

Lab 4

Drew Bixler, Chris Cross, Alexander Kopeny

Introduction

In this lab, we were tasked with creating a model for the inverted pendulum that we will be working on for the next few labs. Before the lab, we did the prelab to find our expected model for the system. During the lab, we performed a few different tests to validate this model.

Simulation: Model Analysis

In this portion of the lab, we began by running the required script to save the values for the parameters in our equation into the MATLAB workspace. We then filled in our equations for matrices A, B, C and D. Finally, we verified that our matrices matched what our TA had found to ensure that our model was correct. Included below is our MATLAB code that demonstrates what we found for our matrices, as well as the output on MATLAB showing the matrices after applying the parameter values. We also included a table showing what we got for each matrix and the open loop poles after inputting the parameter values. After doing this, we used the eigenvalues of the A matrix to compute the open loop poles. We found that one of the poles was in the right half plane, meaning that our system will be unstable.

```
% State Space Representation
A = [0, 0, 1, 0;
     0, 0, 0, 1;
     0, ((Lp^2*Lr*g*Mp^2)/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)),
     -(Mp*Lp^2+4*Jp)*0.0024/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr) + ((eta_g*Kg^2*km*eta_m
     *kt)/Rm)*((Mp*Lp^2+4*Jp)/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)),
     -(2*0.0024*Lp*Lr*Mp)/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)];
     0, ((2*Lp*g*Mp*(Mp*Lr^2+Jr))/(Jr*Mp*Lp^2+4*Mp*Jp*Lr^2+4*Jp*Jr)),
     -(((2*0.0024*Lp*Lr*Mp)/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)) + ((eta_g*Kg^2*km*eta_m
     *kt)/Rm)*((2*Lp*Lr*Mp)/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr))),
     -(Mp*Lr^2+Jr)*4*0.0024/(Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)];
B = [0; 0;
     (Mp*Lp^2+4*Jp)*eta_g*Kg*eta_m*kt/((Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)*Rm);
     (2*Lp*Lr*Mp*eta_g*Kg*eta_m*kt)/((Jr*Mp*Lp^2+4*Jp*Mp*Lr^2+4*Jp*Jr)*Rm)];
C = eye(2,4);
D = zeros(2,1);
% Add actuator dynamics
A(3,3) = A(3,3) - Kg^2*kt*km/Rm*B(3);
A(4,3) = A(4,3) - Kg^2*kt*km/Rm*B(4);
%B = Kg * kt * B / Rm;
```

MATLAB R2023a

HOME
PLOTS
APPS
EDITOR
PUBLISH
VIEW

Search (Ctrl+Shift+Space)

New
Open
Save
Print
Compare

Go To
Find
Bookmark
NAVIGATOR

Refactor
CODE

Code Issues
Debugger
Generate Test
Find Tests
Run Section
Run and Advance
Run to End

Run
Step
Stop

Files
U > Y452 > EE 476 > Module 2 > studentMaterials

Name
Size
Date Modified

csc_conversion_constants.m 1 KB 2/27/2026 1:13 PM

config_gm.m 5 KB 2/27/2026 1:13 PM

config_sn02.m 9 KB 2/27/2026 1:13 PM

d_pole_placement_student.m 126 KB 2/27/2026 1:13 PM

q_rotpen_bal_student.mdl 115 KB 2/27/2026 1:13 PM

q_rotpen_wingtip_student.m 139 KB 2/27/2026 1:13 PM

Rotary Pendulum Workbook... 1.82 MB 2/27/2026 1:13 PM

ROTPEN_ABCO_egns_stude... 1 KB 2/27/2026 1:21 PM

ROTPEN_ABCO_egns_stude... 1 KB 2/27/2026 1:13 PM

setup_rotpen.m 3 KB 2/27/2026 1:13 PM

s_rotpen_bal.mdl 37 KB 2/27/2026 1:13 PM

setup_rotpen.m (x)
ROTPEN_ABCO_egns_student.m x

U:\Y452\EE_476\Module 2\studentMaterials\ROTPEN_ABCO_egns_student.m

```

1
2
3
4
5
6
7
8
9
10
11
12
13
14

```

Workspace

Name
Value
Size
Class

Dr 0.0024 1x1 double

epsilon 0.2094 1x1 double

eta_g 1 1x1 double

eta_m 1 1x1 double

ENVCODER_TV 'E' 1x1 char

Er 0 1x1 double

DXT_OEAR_C... 'HIGH' 1x4 char

g 9.8100 1x1 double

IMAX_AMP 4 1x1 double

Im 0.0021 1x1 double

Im 4.6061e-07 1x1 double

Ip 0.0012 1x1 double

Jn 9.9829e-04 1x1 double

kn 0.0077 1x1 double

kt 0.0077 1x1 double

K [0 0 0] 1x4 double

K_AMP 1 1x1 double

K_ENC 0.0015 1x1 double

K_FOT -0.6144 1x1 double

Command Window

```

A =
    0    0    1.0000    0
    0    0    0    1.0000
    0   81.4013  -45.8259  -0.3129
    0  122.0545  -44.0966  -1.3972

B =
    0
    0
    83.4659
   80.3162

```

Editor

90% UTF-8 CR LF
Script Ln 5 Col 26

MATLAB R2023a

HOME

PLOTS

APPS

EDITOR

PUBLISH

VIEW

Search (Ctrl+Shift+Space)

New

Open

Save

Print

Compare

Go To

Find

Find in Files

File

Edit

View

Window

Help

Refactor

Code Issues

Debugger

Generate Test

Find Tests

Run Section

Section Break

Run and Advance

Run to End

Run

Step

Stop

CODE

ANALYZE

TEST

SECTION

RUN

Files

U: > Y452 > EE 476 > Module 2 > studentMaterials

Files

Name	Size	Date Modified
calc_conversion_constants.m	1 KB	2/27/2026 1:13 PM
config_gain.m	5 KB	2/27/2026 1:13 PM
config_pid2.m	9 KB	2/27/2026 1:13 PM
q_rotpen_placement_student.m	1 KB	2/27/2026 1:13 PM
q_rotpen_bal_student.mdl	126 KB	2/27/2026 1:13 PM
q_rotpen_md1_student.mdl	115 KB	2/27/2026 1:13 PM
q_rotpen_swingup_student...	139 KB	2/27/2026 1:13 PM
Rotary Pendulum Workbooks	182 KB	2/27/2026 1:13 PM
ROTFEN_ABCD_eqns_student...	1 KB	2/27/2026 1:21 PM
ROTFEN_ABCD_eqns_student...	1 KB	2/27/2026 1:13 PM
setup_rotpen.m	3 KB	2/27/2026 1:13 PM
s_rotpen_bal.mdl	37 KB	2/27/2026 1:13 PM

ROTFEN_ABCD_eqns_student.m

```

1 % State Space Representation
2 A = [0, 0, 1, 0;
3       0, 0, 0, 1;
4       ((Lp^2*Lr*gm^2)/(2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3))-((Lp*Lr^2+4*Lp^3)*0.0024/(2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3))-((eta_g*kg^2*km*eta_mkt)/Bm)*(Lp*Lr^2+4*Lp^3)/(2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3)
5       0, ((2*Lr*gm^2*Lp*Lr^2)/(2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3))-(((2*Lr*0.0024*Lp*Lr^2)/(2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3))+((eta_g*kg^2*km*eta_mkt)/Bm)*(2*Lp*Lr*Lp)/(2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3)
6       B = [0, 0, (Lp*Lr^2+4*Lp^3)*eta_g*kg*eta_mkt)/((2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3)), (2*Lp*Lr*Lp*eta_g*kg*eta_mkt)/((2*m*Lp*Lr^2+4*Lp^3*Lr^2+3*Lr^3))]
7       C = eye(2,4);
8       D = zeros(2,1);
9
10 % Add actuator dynamics
11
12 NA(3,3) = A(3,3) - Kg^2*kt*km/Bm/8(3);
13 NA(4,3) = A(4,3) - Kg^2*kt*km/Bm/8(4);
14 NB = Kg + kt + 0 / Bm;
15
16 %N Question 3
17
18 open_loop_poles = eig(A)

```

Workspace

Name	Value	Size	Class
B	[0.0834659 0.03]	4x1	double
Bc	[0.00 1]	4x1	double
Beq	0.0150	1x1	double
chk_poles	[0.0 0.0]	2x4	double
C	[1.0 0.0 0.0 1.0]	2x4	double
D	[0 0]	2x1	double
Dp	0.0024	1x1	double
Dr	0.0024	1x1	double
epsilon	0.2094	1x1	double
eta_g	1	1x1	double
eta_m	1	1x1	double
ENCODER_TY...	'E'	1x1	char
Er	0	1x1	double
EXT_GEAR_C...	'HIGH'	1x4	char
g	9.8100	1x1	double
IMAX_AMP	4	1x1	double
Ieq	0.0021	1x1	double
Im	4.6083e-07	1x1	double
Ip	0.0012	1x1	double

Command Window

```

>>
0
-57.4164
6.9513
-6.9367

>> ROTFEN_ABCD_eqns_student

open_loop_poles =

0
-48.4156
7.0509
-5.8644

>>

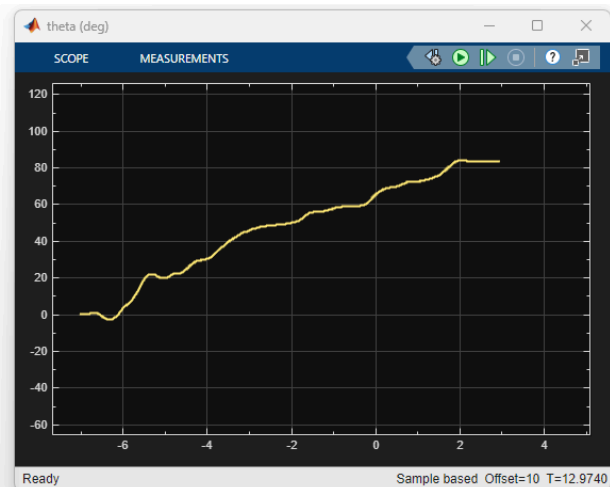
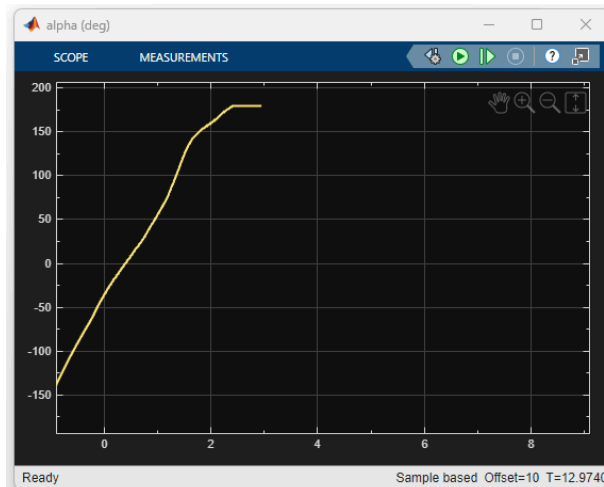
```

Editor: 90% UTF-8 CRLF Script Ln 13 Col 1

Description	Symbol	Value	Units
State-Space Matrix	A	[0, 0, 1, 0 0, 0, 0, 1 0, 81.4033, -45.8259, -0.9319; 0 122.0545, -44.0966, -1.3971]	[m/s; m/s; m/s^2; m/s^2]
State-Space Matrix	B	[0; 0; 83.4659; 80.3162]	[m/s; m/s; m/s^2; m/s^2]
State-Space Matrix	C	[0, 0, 0, 0]	[m/s; m/s; m/s^2; m/s^2]
State-Space Matrix	D	[0;0;0;0]	[m/s; m/s; m/s^2; m/s^2]
Open-Loop poles	<i>OL</i>	[0;-48.4156; -5.8644; 7.0569]	

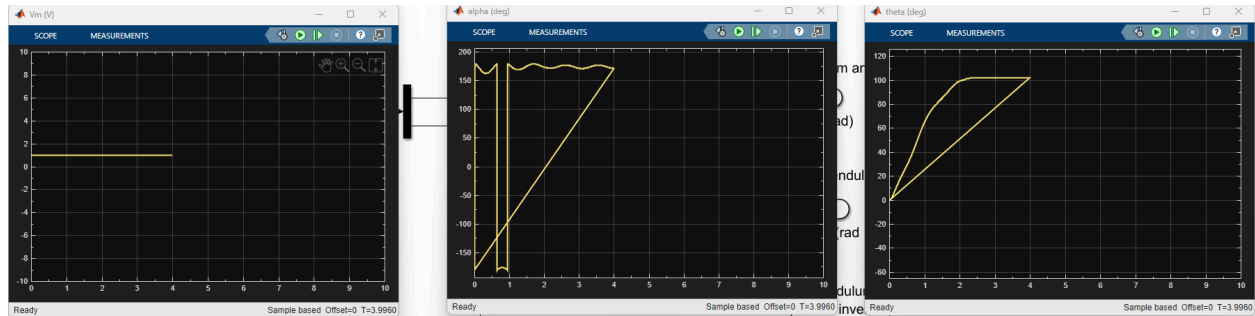
Experimental Setup

5) Chris Cross



When the rotary arm angle and the pendulum angle were rotated counterclockwise, both measurements increased in the positive direction. This observation matches the modeling conventions in our manual and discussed in class. According to the model, counterclockwise rotation corresponds to a positive angular displacement. The readings from the system confirmed that the hardware follows the same sign convention used in the math model, indicating that the experimental setup is consistent with the modeling assumptions.

Adjusting Source Block for Convention



Initially with convention gain control set to the default value 0, the output of the rotary arm was inverted from convention. We then set the *Direct Convention* gain block to -1, which fixed our inversion issue and the data displayed as expected. No further trials were attempted, as the issue with convention had been fixed in the first trial.

Conclusion

In this lab we were given a rotary arm and tasked with building the model for the inverted pendulum which is at the center of the module 2 labs. First we completed the prelab to find the expected model of the system, and from that we calculated the system's open-loop poles. In the experimental section of the lab we rotated the rotary arm manually and recorded data in an effort to determine whether or not the arm complied with the convention of counter-clockwise movement being positive. Once this was verified, we similarly tested convention on the rotary arm under a 1 V input. We determined that using a value of -1 in the *Direct Convention* gain block would resolve issues associated with convention under these circumstances.

Members: Drew Bixler, Alex Kopeny, Alex Hardcopf, Chris Cross

Lab 5 Report

Introduction:

In this lab, we implemented the balance controller that we designed in the prelab. We both simulated it and used the controller with the pendulum to ensure that our controller worked properly. While doing this, we also needed to create the switch that would decide whether the system should use the balance controller, or a gain of zero (this will be switched in the next lab to be the swing up controller).

Control Design:

Our control design begins from where the previous lab left off, with *setup_rotpen_student.m*

Using MATLAB commands, the team was tasked to see if our system was controllable. After finding our controllability matrix, we were able to check the rank of the matrix using MATLAB commands. Since the rank of our 4x4 matrix was 4, we were able to determine that the system was controllable and can be seen below:

```
T =  
  
1.0e+06 *  
  
     0     0.0001    -0.0039     0.1888  
     0     0.0001    -0.0038     0.1871  
  0.0001    -0.0039     0.1888    -9.1342  
  0.0001    -0.0038     0.1871    -9.0489  
  
rank_T =  
  
4
```

In the next step, we are supposed to open the *d_pole_placement_student.m* script to perform the pole placement design for our rotary pendulum system. Running this script will automatically generate the companion form matrices A_c and B_c from the system matrix A . It is then modified to compute the controllability matrices T and T_c , the inverse T_c^{-1} , and the transformation matrix W . These are used to convert the companion-form gain into the actual state-feedback gain K .

% Characteristic equation: $s^4 + a_4s^3 + a_3s^2 + a_2s + a_1 = \text{poly}(A)$;

% Companion matrices (A_c , B_c)

$A_c = [0 \ 1 \ 0 \ 0; 0 \ 0 \ 1 \ 0; 0 \ 0 \ 0 \ 1; -a(5) \ -a(4) \ -a(3) \ -a(2)]$;

$B_c = [0; 0; 0; 1]$;

% Controllability $T = 0$;

% Controllability of companion matrices $T_c = 0$;

% Transformation matrices $W = 0$;

```
setup_rotpen.m × ROTPEN_ABCD_eqns_student.m × d_pole_placement_student.m × +
U:\Y4S2\EE 476\Module 2\student\Materials\d_pole_placement_student.m
1 % IMPORTANT: Make sure you run setup_rotpen.m first. You need the (A,B)
2 % state-space matrices.
3 %
4 %% Find Transformation Matrix W
5 % Characteristic equation:  $s^4 + a_4s^3 + a_3s^2 + a_2s + a_1$ 
6 a = poly(A);
7 %
8 % Companion matrices (Ac, Bc)
9 Ac = [ 0 1 0 0;
10       0 0 1 0;
11       0 0 0 1;
12       -a(5) -a(4) -a(3) -a(2)];
13 %
14 Bc = [0; 0; 0; 1];
15 %
16 % Controllability
17 T = [B A*B A^2*B A^3*B];
18 rank_T = rank(T);
19
20 % Controllability of companion matrices
21 Tc = [Bc Ac*Bc Ac^2*Bc Ac^3*Bc];
22 rank_Tc = rank(Tc);
23 Tci = inv(Tc);
24
25 % Transformation matrices
26 W = T*Tci;
27 %
```

T =

1.0e+06 *

0	0.0001	-0.0039	0.1888
0	0.0001	-0.0038	0.1871
0.0001	-0.0039	0.1888	-9.1342
0.0001	-0.0038	0.1871	-9.0489

Tc =

1.0e+05 *

0	0	0	0.0000
0	0	0.0000	-0.0005
0	0.0000	-0.0005	0.0233
0.0000	-0.0005	0.0233	-1.1267

Tci =

1.0e+03 *

-2.0036	-0.0991	0.0472	0.0010
-0.0991	0.0472	0.0010	-0.0000
0.0472	0.0010	0	0
0.0010	-0.0000	0	-0.0000

W =

1.0e+03 *

-3.6494	0.0418	0.0835	-0.0000
0.0000	-0.0000	0.0803	-0.0000
-0.0000	-3.6494	0.0418	0.0835
-0.0000	0.0000	0.0000	0.0803

Plugging in the known matrices for A, B, and K, the resulting eigenvalues lined up with our desired results, and no further iterations of the control design were required to find a gain that places the poles in the required locations.

For previous exercises, the team was able to find the gain K manually. In order to verify that K was the same as before using the MATLAB's pole-placement command. We found the same gain values when we used the Matlab pole-placement command.

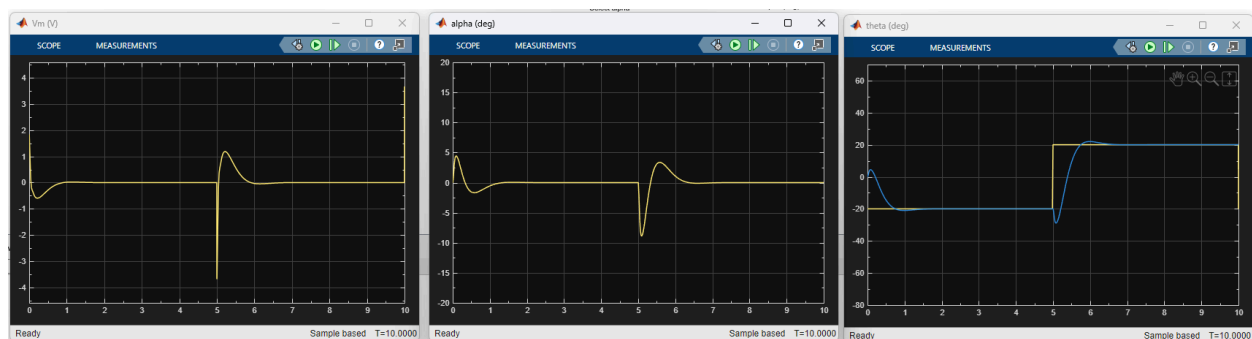
$K =$

$-5.2620 \quad 28.1577 \quad -2.7576 \quad 3.2191$

Simulation of the Balance Control:

This section begins the simulation portion. Running `setup_rotpen.m` loads the rotary pendulum model and ensures the previously calculated state feedback gain K is available in the MATLAB workspace before the Simulink model is loaded. This is important because the simulation reads the system matrices and controller gain directly from the workspace.

In this section, we used the simulation in simulink to plot our expected theta, alpha, and voltage responses using the controller that we designed. This will allow us to see if our controller should accomplish the desired goal of balancing the pendulum and staying within the desired specs. The output plots are included below.



We measured pendulum deflection at below 5 degrees and the voltage used below 1 volt. The given measurements in the module manual specify a maximum deflection of 15 degrees and a maximum voltage of 10 volts. Given these parameters our measured data is well within acceptable limits.

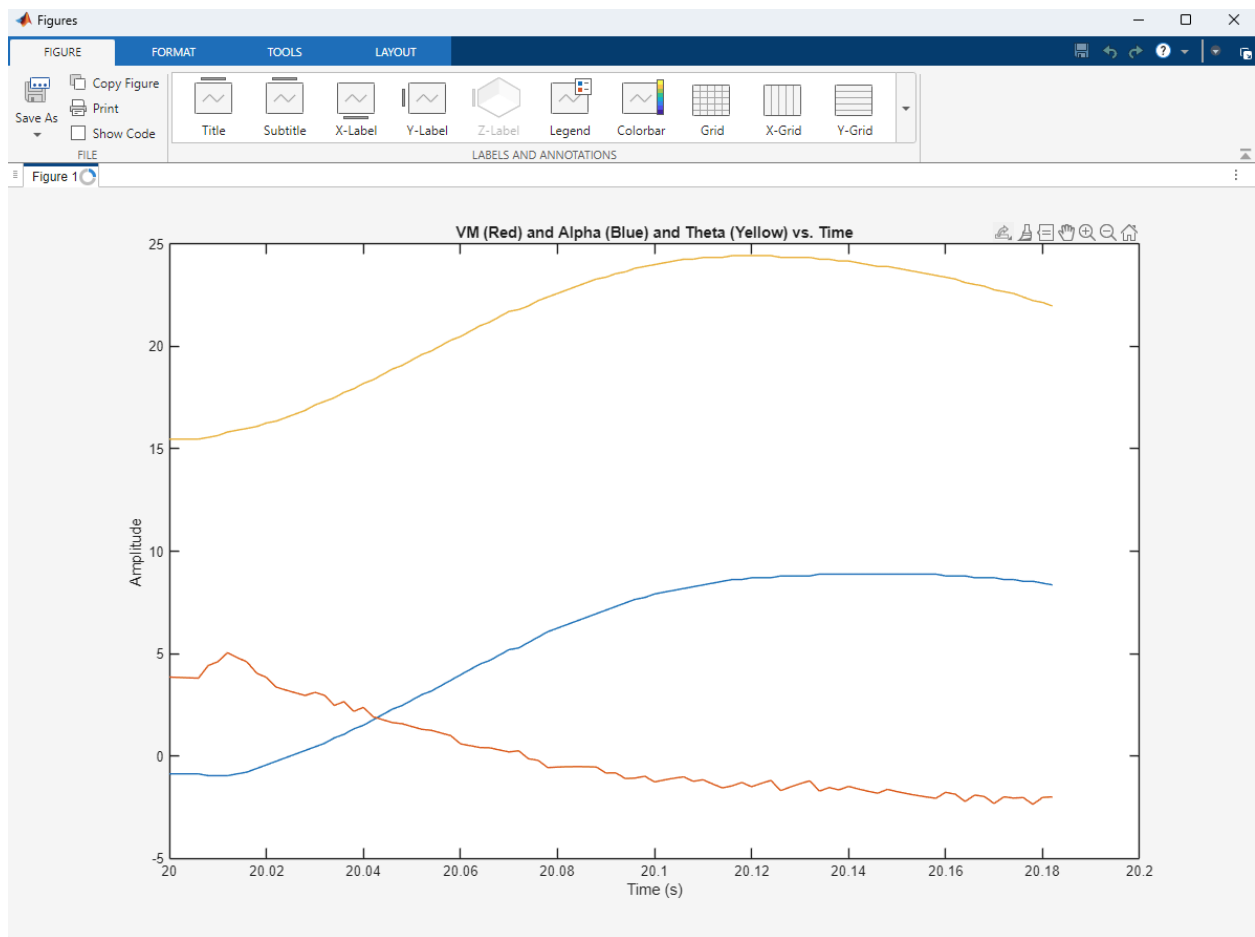
Implementing the Balance Controller:

This section begins the hardware implementation. The purpose is again to load the rotary pendulum model and controller parameters into MATLAB before running the controller on the actual Quanser setup. In the implementation state, this script preps the model so that the Simulink diagram can communicate correctly with the balance controller.

After opening the q_rotpen_bal_student Simulink Diagram and turning on the power amplifier, Quarac is used to compile the control model for real time implementing on the rotary pendulum hardware. This step converts the Simulink controller into an executable Quarac application so it can interact with the DC motor and system sensors.

Using blocks from the Simulink library, balance control is implemented. In a similar fashion we implemented switch logic in order to make sure the pendulum does not attempt to balance itself while inverted, as it will begin in real tests

After the pendulum is balance and the $\pm 20^\circ$ rotary arm command is applied, we now have to press the stop button. Once stopped, the response data is saved to our MATLAB workspace, which we can then use to plot the rotary angle, pendulum angle, and motor input voltage response in a MATLAB graph. This allows the experimental hardware response to be compared with the simulation results and checks against the lab specifications.



When we measured the pendulum deflection and voltage used we found that the pendulum deflection was about 9.5 degrees at its peak, well below the 15 degree specification. When measuring the voltage, we found that it maxed out around 5.1V which was well below the 10V limit.

Description	Symbol	Value	Units
Pre-Lab Questions			
Desired Poles	DP	-2.8+j2.8566, -2.8-j2.8566, -30, -40	
Companion Gain	$\tilde{K}$	[19203, 9843.65, 1707.12, 28.38]	
Simulation: Control Design			

Transformation Matrix	W	<pre> rank_T x T x A x W x 4x4 double 1 2 3 4 1 -3.6494e+03 41.7771 83.4659 -1.8584e-15 2 5.8208e-11 -3.8855e-13 80.3162 -1.8249e-15 3 -1.8626e-09 -3.6494e+03 41.7771 83.4659 4 -1.8626e-09 3.3686e-11 9.0949e-13 80.3162 </pre>	
Control Gain	K	<pre> K = -5.2620 28.1577 -2.7576 3.2191 </pre>	
Closed Loop Poles	CLP	<pre> cls_poles = -40.0138 + 0.0000i -29.9895 + 0.0000i -2.7999 + 2.8571i -2.7999 - 2.8571i </pre>	
Simulation: Closed Loop System			
Maximum Deflection	$ \alpha _{\max}$	5	deg
Maximum Voltage	$ V_m _{\max}$	1	V
Implementation			
Control Gain	K	<pre> K = -5.2620 28.1577 -2.7576 3.2191 </pre>	
Maximum Deflection	$ \alpha _{\max}$	8.34	deg
Maximum Voltage	$ V_m _{\max}$	4.28	V

Conclusion:

In this lab we worked on creating a program that could effectively balance the pendulum when turned upright. We started by simulating the control system, testing a version on the real pendulum arm only after the simulations yielded the desired results. In addition, this lab implemented a switch control, only starting balance control when the pendulum arm was upright and giving the system a gain of zero otherwise. In the next lab, this gain of zero will instead be changed to our swing-up control, and the pendulum arm control system will be completed.

EE 4760 LAB 6

Emma Heithoff, Drew Bixler, Caleb Rezmer, Chris Cross, Alexander Kopeny, Jacob Feldmann

Introduction:

In this lab, we are working with the pendulum system again to implement an energy-based swing-up controller. We will use this to complement our existing system, which is modeled and set up so we can balance it. Through our prelab work, we laid the groundwork for setting up this system and did the math to determine what we need to get the swing-up function working for our pendulum system.

6.2.1 Swing-Up Design and Implementation

To start this lab, we opened the 'setup_rotpen.m' script and ensured that the value 'K' was loaded as our gain from the control section. Next, we opened the 'q_rotpen_swingup_student' Simulink diagram and started up our system. As we went through the diagram, we ensured that the modifications we made in Lab 5 were transferred over. Before we applied the new swing-up control, we retested the balance control and verified that it still worked as expected.

From there, we could move on to the swing-up function, so we opened up that subsystem. We then modified the subsystem to measure the system's total potential energy. After we made those modifications using the existing pendulum parameters and the 'config_sp' function, we were able to measure the system's potential energy. So, we compiled the system and ran the controller to get a reading. When we rotated the pendulum to the upright position and ensured that it was balancing we were able to capture the measured potential energy. The value that we measured in the system was 0.4193 J, which is shown in Figure 1, with the modified subsystem used to get this reading shown in Figure 2, and when we compare this to our calculated expected potential energy of 0.4199 J, we can see that we have realized in practice what we expected in theory, therefore validating our system.

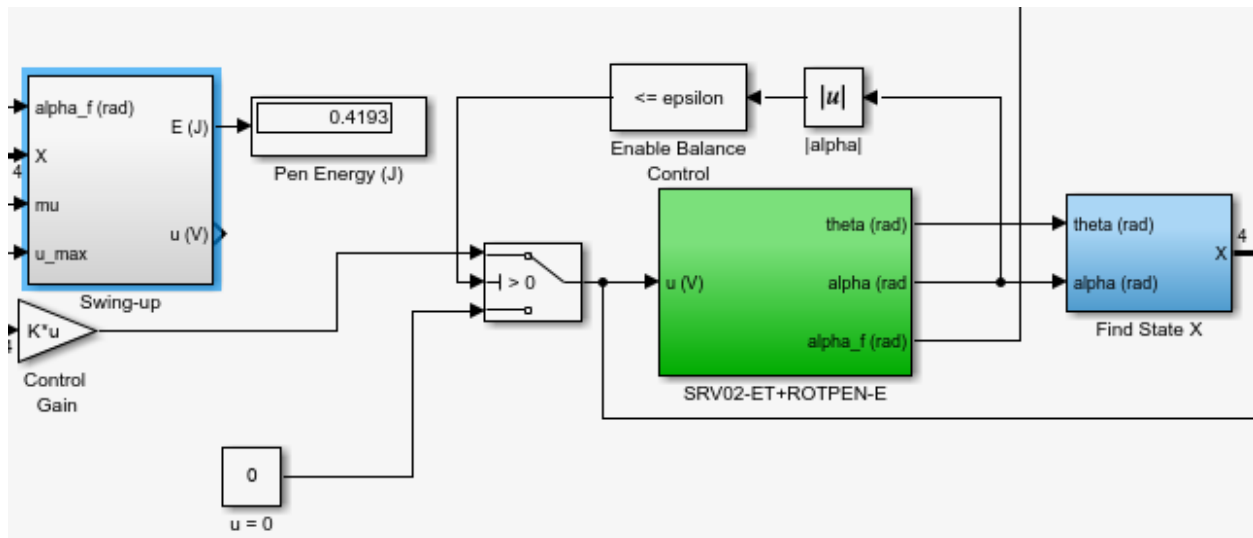


Figure 1: Our block diagram demonstrating a measured PE of 0.4193 J

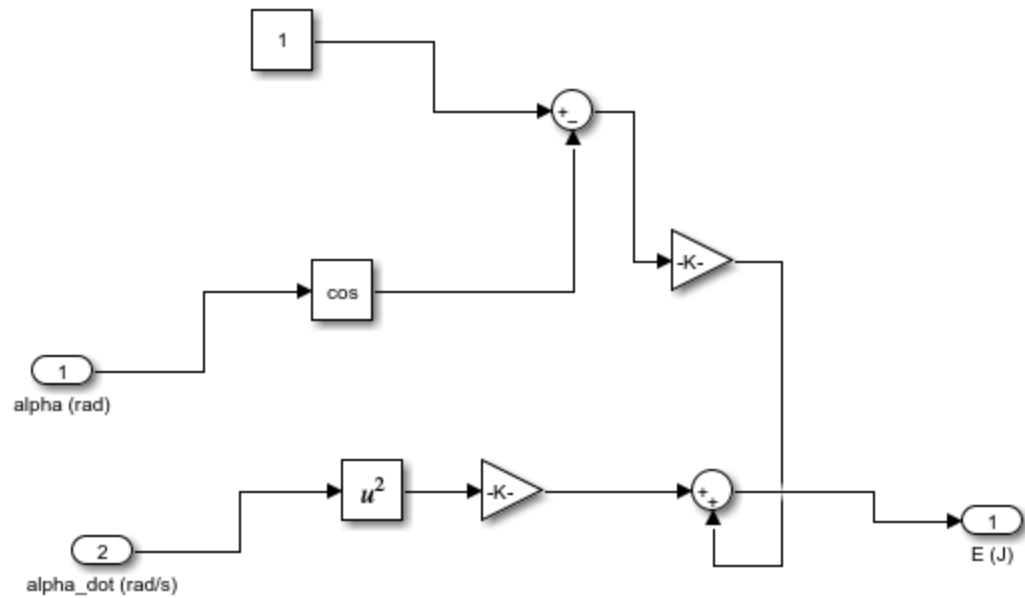


Figure 2: Modified potential Energy Subsystem

At this point, we are ready to implement the energy-based swing-up controller that is the subject of this lab. To do this, we had to modify the 'swing-up control subsystem'. Through these modifications, we were sure to use the source block with 'E_r' and the inputs 'u_{max}' and 'μ' that were already included. We then made the necessary modifications to convert the generated acceleration to a voltage for the servo by editing the subsystem and using the SRV02 parameters already defined for us to meet the required parameters. From that work, we were able to implement the control that includes both the swing-up function and the balance attribute. With this, we plotted it, and our plot is shown in Figure 3, demonstrating the rotary arm, pendulum, and servo voltage. The modified subsystem is also shown below in Figure 4, demonstrating the work that we did in order to reach this point. As we worked to marginally increase μ to a point where the swing-up function worked effectively and efficiently.

SRV02 Inverted Pendulum: Balance Control

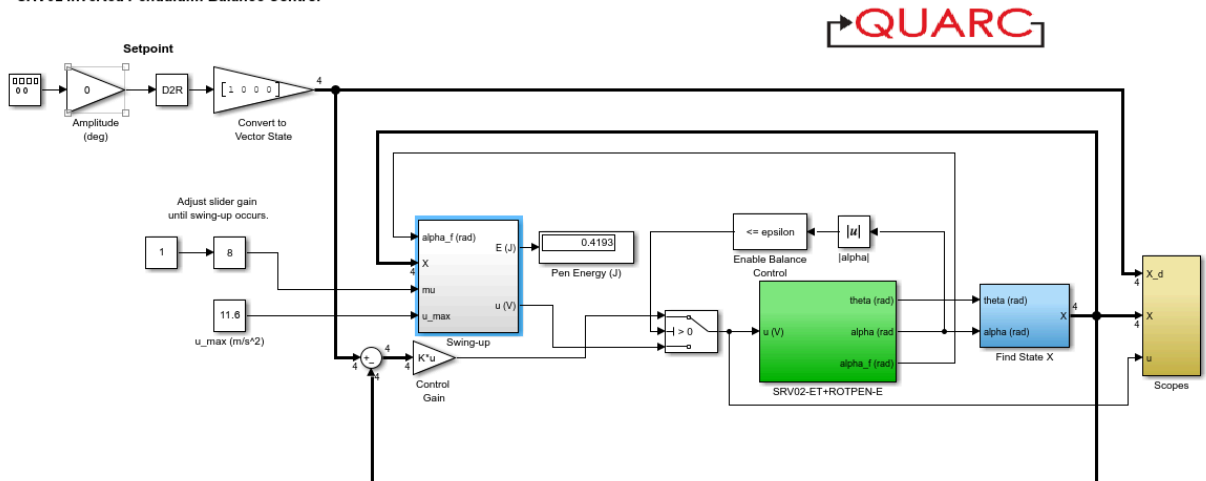


Figure 3: Modified Subsystem

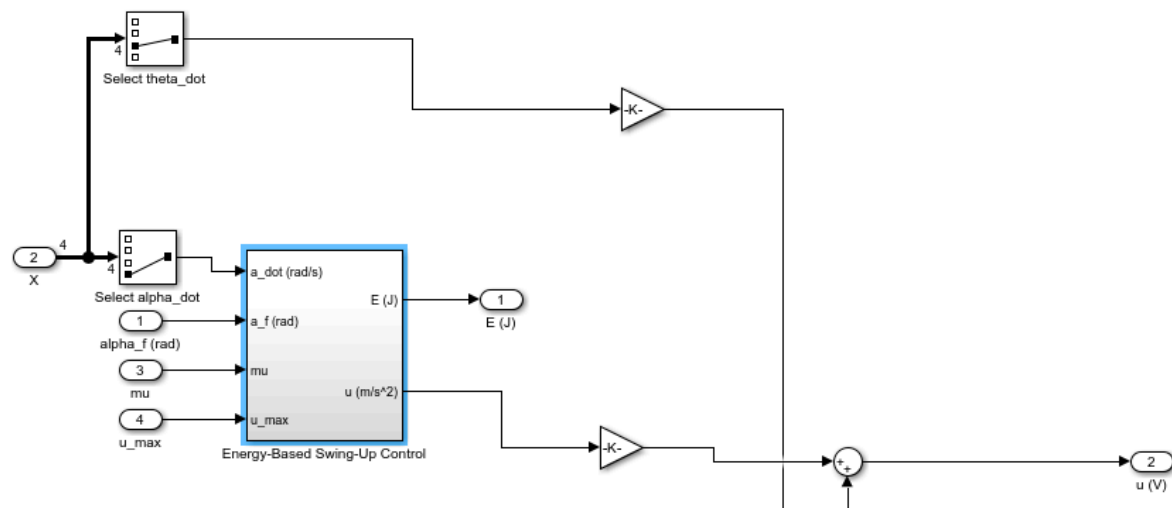


Figure 4: Swing-Up Controller Subsystem

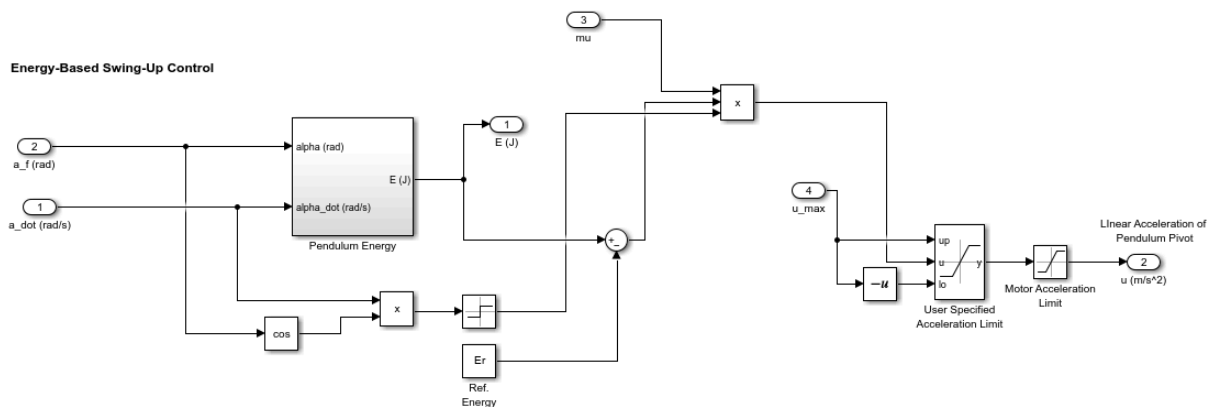


Figure 5: Energy-Based Swing-Up Control

From this point, we can set the reference energy, maximum acceleration, and proportional gain parameters to the values given in the lab, with special attention to the reference energy being our measured potential energy. We then compiled and ran the system, and it began to move. We then, again, marginally increased μ and $u_{\max}$ until the pendulum went up, paying special attention to ensure that our $u_{\max}$ did not exceed the maximum acceleration we found for SRV02 in the prelab. After fine-tuning these values, we noticed that once the control could swing the pendulum up, the balance function took over and kept the pendulum upright. This was exciting and gratifying to see, signifying success after several attempts and many lab sessions. From this demonstration, we got a plot of the response of the arm, pendulum angles, and control voltage, and these are shown in Figure 6, 7, and 8. We also have the parameters and implementation variables used to achieve this successful demonstration, as recorded in Table 1. From here, we were able to call the lab successful, so we turned off our controller and the pendulum system.

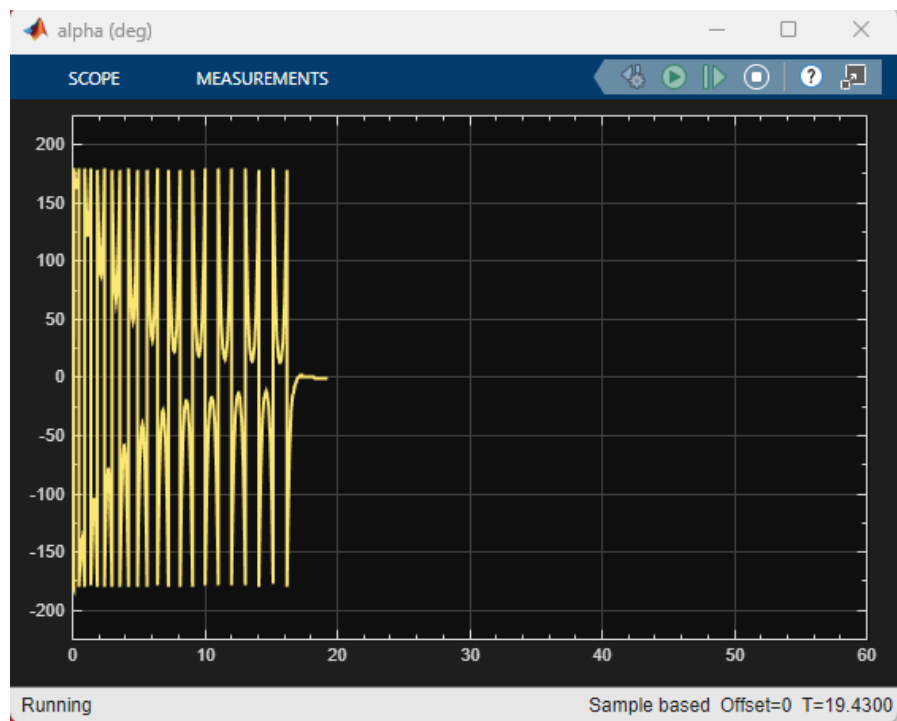


Figure 6: Plot of Alpha

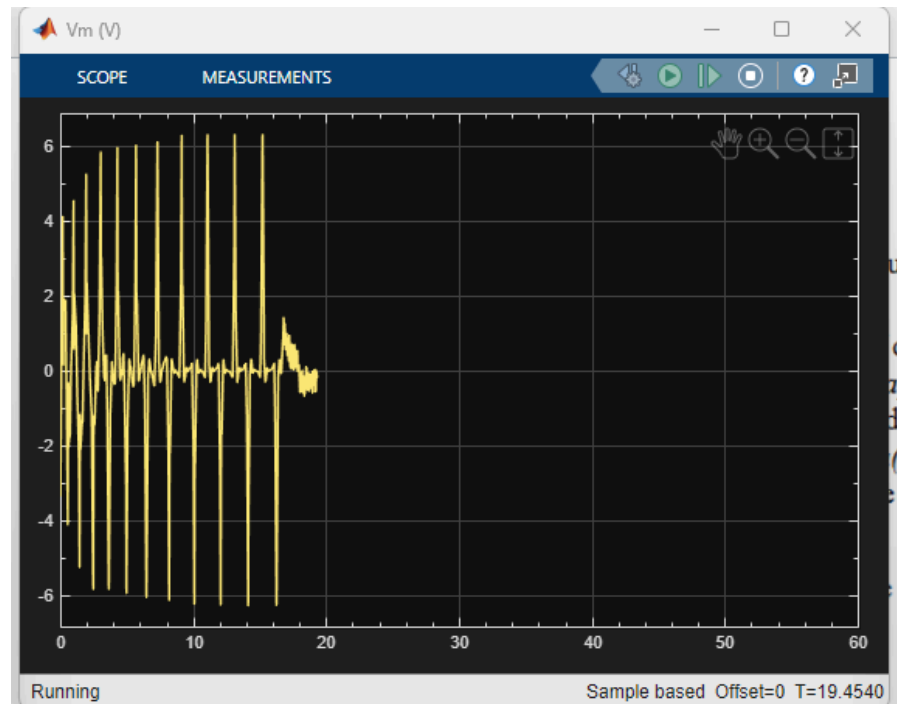


Figure 7: Plot of Vm

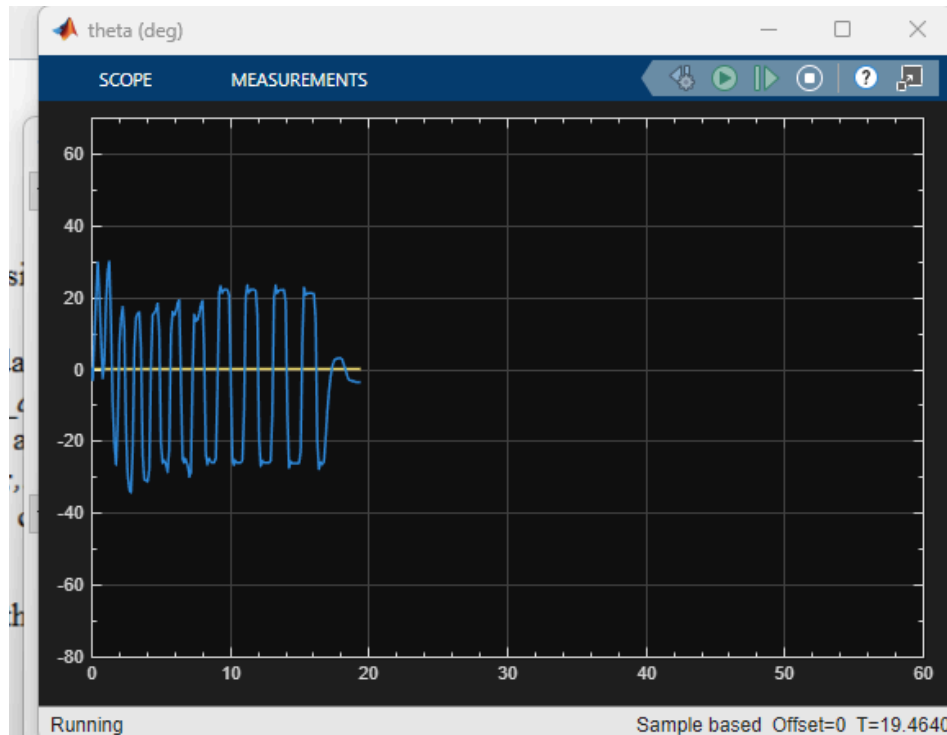


Figure 8: Plot of theta angle as our pendulum goes from off to balanced

Description	Symbol	Value	Units
Pre-Lab Questions			
Potential Energy	E_p	0.4193	J
Maximum Acceleration of SRV02	$u_{\max}$	11.6	m/s^2
Implementation			
Balance Control Gain	K	$\begin{bmatrix} -5.2620 & 28.1577 & -2.7576 \\ 3.2191 \end{bmatrix}$	
Reference Energy	E_r	0.42	J
Control Maximum Acceleration	$u_{\max}$	11.6	m/s^2
Swing-Up Proportional Gain	μ	8	

Table 1: Results from throughout the lab

Conclusion:

In this lab, we successfully modeled, designed, simulated, implemented, and validated a swing-up controller for our existing pendulum system. As we started off by customizing our block diagram and finding the total potential energy of the pendulum arm, we were able to use that along with further given and determined parameters to first simulate the swing-up function, and get that to a working point where the pendulum could swing all the way up by incrementing μ . We then implemented it on the actual system, ran it, and followed a similar procedure by incrementing μ and $u_{\max}$ until we reached a point where it could swing the pendulum all the way up. After it swung to a vertical position, we ensured the balancing function took over and kept the pendulum arm vertical and balanced. Ultimately, this success and demonstration proved the controller that we had designed can successfully control and operate this system, which meant a successful lab for us. Through this project as the finale to a series of labs with the pendulum system, we were able to realize success by modeling, then designing and implementing a system that could balance the pendulum arm, and finally designing and implementing a system that can swing the pendulum arm from a downward to an upward position. Together, we were able to see how many parts, pieces, and individual tasks are required to achieve an ultimate goal, especially in a comprehensive system and controller such as this. As we move from this non-linear system to another similar yet more complex one in the drones, it is exciting to see what we can do with the skills we have learned and the experience we have built. Overall, it was a good, successful lab, and we are excited for next week.