Comparison of PID Control Algorithms

(All Controllers Are Not Created Equal)

One fine day, a plant engineer, replaced his controllers. Even though he used the same settings on the new controllers, the retrofitted loops went out of control in automatic. He tried to tune these controllers the same way he had tuned the old ones. The loops seemed to get more unstable.

This mysterious and very real situation is the result of two manufacturer's using different PID algorithms. Read on to solve this and other common mysteries about PID controllers.

In practice, manufacturers of controllers don't adhere to any industry wide standards for PID algorithms. Different manufacturers and vendors use different PID algorithms and sometimes have several algorithms available within their own product lines.

The figures and graphs used in this article were produced using the ExperTune Loop Simulator. For PID loop tuning, analysis and simulation contact ExperTune.

The Name Game

Just as there are no adhered to industry standards for PID controllers, nomenclature and action for similar modes varies.

P Proportional Band = 100/gain
I Integral = 1/reset
D Derivative = rate = pre-act

Some manufacturers call Proportional Band the Proportional Gain. Manufacturers interchange names and units for integral or reset action. In this article, integral action is defined in time/repeat and reset in repeat/time. One is the reciprocal of the other. The action of either reset or integral can be reversed depending on the manufacturers units.

The Algorithms

There are three major classifications of PID algorithms that most manufacturer's algorithms fit under. These three are: series, ideal, and parallel. Again, manufacturers vary on the their names for these categories. The only way to really tell which one you have is to look at the equation for the controller. In simple form these are:

Ideal algorithm:

$$OUTPUT = Kc \left[e(t) + \frac{1}{I} \int e(t)d(t) + D \frac{d\,e(t)}{dt} \right]$$

Parallel:

$$OUTPUT = Kp\,[e(t)] + \frac{1}{I} \int e(t)d(t) + D \frac{d\,e(t)}{dt}$$

Series (Interacting):

$$OUTPUT = Kc \left[e(t) + \frac{1}{I} \int e(t)d(t) \right] \left[1 + D \frac{d}{dt} \right]$$

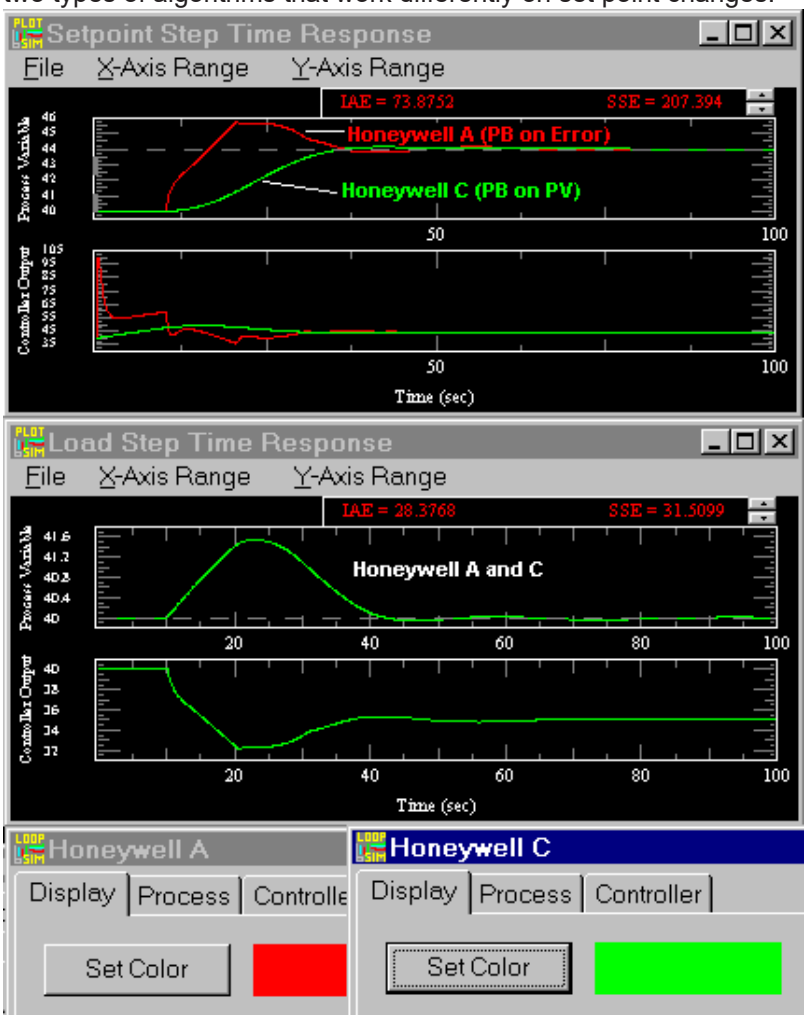
Kc, Kp are gain; I, Ip are integral and D, Dp are derivative settings. The series controller's strange looking form makes it act like an electronic controller. A three term controller can be made with only one pneumatic (or electronic) amplifier using the series form. Thus, pneumatic controllers and early electronic controllers often used the series form to save on amplifiers which were expensive at the time. Some manufacturers use the series form in their digital algorithms to keep tuning similar to electronic and pneumatic controllers.

Differences in Proportional Band Or Gain

If you use only proportional action, the main difference between series and ideal algorithms is that some manufacturers use proportional band while others use gain. On a controller using the gain setting, increasing this setting makes the loop more sensitive and less stable. While decreasing proportional band on controllers using it will have the same effect.

Different control algorithms, even from the same vendor

Some manufacturers allow more flexibility with P action by letting you choose whether gain (proportional) action works on set point changes. For example, Honeywell TDC has two types of algorithms that work differently on set point changes. With their type A algorithm, gain action acts on set point changes and with their type C it does not. For load upsets, type A and C act the same, but for set point changes, the difference is dramatic.



ExperTune Loop Simulator Windows compare responses of Honeywell TDC type A (proportional action on error) and C (proportional action on PV only) controllers on a simulated temperature loop. Bottom graph shows either type controller response to a step load change. Top plot red is response of Type A, top plot green of Type C. With the type C algorithm, damping and overshoot to step set point changes are similar to damping and overshoot on step load disturbances. Type C may be desirable over type A, since tuning for load or set point changes is similar with type C.

Because of the sensitive set point response, you may want to use type A for the inner loop or slave in a cascade. Type C with smooth set point response may be better for the outer or master loop.

Bailey's "error input" and "PV and SP" algorithms are analogous to Honeywell's type A and C. Bailey's "error input" has sensitive set point response while Bailey's "PV and SP" has smooth set point responses.

Like Honeywell's, the two Bailey algorithms give identical load responses. Because the load responses are the same for the different Bailey and Honeywell algorithms, they have the same stability. For a fast analysis of the stabilities, ExperTune allows a comparison of the robustness plots of the algorithms.

Differences In Integral Action

Once you convert integral and reset values to the same units, PI controllers respond mostly the same for load disturbances. The proportional action may be different as described above. Anti-reset windup is usually done differently, but the effect of these differences is usually minor compared to other differences between algorithms.

Differences In Derivative Action

The largest variation among controllers from different manufacturers is the way they handle derivative action. Virtually no two are the same. This is part of the reason why many people don't use derivative action. The differences are caused from different methods of filtering or not filtering at all, whether the derivative works on set point changes or not, and how derivative interacts or does not interact with the integral action.

On controllers, when you set derivative to something besides zero, you get derivative action. In a series controller, when you use both integral and derivative actions, the integral and derivative modes interact. Interaction causes the effective

controller action to be different from what it would be in a ideal controller.

The effective proportional band is:

PB (effective) = PB/(1 + D/I)

The effective integral time is:

I (effective) = I + D

The effective derivative time is:

D (effective) = 1 / (1/I + 1/D)

Where PB, I and D are the proportional band, integral and derivative values you set or entered into the series controller. The effective values are equivalent ideal controller settings.

These equations show that for the series controller you cannot make the effective derivative time greater than 1/4 the effective integral time. The largest effective derivative occurs when D=I. When D is set larger than I, the effective integral time is adjusted more with D and the effective derivative is adjusted more with I! Therefore it is usually good control practice to keep the values of D less than I for a series controller.

For example: Foxboro and Fisher use a series algorithm. AEG Modicon, Texas Instruments controllers use the ideal type. Honeywell has both series and ideal algorithms. Bailey, Allen Bradley, and GE have both ideal and parallel algorithms.

Other Differences in Derivative

Besides the interaction differences described above, derivative action among the series and ideal groups varies.

With most controllers, derivative works only on measurement. On some controllers however, derivative action works on set point changes. Although response to a load disturbance will be the same, set point response on these controllers can get out of hand.

Since most controllers are used for regulating disturbances, derivative action working on set point changes is usually not a problem except in cascade loops or ones where the set point is being manipulated.

Of more significance is whether and how filtering is done when you dial in derivative.

The Unlimited Derivative Problem

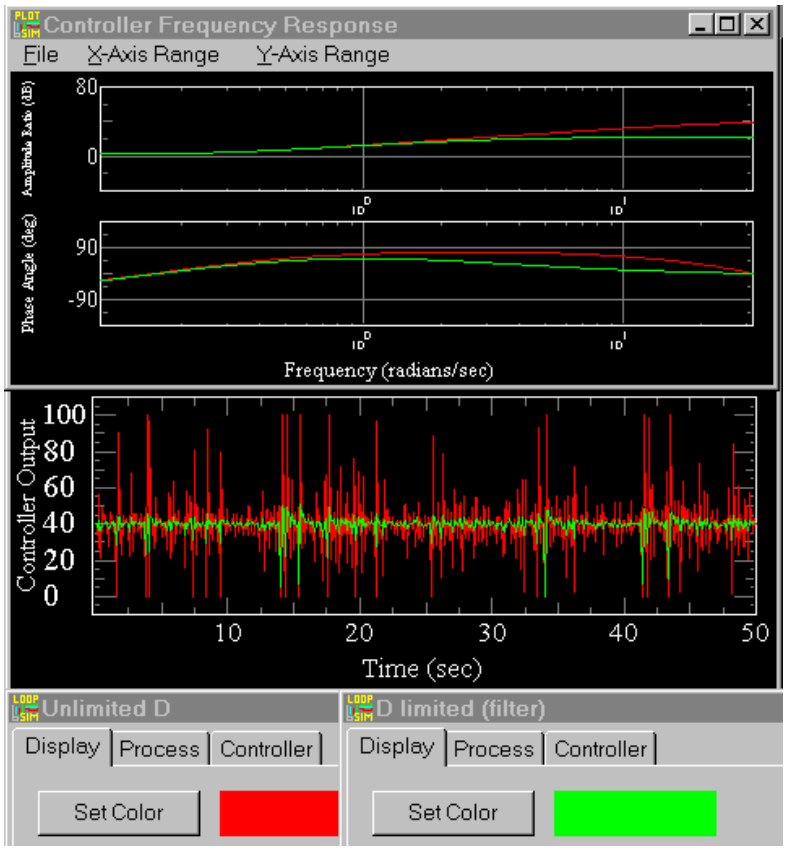
Some manufacturers do not filter or limit derivative action. Thus, at high frequencies, the amplitude ratio gets large. In the Figure the red line shows the amplitude ratio of unlimited derivative action.

Frequency Response
The Figure shows the same PV noise added to a PID controller with (green) and without (red) PV filtering. PV filtering limits the derivative gain.

Unlimited derivative action does not help good loop control but does amplify measurement noise in the controller output. The result of unlimited derivative is a “jumpy” or nervous and noisy controller output. The lower graph in the Figure - red line is the time response of a controller to measurement noise. This can wear out valves, or drive a slave loops set point crazy. Worse yet, the noise can drive the controller into saturation which causes the anti-reset code to take over. No wonder derivative is seldom used!

Filtering Limits Derivative Noise

On the controllers that use filtering with derivative, usually the measurement signal gets the filtering. The time constant of filtering is usually calculated by these algorithms based on the derivative value dialed in. The amount of filtering changes with the amount of derivative. This has the effect of limiting derivative action at high frequencies. In Figure the green line shows the amplitude ratio and controller output using limited derivative action. Control loop performance is the same on both since unlimited derivative does not improve control loop performance.



Parallel Controllers

With parallel controllers, controller gain is not multiplied by the error signal only. Integral and derivative actions are “independent” of the controller gain.

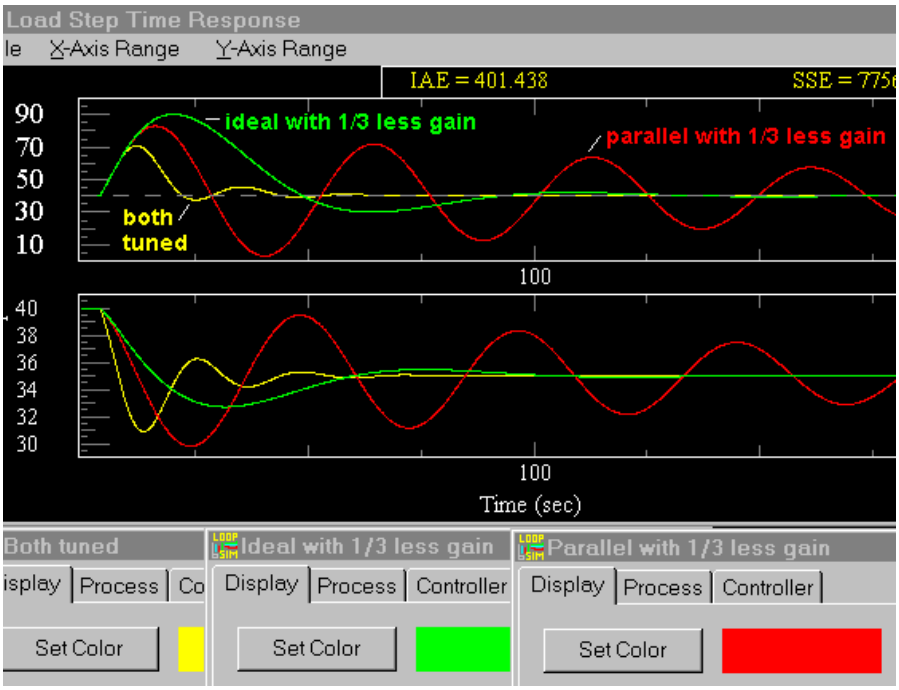
At first it might seem that the parallel controller is easier to work with because of this “independence”. But, parallel algorithms require very different integral and derivative tuning parameters than other controllers. These equations show how to convert from parallel to ideal settings:

Kc = Kp

I = Ip Kp (units of time/repeat)

D = Dp/Kp (units of time)

There is more of a difference between parallel versus ideal controller tuning than series versus ideal tuning. The intuitive feel for tuning a parallel controller is very different from the others. The Figure below shows load response in a level loop. The yellow curve is for a tuned (ideal or parallel) controller. Normally it would seem that lowering the controller gain will make the loop more stable as in fact it does with the ideal controller in the Figure. However, the parallel controller gets less stable with lower gain!. Like all controllers, it also gets less stable with more gain. So either increasing or decreasing the gain on a parallel controller can drive the loop unstable! The controllers in the Figure are PI controllers. The situation is more pronounced when you use derivative.



The impact of tuning depends on the algorithm

With the parallel controller, the effective integral and derivative values change with the gain setting. So, lowering the controller gain also lowers the effective I; increasing controller phase. Lowering gain also increases the effective D, moving the derivative phase to higher frequencies; eliminating its stabilizing effect on controller integral action. The overall effect is unstabilizing as the Figure shows. For example: Bailey and Allen Bradley both have a parallel algorithm available that they describe as a “non-interacting” algorithm. They call the ideal algorithm an “interacting” one.

Conclusions

Choosing the best algorithm for your process is dependent on your process control needs and objectives. Different algorithms perform better in different situations. By using ExperTune simulation software, these differences are easier to understand.