M. KING, Whitehouse Consulting, Isle of Wright, UK

Update your process control algorithms

Within the distributed control system (DCS), commonly used in the hydrocarbon processing industry (HPI), lies a rarely-used version of the proportional-integral-derivative (PID) control algorithm. This algorithm, when properly tuned, can reduce by a factor of four the quantity of off-spec material made during a process upset. As such, it can be the single largest contributor captured by improved process control.

Basics. Most control engineers are familiar with the conventional form of the PID algorithm, which determines the controller output, M, based on the controller error, E:

$$M = K_c\left(E + \frac{1}{T_i}\int E \times dt + T_d\frac{dE}{dt}\right) \quad (1)$$

The algorithm has tuning constants controller gain, K_c, integral time, T_i, and derivative time, T_d. It can also be written in the Laplace form:

$$M = K_c\left(1 + \frac{1}{T_i s} + T\,s\right)E \quad (2)$$

These forms are usually used by control-system vendors to document the algorithms included within their products. However, both forms should strictly only be applied to analog systems. To describe the digital equivalent, let's first differentiate in Eq. 1:

$$\frac{dM}{dt} = K_c\left[\frac{dE}{dt} + \frac{E}{T_i} + T_d\frac{d}{dt}\left(\frac{dE}{dt}\right)\right] \quad (3)$$

This is often described as the *velocity form* of the algorithm, as opposed to the *full-position form* represented by Eq. 1. The *discrete form* is converted by replacing dM with ΔM (the change in controller output) and dt with ts (the controller scan interval). The change in error, dE, is given by E_n (the current error) less E_{n-1} (the error at the previous scan). It is simplified as:

$$\Delta M = K_c\left[(E_n - E_{n-1}) + \frac{ts}{T_i}E_n + \frac{T_d}{ts}(E_n - 2E_{n-1} + E_{n-2})\right] \quad (4)$$

This *incremental form* also provides the benefit of *bumpless initialization.* It generates the change in output, rather than the absolute value. So if, when the controller is switched from manual to automatic, the setpoint (SP) has been tracking the process variable (PV) and there will be no change made to the process.

However, this form is rarely included in the control system because it is prone to a *derivative spike.* Imagine that the controller is in automatic mode and that the process has been steady at SP for some time. Both the current error and its recent values will be zero. Consider how the derivative action then responds to the process operator, changing SP by ΔSP. Assuming the error is defined as ($PV - SP$), the change in output generated by the derivative action is:

$$\Delta M = -\frac{K_c T_d}{ts}\Delta SP \quad (5)$$

Since the deadtime of the process is likely to be greater than ts, it will not have responded to this change before the next controller scan. So, the error will remain fixed. The derivative action will cause a change given by:

$$\Delta M = \frac{K_c T_d}{ts}\Delta SP \quad (6)$$

At the next scan, the two previous errors will be the same as the present error, and the derivative action will, therefore, be zero. This will remain the case until the process deadtime expires. The net effect is that the derivative action has generated a spike in the controller output of not an insignificant size. **Remember:** In this case, ts is typically around one second; even very modest values of 1 for K_c and 0.5 minute for T_d will cause a spike that is 30 times larger than ΔSP. Quite easily, this action could exceed the output range. When the process deadtime does expire, this will cause a large deviation from the SP, to which the proportional action will respond, reproducing the spike as an unnecessary corrective action.

Corrective action. To avoid this problem, control-system vendors usually modify the derivative action so that it is based on PV rather E:

$$\Delta M = K_c\left[(E_n - E_{n-1}) + \frac{ts}{T_i}E_n + \frac{T_d}{ts}(PV_n - 2PV_{n-1} + PV_{n-2})\right] \quad (7)$$

Changes in SP will no longer result in derivative action. If the SP is constant, then changes in PV will be the same as changes in E (since $E = PV - SP$). Now, the response of derivative action to process disturbances will remain unaffected.

Surprisingly, some control-system vendors retain the derivative-on-E version of the algorithm, sometimes as an option. This might explain why derivative action generally has a poor reputation. For example, if the controller is the secondary of a cascade, and, thus, experiences frequent SP changes, the derivative spikes will appear as noise in the controller output. However, the most misunderstood algorithm arises from the option that most the DCS offer, i.e., also basing the proportional action on the PV:

$$\Delta M = K_c\left[(PV_n - PV_{n-1}) + \frac{ts}{T_i}E_n + \frac{T_d}{ts}(PV_n - 2PV_{n-1} + PV_{n-2})\right] \quad (8)$$

At first glance, this would appear to significantly undermine the benefit of the proportional action. With the proportional-on-*E* version the, change in *SP* will cause the proportional action to change the controller output according to:

$$\Delta M = -K_c \Delta SP \quad (9)$$

This one-off *proportional kick* does much to ensure that the *PV* approaches the *SP* as quickly as possible. With the proportional-on-*PV* version, only the integral action responds to the change in *SP*—providing more of a ramp function than a step. This is illustrated in **FIG. 1,** where the performances of the two algorithms are compared. This has given rise to myths such as the proportional-on-*PV* algorithm should be used if a slow response is required or if the process is deadtime dominated. In fact both of these situations can be addressed effectively by the correct tuning of the conventional proportional-on-*E* algorithm. What engineers overlook is that the proportional-on-*PV* algorithm can be tuned to give a performance very similar to that of a well-tuned proportional-on-*E* algorithm, as illustrated in **FIG. 2.**

This begs the question: Why are both made available? To answer this question, consider how each algorithm responds to process disturbances (*load changes*). Subjecting both algorithms to a load change, with the tuning kept the same as that used in **FIG. 2,** shows that the proportional-on-*PV* version substantially outperforms the proportional-on-*E* version. From **FIG. 3,** the duration of the disturbance is typically halved, as well as the maximum deviation from *SP*. If the controller were, even indirectly, controlling a key variable such as product quality, then the quantity of off-grade product would be reduced by a factor of at least four.

Tuning vs. algorithm. What should be emphasized is that it is not the choice of algorithm that has brought about this improvement, but the choice of tuning. With the same tuning, modifying the controller to use *PV* instead of *E* has no effect on the way it responds to process disturbances. However, if the proportional-on-*E* version were installed as the tuning designed for the proportional-on-*PV* algorithm, its response to *SP* changes would be unacceptably aggressive—particularly in terms of manipulated variable (*MV*) movement—as shown in **FIG. 4.** The choice of the proportional-on-*PV* algorithm merely allows installing the tuning preferred by the engineers without causing problems when the SP is changed.

Why industry has not generally adopted this algorithm can be explained by engineers largely assessing controller performance by its response to *SP* changes in the belief that load changes will be handled similarly. Engineers are simply

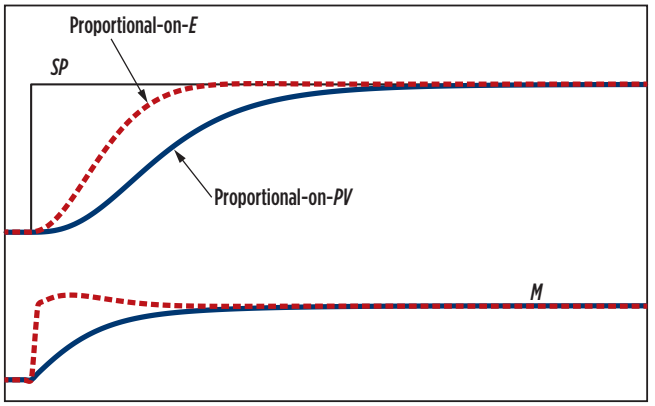

FIG. 1. Relative performance of the proportional-on-*PV* and proportional-on-*E* algorithms in response to a *SP* change (using the same tuning for both).

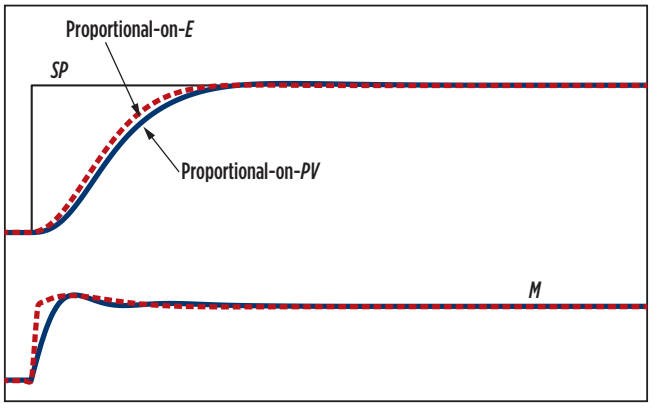

FIG. 2. Performance of the proportional-on-*PV* algorithm (optimally tuned for *SP* changes).

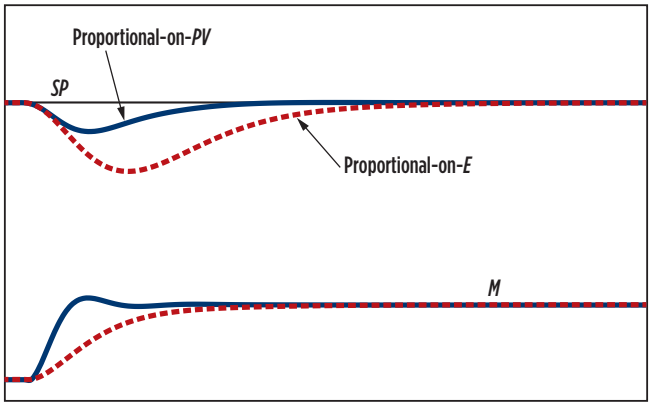

FIG. 3. Response to a load change if both algorithms are tuned for *SP* changes.

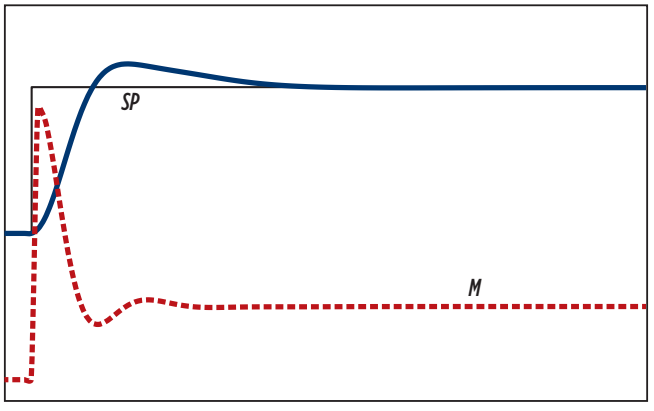

FIG. 4. Performance of proportional-on-*E* algorithm in response to an *SP* change (using the *SP* tuning designed for the proportional-on-*PV* algorithm).

unaware of what improvements can be achieved by switching algorithms. This is partly the fault of the system vendors; many of their own staff fail to appreciate why the algorithm is included in the system and therefore cannot properly explain its purpose to users. But, since it is the tuning that gives the improved response, perhaps the major limitation is the lack of an effective tuning method.

Well over 200 methods have been published.[1] Most require that the process dynamics are known—usually as a first-order model with process gain, K_p, deadtime, θ, and lag, τ. Almost all are designed for the conventional algorithm as described by Eq. 1. But, even for this purpose, they often prove inadequate. Designed to make the controller approach the *SP* as fast as possible, most neglect the effect this has on the *MV*. The recommended tuning will often cause unacceptably fast corrections—particularly when the θ/τ ratio is small. Indeed a means of evaluating a tuning method is to determine what value it recommends for K_c as θ/τ approaches zero. If K_c approaches infinity, then the method is theoretically correct but it has little practical use.

There are a number of methods that allow the user to specify the aggressiveness of the controller. Commonly used is the IMC method.[2] This requires the engineer to specify the term λ—the time constant of the approach of the controlled variable to a new *SP*. However, λ does not explicitly define the impact on the *MV*. It is necessary to adjust λ by trial and error to achieve the best compromise between a fast return to *SP* and acceptable changes to the *MV*.

It is unrealistic to expect simple formula-based methods to generate effective tuning constants for all the situations where they are likely to be applied. First, there are two fundamentally different process types—*self-regulating* and *integrating*. Second, most control systems offer multiple versions of the PID algorithm. Further, different vendors have modified the algorithm in different ways—greatly increasing the number of variations. Third, the consideration given to the changes made to the *MV* will be process-specific. Some will tolerate very fast and large changes, while others will not. And, finally, the engineer must take account of the controller scan interval. Usually, because the interval is often small compared to the process dynamics, users can apply methods designed for analog control. But for very fast processes (such as compressor surge protection) or for controllers with very long scan intervals (such as those based on discontinuous onstream analyzers), a method that incorporates *ts* is needed.

To consider all these factors, an impractically large catalog of tuning formulas is required. However, applying trial-and-error methods may be the optimum approach. While time-consuming and not always properly applied on the real process, trial-and-error methods can be replicated relatively simply in a computer simulation.[3] Such tools are available on the Internet and can account of all the issues raised in a concise review of published tuning methods.[4] Using the software, the points plotted in **FIGS. 5–7** were derived for the so-called *ideal* proportional-on-*PV*, derivative-on-*PV* algorithm described by Eq. 8. They apply to a self-regulating process with dynamics much larger than the controller scan interval. The tuning criterion was to minimize integral over time of absolute error (ITAE) subject to a limit placed on the movement of the *MV* so that it did not overshoot the necessary steady-state change by more than 15%. No limit was placed (or was necessary) on the *PV* overshoot. Curve fitting gave these tuning formulas:

$$K_c = \frac{1}{K_p}\left[\left(1.038\frac{\theta}{\tau}+0.353\right)^{-1.644}+0.583\right] \quad (10)$$

$$T_i = \tau\left[\left(0.588\frac{\theta}{\tau}+4.164\right)^{0.929}-2.971\right] \quad (11)$$

$$T_d = \tau\left[\left(1.190\frac{\theta}{\tau}+3.850\right)^{0.487}-1.857\right] \quad (12)$$

The controller used as illustration in this article was tuned using these formulas (Eqs. 10–12). It should be emphasized that they are not applicable to all situations. They cannot be applied to integrating processes. Their accuracy would be suspect if applied to any system-specific modification of the control algorithm or if the process dynamics are of similar magnitude to the scan interval. And, while an effective rule-of-thumb, the 15% *MV* overshoot limit might be relaxed on processes that would tolerate more aggressive corrective action. However any of the assumptions made in developing the formulas can be modified by reverting to the simulation tool, using it to generate either other sets of curves or tuning for any specific case.

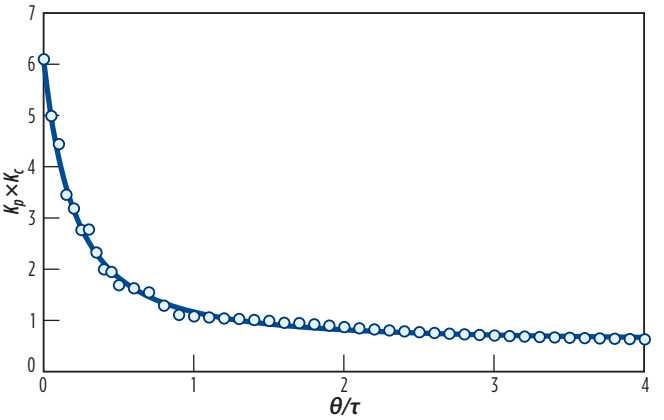

FIG. 5. Tuning chart for controller gain.

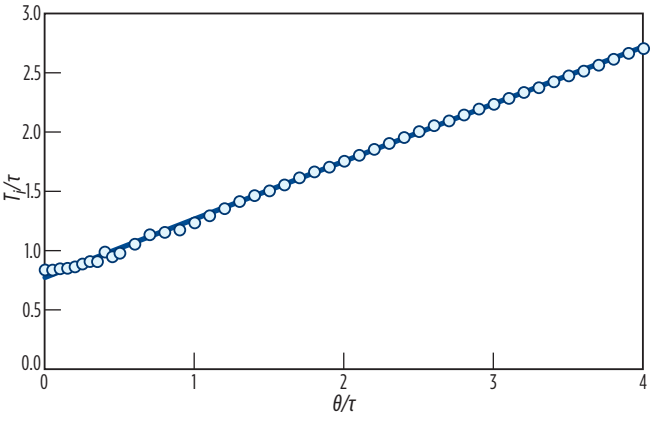

FIG. 6. Tuning chart for integral time.

How the engineer specifies the use of the proportional-on-*PV* algorithm varies from vendor to vendor and even between different systems provided by the same vendor. Careful review of the system documentation is required. Some vendors offer the choice of control equations. Others provide additional parameters in a single equation, for example, as:

It is unrealistic to expect simple formula-based methods to generate effective tuning constants for all the situations where they are likely to be applied. First, there are two fundamentally different process types—*self-regulating* and *integrating*.

$$\Delta M = K_c\left[(x_n - x_{n-1}) + \frac{ts}{T_i}E_n + \frac{T_d}{ts}(y_n - 2y_{n-1} + y_{n-2})\right] \quad (13)$$

$$x = PV - \alpha SP \quad \text{and} \quad y = PV - \beta SP$$

This is sometimes described as the *two degrees of freedom controller.* Setting α and β to 0 will result in the controller described in Eq. 4, while setting them both to 1 will result in the version described by Eq. 8.

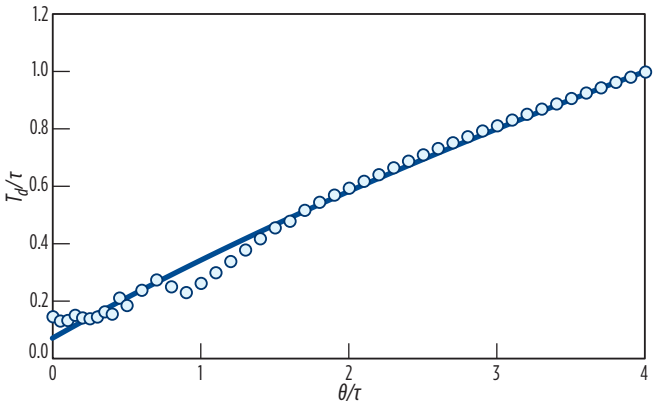

FIG. 7. Tuning chart for derivative time.

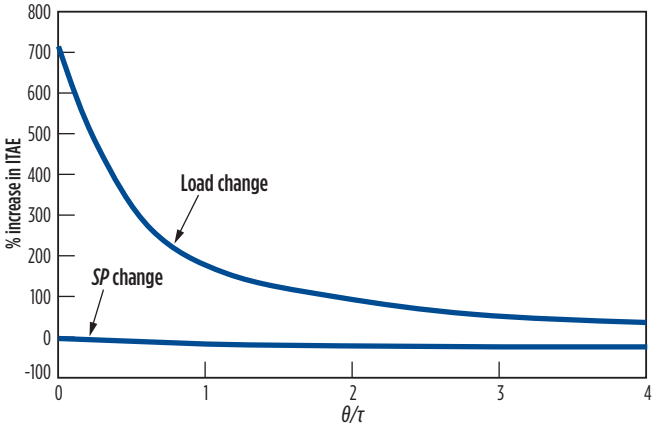

FIG. 8. Impact of switching from the proportional-on-*PV* to the proportional-on-*E* algorithm.

We should consider under what circumstances we should apply the proportional-on-*PV* algorithm. Its omission of any proportional action when the *SP* is changed will always make it less effective than the proportional-on-*E* version, even if tuned specifically to handle this disturbance. However, as shown in **FIG. 2,** the difference in performance would probably be indistinguishable in practice. **Remember:** Stand-alone controllers are generally subject to frequent process disturbances and relatively few *SP* changes; thus, the decision to select the proportional-on-*PV* algorithm is clear.

Less clear is when the controller is the secondary of a cascade. Under these circumstances, *SP* changes will be frequent—as indeed they would be if the *SP* is the *MV* of a higher level multivariable predictive controller (MPC). However, as shown in **FIG. 8,** while selecting the proportional-on-*E* will slightly reduce the ITAE for *SP* changes, and it will substantially worsen it for load changes. Given that secondary controllers will usually be on processes with a very low θ/τ ratio, the approximately 10% improvement in response to *SP* changes will be at the expense of a 600% degradation in the response to load changes. The secondary would be subject to very few process disturbances to justify selection of the proportional-on-*E* algorithm.

Most controllers in industry are configured as the proportional-on-*E* type. Modification is not trivial, since they then require re-tuning. There is no simple formula for converting the tuning from one algorithm to that for another. Plant testing would be necessary to obtain the process dynamics. Further, some controllers could generate far greater redesign. For example, those on a process which has MPC installed could require repetition of the step-testing performed for MPC design. Control engineers must address the question as to which controllers would show sufficient improvement to justify modification. Certainly, improving a simple flow controller so that it handles disturbances in two seconds, where before it took around five seconds, would probably go unnoticed. But halving the time it takes a relatively slow temperature controller so that it recovers from a disturbance in five minutes rather than 10 minutes can substantially improve process profitability. On new installations, there is much to be said for adopting the proportional-on-*PV* algorithm throughout. HP

LITERATURE CITED

[1] O'Dwyer, A., "A summary of PI and PID controller tuning rules for processes with time delay," IFAC Digital Control: Past, Present and Future of PID Control, Terrassa, Spain, 2000.
[2] Chien, I-L. and P. S. Fruehauf, "Consider IMC tuning to improve controller performance," *Chemical Engineering Progress,* June 1990, pp. 33–41.
[3] http://www.whitehouse-consulting.com/tune.htm.
[4] King, M., *Process Control: A Practical Approach,* Wiley, Chichester, 2011, pp. 51–77.

MYKE KING is the author of *Process Control: A Practical Approach.* He is the director of Whitehouse Consulting and is responsible for consultancy services, assisting clients with improvements to basic controls, and with the development and execution of advanced control projects. Mr. King has 35 years of experience in such projects, working with many of the world's leading oil and petrochemical companies. He holds an MS degree in chemical engineering from Cambridge University, and is a Fellow of the Institute of Chemical Engineers.