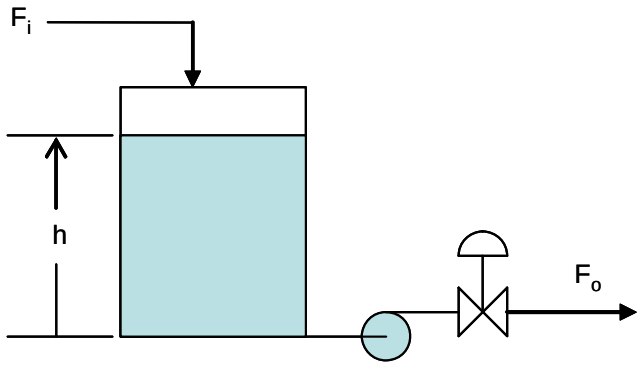
Problem Set 5

The tank is 1 m tall and 60 cm in diameter. The flow to the tank varies unpredictably and without regular pattern, but has historically been between 0.06 and 0.085 m^3 min^{-1}. We desire to pump from the tank at 0.07 m^3 min^{-1}, but are willing to adjust this flow to maintain the liquid level at 60 ± 5 cm. We obtain a level instrument that measures over the full range of the tank. We also obtain a proportional controller, which takes a 0 - 100% input and delivers a 0 - 100% output. We need to size an appropriate control valve for the outlet stream (part (2)), and we need to tune the controller (part (3)).



(1) From an earlier problem set, you know that the pumped-outlet tank is unstable. You've already derived a system model for such a tank. For warm-up, present it in Laplace domain, so that the transform of the output liquid level [h'(s)] is clearly related to the inputs via transfer functions. Determine the poles of these transfer functions. Do these indicate the instability?

(2) "Size the valve." Ultimately, that means a physical size, something you select from a catalog. However, at this point we will be content to specify the required flow capacity. Do this by determining the outlet flow you will need to compensate for the most demanding disturbance. Then make your specification 10% larger, so that you can avoid banging the valve fully open. Finally express your valve "size" as the gain Kv of the transfer function between controller and outlet flow rate. In writing this transfer function, presume that the valve is much faster than the tank, not like the one in Lesson 4.

(3) Download the (new and slightly improved) file template.m from the MIT Server. Modify it to describe closed-loop operation of the tank, using the equipment we have described above: process, sensor, controller, valve. This means that you will need to code the controller algorithm. Remember that the controller's job is to calculate an output that positions a valve. That controller output depends, in proportional mode, on the controller bias (xco_init), the set point (ysp), the sensor reading (ys), and the controller gain (Kc). The subfunction argument list contains all the items you should need. When you've coded things, do some common-sense checks to catch blunders. For example, run it both open- and closed-loop with steady inputs to verify that it holds steady state. Then run it open loop with a simple disturbance to see if the process behaves as it should.

The goal of this part (3) is to find the smallest controller gain that satisfies the control objective stated in the introduction, for the worst disturbances (under the limits stated above) you throw at it. Do so; state the gain, and turn in a disturbance response plot that shows its triumph.