Everyone uses control loops. Anytime you adjust how you do something based on previous results, you are forming your own control loop.

For example, when you want to drive your car at 65 mph, you depress the accelerator until the speedometer reports the target speed — simple. But what happens when you start driving up a hill? The car slows because the torque to move a 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 have offered automated speed control loops — also known as cruise control — since 1958.

Similarly, modern industrial controls are often required to regulate processes as part of a control loop. The controller receives a setpoint request from the programmer and compares it to a measured feedback. The setpoint can be thought of as where I want to be and feedback can be thought of as where I really am. The difference between the setpoint and feedback is called the error $\varepsilon$. The job of the controller is to eliminate the error — so where I am is where I want to be.

**How do PID loops work?**

PID is an acronym for the mathematical terms Proportional, Integral, and Derivative. Proportional means a constant multiple. A number is said to be a proportion to another if there exists a constant $n$ such that $y = nx$. This $n$ can be positive or negative, greater or less than one. To make the formula more accurate by PID controller standards, proportion is given by $K_p$ and the $x$ term is

**PID benefits almost anything measurable and regulable**

The main benefit of any PID loop is that a designer can “set it and forget it” while still maintaining a well-regulated system. PID control is so universal that PI and PID loops can be small and fast like a current-regulating loop inside a servo drive or vector controller, or a slower loop regulating the liquid level in a giant tank holding thousands of gallons. PID loops are one of the simplest yet most effective means to achieve that control on almost anything measurable and regulable. 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 variety of conditions, from temperature to speed and everything in between. The loop’s control is used to modify application behavior to keep output at stable and improve response rates. What’s particularly exciting is that special software and computers can perform calculations to make PID design easier.
the control loop error $\varepsilon$: $y = K_p(\varepsilon)$.

The term Integral means the summation of a function over a given interval. In the case of controller PID that is the sum of error over time: $y = \int f(\varepsilon)dt$.

Finally, Derivative is the rate of change during a given interval. Interpreted by a PID controller:

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

All three of these PID controller components create output based on measured error of the process being regulated. If a control loop functions properly, any changes in error caused by setpoint changes or process

-17
MARCH 2009
MOTION SYSTEM DESIGN • www.motionsystemdesign.com

PID s play a pivotal role in making it easier to regulate and control things. Each part of a PID loop functions as a tool of adjustment, with each having specific purpose. Sometimes only one is necessary for proper regulation and sometimes only all three properly provide functionality necessary for a successful application. Above, loaded with PID loop functions, some controllers steer up to 256 rotary and linear axes.
disturbances are quickly eliminated by the combination of the three factors P, I, and D. Let us now analyze how each contributes to performance.

**Proportional factor**

The proportional factor is easiest to understand: The output of the proportional factor is the product of gain and measured error \(\varepsilon\). Hence, larger proportional gain or error makes for greater output from the proportional factor. Setting the proportional gain too high causes a controller to repeatedly overshoot its setpoint.

The downside to a proportional-only loop is that when error becomes too small, loop output becomes negligible. Therefore, even when the proportional loop reaches steady state, there is still error. The larger the proportional gain, the smaller the steady state error — but the larger the proportional gain, the more likely the loop is to become unstable. This dilemma leads to inevitable steady-state error called offset.

**Integral factor**

Think of the integral factor as a basket in which the loop stores all measured error \(\int \varepsilon\). Remember that error can be positive or negative, so sometimes error fills the basket (when positive error is added to positive error or negative error is added to negative) and sometimes it empties the basket — as when positive error is added to negative, or vice versa.

When the integral factor functions properly in the control loop, the basket is nearly empty. Even when error is so small that the proportional factor is no longer effective, the integral is still hard at work, collecting error until it is large enough to matter — as part of the integral’s function is to eliminate steady-state offset.

In fact, most control loop action at steady state is due to the integral factor. Controllers that feature integral reset prove it: Resetting the integral when a loop is in steady state causes controller output to momentarily drop to zero as the integral “basket” is emptied.

The downside to the integral factor is that it strongly contributes to controller output overshoot past the target setpoint. The shorter the integral time, the more aggressively the integral works.

**Derivative factor**

The derivative factor is the least understood and 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 in which the derivative plays a very important role. The proportional corrects instances of error, the integral corrects accumulation of error, and the derivative corrects present error versus error the last time it was checked.

In other words, the derivative is looking at the rate of change of the error \(\Delta \varepsilon\). The more error changes or the longer the derivative time, the larger the derivative factor becomes. The effect of the derivative is to counteract the overshoot caused by P and I. When the error is large, the P and the I will push the controller output. This controller response makes error change quickly, which in turn causes the derivative to more aggressively counteract the P and the I. A properly used derivative allows for more aggressive proportional and integral factors. Larger derivative time makes the derivative more aggressively dampen P and I.
Techniques for fine tuning

Tuning a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to optimum values for a target response. Tuning is part of loop design, usually required if the system oscillates too much, responds too slowly, has steady-state error, or is unstable.

One must be careful when determining whether a PID needs tuning or not: Always check hardware first, as it could be the problem and not the controller that needs tuning. A PID most likely needs tuning if:

• The operator thinks that the controller can perform better
• Process dynamics weren't well understood when gains were first set
• Dynamics were changed
• Some control-system characteristics are direction dependent, or
• Careful consideration wasn't given to the units of gains and other parameters.

On the other hand, problems may have nothing to do with the PID loop tuning 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 widely varied strengths — and there is no single definition of “best...
Integral: Eliminates steady-state offset

The integral factor is like a basket that stores all measured error. When the integral is functioning correctly, the basket is always nearly empty.

The integral factor is like a basket that stores all measured error. When the integral is functioning correctly, the basket is always nearly empty.

tuned” that applies to all loops, so no single method optimizes all loops.

The optimum behavior in a process or setpoint change depends on the application. Some processes must not allow overshoot of a process variable, while others must minimize energy expended in reaching a new setpoint. Generally, response must be stable and systems must not oscillate for any combination of conditions and setpoints. Tuning of loops is further complicated by process response time, as it may take minutes or hours for a setpoint change to produce a stable effect. Some processes also exhibit nonlinearity, so parameters that work well at full-load conditions don’t work when starting from no-load.

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

If the system must remain online, one tuning method is to first set I and D values to zero and increase P until loop output oscillates — then increase I until oscillation stops, and increase D until the loop is accept-

Derivative: Pattern correction

The derivative factor is the least understood and used of the three factors. It accounts for and corrects present error versus error at last check.
ably quick in reaching its reference. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly.

Another is known as the Ziegler-Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols of Taylor Instruments in 1942. This technique also involves setting I and D gains to zero and then increasing P gain until the loop output starts to oscillate. Document critical gain $K_c$ and the oscillation period of the output before adjusting P to 0.5. K, I to 0.45 $K_c$, and D to 0.6. $K_c$. This proven online method is adequate for loops where quarter-wave decay is acceptable.

In fact, most industrial facilities no longer tune loops with manual calculation, but use tuning and loop optimization software. These software packages gather data, develop process models, suggest optimal tuning, and even develop tuning by gathering data from reference changes. This can be done both on and offline. It may also include valve and sensor analysis, and simulation before downloading. The only drawbacks: Software is somewhat costly and involves some training.

The analytical approach involves mathematics. PID loop tuning induces an impulse in the system, and then uses the controlled system’s frequency response to design 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 large or expensive processes. Some digital controllers even feature self tuning, in which small setpoint changes are sent to the process, allowing the controller itself to calculate optimal values.

One can also tune by feel, which is an online method that doesn’t require 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. This offline method involves some math, but is 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. From the measurements based on the step test, evaluate the process parameters. Based on these, formulas should prescribe controller settings.

For more information, call (800) 927-5292 or visit yaskawa.com.

### Ziegler–Nichols method

<table>
<thead>
<tr>
<th>Control type</th>
<th>$K_p$</th>
<th>$K_i$</th>
<th>$K_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>P only</td>
<td>0.5 $K_c$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>PI</td>
<td>0.45 $K_c$</td>
<td>1.2 $K_p / P_c$</td>
<td>-</td>
</tr>
<tr>
<td>PID</td>
<td>0.6 $K_c$</td>
<td>2 $K_p / P_c$</td>
<td>$K_p P_c / 8$</td>
</tr>
</tbody>
</table>

Mathematical gyrations occur during Ziegler-Nichols tuning. With this technique, I and D gains are set to zero and then P gain is increased until the loop output starts to oscillate.