

# ECE 276 Assignment 1 : Classical Control

October 7, 2020

In this assignment, you will implement and tune a trajectory following controller for a 2-degree of freedom(DoF) robotic arm and a race car. The most common structure is the gym environment used by OpenAI(Read through <https://gym.openai.com/docs/> the page for more details.)

**Note:** Before starting the assignment, make sure you have downloaded and installed the necessary packages for setting up the environment.

## Question 1 - 2 DoF Arm

To import the environment use the following command:

```
import gym
import pybulletgym.envs
env = gym.make("ReacherPyBulletEnv-v0")
```

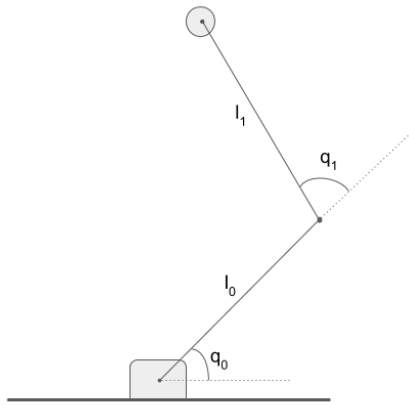


Figure 1: 2 DoF Arm

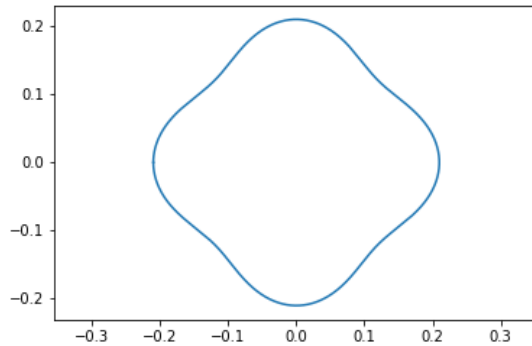


Figure 2: Arm Trajectory: The  $(x, y)$  point is given by  $x = (0.19 + 0.02\cos(4\theta))\cos(\theta)$ ,  $y = (0.19 + 0.02\cos(4\theta))\sin(\theta)$  for  $\theta \in [-\pi, \pi]$

The action space of environment is  $[\tau_{q_0}, \tau_{q_1}]$ , where  $\tau_{q_0}$  and  $\tau_{q_1}$  are the joint torques to joint  $q_0$  and  $q_1$  respectively. The links  $l_0$  and  $l_1$  are 0.1m and 0.11m respectively. Rather than using the observation space of the environment, use the following class methods to get the states of the robot.

```
env = gym.make("ReacherPyBulletEnv-v0")
```

```

# To set joint q0 to a particular position
env.unwrapped.robot.central_joint.reset_position(position,0)
# To set joint q1 to a particular position
env.unwrapped.robot.elbow_joint.reset_position(position,0)
# To get the current position and angular velocity of q0
q0,q0_dot = env.unwrapped.robot.central_joint.current_position()
# To get the current position and angular velocity of q1
q1,q1_dot = env.unwrapped.robot.elbow_joint.current_position()

```

1. In your report, derive the forward model for the robot as a single, closed-form expression expressed in joint angles and link length. Write a function *getForwardModel(q0,q1)* that takes the joint states and returns the end-effector position.
2. In your report, derive the expression for the Jacobian of the robot. Write a function *getJacobian(q0,q1)* that takes the joint states and returns the Jacobian.
3. Implement a closed loop PD-controller that controls the robot along the given trajectory (Figure 2) using the error in the end-effector as the input signal (keep  $v_{ref} = 0$ ). Plot the trajectory of the robot juxtaposed over the required trajectory and calculate the mean square error between the paths. In your report, describe the algorithm used to calculate the inverse.
4. Write out an inverse kinematic solution to the robot that uses the methods from (1) and (2). Also, derive the analytical inverse kinematic solution; explain what challenges there would be to use the analytical IK solution to track trajectories.
5. Implement a closed loop PD-controller that controls the robot along the given trajectory (Figure 2) using the error in the joint-angles as the input signal (keep  $\dot{q}_{ref} = 0$ ). Plot the trajectory of the robot juxtaposed over the actual trajectory and calculate the mean square error between both paths. Mention the gains used.

## Question 2 - Race Car

The objective of this environment is to make the race car travel as far as possible on a track within the given time. There are 3 tracks available - FigureEight, Linear and Circle, each having its corresponding horizon length. To set up an environment with a particular track, you can pass the track name while instantiating the environment. For example, to set up the figure eight trajectory :

```

from racecar.SDRaceCar import SDRaceCar
env = SDRaceCar(render_env=True, track='FigureEight')

```

The action space of the environment consists of [wheel angle, thrust]. The range of both these inputs are normalized to be between  $\pm 1$ . The observation space consists of  $[x, y, \theta, v_x, v_y, \dot{\theta}, h]$ , where  $(x, y, \theta)$  is the inertial frame position of the car and  $v_x, v_y$  is the longitudinal and lateral velocities respectively, and  $\dot{\theta}$  is the angular rotation of the car.  $h$  is the co-ordinate

on the track the car has to reach. Using these observations, come up with any controller that follows each of the three tracks and reaches as far as possible on the track.

1. Describe the controller used for following the trajectories. You may use different gains for different tracks.
2. Plot the paths of the robot juxtaposed over the desired trajectory.

Note, intentionally, no dynamical model of the car is given, you do not necessarily need it to write a closed-loop controller.