EE 4360 Final Project

Fall 2020, 03 NOV 2020

Roberto Colon

A04973205

Section I. Description

A Proportional-Integral-Derivative (PID) controller is a type of control mechanism used in a control loop feedback system to control a wide range of processes. This type of control system works on a feedback loop to keep the output of a process as close to the desired output as possible. These control systems are the most accurate and stable type and are used to control processes that include regulation of temperature, pressure, speed, acceleration, flow rate, as well as many others. Examples of systems employing PID controllers are elevators, cruise control systems, flight control surfaces, and pretty much any system that benefits from more stable control through a feedback loop. A PID controller is not applicable in open loop systems, or systems that do not require any type of feedback. A simple example is a toaster oven which does not collect feedback data to control the output or a simple blender.

The design process used here is as follows:

1. Plant Analysis and pole verification

    Start with analyzing parameters such as Zeta, %OS, and dominant poles then mathematically calculating the design targets which are to be met.

2. Verify Plant Poles:

    Find values of circuitry to represent in SPICE with a simple circuit and run simulations to compare with MATLAB bode plot analysis of plant to verify attributes such as cutoff frequency match.

3. Verify the Uncompensated Step Response

    Use both MATLAB and spice to run step response and ensure they both match. This confirms that the simulation was built correctly and that the system can be realized physically. We check parameters such as peak time, settling time, overshoot, and steady state error to compare.

4. Root Locus Analysis (Uncompensated)

    Check target values of zeta for %OS, tag dominant poles for analysis and comparison for next design phase (PI), which should modify the steady state error leaving the rest of the plant virtually unchanged.

5. Design a PI to minimize/eliminate Steady State Error

    PI involves adding a pole at the origin as well as a zero which is close enough to cancel with the added pole. This has the desired effect of minimizing steady state error without changing other aspects of the plant.

    Add a PI to the plant and simulate. The desired goal is to eliminate as much steady state error as possible while leaving the rest of the plant unchanged. We confirm by analyzing root locus to endure that our dominant poles, gain factor (K), and zeta are unchanged.

    Run step response to check that steady state error has been driven to nearly zero. Compare results pre- and post-PI for both root locus and step response in MATLAB

6. PD Design

    PD design involves analysis of a time parameter to improve and then using equations to find a new dominant pole. We then use root locus design techniques so that our root locus runs through our new calculated pole which has our improved parameters built in. We ‘bend the spoon’.

    We verify the PID with MATLAB using root locus to ensure that our calculated pole is on the root locus. In my case, I was given a task of improving peak time. Using the equation for Tp, I was able to

get an Imaginary (vertical) value for my new pole location. Using that value along with an already know value of zeta, which gives me an angle to work with; I was able to calculate the location of a pole which would give me the desired performance. Since the pole was not on the root locus to start, I used the design technique of calculating angles and then adding a zero which would give me stability and put the pole on the root locus. The design criteria for this angle is that the sum of the angles must be an odd integer multiple of 180 degrees.

We then verify the pole location is on the root locus in MATLAB, run steady state in MATLAB to verify the performance parameter has been met, and only then move on to attempting to verify in SPICE.

To verify in SPICE, gain values must be derived from the new transfer function which includes the PI and PD controllers. Using these gains, we realize the physical system through OP Amps, resistors, and capacitors and then simulate to compare results with MATLAB. If designed correctly, the system will have similar, but not exact, values for the parameters tested. These parameters are Ts, Tp, %OS, and steady state error.

7. PID Tuning

Once everything has been successfully built, tested, and compared so that we have met the initial performance goals, we turn to improvement. Improvement is done by one of several methods which are outlined in section V.2. I chose to improve the settling time of my system since I was unimpressed by the time the system took to get to zero steady state error. This process involved using some calculation and simulation to ensure that the parameter chosen is improved without completely changing the performance of the system. Since other parameters were given, my goal was to try to leave those parameters unchanged or slightly improved.

Section II. The Plant

II.1 Plant Transfer Function and Design Targets

-Poles:

Pole 1	Pole 2	Pole 3
6.283	100.531	125.664

Table 1: Given Pole Locations for plant.

-G(s) of given Plant is:

$$G(s) = \frac{79{,}374}{(s+6.283)(s+100.531)(s+125.664)} \qquad \text{EQ (1)}$$

-%OS Target is 10%.

-Tp% improvement target is 20% improvement of original Tp.

II.2 Verification of Plant Poles in SPICE

-Calculation of R & C values for the circuitry representing the Plant:

For Pole 1:

Using the equation:

$$\omega = \frac{1}{RC} \qquad \text{EQ (2)}$$

We can let R = 100kΩ and calculate the value of C for our circuit for all three poles. This means that the following algebraic manipulation of equation 2 gives us our value of C.:

$$C = \frac{1}{\omega R} \qquad \text{EQ (3)}$$

Using equations 2 & 3, we can solve for R & C for all three poles given the boundary conditions that:

$1k\Omega \leq R \leq 100k\Omega$ and $1nF \leq C \leq 100\mu F$

This leads to the table below for all three given pole locations listed in Table 1.

ω_c	Resistance	Capacitance (solved using EQ3)
6.283	100kΩ	1.592µF
100.531	100kΩ	99.472nF
125.664	100kΩ	79.577nF

Table 2: Solved Capacitance values for the circuitry representing our given Plant.

Once these calculations were completed, the final calculation involved finding the cutoff frequency mathematically so that we could confirm it in the Bode Plot in SPICE. The equation for finding a frequency given omega in radians per second is:

$$f = \frac{\omega_c}{2\pi} \qquad \text{EQ (4)}$$

Solving Equation 4 gives us the cutoff frequency of each pole in Hz. These frequencies are given in Table 3 below:

Table 3: Cutoff Frequencies		
Pole 1 ω_c	Pole 2 ω_c	Pole 3 ω_c
1 Hz	16 Hz	20 Hz

Table 3: Calculated cutoff frequencies of given poles from lowest to highest frequency as labeled in Table 1.

-SPICE Schematic

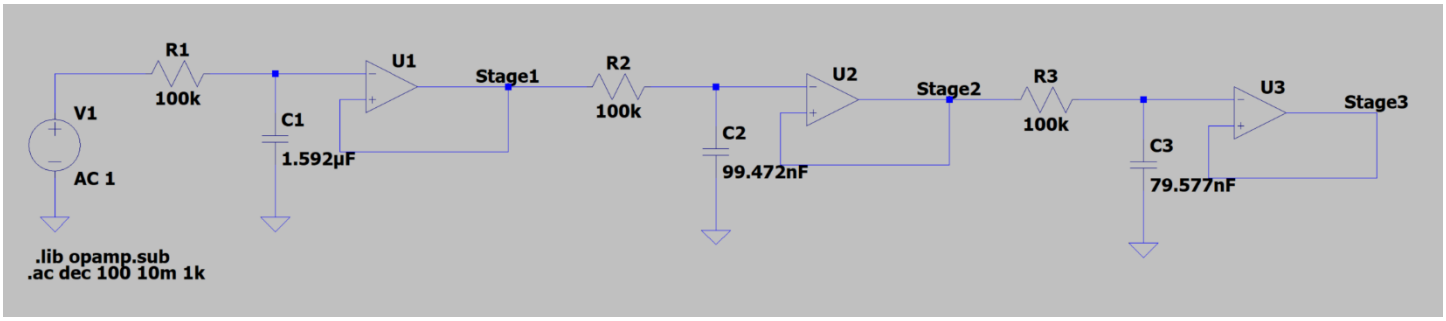


Fig. 1: Schematic of the plant circuit with calculated values for R and C. The extra code is required to run the OpAmps in the schematic. This is the schematic used to run the cutoff frequency checks below, but the stages were separated in order to alleviate cascading effects and ensure the components were matched properly. Once the frequencies were confirmed, the stages were linked for the combined output.

-Verification of Frequencies in SPICE

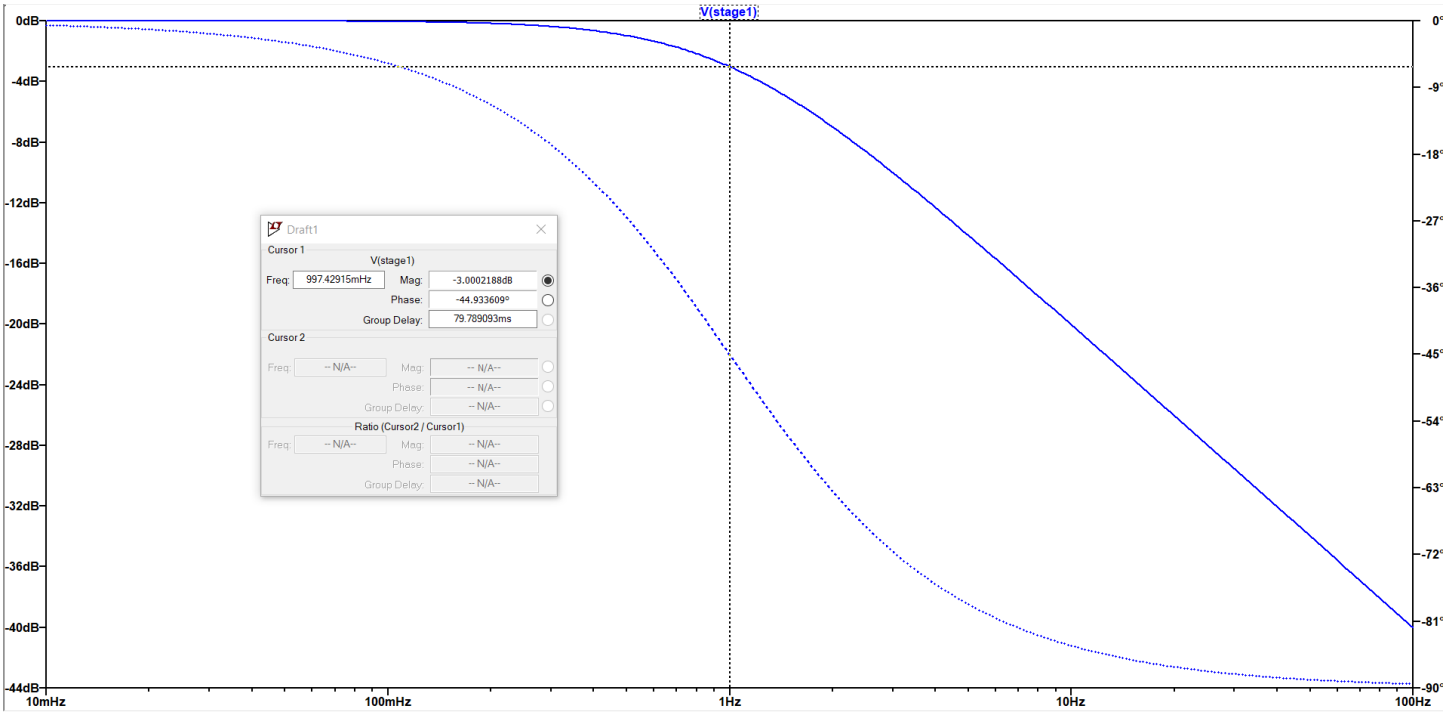


*Fig. 2: The Bode plot of **Stage 1** with a callout of the cutoff frequency calculated to be 1Hz. The -3dB point is as calculated. The magnitude is on the left axis in units of dB and the phase is on the right in units of degrees.*

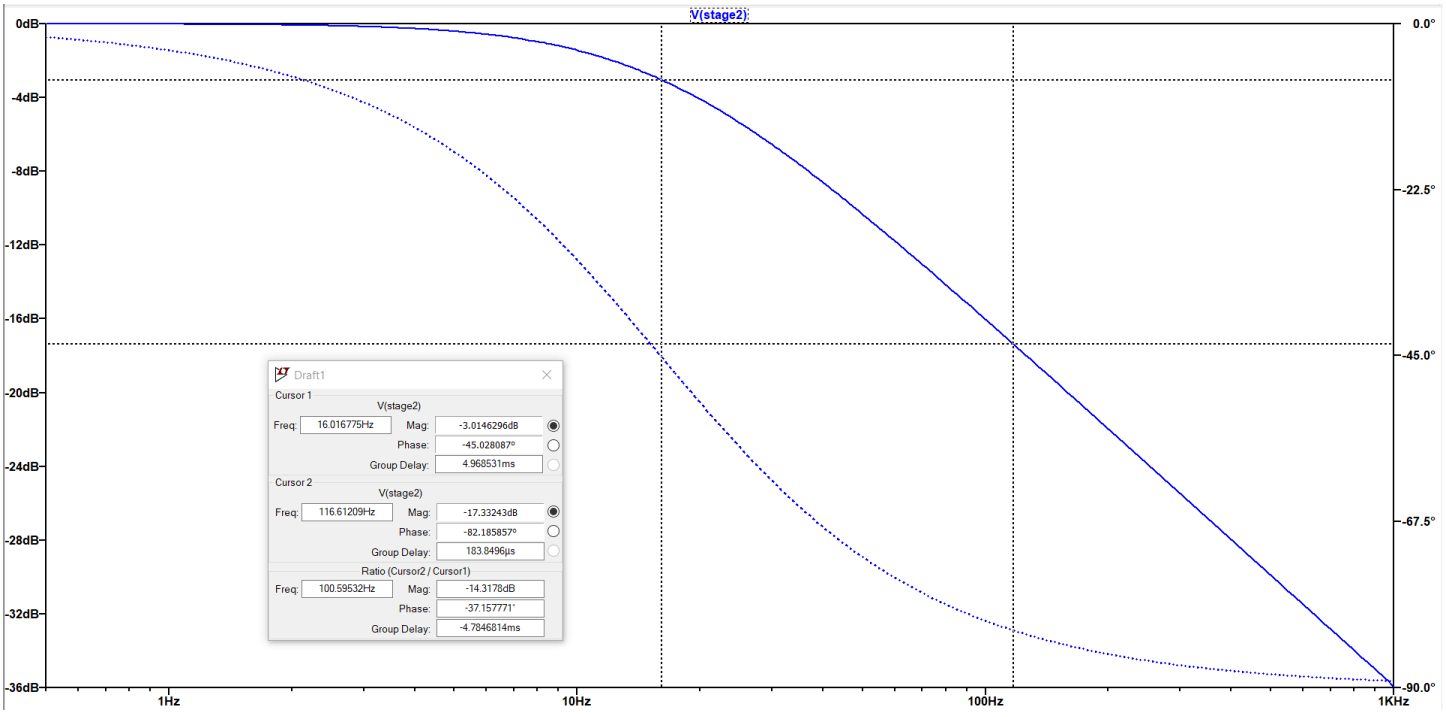


*Fig. 3: The Bode plot of **Stage 2** with callouts to mark the cutoff frequency as well as one decade further to calculate behavior of the slope when ran alone as a single unit separate from the rest of the circuit. We can see callout one (left, at approximately -3dB) has the calculated frequency of 16Hz from Table 3.*

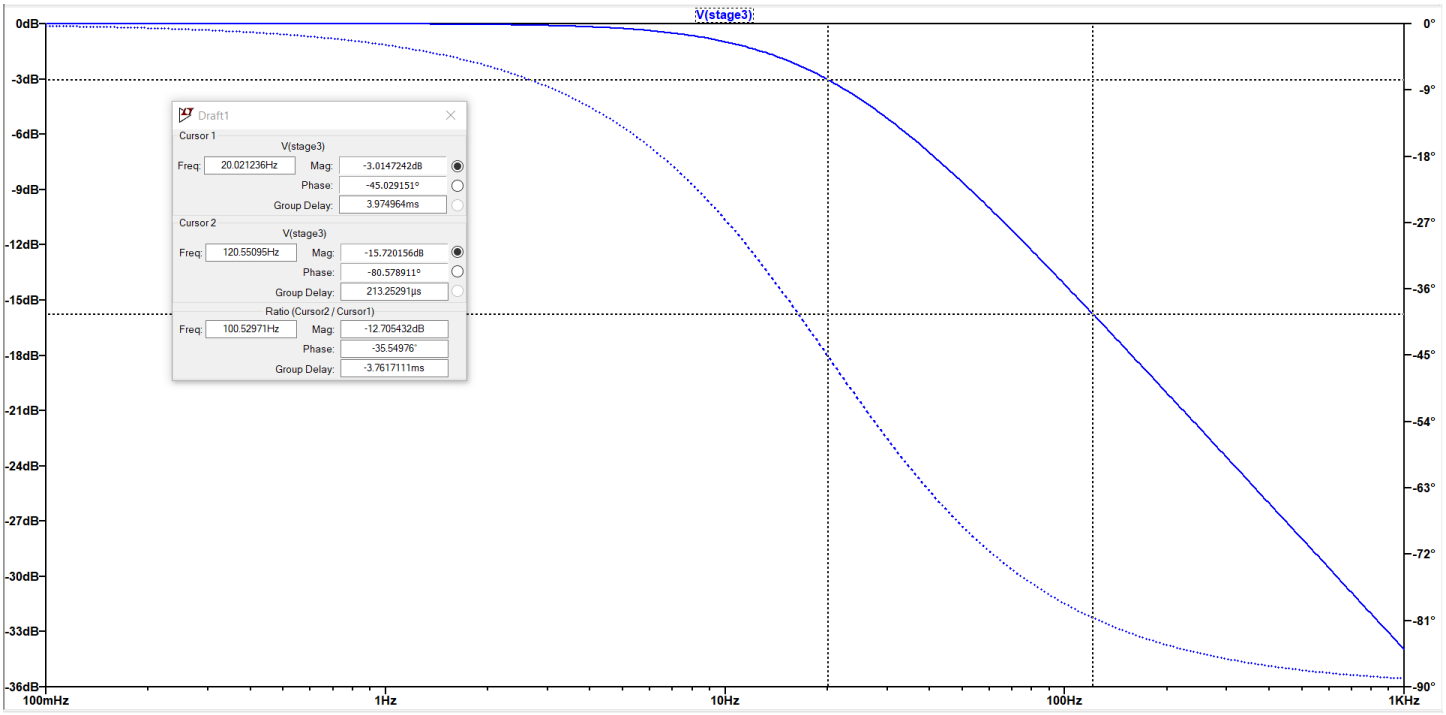


*Fig. 4: The Bode plot of **Stage 3** which has callouts for the cutoff frequency as well as a callout one decade higher to analyze slope behavior when separate from the rest of the system. The left callout, marking the -3dB point is as calculated in Table 3 at 20Hz.*

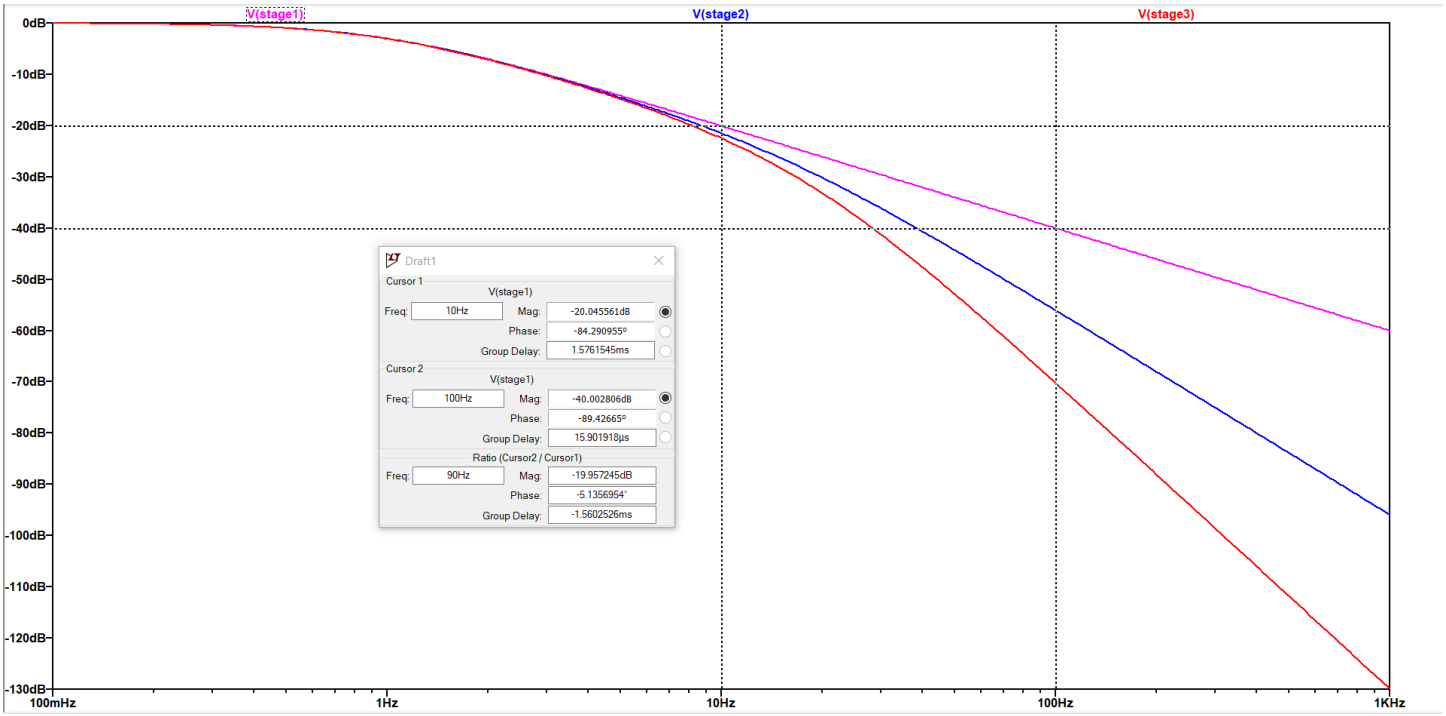


*Fig. 5: The system, now tied together, behaves as depicted above. The slope analysis for **Stage 1** is marked here. The slope is approximately -20dB per decade for the first stage.*

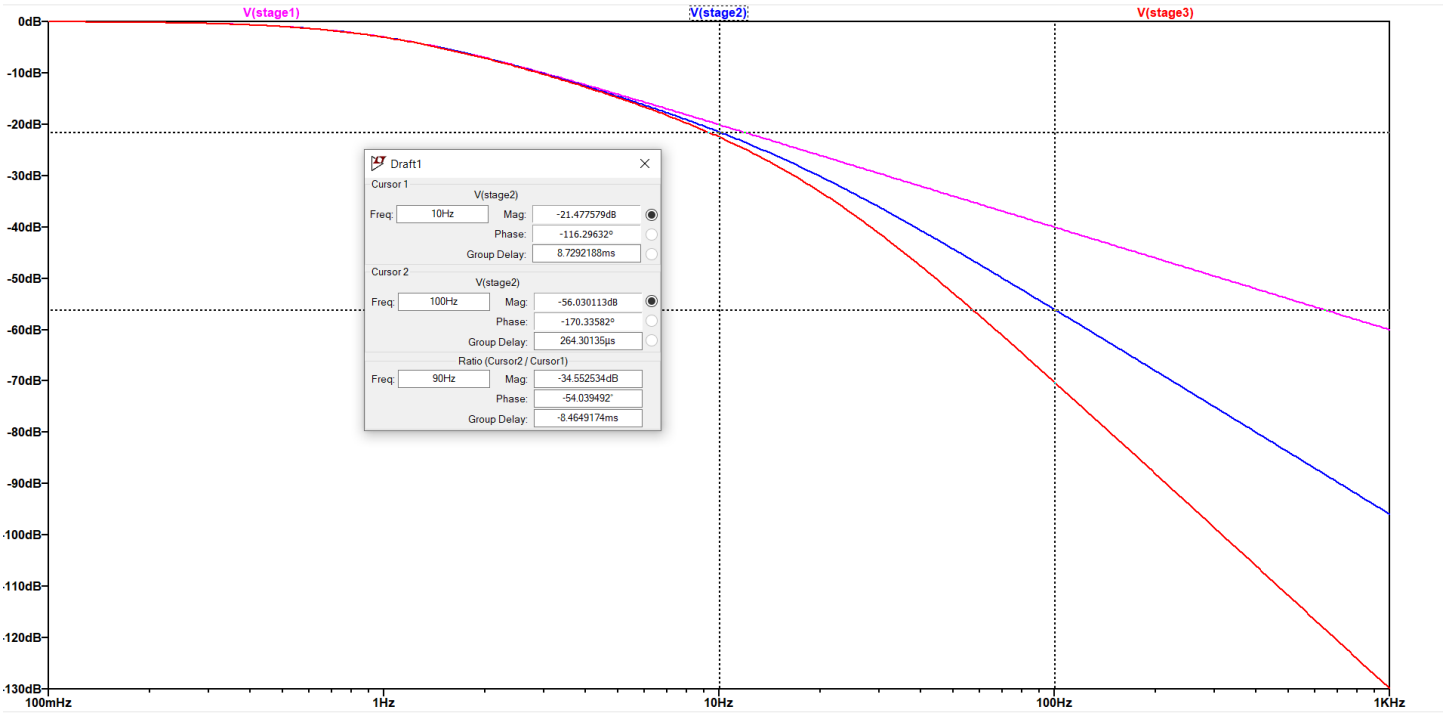


*Fig. 6: The slope analysis of **Stage 2** with all circuits ran in series with each other as in Figure 1. The slope of stage 2 is approximately -35 dB per decade.*

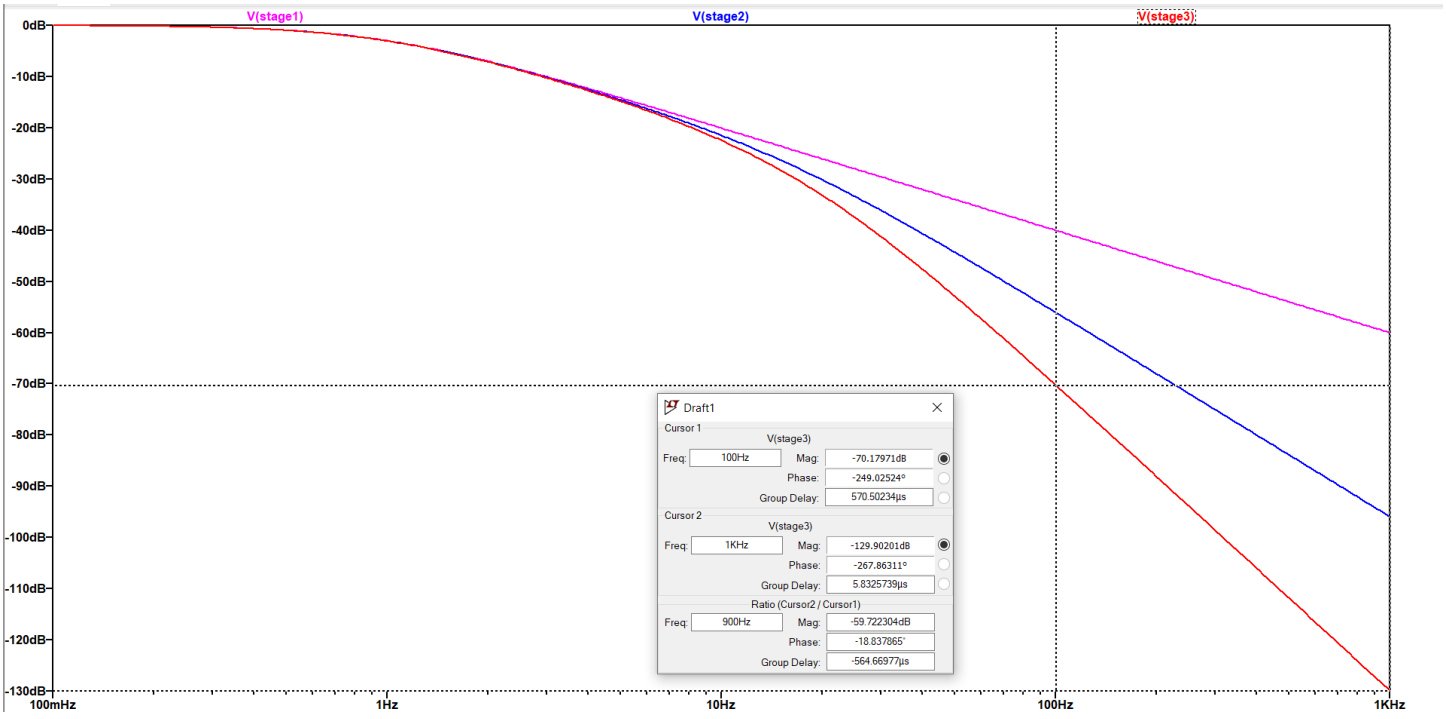


*Fig. 7: The bode plot slope analysis of **Stage 3**. The final cascaded stage has a slope of approximately -60dB per decade. The second callout is at the bottom right edge of the plot at 1kHz and is therefore not visible with the exception of the dashed line across the horizontal frequency values at the bottom of the plot.*

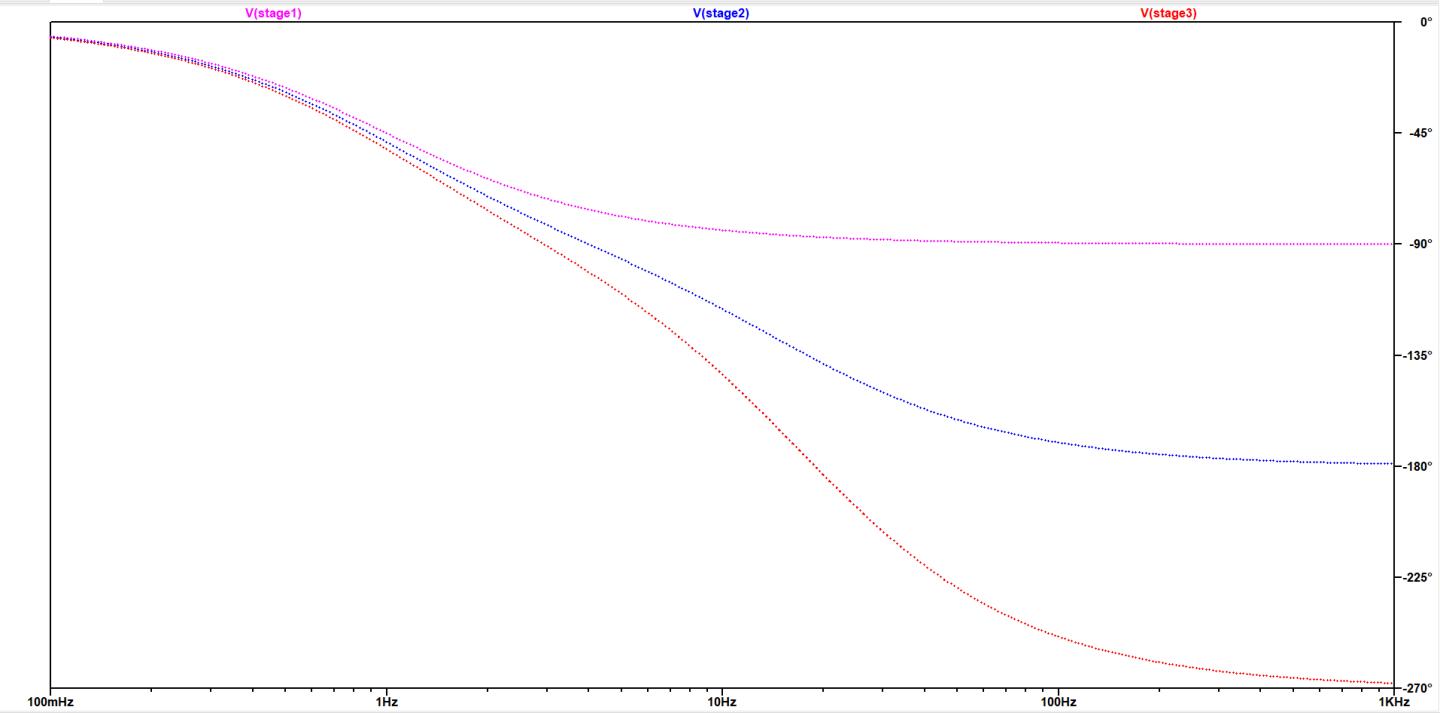


Fig. 8: The phase plot of the cascaded system.

-Verification of Plant Poles using both MATLAB and SPICE

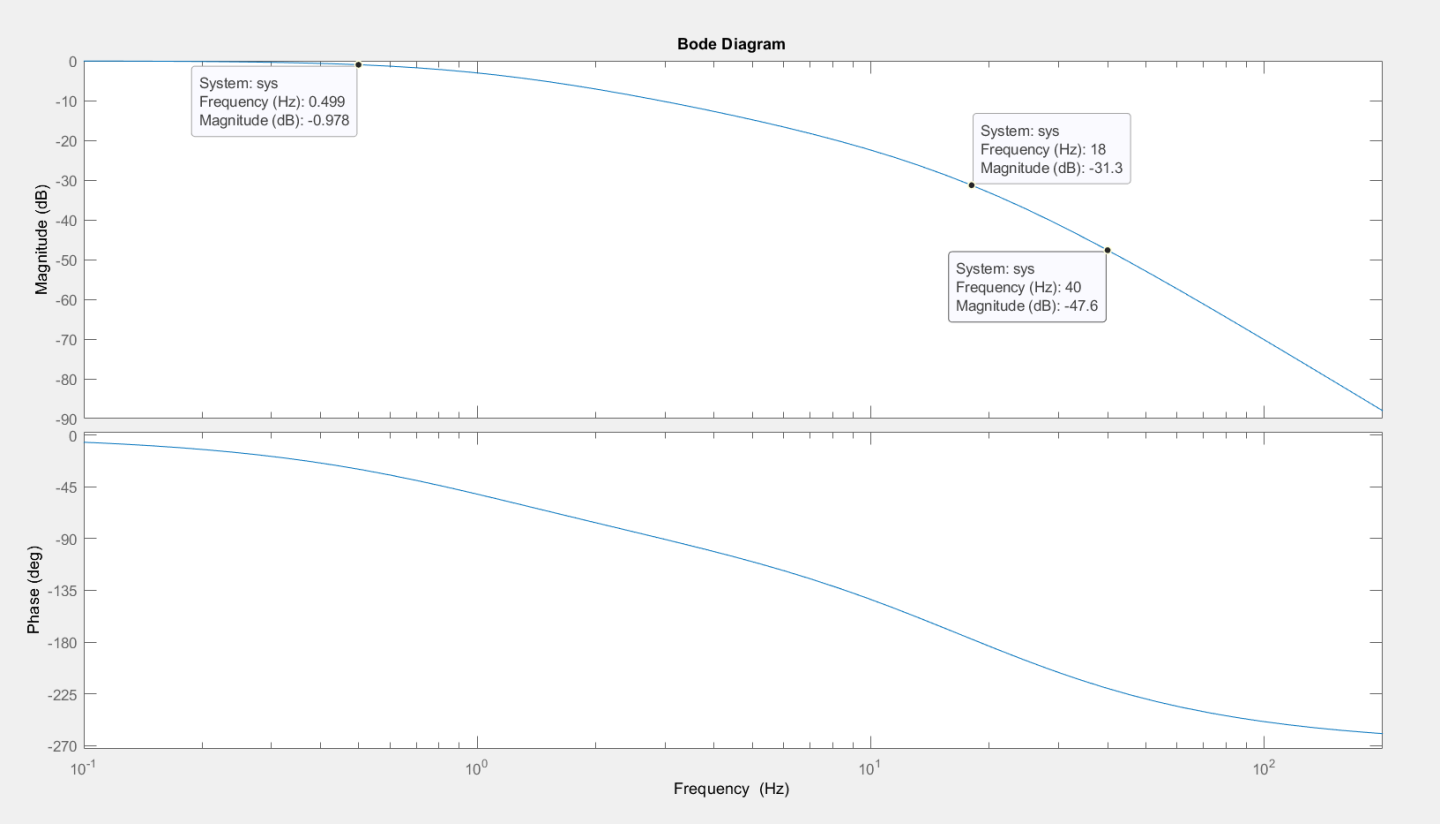


Fig. 9: The Bode plot of G(s) in MATLAB with three callouts for: A frequency below our lowest (0.5Hz), a frequency between our two highest (18 Hz), and a frequency higher than our highest frequency from table 3 (40 Hz). The magnitudes of each will be used for comparison in SPICE matching the same frequencies. We are looking for a match of amplitude to frequency in both plots to confirm our system poles.

*Note: Because SPICE does not allow more than 2 callouts, there is a requirement for more plots in order to show all three poles. The SPICE plots confirming our poles follow below with the first two chosen frequencies on the first plot, followed by a second plot with the third chosen frequency for pole confirmation.

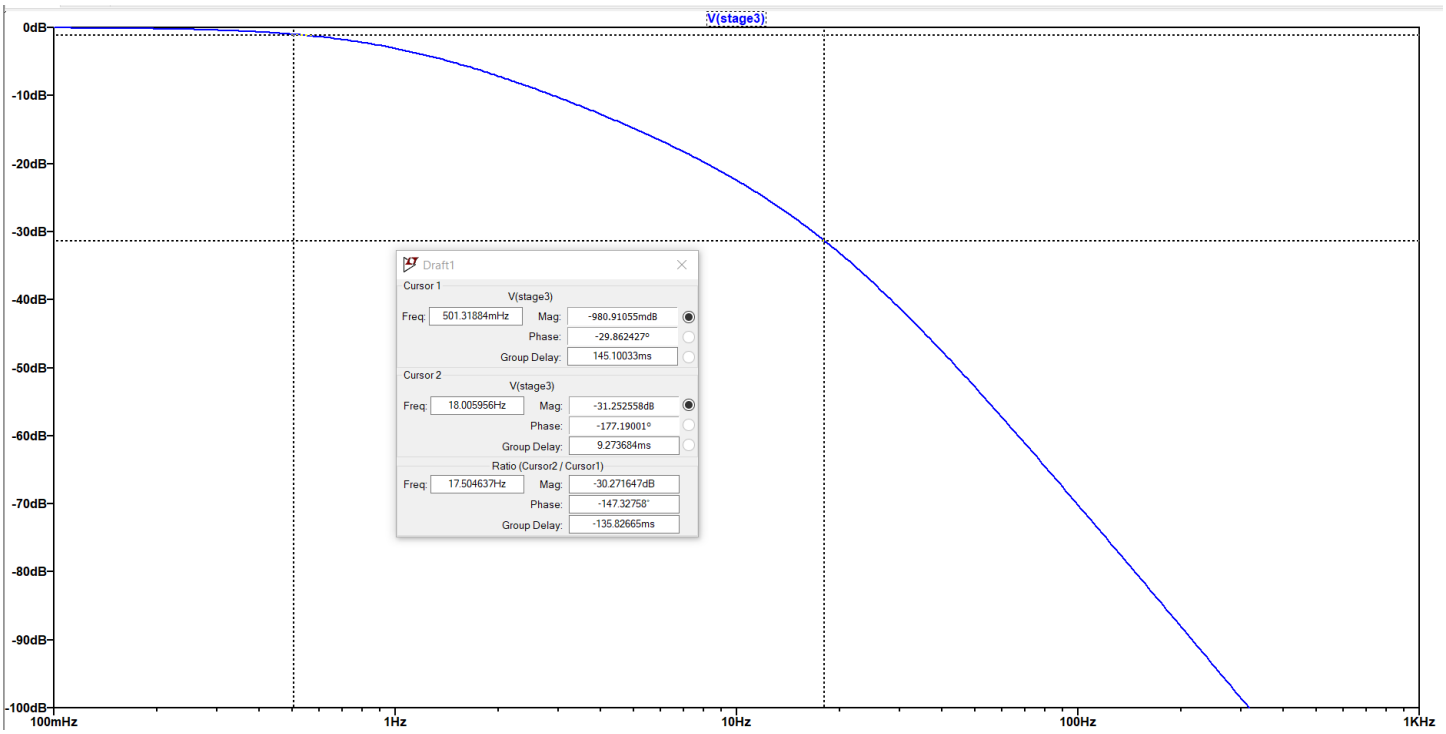


Fig. 10: Confirmation of the first 2 poles is complete. The magnitude at the chosen frequency matches our MATLAB plot.

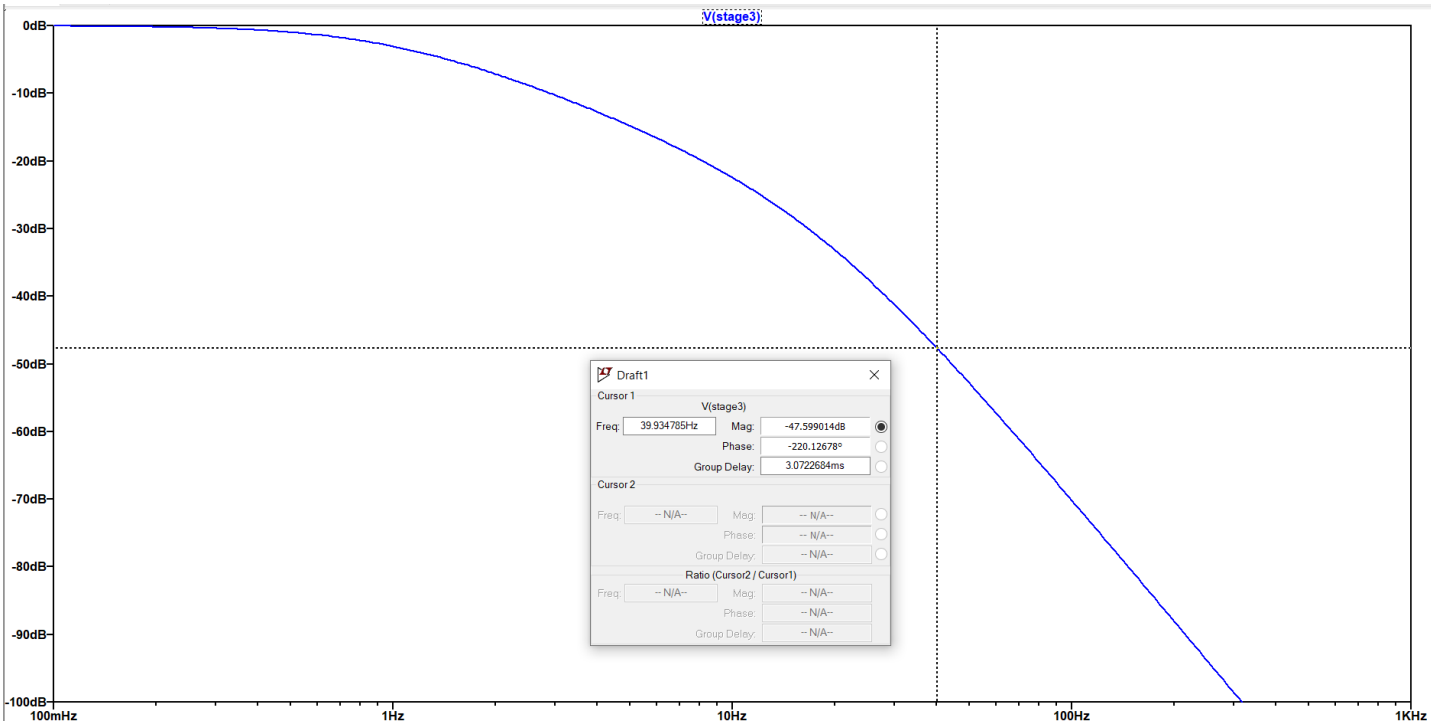


Fig. 11: Confirmation of the third pole is complete. The magnitude at the chosen frequency matches our MATLAB Plot.

II. 3 Verification of Uncompensated Step Response

-Plant Transfer Function with Unity Gain Feedback Loop:

$$T(s) = \frac{79{,}374}{(s+6.283)(s+100.531)(s+125.664)} \qquad \text{EQ (5)}$$

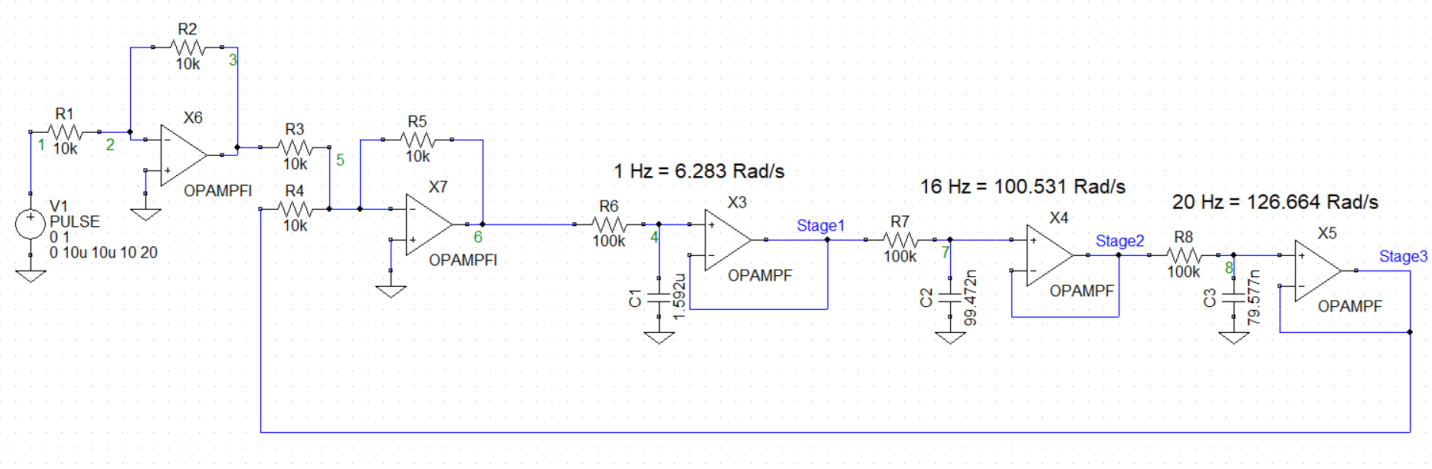


Fig. 12: The system modeled in Top SPICE with a unity gain feedback applied.

-Plant Response in MATLAB/SPICE with K = 1.

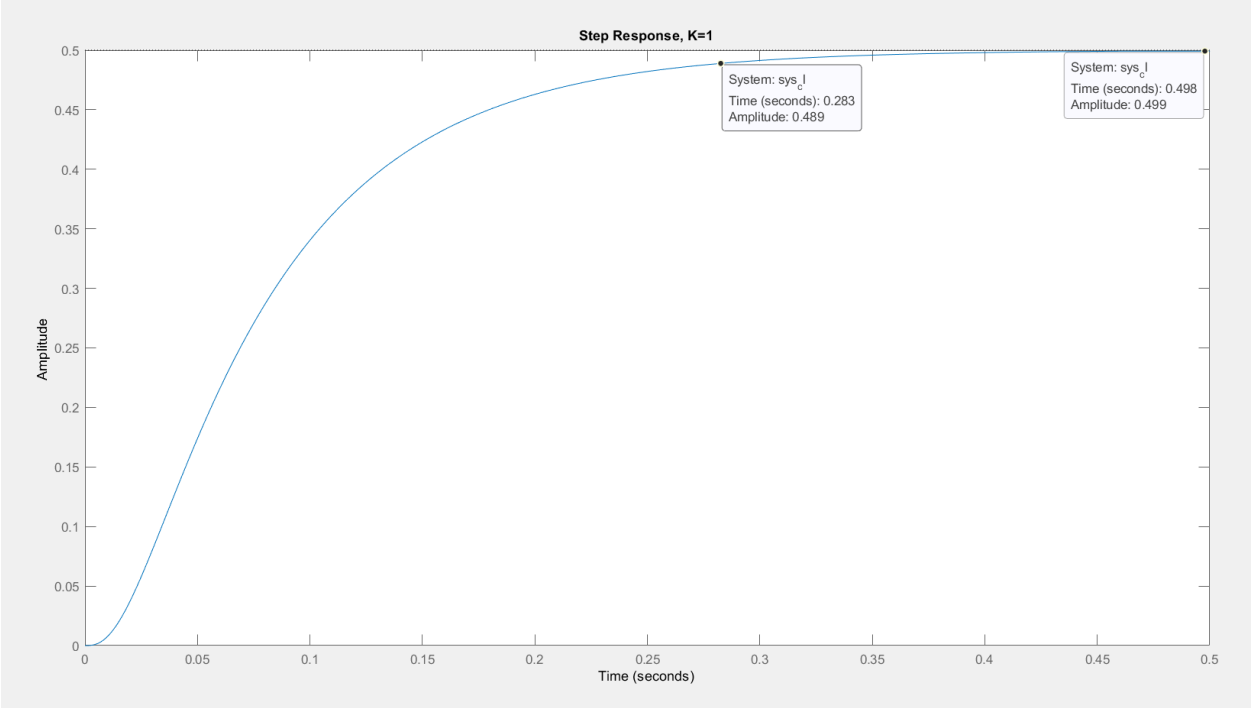


Fig. 13: Step Response of unity gain feedback system with K=1. The callouts indicate, from left to right, Ts = 0.283s, and steady state value at approximately (0.5), which indicates that steady state error is e(∞) = (1 - 0.5 = 0.5).

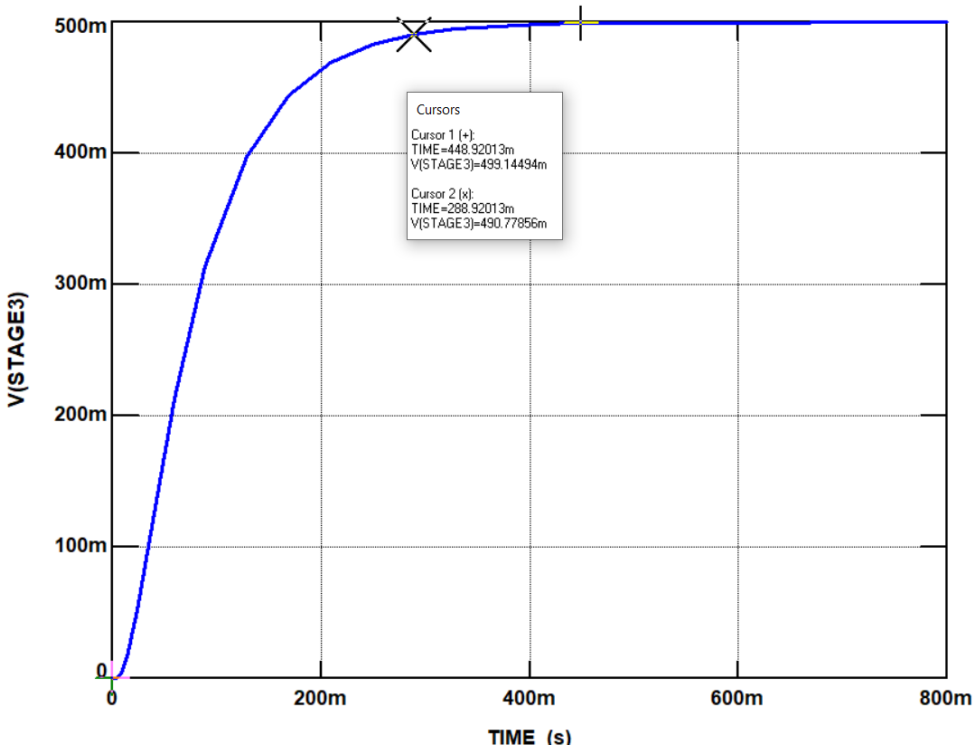


Fig. 14: The two system step responses side by side. The callouts for Ts and steady state value are marked. As we can see, both simulations match up with values being quite similar (barring the scrub rate for the data points). Both steady state values match perfectly and Ts ≈0.283 s. for both plots. The system is confirmed.

Table 4: Uncompensated Step Response Comparison		
	MATLAB	**SPICE**
Ts	0.283s	0.288s
Tp	NA (no peaking)	NA (no peaking)
%OS	0%	0%
Steady State Error	0.5	0.5

Table 4: Uncompensated Step Response Comparison of our two simulations. As we can see, the system behaves as expected in both simulation suites which confirms that the system was assembled correctly. We have succeeded in taking our paper calculations and turning our math into a realized system of physical parts to carry out our intent. Engineering!

II.4 MATLAB Analysis of Uncompensated Plant

Calculation of Zeta:

Given the target of 10% Over-Shoot (OS), we can find a value of Zeta, also known as the damping ratio. The equation for finding Zeta given %OS is given as:

$$\zeta = \frac{-\ln(\%OS)}{\sqrt{\pi^2 + ln^2(\%OS)}} \qquad \text{EQ (6)}$$

Using equation 6, the value of Zeta for the given percent OS was found to be:

$$\zeta = 0.591$$

MATLAB Root Locus of Plant:

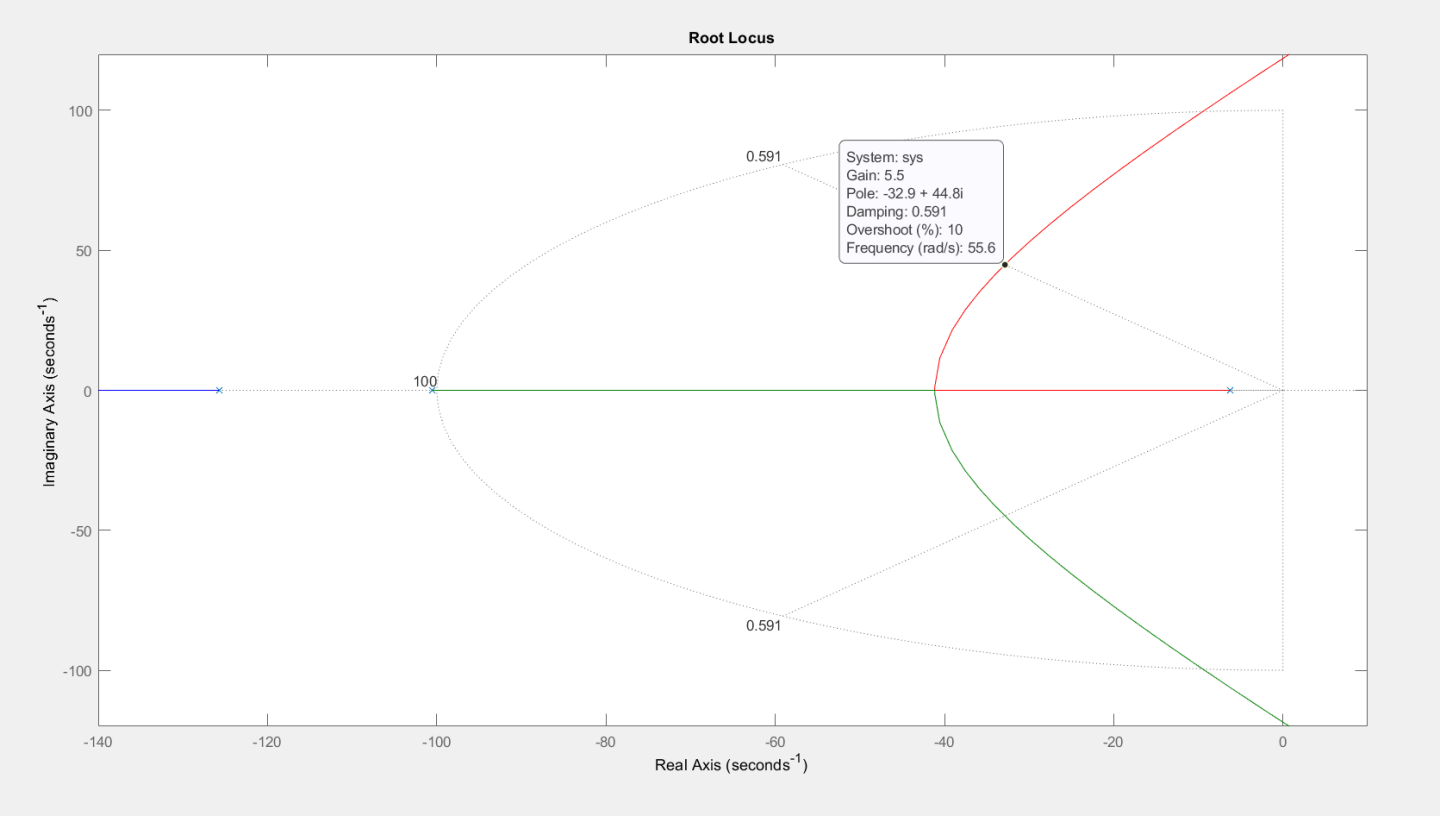


Fig. 15: The Root Locus of the given plant. The callout indicates the pole on the root locus for the given value of Zeta using the percent overshoot of 10% as the factor used to calculate Zeta=0.591.

The pole is located at (s = -32.9±j44.8).

Editor - C:\Users\rober\OneDrive\Desktop\PS8_Root_locus_Code.m

PS8_Root_locus_Code.m

```
clc, clear, close all
s = tf('s');
sys = ((79374)/((s+6.283)*(s+100.531)*(s+125.664)));
%bode (sys)
%pzmap (sys)
rlocus (sys)
%axis([-2 0 -2 2])
zeta = 0.59116;
wn = 100;
sgrid (zeta,wn)
%k = 905;
%sys_cl = feedback (k*sys, 1)
%step (sys_cl)
%step(sys)
```

Fig. 16: The code used to run Root Locus on the given Plant. The 'sgrid' command was used to draw the angle of Zeta on the plot in order to call out the pole located on it accurately.

III.1 Function of PI Controller

The Proportional-Integral (PI) controller is a mechanism used to control the feedback of a control system. It is used mainly to improve steady state error of a closed loop system (although it can be cascaded as well). The PI controller employs two different types of control techniques into one, as the hyphenated name implies. Namely, the use of a proportional controller in conjunction with an integrator. The proportional controller also works to decrease steady state error and can assist in decreasing rise time but does so at the expense of added overshoot at higher gains. The integral controller is used in conjunction with the proportional to cancel out the negative effects that they may produce by themselves. This is possible because the two controls work on different parts of our transfer function. That is, the proportional control works on the numerator while the integral control works on the denominator of our transfer function. By using the two in conjunction, we can change the output of the system, decreasing time to peak and steady state error while having little to no negative effect. The main negative effects that come from the use of a PI controller are increase in percent overshoot and settling time.

We are able to ensure that the two work closely together to produce the desired effects without magnified negative effects by exploiting the placement of the two (one zero, and one pole) such that they are close enough together to hypothetically 'cancel' each other out of the original transfer function. This leaves us with the ability to improve steady state error and time to peak while leaving the rest of the system response virtually unchanged. While the target for PI controller is to reduce steady state error to zero, this is not always possible, as some systems may operate at gain levels that are impossible to reduce to zero steady state error. For these systems, other methods can be employed in conjunction with the PI controller for nominal results.

III.2 Design of the PI Controller

For the PI controller, the pole at s = 0 is a given. My decision was to place the zero at s = -0.1 initially in order to have a starting point as close to the added pole as possible. The zero chosen worked well so there was no modification to it. This gives the compensated equation below:

$$G_{PI}(s) = \frac{79{,}374(s+0.1)}{s(s+6.283)(s+100.531)(s+125.664)} \qquad \text{EQ (7)}$$

PI Controller Root Locus and Step Response:

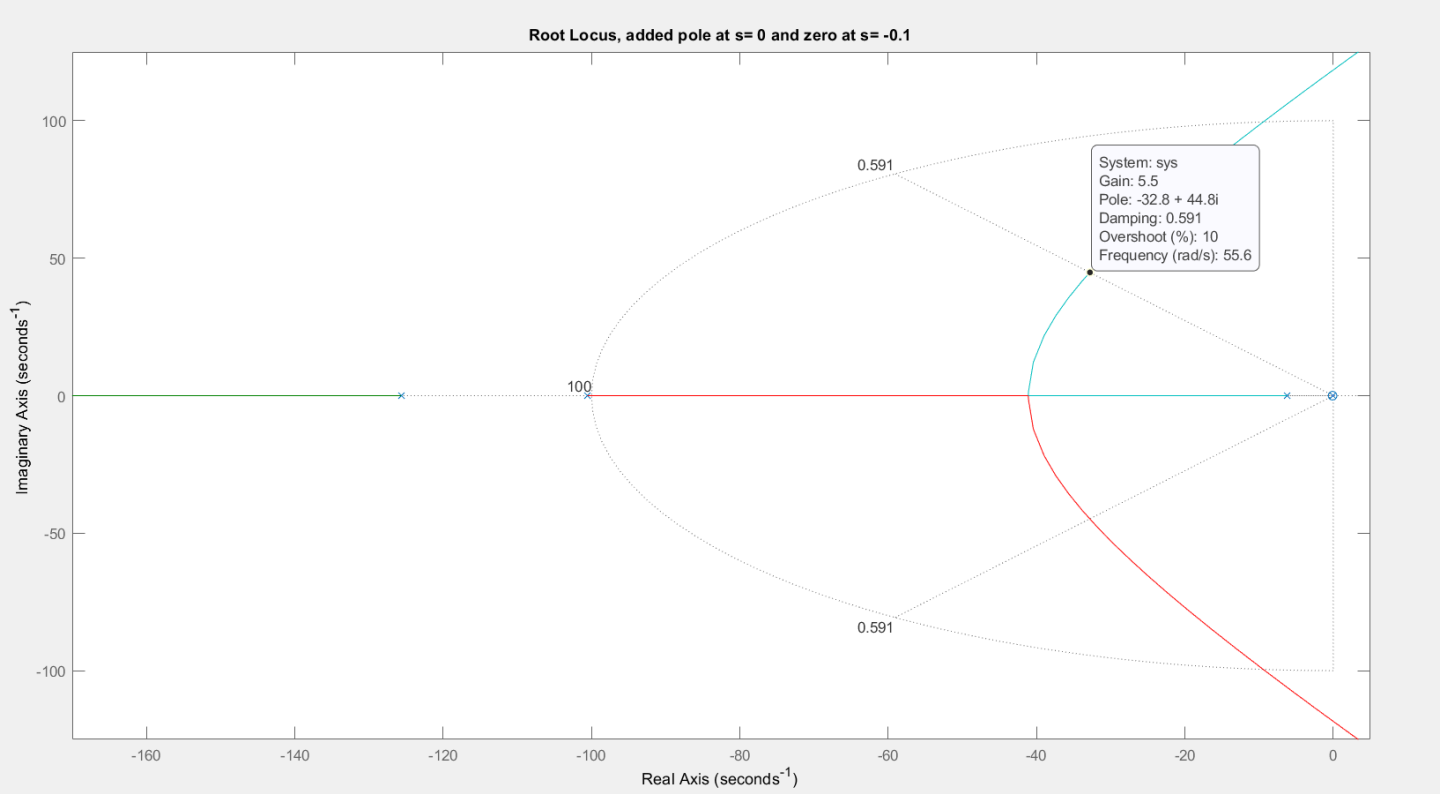


Fig. 17: The root locus with the PI compensator is virtually unchanged, as expected. We want to keep our dominant pole on the root locus while improving steady state error. This goal is accomplished as shown in Fig. 19.

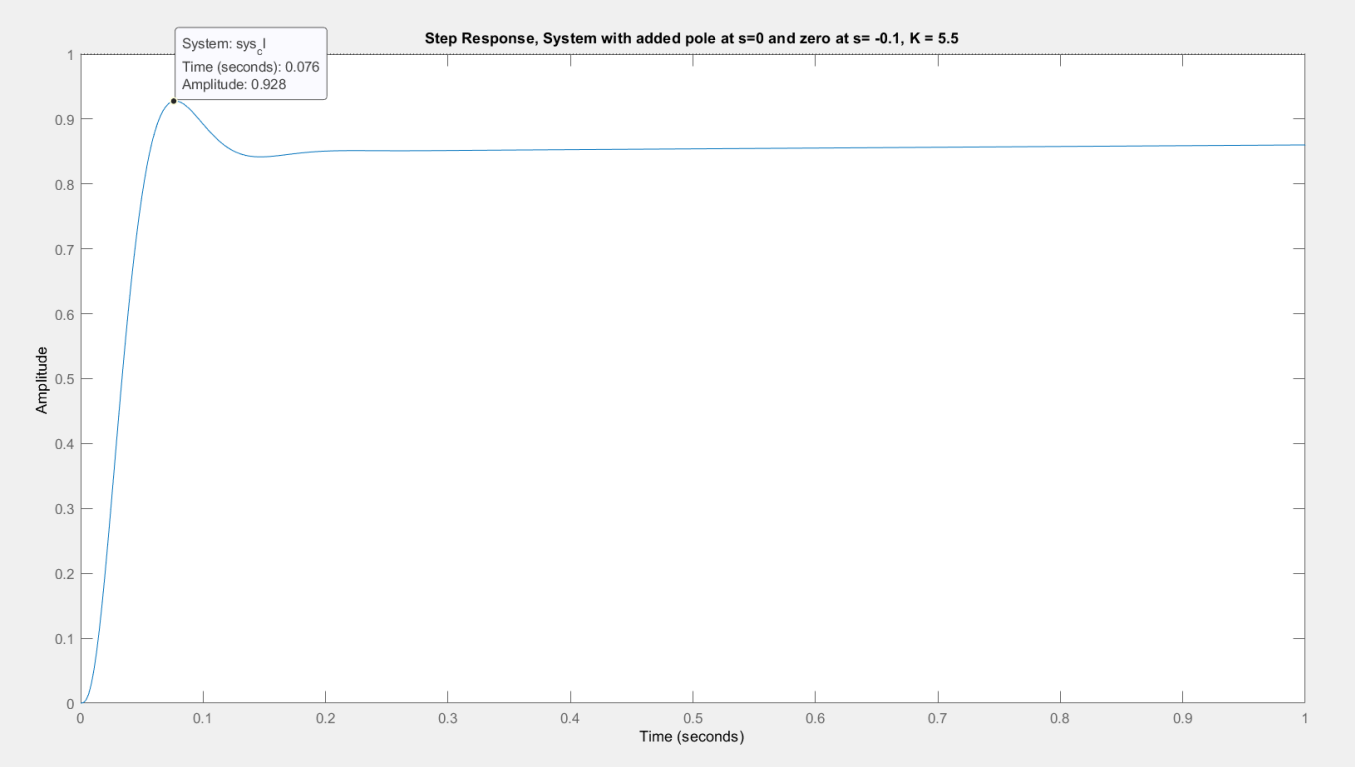
Documents ▸ MATLAB

Editor - C:\Users\rober\OneDrive\Desktop\PS8_Root_locus_Code.m

PS8_Root_locus_Code.m

```
1 -    clc, clear, close all
2 -    s = tf('s');
3 -    sys = ((79374)*(s+0.1))/(s*(s+6.283)*(s+100.531)*(s+125.664));
4      %bode (sys)
5      %pzmap (sys)
6 -    rlocus (sys)
7      %axis([-2 0 -2 2])
8 -    zeta = 0.591;
9 -    wn = 100;
10 -   sgrid (zeta,wn)
11     %k = 5.5;
12     %sys_cl = feedback (k*sys, 1)
13     %step (sys_cl)
14     %step(sys)
```

Fig. 18: The code used to run the PI controller. The equation is changed to reflect the design parameters.



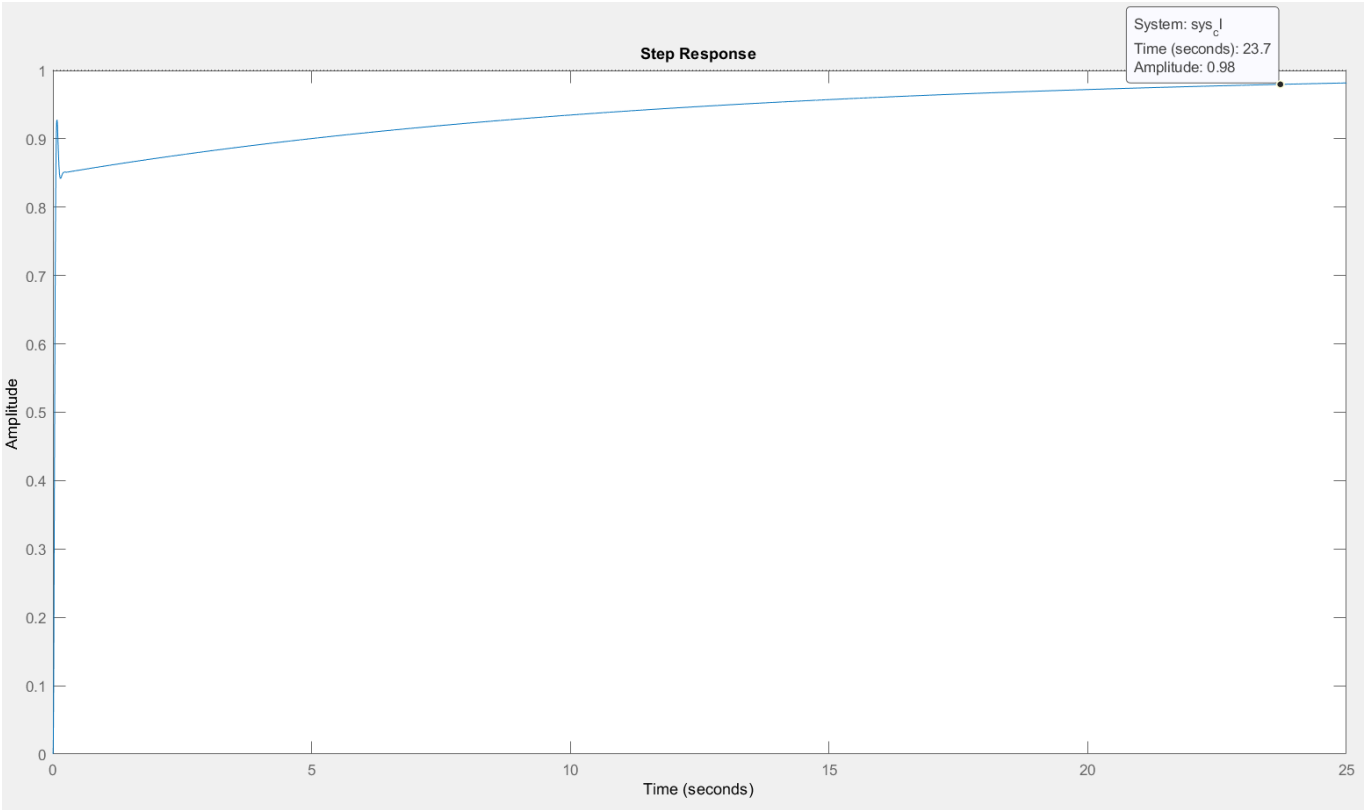


Fig. 19 a/b: The two figures above show the step response with the PI controller. The top plot shows the new response is underdamped rather than overdamped, as it was without compensation. The peak time is now Tp = 0.076 seconds with an amplitude of 0.928. The second plot is necessary to show the settling time, Ts = 23.7 s to get within 2% of the final value. The steady state error is now [e (∞) = 0.002], but the system takes 48.6 seconds to completely reach this final value.

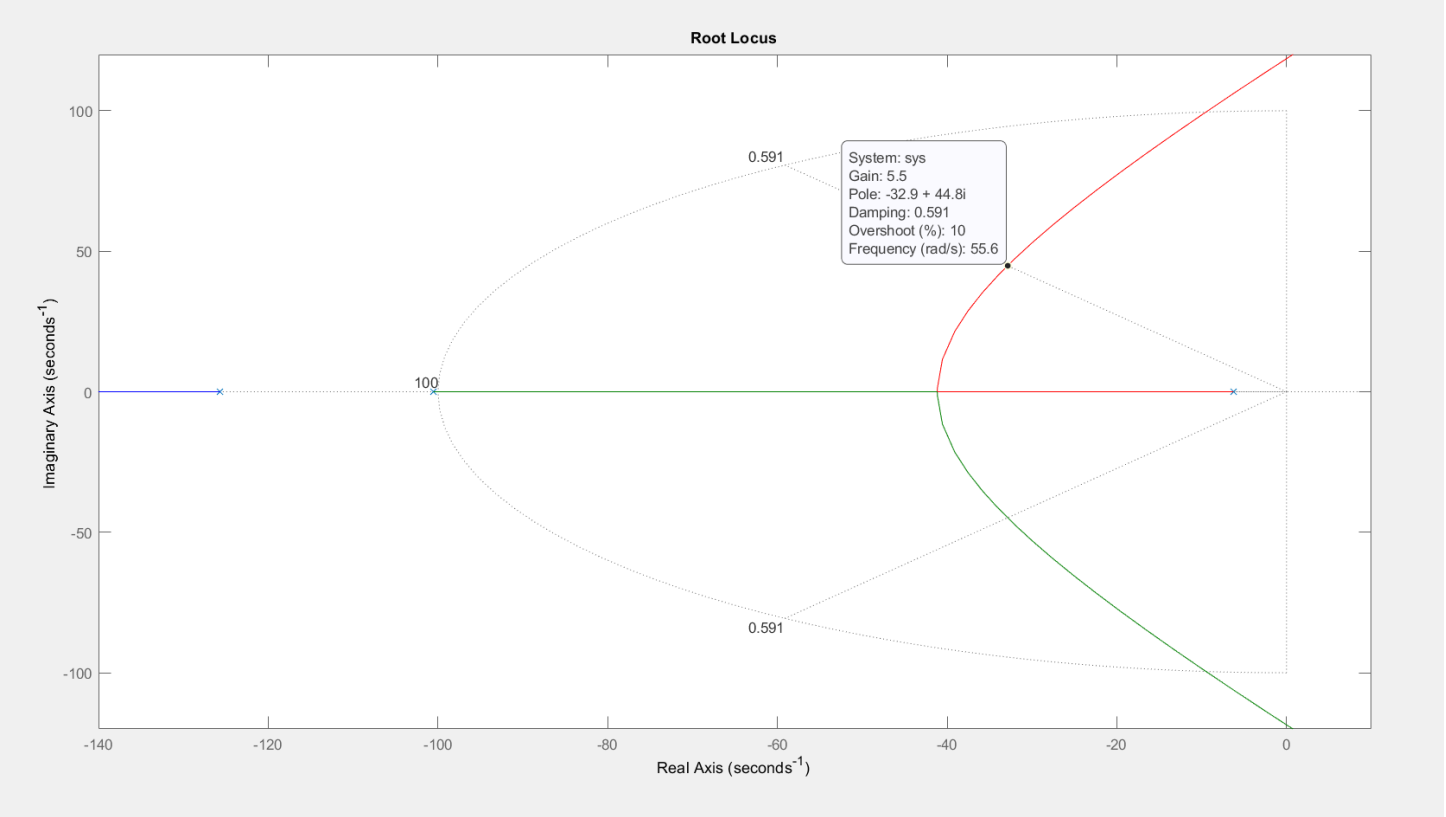
Documents ▸ MATLAB

Editor - C:\Users\rober\OneDrive\Desktop\PS8_Root_locus_Code.m

PS8_Root_locus_Code.m

```
clc, clear, close all
s = tf('s');
sys = ((79374)*(s+0.1))/(s*(s+6.283)*(s+100.531)*(s+125.664));
%bode (sys)
%pzmap (sys)
%rlocus (sys)
%axis([-2 0 -2 2])
%zeta = 0.591;
%wn = 100;
%sgrid (zeta,wn)
k = 5.5;
sys_cl = feedback (k*sys, 1)
step (sys_cl)
%step(sys)
```

Fig. 20: Code used to run steady state on PI controlled system. Gain was set to match parameters for dominant pole found during root locus analysis of uncompensated system.



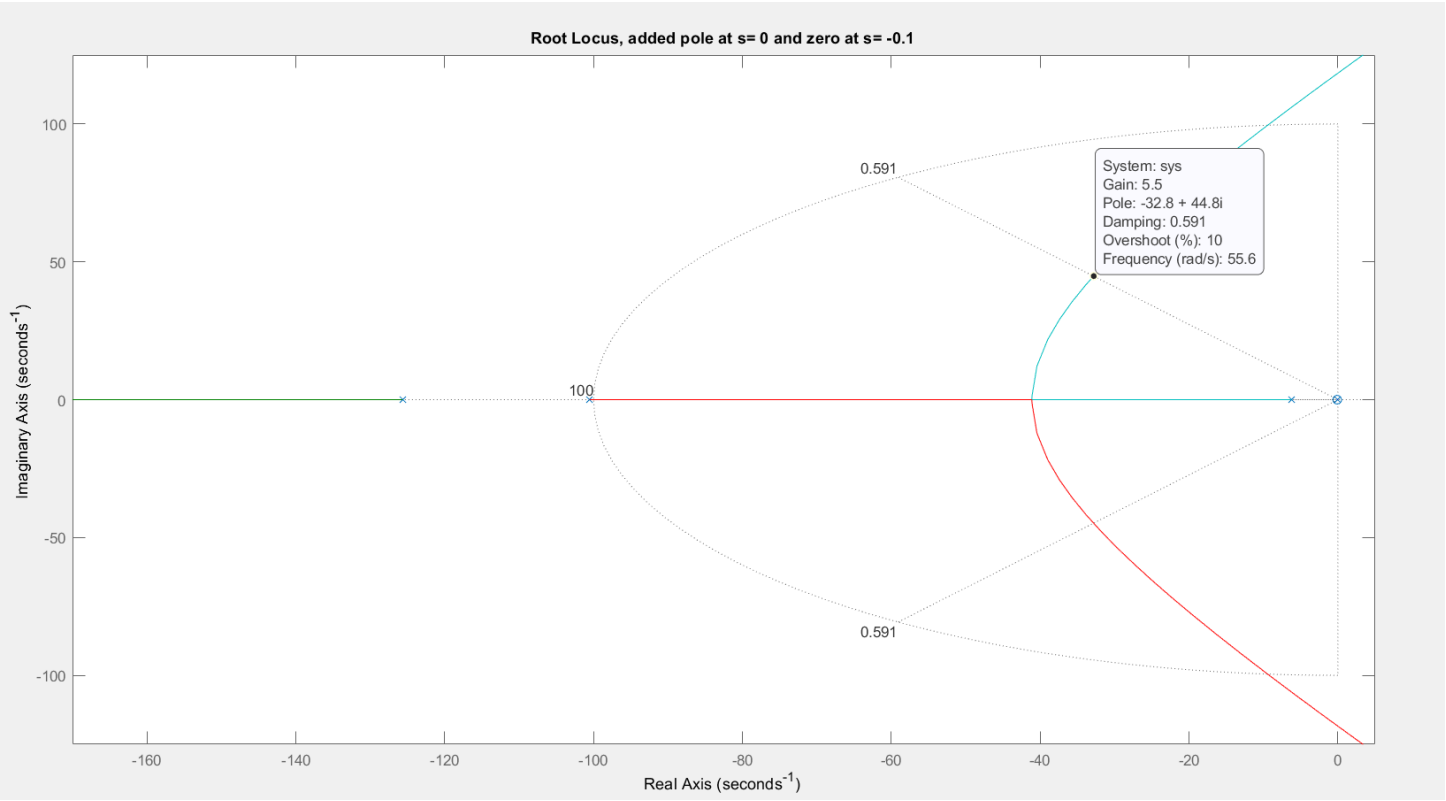
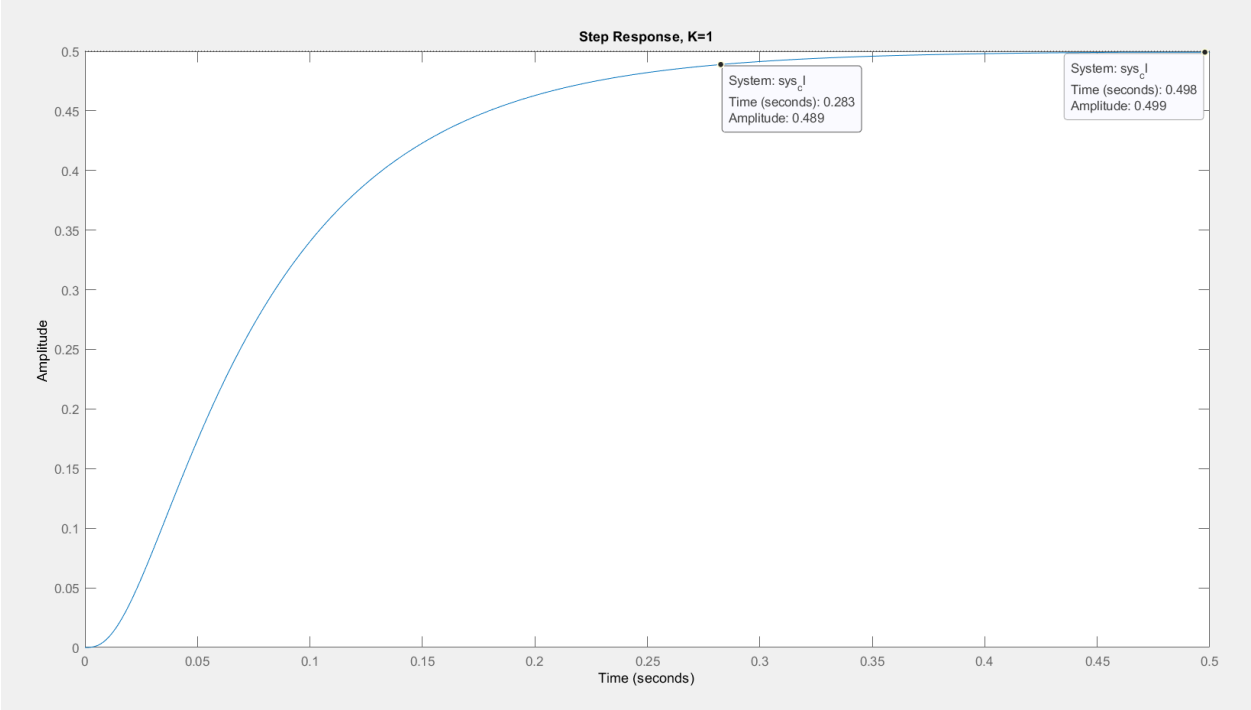


Fig. 21: The comparison of the root locus before and after the PI controller is added respectively. The plot headings are written to indicate which system is which. There is virtually no change to the root locus after adding the PI. This is what was expected as we want our root locus after PI to run through our dominant pole. Of note, the dominant pole at s=-125.664 moves to s=-167 when k=5.5



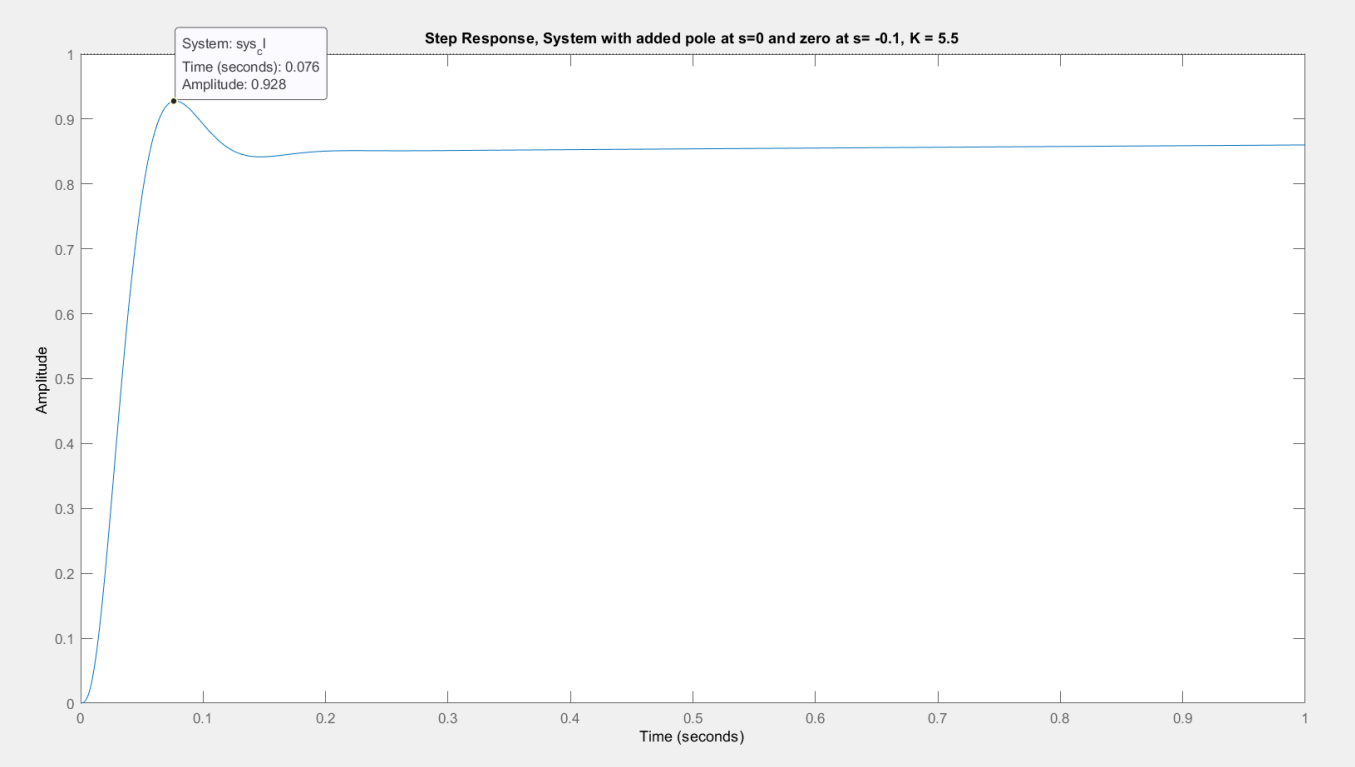


Fig. 22 A/B: The comparison of the step response of uncompensated versus PI compensated. The steady state response changes from overdamped to underdamped with the addition of the PI compensation. Steady state error is improved with the new compensator at a sacrifice of Ts, but most of that time is spent gradually moving to a steady state error of almost zero. We now have a peak time with the compensator as well.

Table 5: Comparison of Dominant Pole Location		
	Pre-PI	**Post-PI**
Dominant Pole Locations	$s = -32.9 \pm j44.8$	$s = -32.8 \pm j44.8$
K	5.5	5.5

As shown above, the root locus is unchanged by our PI control as desired. We have succeeded in driving steady state error to approximately zero while maintaining the original system.

IV.1 PD Design

Currently, our system stands as such. We have taken the original plant, G(s), and added a PI controller which successfully drove the steady state error almost to zero (.2% shy at a value of 0.998). We now have our new function $G_{PI}(s)$ with the poles and zeros indicated in the table below.

Table 6: Poles and Zeros of System before PD Controller

$$G_{PI}(s) = \frac{79{,}374(s + 0.1)}{s(s + 6.283)(s + 100.531)(s + 125.664)}$$

Zeros	Poles
-0.1	0
	-6.283
	-100.531
	-125.664

Given a target parameter of 20% improvement of T_p, we can calculate a new T_p. Using the given information, we want 80% of the original T_p. The original time is T_{po} = 0.076s, and the new peak time is T_{pn} = 0.0608s.

We can use the new peak time to find the Imaginary component of our desired dominant pole by the relationship:

$$T_p = \frac{\pi}{|Im|} \qquad \text{EQ (8)}$$

$$\cos\theta = Zeta \qquad \text{EQ (9)}$$

This gives us a new imaginary component of 51.671. We can combine this with our factor for Zeta. We need this because we want our damping ratio to remain the same while finding a new point along it which will give us the desired improvement in peak time. With some trigonometry, we use EQ 9 to give us an angle of 53.77 deg. (using the ARCCOS function and Zeta=0.591). Once we have the angle and the imaginary value from EQ 8, we use the tangent function to find the real component of our desired pole.

$$\frac{51.671}{\tan(53.77)} = 37.859 = Real\ Component\ of\ desired\ Pole\ location \qquad \text{EQ (10)}$$

Our new desired pole location for our 20% improvement to T_p is:

$$s = 37.859 \pm 51.671$$

Using root locus design techniques and finding the sum of all angles, we find that the new zero needs to be at

$$s = -254.43$$

Making the new equation with PD controller added:

$$G_{PID}(s) = \frac{0.0293(s+0.1)(s+254.43)}{s} \qquad \text{EQ (11)}$$

So, we have separately, without the plant or gain values:

$$G_{PI}(s) = \frac{(s+0.1)}{s}$$

$$G_{PD}(s) = (s+254.43)$$

IV.2 PID Verification in MATLAB

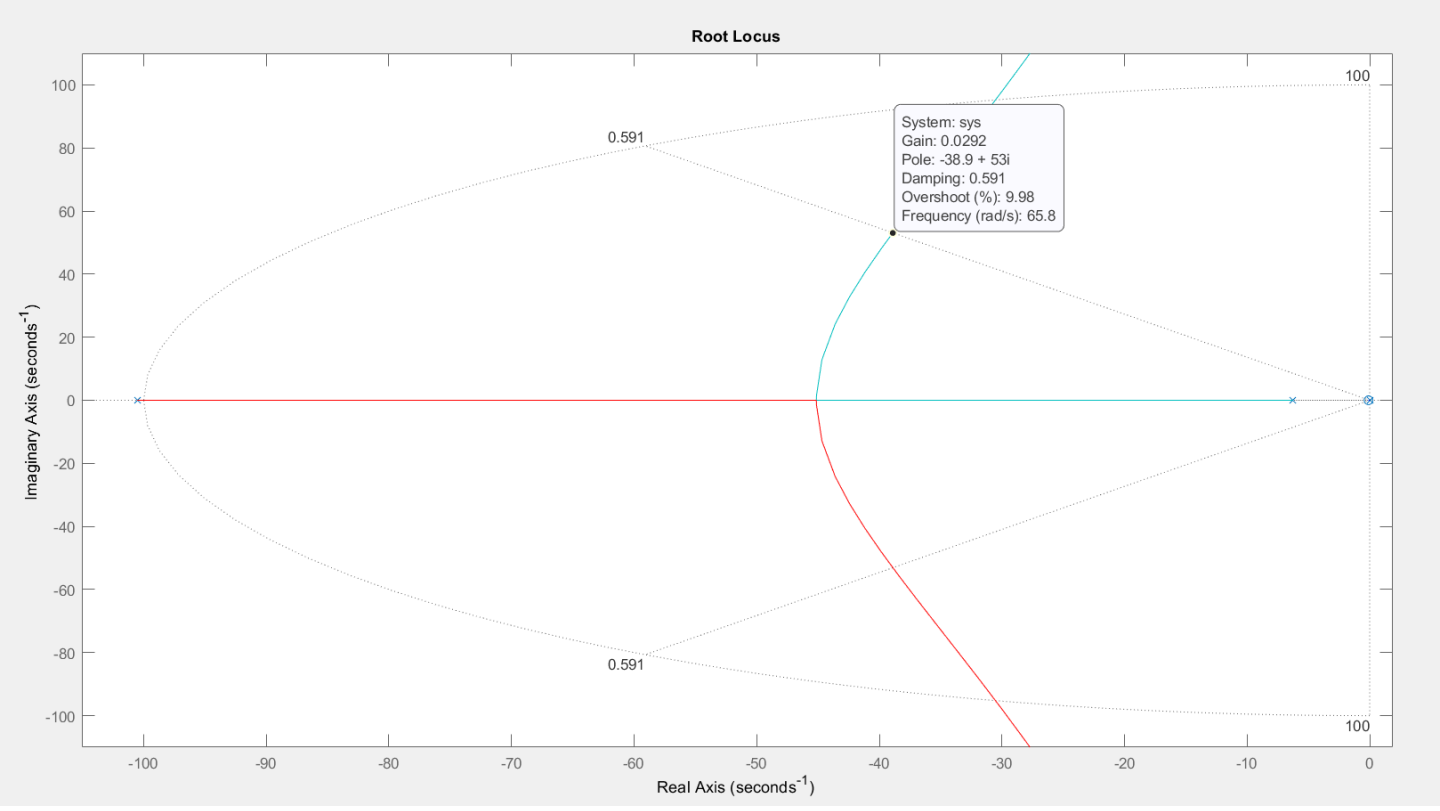


Fig. 23: PID controller root locus with dominant pole called out. The pole location is off by (-1.04-j1.43) but the steady state response still acquires the desired effect of a 20% improvement in Tp as will be shown below.

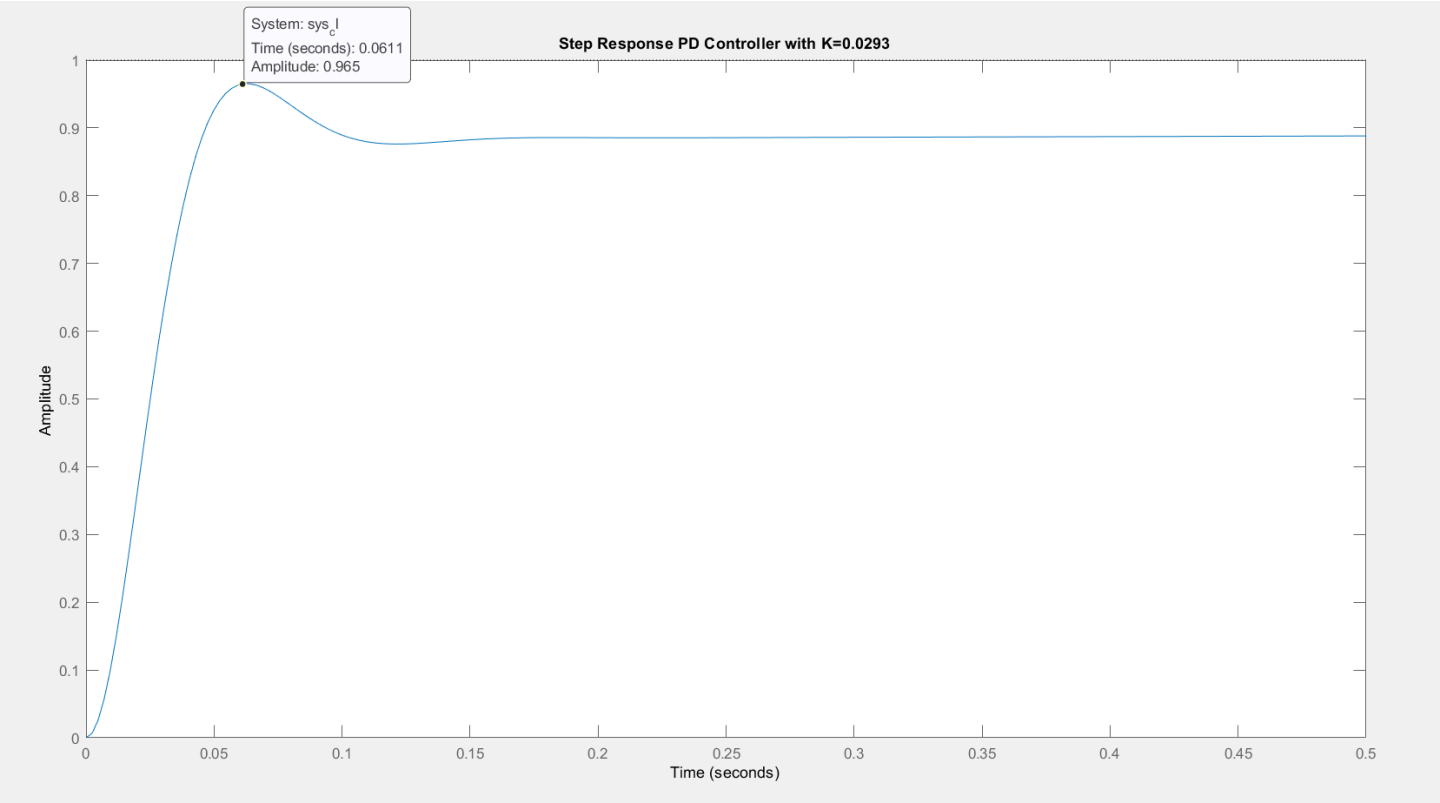


Fig. 24: The step response of the PID controller with gain indicted in title and callout for Tp. The 20% improvement of peak time is successful and we have maintained steady state error of approximately zero.

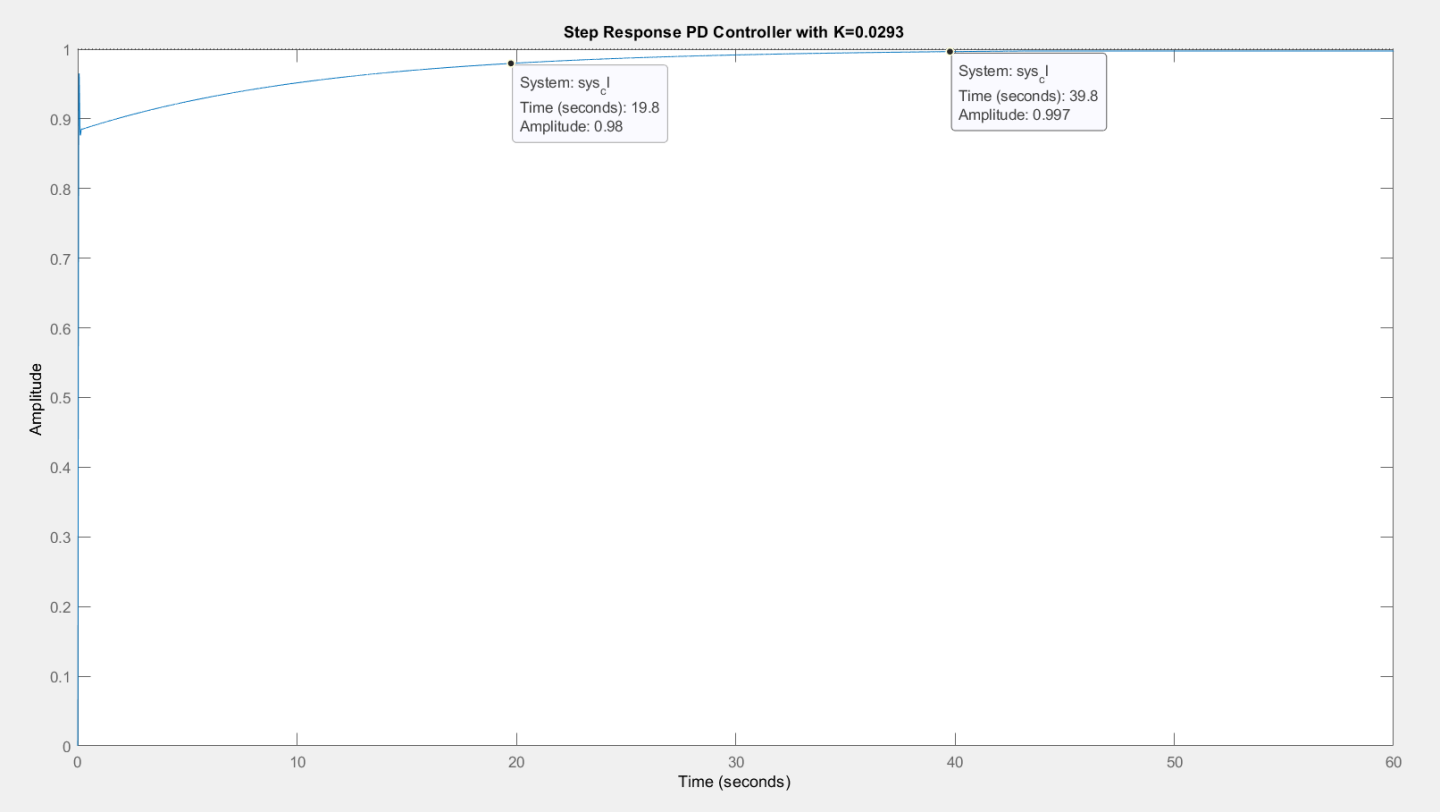


Fig. 25: The expanded time lapse of the steady state response shows that Ts = 19.8 seconds (within 2% of FV) and that our steady state is reached at t=39.8s with a steady state error of [e (∞) = 0.003].

IV.3 PID Verification in SPICE

Using our G_{PI} and G_{PD} equations above along with our PID gain of k=0.0293 from the MATLAB output, we are able to realize physical circuitry to model our system in SPICE. The table below will show the calculations:

Table 7: Gain Calculation				
Component	Equation	Gain	(Resistor/Capacitor) $_s$	(Capacitor/Resistor) $_f$
P	$K = \frac{R_f}{R_s}$	K_1 = 7.455	10kΩ	74.55kΩ
I	$K = \frac{1}{RC}$	K_2 = 0.7455	100kΩ	13.414uF
D	$K = RC$	K_3 = 0.0293	2.93uF	10kΩ

Table 7: The equations above were used to build the op-amps for the PID controller. The schematic follows.

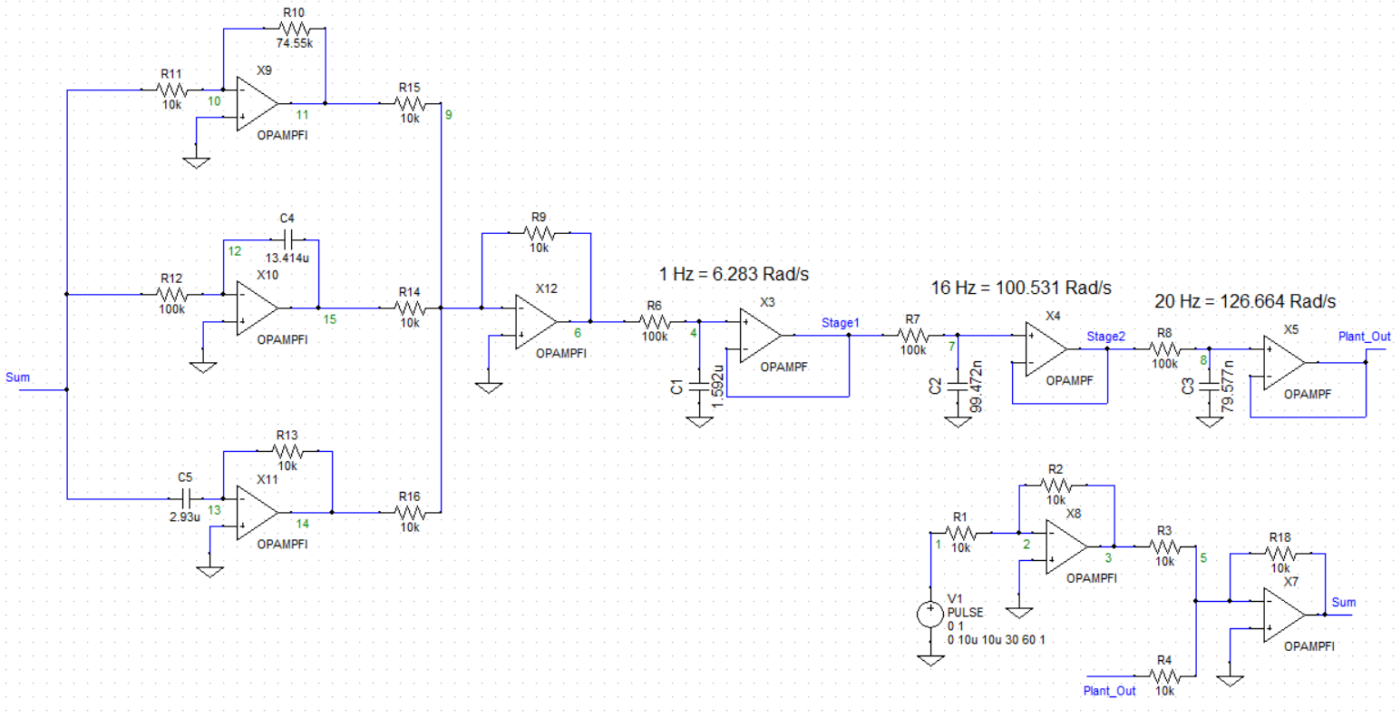


Fig. 26: The schematic of the PID controller, the Summing Junction, and the Plant. The values in Table 7 were used to build the Op-Amps in the PID controller to the left.

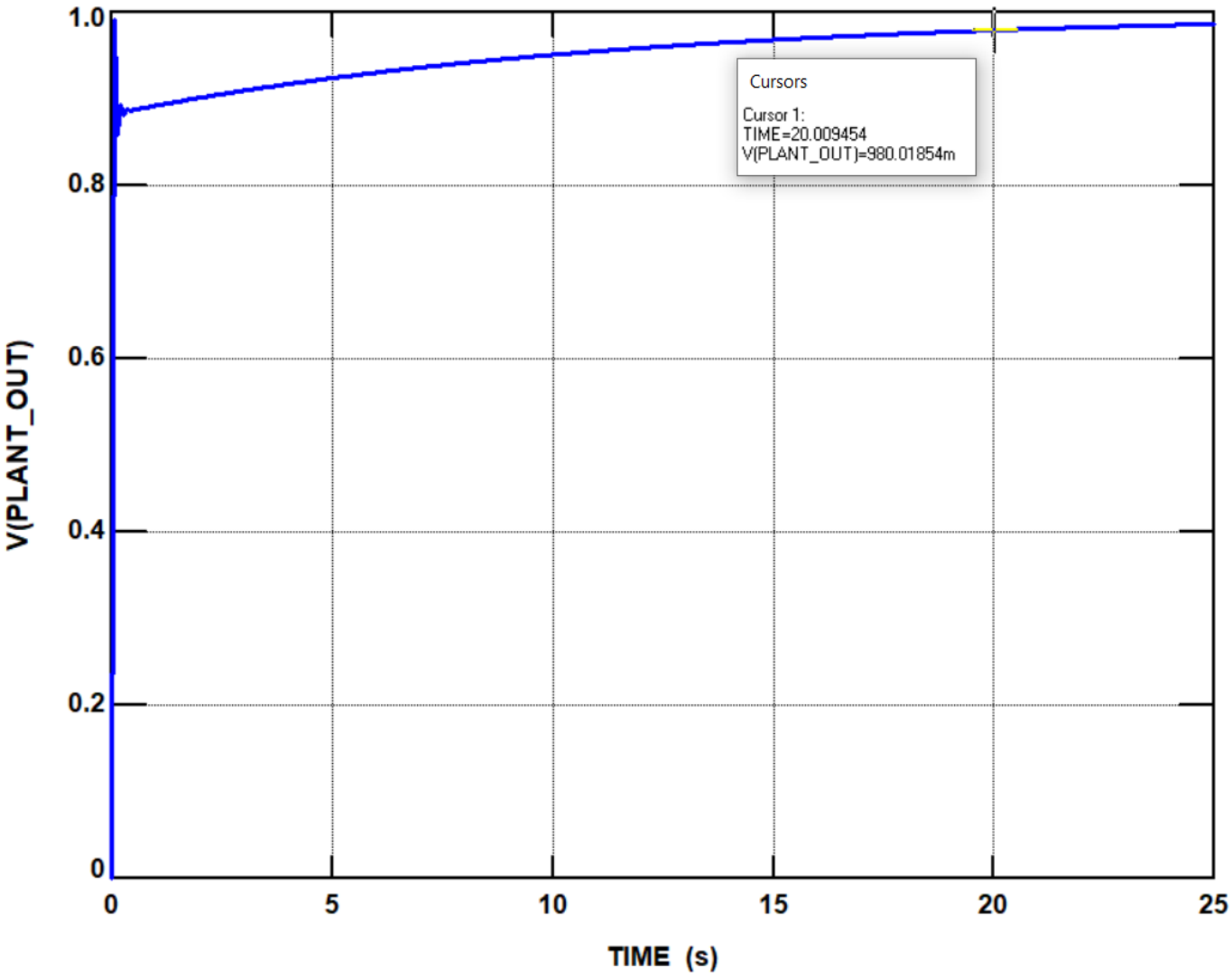


Fig. 27: The step response of the realized circuit. We have result which matches our MATLAB simulation closely. Here, the marker indicates settling time within 2% of final value and is Ts = 20 seconds. Our MATLAB plot had Ts = 19.8 seconds.

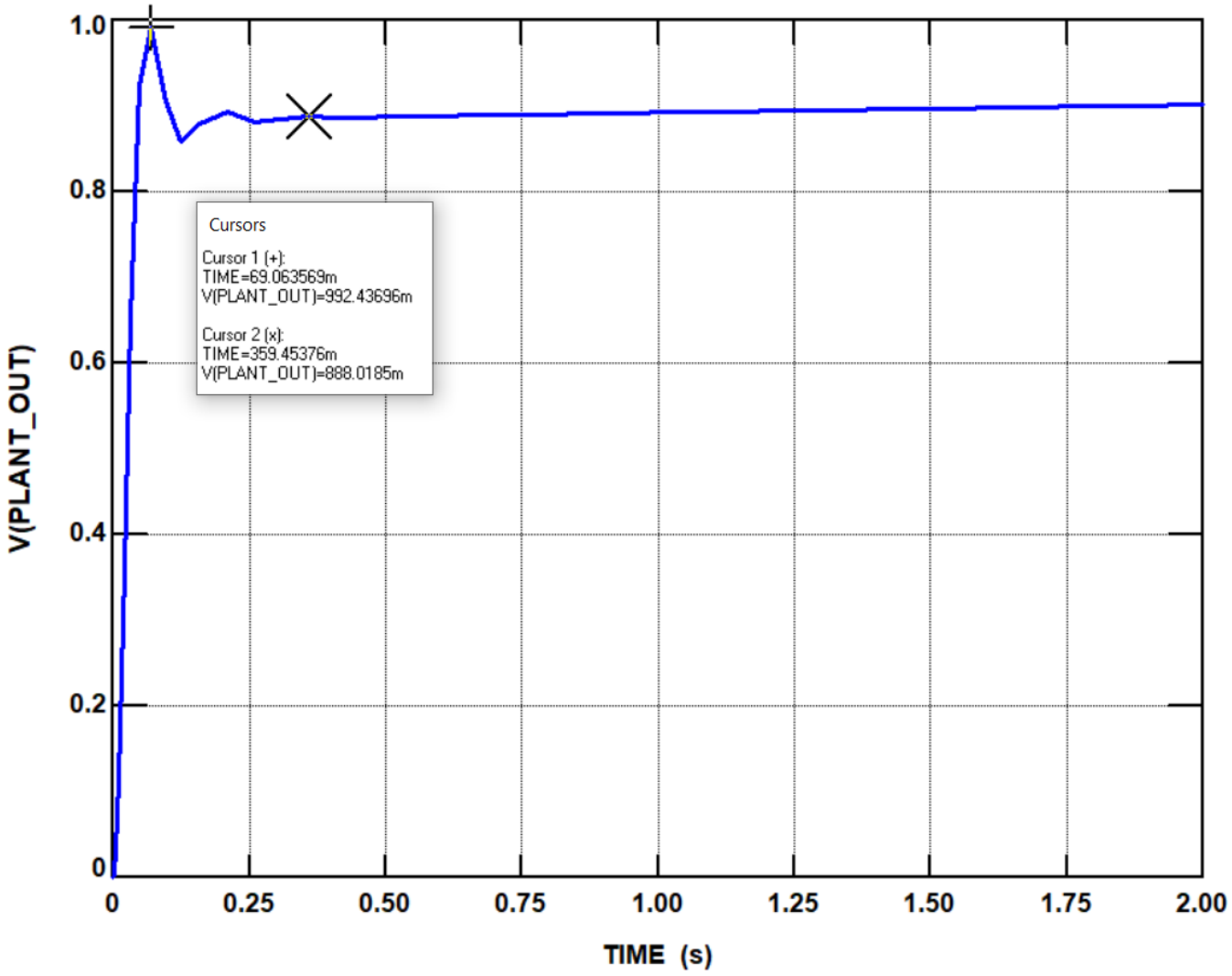


Fig. 28: The time to peak of our Spice simulation is Tp = 0.069 seconds, indicated by 'cursor 1'. This matches closely to the MATLAB output of Tp = 0.061. The difference is likely due to the amount of data points but the result matches closely as is. 'Cursor 1" is a marker for the calculation of %OS which is 11.7% using equation 12 below.

$$\%OS = \frac{C_{max} - C_{final}}{C_{final}} * 100 \qquad \text{EQ (12)}$$

IV.4 PID Tuning

After spending time researching methods of PID control, I was able to find several methods to include manual tuning, Ziegler-Nichols, Tyreus Luyben, Cohen Coon, Astrom-Hagglund, as well as examples of software tuning to include MATLAB's own PID tuning code which uses the syntax 'pidTuner(sys,type)'. After mulling the options over, I decided to go with simply improving the settling time by 90%. I chose this parameter for improvement because the other parameters of %OS, as well as improvement of peak time were given to me by my "customer" and I figured those were features which should be held since they were requested. I revisited slides as well as tuning methods and decided to use a bit of loop tuning of the manual sort, putting calculations to my improvement parameter. Just as above, when designing the PD controller, we can use the equation for Ts to move determine a good location for zero placement for the tuning, I chose to use the zero that was placed with our PI controller at s=0.1 because I read that this parameter could be used to improve transient response of our system which is exactly what I am improving.

I used the equation for Ts to get a zero location for my parameter improvement. I took this parameter and halved it and then took away the amount of the zero already placed in my PI controller which was [s=0.1]. The calculation and results are shown below. My goal was solely to improve TS while keeping all other parameters the same as those were requested parameters.

$T_{s\,(PID)}$ = 19.8 seconds (within 2% of FV), 19.8*0.1 = 1.98s = $T_{s\,(NEW)}$

$$Re = \frac{4}{T_{S(NEW)}} = 2.02 \qquad \text{EQ (13)}$$

Since a new zero was not needed and I wanted to use the current zero at [s = -0.1], which was our PI zero, I split the difference and subtracted the current pole value of 0.1 for a new pole location of [s= -0.91]. This zero was close and yielded a new settling time of Ts = 2.02s. Since I wanted to speed this parameter up, I knew that the zero needed to move further away from the imaginary axis so the new zero was placed at [s = -0.925]. This gave the desired settling time of Ts = 1.98s.

Root Locus in MATLAB of Complete System (plant, controller with tune):

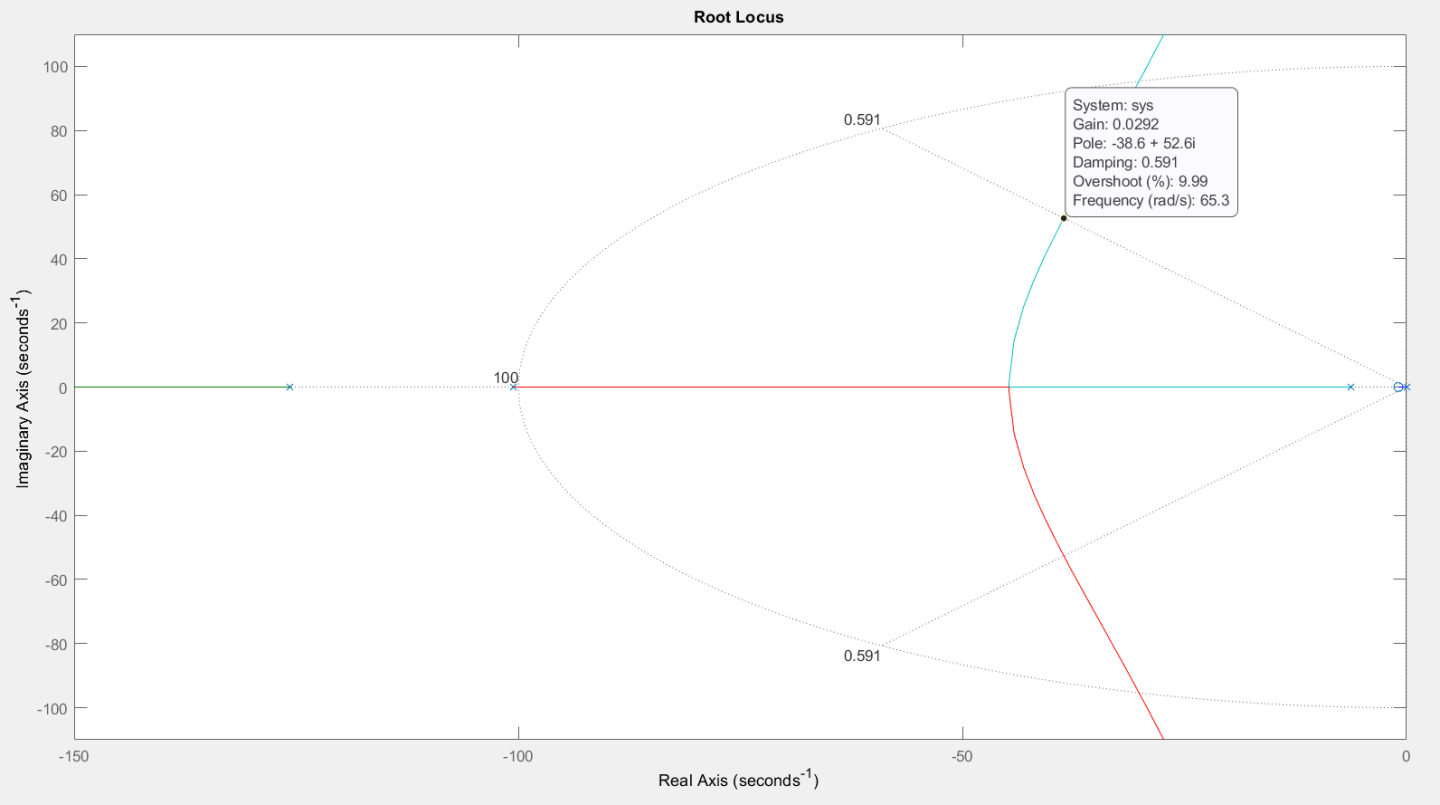


Fig. 29: Our Root Locus remains unchanged as desired. There are marginal changes to the gain and overshoot of the Tuned PID but it appears that we have kept the performance parameters requested initially. Those being 10% OS giving us Zeta = 0.591as well as the improved peak time Tp = 0.061s.

Step Response in MATLAB:

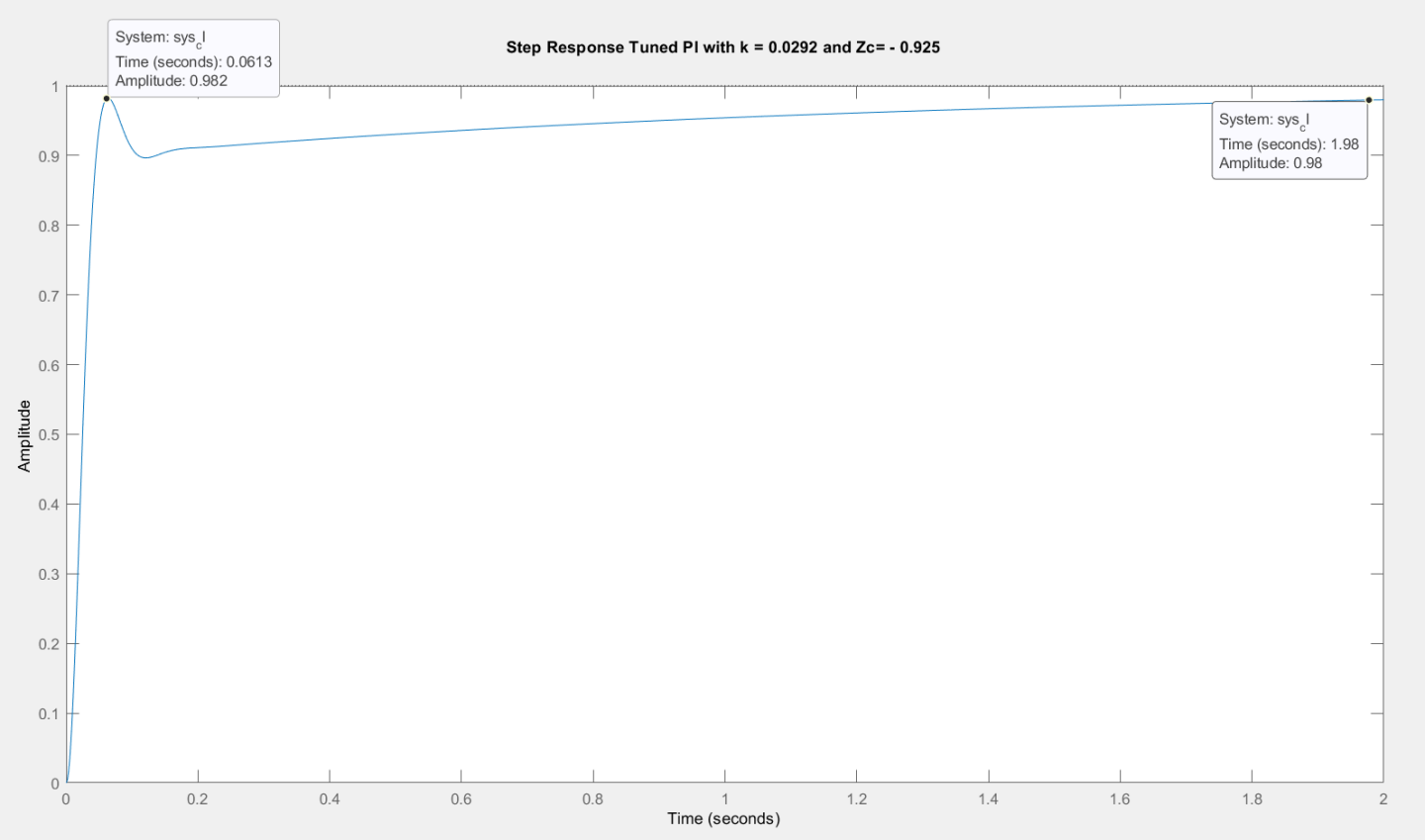


Fig. 30: The settling time has improved by our target of 90%. We can see a new settling time of Ts = 1.98 seconds and we have maintained our peak time performance of Tp = 0.061s, which was an assigned parameter. From the root locus, we can confirm that %OS, another given parameter remains the same at a value of 9.99%OS.

Editor - C:\Users\rober\OneDrive\Desktop\PS8_Root_locus_Code.m

PS8_Root_locus_Code.m

```
1 -    clc, clear, close all
2 -    s = tf('s');
3 -    sys = ((79374)*(s+0.925)*(s+254.43))/(s*(s+6.283)*(s+100.531)*(s+125.664));
4      %bode (sys)
5      %pzmap (sys)
6      %rlocus (sys)
7      %axis([-2 0 -2 2])
8      %zeta = 0.591;
9      %wn = 100;
10     %sgrid (zeta,wn)
11 -   k = 0.0292;
12 -   sys_cl = feedback (k*sys, 1)
13 -   step (sys_cl)
14     %step(sys)
```

Fig. 31: Code snippet for tuned PID. The gain was adjusted to match the root locus gain and the PI zero has been changed to s = -0.925

PID Verification in SPICE:

Our new tuned compensator function is:

$$G_{PID(Tuned)}(s) = \frac{0.0292(s+0.925)(s+254.43)}{s} \qquad \text{EQ (14)}$$

Calculation of Resistor Capacitor Values:

Using equation 14 above, we are able to realize physical circuitry to model our system in SPICE. The table below will show the calculations:

Table 7: Gain Calculation				
Component	Equation	Gain	(Resistor/Capacitor) $_s$	(Capacitor/Resistor) $_f$
P	$K = \frac{R_f}{R_s}$	K_1 = 7.457	10kΩ	74.57kΩ
I	$K = \frac{1}{RC}$	K_2 = 6.872	100kΩ	1.455uF
D	$K = RC$	K_3 = 0.0292	2.92uF	10kΩ

Table 8: Gain Calculations for Op Amp Builds in SPICE

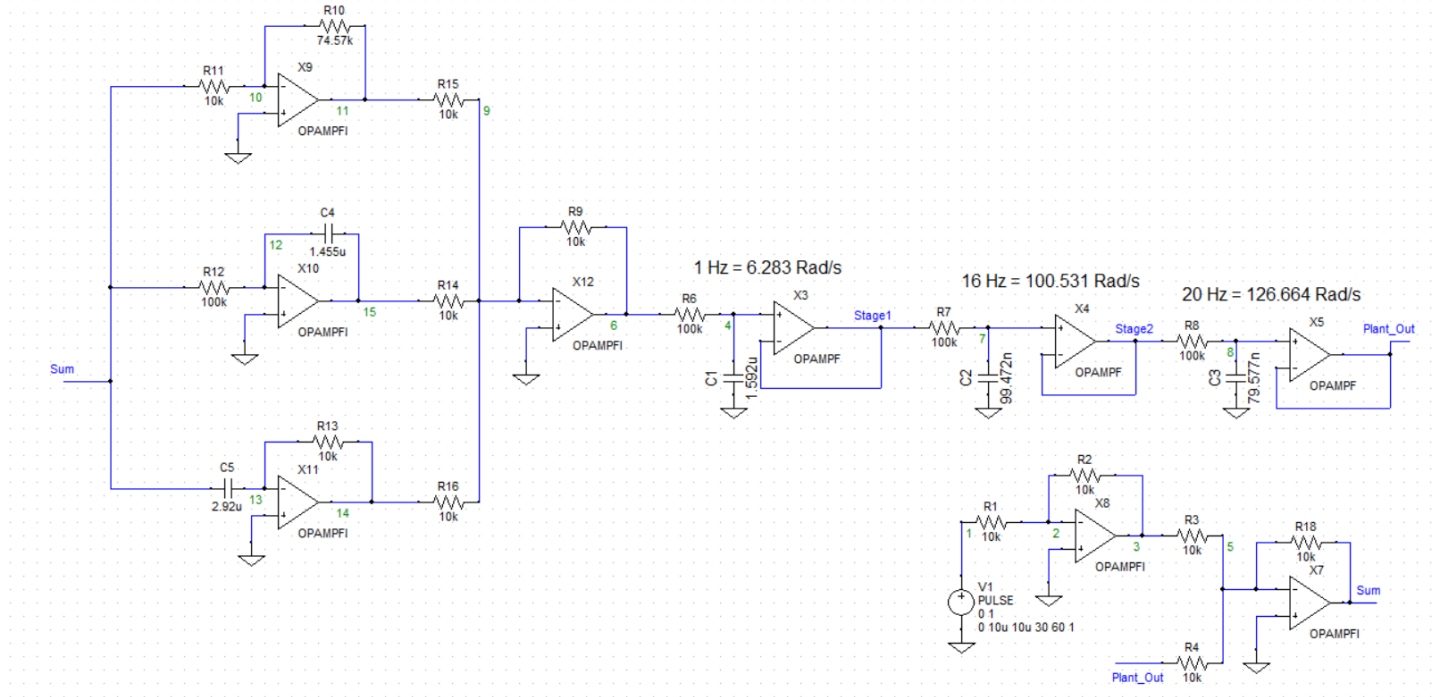


Fig. 32: The tuned PID circuit in Spice

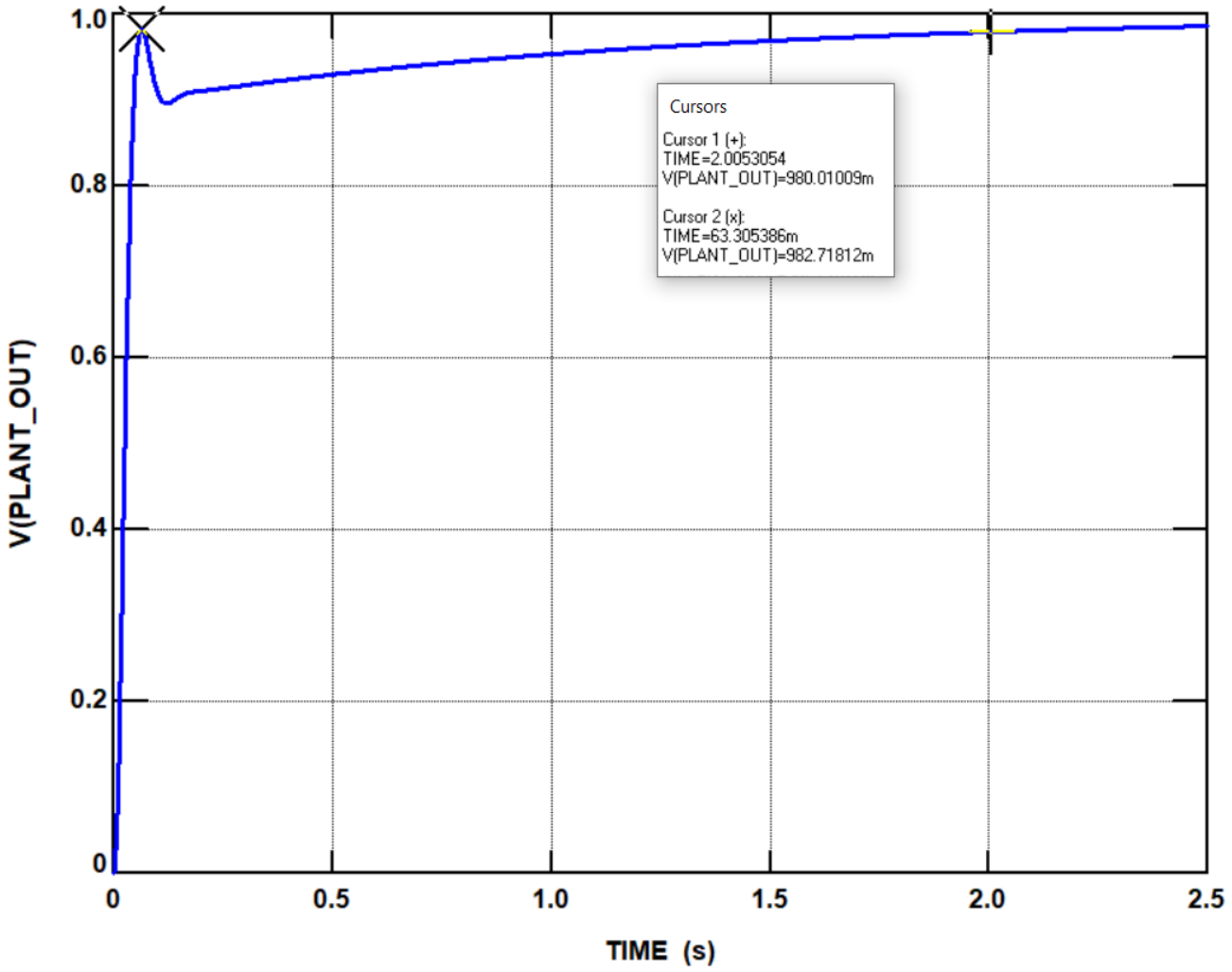


Fig. 33: We have managed to maintain our peak time improvement with little change. Tp = 0.063s. Our improved settling time is also very close to our MATLAB value of Ts = 1.98s. Here in the realized circuit, Ts = 2s. Steady state error is still maintained and complete steady state value is reached at an improved time of 5.75 seconds as shown in the next figure.

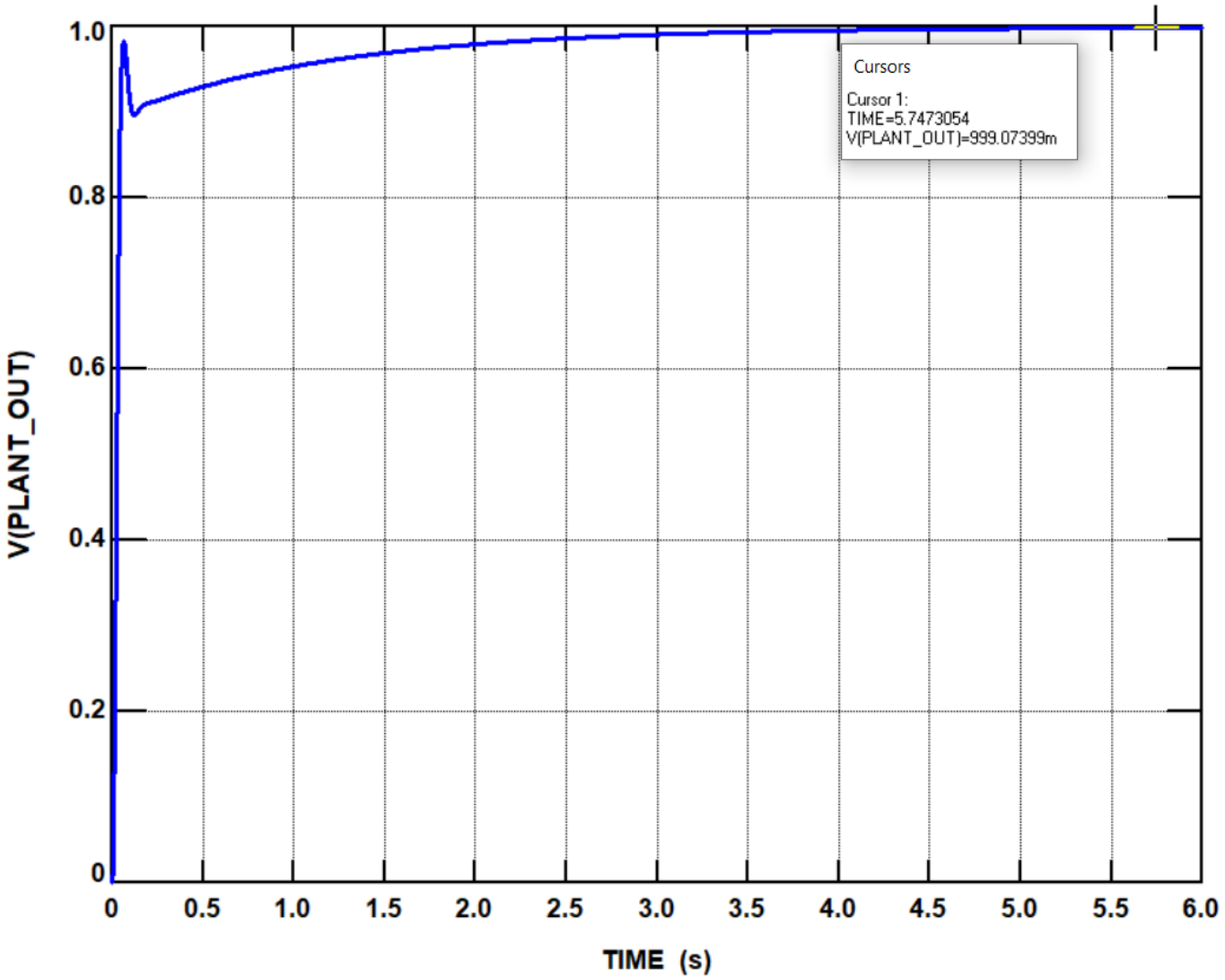


Fig. 34: Improvement of time to complete steady state after tuning, our original time to steady state was t = 39.8s with our PID controller. After tuning, we have decreased it significantly. Our new zero error time is t = 5.75s.

IV.5 PID Performance Comparison

MATLAB: Original PID (top), Tuned PID (bottom)

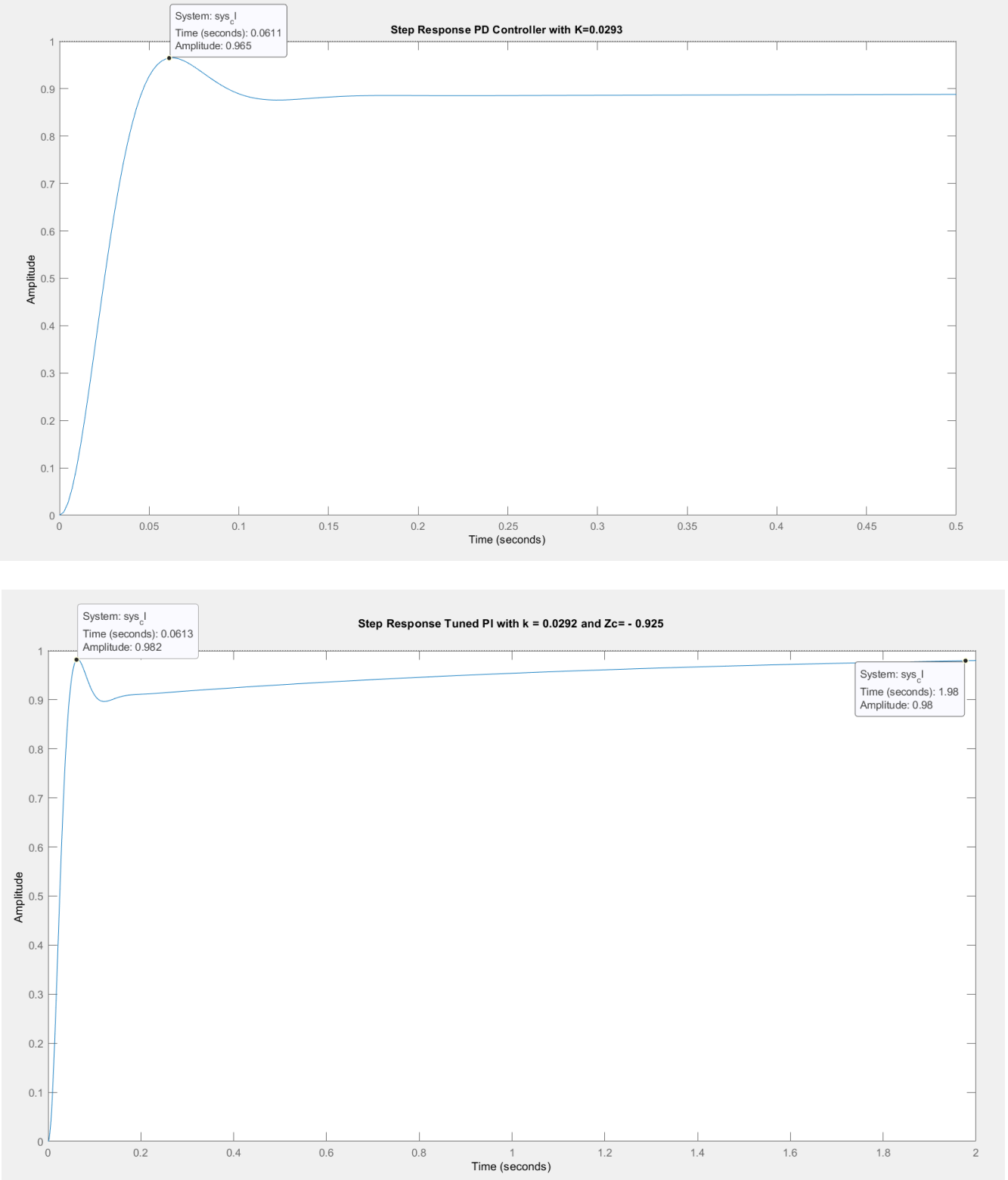


Fig 34 a/b: The PID before tuning (above) and after tuning(below). The steady state response is improved with the new target settling time of Ts=1.98 seconds. The %OS taken off of the plots are 9.65% for PID and approximately 8% for the tuned PID.

SPICE: Original PID (top), Tuned PID (bottom)

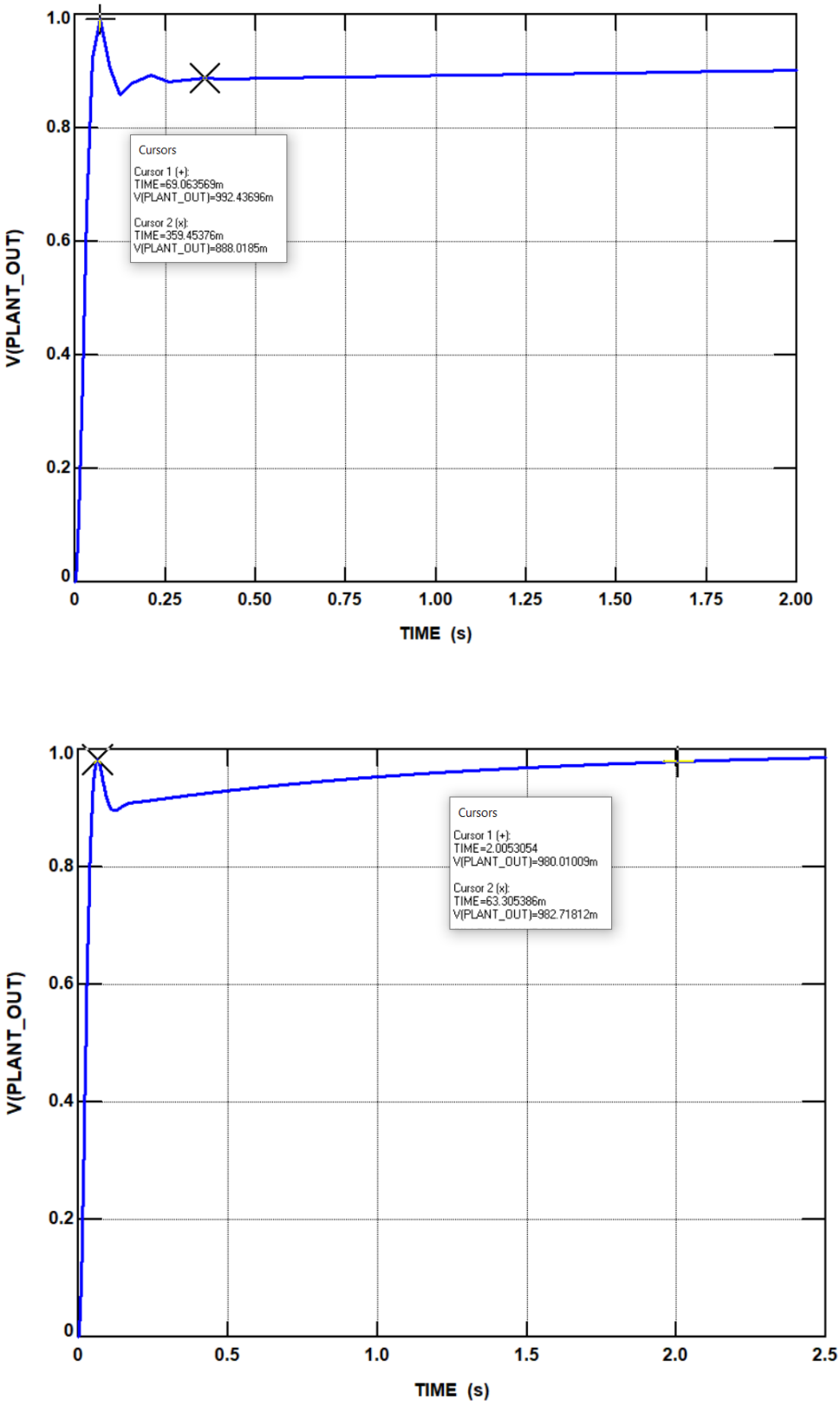
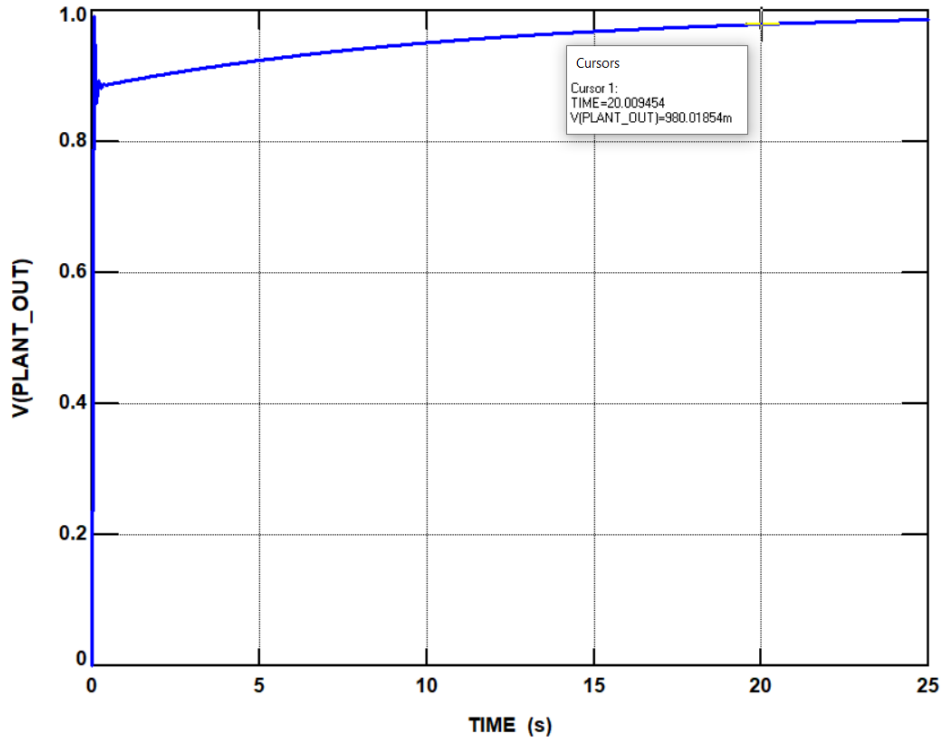


Fig. 35 a/b: The SPICE comparison shows our target improvement has been met with 90% improvement in Ts. Our peak time has been shifted, but only marginally so. Our target for the PID was Tp=0.061 which was met with the PID giving us Tp=0.069. Our tuned PID gives us Tp = 0.063 with similar magnitude and remains close to the desired value.



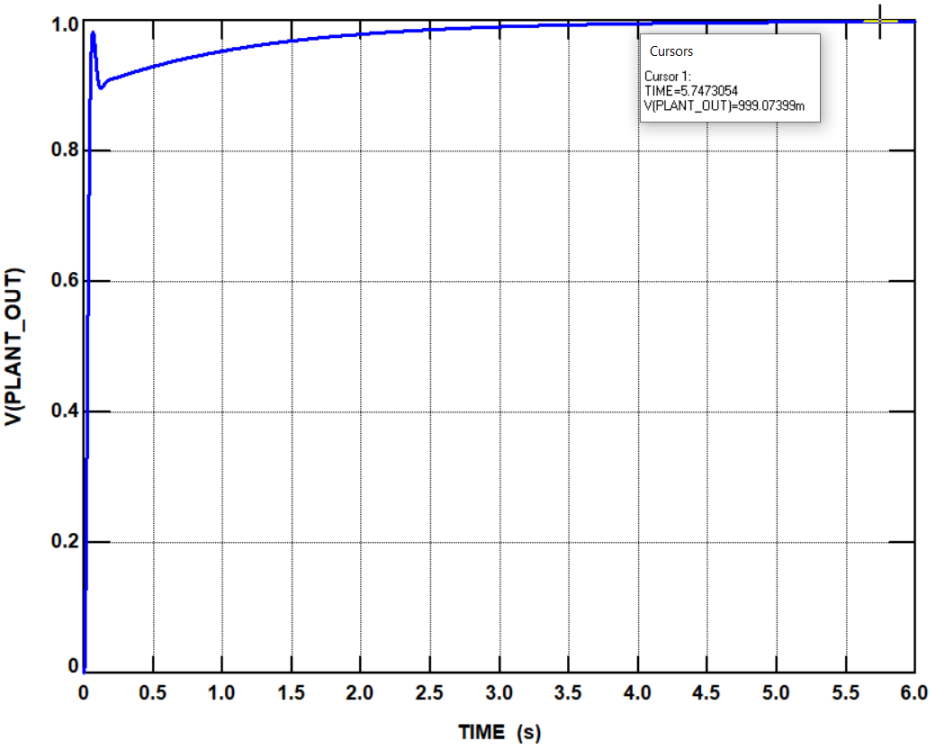


Fig. 36 a/b: This is the longer view of the two systems, PID above and PID tuned below. Our original system took 39.8 seconds to reach the final zero steady state error value while the tuned system completes this task in 5.74 seconds.

Table 9: PID Performance Comparison				
	MATLAB, Original	**MATLAB, After Tuning**	**SPICE, Original**	**SPICE, After Tuning**
T_s(w/in 2% of FV)	19.8	1.98	20	2
T_P	0.0611	0.0613	0.069	0.063
%OS	9.04%	8.62%	11.7%	8.04%

Table 9: The performance comparison shows that we have been able to improve the PID controller with little change in the parameters that we designed into the controller as requested. The changes effected were that we had a decrease in %OS as well as a slight deviation in our peak time performance. The peak time was slowed but only by 2 thousandths of a second while %OS decreased by 0.5% in MATLAB and 2% in the realized circuit. If these parameters were extremely important, we could increase gain in order to get the overshoot closer to 10%.

V. Conclusion and Discussion

V.1: Advantages and Disadvantages of PID

PID controllers are advantageous because they can be designed and implemented easily with little information about the system. They offer control over a range of adjustments to include peak time, settling time, damping ratio/percent overshoot, steady state error, and stability through gain adjustments. Once implemented, they can be further tuned, but this tuning ability has limitations just as all things engineering do.

PID controllers are a feedback control system for error improvement, but they have limitations based on linearity and time. With respect to time, the PID controller uses each control type to react to a specific time mode, proportional is present time adjustments, integral is past time adjustments, and derivative predicts future time adjustments based on the rate of change with respect to time that it reads from the system. If the time constant is too large, PID controllers do not perform well or may perform with inconsistent results. If there is any non-linearity in the system, for example a warm up of a system with two separate behavior models, then a single PID controller will have variable performance.

V.2:

When conducting research for this project, I used three sources. They were the course slides pertaining to design via root locus, the text book (which did not offer much in tuning but helped give deeper understanding), and internet sites related to Electrical Engineering (open-source college papers and circuits/electronics information sites on the subject matter). I found that tuning a PID controller can be done several ways and that each has their own pros and cons. The advantages rely on whether the system needs to be taken offline or if software is used while disadvantages take the form of costs, training, or process upset (in manufacturing). The first three: Manual, Ziegler-Nichols, and Cohen-Coon all use the method of increasing gain up until the system oscillates with constant frequency (find where the system behaves ‘undamped’) and then using a set of mathematically proportional constants to scale the gain of each (P-I-D) controller.

Another method is by use of tuning software like MATLAB. This can be fast if the system has a loop process that is also fast, say less than a minute. Otherwise, it is suggested that mathematical operations are performed as software tuning could take longer than patience allows.

I did not resort to using the algorithm type tuning of Ziegler, or Cohen and instead went with a manual approach, deciding the parameter to improve, researching the options to change (zeros or poles), then calculating a guess finalizing with a confirmation and or subsequent repeats of the process. This approach was quick and took only three tries to meet the desired results. I thought about using Ziegler, but was not sure how to appropriately get through step one of the process without completely changing the system. My method was quicker, but only because I was allowed to decide a single parameter to change and knew how to get there

spatially. I was able to get the results that I desired rather quickly without completely changing the system and while maintaining the improvements made leading up to the tuning process. The results are below.

V.3 Compare/Contrast MATLAB v. SPICE step response results

Table 10: MATLAB v SPICE Performance Comparison (tuned system)			
	MATLAB	SPICE	%Difference
T_S	1.98	2.00	1.01%
T_P	0.0613	0.063	2.77%
%OS	8.62%	8.04%	7.21%

Table 10: We can see that the tuned PID performs with some slight differences in each system. From MATLAB to SPICE, we have increases Ts and Tp, and a decrease in %OS.

I believe the differences outlined in Table 10 are due to mathematical interpretation with regards to each system. Namely, the number of significant figures carried from one system to the next. Additionally, I believe that circuitry has an element of time differentiation to it which lends itself to a more realistic model than MATLAB may give. Specifically, our MATLAB work involved numerical manipulation on the Plant and controllers only, while the SPICE simulation involved a physically realized system with a time model of active and passive components which had gains calculated and carried over with 4 significant figures. It is my belief that the SPICE model therefore offers a more realistic simulation of what you could expect from your system once realized.

V.4 How do you believe that a digital implementation of a PID controller may differ from this analog implementation? Think about this. Be specific.

As stated above, I believe that the digital implementation is a closer approximation to reality because of the physical components operating in the time domain both passively and actively. In the digital model, there is no time domain representation of the active circuit components which change with respect to time. The analog representation has this aspect attached to it and can be built a few different ways, so we have both more realistic approximation as well as an aspect of modularity with respect to the way the system can be realized by the designer.

V.5 What did you learn from this Final Project? Please don't say, "How to design a PID controller," because we already know that.

I have not taken electronics 2 yet. So, the biggest takeaway from this project was finally learning how to realize a mathematical model with physical parts. It is the aspect of this major which I have been waiting to get my hands on because it is the most fascinating to me. We have taken a transfer function which is just a

mathematical model, done work and analysis on it in order to get the desired behavior, and then worked out and built a physical representation which behaves as we expected. Fascinating, to say the least.

*****HAND CALCULATIONS FOLLOW*****

EE4360 FINAL PROJECT

R. GLOW

ASSIGNED PARAMETERS:

ID	P1	P2	P3	%OS TOT	TP IMPROVEMENT	STEP INP. GRAPH
3205	(6.283 RAD/S)	(100.531 RAD/S)	(125.664 RAD/S)	10	20%	ZERO

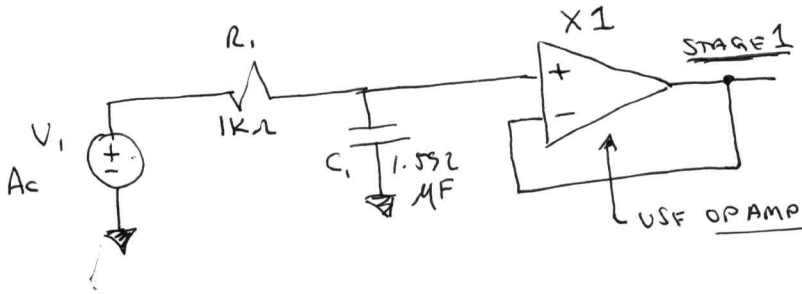
*VALUES: [R ->> 1KΩ ≤ R ≤ 100KΩ] [C ->> 1E-9F ≤ C ≤ 100E-6F]

POLE 1: $\omega = 6.283$ RAD/S $\quad \omega_c = \frac{1}{RC}$ LET: ~~R = 1KΩ~~ $R_1 = 100K\Omega$

$C = \frac{1}{\omega_c R} = \frac{1}{(6.283\ R/S)(100K\Omega)} =$ ~~159.16 MF = C~~ * NOT IN ACCEPTABLE RANGE!

↳ $C_1 = 1.592$ MF

$\omega_c = 2\pi f_c \quad \therefore \quad f_c = \frac{\omega_c}{2\pi} = \frac{6.283\ RAD/S}{2\pi} \Rightarrow 0.99997 \cong$ 1 HZ = f_c



(OP AMP FNCTN)

TOP SPICE PART: XOPAMPF OR XOPAMPFI

* AC SWEEP
100 PTS/DEC.) (SWEEP O/POLE f_c INSIDE RANGE)
↳ 10MHz -> 100 Hz COVERS THIS

PRINT DATA: VdB (STAGE 1), Vp (STAGE 1)
↑ THIS IS THE NAME ASSIGNED TO THE OUTPUT

* USE MARKERS FOR ω_c & PHASE PLOT

POLE 2: $\omega_c = 100.531$ RAD/S. LET: $R_2 = 100K\Omega$

$\omega_2 = 2\pi f$

$f_2 = \frac{\omega_2}{2\pi} = \frac{100.531}{2\pi} = 16\ Hz = f_2$

$C = \frac{1}{\omega_c R} = \frac{1}{(100.531\ RAD/S)(100K\Omega)} =$ 99.472 nF = C_2 ✓

POLE 3: $\omega_c = 125.664$ RAD/S LET: $R_3 = 100K\Omega$ $f_3 = \frac{\omega_3}{2\pi} =$ 20 Hz = f_3

$C = \frac{1}{\omega_c R} = \frac{1}{(125.664)(100K)} =$ 79.577 nF = C_3

EE 4360 FINAL PROJECT

↳ $B(s) = \dfrac{?}{(s+6.283)(s+100.531)(s+125.664)}$ ——> MUST WRITE IN STD FORM. HOW? WRITE S/T ~~SOME~~ POLE VALUE IS DRIVEN TO (1)!

$\therefore G(s) = \dfrac{(6.283)\left(\frac{s}{6.283}+1\right)(100.531)\left(\frac{s}{100.531}+1\right)(125.664)\left(\frac{s}{125.664}+1\right)}{}$

↳ $6.283 \cdot 100.531 \cdot 125.664 = \boxed{79,374}$

HENCE: $\boxed{\dfrac{G(s)}{\cancel{T(s)}} = \dfrac{79,374}{(s+6.283)(s+100.531)(s+125.664)}} \rightarrow \left[s^3 + 232.48s^2 + 14.054E3s + 79,374\right]$

*** BEWARE WHEN COMPARING SPICE → MATLAB B/C MATLAB HORIZ. AXIS IS IN (ω) WHILE SPICE HORIZ. AXIS IS IN (f_c) *

↳ CAN CHANGE MATLAB TO Hz ???

$\%OS = 10\%$ (TGT) NEED: ζ & $\zeta = \dfrac{-\ln(\%OS)}{\sqrt{\pi^2 + \ln^2(\%OS)}}$

① UNCOMPENSATED

$\boxed{\zeta = 0.59116}$

AT $\zeta = 0.591$: $\boxed{K = \dfrac{5.5}{\cancel{5.44}}}$; $\boxed{s = -32.9 \pm j\,44.8}$; $\%OS = \cancel{7.58}\ 10\%$

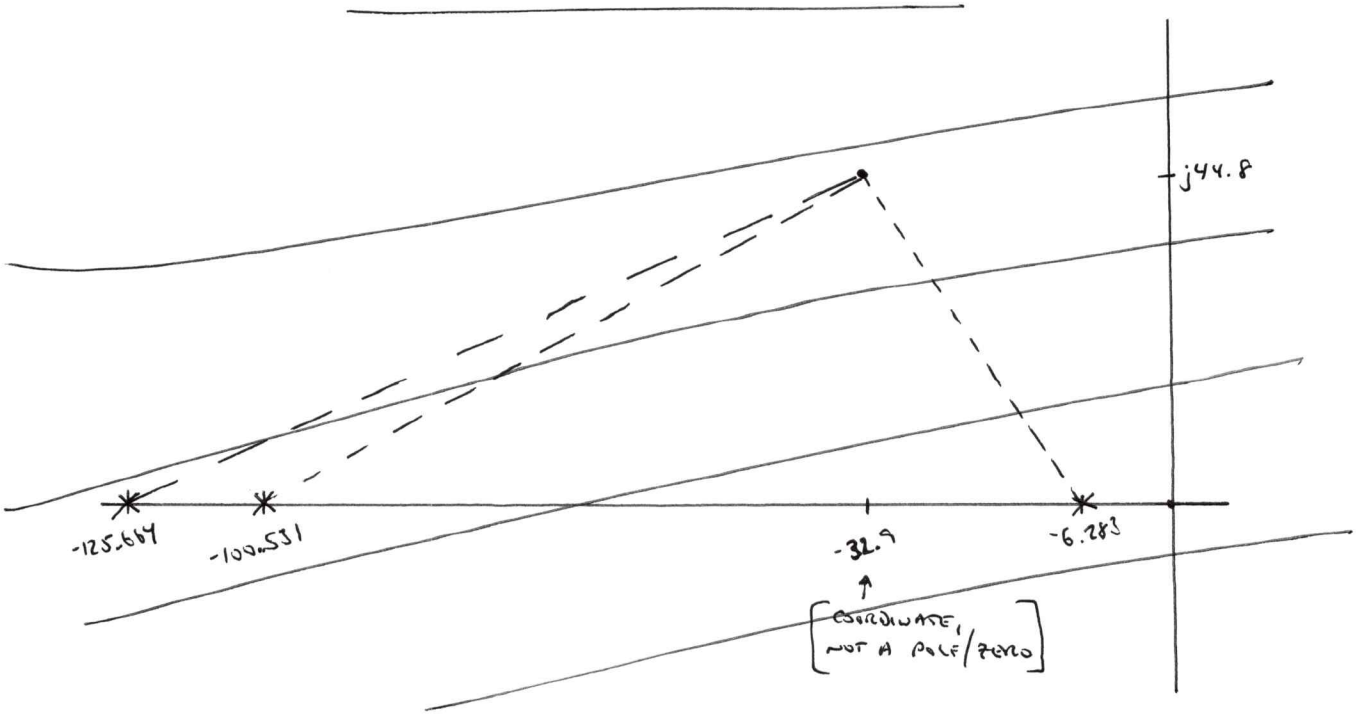
$\omega_n = 55.6$

$T_p = \dfrac{\pi}{|Im|} \Rightarrow \dfrac{\pi}{44.8} = \boxed{0.07\,s = T_p}$

$T_s = \dfrac{4}{|Re|} \Rightarrow \dfrac{4}{32.9} = \boxed{0.122\,s = T_s}$

EE4360 FINAL PROJECT [P-I]

->



=> For: $\zeta = 0.591$ DOMINANT POLE IS $[s = -32.5 \pm j\,44.8]$ $[K = 5.5]$

NEED: 10%OS ✓ [20% IMPROVEMENT IN T_p] *

② [PI]

ADD: $[P = \frac{1}{s} \quad | \quad z = -0.1]$ * AT $K = 5.5$ THIRD POLE AT [-167]*

-> ROOT LOCUS W/NEW P-I COMPENSATOR IS ~~UNCHANGED~~. ALMOST UNCHANGED.

$K = 5.49$, $\omega_n = 55.5$, $\zeta = 0.591$, $[P = -32.8 \pm j\,44.8]$

-> TO W/IN 2% (0.98 OF UNITY) $[T_S = 23.7s]$ $G_{PI}(s) = \left[\frac{(s+0.1)}{s}\right]$

* SYSTEM NOW HAS $[T_p = 0.076]$ AT $A = 0.528$

& $[e(\infty) = 0]$ BUT $T_S = 23.7s$ TO 98% & $\cong$ 45s TO UNITY

$$\left[G_{PI}(s) = \frac{79,374\,(s+0.1)}{s\,(s+6.283)(s+100.531)(s+125.664)}\right]$$

③ [PD]

-> PD: $\left\{T_S = \frac{4}{|Re|} \quad T_p = \frac{\pi}{|Im|}\right\}$ $T_p = 0.076(0.8)$ $T_S = 23.7$ ~~* 0.8 = 18.96~~

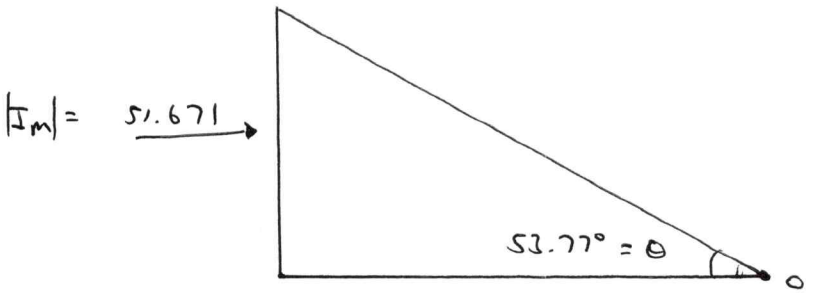
$[T_{p_{new}} = 60.8E{-3}]$ *

EE 4360 Final Project (P1)

~~$\rightarrow T_{P(NEW)} = 18.856 \quad \therefore \quad 18.56 = \frac{\pi}{I_m}$~~

~~$I_m = \pi / 18.56 = 165.7E-3 = I_{m\,NEW}$~~

$T_{P(NEW)} = 60.8E-3$ $\quad |I_m| = \frac{\pi}{60.8E-3} = \boxed{51.671 = |I_m|_{NEW}}$ ✓



$\zeta = 0.591$ $\quad \theta = \cos^{-1}(\zeta)$

$\theta = 53.77°$

$\tan\theta = \frac{I_m}{R_e}$

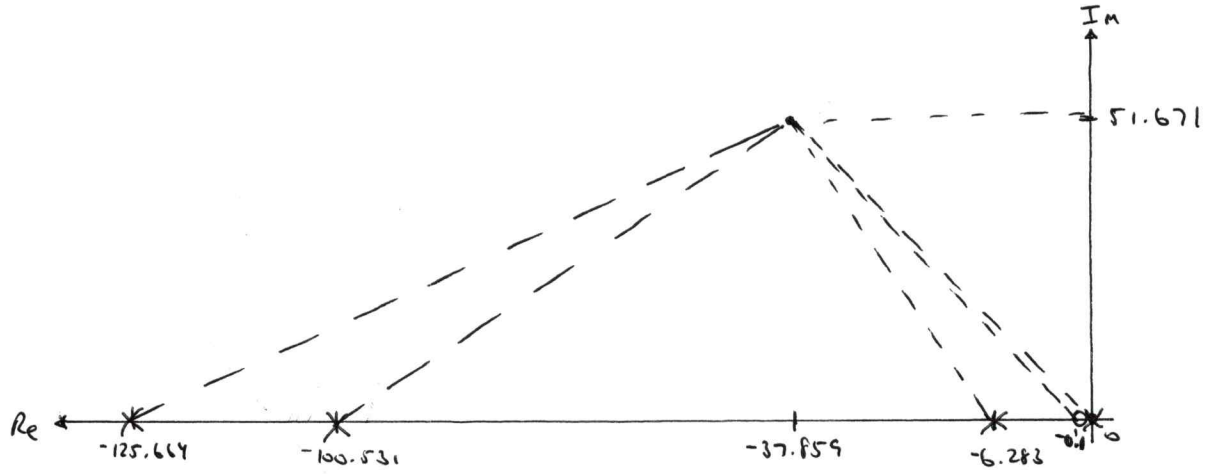
$R_e = -37.859$

$R_e = \frac{I_m}{\tan\theta} = \frac{51.671}{\tan(53.77°)}$

$\boxed{R_e = 37.859}$

$\therefore \boxed{S_{NEW} = -37.859 \pm j\,51.671}$

↳ NOT ON Root Locus ∴ Following Design: ↓



Poles: $S = 0, -6.283, -100.531, -125.664$ $\quad$ Zeros: $S = -0.1$

$S(0)$: $\quad 180 - 53.770 = \boxed{126.23° = \angle S(0)}$

$S(-6.283) \quad \theta = \tan^{-1}\left(\frac{51.671}{31.576}\right) = 58.57° \rightarrow \phi = 180 - 58.570 = \boxed{121.430° = \angle S(-6.283)}$

* $S(-100.531) \quad \phi = \tan^{-1}\left(\frac{51.671}{62.672}\right) = \boxed{39.504° = \angle S(-100.531)}$

* $S(-125.664) \quad \phi = \tan^{-1}\left(\frac{51.671}{87.805}\right) = \boxed{30.476° = \angle S(-125.664)}$

Zero: $s = -0.1$

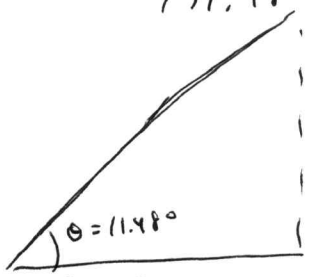
$s(-0.1) \Rightarrow \theta = \tan^{-1}\left(\frac{51.671}{37.759}\right) = 53.842° \Rightarrow \phi = 180 - \theta = \boxed{126.160° = \angle s(-0.1)}$

$\theta = \left(\sum \theta z_i\right) - \left(\sum \theta p_i\right)$

$\theta = (126.160°) - (126.230° + 121.438° + 39.504° + 30.476°)$

$\theta = \cancel{(126.160°)} \quad \boxed{-191.48° = \theta}$

$191.48° - 180° = \boxed{11.48°}$ NEEDED FOR COMPENSATOR



$\tan\theta = \frac{Im}{Re} \therefore Re = \frac{Im}{\tan\theta}$

$Re = \frac{51.671}{\tan(11.48°)} = \boxed{254.43 = Re}$

254.43

$\therefore \quad z_2 = (s + 254.43) = \boxed{G_{PD}(s) = (s + 254.43)}$

$\therefore \left\{ G_{PD}(s) = \frac{79,374\,(s+0.1)(s+254.43)}{s\,(s+6.283)(s+100.531)(s+125.664)} \right\}$ AND AWAY WE GO!

(3) PD

-> MATLAB Root Locus: $\begin{cases} \text{At } \zeta = 0.591, \boxed{s = -38.9 \pm j\,53.1} \text{ ACTUAL,} \quad \boxed{s = -37.859 \pm j\,51.671} \text{ TARGET FOR PD} \\ K = 0.0293 \\ \omega_n = 65.8 \end{cases}$ w/ 10% OS

DIFFERENCE: $\boxed{-1.04 \pm j\,1.43}$

-> STEADY STATE: $T_p = 0.061$ s w/ $A = 0.965$

$T_s = 19.8$ s TO GET w/IN 2%, $A = 0.98$

$$\Rightarrow G_{PID}(s) = \frac{79.374\,(s+0.1)(s+254.43)}{s(s+6.283)(s+100.531)(s+125.664)}$$

PLANT = G(s)

$$G(s) = \frac{79.374}{(s+6.283)(s+100.531)(s+125.664)}$$

$K_{PID} = 0.0293$

ADD: $G_{PID}(s) = \frac{79.374\,(s+0.1)(s+254.43)}{s}$

$$= \frac{0.0293(79.374)(s^2+254.53s+25.443)}{s}$$

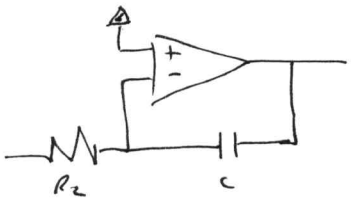
$\Rightarrow K_3 s \Rightarrow \frac{0.0293\,s}{s} =$ **$0.0293 = K_3$** s

$I \Rightarrow \frac{K_2}{s} \Rightarrow (0.0293)\frac{25.443}{s} = \frac{0.7455}{s} = \frac{K_2}{s}$ ∴ **$K_2 = 0.7455$** /s

$P \Rightarrow K_1 \Rightarrow \frac{254.43}{s}(0.0293) = \frac{7.455}{1} =$ **$K_1 = 7.455$**

PI GAIN: $\left[K_2 = \frac{1}{R_2 C}\right]$ $K_2 = 0.7455$ IF **$R_2 = 100\,k\Omega$**

$\frac{1}{K_2 R} =$ **$C_2 = 13.414\,\mu F$**

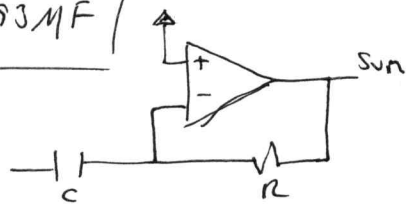


SUMMING RESIST

P GAIN: $\left[K_3 = RC\right] \rightarrow K_3 = 0.0293$ $C = \frac{K_3}{R}$

$R = 10k\Omega$

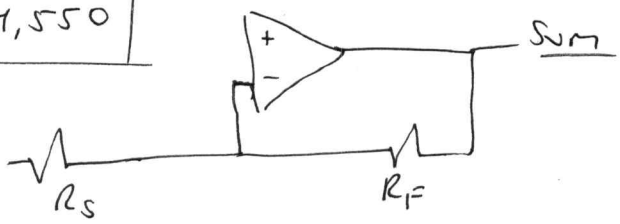
THEN $C = 2.93\,\mu F$



P GAIN: $\left[\frac{R_F}{R_S} = K_1\right]$ $K_1 = 7.455$

$R_F = K_1 R_S \rightarrow$ **$R_S = 10{,}000$**

$R_F = 74{,}550$



$* 0.(0.92) \Rightarrow 1.88 = T_s$

EE 4360 FINAL PROJECT

$T_s = 15.8\ s$ (TO 4/N 2% OF FV)

I WANT 50% IMPROVEMENT OF T_s

$100 - 90 = 10 \Rightarrow 15.8(0.1) = \boxed{1.58\ s = T_{s\ NEW}}$

$T_s = \frac{4}{|Re|} \Rightarrow Re = \frac{4}{T_s} = \boxed{2.02}$ * {WE CHANGE ζ, ω_n OR T_p

→ MAGNITUDE OF Re, BUT WE HAVE A ZERO AT $s = -0.1$ THAT WE CAN MOVE:

$\therefore 2.02/2 = 1.01$ & $1.01 - 0.1 = \boxed{0.910} \Rightarrow s = -0.91$

FOR ($s = -0.91$): MATLAB $\Rightarrow$ $2.02\ s = T_s$ * NOR $s = -0.92$ $\boxed{T_s = 1.58\ s}$

*BUT LOCUS REMAINS UNCHANGED ✓ ∴ SAME $\%OS$, ζ, ω_n & Poles

$$G_{PID(TUNED)}(s) = \left[\frac{75,374\,(s+0.52)(s+254.43)}{s(s+6.283)(s+100.531)(s+125.664)}\right]$$

$$\left[G_{C(TUNED)}(s) = \left[\frac{0.0292\,(s+0.52)(s+254.43)}{s}\right]\right]\left[\begin{array}{l} K = 0.0292 \\ s = -38.6 \pm j\,52.6 \\ \zeta = 0.591 \\ \%OS = 5.99 \\ \omega_n = 65.3 \end{array}\right]$$

$$\boxed{G_{C(TUNED)}(s) = \frac{0.0292\left(s^2 + 255.35\,s + \overset{235.35}{\cancel{235.08}}\right)}{s}}$$

$D = K_3 s = \boxed{0.0292 = K_3 = D}$

$I = \frac{K_2}{s} = (0.0292)\frac{235.35}{s} = \boxed{\underset{I}{6.872} = K_2}$

$P = K_1 = (0.0292)\frac{255.35\,s}{s} = \boxed{\underset{P}{7.457} = K_1}$

EE 4360 FINAL PROJECT — R. Colón

$$\left[G(s) = \frac{75,374}{(s+6.283)(s+100.531)(s+125.664)} \right] - \text{PLANT}$$

$$\left[G_{PID}(s) = \frac{0.0293\,(s+0.1)\,(s+254.43)}{s} \right]$$

$$\rightarrow \left[G_{PID}(s) = \frac{0.0293\,(s^2 + 254.535 + 25.443)}{s} = G_c(s) \right]$$

$$G_c(s) = k_1 + \frac{k_2}{s} + k_3 s \quad \Rightarrow \quad k_1 = P \;,\; k_2 = I \;\; k_3 = D$$

$$= k_3\left(s^2 + \frac{k_1}{k_3}s + \frac{k_2}{k_3}\right) \quad \therefore \quad D = k_3 = 0.0293$$

$$I = k_2 = 0.7455$$

$$P = k_1 = 7.455$$

$$\therefore \left[G_c(s) = \frac{\overset{D}{0.0293}s^2 + \overset{P}{7.455}s + \overset{I}{0.7455}}{s} \right]$$

FE 4350 Final Project

Gains

D => $K_3 = 0.0292$

I => $K_2 = 6.872$

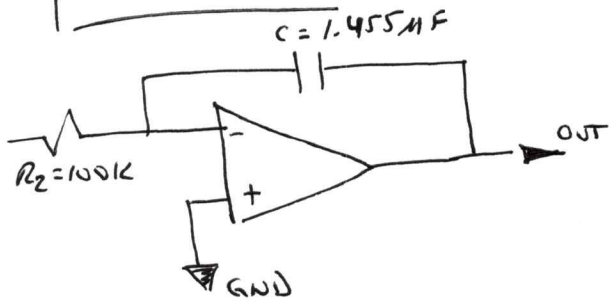
P => $K_1 = 7.457$

PI Gain: $K_2 = 6.872$

$K_2 = \frac{1}{R_2 C}$

Let $R_2 = 100K\Omega$

$C = \frac{1}{K_2 R} = \boxed{1.455\,MF = C}$

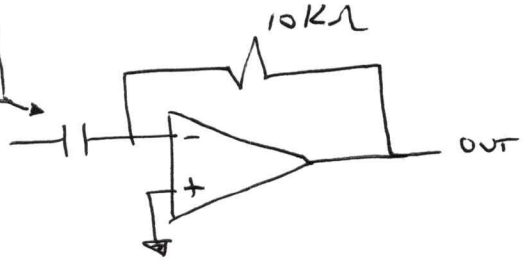


PD Gain: $K_3 = 0.0292$

$K_3 = RC$

Let $R = 10K\Omega$

$C = \frac{K_3}{R} = \boxed{2.92\,MF = C}$



P Gain: $K_1 = 7.457$

$K_1 = \frac{R_F}{R_S} \Rightarrow$

~~$R_S = \frac{R_F}{K_1}$~~

Let $R_S = 10K\Omega$

$R_F = K_1 R_S = \boxed{74.57\,K\Omega = R_F}$