

ENB301: Practical Report

Lachlan Nicholson (n8866864) Declan Gilmour (n8871566)

24th 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 suggested 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 transferred 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 :

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**

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. Additionally, the systems response for each frequency was also recorded and saved in-case of future need.
5. 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.
6. simulink/matlab shit
7. 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

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 12, the full method and calculations has been provided in section 7.2.2; the in depth analysis resulted in an estimated overall transfer function for the system.

$$G_c(s) = \frac{\frac{R_f}{R_1} K_m}{s^2 + s\alpha + \frac{R_f}{R_1} K_m}$$
$$G_c(s) = \frac{1075.8}{s^2 + 38.61s + 1075.8}$$

This is/isnt what we'd expect

After constructing the circuit shown in figure 12 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 3 times, this graphical representation of this data has been included below.

FIGURES

WHAT WE CAN SEE FROM THIS (AS VOLTAGE GOES UP, THIS GOES UP), and here is the closed loop response with experimentally found gain required to produce an overshoot of 5%.

FIGURES

WHAT WE CAN SEE FROM THIS

compare models
compare gains
new method

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

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, overshoot and settling time, for five different input frequencies; the following table was constructed.

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

Refer to section 7.2.3 for the construction of the table. Both the above table, and the systems response (figure 16) 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 were to continue increasing, eventually the operation amplifier will begin **decreasing/squishing/dropping voltage**.

The reason the frequency has no effect on these values, as explained in section 7.2.3, is that the circuit contains no energy storing devices (such as capacitors or inductors), and as such- **frequency should not affect any circuit properties**. Additionally, the approximated transfer function of this system is independent of the input frequency, or more simply;

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

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 17, it can be seen that increasing the gain resulted in an increased overshoot and a decreased settling time. This has been proven more in depth in the associated answers section, but a summary of the calculations can be found below.

SUB IN TO FIND OS and Ts AND SHOW THEY RELY ON GAIN, PROP/invPROP WHY IS THIS USEFUL - prop controller?

D4/D5

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.

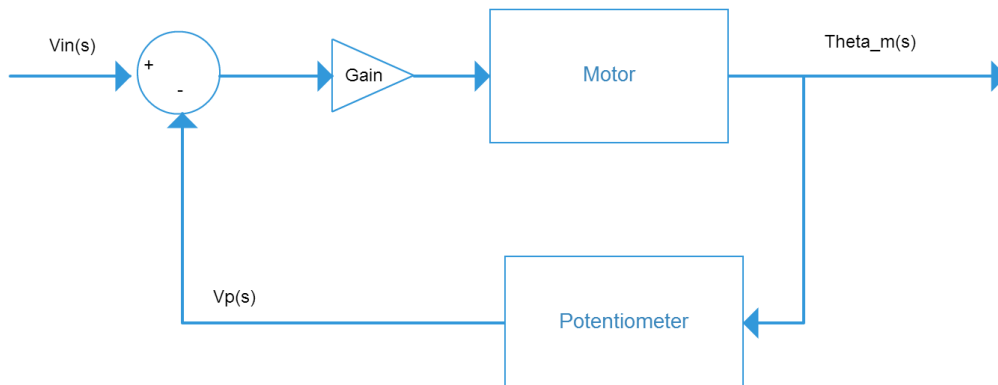


Figure 2: Updated Functional Diagram

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



Figure 3: Simulated Response

```
1 %% A3
2 Alpha = 1;
3 K_m = 1;
4
5 t = linspace(0,10,100);
6 G_0 = tf([K_m],[1 Alpha 0]);
7 figure();
8 step(G_0, t, 'r')
9 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

```
1 %% A4
2 Alpha = 1;
3 K_m = 1;
4 t = linspace(0,10,100);
5 G_0 = tf([K_m],[1 Alpha 0]);
6
7 load ENB301TestData_2015.mat
8 figure();
9 plot(t,y1);
10 hold;
11 step(G_0, t, 'r')
12 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

```

1 %% A6
2 Alpha = 1; %0.5-1.5
3 K_m = 1.3; %1-2
4 G_0 = tf([K_m],[1 Alpha 0]);
5 figure();
6 plot(t,y1);
7 hold;
8 step(G_0, t, 'r');
9 print('-depsc','a6a')

```



Figure 6: Inverting Amplifier System

```

1 %% Automated:
2 % loop values of alpha and k_m
3 % calculate RMS error each time
4 % pick values with the least error
5
6 y1_rms = rms(y1);
7 prevdiff = 10000;
8
9 for Alpha = 0.1:0.1:3
10     for K_m = 0.1:0.1:3
11         G_0 = tf([K_m],[1 Alpha 0]);
12         G_t = step(G_0, t, 'r');
13         G_t_rms = rms(G_t);
14         diff = y1_rms - G_t_rms;
15         if (abs(diff) < abs(prevdiff))
16             prevdiff = diff;
17             K_m_f = K_m;
18             Alpha_f = Alpha;
19         end
20     end
21 end
22
23 G_0 = tf([K_m_f],[1 Alpha_f 0]);
24 figure();
25 plot(t,y1);
26 hold;
27 step(G_0, t, 'r');
28 print('-depsc','a6b')
29
30 % Final values:
31 % Alpha_f = 1.3
32 % 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**

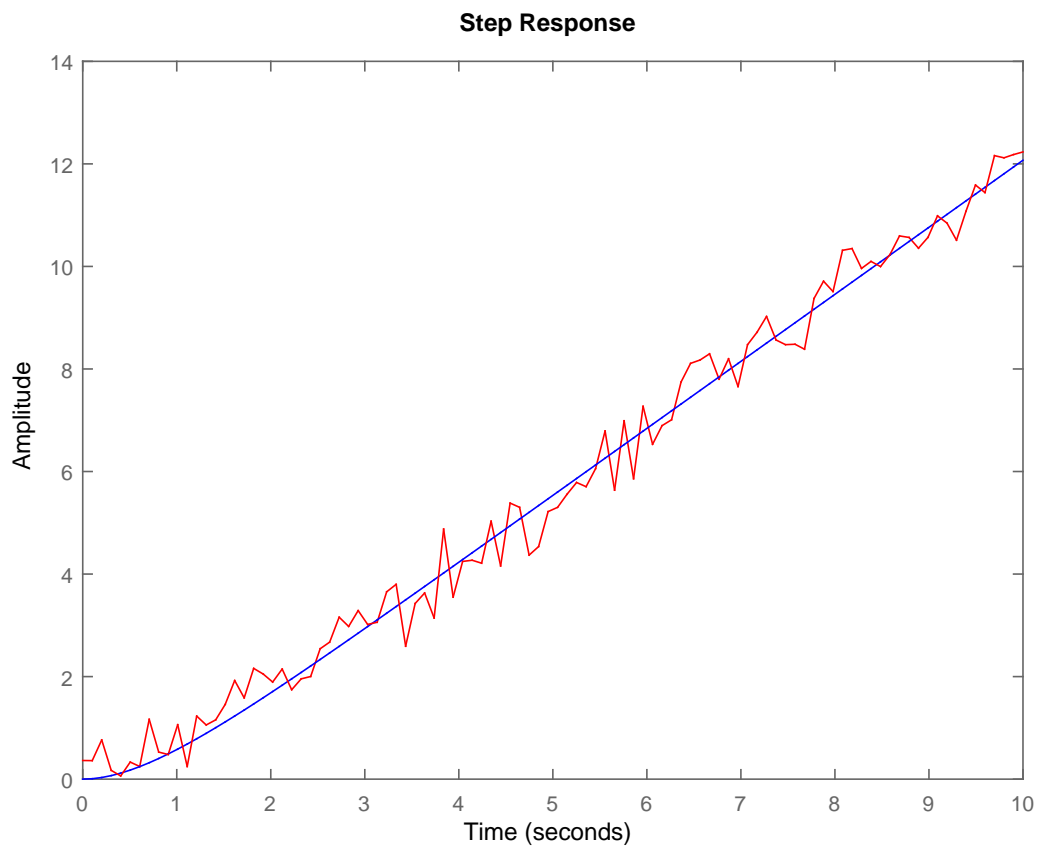


Figure 7: Estimated System Response + Uncertainty

```
1 %% A7
2 rand_power = 2;
3
4 y_1 = step(G_0, t, 'r');
5 y_n = y_1 + (rand_power/2)*rand(size(y_1,1),1) - ...
        (rand_power/2)*rand(size(y_1,1),1);
6
7 figure();
8 step(G_0, t, 'b');
9 hold;
10 plot(t,y_n,'r');
11 print('-depsc','a7')
```

7.1.2 Declan Gilmour

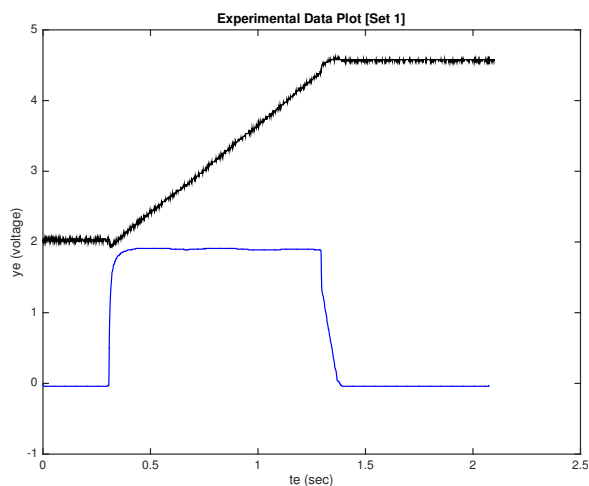
- 1.
- 2.
- 3.
- 4.

7.2 Answers:

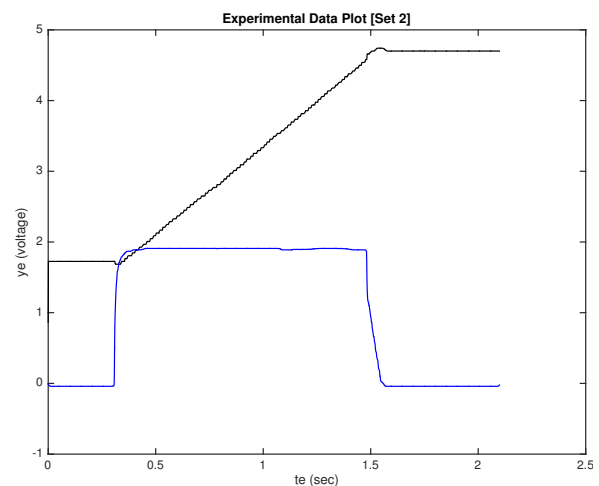
7.2.1 Part B

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

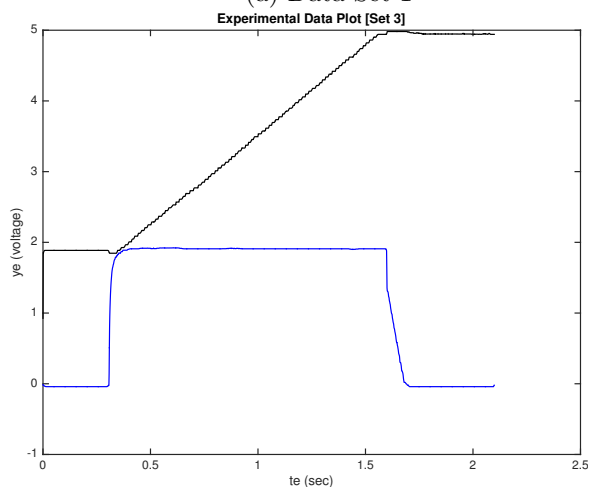
```
1 %% B1
2 % Load experimental data and plot in black
3 data = csvread('PartB.Test1.csv',2,0); % Read in test 1
4 te_1 = data(1:end,1); % Store te variable
5 te_1 = te_1 + abs(te_1(1)); % Move te variable to start at zero
6 ye_1 = data(1:end,2); % Store ye variable
7 ye_1step = data(1:end,3); % Store ye step input variable
8
9 figure
10 plot(te_1, ye_1, 'k', te_1, ye_1step, 'b') % Plot ye in black and ye in blue
11 title('Experimental Data Plot [Set 1]')
12 xlabel('te (sec)')
13 ylabel('ye (voltage)')
14 print('-depsc', strcat('figures', filesep, 'B1.dataset1')); % Store figure
15 close
```



(a) Data Set 1



(b) Data Set 2



(c) Data Set 3

Figure 8: Open Loop response

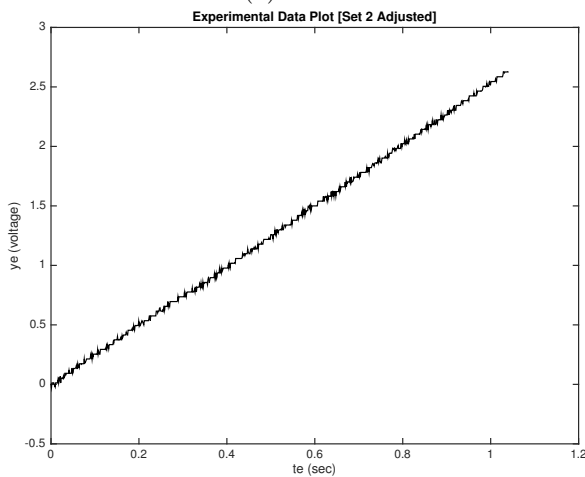
2. Modify the MATLAB script created in the pre-labs to simultaneously plot both the experimental data and $y_1(t)$. No real idea what to put here...
3. Derive a figure of merit for the estimation compared with experimental results.
Mean Square error calculation: $error = (G_0 - y_e)^2$
Root Mean Square 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

```

1  % Plot experimental data ye against systems estimated tf y1
2  [te_1new, ye_1new] = timing_fix(te_1, ye_1);
3  [te_2new, ye_2new] = timing_fix(te_2, ye_2);
4  [te_3new, ye_3new] = timing_fix(te_3, ye_3);
5
6  figure
7  plot(te_1new, ye_1new, 'k')
8  title('Experimental Data Plot [Set 1 Adjusted]')
9  xlabel('te (sec)')
10 ylabel('ye (voltage)')
11 print('-depsc', strcat('figures', filesep, 'B2_dataset1'));
12 close
13
14 figure
15 plot(te_1new, ye_1new, 'k')

```

```

16 title('Experimental Data Plot [Set 1 Adjusted]')
17 xlabel('te (sec)')
18 ylabel('ye (voltage)')
19 print('-depsec',strcat('figures',filesep,'B2-dataset1'));
20 close

```

```

1 % Prepare experimental data for alpha and km parameter calculations
2 [te_1trim, ye_1trim] = trimForCalculation(te_1new, ye_1new, mf1);
3 [te_2trim, ye_2trim] = trimForCalculation(te_2new, ye_2new, mf1);
4 [te_3trim, ye_3trim] = trimForCalculation(te_3new, ye_3new, mf1);
5
6 km_num = 500; % number of km values used
7 km_max = 500; % max km value used
8 alpha_num = 500; % number of alpha values used
9 alpha_max = 500; % max alpha value used

```

```

1 % Preallocate size for speed
2 output_ms = zeros(3, km_num*alpha_num);
3 output_rms = zeros(3, km_num*alpha_num);
4 km_ms = zeros(1, 3);
5 alpha_ms = zeros(1, 3);
6 km_rms = zeros(1, 3);
7 alpha_rms = zeros(1, 3);
8
9 for iteration = 1 : 3
10
11     % Set te and ye based on iteration
12     if (iteration == 1)
13         te = te_1trim;
14         ye = ye_1trim;
15     elseif (iteration == 2)
16         te = te_2trim;
17         ye = ye_2trim;
18     else
19         te = te_3trim;
20         ye = ye_3trim;
21     end
22
23     % Set cycle variables
24     error_ms = 0;
25     error_rms = 0;
26     ii = 1;
27     count = 0;
28
29     for km = linspace(0, km_max, km_num) % Cycle km values
30         for alpha = linspace(0, alpha_max, alpha_num) % Cycle alpha values
31             G = tf(km, [1 alpha 0]);
32             G_0 = step(G, te);
33
34             % Calculate error
35             for jj = 1 : length(te)
36                 % Calculate mean square error
37                 error_ms = error_ms + (G_0(jj) - ye(jj))^2;
38
39                 % Calculate root mean square error
40                 error_rms = error_rms + rms(G_0(jj) - ye(jj));
41             end
42
43             % Store km, alpha and the error taken to calculate
44             output_ms(:, ii) = [km; alpha; error_ms];
45             output_rms(:, ii) = [km; alpha; error_rms];
46
47             % Reset cycle variables
48             ii = ii + 1;

```

```

49         error_ms = 0;
50         error_rms = 0;
51     end
52
53     % Output km iterations
54     %count = count + 1;
55     %fprintf('%d %s %d\n',count,'/',km_num);
56 end
57
58 % Calculate km and alpha values for mean square error calculation
59 [~,index] = min(output_ms(3,:));
60 km_ms(iteration) = output_ms(1,index); % Output variable
61 alpha_ms(iteration) = output_ms(2,index); % Output variable
62
63 % Calculate km and alpha values for root mean square error calculation
64 [~,index] = min(output_rms(3,:));
65 km_rms(iteration) = output_rms(1,index); % Output variable
66 alpha_rms(iteration) = output_rms(2,index); % Output variable
67 end
68 km_mean = (mean(km_ms) + mean(km_rms)) / 2;
69 alpha_mean = (mean(alpha_ms) + mean(alpha_rms)) / 2;

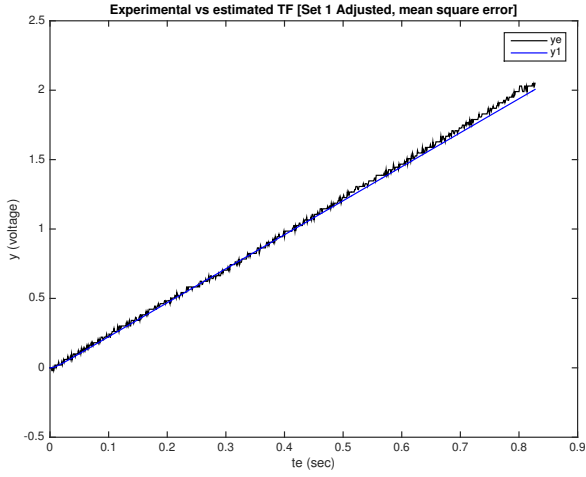
```

```

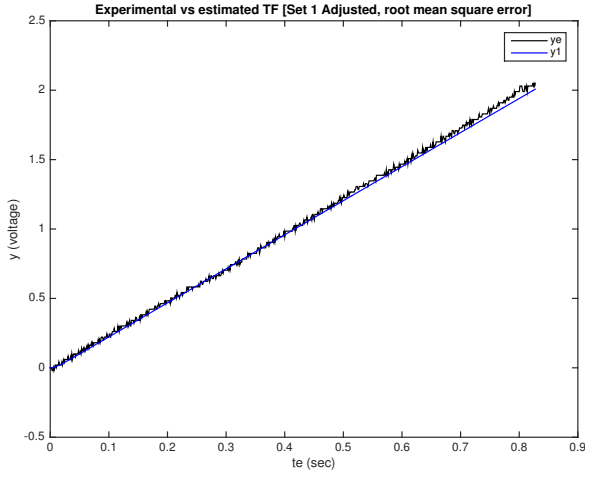
1 % Plot y1 against ye
2 % Data Set 1
3 G = tf(km_ms(1), [1 alpha_ms(1) 0]);
4 y1_ms = step(G,te_lnew);
5 figure
6 plot(te_lnew,medfilt1(ye_lnew,1),'k',te_lnew,medfilt1(y1_ms,1),'b')
7 title('Experimental vs estimated TF [Set 1 Adjusted, mean square error]')
8 xlabel('te (sec)')
9 ylabel('y (voltage)')
10 legend('ye','y1')
11 print('-depsc',strcat('figures',filesep,'y1-dataset1_ms'));
12 close

```

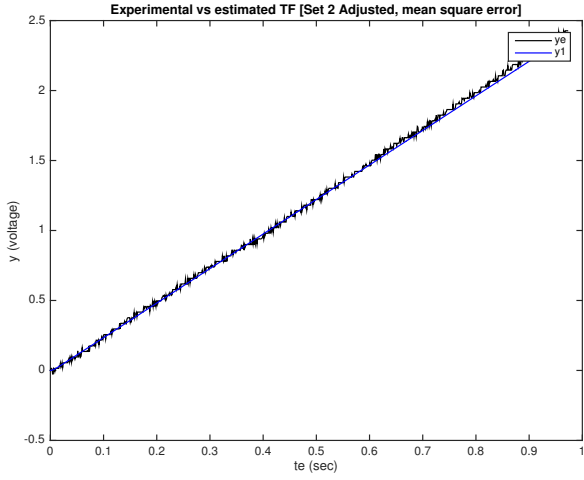
The average constants found are as follows: $\alpha = 170.3407$ $k_m = 422.0107$



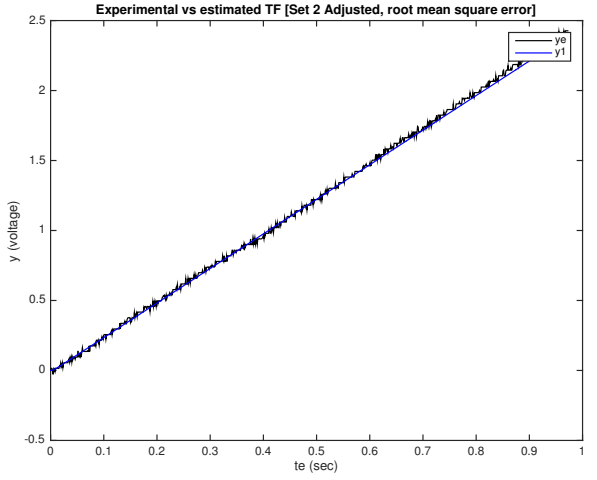
(a) Data Set 1 (ms)



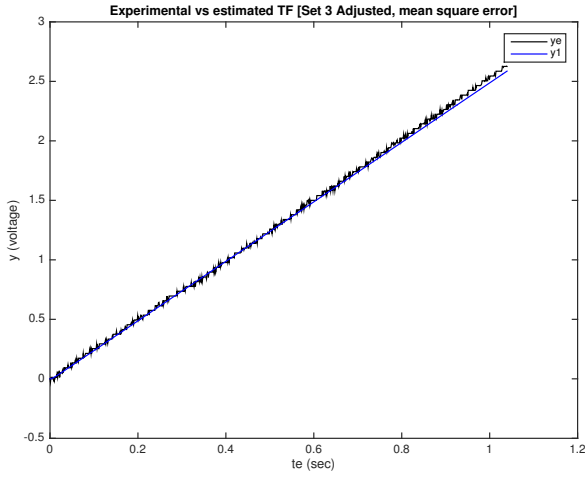
(b) Data Set 1 (rms)



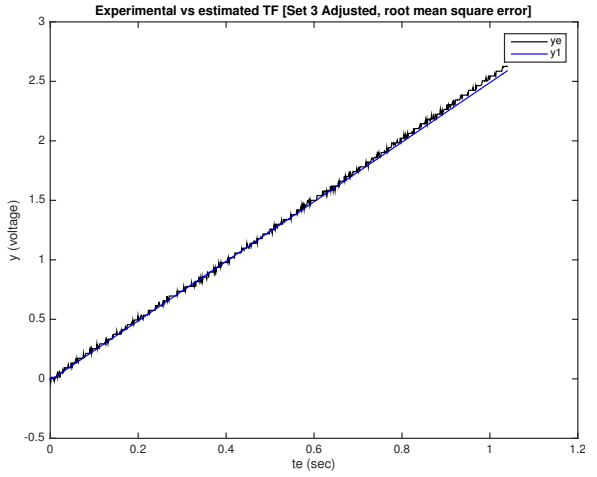
(c) Data Set 2 (ms)



(d) Data Set 2 (rms)



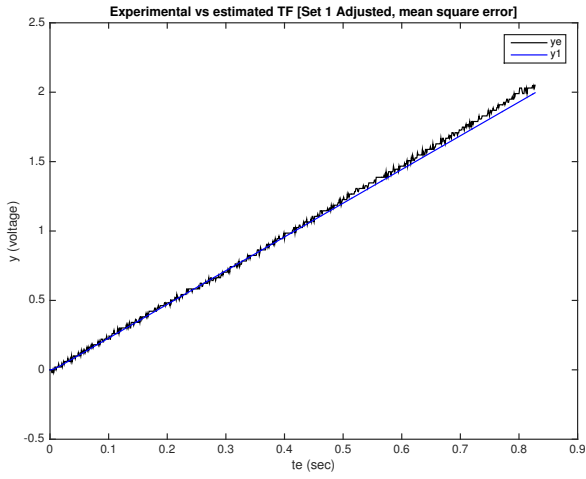
(e) Data Set 3 (ms)



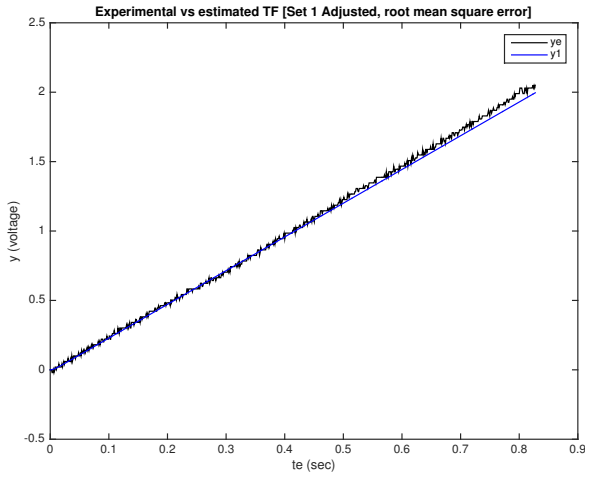
(f) Data Set 3 (rms)

Figure 10: Servo Motor Response Modeled

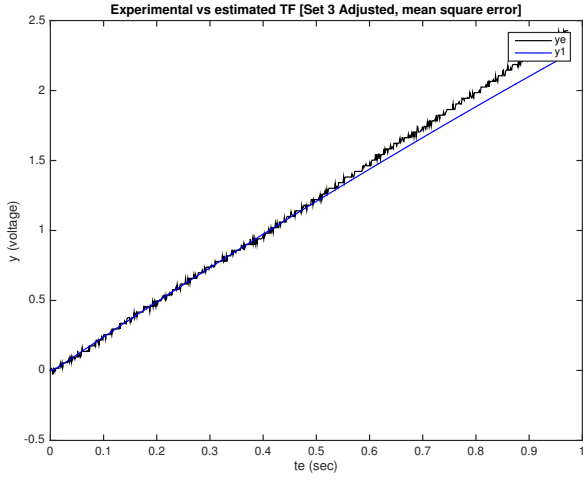
5. The mechanical constant α was found to be $\alpha = \alpha_{mean}$. Using the second order equation $G(s) = \frac{K_m}{(s+\alpha)}$ new values of gain constant k_m and electrical constant β . Values were found for $y_1(t) \approx y_2(t)$. The average constants found are as follows: $\beta = 0.0646$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



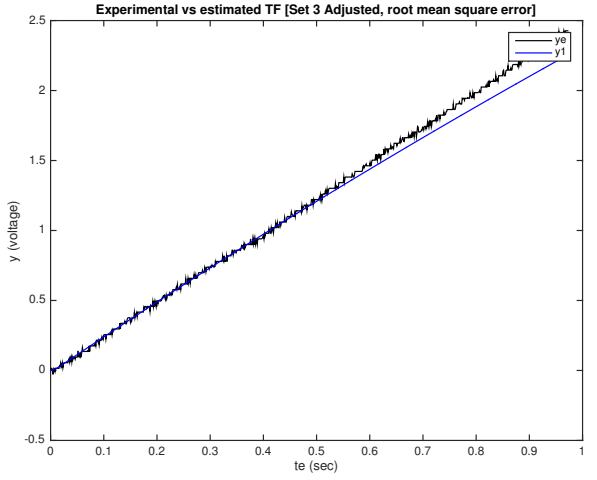
(a) Data Set 1 (ms)



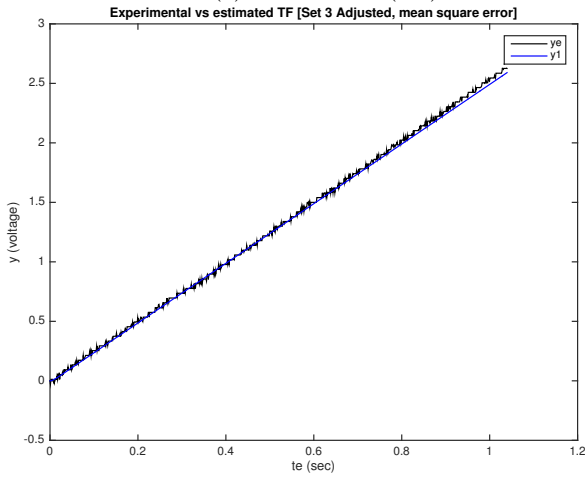
(b) Data Set 1 (rms)



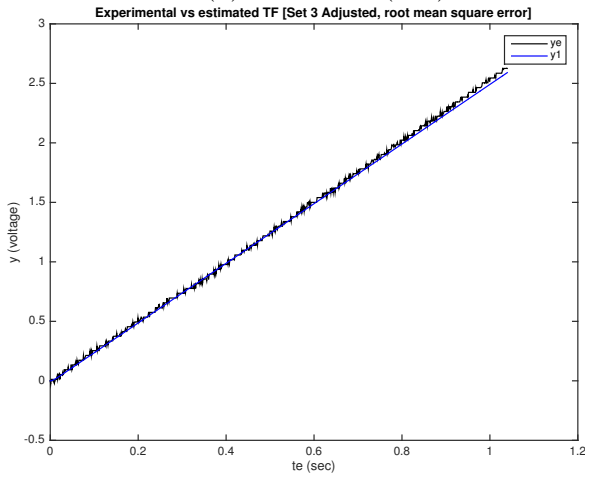
(c) Data Set 2 (ms)



(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

```

1 km_num = 500;      % number of km values used
2 km_max = 500;      % max km value used
3 beta_num = 50;     % number of alpha values used
4 beta_max = 1;      % max alpha value used
5
6 % Preallocate size for speed
7 output_ms = zeros(3,km_num*beta_num);
8 output_rms = zeros(3,km_num*beta_num);
9 km2_ms = zeros(1,3);
10 beta_ms = zeros(1,3);
11 km2_rms = zeros(1,3);
12 beta_rms = zeros(1,3);
13
14 for iteration = 1:3
15     if(iteration == 1)
16         te = te_1trim;
17         ye = ye_1trim;
18     elseif(iteration==2)
19         te = te_2trim;
20         ye = ye_2trim;
21     else
22         te = te_3trim;
23         ye = ye_3trim;
24     end
25
26     % Set cycle variables
27     error_ms = 0;
28     error_rms = 0;
29     ii = 1;
30     count = 0;
31
32     for km = linspace(0,km_max,km_num) % Cycle km values
33         for beta = linspace(0,beta_max,beta_num) % Cycle alpha values
34             G = tf(km, [1 (alpha_mean+beta) (alpha_mean*beta)]);
35             G_0 = step(G,te);
36
37             % Calculate error
38             for jj = 1 : length(te)
39                 % Calculate mean square error
40                 error_ms = error_ms + (G_0(jj) - ye(jj))^2;
41
42                 % Calculate root mean square error
43                 error_rms = error_rms + rms(G_0(jj) - ye(jj));
44             end
45
46             % Store km, alpha and the error taken to calculate
47             output_ms(:,ii) = [km;beta;error_ms];
48             output_rms(:,ii) = [km;beta;error_rms];
49
50             % Reset cycle variables
51             ii = ii + 1;
52             error_ms = 0;
53             error_rms = 0;
54         end
55
56         % Output km iterations
57         count = count + 1;
58         fprintf('%d %s %d\n',count,'/',km_num);
59     end
60
61     % Calculate km and alpha values for mean square error calculation
62     [~,index] = min(output_ms(3,:));
63     km2_ms(iteration) = output_ms(1,index); % Output variable
64     beta_ms(iteration) = output_ms(2,index); % Output variable
65
66     % Calculate km and alpha values for root mean square error calculation

```

```

67     [~,index] = min(output_rms(3,:));
68     km2_rms(iteration) = output_rms(1,index); % Output variable
69     beta_rms(iteration) = output_rms(2,index); % Output variable
70 end
71
72 km_mean2 = (mean(km_ms) + mean(km2_rms)) / 2;
73 beta_mean = (mean(beta_ms) + mean(beta_rms)) / 2;

```

```

1 % Plot y2 against ye
2 % Data Set 1
3 G = tf(km2_ms(1), [1 (alpha_mean+beta_ms(1)) (alpha_mean*beta_ms(1))]);
4 y2_ms = step(G,te_1new);
5 figure
6 plot(te_1new,medfilt1(ye_1new,1),'k',te_1new,medfilt1(y2_ms,1),'b')
7 title('Experimental vs estimated TF [Set 1 Adjusted, mean square error]')
8 xlabel('te (sec)')
9 ylabel('y (voltage)')
10 legend('ye','y1')
11 print('-depsc',strcat('figures',filesep,'y2_dataset1.ms'));
12 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

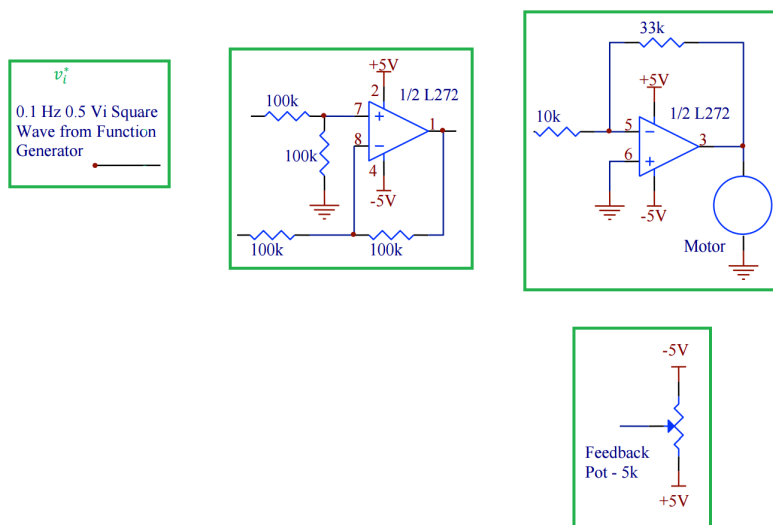


Figure 13: Closed Loop Motor Control System

- 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:

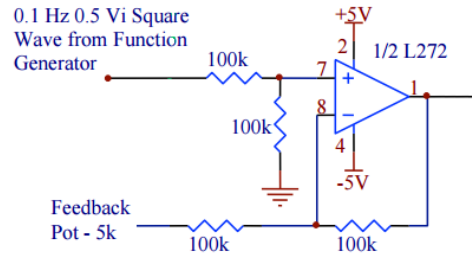


Figure 14: Difference Op-amp System

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

Inverting Op-amp:

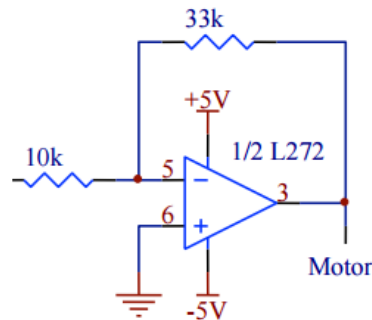


Figure 15: Inverting Amplifier System

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

Motor and pot:

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

something

3. NONIDEAL:

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

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

$$\frac{R_2}{R_1} V_1(s) = \frac{s(s + \alpha)}{K_m} V_p(s)$$

$$V_1(s) = \frac{R_1 s(s + \alpha)}{R_2 K_m} V_p(s)$$

$$V_1 =$$

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,

$$\alpha = 2\zeta W_n$$

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

$$W_n^2 = k_m \frac{R_f}{R_1}$$

$$\frac{R_f}{R_1} = W_n^2 / k_m$$

$$K = \left(\frac{\alpha}{2\zeta}\right)^2 / k_m$$

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

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

$$K = 2.4$$

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% overshoot 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?**.

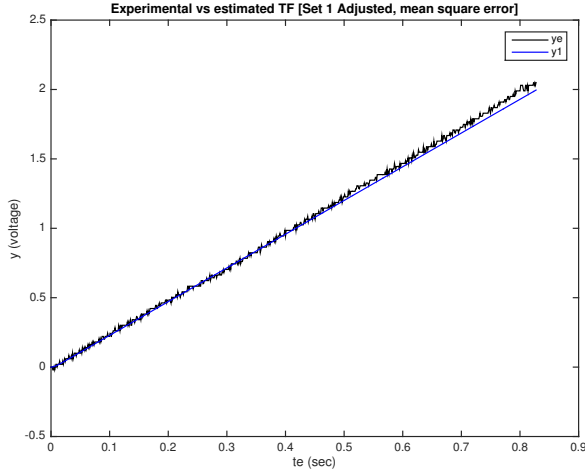
Drawbacks/application?

Add the actual OS from prac/theo

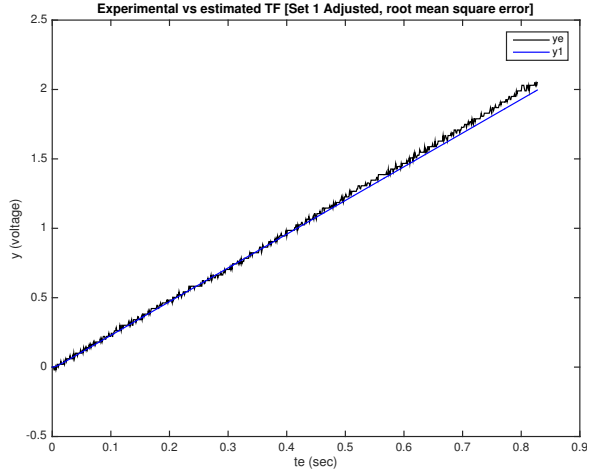
8. Another method of calculating TF? (bode)

7.2.3 Part D

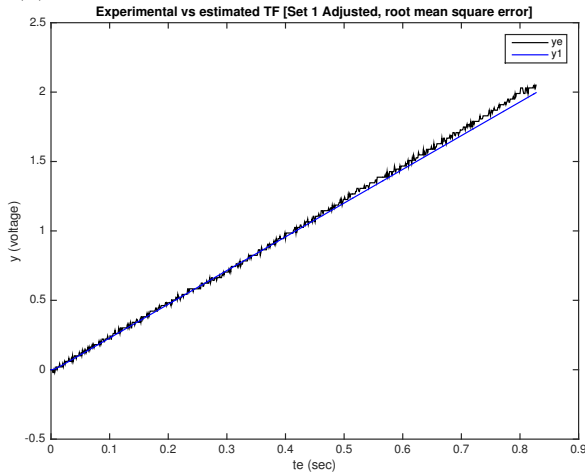
1. stuff
2. The overshoot and settling time were measured over five different input frequencies in order to determine the effect over varying the input signals frequency. The systems response to these signals was also recorded, and can be found as a set of graphical time domain plots below.



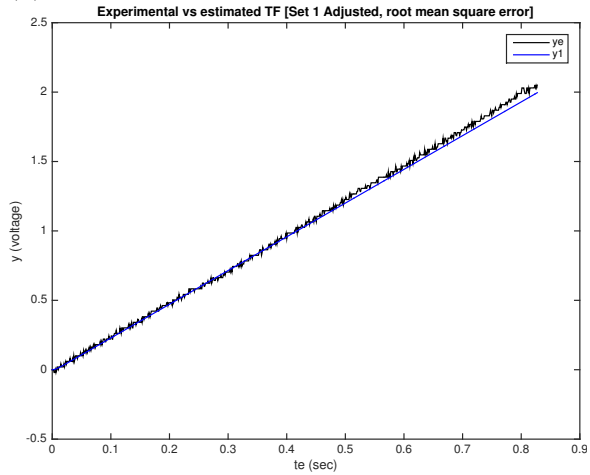
(a) Closed loop time domain response to 0.5Hz input.



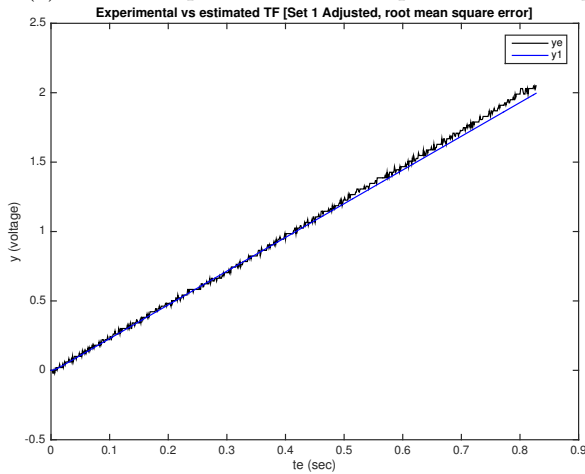
(b) Closed loop time domain response to 0.75Hz input.



(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 16: Systems response to various input frequencies.

The peak overshoot and **settling time** were measured using the cursors **stuuff**

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

$$OS(0.5Hz) = \frac{525 - 495}{495} = 6.1$$

$$OS(0.75Hz) = \frac{523 - 495}{495} = 5.7$$

$$OS(1Hz) = \frac{525 - 495}{495} = 6.1$$

$$OS(1.25Hz) = \frac{525 - 495}{495} = 6.1$$

$$OS(1.5Hz) = \frac{524 - 495}{495} = 5.8$$

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

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

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. Moreover, once the time elapsed per input cycle is reduced below the settling time, **the steady state response of the system is lost, and only the transient response has enough time to be viewable.**

Eventually, as the frequency increases, the output doesn't have enough time to settle. It is also evident from the previous figures, that neither the overshoot, settling time or the steady state error is effected; until eventually the system does not have enough time to reach a steady state value.

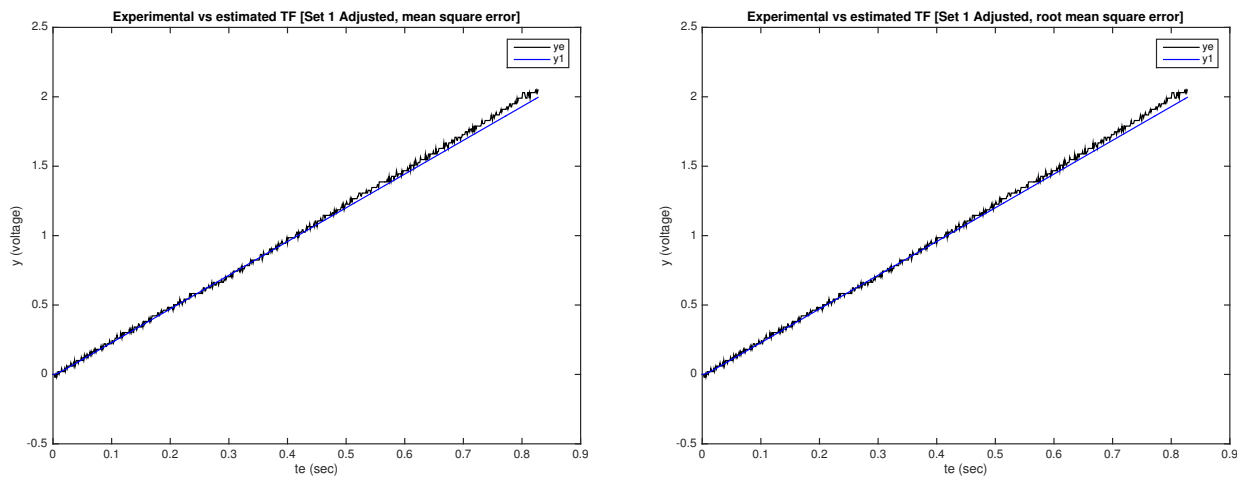
Freq only affects time available to find a steady state value.

Eventually, there op-amps physical components would not be able to switch fast enough, and **WHAT** would occur.

This occurs because **why**

Frequency doesn't affect zeta and omega, so settling time and OS should stay the same. no energy storing components in the circuit.

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



- (a) Closed loop time domain response to 0.5Hz input. (b) Closed loop time domain response to 0.75Hz input.

Figure 17: Systems response to various gain values.

As you would expect, changing the gain value **increases OS?** and **decreases Ts?**. The above images show large evidence to support this comment; as the correlation between these values can be seen clearly.

This is useful because **WHY**

4. **C4 - Matlab - Declan?** simulink
5. **C4 - Matlab - Declan?** PD/PI/PID controller

8 SHIT

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