George Mason University

Review of Computer-Aided Control System Analysis and Design Software

Report from laboratory experiment 1, conducted on 30 January 2016
As part of ECE 429 Control Systems Lab
Course Instructor: Dr. Daniel M. Lofaro

Michael Kepler
G00804828

02 February 2016

The Volgenau School of Engineering

Introduction and Objective

MATLAB offers an extensive collection of commands and tools to help design and simulate control systems. The objective of this lab was to have the student explore these tools and commands. This was accomplished by having the student research the commands. Then having the student become more familiar with them by using those commands to design, implement, and analyze a 3rd order control system, in particular analyze the effects of different gains on a system.

Tasks

Task 1

Review the capabilities for control system analysis and design MATLAB commands:

Command	Description
bode	Plots magnitude and phase of frequency response. Can plot 'n' systems and return magnitude phase data for a given range or set of frequency values.
margin	Calculates minimum gain (return value is absolute, plot value is dB) and phase (degrees) margin and their associated frequencies for Single Input Single Output (SISO) open-loop (OL) systems.
rlocus	Calculates closed-loop (CL) pole trajectories as a function of gain, pole location, damping, overshoot, and frequency for 'n' SISO systems. Can also return pole locations and gain data, as well as, pole locations corresponding to a given gain.
rlocfin	Finds the root locus gains for a given set of roots.
series	Connects two systems in series. Both systems must be of the same type, i.e. both must be either continuous or both must be discrete (equivalent of multiplying two systems in the 'Laplace' domain). For multi-input multi-output systems, connections can be specified as desired.
parallel	Connects two systems in parallel. Both systems must be of the same type, i.e. both must be either continuous or both must be discrete (equivalent of adding two systems in the 'Laplace' domain). For multi-input multi-output systems, connections can be specified as desired.
feedback	Returns a system (SYS), that results from system 1 (SYS1) fed back with system 2 (SYS2). Negative feedback is default. (use '+' for positive feedback)
tf	Takes in two arrays, one of numerator coefficients and one of denominator coefficients, and optionally sampling period (Ts) and generates a system object in the Laplace domain. Can switch to 'z' domain using tf('z').
tfdata	Used to extract the sampling period, and numerator and denominator coefficients of a transfer function.
Logspace(a,b,n)	Generates a row vector of n logarithmically spaced points between 10^a and 10^b. (n is 50 by default)
semilogx	Plots data where the Y-axis is in linear units and the X axis is in logarithmic units.
step	Returns and plots the step response of 'n' systems for a given set of time values or time range.
lsim	Simulates the time response of linear systems with arbitrary inputs, where time, initial conditions, and method of interpolation can be specified.
roots	Returns the roots of a polynomial.
tzero	Returns the invariant zeros of MIMO systems.

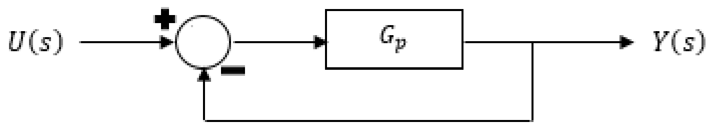
damp	Returns the damping ratio, natural frequency (expressed as the reciprocal of the time units of *sys*), and time constant of the poles of a system.
angle	Returns the phase angles (radians) for each element in a complex array.
abs	Returns the absolute value of a scalar or vector, and the magnitude of a complex number.
poly	Returns the coefficients of the characteristic polynomial whose roots are the element of the input (vector or matrix).
polyval	Returns the value(s) of a given polynomial, with coefficients defined in vector **p,** evaluated at a given value(s)
pzmap	Returns and plots the poles(as 'x's) and zeros('o's) for 'n' systems
find	Return the indices of the nonzero elements in array **x.** Use find(x==n) to find the indices of the elements containing 'n.'
unwrap	Adjusts phase angles to within a specified window by adding $\pm 2\pi$
linspace	Linspace(x1, x2, n) generates a row vector of n evenly spaced points from x1 to x2. (n=100 by default)
conv	Returns the convolution of two vectors. Can specify different portions such as 'same' for the central part of the convolution, or 'valid' for the computation without zero-padded edges.
size	Returns the sizes of each dimension of an array.
length	Returns the length of the largest dimension of an array.
real	Returns the real components of elements within a complex array.
imag	Returns the imaginary components of elements within a complex array.
sum	Returns the sum of column (default) vectors or row vectors.
prod	Returns the sum of column (default) vectors or row vectors.
eval	Executes the MATLAB code argument and returns the answer into a specified variable.

Task 2

The following system was to be controlled using a unity feedback configuration:

$$G_p(s) = \frac{K(s+10)}{s(s+0.4)(s+5)}$$

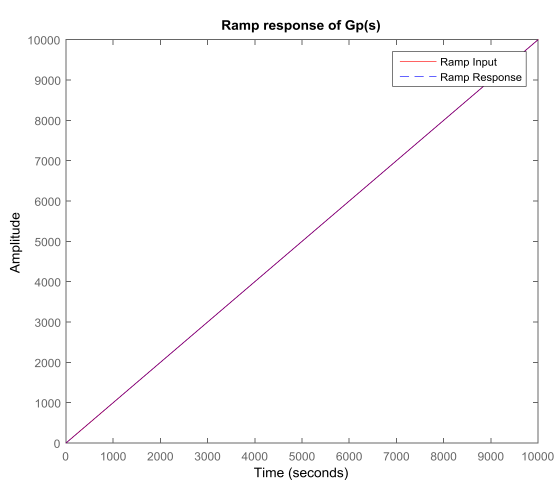
Unity configuration implies a feedback loop with a gain of -1.

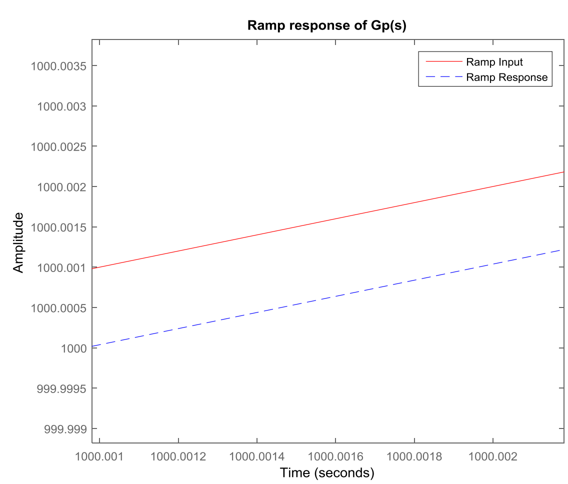


The gain that yields a steady state error of 0.001 was to be found.

The following is the MATLAB code and results of the system simulation:

```
%Generate the system Gp with unity feedback
s = tf('s');
k = 0.7350483;
Gp_ol = k*(s+20)/(s*(s+0.4)*(s+5));
Gp_cl = feedback(Gp_ol, 1);

%Plot the ramp response of the system and the ramp input
t = 0:1:10000;
figure(1)
plot(t,t, 'r')
hold on
step((Gp_cl/s),t, '--b');
title('Ramp response of Gp(s)')
```





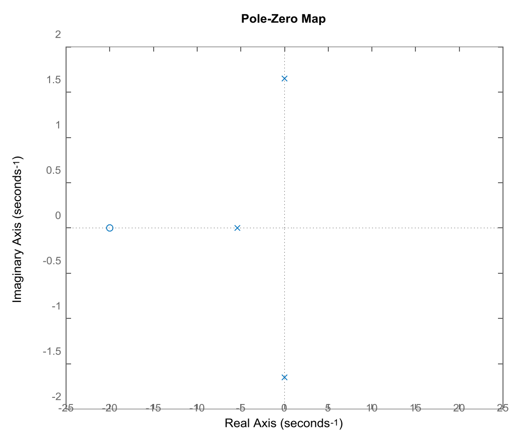
As the gain increased past a certain threshold the system became less and less stable, but as the gain decreased past this threshold the steady state error increased. After observing this relationship amongst the gain of the system, stability, and the steady state error, the gain was fine-tuned to this threshold through trial and error:

$$K = 0.7350483$$

to yield the steady-state error of approximately 0.001 as specified in lab manual. This can be observed along the Y-axis of the graph and noting the ~0.001 difference along the axis between the ramp input and the ramp response.

Next the poles for the gain above were found to determine the stability of the system:

```
%Find the poles of the closed loop system
figure(2)
pole(Gp_cl)
pzmap(Gp_cl)
```



With the poles located at:

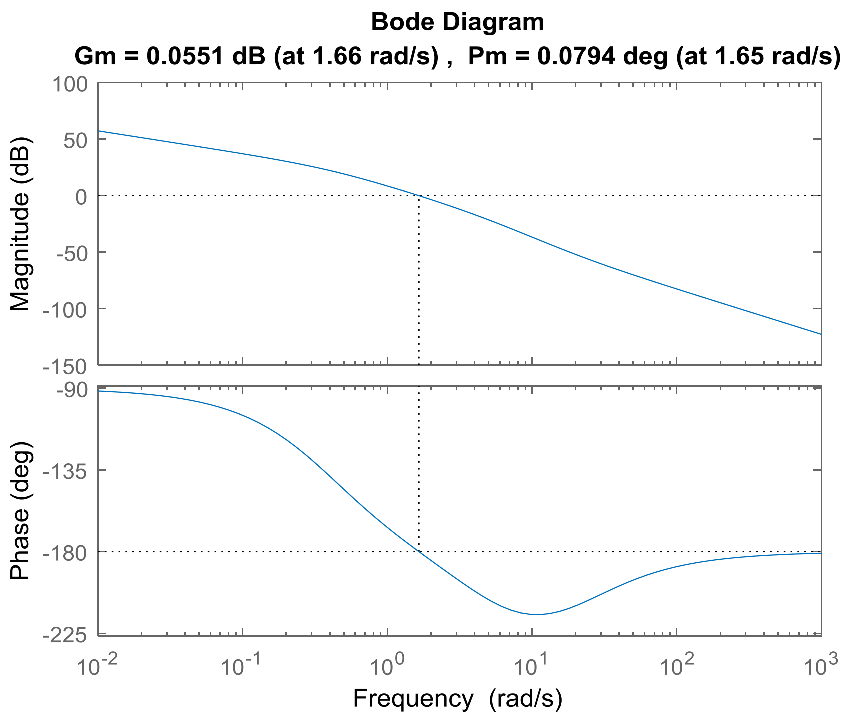
```
ans =

  -5.3979 + 0.0000i
  -0.0011 + 1.6503i
  -0.0011 - 1.6503i
```

Given that all of the poles of the system are located on the left-half plane (LHP) of the imaginary axis, the system was stable.

Task 3

The Bode plots for the magnitude and phase of the open-loop system were obtained and the phase and gain margins were found:

```
%Obtain the bode plot for the open loop system of Gp
figure(3)
margin(Gp_ol)
[Gm, Pm, Wcm, Wcg] = margin(Gp_ol)
```



$Gain\ Margin = 0.0551\ dB = 1.0064$

Located at 1.6552 rad/s

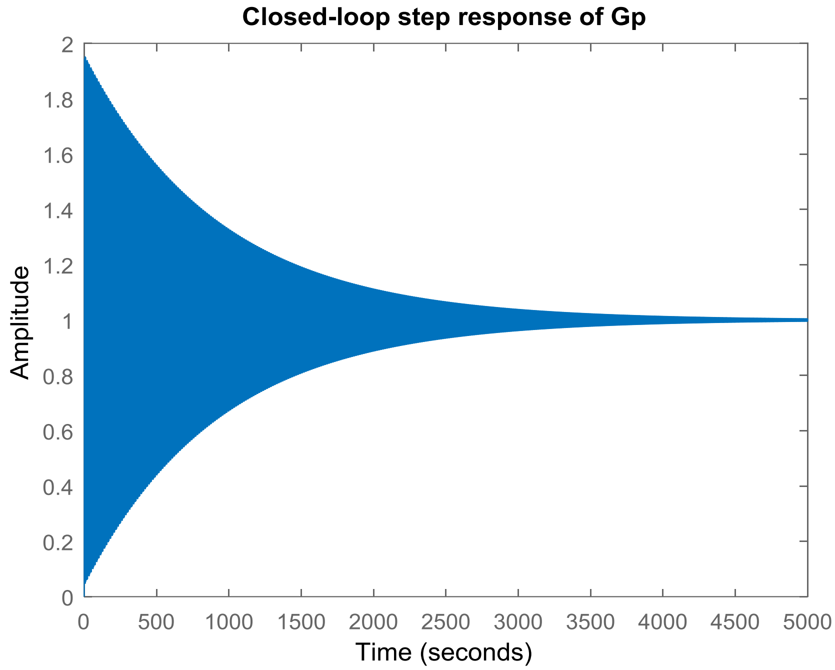
$Phase\ Margin = 0.0794\ degrees = 0.0028\ radians$

Located at 1.6501 rad/s

Since the gain margin (the amount of gain required to achieve unity gain at the frequency corresponding to the -180 phase mark) was not negative the system was stable. This was consistent with the pole locations being within the LHP that were found in *task 2*.

Next, the step response of the closed-loop system was plotted:

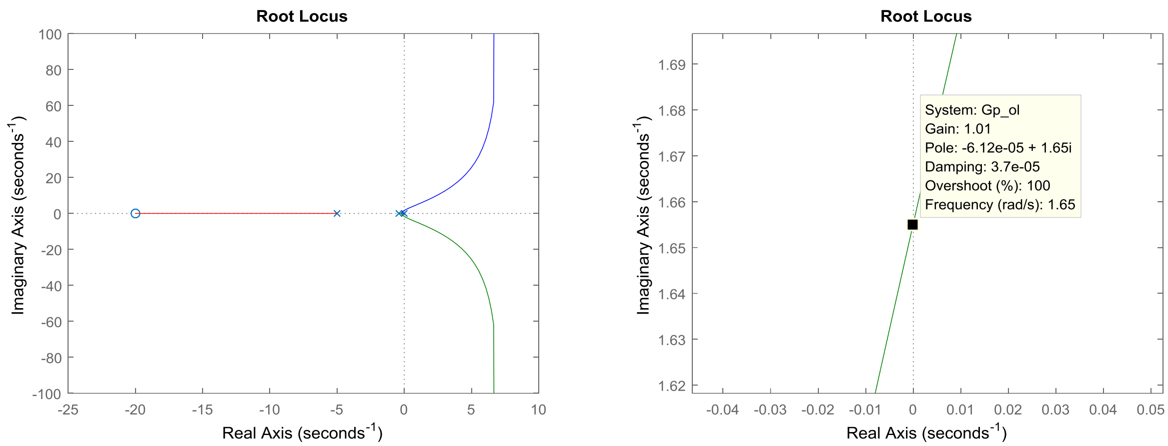
```
%Plot the closed-loop step-response
figure(4)
step(Gp_cl)
title('Closed-loop step response of Gp')
```




```
ans =

        RiseTime: 0.6772
    SettlingTime: 3.6095e+03
     SettlingMin: 0.0483
     SettlingMax: 1.9518
       Overshoot: 95.1797
      Undershoot: 0
            Peak: 1.9518
        PeakTime: 5.8027
```

Task 4

The value of the gain K, where the branches of the root locus crossed the imaginary axis, and the frequency value at this crossing were determined.



$$K = 1.01 \qquad Frequency = 1.65\ rad/s$$

As can be observed from the plots, the gain was 1.01 and the frequency 1.65 rad/s, where the branches crossed the imaginary axis. Both values were in accordance with the gain margin 1.0064 at the frequency of 1.6552 rad/s found in *task 3*. If the poles were in the RHP, then the system would be unstable. This is why the gain and frequency of the poles along the imaginary axis coincide with the gain margin

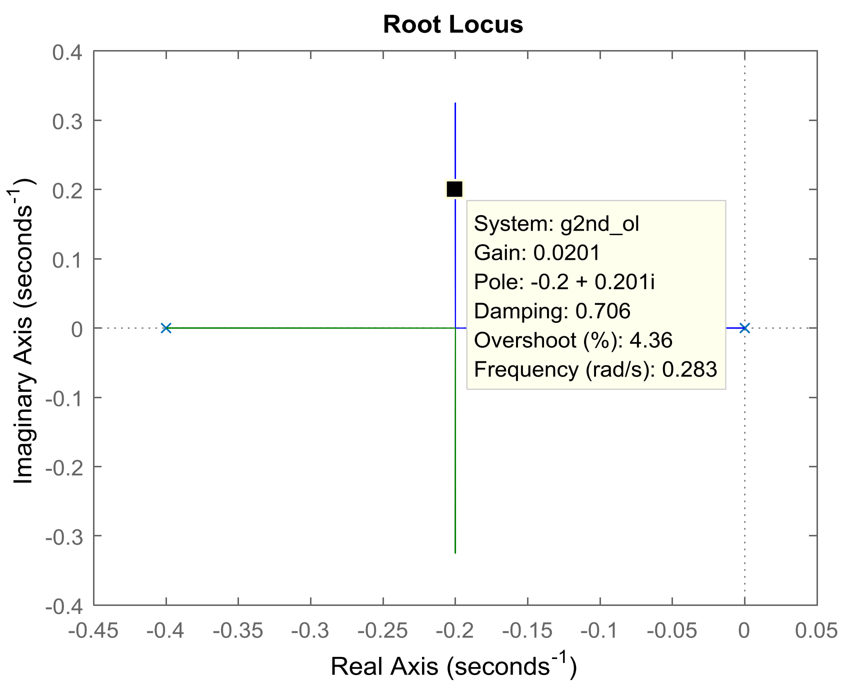
Task 5

Next, the system G_p was approximated as a 2nd order system:

$$G_p(s) \approx \frac{4K}{s(s+0.4)}$$

Then the K was selected such that the damping ratio $\zeta = 0.707$. This was done by plotting the root locus and observing the damping ratio at different gains:

```
%Approximate Gp to a 2nd order system
k = 1;
g2nd_ol = k*4/(s*(s+0.4));
figure(7)
rlocus(g2nd_ol, 0:0.00001:1.01)
```



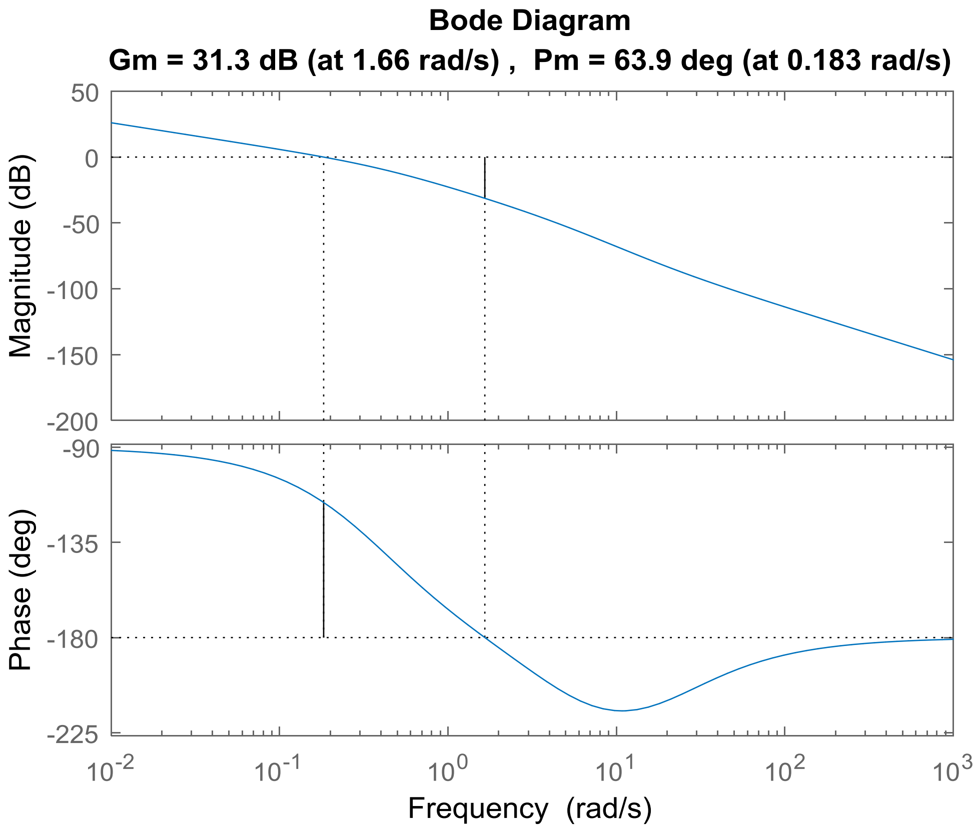
$$K = 0.0201$$

Applying this new value of K to the complete transfer function, the following equation was obtained:

$$G_p(s) = \frac{0.0201(s+20)}{s(s+0.4)(s+5)}$$

To determine the closed-loop stability of the system, the gain and phase margins were calculated:

```
%The gain and phase margins at the new value of K
s = tf('s');
k = 0.0201;
Gp_ol = k*(s+20)/(s*(s+0.4)*(s+5));
figure(2); margin(Gp_ol)
[Gm, Pm, Wcm, Wcg] = margin(Gp_ol)
```

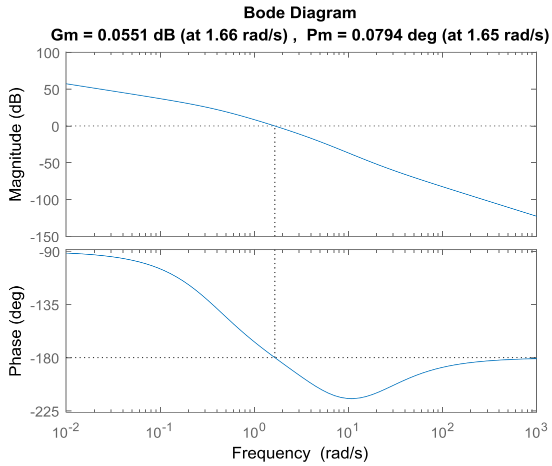


$Gain\ Margin = 31.3\ dB\ = 36.8024$

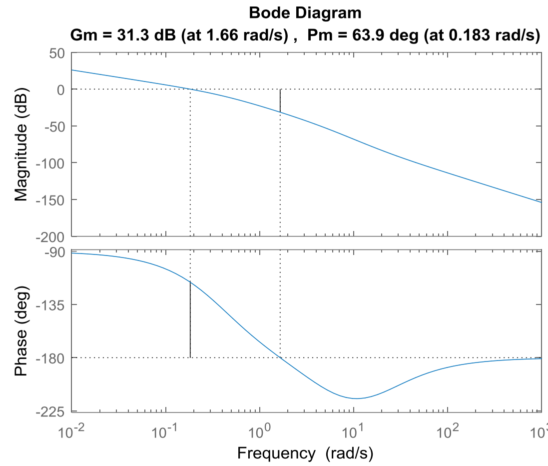
Located at 1.6552 rad/s

$Phase\ Margin = 63.9\ degrees = 1.1153\ radians$

Located at 0.1827 rad/s



[K = 0.7350483]



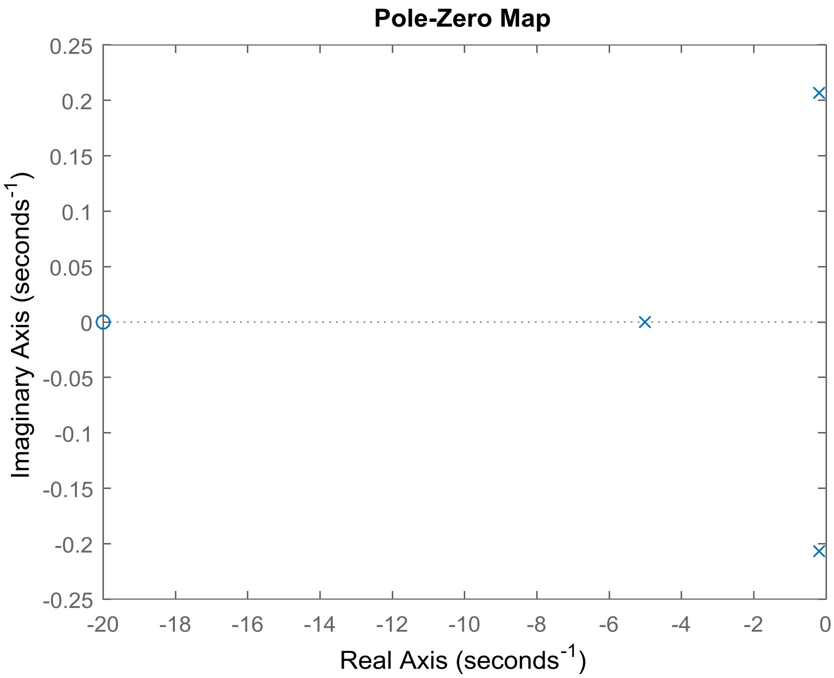
[K = 0.0201]

The table below summarizes the effects that changing the gain has on the Bode plots, as well as, the gain and phase margins:

	Change of Gain (K)	
	Decrease	Increase
Magnitude Bode plot shift	Down	Up
Phase Bode plot shift	None	None
Gain Margin Crossover Freq.	None	None
Phase Margin Crossover Freq.	Decrease	Increase
Gain Margin	Increase	Decrease
Phase Margin	Increase	Decrease

The stability of the system can be determined by obtaining the location of the closed loop poles of the system:

```
%Getting closed-loop pole locations for stability analysis
Gp_cl = feedback(Gp_ol, 1)
figure(3)
pole(Gp_cl)
pzmap(Gp_cl)
```

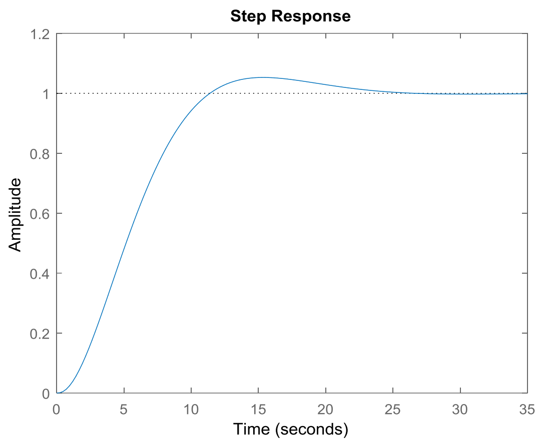


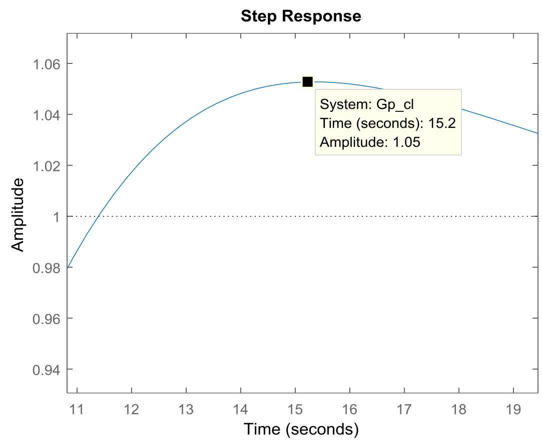

```
ans =

  -5.0130 + 0.0000i
  -0.1935 + 0.2068i
  -0.1935 - 0.2068i
```

Note that the pole locations were all within the LHP, confirming the system was stable.

Finally the step and ramp responses of the system were observed:

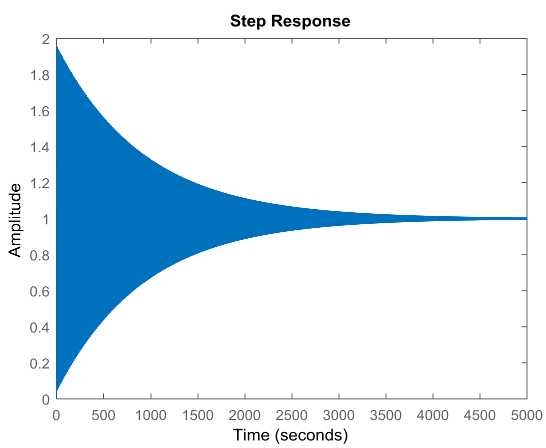
```
%Obtain step and ramp response
k = 0.0201;
Gp_ol = k*(s+20)/(s*(s+0.4)*(s+5));
Gp_cl = feedback(Gp_ol, 1)
%Step Response
stepinfo(Gp_cl)
figure(1); step(Gp_cl);
%Ramp Response
figure(2); step(Gp_cl/s); hold on
t=(0:0.1:3500)
plot(t,t,'--g')
title('Ramp Response')
legend('Ramp Response','Ramp Input')
```



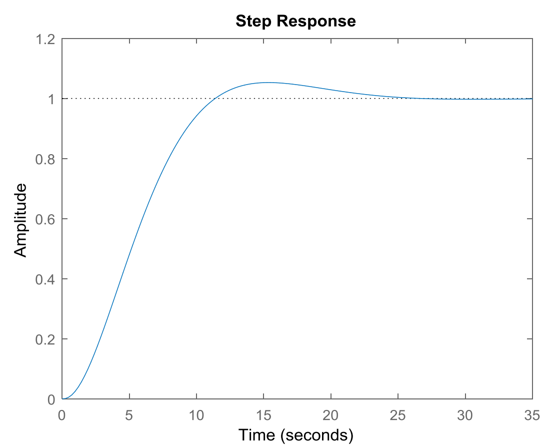



```
ans =

     RiseTime: 7.3446
 SettlingTime: 21.3509
  SettlingMin: 0.9012
  SettlingMax: 1.0528
    Overshoot: 5.2765
   Undershoot: 0
         Peak: 1.0528
     PeakTime: 15.2326
```

Maximum overshoot was calculated by *stepinfo()* as 5.2765.



[K = 0.7350483]



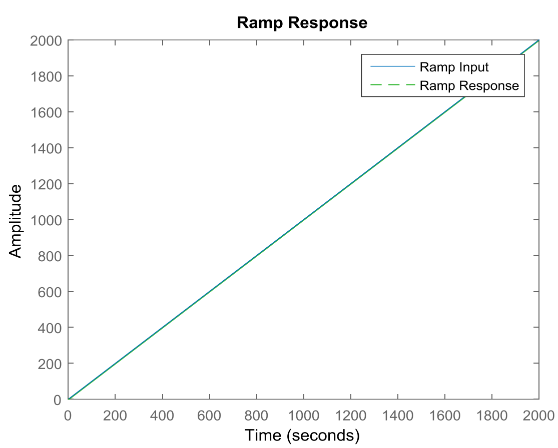
[K = 0.0201]

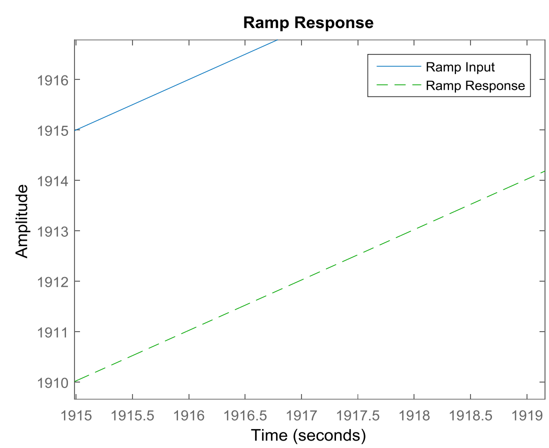
```
ans =

       RiseTime: 0.6772
   SettlingTime: 3.6095e+03
    SettlingMin: 0.0483
    SettlingMax: 1.9518
      Overshoot: 95.1797
     Undershoot: 0
           Peak: 1.9518
       PeakTime: 5.8027
```

```
ans =

       RiseTime: 7.3446
   SettlingTime: 21.3509
    SettlingMin: 0.9012
    SettlingMax: 1.0528
      Overshoot: 5.2765
     Undershoot: 0
           Peak: 1.0528
       PeakTime: 15.2326
```

Upon comparison, it was observed that as the gain decreased from 0.735 to 0.0201, overshoot significantly decreased, but settling time, rise time, and peak time all increased for a step input.

The ramp response is as follows:





The steady state error was observed along the Y-axis of the graph. There was a difference of 5 between the ramp input and the response, resulting in a steady-state error of 5. Note that as the gain decreased the steady-state error increased, as was mentioned previously in *task 2.*

Discussion

MATLAB simplifies control system design by offering an assortment of commands that ease the designing, creation, and implementation processes of control systems, from the commands that present different stability considerations, to the plots that allow for design based on closed-loop poles, damping factor, or gain.

After performing this lab experiment, the large influence that the gain has on system performance and stability has become very apparent. As mentioned within the report, if the gain is too large the system will become unstable, but if the gain is too small, the steady-state error in response to a ramp input may be too large. Most control system designs entail some element of compromise in performance, and the tools MATLAB offers makes it easier for the user to get the most benefit for the least amount of compromise.