Week 1 Assignment

Phys 13H / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

1. Read Chapter 1 in the textbook (Essick).
2. Figure out how to start LabVIEW on the lab computers.
3. Write a virtual instrument (VI) that converts a Celsius temperature to Fahrenheit.
4. Run the VI you wrote, and convert 37 °C to Fahrenheit.
5. Write a VI that lets you enter a number x, and which then displays the value of $2x^2 - 3x + 2$.
6. Write a VI that lets you enter two numbers x and y, then displays the values of $x-y$ and x/y. If $y = 0$, a red LED should light up indicating "Division by Zero". Set the controls and indicators so that 4 digits are shown following the decimal points.

Please note:

- When using the lab computers, **you must back up your work on a personal flash drive or via email!** Files saved on the lab computers may be wiped out at any time by software updates.
- You are free to work on your assignments on any computer you want, however **the code you turn in must run on the lab computers.** The TA will not spend time trying to port your code if it doesn't run.
- Please do not install software on, or otherwise alter, the lab computers. As mentioned above, such changes would be wiped out by software updates. If there is a change you think would be useful, mention it to the TA or instructor.
- Please try to use the same computer each time you come to lab. This will help us keep things organized. If someone from the other section is using your computer, you can use one of the others, but it would be best to transfer your code back to your machine when you get a chance.

Week 2 Assignment

Phys 13H / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

1. Read sections 3.12, 3.13, and 3.14 in the textbook and/or Chapter 7 of LabVIEW Fundamentals (see "Handouts" section of course web page) to learn about sub-VIs.

2. Write a VI that displays the average of four numbers that the user inputs.

3. Use your averaging VI from above and the one from last week that computes $2x^2 - 3x + 2$ **as sub VIs** in a program that averages four numbers input by the user and computes $2x^2 - 3x + 2$ using the average as x. It should do this if a switch is set to "on," but should display zero if the switch is set to "off".

4. Write a VI that monitors a numerical control to see if the user has changed the number being entered. Whenever the user does change the number, your VI should make a normally green LED turn red for 0.5 seconds. Be sure your VI monitors the input frequently, not just every 0.5 seconds.

Please remember to include any necessary sub-VIs in this week's folder on your flash drive!

Week 3 Assignment

Phys 13H / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

1. Read Chapter 2 in the textbook.

2. Write a VI that generates a random number between 0.0 and 10.0 once every 0.5 seconds, and adds the numbers together until the running sum reaches or exceeds 100.0. Once the sum reaches or exceeds 100.0, your program should continue generating random numbers, but it should now subtract each new number from the sum, until the sum reaches 0.0 or less. It should then begin adding them again, and so on, *ad infinitum*. The running sum should be displayed numerically and by means of a waveform chart.

Week 4 Assignment

Phys 13H / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

1. Read Chapter 4 in the textbook.

2. Read the "Viewing PostScript Files" handout from the course webpage.

3. Write a VI that generates a random number between 0.0 and 1.0 every 0.2 seconds, and computes the average of the last n numbers. The VI should be written so that the number n, between 1 and 10,000, can be chosen by the user. It is ok if the user has to stop the program to change n.

   This program computes what is called a *running average*. For example, if $n = 10$ and you have generated 67 random numbers so far, you want to calculate the average of the 58^{th} through the 67^{th} numbers. Both the random numbers and the running average should be displayed on waveform charts or graphs. Your program should keep generating random numbers until you stop it.

   During the initial period before n numbers have been generated, your program should display the average of the numbers that have. That way the user gets an idea immediately of what is going on, without having to wait for n numbers to be generated (for example, 33 minutes for 10,000 numbers).

   The steady stream of random numbers represents a noisy signal, and your program is a type of digital filter to reduce the noise. This type of filter is called a *boxcar averager*. Be sure to keep a copy of this program because you will use it later with your servo program to control temperature.

4. Following the instructions given in class, strip about 5 mm of both ends of two short (about 5 cm) pieces of 22 AWG wire and insert them in the Analog Input 0 and Analog Ground screw terminals on the USB-6009 data acquisition device.

5. Write a VI to make continuous measurements of Analog Input Channel 0 and display them on a waveform graph. You may find it helpful to consult Chapter 4 of the *Getting Started with LabVIEW* document on the course web page in addition to the textbook. Some things have changed since that document was written, so the interface will differ slightly from what is shown. Set the `Terminal Configuration` to `RSE` (single-ended input), the `Custom Scaling` to `<No Scale>`, `Samples to Read` to `1k`, and the `Rate(Hz)` to `10k`.

If you touch the Analog Input 0 wire while the computer is reading the channel, you should see the display change. By right-clicking on your graph, save two images of the signal as `.eps` files, one while you are touching the wire and the other while you are not. **Turn in these files with your code.**

Week 5 Assignment

Phys 13H / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

1. **Get checked out about earth grounds, inadvertent short circuits, and wiring procedures so that you are sure you know how to connect the signal generator and the oscilloscope. Do not proceed until either the TA or the Professor is satisfied that you understand these topics.**

2. Use a signal generator to produce a 100 Hz, 4 V peak-to-peak sine wave centered at 0 V. Examine this wave using the oscilloscope.

3. Connect the signal generator output to the USB-6009 so that it can be measured by Analog Input 0. Be sure you know where the ground and signal leads should be connected. The black clip lead must be connected to Analog Ground, and the red clip lead to Analog Input 0.

4. Using the VI you wrote last week, display the sine wave on a waveform graph. Save the VI to a new name. Set the axes on the waveform graph so that they no longer autoscale. Set the y-axis to display from -2 to +2 V, and the x-axis to run from 0–0.02 s. Check that the frequency and amplitude on the graph match what you see on the oscilloscope. **Save an image of the graph as a `.eps` file and turn it in.**

5. Increase the frequency of the sine wave to 5 kHz. Set the graph x-axis to run from 0–0.01 s. Save an image of the graph as a `.eps` file and turn it in. How does the graph image compare with the oscilloscope trace? Why does it look like it does? **Turn in your answers to these and the questions below.**

6. Leaving the sine wave at 5 kHz, change the x-axis so that it runs from 0–400 μs (0.0004 s). Save an image of the graph as a `.eps` file and turn it in. Is the graph different from the oscilloscope trace? **Explain what you see on the graph.**

7. Write a VI that lets the user enter a desired output voltage between 0.0 and +5.0 V, and produces an error message if the entered voltage is out of range. If the entered voltage is within the range, the VI should set this voltage on the USB-6009 Analog Output 0 when the user pushes a button.

   Use a multimeter to verify that your VI is working correctly.

Week 6 Assignment

Phys 13 / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

Please turn in your code and answers to all questions shown in boldface.

1. Read Chapter 5 in the textbook.

2. Write a VI that generates a sinusoidal voltage on the USB-6009 Analog Output 0. Use a loop with a programmed delay to generate values for the quantity $2.5 + A\sin(2\pi i/N)$, where i is the loop index and N is an integer, and send these to the output. The user should be able to choose the number of data points per cycle (N), the amplitude (A) of the signal, and the frequency, which will be controlled by the delay. A should not be allowed to exceed 2.5 V. Your VI should display the output (or at least its notion of the output) on a waveform graph. Set the graph so that it does not autoscale the x-axis.

3. Read the specifications for the USB-6009 analog outputs in the manual (available on the course web page).

4. Set your VI to produce a 6 Hz sine wave with 20 points per cycle and a 2 V amplitude. Examine the USB-6009 output on an oscilloscope. Use the oscilloscope to measure the frequency of the output. **Does the oscilloscope measurement correspond to the VI setting?**

5. Increase the frequency to 11 Hz, and leave the number of points per cycle at 20. **Based on what you read in the manual, do you expect this to work?** Measure the frequency and examine the waveform using the oscilloscope. **Does the waveform differ from what you told the VI to produce? If so, how?**

6. Download the shell script `slowdown.sh` from the course web page. Using the xterm shell (see the *Viewing PostScript Files* handout), change to the directory where you put the script. Type
   `cat slowdown.sh`
   to see the script code. This script is designed to slow the computer down by repeatedly printing a message. Try running the script by typing
   `./slowdown.sh`

7. Set your VI to produce a 6 Hz sine wave with 20 points per cycle. Examine the USB-6009 output on an oscilloscope. While the VI is running, start

one or more instances of the `slowdown.sh` script. **How is the output affected by the slowing of the computer? To what extent do you think a computer running Windows and LabVIEW can be used to control tasks that depend on precise timing?**

8. Write another VI with similar controls, but with N fixed at 20. This VI should produce a sine wave by calculating the values once, ahead of time, and then using them as needed in a loop. Store the values in an array, and arrange to cycle through, sending one number after another to the Digital to Analog Converter. This way, the computer is not wasting time calculating the sine wave repeatedly. Examine the USB-6009 output with the oscilloscope while you slow the computer down using the script. **Is this VI any more immune to the slowing of the computer? Do you think the limiting factor in the output timing precision is the calculation of values of the sine function?**

9. **Extra Credit:** Working with a partner and a second computer, figure out a way to capture an image of the USB-6009 output while you are running your first VI and the `slowdown.sh` script. Save the image as an EPS file and turn it in.

Week 7 Assignment

Phys 13 / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

Please turn in your code and answers to all questions shown in boldface.

1. This week we will begin using the USB-6009 to try to do something real, namely measure the temperature of a copper rod. The rod has a hole drilled down its center where you can put a thermometer, and it has a small temperature-sensitive resistor called a "thermistor" buried inside the copper. The rod has two sets of wires. One goes to a heater, and the other goes to the thermistor. The heater is about 10 ohms (Ω), and the thermistor is about 100,000 ohms (100 kΩ). **Why does the heater have such a low resistance? Calculate the voltage you must apply to the heater to produce 10 Watts of heating. How much voltage would be needed for a 100 kΩ heater to produce 10 W?**

2. Read the specifications for the USB-6009 5 V output. **What is the maximum current you can safely draw from the 5 V output?**

3. Read the analog input specification in the USB-6009 manual. **About what percent accuracy do you expect of the analog inputs, assuming you keep the device at room temperature?**

4. Familiarize yourself with the thermistor datasheet. We use a Negative Temperature Coefficient (NTC) thermistor made by Vishay, part number 04T1003FP. **What is the nominal resistance of the thermistor at 25 °C? Will the resistance be higher or lower if the temperature drops to 22 °C?**

5. Have a look at the analog input circuitry in Fig. 7 on page 17 of the USB-6009 manual. You can assume that the multiplexer has infinite input impedance (that is, it draws no current from the point in the circuit to which it is connected), and that the programmable-gain amplifier has a gain of 1 (since we use the analog inputs in single-ended mode).

   The analog-to-digital converter (ADC) in the USB-6009 actually has a range of 0–2.5 V. In order to provide the more useful input range of ±10 V, the engineers at National Instruments constructed the circuit you see in the figure. **Calculate the voltage V_m that appears at the node where the three resistors meet, which will be measured by the ADC. Your answer should be a function of the input voltage V_i that appears at the "AI" node. What is the range of V_m if the range of V_i is ±10 V?** The USB-6009 measures V_m, then inverts the function you just calculated and sends a number to the computer that

corresponds to the original voltage V_i. Although the NI engineers accomplished their goal, there is a big drawback to this circuit: the connections inside the USB-6009 will draw current from the point at which voltage is being measured, and may well affect the measured value. Because of the input circuit, the USB-6009 has an input impedance in the neighborhood of 100 kΩ. Typical multimeters, for comparison, have 10 MΩ input impedance, so they are unlikely to affect the circuits that they are measuring. **Be sure to carefully consider the USB-6009 analog input circuit in the next step!**

6. Assume you have available a 5 Volt power supply and accurate resistors with values of 1 kΩ, 10 kΩ, and 100 kΩ. Design a circuit to produce a voltage at one of the analog input channels from which you can determine the resistance of the thermistor. Ultimately, you will convert this resistance to the measured temperature. Try to make your circuit produce a voltage that increases, rather than decreases, when the thermistor gets hotter. This is good human engineering; it makes it easier to keep things straight in your own mind when complications arise. **Turn in a diagram of your circuit.**

7. Figure out which wires are connected to the thermistor. Hint: use a multimeter!

8. After checking the resistors with a multimeter, build your circuit using the screw terminals on the USB-6009 to connect things together. Note that the 5 V supply is available right on the USB-6009. **Do not exceed the rated output current of the 5 V supply!** Write a VI that measures and displays the voltage from your circuit on a waveform chart. Make sure the voltage you are measuring is what you expect based on your circuit design. You should see the measured voltage change if you grab the rod with your hand to warm it up.

9. Improve your VI by making it calculate and display the thermistor resistance (instead of the voltage) on a waveform chart, and then further improve it by incorporating the code from your boxcar averaging program (week 4) as a filter to clean up the signal.

10. Further improve your measurement technique by taking advantage of the fact that A-to-D converters are more linear than they are accurate. By this, I mean that if it reads 0.950 V when you put in 1.000 V, and if it then reads 1.900 V, i.e. exactly twice as much, when you put in 2.000 V, then it is perfectly linear, but only accurate to 5%. Your USB-6009 does not have 8 A-to-D converters, instead it has one of them, and it uses an electronic switch (called a "multiplexer") to choose which of the input signals to measure. Thus, if you measure the ratio of two voltages, that ratio is likely to be more accurate than the value of either measurement alone. **Turn in your new VI and a diagram of any circuit which you design for this part of the assignment.**

Week 8 Assignment

Phys 13 / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

Warning: Be sure to set your analog outputs to 0 V and turn off all equipment before leaving the lab!

Please turn in your code and answers to all questions shown in boldface.

1. Improve the friendliness of your VI from last week by making it convert the resistance to temperature in °C. The resistance of the thermistor, $R(T)$, is related to the temperature T as follows:

$$R(T) = R(T_0)\, e^{[\alpha(T-T_0)+\beta(T-T_0)^2]}, \tag{1}$$

   where $R(T_0)$ is the resistance at any convenient temperature T_0. Your VI should accept as controls $R(T_0)$, T_0, α, and β. For now, set $\alpha = -0.048$, and $\beta = 0.0001$. Allow the user to enter a delay between subsequent voltage measurements, and configure the temperature chart so that the displayed span is long enough for you to see how the temperature changes with time. For doing calculations, you can use a LabVIEW Formula Node. You must use the correct syntax within a Formula Node, so look it up if problems arise. For example, it does not accept `^` as meaning "raised to the power of." Instead, it requires `**`, as in `2**3 = 8`.

2. Now for the fun part! We want to make a temperature controller which accepts as input a desired temperature in the range from 30 °C to 65 °C, and then brings the rod as close to that temperature as possible and holds it there. This can be done by measuring the temperature and changing the voltage applied to the heating wire in such a way as to make the temperature go to the desired *Set Temperature*. The set temperature must, of course, be above room temperature, since we have no active cooling mechanism for the rod.

   To go about this, you need to do two things. One is to calibrate your thermistor, that is, measure $R(T)$ at various T values, and then fit your calibration data using the equation for $R(T)$. The other is to have some means of heating the rod under the control of your computer. Of course, if you are going to control the temperature of the rod, it must also be able to cool off, but we will have to rely on the room air to achieve our cooling for us. That is why you don't want any insulation on the rod. Strangely enough, it is best to begin by implementing the controller, even if it controls

at a temperature rather different from the one you enter. The reason is that it will control to some stable temperature for each temperature setting, and you can then record the actual T and $R(T)$ as your calibration data. This is better than heating it up and trying to measure while things are changing. Some specific steps to take are:

(a) Learn how to use the power operational amplifier (Op-Amp) and the power supply that drives it. Your first thought might be to use the D-to-A output to drive the heater directly, but alas this won't work, because the heater will require an ampere or so to get anywhere, and the D-to-A output can't supply nearly this much current. **How much current can the USB-6009 analog outputs provide?** To deal with this problem, we will use the D-to-A output to control a power amplifier, and let the amplifier drive the heater. **Suppose we didn't want to use an amplifier. How many watts of heating could we produce using the USB-6009 and a 10 Ω heater?**

The Op-Amp just controls the flow of current from the voltage supply to the load (heater). The larger the input voltage you apply to the BNC connector on the left, the greater the voltage the Op-Amp will produce at the output on the right (the output voltage is a voltage difference that appears between the white and black outputs). To function, the Op-Amp requires a special set of supply voltages, namely both a positive and a negative supply voltage, equal in magnitude. That is the purpose of the three banana connectors, red (positive), black (common) and yellow (negative), between the input and the output. The positive supply has a voltage above the common voltage, and the negative supply must be as far below the common voltage as the positive supply is above. To get all this working, first master the power supply without connecting it to anything, so that you know how to make it produce both the positive and the negative voltages desired. To do this, use the series-tracking mode. In this mode, the two main supplies are actually connected in series, like the batteries in a flashlight. Some of the meters on the supplies don't work, so use a multimeter to see what is actually happening. When you can get it to produce -20 V and $+20$ V, with respect to common, go over it with the instructor or the TA, then turn it off, connect it to the Op-Amp, and turn it back on. Don't worry that the Op-Amp is labeled -30 V and $+30$ V, those are just the maximum allowed supply voltages; it will be perfectly happy with our ± 20 V, and that will prevent overheating the rods.

(b) Use the program you wrote in Week 5 to produce an analog voltage output of $+1.00$ V, and connect the analog output to the Op-Amp input. Remember that the analog output is with respect to analog ground, and figure out what to connect to the black banana input and what to connect to the red one. With the heater disconnected, measure

the output voltage of the Op-Amp and see what gain it has. It should be around 5. Now connect it to the rod's heater, and measure the output voltage again to see if the Op-Amp is having any trouble driving the heater. Increase the input to 2.00 V and use your temperature-measuring VI to watch as the rod warms up. **Be sure to set the USB-6009 output back to 0 V and disconnect the heater when you are finished, or the rod may overheat. To be safe, don't let the rod exceed 70 °C. Note that this is hot enough to burn you.**

(c) Now, modify your temperature-measuring program to make it into a controller. I would advise you to first `Save As` and then work on the new copy! Begin with the simplest possible controller. Every time you measure the temperature, compare it to the desired temperature. If the rod is too cold, put out 2 to 3 V using the D-to-A. This will cause the Op-Amp to apply 10 to 15 V to the heater. If the rod is too hot, put out 0.0 V, and this will let it cool off. It will heat up a lot faster than it will cool off! This very simple approach is called a bang-bang servo, and can be quite effective when the object to be controlled responds much more slowly than the "on" or "off" decisions are being made.

Week 9 Assignment

Phys 13 / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

Warning: Be sure to set your analog outputs to 0 V and turn off all equipment before leaving the lab!

Please turn in all your code and answers to all questions shown in boldface.

1. To save time, we will leave the equipment connected from now on. So that everyone is on the same page with regard to software, we will all use identical circuits. Connect the thermistor between the +5 V supply and Analog Input 0, with no external resistors. Using the circuit diagram on page 17 of the USB-6009 manual, figure out the measured voltage $V_m(R_T)$ at the node where the three fixed resistors meet. R_T is the thermistor resistance. Assume the multiplexer has infinite input impedance.

   The USB-6009 sends a reading to the computer which is

   $$V_o = 8.35\,(V_m - 1.23)\,. \tag{1}$$

   Calculate R_T as a function of V_o, and turn in your answer. Modify your servo and temperature VIs from week 8 so that they work with the new circuit.

2. Measure the room temperature, which we will use for T_0, and $R(T_0)$ (using your program). Print these values with the label maker, and stick the label on the plastic support at the end of the rod you are using. **Turn in the values you measured.**

3. Use your bang-bang servo with a set point of 30 °C, and when the temperature of the rod stops changing, use a glass-stem thermometer to measure the actual temperature. **Be very careful with the thermometer; it is much too easy to break.** The thermometers go in the hole along the axis of the rod, which is accessible from one end. Record the temperature and the resistance of the thermistor, and repeat this calibration process about every 5 °C up to about 65 °C or so. Because every thermistor is different, now would be a good time to put a label with your name on your rod (on the plastic support, not the rod itself). You might even want to label your thermometer if you are the least bit paranoid! **Turn in your calibration data.**

4. Use `Grace` to fit the thermistor equation to your data and determine the best values for $R(T_0)$, α, and β. The quantity α is probably about -0.048 or so, and β will be quite small. Physically this means the thermistor resistance decreases about 4.8% for each one degree Celsius increase in temperature. You want to be able to read the resistance and convert it to temperature with a precision of 0.1 °C or better. Glass-stem thermometers are notoriously inaccurate, so think of the 0.1 °C as precision, not accuracy. **Turn in the values you found for the constants and an EPS file containing a plot of your fit.**

5. Modify your servo VI to use the constants you found from your calibration.

Week 10 Assignment

Phys 13 / Phys CS 15 Winter 2010
Professor Everett Lipman

This is a modified version of the original handout written by Professor David Cannell.

Warning: Be sure to set your analog outputs to 0 V and turn off all equipment before leaving the lab!

Please turn in your code and answers to all questions shown in boldface.

So far, our efforts at temperature control have been very simple, but there is an entire branch of engineering devoted to understanding how best to implement such "feedback control systems." You have already implemented the simplest such servo, which applies full heat if the rod is below T_{set}, and no heat if the rod is above T_{set}. This simple solution can work quite well for systems that do not have to respond very rapidly and have a lot of "inertia." It has a hidden advantage, namely it offers very cheap and efficient control of large currents (or voltages). For example, the entire power amplifier could be replaced by a voltage-controlled switch. The point being that a switch does not dissipate any significant amount of power, while a continuously variable linear power amplifier may well dissipate as much power into heat as it delivers to the load. Why don't switches dissipate much power? In other words, why don't they heat up? Well, if a switch is open, how much current is flowing through it? How much voltage difference is there across a closed switch (not the load)? The power delivered to a switch (or anything else for that matter) is the product of the current through it and the voltage across it. Thus, a perfect switch dissipates no power at all, open or closed!

The next step up in sophistication from the "bang-bang" is a "proportional servo." Now the servo applies a correction that is strictly proportional to the "error signal." In our case, the error signal is the difference between the set temperature T_{set} and the measured temperature T. You might be tempted to interpret this to mean that you should put out a control voltage to the power amplifier that is proportional to $T_{\text{set}} - T$, but this is a little naive. We are controlling temperature, and it is power that changes the temperature, not voltage. So, we want to apply an amount of heating power proportional to the error, not a voltage proportional to it. The heating power is V^2/R, where R is the resistance of the heater, so we should make our control program produce a value of V^2/R proportional to the error. A given change in voltage will change the heating power much more if the voltage is already large than if it is small. This "non-linearity" can make a servo work well for situations where the applied voltage is not too large, but then begin oscillating when larger corrections are required. To be specific, you might have a program that works well holding the rod at 40 °C, but which fails when you ask to hold it at 60 °C.

1. Write a VI to implement a proportional servo. Because we cannot apply negative power, it will be necessary to put out a power that is the sum of a fixed power P_0, plus a power proportional to the error: $K_P\,(T_{\text{set}} - T)$. I suggest you make both P_0 and K_P controls so you can vary them easily. See how well this servo can hold the rod temperature at 50 °C. Play with K_P and see what effect it has. If it is too low, the servo should be inaccurate, but what happens if it is too large? Such uncontrolled oscillations are fatal, of course, and much of the literature on servo mechanisms deals with avoiding them!

   The strictly proportional servo can provide finer control than a bang-bang servo, but both it and the bang-bang have an "obvious" problem. The hotter you want the rod to be, the more heat you must apply, because more heat leaks into the room from a hot rod. Alternatively, if the room gets colder, you must apply more heat to keep the rod at a given temperature. To do this, the proportional servo must have a larger error signal, because it is applying an amount of heat proportional to the error signal. Alternatively, for the bang-bang, it must spend a bit more time below the set temperature. So whenever the room cools off, your proportional servo will have to let the rod cool off a bit too in order to generate the larger error signal required for applying more heat. In fact, if you think about it, such a servo can never hold $T = T_{\text{set}}$, can it? It needs to keep T below T_{set} to have an error signal so it can apply any heat at all! If T ever reached T_{set}, the servo would apply no power at all, and the rod would cool off. This behavior is common to all strictly proportional servos, and is called "steady state positional error." The thing never goes exactly where you tell it to go! Of course the larger you can make the response to a given error, the closer it will come to forcing T to equal T_{set}. However, if the gain is increased too much, the servo will begin to oscillate, first overheating then undercooling the rod.

2. Do we have to live with steady state positional errors? Certainly not. **What can we do so that T will eventually be driven to exactly equal T_{set}, at least on average?** Try to figure this one out for yourself and then write a VI to implement it!