Problem 4

1 Problem Formulation

1.1 MDP Formulation for Inverted Pendulum

Inverted Pendulum (IP) optimization control problem can also be solved by formulating it into an infinite horizon MDP with continuous state space and control space.

- State space: $\mathbf{x} \in \mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 = [-\pi, \pi] \times [-v_{max}, v_{max}]$, where $v_{max} > 0$ is the max angular velocity, $x_1 \in \mathcal{X}_1$ is the angle of the pendulum and $x_2 \in \mathcal{X}_2$ is the angular velocity.
- Control space: $\mathcal{U} = [-u_{max}, u_{max}]$, where $u_{max} > 0$ is the max torque.
- Motion model: $(\mathbf{x}'|\mathbf{x}, u) \sim N(\mathbf{x} + f(\mathbf{x}, u)dt, \sigma\sigma^T dt)$, where

$$f(\mathbf{x}, u) = \begin{bmatrix} x_2 \\ a\sin x_1 - bx_2 + u \end{bmatrix}, \tag{1}$$

  $\sigma \in \mathbb{R}^2$ is covariance, $a > 0$ summarizes the effects of the effects of gravity, mass and length of the pendulum, $b > 0$ is the damping and friction, dt is the differential of time t.

- Stage cost. The stage cost at each state $\mathbf{x}$ and control u is

$$l(\mathbf{x}, u) = 1 - \exp(k\cos x_1 - k) + \frac{r^2}{2}u^2. \tag{2}$$

- Terminal cost $\mathfrak{q} = 0$. In fact, there is no terminal states.

Then we need to solve Bellman Equation for optimization:

$$V^*(\mathbf{x}) = \min_{\pi} l(\mathbf{x}, u) + \gamma \mathbb{E}_{\mathbf{x}' \sim p_f(\cdot|\mathbf{x},u)}[\min_{u \in \mathcal{U}(x)} V^*(x')] \tag{3}$$

$$\pi^*(x) = \arg\min_{u \in \mathcal{U}(x)} V^*(\mathbf{x}) \tag{4}$$

$$s.t.\ \mathbf{x}_0 = (x_{10}, 0) \tag{5}$$

In order to actually solve this problem, one possible way is to discretize the state space and control space and then using interpolation techniques to fill out remaining the rest of the space.
So we will discretize state space $\mathcal{X}$ and control space $\mathcal{U}$ as discrete step d_{x_1}, d_{x_2} and d_u:

$$\mathcal{X} = \{-\pi, -\pi + d_{x_1}, -\pi + 2d_{x_1}, ..., \pi\} \times \{-v_{max}, -v_{max} + d_{x_2}, -v_{max} + 2d_{x_2}, ..., v_{max}\} \tag{6}$$

and

$$\mathcal{U} = \{-u_{max}, -u_{max} + d_u, -u_{max} + 2d_u, ..., u_{max}\}, \tag{7}$$

then using fixed time step δt to calculate the motions. Hence the model then becomes a finite-space MDP.
After this, we can use interpolation techniques to fill up to make the spaces more smooth.

1.2 Interpolation for Policy

Since the state $\mathbf{x} \in \mathbb{R}^2$, we consider this problem as a 2D interpolation problem for policy $\pi(\mathbf{x})$ over the discrete state space as Eq. 6 states.
After interpolation, we will have our policy over a precise and "continuous" state space $\mathcal{X}' = \{-\pi, -\pi + d_{x_1'}, -\pi + 2d_{x_1'}, ..., \pi\} \times \{-v_{max}, -v_{max} + d_{x_2'}, -v_{max} + 2d_{x_2'}, ..., v_{max}\}$, where $d_{x_1'} < d_{x_1}, d_{x_2'} < d_{x_2}$.

2 Technical Approach

Since the optimal cost of discounted problem satisfies the Bellman Equation (Eq. 3) via its equivalence to Stochastic Shortest Problem (SSP), we can use Value Iteration (VI) or Policy Iteration (PI) for solution.

2.1 Value Iteration

VI starts at the terminals and then works backwards (Dynamic Programming) although there is no real "terminals" probably. It usually initializes V_0 and updates V for convergence as Eq. 8. The pseudocode is shown in Algorithm 1. Usually, a threshold θ is required for termination of the algorithm.

$$V_k(\mathbf{x}) \leftarrow \min_{u \in \mathcal{U}} [l(\mathbf{x}, u) + \gamma \sum_{x'} p_f(\mathbf{x}'|\mathbf{x}, u) V_{k-1}(x')] \tag{8}$$

Algorithm 1: VI Algorithm

```
Initialize $V(\mathbf{x})$, given threshold $\theta$.
$k \leftarrow 0$
repeat
    $k \leftarrow k + 1$
    for each state $\mathbf{x}$ do
        $V_k(\mathbf{x}) \leftarrow \min_{u \in \mathcal{U}} [l(\mathbf{x}, u) + \gamma \sum_{x'} p_f(\mathbf{x}'|\mathbf{x}, u) V_{k-1}(x')]$
    end
until $max|V_k(\mathbf{x}) - V_{k-1}(\mathbf{x})| < \theta$;
for each state $\mathbf{x}$ do
    $\pi(\mathbf{x}) = \arg\min_{u \in \mathcal{U}} [l(\mathbf{x}, u) + \gamma \sum_{x'} p_f(\mathbf{x}'|\mathbf{x}, u) V_{k-1}(x')]$
end
return $\pi, V_k$
```

2.2 Policy Iteration

PI starts at a different point of view: what if we update V given a policy $\pi(\mathbf{x})$? This is valid due to the Policy Evaluation Theorem.
In general, we want solve equations:

$$(I - \tilde{\mathbf{P}})\mathbf{v} = \mathbf{l} \tag{9}$$

for $\mathbf{v}$, where $\mathbf{v}_i = V(i)$, $\mathbf{l}_i = l(i, \pi(i))$, $\tilde{\mathbf{P}}_{ij} = \tilde{p}_f(j|i, \pi(i))$. Since the uniqueness of the solution to $\mathbf{v}$, we can get iterative solution to Eq. 9 by:

$$\begin{aligned}
\mathbf{v}_1 &= \mathbf{l} + \tilde{\mathbf{P}}\mathbf{v}_0 \\
\mathbf{v}_2 &= \mathbf{l} + \tilde{\mathbf{P}}\mathbf{v}_1 = \mathbf{l} + \tilde{\mathbf{P}}\mathbf{l} + \tilde{\mathbf{P}}^2\mathbf{v}_0 \\
&... \\
\mathbf{v}_T &= (I + \tilde{\mathbf{P}} + \tilde{\mathbf{P}}^2 + ... + \tilde{\mathbf{P}}^{T-1})\mathbf{l} + \tilde{\mathbf{P}}^T\mathbf{v}_0 \\
&... \\
\mathbf{v}_\infty &\to (I - \tilde{\mathbf{P}})^{-1}\mathbf{l}
\end{aligned}$$

if $I - \tilde{\mathbf{P}}$ is invertible. This iterative processing is called Policy Evaluation.
Therefore, the procedure is first we do Policy Evaluation given policy π until V is convergent to solve Eq. 9, then we improve policy π according to V by

$$\pi(\mathbf{x}) = \arg\min_{u \in \mathcal{U}} [l(\mathbf{x}, u) + \gamma \sum_{x'} p_f(\mathbf{x}'|\mathbf{x}, u) V_{k-1}(\mathbf{x}')] \tag{10}$$

until the policy is stable for all states.

Algorithm 2: PI Algorithm

```
Initialize $V(\mathbf{x})$ and policy $\pi(\mathbf{x})$, given threshold $\theta$.
$k \leftarrow 0$
repeat
    $k \leftarrow k + 1$
    for each state $\mathbf{x}$ do
        $V_k^\pi(\mathbf{x}) \leftarrow l(\mathbf{x}, \pi(\mathbf{x})) + \gamma \sum_{x'} p_f(\mathbf{x}'|\mathbf{x}, \pi(\mathbf{x}) V_{k-1}^\pi(\mathbf{x}')$
    end
until $max|V_k(\mathbf{x}) - V_{k-1}(\mathbf{x})| < \theta$;
repeat
    for each state $\mathbf{x}$ do
        $\pi'(\mathbf{x}) \leftarrow \pi(\mathbf{x})$
        $\pi(\mathbf{x}) = \arg\min_{u \in \mathcal{U}} [l(\mathbf{x}, u) + \gamma \sum_{x'} p_f(\mathbf{x}'|\mathbf{x}, u) V_{k-1}(\mathbf{x}')]$
    end
until $\pi(\mathbf{x}) = \pi'(\mathbf{x})$ for every state;
return $\pi, V_k$
```

2.3 Cubic Interpolation

In this problem we simply choose cubic interpolation to expand policy over state space. Cubic interpolation is the simplest method taht offers true continuity between the segments. It computes a spline where each piece is a third-degree polynomial specified in Hermite form, i.e by its values and first derivatives at the end points of the correspondence domain interval. A brief example of cubic interpolation is as follows[1]:
On the unit interval [0, 1], given a start point p_0 at $t = 0$ and an ending point p_1 at $t = 1$ with starting tangent m_0 at $t = 0$ and the ending tangent m_1 at $t = 1$, the polynomial can be defined by

$$p(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)m_0 + (-2t^3 + 3t^2)p_1 + (t^3 - t^2)m_1, \tag{11}$$

where $t \in [0, 1]$.
More generally, we can adapt Eq. 11 to arbitrary interval:

$$p(x) = h_{00}(t)p_k + h_{10}(t)(x_{k+1} - x_k)m_k + h_{01}(t)p_{k+1} + h_{11}(t)(x_{k+1} - x_k)m_{k+1} \tag{12}$$

with $t = \frac{x - x_k}{x_{k+1} - x_k}$ and h refers to the basis functions (See Table 1).

$$B_k(t) = \binom{3}{k} t^k (1 - t)^{3-k} \tag{13}$$

Table 1: Basis Functions, the "expanded" column shows the representation used in example above. The "factorized" column shows immediately, that h_{10} and h_{11} are zero at the boundaries.The "Bernstein" column shows the decomposition of the Hermite basis functions into Bernstein polynomials of order 3 shown in Eq.13

	expanded	factorized	Bernstein
$h_{00}(t)$	$2t^3 - 3t^2 + 1$	$(1 + 2t)(1 - t)^2$	$B_0(t) + B_1(t)$
$h_{10}(t)$	$t^3 - 2t^2 + t$	$t(1 - t)^2$	$\frac{1}{3}B_1(t)$
$h_{01}(t)$	$-2t^3 + 3t^2$	$t^2(3 - 2t)$	$B_3(t) + B_2(t)$
$h_{11}(t)$	$t^3 - t^2$	$t^2(t - 1)$	$-\frac{1}{3}B_2(t)$

For this problem, we need to consider using 2D cubic interpolation, fortunately, which can be done directly by `scipy.interpolate.interp2d(**, kind='cubic')`.

3 Results and Discussion

3.1 Environment and Experiment Setting

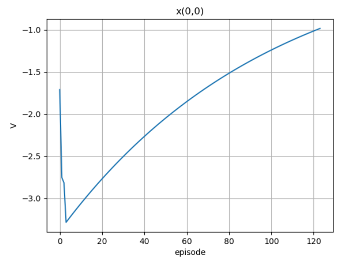
For pendulum, assume $a = 4.0$, damping $b = 1.0$, shape of the cost $k = 3.0$, and scale of the cost $r = 0.001$. The covariance matrix $\Sigma = \sigma\sigma^T = 0.001 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Time step $\delta t = 0.5$.
For discretization, $d_{x_1} = 0.1$, $d_{x_2} = 0.2$, $d_u = 0.25$, $v_{max} = 4$, $u_{max} = 3$, then the discrete state space $\mathcal{X} = \{-\pi, -\pi + 0.1, ..., \pi\} \times \{-4.0, -3.8, -3.6, ..., 4.0\}$, the discrete control space $\mathcal{U} = \{-3.0, -2.75, ..., 3.0\}$.
For VI and PI, the threshold $\theta = 1e - 3$ for termination, discount factor $\gamma = 0.99$. Value functions V is initialized by random values from a standard Gaussian distribution.

By this setting, $\frac{2v_{max}}{n_2}\delta t / \frac{2\pi}{n_1} = 0.9937 \approx 1$. After we get the optimal policy, a test is executed to test the policy. After VI or PI, a cubic interpolation is implemented, making $d_{x_1} = 0.01$, $d_{x_2} = 0.1$.

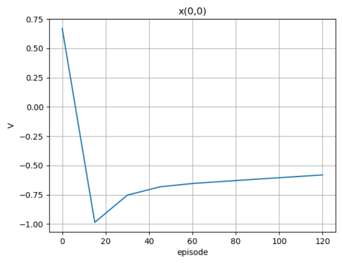
3.2 Results

Two qualitative results can be found in two video .mp4 files for VI and PI, respectively.

Value function V(x) over episodes is shown in Fig.3 and Fig.4. As we can see, value functions gradually converge and it looks like a "convex" function centered at (0,0). This makes sense since we want to push the pendulum to the origin position. For a specific set of states $\mathbf{x} \in \{(0,0), (2,-1)\}$, the trending of V over the episodes is shown in Fig.1 and Fig.2.

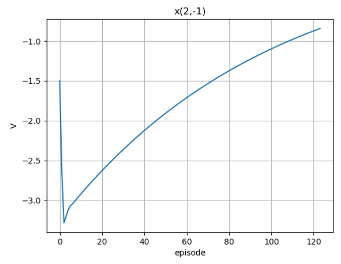


(a) Value Iteration

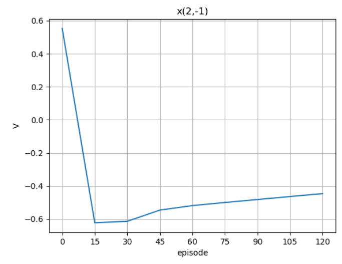


(b) Policy Iteration

Figure 1: V((0,0))



(a) Value Iteration

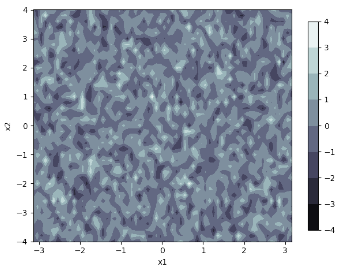


(b) Policy Iteration

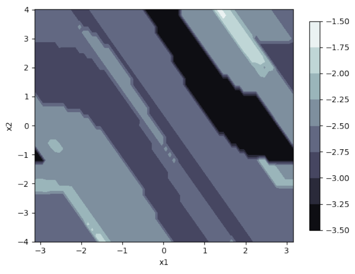
Figure 2: V((2,-1))

From Fig.1, Fig.2, Fig.3 and Fig.4 we can see that VI will converge a little bit faster for this problem.

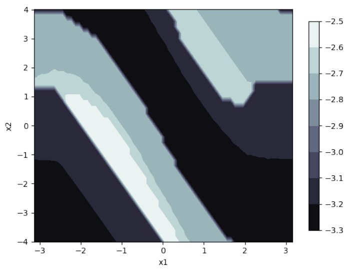
The policy $\pi(x)$ after interpolation over the discrete space is shown in Fig.5. We can see that the final optimal policy via two algorithms are almost the same. The results show that with the policy is centered around the origin $\mathbf{x} = (0,0)$. There is little difference between VI and PI in the final result.



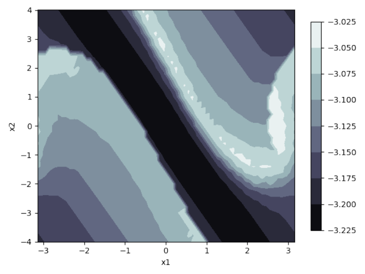
(a) Episode 0



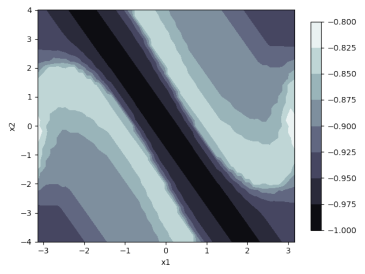
(b) Episode 2



(c) Episode 4



(d) Episode 6



(e) Final

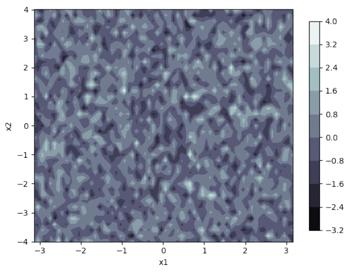
Figure 3: V(x) over State Space, Value Iteration

3.3 Discussion

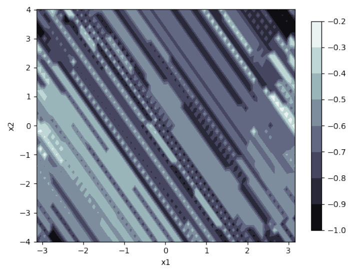
Parameters Setting In this problem, we need to define parameters of pendulum by ourselves. The most critical one is r (Eq.2), which is used to scale the control cost. The purpose of r is that, we cannot let the cost contributed by control u exceeds the part from angle displacement, while we also want to control influences the cost somehow. If the cost contributed by u is more significant than the state x, u will suppress the angle change, which will make V has little to do with the state x but dependent on u. Once we choose a larger r, it is better to use a larger k to amplify the contribution of state. By my observation, $r = 0.001$, $k = 3$ are appropriate for $a = 4.0$.

Another thing I notice is the time step δt and the control space discretization. If δt is small, then we need a more "continuous" and larger control space, i.e a smaller d_u and a larger u_{max}. For example, if $\delta t = 0.05$ and $d_u = 0.1, u_{max} = 2$, by Eq.1, we can see that the contribution of u to $f(\mathbf{x}, u)$ is $u_{max} \times \delta t = 0.1$ at most. Meanwhile, if the discretization of state space in x_2 is large, e.g. $d_{x_2} = 0.2 > 0.1$, then the contribution of u will probably be ignored. This will cause the motion model is significantly dependent on $ax_1 - bx_2$, which is the gravity (almost) and the damping.

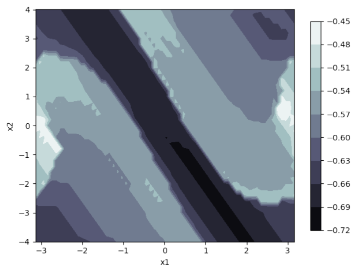
How noise influence the motion? Noise level is related to the system. If a, which is related to the gravity, mass and lengthx, is large, then it can take more noise. Assume for motion model, the covariance matrix $\Sigma = f \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, where$f > 0$. In my case, $a = 5$, then $f > 1.0$ the motion model will be influenced badly and we cannot achieve stable equilibrium pretty well.



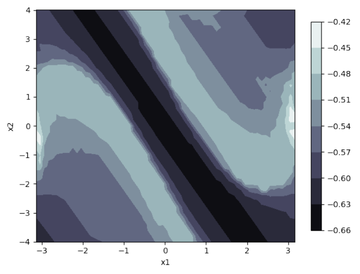
(a) Episode 0



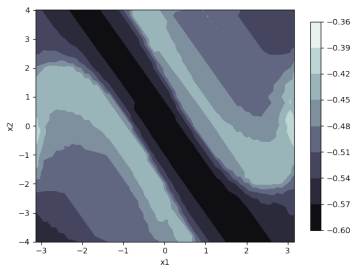
(b) Episode 2



(c) Episode 4

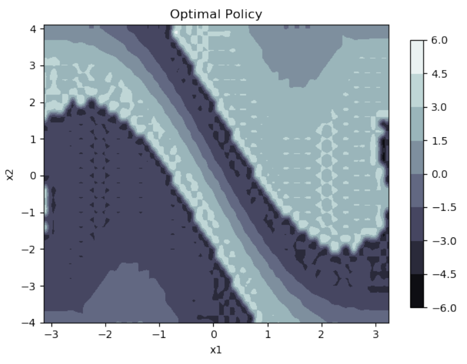


(d) Episode 6

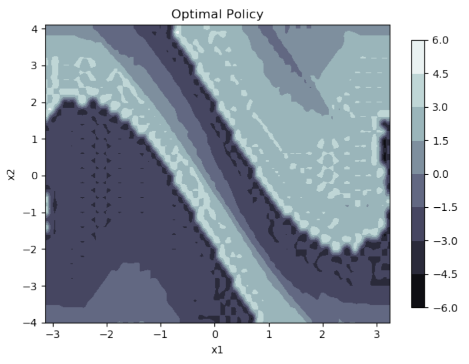


(e) Final

Figure 4: V(x) over State Space, Policy Iteration



(a) Optimal Policy, Value Iteration



(b) Optimal Policy, Policy Iteration

Figure 5: Optimal Policy with Interpolation, colorbar shows the values of control inputs

4 Acknowledge

Thanks for Shih-Chen Lin and Shuo Xu for discussion about parameters of the pendulum.

References

[1] WikiPedia: https://en.wikipedia.org/wiki/Cubic_Hermite_spline