# Course Project Report, Phase 1

Chintan Desai[*], Rohan Walia[†]

22 November, 2020

## Abstract

The objective of this work is to design a linear controller for a Quadrotor to stabilize it in hover and also to enable it to effectively follow a trajectory, using the Linear-Quadratic Regulator (LQR) control strategy. The report includes a derivation of the mathematical model of Quadrotor dynamics using Newton's and Euler's laws and linearization of the model using small angle approximation. The linearized model is then used to calculate the optimal gain matrix K by setting up appropriate Q and R cost matrices in the cost optimization function. The obtained K is then fed to the nonlinear model of Quadrotor dynamics for stabilization and following a trajectory. The trajectory followed by the Quadrotor is visualized in 3D using the plot3 function in MATLAB. The work also details a comparison of LQR with the classical PID control strategy.

## 1 Introduction

Quadrotors have gained popularity in the last decade or so as the go-to solution for myriad problems ranging from something as critical as border security and reconnaissance to something as banal as agricultural surveys and crop health monitoring. These endeavors require a thorough understanding of not only how Quadrotors work but also how to control them. The task of a control engineer is then to design a controller according to the application, which could be a Quadrotor used for racing where the time response has to be in split-seconds, or a relatively dreary but a very heavy Quadrotor used for spraying pesticides on a farmland.

Use of Quadrotors for surveillance is among their most practical applications given their agility and range of operation. Due to the proliferation of UAVs, it is not too far into the future when miscreants would have easy access to these capable vehicles and use them for mischief. It is thus the need of the hour to think about strategies to counter such rogue UAVs intended to be used for malicious purposes.

The objective of this work is to design a control strategy to enable a Quadrotor to follow and capture a rogue target UAV that intrudes in its airspace in the least possible time. The most common control strategy in use currently is the classical PID control which requires tuning of individual P, I and D gains for each individual axis of motion, namely, roll, pitch and yaw. This controller suffers in case of strong external perturbations.[1]

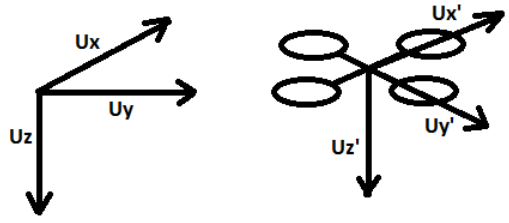

Figure 1: Inertial and Body reference frames.

The optimal control approach discussed in this report, called the Linear-Quadratic Regulator (LQR) allows us to think of the control problem in terms of an optimization problem. This is because of the way the optimal gain matrix K is derived by minimizing the quadratic cost function. It considers the expense of providing inputs with respect to the expense of getting to the desired output states. This is particularly helpful when it comes to control of a Quadrotor because a Quadrotor is severely resource constrained in terms of number of actuators and battery back-up. The constraint on energy forces us to think of the cost of expending an input, i.e. thrust generated by motors, at the expense of generating a response, i.e. getting the Quadrotor to stabilize or moving form one point to another. This is an optimal control approach that not only takes into consideration the time response but also the trade-off relationship between the input and output.[2]

## 2 Methodology

We begin by deriving a mathematical model of the Quadrotor dynamics. However, it is important to setup appropriate reference frames - the inertial frame and the body frame of the Quadrotor. The inertial frame or the world frame follows a North-East-down (NED) convention whereas the body frame of the Quadrotor follows the Aircraft Body Center (ABC) convention as shown in Figure 1.

The orientation of the Quadrotor is represented using

[*]cmdesai@wpi.edu
[†]rwalia@wpi.edu

Euler angles $\phi$, $\theta$ and $\psi$ where angle $\psi$ is the rotation around body z-axis, angle $\theta$ is the rotation around body y-axis and angle $\phi$ is the rotation around body x-axis. The rotation from body frame to the inertial frame is hence given by,

$$R_I^B = \begin{bmatrix} c\theta c\psi & s\phi s\theta c\psi - c\phi s\psi & c\phi s\theta c\psi + s\phi s\psi \\ c\theta s\psi & s\phi s\theta s\psi + c\phi c\psi & c\phi s\theta s\psi - s\phi c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{1}$$

The vector for linear and angular position of Quadrotor in inertial frame can be written as $[x\ y\ z\ \phi\ \theta\ \psi]^T$ and the vector for linear and angular velocities in the body frame as $[u\ v\ w\ p\ q\ r]^T$. The mathematical model can now be derived by using Newton's equations of motion and Euler's equations.

We use the state-space model of the system to design an LQR controller. The state vector is written as,

$$\dot{\chi} = [x\ y\ z\ \psi\ \theta\ \phi\ \dot{x}\ \dot{y}\ \dot{z}\ p\ q\ r]^T \in R^{12} \tag{2}$$

The equation of the Quadrotor in the state-space can be written as,

$$\dot{\chi} = f(x) + G(x)u \tag{3}$$

where,

$$f(x) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ q\frac{s\phi}{c\theta} + r\frac{c\phi}{c\theta} \\ qc\phi - rs\phi \\ p + qs\phi t\theta + rc\phi t\theta \\ 0 \\ 0 \\ g \\ \frac{(I_y - I_z)}{I_x}qr \\ \frac{(I_z - I_x)}{I_y}pr \\ \frac{(I_x - I_y)}{I_z}pq \end{bmatrix} \tag{4}$$

and

$$G(x) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ g_1^7 & 0 & 0 & 0 \\ g_1^8 & 0 & 0 & 0 \\ g_1^9 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix} \tag{5}$$

with,

$$g_1^7 = -\frac{1}{m}[s\phi s\psi + c\phi c\psi s\theta]$$
$$g_1^8 = -\frac{1}{m}[c\psi s\phi - c\phi s\psi s\theta]$$
$$g_1^9 = -\frac{1}{m}[c\phi c\theta]$$

This model is then linearized about the equilibrium point, taken to be the hover case, using small angle approximation to give matrices A and B as in equations [2.31 and 2.32] in [3]. The dynamics then are transformed in the form,

$$\dot{\chi} = Ax + Bu \tag{6}$$

Now that we have a linear model of the Quadrotor, we proceed to use LQR to obtain the optimal gain matrix K of the controller. This is done by choosing appropriate diagonal matrices R and Q, where R is the cost of using actuators and Q is the cost of states in the cost optimization function as in equation [3.5] in [3]. To begin with, the matrices R and Q are assumed to be identity matrices of appropriate dimensions to depict equal cost associated with all control input and desired output states of the system. The lqr function of MATLAB takes the matrices A, B, R and Q as the arguments to return the optimal gain matrix K.

The dynamics of the system are then converted in the form of error dynamics of the form,

$$\dot{e} = -f(x_d - x) - G(x_d - x)u(e) \tag{7}$$

where,

$$u = -Ke \tag{8}$$

Using these as the arguments for the ODE45 function then returns the system response plot as in the following section, which can be modified by altering the cost matrices R and Q to give priority to either economy in the use of actuators or the quickness in system response.

In this implementation, a **mg** term was added to the z-direction state in order to account for increasing steady state error. [1]

## 3 Simulation Results

Multiple scenarios were utilized to test the system. For each scenario, a 12 x n matrix (**x_d**) containing trajectory data was created beforehand, where n is the number of points in the trajectory excluding the origin. Each column of this matrix contains system states for points in the trajectory. At each point in the trajectory, all desired system states except the x, y and z coordinates are zero as the system is assumed to be in a 'hover state'. A time vector is also provided to the system telling the Quadrotor when to reach the corresponding points in the trajectory matrix.

The following scenarios were chosen to study system behavior -

**Scenario 1) Points on straight line**

This scenario was used to tune the controller based on how long it takes to cover a euclidean distance of 1 unit. The trajectory and time matrices for this case are as follows -

$$x=\begin{bmatrix}0 & 1\end{bmatrix}\ y=\begin{bmatrix}0 & 1\end{bmatrix}\ z=\begin{bmatrix}0 & 1\end{bmatrix}\ \text{timeVector}=\begin{bmatrix}0 & 5\end{bmatrix}$$

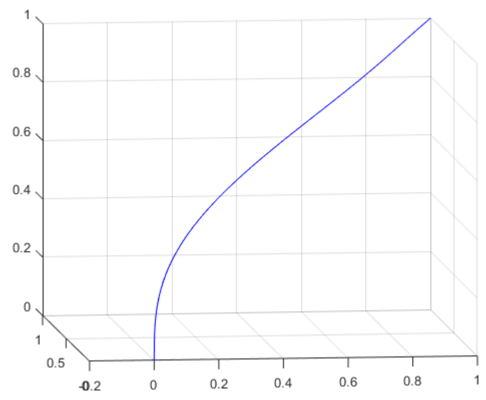

Figure 2: Path taken to go from [0 0 0] to [1 1 1]

Using an iterative tuning process,it was found that the fastest settling times were achieved using the following values for Q and R -

$$Q = 1000 I_{12},\ R = 0.01 I_4$$

The corresponding settling times for x, y and z directions were recorded in seconds as -

X: 4.3446 Y: 4.3489 Z: 3.9125

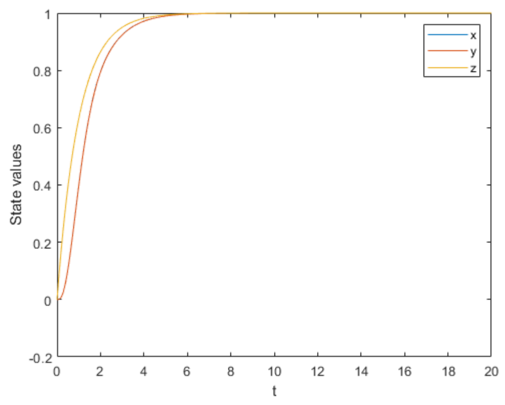

Figure 3: Step response for going from [0 0 0] to [1 1 1] in 5 seconds

**Scenario 2) UAV escapes**

This scenario utilizes a custom path to show system behavior when the target UAV is faster than the Quadrotor. The trajectory and time vectors for this case are as follows -

$$x=\begin{bmatrix}1 & 3 & 11\end{bmatrix}$$
$$y=\begin{bmatrix}3 & 2 & 5\end{bmatrix}$$
$$z=\begin{bmatrix}3 & 4 & 2\end{bmatrix}$$
$$\text{timeVector}=\begin{bmatrix}0 & 1 & 2 & 3\end{bmatrix}$$

Following is the trajectory plot for this scenario -

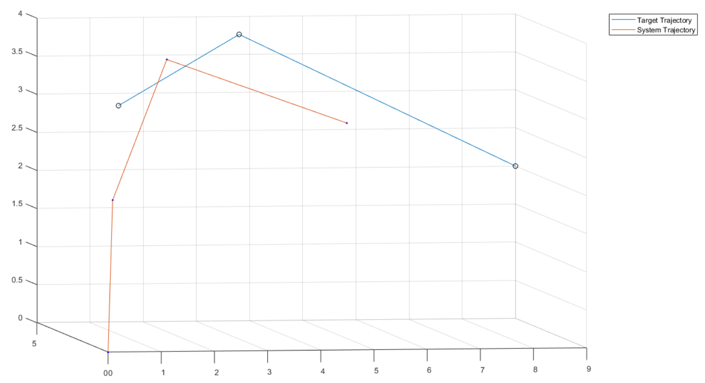

Figure 4: Path taken for Scenario 2

The black **o**'s represent the desired trajectory, the blue dots represent intermediate points along the trajectory, and the red line represents the path taken.

In this scenario, the Quadrotor starts from the nest, and tries to capture the UAV. The UAV appears at [1 3 9] and disappears at [3 4 2] for the simulation. However, the Quadcopter response is slow and the UAV escapes before the Quadcopter can reach the desired points at the specified schedule.

**Scenario 3) Capturing the UAV**

This scenario utilizes a more aggressive path but with relaxed time constraints. The trajectory and time vectors for this case are as follows -

$$x=\begin{bmatrix}9 & 1 & 3 & 3\end{bmatrix}$$
$$y=\begin{bmatrix}9 & 3 & 2 & 3\end{bmatrix}$$
$$z=\begin{bmatrix}9 & 3 & 4 & 4.2\end{bmatrix}$$
$$\text{timeVector}=\begin{bmatrix}0 & 5 & 10 & 15 & 20\end{bmatrix}$$

The UAV starts at [9 9 9] and gets captured at [3 3 4.2] (shown by the magenta circle marker). After the point of capture, the paths of the UAV and Quadrotor are modified to include the nest (origin) as the final point in both the system, and the UAV trajectory.

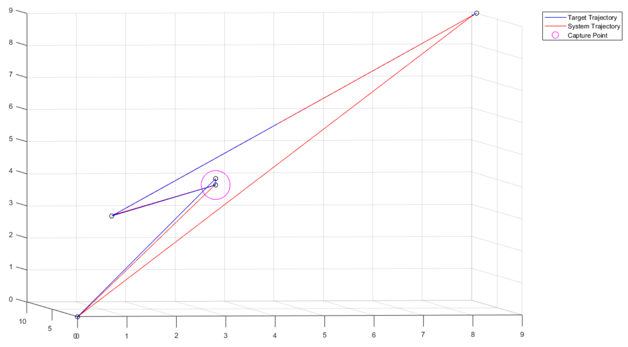

Figure 5: Path taken for Scenario 3

**Scenario 4) Out of bounds**

This scenario shows that the Quadrotor returns to the nest if it encounters any point in the target trajectory that is out of bounds. The target trajectory and time vectors for this scenario are -

$$x=\begin{bmatrix}1 & 3 & 11\end{bmatrix}\ y=\begin{bmatrix}3 & 2 & 5\end{bmatrix}\ z=\begin{bmatrix}3 & 4 & 2\end{bmatrix}$$
$$\text{timeVector}=\begin{bmatrix}0 & 5 & 10 & 15\end{bmatrix}$$

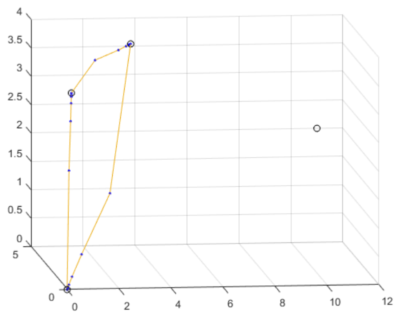

Figure 6: Path taken for Scenario 4

The black dots represent points on the target path. Notice that the last point is out of bounds (x coordinate is greater than 10). The yellow line shows the trajectory of the Quadrotor. As soon as the system detects a point in the target trajectory that is out of bounds, it modifies the system trajectory to make the last point to be the nest ([0 0 0]).

## 4 Conclusions

The simulation results show that the LQR controller is successful in stabilizing the Quadrotor and effective in trajectory tracking. However, it is observed that the time response of the controller suffers due to lack of control over each individual state. Making a change in the optimal gain matrix K has an effect on all states. Also, R and Q cost matrices are obtained by an iterative method and aren't easily optimized to obtain desired system performance. Comparing this to classical PID control, the PID gains of each state provide a way to effectively alter the time response of each individual state. This sort of a control strategy has many advantages but it is difficult to tune and suffers in case of extreme external perturbations. The weaknesses such as slow time response of the LQR controller could be overcome by coupling it with a nonlinear control technique such as feedback linearization.[3]

One advantage of using LQR over other control methods, such as PID control, is the apparent trade-off between importance given to how much control inputs should be consumed (tuning R) vs how fast the desired system states (tuning Q) should be reached. As discussed before, this approach is more intuitive while tuning the system when it is dependent on using a particular resource sparingly (in this case the input force). In a PID control scenario, the system would have to be tuned based on time response, with no direct connection to how much input force is applied. [2]

One disadvantage of using an LQR controller is the missing integral control. Without this term, the system states would not settle due to the steady state error. In this implementation, an **mg** term was added to the z-direction state in order to account for increasing steady state error. [2]

In this implementation, a linearized model of the system was used to obtain a stable equilibrium point for designing a full-state feedback controller. As soon as the system goes out of these 'linear' bounds, the system would become unstable. To account for more realistic scenarios, the controller could be linearized around each state instead of around a single equilibrium point. [1]

## References

[1] A. Noth S. Bouabdallah and R. Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. *In International Conference on Intelligent Robots and Systems*, 2004.

[2] Christopher Lum. Introduction to linear quadratic regulator (lqr) control.

[3] Francesco Sabatino. Quadrotor control: modeling, nonlinear control design, and simulation. *KTH Electrical Engineering, Master's Degree Project*, 2015.