

Lab 4: Model-Based Position Control of a Cart

Name: Mingjun Wu, Matthew Domine

Date: 4/8/2021

Objectives

The goal of this lab is to help understand the methodology to design a controller, given plant dynamics. Specifically, we will do position control of a cart by developing various controllers and then comparing their performance.

Equipment

Quanser cart system (no attachments) and power supply. See Lab 3 for details.

Theory

1. Plant Dynamics

The plant in this lab is the same as in Lab 2. Recall its dynamics

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

where the parameters are described in Table 1.

Parameter	Value	Unit	Description
V	-	V	input voltage
m_c	0.94	kg	mass of the car
r	6.36×10^{-3}	m	radius of the motor gears
R_m	2.6	Ω	resistance of the motor windings
K_t	7.67×10^{-3}	Nm/A	torque motor constant
K_m	7.67×10^{-3}	Vs/rad	back EMF constant
K_g	3.71	-	gearbox ratio
J_m	3.9×10^{-7}	kg m ²	moment of inertia of the motor

Table 1: Parameters of the cart system

2. Second order Dynamics

A second order linear system is described by the general differential equation of the form

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

where $u(t)$ is the input to the system. The parameter ω_n is called the natural frequency and is a measure of the speed of the response of the second order system while ξ is a measure of the (viscous) damping in the system

When expressed in the Laplace domain, Equation (2) reads

$$\frac{Y(s)}{U(s)} = \frac{b}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad (3)$$

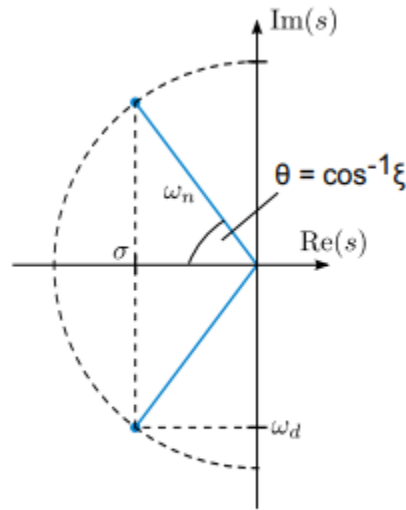


Figure 1: Location of the poles of (3) in terms of ω_n and ξ .

For $\xi \leq 1$, the above system has complex poles, which are depicted in the complex plane in Figure 1. For complex poles in the left half plane, we make the following important observations from Figure 1:

- The length of the complex vector from the origin to the pole is ω_n
- The cosine of the angle of the vector with the negative real axis equals ξ , i.e. $\cos \theta = \xi$

A typical step response for a second order system of the form (3) is shown in Figure 2. Two important

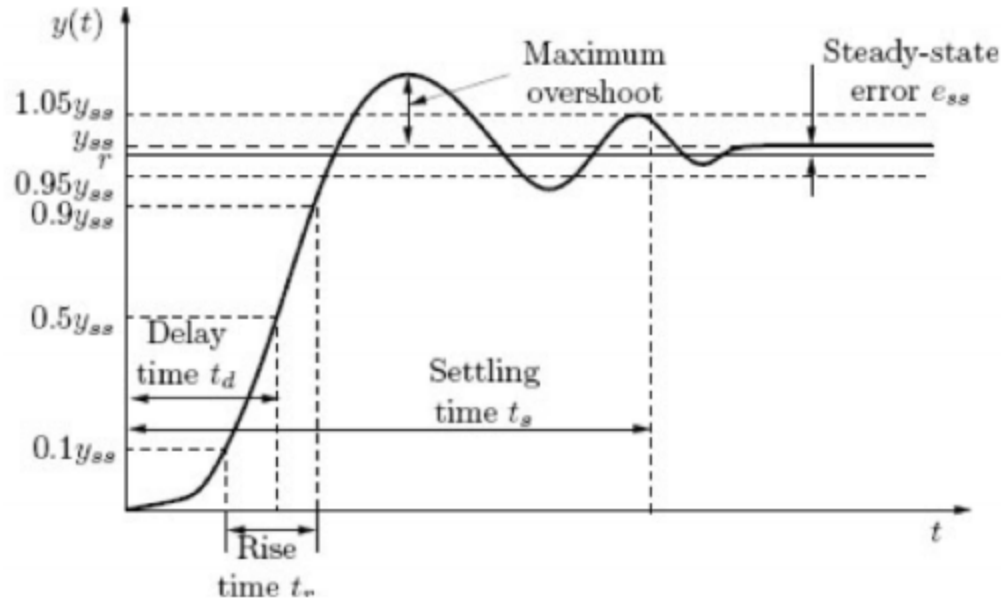


Figure 2: Typical step response of a linear second order system

performance metrics for a SISO system is its rise time and maximum overshoot. For a second order system as in (3) the overshoot can be determined analytically and is given by

$$M_p = e^{-\pi\xi/\sqrt{1-\xi^2}} \quad (4)$$

There is no explicit formula for the rise time as a function of the parameters ω_n and ξ . However, the following third order polynomial provides an approximation with maximum error $< 0.5\%$ for values $0 < \xi < 0.9$ (see Nise, page 181):

$$\omega_n t_r \approx 1.76 \xi^3 - 0.417 \xi^2 + 1.039 \xi + 1 \quad (5)$$

In order to get a fast response (small t_r) and small overshoot (small M_p) we would like the closed loop poles to have large radial distance (leading to a large natural frequency ω_n) and large angle θ (leading to a high damping ξ).

Analysis and Results

Pre-Lab

1. Position Controller Design

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

1. Rise time $t_r \leq 0.2$ s (This is several times faster than Pre-Lab 3)

2. Maximum overshoot $M_p \leq 5\%$

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

No the amplitude will not affect the rise time nor the maximum overshoot. It is not considered in the calculation of the two (see equations 4 and 5).

The objective is to design a feedback system that will help us achieve these desired performance specifications. The general guidelines on how to proceed with the design are outlined below.

2. Plant Model

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

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

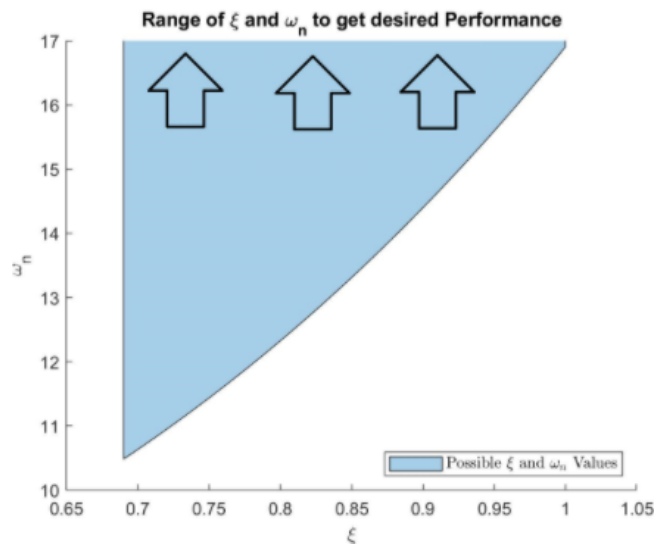
Poles: 0, -7.18

System is marginally stable

- Using the given performance objectives and equations (5) and (4), determine ranges for the desired values of ω_n and ξ .

$$\xi \geq 0.69$$

$$\omega_n \geq \frac{1.76\xi^3 - 0.417\xi^2 + 1.039\xi + 1}{0.2} \text{ (note that it varies as } \xi \text{ varies)}$$

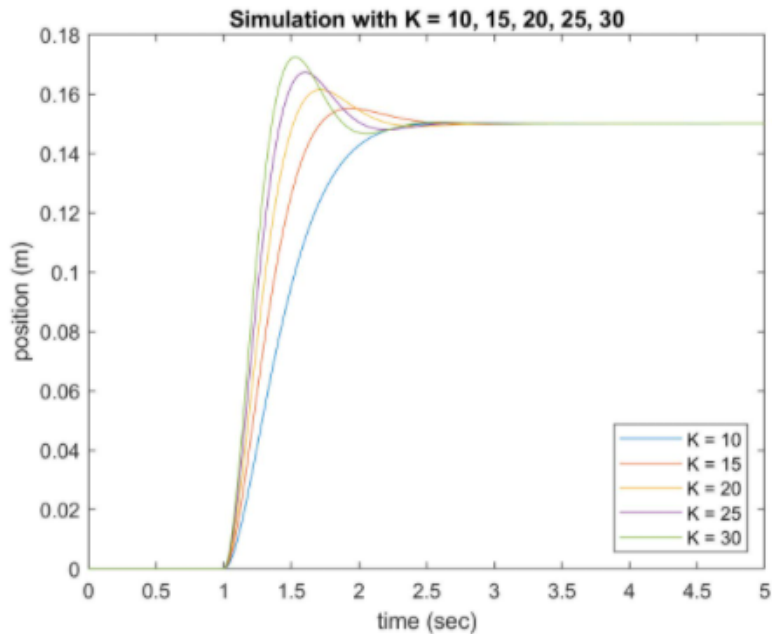


Note that range is not bounded from the top

3. Proportional Controller

The first controller that we will try is the proportional controller K from the previous lab. Feel free to use your Simulink diagram from the last lab, but please include it again in this lab report. **Make sure your system output is in meters and your system input is in Volts. Vary the value of K over the range 10-60.** Plot the step functions for five of these values superimposed in a single plot.

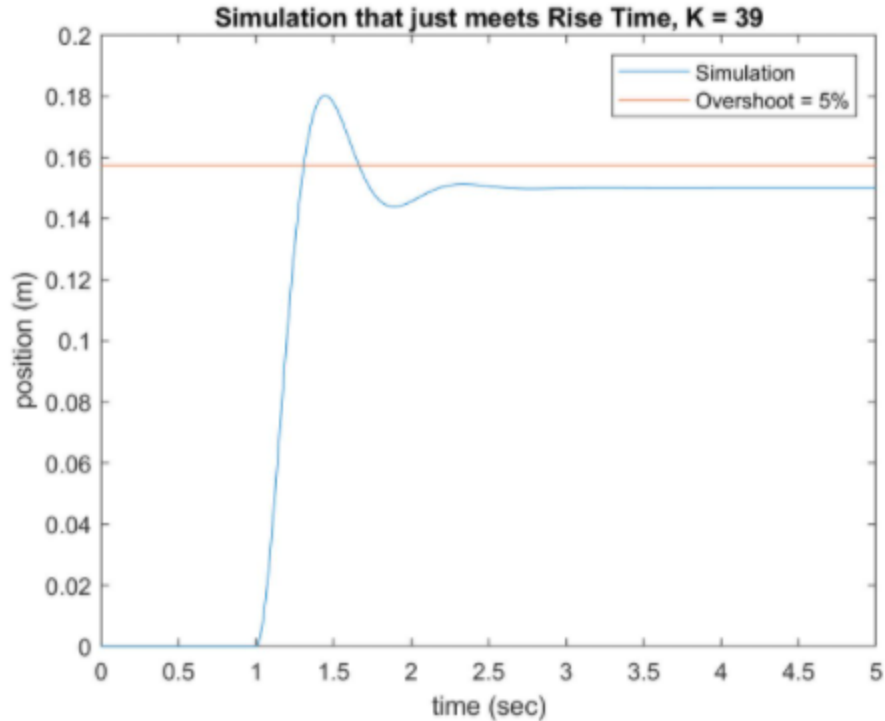
- As K increases, how does it affect the rise time and overshoot?
- With just a P controller, can the desired performance specifications be achieved?



- As K increases, rise time decreases and overshoot increases.
- The desired performance cannot be reached with just a proportional controller.

Find the smallest integer K value for which the rise time performance specification is met. Plot the response for this K and mention whether it meets the overshoot specification.

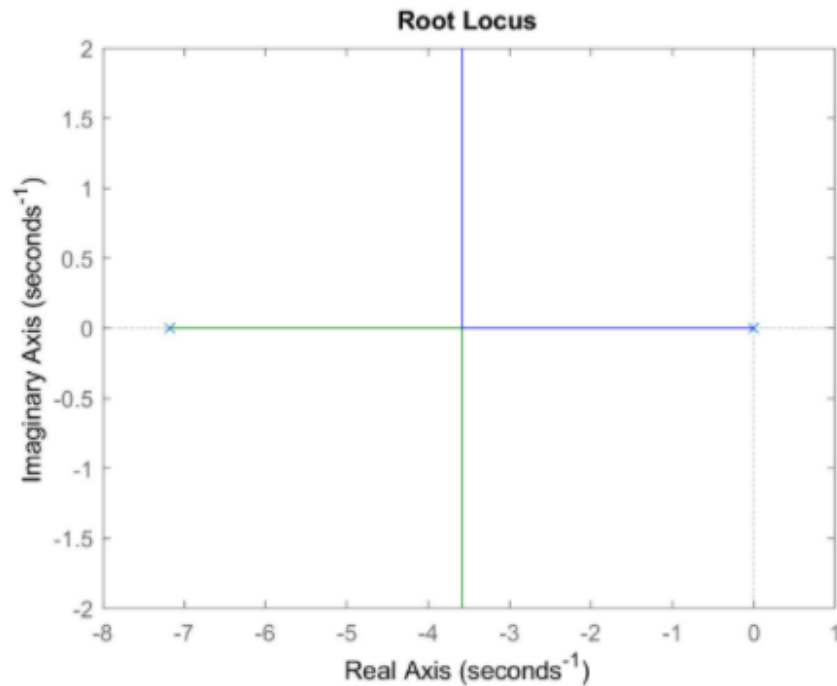
The smallest integer value of K that has a rise time less than 0.2 s is 39.



The above observations can also be made from the root locus plot. Plot the root locus for the plant transfer function and answer the following:

- For complex poles of the closed loop transfer function, as K increases what happens to the radial distance and the angle θ ?
- Going back to Figure 1 and equations (5) and (4), how does an increase in K affect ξ and ω_n ?

Root locus:



- As K increases, the complex poles' radial distance and angle increase.
- As K increases, ω_n increases and ξ decreases.

4. PD Controller

In order to meet the design constraints, we will need derivative action which will help “apply the brakes earlier.” We will determine the appropriate action through the following steps:

- Using the predetermined rise time and overshoot specifications, we determine the appropriate damping, ξ , and natural frequency ω_n .
- We will determine the appropriate controller values, K_d and K_p , using the root locus that correspond to the determined damping and natural frequency values.

In order to reduce the overshoot we use derivative action in conjunction with proportional action. This type of controller is called “PD Controller”. Its form is as follows:

$$k(t) = k_P(x_{ref} - x) + k_D(\dot{x}_{ref} - \dot{x}) = k_P e(t) + k_D \dot{e}(t) \quad (6)$$

where $e(t) = x_{ref}(t) - x(t)$ is the error.

Observe that a PD controller introduces a zero in the plant transfer function at $s = -k_P/k_D$. MATLAB does not allow you to put a pure zero. Why is introducing a pure zero a bad idea?

Introducing a pure zero is the same as introducing a derivative.

Instead, we will introduce a pole/zero pair with the pole so far away that it hardly affects the rest of the system dynamics. We choose to set the pole at $s = -250$, which corresponds to a denominator $1/250 s + 1$ of the controller¹.

Now put the plant dynamics equation from Step 1 in unity negative feedback with this new PD controller and obtain the transfer function from x_{ref} to x . For further analysis we assume that we will place the zero at $s = -12$, that is, $k_P k_D = 12$. This location of the zero can be found through design techniques which will be covered in subsequent labs. The numerator of the PD controller's transfer function becomes $k_D(s + 12)$. That is, the controller has the following dynamics:

$$H(s) = k_D \frac{s + 12}{\frac{1}{250}s + 1} \quad (7)$$

Plot the root locus for this transfer function of the “modified” plant which includes the controller (make sure to zoom in towards the origin, that's where the interesting things happen). By comparing this root locus with the one plotted in Step 3 make the following observations:

- A left half plane zero tends to pull the root locus towards it.
- There exists a portion of the root locus which has the following properties:
 - it has complex closed loop poles
 - ω_n and ξ both increase as k_D increases

This is a desired behavior, as we can increase the systems response speed (linked to ω_n) and at the same time decrease its overshoot (linked to the damping ξ).

Determine a small k_D value (less than 50) from this root locus for which the performance specifications are met. Hint: For this task, you will find two things handy: 1) You can specify gain values to plot on your root locus (see doc rlocus). 2) Once you plot your root locus, the Data Cursor (Tools → Data Cursor) will display the gain k_D , damping ξ , and frequency ω_n . You can drag this cursor across all the plotted points until you find a k_D value that meets the parameters you solved for in Step 2.

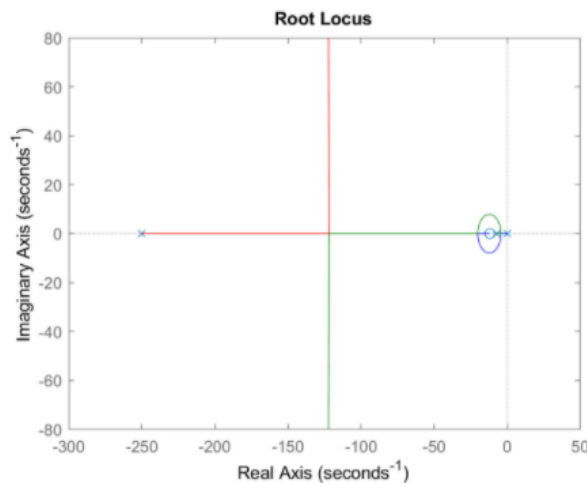
Now create a Simulink block diagram to simulate the cart in feedback with a PD controller. The zero is set at $s = -12$, i.e. the controller numerator is of the form $k_D(s + 12)$

- Verify that your chosen value of k_D meets the design specifications. Observe that there is a discrepancy in overshoot value between the root locus plot and simulation. This is due to the fact that rlocus approximates the value of rise time by assuming that the system is second-order with no zeros, i.e. it computes the theoretical value of overshoot if the system had two dominant poles and no zero.

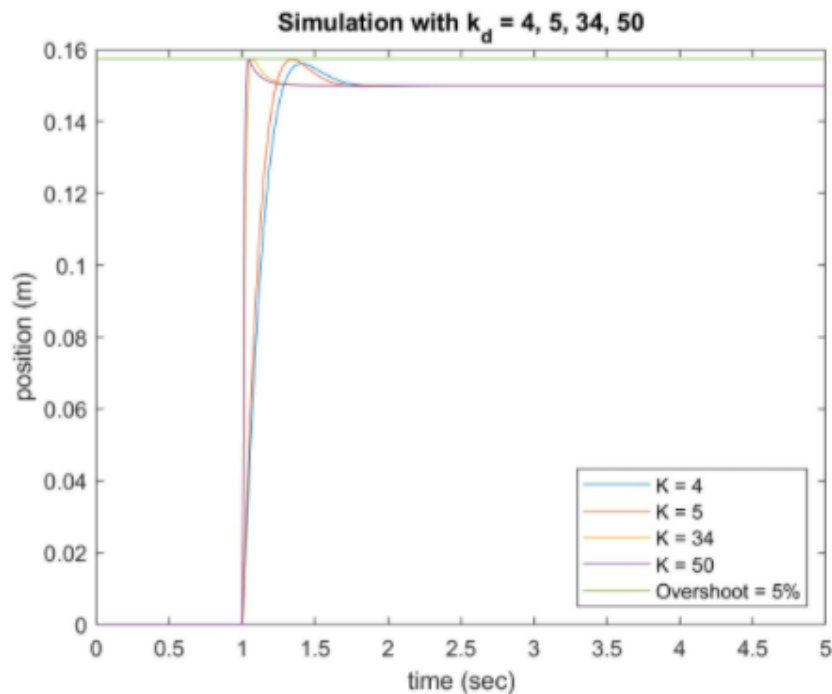
Our system has a zero at $s = -12$. This zero is close to the poles for the selected gain parameter, so it will affect the step response quite a bit.

- Find the range of values of the gain $k_D \leq 50$ for which the specifications are met. Plot the system response for 4 values of k_D in this range.
- To test the effect of the zero, change the zero to $k_P k_D = 15$. Can you find a value of k_D (less than 50) for which the system meets the specifications?

Root locus:



- Choosing $k_d = 4.24$, Matlab's root locus cursor gives an overshoot of 2.38%. However, the actual overshoot is 4.30%.
- Possible values of k_d to meet performance specifications: about 3.9 to 5.2 and from 33.6 to 50.



- Having a zero at 15 would not allow you to get the performance specifications needed.

Lab

1. Proportional Control

Experiment with the proportional controller on the “hardware” by trying out various gain values and observe the variation of rise time and overshoot. Provide the K value and plot the system response for the following:

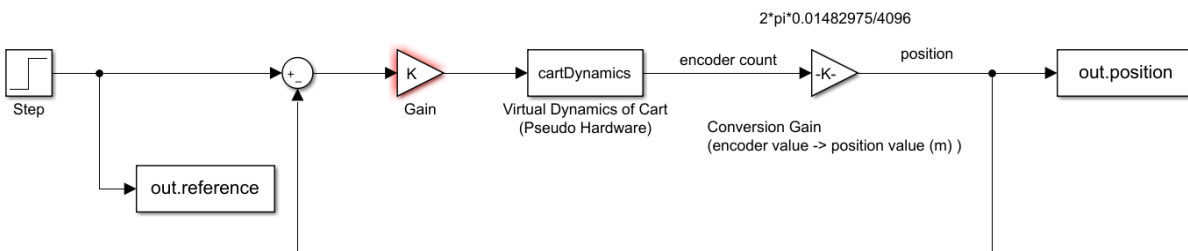


Figure 3: Simulink Block Diagram of the Closed Loop System w/ Proportional Controller & Hardware Block

- minimal K value such that the rise time is less than 0.2 s

```

%% Condition 1: Minimal Value of K for tr < 0.2s
clear,clc,close all;

n = zeros(1,91);
%Parameters
for K = 10:100

    %Simulation
    tstop = 10;
    out = sim('cartdynamicsh',tstop);
    position = out.position.data;
    time = out.tout;

    %Step info
    S = stepinfo(position,time); %display stepinfo

    if (S.RiseTime < 0.2)
        n(K)= K;
    end
end
minimal = min(nonzeros(n))
fprintf("K = %f is the minimal K value for tr < 0.2s\n",minimal)

K = minimal;
%Simulation
tstop = 10;
out = sim('cartdynamicsh',tstop);
reference = out.reference.data;
position = out.position.data;
time = out.tout;

S = stepinfo(position,time) %display stepinfo

%Plot
plot(time,position)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 0 0.18])
title(['Step Response of Cart System for K =',num2str(K)]) %Title
legend('Step Response of psuedoHardware','Reference Step Input')

```

Figure 4: Matlab Code for the Specified Condition

```

minimal =

    82

K = 82.000000 is the minimal K value for tr < 0.2s

S =

    struct with fields:

        RiseTime: 0.1997
        SettlingTime: 1.6326
        SettlingMin: 0.1328
        SettlingMax: 0.1638
        Overshoot: 11.2794
        Undershoot: 0
        Peak: 0.1638
        PeakTime: 1.4070

```

Figure 5: Matlab Code Step Response Info for $K = 82$

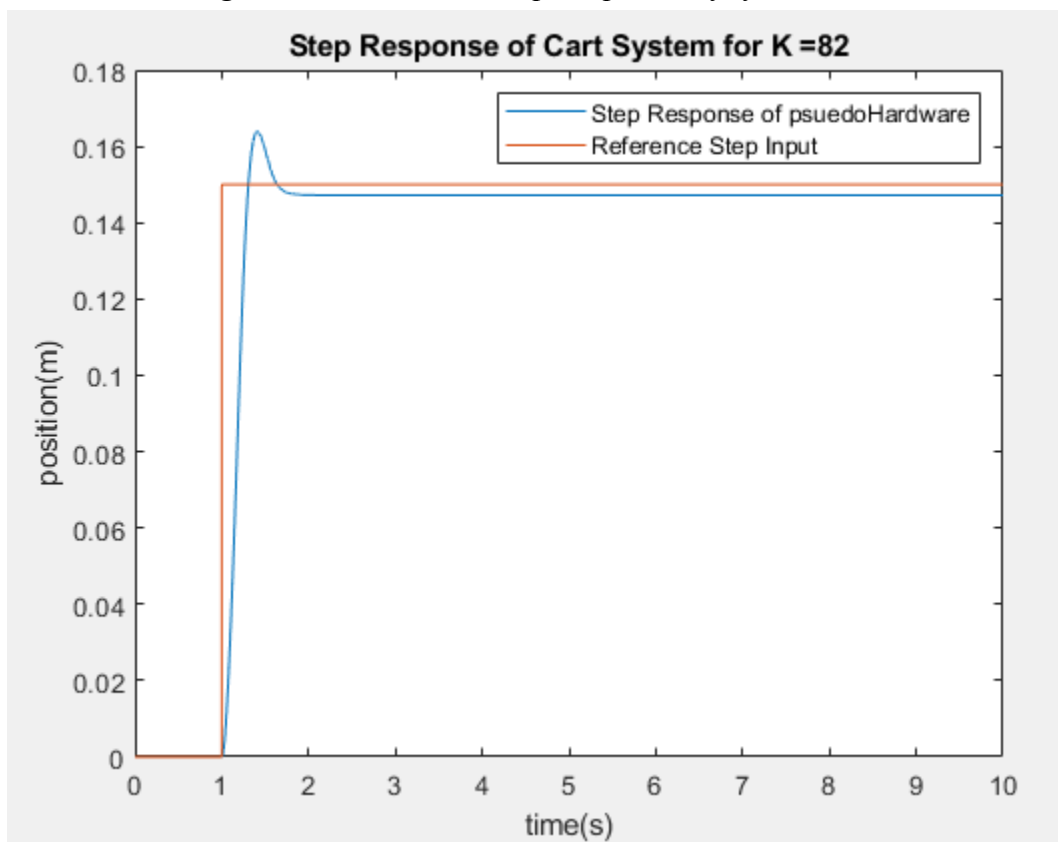


Figure 6: Hardware Step Response Plot for $K = 82$

The minimal K value such that the rise time is less than 0.2 s was evaluated to be **K = 82**.

- maximal K value such that the overshoot is less than 5%

```

%% Condition 2: Maximal Value of K for OS < 5
clear,clc,close all;

n = zeros(1,91);
%Parameters
for K = 10:100

    %Simulation
    tstop = 10;
    out = sim('cartdynamicsh',tstop);
    position = out.position.data;
    time = out.tout;

    %Step info
    S = stepinfo(position,time); %display stepinfo

    if S.Overshoot < 5
        n(K)= K;
    end
end

maximal = max(nonzeros(n))
fprintf("K = %f is the maximal K value for OS < 5\n",maximal)

K = maximal;
%Simulation
tstop = 10;
out = sim('cartdynamicsh',tstop);
reference = out.reference.data;
position = out.position.data;
time = out.tout;

S = stepinfo(position,time) %display stepinfo

%Plot
plot(time,position)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 0 0.18])
title(['Step Response of Cart System for K =',num2str(K)]) %Title
legend('Step Response of psuedoHardware','Reference Step Input')

```

Figure 7: Matlab Code for the Specified Condition

```

maximal =

    80

K = 80.000000 is the maximal K value for OS < 5

S =

    struct with fields:

        RiseTime: 0.2219
        SettlingTime: 1.4653
        SettlingMin: 0.1430
        SettlingMax: 0.1635
        Overshoot: 3.0833
        Undershoot: 0
        Peak: 0.1635
        PeakTime: 1.4100

```

Figure 8: Matlab Code Step Response Info for $K = 80$

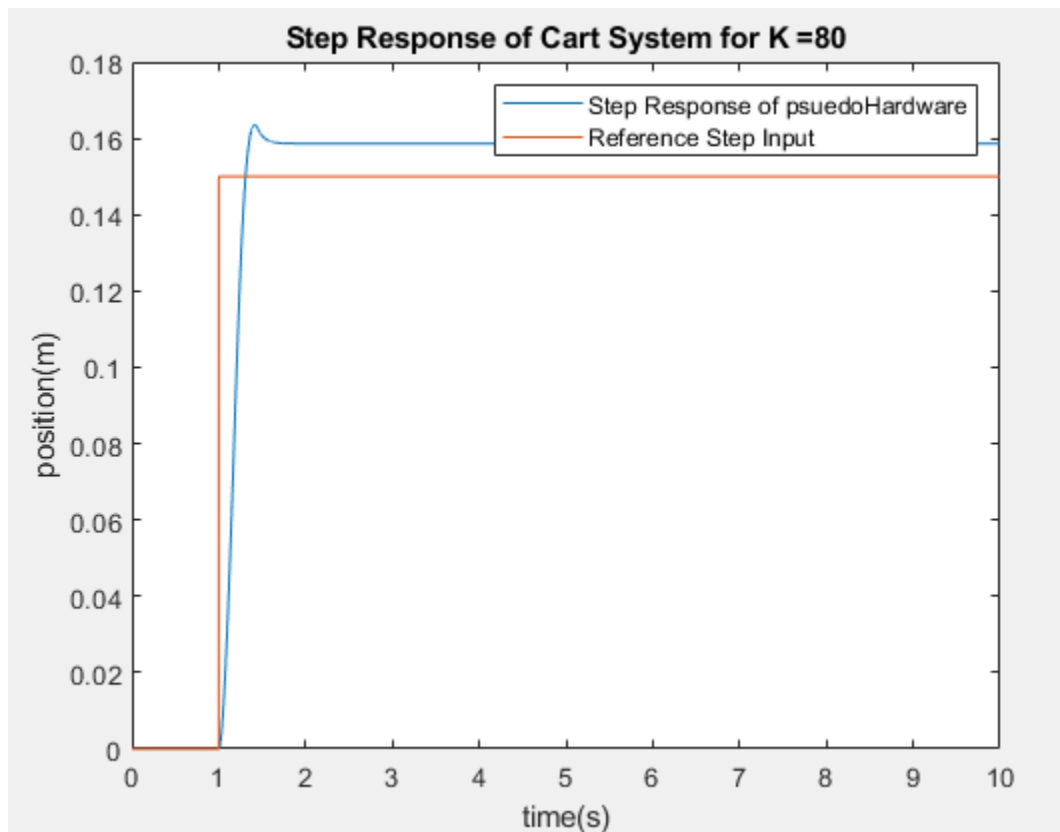


Figure 9: Hardware Step Response Plot for $K = 80$

- one value between these two

A value between these two would be **K = 81**.

```
%% Condition 3: K in between Conditions 1 & 2
clear,clc,close all;

%Parameters
K = 81;

%Simulation
tstop = 10;
out = sim('cartdynamicsh',tstop);
reference = out.reference.data;
position = out.position.data;
time = out.tout;

%Plot
plot(time,position)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 0 0.18])
title(['Step Response of Cart System for K =',num2str(K)]) %Title
legend('Step Response of psuedoHardware','Reference Step Input')

%Step info
S = stepinfo(position,time) %display stepinfo
```

Figure 10: Matlab Code for the Specified Condition

S =

struct with fields:

```

    RiseTime: 0.2004
SettlingTime: 1.6332
SettlingMin: 0.1331
SettlingMax: 0.1637
    Overshoot: 11.0682
    Undershoot: 0
        Peak: 0.1637
    PeakTime: 1.4100
```

Figure 11: Matlab Code Step Response Info for K = 81

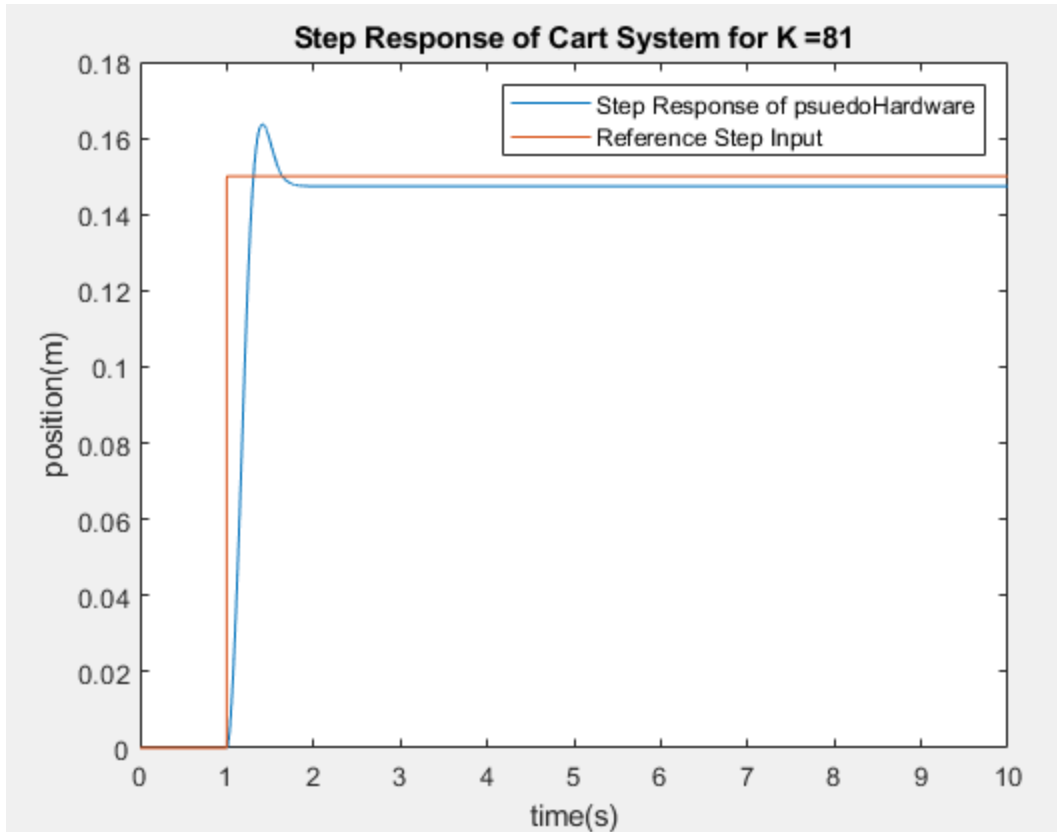


Figure 12: Hardware Step Response Plot for $K = 81$

You will likely observe that the system has a non-zero steady state error. Why do you think that is? We will come back to the steady-state error problem later. For now we focus on reducing the overshoot, by adding a derivative term to the controller.

Non-zero steady state errors stem from nonlinearities within the physical system which is not considered in the theoretical model which assumes they are negligible. In a broad sense two main resources for steady errors are: Non linearities in the system and the Configuration or the type of input to the system.

It is observed that no steady state error in the system model and the same step input is used so input to the system does not cause the steady-state error in this case. Non linearities and imperfections in the system components such as static friction, aging or deterioration, inductance, gearbox backlash (a sudden strong response), motor to gearbox efficiency and other physical limitations are considered in the hardware block. These are the factors that could cause the steady state error in the system.

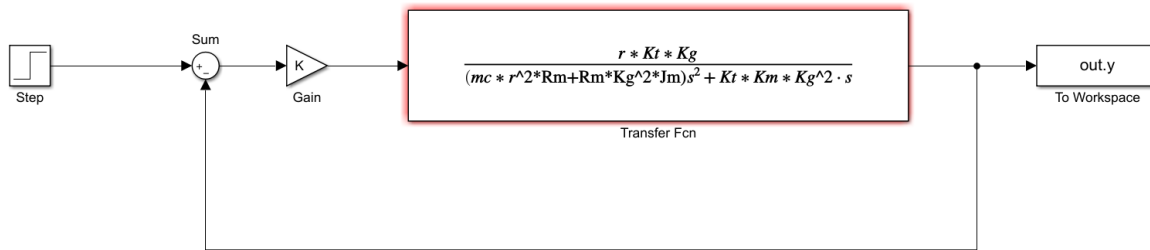


Figure 13: Simulink Block Diagram of the Closed Loop System w/ Proportional Controller & System Model Block

```
%% Condition 4: K = 100
clear,clc,close all;
% Define Parameters
mc = 0.94; %[kg]
r = 6.36 * 10^(-3); %[m]
Rm = 2.6; %[Ohms]
Kt = 7.67*10^(-3); %[Nm/A]
Km = 7.67*10^(-3); %[Vs/rad]
Kg = 3.71;
Jm = 3.9 * 10^(-7); %[kgm^2]
K = 100;

%Simulation
tstop = 10;
out2 = sim('cartdynamics',tstop);
out = sim('cartdynamicsh',tstop);
reference = out.reference.data;
position = out.position.data;
position2 = out2.y.data;
time = out.tout;

%Plot
plot(time,position)
hold on
plot(time,position2)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 0 0.25])
title(['Step Response of Cart System for K = ',num2str(K)]) %Title
legend('Step Response of psuedoHardware','Step Response of the System Model','Reference Step Input')

%Step info
S = stepinfo(position,time) %display stepinfo
S2 = stepinfo(position2,time) %display stepinfo
```

Figure 14: Matlab Code for the Specified Condition

```

S =
    struct with fields:
        RiseTime: 0.1916
        SettlingTime: 1.6181
        SettlingMin: 0.1309
        SettlingMax: 0.1660
        Overshoot: 14.1762
        Undershoot: 0
        Peak: 0.1660
        PeakTime: 1.3880

S2 =
    struct with fields:
        RiseTime: 0.1027
        SettlingTime: 2.0858
        SettlingMin: 0.1266
        SettlingMax: 0.2093
        Overshoot: 39.5267
        Undershoot: 0
        Peak: 0.2093
        PeakTime: 1.2590

```

Figure 15: Step info for hardware response (left) and system model response (right)

Plot the response of the system for $K = 100$, and compare it to the system model from the pre-lab. You will observe in particular that the rise time is longer for the actual system. Give reasons for why this might be the case.

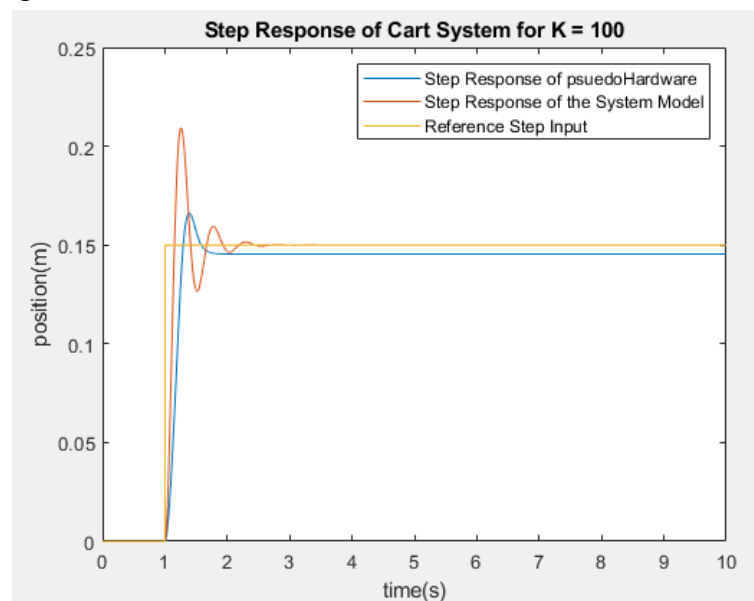


Figure 16: Hardware and Model Step Response Plot for $K = 100$

If we consider the percent error between rise times, there is a 88.4% difference in comparing the rise time of the hardware to the model. The rise time for the hardware response is greater than the model response since:

- Physical Limitations or imperfections of the system are considered in the hardware (i.e. friction, backlash, inductance, efficiencies, etc.)
- Due to these physical limitations there are effects of nonlinearities such as backlash and saturation effects which not only can cause the rise time to be greater but also result in a non zero steady state error.

The simulink model is faster and hence smaller rise time since things like friction and drag effect are not considered.

2. PD Control

Now implement the PD controller from Step 4 of the Pre-lab that satisfies the design constraints in theory. The controller is of the form $k_d \frac{s+12}{s/250+1}$. When you run it on the hardware for k_D in the range computed in the pre-lab, are the desired performance specifications met? Include a plot of your hardware response.

Possible values of k_d to meet performance specifications for the pre-lab: about 3.9 to 5.2 and from 33.6 to 50.

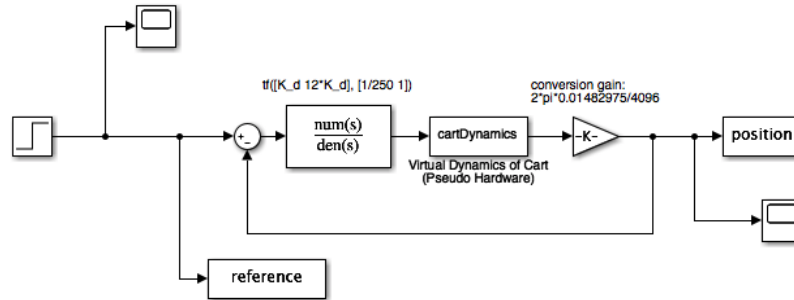


Figure 17: Simulink block diagram of the PD Control With Hardware Plant

```
%Plot the system response for 4 values of kD in the range from the pre-lab
array = [4 5 4.5 34 50];
figure;
for i = 1:length(array)
    K_d = array(i);
    open_system('lab4_5_1.mdl');
    sim('lab4_5_1.mdl');
    plot(position.Time, position.Data);
    hold on
    info = stepinfo(position.Data, position.Time);
    Overshoot = info.Overshoot;
    risetime = info.RiseTime;
    disp(['The Rise Time for K = ', num2str(K_d), ' is: ', num2str(risetime), ' seconds'])
    disp(['The Percent Overshoot for K = ', num2str(K_d), ' is: ', num2str(Overshoot), '%'])
end
title('Proportional Controller Step Response with Varying K');
axis([0 1.5 0 0.2])
xlabel('Time (s)');
ylabel('Position (m)');
legend('K = 4', 'K = 5', 'K = 4.5', 'K = 34');
```

Figure 18: Matlab Code of Plotting the responses of the 4 values of K_d

```
The Rise Time for K = 4 is: 0.38692 seconds
The Percent Overshoot for K = 4 is: 0%
The Rise Time for K = 5 is: 0.34438 seconds
The Percent Overshoot for K = 5 is: 0%
The Rise Time for K = 34 is: 0.24472 seconds
The Percent Overshoot for K = 34 is: 0%
The Rise Time for K = 50 is: 0.2432 seconds
The Percent Overshoot for K = 50 is: 0%
```

Figure 19: Percent Overshoot and Rise Time of the responses for the 4 values of K_d

As shown in the results above, all the four k_D values in the range computed in the pre-lab do not meet the desired performance specifications, and hence one could conclude that the range computed in the pre-lab do not meet the desired performance specifications.

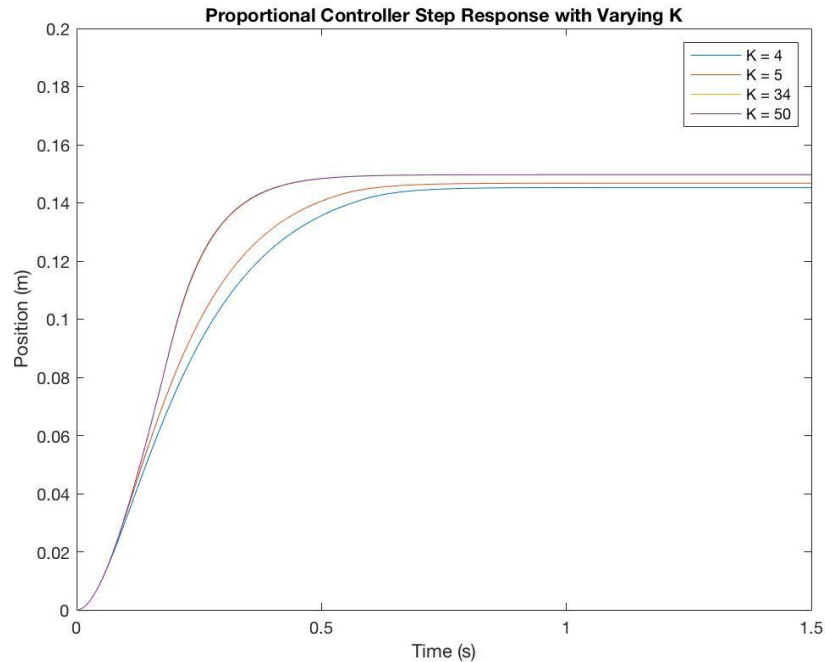


Figure 20: Plot of the Hardware Responses with 4 K_d Values

If the constraints are not met, then do an “intelligent” tuning of the gain values K_d and K_p until the required specifications are met. By now you should know different tuning methods and their general effect: modifying k_D and the zero $\frac{k_p}{k_D}$. Show a plot of your final run and show that it meets the design specifications. Don’t forget to report your final controller values.

Since the constraints are not met, then I did an “intelligent” tuning of the gain values K_d and K_p until the required specifications were met. Theoretically, increasing K_d would reduce the overshoot of the response and increasing K_p results in decreasing the rise time.

```

step = linspace(1,50,50);
K_range = [];
for K_d = step
    open_system('lab4_5_1.mdl')
    sim('lab4_5_1.mdl');
    info = stepinfo(position.Data, position.Time);
    Overshoot = info.Overshoot;
    risetime = info.RiseTime;
    if risetime <= 0.2 && Overshoot/100 <= 0.05
        K_range = [K_range K_d];
    end
end
disp(['The range of acceptable K_d values are from ', num2str(K_range(1)), ' to ', num2str(K_range(end))])

```

Figure 21: Matlab Code of the “Intelligent” Tuning

I changed the value of $\frac{k_p}{k_d}$ to 20 and then ran the matlab code, the range of Kd from 1 to 50 which meets the required specifications was not found. As I changed the value of $\frac{k_p}{k_d}$ to 30 and ran the code again, the range was computed:

The range of acceptable K_d values are from 8 to 50

I picked Kd = 10 as my final value.

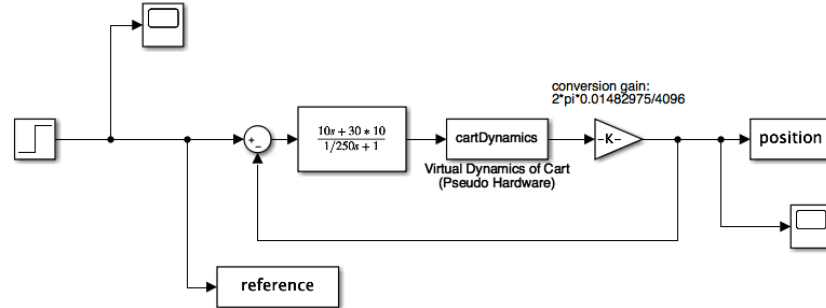


Figure 22: Simulink block diagram of the PD Control with Tuned Controller Values

```

open_system('lab4_5_1.mdl')
sim('lab4_5_1.mdl')
plot(position.Time, position.Data);
title('Proportional Controller Step Response with Kd = 10 and Kp/Kd = 30');
axis([0 10 0 0.2]);
xlabel('Time (s)');
ylabel('Position (m)');
legend('Step Response');
info = stepinfo(position.Data, position.Time)

```

info =

struct with fields:

RiseTime:	0.1966
SettlingTime:	0.3802
SettlingMin:	0.1364
SettlingMax:	0.1550
Overshoot:	2.6046
Undershoot:	0
Peak:	0.1550
PeakTime:	0.3480

Figure 23: Matlab Code of Plotting the Response

Figure 24: Output Result of the Step Response

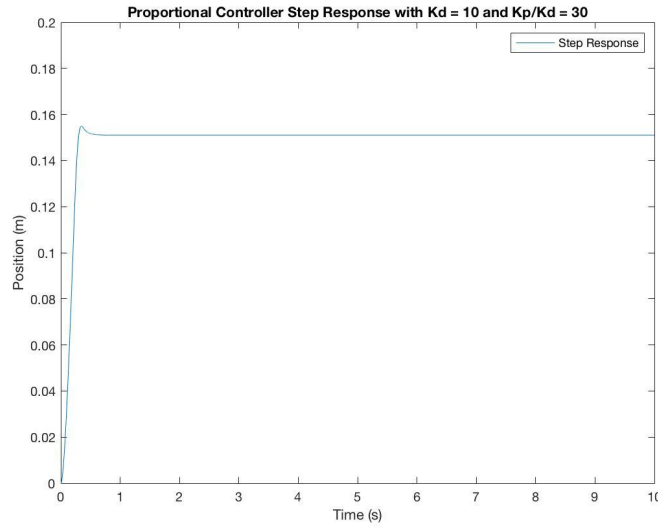


Figure 25: Plot of the Step Responses with $K_d = 10$ and $K_p/K_d = 30$

Final Value of $\frac{k_p}{k_d}$: 30

Final value of K_d : 10

3. Performance for Different Input Signals

Using your tuned PD controller from part 5.2, what is the performance like for the following input signals? Make sure your simulation time is finite and ≤ 20 (in case something goes wrong).

- Pulse generator (50% pulse width, amplitude ≤ 7.5 cm, period ≥ 3 sec)
- Sine wave (amplitude ≤ 10 cm, frequency ≤ 1 Hz)
- Saw/Triangle wave (amplitude ≤ 5 cm, frequency ≤ 0.5 Hz)
- Exponentially-decaying sine wave (initial amplitude ≤ 15 cm, pick a reasonable time constant and frequency so that the decay is evident, but not too sudden or too gradual)

Include your modified Simulink models as well as plots of the input signal and the actual hardware output (superimposed) for at least 2 periods.

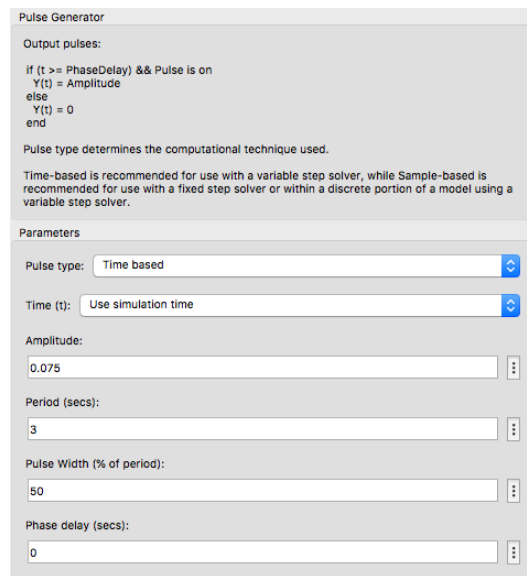
```

plot(tout, position.Data);
hold on
plot(tout, reference.Data);
legend('Output Response', 'Input Response (Tracking)');
xlabel('Time (s)');
ylabel('Position (m)');
axis([0 18 0 0.1]);
title('Response with Kd = 10 and Kp/Kd = 30');

```

Figure 26: Matlab Code to Plot out input signal and the actual hardware output (superimposed) for at least 2 periods.

- Pulse generator (50% pulse width, amplitude ≤ 7.5 cm, period ≥ 3 sec)



The image shows the 'Pulse Generator' block parameters in a Simulink environment. The 'Parameters' section is expanded, showing the following settings:

- Pulse type:** Time based
- Time (t):** Use simulation time
- Amplitude:** 0.075
- Period (secs):** 3
- Pulse Width (% of period):** 50
- Phase delay (secs):** 0

The 'Output pulses' section contains the following MATLAB code:

```

if (t >= PhaseDelay) && Pulse is on
    Y(t) = Amplitude
else
    Y(t) = 0
end

```

Below the code, a note states: 'Pulse type determines the computational technique used. Time-based is recommended for use with a variable step solver, while Sample-based is recommended for use with a fixed step solver or within a discrete portion of a model using a variable step solver.'

Figure 27: Impulse Generator Setting

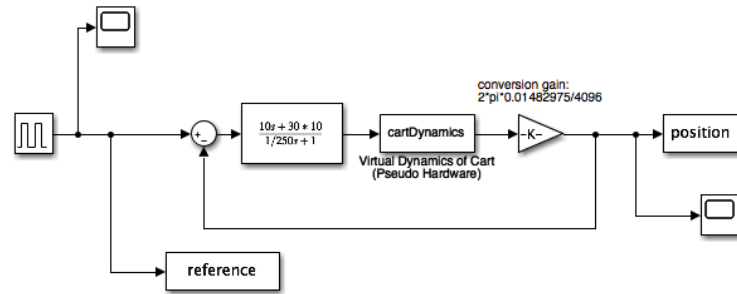


Figure 28: Simulink block diagram of the PD Control with Tuned Controller Values and Impulse Generator Input

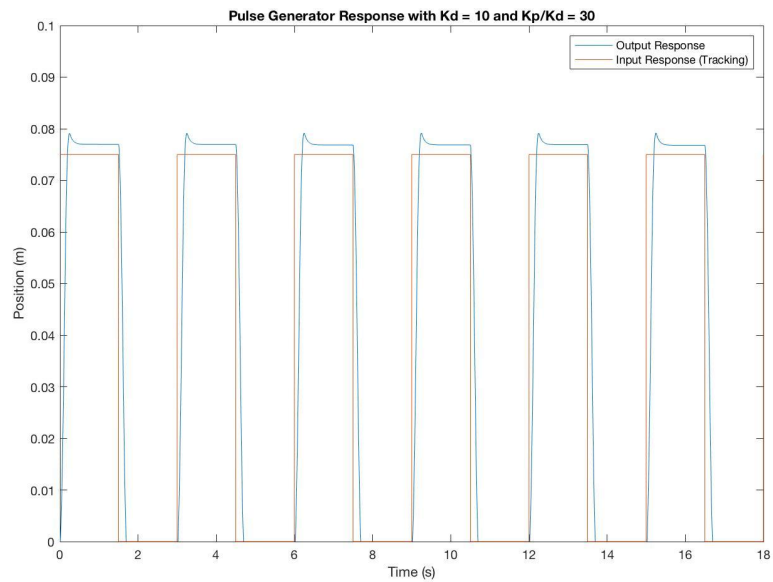


Figure 29: Plot of the Pulse Generator Response

- Sine wave (amplitude ≤ 10 cm, frequency ≤ 1 Hz)

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 0.1

Bias: 0

Frequency (rad/sec): 2π

Phase (rad): 0

Sample time: 0

Figure 30: Sine Wave Setting

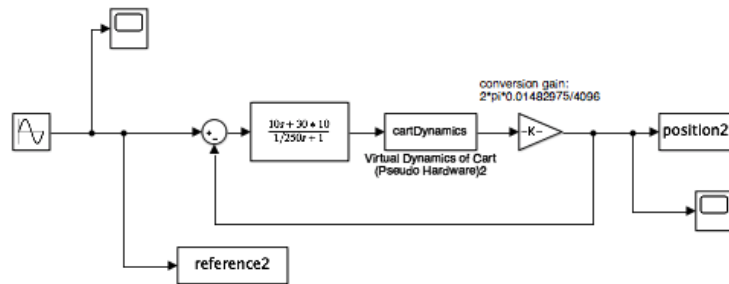


Figure 31: Simulink block diagram of the PD Control with Tuned Controller Values and Sine Wave Input

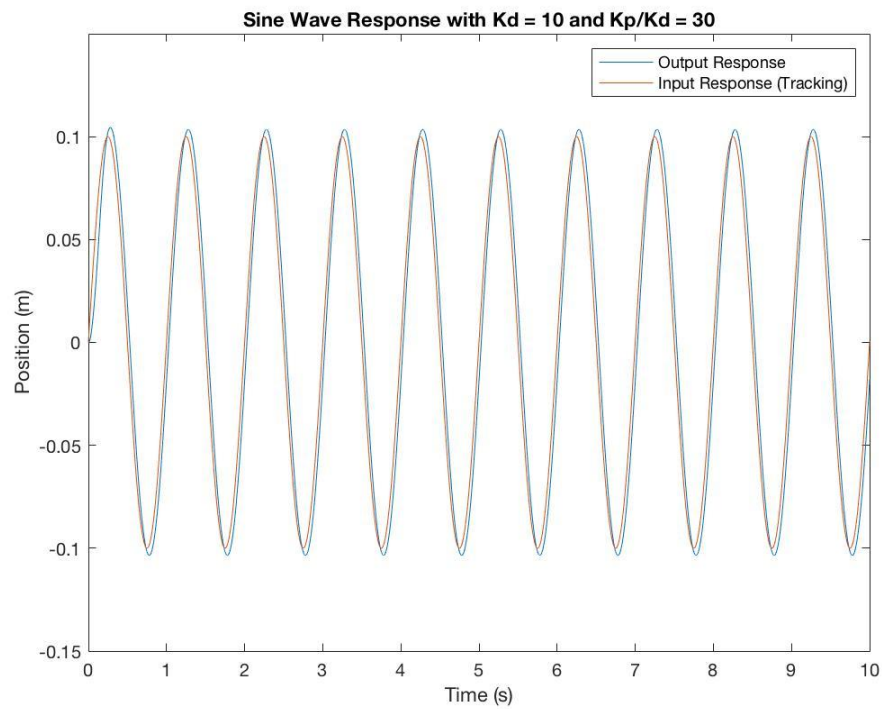


Figure 32: Plot of the Sine Wave Response

- Saw/Triangle wave (amplitude ≤ 5 cm, frequency ≤ 0.5 Hz)

Parameters

Wave form: sawtooth

Time (t): Use simulation time

Amplitude: 0.05

Frequency: 0.5

Units: Hertz

☒ Interpret vector parameters as 1-D

Figure 33: Saw Wave Setting

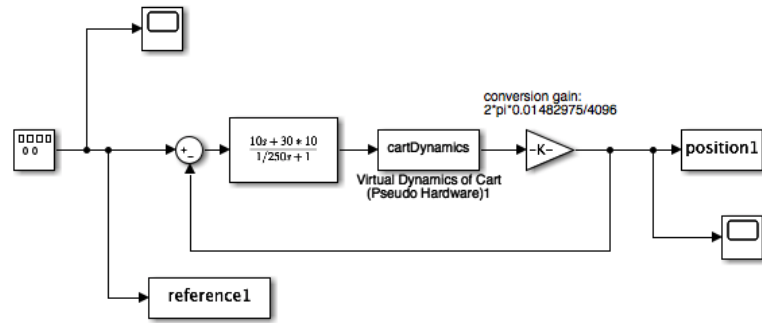


Figure 34: Simulink block diagram of the PD Control with Tuned Controller Values and Saw Wave Input

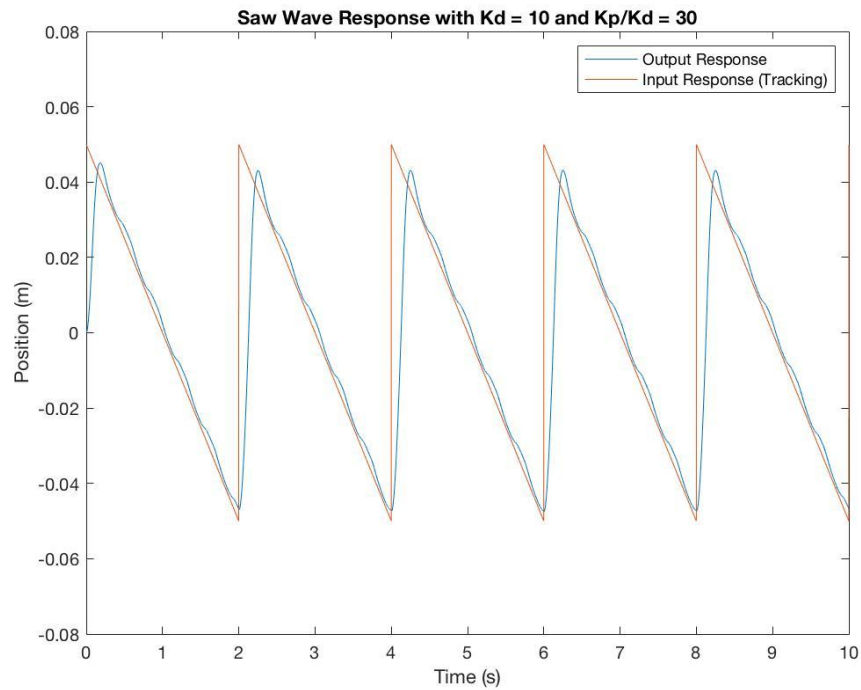


Figure 35: Plot of the Saw Wave Response

- Exponentially-decaying sine wave (initial amplitude ≤ 15 cm, pick a reasonable time constant and frequency so that the decay is evident, but not too sudden or too gradual)

Parameters

Sine type: Time based

Time (t): Use simulation time

Amplitude: 0.1

Bias: 0

Frequency (rad/sec): pi

Phase (rad): 0

Sample time: 0

Figure 36: Exponentially-decaying Saw Wave Setting

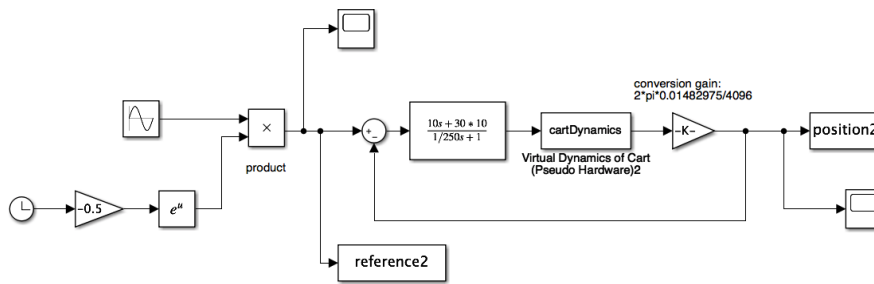


Figure 37: Simulink diagram of PD Control with Tuned Controller Values and Decaying Sine Wave Input

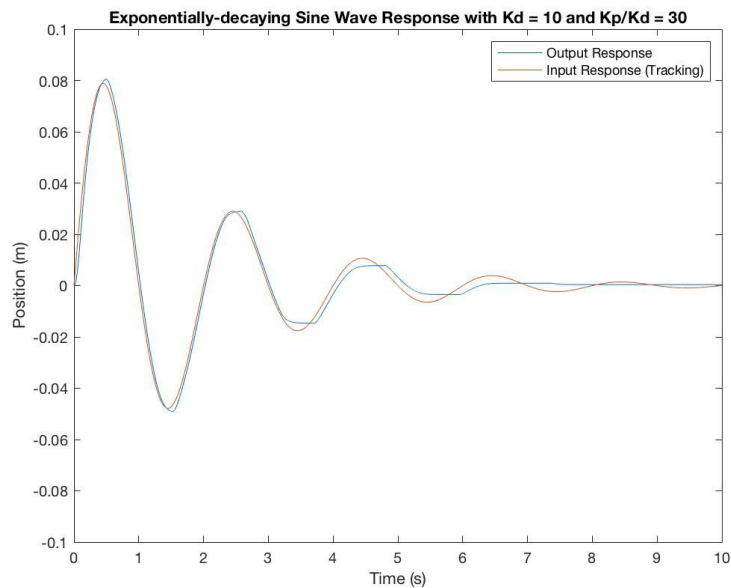


Figure 38: Plot of the Exponentially-decaying sine wave Response

Discuss the performance of the controller for the different input signals. Which signals can the controller track easily? Why is this the case?

The controller can track the two sine wave signals which are sine wave and exponentially-decaying sine wave easily, and this is because sine wave signals are smoother than the pulse signal and the triangle wave signal. As observed in the plots above, the reference responses and the output responses do not match well at the peaks because there are sudden changes in magnitude. The pulse signal and the triangle wave signal have sudden changes in magnitude all over the place causing the controller to fail to track the signals well. Also, the controller seems to track the sine wave signal a little better than the exponentially-decaying sine wave signal because the sine wave signal is smoother and more continuous in the sense that retains its sinusoidal shape while the exponentially-decaying sine wave decays over time until it reaches steady state making it less smooth and therefore harder to track. Theoretically speaking, faster signals with longer rise time and higher frequencies or more oscillations are also harder to track.

4. Performance for Limited Braking Ability

Suppose you are limited in your ability to slow the cart. How does limiting the negative voltage threshold to $-3V$ affect the step response. What properties (rise time, maximum peak, settling time, etc.) are affected? Include plots of the voltage input and the position of the cart.

Observations from the stepinfo results below, generally speaking, both the rise time, overshoot and undershoot (if applicable) are affected with saturation. There are sometimes small changes in peak, peak time, settling time, settlingMin and SettlingMax but insignificant which could be ignored.

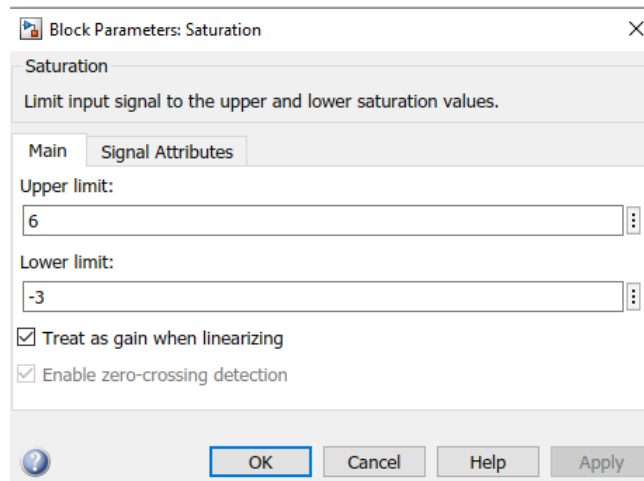


Figure 39: Configuration of the Saturation Block Parameters

- Pulse generator (50% pulse width, amplitude ≤ 7.5 cm, period ≥ 3 sec)

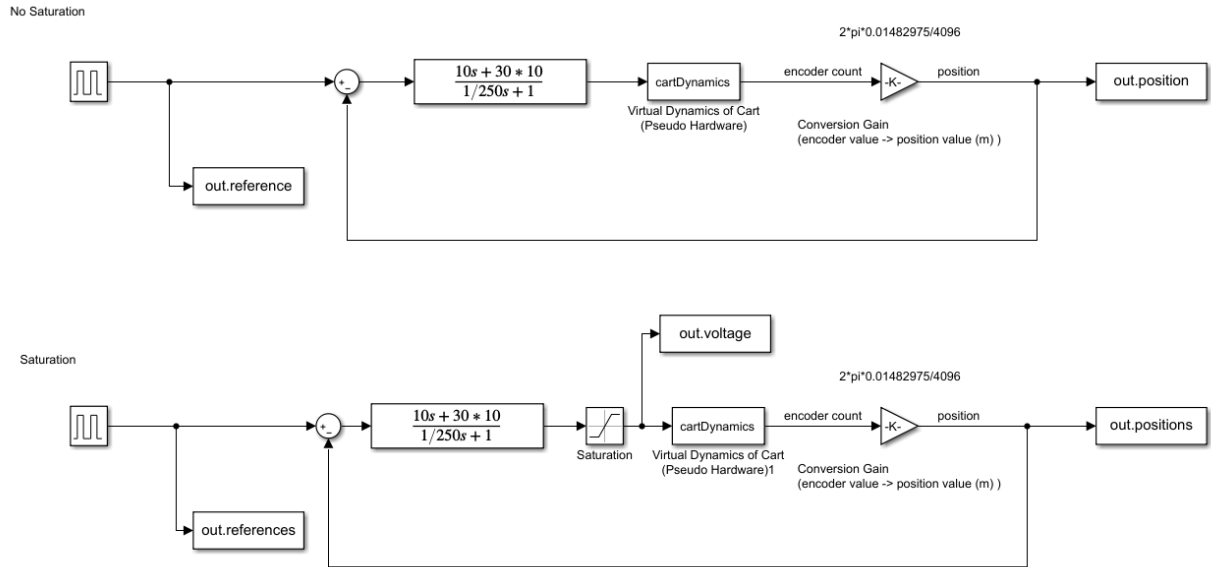


Figure 40: Simulink Block Diagram of the Closed Loop System with PD Controller for a pulse generator input w/o saturation block (top) and w/ saturation block (bottom).

```

%% 5.4 Pulse Generator Saturation (50% pulse width, amplitude ≤ 7.5 cm, period ≥ 3 sec)
clear,clc,close all;
%Parameters
tstop = 10;

%Simulate
out = sim('cartdynamicshdp',tstop);
reference = out.reference.data;
position1 = out.position.data;
position2 = out.positions.data;
voltage = out.voltage.data;
time = out.tout;

%Plot
plot(time,position1)
hold on
plot(time,position2)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 -0.1 0.15])
title('Response of Cart System for Pulse Generator w/ Saturation') %Title
legend('Original Response w/ Pulse Generator','Response w/ Saturation Block','Reference')
figure
plot(time,voltage)
xlabel('time(s)'),ylabel('voltage(V)') %Label Axes
title('Voltage Input')
axis([0 10 -5 10])
legend('Voltage Input')

%Step Response Info
stepinfofonosat = stepinfo(position1,time)
stepinfofowithsat = stepinfo(position2,time)

```

Figure 41: Matlab code to obtain plots of system response and step info.

stepinfofonosat =	stepinfofowithsat =
<u>struct</u> with fields:	<u>struct</u> with fields:
RiseTime: 0.1289	RiseTime: 0.1291
SettlingTime: 9.2848	SettlingTime: 9.2856
SettlingMin: -0.0041	SettlingMin: -0.0032
SettlingMax: 0.0791	SettlingMax: 0.0792
Overshoot: 3.0204	Overshoot: 2.9586
Undershoot: 5.3894	Undershoot: 4.1716
Peak: 0.0791	Peak: 0.0792
PeakTime: 3.2450	PeakTime: 3.2470

Figure 42: Step info for original response (left) and response with saturation block (right).

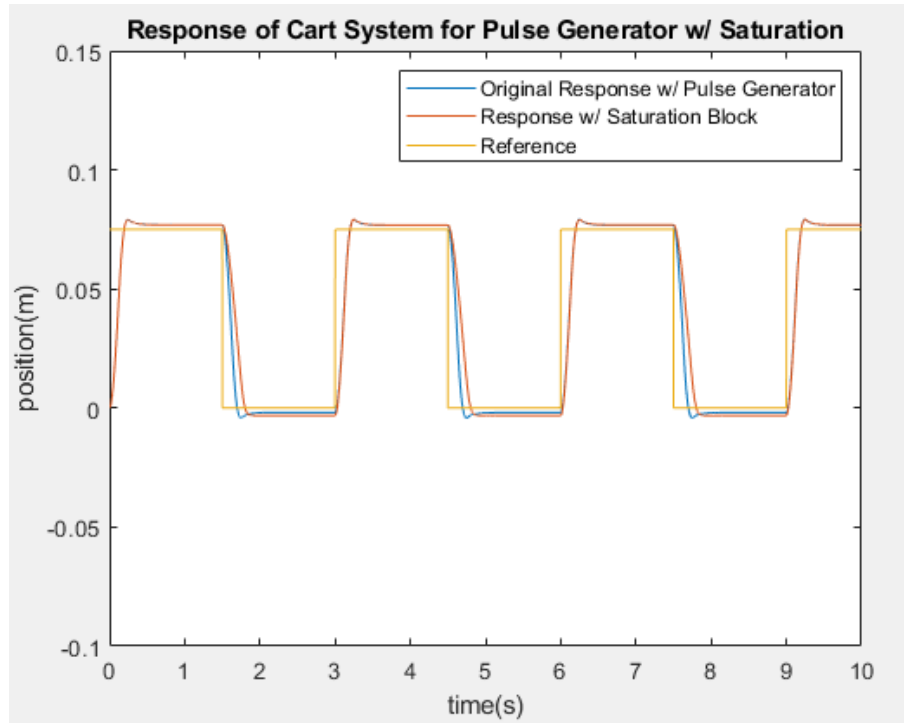


Figure 43: Response of the cart system for a pulse generator input including the response of the system, response of the system with a saturation block, and reference (tracking).

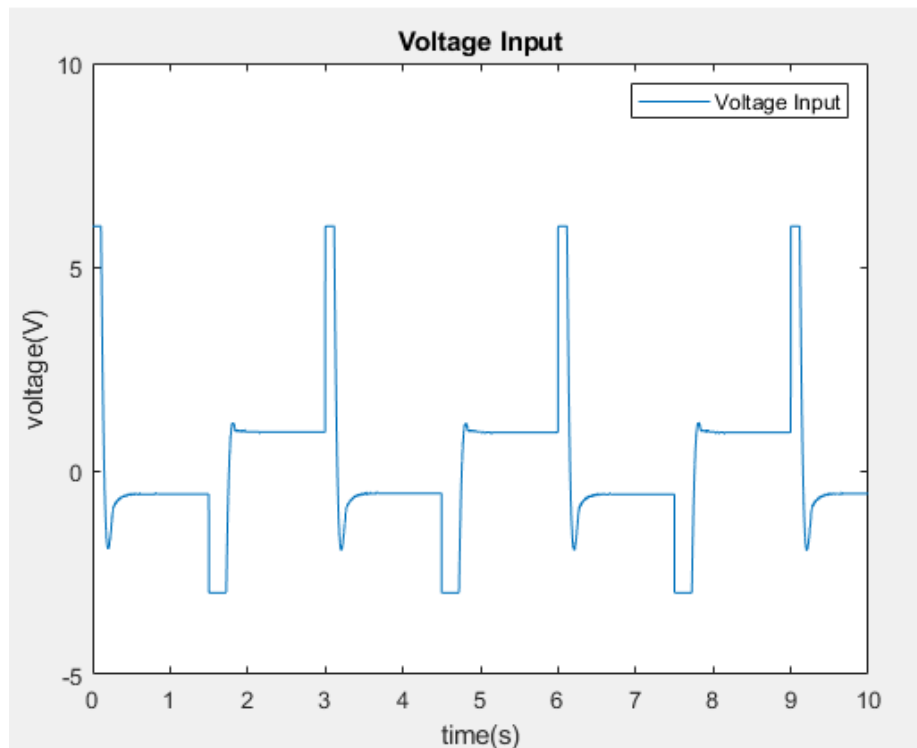


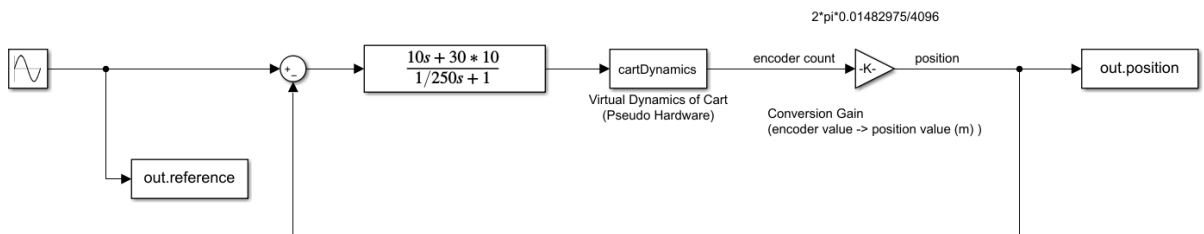
Figure 44: Plot of Input Voltage over Time.

Properties Affected:

- Rise Time
- Settling Time
- SettlingMin
- SettlingMax
- Overshoot
- Undershoot
- Peak
- PeakTime

- Sine wave (amplitude ≤ 10 cm, frequency ≤ 1 Hz)

No Saturation



Saturation

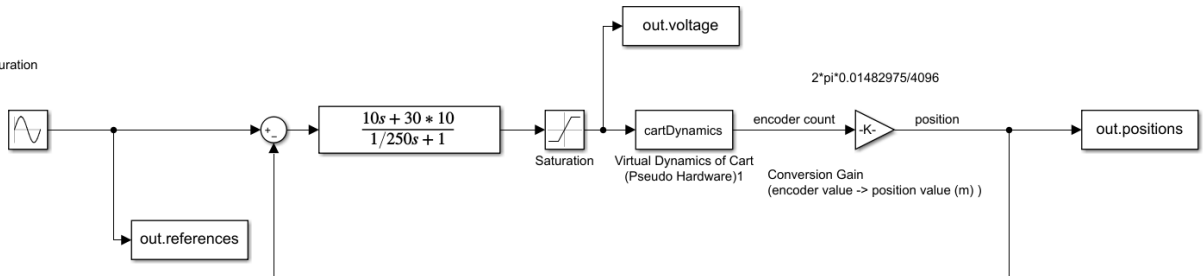


Figure 45: Simulink Block Diagram of the closed loop system with PD Controller for a sine wave input w/o saturation block (top) and w/ saturation block (bottom).


```

%% 5.4 Sine Wave Saturation (amplitude  $\leq 10$  cm, frequency  $\leq 1$  Hz)
clear,clc,close all;
%Parameters
tstop = 10;

%Simulate
out = sim('cartdynamicshds',tstop);
reference = out.reference.data;
position1 = out.position.data;
position2 = out.positions.data;
voltage = out.voltage.data;
time = out.tout;

%Plot
plot(time,position1)
hold on
plot(time,position2)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 -0.18 0.18])
title('Response of Cart System for Sine Wave w/ Saturation') %Title
legend('Original Response w/ Sine Wave','Response w/ Saturation Block','Reference')
figure
plot(time,voltage)
xlabel('time(s)'),ylabel('voltage(V)') %Label Axes
title('Voltage Input')
axis([0 10 -5 10])
legend('Voltage Input')

%Step Response Info
stepinfofonosat = stepinfo(position1,time)
stepinfofowithsat = stepinfo(position2,time)

```

Figure 46: Matlab code to obtain plots of system response and step info.

stepinfofonosat =	stepinfofowithsat =
<u>struct</u> with fields:	<u>struct</u> with fields:
RiseTime: 0.0232	RiseTime: 0.0603
SettlingTime: 9.9961	SettlingTime: 9.9956
SettlingMin: -0.1035	SettlingMin: -0.0800
SettlingMax: 0.1035	SettlingMax: 0.1040
Overshoot: 452.0631	Overshoot: 183.7097
Undershoot: 557.4029	Undershoot: 370.4032
Peak: 0.1045	Peak: 0.1045
PeakTime: 0.2800	PeakTime: 0.2800

Figure 47: Step info for original response (left) and response with saturation block (right).

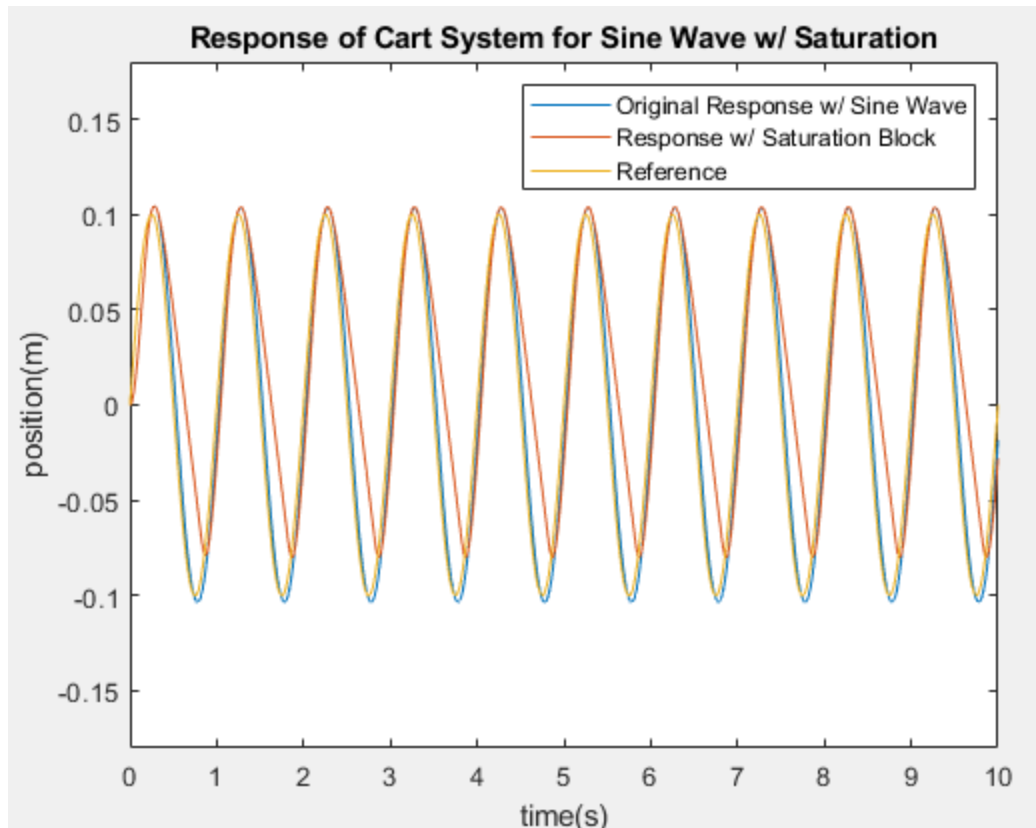


Figure 48: Response of the cart system for a sine wave input including the response of the system, response of the system with a saturation block, and reference (tracking).

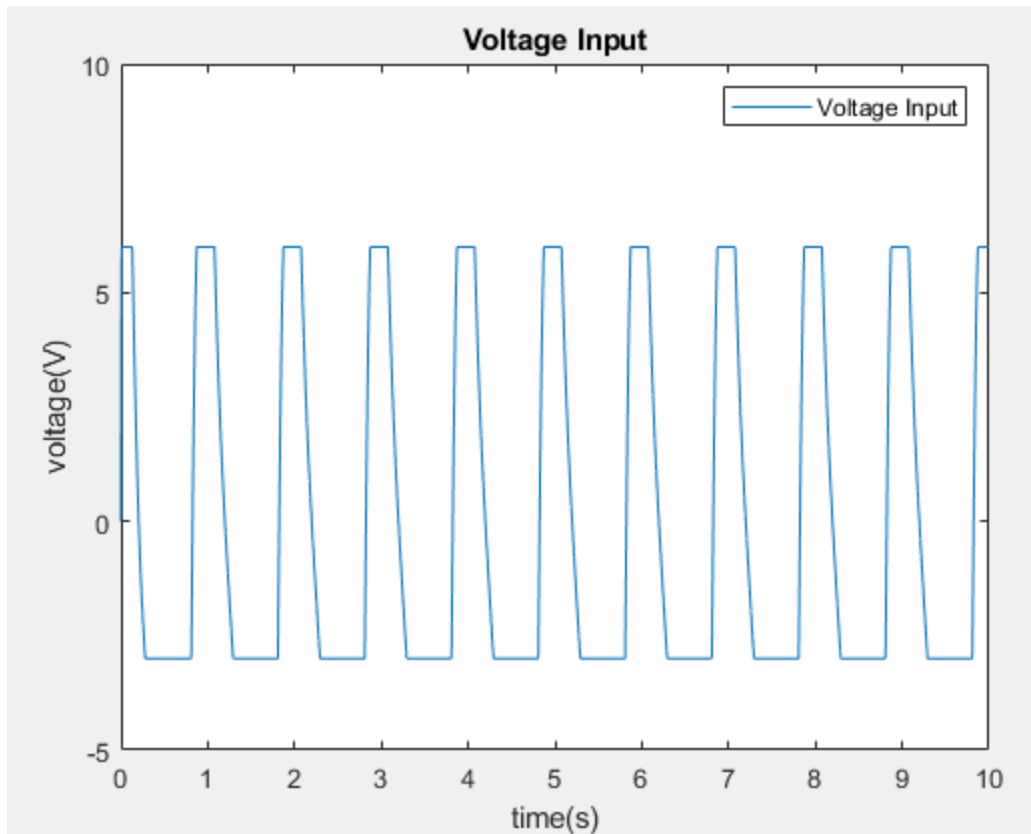


Figure 49: Plot of Input Voltage over Time.

Properties Affected:

- **Rise Time**
- **Settling Time**
- **SettlingMin**
- **SettlingMax**
- **Overshoot**
- **Undershoot**

- Saw/Triangle wave (amplitude ≤ 5 cm, frequency ≤ 0.5 Hz)

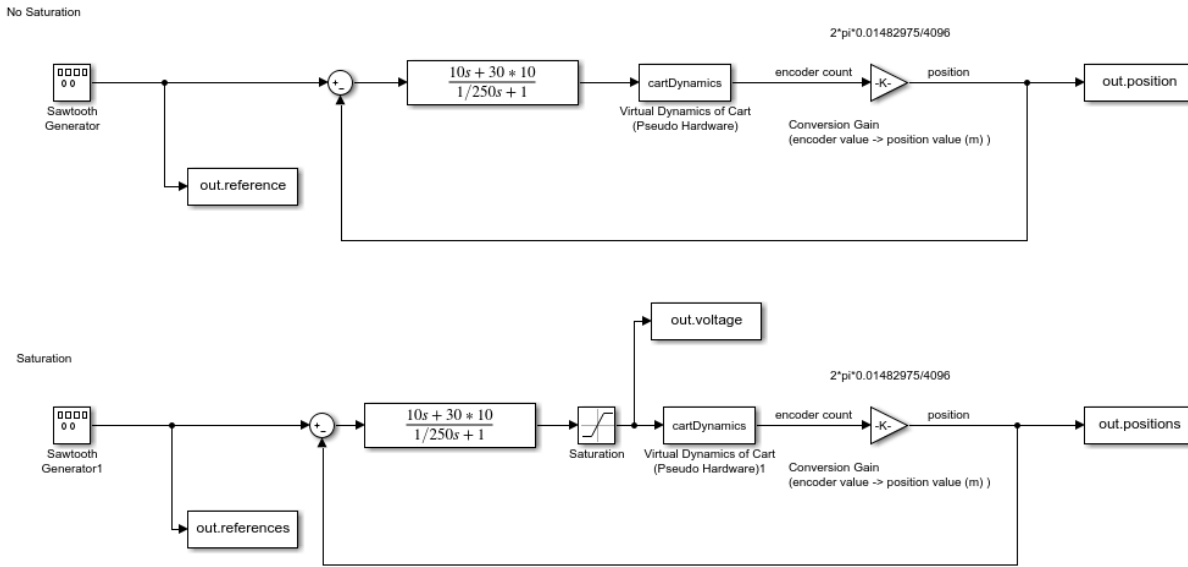


Figure 50: Simulink Block Diagram of the Closed Loop System with PD Controller for a saw/triangle wave w/o saturation block (top) and w/ saturation block (bottom).

```
%% 5.4 Saw/Triangle Wave Saturation (amplitude  $\leq 5$  cm, frequency  $\leq 0.5$  Hz)
clear,clc,close all;
%Parameters
tstop = 10;

%Simulate
out = sim('cartdynamicshdst',tstop);
reference = out.reference.data;
position1 = out.position.data;
position2 = out.positions.data;
voltage = out.voltage.data;
time = out.tout;

%Plot
plot(time,position1)
hold on
plot(time,position2)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 -0.1 0.15])
title('Response of Cart System for Sawtooth Generator w/ Saturation') %Title
legend('Original Response w/ Sawtooth Generator','Response w/ Saturation Block','Reference')
figure
plot(time,voltage)
xlabel('time(s)'),ylabel('voltage(V)') %Label Axes
title('Voltage Input')
legend('Voltage Input')

%Step Response Info
stepinfo nosat = stepinfo(position1,time)
stepinfo withsat = stepinfo(position2,time)
```

Figure 51: Matlab code to obtain plots of system response and step info.

<code>stepinfofonosat =</code>	<code>stepinfofowithsat =</code>
<u>struct</u> with fields:	<u>struct</u> with fields:
RiseTime: 0.7519	RiseTime: 0.7526
SettlingTime: 9.9375	SettlingTime: 9.9375
SettlingMin: -0.0480	SettlingMin: -0.0477
SettlingMax: 0.0431	SettlingMax: 0.0431
Overshoot: 2.1813	Overshoot: 1.6957
Undershoot: 96.2676	Undershoot: 96.2209
Peak: 0.0480	Peak: 0.0477
PeakTime: 8.0020	PeakTime: 4.0010

Figure 52: Step info for original response (left) and response with saturation block (right).

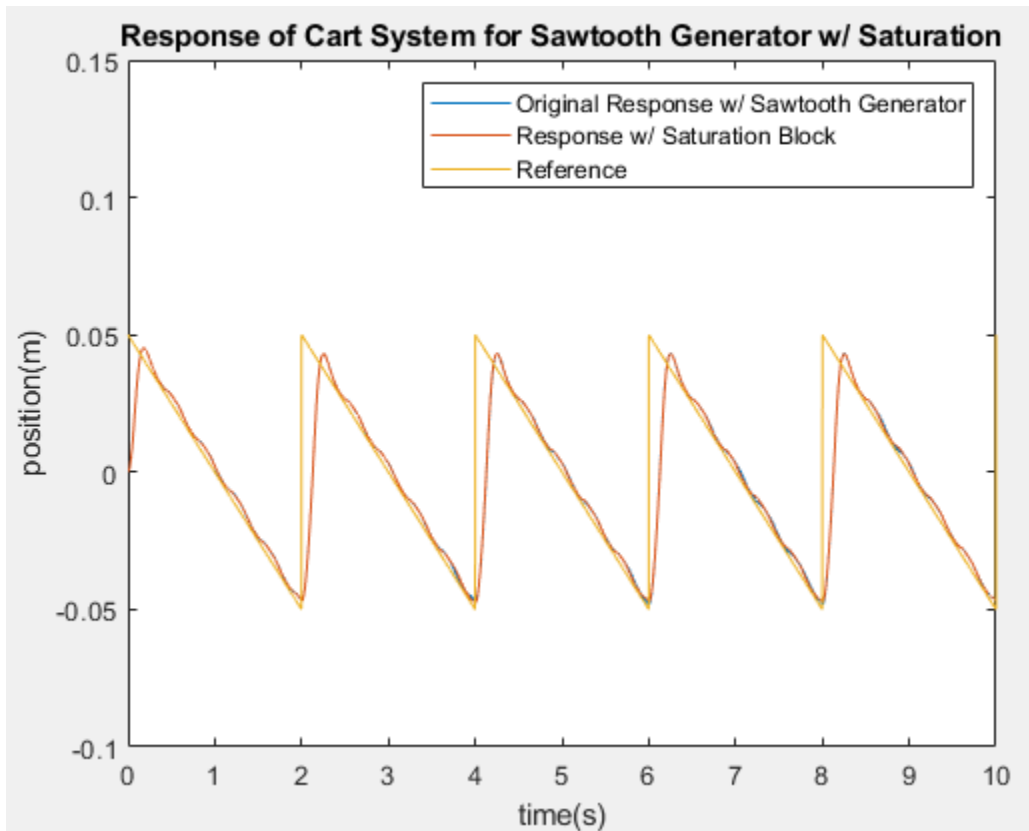


Figure 53: Response of the cart system for a saw/triangle wave input including the response of the system, response of the system with a saturation block, and reference (tracking).

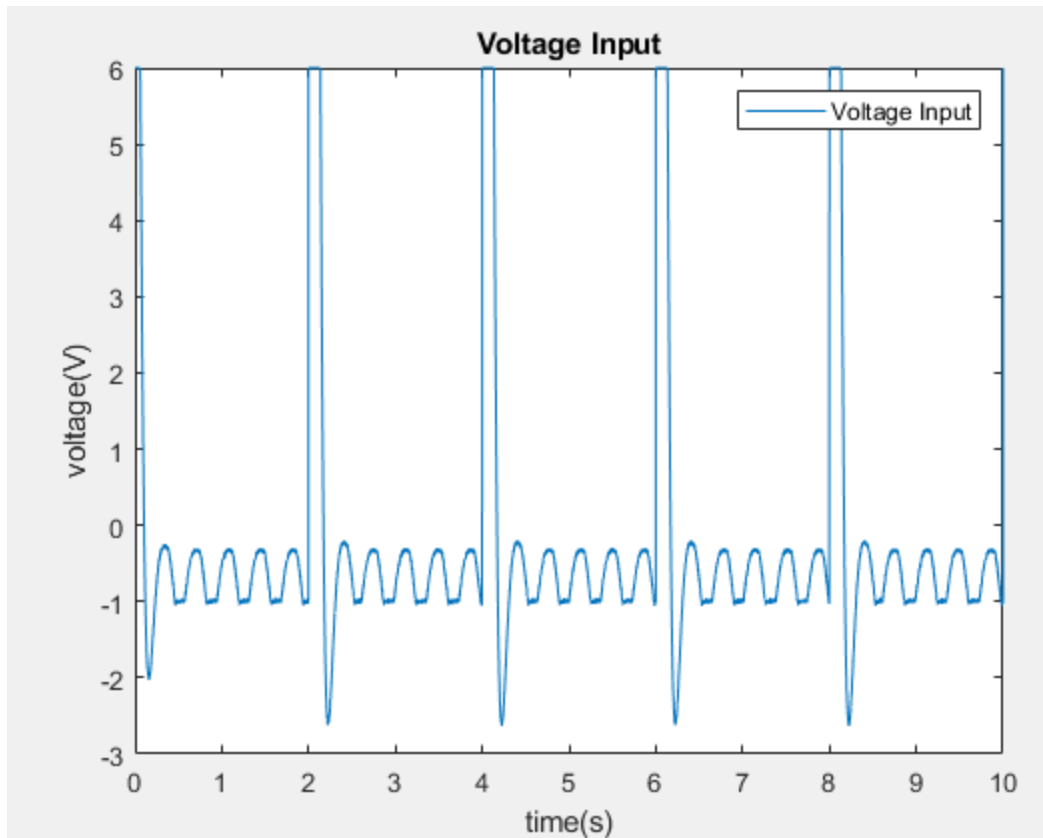


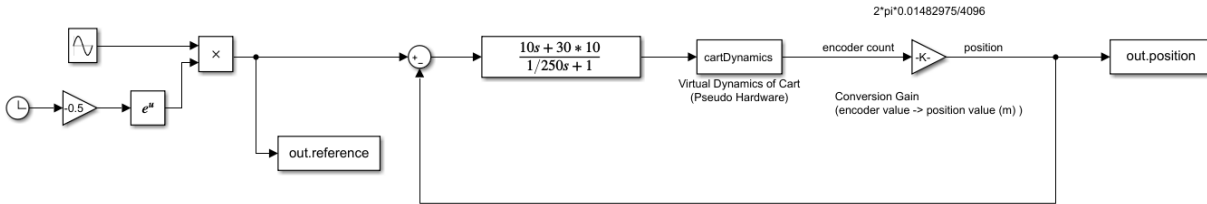
Figure 54: Plot of Input Voltage over Time.

Properties Affected:

- **Rise Time**
- **Settling Min**
- **Overshoot**
- **Undershoot**
- **Peak**
- **PeakTime**

- Exponentially-decaying sine wave (initial amplitude ≤ 15 cm, pick a reasonable time constant and frequency so that the decay is evident, but not too sudden or too gradual)

No Saturation



Saturation

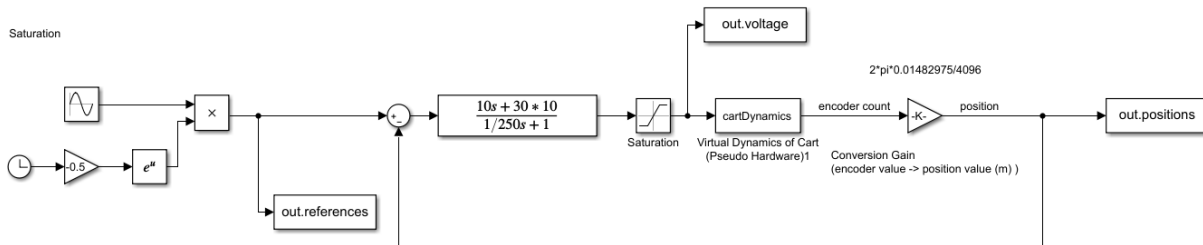


Figure 55: Simulink Block Diagram of the Closed Loop System with PD Controller for an exponentially-decaying sine wave w/o saturation block (top) and w/ saturation block (bottom).

```

%% 5.4 Exponentially-decaying sine wave
%Exponentially-decaying sine wave (initial amplitude ≤ 15 cm, pick a reasonable time constant and
%frequency so that the decay is evident, but not too sudden or too gradual)
clear,clc,close all;
%Parameters
tstop = 10;

%Simulate
out = sim('cartdynamicshdedes',tstop);
reference = out.reference.data;
position1 = out.position.data;
position2 = out.positions.data;
voltage = out.voltage.data;
time = out.tout;

%Plot
plot(time,position1)
hold on
plot(time,position2)
hold on
plot(time,reference)
xlabel('time(s)'),ylabel('position(m)') %Label Axes
axis([0 10 -0.1 0.15])
title('Response of Cart System for Exponenetially-Decaying Sine Wave w/ Saturation') %Title
legend('Original Response w/ Exponenetially-Decaying Sine Wave','Response w/ Saturation Block','Reference')
figure
plot(time,voltage)
xlabel('time(s)'),ylabel('voltage(V)') %Label Axes
title('Voltage Input')
axis([0 10 -5 5])
legend('Voltage Input')

%Step Response Info
stepinfo nosat = stepinfo(position1,time)
stepinfo withsat = stepinfo(position2,time)

```

Figure 56: Matlab code to obtain plots of system response and step info.

<code>stepinfo nosat =</code>	<code>stepinfo withsat =</code>
<u>struct</u> with fields:	<u>struct</u> with fields:
RiseTime: 0.0126	RiseTime: 0.0088
SettlingTime: 6.1950	SettlingTime: 6.1546
SettlingMin: -0.0491	SettlingMin: -0.0491
SettlingMax: 0.0805	SettlingMax: 0.0805
Overshoot: 8.5341e+03	Overshoot: 1.5991e+04
Undershoot: 5.2659e+03	Undershoot: 9.8182e+03
Peak: 0.0805	Peak: 0.0805
PeakTime: 0.4930	PeakTime: 0.4920

Figure 57: Step info for original response (left) and response with saturation block (right).

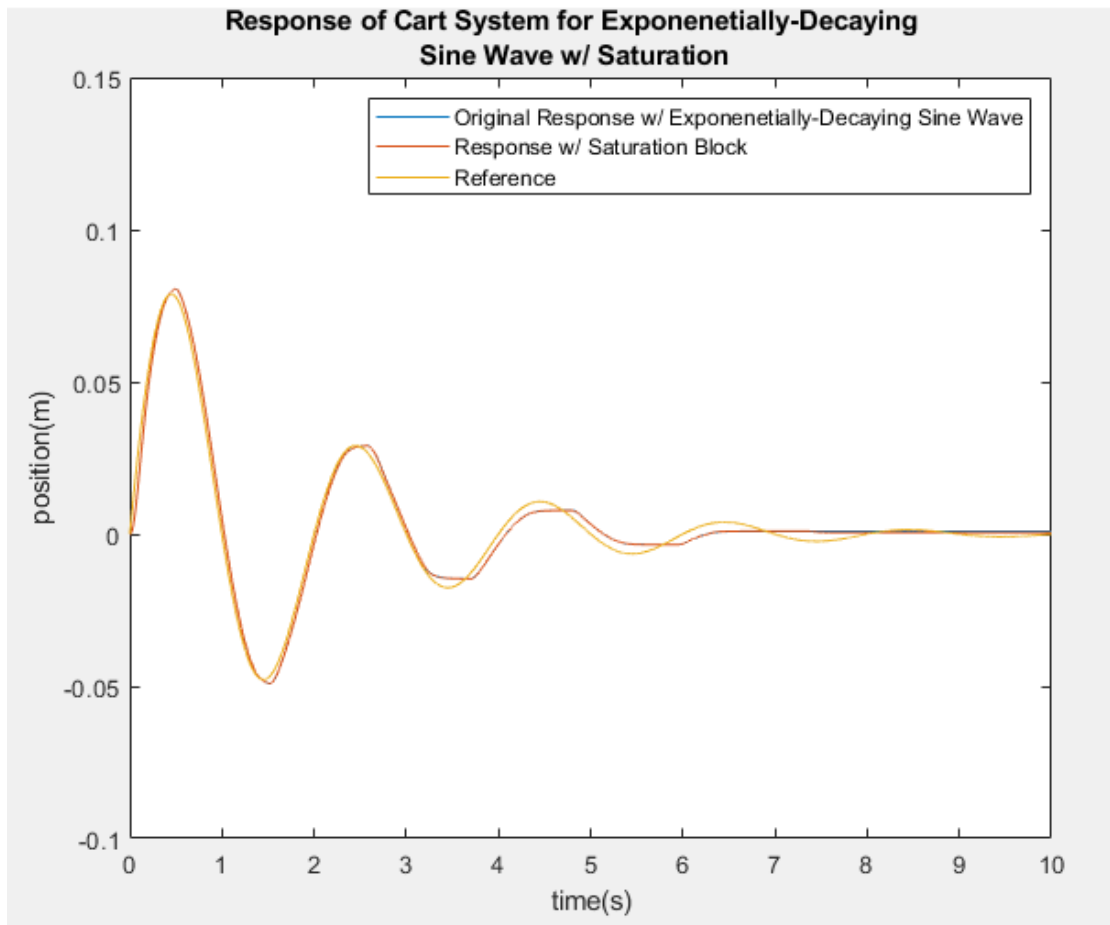


Figure 58: Response of the cart system for an exponentially-decaying sine wave input including the response of the system, response of the system with a saturation block, and reference (tracking).

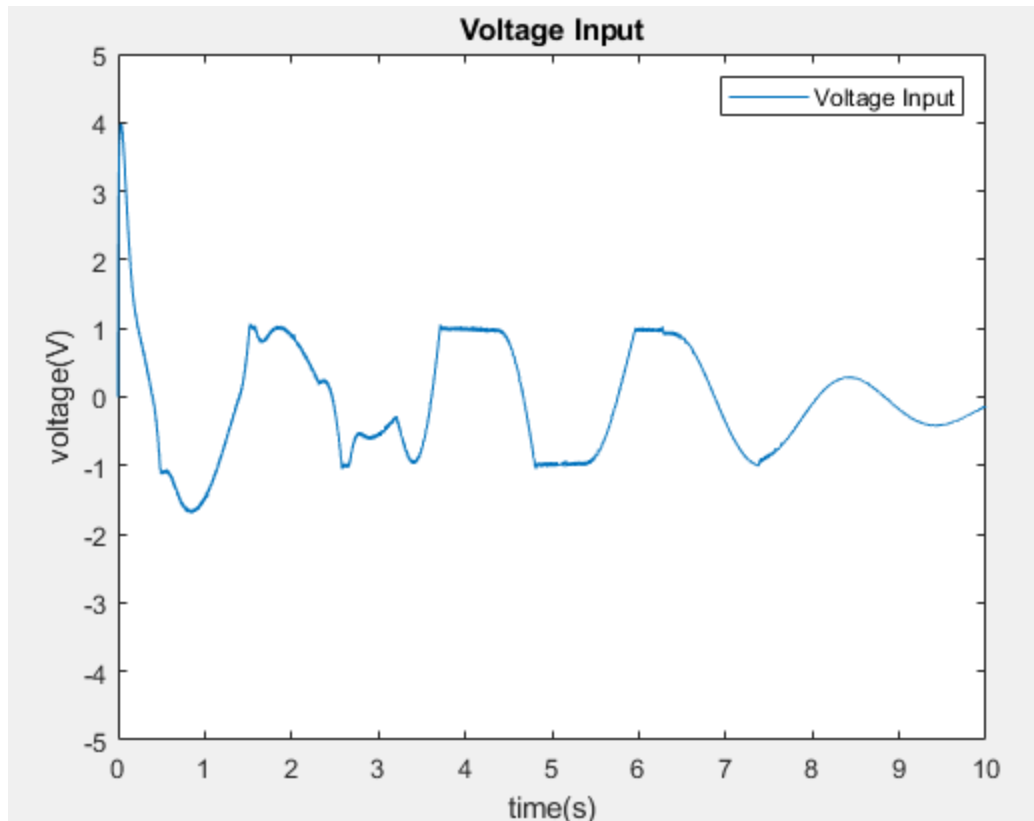


Figure 59: Plot of Input Voltage over Time

Properties Affected:

- **Rise Time**
- **Settling Time**
- **Overshoot**
- **PeakTime**

How does this affect the performance when tracking the sine wave input?

From Figure 48, it can be observed that as the response with the saturation block goes to track the negative peaks of the reference, it stops before it reaches the negative peaks and goes back to track to the positive peaks. From this we can infer that the saturation block, which limits the negative voltage threshold to -3V and our positive voltage to 6V, causes this behavior which makes it inaccurate in tracking negative peak (the minimums) while still being relatively good in tracking the positive peaks (the maximums). This is evident in the input voltage plot which corresponds to the response plot, where the voltage plot has a max of 6V and a min of -3V. This causes the response plot with saturation to stop before it can completely track the negative peak.