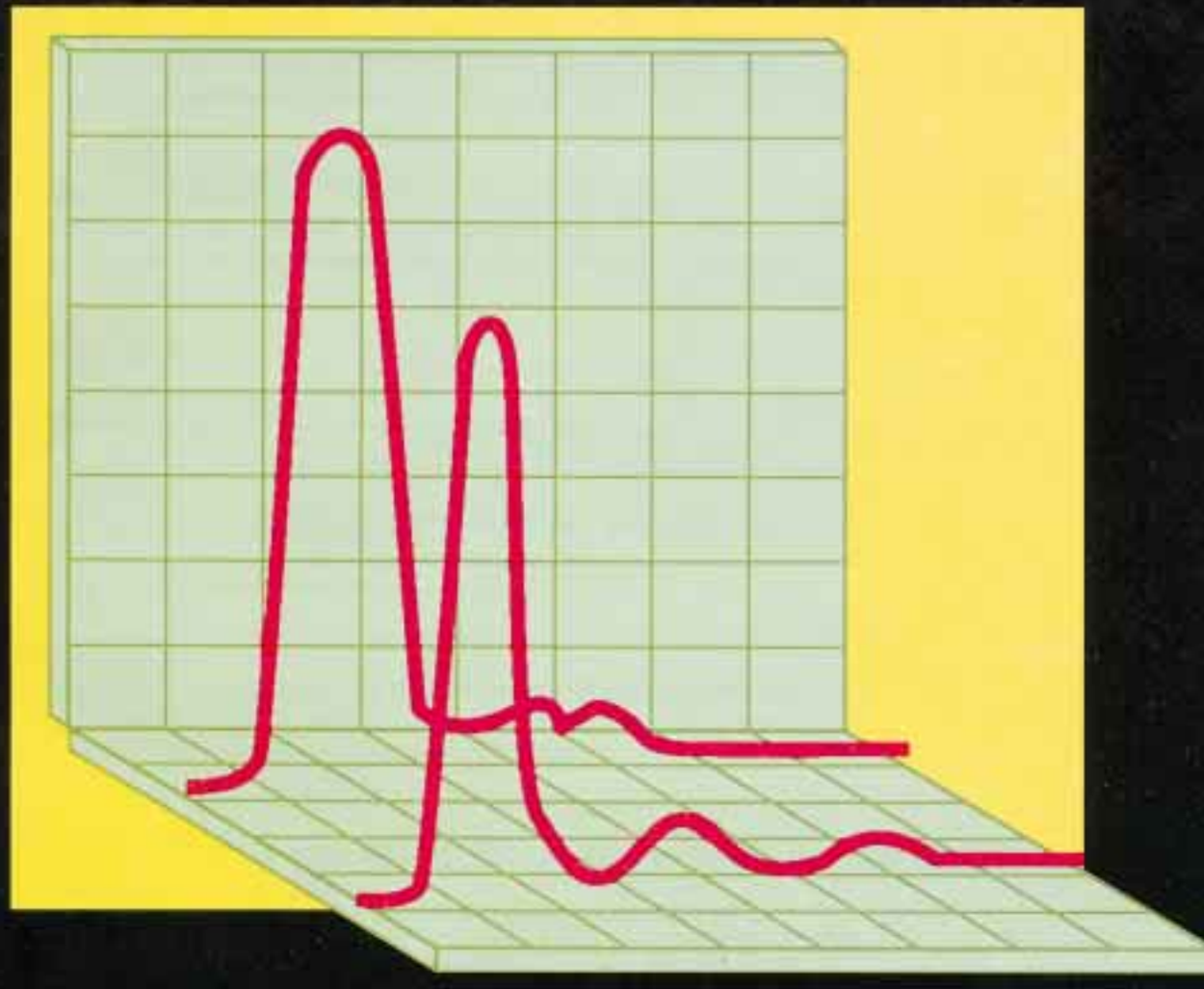
CONTROL ENGINEERING

A CAHNERS PUBLICATION

for designers and users of control and instrumentation equipment and systems world wide

Reference Guide to PID Tuning

Part 2



A collection of reprinted articles of PID tuning techniques

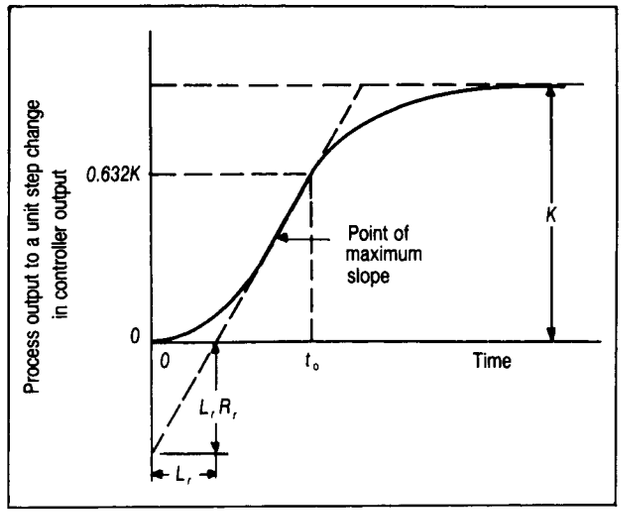


FIG. 1. Many industrial processes can be approximated by a first order lag plus dead time. The figure illustrates two methods for graphic approximation, both utilizing a tangent constructed on process response.

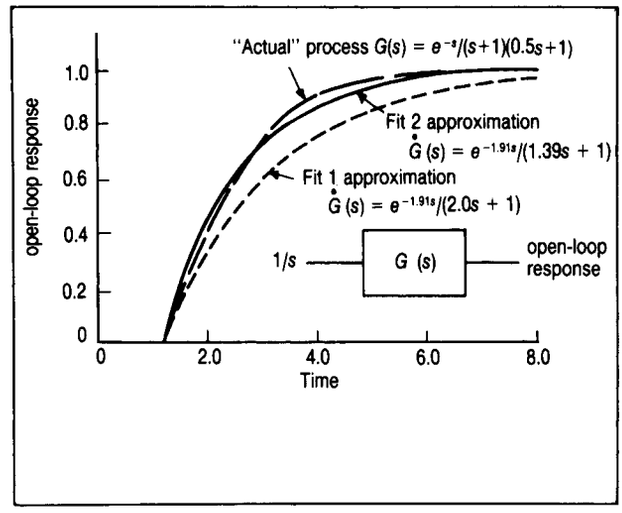


FIG. 2. Comparison of the two approximation methods (Fits 1 and 2). As evident in the diagram, Fit 2 proves to be a better approximation where the process is assumed to be a second order lag plus dead time.

A Comparison of Controller Tuning Techniques

J. A. MILLER, A. M. LOPEZ, C. L. SMITH, and P. W. MURRILL, Lousiana State University

Here is a clear, concise comparison of techniques for tuning controllers, and a set of conclusions that explain why one of the procedures appears superior to the others. The authors begin with an investigation of two methods for approximating the process reaction curve and demonstrate the one that is better. Using this method they develop tuning relations by applying several techniques. The resultant graphical comparisons document their choice of the best technique.

From among the many techniques for adjusting controllers based on the open-loop process response, four have been chosen for purposes of comparison (Ref. 1—4). All four are based on the process reaction curve discussed below. The tuning parameters are proportional gain K_c, reset time T_i, and rate time T_d. They are derived from the assumption of an ideal controller characterized by the algorithm:

$$m(t) = K_c \left[e(t) + \frac{1}{T_i} \int e(t)dt + T_d \frac{de(t)}{dt} \right] \quad (1)$$

where $m(t)$ is the controller output signal, and $e(t)$ is the error signal.

Process reaction curve

A process reaction curve is the process response to a unit step change in the manipulated variable (controller output). Many industrial processes can be adequately approximated by a relatively simple mathematical model consisting of a first order lag with a deadtime:

$$\text{Output} = \text{Input} \cdot \frac{K \exp(\theta_o s)}{(\tau + 1)} \quad (2)$$

Figure 1 illustrates two approximation methods. In each method a tangent is drawn to the process reaction curve at its point of steepest ascent. Extreme care must be exercised to select this point accurately. Then reaction rate R_r (tangent slope), process gain K, and the time delay L_r are determined.

The two methods differ in their manner of obtaining first order lag time. For Fit 1 the time constant $\tau_1 = K/R_r$, and for Fit 2 it is $\tau_2 = t_o - \theta_o$. Since L_r is the dead time in Figure 1, $\theta_o = L_r$, and $\tau_2 = t_o - L_r$.

Figure 2 presents a comparison of these two fits where the actual process is taken to be a second order lag plus dead time. Fit 2 proves to be a much better approximation.

Going from open- to closed-loop comparison of fit methods, Figure 3 confirms the superiority of Fit 2. Two sets of plots are required since different con-

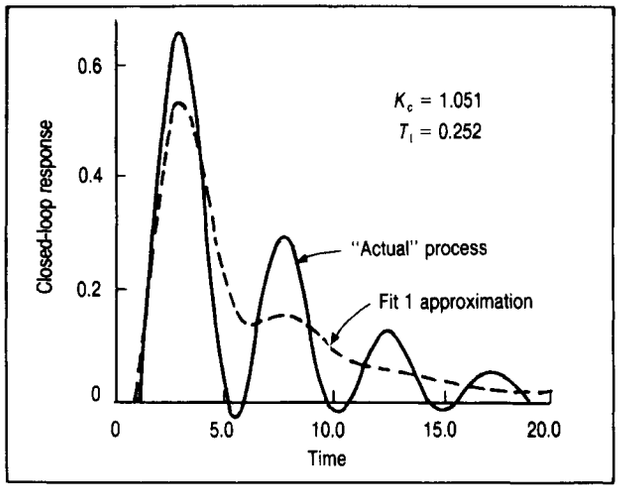


FIG. 3. The superiority of Fit 2 is confirmed in a closed-loop comparison. Two sets of plots are required since different controller settings were obtained for Fits 1 and 2 because of their different time constants.

troller settings were obtained for Fits 1 and 2 because of their different first order time constants.

It is concluded that Fit 2 should be used in comparing controller tuning techniques.

Open-loop tuning relations

The purpose of open-loop tuning techniques is to determine values of tuning parameters (K_c, T_i, and T_d), given values of process parameters (K, L_r, and R_r). The procedure for developing tuning relations has been to establish a criterion for optimal control, then determine values of tuning parameters that will satisfy this criterion for a given combination of process parameters.

The Ziegler-Nichols criterion for optimal control (Ref. 1) was that the response of the controlled process to a unit step change in distrubance should have a 1/4 decay ratio, Figure 4. A similar criterion was developed by Cohen and Coon, Ref. 2. The difference was in the presentation of the terms L_r, R_r, and K. An index of self-regulation was introduced, defined as $\mu = R_r L_r / K$. The resulting model was more complex than Ziegler and Nichols'.

A third set of tuning relations took advantage of the simplicity of the 1/4 decay ratio criterion, and overcame its disadvantages by adding three constraints—hence, its designation $3C$. These disadvantages were the inability of 1/4 decay ratio criterion to determine unique values of tuning parameters for two and three mode controllers, and the inability to characterize an entire closed-loop response (Ref. 4).

Tuning relations designated as $3C$ are expressed as dead time θ_o first order time constant τ, and gain K. They are related to L_r, R_r, and μ:

$$\left.\begin{aligned} \theta_o &= L_r \\ \tau &= KL_r/L_rR_r = K/R_r \\ \theta_o/\tau &= \mu = R_rL_r/K \end{aligned}\right\} \qquad (3)$$

Originally presented in graphical form, $3C$ relations are least squares approximated very well by equations of the form:

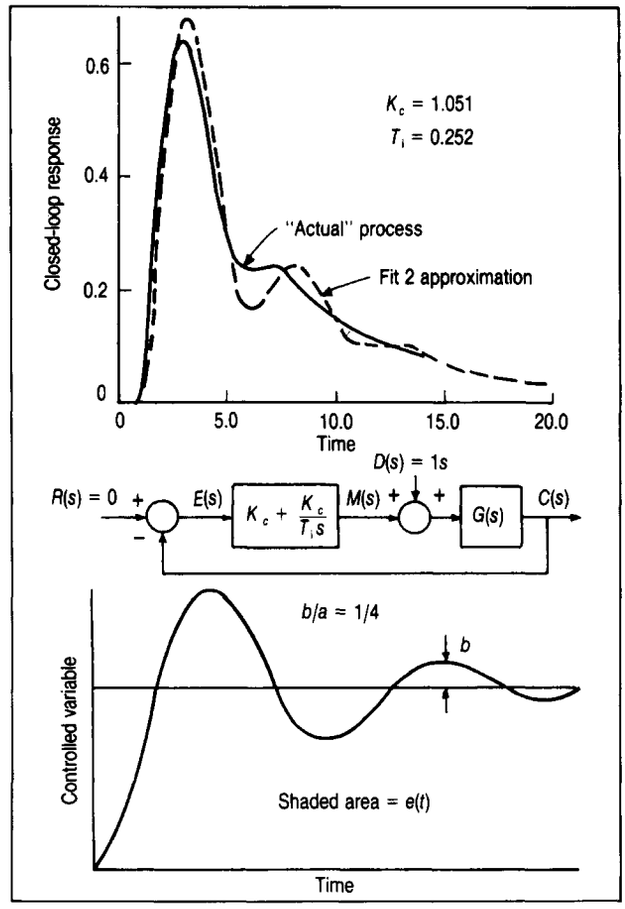


FIG. 4. Optimum control criterion based on requirement that response to a unit step change in disturbance should have a 1/4 decay ratio.

Table. Equations for six sets of tuning relations

Cohen-Coon tuning relations			
P	Proportional	KK_c =	$(\theta_o/\tau)^{-1.0} + 0.333$
PI	Proportional	KK_c =	$0.9(\theta_o/\tau)^{-1.0} + 0.082$
	Reset	T_i/τ =	$[3.33(\theta_o/\tau) + 0.3(\theta_o/\tau)^{2.0}]/[1 + 2.2(\theta_o/\tau)]$
PID	Proportional	KK_c =	$1.35(\theta_o/\tau)^{-1.0} + 0.270$
	Reset	T_i/τ =	$[2.5(\theta_o/\tau) + 0.5(\theta_o/\tau)^{2.0}]/[1 + 0.6(\theta_o/\tau)]$
	Rate	T_d/τ =	$0.37(\theta_o/\tau)/[1 + 0.2(\theta_o/\tau)]$
Ziegler-Nichols (ZN), 3C, IAF, ISE, and ITAE tuning relations:			
General algorithm:		$m(t)$ =	$K_c[1 + 1/T_i s + T_d s]\, e(t)$
Form of equations:		KK_c =	$A(\theta_o/\tau)^{-B}$
		T_i =	$C(\theta_o/\tau)^{D}$
		T_d =	$E(\theta_o/\tau)^{F}$

Computed constants for substitution in above equations:

Technique	Mode	A	B	C	D	E	F
ZN	P	1.000	1.000				
3C	P	1.208	0.936				
IAE	P	0.902	0.985				
ISE	P	1.411	0.917				
ITAE	P	0.490	1.084				
ZN	PI	0.900	1.000	3.333	1.000		
3C	PI	0.928	0.946	0.928	0.583		
IAE	PI	0.984	0.986	1.644	0.707		
ISE	PI	1.305	0.959	2.033	0.739		
ITAE	PI	0.859	0.977	1.484	0.680		
ZN	PID	1.200	1.000	2.000	1.000	0.500	1.000
3C	PID	1.370	0.950	0.740	0.738	0.365	0.950
IAE	PID	1.435	0.921	1.139	0.749	0.482	1.137
ISE	PID	1.495	0.945	0.917	0.771	0.560	1.006
ITAE	PID	1.357	0.947	1.176	0.738	0.381	0.995

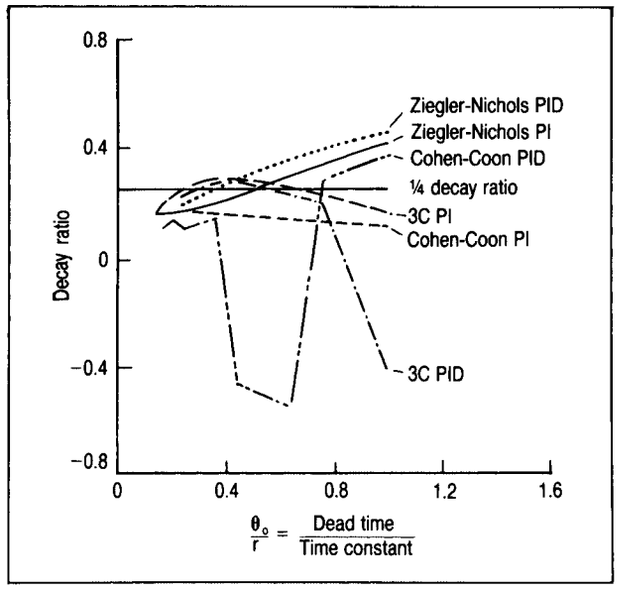


FIG. 5. Plots of decay ratio versus ratio of dead time to time constant indicating 1/4 ratio is poor criterion for optimum control.

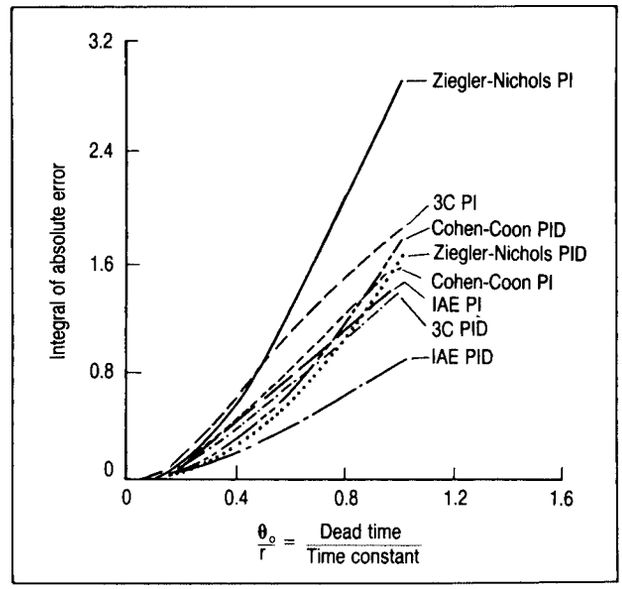


FIG. 6. Generalized comparison of tuning relations for PI and PID controllers based on the criterion of integral of absolute error (IAE).

$$Y = A\left(\frac{\theta_o}{\tau}\right)^B \tag{4}$$

where Y = KK_c for proportional mode
= T_d for rate mode
= T_i for reset mode
A, B= constants for a given controller and mode.

A fourth set of tuning relations is based on an optimum control criterion of minimizing some form of error integral (Ref. 3). An entire closed-loop response can be expressed by a single number which is a direct measure of the extent to which feedback eliminates error in the controlled variable.

These integral criteria may take three forms:

$$\text{IAE (minimum integral of absolute error)} = \int_0^\infty |e(t)|\,dt \tag{5}$$

$$\text{ISE (minimum integral of error squared)} = \int_0^\infty e(t)^2 dt \tag{6}$$

$$\text{ITAE (minimum integral of absolute error multiplied by time)} = \int_0^\infty t\,|e(t)|\,dt \tag{7}$$

Comparison based on open-loop, and closed-loop responses

The four approaches to tuning controllers actually yield six sets of tuning relations, since the fourth approach expands as indicated by equations 5, 6, and 7. Analytical comparison of the six methods for P, PI, and PID configurations is presented in the Table. Plotting these equations yielded notable variation in K_c from one tuning method to another. But the variations in tuning parameters were generally not sufficient for positive indications of which tuning relation was best. For this reason the comparison was made on the basis of closed-loop response.

Computer simulation was used to calculate closed-loop response to a unit step change in disturbance for a first order lag plus dead time process with a PI or PID controller. The simulation determined valures of the four criteria (decay ratio, IAE, ISE, and ITAE) for several values of normalized process paramter θ_o/τ. This was done for each value of θ_o/τ with a proportional plus reset and then a three-mode controller, each being tuned by all six tuning relations. The results obtained on an x-y plotter are presented in Figures 5 through 8.

Figure 5 indicates that 1/4 decay ratio is not only a poor criterion for optimum control, but also does not indicate which tuning method is best. Figures 6 through 8, on the contrary, indicate that integral criteria are good criteria, and identify the best method. They further indicate that the best integral tuning relation, providing optimum control, corresponds to its particular integral. For example, the IAE-PID tuning relation minimizes IAE, while the ISE-PID tuning relation minimizes ISE. Since the Ziegler-Nichols, Cohen-Coon, and 3C relations do not minimize integral criteria, the question of which technique is best reduces to a choice among IAE, ISE, and ITAE.

Figures 6 through 8 also indicate important points about the controller to be used for a particular process. The curves show that for values of θ_o/τ less than 0.4 a PI and PID configuration give essentially the same criterion value. Therefore a PID controller does not offer much advantage over a PI controller except where θ_o/τ is greater than 0.4. In this case, however, the PID controller should be tuned with an integral tuning relation to obtain full benefit from the derivative mode. For example, a PID controller tuned with the Ziegler-Nichols, Cohen-Coon, or 3C relations results in a higer ITAE than a PI controller

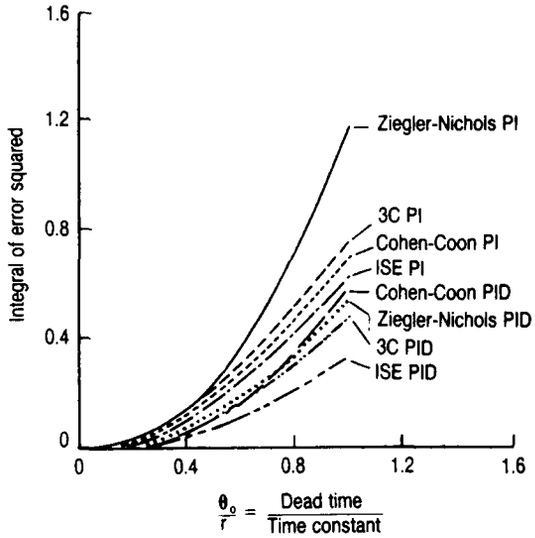


FIG. 7. Generalized comparison of tuning relations for PI and PID controllers based on the criterion of integral of squared error (ISE).

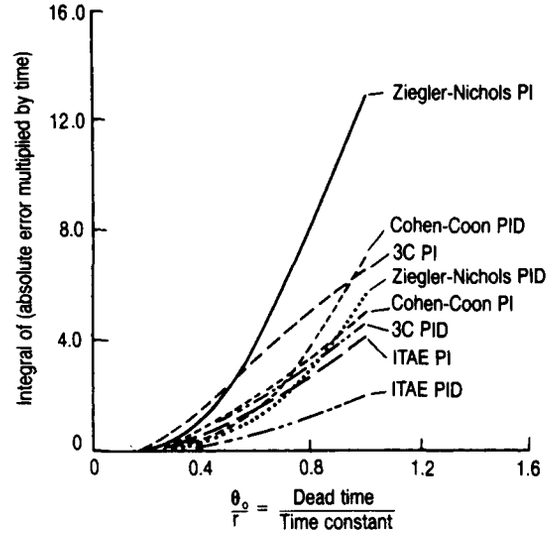


FIG. 8. Generalized comparison of tuning relations for PI and PID controllers based on the criterion of integral of absolute error multiplied by time (ITAE).

tuned with the ITAE relations. In general the full advantage of a PID controller is only realized when it is ITAE tuned and θ_o/τ is greater than 0.4.

Two specific cases are plotted in Figures 9 and 10 to illustrate the effect of θ_o/τ varying above and below the value of 0.4. Figure 9 shows that for $\theta_o/\tau = 0.1$, controller type and tuning is not too important, so a PID controller offers no great advantage. Figure 10 shows that for $\theta_o/\tau = 1.0$, the PID ITAE tuned controller provides a much better response than either PI controller. Note also that the ITAE tuned PI controller has a better response than the Ziegler-Nichols tuned PID controller.

It is concluded that a controller tuning method that utilizes integral criteria is superior to other methods that do not utilize such criteria. It is further concluded that of the three variations of integral criteria, the best one is ITAE, or the minimum integral of error—the integration taking place after the error function has been subjected to the weighting factor of time.

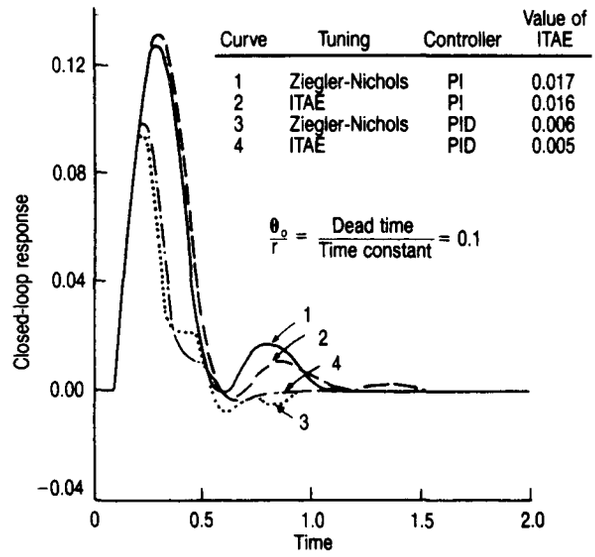


FIG. 9. Comparison of closed-loop responses for a first order lag plus dead time process with ratio of dead time to time constant equal to 0.1 (see front cover).

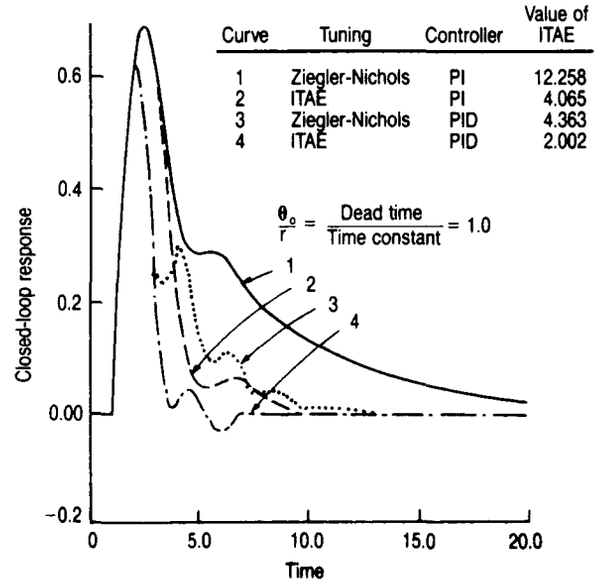


FIG. 10. Comparison of closed-loop responses for a first order lag plus dead time process with ratio of dead time to time constant equal to 1.0 (see cover).

REFERENCES

1. OPTIMUM SETTINGS FOR AUTOMATIC CONTROLLERS, J. G. Ziegler and N. B. Nichols, Transaction ASME, 64, pp. 759-765, (1942).
2. THEORETICAL CONSIDERATIONS OF RETARDED CONTROL, G. H. Cohen and G. A. Coon, Taylor Instrument Companies' Bulletin, #TDS-10A102.
3. CONTROLLER TUNING RELATIONSHIPS BASED ON INTEGRAL PERFORMANCE CRITERIA, A. M. Lopez, J. A. Miller, C. L. Smith, and P. W. Murrill, Instrumentation Technology, November 1967.
4. A MORE PRECISE METHOD FOR TUNING CONTROLLERS, C. L. Smith and P. W. Murill, ISA Journal, May,1966

A Comparison of PID Control Algorithms

JOHN P. GERRY, Gerry Engineering Software, Hubertus, Wis.

There are no industry-wide standards for PID controller algorithms, and definitions vary widely. A new process simulation program provides a look at these differences.

One fine day, a plant engineer replaced his old controllers with a different manufacturer's equipment. 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 the two manufacturers using different PID algorithms. Read on to solve this and other common mysteries about PID controllers.

The nomenclature game

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

The following equations show the relationships:

P (proportional band) = 100/gain

I (integral) = 1/reset (units of time)

D (derivative) = rate = pre-act (units of time)

Manufacturers interchange both names and units, sometimes independently, for integral or reset action. In this article, integral action is defined as time/repeat and reset as repeats/time. One is the reciprocal of the other. Be careful, the action of either reset or integral can actually be reversed, depending on the manufacturer's own terminology.

Most manufacturer's PID algorithms fit under one of three major classifications: interacting, noninteracting, and parallel. Here again, manufacturers differ on their names for these categories, so the only way to really tell which one you have is to look at the equation for the controller. In ideal form these are:

- Ideal noninteracting PID controller or ISA algorithm:

Controller output =

$$K_c \left[e(t) + \frac{1}{I} \int e(t)\,dt + D\frac{de}{dt} \right]$$

- Ideal parallel PID controller:

Controller output =

$$K_p\, e(t) + \frac{1}{I_p} \int e(t)\,dt + D_p \frac{de}{dt}$$

- Interacting PID controller:

Controller output =

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

where K_c and K_p are gain; I and I_p are integral; and D and D_p are derivative settings of the controller.

The interacting controller's odd-looking form makes it act like an electronic controller. By using the interacting form, a three-term controller can be made with only one amplifier. Thus, pneumatic controllers and early electronic controllers often used the interacting form to save on amplifiers, which were expensive at the time. Some manufacturers purposely use the interacting form in their digital algorithms in order to keep tuning similar to the tuning of electronic and pneumatic controllers.

Proportional action, or gain

If you use only proportional action, the main difference between interacting and noninteracting algorithms is that some manufacturers call the proportional gain coefficient "gain," while others call it "proportional band." On a controller using the "gain" nomenclature, increasing this setting generally makes the loop more sensitive and less stable. On the other hand, decreasing the "proportional band" will have the same effect.

Some manufacturers allow more flexibility with proportional action by letting you choose whether or not gain (proportional) action works on setpoint changes. For example, Honeywell offers two algorithms that work differently on setpoint changes. With Honeywell's type A algorithm, gain acts on set point changes, and with their type B it does not. For load upsets, type A and B act the same way, but for setpoint changes, the difference is dramatic.

The illustrations show screen copies printed out from simulations done on a new software program called ProcessPlus (Gerry Engineering Software).

In the first illustration, with the Honeywell type A algorithm, damping and overshoot are similar on both setpoints and load disturbances. Thus, tuning is similar for load or setpoint changes with type A. Because of its sensitive setpoint response, you may want to use type A for the inner loop or slave in a cascade. On the other hand, Type B, with its smooth setpoint 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 B. Bailey's "error input" has sensitive setpoint response while Bailey's "PV and SP" has smooth setpoint response.

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, ProcessPlus allows

comparison of robustness plots of the competitors' algorithms.

Differences in integral action

Once integral and reset values are converted to the same units, PI controllers all respond in very much the same way to load disturbances. The proportional action may be different as described above. Anti-reset windup is often implemented differently, but the effect of these differences is usually minor compared to other differences between the algorithms.

Differences in derivative action

The largest variation among controllers from different manufacturers is found in the way they handle derivative action. Virtually no two are alike. This is part of the reason that most people avoid using derivative action. The differences are due to different methods of error signal filtering (or no filtering at all), whether or not the derivative action works on setpoint changes, and how derivative interacts or does not interact with the integral action.

On controllers, you get derivative action when you set the derivative or rate adjustment to anything except zero. In an interacting controller, integral and derivative actions interact with each other. This interaction can cause the effective controller action to be very different from what it would be in a noninteracting controller.

To understand how the interaction works, consider the following equations. The effective noninteracting proportional band in an interacting PID loop controller is as follows:

$$PB = \frac{PB}{1 + D/I}$$

Similarly, the effective integral time in an interacting controller is:

$$I = I + D$$

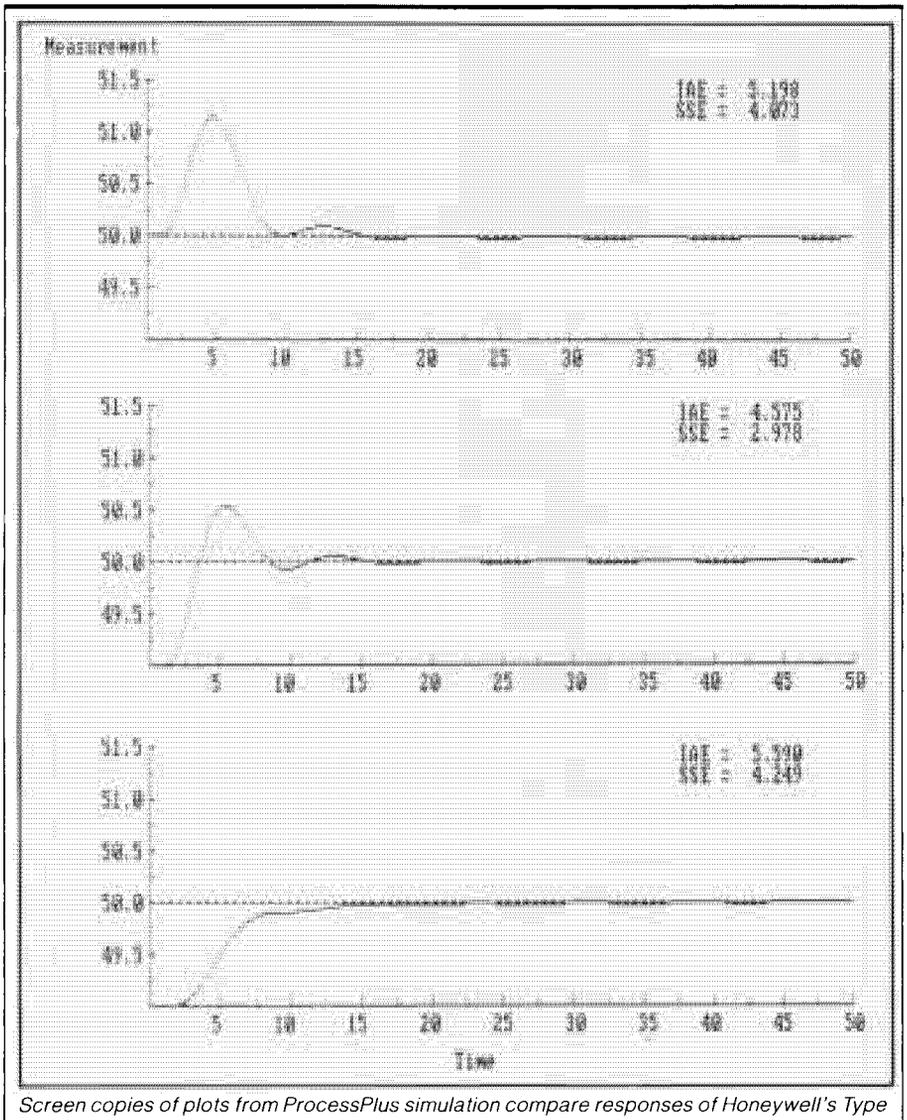
and the effective derivative time in an interacting controller is:

$$D = \frac{I}{\frac{1}{I} + \frac{1}{D}}$$

where *PB*, *I*, and *D* are the proportional band, integral and derivative values you set or enter into the interacting controller. The effective values are the equivalent settings for the noninteracting controller.

These equations show that the effective derivative time for the interacting controller cannot be made 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 an interacting controller.

Foxboro and Fisher use a noninteracting algorithm. Honeywell, Gould and Texas Instruments controllers use the interacting type.



Screen copies of plots from ProcessPlus simulation compare responses of Honeywell's Type A and Type B digital controller algorithms on a simulated temperature control loop. Note that in Type A, proportional action works on the setpoint input, while in Type B it works on the measurement signal only. The top plot shows the response of either type controller to a step load change on the simulated process. The middle plot is the response of the Type A controller, and the bottom plot of the Type B controller, to a step change in the setpoint.

Other differences in derivative

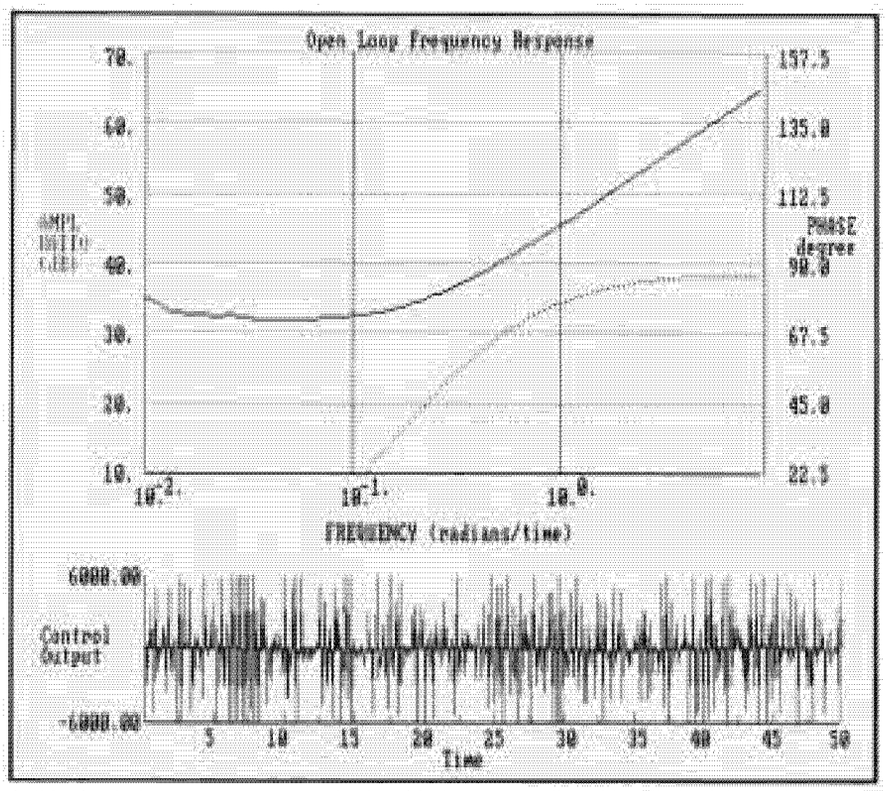
Beside the interaction differences described above, derivative action itself varies among the interacting and noninteracting groups of different manufacturers.

With most controllers, derivative action works only on the measurement signal. On some controllers, however, derivative action works on setpoint changes. Although response to a load disturbance will be the same, setpoint response on these controllers can get out of hand. Since most controllers are used for regulating disturbances, derivative action that works on setpoint changes is usually not a problem except in cascade loops or other cases in which the setpoint is being manipulated.

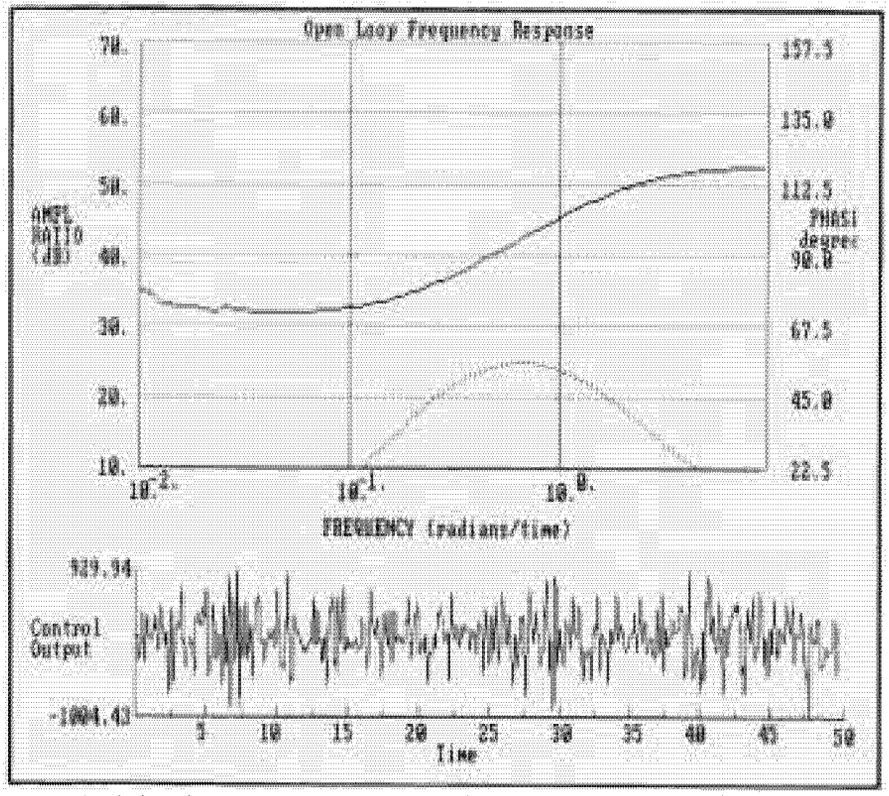
Of more significance is whether or not the signal is filtered before the derivative action is applied.

The unlimited derivative problem

Some manufacturers do not filter the signal to limit derivative action. Thus, at high frequencies, the amplitude ratio gets large. In the second illustration, a frequency response screen copy from ProcessPlus shows the



In this simulation, measurement noise is added to a controller with unlimited derivative action, that is, no signal filtering. ProcessPlus screen copies with zoom feature turned on show Bode plots of controller frequency and phase responses (top), and time response of the controller output (bottom). Note scale and wildness of controller output behavior without filtering.



In this simulation, the same measurement noise is added to a control loop, but this time with input signal filtering (limited derivative). The same zoom feature on ProcessPlus permits comparison of the effects of the filtering on the controller frequency and phase responses (top), and the reduction in controller output action (bottom). Note much reduced scale.

amplitude ratio of unlimited derivative action increasing with frequency.

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 second illustration is a screen copy of the time response of a controller to measurement noise. This can wear out valves, or drive a slave loop's setpoint crazy. Worse yet, the noise can drive the controller into saturation which causes the anti-reset circuitry or coding to take over. No wonder derivative is seldom used!

Filtering limits derivative noise

On the controllers that use filtering with derivative, the measurement signal usually gets the filtering. The time constant of filtering is usually calculated by these algorithms, based on the derivative value dialed in. Thus, the amount of filtering applied to the signal changes with the amount of derivative chosen. This has the effect of limiting derivative action at high frequencies.

In the third illustration, another plot from ProcessPlus shows the amplitude ratio of limited derivative action. This figure is the time response of a controller to the same measurement noise. Compare these responses with those of the illustration above. 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 multiplied by the error signal only. Integral and derivative actions are independent of the gain of the controller.

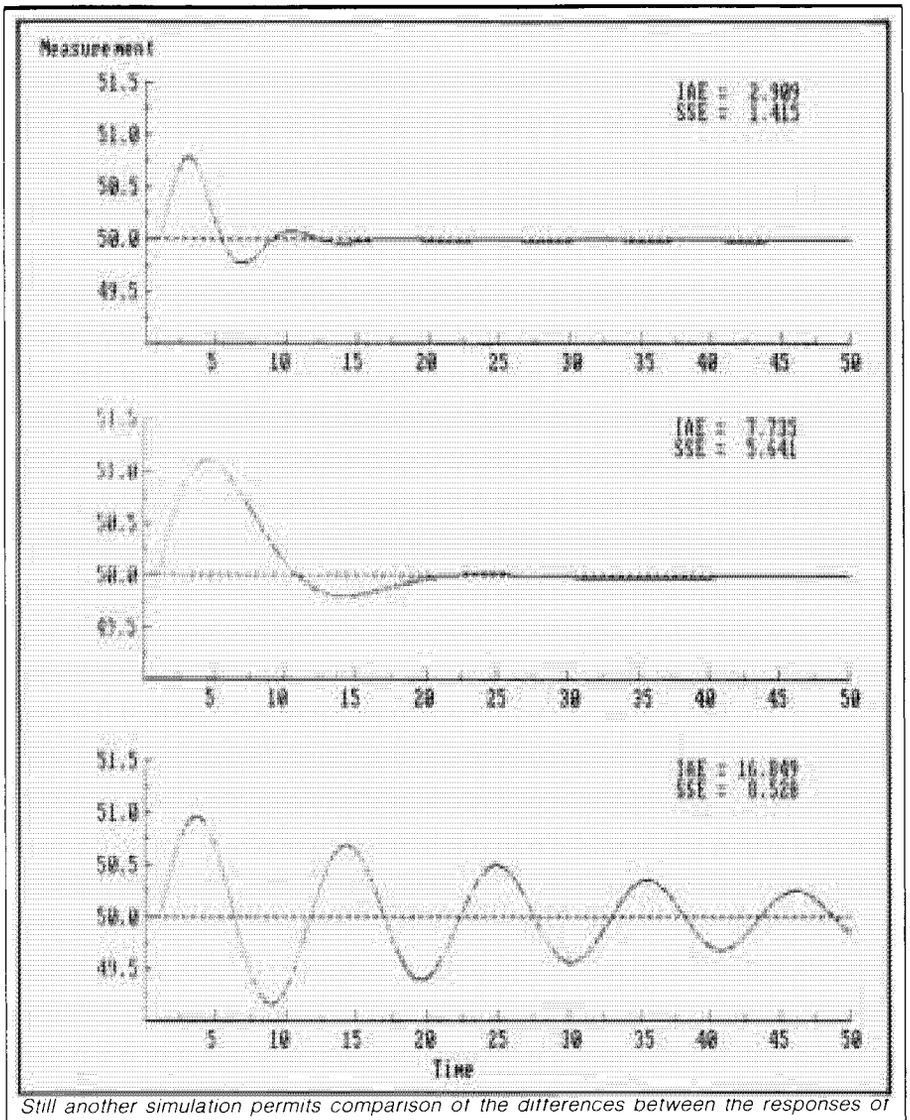
Parallel algorithms require very different integral and derivative tuning parameters than other controllers. The following equations show how to convert between parallel and noninteracting controller settings:

$$K_c = K_p$$

$$I = I_c K_p \text{ (units of time/repeat)}$$

$$D = D_c / K_p \text{ (units of time)}$$

There is more of a difference between parallel versus noninteracting controller tuning than between interacting versus noninteracting tuning. The intuitive feel for tuning a parallel controller is very different from that for other types. Look at the final figure. Normally it would seem that low-



Still another simulation permits comparison of the differences between the responses of noninteracting and parallel PI controller algorithms. The top curve represents the plotted response of either type controller, when carefully tuned, on a simulated flow loop. The middle plot shows the decreased but stable performance of the noninteracting controller when gain is reduced by a factor of three. The bottom plot for the same reduced gain with the parallel type controller tends to instability, much as if the gain were raised instead of lowered!

ering the controller gain should make the loop more stable, as in fact it does with the noninteracting controller in the figure. However, the parallel controller gets *less* stable with lower gain! Like all controllers, the parallel controller also gets less stable with higher gain.

So either increasing or decreasing the gain on a parallel controller can drive the loop unstable! The controllers in the the last illustration are PI controllers. The situation is even more pronounced when derivative action is used.

With the parallel controller, it is apparent that the effective integral and derivative values change with the gain setting. For example, lowering the controller gain also lowers the effective *I*, increasing controller phase, and decreasing the phase margin. Lowering the gain also increases the effective *D*, moving the derivative's phase contribution to higher frequencies and reducing its stabilizing effect on the controller's integral action. The overall effect is destabilizing, as is shown by the bottom plot in the final illustration at left.

Bailey and Allen-Bradley both have a parallel algorithm available that they describe as a "noninteracting" algorithm. They call their noninteracting algorithm an "interacting" one.

Choosing the best algorithm for any process depends on the specific process control needs and objectives. As demonstrated by these simulations, different algorithms perform better in different situations. So-called "parallel" controller algorithms are not necessarily noninteracting, and "noninteracting" controllers can in fact be interacting, if one goes by the disagreeing definitions of the different controller manufacturers. □

Our Own Measure of "ProcessPlus"

ProcessPlus requires an IBM-compatible PC, XT, or AT with a minimum memory of 320K. An unusual additional requirement is the presence of an 8087 numeric data processor—Intel's math coprocessor chip. Most math-intensive programs are designed to limp along without the coprocessor, but ProcessPlus just won't work at all.

ProcessPlus produces several types of on-screen time response and frequency response graphs designed for direct usefulness to control engineers and others familiar with feedback design techniques. Bode plots include a user-adjustable zoom feature that lets the user inspect critical areas more carefully. The program automatically makes use of IBM color and extended (EGA) graphics capabilities, and the color enhancements, though limited, are extremely useful.

The program includes simulations of seven much-used PID process controllers: Honeywell Type A and Type B noninteracting, Foxboro pre-1985 and post-1985 interacting, Fisher Controls interacting with limited derivative, a digital pole-cancellation controller (Dahlin type), and an ideal noninteracting algorithm. The process model is adjustable by inserting coefficients for gains, time constants (one zero, four first-order poles and a quadratic or cubic pole), and dead time. This model is a very flexible process simulator.

ProcessPlus is highly recommended as an easy-to-use professional tool that will let control engineers test closed-loop designs quickly, before hardware design is undertaken.

One quibble—your printer must accept IBM graphic codes to make screen copies like those shown. No other printers are supported. —*E. J. Kompass*

PID Tuning Without the Math

GEORGE BLICKLEY, CONTROL ENGINEERING

This primer explains loop performance without appealing to calculus or higher math functions.

After 40 years of practice in instrumentation, much of it at DuPont, David W. St. Clair decided to distill his experience into a short tutorial for those who don't have a background in frequency response analysis. Titled *Controller Tuning and Control Loop Performance*, the book is now available for those in the business of solving control problems.

The primer starts with a simplification of the Zeigler and Nichols closed-loop tuning method, namely, with reset and rate out of the controller, increase the gain until the loop maintains a small sustained cycle.

This cycle is the natural frequency, P_n, of the loop and is central to all concepts. For instance, the initial controller adjustments suggested are:

- Set the proportional gain to one-half of the gain that produced the natural frequency P_n;
- Set the reset rate to P_n;
- Set the derivative time to one-eighth of the natural frequency P_n.

The subsequent discussion leads to an important, yet simple, concept in control: P_n is approximately four times the apparent dead time. The author states, "Place permanently in your mind that P_n = 4L and you will cover a preponderant majority of industrial control loops."

Important points on lags

A discussion of lags, in general, first order lags in particular, integrators, and dead time—interspersed with some easy-to-follow curves all in the time domain—leads to the following:

- In real life, mathematically exact descriptions of lags are very complex, often beyond definition. Their effect in a control loop can be represented by a small set of relatively simple building block lags. While these lags are not exactly the same as the real system, they are close enough. Usually two, occasionally three, lags will adequately represent a real system;
- In normal loops, lags act in series; the output of one is input to another;
- The physical order of lags (transmission, component, or process lags) does not matter. The response to a step change is the same.

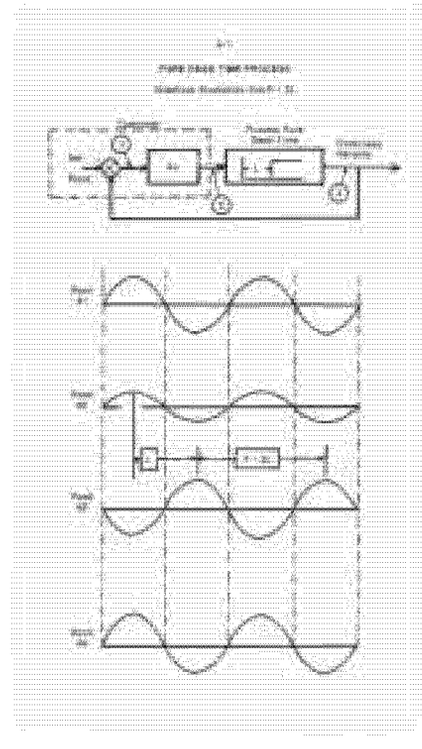
Pure dead time is used to explain the responses to a step input, but most processes do not have true dead time. The author proposes that all the little lags in the loop create the effective dead time, and thus the natural period. But this presents a paradox: the largest lag in a loop has little or nothing to do with the natural period.

Next is the dissection of the step response into simple lag building blocks. The author states, "You will not be able to get as many lag elements as there actually are. You should be able to satisfy yourself that there are lag elements which could combine to be the pure dead time, the effective dead time, the long first-order lag, etc., which were observed. Indeed, this reverse fit—trying to take a test result and determine if its components are reasonable—is a very useful trouble-shooting technique."

To help in the estimate of component and process lags, some helpful examples are given. Cascade control and loop interactions are also briefly discussed. A concise explanation is given to that least understood and most complicated action—derivative.

Controller Tuning and Control Loop Performance ($10, 37 pp, 8½ × 11; Straight-line Control Co., 3 Bridlebrook Lane, Newark, DE 19711) is for people who need a quick refresher or mini-course on PID. A promotional brochure with pricing is also available. The booklet should find its way to many labs, offices, and control rooms in the process industries, just as it did at DuPont, which released it to the author for publication. □

Author David St. Clair develops the concept of lags by using a pure dead time process, as shown in this example taken from his book. The output at point #2 of a proportional-only controller (K_c=½) is one-half the error signal at point #1. Point #3 is #2 delayed by L and multiplied by the process gain, shown here as 2. P_n is equal to twice the dead time.



Find Out How Good That PID Tuning Really Is

JOHN P. GERRY, P.E., GES, Lockport, IL

Tuning a loop manually is often an art. Once a loop appears to be tuned, questions remain: How sensitive will the tuning be to process changes? Could the tuning be better? How good is the tuning really?

By using ProcessPlus, a software package that runs on an IBM PC, a user can try those "what if" questions very quickly. One can use several powerful analysis techniques without ever having an operator scowl. This article will discuss some of these available techniques, focusing in on the robustness plot. This plot is a state-of-the-art tool allowing the user to analyze loop stability in a simple way not readily available without ProcessPlus software. It quickly reveals how sensitive a loop is to any combination of process gain or process dead time changes.

A process model is easy to get

A user could try and determine the plant dynamics from equipment sizing or design data. These calculations can be quite tedious and frequently result in estimated plant dynamics that are far from reality.

A plant test is simpler and more accurate. Loop response to one of several possible tests is shown (Figure 1). Some general characteristics of the response curve are entered into ProcessPlus which determines the process from this data.

Model determines tuning

From the plant model, ProcessPlus calculates PI and PID tuning for:

- Optimal response;
- Quarter amplitude damping; and
- ten percent overshoot.

Calculated tuning is specific for the process model and the industrial PID controller used in our loop.

Are we done now? Why not just enter one of these settings into our controller and call it a day? Because we might feel a little nervous about what happens to our loop when the process gain or dead time changes slightly. We don't know how good the tuning is. Looking at simulated time response and robustness plots quickly answers these questions.

Look at how our simulated control loop responded to an upset in Figures 2, 3, and 4. Each response curve is with one of the three different categories of PID tuning.

The solid line in the upper time plot is how the measurement (temperature, flow, pressure, etc.) responds to an upset. The dash line is the setpoint. The line in the lower plot is the controller output response. Noise was added to better approximate real process conditions.

The integrated absolute error (IAE) and sum of the squared error (SSE) is printed in the upper right portion of the time plots. Some people like to use these numbers as performance criteria for time response. The smaller the number, the better the response and tuning. In some loops, the integrated absolute error is proportional to the amount of energy dollars spent to recover from a process upset. Hence, a smaller IAE can equal greater savings.

ProcessPlus has calculated tuning for minimum IAE. Figure 2 shows how the loop responds to this tuning. Figure 3 shows the loop tuned for "quarter amplitude damping", a very popular response. This tuning has a slightly higher IAE. Finally, Figure 4 shows a loop tuned for ten percent overshoot. This tuning has the least desirable IAE.

Tuning for minimum integrated absolute error seems the fastest from the look of the response and the IAE and SSE values. It also seems to have the least overshoot. Now this has settled it. Just use the tuning for minimum

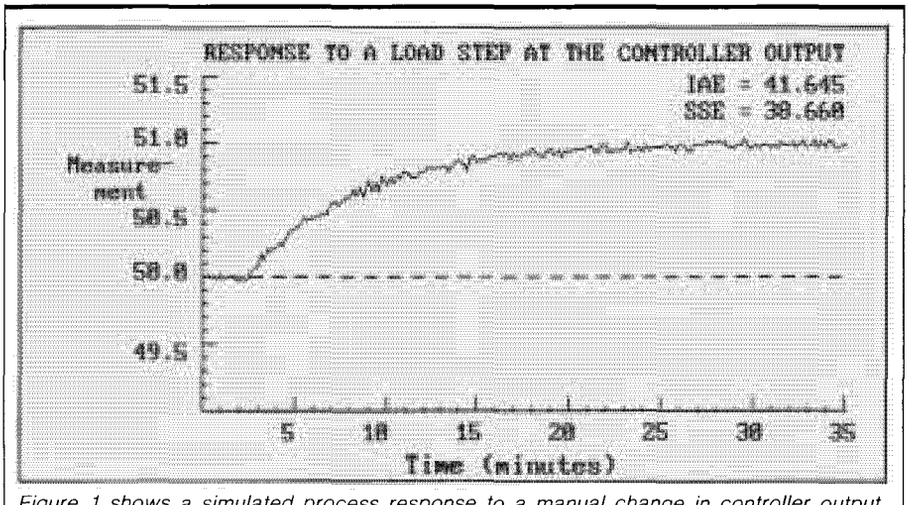


Figure 1 shows a simulated process response to a manual change in controller output.

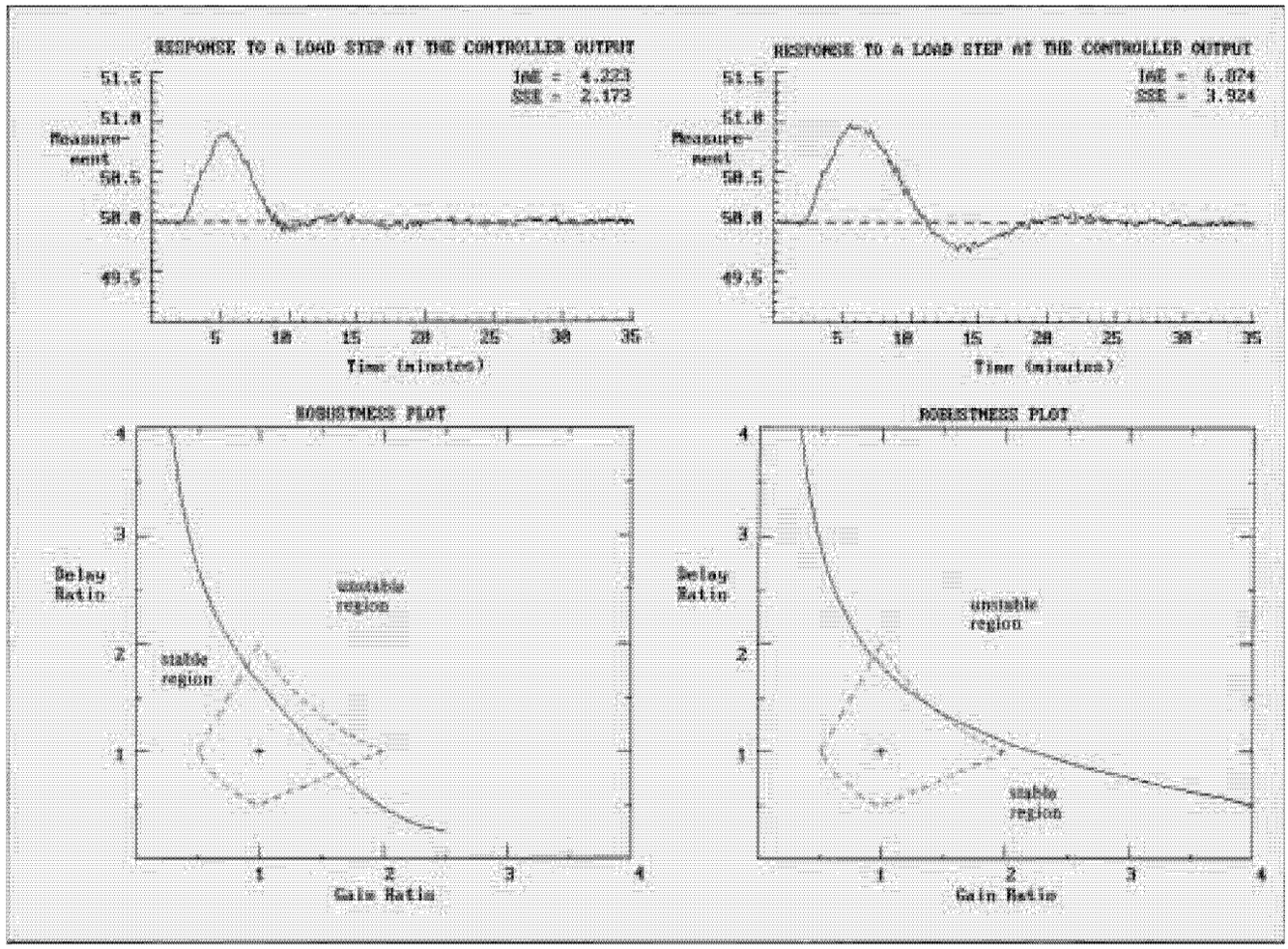


Figure 2 shows simulated measurement response (noise added) to a load disturbance and the robustness plot for the loop. The controller has been tuned for MIAE (minimum integrated absolute error).

Figure 3 ProcessPlus screen shows simulated measurement response (noise added) to a load disturbance and the robustness plot for the loop. The controller is tuned for quarter amplitude damping.

IAE, right? Well, maybe.

Most loops are somewhat nonlinear where the loop gain and dead time change over the loop's operating range or with different loads. How effective are these three categories of tuning when the process gain or dead time change?

What is robustness?

The robustness plot is a powerful analysis tool. A glance at this plot will inform the user if there is a need to adjust settings for more or less safety factor. It quickly shows how sensitive (or robust) the loop is to any combination of process gain and process dead time changes. The two axes of the plot are the delay ratio and the gain ratio. The delay ratio is calculated by dividing process dead time by the process dead time that the controller was tuned for. The gain ratio is calculated by dividing the process gain by the process gain that the controller was tuned for.

The plot has a region of stability and a region of instability separated by a solid line: the limit of stability line. To the right and above the solid line (higher gain and delay ratios) the control loop is unstable. To the left and below the solid line, the control loop is stable. At the cross, where both ratios are one, the process gain and dead time are at the process values tuned for.

Lets see how this plot works by looking at an example. Look at the robustness plot for the loop tuned for minimum IAE, integrated absolute error, in Figure 2. Starting at the cross, look straight up to the line. Here the dead time ratio read off the vertical axis is about 1.6. This means that if the dead time increases by a factor of about 1.6, the loop being controlled will be unstable.

Now start at the cross and visually trace right to the solid line. The gain ratio at this point is about 1.4. This means that if the controller or process gain increases by a factor of 1.4, the loop will become unstable. If the gain and dead time increase by a factor of about 1.25 (move diagonally from the cross), the loop being controlled will be unstable.

What about the dash line box?

A safety factor of two is generally considered "reasonable" for a well known loop. The vertices of the weird box are a factor or divisor of two on the gain or delay ratio. These vertices are connected by lines that represent a combination changing gain and dead time that give a factor of two. So the dash line figure is a design aid much like a grid on another type of plot. Generally the solid limit of the stability line should be outside the dash line box for the control loop to be "reasonably" stable.

A user may want more or less of a safety margin in his loop depending on a "feel" for how much the loop gain and dead time will change.

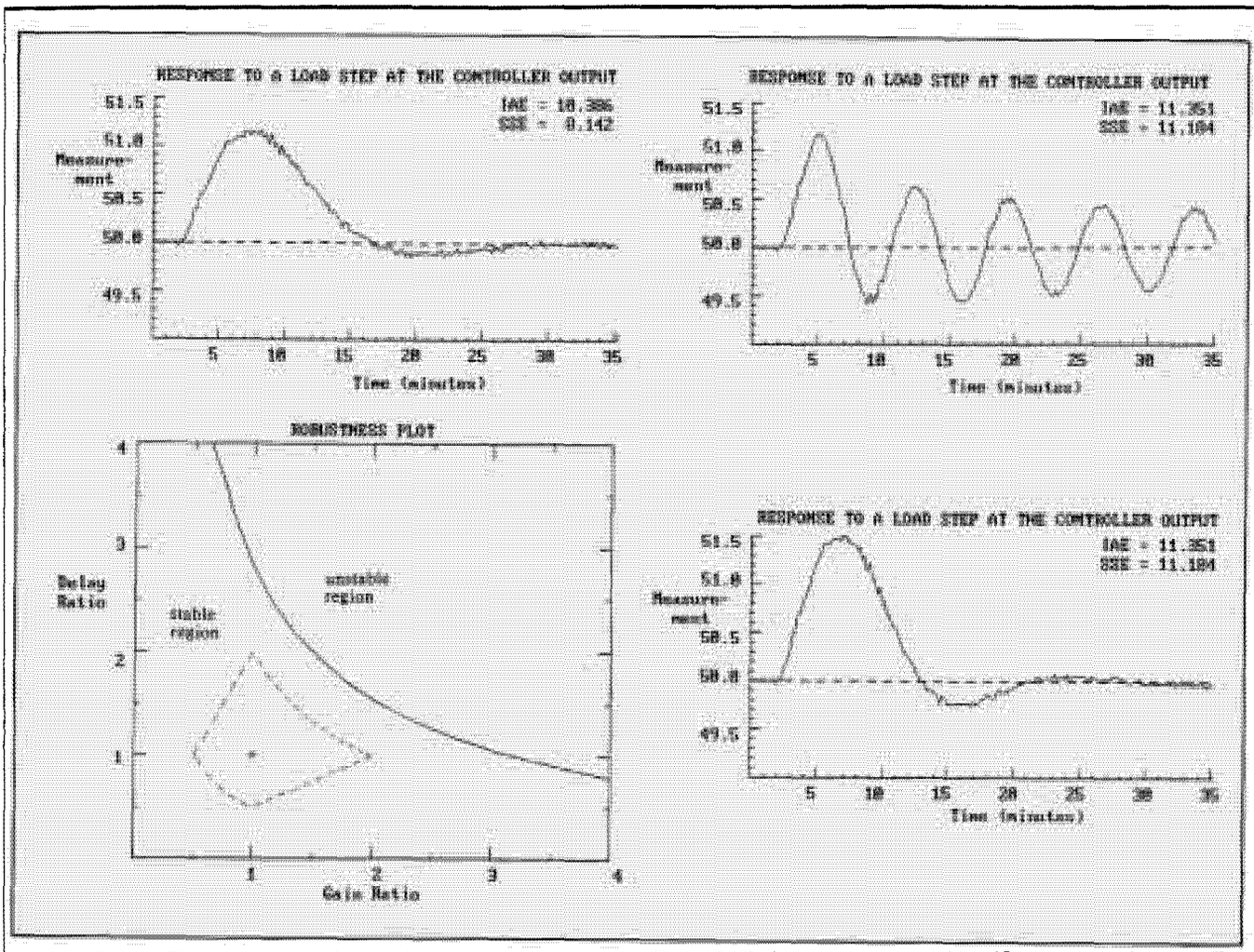


Figure 4 ProcessPlus screen show simulated measurement response (with noise added) to a load disturbance and the robustness plot for the loop. The controller is tuned for ten percent overshoot.

Figure 5 (Top) shows response with a 40 percent gain increase. Controller tuned for MIAE. Figure 6 (Bottom) shows response with 40 percent gain increase. Controller tuned for ten percent overshoot

Robustness plot examples

The loop tuned for minimum IAE is not very robust. That is, a relatively small change in process gain or dead time and our loop will be unstable. The loop tuned for ten percent overshoot is the least sensitive to changing process conditions.

Looking at the robustness plots, we can put the time plots in perspective. We always knew there was a trade-off between tight tuning and stability—now robustness plots quantify this.

Still in control with more gain?

Do robustness plots really work? To test this using ProcessPlus, we simply change the process to see the effect on our control loop. Lets try changing the process gain and see the effect.

A 40 percent increase in process gain doesn't seem like much, right? Take a look at Figure 5. This loop had the minimum IAE tuning. We increased the process gain by 40 per cent and tried the same upset used in Figure 2. Compare this response to the response shown in Figure 6. In Figure 6, this loop has a ten percent overshoot tuning.

With the gain increase, the loop tuned for minimum IAE is almost unstable. This agrees with what the robustness plot of Figure 2 indicates. In Figure 2 moving to the right of the cross, we intersect the limit of stability line at about 1.4.

The process gain increase has less of an effect on the loop tuned for ten percent overshoot. Again, the robustness plot for this loop (Figure 4) indicates this. In Figure 4, the limit of stability line is about a factor of 3 away from the cross. The loop gain, therefore, would have to increase by much more than 40 percent to result in any instability. As the robustness plots have shown us, using the ten percent overshoot tuning results in less sensitivity to changing conditions than the tighter tuned loop.

Which tuning is the best?

There is a trade-off between tight tuning and robustness. The tighter the loop is tuned, the less robust and more sensitive it becomes. Using ProcessPlus, a engineer or technician can quickly analyze the trade-off between tight tuning and the resulting robustness for any industrial loop.

The best tuning to use in a plant loop depends on the knowledge and confidence the user has about that plants dynamics, linearities and changing conditions. If the loop is very linear, very well understood and doesn't change, use the tightest time response tuning (minimum integrated absolute error). Most plant loops will require tuning that is even more robust than the ten percent overshoot tuning example.

Acknowledgement

The author would like to acknowledge Dr. P.D.Hansen who invented robustness plots. □