Project-1

Venkata Amith Palacherla

November 28, 2022

Abstract

The objective of the project is to write different controllers for 2D quadrotor to perform acrobatics moves. State of the quadrotor is defined by six variable and it can be controlled using 2 inputs. The project start with finding the descritized dynamics of the quadrotor then finding stable position and control. In the part-2 LQR controller to make the quadrotor to stay at predefined position is designed. In part 3 a LQR controlled is designed to make drone follow a predefined trajectory is designed. In the last part a iLQR controller is designed to make drone flip.

1 Part 1

1.1 Discretization of Dynamics

For position in X direction -

$$x_{n+1} = x_n + \Delta t * v_{x_n}$$

For velocity in X direction -

$$v_{x_{n+1}} = -\frac{u_1 + u_2}{m} * \sin\theta * \Delta t + v_{x_n}$$

For position in Y direction -

$$y_{n+1} = y_n + \Delta t * v_{y_n}$$

For velocity in X direction -

$$v_{y_{n+1}} = (\frac{u_1 + u_2}{m} * \cos\theta - g) * \Delta t + v_{y_n}$$

For angle -

$$\theta_{n+1} = \theta_n + \omega_n * \Delta t$$

For angular velocity

$$\omega_{n+1} = \omega_n + r * \frac{u_1 - u_2}{I} * \Delta t$$

State and Control matrix are written as -

$$z_n = \begin{bmatrix} x_n \\ v_{x_n} \\ y_n \\ v_{y_n} \\ \theta_n \\ \omega_n \end{bmatrix} \qquad u_n = \begin{bmatrix} u_{n_1} \\ u_{n_2} \end{bmatrix}$$

1.2 Finding Control for stable state $Z_n = 0$

Stable state and control are defined as when a control u_n is applied to system at some state z_n the system will stay in same state.

$$z_n = f(z_n, u_n)$$

Control input of stable position $z_n = 0$ -

$$z_{n+1} = z_n = 0$$

By using this condition on discretized dynamics -

$$(\frac{u_1 + u_2}{m} * \cos\theta - g) = 0$$

$$r * \frac{u_1 - u_2}{I} = 0$$

By solving these to equations -

$$u_n = 0.5 * \begin{bmatrix} m * g \\ m * g \end{bmatrix}$$

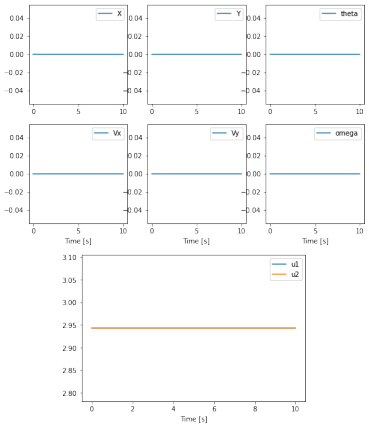

Figure 1: System at the resting state and corresponding control

1.3 Movement in X direction

When $\theta = 0$, velocity in x direction will be -

$$v_{x_{n+1}} = v_{x_n} + 0$$

This means when θ is zero irrespective of control quadrotor will have same velocity in x-direction. when the quadrotor starts with zero velocity in x-direction it won't be able to move in x-direction for any control input.

1.4 Movement in Y direction

When $\theta = \pi/2$, velocity in y direction will be -

$$v_{yn+1} = -g * \Delta t$$

When θ is $\pi/2$ quadrotor will always have some velocity and acceleration in y-direction irrespective of control inputs.

2 LQR to stay in place

2.1 Linearization of dynamics around $z*$ and $u*$

For function $f(x, u)$ linearization $x*$ and $u*$ can be written as -

$$x_{n+1} = f(x*, u*) + \frac{\partial f}{\partial x}(x - x*) + \frac{\partial f}{\partial u}(u - u*)$$

this equation for quadrotor can be written as -

$$\bar{x_{n+1}} = A\bar{x_n} + B\bar{u_n}$$

where,

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

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

$$\bar{z_{n+1}} = z_{n+1} - f(x*, u*)$$

$$\bar{z_n} = z_n - z*$$

$$\bar{u_n} = u_n - u*$$

2.2 get linearization function

Function which linearizes the system dynamics around x^* and u^* is written in ipynb file.

2.3 Stabilizing drone over a resting point

Let the cost function for the system be -

$$\sum_{n=0}^{\infty}(x_n - x*)^T Q_n(x_n - x*) + (u_n - u*)^T R_n(u_n - u*)$$

where,x^* and u^* are the resting state and corresponding control.
cost can be rewritten as -

$$\sum_{n=0}^{N}(\bar{x_n})^T Q_n(\bar{x_n}) + (\bar{u_n})^T R_n(\bar{u_n})$$

and dynamics of the system will be -

$$\bar{x_{n+1}} = A\bar{x_n} + B\bar{u_n}$$

By solving Riccati's recursion equations control gain(K) can be calculated. For a infinite horizon problem gain will be constant over the horizon. The gains calculated from Riccati's equations are for $\bar{x}$ and $\bar{u}$.

$$\bar{u} = K * (\bar{x})$$

by rearranging the terms

$$u = K * (x - x^*) + u^*$$

2.4 Stabilizing around z_n=0

Stabilizing the quadrotor without disturbance. The system is linearized using $get_linearization$ function from 2.2. System is linearized around $z_n = 0$ and

$$u_n^* = 0.5 * \begin{bmatrix} m * g \\ m * g \end{bmatrix}$$

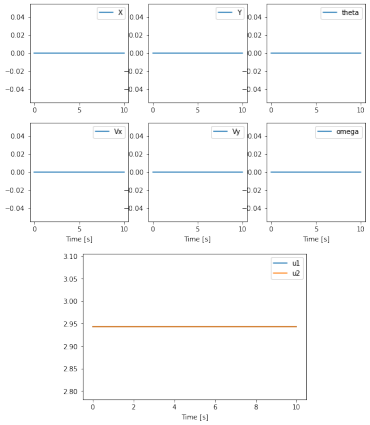

Figure 2: System at the resting state and corresponding control

2.5 Designing the cost function

Cost function was designed such that system is penalized whenever it is away from the resting position x^* and corresponding input u^*. Therefore cost function is -

$$\sum_{n=0}^{N}(x_n - x^*)^T Q_n(x_n - x^*) + (u_n - u^*)^T R_n(u_n - u^*)$$

$$Q_n = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

$$R_n = \begin{bmatrix} 0.001 & 0 \\ 0 & 0.001 \end{bmatrix}$$

The Q_n is designed such the system is more penalized when it is out rest position, therefore x,y,θ have higher values than velocity. These values were found out by trail and error. In the case of disturbance quadrotor still stays near the resting position. But then disturbance are more linearized dynamics will no longer work and it will not be at rest.

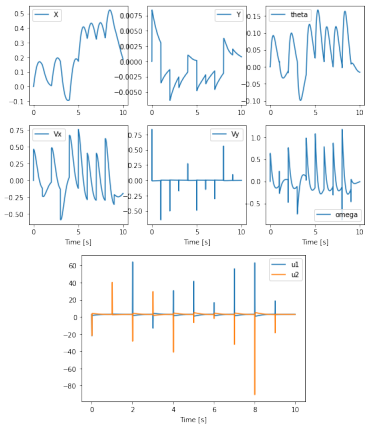

Figure 3: System at the resting state and corresponding control with disturbances

3 Controller to follow a trajectory

To make quadrotor follow a trajectory first system dynamics have to be linearized around the desired states and desired controls are different time steps.
Linearized system dynamics can be written as -

$$\bar{z_{n+1}} = A_n \bar{z_n} + B_n \bar{u_n}$$

where, $A_n = \frac{\partial f}{\partial z}$ linearized around z_n^*, u_n^*,
$B_n = \frac{\partial f}{\partial u}$ linearized around z_n^*, u_n^*,
x_n^*, u_n^* are the desired state and desired control to follow the trajectory.

To make quadrotor follow a circle desired state at different time steps

$$z_n^* = \begin{bmatrix} \cos\frac{2\pi * 1000}{n} \\ 0.68\sin\frac{2\pi * 1000}{n} \\ \sin\frac{2\pi * 1000}{n} \\ 0.68\cos\frac{2\pi * 1000}{n} \\ 0 \\ 0 \end{bmatrix}$$

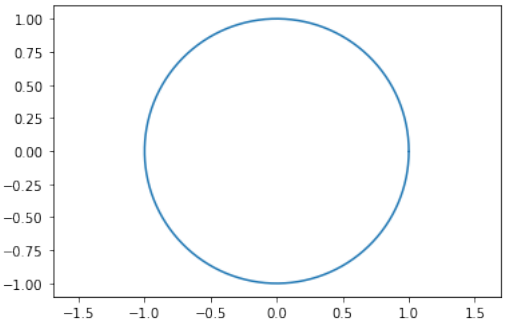

Figure 4: Quadrotor trajectory. It will start from 1,0 and move in anti-clockwise direction

desired control was taken to as -

$$u_n^* = 0.5 * \begin{bmatrix} mg \\ mg \end{bmatrix}$$

System dynamics cannot be linearized around single point as they are different at different points of the trajectory. System will not move in the desired trajectory if it is not linearized at all desired positions.

The cost function to make quadrotor follow the trajectory can be written as -

$$\sum_{n=0}^{N} (z_n - z_n^*)^T Q_n (z_n - z_n^*) + (u_n - u_n^*)^T R_n (u_n - u_n^*)$$

subjected to

$$\bar{z_{n+1}} = A_n \bar{z_n} + B_n \bar{u_n}$$

$$Q_n = \begin{bmatrix} 1e+05 & 0 & 0 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1e+05 & 0 & 0 & 0 \\ 0 & 0 & 0 & 50 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1e+05 & 0 \\ 0 & 0 & 0 & 0 & 0 & 50 \end{bmatrix}$$

$$r_n = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}$$

By solving Riccati's recursive equations control gains(k_n) and feed forward gain(k_n) can be calculated.

$$\bar{(u)} = K_n * (\bar{x}) + k_n$$

$$u = K_n * (x - x^*) + u^* + k_n$$

Function in code-**trajectoryController(self,state,i)**

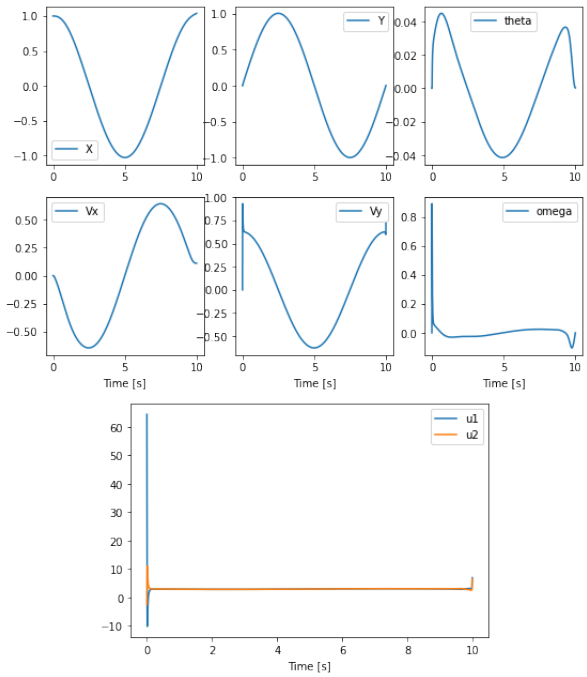

Figure 5: Plot of trajectory controller without disturbance

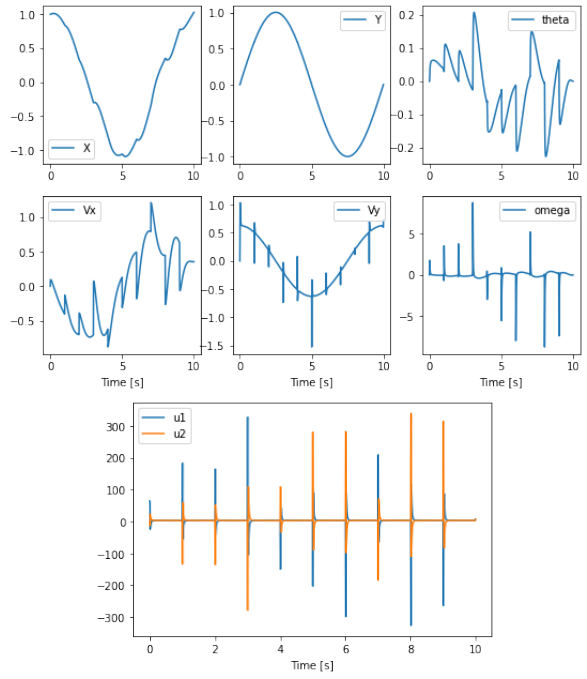

Figure 6: Plot of trajectory controller with disturbance

When there are no disturbances the quadrotor is completely following the trajectory with θ very close to 0. During the disturbances θ is not more than 0.2 away from the desired θ.
Given trajectory lq controller can be easily implemented without much tuning of Q and R. And it can be seen from the disturbance plot that it's quite robust to the noise. Downside can the path need to be predefined. And the it system goes far away from the desired state linearization will not work and it wouldn't be able to follow trajectory in the following time steps.

It is not possible for the drone to go in a circle with $\theta = pi/4$ orientation because at that θ velocity in x and y direction will have same sign at any time which is required for it to follow a circle. When moving in anticlockwise direction both the velocities should be negative in second quadrant and positive in fourth quadrant.

4 iLQR

Cost function for desired movement -

$$\sum_{n=0}^{N}(z_n - z_n^*)^T Q_n (z_n - z_n^*) + (u_n - u_n^*)^T R_n (u_n - u_n^*)$$

where,

$$z_n^* = \begin{cases} 0 & \text{if} t < 4.5 and t > 5.5 \\ z_v & \text{otherwise,} \end{cases}$$

$$z_v = \begin{bmatrix} 3 \\ 0 \\ 3 \\ 0 \\ \pi/2 \\ 0 \end{bmatrix}$$

$$Q_n = \begin{cases} Q_1 & \text{if} t < 4.5 and t > 5.5 \\ Q_2 & \text{otherwise,} \end{cases}$$

$$Q_1 = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} 100 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$R_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Approximation of cost can be written as -

$$Q_n = 2 * Q_n$$

$$R_n = 2 * R_n$$

$$q_n = Q_n(z^* - z_v)$$

$$r_n = R_n(u^* - u_v)$$

Task 1

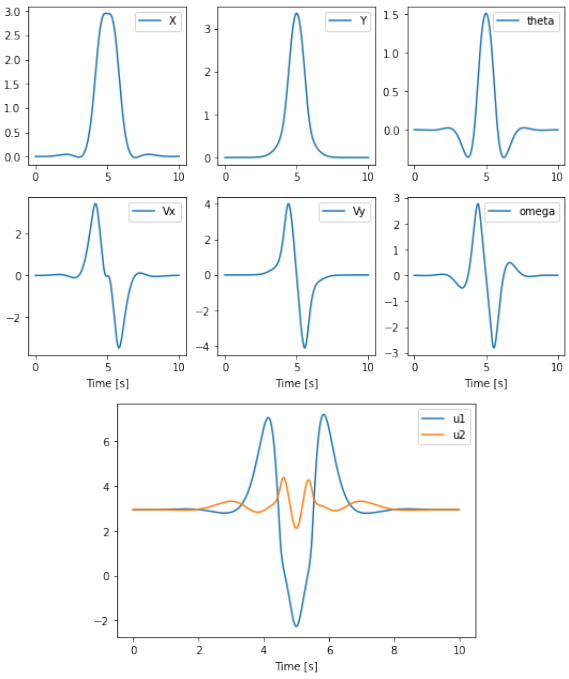

Figure 7: State and control plots for the task 1

Function in code-**ilqr1(self,state,i)**

The following approach doesn't require a predefined trajectory for the system. It is rather dependent on the cost function of the system. Main benefit with this approach is system can create a trajectory from any position to reach the desired position. Downside of the approach is its heavy dependence on the cost function and tuning. In a real time scenario running many iteration to get a desired trajectory might not be possible always.

Task 2

Cost function for desired movement -

$$\sum_{n=0}^{N}(z_n - z_n^*)^T Q_n (z_n - z_n^*) + (u_n - u_n^*)^T R_n (u_n - u_n^*)$$

where,

$$z_n^* = \begin{cases} 0 & t < 4.5 \\ z_{v_2} & t > 5.5 \\ z_v & \text{otherwise} \end{cases}$$

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

$$z_{v_2} = \begin{bmatrix} 3 \\ 0 \\ 0 \\ 0 \\ pi \\ 0 \end{bmatrix}$$

$$Q_n = \begin{cases} Q_1 & t < 4.5 \\ Q_3 & t > 5.5 \\ Q_2 & \text{otherwise,} \end{cases}$$

$$Q_1 = \begin{bmatrix} 5 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$Q_2 = \begin{bmatrix} 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 200 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$Q_3 = \begin{bmatrix} 12.25 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 23.2500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 201.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

$$R_n = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Approximation of cost can be written as -

$$Q_n = 2 * Q_n$$

$$R_n = 2 * R_n$$

$$q_n = Q_n(z^* - z_v)$$

$$r_n = R_n(u^* - u_v)$$

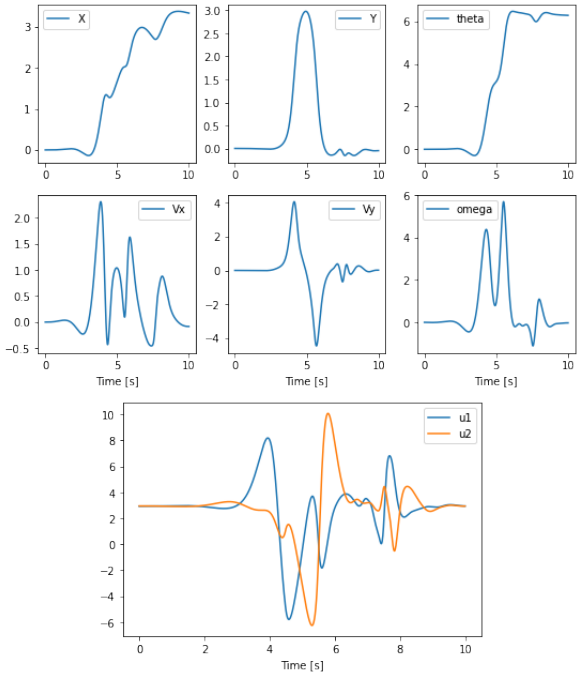

Figure 8: State and control plots for the task 2

Function in code-**ilqr1(self,state,i)**

To run a iLQR controller on any robot initially has

to be run on a simulation platform and trajectory to desired positions have to be computed. Then it can be used on the robot.