Module 3, In-Class, Nov. 14. Thermal Control

The circle is now complete.

November 14, 2016

Keeping Stuff at a Constant Temperature

Last class, you got to play with a Simulink model for control of a simple thermal system. Today we are going to use what you learned from the reading over the weekend on PID control to predict the correct parameters to achieve quick, robust control of our thermal system, and implement it in real life. In this exercise, you will use the same resistor/sensor thermal system we used back when we were learning about first order systems

Recall that the governing equation for this system is:

$$mc_p\frac{d\theta}{dt} = P - hA\theta \tag{1}$$

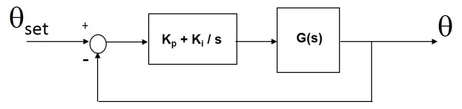
where m = mass, c_p = heat capacity (J/kgK), h = heat transfer coefficient (W/m^2K), A = surface area, P = power or heat (i^2R) generated by the resistor (W), and θ = $T - T_0$ is the difference between the temperature, T, and T_0, the (constant) ambient temperature. Using your experimental measurements a few weeks ago, you found empirical values for quantities hA and mc_p for the resistor system. If you have these values, you may use them for this. If you no longer have them, you can either remeasure them for your specific system, or you can use the approximate values here. For the small size 50 ohm resistor with temperature sensor the constants are approximately $mC_p = 4J/C$ and $hA = 0.03W/K$. The values for the small size 25 ohm resistor should be similar, if you are using the larger size 50 ohm resistor, you will approximately need to double both values.

Finding the Optimal Control Settings

We are going to use PI control for this system. This is the same system we used for the Simulink exercise last class. The first thing we need is the transfer function for the combined heating system and control system.

1. Last time in class, you drew a block diagram of the controlled thermal system which should have included the thermal system block, the control block, a setpoint input, a temperature output, and a feedback loop which combines with the setpoint to become the error signal which is the control block input.

like this for example ...



2. Using the governing equation given above, find the transfer function for the thermal system. (You may have derived it last time for use with the transfer function block in Simulink.

3. The control system has an output which is the sum of a term proportional to the input signal with gain constant k_p and a term proportional to the time integral of the input signal with gain constant k_i. The Laplace transform of this was given in the last class, but make sure you understand where this expression comes from!

4. Now, find the closed loop transfer function of the combination of the two including the feedback loop. (Hint: this was in your readings from a week ago.)

Ok, now we have the closed-loop transfer function of our controlled thermal system. Write it so that both the numerator and denominator are polynomials in s.

5. What do you notice about the denominator? Have you seen this equation before? In what context?

6. Where are the poles of this transfer function? When are they purely real? When are they complex?

7. Just like for the mass-spring-damper or pendulum models, we can define two different regimes: overdamped (purely real roots) where exponential behavior dominates and underdamped (complex roots) where the response oscillates. For what relative values of k_p and k_i do you expect which behavior?

8. The boundary between the underdamped and overdamped is called critically damped. A second order system is *critically damped* if its poles are repeated, i.e. if its two poles are at the same location. As an example, a system with transfer function

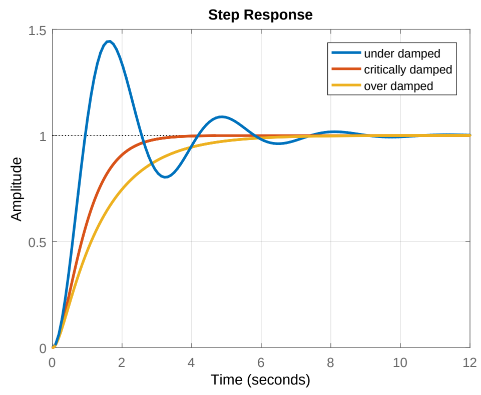
$$H(s) = \frac{1}{(s+2)(s+2)} \tag{2}$$

   is critically damped, since it has two poles at -2.

   The critically damped condition ensures the fastest convergence to steady state, without overshoot (when initial conditions are zero). This is a desirable property for many control systems where rapid convergence to the steady state is desired but there is no tolerance for overshoot.

   Find the equation relating k_p and k_i which determines this condition in terms of your system parameters. Using a k_p value of one, find the corresponding value of k_i.

Example of step responses of a generic second order system in the under, over and critically damped regimes.



Experiment

Now it is time for you to test your predictions. Please note, thermal systems are VERY VERY VERY SLOW. You will not want to be using trial and error to find your optimal proportional and integral constants.

You will need your breadboard, your Analog Discovery, a power supply and board (or the power supply on your desk: both channels), an OPA551 opamp, and one of the resistor/sensor combinations. You will also be using one of the QEA Arduino Uno's: these belong to the class, please make sure they come back to the front of the room when you are finished. Also, please follow the instructions for wiring this up very carefully so you don't burn anything out.

9. As a reminder, the temperature sensor has three connections: from left to right (looking at the black side) they are $V+$ which should be connected to 5 Volts, ground, and signal. Hook up your Analog Discovery ground to ground, V+ to V+, and your Channel 1 to measure between ground and the signal. Turn on the positive supply (under supplies in Waveforms) and start up the scope. The output signal of the temperature sensor is conveniently 10 mV/degree so if you are connected properly, your scope should be reading a constant value around 0.2 V on channel 1. This gives your T_0. BTW, you should adjust your Volts/division appropriately to see a moderate temperature increase from this reading. Also this system is going to be responding SLOWLY. You will want to change your time per division. Something like two minutes per division works well.

10. We are going to drive current through the resistor using the OPA551 powered off the $\pm$ 12V supply in follower configuration off of the Arduino PWM output. (wait, what?!?!!? Don't worry, just take it step by step).

    (a) IF YOU ARE USING THE POWER BRICK AND BOARD Make sure you don't have any part of the temperature sensor wiring hooked up to the rails on your board. Plug in the power supply board to one end of the board and attach the cable from the power supply. Verify that you have $\pm$ 12V, ground and 5V on the appropriate rails. You can then tie your Analog Discovery ground (connected to the sensor) to the ground rail.

    (b) IF YOU ARE USING THE POWER SUPPLY AT YOUR DESK You will need both channels, one for plus 12 volts, one for minus 12 volts. Run the positive output of one channel to one rail and the negative output of the other channel to another rail. Make sure the grounds for both channels are hooked together, and connected to the ground on your board, too, so your Analog Discovery, Arduino and power supply all share a ground. Make sure to verify all voltage levels by measuring with your analog discovery BEFORE you hook up your OPA551.

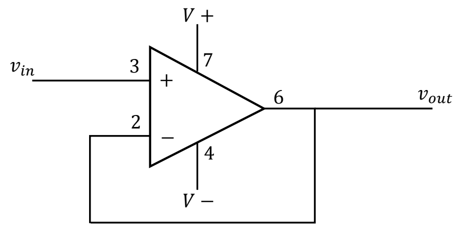
(c) Put the OPA551 opamp across the center of the board and hook it up to power: -12V goes on pin 4, and +12V goes on pin 7.

(d) Wire up the opamp in follower configuration: connect the output pin (pin 6) to the negative input pin (pin 2). (Note: if you are comfortable with it, you can put the opamp in 2:1 amplifier configuration and you will likely get a little bit more speed, but this is up to you. If you are using the larger size resistor, you may want to seriously consider this).

(e) Now hook up your Arduino. Tie your Arduino ground to the ground rail on your board (it is always good practice to tie all your grounds together). You will measure the state of the system using the A1 analog input, so connect this to the measurement pin of your sensor. You will control the system with a voltage applied via the 5th Digital/PWM output, so tie this port of your Arduino to the + input of your OPA551 (pin 3). Also make sure to plug the Arduino into your computer with a USB cable and make sure it is recognized and proper drivers are installed.

(f) Lastly, hook up the resistor between the output (pin 6) of your OPA551 and ground. (Make sure to do this last: unfortunately if the OPA551 is powered and the input is floating, it will tend to rail the output, which will heat up your resistor if it is plugged in and then you will be sad waiting for it to cool before you can begin). You will want to hook up Channel 2 measurement on your Analog Discovery across the resistor, too, so you can see what the control system is doing.

OPA 551 in voltage follwer configuration. The numbers indicate the pin positions.



11. Now download the sketch file thermal_PI from Canvas and read through it. Identify where and how the PI control is being implemented. Where do you input the control parameters k_p and k_i? Where do you input the set 'temperature'?

Ok, you're all set up! Now the fun part. The system changes very very slowly, so we will not want to do a lot of exploration.

12. First, input a 35 degree Centrigrade set point and your derived PI control parameters. Upload the sketch to begin it and take data for a long sweep (2min/div) of the Analog Discovery. How

did you do? Is your system well controlled? Did it reach the set point? Does it oscillate around the set point? If you were to adjust the control parameters by hand to improve the situation, which parameter would you adjust and in which direction?

13. Now we are going to purposely detune the controller so you can experience what a BAD controller looks like. First change the setpoint in the Arduino file to room temperature and upload it so your resistor has time to cool. Wait for a few minutes. Then put the setpoint back and turn up the integral control by setting $k_i = 10$ and leave the proportional control at 1. Upload the sketch to the Arduino to initiate the new control system. What happens? Does this agree with what you would predict would happen in this situation?

14. (IF you have time). Detune the controller in the other direction. Once again, return the setpoint to room temperature and let the resistor cool. Now remove the integral control completely by setting $k_i = 0$ and reinitiate the program. What happens now? What would you expect would happen?