ECE 276C Assignment 1 Report: Classical Control

Jingpei Lu
Jacob School of Engineering
University of California, San Diego

Date: October 14, 2019

1 2 DoF Arm

Our 2 DoF arm has two joint and two links. The central joint has joint angle q_0 and the elbow joint has the joint angle q_1. The link l_0 is from central joint to elbow joint, and the link l_1 is from elbow joint to the end-effector. The length of the links are fixed, where $l_0 = 0.1m$ and $l_1 = 0.11m$. The end-effector position is $[x, y]^T \in \mathcal{R}^2$.

1.1 Forward kinematics

Forward kinematics for a robot arm involves figuring out a function that takes as its inputs the angles of each joint and computes the end-effector position. The mapping from joint angles to the end-effector position of our 2 DoF arm is as follow:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_0 \cos(q_0) + l_1 \cos(q_0 + q_1) \\ l_0 \sin(q_0) + l_1 \sin(q_0 + q_1) \end{bmatrix} \tag{1}$$

1.2 Jacobian Matrix

Using the Jacobian matrix, we can describe the relationship between the joint velocities and the end-effector velocities as $\delta x = J(q)\delta q$, where $q = [q_0, q_1]^T \in \mathcal{R}^2$. For our 2 DoF arm, the Jacobian is as follow:

$$J(q) = \begin{bmatrix} -\sin(q_0)l_0 - \sin(q_0 + q_1)l_1 & -\sin(q_0 + q_1)l_1 \\ \cos(q_0)l_0 + \cos(q_0 + q_1)l_1 & \cos(q_0 + q_1)l_1 \end{bmatrix} \tag{2}$$

1.3 Inverse kinematics

Inverse kinematics for a robot arm entails finding a function which takes the target position of the end-effector as an input and computes the join angles q_0 and q_1 which put the end-effector of the arm at the target position. The algorithm of calculating the inverse kinematics is as follow:

Algorithm 1 Find Inverse kinematics

```
1: procedure GETIK($x_e, y_e$)        ▷ $x_e, y_e$ are coordinates of target end-effector position
2:     Set the error tolerance $e_t = 1e-5$
3:     Get current joint angles $q = [q_0, q_1]^T$
4:     Get current end-effector position $[x_t, y_t]^T$
5:     while $||x_e - x_t, y_e - y_t|| > e_t$ do
6:         Calculate the Jacobian matrix $J$ for current joint angles $q$
7:         Calculate the pseudoinverse of the jacobian $J^\dagger$
8:         $\delta q \leftarrow J^\dagger [x_e - x_t, y_e - y_t]^T$
9:         $q \leftarrow q + \delta q$
10:        $[x_t, y_t]^T \leftarrow$ calculate forward kinematics from updated $q$
11:    return $q$
```

1.4 PD-controller with End-effector Position Error

To controls the robot arm moving along the given path, I sampled 60 points along the path and implemented trajectory control. For the PD-controller, I was using the error in the end-effector as the input signal. The gain for the proportional term is $K_p = 0.0005$ and the gain for the the derivative term is $K_d = 0.00001$. The resulting trajectory is on the figure below, and the mean square error between the resulting trajectory and desired trajectory is 0.00019.

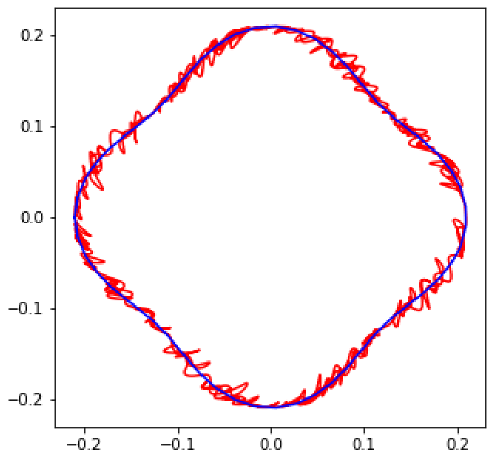

Figure 1: This figure shows the desired trajectory (blue) and the resulting trajectory after each step (red).

1.5 PD-controller with Joint-Angles Error

This time, I also divied the given path to 60 small trajectory while using the error in the joint-angles as the input signal for the PD-controller. The gains for different joints are different. For the proportional term, the gain for the central joint is $K_{p0} = 0.001$ and the gain for the elbow joint is $K_{p1} = 0.002$. For the derivative term, the gain for the central joint is $K_{d0} = 1e-5$ and the gain for the elbow joint is $K_{d1} = 1e-5$. The resulting trajectory is on the figure below, and the mean square error between the resulting trajectory and desired trajectory is 0.00021.

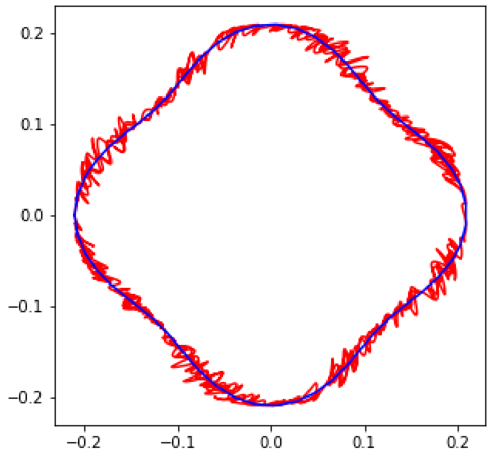

Figure 2: This figure shows the desired trajectory (blue) and the resulting trajectory after each step (red).

2 Race Car

The objective of this environment is to make the race car complete a trajectory within the given time. The action space of the environment is $[\delta_f, a]$, where δ_f is wheel steering angles and a is the thrust. They are all normalized to the range between -1 and 1. The observation space consists of $[x, y, \theta, v_x, v_y, \dot{\theta}, h]$, where (x, y, θ) 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. $h \in \mathcal{R}^2$ is the co-ordinate on the track the car has to reach.

The algorithm of computing the action from given observation is described in the algorithm 2, and the resulting trajectories are shown in the figure 3. For the performance, this controller takes 380 steps to finish the "FigureEight" and 196 steps to finish the "Circle" by setting $max\_speed\_threshold$ to 8, $min\_speed\_threshold$ to 6, $K_{p,f}$ to 1, and $K_{p,a}$ to 1/20.

Algorithm 2 Race Car Controller

```
1:  procedure GET-ACTION($x, y, \theta, v_x, v_y, \dot{\theta}, h$)
2:      $\dot{\phi} \leftarrow$ angles between moving direction of the car and the direction from the position of the car $[x, y]$ to target position $h$
3:      $||v|| \leftarrow \sqrt{v_x^2 + v_y^2}$
4:      $\beta \leftarrow \arcsin(\frac{l_r}{||v||}\dot{\phi})$                    ▷ $l_r$ is the distance from the center of the mass to the rear axles
5:      $\delta_f \leftarrow \arctan(\tan(\beta)\frac{l_r+l_f}{l_r})$             ▷ $l_f$ is the distance from the center of the mass to the front axles
6:      $\delta_f \leftarrow K_{p,f} * \delta_f \frac{||v||}{\pi/2}$                    ▷ normalize to [-1,1] and proportional to $||v||$
7:      $d \leftarrow$ distance between the coordinates of the car and target coordinates
8:      $a \leftarrow K_{p,a} * d$ and normalize to [-1,1]
9:      if $||v|| > max\_speed\_threshold$ then $a \leftarrow -1$                ▷ no thrust if too fast
10:     if $||v|| < min\_speed\_threshold$ then $a \leftarrow 1$                 ▷ full thrust if too slow
11:     return $[\delta_f, a]$
```

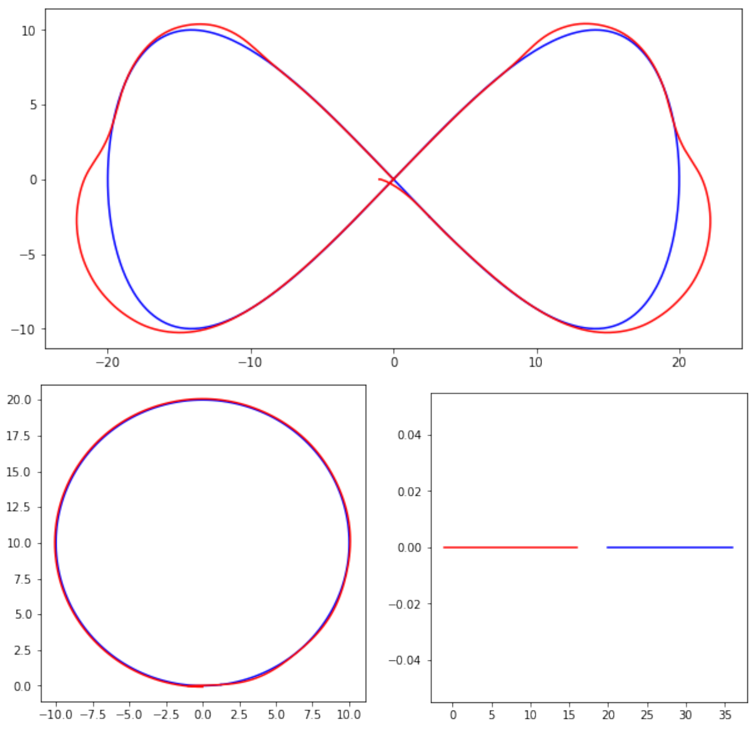

Figure 3: The resulting trajectories (red) and the desired trajectories (blue) in different environments. Top plot: "FigureEight". Bottom left: "Circle". Bottom right: "Linear".