Unmanned Free-Swimming Submersible Vehicle Heading Control System

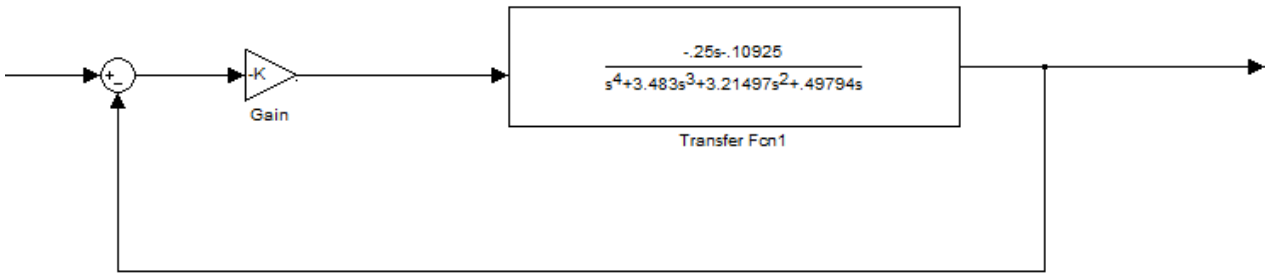
Group Members: Chris, Matt, Afshin, and Messan
EECS 444 Phase II Project

Abstract: This report describes the closed loop response of the Unmanned Free-Swimming Submersible Vehicle Heading Control System with 5 different controllers. These controllers are Phase Lead, Phase Lag, Lead/Lag, PID, and Rate Feedback. The value held constant between all the controllers was the steady state error to a ramp input, which was set to be 0.05 [rad]. This error proved to be unattainable for all the controllers but the PID. Therefore, the overall best controller was the PID. This controller was then digitized with a zero-order hold. Finally, the group decided to investigate load disturbance on this controller as the other design criteria.

Uncompensated System:

The uncompensated system from Phase I, which details the Unmanned Free-Swimming Submersible Vehicle Heading Control System, is a very important part of Phase II because the behavior of the initial system will be compensated for better performance through compensation controller design. The controllers that will be applied to the system include: phase-lag, phase-lead, lead-lag, rate-feedback, and PID controller. Each controller's effects on the phase I system will be optimized by fastest rise and settling times, and by best %OS. Furthermore the initial system is important because it gives a common denominator to measure each compensated system against the others. The way this is done is by keeping the steady state error caused by a unit ramp to be as close to 0.05 [radians/sec] as possible. The value calculated for K was -91.651; this value was calculated to give a steady state error of 0.05 for the uncompensated system, but as we'll see this gain is too much and it makes the uncompensated system unstable. By using the different controllers and changing K-values, it will re-stabilize the system.

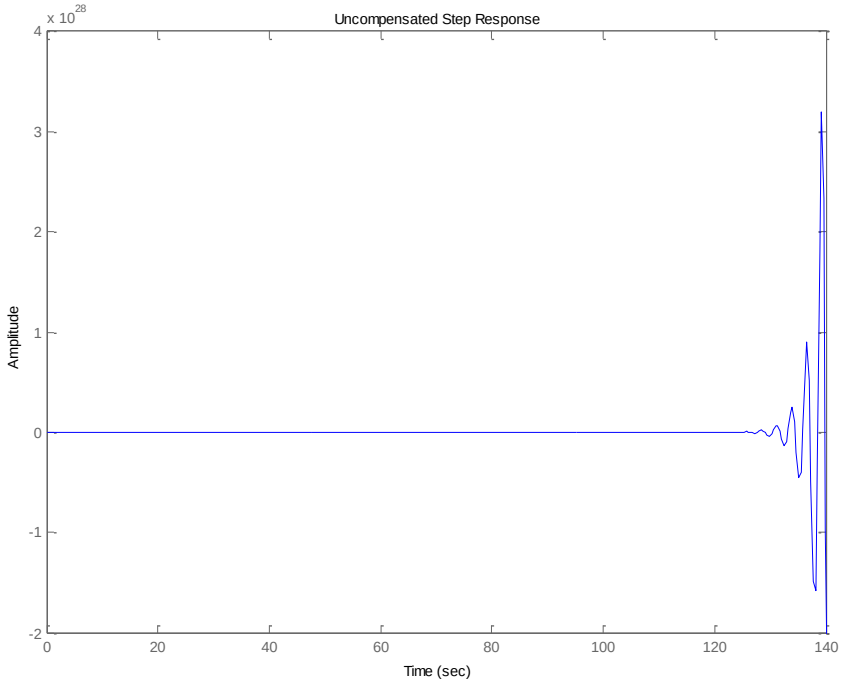
What follows is a Simulink image of the uncompensated system, a table of the uncompensated system values, and graphs using Matlab showing each individual plot.



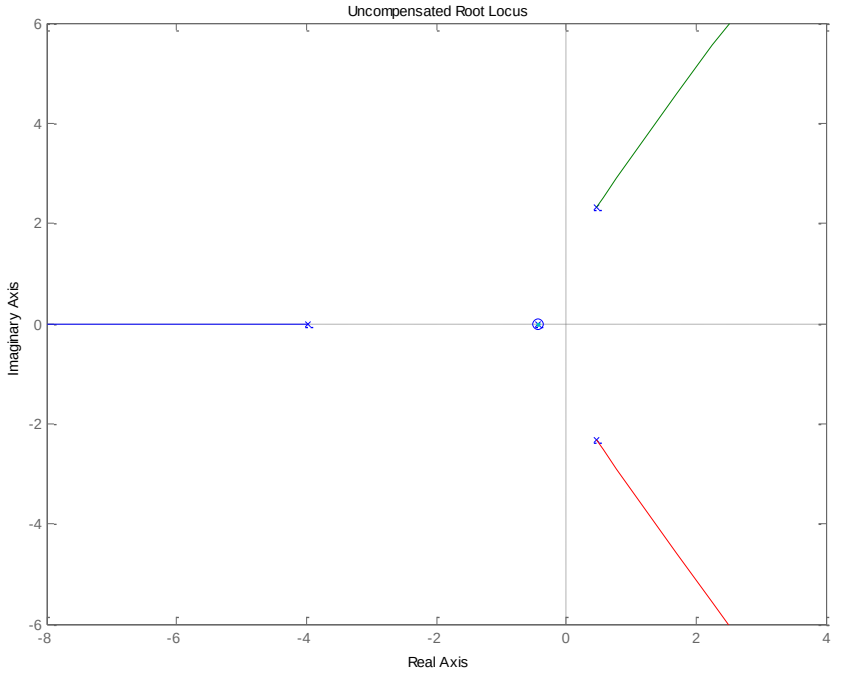
Uncompensated System

Variable	Uncompensated
% OS	0
K	-91.651
Rise Time (s)	8.3583
Setting Time(s)	328.9105
K_v	0
e_{ss}	infinite
GM(dB)	0.2552
PM(Degrees)	-30.0048
Peak Magnitude	1.5210e+025

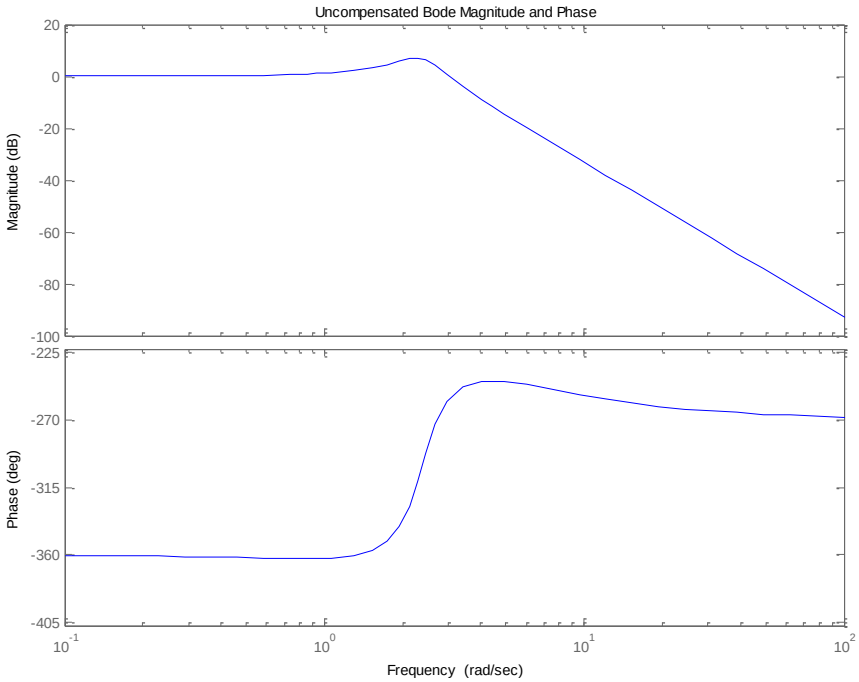
Attributes – Uncompensated System



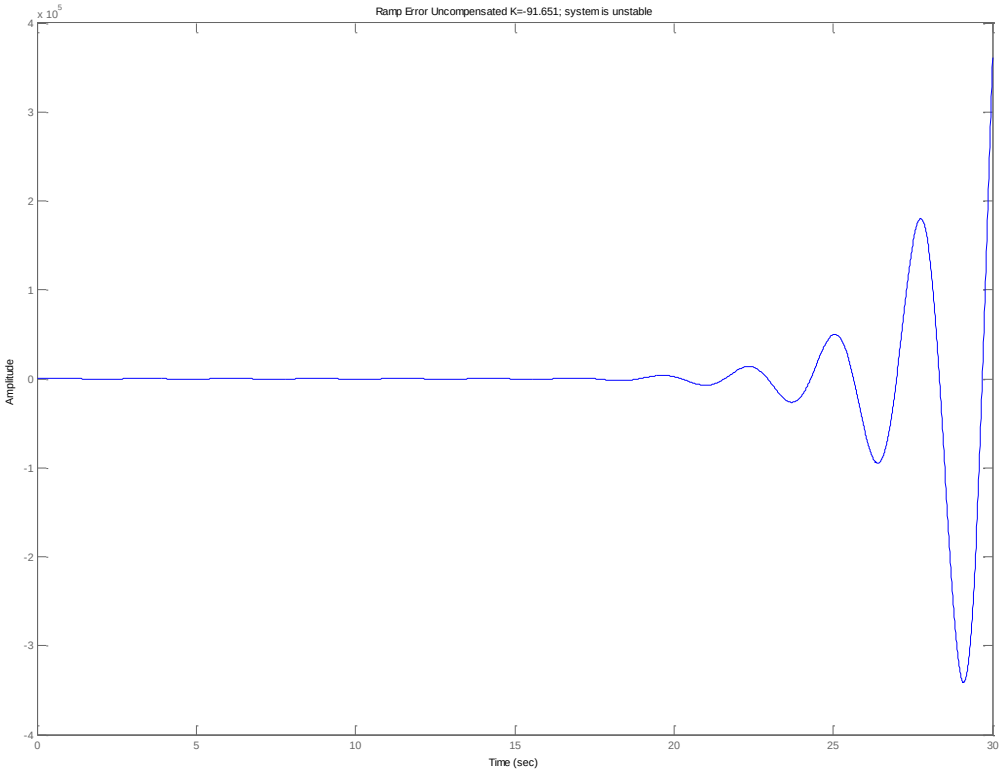
Uncompensated Step Response



Uncompensated Root Locus



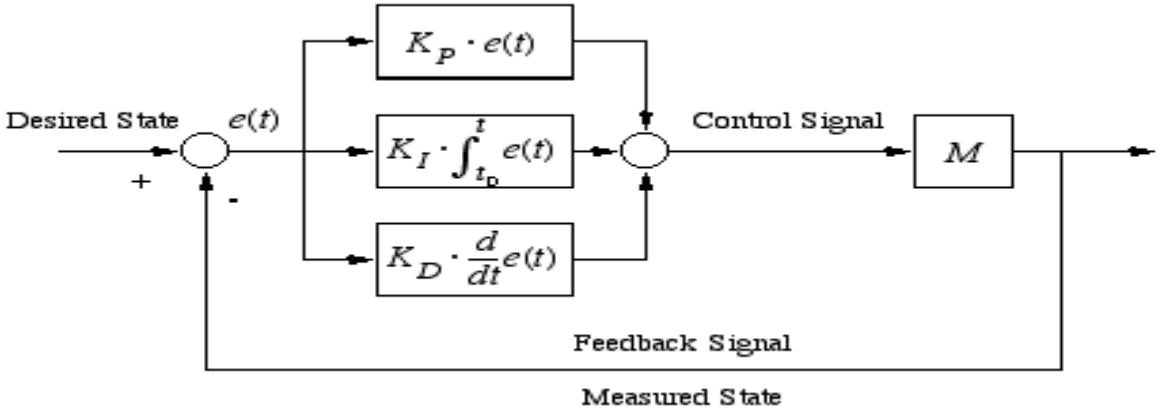
Uncompensated Bode



Uncompensated Ramp Error

Proportional – Integral – Derivative (PID) Controller:

The PID controller is very commonly used in control systems to compensate for specific unwanted qualities about the plant. The PID controller is made up of two separate parts called the Proportional-Integral (PI) controller and the Proportional-Derivative (PD) controller. Both parts are not required to compensate some systems. For example the PI portion might be used without the PD portion if it is able to satisfy the requirements of the overall system. An example of a PID controller is shown below:



Generic PID Controller

Looking at the image above shows the three variables that can define a PID controller in practice, and those variables are K_P, K_I, and K_D. Each variable affects specific parts of how the system behaves. For example the K_P portion, proportional, which is effective at improving the rise time of the compensated system. The K_I portion, integral, is effective at limiting the steady-state error, which normally goes to zero. The last part is K_D, derivative, which is used to improve the %OS which also helps transient response. A major benefit to the PID controller is that both the steady-state error and the transient response are effectively compensated without one interfering with compensating the other. This is not true with other controllers, where decreasing the %OS will increase the rise time and other aspects of a system, and in cases like these, a very fine balance must be achieved between each compensated value of the system. The PID controller can be modeled mathematically by the following expression:

$$\mathbf{K_P} + \frac{\mathbf{K_I}}{\boldsymbol{s}} + \mathbf{K_D}\boldsymbol{s} = \frac{\mathbf{K_D}\boldsymbol{s}^2 + \mathbf{K_P}\boldsymbol{s} + \mathbf{K_I}}{\boldsymbol{s}}$$

Equation describing the PID Controller

The PID controller effectively adds two zeros and a single pole at the origin. One zero is for the ideal derivative. The other zero and pole are for the ideal integrator. The controller designed aimed to decrease the %OS and rise time of the plant while keeping in mind the peak value and other attributes such as the settling time. The condensed transfer function for the PID controller implemented is shown below:

$$\frac{0.06989 * s^2 + 0.03685 * s + 0.0001835}{s}$$ (Equation for our PID Controller)

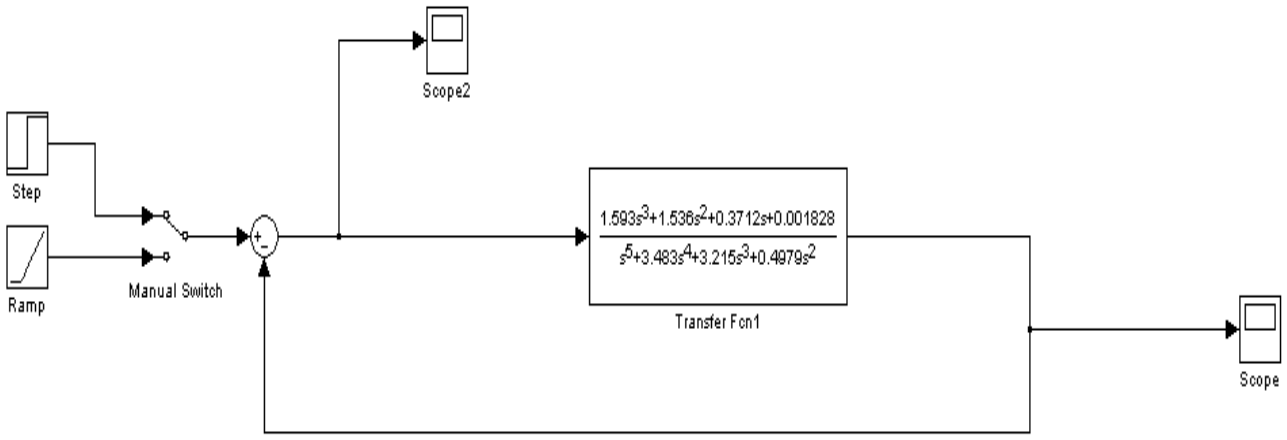
The design method used to optimize each variable value of the PID controller can be summarized in the following graphic:

CL RESPONSE	RISE TIME	OVERSHOOT	SETTLING TIME	S-S ERROR
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small Change

The designing consisted of cascading the controller into the plant, and then adjusting the values of K_P, K_I, and K_D to optimize the %OS and rise time of the total system. The values obtained by using the table above and the art of guessing and checking yielded the values as follows:

Variable	Value
K_P	0.0698888
K_I	0.0001835
K_D	0.0368461

The Simulink image for the cascaded controller and the plant of the system along with the gain is shown below:

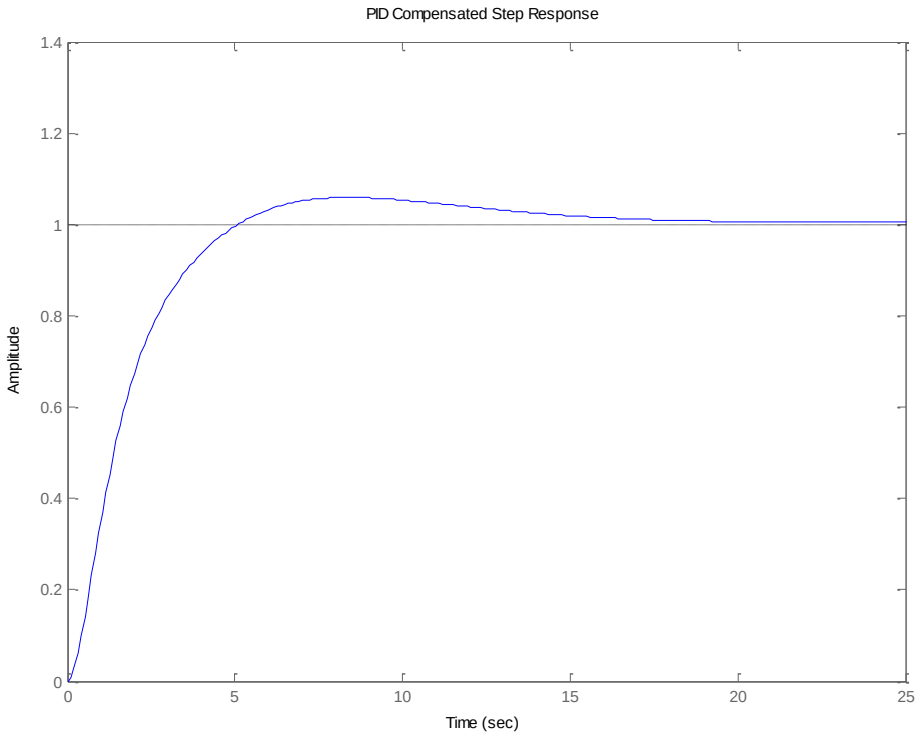


Collapsed Transfer Function

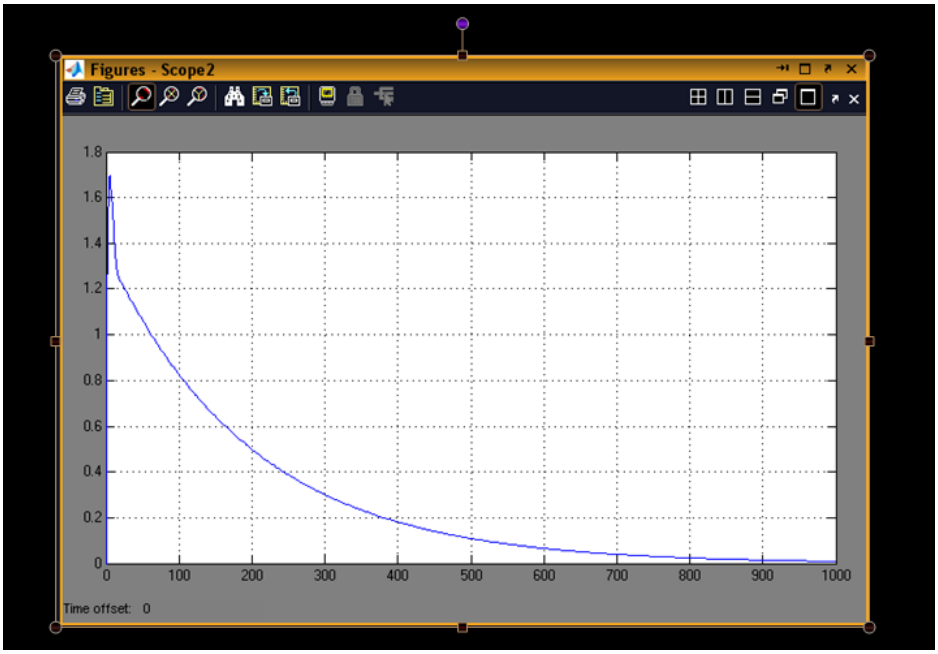
The values for the PID controller are summarized in the table below:

Variable	Compensated
% OS	5.9210
K	-91.651
Rise Time (s)	3.1322
Setting Time(s)	14.8391
K_v	0.0109109
e_{ss}	~0
GM(dB)	0
PM(Degrees)	77.9145
Peak Magnitude	8.3721

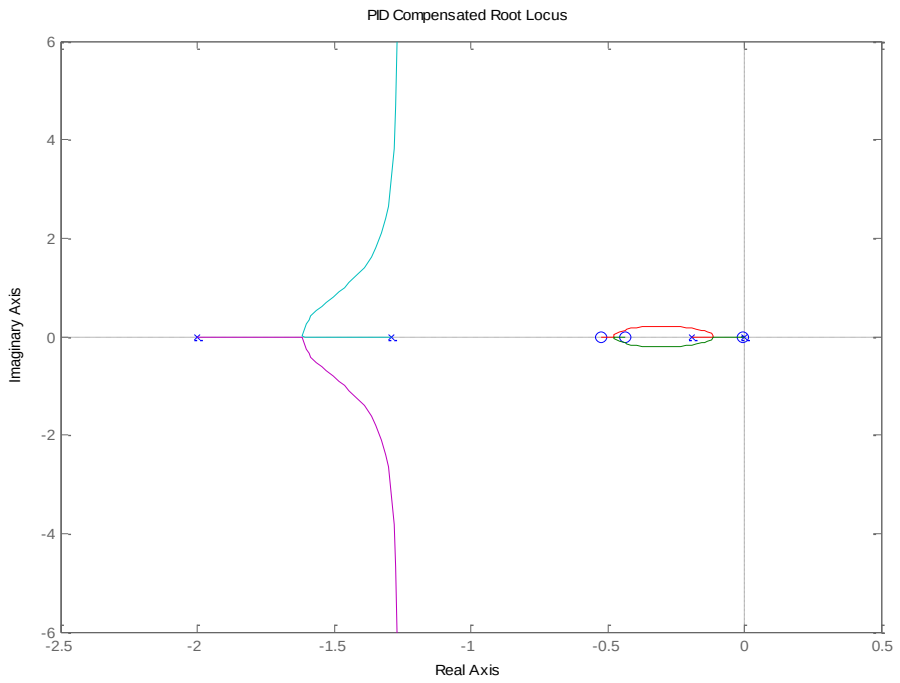
The following plots, generated by Matlab, are for the PID controller:



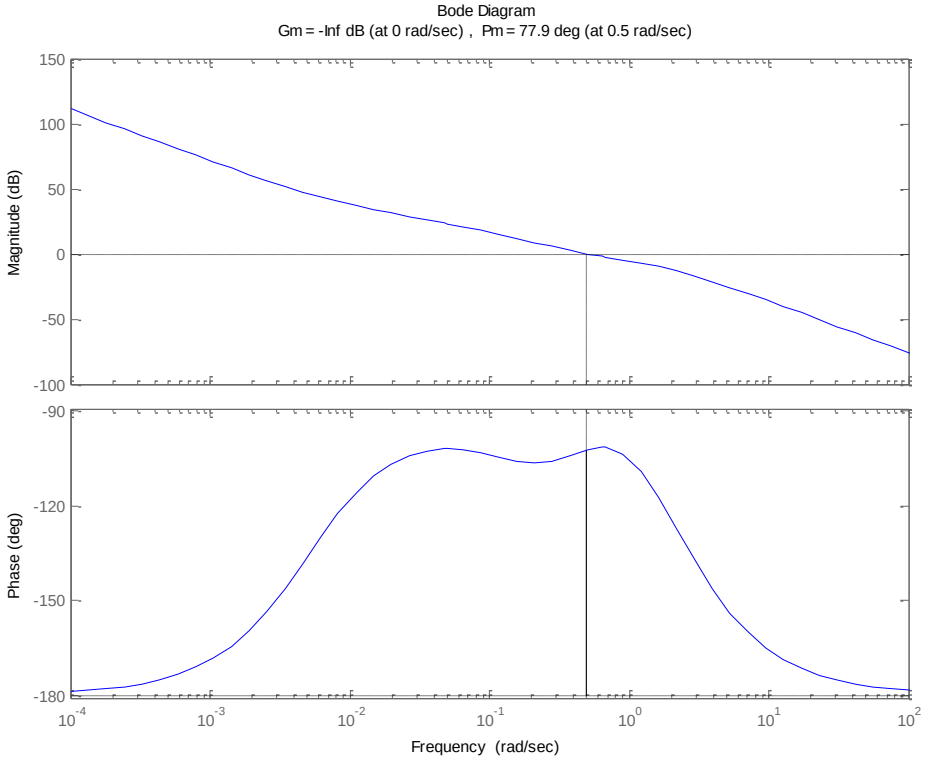
PID Step Response



Error vs. Time for Ramp Input



PID Root Locus



PID Bode Plot

Lead Controller:

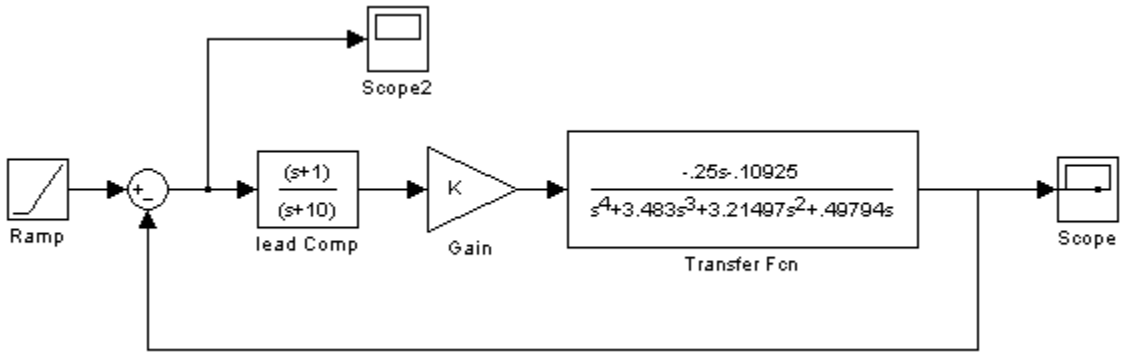
A lead compensator can increase the stability and speed of response of a system. The equation to determine the intersection of the asymptotes along the real axis is:

$$\alpha = \frac{\sum(poles\ ) - \sum(zeros)}{(\#poles) - (\#zeros)}$$

When a lead compensator is added to a system, the value of this intersection will be a larger negative number than it was before. The net number of zeros and poles will be the same (one zero and one pole are added), but the added pole is a larger negative number than the added zero. Thus, the result of a lead compensator is that the asymptotes' intersection is moved further into the left half plane, and the entire root locus will be shifted to the left. This can increase the region of stability as well as the response speed.

The lead compensator: $\frac{(S+1)}{(S+10)}$

Figure below shows the block diagram of the compensated system.

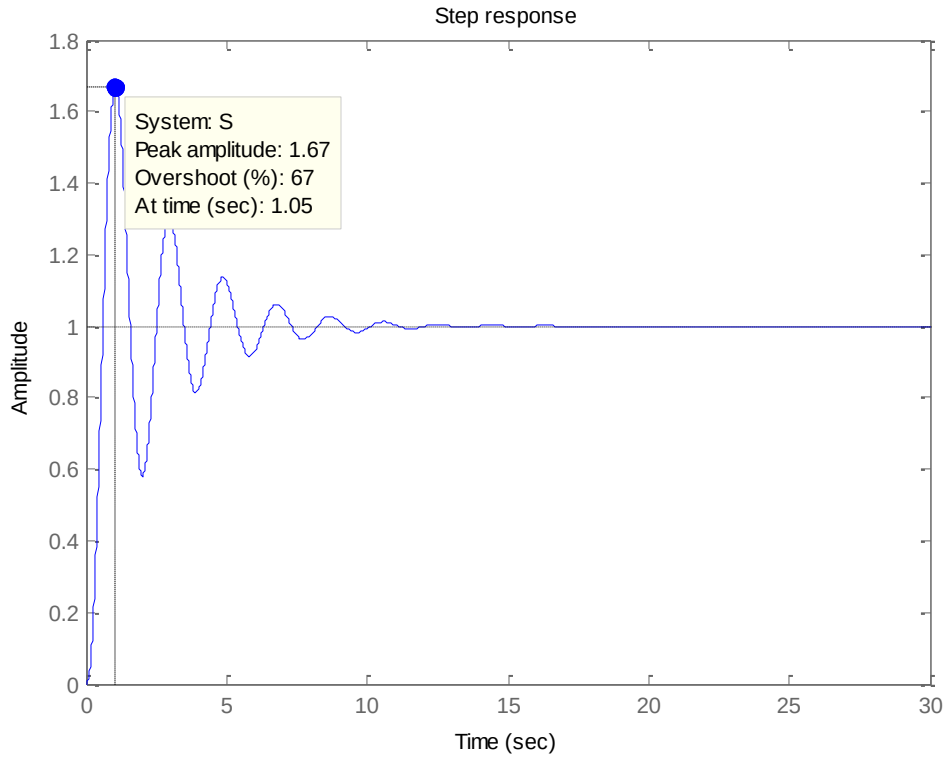


Block Diagram of the Compensated System

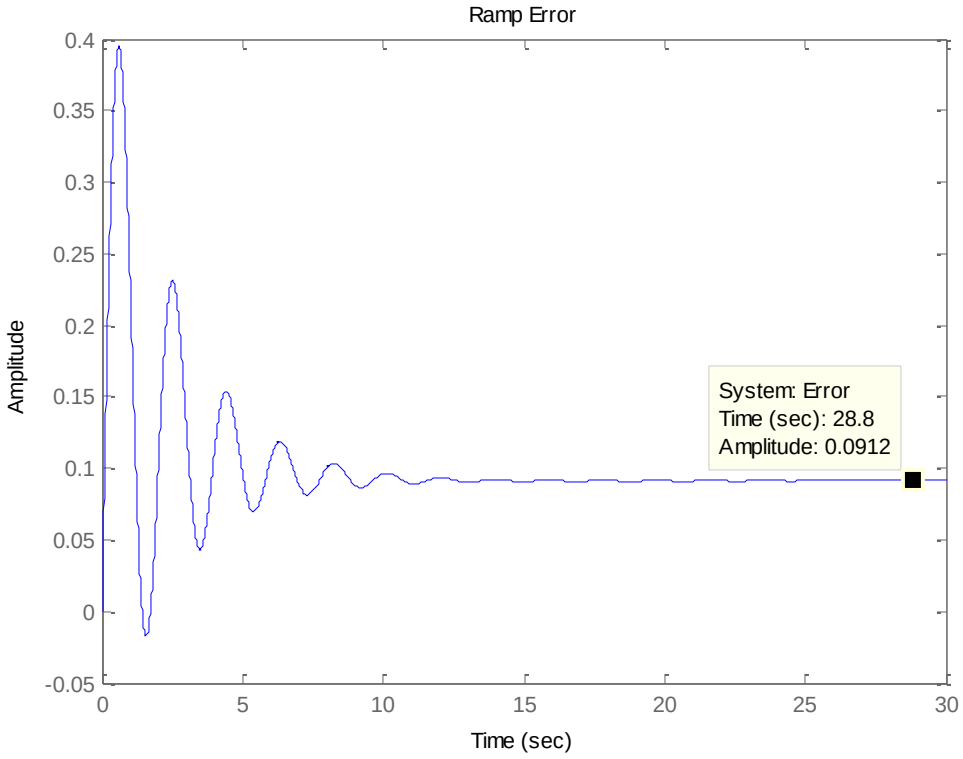
Using Matlab and our system, an appropriate lead controller was found by changing the pole and zero until the best result was found. The following values were measured using the transient response from Matlab.

Responses of the Compensated System

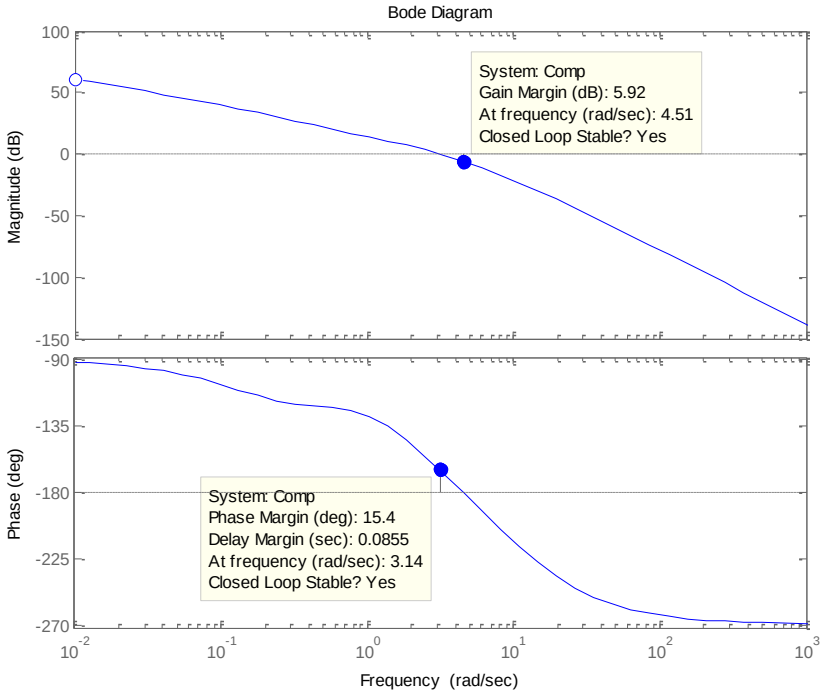
Variable	Compensated
% OS	67.16
K	500
Rise Time (s)	13.82
Setting Time(s)	345.0372
K_v	10.9649
e_{ss}	0.0912
GM(dB)	0.9769
PM(Degrees)	-0.5191
Peak Magnitude	41



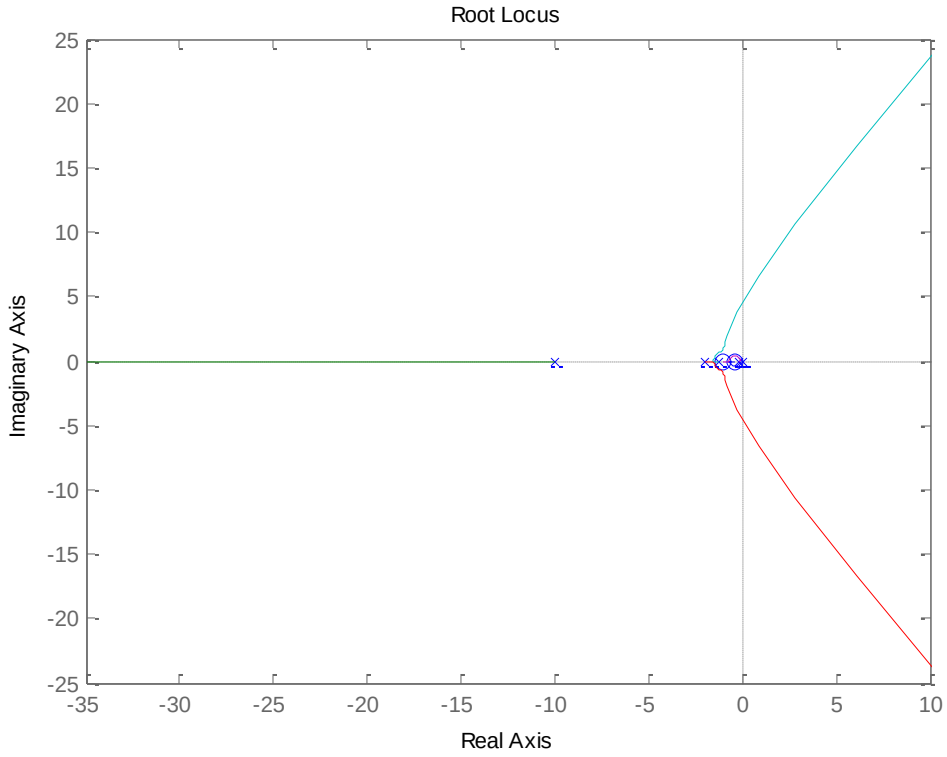
Transient Response of the Compensated System



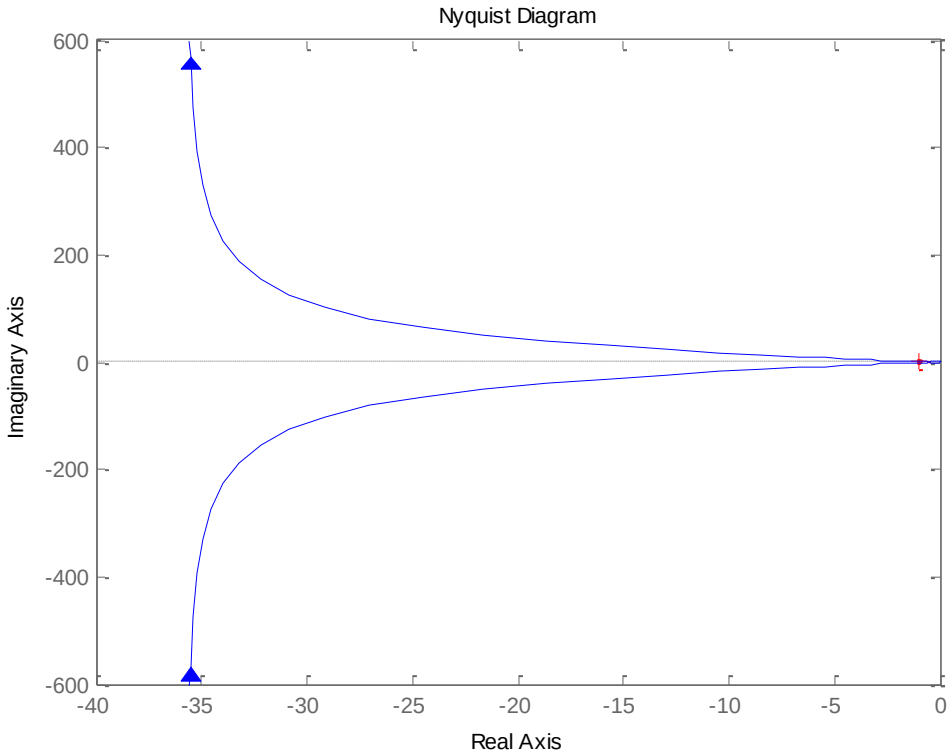
Ramp Error Plot of the Compensated System



Bode Diagram Plots of the Compensated System



Root Locus Plot of the Compensated System



Nyquist Plot of the Compensated System

Lag Controller:

A lag compensator is a device that provides phase lag in its frequency response. This compensator can be used to adjust frequency response by adding equal numbers of poles and poles to a system. Those added poles and zeros may possibly be manipulated to give better stability, better performance and general improvement. The lag compensator designed for this system was used to reduce the steady state error while having little effect upon the transient response; this was achieved by adding the pole and zero very close to the origin.

The specifications for our system were to have the lowest possible (%OS) when the steady- state error (ess) due to a unit-ramp input is equal to 0.05 radians. **The lag compensator is**: 5(s+0.02)/(s+0.001).

Figure 1 is the block diagram of the compensated system.

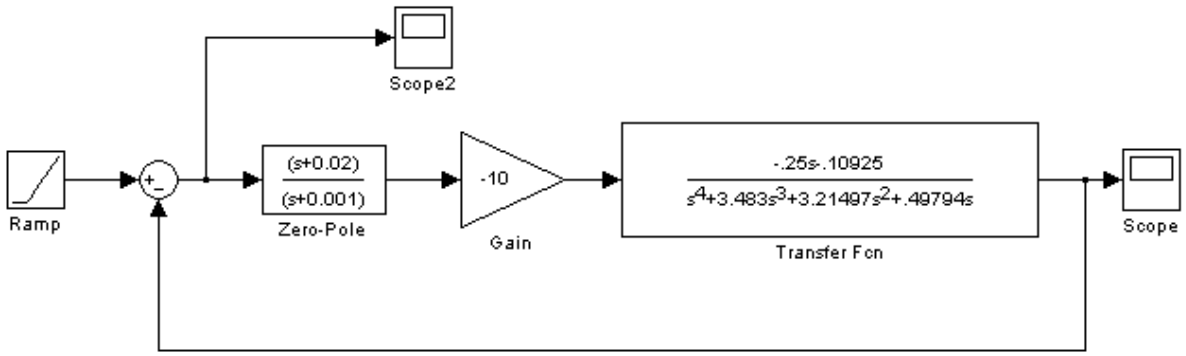
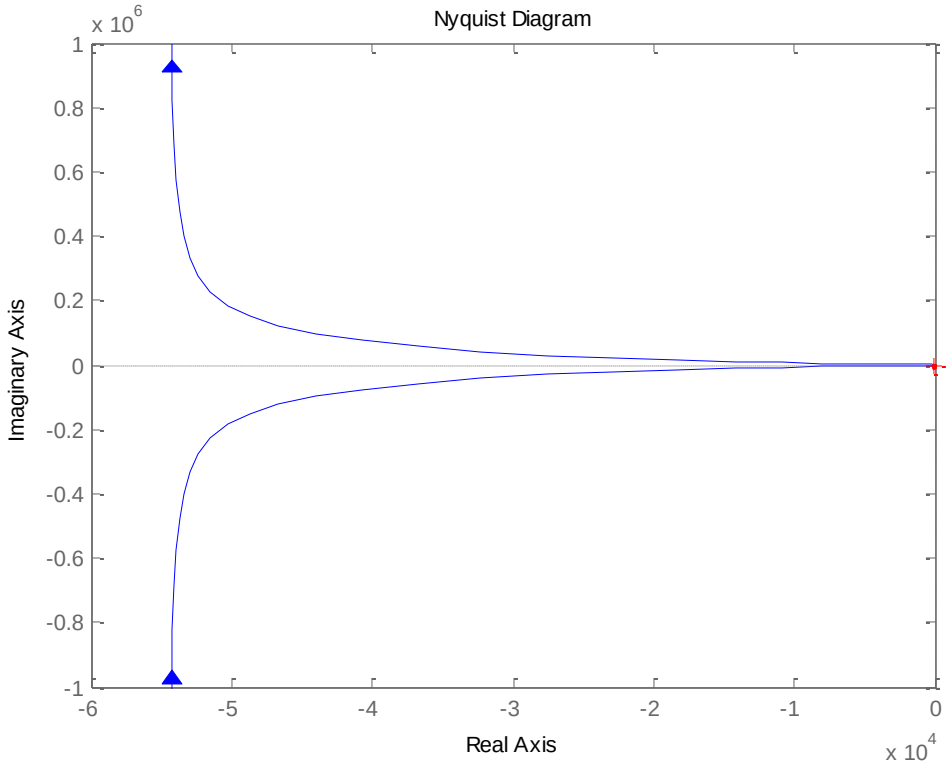
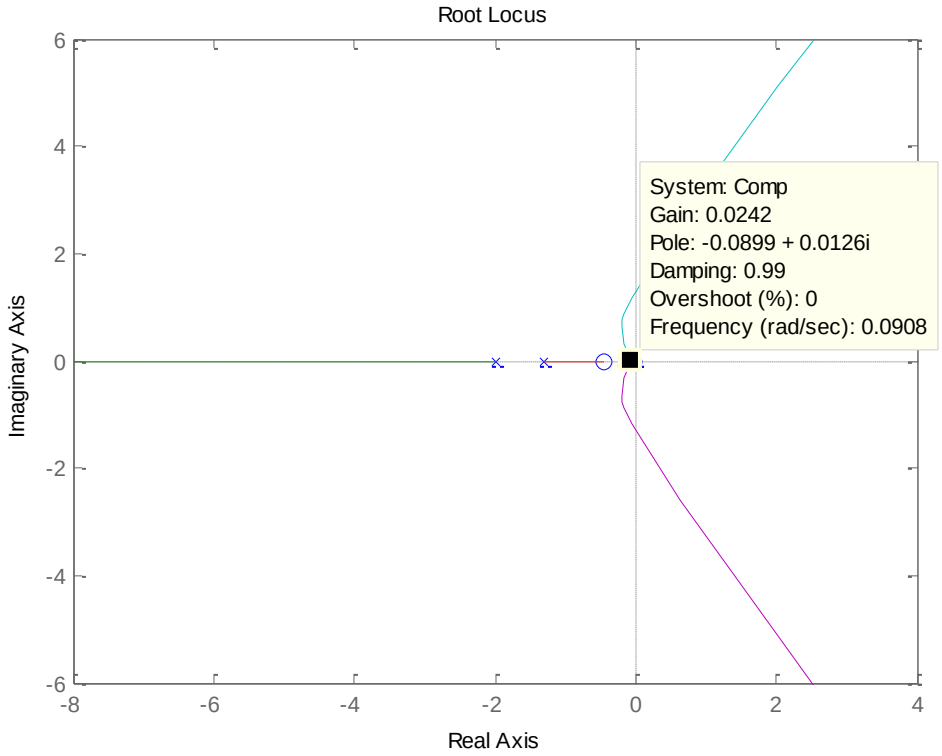


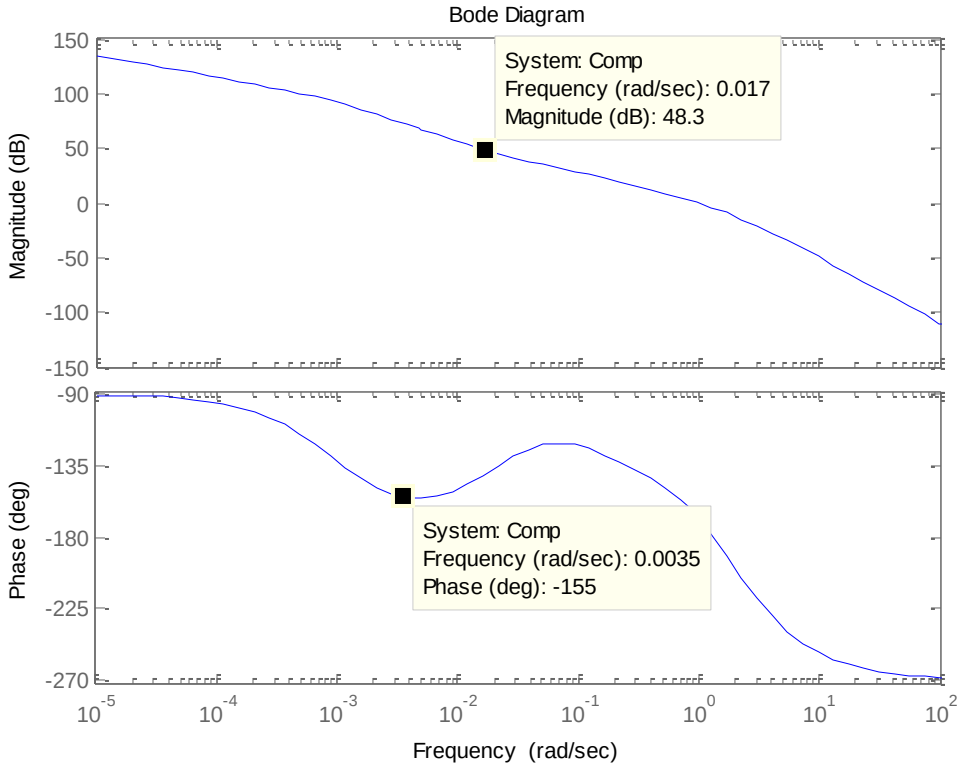
Figure 1: Block Diagram of the Compensated System

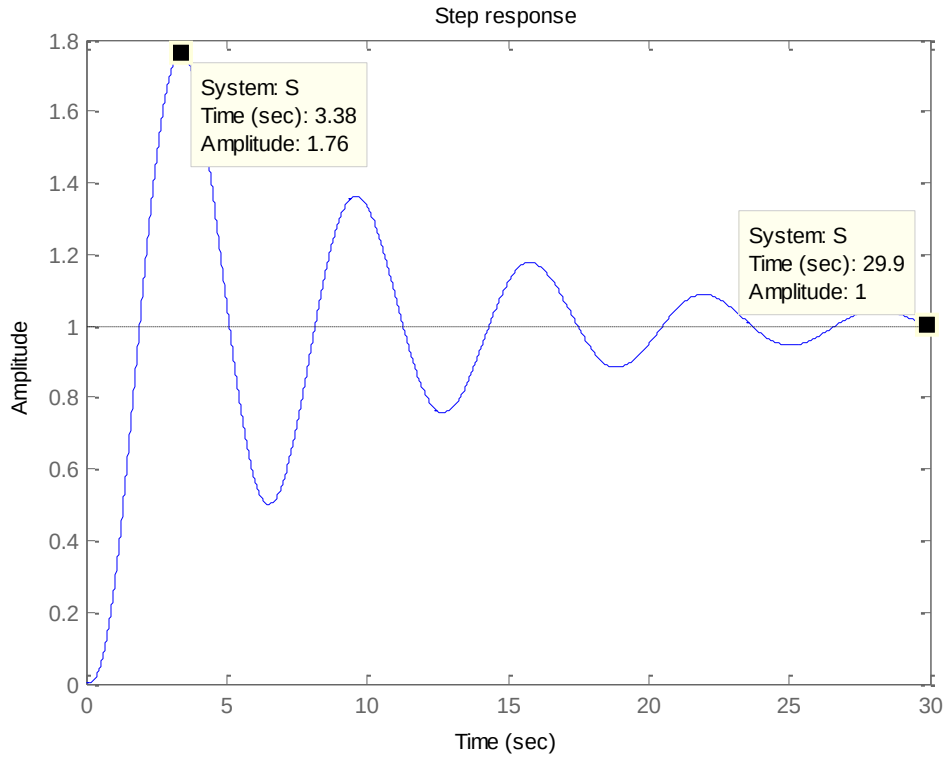
Using Matlab and this system, and appropriate lag controller was found by changing the pole and zero until the best result was found. The following values were measured using the transient response from Matlab.

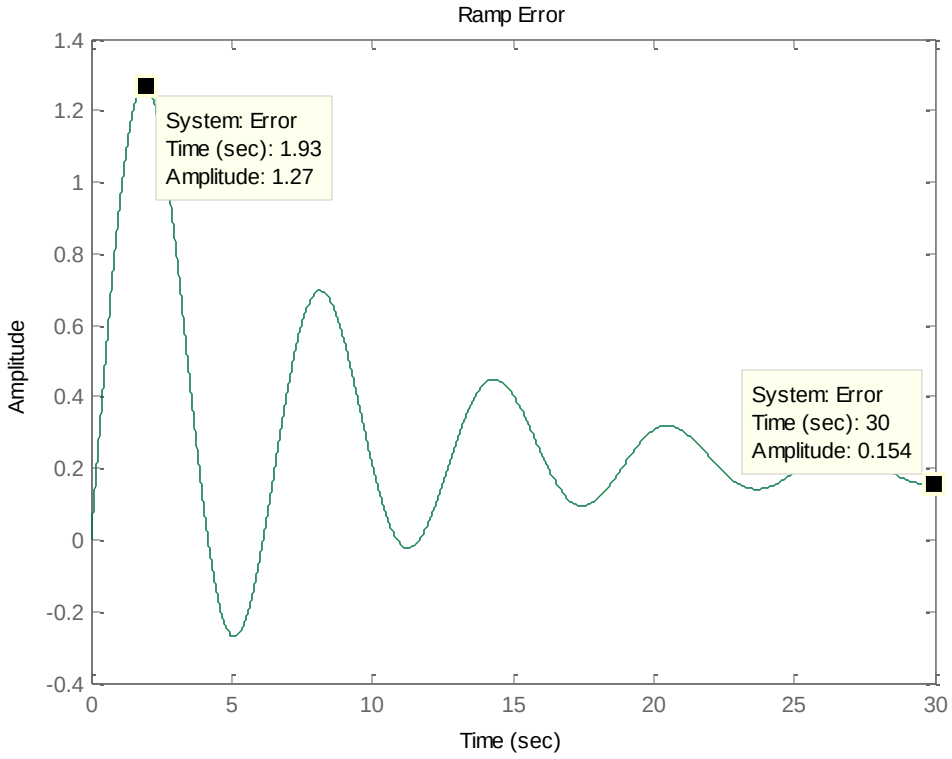
Variable	Compensated
% OS	77.11
K	13
Rise Time (s)	3.6827
Setting Time(s)	115.5381
K_v	6.49
e_{ss}	0.154
GM(dB)	0.7304
PM(Degrees)	-6.839
Peak Magnitude	1.764


Nyquist Diagram
x 10^6
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
Imaginary Axis
-6
-5
-4
-3
-2
-1
0
Real Axis
x 10^4


Root Locus
6
4
2
0
-2
-4
-6
Imaginary Axis
-8
-6
-4
-2
0
2
4
Real Axis
System: Comp
Gain: 0.0242
Pole: -0.0899 + 0.0126i
Damping: 0.99
Overshoot (%): 0
Frequency (rad/sec): 0.0908


Bode Diagram
System: Comp
Frequency (rad/sec): 0.017
Magnitude (dB): 48.3
System: Comp
Frequency (rad/sec): 0.0035
Phase (deg): -155
Magnitude (dB)
Phase (deg)
Frequency (rad/sec)


Step response
System: S
Time (sec): 3.38
Amplitude: 1.76
System: S
Time (sec): 29.9
Amplitude: 1
Amplitude
Time (sec)


Ramp Error
System: Error
Time (sec): 1.93
Amplitude: 1.27
System: Error
Time (sec): 30
Amplitude: 0.154
Amplitude
Time (sec)
1.4
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
5
10
15
20
25
30

Lead-Lag Controller:

A lead-lag compensator combines the effects of a lead compensator with those of a lag compensator. The result is a system with improved transient response, stability, and steady-state error. The controller adds two zeros and two poles to the original system. The lead controller improves the systems transient response while the lag controller improves the steady state error.

The lead controller was designed to target a low percent overshoot (%OS) and lower the settling time. Matlab was used to design the lead controller and develop one that met those specifications. From the results, the following transfer function was found:

500*(s+1)/(s+10) [lead controller transfer function]

The lag controller has a zero very close to the middle pole and the pole is located very far to the left of all the poles.

Next, the lag controller was designed to improve the steady state error and reduce rise time, settling time, peak time. Using Matlab, again to design the controller the following transfer function was found:

13*(s+0.02)/(s+0.001) [lag controller transfer function]

The zero is located very close to the origin but not directly on it and the pole is located just to the right of that, again very close to the origin. Having both lead and lag controllers transfer function, they were both multiplied into the original control system transfer function, giving the following final lead-lag transfer function:

220*(s+0.02)(s+1)/(s+0.001)(s+10) [lead and lag together]

Figure 1 is the block diagram of the compensated system.

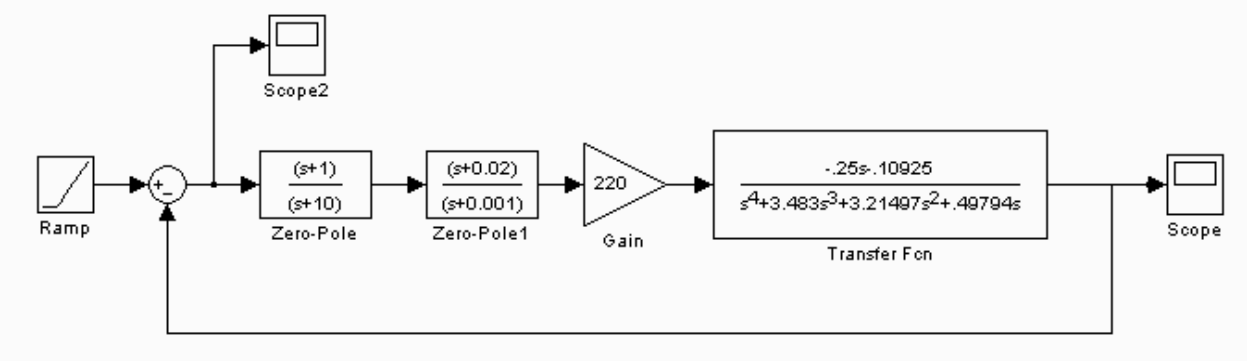
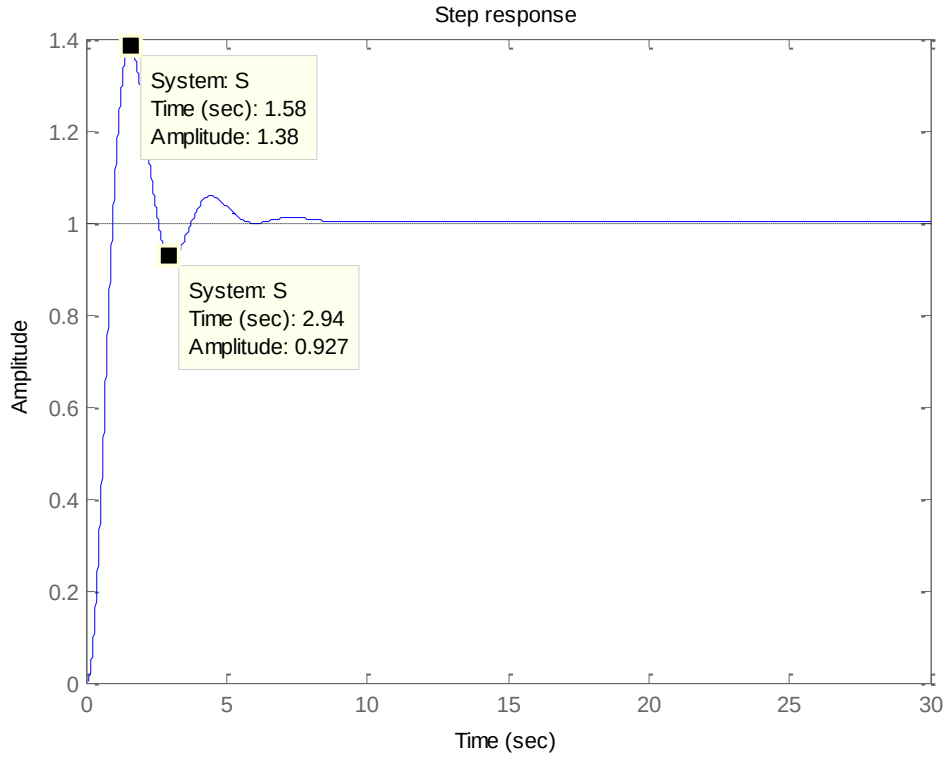
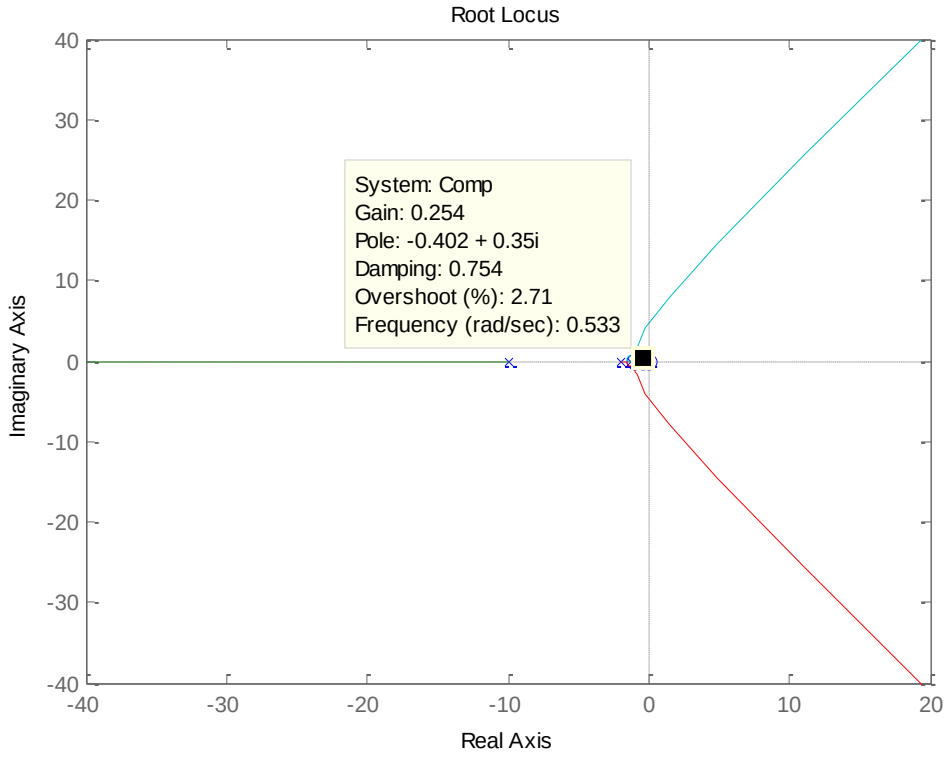


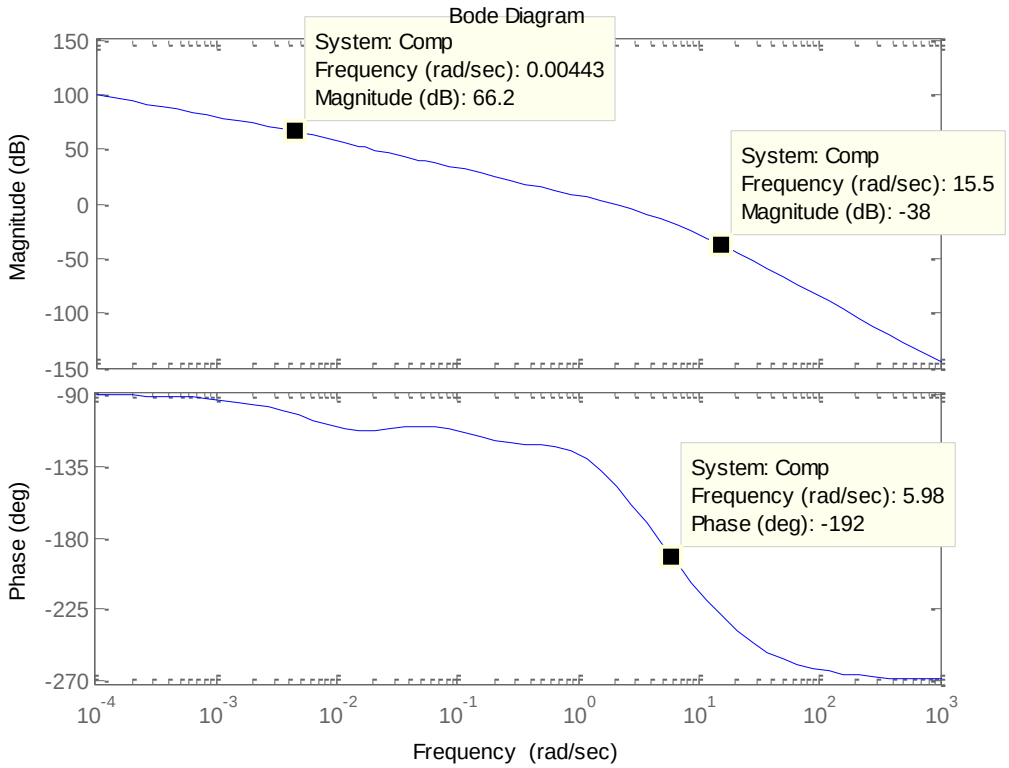
Figure 1: Block Diagram (Compensated)

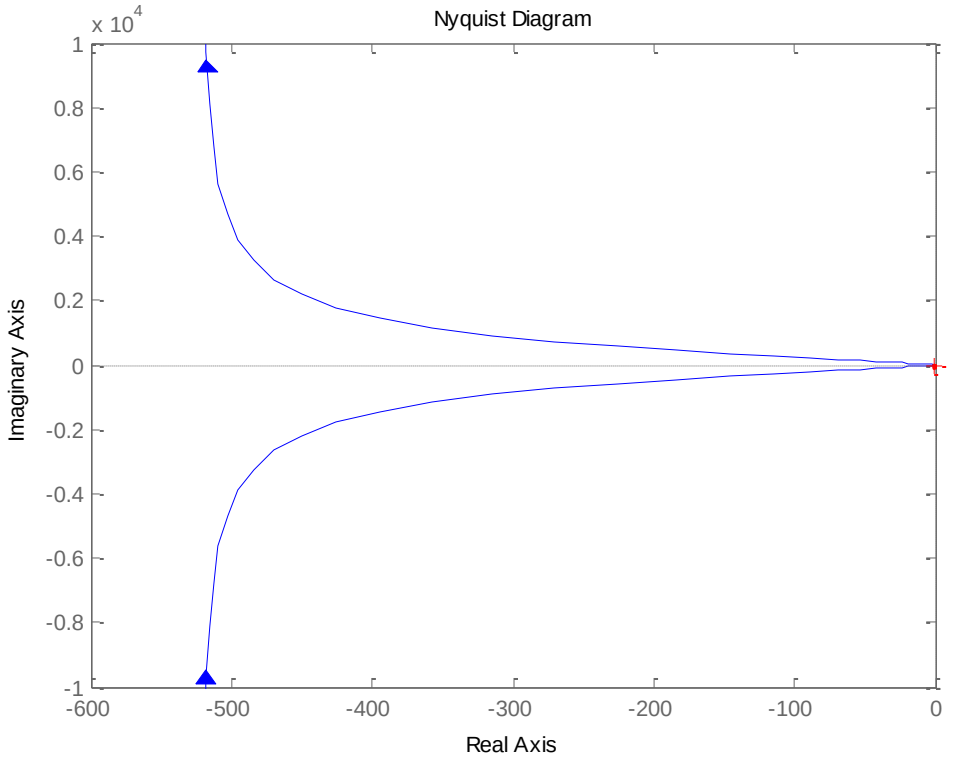
Table 2: Lead-Lag Controller Results

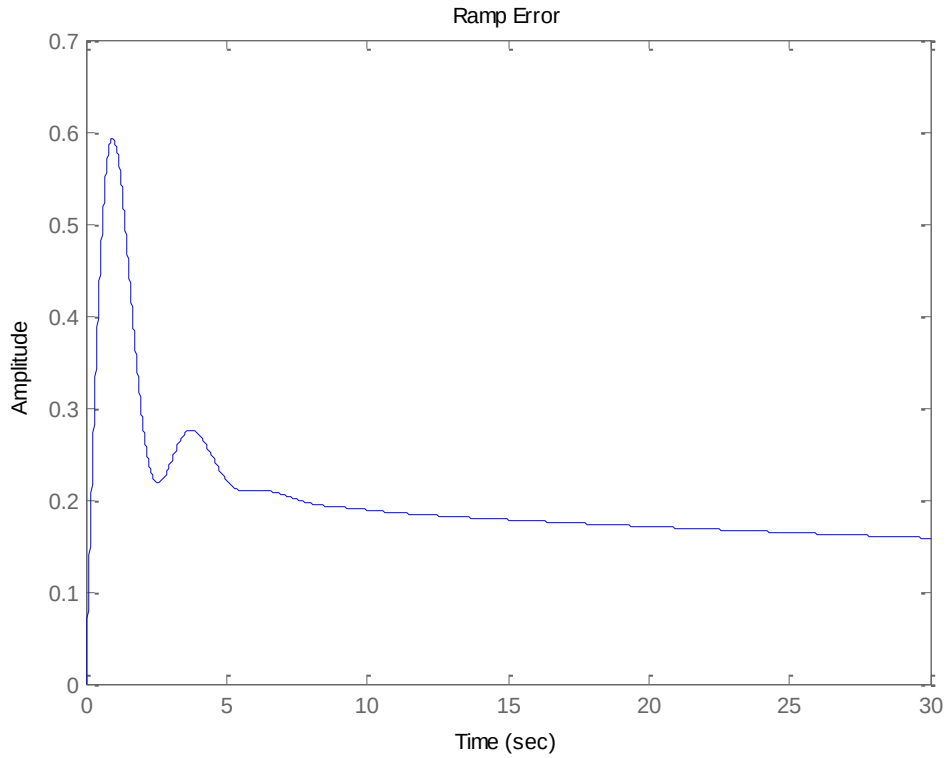
Variable	Compensated
% OS	37.68
K	220
Rise Time (s)	6.0705
Settling Time(s)	53.2029
K_v	6.329
e_{ss}	0.158
GM(dB)	3.478
PM(Degrees)	40.047
Peak Magnitude	17




Root Locus
System: Comp
Gain: 0.254
Pole: -0.402 + 0.35i
Damping: 0.754
Overshoot (%): 2.71
Frequency (rad/sec): 0.533
Imaginary Axis
Real Axis
40
30
20
10
0
-10
-20
-30
-40
-40
-30
-20
-10
0
10
20


Bode Diagram
System: Comp
Frequency (rad/sec): 0.00443
Magnitude (dB): 66.2
System: Comp
Frequency (rad/sec): 15.5
Magnitude (dB): -38
System: Comp
Frequency (rad/sec): 5.98
Phase (deg): -192
Magnitude (dB)
Phase (deg)
Frequency (rad/sec)
150
100
50
0
-50
-100
-150
-90
-135
-180
-225
-270
10^-4
10^-3
10^-2
10^-1
10^0
10^1
10^2
10^3


Nyquist Diagram
x 10^4
Imaginary Axis
Real Axis
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-600
-500
-400
-300
-200
-100
0


Ramp Error
Amplitude
Time (sec)
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5
10
15
20
25
30

Rate Feedback:

Figure 1 shows the original uncompensated system.

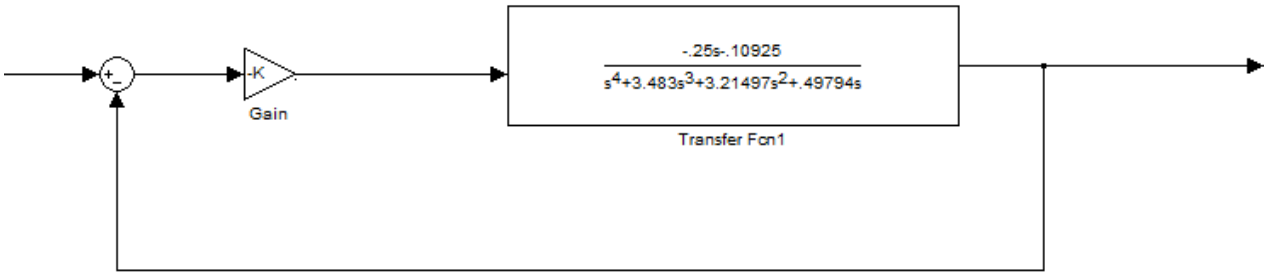


Figure 1- Uncompensated System

Figure 2 shows the block diagram of the original system with a rate feedback controller added.

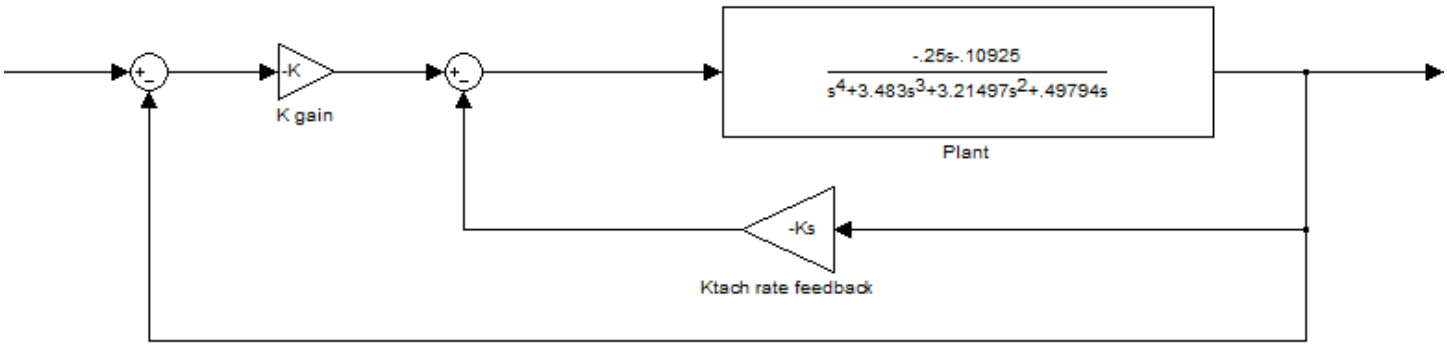


Figure 2-Uncompensated system with rate feedback controller

Reducing figure 2 to a single plant with a unity gain feedback; the equation for the new plant turns out to be:

$$G_p(s) = \frac{-K(.25s + .10925)}{s[s^3 + 3.483s^2 + (3.21497 - .25K_{tach})s + (.49794 - .10925K_{tach})]}$$

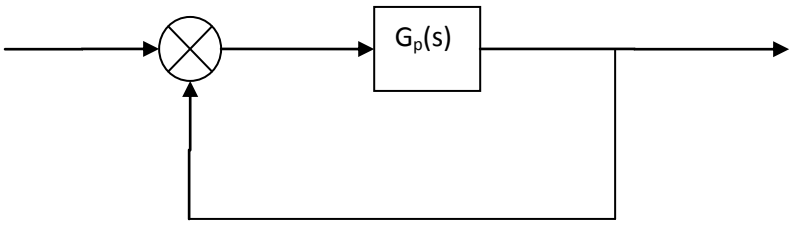
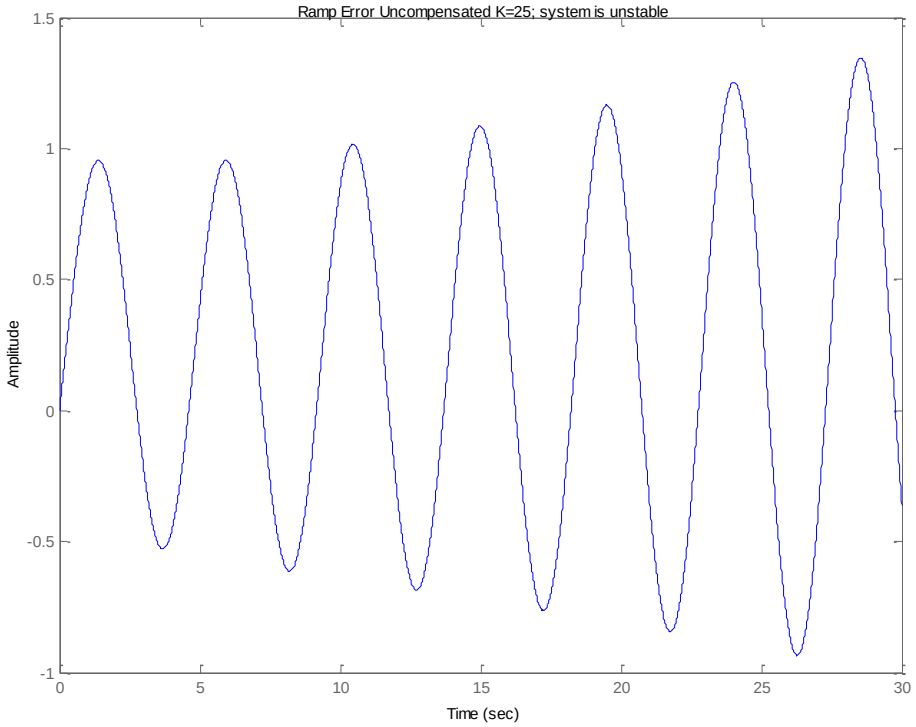


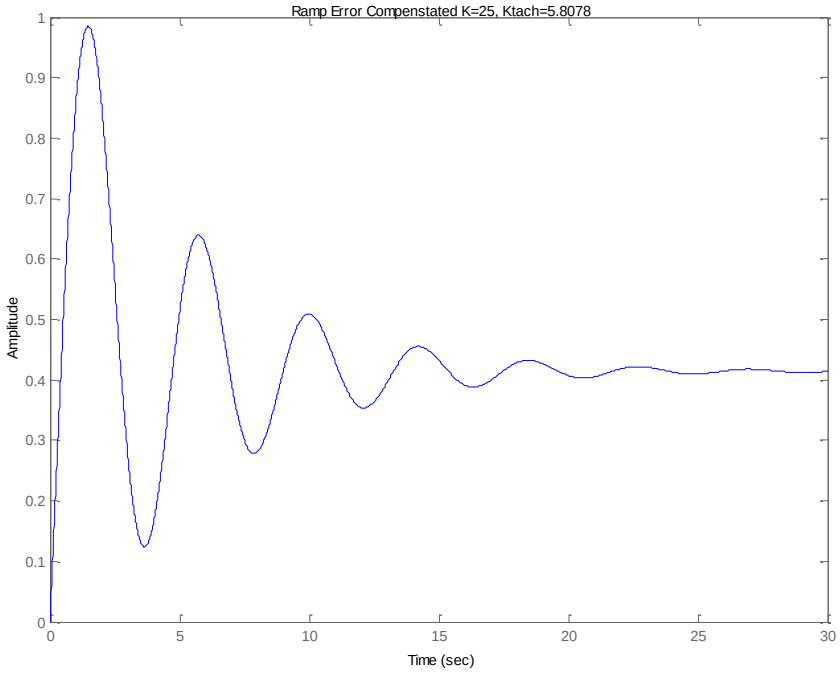
Figure 3- Reduction of figure 2

To find the values for K and K_{tach} that will theoretically give a steady state error of 0.05 with a ramp input, the following steps were taken. Take the $\lim_{s\to 0} sG_p(s)$ to find K_v. Then set $\frac{1}{K_v}$ equal to 0.05 and solve the equation for either K or K_{tach}. The following result was obtained.

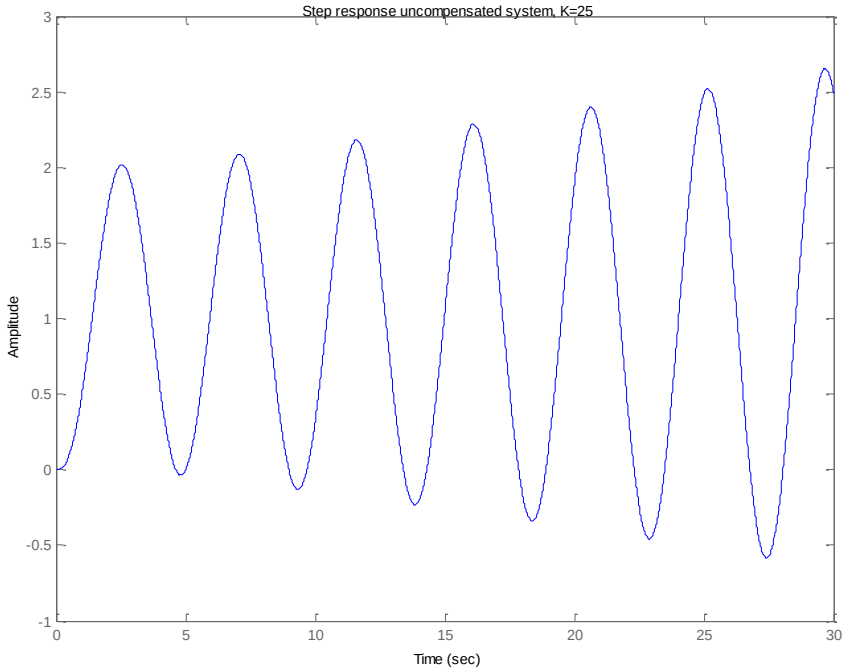
$$K_{tach} = .05K + \frac{.49794}{.10925} \quad \text{(This equation relates the values of K and } K_{tach})$$

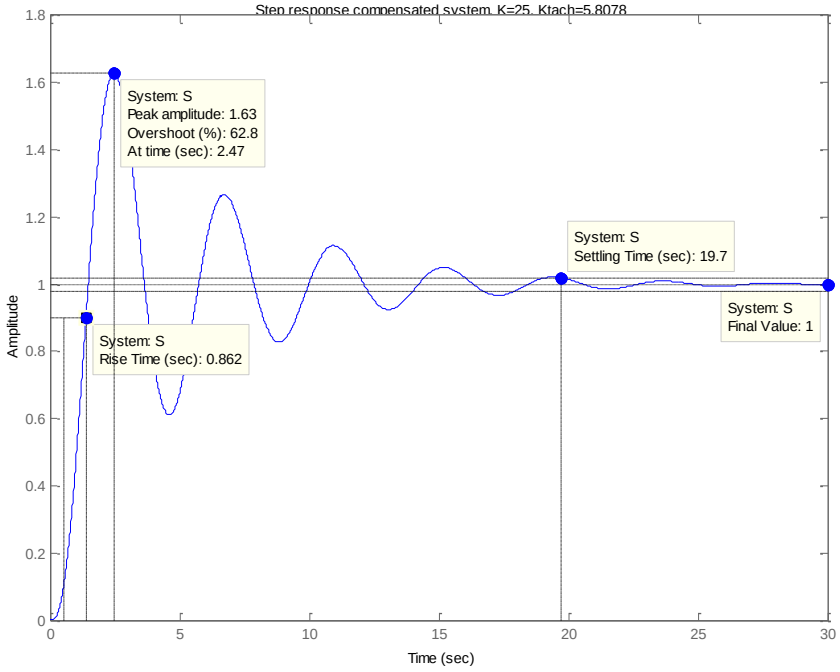
This controller did not work well. Numerous combinations of K and K_{tach} values were tried; even ones that were not constrained by the above equation and none resulted in a steady state error of 0.05. The lower the error got, the higher the percent over-shoot was, and vice versa. It seems that no compromise between percent over-shoot and error was a particularly good one. Therefore, I choose the K and K_{tach} values that gave the lowest steady state error. For the compensated system **K**=25 and $\mathbf{K_{tach}}$=5.8078. To give a comparison the same gain of **K**=25 was used for the uncompensated system. Following are the Matlab plots.



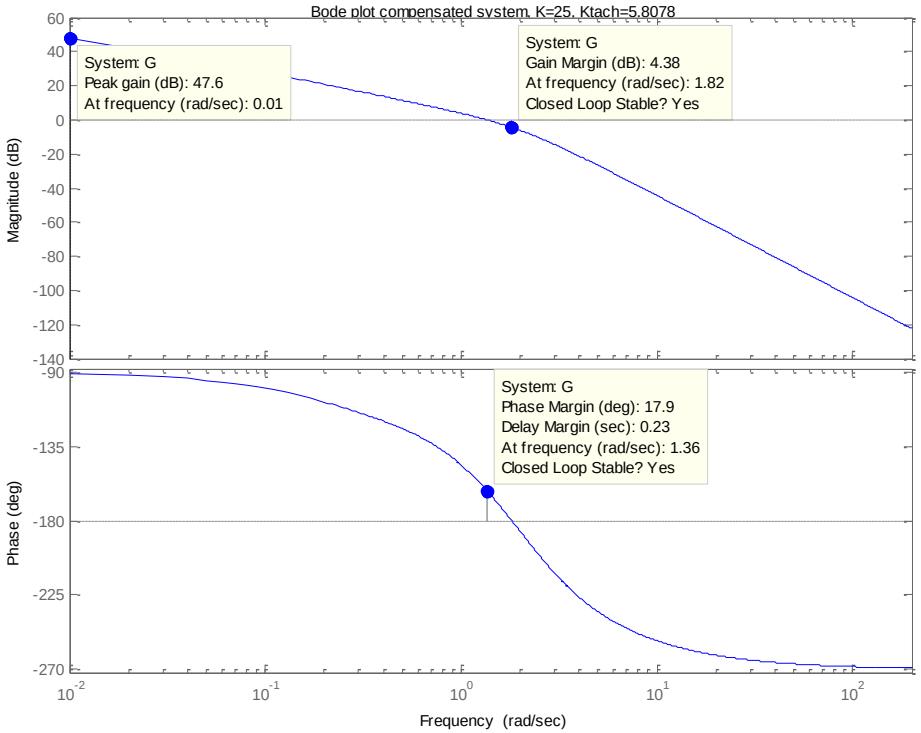


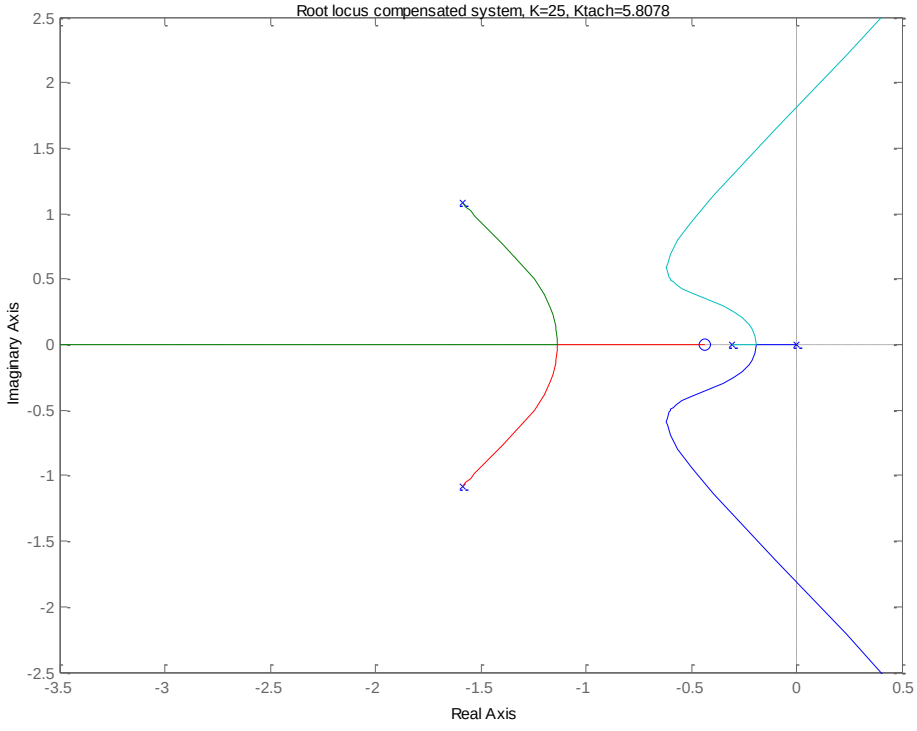
As can be seen between the two plots the uncompensated system has an unstable error, but the compensated system settles to a steady state error of 0.41, which is not very good, but was the best I could achieve. One thing is for sure, the controller is at least working properly. Following are the step responses of both uncompensated and compensated systems.

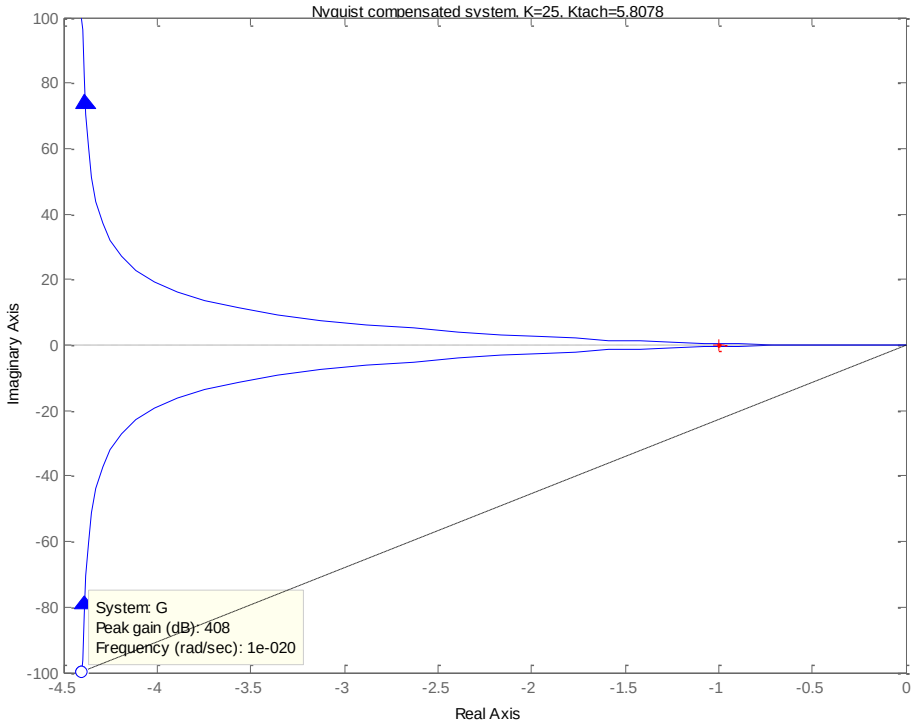




Note, the uncompensated systems step response is unstable, but the compensated system is stable. However, it has a percent over-shoot of 62.8%, which is undesirable. The design criteria were to achieve a 0.05 steady state error with a ramp input. The closest obtained was 0.41, with a huge penalty of a 62.8% over-shoot. This controller will not be chosen. Following are the bode, nyquist, and root locus plots for the open loop compensated system.


Bode plot compensated system, K=25, Ktach=5.8078
System: G
Peak gain (dB): 47.6
At frequency (rad/sec): 0.01
System: G
Gain Margin (dB): 4.38
At frequency (rad/sec): 1.82
Closed Loop Stable? Yes
System: G
Phase Margin (deg): 17.9
Delay Margin (sec): 0.23
At frequency (rad/sec): 1.36
Closed Loop Stable? Yes
Magnitude (dB)
Phase (deg)
Frequency (rad/sec)


Root locus compensated system, K=25, Ktach=5.8078
Imaginary Axis
Real Axis



Below is a table summarizing the important values from the analysis of the compensated rate feedback system.

K	K_{tach}	Rise time [sec]	Settling Time [sec]	e_{ss} (ramp) [rad]	% OS	GM [dB]	PM [degrees]
25	5.8078	0.862	19.7	0.41	62.8	4.38	17.9

Once again, this controller did not work well and will not be chosen as the overall "best". The rise time is pretty fast, which is good, but the settling time is horrible. The lowest steady state error that could be achieved was 0.41 [rad], which is about 23.5 degrees; this is still bad. Lastly, the %OS of 62.8% is not good at all.

Best Controller:

The following table summarizes the values given by each differently compensated and the uncompensated system:

Variable	Uncompensated	PID Controller	Rate-Feedback	Lead	Lag	Lead-Lag
% OS	0	5.9210	62.8	67.16	77.11	37.68
K	-91.651	-91.651	25 (K_{tach} 5.8078)	500	13	220
Rise Time (s)	8.3583	3.1322	0.862	13.82	3.6827	6.0705
Settling Time	328.9105	14.8391	19.7	345.0372	115.5381	53.2029
K_v	0	N/A	2.439	10.9649	6.49	6.329
e_{ss}	infinite	0	0.41	0.0912	0.154	0.158
GM(dB)	0.2552	0	4.38	0.9769	0.7304	3.478
PM(Degrees)	-30.0048	77.9145	17.9	-0.5191	-6.839	40.047
Peak Mag.	1.5210e+025	8.3721	Not provided	41	1.764	17

When examining the table to identify the best controller for our system, it is clear that the PID controller is superior to the others. The reasoning behind the selection of the PID controller to digitize is as follows: First the %OS with the PID was 5.91, which was by far the lowest of the compensated systems. The uncompensated system has 0 %OS because it is a stable line up until the point that it becomes wildly unstable with an amazing approximated peak magnitude of 1.521 x 10^25th power! Looking first at %OS makes sense because it is one of the major characteristics when designing a system. The reason for this is because too much %OS with a controller will cause the plant to be chaotically oscillating. The next attribute that stood out in the table was the 77.91 phase margin of the PID controller. This is vastly superior to the uncompensated with -30, as well as all the compensated systems. Phase margin is an important attribute when designing systems because too much phase shift will cause instability. Looking at other items like the PID's rise and settling times it can be seen that they are both relatively good in the overall view, but the rate-feedback beat, overwhelmingly, the other systems with a rise time of 0.862.

It was for these reasons that the PID controller was selected as the best fitting controller to our plant, and will continue on to the digital realm.

Digital Controller:

A digital control system is sometimes superior to the analog system because they can be created simpler which reduces the cost, and digital systems can help other things like external noise. In order to convert from an analog to digital system, one must sample values from the analog system at a set rate. This makes sense because an analog system is continuous with theoretically infinite number of possible values, while the digital system is finite and has a limited number of values. Therefore we must approximate the analog system in order for it to become a digital system. In designing a digital system from an analog, a zero-order-hold (ZOH) is used which samples the analog system over an interval of values at a specific periodicity. This will visually yield a stair case looking plot when compared with the continuous system. When you take the sampling rate to infinity of the analog system, the digital system acquired would be equivalent to the analog, continuous, system. This however, in reality, is not possible because even with gigahertz processors and higher, if all resources of the system went only to sampling, it could reach nowhere near infinity. However a super computer doing only sampling will sure look continuous until zooming in to values near its sampling rate.

The PID controller was selected as our best fitting for Phase II. Matlab was used to implement a ZOH at three different sampling values. The Z-transforms of each digital system at each sampling time are shown below along with the plots containing both the digital and analog systems. Note the effect the sampling time has on the response. The smaller the sampling time, the more accurate the response is (closer to the continuous time response).

Sampling time -- 1 second:

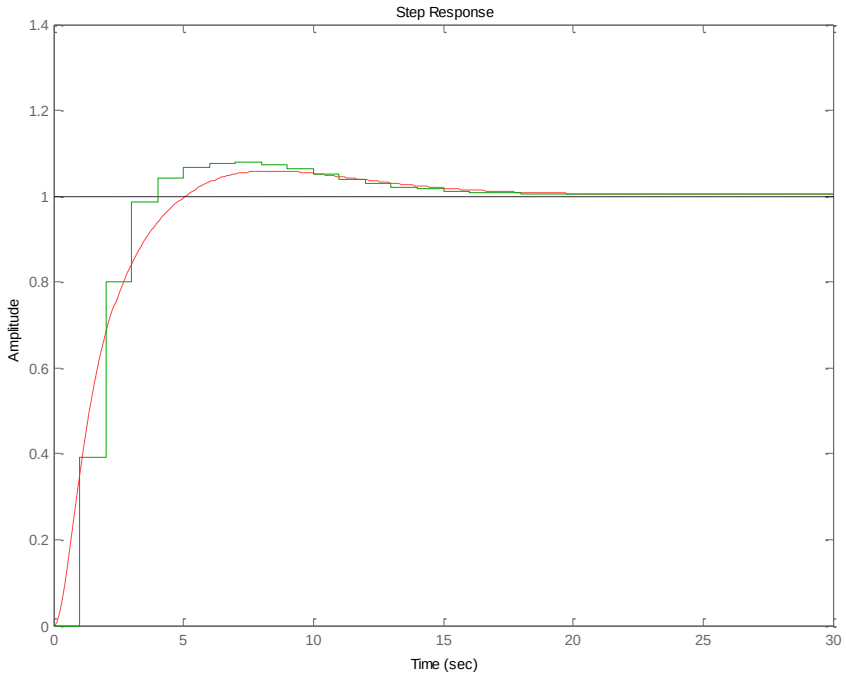
$$\frac{0.3921\,z^4 - 0.7063\,z^3 + 0.2544\,z^2 + 0.1249\,z - 0.06477}{z^5 - 3.235\,z^4 + 3.846\,z^3 - 2.017\,z^2 + 0.4372\,z - 0.03072}$$

Sampling time -- 0.5 second:

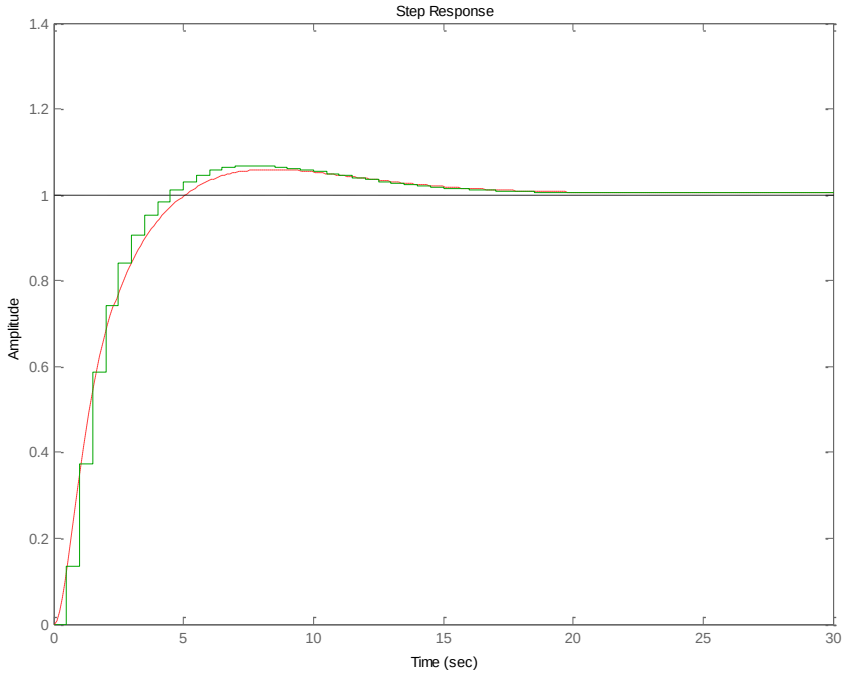
$$\frac{0.1358\,z^4 - 0.26\,z^3 + 0.06772\,z^2 + 0.1116\,z - 0.05515}{z^5 - 3.801\,z^4 + 5.605\,z^3 - 3.983\,z^2 + 1.354\,z - 0.1753}$$

Sampling time -- 0.1 second:

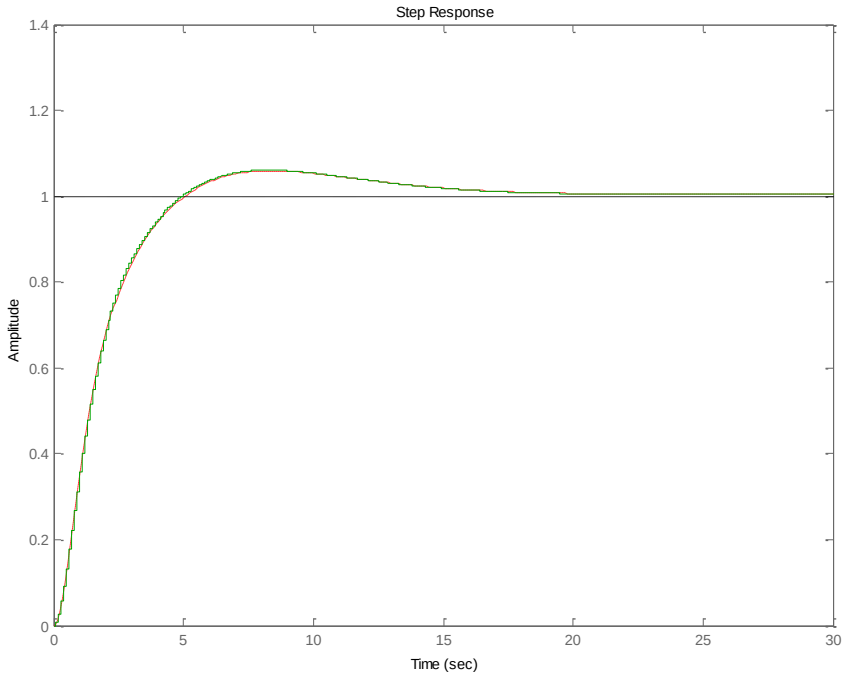
$$\frac{0.007372\,z^4 - 0.01464\,z^3 + 0.001047\,z^2 + 0.01238\,z - 0.006155}{z^5 - 4.679\,z^4 + 8.742\,z^3 - 8.154\,z^2 + 3.797\,z - 0.7059}$$



Sampling time 1 second



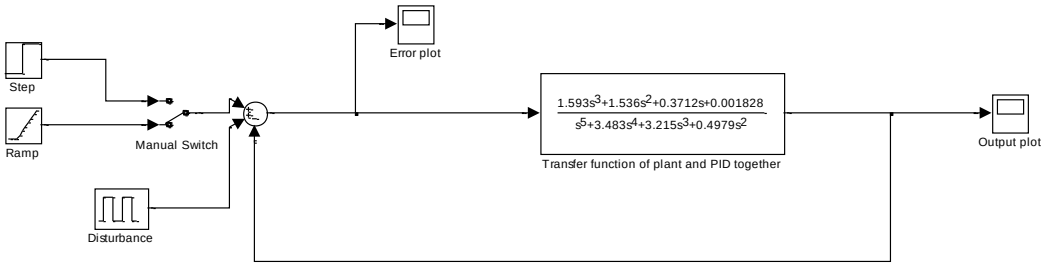
Sampling time of 0.5 seconds



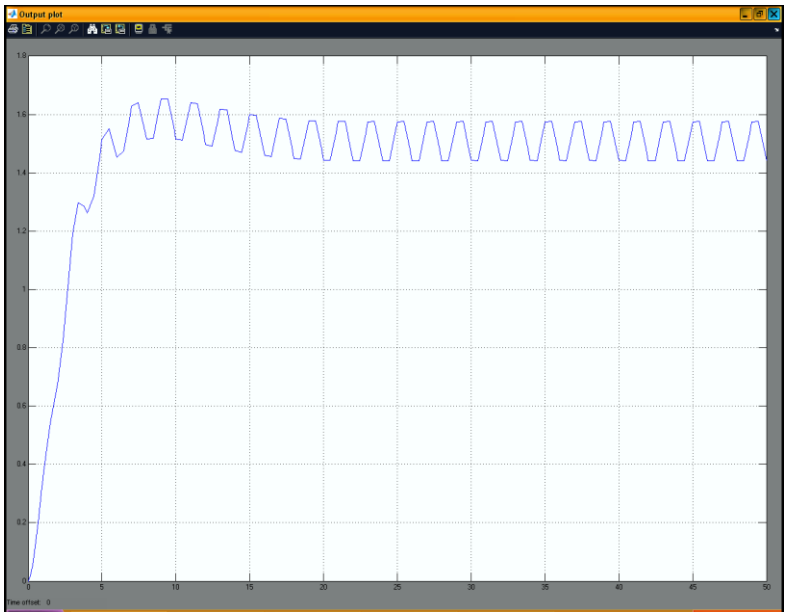
Sampling time of 0.1 seconds

Load Disturbance:

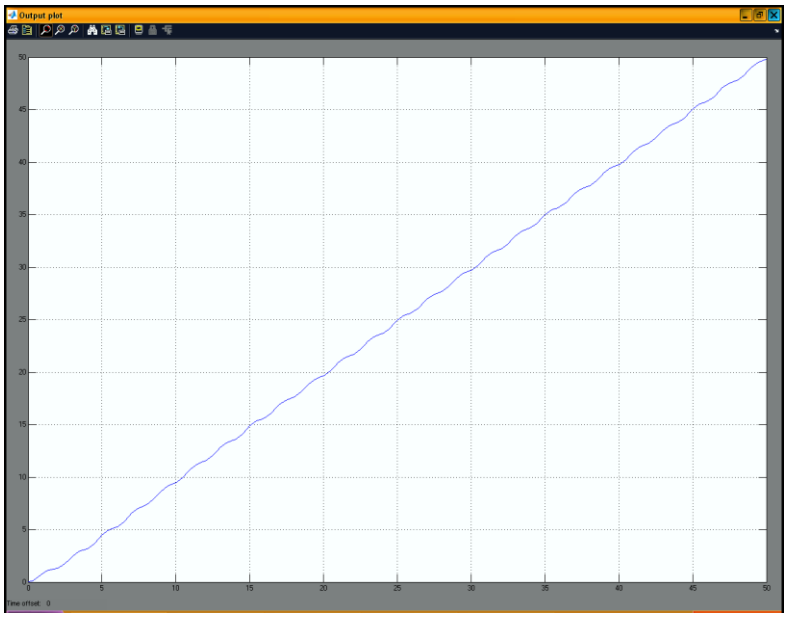
Below is the Simulink model used to simulate the load disturbance. The disturbance is a pulse.



Following are the plots of the step and ramp response with the disturbance added.



Step output response with disturbance



Ramp output response with disturbance

From the plots, it is noted that disturbance is bad. The step response never reaches a steady state; this is equivalent to having a constant percent over-shoot, which is bad. In the case of our system, which is for the heading control system of an unmanned free-swimming submersible vehicle; the vehicle would not be able to go in a straight line because of the disturbance.

Overall Conclusion:

This system seemed to be very picky as to what controller would tame it. It proved to be very difficult to get the controllers anywhere close to a steady state error of 0.05; the PID controller was the only one that achieved this with an excellent percent over-shoot. The rest of the controllers had increases in percent over-shoot as the steady state error got lower. It appears that, with our limited experience, the PID controller is the only one that effectively works on this system. The only downside was that it takes a long time to reach the steady state error of 0.

Group Meetings:

April 26, 2011

- Met with professor Rowland to discuss our system and decide whether to keep it or not.
- The decision was made to keep our phase I system for the heading control of an unmanned free-swimming submersible vehicle.

May 2, 2011

- Short meeting to assign each group member a controller to simulate
- Also discussed general methods of simulation

May 6, 2011

- After a busy week of tests, had a quick meeting to discuss our schedules for the upcoming week, when the report is due (May 12th).
- We concluded that most of our schedules were clear that upcoming week and that the bulk of the work would be done over the weekend and that week.

May 10, 2011

- Discussed controller problems with professor Rowland after class
- Meet as a group to see our Matlab progress

May 11, 2011

- Worked all day as a group, helping each other with last minute controller problems
- Afshin and Messan got help from professor Rowland on pole-zero placement for their phase lead and lag controllers.
- Chris talked to professor Rowland about his rate feedback controller and how it was not able to obtain a decent steady state error or percent overshoot. The conclusion was made, that this controller did not work well with the system and it was time to get the best results possible and move on.
- Matt's PID controller works well and seems to be the "Best" choice.
- Everybody completed a write up on their respective controllers

May 12, 2011

- Met in the morning to put everything together, proof read, and make final touchups
- Whew, I think we did it!