Lab 4: Model-based Position Control of a Cart

Adolfo Tec
Oladipo Toriola
Ajay Krishnan

13 October 2016

Lab Section 4: Thursday 8-11

MEC134/EEC128: Feedback Control Systems
Fall 2016

Prof. Seth Sanders, GSI Chen-Yu Chan

1. Purpose

The main objectives of this lab were to understand how to design different types of controllers based on given plant (cart) dynamics. This is done by implementing different position controllers such as a proportional and a PD controller using Quanser's QuaRC and a Q4-usb Acquisition board.

2. Pre-Lab

The plant in this lab is the same as in Lab 3. Its dynamics are given by:

$$(m_c r^2 R_m + R_m K_g^2 J_m)\ddot{x} + \left(K_t K_m K_g^2\right)\dot{x} = \left(r K_t K_g\right)V \tag{1}$$

where the parameters are described in Figure 1.

Parameter	Unit	Description
V	V	input voltage
m_c	kg	mass of the car
r	m	radius of the motor gears
R_m	Ω	resistance of the motor windings
K_t	Nm/A	torque motor constant
K_m	Vs/rad	back EMF constant
K_g	-	gearbox ratio
J_m	kg m^2	moment of inertia of the motor

Figure 1: Parameters of the cart system

A generalized second order linear system is described by the general differential equation:

$$\ddot{y} + 2\zeta\omega_n\dot{y} + \omega_n^2 y = bu(t) \tag{2}$$

The maximum overshoot of a second order system can be found analytically from the equation:

$$M_p = e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}} \tag{3}$$

and its rise time can be estimated from the equation:

$$\omega_n t_r \approx 1.76\zeta^3 - 0.417\zeta^2 + 1.039\zeta + 1 \tag{4}$$

2.1 Position Controller Design

We set the following performance objectives to be achieved by the feedback system for car position control (step response) for a step amplitude of 0.15m:

1. Rise time $t_r \leq 0.16\ s$ (ten times faster than Lab 3)
2. Maximum overshoot $M_p \leq 8\%$.

Question: Will the amplitude of the input affect rise time and maximum overshoot? Explain.

Answer: Given equations (3) and (4), we can see that the rise time and maximum overshoot are only a function of the system's (plant's) damping ratio, ζ, and its natural frequency ω_n. Therefore, the amplitude of the input will not affect rise time and maximum overshoot, as those parameters are characteristics of the plant alone.

2.1.1 Plant Model

Use your state space or transfer function representation of the system from Lab 3.

Question: Determine the poles of this transfer function. Is the plant stable?

Answer: From Lab 3, we found that the spate space representation of the system is given by:

$$\begin{bmatrix} \dot{x_1} \\ \dot{x_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\dfrac{K_f K_m K_g^2}{m_c r^2 R_m + R_m K_g^2 J_m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{r K_g K_t}{m_c r^2 R_m + R_m K_g^2 J_m} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u$$

The poles of this state space representation were found in MATLAB using the `pole` function and are given by: $p_1 = 0$ and $p_2 = -7.1774$. We know that plants are only stable if all the poles are in the open LHP, so this plant is only marginally stable as one of the poles is on the imaginary axis.

Question: Using the given performance objectives and equations (3) and (4), determine ranges for the desired values of ω_n and ζ.

Answer: From equation (3) and the given objective, we can solve for the range for ζ:

$$\ln(M_p) \leq e^{-\frac{\pi\zeta}{\sqrt{1-\zeta^2}}}$$

$$\text{or } \zeta \geq \sqrt{\frac{\ln(M_p)^2}{\pi^2 + \ln(M_p)^2}}$$

Plugging in our objective of $M_p \leq 8$, we get $\boxed{\zeta \geq 0.627}$

And for ω_n: $\omega_n \approx \frac{1.76\zeta^3 - 0.417\zeta^2 + 1.039\zeta + 1}{t_r}$ which is: $\boxed{\omega_n \geq 12.0}$

2.1.2 Proportional Controller

The first controller to attempt to meet the given objectives will be a proportional controller.

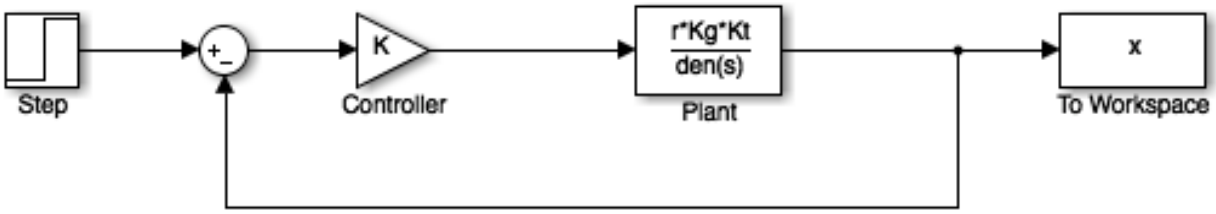


Figure 2: Simulink model for a proportional controller

The above Simulink diagram is the overall model's output is in meters while the input is in Volts. The plant in the diagram is given by:

$$H(s) = \frac{X(s)}{V(s)} = \frac{rK_gK_t}{\left(m_c r^2 R_m + R_m K_g^2 J_m\right)s^2 + \left(K_t K_m K_g^2\right)s}$$

To study the response of the proportional controller, we will vary the value of K over the range 10-60 and plot their responses on top of each other.

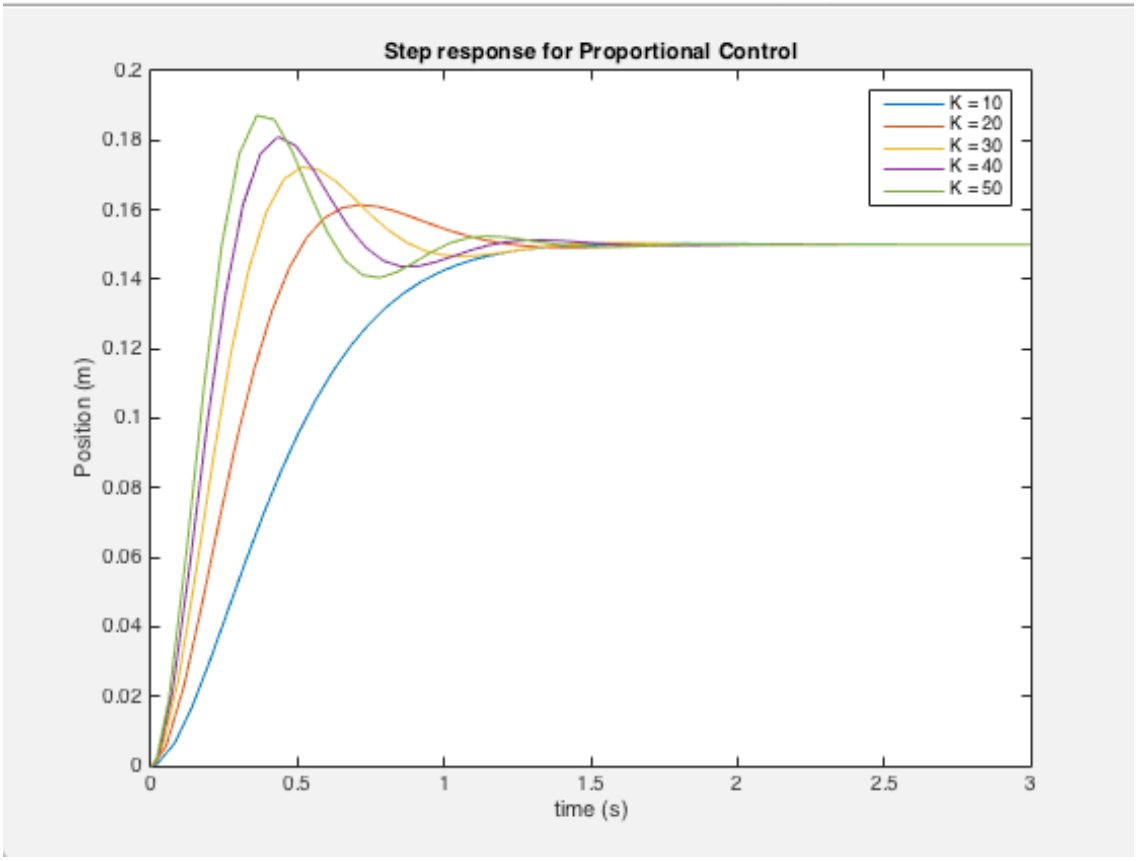


Figure 3: Step response for various K values of a proportional controller

From this plot, we can see that as K increases, the overshoot increases, but the rise time decreases. With just a proportional controller, the desired specifications cannot be achieved. This is verified in MATLAB by looking at the rise times and maximum overshoots for various K values.

K	Rise Time, t_r (s)	Maximum overshoot M_p
10	0.7178	0.1743
20	0.3477	7.6337
30	0.2476	14.8779
40	0.1960	20.6946
50	0.1681	24.8125

Figure 4: Rise times and maximum overshoots for plotted K values

Since both of the performance specifications cannot be met, we will find the smallest integer K value that meets the rise time performance specification and analyze its root locus and step response plot. We see that this happens after a K value of 51.

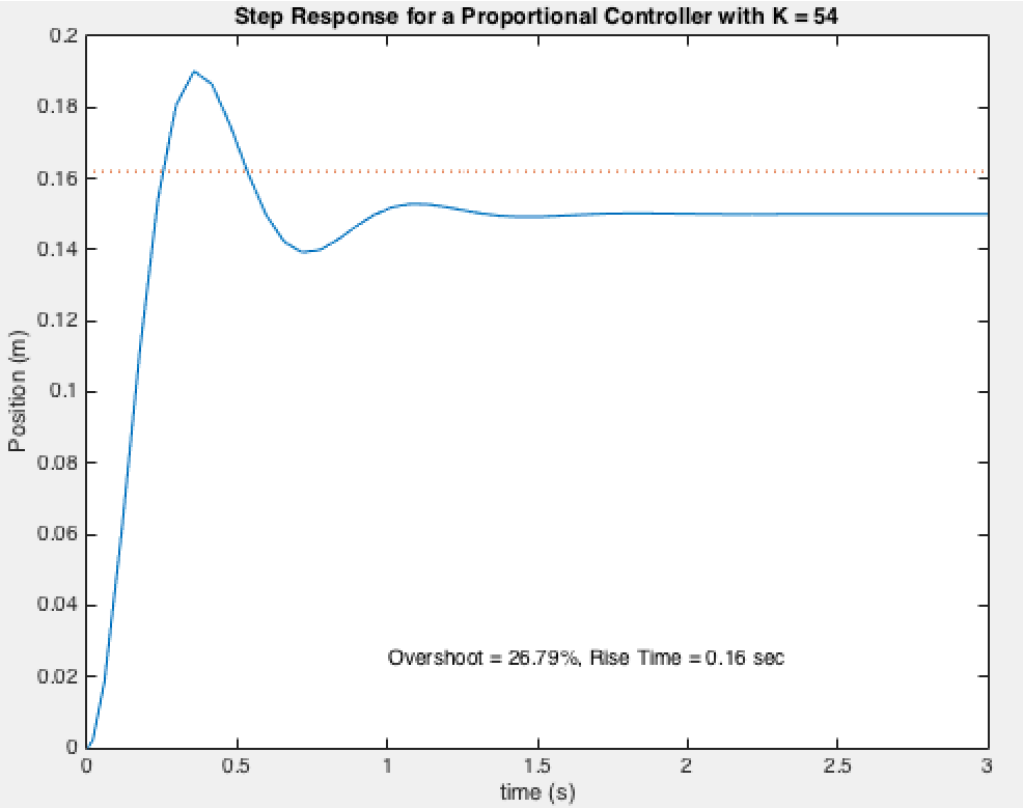


Figure 5: Step response for smallest integer K value that meets rise time specification

From the step response plot, we can see that the K value that meets the rise time specification does not meet the overshoot specification.

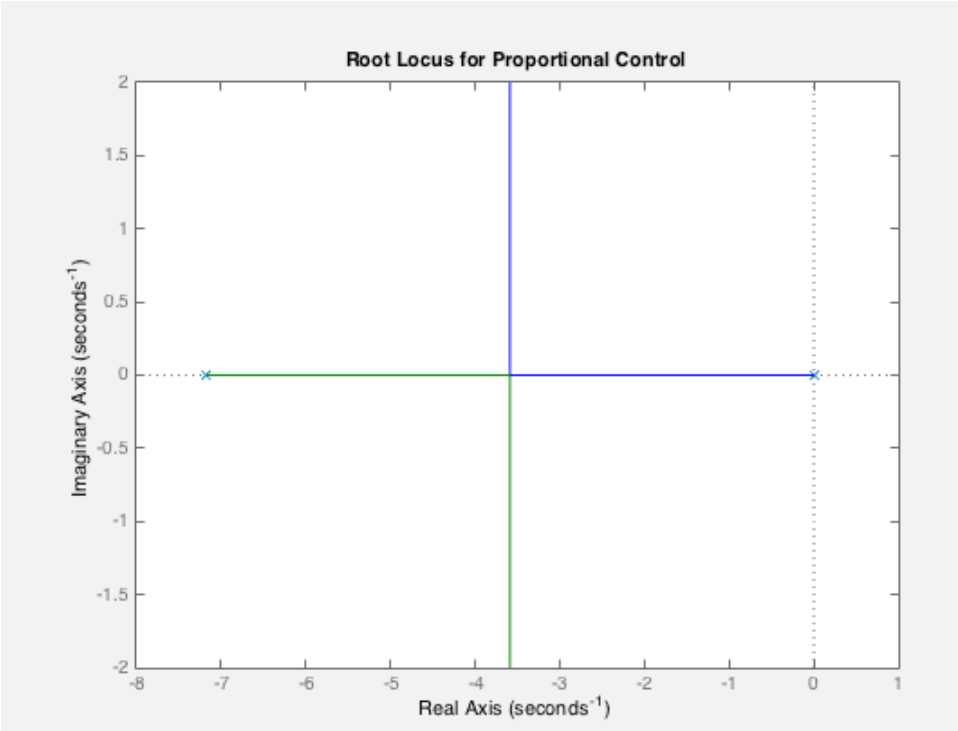


Figure 6: Root Locus plot using a proportional controller

Analyzing the root locus plot, we can see that for complex poles of the closed loop transfer function, as K increases, the radial distance increases while the angle θ, taken from the negative real axis, increases from 0º to 90º. We can plot the poles in the complex plane in terms of ω_n and ζ.

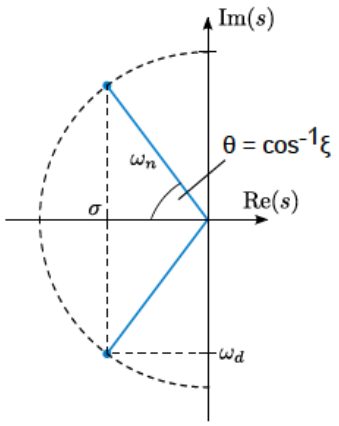


Figure 7: Location of poles in terms of ω_n and ζ

From this plot, we can see that as the radial distance increases, the natural frequency ω_n al)so increases, since ω_n is defined as the radial distance. Similarly, we can see that as θ increases, the damping ratio ζ decreases, since the cosine of an angle decreases as the angle increases. Therefore, as K increases, ω_n increases and ζ decreases.

2.1.3 PD Controller

In order to reduce the overshoot and meet the rise time and overshoot specifications while still maintaining proportional action, we will use derivative action in conjunction, or a PD controller. A PD controller has the form:

$$k(t) = k_p\left(x_{ref} - x\right) + k_D\left(\dot{x}_{ref} - \dot{x}\right) = k_P e(t) + k_D \dot{e}(t)$$

where $e(t) = x_{ref}(t) - x(t)$ is the error. A PD controller introduces a zero in the plant transfer function at $s = -k_P/k_D$. However, MATLAB doesn't allow us to put a pure zero. This is because introducing a pure zero is a bad idea because of the fact that noise will become more prevalent as a signal to the plant. Therefore, we will introduce a pole/zero pair with the pole so far away, that it hardly affects the rest of the system dynamics. Namely, we will choose to set the pole at s = -250, or a denominator of the controller as $\frac{1}{250}s + 1$. Furthermore, to use the derivative action in conjunction with the proportional action, we will set the zero at $s = -12$, or $\frac{k_P}{k_D} = 12$. Therefore, the controller has the following dynamics:

$$H(s) = k_D \frac{s + 12}{\frac{1}{250}s + 1}$$

We can then plot the root locus for the transfer function of the "modified" plant.

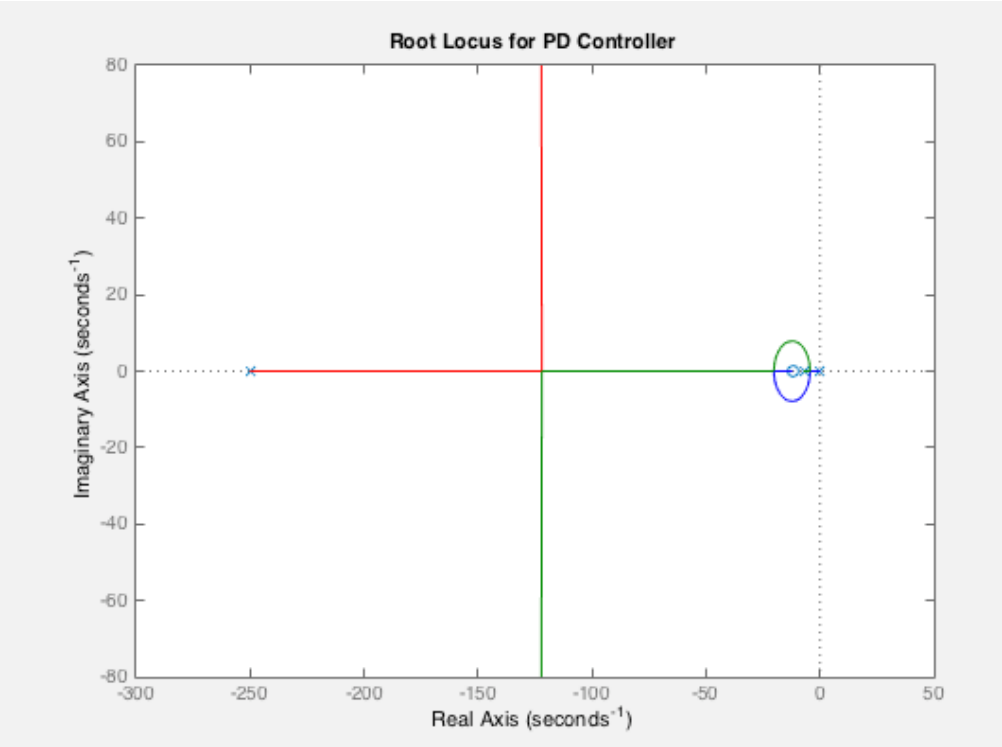


Figure 8: Root Locus plot using the PD controller

Comparing the root locus plot using a PD controller and the root locus plot using just a proportional controller, we can see that having a left half plane zero tends to pull the root locus towards it and that there exists a portion of the root locus which has complex closed loop poles and that $\omega_n$and ζ both increase as k_D increases. From the root locus plot using a PD controller, we can find a gain that meets the specifications given. One k_D value found that meets the criteria is $k_D = 8.08$, which corresponds to $\zeta = 0.803$ and $\omega_n = 12.8$.

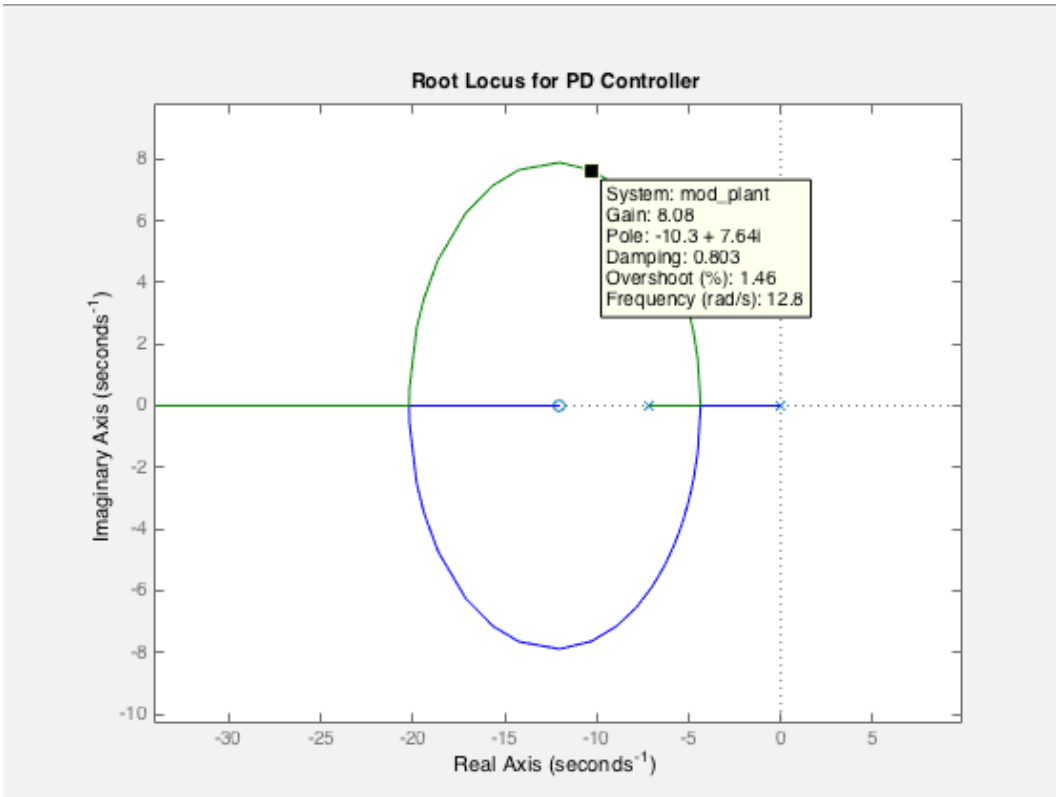


Figure 9: Zoomed in snapshot of root locus plot showing gain properties

The verification that the chosen value of $k_D = 8.08$ meets the criteria is done in MATLAB and Simulink. We find that the rise time associated with the k_D value is $t_r = 0.1055$s and the maximum overshoot is $M_P = 5.8383$.

From this analysis, we can see that the specifications can be met using a PD controller. Thus, we can find a range for which $k_D \leq 50$ for which the specifications are met. From MATLAB and Simulink, we find that the range for k_D that meets the specifications is: $5 \leq k_D \leq 50$. Four of these values' step response are plotted.

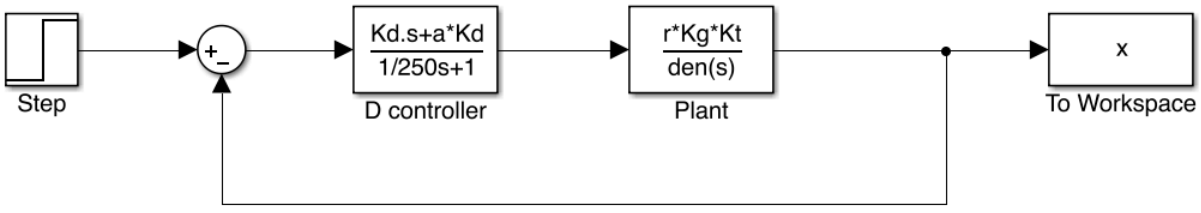


Figure 10: Simulink model using a PD controller

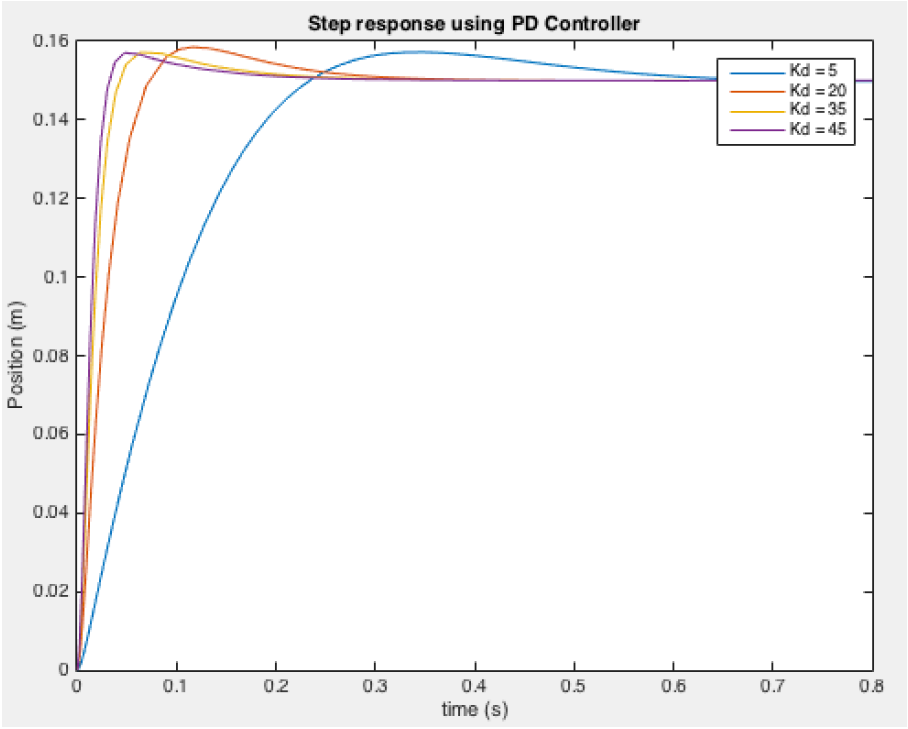


Figure 11: Step response for various k_D values within the found range that meet specifications

The rise times and maximum overshoots associated with these k_D values are shown below. We can see that all of the k_D values found within the range meet the given specifications.

k_D	Rise Time, t_r (s)	Maximum Overshoot, M_P
5	0.1590	4.8315
20	0.0464	5.6866
35	0.0264	4.7935
45	0.0203	4.6957

Figure 12: Rise times and maximum overshoots for k_D values within range

Now, we can test the effect of the zero by changing it to $\frac{k_P}{k_D} = 15$. By running a simulation in MATLAB, we can see that there is no value of k_D less than 50 that meets the specification.

3. Lab

3.1 Proportional Control

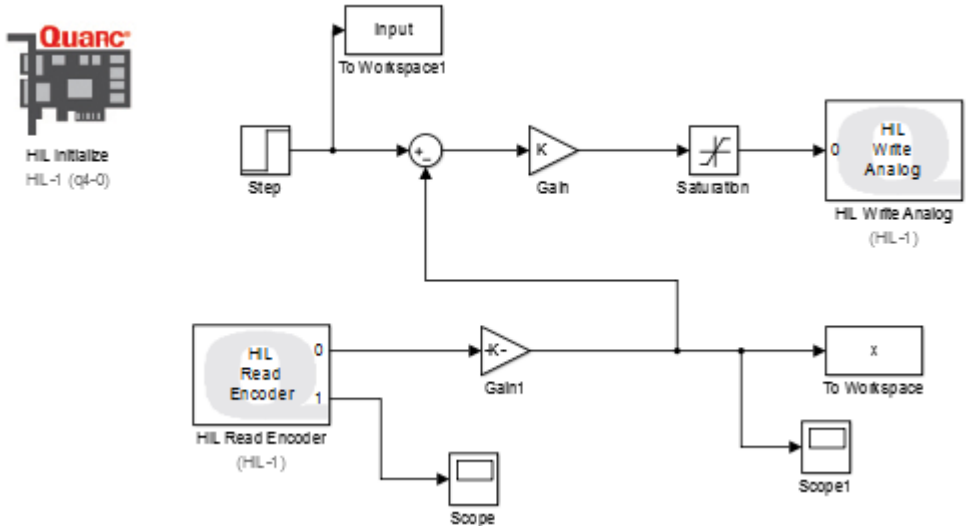


Figure 13: Block diagram for implementing proportional control on hardware

Through experimentation with the proportional controller, we can try various gain values and observe the variation of rise time and overshoot. We know from the pre-lab that a proportional controller alone cannot satisfy both the rise time specification and the maximum overshoot criteria at the same time, thus we will find the minimal K value such that the rise time is less than 0.16s, the maximal K value such that the overshoot is less than 8%, and a K value in between these two and plot their responses.

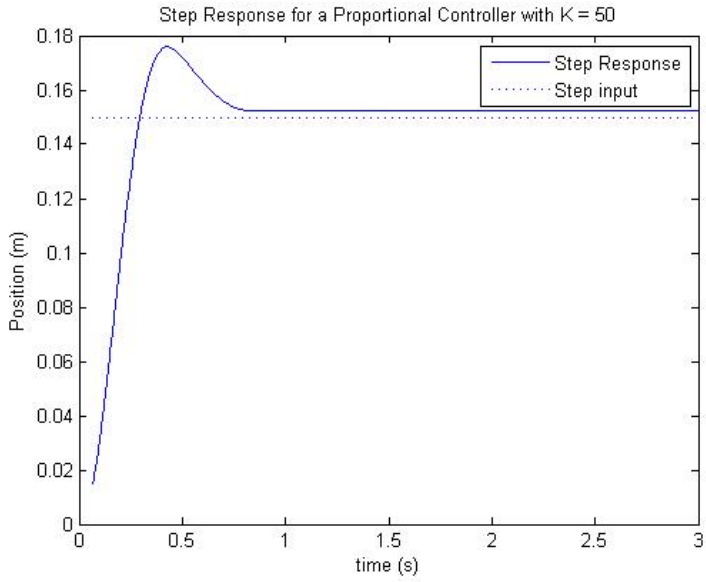


Figure 14: Minimal K value such that the rise time is less than 0.16s (K=50)

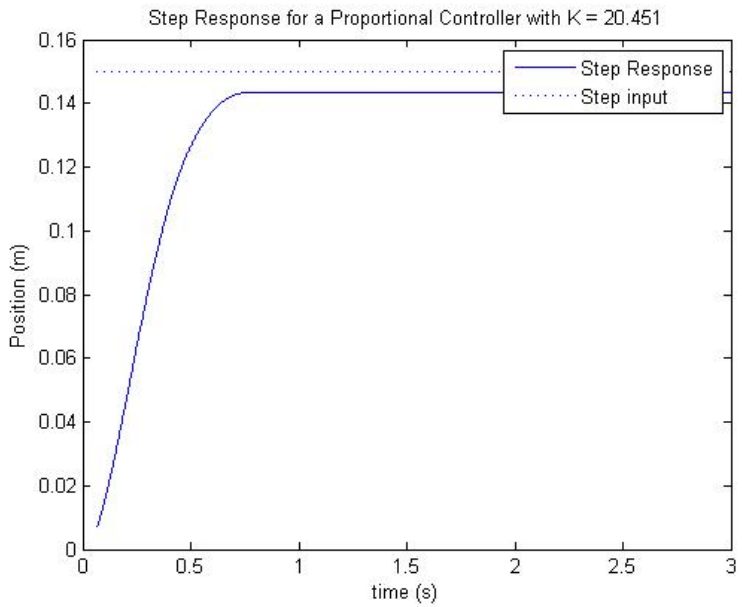


Figure 15: Maximal K value such that the overshoot is less than 8% (K=20.45)

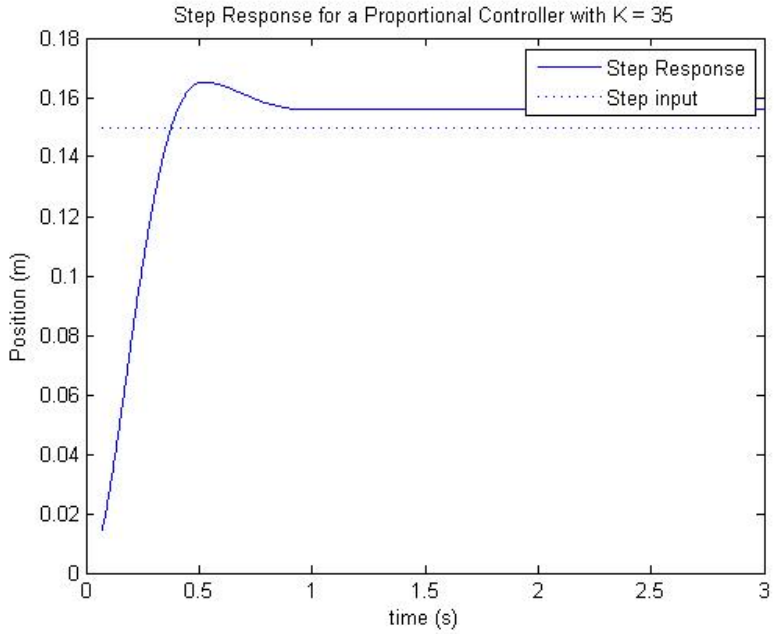


Figure 16: K value in between the rise time and overshoot criteria (K=35)

We can see from all of the responses that the system has a non-zero steady state error. This is most likely due to the plant model not being 100 percent representative of the actual dynamics of the cart.

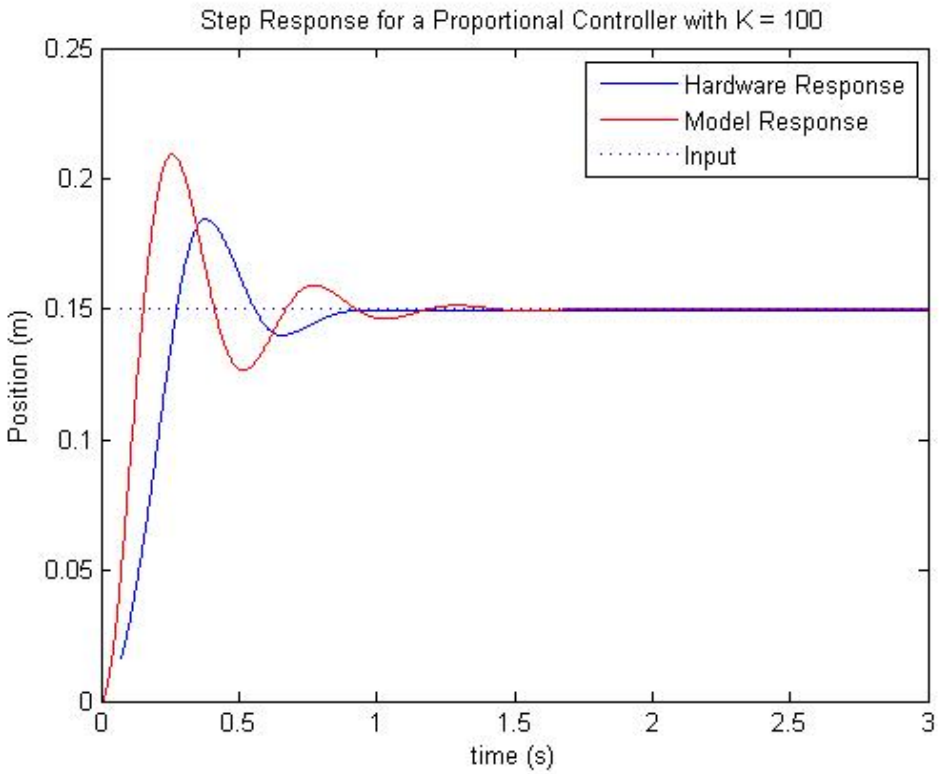


Figure 17: Comparison between the model and the hardware response with K = 100

Lastly, we compared the responses between the plant model and the actual hardware system. We can see that the actual hardware system responds slower than the plant model. This is expected as there is some natural error between the two. Namely, some reasons why this is the case would be because of the fact that the actual hardware system needs some time to actuate the data, the fact that there is a saturation block to protect the hardware system, and because of the fact that there is no 100% accurate plant model.

3.2 PD Control

Seeing as the proportional controller alone does not meet the specifications desired, we will implement a PD controller. The PD controller will be the controller from the pre-lab.

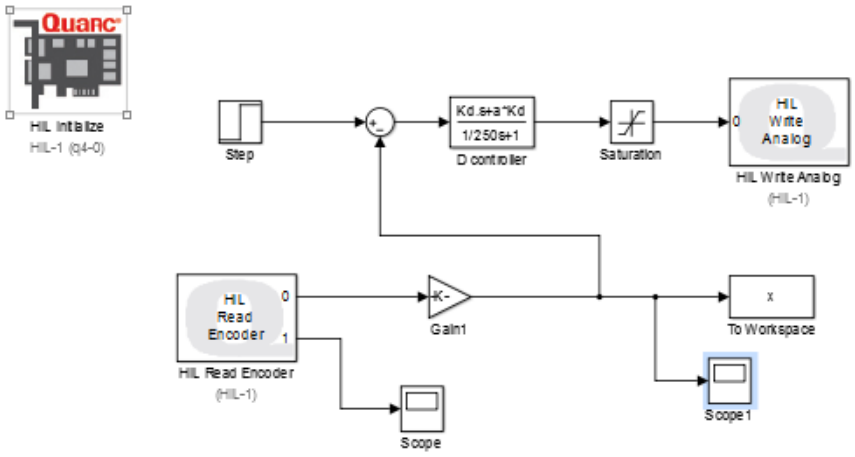


Figure 18: Block diagram for implementing PD controller on hardware

The value of the gain found in the pre-lab that was to meet the specification was $k_D = 8.08$. However, when actually using this gain value on the hardware system, it became evident that the gain did not meet the specifications. Therefore, we performed "intelligent" tuning of the gain values to try to meet the specifications. However, after multiple attempts of changing the gain within the range found, the specifications could not be meet on the hardware system. Ultimately, the gain values of $k_D = 25$ and $\frac{k_P}{k_D} = 12$ were used to attempt to meet the specifications. The rise time of the tuned controller gives a value of $t_r = 0.224s$.

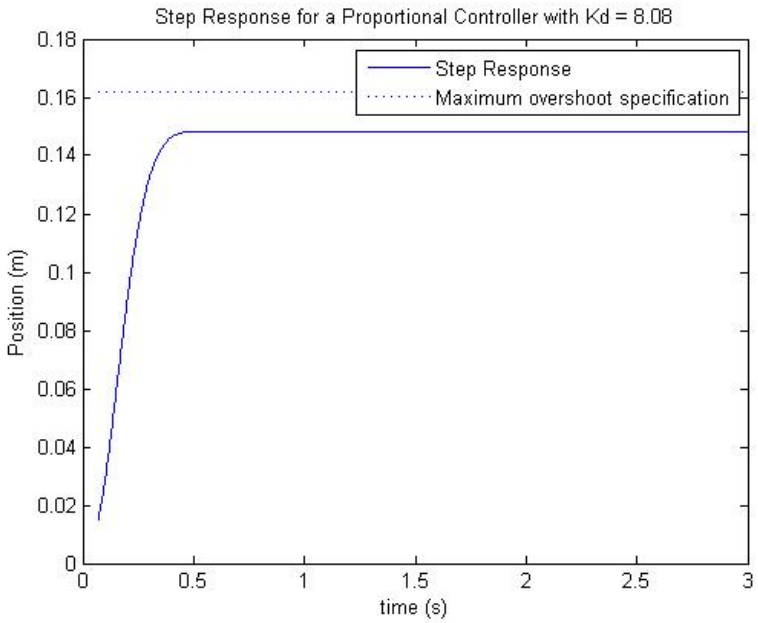


Figure 19: Initial test of PD controller with $k_D = 8.08$

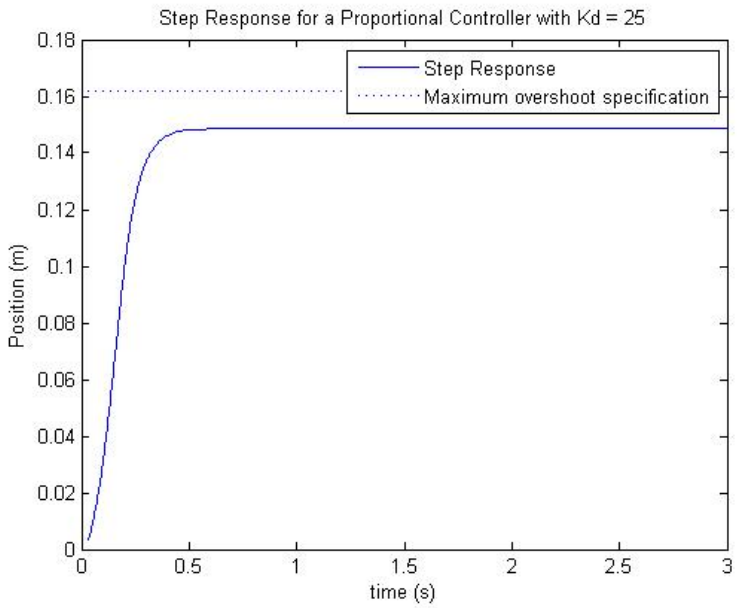


Figure 20: Tuned PD controller response from hardware system

3.3 Performance for Different Input Signals

Using the tuned PD controller from the previous part of the lab, the response for various input signals were examined. The following inputs were used for investigation:

- Pulse generator (50% pulse width, amplitude = 7.5 cm, period = 3s)
- Sine wave (amplitude = 10cm, frequency = 1 Hz)
- Sawtooth wave (amplitude = 5cm, frequency = 0.5 Hz)
- Exponentially-decaying sine wave (initial amplitude = 15cm, time constant = 0.25s)

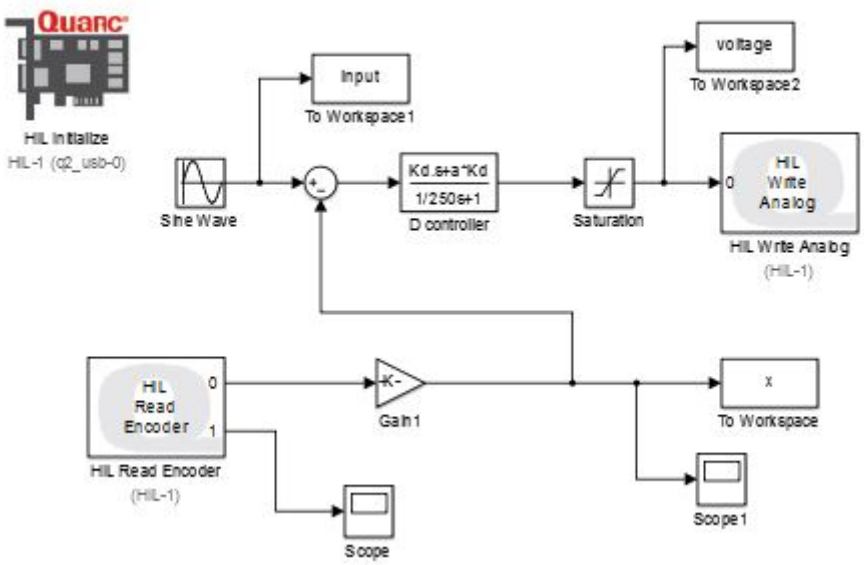


Figure 21: Block diagram for pulse generator input

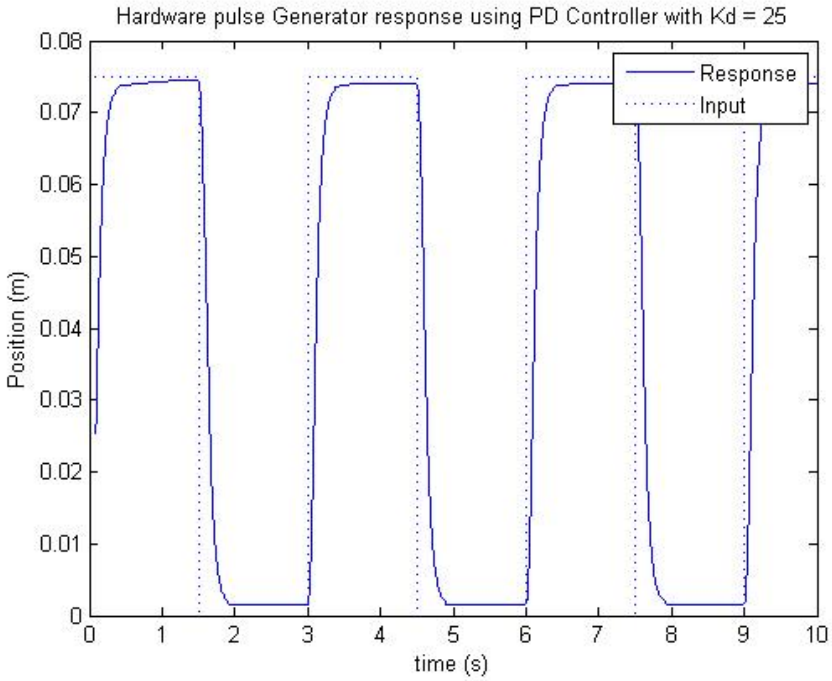


Figure 22: Hardware response for a pulse generator input

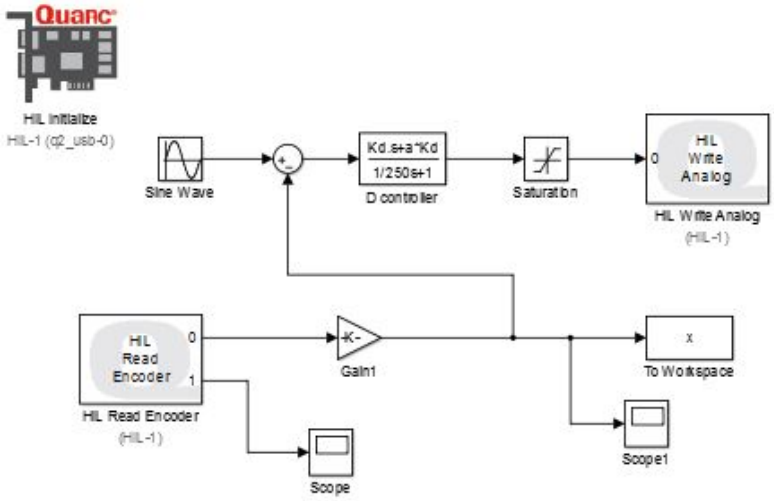


Figure 23: Block diagram for sine wave input

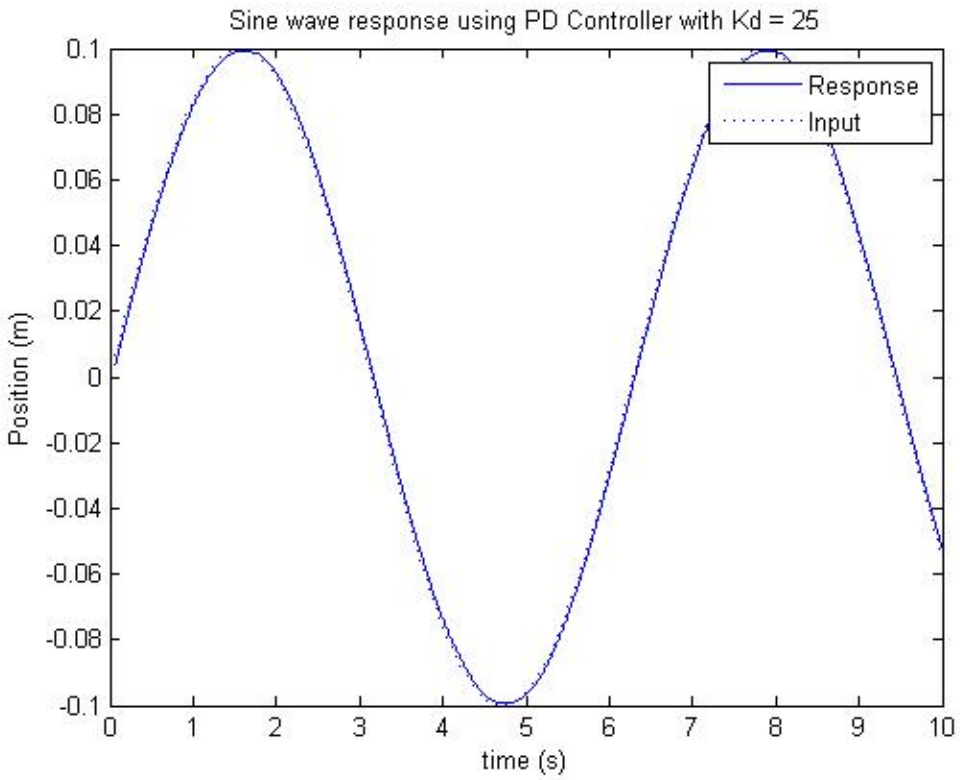


Figure 24: Hardware response for a sine wave input

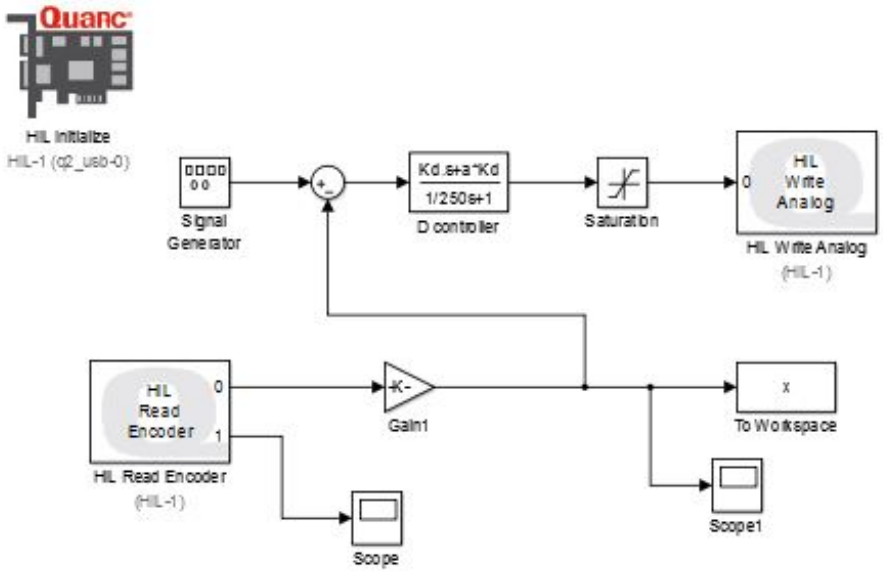


Figure 25: Block diagram for sawtooth wave input

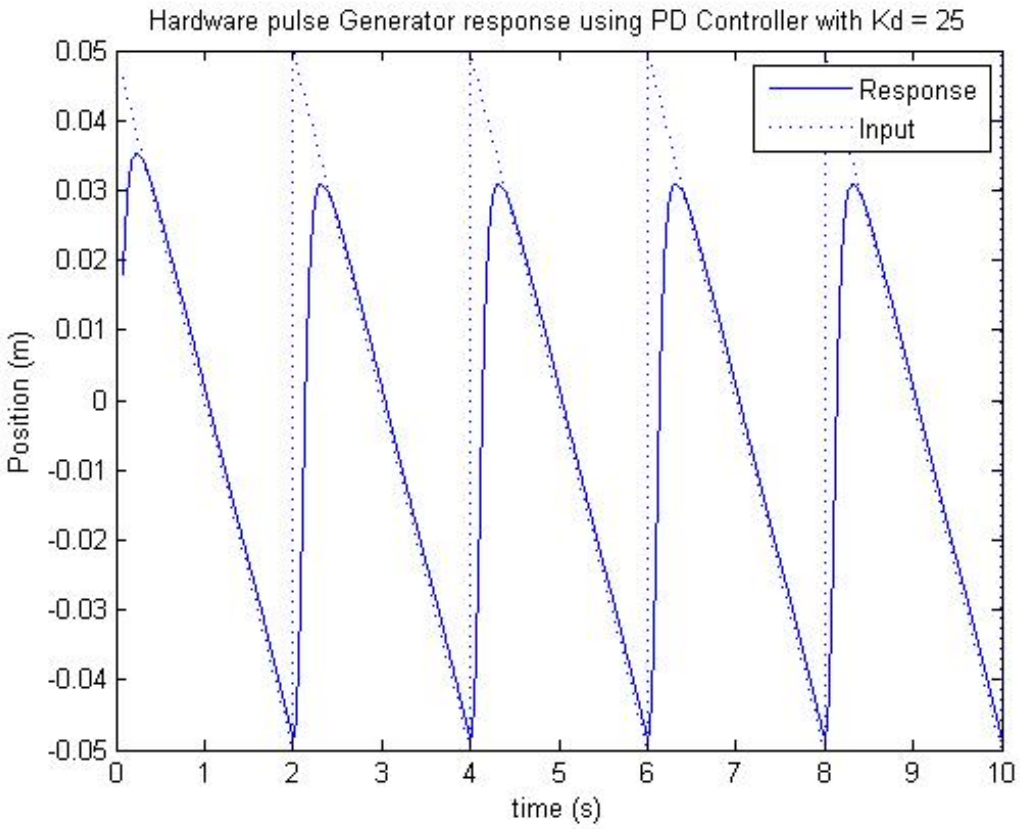


Figure 26: Hardware response for a sawtooth wave input

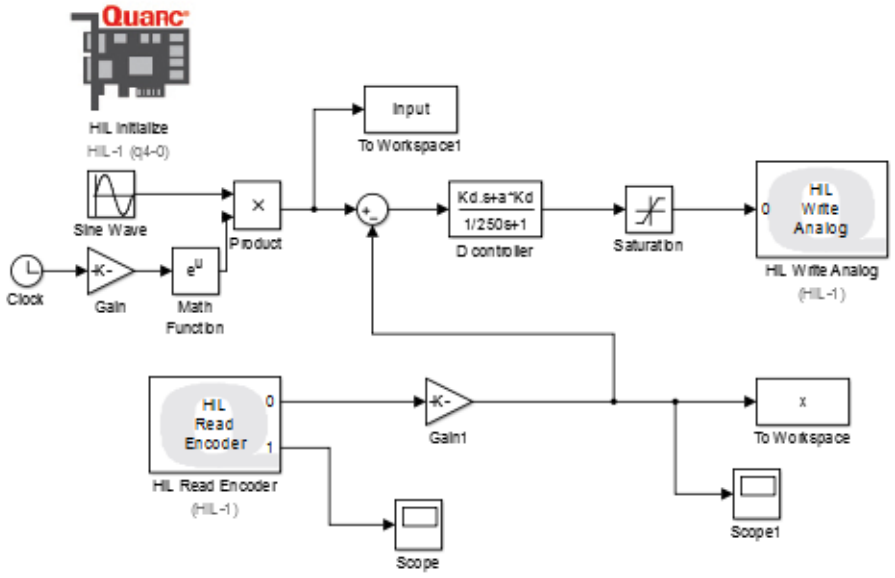


Figure 27: Block diagram for an exponentially-decaying sine wave input

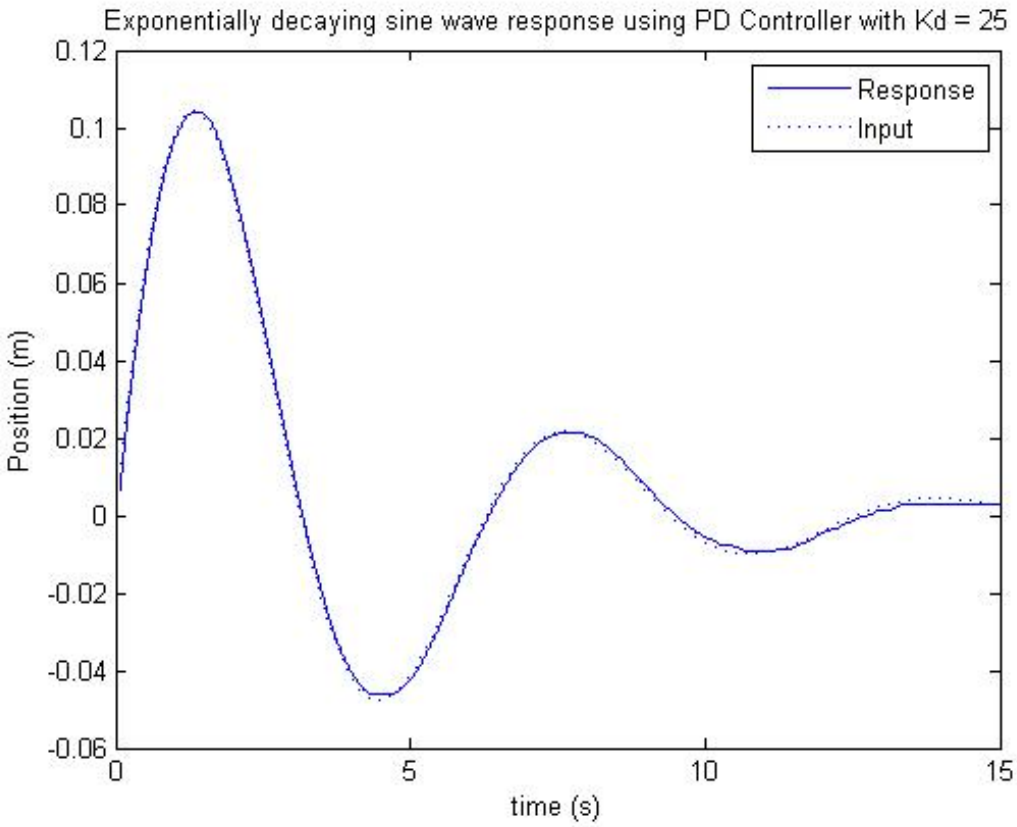


Figure 28: Hardware response for an exponentially-decaying sine wave input

Analyzing the hardware responses from the different input signals, we see that the controller performs well under certain kinds of inputs. Namely, the controller tracks continuous sinusoidal inputs, such as the sine wave input and the exponentially decaying sine wave input. The controller does not track inputs with discontinuities, such as the pulse generator input and the sawtooth wave input, too well. This is because of the fact that the controller and hardware need time to respond, and thus cannot track "jumps," or discontinuities, easily.

3.4 Performance for Limited Braking Ability

Suppose that we are limited in our ability to slow the cart, meaning that the negative voltage threshold is limited to -3V instead of -6V. The step response and the sine wave response are plotted to see how this affects tracking performance. It is evident that tracking performance is not changed when limiting the input.

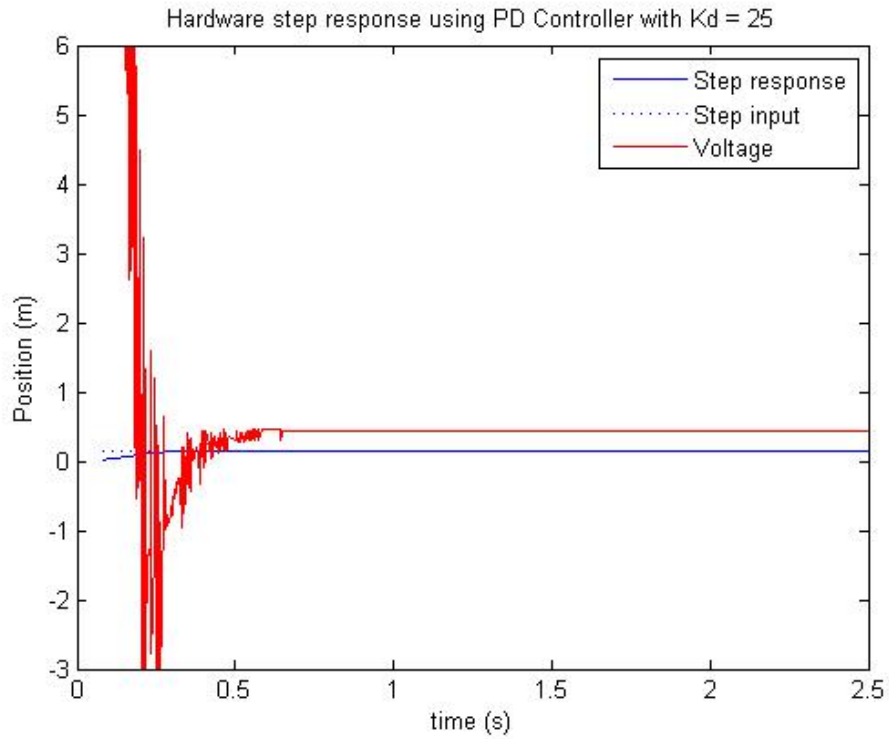


Figure 29: Step response with limited voltage

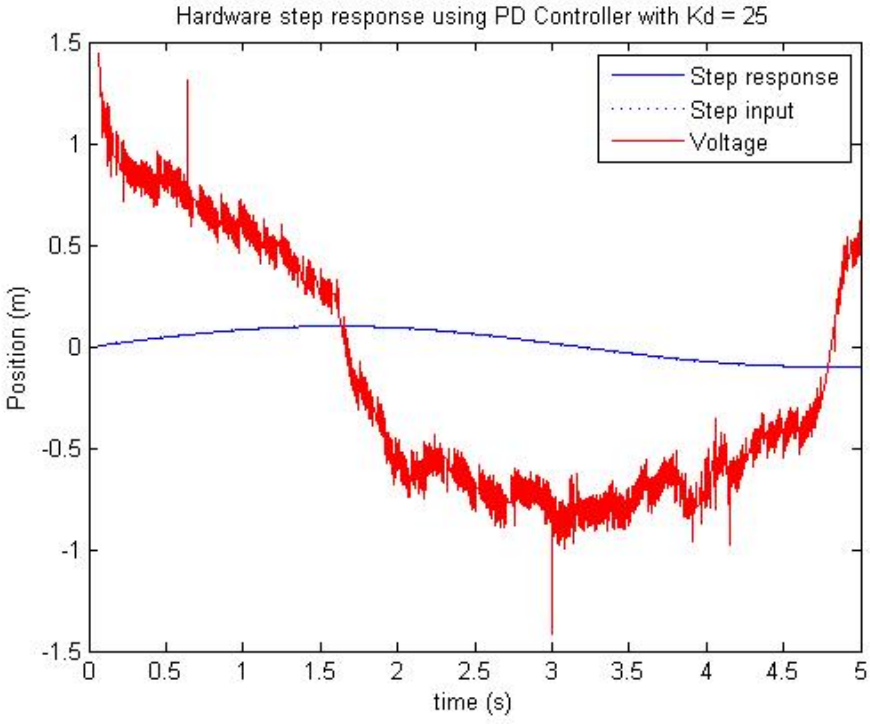


Figure 30: Sine wave response with limited voltage

Appendix: MATLAB Scripts

```
%% 4.1.1 Plant Model
mc = 0.94; r = 6.36e-3; Rm = 2.6; Kt = 7.67e-3; Km = 7.67e-3; Kg = 3.71; Jm =
3.9e-7;
tr = 0.16; Mp = 0.08;

A = [0 1; 0 -(Kt*Km*Kg^2)/(mc*r^2*Rm + Rm*Kg^2*Jm)];
B = [0; (r*Kg*Kt)/(mc*r^2*Rm + Rm*Kg^2*Jm)];
C = [1 0];
D = 0;
plant_sys = ss(A,B,C,D);

poles_sys = pole(plant_sys)

zeta = sqrt( ((log(Mp))^2)/(pi^2 + (log(Mp))^2))
wn = (1.76*zeta^3 - 0.417*zeta^2 + 1.039*zeta +1)/tr
```

A1: Pre-Lab 4.1 and 4.1.1

```
%% 4.1.2 P Controller
% Want tr <= 0.16 and Mp <= 8%
clear all; close all;

% Define Specification
tr_desired = 0.16;

% Choose a K range sample
K_sample = [10:10:60];
tr = [];
Mp = [];

figure
 for i = 1:length(K_sample)-1
     K = K_sample(i);
    sim('PL_4_1_2');
    stepresponse = stepinfo(x.data, x.time);
    tr = [tr stepresponse.RiseTime];
    Mp = [Mp stepresponse.Overshoot];
    plot(x.time, x.data); hold on;
 end

title('Step response for Proportional Control');
xlabel('time (s)'); ylabel('Position (m)');
legend('K = 10', 'K = 20', 'K = 30', 'K = 40', 'K = 50');

disp('Rise times'); tr
disp('Overshoot'); Mp

% As seen from the rise times, K = 50 didn't meet spec, so narrow down the
% sample range to find smallest integer value of K that meets spec.
K_sample = 51:60;
for i = 1:length(K_sample);
    K = K_sample(i);
    sim('PL_4_1_2');
    stepresponse = stepinfo(x.data, x.time);

    if stepresponse.RiseTime <= tr_desired
        break
    end
end

figure
plot(x.time, x.data)
title(['Step Response for a Proportional Controller with K = ' num2str(K)])
xlabel('time (s)'); ylabel('Position (m)');

disp(['The smallest interger value of K = ' num2str(K) ' gives a rise time: '
num2str(stepresponse.RiseTime) 's and overshoot: '
num2str(stepresponse.Overshoot)])

% Plot the root locus for the plant
figure
rlocus(plant_sys)
title('Root Locus for Proportional Control')
```

A2: Pre-Lab 4.1.2

```
%% 4.1.3 PD Controller
clear all; close all;
%Initialize Parameters
mc = 0.94; r = 6.36e-3; Rm = 2.6; Kt = 7.67e-3; Km = 7.67e-3; Kg = 3.71; Jm =
3.9e-7;
a = 12; %Kp/Kd
%%
% Plot root locus for new plant with PD Controller
figure
mod_plant = tf([1, a], [1/250, 1]) * tf([r*Kg*Kt], [mc*(r^2)*Rm+Rm*(Kg^2)*Jm,
Kt*Km*(Kg^2), 0]);
rlocus(mod_plant)
title('Root Locus for PD Controller');

% From our previous calculations, we see that zeta >= 0.627 and wn >= 12.0
% Looking at the root locus plot, we see that a gain of 8.08 meets this
% spec as zeta = 0.803 and wn = 12.8

% Verify that the Kd found meets spec
Kd = 8.08;
sim('PL_4_1_3')
stepresponse = stepinfo(x.data, x.time);
tr_response = stepresponse.RiseTime
Mp_response = stepresponse.Overshoot
```

A3: Pre-Lab 4.1.3

```
% Find range where Kd <= 50 for which specs are met
tr_desired = 0.16;
Mp_desired = 8;

Kd_sample = [50:-1:1];
k_range = [];

for i = 1:length(Kd_sample);
    Kd = Kd_sample(i);
    sim('PL_4_1_3');
    stepresponse = stepinfo(x.data, x.time);

    if stepresponse.RiseTime <= tr_desired && stepresponse.Overshoot <=
Mp_desired;
        k_range = [k_range, Kd];
    end
end
range = [min(k_range), max(k_range)];

%%
figure
% Plot 4 values found
for i = 1:4
    Kd = k_range(2*i); %Pick 4 random Kd gains
    sim('Pl_4_1_3');
    plot(x.time, x.data); hold on;
    legendInfo{i} = ['Kd = ' num2str(Kd)];
end
title('Step response for PD Controller');
xlabel('time (s)'); ylabel('Position (m)');
xlim([0 0.8])
legend(legendInfo);

% Change Kp/Kd to 15 and find a range of Kd that meet criteria if possible
a = 15;
Kd_sample = [50:-1:1];
k_range = [];

for i = 1:length(Kd_sample);
    Kd = Kd_sample(i);
    sim('PL_4_1_3');
    stepresponse = stepinfo(x.data, x.time);

    if stepresponse.RiseTime <= tr_desired && stepresponse.Overshoot <=5;
        k_range = [k_range, Kd];
    end
end

k_range
% No values of Kd will meet critera with this zero
```

A4: Pre-Lab 4.1.3 Finding and testing range

```
K = 50; % min K that satisfies tr
figure
plot(x.time, x.signals.values); hold on;
plot(input.time, input.signals.values, ':')
title(['Step Response for a Proportional Controller with K = ' num2str(K)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Step Response', 'Step input')

%%
K = 20.451; % max? value that satisfies MP
%run simulation
figure
plot(x.time, x.signals.values); hold on;
plot(input.time, input.signals.values, ':')
title(['Step Response for a Proportional Controller with K = ' num2str(K)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Step Response', 'Step input')

%%
K = 35; % In between
figure
plot(x.time, x.signals.values); hold on;
plot(input.time, input.signals.values, ':')
title(['Step Response for a Proportional Controller with K = ' num2str(K)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Step Response', 'Step input')

%%
K = 100;
%Run Simulation
figure
plot(x.time, x.signals.values); hold on

mc = 0.94; r = 6.36e-3; Rm = 2.6; Kt = 7.67e-3; Km = 7.67e-3; Kg = 3.71; Jm =
3.9e-7;
sim('PL_4_1_2');
plot(x.time, x.data, 'r')
plot(input.time, input.signals.values, ':')
title(['Step Response for a Proportional Controller with K = ' num2str(K)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Hardware Response', 'Model Response', 'Input');
```

A5: Lab 5.1 Proportional Control

```
%%
Kd = 8.08;
a = 12;
% Run hardware simulation
%%
figure
plot(x.time, x.signals.values); hold on
title(['Step Response for a Proportional Controller with Kd = ' num2str(Kd)])
xlabel('time (s)'); ylabel('Position (m)');

SS = 0.15;
Mp_desired = 0.08;
x1 = find(x.signals.values >= 0.1*SS, 1);
x2 = find(x.signals.values >= 0.9*SS, 1);
tr = x.time(x2) - x.time(x1) % Find rise time with associated gain

Mp = (SS + Mp_desired*SS)*ones(size(x.time));
plot(x.time, Mp, ':')
legend('Step Response', 'Maximum overshoot specification')



%%
Kd = 25; %Kd = 8.08, found in the pre lab did not work. Let's try tuning the
gain
a = 12;
% Run hardware simulation
%%
figure
plot(x.time, x.signals.values); hold on
title(['Step Response for a Proportional Controller with Kd = ' num2str(Kd)])
xlabel('time (s)'); ylabel('Position (m)');

SS = 0.15;
Mp_desired = 0.08;
x1 = find(x.signals.values >= 0.1*SS, 1);
x2 = find(x.signals.values >= 0.9*SS, 1);
tr = x.time(x2) - x.time(x1) % Find rise time with associated gain

Mp = (SS + Mp_desired*SS)*ones(size(x.time));
plot(x.time, Mp, ':')
legend('Step Response', 'Maximum overshoot specification')

% Could not get gain to meet specifications. Probably due to hardware
```

A6: Lab 5.2 PD Control

```
%%
Kd = 25;
a = 12;
tau = 0.25;
% Run simulation
% For exponentially decaying sine wave, used same values for regular sine
%%
figure
plot(x.time, x.signals.values); hold on
plot(input.time, input.signals.values, ':');
title(['Exponentially decaying sine wave response using PD Controller with Kd
= ' num2str(Kd)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Response', 'Input');
%%
figure
mc = 0.94; r = 6.36e-3; Rm = 2.6; Kt = 7.67e-3; Km = 7.67e-3; Kg = 3.71; Jm =
3.9e-7;
sim('Lab_5_3_Model')
plot(x.time, x.data); hold on;
plot(input.time, input.signals.values, ':')
title(['Model decaying sine wave response using PD Controller with Kd = '
num2str(Kd)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Response', 'Input');
```

A7: Lab 5.3 Performance for Different Input Signals

```
%%
Kd = 25;
a = 12;

% Run simulation
%%
figure
plot(x.time, x.signals.values); hold on
plot(input.time, input.signals.values, ':');
plot(voltage.time, voltage.signals.values, 'r--');
title(['Hardware step response using PD Controller with Kd = ' num2str(Kd)])
xlabel('time (s)'); ylabel('Position (m)');
legend('Step response', 'Step input');
```

A8: Lab 5.4 Performance for Limited Braking Ability

Lab 6a: Pole Placement for the Inverted Pendulum

Adolfo Tec
Oladipo Toriola

17 November 2016

Lab Section 4: Thursday 8-11

MEC134/EEC128: Feedback Control Systems
Fall 2016

Prof. Seth Sanders, GSI Chen-Yu Chan

1. Purpose

The main objectives of this lab is to achieve simultaneous control of both the angular position of the pendulum and the horizontal position of the cart on the track using full-state feedback. In this lab, we will be considering small angle perturbations and sine wave reference tracking of the cart position.

2. Pre-Lab

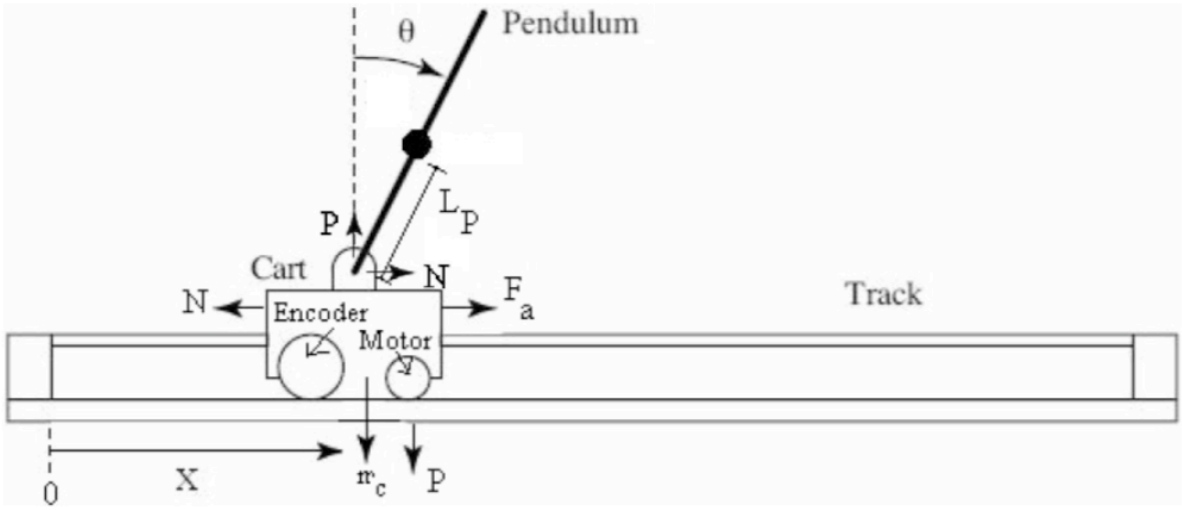


Figure 1: Free body diagram of the inverted pendulum setup (ignoring friction)

Equations of Motion of the Mechanical System

To effectively control the inverted pendulum system, as shown in Figure 1, we must understand the dynamics of the cart and the pendulum. First, we will find the equations of motion for the system and then create an appropriate controller. To find the equations of motion, we will consider the free-body diagrams of the cart and pendulum separately, as shown in Figure 2, and then combine them to find the equations of motion for the entire system.

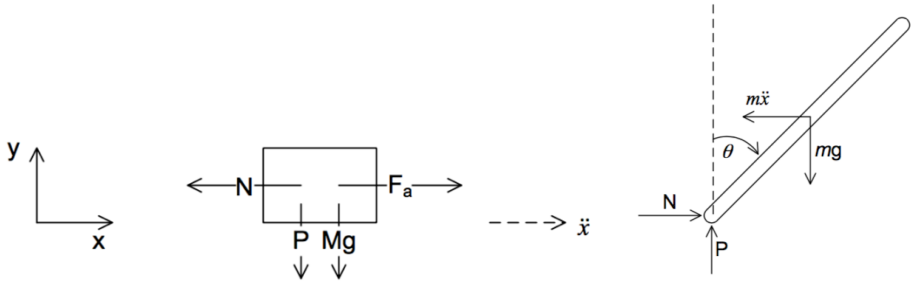


Figure 2: Free body diagram of the cart and pendulum shown individually

In finding the equations of motion, two assumptions were made. First, since we are only considering small angle perturbations, as stated in the purpose, we use the small angle approximation: $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. Secondly, we ignored friction and assumed that the

mass of the rod is uniformly distributed. After these assumptions were made, the following equations of motion were found:

$$(M+m)\ddot{x} + mL_p\ddot{\theta} = F_a \tag{1}$$

$$mL_p\ddot{x} + \frac{4mL_p^2}{3}\ddot{\theta} - mgL_p\theta = 0 \tag{2}$$

The parameter values of the physical system are given in Table 1.

Parameter	Value	Description
	439.6 counts/cm	Resolution of the cart position encoder
	651.9 counts/rad	Resolution of the angle encoder
M	0.94 kg	Mass of cart and motor
m	0.230 kg	Mass of pendulum
L_p	0.3302 m	Pendulum distance from pivot to center of mass
I_c	$m\,L_p^2/3$	Moment of inertia of pendulum about its center
I_e	$4m\,L_p^2/3$	Moment of inertia of pendulum about its end
K_t	$7.67 \cdot 10^{-3}$ Nm/A	Motor torque constant
K_m	$7.67 \cdot 10^{-3}$ Vs/rad	Motor back EMF constant
K_g	3.71	Motor gearbox ratio
R_m	2.6 Ω	Motor winding resistance
r	$6.36 \cdot 10^{-3}$ m	Radius of motor gear
J_m	$3.9 \cdot 10^{-7}$ kg m^2	Motor moment of inertia

Table 1: Parameters of the inverted pendulum setup

Full System Dynamics of Linearized System

To find the full system dynamics of the linearized system, we must first note that our system is a Single Input Multiple Output (SIMO) system as we are trying to control both the position of the part and the angle of the pendulum by using only the motor voltage. From previous analysis of the cart system, we found that the applied force is given by:

$$F_a = \frac{K_g K_t}{R_m r}V - \frac{K_g^2 K_t K_m}{R_m r^2}\dot{x} - \frac{J_m K_g^2}{r^2}\ddot{x} \tag{3}$$

Substituting Equation 3 into Equations 1 and 2 and solving for $\ddot{\theta}$ and $\ddot{x}$, we can obtain the state-space model of the inverted pendulum system in the form $\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$, where

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} \qquad \boldsymbol{x} = \begin{bmatrix} x \\ \dot{x} \\ \theta \\ \dot{\theta} \end{bmatrix} \qquad A = \begin{bmatrix} 0 & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & 0 & a_{34} \\ 0 & a_{42} & a_{43} & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ b_4 \end{bmatrix} \qquad u = V$$

and the elements of A and B are defined as:

$$a_{12} = a_{34} = 1$$

$$a_{22} = -\left(\frac{1}{M+\frac{m}{4}+\frac{K_g^2 J_m}{r^2}}\right)\left(\frac{K_g^2 K_t K_m}{r^2 R_m}\right)$$

$$a_{23} = -\frac{3}{4}\left(\frac{mg}{M+\frac{m}{4}+\frac{K_g^2 J_m}{r^2}}\right)$$

$$a_{42} = \frac{3}{4L_p}\left(\frac{1}{M+\frac{m}{4}+\frac{K_g^2 J_m}{r^2}}\right)\left(\frac{K_g^2 K_t K_m}{r^2 R_m}\right)$$

$$a_{43} = \left(1+\frac{3}{4}\left(\frac{m}{M+\frac{m}{4}+\frac{K_g^2 J_m}{r^2}}\right)\right)\left(\frac{3g}{4L_p}\right)$$

$$b_2 = \left(\frac{1}{M+\frac{m}{4}+\frac{K_g^2 J_m}{r^2}}\right)\left(\frac{K_g K_t}{r R_m}\right)$$

$$b_4 = \left(-\frac{3}{4L_p}\right)\left(\frac{1}{M+\frac{m}{4}+\frac{K_g^2 J_m}{r^2}}\right)\left(\frac{K_g K_t}{r R_m}\right)$$

Analysis and Controller Design

Using MATLAB, we use eig command to find the poles of our system (i.e. the eigenvalues of our A matrix). These eigenvalues were found to be:
$\lambda_1 = 0, \lambda_2 = -7.5515, \lambda_3 = -4.129, \lambda_4 = 4.8682$. Since there is a positive real pole (i.e. since λ_4 is in the RHP), the system is not BIBO stable.

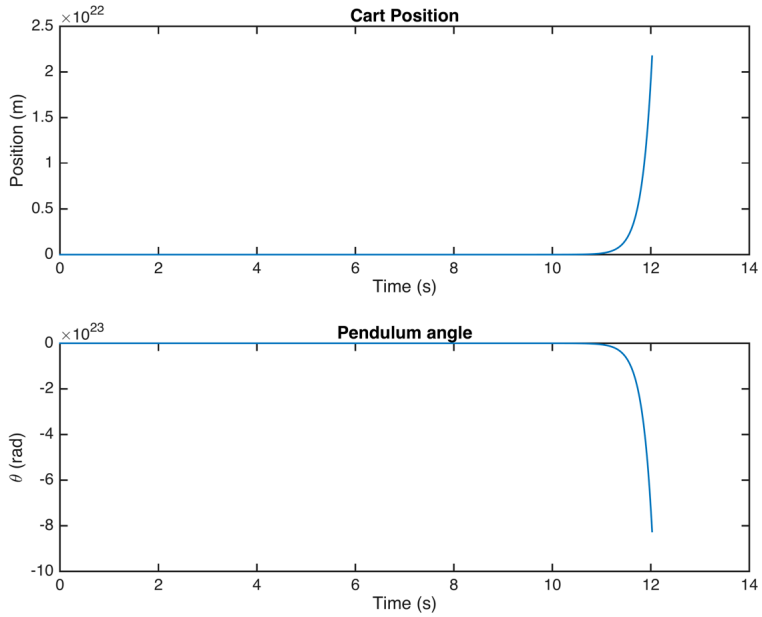


Figure 3: System's response for a step input in the motor voltage

A simulation of the response of the system for a step input in the motor voltage is shown in Figure 3. We see that both the outputs of the cart position and the pendulum angle do not converge to a steady-state value, and instead go towards positive and negative infinity, respectively. From the simulation, we verify that the system is unstable. However, there are discrepancies between the simulation and the actual physical system as the pendulum angle would not actually exponentially decrease to negative infinity in the physical system. Instead, the pendulum would just stay at a constant angle, or even rotate. This discrepancy stems from the small angle approximation assumption made when deriving the equations of motion for the system.

To achieve the desired performance specifications for the system, we will implement the full state-feedback controller $u = -K\boldsymbol{x}$ where K is of the form $K = [k_1 \quad k_2 \quad k_3 \quad k_4]$. The desired closed-loop eigenvalues desired to meet our performance specifications are given by $s_{1,2} = -2.0 \pm 10j$ and $s_{3,4} = -1.6 \pm 1.3j$. The closed-loop matrix is given by:

$$A_K = A\text{-}BK = \begin{bmatrix} 0 & a_{12} & 0 & 0 \\ -b_2k_1 & a_{22}\text{-}b_2k_2 & a_{23}\text{-}b_2k_3 & -b_2k_4 \\ 0 & 0 & 0 & 1 \\ -b_4k_1 & a_{42}\text{-}b_4k_2 & a_{43}\text{-}b_4k_3 & -b_4k_4 \end{bmatrix}$$

The characteristic polynomial was then found by computing $P(K;s) = \det(sI\text{-}A_K)$ and is given by:

$$\begin{aligned} s^4 + (-a_{22} + b_2k_2 + b_4k_4)s^3 + (-a_{43} + b_4k_3 + a_{12}b_2k_1\text{-}a_{22}b_4k_4 + a_{42}b_2k_4)s^2 \\ + (a_{22}a_{43}\text{-}a_{23}a_{42}\text{-}a_{22}b_4k_3 + a_{23}b_4k_2 + a_{42}b_2k_3\text{-}a_{43}b_2k_2)s \\ + (a_{12}a_{23}b_4k_1\text{-}a_{12}a_{43}b_2k_1) \end{aligned} \tag{4}$$

Next, we found the desired characteristic polynomial given by:

$$P_{des}(s) = \prod_{i=1}^{4}(s\text{-}s_i) = s^4 + \frac{36}{5}s^3 + \frac{2421}{20}s^2 + \frac{1749}{442}s \tag{5}$$

Finally, comparing Equations 4 and 5 and using the MATLAB command acker, we find that the K matrix yields the following solution:

$$K = [\text{-}13.0283 \quad \text{-}14.7848 \quad \text{-}48.1649 \quad \text{-}6.6214] \tag{6}$$

Finally, we implement the controller $U = K(r\text{-}x)$ where r is the reference input and find that the transfer function is given by:

$$G(s) = \frac{\text{-}19.8367s^2 + 442}{s^4 + 7.2s^3 + 121.05s^2 + 349.8s + 442} \tag{7}$$

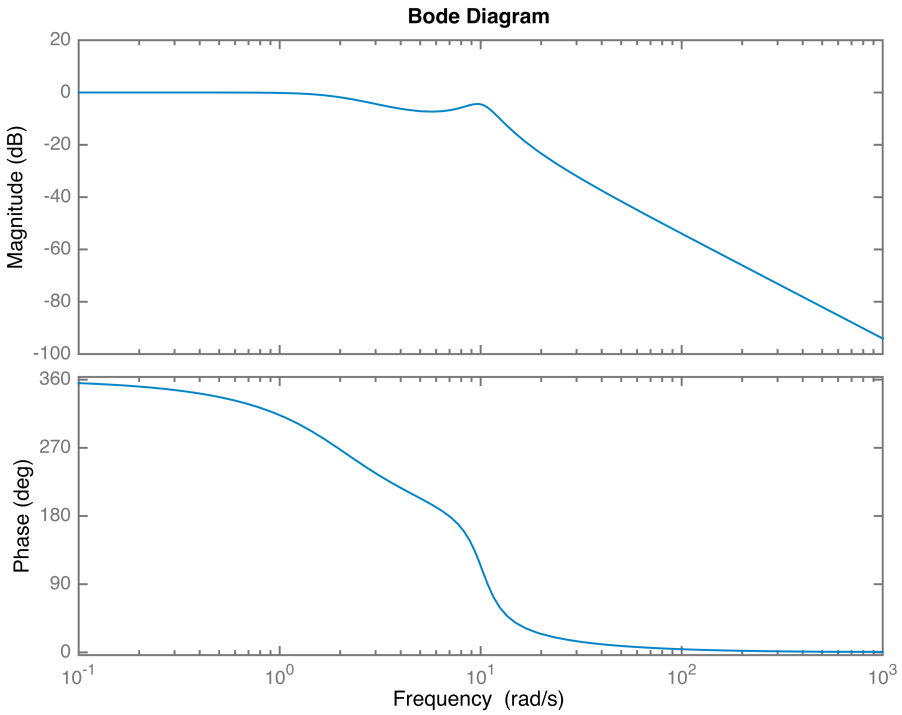


Figure 4: Bode plot of the transfer function

3. Lab

In this lab, we will be using the state-feedback controller found using the Quanser hardware and I/O blocks in Simulink.

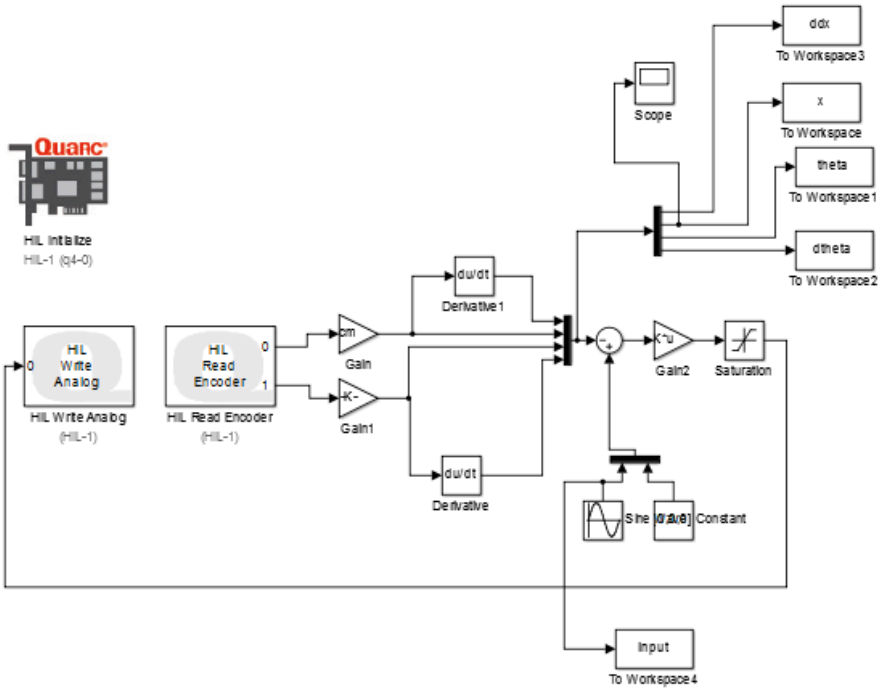


Figure 5: Block diagram of system using full state-feedback controller in Simulink

In Figure 5, the gain cm is used to convert the encoder values to meters and its value is cm = 1/42958 m/count, while the gain in Gain 1 is used to convert the encoder values to radians and is given by -1/651.8 radians/count. The gain in Gain 2 is the gain matrix K found in the pre-lab. Finally, the saturation block was used for gear protection and was set to $\pm 6V$.

When implementing this controller on the hardware, we first set the reference **r** to $[0 \quad 0 \quad 0 \quad 0]^T$ and made sure it balanced at the equilibrium point. The hardware reacted as expected when the system was initially set to the equilibrium position as it did not move. Once the pendulum was balanced *small* perturbations were manually applied to the pendulum to see how the system responded. With small perturbations applied to the top of the pendulum in a certain direction, the pendulum responded by accelerating in the same direction to bring the pendulum back to its equilibrium position. This is so that the cart will return to its equilibrium position. The hardware oscillates around its equilibrium point due to the fact the the controller tries to get the cart and pendulum as close to its reference position as possible. When the cart nears its equilibrium position, it causes the pendulum to rotate and thus now the pendulum is not in equilibrium. The cart moves in order to compensate and catch the pendulum, and not the cart is not in its equilibrium position. The cycle continues.

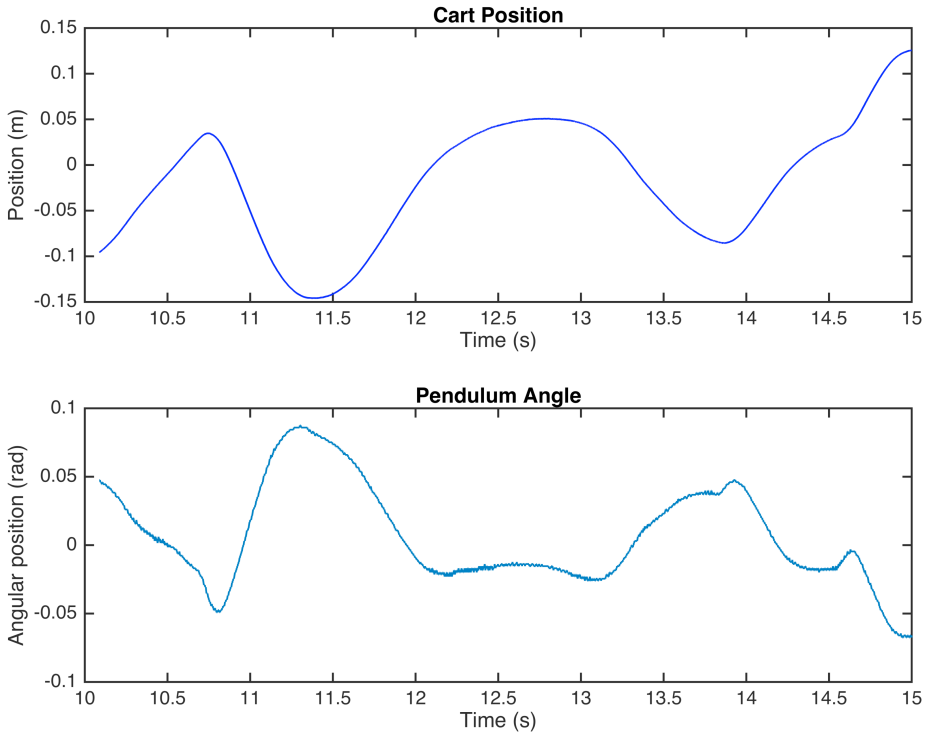


Figure 6: Cart position and pendulum angle response to small perturbations

Next, to look at the response of the system to different inputs, we will introduce a sine wave reference signal so that the reference input will be $\boldsymbol{r} = [Msin\omega t \quad 0 \quad 0 \quad 0]^T$ where M is the amplitude = 0.1m, and the frequency ω is changed with $\omega = 1,2,5$ rad/s.

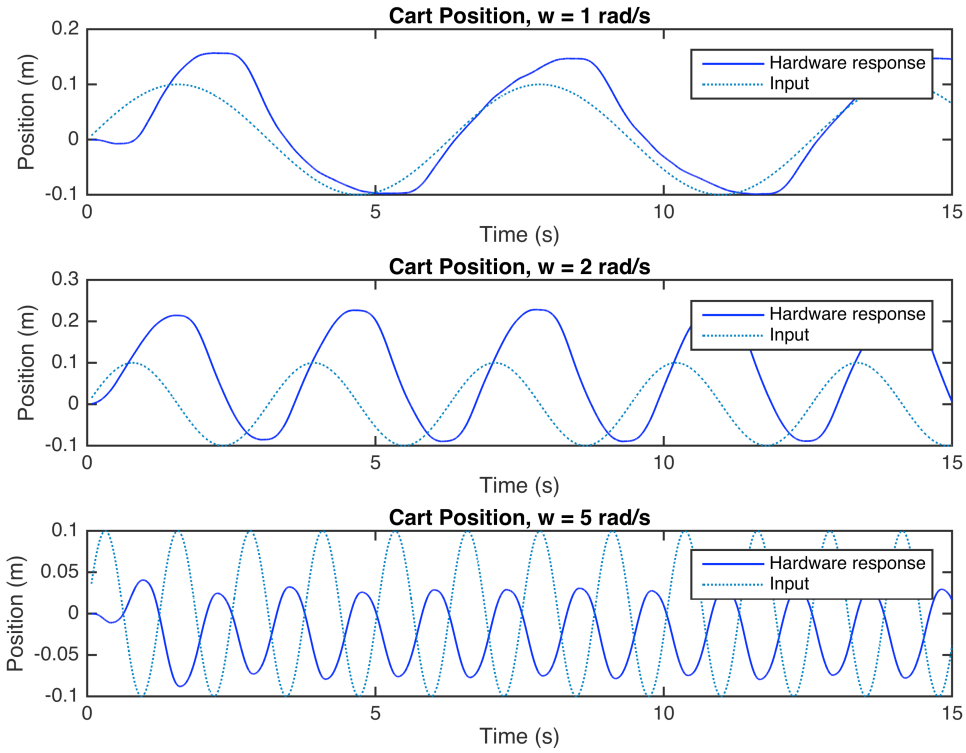


Figure 7: Cart position response for various frequency inputs

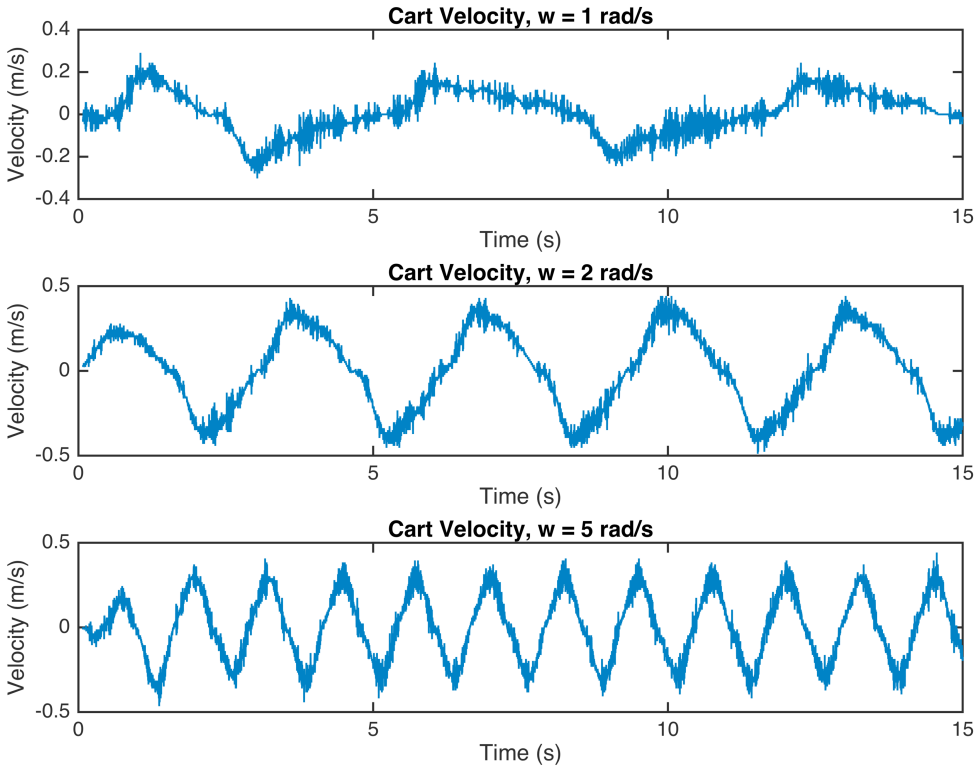


Figure 8: Cart velocity response for various frequency inputs

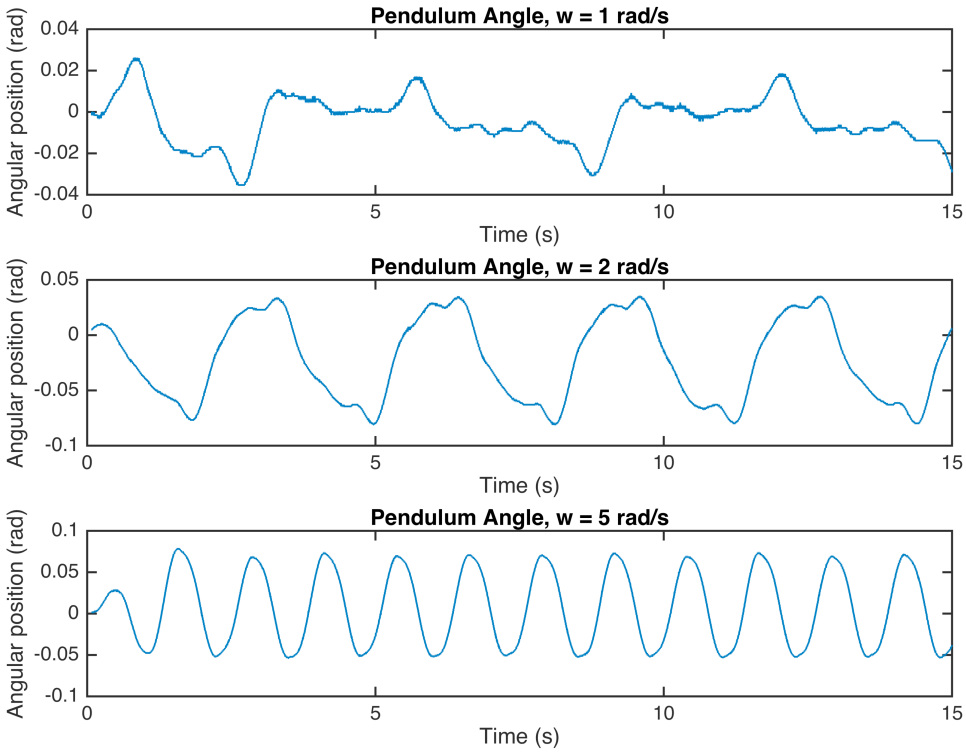


Figure 9: Pendulum angle response for various frequency inputs

Analyzing Figure 7, we see that the response of the system is a sinusoid with various phase shifts and amplitudes depending on the input frequencies. The gain and phase were found in MATLAB (see Appendix) and are compared to the values from the bode plot in Figure 4.

Frequency [rad/s]	Actual Gain [dB]	Actual Phase [degrees]	Simulated gain [dB]	Simulated Phase [degrees]
1	1.2781	35.2486	0.98	313.3
2	1.6176	174.6386	0.81	267.9
5	0.6433	325.5769	0.44	204.1

Table 2: Comparison of actual gain and phase with simulated values

Looking at Table 2, we can see that the gains and phases from the hardware do not completely match the simulated gains and phases from the bode plot. The hardware gains are larger than the simulated gains because the system needs to move farther than predicted to correct the pendulum angle due to nonlinearities. We also see that the hardware phase increases as opposed to the decreasing predicted phase. This is due to the terms not represented in the model such as friction, real angles, and other variables unaccounted for.

Finally, we will see how the response of the system changes when the closed-loop poles are slightly altered.

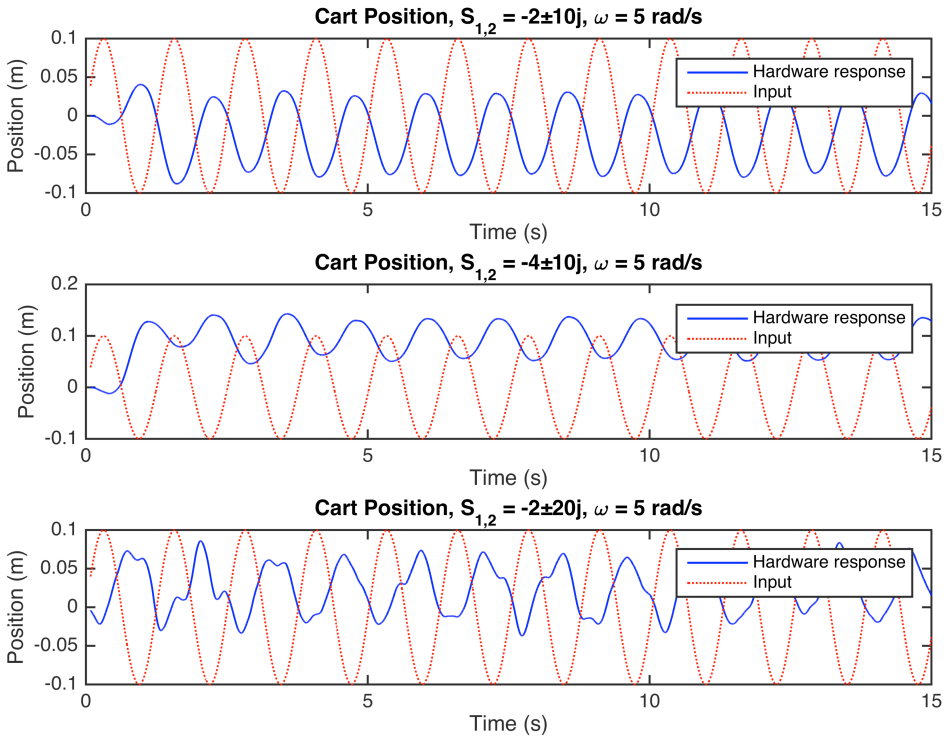


Figure 10: Cart position response for various closed-loop poles

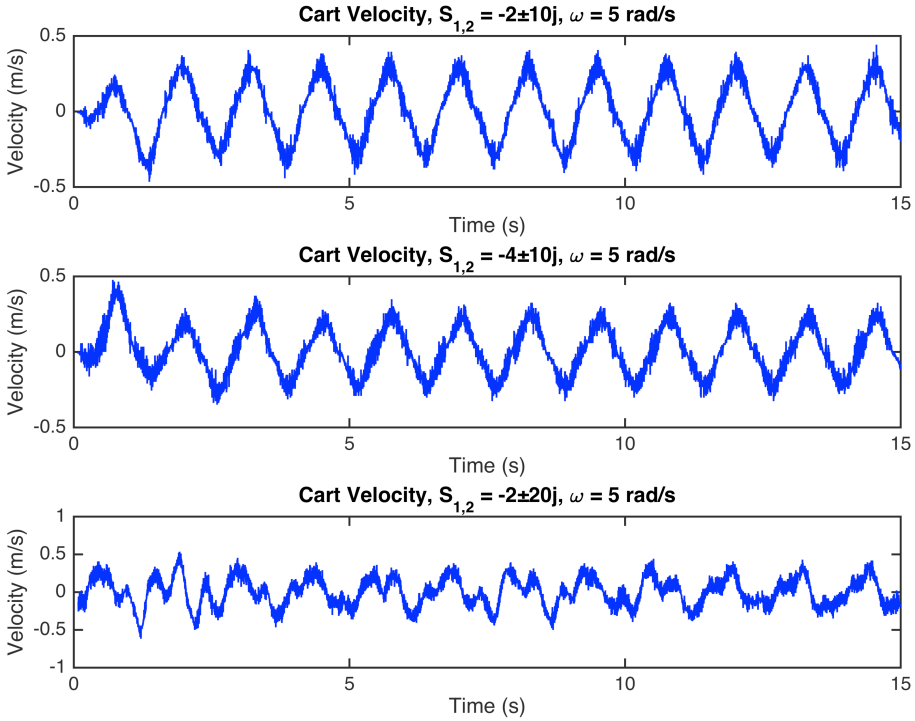


Figure 11: Cart velocity response for various closed-loop poles

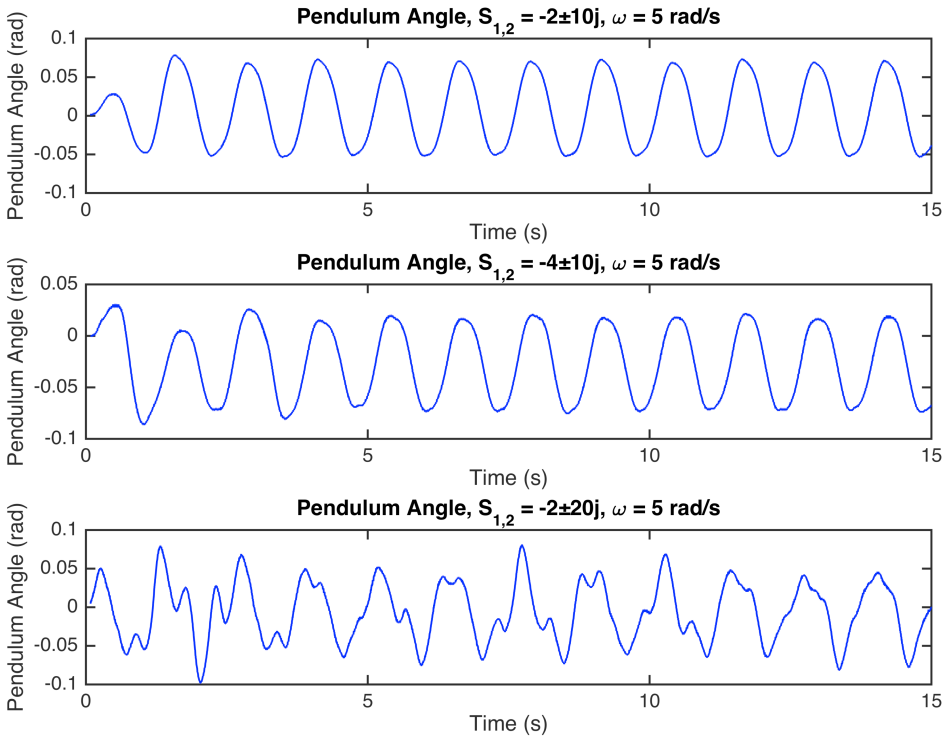


Figure 12: Pendulum angle response for various closed-loop poles

The first plot of each subplot shows the original response from the desired closed-loop pole used for the majority part of the lab. The first change made was to change the real part of the closed-loop pole by a factor of 2. With this change, we see that the amplitude and the phase shift of the response is altered. We see that the amplitude is much smaller than the original response. In fact, we see that the cart position response does not oscillate around the equilibrium position. The next change made was to keep the real part fixed, while doubling the imaginary component of the desired closed-loop pole. From Figures 10,11, and 12, we see that the response exhibits odd behavior as there are more oscillations. We see that the cart oscillates at a higher position with about the same amplitude as the unchanged pole.

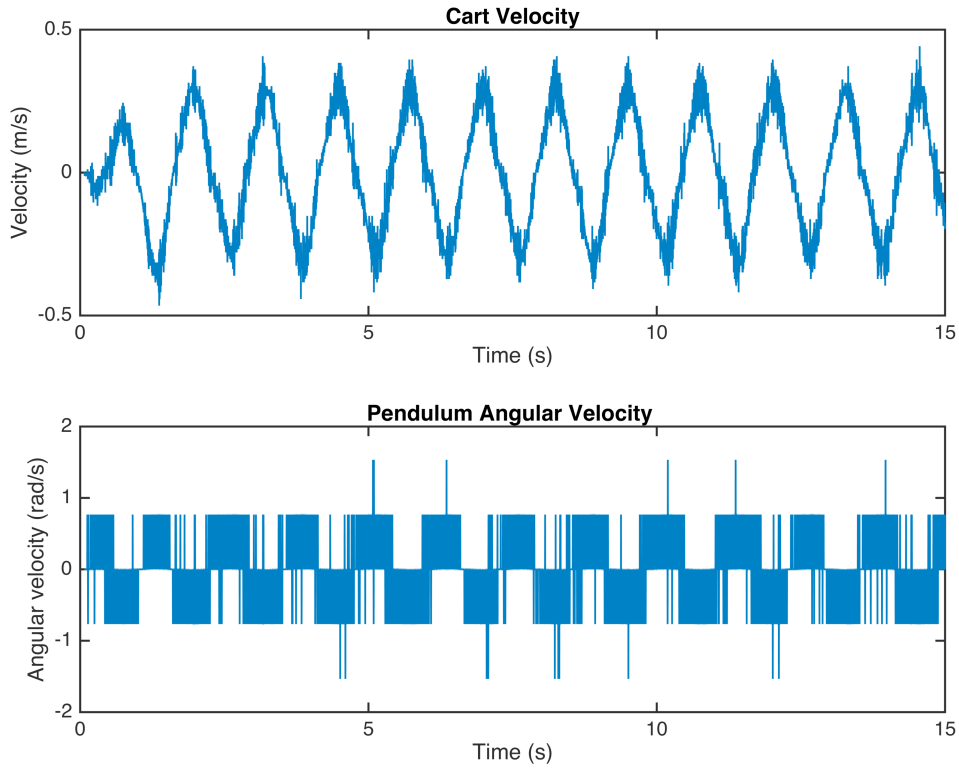


Figure 13: Cart velocity and pendulum angular velocity

Figure 13 shows the cart velocity and pendulum angular velocity found by numerically differentiating the signals x and θ, respectively. We see that the plots are very noisy due to the resolution of the cart position and pendulum angle encoders. These were found with discrete counts, so the resulting plots will not be smooth as shown.

Appendix: MATLAB Code

```
%% 1) Stability
M = 0.94;
m = 0.23;
Lp = 0.3302;
Kt = 7.67e-3;
Km = Kt;
Kg = 3.71;
Rm = 2.6;
r = 6.36e-3;
Jm = 3.9e-7;
g = 9.81;

c1 = Kg^2*Kt*Km/(r^2 * Rm);
c2 = 1/(M + 0.25*m + Kg^2*Jm/r^2);
c3 = Kg*Kt/(r*Rm);

A1 = [0, 1, 0, 0; 0, -c2*c1, -0.75*m*g*c2, 0; 0 0 0 1; 0,
0.75*(1/Lp)*c2*c1, (1 + 0.75*m*c2)*(0.75*g/Lp) 0];
B1 = [0; c2*c3; 0; -0.75*c2*c3/Lp];
C = [1 0 0 0; 0 0 1 0];
D = 0;

A12 = A1(1,2);
A22 = A1(2,2);
A23 = A1(2,3);
A34 = A1(3,4);
A42 = A1(4,2);
A43 = A1(4,3);
B2 = B1(2);
B4 = B1(4);


eig(A1)
```

```
%% 2) Simulation
sys = ss(A1,B1,C,D);
[Y,T] = step(sys);

subplot(2,1,1)
plot(T, Y(:,1));
title('Cart Position');
xlabel('Time (s)'); ylabel('Position (m)');

subplot(2,1,2);
plot(T, Y(:,2));
title('Pendulum angle');
xlabel('Time (s)'); ylabel('\theta (rad)');
```

```
%% 3a) Ak
syms a12 a22 a23 a34 a42 a43 b2 b4 k1 k2 k3 k4

A = [0 a12 0 0; 0 a22 a23 0; 0 0 0 1; 0 a42 a43 0];
B = [0; b2; 0; b4];
K = [k1, k2, k3, k4];
Ak = A-B*K
```

```
Ak1 = A1 - B1*K; % my plugged in values
```

```
%% 3b) P(K;s)
syms s
P = det(s*eye(4) - Ak);
```

```
syms s
s1 = -2 + 10j;
s2 = -2 - 10j;

s3 = -1.6 + 1.3j;
s4 = -1.6 - 1.3j;


P_des = expand((s-s1)*(s-s2)*(s-s3)*(s-s4))
```

```
%%
p = [s1, s2, s3, s4];
K = acker(A1, B1, p)
```

```
%%
A_hat = A1-B1*K;
B_hat = B1*K;
C_hat = [1 0 0 0];
D_hat = [0 0 0 0];
[num, den] = ss2tf(A_hat, B_hat, C_hat, D_hat,1);

bode(tf(num(1,:), den))
```

```
%% LAB SECTION
cm = 1/42958; % meters/count
ctheta = 1/651.8; % radians/count

%%
figure
subplot(2,1,1);
plot(x.time, x.signals.values, 'b'); hold on;
title('Cart Position');
xlabel('Time (s)'); ylabel('Position (m)');

subplot(2,1,2);
plot(theta.time, theta.signals.values);
title('Pendulum Angle');
xlabel('Time (s)'); ylabel('Angular position (rad)');
```

```
%%
S1 = load('M01_w1', 'x', 'theta', 'input', 'ddx');
S2 = load('M01_w2', 'x', 'theta', 'input', 'ddx');
S3 = load('M01_w5', 'x', 'theta', 'input', 'ddx');

[xmax, ind] = max(S1.x.signals.values);
xmin = min(S1.x.signals.values);
amp = (xmax - xmin)/2;
gain1 = amp/0.1;
period = 2*pi/1;
t = S1.x.time(ind);
phase1 = (t - period/4)*(360/period);
```

```
[xmax, ind] = max(S2.x.signals.values);
xmin = min(S2.x.signals.values);
amp = (xmax - xmin)/2;
gain2 = amp/0.1;
period = 2*pi/1;
t = S2.x.time(ind);
phase2 = (t - period/4)*(360/period);

[xmax, ind] = max(S3.x.signals.values);
xmin = min(S3.x.signals.values);
amp = (xmax - xmin)/2;
gain3 = amp/0.1;
period = 2*pi/1;
t = S3.x.time(ind);
phase3 = (t - period/4)*(360/period);

gain = [gain1 gain2 gain3]
phase = [phase1 phase2 phase3]
```

```
%%

figure
subplot(3,1,1);
plot(S1.x.time, S1.x.signals.values, 'b'); hold on;
plot(S1.input.time, S1.input.signals.values, ':');
title('Cart Position, w = 1 rad/s');
xlabel('Time (s)'); ylabel('Position (m)');
legend('Hardware response', 'Input')

subplot(3,1,2);
plot(S2.x.time, S2.x.signals.values, 'b'); hold on;
plot(S2.input.time, S2.input.signals.values, ':');
title('Cart Position, w = 2 rad/s');
xlabel('Time (s)'); ylabel('Position (m)');
legend('Hardware response', 'Input')

subplot(3,1,3);
plot(S3.x.time, S3.x.signals.values, 'b'); hold on;
plot(S3.input.time, S3.input.signals.values, ':');
title('Cart Position, w = 5 rad/s');
xlabel('Time (s)'); ylabel('Position (m)');
legend('Hardware response', 'Input')

figure
subplot(3,1,1)
plot(S1.ddx.time, S1.ddx.signals.values);
title('Cart Velocity, w = 1 rad/s');
xlabel('Time (s)'); ylabel('Velocity (m/s)');

subplot(3,1,2)
plot(S2.ddx.time, S2.ddx.signals.values);
title('Cart Velocity, w = 2 rad/s');
xlabel('Time (s)'); ylabel('Velocity (m/s)');

subplot(3,1,3)
plot(S3.ddx.time, S3.ddx.signals.values);
title('Cart Velocity, w = 5 rad/s');
xlabel('Time (s)'); ylabel('Velocity (m/s)');
```

```
figure
subplot(3,1,1)
plot(S1.theta.time, S1.theta.signals.values);
title('Pendulum Angle, w = 1 rad/s');
xlabel('Time (s)'); ylabel('Angular position (rad)');

subplot(3,1,2)
plot(S2.theta.time, S2.theta.signals.values);
title('Pendulum Angle, w = 2 rad/s');
xlabel('Time (s)'); ylabel('Angular position (rad)');

subplot(3,1,3)
plot(S3.theta.time, S3.theta.signals.values);
title('Pendulum Angle, w = 5 rad/s');
xlabel('Time (s)'); ylabel('Angular position (rad)');
```

```
%%
S = load('M01_w5.mat', 'x', 'theta', 'input', 'ddx');
S_real = load('M01_w5_4_10j.mat', 'x', 'theta', 'input', 'ddx');
S_im = load('M01_w5_2_20j.mat', 'x', 'theta', 'input', 'ddx');

% position plot
figure

subplot(3,1,1);
plot(S.x.time, S.x.signals.values, 'b'); hold on;
plot(S_real.input.time, S_real.input.signals.values, 'r:');
title('Cart Position, S_{1,2} = -2?10j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Position (m)');
legend('Hardware response', 'Input')

subplot(3,1,2);
plot(S_real.x.time, S_real.x.signals.values, 'b'); hold on;
plot(S_real.input.time, S_real.input.signals.values, 'r:');
title('Cart Position, S_{1,2} = -4?10j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Position (m)');
legend('Hardware response', 'Input')

subplot(3,1,3);
plot(S_im.x.time, S_im.x.signals.values, 'b'); hold on;
plot(S_im.input.time, S_im.input.signals.values, 'r:');
title('Cart Position, S_{1,2} = -2?20j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Position (m)');
legend('Hardware response', 'Input')

% velocity plot
figure

subplot(3,1,1);
plot(S.ddx.time, S.ddx.signals.values, 'b'); hold on;
title('Cart Velocity, S_{1,2} = -2?10j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Velocity (m/s)');

subplot(3,1,2);
plot(S_real.ddx.time, S_real.ddx.signals.values, 'b'); hold on;
title('Cart Velocity, S_{1,2} = -4?10j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Velocity (m/s)');

subplot(3,1,3);
```

```
plot(S_im.ddx.time, S_im.ddx.signals.values, 'b'); hold on;
title('Cart Velocity, S_{1,2} = -2?20j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Velocity (m/s)');

% pendulum angle plot
figure
subplot(3,1,1);
plot(S.theta.time, S.theta.signals.values, 'b'); hold on;
title('Pendulum Angle, S_{1,2} = -2?10j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Pendulum Angle (rad)');

subplot(3,1,2);
plot(S_real.theta.time, S_real.theta.signals.values, 'b'); hold on;
title('Pendulum Angle, S_{1,2} = -4?10j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Pendulum Angle (rad)');

subplot(3,1,3);
plot(S_im.theta.time, S_im.theta.signals.values, 'b'); hold on;
title('Pendulum Angle, S_{1,2} = -2?20j, \omega = 5 rad/s');
xlabel('Time (s)'); ylabel('Pendulum Angle (rad)');
```

```
%%
S = load('M01_w5.mat', 'ddx', 'dtheta');
figure
subplot(2,1,1)
plot(S.ddx.time, S.ddx.signals.values);
title('Cart Velocity');
xlabel('Time (s)'); ylabel('Velocity (m/s)');

subplot(2,1,2)
plot(S.dtheta.time, S.dtheta.signals.values);
title('Pendulum Angular Velocity ');
xlabel('Time (s)'); ylabel('Angular velocity (rad/s)');
```

Lab 6b: Luenberger Observer Design for Inverted Pendulum

Adolfo Tec
Oladipo Toriola

24 November 2016

Lab Section 4: Thursday 8-11

MEC134/EEC128: Feedback Control Systems
Fall 2016

Prof. Seth Sanders, GSI Chen-Yu Chan

1. Purpose

The purpose of this lab is to design a full-state observer to estimate the state of an inverted pendulum system given just the position of the cart and the pendulum. We will be designing the observer gain matrix L and using the state estimator for feedback control of the inverted pendulum system.

2. Pre-Lab

2.1 Controllability and Observability

We will begin designing the observer matrix by first analyzing the system's controllability and observability. The system, in its state space form, is given by the following equations:

$$\dot{\boldsymbol{x}} = A\boldsymbol{x} + B\boldsymbol{u}$$
$$\boldsymbol{y} = C\boldsymbol{x}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\dfrac{K_g^2 K_t K_m}{R_m\left[r^2\left(M+\frac{m}{4}\right)+K_g^2 J_m\right]} & -\dfrac{3}{4}\cdot\dfrac{r^2 mg}{\left[r^2\left(M+\frac{m}{4}\right)+K_g^2 J_m\right]} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \dfrac{3}{4L_p}\cdot\dfrac{K_g^2 K_t K_m}{R_m\left[r^2\left(M+\frac{m}{4}\right)+K_g^2 J_m\right]} & \dfrac{3g}{4L_p}\cdot\dfrac{r^2(M+m)+J_m K_g^2}{\left[r^2\left(M+\frac{m}{4}\right)+K_g^2 J_m\right]} & 0 \end{bmatrix}$$

$$B = \frac{rK_g K_t}{R_m\left[r^2\left(M+\frac{m}{4}\right)+K_g^2 J_m\right]}\cdot\begin{bmatrix} 0 \\ 1 \\ 0 \\ -\dfrac{3}{4L_p} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The parameters of each variable are given in Table 1.

Parameter	Value	Description
	439.6 counts/cm	Resolution of the cart position encoder
	651.9 counts/rad	Resolution of the angle encoder
M	0.94 kg	Mass of cart and motor
m	0.230 kg	Mass of pendulum
L_p	0.3302 m	Pendulum distance from pivot to center of mass
I_c	$m\,L_p^2/3$	Moment of inertia of pendulum about its center
I_e	$4m\,L_p^2/3$	Moment of inertia of pendulum about its end
K_t	$7.67 \cdot 10^{-3}$ Nm/A	Motor torque constant
K_m	$7.67 \cdot 10^{-3}$ Vs/rad	Motor back EMF constant
K_g	3.71	Motor gearbox ratio
R_m	2.6 Ω	Motor winding resistance
r	$6.36 \cdot 10^{-3}$ m	Radius of motor gear
J_m	$3.9 \cdot 10^{-7}$ kg m^2	Motor moment of inertia

Table 1: Parameters of the inverted pendulum setup

Using MATLAB's `ctrb, obsv`, and `rank` functions, we see that the system is both controllable and observable because of the fact that the controllability matrix and observability matrices have a full rank.

2.2 Observer Design

With our current hardware, we are able to measure the system's linear position x and its angle θ. However, to get the system's full state, we are also interested in knowing its linear and angular velocity, $\dot{x}$ and $\dot{\theta}$. Previously, we estimated these parameters simply by using the derivative block in Simulink. However, we observed that this method yields a poor-quality estimate of $\dot{x}$ and $\dot{\theta}$. In this lab, we will implement a Luenberger observer, which will provide a state estimate $\hat{x}$ to use for state-feedback, i.e. $u(t) = K\big(r(t)\text{-}\hat{x}(t)\big)$. This is to reduce the poor quality output of the controller. We will use the desired closed-loop poles:

$$s_{1,2} = -1.9 \pm 10j$$
$$s_{3,4} = -1.6 \pm 1.3j$$

to get the matrix K as: $K = [\text{-}12.9795 \quad \text{-}14.7230 \quad \text{-}47.8456 \quad \text{-}6.5363]$.

Given that the size of $A\text{-}LC$ must be the same size as A, we see that the dimension of L is $4x2$. This is because of the fact that $A \in \mathbb{R}^{4x4}$ and $C \in \mathbb{R}^{2x4}$, L must be $L \in \mathbb{R}^{4x2}$ to get $A\text{-}LC$ as the same size of A.

We will choose L such that the matrix $A\text{-}LC$ has at $\text{-}10 \pm 15j$ and $\text{-}12 \pm 17j$. Using MATLAB's `place` command, we find that the matrix L is:

$$L = \begin{bmatrix} 16.1595 & \text{-}2.3767 \\ 254.978 & \text{-}5.4834 \\ 15.7136 & 21.0282 \\ 180.7734 & 378.2196 \end{bmatrix}$$

2.3 Simulation

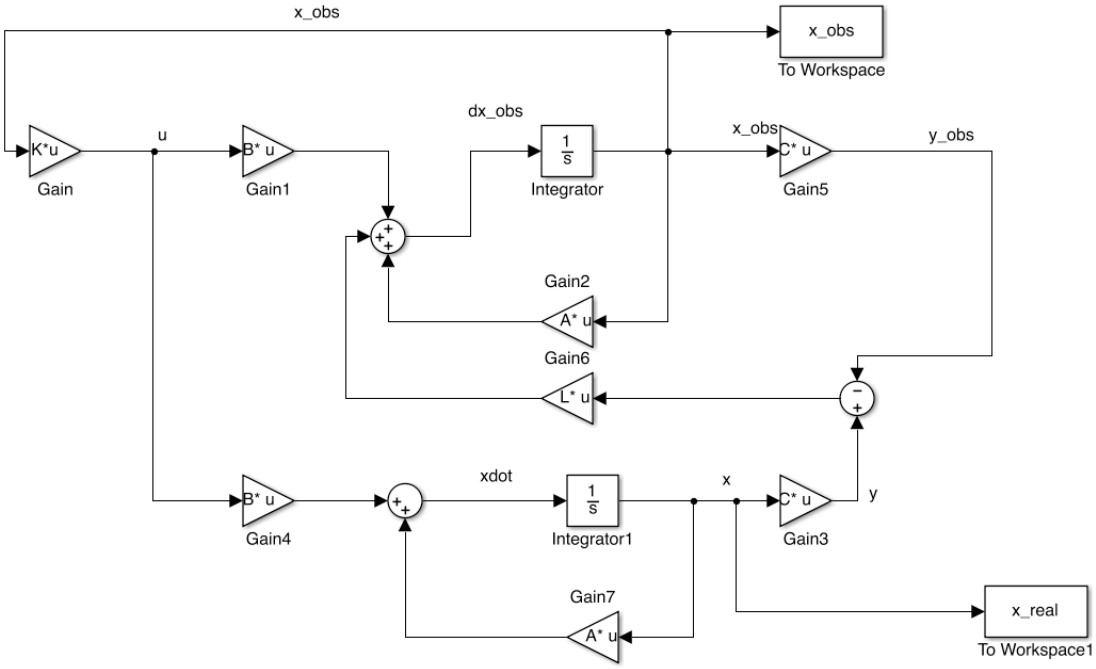


Figure 1: Simulink block diagram of inverted pendulum system with Luenberger Observer

To see if the correct observer matrix was designed, we will simulate the system with a 10cm position perturbation and a 5 degree angle perturbation of the plant. This is achieved by setting the initial conditions of x_0 (Integrator1) of the plant as $x_0 = [0.1, 0, 0, 5*\frac{\pi}{180}, 0]$.

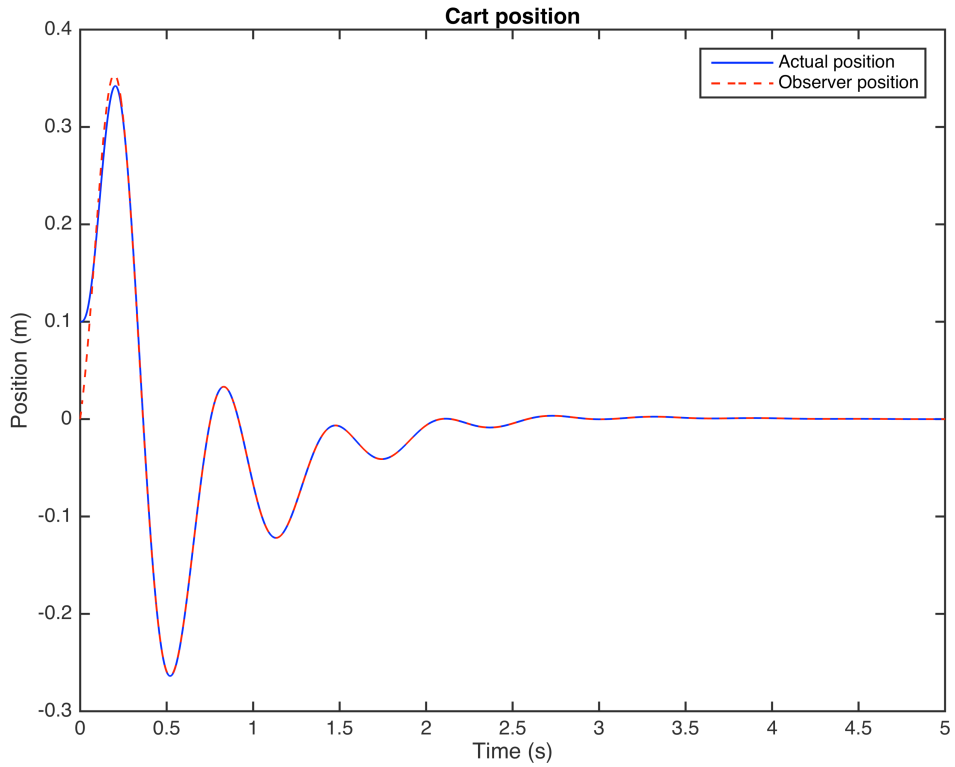


Figure 2: Actual cart position and estimated observer position

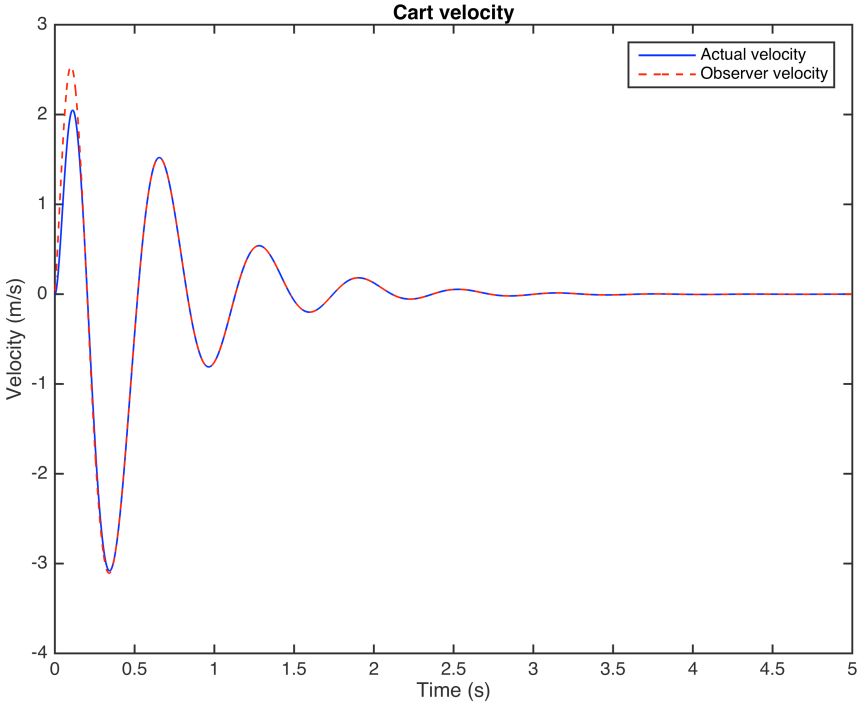


Figure 3: Actual cart velocity and estimated observer velocity

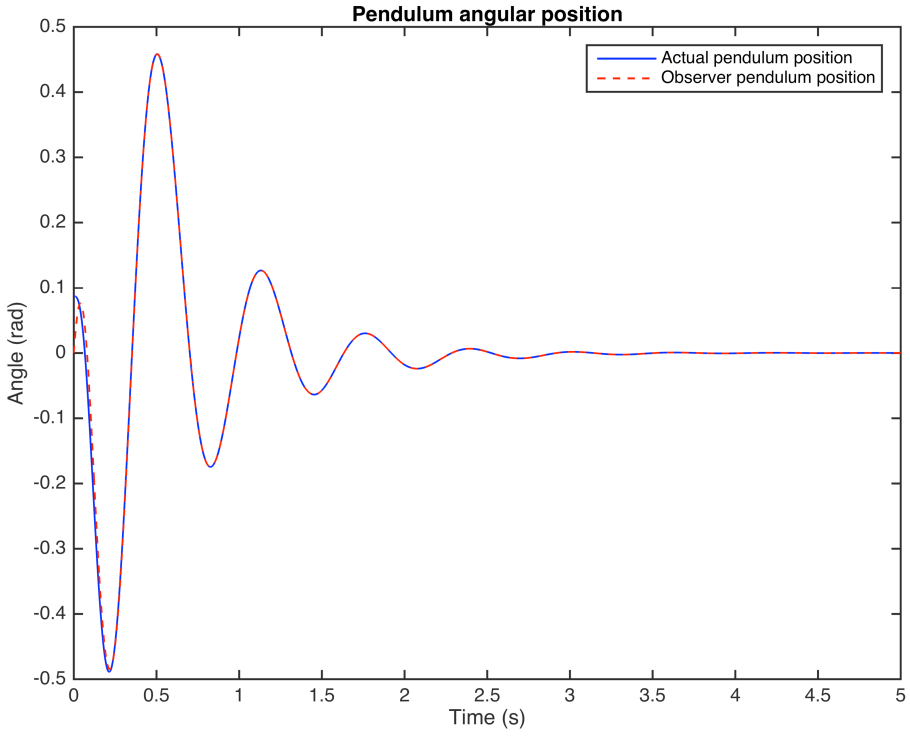


Figure 4: Actual pendulum angle and estimated observer angle

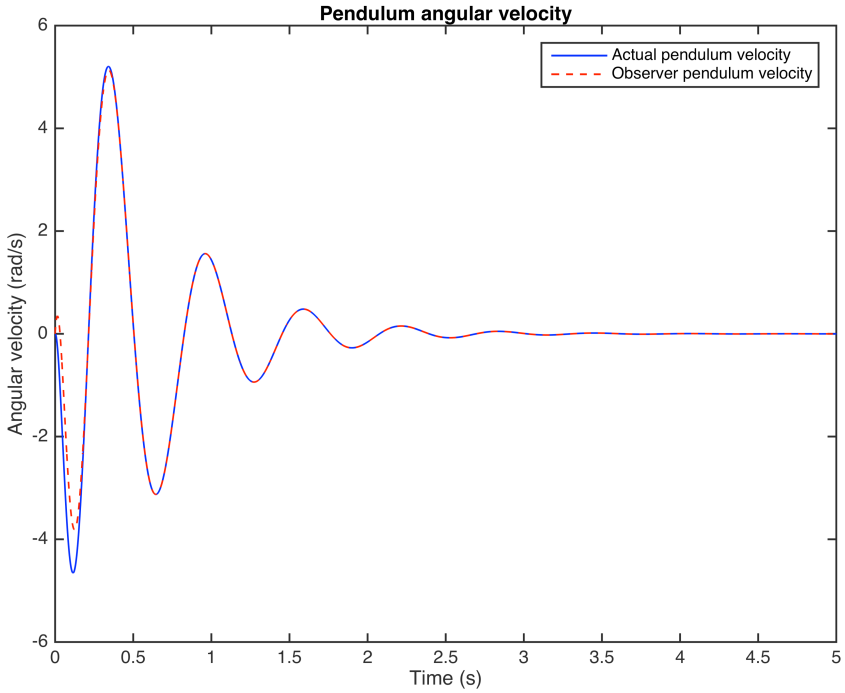


Figure 5: Actual pendulum angular velocity and observer angular velocity

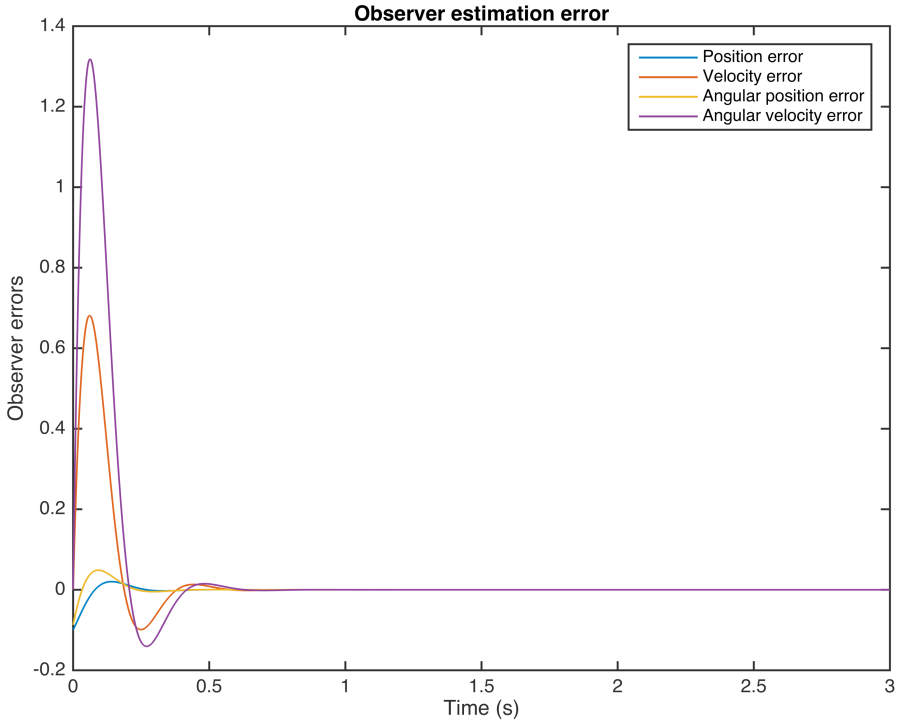


Figure 6: Estimation errors for each state variable

From Figure 6, we see that the initial errors of the cart position and pendulum angular position are nonzero due to the fact that the perturbation is simulated at t=0. We also see that both velocity errors are zero initially and then spike immediately before converging to a zero error term. We verify that the correct observer matrix is formed due to the fact that the errors converge to 0.

3. Lab

In this lab, we will implement the designed Luenberger observer on the hardware to estimate the states of the cart velocity and the pendulum angular velocity. This is achieved by feeding the output of the plant into the observer and using the full state output from the observer as the feedback into the controller.

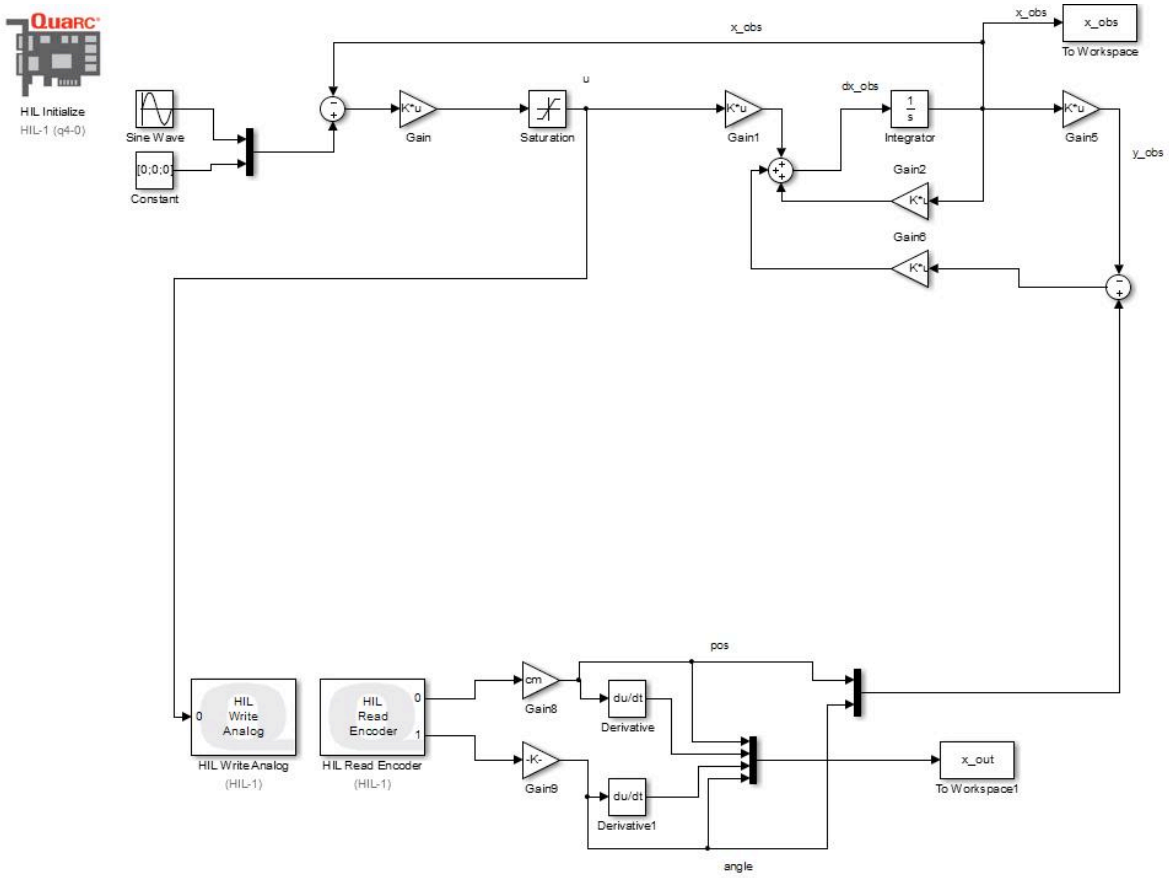


Figure 7: Hardware Simulink block diagram implementing the Luenberger observer

To see how the actual system hardware reacts with the Luenberger observer implemented, we set the reference signal to zero and recorded the output $\hat{y}$ of the observer and the actual measurement y when manually applying small perturbations. Figures 8 and 9 show the cart position and pendulum angular position plots along with their associated errors where the error is defined as the difference between the actual encoder position signal and the observer signal. Note that the first 10 seconds of the hardware response were not properly recorded due to Simulink settings.

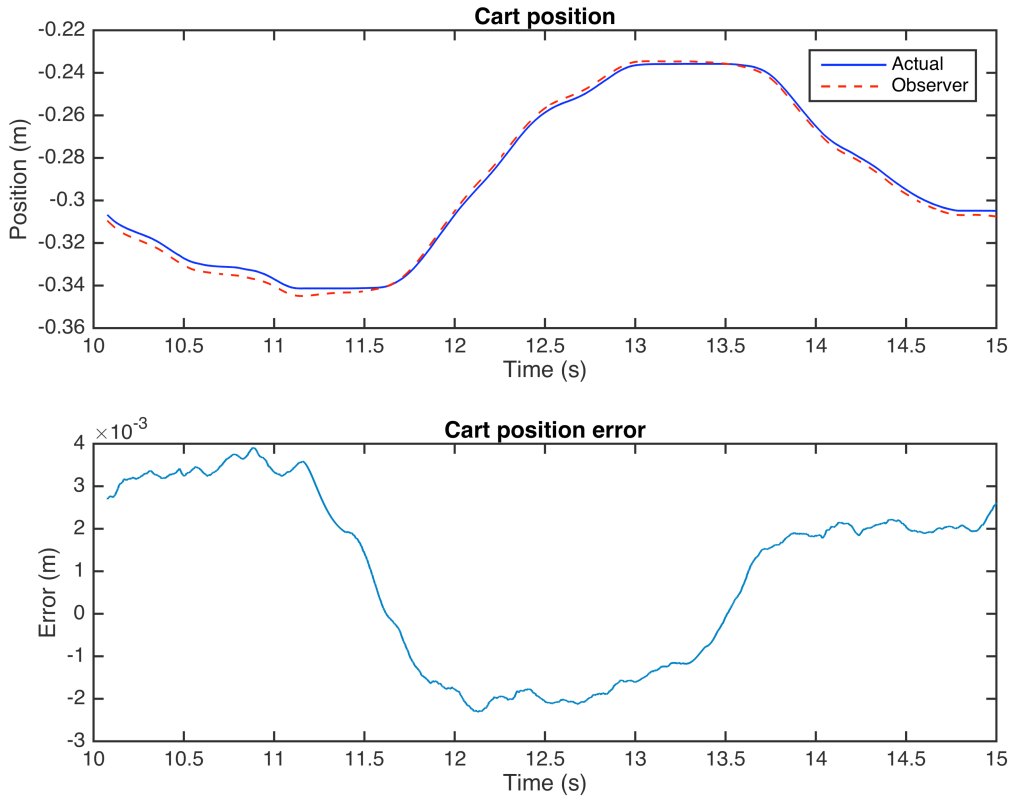


Figure 8: Hardware cart position and observer cart position due to small perturbations

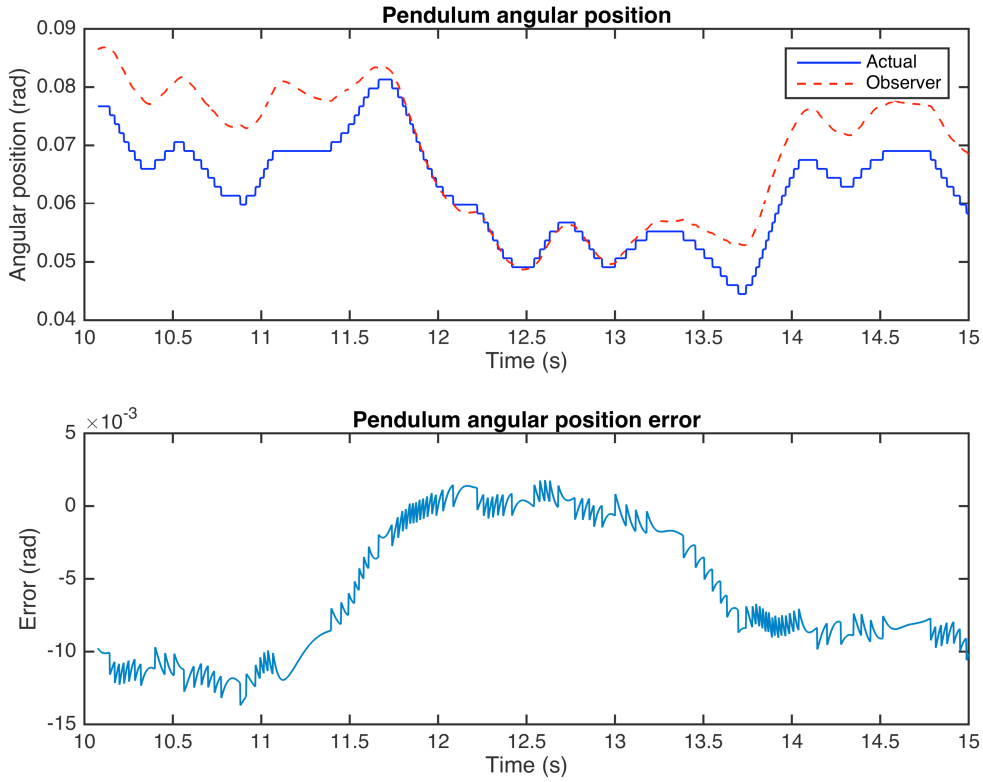


Figure 9: Hardware pendulum angle and observer pendulum angle due to small perturbations

In order to report how well the controller with the Luenberger observer behaves, we will now compare the observer controller to the simple controller using the derivative blocks as discussed in the prelab section with a variety of input signals.

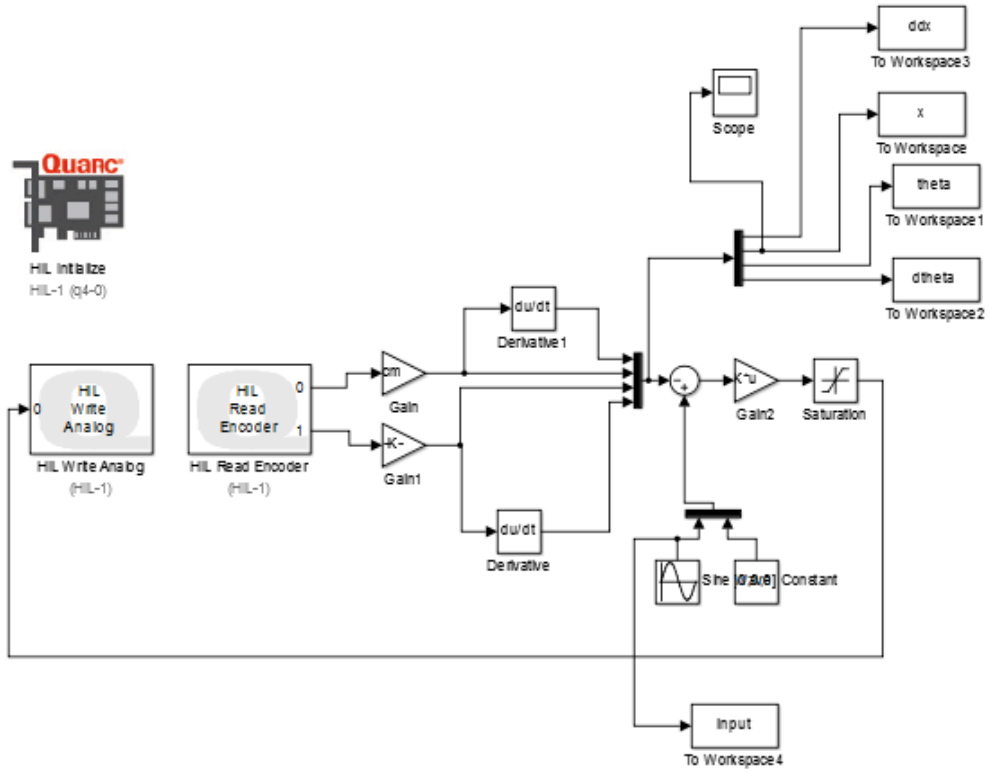


Figure 10: Previously used controller using derivate blocks to estimate system velocities

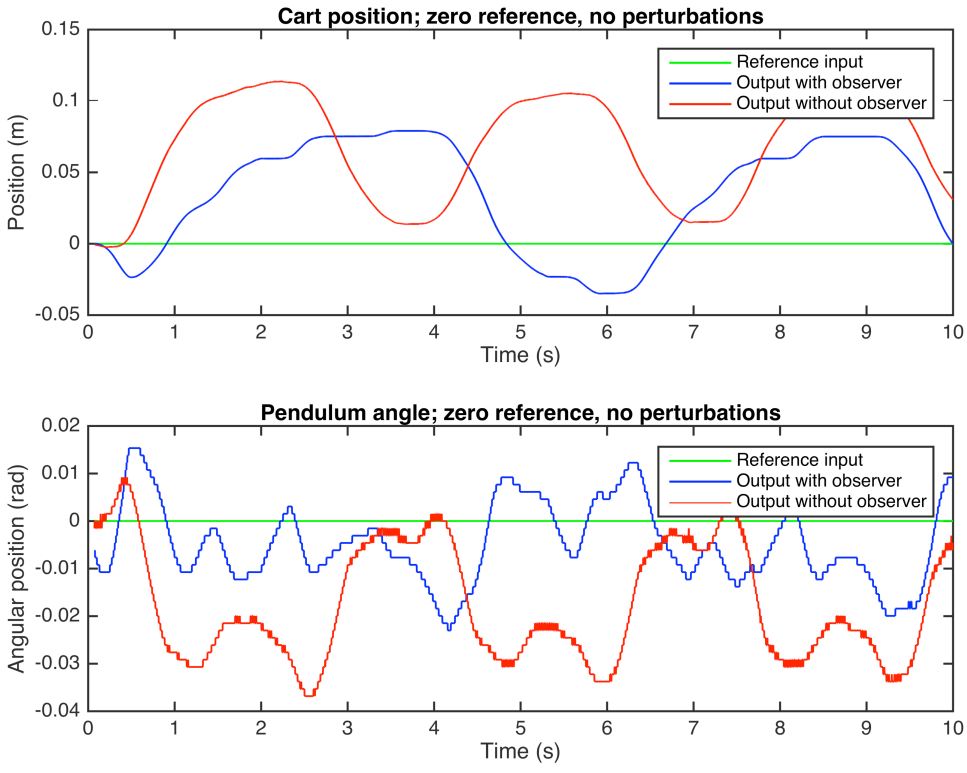


Figure 11: Cart position and pendulum angle with zero reference and no perturbations

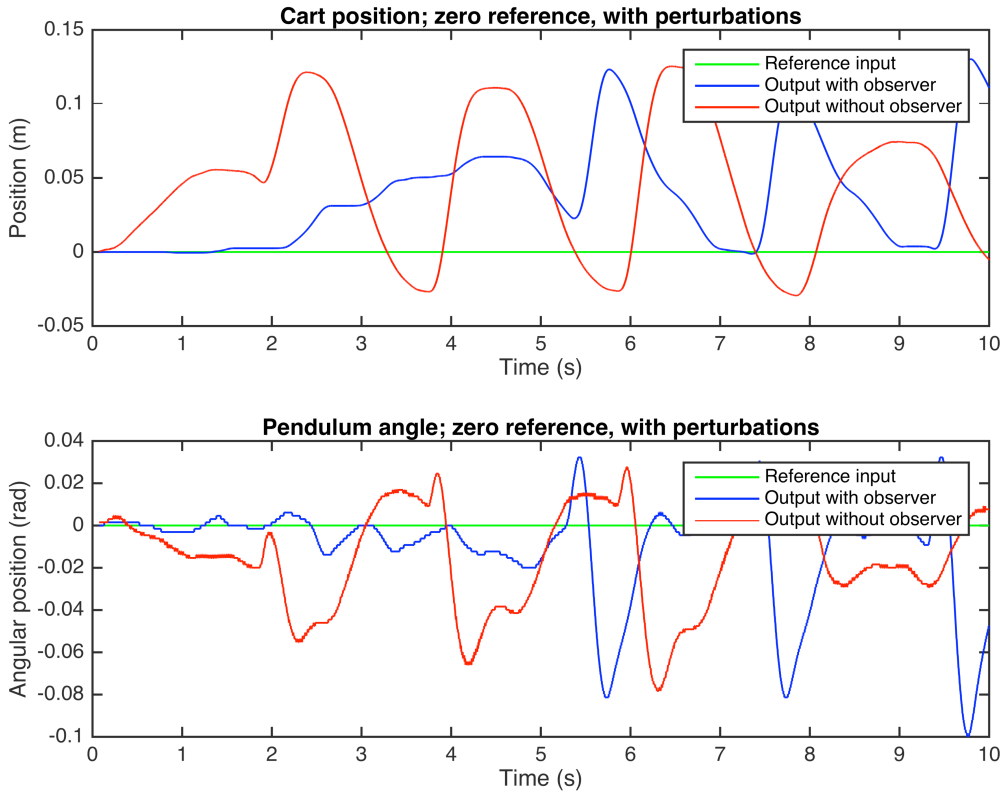


Figure 12: Cart position and pendulum angle with zero reference and perturbations

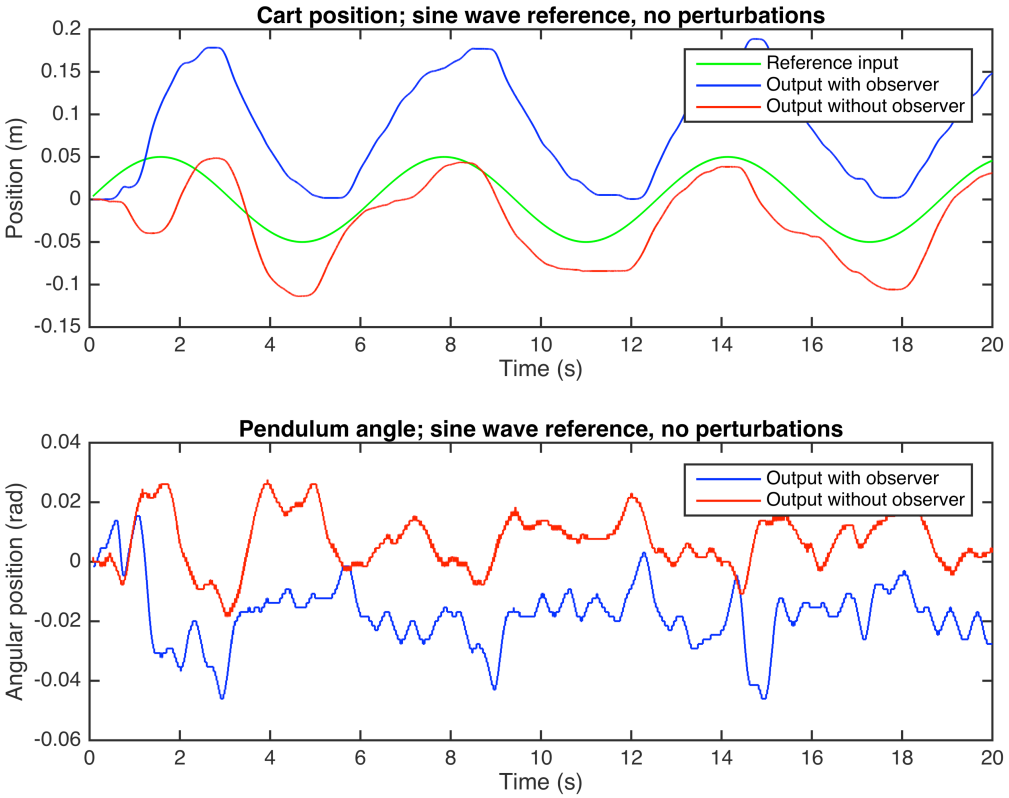


Figure 13: Cart position and pendulum angle with a sinusoidal reference input

The three tested cases to compare how the system responded with and without an observer were a zero reference input without perturbations, a zero reference input with perturbations, and a sinusoidal reference position input with amplitude 5 cm and frequency 1 rad/s without perturbations (i.e. $r_1(t) = 0.05\sin(t)$) shown in Figures 11, 12, and 13, respectively. In all of the cases tested, one major aspect noticed when comparing the two different controllers was the fact that the controller using an observer resulted in the system responding far quieter than the response of the system without an observer. This was one of the goals and purposes of using an observer in the first place, so that performance was met. Another response behavior noticed was that both controllers yielded in an offset around the reference input, showing that an observer does not help in getting rid of oscillations about the equilibrium point.

To further investigate the differences in performance from the two controllers, we will compare the estimates of the cart velocity and the pendulum angular velocity from the input $r_1(t) = 0.05\sin(t)$.

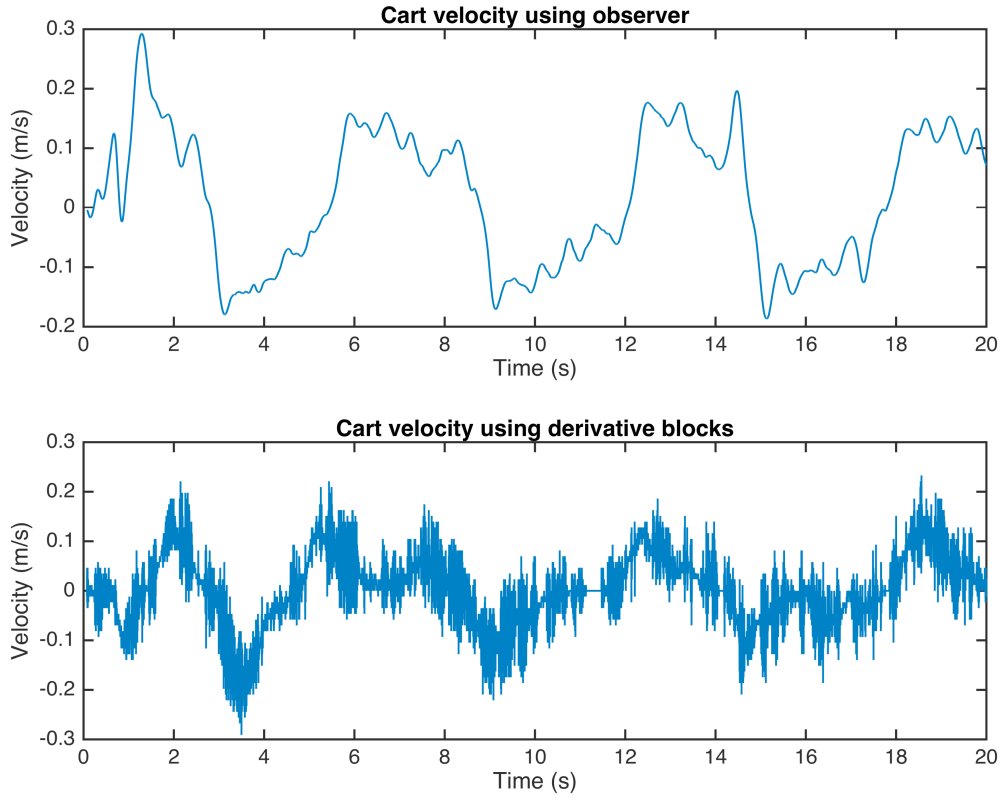


Figure 14: Estimates of the cart velocity using an observer and not using an observer

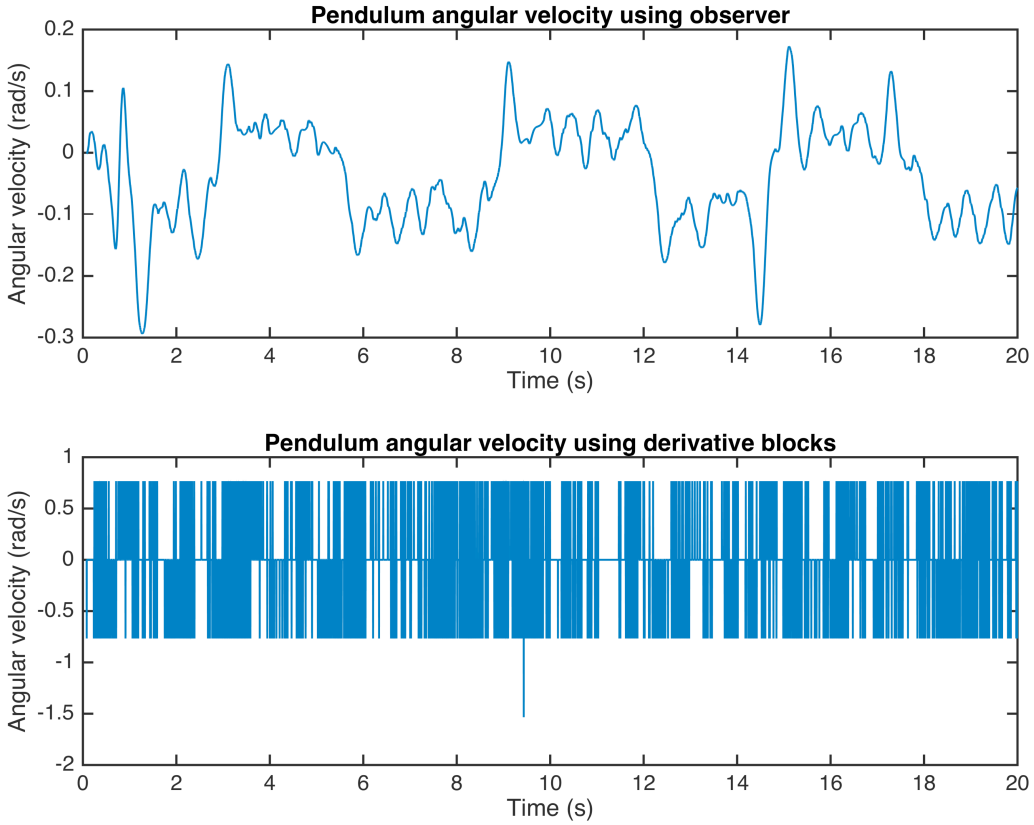


Figure 15: Estimates of the pendulum angular velocity using an observer and not using an observer

From Figures 14 and 15, we see that there is a clear difference between the estimates of the cart velocity and the pendulum velocity with the most noticeable difference being that the controller using the derivative blocks is far noisier than the controller using the Luenberger observer. This noise difference is due to the way the derivatives of the system sensors are estimated: whereas the controller using the observer corrects future estimates of the state based on the present error, the controller using the derivative blocks computes the numerical derivative of the sensor values in discrete times.

This reduction in noise leads us to conclude that the controller using the Luenberger observer gives the better performance. Without all the noise present in the estimates of the full state, the feedback input fed into the encoder was thus less noisy and led to better results in the sense that the grinding motor noises were severely reduced. Although both controllers were able to regulate the cart and pendulum system, the controller using the observer only deals with one clear signal rather than a noisy signal, something that can be useful for future analysis of the system.

Appendix: MATLAB Code

```
%% Initialize system parameters
M = 0.94;
m = 0.23;
Lp = 0.3302;
Kt = 7.67e-3;
Km = Kt;
Kg = 3.71;
Rm = 2.6;
r = 6.36e-3;
Jm = 3.9e-7;
g = 9.81;

cm = 1/42958; % meters/count
ctheta = 1/651.8; % radians/count

c1 = Kg^2*Kt*Km/(r^2*Rm);
c2 = 1/(M + 0.25*m + Kg^2*Jm/r^2);
c3 = Kg*Kt/(r*Rm);

A = [0, 1, 0, 0; 0, -c2*c1, -0.75*m*g*c2, 0; 0 0 0 1; 0,
0.75*(1/Lp)*c2*c1, (1 + 0.75*m*c2)*(0.75*g/Lp) 0];
B = [0; c2*c3; 0; -0.75*c2*c3/Lp];
C = [1 0 0 0; 0 0 1 0];
D = 0;

s1 = -1.9 + 10j;
s2 = -1.9 - 10j;
s3 = -1.6 + 1.3j;
s4 = -1.6 - 1.3j;
p = [s1, s2, s3, s4];
K = acker(A, B, p);

rank_co = rank(ctrb(A,B));
rank_ob = rank(obsv(A,C));
```

```
%% Finding L matrix
Lpoles = [-10+15j -10-15j -12+17j -12-17j];
L_trans = place(A', C', Lpoles);
L = L_trans';
```

```
%% Define variables for simulation
x0_obs = [0;0;0;0];
x0 = [0.1; 0; 5*pi/180; 0];
sim('prelab6b')

t = x_real.time;
x = x_real.signals.values(:,1);
dx = x_real.signals.values(:,2);
theta = x_real.signals.values(:,3);
dtheta = x_real.signals.values(:,4);

x_obs_x = x_obs.signals.values(:,1);
dx_obs = x_obs.signals.values(:,2);
theta_obs = x_obs.signals.values(:,3);
dtheta_obs = x_obs.signals.values(:,4);

e = x_obs.signals.values - x_real.signals.values;
```

```
%% Simulation plots
% Position
figure
plot(t,x, 'b'); hold on
plot(t, x_obs_x, 'r--');
title('Cart position');
ylabel('Position (m)'); xlabel('Time (s)'); xlim([0, 5]);
legend('Actual position', 'Observer position');

% Velocity
figure
plot(t,dx, 'b'); hold on;
plot(t, dx_obs, 'r--');
title('Cart velocity');
ylabel('Velocity (m/s)'); xlabel('Time (s)'); xlim([0, 5]);
legend('Actual velocity', 'Observer velocity');

% Pendulum angle
figure
plot(t,theta, 'b'); hold on;
plot(t, theta_obs, 'r--');
title('Pendulum angular position');
ylabel('Angle (rad)'); xlabel('Time (s)'); xlim([0, 5]);
legend('Actual pendulum position', 'Observer pendulum position');

% Pendulum angular velocity
figure
plot(t,dtheta, 'b'); hold on;
plot(t, dtheta_obs, 'r--');
title('Pendulum angular velocity');
ylabel('Angular velocity (rad/s)'); xlabel('Time (s)'); xlim([0, 5]);
legend('Actual pendulum velocity', 'Observer pendulum velocity');

% Error plot
figure
plot(t,e(:,1)); hold on
plot(t,e(:,2));
plot(t,e(:,3));
plot(t,e(:,4));
title('Observer estimation error');
ylabel('Observer errors'); xlabel('Time (s)'); xlim([0, 3]);
legend('Position error', 'Velocity error', 'Angular position error',
'Angular velocity error');
```

```
%% Hardware plots
% 4.1 Perturbation plots
S = load('perturbation.mat', 'x_out', 'x_obs');
t = S.x_out.time;
x_act = S.x_out.signals.values(:,1); x_obs = S.x_obs.signals.values(:,1);
theta_act = S.x_out.signals.values(:,4); theta_obs =
S.x_obs.signals.values(:,3);
e = x_act - x_obs;
e_theta = theta_act - theta_obs;

% Cart position plot
figure
subplot(2,1,1)
plot(t, x_act, 'b'); hold on;
plot(t, x_obs, 'r--');
legend('Actual', 'Observer');
title('Cart position'); xlabel('Time (s)'); ylabel('Position (m)');

subplot(2,1,2)
plot(t, e)
title('Cart position error'); xlabel('Time (s)'); ylabel('Error (m)');

% Pendulum angular position plot
figure
subplot(2,1,1)
plot(t, theta_act, 'b'); hold on;
plot(t, theta_obs, 'r--');
legend('Actual', 'Observer');
title('Pendulum angular position'); xlabel('Time (s)'); ylabel('Angular
position (rad)');

subplot(2,1,2)
plot(t, e_theta)
title('Pendulum angular position error'); xlabel('Time (s)');
ylabel('Error (rad)');
clear;
```

```
%% 4.2 Comparing controllers
% Zero reference, no perturbations
S1 = load('zero_ref_obs.mat', 'x_out');
S2 = load('zero_ref_der.mat', 'x', 'theta', 'input');
t_obs = S1.x_out.time; t_der = S2.x.time;
x_obs = S1.x_out.signals.values(:,1);
x_der = S2.x.signals.values;
theta_obs = S1.x_out.signals.values(:,4);
theta_der = S2.theta.signals.values;
input = S2.input.signals.values;

% Cart position plot
figure
subplot(2,1,1)
plot(t_der, input, 'g'); hold on;
plot(t_obs, x_obs, 'b');
plot(t_der, x_der, 'r');
legend('Reference input', 'Output with observer', 'Output without
observer');
title('Cart position; zero reference, no perturbations'); xlabel('Time
(s)'); ylabel('Position (m)');

% Pendulum angular plot
subplot(2,1,2)
plot(t_der, input, 'g'); hold on;
plot(t_obs, theta_obs, 'b');
plot(t_der, theta_der, 'r');
legend('Reference input', 'Output with observer', 'Output without
observer');
title('Pendulum angle; zero reference, no perturbations'); xlabel('Time
(s)'); ylabel('Angular position (rad)');
clear;

%% Zero reference, with perturbations
S1 = load('pert_ref_obs.mat', 'x_out');
S2 = load('pert_ref_der.mat', 'x', 'theta', 'input');
t_obs = S1.x_out.time; t_der = S2.x.time;
x_obs = S1.x_out.signals.values(:,1);
x_der = S2.x.signals.values;
theta_obs = S1.x_out.signals.values(:,4);
theta_der = S2.theta.signals.values;
input = S2.input.signals.values;

% Cart position plot
figure
subplot(2,1,1)
plot(t_der, input, 'g'); hold on;
plot(t_obs, x_obs, 'b');
plot(t_der, x_der, 'r');
legend('Reference input', 'Output with observer', 'Output without
observer');
title('Cart position; zero reference, with perturbations'); xlabel('Time
(s)'); ylabel('Position (m)');

% Pendulum angular plot
subplot(2,1,2)
plot(t_der, input, 'g'); hold on;
plot(t_obs, theta_obs, 'b');
plot(t_der, theta_der, 'r');
legend('Reference input', 'Output with observer', 'Output without
observer');
title('Pendulum angle; zero reference, with perturbations'); xlabel('Time
```

```
(s)'); ylabel('Angular position (rad)');
clear;

%% Sinusoidal input reference
S1 = load('sine_ref_obs.mat', 'x_out');
S2 = load('sine_ref_der.mat', 'x', 'theta', 'input');
t_obs = S1.x_out.time; t_der = S2.x.time;
x_obs = S1.x_out.signals.values(:,1);
x_der = S2.x.signals.values;
theta_obs = S1.x_out.signals.values(:,4);
theta_der = S2.theta.signals.values;
input = S2.input.signals.values;

% Cart position plot
figure
subplot(2,1,1)
plot(t_der, input, 'g'); hold on;
plot(t_obs, x_obs, 'b');
plot(t_der, x_der, 'r');
legend('Reference input', 'Output with observer', 'Output without
observer');
title('Cart position; sine wave reference, no perturbations');
xlabel('Time (s)'); ylabel('Position (m)');

% Pendulum angular plot
subplot(2,1,2)
plot(t_obs, theta_obs, 'b'); hold on;
plot(t_der, theta_der, 'r');
legend('Output with observer', 'Output without observer');
title('Pendulum angle; sine wave reference, no perturbations');
xlabel('Time (s)'); ylabel('Angular position (rad)');
clear;
```

```
%% 4.3 Comparing controller velocities
% Cart velocity
S1 = load('sine_ref_obs.mat', 'x_obs');
S2 = load('sine_ref_der.mat', 'ddx', 'dtheta');
t_obs = S1.x_obs.time; t_der = S2.ddx.time;
dx_obs = S1.x_obs.signals.values(:,2);
dx_der = S2.ddx.signals.values;
dtheta_obs = S1.x_obs.signals.values(:,4);
dtheta_der = S2.dtheta.signals.values;

% Cart position plot
figure
subplot(2,1,1)
plot(t_obs, dx_obs);
title('Cart velocity using observer'); xlabel('Time (s)');
ylabel('Velocity (m/s)');

subplot(2,1,2)
plot(t_der, dx_der);
title('Cart velocity using derivative blocks'); xlabel('Time (s)');
ylabel('Velocity (m/s)');

% Pendulum angular plot
figure
subplot(2,1,1)
plot(t_obs, dtheta_obs);
title('Pendulum angular velocity using observer'); xlabel('Time (s)');
ylabel('Angular velocity (rad/s)');

subplot(2,1,2)
plot(t_der, dtheta_der);
title('Pendulum angular velocity using derivative blocks'); xlabel('Time
(s)'); ylabel('Angular velocity (rad/s)');
clear;
```

Lab 6d: Self-Erecting Inverted Pendulum (SEIP)

Adolfo Tec
Oladipo Toriola

8 December 2016

Lab Section 4: Thursday 8-11

MEC134/EEC128: Feedback Control Systems
Fall 2016

Prof. Seth Sanders, GSI Chen-Yu Chan

1. Purpose

The purpose of this lab is to design a controller that erects a pendulum to the "up" position that starts in the "down" position. This is done by implementing a non-linear controller based on an "energy-pumping" method to erect the pendulum and a Luenberger observer controller to stabilize the inverted pendulum.

2. Pre-Lab

2.1 Energy of the Pendulum

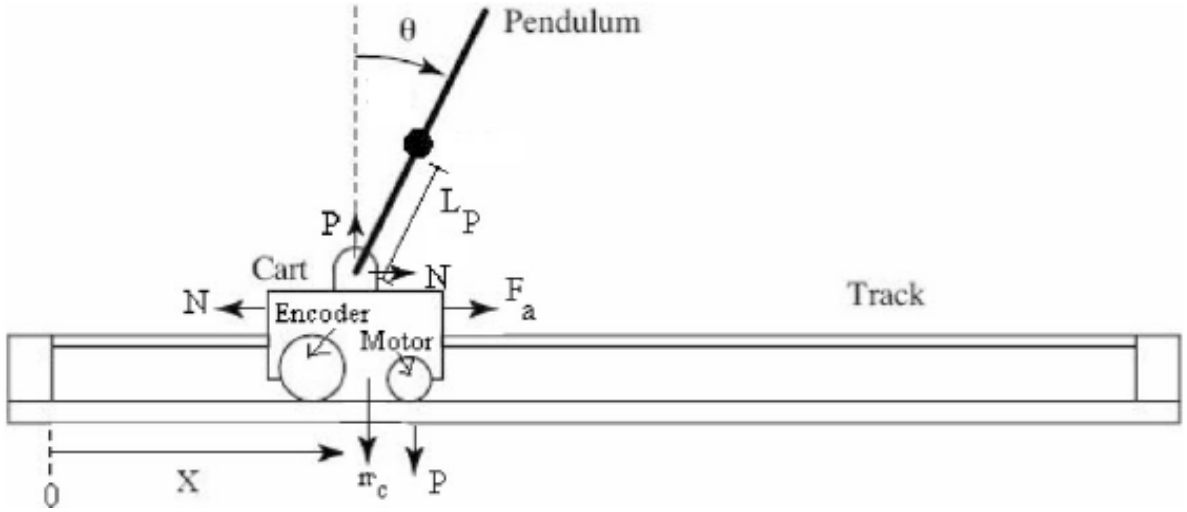


Figure 1: Free body diagram of the inverted pendulum setup (ignoring friction)

In order to implement the energy-pumping method, we must first find the total energy of the system. The total mechanical energy of the pendulum without the cart is given by:

$$E = \frac{1}{2}J\dot{\theta}^2 + mgL_p(cos\theta\text{-}1)$$

Note that at $\theta = 0$, the total energy is zero and the pendulum is in the upright position, i.e. $E(0,0) = 0$.

When the pendulum is at rest at $\theta = \pi$, the total energy is: $E(\pi, 0) = \text{-}2mgL_p$.

Thus, since the energy is not zero at $\theta = \pi$, we must implement the controller that increases the energy from its initial value to the target value of 0. In order to increase the energy, we will compute the derivative of E: Let $\ddot{x}$ be the acceleration of the pivot point. Given that the dynamics of the pendulum is given by:

$$J\ddot{\theta} = mgL_p \sin(\theta) \text{-}mL_p \cos(\theta)\, \ddot{x}$$

where $J = \frac{4}{3}mL_p^2$, we find that the derivative of the total energy of the pendulum is:

$$\frac{d}{dt}E(\theta, \dot{\theta}) = \text{-}mL_p \sin(\theta) \text{-}mL_p \cos(\theta)\, \ddot{x}$$

Therefore, to increase the energy of the system as fast as possible, we would like to apply maximal positive acceleration when $\dot{\theta}\cos(\theta) < 0$ and negative acceleration when

$\dot{\theta}\cos(\theta) > 0$. Because of this, we will implement the simple control law (sometimes called bang-bang control):

$$V = \begin{cases} 0, & when\ E(\theta, \dot{\theta}) \geq 0 \\ V_{\max}, & when\ E(\theta, \dot{\theta}) < 0\ and\ \dot{\theta}\cos(\theta) < 0 \\ -V_{\max}, & when\ E(\theta, \dot{\theta}) < 0\ and\ \dot{\theta}\cos(\theta) > 0 \end{cases}$$

where $V_{max} = 6V$.

2.2 Obtaining Derivatives

As seen in previous labs, taking numerical derivatives for estimates of $\dot{\theta}$ and $\dot{x}$ yielded poor results. Because of this, an observer was implemented to obtain an estimate $\hat{x}$ of the system state. However, the observer implemented used the linear system model and is therefore only valid for $\theta \approx 0$. In this lab, we are using the full range $\theta \in [0, 2\pi)$, so we have to take the non-linearities in the system into account. Therefore, we will use a dynamical system approximating a derivative for obtaining estimates of $\dot{\theta}$ and $\dot{x}$.

We will approximate a transfer function in the form:

$$D(s) = sH_{lp}(s) = s\frac{1}{c_{lp}s + 1}$$

with parameters c_{lp} (l_p for low-pass) due to the fact that we cannot implement a pure differentiator with transfer function $G(s) = s$ in MATLAB. This is, in essence, a differentiator combined with a lowpass filter, resulting in a phase-shift.

We wish to ensure that the phase shift is small while not attenuating frequencies of interest too much, so we will find the largest value of c_{lp} such that at the natural frequency $\omega_0 = \sqrt{\frac{g}{l}}$, the phase shift is $\left|\phi\left(H_{lp}(j\omega_0)\right)\right| \leq 5º$. We find that the largest value of $c_{lp} = \sqrt{\frac{2L_p}{g}}$ tan (5).

3. Lab

3.1 Implementing the Swing-Up Controller on the Hardware

The full system block diagram is shown below in Figure 2. Within this block diagram are subsystems which will be described.[1]

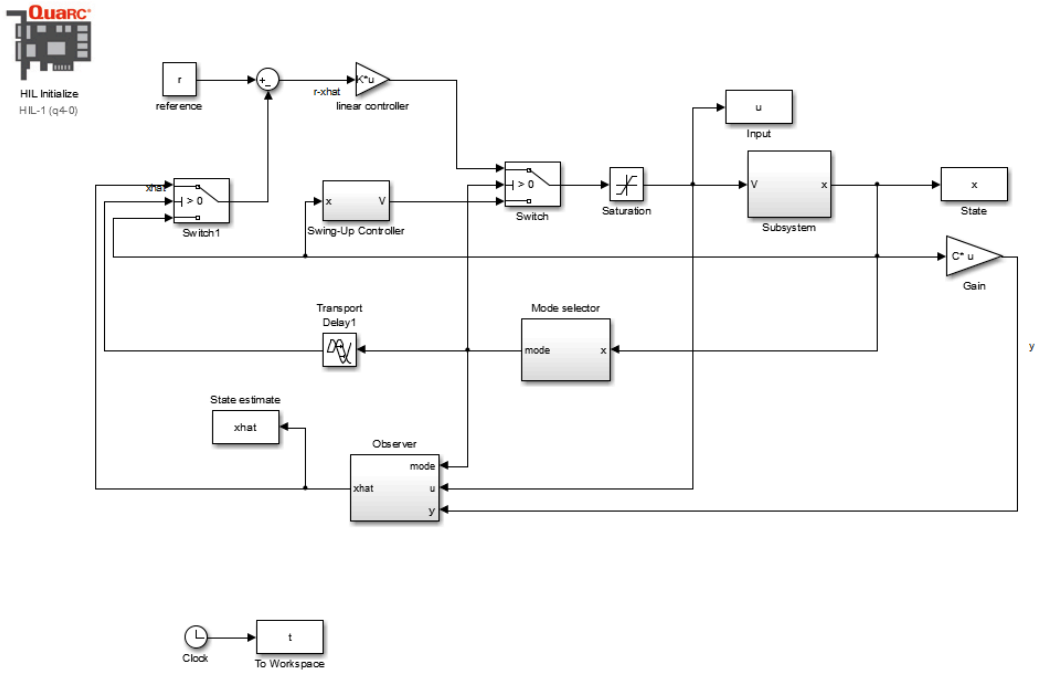


Figure 2: SEIP block diagram with subsystems

3.1.1 The Inverted Pendulum Hardware

The inverted pendulum hardware block diagram is shown below in Figure 3. Since we have to take the non-linearities in the system into account when swinging up, we cannot simply use a linear observer for providing estimates of the cart velocity $\dot{x}$ and the pendulum angle $\dot{\theta}$. This is taken into account with the blocks labeled Derivative_x and Derivative_theta. The block labeled "MATLAB Function" is used to transform the coordinates of the system such that $\theta = 0$ in the "up" position.[2] This was achieved by letting $\theta = mod(\theta', 2\pi)$-$\pi$

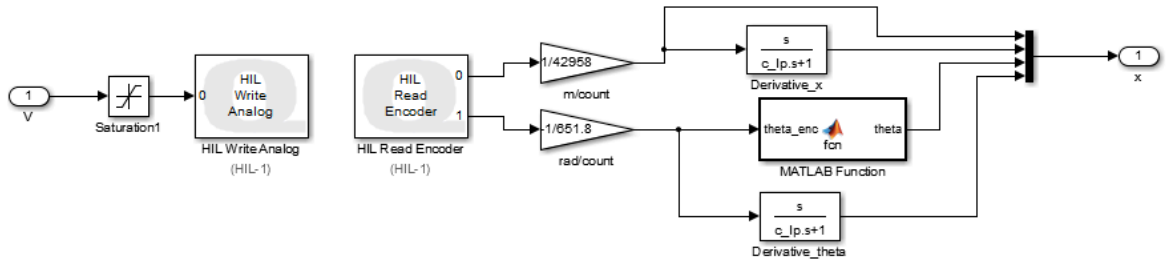


Figure 3: Inverted pendulum hardware subsystem

[1] The block labeled "Subsystem" is the Inverted Pendulum Hardware subsystem.
[2] Originally, the block was labeled "Angle Initialization" but the name of the block was not changed back.

where θ' is the angle reading from the encoder (after the conversion factor).

3.1.2 The Mode Selector

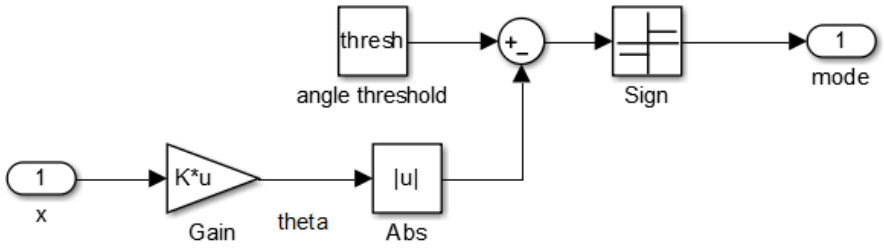


Figure 4: Mode selector subsystems

The mode selector block diagram is shown in Figure 4. In this block, we check whether the absolute value of θ is less than a set threshold value (called thresh in the block diagram) and choose the value of the mode signal accordingly. The mode signal is used to switch between Swing-Up and stabilizing controller. It follows the behavior:

$$\text{mode} = \begin{cases} 1, |\theta| \leq \text{thresh} \\ -1, \text{otherwise} \end{cases}$$

The value of the block "thresh" was found experimentally in the lab with the system with the starting point thresh = 10º. The value was then tuned to yield appropriate results and had the value thresh = 12º.

3.1.3 The Swing-Up Controller

The swing-up controller implements the "energy-pumping" method to swing the pendulum to the "up" position as described above. The block labeled "Subsystem" computes the energy of the pendulum as a function of the angle θ and angular velocity $\dot{\theta}$.[3]

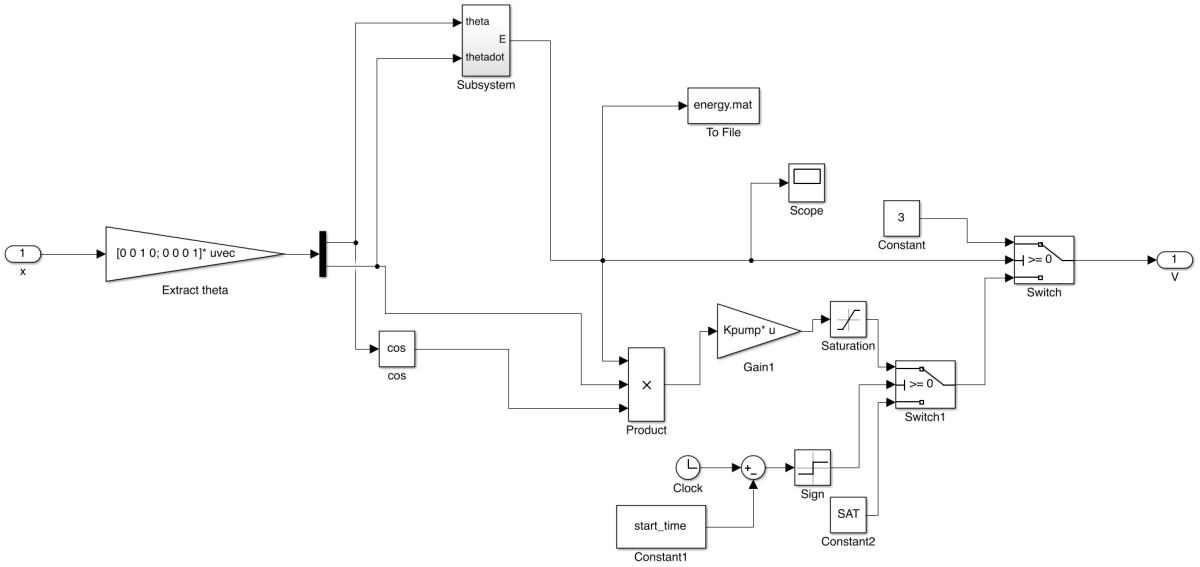


Figure 5: Swing-up Controller subsystem

[3] The block is supposed to be labeled "Energy function" but was not renamed

Switch 1 ensures that the pendulum will swing to the right initially. The gain K_{pump} was found experimentally during the testing stage and was tuned to be $K_{pump} = 100$. The switch in Figure 5 makes sure that the swing-up controller stops pumping energy into the pendulum as soon as $E \geq 3$J instead of 0. It was found that giving the controller some buffer allowed the system to behave as desired.

3.1.4 The Observer

The observer implemented in this lab is similar to the Luenberger observer implanted in Lab 6b. The only difference is that there is an extra part added to the left side of the block diagram in Figure 6 to ensure that the observer is only in use when the linearization of the system is reasonably accurate.

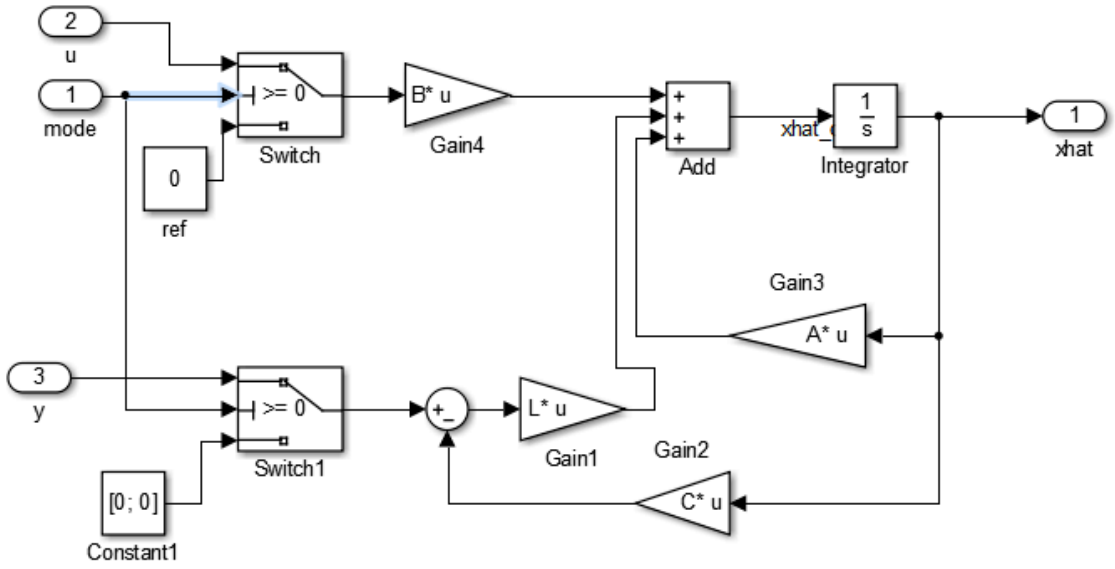


Figure 6: Observer subsytems

3.2 Testing and Debugging the Swing-Up Controller

When implementing the initially controller we had, the pendulum did not swing up all the way to the desired position of "up." Therefore, to make the system behave correctly, some parameters were tuned such as K_{pump} and thresh.

Figure 7 shows the plots of the cart position x and pendulum angle θ. The threshold is set to 12º and is also plotted on the pendulum angle plot.[4]

[4] All plots except the energy plot do not depict the whole experiment as the first 10 seconds are missing due to incorrect Simulink settings. Due to this, we cannot see when the switch occurs as it occurred before 10 seconds.

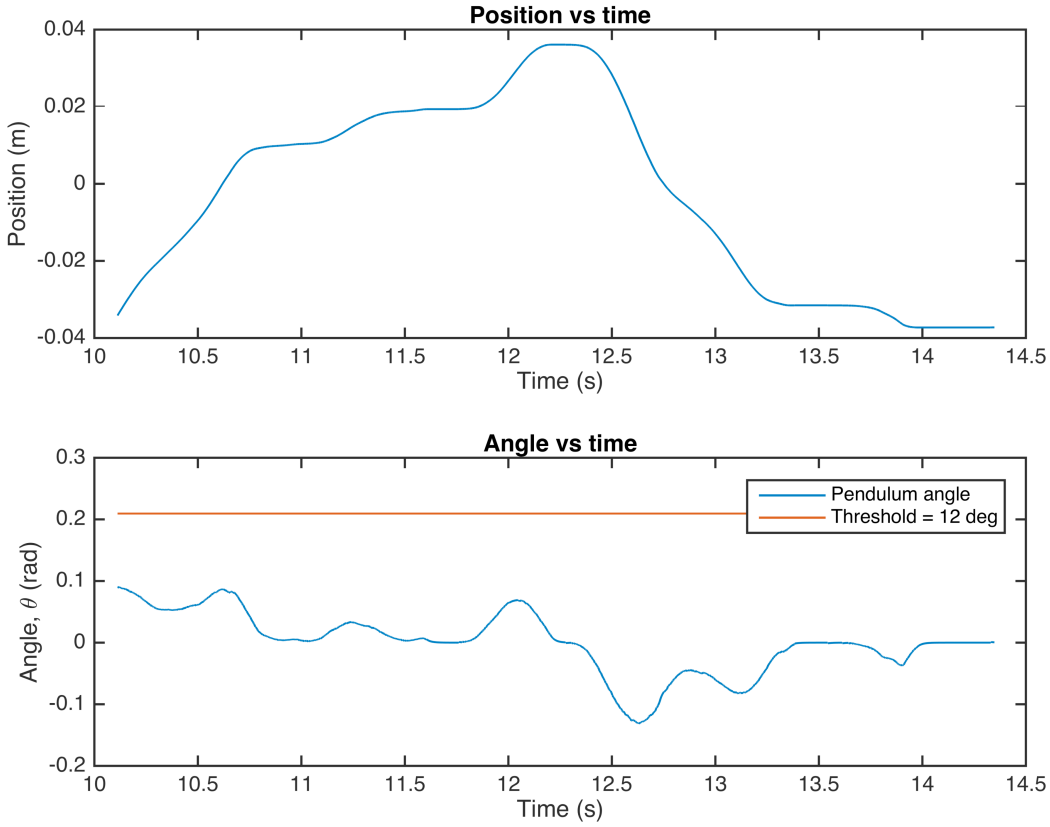


Figure 7: Cart position and pendulum angle

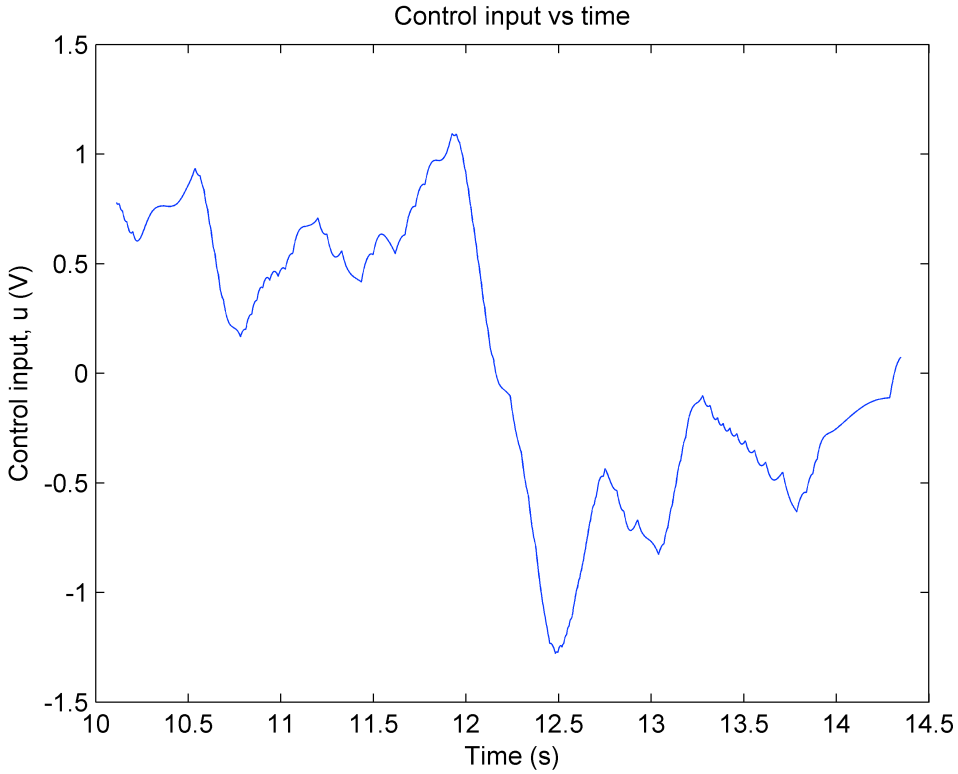


Figure 8: Control input vs. time

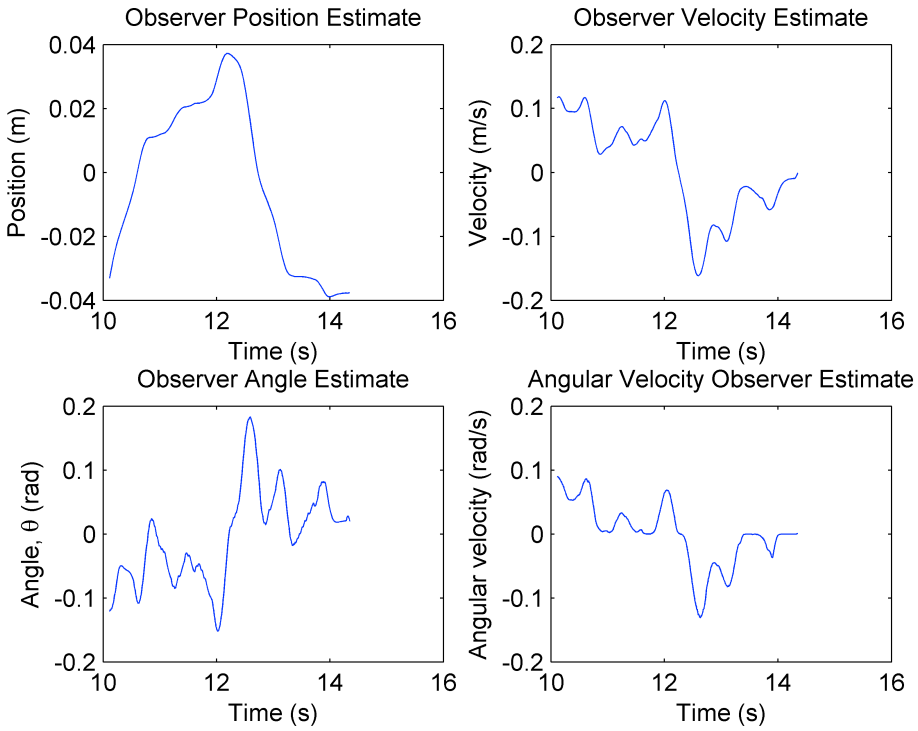


Figure 9: Observer state estimates $\hat{x}$

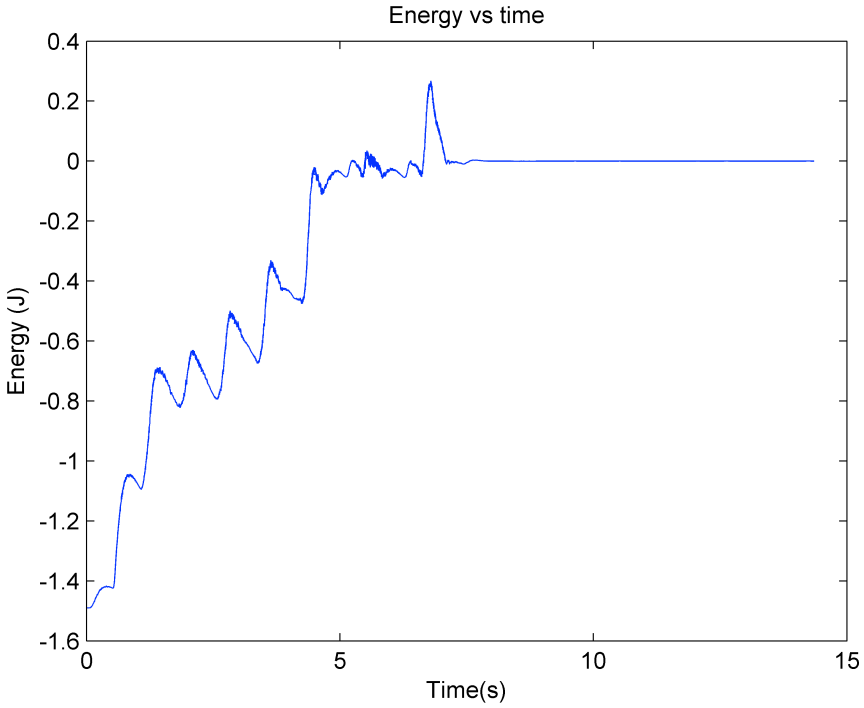


Figure 10: System energy vs time

From Figure 10, we see the "energy-pumping" controller in action until it reaches the desired steady-state value of 0 at around 7 seconds. This is when the switch from swing-up controller to stabilizing controller.

Implementation of the inverted pendulum in action can be found at:
https://youtu.be/6crAh9LXtlg

Appendix: MATLAB Code

```
%% Initialize system parameters (always load before running other
sections)
M = 0.94;
m = 0.23;
Lp = 0.3302;
Kt = 7.67e-3;
Km = Kt;
Kg = 3.71;
Rm = 2.6;
r = 6.36e-3;
Jm = 3.9e-7;
g = 9.81;
J = (4/3)*m*Lp^2;

c1 = Kg^2*Kt*Km/(r^2*Rm);
c2 = 1/(M + 0.25*m + Kg^2*Jm/r^2);
c3 = Kg*Kt/(r*Rm);

A = [0, 1, 0, 0; 0, -c2*c1, -0.75*m*g*c2, 0; 0 0 0 1; 0,
0.75*(1/Lp)*c2*c1, (1 + 0.75*m*c2)*(0.75*g/Lp) 0];
B = [0; c2*c3; 0; -0.75*c2*c3/Lp];
C = [1 0 0 0; 0 0 1 0];
D = [0;0];
plant_sys = ss(A,B,C,D);

p = [-1.9+10j, -1.9-10j, -1.6+1.3j, -1.6-1.3j];
K = acker(A, B, p);

Lpoles = [-10+15j -10-15j -12+17j -12-17j];
L_trans = place(A', C', Lpoles);
L = L_trans';

c_lp = sqrt(2*Lp/g)*tand(5);
thresh = 12*pi/180;
Kpump = 100;
start_time = 0.05;
SAT = 1;
r = [0; 0; 0; 0];
```

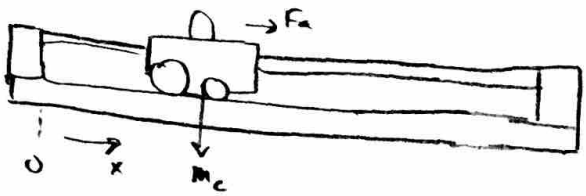
```
%% Plots
% Position and angle
figure
subplot(2,1,1)
plot(t, x(:,1));
xlabel('Time (s)'); ylabel('Position (m)'); title('Position vs time');

subplot(2,1,2)
plot(t, x(:,2)); hold on;
plot(t, thresh*ones(size(x(:,2))));
xlabel('Time (s)'); ylabel('Angle, \theta (rad)'); title('Angle vs
time');
legend('Pendulum angle', 'Threshold = 12 deg');

% Control input
figure
plot(t, u);
xlabel('Time (s)'); ylabel('Control input, u (V)'); title('Control input
vs time');

% Observer state estimates, xhat
figure
subplot(2,2,1)
plot(t, xhat(:,1));
xlabel('Time (s)'); ylabel('Position (m)'); title('Observer Position
Estimate');

subplot(2,2,2)
plot(t, xhat(:,2));
xlabel('Time (s)'); ylabel('Velocity (m/s)'); title('Observer Velocity
Estimate');

subplot(2,2,3)
plot(t, xhat(:,4));
xlabel('Time (s)'); ylabel('Angle, \theta (rad)'); title('Observer Angle
Estimate');

subplot(2,2,4)
plot(t, x(:,2));
xlabel('Time (s)'); ylabel('Angular velocity (rad/s)'); title('Angular
Velocity Observer Estimate');

% Energy
S = load('energy.mat');
energy_t = S.ans(1,:);
energy = S.ans(2,:);
figure
plot(energy_t, energy);
xlabel('Time(s)'); ylabel('Energy (J)'); title('Energy vs time');
```

PRE-LAB

ADOLFO TEC

4.1 Derive EOM: $(m_c r^2 R_m + R_m k_g^2 J_m)\ddot{x} + (k_t k_m k_g^2)\dot{x} = (r k_t k_g) V$

FBD



$\underline{r} = x\underline{E}_x + y_0\underline{E}_y + z_0\underline{E}_z$

$\underline{v} = \dot{x}\underline{E}_x$

$\underline{a} = \ddot{x}\underline{E}_x$

$\underline{F} = F_a\underline{E}_x - m_c g\underline{E}_y = m_c\ddot{x}\underline{E}_x$

$\cdot\underline{E}_x: \quad F_a = m_c\ddot{x}$

$\cdot\underline{E}_y: \quad -m_c g = 0$

$$\boxed{F_a = m_c\ddot{x}}$$

2. Given: $T_m = k_t I_m - J_m\ddot{\theta}$ (1)

Find relationship b/t V and Fa

$V = I_m R_m + E_{mf} = I_m R_m + k_m\dot{\theta}$ (2)

$k_g T_m = F_a \cdot r$ (3)

$k_g\dot{x} = \dot{\theta} r \Rightarrow k_g\ddot{x} = \ddot{\theta} r$ (4), (5)

(2): $F_a = \dfrac{k_g T_m}{r} = m_c\ddot{x}$

→ plug in (1): $F_a = \dfrac{k_g}{r}\left[k_t I_m - J_m\ddot{\theta}\right]$

→ rearrang (2): $I_m = \dfrac{V - k_m\dot{\theta}}{R_m}$

Combine:

$$F_a = \frac{k_g}{r}\left[k_t\cdot\frac{V - k_m\dot{\theta}}{R_m} - J_m\ddot{\theta}\right]$$

$$\rightarrow F_a = \frac{k_g}{r}\left[\frac{k_t V}{R_m} - \frac{k_t k_m\dot{\theta}}{R_m} - J_m\ddot{\theta}\right]$$

→ rearrange (4) and (5) and plug in:

$$\frac{k_g}{r}\left[\frac{k_t}{R_m}V - \frac{k_t k_m}{R_m}\cdot\frac{k_g}{r}\dot{x} - J_m\frac{k_g}{r}\ddot{x}\right] = F_a = m_c\ddot{x}$$

$$\Rightarrow \frac{m_c r}{k_g}\ddot{x} + \frac{J_m k_g}{r}\ddot{x} + \frac{k_t k_m k_g}{R_m r}\dot{x} = \frac{k_t}{R_m}V$$

multiply by $(k_g \cdot R_m r)$: $\Rightarrow m_c r^2 R_m\ddot{x} + k_g^2 R_m J_m\ddot{x} + k_t k_m k_g^2\dot{x} = k_t k_g r V$ linear!

$$\Rightarrow \boxed{(m_c r^2 R_m + R_m k_g^2 J_m)\ddot{x} + (k_t k_m k_g^2)\dot{x} = (r k_t k_g)V} \checkmark$$

4.2

1. Apply Laplace transformation (Assume 0 initial cond):

$$\left[\left(m_c r^2 R_m + R_m k_g^2 J_m\right) s^2 + \left(k_t k_m k_g^2\right) s\right] X(s) = \left(r k_t k_g\right) V(s)$$

$$H(s) = \frac{X(s)}{V(s)} = \frac{r k_g k_t}{\left(m_c r^2 R_m + R_m k_g^2 J_m\right) s^2 + \left(k_t k_m k_g^2\right) s}$$

2 State Space Model: $x = x_1$

$$v = \dot{x} = \dot{x}_1 = x_2$$

$$y = x_1$$

$$u = V$$

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-k_t k_m k_g^2}{m_c r^2 R_m + R_m k_g^2 J_m} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{r k_g k_t}{m_c r^2 R_m + R_m k_g^2 J_m} \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]\, u$$

$$\Rightarrow A = \begin{bmatrix} 0 & 1 \\ 0 & \frac{-k_t k_m k_g^2}{m_c r^2 R_m + R_m k_g^2 J_m} \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{r k_g k_t}{m_c r^2 R_m + R_m k_g^2 J_m} \end{bmatrix}, \quad C = [1 \;\; 0], \quad D = [0]$$

3 $G(s) = C(sI - A)^{-1} B + D$ 20

$$(sI - A) = \begin{bmatrix} s & 1 \\ 0 & s + \frac{k_t k_m k_g^2}{m_c r^2 R_m + R_m k_g^2 J_m} \end{bmatrix}$$

⋮

SEE CODE. VERIFIED ✓

3

According to MATLAB, $k = 15.7$

Pre Lab 3

Table of Contents

4.2 SS to TF

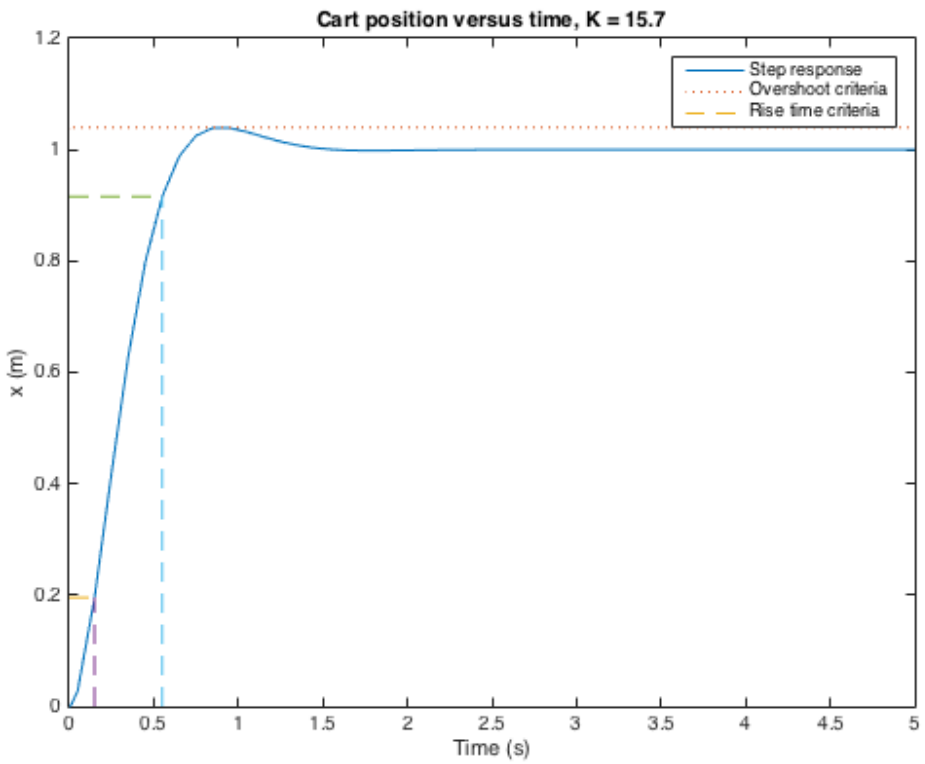
```
syms s Kt Km Kg mc r Rm Jm
A = [0 1; 0 -(Kt*Km*Kg^2)/(mc*r^2*Rm + Rm*Kg^2*Jm)];
B = [0; (r*Kg*Kt)/(mc*r^2*Rm + Rm*Kg^2*Jm)];
C = [1 0];
I = eye(2);

G = C*inv(s*I-A)*B
```

```
G =

(Kg*Kt*r)/(s*(Kg^2*Km*Kt + Jm*Kg^2*Rm*s + Rm*mc*r^2*s))
```

4.3 MATLAB Step Response

```
mc = 0.94; r = 6.36e-3; Rm = 2.6; Kt = 7.67e-3; Km = 7.67e-3; Kg =
 3.71; Jm = 3.9e-7;

% Let us find K that achieves percent max overshoot <4.0% and rise
 time <
% 1.5s
K_sample = [30:-0.1:1];
for i = 1:length(K_sample)

    K = K_sample(i);
    sim('Prelab_4_3'); %Run simulation for different K values

    % Calculate percent overshoot for each K tested
    povershoot = max(x.data - 1)*100;

    % Calculate rise time
    x1 = find(x.data>=0.1, 1); %Probably never get 0.1 exactly, so
 find first instance after 0.1
    x2 = find(x.data>=0.9, 1); %Same argument for x2
    tr = x.time(x2) - x.time(x1);

    % If we get a K value where overshoot and rise time criteria are
 met,
    % break
```

```
    if (povershoot<4 && tr<1.5)
        break;
    end
end

plot(x.time, x.data); hold on;
xlabel('Time (s)'); ylabel('x (m)');
title(['Cart position versus time, K = ' num2str(K)]);

%Plot percent overshoot and rise time criteria
t1 = x.time(x1); t2 = x.time(x2);
x_crit = 1.04*ones(size(x.data));

plot(x.time, x_crit, ':');
plot([0, t1], [x.data(x1), x.data(x1)], '--', [t1, t1], [0,
 x.data(x1)], '--')
plot([0, t2], [x.data(x2), x.data(x2)], '--', [t2, t2], [0,
 x.data(x2)], '--')

legend('Step response', 'Overshoot criteria', 'Rise time criteria');
```



Published with MATLAB® R2015a