Laboratory #9
PID Control of Liquid Height in Tank

Introduction

In this lab, you will build a PID control algorithm in MATLAB and use it along with the Arduino Uno, Arduino motor shield, valve, motor, and pressure sensor in a feedback loop to control the volume of water in a tank. You will construct a system in which a continuously draining tank is supplied with water via a continuously running centrifugal pump. A pressure transducer mounted on the side of the tank returns a voltage that is proportional to the height of water above the transducer. This voltage serves as the input to the controller. The controller actuates a valve that controls the water level in the tank.

In industry, one typically uses control valves with integrated actuators that are mounted directly on the stem of the valve. Actuation is often done pneumatically with compressed air that is controlled by an electrical signal via a transducer. Servo motors are typically not used due to their slower response time and higher capital and maintenance costs. Using servo motor for valve control allows us to be independent of a compressed air source.

The rationale of this lab is to put the theory that you learned in class to practical use as well as to get first-hand experience with the problems that one runs into when trying to implement a PID algorithm. Please read this entire document before getting started on the last lab assignment.

Safety and Lab Behavior

Due to liability concerns, the Harshbarger lab can only be used during normal business hours. If you are working outside of normal business hours you must be supervised by one of the TA's.

You will be sharing the space with other people. Be courteous to your fellow students. Clean up your work area when you are done. You must wear safety glasses, closed toe shoes, and pants

while working in the Harshbarger lab area.

Each centrifugal pump has been equipped with a GFI protected plug. This is to protect you from possible electric shock. Do not attempt to use the pump without the GFI plug and ensure your outlet is properly grounded. Similarly, use care to avoid spills; position your tank so that if it does spill, your computer, the Arduinos, and the motor do not get wet. Clean up any spills to avoid hazards. Do not plug or unplug equipment with wet hands.

Objectives

To successfully complete this lab you will:

- Construct an appropriate and user friendly GUI with `uicontrol` commands in MATLAB to implement the front end of the controller. If you are still having trouble getting `uicontrol` to do your bidding, take a look at the solution of problem 1 on exam 2 in 2013. Don't feel obligated to follow this approach. Several of you have built very nice `uicontrol` programs.
- Interface your MATLAB script with the Arduino Uno and Arduino motor shield (Labs #4 and 5).
- Design and assemble the tank control system using the pump, valve, servo motor, pressure transducer, op amp (Labs #1, 2, and 3), tank, cables, Lego blocks, piping, and fittings provided to your group. Use the op amp to electrically isolate the pressure sensor, which is in contact with water, by running the signal through a unity gain, inverting buffer amplifier (Lab #3).
- Construct a calibration curve to relate the voltage signal to inches of water (Lab #8).
- Perform a frequency response analysis (not yet covered in class).
- Calculate PID parameters for your system using empirical tuning correlations (not yet covered in class).
- Test the calculated parameters by asking the system to transition from steady-state at one set point to a new set point. Think about how you can define or quantify adequate performance. Decide if or how to adjust your parameters to improve the PID performance. Rationally tune your parameters to achieve optimal control.
- Allow the instructors to use your system and check its basic features. This is best done early to get (positive) feedback on system operation.

- Propose and implement an improved control algorithm. This could be as simple as incorporating a derivative filter or as involved as developing a new algorithm specific to the tank. Demonstrate capabilities by asking the system to transition from steady-state at one set point to a new set point.
- Perform an error analysis consisting of a) computing the 95% confidence interval of a parameter of your choosing and b) completing one propagation of error calculation (this topic will be discussed in lecture).
- Submit one written report per group documenting your results. The report is due on the last day of class. Follow the lab memo writing guide on D2L.

Work as a team. Try to allow everyone to contribute to each part of the lab. The point is not just to efficiently complete the lab and generate a report, but to learn something and this can only happen by engaging everyone in the effort.

Practical Considerations

- Don't forget to turn off hardware when your script is done. A well-written script will brake the motor and clear the memory buffer and delete the Arduino object from the MATLAB workspace when desired.
- Always provide an external (battery) power source to the Arduino motor shield.
- Write scripts that are intelligent enough to not attempt moving the valve beyond its mechanical limits. Driving the valve too far into the open or closed positions can damage the threads. Needle valves, such as the ones used here, are not designed to totally shut off flow, but are rather intended just to throttle the flow rate.
- If the valve is fully closed while the pump is running, you will effectively "dead-head" the pump. Centrifugal pumps are forgiving in that they can be temporarily operated in a dead-head condition without damage or generating unsafe pressures, but leaving the pump "dead-headed" for too long generates heat and can shorten its lifetime.
- Actuating the valve to its mechanical limits will also force the motor into a stall. Please do not stall the motors–the DC motors draw high current when stalled and will quickly generate excessive heat. You can achieve satisfactory control without ever driving the valve to its extremes.

- As the stem of the valve is rotated open or closed, the threads on the stem cause it to translate in or out. When you mount the motor to the valve, either allow the motor to translate with the valve stem or allow the stem to translate with respect to the motor.
- Depending on how quickly the water flows out the drain, the filling/draining behavior of the tank could be very non-symmetric. For example, if the water flows out the drain almost as fast as the pump can add water, the system will be very slow to fill but quick to drain. Conversely, if the drain is restricted, the system will be quick to fill but slow to drain. You can decrease the natural flow rate of water from the tank, if desired, by using the clamp provided to crimp the hose attached to the tank drain.
- The brass fittings used to connect the pieces of brass tubing are designed to be connected and disconnected quickly and easily. If you have any difficulty figuring out how the fittings work, please ask someone for help.

Motor Tracking Assignment

Servo motors utilize a mechanism to sense internal position called an encoder allowing the motors to be used for precise movements. This feature is essential to using the motors to control a robot, for example. See Lab #7 for an explanation of how the encoder works that is built into the Lego motor. The encoder is an internal angle sensor. In principle, this signal would be read using MATLAB and used to ensure precise motor control. This way we could be certain that the motor was not trying to open or close the valve more than physically possible because we could monitor the current valve position. Unfortunately the instructor has not yet been able to get MATLAB to read the encoder information via the Arduino Uno and motor shield. The problem is caused by the slow signal speed through usb ports. MATLAB can only read at a frequency of about 100 Hz (or 100 reads per second). This is much faster than required to keep up with the liquid level in the tank, of course, but too slow to record all of the information on internal position generated by the encoder, which generates data at a rate of about 1 kHz. There are two special pins on the Arduino Uno (digital pins 2 and 3) designed to handle so-called interrupts, which can take data at very high rates. Alas, not for want of trying, we have been unable to record meaningful encoder information.

Try and develop a work around to keep track of how far the valve is open and limit the movement of the valve to stay within its physical limits (see the control script assignment below as well for requirements).

Control Script Assignment

Further details on the expectations for the PID control script follow. Write a MATLAB script that implements PID control of the water level in the tank, accepting input from the pressure transducer and influencing the flow into the tank by actuating a control valve located between the pump and the tank.

- The script must run as a continuous loop until the user chooses to exit.
- The script should be capable of accepting multiple set point changes without interruption.
- Ensure that the servo motor does not attempt to drive the valve beyond its mechanical limits. The script should be able to adjust the valve anywhere between fully closed (0°) and fully open (about 4 complete turns, about 1440° of rotation). The script should continuously check the actual position of the valve to make sure accumulated error does not cause the script to actuate the valve beyond its mechanical limits.
- While the script is running, collect and write performance data to a text file that can be used to demonstrate the performance of the control script for the final report. Make sure that the data is saved to avoid loss of data in the event of a computer or program crash.