ENB301: Practical Report

Lachlan Nicholson (n8866864) Declan Gilmour (n8871566)

25th May, 2015

1 Executive Summary:

This report provides the documentation and analysis of three closely associated practical experiments; the open loop system, the closed loop motor control system, and the effects of varying the systems gain and input signal frequency. The methods of analysis conducted within this document include thorough mathematical estimations and computational simulations. The results of the previously mentioned experiments, beyond showing clearly the benefits and application of closed loop control systems, conclusively show that the input frequency does not effect the overshoot or settling time of the closed loop system, but the gain is proportional these properties. Moreover, the difference in theoretical calculations and practical applications due to un-modeled system interactions was shown in depth.

2 Aim:

The purpose of this report is to demonstrate sound knowledge and understanding of basic control systems; more specifically, closed loop feedback systems. This report entails (beyond other things) the procedures followed during the practical lab, the associated results or observational data, and the answers to all questions posed during the lab.

3 Introduction:

Part A was the the laboratory work performed by each individual member of the group. Basic analysis of the of an example set of servo motor was used.

Part B considered an open loop servo motor system and used the prelab completed in part A to calculate the K_m and α values.

Part C and D considered the close loop response and a detailed examination of closed loop aspects were investigated. A PID controller was used to optimise the system.
Time response characteristics were also analysed.

Contents

List of Figures

4 Acknowledgments

Two groups worked in combination during the lab:

1. Lachlan Nicholson (n8866864) and Declan Gilmour (n8871566).
2. Thomas Wagner (n8840121) and Antony Foster (n8647780).

Work related to the prac is set up in three parts:

a) **Pre-lab**
All individual prelab responses and code incuded in this report has been completed solely by Lachlan Nicholson and Declan Gilmour respectively.

b) **Lab**
Code used within the lab predominanly belongs to Antony Foster and Thomas Wagner. All code beloning to this group was also performed by Declan Gilmour and Lachlan Nicholson and included in this report.

c) **Post-lab**
All work performed post lab were produced in combination of Declan Gilmour and Lachlan Nicholson. This section contains all matlab code used during the practical lab. Lachlan Nicholson (n8866864) and Declan Gilmour (n8871566) worked in a prac group with Thomas Wagner (n8840121) and Antony Foster (n8647780).

5 Procedure:

This section consists of individual summaries of the procedures required in each section, accomplished during the practical labs. Briefly cover what each part of prac is to accomplish.

5.1 Experiment B

Experiment B utilizes experimentally measured time response data to calculate approximate values for K_m and α. The pre-lab preparation for this section of the practical required familiarization with all aspects of the oscilloscopes and their functions (triggers, scales, saving data); moreover, it was suggest that the user should also be familiar with the construction process outlined within the coming procedures. In an attempt to better show the procedures of this particular experiment, the required tasks will be separated into two categories; the setup, and the analysis. Both of the aforementioned sections will be displayed in numbered point formation to show clearly the extent of each step.

SETUP:

1. The dual power supply was setup in independent mode, with 5.0 V and 2.0 V respectively. (measured using a multimeter to ensure accuracy)
2. Next, the 5V supply was connected across the potentiometer (outer wires), whilst using a multimeter to measure the wiper voltage (middle wire). Moreover, turning the flywheel clockwise increased the wiper voltage; however, if the inverse was true, the 5V and 0V wires would have been swapped.
3. After having adjusted the flywheel to the center position, and after connecting the 0V rail to one side of the motor; the circuit was briefly connected using the 2V supply. The polarity of the two connections were then adjusted to ensure the motor moves in a clockwise direction when voltage is applied.
4. The flywheel was returned to the central location whilst the wiper voltage of the potentiometer and the positive terminal of the motor were measured using the digital CRO. The trigger was set to 0.5 V and a USB inserted.
5. After briefly completing the motor circuit, the CRO was triggered and recorded the response of the potentiometer and input voltage. The channel gain was inspected to ensure it had been set correctly and all important regions of the response can be seen, then the data was saved to the USB.
6. The previous step was repeated multiple times to ensure viable data had been collected.

ANALYSIS:

1. After the experimental data had been saved to a USB and transfered to a computer, it was then plotted in matlab.
2. The experimental data was also plotted against the systems estimated transfer function $y_1(t)$.
3. Using a robust self constructed function to estimate the system parameters according the experimental data, approximate values for K_m and α were obtained.
4. The system transfer function $y_1(t)$ was then adjusted, and another diagram constructed to compare the experimental and calculated systems response.
5. Lastly, using the second order model for the motor and the previously calculated α value; K_m and β were approximated.

5.2 Experiment C

Section C utilized the open loop response of the motor that had been measured and approximated in the previous procedures to achieve a closed loop control system for the position of the motor (voltage across the potentiometer). The pre-lab component of experiment C required the extensive analysis of a provided circuit diagram; breaking the system down into functional elements of the control system, calculating individual transfer functions for these elements, and constructing an overall transfer function for the system ($G_c(s)$). As mentioned previously, the procedures followed in this experiment will also be separated into the setup or preparation, and the analysis.

The pre-lab aspect of this experiment required the operator to be familiar with typical breadboard design strategies, the pin-out of the op-amp used within the experiment, common resistor code colours, and the use of noise mitigation capacitors.

Refer to figure 22 for the complete closed loop motor control system schematic used in the following procedures.

SETUP:

1. The function generator was setup to output a 0.1 Hz square wave an amplitude of 0.5 V.
2. The aforementioned circuit was constructed, but the motor was only connected after taking multiple measurements to ensure the circuit was operating as expected.
3. With the motor connected, the response of the system was captured and saved by the digital CRO; this step was repeated multiple times to ensure accuracy.
4. The gain was then adjusted to produce a 5% overshoot, the resistor values used and the systems response were recorded.

ANALYSIS:

1. The collected data was imported into matlab, and compared against the predicted model derived previously in **part A and B**.
2. After which, the experimentally found gain required to produce an overshoot of 5% was compared against the predicted gain value.
3. Lastly, a discussion took place surrounding the use of alternate methods to derive the open and/or closed loop response for the system using the same equipment.

5.3 Experiment D

The final experiment, using the same circuit constructed in the previous experiment, examined the response of the system to different input frequencies, the impact of gain on the systems response, and the implementation of a new method to approximate the open and/or closed loop response.

ANALYSIS:

1. In an attempt to estimate the closed loop response of the system post-lab, using only the available equipment; the input signal was set to a frequency sweep in order to create a bode plot from the resulting output data. The data was saved to the USB.

2. Using the same circuit constructed in the previous procedure (system gain set at the experimentally found gain required to produce a 5% OS) the overshoot was measured and recorded, for input frequencies of 0.5Hz, 0.75Hz, 1Hz, 1.25Hz and 1.5Hz.

3. Additionally, the systems response for each frequency was also recorded and saved in-case of future need.

4. The impact of altering the gain was then examined, as the response of the system was recorded twice more; using a new gain value each time.

5. The closed loop system was then simulated in Simulink, using the model parameters determined in the previous experiment. A group discussion was raised over the quality of the model, the results of which can be found in section 8.2.3.

6. Another group discussion began, as a proposal for a PID controller that would achieve a faster response was formed and simulated. Refer to the previously mentioned section for the in depth proposal.

6 Results:

summary of what you observed in parts B,C,D (less than 2 pages per part), noting detailed answers are to be provided in the appendix.

6.1 Part B

calculate predicted K_m and α values, correct polarity of the motoraboveB5

1. The csv data files were loaded into matlab and stored in respecive vectors. The time vectors started at $t < 0$.
   All data began on the third row of the CSV files and the following code was used to import all data but the first two rows:

```
data = csvread('PartB_Test1.csv',2,0);
```

   To relocate the data starting at t = 0 the following matlab code was used:

```
te_1 = te_1 + abs(te_1(1));
```

2. Simulated motor response vectors were created following based on pre-lab code. This was then compared to the experimental data previously loaded with the same time vector.
   The experimental data was offset at 2 volts due to the step input. y_e was then shifted to start at zero for comparison with simulated motor response.

```
plot(te_1,ye_1-ye_1(1),'b')
```

3. Two different figures of merit were considered to compare outputs and eventually find K_m and α values. The error between the simulated response and experimental response can be calculated through use of root mean square and mean square comparisons.

```
% Calculate mean square error
error_ms = error_ms + (G_0(jj) - ye(jj))^2;

% Calculate root mean square error
error_rms = error_rms + rms(G_0(jj) - ye(jj));
```

4. The use of figures of merit has then been applied to testing a range of K_m and α values to optimise the simulated response. The simplified input voltage to motor shaft equation was used for this process: $G_o(s) = \frac{V_p(s)}{V_m(s)} = \frac{K_m}{s(s+\alpha)}$
Each K_m, α and error value were stored in output vectors.

```
for km = linspace(0,km_max,km_num)    % Cycle km values
   for alpha = linspace(0,alpha_max,alpha_num)  % Cycle alpha values
       G = tf(km, [1 alpha 0]);
       G_0 = step(G,te);

       % Calculate error
       for jj = 1 : length(te)
          % Calculate mean square error
          error_ms = error_ms + (G_0(jj) - ye(jj))^2;

          % Calculate root mean square error
          error_rms = error_rms + rms(G_0(jj) - ye(jj));
       end

       % Store km, alpha and the error taken to calculate
       output_ms(:,ii) = [km;alpha;error_ms];
       output_rms(:,ii) = [km;alpha;error_rms];

       % Reset cycle variables
       ii = ii + 1;
       error_ms = 0;
       error_rms = 0;
end
```

The K_m and α values were then chosen by selecting the entry in the output vector with the smallest error.

Due to an ossilliscope malfunciton, the spacing between time increments in some points was not always consistant. This spacing was fixed through use of the following Matlab Code:

```
clear te_cpy g te_new
te_cpy=te;
g=0;
ye_cpy=ye;
for i=1:(length(te_cpy)-1)
    if ((round(abs((te_cpy(i)-te_cpy(i+1))),3))==0.001)
        te_new(i+g)=te_cpy(i);
        ye_new(i+g)=ye_cpy(i);
    else
        i;
        te_new(i+g)=te_cpy(i);
        ye_new(i+g)=ye_cpy(i);
        te_new(i+g+1)=te_cpy(i)+0.001;
        ye_new(i+g+1)=(ye_cpy(i)+ye_cpy(i+1))/2;
        g=g+1;
    end
end

te_new(i+g+1)=te_cpy(end);
ye_new(i+g+1)=ye_cpy(end);
te_new=transpose(te_new);
ye_new=transpose(ye_new);
```

The following code was then used to optimisations was then performed on the experimental vectors.

```
% Remove initial zero gradient before resonse
f = ye_new > ye_new(1) * 1.05;
indice = find(f,1,'first');
ye_new = ye_new(indice:end);
te_new = te_new(indice:end);

% Shift time to start at zero
te_new = te_new - te_new(1);

% Shift amplitude to start at zero
ye_new = ye_new - ye_new(1);

% Look for max
indice = find(ye_new == max(ye_new));
indice = round(indice * 0.85);
ye_new = ye_new(1:indice);
te_new = te_new(1:indice);
```

The experimental data vectors were optimised before calculations were performed and then saved as new vectors.

```
[te_1trim, ye_1trim] = trimForCalculation(te_1new,ye_1new,mf1);
[te_2trim, ye_2trim] = trimForCalculation(te_2new,ye_2new,mf1);
[te_3trim, ye_3trim] = trimForCalculation(te_3new,ye_3new,mf1);
```

The vectors were first trimmed to only perform calculations between the first and third quartile. From here the two vectors were smoothed using medfilt1.

```
function [ te_new,ye_new ] = trimForCalculation(te,ye, mf1)
    indice = round(length(te) / 4);

    % Output vectors between 1/4 and 3/4 of inputed vectors
    te_new = te(indice:indice*2);
    ye_new = ye(indice:indice*2);

    % Smooth vectors using filtering
    medfilt1(te_new,mf1);
    medfilt1(ye_new,mf1);
end
```

Three sets of experimental data were used with two figure of merit calculations. The average K_m and α values were then taken from the 6 optimised values respectively.
$K_m = 422.0107$ and $\alpha = 170.3407$

```
km_mean = (mean(km_ms) + mean(km_rms)) / 2;
alpha_mean = (mean(alpha_ms) + mean(alpha_rms)) / 2;
```

It should be noted, during the practical Lachlan Nicholson (n8866864) and Declan Gilmour (n8871566) worked with another group of two students Tony Foster (n8647780) and Tom Wagner (n8840121). A different matlab script belonging to these two students was used to find K_m and α. The code use to do so is attached in the appendix of this report.
From the code used during the prac it was found $K_m = 326$ and $\alpha = 38.61$

5. The α value found above was then used to calculate K_m and β values. The general transfer function of input voltage to motor shaft velocity was then used: $G(s) = \frac{K_m}{(s+\alpha)(s+\beta)}$.
The same process as in part 4 of this section was used, with three sets of data and two different figure of merrit calculations to find the optimum K_m and β values:

```
for km = linspace(0,km_max,km_num)    % Cycle km values
   for beta = linspace(0,beta_max,beta_num)  % Cycle alpha values
       G = tf(km, [1 (alpha_mean+beta) (alpha_mean*beta)]);
       G_0 = step(G,te);

       % Calculate error
       for jj = 1 : length(te)
          % Calculate mean square error
          error_ms = error_ms + (G_0(jj) - ye(jj))^2;

          % Calculate root mean square error
          error_rms = error_rms + rms(G_0(jj) - ye(jj));
       end

       % Store km, alpha and the error taken to calculate
       output_ms(:,ii) = [km;beta;error_ms];
       output_rms(:,ii) = [km;beta;error_rms];

       % Reset cycle variables
       ii = ii + 1;
       error_ms = 0;
       error_rms = 0;
   end
end
```

$K_m = 426.6867$ and $\beta = 0.0646$. When considering the eneral transfer function of input voltage $G(s) = \frac{K_m}{(s+\alpha)(s+\beta)}$, a very small β value would turn this into the simplified input voltage to motor shaft transfeer function $G_o(s) = \frac{V_p(s)}{V_m(s)} = \frac{K_m}{s(s+\alpha)}$.

Therefore, the β value of 0.0646 suggests the original assumption of the simplified input voltage to motor shaft transfer function was correct. The useful approximation if $|\beta| < 10|\alpha|$ the transient response will be dominated by α satisfies this system.
In otherwords, the electrical constant β is negligable.

6.2 Part C

As mentioned previously, the goal of experiment C was to construct a closed loop control system (feedback) for the position of the motor (voltage across the potentiometer); making use of the estimated open loop response variables found in part B (K_m and α).

The pre-experimental aspect of this section required the breakdown and analysis of the circuit diagram captured in figure 22, the full method and calculations has been provided in section 8.2.2; the in depth analysis resulted in an estimated overall closed loop transfer function for the system.

$$G_c(s) = \frac{\frac{R_f}{R_1}K_m}{s^2 + s\alpha + \frac{R_f}{R_1}K_m}$$

$$G_c(s) = \frac{1075.8}{s^2 + 38.61s + 1075.8}$$

After constructing the circuit shown in figure 22 and asserting the correct polarity of the motor, the response of the system was captured and saved by the digital CRO. The closed loop response of this system was measured, a capture of the scope can be found below, whilst the graphical representation of this data has been included in figure 28.

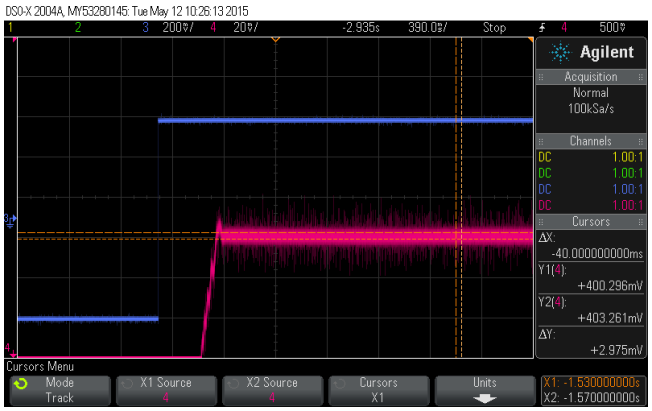


Figure 1: Theoretical Closed Loop Response

Again, refer to the appendix for a detailed analysis, however; the measured overshoot for the calculated values was under the expected 5%. Additionally, below is the recorded systems response after experimentally changing the resistor values, and hence the gain, to produce a 5% overshoot. The experimental resistor values used to produce this overshoot were as follows; $R_1 = 10k$, $R_f = 62.8k \approx 56k + 6.8k$, resulting in a gain of $K = 6.28$.

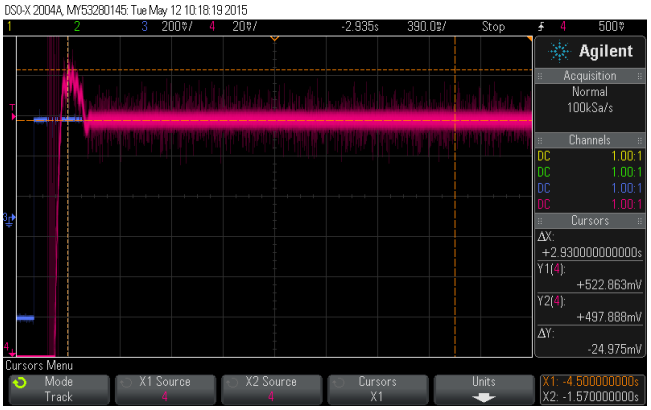


Figure 2: Experimental Closed Loop Response

From the figures above, it can be seen that whilst the circuit exhibits the expected features of a closed loop transfer function of this type, there is also a large difference between the expected and actual overshoot resulting from the calculated resistor values. In section 8.2.2, the $\%_{error}$ was calculated to be 61.8%, where the difference in gain between the theoretical and practical values is likely caused by un-modeled losses such as gearbox, friction.

new method
comparisons
C8

6.3 Part D

Experiment D explored the effect of input frequency on the closed loop system response, as well as examining the effect of increasing or decreasing the systems gain. and bode

D1/C8

After measuring and recording the systems response and percentage overshoot for five different input frequencies; the following table was constructed (refer to section 8.2.3 for calculations responsible for the construction of this table).

Input Frequency (Hz)	Overshoot (%)
0.5	6.1
0.75	5.7
1	6.1
1.25	6.5
1.5	5.8

Both the above table, and the systems response (figure 31) show clearly that increasing the frequency has no effect on overshoot, settling time, or steady state error. However, after the frequency exceeds a certain limit ($1/f < Ts$) the system does not have enough time to reach a steady state value. Moreover, if the frequency of the input where to continue increasing, eventually the operation amplifier will begin attenuating the output voltage.

$$G_c(s) = \frac{\frac{R_f}{R_1}K_m}{s^2 + s\alpha + \frac{R_f}{R_1}K_m}$$

and recall,

$$W_n = \sqrt{b} = \sqrt{K_m * \frac{R_f}{R_1}}$$
$$\zeta = \frac{a}{2b} = \frac{a}{2 * W_n} = \frac{\alpha}{2 * \sqrt{K_m * \frac{R_f}{R_1}}}$$

and

$$\%OS = e^{\frac{-\pi\zeta}{\sqrt{1-\zeta^2}}} * 100$$
$$T_s = \frac{4}{\zeta W_n}$$

Thus, as neither W_n or ζ are dependent on frequency, both overshoot and settling time are also independent of frequency. This explains why besides the reduced time available for the signal to settle, no visible changes to these properties were observed as the frequency was increased.

The impact of the systems gain however, was much more noticeable. From figure 33, it can be seen that increasing the gain resulted in an increased overshoot and settling time. This has been proven more in depth in the associated answers section, but a summary of the findings can be found below. From the previous equations, it can be seen that both ζ and W_n rely on the gain ($K = R_f/R_1$). Furthermore, this means both the overshoot and settling time rely on the gain factor. This is incredibly useful in control systems applications, as changing the gain alone can alter system properties such as overshoot and settling time.

Refer to Appendix for PID controller information.

7 Discussion:

Throughout the practical experiment, many important observations and analytical breakthroughs had taken place; these observations included the system response of an open loop motor control system and how fundamental properties of the estimated transfer function affect the simulated response. Moreover, the practical application of closed loop control systems were explored, and the advantages of using a closed loop system became evident, specifically in relation to controlling a servo motors shaft angle output. In addition to this, the effects of varying the input frequency where found to be minimal, whilst altering the systems gain provided a unique method of controlling system properties (such as overshoot, and settling time).

Many further analysis procedures were conducted to show relationships such as how a PI compensator can be utilized to remove a systems steady state error; or how the more flexible PID controller can alleviate steady state error and provides the ability to manipulate and control the systems response to a greater degree.

However, in future the following improvements to the program could be taken into consideration;

1. A longer and more thorough introduction to some of the more technical features matlab has to offer.
2. Reduce the complexity of the practical layout.
3. Introduce assessed weekly lab sessions devoted to experiment preparation.

8 Appendices

8.1 Pre-lab:

8.1.1 Lachlan Nicholson

PART A ANSWERS:

1. Below is the requested functional diagram for the complete servo motor control system.

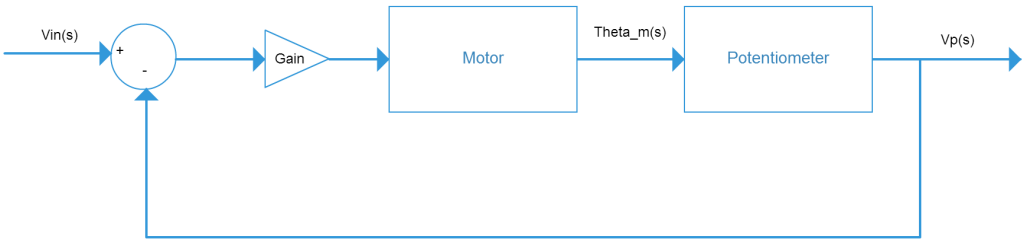


Figure 3: Functional Diagram of the control system

2. The updated functional diagram has been included below.

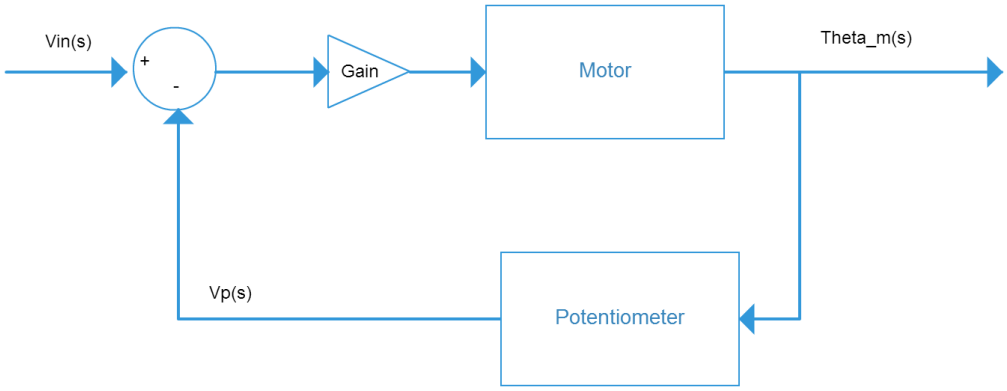


Figure 4: Updated Functional Diagram

3. The output of the requested matlab script has been included below, after which, the code itself has been provided.

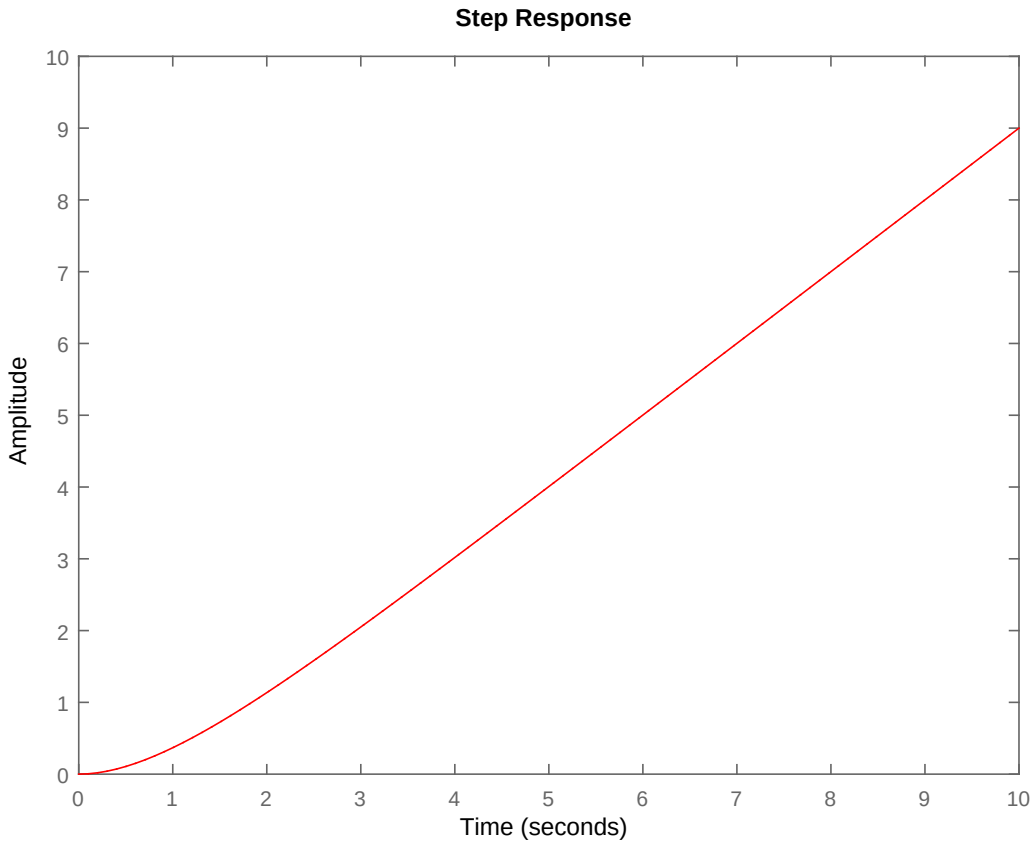


Figure 5: Simulated Response

```
%% A3
Alpha = 1;
K_m = 1;

t = linspace(0,10,100);
G_0 = tf([K_m],[1 Alpha 0]);
figure();
step(G_0, t, 'r')
print('-depsc','a3')
```

4. The given data describing the step response of an ideal servo model has been compared to the previously estimated system response.

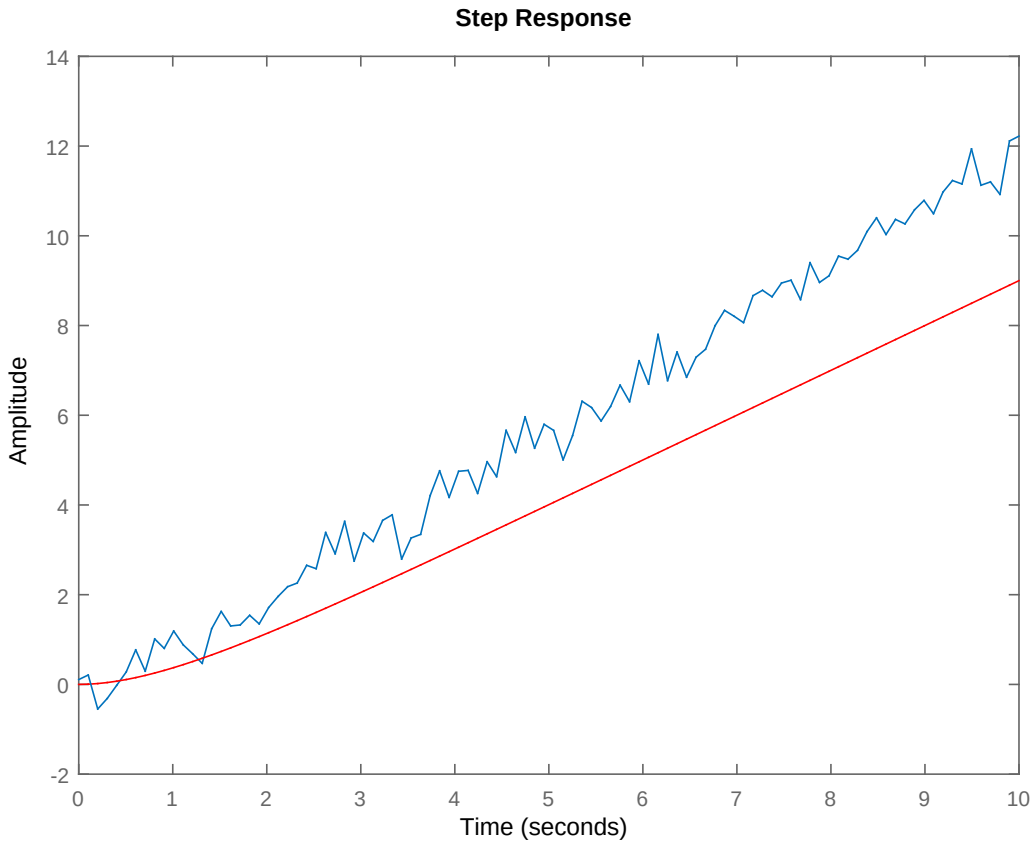


Figure 6: Given Data and Estimated Response

```
%% A4
Alpha = 1;
K_m = 1;
t = linspace(0,10,100);
G_0 = tf([K_m],[1 Alpha 0]);

load ENB301TestData_2015.mat
figure();
plot(t,y1);
hold;
step(G_0, t, 'r')
print('-depsc','a4')
```

5. As can be seen by the supplied transfer function $G_0(s) = \frac{K_m}{s(s+\alpha)}$, the steady state response is predominantly determined by K_m, whereas the transient response is determined by the pole at $s = -\alpha$. This what you would expect from such a system.

6. After changing the values of K_m and α in the previous matlab script, it was found that values of $K_m = 1.3$ and $\alpha = 1$ resulted in an output that matched the test data. Furthermore, a script was created specifically to automatically find the best estimates for values of K_m and α within a given range. The results of both the manual and automatic estimations have been included below.

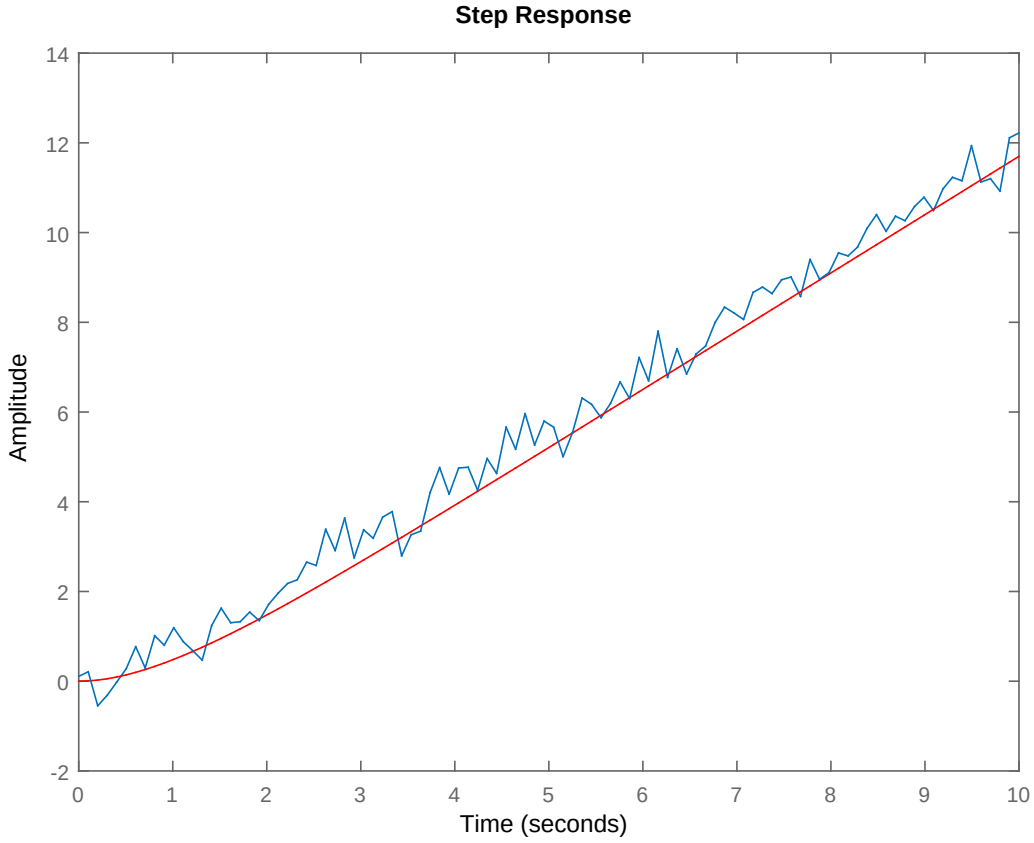


Figure 7: Manually Estimated System Values

```
%% A6
Alpha = 1; %0.5-1.5
K_m = 1.3; %1-2
G_0 = tf([K_m],[1 Alpha 0]);
figure();
plot(t,y1);
hold;
step(G_0, t, 'r');
print('-depsc','a6a')
```

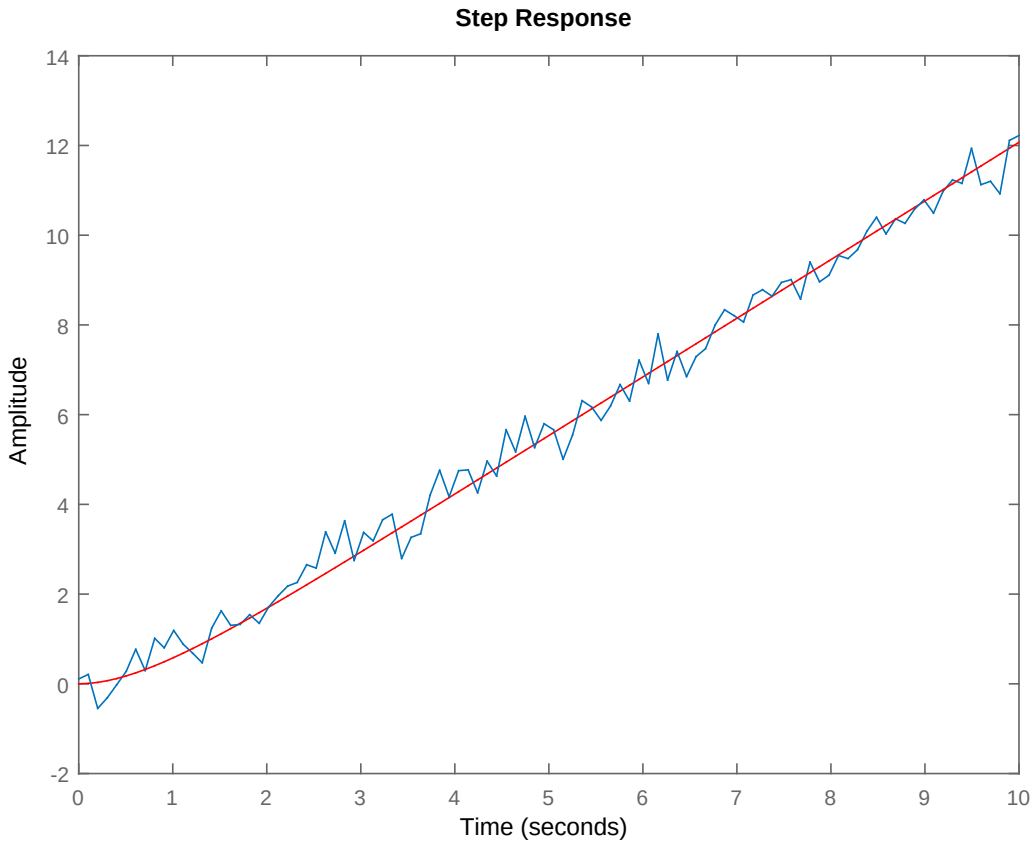


Figure 8: Inverting Amplifier System

```
%% Automated:
% loop values of alpha and k_m
% calculate RMS error each time
% pick values with the least error

y1_rms = rms(y1);
prevdiff = 10000;

for Alpha = 0.1:0.1:3
    for K_m = 0.1:0.1:3
        G_0 = tf([K_m],[1 Alpha 0]);
        G_t = step(G_0, t, 'r');
        Gt_rms = rms(G_t);
        diff = y1_rms - Gt_rms;
        if (abs(diff) < abs(prevdiff))
            prevdiff = diff;
            K_m_f = K_m;
            Alpha_f = Alpha;
        end
    end
end

G_0 = tf([K_m_f],[1 Alpha_f 0]);
figure();
plot(t,y1);
hold;
step(G_0, t, 'r');
print('-depsc','a6b')

% Final values:
% Alpha_f = 1.3
% K_m_f = 1.7
```

7. Using matlabs random number generator, uncertainty was added to the output $y_1(t)$, and both the ideal and noisy response were plotted. If the noise value where changed, it would increase the deviation from the estimated system response proportional to the amount changed. The modeling of noise in this system is not completely accurate; this is because in real world application, noise is not evenly distributed along the domain of the signal, the noise is also not usually distributed in a set range of deviation away from the clean signal. Common sources of noise in these real world applications include electrical noise from power sources and grounds, external disruptions from whether and EMF, and locational noise being induced from the inductance in the circuits wires.

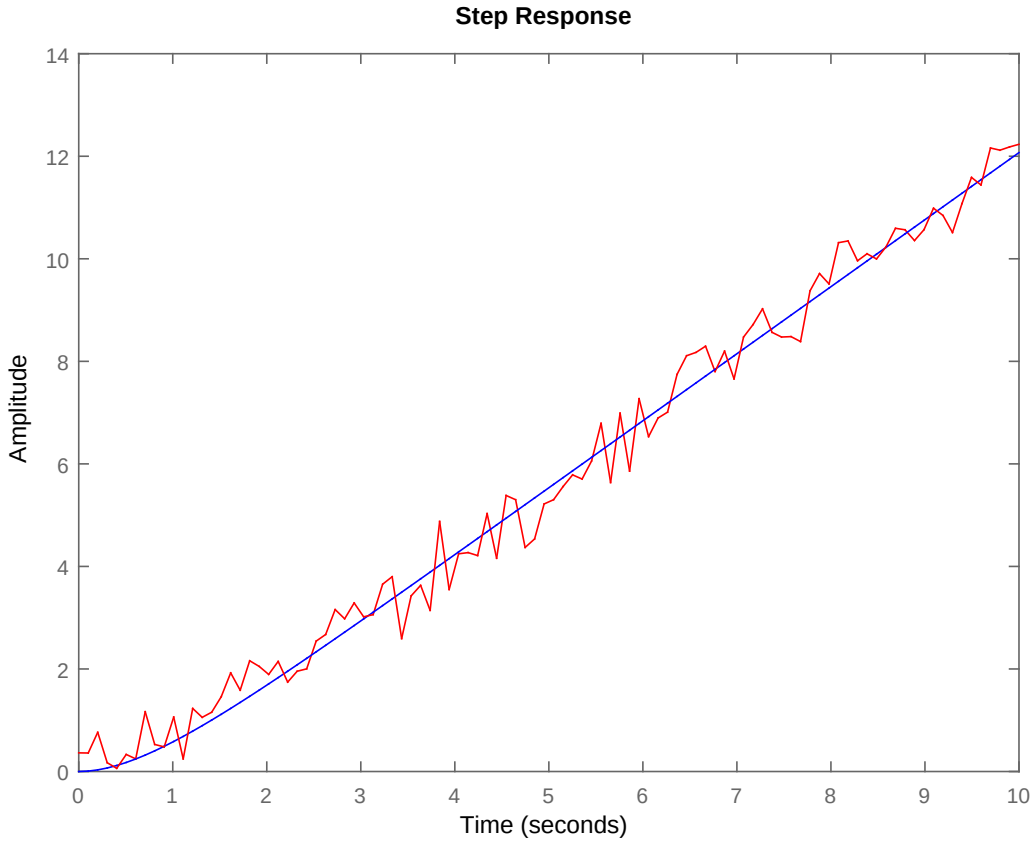


Figure 9: Estimated System Response + Uncertainty

```
%% A7
rand_power = 2;

y_1 = step(G_0, t, 'r');
y_n = y_1 + (rand_power/2)*rand(size(y_1,1),1) - ...
    (rand_power/2)*rand(size(y_1,1),1);

figure();
step(G_0, t, 'b');
hold;
plot(t,y_n,'r');
print('-depsc','a7')
```

PRELAB QUESTIONS:

1. 2
2. True
3. False, True, Increase Constantly, $G_0(s)$
4. data = data(527:end), calc = calc(100:end), then pad end of the shorter vector with zeros (to ensure the same length).
5. Divide 1st response by gain of 1.38, or multiply with 2nd response.
6. Refer to the previous section.

8.1.2 Declan Gilmour

PART A ANSWERS:

1. **Model the complete servo motor control system using a functional diagram. Label all signals and assumptions made including measurement units.**

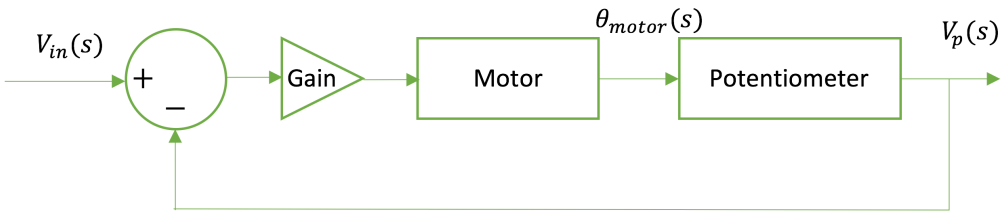


Figure 10: Servo Motor Functional Diagram

A voltage is fed into the motor with a magnitude gain. From here the voltage is converted into rotational movement $\theta_{motor}(s)$ by the motor. This rotational movement alters the displacement of the potentiometer wiper. The potentiometer acts as a sensor to indicate arm displacement in the form of output voltage $V_p(t)$.

2. **Update the servo motor functional diagram and make any necessary changes.**

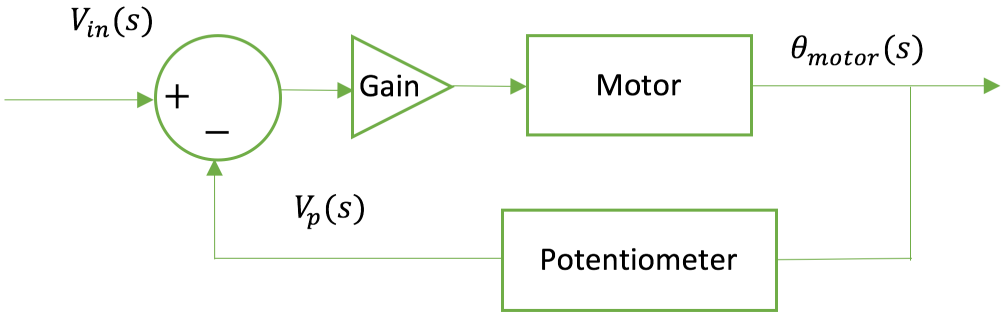


Figure 11: Updated Servo Motor Functional Diagram

To calculate the two servo motor system characteristics, α and K_m the system response would need to be recorded to a step input. This open loop time response is measured in terms of how the voltage applied to the motor affects the voltage seen at the potentiometer.
The selection of optimum values of α and K_m, wold require the potentiometer voltage to be measured with respect of the input voltage. The modification of the Servo Motor Functional Diagram is needed, incorperating a feedback loop as shown.

3. **Create code to simulate motor shaft angle for various values of K_m and α**
   Bellow is the Matlab code used to simulate the motor shaft angle.

```
%% A3 - Simulate motor shaft angle for values of km and alpha
alpha = 1;
km = 1;
t = linspace(0,10,100);

% Build transfer function
G = tf(km, [1 alpha 0]);    % Set G(s) = km / (s + a)
G_0 = step(G,t);    % Set G_0(s) = km / (s * (s + a))

% Plot motor set response
figure
plot(t,G_0,'r')
title('Simulated Step Response')
xlabel('t (sec)')
ylabel('Amplitude')
print('-depsc','A3')
close
```

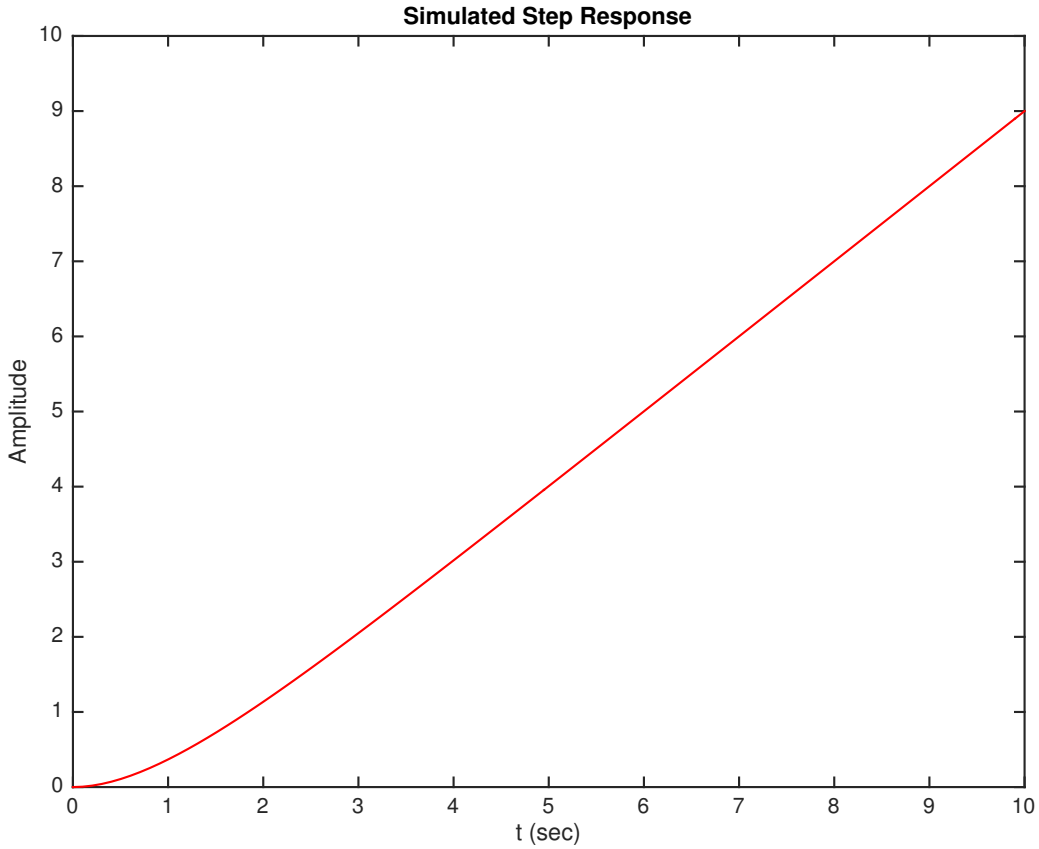


Figure 12: Updated Servo Motor Functional Diagram

4. **Plot $y_1(t)$ against the servo motor response in ENB301_TestData and compare.** Bellow is the Matlab code used to generate the simulated step response against imported test data.

```
%% A4 - Plot motor unit step response data against simulated response
load ENB301TestData_2015.mat
alpha = 1;
km=1;
G = tf(km,[1 alpha 0]);
G_0 = step(G,t);

figure
hold on
plot(t,y1,'b')
plot(t,G_0,'r')
title('Simulated Step Response and Test Data')
xlabel('t (sec)')
ylabel('Amplitude')
legend('y1(t): Simulated Step Response','Test Data')
hold off
print('-depsc','A4')
close
```

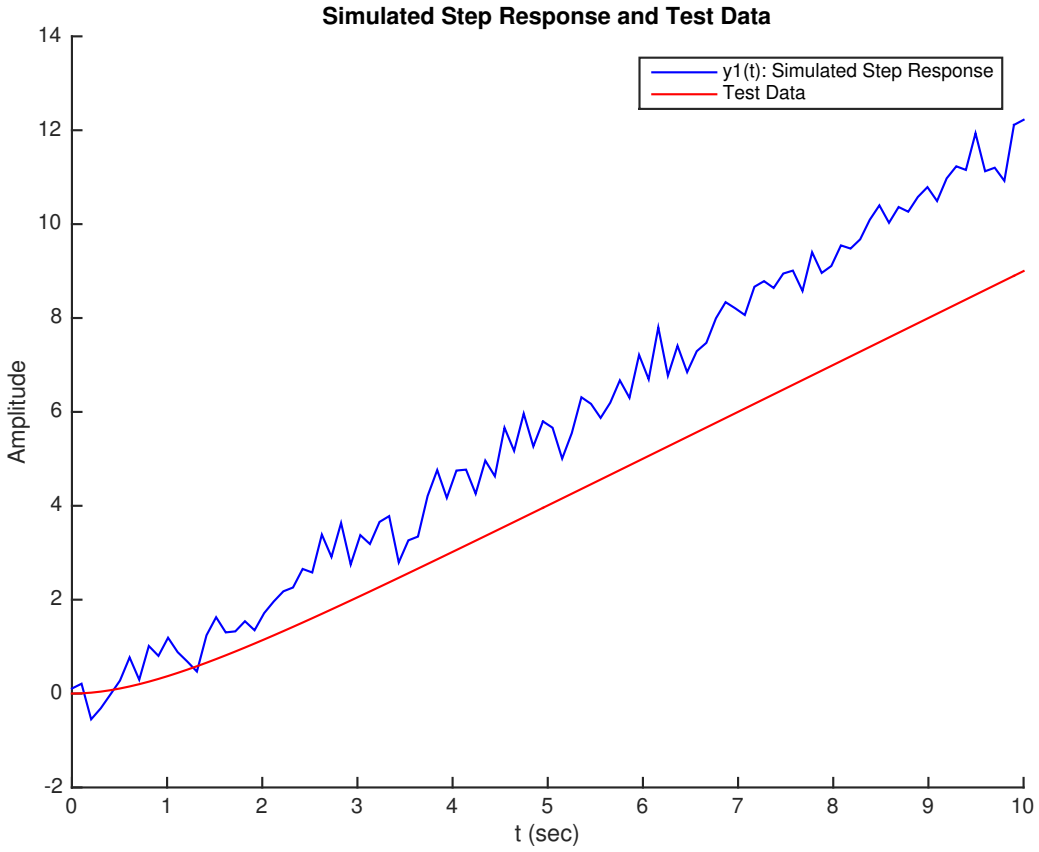


Figure 13: Updated Servo Motor Functional Diagram

5. **What is the steady state response and transient response predominantly determined by?**

6. **Change the values of K_m and α for to match the simulated response with the test data.**
   The estimated K_m value is 2.3838 and α 1.8990. Bellow is the Matlab Code to estimate km and alpha variables.

```
%% A6 - Estimate km and alpha variables
load ENB301TestData_2015
output = zeros(3,100*100);
error = 0;
ii = 1;
for km = linspace(0,4,100)
   for alpha = linspace(0,4,100)
       G = tf(km, [1 alpha 0]);
       G_0 = step(G,t);

       % Calculate mean square error
       for jj = 1 : length(t)
          error = error + (G_0(jj) - y1(jj))^2;
       end

       output(:,ii) = [km;alpha;error];
       ii = ii + 1;
       error = 0;
   end
end

[~,index] = min(output(3,:));
km = output(1,index); % Output variable
alpha = output(2,index);  % Output variable

% Build transfer function
G = tf(km, [1 alpha 0]);    % Set optimal G(s) = km / (s + a)
G_0 = step(G,t);    % Set optimal G_0(s) = km / (s * (s + a))

% Plot step function G_0(s)
figure
hold on
plot(t,G_0,'-b');
plot(t,y1,'-r');
title('Estimated Step Response of Test Data')
xlabel('t (sec)')
ylabel('Amplitude')
legend('Estimated Step Response', 'Test Data');
hold off;
print('-depsc','A6')
close

% Output km and alpha
disp(km)    %2.3838
disp(alpha) %1.8990
```

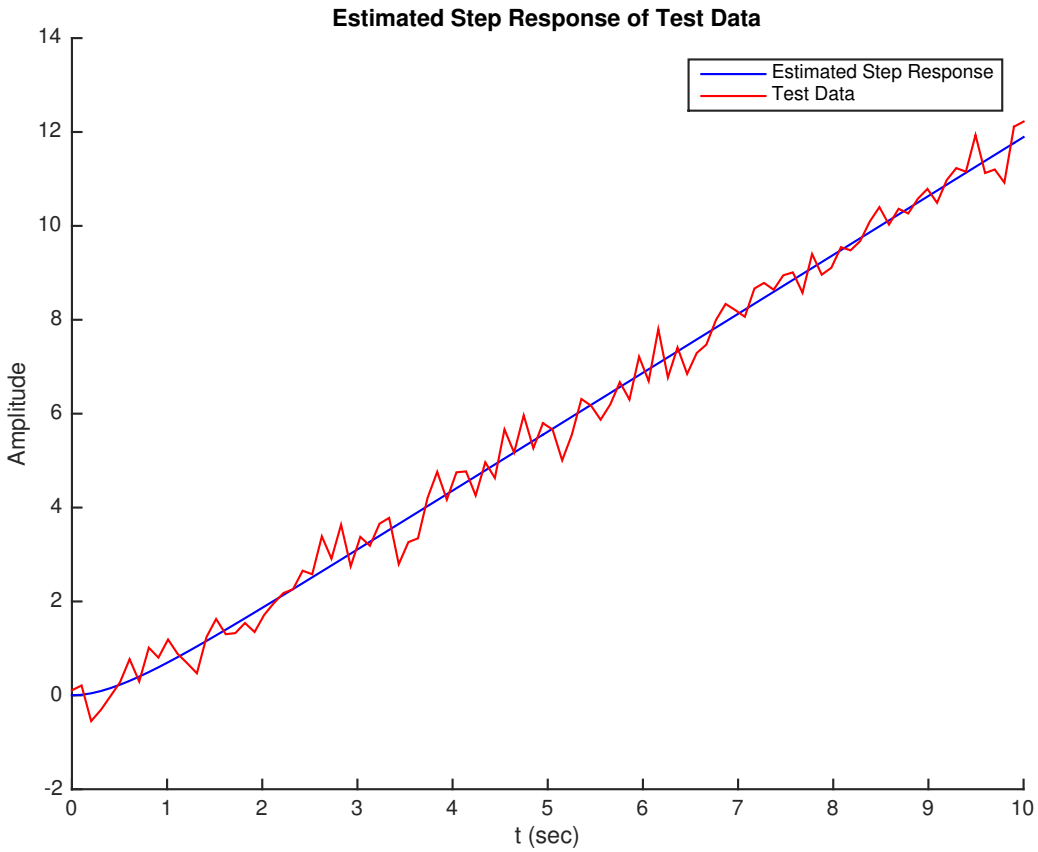


Figure 14: Estimated K_m and α values

7. **Plot the step response from original signal and a step response from a noisy signal.** Bellow is the Matlab Code to plot a step response from both original and noisy signals. $K_m = 2.4242$ and $\alpha = 1.9394$.

```
%load ENB301TestData_2015
% Generate noise
% Standard: sigma = 5, Decreased: signma = 0.2, Increased: sigma = 20
sigma = 5; %noise standard deviation
noise = sigma*randn(size(G_0)); %noise vector

% Create noisy signal
yn = y1 + noise;

output = zeros(3,100*100);
error = 0;
ii = 1;
for km = linspace(0,4,100)
   for alpha = linspace(0,4,100)
       G = tf(km, [1 alpha 0]);
       G_0 = step(G,t);

       % Calculate mean square error
       for jj = 1 : length(t)
          error = error + (G_0(jj) - yn(jj))^2;
       end

       output(:,ii) = [km;alpha;error];
       ii = ii + 1;
       error = 0;
   end
end

[~,index] = min(output(3,:));
km = output(1,index); % Output variable
alpha = output(2,index);  % Output variable

% Build transfer function
G_noisy = tf(km, [1 alpha 0]);    % Set optimal G(s) = km / (s + a)
G_0_noisy = step(G_noisy,t);    % Set optimal G_0(s) = km / (s * (s + a))

% Output km and alpha
disp(km)    %2.4242
disp(alpha) %1.9394

% Plot noisy signal vs noisy test data
figure(2)
hold on
plot(t,G_0_noisy, '-r');
plot(t,G_0,'-b');
title('Estimated Step Response of Noisy Test Data')
xlabel('t (sec)')
ylabel('Amplitude')
legend('Noisy Step Response', 'Original Step Response');
hold off
print('-depsc','A7')
close
```

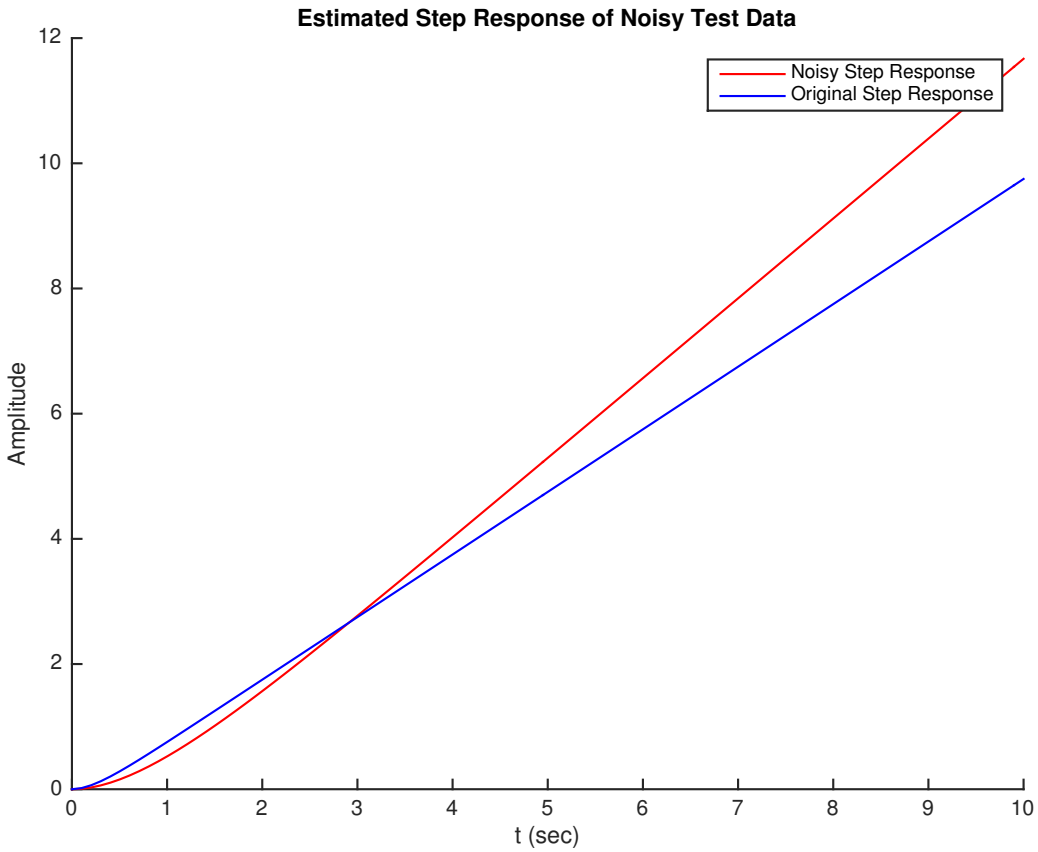
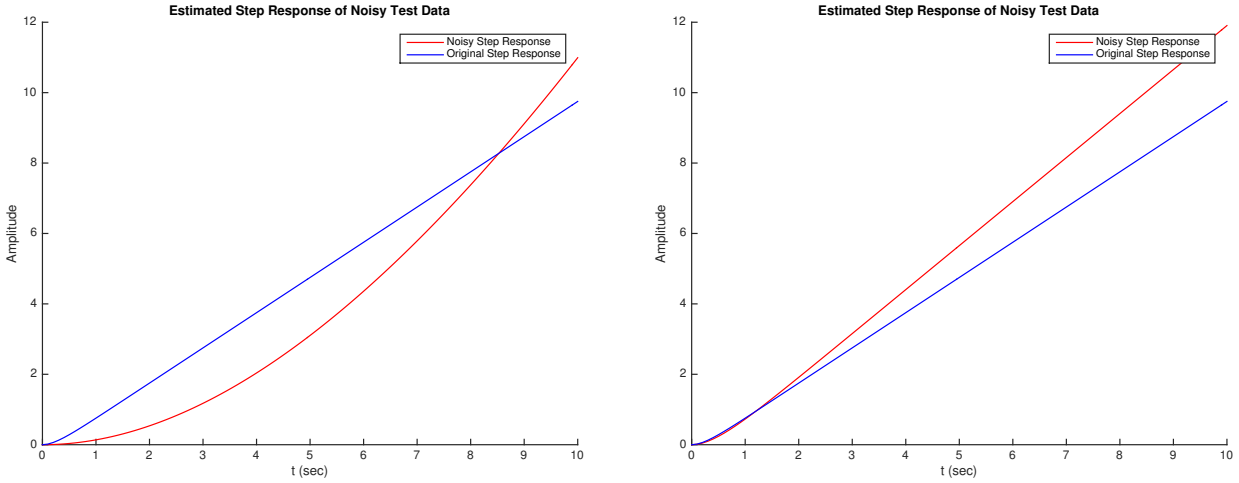


Figure 15: Original Step Response and Noisy Step Response

For this noisy step response, $K_m = 1.4949$ and $\alpha = 1.1717$. When the noise noise decreases, K_m and α decreases in magnitude. When the noise increases, K_m and α increases in magnitude. The transient response of the system appears to increase as the noise of the system increases.

Is this Realistic?

Whare are the source of uncertanty or noise in a real system?
Sources of uncertainty or noise in a system can include: control error, electrical noise, external factors such as weather and human error.



(a) Increased Noise: $K_m = 0.2828$ and $\alpha = 0.0808$ (b) Decreased Noise: $K_m = 2.6263$ and $\alpha = 2.1010$

Figure 16: Increased and Decreased noisey Step Response

PRELAB QUESTIONS:

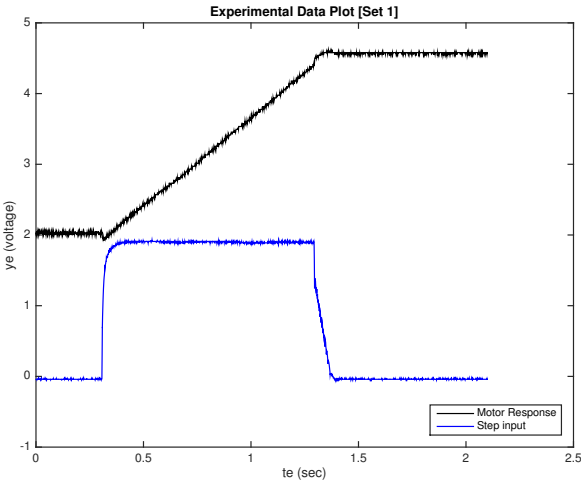
1. 2
2. True
3. False, True, Increase Constantly, $G_0(s)$
4. experimentalFixed = experimental(527:end);
   matlabFixed = matlab(100:end);
5. Divide first response by 1.38 or multiply second response by 1.38.

8.2 Answers:

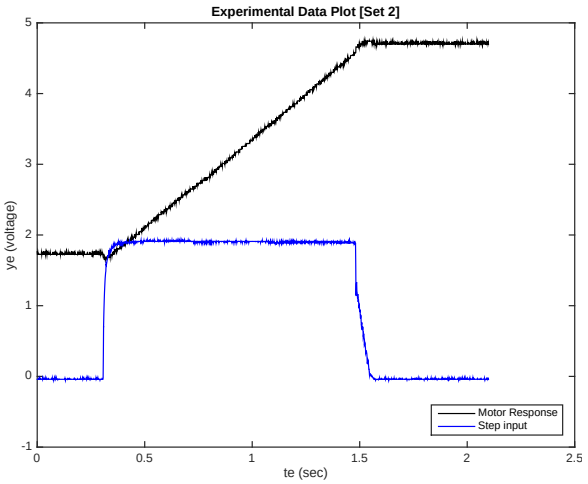
8.2.1 Part B

1. Read experimental data file(s) and store in vectors $y_e(t)$ and t_e. Plot the experimental results in black.

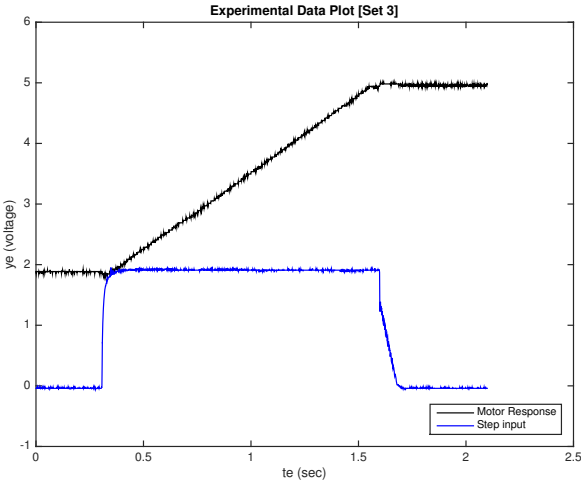
```
%% B1
% Load experimental data and plot in black
data = csvread('PartB_Test1.csv',2,0);  % Read in test 1
te_1 = data(1:end,1);   % Store te variable
te_1 = te_1 + abs(te_1(1)); % Move te variable to start at zero
ye_1 = data(1:end,2);   % Store ye variable
ye_1step = data(1:end,3);   % Store ye step input variable

figure
plot(te_1,ye_1,'k',te_1,ye_1step,'b')   % Plot ye in black and ye in blue
title('Experimental Data Plot [Set 1]')
xlabel('te (sec)')
ylabel('ye (voltage)')
print('-depsc',strcat('figures',filesep,'B1_dataset1'));    % Store figure
close
```



(a) Data Set 1



(b) Data Set 2

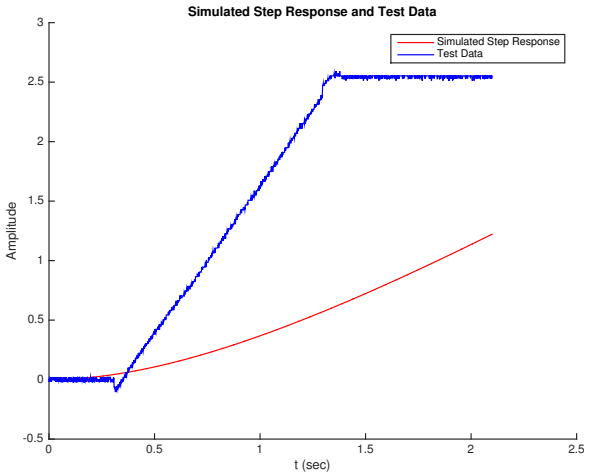


(c) Data Set 3

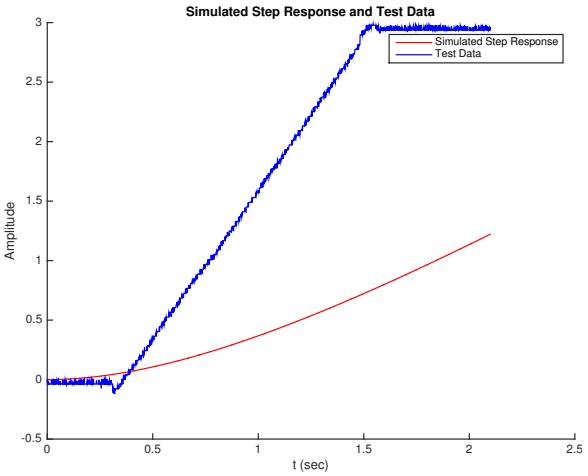
Figure 17: Open Loop response

2. Modify the MATLAB script created in the pre-labs to simultaneously plot both the experimental data and $y_1(t)$.

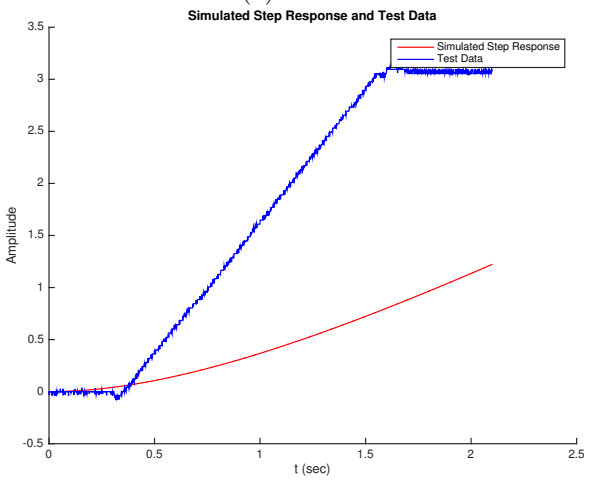
```
%% B2
alpha = 1;
km=1;
G = tf(km,[1 alpha 0]);

G_0 = step(G,te_1);
figure
hold on
plot(te_1,G_0,'r')
plot(te_1,ye_1-ye_1(1),'b')
title('Simulated Step Response and Test Data')
xlabel('t (sec)')
ylabel('Amplitude')
legend('Simulated Step Response','Test Data')
hold off
print('-depsc',strcat('figures',filesep,'B2_dataset1'));
close
```



(a) Data Set 1



(b) Data Set 2

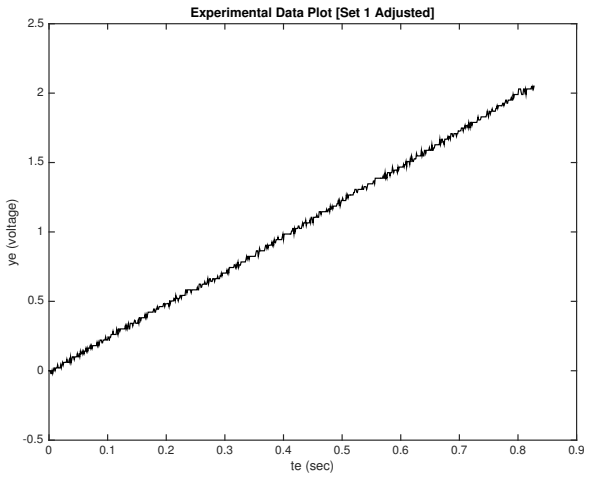


(c) Data Set 3

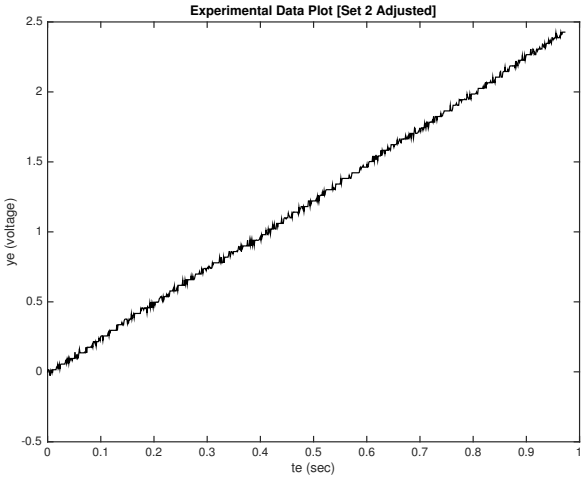
Figure 18: Open Loop response

3. Derive a figure of merit for the estimation compared with experimental results.
   Mean Square error calculation: $error = (G_0 - y_e)^2$
   Root Mean Square error calculation: $error = \sqrt[2]{(G_0 - y_e))^2}$

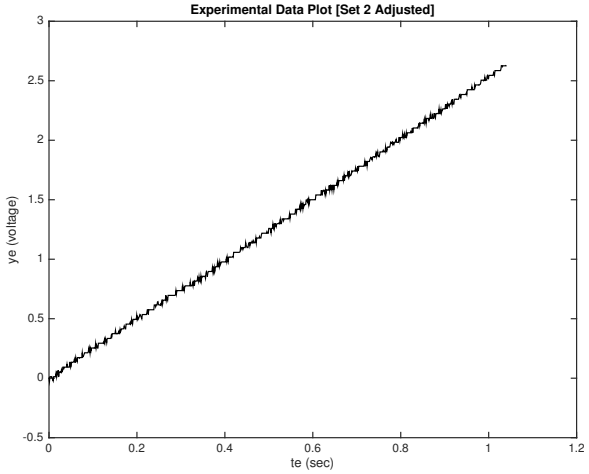
4. Improve estimates using the plots and figure of merit calculations. Derive the two parameters for the servo motor function $G(s) = \frac{K_m}{(S+\alpha)(S+\beta)}$.



(a) Data Set 1



(b) Data Set 2



(c) Data Set 3

Figure 19: Servo Motor Response Modified: k_m and α

```
% Plot experimental data ye against systems estimated tf y1
[te_1new,ye_1new] = timing_fix(te_1,ye_1);
[te_2new,ye_2new] = timing_fix(te_2,ye_2);
[te_3new, ye_3new] = timing_fix(te_3,ye_3);

figure
plot(te_1new,ye_1new,'k')
title('Experimental Data Plot [Set 1 Adjusted]')
xlabel('te (sec)')
ylabel('ye (voltage)')
print('-depsc',strcat('figures',filesep,'B2_dataset1'));
close

figure
plot(te_1new,ye_1new,'k')
title('Experimental Data Plot [Set 1 Adjusted]')
xlabel('te (sec)')
ylabel('ye (voltage)')
print('-depsc',strcat('figures',filesep,'B2_dataset1'));
close
```

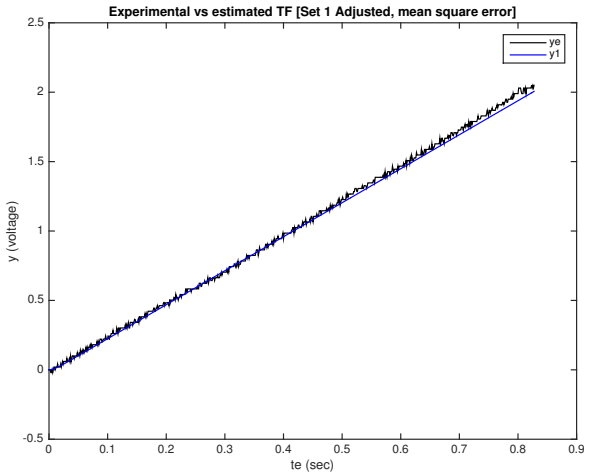
```
% Pepare experimental data for alpha and km parameter calculations
[te_1trim, ye_1trim] = trimForCalculation(te_1new,ye_1new,mf1);
[te_2trim, ye_2trim] = trimForCalculation(te_2new,ye_2new,mf1);
[te_3trim, ye_3trim] = trimForCalculation(te_3new,ye_3new,mf1);

km_num = 500;    % number of km values used
km_max = 500;   % max km value used
alpha_num = 500;    % number of alpha values used
alpha_max = 500;    % max alpha value used
```

```
% Preallocate size for speed
output_ms = zeros(3,km_num*alpha_num);
output_rms = zeros(3,km_num*alpha_num);
km_ms = zeros(1,3);
alpha_ms = zeros(1,3);
km_rms = zeros(1,3);
alpha_rms = zeros(1,3);

for iteration = 1 : 3

    % Set te and ye based on iteration
    if (iteration == 1)
        te = te_1trim;
        ye = ye_1trim;
    elseif (iteration == 2)
        te = te_2trim;
        ye = ye_2trim;
    else
        te = te_3trim;
        ye = ye_3trim;
    end

    % Set cycle variables
    error_ms = 0;
    error_rms = 0;
    ii = 1;
    count = 0;

    for km = linspace(0,km_max,km_num)   % Cycle km values
       for alpha = linspace(0,alpha_max,alpha_num)  % Cycle alpha values
           G = tf(km, [1 alpha 0]);
           G_0 = step(G,te);

           % Calculate error
           for jj = 1 : length(te)
              % Calculate mean square error
              error_ms = error_ms + (G_0(jj) - ye(jj))^2;

              % Calculate root mean square error
              error_rms = error_rms + rms(G_0(jj) - ye(jj));
           end

           % Store km, alpha and the error taken to calculate
           output_ms(:,ii) = [km;alpha;error_ms];
           output_rms(:,ii) = [km;alpha;error_rms];

           % Reset cycle variables
           ii = ii + 1;
           error_ms = 0;
           error_rms = 0;
       end

       % Output km iterations
       %count = count + 1;
       %fprintf('%d %s %d\n',count,'/',km_num);
```

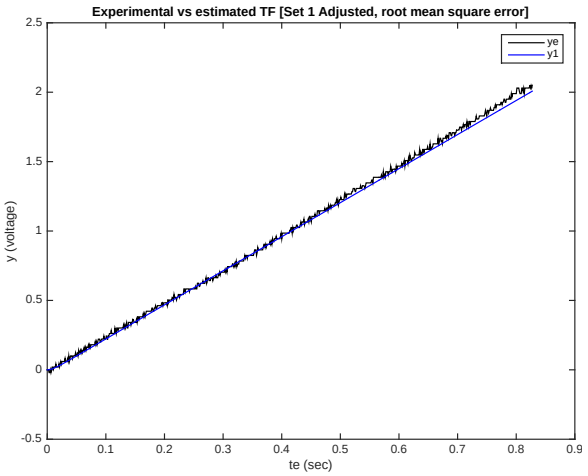
```
    end

    % Calculate km and alpha values for mean square error calculation
    [~,index] = min(output_ms(3,:));
    km_ms(iteration) = output_ms(1,index); % Output variable
    alpha_ms(iteration) = output_ms(2,index);  % Output variable

    % Calculate km and alpha values for root mean square error calculation
    [~,index] = min(output_rms(3,:));
    km_rms(iteration) = output_rms(1,index); % Output variable
    alpha_rms(iteration) = output_rms(2,index);  % Output variable
end
km_mean = (mean(km_ms) + mean(km_rms)) / 2;
alpha_mean = (mean(alpha_ms) + mean(alpha_rms)) / 2;
```

```
% Plot y1 against ye
% Data Set 1
G = tf(km_ms(1), [1 alpha_ms(1) 0]);
y1_ms = step(G,te_1new);
figure
plot(te_1new,medfilt1(ye_1new,1),'k',te_1new,medfilt1(y1_ms,1),'b')
title('Experimental vs estimated TF [Set 1 Adjusted, mean square error]')
xlabel('te (sec)')
ylabel('y (voltage)')
legend('ye','y1')
print('-depsc',strcat('figures',filesep,'y1_dataset1_ms'));
close
```

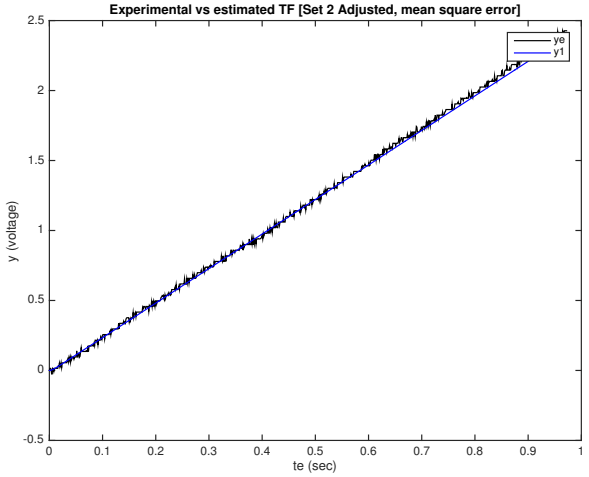
The average constants found are as follows: $\alpha = 170.3407$ $k_m = 422.0107$



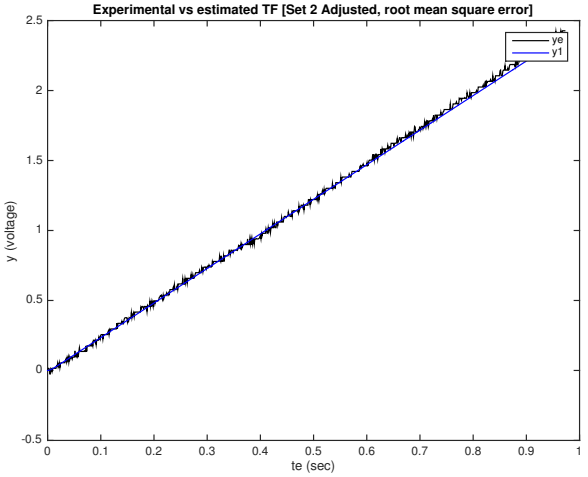
(a) Data Set 1 (ms)



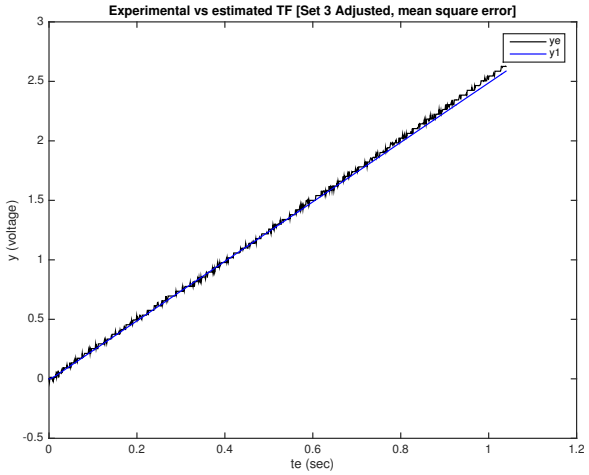
(b) Data Set 1 (rms)



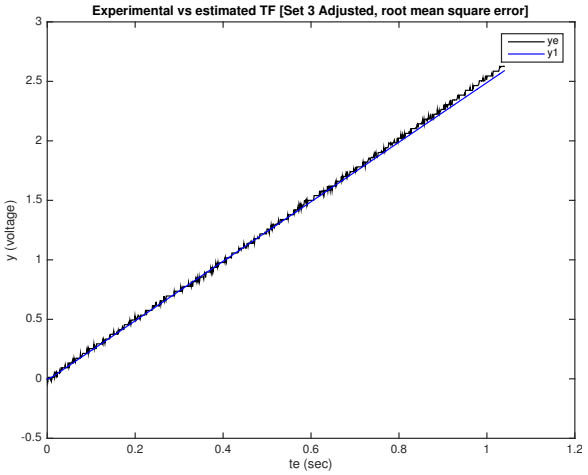
(c) Data Set 2 (ms)



(d) Data Set 2 (rms)



(e) Data Set 3 (ms)



(f) Data Set 3 (rms)

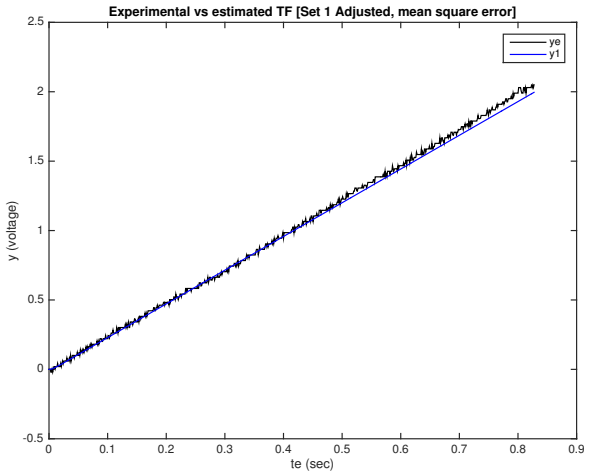
Figure 20: Servo Motor Response Modeled

5. $K_m = 426.6867$ and $\beta = 0.0646$. When considering the general transfer function of input voltage $G(s) = \frac{K_m}{(s+\alpha)(s+\beta)}$, a very small β value would turn this into the simplified input voltage to motor shaft transfeer function $G_o(s) = \frac{V_p(s)}{V_m(s)} = \frac{K_m}{s(s+\alpha)}$.

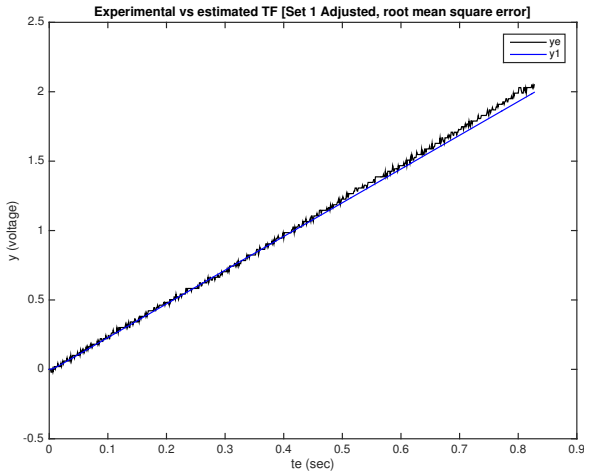
   Therefore, the β value of 0.0646 suggests the original assumption of the simplified input voltage to motor shaft transfer function was correct. The useful approximation if $|\beta| < 10|\alpha|$ the transient response will be dominated by α satisfies this system.
   Values were found for $y_1(t) \approx y_2(t)$ as shown in Fig 20 and Fig 21 respectively.

From these observations, it is evident the electrical constant β is negligable in this system.

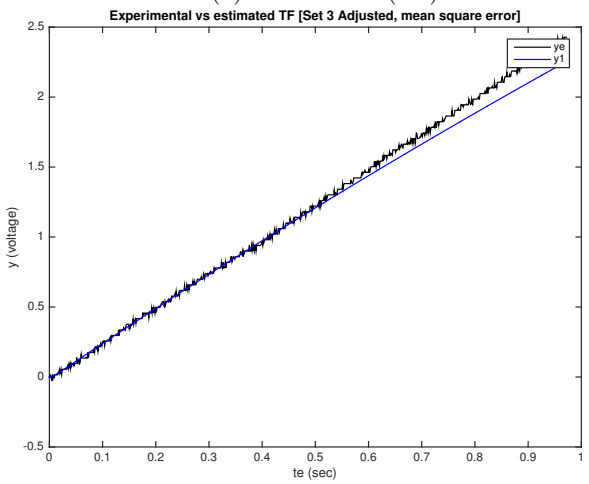
The mechanical constant α was found to be $\alpha = alpha_{mean}$. Using the second order equation $G(s) = \frac{K_m}{(S+\alpha)}$ new values of gain constant k_m and electrical constant β. Values were found for $y_1(t) \approx y_2(t)$. The average constants found are as follows: $\beta = 0.0646$ $k_m = 426.6867$



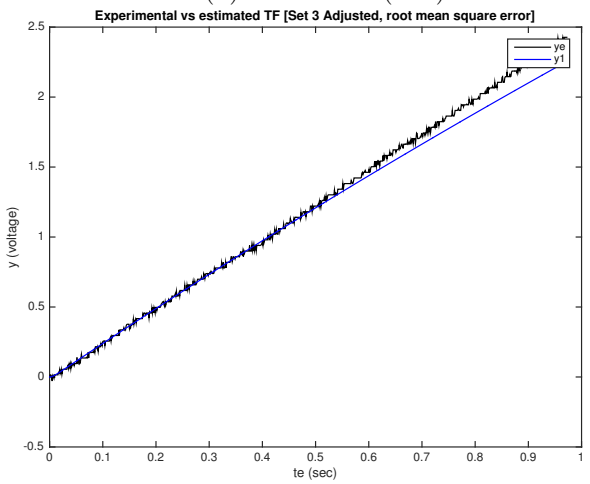
(a) Data Set 1 (ms)



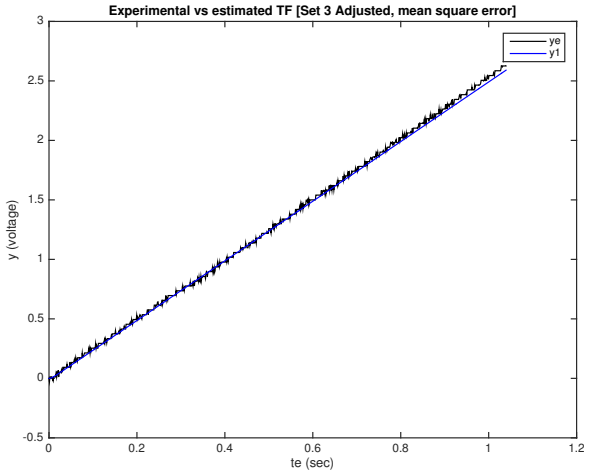
(b) Data Set 1 (rms)



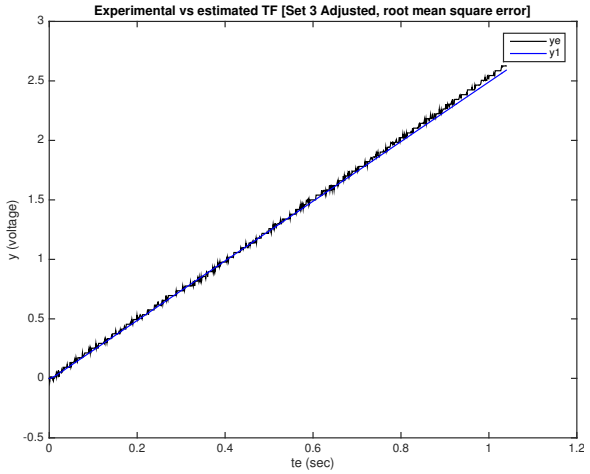
(c) Data Set 2 (ms)



(d) Data Set 2 (rms)



(e) Data Set 3 (ms)



(f) Data Set 3 (rms)

Figure 21: Servo Motor Response Modeled: k_m and β

```
km_num = 500;     % number of km values used
km_max = 500;    % max km value used
beta_num = 50;     % number of alpha values used
beta_max = 1;     % max alpha value used

% Preallocate size for speed
output_ms = zeros(3,km_num*beta_num);
output_rms = zeros(3,km_num*beta_num);
km2_ms = zeros(1,3);
beta_ms = zeros(1,3);
km2_rms = zeros(1,3);
beta_rms = zeros(1,3);

for iteration = 1:3
    if(iteration == 1)
        te = te_1trim;
        ye = ye_1trim;
    elseif(iteration==2)
        te = te_2trim;
        ye = ye_2trim;
    else
        te = te_3trim;
        ye = ye_3trim;
    end

    % Set cycle variables
    error_ms = 0;
    error_rms = 0;
    ii = 1;
    count = 0;

    for km = linspace(0,km_max,km_num)    % Cycle km values
       for beta = linspace(0,beta_max,beta_num)  % Cycle alpha values
           G = tf(km, [1 (alpha_mean+beta) (alpha_mean*beta)]);
           G_0 = step(G,te);

           % Calculate error
           for jj = 1 : length(te)
              % Calculate mean square error
              error_ms = error_ms + (G_0(jj) - ye(jj))^2;

              % Calculate root mean square error
              error_rms = error_rms + rms(G_0(jj) - ye(jj));
           end

           % Store km, alpha and the error taken to calculate
           output_ms(:,ii) = [km;beta;error_ms];
           output_rms(:,ii) = [km;beta;error_rms];

           % Reset cycle variables
           ii = ii + 1;
           error_ms = 0;
           error_rms = 0;
       end

        % Output km iterations
       %count = count + 1;
       %fprintf('%d %s %d\n',count,'/',km_num);
    end

    % Calculate km and alpha values for mean square error calculation
    [~,index] = min(output_ms(3,:));
    km2_ms(iteration) = output_ms(1,index); % Output variable
    beta_ms(iteration) = output_ms(2,index);  % Output variable

    % Calculate km and alpha values for root mean square error calculation
```

```
    [~,index] = min(output_rms(3,:));
    km2_rms(iteration) = output_rms(1,index); % Output variable
    beta_rms(iteration) = output_rms(2,index);  % Output variable
end

km_mean2 = (mean(km_ms) + mean(km2_rms)) / 2;
beta_mean = (mean(beta_ms) + mean(beta_rms)) / 2;
```

```
% Plot y2 against ye
% Data Set 1
G = tf(km2_ms(1), [1 (alpha_mean+beta_ms(1)) (alpha_mean*beta_ms(1))]);
y2_ms = step(G,te_1new);
figure
plot(te_1new,medfilt1(ye_1new,1),'k',te_1new,medfilt1(y2_ms,1),'b')
title('Experimental vs estimated TF [Set 1 Adjusted, mean square error]')
xlabel('te (sec)')
ylabel('y (voltage)')
legend('ye','y1')
print('-depsc',strcat('figures',filesep,'y2_dataset1_ms'));
close
```

8.2.2 Part C

1. **Re-draw figure 22 and place boxes around the set of components that correspond to each functional element of the control system.**

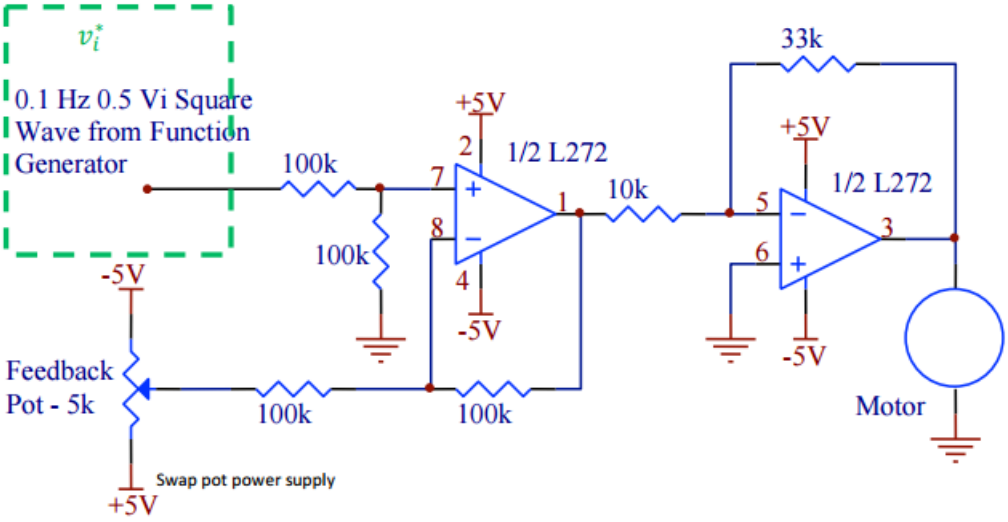


Figure 22: Closed Loop Motor Control Schematic

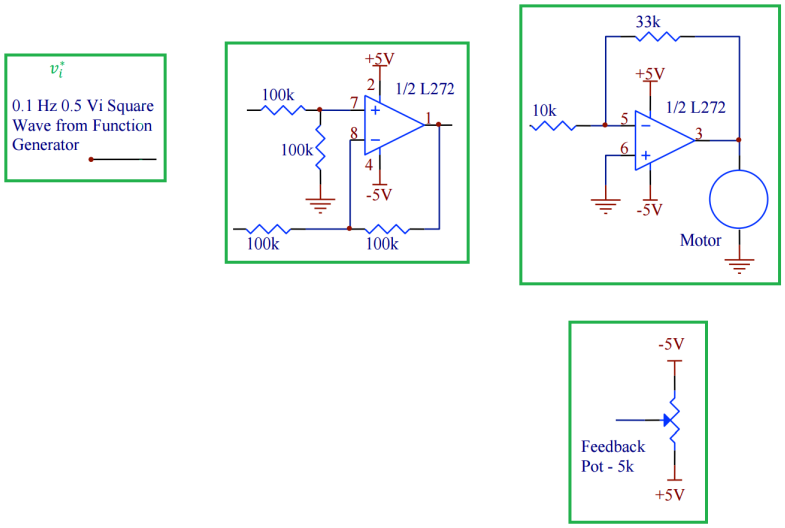
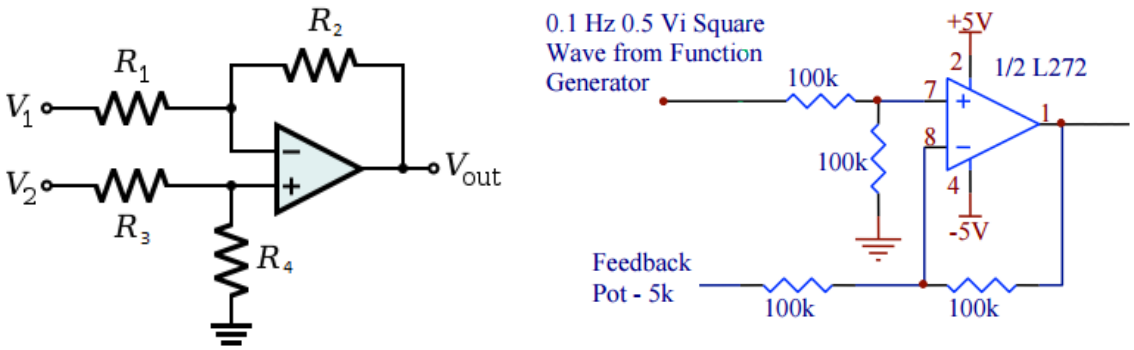


Figure 23: Closed Loop Motor Control System

2. **Based on the results from Parts A and B, the component values given in figure 22 and your research in parts C1, calculate all of the transfer functions in your functional diagram (figure 23). Update the functional diagram, labeling all components and interfaces.**

Difference Op-amp:



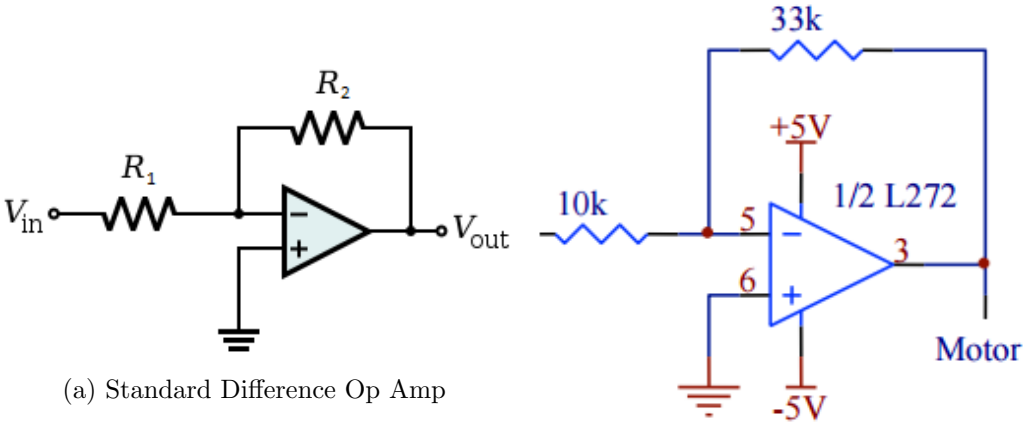
(a) Standard Difference Op Amp

(b) System Difference Op Amp

Figure 24: Difference Op-amp System

$$\frac{V_o(s)}{V_{in}(s)} = \frac{1+\frac{R_2}{R_1}}{1+\frac{R_3}{R_4}}V_2 - \frac{R_2}{R_1}V_1$$
$$\frac{V_o(s)}{V_{in}(s)} = V_o(s) - V_p(s)$$

Inverting Op-amp:



(a) Standard Difference Op Amp

(b) Inverting Op-amp System

Figure 25: Inverting Op-amp System

$$\frac{V_i(s)}{V_d(s)} = \frac{R_2}{R_1}$$

Note: The polarity on the inverter transfer function has been flipped.

Motor:

$$\frac{V_p(s)}{V_i(s)} = \frac{K_m}{s(s+\alpha)}$$

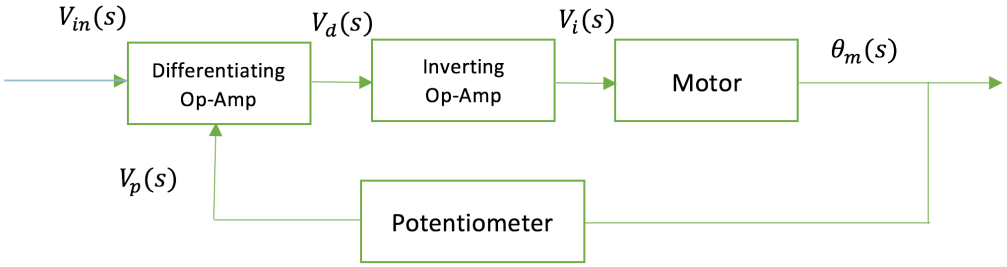


Figure 26: Updated Functional Diagram

3. Calculate the complete system transfer function $G_c(s)$.

System Transfer Function:

$$\begin{aligned} G_c(s) &= \frac{V_p(s)}{V_{in}(s)} \\ &= \frac{V_p(s)}{V_i(s)} \times \frac{V_i(s)}{V_d(s)} \times \frac{V_d(s)}{V_{in}(s))} \\ &= \frac{K_m}{s(s+\alpha)} \times \frac{R_2}{R_1} \times (V_i(s) - V_p(s)) \\ &= \frac{-K_m R_2}{R_1}\left(\frac{V_i(s) - V_p(s)}{s(s+\alpha)}\right) \end{aligned}$$

From Inverting OP amp: $V_m(s) = \frac{R_2}{R_2} \times V_o(s)$

From Motor: $V_m(s) = \frac{s(s+\alpha)V_p(s)}{K_m}$

$$\begin{aligned} \therefore \frac{s(s+\alpha)V_p(s)}{K_m} &= \frac{R_2}{R_2} \times V_o(s) \\ V_o(s) &= \frac{R_1 s(s+\alpha)}{K_m R_2} V_p(s) \end{aligned}$$

From Difference Op Amp: $V_o(s) = V_i(s) - V_p(s)$

$$\begin{aligned} \therefore \frac{R_1 s(s+\alpha)}{K_m R_2} V_p(s) &= V_i(s) - V_p(s) \\ V_p(s)\left[1 - \frac{R_1 s(s+\alpha)}{K_m R_2}\right] &= V_i(s) \\ \frac{V_p(s)}{V_i(s)} &= \frac{1}{1 + \frac{R_1 s(s+\alpha)}{K_m R_2}} \\ &= \frac{1}{1 + \frac{R_1}{K_m R_2}(s^2 + \alpha s)} \\ &= \frac{1}{(\frac{R_1}{K_m R_2})s^2 (\frac{\alpha R_1}{K_m R_2})s + 1} \\ &= \frac{\frac{K_m R_2}{R_1}}{s^2 + (\frac{\alpha R_1 K_m R_2}{K_m R_2 R_1})s + \frac{K_m R_2}{R_1}} \\ &= \frac{\frac{K_m R_2}{R_1}}{s^2 + \alpha s + \frac{K_m R_2}{R_1}} \end{aligned}$$

Ideal System Transfer Function:

$$G_c(s) = \frac{KG(S)}{1 + KG(s)H(s)}$$

$$G(s) = \frac{K_m}{s(s+\alpha)}$$
$$H(s) = 1$$
$$K = \frac{R_f}{R_1}$$

$$G_c(s) = \frac{\frac{R_f}{R_1}G(s)}{1 + \frac{R_f}{R_1}G(s)}$$
$$G_c(s) = \frac{\frac{R_f}{R_1}\frac{K_m}{s(s+\alpha)}}{1 + \frac{R_f}{R_1}\frac{K_m}{s(s+\alpha)}}$$
$$G_c(s) = \frac{\frac{\frac{R_f}{R_1}K_m}{s(s+\alpha)}}{1 + \frac{\frac{R_f}{R_1}K_m}{s(s+\alpha)}}$$
$$G_c(s) = \frac{\frac{R_f}{R_1}K_m}{s(s+\alpha) + \frac{R_f}{R_1}K_m}$$
$$G_c(s) = \frac{\frac{R_f}{R_1}K_m}{s^2 + s\alpha + \frac{R_f}{R_1}K_m}$$

As can be seen, both the of the above methods produce the same result, the first requiring more complex algebra and all of the calculated transfer functions; the second method understands that the summing amplifier and inverting amplifier can be modeled as a summing node and gain block respectively (refer to figure 4).

From previously, $K_m = 326$, $\alpha = 38.61$, $R_f = 33k$ and $R_1 = 10k$. Thus,

$$G_c(s) = \frac{\frac{33000}{10000}K_m}{s^2 + s\alpha + \frac{33000}{10000}K_m}$$
$$G_c(s) = \frac{3.3 * K_m}{s^2 + s\alpha + 3.3 * K_m}$$
$$G_c(s) = \frac{3.3 * 326}{s^2 + 38.61s + 3.3 * 326}$$
$$G_c(s) = \frac{1075.8}{s^2 + 38.61s + 1075.8}$$

4. $G_c(s) = \frac{1075.8}{s^2+38.61s+1075.8}$ **has been used to produce the expected time responce $y_c(t)$ of this system. The calculated characteristics are as follows:**
   Damping ratio (zeta): 0.690107
   Natural frequency (Wn): 27.973934
   Peak time (Tp): 0.155179
   %OS (Mp): 0.050000
   Settling Time (Ts): 0.207200

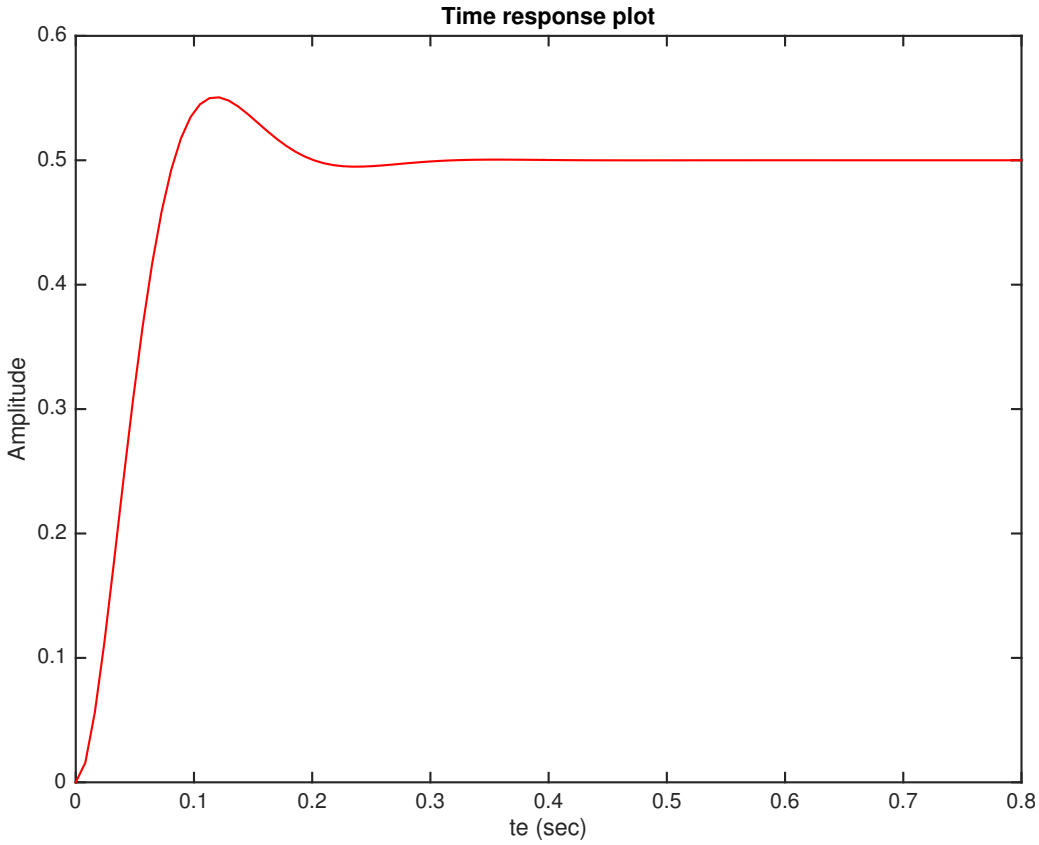


Figure 27: Time response of system.

```
1
2  %% C4
3  %Calculate expected time response
4  Rf = 33 * 10^3;
5  R1 = 10 * 10^3;
6  alpha = 38.61;
7  Km = 326;
8  K = Rf / R1;
9
10 t = linspace(0,0.8,100);
11 G_c = tf(K * Km, [1 alpha K * Km]);
12 Y_c = 0.5 * step(G_c,t);
13
14 figure
15 plot(t,Y_c,'k')
16 title('Experimental vs estimated TF [Set 3 Adjusted, root mean square error]')
17 xlabel('te (sec)')
18 ylabel('Amplitude')
19 print('-depsc',strcat('figures',filesep,'C4'));
20 close
21
22 %Y_c = tf(K * Km, [1 alpha K * Km 0]);
23 num = 0.5 * K * Km;
24 den = [1 alpha K*Km 0];
25 % [num, den] = tfdata(Y_c,'v');
26 %r = gain, p = pole, k = direct term ie r / (s + p) + k
27 [r, p, k] = residue(num,den);
28 %
29 syms s;
30 partialFrac = [r(1)/(s-p(1)) r(2)/(s-p(2)) r(3)/(s-p(3))];
31 time_response = ilaplace(partialFrac);  % Inverse laplace from s domain to t ...
       domain
32 time_response = vpa(time_response,5); % Round to 5 sig figs
33
34 zeta = -log(5/100) / sqrt(pi^2 +log(5/100)^2);  % Damping ratio
35 Wn = alpha / (2 * zeta);    % Natural frequency
36 Wd = Wn * sqrt(1 - zeta^2);
37 Tp = pi / Wd;   % Peak time
38 Mp = exp((-pi  * zeta) / sqrt(1 - zeta^2)); % Percentage overshoot
39 Ts =  4 / (zeta * Wn);   % Settling time
```

5. **Calculate the gain required in the final stage to produce a 5% overshoot. Choose resistor values to match the required gain.**

Recall, for 5% overshoot, $\zeta = \frac{-\ln(5/100)}{\sqrt{\pi^2+\ln^2(5/100)}} = 0.69$

And, the systems estimated overall transfer function; $\frac{k_m \frac{R_f}{R_1}}{s^2+s\alpha+k_m \frac{R_f}{R_1}}$

Whereas, the general second order transfer function; $\frac{W_n^2}{s^2+2\zeta W_n s+W_n^2}$

Moreover,

$$\alpha = 2\zeta W_n$$
$$W_n = \frac{\alpha}{2\zeta}$$

$$W_n^2 = k_m \frac{R_f}{R_1}$$
$$\frac{R_f}{R_1} = W_n^2/k_m$$
$$K = (\frac{\alpha}{2\zeta})^2/k_m$$

From section B, we know $k_m = 326$ and $\alpha = 38.61$. We also calculated the required ζ previously, as 0.69.

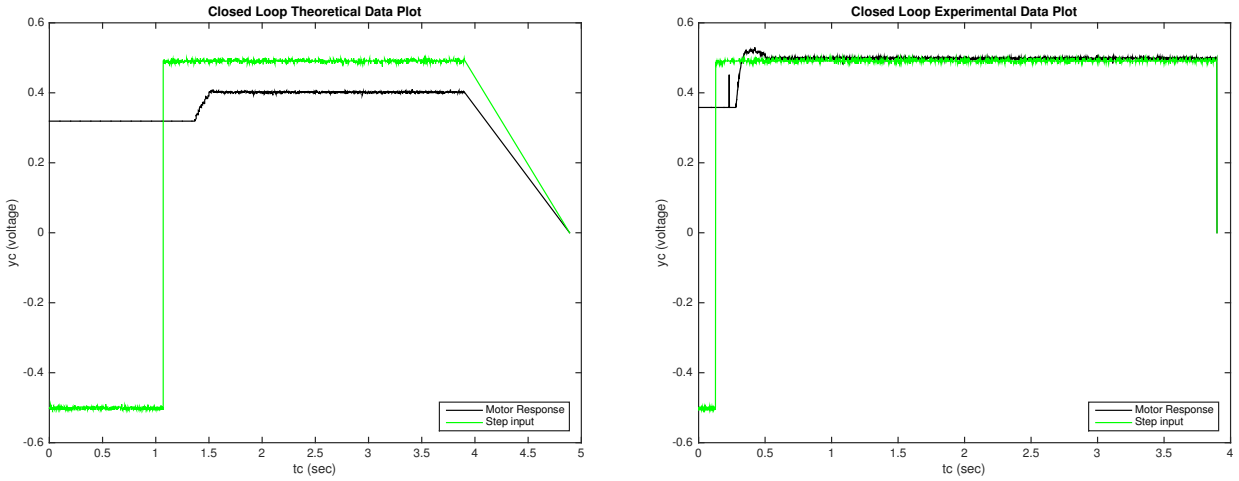
$$K = (\frac{38.61}{2 * 0.69})^2/326$$
$$K = 2.4$$

Therefore, the gain required to achieve a 5% overshoot is as stated above; Moreover, to calculate the desired resistor values to achieve this game, we must make an initial assumption about either R_f or R_1.
Assuming $R_1 = 10k$ (as to avoid changing both resistors);

$$K = 2.4$$
$$\frac{R_f}{R_1} = 2.4$$
$$R_f = 2.4 * R_1$$
$$R_f = 2.4 * 10000$$
$$R_f = 24000$$

Thus, theoretically, to achieve 5% overshoot a gain of $K = 2.4$ is required, to achieve this, $R_f = 24k \approx 22k + 1.8k + 220 = 24.2k$, and $R_1 = 10k$.

6. **Import experimental results into MATLAB. Compare your closed loop response data against your predicted model data $Y_c(t)$. Note any differences between the experimental result and the predicted result. What does this suggest about the model derived in part A and B?**



(a) Theoretical Closed Loop Response

(b) Experimental Closed Loop Response

Figure 28: Servo Motor Closed Response

$$\%OS = \frac{V_{max} - V_{avg}}{V_{avg}} \times 100$$

$$\begin{aligned} \%OS_{theoretical} &= \frac{403.261 - 400.296}{400.296} \times 100 \\ &= 0.7407\% \end{aligned}$$

$$\begin{aligned} \%OS_{experimental} &= \frac{522.863 - 497.888}{497.888} \times 100 \\ &= 5.016\% \end{aligned}$$

The motor response of the theoretical response was derived from $R_f = 25k$ ohms and $R_1 = 10k$ ohms with a calculated overshoot of 5%. As shown in the calculations above, the theoretical percentage overshoot is much smaller. To obtain a 5% overshoot,

7. **Compare your experimentally derived 5% overshot gain value against your predicted value. What is the percentage error? If your overshoot was too large for your derived gain value, could you use a controller other than the proportional controller to reduce overshoot?**

   If your steady state error was large, what other controller type could you use to minimize this error? What are the drawbacks of this type of controller? What control applications can you think of that require very low steady state error?

   From the procedure outlined in **experiment C**, an experimental gain of $K = 6.28$ resulted in an overshoot of 5%, and in **step 5** it was shown that the theoretical gain required to achieve this was $K = 2.4$.

   The percentage error between the calculated and experimentally found gain values has been calculated below.

$$\%_{error} = |\frac{K_{theoretical} - K_{experimental}}{K_{experimental}}| \times 100$$
$$\%_{error} = |\frac{6.28 - 2.4}{2.4}| \times 100$$
$$\%_{error} = 61.8\%$$

   A large % error value, as indicated above; provides strong evidence for model inaccuracies. The difference in gain between the theoretical and the practical values is likely caused losses that have been neglected from the simulation.

   The overshoot resulting from the derived gain value was under 5%.

   Steady state error was noticeable in the system, likely caused by mechanical losses (gearbox, heat, etc). To alleviate this issue, a PI compensator (integrator) could be used; this is because the PI controller adds up error over time and would remove the steady state error. Moreover, a PI controller retains a maximum overshoot and settling time similar to that of the proportional controller (adjusting the gain alone).

   However, the drawbacks for a PI controller, depending on the application may include; the relatively faster settling time, or the possible overshoot, when compared to the effects of a PD controller.

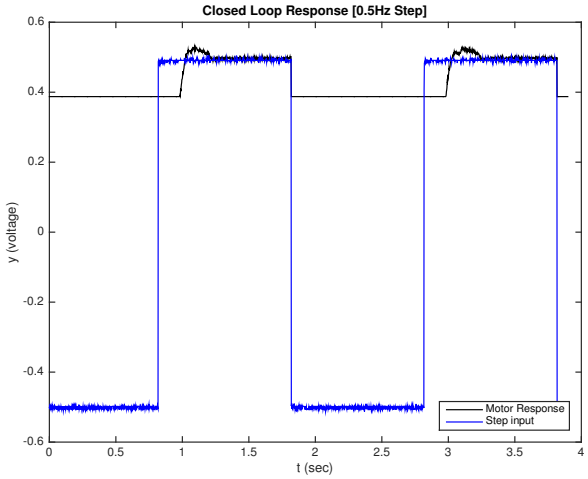
   A good example of a control system that requires low steady state error is the cruise control on a car. This is largely to do with its specific application where the output were to be increased beyond the target output due to stead state error, the result could be decremental to safety, cause injury or even fatality. Moreover, any overshoot in the response would also have the ability to incur the previously mentioned effects. However, the systems response time is arguably less important, in fact it is dispersible to have a slower response time to avoid a large acceleration which would result in larger forces acting upon the driver, and could even reduce the vehicles stability or traction.

   As mentioned previously, a PI compensator would remove the steady state error, but a PID controller would be the likely choice for a cruise control system, as elements from both a PI and a PD controller are wanted. The lack of steady state error, and a longer settling time and no peak overshoot are all possible when using a PID controller (with a focus on the derivative element).

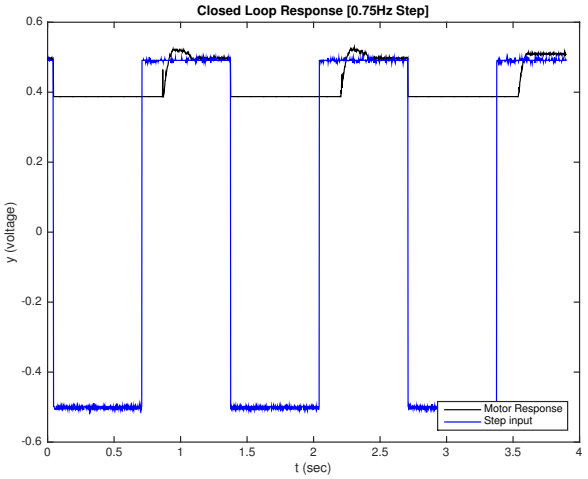
8.2.3 Part D

1. If time permitted, a bodeplot would be included.

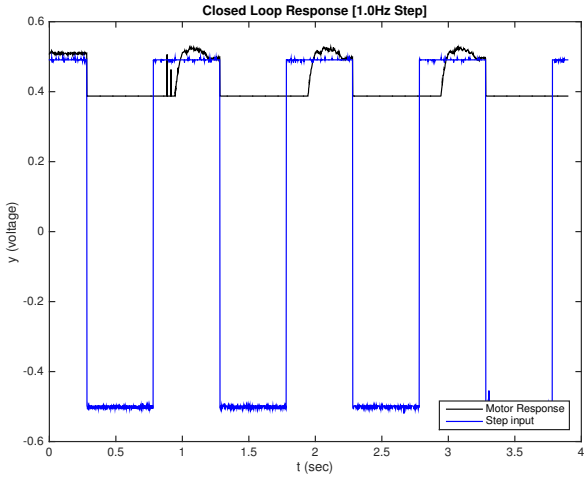
2. The overshoot was measured over five different input frequencies in order to determine the effect over varying the input signals frequency. The systems response to these signals was also recorded, and can be found as a set of graphical time domain plots below.



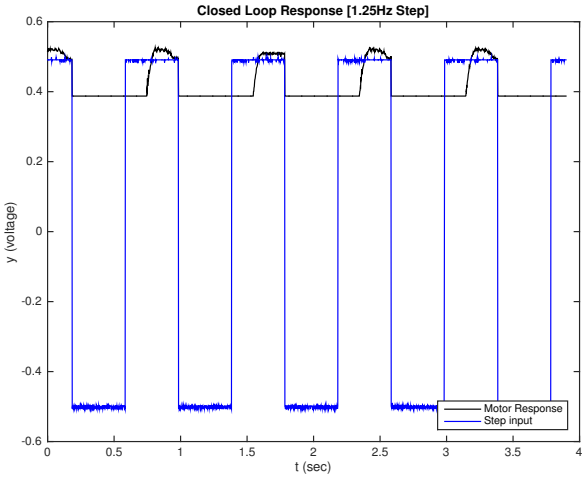
(a) Closed loop time domain response to 0.5Hz input.



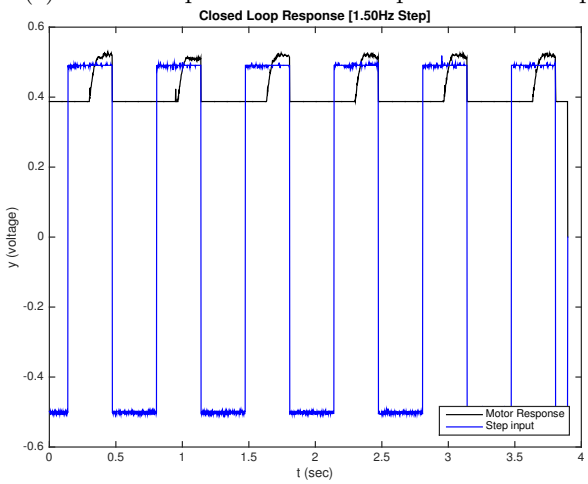
(b) Closed loop time domain response to 0.75Hz input.



(c) Closed loop time domain response to 1Hz input.



(d) Closed loop time domain response to 1.25Hz input.



(e) Closed loop time domain response to 1.5Hz input.

Figure 29: Systems response to various input frequencies.

The percentage overshoot was calculated using the cursors on the digital CRO machine to measure the peak overshoot and the steady state voltage (average).

$$\%OS = \frac{V_{peak} - V_{average}}{V_{average}}$$

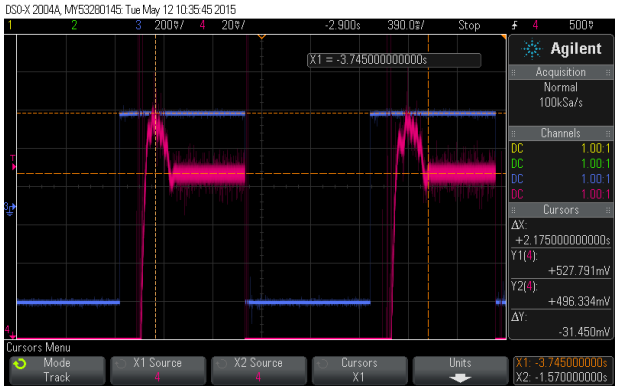
$$\begin{aligned} OS(0.5Hz) &= \frac{525 - 495}{495} = 6.1 \\ OS(0.75Hz) &= \frac{523 - 495}{495} = 5.7 \\ OS(1Hz) &= \frac{525 - 495}{495} = 6.1 \\ OS(1.25Hz) &= \frac{525 - 495}{495} = 6.1 \\ OS(1.5Hz) &= \frac{524 - 495}{495} = 5.8 \end{aligned}$$

The results of this analysis can be found in the following table.

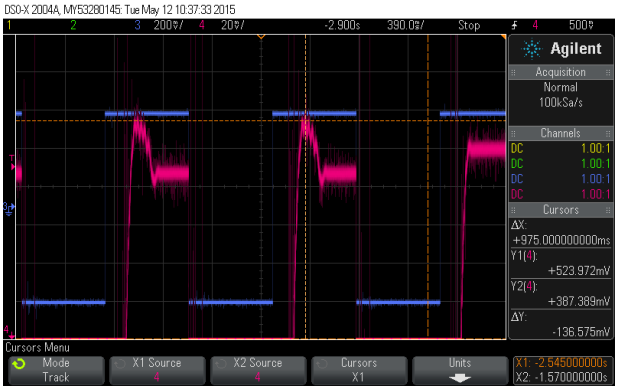
Input Frequency (Hz)	Overshoot (%)
0.5	6.1
0.75	5.7
1	6.1
1.25	6.5
1.5	5.8

As can be seen almost immediately in the previous plots, as the frequency of the input signal is increased, the output has less time to reach a steady state value. Eventually, the increasing frequency limits the systems ability to reach a steady state value.

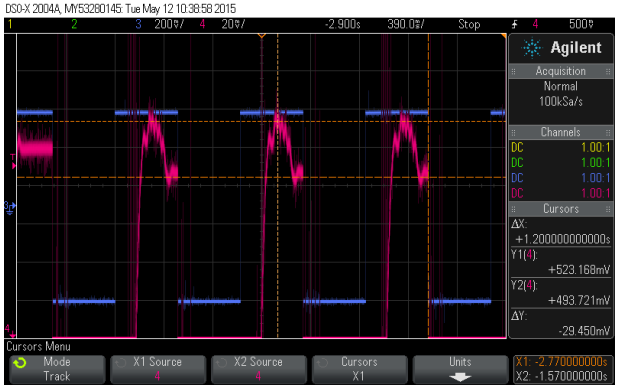
It is also evident from the previous figures and tables, that neither the overshoot, settling time or the steady state error are effected; until eventually the system does not have enough time to reach a steady state value. In addition to this, were the frequency to keep increasing, eventually the op-amps physical components would not be able to switch fast enough, and the output would be attenuated.



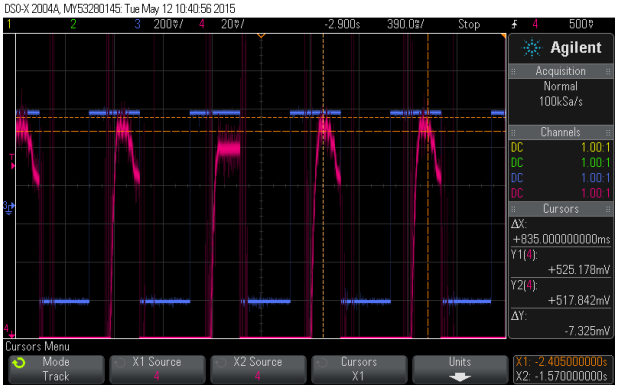
(a) Overshoot 0.5Hz



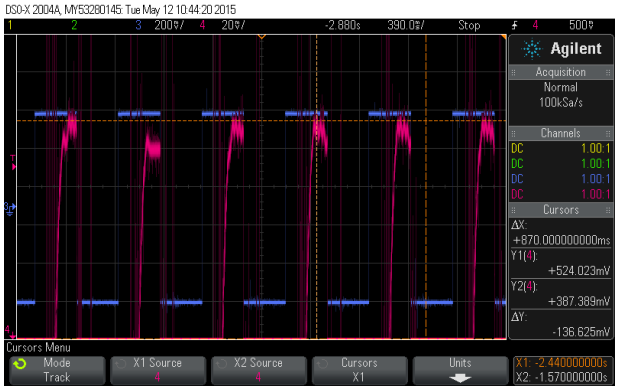
(b) Overshoot 0.75Hz



(c) Overshoot 1Hz

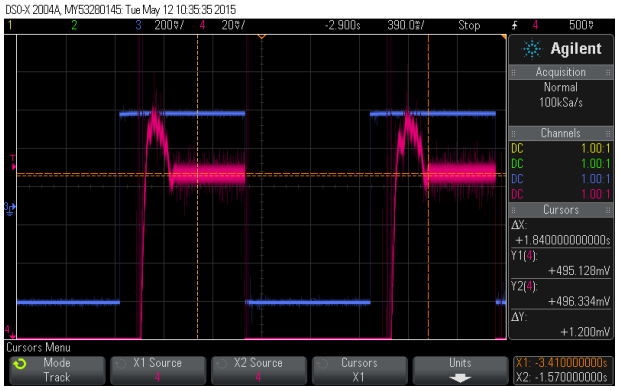


(d) Overshoot 1.25Hz

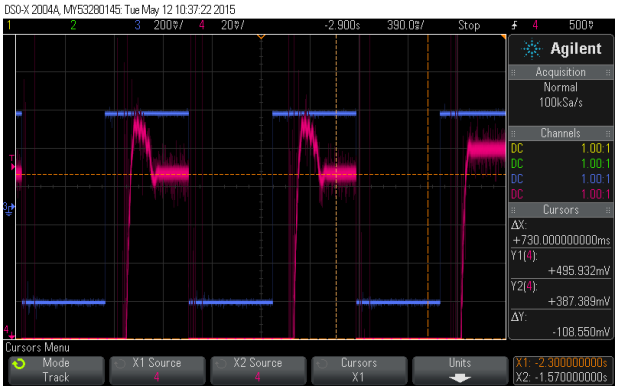


(e) Overshoot 1.5Hz

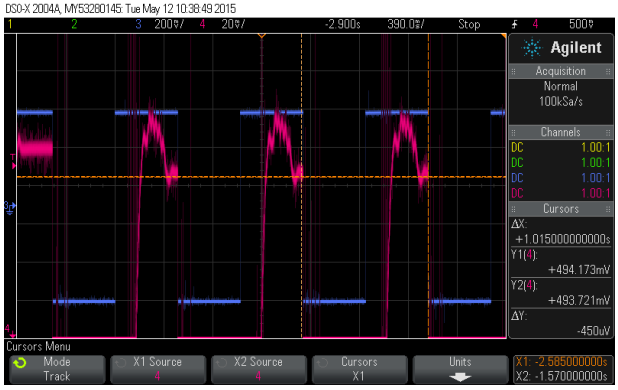
Figure 30: Oscilloscope Overshoot Measurements



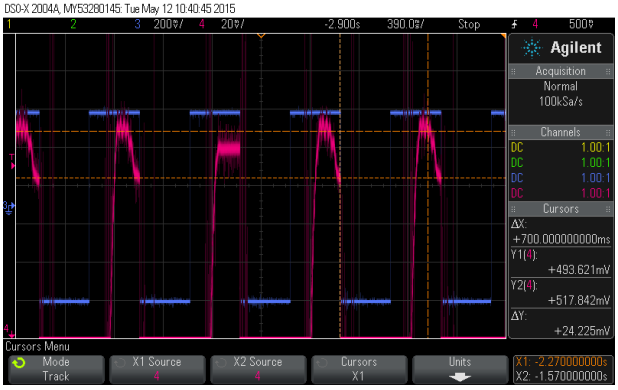
(a) SSE 0.5Hz



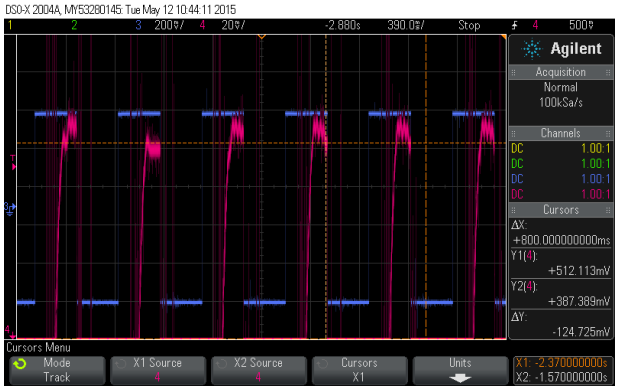
(b) SSE 0.75Hz



(c) SSE 1Hz



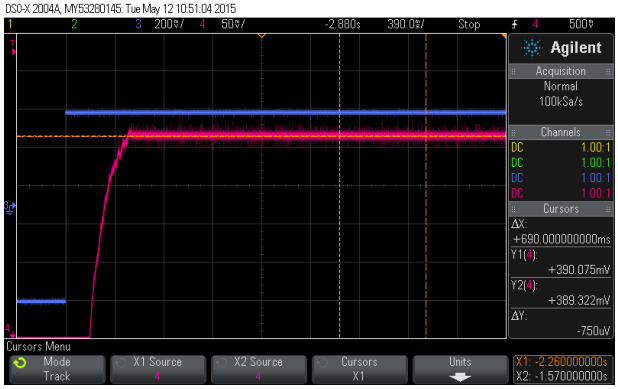
(d) SSE 1.25Hz



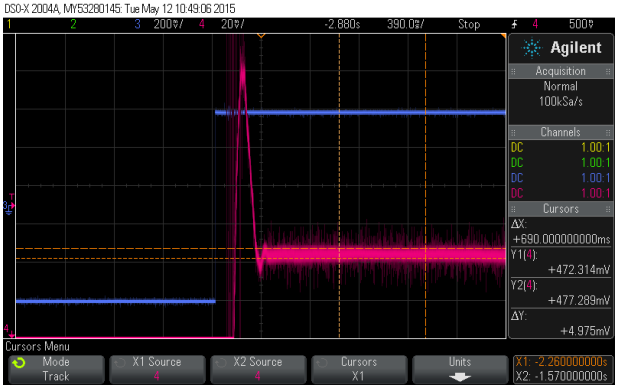
(e) SSE 1.5Hz

Figure 31: Oscilloscope SSE Measurements

3. To examine the impact of changing the gain value, the systems response data has been captured, recorded, and displayed below.

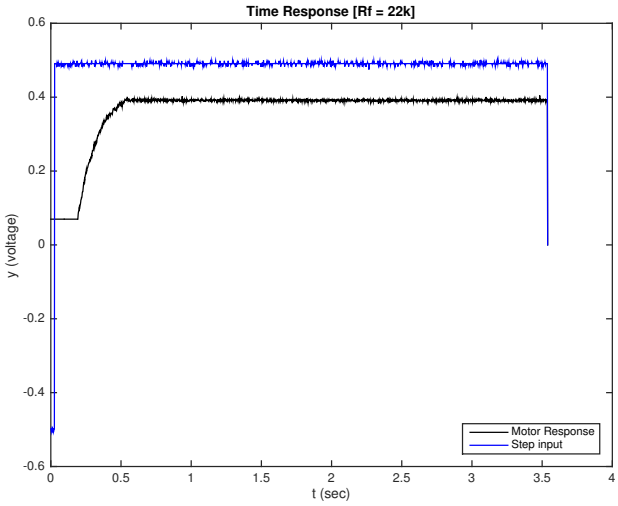


(a) Oscilloscope System Response to low gain ($R_f = 22kohms$).

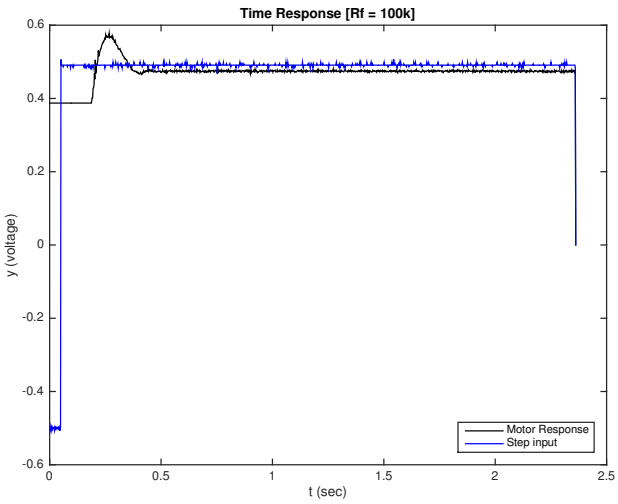


(b) Oscilloscope System Response to high gain ($R_f = 100kohms$).

Figure 32: Systems response to various gain values.



(a) Oscilloscope System Response to low gain.



(b) Oscilloscope System Response to high gain.

Figure 33: Systems response to various gain values.

As you would expect, changing the gain value proportionally affects the peak overshoot and settling time. The above images show large evidence to support this comment; as the correlation between these values can be seen clearly. From the equations required to calculate $\%OS$ and T_s mentioned previously, it can be seen that the aforementioned statement is correct, as increasing the gain increases both the overshoot and settling time. This is extremely useful in control systems engineering, as changing the gain alone can alter system properties such as those mentioned previously. This type of controller is known as a proportional controller, and has an extremely large number of real world applications.

```
%% D3
data = csvread('PartD3_22k.csv',2,0);  % Read in 0.5Hz
td_22k = data(1:end,1);    % Store tc variable
%td_22k = td_22k + abs(td_22k(1));
yd_22k = data(1:end,3);    % Store yc variable
yd_22kStep = data(1:end,2);    % Store tc step input variable
[td_22k, yd_22k,yd_22kStep] = timing_fix_D3(td_22k,yd_22k,yd_22kStep);

figure
plot(td_22k,yd_22k,'k',td_22k,yd_22kStep,'b')
title('Time Response [Rf = 22k]')
xlabel('t (sec)')
ylabel('y (voltage)')
legend('Motor Response', 'Step input', 'Location','SouthEast')
print('-depsc',strcat('figures',filesep,'D3_22k'));
close

function [ te_new,ye_new, ye_stepNew ] = timing_fix_D3(te,ye,ye_step)
    te_new = te;
    ye_new = ye;
    ye_stepNew = ye_step;

    % Remove initial zero gradient before resonse
    f = ye_new > ye_new(1) * 1.05;
    indice = find(f,1,'first');
    ye_new = ye_new(indice-100:end);
    te_new = te_new(indice-100:end);
    ye_stepNew = ye_stepNew(indice-100:end);

    % Shift time to start at zero
    te_new = te_new + abs(te_new(1));

    % Shift amplitude to start at zero
    %ye_new = ye_new + ye_new(1);
end
```

4. **Build closed loop model of servo motor and replicate the closed loop response obtained in Part C step 6.**

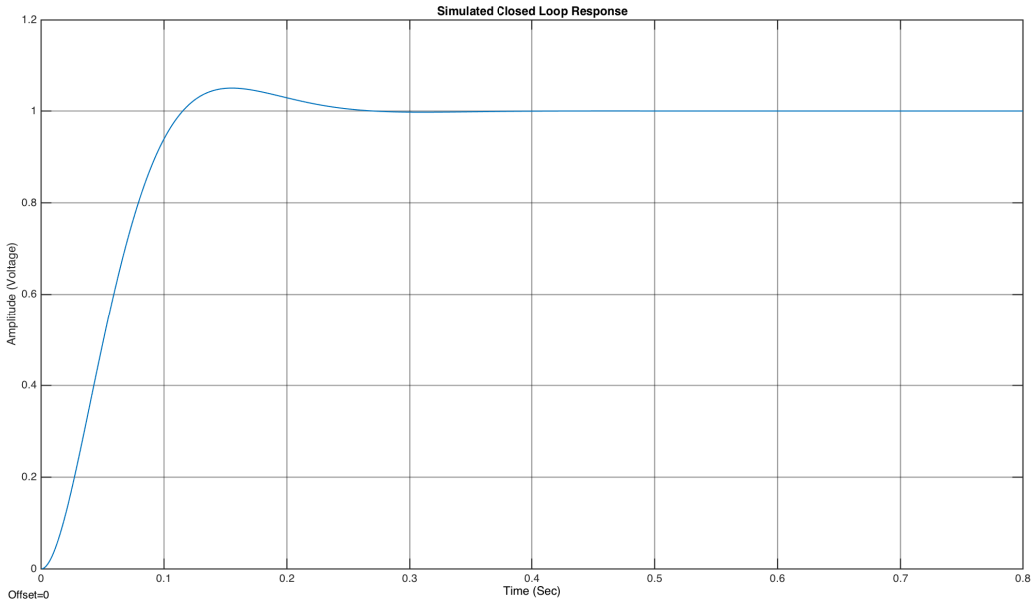


Figure 34: Simulated PID Controller

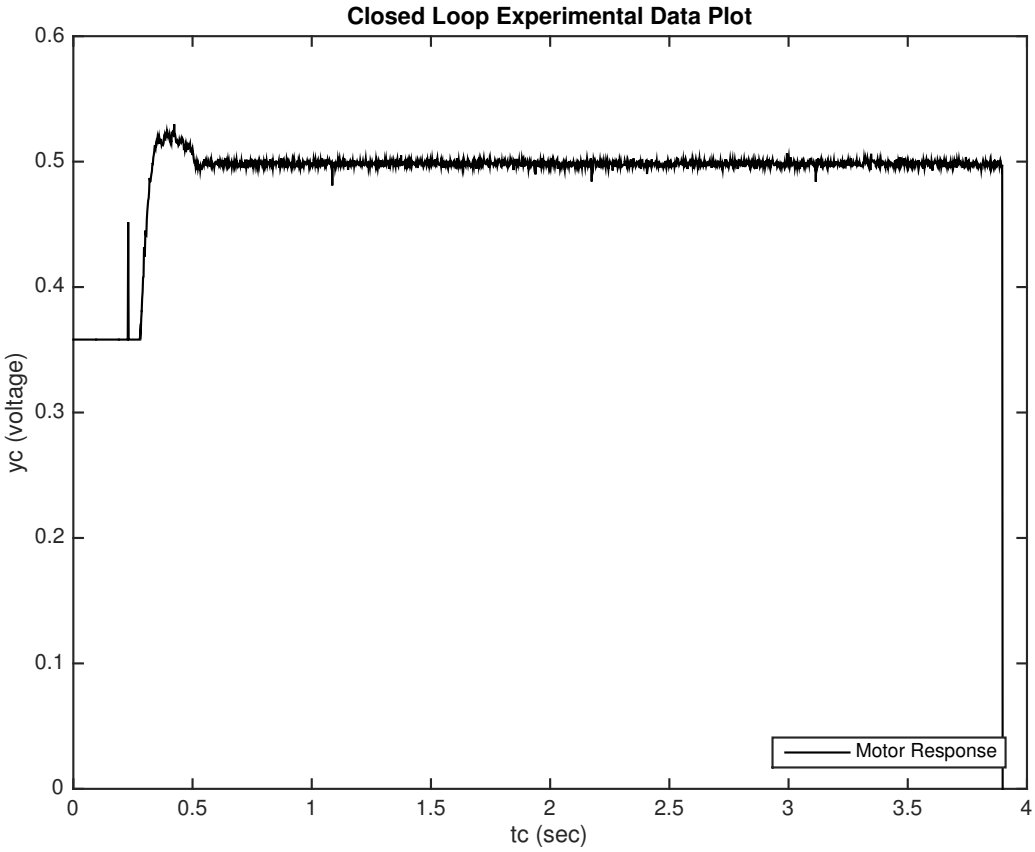


Figure 35: Closed loop response Experimental

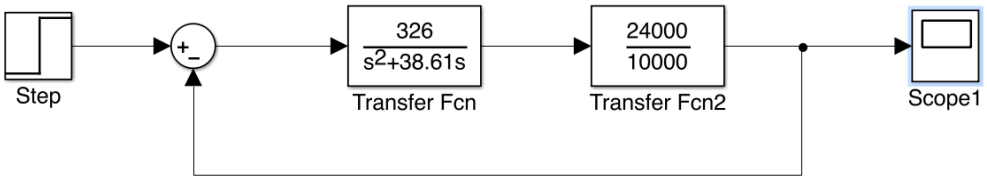


Figure 36: Simulated PID Controller Simulink

5. **Propose a PID controller that achieves a faster response. Simulate close-loop response and plot on same graph as the closed loop reponse of the Part C step 6 controller**

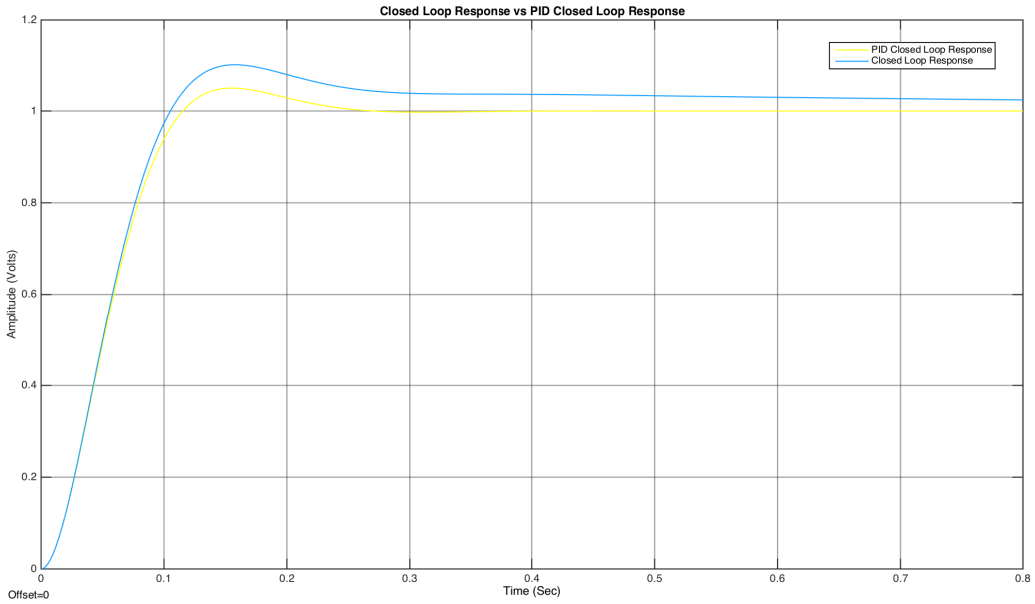


Figure 37: PID Controller Vs Closed Loop Response

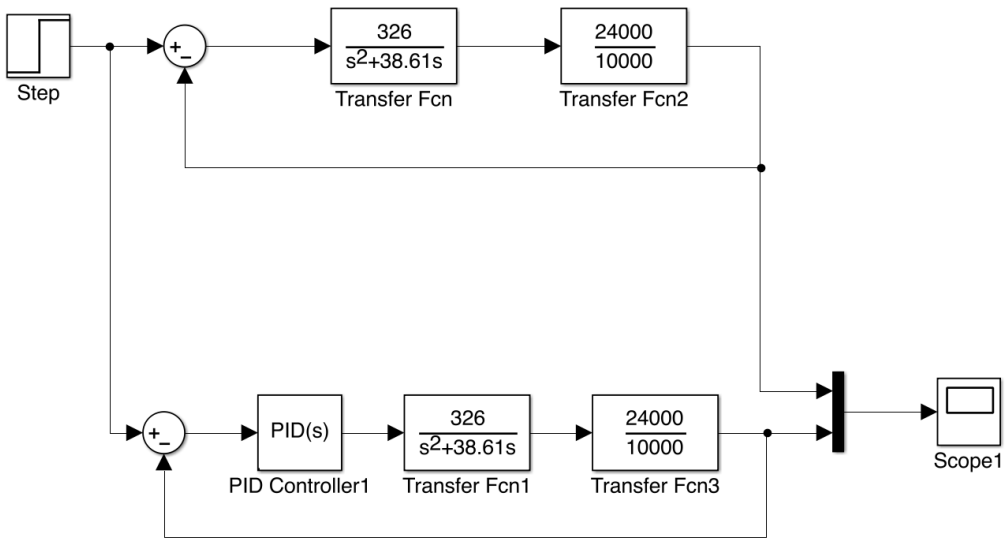


Figure 38: PID Controller and Closed Loop Response Simulink

8.3 Prac Code

This section contains all matlab code used during the practical lab. Lachlan Nicholson (n8866864) and Declan Gilmour (n8871566) worked in a prac group with Thomas Wagner (N8840121) and Antony Foster (N8647780).
This code was produced by Thomas Wagner N8840121 and Antony Foster N8647780.

```
%------------B1. Loading data---------------------------%
[te,ye] = Data_cutting(closed,2,1);
[te,ye] = timing_fix(te,ye);
figure(i);i=i+1;
plot(te,ye,'k');
title('Test data plotted');
ylabel('Amplitude');
xlabel('Time_{sec}');

%------------B2 Plotting y2 and Experimental data---------%

Vm=1;
y3=step(G_1*Vm,te);
figure(i);i=i+1;
plot(te,y3,te,ye);
title('Test data Vs Calculated Km and Alpha ');
ylabel('Amplitude');
xlabel('Time_{sec}');

%-------------B3 Figure of merit-------------------------%
y_rms =@(x0)Variable_finder2(x0,'results.csv',te,ye);

%-------------B4 Recalculated values---------------------%

params = fminsearch(y_rms,[Km, a]);
y3_num =  [params(1)];
y3_den = [1 params(2) 0];
G_3 = tf(y3_num,y3_den)
y3= step(G_3*Vm,te);

figure(i);i=i+1;
plot(te,y3,te,ye);
title('Test data Vs ReCalculated Km and Alpha');
ylabel('Amplitude');
xlabel('Time_{sec}');
```