AE 353 Design Project 3

Ganesh Ravisankar
Department of Aerospace Engineering, University of Illinois at Urbana-Champaign

The objective of this design project is to design and develop a control law for a drone such that it passes through an obstacle course of rings in the shortest amount of time. In order to achieve this objective, the techniques of a linear quadratic regulator (LQR) were employed. The controller and simulation were both implemented in a Jupyter Notebook titled `GenerateResults.ipynb`.

I. Introduction

The drone considered in this design project is a quadrotor. The quadrotor has the capability to move in three dimensions. Due to its ability to move freely in three dimensions, the quadrotor's configuration at a given point in time is wholly specified by six parameters: $p_x, p_y, p_z, \phi, \theta$, and ψ. p_x, p_y, and p_z specify the drone's location in space. ϕ, θ, and ψ specify the drone's orientation at its location.

The objective of this report is to detail the theory, design, and implementation of a controller for a quadrotor. The function of this controller is to enable the drone to complete the course in the shortest amount of time possible without any collisions. The controller will rely on the implementation of trajectory tracking and a linear quadratic regulator (LQR) to achieve optimal transient stability and speed.

II. Background Theory

The objective of this design project is to enable the drone to direct itself through the course efficiently. The drone's internal state is denoted by x while the input torque and thrust from rotors is encapsulated in u.

A. Equations of motion

Upon looking closely at the problem statement, it is noted that the influence of gravity is relevant in this problem. The drone flies freely in 3-dimensional space and it is imperative that the drone is able to accomplish a controlled ascent and descent without crashing. The drone has six degrees of freedom: spatial movement, pitch, roll, and yaw. As such, the system dynamics of the drone itself have six independent variables corresponding to spatial movement, pitch, roll, and yaw. These six independent variables characterize 12 internal states of the quadrotor, indicating the system is non-holonomic. Below are the differential equations governing the motion of the drone:

$$\begin{bmatrix} \dot{p}_x \\ \dot{p}_y \\ \dot{p}_z \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{v}_x \\ \dot{v}_y \\ \dot{v}_z \\ \dot{w}_x \\ \dot{w}_y \\ \dot{w}_z \end{bmatrix} = f\left(p_x, p_y, p_z, \phi, \theta, \psi, v_x, v_y, v_z, w_x, w_y, w_z, \tau_x, \tau_y, \tau_z, f_z\right) \tag{1}$$

The function f itself is given by:

$$f\left(p_x, p_y, p_z, \phi, \theta, \psi, v_x, v_y, v_z, w_x, w_y, w_z, \tau_x, \tau_y, \tau_z, f_z\right) = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \frac{w_x \cos(\psi) - w_y \sin(\psi)}{\cos(\theta)} \\ w_x \sin(\psi) + w_y \cos(\psi) \\ -w_x \cos(\psi) \tan(\theta) + w_y \sin(\psi) \tan(\theta) + w_z \\ 2 f_z \sin(\theta) \\ -2 f_z \sin(\phi) \cos(\theta) \\ 2 f_z \cos(\phi) \cos(\theta) - \frac{981}{100} \\ \frac{10000\tau_x}{23} - \frac{17 w_y w_z}{23} \\ \frac{10000\tau_y}{23} + \frac{17 w_x w_z}{23} \\ 250\tau_z \end{bmatrix} \tag{2}$$

The function f was obtained by stacking the results of computing linear rates, angular rates, forces, and Euler's equations. It is desirable to obtain a system of the form

$$\dot{x} = Ax + Bu \tag{3}$$

where:

$$x = \begin{bmatrix} p_x - p_{xe} \\ p_y - p_{ye} \\ p_z - p_{ze} \\ \phi - \phi_e \\ \theta - \theta_e \\ \psi - \psi_e \\ v_x - v_{xe} \\ v_y - v_{ye} \\ v_z - v_{ze} \\ w_x - w_{xe} \\ w_y - w_{ye} \\ w_z - w_{ze} \end{bmatrix} \tag{4}$$

and

$$u = \begin{bmatrix} \tau_x - \tau_{xe} \\ \tau_y - \tau_{ye} \\ \tau_z - \tau_{ze} \\ f_z - f_{ze} \end{bmatrix} \tag{5}$$

The input u can also be expressed in the form $-Kx$, where K is the gain matrix (explained in an upcoming section).

An observer will also be designed. An observer is a system that estimates the state of the drone by measuring the input (u) and output $y = Cx$. The error e is defined below:

$$\dot{e} = (A - LC)e \tag{6}$$

where $e = \hat{x} - x$ and $\dot{e} = \dot{\hat{x}} - \dot{x}$.

The system can be linearized in two steps:

1) First, choose a suitable equilibrium point. At the equilibrium point, it is desirable that the function f drops out. This means that the system dynamics described by $\dot{x}$ also drops out. A trivial equilibrium point would be when all states are set to 0. In a system containing forces, equilibrium requires that net force $F_{net} = 0$. In order to balance out all forces, we set $f_{ze} = W_{drone} = g/2$ where g is the gravitational constant and W_{drone} is the weight of the drone. All other equilibrium values are set to zero.
2) Second, obtain the matrices A, B, and C by taking the Jacobian of f (for matrices A and B) and g (for matrix C) at the desired equilibrium point.

After performing the two steps above, the following are the matrices A, B, and C:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.0 \\ 434.782608695652 & 0 & 0 & 0 \\ 0 & 434.782608695652 & 0 & 0 \\ 0 & 0 & 250.0 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

B. Verifying controllability

In order to obtain a system for which the state can be manipulated through a linear relationship, the system first needs to be controllable. A system is controllable if the controllability matrix W is invertible (i.e., is of full rank). The controllability matrix is defined below:

$$W = \begin{bmatrix} B & AB & A^2B & \dots & A^{n-1}B \end{bmatrix} \tag{7}$$

where n is the number of rows or columns in A. The matrix W was verified to be of full rank (using the NumPy method `numpy.linalg.matrix_rank()`), indicating that the system is controllable for the given value A and B. Verifying the controllability of the system is a significant step - a controllable system is one whose eigenvalues can be placed. It is thus possible to manipulate the rate at which the system reaches it equilibrium state. Furthermore, controllability implies asymptotic stability.

The controller for this drone was verified to be controllable and thus stable.

C. Verifying observability

In order to obtain a system for which the state can be manipulated through a linear relationship, the system first needs to be observable. A system is observable if the observability matrix W is invertible (i.e., is of full rank). The observability matrix is defined below:

$$O = \begin{bmatrix} C & CA & CA^2 & \dots & CA^{n-1} \end{bmatrix}^T \tag{8}$$

where n is the number of rows or columns in A. The matrix O was verified to be of full rank (using the NumPy method `numpy.linalg.matrix_rank()`), indicating that the system is controllable for the given value A and B. Verifying the observability of the system is a significant step - a controllable system is one whose eigenvalues can be placed. It is thus possible to manipulate the rate at which the system reaches it equilibrium state. Furthermore, observability implies detectability.

The observer for this drone was verified to be observable and thus detectable.

D. Note on controllers and observers

At this point, it will be clarified what is meant by a controller and an observer. A controller is a device that attempts to bring the state x to equilibrium through input u. It is given as $A - BK$. An observer is a device that attempts to infer the state x from y, where $y = Cx$. An observer is defined as $A - LC$. Here, both L and K are gain matrices.

E. The gain matrix

It was previously suggested that the input u could be written in form $u = -Kx$. The gain matrix K could be construed as a linear transformation that relates the input to state-space model. K answers the question of by how much the drone's internal state will change when an input is applied.

Similarly, the matrix L is also a gain matrix. However, since it applied to an observer, it is a linear transformation applied to the output error estimate:

$$\dot{e}_o utput = -L(C\hat{x} - y) \tag{9}$$

There are two ways to calculate K and L - pole placement and linear quadratic regulator (LQR). Pole placement can be done using the method `scipy.signal.place_poles()`. It is possible to select 'very negative' eigenvalues that will cause the system to quickly reach asymptotic stability. However, the path the controller sets for the drone using pole placement may not be the optimum one. Pole placement ensures reaching the correct destination asymptotically, but without regard for the transient stability. LQR is a solution to this as it will enable the drone reach its desired state quickly without compromising on transient stability.

Linear Quadratic Regulator (LQR)
LQR is given by the following integral:

$$\int_{t_0}^{t_f} x^T Q x + u^T R u \, dt \tag{10}$$

subject to:

$$\dot{x} = Ax\left(t\right) + Bu\left(t\right)$$

$$x\left(t_0\right) = x_0$$

The matrices Q, R, Q', and R' can be arbitrarily chosen:

$$Q^{12x12} = diag\left(q_i\right)$$

$$Q'^{6x6} = diag\left(q_i\right)$$

$$R = I^{4x4}$$

$$R' = I^{12x12}$$

For instance, by varying the entries in the matrix Q, one can obtain different gain matrices K. The pipe line for solving for the gain matrices of the drone for specified Q, R and Q', R' are found through the method `scipy.linalg.solve_continuous_are()`, a numerical method to find the solution to the algebraic Ricatti equation. The following code block demonstrates this:

```
P = la.solve_continuous_are(A, B, Q, R)
K = np.linalg.inv(R) @ B.T @ P
P1 = la.solve_continuous_are(A.T, B.T, Q1, R1)
L = np.linalg.inv(R1) @ B @ P1
```

III. Methods

A. Requirements

In this project, success and/or failure will be measured by several benchmarks.

1) The drone must be able to complete a course of 5 rings under 2 minutes (120 s)
2) The drone must not collide with any of the rings.
3) When the drone passes a ring, it must be approximately at the center of the ring `pos_ring` without any inclination ϕ, θ or ψ.

B. Verification

To verify the results, PyBullet will be used to simulate the robot. This will provide both a visual and numerical basis by which to verify the results produced in the Python notebook. At each time step, the position and inclination of the drone will be calculated. It can be visually confirmed that the drone is at the center at the ring for each of the five rings.

IV. Results and Evaluation

A. Results

The matrices Q, R, Q', and R' that were chosen are given below:

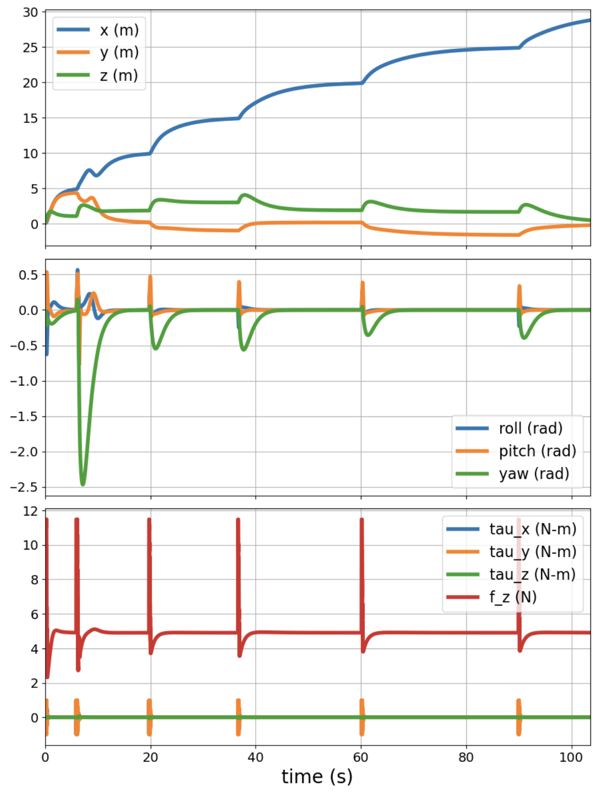
$$Q = \begin{bmatrix} 40000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 60000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 60000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} 5.0 & 0 & 0 & 0 \\ 0 & 5.0 & 0 & 0 \\ 0 & 0 & 5.0 & 0 \\ 0 & 0 & 0 & 2.5 \end{bmatrix}$$

$$Q' = \begin{bmatrix} 500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 500 & 0 & 0 & 0 & 0 \\ 0 & 0 & 500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 500 & 0 & 0 \\ 0 & 0 & 0 & 0 & 500 & 0 \\ 0 & 0 & 0 & 0 & 0 & 500 \end{bmatrix}$$

$$R' = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

It is observed that the values in the matrix Q and Q' are far larger than those in R and R'. This facilitates the drone to reach the position quickly by foregoing transient stability. The following are the graphs for the drone at the aforementioned values of the matrices:



We observe that a few details from the graph above:

1) The graph for the position of the drone has five curves that look similar to asymptotic function. These curves correspond to the drone asymptotically reaching the center point of each of the five rings. This matches the criteria for success
2) In the RPY graph, the roll, pitch, and yaw are all zero when the drone reaches a curve. This matches the criteria for success. Immediately after crossing the rings, there is a sudden change in the values for roll, pitch, and yaw. This is triggered by the fact that the values in the matrices Q are far larger than those in R. Due to this, the controller penalizes the state x heavily causing $x(t) \to x_e$ in finite time. Resultantly, the control $u(t) = -Kx(t)$ is penalized proportionally to $x(t)$.
3) Another observation that can be made is that the drone reaches each ring increasingly slower. A possible reason for this could be that the change in the roll, pitch, and yaw reduces with time. This results in slower transient speeds, causing the drone to reach the next ring in greater time.
4) Finally, the force f_{ze} applied at each ring is greater than the weight of the drone. This means that after crossing the ring, the drone will elevate slightly before crossing the next ring.

B. Evaluation

The overall performance of the drone is satisfactory as it is able to meet all design requirements. There is room for improvement for the speed, however. A significant problem to the design of this drone is that it has to meet asymptotic stability at each ring. This increases the amount of time that it has to spend at each ring. One way to overcome this problem is to plan the trajectory of the drone through the course beforehand. The trajectory could be a function of position and time that would allow the drone to smoothly pass through the ring to ensure stability. Furthermore, the orientation (RPY) of the drone could be removed as a criteria for crossing the ring. This way, the drone will not have to spend extra time to ensure it is at a horizontal configuration before proceeding to the next position.

V. Conclusion

The goal of this exercise was to design a controller for a drone that would ensure both stability and speed. This goal was met, with the drone completing the 5-ring course in 103.99 s. The theory and the results of the simulation that was conducted in PyBullet were discussed in length. Finally, the drone was evaluated for its performance and methods for improving its performance were considered.