Yinhong Qin
yq2021@nyu.edu

Project 1

ROB 6323
November 28, 2022

Part 1

1. The state variable of this 2D quadrotor is

$$z = \begin{bmatrix} x \\ v_x \\ y \\ v_y \\ \theta \\ \omega \end{bmatrix}$$

We also have the system dynamic equations of this quadrotor model.

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

Thus the discretized system dynamics is:

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

$$z_{n+1} = \begin{bmatrix} x_{n+1} \\ v_{x,n+1} \\ y_{n+1} \\ v_{y,n+1} \\ \theta_{n+1} \\ \omega_{n+1} \end{bmatrix} \quad s = \begin{bmatrix} x_n + \Delta t \cdot v_x \\ v_{x,n} + \Delta t \cdot -\frac{(u_1+u_2)\sin\theta}{m} \\ y_n + \Delta t \cdot v_y \\ v_{y,n} + \Delta t \cdot (\frac{(u_1+u_2)\cos\theta}{m} - g) \\ \theta_n + \Delta t \cdot \omega \\ \omega_n + \Delta t \cdot \frac{r(u_1-u_2)}{I} \end{bmatrix}$$

2. For this question, we want the drone stay at the position (x_0, y_0) forever with the initial condition which is:

$$z_0 = \begin{bmatrix} x_0 \\ 0 \\ y_0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

The derivative of z at time 0 is:

$$\dot{z}_{t=0} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{(u_1+u_2)}{m} - g \\ 0 \\ \frac{r(u_1-u_2)}{I} \end{bmatrix}$$

If we want to keep the drone at the initial position forever, we can have

$$\begin{cases} \frac{(u_1+u_2)}{m} - g = 0 \\ \frac{r(u_1-u_2)}{I} = 0 \end{cases}$$

Thus we can know

$$u_1 = u_2 = \frac{mg}{2}$$

$$u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix} = \begin{bmatrix} \frac{mg}{2} \\ \frac{mg}{2} \end{bmatrix}$$

3. To move the drone in the x direction and keep $\theta = 0$, we can have

$$\begin{cases} v_x > 0 \\ \dot{v_x} = -\frac{(u_1+u_2)\sin 0}{m} = 0 \\ v_y = 0 \\ \dot{v_y} = \frac{(u_1+u_2)}{m} - g = 0 \\ \dot{\theta} = 0 \\ \dot{\omega} = \frac{r(u_1-u_2)}{I} = 0 \end{cases}$$

Thus we can know

$$u_1 = u_2 = \frac{mg}{2}$$

Therefore, this problem really depends on the initial speed of this drone. If it has an initial speed $v_{x,0} > 0$, then after applying the control unit

$$u^* = \begin{bmatrix} u_1^* \\ u_2^* \end{bmatrix} = \begin{bmatrix} \frac{mg}{2} \\ \frac{mg}{2} \end{bmatrix}$$

it can move in the x direction with this constant speed $v_{x,0}$. Otherwise, it will stay in the initial position forever.

4. If we want this drone to stay with $\theta = \frac{\pi}{2}$, we can have

$$\begin{cases} \dot{v_x} = -\frac{(u_1+u_2)\sin\frac{\pi}{2}}{m} = -\frac{(u_1+u_2)}{m} \\ \dot{v_y} = \frac{(u_1+u_2)\cos\frac{\pi}{2}}{m} - g = -g \\ \dot{\theta} = 0 \\ \dot{\omega} = \frac{r(u_1-u_2)}{I} = 0 \end{cases}$$

Thus, we can notice that this drone will never stay at its original position because of gravity. No matter how much control force we apply on it, it will fall down to the ground.

Part 2

1. From part 1 we could know the state variable and discretized system dynamics is:

$$z = \begin{bmatrix} x \\ v_x \\ y \\ v_y \\ \theta \\ \omega \end{bmatrix}$$

$$z_{n+1} = \begin{bmatrix} x_{n+1} \\ v_{x,n+1} \\ y_{n+1} \\ v_{y,n+1} \\ \theta_{n+1} \\ \omega_{n+1} \end{bmatrix} s = \begin{bmatrix} x_n + \Delta t \cdot v_x \\ v_{x,n} + \Delta t \cdot -\frac{(u_1+u_2)\sin\theta}{m} \\ y_n + \Delta t \cdot v_y \\ v_{y,n} + \Delta t \cdot (\frac{(u_1+u_2)\cos\theta}{m} - g) \\ \theta_n + \Delta t \cdot \omega \\ \omega_n + \Delta t \cdot \frac{r(u_1-u_2)}{I} \end{bmatrix}$$

Let $z_{n+1} = f(z_n, u_n)$ and to perform the linearization at operation point (z^*, u^*), we can have

$$z_{n+1} \approx f(z^*, u^*) + \frac{\partial f}{\partial z}_{z=z^*, u=u^*}(z_n - z^*) + \frac{\partial f}{\partial u}_{z=z^*, u=u^*}(u_n - u^*)$$

Moreover, the explicit form of $\frac{\partial f}{\partial z}$ and $\frac{\partial f}{\partial u}$ are:

$$A = \frac{\partial f}{\partial z} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t \cdot -\frac{(u_1+u_2)\cos\theta}{m} & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \cdot -\frac{(u_1+u_2)\sin\theta}{m} & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \frac{\partial f}{\partial u} = \begin{bmatrix} 0 & 0 \\ \Delta t \cdot (-\frac{\sin\theta}{m}) & \Delta t \cdot (-\frac{\sin\theta}{m}) \\ 0 & 0 \\ \Delta t \cdot \frac{\cos\theta}{m} & \Delta t \cdot \frac{\cos\theta}{m} \\ 0 & 0 \\ \Delta t \cdot \frac{r}{I} & -\Delta t \cdot \frac{r}{I} \end{bmatrix}$$

Thus, the linearized dynamics around operation point (z^*, u^*) is

$$\bar{z}_{n+1} = A_{z=z^*, u=u^*} * \bar{z}_n + B_{z=z^*, u=u^*} * \bar{u}_n$$

where

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

and

$$A_{z=z^*,u=u^*} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t \cdot -\frac{(u_1+u_2)\cos\theta^*}{m} & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \cdot -\frac{(u_1+u_2)\sin\theta^*}{m} & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_{z=z^*,u=u^*} = \frac{\partial f}{\partial u} = \begin{bmatrix} 0 & 0 \\ \Delta t \cdot (-\frac{\sin\theta^*}{m}) & \Delta t \cdot (-\frac{\sin\theta^*}{m}) \\ 0 & 0 \\ \Delta t \cdot \frac{\cos\theta^*}{m} & \Delta t \cdot \frac{\cos\theta^*}{m} \\ 0 & 0 \\ \Delta t \cdot \frac{r}{I} & -\Delta t \cdot \frac{r}{I} \end{bmatrix}$$

2. Please check in the Jupyter Notebook.

3. Assume we have the quadratic cost matrix Q and R. To stablize at the resting point, we can calculate the controller by solving

$$\min_{\bar{u}_k}(\sum_{k=0}^{\infty} \bar{z}_k^T Q \bar{z}_k + \bar{u}_k^T R \bar{u}_k)$$

subject to

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

To solve this problem, we need to solve the algebraic Riccati equations which are:

$$\begin{aligned} P &= Q + A^T PA + A^T PB(B^T PB + R)^{-1}(B^T PA) \\ K &= -(B^T PB + R)^{-1} B^T PA \end{aligned}$$

Thus, the controller is

$$\bar{u}_n = K\bar{z}_n$$

and for the original system, we can have

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

4. For this problem, to stablize the drone at the origin, we can set the operating point

$$(z^*, u^*) = 0$$

Then, after adjusting the Q and R I choose are:

$$Q = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}, R = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

We can get the A and B matrix from the linearized model mentioned in 2.1. Finally, plug in A, B, Q, R to the Riccati equations, get the solution K matrices and calculate the control force u_n according to 2.3.

Please check the code in Jupyter notebook too.

5. For Part2, since we are using the LQR to get the controller, our cost function should be quadratic like

$$\sum_{k=0}^{\infty} \bar{z}_k^T Q \bar{z}_k + \bar{u}_k^T R \bar{u}_k$$

We want to keep the drone to stay in the origin position, the desired state z^* should always be $\mathbf{0}^T$ and the corresponding desired control force is $u^* = [\frac{1}{2}mg, \frac{1}{2}mg]$ used to compensate the gravity. The control law we found actually helps the drone to resist possible perturbations. For this problem, if there is not any disturbance, the control law is the same as $u^* = [\frac{1}{2}mg, \frac{1}{2}mg]$. Otherwise it will help compensate any deviation from the origin.

For more details, we plug in the z^* and u^* into the optimization problem defined in question 2.3, perform the Riccati recursion and finally get the optimal control forces to make the drone stay in the orgin. Here we set a higher value of Q matrix (1000) for the penalty caused by the deviation from the desired state. Besides, we also set a relatively lower value of R matrix (10). The reason is if the penalty of input force is too large, the control force will be suppressed so this drone will take longer time to compensate the error caused by random disturbance.

When the "disturbance" feature is turned on, please check the animation in the jupyter notebook.

Part 3

1. If we want to tracking a circle of radius 1 and centered at $(0, 0)$ while keeping $\theta = 0$, we can have:

$$\begin{cases} x^2 + y^2 = 1 \\ \theta = 0 \end{cases}$$

For the desired state z^*, it is

$$z_t^* = \begin{bmatrix} \cos(\gamma) \\ \frac{2\pi}{T} \cdot -\sin(\gamma) \\ \sin(\gamma) \\ \frac{2\pi}{T} \cdot \cos(\gamma) \\ 0 \\ 0 \end{bmatrix}$$

where $T = 10$ and $\gamma = \frac{t}{T} \cdot 2\pi$

According to part 2, the result of linerization, A, B matrix, will then become

$$A_{\theta=0,u=u^*} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t \cdot -g & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B_{\theta=0,u=u^*} = \frac{\partial f}{\partial u} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \Delta t \cdot \frac{1}{m} & \Delta t \cdot \frac{1}{m} \\ 0 & 0 \\ \Delta t \cdot \frac{r}{I} & -\Delta t \cdot \frac{r}{I} \end{bmatrix}$$

So we can notice that, A and B matrix both become constant because $\Delta T, m, r, I$ are all given values.

2. We can get the points on the trajectory (or the operating points mentioned in part 2) $\bar{z}$ by uniformly sampling on this circle centered at $(0, 0)$ with a radius of 1. For personal design, the drone will start from the origin and move counterclockwisely. The destination at $t = 10$ is $(1, 0)$.

For the cost function, we use quadratic function and the Q and R matrix are:

$$Q = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1000 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1000 \end{bmatrix}, R = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

The A and B matrix from the linearized model mentioned in 3.1. Solving trajectory following problem use Riccati recursion is to solve these equations:

$$\begin{aligned} K_n &= -\left(B_n^T P_{n+1} B_n + R_n\right)^{-1} B_n^T P_{n+1} A_n \\ P_n &= Q_n + A_n^T P_{n+1} A_n + A_n^T P_{n+1} B_n K_n \\ k_n &= -\left(B_n^T P_{n+1} B_n + R_n\right)^{-1} B_n^T p_{n+1} \\ p_n &= q_n + A_n^T P_{n+1} A_n + A_n^T P_{n+1} B_n k_n \end{aligned}$$

where N is the horizon length, $n = N-1, \cdots, 0$ and $q_n = -Q_n \bar{z}, p_N = q_N, P_N = Q_N$

The found control law in the original coordinates (z, u) is

$$u_n = u_n^* + K_n(z_n - \bar{z_n}) + k_n$$

3. Please check the jupyter notebook for the animation about tracking a circle trajectory with and without disturbances.

4. Here are the plots:

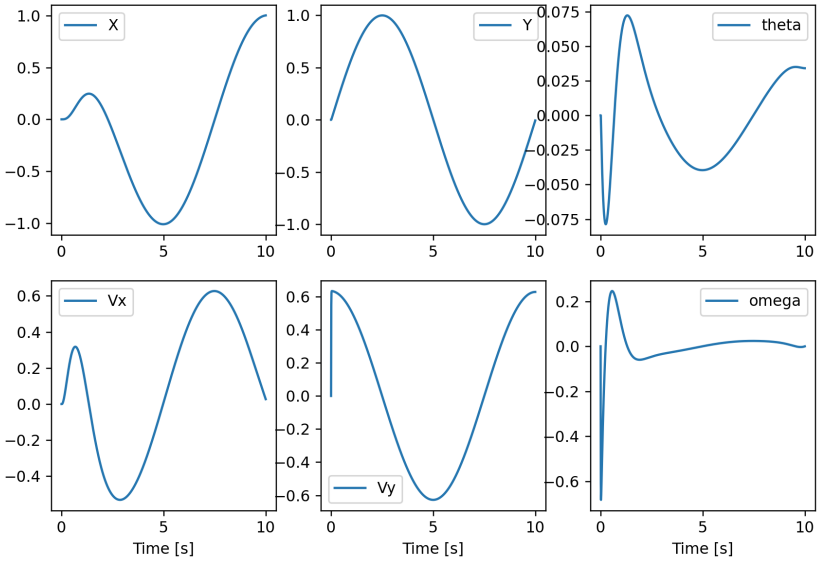

Figure 3.4.1. State variables (without disturbance).

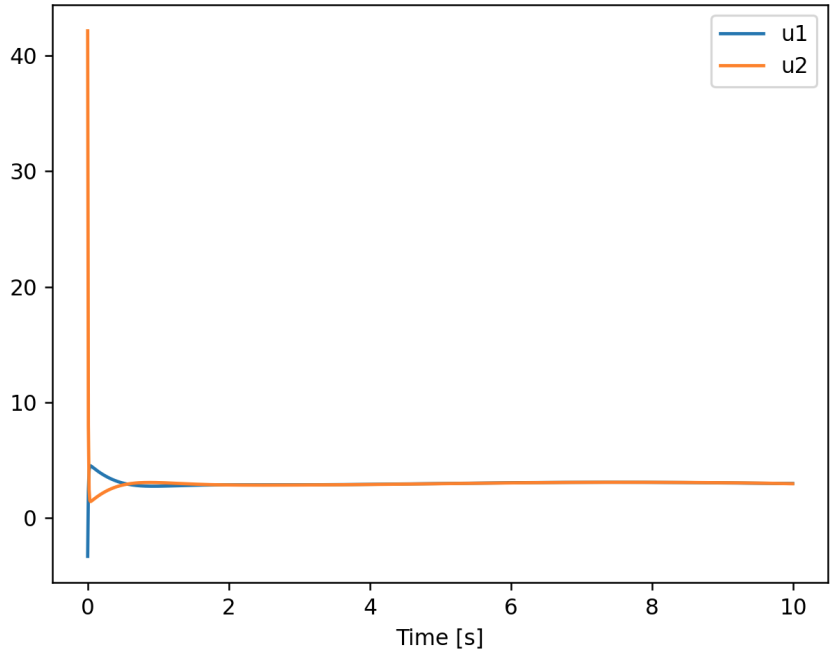

Figure 3.4.2. Control forces (without disturbance).

After we turn on the disturbance:

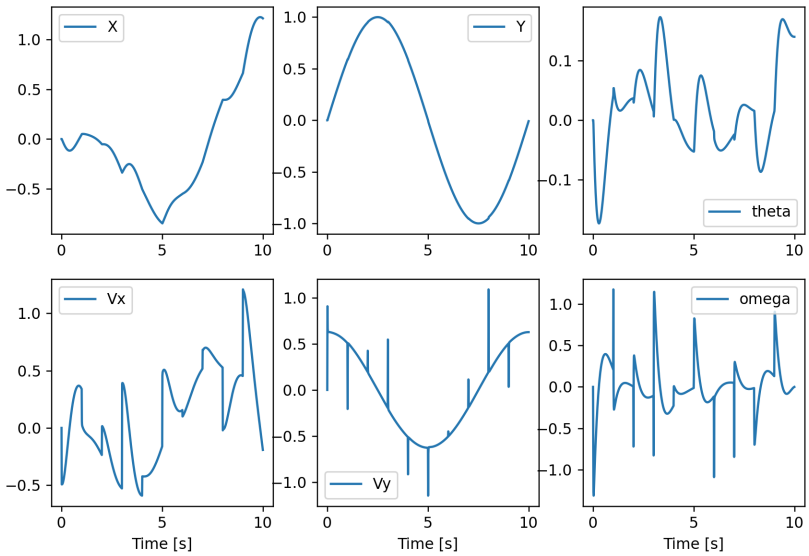

Figure 3.4.3. State variables (with disturbance).

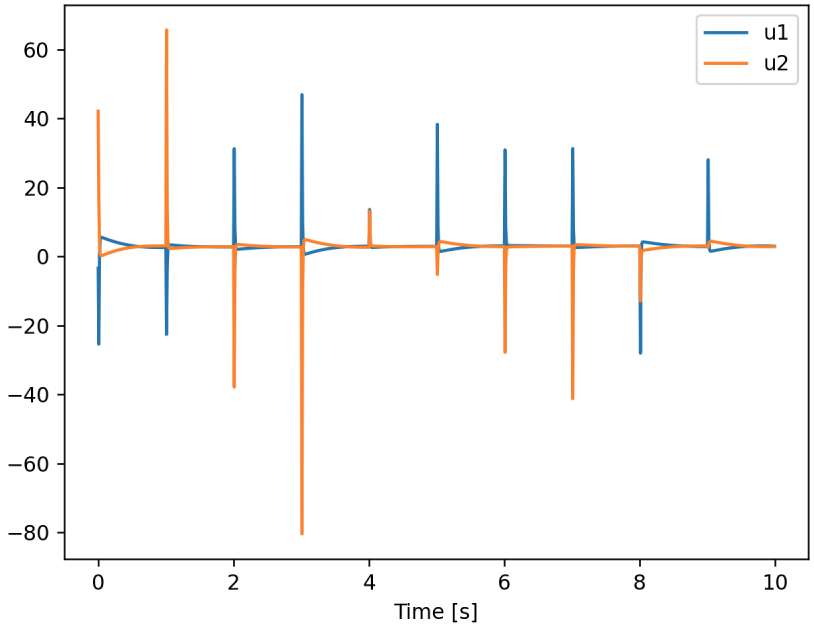

Figure 3.4.4. Control forces (with disturbance).

From these plots, we can see the controller performs very well when tracking the position

and velocity the y direction. For the x direction, because we start from the origin, it has some deviation at the beginning. But after a few second it can also follow the circle trajectory very accurately.

When we turn on the disturbance, we can notice that the tracking on y-position is still nice. For other state variables, the deviation is larger because of the random disturbance applied every second. The controller can make correct reaction to compensate the disturbance, which is the bursts shown in the pic 3.4.4.

5. It is possible to do the same thing while keeping $\theta = \frac{\pi}{4}$. We can change the operating point and calculate a new controller to do it.

Part 4

For task 1:

1. For part 4, We can set the cost function as:

$$\begin{aligned} cost(z,u) =& \frac{1}{2}\sum_{t=0}^{T/2}(z_t - z_{goal1})^T Q_1 (z_t - z_{goal1}) + u_t^T R_1 u_t \\ & \frac{1}{2}\sum_{t=T/2}^{T} +(z_t - z_{goal2})^T Q_2 (z_t - z_{goal2}) + u_t^T R_2 u_t \end{aligned}$$

   where

$$z_{goal1} = \begin{bmatrix} 3 \\ 0 \\ 3 \\ 0 \\ \frac{\pi}{2} \\ 0 \end{bmatrix}$$

   and

$$z_{goal2} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2. Please check the code in the jupyter notebook.

3. The quadratic approximation of this cost function is:

$$cost(z,u)_{z_t,u_t} \approx cost(z_t,u_t) + \frac{\partial cost(z,u)}{\partial z} + \frac{\partial cost(z,u)}{\partial u} + \frac{\partial^2 cost(z,u)}{\partial z^2} + \frac{\partial^2 cost(z,u)}{\partial u^2}$$

   The first order of derivatives (Jacobians) are:

$$\begin{aligned} \frac{\partial cost(z,u)}{\partial z} =& \frac{1}{2}\sum_{t=0}^{T/2} Q_1 (z_t - z_{goal1}) + \frac{1}{2}\sum_{t=T/2}^{T} Q_2 (z_t - z_{goal1}) \\ \frac{\partial cost(z,u)}{\partial u} =& \frac{1}{2}\sum_{t=0}^{T/2} R_1 u_t + \frac{1}{2}\sum_{t=T/2}^{T} R_2 u_t \end{aligned}$$

The second order of derivatives (Hessians) are:

$$\frac{\partial^2 cost(z,u)}{\partial z^2} = \frac{1}{2}\sum_{t=0}^{T/2} Q_1 + \frac{1}{2}\sum_{t=T/2}^{T} Q_2$$

$$\frac{\partial^2 cost(z,u)}{\partial u^2} = \frac{1}{2}\sum_{t=0}^{T/2} R_1 + \frac{1}{2}\sum_{t=T/2}^{T} R_2$$

4. Please check the code in the jupyter notebook.

5. Please check the code in the jupyter notebook. From the animation we can see the deviation of the drone's orientation is very small so we can say we successfully track θ.

6. Here are the plots:

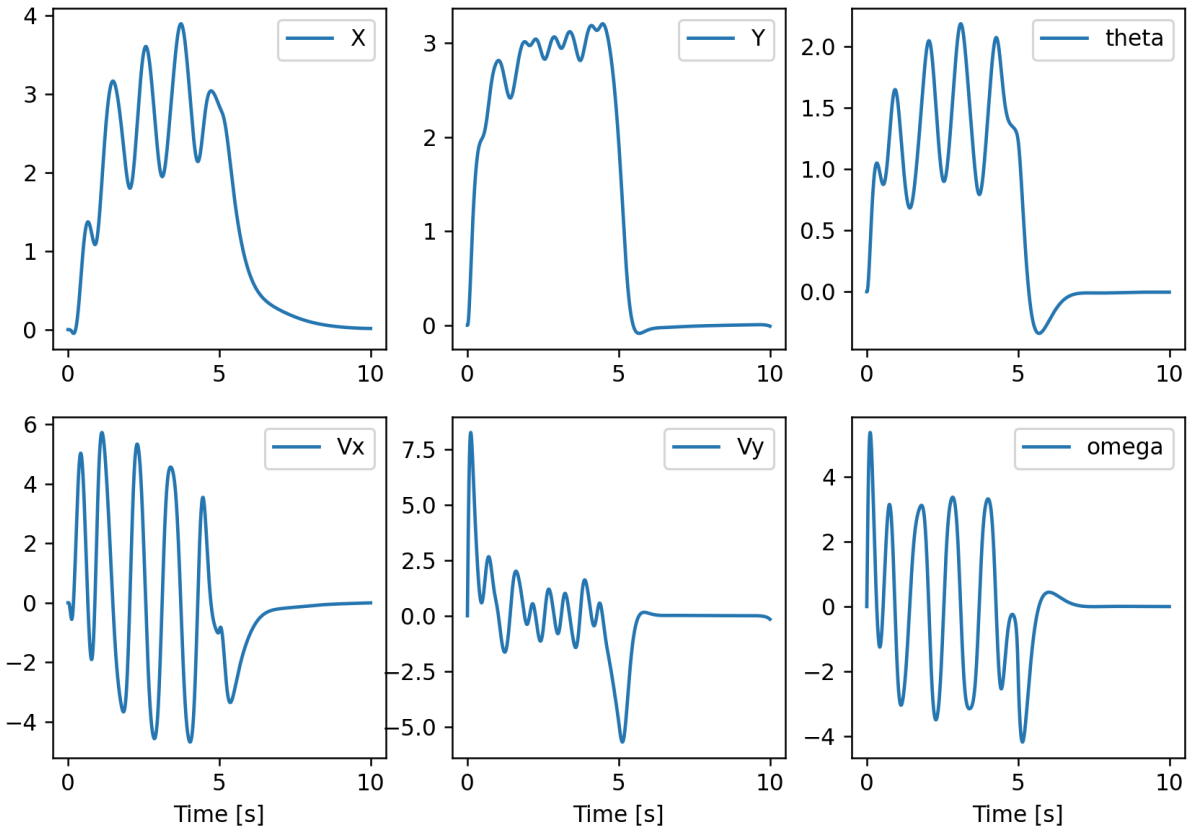

Figure 4.1.1. State variables.

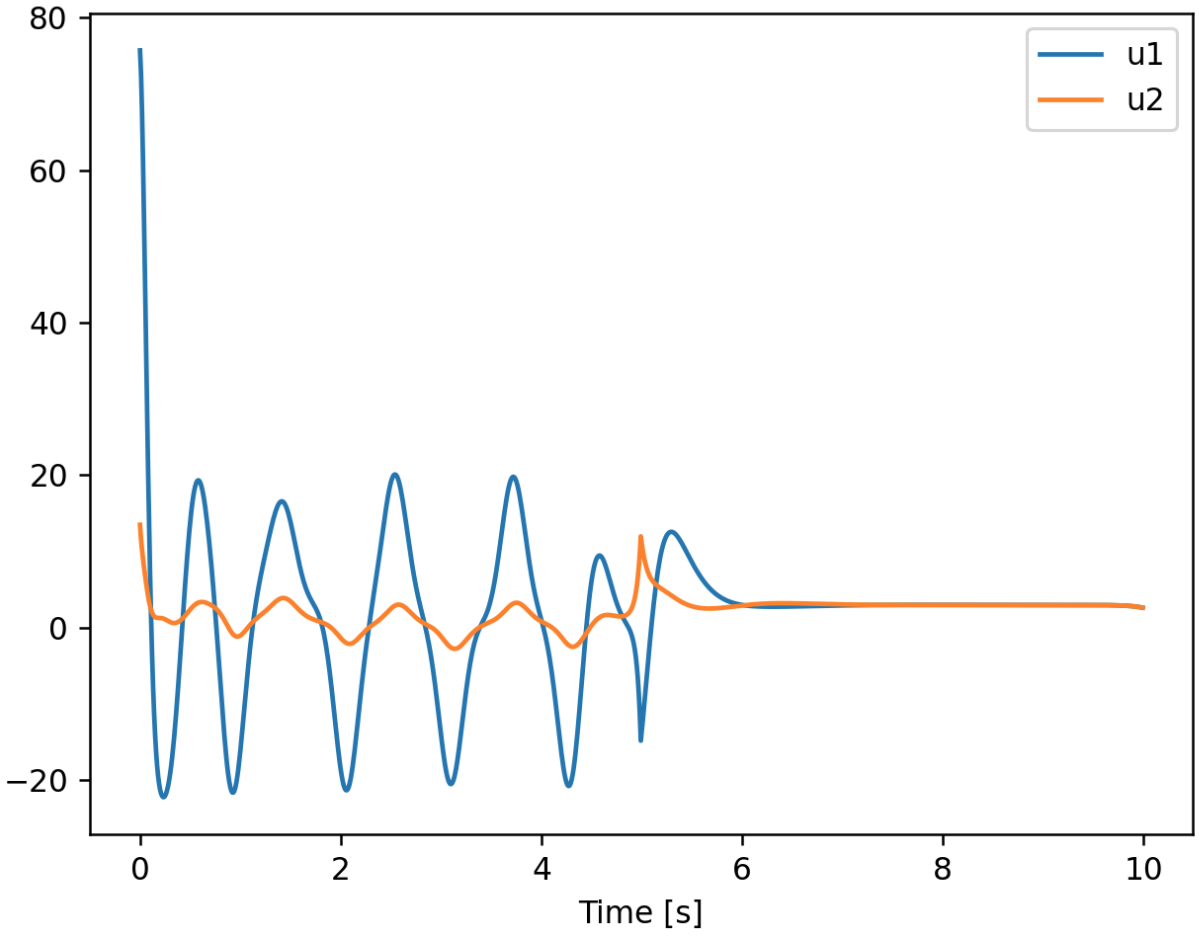

Figure 4.1.2. Control forces.

7. From the plots we can notice that the controller can first drive the drone to destination $(3, 3)$. But the issue is when it reaches this point it is trying to maintain the orientation at $\theta = \frac{\pi}{2}$. Thus it starts to oscillate at that position. After 5 seconds, it will soon be back to the start point.

8. For this flip task, we need to change our goals. Here we use

$$z_{goal1} = \begin{bmatrix} 1.5 \\ 0 \\ 3 \\ 0 \\ \pi \\ 0 \end{bmatrix}$$

and

$$z_{goal2} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ 2\pi \\ 0 \end{bmatrix}$$

We use quadratic cost function similarly as we used in task1. The Q and R matrix we used are:

Phase 1:

$$Q_1 = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}, R_1 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

Phase 2:

$$Q_2 = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 \\ 0 & 100 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1000 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \end{bmatrix}, R_2 = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

For the animation please check the jupyter notebook. The plots are:

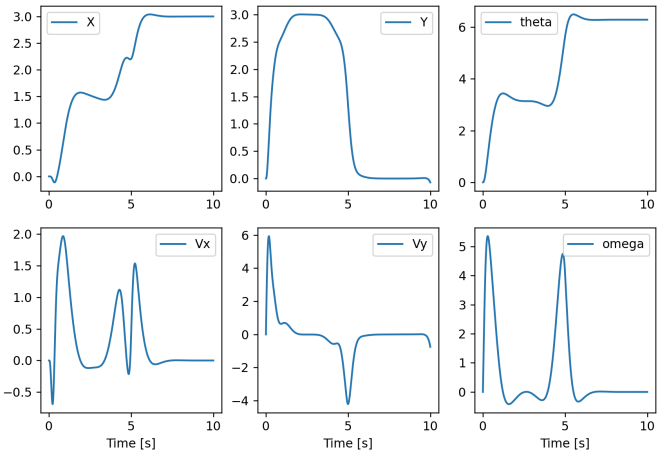

Figure 4.2.1. State variables.

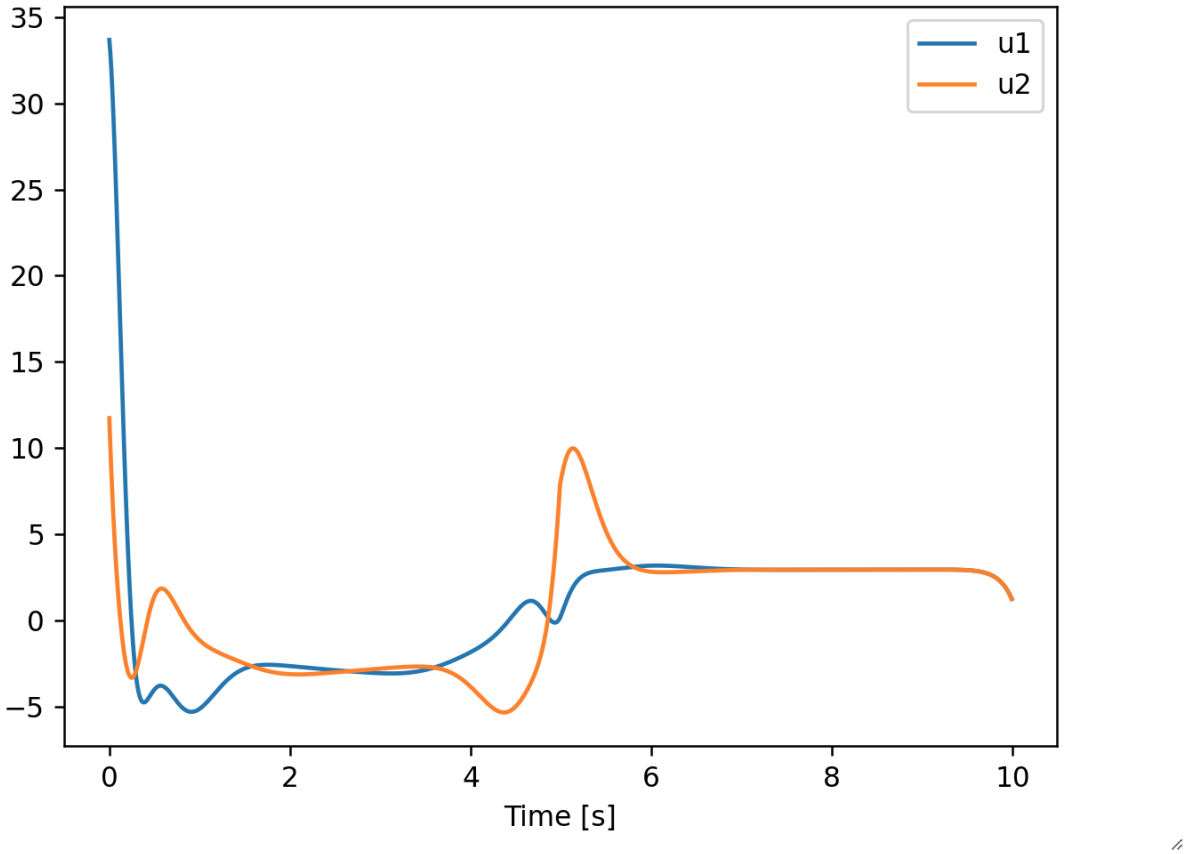

Figure 4.2.2. Control forces.

9. From the plots and animation, we can notice that a benefit for this controller is it gives a very fast responses to each goal and can stay at that position. The issue is during the transition from the first goal to the second goal, the controller seems to move in advance. In practice, we need to adjust the parameters according to the physical limitations of the drone such as the maximum velocity and angular velocity etc.