ENB301: Practical Report

Lachlan Nicholson (n8866864) Declan Gilmour (n8871566)

22nd May, 2015

1 Executive Summary:

1/2 paragraph overview of the document

2 Aim:

1-2 sentences on the report purpose
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:

0.5-1 page overview of the lab

4 Procedure:

brief summary of prac procedure as described in this document (0.5-1page for each part (bcd))
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.

4.1 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 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. Fifth step talks about selecting values for K_m and β based on an α value :s

4.2 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 12 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.

4.3 Experiment D

The final experiment, using the same circuit constructed in the previous experiment, examined the response of the system to different input frequencies. **and more**

SETUP:

1. **using old circuit with set gain**

ANALYSIS:

1. The closed loop response of the system was estimated **using the method outlined in the previous procedures**
2. The K_m and α values were then compared to **the method outlined in worksheet**
3. 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 **and settling time** were measured and recorded, for input frequencies of 0.5Hz, 0.75Hz, 1Hz, 1.25Hz and 1.5Hz.
4. The impact of altering the gain was then examined **BY STUFF IDK**
5. simulink/matlab shit
6. PD/PID controller

5 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.

5.1 Part B

calculate predicted K_m and α values. correct polarity of the motor above B5

5.2 Part C

closed loop control system for the position of the motor overall TF estimated gain for 5% OS practical gain for 5% OS comparisons C8

5.3 Part D

the response of the closed loop system to different input frequencies C8

6 Discussion/Recommendations:

0.5-1page

7 Appendices

7.1 Pre-lab:

7.1.1 Lachlan Nicholson

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

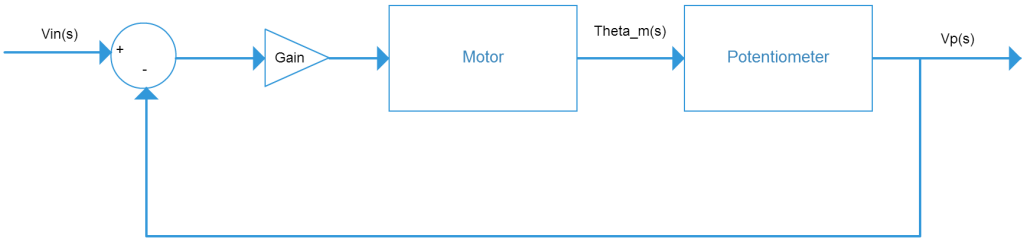


Figure 1: Functional Diagram of the control system

2. The updated functional diagram has been included below.

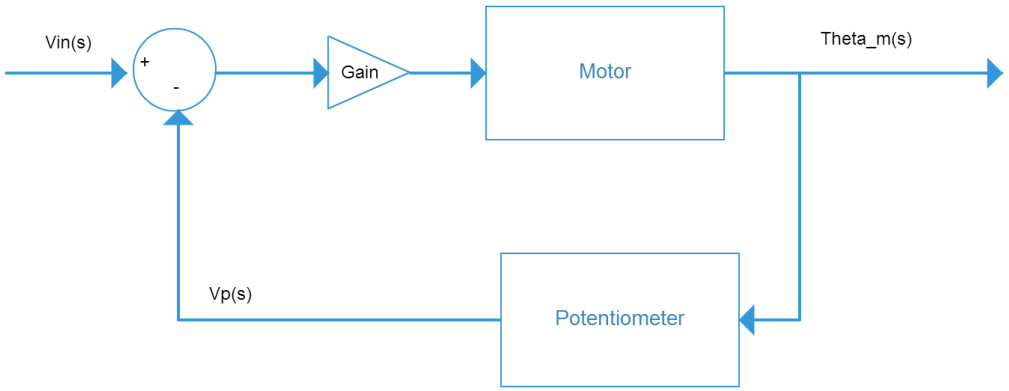


Figure 2: Updated Functional Diagram

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

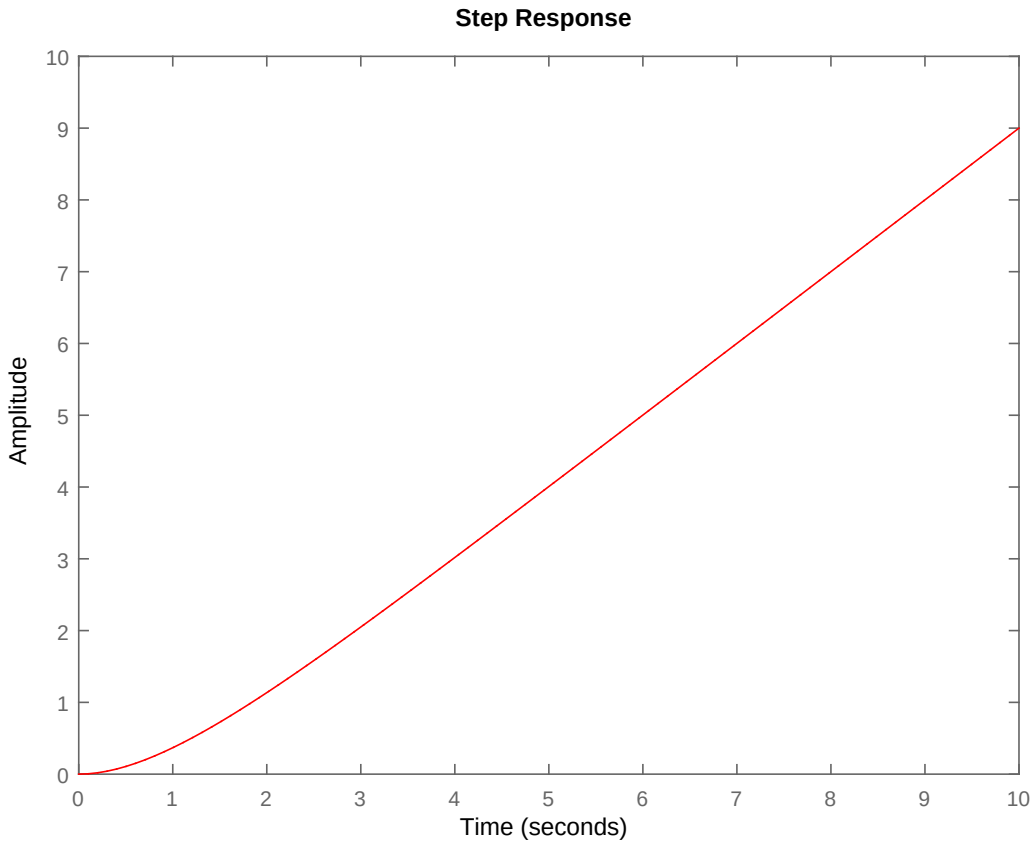


Figure 3: Simulated Response

```
%% 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.

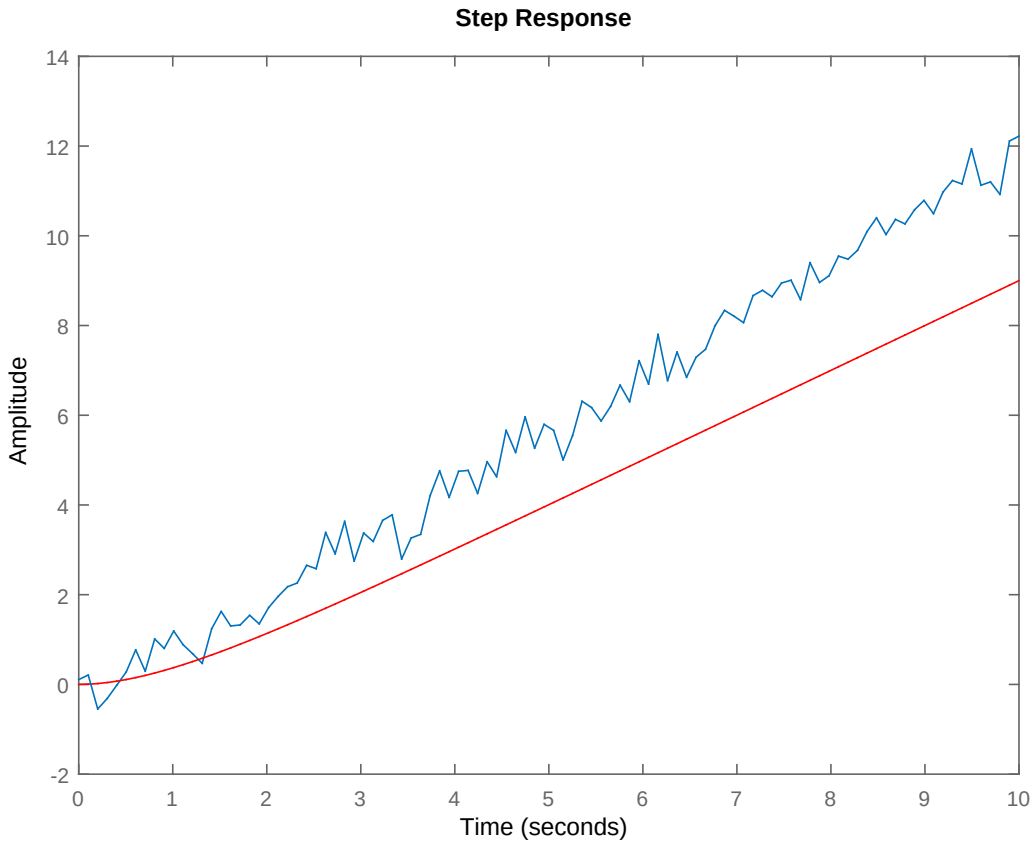


Figure 4: Given Data and Estimated Response

```
%% 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. **steady/transient Q**

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.

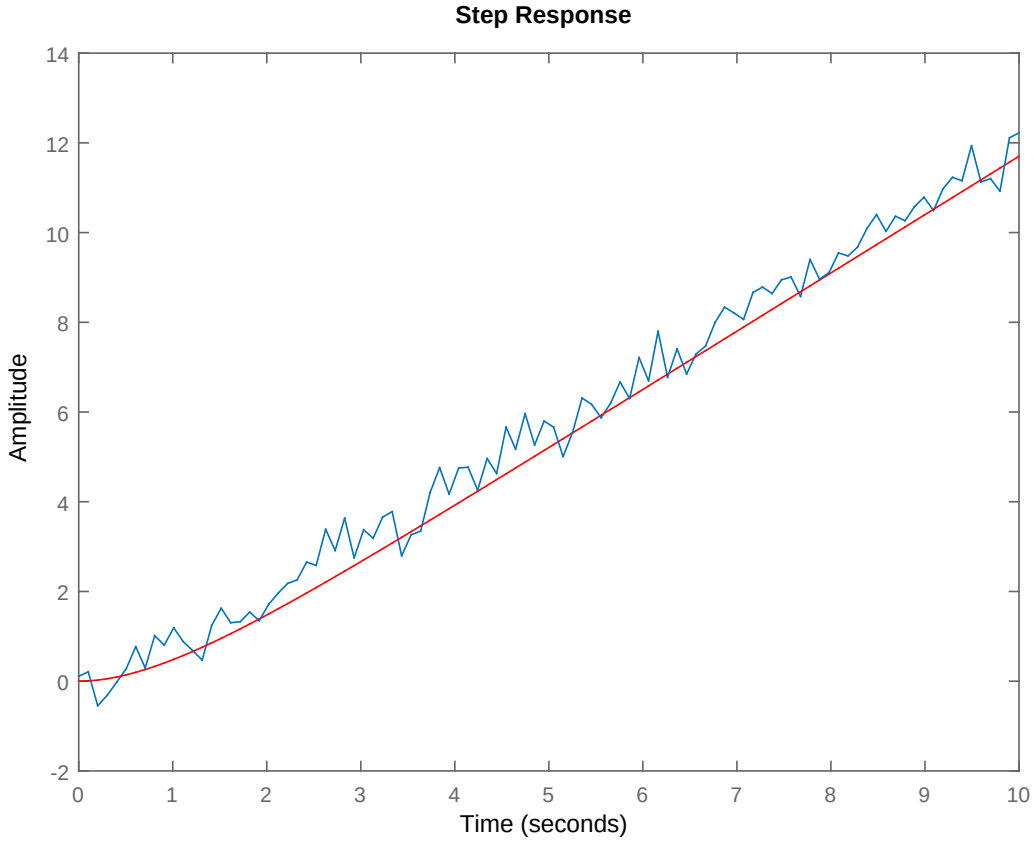


Figure 5: Manually Estimated System Values

```
%% 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')
```

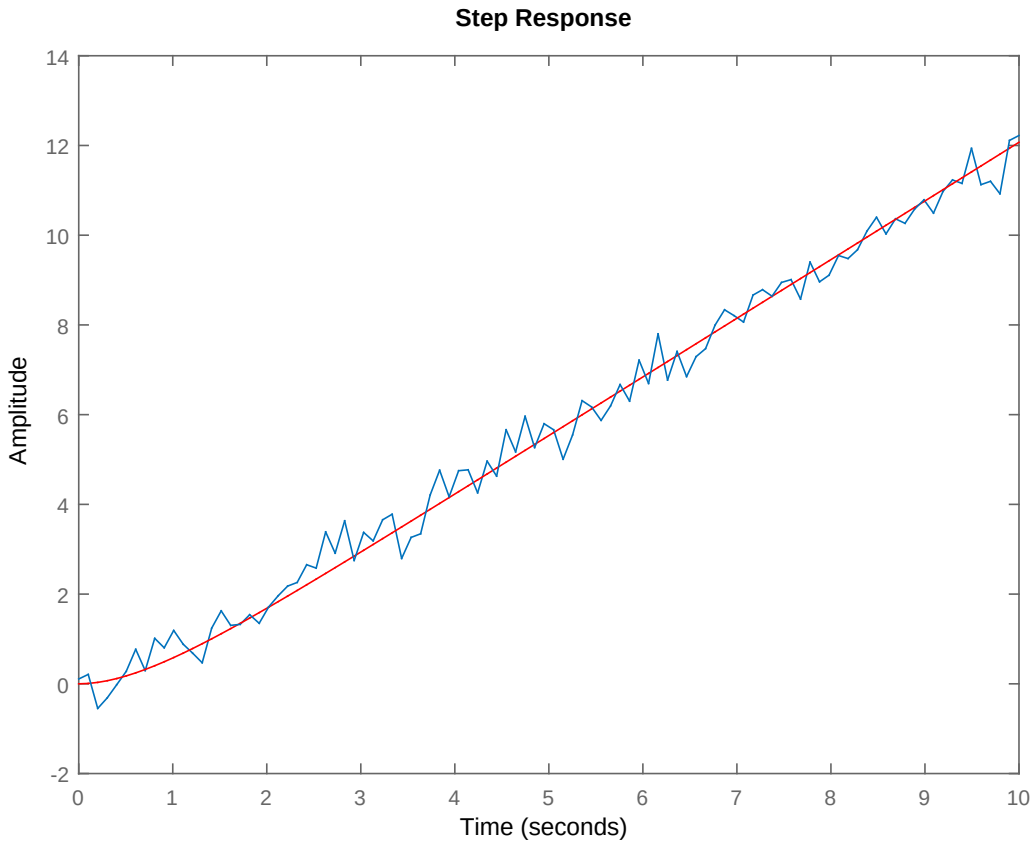


Figure 6: Inverting Amplifier System

```
%% 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. **QUESTIONS**

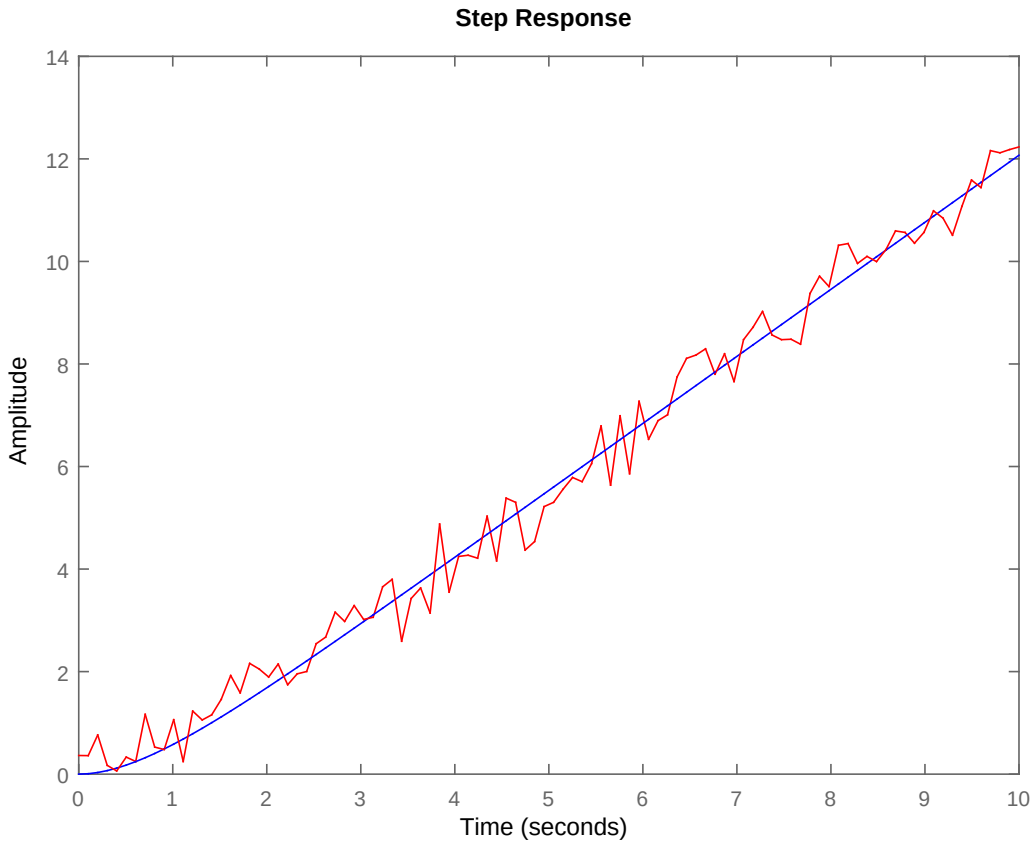


Figure 7: Estimated System Response + Uncertainty

```
%% 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')
```

7.1.2 Declan Gilmour

1.

2.

3.

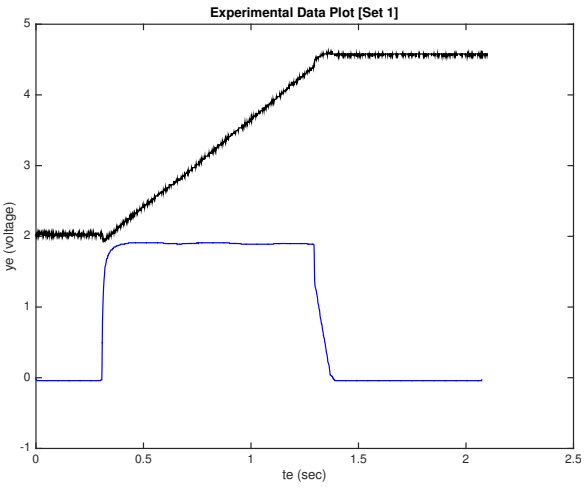
4.

7.2 Answers:

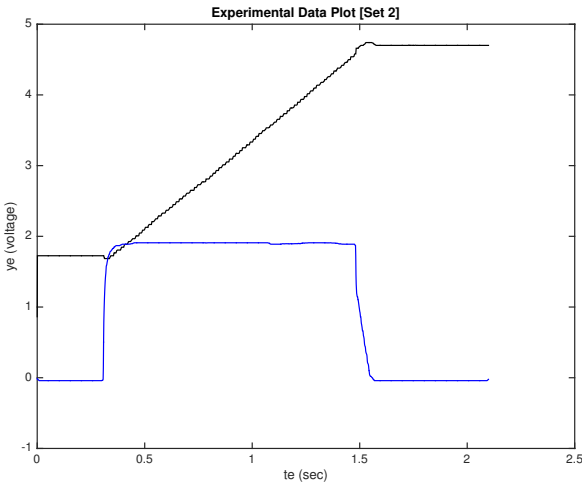
7.2.1 Part B

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

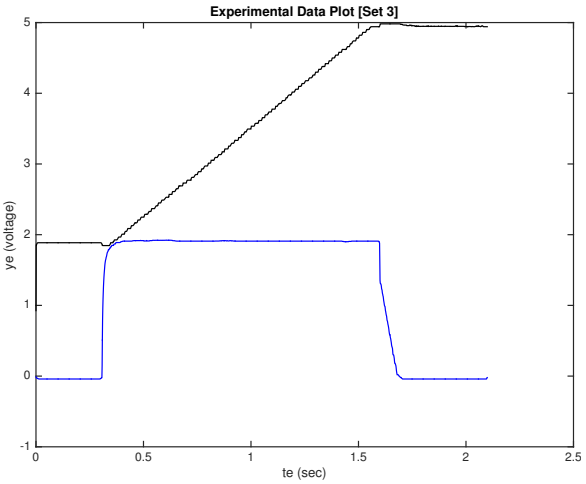
```
%% B1
% Load experimental data and plot in black
data = csvread('PartB_Test1.csv',2,0);  % Read in test 1
te_1 = data(1:end,1);   % Store te variable
te_1 = te_1 + abs(te_1(1)); % Move te variable to start at zero
ye_1 = data(1:end,2);   % Store ye variable
ye_1step = data(1:end,3);   % Store ye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



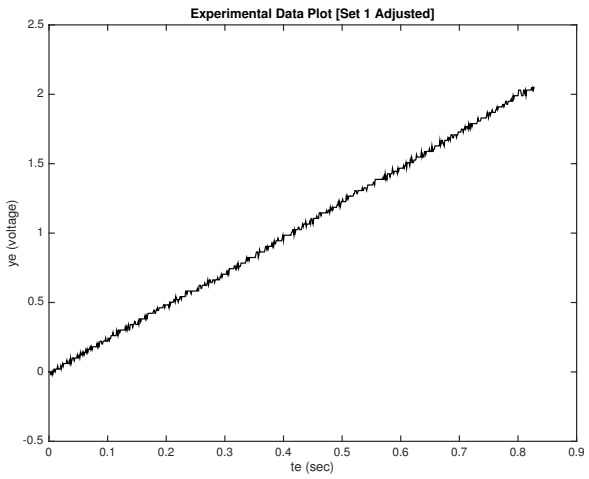
(c) Data Set 3

Figure 8: Open Loop response

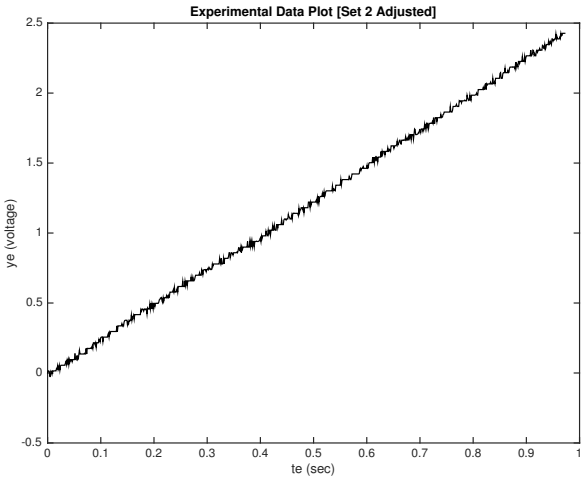
2. Modify the MATLAB script created in the pre-labs to simultaneously plot both the experimental data and $y_1(t)$. No real idea what to put here...

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: $error = \sqrt[2]{(G_0 - y_e))^2}$

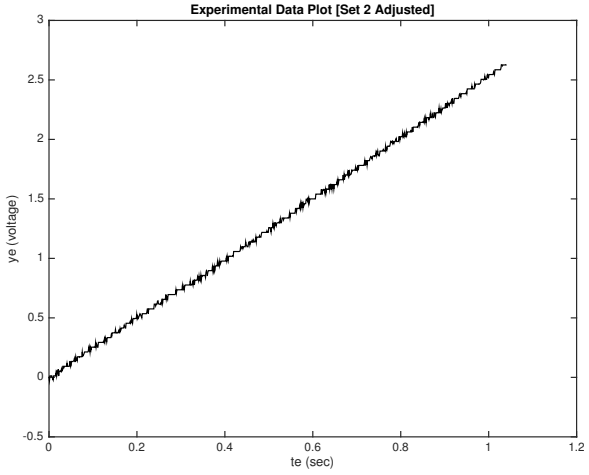
4. Improve estimates using the plots and figure of merit calculations. Derive the two parameters for the servo motor function $G(s) = \frac{K_m}{(S+\alpha)(S+\beta)}$.



(a) Data Set 1



(b) Data Set 2



(c) Data Set 3

Figure 9: 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
```

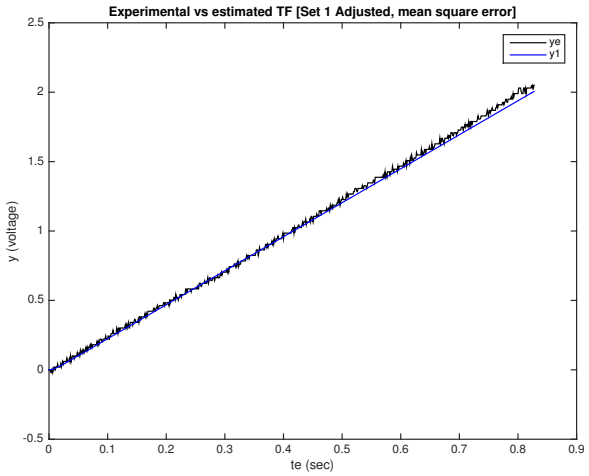
```
% Pepare experimental data for alpha and km parameter calculations
[te_1trim, ye_1trim] = trimForCalculation(te_1new,ye_1new,mf1);
[te_2trim, ye_2trim] = trimForCalculation(te_2new,ye_2new,mf1);
[te_3trim, ye_3trim] = trimForCalculation(te_3new,ye_3new,mf1);

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
```

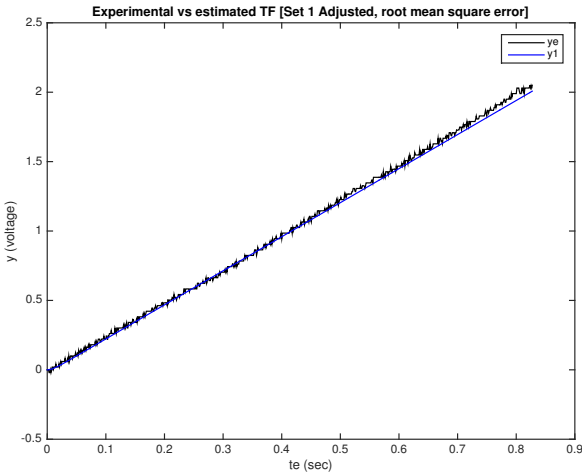
```
end
km_mean = (mean(km_ms) + mean(km_rms)) / 2;
alpha_mean = (mean(alpha_ms) + mean(alpha_rms)) / 2;
```

```
% 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
```

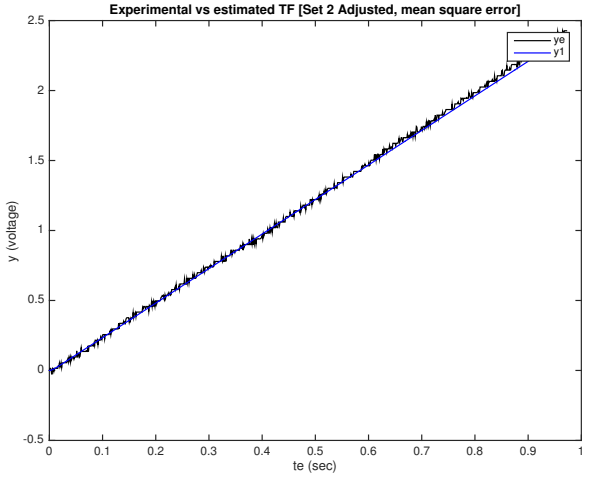
The average constants found are as follows: $\alpha = 170.3407$ $k_m = 422.0107$



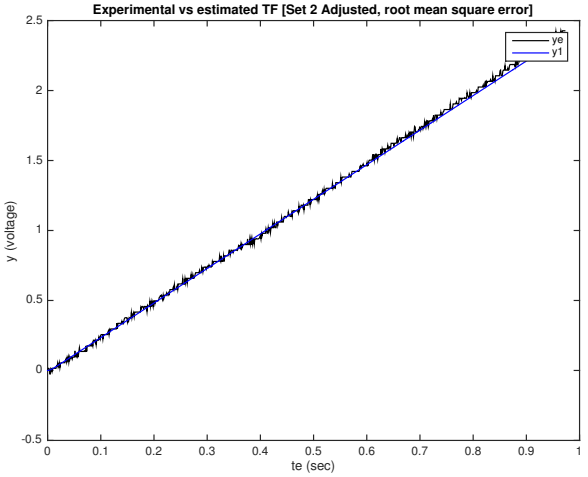
(a) Data Set 1 (ms)



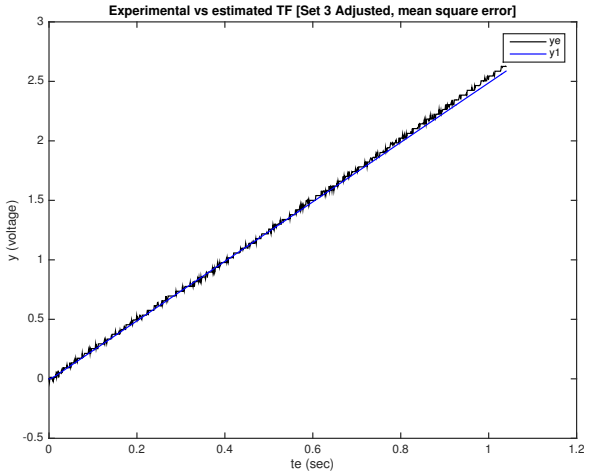
(b) Data Set 1 (rms)



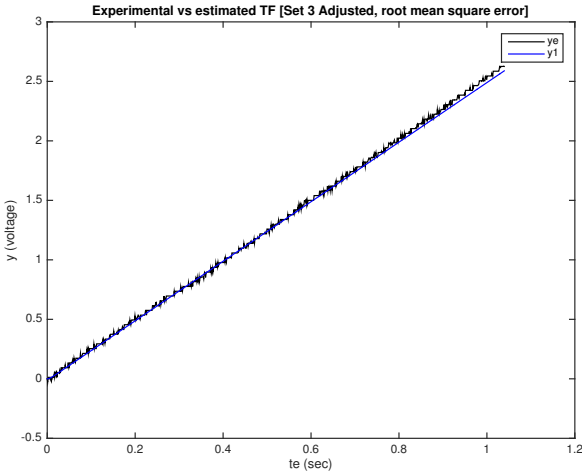
(c) Data Set 2 (ms)



(d) Data Set 2 (rms)



(e) Data Set 3 (ms)



(f) Data Set 3 (rms)

Figure 10: Servo Motor Response Modeled

5. 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: $\beta = 0.0646$ $k_m = 426.6867$

```
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
```

7.2.2 Part C

1. **Re-draw figure 12 and place boxes around the set of components that correspond to each functional element of the control system.**

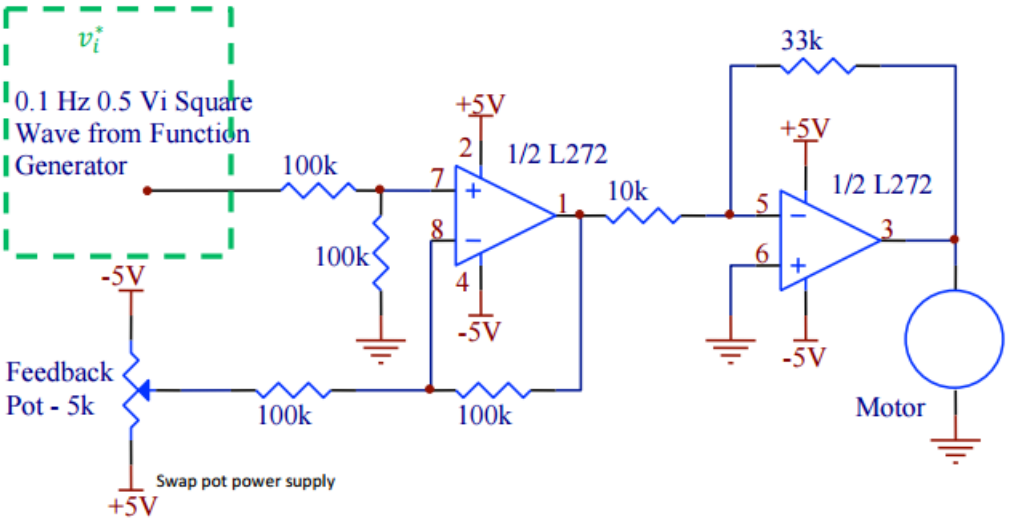


Figure 12: Closed Loop Motor Control Schematic

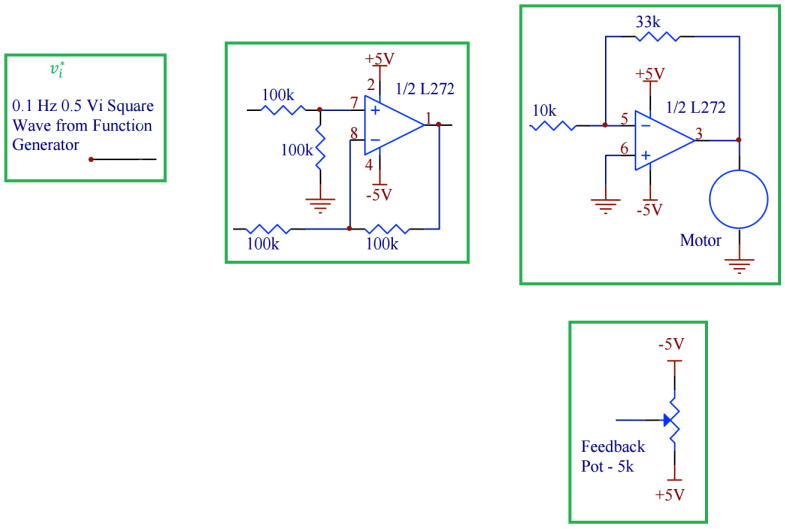


Figure 13: Closed Loop Motor Control System

2. **Based on the results from Parts A and B, the component values given in figure 12 and your research in parts C1, calculate all of the transfer functions in your functional diagram (figure 13). Update the functional diagram, labeling all components and interfaces.**

Difference Op-amp:

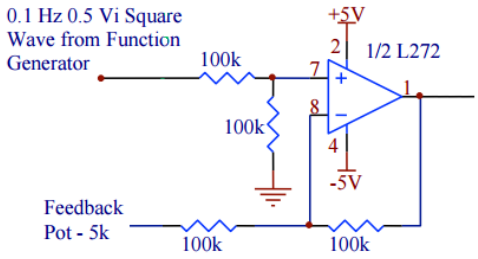


Figure 14: Difference Op-amp System

$$\frac{V_1(s)}{V_{in}(s)} = V_{in}(s) - V_p(s)$$

Inverting Op-amp:

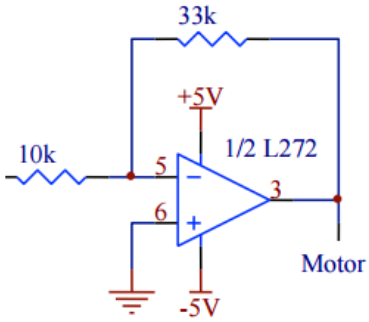


Figure 15: Inverting Amplifier System

$$\frac{V_m(s)}{V_1(s)} = \frac{R_2}{R_1}$$

Motor and pot:

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

something

3. **NONIDEAL:**

$$
\begin{aligned}
V_m(s) &= \frac{R_2}{R_1} V_1(s) \\
V_m(s) &= \frac{s(s+\alpha)}{K_m} V_p(s) \\
\frac{R_2}{R_1} V_1(s) &= \frac{s(s+\alpha)}{K_m} V_p(s) \\
V_1(s) &= \frac{R_1 s(s+\alpha)}{R_2 K_m} V_p(s) \\
V_1 &=
\end{aligned}
$$

IDEAL:

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

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

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

Where $K_m = 326$, $\alpha = 38.61$, $R_f = 33k$ 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. **C4 - Matlab - Declan?**

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. **C6 - Matlab - Declan?**

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 $K = 2.4$.

   Percentage error = blah

   something

   Steady state error **was** noticeable in the system, likely caused by mechanical losses (gearbox, heat, etc). To alleviate this issue, an PI compensator (integrator) could be used; this is because the controller adds up error over time, and would **know that it had still not yet reached the desired output**. Whereas, currently; the error signal of the proportional controller **does what?**.

   Drawbacks/application?

   Add the actual OS from prac/theo

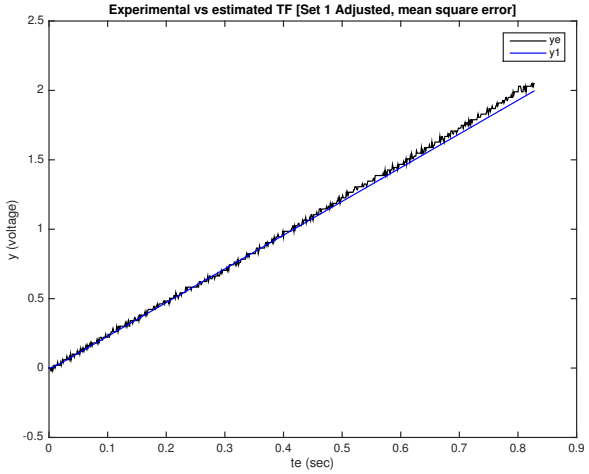
8. Another method of calculating TF? (bode)

7.2.3 Part D

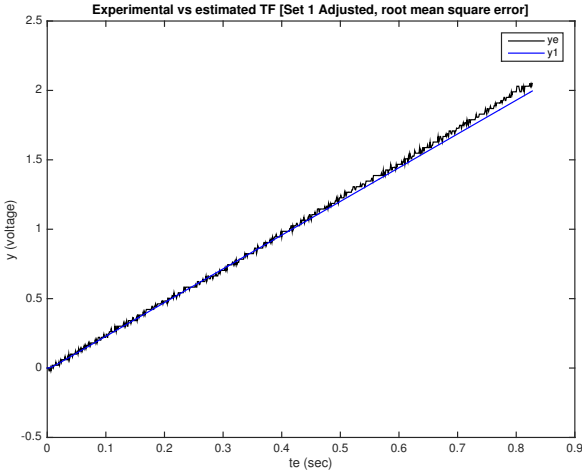
1. stuff
2. comment
3. gain
4. simulink
5. PD/PI/PID controller

8 SHIT

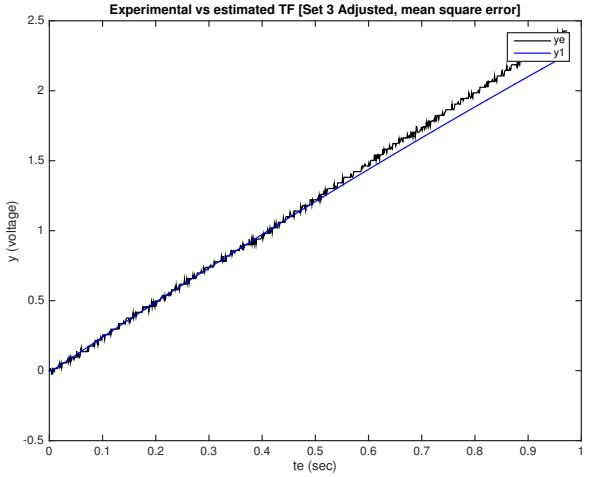
NOTES: on first page, acknowledge assistance from students list group members / lab members



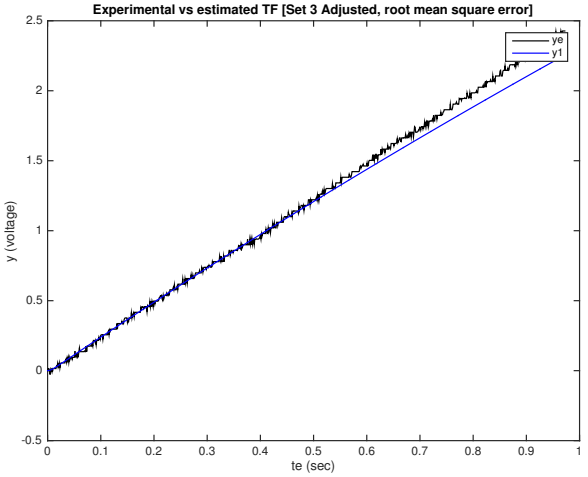
(a) Data Set 1 (ms)



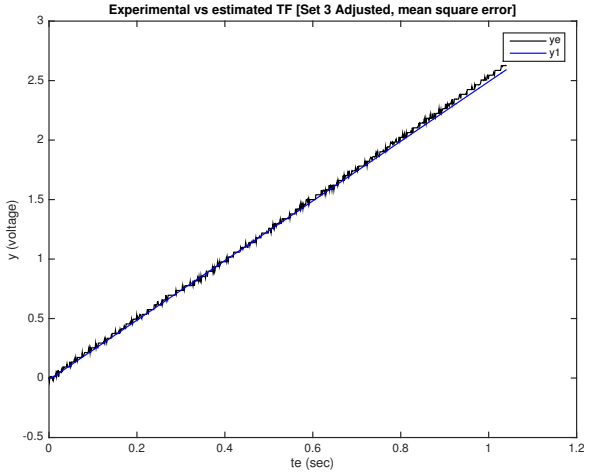
(b) Data Set 1 (rms)



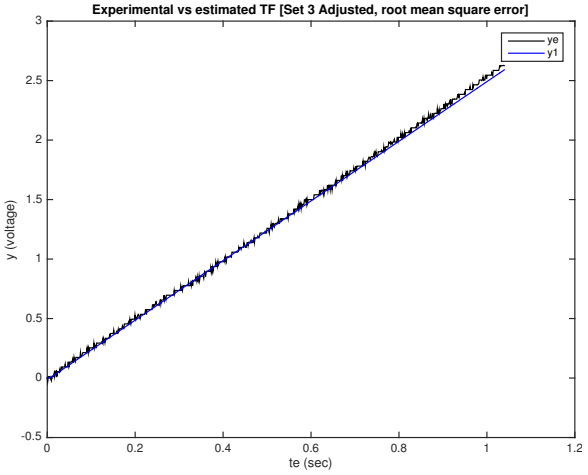
(c) Data Set 2 (ms)



(d) Data Set 2 (rms)



(e) Data Set 3 (ms)



(f) Data Set 3 (rms)

Figure 11: Servo Motor Response Modeled: k_m and β