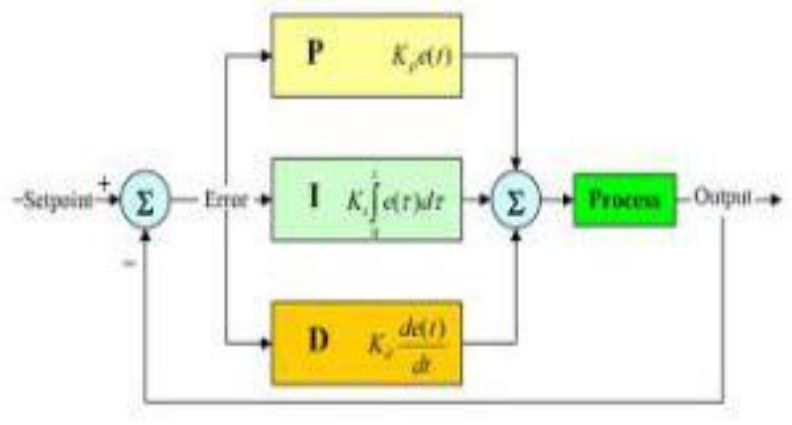
PID Loop Tuning 101



Introduction

I found myself reviewing the basic method I use to tune a PID loop. I would like to post my method here roughly as it appeared in my response to a question posed in the LinkedIn "Automation Engineers" discussion grouplinked in my "PID Tuning Discussion" post.
This was in response to tuning a temperature loop, but the idea is the same for most types of basic PID type loops. A classic (or standard) PID algorithm (shown below) is assumed, (not the parallel or independent algorithm in the graphic above.. It just looked nice).

$$\mathrm{MV(t)} = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{d}{dt} e(t) \right)$$

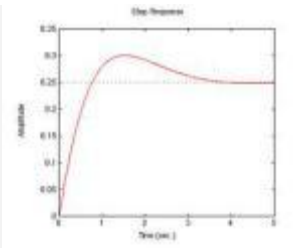
The classic PID algorithm

PID Loop Tuning 101:

We will need one key tool to do the tuning. We must be able to monitor the process temperature when a step change to the setpoint is made. We can do this by hand, recording the time and SP (setpoint temp) and PV (measured Temperature). Since this can take hours on a typical temperature loop some kind of trending tool will be most helpful. The purpose of this is to monitor the response to the step changes to the setpoint you will be making.

Step 1:

For starters I tend to use a gain of 1 or 2 (P = 1 or 2) and an I and D of "0". What a gain of 1 or 2 means varies by controller, so I adjust the P value until I see a full output from the PID controller (CV = 100%) when the error (SP-PV) is at a fairly large variation. In the case of a 0 - 200 DegC temperature loop, adjust the P value until a 50 DegC temperature difference (SP – PV) equals 100% output (CV =100).



Something like this is the result of Step 2 tuning.

Looking at the response will tell you whether to add more gain or less. If the temperature stabilizes without overshoot (overshoot is a temperature that is above the final stable temperature. On a trend graph this is readily apparent. (See graph at the right for an example of overshoot).

Step 2:

Increase the "P" value until you get some overshoot where the temperature settles down fairly quickly. (Quickly means within 2 or 3 overshoot cycles. This can still take hours however). If the temperature never stops oscillating then there is too much gain.

Step 3:

Reduce the gain by 25 – 50% from the value set in step 2. This will make for improved loop stability and robustness to small process changes over time.

The reduced P and final tuning trend should look like this.

Step 4:

Add Integral or "I" to your PID controller to get to setpoint. (Up until now we have not worried about getting to setpoint, only about the correct response and getting as much gain on the system as possible for stability).

The amount of Integral gain depends on the response observed in your system. For set it and forget it tuning, most of the time one can just put in an integral term that allows the output to change by 10 – 20% of its total range in the response time of the system.

The response time of the system is normally when the PV gets to 67% of the final value.

However I tend to use the full response time, so more like getting to 95% – 98% of the final temperature. For example if it takes 4 hours for the temperature to stabilize after making a setpoint change, then adjust the integral term so it adds 20% to the output in 4 hours.

Step 5:

Basic tuning is done. About 90% of the time, you'll have a stable loop that gets to setpoint usually within the response time of the system.

Step 6 and beyond:

If this loop gets to setpoint to slowly, more integral can be added but this requires additional time which if you have it great. I would advise to tune these slow types of loops as early as possible and to have the trend tools necessary to monitor any new tuning and step change responses while you are doing other things. In this way you'll have more time for experimenting to "optimize" the tuning before you need to move on.

Notes:

Disclaimer: There are other ways to tune, some have been mentioned in the LinkedIn discussions. This "closed loop" method represents little more then the basics. However it is a common method that is used by experienced PID loop tuners to get close quickly when no other tools are available. For this author other tools are typically not available, nor are they needed in most applications.

For temperature loops many in the LinkedIn forums have suggested some "D" to stabilize the loop. I consider this advanced stuff and suggest to not add any "D" unless you live in the plant and are available to keep an eye on things, especially when something changes in the process operations. Derivative is also often suggested if there is dead time in the system. Again the same cautions. I prefer to detune the loop even further over adding "D" wherever possible.

Also, if some kind of trending tool where you can monitor the temperature is not available, it might be better to start by tuning a faster loop like a pressure or flow loop. With these you can see the response as you watch…. a trending tool while still helpful, is usually not needed.

Sorry if anyone thinks that you can predict the needed tuning parameters without any process knowledge. Tuning a PID loop isn't based on just numbers, it's based on knowing your process. Fortunately knowing the process usually just means measuring the response to a step change in the setpoint or some other "upset", and we typically just measure it on the actual system.

For actual parameters, no one will be able to do any better than the P, I, and D parameters you will find in a working controller.

For those that can experiment with a working system, one can try changing the setpoint and measuring the response This will tell you whether you can try adding (or if you need to reduce) the gain.

One more note: If you change the "P" gain after you have added an I value, the I value will produce a different response. In the classic PID control, the gain is multiplied by each of the 3 terms (P, I, and D). Fortunately this is what is usually desired as the Integral and Derivative terms scale with the Proportional term.

Alternative Methods of Tuning:

One LinkedIn commentator has disagreed with me on my approach, preferring primarily to tune using the "open loop" method first. I was unfamiliar with this method and I thank Michael Taube for educating me on this. I certainly intend to try this method on an upcoming temperature control job where I'll need to tune about 100 loops. In controlling drive and motion control loops such as dancer and tension loops in web lines, I don't see how one could tune such loops except using closed loop tuning. Open loop tuning would make everything go out of control before you could get any useful data.

Additional Links to P&ID Tuning:

1. "Model Based Tuning Methods" Contains a useful "PI tuning map" that shows the effects of increasing or decreasing P or I from an "optimal" setting.
2. "Open Loop & Closed Loop Tuning Rules" by Apco Inc. I especially liked the concise summary on Open Loop Tuning in this article.
3. "PID Loop Tuning Pocket Guide Version 2.2" by ControlSoftInc.com; Describes the relationship between PID terms, names and describes common PID controller implementations, describes basic closed loop and open loop tuning methods.
4. "Closed Loop vs Open Loop Tuning" by Michel Ruel of Top Control Inc.
5. "PID Controller" Wikipedia Article shows the standard/classical and parallel/independent forms of the equation and some interesting history.
6. "PIDTuningClassical" Wikipedia Article with some nice graphics and examples on tuning the classical PID algorithm
7. "PID Algorithm and Tuning Methods" by John A. Shaw, Process Control Solutions; Rochester NY; Looks like a nice tutorial.

Source: http://hennulat.wordpress.com/2011/01/12/pid-loop-tuning-101/