PID Temperature Control

A PID controller is a general-purpose control system that attempts to automatically adjust a system so as to maintain its operation at a desired setpoint. The purpose of this experiment is to gain experience in PID control by precisely controlling the temperature of a piece of aluminum. It is possible, with this apparatus, to set the temperature of the metal to anything between -5° C and 40° C, with temperature fluctuations on the order of 0.02° C.

Theory

In any control system, there will be some setpoint S_o. S_o is the desired output: it could be the temperature, the height of a levitating ball, the position of a beam of protons, *whatever.* S_o is (generally) set by the user: the thermostat setting at your house, for example, would be S_o.

There will also be an error signal δ in any control system. δ is the difference between the setpoint S_o and the system state S. Ideally we want δ to be zero, but that is rarely the case for any system complex enough to need a PID controller.

Finally, there is the power output U for the control system. U is the system output, which attempts to bring the value of δ back to zero. For the case of a thermostat, U would be the power to the heater (cooler) if the temperature was too cold (hot).

Proportional gain

The first and simplest approach to minimizing δ is to set U to be proportional to δ.

$$U_p = K_p\delta$$

K_p is the *proportional* gain. If K_p is too low, then the system may take a long time to get to the setpoint after a change in S_o. If K_p is too high, the system may go into oscillation around S_o. Finding the "Goldilocks point" in the middle is the challenge!

Integral gain

One problem with proportional gain is that if the system state S is equal to the setpoint S_o, $\delta = 0$ and $U_p = K_p\delta = 0$ also. For most real systems, there are losses: heat loss through the building walls in the case of central

heating, for example. These losses require a steady-state value of U that is not generally 0. As a result, the system will reach some state $S' < S$ such that $U_p = K_p\delta = U_{loss}$. This "droop" is often undesirable.

The way of fixing "droop" is to keep track of the value of δ in the past. If δ has been positive for awhile, increase U. Mathematically, this is most conveniently expressed as an integral:

$$U_i = K_i \int \delta(t)\, dt$$

This eliminates droop: if there is a constant value of δ, it contributes to the integral and increases the power output U_i until δ goes back to zero.

If K_i is too low, it has little effect; but if it's too high then it can cause oscillation.

Differential gain

If K_p is the gain due to the present value of δ and K_i is the gain due to the past value of δ, K_d is the gain due to the *future* value of δ.

$$U_d = K_d \frac{d\delta}{dt}$$

By looking at the time rate of change in δ, K_d allows the system to correct for rapidly-changing values before they cause overshoot and oscillation.

The disadvantage of K_d is noise sensitivity: high-frequency noise in δ has a large derivative, so large values of K_d can cause noise-induced oscillation.

PID

PID stands for Proportional, Integral, Differential control.

$$U = U_p + U_i + U_d$$

With appropriately-chosen values of K_p, K_i, and K_d the PID controller can bring the system to the setpoint quickly and hold it at that value with minimal variation.

The traditional way of doing a PID controller is to use a set of 4 op-amps: one each for proportional, integral, and differential gain, with the fourth configured as a summing amplifier so that the output is the sum of the P, I, and D, components. For high-speed applications, this may still be the best way of building a controller. For the relatively low speeds required for thermal control, however, it may be more convenient to use a digital

implementation of the PID algorithm. In the digital case, we replace the integral with a sum and the derivative with a difference:

$$U_j = K_p \delta_j + K_i \sum_{\ell=0}^{j} \delta_\ell \Delta t + K_d \frac{\delta_j - \delta_{j-1}}{\Delta t}$$

We can use the same idea to express the *previous* value of U:

$$U_{j-1} = K_p \delta_{j-1} + K_i \left(\sum_{\ell=0}^{j} \delta_\ell \Delta t - \delta_j \Delta t \right) + K_d \frac{\delta_{j-1} - \delta_{j-2}}{\Delta t}$$

We can then use Euler's method to approximate the next value of U:

$$\begin{aligned} U_{j+1} &= U_j + \frac{dU}{dt}\Delta t \approx U_j + \left(\frac{U_j - U_{j-1}}{\Delta t} \right) \Delta t \\ &= \dots \\ &= U_j + \left(K_p + K_i \Delta t + \frac{K_d}{\Delta t} \right) \delta_j + \left(-K_p - 2\frac{K_d}{\Delta t} \right) \delta_{j-1} + \left(\frac{K_d}{\Delta t} \right) \delta_{j-2} \end{aligned}$$

This trick allows us to manage a digital PID efficiently without keeping track of all previous values of δ, as long as the values of δ are measured at a constant time interval Δt.

Apparatus

Our apparatus for this exercise consists of three main components:

Temperature block and heat sink The small aluminum rectangle on the top of this component is the one for which we will be controlling the temperature. A small thermistor is embedded in this block at the blue/white wire, which should be attached to the thermistor driver. Between this block and the main heat sink there is a Peltier device: this device converts current to a difference in temperature. Current one way will pump heat from top to bottom, reversing the direction of current reverses the direction of the heat pump. This Peltier device should be driven with an external bipolar power supply via the Voltage-Controlled Bidirectional Current Driver circuit.

Thermistor driver The thermistor driver box provides a constant current to the thermistor which measures the temperature of the aluminum block. The output of the driver box is the resulting voltage across

the thermistor. If you send this output to LabVIEW as an "Iex thermistor", with the appropriate {A,B,C} values, LabVIEW will convert directly to temperature.

$$\begin{aligned} A &= 1.11253 \times 10^{-3} \\ B &= 2.34711 \times 10^{-4} \\ C &= 8.56943 \times 10^{-8} \\ I &= 100\mu A \end{aligned}$$

Current amplifier The output current capacity of a LabVIEW board is insufficient to drive the Peltier device directly. We need a current-amplifier circuit: one that takes the voltage from a high-Z input and converts it to the same voltage at low Z. The circuit shown in figure 1 will do the job: take the time to figure out how it works, because it's a pretty useful technique. The portion of the circuit that is outlined with a dashed line is built for you already on the black heatsink.

Blue	Transistor base input
Red	+V input (8V 2A max)
Orange, White	Peltier (red) input
Black	-V input (-8V, 2A max)
Grey	Ground

Experiment

Write a LabVIEW program that allows you to control the temperature of the aluminum block. The program should have a slider control that allows the user to set the temperature. It should also display the actual temperature, preferably in graphical form so one can see how it behaves over time.

You may be tempted to do this with an Arduino rather than LabVIEW. This is also acceptable, but keep in mind that the analog output for Arduino is a one-sided PWM output and the current driver circuit requires a bipolar analog signal so that it can drive the current smoothly in either direction. It's not impossible to control it with a microcontroller, but you'll need a bipolar DAC circuit between the Arduino and the current driver.

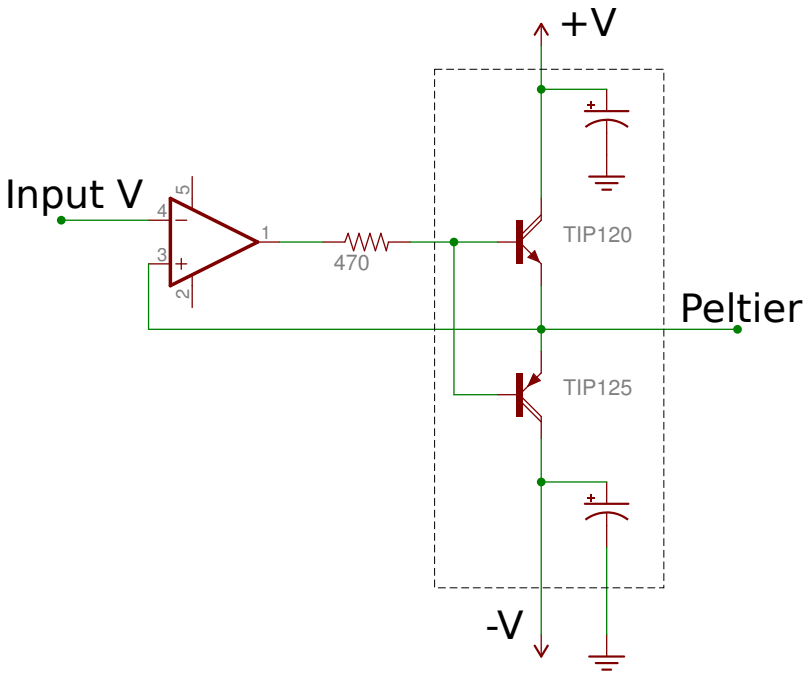

Figure 1: Voltage-Controlled Bidirectional Current Driver