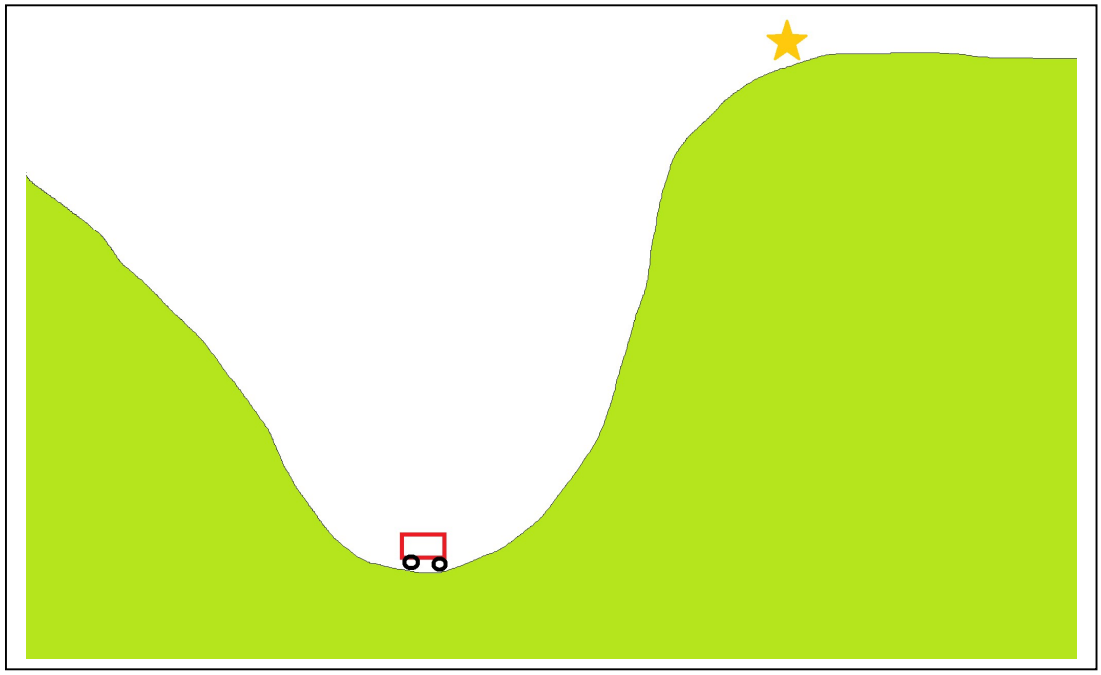
Reinforcement Learning Assignment #3

Mountain Car Problem

Submitted By: Anoushka Mathews

Introduction to the Mountain Car Problem

In the mountain car problem, the goal is to drive an underpowered car up a steep mountain road. The only solution is for the car to back up to build up enough momentum to get to the top goal on the other side.



Using Sutton and Barto, we have the following system model:

$$x_{t+1} = bound[x_t + \dot{x}_{t+1}]$$

$$\dot{x}_{t+1} = bound[\dot{x}_t + 0.001a_t + \ -0.0025\cos(3x_t)]$$

Where the bound enforces that $-1.2 \leq x_{t+1} \leq 0.5$ and $-0.07 \leq \ \dot{x}_{t+1} \leq 0.07$. When x_{t+1} reaches the left bound, we set the $\dot{x}_{t+1} = 0$, and when it reaches the right bound, we end the episode with a reward of 0 (rest of the time the reward is -1). (Sutton)

Introduction to the Fourier basis Function

In real world problems, we don't always have a system model. In such cases, we try to approximate the value function for the system.

The Fourier basis is a linear function approximation scheme that uses the terms in Fourier series as features (Konidaris). The Fourier series is used to approximate periodic functions. A multivariate Fourier series can approximate higher dimensional value functions using sinusoidal waves with weights assigned to each of them. An n-th order Fourier Basis for d state variables is defined as:

$$\phi_i(x) = \cos(\pi\, c^i\, .x),$$

Where $c^i = [c_1, \dots, c_d], \quad c_j \in [0, \dots, n], \quad 1 \leq j \leq d.$

In our case, the 2 state variables are position of the car, and the velocity of the car, i.e., x and $\dot{x}$. Since we only have 2 state variables, $d = 2$.

$$c^i = [c_1, c_2]$$

$$c_1 \in [0, 1, 2, 3] \; and \; c_2 \in [0, 1, 2, 3]$$

So, it makes sense to simply create a 2d vector that holds the c^i vector:

[0, 0]
[0, 1]
[0, 2]
[0, 3]
[1, 0]
…
…
[3, 3]

Gradient Descent Updates

Lower frequency terms should have a faster learning rate than the higher frequency terms. This will let the agent learn the general shape of the target function rapidly with slower detailed refinements. For this, the learning rate α is scaled according to the following formula

$$\alpha_i = \frac{\alpha_1}{||c_i||_2} \text{ where } \left||c_i|\right|_2 \neq 0$$

$$\alpha_i = \alpha_1 \text{ where } \left||c_i|\right|_2 = 0$$

SARSA(λ)

Sarsa(λ) is the temporal-difference method for state-action values. The eligibility trace, λ, decays the eligibility trace vector, z, that updates the weight vector for a linear function approximator. For this assignment, I used the true online Sarsa(λ) algorithm. Here is the snippet of the algorithm from Sutton and Barto.

True online Sarsa(λ) for estimating $\mathbf{w}^\top \mathbf{x} \approx q_\pi$ or q_*

Input: a feature function $\mathbf{x} : \mathcal{S}^+ \times \mathcal{A} \to \mathbb{R}^d$ such that $\mathbf{x}(terminal, \cdot) = \mathbf{0}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$, small $\varepsilon > 0$
Initialize: $\mathbf{w} \in \mathbb{R}^d$ (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  Initialize S
  Choose $A \sim \pi(\cdot|S)$ or ε-greedy according to $\hat{q}(S, \cdot, \mathbf{w})$
  $\mathbf{x} \leftarrow \mathbf{x}(S, A)$
  $\mathbf{z} \leftarrow \mathbf{0}$
  $Q_{old} \leftarrow 0$
  Loop for each step of episode:
    Take action A, observe R, S'
    Choose $A' \sim \pi(\cdot|S')$ or ε-greedy according to $\hat{q}(S', \cdot, \mathbf{w})$
    $\mathbf{x}' \leftarrow \mathbf{x}(S', A')$
    $Q \leftarrow \mathbf{w}^\top \mathbf{x}$
    $Q' \leftarrow \mathbf{w}^\top \mathbf{x}'$
    $\delta \leftarrow R + \gamma Q' - Q$
    $\mathbf{z} \leftarrow \gamma\lambda \mathbf{z} + \left(1 - \alpha\gamma\lambda \mathbf{z}^\top \mathbf{x}\right) \mathbf{x}$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha(\delta + Q - Q_{old})\mathbf{z} - \alpha(Q - Q_{old})\mathbf{x}$
    $Q_{old} \leftarrow Q'$
    $\mathbf{x} \leftarrow \mathbf{x}'$
    $A \leftarrow A'$
  until S' is terminal

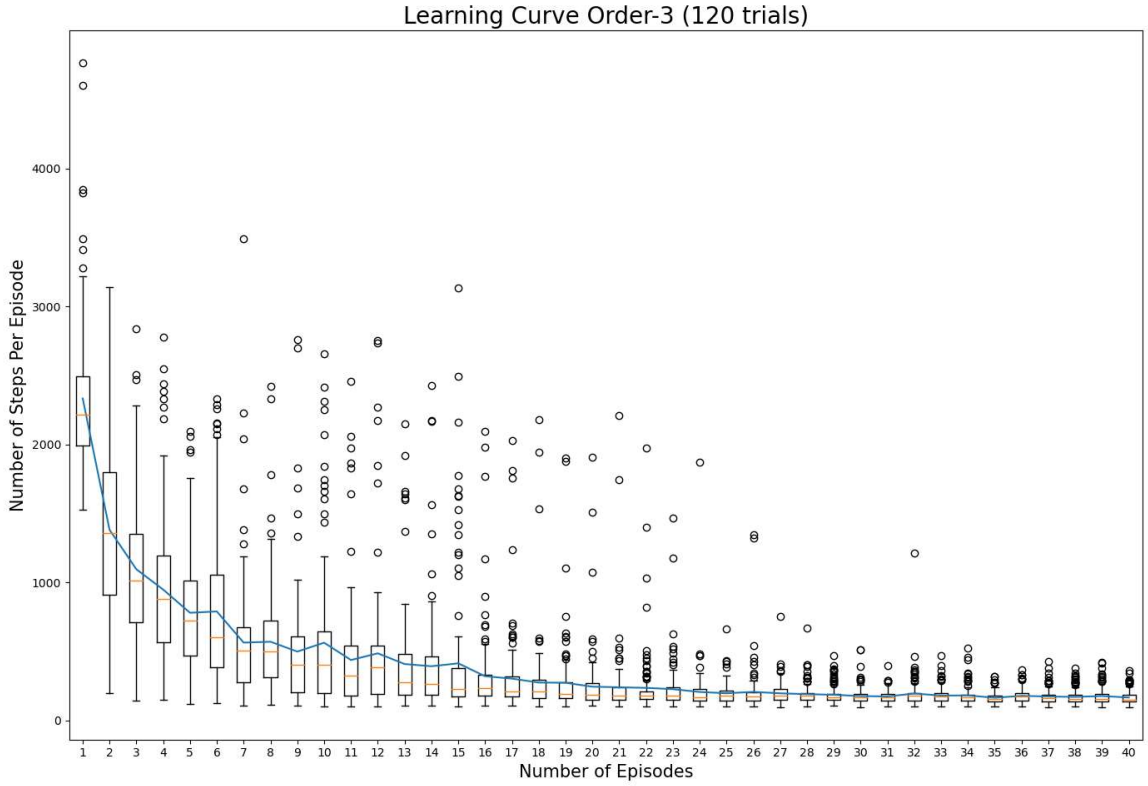
Implementation

For the implementation of the True Sarsa(λ) algorithm, I have 3 different weight variables for the 3 different actions. Similarly, I have 3 different eligibility trace vectors that all depreciate with time (except eligibility trace vector for the action in use currently). I use the Fourier Basis Function as the linear State-Action value function approximator.

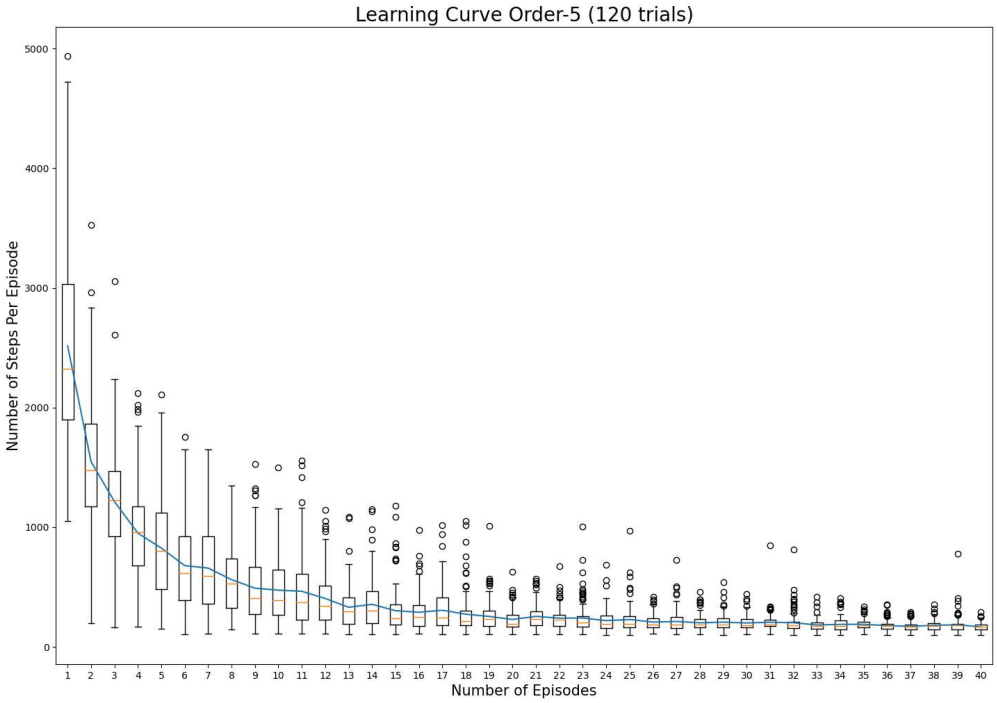
Policy used is ϵ-greedy, but since $\epsilon = 0$, it always picks the best action. Since the visited states are always perceived as bad states (unless terminal), the non-visited states always look like a better option, this leads to enough of exploration even with $\epsilon = 0$.

Results

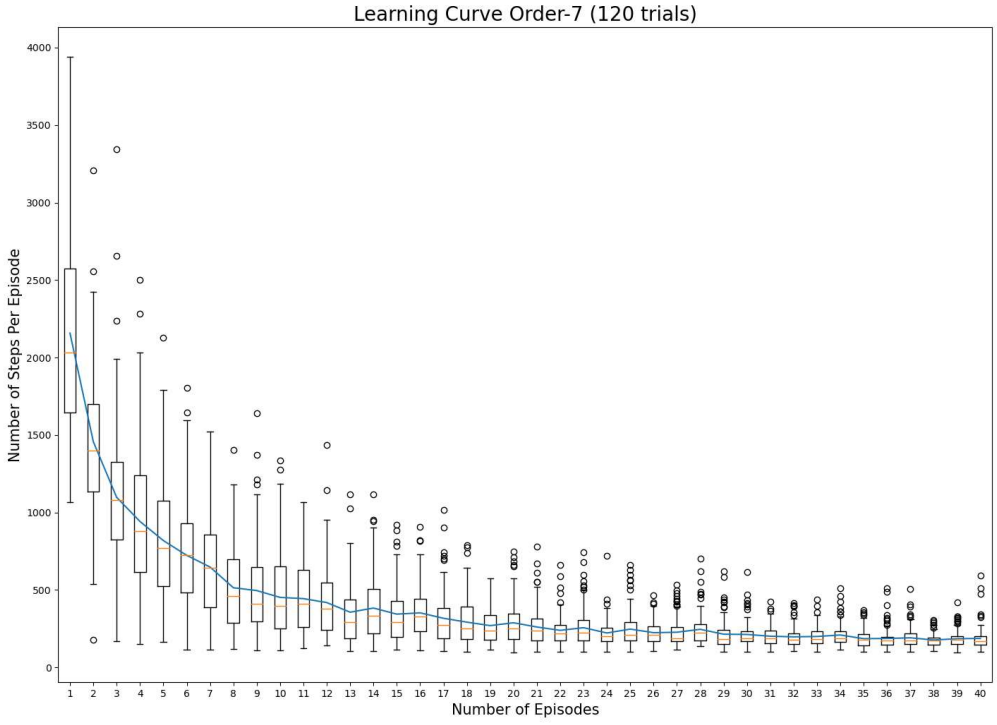
Learning Curve for **order-3** Fourier bases for $\alpha = 0.001,\ \varepsilon = 0.0,\ \gamma = 1.0,\ \lambda = 0.9.$ The number of steps per episode converges to about 200.



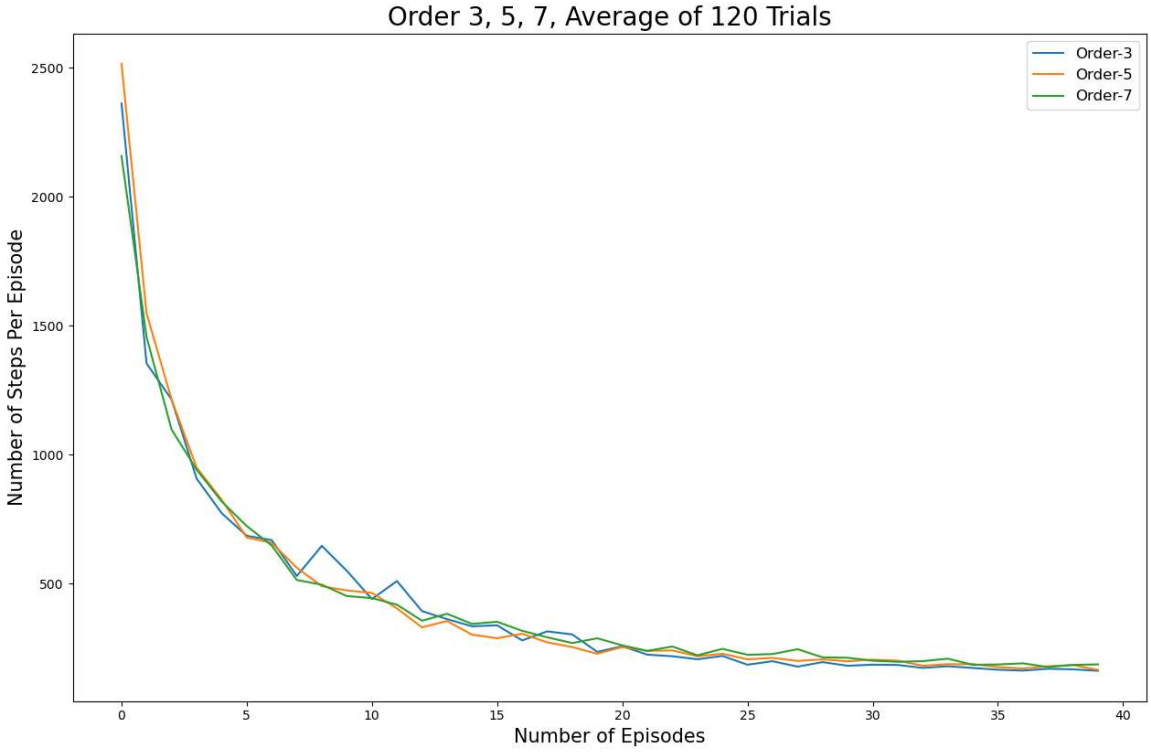
Learning Curve for **order-5** Fourier bases for $\alpha = 0.001,\ \varepsilon = 0.0,\ \gamma = 1.0,\ \lambda = 0.9$. The number of steps per episode converges to about 200.



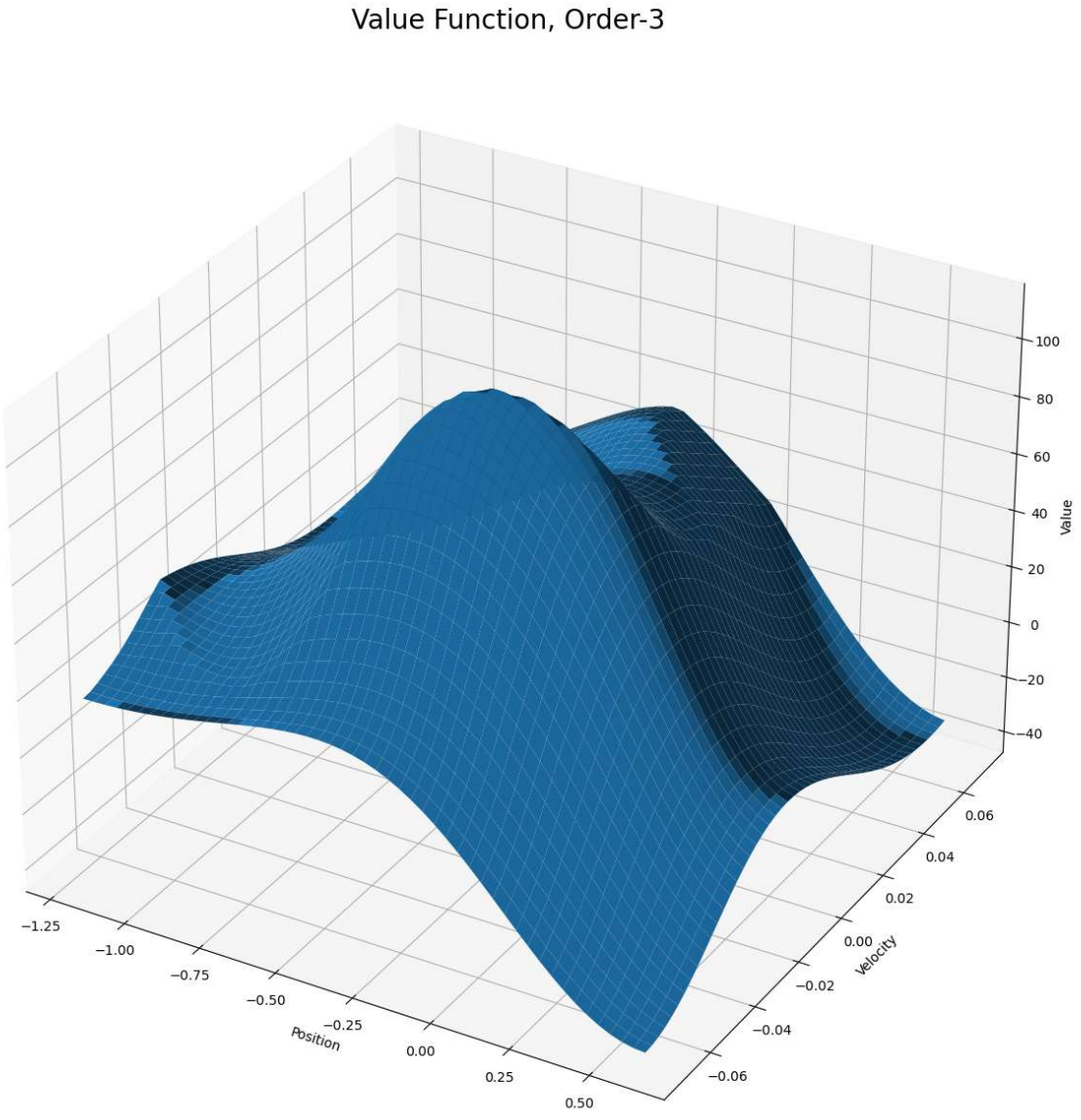
Learning Curve for **order-7** Fourier bases for $\alpha = 0.001,\ \varepsilon = 0.0,\ \gamma = 1.0,\ \lambda = 0.9$. The number of steps per episode converges to about 200.



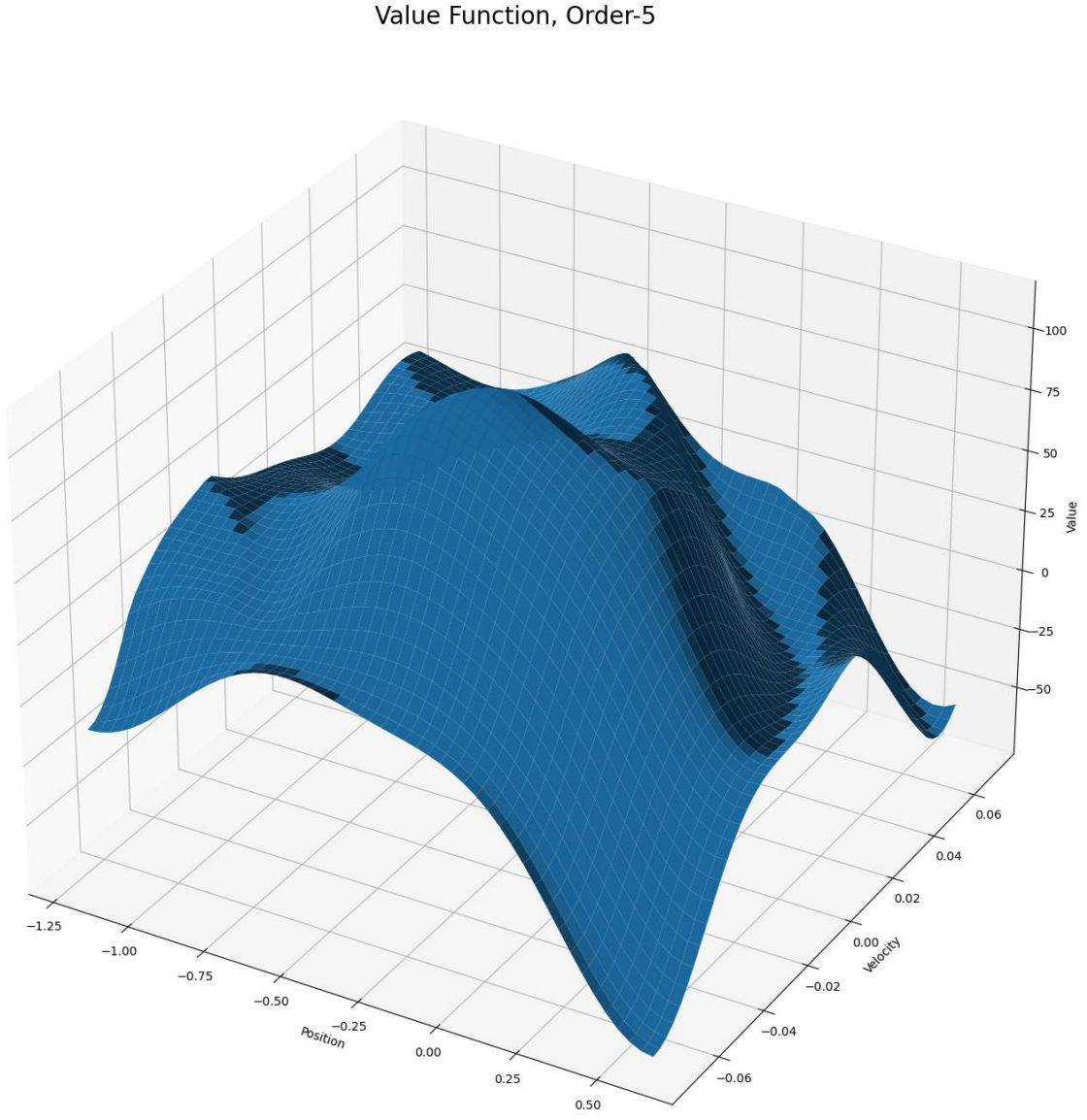
Order 3, 5, and 7 Learning Rate Means Plotted Together (converging down to about **200 steps**)



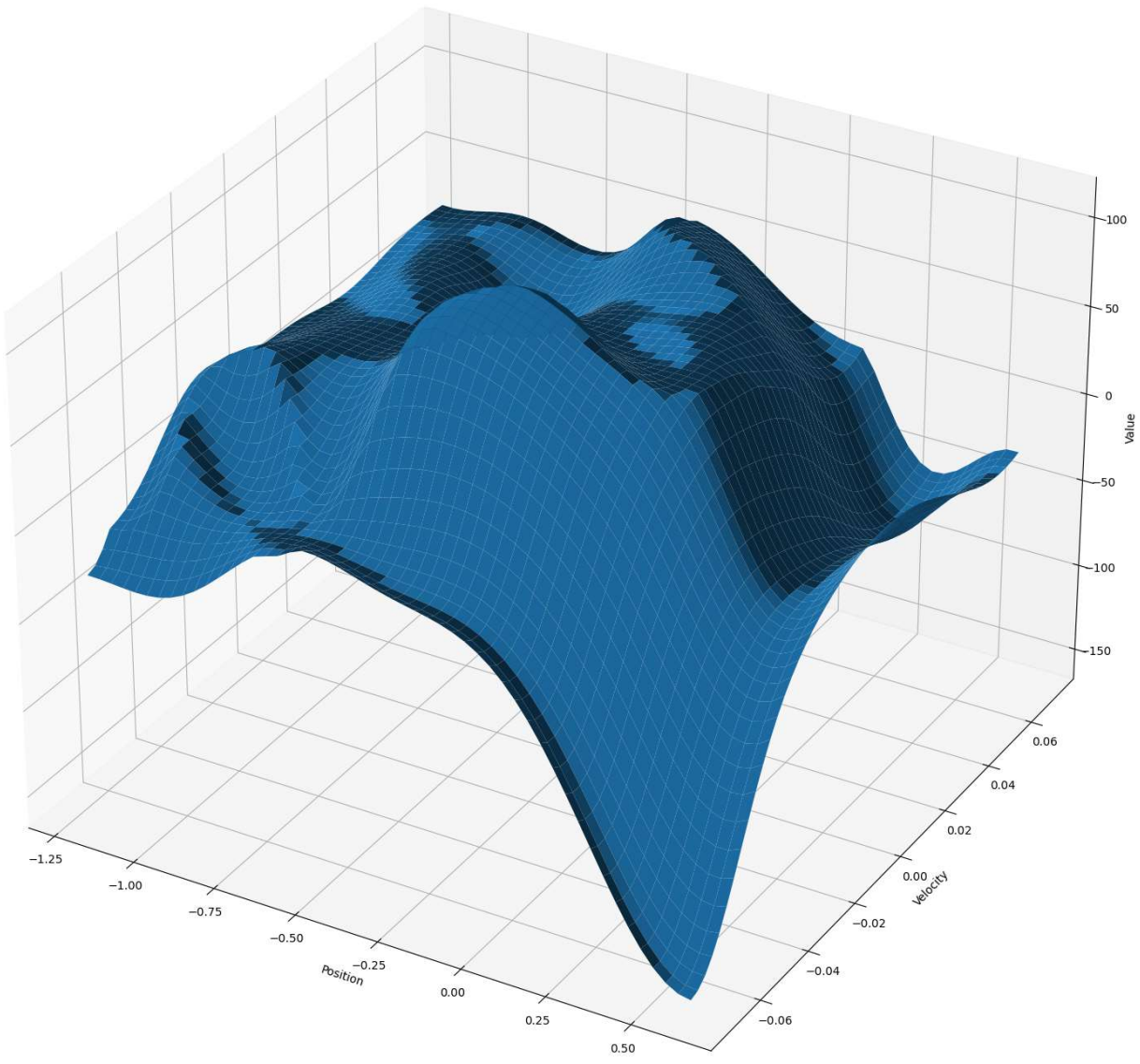
Value Function Plot for Order-3



Value Function Plot for Order-5



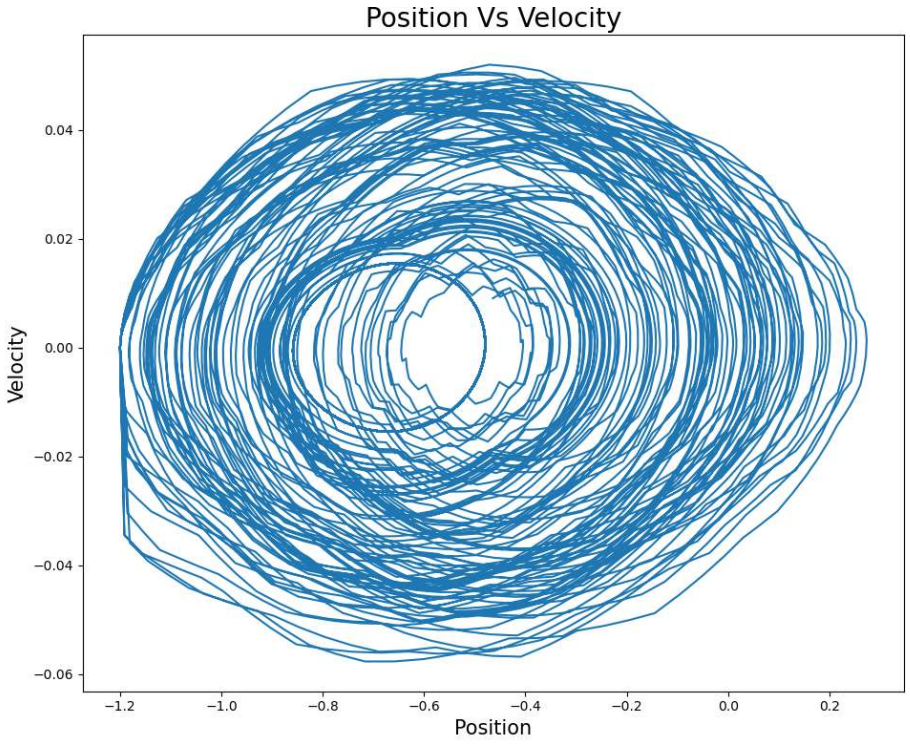
Value Function Plot for Order-7



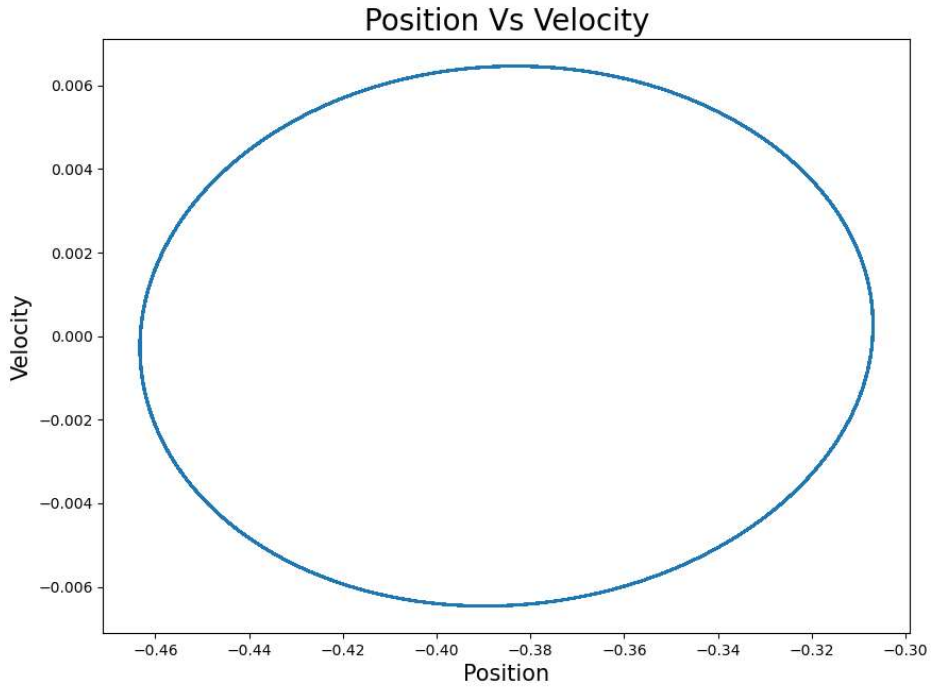
Discussion

The mountain car contains a negative step reward and a zero-goal reward. What would happen if $\gamma < 1$ and the solution was many steps long? What would happen if we had a zero step cost and a positive goal reward, for the case where $\gamma = 1$ and the case where $\gamma < 1$.

γ, the discount factor, says future rewards aren't as important as the immediate rewards. $\gamma = 1$ uses the future rewards the most. $\gamma < 1$, then, would use the future rewards less, which would mean that almost all learning predominantly would be local. If the solution was many steps longs, that might pose a risk of slow learning or potentially never reaching the goal. With $\gamma = 0.5$, the agent gets stuck in a loop as shown by the figure below. This is the first episode after about 300,000 steps:



If we had a zero step cost and a positive reward cost, the agent would simply be satisfied in its performance and merrily go about never learning. It could accidentally fall closer to the goal as the starting state and find the treasure but the probability of that happening is pretty low. Basically, the agent would learn nothing regardless of the value of γ, but $\gamma < 1$ will make things worse. The agent would get stuck in a loop as shown by the figure below:



Conclusion

Mountain car is a Reinforcement Learning solution to the problem of an underpowered car stuck in a valley. The only way for the car to make it up is for it to move backwards to build some inertia to help it against gravity. In real world problems, we don't always have a system model. In such cases, we try to approximate the value function for the system. To do this, I used the Fourier Bases introduced in *Value Function Approximation in Reinforcement Learning Using the Fourier Basis* (Konidaris).

Using the correct parameters, the RL agent was able to learn the trick in just a few episodes. For order 3 Fourier Bases, it took about 20 episodes for the steps per episode to reach a steady value. For order 5 and 7 Fourier Bases, it took about 18 episodes. The final steps per episode to reach the goal state was about 200 steps. There wasn't a major advantage of using higher order Fourier Bases for this problem.

The value function plots showed differences for different orders. Order 7 had a lot more sinusoidal waves than order 3. These plots represent negative of the actual value function for plotting purposes.

Citations

Konidaris, Goerge, et al. *Value Function Approximation in Reinforcement Learning Using the Fourier Basis*, https://people.cs.umass.edu/~pthomas/papers/Konidaris2011a.pdf.

Sutton, Richard S., et al. *Reinforcement Learning: An Introduction*. MIT Press Ltd, 2018.