PID-f PID control for fast dynamical systems

Ravelle Scientific has developed a new flavour of PID controller. Simply put, the derivative term is replaced by an hf filter. We call this a PID-f controller.

A PID Controller recap… *(skip this if you are familiar with PID, and jump straight to PID-f)*
The acronym PID stands for Proportional Integral Derivative controller. Let's review the variables associated with PID control, before looking at it's standard implementation and our variation on it.

In control theory, the device to be controlled is often called the Plant or Process. The desired new output, level or position of a physical aspect of that process is called a Setpoint, **SP**. In order to get feedback from the system, a Process Variable **PV**, is updated by sampling a sensor. The data from the Process Variable is used by the PID loop to adjust an Actuator. It is the aim of the control loop to adjust the Actuator such that the Process variable rapidly converges to the desired Setpoint.

Example: Cruise control in an auto mobile. Here, the Setpoint is our desired speed, 70mph say, and the Process variable is our current speed. This could be obtained by sampling a hall-effect sensor on one of the wheels, or from a sensor in the speedometer instrument. Ignoring the possibility of gear changes, the Actuator is the throttle (which on a modern car has it's own control loop, but on an old car was just a lever on the side of the carburettor).

Let's say our cruise control is set to 70 mph and the car is cruising along happily at that speed until we reach the bottom of a hill. The SP indicates that our speed has dropped. It is the job of the control loop to adjust the Actuator (the throttle) so that the SP (70mph) as closely as possible. During the execution of some control loops, there is often the posiblity that the PV will overshoot the SP before settling back to the desired SP. In systems where the sensor updating the PV has a lag or delay, there can even be osculations before the system settles. In a car, the manufacturer may deem overshoot undesirable because it could cause the car to temporarily break the speed limit. In many types of industrial plant, it is not desirable to constantly make small changes to the actuator position, as this may ware it out. Thus a “deadband” is introduced, and the actuator position is only updated when the error is deemed large enough. So in practice, few PID loops operate in their pure form, there are always constraints applied so that the process is coerced to operate within designed parameters.

PID Loop, author TravTigerEE

In the classical form of this type of controller, the P, I and V terms are all derived from the Error term, E. The error is simply Setpoint – PV. So going back to our auto-mobile above, if our target (SP) is 70mph and we have slowed to 69mph, the Error is -1 mph.

The P term is simply E scaled by the P-gain, usually written as Kp
The I term is the sum off all the Errors so far. Again this is scaled by a gain, Ki
The D term is the derivative of the E term, usually calculated by storing the previous level of the E term and subtracting it from the current E term. This term also needs to be scaled by a gain factor Kd.

In PID diagrams in the literature, it looks at a causal glance, that after being scaled by their respective gains, the terms are simply summed together. This is misleading because of the subtlety that the derivative is derived form the error, and the error is setup to oppose (or correct) the plant. Thus, when it is casually mentioned in discussions on PID loops, that the D term is added, it does not give the correct intuitive explanation. What is really happening is that by adding the derivitive of the error to the output, the derivative of the PV is actually being subtracted, so that any movement of the PV tends to be canceled by the D term.

So the formula for a clasical PID is:

$$Actuator = P \cdot K_p + I \cdot K_i + D \cdot K_d$$

In other words, and glossing over the scaling, the output is made up of the error, added to the long term accumulation of the error, plus the change in the error. In a normal setup, all these terms work in opposition to the movement of the PV, so that the PV is brought to and kept at the desired SP.

Contribution of each term

Let’s look at the terms one at a time to see what they do.

Some simple plants can get by with the **P** term alone. An error will result in some P, and this will drive the actuator in the right direction to reduce the Error. This in turn will reduce the P. The main problem with using P alone, is that most systems will never converge to zero Error. If some steady state value is needed on the Actuator to achieve a desired SP, but the Error were to drop to zero, the P term would be zero, and thus the actuator would not be driven. Because of this, zero error can never be reached for most values of SP. However, where P gain (Pk) is high, the error can be quite small. A good example of this is a simple op-amp circuit with negative feedback. The open-loop gains of op-amps are generally in the 10s to 100s of thousands, thus the tracking of the input is very good.

The **I** term is the accumulation of all the Errors so far. In a properly behaved system, this number will not grow indefinitely, because the Error will fluctuate between positive and negative. Eventually, the I term will converge to a steady state, which represents the long term average setting the actuator needs to achieve the SP with no steady state error (SSE). Intergral control is not often used on it’s own because at low gains it would be very slow to converge, and at high gains it will overshoot and oscillate wildly above and below the SP.

Putting it all together
By using **P** and **I** together, a so called **PI** controller can be implemented. The I term causes the system to slowly converge to zero steady state error, while the P term makes it more responsive to changes in SP, and to some extent can also prevent osculations that P control alone would suffer from. However, PI control on it’s own will tend to suffer from overshoot and even oscillation, unless low gain settings are used. This is where the D term can help.

The **D** term tells us how fast the system is moving, and also changes sign when the direction of this movement changes. By adding this term to the PI terms, very good damping can be achieved for a wide variety of processes and plants. Thus the gains of P and I can be set higher, resulting in faster action and convergence. Adding D, subtracts energy from the system at more or less the frequencies likely to cause trouble. However, like all magic-bullets, the D term does not come without side-effects. Because the Error will typically change quite slowly, a large constant for the D gain will often be needed. Noise in the system results in small changes to to the Error, and this propagates straight to the D term, only to be multiplied by the high gain of Kd. Thus attempting to use a big number for Kd in order to achieve ideal damping, can cause jumpiness in the system, and in the worst case, can cause oscillation itself. So there is a trade-off that must be made between responsiveness and noise.

Modern implementations sometimes re-arrange the maths somewhat, and for instance apply the Error only to the I term, and use the PV readings to directly create the P and D terms.

PID-*f*
A new way of extracting more performance from PID type loops.

Looking at the **D** term above, we can see that it represents the high frequencies in the system (but with inveted phase). Whereas the **I** term represents DC. By adding **D** to the system, the unwanted oscillations are damped out before they can even take hold. It turns out, that it is the adding of an anti-phase version of the system's **hf** output, that cancels overshoot and oscillation. Basically, what derivative action does, is to subtract energy from the system, at the trouble-making frequencies. The injection of energy by the control loop causes energy to accumulate in the system in the form of overshoot and oscillation. Derivative action removes this energy. Once this is understood, it soon becomes obvious that taking the differential is a very crude way to get at the high frequencies. Loop frequency becomes critical – run the loop too often and the differences between the current Error and the last may be less than 1, which on a integer system is useless. Conversely, running the loop at low repetition rates will fail to capture all of the higher frequencies present, and will cause aliasing.

Instead of using the differential term directly, PID-f replaces the D term with a high frequency, digital filter. The output of this filter is multiplied by the gain, Kf, and and added *in anti-phase* to the (P·Kp + I·Ki) terms. The anti-phase is achieved by making the kf term negative.

$$Actuator = P \cdot K_p + I \cdot K_i + hf \cdot K_f$$

(The K_f gain is always negative)

A further and important enhancement is that the hf signal need not be the entire band of high frequencies. If upon investigation, it is found that a system has a tendency to oscillate at say, two frequencies, these two frequencies can be isolated, and their gain can be adjusted to the right relative proportions. When these frequencies are taken in anti-phase and added to the (P·Kp + I·Ki) terms, the best possible cancellation will result.

Case Study of a PID-f controller
We worked on a swinging arm micro-balance. The system had an electromagnetic actuator that endeavoured to keep the swing-arm exactly horizontal. When a test weight was put on the arm, the arm weighed downwards, and the PID loop adjusted the arm back to horizontal by increasing the current in the actuator coil. By measuring how much current it took to keep the arm level, the test weight's mass could be determined.

The system worked but exhibited two osculations. A slow oscillation at about 0.5 Hz and a faster oscillation at about 15Hz. The 0.5 Hz oscillation was damped out over time by the PID loop. However, no adjustment of the PID parameters could produce a satisfactory damping of the 15Hz oscillations. By switching to the PID-f method, and mixing back an anti-phase version of these two frequencies in the right proportion, it proved possible to squeeze out both oscillations to a high degree. By selectively amplifying only these two frequencies, broad-band noise was not amplified, further aiding the system’s reliability.

I have put this technique into the public domain, but would appreciate a mention or a link if you or your company successfully uses it.

First printed may 2014
Author Mark Ravelle

Please note that although the results are similar, PID-f is not quite the same as the PIDF controller described by Galal A. Hassaan et al: http://www.ijetae.com/files/Volume3Issue3/IJETAE_0313_162.pdf

Top