MAE 3260 Final Group Work: Exploring a System of Interest

Report

Outline:

Too Hot To Handle

Exploring a Household Thermostat Control System

Abstract: For this project, we will be studying the dynamics of a thermostat in order to design a closed-loop controller to allow for thermal control of a basic household space. We decided on this project because, while basic, thermostat systems are present in virtually all modern living and working spaces and their performance and efficiency matters greatly in terms of comfort and energy consumption. We thought therefore that this system would be a great one to analyze and study. As part of this project, we will begin by using ODEs and block diagrams to model the system. We will set system requirements for maximum overshoot, rise time, settling time, and max control effort, and then utilize a feedback control law commanded by a PI controller to design our controller accordingly such that the thermostat can heat the space quickly, accurately, and efficiently. Another aspect that we plan to study is disturbance rejection: thermostat systems often have to deal with disturbances in the form of open windows or doors introducing hot or cold air from the outdoors. We will study how thermostat systems respond to disturbances like these and explore ways to implement effective disturbance rejection in our model. Finally, we will design and analyze our own system, tuning parameters and gains to optimize both system performance and overall efficiency.

Student	Task/Role	Portfolio
Maya Watts	Modeled system dynamics for a household thermostat using ODEs and state space representation. Developed an open-loop thermal plant based on heater power and heat loss, and converted it into a block-diagram and state variables (T, R, C). Implemented an on/off temperature control law based on error and integrated disturbance effects from outside temperature to complete the closed-loop	Link

	model. One key takeaway was that I gained a much clearer understanding of how simple PI controllers and deadband logic maintain temperature stability in real heater systems.	
Aidan Friedman	Implemented the PI controller into MATLAB using Simulink. Following the block diagram and utilizing the ODE, this system's response and control effort could be visualized over time. One of the key takeaways from this project was how the type of controller that you use (e.g. PI vs. PID) can vastly affect how computationally intensive your system is to model, and can drastically affect the way your system behaves, even when gains remain constant between controllers.	Link
Clare Simon	Tested different types of controllers (P, PI, PD, PID) in order to determine which method was most effective using the Simulink model. Analyze testing of each type of controller and associated gain values for each. Designed a PI controller to meet system requirements including zero steady state error, minimizing settling time, overshoot, and rise time in the response, and applying real-life considerations such as power usage in deciding on a PI controller.	Not MAE undergrad, no github portfolio link
Samantha Pochet	Identify and model disturbances in the system influencing closed-loop, using linearization and frequency response analysis to interpret system response.	Link

System Modeling - Maya

We modeled a heater for our System. Which means that we were looking at a system that would need to rise when below a certain set temperature and turn off when the heat was at our desired temperature. The reference input was our desired temperature T_{set}, while the system output was the measured temperature T(t). A disturbance term ΔT_{out} was also included to account for variations and this mainly went with the temperature outside because a heater would work differently and have to do less work in early fall versus in the middle of winter. To model this I first had to look at creating an open loop thermal model. The rate of our temperature change would depend on the heat added by the heater and the heat lost through it escaping to to the environment, like through cracks in window sills ect. This resulted in a first order differential equation that relates temperature to heater input and environmental temperature.

To organize the system in control terminology, I defined the state as the measured temperature x(t)=T(t), the disturbance as $d(t)=T_{out}(t)$, and the output y(t)=T(t). The control input was the heater power u(t), which takes certain values depending on whether the heater is ON or OFF. This got me our open loop plant dynamics were as

$$C\frac{dT}{dt} = \frac{1}{R}(T_{out}(t) - T(t)) + u(t)$$

which simplifies to the first order thermal system with capacitance C and thermal resistance R. we wanted. This equation clearly shows that heat loss is proportional to the temperature difference between the system and the environment, while power from the heater adds energy to increase temperature.
To then express this model in state-space form. I defined x(t)=T(t) which got me

$$xdot(t) = Ax(t) + Bu(t) + ET_{out}(t)$$

where $A =- \frac{1}{RC},\ B = \frac{1}{C},\ E = \frac{1}{RC}$.

The output equation for the just returns temperature y(t)=x(t). This helps to make it clear how each parameter affects the system dynamics. We see that when we have a higher thermal resistance or higher capacitance that slows the rate of change of the temperature. We also can see from this that the control input produces a direct heating response.

To keep the temperature where it was desired, I used an on/off PI controller using the error $e(t)=T(t)-T_{set}$. This helps us make sure that when the measured temperature is below the setpoint ($e<0$), the controller switches the heater ON, and when the temperature rises above the setpoint ($e\geq0$), the heater switches OFF. This makes the design of the heater much better because if this heater was paired with an AC you wouldn't want when it gets too hot in a room for the AC to switch on, you would just want the heater to switch off and allow the room settle back down staying in a good temperature range. This design produces a temperature that rises to a max value slightly above the reference and then settles down to the desired value. This then repeats if the room temperature gets too low again, rather than just converging exactly to the reference temp, which makes our system consistent with real heater/thermostat behaviors.

$$Tdot(t) = -\frac{1}{RC}T(t) + \frac{1}{RC}T_{out}(t) + \frac{1}{C}u(e(t)).$$

I then combined the PI controller with the plant results in the closed-loop thermal differential equation. This part of the model basically shows how the heater turning on and off actually changes the temperature over time, and how the outside temperature also affects it. In the closed-loop system, the temperature doesn’t go perfectly to the setpoint, but the on/off controller keeps switching the heater and keeps the temperature kind of hovering just around where we want it, even if the room or outside air is cooler or warmer.

Going through the steps of building this model really helped me understand how the thermal system behaves and what the main parts are doing. Writing the equations and putting it into state space made it easier to see how the temperature, heat loss, and heater power all related and helped me tie it all back into how a real system works/ to what we have been learning in class. It also helped me get a better idea of things like how a deadband works and why it is so important to the function of technologies like a heater.

Variables and parameters

State $\left(\begin{smallmatrix}\text{room}\\\text{temp}\end{smallmatrix}\right)$: $x(t) = T(t)$ — loop

Reference : T_{set}

Disturbance : $d(t) = T_{out}$

Control Input: $u(t)$

Output : $y(t) = T(t)$

Error: $e(t) = T(t) - T_{set}$

if $e < -1^\circ$ then heat ON until $e = 0$ set point

Thermal plant (open loop ODE)

$$C\frac{dT}{dt} = \frac{1}{R}\left(T_{out}(t) - T(t)\right) + u(t)$$

$$\frac{dT}{dt} = -\frac{1}{RC}T(t) + \frac{1}{RC}T_{out}(t) + \frac{1}{C}u(t)$$

State Space

Using $x(t) = T(t)$

$$\dot{x}(t) = Ax(t) + Bu(t) + ET_{out}(t) \quad , \quad y(t) = Cx(t) + Du(t)$$

$$A = \left[-\frac{1}{RC}\right], \ B = \left[\frac{1}{C}\right], \ E = \left[\frac{1}{RC}\right], \ C = [1], \ D = [0]$$

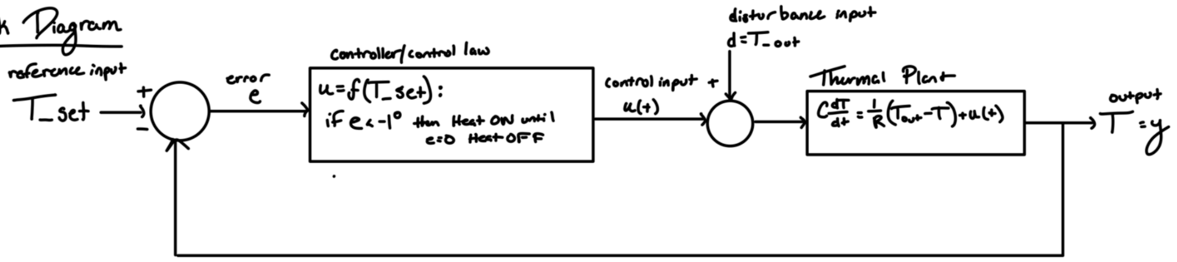
Controller

$$u(t) = \begin{cases} \text{Heat ON}, & e(t) < -1 \\ \text{Heat OFF}, & e(t) \geq 0 \end{cases}$$

Closed Loop ODE plant + controller

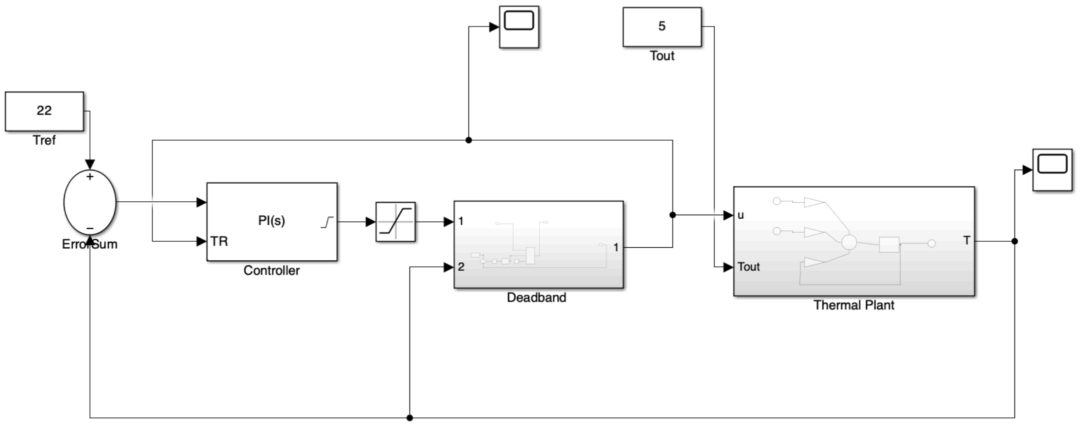
$$\dot{T}(t) = -\frac{1}{RC}T(t) + \frac{1}{RC}T_{out}(t) + \frac{1}{C}u(e(t)) \quad , \quad e(t) = T(t) - T_{set}$$

Block Diagram



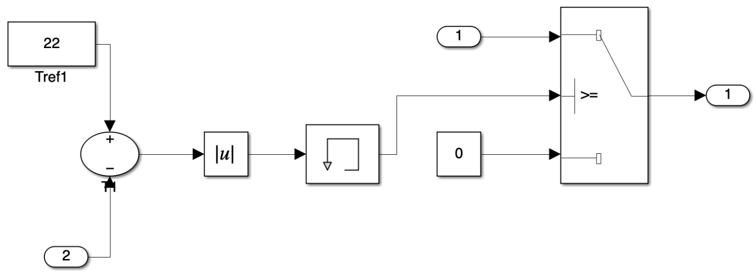
MATLAB Implementation and Plotting - Aidan

Based on the system ODE and the block diagram presented in the previous section, the system behavior can be modelled via MATLAB Simulink.

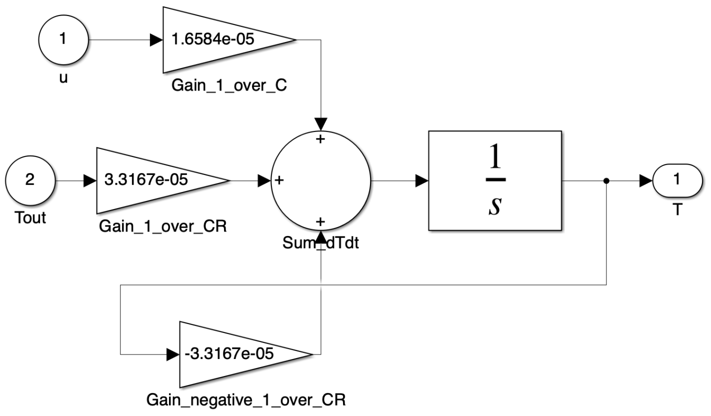


The model above accurately implements the block diagram, where Tref is the reference temperature of the home that the thermostat is set to, and Tout is the “disturbance” temperature outside of the house. The model above also includes a saturation block, which limits the maximum control input of the heater system. For the modelling in this report, the saturation value was set to be 1500W, which is typical for many home heating systems [1].

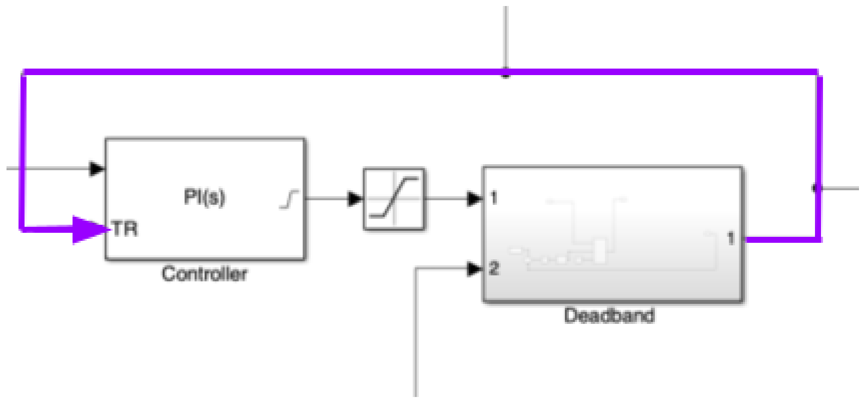
The logic of the control law is encapsulated in the “Deadband” block, which takes in two inputs: control input, U, and current temperature, T. This block outputs U whenever the error between the current temperature and the reference temperature has a magnitude greater than 0.25C, and outputs a control input of 0 whenever the error is less than 0.25C, effectively turning the heater off. This mimics the behavior of real thermostat systems, which turn on/off given a specified temperature threshold. The “Deadband” block is implemented according to the Simulink model below:



After the Deadband block, the Thermal Plant block implements the ODE of our system, simulating how the temperature of the room changes over time based on the input power (U) and the outside environment temperature. Inside the block, the heater input and the heat loss effects are combined, and the total rate of change is fed into an integrator which outputs an actual room temperature as a function of time. There are 3 gains in the Thermal Plant, all of which are dependent on C (the thermal capacitance of the room), and R (the effective thermal resistance between the room and the outside environment). Since it was difficult to find reasonable C and R values online, ChatGPT was used to estimate C and R for an average home. The Thermal Plant is implemented according to the Simulink model below:

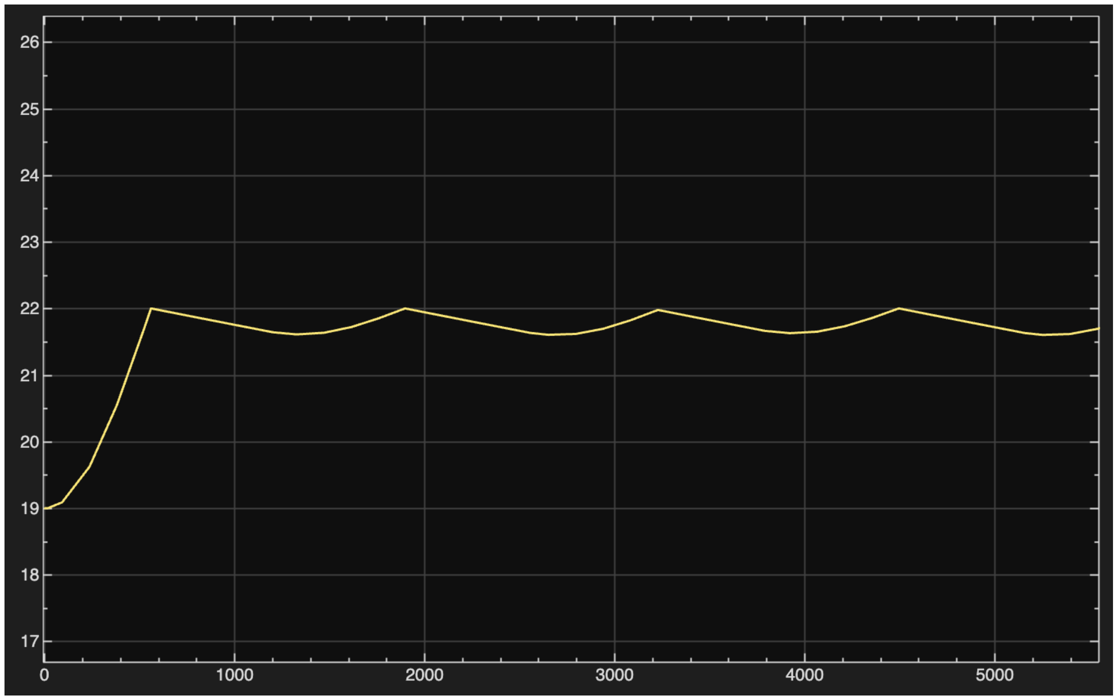


One of the most important aspects of the Simulink implementation turned out to be enabling tracking mode on the PI controller, which is necessary when utilizing deadband logic. What this does is prevent integrator wind-up in the controller by forcing the output to zero during periods when the heater is off. Without tracking, the integral term would typically accumulate error over periods when the heater is off, which led to a constantly increasing control signal which would ultimately cause it to saturate. The tracking can be seen in the extra “TR” input to the PI controller, which routes the actual control signal passed to the thermal plant back to the controller. This stabilizes the controller, and keeps the controller output from building up artificially during periods when the heater is not on. The tracking is highlighted in purple below:

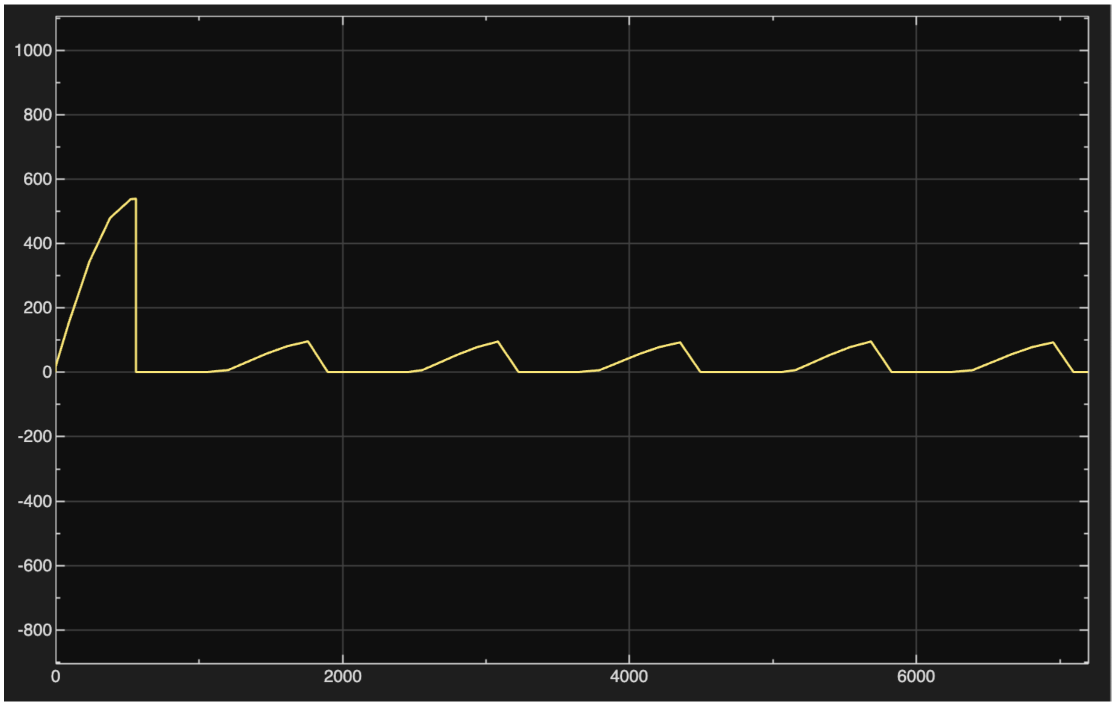


Once the Simulink was set up, both the system response and the control output could be plotted, as shown below:

Temperature (C) vs. Time (s)



Control Input (W) vs. Time (s)

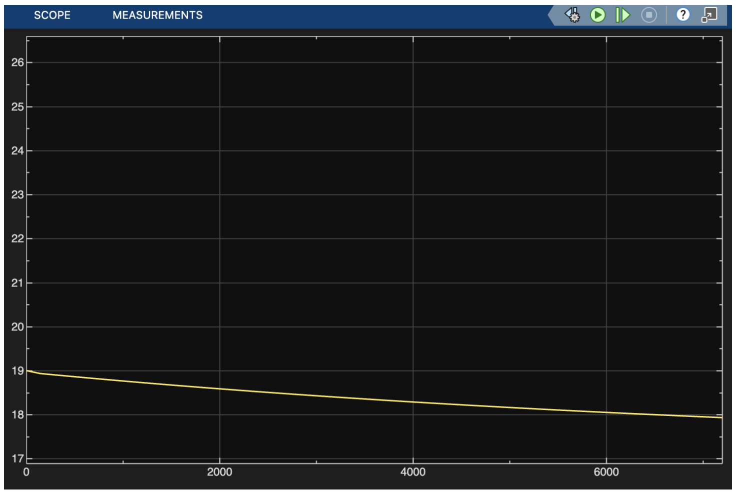


System Requirements - Clare

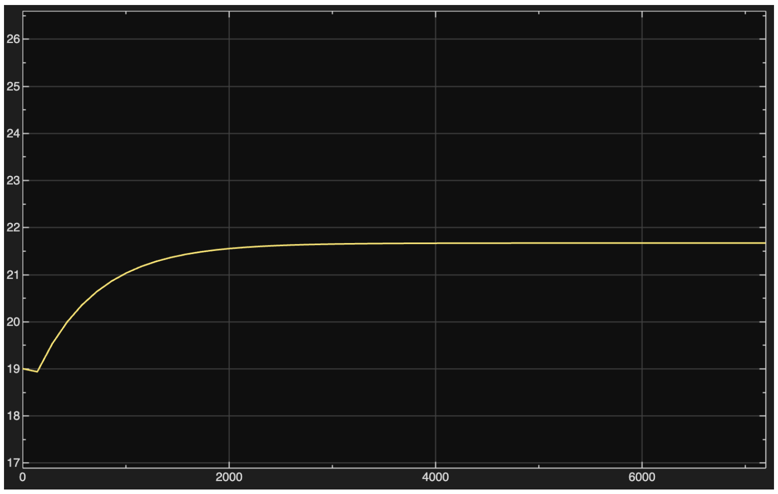
Based on the system requirements we defined in our proposal, namely to minimize overshoot, oscillations, rise time, and settling time, we tested different types of controllers. We then analyzed the output response to identify the advantages and disadvantages of each controller type, all using values of $T_{ref} = 22$ and $T_{out} = 5$, in order to find the most efficient controller.

Proportional Control

We began our design of the controller with a Proportional-only control system. We tested a large range of values for K_P using the Simulink model, ranging from 1 to 200. Below are two examples of output responses with K_P values of 5 and 100. As shown in the graph on the left, a gain value that is too low will not be effective in helping the system reach the reference input temperature, instead the room gradually cools due to the lower outside temperature. In the graph on the right, a larger K_P value allows for the system to ramp up and attempt to reach the reference input of 22.



Graph 1: P-only system with $K_P = 5$



Graph 2: P-only system with $K_P = 100$

Let y (output) be the actual temperature of the room. In this case, analyzing the error in the system goes as follows:

$$e_{ss} = T_{ref} - y_{ss}$$

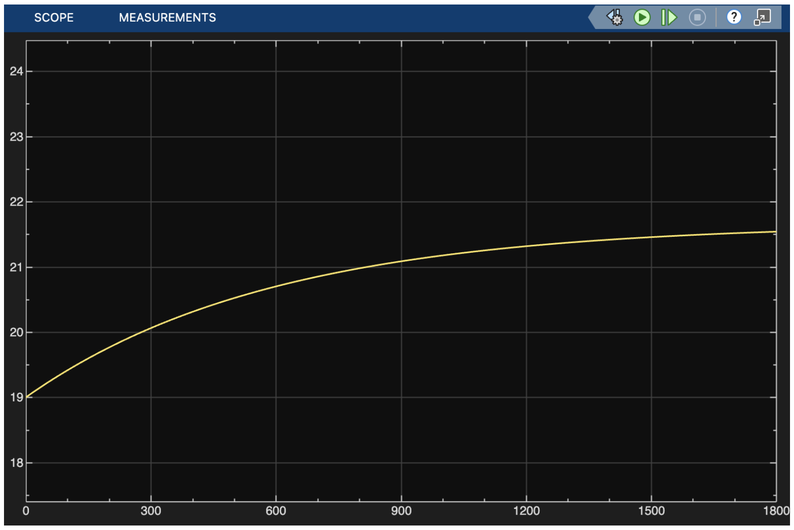
From Graph 1, the response approaches $y_{ss} = 18$ over the time period of 2 hours (7200s). This constitutes a steady state error of 4 degrees. Steady state error is a feature of a Proportional-only control system. In this case, it appears that the system would continue to cool due to a negative slope at the rightmost edge of the graph, even after the period of 2 hours. From Graph 2, the response approaches 21.67 degrees over the same time period. This constitutes a steady state error of 0.33 degrees. Although this is much improved from the case with a low K_P value, our design goals are to reach a steady state error of 0 degrees.

With a Proportional-only controller, the system will always have a steady state error since it is a feature of the controller type. Thus, we continued our design testing by adding an integral term in order to eliminate this error, using a PI controller (non-zero K_i).

Proportional-Derivative Control

We decided to test a PD controller in order to see if it was effective in damping oscillations, reducing overshoot, and decreasing settling time. Although the error system requirement would not be met, we decided to test this controller in order to understand the effect of derivative control most directly.

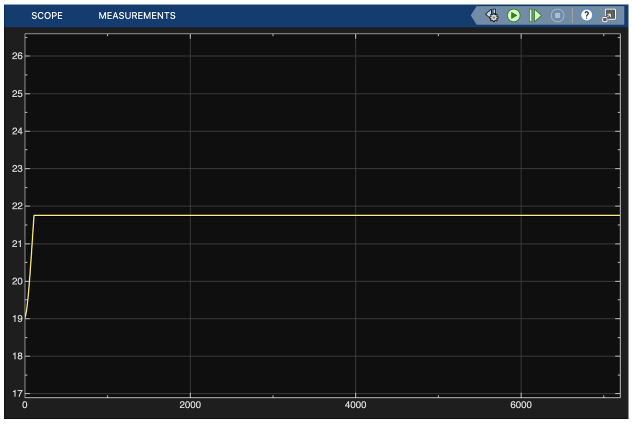
As we tested different values for K_d and K_P, the Simulink took a very long time to load, leading us to decrease the stop time to 1800s (0.5 hr). With a K_P value of 100 (as before) and a K_i value of 0.01 (very low because higher values extended the time taken for the simulation even longer), there was very slow progress made over the time domain toward the reference temperature. It can be observed that the graph shows a slow increase in the temperature, and does not yet reach the steady state at the end of the time domain. Without integral control, the system cannot eliminate small errors that persist, causing the response to still have a steady state error of approximately 0.5 degrees.

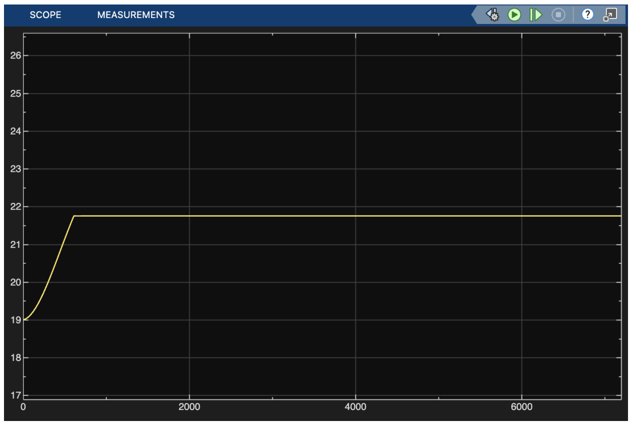


Graph 3: PD system with K_P = 100, K_d = 0.01

Although K_P is intended to improve rise time, it appears that adding derivative control made it less effective in doing so. Thus, it seems that a PD controller is not a very effective method due to its inability to correct errors and the slowing of the system response. Additionally, a feature of a PD controller is that it has the potential to amplify noise, so if the sensor measurements are noisy, it would further disturb the system response.

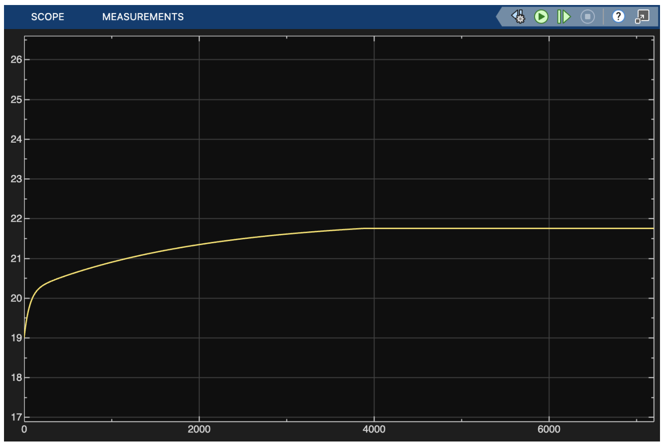
Proportional-Integral-Derivative Control





Graph 4: PID with K_P = 100, K_d = 10, K_i = 10

Graph 5: PID with K_P = 100, K_d = 10, K_i = 0.5



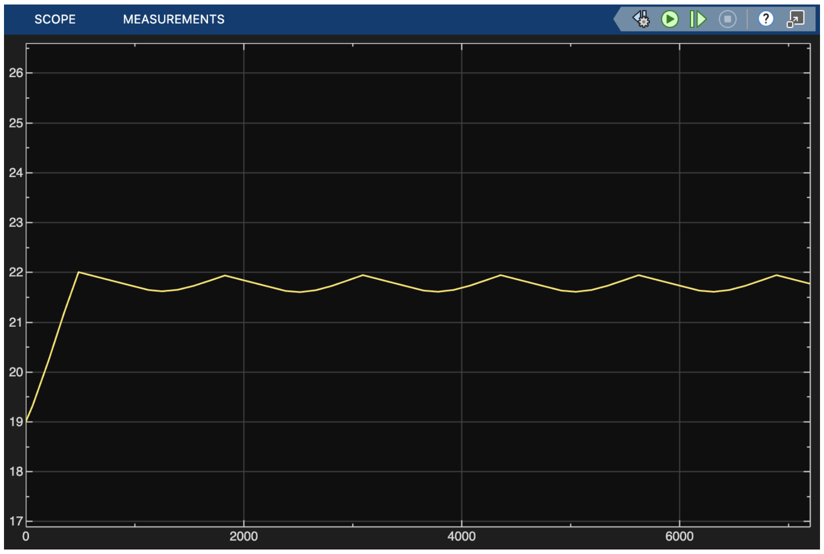
Graph 6: PID with $K_P = 1000$, $K_d = 10$, $K_i = 0.5$

For PID control, we tried multiple cases, varying each gain value. As shown in Graphs 4 and 5, an increase in K_i led to a faster response, but arguably too fast. At that rate, the room heats up by nearly 3 degrees within 127 seconds (just over 2 minutes). This would cause a huge spike in the power it takes to run the system, causing it to be more expensive for the user, and applying unnecessary strain on the system. As shown in Graphs 5 and 6, increasing K_P caused the response to take much longer, not reaching a steady state until nearly an hour. Varying the K_d value did not show much of an effect on the system from observation, even when changing it by factors of 10.

Overall, the PID system in all cases had a non-zero steady state error. This could not be remedied by increasing the integral gain, nor changing the other gain values. Thus, we decided that derivative control was causing more harm than helping the system, leading us to settle on PI control.

Proportional-Integral Control

With our PI control system, we observed the response shown in Graph 7. We used a value of K_P =5 in order to achieve an appropriate rise time. The system changes by about 3 degrees in about 10 minutes, which is feasible for a small, well insulated room. We did not choose a higher K_P as to not overload the system's power. We chose a K_i value that does not saturate the system and still allows the temperature to reach the desired reference value. We omitted the derivative control which interfered with achieving steady state error and did not considerably improve system performance.



Graph 7: PI with $K_P = 5$, $K_i = 0.5$

Although the system still exhibits some oscillations, this is appropriate for a realistic system. The thermostat is set to turn off when the room is at the desired temperature, which saves power. Due to T_{out} being less than inside, the temperature decreases, and the thermostat is turned back on when there is an error of 0.5 degrees. The system reaches the requirements we set, including no overshoot, no steady state error, quick rise and settling time. The use of a PI controller allows for the user to maximize the efficiency of the response without using too much power or sacrificing the accuracy of the final temperature.

Disturbances - Sam

The thermostat system is modeled as a feedback control loop in which the controller compares a desired indoor temperature (*T_{set}*) to the measured current temperature *T* and adjusts the heating input (*u(t)*) to close the difference between the two temperatures. The thermal dynamics of the room are modeled as a first order equation, and the outside temperature T_{out} is treated as a disturbance input on the system.

The thermal system's first-order equation includes the disturbance T_{out} which enters additively through the (1/RC)* T_{out} term. Because this term is directly proportional to the temperature difference between indoor/outdoor, any sharp or gradual change in external temperature immediately affects the dynamics of the indoor temperature. The magnitude of the influence the disturbance has on the system is influenced by the thermal resistance *R*. For example, a building space with lower resistance would allow for the outdoor disturbance to have a greater effect on the thermostat temperature control system.

To analyze how exactly an outdoor disturbance affects the system, the closed-loop ODE needs to be linearized around a steady operating point.

To analyze the disturbance propagation, the non-linear ODE is linearized around a steady operating temperature value.
A steady operating condition is defined by constant values

$$T^* = T_{set}, \qquad T^*_{out}, \qquad u^*,$$

Substituting into our original ODE gives:

$$0 = -\frac{1}{RC}T^* + \frac{1}{RC}T^*_{out} + \frac{1}{C}u^*,$$

And results in the equilibrium control input:

$$u^* = \frac{1}{R}(T^* - T^*_{out}).$$

This equation represents the heating power required to maintain the space indoors at the chosen temperature under nominal outdoor conditions.

To then linearize the controller in the closed loop equation, we conduct a first-order Taylor expansion, which gives:

$$\tilde{x} = T - T^*, \qquad \tilde{d} = T_{\text{out}} - T^*_{\text{out}}, \qquad \tilde{u} = u - u^*, \qquad \tilde{e} = e - e^* = \tilde{x}.$$

Substituting these equations into the original equation gives:

$$\dot{\tilde{x}}(t) = -\frac{1}{RC}\tilde{x}(t) + \frac{1}{RC}\tilde{d}(t) + \frac{1}{C}\tilde{u}(t).$$

This equation includes the linear contributions of the disturbance and control.

We then linearize the control to ensure it's compatible with the linear system analysis.
First order Taylor expansion of u(e) results in:

$$\tilde{u} \approx k_e\tilde{e} = k_e\tilde{x}, \qquad k_e = \left.\frac{du}{de}\right|_{e=0},$$

Where ke represents the local control gain.

Substituting into the new ODE results in the linearized, closed-loop ODE:

$$\dot{\tilde{x}}(t) = \left(-\frac{1}{RC} + \frac{k_e}{C}\right)\tilde{x}(t) + \frac{1}{RC}\tilde{d}(t).$$

Taking the Laplace transform, assuming zero initial conditions, results in:

$$s\tilde{X}(s) = \left(-\frac{1}{RC} + \frac{k_e}{C}\right)\tilde{X}(s) + \frac{1}{RC}\tilde{D}(s),$$

The frequency response of the disturbance is shown through:

$$G_d(j\omega) = \frac{\tilde{X}(j\omega)}{\tilde{D}(j\omega)}.$$

For low frequency disturbance, the feedback loop exhibits high gain, so the Gd term becomes small. For higher frequency disturbances, the loop gain decreases, and Gd allows a larger portion of the disturbance to reach the indoor temperature. The disturbance analysis expresses the system in its linearized form with both the combined state equation and controller.

References

[1]P.C. Thijssen, “State estimation,” *Elsevier eBooks*, pp. 20–36, Jan. 2010, doi: https://doi.org/10.1533/9780857099372.19.
[1]S. Liu and B. Yao, “Characterization and Attenuation of Sandwiched Deadband Problem Using Describing Function Analysis and Application to Electrohydraulic Systems Controlled by Closed-Center Valves,” *Journal of Dynamic Systems Measurement and Control*, vol. 131, no. 3, Mar. 2009, doi: https://doi.org/10.1115/1.3089557.
[1] W. “How Much Electricity Does an Electric Heater Use? Complete 2025 Cost Analysis,” SolarTech Online, Aug. 15, 2025.