ECE 4470:
Final Project

Time-Domain and Frequency Domain Controller Design

David Baron-Vega: GF7068
Professor: Dr. Feng Lin
Wednesday, April 17, 2024

Table of Contents:

Introduction:

Controllers allow us to manage the behavior of complex systems, like industrial manufacturing processes and automotive electronics. The choice and 'tuning' of each controller significantly influences a system's performance and parameters such as its response time, overshoot, and stability.

5 controllers were designed with specified parameters: (1)Proportional-Derivative (PD), (2)Proportional-Integral (PI), (3)Proportional-Integral-Derivative (PID), (4)Phase-lead, and (5)Phase-lag. We considered the same system for each type of controller, represented by the open-loop transfer function G(s)H(s):

Expanding the given G(s)H(s):

Expanded form

$$\frac{10}{s^3 + 27\,s^2 + 162\,s}$$

This project employs MATLAB and Simulink simulations to perform calculations and visualize the system's and controller's behavior and impact on system performance. Analytical/handwritten results are provided at the end of the report.

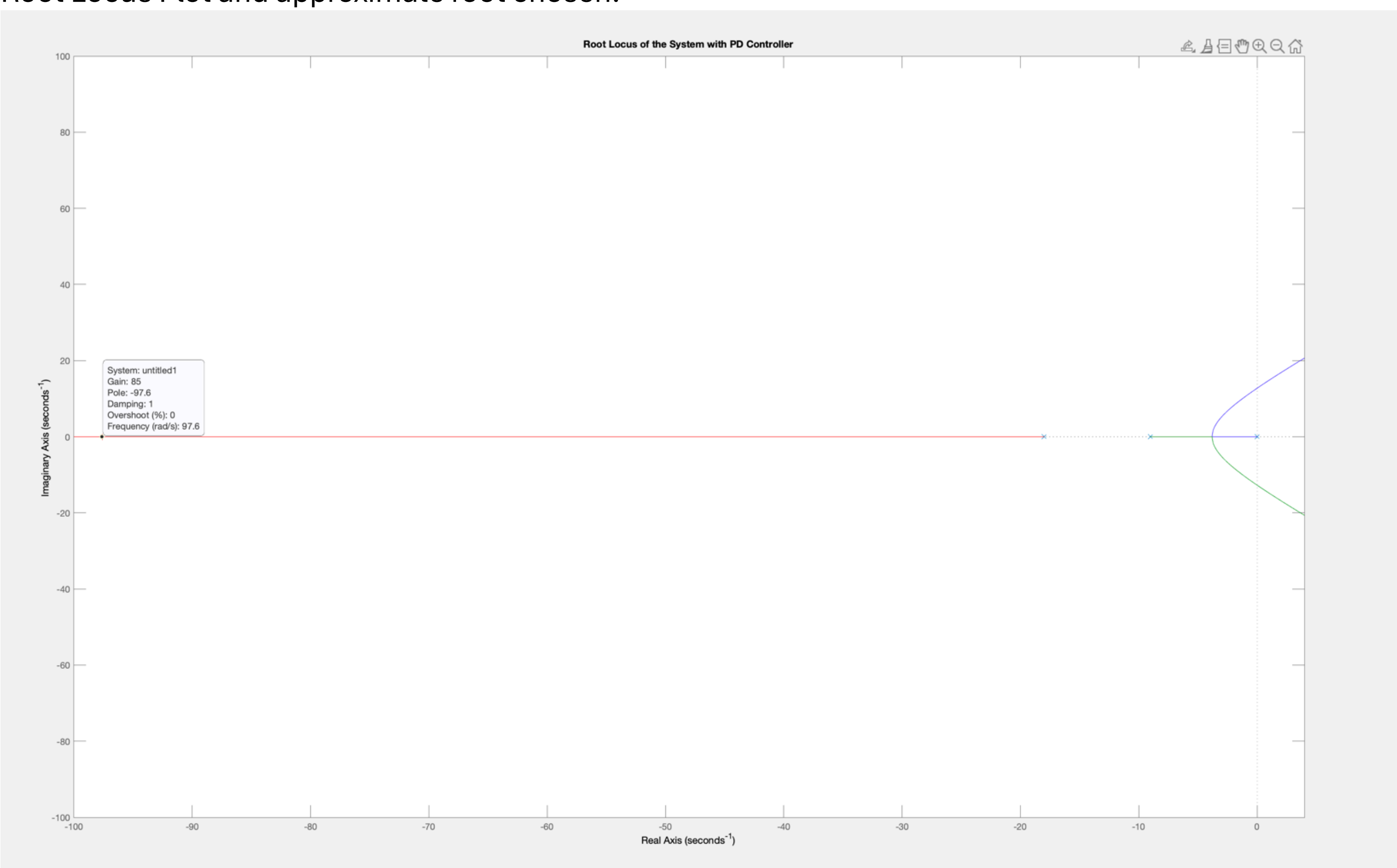
1. MATLAB/Simulink Results:

Controller 1: PD

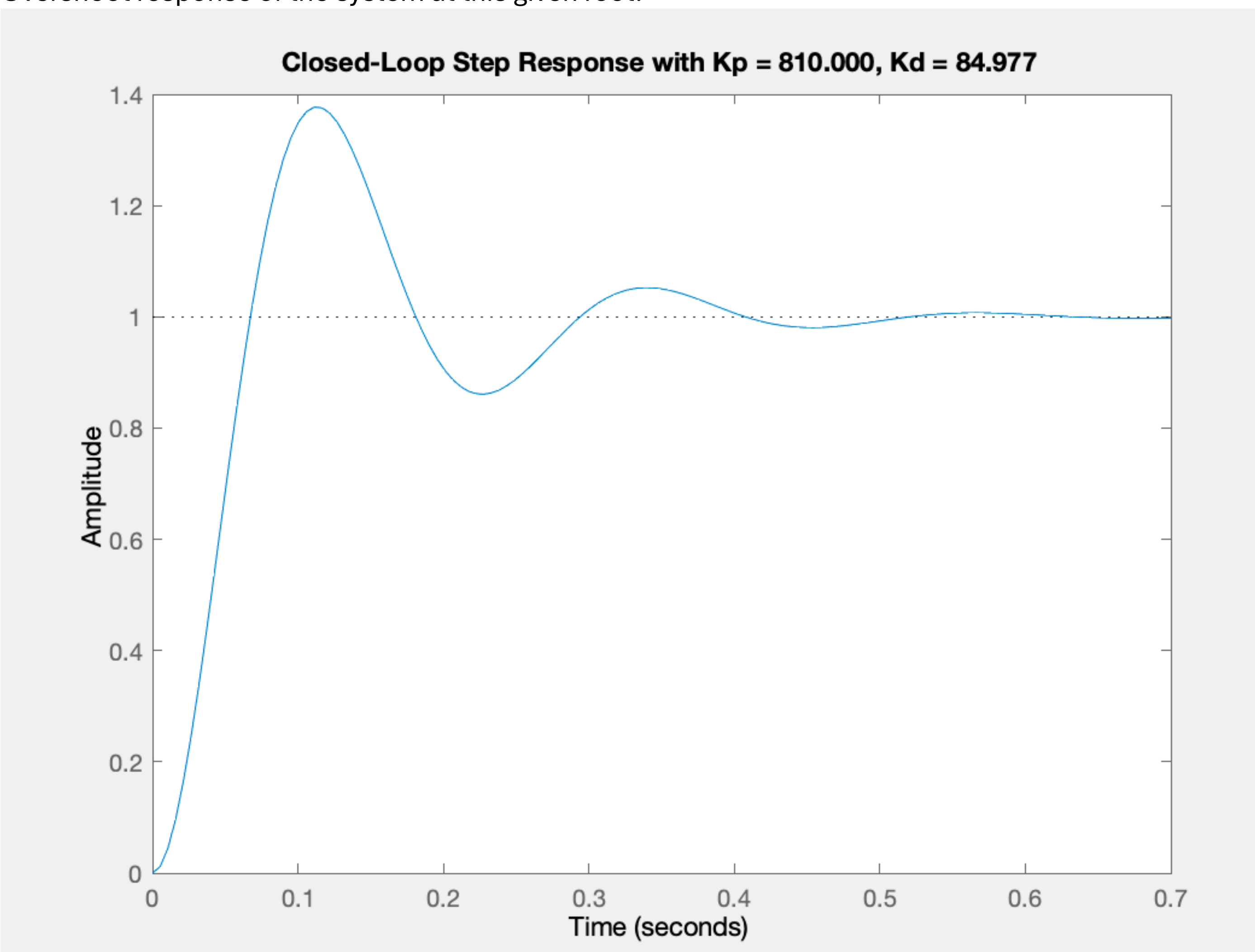
Code:

```matlab
%%% ECE 4470 - Final Project %%%
%%% David Baron-Vega - GF7068
%%% Started: Wednesday, April 3. Due: Monday, April 22

%{
"The goal of this project is to design controllers using the design techniques discussed; to simulate
closed loop systems; and to compare different controllers.
The system to be controlled is modeled by the following transfer function:

G(s)H(s) = 10/(s)(s+9)(s+18)"
%}

%% Part 1 %%
%1. Design a PD controller using time-domain design method such that (1) Kv = 50 and (2) overshoot is smallest.

%Defining the system we are working with. We have been provided with the
%system's Open-Loop transfer function, and can use built in MATLAB
%functionality 'tf' to work with transfer functions.

%We need to define the Laplace variable as a continuous function to work with it.
s = tf('s');

%Open-loop transfer function coefficients
num = 10;
den = [1 27 162 0]; % s(s+9)(s+18) expands to s^3 + 27s^2 + 162s

%Displaying the open-loop transfer function
fprintf('The open-loop function of the system is:\n');
G = tf(num, den)

%Calculating the velocity constant Kv as s approaches 0
Kv = dcgain(s*G);

% Displaying the required open-loop gain to achieve a ramp-error constant Kv of 50
fprintf('The open-loop gain necessary to achieve a (Kv) ramp-error of 50 in our system is:\n');
Kp = 50 / Kv;
disp(Kp);

%Keeping Kp constant at the value calculated to maintain Kv = 50
%Kd will be determined using the rlocfind function on the root locus plot
%Defining Td as a placeholder for the derivative time constant
Td = s;

%Ploting the root locus for the system with PD control (using Kp)
%The PD controller's derivative term will be represented by Kd*s in the root locus
figure;
rlocus(Kp*G);
title('Root Locus of the System with PD Controller');
xlim([-100 4]);
ylim([-100 100]);

%Using rlocfind to pick a point on the root locus to minimize overshoot
%This will pause execution and wait for the user to select a point on the plot
[Kd, poles] = rlocfind(Kp*G);

%Displaying the selected Kd and poles of the closed-loop system
disp('The selected derivative gain Kd is:');
disp(Kd);
disp('The closed-loop poles at the selected Kd are:');
disp(poles);

%Forming the PD controller with the fixed Kp and the selected Kd
%The derivative term is now properly represented by Kd*s
G_PD = Kp + Kd*s;

%Creating the closed-loop transfer function T with the PD controller and the plant G
T = feedback(G_PD*G, 1)

%Simulating and plot the step response of the closed-loop system with the PD controller
figure;
step(T);
title(sprintf('Closed-Loop Step Response with Kp = %.3f, Kd = %.3f', Kp, Kd));
```

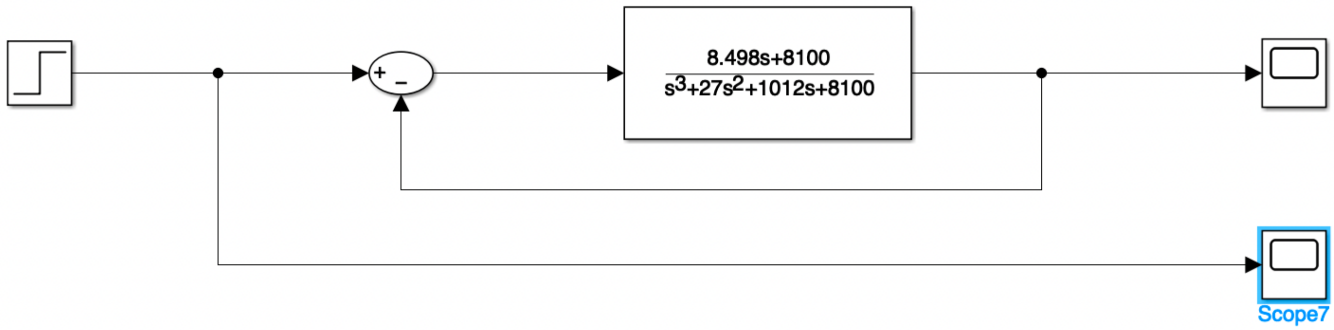
Root Locus Plot and approximate root chosen:

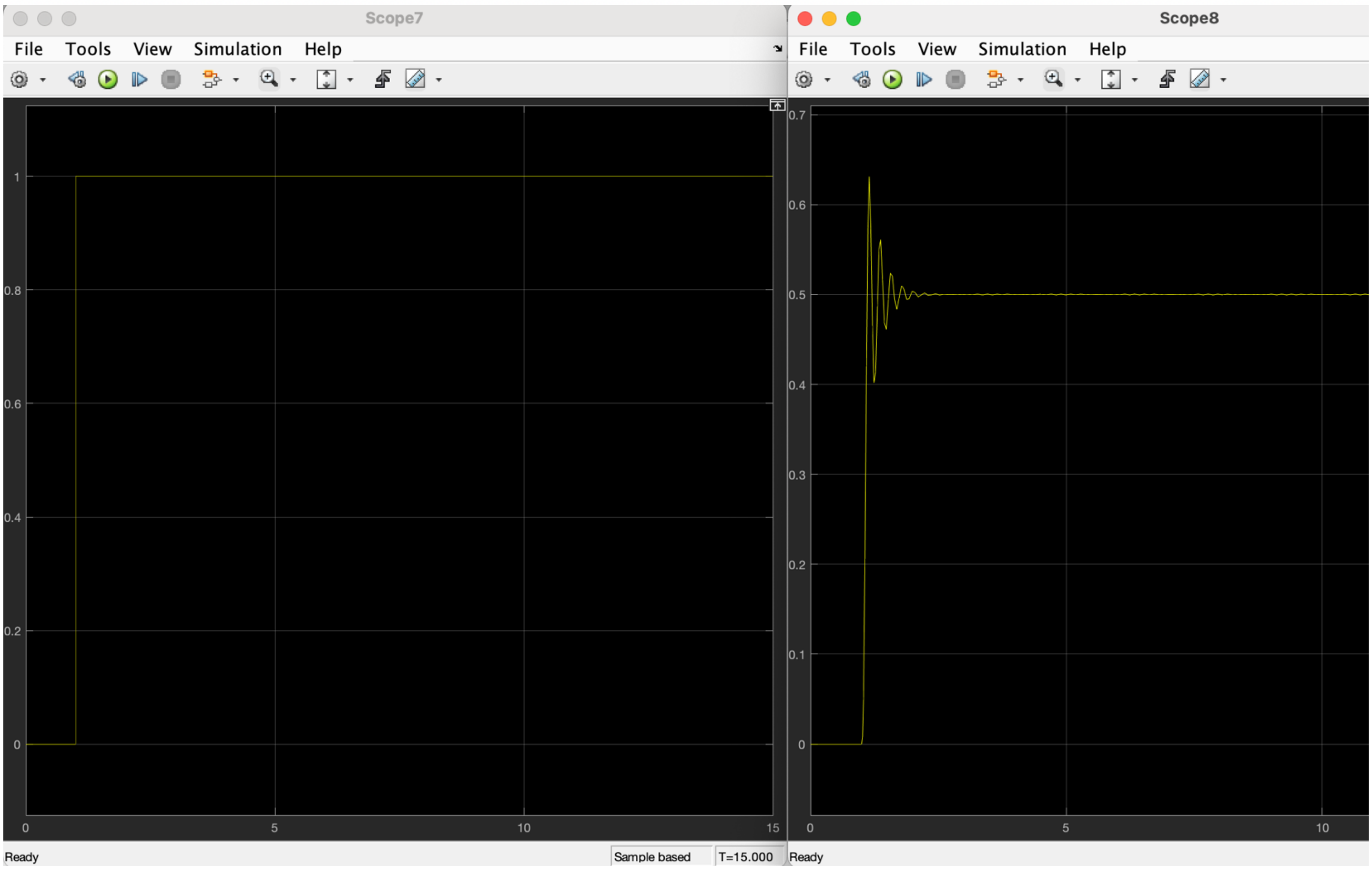


Overshoot response of the system at this given root:



Modeling the closed-loop system in Simulink:





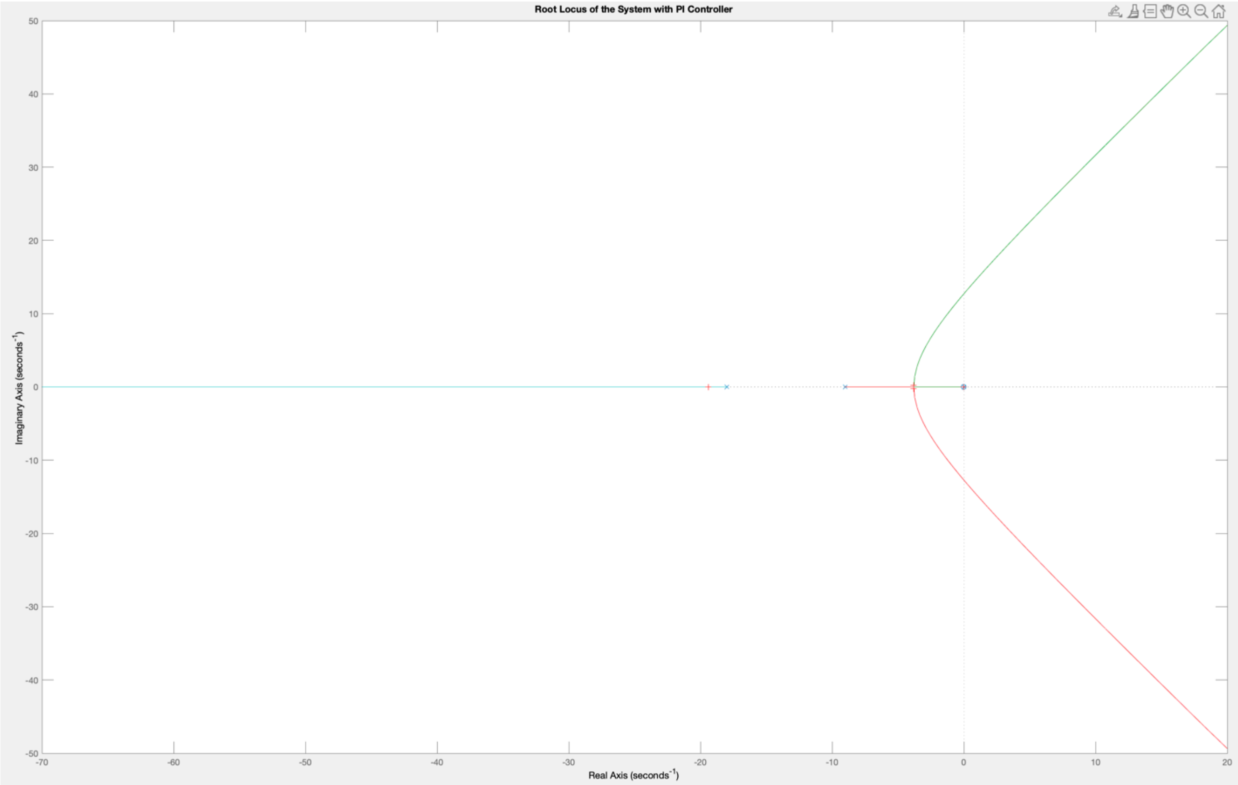
Controller 2: DI

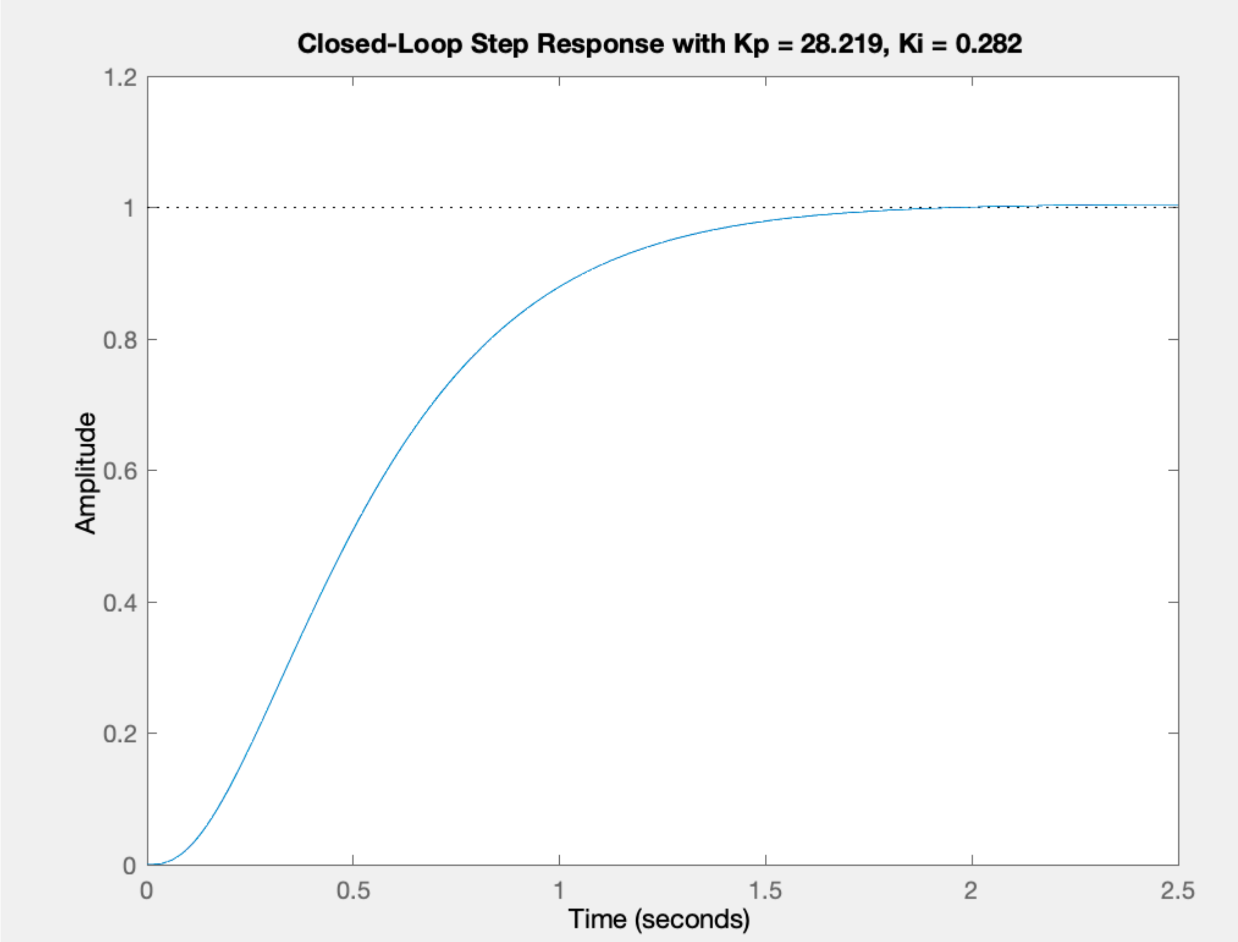
Code:

ECE4470_Project_Test1.m | ECE4470_Controller2Testing.m

```
%Defining the given system
s = tf('s');
G = 10 / (s*(s+9)*(s+18));

%PI Controller parameters (initial guesses, tweaked below)
Kp = 100; %Starting with a small value for Proportional gain
Ki = 1;   %Starting with a small value for Integral gain

%Defining the PI Controller Open Loop transfer function
PI = Kp + Ki/s;
OL_TF = PI * G;

%Generating the root locus plot of the open-loop transfer function
figure;
rlocus(OL_TF)
title('Root Locus of the System with PI Controller');

%Using rlocfind to test and select points on the root locus plot

[K, poles] = rlocfind(OL_TF);

%Updating Kp and Ki based on the selected gain K
%The gain K returned from rlocfind is used to adjust the PI controller gains
Kp = Kp * K
Ki = Ki * K

%Redefining the PI controller with the updated gains
PI = Kp + Ki/s

%Creating the closed-loop transfer function with the updated PI controller
T = feedback(PI * G, 1)

%Simulating and plot the step response of the closed-loop system
figure;
step(T);
title(sprintf('Closed-Loop Step Response with Kp = %.3f, Ki = %.3f', Kp, Ki));

%Fetching step response characteristics
info = stepinfo(T);

%Displaying the overshoot and settling time
fprintf('Overshoot: %.2f%%\n', info.Overshoot);
fprintf('SettlingTime: %.2f seconds\n', info.SettlingTime);
```

Root Loci plot for when initial guess of Kp = 100:

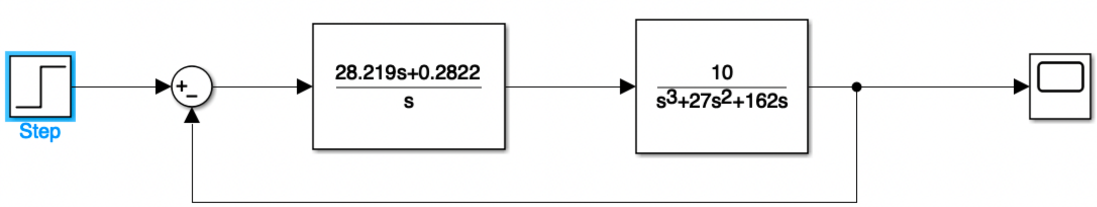




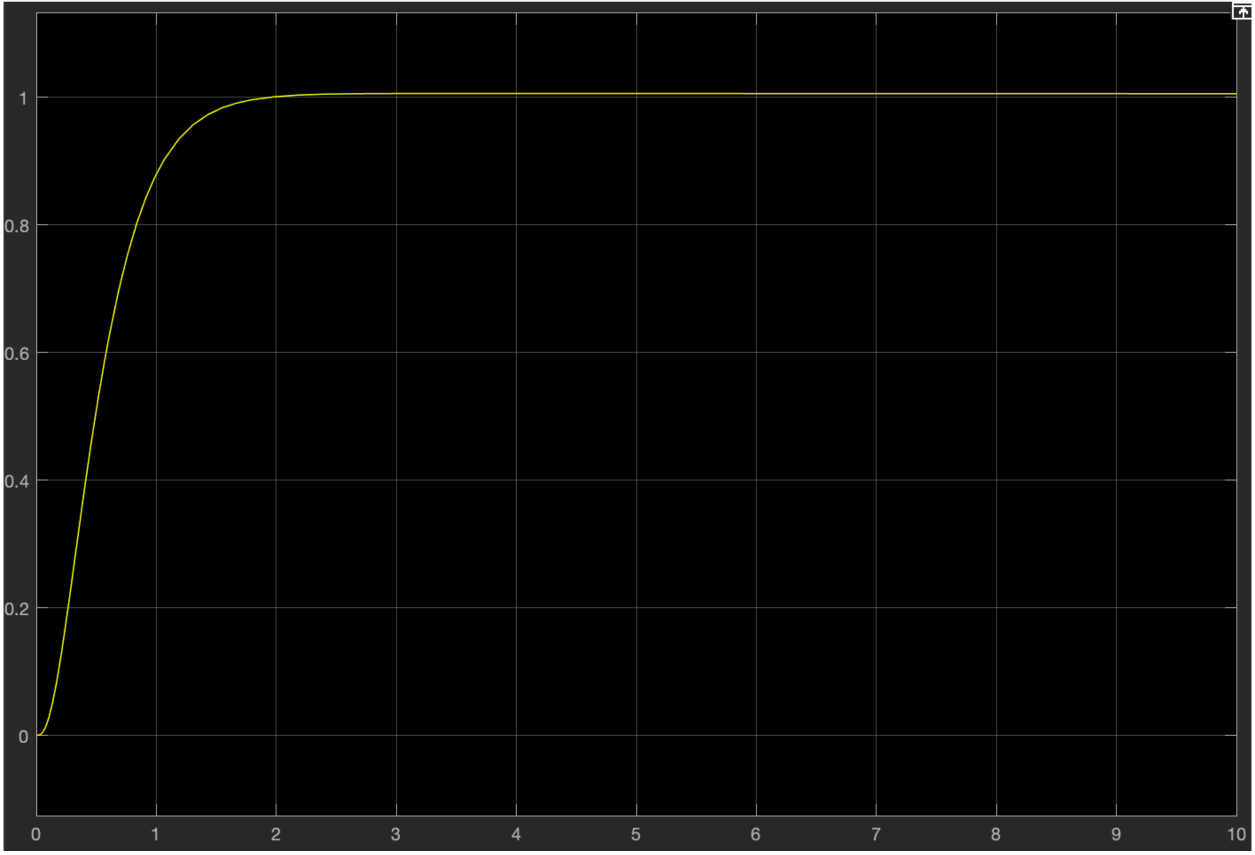
Results:

```
>> ECE4470_Controller2Testing
Select a point in the graphics window

selected_point =

  -3.8072 + 0.2349i


Kp =

   28.2190


Ki =

    0.2822


T =

               282.2 s + 2.822
  ----------------------------------------
  s^4 + 27 s^3 + 162 s^2 + 282.2 s + 2.822

Continuous-time transfer function.

Overshoot: 0.45%
SettlingTime: 1.51 seconds
```

Implementing in Simulink:
Block Diagram showing (Kp + Ki/s) * G(s)H(s):

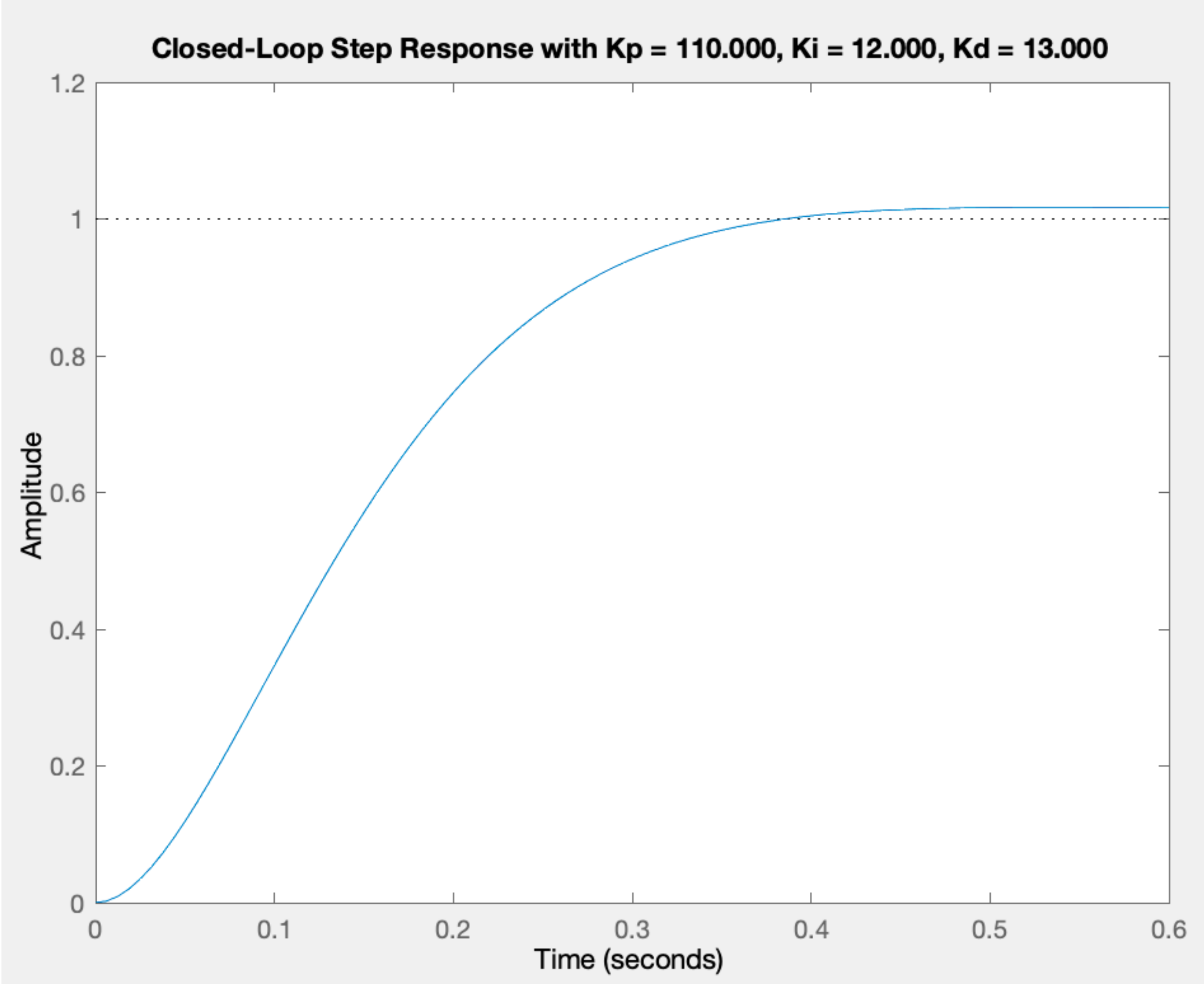


Output Oscilloscope results, matches the MATLAB step response simulation.



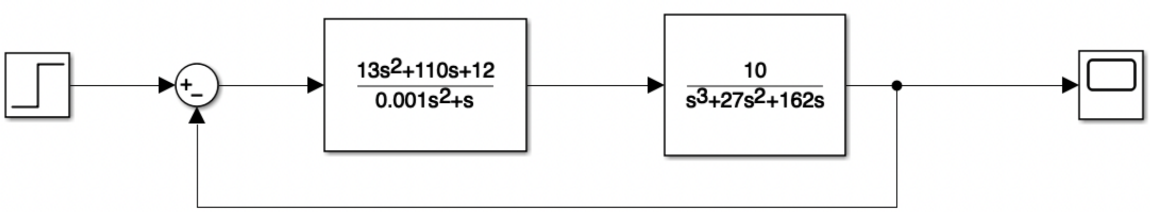
Controller 3: PID

```
%Defining the system:
s = tf('s');
G = 10 / (s*(s+9)*(s+18));

%Tuning the PID Controller Gains;
Kp = 110;
Ki = 12;
Kd = 13;

%PID Controller Transfer Function
PID = Kp + Ki/s + Kd*s

%Closed-loop Transfer Function with PID controller
T_PID = feedback(PID * G, 1)


%Simulating, plotting the step response of the closed-loop system
figure;
step(T_PID);
title(sprintf('Closed-Loop Step Response with Kp = %.3f, Ki = %.3f, Kd = %.3f', Kp, Ki, Kd));

%Step Response Simulation without plotting:
[response, t] = step(T_PID);

%step response characteristics
info = stepinfo(T_PID);

%Displaying the performance metrics:
fprintf('RiseTime: %.2f seconds\n', info.RiseTime);
fprintf('SettlingTime: %.2f seconds\n', info.SettlingTime);
fprintf('Overshoot: %.2f%%\n', info.Overshoot);
```

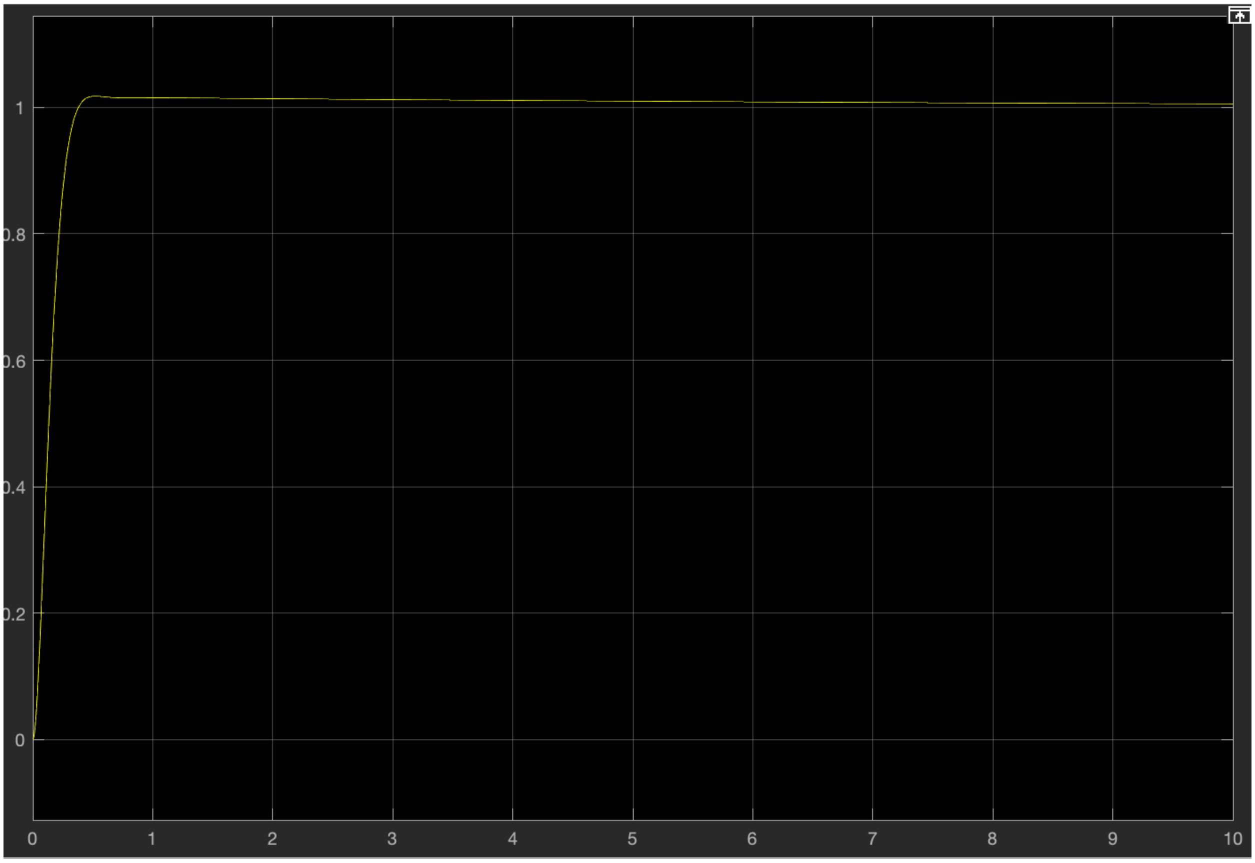



```
T_PID =

       130 s^2 + 1100 s + 120
  -------------------------------------
  s^4 + 27 s^3 + 292 s^2 + 1100 s + 120

Continuous-time transfer function.

RiseTime: 0.22 seconds
SettlingTime: 0.34 seconds
Overshoot: 1.70%
```

Simulink Modeling and Simulation Results:



Output Oscilloscope Results:

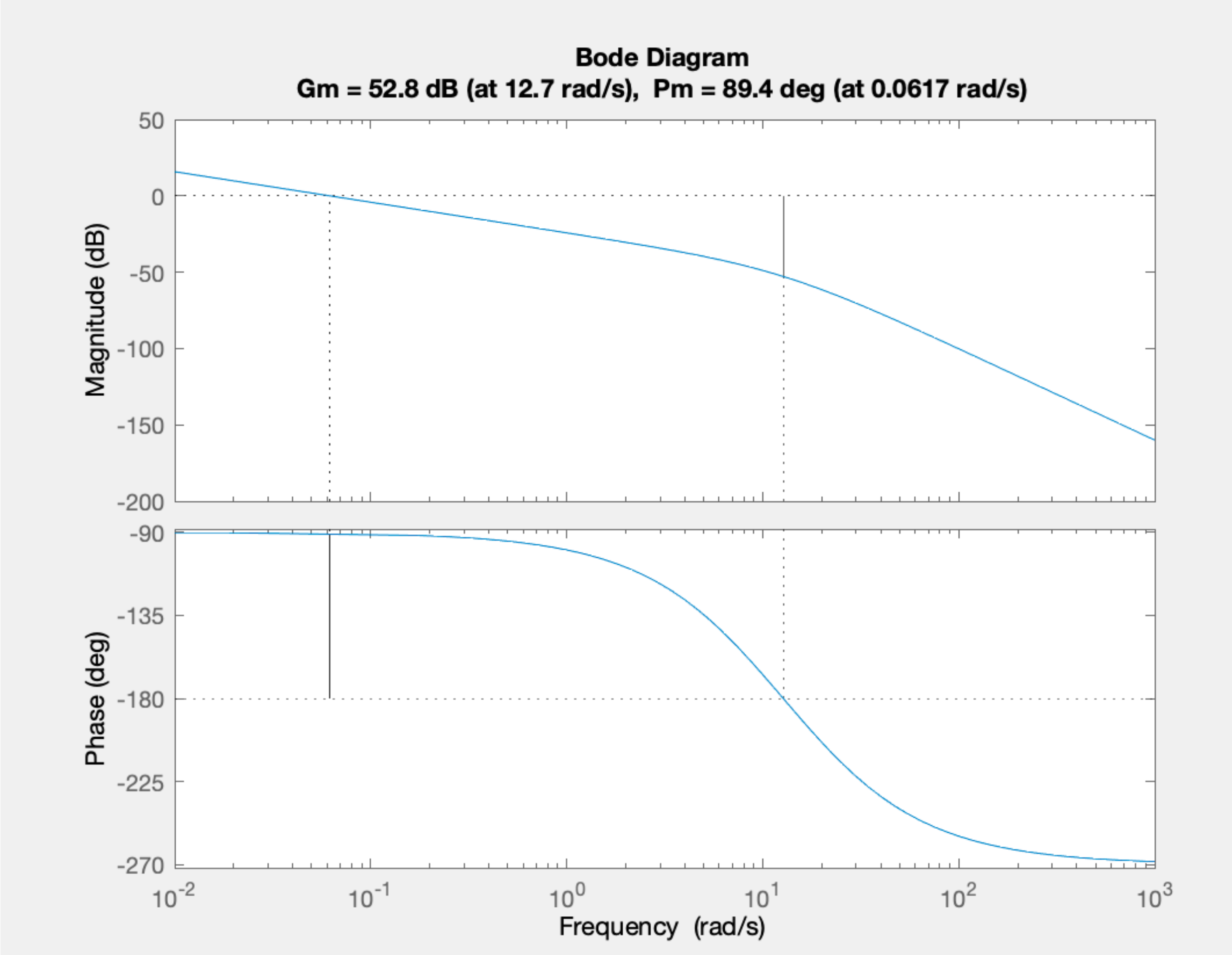


Controller 4: Phase-Lead

Code:

```
%Defining the System:
s = tf('s'); % We need to define the Lapace variable as a continous function to work with it.
num = 10;
den = [1 27 162 0]; % s(s+9)(s+18) = s^3 + 27s^2 + 162sa
G = tf(num, den)

%Bode plot to find initial Kv and phase margin of the system without
%phase-lead.

figure;
margin(G);
[Gm, Pm, Wcg, Wcp] = margin(G);


%Design parameters for the phase-lead compensator
alpha = 5.828; %Solving for a where phi_m is 45, we have a value of 5.828
T = .3573; %solved for analytically by deriving desried crossover frequency.

% Phase-lead transfer function:
Gc = (s*T + 1) / (s*alpha*T + 1);
K = 161.99; %K needed fpor Kv = 10: (Found analytically) %161.999



%Open-loop transfer function with compensator
OL_TF = K * Gc * G

% Bode plot of system with compensator
figure;
margin(OL_TF);
[Gm, Pm, Wcg, Wcp] = margin(OL_TF);


%Tweaking the parameters, these are the best I could make!
alpha = 7;
T = .858;

% Phase-lead transfer function:
Gc = (s*T + 1) / (s*alpha*T + 1);
K = 161.99; %K needed fpor Kv = 10: (Found analytically) %161.999

%Open-loop transfer function with compensator
OL_TF = K * Gc * G


% Bode plot of system with compensator
figure;
margin(OL_TF)
[Gm, Pm, Wcg, Wcp] = margin(OL_TF);
```

Kv = 10
Phase-Margin = 45 degrees.
First step is to calculate the correct K gain of compensator needed to achieve Kv = 10.
We then plot the initial Bode plot of the system with K, excluding the controller:



We see that our initial phase margin is 180-89.4 = 90.6 degrees.
We can analytically solve for a using the following formula:

$$\phi_m = \sin^{-1}\left(\frac{a-1}{a+1}\right)$$

A was analytically found to be 5.828.

Using this value of A, we can specify a value of T that will further calibrate the controller by deriving a new crossover frequency using

$$20\log|G(j\omega_m)| = -10\log a = -3.91$$

From the Bode plot, we find $\omega_m = 60$.

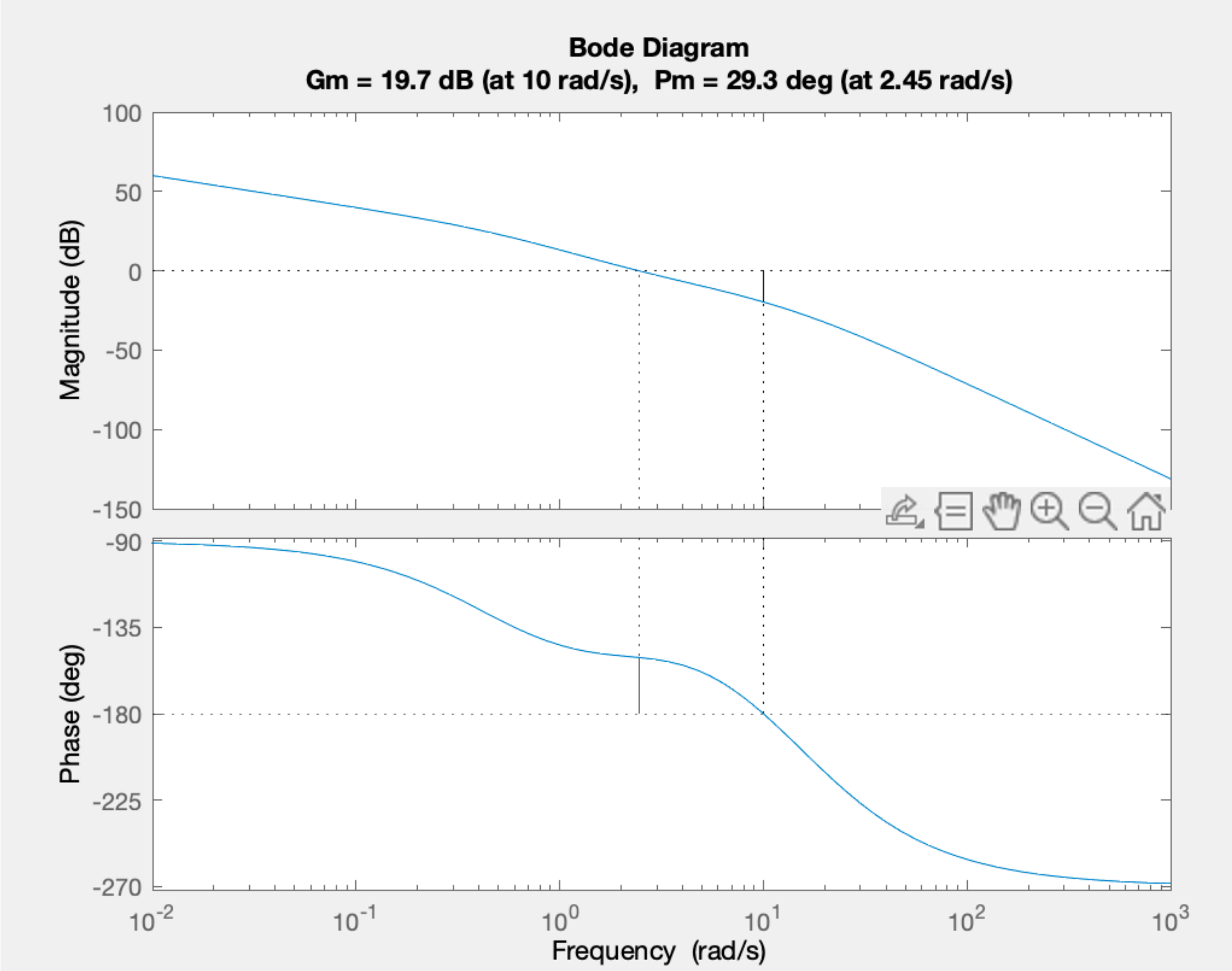
Hence,

$$\omega_m = \frac{1}{\sqrt{a}T}$$

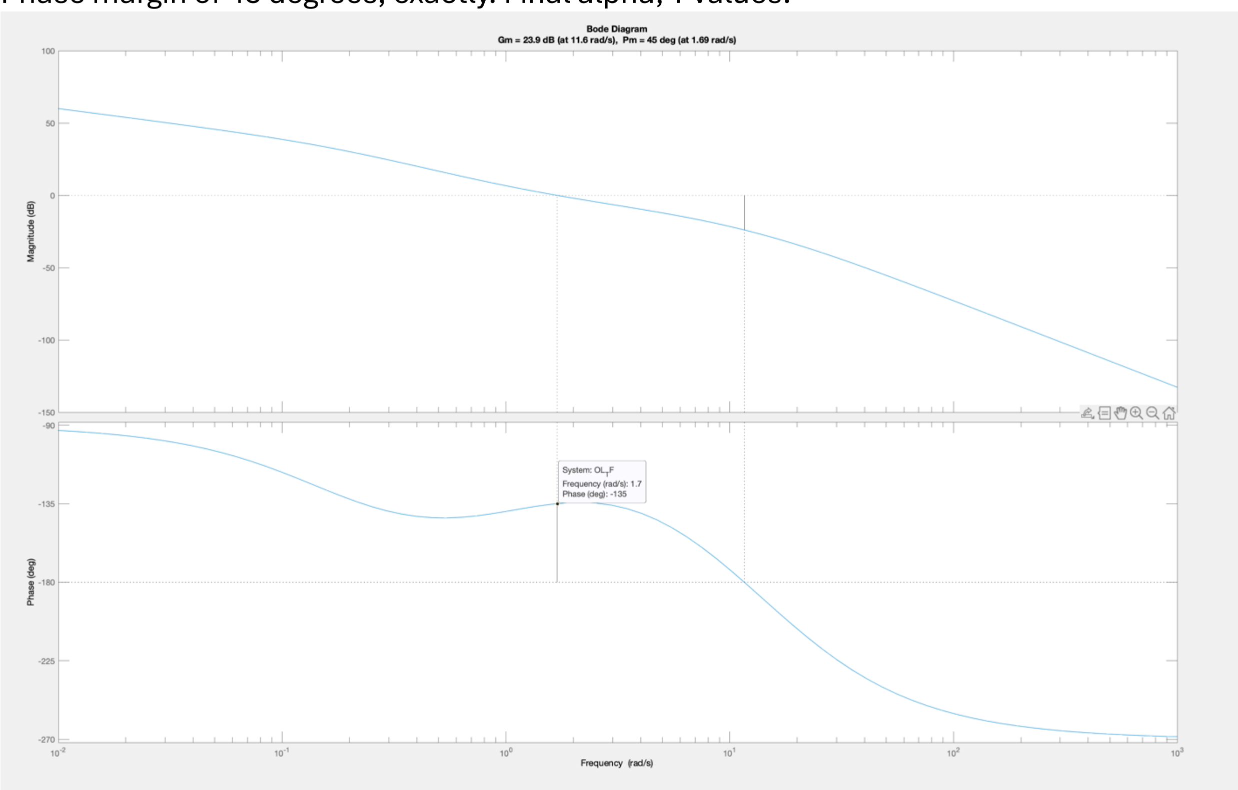
$$\Rightarrow T = \frac{1}{\sqrt{a}\omega_m} = 0.0106$$

The new T value I used was: .3573.

New Bode Plot Results:



The Phase margin is too low now, so I will tweak the values of a and T until I approximate 45 degrees. Phase margin of 45 degrees, exactly. Final alpha, T values:

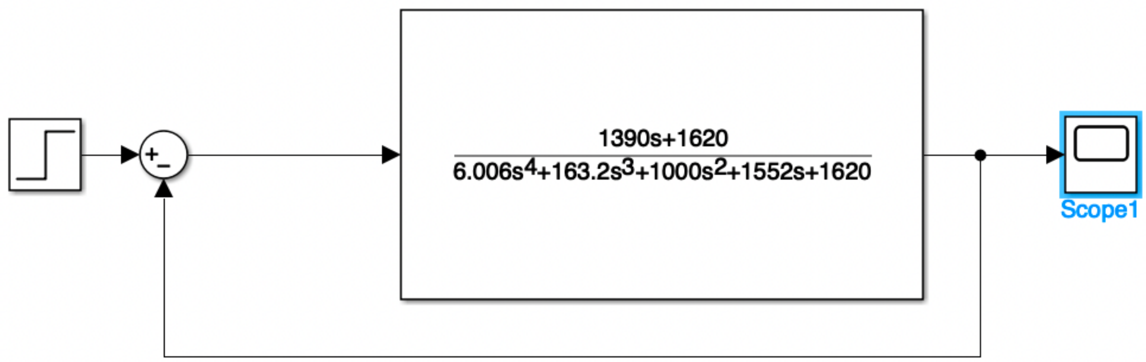



```
alpha = 7;
T = .858;
```

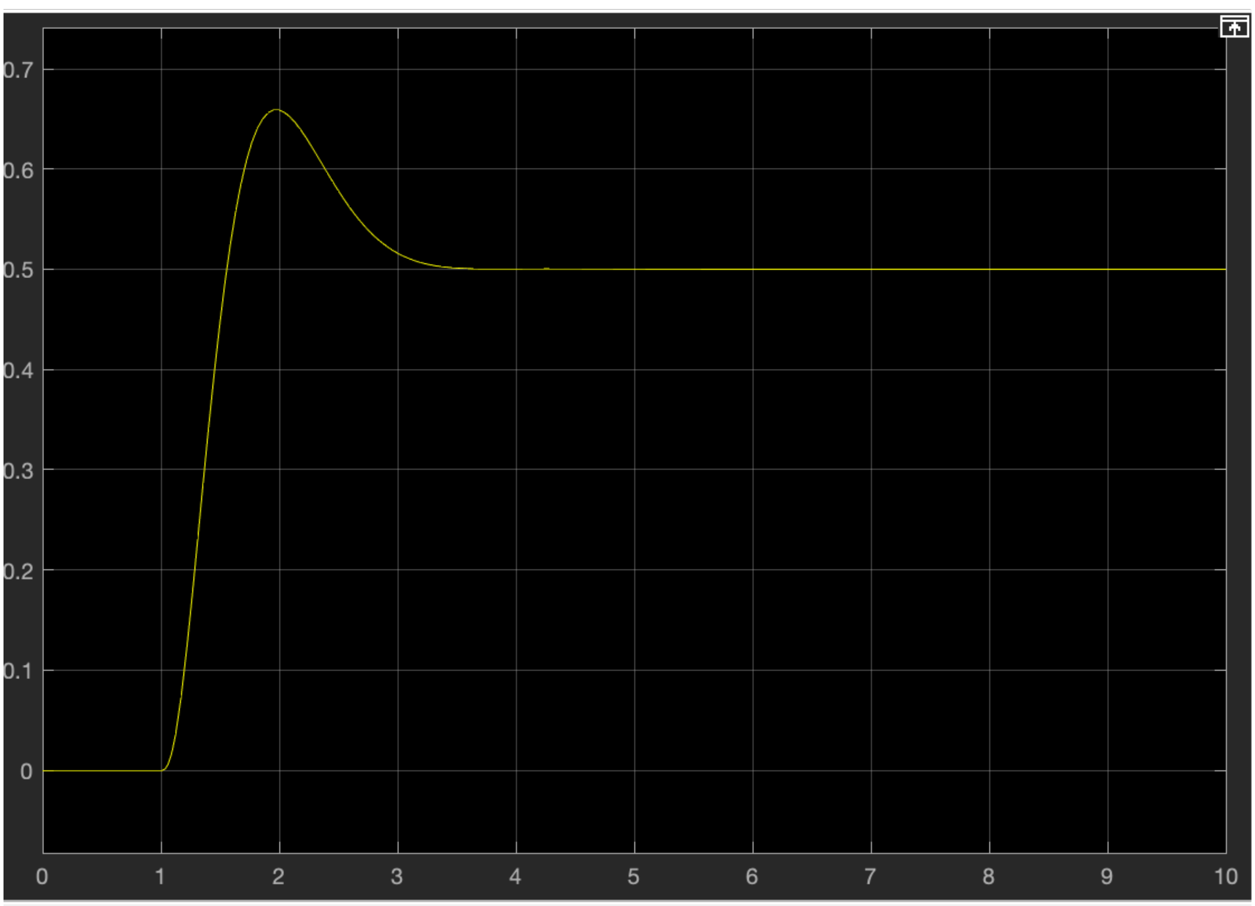
```
OL_TF =

                1390 s + 1620
  ----------------------------------------
  6.006 s^4 + 163.2 s^3 + 1000 s^2 + 162 s
```

Simulink Model and Simulation:

Block diagram of the overall system:



Step response of the system, we can see it is stable.



Controller 5: Phase-Delay

Code:

ECE4470_Controller5Testing.m

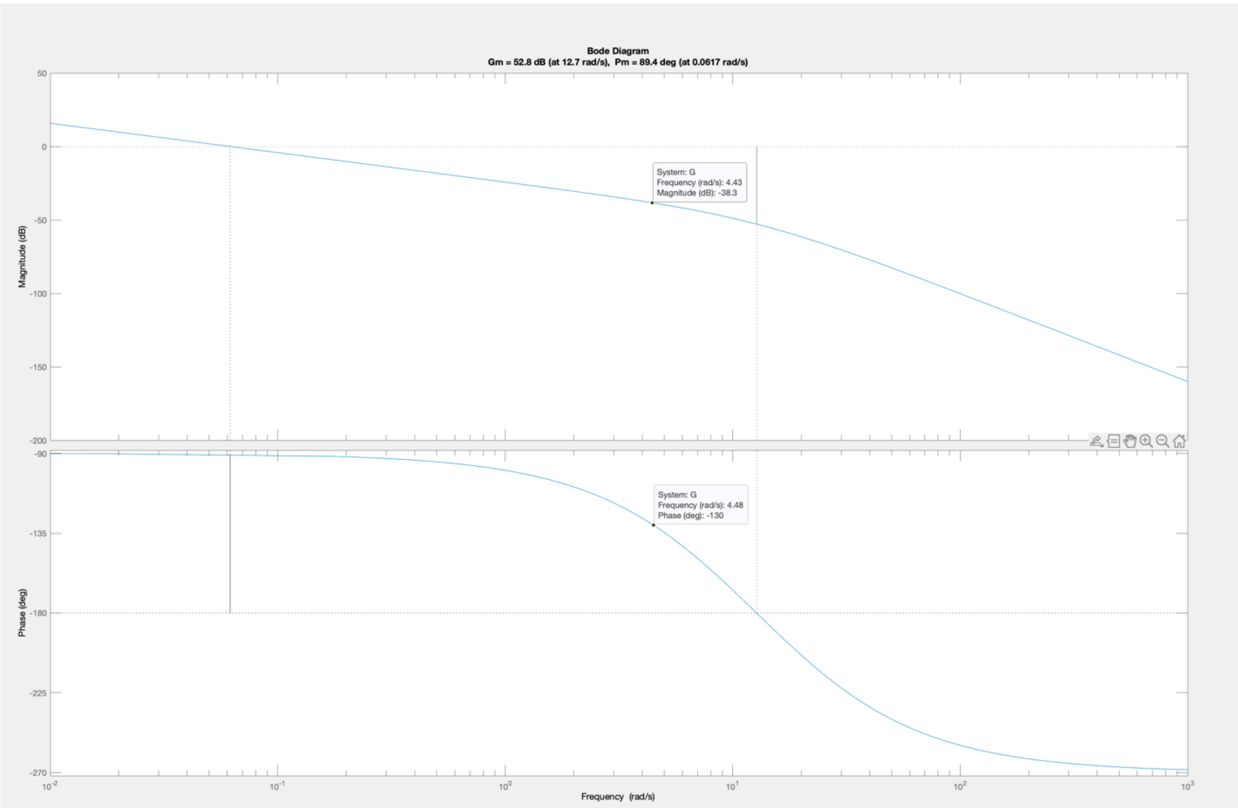
```
%%Step 5: Design a Phase-lag controller using frequency-domain design
%%method such that Kv = 10 and Phase margin is 45 degrees.


%Defining the System:
s = tf('s');
num = 10;
den = [1 27 162 0]; % s(s+9)(s+18) = s^3 + 27s^2 + 162sa
G = tf(num, den)

%Bode plot to find initial Kv and phase margin of the system without
%phase-lead, same as before.
figure;
margin(G);
[Gm, Pm, Wcg, Wcp] = margin(G);


%Design parameters for the phase-lead compensator
alpha = 79.433;
T = .0281;

%Phase-lead transfer function:
Gc = (s*T + 1) / (s*alpha*T + 1);
K = 161.99; %K needed fpor Kv = 10: (Found analytically) %161.999. Same as controller 4.

%Open-loop transfer function with compensator, untuned.
OL_TF = K * Gc * G

%Bode plot of system with phase-lag controlled:
figure;
margin(OL_TF)
[Gm, Pm, Wcg, Wcp] = margin(OL_TF);


%%Definitely needs to be tweaked!

%Design parameters for the phase-lead compensator
alpha = .01259;
T = .042;

%Phase-lead transfer function:
Gc = (s*T + 1) / (s*alpha*T + 1);
K = 161.99; %K needed fpor Kv = 10: (Found analytically) %161.999. Same as controller 4.

%Open-loop transfer function with compensator, untuned.
OL_TF = K * Gc * G

%Bode plot of system with phase-lag controlled:
figure;
margin(OL_TF)
[Gm, Pm, Wcg, Wcp] = margin(OL_TF);
```

Controller 5 is very similar to controller 4, except we go about changing the phase margin in a different way. Instead of estimating values for a and T that will add phase to the output signal, we estimate values for a and T that reduce the magnitude, and therefore change the location relative to phase where the crossover frequency occurs.

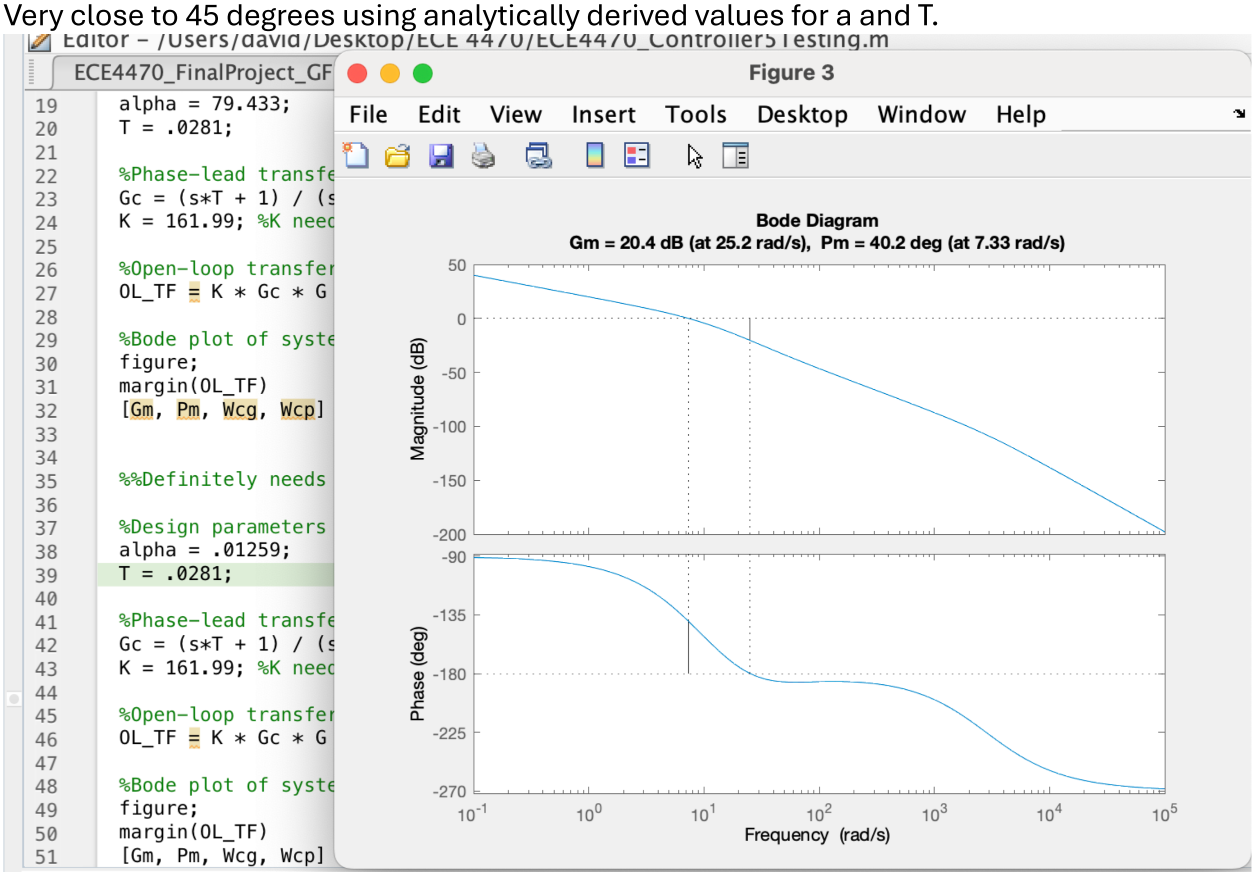
We want to change where Wc is to where our desired phase margin is. Since we want 45 degrees of margin, that is currently found at w = 4.48 in our system.

The gain at this frequency is approximately 38dB in our system.

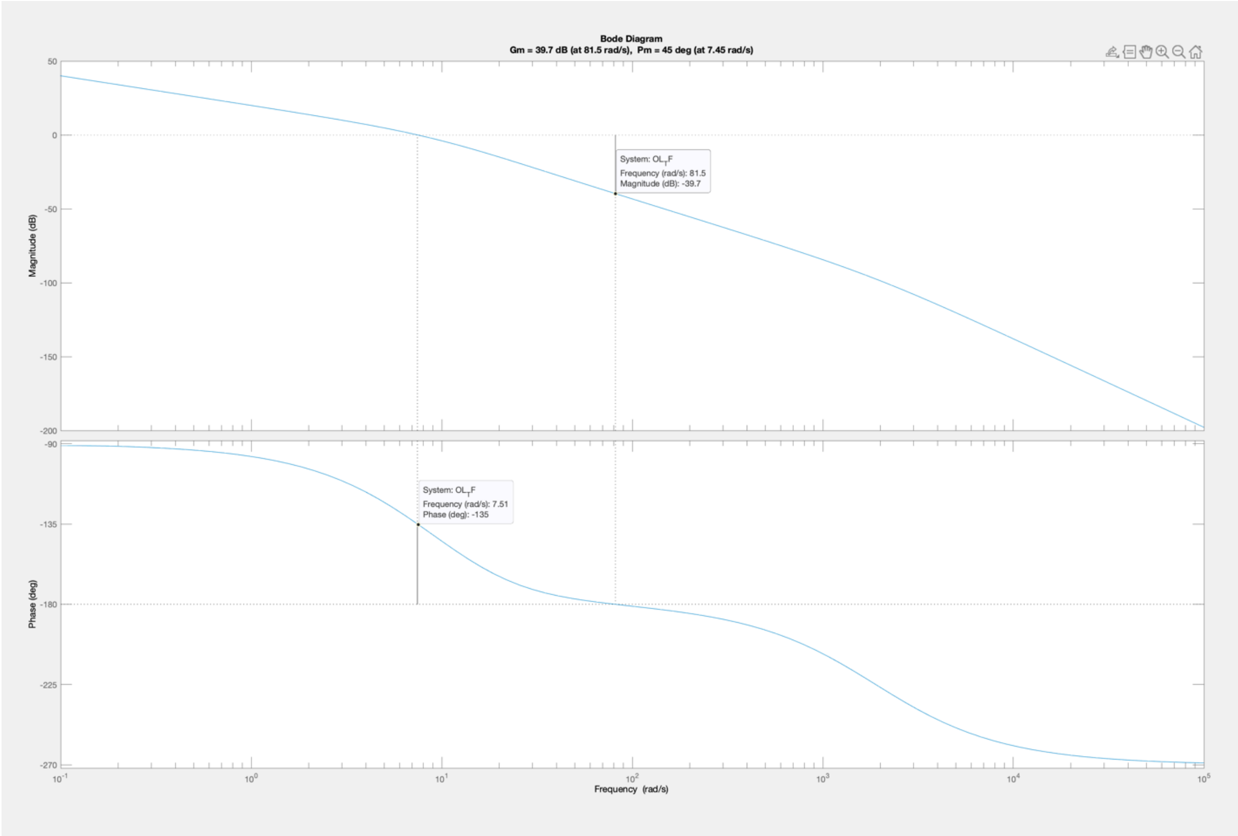
Numerical work:



Very close to 45 degrees using analytically derived values for a and T.



After tweaking the values:



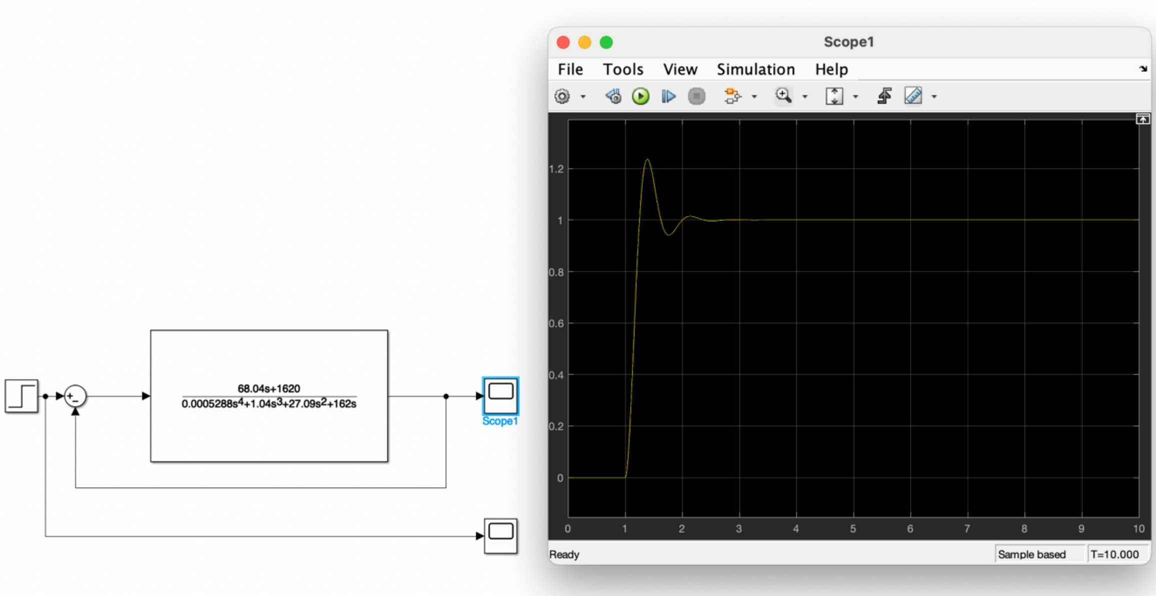
45 degree phase margin achieved.

Alpha and T values, resulting closed-loop transfer function of the system:

```
%Design parameters for the phase-lead compensator
alpha = .01259;
T = .042;
```

```
OL_TF =

                 68.04 s + 1620
  ---------------------------------------------
  0.0005288 s^4 + 1.014 s^3 + 27.09 s^2 + 162 s
```

Simulink Model and Simulation:



Comparing/Contrasting the controllers, discussing Pros/Cons, Conclusion:

The comparative analysis of PD, PI, PID, Phase-lead, and Phase-lag controllers on a given transfer function G(s)H(s) gives insights into the operational strengths and weaknesses of each controller. The PD controller, designed for minimal overshoot, shows good transient response but lacking in steady-state accuracy, making it well-suited for applications where rapid response is critical, and more than long-term precision. In contrast, the PI controller which we focused to eliminate steady-state error, showed superior long-term accuracy at the expense of transient performance. This is good to use in systems where steady-state stability is most important.

The PID controller is a more versatile solution. It balances the fast response with a minimal overshoot, making it an all-round useful design in control applications. This is achieved at the expense of increased complexity, both numerically and analytically. The values of Kp, Ki, Kd had to be tweaked, a much more trial-and-error process.

The Phase-lead controller was used to create a phase margin increase, which in this case improved system stability. This is a valuable application in systems requiring consistent phase characteristics. In contrast, the Phase-lag controller optimized the system's gain margin which enhanced the low-frequency response. This is useful in processes that require attenuation of high-frequency noise, IE, applying an LPF within the system.

Conclusion:
This project explains to us that there is no one-size-fits-all controller. The choice of controller is a chose made to balance and fine-tune specific parameters of the system required.

This was challenging, but fun! I feel that I've learned a lot.

Thank you!

Hand-Written Work:

Controller 1 :

- Type 0 system, 3rd Order.

- $G(s)H(s) = \left(\frac{10}{(s)(s+9)(s+18)}\right)\left(K_p + K_D(s)\right)$

$e_{ss} = \frac{1}{K_v}$

- Input : $R(s) = 1/s^2$

$e_{ss} = \lim_{s \to 0} \frac{(s)(1/s^2)}{1 + \frac{10(K_p + K_D(s))}{(s)(s+9)(s+18)}} = \lim_{s \to 0}$

- $K_v = \lim_{s \to 0} (s)\,G(s)H(s) \rightarrow (\cancel{s})\left(\frac{10}{(\cancel{s})(s+9)(s+18)}\right)\left(K_p + K_D(s)\right)$

$\lim_{s \to 0} \frac{(10)(K_p + K_d(0))}{(0+9)(0+18)} = \frac{10K_p}{162} = K_v$

$K_v = 50 = \frac{10}{162}K_p$, $\boxed{K_p = 810}$

- with K_p and $K_D = \phi$, this system becomes unstable.

CL - Characteristic Eq : $(1 + G(s)H(s)) = 0$

$\longrightarrow \quad 1 + \frac{10(K_p + K_d(s))}{(s)(s+9)(s+18)} = 0 \rightarrow (s)(s+9)(s+18) + 10K_p + 10K_d s =$

$+10(K_p + K_d s)$ 8100

$\longrightarrow \quad s^3 + 27s^2 + 162s + 10K_d s + 10K_p = 0$

$1 + (10)(K_d)\left[\frac{(s)}{s^3 + 27s^2 + 162(s) + 8100}\right] = 0$

$= 1 + K\frac{(s)}{(s+30.426)(s-1.713-16.226i)(s-1.713+16.226i)}$

$\left[K = 10K_d\right]$ $138 = 10K_d$

$K_d = 18.8...$

MIQUELRIUS

$$s^3 + 27s^2 + 162s + Ks + 8100 = 0$$

$$\begin{array}{lll} s^3 & 1 & 162 + k \\ s^2 & 27 & 8100 \\ s^1 & & \\ s^1 & & \end{array}$$

$$x_{3,1} = \left(\frac{1}{27}\right)\begin{bmatrix} 27 & 8100 \\ 1 & 162+k \end{bmatrix}$$

$$= \frac{1}{27}\left(27(162+k) - 8100\right)$$

$$= 162 + k - 300 > 0$$

$$\left[k \geq 138\right]$$

$$k = 138 \,,\quad k = 10\,k\delta \,,$$

$$k = \frac{l_1 l_2 l_4}{l_3}$$

$$k\delta \geq 13.8$$

$$\text{if } K = .005 : \quad \left\{ CL \quad T(s) = \frac{0.4708s + 381.3}{s^3 + 27s^2 + 162.5s + 381.3} \right]$$

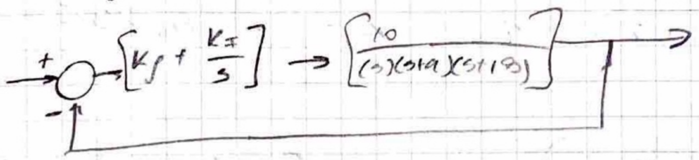
Final

$\left(10k_p + 10\frac{k_i}{s}\right) = 10k_p\left(1 + \frac{k_i}{k_p}(s)\right)$

$= 10k_p(1 + .01s)$

Controller 2: PI , $\zeta = .707$.

- $\zeta = .707$



- $\frac{k_I}{k_p} = (.01) \longrightarrow \quad G(s)H(s) = (10)k_p(s + .01)$

$$\frac{(10)\left(k_p + \frac{k_I}{s}\right)}{(s)(s+9)(s+18)}\frac{(s)}{(s)} = \frac{(10)(k_p(s) + k_i)}{(s)^2(s+9)(s+18)} \longrightarrow$$

$$\longrightarrow \left[\frac{k_I}{k_p}\right] = .1 \rightarrow \frac{10k_p(s + .01)}{(s^2)(s+9)(s+18)} \approx \frac{10k_p}{(s)(s+9)(s+18)}$$

CL $T(\zeta)$:
Char Eq:

$$1 + \frac{10k_p}{(s)(s+9)(s+18)} \longrightarrow 1 + [k]\frac{1}{(s)(s+9)(s+18)} = \phi$$

$\downarrow$

$[k = 10k_p]$

- $k = 0$, then poles @ $0, -9, -18$

$\longrightarrow s^3 + 27s^2 + 162s + k = \phi$

s^3	1	162
s^2	27	k
s^1		
s^0		

$$x_{3,1} = \left(\frac{1}{27}\right)\begin{bmatrix} 27 & k \\ 1 & 162 \end{bmatrix} = \frac{1}{27}\left((27)(162) - k\right)$$

- stability: $162 - \frac{k}{27} > 0 \;\land\; k > 0$

$\downarrow$ $0 < k < 4374$?

$k > \frac{-162}{-27}$

- O.S. $\approx e^{-\zeta\pi/\sqrt{1-\zeta^2}}$ [if $\zeta = .707$, O.S. $\approx .04325$ or $\approx 4.3\%$]

$$\left[K_p = 28.219 \quad , \quad K_i = 0.2822 \right]$$ Values used for Simulation !

$$\left(K_p + K_i/s\right) \longrightarrow \frac{28.219(s) + .2822}{(s)}$$

$$\left(K_p + K_i/s\right)\left(\frac{10}{(s)(s+9)(s+18)}\right) = \frac{28.219(s) + 2.822}{s^3}$$

$$\downarrow$$

$$(s^3 + 27s + 162(s))(s) = \ldots$$

Fast response, small overshoot, no e_{ss} for $R(s) = 1/s^2$

#3 PID :

$$K_p + \frac{K_I}{s} + K_D(s)$$

$$\circ = (1 + K_{D_1}(s))\left(K_{p2} + \frac{K_{I2}}{s}\right)$$

$$G(s)H(s) = \frac{(10)(1 + K_{D_1}(s))(K_{I2} + K_{p2}(s))}{(s)^2\ (s+9)(s+18)}$$ $\left[110, 12, 13 \right]$

PI : Let $K_{D2} = \phi.$, $\frac{K_{I2}}{K_{p2}} = .1$

$$\longrightarrow G(s)H(s) \approx (10)(1)(.1 + (s)) - \ldots$$

$$\approx \frac{10 K_p}{(s)(s+9)(s+18)}$$

$\circ$ CL Char eq : $1 + [K] \frac{1}{(s)(s+9)(s+18)}$, $K = 10 K_{p2}$

30, 5, 5 is the best I've got ...

Controller #4.) Phase-Lead $[K > 0,\ T > 0,\ \alpha > 1]$

- $K_v = 10$; - $\Phi M = 45°$.

$$K\left(\frac{1+\alpha T_s}{1+Ts}\right) \longrightarrow \left[\frac{10}{(s)(s+9)(s+18)}\right]$$

- $K_V = \lim_{s\to 0} (s)\,G(s)\,H(s) = \lim_{s\to 0} \cancel{(s)}(K)\left(\frac{1+\alpha T(s)}{1+T(s)}\right)\left(\frac{10}{\cancel{(s)}(s+9)(s+18)}\right)$

$$= \frac{(K)(1)(10)}{(1)(9)(18)} = (K)\frac{5}{81}.$$

- If $K_v = 10$, $10 = (K)(5/81)$, $\boxed{K = 161.99\overline{9}.}$

$\downarrow$

$45° =$ Well damped, good transient response.

0dB @ $\omega \cdot G(\omega) = 10$

When drawing Bode of $K = 161.99\overline{9}$,

$\omega_{co} = 12.7$ rad/sec, $\Phi M = 89.4°$ @ $.0617$ rad/s

$\phi M = 180° - 89.4 = 90.6°$

For $\phi M = 45°$, increase: $\cancel{45 - 90.6° = -45.6}$??

$$\phi_M = \sin^{-1}\left(\frac{a-1}{a+1}\right)$$

phase at $G_{\omega_c} = -135°$

$$a = \frac{1+\sin\Phi_m}{1-\sin\phi_m}$$

$180 - 135 = 45°$

- Low T, increase a?

$$a = \frac{1+\sin(45)}{1-\sin(45)} \rightarrow a \approx 5.828!$$

Select Cross Over frequency ω_m

$$20\log|G(j\omega_m)| = -10\log(a)$$
$$= -7.655$$

o looking at Bode plot, we see a Gain of dB $= -7.61$ pertains to $\omega = 0.148$ rad/sec

Solve now for T: $T = \dfrac{1}{(\sqrt{a})(\omega)} = \boxed{.3543}$

Controller # 5 PL Controller.

$K_v = 10, \quad \phi_M = 45°$

→ This yields the same O.L. function as Controller # 4.

∘ K necessary for $K_v = 10 = 161.\overline{999}$

→ Achieve $\Delta\phi_m$ by reducing $|G(j\omega)|$

• ω_c' $\angle G(j\omega_c') = -180° + \phi_M + (5°\sim)$

$= -180° + 45° + (5° \sim 10°$

∘ $\omega_c' = \cancel{22}$ 4.48 ⟶ $\angle G(j\omega_c') = -130°$

$20\log|G(j\omega_c')| = -20\log a$

if $\omega_c' = 20, \quad 20\log|G(j20)| = ?$ → Lets check in Matlab.

$-38\,dB = -20\log(a)$ → .01259 ?

$a = 10^{+(30/20)} = \underline{79.433}$. That's big!

$a = 79.433$

$\frac{1}{aT} = \frac{\omega_c'}{10} \longrightarrow \frac{(T)\omega_c'}{10} = \frac{1}{a}, \quad T = \frac{1}{a}\frac{10}{\omega_c'}$

$T = \frac{10}{(79.433)(4.48)} = \underline{.0281}$ //

1.77 ?