Lab 4: Manual Point-to-Point Control

Aaryaman Batra*, Vignesh Sella*
University of Illinois Urbana-Champaign, Urbana, IL, 61801

In this lab report, the students develop a controller capable of controlling a quadrotor on a point-to-point trajectory with minimal deviation. This controller is rigorously tested in MATLAB simulation, and then implemented on flight hardware for further verification. Since the students are online, their controller was not utilized on the flight hardware, however the course-staff provided controller is compared to local simulation and analyzed. The student developed controller performed nominally on a standard flight test, and gradually declined in performance as the trajectory grew in distance or shortened in time. The experimental data matched up well with simulation, and the gain matrices utilized in each trial succeeded in their purported goals.

I. Nomenclature

x = state
u = input
σ = spin rate (rad/s)
ψ = yaw (rad)
θ = theta (rad)
ϕ = roll (rad)
ω = angular velocity (rad/s)

II. Introduction

This lab is a culmination of the experiences gained in the previous labs of characterizing the properties of a quadrotor's rotors and how to control the quadrotor in 3-D. A controller was designed and adapted in order to maximize tracking accuracy of a preplanned trajectory, i.e. a point-to-point path. This controller was iterated upon by analyzing its performance in MATLAB simulation, and finally was put to test on flight hardware, namely a Hummingbird quadrotor. Unfortunately, since this lab was conducted by students taking the course online, the controller utilized in lab was not the one developed but rather a TA provided one. The real-world performance of the TA controller was nonetheless also analyzed and commented upon. Point-to-point control is a standard objective for a quadrotor, and if the application involved precise tracking as is the case for video-camera drones, then the quadrotor must reach each point in a timely manner with minimal tracking error. Therefore, care was taken to minimize the overshoot of the quadrotor, and execute each point-to-point motion with zero yaw angle. The design process of the controller used to implement point-to-point tracking is discussed, then the experimental results of the TA controller is discussed, and finally a conclusion of the results of the work produced.

III. Control Design and Implementation

The model used in this report is the same as the one designed in the report for Lab 3 "Hovering Control" [1]. It is assumed that the model for this experiment remains the same, since the addition of point-to-point tracking does not change the base equations of motions utilized to form the state-space model.

A. Linearization & State-Space Form

In order to be able to optimally control the quadrotor, a representation in state-space form is required. This model is derived from the ordinary differential equations formulated in the report for Lab 3 [2]. Firstly, we define the state and input of the system.

$$\boldsymbol{x}^T = \begin{bmatrix} \mathbf{r} & \psi & \theta & \phi & \mathbf{v} & \mathbf{w} \end{bmatrix}^T \qquad \boldsymbol{u}^T = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 \end{bmatrix}^T \tag{1}$$

*Student, Department of Aerospace Engineering, AIAA Student Member

In Eq. (1), the state vector is a 12 x 1 matrix consisting of the position, Euler angles, velocity, and angular velocity of the quadrotor. And the input vector is 4 x 1 matrix, consisting of the three input commands, and u4, which is defined as -f_{rotor}. Then, to linearize the equations the "state" is defined as the error between the current quantity and some equilibrium/desired state denoted by the subscript 'e' as shown in Eq. (2).

$$\boldsymbol{x_c} = \boldsymbol{x} - \boldsymbol{x_e} \qquad \boldsymbol{u_c} = \boldsymbol{u} - \boldsymbol{u_e} \tag{2}$$

To linearize the equations, a quantity, $\boldsymbol{g}$, is first defined as the time-derivative of my state vector - where this derivative result comes from EOMs described in the previous Lab report. [2]. In Eq. (3), $\boldsymbol{g}$ is also a 12 x 1 matrix since $\mathbf{v}$ is the time-derivative of the position.

$$\boldsymbol{g}^T = \begin{bmatrix} \mathbf{v} & N\boldsymbol{\omega} & \dot{\boldsymbol{v}} & \dot{\boldsymbol{\omega}} \end{bmatrix}^T \tag{3}$$

Using MATLAB's `matlabFunction`, the symbolic expression from Eq. (3) was converted to a numeric actionable function, and it's zeros were solved for using `fsolve`. This method requires an estimation for the equilibrium values, which was set as all zeros except for the input u_4 - which was equal to mg - so that the equilibrium state is a steady hover. When the hover height was not 0, i.e. 0.5 - 1.5 m, then the equilibrium z state was set to this value. Then, to linearize the equations and express the system in state-space form, the Jacobian of (3) was taken with respect to the state and input to produce the A and B matrices respectively in Eqs. (4) & (5).

$$A = \frac{\partial g}{\partial x}\Big|_{x_e, u_e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -9.81 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 9.81 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4}$$

$$B^T = \frac{\partial g}{\partial u}\Big|_{x_e, u_e} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 250 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 250 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 142.9 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1.36 & 0 & 0 & 0 \end{bmatrix}^T \tag{5}$$

Thus the equation has been linearized about its equilibrium point and can be expressed in state-space form, as such:

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x_c} + \boldsymbol{B}u_c \tag{6}$$

B. State Feedback

To implement state feedback and optimal control, a Linear Quadratic Regulator method is employed. This method minimizes a cost function using two parameter matrices, Q and R. The controller then attempts to minimize this cost function comprising of both the error in the current state and the cost of the input. The input into the system is dependent on both the current state of the system, and a parameter matrix K, referred to as a gain matrix. The values of this gain matrix are obtained using the MATLAB function `lqr(A,B,Q,R)`. Wherein, A and B are from Eqs. (4) & (5) and Q and R must be decided upon. The values of the Q and R must be fine tuned to correctly respond various states. The values of the Q matrix, or rather their relative order of magnitude in comparison with the other values, determine how costly it is to effect a particular state. To choose my Q matrix, the states that were most important to hovering stable with 0 yaw were emphasized with larger values. Therefore, in the Q matrix the height (Z position) has the largest magnitude,

followed by the yaw angle, then the other positions, and lastly the angular velocities with the least magnitude. For the R matrix, I a multiplier of 10 was set so that the ratio of the Q matrix to the R matrix was not too large, since there was no need to overly penalize the cost of operating the drone. The input, as described earlier, is scaled by the gain matrix K. The gain matrices found and tweaked here were designed around hovering in one position. Further changes to the gain matrix were made to make the controller perform optimally for the provided trajectory. These changes are detailed in section IV

$$Q = \text{diag}([1, 1, 10, 5, .5, .5, .5, .5, .5, .5, .5, .5]), \quad R = \text{diag}([10, 10, 10, 10]) \tag{7}$$

$$K = \begin{bmatrix} 0 & 0.71 & 0 & 0 & 0 & 1.57 & 0 & 0.5 & 0 & 0.25 & 0 & 0 \\ -0.71 & 0 & 0 & 0 & 1.57 & 0 & 0.5 & 0 & 0 & 0 & 0.25 & 0 \\ 0 & 0 & 0 & 0.71 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.24 \\ 0 & 0 & 3.16 & 0 & 0 & 0 & 0 & 0 & -2.18 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

$$u = -K\boldsymbol{x_c} \tag{9}$$

C. Motor Command Limit

The On-board hummingbird computer does not run as per the spin rates but with the motor command. The relation between the spin rates and motor command can be written as follows where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are arbitrary constants calculated previously in Lab 2 [2]

$$\boldsymbol{\sigma} = \boldsymbol{\alpha}\boldsymbol{\mu} + \boldsymbol{\beta} \tag{10}$$

Since there is a limit of 200 on our motor command, MATLAB was used to calculate $\boldsymbol{\mu}$ using the above equation and limiting it to 200. The value of the motor command was also not allowed to drop below zero. This was done so the simulation follows the same restrictions that our experiment undergoes.

D. Point-to-Point Tracking

In order to create a simulation that can track our drone at different equilibrium points, the simulation used in Lab 3 was altered. The equilibrium point which was previously constant was set to change as a function of time. If statements in MATLAB were used to create a trajectory as provided in the pre-lab. This trajectory can be observed in Table 1 that provides the x,y and z coordinates with respect to time.

IV. Analysis and Improvements

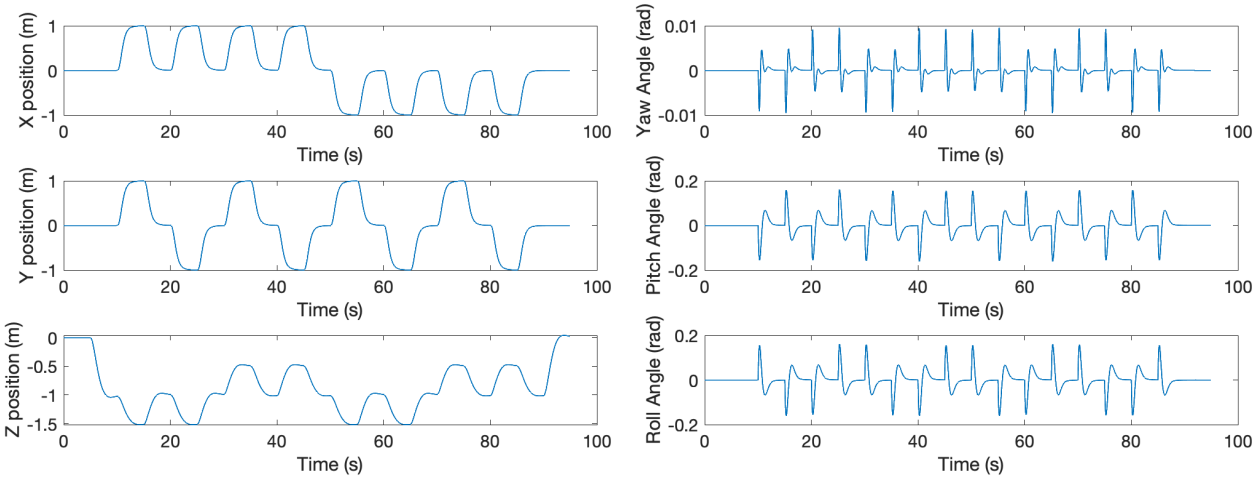
A. Gain Tuning

The initial controller utilized in simulation was replicated from Lab 3, which was designed for hovering. The simulation utilized to test point-to-point tracking followed the trajectory described in Table 1, which is from the Pre-lab Manual [3].

Table 1 Standard Flight Test [3]

Waypoints	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
t, s	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90
x_{des}, m	0	0	1	0	1	0	1	0	1	0	-1	0	-1	0	-1	0	-1	0	0
y_{des}, m	0	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	1	0	-1	0	0
z_{des}, m	0	-1	-1.5	-1	-1.5	-1	-0.5	-1	-0.5	-1	-1.5	-1	-1.5	-1	-0.5	-1	-0.5	-1	0

As expected, performance of this controller was not optimal, and significant overshoot and instability was witnessed in the z-position of the quadrotor. The x, y, and z positions and Euler angles are plotted in the Fig. 1a & 1b below. From visual inspection it is also clear that the controller had a long response time to the next point, and it spent a large amount of time in transition between points. The mean absolute error between the simulation and the desired trajectory for the x and y positions with this rudimentary (Lab 3) controller were negligible for the yaw angle and x and y-positions, but was 0.0463m for the z-position which was small but not up to par. Furthermore, the quadrotor never reached steady state and was constantly in transition between each point, and thus the steady state error was so poor as to be unquantifiable.



(a) X,Y, Z Positions vs. Time

(b) Yaw, Pitch, Roll Angles vs. Time

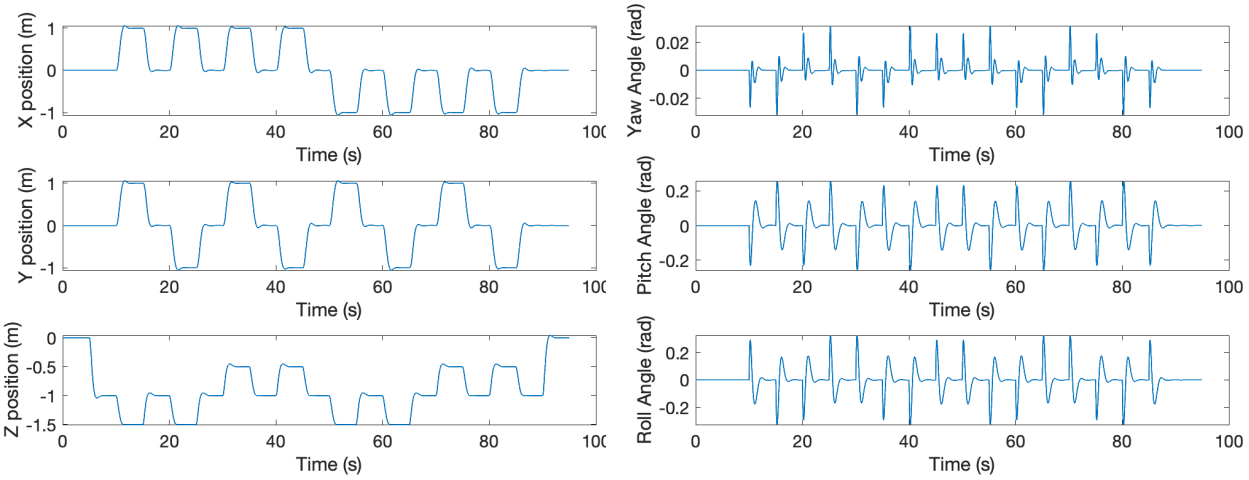
Fig. 1 First Controller in MATLAB Simulation

In order to rectify this, the third element on the diagonal of the Q matrix, which corresponds to the z-position of the quadrotor was increased by an order of magnitude from 10 to 100. This should make the cost of any error in the z-position larger, encouraging the drone to correct this with more input. The adverse effect of increasing this too far is error in the other states and potential overshoot since the drone may try to correct to the next position with too large a velocity. To prevent this, the z-velocity or the ninth element on the diagonal of Q was increased from 0.5 to 1 and the x and y-velocities corresponding values were decreased from 0.5 to 0.25. This controller was superior to the initial control, and was further refined. The final gain matrix Q is highlighted in Eq. (11).

$$Q_{\text{final}} = \text{diag}([5, 5, 500, 5, .5, .5, .25, .25, 1, .5, 1, .5]), \tag{11}$$

This final gain matrix Q featured a larger z-position value, from 100 to 500. The yaw angle value was kept at 5, as this was a larger magnitude than the other Euler angles and performed nominally. Lastly it was deemed unnecessary to make any changes to the R matrix. The performance of this controller in simulation is shown in Fig. 2a & 2b.

Immediately, it is clear that the position tracking was responsive and featured minimal overshoot. The sharp features of the position vs time plot are desired so that the quadrotor does not linger between points during its point-to-point tracking. Furthermore, the yaw angle deviance was slightly increased from a maximum of approximately 0.01 rad to 0.02, however, this was deemed negligible. The mean absolute error for the final gain matrices was essentially 0 for the x and y positions and yaw angle. For the z-position the mean absolute error was reduced from an initial value, previously mentioned, of 0.0463 to 0.0066m. The steady state error for this final controller was quantifiable and was calculated by taking the average of a three steady state differences between the desired and simulated positions, and it was on the order of 10^{-5}. Thus, this new gain matrix was both successfully responsively bringing the drone to steady state and maintaining its position in steady state with minimal to no error, in simulation.



(a) X,Y, Z Positions vs. Time

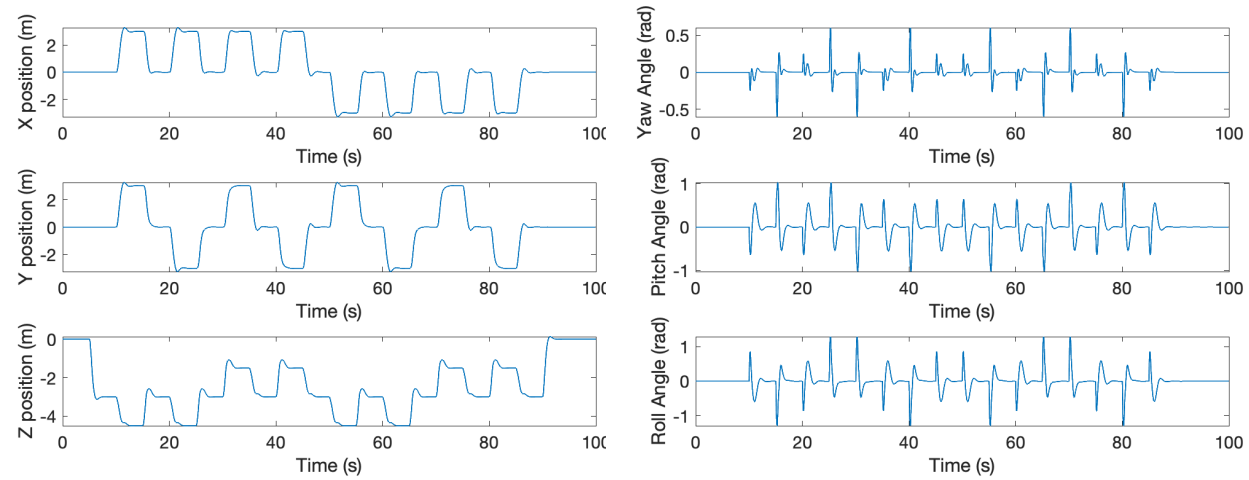
(b) Yaw, Pitch, Roll Angles vs. Time

Fig. 2 Final Controller in MATLAB Simulation

B. Controller Performance Analysis

In order to understand the performance of the controller under more demanding trajectories, shorter and farther positions were experimented with in simulation. The standard flight test trajectory, described in Table 1, utilized brackets of 5 seconds between each point on the point-to-point path. In order to stress test the controller time brackets of 5, 5/1.25, 5/1.5, 5/1.75, and 5/2 were tested with. Shown in Fig. 4 is the standard flight test at 1.25x quicker point-to-point times (4s) and double time (2.5s). From Fig. 4a it is clear that the as compared to when the point-to-point times were 5s, seen again in Fig. 1a, the x and z-position curves are beginning to spend more time in transition rather than steady state. This effect is significantly obvious at double time, in Fig. 4c, wherein the x-position appears completely curved and spends very little time in steady state. Likely, this is due to the fact that there is not enough time for the quadrotor to arrive at each point, and since it has motor command limiting, it cannot provide any further thrust to move quicker.

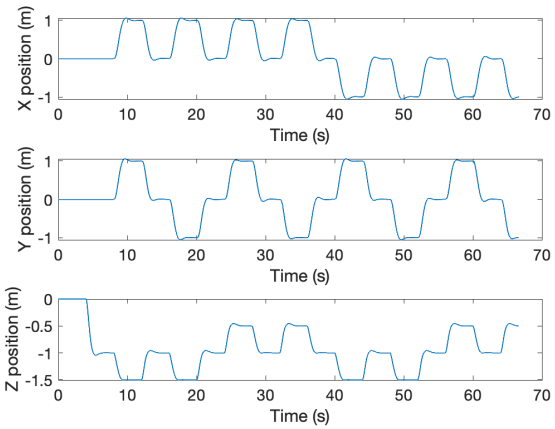
Lastly, an extended distance trajectory was also tested for 2x and 3x the distances in the standard flight test. It can be seen clearly in Fig. 3a & 3b that the longer distances between each point directly result in longer transition times - which is reasonable since the quadrotor can only move with so much thrust. Furthermore, yaw error is an order of magnitude larger, .05 to .5, than when the quadrotor moves the standard flight test distance.



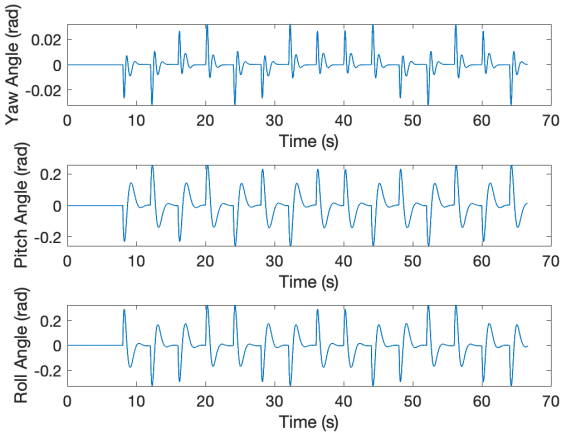
(a) X,Y, Z Positions vs. Time

(b) Yaw, Pitch, Roll Angles vs. Time

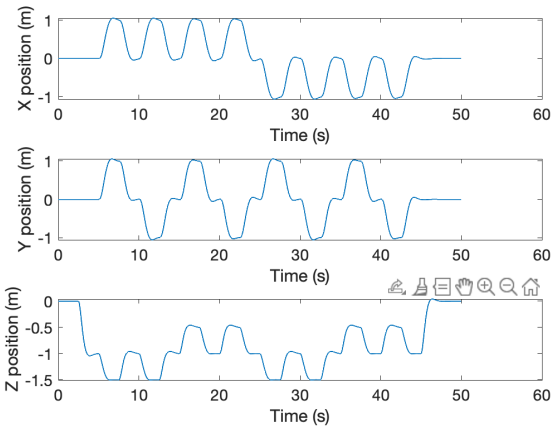
Fig. 3 Final Controller in MATLAB Simulation with 3x Distance Trajectory



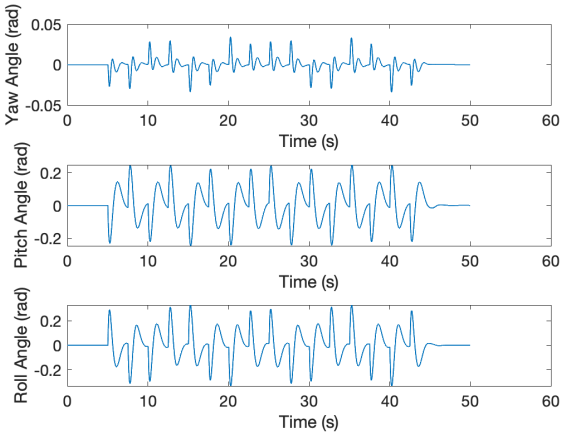
(a) 1.25x Time Trajectory Position vs Time



(b) 1.25x Time Trajectory Euler Angles vs Time



(c) Double Time Trajectory Position vs Time



(d) Double Time Trajectory Euler Angles vs Time

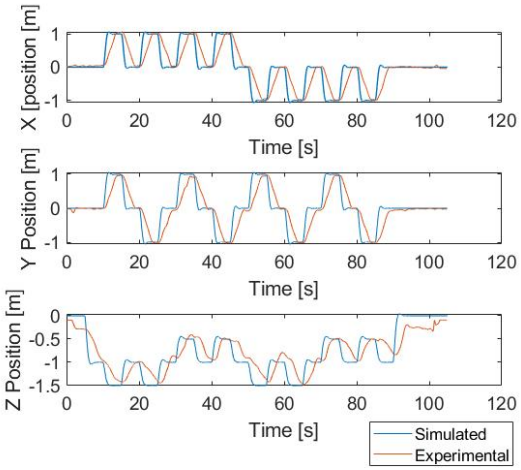
Fig. 4 Final Controller in MATLAB Simulation with Quicker Trajectories

V. Experimental Results

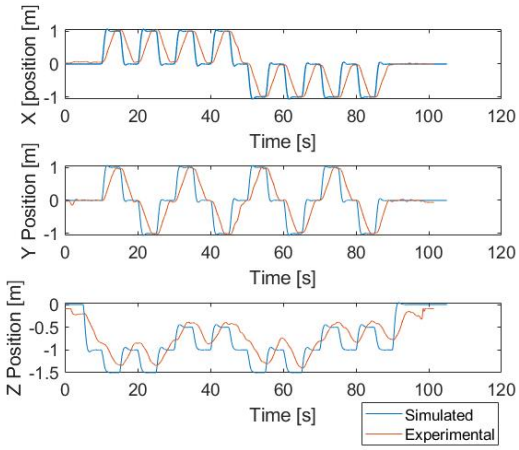
The results of the experiments were plotted against the simulation data to see the real-word accuracy of the predictions made by the model. A caveat is that the experimental data had potential biases/initialization errors present in the state of the quadrotor. As seen in figure (5) our simulation achieves the desired position much quicker than that of the experimental results for all different flights and gains tested.It is noticeable that the data shows an offset if around 10 cm in the z-direction in the beginning of each flight data. This offset can be attributed to the mocap device which is mounted towards the top of the drone. When the drone is sitting on the ground at 0 m, it can be seen that the mocap device measures its own position which cannot be 0 since the drone cannot fly below the ground. For this reason the initial position in the z-axis in the experimental data is never 0.

As expected and discussed in Section IV, our simulated flights showed data trends similar to what we found in experiment. The gains used in the first flight were the gains that were implemented in Lab 3 for making the drone hover. As expected there were large inconsistencies in desired and actual position in the z axis since the quadcopter does not have enough time to achieve steady state before it moves to the next assigned point. The Yaw angle deviates significantly from the desired yaw angle of zero. To rectify this the gains were adjusted to over value the gain value that represents the yaw angle. This showed significant improvement in the raw error between desired and actual yaw values as seen in table (2). In flight 2 however there was an increase in positional error which was accounted for in flight 3 by increasing the Q values that corresponded to the position in the z axis and the velocity in the z axis. This further reduced the steady

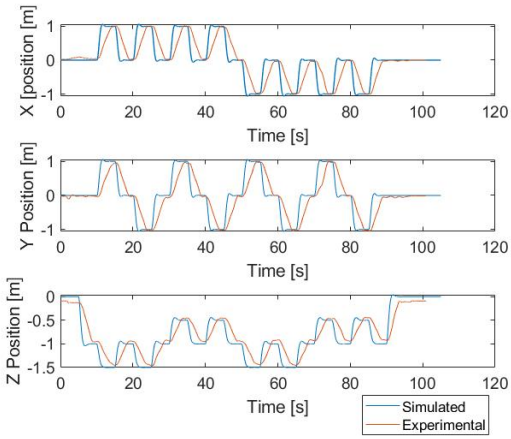
state error in the experimental data which agrees with our simulation data. Table (2) presents the values of Errors that were found from experimental data.



(a) Flight 1



(b) Flight 2



(c) Flight 3

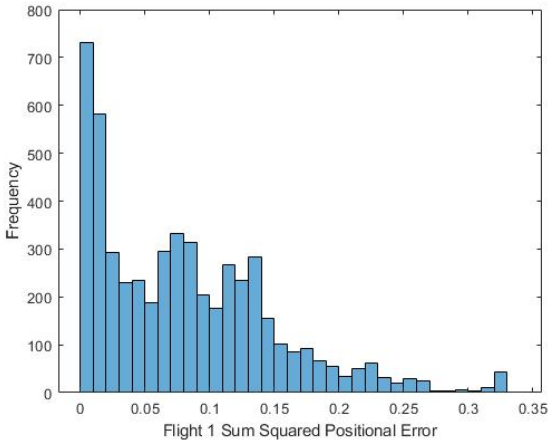
Fig. 5 Experimental and Simulation Positions on the x,y and z Axis Plotted Against Time for 3 Flights

The trends predicted in the simulation can be seen in the experimental data as well. The flights conducted in the lab chose similar weights and used similar reasoning as our simulation thus showing similar results.

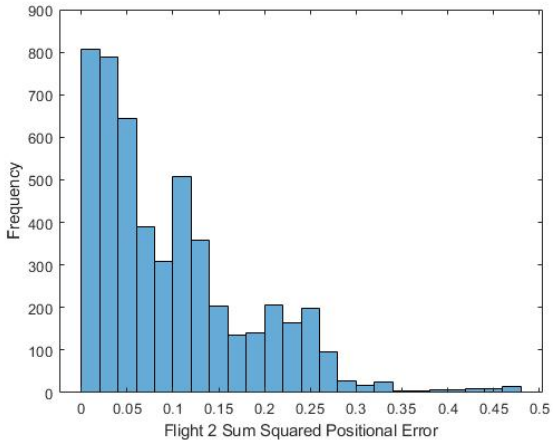
Fig. 6 shows a histogram of the sum squared positional errors found in experimental data. The plot makes it extremely easy to see that the least error in our experiments came from the third flight where our gains were tuned to give more weightage to the z-axis as well reduce the error in the yaw angle.

Table 2 Positional and Angular Errors for 3 Experimental Flights

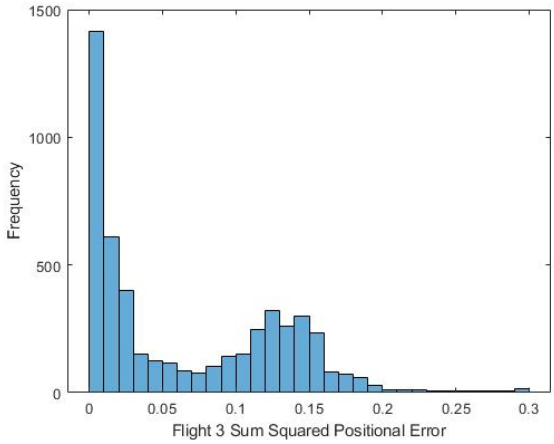
	Sum Squared Positional Error		Steady State Error(m)			Mean Absolute Yaw Error(rad)
	Mean	Standard Deviation	x	y	z	
Flight 1	0.0814	0.0681	0.0023	0.050	0.141	0.020
Flight 2	0.0993	0.0845	0.0099	0.009	0.173	0.003
Flight 3	0.0669	0.0637	0.0145	0.019	0.057	0.008



(a) Flight 1



(b) Flight 2



(c) Flight 3

Fig. 6 Sum Squared Positional Error Presented in a Histogram for Three Experimental Flights

The trends predicted in the simulation can be seen in the experimental data as well. The flights conducted in the lab chose similar weights and used similar reasoning as our simulation thus showing similar results.

Fig. 6 shows a histogram of the sum squared positional errors found in experimental data. The plot makes it extremely easy to see that the least error in our experiments came from the third flight where our gains were tuned to give more weightage to the z-axis as well reduce the error between the actual and the desired yaw angle.

VI. Conclusion

In this report a previously designed closed feedback controller was tested at multiple points along a trajectory. The controller was adjusted by changing gains to make sure that the hummingbird quadcopter maintained a yaw angle of zero and could reach the target destination quick enough to stabilize before moving to the next point in the trajectory. The controller was tested in a MATLAB simulation, and a course-staff controller through a real world experiment. The results from the experiment were compared to the results in simulation and analyzed. Overall, the simulation shows the same trends and mostly matches up with the real-world data. The controller the students designed was verified through MATLAB simulation, and performs nominally in the standard flight test, but with less accuracy in stress tests where the distances between points are doubled or the times between each points are halved. The difference was only noticeable at double, or triple the standard flight test regime.

The work done in this report can be vastly improved by two methods. First the simulation and controller design can be improved by extensive tweaking of the gain matrix. This can be done manually or through an optimization algorithm. A second method of improving the results of this report could be by conducting the experiment in a zero-disturbance environment so as to avoid any environmental disturbances in the experimental data.

Acknowledgments

In this lab report, Vignesh Sella worked on the formatting, introduction, analysis and improvements, and a paragraph in the conclusion. Aaryaman Batra worked on the control design and implementation, experimental results, and a paragraph in the conclusion. Both students worked together on the abstract and developing the simulation code and controller.

References

[1] Sella, V., "Lab 3: Hovering Control," 2020.

[2] Sella, V., "Lab 2: Parameter Estimation and Pitch Control," 2020.

[3] AE483, "Lab 4 Pre-Lab Manual Point-to-Point Control," 2020. URL https://compass2g.illinois.edu/bbcswebdav/pid-4921852-dt-content-rid-64501907_1/courses/ae_483_120208_194370/lab4_prelab%281%29.pdf.