Database Hands-On: Water Tank and Heater

Exercise 1: Water Tank Simulation

The goal is to simulate the water temperature in a heated tank.
Implement an EPICS database called "tank.db" to accomplish this. The records that drive the calculation should process at 1 Hz. Assume that the water volume is constant, i.e. only the heater and the room temperature influence the water temperature.
Implement analog (input, output?) records that allow configuration of the following:

- Room Temperature T_R, initialized to 25 deg Celsius.
- Tank Isolation Constant K_I, initialized to 0.01.
- Water Volume Heat Capacity c_V, initialized to 0.001.
- Water Heater Voltage V_H, restricted to 0…110 Volts.

Use record names that start with a macro like "$(user):" to assert that your records are unique on the network, and set user=t1, t2, … depending on your group!

Create an EDM GUI that allows configuration of these records.

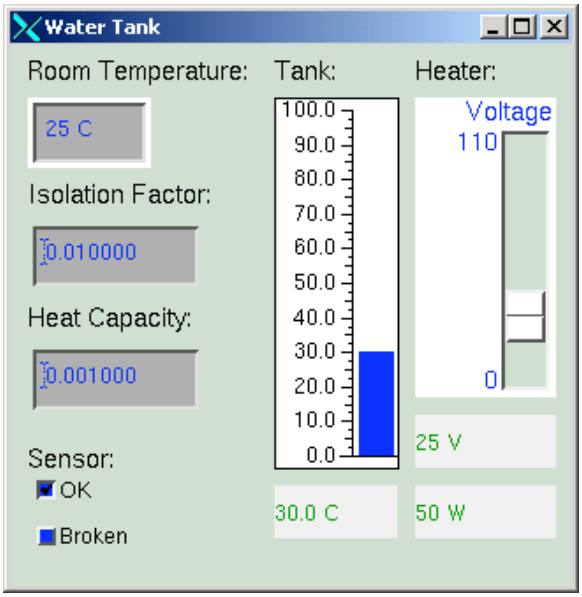
Add a calc record to transform heater voltage V_H into heater power P_H.
Hint: If your heater coil has a resistance R, its Power dissipation for a given voltage will be $P_H = (V_H)^2/R$. Pick your resistance such that your heater e.g. produces about 1100Watts at 110Volt.

Add a slider for the voltage to the GUI, and an indicator for the resulting heater power.

Implement a calc record that simulates the water temperature T, processing once per second. Each time the calc record is processed, it should calculate the "new" water temperature $T(n)$ based on the "previous" water temperature $T(n-1)$. Example:

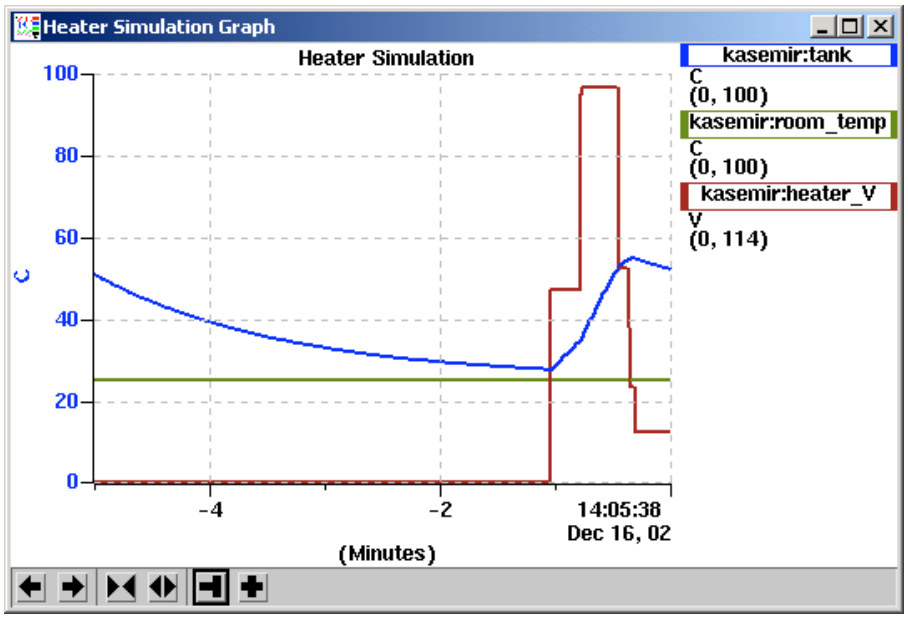
$$T(n) = T(n-1) + [T_R - T(n-1)] \times K_I + P_H \times c_V$$

In here, the tank isolation constant would better be named "tank wall heat conductivity". Extend the EDM screen and add a StripTool configuration so that you can exercise the database. In essence, the water temperature should increase when the heater is "on", and approach room temperature when the heater is "off".



Example EDM User Interface
(already including the 'broken sensor' simulation from a later exercise)

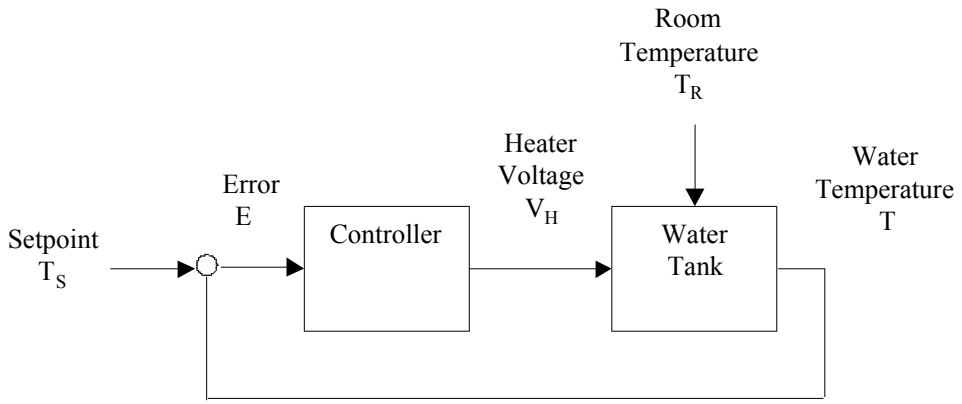
In the preceding example, the parameters K_I and c_V may be chosen to simulate anything from a big water tank with excellent isolation down to a 1 oz paper cup. The initial suggestions were such that you can observe changes in water temperature on a reasonable time scale. In the example shown below, it takes the water a few minutes to cool from 50ºC down to the room temperature of 25ºC, and one can heat it back up in a minute:



Example StripTool Graph

Change the values temporarily to verify that you can simulate a system that heats 'quicker' or has better thermal isolation.

Exercise 2: Heater Controller



The goal is to add computer control to the water heater simulation. Create a new database "control.db", meant to run together with the previous "tank.db", that contains an analog input record for entry of the

- Setpoint Water Temperature T_S
- Control Parameters (see below)

Add calc records that determine the temperature error $E(n) = T_s - T(n)$ and generate an appropriate heater voltage. One commonly used generic control algorithm is the "PID" controller, which has proportional, integral and differential portions. Based on the current error $E(n)$, it calculates a new output $O(n)$ as follows, considering previous errors $E(n-1)$, $E(n-2)$, ...:

$$O(n) = K_P \times E(n) + K_I \times \sum_i E(i)\, dt + K_D \times [E(n)-E(n-1)]/dt$$

Some notes, including a simple PID introduction:

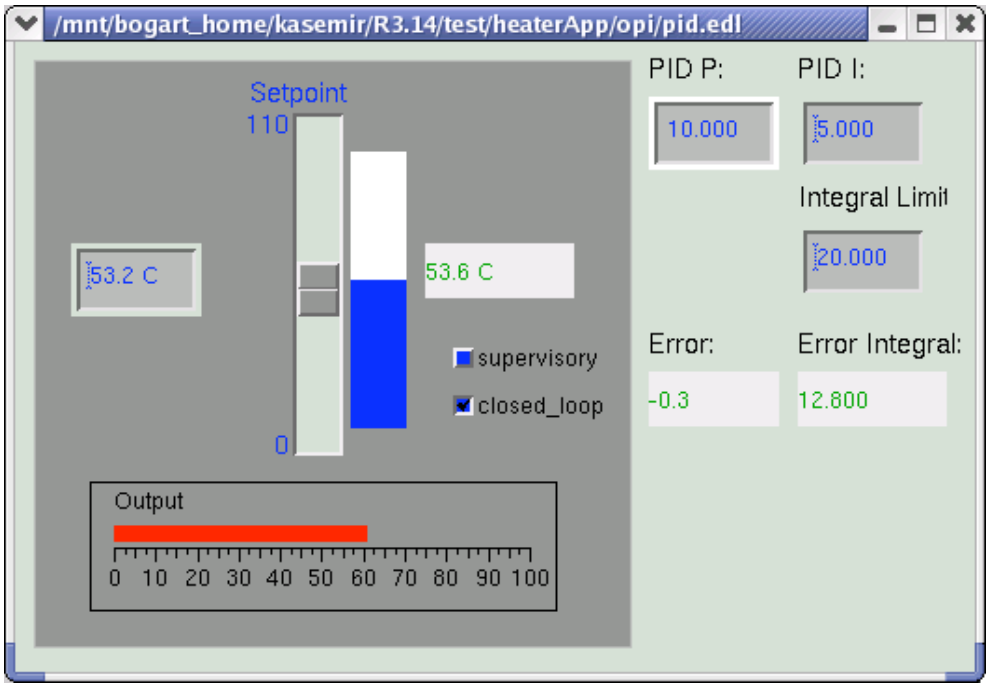
- dT, the time step, can be "1 second".
  In practice, the differential constant K_D is often set to 0, because it is quite difficult to adjust. You can ignore the whole differential term of the PID equation. So we get:
  $$O(n) = K_P \times E(n) + K_I \times \sum_i E(i)$$
- Assume our error is $E(n)>0$, meaning the water is too cold. Using a heater voltage of $K_P \times E(n)$ will turn the heater on whenever the temperature is too cold. If we're way below the setpoint, we'll add a lot of heat. When we're close, we'll add a little heat. That's the proportional or P part of the PID.
- Unfortunately, that leaves us a little short of the goal. In case we reached the setpoint, the error is of course zero, hence the heater turns off, and the temperature drops. This again results in a small error and thus a little heat, but not enough to reach the setpoint.
  A small, constant amount of heat is required to keep the setpoint. This could be added as a manual offset:

$$O(n) = K_P \times E(n) + Offset,$$

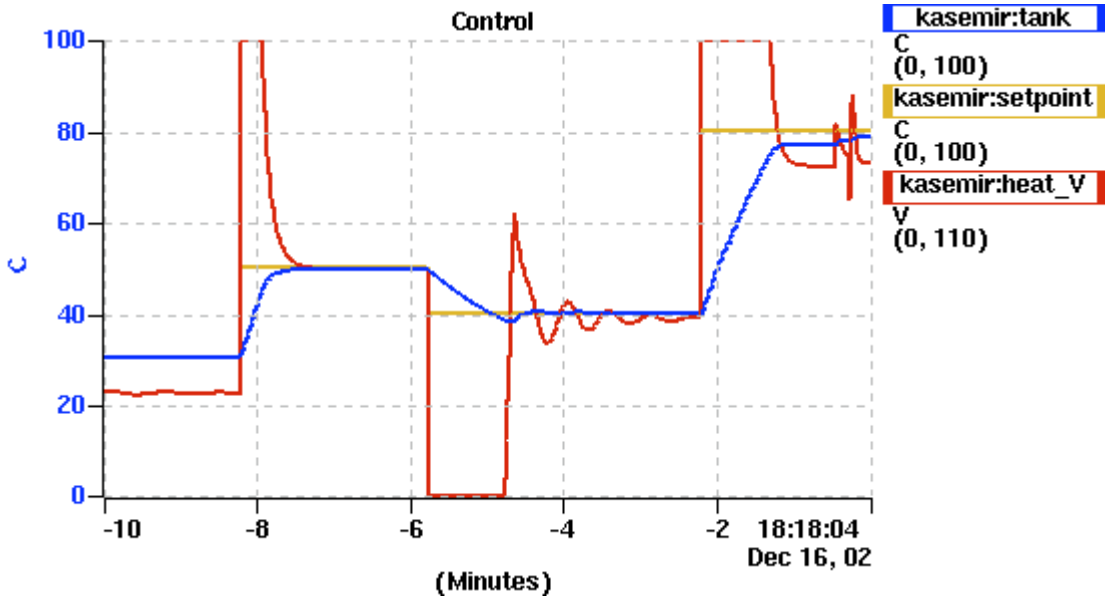
but then of course somebody has to tweak that manual offset to be 'correct'…

- The integral part of the PID controller, $K_I \times \Sigma_i E(i)\, dT$, is also known as *automated offset*. Instead of manually adjusting the offset for each given setpoint, the controller determines the offset by accumulating the small errors until they vanish.
- When using the integral part, it is useful to limit the integral value so that it does not grow out of bounds.
- Assert that the heater voltage stays within 0…110V!
- In the previous exercise, we manually adjusted the heater voltage. Now the output of the PID is supposed to set the heater voltage. Especially for testing, it is nice to be able to switch back and forth between "manual" and "automated" operation.
  One solution is to use an analog output record, DOL field set to read the output of your PID. The OMSL field then allows you to switch from "closed_loop" (DOL is used) to "supervisory" (operator interface can adjust value).
- We have only "heating" and no "cooling". When experimenting, your tank temperature might soon hit 100ºC and you would have to wait until it cools down to room temperature. Remember that you can trick the simulation by simply setting the tank temperature to 0ºC via a "caput" or a "message button" on your EDM screen.

Provide a user interface to the "control.db" records. The following example somewhat resembles the traditional look of hardware PID control boxes: Setpoint and readback shown in parallel, output of PID reflected below.

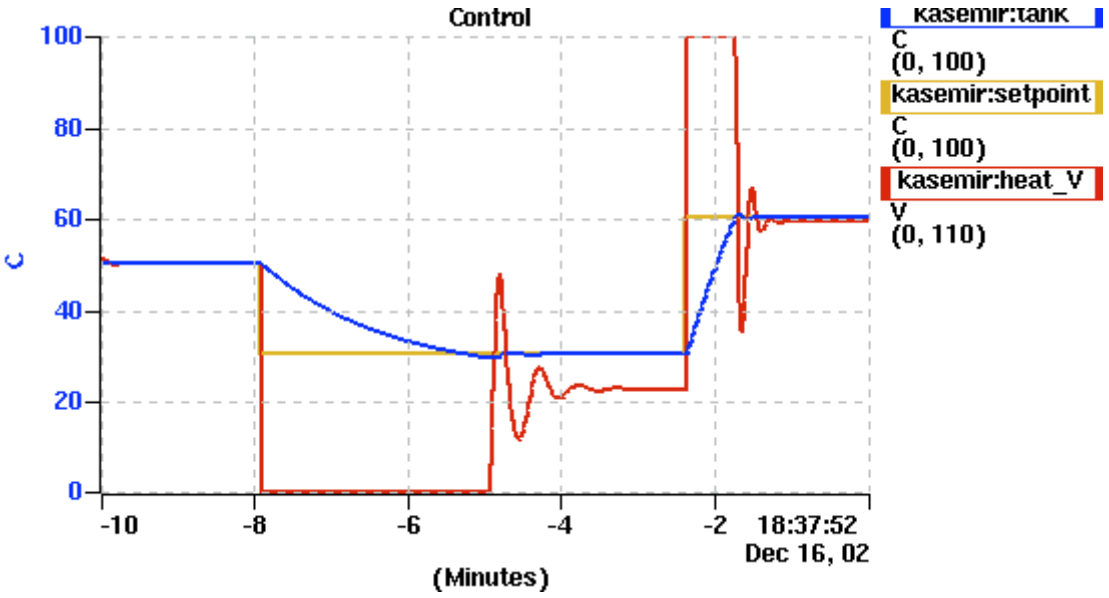


Perform some tuning of the K_P and K_I parameters.

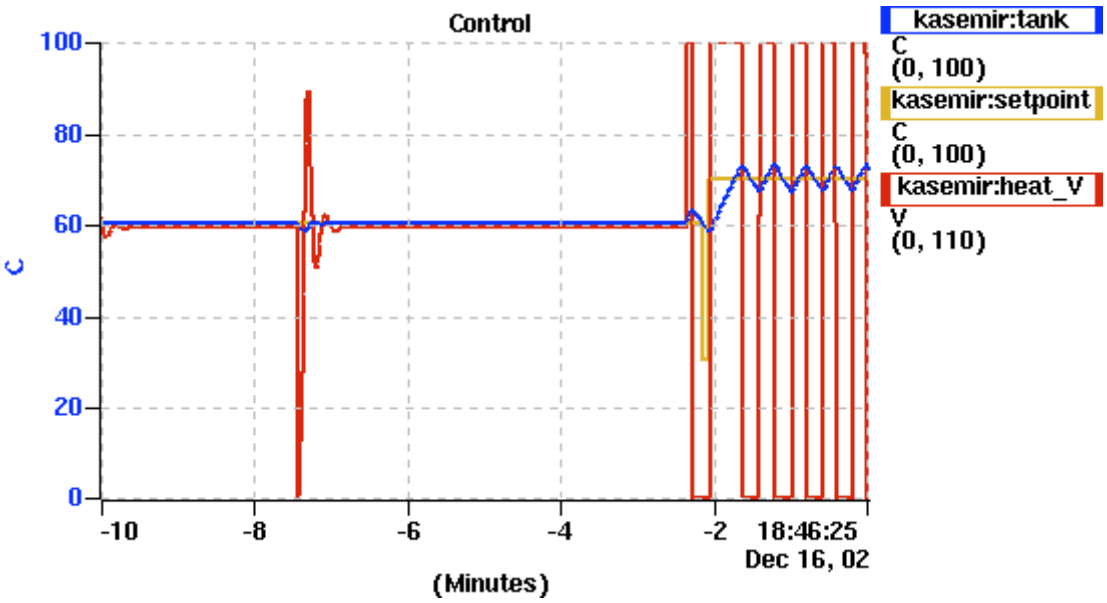


In the above example, the parameters where adjusted such that when modifying the setpoint, the heater initially goes to 0 or 110V (saturation), allowing the water temperature to quickly fall or rise. When the water temperature is getting close to the setpoint, the heater is regulated such that the setpoint is reached.

In the following example, the parameters K_P and K_I were increased. The water temperature "overshoots" the setpoint and more violent adjustments of the heater follow, though in the end the setpoint is reached more quickly:



In the last example, the chosen parameters are too big. The water temperature oscillates around the setpoint, but never reaches it:



Exercise 3: Additional Ideas

- Add a "noise" record that adds to the measured temperature and simulates a less-than-perfect measurement.
- Simulate a "broken sensor". In reality, the temperature sensor for the tank could fail, often because of an open connection. Many A/D boards and well-written drivers would recognize this and put the associated analog input record in e.g. READ/INVALID alarm. All the records linked to this input record should use "MS" (maximize severity) links so that they, too, turn invalid.
  Simulate this situation: add a button to the user interface for selecting "broken sensor". Depending on the setting of the button, you somehow fake a broken connection by e.g. switching from the simulated tank temperature to a constant ai record with HIHI/HSV settings that result in an INVALID status/severity.
- Add SNL code which determines the time from a changed setpoint until the water temperature reaches the new setpoint.
- Add calculated EDM fields that display the temperatures in Fahrenheit.
- Consider alarm handler configuration: What possible alarms are there? Add a configuration for ALH and run it.