Control Lab 4: Design and Analysis of PID Controller

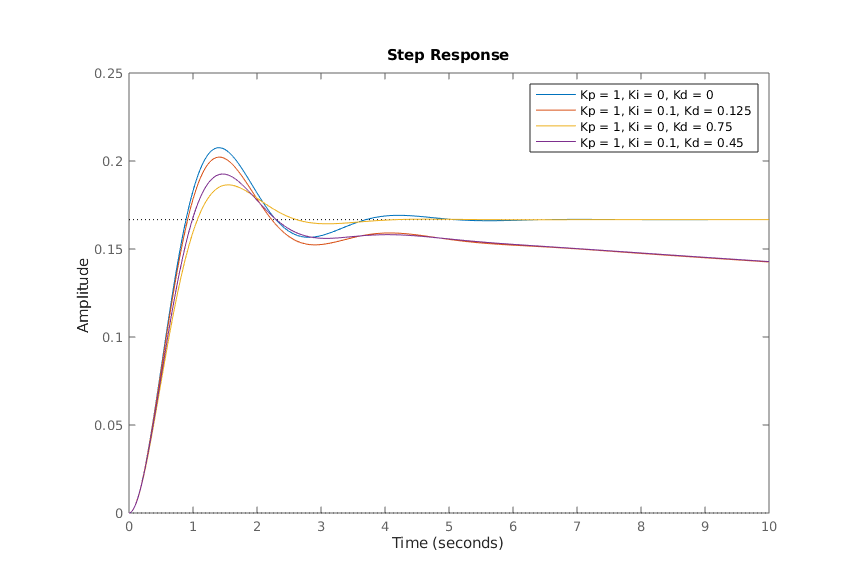
Group 6

| **Name** | **Work (%)** |
| --- | --- |
| Nate | 33% |
| Matt | 33% |
| Casey | 33% |

**Intro**  
 PID control is a widespread control scheme for many kinds of systems, such as a building heating system or maintaining fluid levels in a holding tank. In this lab, we closely analyze the characteristics of utilizing a PID control function. By plotting the simulated system over time and changing various parameters we can gauge the effect of those parameters, namely the Kp, Ki, and Kd coefficients to “tune” the system to reach the target value quickly.

**Code:**

| % Define the system transfer function G(s) = 1/(s^2 + 2s + 5)  num = 1;  den = [1 2 5];  G = tf(num, den);  % Define unit step input s  s = 0:0.01:10;  % Create the open-loop transfer function  G = tf(num, den);  % Create a figure for the plots  figure;  hold on;  % Create a legend for the plots  legendEntries = {};  % Create a list to store the PID systems  sys = [];  % Plot the step response of the system with different PID parameters  [sys, legendEntries] = plotPidSystem(1, 0, 0, G, s, legendEntries);  [sys, legendEntries] = plotPidSystem(1, 0.1, 0.125, G, s, legendEntries);  [sys, legendEntries] = plotPidSystem(1, 0, 0.75, G, s, legendEntries);  [sys, legendEntries] = plotPidSystem(1, 0.1, 0.45, G, s, legendEntries);  % Function to plot the step response of the system with a PID controller  function [sys, legendEntries] = plotPidSystem(Kp, Ki, Kd, G, s, legendEntries)  % Create the PID controller transfer function C(s) = Kp + Ki/s + Kd\*s  C = pid(Kp, Ki, Kd);  % Create Closed-loop system with the PID controller  sys = feedback(G, C);  % Simulate Step response of the closed-loop system  step(sys, s);  disp("Kp = " + Kp + ", Ki = " + Ki + ", Kd = " + Kd);  disp(stepinfo(sys));  legendEntries{end+1} = ['Kp = ' num2str(Kp) ', Ki = ' num2str(Ki) ', Kd = ' num2str(Kd)];  % Add a legend entry for the current PID parameters  legend(legendEntries);  end  % Add a legend to the plot  legend(legendEntries);  hold off; |
| --- |



**Figure 1:**  Plot of the amplitude response of various closed-loop pid systems with respect to time.

**Listing 1:** Step info of each analyzed PID system

| Kp = 1, Ki = 0, Kd = 0  RiseTime: 0.6039  TransientTime: 3.4322  SettlingTime: 3.4322  SettlingMin: 0.1557  SettlingMax: 0.2075  Overshoot: 24.5005  Undershoot: 0  Peak: 0.2075  PeakTime: 1.4276  Kp = 1, Ki = 0.1, Kd = 0.125  RiseTime: 0  TransientTime: 222.5219  SettlingTime: NaN  SettlingMin: 0  SettlingMax: 0.2022  Overshoot: Inf  Undershoot: 0  Peak: 0.2022  PeakTime: 1.3980  Kp = 1, Ki = 0, Kd = 0.75  RiseTime: 0.7207  TransientTime: 2.3921  SettlingTime: 2.3921  SettlingMin: 0.1521  SettlingMax: 0.1864  Overshoot: 11.8700  Undershoot: 0  Peak: 0.1864  PeakTime: 1.5406  Kp = 1, Ki = 0.1, Kd = 0.45  RiseTime: 0  TransientTime: 225.3407  SettlingTime: NaN  SettlingMin: 0  SettlingMax: 0.1925  Overshoot: Inf  Undershoot: 0  Peak: 0.1925  PeakTime: 1.4384 |
| --- |

**Analysis**

When analyzing the data produced by the stepInfo() function, there are a few noticeable trends. For this experiment, however, we decided to keep the Kp values consistent throughout the varying PID systems to draw more attention to the Ki and Kd values. For certain PID systems with a nonzero Ki value, the RiseTime typically produces zero, the overshoot goes off into infinity, and there is a noticeably larger TransientTime in comparison to systems without Ki. When evaluating the varying Kd values, it is evident that it affects the SettlingMax value. The larger the Kd, the smaller the SettlingMax time.

**Discussion and Conclusion**

In this lab we gained a deeper understanding of a PID controller and how its gain coefficients affect its dynamics. Using MATLAB’s built-in functions pid(), feedback(), and stepinfo(), to establish our transfer function and generate a system to control. In general, Kp mostly affects transient response, unless overly dampened by Kd. And Ki helps keep the system from lingering in a steady state offset from a target value. All in all these functions and PID controllers in general prove to be very useful and customizable to suit a large variety of system control scenarios.