Lab Week 6: Liquid Level PID Controller

0. Objectives

- Learn about PID controllers and optimize controller gains for the liquid level system

1. Overview

This multi-week lab consists of using MATLAB Simulink and Arduino microprocessors to control a liquid level system. The system operates by manipulating the outlet valve to maintain a desired water height regardless of inlet flowrate.

2. System Description

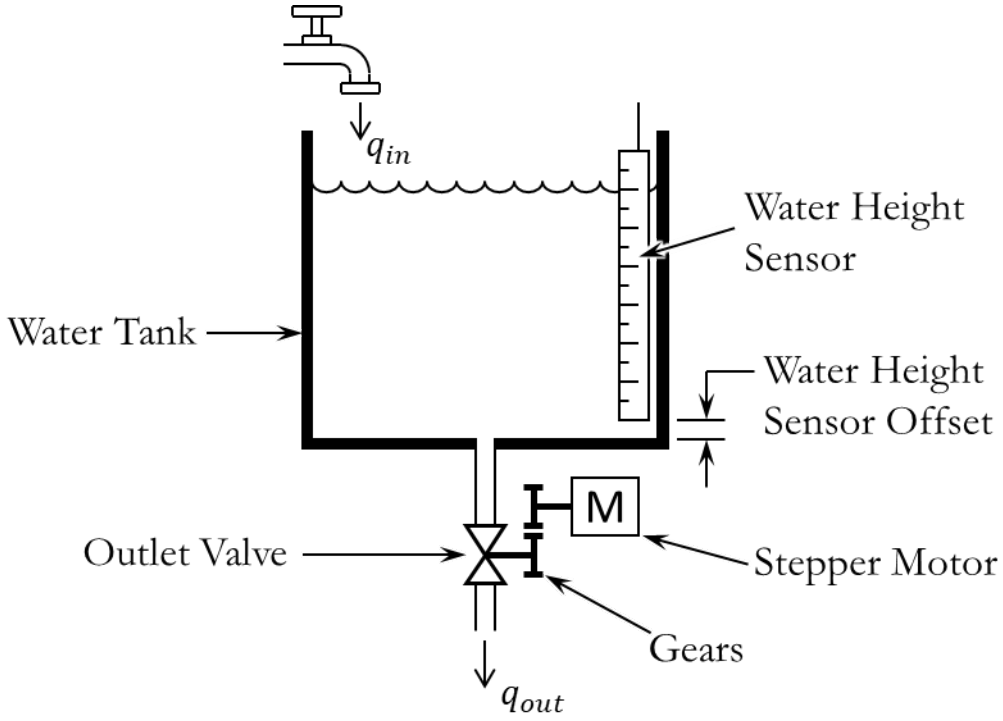

Figure 1: Mixing tank schematic

2.1 Main System Components:

- Outlet valve: controls outlet flowrate to maintain desired water height
- Stepper Motor
  - Opens and closes outlet valve via gears
  - When outlet valve is fully closed, the stepper motor position is 'steps = 0'
  - When outlet valve is fully open, the stepper motor position is 'steps = 1020'
- Water height sensor
  - Detects water height
  - Also referred to as an eTape sensor
- Water tank

2.2 Main system variables:

- Inlet flowrate (q_{in})
  - Inlet flowrate of water
  - Value of flowrate is unknown, uncontrollable, and can vary
  - Units: cm^3/s
- Outlet flowrate (q_{out})
  - Outlet flowrate of water
  - Flowrate will vary depending on outlet valve position **and** actual water height
  - Initial outlet flow rate is 0 cm^3/s
  - Units: cm^3/s
- Outlet valve position (X_{valve})
  - Fully closed position = 0 steps
  - Fully open position = 1020 steps
  - The valve position can be anywhere between 0 steps and 1020 steps
  - Initial valve position is 0 steps
  - Units: steps
- Water tank cross-sectional area (A_{tank})
  - Assume constant
  - 643 cm^2
  - Units: cm^2
- Setpoint water height ($h_{setpoint}$)
  - The desired water height
  - Units: cm
- Actual water height (h_{actual})
  - This is the output variable; the variable we want to control
  - The setpoint water height will be specified. By adjusting the outlet valve, the actual water height will change until it matches the setpoint water height
  - The initial actual water height is 0 cm
  - Units: cm
- Sensor water height (h_{sensor})
  - This is the output/reading of the water height sensor
  - Units: cm
- Water height sensor offset (h_{offset})
  - 3.5 cm
  - Units: cm

3. Objectives

Using MATLAB Simulink and Arduino microprocessors, a liquid level system will be controlled by manipulating the outlet valve to maintain a desired water level regardless of inlet flowrate. The basic steps for completing the multi-week lab are as follows:

1. Create a block diagram of the full system simulation model (completed Lab Week 4)
2. Characterize and model system components (completed Lab Week 5)
3. Build simulation model of full system using Simulink (completed Lab Week 5)
4. Run simulation model to optimize controller and predict behavior
5. Build physical system control model
    - Create new block diagram that will be used for the physical control model
6. Compare simulated results and experimental results

4. Background: PID controller

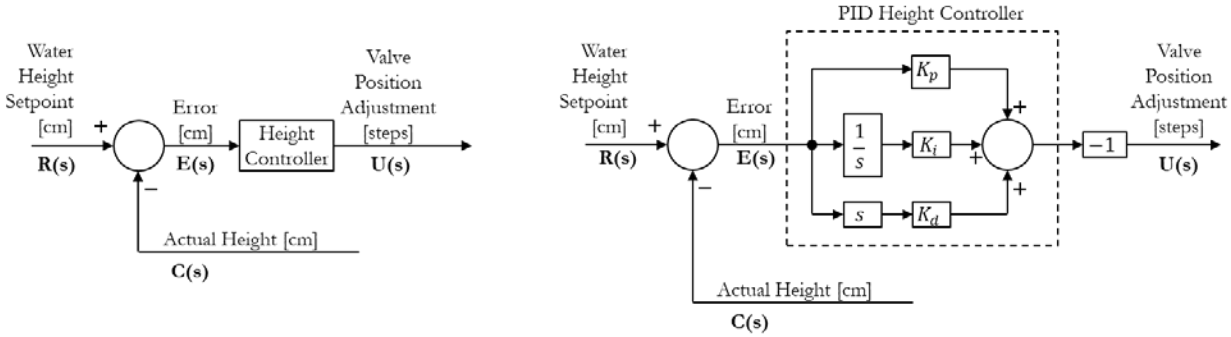

Figure 2: PID Block Diagram (s-domain)

The beginning of a block diagram starts with the setpoint, $R(s)$, being compared to the output, $C(s)$, and forming the error, $E(s)$. This error signal is fed into a controller block and is transformed into a command signal depending on the type of controller used. A PID controller has three components that act on the incoming error signal: proportional, integral, and derivative. The three components (also called control actions) are then added together and become the output of the PID controller block, $U(s)$. This output represents the total controller command that provides the input to the physical system block (the physical system block is not shown in Figure 2).

Summing the three control actions results in the following transfer function and time equivalent equations, where U(s) is the controller output, E(s) is the error, and K_p, K_i, and K_d are the proportional gain, integral gain, and derivative gain, respectively:

$$\frac{U(s)}{E(s)} = K_P + \frac{K_i}{s} + K_d\, s \qquad u(t) = K_P e(t) + K_i \int e(t)dt + K_d \frac{de(t)}{dt} \qquad \text{(Eqn. 1)}$$

Using the PID transfer function as shown above, it is often common to combine the three blocks to a single block as shown in Figure 3 below. In Simulink, a 'PID Controller' block is used.

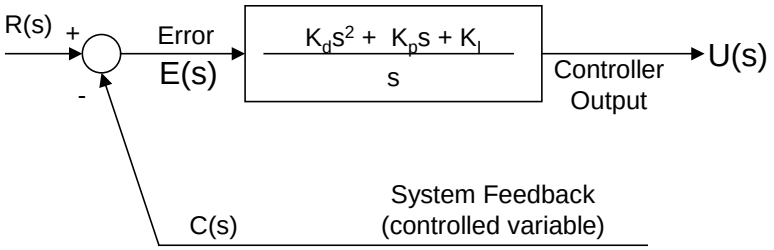

Figure 3: Single transfer function block for PID controller

The easiest way to illustrate the equations is to examine the controller output when a known error is the input. If a ramp change in error is the input into a PID controller, then each component of the PID controller will contribute the following control action to the error, as shown below. The total controller output is simply the three components added together.

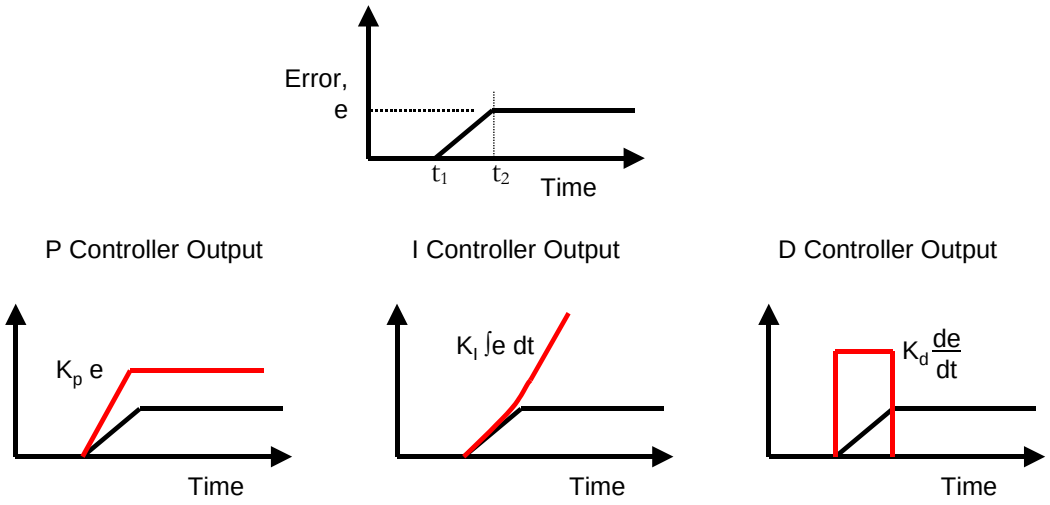

Figure 4: Contributions from each PID component

- Time = 0 to t_1
  - Error is a constant zero
  - P, I, and D components have no contribution
- Time = t_1 to t_2
  - Error begins to increase with a ramp change
  - Because the error is non-zero, P and I both begin to have a non-zero output
  - Since the rate-of-change of the error is non-zero, D has a non-zero output
- Time = t_2 and greater
  - Error is a constant non-zero number
  - P and I continue to have a non-zero output
  - Since the rate-of-change of the error is zero, D has an output of zero

If a controller utilizes all three components (proportional, integral, and derivative), it is called a PID controller. This means that the gains K_p, K_i, and K_d are some constant value greater than zero. Other controllers based on the PID architecture include

- P controller
  - $K_p > 0$
  - $K_i = K_d = 0$
- PI controller
  - $K_p > 0$
  - $K_i > 0$
  - $K_d = 0$
- PD controller
  - $K_p > 0$
  - $K_i = 0$
  - $K_d > 0$

4.1 General Characteristics of PID Gains

- Proportional gain: K_p
  - Speeds up response time
  - Adds oscillations and instability
  - Reduces steady state error but can never fully eliminate steady state error
- Integral Gain: K_i
  - Speeds up response time
  - Eliminates steady state error
  - Adds oscillations and instability
- Derivative Gain: K_d
  - Anticipates errors before they happen and attempts to correct the error before it occurs
  - Decreases oscillations and dampens the system
  - Amplifies noisy signals and causes chatter

5. Assignment

Using Simulink, build the model shown below in Figure 6. Below are various parameters to set within your model (keep all other options unchanged). Save your Simulink model to a **specific folder location** using the filename **`Lab_Week_6_model`**

- Step function
  - Step time: 0
  - Initial value: 0
  - Final value: 5
- PID Controller (see Figure 5, disregard any error that appears)
  - Proportional (P): **`Kp`**
  - Integral (I): **`Ki`**
  - Derivative (D): **`Kd`**
- Transfer Function
  - As shown in Figure 6
- To Workspace
  - Variable name: **`setpoint`** and **`output`**
  - Save format: Timeseries
- Simulation → Model Configuration Parameters
  - Solver → Solver: ode23t
  - Solver → Stop Time: 10
- Simulink model
  - Save your model in a specific folder location
  - Name your Simulink model **`Lab_Week_6_model`**

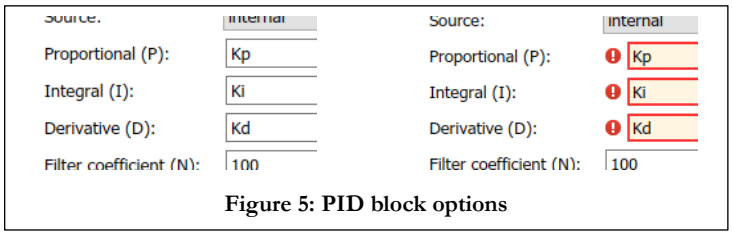

Figure 5: PID block options

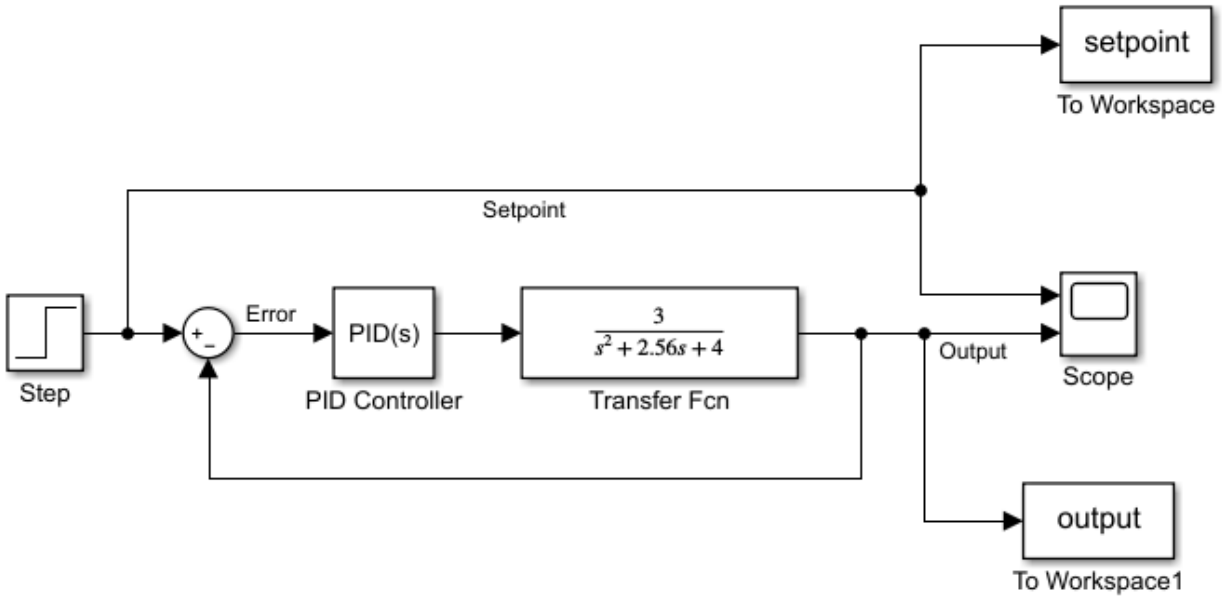

Figure 6: Simple Simulink unity feedback model

5.1 Vary K_p and plot the results ($K_i = K_d = 0$)

- In the **same folder location** that your Simulink model is saved, create a Matlab script that is the **same** as the script shown in the Appendix. If you copy-and-paste the Matlab script from the Appendix, you may encounter some errors. Sometimes copy-and-pasted text from a PDF may not work correctly within Matlab. You can give the script any filename you want.
- Run the script in order generate a plot comparing different values of K_p (1, 5, and 10)
  - $K_p = 1, 5, and\ 10$
  - $K_i = K_d = 0$
  - Notice that the script will run the Simulink model three times, save the data each time, and then create a single plot with all the traces from the three model runs
  - Your plot should look like Figure 7

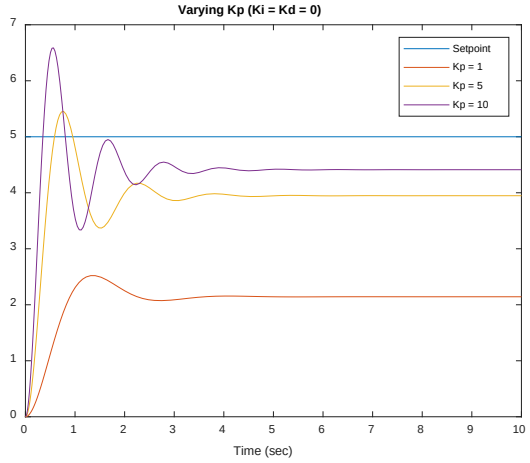

Figure 7: Problem 5.1 varying K_p

5.2 Vary K_i and plot the results

- Modify the script and run it to generate a plot comparing different values of K_i
  - $K_p = 5$
  - $K_i = 0, 1, 2.5, and\ 5$
  - $K_d = 0$
  - Your plot will have 5 traces

5.3 Vary K_d and plot the results

- Modify the script and run it to generate a plot comparing different values of K_d
  - $K_p = 5$
  - $K_i = 5$
  - $K_d = 0, 0.5, 1, and\ 2$
  - Your plot will have 5 traces

5.4 Tune controller gains on the liquid level model from Lab Week 5

Using what you learned and observed in Problems 5.1 – 5.3, adjust the controller gains from your liquid level model that was created in Lab Week 5 in order to achieve a good result (minimize oscillations, minimize overshoot, have a fast settling time, etc.).

- Plot your best result (water height vs. time)
- Hint: all of your gains will be less than 50

6. Deliverables

- Problem 5.1
  - Plot of system response with varying values of K_p as specified above. Your plot will have four traces. Include title, legend, and x-axis label and units. You do not need a y-axis label or units. Your plot should look like Figure 7.
  - Answer the following questions
    - What effect does K_p have on the system response?
    - What happens to the steady state error when you increase K_p?
    - Can you get rid of steady state errors by only using K_p?
    - What happens if you keep increasing K_p past 10?
- Problem 5.2
  - Plot of system response with varying values of K_i as specified above. Your plot will have five traces on it. Include title, legend, and x-axis label and units. You do not need a y-axis label or units.
  - Answer the following questions
    - What effect does K_i have on the system response?
    - What happens if you keep increasing K_i past 5?
- Problem 5.3
  - Plot of system response with varying values of K_d as specified above. Your plot will have five traces on it. Include title, legend, and x-axis label and units. You do not need a y-axis label or units.
  - Answer the following questions
    - What effect does K_d have on the system response?
    - What happens if you keep increasing K_d past 2?
- Problem 5.4
  - Plot of system response (water height vs. time). Your plot needs two traces on it (setpoint height and actual height). Include title, legend, x-axis label and units, and y-axis label and units.
  - What are the values that you used for K_p, K_i, and K_d?
- Don't forget a cover sheet
- You do not need to write an executive summary

7. Appendix – Matlab script

```
% your name
% ABE 460
% Lab Week 6
% Date of lab assignment
% problem number (e.g. Problem 5.1)

clear all % clears all variables and functions from memory
close all % closes all open figure windows
clc % clears the command window

Kp = 1;
Ki = 0;
Kd = 0;
sim('Lab_Week_6_model') % runs the model called Lab_Week_6_model
output_a_time = output.Time;
output_a_data = output.Data;

Kp = 5;
sim('Lab_Week_6_model') % runs the model called Lab_Week_6_model
output_b_time = output.Time;
output_b_data = output.Data;

Kp = 10;
sim('Lab_Week_6_model') % runs the model called Lab_Week_6_model
output_c_time = output.Time;
output_c_data = output.Data;


figure(1)
plot(setpoint.Time, setpoint.Data)
hold on
plot(output_a_time, output_a_data)
plot(output_b_time, output_b_data)
plot(output_c_time, output_c_data)
hold off
xlabel('Time (sec)')
legend('Setpoint','Kp = 1', 'Kp = 5', 'Kp = 10')
title('Varying Kp (Ki = Kd = 0)')
```