PID math demystified, part 2

You've see the equations, but have you thought about how those elements work together? Part 2: Adding integral and derivative to the mix.

Scott Hayes
05/13/2013

Last week we started with proportional. (Read Part 1) Now let's look at the next part of the equation, the integral component:

$$K_i \int_0^t e(t)\,dt$$

The most striking (and scariest) part of this equation is the big integral sign in the middle. If you've had high school calculus, you think to yourself, "I've got this. Integrals don't scare me. I just need to find the area under the curve from time zero to time t of the error function."

But, this is the real world. What is time zero? How do I integrate an error function? The good news is that the real definition is much simpler than calculus. What the PID function does is take a portion of the error and adds it to a running total. This running total, sometimes called reset, is added to the output. Since reset increases or decreases a little at a time, it adjusts the output of the valve incrementally each scan.

For a PI controller, the two factors that we have covered so far are K_p and K_i, but if you look at the faceplate for most industrial systems, there is only one K (gain) that has no units, and a τ_i (integral time constant) designated as seconds or minutes per repeat. So, a little translation is required. Most industrial controllers don't use the independent form of the equation shown above. Instead, they use the dependent form of the equation:

$$u(t) = K\left(e(t) + \frac{1}{\tau_i}\int_0^t e(t)\,dt + \tau_d\frac{de(t)}{dt}\right)$$

The *K* is typically the same as the proportional gain, K_p. The factor τ_i determines how much of the error is going to be applied to the accumulated reset on each scan. So in the big mathy equation, K_i can essentially be replaced by the faceplate parameters: K/τ_i.

What's important to understand from this is that the gain that affects the proportional action of a controller also affects the integral action. But, the integral time constant τ_i only affects the integral action.

In pseudo code this would look like:

Error := Setpoint - ProcessValue;

Reset := Reset + K/tau_i * Error;

Output := K * Error + Reset;

The unit's minutes per repeat for the integral time constant τ_i comes from the fact that if the error stays constant, that is how long would it take for the integral accumulator to repeat the proportional change in output.

Note: Another way of specifying the integral tuning parameter is in seconds, and then it is the reciprocal of seconds per repeat. If the integral time constant is in seconds, the bigger the number, then the slower the response. If the integral is in seconds per repeat, the opposite is true.

Derivative

Now let's look at derivative:

$$K_d\frac{de(t)}{dt}$$

Again, this is another mathy looking equation with a simple explanation. The mathy definition first, the output will be changed by the derivative (or rate of change) of the error function. What this means is that the output will be affected by the change in error from one scan to another. Adding this to our pseudocode gives us:

Error := Setpoint - ProcessValue;

Reset := Reset + K/tau_i * Error;

Output := K * Error + Reset + ((PreError – Error) * K/tau_d));

PreError := Error; //Save the error for the next scan

What is intended is for the output to change as soon as the process variable begins to move either toward or away from the setpoint. What results can be a very quick response to a change in error from one scan to another.

The intention of derivative action is to respond to changes as they begin to occur. For example, if a temperature is starting to rise, the valve will begin to open as soon as it sees the change instead of waiting for it to cross a setpoint. This can result in a very rapid response to a small change. This rapid response can become unstable if there is noise in the process variable or on a setpoint change. So, the derivative action is often filtered separately and is sometimes calculated on PV only to ignore setpoint changes.

Summary

So now we have reviewed the three components of the PID algorithm. One way they have been described is in terms of the flow of time. P depends on the present error, I on the accumulation of past errors, and D on the prediction of future errors based on current rate of change.

This post was written by Scott Hayes. Scott is a senior engineer at MAVERICK Technologies, a leading system integrator providing industrial automation, operational support, and control systems engineering services in the manufacturing and process industries. MAVERICK delivers expertise and consulting in a wide variety of areas including industrial automation controls, distributed control systems, manufacturing execution

systems, operational strategy, and business process optimization. The company provides a full range of automation and controls services – ranging from PID controller tuning andHMI programming to serving as a main automation contractor. Additionally MAVERICK offers industrial and technical staffing services, placing on-site automation, instrumentation and controls engineers.

Source:

http://www.controleng.com/single-article/pid-math-demystified-part-2/a88740698bafaa3ed8d1d08ffc8a1ee0.html