# 3
# Basic Feedback Control

### Learning Objectives

A. **Be able to select the best PID structure and form for an application.**

B. **Know how to tune PID controllers using a unified approach.**

C. **Recognize how diverse tuning methods can be reduced to a common form for tight control.**

D. **Discover how to reduce the process test time by an order of magnitude.**

E. **Recognize how to prevent fast oscillations caused by aggressive tuning and slow cycling and offsets caused by sluggish tuning.**

F. **Understand the implications of an integrating response on PID tuning.**

G. **See how an adaptive control can provide process knowledge in addition to automatically identifying tuning settings.**

H. **Find out how to significantly speed up the set point response to reduce batch time.**

## 3-1. Introduction

Adding feedback measurement is essential for important process outputs because of the uncertainties and variability both in the process inputs and within the process. Using measurement in a feedback control loop offers automatic compensation. Since the proportional-integral-derivative (PID) controller is the predominant controller used in industry for basic feedback control, this chapter focuses on how to set up, tune, and optimize the PID.

Chapter 4 discusses how a control loop's ultimate performance depends on a process model but that the actual performance is determined by the PID controller's tuning settings. This chapter details the procedure for computing the PID tuning settings from the parameters of a process model. Though a dynamic model is implied in the tuning, as long as the PID controller is stable, it corrects for unknowns, loads, and disturbances.

In general, the user backs off from the "hottest" tuning settings for the "tightest" control in order to reduce the potential for oscillations in the process variable or manipulated variables. A trade-off must always be made between performance and robustness. High controller gains transfer more variability from the process variable to the manipulated variable. Fortunately, column, reactor, and vessel loops have fewer interactions than do many other unit operations, so variability in the manipulated variable is less disruptive. However, fluctuations in the controller output as a result of process or measurement noise that exceeds the final element's resolution can disturb the loop. Aggressive control makes the loop more vulnerable to oscillations from the inevitable changes in the process dynamics. Also, operators tend to dislike the large kicks in the controller output that can occur from set point changes to controllers with a high gain even though these kicks may in fact be beneficial in terms of a creating a faster response.

This chapter shows how process dynamics permit controller gains for the primary loops that are higher than those users have used on loops in other unit operations. In practice the controller gains actually used are far below the optimum for tight control because of the user's comfort zone combined with concerns about robustness and amplification of noise.

The oscillations caused by hot controller tuning and noise amplification are relatively fast and better understood than are the slower oscillations from less obvious causes. This chapter details how sluggish tuning of secondary controllers causes cycling in a cascade control system, how low controller gains cause reset cycling, and how sluggish tuning settings aggravate the observed limit cycling from the final element's resolution limit.

PID controllers are commonly used in temperature control. Before tuning tests are done and tuning decisions made, users need to select the proper structure and form of the PID algorithm. The effect that any set of tuning settings will have on loop performance depends on the PID's structure and form. A modern distributed control system (DCS) offers two or more forms and eight or more structures. It's important for users to understand the functional capability of each choice in order to standardize on the best algorithm for the application. When migrating projects or relocating settings from laboratories or pilot plants to production units, it is important to know the effects of different controller algorithms and the tuning parameter units. This knowledge enables users to reuse what they learned from tuning previous systems or similar applications in current systems.

Loop tuning and analysis tools also need to take into account tuning setting units, PID structure and form, and the limitations on the testing of production systems. The test times and step sizes are often beyond what is acceptable to plant operations. This chapter discusses how test time can be reduced by an order of magnitude for the primary loops, considerations for using the tuning settings from bench-top units and pilot plants, how to evaluate batch set point responses, and tuning methods that can be extended to provide optimal estimates of the tuning settings for industrial production. Ultimately, the user needs to understand the functional contribution of each mode in order to verify, analyze, and improve controller tuning and performance.

Section 3-2 describes the relative importance of various PID forms and structures; issues to consider when moving tuning settings between bench tops, pilot plants, and production units; and the important advantages offered by external feedback. Section 3-3 develops a unified approach for tuning controllers, discusses how tuning methods can be reduced to a common form, and details the impact of process dynamics on primary controller tuning.

Section 3-4 provides an overview of an adaptive controller that identifies the process model from normal set point or output changes and eliminates the need for manual or programmed test sequences. The adaptive controller computes settings based on the users' preference of robustness versus performance and users' concerns over the amplification of noise. The tuning settings identified from past batches can be used to schedule (predict) settings as a function of controller output or any process or computed variable such as batch time. Limits can be placed as necessary on the range of settings. Whether these settings are used or not, the models identified provide process knowledge and diagnostics. Changes in the process model can alert operations staff to changes in column, reactor, and vessel behavior.

Section 3-5 compares conventional techniques for pre-positioning controller outputs and tuning controllers against a new innovative approach that considerably increases the speed of the primary loop's set point response through a simple calculation of the controller output's optimal switching time and final resting value.

## 3-2. PID Modes, Structure, and Form

Basic feedback control is performed by a controller that has proportional, integral, and derivative modes. Except for temperature loops, the derivative mode is usually turned off by setting the rate time (derivative time) setting to zero. Users make a distinction here and call a controller with no derivative action a PI controller, but the open literature often does not do this. In many papers, the performance of PI controllers (labeled as PID controllers) is frequently compared to advanced control algorithms. This section will discuss how heavily performance comparisons depend on PID structure. Equations 4-1 and 4-2 in Chapter 4 showed how strongly performance depends on tuning settings. The authors of the technical literature choose structures and methods to prove the value of their new tuning methods or algorithms. In cases where derivative control is useful and noise and interaction are negligible, an aggressively tuned PID controller offers the best rejection of unmeasured disturbances at the input to a process [1] [2]. Often overlooked are the special techniques that can be readily added to the PID controller, such as batch preload and dead-time compensation via external reset (mentioned in this section) and the optimal switching of PID output to its final resting value (detailed in Section 3-5).

### Proportional Mode Structure and Settings

The discrete contribution that the proportional mode makes to the controller output for the "standard" form of the PID algorithm is shown in Equation 3-1. The set point is multiplied by a $\beta$ factor that ranges between 0 and 1 and is used to provide a proportional kick to speed up the response to a set point change. The kick is a step change in output. For slow loops, the step drives the output beyond its final resting value. Without this kick, the PID controller output relies on the integral action, which provides a slow approach to the set point by way of a slow reset time that is set to match the slow process response. The benefit the kick provides is greatest for loops in which the process time constant or integrating process gain is much slower than the process dead time. Thus, the kick provides minimal benefit to a flow loop since the process time constant is so fast. However, if the flow loop has a cascade set point or remote cascade set point, the kick can be used to provide a more immediate response to the demands of a primary controller or a batch sequence, as discussed in Sections 3-3 and 3-5, for cascade and batch control, respectively. However, whenever the operator changes a set point, this kicks the output. For a primary loop or a single loop, a set point rate limit or filter can be added to smooth out the kick from an operator set point or the β factor can be set to 0. It's not advisable to use set point rate limits and filters for secondary loops because they degrade the primary loop's performance. They have the same effect on the controller output as a velocity limit or filter. Although the β factor in a secondary loop degrades a primary loop's ability to reject disturbances, it has no deteriorating effect for disturbances that originate within a secondary or single loop.

***A structure of "PI action on error," which sets the β factor = 1.0, provides the fastest set point response. This is important for secondary loops and the batch sequences.***

In the equations for the PID controller used in this book, we use the term *controlled variable* (CV) in place of *process variable* (PV) both to denote that the units are percent of scale range instead of engineering units of the process variable and to make the nomenclature for the PID and model predictive controller (MPC) more consistent. It is important to remember that the configuration, displays, trend charts, and documentation of most modern control systems use process variable (PV), feedforward variables, and controller outputs in engineering units. When computing the process gain, users must take this into account because in a distributed control system (DCS) the PID algorithm is based on feedback and feedforward inputs and an output in percent of the respective scales. Special-purpose and user-constructed PIDs, as well as a few programmable logic controllers (PLC), use engineering units in the algorithm. This severely reduces the portability of tuning settings and the users' understanding of the effects of the process. Section 3-3 on PID tuning methods shows that the change in engineering units cancels itself out for PID algorithms that use percent for the controller inputs and outputs. The specification of engineering units for controller output in modern DCS and fieldbus systems does not mean these engineering units are used in the PID algorithm. The conversion of controller output from percent to engineering units is done after the PID algorithm.

***The use of percent for inputs and outputs in the PID control algorithm increases the portability and comprehensibility of controller gain settings.***

$$P_n = K_c * (\beta * SP_n - CV_n) \qquad (3\text{-}1)$$

Though most digital controllers use controller gain, proportional band (PB) was once a prevalent tuning setting for the proportional mode in analog controllers. Proportional band was devised to be the percent change in the controlled variable that is needed to cause a 100 percent change in the controller output [2] [3]. Some digital controllers still use it. It is critical that the user know whether the proportional mode tuning setting is a gain or proportional band because this can have a huge effect as indicated by the inverse relationship shown in Equation 3-2.

***It is critical to know whether the proportional mode tuning factor is a proportional band in percent (%) or a dimensionless controller gain.***

$$PB = 100\% \,/\, K_c \qquad (3\text{-}2)$$

**Integral Mode Structure and Settings**

The discrete contribution of the integral mode to the controller output for the "standard" form of the PID algorithm is shown in Equation 3-3. The controller gain setting divided by the integral (reset) time setting is multiplied by the error between the set point and the controlled variable for the current execution ($n$). This result is multiplied by the integration step size ($\Delta t$), which is the module execution time, and added to the integral mode result for the last execution ($n$–1). The result is an integration of the error factored by the ratio of the controller gain to reset time setting. The reset time is the time required for the integral mode to repeat the contribution from the proportional mode. In the ISA standard algorithm, the reset time setting is in seconds per repeat and is simply called "reset." Often the units are stated as just seconds. Some manufacturers use minutes per repeat, or its inverse (repeats per minute), with no change in the use of the term reset. It is imperative that the user know the units of the reset tuning setting.

***It is critical to know whether the integral mode tuning factor is a reset time (seconds or minutes per repeat) or its inverse (repeats per second or minute).***

$$I_n = (K_c / T_i) * (SP_n - CV_n) * \Delta t + I_{n-1} \qquad (3\text{-}3)$$

**Derivative Mode Structure and Settings**

The discrete contribution of the derivative mode to the controller output for the "standard" form of the PID algorithm is shown in Equation 3-4. The set point is multiplied by a $\gamma$ factor that ranges between 0 and 1 and can be used like the $\beta$ factor to provide a proportional kick to speed up the response to a set point change. In this case, the kick is a spike or bump, whereas the kick from the $\beta$ factor is a step. The effect is short term, and the burden is still on the integral mode to change the output enough to accelerate the process variable. Although this jump in the controller output can help the signal get through the dead band or resolution limit of a control valve, the better solution is to reduce backlash and sticktion by using a more precise throttle valve and digital positioner. The $\gamma$ factor is normally set to zero to prevent the derivative mode from overreacting to operator-entered set points. A set point filter or velocity limit can be used just as it was to reduce the kick from the $\beta$ factor for single and primary loops. The $\gamma$ factor, like the $\beta$ factor, does not affect a loop's ability to reject a load.

***The $\gamma$ and $\beta$ factor do not affect the load response of a control loop.***

Since the derivative mode provides a contribution that is proportional to the slope of the change, step changes and noise are disruptive. Most controllers have an inherent filter whose time constant is a fraction $\alpha$ of the derivative (rate) time setting; this ensures that the spike in the output becomes a bump. A typical value for $\alpha$ is 1/8 to 1/10. Note in the Equation 3-4 that for $\alpha$ and $\gamma$ factors of zero, the numerator can be simplified to just the change in controlled variable multiplied by the controller gain and rate time. The denominator can be simplified to just the execution time. For this case, it is easier to see that the derivative mode provides a contribution that is proportional to the rate of change of the controlled variable ($[CV_n - CV_{n-1}] / \Delta t$).

***It is critical to know whether the time units of the derivative mode tuning factor (rate time) is seconds or minutes.***

$$D_n = \frac{(K_c * T_d) * [\gamma * (SP_n - SP_{n-1}) - (CV_n - CV_{n-1})] + \alpha * T_d * D_{n-1}}{\alpha * T_d + \Delta t} \quad (3\text{-}4)$$

**Summary of Structures**

The β and γ factors are sometimes called *set point weighting factors* and are usually found under the category or term structure in the *controller configuration*. A controller in which both factors are adjustable is called a two-degree freedom controller. Other structures have the β and γ factors set equal to 0 or 1. The user can also omit a mode entirely to get P-only, I-only, ID, and PD control with various assigned factors. PI control is achieved by simply setting the derivative (rate) time to zero. In general, the user must not set the controller gain equal to zero in an attempt to get I-only or ID control. Nor should the user set the integral (reset) time to zero in an attempt to get P-only or PD control. Note that using P-only or PD control requires that additional choices be made about how to set the bias and its ramp time. Table 3-1 lists eight choices offered by one major DCS supplier.

**Table 3-1. List of major pid structure choices**

1. PID action on error (β = 1 and γ = 1)
2. PI action on error, D action on PV (β = 1 and γ = 0)
3. I action on error, PD action on PV (β = 0 and γ = 0)
4. PD action on error (β = 1 and γ = 1)
5. P action on error, D action on PV (β = 1 and γ = 0)
6. ID action on error (γ = 1)
7. I action on error, D action on PV (γ = 0)
8. Two degrees of freedom controller (β and γ adjustable 0 to 1)

***Normally a structure of "D action on PV" ($\gamma = 0$) is used to prevent spikes or bumps from set point changes but "D action on error" ($\gamma > 0$) can be a temporary fix to minimize the effect of valve backlash and sticktion for set point changes.***

## Algorithms

In the "standard" form of the PID used in most DCS applications, the contributions of the three modes are added to the initial value of the controller output ($CO_i$) set to provide a bumpless transfer to the execution of the PID algorithm. This initialization of the controller output occurs for transitions from modes in which the output is manually set, the output is tracking an external variable for cascade or override control, or the output is remotely set by batch sequence. Execution of the PID algorithm occurs for the AUTO (automatic), CAS (cascade), and RCAS (remote cascade) modes [2].

Equation 3-1 through 3-5 are a simplified representation of the "positional" algorithm that is predominantly used in industrial control systems [3]. An "incremental" or "velocity" algorithm used by special-purpose supervisory computers developed in the 1970s would compute the change in controller output from each mode for each execution, and add it to the full controller output from the last execution of the PID algorithm. The "incremental" algorithm inherently eliminates bumps, windup, and synchronization considerations. For supervisory computers, the last output was read back as the current set point so the recovery from a failure of execution or the communication link was smooth. However, these and other concerns are now addressed by the use of the back-calculate (BKCAL) feature and initialization calculations of Fieldbus functional blocks. Each control system supplier also has methods for preventing reset windup and coming off of the controller output limits for the "positional" algorithm. The "positional" offers the advantage of using external feedback for improved override control through more effective transitions between the selections of controller outputs. The "positional" algorithm also offers proportional-only (P-only) and proportional-plus-derivative (PD) control and a fixed bias. The "incremental" algorithm has inherent integral action and no fixed bias. Since the incremental algorithm is not recommended, this book will focus on the equations and function of the positional algorithm.

***The "standard" form of a PID positional algorithm is the most common type used in a DCS and offers P-only and PD control with an adjustable bias.***

$$CO_n = P_n + I_n + D_n + CO_i \tag{3-5}$$

where:

| | | |
|---|---|---|
| $CO_i$ | = | controller output at transition to AUTO, CAS, or RCAS modes (%) |
| $CO_n$ | = | controller output at execution n (%) |
| $CV_n$ | = | controlled variable at execution n (%) |
| $D_n$ | = | contribution from derivative mode for execution n (%) |
| $I_n$ | = | contribution from integral mode for execution n (%) |
| $K_c$ | = | controller gain (dimensionless) |
| $P_n$ | = | contribution from proportional mode for execution n (%) |
| $PB$ | = | controller proportional band (%) |
| $SP_n$ | = | set point at execution n (%) |
| $T_d$ | = | derivative (rate) time setting (seconds) |
| $T_i$ | = | integral (reset) time setting (seconds) |
| $\alpha$ | = | rate time factor to set derivative filter time constant (1/8 to 1/10) |
| $\beta$ | = | set point weight for proportional mode (0 to 1) |
| $\gamma$ | = | set point weight for derivative mode (0 to 1) |

Figure 3-1 shows the combined response of the PID controller modes to a step change in the set point ($\Delta SP$). The proportional mode provides a step change in the controller output ($\Delta CO_1$). If there is no further change in the SP or the CV does not respond, no additional change in the output occurs even though there is a persistent error (offset). The size of the offset is inversely proportional to the controller gain. Integral action ramps the output unless the error is zero. Since the error is rarely exactly zero, the reset is always driving the output. Even if there are no disturbances, reset action causes a continuous equal amplitude oscillation (limit cycle) as it moves the PID output and the process variable through the resolution limits of the final element and measurement, respectively. The probability of a controlled variable coming to rest exactly at set point is next to zero because of the quantization of the process input by the final element resolution and the process output by the measurement resolution. Thus, a PV

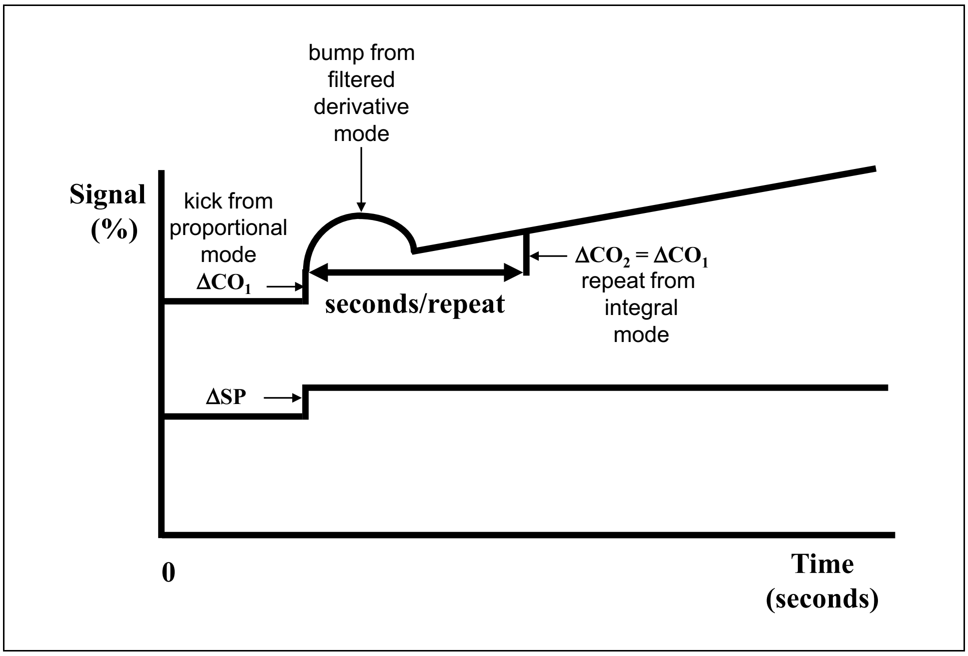

bump from filtered derivative mode
Signal (%)
kick from proportional mode $\Delta CO_1$
seconds/repeat
$\Delta CO_2 = \Delta CO_1$ repeat from integral mode
$\Delta SP$
0
Time (seconds)

**Figure 3-1. Contribution of each pid mode for a step change in the set point (β=1 and γ=1)**

nearly always passes through the set point and never stays at a resting value for a controller that has integral action [2].

***The integral mode continually works to eliminate an offset and is always moving the PID output since the error is rarely exactly zero.***

The contribution made by the integral mode equals the contribution made by the proportional mode ($\Delta CO_1 = \Delta CO_2$) after the integral time. Hence, the integral (reset) time is the time (seconds) it takes to repeat the contribution of the proportional mode, which is the basis of the units "seconds per repeat," for a step change in the SP or CV [2] [3].

The contribution made by the derivative mode for the same step change is a bump rather than a spike because of the built-in derivative filter. If there is no further change in the SP and the CV does not respond, the contribution from the derivative mode goes to zero [2] [3]. Like the proportional mode, the derivative mode does not attempt to eliminate an offset. Consequently, proportional-plus-derivative (PD) controllers require that a bias be properly adjusted to minimize the offset, particularly for low controller gains.

***Proportional-plus-derivative (PD) controllers with low controller gain settings require that attention be paid to the proper setting of the bias to ensure that the offset is acceptable.***

Before the advent of the DCS, most industrial control systems used the "series" PID algorithm in which the derivative mode was computed first as the input to the proportional and derivative modes, as shown in Figure 3-2. The computation of the derivative mode in series with other modes was the most practical method for implementing derivative action in analog controllers and was known as the "real" algorithm. In analog controllers, the derivative mode with its built-in filter was actually a lead-lag in which the lead time was the derivative or rate time setting ($T_d$) and the lag time was the filter time ($\alpha * T_d$). In these analog controllers, the derivative action was on PV instead of error ($\gamma = 0$).

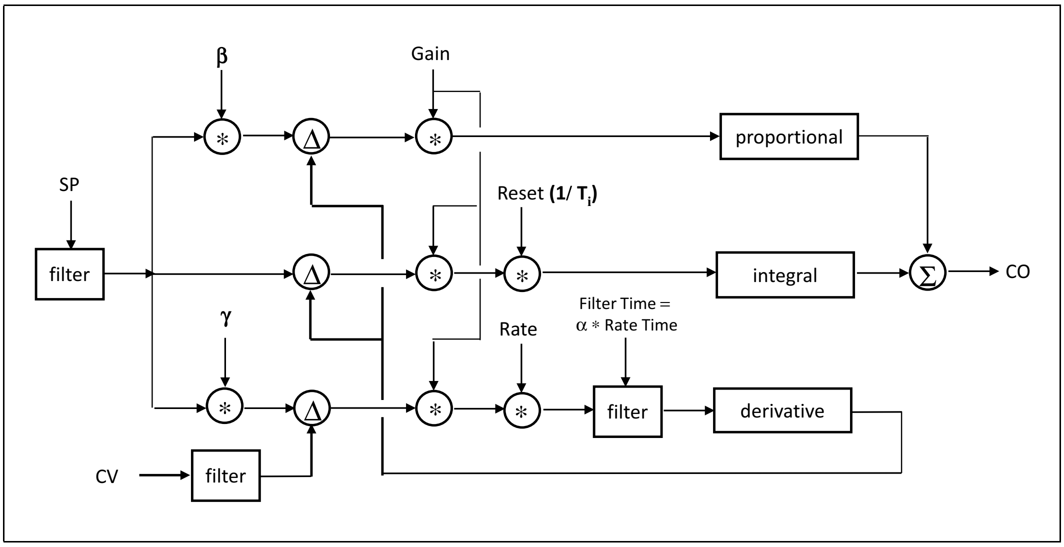


**Figure 3-2. Block diagram of "series," "real," or "interacting" PID algorithm**

The "series" algorithm was also known as the "interacting" algorithm because the derivative and integral time settings had an interacting effect on the contribution of all modes, as defined in Equations 3-6 through 3-9. From Equation 3-9, the interaction factor would ensure that the ratio of the derivative time to integral time never exceeded ¼. In theory, a ratio of ½ is optimal, but the prevalence of noise makes this goal impractical. A derivative time effectively greater than the integral time setting causes instability. If the user made a mistake and set the derivative time greater than the integral time, the "series" or "interacting" algorithm prevented the effective ratio from exceeding ¼ [3].

The "standard" algorithm computes the derivative mode in parallel with the other modes, as shown in Figure 3-3. It is the form of the PID algorithm adopted by ISA as its standard. The "standard" or ISA algorithm is also known as the "ideal" or "noninteracting" algorithm and is the default choice in most twenty-first-century control systems. The settings on the left side of Equations 3-6 to 3-8 are the settings for the standard algorithm. Note that when the derivative time is zero, the "standard" and "series" algorithm are the same, the interaction factor is 1.0, and no conversion of settings via these equations is needed [3]. Most of the literature on controller tuning assumes a "standard" structure. Hence, the derivative time is often listed as ¼ the integral time for PID control. In Section 3-3 on controller tuning, we discuss how derivative action is mostly used on temperature loops; it should be set equal to the next largest time constant (second lag). For thermal systems, the second lag is about 1/10 of the largest time constant.

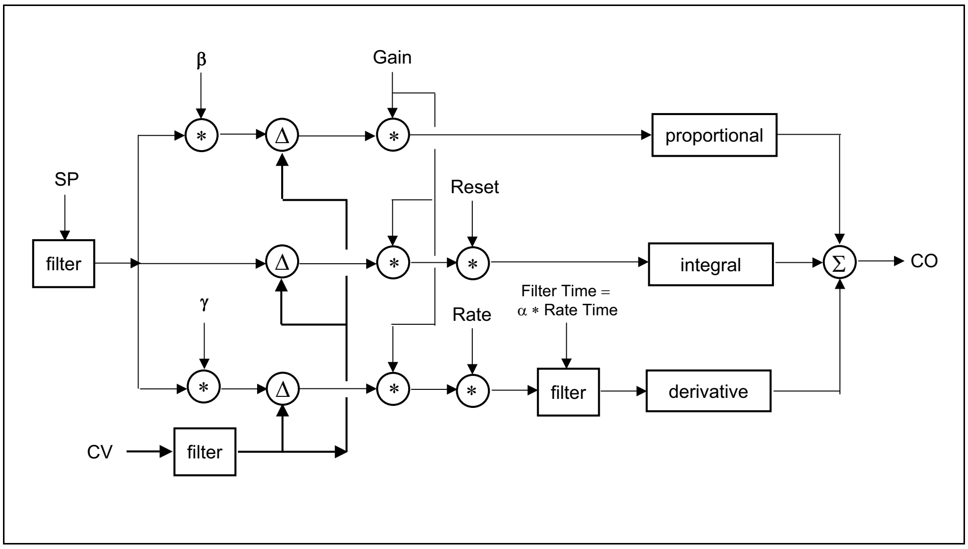


**Figure 3-3. Block diagram of "standard," "ideal," or "noninteracting" PID algorithm**

Much less common is the "parallel" algorithm, which also computes the modes in parallel. However, the controller gain setting only affects the contribution of the proportional mode. The controller gain is not in the input path to the derivative and integral modes. Only a few systems offer this as a choice. Although there is no performance incentive, some users may prefer that the tuning settings be completely independent. Other users are unappreciative of the drastically different "feel" and settings encountered when tuning a "parallel" controller. Also, the portability between the "parallel" and "standard" systems is much less than the portability between the "series" and "standard" systems. Unless the controller gain is close to 1.0, it is critical that Equations 3-10 through 3-12 be used to convert the settings between "parallel" into a "standard" controller when moving settings, regardless of whether derivative action is used.

As always, it is essential to take into account the different units of the tuning settings.

$$K_c = K_c' \;/\; I_f \tag{3-6}$$

$$T_i = T_i' \;/\; I_f \tag{3-7}$$

$$T_d = T_d' * I_f \tag{3-8}$$

$$I_f = T_i' \;/\; (\,T_i' + T_d'\,) \tag{3-9}$$

$$K_c = K_c'' \tag{3-10}$$

$$T_i = T_i'' * K_c'' \tag{3-11}$$

$$T_d = T_d'' \;/\; K_c'' \tag{3-12}$$

where:

| | | |
|---|---|---|
| $K_c$ | = | controller gain for "standard" algorithm (dimensionless) |
| $K_c'$ | = | controller gain for "series" algorithm (dimensionless) |
| $K_c''$ | = | controller gain for "parallel" algorithm (dimensionless) |
| $I_f$ | = | interaction factor (dimensionless number < 1.0) |
| $T_i$ | = | integral (reset) time for "standard" algorithm (sec/repeat) |
| $T_i'$ | = | integral (reset) time for "series" algorithm (sec/repeat) |
| $T_i''$ | = | integral (reset) time for "parallel" algorithm (sec/repeat) |
| $T_d$ | = | derivative (rate) time for "standard" algorithm (sec) |
| $T_d'$ | = | derivative (rate) time for "series" algorithm (sec) |
| $T_d''$ | = | derivative (rate) time for "parallel" algorithm (sec) |

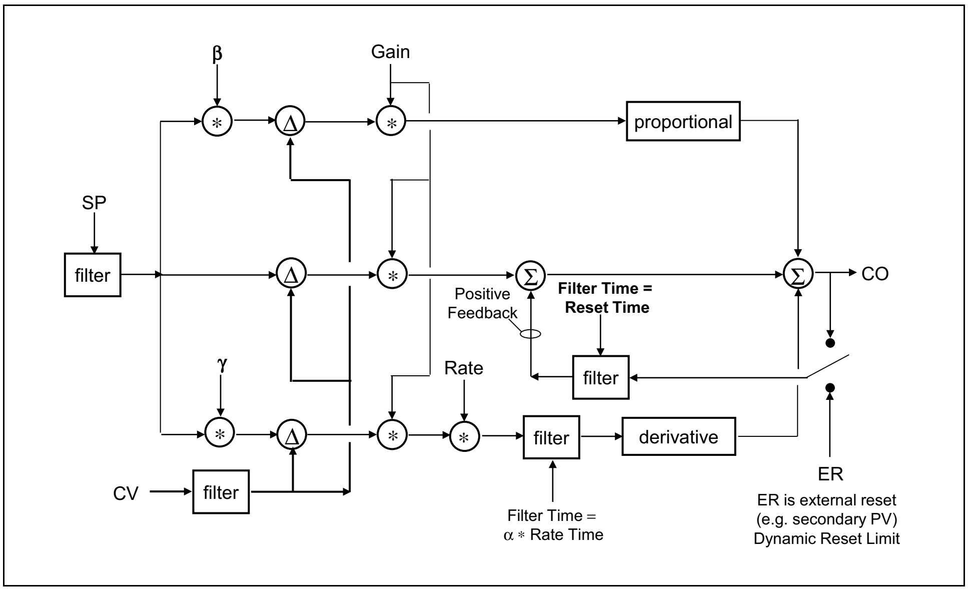


**Figure 3-4. Positive feedback implementation of integral mode**

Instead of an integration of the error for the integral mode calculation, some manufacturers have found it advantageous to use a filtered positive feedback, as shown in Figure 3-4 [4]. The input to the filter is either the controller output or an external feedback signal. The output of the filter is added to the net of the proportional and derivative modes in conformance to either the "series" or "standard" form. The filter time is the reset time setting. This positive feedback arrangement facilitates three important features [16]. First, the positive feedback makes it possible to use a "dynamic reset" option in which the external feedback is the PV of a secondary loop or control valve position. This option prevents the controller output from trying to go faster than the velocity-limiting effect of the reset time or the process time constant of the secondary loop, or faster than the slewing rate of a control valve. Second, the positive feedback arrangement inherently prevents a "walk-off" of controller outputs for override control. In the "walk-off," the controller outputs gradually move to an output limit from a continual back-and-forth selection of controller outputs when the integral mode uses an integration of error. The solution for these controllers is to add a filter to each external feedback signal with the filter time set equal to the respective controller's reset time. This effectively creates the same configuration offered by the positive feedback method when the controller is in the integral tracking mode. Third, a dead-time block can be added to the positive feedback path with its dead time set equal to the loop dead time. This provides a dead-time compensator as effective as a Lambda-tuned Smith Predictor but without the adjustments of process gain and lag.

***The positive feedback type of integral mode prevents reset action from outrunning the speed of response of secondary loops and final elements, eliminates walk-off of override controller outputs, and facilitates simplified dead-time compensation.***

## 3-3. PID Tuning

**Focus**

The wide spectrum of methods and results for tuning controllers can be bewildering. Most of the published literature analyzes methods for a particular range and type of process dynamics, disturbances at the process's output rather than its input, a set point response, fixed small dead times, and an accuracy of tuning settings not obtainable in industry. In general, the applications are high-speed servomechanism-type responses with measurement noise. Process disturbances enter into the process as changes in the charges, flows, or metabolic processes of the cell. These load upsets are process inputs that enter the process just as the manipulated variables do, as shown in the block diagram in Figure 2-2a of Chapter 2. Disturbances and the manipulated variables are process inputs that change the charge, component, and energy balances. The dead times and time constants associated with the primary loops' response are large and variable. Continual load upsets, the moving target of a batch profile, the nonlinearities enhanced by changing batch conditions, variable dynamics, resolution limits, and process noise from imperfect mixing place severe practical limits on the repeatability of tuning settings.

***Control theory centers on high-speed servomechanism response with noise.***

This section focuses on the Lambda method, showing how it can be adjusted to meet process goals and giving when desirable an equation similar to other tuning methods that are touted for loop performance. A translation of form also enables a much faster test time. The Lambda method has almost become a universal method because of its fundamental design.

***The Lambda tuning method can be adjusted or transformed to meet any type of process objective and offers a unified approach to controller tuning.***

Most of the articles to date on loop tuning are relevant to self-regulating processes with dead-time-to-time-constant ratios in the range 0.5 to 2.0, set point changes or step disturbances entering into the process output whose effect is similar to the set point changes for the most common type of controller (PI with $\beta$=1), and a presupposed accuracy of tuning settings of 10 percent or better. When integrating processes have been evaluated for tuning, it has typically been a level loop on a surge tank.

***Most of the literature on tuning deals with a narrow range of dynamics, self-regulating processes, and a set point response for PI control action on error ($\beta$=1).***

In the process industry, 99 percent of the self-regulating control loops have dead-time-to-time-constant ratios that range from 0.05 to 20.0. A true or "near integrating" type of response is prevalent in primary loops for concentration, pressure, and temperature control. The integrating process gain is usually extremely small and is equivalent to a very large process time constant. Process load upsets and disturbances occur at the process input instead of at the process output as commonly shown in the literature. Consequently, in industrial applications a process variable's rate of change from an upset or disturbance usually depends on the process time constant or integrating process gain. For columns, reactors, and vessels, the rate of change is very slow.

These same loops exhibit a variability of 50 percent or more in tuning settings from process tests as a result of noise, nonlinearity, resolution limits, and unmeasured disturbances [2]. A repeatability of 25 percent of tuning test results is considered exceptional. Realizing tuning setting precision in industrial applications eliminates a lot of the hype associated with methods and software. It also prevents false expectations, which is particularly important since engineers and scientists are accustomed to computing numbers to two or more digits [5].

***Differentiating tuning methods and software based on the second digit of the tuning setting computation have little value in industry.***

## Temperature Loops

Temperature loops commonly have a process dead time and second time constant that are 0.01 to 0.1 of the largest time constant because of the interactive thermal lags [6]. A second-order model (two time constants) is useful for identifying the derivative time setting. However, in practice, it is difficult to accurately identify this second time constant so a first-order model (one time constant) is often used. This is often adequate for temperature loops since about half of the unidentified second time constant ends up as additional dead time and half ends up as an incremental increase in the largest time constant. The dead-time-to-time-constant ratio is about 0.01 to 0.1 whether a first- or second-order model is used.

***The dead-time-to-time-constant ratio for most temperature loops on mixed volumes is about 0.01 to 0.1 whether a first-order or second-order model is used.***

Because processes with large time constants and small dead times appear to ramp in the region of interest, the difference between a self-regulating and integrating process response is blurred and the distinction becomes a matter of convenience. If the process is visualized as integrating, the test does not have to wait until the process variable plateaus to a new final value. This is particularly advantageous for loops that have large time constants, since it takes about four time constants plus the dead time to reach steady state. If the user prefers to use tuning equations for self-regulating processes, the integrating process dynamics can be converted into self-regulating process dynamics, and vice versa.

The lack of a liquid discharge flow in a batch operation reduces the self-regulation of the temperature loop. However, for a reactor that has a well-designed coolant, the change in driving force across the heat transfer surface means that the temperature loop is not a pure integrator. For example, if the temperature change is large enough, the temperature difference between the process and coolant should be large enough to change the heat transfer to the coolant sufficiently for the temperature to reach a steady state. However, the temperature change may be much larger than the desired change in temperature and the time to reach a steady state too slow.

***Reactor temperature loops have a time constant so large that the distinction between a self-regulating and integrating response is a matter of test time requirements and computational preferences.***

**Secondary Loops**

In a cascade control system the output of the primary controller, such as column, reactor, and vessel temperature, is the set point of a secondary controller, which helps linearize the loop and compensate for disturbances that affect the final element's ability to do its job. Secondary flow and speed loops have a dead-time-to-time-constant ratio of between 1.0 and 4.0 unless a large signal filter has been added to measurement or there is significant velocity limiting in the drive or valve response. Most of the dead time in these loops comes from the module execution time and the resolution of the drive or valve response. Secondary coolant temperature loops have a much larger process dead time and time constant whose ratio and magnitudes significantly vary with the coolant system design.

**Closed and Open Loop Responses**

The terms *closed loop* and *open loop* are commonly used in control and are essential to understanding Lambda tuning. The term *closed loop* is used to denote that PID control action is active (PID is in AUTO, CAS, or RCAS modes). This means it is changing a manipulated variable in response to a process variable, effectively closing the signal path between the CV and CO of the PID in the block diagram of a control loop (see Figure 4-1 in Chapter 4). Conversely, the term *open loop* is used to denote that PID control action is suspended (PID in IMAN, MAN, or ROUT) and that the signal path is open between the CV and CO. The open loop response is the response of the process only. The closed loop response is the combined response of the process and PID and shows the effect of the control algorithm and tuning settings [2].

Figure 3-5 shows the open loop time constant ($\tau_o$) that is created by making a step change in the controller output when the controller is in manual for a self-regulating process. Figure 3-6 shows the closed loop time constant known as Lambda ($\lambda$) that is created by making a step change in the controller set point when the controller is in automatic. In both cases, the time constant is the time it takes the controlled variable to reach 63 percent of its final value after the process variable starts to change (after the loop dead time). In tuning methods in which the time constant is estimated as

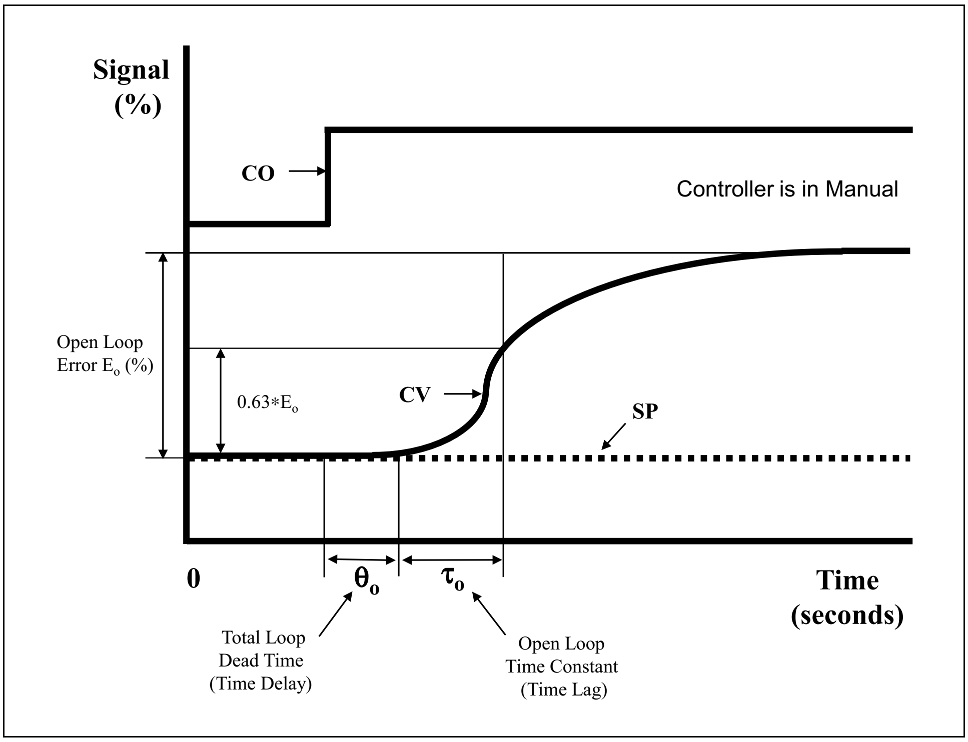
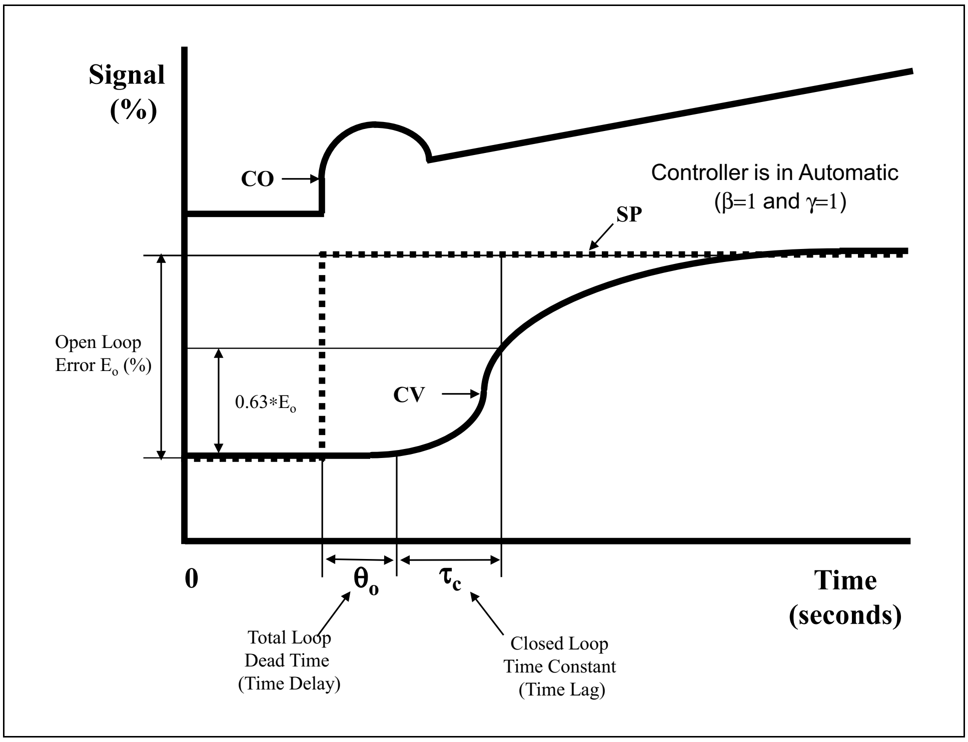

Signal (%)
CO
Controller is in Manual
Open Loop Error $E_o$ (%)
$0.63*E_o$
CV
SP
0
$\theta_o$
$\tau_o$
Time (seconds)
Total Loop Dead Time (Time Delay)
Open Loop Time Constant (Time Lag)

**Figure 3-5. Open loop time constant**

the time required to reach 98 percent of the final response divided by 4, the resulting time constant includes ¼ of dead time. In actual applications, there is no sharp transition from the negligible response caused by loop dead time and the exponential response associated with the primary time constant. The smoothing that occurs at the beginning of the exponential response is caused by a smaller second time constant ($\tau_2$). Though this small time constant is difficult in practice to identify, it can be estimated for temperature and many concentration loops on well-mixed volumes as being 1/10 of the largest time constant.

The open loop time constant is the largest time constant in the loop. If it is in the process between the point of entry of the disturbances and the sensor, the open loop time constant slows down the excursion caused by a disturbance. This gives the controller a better chance of catching up to the disturbance. If the largest time constant is in the measurement, it provides an attenuated view of process variability and slows down the reaction to unmeasured disturbances. Coated electrodes can cause the sensor time constant to approach or even exceed the process time constant. Also, in the case of a liquid flow or speed loop, a transmitter damping setting or process variable filter time setting that is greater than two seconds generally


Signal (%)
CO
Controller is in Automatic (β=1 and γ=1)
SP
Open Loop Error E$_o$ (%)
0.63*E$_o$
CV
0
θ$_o$
τ$_c$
Time (seconds)
Total Loop Dead Time (Time Delay)
Closed Loop Time Constant (Time Lag)
**Figure 3-6. Closed loop time constant**

means the largest time constant is in the measurement. This is because the liquid flow process and final element time constant is less than two seconds [2].

## Lambda Tuning

The ratio of the closed-loop-to-open-loop time constant is the Lambda factor, as in Equation 3-13. For secondary flow and speed loops, a Lambda factor of 2 ($\lambda_f = 2$) is a good starting point. This gives a closed loop time constant that is twice the open loop time constant. For primary loops, the Lambda factor is set less than one to provide tighter loop control.

$$\lambda = \lambda_f * \tau_1 \tag{3-13}$$

$$T_i = \tau_1 \tag{3-14}$$

$$K_c = \frac{T_i}{K_o * (\lambda + \theta_o)} \tag{3-15}$$

$$T_d = \tau_2 \tag{3-16}$$

where:

| | | |
|---|---|---|
| $K_c$ | = | controller gain (dimensionless) |
| $K_o$ | = | open loop gain (also known as process gain) (%/%) |
| $\lambda$ | = | Lambda (closed loop time constant) (sec) |
| $\lambda_f$ | = | Lambda factor (ratio of closed to open loop time constant) (dimensionless) |
| $\theta_o$ | = | total loop dead time (sec) |
| $\tau_1$ | = | largest open loop time constant (also known as process time constant) (sec) |
| $\tau_2$ | = | second largest open loop time constant (sec) |
| $T_i$ | = | integral (reset) time setting (sec/repeat) |
| $T_d$ | = | derivative (rate) time setting (sec) |

For derivative (rate) action to be effective, the loop response should be smooth and have a significant second time constant [3]. This is generally the case for temperature loops on mixed volumes. Since the second time constant is about 1/10 of the largest time constant, the rate time setting for temperature loops is about 1/10 of the reset time setting.

***Derivative action that has a rate time setting of about 1/10 of the reset time setting can provide tighter temperature control.***

**Contribution of Loop Components**

The open loop gain is the product of the steady-state gains for each major component in the loop, that is, the final element (manipulated variable), process piping and equipment (process variable), and sensor and transmitter (controlled variable), as shown in Equation 3-17. The result must be dimensionless (%/%), and if it is not, a component is missing or the engineering units are not consistent. The manipulated variable gain is generally more linear for variable speed drives than for control valves. The process gain is usually nonlinear for temperature and concentration loops. The process gain is 1.0 for flow. The measurement gain is linear and provides a conversion from the engineering units of the process variable into percent of the measurement span ($K_{cv}$ =100%/EUspan) [2] [3] [7].

$$K_o = K_{mv} * K_{pv} * K_{cv} \tag{3-17}$$

where:

| | | |
|---|---|---|
| $K_o$ | = | open loop gain (%/%) |
| $K_{cv}$ | = | controlled variable (measurement) gain (%/EU) |
| $K_{pv}$ | = | process variable (process) gain (EU/EU) |
| $K_{mv}$ | = | manipulated variable (final element) gain (EU/%) |

Note that in the literature on control, no analysis is usually given of the contribution of loop components to the open loop time constant or the open loop gain. In fact, everything outside of the controller is often referred to as the process, and there is no consideration or even appearance on a control diagram of final element and instrumentation details or location. Consequently, the open loop time constant, open loop gain, and the total loop dead time are called the *process time constant*, *process gain*, and *process dead time*, respectively. The total loop dead time is the summation of all pure dead times and time constants smaller than the second-largest time constant, no matter where they appear in the loop.

***The open loop gain is commonly called a process gain despite the fact that it is dependent on the product of the final element, process, and measurement gains.***

***The open loop time constant is commonly called a process time constant even though it is the largest time constant in the final element, process, or measurement.***

***The total loop dead time is commonly called a process dead time even though it is really the summation of all time constants smaller than the second-largest time constant and the dead times in the final element, processes, and measurement.***

**Unified Approach**
Since in columns, reactors, and vessels there are few interactions and small changes in controller outputs can cause resolution delays, smoothing the manipulated flow may be less important than the ability to control a primary loop at its set point. For the tightest control, the tuning is set to transfer as much variability from the process variable (controlled process output) to the controller output (manipulated process input) as possible. Suppose there is a Lambda factor of 0.1 and a dead-time-to-time-constant

ratio of 0.1, as commonly found in temperature loops. In this case, the Lambda tuning equation reduces to ½ the open loop time constant divided by the product of the open loop gain and total loop dead time. The result is Equation 3-18, which is the simplified internal model control (SIMC) equation for tight control. Most documentation on Lambda tuning has focused on slowing down the response of fast loops so as to provide a smoother and a more consistent response, which is important for improving the coordination or reducing the interaction between loops on unagitated volumes. However, nothing prevents the Lambda tuning factor from being set less than 1.0 in order to provide much tighter control. Furthermore, unlike most other tuning knobs, setting the Lambda factor for more aggressive control does not cause the loop to become unstable. Rather, the loop just approaches the tuning equations cited for best load rejection capability. In fact, most of the equations developed in the 1960s and 1970s that focused on excellent load rejection have this common form of the controller gain, in which the gain is set equal to some factored ratio of the process time constant to the product of the open loop gain and total loop dead time [2]. Appendix C, "Unification of Controller Tuning Relationships," in this book, shows how Ziegler-Nichols, Lambda tuning, and internal model control tuning equations reduce to this common form. This reduction confirms diverse methods and opens the door for a unified approach to tuning [8].

$$K_c = 0.5 * \frac{\tau_o}{K_o * \theta_o} \tag{3-18}$$

***For tight control, most of the tuning settings that have been developed are reducible to the common form, in which the controller gain is proportional to the time-constant-to-dead-time ratio.***

The Lambda factor should be set based on the size of the open loop time constant and whether the goal is tighter control or better coordination and less interaction between loops. For cascade control, the secondary loop's time response should be five times faster than the primary loop's to prevent interaction between these loops [2] [3]. The ratio of the closed loop time constants of the primary to secondary loop should be greater than five to meet this criterion. If the Lambda factor for the secondary loop is 2, then the ratio should be large enough for even bench-top units. For a flow or speed loop that has a two-second time constant, the secondary loop

Lambda would be four seconds, which means the primary loop Lambda should be at least twenty seconds. This is not a problem for temperature, but could be a consideration for a pressure loop that is near the end of a batch when the off-gas flow is high.

**Near Integrators**

For "near" integrators, the integrator gain is the steady-state open loop gain divided by the open loop time constant, as shown in Equation 3-20. Note that the units of %/% per second or 1/second are better understood as the %/sec change in the ramp rate of the CV per percent change in the CO for the controller in manual. Since the lack of a steady state means that the process ramps when the controller is in manual, the integrator gain is measured as the change in ramp rates from before to after the change controller output. This method is also known as the "short cut method" and is an extension of the process reaction curve method developed by Ziegler and Nichols [3] [8] [9]. The dead time is the time from the change in controller output to a significant change in ramp rate. Normally, the output change is in the direction that will change the direction of the ramp, which means the dead time is the time until the reversal of slope [3].

$$K_i = \frac{K_o}{\tau_o} \tag{3-19}$$

$$K_i = \frac{CV_2/\Delta t - CV_1/\Delta t}{\Delta CO} \tag{3-20}$$

If we substitute Equation 3-19 into Equation 3-15, we have the controller gain for a "near integrating" process [2] [3] [8]. The tuning settings can now be determined from the change in ramp rates per % and total loop dead time and a desired closed loop time constant (λ). The tuning method or software does not need to wait for the time to steady state but can usually identify the new ramp rate in five loop dead times. If you consider that the loop dead time is 0.1 or less than the process time constant and that it would take five time constants plus the dead time if a steady state could be reached, then the test time to identity the model and hence the tuning settings is reduced by an order of magnitude. For example, the time required identifying the dynamics of a primary loop with a dead time of one minute and a time constant of ten minutes for a step change in controller output would be fifty minutes as a self-regulating response and five minutes as an integrating response. The actual time that software takes may be twice as long as stated to help screen out the effect of noise and disturbances. The test should be repeated in both directions if possible, and it may require performing tests at different batch times for several batches to see how much the dynamics change with batch conditions. For these reasons, the time saved by modeling processes with large time constants as "near integrators" is significant.

***Modeling processes that have large time constants as "near integrators" can reduce by a factor of ten the open loop test time for identifying the process dynamics.***

$$K_c = \frac{1}{K_i * (\lambda + \theta_o)} \tag{3-21}$$

If the controller on a process that has a large time constant is properly tuned with a Lambda factor of much less than one, then the time required for a closed loop test is reduced dramatically for the self-regulating model. This is because the closed loop time constant is a fraction of the open loop time constant.

***Tight tuning of primary loops with large process time constants makes it possible to much more quickly identify self-regulating models for a closed loop test.***

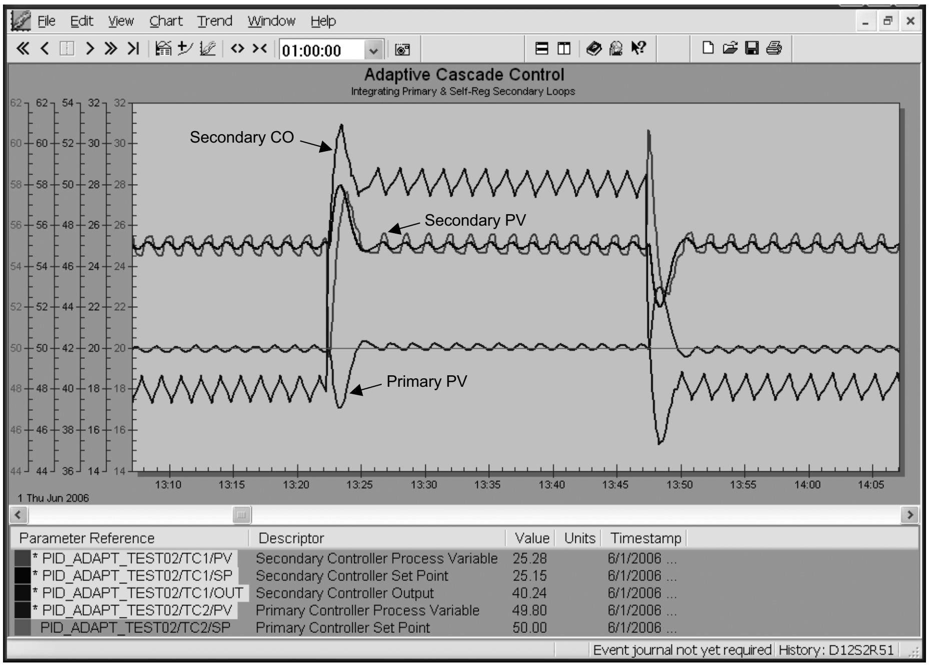
**Loop Cycling**

"Near integrating" and "pure integrating" processes develop slow rolling oscillations if the integral (reset) time is too small (i.e., too fast), and they have the unusual characteristic of an increasing propensity to develop these oscillations as the controller gain is decreased. In Appendix C, "Unification of Controller Tuning Relationships," Equation 3-22 is developed for a reset time that gives zero overshoot (critically damped response) [8]. If a slight overshoot and oscillations with a decay ratio of 1/20 is permissible, then the numerator can be reduced to 0.7 for faster load rejection. A range of 1 to 4 for the numerator is used in practice.

$$T_i > \frac{4}{K_i * K_c} \tag{3-22}$$

***Integrating processes exhibit the following nonintuitive behavior: integral time is reduced if the controller gain is increased because the minimum integral (reset) time setting is inversely proportional to the controller gain.***

Primary loops that have too large a controller gain develop oscillations that have a period of close to the ultimate period (e.g., four dead times). This is much faster than the slow rolling oscillations that are caused by combining a low controller gain and low reset time [2] [3]. Ultimate oscillations are uncommon in column, reactor, and vessel loops because the maximum gain (ultimate gain) that would trigger these oscillations is quite high for well-designed primary loops. Relatively fast oscillations in the primary loop often indicate a piping or mixing problem. More common if not hidden by historian data compression is the limit cycle from the resolution limits of the final elements and measurements (Figure 3-7), the slowly decaying oscillations caused by a reset setting too fast for a controller gain setting (Figure 3-8), or secondary loops that are tuned for too slow of a response (Figure 3-9).

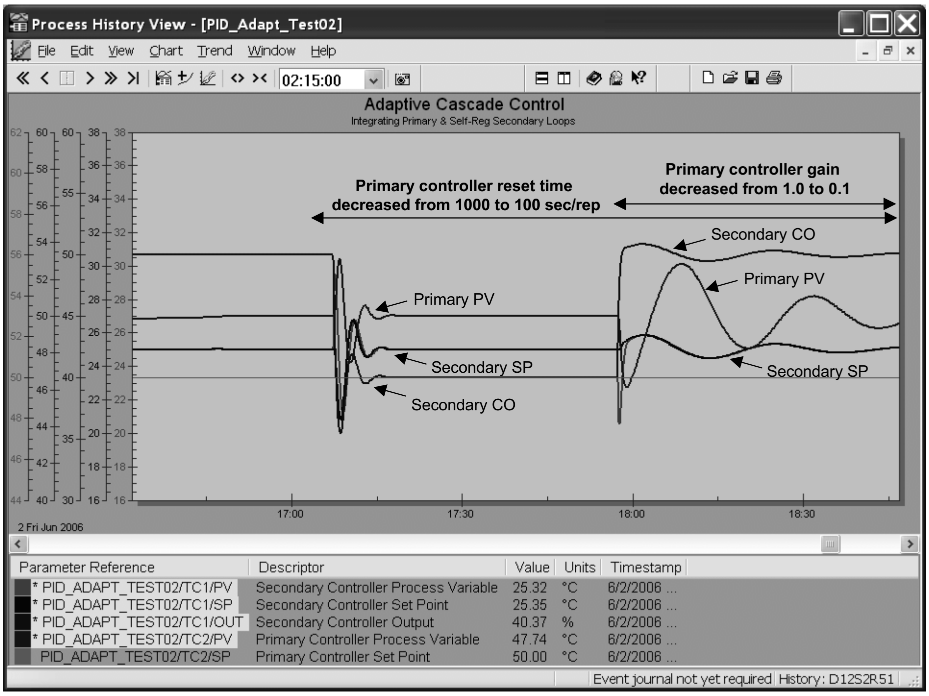


| | Parameter Reference | Descriptor | Value | Units | Timestamp |
|---|---|---|---|---|---|
| ■ | * PID_ADAPT_TEST02/TC1/PV | Secondary Controller Process Variable | 25.28 | | 6/1/2006 ... |
| ■ | * PID_ADAPT_TEST02/TC1/SP | Secondary Controller Set Point | 25.15 | | 6/1/2006 ... |
| ■ | * PID_ADAPT_TEST02/TC1/OUT | Secondary Controller Output | 40.24 | | 6/1/2006 ... |
| ■ | * PID_ADAPT_TEST02/TC2/PV | Primary Controller Process Variable | 49.80 | | 6/1/2006 ... |
| ■ | PID_ADAPT_TEST02/TC2/SP | Primary Controller Set Point | 50.00 | | 6/1/2006 ... |

Event journal not yet required | History: D12S2R51

**Figure 3-7. Limit cycle in cascade loop from final element resolution limit**

*Most of the oscillations in primary loops result from limit cycles caused by resolution limits, slowly decaying cycles from too much reset action, and interaction with secondary loops.*

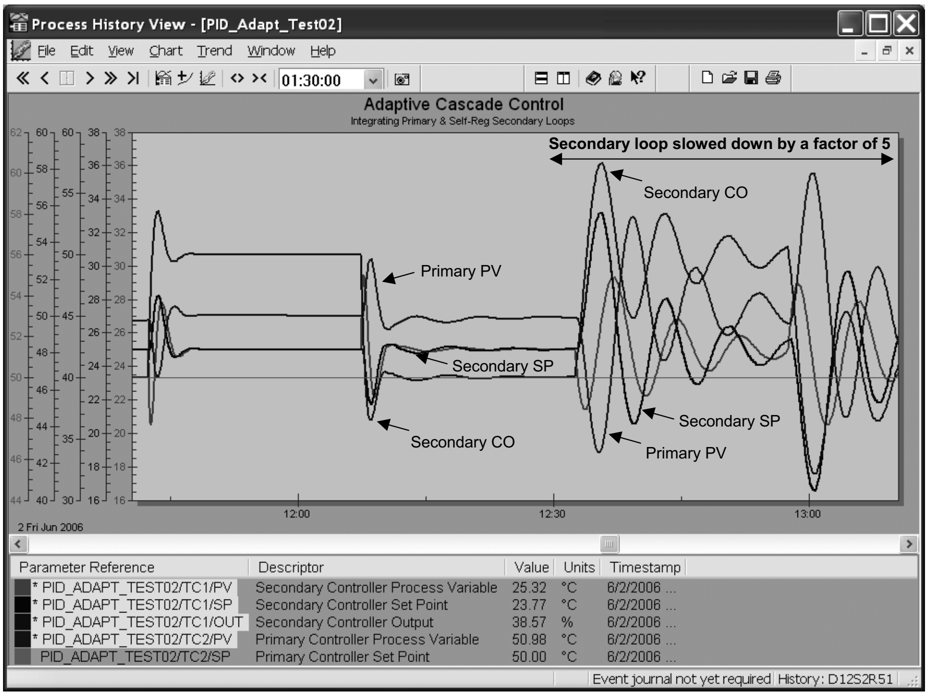

Process History View - [PID_Adapt_Test02]
File Edit View Chart Trend Window Help
02:15:00
Adaptive Cascade Control
Integrating Primary & Self-Reg Secondary Loops
Primary controller reset time decreased from 1000 to 100 sec/rep
Primary controller gain decreased from 1.0 to 0.1
Secondary CO
Primary PV
Primary PV
Secondary SP
Secondary SP
Secondary CO
17:00
17:30
18:00
18:30
2 Fri Jun 2006

| Parameter Reference | Descriptor | Value | Units | Timestamp |
|---|---|---|---|---|
| * PID_ADAPT_TEST02/TC1/PV | Secondary Controller Process Variable | 25.32 | °C | 6/2/2006 ... |
| * PID_ADAPT_TEST02/TC1/SP | Secondary Controller Set Point | 25.35 | °C | 6/2/2006 ... |
| * PID_ADAPT_TEST02/TC1/OUT | Secondary Controller Output | 40.37 | % | 6/2/2006 ... |
| * PID_ADAPT_TEST02/TC2/PV | Primary Controller Process Variable | 47.74 | °C | 6/2/2006 ... |
| PID_ADAPT_TEST02/TC2/SP | Primary Controller Set Point | 50.00 | °C | 6/2/2006 ... |

Event journal not yet required | History: D12S2R51

**Figure 3-8. Slowly decaying oscillation in a cascade loop from an integrating primary loop reset time that is too fast per equation 3-13j**

Most of the "near integrator" processes have a large process time constant and a small process gain. Consequently, integrator gains of about 0.001 to 0.1 and 0.00001 to 0.001%/sec per % are common for reactor temperature loops. Tuning the controller with maximum gain for a dead time of 100 seconds would correspond to a controller gain of greater than 10, which is beyond the comfort zone of most users. Consequently, a more moderate controller gain is used that transfers as much of the variability from the controlled variable to the manipulated variable as desired. Since this controller gain is often significantly less than the maximum allowed by Equation 3-21, it is critical that Equation 3-22 be used to prevent slow rolling oscillations.


Process History View - [PID_Adapt_Test02]
File Edit View Chart Trend Window Help
01:30:00
Adaptive Cascade Control
Integrating Primary & Self-Reg Secondary Loops
Secondary loop slowed down by a factor of 5
Secondary CO
Primary PV
Secondary SP
Secondary CO
Secondary SP
Primary PV
12:00
12:30
13:00
2 Fri Jun 2006

| Parameter Reference | Descriptor | Value | Units | Timestamp |
|---|---|---|---|---|
| * PID_ADAPT_TEST02/TC1/PV | Secondary Controller Process Variable | 25.32 | °C | 6/2/2006 ... |
| * PID_ADAPT_TEST02/TC1/SP | Secondary Controller Set Point | 23.77 | °C | 6/2/2006 ... |
| * PID_ADAPT_TEST02/TC1/OUT | Secondary Controller Output | 38.57 | % | 6/2/2006 ... |
| * PID_ADAPT_TEST02/TC2/PV | Primary Controller Process Variable | 50.98 | °C | 6/2/2006 ... |
| PID_ADAPT_TEST02/TC2/SP | Primary Controller Set Point | 50.00 | °C | 6/2/2006 ... |

Event journal not yet required | History: D12S2R51

**Figure 3-9. Interacting oscillations in cascade loop from a slow secondary loop**

***Since the primary loop controller gain is set significantly below its maximum, it is imperative that the reset time be increased to prevent slowly decaying oscillations.***

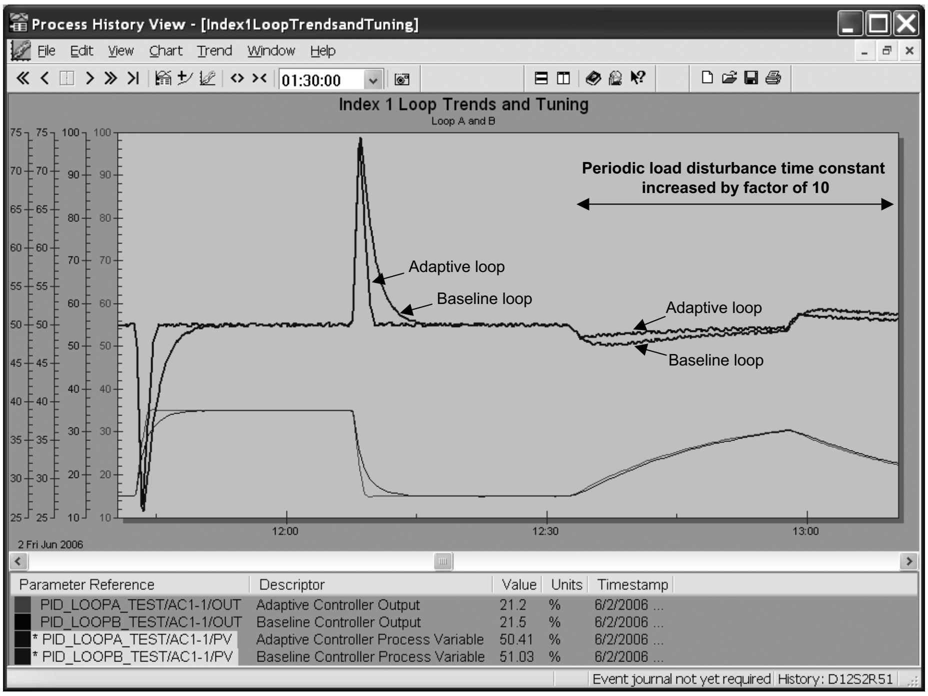
It is important to remember that there is a balance point with all integrators, that the degree of imbalance determines the ramp rate, and that the ramp does not change direction until the unbalance changes sign. The implications of this behavior are explored in Section 3-5 on optimizing the set point response.

***The ramp rate of an integrating process is proportional to the magnitude of the difference between the feed and the demand or exit flow, and the ramp rate cannot reverse direction until the unbalance reverses sign.***

The ramp rate of the integrator response is inversely proportional to the mass holdup, which is the volume for a constant density. Of course, the scaling of flows should match the volume, but the sizing and unbalance may not match. Bench-top units tend to have much faster ramp rates. The change in ramp rate (integrating process gain) must be considered when moving tuning settings to production units.

***The ramp rate of an integrating process is inversely proportional to the volume.***

Process disturbances for columns, reactors, and vessels are generally very slow because of the volume. The loop's performance for slow disturbances depends more on integral action than on proportional action. The rate of change of the controller output needs to effectively exceed the load disturbance's rate of change in order to return the PV to set point. Figure 3-10 shows how a small persistent offset develops in response to a second upset that is the same size as the first upset but ten times slower.


Process History View - [Index1LoopTrendsandTuning]
File Edit View Chart Trend Window Help
01:30:00
Index 1 Loop Trends and Tuning
Loop A and B
Periodic load disturbance time constant increased by factor of 10
Adaptive loop
Baseline loop
Adaptive loop
Baseline loop
12:00
12:30
13:00
2 Fri Jun 2006
Event journal not yet required
History: D12S2R51

| Parameter Reference | Descriptor | Value | Units | Timestamp |
|---|---|---|---|---|
| PID_LOOPA_TEST/AC1-1/OUT | Adaptive Controller Output | 21.2 | % | 6/2/2006 ... |
| PID_LOOPB_TEST/AC1-1/OUT | Baseline Controller Output | 21.5 | % | 6/2/2006 ... |
| * PID_LOOPA_TEST/AC1-1/PV | Adaptive Controller Process Variable | 50.41 | % | 6/2/2006 ... |
| * PID_LOOPB_TEST/AC1-1/PV | Baseline Controller Process Variable | 51.03 | % | 6/2/2006 ... |

**Figure 3-10. The effect of load disturbance speed on process recovery**

Lambda tuning equations specifically designed for pure integrating processes are available. Appendix C, "Unification of Controller Tuning Relationships," details the use and reduction of these more exact equations into the simpler unified form shown here for the "near" integrating processes of common column, reactor, and vessel loops. The appendix shows how the more exact equations for “pure” integrating processes provide a maximum controller gain that is about 50 percent larger than what is estimated from Equation 3-21 for the low integrator gains seen in columns, reactors, and vessels. Since the actual controller gain used in practice is typically far below this maximum, the more complex equations in Appendix C are more useful for deploying in software packages for controller tuning than for understanding or estimating tuning settings.

## 3-4. Adaptive Control

**Theory and Reality**
Process control systems assume a constant linear process. Unfortunately, all process variables and control valves are nonlinear to some degree: the process response to a given change in the controller output changes with batch time. The lack of consistency in the process response has significant implications for the process's performance not only in terms of tuning controllers but of recognizing degradations and achieving optimums [12].

**Road Maps and Terrain**
Consciously or subconsciously, tuning controllers involves a tradeoff between performance and robustness. The controller's ability to tightly control at an operating point is inversely proportional to its ability to weather changes in the plant's behavior without become oscillatory. The operating environment for most loops is stormy, and the last thing you want is for a control loop to introduce more variability. Consequently, all controllers are detuned (backed off from maximum performance) to some degree to provide a smooth response, despite the inevitable changes in the process dynamics. A PID controller approaches turns cautiously since it doesn't know what lies ahead [12].

***PID controllers are backed off from best performance because of the uncertainty of tuning settings.***

Controller tuning settings can be computed from a first-order or an integrating-plus-dead-time process model. The changes in the model parameters reveal changes in the cells, process conditions, equipment, final elements, and sensors. The size, direction, and characteristics of these changes can provide a road map and knowledge of the terrain [12].

***Changes in the parameters of a dynamic process model identified by an adaptive controller provide insight into changes in the process, equipment, and sensor.***

**Glimpses and Grimaces**

Nearly all of the industrial adaptive controllers presently used in industrial processes require that changes in the process variable be observed over rather a long time and they show the results in terms of new tuning settings. The tuning rules are imbedded and usually unknown. The most commonly used adaptive controller today operates by using pattern recognition and, if it's deemed necessary, it increases the controller gain to induce oscillations. The size of the transients or oscillations and the time required for identification can translate into significant process variability and an adaptation rate that is slower than the rate of change of the process parameters. In fact, most adaptive controllers are playing catch-up even if they have seen the same situation a thousand times before. At best, these controllers provide a snapshot of the current tuning requirements but no real process insight into where the process has been or where it is going. Also, sudden unexplained shifts in the tuning settings or bursts of oscillations reduce the operator's confidence and decrease the likelihood that the controllers will run in the adaptive mode and be used in future applications [11] [12].

***Model-free and pattern-recognition adaptive controllers do not offer process knowledge and are playing catch-up.***

**Watching but Not Waiting**

The next generation of adaptive controllers can identify a process model quickly and automatically and provide process model parameters that can be displayed, trended, and diagnosed. Furthermore, these controllers remember the results for similar conditions, eliminate repetitious identification, and take the initiative [12].

An adaptive controller with these desirable features has been demonstrated in plant tests. The controller can identify the dead time, process gain, and time constant for both manipulated and disturbance variables and save these as a function of a key variable. The user can use the recommended tuning method or elect to choose an alternative method to compute the current tuning settings for the current and memorized conditions. When the key variable indicates that the process has changed, the tuning is then scheduled based on the process model saved in the operating region. The adaptive controller remembers the results from previous excursions and does not wait to recognize old territory. For example, in loops that have nonlinear installed valve characteristics and nonlinear controlled variables, the model and tuning are scheduled based on the controller output and input, respectively. To change dynamics as the batch progresses, the model and tuning are scheduled based on totalized feed. The adaptive controller takes preemptive action based on operating region and uses the opportunity to refine its knowledge of the process model. Changes in these models can flag changes in seed cultures [12].

***Adaptive controllers should learn, remember, and utilize knowledge gained from previous batches.***

The adaptive controller computes the integrated squared error (ISE) between the model and the process output for changes in each of three model parameters from the last best value. To explore all combinations of three values (low, middle, and high) for three parameters, twenty-seven models are ultimately generated. The correction in each model parameter is interpolated by applying weighting factors that are based on the ISE for each model, normalized to a total ISE for all the models over the period of interest. After the best values are computed for each parameter, they are assigned as the middle values for the next iteration [13]. This model switching with interpolation and recentering has been proved mathematically by the University of California, Santa Barbara, to be equivalent to least square identification and provides an optimum approach to the correct model [14]. The search is actually done sequentially, first for the process gain, then the dead time, and finally the time constant, which reduces the number of models to nine [13]. Figure 3-11 shows the setup of an adaptive controller that identifies the process model for the controlled variable's (ΔCV) response to changes in the controller output (ΔCO) and disturbance variable (ΔDV). This response is then used to compute the feedback controller tuning settings and the feedforward dynamic compensation, respectively [12].

The adaptive controller starts in the "Observe" mode, in which it continuously and automatically identifies the process model when it sees changes in the controller's set point, output, or feedforward. The adaptive controller can also be switched to the "Learn" mode, in which it updates the feedback and feedforward process models in each region for which it sees an

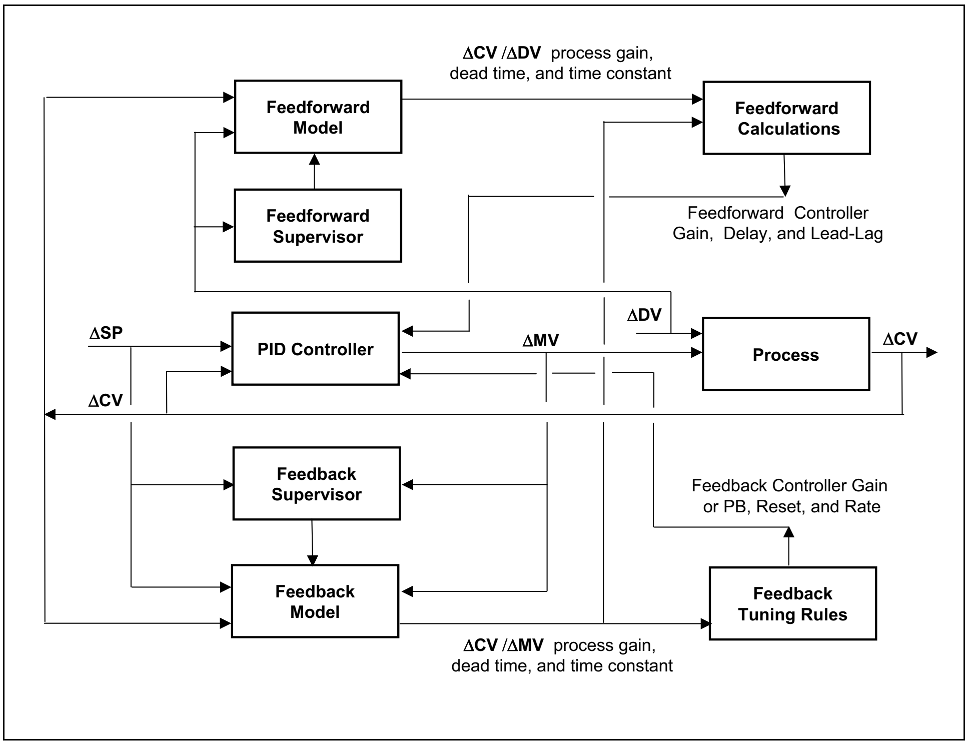

ΔCV /ΔDV process gain, dead time, and time constant
Feedforward Model
Feedforward Calculations
Feedforward Supervisor
Feedforward Controller Gain, Delay, and Lead-Lag
ΔDV
ΔSP
PID Controller
ΔMV
Process
ΔCV
ΔCV
Feedback Supervisor
Feedback Controller Gain or PB, Reset, and Rate
Feedback Model
Feedback Tuning Rules
ΔCV /ΔMV process gain, dead time, and time constant

**Figure 3-11. Adaptive controller setup based on identifying process models**

excursion. The next option is the "Schedule" mode, where the adaptive controller uses the models in each region to change the controller settings. The highest level is the "Adapt" mode, in which the adaptive controller immediately uses any identified improvements [12].

***An adaptive controller can be run in the "Observe" and "Learn" mode in order to see what it can capture as process knowledge as a function of batch time or variables.***

## Back to the Future

This new generation of adaptive controllers allows all PID loops to run in the adaptive mode and provides process model parameters that are saved in a data historian and analyzed for changes in the plant, sensors, and valves. The information on changes in the process model may be directly used to monitor loop performance and to provide more intelligent diagnostics. The models can provide the dynamics for simulations and identify candidates for advanced control techniques. For example, loops that have large dead times or a one-way integrating response are prime candidates for model predictive control [12].

## 3-5. Set-Point Response Optimization

**Nothing Says Forever Like Tradition**
There are four major practices for starting up a loop that has a large process time constant or slow ramp time compared to the dead time [15]:

**Loop Practices for Fast Batch and Startup Response**

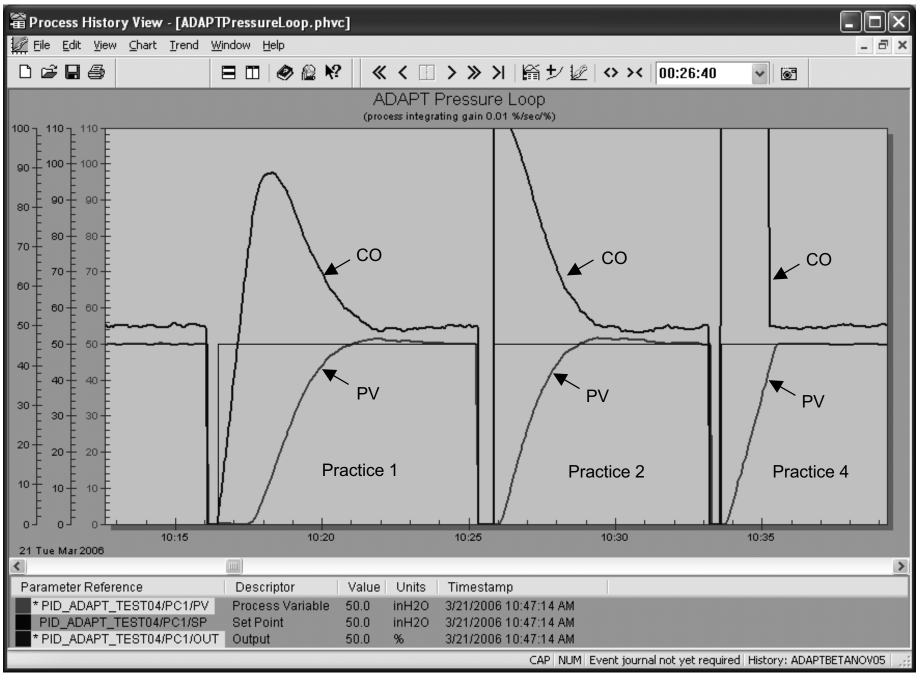
1. Switch controller to auto with the final set point.
2. Switch controller to auto with an initial set point and then switch to final set point.
3. Put controller in manual or output tracking with final set point, set valve to its normal position, wait, and switch to auto.
4. Put controller in manual or output tracking with final set point, set valve to extreme position, wait, switch valve to normal position, wait, and switch to auto.

Figure 3-12 shows the batch or startup response of a pressure loop that has an integrating response for practices 1, 2, and 4. Practice 3 is not shown because it is not viable for integrating processes. Other practices exist, such as ramping the set point, for unit operations in which it is desirable that the process variable's approach to set point and the output to its final resting value are moderated or that a profile is enforced [15].

***The controller output must be positioned beyond the final resting value (balance point) in an integrating process in order to get the CV to move toward set point.***

In the first and second practices, the controller output is at its initial value at one end or the other of the output scale (often zero). All methods assume that the pump and block valves have already been started and opened, respectively [15].

In the first practice, if the loop is tuned to minimize variability in the controller output, which is the case for surge volume level control, then the batch phase may time out before the process reaches set point. For example, if the process time constant is 50 minutes and a Lambda factor of five is used, then the closed loop time constant is 250 minutes and the time to


Process History View - [ADAPTPressureLoop.phvc]
File Edit View Chart Trend Window Help
00:26:40
ADAPT Pressure Loop
(process integrating gain 0.01 %/sec/%)
CO
CO
CO
PV
PV
PV
Practice 1
Practice 2
Practice 4
10:15
10:20
10:25
10:30
10:35
21 Tue Mar 2006
CAP NUM Event journal not yet required History: ADAPTBETANOV05

| Parameter Reference | Descriptor | Value | Units | Timestamp |
|---|---|---|---|---|
| * PID_ADAPT_TEST04/PC1/PV | Process Variable | 50.0 | inH2O | 3/21/2006 10:47:14 AM |
| PID_ADAPT_TEST04/PC1/SP | Set Point | 50.0 | inH2O | 3/21/2006 10:47:14 AM |
| * PID_ADAPT_TEST04/PC1/OUT | Output | 50.0 | % | 3/21/2006 10:47:14 AM |

**Figure 3-12. Batch and startup performance of an integrating loop**

reach 98 percent of set point is 1000 minutes (four closed loop time constants). A similar situation exists for slow ramp rates [15].

A dead time that is much faster than the process time constant or ramp rate usually means that a Lambda factor of less than one (i.e., a closed loop time constant or arrest time that is less than the open loop time constant or arrest time) is permissible to achieve stability and is desirable to achieve fast control of these process variables. This is particularly important for the practice 1 because you are relying on reset action to get you set point. All the batch or startup responses in Figure 3-12 use a Lambda factor of less than one [15].

***If the kick from the proportional mode is negligible, the burden is entirely on the integral mode (reset action), and the approach to set point will be much slower.***

In the second practice, the set point is changed from its initial to final value one execution or more after the controller is switched to auto. Note that if you switch the set point within the same execution of the module as the switch of the mode, you will probably end up with the same response as the first practice. In the second batch or startup response, the set point change kicks the output, which gives the process variable a boost on its way to set point. The time to reach set point (rise time) is nearly cut in half, but the settling time is about the same. Since the overshoot is minimal, the rise time might be more important. Also, the controller tuning could be tweaked to reduce settling time [15].

***If the set point change is made during the same execution as a mode change, there will probably be no kick from the set point change, despite a $\beta$ factor greater than zero.***

Many of the more astute automation engineers pre-position the controller output by what is called a *head start* or *process action*. For self-regulating loops, the valve position might be set at what was considered to be a normal throttle position or final resting value (FRV), as seen from previous trends when the process variable had settled out at set point. This corresponds to a Lambda factor of one because, if held at this position, it drives the process variable with a time constant that is equal to the process time constant [15].

***A Lambda factor of one for self-regulating loops gives an approach to set point at the speed seen in the open loop response for a controller output that is moved to the FRV.***

For integrating responses, the process variable won't go anywhere until the valve is positioned beyond its FRV. This leads us to practice 4, in which the valve is set to an extreme position allowable by the process in order to give the fastest approach to set point. Then the brakes are slammed on so the process variable does not run over the set point. The question is, When do you hit the brakes [15]?

**The Wait**

The plot for the fourth practice shows the response for a technique in which the rate of change is computed from the change in the controlled variable (CV) over a time period long enough to get a good signal-to-noise ratio. The old value of the CV, created by passing the CV through a dead-time block, is subtracted from the new CV. The delta CV is divided by the block dead time to create a rate of change. The rate of change multiplied by the process dead time is then the predicted change in the CV that when added to the new CV is the predicted end point as shown in Equation 3-23. When the end point equals or exceeds the final set point, the controller output is switched from maximum throttle to its FRV. It is held at this FRV for one process dead time and is then released for feedback control. This method compensates for nonlinearities and disturbances that are evident at the time to hit the brakes [15].

***The controlled variable's rate of change multiplied by the loop's dead time and added to the old controller's output provides a prediction of the end point.***

$$CV_f = [(CV_n - CV_o) / DT] * \theta_o + CV_n \tag{3-23}$$

where:

| | | |
|---|---|---|
| $CV_f$ | = | predicted CV one dead time into the future (%) |
| $CV_n$ | = | new CV (%) |
| $CV_o$ | = | old CV (output of dead time block) (%) |
| $DT$ | = | DT block dead time (sec) |
| $\theta_o$ | = | total loop dead time (sec) |

If the process dead time is underestimated, the loop overshoots the set point. Therefore, it is important to be generous in the dead time estimate. It is especially important that the dead time not be too short for the zero load integrating process, where the FRV is zero and there is nothing to bring the process variable back to set point [15].

***It is better to overestimate the dead time when predicting the end point.***

**Without Dead Time I Would Be Out of a Job**

If the loop dead time is zero, the loop could switch to the FRV when the CV reached set point. Furthermore, the sky is the limit, for the controller gain and feedback action could provide instantaneous correction. My life-style is largely the result of dead time.

A better term than process dead time is *total loop dead time* because there are many sources of dead time outside of the process. The biggest source for slow ramp times and process time constants is the measurement and valve resolution. The time it takes for the change to get through the resolution limit is dead time. The dead time from the measurement and valve resolution are inversely proportional to the rate of change of the process variable and controller output, respectively. Consequently, for closed loop tests the identified dead time depends on the controller tuning and the size of the change in the set point. Fortunately, the changes in valve position are quite large, and the dead time from resolution is minimal for the optimal switching method described in practice 4.

Other sources of instrument dead time include measurement transportation delay, sensor lags, transmitter dampening, analog and digital filters, module execution time, valve dead band, and actuator prestroke dead time.

An adaptive controller can identify the total loop dead time accurately if the trigger points in terms of output changes are large enough. Note that the ultimate proof in the pudding is the output change rather than the set point change. This is because output change includes the effect of tuning and is ultimately what is driving the process. Given the measurement and valve resolution, the adaptive controller with its knowledge of the integrating process gain can correct the observed dead time to give a value that is closer to the output changes associated with the optimal switching [15].

An adaptive controller can also identify the integrating process gain. This can be used with the current ramp rate and the pre-positioned extreme controller output to estimate the FRV, following Equation 3-24. Note that if the extreme output ($CO_X$) is less than the FRV, the signs of each expression are reversed to get a positive FRV. Of course, limits should be enforced on the calculated value, and it may be desirable to estimate the new FRV by using a portion of the difference between the calculated FRV and the last captured FRV added to the last captured FRV. For primary loops in a cascade control system, the extreme output must match up with the set point limits of the secondary loop and the FRV *is* a set point of the secondary loop. It is necessary to keep the units of the process variable and output consistent with the process integrating gain. If process integrating gain is expressed in %/sec/%, then the process variable and output must both be in % [15].

***An adaptive controller can identify the process gain and dead time that are essential for improving the evaluation, performance, and monitoring of optimal switching.***

For integrating processes and $CO_x > FRV$:

$$FRV = CO_x - [(CV_n - CV_o) / DT] / K_i \tag{3-24}$$

where:

| | | |
|---|---|---|
| $FRV$ | = | final resting value (%) |
| $K_i$ | = | integrating process gain (%/sec/%) |
| $CO_x$ | = | controller output at extreme allowed by process (%) |
| $CV_n$ | = | new CV (%) |
| $CV_o$ | = | old CV (output of dead time block) (%) |
| $DT$ | = | DT block dead time (sec) |

With a little ingenuity, similar equations can be developed for estimating the FRV of self-regulating processes based on an identified process gain. These equations can be put on line in the observation mode to see how well they estimate the FRV before you actually use the FRV for practice 4. If the FRV is too variable and cannot be accurately captured or calculated, it is best to revert to the second practice. The second practice depends more heavily on the controller's tuning and in particular on the relative amount of proportional and reset action. This is because the tuning is responsible not just for correcting the FRV but for taking the output all the way from its extreme to the FRV. It is important that gain dominates reset action in the approach to set point. Proportional action must kick the output to the allowable extreme and then back it off as the CV approaches set point. This is despite the effect of the reset action, which works to force the output to its limit until the CV crosses set point [15].

The optimal switching technique based on the CV's rate of change is ideally suited for an integrating or ramping process. However, it works well for self-regulating processes in which the fastest possible approach to set point is desired. It also reduces the dependency on tuning since the PID only has to correct for errors in the dead time and the FRV [15].

## Exercises

3-1. Why does a valve set at a fixed position set by operations or sequences to deliver a flow on a process flow diagram result in excessive process variability?

3-2. Why are controller gains for vessel temperature control not as high as permitted by the large process time constant to dead time ratio?

3-3. Why do PID structures with proportional mode on error provide a faster set point response?

3-4. For the ISA standard form of the PID, what is the limitation on the rate time setting?

3-5. How does the dynamic reset limit improve loop stability?

3-6. Why does the open loop response of a vessel temperature loop ramp like an integrator?

3-7. To reduce the slow rolling oscillations in an integrating process from too low of a reset time setting, what should be done to the gain setting?

3-8. What are the advantages of software that identifies the process dynamics for various modes and ranges of operation?

3-9. For a slow integrating type of response, what can be done to make the set point response as fast as possible?

## References

1. McMillan, Gregory K., *A Funny Thing Happened on the Way to the Control Room*. Reprint via ProQuest UMI "Books on Demand", 1989.

2. McMillan, Gregory K., *Good Tuning: A Pocket Guide*. 2d ed. ISA, 2005.

3. McMillan, Gregory K., *Tuning and Control Loop Performance*, 3d ed. ISA, 1992.

4. Åstrom, Karl, and Hägglund, Tore, *Advanced PID Control*. ISA, 2006.

5. McMillan, Gregory K., and Weiner, Stan, "Control Mythology." *Control* (April 2006).

6. Shinskey, F. G., *Feedback Controllers for the Process Industries*. McGraw-Hill, 1994.