Optimal and Learning Control for Robotics - Final Project

Smrithi, srt381

Objective:

The goal of the project was to make a 2D quadrotor go through a window using an iLQR algorithm. In this project 2 ways have been implemented to solve the problem: 1. Generate a trajectory based on a known window position 2. Consider the area outside the window as an obstacle so that the quadrotor moves through the window.

Introduction:

Given a linear system dynamics and quadratic cost the LQR algorithm tries to find a control law that minimizes the cost. While LQR may be used to control a system with non linear dynamics by linearizing it, this usually considers linearization around a single point. If the system is highly non linear, this approximation may generate an inaccurate solution. Iterative LQR is an extension of the LQR which can be used for systems with non linear dynamics and/or nonlinear costs as it involves applying LQR iteratively by linearizing the system at different points along the trajectory and find controls that minimize the cost.

Dynamic Model: The dynamic model for a 2d quadrotor can be given by the following equations:

$$m\ddot{y} = -(u_1 + u_2)sin\theta$$

$$m\ddot{z} = (u_1 + u_2)cos\theta - mg$$

$$I\ddot{\theta} = r(u_1 - u_2)$$

Where m is the mass of the quad rotor, I is the inertia, g is the acceleration due to gravity. u_1 and u_2 are the control inputs which correspond to the thrust generated by each motor.θ is the angle between the quadrotor base (body fixed y axis) and the y axis of the inertial frame of reference. r is the distance from the centre to the motor along the body fixed y axis.

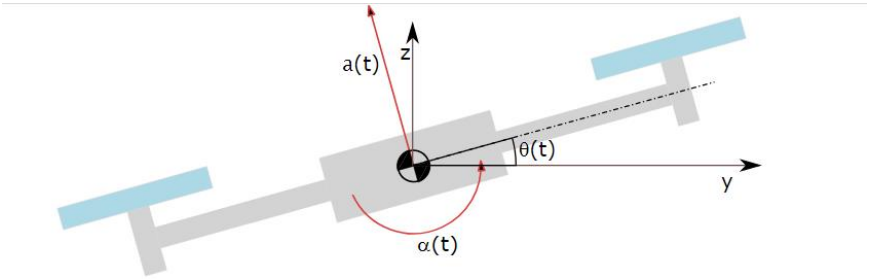


Figure 1 2D Quadrotor model. Source:https://moodle.fel.cvut.cz/pluginfile.php/84014/mod_assign/intro/2d_quadrotor_rotated.svg

Method:

The general ILQR algorithm involves the following steps:

Given an arbitrary (linear/non linear cost) and non linear system dynamics

$$\min_{u_0,u_1,\dots,u_T} l_T(x_T) + \sum_{t=0}^{N-1} l_t(x_t, u_t)$$

$$x_{n+1} = f(x_n, u_n)$$

1. Initialize the initial state x_0 and control sequence $U = u_0^*, u_1^*, \dots, u_{T-1}^*$
2. Integrate the system with current control guess and. i.e find nominal trajectories X $=x_0^*, x_1^*, \dots, x_T^*$
3. Linearize the dynamics and get a quadratic approximation of the cost
4. Do a backward pass using backward Ricetti equations to calculate the gain values
5. Do a forward pass – calculate the updated control input and updated trajectory.
6. Go back to step 2, until convergence.

2 different cases of the above algorithm have been considered for the 2 different solutions of passing through a window.

It is important to choose a good initial control sequence as the algorithm's performance and convergence depends on it. Hence for both the cases, a simple finite horizon LQR was used to get the initial first guess. For this, the dynamics were linearized about an equilibrium point considered to the hover state of the quadrotor. i.e., state $x = [y_k, z_k, 0,0,0,0]^T$ and control $u = \left[\frac{mg}{2}, \frac{mg}{2}\right]^T$ (The chosen states and control will be explained in the next section)

States, Controls and Linearization:

Lets consider the states of the quadrotor as $x = [y, z, \theta, v, k, \omega]^T$. Here the first 2 values are the position (y, z) in space, θ is the angle the quadrotor makes with respect to inertial frame, so they give the information of the pose and the next 3 values are the velocities $(\dot{y}, \dot{z}, \dot{\theta})$ respectively.

Therefore,

$$\dot{x} = \left[\dot{y}, \dot{z}, \dot{\theta}, \dot{v}, \dot{k}, \dot{\omega}\right]^T = f(x, u)$$

The above system is linearized using Taylor series expansion:

$$\dot{\mathrm{x}} = \mathrm{f}(\mathrm{x}, \mathrm{u}) \simeq \mathrm{f}(\bar{\mathrm{x}}, \bar{\mathrm{u}}) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} (\mathrm{x} - \bar{\mathrm{x}}) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} (\mathrm{u} - \bar{\mathrm{u}})$$

The above equation can further be discretized and written as

$$\frac{\Delta x}{\Delta t} = \mathrm{f}(\mathrm{x}, \mathrm{u}) \simeq \mathrm{f}(\bar{\mathrm{x}}, \bar{\mathrm{u}}) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} (\mathrm{x} - \bar{\mathrm{x}}) + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} (\mathrm{u} - \bar{\mathrm{u}})$$

$$\Rightarrow \delta x_{n+1} - \delta x_n = \Delta t \left(\left.\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} \delta x_n + \left.\frac{\partial \mathrm{f}}{\partial \mathrm{u}}\right|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} \delta \mathrm{u}_\mathrm{n} \right) \Rightarrow$$

$$\delta x_{n+1} = \delta x_n + \Delta t \left(\frac{\partial \mathrm{f}}{\partial \mathrm{x}}\Big|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} \delta x_n + \frac{\partial \mathrm{f}}{\partial \mathrm{u}}\Big|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}} \delta \mathrm{u_n} \right)$$

The linear system can then be written as $\delta x_{t+1} = A_t\, \delta x_t + B_t \delta u_t, where\ A = I + \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{x}}\Big|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}}\ and\ B = \Delta t \frac{\partial \mathrm{f}}{\partial \mathrm{u}}\Big|_{\mathrm{x}=\bar{\mathrm{x}},\mathrm{u}=\bar{\mathrm{u}}}$

The resulting A and B matrices were

$$A_t = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & -\frac{\Delta t}{m}(u_1+u_2)cos\theta & 1 & 0 & 0 \\ 0 & 0 & -\frac{\Delta t}{m}(u_1+u_2)sin\theta & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, B_t = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{m}sin\theta & -\frac{1}{m}sin\theta \\ \frac{1}{m}cos\theta & \frac{1}{m}cos\theta \\ \frac{r}{I} & -\frac{r}{I} \end{bmatrix}$$

Going through a Window:

As the quadrotor moves through the environment, it is possible to calculate the position of the object built sensors like cameras, lidars or through algorithms like SLAM.

Case 1:

Knowing the location of the window (in our case in 2d plane – (y_w, z_w), the start point (y_s, z_s) and the goal location (y_g, z_g) , we can first generate a trajectory and make the quadrotor follow the trajectory. The windows centre can be considered as the point to go through. Here a function called interp1d was used from scipy library to generate a trajectory. Example:

$$y = interp1d\left(\left[0, \frac{t}{6}, \frac{t}{4}, \frac{t}{2}, t\right], \left[y_s, y_w, y_w, y_g, y_g\right]\right);$$

$$z = interp1d\left(\left[0, \frac{t}{6}, \frac{t}{4}, \frac{t}{2}, t\right], \left[z_s, z_w, z_w, z_g, z_g\right]\right)$$

(The rest of the states were considered as zero).

This then becomes a trajectory generation problem. The cost function considered here was:

$$\min_{u_0, u_1, \ldots, u_T} \frac{1}{2}(x_T - x_T^*)^T Q (x_T - x_T^*) + \frac{1}{2}\sum_{t=0}^{T-1} (x_t - x_t^*)^T Q (x_t - x_t^*) + (u_t - u_t^*)^T R (u_t - u_t^*)$$

Approximated to

$$J \simeq \mathrm{q_T^T}\bar{x}_T + \frac{1}{2}\bar{x}_T^T Q \bar{x}_T + \sum_{t=0}^{T-1} \mathrm{q_t^T}\bar{x}_t + \frac{1}{2}\bar{x}_t^T Q \bar{x}_t + \mathrm{r_t^T}\bar{u}_t + \frac{1}{2}\bar{u}_t^T R \bar{u}_t$$

Where $\bar{x}_t = (x_t - x_t^*),\ \bar{u}_t = (u_t - u_t^*)$

With the above mentioned quadratic approximation of cost function and linear approximation dynamics, ILQR algorithm is applied with the backward pass solved using Backward Riccati recursion:

Initialize $P_T = Q_T, p_T = q_T$

Iterate from T - 1 to 0

$$K_t = -(R_t + B_t^T P_{t+1} B_t)^{-1} B_t^T P_{t+1} A_t$$

$$P_t = Q_t + A_t^T P_{t+1} A_t + A_t^T P_{t+1} B_t K_t$$

$$k_t = -(R_t + B_t^T P_{t+1} B_t)^{-1} (B_t^T p_{t+1} + r_t)$$

$$p_t = q_t + A_t^T p_{t+1} B_t k_t$$

The control can be given by

$$u_t = u_t^* + K_t(x_t - x_t^*) + k_t$$

$$x_{t+1} = f'(x_t, u_t)$$

Case 2: Consider area outside the window as an obstacle, so that the quadrotor can be thought of as going between 2 obstacles (planar case).

In this case the quadrotor must maximize its distance from the obstacle but also find an optimal control sequence to go from point start to the goal location.

In this case, the obstacle was considered to be a circle since this is the simplest case and the distance from the quad to the centre of the circle needs to be maximized. For maximizing the distance from the obstacle, the instantaneous cost (g) considered was $\gamma \sum_i \exp(-d_i)$ where, 'i' refers to the i^{th} obstacle and $d_i = \sqrt{(y - y_o)^2 + (z - z_o)^2}$ is the distance from the centre of the obstacle to the quadrotors location and gamma is a positive scalar. The minimization of distance from goal is also considered in new cost.

The new cost function can be given as

$$J = l_f + \sum_0^{T-1} l_t = \min_{u_0, u_1, \ldots, u_T} \frac{1}{2}(x_T - x_g)^T Q (x_T - x_g) + \frac{1}{2} \sum_{t=0}^{T-1} \left((x_t - x_g)^T Q (x_t - x_g) + u_t^T R u_t + \gamma \left(\sum_i \exp(-d_i) \right)_t \right)$$

This cost is non linear and is linearized using 2nd order Tayler series expansion into a quadratic approximation of the cost.

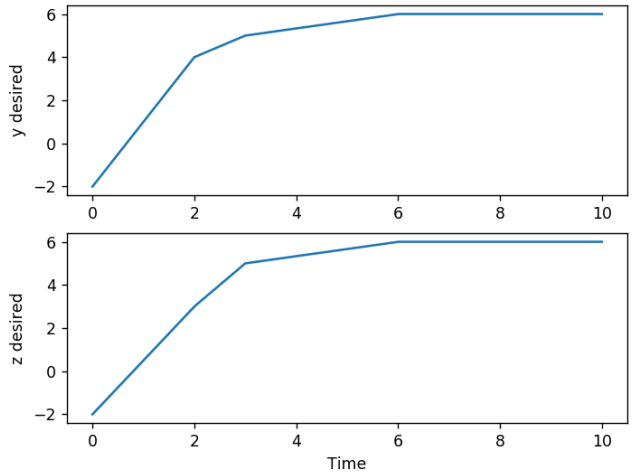
The same ILQR backward pass as above (Case 1) was applied but with a slight variation and a slightly different notation. The material was followed from [1]. (On solving the backward pass equations from [1] and comparing them with the above version, they were found to be exactly the same because there is no term in the linearized cost that is dependent on both x and u i.e., l_{ux} or $\frac{\partial^2 l}{\partial u \partial x} = 0$, except the regularization part). On linearizing the cost and applying iLQR

the system did not converge as expected due to singularity. To solve this regularization was added.

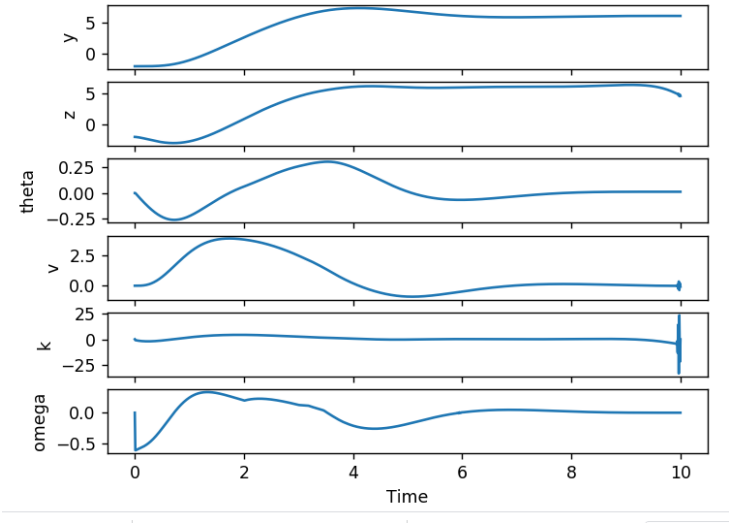
Results:

Some of the main results are presented here.

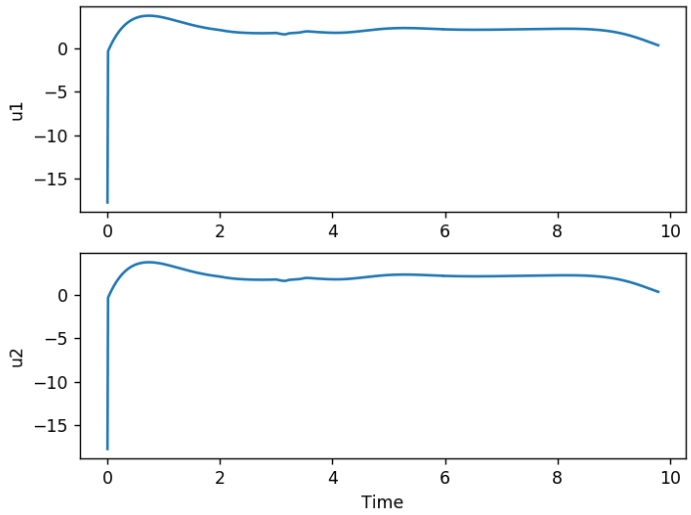
1. Obstacle Avoiding Given Trajectory
   Desired Trajectory:



Obtained States:

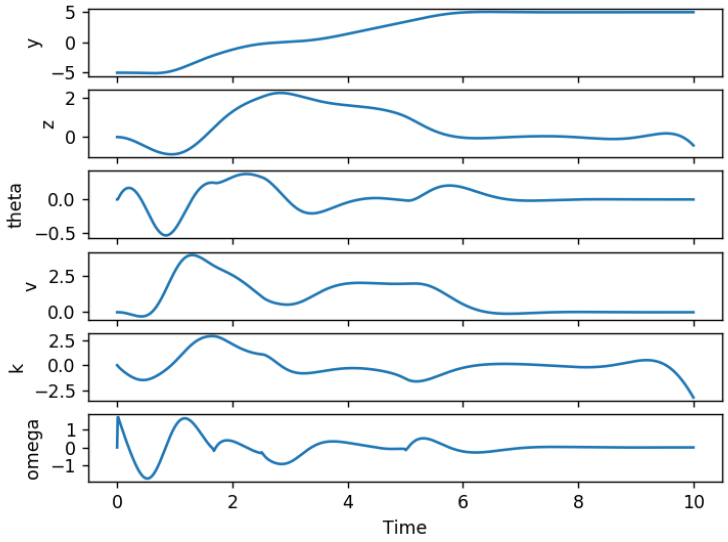


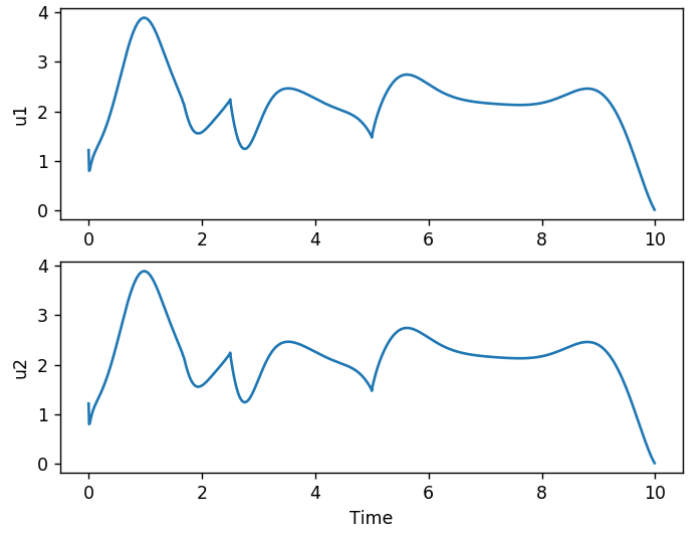
Controls:



2. Known Location of Window – Given start, goal and window locations

$$y_{goal}, z_{goal} = 5,0;\ y_{start}, z_{start} = -5,0;\ y_{window}, z_{window} = 0,2$$

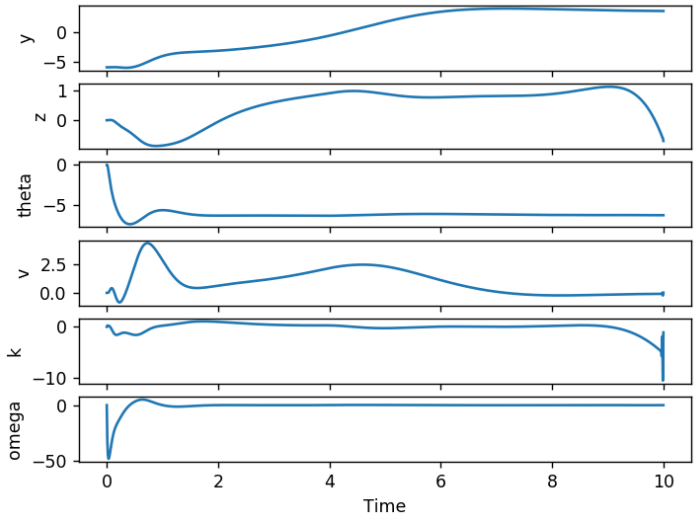


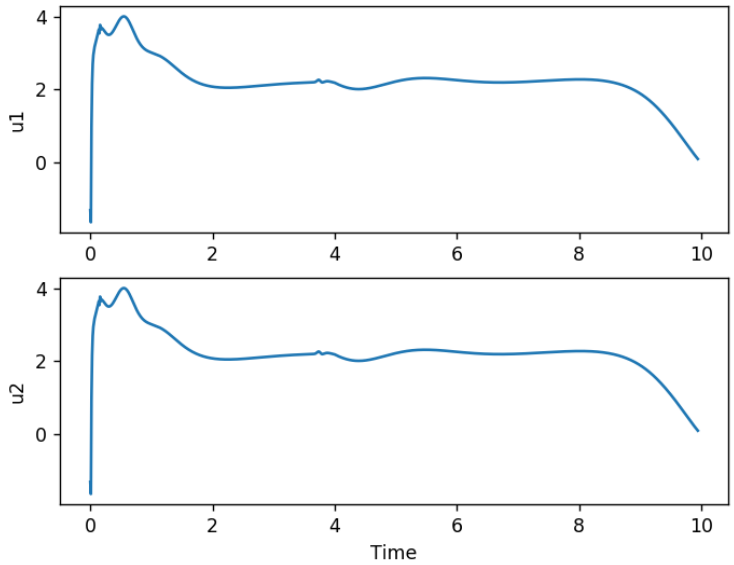


3. Going between Obstacles – Consider area outside window as obstacle:

$$y_{goal}, z_{goal} = 4,1;\ y_{start}, z_{start} = -6,0$$
$$x_{obs} = [(0,-2),(0,-1),(0,0),(0,3),(0,4),(0,5)]$$

Multiple obstacles in a line to consider a large obstacle – Quad goes between (0,0) *and* (0,3)





Using iterative LQR we were able to make the quadrotor pass follow any desired trajectory after which obstacle avoidance was tested and was extended to the 2 cases where the quadrotor goes through a window knowing the location of the window and considering area outside the window as obstacle.

A few issues were observed during the process:

1. The main problem is the tuning of parameters like Q, R, γ manually for every case.
2. Though the algorithm worked, towards the end of the horizon there was a lot of jitter. One way to solve this is by extending the horizon length a little beyond the actually needed horizon and ignore the end.

3. Though we were able to make the quad go between obstacles, the obstacles were considered as points objects. Tuning γ gave the scope to increase of decrease the threshold and this is by know means the end. This can be extended to considering whole objects as obstacles which requires further research.

Reference:

[1] https://rexlab.stanford.edu/papers/iLQR_Tutorial.pdf