PID math demystified, part 3: More on derivative control

Consider how a PD controller would work, without an integral function. Would you ever want to use that approach?

Scott Hayes

10/21/2013

To investigate how derivative action works, let's look at a proportional derivative or PD controller. PID controllers are far more common than PD alone, but we already have an understanding of the integral component's effects from the first two parts of this series, so we don't need to review it again.

For a PD controller, we add the derivative of the error into the equation. Similar to what we discussed in the previous posts, we're not really interested in deriving a derivative of the error function. Conceptually, the derivative in this case refers to how fast the error is changing. So, if we take the change in error divided by the change in time we get the slope.

$$u(t) = K_p \times e(t) + K_d \frac{de(t)}{dt}$$

$$u(t) = K \times e(t) + \frac{K}{\tau_i} \Delta e(t)$$

$$u(t) = K \times \left[e(t) + \frac{1}{\tau_d} \frac{de(t)}{dt} \right]$$

To explain how this works, let's look at the pseudo code for this controller. The calculation of error is the same as before: setpoint minus process value. Since the derivative reduces down to the change in error, the output is now the same proportional component as before: gain times error. The current error minus last error is multiplied by the gain and divided by the derivative time constant. The current error is stored in last error for use in the next scan.

```
Error : = Setpoint – ProcessValue;

Output : = K * Error + K/tau_i * (Error – LastError);

LastError : = Error; // save for next scan
```

So what does this mean?

The proportional component is affected by the error at that time, in the present. The integral component is affected by an accumulation of the error, or the past. The derivative component is a measure of how fast the error is changing, or a prediction of the future error.

How is this prediction of the future used? At first glance you might think that this term would be used to get you to your setpoint that much faster. But, that is not really the case. In practice, the derivative component is used to detect when the process variable is changing too fast, and it puts the brakes on to prevent overshooting the setpoint.

So if the derivative component acts as brakes on the momentum, how does it get you to the setpoint faster? It does this by allowing you to use a higher proportional gain to get you there quickly, but dampening the overshoot that would normally make that level of gain unstable.

The tuning constants for derivative control are typically the same units as the time constant for reset. A couple of other considerations though are important. On many controllers the derivative term is filtered independently. This prevents signal noise or spurious disturbances from being interpreted as a change in momentum, which causes the derivative action to overreact. Also, on some controllers the derivative does not actually derive from the error, but instead on the process variable signal alone. This prevents a change is set point from being seen as a change in momentum.

Source:

http://www.controleng.com/single-article/pid-math-demystified-part-3-more-on-derivative-control/4bda40a405936d24c57475bbd593cd20.html