ECE 276B Hw3

s3xu@ucsd.edu

June 2019

1 Problem 1

1.1 (a)

- State space: $x \in \mathcal{X} = \{10000i | i \in \mathcal{N}\}$. $\mathcal{N}$ is the set of natural number.
- Control space: $u \in \mathcal{U} = \{Red, Black\}$
- Motion model

$$x_{i+1} \sim P(\cdot|x_i, u) = \begin{cases} 0.7 & x_{t+1} = 2x_i, u = Red \\ 0.3 & x_{t+1} = 0, u = Red \\ 0.2 & x_{t+1} = 11x_i, u = Black \\ 0.8 & x_{t+1} = 0, u = Black \\ 0 & else \end{cases} \tag{1}$$

- Stage cost: $\ell(x_i, u) = 0$
- Terminal cost: $\mathfrak{q}(x_T) = -x_T$

1.2 (b)

After running simulations, the results are as follows.

Table 1: Crude value and policy

x	V^π	$\bar{\pi}'$ (action + gambling fund)
0	0	-
10	0.0	Red + 10000
20	-21000.0	Red + 10000
30	-30000.0	-
40	-40000.0	-
110	-110000.0	-
120	-120000.0	-
220	-220000.0	-

Table 2: Precise estimate value and policy

x	$\hat{V}^{\pi}$	$\hat{\pi}'$ (action + gambling fund)
0	0	-
10	-18607.594936697122	Red + 10000
20	-26582.278481009137	Red + 10000
30	-30000.0	-
40	-40000.0	-
110	-110000.0	-
120	-120000.0	-
220	-220000.0	-

1.3 (C)

The policies (action + gambling fund) are as follows.

The policies are not the same as original policy. The actions are changed into black rather than red. And in the precise estimation, the gambling fund is raised from 10000 to 20000.

Table 3: Crude value and policy

x	old $\bar{\pi}'$	improved $\bar{\pi}'$
0	-	-
10	Red + 10000	Black + 10000
20	Red + 10000	Black + 20000
30	-	-
40	-	-
110	-	-
120	-	-
220	-	-

Table 4: Precise estimate value and policy

x	old $\hat{\pi}'$	improved $\hat{\pi}'$
0	-	-
10	Red + 10000	Black + 10000
20	Red + 10000	Black + 20000
30	-	-
40	-	-
110	-	-
120	-	-
220	-	-

1.4 (d)

The optimal value and policy are as follows.

Table 5: Values

x	$V^*(x)$	π^* (action + gambling fund)
0	0	-
10	-38181.81818170846	Red + 10000
20	-54545.454545366774	Black + 10000
30	-30000.0	-
40	-40000.0	-
110	-110000.0	-
120	-120000.0	-
220	-220000.0	-

Based on the optimal policy, I should bet \$10000 on red if I currently have \$10000, and bet \$10000 on Black if with \$20000, and stop if I reached the terminal condition (earned more than \$30000 or went broke).

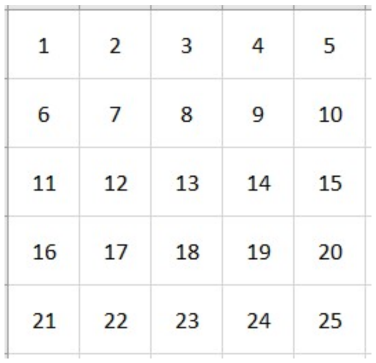
The expected money to walk away with is about \$38181.818.

Celebration? Of course! Let's go to Las Vegas :)

2 Problem 2

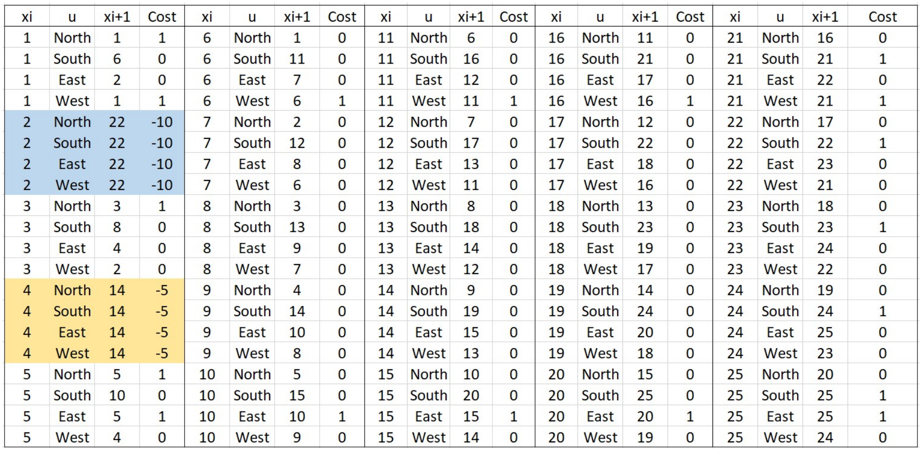
Problem Formulation

We assign following numbers to each grids.



1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

- State space: $x \in \mathcal{X} = \{1, 2, 3, \cdots, 25\}$.
- Control space: $u \in \mathcal{U} = \{North, South, East, West\}$
- Motion model and stage cost (in pre-computed tabular form)



xi	u	xi+1	Cost	xi	u	xi+1	Cost	xi	u	xi+1	Cost	xi	u	xi+1	Cost	xi	u	xi+1	Cost
1	North	1	1	6	North	1	0	11	North	6	0	16	North	11	0	21	North	16	0
1	South	6	0	6	South	11	0	11	South	16	0	16	South	21	0	21	South	21	1
1	East	2	0	6	East	7	0	11	East	12	0	16	East	17	0	21	East	22	0
1	West	1	1	6	West	6	1	11	West	11	1	16	West	16	1	21	West	21	1
2	North	22	-10	7	North	2	0	12	North	7	0	17	North	12	0	22	North	17	0
2	South	22	-10	7	South	12	0	12	South	17	0	17	South	22	0	22	South	22	1
2	East	22	-10	7	East	8	0	12	East	13	0	17	East	18	0	22	East	23	0
2	West	22	-10	7	West	6	0	12	West	11	0	17	West	16	0	22	West	21	0
3	North	3	1	8	North	3	0	13	North	8	0	18	North	13	0	23	North	18	0
3	South	8	0	8	South	13	0	13	South	18	0	18	South	23	0	23	South	23	1
3	East	4	0	8	East	9	0	13	East	14	0	18	East	19	0	23	East	24	0
3	West	2	0	8	West	7	0	13	West	12	0	18	West	17	0	23	West	22	0
4	North	14	-5	9	North	4	0	14	North	9	0	19	North	14	0	24	North	19	0
4	South	14	-5	9	South	14	0	14	South	19	0	19	South	24	0	24	South	24	1
4	East	14	-5	9	East	10	0	14	East	15	0	19	East	20	0	24	East	25	0
4	West	14	-5	9	West	8	0	14	West	13	0	19	West	18	0	24	West	23	0
5	North	5	1	10	North	5	0	15	North	10	0	20	North	15	0	25	North	20	0
5	South	10	0	10	South	15	0	15	South	20	0	20	South	25	0	25	South	25	1
5	East	5	1	10	East	10	1	15	East	15	1	20	East	20	1	25	East	25	1
5	West	4	0	10	West	9	0	15	West	14	0	20	West	19	0	25	West	24	0

- Terminal cost: $\mathfrak{q}(x_T) = 0$

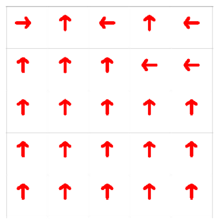
2.1 (a), (b) and (c)

After running simulation, I get the same optimal values and policies for different algorithms as follows.

state	Policy Iteration	Value Iteration	Q-value iteration
1	-21.977	-21.977	-21.977
2	-24.419	-24.419	-24.419
3	-21.977	-21.977	-21.977
4	-19.419	-19.419	-19.419
5	-17.477	-17.477	-17.477
6	-19.780	-19.780	-19.780
7	-21.977	-21.977	-21.977
8	-19.780	-19.780	-19.780
9	-17.802	-17.802	-17.802
10	-16.022	-16.022	-16.022
11	-17.802	-17.802	-17.802
12	-19.780	-19.780	-19.780
13	-17.802	-17.802	-17.802
14	-16.022	-16.022	-16.022
15	-14.419	-14.419	-14.419
16	-16.022	-16.022	-16.022
17	-17.802	-17.802	-17.802
18	-16.022	-16.022	-16.022
19	-14.419	-14.419	-14.419
20	-12.977	-12.977	-12.977
21	-14.419	-14.419	-14.419
22	-16.022	-16.022	-16.022
23	-14.419	-14.419	-14.419
24	-12.977	-12.977	-12.977
25	-11.680	-11.680	-11.680

Here are the visualized policies with $\gamma = 0.9$.

Figure 1: Policy Iteration with gamma=0.9



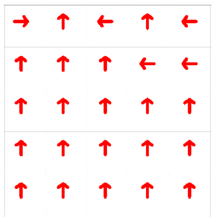


Figure 2: Value Iteration with gamma=0.9

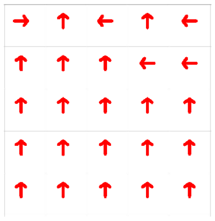


Figure 3: Q-Value Iteration with gamma=0.9

2.2 (d)

Here are the visualized policies with $\gamma = 0.8$. One obvious difference is that the policy at states 9 and 10 changed from West to North. This is because the policy becomes ”near-sighted” due to the value decrease in gamma, in which the policy tends to consider most immediate cost in a greedy manner. So instead of moving to position A(2) to get a lower cost, the system choose to move to a nearer position B(4) which leads to a immediate -5 stage cost.

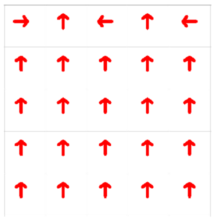


Figure 4: Policy Iteration with gamma=0.8

Figure 5: Value Iteration with gamma=0.8

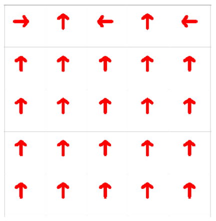
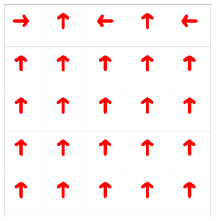


Figure 6: Q-Value Iteration with gamma=0.8



Here are the visualized policies with $\gamma = 0.99$, which makes no difference to the case when $\gamma = 0.9$. This suggests both policies are far-sighted enough. Compared to the case with $\gamma = 0.8$, it chooses not to take immediate stage cost -5 at position 4 but goes further to position 2 for a lower cost of -10.

Figure 7: Policy Iteration with gamma=0.99

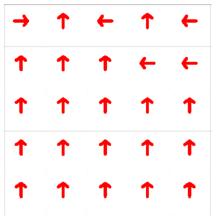


Figure 8: Value Iteration with gamma=0.99

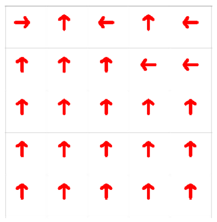
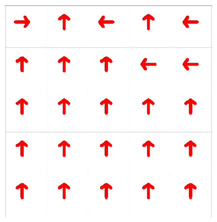


Figure 9: Q-Value Iteration with gamma=0.99



3 Problem 3

3.1 Problem Formulation

- State space: The state space was supposed to be continuous, however, it's descritized into the space as follow

  $$\mathcal{X} = \{(speed, position) | speed \in \{-0.07+0.01n | n \in 0, 1, \cdots, 14\}, position \in \{-1.2+0.1n | n \in 0, 1, \cdots, 18\}$$

  All the states with $position \geq 0.6$ are terminal states which suggest that the car has reached goal.

- Control Space: $\mathcal{A} = \{left, right, lazy\}$, where lazy means the car takes no action.

- Stage cost and Terminal cost:

  $$\ell((speed, position), u) = \begin{cases} 0 & position \geq 0.6(\text{ i.e. terminal cost}) \\ -1 & else(\text{ i.e. stage cost}) \end{cases}$$

- Objective Function: $\pi^*(x) = \underset{u \in \mathcal{U}(x)}{\arg\min} Q^{\pi^*}(x, u)$

3.2 Technical Approach

The problem can be solved using TD policy iteration algorithm. There are two specific types: on-policy (SARSA) and off-policy (Q-learning).

The main idea of model-free algorithm is to approximate $V^{\pi}(x_0)$ from several episodes $\rho_0^{(k)} := x_{0:T}^{(k)}, u_{0:T-1}^{(k)}$ under policy π. Compared to Monte-Carlo algorithm, TD algorithm used bootstrapping, so that the value estimate of state x relies on the value estimate of another state.

The policy evaluation step updates the value estimate $V^{\pi}(x_t)$ towards an estimated long-term cost $\ell(x_t, u_t) + \gamma V^{\pi}(x_{t+1})$:

$$V^{\pi}(x_t) \leftarrow V^{\pi}(x_t) + \alpha\left(\ell(x_t, u_t) + \gamma V^{\pi}(x_{t+1}) - V^{\pi}(x_t)\right)$$

The policy improvement step computing π' based on V^{π} requires access to $\ell(x, u)$ but based on Q^{π} can be done without knowing $\ell(x, u)$.

$$\pi'(x) = \underset{u \in \mathcal{U}(x)}{\arg\min} Q^{\pi}(x, u)$$

Specially, we used ϵ-greedy method to select control as bellow.

$$\pi(u|x) \leftarrow \begin{cases} 1 - \epsilon + \frac{\epsilon}{|\mathcal{U}(x)|} & \text{if } u = u^* \\ \frac{\epsilon}{|\mathcal{U}(x)|} & \text{if } u \neq u^* \end{cases}$$

3.2.1 SARSA

SARSA converge to the optimal action-value function, $Q(x,u) \rightarrow Q^*(x,u)$, as the number of episodes $k \rightarrow \infty$ as long as:

- the sequence of ϵ -greedy policies $\pi_k(u|x)$ is GLIE,
- the sequence of step sizes α_k is Robbins-Monro.

Algorithm 1 SARSA(epsilon, n_actions, n_states, episodes)

```
Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal-state, \cdot) = 0$
while $episode \leq episodes$ do
    Initialize $S$
    Choose $A$ from $S$ using $\epsilon$-greedy method
    while S is not terminate node or episode does not finish do
        Take action $A$, observe $R, S'$
        Choose $A'$ from $S'$ using $\epsilon$-greedy method
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma Q\left(S', A'\right) - Q(S,A)\right]$
        $S \leftarrow S'; A \leftarrow A'$;
    end while
    episode += 1
end while
```

3.2.2 Q-Learning

Q-Learning converges almost surely to Q^* assuming all state-control pairs continue to be updated and the sequence of step sizes α_k is Robbins-Monro.

Algorithm 2 Q-learning(epsilon, n_actions, n_states, episodes)

```
Initialize $Q(s,a)$, for all $s \in \mathcal{S}, a \in \mathcal{A}(s)$, arbitrarily, and $Q(terminal-state, \cdot) = 0$
while $episode \leq episodes$ do
    Initialize $S$
    while S is not terminate node or episode does not finish do
        Choose $A$ from $S$ using $\epsilon$-greedy method
        Take action $A$, observe $R, S'$
        $Q(S,A) \leftarrow Q(S,A) + \alpha\left[R + \gamma \max_a Q\left(S', a\right) - Q(S,A)\right]$
        $S \leftarrow S'$;
    end while
    episode += 1
end while
```

3.3 Results

Here're the hyper-parameters I used in experiment. The x-axis denotes iterations, the y-axis denote corresponding Q-value.

- gamma = 0.9
- learning rate = 0.3
- epsilon = 0.8
- max episodes = 7000
- greedy ratio = 0.8. Greedy ratio denotes how many percents of the episodes should use ϵ-greedy algorithm. We only used greedy strategy at the beginning of the training, while the rest portion just exploit the best policy.

The video of the running car has been uploaded with code bundle on Gradescope. After learning from 5000 episodes, the car can almost arrives at the goal every time.

Here're the plots for Q values of $(0, 0, u), (-1.0, .05, u), (.25, -.05, u)$ for all $u \in \mathcal{U}$.

Figure 10: Q(0,0,0) - SARSA

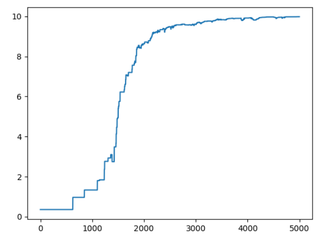


Figure 11: Q(0,0,1) - SARSA

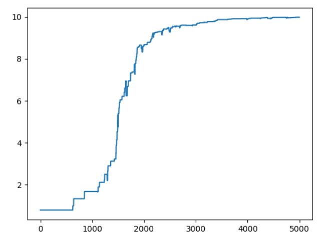


Figure 12: Q(0,0,2) - SARSA

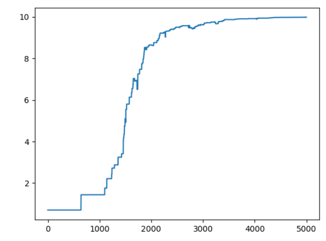


Figure 13: Q(-1.0,.05,0) - SARSA

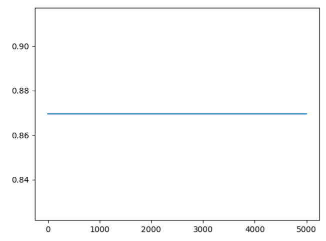


Figure 14: Q(-1.0,.05,1) - SARSA

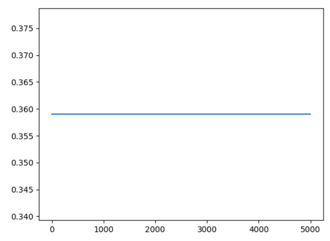


Figure 15: Q(-1.0,.05,2) - SARSA

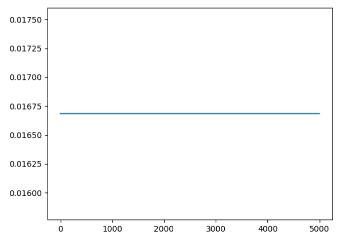


Figure 16: Q(.25,-.05,0) - SARSA

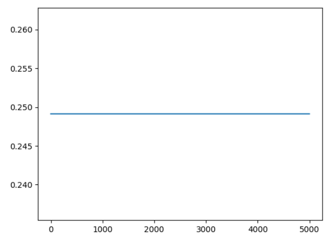


Figure 17: Q(.25,-.05,1) - SARSA

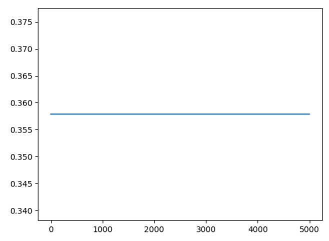


Figure 18: Q(.25,-.05,2) - SARSA

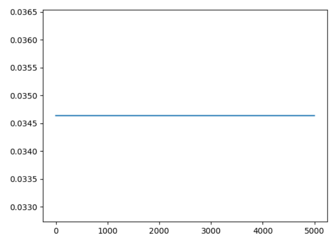


Figure 19: Q(0,0,0) - Q Learning

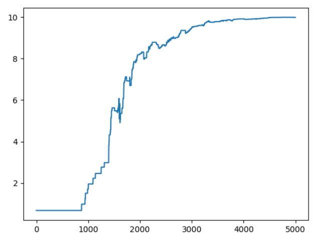


Figure 20: Q(0,0,1) - Q Learning

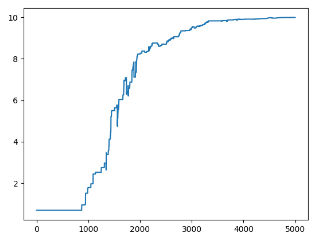


Figure 21: Q(0,0,2) - Q Learning

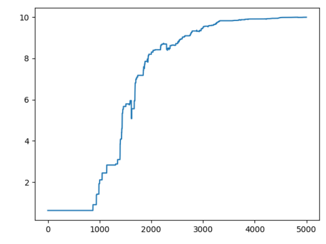


Figure 22: Q(-1.0,.05,0) - Q Learning

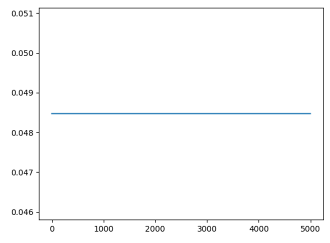


Figure 23: Q(-1.0,.05,1) - Q Learning

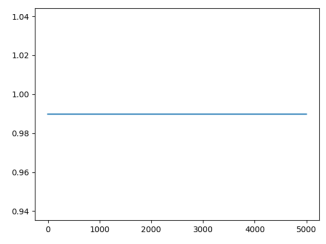


Figure 24: Q(-1.0,.05,2) - Q Learning

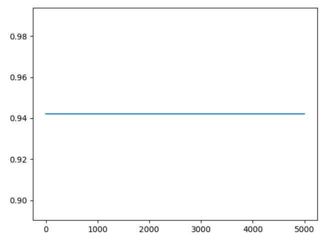


Figure 25: Q(.25,-.05,0) - Q Learning

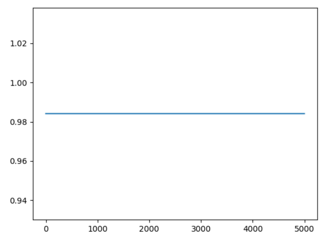


Figure 26: Q(.25,-.05,1) - Q Learning

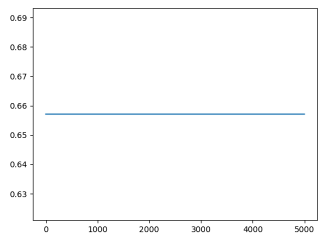
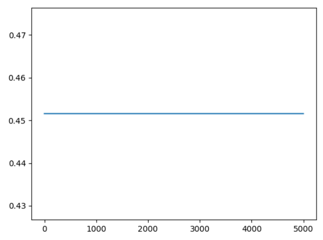
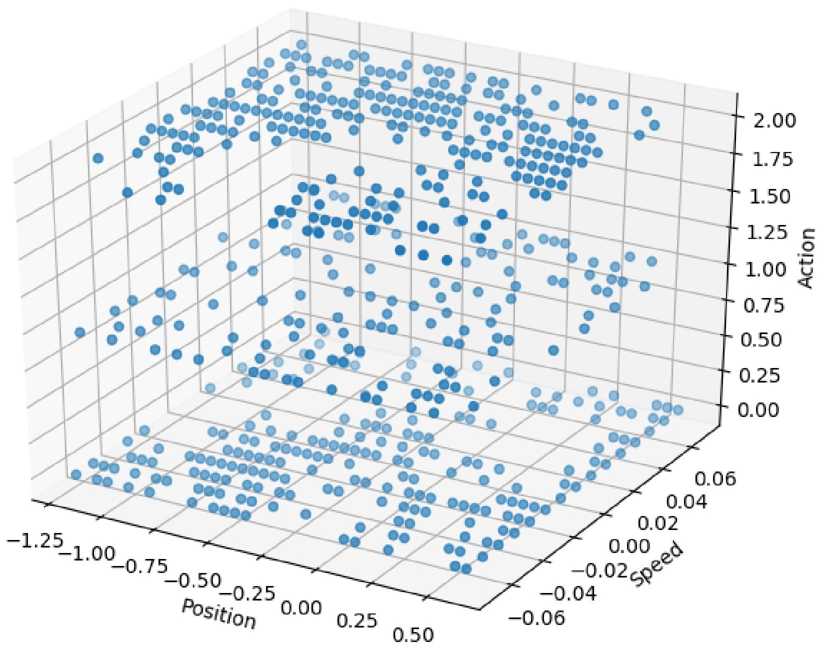


Figure 27: Q(.25,-.05,2) - Q Learning



Here are the plot of policies.

Figure 28: Policy of SARSA



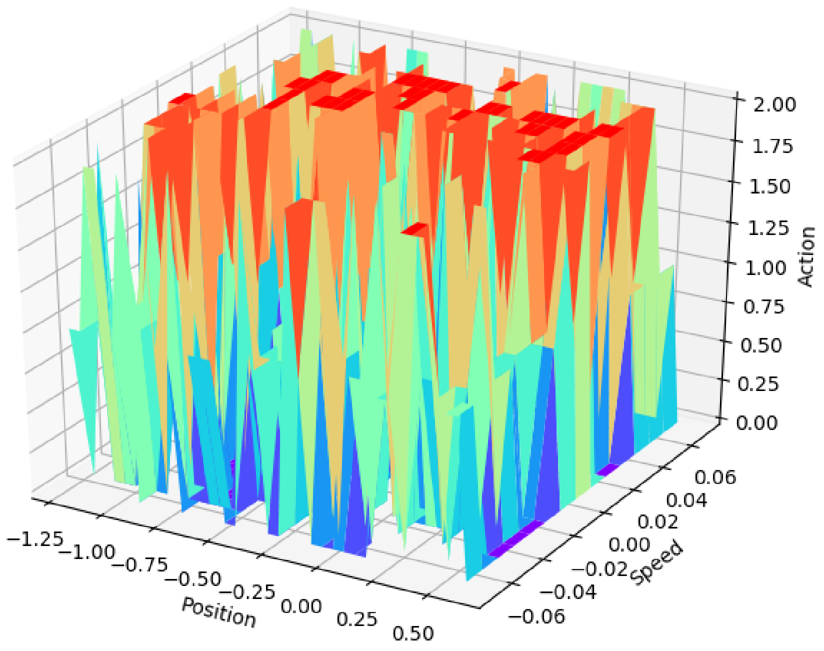
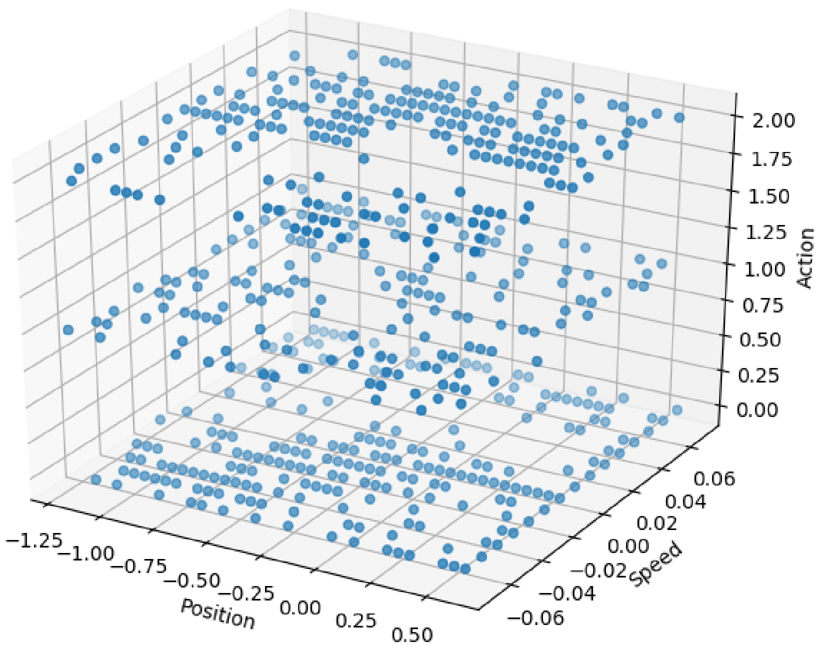
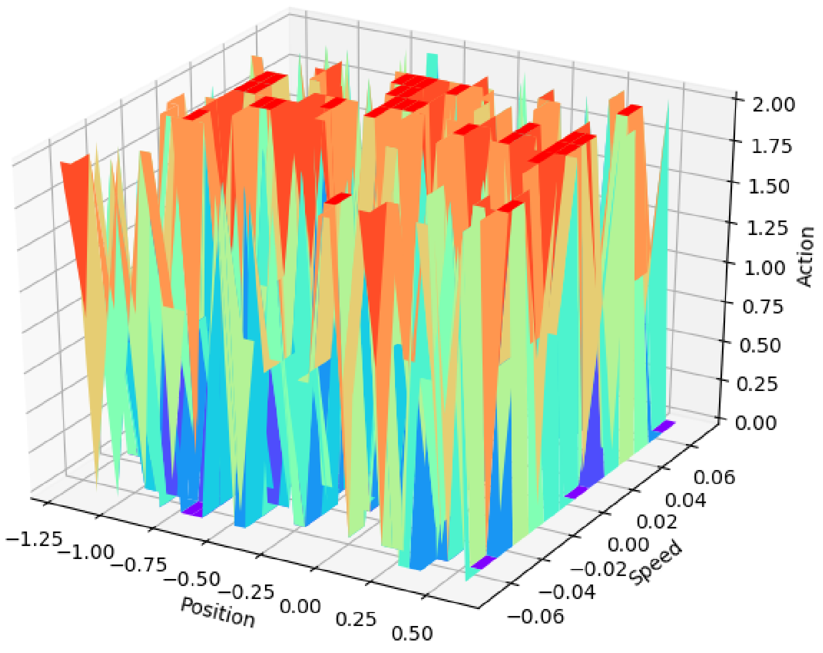


Figure 29: Policy of Q Learning





One interesting observation is that the direction of the action tends to be the same as the direction of the speed. For example, if the speed is negative(left), the action is most likely to be 0(left). This is intuitive in physics because this helps the car to gain energy.

The unexpected part is that there are still some outliners that doesn't obey the rule above. This might be caused by the uncertainty of ϵ-greedy exploration, and the system need more episodes to become fully converged.

4 Problem 4

4.1 Problem Formulation

4.1.1 MDP Definition

- State space: $\mathcal{X} = \{(x_1, x_2)|x_1 \in [-\pi, \pi], x_2 \in [-2, 2]\}$.
  In the experiment, I discretized the space of theta with interval $\delta_1 = \frac{2\pi}{125}$, and the space of velocity with $\delta_2 = \frac{4}{50}$.
- Control space: $\mathcal{U} = \{u|u \in [-2, 2]\}$.
  In the experiment, I discretized the space of theta with interval $\delta_u = \frac{4}{20}$
- Motion model: According to physics formula, we have $\mathbf{x}' = (x_1', x_2') = \mathbf{x} + f(\mathbf{x}, u)\delta t + \sigma d\omega$, where ω is Gaussian motion noise (Brownian motion). Due to noise, the transition probability $p_f(\mathbf{x}'|\mathbf{x}, u)$ is Gaussian with mean $\mathbf{x} + f(\mathbf{x}, u)\delta t$ and covariance $\sigma\sigma^T\delta t$. i.e.

$$p_f(\mathbf{x}'|\mathbf{x}, u) \sim \mathcal{N}(\mathbf{x} + f(\mathbf{x}, u)\delta t, \sigma\sigma^T\delta t)$$

- Stage cost: $\ell(\mathbf{x}, u) = 1 - \exp(k\cos x_1 - k) + \frac{r}{2}u^2$
- Terminal cost: $\mathfrak{q}(\mathbf{x_T}) = 1 - \exp(k\cos x_1 - k)$

4.1.2 Optimization Problem

Out goal is to find the optimal control policy, which satisfies

$$\pi^*(x) = \underset{u \in \mathcal{U}(x)}{\arg\min}\left\{\ell(x, u) + \gamma\mathbb{E}_{x' \sim p_f(\cdot|x,u)}\left[V^*(x')\right]\right\}$$

4.1.3 Interpolation Problem

The state space is actually continuous, but it's discretized for the convenience to be solved by MDP, thus the policy is not continuous anymore. So for any non-grid state, we need to utilize the descritized policy with interpolating strategy to estimate the control of any given state.

4.2 Technical Approach

I intend to use value iteration and policy iteration to solve the problem. The key difference between these two algorithms is that it takes an infinite number of iterations for VI to converge, but Pl converges in $|\mathcal{U}|^{|\mathcal{X}|}$ iterations (all possible policies) in the worst case.

4.2.1 Value iteration

Value Iteration (VI) algorithm: applies the DP recursion with an arbitrary initialization $\overline{V}_0(x)$ for all $x \in \mathcal{X}$ until it converges to a stationary point (or until the difference is under a small threshhold):

$$\overline{V}_{t+1}(x) = \min_{u \in \mathcal{U}(x)} \left[\ell(x,u) + \gamma \sum_{x' \in \mathcal{X}} p_f\left(x' | x, u\right) \overline{V}_t\left(x'\right) \right], \quad \forall x \in \mathcal{X}$$

4.2.2 Policy iteration

The PI algorithm iterates the following two steps:

1. Policy Evaluation: given a policy π, compute V^π by solving the linear system of equations:

$$V^\pi(x) = \tilde{\ell}(x, \pi(x)) + \sum_{x' \in \tilde{\mathcal{X}} \backslash \{0\}} \tilde{p}_f\left(x' | x, \pi(x)\right) V^\pi\left(x'\right), \quad \forall x \in \tilde{\mathcal{X}} \backslash \{0\}$$

2. Policy Improvement: obtain a new stationary policy π' :

$$\pi'(x) = \underset{u \in \tilde{\mathcal{U}}(x)}{\arg\min} \left[\tilde{\ell}(x,u) + \sum_{x' \in \tilde{\mathcal{X}} \backslash \{0\}} \tilde{p}_f\left(x' | x, u\right) V^\pi\left(x'\right) \right], \quad \forall x \in \tilde{\mathcal{X}} \backslash \{0\}$$

   Repeat the two steps above until $V^{\pi'}(x) = V^\pi(x)$ for all $x \in \tilde{\mathcal{X}} \backslash \{0\}$ (or until the difference is under a small threshhold).

4.2.3 Interpolation

My strategy is intuitive: the control of one state should be similar to its neighbouring states. For any given state x, I calculated the weighted sum over the whole descritized grid points. The weight is the normalized Gaussian probability with the mean of current state and $\sigma = 1$. i.e. $p \sim N(\mathbf{x}; \mu, \sigma)$.

4.3 Result

4.3.1 Value over iterations

Here are 3 iteration-value plots over state $(theta\_min, vmin)$, $(theta\_max, vmin)$ and $(theta\_min, vmax)$.

Figure 30: Policy of value iteration

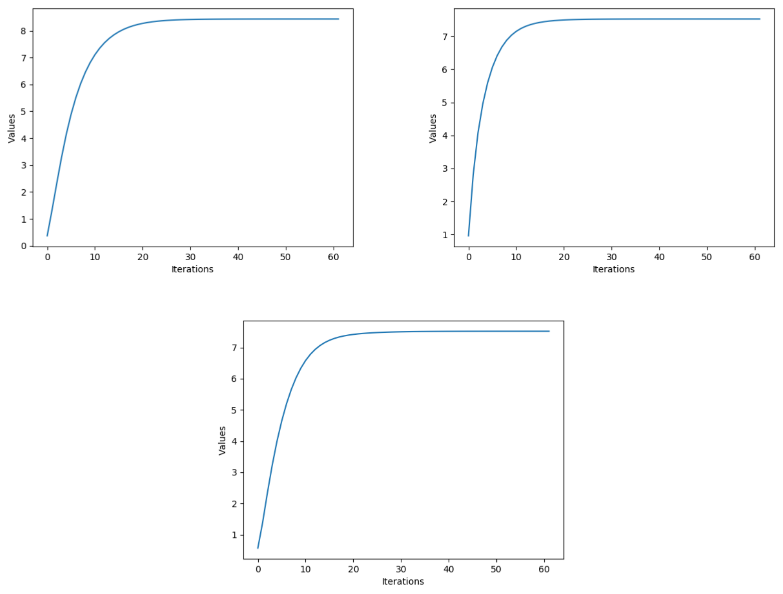
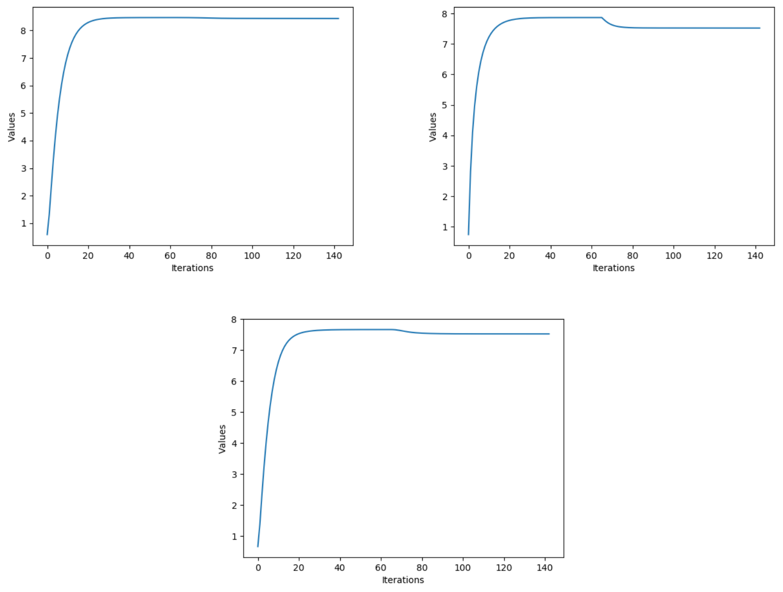


Figure 31: Policy of policy iteration



Here are policies of 2 different iteration algorithms.

Figure 32: Policy of value iteration

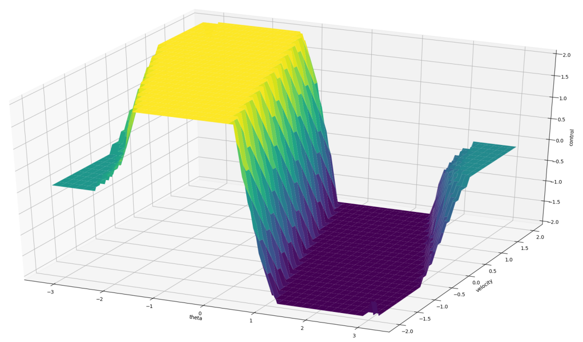
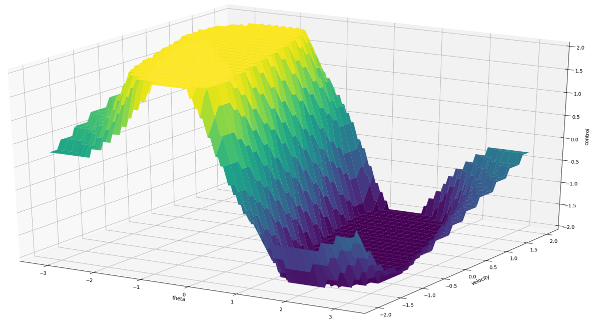


Figure 33: Policy of policy iteration



The videos have been uploaded to gradescope. Both algorithms converges to nearly the same policy, but VI used less iterations to converge.

As for the noise, I gradually increased σ from 0.05 to 10. The system achieved stable equilibrium in the vertical state when $\sigma \leq 0.1$. The vibration is bearable when $0.1 \leq \sigma \leq 0.5$, and above that the system gradually lost control to erect the pendulum.