COMP 765 - Assignment 1

Jad Wehbeh - jad.wehbeh@mail.mcgill.ca

February 7 2020

Question 1

We begin by defining the state vector

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & \dot{\theta} & \theta \end{bmatrix}^\top. \tag{1}$$

Around $\theta^* = \pi$, we have

$$\sin(\theta) \approx \pi - \theta, \tag{2}$$
$$\cos(\theta) \approx -1. \tag{3}$$

The equations of motion therefore simplify to

$$\ddot{x} = \frac{2ml\dot{\theta}^2(\pi - \theta) - 3mg(\pi - \theta) + 4(u - b\dot{x})}{4(M + m) - 3m}, \tag{4}$$
$$\ddot{\theta} = \frac{-3(-ml\dot{\theta}^2(\pi - \theta) + 2((M + m)g(\pi - \theta) - (u - b\dot{x}))}{l(4(M + m) - 3m)}. \tag{5}$$

Calculating the state and input jacobians, we obtain the LQR matrices

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-4b}{4M+m} & 0 & \frac{3mg}{4M+m} \\ 0 & \frac{-3b}{l(4M+m)} & 0 & \frac{6(M+m)g}{l(4M+m)} \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{6}$$

$$\mathbf{B} = \begin{bmatrix} 0 \\ \frac{4}{4M+m} \\ 0 \\ \frac{3}{l(4M+m)} \end{bmatrix}. \tag{7}$$

The actual magnitudes of the chosen $\mathbf{Q}$ and $\mathbf{R}$ matrices are unimportant, with only the ratio of magnitudes affecting performance. Since we are provided with no incentive to minimize control input, we choose a diagonal $\mathbf{Q}$ that equally weighs all states 100 times more than the control input. The obtained system performs well in the region of the linearization but fails to flip the pendulum upwards, as is to be expected. In order to achieve swing-up, the algorithm would have to be modified to use ILQR or another such nonlinear control algorithm.

Question 2

The obtained solution is capable of stabilizing the pendulum within a very small range of equilibrium, and performs better at keeping the pendulum upright than returning the cart to the origin. The cart often drifts sideways until it leaves the discretization area, causing the pendulum to fall. The proposed solution uses value iteration with a prediction model based on the linearized dynamics, a fixed time step, and a snap-to-nearest discretization. The reward function used rewards any state sufficiently close to an upright pendulum position based on distance to the desired goal.
This approach faced multiple issues during development and in its final iteration. Several hours were wasted on attempting to resolve issues relating to the evolution of asymmetric policies, such as the one seen in 1 which were caused by a bug in a comparison and the choice to use negative indices for invalid predictions.

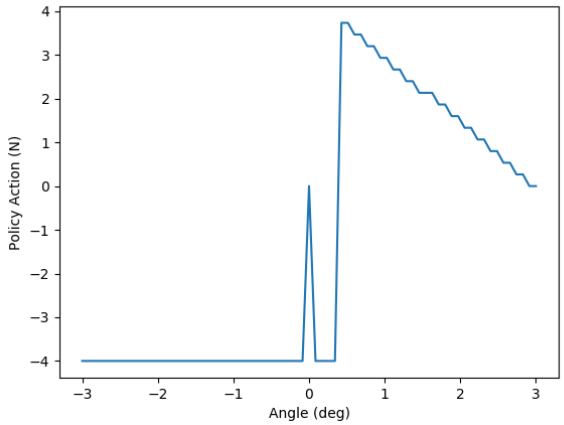

Figure 1: Early policy around equilbirum x, v, and $\dot{\theta}$ values

More recent policies perform significantly better, but still suffer from a number of issues, possibly due to the limited exploration space available. The policy of Figure 2 succeeds in achieving some level of stability for the system, but still is not completely symmetric as expected. This is likely due to the policy selection method used opting for the leftmost extreme action when all actions lead to equivalent values, which can happen around discretization limits.

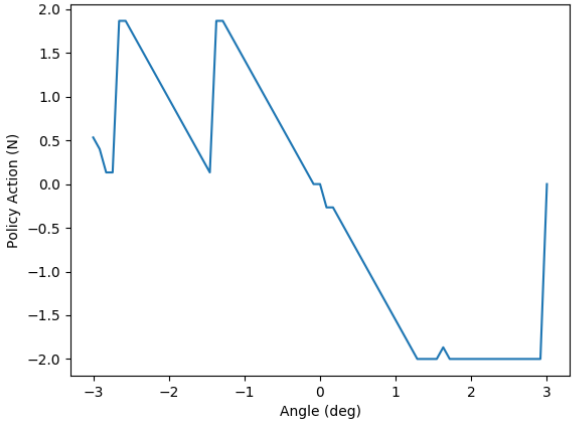

Figure 2: Recent policy around equilbirum x, v, and $\dot{\theta}$ values

Attempted approaches:

- Discretizing θ non-uniformly so as to sample a wider range of angles while maintaining precision around equilibrium: Did not succeed in expanding stabilization range as larger angles required wider sampling of other states, which was computationally prohibitive.
- Replacing snap-to-nearest discretization by probabilistic prediction model: Possibly improved performance and helped with reward propagation but was too computationally expensive for use.
- Modifying prediction time-step used during value iteration: Smaller values seem to improve performance in the neighborhood of the equilibrium but encourage input saturation. Larger values produce plots which look "smoother" but fail to achieve equilibrium.
- Modifying reward function to give a reward to any state with upright θ, with the reward diminishing based on state distance to goal: Helped maintain stability and better propagate rewards. Slightly deteriorated convergence to $x = 0$ and $v = 0$ but was still adopted.

Possible improvements to achieve swing-up:

- Replace linear prediction by nonlinear prediction.
- Increase sampling ranges and run on more powerful hardware.
- Use a probabilistic prediction model.
- Re-examine the idea of non-uniform discretization.
- Optimize value iteration by selectively updating array entries.

Time spent:

- LQR Linearization: 1 hr
- LQR Implementation: 1 hr
- Value Iteration Implementation: 3 hrs
- Value Iteration Modifications: 10 hrs
- Value Iteration Testing: 5 hrs
- **Total**: 20 hrs

Value iteration runs were performed on a laptop computer, with each of the approximately 20 runs attempted taking anywhere between 1 and 3 hours based on parameters chosen.