Control Experiment: Model Helicopter

A Senior Project

presented to

the Faculty of the Department of Aerospace Engineering

California Polytechnic State University, San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Bachelor of Science

by

Matthew D. Lattanzi

June, 2012

Control Experiment: Model Helicopter

Matthew D. Lattanzi[1]
California Polytechnic State University, San Luis Obispo, CA, 93401

This report describes the design and analysis of a control experiment to be implemented in the AERO 320 curriculum. The purpose of the experiment is to give the students hands-on experience working with a control system. In addition, the experiment aims to demonstrate the effect a proportional, integral, derivative (PID) controller has on a control system. The system to be controlled is a model helicopter, constrained to vertical motion. The physical system was built using radio controlled (RC) components, off-the-shelf products, and custom designed parts. The system was tested using an RC transmitter and receiver to manually control the height of the model. A Simulink model with PID controller and simplified plant model was developed and analyzed. A Routh-Hurwitz tabulation was conducted to determine the range of controller gains that would stabilize the system. The gains were varied to determine the effect on the step response and Bode diagram for the helicopter model. The results showed that the changing gain values did not affect the phase margin or gain crossover frequency of the system. Future improvements to the physical system and Simulink model are covered in detail at the end of the report with end goal of testing the system closed loop.

I. Introduction

The AERO 320 course is an introduction to linear time-invariant (LTI) control theory and currently it is limited to just that, theory. Introductory control theory classes are often dreaded by students because the material can be abstract and difficult to understand. The goal of this project is to design, build, and implement a lab experiment for the AERO 320 class. This experiment is intended to be a teaching aid for the students by giving them the opportunity to work hands on with a control system.

The proposed design of this lab experiment is a simplified model helicopter. The helicopter will be constrained to vertical motion, moving up and down a linear steel pole by varying the speed of the brushless DC motors and therefore the lift created by rotor blades. Using MATLAB/Simulink students will command the helicopter up some height of the pole. In real time, by utilizing the plotting capabilities of Simulink, they will be able to see the distance, velocity and acceleration plots versus time. The students will determine the gains of a proportional, integral, derivative (PID) controller that will filter the error signal produced by comparing the student input height and the fed back current height. Adjusting the gains associated with the different components of the PID controller will demonstrate the effects on the stability and closed-loop system response parameters.

This report will explain the methodology while designing this experiment. In addition, the current system components are described as they pertain to the overall test apparatus. Testing was done to ensure the system was functioning correctly and the results are presented in the body of the report. Problems with the design were discovered during the course of this project. The problems along with their solutions are described. Finally, the remaining problems are described along with their proposed solutions and suggestions for improving the experiment. The Appendix contains all additional information including technical drawings, oscilloscope measurements, and code for the Arduino microcontroller.

II. Apparatus

The design of the experiment was kept as simple as possible with a majority of the components purchased off the self. Off the shelf products allow the experiment to be replicated easily and for changes and replacements to be implemented without any custom manufacturing. The following section will go through the major assemblies and components that make up the system and describe the components within the assemblies. In addition, this section will provide any specific information that would be valuable to anyone trying to assemble the experiment or trying to replace parts within the system.

[1] Undergraduate Aerospace Student, Aerospace Engineering Dept., 1 Grand Ave., San Luis Obispo, CA 93407

1. Carriage Assembly

The carriage assembly is where the components essential for flying model are mounted. The carriage consists of an aluminum strip, two linear bearings, and a black plastic sleeve. The aluminum strip measures 15.5”x1.5”x0.125” and has holes drilled into it, allowing components to be securely attached. The two linear bearings have an inner diameter of 0.5” and allow for smooth vertical motion. The sleeve separates the linear bearings, which prevents the bearings from binding on the pole when a moment is produced by unbalanced lift forces from the two propellers on either side of the carriage. Each linear bearing is attached to the sleeve with two screws. One set of screws attaches the aluminum strip and the linear bearing to the sleeve. Table 1 lists the components of the carriage assembly along with a picture and its specific purpose. Figure 1 shows the assembled carriage.

Detailed drawings of the aluminum strip and sleeve can be found in the Appendix A. The two linear bearings were purchased on vxb.com, part number: Kit8084.

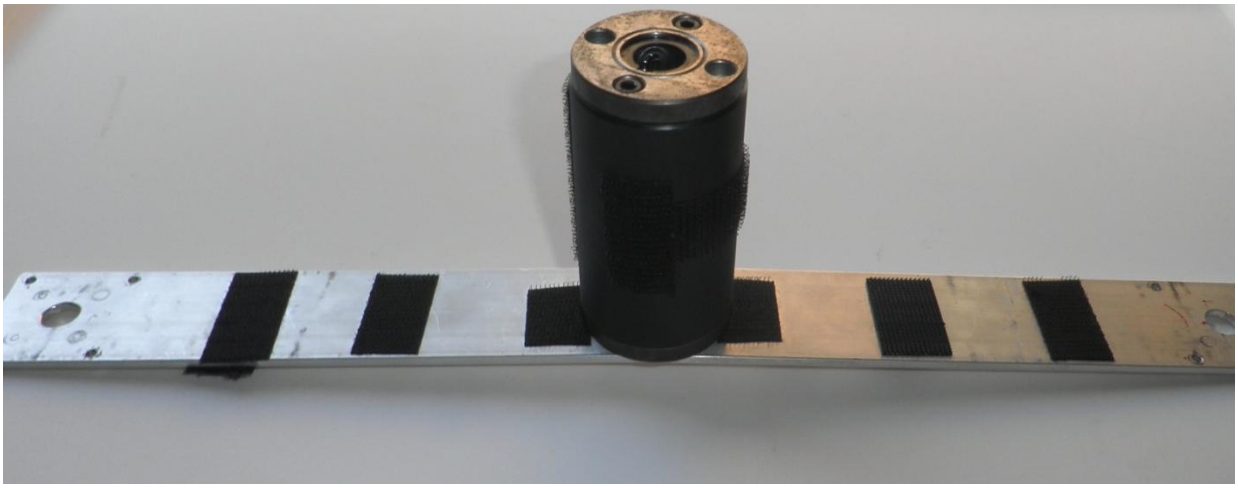
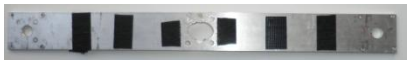


Figure 1. Carriage Assembly

Table 1. Carriage Assembly Components

Component Picture	Component Name	Purpose
	Aluminum Mounting Strip	– Mounting surface for flight essential components
	Sleeve	– Separate linear bearings to prevent binding – Mount flight hardware
	Linear Bearing	– Minimize friction between pole and helicopter model
	Allen Head Screws	– Attach linear bearings to sleeve and aluminum mounting strip

2. Motor Assembly

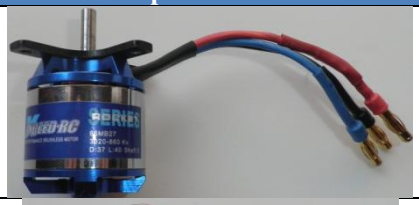
The motor assembly consists of all the components required to power and run the two motors. The assembly consists of two brushless DC motors, two electronic speed controls (ESC), two lithium polymer batteries, the propellers, and propeller adapters. When controlling the motor manually, this assembly also includes the radio controlled (RC) transmitter and receiver. Table 2 lists all of the components and their function within the system.

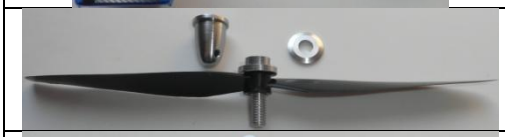
The motors used in this experiment are two Exceed-RC Rocket Series 3020-860 Kv brushless DC motors. The 860 Kv rating is the no-load gain constant of the motor. The motor speed increases by 860 rpm every volt increase seen by the motor. Each motor has two propellers attached, one propeller to the main motor shaft and one propeller to an adapter that can be screwed on to the bottom of the motor. The extra propellers were added to increase the lift of the overall system. The propellers used in this experiment are two 9x5 and two 9x5R plastic propellers. The first number in the propeller title indicates the diameter of the blades and the second number is the pitch of the propeller. Theoretically, the propeller will move 5 inches every revolution. The higher the pitch number the farther the propeller will travel per revolution. The capital "R" means that the blades are reversed. The reverse propellers spin opposite of the normal propeller to create positive thrust. One motor has a pair of normal pitch propellers and the other has a pair of reverse propellers attached to the motor shafts. The opposite spinning motor prevents the entire carriage from spinning, canceling out angular moment of the other motor. The motors are attached to the carriage assembly by four 4-40 bolts and nuts. In addition, #6 lock washers were utilized to prevent the motors coming loose in the middle of a test run because of the considerable amount of vibration in the system.

The ESCs generate the required pulses to keep the motor spinning based on the input from the receiver. The ESCs are two Exceed-RC Proton-18A. The ESC has three separate sets of wires. The thick red and black wires are used to connect with the battery. The connection type is a 3.5 mm bullet connector. The trio of wires, opposite of the battery wires, connect with the motor. The connection type for the motor wires is a 3.5 mm female banana plug. The ESC is controlled via a trio of wires, one of which is the white signal wire. The signal going into the input is a pulse width modulated (PWM) signal. The two remaining wires in the trio are ground (black) and a 5 V power source (red) used to power the RC receiver. The red wire is not useful when controlling the system closed loop via the Arduino rather than an input from the RC receiver and transmitter. Table 5 in the Appendix B gives the signal characteristics being sent to the ESC, leaving the RC receiver. The measurements were taken at different transmitter states: throttle low/trim low, throttle low/trim high, throttle high/trim low, and throttle high/trim high. The PWM signal output of the Arduino should match the output signal characteristics of receiver when the helicopter is controlled closed loop.

The battery is the main power source for the motors. The current batteries are two Blue LiPo 15C 3-cell 950mAh lithium polymer (LiPo) batteries. The 950mAh batter capacity was chosen to give the motors about 15 minutes of operation before recharging the batteries. The 15C is a measure of how much current can be output by the battery at any time. The number in front of the 'C' is a multiplier and the maximum output current can be calculated by multiplying the number in front of the 'C' with the battery capacity. The resulting number should be higher than the maximum burst current of the motor to ensure full performance from the motors. The connector type for the battery is a 3.5 mm bullet connector.

Table 2. Motor Assembly Components

Component Picture	Component Name	Purpose
	Brushless DC Motor	– Provide power to spin the propellers
	Electronic Speed Controller (ESC)	– Convert PWM signal from receiver to coordinated pulses that keep the motor spinning

	3-Cell, Lithium Polymer Battery	– Power source for the motor
	Motor Shaft Propeller and Adapter	– Attach propeller to main motor shaft – Create lift
	Bottom Motor Propeller and Adapter	– Additional propeller attached to motor – Create lift
	Fastener, Nut, and Lock Washer	– Secure motor to aluminum mounting strip
	RC Transmitter	– Control motor speed, when manually controlling model
	RC Receiver	– Receive transmitter signal and generate a PWM signal for ESCs

3. Arduino UNO Microcontroller

The Arduino Uno microcontroller is an open-source electronics prototyping platform which is easily programmable and interfaces with MATLAB/Simulink. The Arduino language is based on C/C++ and the software can be downloaded for free at www.arduino.cc. Users looking for a more user friendly coding experience can use Microsoft Visual Studio with the Arduino add-in.

The Arduino functions as the brain of the system. It does all of the loop calculations and is the interface between MATLAB/Simulink and the actual helicopter model. Table 3 lists all of the components that the Arduino is connected to and the connection type. The Arduino calculates the distance that the helicopter carriage has traveled based on the outputs of the Ping sensor. Next, the Arduino will calculate the output of the PID controller, generate a PWM signal, and send the new duty cycled PWM signal to the ESCs on the carriage. During a test run, the Arduino will be sending the distance information back to MATLAB/Simulink for the students' analysis. Coding in the Arduino is not part of the lab, so the desired gains will need to be sent to the Arduino for processing from MATLAB/Simulink via the serial link.

Table 3. Arduino Microcontroller Connections

Component	Connection Type
MATLAB/Simulink	USB A-B
Motor ESC (2)	22 AWG wire
Ping Distance Sensor	22 AWG wire

4. Ping))) Ultrasonic Distance Sensor

The Ping sensor will provide the feedback for the control loop. The Ping works by sending out an ultrasonic burst, waits for its return and outputs a pulse that is proportion to the time it took to travel from the Ping

to the object and back again. The Ping has a range of about 1-118 inches. The Ping has three connecting pins: Ground, 5 V, and I/O signal pin. All pins will be connected to the Arduino via 22 AWG wires.

A distance sensor was chosen for the feedback, as opposed to an accelerometer, because it eliminated the need for extra signal processing. If an accelerometer was used to indirectly measure the distance travelled by the model helicopter, two integrators would have to be added to the forward path of the control loop. The extra two poles would have complicated the loop dynamics and compromised simple models for students.

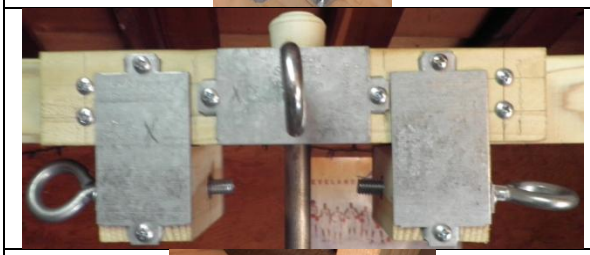
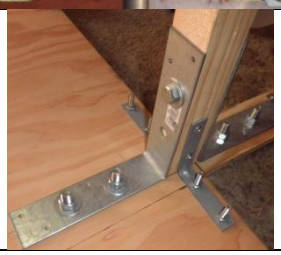
5. Test Stand Assembly

The test stand assembly is the platform on which the helicopter model can be tested. The main function of the test stand is to support the 6' high, ½" diameter linear steel pole which guides the motion of the model helicopter. The pole is supported at the bottom and top to help damp out vibrations from the spinning blades. Table 4 highlights some of the main features of the test stand assembly. Figure 2 shows the test stand assembly with the model flying in manual control mode with RC transmitter and receiver.

The base is made from ½" thick plywood cut into a 3' square. The base is set on eight rubber feet to keep the test stand off and level to the ground. The base support holds the pole steady when the model is being tested. The base support is a 10" long, 4"x4" piece of wood with a 5/8" diameter hole bored through it. Eight 3/8" diameter, 3 ½" long eye screws, two on each long face of the 4"x4", secure the pole and, in addition, allow the pole to be adjusted to be straight up and down.

The top support for the pole is a cross-bar that is supported by two vertical 70", 2"x2" pieces of wood. The cross-bar is made from shorter sections of 2"x2" wood pieces. There are two 3/8" diameter, 4 ½" long eye screws and two 3/8" diameter, 3 ½" long eye screws that secure the pole at the top support. The two vertical supports are made rigid by six 1/8" diameter, adjustable tension steel cables. The cables not only support the vertical pieces but allow the cross-bar to be moved in two dimensions, which enables the eye screws to grip the pole.

Table 4. Main Features of Test Stand Assembly

Component Picture	Component Name	Purpose
	Base Support	– Hold the pole stable and adjust verticality
	Top Support	– Damp out vibration from spinning propellers
	Base of Vertical Cross-bar Supports	– Support cross-bar

		Variable Tension Steel Cables	– Damp out vibration – Add rigidity to cross-bar supports

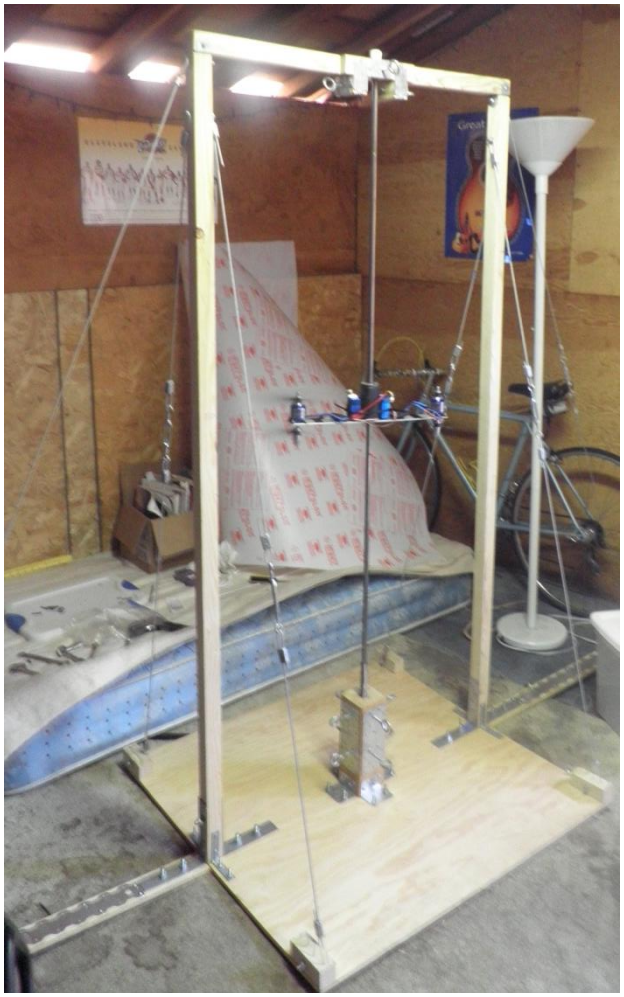


Figure 2. Test Stand Assembly with System in Manual Control Mode

III. Procedure for Manual Control

The manual control mode is an open loop setup with helicopter motors being controlled by the RC transmitter, the first step before testing the helicopter model closed loop. The purpose is to test whether all the mechanical components work together and can lift the model. The model was successfully tested with manual control. In the process of testing the model, it was discovered that the ESCs do not start at the same throttle position, or duty cycle. To overcome this problem, the ESCs were connected to two different receiver outputs, and controlled with two separate throttles on the transmitter. In addition, the motor gains may not be exactly the same, meaning that the speed of the motors do not increase at the same rate with increasing throttle position or duty cycle. To counteract the differences in the ESCs and motor start times during closed loop testing, the Arduino will be required to output two different PWM signals with separate offsets for the start times and separate gains to compensate for different gains through the ESCs and motors. Table 6. Motor/ESC Initial Spin Duty Cycle Commanded Using ArduinoTable 6 in Appendix D shows the difference in initial duty cycle to start each motor/ESC combo spinning. The gains will also need to be determined experimentally.

The following list describes the steps necessary for the setup and control of the helicopter in the manual control mode.

Necessary Items

1. Test stand assembly
2. Carriage assembly
3. Motor assembly
4. All fasteners, nuts, and lock washers that connect the components within the assemblies
5. RC transmitter and receiver

Preparation

1. Charge both LiPo batteries to full capacity
    a. This step is absolutely necessary
2. Assemble test stand
3. Assemble carriage
4. Attach motors to carriage with correct fasters, nuts, and lock washers
5. Attach propellers to motors

Motor Setup and Test

1. Connect motor to ESC
2. Connect ESC signal wires to "BAT" receiver output
3. Connect battery to ESC
    a. The motor should start to beep steadily, this is because it does not register a signal from an active transmitter
4. Press and hold the "Bind" button on the transmitter
5. While still holding "Bind", switch on the transmitter
6. Continue to hold the "Bind" button for 3 seconds
    a. This initializes the ESCs
7. Switch off the transmitter and let go of the "Bind" button
    a. The motor will continue to beep
8. Move the ESC signal wires to **Channel 3** on the receiver
    a. **Channel 3** is connected to the left, up and down stick on the transmitter
    b. Check that the ESC signal wires are in correct orientation in receiver output port
        i. The correct orientation is pictured on the receiver
9. Move the throttle (stick) to the lowest position
10. Check that no wires will be hit by spinning blades
11. Turn the transmitter on
    a. The motor should emit a series of quick beeps, followed by a longer beep
12. Move the throttle up slowly to check that the motor is spinning in the correct direction
    a. Because of the complex control necessary to initially start a brushless DC motor, the motor can start to spin the opposite direction then switch directions with increased throttle. This actually happens often
    b. If **yes**, skip to Step 13
    c. If **no**, continue to next step
13. Switch the connection of any two ESC motor wires to reverse the direction of the motor
    a. Again, check the direction of the motor
14. Turn off transmitter
15. Disconnect the first ESC from the receiver
    a. It will start to beep again
16. Repeat Steps 1-13 for the second motor
17. Disconnect the second ESC from Channel 3 and connect to **Channel 2**
    a. **Channel 2** is connected to the right, up and down throttle on the transmitter
18. Reconnect the first ESC to Channel 3
19. Move left throttle to lowest position
20. Again, check that the wires are out the way of the propeller blades
21. With right thumb, hold the right vertical throttle in the low position

    a. The right throttle is returned to middle position because of springs inside the transmitter
    b. Useful for servo motors, which would normally connect to all channels, except Channel 3
22. While still holding the right throttle in low, switch on transmitter
23. Slowly ramp up both motors, keeping the carriage from rotating on the pole
24. Apply a higher throttle position for more lift
25. The batteries should provide about 10 minutes of flight before they are discharged below the level where cannot provide enough power to propellers

IV. Analysis

An important step in the process of building a physical system to be controlled is to develop a computer model. The computer model enables the user to predict the response of the system to different inputs or controller gains. In addition, the user can determine the controller parameters to obtain desired performance characteristics. There are many challenges when trying to model a real world system. There will always be nonlinearities in real world systems, which are hard to predict and model. Some nonlinearities include: friction, actuator saturation, and backlash. It is usually beneficial to start with a simplified model and add more complexity as the system is further developed. This project starts with a simplified version of the model helicopter in series with a PID controller to obtain theoretical results and suggests additions that will possibly improve the model for future use.

Traditional state space models begin with the equations of motion that govern the dynamics of the system. For this system, Newton's 2nd Law of motion is the governing equation and is as follows

$$\sum F = L - m * g - F_f = m * a$$

where L is the lift created by the model, m is the mass of the model, g is acceleration due to gravity, F_f is the friction force between the pole and linear bearings, and a is the net acceleration of the system. The mass is easily obtained by using a scale. The acceleration due to gravity is constant at $g=32.2\ ft/s^2$. Friction could be determined through testing of the actual linear bearings or by analyzing the experimental results. For this project it is an unknown quantity. The lift created by the model is a difficult parameter to calculate. *An Introduction to Aerospace Propulsion* describes two different methods for predicting the thrust created by a propeller, both using control volumes. The Integral Momentum Theorem applied to propellers makes the conclusion that the pressure forces outside of the control volume are equal and therefore, the thrust created is

$$T = \dot{m}(u_e - u_o)$$

where $\dot{m}$ is the mass flow rate of the air through the control volume, u_e is the exit velocity of the air leaving the control volume, and u_o is the velocity of the air entering the control volume[1]. In order for the method to be used for the helicopter model, it needs to be expanded to account for two propellers, spinning in close proximity. In addition, experimental data would need to be taken for the propellers used on the model to figure out velocities before and after each propeller at different throttle settings or duty cycles. This project did not go through the steps to experimentally figure out the lift characteristics of the model, but it would be a good addition to the project in the future.

The first iteration of the Simulink model is a basic representation of the actual system with four main components:

- Input, *X(s)*
- PID controller, *D(s)*
- Helicopter plant model, *G(s)*
- System Output and Feedback, *Y(s)*

The model block diagram is pictured in Figure 3.

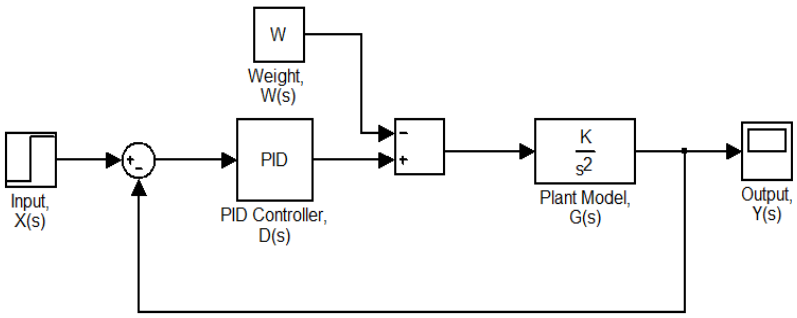


Figure 3. Block Diagram of Simulink Model

The input is a simple step source block that represents the desired distance for the helicopter to travel. The controller for this system is a PID controller. The classic PID controller has three terms: proportional, derivative, and integral. For this report, it is assumed that the reader has a basic understanding of the use and design of PID controllers. An in depth discussion of PID controllers and their design can be found in *Automatic Control Systems* by Golnaraghi and Kuo. The ideal PID controller is represented as the sum of the three terms

$$PID\ Controller \equiv D(s) = K_p + \frac{K_i}{s} + K_d s$$

where K_p, K_i, K_d are the gains associated with the different terms of the controller and s is the Laplace Transform frequency variable. If the PID controller terms are combined with a common denominator, the following transfer function results

$$D(s) = \frac{K_d s^2 + K_p s + K_i}{s}$$

As a first iteration, the helicopter plant model was simplified to a single gain term and two integrators. The current plant model is

$$plant\ model\ \equiv G(s) = \frac{K}{s^2}$$

where K is the gain term and s^2 is the double integrator necessary for turning acceleration into distance. The gain term, K, will be assumed to be the inverse of the mass of the model. The weight, W, in the Simulink model is assumed to be around 2.5 lbf, which makes $K=13$. The value of K will be constant throughout the analysis of this report. By making the assumption that the time constant of the motor is smaller than that of the system, the motor transfer function can be ignored in the plant model. Another way to state the assumption is that the frequency location of the motor pole is greater than the crossover frequency of the open loop system; therefore, it will not affect the system response. The PID gains take into account several different aspects of the system including the motor gain, RPM to lift gain, and the friction force. The feedback gain is unity and represents the distance the model has traveled.

Now that the closed-loop block diagram has been developed, gains need to be selected that will stabilize the system. The first step to stabilizing the system is to find the closed loop transfer function (CLTF). As in *Feedback Control of Dynamic Systems*, the output can be determined by the superposition of the two inputs to the system

$$Y(s) = \frac{D(s)G(s)}{1 + D(s)G(s)} X(s) - \frac{G(s)}{1 + D(s)G(s)} W(s)$$

where *Y(s)* is the system output, *X(s)* is the reference input, and *W(s)* is the constant disturbance to the plant, in this case it is the weight of the helicopter model[2]. The next step is to find the closed loop transfer function. Again, *Feedback Control of Dynamic Systems* gives the closed loop transfer function as

$$CLTF = T(s) = \frac{D(s)G(s)}{1 + D(s)G(s)} = \frac{KK_d s^2 + KK_p s + KK_i}{s^3 + K_d K s^2 + K_p K s + K_i K}$$

It is important to note that the above equation is the CLTF for the input $X(s)$[2]. In addition, the inputs $X(s)$ and $W(s)$ do not affect the characteristic equation of the system. Routh-Hurwitz tabulation was conducted on the denominator of the CLTF, the characteristic equation, to determine the range of gains which would stabilize the system. The resulting gain ranges were

$$K_p K_d K > K_i$$
$$KK_d > 0$$
$$KK_i > 0$$

A set of gains: K=13, K_p=8, K_i=2, and K_d=10 will stabilize the system. The closed loop pole locations are -9.50 and -0.25±0.38i. The calculations for the Routh-Hurwitz tabulation can be found in the Appendix C. Figure 4 shows the closed loop step response. The gains were chosen because they produce a nice step response with <10% overshoot and around 10 second for a 2% settling time.

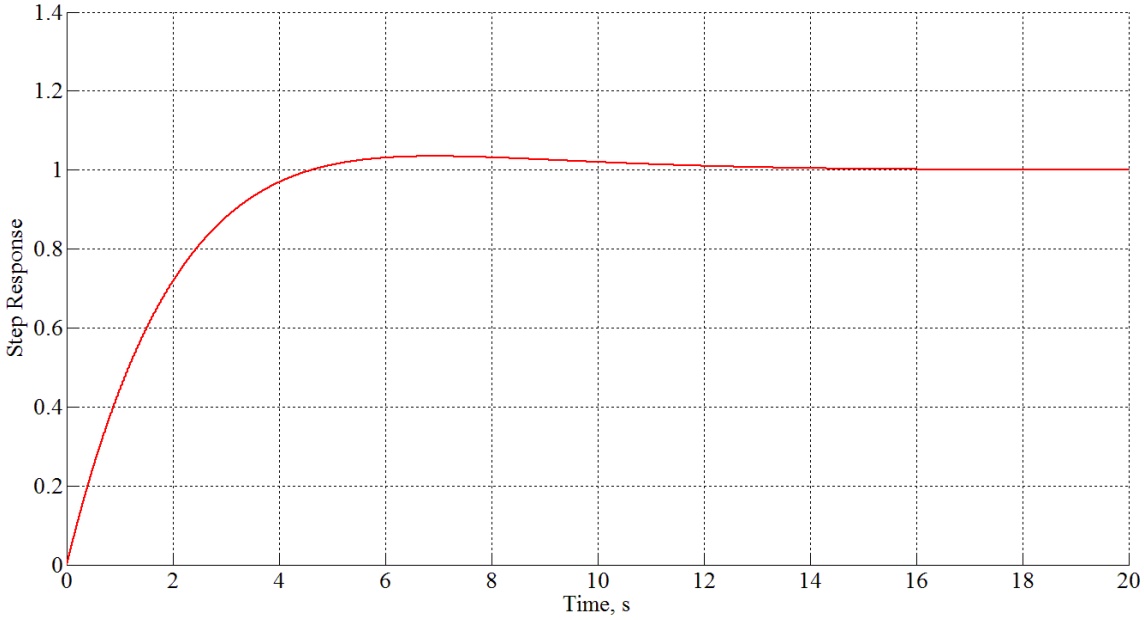


Figure 4. Closed Loop Step Response with Ideal PID Controller: *K=13, K_p=8, K_i=2, K_d=10, W=2.5*

An important system parameter is the steady state error due to the inputs to the loop. The closed loop error function can be found by inspection of the block diagram

$$E(s) = Y(s) - X(s)$$

Plugging in the values for $Y(s)$ and $X(s)$ will yield the closed loop error function of the system

$$E(s) = \frac{1}{1 + D(s)G(s)} X(s) + \frac{G(s)}{1 + D(s)G(s)} W(s)$$

where $E(s)$ is the error function of the system[2]. The Final Value theorem can be used to determine the steady state error for the system [3]. The Final Value theorem states that

$$E_{ss} = \lim_{s \to 0} s\left[\frac{s^3}{s^3 + K_d K s^2 + K_p K s + K_i K} * \frac{X}{s} + \frac{Ks}{s^3 + K_d K s^2 + K_p K s + K_i K} * W\right] = 0$$

where the input *X(s)* is equal to a step and *W(s)* is a constant. It can be seen from the equation above that the steady state error is zero for any level of step response into the system and weight disturbance to the plant.

It is important to understand the effects of changing the controller gains on a system. It is for this reason that all three controller gains were swept through a range of values to see the effects while holding the other values constant. The nominal values for the system were chosen to be the same as in Figure 4. The gains were increased and decreased twice by increments of two. Figure 5 shows the family of step responses while varying the gain K_p. From the step responses it can be seen that the helicopter system does not respond like a classical two-pole dominated system. As expected, the rise time is quicker with increasing K_p but unexpectedly the overshoot decreases. In addition, the settling time is quicker with increasing K_p.

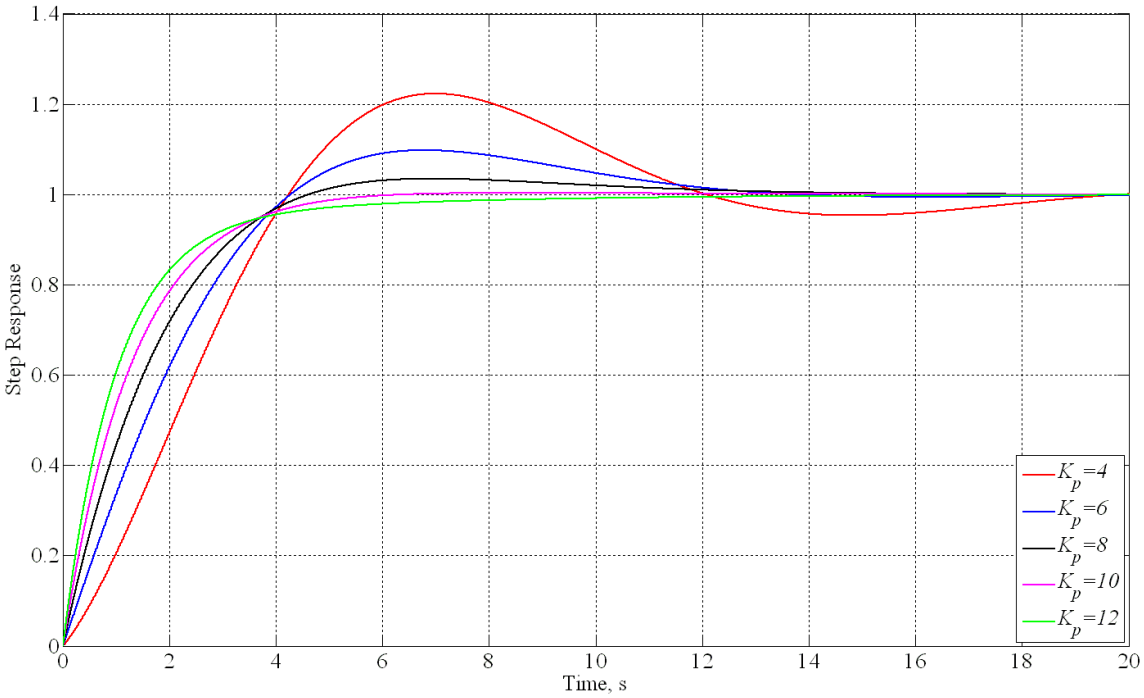


Figure 5. Step Responses with Varying K_p Gain, K_i=2, K_d=10, K=13

Figure 6 shows the Bode diagram for the open-loop system with varying K_p. It is interesting to note that all the curves cross -180 degrees at the same point, *frequency=0.445 rad/sec*. All of the magnitude curves crossover at about 100 rad/sec with 90 degrees of phase margin. This indicates that the two zeros of the system are below crossover, having contributed +180 degrees of phase shift before gain crossover. The plots do not respond to changing gain parameters as a classic two-pole dominated system would because of the two zeros below crossover. Therefore, it is necessary to simulate all gain selections to predict how the system will respond.

Only the plots of varying K_p are shown in the body of this report. The step response plots and bode diagrams of varying K_i and K_d are shown in Figure 10 through Figure 13 in the Appendix D. The varying terms of K_i and K_d result in similar Bode Diagrams. The phase margin is the same at 90 degrees and all of the magnitude plots show a gain crossover of about 100 rad/sec yet the step responses are all different. This indicates how strongly zeroes below crossover can affect the system.

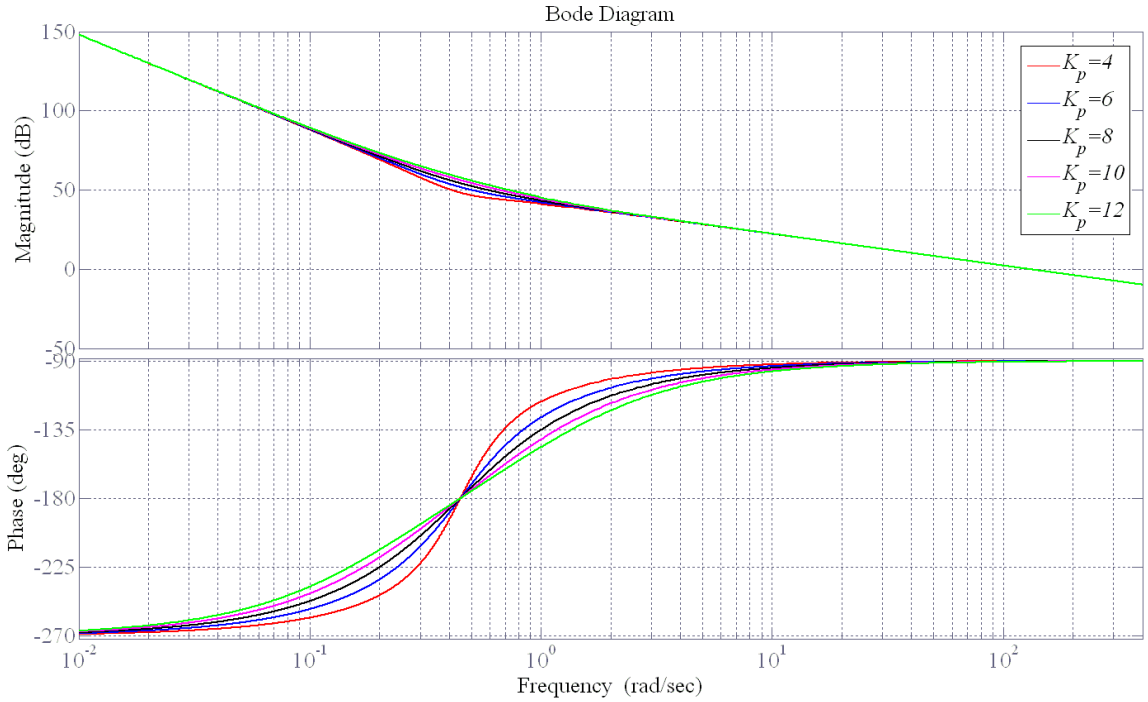


Figure 6. Bode Diagram with Varying K_p Gain, K_i=2, K_d=10, K=13, and W=2.5

For completeness, a PID controller with a real zero and pole was analyzed in addition to the ideal PID controller. The real zero and pole can be placed at an arbitrary nonzero frequency location with the use of a time constant term, τ. The real pole and zero locations have some benefits over the ideal PID controller. Real zeros have the ability to cancel out secondary poles in the system. If the assumption that the motor time constant is not smaller than the system's time constant is proved wrong, and the motor pole frequency location is closer to crossover than originally thought, the real zero could be used to cancel out the effects of the motor pole. Similarly, if the open loop gain is increased so that the motor pole is now below crossover, the real zero could be used to cancel out the motor pole. Finally, the real zero would allow the user to add positive phase to the system at a frequency range of their choice, with the purpose of counteracting the pure phase delay associated with the Ping sensor and Arduino microcontroller processing time. The same process was followed to find the range of gains that would stabilize the closed loop transfer function. The PID controller with nonzero frequency zero and pole locations is defined as the following

$$PID\ Controller \equiv D(s) = K_p + \frac{K_i}{\tau_i s + 1} + K_d(\tau_d s + 1)$$

where the gain terms are the same as the ideal controller, s is the Laplace frequency variable, and τ_i and $\tau_d$are the time constants of the pole and zero, respectively. The time constants are defined as

$$\tau = \frac{1}{2\pi f}$$

where f is the frequency location of the zero or pole in Hertz. Combining the controller terms with a common denominator, yields the following PID controller transfer function

$$D(s) = \frac{K_d(\tau_d s + 1)(\tau_i s + 1) + K_p(\tau_i s + 1) + K_i}{\tau_i s + 1}$$

The CLTF is found in the same manner as before, which results in the following transfer function

$$CLTF = T(s) = \frac{(KK_d\tau_d\tau_i)s^2 + (KK_d\tau_d + KK_d\tau_i + KK_p\tau_i)s + (KK_p + KK_i + KK_d)}{\tau_i s^3 + (KK_d\tau_d\tau_i + 1)s^2 + (KK_d\tau_d + KK_d\tau_i + KK_p\tau_i)s + (KK_p + KK_i + KK_d)}$$

By analyzing the characteristic equation of the CLTF and conducting a Routh-Hurwitz tabulation the stabilizing gain ranges were found to be

$$\tau_i > 0$$
$$KK_d\tau_d\tau_i > -1$$
$$K_p + K_i + K_d > 0$$
$$K({K_d}^2{\tau_d}^2\tau_i + {K_d}^2\tau_d{\tau_i}^2 + K_pK_d\tau_d{\tau_i}^2) - (K_d\tau_d - K_i\tau_i) > 0$$

A gain set of K=1, K_p=1, K_d=1, K_i=5 and two time constants set to τ_d=10, and τ_i=1 successfully stabilize the system. The closed-loop pole locations for that specific set of gains and times constants are -9.85 and -0.57±0.62i. The step response can be seen in Figure 7. The gains were chosen because they gave a step response with <10% overshoot and <5 second 2% settling time. The rise time is likely faster than it would be for the actual system because of the limited torque created by the motors.

The original intent of analyzing a PID controller with a real zero and pole was to make the response of the system more predictable. The issue with the ideal PID controller is that the location of the two zeros is hard to predict because the locations depend on all three terms: K_p, K_i, and K_d. By adding a zero and pole with nonzero frequency locations it was thought that the system could be tuned more accurately. After the previously discussed analysis, it was determined that the PID controller with real zero and pole did not provide any added benefit. This work was included in the report in the case that anyone working on the project in the future decides to try and implement this type of controller.

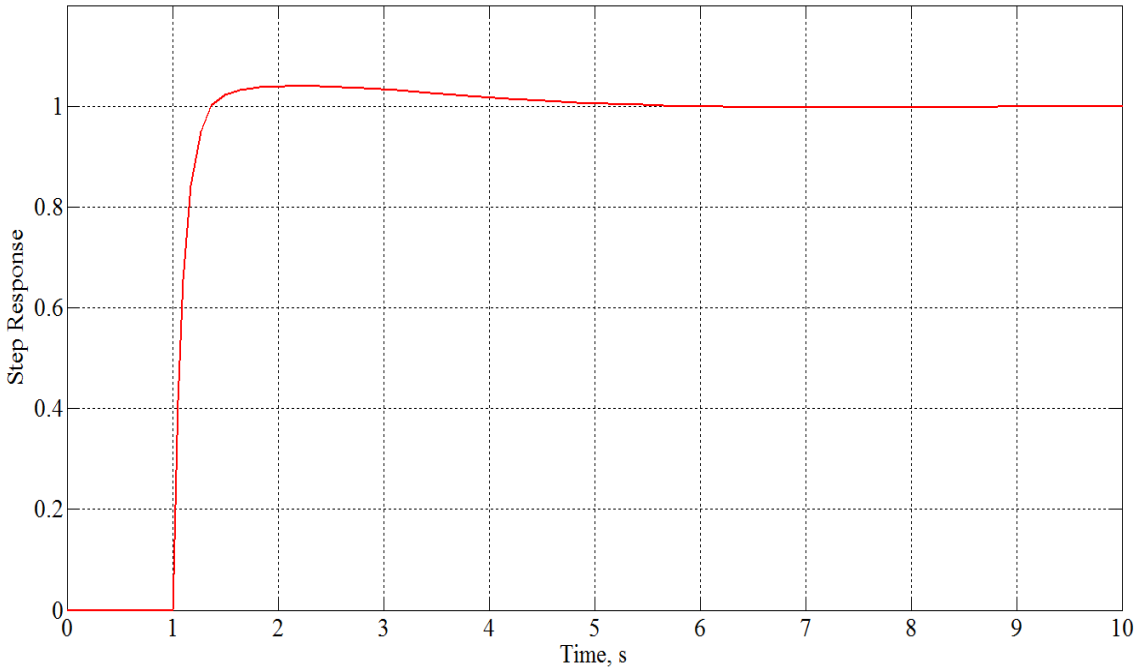


Figure 7. Closed-Loop Step Response of Model with PID Controller, real pole and zero

Arduino code was written to implement a discrete PID controller inside of the microcontroller. The function is meant to work inside the control loop, computing the PID output at every time step. The integrator and derivative terms are discrete because the system is, in actuality, discrete. The inputs to the function are error, K_p, K_i, K_d, τ_d, and τ_i. The controller terms are computed independently and summed together to produce the output of the PID controller. The proportional term is simply K_p multiplied by the error, but the integrator and differentiator terms are more complex. The Laplace transfer functions for an integrator and differentiator with arbitrary frequency locations are

$$Integrator \equiv \frac{1}{\tau_i s + 1}$$

$$Differentiator \equiv \tau_d s + 1$$

where τ_i and τ_d are the time constants of the differentiator and integrator. Using difference equations, the discrete integrator and differentiator terms were determined to be the following

$$I(k) = K_i \left[I(k-1) + \frac{[e(k) - I(k-1)]}{\tau_i} \right]$$

$$D(k) = K_d \left[[e(k) - e(k-1)] + D(k-1) - \frac{D(k-1)}{\tau_d} \right]$$

where *I* is the integrator term, and *D* is the differentiator term, *e* is the error signal, *k* is the current time step, and *k-1* is the previous time step.

The integrator and differentiator terms were tested with a step input to ensure the correct values were being calculated by the Arduino. Figure 8 and Figure 9 show the outputs of the integrator and differentiator implemented in the Arduino microcontroller. In both plots the time constants are equal to 100. In Figure 8, it can be seen that when the time step is equal to 100 the integrator value is 0.63 or $(1 - e^{-1})$, the equivalent to a continuous time plot of an integrator. Likewise in Figure 9, the output value is 0.37 or e^{-1} when the time step is equal to 100, producing the equivalent value to the continuous plot of a differentiator. The code that generated Figure 8 and Figure 9 can be found in Appendix E.

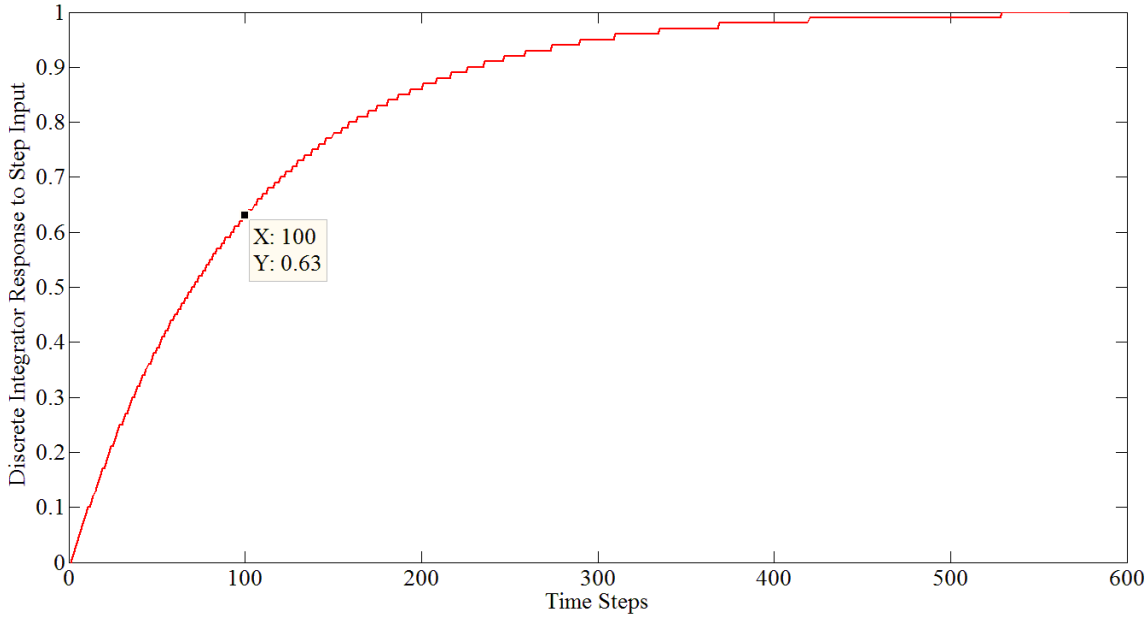


Figure 8. Step Response of Discrete Integrator, $\tau_i = 100$

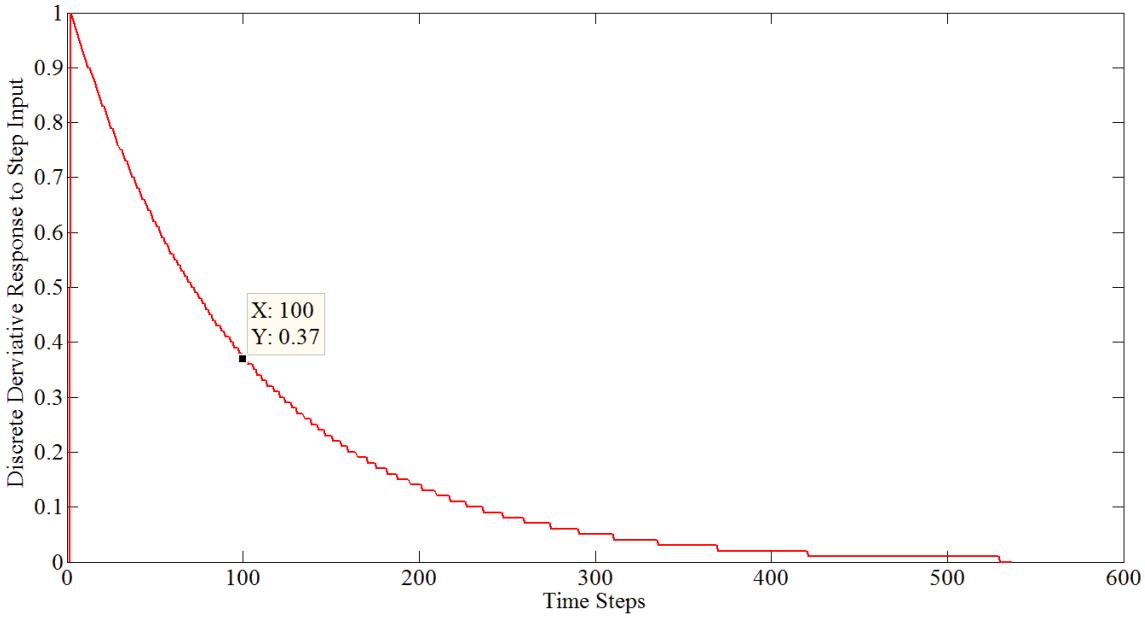


Figure 9. Step Response of Discrete Differentiator, $\tau_d = 100$

V. Future Improvements

There are several future improvements to the model and supporting components that will increase the performance and robustness of the experiment. The following section will call attention to these areas that need improvement and suggest alternatives to the present state of the model. This section could also serve as a starting point to anyone who will be continuing on with the project.

1. *Test Stand Assembly*

The base of the test stand is currently ½" plywood, which seemed to provide enough rigidity, until the steel cables were tensioned. The base of the test stand starts to bend as the cables are tensioned and this makes it hard to keep the test stand standing vertical. The corners are concave up, preventing the rubber feet from touching the ground. One solution to this would be to replace the current base with a piece of plywood that is ¾" thick. The down side is that the bolt holes would all have to be re-drilled. The original bolt pattern could be copied from the old base. Another solution to this would be to add metal straps on the bottom of the base, in an "X" shape. Two straps would run corner to corner, giving the base more bending stiffness. The problem with this solution is, if the straps cover any bolt holes, holes would have to be drilled through the metal straps.

Obviously, the test stand is fairly large. The ultimate goal of this project is to create a portable controls experiment for the students. The steel pole could be cut down, from 6' to a more appropriate height. This would scale down the whole project. The smaller pole would reduce the structure needed to support it, possibly eliminating many of the test stand assembly components.

2. *Motor Assembly*

The motor mounts are currently attached to motors via four small screws. Due to the vibrations caused by the spinning blades, the tiny screws can start to loosen while the model is flying. When this happens the motors are re-oriented by the screws pushing on the motors and aluminum mounting strip. The re-orientation of the thrust vectors can cause even more vibrations in the system, which will degrade the performance or even damage the blades. A simple solution to this problem is to make use of a product like Loctite. An adhesive in the tapped holes should be sufficient to prevent this problem in the future.

3. *Carriage Assembly*

The aluminum mounting strip that is part of the carriage is flat, when the propellers are spinning it vibrates and flexes visibly. It might be beneficial to find a mounting surface for the components that is stiffer than this aluminum strip. It cannot be steel because that would be too heavy. Aluminum C-Channel would add bending and torsional stiffness to the component which bares the thrust loads in flight and is subject to the most vibration. The C-

Channel would have to be wide enough to mount the motors and the linear bearings. The inner dimension of the C-Channel would need to be about 1 ½" wide.

The linear bearings are encased in steel, which was overlooked when the components were first purchased. The linear bearings are the heaviest components on the carriage assembly. Replacing the steal linear bearings with an aluminum cased set of bearings would reduce the weight and increase the performance of the system.

4. *Simulink Model and Control Loop*

The Simulink model could be expanded in a couple ways

- Account for the transformation between PID output and PWM signal into the ESCs
- Developing an equation to relate duty cycle/RPM to lift
- Consider both time constants of the motor, static and dynamic
- Model the discrete components of the system
- Account for delay of Arduino and Ping feedback

The lift of the motors could be estimated using the Integral Momentum Theorem applied to propellers as a theoretical analysis. A more accurate way to predict the total lift of the model would be to experimentally determine the lift at discrete duty cycles points or different rpm values. The motors should be swept through different duty cycles or rpm levels while measuring the lift on a test stand in one of the Aerospace labs. The data points should be plotted and a curve fitted to the data.

The control loop right now only considers one degree of freedom, vertical movement. The model actually has two degrees of freedom, vertical movement and rotation around the pole. Spinning around the pole is partially mitigated by the motors spinning in opposite directions. While the spinning is easy to prevent when controlling the model manually, it will be more complex when the model is flown closed loop. Therefore, the need for inner loop speed control of the motors arises. This could be accomplished through the use of two components, either separately or in tandem. The first component is a single axis gyro to measure rotational speed around the pole. The second component is a tachometer for measuring motor RPM. Both components could be used to stabilize motor speed and prevent rotation around the pole, but the tachometer would have the added value of aiding in the calculation of the lift being produced. The down side to the tachometer is that it might prove to be heavy and take up too much space on a system which is sensitive to both weight and volume.

There are several additions to the Arduino code that need to be implemented before the system can be tested closed loop. A kill switch should be added to the Arduino setup for safety. In case a test run needs to be aborted for any reason, the kill switch would stop the signals going to the motors. In addition, another switch could be added that does not instantly cut the signals going to the motors, but sends the code into a separate loop which slowly ramps down the motors. Next, the differentiator term output should be filtered because it is sensitive to process noise. A single input Kalman filter or weighted average calculation would be sufficient. Finally, to make it easier on the students, it should be possible for them to run a MATLAB code while the Arduino is active that sends the gain parameters they calculated through the serial link before the test run begins. This is not critical to running the system closed loop, but would be a nice feature, keeping the students from changing the Arduino code.

A big step is to eventually close the loop on the model to compare the actual system performance to the output of the Simulink model. Currently, the model does not take into consideration either of the motor time constants. The assumption that the dynamic motor time constant is small when compared to the overall system time constant is most likely true, but that assumption may not be true for the static time constant. That means that if the model is run from a complete stop the system output may not resemble the model output very closely. A way to get around this problem would be to command the model to a very small height to get the motors spinning, then change the input to the desired height. The comparison between the Simulink model and the actual system would start from the hovering state and continue to the final height.

VI. Conclusion

This project is the first iteration of a model helicopter control experiment for the AERO 320 class. The original goal of the project was to design, build, test, and write a complete lab experiment. Because of setbacks in the project, the original goal was not met. This project did successfully fly a first iteration of the physical model. In addition to flying the model, a test stand and carriage were built to support the model. A theoretical Simulink model was developed to predict the effect of PID controller gains on the performance of the system. An ideal PID controller was analyzed, along with a PID controller with real zero and pole locations. The PID controller with the real zero and pole was determined to not have any additional benefits to the normal PID controller.

An Arduino microcontroller was used to implement a Ping Ultrasonic Distance Sensor as a potential feedback sensor, test the control of motor speed in a loop, and develop a PID controller with discrete controller terms. The Arduino will provide the necessary signals to run the motors and control the speed separately through dual PWM outputs which compensate for the offset and gain mismatch in the ESCs and motors.

If the goal of the project remains the same, the Future Improvements section of this report outlines some areas to focus on to run the helicopter model closed loop.

VII. References

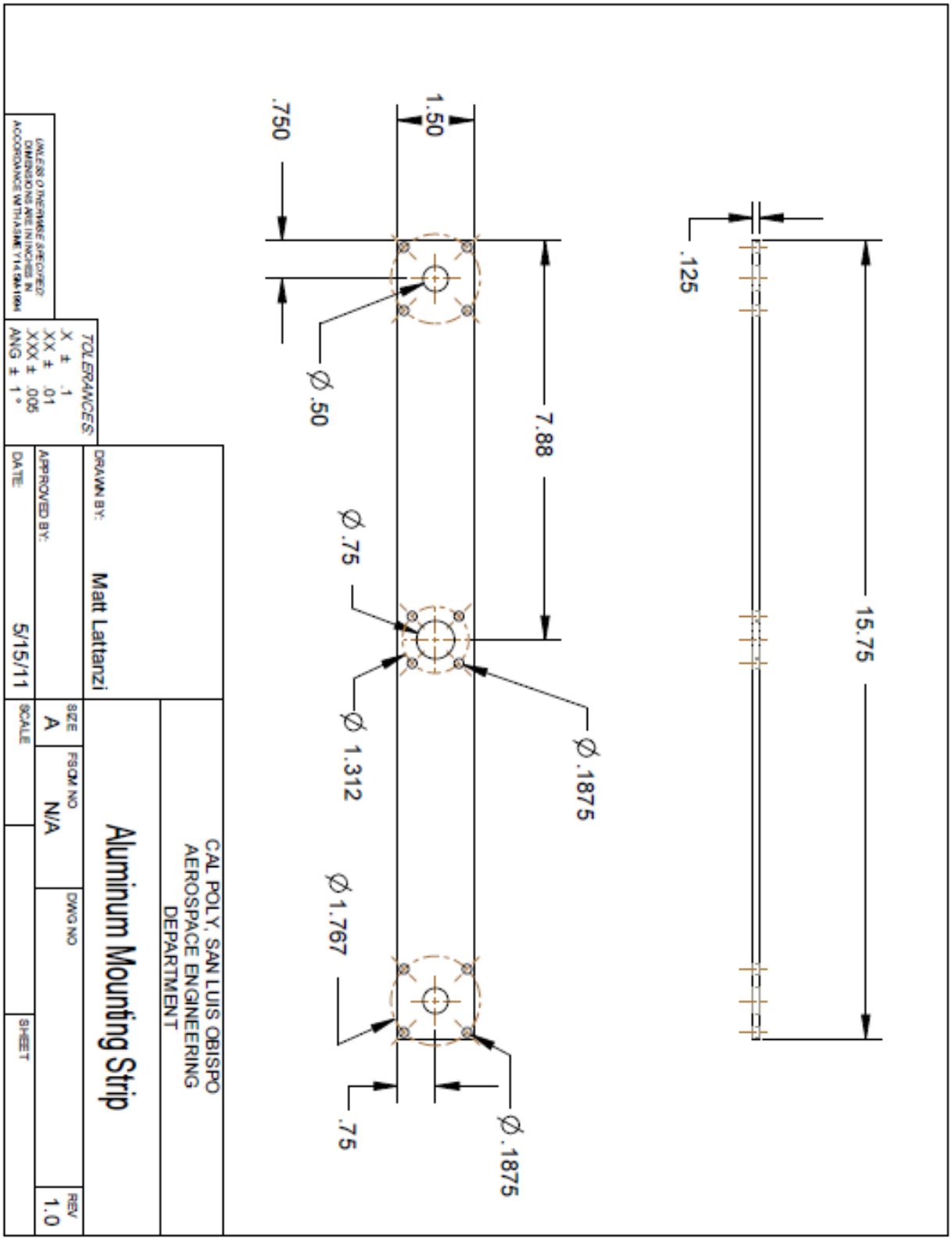
[1] R. S. M. Douglas Archer, An Introduction to Aerospace Propulsion, Upper Saddle River: Prentice-Hall, 1996.

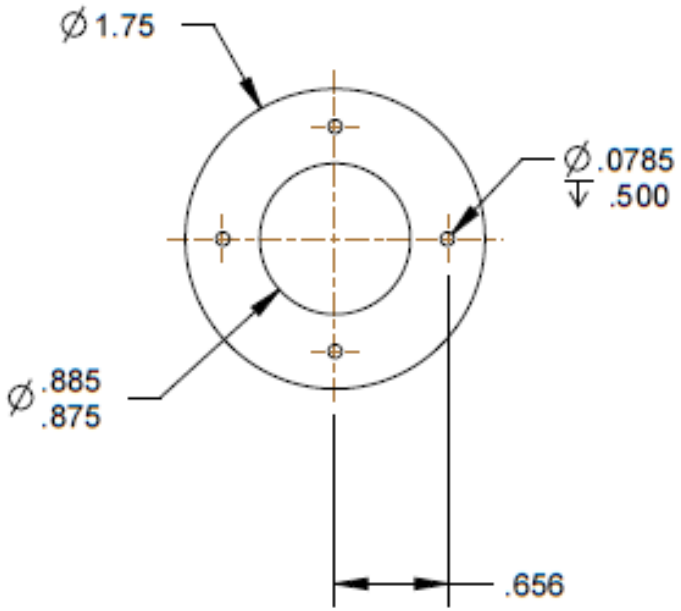
[2] G. D. P. J. E.-N. A. Frankilin, Feedback Control of Dynamic Systems, Upper Saddle River: Pearson Saddle River, 2010.

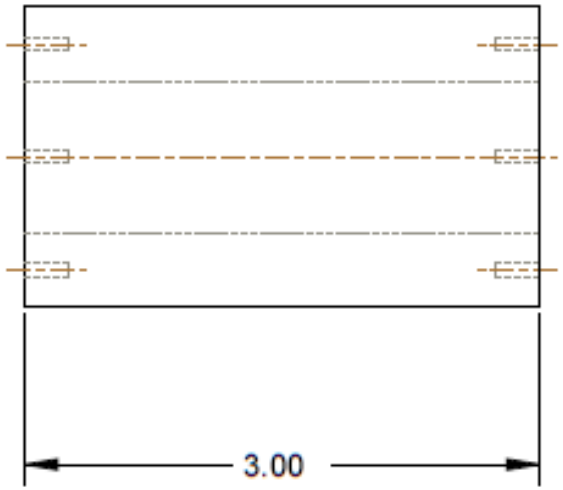
[3] F. K. B. C. Golnaraghi, Automatic Control Systems, Hoboken: John Wiley & Sons, Inc., 2010.

VIII. Appendix

Appendix A. Technical Drawings







CAL POLY, SAN LUIS OBISPO
AEROSPACE ENGINEERING
DEPARTMENT

Sleeve

DRAWN BY: Matt Lattanzi	SIZE A	FSCM NO N/A	DWG NO	REV 1.0
APPROVED BY:				
DATE: 5/15/11	SCALE		SHEET	

TOLERANCES:
X ± .1
.XX ± .01
.XXX ± .005
ANG ± 1 °

UNLESS OTHERWISE SPECIFIED:
DIMENSIONS ARE IN INCHES IN
ACCORDANCE WITH ASME Y14.5M-1994

Appendix B. Oscilloscope Measurements and Motor/ESC Start Times

Table 5. Oscilloscope Measurements of RC Receiver Signal Output

	Transmitter State			
	Throttle Low, Trim Low	Throttle Low, Trim High	Throttle High, Trim Low	Throttle High, Trim High
Pk-Pk (V)	3.36	3.36	3.36	3.34
Frequency (Hz)	45.245	45.237	45.245	45.232
PWM Period (ms)	22.102	22.106	22.102	22.108
On-time (ms)	1.2	1.4	2.0	2.2
Duty Cycle (%)	4.8	5.5	8.1	8.9

Table 6. Motor/ESC Initial Spin Duty Cycle Commanded Using Arduino

	ESC (Red/Blue/Black Motor Wires)	ESC (All Blue Motor Wires)
Start Duty Cycle (%)	5.3	5.7
Frequency (Hz)	45.25	45.25

Appendix C. Routh-Hurwitz Tabulation Calculations

The Hurwitz criterion states that for a given characteristic equation, all coefficients must be positive to ensure the roots of the equation lie in the left half of the s-plane. Routh's tabulation is a way to guarantee that those roots are in the left half of the s-plane by constraining the parameters that define the coefficients of the characteristic equation. Given a third-order characteristic equation

$$a_3 s^3 + a_2 s^2 + a_1 s^1 + a_0 = 0$$

the tabulation can be constructed using Table 7. The roots of the characteristic equation will all lie in the left half of the s-plane if the values in the 2nd column of the tabulation are all the same sign[3].

Table 7. Example Routh-Hurwitz Tabulation

s^3	a_3	a_1
s^2	a_2	a_0
s^1	$\frac{a_2 a_1 - a_3 a_0}{a_2}$	0
s^0	a_0	0

Ideal PID Controller

The closed loop transfer function for system with an ideal PID controller is

$$CLTF = T(s) = \frac{D(s)G(s)}{1 + D(s)G(s)} = \frac{KK_d s^2 + KK_p s + KK_i}{s^3 + K_d K s^2 + K_p K s + K_i K}$$

By analyzing the denominator of the CLTF, a Routh-Hurwitz tabulation can be constructed. The resulting table can be seen in Table 8.

Table 8. Routh-Hurwitz Tabulation for System with Ideal PID Controller

s^3	1	KK_p
s^2	KK_d	KK_i
s^1	$\frac{K^2 K_p K_d - KK_i}{KK_d}$	0
s^0	KK_i	0

All of the values in the 2nd column must be the same sign for the system to be stable. This results in the following constraints for the gains in the system:

$$K_p K_d K > K_i$$
$$KK_d > 0$$
$$KK_i > 0$$

PID Controller with Real Zero and Pole

The closed loop transfer function for the PID controller with zero and pole at nonzero frequency locations is the following

$$CLTF = \frac{(KK_d\tau_d\tau_i)s^2 + \left(KK_d\tau_d + KK_d\tau_i + KK_p\tau_i\right)s + (KK_p + KK_i + KK_d)}{\tau_i s^3 + (KK_d\tau_d\tau_i + 1)s^2 + \left(KK_d\tau_d + KK_d\tau_i + KK_p\tau_i\right)s + (KK_p + KK_i + KK_d)}$$

Again, using the coefficients of the denominator, a Routh-Hurwitz tabulation can be constructed and can be seen in Table 9.

Table 9. Routh-Hurwitz Tabulation for System with PID Controller with Real Zero and Pole

s^3	τ_i	$KK_d\tau_d + KK_d\tau_i + KK_p\tau_i$
s^2	$KK_d\tau_d\tau_i + 1$	$KK_p + KK_i + KK_d$
s^1	$\frac{K^2\left({K_d}^2{\tau_d}^2\tau_i + {K_d}^2\tau_d{\tau_i}^2 + K_p K_d\tau_d{\tau_i}^2\right) + K(K_d\tau_d - K_i\tau_i)}{K_d\tau_d\tau_i K + 1}$	0
s^0	$KK_p + KK_i + KK_d$	0

All of the values in the 2nd column must be the same sign for the system to be stable. This results in the following constraints for the gains in the system:

$$\tau_i > 0$$
$$KK_d\tau_d\tau_i > -1$$
$$K_p + K_i + K_d > 0$$
$$K\left(K_d{}^2\tau_d{}^2\tau_i + K_d{}^2\tau_d\tau_i{}^2 + K_pK_d\tau_d\tau_i{}^2\right) - (K_d\tau_d - K_i\tau_i) > 0$$

Appendix D. Step Response Plots and Bode Diagrams with Varying K_i and K_d Gains

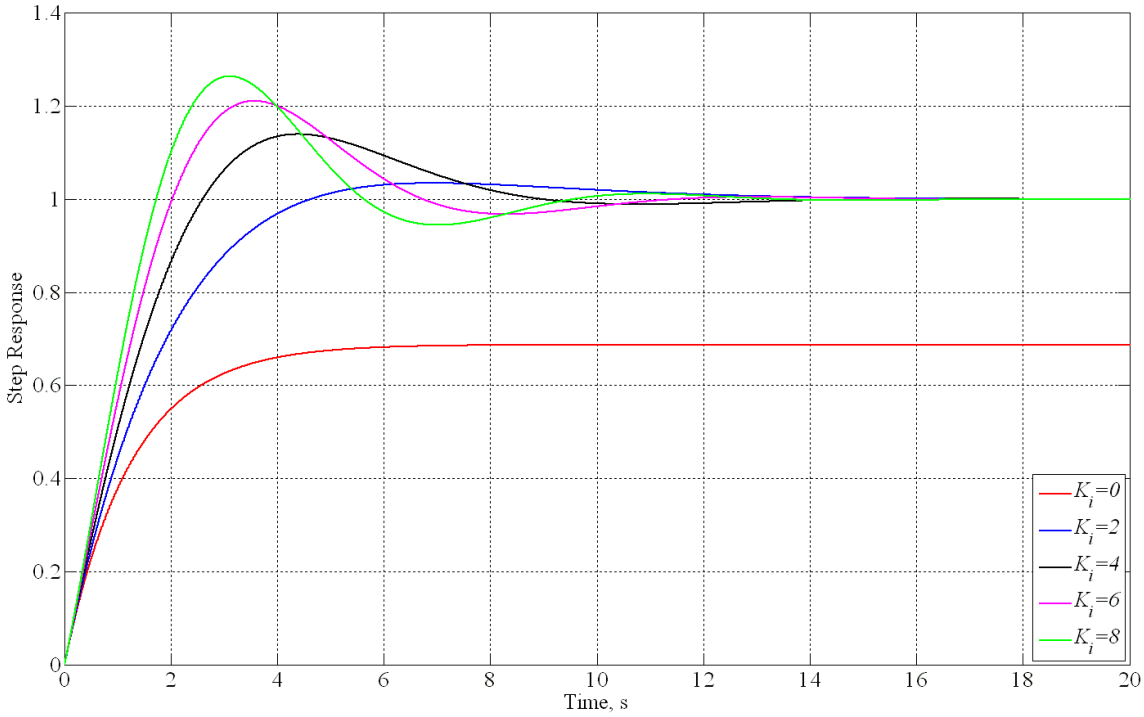


Figure 10. Step Response with Varying K_i Gain, K_p=8, K_d=10, K=13, and W=2.5

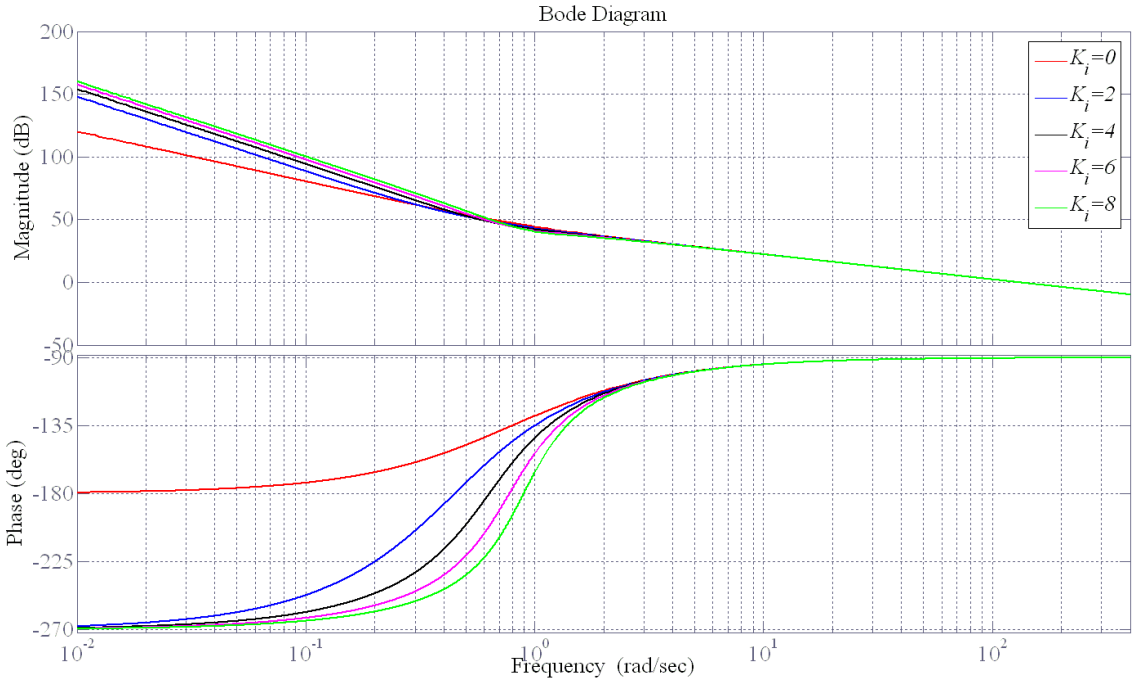


Figure 11. Bode Diagram with Varying K_i Gain, K_p=8, K_d=10, K=13, and W=2.5

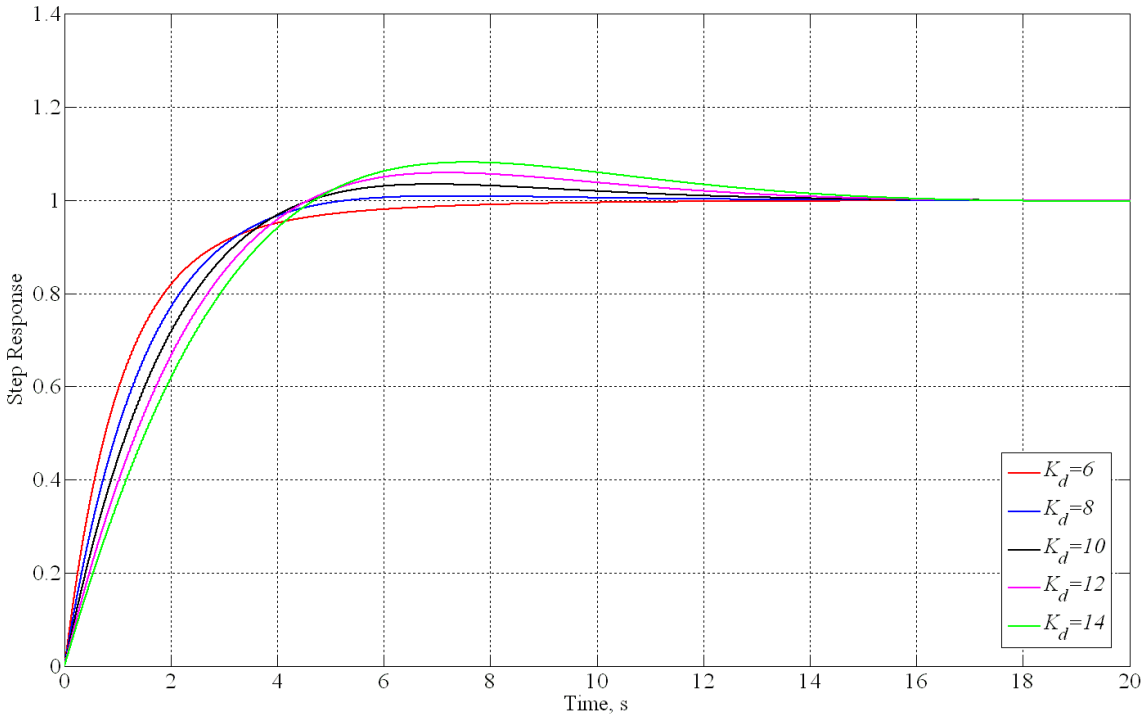


Figure 12. Step Response with Varying K_d Gain, K_p=8, K_i=2, K=13, and W=2.5

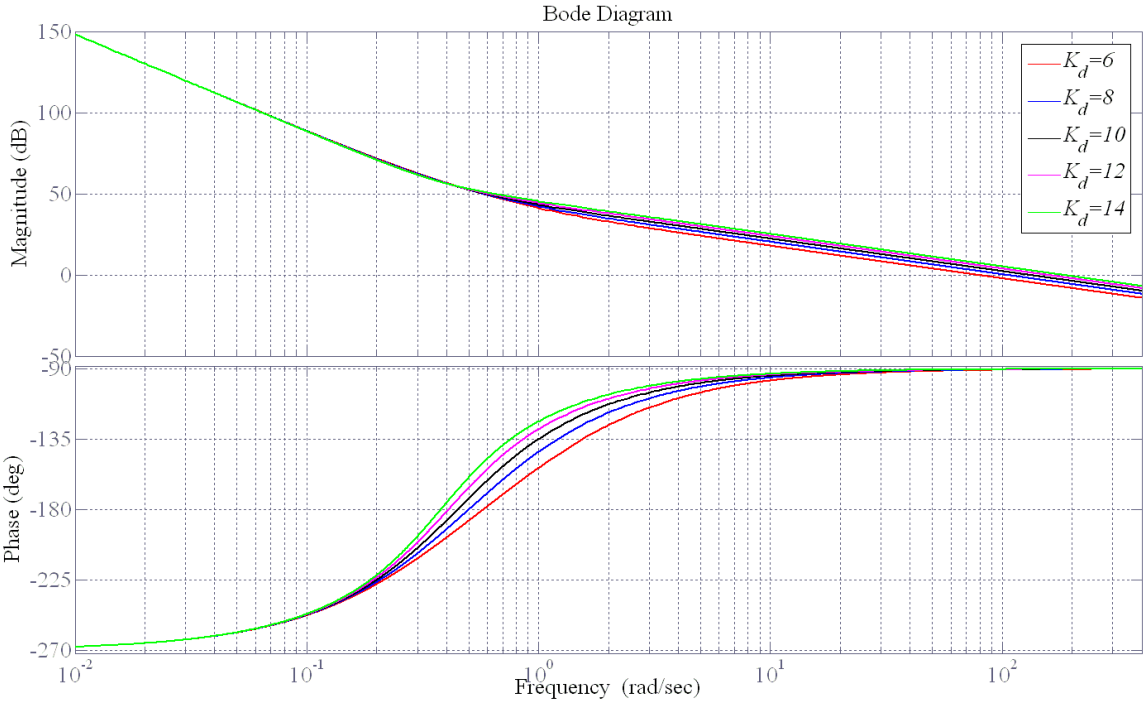


Figure 13. Bode Diagram with Varying K_d Gain, K_p=8, K_i=2, K=13, and W=2.5

Appendix E. Arduino Sample Code

PID Function

This code was developed to test the discrete PID controller. It is the code that generated in the body of the report. It could be implemented to into a control loop when the model is ready to be tested closed loop.

```
int x=1;
int y;

// Function Variables
double Pout;
double Dout;
double Iout;
double lastIout=0;
double lastDout=0;
double Freq_pole = .000159;  // Hz
double Freq_zero = .00159;  // Hz
double error;
double lasterror;
double PIDout;
double tau_i = 1/(2*3.14159*Freq_pole);
double tau_d = 1/(2*3.14159*Freq_zero);
double Kp =1;
double Ki =1 ;
double Kd = 1;

void setup(){
  Serial.begin(9600);
}

void loop(){
      for(x>0; x<1000; x++){
          if(x<25){
                    y=0;
                    out=PIDcomp(y,Kp,Ki,Kd,tau_d,tau_i);
                    Serial.println(out);
          }
          else{
                    y=1;
                    out=PIDcomp(y,Kp,Ki,Kd,tau_d,tau_i);
                    Serial.println(out);
          }
      }
}
double PIDcomp(double error, double Kp, double Ki, double Kd, double tau_d, double tau_i){
  Pout = Kp*error;
  Iout = (error - lastIout)/tau_i + lastIout;
  Iout = Ki*Iout;
  Dout = (error - lasterror) + lastDout - lastDout/tau_d ;
  Dout=Kd*Dout;
  lastDout = Dout;
  lastIout = Iout;
  lasterror = error;
  PIDout = Dout+Pout+Iout;
  return PIDout;
}
```