Fixing PID

Proportional-integral-derivative controllers may be ubiquitous, but they're not perfect.

Vance VanDoren, PhD, PE for Control Engineering

11/30/2012

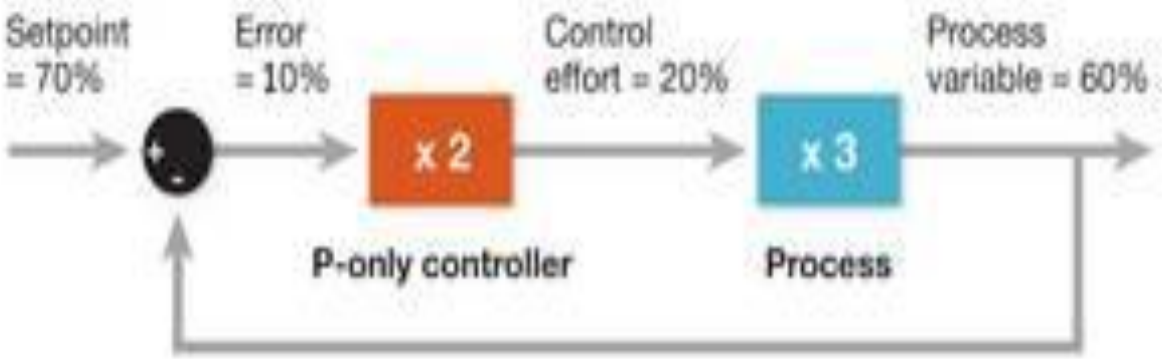
Although the PID algorithm was first applied to a feedback control problem more than a century ago and has served as the de facto standard for process control since the 1940s, PID loops have seen a number of improvements over the years. Mechanical PID mechanisms have been supplanted in turn by pneumatic, electronic, and computer-based devices; and the PID calculation itself has been tweaked to provide tighter control.

Integral action or automatic reset was the first fix introduced to improve the performance of feedback controllers back when they were equipped with only proportional action. A basic "P-only" controller applies a corrective effort to the process in proportion to the difference or error between the desired setpoint and the current process variable measurement.

When the setpoint goes up or the process variable goes down, the error between them grows in the positive direction, causing a P-only controller to respond with a positive control effort that drives the process variable back up towards the setpoint. In the converse case, the error grows in the negative direction and the controller responds with a negative control effort that drives the process variable back down.

A P-only controller is intuitive and easy to implement, but it tends to reach a point of diminishing returns. As the error decreases with each new control effort, so does the next control effort. That in turn slows the rate at which the error decreases until it ceases to change altogether. Unfortunately, that

steady-state error is never zero. A P-only controller always leaves the process variable slightly off the setpoint. See the “Steady-state error” graphic for a demonstration of why this happens.



In this simple example of a generic control loop, a proportional-only controller with a gain of 2 drives a process with a steady-state gain of 3. That is, the controller amplifies the error by a factor of 2 to produce the control effort, and the process multiplies the control effort by a factor of 3 (and adds a few transient oscillations) to produce the process variable. If the setpoint is 70%, the process variable will end up at 60% after the transients die out. The error remains nonzero, yet the controller stops trying to reduce it any further.

Integral action operating in parallel with proportional action eliminates that steady-state error. It increases the control effort in proportion to the running total or integral of all the previous errors and continues to grow so long as any error remains. As a result, a proportional-plus-integral or “PI” controller never gives up. It continues to apply an ever-increasing control effort until the process variable reaches the setpoint.

Unfortunately, the introduction of integral action causes other problems that require their own fixes. Arguably the most significant is an increased risk of closed-loop instability where the integral action's persistence does more harm than good. It can grow so large that it forces the process variable to overshoot the setpoint.

If the controlled process happens to be particularly sensitive to the controller's efforts, that overshoot will cause an even larger error in the opposite direction. The controller will eventually change course in an effort to eliminate the new error but will end up driving the process variable back the other way, ending up even further from the setpoint. Eventually, the controller will start oscillating from fully on to fully off in a vain effort to reach a steady state.

The simplest solution to this problem is to reduce the amplification factor or gain by which the controller multiplies the integrated error to generate the integral action. On the other hand, reducing the integral gain risks increasing the time required for the closed-loop system to reach a steady state with zero error. Hundreds of analytical and heuristic techniques have been developed over the years to determine values for the integral and proportional gains that are just right for a particular application.

Reset windup

Integral action can also cause reset windup when the actuator is too small to implement an especially large control effort requested by the controller. That can happen when a burner isn't large enough to supply the necessary heat, a valve is too small to generate a high enough flow rate, or a pump reaches its maximum speed. The actuator is said to be saturated at some limiting value—either its maximum or minimum output.

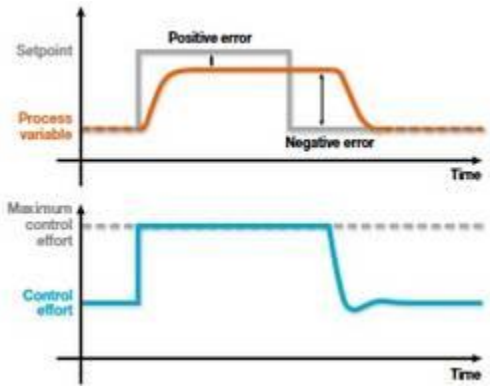
When actuator saturation prevents the process variable from rising any further, the controller continues to see positive errors between the setpoint and the process variable. The integrated error continues to rise, and the integral action continues to call for an increasingly aggressive control effort. However, the actuator remains stuck at its maximum output, unable to affect the process any further, so the process variable doesn't get any closer to the setpoint.

An operator can try to fix the problem by reducing the setpoint back into the range that the actuator is capable of achieving, but the controller will not respond because of the enormous value that the total integrated error will have achieved during the time that the actuator was saturated fully on. That total will remain very large for a very long time, no matter what the current value of the error happens to be. As a result, the integral action will remain very high, and the controller will keep pushing the actuator against its upper limit.

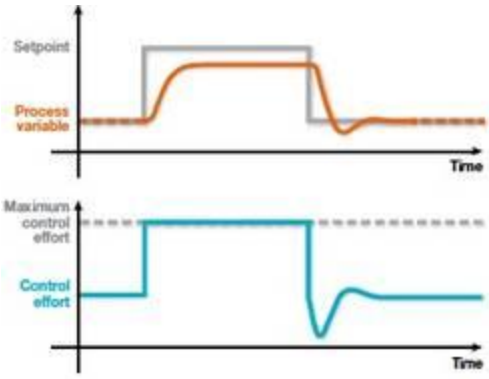
Fortunately, the error will turn negative if the operator drops the setpoint low enough, so the total integrated error will eventually start to fall. Still, a long series of negative errors will be required to cancel the long series of positive errors that had been accumulating in the integrator's running total. Until that happens, the integral action will remain large enough to continue saturating the actuator, no matter what the simultaneous proportional action is calling for. See the "Reset windup" graphic.

Several techniques have been devised to protect against reset windup. The obvious solution is to replace the undersized actuator with a larger one capable of actually driving the process variable all the way to the setpoint or

selecting setpoints that the existing actuator can actually reach. Otherwise, the integrator can simply be turned off when the actuator saturates.



In this example, the operator has tried to increase the setpoint to a value higher than the actuator is capable of achieving. After observing that the controller has been unable to drive the process variable that high, the operator returns the setpoint to a lower value. Note that the controller does not begin to lower its control effort until well after the setpoint has been lowered because the integral action has grown very large during the controller's futile attempt to reach the higher setpoint. Instead, the controller continues to call for a maximum control effort even though the error has turned negative. The control effort does not begin to drop until the negative errors following the setpoint change have persisted as long as the positive errors had persisted prior to the setpoint change (or more precisely, until the integral of the negative errors reaches the same magnitude as the integral of the positive errors).



In this case, the operator has repeated the same sequence of setpoint moves, but this time using a PID controller equipped with reset windup protection. Extra logic added to the PID algorithm shuts off the controller's integrator when the actuator hits its upper limit. The process variable still can't reach the higher setpoint, but the controller's integral action doesn't wind up during the attempt. That allows the controller to respond immediately when the setpoint is lowered.

Pre-loading

Reset windup also occurs when the actuator is turned off but the controller isn't. In a cascade controller, for example, the outer-loop controller will see no response to its control efforts while the inner loop is in manual mode. If the outer-loop controller is left operating during that interval, its integral action will "wind up" as the error remains constant.

Similarly, when the burner, valve, or pump is shut down between cycles of a batching operation, the process variable is prevented from getting any closer to the setpoint, again leading to windup. That's not a problem while the actuator remains off, but as soon the actuator is reactivated at the beginning of the next batch, the controller will immediately call for a 100% control effort and saturate the actuator.

The obvious solution to this problem is to turn off the controller's integrator whenever the actuator is turned off or to eliminate the error artificially by adjusting the setpoint to whatever value the process variable takes between batches. But there's another approach that not only prevents reset windup between batches but actually improves the controller's performance during the next batch.

Pre-loading freezes the output of the controller's integrator between batches so that the integral action starts the next batch with the total integrated error that it had accumulated as of the end of the previous batch. This technique assumes that the controller is eventually going to need the same amount of integral action to reach the same steady state as in the previous batch, so there's no point in starting the integrated error at zero. With pre-loading, the integral action essentially picks up right where it left off at the end of the previous batch, thereby shortening the time required to achieve a steady state in the next batch.

Pre-loading works best if each successive batch is more-or-less identical to its predecessor so that the controller is attempting to achieve the same setpoint under the same load conditions every time. But even if the batches aren't identical, it is sometimes possible to use a mathematical model of the process to predict what level of integral action is eventually going to be required to achieve a steady state. The required integrated error can then be deduced and pre-loaded into the controller's integrator at the start of each batch. This approach will also work for a continuous process if it can be modeled prior to the initial start-up.

Bumpless transfer

One potential drawback to pre-loading is the abrupt change it can cause in the actuator's output at the beginning of each batch. Although the actuator won't immediately slam all the way open if pre-loading is used to prevent reset windup, it will try to start the next batch at or near the position it was in at the end of the previous batch. If that causes an abrupt change that is likely to damage the actuator, the controller's integrator can be ramped up gradually to its pre-load value.

A similar problem can occur when a controller is switched from automatic to manual mode and back again. If an operator manually modifies the control effort during that interval, the controller will abruptly switch the control effort to whatever the PID algorithm is calling for at the time automatic mode is resumed.

Bumpless transfer solves this problem by artificially pre-loading the integrator with whatever value is required to restart automatic operations without changing the control effort from wherever the operator left it. The controller will still have to make adjustments if the operator's actions, changes in the process variable, or changes in the setpoint have created an error while the controller was in manual mode, but less of a "bump" will result when the controller is "transferred" back to automatic mode.

Still more windup problems

All of these windup-related problems are compounded by deadtime—the interval required for the process variable to change following a change in the control effort. Deadtime typically occurs when the process variable sensor is located too far downstream from the actuator. No matter how hard the controller works, it cannot begin to reduce an error between the process

variable and the setpoint until the material that the actuator has modified reaches the sensor.

As the controller is waiting for its efforts to take effect, the process variable and the error will remain fixed, causing the integral action to wind up just as if the actuator were saturated or turned off. The most common fix to this problem is to de-tune the controller; that is, reduce the integral gain to reduce the maximum integral action caused by windup.

The same effect can be achieved by equipping the integral action with its own deadtime so that windup does not begin until at least some of the process deadtime has elapsed. This in turn lowers the total integrated error that the controller can accumulate during the interval that the error is fixed.

Deadtime-induced windup can also be ameliorated by making the integral action intermittent. Let the proportional action do all the work until the process variable has settled somewhere close to the setpoint, then turn on the integral action only long enough to eliminate the remaining steady-state error. This approach not only delays the onset of windup, it gives the integral action only small errors to deal with, thereby reducing the maximum windup effect.

But wait, there’s more

This only scratches the surface of the many ways engineers have sought to improve control performance. The PID algorithm has also been modified to deal with velocity-limited actuators, time-varying process models, noise in the process variable measurement, excessive derivative action during setpoint changes, and more. Future installments of this series will look at the effects these problems cause and how they can be avoided.

Vance VanDoren, PhD, PE, is Control Engineering *contributing content specialist. Reach him at controleng(at)msn.com.*

Source: http://www.controleng.com/single-article/fixing-pid/3975cad3f121d8df3fc0fd67660822b1.html