Are you losing money when tuning controllers?

Here are 10 rules, if followed, that will result in poor process performance

M. J. KING, Whitehouse Consulting, Isle of Wight, UK

From published literature, there is a wide range of pitfalls into which control engineers frequently stumble.[1-3] As these pitfalls were documented, more were discovered. In this article, we will focus on a very specific area of control design. The following rules investigate how substantial deterioration in process performance is possible in proportional, integral and differential (PID) control systems.

Rule 1. Use the 'derivative-on-error' algorithm. The PID algorithm in its conventional analog form is usually written as:

$$M = K_c\left[E + \frac{1}{T_i}\int E \times dt + T_d\frac{dE}{dt}\right]$$

Despite this or, more often, its equivalent in Laplace form, being used in most distributive control systems (DCSs) vendors' documentation it strictly applies only to analog control. A close digital equivalent is:

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

The problem with this algorithm is that when the setpoint (SP) is changed, assuming the process was previously at steady state, the derivative action causes an immediate step change in output, given as:

$$\Delta M = \frac{K_c T_d}{ts}\Delta SP$$

This is followed, at the next scan interval, by the same change in the opposite direction. Known as the "derivative spike," it can readily move the manipulated variable (MV) full scale. T_d might typically have a value of around 1 minute, and ts will be about 1 second. Even with quite a modest value for K_c, ΔM can exceed 100%. Fortunately, most DCS vendors have modified the algorithm to:

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

Known as the "derivative-on-PV" algorithm, the derivative action no longer responds to changes in SP. However, the response to changes in process variable (PV), caused by process disturbances (or "load" changes), is unaffected. Some DCS vendors have retained the derivative-on-error version as an option—unfortunately, often as the default version. A poorly trained engineer might think that, since it bears the closest resemblance to the conventional analog version, it should be the one to apply. This seriously limits the use of derivative action in those situations where it would be particularly beneficial (See Rule 7).

Rule 2. Use the 'proportional-on-error' algorithm. Using this algorithm is almost entirely to blame for hiding opportunities to substantially improve the performance of controllers responding to process disturbances. The alternative "proportional-on-PV" offered as an option in most DCS is described as:

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

At first glance, this might appear to have a serious disadvantage. When the SP is changed, the more conventional proportional-on-error algorithm generates a "proportional kick" equal to $K_c \Delta SP$—doing much to ensure that the SP is approached rapidly. The proportional-on-PV version does not do this, relying entirely on the much slower integral action. Many engineers reject this algorithm solely because of this perceived problem. However, they overlook the fact that the controller can be re-tuned to compensate for the loss of the proportional kick. As shown in Fig. 1, with effective tuning, its response to SP changes would be virtually indistinguishable, by the process operator, from that of the algorithm it replaces.

Its benefit becomes clear when the performance of the two algorithms, both tuned for SP changes, is compared for load

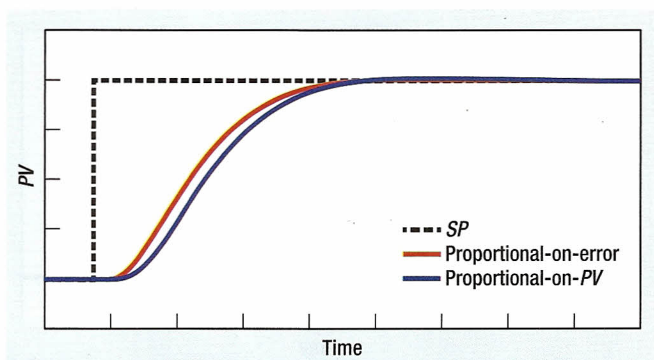

FIG. 1 Response to a SP change.

changes. With the same tuning, provided the *SP* remains constant, the two algorithms perform identically. The much faster tuning necessary to make the proportional-on-*PV* algorithm perform well for *SP* changes causes it to respond much faster to load changes. Fig. 2 shows that both the duration of the disturbance and the maximum deviation from *SP* are typically halved. Were the *PV* to be related to product composition, the volume of off-spec production would be reduced by more than 75%.

Of course, it would be possible to achieve the same improvement by applying the tuning developed for the proportional-on-*PV* algorithm to the proportional-on-error version. However, it would then cause a major process upset whenever the *SP* is changed. This perhaps explains why the algorithm is not fully appreciated. Many engineers select the more conventional proportional-on-error algorithm and tune it for *SP* changes. Its response to load changes will then appear reasonable but will disguise the fact that the response can be substantially improved.

Rule 3. Use the interactive algorithm. There is an alternative derivation of the PID controller. It starts with a conventional PI controller, but adds the derivative action by replacing the *E* term with a "projected error" defined as:

$$\hat{E} = E + T_d \frac{dE}{dt}$$

This results in a slightly different algorithm:

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

Comparison with the so-called "ideal" form described earlier shows that the integral and derivative actions are unchanged but the proportional action depends not only on K_c but also now on T_i and T_d—thus earning the algorithm its "interactive" name. Some DCS use this version, either as the only choice or as an option. It exists primarily because it closely matches the action of pneumatic analog controllers and their early electronic replacements.

Using it these days presents no problem provided the tuning method chosen is specifically designed for the changed algorithm. Indeed, provided that in the ideal algorithm T_d is less than 0.25 T_i, it is possible to calculate equivalent tuning for the interactive version so that the performance of the two algorithms is identical. And if the derivative is not used, then both algorithms are the same in any case.

The problem arises because DCS vendors rarely retain the algorithm in its pure form. It is common to include a "derivative filter" (usually given the nomenclature as a or α) or a "derivative gain limit" (which is the reciprocal of a). This value may be fixed within the system or configurable by the engineer. It usually makes impossible adapting a tuning method designed for the ideal algorithm for use with the interactive form.

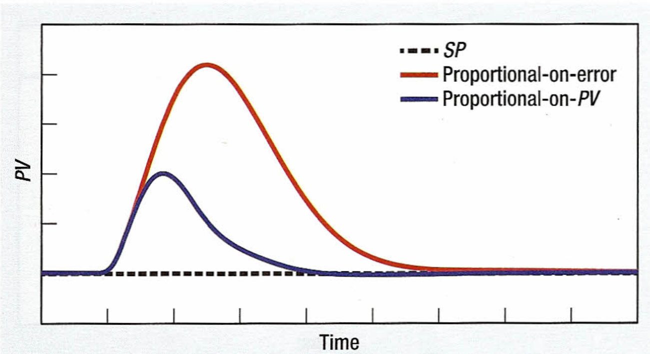

FIG. 2 Response to a load change.

Rule 4. Apply Ziegler-Nichols tuning. Amazingly, Ziegler-Nichols is still by far the most popularly taught tuning method. It was developed 70 years ago.[4] Few appreciate that it assumes the now rare interactive version of the PID algorithm. Even fewer know that it was developed for load changes and so, if applied to the normal proportional-on-error algorithm, will result in far too an aggressive response to a change in *SP*. And, even if these issues are resolved, its main objective is to deliver the "quarter decay ratio," where the height of the second *PV* overshoot is one quarter of the height of the first. Few now accept that *any* amount of second overshoot is the sign of a well-tuned controller. The more cynical control engineer might think inclusion of the method in papers and textbooks is to establish a benchmark by which even a poorly performing alternative would look good.

Another commonly reproduced method is that developed by Cohen-Coon.[5] It too uses the quarter decay ratio and was developed using analog control almost certainly equivalent to the interactive algorithm. If anything, its performance is somewhat inferior to Ziegler-Nichols.

Rule 5. Ignore the MV. Effective controller tuning is often a compromise between a fast return to *SP* and avoiding excessive changes to the *MV*. Many tuning methods use a penalty function, such as the integral over time of absolute error (ITAE), as a measure of control performance:

$$ITAE = \int_0^\infty |E| t.dt$$

Minimizing such functions results in the fastest possible return to *SP* but, if the deadtime-to-lag ratio is small, this will result in excessive adjustments to the *MV*. As the deadtime-to-lag ratio approaches zero, such methods recommend a controller gain approaching infinity. One such method is that developed by Smith, Murrill and others.[6,7] Defining the *MV* overshoot as the percentage by which the peak change in *MV* exceeds the necessary steadystate change, we can supplement this type of tuning criterion by minimizing the penalty function subject to a limit on *MV* overshoot. Typically, a 15% limit results in what most would accept as a well-tuned controller. However, the limit may be increased if large changes in *MV* do no harm and similarly reduced if the aim is to minimize *MV* movement. Indeed the latter, in the case of surge vessel level control, is the overriding consideration, and large deviations from level *SP* should be the norm.

One of the few published methods that permits the engineer to specify the compromise between fast return to *SP* and *MV* movement is internal model control (IMC) tuning. Several companies have adopted this method as standard. However, it does have a number of disadvantages. The method is derived using "direct synthesis," which develops a control algorithm that will respond to an *SP* change according to a defined trajectory. This is usually specified as an approach to *SP* with a user-specified lag of λ. The resulting tuning equations vary greatly. For example, it can be applied to both self-regulating and integrating processes, using either the ideal or interactive algorithm. The synthesis usually includes terms that are not part of the PID algorithm and, so, some approximation is necessary or the terms simply ignored. Different developers reach different conclusions. But a common example for the ideal PID algorithm applied to a self-regulating process is:

$$K_c = \frac{1}{K_p}\frac{\tau+\frac{\theta}{2}}{\lambda+\theta} \qquad T_i = \tau+\frac{\theta}{2} \qquad T_d = \frac{\tau\theta}{2\tau+\theta}$$

While the method permits the user to decide how aggressive the control should be, the value of λ has to be determined by the trial-and-error method. While some texts provide some guidance, there is no predictable relationship between its value and *MV* overshoot. Under a different set of process dynamics, the relationship changes. It is possible to develop formulae for the best choice of λ. For example, choosing a value given by 0.31θ + 0.88τ will give an *MV* overshoot of 15%, but only for the proportional-on-error form of the ideal controller applied to a self-regulating process. We would need to develop such formulae not only for different controllers and for integrating processes but also for different *MV* overshoot limits. While perhaps possible, the most damning limitation of this tuning method is that no one has yet published the formulae for the preferred algorithm—where both proportional and derivative actions are based on *PV* rather than error.

Rule 6. Ignore the scan interval. The industry has now begun replacing first generation DCSs with their more modern counterparts. Engineers have been surprised to find in some cases that this has apparently increased the level of measurement noise. This can arise because of the faster scanning that may be available in the new system. Fig. 3 shows how the total valve travel generated by a PID controller varies as scan interval changes. The curve starts at a ts/τ ratio of 1/120—equivalent to a controller with a scan interval of 1 second on a process with a lag of 2 minutes. Defining the total valve travel under these conditions as 100%, we can see that, for a PID controller, reducing the scan interval from 2 seconds to 1 would increase valve travel by a factor of 4.

All DCS include the ability to filter a measurement and most use the first order exponential type. The digital version of this filter is often defined as:

$$Y_n = P \times Y_{n-1} + (1-P)X_n$$

where

$$P = \exp(-ts/\tau_f)$$

Changing the scan interval of a controller in a system in which the engineer defines *P* directly will result in a different filter lag. Even the most modern of controller tuning methods still assumes analog control. While this is of little concern when the scan interval is small compared to the process dynamics, it can cause problems otherwise. For example, compressor-surge protection systems are applied to a process where the deadtime is effectively close to zero and the lag only a few seconds. Tuning such controllers without taking account of scan interval will drastically affect performance. It goes some way to explain why package vendors (usually mistakenly) insist that compressor controls can only be implemented in special purpose control systems that have a much shorter scan interval.

Rule 7. Avoid using derivative action. Depending on the textbook a control engineer might read, if the process has a large deadtime, the derivative action is either beneficial or becomes less effective. In fact, it offers an advantage on processes with either little or a large deadtime—depending on the disturbance source. Fig. 4 shows the impact on ITAE of removing deadtime from a well-tuned controller, and retuning the *PI* controller as well as possible. It shows that for *SP* changes, removing derivative action causes controller performance to deteriorate more on processes that have a larger deadtime-to-lag ratio. For load changes, the opposite is true. But, for both cases, the effect of removing it is always adverse, and, in any case, most controllers have to deal with both disturbance types.

In practice, the derivative action is only used by a minority of controllers. There are several reasons for this. First, it has a reputation for causing problems if there is measurement noise. Certainly, it will grossly amplify noise, but modern DCSs do offer a wide range of filtering techniques that can readily reduce noise to a point where derivative action is viable. Second, it adds another tuning parameter. Adding derivative action requires the proportional and integral tuning to be readjusted. Fig. 5 shows that the addition of derivative action is beneficial because it permits a larger controller gain. If the engineer has already spent hours tuning a PI controller by the trial-and-error method, there will be an understandable reluctance to abandon this tuning and start afresh with a three-dimensional search.

Rule 8. Use filters to improve PV trending. Most control engineers use filters to make the *PV* trend look good. Gone are the days when we have to concern ourselves with the amount of ink used in drawing such trends. A better criterion is to examine the movement of the final actuator, usually a control valve. This will depend not only on the amplitude of the measurement

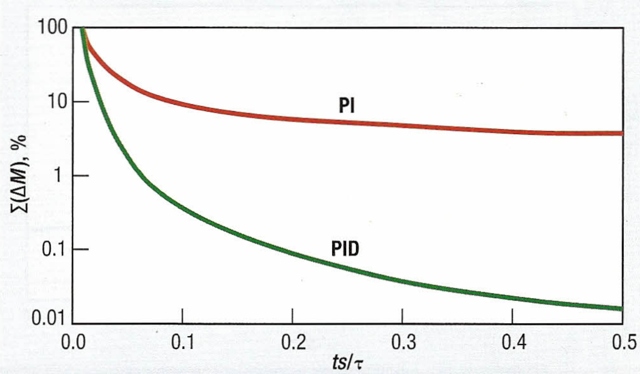

FIG. 3 Effect of scan interval on noise passed to an actuator.

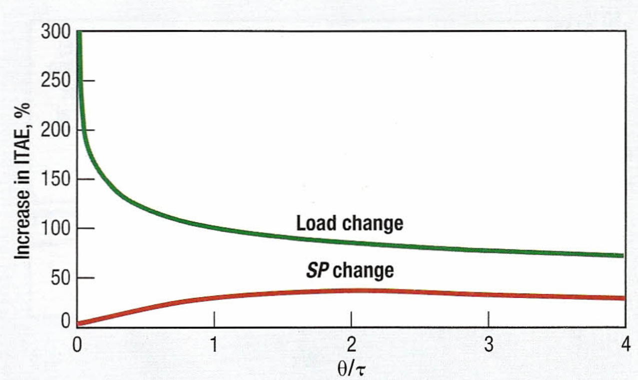

FIG. 4 Impact in ITAE of removing derivative action.

noise but also on the controller tuning. If the impact on valve movement is acceptable, then the filter serves no purpose and will reduce the controllability of the process. Its presence means that tuning has to be relaxed to maintain stability. Conversely, we must remember that, if a filter is removed, then the benefit will not be apparent until the controller is re-tuned to accommodate the change in apparent process dynamics.

Filtering can be beneficial if it permits greater use of derivative action. Since derivative action responds to the rate of change error, the small fluctuations in signal occurring at a high frequency are greatly amplified. Many DCSs now offer the facility to selectively filter only the measurement passed to derivative action. This permits derivative to be used without changing the dynamics seen by the proportional and integral actions.

Rule 9. Tune by trial-and-error methods. Over 200 tuning methods have been published.[8] All of them have at least one flaw. It is not surprising that control engineers have generally adopted the trial-and-error method as the main tuning method. It requires no knowledge of the process dynamics and little understanding of the control algorithm being applied. But its main disadvantage is that it is extremely time-consuming. Trials conducted on a simulated process with dynamics of a few minutes showed that engineers would spend around 30 minutes finding the best tuning. Quite a modest investment one might think until one realizes that the simulation was running much faster than real time and each test was exactly reproducible. On the equivalent real process such an exercise would easily have filled a working week. In practice, no engineer can commit this time to a single controller and will stop trying to improve its performance once it is stable and looks "about right." The result is that the process operator will likely be unimpressed by its performance during the next process upset and will switch the controller to manual.

Developers of tuning methods have attempted to develop a set of tuning formulae that can be applied to any situation. In reality, such an approach is unlikely ever to be successful. There are two fundamentally different processes: self-regulating and integrating. There are two fundamentally different PID algorithms: ideal and interactive. Some versions of the algorithm include a derivative filter that cannot be changed by the user. Proportional action can be based on error or *PV*, as can derivative action. These options are not mutually exclusive; just considering those listed so far gives 32 possible combinations. If we add to this the requirement to specify the aggressiveness of the control, allow for different scan intervals and to take account of vendor-specific modifications to the algorithm, then the number of sets of tuning formulae grows to an impractical level.

Figs. 6–8 show comparisons between the commonly published tuning methods and user-defined optimum tuning. For the comparisons to be fair, the controller was assumed to be analog and subject to a *SP* change. The results were obtained by using a tuning constant optimizer freely available.[9] In this case, the optimum tuning was specified as minimum ITAE subject to a 15% *MV* overshoot limit. So, unlike many methods, the optimized controller gain does not approach infinity as θ/τ approaches zero. The IMC method appears to estimate the con-

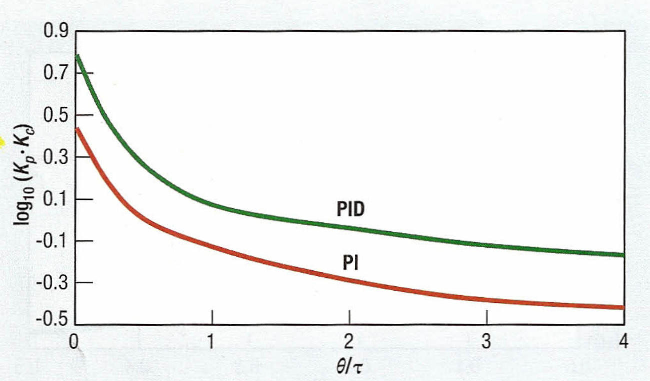

FIG. 5 Impact on Kp of inclusion of derivative action.

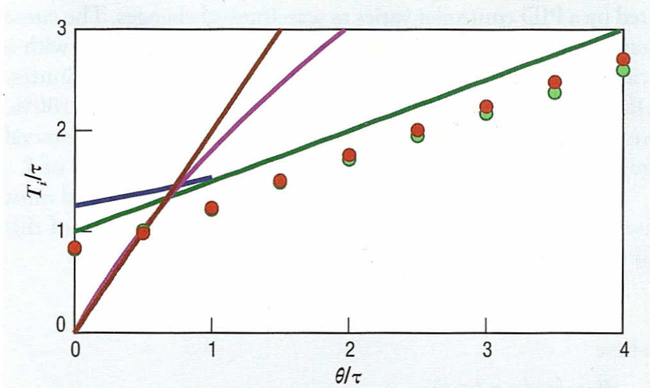

FIG. 7 Determination of integral time.

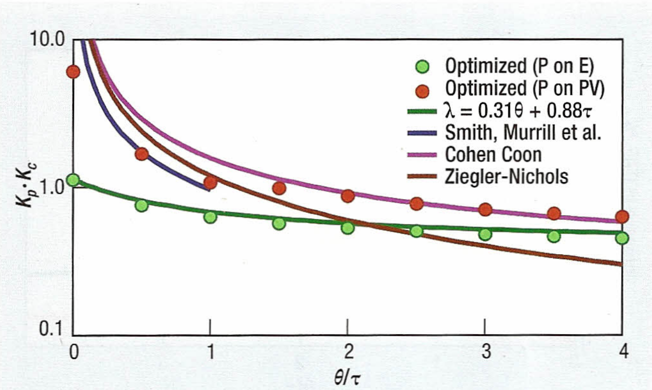

FIG. 6 Determination of process gain.

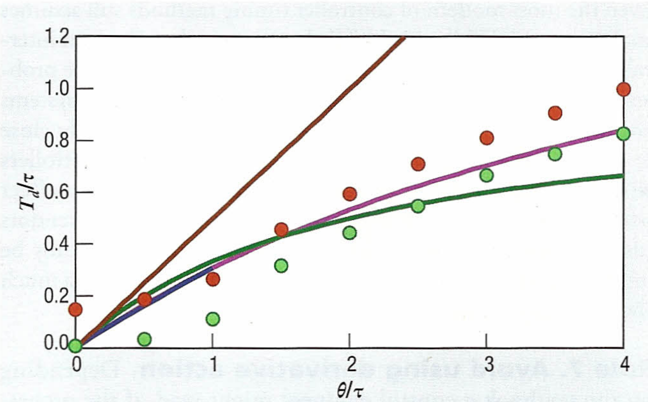

FIG. 8 Determination of derivative time.

troller gain well, but only because the choice of λ has been optimized for this particular case. Note: The method developed by Smith, Murrill and others is only applicable to values of θ/τ less than 1. Outside of this range, it can generate negative tuning constants. But, most importantly, optimization permits tuning to be derived also for the preferred proportional-on-*PV* algorithm. The much higher gains derived for this controller will substantially reduce the impact of process disturbances.

Rule 10. Don't train engineers in basic control. The most effective way of reducing process profitability is to ensure that the control engineers are kept completely unaware of what can be achieved by minor changes to PID control. Those that have studied control theory at university will have been subjected to daunting mathematics, much of which is irrelevant to the process industry. Almost certainly little will have been covered on the alternative forms of the PID algorithm, let alone which one to use and how to properly tune it.

While it is common practice to send staff on vendor supplied courses in DCS programming and multivariable predictive control (MPC), it is rare to consider also training in basic control techniques. Industry seems to expect engineers to somehow acquire this expertise without outside assistance. This ensures that the techniques described above, many of which have been available for over 30 years, are still not properly appreciated and that plants continue to operate away from maximum profitability. **HP**

NOMENCLATURE

Complete nomenclature available online at HydrocarbonProcessing.com.

LITERATURE CITED

1 King, M. J., "How to lose money with advanced controls," *Hydrocarbon Processing,* June 1992, pp. 47–50.

2 King, M. J., "How to lose money with basic controls," *Hydrocarbon Processing,* October 2003, pp. 51–54.

3 King, M. J., "How to lose money with inferential properties," *Hydrocarbon Processing,* October 2004, pp 47–52.

4 Ziegler, J. G. and N. B. Nichol," Optimum settings for automatic controllers," Transactions of the ASME, 64, pp. 759–768, 1942.

5 Cohen, G. H. and G. A. Coon, "Theoretical considerations of retarded control," Transactions of the ASME, 75, pp. 827–834, 1953.

6 Smith, C. L., *Digital Computer Process Control,* Intext Educational Publishers, p. 147, 1972.

7 Lopez, A. M., J. A. Miller, C. L. Smith and P. W. Murrill, "Tuning controllers with error-integral criteria," Instrumentation Technology, 14, pp. 57–62, 1967.

8 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.

9 http://www.whitehouse-consulting.com/tune.htm.

BIBLIOGRAPHY

King, M., *Process Control: A Practical Approach,* published by Wiley, ISBN 978-0-470-97587-9.

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