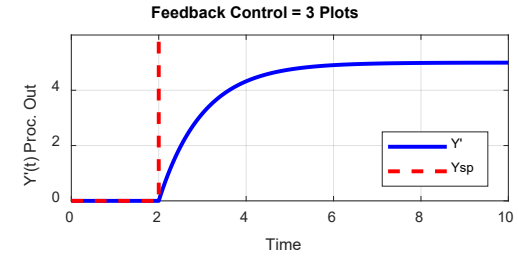
Studio 3: PI Controller for Kitchen Oven Control CHE 461

In this studio we will continue with our "Kitchen Oven" example from Studio 2 to keep exploring simple controllers and performing control loop performance "experiments." Last week we used a P-only controller to make setpoint changes, here you will investigate a PI-controller to carry out similar experiments.

In addition to all of the Studio 2 files, download these **4** files to your "current folder:"
`Studio3_Ponly.slx, Studio3_3.slx, PIpartsP.m, ISE1.p`

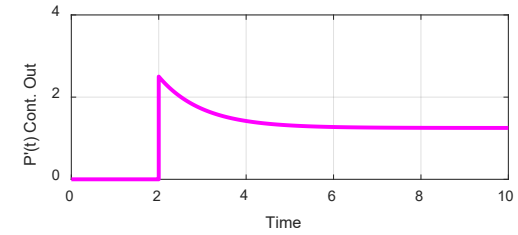
1. Review of P-only control for the Kitchen Oven control loop

a. Use the file you created last week (studio2b), or the provided file `Studio3_Ponly.slx`, to run a servo (setpoint) change of +10 degrees at time = 2 when your controller gain K_c = **0.25**. Create the YPE plots shown below by double-clicking on the Loop_1_plots "button box." **Answer the three questions below:**

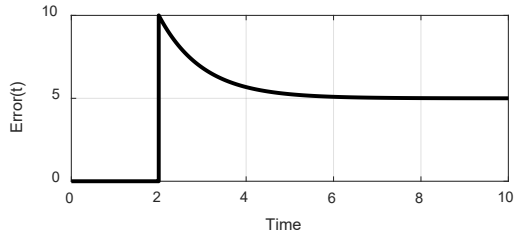


i. What is the mathematical definition of Error(t)?

E(t) = _____



ii. What is a ***key*** characteristic of the closed-loop response when a P-only controller is used? (Hint: this is a potential ***disadvantage***)



iii. List one or more potential ***advantages*** of using P-only control

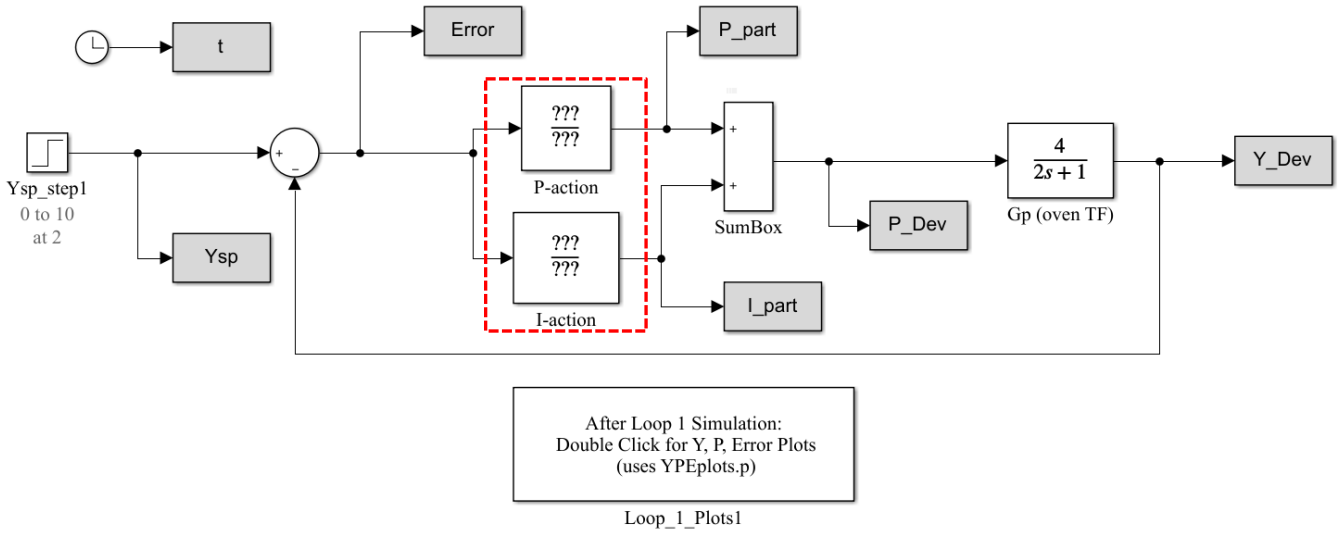
2. Add integral action to the control loop by adding a parallel I-action block.

a. The ideal PI controller can be written in "series" or "parallel" form (Chap. 8, pg 140). In either case there is no derivative action in the PID so τ_D in the PID equations is always 0.

PI controller:

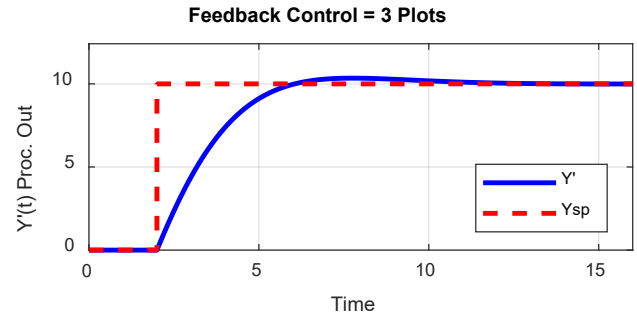
$$G_{c,\,\mathrm{PI}}(s) = K_c\left(\frac{\tau_I s + 1}{\tau_I s}\right) = K_c + \frac{K_c/\tau_I}{s} \quad \text{(parallel form)}$$

Modify the file you used in part 1 so that it uses the ***parallel*** form of a PI controller as shown in the block diagram below and **save it with the the new name: `Studio3_2.slx`.** Pay careful attention the new blocks required. For now, use these values for your PI controller: K_c = **0.25 and** τ_I = **1.25**.

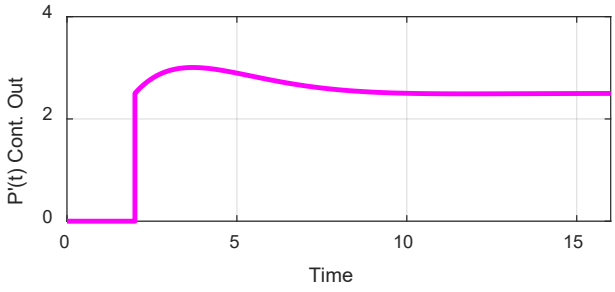


Note: The denominator polynomial in the I-action block is defined by the polynomial coefficients of [1.25 0]. **DO NOT** input the letter "s" in the denominator.

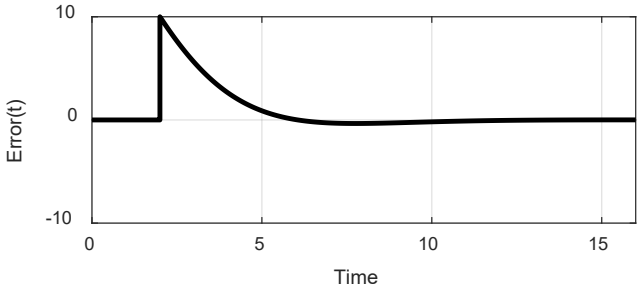
b. Run your new PI control model with simulation **run time = 16** and check that you get the **same three plots shown below (YPE plots).** Then answer the following three questions:



i. Compare the closed-loop performance of the PI controller versus the P-only controller. **How did I-action change the closed-loop response?**



ii. Recall from Studio 2a (open-loop tests) that the required power input for a 10 degree temperature change was 2.5 kW. **What is the controller output when P-only control is used (part 1a)? What is the controller output when P- and I-action are both used?**



iii. What is a key advantage of using a PI controller, rather than P-only?

c. Now to try "tease apart" the individual contributions of the P-action and I-action from your PI servo experiment that you ran in part (**2b**). Run the plotting program **`PIpartsP.m`**, which shows the P- and I-action separately, along with the combined controller output (the individual P and I contributions sum up to the total controller output *P'*). Inspect your plot and answer the following four questions:

i. How is this plot related to the three plots shown in "YPEplots.p?"

ii. From about time = 6 to 10, the value of the P-part is negative. Why is this the case? (**Hint**: look at the *Y'* and *Error* plots)

iii. At time = 2 (the moment the setpoint changed) what is the value of the P-part and the I-part? Briefly explain why this is the case.

P-part = _____ I-part = _____

iv. At steady state, what is the value of the P-part and the I-part? Briefly explain why this is the case.

P-part = _____ I-part = _____

3. Running simulations using the "PIDalpha" function block for a PI controller

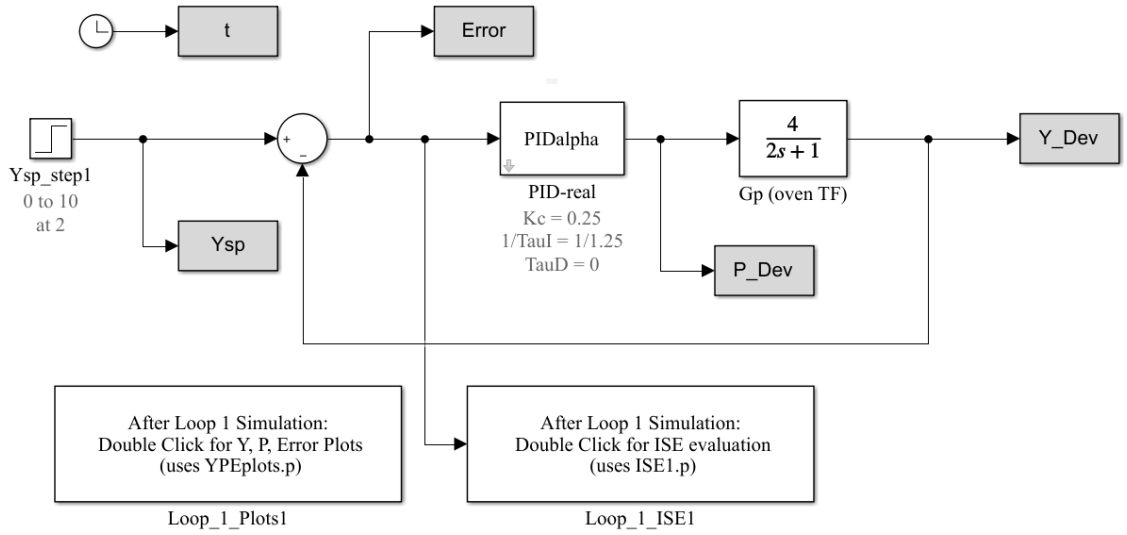
a. The standard way we will carry out PI or PID control simulations in CHE 461 is using the "PIDalpha" function block. This function block uses the series form of a PID controller shown below:

Commercial PIDalpha controller for CHE 461:

$$G_{c,\text{ PID-real}}(s) = K_c\left(\frac{\tau_I s+1}{\tau_I s}\right)\left(\frac{\tau_D s+1}{\alpha\tau_D s+1}\right) \text{ with } \alpha=0.05$$

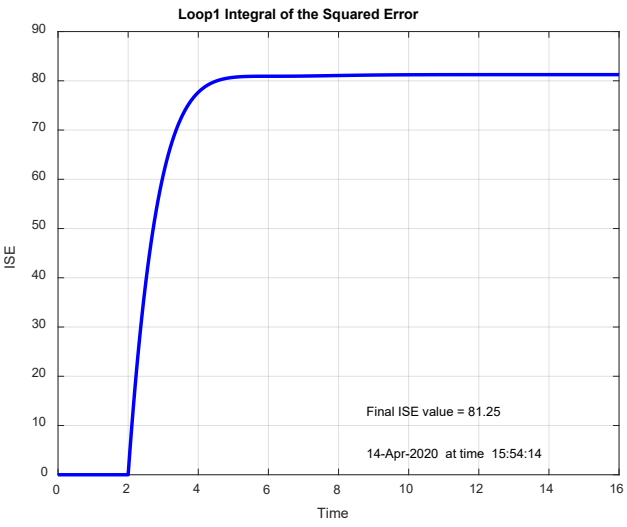
Open the file **`Studio3_3.slx`** and finish building the block diagram so it looks like the completed diagram below. You will need to copy/paste some blocks from **`Blocks461_S20.slx`**. Rename blocks as needed and connect blocks as shown. **Then save your file with a name of your choice**, e.g. Studio3_3_XX.slx where XX are the initials of your first and last name.

Use the same values that we used before for the PI controller: K_c **= 0.25,** τ_I **= 1.25,** τ_D **= 0 (no derivative action). Note: the value you input for the I-action is** $1/\tau_I$



b. Run your Simulink model (step *Ysp* to 10 at $t = 2$) and you should get the same three plots shown for part **2b**. Confirm that the simulation was setup correctly by double clicking the Loop_1_ISE "button box." You should get the same plot shown below with a value of **ISE = 81.25**. **Check this before continuing!**

The ISE = **I**ntegral of the **S**quared **E**rror. The value that is reported for ISE is the integral of the Error vs time curve (shown in YPE plots) from t = 0 to t = 16 (simulation time). **Recall the mathematical definition of E(t) in part 1**! The ISE is one metric that control engineers can use to evaluate the closed-loop controller performance. **Answer the following questions:**



i. Why does the ISE curve (shown to the left) initially increase, then eventually plateau to a constant value?

ii. For "good" control performance, would you desire a ***low*** or a ***high*** ISE value? Why?

iii. What does the ISE curve look like if you turn off the I-action (use a P-only controller)? Try it! (set to 0 in PIDalpha block). Why does the ISE curve have this shape?

c. Reset your PID controller to the **BASE CASE: $K_c = 0.25$, $\tau_I = 1.25$, $\tau_D = 0$,** then run the following experiments and answer the questions for each simulation:

i. More Proportional Action. Increase P-action by making K_c a factor of 4 larger ($K_c = 1$). Keep integral and derivative action the same ($\tau_I = 1.25$, $\tau_D = 0$). Look at YPE and ISE plots.

Relative to the **BASE CASE…**

What effect did this have on the closed-loop response (y' vs t)? (**Check ISE = 23.12**). Do you think closed-loop performance improved? Why?

ii. More Integral Action. Reset your controller to have $K_c = 0.25$. Now increase the I-action by ***reducing*** τ_I by a factor of 5 ($\tau_I = 0.25$).

Relative to the **BASE CASE…**

What effect did this have on the closed-loop response (y' vs t)? (**Check ISE = 56.25**) Do you think closed-loop performance improved? Why?

iii. Less Integral Action. Keeping $K_c = 0.25$, reduce the I-action by ***increasing*** τ_I by a factor of 5 ($\tau_I = 6.25$).

Relative to the **BASE CASE…**

What effect did this have on the closed-loop response (y' vs t)? (**Check ISE = 194.01**) Do you think closed-loop performance improved? Why?