_n}{C_n} = \boxed{V_n + \frac{W_n}{C_n}\left[G_n - V_n\right] = V_{n+1}}$$

b.) $\pi(A_t | S_t)$ represents the probability of trajectory under policy π, or the probability of selecting Action "a" at State "s" at time step "t". In this case, π represents a greedy policy, meaning it is deterministic at all states. That makes $\pi(A_t | S_t) = 1$ for all states.

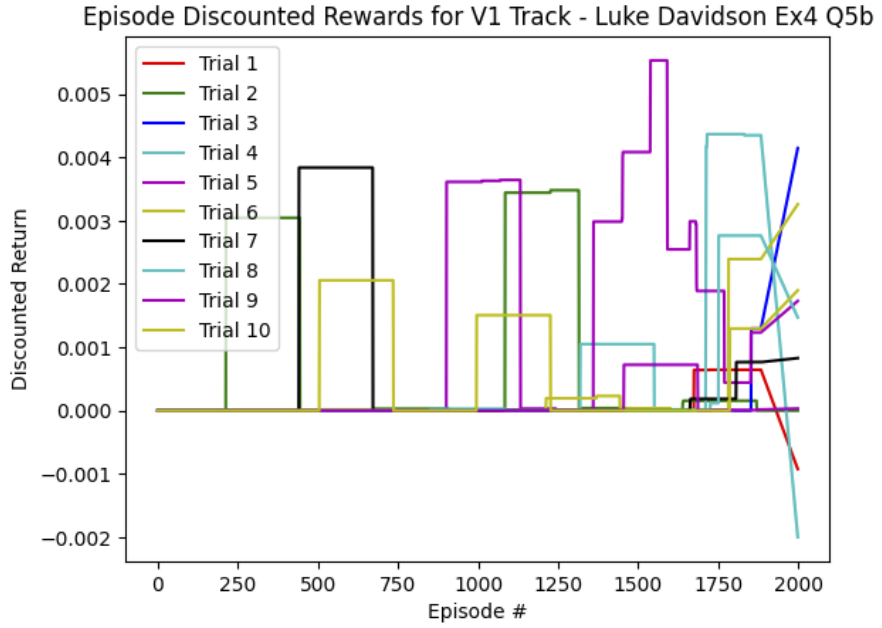
5a.) The results I got for each track after 10 trials, each 2000 steps, are shown below:

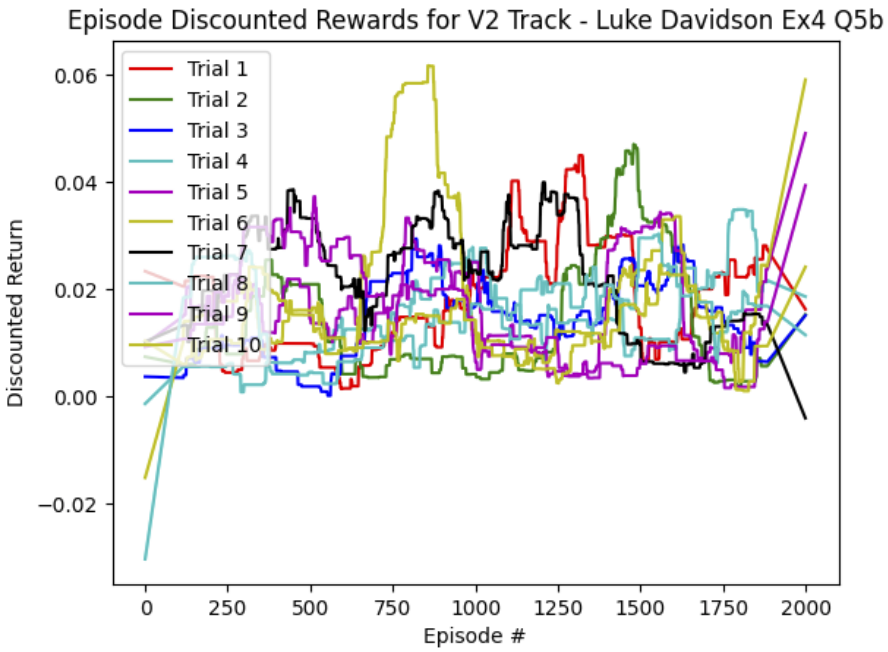




Again, the data came out extremely noisy so these graphs show the smoothed data using a Savitsky-Golay filter again.

5b.) My off policy plots are shown below:





As you can tell, I did not have as much luck with the off policy implementation as the on-policy implementation. Many of my episodes were reaching their max length, leading to a discounted return of near 0. I also noticed there was a lot of variability of results with the initial Q values of

the states. The way the state space and rewards function are set up for this problem makes it an interesting problem to play around with optimistic Q values. Some of the questions I had to think about were what actions do we want to favor at the beginning of the track, and by how much? We want to favor forward at the beginning to achieve the best results quickest, but not by too much because we want that Q value to be accurate once we approach the turn, where we want to slow down and turn right.

5c.) Clearly there are a few differences between the on-policy and off-policy charts. First and foremost, I will admit there is a pretty good chance the off-policy charts are not correct. I believe I had trouble with my exploration policy, b, although I was unable to resolve why my results were so different than that of the on-policy implementation. If you see any obvious errors please let me know (I debugged 2b for a while just to find out that I spelt “usable” wrong in one place). Another possible issue with off-policy is that it takes much longer to converge to the optimal values and performance, compared with the same number of episodes and trials as the on-policy method, due to the higher level of exploration. Since in this case, the reward for hitting the wall is so negative (move all the way back to the start), off-policy results will likely result in worse results as we do not want to explore as much and want to stay on the track.

My on-policy charts between track 1 and track 2 are pretty similar, although my results for track 2 were able to reach a slightly higher level of accuracy than track 1. This may be because there is more room for exploration at the beginning of the track of track 2 (more open space, less walls) for the agent to get a good start at and not have to restart as often. Although, a counterpoint to that is that there is a larger state space in track 2 than track 1, so one would assume it would learn the state space faster if it was smaller given the same number of episodes and trials.