PID Loops in Software applications

M. Simon, retired Aerospace electronics consultant

 I want to take a look at PID loops and how they are typically handled in software. I'm going to contrast how the motor control folks do things vs. how the chemical industry folks do things.

Now there are all kinds of things you can control with such a loop (in a motor control context). You might want to control position, velocity, or torque with such a loop. You also might want to limit acceleration, torque, maximum velocity or some other factor.

Which leads us to the fundamental measurement in such a system. Error = (Set point - actual output). That is the fundamental knowledge required for any method of control. So how do we go about reducing the error? We can start with proportional control which is the P in PID. You multiply the error by some constant Kp and this is used to control the output of the controller. The bigger the error the bigger the output of the controller. However, the output of the controller is dependent on having an error. So the error can never go to zero long term.

Which brings up the next term I. For I you sum the errors and multiply them by a

constant Ki. Then you add that to the proportional term and that sum determines the output of the controller. By using an integration term the error can go to zero and the controller will still have the output required to keep the system at the set point.

Finally you have the D term. It is the rate of change of the error multiplied by a the constant Kd. The purpose of this term is to keep the system from changing too fast. It eases you in to the required output. You add that to all the other terms and you get the controller output.

You can watch how all these terms affect a control system by watching such a system in operation. The most accessible such system is the cruise control on your auto. Set in on a flat stretch of road and watch how it operates in hilly country. That will give you a feel for how properly tuned (all the various Ks are correct) PID loops operate.

That describes how a traditional motor control system does the job. All the Ks are independent and you figure them out more or less by trial and error. National semiconductor has a motor controller that has been around for quite some time, the LM-628. You can find details on it at:

http://www.national.com/mpf/LM/LM628.html [1]

The application notes that go with it are very useful study guides for the above type controller. AN-693 and AN-706. They are very good and go into much more detail on the above type of controller along with details on how National implemented the PID loop.

Which brings us to the way the chemical guys handle the problem. It is a little different and instead of just numbers that are arbitrary the tuning numbers are

actually physical and can be easily derived from a simple test of the controlled system.

We still start with the error. But the equation looks a little different. The first term is the loop gain term. It is called Kp. It can in fact be the same Kp as above but it is multiplied by the sum of all the rest of the terms.

And the rest of the terms include the error term. Which is the same as above. What is next term is the integral time of the loop Ti. You multiply one over that by the integrated error. Finally you have the derivative time of the loop Td. You multiply that by the rate of change of the error.

So the final equation looks like the gain (Kp) multiplied by the error term plus integral term (1/Ti * integrated error) plus the derivative term (Td * rate of change of error). That gives you the output of the controller.

If you want the pretty version (not text) you can have a look at:

http://en.wikipedia.org/wiki/PID_controller#Ideal_versus_standard_PID_form [2]

Now what does all this get you? An easy way to tune the loop. You do that by effectively shutting off the controller and applying a step change to the system. Then you watch the system. The dead time (the time from the step change until the system starts changing) gives you Ti. The slope of the change gives you the gain. Use a Td of 1/10th Ti and you then have at least a starting point for system tuning. Of course you may want to use a constant times Ti and yet another constant times Td to give you the overshoot and other system characteristics you actually want.

So what is the advantage? All the constants used are physical rather than arbitrary.

You can find out more about how the chemical guys do it in David St. Clair's truly wonderful book "Controller Tuning and Control Loop Performance" available here:

http://www.amazon.com/gp/product/0966970306?ie=UTF8&tag=poweandcont-20&linkCode=as2&camp=1789&creative=390957&creativeASIN=0966970306 [3]

The above discourse is only the barest outline of the subject. There are issues like preventing too much accumulated error (controller windup), limiting the rate of change of the control output, and other questions. The above is only meant to be an outline of two different ways of looking at PID loops.

If you need help with your PID loops including designing an auto tuning controller you can contact M. Simon by getting his e-mail address from the side bar of "IEC Fusion Technology". A simple search will get you there.

Source URL (retrieved on *12/24/2014 - 7:10pm*):
http://www.ecnmag.com/articles/2009/10/pid-loops-software-applications

Links:
[1] http://www.national.com/mpf/LM/LM628.html
[2] http://en.wikipedia.org/wiki/PID_controller#Ideal_versus_standard_PID_form
[3] http://www.amazon.com/gp/product/0966970306?ie=UTF8&tag=poweandcont-20&linkCode=as2&camp=1789&creative=390957&creativeASIN=0966970306