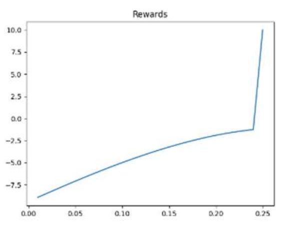
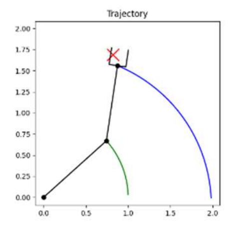
Optimal Control and Reinforcement Learning Assignment 1

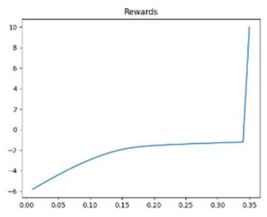
Student ID : 2016145015

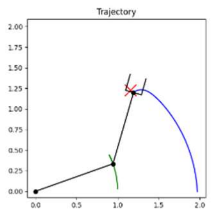
Name : Doohyun Lee

First, I tried to use Inverse kinematics. So, I calculated theta21d and theta1d using rd, alphad, arm1, arm2. Theta21d = $\cos^{-1}(\frac{x_d{}^2 + y_d{}^2 - arm1^2 - arm2^2}{2*arm1*amr2})$, theta1d = $\tan^{-1}(\frac{y_d}{x_d}) - \tan^{-1}(\frac{\text{arm2}*\sin(theta21d)}{arm1+arm2*\cos(theta21d)})$. Then I set obs_high as np.array([np.pi / 2 , np.pi /2, np.pi / 2, np.pi /2]). Finally I used N2 norm to get reward function like np.linalg.norm([self.theta1d – self.theta1 , self.theta21d – self.theta21]). However, this attempt did not properly learn, so the result trajectory was not drawn well. As a result, I decided to change only reward slightly, using original state-space. When l < tol, I changed a reward from 1 to 10, otherwise I changed a reward from $-l^2$ to $-(l+1)^2$ because when l < 1, l^2 makes minus reward more smaller and I thought it would be bad for learning. The figure below is result.



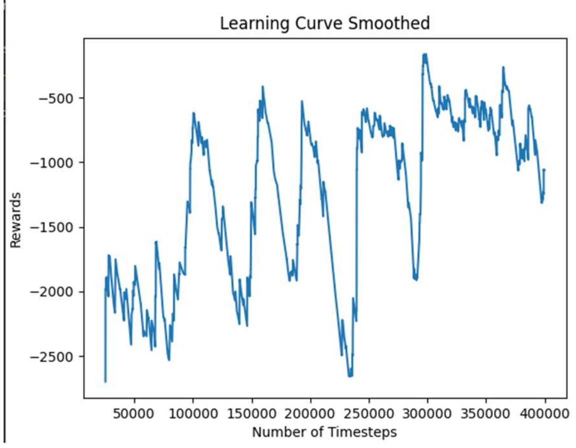






As you can see, trajectories are smooth and well drawn. Also, it didn't take long time to finish.

Also, I used a custom callback called "SavebestCallback" to store model with the highest mean reward during 400,000 timesteps. And the figure below is graph of timesteps and rewards.



So, I saved 300,000th timesteps model and I tried the env.render(True) with this model.