Section 2

2.1

In the optimal policy, states 2 and 3 have values 0.0. This is because they are never visited during the rollouts we performed starting from the initial state. MC Policy Evaluation creates the estimates of each state's valye by averaging the returns observed when the state appears in an episode, so if a state is never encountered its value will default to 0. Under the optimal policy, the agent's path reaches the +1 terminal state and avoids the risky regions near the -1 terminal state, so this path doesn't pass through states 2 ad 3. Thus, these states never get visited during sampling and receive values of 0.0.

2.2

In the new strucure, the terminal state at index 12 was introduced but its corresponding reward is 0. TODO?? expression doesnt change??

The values remain almost the same because introducing the terminal state only changes when the ±1 reward is received, not the overall return structure of the MDP. In the previous MDP, for states $s \in {11,7}$ we had $V(s)=\mathbb{E}[r + \gamma V(s')]=\mathbb{E}[r]$ since termination implied $V(s')=0$, whereas now $V(s)=\mathbb{E}[r + \gamma V(12)]$ and since $V(12)=0$, this either remains $\mathbb{E}[r]$ or becomes $\gamma \mathbb{E}[r]$ if the reward is delivered one step later on the transition to state 12.

Section 3

3.1

[\overline{VE}(w) = \mathbb{E}_\pi \left[\left(\sum_i \gamma^i R_i - \hat{V}(S,w) \right)^2 \right]]

Let
[G = \sum_i \gamma^i R_i] as Monte Carlo return.

Then,

[\overline{VE}(w) = \mathbb{E}_\pi \left[(G - \hat{V}(S,w))^2 \right]]

Take gradient:

[\nabla_w \overline{VE}(w) = \mathbb{E}_\pi \left[2 (G - \hat{V}(S,w)) \nabla_w (G - \hat{V}(S,w)) \right]]

[= \mathbb{E}_\pi \left[2 (G - \hat{V}(S,w)) (- \nabla_w \hat{V}(S,w)) \right]]

Now,

[\nabla_w \hat{V}(S,w) = \nabla_w \left(\sum_{i=0}^{k} w_i \psi_i(S) \right)]

[= \nabla_w \left(w^T \phi(S) \right)]

[= \phi(S)]

Therefore,

[\nabla_w \overline{VE}(w) = \mathbb{E}_\pi \left[2 (G - \hat{V}(S,w)) (-\phi(S)) \right]]

For stochastic gradient descent (update single sample):

[w_{t+1} = w_t - \alpha \nabla_w \overline{VE}(w_t)]

[w_{t+1} = w_t - \alpha \left[-2 (G - \hat{V}(S,w_t)) \phi(S) \right]]

[w_{t+1} = w_t + 2\alpha (G - \hat{V}(S,w_t)) \phi(S)]

Absorb constant 2 into learning rate:

[w_{t+1} = w_t + \alpha (G - \hat{V}(S,w_t)) \phi(S)]

3.2

The bottom right states now have values because with the feature based linear approximation, it uses shared weights across states so updating from any visited state will indirectly update the estimated value of all other states, which would include the states that aren't visited. In the previous MC example, those states that were never visited didn't get updated because there is no connection between the states with the indices. With the feature vectors, the states share structure so weight updates are generalized.

3.3

Again, because of the states sharing of features, the weights get pulled in multiple directions at once. The -1 state has x position = 4, but so does the +1 state, so the weights have to be balanced to satisfy both of those states being represented, which means the -1 state doesn't get represented perfectly and gets dragged up towards 0. Essentially the learned weights have to fit all the states.

3.4

If using just the one hot vectors for the state indices, then this is the same as the standard Monte Carlo method since each state's update is independent which is like a separate running average of the returns per state. Essentially, the weight being updated would be just the weight corresponding to that state since the weights aren't being shared/generalized in the same manner.

Section 6

6.1

The plots showed that the relationship between the Q values and state variables is nonlinear. But linear approximation requires a linear relationship, so the model cannot capture the true Q function or learn the optimal policy with only the vanilla features.

6.2

I represented the state with [\phi_s = \big[1,; \text{pos},; \text{vel},; \text{angle},; \text{angle_vel},; \text{angle}^2,; \text{angle_vel}^2,; \text{angle} \cdot \text{angle_vel},; \text{pos} \cdot \text{angle},;

\text{vel} \cdot \text{angle_vel},; \big]] This adds quadratic and interaction terms to the raw observations.

6.3

From the 3d plots, we see curved parabolic surfaces with respect to the state variables, so I thought that the Q function has a quadratic dependence on the features. Squaring each variable angle and angle_vel as well as adding an interaction between the two most important variables of angle-angular velocity and position-angle could make the linear model able to approximate the curves.