Assignment 7

Christian M. Schrader

AE 3713 – Introduction to Aerospace Controls Systems

# Table of Contents

[1 Table of Contents 1](#_Toc40537543)

[2 Physical Device & Testing 2](#_Toc40537544)

[2.1 Test Setup 2](#_Toc40537545)

[2.1.1 Sensor Test 3](#_Toc40537546)

[2.1.2 Logging Test 4](#_Toc40537547)

[2.1.3 ESC Test 4](#_Toc40537548)

[2.2 PID Test 5](#_Toc40537549)

[3 Dynamic Model 5](#_Toc40537550)

[3.1 Model Derivation 5](#_Toc40537551)

[3.2 Calculation of Constants 7](#_Toc40537552)

[4 Controller 9](#_Toc40537553)

[4.1 Linearization 9](#_Toc40537554)

[4.2 Controller Design 9](#_Toc40537555)

[4.3 Change in Angle 11](#_Toc40537556)

[5 Conclusion 12](#_Toc40537557)

[6 Appendix 12](#_Toc40537558)

[6.1 ESC Test Video 12](#_Toc40537559)

[6.2 Initial PID Controller 12](#_Toc40537560)

[6.3 Final PID Controller 13](#_Toc40537561)

[6.4 Code 13](#_Toc40537562)

# Physical Device & Testing

## Test Setup

The hardware experiment was assembled successfully from the instructions. Joints were made using wood glue and everything fit well. The arm rotates freely, and everything is well secured. Initially, the ESC did not function correctly, however, after trouble shooting it was determined to be an incorrectly soldered power jack. I was able to correct this by desouldering the incorrect wire and resouldering it to the correct lead. Additionally, there was also some play between the pin and the encoder which was removed by inserting a small shim. It was secured using a clamp.



Figure 1 Test setup

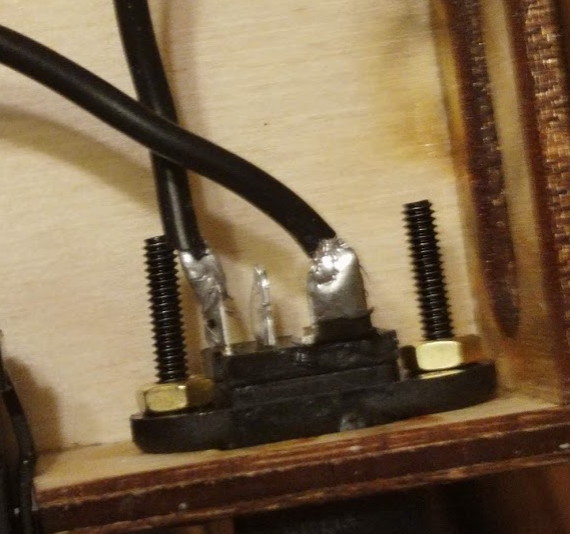


Figure 2 Wiring from factory (left) and corrected wiring (right)

All four of the test files were fun and demonstrated that the setup functioned as intended. A video of the PID and ESC test has been submitted with this report.

### Sensor Test

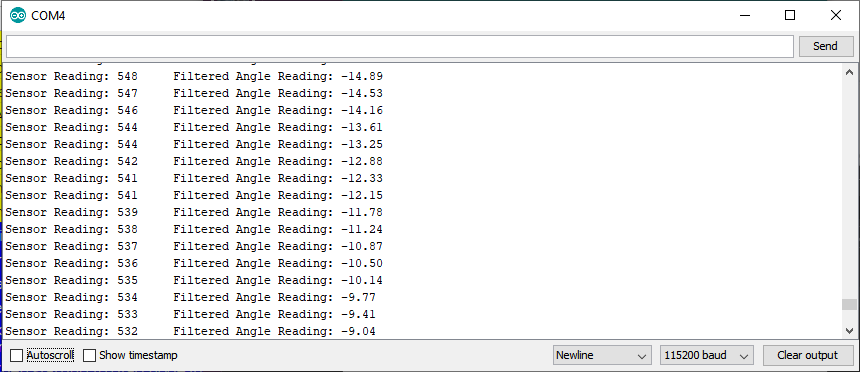


Figure 3 Sensor test raising output.

The screen shot of the sensor test was taken while I was raising the arm. The increasing angle shown was the expected result. As to be expected, there is some noise in the readings. To compete the sensor test, I held the arm approximately horizontally and received the expected output.

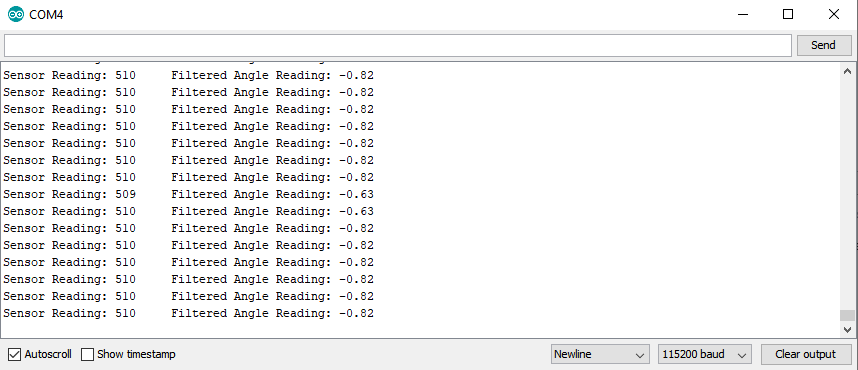


Figure 4 Sensor test horizontal output.

### Logging Test

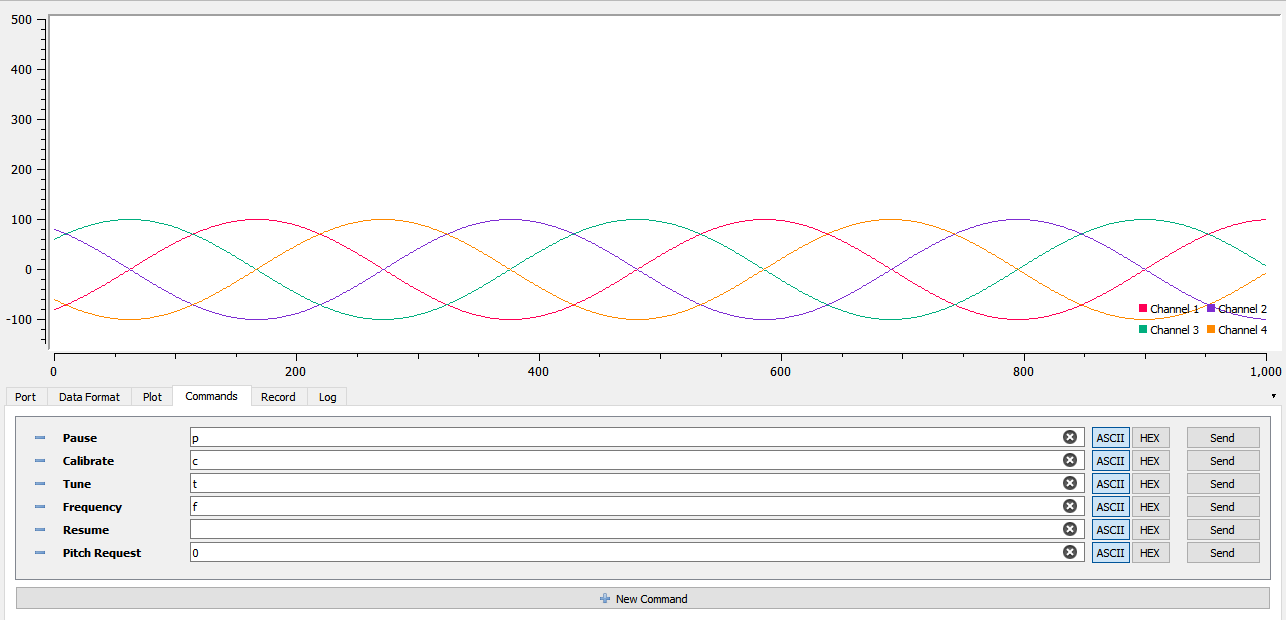


Figure 5 Logging test output.

As seen in the figure, the logging test displayed the expected 4 sinusoids when read using SerialPlot.

### ESC Test

The ESC test produced expected results. The ESC was successfully calibrated, causing the motor to beep in the calibration tones specified in the ESC datasheet. After that, varying throttle signals were sent. The ESC correctly read throttle commands causing it to produce higher thrust with higher control commands. A video of this test can be seen using the link in Appendix 6.1.

## PID Test

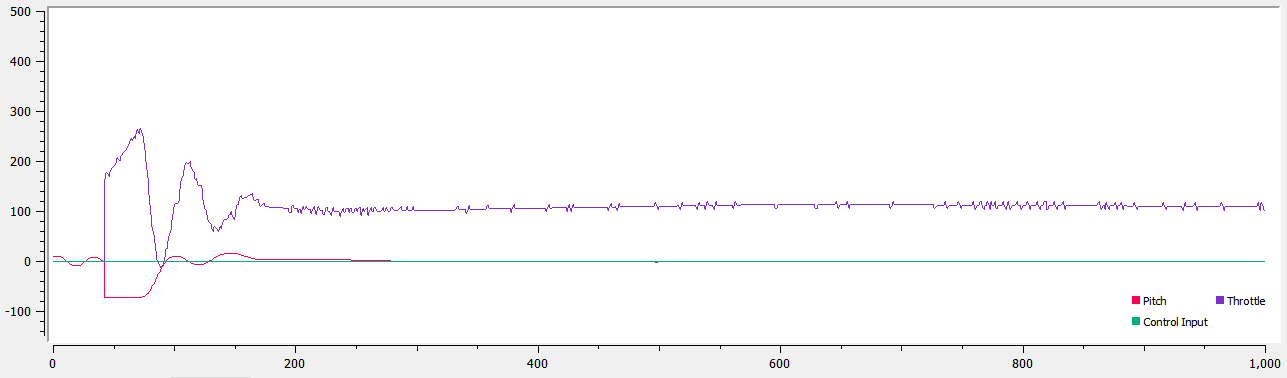


Figure 6 PID test output.

The PID test successfully demonstrated the system. When given the command to start, the motor spun up. As can be seen in the figure, the controller is not optimal. It overshoots and takes a bit of time to oscillate before settling. It did settle eventually. It does not have any steady state error. The default gains were the following.

Unfortunately, the test shown in Figure 6 was not recorded on video. While I was going to record another test, an unexpected problem occurred with the controller. The controller no longer communicates over serial nor does it execute its programming. Because a functioning replacement could not be procured in time, later tests were run using Kevin Schultz’s system.

# Dynamic Model

## Model Derivation

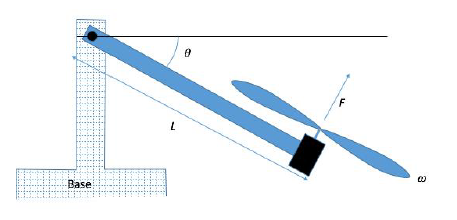


Figure 7 T-RECS diagram.

To design a more effective controller, we begin with modeling the device. To start, we will first define two state space variables using the angle of the arm, , and its derivative, .

To find the derivative of the state space, we must consider all moments that act on the arm. First, the equation for the thrust is given. This force acts on the end of the arm, a length away from the pivot.

is proportional to control input , therefore the equation can be further expanded.

Next, gravity acts on the system. Treating the system as a pendulum with a massless rod, we can find the component of gravity that acts perpendicularly to the arm. This force acts on the tip of the arm.

Finally, we model rotational friction. This acts at the pivot where the bearings provide a small, but not negligible, amount of friction. It is proportional to the angular speed by the coefficient of angular friction and acts at a length L.

The sum of all moments on the arm about the pivot is then given as follows.

We can then use newtons second law to find the second derivative of .

The model can be simplified by defining three constants.

Equation 1

With this, we can write the full state space model as follows.

Equation 2

The system also has two constraints. is limited by the maximum signal that can be sent to the ESC while the bounds on were determined experimentally using the sensor test code to measure the minimum and maximum angles that the arm can be held at.

## Calculation of Constants

To begin solving for the three constants, a drop test was conducted. The Sensor Test code was slightly modified to print out the time using the function and the filtered angle. The arm was held at before being released. The data was copy pasted from the serial monitor into Google Sheets and trimmed to the relevant window of time where the arm is in motion.

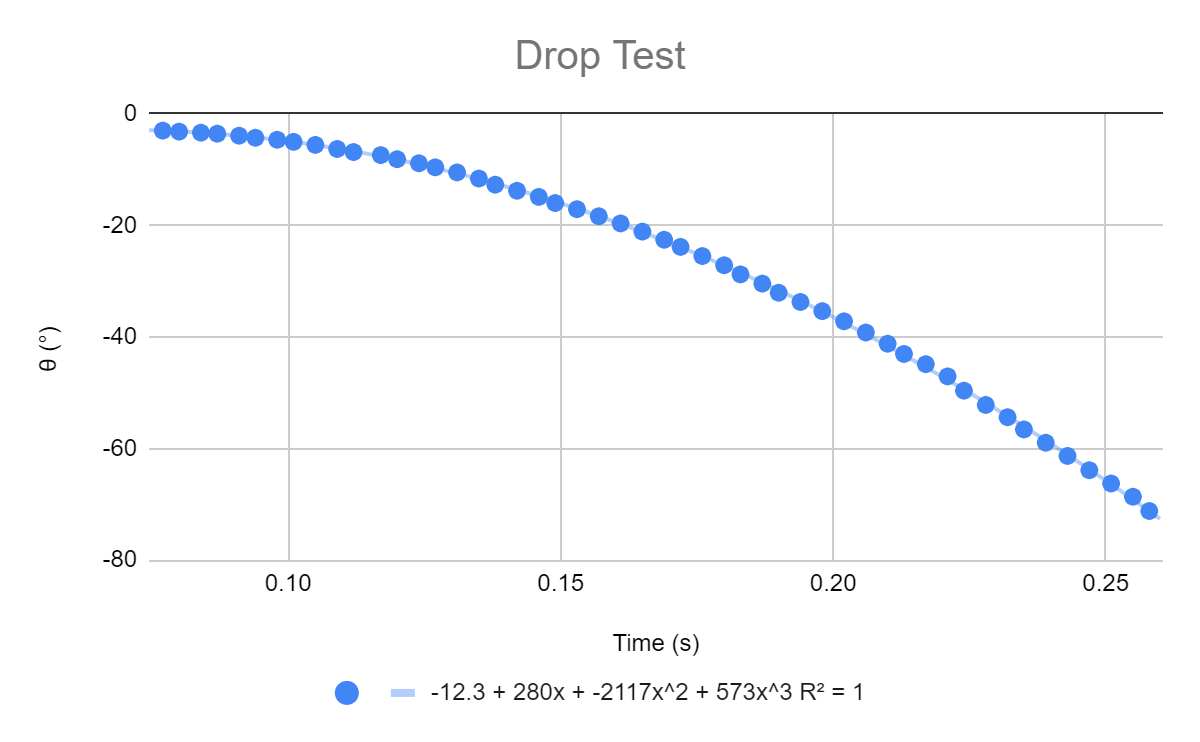


Figure 8 Drop Test Graph

As can be seen in Figure 8, a cubic regression was fit to the data with an value of .9985. This equation gives us as a function of time. We can use this to find a regression for .

Equation 3

During the drop test, the motor is off therefore . We can then equate Equation 1 to Equation 3.

To calculate , the arm was held at a series of equilibrium points. This was done by setting a specific value for and observing the at which it stabilized at. Under equilibrium, becomes zero which causes Equation 1 to simplify allowing us to find an equation for int terms of , , and . This is then used to create a table of for each measurement point.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| 50 | -64.8 | 0.7211 |
| 55 | -61.5 | 0.6679 |
| 60 | -60.04 | 0.5873 |
| 65 | -59.1 | 0.5146 |
| 70 | -57.48 | 0.4645 |
| 75 | -53.83 | 0.4442 |
| 80 | -48.34 | 0.4397 |
| 85 | -44.32 | 0.4193 |
| 90 | -41.03 | 0.3943 |
| 95 | -31.53 | 0.3999 |
| 100 | -27.14 | 0.3768 |
| 105 | -22.75 | 0.3542 |
| 110 | -1.55 | 0.3498 |

Table 3.1 Equilibrium test

The average of of all datapoints is .

Lacking any more efficient method, we can guess and check to find . This was done using ode45()in MATLAB to simulate the model over the period of the drop test. After some trial and error, a of .05 was decided on.

# Controller

## Linearization

To design the controller for the system, we first linearize the system about the desired equilibrium Using the default PID gains, the arm reaches this equilibrium at .

Equation 4

## Controller Design

We can now design a PID controller to stabilize the system. The arm will start at rest at the minimum angle so . The PID controller will take the following form.

The whole system can be modeled in Simulink using the as a simple block diagram.

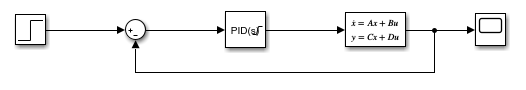


Figure 9 Closed Loop System

The state space block takes the and matricies from equation 4 in addition to and . We can then use the built in PID tuner to design a controller that works well for the system. Initially, the following PID values were selected.

Looking at the scope, these constants produced the following output.

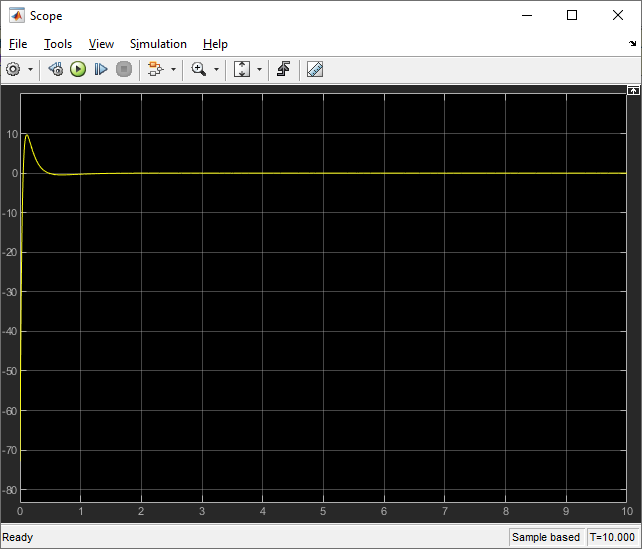


Figure 10 Initial scope output

These values were then run on Kevin Schult’s system. This can be seen in this video linked in Appendix 6.2. The system was observed to oscillate significantly more than expected by the simulation. To try and fix this, the was lowered to 1.2 leading to the final values shown below.

Looking at the scope, these constants produced the following output.

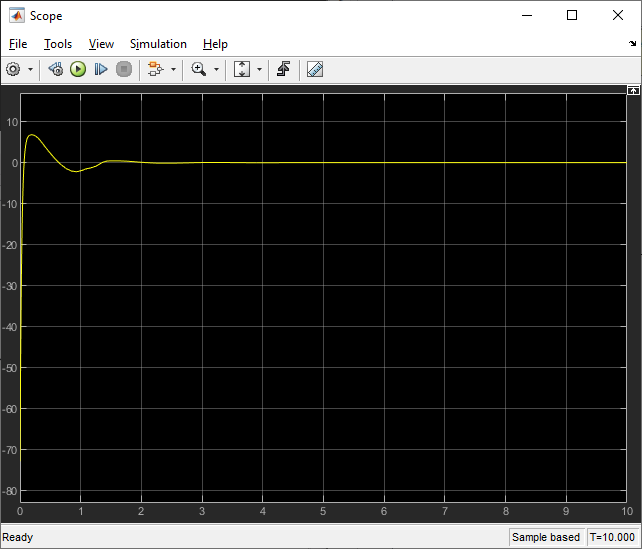


Figure 11 Final scope output.

These values were then run on Kevin Schultz’s system. This can be seen in the video linked in Appendix 6.3. This controller was much more effective than previous one as it didn’t oscillate nearly as bad. According to Simulink, the final PID controller has a rise time of .0445 seconds, a settle time of 1.17 seconds, and an overshoot of 9.04 %.

## Change in Angle

Finally, we can check the model’s response to a change in control input. By setting the step block to step from to and by setting the initial conditions of the system to , the system can then execute this change in input.

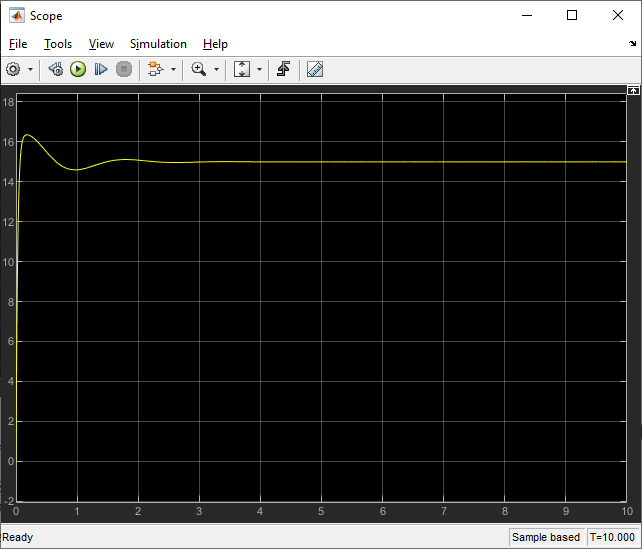


Figure 12 Scope response to input change.

As can be seen in the graph, the angle moves to the desired angle successfully with a reasonable settle time. Unfortunately, on the day I am finishing this last section, Kevin Schultz is away from home and rendering me unable to demonstrate this angle change experimentally.

# Conclusion

Using the model in Equation 2, a simulation was created to simulate the dynamics of the T-RECs system. Using a model of the system in Simulink and the MATLAB PID tuner, a PID controller was designed to hold the system at a desired angle.

As a last note, I would like to thank Kevin Schultz for lending me his time and his system. Without his help in running my PID controller, I would not have been able to complete the assignment.

# Appendix

## ESC Test Video

<https://youtu.be/gQ7j7VLx5As>

## Initial PID Controller

<https://youtu.be/WFtJUjFMEV0>

## Final PID Controller

<https://youtu.be/uL6JrxqeVlg>

## Code

What follows is the matlab script used to complete the project. Please note that some of the functions are not used in the final report but are included nonetheless.

%% Initialization

clc; clear; close; clf;

expr = readtable("droptest.csv");

% Initial Variables

dt = .01;

t0 = 0;

tf = 3;

time = t0:dt:tf;

init\_state = [-72.8; 0];

% State Space

A = [0 1

0 -.05];

B = [0; 105.6832];

C = [1 0];

D = 0;

%x = rk4(@state, init\_state, t0, tf, dt);

[tn,xn] = ode45(@nocontrol, [t0 max(expr.Var1)], [expr.Var2(1) 0]);

[tp,xp] = ode45(@pcontrol, [t0 tf], init\_state);

%% Graph

figure(1);

hold on;

plot(tn, xn(:,1), "b");

plot(expr.Var1, expr.Var2), "g";

%plot(tn, xn(:,2), "r");

title("Drop Test");

xlabel("Time (s)");

legend("\theta (ODE45) (deg)", "d\theta (Experimental) (deg)");

figure(2);

hold on;

plot(tp, xp(:,1), "b");

plot(tp, xp(:,2), "r");

title("P Controller");

xlabel("Time (s)");

legend("\theta (deg)", "d\theta (deg/s)");

function xdot = nocontrol(t,x)

xdot = model(x, 0);

end

function xdot = pcontrol(t,x)

Kp = 2.3;

xdot = model(x, -Kp\*x(1));

end

function xdot = model(x, u)

C1 = .4718;

C2 = 4234;

C3 = .05;

if (u > 1000)

u = 1000;

elseif (u < 0)

u = 0;

end

xdot = [x(2)

C1\*u^2-C2\*cosd(x(1))-C3\*x(2)

];

end