Control of a Quadrotor

Rohen A. Agarwal*
University of Illinois at Urbana-Champaign, Champaign-Urbana, Illinois, 61801

Quadrotors (or drones) are proving to be increasingly useful in surveillance, deliveries, or just for the purpose to going from one point to another. In this project, my quadrotor will traverse through a randomly set arrangement of floating rings as obstactles in the least time possible to reach the end. I will design and implement a controller to do the same. This report outlines my process, test, and results.

I. Nomenclature

x	=	state of system
$\hat{x}$	=	state estimate of system
u	=	input matrix
y	=	output matrix
K	=	state feedback matrix
L	=	observer gains matrix
W	=	controllability matrix
O	=	observability matrix
p_x	=	x position (m)
p_y	=	y position (m)
p_z	=	z position (m)
Φ	=	roll angle (rad)
Θ	=	pitch angle (rad)
Ψ	=	yaw angle (rad)
v_x	=	linear velocity about z-axis (m/s)
v_y	=	linear velocity about z-axis (m/s)
v_z	=	linear velocity about z-axis (m/s)
ω_x	=	angular velocity about x-axis (rad/s)
ω_y	=	angular velocity about y-axis (rad/s)
ω_z	=	angular velocity about z-axis (rad/s)
τ_x	=	torque about body fixed x-axis (N.m)
τ_y	=	torque about body fixed y-axis (N.m)
τ_z	=	torque about body fixed z-axis (N.m)
f_z	=	net force about body fixed z-axis (N.m)

II. Introduction

The aim of my controller is to control a quadrotor/drone so that it goes through several rings (obstacles) and gets to the end in the shortest time possible. I would consider my controller to be successful if it is able to get the drone to the end within 30s more than 70% times. I chose 30s since it is the max time the simulation will run for and 70% for the success rate since it seemed liked a reasonable criteria given the track length and randomisations. Additionally, my drone must have an average flying time less than 15s. To verify this I'll run the simulation in my PyBullet environment for 100 times and record successes (times less than 30s), failures (times more than 30s), the average finish time and check if it meets my requirements or not.

*Junior, Aerospace Engineering

III. System

A. Equations of Motion

The governing equations of motions for our system were provided by Prof. Bretl and are:

$$\begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{w_x} \\ \dot{w_y} \\ \dot{w_z} \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ \frac{w_x cos(\psi) - w_y sin(\psi)}{cos(\theta)} \\ w_x sin(\psi) + w_y cos(\psi) \\ -w_x cos(\psi) tan(\theta) + w_y sin(\psi) tan(\theta) + w_z \\ 2f_z sin(\theta) \\ -2f_z sin(\phi) cos(\theta) \\ 2f_z cos(\phi) cos(\theta) - \frac{981}{100} \\ \frac{10000\tau_x}{23} - \frac{17w_y w_z}{23} \\ \frac{10000\tau_y}{23} + \frac{17w_x w_z}{23} \\ 250\tau_z \end{bmatrix} \tag{1}$$

B. Defining state matrix and input matrix and sensor model

I defined my state matrix (x) and input matrix (u) and sensor model are:

$$x = \begin{bmatrix} p_x - p_{xe} \\ p_y - p_{ye} \\ p_z - p_{ze} \\ \phi - \phi_e \\ \theta - \theta_e \\ \psi - \psi_e \\ v_x - v_{xe} \\ v_y - v_{ye} \\ v_z - v_{ze} \\ w_x - e_{xe} \\ w_y - e_{ye} \\ w_z - w_{ze} \end{bmatrix} \qquad u = \begin{bmatrix} \tau_x - \tau_{xe} \\ \tau_z - \tau_{ze} \\ \tau_z - \tau_{ze} \\ f_z - f_{ze} \end{bmatrix} \qquad g = \begin{bmatrix} p_x \\ p_y \\ p_z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{2}$$

My chosen set of equilibrium values were:

$$\begin{bmatrix} p_{xe} \\ p_{ye} \\ p_{ze} \\ \phi_e \\ \theta_e \\ \psi_e \\ v_{xe} \\ v_{ye} \\ v_{ze} \\ e_{xe} \\ e_{ye} \\ w_{ze} \\ -\tau_{xe} \\ \tau_{ze} \\ \tau_{ze} \\ f_{ze} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{9.81}{2} \end{bmatrix} \tag{3}$$

These equilibrium values are such that function f becomes a matrix with all entries as 0, which is intended.

IV. Design

The next step in linearising my system is finding matrices A and B. By taking the jacobian of the state time derivative with respect to the state variables I found matrix A while taking the jacobian with respect to the input variables I found matrix B.

$$A = \left.\frac{\partial f}{\partial x}\right|_{x_{equi}, u_{equi}} \qquad B = \left.\frac{\partial f}{\partial u}\right|_{x_{equi}, u_{equi}} \tag{4}$$

My final matrices A and B were:

$$A = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & -0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.00 & 0.00 & 1.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 9.81 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & -9.81 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & -0.00 & -0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & -0.00 & -0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix} \quad B = \begin{bmatrix} 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & -0.00 \\ 0.00 & 0.00 & 0.00 & 2.00 \\ 434.78 & 0.00 & 0.00 & 0.00 \\ 0.00 & 434.78 & 0.00 & 0.00 \\ 0.00 & 0.00 & 250.00 & 0.00 \end{bmatrix} \tag{5}$$

Similar to A and B, we need to also find matrix C which is found by taking the jacobian of the sensor model with respect

to state variables:

$$C = \begin{bmatrix} 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix} \tag{6}$$

A. Controllability

Now that we have the necessary matrices to define our state space model, It is critical to determine if the system is even controllable or not. To do this, we need to find the controllability matrix W that is defined as

$$W = \begin{bmatrix} B & AB & A^2B & A^3B & \dots & A^{n-1}B \end{bmatrix} \tag{7}$$

A system is controllable if the rank of the controllability matrix W is equal to the number of states in the system. In other words, the rank of matrix W must equal the number of rows in matrix A (Agarwal, DP2/3, 2021). From above, we can see that A is a (12x12) matrix and therefore has 12 states. Computing the rank of the controllability matrix W using python's np.linalg.matrix rank function, I found the rank of W to equal 12. The system is controllable!

B. Observability

Since we are dealing with sensor data to give us approximates of the state of the spaceship in time, we need to check observability (Agarwal, DP2, 2021). The observability matrix is defined as:

$$O = \begin{bmatrix} C & CA & CA^2 & CA^3 & \dots & CA^{n-1} \end{bmatrix}^T \tag{8}$$

A system is observable if the rank of the observability matrix O is equal to the number of states in the system. In other words, the rank of matrix O must equal the number of rows in matrix A (Agarwal, DP2/3, 2021). We know that A is a (12x12) matrix and therefore has 12 states. Computing the rank of the observability matrix O using python's np.linalg.matrix rank function, I found the rank of O to equal 12. The system is observable!

C. Using LQR to find gain matrix and injection matrix

The reason for choosing LQR (Linear Quadratic Regulator) as the method to find suitable gain matrix K and injector matrix L is because not only is it easy to implement, but is also efficient since the the entire idea of LQR is to reduce the "cost" and yet make the system attain equilibrium the fastest (Agarwal, DP2/3, 2021). The LQR cost functions for my controller and observer are defined as:

$$controller\ cost = \int_{t_0}^{\infty} \left(x(t)^T Q_c x(t) + u(t)^T R_c u(t) \right) dt \tag{9}$$

$$observer\ cost = \int_{-\infty}^{t_0} \left(n(t)^T Q_c n(t) + d(t)^T R_o d(t) \right) dt \tag{10}$$

where Q and R are the weights associated with x(t) and u(t) respectively. Each entry in x(t) and u(t) are weighted by their corresponding entry in Q and R (Agarwal, DP2/3, 2021). In python I formed two LQR functions, one for the matrix K and one for L. My final matrices K and L were:

$$K = \begin{bmatrix} 0.00 & -1.04 & -0.00 & 1.67 & 0.00 & 0.00 & -0.00 & -0.63 & -0.00 & 0.23 & -0.00 & 0.00 \\ 0.66 & 0.00 & -0.00 & -0.00 & 1.73 & 0.00 & 0.67 & 0.00 & 0.00 & -0.00 & 0.23 & -0.00 \\ -0.00 & -0.00 & -0.00 & 0.00 & -0.00 & 0.21 & -0.00 & -0.00 & 0.00 & 0.00 & -0.00 & 0.21 \\ 0.00 & -0.00 & 0.93 & 0.00 & 0.00 & 0.00 & 0.00 & -0.00 & 1.07 & -0.00 & 0.00 & 0.00 \end{bmatrix} \tag{11}$$

$$L = \begin{bmatrix} 27.51 & 0.00 & 0.00 & 0.00 & 0.19 & 0.00 \\ 0.00 & 27.51 & 0.00 & -0.19 & 0.00 & 0.00 \\ 0.00 & 0.00 & 27.44 & 0.00 & 0.00 & 0.00 \\ 0.00 & -0.22 & 0.00 & 25.47 & 0.00 & 0.00 \\ 0.22 & 0.00 & 0.00 & 0.00 & 25.47 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 25.48 \\ 28.50 & 0.00 & 0.00 & 0.00 & 9.79 & 0.00 \\ 0.00 & 28.50 & 0.00 & -9.79 & 0.00 & 0.00 \\ 0.00 & 0.00 & 26.46 & 0.00 & 0.00 & 0.00 \\ 0.00 & -0.02 & 0.00 & 24.49 & 0.00 & 0.00 \\ 0.02 & 0.00 & 0.00 & 0.00 & 24.49 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 24.49 \end{bmatrix} \tag{12}$$

Using K and L and what we learned in class, I defined:

$$F_K = A - BK \tag{13}$$

$$F_L = A - LC \tag{14}$$

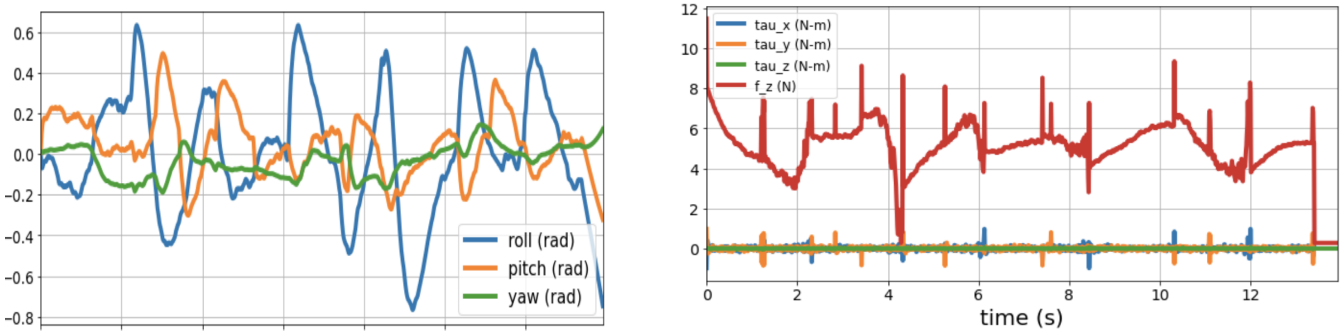
D. Stability

We ascertained that the system we have is controllable as well as observable but we still need to test stability. A stable system is one that, when given a definite, bounded input, returns a bounded output. Such a system is called "asymptotically stable". Mathematically, if all the real parts of the eigenvalues of a matrix are negative, the system is asymptotically stable (Agarwal, DP3, 2021). The eigenvalues for my F_K and F_L as defined in [**??**] and [14] turned out to be negative. I have a stable control in theory, but I must implement it and analyse the results to see if my controller met the requirements or not.

V. Results

A. Implementation

I implemented my controller with the GUI and plotted the results. Each time I tested my controller before my final simulation, the obstacle rings were randomly positioned to make sure that my controller works for any given course and doesn't just get lucky. My results for the simulation are shown below:



(a) Roll, Pitch, Yaw, wrt. time

(b) Torques and f_z wrt. time

Fig. 1 Simulation results

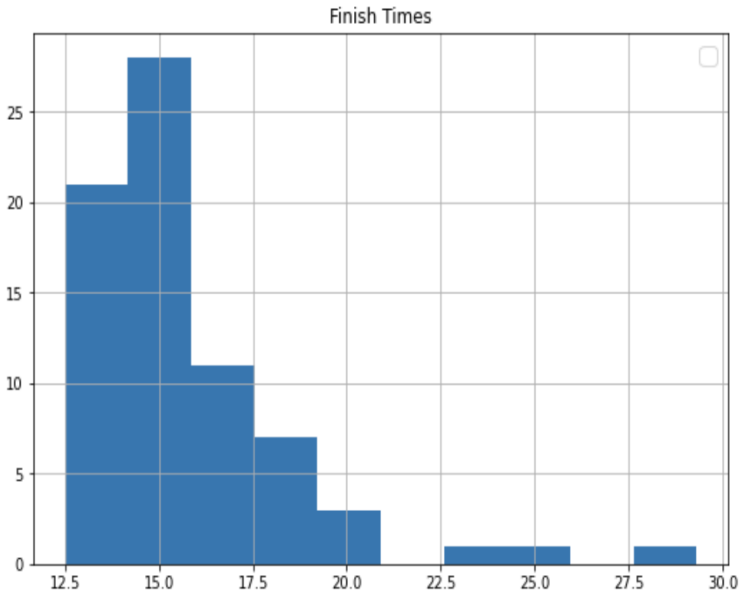


Fig. 2 Histogram of finish times for 100 sims

My controller works like it should. Now to verify that my controller meets my set requirements I run the simulation a hundred times without the GUI, and counting the success and failure rate. My findings were:

Trials	Successes	Failures	Success rate (%)	Average flight time (s)	Min finish time (s)
100	73	27	73	15	12.5

The results show that requirements 1 and 2 are satisfied and therefore I can say that my controller is successful.

VI. Conclusion

I began my process by linearising my system to get a state space model, and found matrices A, B, and C, after which I checked for controllability, observability. I then found gain matrix K and injection matrix L using the very efficient LQR method. I checked for stability and found that both K and L matrices are stable. Next I implemented my controller 101 times and checked if it met my preset requirements or not. I came to the conclusion that my controller works perfectly since my drone completes the course within 30s 73% of the time, as well keep the average flying time under 15s. The next step for me would be to improve the efficiency of my controller so that the average flying time is under 10s while the success rate being north of 90%.

Acknowledgments

I would like to thank professor Timothy Bretl for the very helpful lectures, example in-class codes as well as the homework that helped me in coding my controller. I also want to thank professor Timothy Bretl, for the AE353 course website off which I used the EOMs pre derived for me, as well as the images and description of the system. Additionally, his DroneDemo file as well as the DeriveEOM file were extremely helpful in my own implementation of my controller. I also want to credit Prahersh Kumar because I took inspiration from his code to use in my own. His code helped me find stable gain matrix K and injection matrix L in a really efficient way. I also want to thank all my peers for the continuous fruitful discussions we had on the topics as well as the project on Campuswire or Discord which helped me understand the content and do the project better.

References

[1] Bretl, Timothy.
https://tbretl.github.io/ae353-sp21/projectsdesign-project-4-drone

[2] Agarwal, Rohen (2021).
Design Project 2 - Differential Drive Robot

[3] Agarwal, Rohen (2021).
Design Project 3 - Controller-Observer implementation for Star-tracker Spaceship