

Project 1

Reinforcement learning and optimal control

Arjun Raja – ar6841

Part 1

1. To discretize the system, we find the derivative of the state \dot{z}

From Euler's approximation we have:

$$\dot{z} \approx \frac{z_{n+1} - z_n}{\Delta t}$$

$$z_{n+1} = z_n + \Delta t \dot{z}$$

$$z_{n+1} = f(z, u) = \begin{bmatrix} x + \Delta t v_x \\ v_x - \Delta t \left(\frac{u_1 + u_2}{m} \right) \sin \theta \\ y + \Delta t v_y \\ v_y + \Delta t \left[\left(\frac{u_1 + u_2}{m} \right) \cos \theta - g \right] \\ \theta + \Delta t \omega \\ \omega + \Delta t \frac{r}{I} (u_1 - u_2) \end{bmatrix}$$

2. Starting from an arbitrary position $x(0)=x_0$, $y(0)=y_0$ and $\theta(0)=0$. And we have,

$$v_x=0, v_y=0, \omega=0$$

Thus to stay at rest forever, $z_{n+1} = z_n$. This leads to equations from the system dynamics,

$$\begin{aligned} \dot{v}_x &= 0 \\ \dot{v}_y &= 0 \\ \dot{\omega} &= 0 \\ u_1 + u_2 &= mg \\ u_1 - u_2 &= 0 \\ u_1 = u_2 &= \frac{mg}{2} \end{aligned}$$

3. It is not possible to *change* the x direction while keeping $\theta=0$ as $\dot{v}_x = 0$ ($\sin 0 = 0$)
The drone will either hold the same x position or if there was a velocity before it will move in the x direction with constant velocity.
4. When $\theta = \frac{\pi}{2}$ $\dot{v}_y = -g$ So the y position and velocity cannot be controlled as there is a constant acceleration in y direction. Thus, the system cannot be at rest.

Part 2

1. About some point (z^*, u^*) , from the first order Taylor series expansion:

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

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

Where,

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

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

1. The controller follows the equation:

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

2. The cost function used is :

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

The LQR solution for infinite horizon is :

control law found : $u_n = K (z_n - z^*) + u^*$

$$K = \begin{bmatrix} 48.4474 & -48.4474 \\ 58.2126 & -58.2126 \\ -24.0417 & -24.0417 \\ -24.4604 & -24.4604 \\ -102.5956 & 102.5956 \\ -9.1299 & 9.1299 \end{bmatrix}$$

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

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

Where K is the converged feedback gain obtained after solving the Riccati recursion 1000 times, We solve the equations :

$$K_n = -(B^T P_{n+1} B + R)^{-1} B^T P_{n+1} A$$

$$P_n = Q + A^T P_{n+1} A + A^T P_{n+1} B K_n$$

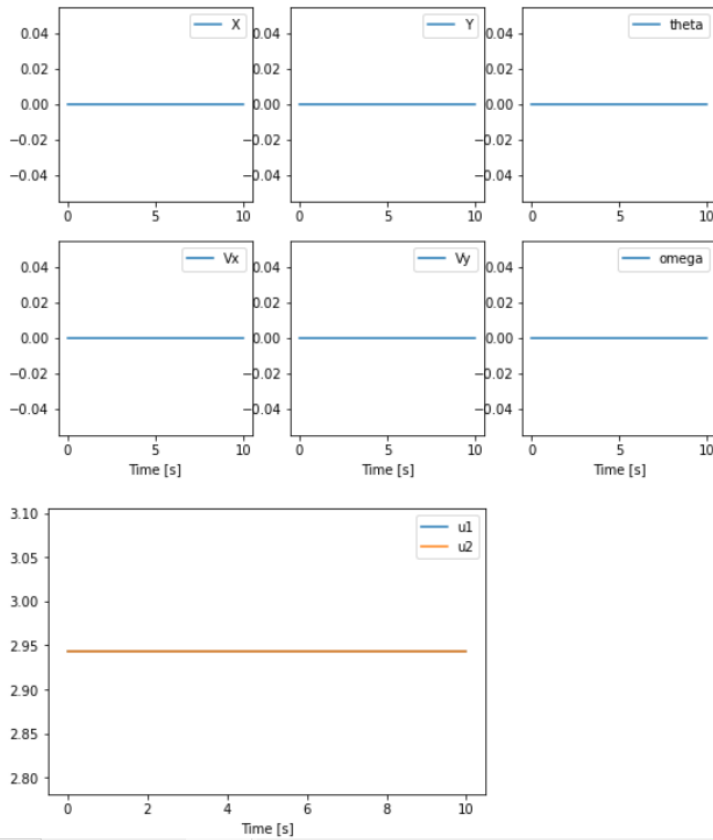
Till convergence of K_n

And the cost matrix,

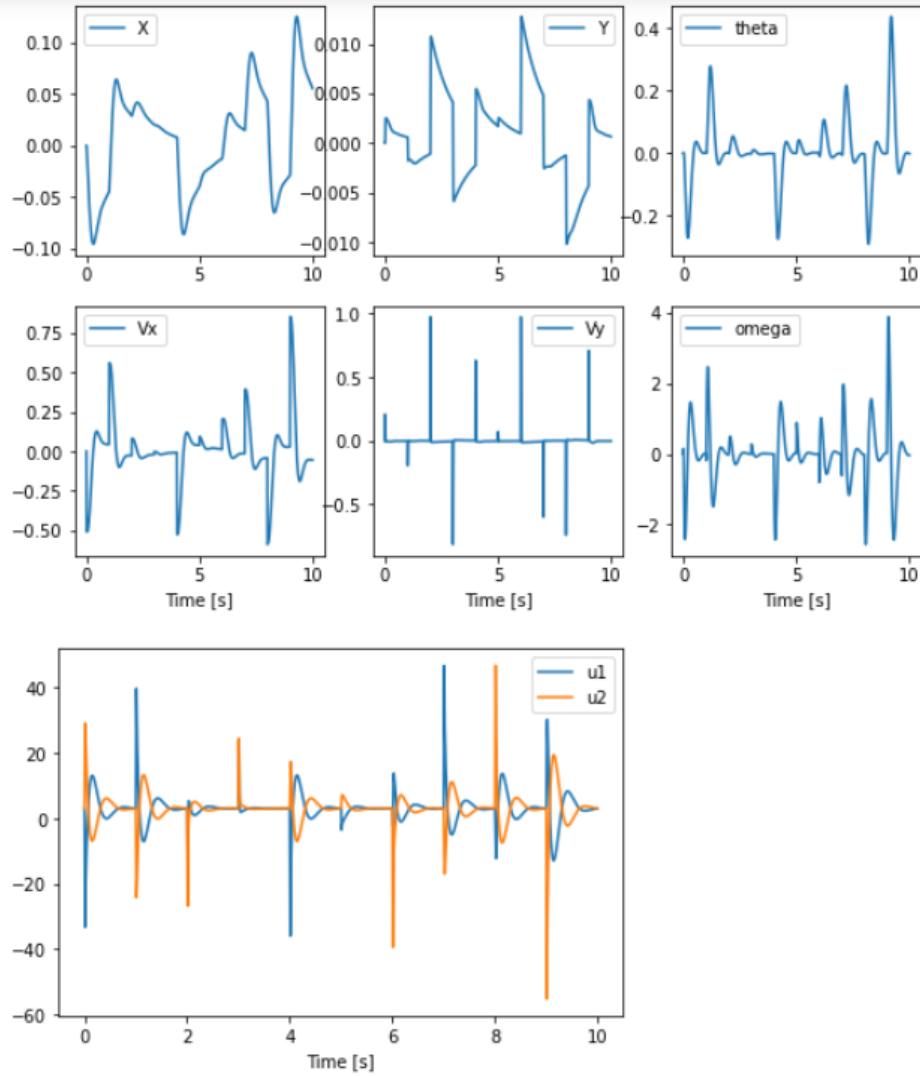
$$Q = \begin{bmatrix} 600 & 0 & 0 & 0 & 0 & 0 \\ 0 & 600 & 0 & 0 & 0 & 0 \\ 0 & 0 & 600 & 0 & 0 & 0 \\ 0 & 0 & 0 & 600 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

This was made to punish the deviation from $z = 0$ of the x, v_x, y, v_y terms of the system more than the θ, ω terms as we need to deviate from $\theta = 0, \omega = 0$ to control x position. Also, the R matrix is <1 so that control is not punished. The system can handle perturbations very well.

Plots (Disturbance is False):



Plots (Disturbance is true):



Part 3

1. We must now linearize about each new set of (z_n^*, u_n^*) along the trajectory points. The linearization of the dynamics will take the form :

$$\bar{z}_{n+1} = A_n \bar{z}_n + B_n \bar{u}_n$$

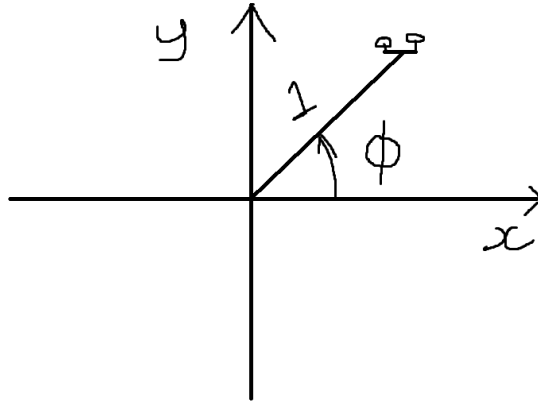
$$\bar{z}_n = z_n - z_n^* \text{ and } \bar{u}_n = u_n - u_n^*$$

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

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

Where each A_n and B_n must be found along the changing trajectory points.

When we are on a circle of radius 1. Let ϕ denote this angle:



Let $\phi = \frac{2\pi}{T}(t)$ where t is the current time (in seconds) and T is the total time taken to complete a circle

The trajectory state points will take the form :

$$z_n^* = \begin{bmatrix} \cos \phi \\ \frac{-2\pi}{T} \sin \phi \\ \sin \phi \\ \frac{2\pi}{T} \cos \phi \\ 0 \\ 0 \end{bmatrix}$$

From the dynamics we can find,

$$\dot{v}_y^* = -\frac{4\pi^2}{T^2} \sin \phi$$

$$m\dot{v}_y^* = u_1^* + u_2^* - mg$$

$$\dot{\omega}^* = u_1^* - u_2^* = 0$$

Thus the control points will take the form:

$$u_n^* = \begin{bmatrix} \frac{m}{2} \left(g - \frac{4\pi^2}{T^2} \sin \phi \right) \\ \frac{m}{2} \left(g - \frac{4\pi^2}{T^2} \sin \phi \right) \end{bmatrix}$$

Where T is the time taken (in seconds) to complete a full circle

2. The cost function used is :

$$\left(\sum_{n=0}^{N-1} (z_n - z_n^*)^T Q (z_n - z_n^*) + (u_n - u_n^*)^T R (u_n - u_n^*) \right) + (z_N - z_N^*)^T Q (z_N - z_N^*)$$

The tracking LQR solutions in this case are:

$$u_n = K_n z_n + k_n + u_n^*$$

Where we find K_n and k_n after solving the backward Riccati equations from $N - 1$ to 0

Initialize $P_N = Q_N$ $p_N = q_N$

$$q_n = -Q \ z_n^*$$

Iterate from $N - 1$ to 0

$$K_n = -(R_n + B_n^T P_{n+1} B_n)^{-1} B_n^T P_{n+1} A_n$$

$$P_n = Q_n + A_n^T P_{n+1} A_n + A_n^T P_{n+1} B_n K_n$$

$$k_n = -(R_n + B_n^T P_{n+1} B_n)^{-1} B_n^T p_{n+1}$$

$$p_n = q_n + A_n^T p_{n+1} + A_n^T P_{n+1} B_n k_n$$

The k_n term takes care of the feedforward required to track z_n^*

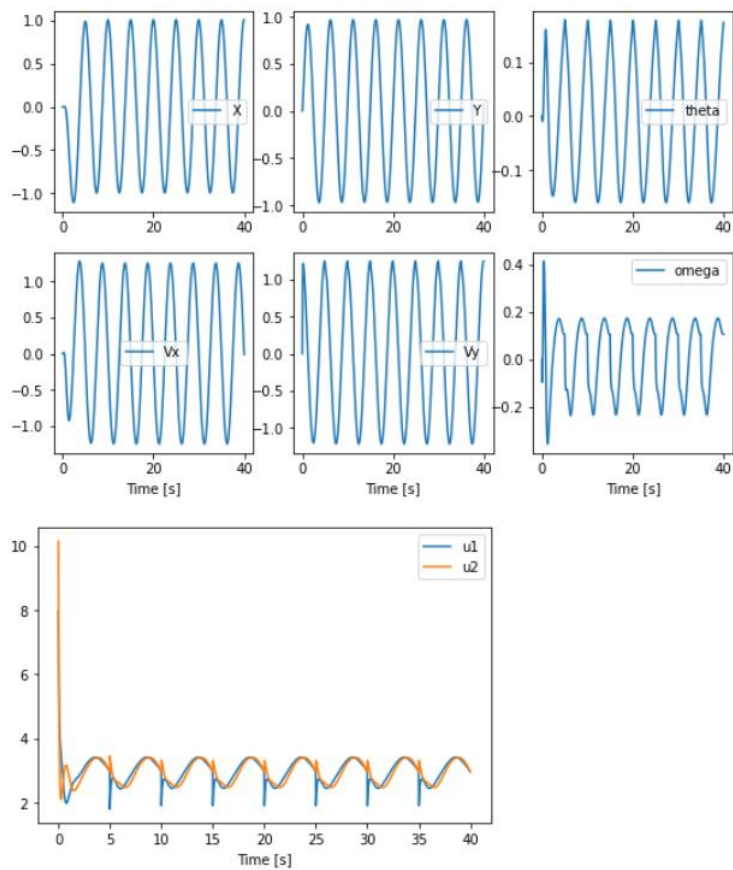
Similar to part 2 of the project, the cost matrixes are given below. The difference here is that we punish deviation from the control trajectory u_n^* as we want to stay on the circle

$$Q = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

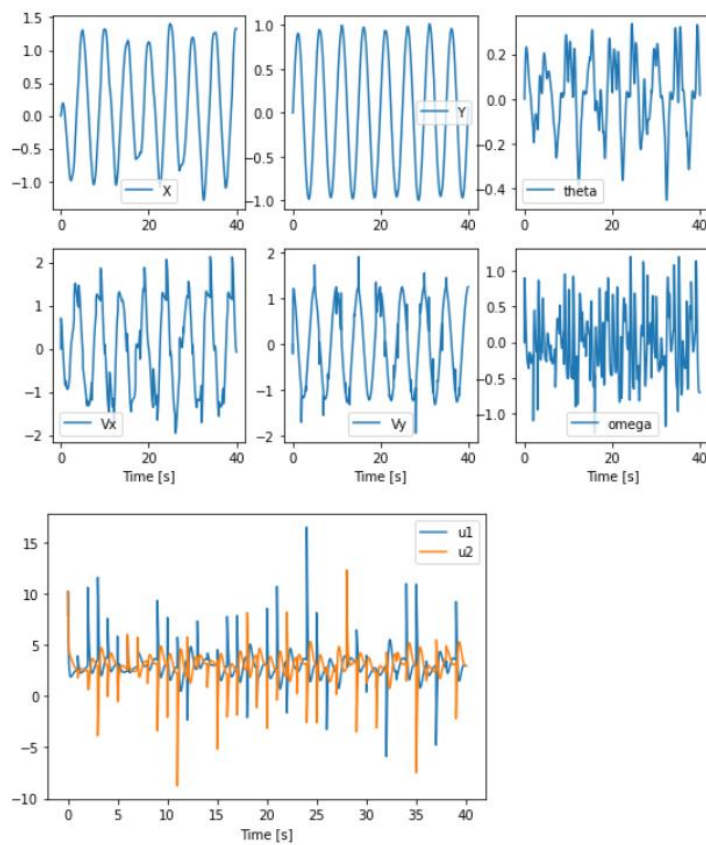
3. The θ **cannot** be tracked well as we need to move in the x direction to complete the circular trajectory. For this reason, the Q matrix does not punish deviations in θ as heavily.

4. Plots :

Disturbance off



Disturbance on:



The results show that the x, y and velocities are sinusoidal waves as expected. The initial omega is high as the quadrotor spawns at 0,0 and must reach the circle, so the controller changes the θ quickly to control the x and y locations. The controller also does a very good job at handling perturbations and returning to the required trajectory, although the x value deviates a bit as theta sharply changes due to wind disturbances.

The control input also follows sinusoidal waves but initially have to overcompensate to reach the circle.

Benefits:

- The controller is able to track the trajectory for x,y very well.
- Able to handle wind disturbances

Drawbacks:

- If the deviation from the trajectory is too large, the linearization does not hold true anymore and the controller will not be able to follow the trajectory
 - So if the radius of the circle was too big, the controller would never reach the circle from 0,0
 - We must solve the backward Riccati equations on every step. This can be slow for bigger states and controls. So if this problem was in 3D then every control step will be very slow.
 - The problem above would be magnified even more if we used an infinite horizon controller for following a trajectory on every step
5. If we keep $\theta = \frac{\pi}{4}$ we will not be able to find a control trajectory from the dynamics, as there are 3 equations and two unknowns u_1 and u_2 . To satisfy the state trajectory, we need to satisfy all the 3 equations:

$$\begin{aligned}m\dot{v}_x &= -(u_1 + u_2) \sin \theta \\m\dot{v}_y &= (u_1 + u_2) \cos \theta - mg \\I\dot{\omega} &= r(u_1 - u_2) = 0\end{aligned}$$

Which **will not be possible**.

Part 4

Task 1

1. Cost function used to promote such behaviour:

Handwritten cost function equation:

$$\begin{aligned}
 g(\mathbf{z}, \mathbf{u}) &= \mathbf{z}_N^T \mathbf{Q}_N \mathbf{z}_N + \sum_{n=0}^{499} \left(\mathbf{z}_n^T \mathbf{Q}_1 \mathbf{z}_n + \mathbf{u}_n^T \mathbf{R} \mathbf{u}_n \right) \\
 &\quad + \sum_{n=490}^{510} \left(\mathbf{z}_n - \tilde{\mathbf{z}} \right)^T \mathbf{Q}_2 \left(\mathbf{z}_n - \tilde{\mathbf{z}} \right) + \mathbf{u}_n^T \mathbf{R} \mathbf{u}_n \\
 &\quad + \sum_{n=511}^{999} \left(\mathbf{z}_n^T \mathbf{Q}_1 \mathbf{z}_n + \mathbf{u}_n^T \mathbf{R} \mathbf{u}_n \right)
 \end{aligned}$$

Where $\mathbf{Q}_1 =$

$$\begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}$$

$$\mathbf{Q}_2 = 100000 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{Q}_N = 100000 \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

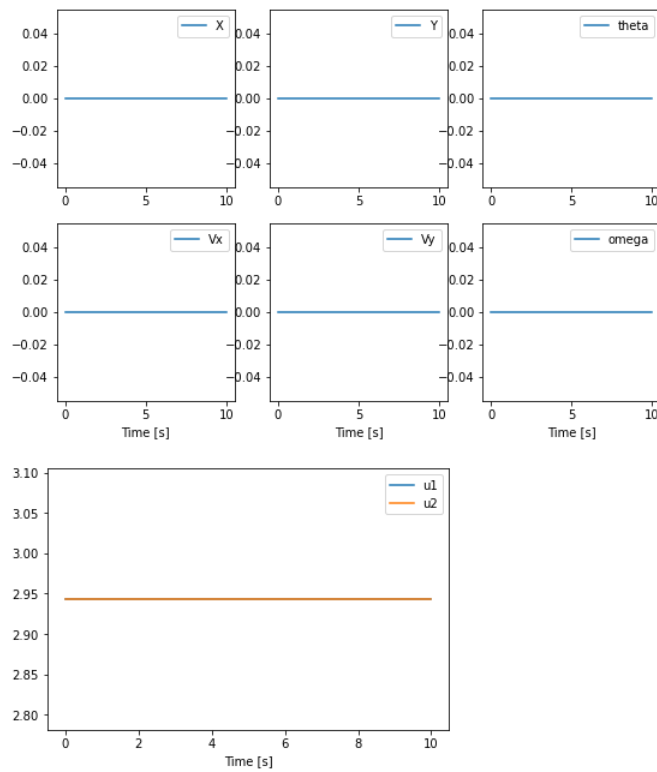
$$\mathbf{R} = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$\tilde{z} = \begin{bmatrix} 3 \\ 0 \\ 3 \\ 0 \\ \pi \\ 2 \\ 0 \end{bmatrix}$$

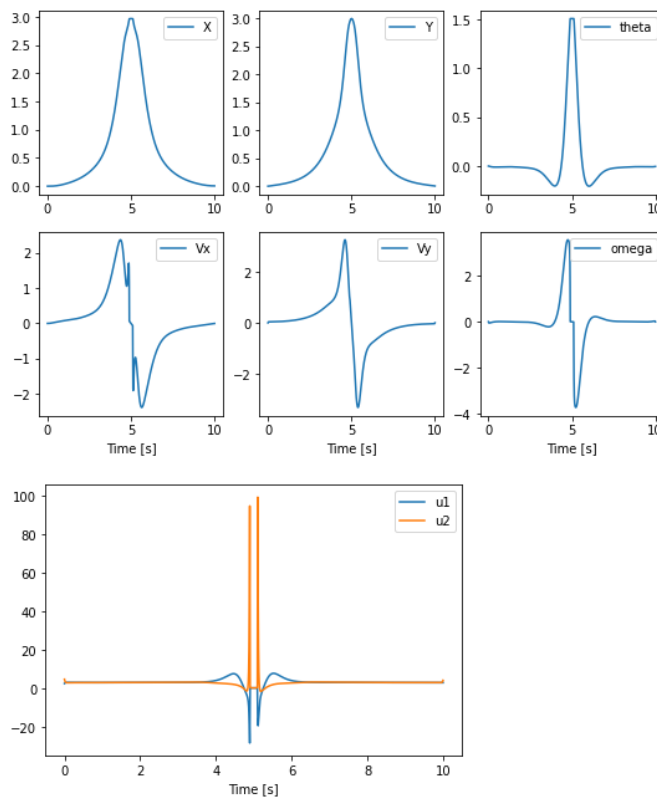
2. Quadratic approximation of cost: (z_n^*, u_n^*) are trajectory states and controls

$$\begin{aligned}
 J \approx & \text{const} \\
 & + 2\theta_N z_N^* \bar{z}_N + \frac{1}{2} \bar{z}_N (2\theta_N) \bar{z}_N \\
 & + \sum_{n=0}^{499} \left(2\theta_1 z_n^* \bar{z}_n + \frac{1}{2} \bar{z}_n 2\theta_1 \bar{z}_n \right. \\
 & \quad \left. + 2R u_n^* \bar{u}_n + \frac{1}{2} \bar{u}_n 2R \bar{u}_n \right) \\
 & + \sum_{n=490}^{510} \left(2\theta_2 (z_n^* - \tilde{z}) \bar{z}_n + \frac{1}{2} \bar{z}_n 2\theta_2 \bar{z}_n \right. \\
 & \quad \left. + 2R u_n^* \bar{u}_n + \frac{1}{2} \bar{u}_n 2R \bar{u}_n \right) \\
 & + \sum_{n=511}^{999} \left(2\theta_1 z_n^* \bar{z}_n + \frac{1}{2} \bar{z}_n 2\theta_1 \bar{z}_n \right. \\
 & \quad \left. + 2R u_n^* \bar{u}_n + \frac{1}{2} \bar{u}_n 2R \bar{u}_n \right)
 \end{aligned}$$

3. Initial trajectory:



Final trajectory:



Benefits:

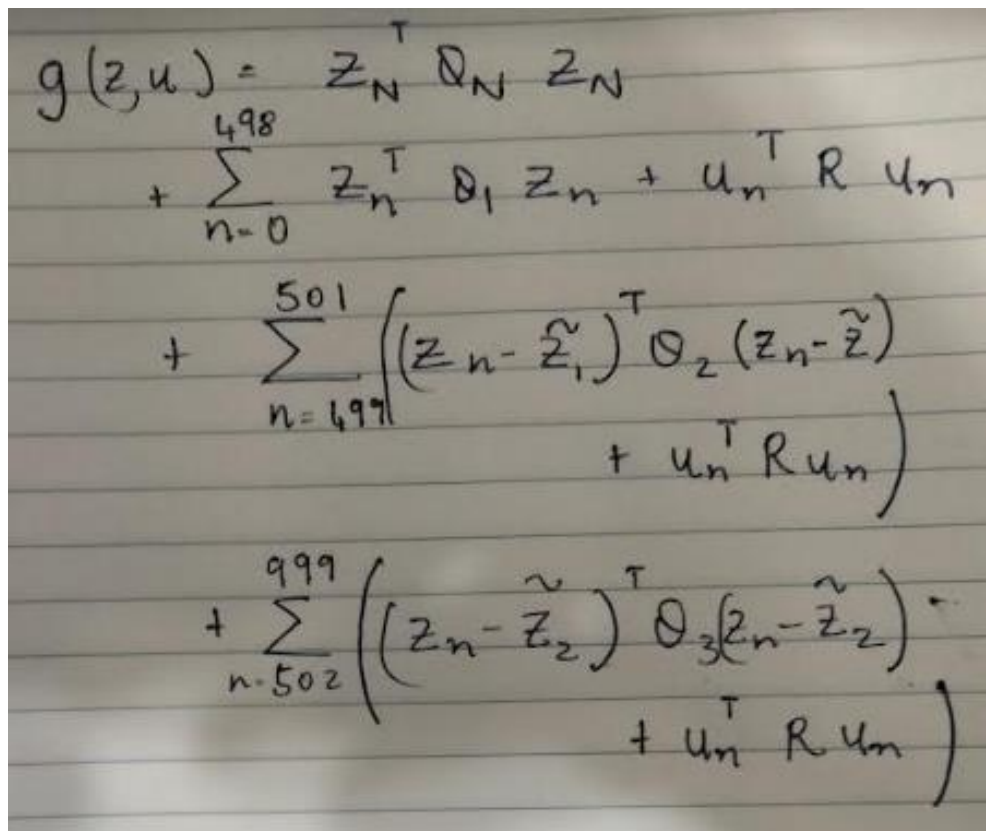
- Able to find a state and control trajectory very quickly
- Quadrotor reaches the target very well
- We were able to find a locally optimal trajectory from the first guess
- Line search is able to fit a best guess for control

Issues:

- If the cost function does not promote the behaviour well, the iLQR algorithm does not converge
- The trajectory found is locally optimal and might not be the best solution.
- There are sharp changes control and the velocities which might not work in real life
- Again, if the problem was in 3D, the state and control vectors would be larger. Then solving the backward Riccati equations on every step and iterating through them would be very computationally expensive

Task 2

New Cost function:


$$\begin{aligned} g(z, u) = & z_N^T Q_N z_N \\ & + \sum_{n=0}^{498} z_n^T Q_1 z_n + u_n^T R u_n \\ & + \sum_{n=499}^{501} \left((z_n - \tilde{z}_1)^T Q_2 (z_n - \tilde{z}_1) + u_n^T R u_n \right) \\ & + \sum_{n=502}^{999} \left((z_n - \tilde{z}_2)^T Q_3 (z_n - \tilde{z}_2) + u_n^T R u_n \right) \end{aligned}$$

Where:

$$Q_1 = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

$$Q_2 = 100000 * \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

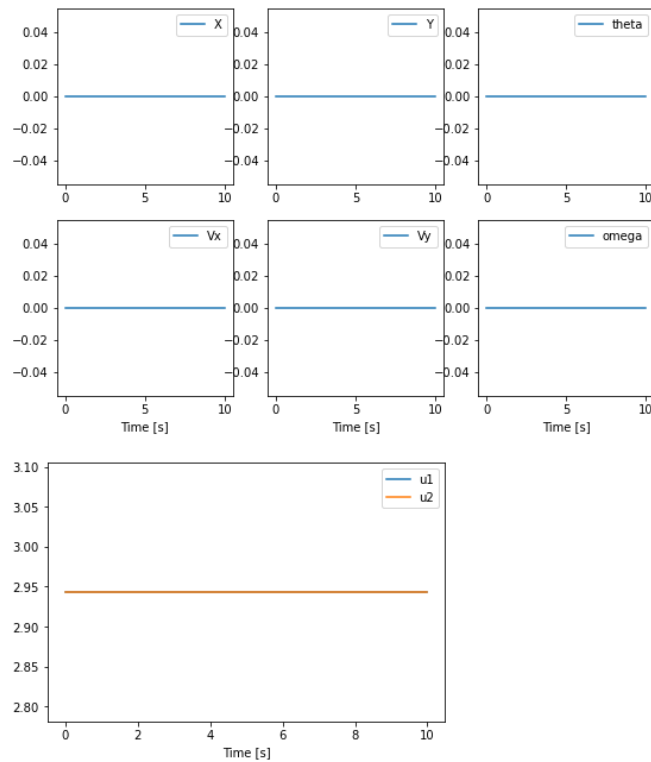
$$Q_3 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix}$$

$$Q_N = 100000 * \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

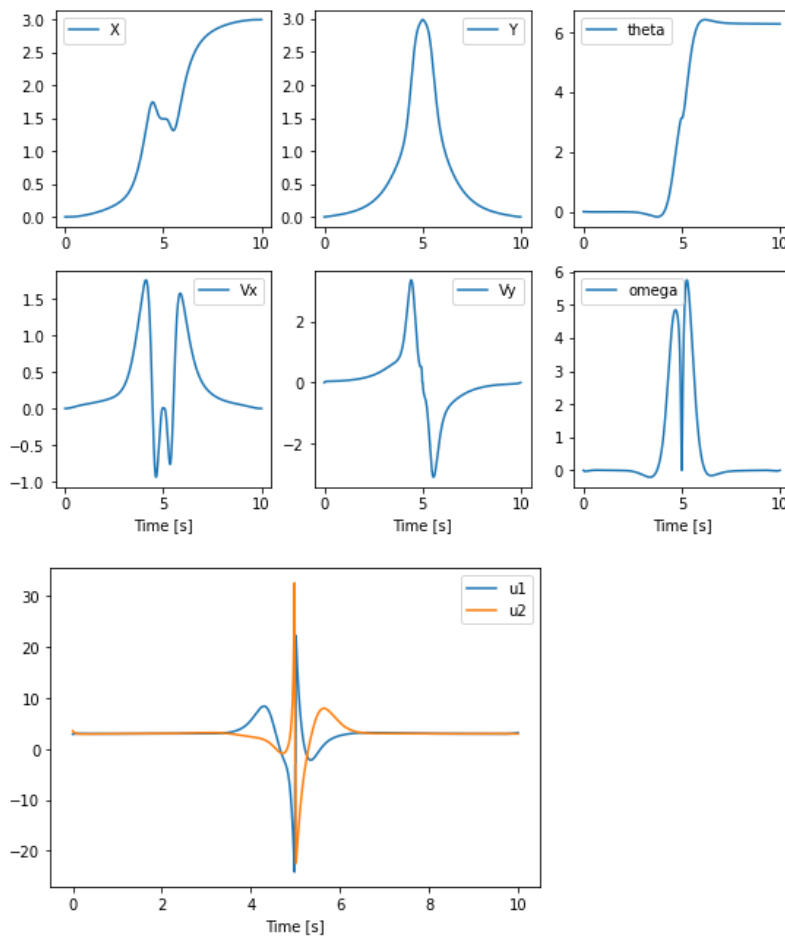
$$\tilde{z}_1 = \begin{bmatrix} 1.5 \\ 0 \\ 3 \\ 0 \\ \pi \\ 0 \end{bmatrix} \quad \tilde{z}_2 = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 2\pi \\ 0 \end{bmatrix}$$

Results :

Initial trajectory:



State and control trajectory found:



The quadrotor is able to follow a trajectory that makes it do a full flip. The x , y and θ positions at $t=5$ are 1.5, 3 and π respectively. This was exactly the target point. The control jumps at $t = 5$ from positive to negative for u_2 and negative to positive for u_1 very steeply. As a result, there are sharp changes in ω , v_x and v_y that make the motion seem a bit jerky.

Benefits:

- Able to find a state and control trajectory very quickly
- Quadrotor reaches the target very well
- We have a good approximation of the system

Issues:

- There are solutions where the control is negative. This is quite clearly not possible in a real quadrotor
- The cost must be very accurate to generate the required trajectory
- Sometimes with small changes in cost, the iLQR algorithm does not converge.
- Again, if the problem was in 3D, the state and control vectors would be larger. Then solving the backward Riccati equations on every step and iterating through them would be very computationally expensive

We cannot run this controller on a real robot as on a real robot we will have constraints, such as u_1 and u_2 cannot be negative and cannot change too quickly. The real motors cannot handle the control generated by this program.