PID Control

Dan Barry, Jenny Barry, and Andy Barry, December 25, 2013

We have a system with a "set point" (SP), a measurement ("process variable" or PV or $y(t)$) to compare to the set point, and an error value, $e(t) = SP(t) - y(t)$ that is fed to the pid controller. The pid controller outputs a value, $u(t)$, that is used as input to a "process." The process may include actuators and sensors but some sensor provides an output which is used as the process variable:

$$u(t) = K_p e(t) + K_i \int e(\tau) d\tau + K_d \frac{de(t)}{dt}.$$

If we use a sampling interval of T_0, then the discrete forms are:

$$\frac{de}{dt} = \frac{1}{T_0} \left[e(k) - e(k-1) \right]$$

$$\int_0^t e(\tau) d\tau = T_0 \sum_{i=0}^{k} e(i)$$

so

$$u(k) = K_p e(k) + K_i T_0 \sum_{i=0}^{k} e(i) + \frac{K_d}{T_0} \left(e(k) - e(k-1) \right) \tag{1}$$

is the pid position equation. In this case, $u(k)$ corresponds to the new feedback control value produced from $e(k)$ and $e(k-1)$. The value $u(k)$ is also the input to the "process" which in turn will produce the next process variable leading to the next error value, $e(k+1)$.

One issue we need to address is "derivative kick." Derivative kick refers to what happens when the setpoint is changed. On the next cycle, and only for that single cycle, the derivative term will be large due to the huge value of $(e(k) - e(k-1))$. Remember that the PID equation is modeling a continuous system. When the setpoint changes, we also see a discontinuous change in $e(t)$. The derivative of this change is infinite. In the discrete system, we will not get an infinite value, but we will get a large one. To supress derivative kick, we note that

$$\frac{de}{dt} = \frac{dSP}{dt} - \frac{dPV}{dt} = \frac{dSP}{dt} - \frac{dy}{dt}.$$

Now for a constant set point we have

$$e(k) - e(k-1) = (SP(k) - y(k)) - (SP(k-1) - y(k-1)) = -(y(k) - y(k-1))$$

so we can substitute $-(y(k) - y(k-1))$ for $(e(k) - e(k-1))$:

$$u(k) = K_p e(k) + K_i T_0 \sum_{i=0}^{k} e(i) - \frac{K_d}{T_0} \left(y(k) - y(k-1) \right).$$

This causes the derivative term to ignore changes in the set point and that is exactly what we want. We get rid of derivative kick yet preserve a derivative term with a parameter large enough to have a useful effect.

One problem here is that we need values for $y(0)$, $SP(0)$, and $PV(0)$ that will generate $u(0)$ or we have to pick a value for $u(0)$. The concern with picking a value for $u(0)$ is that it may not be consistent with the rest of the time series output from the PID equation. We will discuss this in more detail later.

Another issue is "integral windup." Integral windup refers to what happens when the system moves from its current setpoint (or perhaps 0 when starting) to a far-away setpoint. In the course of getting to the new setpoint, the integral error builds up and up, so that once the system arrives at the new setpoint,

there is a large integral term that causes overshoot and takes a long time to dissipate. A typical method to prevent integral windup is to limit the integral term to values that the system can actually attain. By providing a ceiling and floor, we prevent the term from becoming too large. However, it may be difficult to pick such a floor and ceiling.

For both of these reasons, we introduce a different form of the PID equation, the *velocity form*. This is a recursive form that uses the output PID value from the previous time step. While mathematically these two forms will be identical, using the previous output value allows us to seamlessly provide a good initial starting guess and provide a natural cap on integral windup.

The control value on the previous time step, $u(k-1)$, is

$$u(k-1) = K_p e(k-1) + K_i T_0 \sum_{i=0}^{k-1} e(i) + \frac{K_d}{T_0}\left(e(k-1) - e(k-2)\right).$$

In order to create a recursive form, we first find the difference between $u(k)$ and $u(k-1)$:

$$\begin{aligned} u(k) - u(k-1) &= K_p\left(e(k) - e(k-1)\right) + K_i T_0 e(k) + \frac{K_d}{T_0}\left(e(k) - 2e(k-1) + e(k-2)\right) \\ &= e(k)\left(K_p + K_i T_0 + \frac{K_d}{T_0}\right) - e(k-1)\left(K_p + \frac{2K_d}{T_0}\right) + e(k-2)\frac{K_d}{T_0} \\ &= V_0 e(k) - V_1 e(k-1) + V_2 e(k-2), \end{aligned} \tag{2}$$

where

$$\begin{aligned} V_0 &= K_p + K_i T_0 + \frac{K_d}{T_0} \\ V_1 &= K_p + \frac{2K_d}{T_0} \\ V_2 &= \frac{K_d}{T_0}. \end{aligned}$$

As you can see, if T_0 is reasonably constant, then we can simplify to having just three constant parameters and the three error terms.

Going back to Equation 2, the velocity form of the PID equation often used in the real world is:

$$u(k) = u(k-1) + K_p\left(e(k) - e(k-1)\right) + K_i T_0 e(k) + \frac{K_d}{T_0}\left(e(k) - 2e(k-1) + e(k-2)\right).$$

This is a nice form because it allows us to pick a value for $u(0)$, which will then propagate. For instance, if we are modeling a system of the form

$$u(t) = M + \eta(t)$$

where M is a constant and $\eta(t)$ is some noise, we can set $u(0) = M$ and then this form of the PID equation just has to model $\eta(t)$.

Additionally, this form allows us to deal with integral windup by capping our PID output value rather than the integral term only. Often we know the cap for an output value (i.e. the maximum voltage is 12V) while we do not know a good cap value for the integral. Now this cap also propagates to the next time step. It may also be possible to compute a good maximum value for the change in output.

Another advantage is that we can adaptively change the gain without any output change from the PID equation once we reach a steady-state error. That's because in the velocity form, we use *change* in error for the proportional term so if the error is constant, the proportional term is zero. In contrast, the output from the position form (equation 1) changes with a gain change even with a constant value for error.

However, now we have the "derivative kick" issue with both the proportional and the derivative terms. Similar to the position form case, we have a single cycle where the proportional and derivative terms will dominate the result and then a subsequent cycle where the derivative term alone will dominate. The common solution is to use $(e(k) - e(k-1))$ for the proportional term and $y(k) - 2y(k-1) + y(k-2)$ for the derivative term, again called "derivative on measurement."

$$u(k) = u(k-1) + K_p\left(e(k) - e(k-1)\right) + K_i T_0 e(k) - \frac{K_d}{T_0}\left(y(k) - 2y(k-1) + y(k-2)\right)$$

However, that still produces a derivative kick in the proportional term when the setpoint changes. There is another form where we use $y(k-1) - y(k)$ for the proportional term. The indices are reversed because recall that, for a constant setpoint, DError = -dPV. With this form, the only term that responds directly to the error is the integral term.

$$u(k) = u(k-1) + K_p\left(y(k-1) - y(k)\right) + K_i T_0 e(k) - \frac{K_d}{T_0}\left(y(k) - 2y(k-1) + y(k-2)\right)$$

A more traditional form has rearranged parameters:

$$u(k) = K_c\left[y(k-1) - y(k) + \frac{T_0 e(k)}{\tau_I} - \frac{\tau_d}{T_0}\left(y(k) - 2y(k-1) + y(k-2)\right)\right]$$

Implemetation: For fast-acting processes where the process and the sensor have about the same dynamic response and where you want offset-free operation, use a PI controller so that we can get to zero error. If the process itself is an integrating process, implying that it does not need offset-free control, then use a P-only controller. In that case, as integral action is added, control performance degrades.

It is fairly rare to use the derivative component (full PID) because the derivative sensitivity to noise is so high. That means that you have to filter, which adds lag, or that the derivative parameter is so small that it has little impact on the overall control value. But if the process (rather than the actuator or the sensor) is the slowest element and there is not much noise, then the PID controller is good. It allows using generally larger P and I gains that would otherwise cause oscillations.

If you are going to change the setpoint more than occasionally, then avoid derivative kick by using "derivative on measurement" instead of directly using the change in error. In other words, use the process variable instead of error for the terms that have error-change components.

Use hysteresis to avoid high frequency actuator movements. Basically, don't bother with small changes. This is particularly useful in systems where the actuator has a significant deadband (e.g., loose gears, twist valves, etc.). In those cases, moving the actuator a little does not change the output at all. Another advantage of the velocity form is that it allows easily setting a *minimum* change in control value as well as a maximum.

Filtering is useful to reduce the derivative responses to noise. However, remember that low pass filters will introduce lag which will reduce the response time of the controller, so be careful with the time constants.

Useful references: http://www.controlguru.com/pages/table.html

http://www.cds.caltech.edu/~murray/books/AM08/pdf/am06-pid_16Sep06.pdf

http://higheredbcs.wiley.com/legacy/college/seborg/0471000779/dig_control/ch26.pdf