# Table of Contents



```
close all; clear; clc
```

# Reading in Data

8 proportional 1.3 derivative

```
exp1 = readmatrix('data2/8_1pt3');
time1 = (exp1(:,1) / 1000);
time1 = time1 - time1(1);
angle1 = exp1(:,2);

% 10 proportional 0 derivative
exp2 = readmatrix('data/10_0');
time2 = (exp2(:,1) / 1000);
time2 = time2 - time2(1);
angle2 = exp2(:,2);

% 10 proportional 0.5 derivative
exp3 = readmatrix('data/10_0pt5');
time3 = (exp3(:,1) / 1000);
time3 = time3 - time3(1);
angle3 = exp3(:,2) * -1;

% 5 proportional 0 derivative
exp4 = readmatrix('data/5_0');
time4 = (exp4(:,1) / 1000);
time4 = time4 - time4(1);
angle4 = exp4(:,2);

% 5 proportional 0.5 derivative
exp5 = readmatrix('data/5_0pt5');
time5 = (exp5(:,1) / 1000);
time5 = time5 - time5(1);
angle5 = exp5(:,2);

% 8.4 proportional 0.25 derivative
exp6 = readmatrix('data2/8pt4_0pt25');
time6 = (exp6(:,1) / 1000);
time6 = time6 - time6(1);
angle6 = exp6(:,2);
```

# Analysis

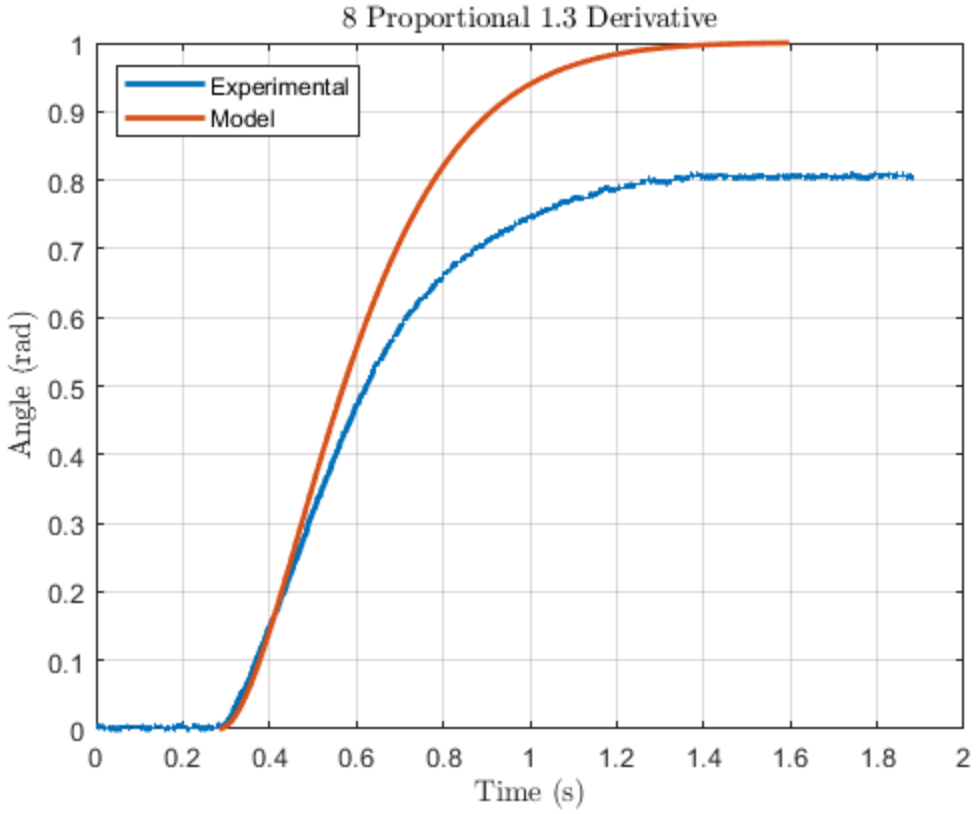
```
%First data set
logVec1 = time1 > 4.1 & time1 < 6;
xFunc1 = angle1(logVec1);
tFunc1 = time1(logVec1);
[xPlot1, tPlot1] = truncate(xFunc1, tFunc1);

% Shifting Up
xPlot1 = xPlot1 - min(xPlot1);

% Shifting Time
tPlot1 = tPlot1 - tPlot1(1);

%Second Data Set, 10 with no derivative
logVec2 = time2 > 0.4 & time2 < 2.2;
xFunc2 = angle2(logVec2);
tFunc2 = time2(logVec2);
[xPlot2, tPlot2] = truncate(xFunc2, tFunc2);

% Shifting Up
xPlot2 = xPlot2 - min(xPlot2);

% Shifting Time
tPlot2 = tPlot2 - tPlot2(1);

%Third Data Set, 10 with 0.5 derivative
logVec3 = time3 > 4.25 & time3 < 5.5;
xFunc3 = angle3(logVec3);
tFunc3 = time3(logVec3);
[xPlot3, tPlot3] = truncate(xFunc3, tFunc3);

% Shifting Up
xPlot3 = xPlot3 - min(xPlot3);

% Shifting Time
tPlot3 = tPlot3 - tPlot3(1);

%Fourth Data Set, 5 with 0 derivative
logVec4 = time4 > 3.4 & time4 < 4.8;
xFunc4 = angle4(logVec4);
tFunc4 = time4(logVec4);
[xPlot4, tPlot4] = truncate(xFunc4, tFunc4);

% Shifting Up
xPlot4 = xPlot4 - min(xPlot4);

% Shifting Time
tPlot4 = tPlot4 - tPlot4(1);

%Fifth Data Set, 5 with 0.5 derivative
logVec5 = time5 > 4.9 & time5 < 6.5;
xFunc5 = angle5(logVec5);
```

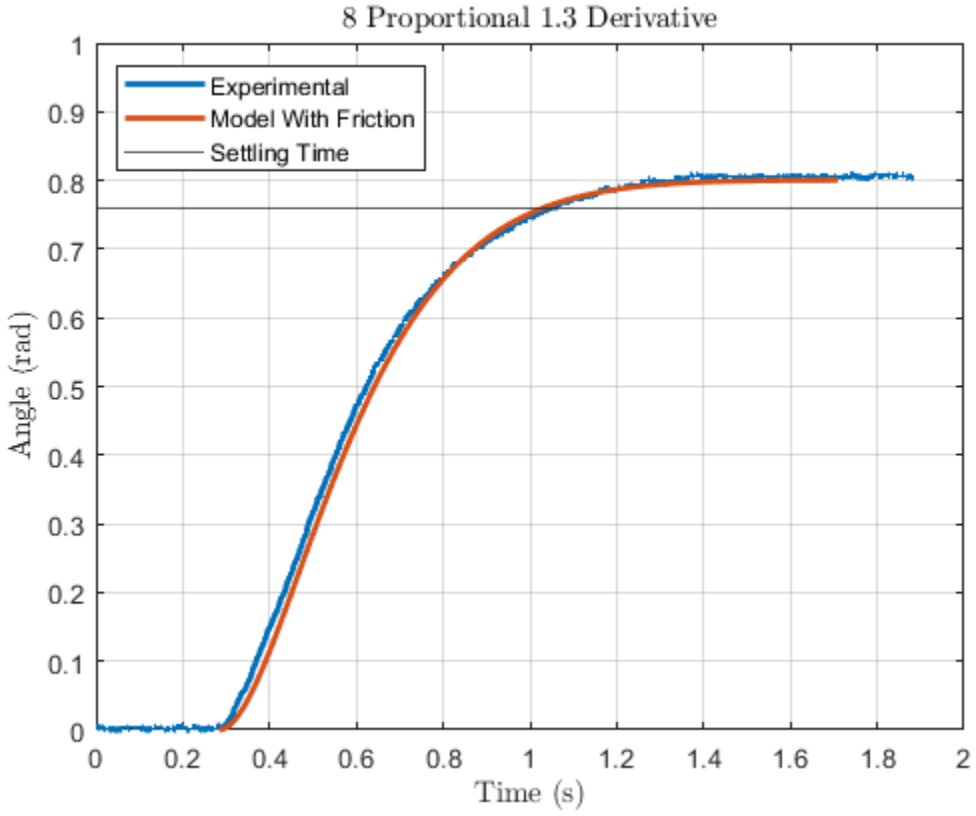
```
tFunc5 = time5(logVec5);
[xPlot5, tPlot5] = truncate(xFunc5, tFunc5);

% Shifting Up
xPlot5 = xPlot5 - min(xPlot5);

% Shifting Time
tPlot5 = tPlot5 - tPlot5(1);

%Fifth Data Set, 5 with 0.5 derivative
logVec6 = time6 > 8.6 & time6 < 15;
xFunc6 = angle6(logVec6);
tFunc6 = time6(logVec6);
[xPlot6, tPlot6] = truncate(xFunc6, tFunc6);

% Shifting Up
xPlot6 = xPlot6 - min(xPlot6);

% Shifting Time
tPlot6 = tPlot6 - tPlot6(1);
```

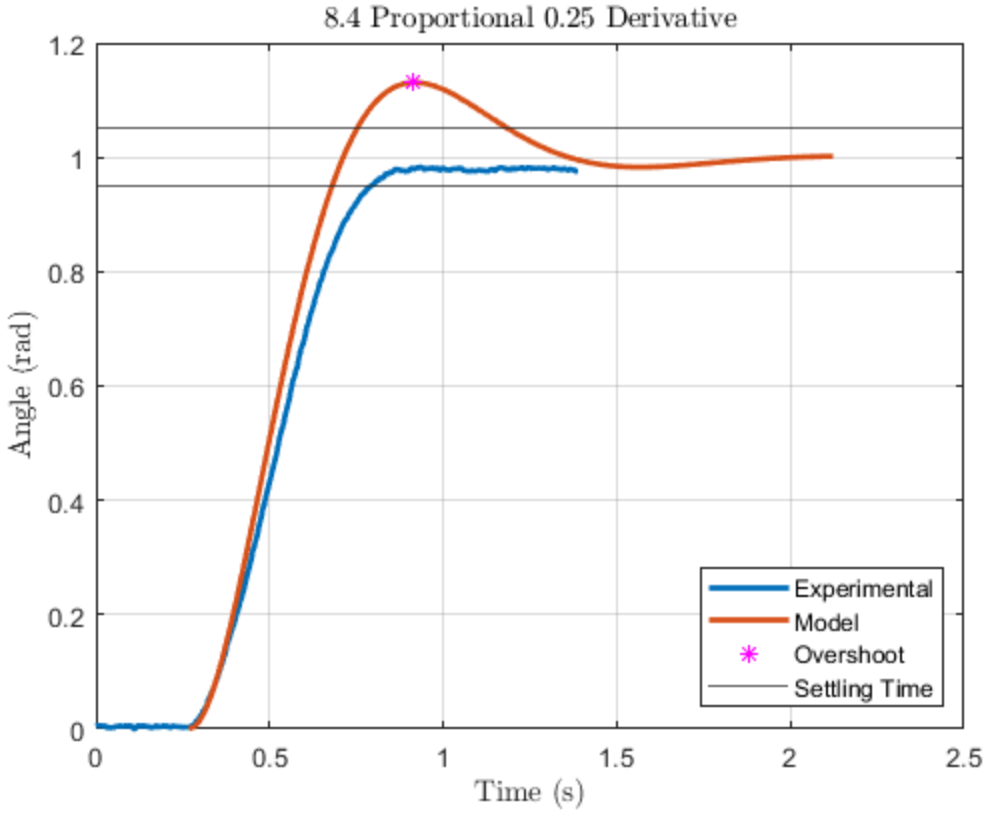
# Plotting

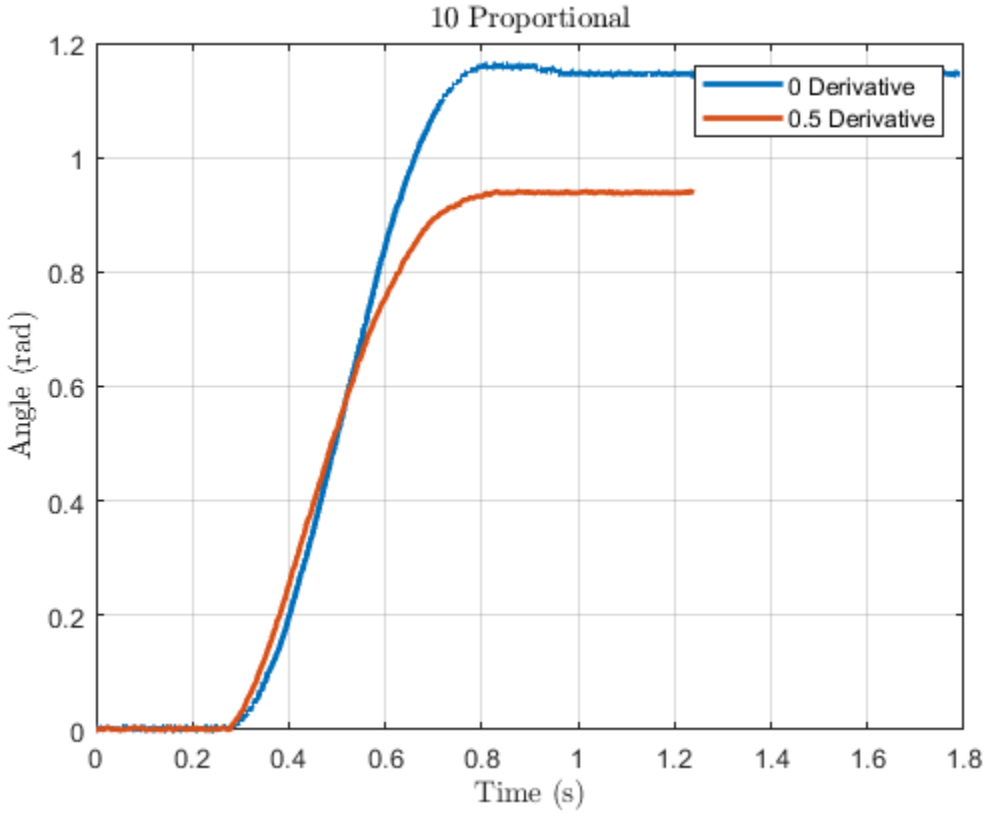
```
load('modelVars.mat'); %8 proportional and 1.3 derivative
figure(1)
set(0, 'defaultTextInterpreter', 'latex');
set(gca, 'FontSize', 12);
plot(tPlot1, xPlot1, 'linewidth', 2);
hold on
grid on
plot(t + 0.284, x, 'LineWidth', 2);
title('8 Proportional 1.3 Derivative');
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('Experimental', 'Model', 'location', 'northwest');
ylim([0 1]);

load('modelFric.mat'); %8 proportional and 1.3 derivative with
 friction
figure(2)
set(0, 'defaultTextInterpreter', 'latex');
set(gca, 'FontSize', 12);
plot(tPlot1, xPlot1, 'linewidth', 2);
hold on
grid on
plot(t + 0.284, x_ff, 'LineWidth', 2);
yline(0.8 * 0.95);
title('8 Proportional 1.3 Derivative');
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('Experimental', 'Model With Friction', 'Settling
 Time', 'location', 'northwest');
ylim([0 1]);
```

```
load('modelVars2.mat'); %8.4 proportional and 0.25 derivative
figure(3)
plot(tPlot6, xPlot6, 'linewidth', 2)
grid on;
hold on;
t = t + 0.268;
%Finding Overshoot
[maxVal, ind] = max(x);
plot(t, x, 'LineWidth', 2);
plot(t(ind), x(ind), 'm*');
yline(0.95);
yline(1.05);
title('8.4 Proportional 0.25 Derivative');
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('Experimental', 'Model','Overshoot', 'Settling
 Time', 'location', 'southeast');
ylim([0 1.2]);

figure(4)
plot(tPlot2, xPlot2, 'linewidth', 2);
grid on
hold on
plot(tPlot3, xPlot3, 'linewidth', 2)
title('10 Proportional');
xlabel('Time (s)');
ylabel('Angle (rad)');
legend('0 Derivative', '0.5 Derivative');
ylim([0 1.2]);
```


8 Proportional 1.3 Derivative
Experimental
Model
Angle (rad)
Time (s)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


8 Proportional 1.3 Derivative
Experimental
Model With Friction
Settling Time
Angle (rad)
Time (s)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2


8.4 Proportional 0.25 Derivative
Angle (rad)
Time (s)
Experimental
Model
Overshoot
Settling Time


10 Proportional
Angle (rad)
Time (s)
0 Derivative
0.5 Derivative

# Nick's Code

```
%Declare constants
Kg = 33.3; % Total Gear Ratio
Km = .0401; %[Nm/amp] Motor Constant
Rm = 19.2; %[Ohms]

J_hub = .00051; %[Kgm^2] Base Inertia
J_extra = .2 * .2794^2; %[Kgm^2]
J_load = .0015; %[Kgm^2] Load inertia of bar
J = J_hub + J_extra + J_load; %[Kgm^2] Total Inertia

Ts = 1; %[s] Settling time

figure()
% xline(Ts);
xlabel('Time (s)')
ylabel('Displacement')
yline(1.05, 'HandleVisibility', 'off')
yline(.95, 'DisplayName', '5% Settling Time')
legend()
hold on
i=1;
ff = -6.314;
n = .2; %Step size
%Calculate laplace function
for Kp0 = 8%-2:n:12 %[rad] Proportional Gain
   for Kd0 = 1.3%0:n:1.5 %[rad] Derivative Gain
        n1_ff = (Kp0 * Kg * Km / (J * Rm)) + ff; %Numerator with
 friction
        n1 = (Kp0 * Kg * Km / (J * Rm)); %Num without friction
        d2 = 1;
        d1 = (Kg^2 * Km^2 / (J * Rm)) + (Kd0 * Kg * Km / (J * Rm));
        d0 = (Kp0 * Kg * Km / (J * Rm));
```

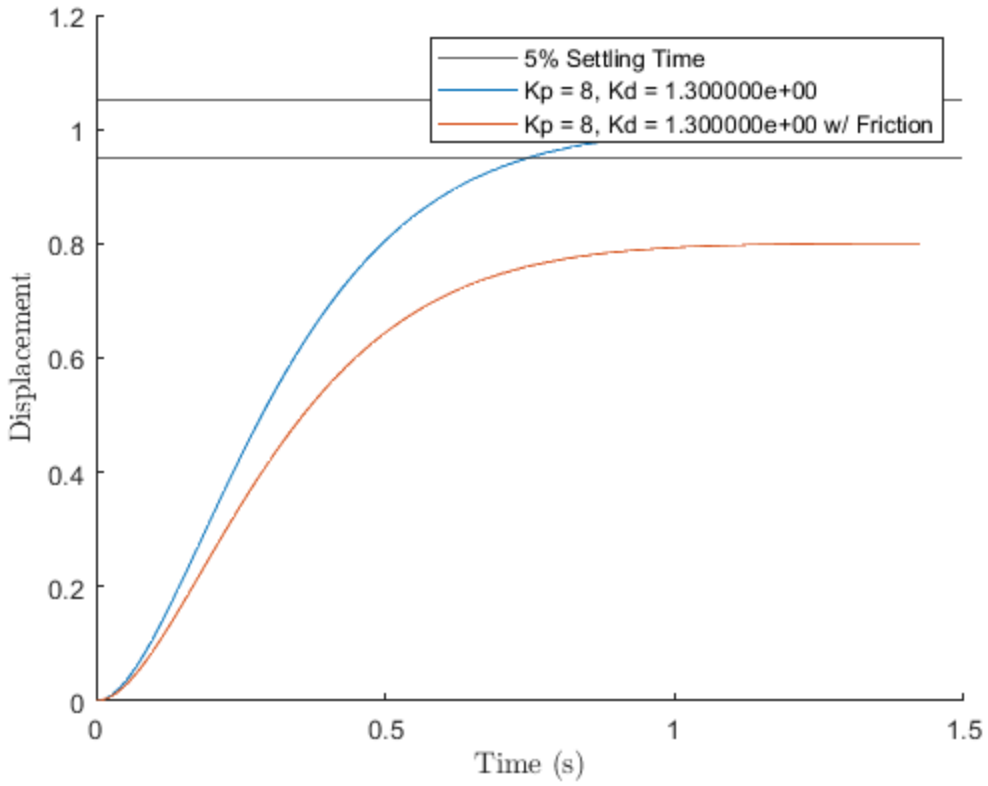
# Closed Loop System

```
        num = n1;
        den = [d2 d1 d0];
        sysTF = tf(num,den);
        sysTF_ff = tf(n1_ff,den);
```

# Step Response

```
        [x,t] = step(sysTF);
        [x_ff,t] = step(sysTF_ff);

        damp = (Kg^2 * Km^2) + (Kd0 * Kg * Km) / (2 * sqrt(Kp0 * Kg *
 Km * J * Rm));
        Mp = exp(-(damp * pi) / (sqrt(1 - damp^2))); %Maximum
 overshoot
```

```
        S = stepinfo(sysTF, 'SettlingTimeThreshold', .05); %Gather
 system properties such as overshoot and settling time
        if S.Overshoot < .2 && S.SettlingTime < Ts
            plot(t,x, 'DisplayName', sprintf('Kp = %d, Kd = %d ', Kp0,
 Kd0)) %Plot results
            hold on
            plot(t,x_ff, 'DisplayName', sprintf('Kp = %d, Kd = %d w/
 Friction', Kp0, Kd0))
            hold on
%             xline(S.SettlingTime, 'DisplayName', 'Settling Time')
%             yline(1 + S.Overshoot, 'DisplayName', 'Overshoot')

            Overshoot(i) = S.Overshoot;
            KP(i) = Kp0;
            KD(i) = Kd0;
        end
        i = i+1;
   end
end

index = find(KP);
KP = KP(index);
KD = KD(index);
Overshoot = Overshoot(index);
```



# Functions

```
function [x, t] = truncate(xRaw, tRaw)
    %Getting rid of noise, averaging 4 points
    %Function outputs averaged values from raw data to get rid of some
    %noise
    x = length(xRaw) / 4;
    t = length(tRaw) / 4;
    for j = 1:length(xRaw) / 4 - 1
        x1 = xRaw(4 * j);
        x2 = xRaw(4 * j + 1);
        x3 = xRaw(4 * j + 2);
        x4 = xRaw(4 * j + 3);

        t1 = tRaw(4 * j);
        t2 = tRaw(4 * j + 1);
        t3 = tRaw(4 * j + 2);
        t4 = tRaw(4 * j + 3);

        x(j) = mean([x1 x2 x3 x4]);
        t(j) = mean([t1 t2 t3 t4]);
    end
end
```

*Published with MATLAB® R2020b*