

Control Systems Lab 2: SeeSaw Balancer

Summary

For lab 2, we used methods we learned in class to design a system that balanced a seesaw. Such class material that we used included: system design by root locus, using second-order approximations by way of dominant poles, and block diagram reduction to find transfer functions. We also got hands-on experience using MATLAB for more than just homework checking and we gained some experience using Simulink.

The goal of the lab was to design the system so that the seesaw would be balanced by adjusting the arm position. To effectively do this, we needed signals that represented both the position and the angle of the seesaw. This relationship was given to us in the block diagram below. The only modification we needed to make was choosing the value for K.

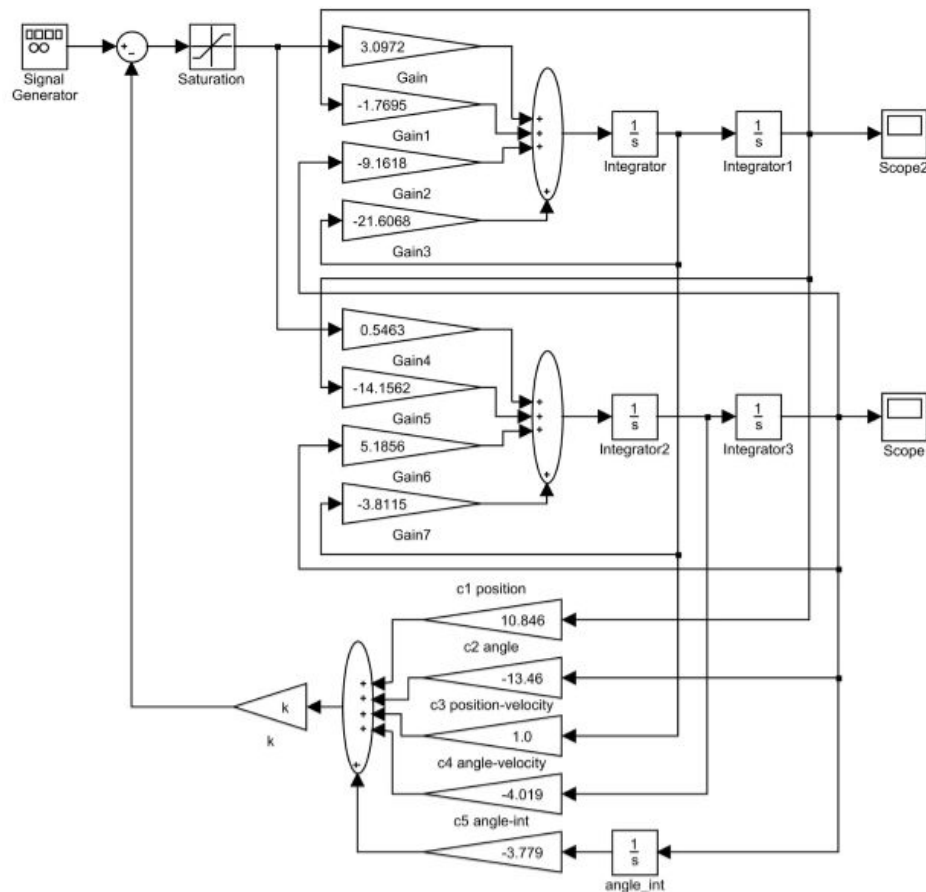


Figure 1: Closed loop block diagram

Pre-Lab Analysis

Verification of Transfer Functions

In order to understand the system, we needed to first understand the open loop system. As such, the first two pre-lab analysis instructions were to verify the open loop transfer functions G_{va} and G_{vp} .

Verify:

- (1) That the transfer function from the signal generator voltage input (modeling the output of the UPM amplifier) to the seesaw angle $\alpha(t)$ is

$$G_{va}(s) = \frac{0.5463s^2 - 0.001183s - 42.88}{s^4 + 21.61s^3 - 3.416s^2 - 147s - 138.9}$$

- (2) That the transfer function from the signal generator voltage input to the cart position $p(t)$ is

$$G_{vp}(s) = \frac{3.097s^2 - 21.07}{s^4 + 21.61s^3 - 3.416s^2 - 147s - 138.9}$$

Figure 2: Pre-lab analysis instructions.

As suggested in the assignment, I used MATLAB to verify the given open loop transfer function.

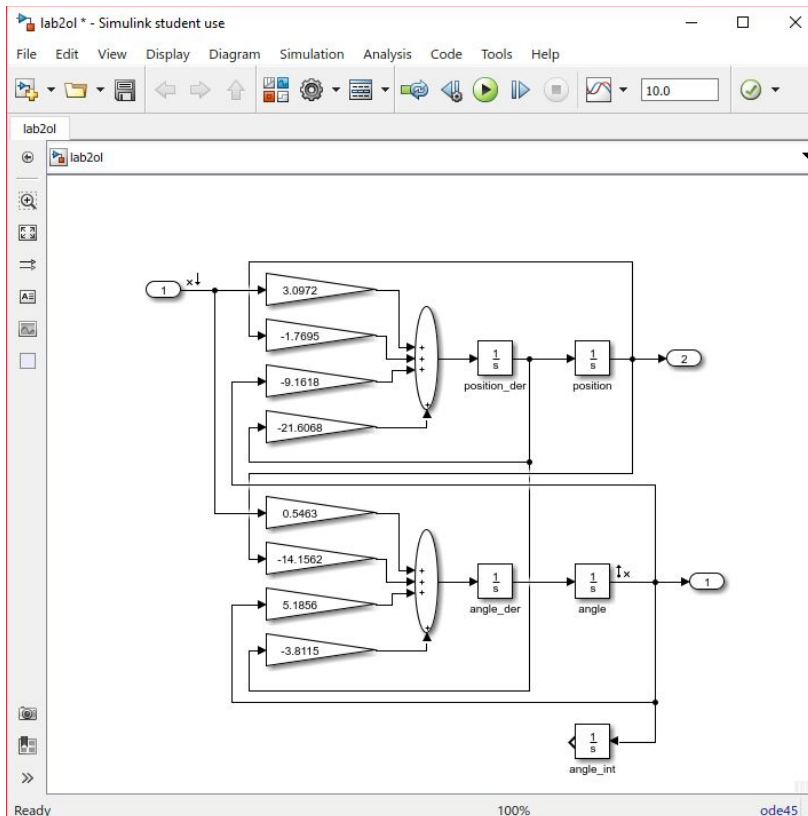


Figure 3: Provided Simulink file.

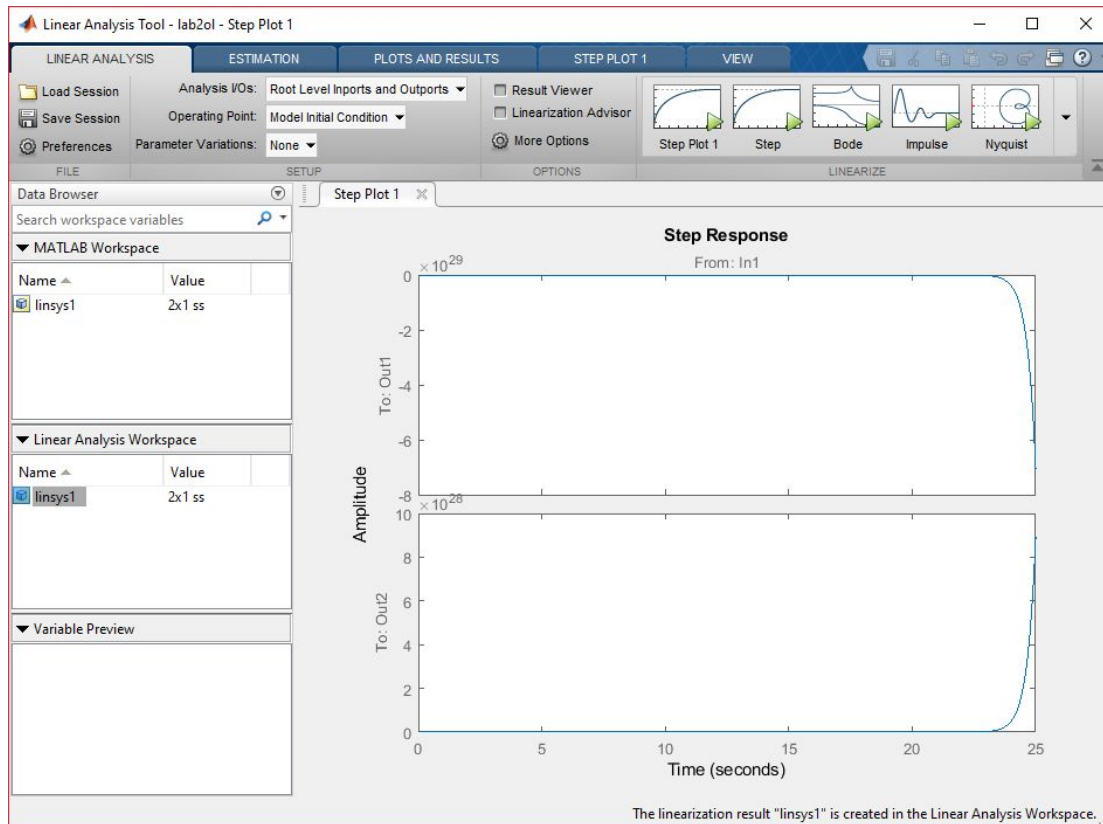


Figure 4: I ran the Linear Analysis Tool in Simulink, generated a Step response, then copied the provided transfer function into the MATLAB Workspace.

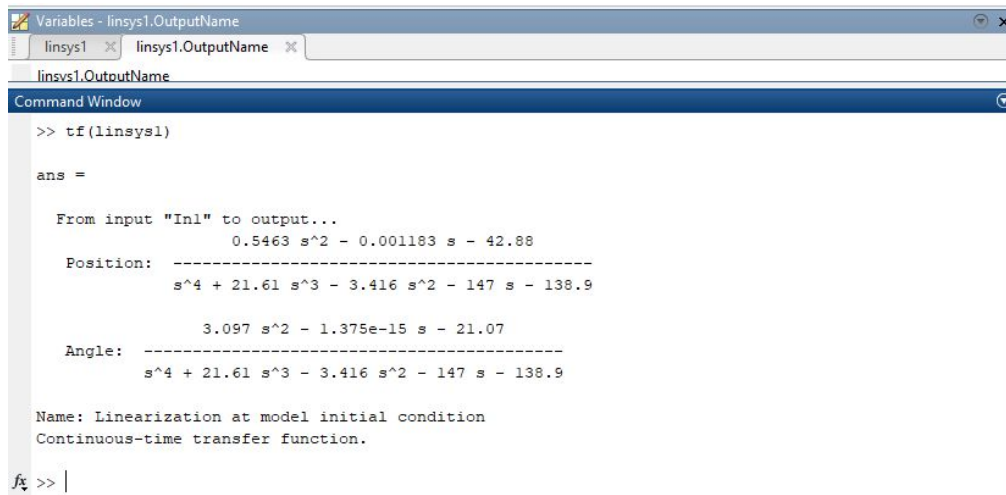


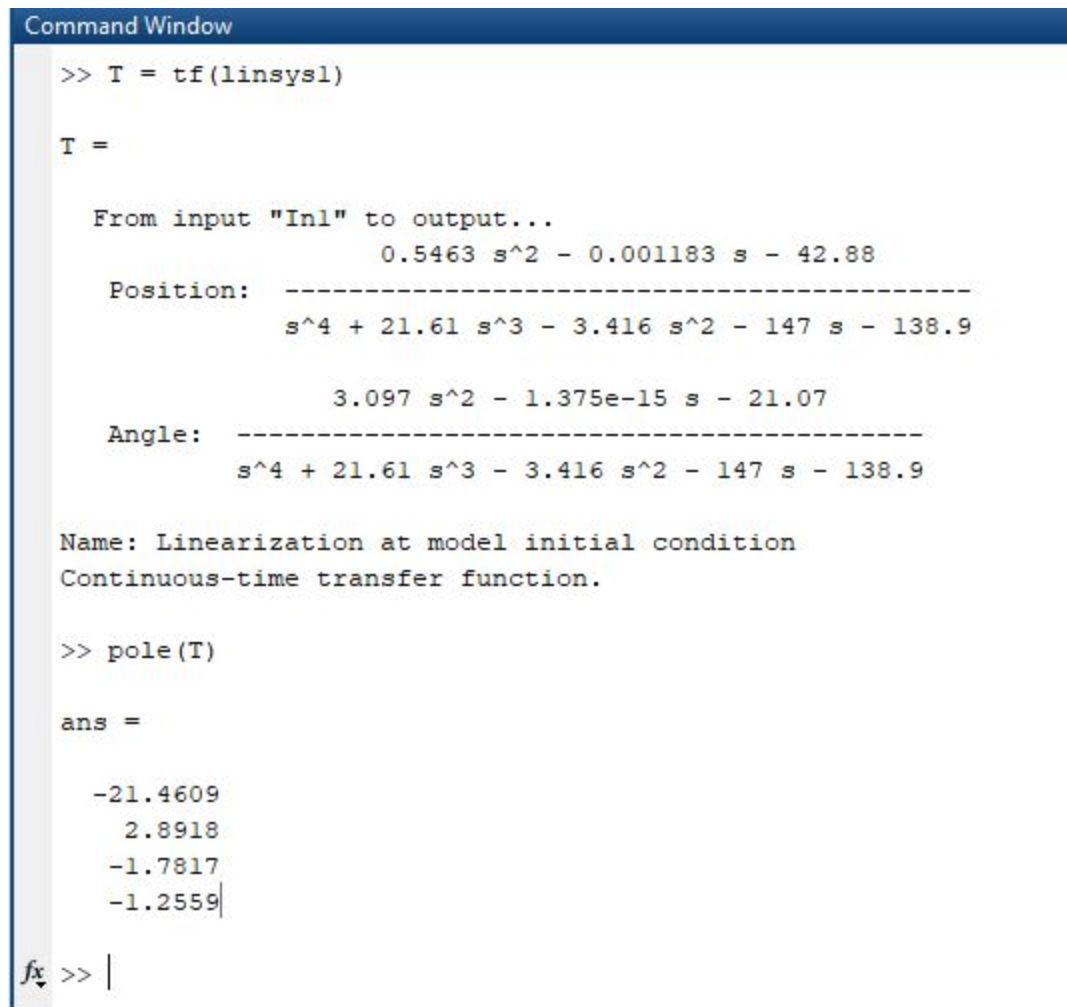
Figure 5: Display the generated transfer functions in the MATLAB Command Window. These are identical to the transfer functions $G_{vp}(s)$ and $G_{va}(s)$, respectively.

The next was to verify that the system's open loop poles were: -21.46, -1.78, -1.26, and 2.89.

(3) That the system's open loop poles are -21.46, -1.78, -1.26 and 2.89.

Figure 6: Lab guide instructions.

Using the same transfer function provided from the Simulink Linear Analysis Tool, I verified this in the MATLAB Command Window:



```

Command Window

>> T = tf(linsysl)

T =

From input "In1" to output...
          0.5463 s^2 - 0.001183 s - 42.88
Position:  -----
          s^4 + 21.61 s^3 - 3.416 s^2 - 147 s - 138.9

          3.097 s^2 - 1.375e-15 s - 21.07
Angle:    -----
          s^4 + 21.61 s^3 - 3.416 s^2 - 147 s - 138.9

Name: Linearization at model initial condition
Continuous-time transfer function.

>> pole(T)

ans =

-21.4609
  2.8918
 -1.7817
 -1.2559
  
```

Figure 7: Open loop pole verification.

I worked through verifying the closed-loop transfer functions by hand using block-diagram reduction. The following photos outline the process:

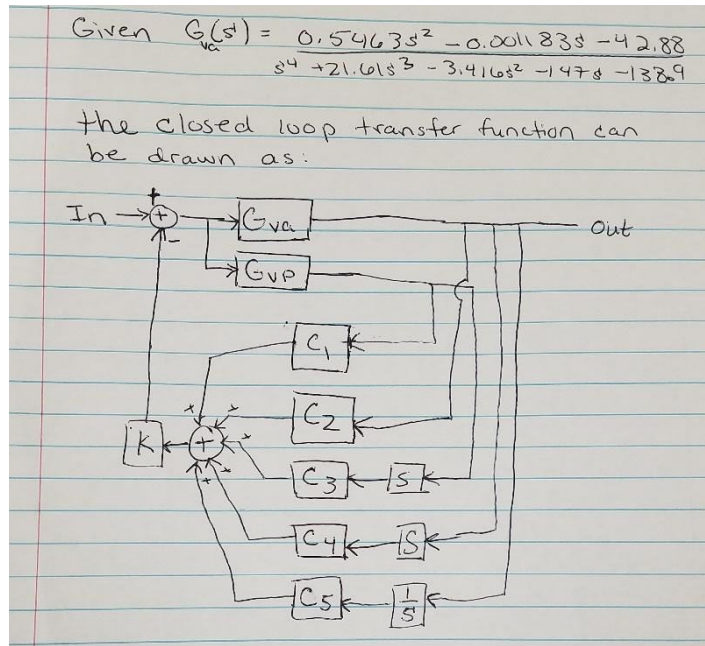


Figure 8: Block diagram reduction, step 1.

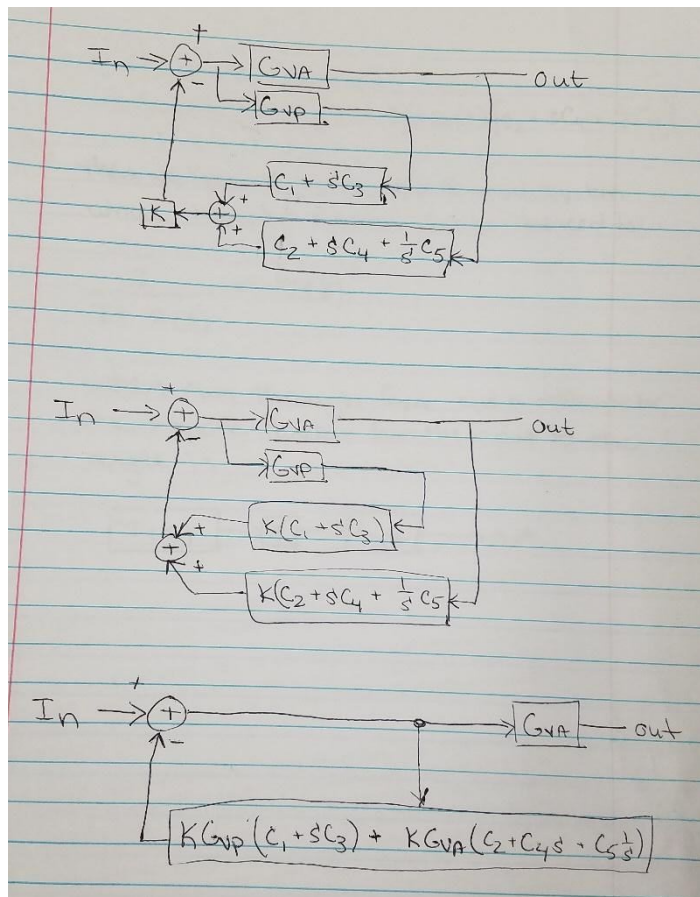


Figure 9: Block diagram reduction, step 2.

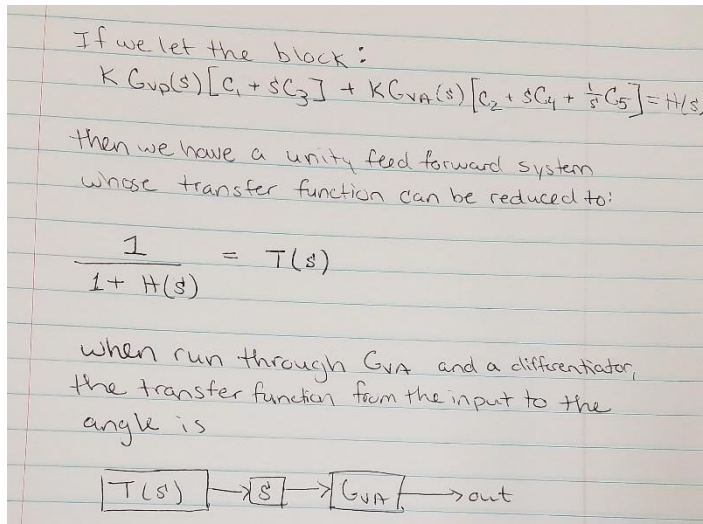


Figure 10: Block diagram reduction, step 3.

Using [Symbolab](#) to verify, the transfer function from Input to G_{va} was confirmed to be:

$$\frac{s \left(\frac{(0.5463s^2 - 0.001183s - 42.88)}{(s^4 + 21.61s^3 - 3.416s^2 - 147s - 138.9)} \right)}{1 + \frac{k(0.5463s^2 - 0.001183s - 42.88)(-13.46 - 4.019s - 3.779 \cdot \frac{1}{s}) + k(3.097s^2 - 21.07)(10.846 + s)}{s^4 + 21.61s^3 - 3.416s^2 - 147s - 138.9}} =$$

$$\frac{s(0.5463s^2 - 0.001183s - 42.88)}{21.61s^3 - 3.416s^2 - 138.9 + 0.9014203ks^3 + 26.24162...ks^2 + 149.21618...ks + 348.64405...k + \frac{s^5 - 147s^2 + 162.04352k}{s}}$$

via Symbolab

Figure 11: Symbolab confirmation of closed loop transfer function.

This is consistent with the transfer function given to us in the lab assignment.

Determination of Gain (K)

We used the root locus method for determining K. To make things easier, Nicholas and I wrote a small MATLAB script to help us graphically determine K.

```
zeta = 0.78; %or more, up to 1
s = tf('s');
D = s^5 + 21.608*s^4 - 3.4161*s^3 - 146.9644*s^2 - 138.8722*s;
N = 0.9012*s^4 + 26.2525*s^3 + 149.2205*s^2 + 348.5348*s + 162.0442;
RLSystem = N/D;
figure(1)
rlocus(RLSystem)
sgrid(zeta,0)
[K,poles]=rlocfind(RLSystem);
K
```

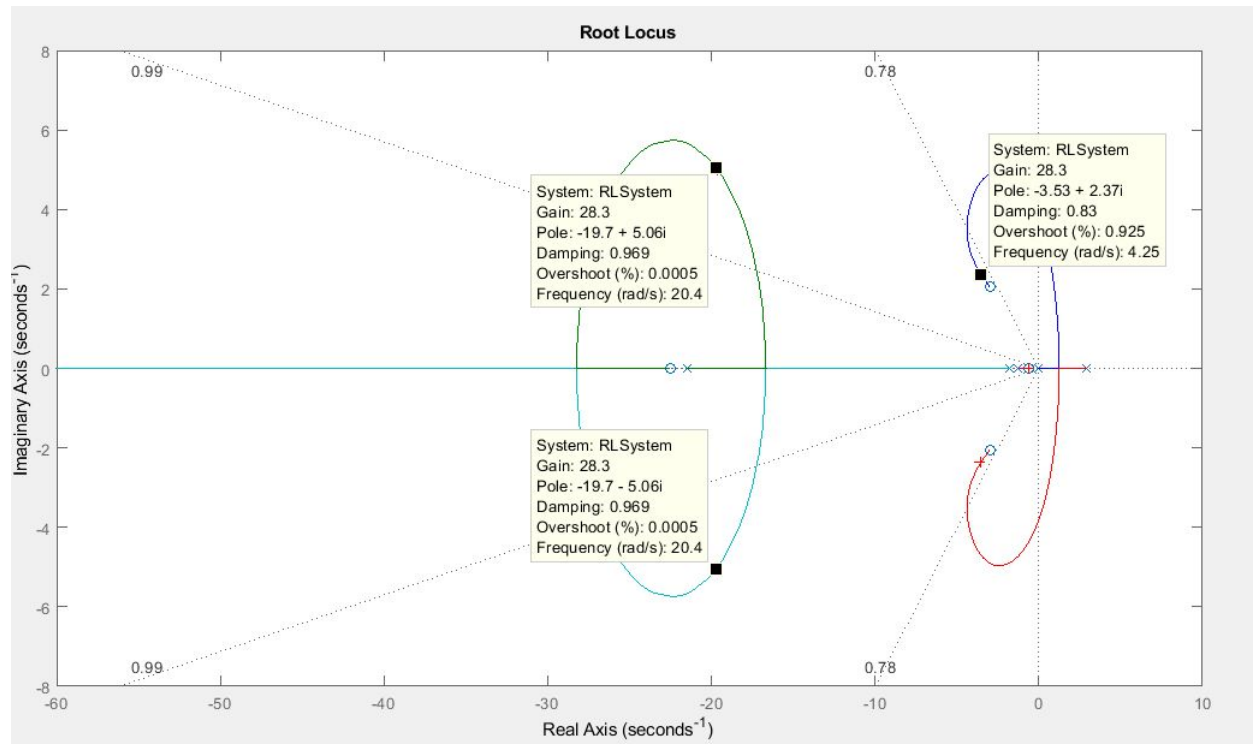


Figure 12: Root Locus of the transfer function with our chosen K value (28.3).

The key to this lab is to assume a second-order approximation holds. In order to justify this assumption, the real part of the dominant pole needs to be 5 times larger (for a weak second-order approximation). Ideally, we would prefer to the real part of the dominant pole be 10 times larger, however none of us in the group were unable to find a gain value to satisfy this. We noticed that as we crept closer to such a gain value, the real pole close to the origin was eliminated, thus ruining our second-order approximation all together. With a gain of 28.3, we managed to get this weak second-order approximation satisfied.

$$\frac{R\{-19.7 \pm 5.06j\}}{R\{-3.53 \pm 2.37j\}} = \frac{-19.7}{-3.53} = 5.6$$

It is worth noting here that our partner, Colton, came to the lab with a different gain value calculated. His K value was 16. For this case:

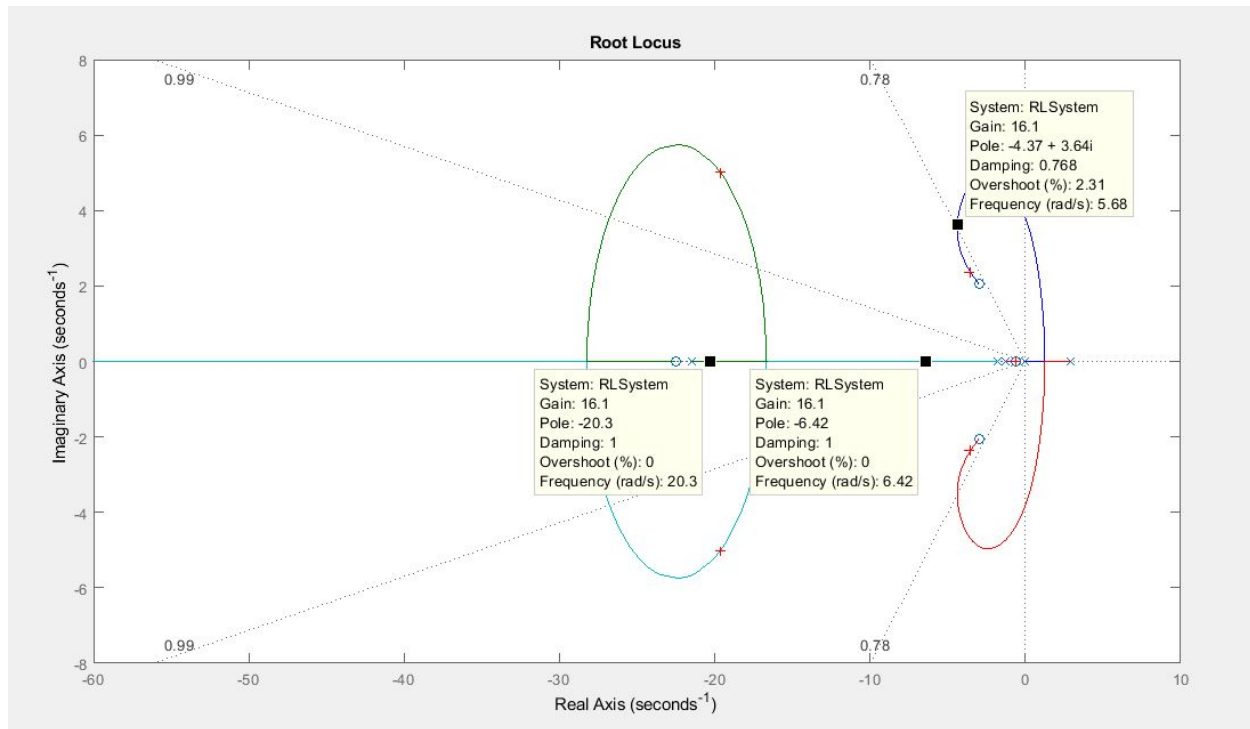


Figure 13: Root Locus of the transfer function with Colton's chosen K value (16.1).

For this K value, the second-order approximation due to dominant pole doesn't hold:

$$\frac{R\{-6.42\}}{R\{-3.53 \pm 2.37j\}} = \frac{-6.42}{-3.53} = 1.81$$

Patrick said that this value would be acceptable to experiment with (i.e. this K value would not harm the lab equipment), though it would likely take much longer to reach settling and oscillate unexpectedly. As a group, we decided to run our lab with the K value of 28.3.

We had two additional constraints for our modified transfer function. The rise time was required to be less than 3 seconds and the settling time was required to be less than 7.5 seconds. With our gain value of 28.3, we found the following step response:

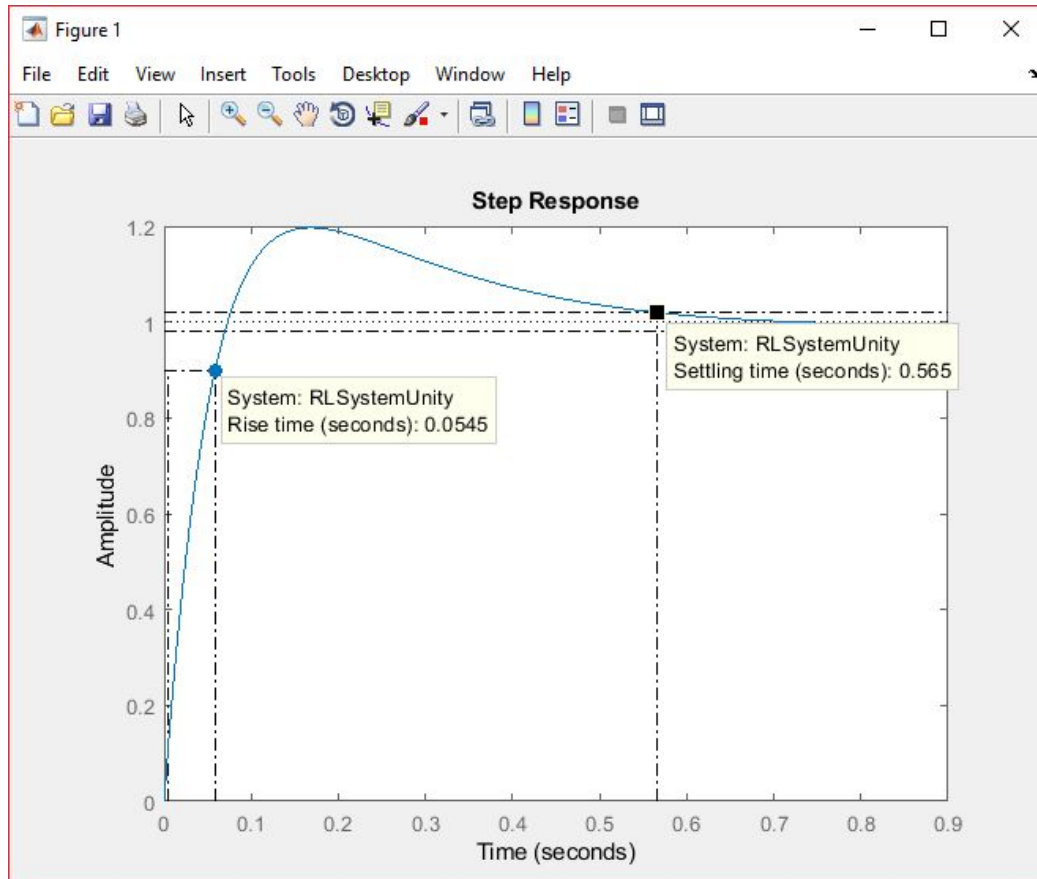


Figure 14: Step response of the transfer function with our chosen K value (28.3).

Simulation

To verify that our K value would be sufficient, we simulated it in Simulink by editing the provided file with our value.

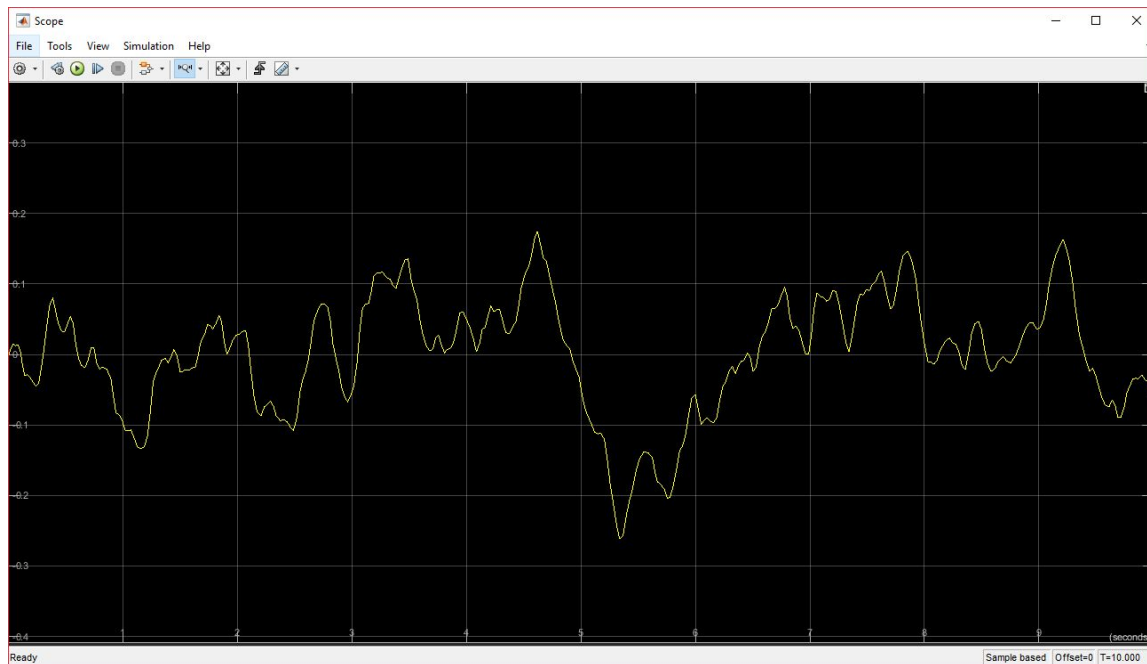


Figure 15: Simulink Scope view for “random” input vs. angle.

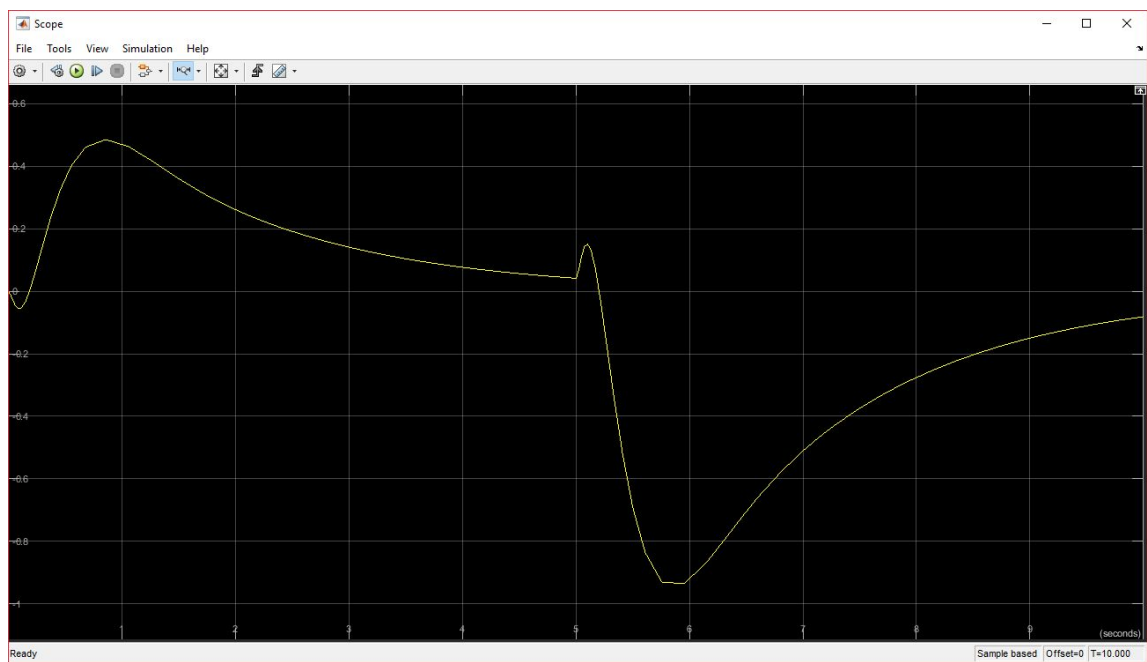


Figure 16: Simulink Scope view for “square” input vs. angle.

Experimental Results

Data Collected

We initially used our K value of 28.3 to run our experiment. What we observed was a bit of back and forth correction from the seesaw apparatus that eventually settled out to small back and forth corrections to keep the system stable. The following output was observed from the scope tool:

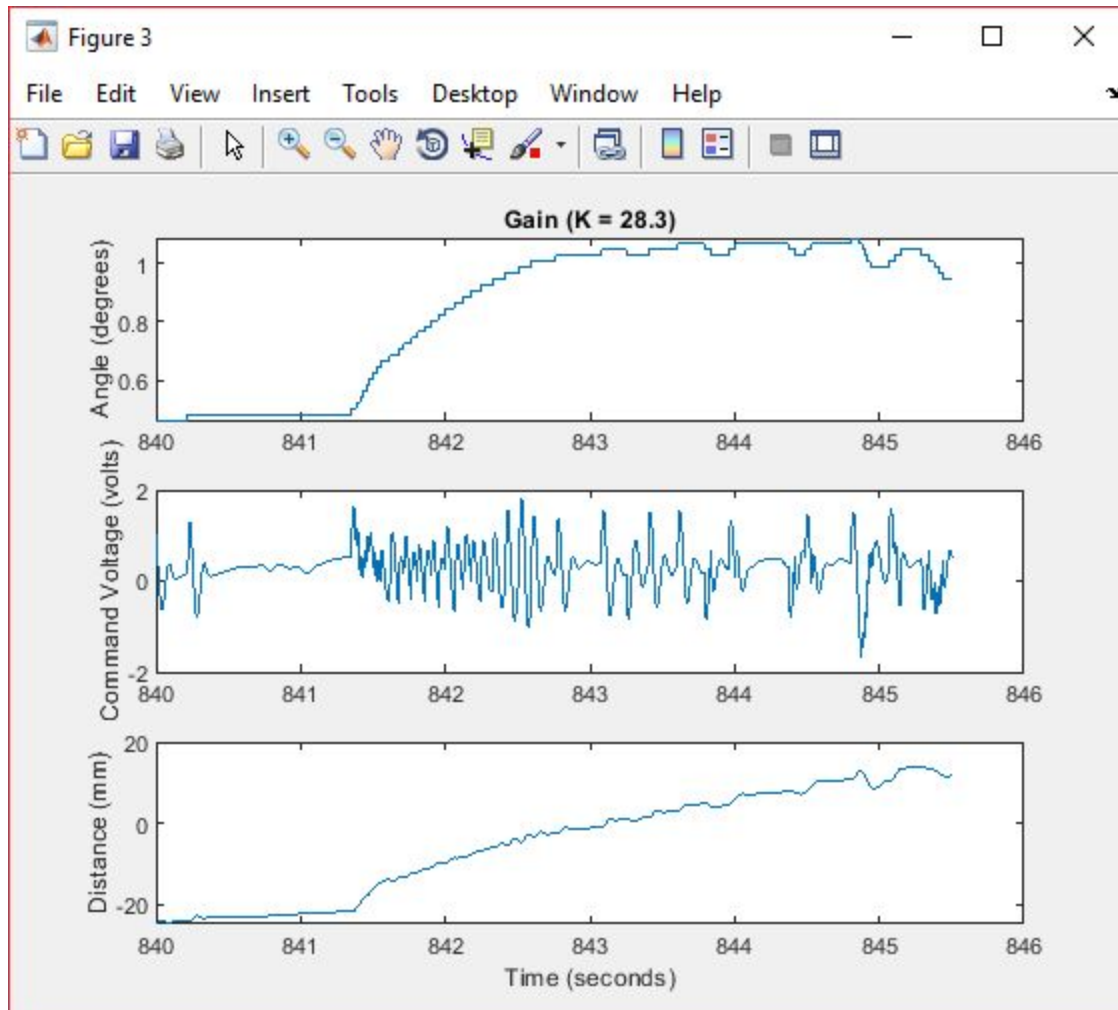


Figure 17: Simulink Scope for $K = 28.3$

In the good spirit of experimentation, we decided to run the experiment with Colton's gain value. The scope had the following output:

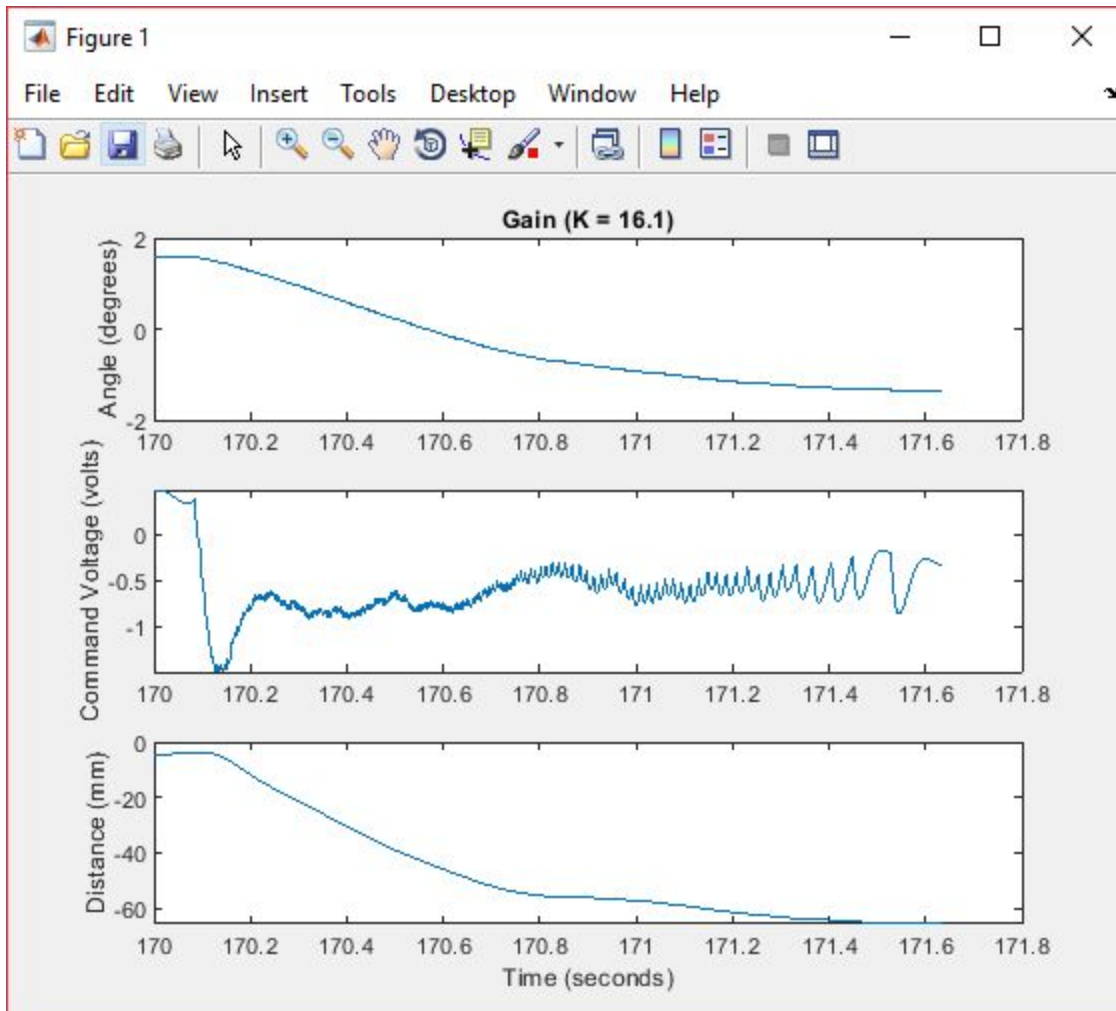


Figure 18: Simulink Scope for K = 16.

With the gain value of 16, the seesaw initially didn't move very much. While it attempted to settle though, the movements from the arm became larger and larger. This behavior fits with our predictions--the seesaw oscillated excessively, but not dangerously so.

As instructed, we introduced a small disturbance into our system (i.e. we tapped the seesaw gently.) The behavior we observed was increasingly smaller over corrections until the system reached relative balance. The scope gave us the following output:

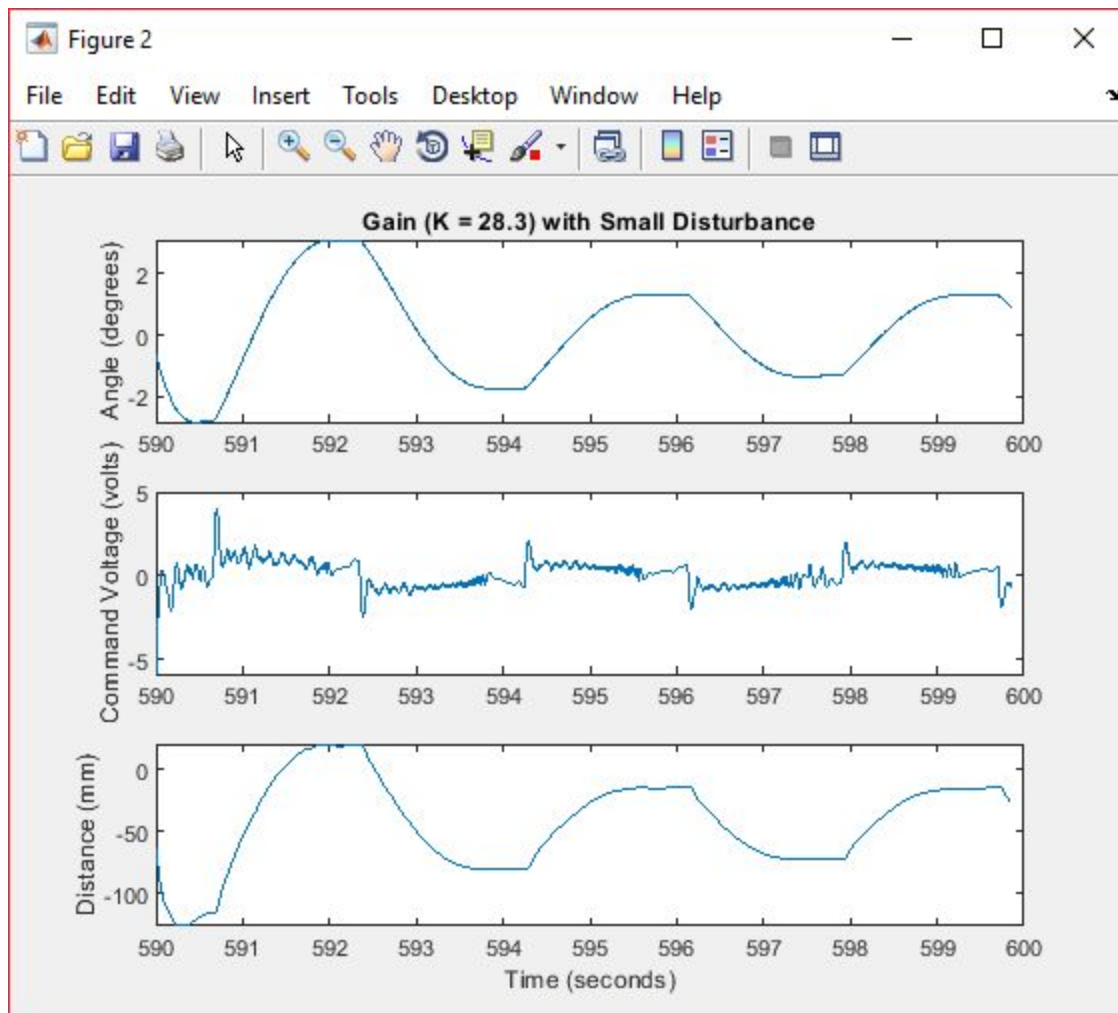


Figure 19: Simulink Scope for $K = 28.3$ and small disturbance.

Discussion

For the most part, I think our results matched our predictions. Going to Patrick's office hours was definitely helpful in clarifying my expectations for the lab. Inconsistencies in the observations likely include forces that we didn't take into consideration in our model (such as friction, for example.)

The following questions were asked in our lab guide:

- 1.) **Does the system achieve balance? Are its motions random? Periodic? Why might it behave this way?**

The system achieved balance to some noticable extent in all of the experiments ran. The

system that reached the most reliable balance (and achieved it fastest) was the gain value of 28.3. The system that used the gain value of 16.1 was not necessarily unbalanced, but the over correction was larger less effective. The system with the gain value of 28.3 seemed more efficient.

The system with the gain value of 28.3 behaved the way we expected. I think the reason for the more erratic behavior in the system with gain value of 16.1 is because the second-order approximation wasn't correctly justified. I think the pole on the real axis was being eliminated, affecting the behavior of the system.

- 2.) **Can we learn anything about the system's response to a tap disturbance from its square wave simulation? Are the response shapes similar? Why does the simulated response initially move in the opposite direction of the initial step? Could the settling time have been predicted?**

I think the response shapes are similar. The reason for the opposite directions is because a step response is starting from 0 and going upward. The tap isn't necessarily starting at 0 and the response of the system will be in the opposite direction of the initial movement. Since the system eventually found balance, I think the settling time likely could have been predicted with more sophisticated methods.

- 3.) **Did your original design meet its design goals when implemented? What role did theory play in development of your controller?**

Yes, our design met the goals when we implemented it. Theory played a vital role in our development, specifically in the pre-lab analysis portion.