



OHIO UNIVERSITY

Russ College of Engineering & Technology
Department of Mechanical Engineering

Date: April 16th, 2020
To: Dr. Bob Williams
From: Anna Kelley, Amanda Przybocki, Jonathan Bowman
Subject: ME 3012 Final Report

Dr. Bob,

This memo is to present our final report for the ME 3012 term project. For this project we choose a real-world mechanism with a single-input, single-output (SISO) linear system. The mechanism we chose is a robot elbow control. For this report you asked us to describe the systems functionality, model the system, design a controller to meet desired specifications, complete a open-loop and closed-loop analysis of the system and summarize the input effort and disturbance response of each controller.

This project uses all the knowledge learned over the course of this semester and uses those outcomes to find the best controller design for our project. While this paper does include a lot of discussion, a great deal of supplemental work is provided. This includes use of MatLab software as well has derivations. The appendices serve as a place to organize this supplemental data.

If you have any question, please feel free to contact us.

Regards,

Anna Kelley, Amanda Przybocki, Jonathan Bowman
Ak256116@ohio.edu, Ap852115@ohio.edu, Jb146214@ohio.edu

ME 3012 Term Project: Final Report

Flex-Arm Robot Elbow Control



Figure 0.1: Robot Elbow Control [3]

Introduction:

A robotic elbow control consists of a lightweight flexible arm that connects to a wrist and an elbow joint. This allows for smooth and controlled movement of the arm. First off, we modeled the system below, showing the diagram as well as the transfer function, followed by our assumptions and discussion. After we modeled the open-loop transfer function using MATLAB and Simulink to accurately portray our results.

Functional Diagram:



Figure 0.2: Functional Diagram of the Robotic Arm Function [2]

Abstract:

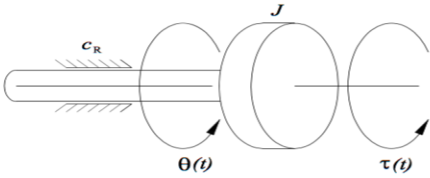
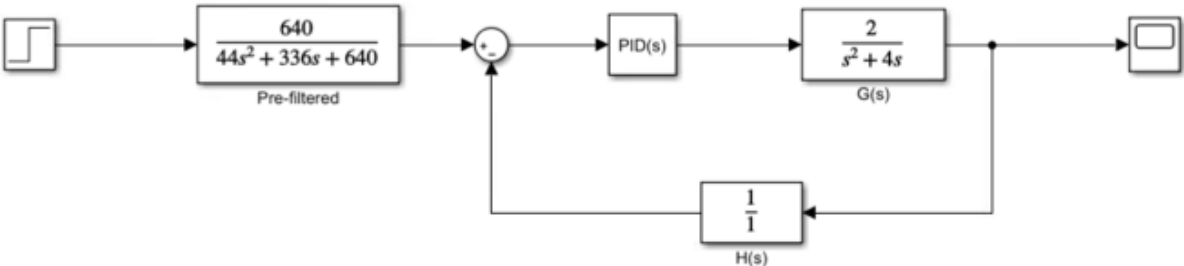
Diagram		Model
		$J\ddot{\theta}(t) + c_R\dot{\theta}(t) = \tau(t)$
Open Loop Transfer Function $G(s)$		
$\frac{\Theta(s)}{T(s)} = \frac{1}{s(Js + c_R)}$	$G(s) = \frac{\Theta(s)}{T(s)} = \frac{2}{s(s+4)}$	$\tau(t)$ torque input $\theta(t)$ output elbow angle
Closed Loop Transfer Function $T(s)$		
$\Delta_{DES}(s) = s^3 + 48s^2 + 336s + 640$		Damping Condition: Critically Damped
Best Controller Type: PID Controller		$H(s) = 1$ (Perfect Sensor)
$G_c(s) = \frac{22s^2 + 168s + 320}{s}$	$G_P(s) = \frac{640}{44s^2 + 336s + 640}$	$T(s) = \frac{44s^2 + 336s + 640}{s^3 + 48s^2 + 336s + 640}$
Closed Loop Block Diagram:		
		

Table of Contents:

Introduction:.....	2
Abstract:.....	3
Table of Contents:.....	4
Free Body Diagrams (FBD).....	5
Ordinary Differential Equation (ODE)	6
Laplace Transforms	6
<i>Open Loop Transfer Function</i>	7
Block Diagram.....	7
Open-Loop Discussion:	7
Open Loop Transfer Function	8
Performance Specs and Effects in the Real-World Response	9
<i>Closed Loop Transfer Function</i>	10
Block Diagram.....	10
Closed-Loop Discussion:.....	10
Desired Performance Specifications:.....	11
Controller (G_c) Methods	14
Lead Controller:	14
Proportional Controller:	17
PID Controller:.....	18
Closed Loop Input Effort.....	21
Lead Controller:	21
Proportional Controller:	22
PID Controller:.....	23
Closed Loop Disturbances.....	24
Lead Controller:	24
Proportional Controller:	25
PID Controller:.....	26
Final Controller Decision	27
References.....	28
Appendix A: MATLAB Code	29
Appendix B: MATLAB Plots	35
Appendix C: Simulink Results.....	38
Appendix D: Lead Controller Calculations	44

Free Body Diagrams (FBD)

The FBD can be seen in Figure 1. This diagram highlights the input of torque and the output of θ . Fortunately, we can model this more simply as a first-order massless rotational mechanical system and this is shown in Figure 2.

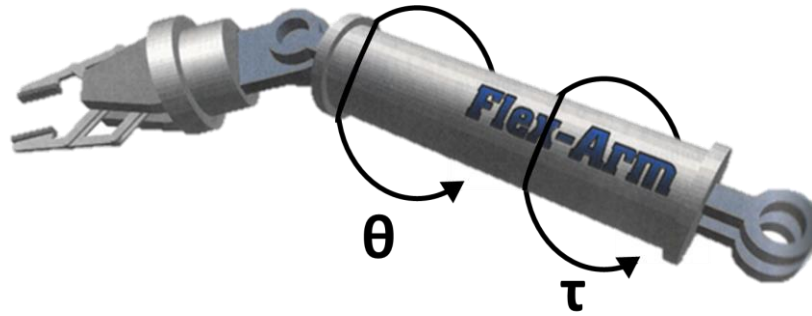


Figure 1: Realistic Arm with FBD Overlay

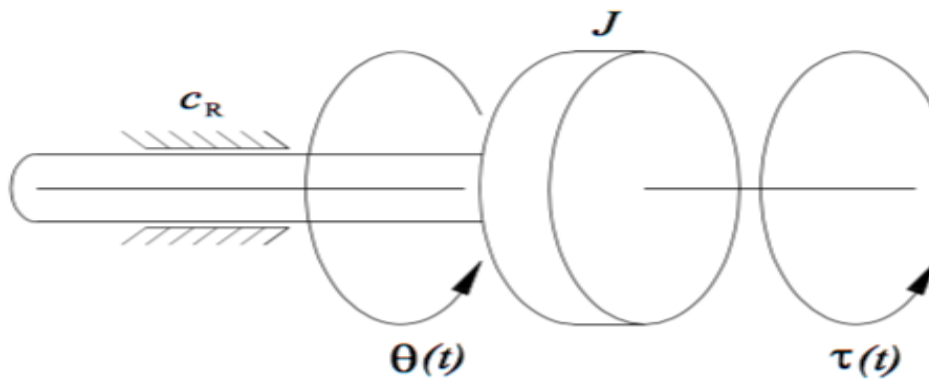


Figure 2: Simplified Free Body Diagram & System Model

Ordinary Differential Equation (ODE)

The ODE was provided in Dr. Bob's Notes Book [1] and is for a springless rotational mechanical system. The equation is reproduced here:

$$J\ddot{\theta}(t) + c_R\dot{\theta}(t) = \tau(t)$$

The input to this system will be torque provided from a motor and the output will be the angle of rotation. See Table # for the complete list of parameter values.

Table 1: Parameter Values for ODE

Parameter	Symbol (Units)	Value
Damping Coefficient	c_R (N-sec/m)	4
Mass Moment of Inertia	J (kg-m ²)	1
Torque	τ (N-m)	2

Laplace Transforms

In order to derive our Transfer Function G_s , we need to convert our ODE to the Laplace Domain. First, we begin with this system's ODE:

$$J\ddot{\theta}(t) + c_R\dot{\theta}(t) = \tau(t)$$

Next, we take the Laplace of Both Sides with zero initial conditions and this results in:

$$J[s^2\Theta(s) + 0 + 0] + c_R[s\Theta(s) + 0] = T(s)$$

Then we simplify this equation:

$$\Theta(s)[Js^2 + c_Rs] = T(s)$$

We can then solve for the transfer function, G_s :

$$G_s = \frac{\Theta(s)}{T(s)} = \frac{2}{s(s+4)}$$
$$G_s = \frac{\Theta(s)}{T(s)} = \frac{2}{s(s+4)}$$

Open Loop Transfer Function

Block Diagram

The open loop block diagram is produced below in Figure 3. This shows both the input and output as well as the plant. The plant is our groups derived transfer function.

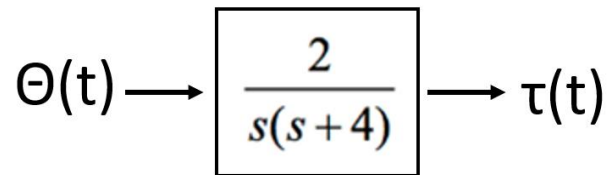


Figure 3: Open loop block diagram

Open-Loop Discussion:

1. *2nd Order ODE* – the highest power in the denominator is “2”
2. Characteristic polynomial (CP) – $(s^2 + 4s)$
3. The roots of the “CP” – two open loop poles – $s_1 = 0, s_2 = -4$
4. Type I system – one “s” can be factored in the G(s) denominator
5. *Marginally stable* – due to the zero pole (which is an integrator)
6. Dampening Condition – *special undampened case* (every plot except impulse)
7. Impulse – *critically dampened*
8. *No open loop zeros* – the G(s) numerator (2) – never go to zero

Open Loop Transfer Function

The responses for the unit step, unit impulse and unit ramp for our system were plotted both in MATLAB and in the extension Simulink. The fourth response was only plotted in Simulink and it would be hard to model in MATLAB. The following plot comparisons will feature the same convention of the left plot being the output of the MATLAB command (as seen in Appendix A) and the right will be the output plots of the functions in Simulink (shown in Appendix C). Overall, the results for the two approaches agree and provide credibility for the results presented.

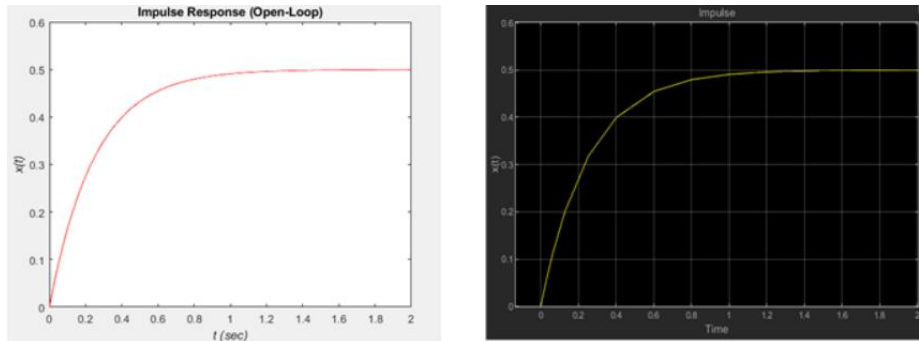


Figure 4: Impulse Response Plots from Simulink & MATLAB

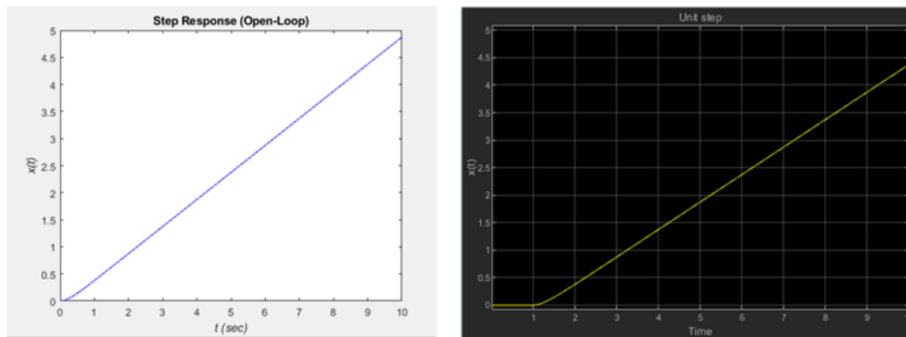


Figure 5: Step Response Plots from Simulink & MATLAB

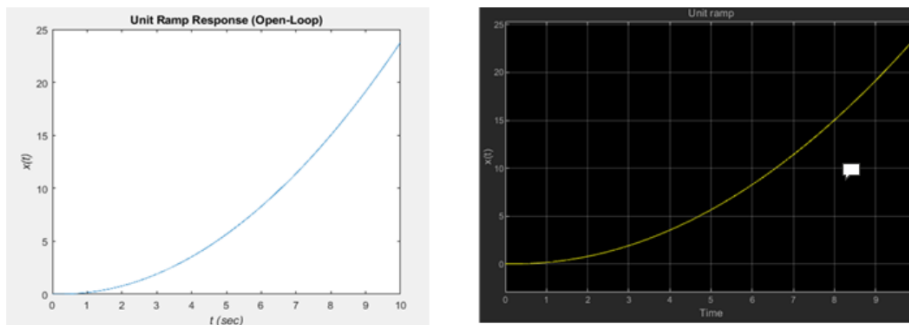


Figure 6: Step Response Plots from Simulink & MATLAB

Performance Specs and Effects in the Real-World Response

The most ideal function for a robotic arm would be one that is stable, has a small Percent Overshoot (PO), has a rise time that is both not too fast and not too slow and reaches a settling time quickly. The open loop transfer function that with a step input is shown in Figure 7. This response has infinite performance specifications as the step never approaches a steady state value.

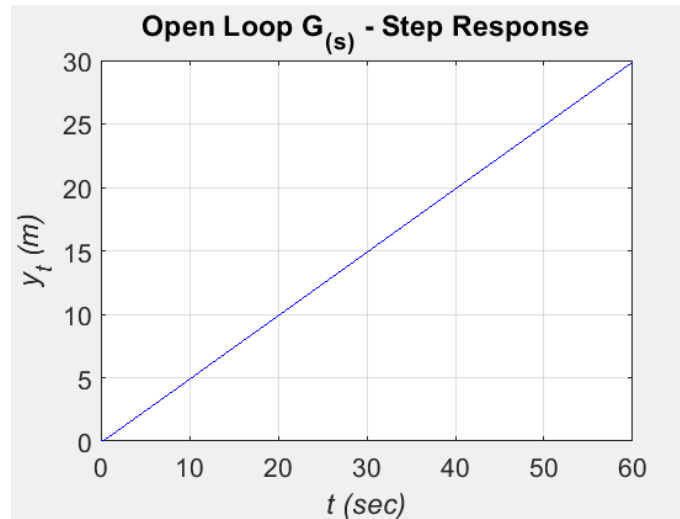


Figure 7: Annotated Step Response

If left using the open loop transfer function such as stated above the robotic elbow would never stop turning, thus making the system effectively useless. It is likely too, though not covered in detail here, that this would wear on the robotic arm components unfavorably and require a lot of repairing. It is apparent that a Controller is needed for this arm. The open loop specifications can be found summarized in Table 2.

Table 2: Open Loop Performance Specifications

Performance Specification	Value
Percent Overshoot, PO	N/A
Settling Time, t_s	∞
Rise Time, t_r	∞
Peak Time, t_p	∞

Closed Loop Transfer Function

Block Diagram

The closed loop block diagram is produced below in Figure 8. This diagram shows the generic closed loop transfer function. Note that this will change in appearance as an attenuation factor or a prefilter is needed, and these will be shown. We will assume that we have a perfect sensor $H(s)$ so this will be 1 and will be eliminated easily in the $T(s)$ solutions.

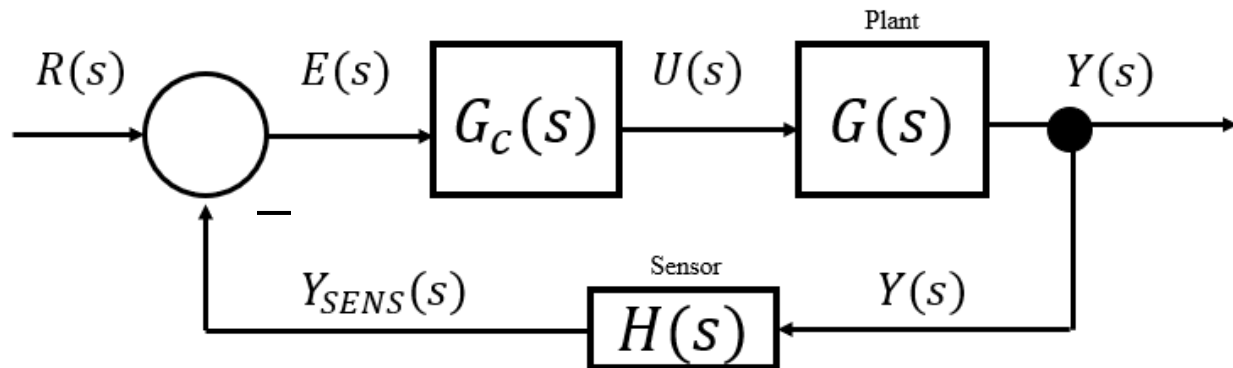


Figure 8: Generic Closed Loop Transfer Function

Where:

$R(s) = \theta = \text{Desired Angle}$

$E(s) = \text{Error}$

$U(s) = \tau = \text{Input Torque}$

$Y(s) = \theta = \text{Output Angle}$

$Y_{\text{sens}}(s) = \theta_{\text{sensed}} = \text{Sensed Output Angle}$

$G_c(s) = \text{Transfer Function}$

$G(s) = \text{Open Loop Transfer Function}$

$H(s) = \text{Sensor}$

Closed-Loop Discussion:

- 1. Designing for a Critically Damped System*
- 2. Assuming a Perfect Sensor, $H(s)$*
- 3. See next section for a continuation of discussion.*

Desired Performance Specifications:

In order to gauge the effectiveness of our controllers design we will need to first describe the desired overall transfer functions output. For this project we chose to design a stable, critically damped system. Additionally, a rise time of 0.75 seconds and a settling time of 1 second was specified. These values can be seen summarized in Table 3.

Using a critically damped system we can eliminate the percent overshoot of the response. If we had a large overshoot, then the robotic arm would use more torque than required and move right past the desired elbow angle. This could jeopardize the arms mechanical system or the effectiveness of the arms usage. Therefore, having no overshoot works well with our system. A critically damped response will have poles that are both real and the same, that is to say $s_1 = s_2$.

It is expected that 1 second settling time will work well with this system. If the settling time that is long, then the transfer function would take a while to reach the desired steady state output. This could affect the application, where timely movements are required. Inversely, if the settling time were to be too short then the movements could seem uncontrolled and experience other errors in application such as the unforeseen vibrational responses.

Table 3: Closed Loop Desired Performance Specifications

Closed Loop Performance Specification	Value
Percent Overshoot, PO	N/A
Settling Time, t_s	1 sec
Rise Time, t_R	0.75 sec
Peak Time, t_p	N/A
Poles (Stability)	$s_{1,2} < 0$
Poles (Critically Damped)	$s_1 = s_2$

Desired Characteristic Equation

Now the Desired Characteristic Equation can be derived from the subsequent performance specifications. To do that this equation can be utilized:

$$\Delta_{DES}(s) = s^2 + (2\omega_n\xi)s + \omega_n$$

This equation introduced two variables ω_n and ξ . In order to have a critically damped system, $\xi = 1$. This gives ω_n as an unknown, but this can be determined from the desired settling time by using:

$$t_S = \frac{4}{\xi * \omega_n} = 1 \text{ second}$$

We can rearrange this equation and then plug in the values of $\xi = 1$ and $t_S = 1$ to find the ω_n .

$$\omega_n = \frac{4}{t_S * \xi} = \frac{4}{1 * 1} = 4 \text{ rad/s}$$

To confirm this result, we can check the rise time t_R .

$$t_R = \frac{2.16\xi + 0.60}{\omega_n} = 0.69 \text{ second}$$

Originally the desired rise time was 0.75 seconds and this result is very close with a percent difference of only 8%. Now that we have a confirmed reasonable result, this can be substituted back into the desired character equation to get:

$$\Delta_{DES}(s) = s^2 + 8s + 16$$

Now the poles can be determined as $s_{1,2} = -4$ and this satisfies the pole requirements stated in Table #. A third order system will need to be designed to mimic a second order system to complete parameter matching for some of the controllers. To do that for this system a third pole will need to be introduced that is ten times the poles previously determined, therefore $s_3 = -40$. When we multiply this to our desired equation, the 3rd order equation is developed:

$$\Delta_{DES}(s) = s^3 + 48s^2 + 336s + 640$$

In order to verify the just how closely this 3rd order equation resembles the 2nd order previously developed, it is plotted in MatLab. In Figure 9 the difference is shown and can be considered marginal.

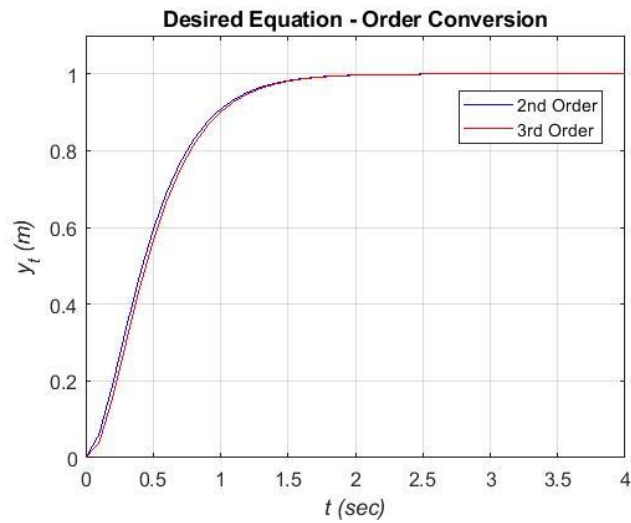


Figure 9: Comparison Plot for the desired 2nd order and 3rd characteristic polynomials.

Controller (G_c) Methods

To meet the desired a controller (G_c) will need to be designed. This paper will explore 3 potential controller designs: Lead Controller, Proportional Controller and a PID Controller.

Lead Controller:

A lead controller will increase stability while speeding up the transient response. Both of which is needed in our step input for the open loop function. To implement a lead controller, it will need to be put in series with the open loop transfer function, this is shown in the block diagram in Figure 10.

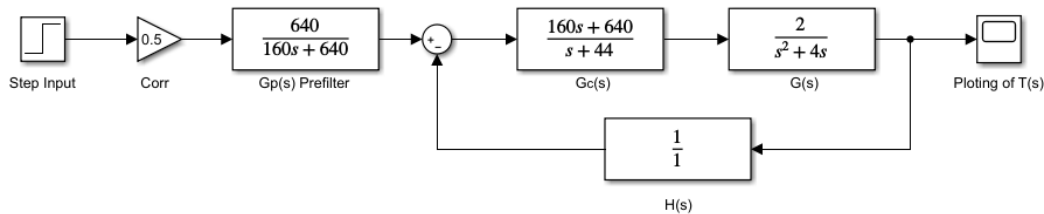


Figure 10: Lead Controller Block Diagram

This figure also shows an attenuation correction factor (Corr) and a prefilter ($G_p(s)$). The attenuation factor corrects the output magnitude to the desired value, which in this case is a steady state value of 1. The implementation of a lead controller produced an unwanted zero and the prefilter eliminates that zero.

This of course is the final system design to implement the lead controller, but several calculations were needed to arrive at this conclusion. Appendix D shows the complete work involved to derive these functions, but a summary is given in the next few paragraphs.

The general form of the lead controller is:

$$G_c(s) = \frac{k(s + a)}{(s + b)}$$

When combined in series with the open loop transfer function, the outcome for the characteristic polynomial is determined to be:

$$\Delta = s^3 + (4 + b)s^2 + (4b + k)s + ka$$

Denominator matching can be used with the desired 3rd order characteristic polynomial equation to solve the variables. The variables solve to be $a = 4$, $b = 44$ and $k = 160$. From here, the lead controller is now:

$$G_c(s) = \frac{160(s + 4)}{(s + 44)} = \frac{160s + 640}{s + 44}$$

Once combined in series the T(s) becomes:

$$T(s) = \frac{320s + 1280}{s^3 + 48s^2 + 336s + 640}$$

Once the limit of T(s) as s goes to 0 is taken, the steady state value will be 2 instead of the desired output of 1. To correct this a correction factor of 0.5 is needed. Additionally, it is shown that there is now an unwanted zero. To eliminate this a prefilter will be needed. Care will be needed in order to not attenuate the output so the numerator of the T(s) will need to remain 640. Therefore, the final T(s) becomes:

$$T(s) * Gp(s) * Corr = \cancel{0.5} * \frac{\cancel{320s + 1280}}{s^3 + 48s^2 + 336s + 640} * \frac{640}{\cancel{160s + 640}} = \frac{640}{s^3 + 48s^2 + 336s + 640}$$

Once plotted, as shown in Figure 11, it is apparent that our controller design was largely successful. There is no percent overshoot as desired and both the rise time and settling time remain close to the target values previously determined. The infinite outputs of the open loop transfer function are fixed by the lead controller. Table 4 summarizes the controller results.

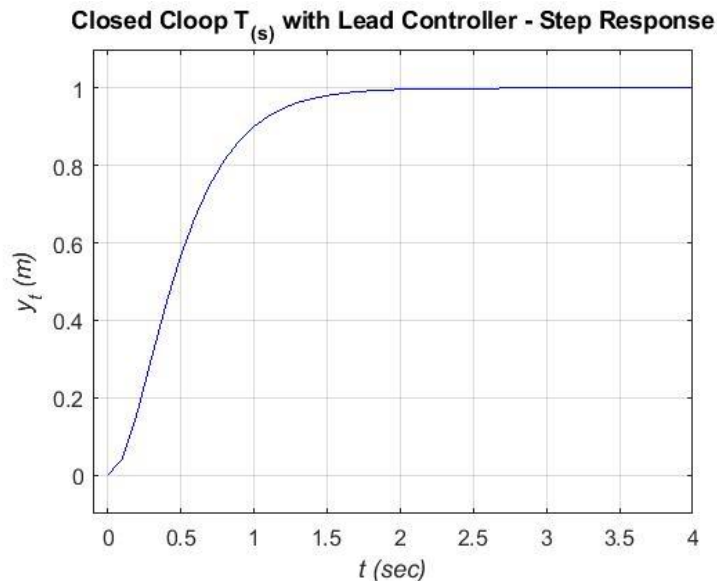


Figure 11: Lead controller closed loop transfer function result.

Table 4: Closed loop performance specifications results

Closed Loop Lead Controller Result	Value
Percent Overshoot, PO	N/A
Settling Time, t_s	1.4846 sec
Rise Time, t_R	0.8418 sec
Peak Time, t_p	N/A
Poles	$S_{1,2} = -4; S_3 = -40$
Zeros	N/A

Proportional Controller:

A proportional controller is a type of linear feedback control system. The controlled variable has a correction applied that is proportional to the difference between the desired value and the measured value. To implement a proportional controller, it will need to be put in series with the open loop transfer function, this is shown in the block diagram in Figure 12.

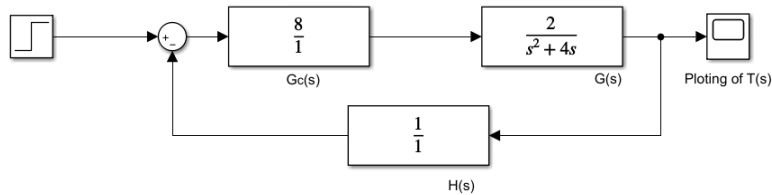


Figure 12: Proportional Controller Block Diagram

To determine the controller transfer function, $G_c(s)$ was implemented using the variable K .

$$G_C(s) = K$$

The variable for K determined based upon the desired closed loop function is found using denominator parameter matching. The $K = 8$ is then used to find $T(s)$. The closed loop transfer function determined using a proportional controller is shown in the equation below.

$$T(s) = \frac{16}{s^2 + 4s + 16}$$

The performance specifications were determined using the variables in the equation above.

Table 5: Proportional controller performance specifications.

Performance Specification	Closed Loop
Percent Overshoot, PO	10.8%
Settling Time, t_s	2 s
Rise Time, t_R	.42 s
Peak Time, t_p	1.11 s
e_{ss}	0
Poles	$S_{1,2} = -4$
Zeros	N/A

By evaluating the controller performance in simulation, the performance specifications below were validated using Figure 13. This shows that with a proportional controller the steady state value of 1 can be achieved at a settling time of 2 s.

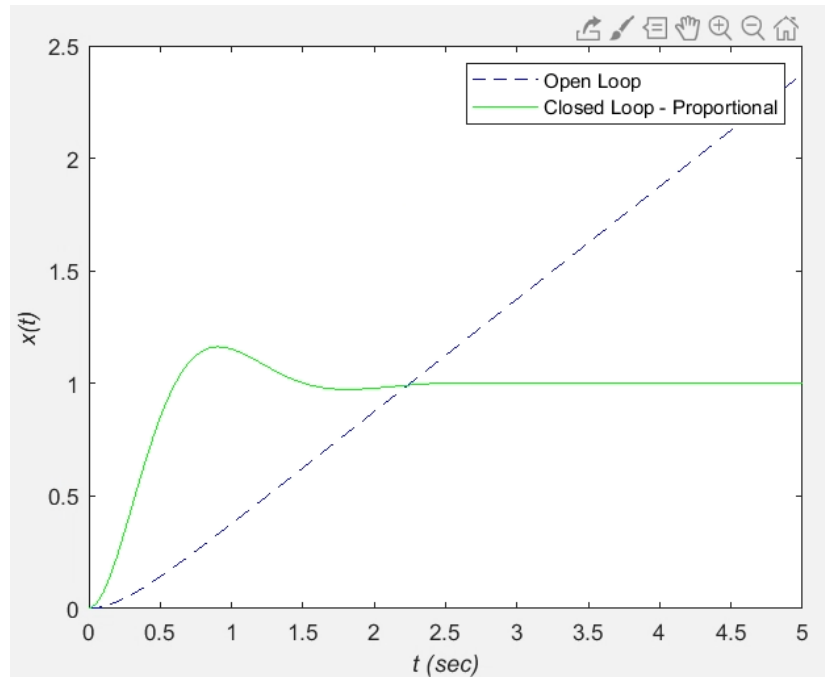


Figure 13: Proportional controller open-loop vs closed-loop plot comparison.

PID Controller:

A PID controller stands for Proportional, Integral, Derivative of error ($E(s)$). PID is a control loop mechanism employing feedback, these are widely used in industrial control systems since they are simple and effective.

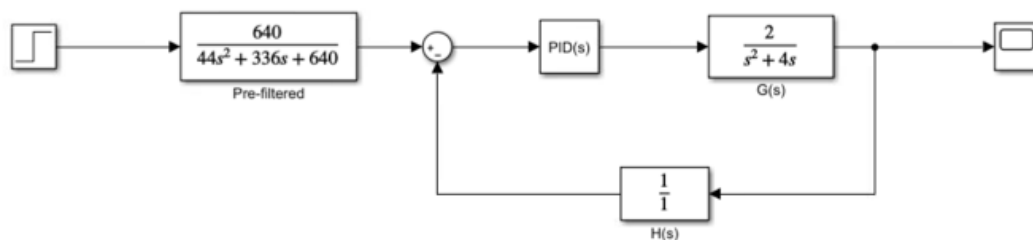


Figure 14: PID Controller Block Diagram

The figure above shows the Simulink for the PID closed- loop controller.

The general form of the PID controller is:

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s$$

The simplified version of the general form is:

$$G_c(s) = \frac{K_D s^2 + K_P s + K_I}{s}$$

Below you can see the delta design equation which is explained and solved on the bottom of page 11.

$$\Delta_{DES}(s) = s^3 + 48s^2 + 336s + 640$$

We then plugged the general form equation into the general transfer function. Then compared the denominator of the transfer function to the delta designed equation to find the values of the unknowns. The value of $K_D = 22$, $K_P = 168$, and to be $K_I = 320$. From here the new PID controller equation is:

$$T(s) = \frac{44s^2 + 336s + 640}{s^3 + 48s^2 + 336s + 640}$$

We then plug this into the general PID equation to get the following:

$$G_c(s) = \frac{22s^2 + 168s + 320}{s}$$

After this we found the Pre-filtered transfer function. The job of the Pre-filtered $G_P(s)$ is to knock out the unwanted zeros. The $G_P(s)$ equation is:

$$G_P(s) = \frac{640}{44s^2 + 336s + 640}$$

After finding all the equations above, we then graphed them all on the same axis to show the comparison between the open loop, closed loop, and the closed loop with Pre-filtered. This can all be found below in Figure 15. As you can see from the figure the open loop and the prefiltered loop both approach zero.

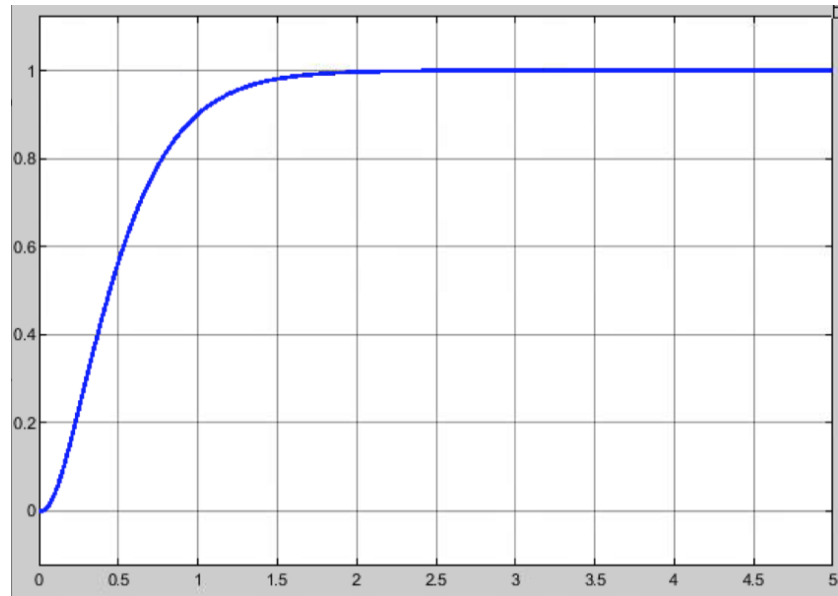


Figure 15: PID controller open-loop vs closed-loop plots comparison.

Table 6: PID controller performance specifications.

Performance Specification	Open Loop Values	Closed Loop Values	Closed Loop Values with Pre-Filter
Percent Overshoot, PO	N/A	10.5%	N/A
Settling Time, t_s	∞	0.56	1.048
Rise Time, t_R	∞	0.13	0.724
Peak Time, t_P	∞	1.07	1
Poles	$S_1 = -4; S_2 = 0;$	$S_{1,2} = -4; S_3 = -40$	$S_{1,2} = -4; S_3 = -40$
Zeros	N/A	$S = -4, S = -3.64$	N/A

Closed Loop Input Effort

Now that the controllers have been designed, it is important to compare the input effort of each controller. This is best analyzed in Simulink. A smaller amount of input effort is desired as this will lessen the required resources. These resources will likely be electricity usage in our robotic arm.

Lead Controller:

For the lead controller the Simulink configuration is shown in Figure 16. The effort scope is located between the lead controller and the open loop transfer function.

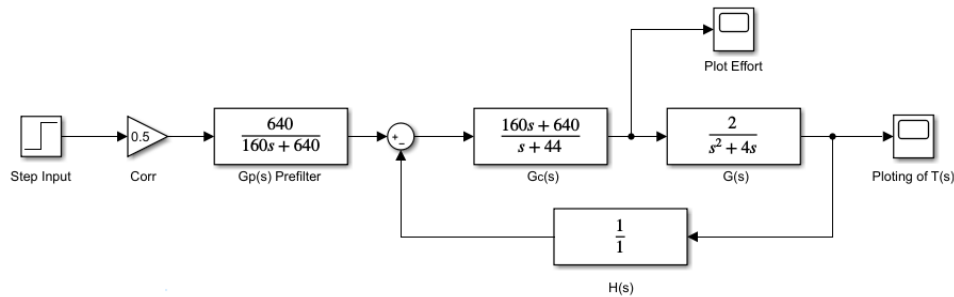


Figure 16: Lead controller block diagram with input effort scope.

The lead controller will require a larger amount of input effort at the beginning of the response. This will then converge to zero, where not much effort is needed after 0.8 seconds. This effort plot can be seen in Figure 17.

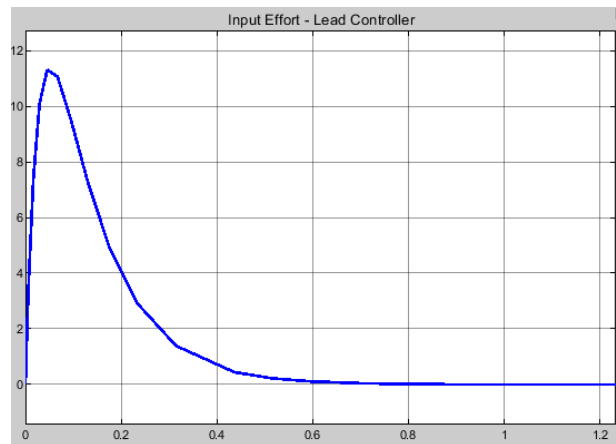


Figure 17: Lead Controller input effort over time plot.

Proportional Controller:

For the proportional controller, the Simulink configuration is shown in Figure 18.

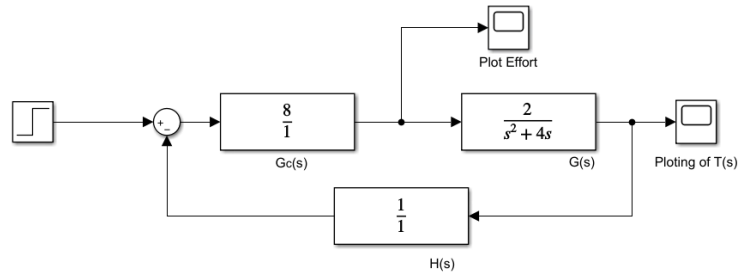


Figure 18: Proportional controller block diagram with input effort scope.

The input effort of the system is shown below in Figure 19. This shows that it takes a large amount of effort right away and then balances out at zero around 2 seconds.

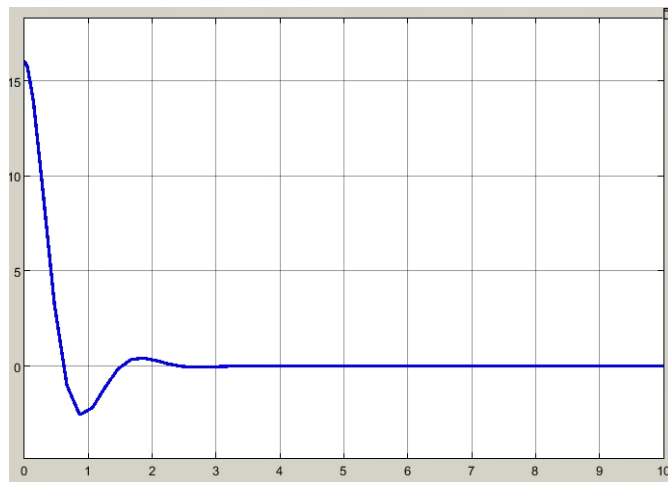


Figure 19: Proportional controller input effort over time plot.

PID Controller:

For the PID controller the Simulink configuration is shown in Figure 20.

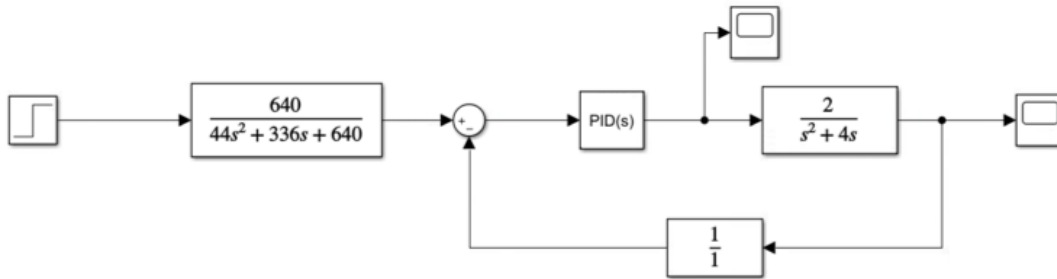


Figure 20: PID controller block diagram with input effort scope.

The PID controller will require a larger amount of input effort at the beginning of the response. This will then converge to zero, where not much effort is needed after 1.5 seconds. This effort plot can be seen in Figure 21.

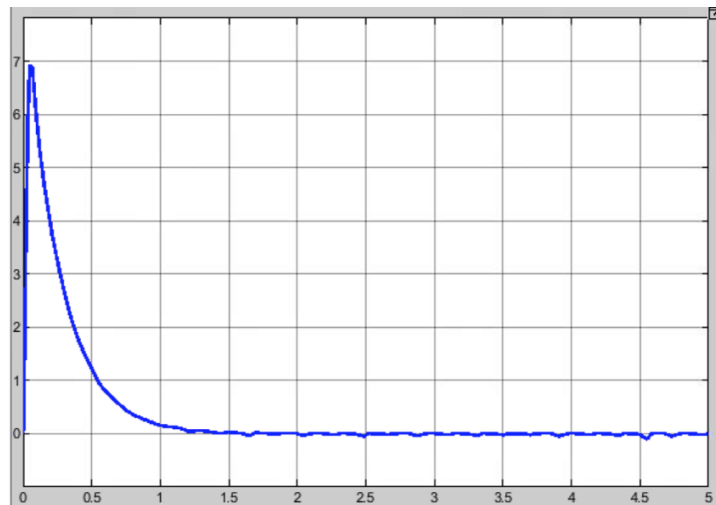


Figure 21: PID controller input effort over time plot.

Closed Loop Disturbances

Disturbances are unaccounted effects in the system, and these can show up in many ways such as friction, gravity effects, wind drag or wearing of the product. In our project we are likely to experience frictional disturbances while the arm is rotating. For this reason, the effect of these disturbances will need to be analyzed to ensure the best controller is chosen. Simulink is used in this section as it has a few advantages over its MatLab counterpart. Mainly that it can handle two input sources, one for the disturbance and the other for reference.

Lead Controller:

For the lead controller the Simulink configuration is shown in Figure 22.

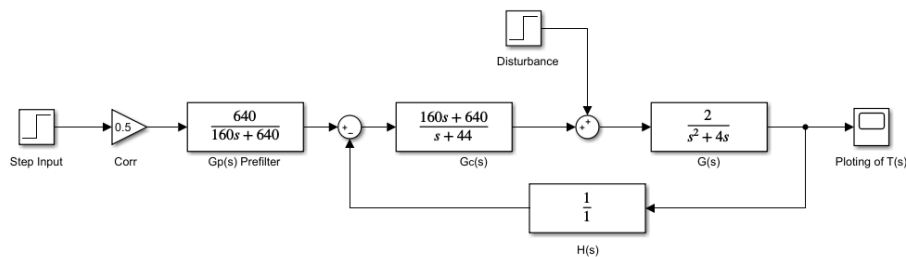


Figure 22: Lead controller block diagram with disturbance input.

The output for this configuration is shown in Figure 23. This plot shows that the steady state value has been altered due to the disturbance. This difference is small and in its steady state value in the output response is 1.1374.

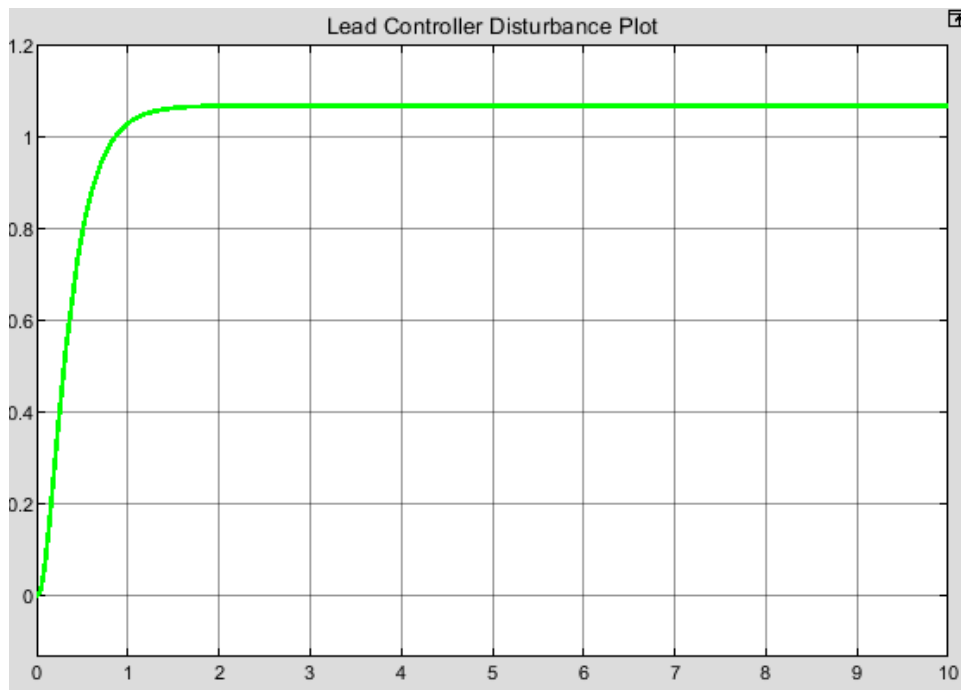


Figure 23: Lead controller output plot with disturbance.

Proportional Controller:

For the proportional controller, the Simulink configuration is shown in Figure 24.

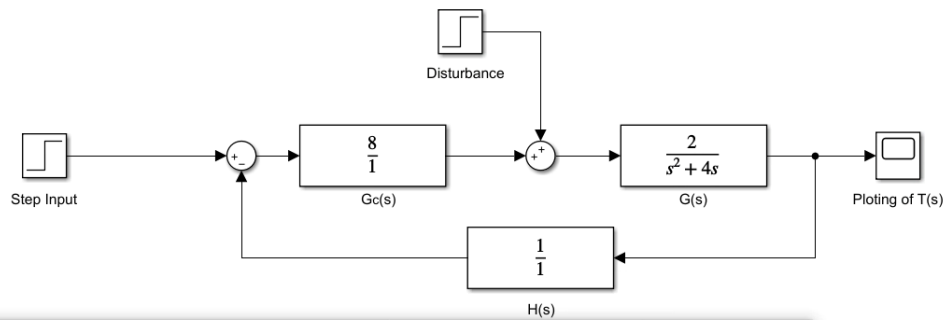


Figure 24: Proportional controller block diagram with disturbance input.

The output for this configuration is shown in Figure 25. This plot shows that a disturbance causes the system to go well above the steady state value of 1. This is not an ideal controller and should not be used.

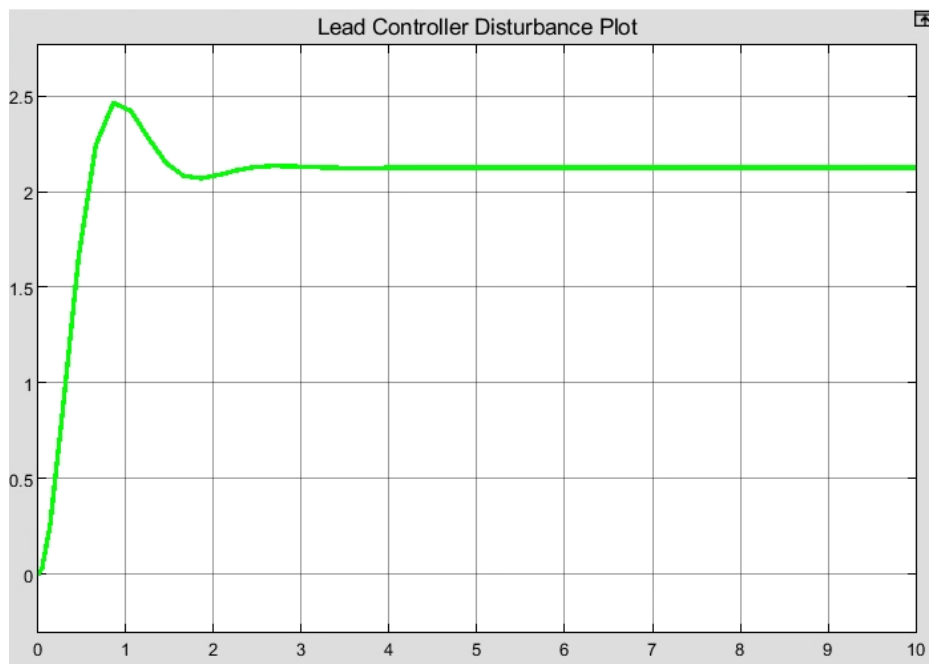


Figure 25: Proportional controller output plot with disturbance.

PID Controller:

For the PID controller the Simulink configuration is shown in Figure 26.

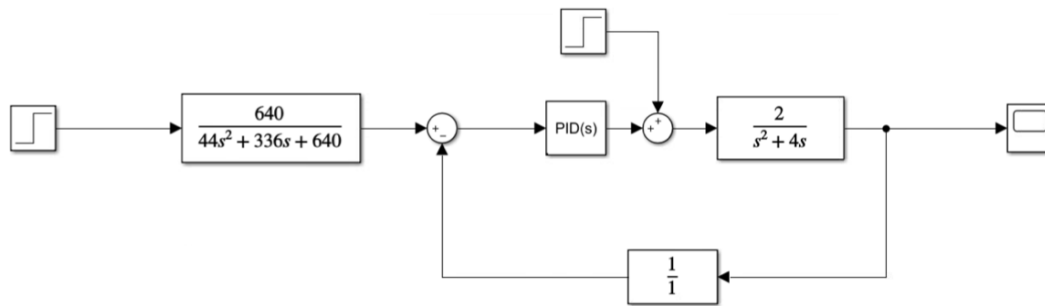


Figure 26: PID controller block diagram with disturbance input.

The output for this configuration is shown in Figure 27. As you can see there is no difference between the pre-filtered plot and the plot for the disturbance. Its steady state value in the output response is still 1.

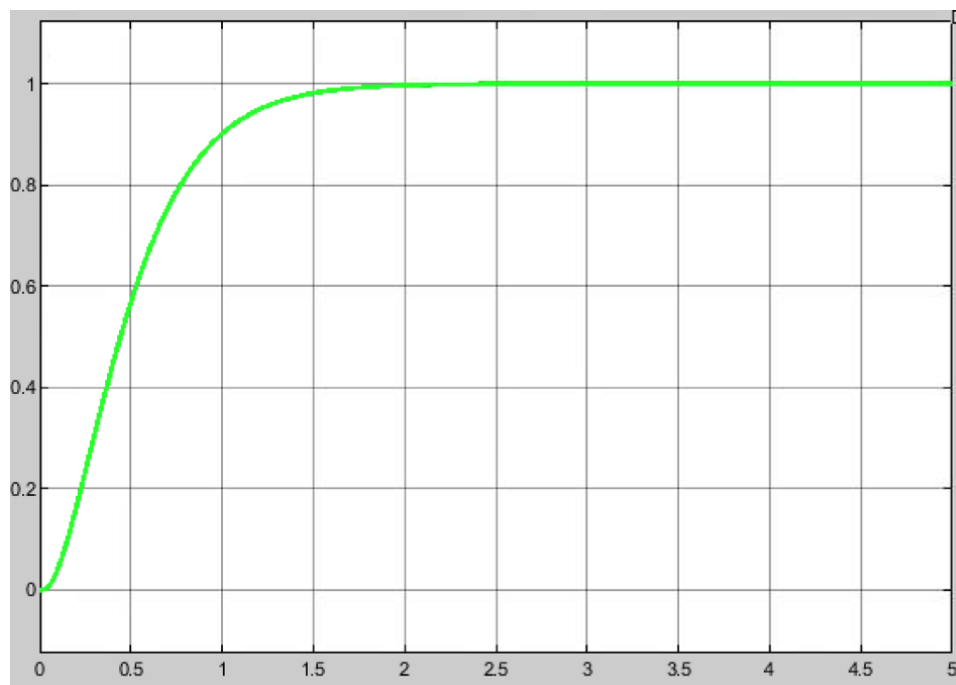


Figure 27: PID controller output plot with disturbance.

Final Controller Decision

The PID controller will work the best for this system. This controller meets or closely matches the desired performance specs of a steady state output of 1, a rise time of 1 second and a settling time of 0.72 seconds. While all the controller methods met these specifications closely, the PID controller was able to do this with the least amount of input effort. Additionally, the PID controller has the best disturbance response showing almost no effect on the output plot. This is an especially good thing to have with our arm that as mentioned previously this will likely see multiple kinds of disturbances.

References

- [1] R. L. Williams, *Linear Systems Control for Mechanical Engineers*. LuLu.
- [2] *Automata aims to "democratise robotics" with \$3000 six axis robot*. YouTube, 2015.
- [3] B. Williams, *An Atlas of Engineering Dynamics Systems, Models, and Transfer Functions*.

Appendix A: MATLAB Code

Impulse function

```
%-----  
% Amanda Przybocki, Anna Kelley, Jon Bowman  
% Dr. Bob  
% ME 3012  
% Interim report Robot Elbow Control (impulse)  
%-----  
  
%clear previous values as well as the command window  
clear;clc;  
  
% format the command window shorter  
format compact;  
  
numG = [2];  
denG = [1 4 0];  
sysG = tf(numG,denG);  
zerosG = roots(numG);  
polesG = roots(denG);  
t = [0: .005: 2];  
[yo,xo] = impulse(sysG,t);  
  
figure;  
grid;  
plot(t,yo,'r');  
axis ([0 2 0 0.6]);  
xlabel('\itt (\itsec)');  
ylabel('\itx(t)');  
title('Impulse Response (Open-Loop)');
```

Unit Step Function

```
%-----  
% Amanda Przybocki, Anna Kelley, Jon Bowman  
% Dr. Bob  
% ME 3012  
% Interim report Robot Elbow Control (step)  
%-----  
  
%clear previous values as well as the command window  
clear;clc;  
  
% format the command window shorter  
format compact;  
  
numG = [2];  
denG = [1 4 0];  
sysG = tf(numG,denG);  
zerosG = roots(numG);  
polesG = roots(denG);  
t = [0: .05: 10];  
[yo,xo] = step(sysG,t);  
  
figure;  
grid;  
plot(t,yo,'b');  
axis ([0 10 0 5]);  
xlabel('\itt (\itsec)');  
ylabel('\itx(t)');  
title('Step Response (Open-Loop)');
```

Unit ramp function

```
%-----  
% Amanda Przybocki, Anna Kelley, Jon Bowman  
% Dr. Bob  
% ME 3012  
% Interim report Robot Elbow Control (unit ramp)  
%-----  
  
%clear previous values as well as the command window  
clear;clc;  
  
% format the command window shorter  
format compact;  
  
numG = [2];  
denG = [1 4 0];  
sysG = tf(numG,denG);  
zerosG = roots(numG);  
polesG = roots(denG);  
t = [0: .005: 10];  
[yo,xo] = lsim(sysG,t,t);  
  
figure;  
grid;  
plot(t,yo);  
axis ([0 10 0 25]);  
xlabel('\itt (\itsec)');  
ylabel('\itx(t)');  
title('Unit Ramp Response (Open-Loop)');
```

Characteristic Equation Order Comparisons

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Jonathan Bowman, Anna Kelley, Amanda Przybocki
% ME 3012 Dr. Bob
% Desired Characteristic Equation - Order Comparison
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear;
clc;

% Desired 2nd Order Characteristic Equation
num1 = [16];
denS1 = [1 8 16];
polesS1 = roots(denS1)

S1 = tf(num1,denS1);
t1 = [0:0.1:60];
y1 = step(S1,t1);

stepinfo(S1)

% Desired 3rd Order Characteristic Equation
num2 = [640];
denS2 = [1 48 336 640];
polesS2 = roots(denS2)

S2 = tf(num2,denS2);
t2 = [0:0.1:60];
y2 = step(S2,t2);

stepinfo(S2)

%Plotting
figure;
plot(t1,y1,'b',t2,y2,'r');
title("Desired Equation - Order Conversion")
set(gca,'FontSize',12);
grid; axis([0 4 0 1.1]);
ylabel('\ity_t (m)'); xlabel('\itt (\itsec)');
legend('2nd Order','3rd Order')
```


Closed Loop - Lead Controller (with Pre-Filter and Corr) vs. Open Loop Gs

```
*****
% Jonathan Bowman, Anna Kelley, Amanda Przybocki
% ME 3012 Dr. Bob
% Step Function for Final T(S) Lead Controller,
% Atten., Prefilt vs the original open loop equation.
*****

clear;
clc;

% Open Loop Function G(s)
num1 = [2];
denS1 = [1 4 0];
polesS1 = roots(denS1)

S1 = tf(num1,denS1);
t1 = [0:0.1:60];
y1 = step(S1,t1);

stepinfo(S1)

% Closed Loop Function T(s)
num2 = [640];
denS2 = [1 48 336 640];
polesS2 = roots(denS2)

S2 = tf(num2,denS2);
t2 = [0:0.1:60];
y2 = step(S2,t2);

stepinfo(S2)

%Plotting
figure;
plot(t1,y1,'b',t2,y2,'r');
title("T_(s) with Lead Controller vs. G_(s) - Step Response")
set(gca,'FontSize',12);
grid; axis([0 4 0 1.1]);
ylabel('\ity_t (m)'); xlabel('\itt (\itsec)');
legend('G_s','T_s')
```

Closed Loop - PID Controller (with Pre-Filter) vs. Open Loop $G(s)$

```
% Amanda Przybocki
% Dr. Bob
% ME 3012
% HW 5
%-----

%clear previous values as well as the command window
clear;clc;

% format the command window shorter
format compact;

num = [2];
den = [1 4 0];
T = tf(num, den);
t = [0: 0.025: 3];
[yo, xo] = step(T, t);

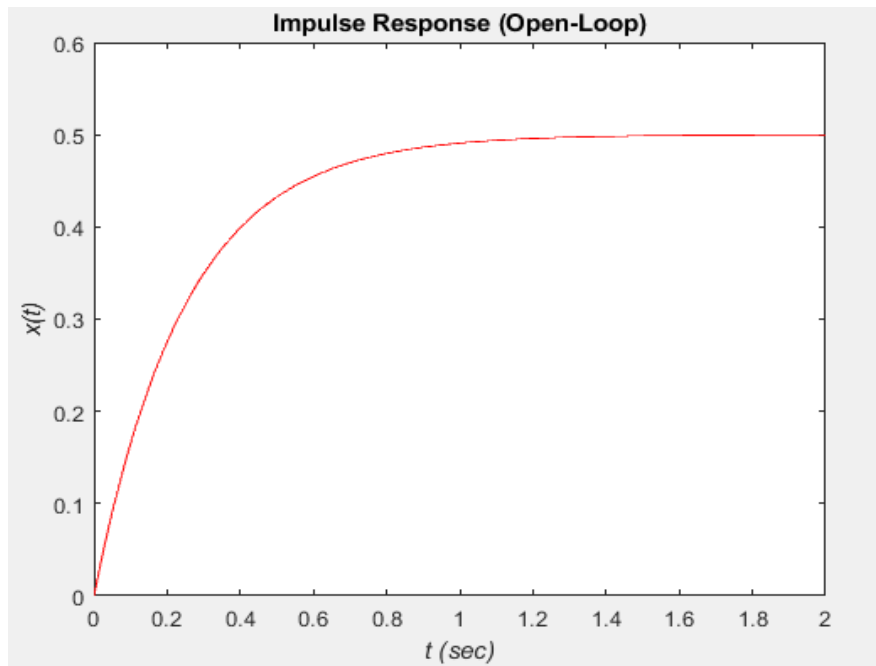
num_c = [44 336 640];
den_c = [1 48 336 640];
G_c = tf(num_c, den_c);
[yc, xc] = step(G_c, t);

num_r = [640];
den_r = [44 336 640];
G_r = tf(num_r, den_r);
[yr, xr] = step(G_r, t);

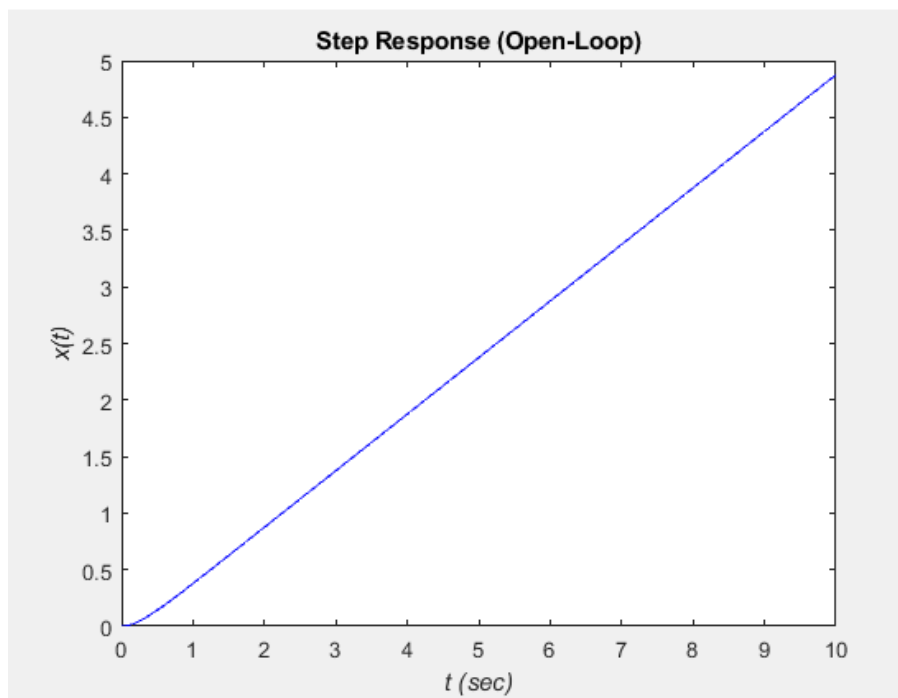
figure;
plot(t, yo, 'b', t, yc, 'r', t, yr, 'g');
legend('Open loop', 'Closed loop', 'Closed loop w/ Pre-filtered');
ylabel('Yt(m)');
xlabel('T(s)');
title('PID Controller');
grid;
```

Appendix B: MATLAB Plots

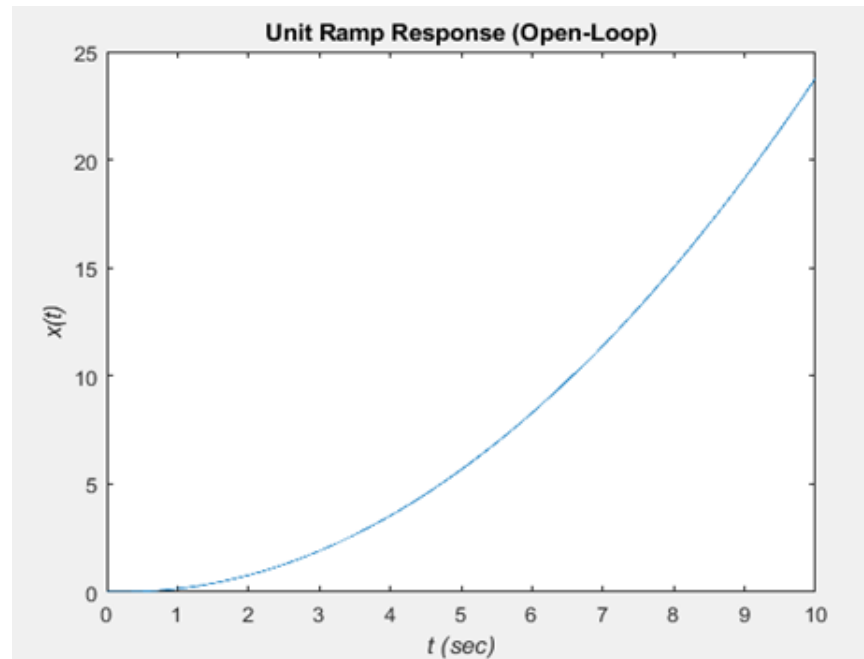
Impulse function



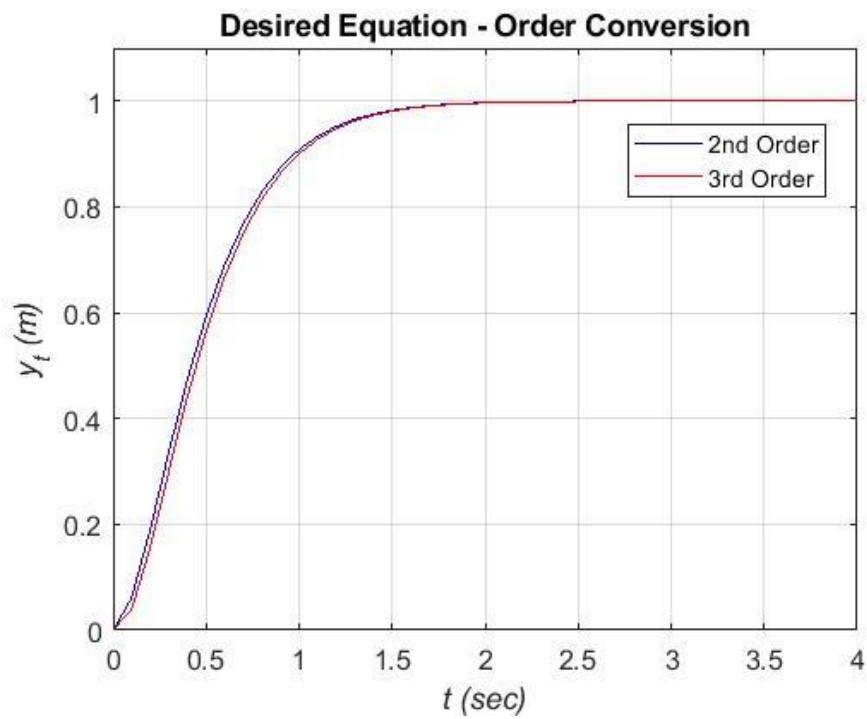
Unit Step Function



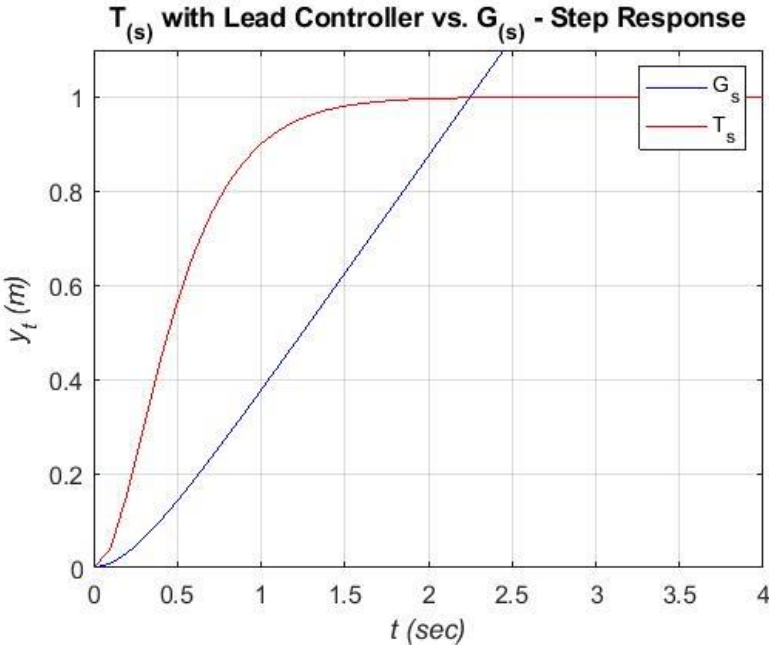
Unit ramp function



Characteristic Equation Order Comparison



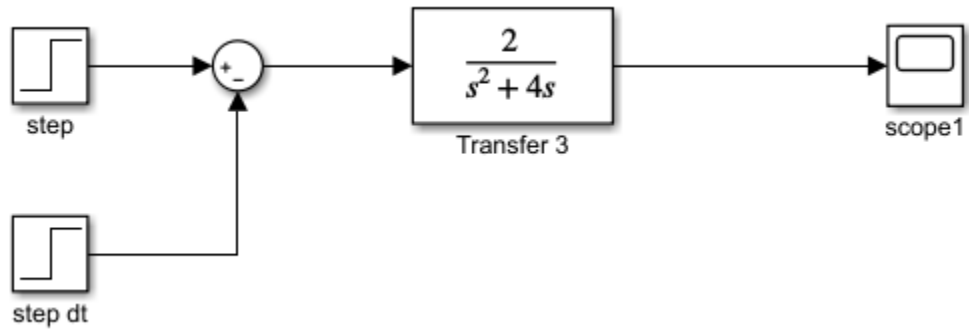
Closed Loop - Lead Controller (with Pre-Filter and Corr) vs. Open Loop G_s



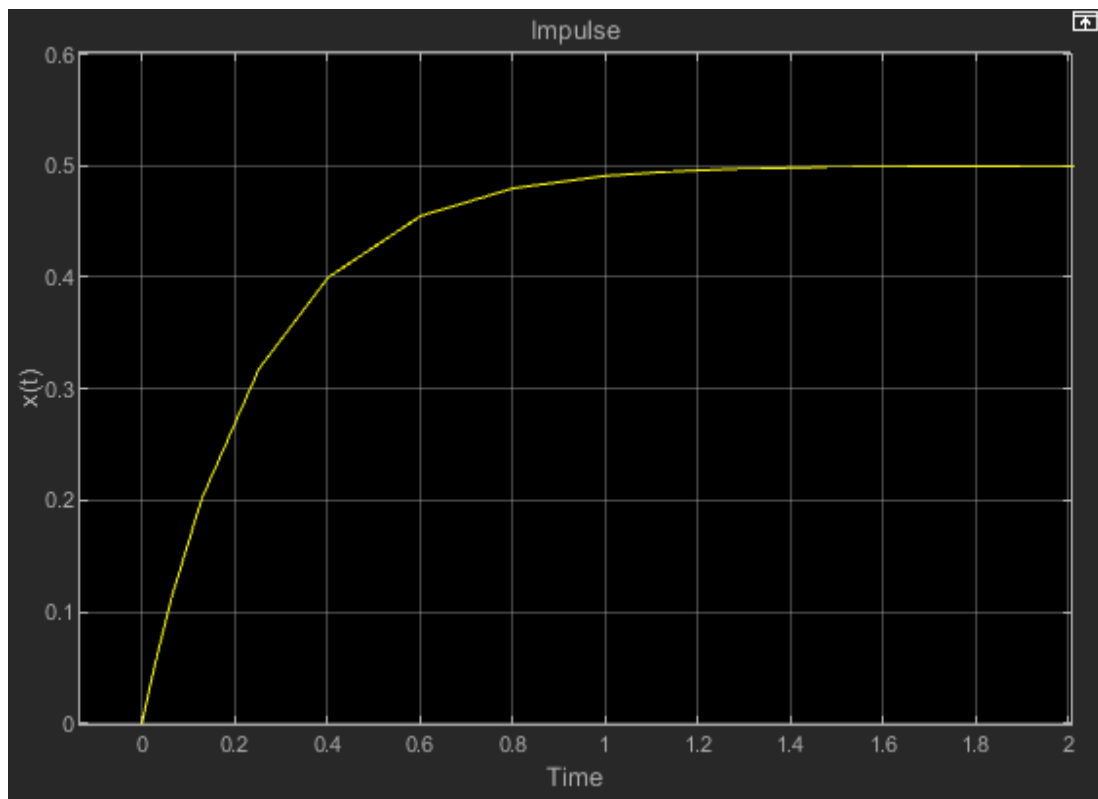
Appendix C: Simulink Results

Impulse Function

- Simulink Block Diagram

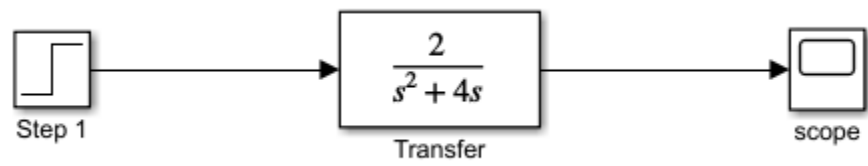


- Simulink Output Plot

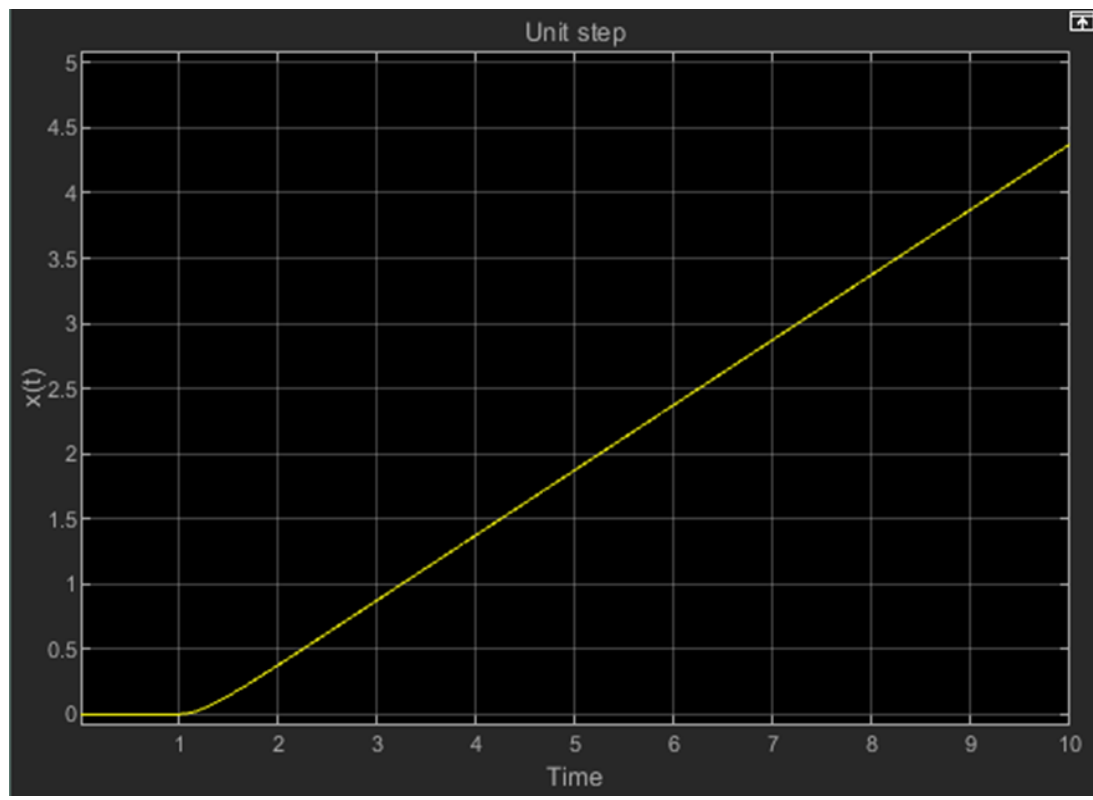


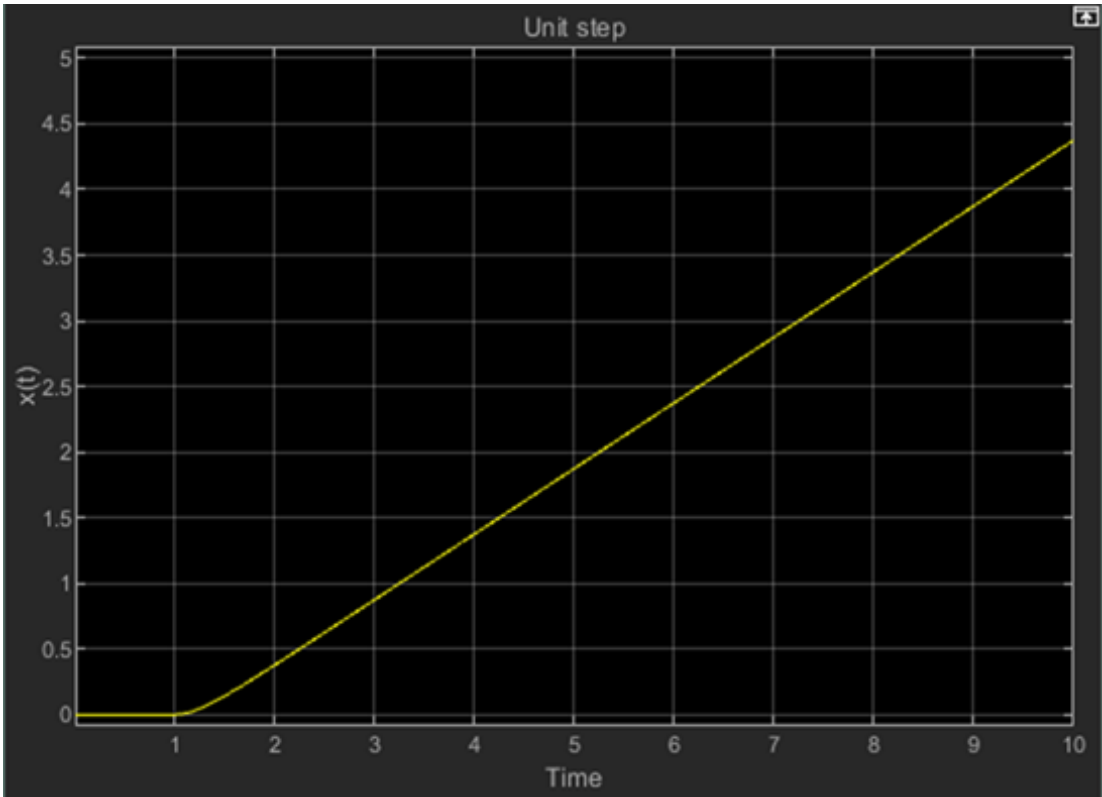
Unit Step Function

- Simulink Block Diagram



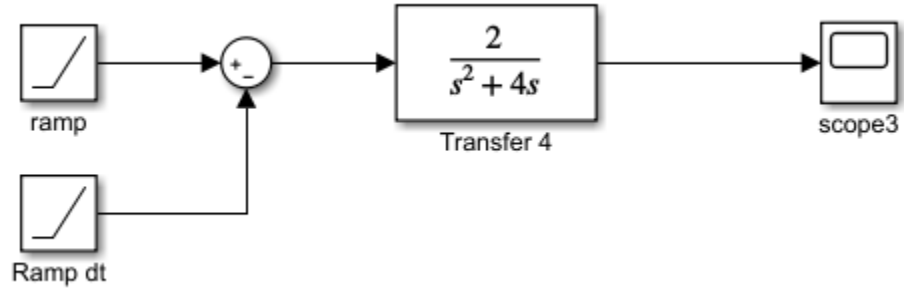
- Simulink Output Plot



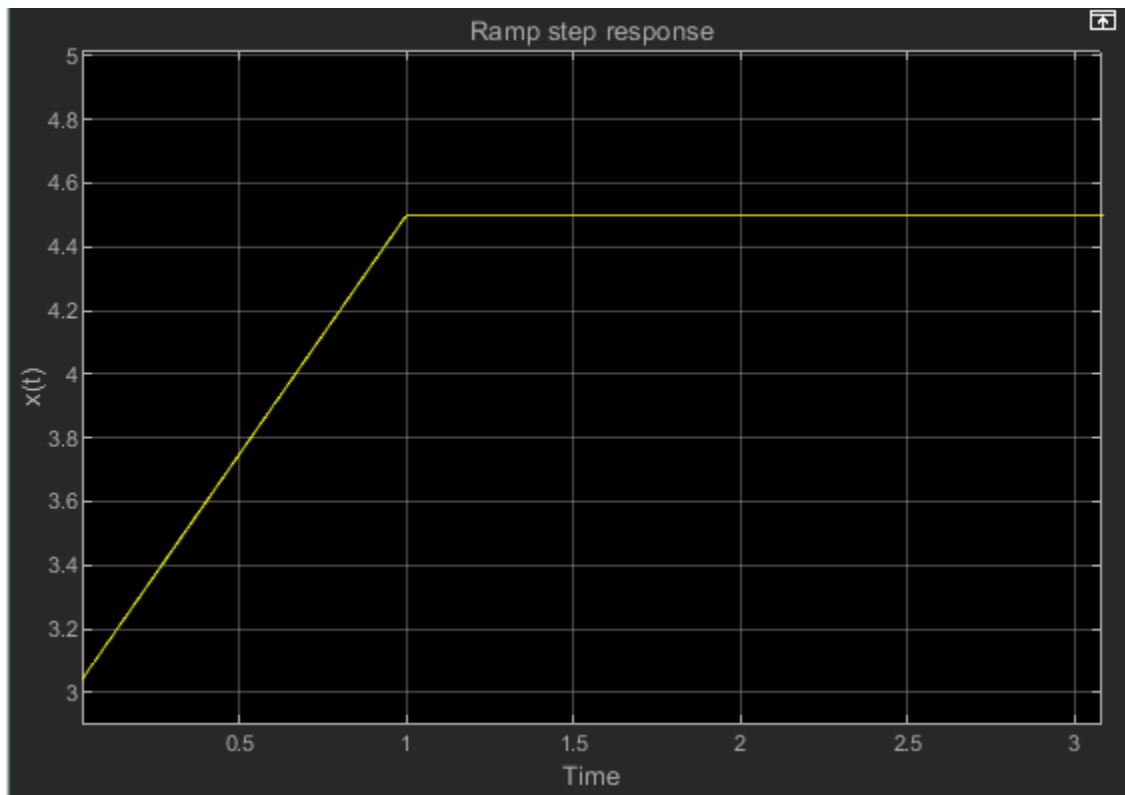


Ramp step function

- Simulink Block Diagram

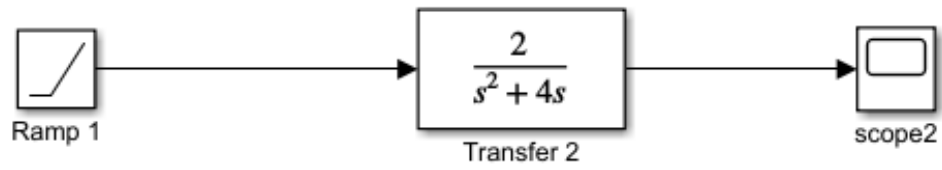


- Simulink Output Plot

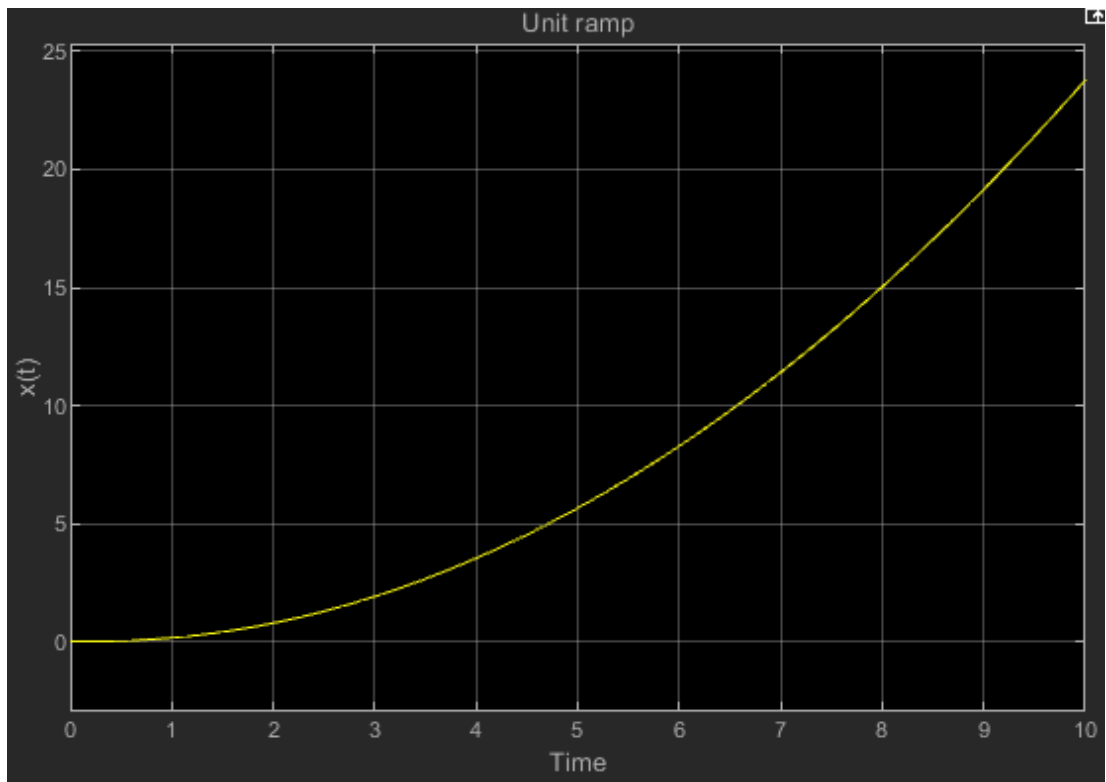


Unit ramp function

- Simulink Block Diagram

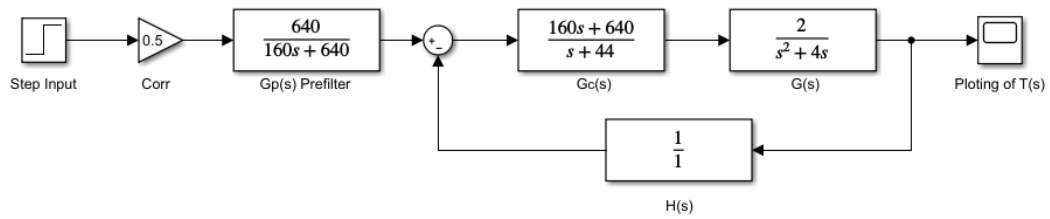


- Simulink Output Plot

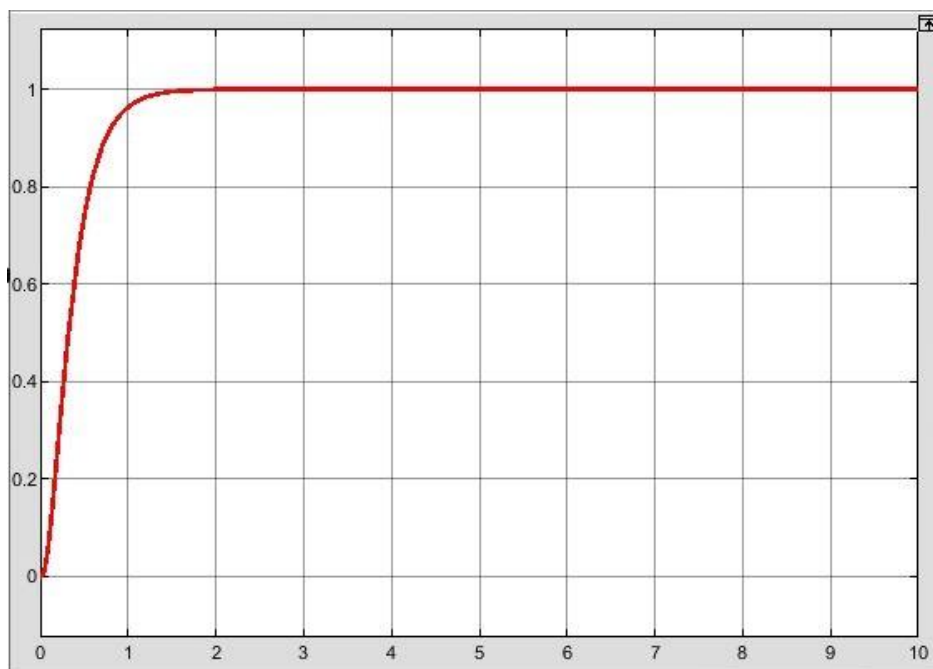


Closed Loop with a Lead Controller

- Simulink Block Diagram



- Simulink Output Plot



Appendix D: Lead Controller Calculations

Generic Lead Controller:

$$G_C(s) = \frac{k(s + a)}{(s + b)}$$

Closed Loop Transfer Function is:

$$T(s) = \frac{G_C(s) * G(s)}{1 + G_C(s)G(s)H(s)}$$

A perfect sensor is assumed so $H(s) = 1$ and this then reduces to:

$$T(s) = \frac{G_C(s) * G(s)}{1 + G_C(s)G(s)}$$

Plug in the values of $G_C(s)$ and $G(s)$ this Closed Loop Transfer Function Becomes:

$$T(s) = \frac{\frac{ks + ka}{s + b} * \frac{2}{s^2 + 4s}}{1 + \frac{ks + ka}{s + b} * \frac{2}{s^2 + 4s}}$$

Expand:

$$T(s) = \frac{\left(\frac{2ks + 2ka}{s^3 + 4s^2 + bs^2 + 4bs} \right)}{\left(\frac{s^3 + 4s^2 + bs^2 + 4bs + ks + ka}{s^3 + 4s^2 + bs^2 + 4bs} \right)}$$

Simplify:

$$T(s) = \frac{2ks + 2ka}{s^3 + 4s^2 + bs^2 + 4bs + ks + ka} = \frac{2ks + 2ka}{s^3 + (4 + b)s^2 + (4b + k)s + ka}$$

Use Denominator Matching

Start with the equations:

$$\Delta_{DES}(s) = s^3 + 48s^2 + 336s + 640$$

$$T(s)Denom = s^3 + (4 + b)s^2 + (4b + k)s + ka$$

Now compare the orders:

$$\begin{aligned}S^3 &\rightarrow 1 = 1 \\S^2 &\rightarrow 48 = 4 + b \\S^1 &\rightarrow 336 = 4b + k \\S^0 &\rightarrow 640 = ka\end{aligned}$$

This system of 3 equations and 3 unknowns solves for the variables:

$$\begin{aligned}a &= 4 \\b &= 44 \\k &= 160\end{aligned}$$

This results in the lead controller of:

$$G_C(s) = \frac{160(s + 4)}{(s + 44)} = \frac{160s + 640}{s + 44}$$

The closed loop transfer function then becomes:

$$T(s) = \frac{320s + 1280}{s^3 + 48s^2 + 336s + 640}$$

The steady state value results in:

$$\lim_{s \rightarrow 0} \frac{320s + 1280}{s^3 + 48s^2 + 336s + 640} = \frac{1280}{640} = 2$$

Since this does not approach 1, we need an output attenuation factor:

$$corr = \frac{Desired}{Actual} = \frac{1}{2}$$

Now we multiply this into the T(s):

$$T(s) = \frac{160s + 640}{s^3 + 48s^2 + 336s + 640}$$

To check this, we can see that:

$$\lim_{s \rightarrow 0} \frac{160s + 640}{s^3 + 48s^2 + 336s + 640} = \frac{640}{640} = 1$$

Finally, there's a zero that needs to be eliminated in the numerator. This can be done using the pre-filter:

$$Gp(s) = \frac{640}{160s + 640}$$

Once this is multiplied by the Closed Loop Function, the new closed loop function becomes:

$$T(s) = T(s) * Gp(s) = \frac{640}{s^3 + 48s^2 + 336s + 640}$$