Rushi Bhvaesh Shah
rs7236
rushi.shah@nyu.edu

ROB-GY 6323 Reinforcement Learning and Optimal Control for Robotics Project 1 Report

Introduction

Finding control for a dynamical system over a period of time so that an objective function is optimized is the focus of the branch of mathematics known as optimal control theory. It has a wide range of uses in operations research, engineering, and science. The goal of optimal control theory is to operate dynamic systems as cheaply as possible. The situation in which a set of linear differential equations explain the system dynamics and a quadratic function define the cost is known as the linear quadratic issue. The Linear-Quadratic Regulator provides the solution, which is one of the theory's primary findings (LQR).

Goal of the project

The goal of this project is to control a 2D quadrotor to perform acrobatic moves. There are 4 parts of the project, where you will build controllers of increasing complexity. The last part will lead to the implementation of the iterative LQR (iLQR) algorithm.

Questions:

Part 1 - Setting up

1. Discretize the system dynamics using the method seen in class - write the time discretization step as Δt (use symbols not numbers for the mass, etc)

$$\begin{aligned}
\dot{x} &= v_x \\
m\dot{v}_x &= -(u_1 + u_2)\sin\theta \\
\dot{y} &= v_y \\
m\dot{v}_y &= (u_1 + u_2)\cos\theta - mg \\
\dot{\theta} &= \omega \\
I\dot{\omega} &= r(u_1 - u_2)
\end{aligned}$$

Ans: From the above equations, we can write $\dot{v_x} = \ddot{x}$, $\dot{v_y} = \ddot{y}$ and $\dot{\omega} = \ddot{\theta}$ as :

$$\dot{v_x} = \ddot{x} = -\frac{(u_1 + u_2)\sin\theta}{m} \tag{a}$$

$$\dot{v_y} = \ddot{y} = \frac{(u_1 + u_2)\cos\theta - mg}{m} \tag{b}$$

$$\dot{\omega} = \ddot{\theta} = \frac{r}{I}(u_1 - u_2) \tag{c}$$

Hence, further the above given quadrotor model can be discretized as follows:

$$x_{t+1} = x_t + \dot{x}\Delta t \tag{1.1}$$

$$\begin{aligned}
\dot{x}_{t+1} &= \dot{x}_t + \ddot{x}\Delta t \\
\dot{x}_{t+1} &= \dot{x}_t - \frac{(u_1 + u_2)\sin\theta}{m}\Delta t
\end{aligned} \tag{1.2}$$

$$y_{t+1} = y_t + \dot{y}\Delta t \tag{1.3}$$

$$\dot{y}_{t+1} = \dot{y}_t + \ddot{y}\Delta t$$
$$\dot{y}_{t+1} = \dot{y}_t + \frac{(u_1 + u_2)\cos\theta - mg}{m}\Delta t \tag{1.4}$$

$$\theta_{t+1} = \theta_t + \dot{\theta}\Delta t \tag{1.5}$$

$$\dot{\theta}_{t+1} = \dot{\theta}_t + \ddot{\theta}\Delta t$$
$$\dot{\theta}_{t+1} = \dot{\theta}_t + \frac{r}{I}(u_1 - u_2)\Delta t \tag{1.6}$$

2. Assume that the robot starts at an arbitrary position $x(0) = x_0$, $y(0) = y_0$ and $\theta(0) = 0$ with 0 velocities. Compute u_1^* and u_2^* such that the robot stays at this position forever after.

Ans : To keep the system at rest, when we input all the given values in the question, from the equation (1.1), (1.2), (1.3), (1.4), (1.5) and (1.6), we get

$$\begin{aligned}
\ddot{x} &= 0 \\
\ddot{y} &= 0 \\
\ddot{\theta} &= 0 \\
u_1 + u_2 &= mg \\
u_1 - u_2 &= 0 \\
\implies u_1 = u_2 &= \frac{mg}{2} \\
\therefore u_1^* = u_2^* &= \frac{mg}{2} \qquad \textbf{(answer)}
\end{aligned}$$

Hence the optimal control inputs to hover the quadrotor in place (in absence of disturbance) are $u_1^* = u_2^* =$ mg/2. The **dummy_controller** stimulates the above behaviour in the code.

3. Analyzing the system dynamics, is it possible to move in the x direction while keeping $\theta = 0$? Explain why.

Ans : In the equation for $\dot{v}_x$, dependency on θ can be observed. Hence, if $\theta = 0$, then $\sin\theta = 0$ as well. Therefore, the drone will tend to have no change in it's state in the x-direction when $\theta = 0$.

4. Analyzing the system dynamics, is it possible to have the system at rest with $\theta = \frac{\pi}{2}$ (i.e. have the quadrotor in a vertical position)? Explain why.

Ans : When the θ is set to $\pi/2$, equation (b) from previous section gives $\ddot{y} = -g$ i.e. the acceleration due to gravity. This will result in drone not being able to maintain its position as $\theta = \pi/2$ will make the control terms (u_1 and u_2) as zero and hence allowing to control that position of the quadrotor.

Part 2 - LQR to stay in place

Now that we have u^* capable of keeping the robot at rest, we can design a simple controller that ensures that the robot stays in place even when pushed around by random disturbances (e.g. due to the wind). Our task here will be to design a LQR controller that keeps the robot at a predefined position. Since the dynamics is not linear, we need to compute a linear approximation of it.

1. Linearize the dynamics at an arbitrary operating point z^*, u^* and write the linearized system dynamics using the variables $\bar{z}_n = z_n - z^*$ and $\bar{u}_n = u_n - u^*$.

Ans : When a first order Taylor series expansion is computed about a point (z^*, u^*), we get

$$z_{n+1} = f(z^*, u^*) + \frac{\delta f}{\delta z}|_{z^*,u^*}(z - z^*) + \frac{\delta f}{\delta u}|_{z^*,u^*}(u - u^*)$$

$$\bar{z}_{n+1} = A\bar{z}_n + B\bar{u}_n$$

Here, A and B from above equations (from part 1) can be written as :

$$\text{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & (-\Delta t * (u_1 + u_2)\cos\theta)/m & 0 \\ 0 & 0 & 1 & \Delta t & (-\Delta t * (u_1 + u_2)\sin\theta)/m & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{B} = \begin{bmatrix} 0 & 0 \\ -\frac{\Delta t}{m}\sin\theta & -\frac{\Delta t}{m}\sin\theta \\ 0 & 0 \\ \frac{\Delta t}{m}\cos\theta & \frac{\Delta t}{m}\cos\theta \\ 0 & 0 \\ \frac{\Delta t}{I}r & -\frac{\Delta t}{I}r \end{bmatrix}$$

5. Explain your intended design in the report, including the cost function and found control law. In particular, verify that it can handle perturbations by calling the "'simulate"' function with "'disturbance = True"' (when setting disturbance to "True", the simulator will generate a random perturbation every 1 second). Simulate your controller for 10 seconds, plot the state evolution and show the animation (include the plots in your report).

Ans : The control equation used in this part is

$$u_n = K(z_n - z^*) + u^*$$

The cost function used is determined as :

$$\sum_{n=0}^{\infty} (\bar{z}_n^T \; Q \; z_n + \bar{u}_n^T \; R \; u_n)$$

Further, P and K are calculated as follows :

$$P = Q + A^T PA - A^T PB(B^T B + R)^{-1}(B^T PA)$$
$$K = -(B^T PB + R)^{-1} B^T PA$$

Here, we obtain the converged feedback gain K after solving the Riccati's equation by iterating during each of the 1000 iterations. The cost matrices Q and R are as follows :

$$\text{Q} = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}$$

$$\text{R} = \begin{bmatrix} 0.05 & 0 \\ 0 & 0.05 \end{bmatrix}$$

The equations solved at every iteration are :

$$K_n = -(B^T P_{n+1} B + R)^{-1} B^T P_{n+1} A \quad \text{and} \quad P_n = Q + A^T P_{n+1} A + A^T P_{n+1} B K_n$$

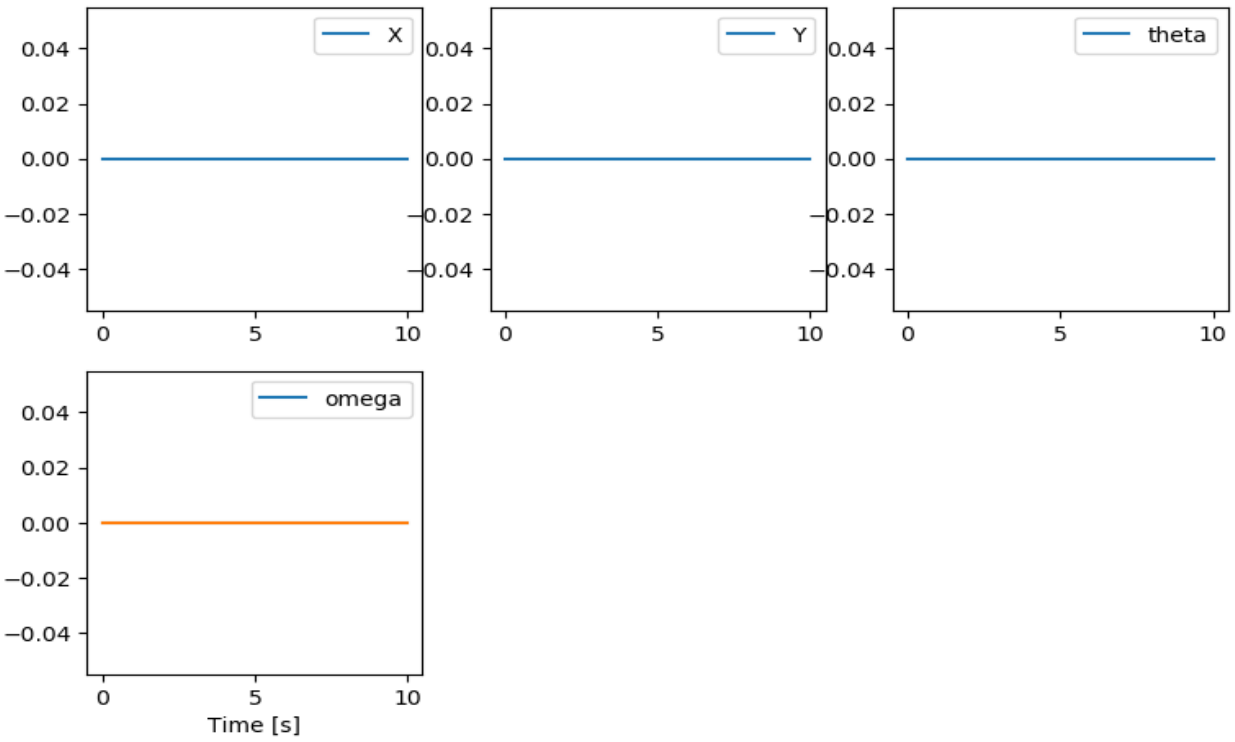

Figure 1: Plots without Disturbance

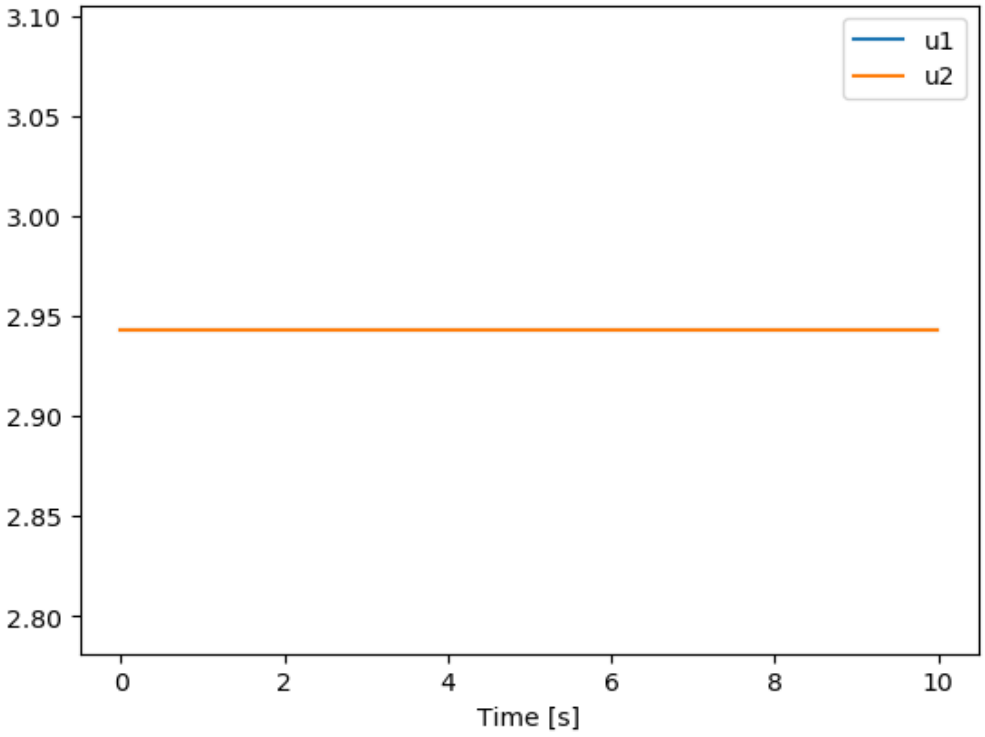

Figure 2: Plot without Disturbance

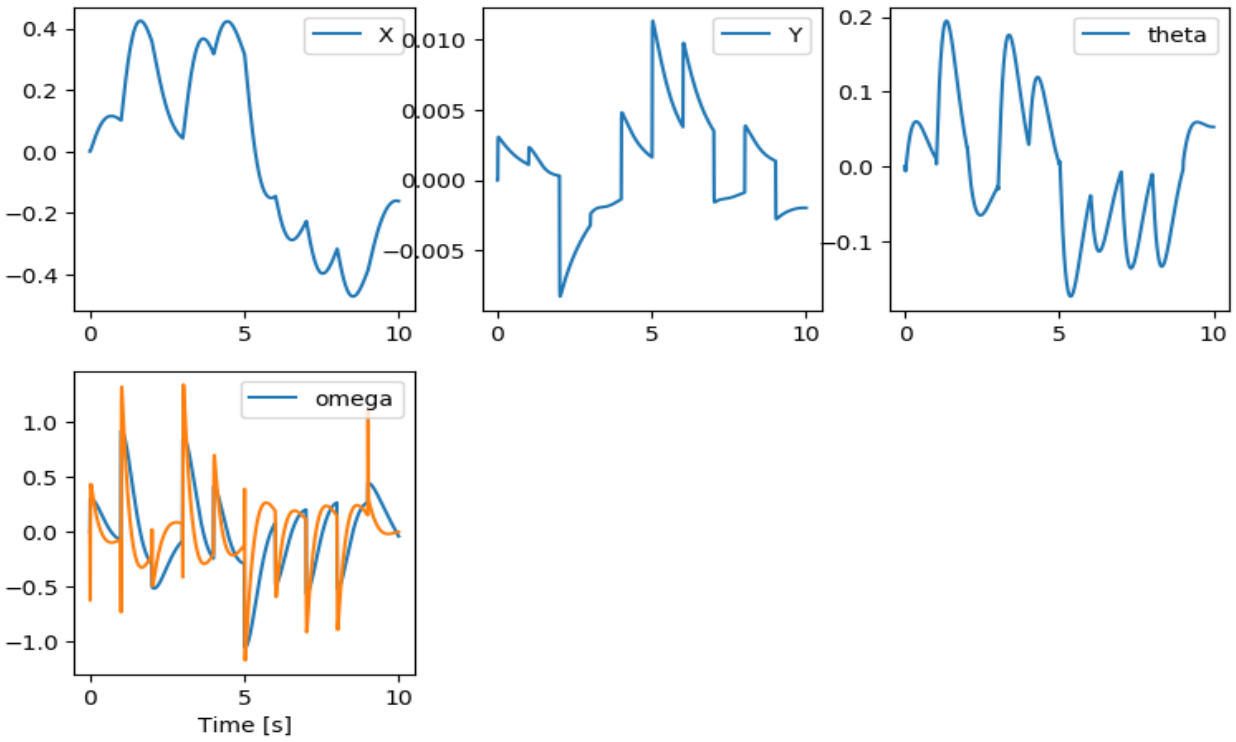

Figure 3: Plots with Disturbance

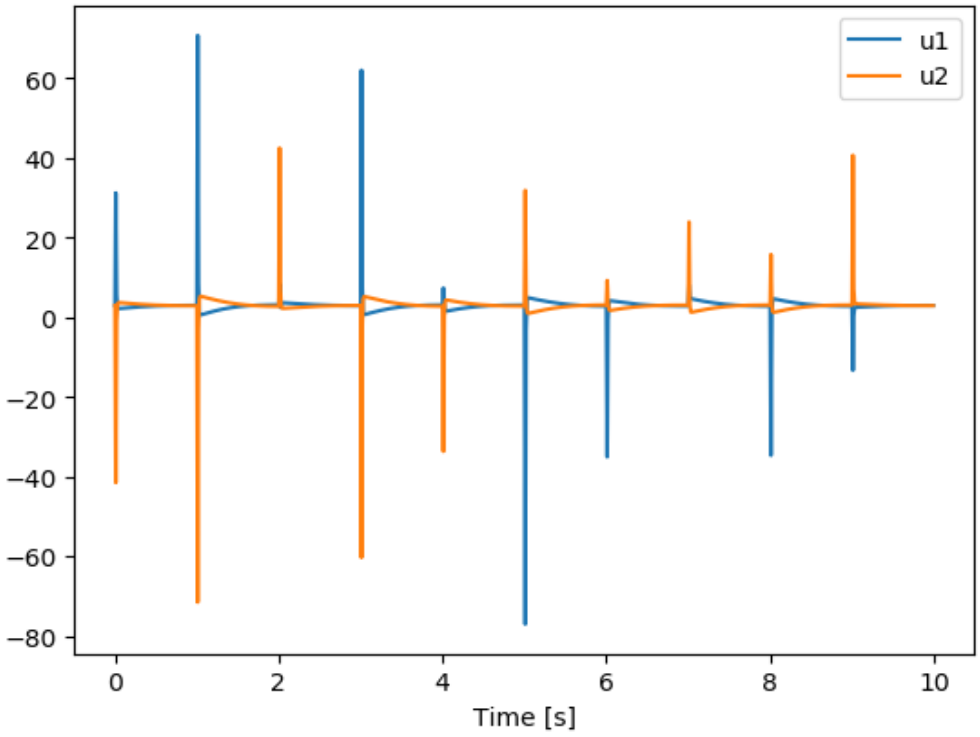

Figure 4: Plot with Disturbance

Part 3 - Following a trajectory using linearized dynamics

1. Assume that we want to follow a circle of radius 1 centered at (0,0) while keeping an orientation $\theta = 0$, how does the linearization of the dynamics change along the desired trajectory? Why?

Ans : For this task, linearization will be done for each set of (z^*, u^*) along the circle, i.e the trajectory to be followed in this case. It will be done using the equations:

$$\bar{z}_{n+1} = A_n \bar{z}_n + B_u \bar{u}_n$$
$$\text{where} \quad \bar{z}_n = z_n - z_n^* \quad \text{and } \bar{u} = u_n - u_n^*$$

Hence, A and B would be

$$\text{A} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & (-\Delta t * (u_1^* + u_2^*) \cos\theta_n^*)/m & 0 \\ 0 & 0 & 1 & \Delta t & (-\Delta t * (u_1^* + u_2^*) \sin\theta_n^*)/m & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\text{B} = \begin{bmatrix} 0 & 0 \\ -\frac{\Delta t}{m} \sin\theta_n^* & -\frac{\Delta t}{m} \sin\theta_n^* \\ 0 & 0 \\ \frac{\Delta t}{m} \cos\theta_n^* & \frac{\Delta t}{m} \cos\theta_n^* \\ 0 & 0 \\ \frac{\Delta t}{I} r & -\frac{\Delta t}{I} r \end{bmatrix}$$

The trajectory state points for N = 1000 iterations are written as

$$\bar{z} = \begin{bmatrix} \cos(-2\pi * i/1000) \\ 0 \\ \sin(-2\pi * i/1000) \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Similar to previous part, Q and R are defined.

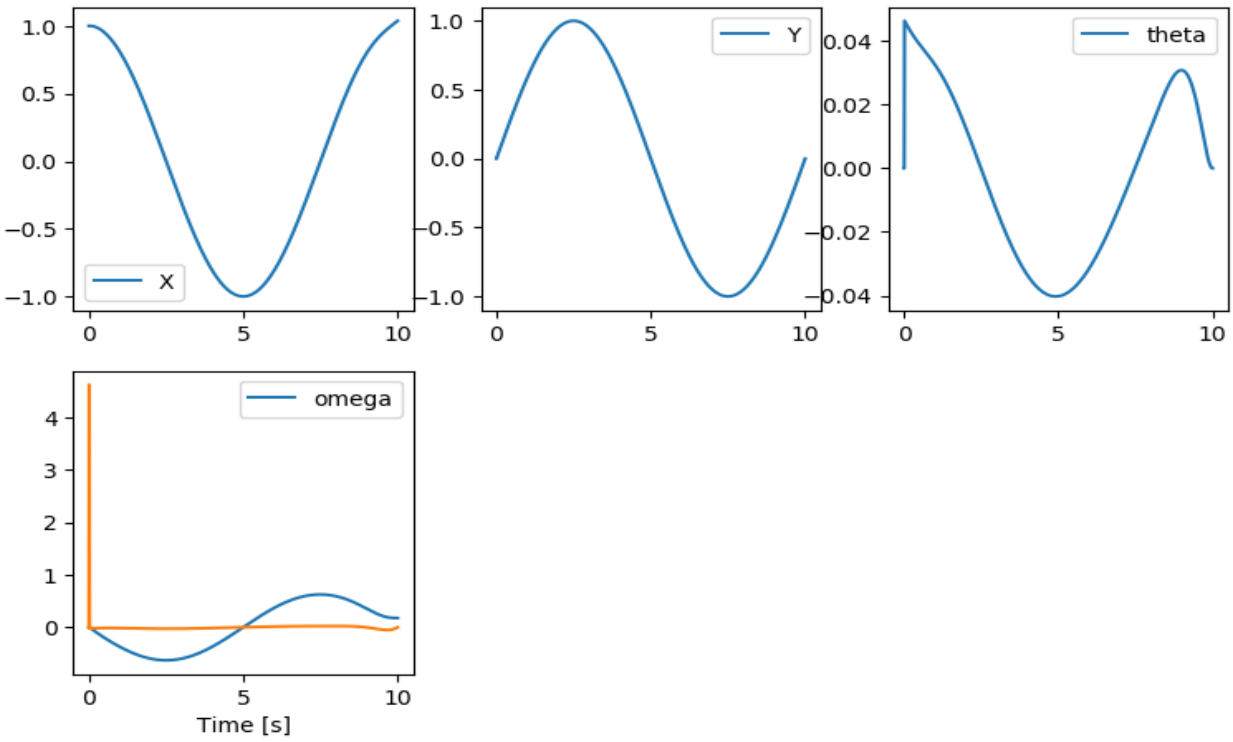

Figure 5: Plots without Disturbance

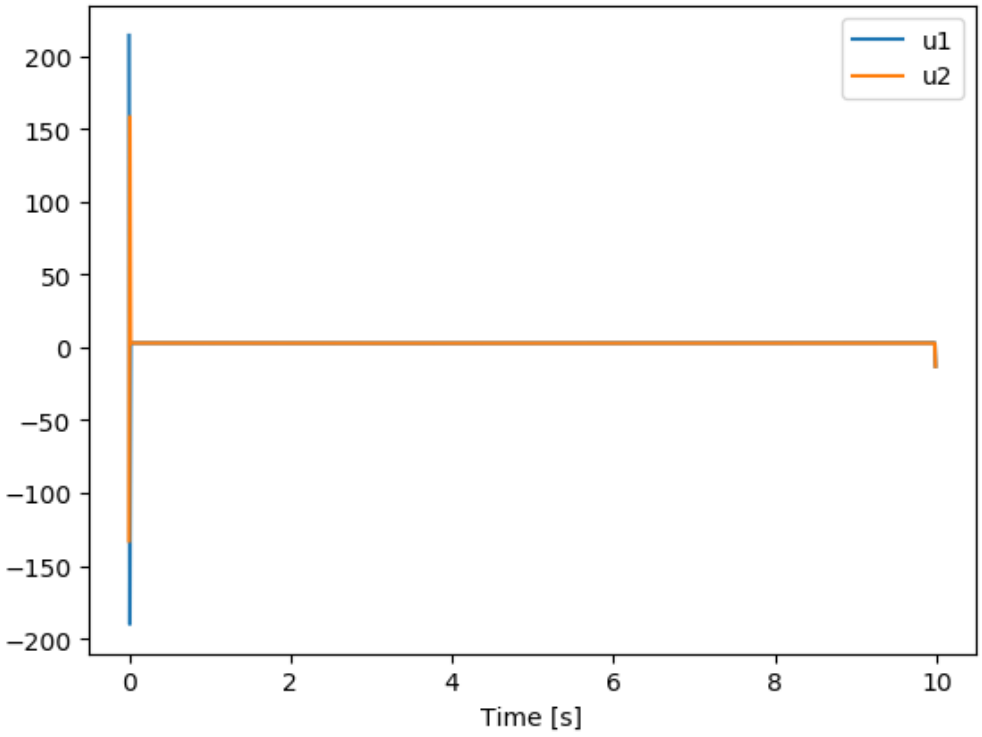

Figure 6: Plot without Disturbance

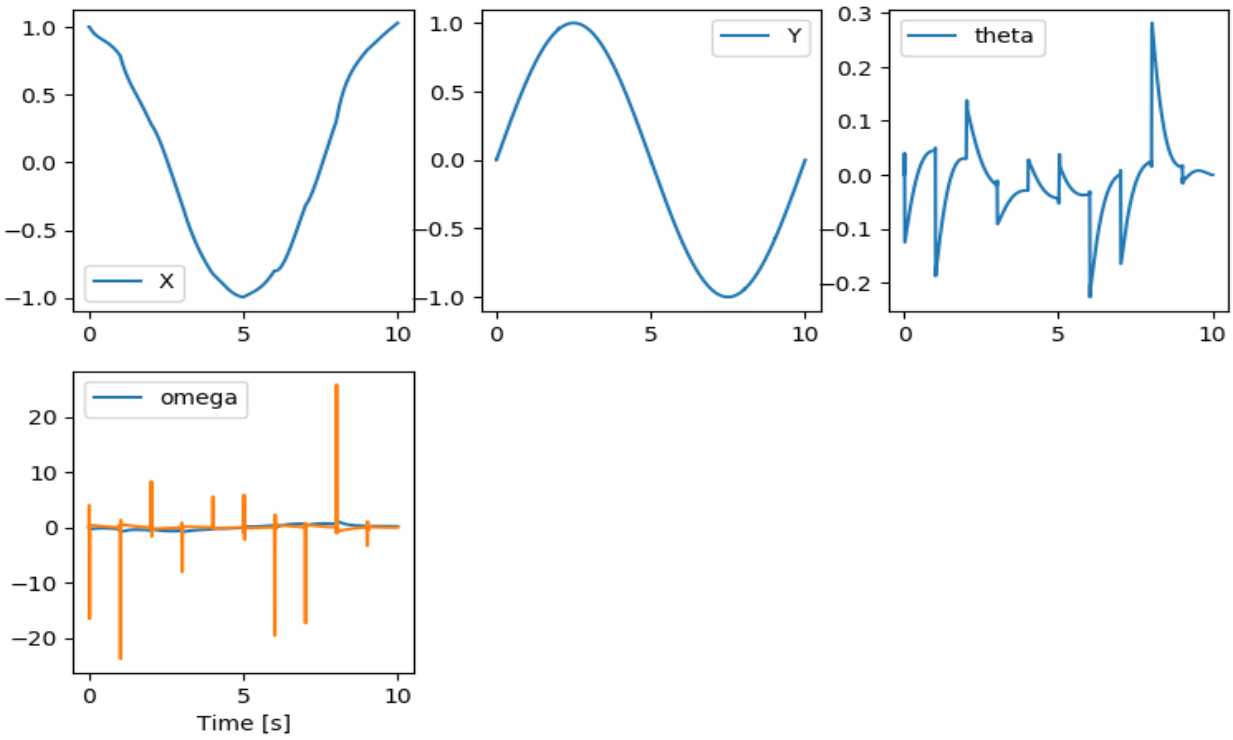

Figure 7: Plots with Disturbance

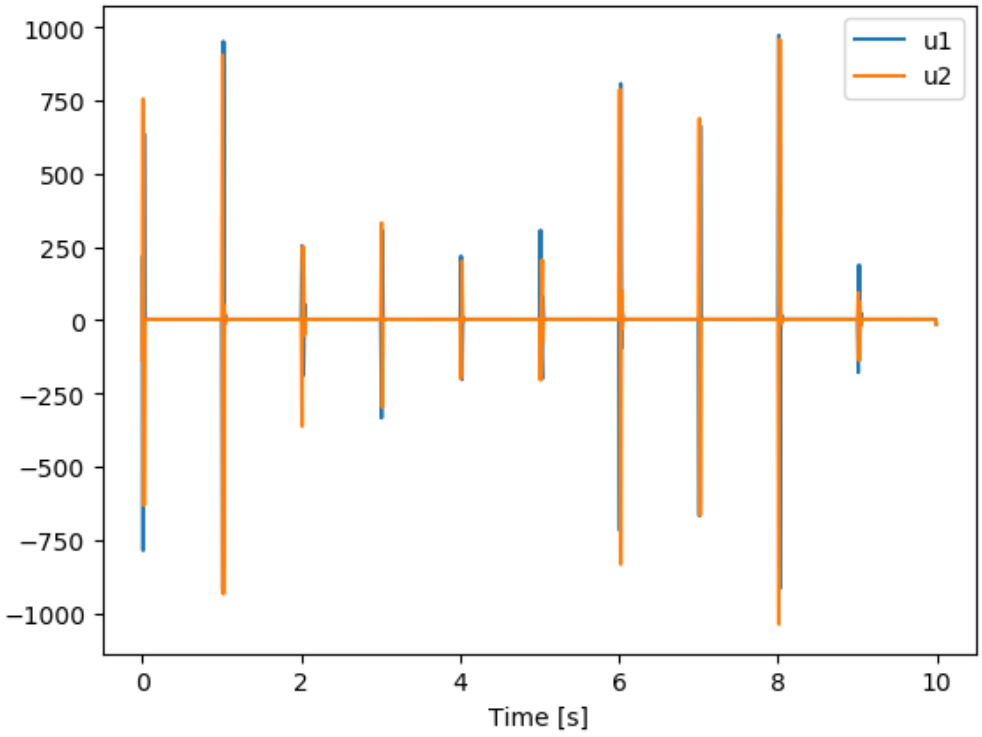

Figure 8: Plot with Disturbance

Part 4 - iterative LQR

0.1 Task 1

Q7 What benefits and issues do you see with this approach?
Benefits
1. The controller is able to track the trajectory for x,y very well.
2. Allows to design and control complex trajectories with robust behaviour

Drawbacks:
1. We need to solve the backward Riccati equations on every step. This can be slow for larger states and controls. So if this problem was in 3D then every control step will be very slow.
2. The problem above would be magnified even more if we used an infinite horizon controller for following a trajectory on every step

If we keep $\theta = \pi/4$ we will not be able to find a control trajectory from the dynamics, as there are 3 equations and two unknowns u1 and u2.

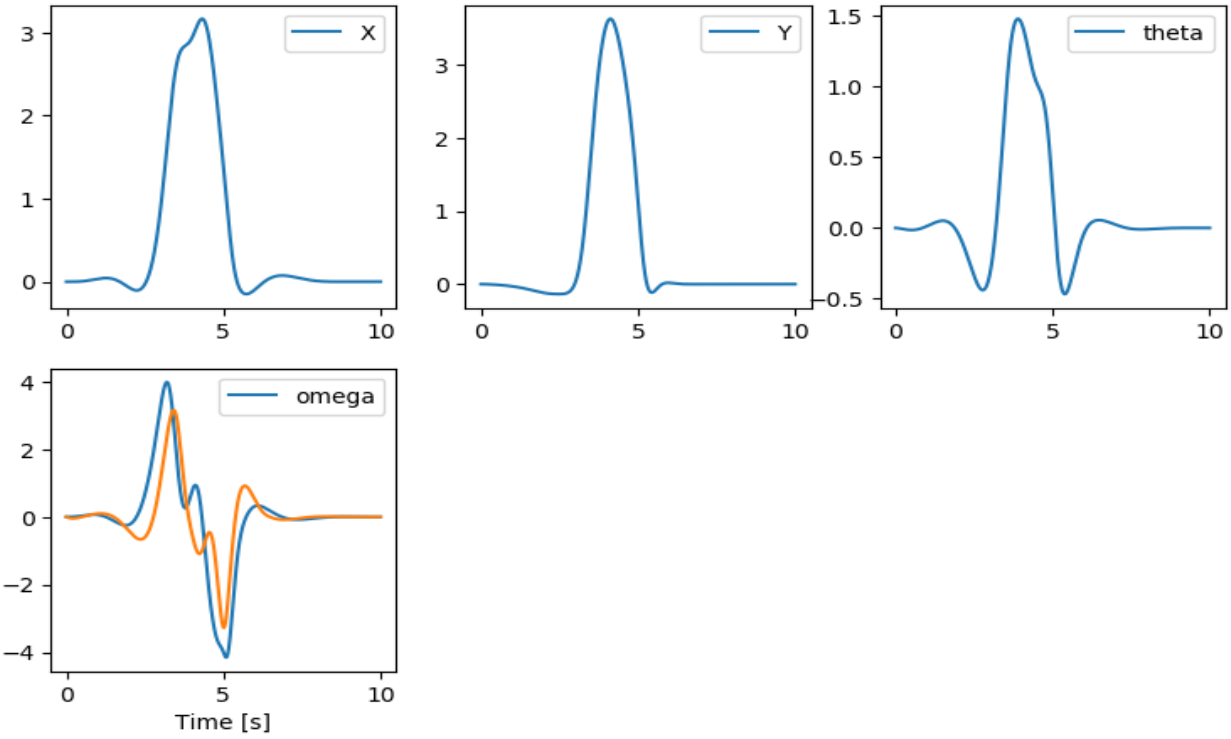

Figure 9: Task 1

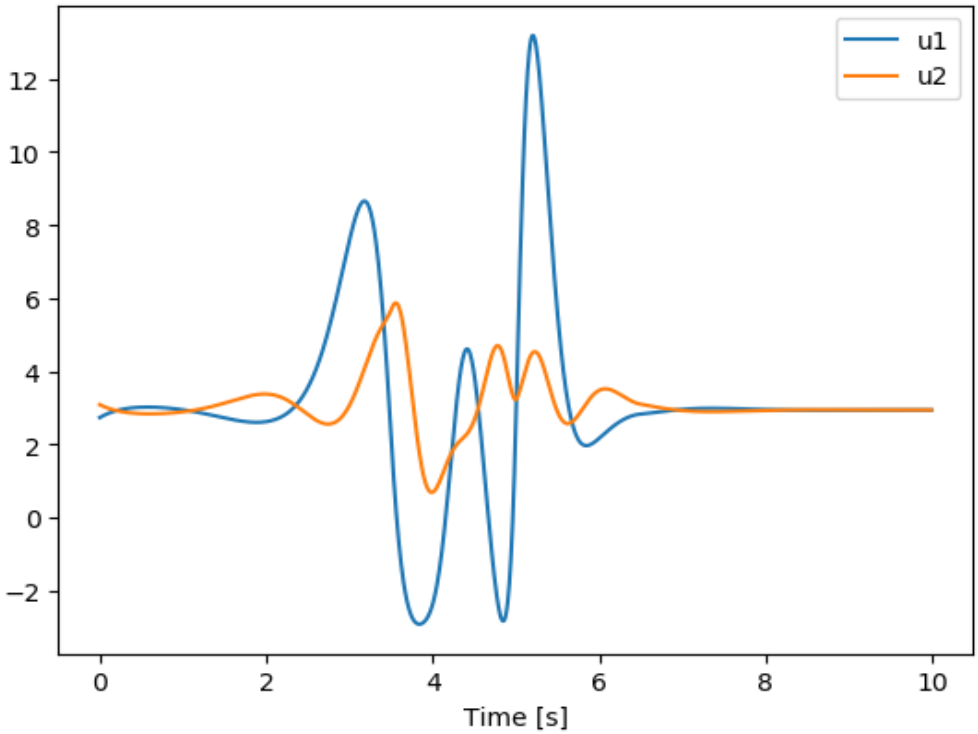

Figure 10: Task 1

0.2 Task 2

Q What benefits and issues do you see with this approach? Could you run the resulting controller on a real robot?

Benefits
1. Fast execution
2. Could find state and control trajectory

Drawbacks
1. Negative control is obtained at places which is not possible
2. Difficult to implement in 3D
3. Very fine tuning is needed

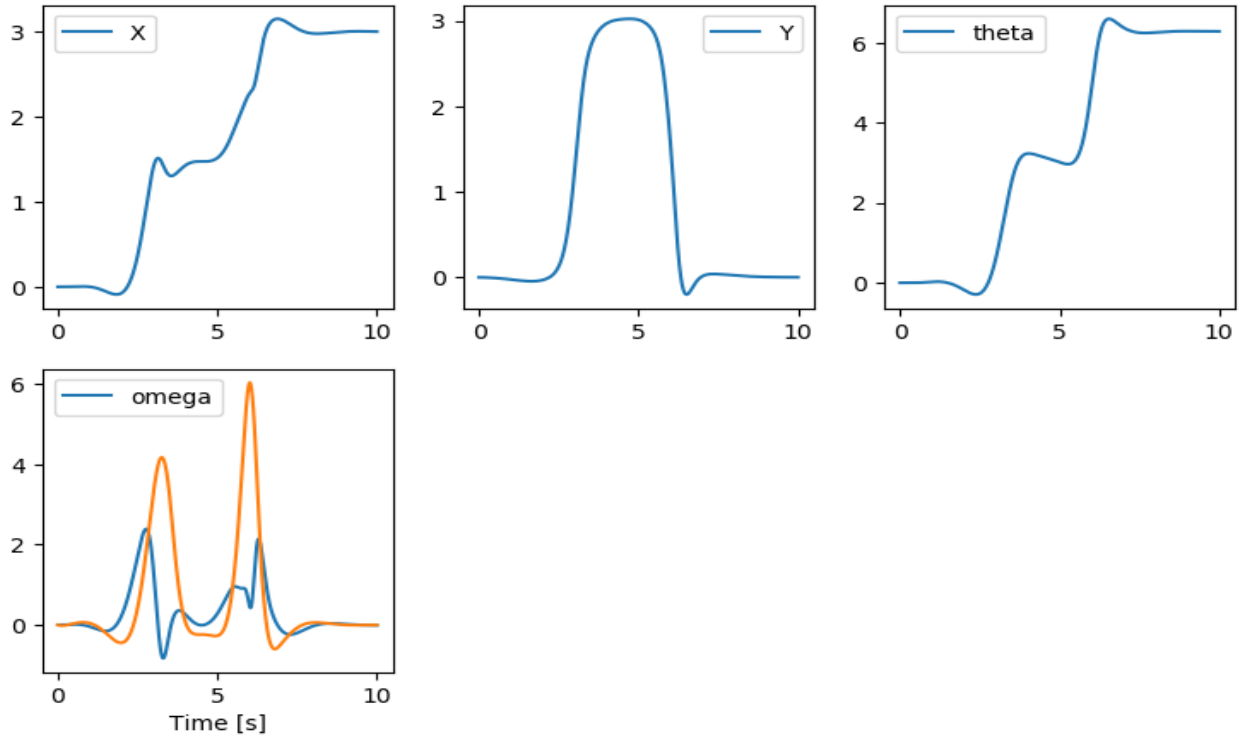

Figure 11: Task 2

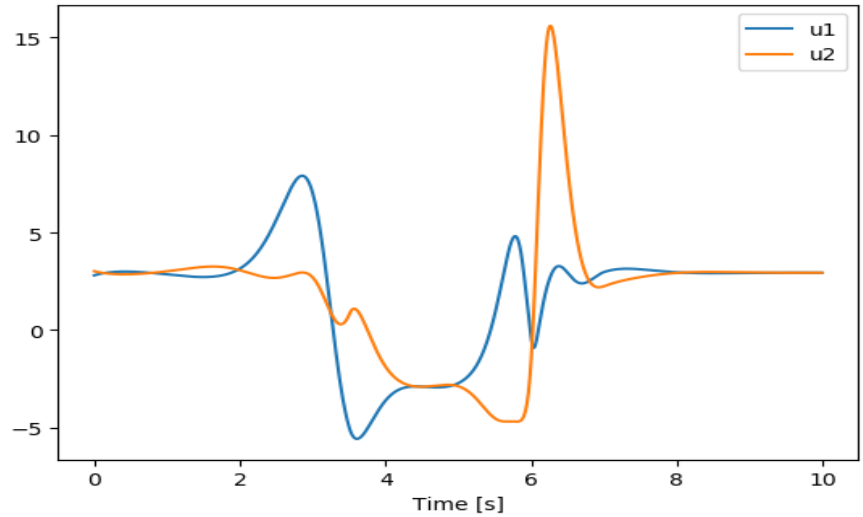

Figure 12: Task 2