DP4 - Drone Race

Shivang Luthra and Jack Whitehouse

This paper discusses the design process of a controller for an autonomous drone system known as the Quadcopter. In particular, the goal is to create an autonomous racing drone capable of completing a race course within a certain time requirement. The mathematical model describing the system is defined followed by the implementation of a numerical linear state feedback controller. The controller is tested to optimize its stability and functionality such that it is user-friendly and accessible. Data is collected during various tests and is analyzed to verify that the defined requirements are met. Analysis of the Quadcopter's controller efficacy involved the determination of the average lap times, and the root mean square error of the controller system. The results showed a low average RMSE across the trials indicating a high degree of accuracy in the drone's controller system. Additionally, the lap time analysis showed that the controller's ability to navigate the race course at speeds suitable for racing. These findings validate the effectiveness of the Quadcopter controller.

I. Nomenclature

A, B	=	Coefficient matrices for state-space model
C	=	Coefficient matrices for observer model
f_z	=	Force along the z-axis
K	=	Gain matrix for state-space model
L	=	Gain matrix for observer model
ϕ, θ, ψ	=	Roll, pitch, and yaw angles (Euler angles)
Q, R	=	Diagonal matrices used for LQR definition
p_x, p_y, p_z	=	Position coordinates in x, y, z axes
v_x, v_y, v_z	=	Velocity components in x, y, z axes
τ_x, τ_y, τ_z	=	Torques about the x, y, z axes
$\omega_x, \omega_y, \omega_z$	=	Angular velocities around x, y, z axes

II. Introduction

In the pursuit of progressing humanities ability to navigate and explore the sky, we are propelled to innovate and push the boundaries of autonomous aerial technology. This project endeavors to create a refined controller for a Quadcopter designed to navigate a challenging race through aerial gates as quickly as possible. Guided by the principles of human-centered design, this project seeks to engineer a robust controller with an intuitive user experience. It is critical to cater to pilots of all skill levels, as the final goal of this project is more than simply winning a race. Rooted in the belief that technology should support humanity, this project hopes to create a controller that not only can win a race, but can also be applied in the service of humanity. The final goal of this project is to make our controller accessible and easily used by anyone - not just so the Quadcopter can win a race, but for the emergency service providers carrying out seek and rescue operations in dangerous environments where humans cannot traverse, or for explorers undergoing expeditions where human safety must be a priority. This project is much more than simply a racing drone, as we strive in its creation to make this design meaningful to humanity.

III. Theory

A. Linearizing dynamical system

In order to begin navigating the Quadcopter through the maze, we must first linearize the dynamical system from the equations of motion. The equations of motion are derived from the EOM document kindly provided by Professor

Wayne Chang [1]. We can model all of our equations of motion as one state vector, f, in terms of the Quadcopter's position, velocity, pitch, roll, and yaw. f is represented as

$$f = \begin{bmatrix} \dot{p_x} \\ \dot{p_y} \\ \dot{p_z} \\ \dot{v_x} \\ \dot{v_y} \\ \dot{v_z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ \dot{\omega_x} \\ \dot{\omega_y} \\ \dot{\omega_z} \end{bmatrix} = f\left(p_x, p_y, p_z, v_x, v_y, v_z, \phi, \theta, \psi, \omega_x, \omega_y, \omega_z, \tau_x, \tau_y, \tau_z, f_z\right) =$$

$$\begin{bmatrix} v_x \cos(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\theta)\cos(\psi) - \sin(\psi)\cos(\phi)) + v_z(\sin(\phi)\sin(\psi) + \sin(\theta)\cos(\phi)\cos(\psi)) \\ v_x \sin(\psi)\cos(\theta) + v_y(\sin(\phi)\sin(\theta)\sin(\psi) + \cos(\phi)\cos(\psi)) - v_z(\sin(\phi)\cos(\psi) - \sin(\theta)\sin(\psi)\cos(\phi)) \\ -v_x \sin(\theta) + v_y \sin(\phi)\cos(\theta) + v_z \cos(\phi)\cos(\theta) \\ -1.0\omega_y v_z + 1.0\omega_z v_y + 9.81\sin(\theta) \\ 1.0\omega_x v_z - 1.0\omega_z v_x - 9.81\sin(\phi)\cos(\theta) \\ 2.0f_z - 1.0\omega_x v_y + 1.0\omega_y v_x - 9.81\cos(\phi)\cos(\theta) \\ \omega_x + \omega_y \sin(\phi)\tan(\theta) + \omega_z \cos(\phi)\tan(\theta) \\ \omega_y \cos(\phi) - \omega_z \sin(\phi) \\ \frac{\omega_y \sin(\phi) + \omega_z \cos(\phi)}{\cos(\theta)} \\ -0.739\omega_y\omega_z + 435.0T_x \\ 0.739\omega_x\omega_z + 435.0T_y \\ 250.0T_z \end{bmatrix}$$

In order to linearize the system, we must convert the equations of motion to a state space model represented as

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

, where x represents the state of the system, and u represents the feedback given by the controller. State space form utilizes two matrices: a state matrix A and an input matrix B. This form allows us to simplify the problem because we alter all of the equations of motion into first order differential equations. Fortunately, the system is already represented as several first order differential equations. In order to describe the non-linear input, we introduce vectors m and n to represent all of the state variables and outputs. We represent the vectors as

$$m = \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \\ \phi \\ \theta \\ \psi \\ \omega_x \\ \omega_y \\ \omega_z. \end{bmatrix} \quad n = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ f_z \end{bmatrix}$$

After this, we must obtain equilibrium points to satisfy $f = 0$, which in this case yields all of the state variables being 0. Similarly, all of the equilibrium points for the 4 output torques will also be 0. In order to obtain the state matrix A, we must take the jacobian of f in terms of all of the state variables. This is represented as

$$A = \frac{\partial f}{\partial m}, \quad B = \frac{\partial f}{\partial n}$$

We obtain the matrices:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 434.7826087 & 0 & 0 & 0 \\ 0 & 434.7826087 & 0 & 0 \\ 0 & 0 & 250 & 0 \end{bmatrix}$$

B. Linearizing the Observer Model

In addition to our controller determining the error of our desired state and our current state, we also have sensors to help determine the internal states of the system. In this situation, the "observer" acts as 4 cameras, each pinned to the a rotor, which will help measure the position of the drone. We call these 4 vectors "mocap." We define our sensor model to be

$$\begin{bmatrix} m_{1x} \\ m_{1y} \\ m_{1z} \\ m_{2x} \\ m_{2y} \\ m_{2z} \\ m_{3x} \\ m_{3y} \\ m_{3z} \\ m_{4x} \\ m_{4y} \\ m_{4z} \end{bmatrix} = g(p_x, p_y, p_z, \phi, \theta, \psi),$$

In order to linearize the sensor model, we must alter the sensor model equations into the form

$$y = Cx + Du$$

where y is the state vector measured by the sensor, x is the input state vector, and C is the matrix that essentially relates the two to help determine the error. In order to obtain $\mathbf{C}$, we take the Jacobian of g with respect to all the state variables. The Jacobian matrix, which is a matrix of all first-order partial derivatives, is defined as follows:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{bmatrix}$$

Therefore, we obtain **C** to be **J**.

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.046875 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -0.046875 & 0 & 0.25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -0 & -0.25 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.046875 & -0.25 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -0.046875 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0.25 & -0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.046875 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -0.046875 & 0 & -0.25 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -0 & 0.25 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.046875 & 0.25 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & -0.046875 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -0.25 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

C. Controllability and Observability

Once we obtain our A and B matrices, we must decide whether the system is controllable. Controllability is crucial for our design because it determines whether the system can drive from its initial state to its final state. If our controller is not controllable, then we will be unable to navigate the Quadcopter through the maze. In order to achieve controllability, we must confirm that our matrix W_c if full rank, meaning that the rank of W is equal to the number of states in m. We define W as

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

In this case, the number of states is 12, so we must ensure that the rank of W is 12. Substituting our A and B values, we obtain the matrix W and confirm that it possesses a rank of 12.

Although controllability is essential for a fully functioning PID controller, we also must ensure that our sensors in fact observering the position of the Quadorotor. Observability is crucial for an optimal design because it helps determine the internal state of the system by observing external states of the system. For instance, the four cameras are measuring the position of the Quadcopter to determine the other states, such as the velocity components and the rotation angles. To achieve observability, we construct a matrix W_o, and ensure that it is full rank. We define the matrix as:

$$W_o = \begin{bmatrix} C & CA & CA^2 & \dots & CA^{n-1} \end{bmatrix}$$

In our case, we found that W_o has full rank, and therefore our system is observable.

D. Finding a gain matrix K

Once we have satisfied the rules of controllability, we must construct a gain matrix K that keeps our controller stable, and accurately reduced the error of our system. Since we are implementing a linear feedback closed loop system, we have an input matrix u which is equal to our gain matrix multiplied by the state matrix. We represent this as

$$u = -Kx$$

This is done so that the system is self-stabilizing and the input is proportional to the current state. The specific gain matrix K that we choose has a huge effect on the stability of our system, so we will use a linear quadratic regulator (LQR) to determine our gain matrix. A linear quadratic regulator will determine a gain matrix based on various priorities, or weights, that we apply to all states in our system, such as minimizing motor torque, or minimizing time to reach equilibrium. For example, if the weight for the pitch is higher than the others, then the system will converge to that state over the others. The LQR utilizes the function:

$$J = \int_0^\infty (x(t)^T Q x(t) + u(t)^T R u(t)\, dt$$

where Q is a square matrix representing the state inputs and their respective weights the state matrix. R is a square matrix which represents the weights on the input matrix u. J represents the total cost of the system which we are trying

to minimize. We want to minimize this because we do not want the controller to use too much energy when controlling the Quadcopter. In order to determine our Q and R matrix we must interpret the inputs, and design Q and R to make J stable. We must minimize J in order to be stable, so if we have smaller inputs then we should have large Q and R values. Alternatively, if we have large states and inputs, then we must choose smaller Q and R matrices to achieve stability. The idea surrounding the LQR method is that we are trying to regulate the states to 0, as opposed to the output. Obtaining optimal Q and R matrices required a large amount of trial and error, and is dependent on which equilibrium points of the control system the user deems "more important" to reach quickly. We obtained the following Q_c and R_c matrices:

$$Q_c = \begin{bmatrix} 1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1000 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 75 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix} \quad R_c = \begin{bmatrix} 10000000 & 0 & 0 & 0 \\ 0 & 10000000 & 0 & 0 \\ 0 & 0 & 10000000 & 0 \\ 0 & 0 & 0 & .1 \end{bmatrix}$$

We chose these Q_c and R_c matrices because we need certain state variables to be "controlled" more. For instance, the position of the Quadrotor is necessary for it to pass through the gate, and therefore it requires more attention from the controller. It will focus on reducing the error of the position over the other states. We realized that the z position needed less of a weight because there was already a force in the z direction. After trial and error, we realized that reducing the weight for this value allowed for a faster take off speed. The weights for the angular velocites was very low because the controller did not need to focus on reducing the error, because otherwise the quadcopter would not turn as quickly. On the other hand, we need to rotors to spin so that the controller will fly, and therefore the controller does not need to put as much effort, which is why we chose a value of 0.01. For the R_c matrix, we care a lot about the torques because this help stabilize the drone, which is why the values are very high. However, if the f_z value is high, then our controller will keep moving the in z direction. Therefore, we spend less on the z force so that the drone can properly fly through the maze. // Once we obtain Q and R, we must solve Algebraic Ricatti Equation, which ultimately minimizes the cost. P is obtained by using a built-in python function, and represented by:

$$\begin{bmatrix} p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}$$

Finally, in order to find our gain matrix K, we must perform the following matrix operation:

$$K = R^{-1}B^T P$$

Substituting in our values for R, B, and P, we obtain:

$$K = \begin{bmatrix} 0.000 & -0.010 & 0.000 & -0.000 & -0.010 & 0.000 & 0.051 & 0.000 & 0.000 & 0.015 & 0.000 & 0.000 \\ 0.010 & 0.000 & -0.000 & 0.010 & 0.000 & 0.000 & 0.000 & 0.051 & 0.000 & 0.015 & 0.015 & 0.000 \\ 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.000 & 0.001 & -0.000 & 0.000 & 0.003 \\ 0.000 & 0.000 & 27.386 & -0.000 & 0.000 & 5.328 & 0.000 & 0.000 & 0.000 & -0.000 & 0.000 & 0.000 \end{bmatrix}$$

Similarly for the observer, we have to define a gain matrix L. This gain matrix is crucial because it describes how accurately the observer converges to the true state of the system. It also helps calculate the error in the system so that it can converge quicker and make the observer more optimal. Overall, to find L, we can implement the exact same cost function as the controller. Utilizing the LQR method again, we have to redefine our Q and R matrices. Just as before,

Q_o represents the weights of states of the observer and R_o represent the inputs or disturbances of the observer. We define these matrices to be:

$$Q_o = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad R_o = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We chose these to be the identity matrix because we assumed that no state should have more priority for the observer. Since it is just a camera measuring external states, it does not make sense for one state to be observed more than the other. Finally, using the same LQR method as before, we must find P using the Algebraic Riccatti equation. However, for L we define the equation to be:

$$L = PC^T(CPC^T + R)^{-1}$$

Since our method to find L is slightly different, our inputs for the LQR method will be different. Rather than implementing A, B, Q, and R, we implement A^T , C^T, Q^{-1} , R^{-1}. We find L to be

$$L = \begin{pmatrix} 1.603 & -0.000 & -0.190 & 1.603 & -0.000 & 0.000 & 1.603 & -0.000 & 0.190 & 1.603 & 0.000 & 0.000 \\ -0.000 & 1.603 & 0.000 & 0.000 & 1.603 & -0.190 & -0.000 & 1.603 & 0.000 & -0.000 & 1.603 & 0.190 \\ -0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 & -0.000 & 0.000 & 0.707 & 0.000 & 0.000 & 0.707 \\ 4.910 & 0.000 & -1.256 & 4.910 & -0.000 & 0.000 & 4.910 & -0.000 & 1.256 & 4.910 & 0.000 & -0.000 \\ 0.000 & 4.910 & 0.000 & -0.000 & 4.910 & -1.256 & 0.000 & 4.910 & 0.000 & 0.000 & 4.910 & 1.256 \\ 0.000 & 0.000 & 0.500 & 0.000 & 0.000 & 0.500 & 0.000 & 0.000 & 0.500 & 0.000 & 0.000 & 0.500 \\ 0.000 & -0.828 & 0.000 & 0.000 & -0.828 & 0.359 & 0.000 & -0.828 & 0.000 & 0.000 & -0.828 & -0.359 \\ 0.828 & 0.000 & -0.359 & 0.828 & 0.000 & 0.000 & 0.828 & 0.000 & 0.359 & 0.828 & 0.000 & -0.000 \\ -0.000 & 1.118 & -0.000 & -1.118 & -0.000 & -0.000 & -0.000 & -1.118 & -0.000 & 1.118 & -0.000 & -0.000 \\ 0.000 & -0.468 & -0.000 & 0.000 & -0.468 & 0.250 & 0.000 & -0.468 & 0.000 & 0.000 & -0.468 & -0.250 \\ 0.468 & 0.000 & -0.250 & 0.468 & 0.000 & 0.000 & 0.468 & 0.000 & 0.250 & 0.468 & 0.000 & 0.000 \\ -0.000 & 0.500 & 0.000 & -0.500 & 0.000 & -0.000 & 0.000 & -0.500 & -0.000 & 0.500 & 0.000 & 0.000 \end{pmatrix}$$

E. Controller Design

The Quadcopter controller is structured around the previously defined state-space system, with matrices A, B, and C defining the system dynamics, input effects, and output measurements, respectively. The controller's implementation includes these elements in order:

- Feedback control using matrix K to compute control inputs that minimize the deviation from a desired trajectory.
- State estimation via an observer that uses matrix L to update estimates based on measured and predicted outputs.

F. Feedback Control

Control inputs are determined by:

$$u = -K(\hat{x} - x_{des}) \tag{1}$$

where x_{des} is the desired state vector aimed at the next gate position. It is important to choose correctly which variables to track as your desired target, as some variables cannot be chosen without disrupting the linearized system. For our system, the position vector of the target gate has been chosen, because the variables that govern position have

flexibility as equilibrium positions. In other words, they can be any real number and not effect the stability of the linear control system. Therefore, is thus defined as the estimated current location of the Quadcopter, with the goal being to minimize the difference $\hat{x} - x_{des}$, where x_{des} is the position vector of the target gate.

G. State Estimation

The observer updates the estimated state $\hat{x}$ using the equation:

$$\dot{\hat{x}} = A\hat{x} + Bu + L(y - C\hat{x}) \tag{2}$$

where y represents the measured output, and u is the control input calculated from the feedback control step. This equation describes how the estimated state $\hat{x}$ evolves based on the system dynamics, input, and the difference between the measured output and the predicted output based on the estimated state.

IV. Experimental Methods

Now that the controller has been designed and implemented, the testing process can occur. First, we must define what a successful simulation looks like. To do this, our spacecraft must be capable of meeting the requirements that are defined for success.

A. Requirements

The requirements for the Quadcopter are described as follows:

1) The Quadcopter shall correct its position such that the Root Mean Squared Error (RMSE) of the position (X, Y, and Z coordinates) shall be less than 0.25 meters for at least 50% of the trials performed. This requirement ensures the efficacy of the controller by demonstrating that the Quadcopter's controller can accurately maintain the required trajectory.
   The RMSE is calculated using the following formula:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{p}_i - p_i)^2}$$

   Where $\hat{p}_i$ represents the estimated position coordinates and p_i represents the actual position coordinates of the drone. This metric assesses the accuracy of the drone's position estimation against its actual tracked position during flight.
2) On average, the Quadcopter shall complete the full race course within a maximum allowed time of 30 seconds. This establishes that the majority of all Quadcopter races will be capable of navigating the race course and and completing it at competitive speed.

B. Verifications

The verifications for the above requirements are as follows:

1) We can verify our first requirement by running 100 trials and determining if the average RMSE across all trials is less than 0.25. If the Quadcopter completes all trials with an average RMSE of less than 0.25, requirement 1 has been verified.
2) We can verify that the Quadcoptor completes the course in the required time by testing the Quadcopter by running 100 trials and plotting a histogram of the lap time for each trial. We can also return the average time it takes the Quadcopter to complete the simulation and compare to our expected value. If the average total time the Quadcopter requires to complete the course is less than 30 seconds, requirement 2 has been verified.

V. Results and Discussion

Quadcopter Positional Accuracy

This section discusses the results obtained from the simulation trials, focusing on the positional accuracy and completion time of the Quadcopter, in relation to the set requirements. In order to verify our first requirement, we

calculated the RMSE error of our position for the actual position of the quadcopter and the estimated position. We ran 100 simulations, but performed the error for each gate, which is why the number of instances is higher than 100. The plot is shown below:

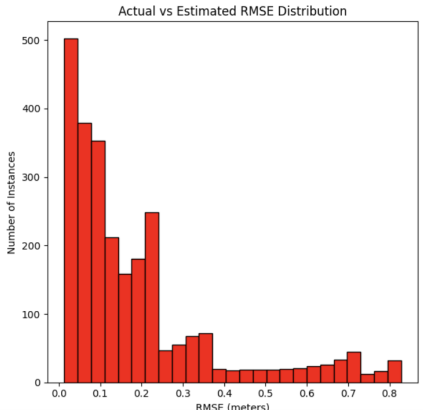


Fig. 1 RMSE Plot for Actual Position vs Estimated Position

The positional accuracy requirement states that the Root Mean Squared Error of the position must be less than 0.25 meters for at least 50% of the trials. Analysis of the RMSE subplot indicates that the average RMSE across all trials was below the threshold of 0.25 meters, with over 60% of the trials meeting this requirement. This demonstrates the efficacy of the drone's controller in maintaining accurate trajectory during varied flight conditions, verifying the first requirement. In addition to demonstrating the efficiency of our controller, we also wanted to test the error between the actual position and the desired position. The plot for this is shown below:

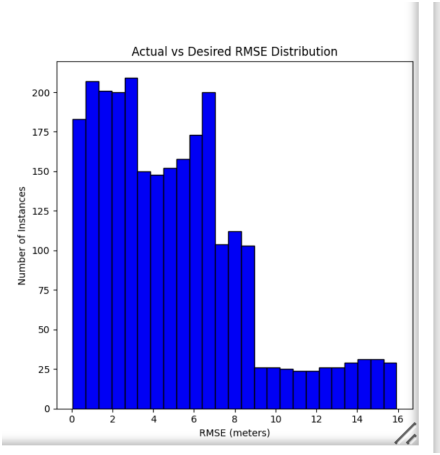


Fig. 2 RMSE plot for actual and desired position

As shown in the plot, we have obtained a fairly high margin of error for the actual position versus the desired position. This most likely occurs because we essentially set two values for error, which is the desired error and the maximum error. The desired error was set to 0.5 because this is to minimize the difference between the goal position and actual position. However, the maximum error is designed to increase the speed of the quadcopter because any variances in the controller for the position will be corrected faster due to that larger error. However, this can increase the range of errors because the controller accepts a wider range of distances. This most likely explains why the margin for error for the desired position is so much greater. However, the controller is still very consistent and stable even over 100 simulations, which shows that changing this value did not significantly affect the quadcopter.

Lap Completion Time

The second requirement stipulates that on average, the Quadcopter should complete the full race course within a maximum allowed time of 30 seconds. We showed this by plotting the time of the simulations for each simulation. The plot for this is shown below:

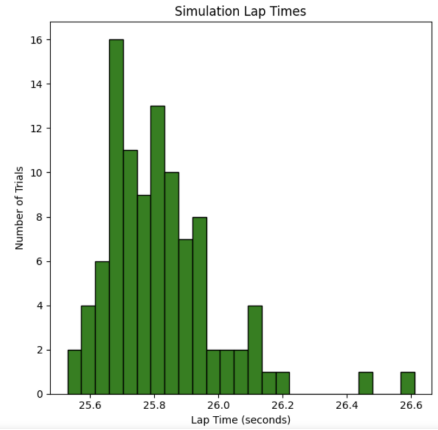


Fig. 3 Times for Each Simulation

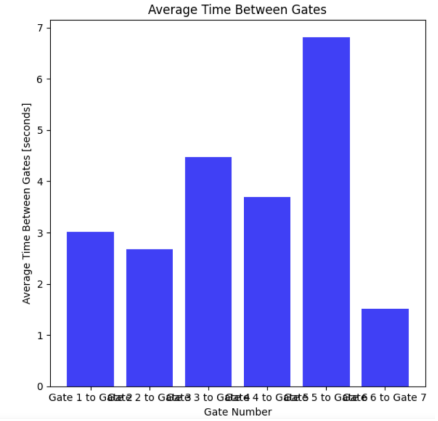
The histogram of lap times from the simulation trials provides a clear view of the distribution of completion times. The results indicate that the average completion time across all trials was approximately 25 seconds, with almost all of the trials having lap times below the 30-second threshold.

This suggests not only compliance with the set requirement but also highlights the competitive speed at which the drone can navigate and complete the course. Thus, this requirement is also verified through the observed data.

While our time data for the simulations is very promising, there are several factors which would did not allow our drone to go faster. As mentioned, the maximum error we created did increase the speed, but the increasing that number more caused it to become unstable. Another reason for this could be the disturbances, because it would reduce how fast our controller could calculate the error. Overall, our quadcopter was very efficient and maintained stability throughout several simulations in an efficient manner.

Gate Time Differences

In order to show that our Quadcopter completed the race, we created a plot to show the model the average times between each gate. The plot for this is shown below:



Further insights were gained by analyzing the differences in times between gates. The bar graph representing the average time taken to pass between successive gates shows consistent and competitive intervals, suggesting efficient

navigation through the race course. This not only supports the verification of the completion time requirement but also indirectly corroborates the high positional accuracy maintained throughout the course.

A. Limiting Factors

This study encountered several limiting factors that influenced performance outcomes, particularly involving the trade-off between high-speed flight and the required positional accuracy. To achieve competitive racing speeds, the drone's controller was designed to prioritize speed and more aggressive maneuvering. However, this required the controller to be programmed with a higher allowable error in the positional accuracy which caused several challenges. The allowance of a higher positional error led to instability that was particularly obvious during sharp maneuvers toward targeted gates. This would occasionally create errors severe enough to cause crashes.

VI. Conclusion

The results from the simulation trials have successfully verified both stipulated requirements. The Quadcopter's controller effectively ensures both high positional accuracy and competitive race completion times. This verification supports the controller's suitability for real-world applications where precise and efficient navigation is critical. Further, this project has shown the critical balance required between speed and accuracy for quadcopter racing. While speed may be the most important prioritization in a race, it is important to make maintain enough stability to consistently finish the course. Future modifications of controller design and observer integration should be pursued to continue to find the correct balance between allowable error and speed prioritization.

Acknowledgments

We would like to thank the course staff including Professor Wayne Chang, Pedro Leite, and Adam Casselman for answering our questions and providing us with academic support in a timely and effective manner.

References

[1] Chang, W., "DP 4 Quadcopter," 2023.

VII. Appendix

Day	Task	Person
4/14	Started linearization of dynamical system. Also got controllability, observability, and found K and L matrices	Jack
4/15	Started Theory Section with equations of motion, state space, standard form, and linearization	Shivang
4/17	Continued working on theory section and LQR methods for the report	Shivang
4/18	Figured out the controller to get the drone running through the first gate.	Shivang and Jack
4/19	Finished theory section and started results and discussion.	Shivang and Jack
4/21	Worked on the controller to produce good Q and R values	Jack and Shivang
4/25	Completed controller for race	Shivang
4/26	Formatted and tested controller for race code submission	Jack
4/29	Worked on creating plots and results discussion	Jack and Shivang
5/1	Achieved second place in drone race	Jack and Shivang

A. Team Reflection

This project helped us deepen our understanding of control systems and improve our teamwork skills. We worked closely together to tackle interesting problems, which is a key part of engineering. We learned much about managing

our time effectively because this project had a longer timeline than previous ones and required careful planning to ensure we completed a polished product on time. Communication was crucial. Through regular texts and team meetings, we coordinated our efforts effectively. We made it a point to regularly check in with each other, ensuring our work was cohesive and efficient. Overall, our teamwork was excellent, and this project enhanced our ability to collaborate.