The Best Type of Temperature Controller to Use in the Physics 240 Contest: An Empirical Analysis of Three Common Controller Types for Fastest Performance and Most Robustness

INTRODUCTION

Temperature is a critical parameter to almost every process in daily life. Proper regulation of temperature is necessary to sustain life, comfort, and to maintain normal operation of equipment. Different situations have different precision requirements for temperature, and different types of temperature controllers are appropriate for different situations. For example, a house's central air conditioning system uses a simpler controller than a biological incubator that keeps temperature precise to within a fraction of one degree for temperature-sensitive chemical reactions, which precision a home air conditioning system could never accomplish. This paper is about the first experiment to determine which temperature controller type is best suited to winning a contest in a particular physics class by completing a certain heating cycle fastest.

In Physics 240, a lab class at BYU required for physics majors and minors, one of the projects is to design a computer-controlled temperature controller. The project's goal is to design a controller that can precisely control the temperature of a cryostat, which is a cylindrical metal apparatus roughly six inches long (see **Figure 1** on the next page).

More specifically, the students design a system that heats the cryostat to two different target temperatures, one at a time, and holds each temperature constant within a specified tolerance range and duration. This is the specific heating cycle the students must complete: the cryostat must begin below 50°C and warm up to 100°C and hold that temperature constant with a tolerance of half a degree (±0.5C°) for 2½ minutes. If the

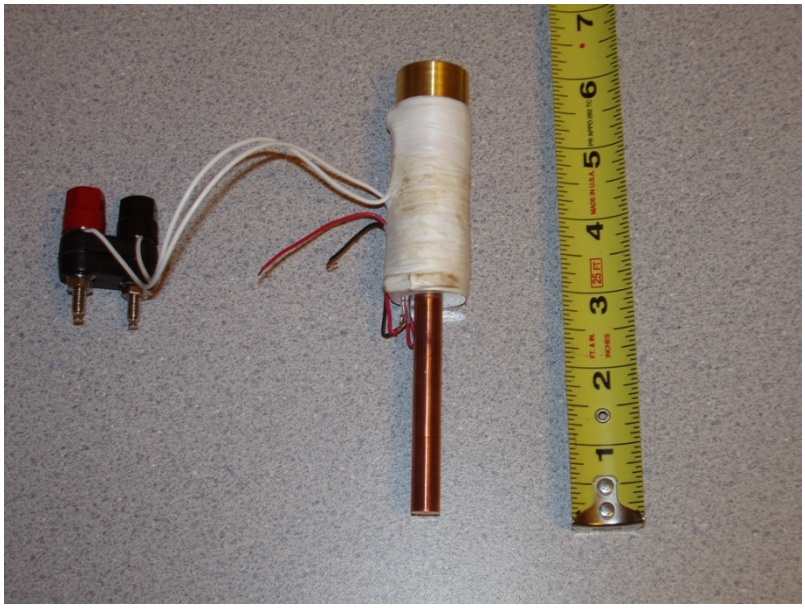
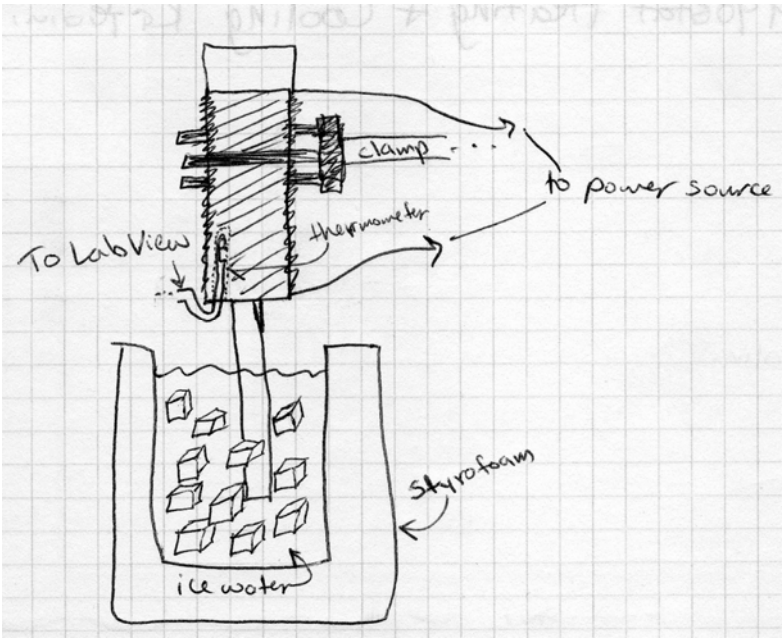


Figure 1 - A cryostat heater block used in Physics 240

temperature drifts outside the tolerance zone, i.e. the narrow temperature band from 99.5°C to 100.5°C, the 2½ minute timer resets and begins again at zero. After holding the temperature steady for 2½ minutes, the controller must then allow the cryostat to cool to 70°C and hold it there with the same tolerance for the same duration. The system must be completely autonomous once the process begins, and if it accomplishes the heating cycle, full credit is given for the assignment no matter how long the cycle took. To motivate the students' best work, a contest is held to see whose system can complete the cycle fastest, with a prize awarded to the winner.

To help the students with design ideas for their controllers, they are given, but not limited to, three possible ways to do it: 1 – the "Brute Force" method; 2 – using a temperature model; and 3, using a "P-I-D" controller. I flesh out the details of these methods in the Research Procedures section. The contest is held every semester the class is offered, and each technique has produced winners at different times.

The controller is designed and run in a computer program called LabView. The computer is connected to external hardware which allows it (the computer) to interface with a power supply. The power supply, in turn, is connected to the cryostat. The cryostat is suspended above an ice water bath with its copper cooling rod submerged in the cold water to provide a path for heat to leave it (the cryostat) more quickly than in air alone (see **Figure 2**).



The LabView program, when running the temperature controller, sends instructions to the external hardware which depend on the controller's particular design concept. Those instructions are translated into voltage levels which the power supply reads and uses to determine how much electrical current to feed into the cryostat. In short, the temperature controller in LabView tells the power supply how much "juice" to pump into the cryostat, which heats it up. The cryostat has a built-in temperature-sensing diode which connects to the computer's external hardware, giving the controller access to the current

temperature of the cryostat at any given moment, which is necessary for the controller's correct operation and for the lab instructor to verify correct operation. That diode, which is essentially an electric thermometer, is the most critical part of the controller because without it the controller would have no way to know if it was properly reaching its temperature targets. Making the temperature information from the cryostat available to the controller is called *feedback* (see **Figure 3**), and all three controller types require feedback to work properly. The functional differences between the different controller types are in how they use the temperature feedback to determine how much power to output to the cryostat. This report focuses on how the three different controllers handle that fed-back temperature information to make the cryostat complete the heating cycle as quickly as possible.

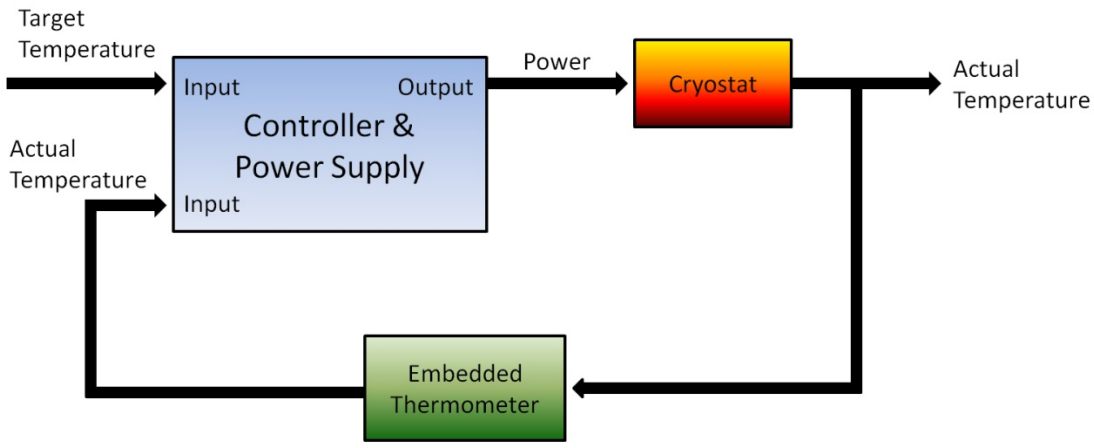


Figure 3 - Closed-loop feedback control

To summarize, the computer tells the power supply how to heat the cryostat according to the controller's instructions, the computer senses the cryostat's temperature, and then it's up to the student to design a controller that can tell the power supply exactly how to correctly heat the cryostat to complete the previously outlined heating cycle.

RESEARCH PROCEDURES & EXPERIMENTAL FINDINGS

For my experiment, I designed and built a controller based on each of the three methods mentioned previously, for a total of three different controllers. I conducted the experiment in a lab classroom in the Eyring Science Center. No matter what method was used to control the temperature, one basic fact was always true: as heat was pumped into the cryostat, heat also left it through the physical interface with the ice water bath. Therefore to hold the cryostat at a target temperature, that temperature was first reached, and then equilibrium was established so that heat entered the cryostat at the same rate it left. When that particular set of conditions was achieved, the "steady state," the temperature stayed constant. In other words, there was a "sweet spot" voltage level, a critical voltage, which kept the temperature at its target level, assuming the physical setup of the cryostat wasn't changed or disturbed during the heating cycle. In the following sections I detail the process of designing each of the three controller types and my findings as I worked with them.

1. BRUTE FORCE CONTROLLER

This was essentially a guess-and-check approach where I programmed the controller to simply output specific voltages at predetermined temperature levels and hold them (the voltages) there until the temperature crossed the next threshold. The aforementioned "sweet spot" voltages logically existed, so I predicted that if I chose the exact voltages and temperature thresholds correctly, the cryostat would theoretically reach the target temperatures and hold them steady, and complete the heating cycle successfully.

First I attempted to locate the critical voltage to hold the cryostat at 100°C. I did so by applying maximum voltage (9.5 volts) to heat it to the neighborhood of 100°C as fast as possible, and then by trial and error I adjusted the control voltage until it seemed to stay as close to 100°C as possible. But with the cryostat that hot for any length of time, the ice immediately around the cooling rod in the ice bath melted in a slowly expanding circle; the water was still icy, but it was a few degrees warmer in the cooling rod's immediate proximity. This meant that the critical voltage did not stay constant as I had assumed it would, but gradually decreased instead as the ice slowly melted around the cooling rod in the cooling bath; as the heat flow from the cryostat into the ice water slowed due to the slightly-warmer water, the influx of heat to the cryostat also had to be reduced to avoid overshooting the tolerance zone and losing precious seconds in the contest. Therefore I decided to try making the system nudge the temperature back and forth within the tolerance zone by applying a certain voltage if the temperature dropped below 100°C, and a certain lower voltage if the temperature rose above 100°C, analogous to rolling a ball along a U-shaped channel. It worked. After several iterations of experimentation, I found that by applying 3 volts when below 100°C and 2.2 volts when above, the temperature gently oscillated about 100°C without leaving the tolerance zone (see **Figure 4** on the next page); below 100°C the 3 volts heated it, but slowly enough that it started cooling again before exceeding 100.5°C, and above 100°C the 2.2 volts prevented it from cooling too quickly for the 3 volts below 100°C to "catch" it before dipping below 99.5°C.

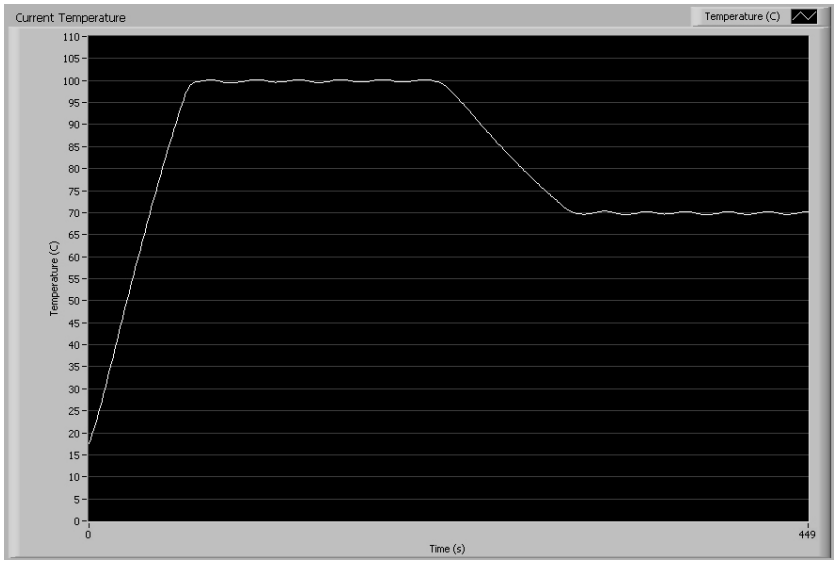


Figure 4 - Heating cycle completed with a brute force temperature controller

That all looked promising, but when I tied it all together to see if the cryostat could properly heat to 100°C from a starting temperature below 50°C, I noticed the temperature severely overshot 100°C after the initial warm-up. I wanted to avoid overshoot because of the time required to cool the system back down into the tolerance zone. After further trial and error, I found that during the initial warm-up, by abruptly cutting the power to zero at 94°C for eight seconds the overshoot peaked the temperature right inside the tolerance zone, after which the smaller voltages took over to nudge the temperature back and forth about its target temperature of 100°C, and successfully kept it there for the required 2½ minutes.

My next challenge was to allow the system to cool to 70°C and hold that temperature the same way. Using the same "nudging" technique with lower voltages appropriate to the new cooler target temperature, the temperature stabilized inside the tolerance zone at 70°C except for one minor problem: just like the overshoot problem in the initial warmup to 100°C, at first the system *under*shot 70°C as it cooled down to that

new target temperature. The controller could heat the system quickly simply by applying enough power, but it couldn't cool the system. The cryostat's only way to cool itself was to radiate and conduct its heat to the ambient air and into the ice bath. The natural cooling rate was slower than the full-power heating rate but too fast for the "nudging" voltages at 70°C to catch the temperature inside the tolerance zone on its way down from 100°C before dropping below 69.5°C. To prevent this, I found that applying a 1-second burst of full power at 71°C, on its way down from 100°C, effectively prevented the undershoot and landed the temperature smoothly into the tolerance zone at 70°C, after which the back-and-forth nudging voltages held the temperature inside the zone long enough to complete the heating cycle. My best time using this controller was 7 minutes, 6 seconds to complete the cycle, and that result is also shown in Figure 3 on the previous page.

The last test for the brute force method was to test its robustness, or ability to work in different thermal environments. I tested that by raising the cryostat an inch out of the ice bath so its cooling rod was less submerged. Using the exact same voltage settings that worked before, the controller failed to complete even the first phase of the heating cycle. It heated the cryostat to 100°C, but the cryostat could not cool quickly enough to stay in the tolerance zone; the 3 volts below 100°C now heated it too fast to stay in the tolerance zone, and it overshot 100.5°C on every oscillation, resetting the 2½ minute timer every time (see **Figure 5** on the next page).

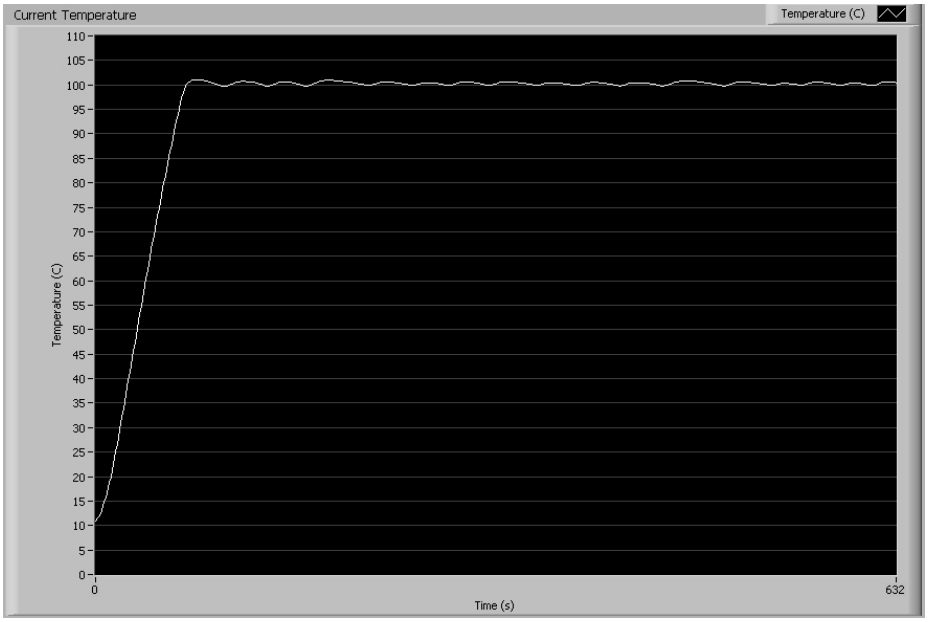


Figure 5 - Brute force controller unable to achieve a steady target temperature with the cryostat raised partially out of the ice bath

2. TEMPERATURE-MODEL-BASED CONTROLLER

My second controller used a temperature model, which is a mathematical equation that predicts the temperature of the cryostat based on the power fed into it. Without going into deep mathematical analysis, this is the equation used:

$$P(t) = C\alpha(T_{desired} - T) + \gamma(T - T_{\infty}) - \frac{C}{\gamma}\frac{dP(t)}{dt}$$

Equation 1 - Temperature model relating cryostat temperature to power input

The variables in this equation represent certain thermal properties of the cryostat. *P(t)* is the power applied to the cryostat as a function of time; lower-case *t* is time; capital *T* is the cryostat's current temperature; $T_{desired}$ is the target temperature (100°C then 70°C); T_{∞} means "temperature at infinite time," or the final temperature to which the cryostat would settle if allowed to cool in the ice bath for a long time; α (Greek "alpha") represents how quickly I want the temperature to approach the target temperature; *C* is

the cryostat's *heat capacity* that relates its heat energy to its actual temperature; γ (Greek "gamma") is the cooling coefficient that depends on how fast the cryostat naturally cools when its power input in zero; and *dP(t)/dt* is the power's *derivative*, or how much the applied power changes from one second to the next. Of these variables, C, γ, and T_∞ are constants that depend on the cryostat's physical characteristics (shape, dimensions, materials, geometry of setup) and its thermal environment. $T_{desired}$ and α are user-selectable parameters. Lower-case t is the flow of time which is obviously not controllable.

The three variables C, γ, and T_∞ required additional experimentation to determine their values. To do that, I heated the cryostat from 30°C to 120°C with a known power input, and then allowed it to cool from 120°C to roughly 10°C while recording the temperature and time data to a text file the whole time. I brought the heating and cooling portions of that data separately into another computer program called Kaleidagraph, which analyzed the data mathematically to fit curves through the points and generate values for the constants C, γ, and T_∞ (see **Figures 6** and **7**).

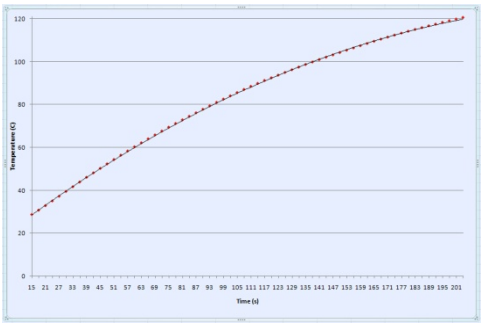


Figure 6 - Heating data used to find parameter values in temperature model

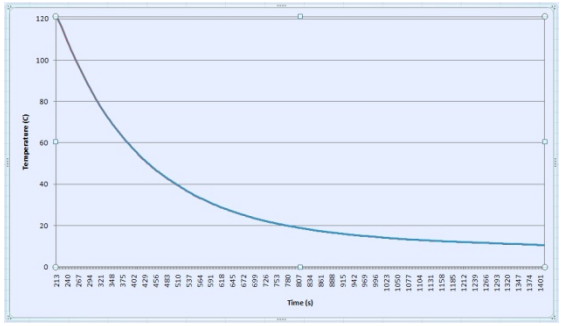


Figure 7 - Cooling data used to find parameter values in temperature model

That thermal characterization process was the most nonintuitive part of the entire experiment because the results were messy and difficult to interpret. Unsurprisingly, when I implemented the power equation into LabView as my controller, using the values I had calculated for those variables, it did not work at all. The control voltage alternated between zero and "NaN," which is computerspeak for "not a number," or in other words, infinity. That clearly was not what I wanted to see, and after several fruitless attempts to debug it, I finally got it to work surprisingly well by completely removing the derivative term, and by making *random guesses* for the numerical values to assign to the three constants C, γ, and T_∞, which were nowhere near the values I had computed through my careful and laborious thermal characterization process. In the end, I had no idea how or why this controller worked. The amazing thing was that it not only somehow got the temperature to each target, it held the temperatures on target *almost perfectly*, without any of the oscillations observed in the brute force method (see **Figure 8**).

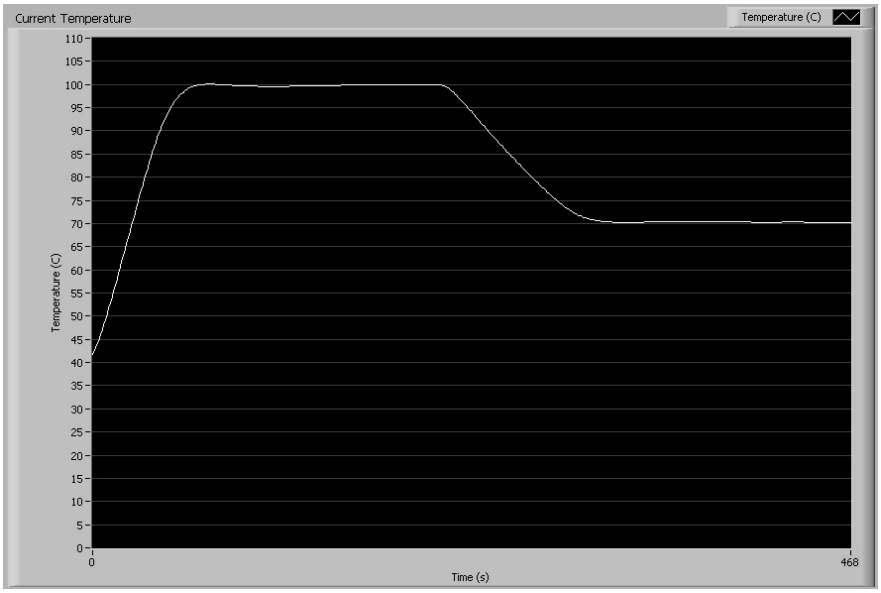


Figure 8 - Heating cycle completed with a controller based on a temperature model

It might have been the fastest controller if not for the "slowdowns" as it approached each target temperature that I couldn't eliminate no matter how I tuned the C, γ, and T_∞ coefficients. The slowdowns smoothed the approaches to 100°C and 70°C at the expense of taking longer to reach those target temperatures (compare **Figures 8** and **4**). Nevertheless, it did complete the heating cycle, although 34 seconds slower than the brute force method (7 minutes 40 seconds) due to those slowdowns. Unfortunately, it also failed when I changed the cryostat's thermal environment by raising it partially out of the ice bath, proving that the temperature model controller type, like the brute force method, also lacks robustness and works for only one set of thermal conditions.

3. PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) CONTROLLER

My third approach used a "P-I-D" controller. P-I-D stands for "Proportional-Integral-Derivative." Unlike the previous two controller types, a PID controller focuses on nothing but the difference between the actual temperature and the target temperature, and it does everything it can to drive that difference to zero, and it does so in a way much more exotic than either the brute force or temperature model controllers. Although the historical development of the PID controller required extremely intense calculus and algebra to fully develop, its resulting manner of operation is actually quite simple and can be explained with basic algebra and basic graphs to illustrate. **Figure 9** on the next page shows the conceptual workings of the PID controller.

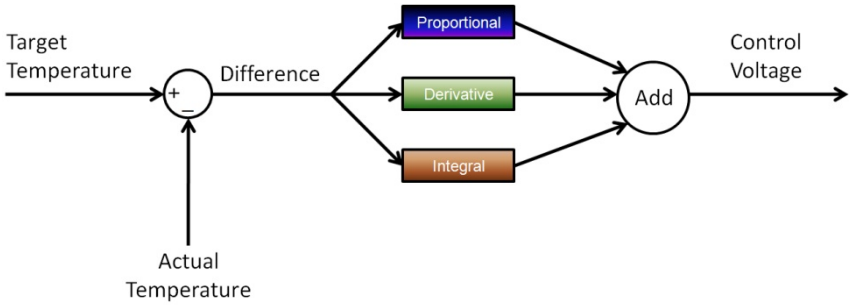


Figure 9 - PID Controller

It may not make much sense at first glance, but hopefully my explanation will clear it up a bit. The three colored blocks make up the heart of the PID controller, and they each merit their own explanation. The difference signal is constructed by subtracting the actual temperature from the target temperature. If the actual temperature matches the target temperature, the difference is zero, which is the goal of the PID controller. The best way to think of it is to visualize that each block receives the same difference signal and does its own operation on that signal, independent of the other two blocks, to determine its individual share of the control voltage (control voltage is the voltage that tells the power supply how much power to pump into the cryostat). The three blocks' control-voltage contributions are added together, and their total sum becomes the control voltage sent to the power supply.

A. Proportional

The Proportional block's share of the control voltage is exactly what its name says: its contribution is directly proportional to the **temperature difference**; if the actual temperature is less than the target temperature, the difference is positive and the

Proportional block adds to the control voltage, and vice versa. **Figure 10** shows a simulated graph of how a PID-controlled temperature curve might look, and the same simulation is used in **Figures 11** and **12** as well in the Derivative and Integral discussions.

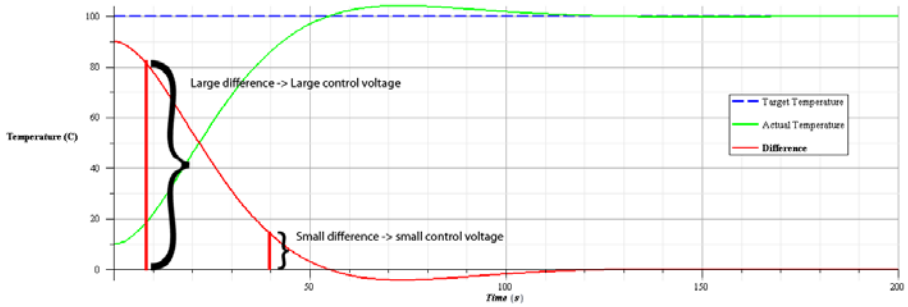


Figure 10 - Proportional control (simulated)

At the beginning of the heating cycle, the actual temperature was much less than the target temperature of 100°C, which meant that the temperature difference started out large, which generated a large control-voltage contribution from the Proportional block. As the cryostat warmed, the temperature difference shrunk toward zero, which reduced the Proportional control voltage accordingly. In the steady state, the difference was zero, which meant the Proportional block's voltage contribution was also zero. One important point to make is that on its own, the Proportional block cannot hold the temperature steady at its target, because if the temperature difference is zero, the Proportional voltage is also zero, which would allow the cryostat to cool in the ice bath. Because of this, the temperature would eventually stabilize to some steady-state value several degrees below the target temperature. That gap is closed with the help of the Integral block. Also during the initial warmup, if used on its own, the Proportional block may generate a temperature overshoot beyond the tolerance zone and oscillate before gradually settling

in to a steady-state temperature. That overshoot and oscillation would cost precious time in a contest, but is prevented with Derivative control.

B. Derivative

Have no fear, no nasty calculus skills required. The graph will show everything. A derivative, simply defined in plain English, is a *rate of change*. For example, when driving a car, the car's position changes with time. The rate of that position-change-in-time is called speed. If the car's position changes at a high rate, the car's speed is considered fast. That means the speedometer really shows the car's derivative, because it indicates the car's rate of position change. In math the derivative simply refers to the slope of a graph. In the case of the cryostat in this experiment, the derivative refers to how much the temperature difference changes in one second (because temperature samples were taken once every second) between the target and actual temperatures. **Figure 11** illustrates this with arrows drawn along the red temperature-difference plot.

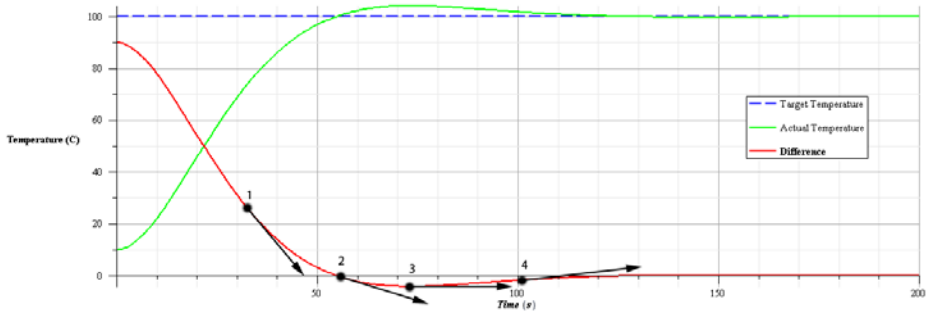


Figure 11 - Derivative control (simulated)

1) Steep downhill slope → Large voltage subtraction from total control voltage
2) Shallow downhill slope → Small voltage subtraction from total control voltage
3) Horizontal slope → No contribution or subtraction from total control voltage
4) Shallow uphill slope → Small voltage contribution to total control voltage

If the temperature gap closed quickly, the temperature difference decreased quickly and had a large negative slope (derivative), indicated by the steep downward-pointing arrow. The Derivative block in the PID controller reacted to this negative derivative by subtracting from the control voltage, which prevents overshoot if tuned properly; its voltage contribution was proportional to the **slope** of the difference graph.

The only reason to use Derivative control was to prevent the overshoot and oscillation discussed previously, because in the steady state the cryostat's actual temperature was held at the some steady-state temperature (hopefully the target temperature), which meant the cryostat's temperature did not change and therefore its rate of change, or derivative, was zero, and therefore the Derivative block contributed nothing to the control voltage in the steady state. I wanted zero temperature difference in the steady state; that meant the contribution from the Proportional block was also zero in the steady state. So with no control-voltage contributions in the steady state from either the Proportional or Derivative blocks, holding the temperature at its target value required another source of voltage, which is where the Integral block came in.

C. Integral

No ugly math here either, just pay attention to the graph and description. The Integral block holds the temperature right at its target value when the Proportional and Derivative blocks lose their influence over the control voltage in the steady state. "Integral" refers to the area under the curve on a graph (see **Figure 12** on the next page).

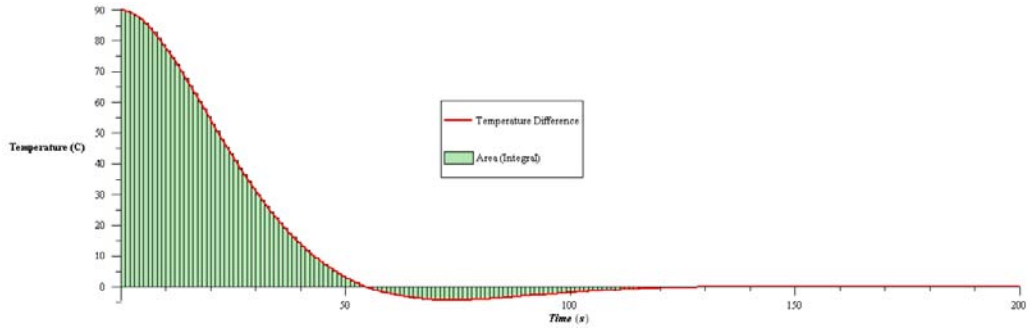


Figure 12 - Integral control (simulated)

The difference between the cryostat's target and actual temperatures decreased and hopefully settled out to zero when the temperature reached the steady state. During that process, the controller recorded the temperature difference once every second and kept a running sum (integral) of temperature values as recorded every second. In Figure 12, the integral sum is represented as the green shaded area between the graph and the time axis. As time flowed, if the temperature difference was positive, the total area under the difference curve increased and added to the integral sum until the difference graph reached or dropped below zero. The green shaded portion *below* the time axis subtracted from the integral sum until the temperature difference reached zero again. In the steady state, the integral sum stayed essentially constant if the temperature difference was zero, since there was no temperature difference to add to or subtract from it. That was how the Integral block held the temperature at its target value; its control voltage was proportional to the **integral sum**. If the current temperature was below or dropped below its target, the integral sum accumulated further and the Integral voltage proportionally increased, closing the temperature gap; if the temperature was above its target, the integral sum decreased due to the negative temperature difference, which correspondingly decreased the Integral voltage to allow the cryostat to cool towards its target temperature. In this

way, Integral control is self-correcting, similar to the brute force method but much, much smoother and more "intelligent."

With that background on PID control I can now discuss the performance of my PID controller. Once the basic framework of the PID controller was built, I had to "tune" it to achieve optimum performance. The P, I, and D parts of a PID controller each generated their respective contributions to the cryostat control voltage in proportion to three different aspects of the temperature-difference signal they received; the 'P' part responded proportional to the actual value of the temperature difference, the 'D' part responded proportional to how quickly the temperature difference changed, and the 'I' part responded proportional to the running sum of the temperature difference as recorded at one-second intervals. Those three proportions are represented by three tunable coefficients, K_p, K_d, and K_i, numbers that tell each part how strongly to respond to its respective temperature-difference signal information. By tuning those three numbers separately by trial-and-error, I achieved the best possible speed for my PID controller in

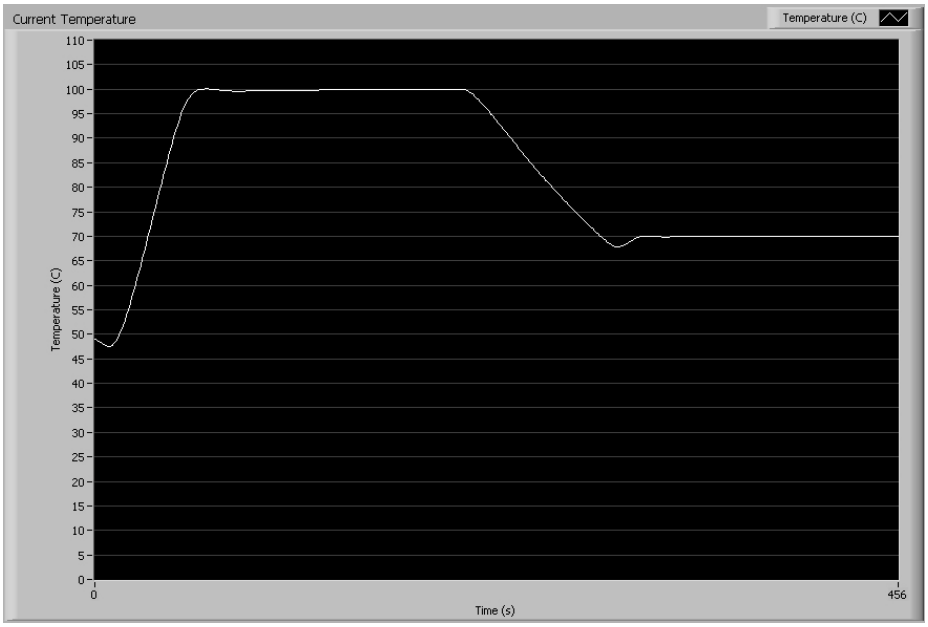


Figure 13 - Heating cycle completed with a PID Controller

this contest, and it held the temperatures at their targets within .02 degrees, 25 times more precisely than the contest rules required. My PID controller's fastest completion time for the heating cycle was 7 minutes, 22 seconds, slower than the brute force method but faster than the temperature model method (see **Figure 13** on the previous page). It would have been faster if not for the undershoot at 70°C, but that was unpreventable because the PID controller could not usefully influence the cryostat's temperature until the actual temperature dropped below the target temperature; it could only heat the cryostat, not cool it. Integral control requires a positive temperature difference to generate an integral sum so that the Integral control block can respond to it with an appropriate control voltage; therefore, for the 70°C phase of the heating cycle, my controller left the Integrator off until the cryostat cooled down to 71°C from 100°C, and the undershoot at 70°C was necessary to gather that positive area under the difference graph for the Integral block to act on.

After completing the heating cycle according to the contest rules, I tested the

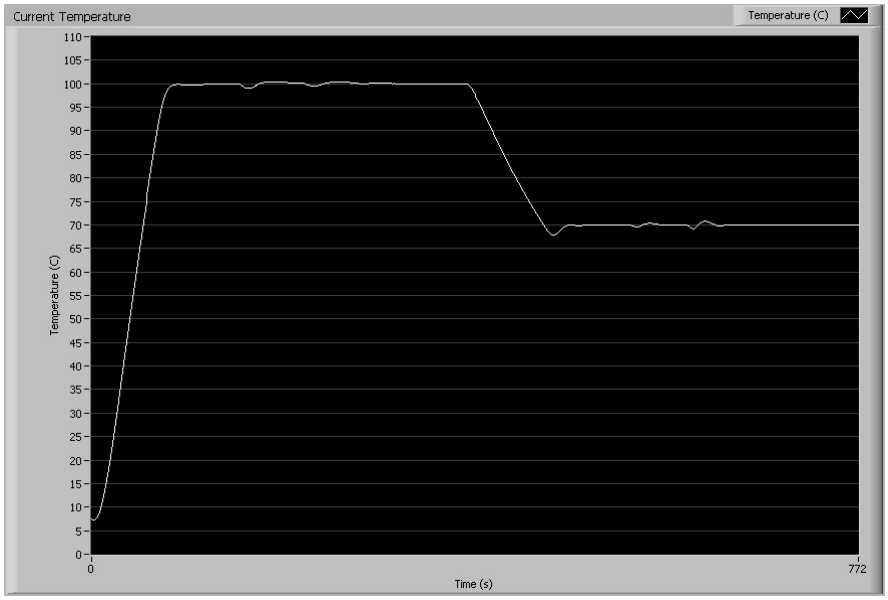


Figure 14 - PID controller recovers from ice disturbance around cooling rod

controller's robustness by testing its response to various disturbances and changes in the cryostat's setup. As described in the brute force section, the hot cryostat slowly melted the ice around the submerged cooling rod. The PID controller automatically compensated for that by the nature of its self-correcting Integral control. I disturbed it by pushing fresh ice to the cooling rod all around it, increasing the heat flow rate out of the cryostat. Its temperature did drop, but the controller smoothly adjusted and returned the temperature back to its target with the same excellent precision as before (see **Figure 14** on the previous page).

I also changed the cryostat's elevation in the ice bath during the heating cycle (see **Figure 15**); during the 100°C phase I raised it an inch out of the ice, half the cooling rod's original submerged depth, and then during the 70°C phase I lowered it back to its original depth.

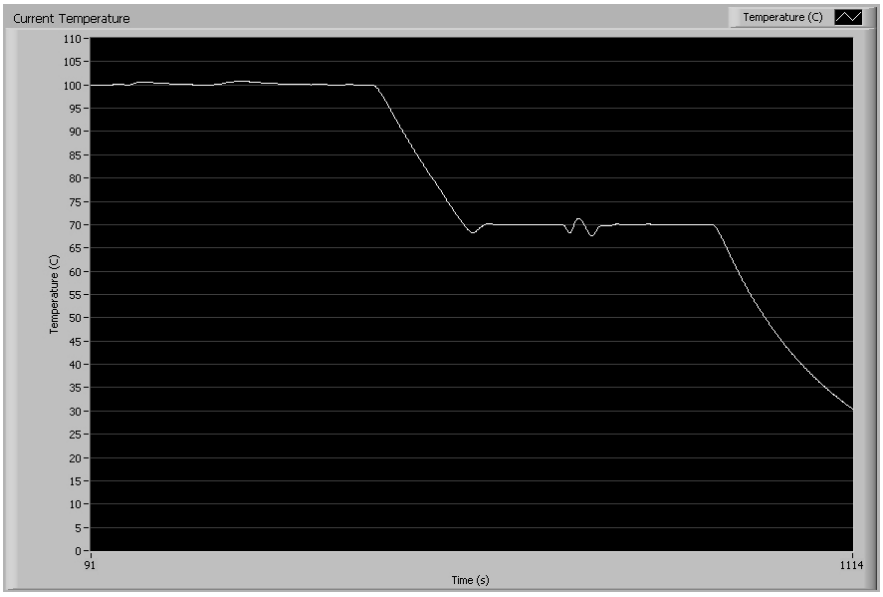


Figure 15 - PID Controller recovers from elevation changes in its ice bath

Figure 15 shows the resulting disturbance ripples during the two phases. Notice that in the 100°C phase, the ripples start by increasing above 100°C when I raised the cryostat out of the ice bath, and in the 70°C phase the ripples start by dipping below 70°C as I lowered it back down into the ice. Notice the complete and successful recovery from both disturbances.

However, the PID controller performed most impressively when I removed the cryostat completely from the ice bath and left it suspended in the air with no cold-water interface to dissipate heat quickly (see **Figure 16**); it took about twice as long to complete the heating cycle, and it took longer to settle in to the target temperatures, but it still completed the heating cycle despite my prediction that removal from the ice bath must surely be more than the controller could handle.

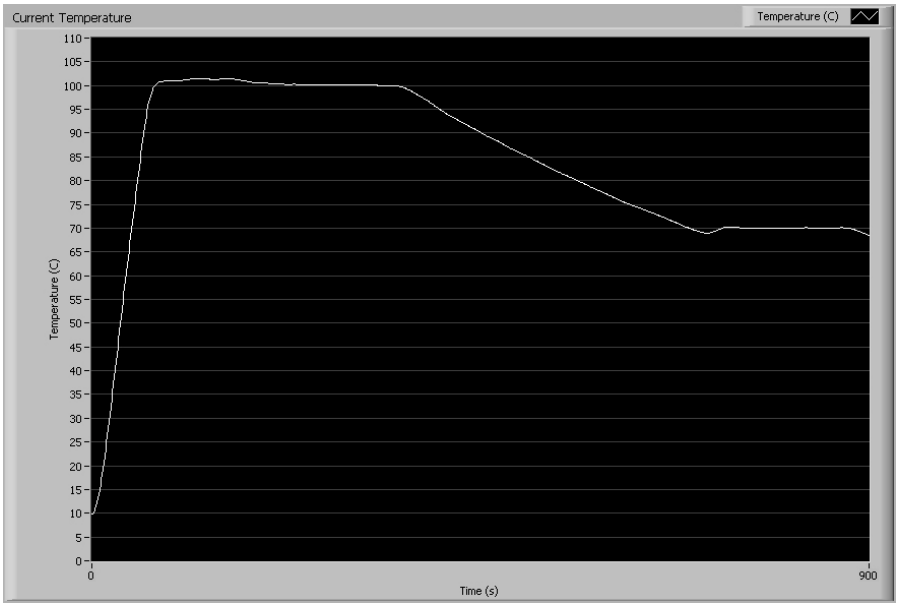


Figure 16 - PID controller completes the heating cycle without the benefit of the ice bath

EXPERIMENTAL RESULTS AND RECOMMENDATIONS

The brute force controller was the best controller for contest purposes because it finished the heating cycle faster than the other methods, and it was easier to design and implement than the other two methods. It was easy to mentally follow its operation, but it had the disadvantage of only working in the exact thermal conditions it was design for, meaning it was not robust.

The PID controller did not have the fastest time, but due to the self-correcting behavior of the Integral part of the controller, it was extremely robust and adapted very well to different thermal environments, and it also recovered well from thermal disturbances, even to the extreme of removing the ice bath from the experiment.

The temperature-model controller was the slowest to complete the heating cycle, and it was definitely the most difficult to design because of the additional thermal experimentation and computer analysis required, and also because of its complete failure to work after the meticulous design process, despite completing the heating cycle after making literally random design changes. It also was not robust to thermal disturbances.

The most interesting thing is that each controller in this experiment was still faster than my fastest time when I actually took the class, and I won the contest back then. Table 1 summarizes my results.

Controller Type	Fastest Time
Brute Force	7 min, 6 seconds
PID	7 min, 22 seconds
Temperature Model	7 min, 40 seconds
My winning contest time from last semester (Fall 2008)	7 min, 59 seconds PID controller

Table 1 - Performance of each controller type

CONCLUSION

In this report I described three different ways to control the temperature of a cryostat used in a thermal contest for Physics 240: a brute force controller, a controller based on a temperature model, and a PID controller. I described how each controller worked and discussed each controller's performance in terms of robustness and how much time each one took to complete the heating cycle. I also compared the performance of the three controllers I built for this experiment to my winning contest time and controller type from when I took Physics 240 last semester.

BIBLIOGRAPHY

Berlin / Heidelberg: Springer. (2009). PID control theory. Retrieved February 17, 2009, from PAControl Web site: http://www.pacontrol.com/PID.html

Dubay, R. (2004). A Temperature-dependent Adaptive Controller, part I: Controller Methodology and Open-loop Testing. *Polymer Engineering and Science. 44*, 1925-1933.

Lewandowski, M. (1977). A Precision Temperature Controller. *Journal of Physics E: Scientific Instrumentation*. *10*, 903-905.

Williams, C. D. H. (2001). School of physics. Retrieved February 17, 2009, from Feedback and Temperature Control Web site: http://newton.ex.ac.uk/teaching/CDHW/Feedback/

Alvarez-Ramirez, J., & Alvarez, J. (2005). Robust temperature control for batch chemical reactors. *Chemical Engineering Science. 60,* 7108-7120