TEXAS A&M UNIVERSITY

MEEN 364 *Spring 2018*

Assignment: Lab 10 and 11 Report
Instructor: Dr. Prabhakar Pagilla
Section: 501
Submission Date: 7 May 2018

All authors have contributed to the preparation of the report and have read the final version of the report.

On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work.

Name of Authors	**Signature**
Andres Contreras	
Jacob Hartzer	
Perry Inhofe	
Colin Michels	

I. Abstract

Labs 10 and 11 revolve around the design and implementation of a position controller for a DC motor. In Lab 10, a proportional-derivative (PD) controller was created to meet the desired specifications of a rise time of 10% of the time constant of the DC motor measured in Lab 4 in addition to a maximum percent overshoot of 10%. The original gain values used in the PD controller were $1.2059*10^4$ and 33.1504 for K_p and K_d. This controller was then simulated in both MATLAB and Simulink in iterations until the specifications were met in both programs. The resulting gain values that were determined were K_r = 12000 and K_d= 58. The an even more rigorous DC motor model, these gain values were changed even more via iteration to K_p = 18000 and K_d = 1000. In Lab 11, this controller was then implemented into a DC motor in order to power it and turn it by a desired angle. While the system was able to reach the desired angle with a correct amount of overshoot, due to power limitations, the rise time could never be adjusted below the maximum allowed value of 0.0145. That being said, it was determined that this controller could be used to accurately change the motors position without much overshoot, with each of the gain values sets resulting in phase margins for small angles of 39.33°, 34.15°, and 80.30° and phase margins for large angles of 29.12°, 27.21°, and 64.93° respectively as the sets were presented here.

II. Introduction

In Labs 10 and 11, a proportional-derivative controller was designed in order to meet given desired performance specifications. In general, a control problem, such as the one to be solved in this experiment, can be broken down into the following steps. First, in order to define the control problem, a set of performance specifications are obtained. Next, the system is modelled mathematically. If necessary due to the system being nonlinear or overly complicated, the model may be simplified or linearized. Then, the controller is designed to meet the desired performance specifications. The controller is tested first on the simplified model and then on the complicated model and adjusted if needed. Finally, the new controller is implemented on the real system. The objectives of these labs were to design and simulate a controller to meet given performance specifications as well as to implement that controller on the real system. In the case of this experiment, the desired performance specifications were a rise time of less than 10% of the time constant of the DC motor and a maximum percent overshoot of less than 10%. The concepts explored in this experiment have wide ranging importance. From heating systems to autopilot altitude-hold, control systems are important to almost all modern technology. This makes control systems a vital topic of knowledge for all engineers.

III. Theory

These labs delve into the idea of developing and implementing a controller for a DC motor. The motor is controlled via an input voltage present in the diagram below. This input voltage was actually what the controller begin designed in these two labs was affecting. As the voltage is increased, the motor turns until it's built in potentiometer communicates with the controller that the desired angle has been reached. Once this happens, the input voltage was quickly changed to a value that makes the angular speed of the motor equal zero, thereby fixing the motor at the given angle and preventing it from being turned any farther. If the motor is rotated in either direction, the voltage will increase or decrease respectively in order to turn the motor back to the desired position.

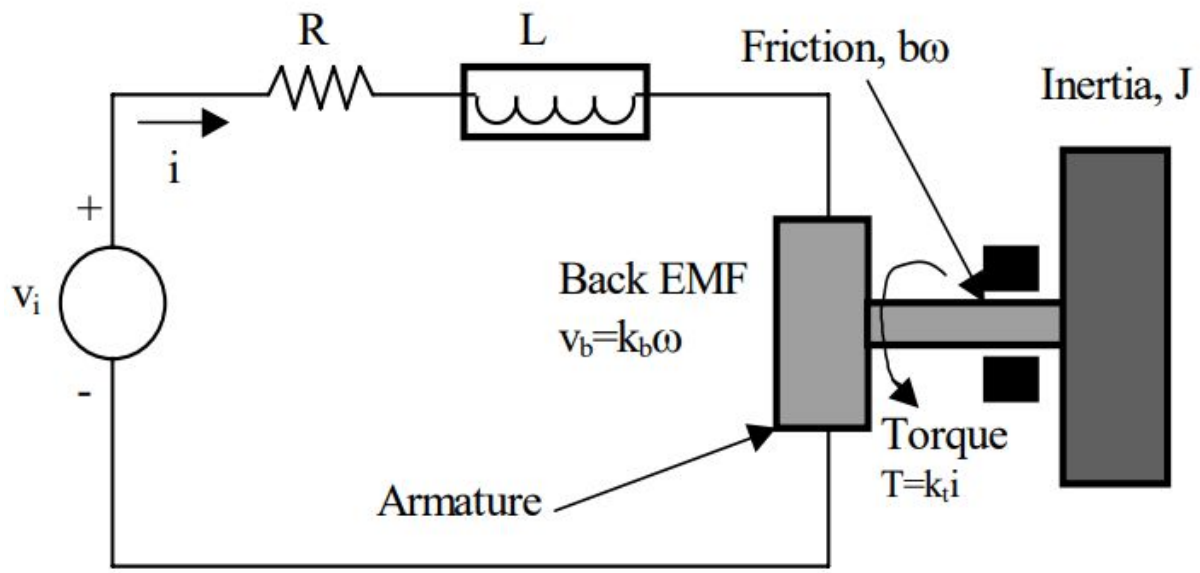

Figure 1. System model[1]

In order to understand how this controller works, one must understand how a PD controller works, especially with respect to how its output is proportional to both the error and the derivative of the error and why it is needed over a proportional controller. With just a proportional (P) controller, as seen below, rise time will increase as the percent overshoot is decreased and vice versa. Therefore, it will be impossible for one to reach the requirements of this system with just being able to change a K_p.

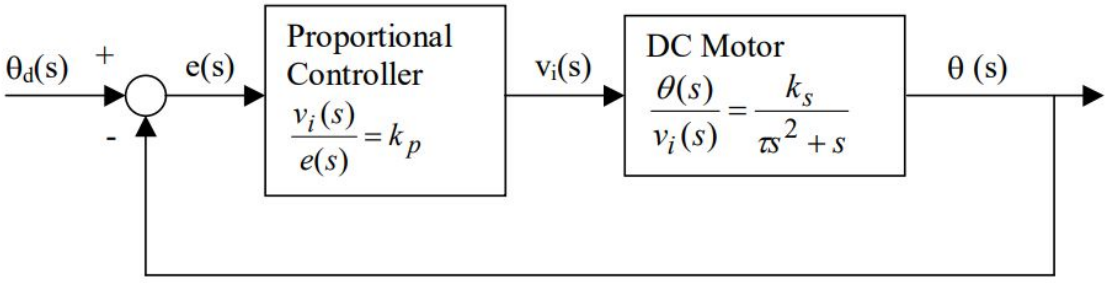

Figure 2. Proportional Controller[1]

In a PD controller on the other hand, as seen below, K_s and K_p can both be manipulated independently. As a result, the user is able to have the simulation reach requirements for both

the maximum overshoot and maximum rise time. Therefore, a PD controller can be used as the controller for this system.

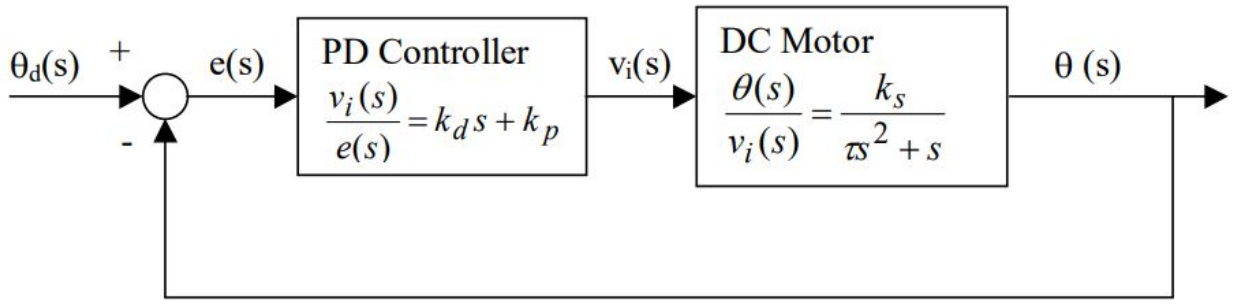

Figure 3. PD Controller[1]

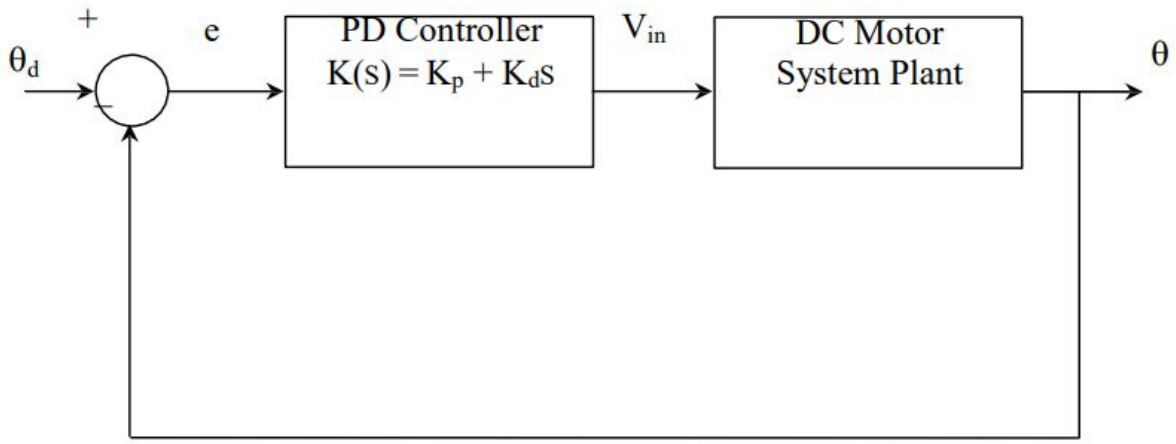

Figure 4. Closed-Loop block diagram[1]

Once this controller has been selected, a preliminary set of data via a given code and a given set of data seen in *Table 1* and *Lab10.m* in Appendix A. With this data, the following equation, derived in Appendix C, can be used to further refine the gain values. This equation more rigorously models the DC motor and allows for a better fit once the controller is applied to the motor in Lab 11.

Table 1. Given rigorous system constants

Parameter	Value
K_t	0.007 *Nm / A*
K_b	0.067 *Vs / rad*
R	1.64 Ω
L	3.0 *mH*
B	$44*10^{-3}$ *Nms / rad*
J	0.03 Kgm^2

$$\frac{LJ}{K_t}\frac{d^2\omega}{dt^2} + \left(\frac{LB+RJ}{K_t}\right)\frac{d\omega}{dt} + \left(\frac{RB}{K_t} + K_b\right)\omega = V_i \qquad (1)^{[1]}$$

IV. Procedure

In Lab 10, the objectives were to design a controller to meet the given specifications and simulate the design for the system model. In order to design the PD controller to accomplish the objectives, the following procedural steps were completed. First, the given time domain specifications of a rise time of 10% of the time constant and a maximum percent overshoot of 10% were converted to the frequency domain specifications of phase margin and gain margin. Next, the plant transfer function was defined and values for the gains K_p and K_d were estimated. Using the plant transfer function, the loop transfer function was derived and the bode plot was generated using MATLAB software. This plot showed the phase margin and the gain margin. If the frequency domain specifications were not met by these values, new values of K_p and K_d were selected until the specifications were met. With the optimal values of K_p and K_d obtained, MATLAB was used to simulated the closed loop system. The characteristics of the response were determined and compared to the desired specifications. The system was simulated again with Simulink. After comparing the response to the desired specifications, the control gains were modified to best meet those specifications. Finally, a more rigorous model of the DC motor was used to derive a loop transfer function. The response of the closed loop system using the

controller design at the beginning of Lab 10 and the loop transfer function that just derived was simulated and plotted in both MATLAB and Simulink.

In Lab 11, the controller designed in Lab 10 was implemented using Simulink. The input for the system was the potentiometer voltage and the output was the voltage to the motor. The potentiometer constant, K_x, from Lab 4 was used to convert the potentiometer voltage into the angular position of the motor. The angular position of the motor was displayed using a numerical indicator and a waveform chart and logged using an angular position vs. time graph with a step size of 0.001 seconds. The controller was implemented to control the system from 0 degrees to two desired motor positions of 60 degrees and 120 degrees. Three sets of control gains were implemented for both of the desired motor positions. These were the control gains that best met specifications for the original MATLAB and Simulink simulations, those that best met specifications at the 60 degree position, and those that best meet specifications at the 120 degree position. The control gains for the 60 and 120 degree positions were found through hardware iteration. The responses of each trial were obtained, and the rise time and percent overshoot for each set of control gains were recorded in *Table 2*. The controller was implemented using radians as the unit of angular position by converting the input to radians and the output back to degrees.

V. Results and Discussion

Lab 10

The system parameters set in Lab 10 specifies a rise time equalling 10% of the time constant and a max percent overshoot of 10%. The code *Lab10.m* estimated the K_p and K_d values that would be required in the implementation of a PD controller to meet these system specifications. These values were $1.2059*10^4$ and 33.1504 for K_p and K_d, respectively. The bode plot for the estimated parameters is shown in *Figure 5*.

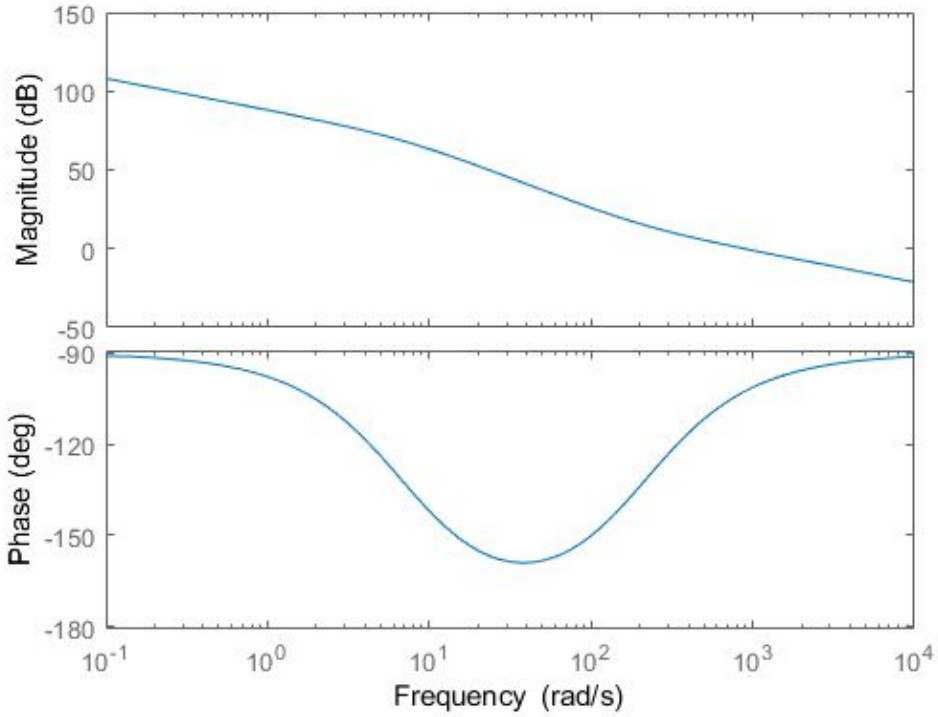

Figure 5. Bode plot of estimated parameters

The MatLab and Simulink simulations showed that the estimated parameters would not satisfy the given system requirements. Therefore, the values were iteratively adjusted for the given model until the requirements were satisfied. *Figure 6* illustrates the responses of the simulations when the system parameters were optimized using the standard model. These values were K_r = 12000 and K_d= 58. Using *Equation 1*, a more rigorous DC motor model was derived (shown in Appendix C). Using this model, the control parameters were again iteratively adjusted to fulfil the given system requirements. *Figure 7* illustrates the responses of the simulations using the rigorous model. These control parameters were K_p = 18000 and K_d = 1000.

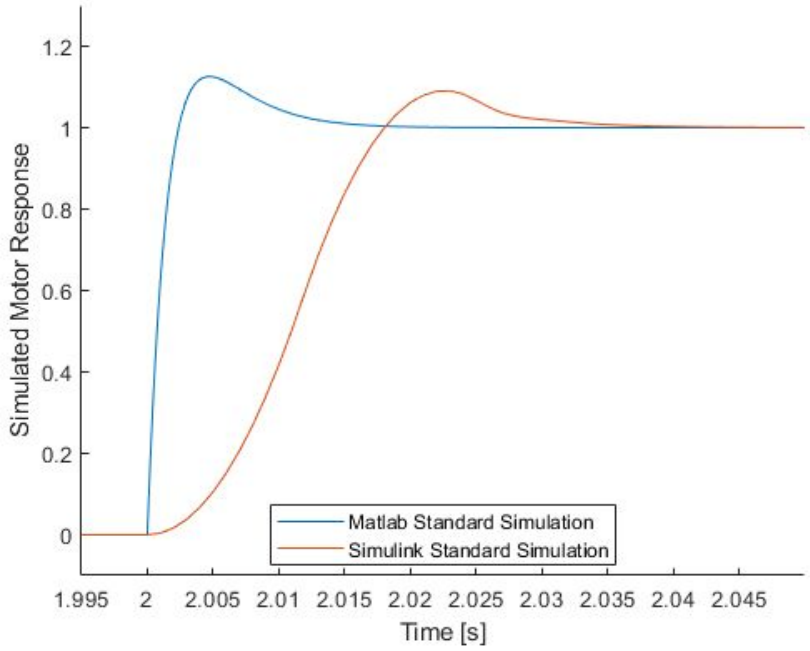

Figure 6. Simulations of optimized parameters using standard DC model

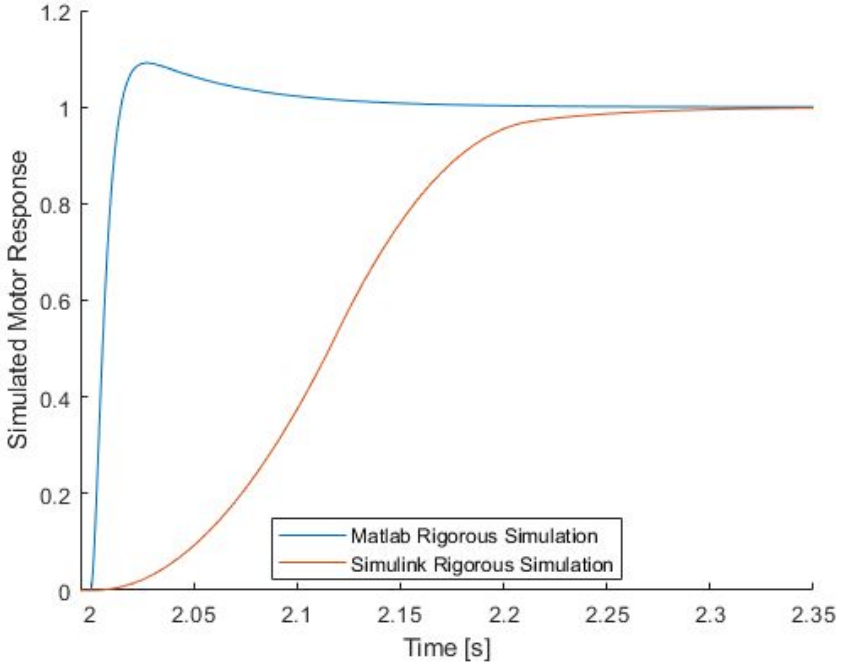

Figure 7. Simulations of optimized parameters using rigorous DC model

Because the rigorous model more accurately represents the rotational inertia of the system, the original and rigorous models did not have the same responses. The more rigorous model had longer rise time but lower overshoot. This is due to the higher order system modeling greater resistance to acceleration. Additionally, it should be noted that the MatLab simulation had much faster rise times on both models. This is due to the lack of power limitation in the MatLab simulation. As shown in Appendix B, the simulink code used a saturation block which limited the voltage output to the motor to be between -10 and 10 V. Therefore, a faster response is not entirely possible with this limitation. Because the MatLab simulation did not have such a control, it showed faster response times.

Lab 11

In order to implement the DC motor controller, the potentiometer was calibrated with the results summarized in *Figure 14* of Appendix D. Using the PD controllers design in the last lab, new Simulink code was designed to control the DC motor. Using the original estimated parameters, the standard model optimized variables, and the rigorous optimized variables, the DC motor was set to 60 and 120 degrees with the responses recorded. The results for the 60 degree runs are shown in *Figure 8* while the results for the 120 degree runs are shown in *Figure 9*. Using the code *analysis.m*, the performances of each trial was calculated and these data are summarized in *Table 2*.

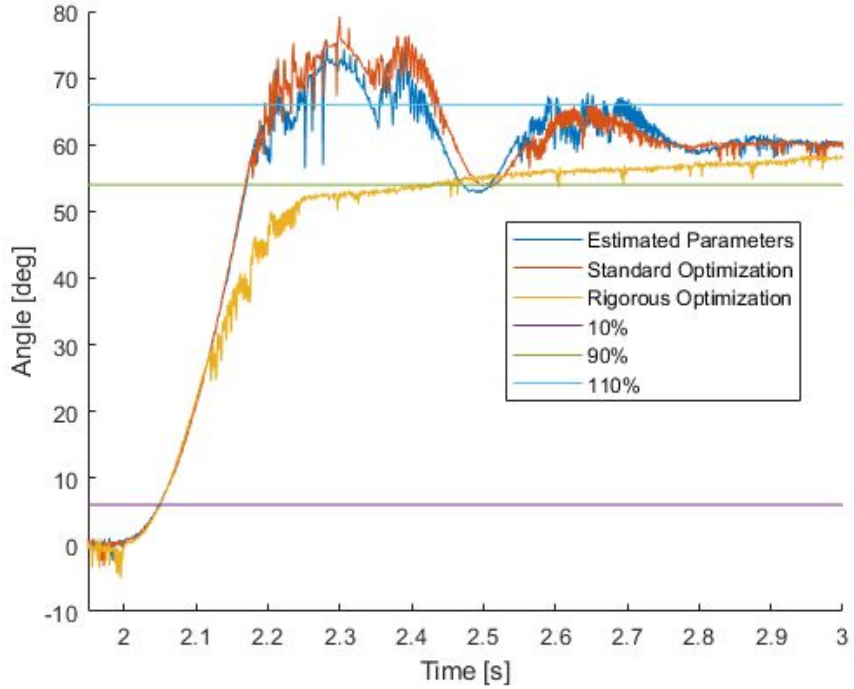

Figure 8. Control of DC motor to small angle using various parameters

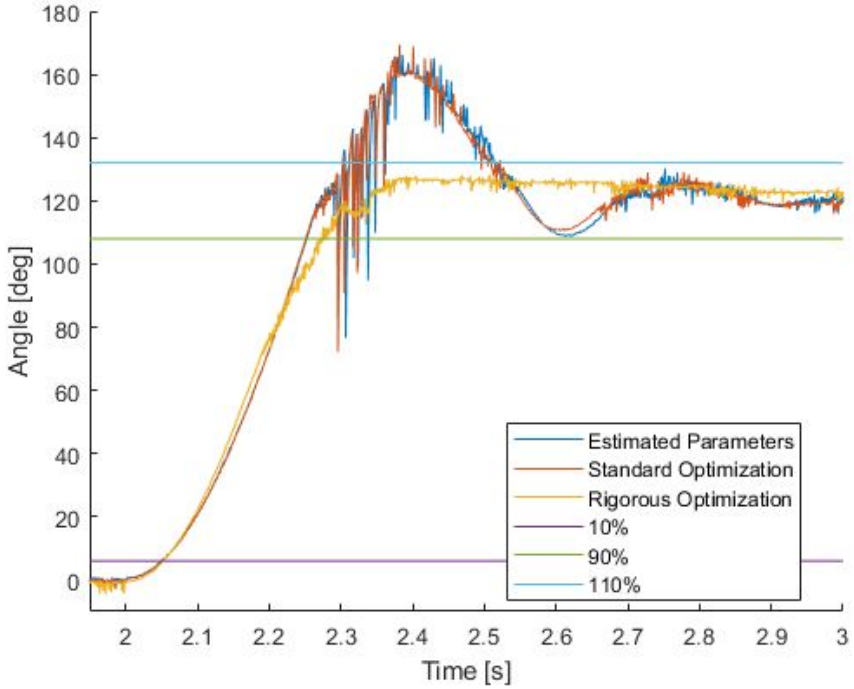

Figure 9. Control of DC motor to large angle using various parameters

Table 2. Performance parameters of each DC motor control trial

Run	Kp [V/rad]	Kd [Vs/rad]	θd [rad]	θss [rad]	θd-θss [rad]	Tr [s]	PO %	Zeta	PM [deg]
60 est	12059	33.15	1.047	1.045	0.0020	0.1220	26.09%	0.39	39.33
60 std	12000	58	1.047	1.047	0.0007	0.1200	31.93%	0.34	34.15
60 rig	18000	1000	1.047	1.049	-0.0020	0.3580	1.45%	0.80	80.30
120 est	12059	33.15	2.094	2.082	0.0123	0.1780	38.43%	0.29	29.12
120 std	12000	58	2.094	2.077	0.0173	0.1770	41.13%	0.27	27.21
120 rig	18000	1000	2.094	2.090	0.0047	0.1990	6.84%	0.65	64.93

To reiterate, the goal of this implementation was to find a set of PD controller gains that would allow the controller to position the motor to a set angle within 10% of the time constant calculated in lab 4 with less than 10% overshoot. This means the goal rise time was 0.01415s, which was not met in any of the sets of parameters. However, this is best explained by the power limitations in the control system. Were it possible to apply a much higher voltage to the motor, then it would be possible to position the motor that quickly. As is shown by the results of the simulations and the implementations, the rigorous model more closely matches the real world implementation of the DC motor control. Using the equations outlined in Appendix C, the phase margins for each control parameter set were calculated. Based on the measured overshoot, the phase margins for the small angle cases were 39.33°, 34.15°, and 80.30° respectively for the estimated parameters, the standard model optimized parameters, and the rigorous model optimized parameters. Similarly, the values for the large angle cases were 29.12°, 27.21°, and 64.93° where only the rigorous model optimized parameters met the phase specifications.

VI. Conclusions

In conclusion, a PD controller was designed in order to meet the set performance parameters of rise time being less than 10% of the motor time constant with less than 10% overshoot. This was done by deriving standard and rigorous models for the DC motor system and iteratively optimizing the control parameters for large and small angle cases. This part of the lab explored the relationship between time and frequency domain specifications as well as analysis using bode plots. Additionally, this showed the methodology of designing PD controllers given performance parameters. In the second lab section, the sets of control parameters were tested and shown to not meet the control specifications. This is expected to be due to power limitations of the DC motor system, as shown by Simulink simulations. This part of the lab explored the implementation of digital controllers as well as experimentation to verify simulation results. Overall, despite none of the sets of parameters meeting the desired specifications, these labs strengthened the skills involved with the derivation, optimization, and implementation of digital controllers in real world systems.

VII. References

[1] Alladi, Vijay et. al. (2001). *MEEN 364 Lab Manual: Lab 10 - Design of Position Controller for DC motor.* Texas A&M University. Web.

[2] Alladi, Vijay et. al. (2001). *MEEN 364 Lab Manual: Lab 11 - Implementation of DC Motor Position Control.* Texas A&M University. Web.

VIII. Appendices

Appendix A. MatLab Code.

Lab10.m

```
%% Parameter Estimation
% Values taken from Lab 4
Ks = 120.879*pi/180;
tau = .1415;

% Estimate Values
tr = 0.03*tau;
Mp = 0.1;
wn = 1.8/tr;
zeta = sqrt((log(Mp))^2/(pi^2+(log(Mp))^2));
Kp_est = wn^2 *tau/Ks;
Kd_est = (2*zeta*wn*tau -1)/Ks;
Theta_d = 1;
% Kp = Kp_est; %1.2059*10^4
% Kd = Kd_est; %33.1504

Kp = 12000;
Kd = 58;

Gc = tf([Kd Kp],1);
Gp = tf(Ks,[tau 1 0]);
T = Gc*Gp;

% stepinfo(T/(1+T));

%% Standard MatLab Simulation
opt = stepDataOptions('InputOffset',0,'StepAmplitude',Theta_d*pi/180);
[y,t] = step(T/(1+T),0:0.0001:0.25,opt);
y = [0;y]*180/pi;
t = [0;t+2];
figure(1);
hold on;
plot(t,y);
theta_mat_std.Time = t;
theta_mat_std.Data = y;
save('theta_mat_std','theta_mat_std');

%% Standard Simulink Simulation

N = 1000; % Filter Coefficient
sim Lab10_Sim_std;
```

```
plot(theta.Time,theta.Data);
title('');
axis([1.995 2.05 -.1 1.3])
xlabel('Time [s]');
ylabel('Simulated Motor Response');
legend('MatLab','Simulink','Location','Best')
theta_sim_std.Time = theta.Time;
theta_sim_std.Data = theta.Data;
save('theta_sim_std','theta_sim_std');


%% Rigorous MatLab Simulation

% Estimation
% Kp_r = 1.2059*10^4;
% Kd_r = 33.1504;

% Previous Optimization
% Kp_r = 12000;
% Kd_r = 58;

% Rigorous Optimization
Kp_r = 18000;
Kd_r = 1000;

Kt = 0.007;
Kb = 0.067;
R = 1.64;
L = 3*10^-3;
B = 44*10^-3;
J = 0.03;

Gc_r = tf([Kd_r Kp_r],1);
Gp_r = tf(1,[L*J/Kt, (L*B+R*J)/Kt, (R*B/Kt+Kb), 0]);
T_r = Gc_r*Gp_r;

% stepinfo(T_r/(1+T_r))

opt = stepDataOptions('InputOffset',0,'StepAmplitude',Theta_d*pi/180);
[y_r,t_r] = step(T_r/(1+T_r),1.5,opt);
y_r = [0;y_r]*180/pi;
t_r = [0;t_r+2];
figure(2)
hold on;
plot(t_r,y_r);
```

```
theta_mat_rig.Time = t_r;
theta_mat_rig.Data = y_r;
save('theta_mat_rig','theta_mat_rig');

%% Rigorous Simulink Simulation

C1 = L*J/Kt;
C2 = (L*B+R*J)/Kt;
C3 = (R*B/Kt+Kb);

sim Lab10_Sim_rig;
plot(theta_r.Time,theta_r.Data);
title('');
axis([1.995 2.35 0 1.2]);
xlabel('Time [s]');
ylabel('Simulated Motor Response');
legend('MatLab','Simulink','Location','Best')
theta_sim_rig.Time = theta_r.Time;
theta_sim_rig.Data = theta_r.Data;
save('theta_sim_rig','theta_sim_rig');
```

Lab11.m

```
%% Values found from Lab 10:
% Estimation | Standard Optimization | Rigorous Optimization
Kp_array = [1.2059*10^4, 12000, 18000];
Kd_array = [33.1504, 58, 1000];
labl = {'est','std','rig'};
Kx = 17.953; %From Lab 4
N = 1000;

% Automatically run all six tests and saves data
for Theta_d = [60, 120]
   for i = 1:3
      Kp = Kp_array(i);
      Kd = Kd_array(i);
      set_param('Lab11_Sim', 'SimulationCommand','start');

      while(~strcmp(get_param('Lab11_Sim','SimulationStatus'),'stopped'))
         pause(0.01);
      end

      ttl = sprintf('theta_%d_%s',Theta_d,labl{i});
      save(ttl,'theta');
      pause(0.5);
   end
```

```
end
```

analysis.m

```
%% Load Data

load('theta_mat_rig.mat');
load('theta_mat_std.mat');

load('theta_sim_rig.mat');
load('theta_sim_std.mat');

theta_60_est = importdata('theta_60_est.mat');
theta_60_std = importdata('theta_60_std.mat');
theta_60_rig = importdata('theta_60_rig.mat');

theta_120_est = importdata('theta_120_est.mat');
theta_120_std = importdata('theta_120_std.mat');
theta_120_rig = importdata('theta_120_rig.mat');

%% Plot Standard Simulations used to optimized gains
figure(1);
hold on
plot(theta_mat_std.Time,theta_mat_std.Data);
plot(theta_sim_std.Time,theta_sim_std.Data);
title('');
xlabel('Time [s]');
ylabel('Simulated Motor Response');
legend ('MatLab Standard Simulation','Simulink Standard Simulation',...
   'Location','Best');
axis([1.995 2.05 -.1 1.3])

%% Plot Rigorous Simulations used to optimized gains
figure(2);
hold on
plot(theta_mat_rig.Time,theta_mat_rig.Data);
plot(theta_sim_rig.Time,theta_sim_rig.Data);

title('');
xlabel('Time [s]');
ylabel('Simulated Motor Response');
legend ('MatLab Rigorous Simulation','Simulink Rigorous Simulation',...
   'Location','Best');
axis([1.995 2.35 0 1.2]);

%% Plot results of small angle tests
```

```
figure(3);
hold on
plot( theta_60_est.time, theta_60_est.signals(1).values,...
      theta_60_std.time, theta_60_std.signals(1).values,...
      theta_60_rig.time, theta_60_rig.signals(1).values);

plot( [0 5],[6 6],...
      [0 5],[54 54],...
      [0 5],[66 66]);

legend ('Estimated Parameters','Standard Optimization',...
   'Rigorous Optimization','10%','90%','110%','Location','Best');
xlabel('Time [s]');
ylabel('Angle [deg]');
axis([1.95 3 -10 80]);


%% Plot results of large angle tests
figure(4);
hold on
plot( theta_120_est.time, theta_120_est.signals(1).values,...
      theta_120_std.time, theta_120_std.signals(1).values,...
      theta_120_rig.time, theta_120_rig.signals(1).values);

plot( [0 5],[6 6],...
      [0 5],[120*.9 120*.9],...
      [0 5],[120*1.1 120*1.1]);

legend ('Estimated Parameters','Standard Optimization',...
   'Rigorous Optimization','10%','90%','110%','Location','Best');
xlabel('Time [s]');
ylabel('Angle [deg]');
axis([1.95 3 -10 180]);

%% Steady State Values
fprintf([ '\nSteady State\n',...
          '60 est: %f\n',...
          '60 std: %f\n',...
          '60 rig: %f\n',...
          '120 est: %f\n',...
          '120 std: %f\n',...
          '120 rig: %f\n\n'],...
          mean(theta_60_est.signals(1).values(end-100:end)),...
          mean(theta_60_std.signals(1).values(end-100:end)),...
          mean(theta_60_rig.signals(1).values(end-100:end)),...
```

```
        mean(theta_120_est.signals(1).values(end-100:end)),...
        mean(theta_120_std.signals(1).values(end-100:end)),...
        mean(theta_120_rig.signals(1).values(end-100:end)));


%% Calculate Tr
T10 = zeros(6,1);
T90 = zeros(6,1);
vars = {theta_60_est,theta_60_std,theta_60_rig,...
    theta_120_est,theta_120_std,theta_120_rig};
step = [60, 60, 60, 120, 120, 120];
for i = 1:6
  for t = 2000:1999+length(vars{i}.time(2000:end))
    if vars{i}.signals(1).values(t) > 0.9*step(i)
      T90(i) = vars{i}.time(t);
      break
    end
  end
  for t = 1999+length(vars{i}.time(2000:end)):-1:2000
    if vars{i}.signals(1).values(t) < 0.1*step(i)
      T10(i) = vars{i}.time(t);
      break
    end
  end
end

Tr = T90-T10;
fprintf([ '\nRise Time\n',...
        '60 est: %f\n',...
        '60 std: %f\n',...
        '60 rig: %f\n',...
        '120 est: %f\n',...
        '120 std: %f\n',...
        '120 rig: %f\n\n'],Tr);

%% Percent Overshoot

fprintf([ '\nPercent Overshoot\n',...
        '60 est: %f\n',...
        '60 std: %f\n',...
        '60 rig: %f\n',...
        '120 est: %f\n',...
        '120 std: %f\n',...
        '120 rig: %f\n\n'],...
        (max(theta_60_est.signals(1).values(2000:end))-60)/.6,...
```

```
(max(theta_60_std.signals(1).values(2000:end))-60)/.6,...
(max(theta_60_rig.signals(1).values(2000:end))-60)/.6,...
(max(theta_120_est.signals(1).values(2000:end))-120)/1.2,...
(max(theta_120_std.signals(1).values(2000:end)-120)/1.2),...
(max(theta_120_rig.signals(1).values(2000:end))-120)/1.2);
```

Appendix B. Simulink Code.

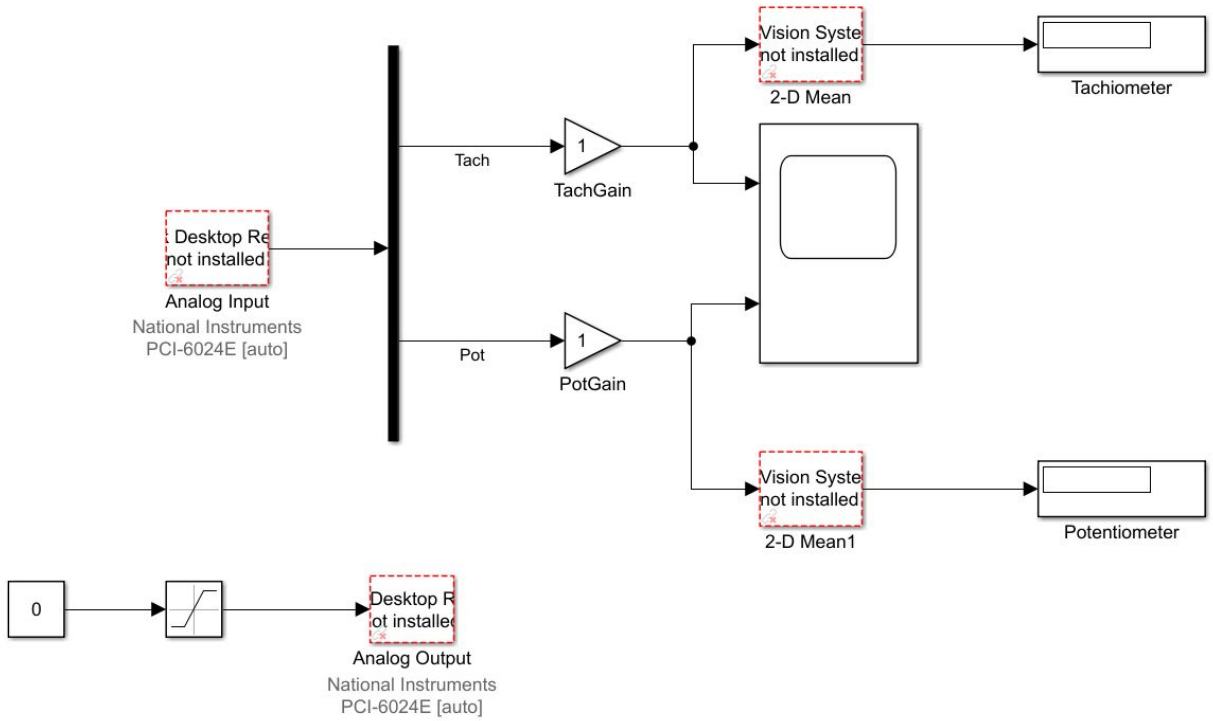

Figure 10. Lab 4 Simulink Code

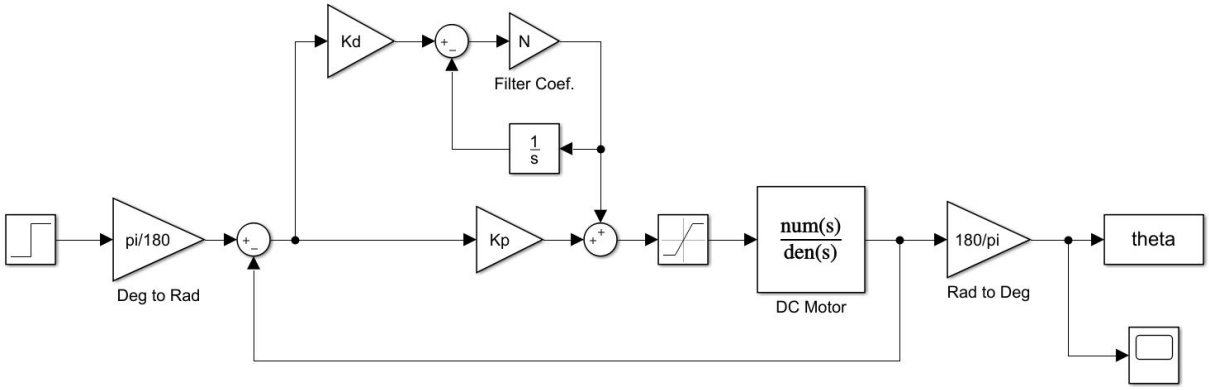

Figure 11. Lab 10 Standard DC Motor Model Simulink Code

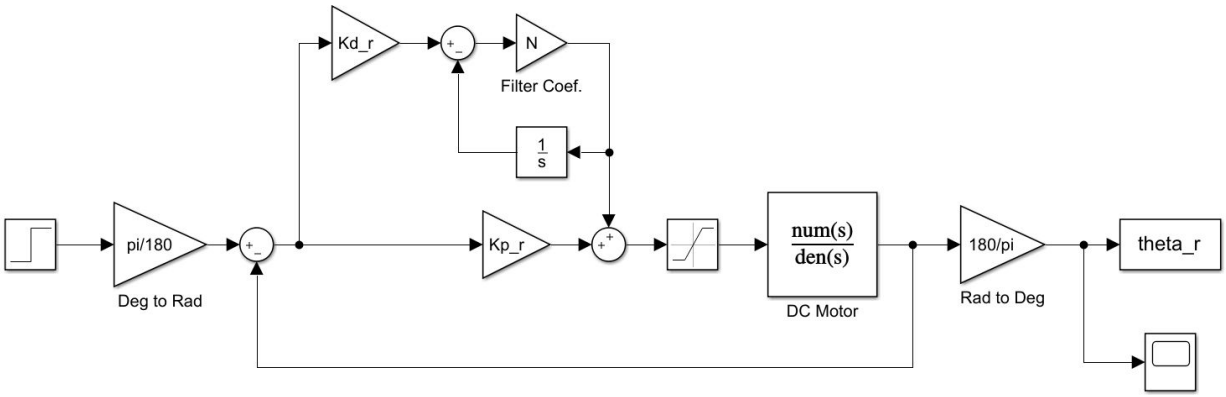

Figure 12. Lab 10 Rigorous DC Motor Model Simulink Code

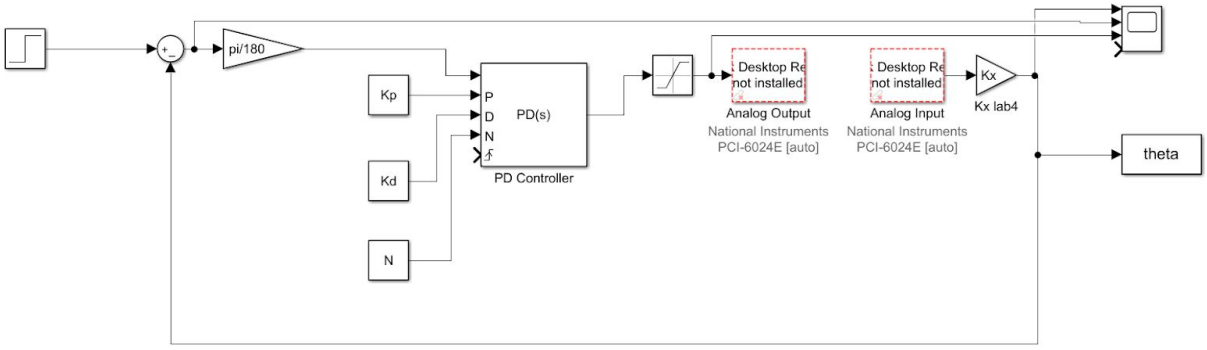

Figure 13. Lab 11 DC Motor Control Simulink Code

Appendix C. Calculations

Derivation of more rigorous model:

$$\left(\frac{L\cdot J}{K_t}\right)\cdot\frac{d^2\omega}{dt}+\left(\frac{LB+RJ}{K_t}\right)\cdot\frac{d\omega}{dt}+\left(\frac{RB}{K_t}+K_b\right)\cdot\omega=V_i \qquad (11)$$

$$\omega(t)=\frac{d\theta}{dt}$$

$$\mathcal{L}\{eq.11\}$$

$$\rightarrow\left[\left(\frac{L\cdot J}{K_t}\right)s^3+\left(\frac{LB+RJ}{K_t}\right)\cdot s^2+\left(\frac{RB}{K_t}+K_t\right)s\right]\Theta(s)=V_i(s)$$

$$\frac{\Theta(s)}{V_i(s)}=\frac{1}{\left(\frac{LJ}{K_t}\right)s^3+\left(\frac{LB+RJ}{K_t}\right)s^2+\left(\frac{RB}{K_t}+K_t\right)s}$$

Phase Margin Calculation Equations

$$M_p{}^{\%}=100*exp(-\frac{\pi\zeta}{\sqrt{1-\zeta^2}})$$

$$\zeta=\frac{ln\left(M_p{}^{\%}/100\right)}{\pi\sqrt{1+\left[\frac{ln(M_p{}^{\%}/100)}{\pi}\right]^2}}$$

$$PM=100\zeta$$

Appendix D. Raw Data and Figures.

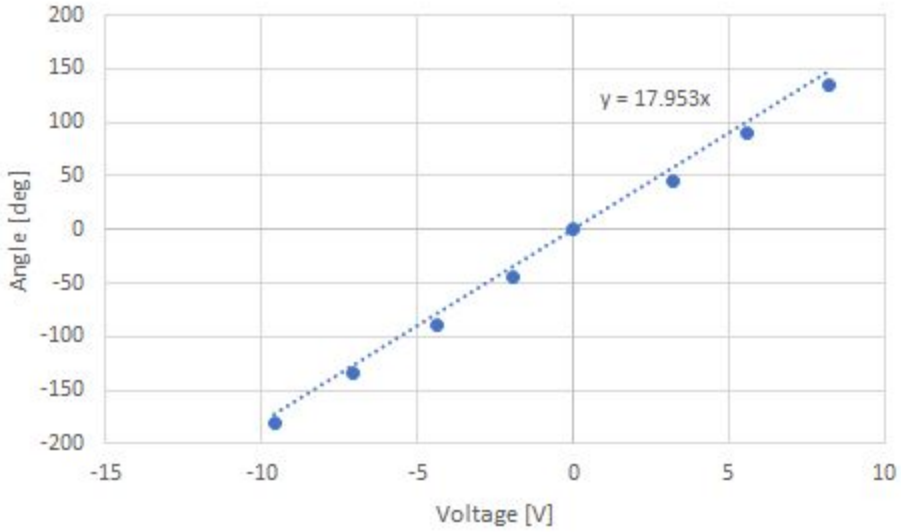

Figure 14. DC motor potentiometer calibration curve