

Group members: Drew, Chris, Alex H, Alex K
Lab 7 - Quadcopter Parameter Estimation

Introduction

In this lab, we will be estimating various parameters of the quadcopters. To do this, we will be using sensors on the quadcopters to measure the states and then we will use MATLAB and a LSE equation to estimate the parameters. Furthermore, we will use this LSE equation to prove that the moment of inertia of the rotor is truly negligible, and can be ignored when doing calculations for quadcopter in the future. This lab will serve as a good introduction into the quadcopter as well as give us good practice with using MATLAB and doing least square estimates.

To start this lab, we plugged the Crazyflie dongle into a computer, which will help transmit and receive data between the computer and the quadcopter. The dongle is placed in a USB port closest to the quadcopter, not behind the computer, for better signal strength.

We set the radio number in our python code to the appropriate number. We looked through the python code to understand each section, including what it does and how it works. This will help us with any debugging required in the lab, as well as learning how to write better code for control systems in the future.

After looking through the Python code, we had to make minor changes to ensure that, after testing, the quadcopter would land perfectly and not crash land unexpectedly. The main change added is a line that makes it move down slowly at the end instead of turning off mid-air and crashing, risking damage to the hardware itself.

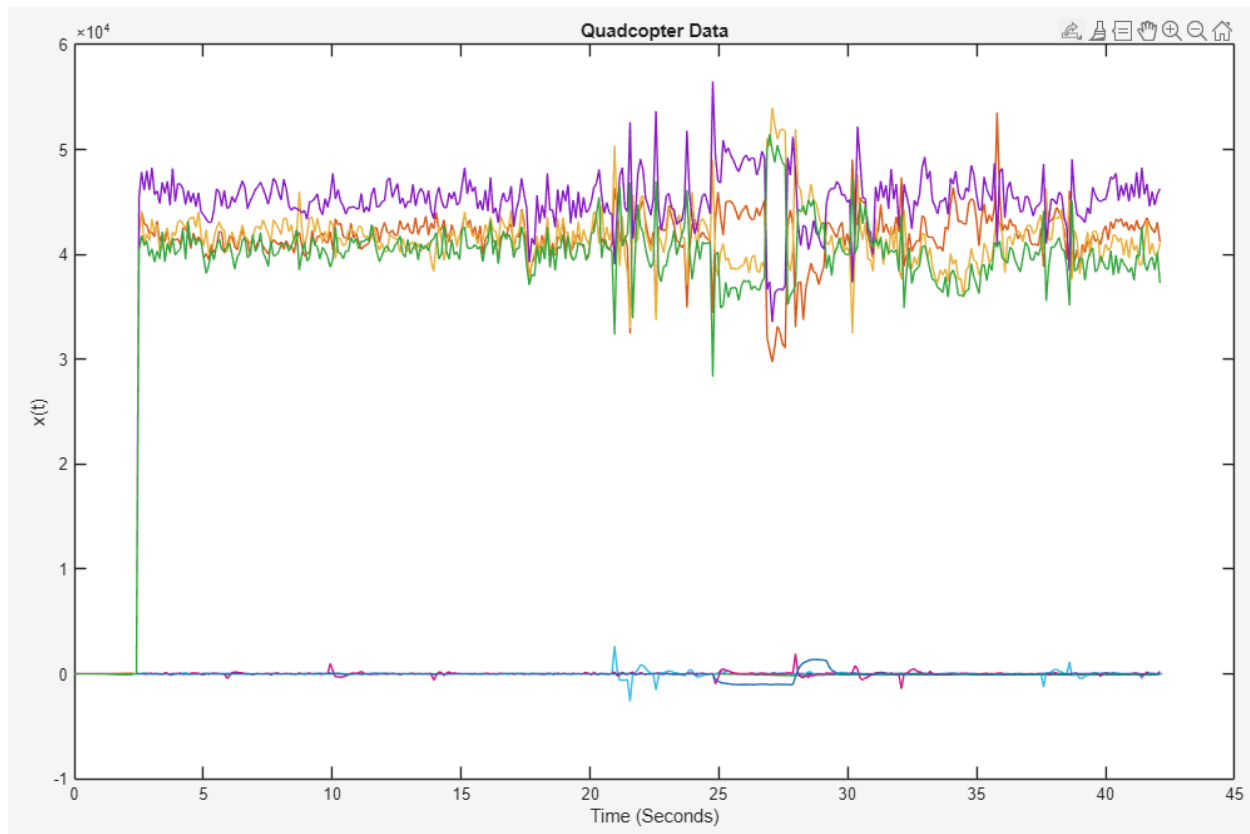
We placed the quadcopter in the middle of the padded section of the lab, and ran [Lab7.py](#). After some significant technical issues involving the propellers on the quadcopter and making the small addition of an ascent and descent to the beginning and end of the code respectively, our quadcopter took off and completed its routine as expected.

We edited the MATLAB code to read in the data from the CSV file and save it to a matrix for easy use as we continue the lab.

After reading the CSV file into MATLAB, we converted the PWM values to angular velocities using a linear mapping equation ($w = 0.04 \times \text{PWM}$).

10: for this part of the lab, we plotted all 19 ($x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}, \ddot{x}, \ddot{y}, \ddot{z}, w_1, w_2, w_3, \text{ and } w_4$) of our data sets against time on

the same axis. The plot can be seen below:



Our group calculated the LSE equation in matrix form in order to calculate the specified unknown parameters. We used MATLAB to solve for all of the variables not directly measured. Now that we have all of the necessary variables, we were able to move on to using LSE to estimate the system parameters.

Using the Formulated $AN \Theta = YN$ matrix equation, we then solved for the five unknown parameters using MATLAB using the inverse. Angular accelerations were estimated using the difference of each variable with sampling time $T = 0.1s$.

Using the general model in MATLAB we were able to estimate the 6th parameter I_r and the output can be seen below. We estimated the value of I_r to be $-1.1051e-7$ which is significantly less than the moment of inertia and is very close to 0, which is why I_r is negligible.

```

theta_n =

1.0e-03 *

0.0971
0.1187
0.0150
0.0000
0.0000
-0.0001

Ixx = 9.7147e-05 kg·m2
Iyy = 1.1865e-04 kg·m2
Izz = 1.4985e-05 kg·m2
KT = 2.9835e-08 N/(rad/s)
KD = 1.1332e-12 N·m/(rad/
Ir = -1.1051e-07 kg·m2

```

Conclusion:

Although the quadcopter itself proved difficult to work with, encountering a number of technical issues which had to be dealt with, all group members were familiarized with the quadcopter and the code we will be working with for this and the next lab. By taking the quadcopter through a simple test run with the provided lab python code, we were able to collect enough data to calculate the necessary unknowns in part 13. We also confirmed that, indeed, the value of I_r is negligible at roughly -1×10^{-7} . This lab served as an introduction to all of the basic concepts involved with quadcopter analysis, many of which will be expanded on in the next lab.

Group members: Drew, Chris,Alex H,Alex K

Lab 8 - Navigation/Obstacle Avoidance

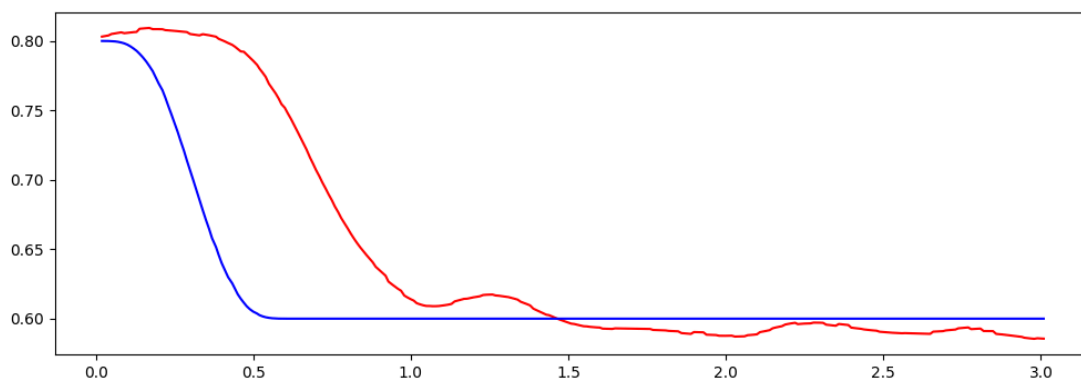
Introduction

In this lab, the objective was to develop and analyze quadcopter navigation and obstacle avoidance using PID control. The quadcopter's motion in the x, y, and z directions was stabilized through systematic tuning of proportional, integral, and derivative gains to achieve desired performance characteristics, including minimal overshoot and stable tracking. These tuned controllers were then applied to waypoint navigation tasks, followed by obstacle avoidance using onboard proximity sensors. The results demonstrate how control parameter adjustments directly influence system stability, trajectory tracking, and responsiveness in dynamic environments.

Part 1 - PID Tuning:

To begin the lab, we ran a python script provided to us in the lab, containing a PID tuning interface to send to the quadcopter. For the PID values in the x, y, and z direction, we experimented with various PID values with the final goal to be to have less than 10% overshoot in the plots. Our final PID values and their resulting plots in the x, y, and z directions can be seen below:

Z-Direction PID Tuning Results



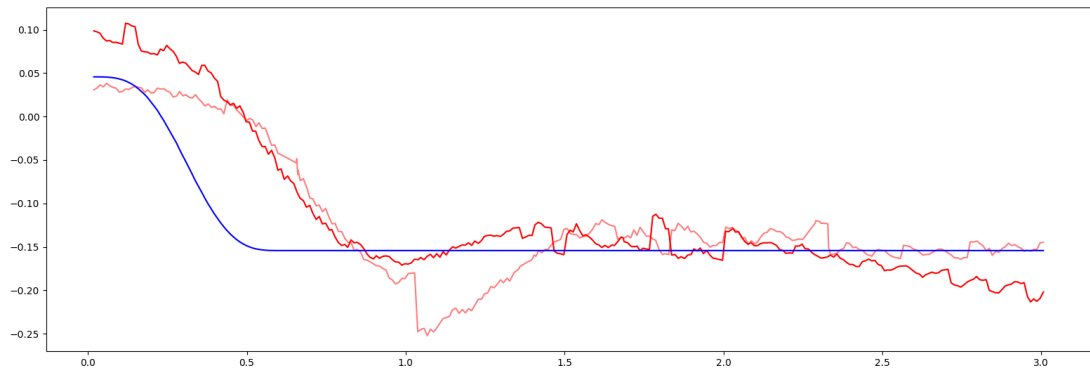
PID Values:

$$K_p = 2$$

$$K_i = 0.5$$

$$K_d = 0$$

y-direction



scale_Kp

4.0

05101520253035404550556065707580859095100

scale_Ki

0.0

036912151821242730333639424548

scale_Kd

0.0

036912151821242730333639424548

vMax

0.0

05

Choose an axis!

y

pos

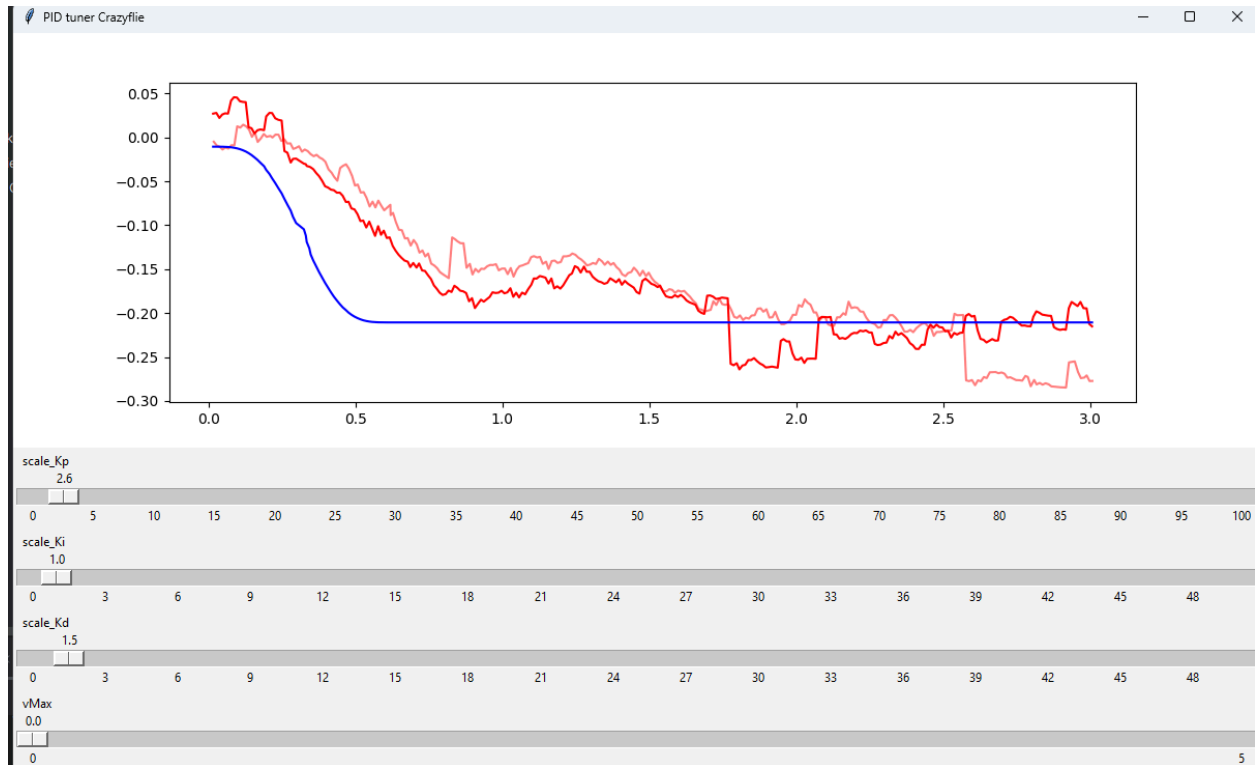
Choose velocity or position!

SEND PID GAINS

DO STEP

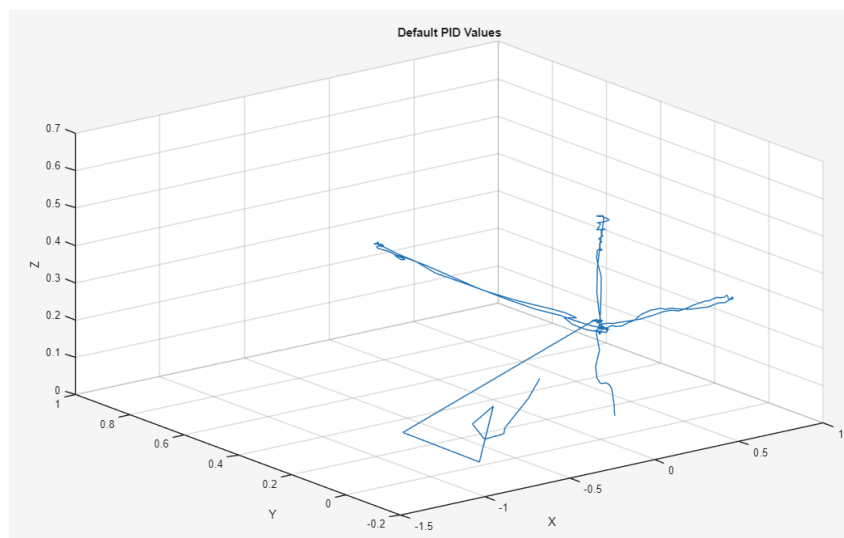
STOP

X-direction

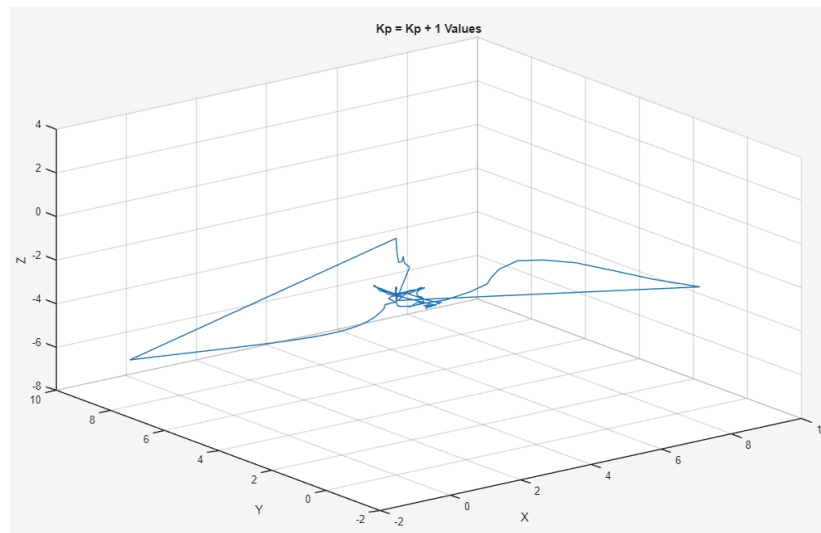


Part 2 - Navigation:

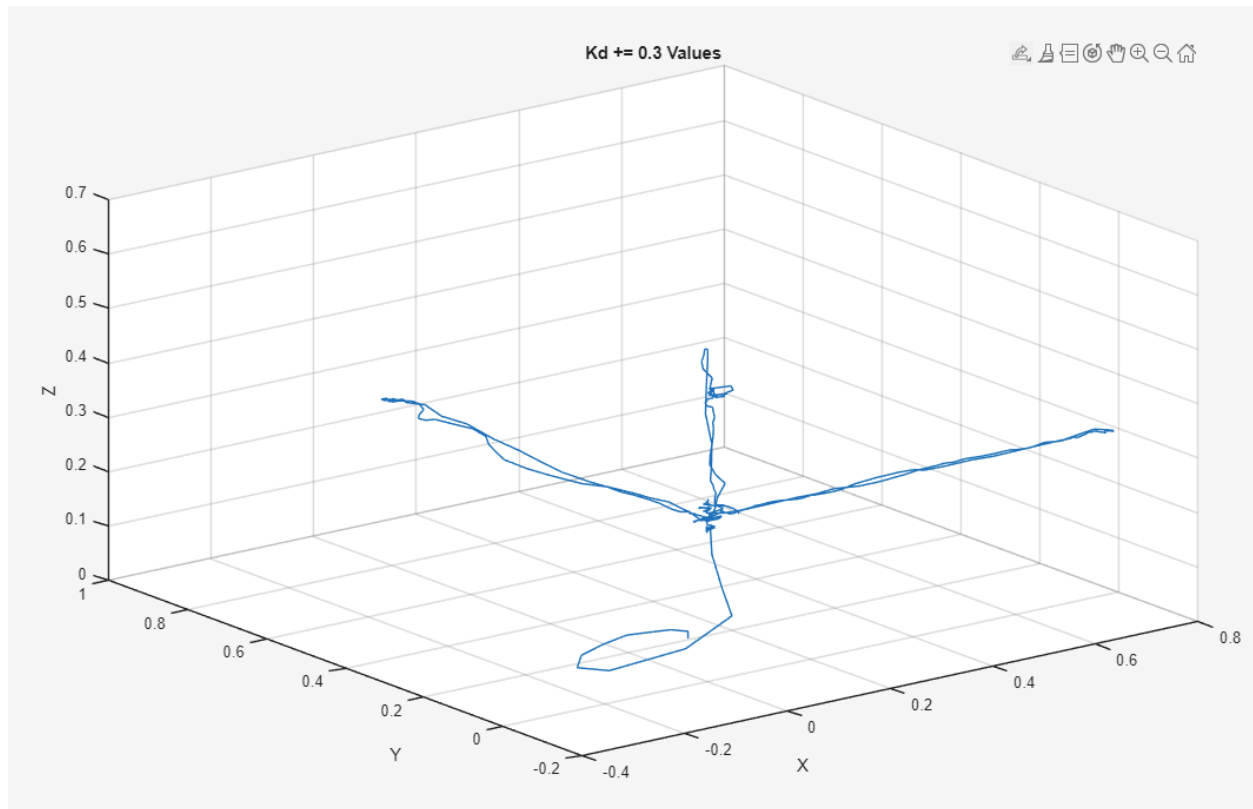
In this portion of the lab, we used the PID values that we had developed to test the quadcopter's navigation abilities. We started by having it fly in a sequence that took it in the x, y, and z directions with the PID values that we found. We found that it did a good job following the sequence, meaning that our PID values worked as expected. A plot of the quadcopter's trajectory is included below.



After testing the quadcopter's navigation with our values, we increased the proportional gain by one and ran the quadcopter again. When doing this, we found that the quadcopter destabilized and crashed almost immediately. This is a good demonstration of the dangers of altering PID parameters without a good tuning method to find the correct new parameters first. A plot of the quadcopter's trajectory is shown below.

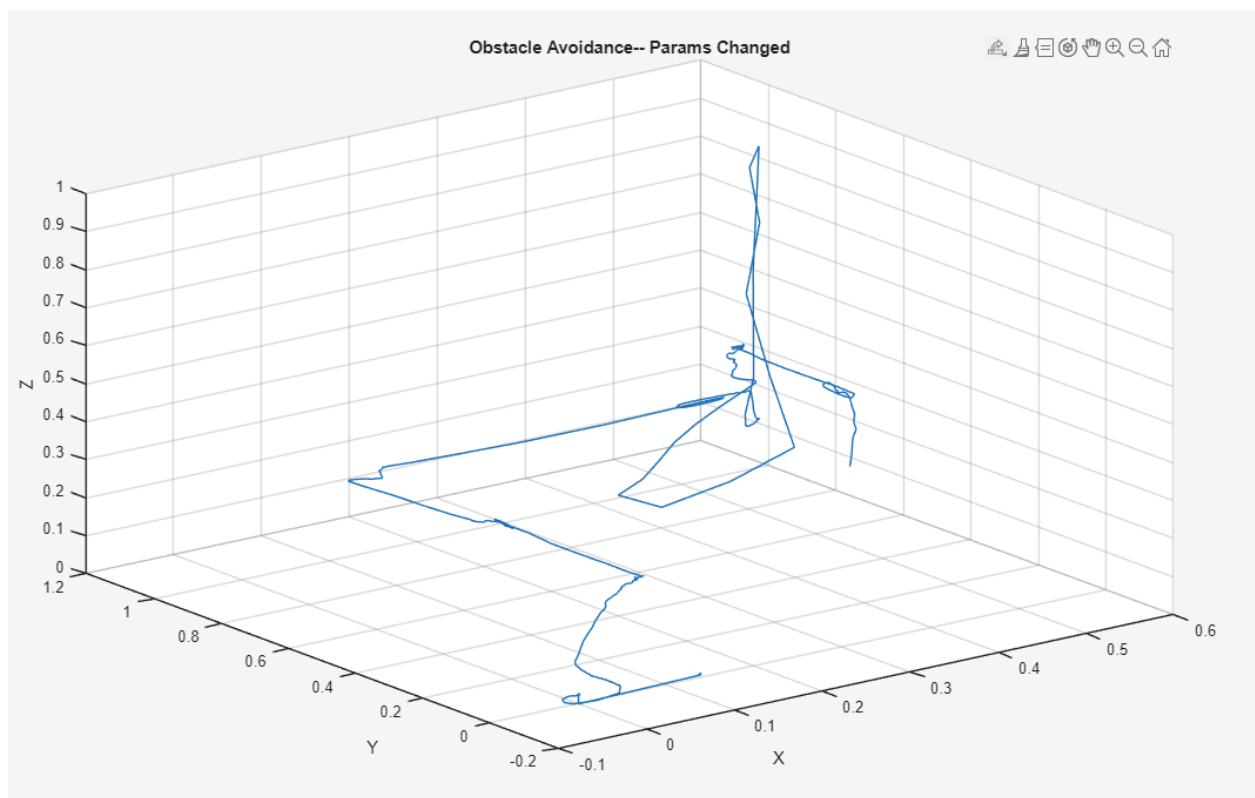
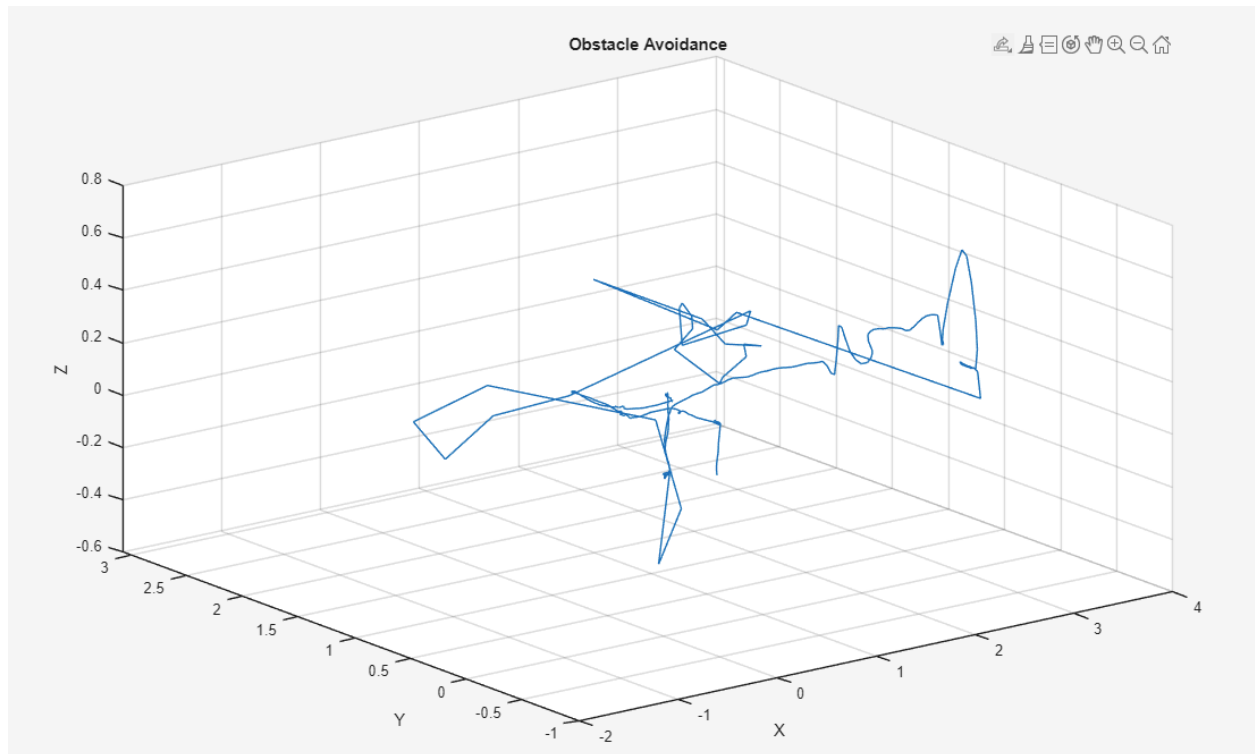


The third navigational test that we performed was setting the proportional gain back to the value that we had tuned and increasing the derivative gain by 0.3. When we ran the quadcopter with these settings, we found that it actually functioned pretty well. It was noticeably choppier in its movements than it was in the original navigation test, but it followed the trajectory well. We believe that this is because while the derivative gain was increased, it wasn't increased enough to cause the system to lose stability like the proportional test did. A plot of our output trajectory is shown below.



Part 3—Obstacle Avoidance:

For the final part of the lab, we were provided a python script used for obstacle avoidance, which works by using the proximity sensors in the quadcopter. Within the MATLAB script, there were two variables we had to manipulate: MIN Distance and Velocity. The MIN Distance variable is the proximity of an obstacle is away from the quadcopter to trigger a response and the Velocity variable is the speed at which the quadcopter moves away from the obstacle. After setting these values, we tested the avoidance by putting our hand close to the drone mid flight and observing how the quadcopter moves away from our hand. The resulting paths are plotted and can be seen below. A possible change to the obstacle avoidance part of the lab would be to specify a specific range that the quadcopter would need to go through, therefore if we needed the quadcopter to go through a field of obstacles, the quadcopter would need to make forward progress through the field.



You can see where we shot it up and it almost crashed

Conclusion:

In this lab we returned to the quadcopter from the previous lab. During the navigation section of this lab we encountered few problems, the notable exception being an issue with battery life. This issue saw us rotating through almost every battery available in the lab looking for one which still held a charge. Eventually finding a serviceable battery, we had no other major issues during the navigation section. Our initial run of obstacle detection was unsuccessful, but an attempt with no changes immediately afterwards was successful. Changing the velocity and detection range went just as planned, and the quadcopter had no issues adapting to the new parameters. At the end of this lab we certainly had a significantly better understanding of our quadcopter and the mechanisms by which it is piloted.