

The Ever-present PID Loop Effect

by Paul Avery and Todd Rohde

A control loop is something that everyone uses every day but few think about. Anytime that you adjust what or how you are doing something based on how things turned out the last time you did it, then you are forming your own control loop. For example, when you when you want to drive your car at 65 MPH, you press down the accelerator until the speedometer tells you that you have achieved your goal. Simple, right? But what happens when you start driving up a hill at 65 MPH? The car starts to slow down because the torque required to move the car at 65 MPH on flat road is no longer enough so you respond by pressing the accelerator further down. Your foot, the speedometer and your brain have formed a control loop. American cars since 1958 have offered an automated control loop for speed more commonly known as cruise control.

Modern industrial controls are regularly required to regulate processes by being part of a control loop. The controller will receive a setpoint (SP) request from programmer and will compare that setpoint to a measured feedback (FB). The setpoint can be thought of as WHERE I WANT TO BE and the feedback can be thought of as WHERE I REALLY AM. The difference between the setpoint and feedback is called the error (ϵ). The job of the controller is to eliminate the error. This means that WHERE I AM IS WHERE I WANT TO BE!

What is a PID Controller?

Where does the PID come in? First PID is an acronym for the words **P**roportional, **I**ntegral and **D**erivative, which are actually mathematically terms. Proportional means constant multiple. A number is said to be a proportion of another if there exists a constant (n) such that:

$$y = nx$$

Of course n can be positive, negative, greater than one, or less than one. To make the formula more accurate by PID controller standards, the proportion would be given by K_P and the x term would be the control loop error (ϵ).

$$y = K_P(\epsilon)$$

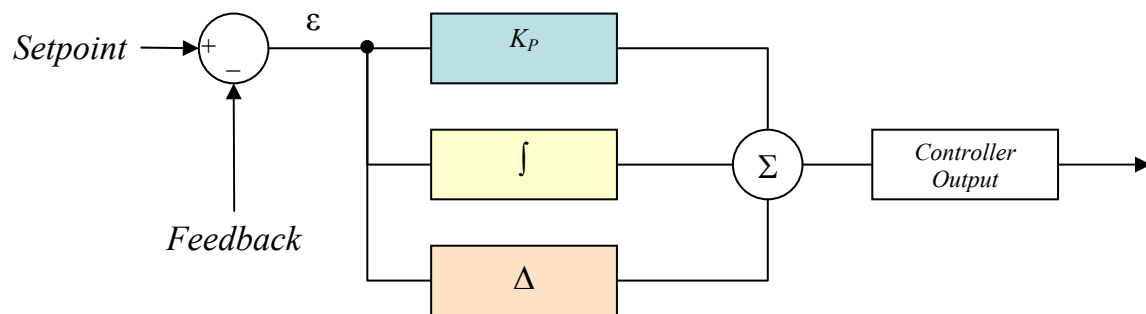
The term Integral means the summation of a function over a given interval. In the case of controller PID that would be the sum of the error over time.

$$y = \int f(\varepsilon) dt$$

Finally, the Derivative means the rate of change during a given interval. Again, interpreted by a PID controller:

$$y = \frac{d(\varepsilon)}{dt}$$

As we can tell from the formulas, all three of the components of the PID controller are creating an output based on the measured error of the process being regulated.



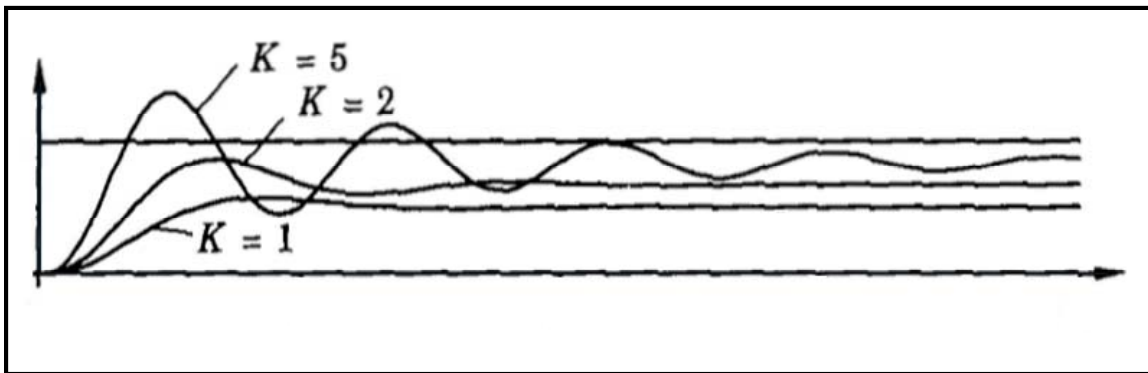
If the control loop functions properly any changes in error caused by setpoint changes or disturbances to the process will quickly be eliminated by the combination of the three factors P, I and D. Lets analyze how each of these factors contributes to the performance individually.

Proportional

It is pretty safe to say that the proportional factor is the easiest to understand. The output of the proportional factor is the product of the gain and the measured error (ϵ). Hence, the larger the proportional gain or the error, the greater the output from the proportional factor. Setting the proportional gain too high will cause the controller to repeatedly overshoot the desired setpoint to the point that the loop output is described as oscillating.

The down side to a “proportional only” loop is that when the error becomes too small, the loop output will become negligible. Therefore even when the proportional loop reaches a steady state there will still be error.

The larger the proportional gain the smaller the steady state error will be, but it will always leave some steady state error. The larger the proportional gain though, the more prone the loop is to become unstable. This leaves inevitable steady state error that is referred to as **offset**.

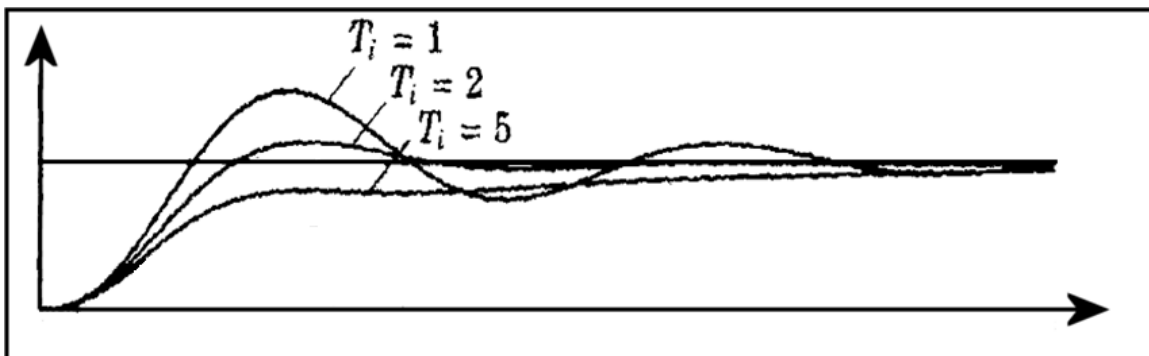


Integral

The integral factor is like a basket where the loop stores all of the measured error ($\int \epsilon$). Remember that the error can be positive or negative, so sometimes the error will be filling the basket (when positive error is added on top of positive error or negative error is added on top of negative error) and sometimes it will be emptying it (when positive error is added to negative error or vice versa).

When the integral is doing its job in the control loop, the basket should be nearly empty. Even when the error is so small that the proportional factor is no longer effective, the integral is still hard at work saving the error until it is larger enough to matter. Therefore, it is part of the integral's job to eliminate steady state **offset**.

In fact, most control loop action at steady state is all due to the integral factor. This can be proven by controllers that feature an integral reset input. Resetting the integral when the loop is in steady state, will cause the controller output to momentarily drop to zero as the integral basket is emptied. The downside to the integral factor is that it will strongly contribute to the controller output shooting past the desired setpoint referred to as **overshoot**. The shorter the Integral time the more aggressive the Integral will work to eliminate error.

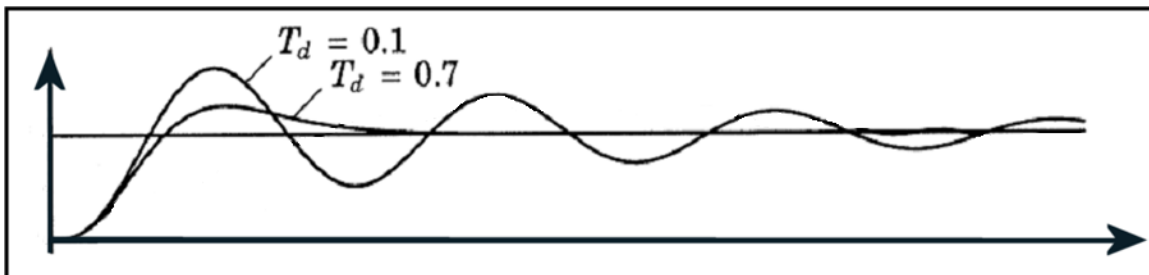


Derivative

The derivative factor is the least understood and the least used of the three factors. In fact, a majority of PID loops in the real world are really just PI loops. That does not negate the fact that there are certain applications and certain instances where derivative will play a very important role.

The proportional looks at the instance of error and the integral looks at the accumulation of error, the derivative is concerned with the error now versus the error the last time it was checked.

In other words, while the derivative is looking at the rate of change of the error ($\Delta\epsilon$). The more the error has changed or the higher the derivative gain, the larger the derivative factor. The effect of the derivative is to counteract the **overshoot** caused by other two factors, P and I. When the error is large, the P and the I will push the controller output. This controller response will make the error change quickly, which in turn causes the derivative to more aggressively counteract the P and the I. A properly used derivative will allow for more aggressive proportional and integral factors. The larger the Derivative time the more aggressively the Derivative will dampen the P and the I.



What a Controller Does and How it Works?

How a PID controller works is a pretty easy concept to understand. The PID controller looks at the current value of an error (ϵ), the integral of the error over a time interval ($\int \epsilon$) and the rate of change of the error ($\Delta \epsilon$) to determine how much of a correction to apply. The controller will continue to apply the correction, until a change is seen on the feedback. Depending on the update rate of the error calculation, which in turn may depend on how often the loop feedback is read, the corrective action can be adjusted at a very fast rate. For instance the analog feedback on a Yaskawa P7 VFD is updated every 10 mS.

A PID controller's job is to force the feedback to match the setpoint. Sometimes the error between the feedback and the setpoint is caused by a change in the setpoint, but in most applications the setpoint is not adjusted very often. More than likely the error in the loop will be caused by disturbances to the measured feedback.

In our example at the beginning, the disturbance to the cruise control regulation was any hill (or valley) that may be encountered on the road the car is traveling. Other examples of disturbances are double doors opening in a building where pressure is being regulated or people taking showers and baths while a control loop is trying to regulate the level in a water tank.

What are the Benefits of PIDs?

The main benefit for any PID loop is the idea that we can SET IT AND FORGET IT, while still maintaining a well-regulated system. PID is so universal that PI and PID loops can be small and fast like a current regulating loop inside a servo drive, vector controller or a slower loop regulating the liquid level in a giant tank that holds hundreds of thousands of gallons. In all reality almost anything that can be measured and regulated and a PID loop is one of the simplest, but effective means to achieve that regulation. Frankly, if PID didn't already exist, we would be forced to invent it or factory automation would be very limited.

PID loops provide technicians and engineers with a customizable way to control a wide variety of different conditions from temperatures to speeds to everything in between. The loop's control is used to modify an application's behavior in efforts of keeping the output at a stable rate and improve the response rates. The biggest advantage here is that it doesn't take nearly as long to accomplish this thanks to using special software and computers that can do the work for us.

Techniques for Fine Tuning a PID Loop

Tuning is part of the design of the loop and it may need to be tuned if it oscillates too much, responds too slowly, has a steady-state error or is unstable. One must be very careful when determining whether or not a PID has to be tuned or not. An individual should remember to always check the hardware first, as this could be the problem and not that the controller has to be tuned. A PID will most likely have to be tuned if the operator thinks that the controller can perform better, the process dynamics weren't well understood when the gains were first set or the dynamics were changed, some characteristics of the control system are direction dependent or careful consideration wasn't given to the units of gains and other parameters. On the same note, it may not have to be tuned if a control valve sticks, measurement taps are plugged, sensors are disconnected or if a control valve is stripped out from high-pressure flow.

Some systems have interactions of all different strengths, which in turn affect the tuning of a PID. There is no single definition of best tuned that applies to all loops, therefore no single tuning method will tune all of the loops optimally.

Tuning a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. The optimum behavior on a process change or setpoint change varies depending on the application. Some processes must not allow an overshoot of the process variable from the setpoint, while others must minimize the energy expended in reaching a new setpoint. Generally, stability of response is required and the process must not oscillate for any combination of process conditions and setpoints. Tuning of loops is made more complicated by the response time of the process, as it may take minutes or hours for a setpoint change to produce a stable effect. Some processes have a degree of non-linearity and so parameters that work well at full-load conditions don't work when the process is starting up from no-load.

There are several methods for tuning a PID loop and the choice of method largely depends on whether or not the loop can be taken "offline" for tuning, and also the response speed of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time and using this response to determine the control parameters.

If the system must remain online, one tuning method is to first set the I and D values to zero and increase the P until the output of the loop oscillates. Then increase I until oscillation stops. Finally, increase D until the loop is acceptably quick to reach its reference. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly.

Another tuning method is known as the “Ziegler-Nichols method,” introduced by John G. Ziegler and Nathaniel B. Nichols of Taylor Instruments back in 1942. This technique also involves setting the I and D gains to zero and then increasing the P gain until the output of the loop starts to oscillate. Document the critical gain (K_c) and the oscillation period of the output (P_c) before adjusting the P to $0.5 \cdot K_c$, the I to $0.45 \cdot K_c$ and the D to $0.6 \cdot K_c$ (see table below). This is a proven online method that will be adequate for process loops, where quarter wave decay is acceptable.

Ziegler–Nichols Method			
Control Type	K_p	K_i	K_d
<i>P only</i>	$0.5 \cdot K_c$	-	-
<i>PI</i>	$0.45 \cdot K_c$	$1.2K_p / P_c$	-
<i>PID</i>	$0.6 \cdot K_c$	$2K_p / P_c$	$K_p P_c / 8$

The above graph shows what occurs during the Ziegler-Nichols method of tuning.

Most modern industrial facilities no longer tune loops using the manual calculation methods, but instead use tuning and loop optimization software. These software packages will gather the data, develop process models, suggest optimal tuning and even develop tuning by gathering data from reference changes. This is a very consistent tuning method that can be done both online and offline. It also may include valve and sensor analysis and allows for simulation before downloading. The only real drawback is that it is somewhat costly and involves some training.

Then there is the analytical approach that involves mathematics. PID loop tuning induces an impulse in the system, and then uses the controlled system’s frequency response to design the PID loop values. In loops with response times of several minutes, mathematical loop tuning is recommended because trial and error can literally take days just to find a stable set of loop values. Optimal values are harder to find, but can save a company huge amounts of money. Commercial software is available from several sources, and can easily pay for itself if a PID loop runs a large or expensive process. Some digital loop controllers offer a self-tuning feature in which very small setpoint changes are sent to the process, allowing the controller itself to calculate optimal tuning values.

One can also tune by feel, which is an online method that doesn't require any math. The main problem with this method is that it is erratic, not repeatable and can be inefficient. The final method of tuning is a quality process model called the Cohen-Coon, which is a modified version of the Ziegler-Nichols approach. It's an offline method that involves some math, but it's only good for the first-order process. Under manual mode, wait until the process is at a steady state, before introducing a step change in the input. Based on the output, obtain an approximate first-order process with a time constant delayed by units from when the step was introduced, when the half point occurs and when the 63.2% point occurs. From the measurements based on the step test evaluate the process parameters. And finally based on the parameters, the formulas should prescribe the controller parameters.

What Kind of Applications Use PID Loops?

A PID controller can be used to control any measurable variable, which can also be affected by manipulating some other process variable. To put it more bluntly, anything that can be measured and manipulated is eligible for a PID loop. Think of the air pressure within a lengthy duct work. A simple pressure sensor can be used to measure the duct's pressure and anything that can increase the pressure in the duct, such a variable speed fan or solenoid controlled dampers, can be used to control the pressure in the duct. Viola, all the ingredients needed for PID control. Other typical PID loop targets are temperature, flow rate, chemical composition, speed or level.

Most PID loops are single loop setups but some control systems arrange PID controllers in cascades or networks. That is, a "master" control produces signals used by "slave" controllers. Coupled and cascaded controls are common in chemical process control, heating, ventilation, and air conditioning systems, and other systems where many parts cooperate.

Summary

All in all, PIDs play a pivotal role in many aspects of our lives, by making it much easier to regulate and control things. By breaking down the concept behind the P, the I, and the D, and taking a look at what they do and how they work, it gives one a much better understanding, perspective and appreciation of their importance. Each part of PID functions as a tool of regulation, with each having a specific purpose. Sometimes only one is necessary for proper regulation and sometimes only all three will properly provide the functionality necessary for a successful application.