3

PID Control

3.1 Introduction

The PID controller is by far the most common control algorithm. Most feedback loops are controlled by this algorithm or minor variations of it. It is implemented in many different forms, as a stand-alone controller or as a part of a DDC (Direct Digital Control) package or a hierarchical distributed process control system. Many thousands of instrument and control engineers worldwide are using such controllers in their daily work. The PID algorithm can be approached from many different directions. It can be viewed as a device that can be operated with a few rules of thumb, but it can also be approached analytically.

This chapter gives an introduction to PID control. The basic algorithm and various representations are presented in detail. A description of the properties of the controller in a closed loop based on intuitive arguments is given. The phenomenon of reset windup, which occurs when a controller with integral action is connected to a process with a saturating actuator, is discussed, including several methods to avoid it. Filters to reduce noise influence and means to improve set-point responses are also provided.

Implementation aspects of the PID controller are presented in Chapter 13.

3.2 The PID Controller

The "textbook" version of the PID algorithm can be described as:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\,\frac{de(t)}{dt}\right), \tag{3.1}$$

where u is the control signal and e is the control error ($e = y_{sp} - y$). The control signal is thus a sum of three terms: the P-term (which is proportional to the error), the I-term (which is proportional to the integral of the error), and the D-term (which is proportional to the derivative of the error). The controller

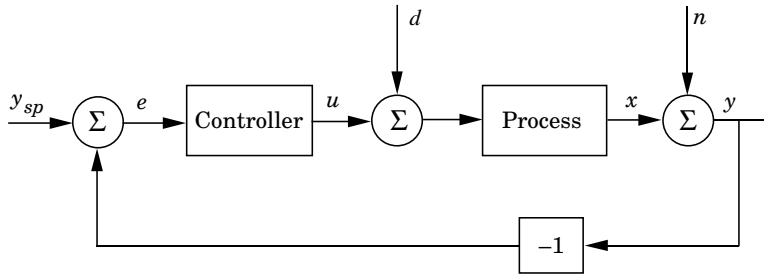


Figure 3.1 Block diagram of a simple feedback loop.

parameters are proportional gain K, integral time T_i, and derivative time T_d.

Proportional Action

In the case of pure proportional control, the control law of Equation 3.1 reduces to

$$u(t) = Ke(t) + u_b. \tag{3.2}$$

The control action is simply proportional to the control error. The variable u_b is a bias or a reset. When the control error e is zero, the control variable takes the value $u(t) = u_b$. Bias u_b is often fixed to $(u_{\max} + u_{\min})/2$, but can sometimes be adjusted manually so that the stationary control error is zero at a given set point.

Static Analysis Several properties of proportional control can be understood by the following argument, which is based on pure static considerations. Consider the simple feedback loop, shown in Figure 3.1, and composed of a process and a controller. Assume that the controller has proportional action and that the process is modeled by the static model

$$x = K_p(u + d), \tag{3.3}$$

where x is the process variable, u is the control variable, d is a load disturbance, and K_p is the static process gain. The following equations are obtained from the block diagram.

$$\begin{aligned} y &= x + n \\ x &= K_p(u + d) \\ u &= K(y_{sp} - y) + u_b. \end{aligned} \tag{3.4}$$

where n is measurement noise. Elimination of intermediate variables gives the following relation between process variable x, set point y_{sp}, load disturbance d, and measurement noise n:

$$x = \frac{KK_p}{1 + KK_p}(y_{sp} - n) + \frac{K_p}{1 + KK_p}(d + u_b). \tag{3.5}$$

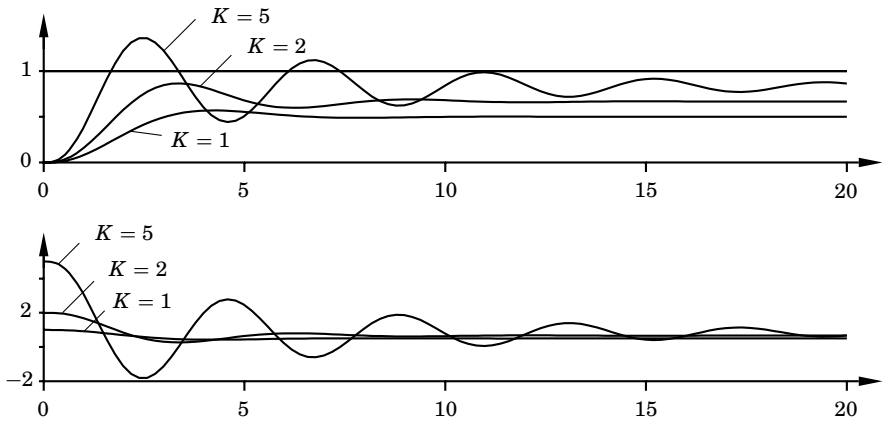


Figure 3.2 Simulation of a closed-loop system with proportional control. The process transfer function is $G(s) = (s+1)^{-3}$. The upper diagram shows set point $y_{sp} = 1$ and process output y for different values of controller gain K. The lower diagram shows control signal u for different controller gains.

Product KK_p is a dimensionless number called the *loop gain*. Several interesting properties of the closed-loop system can be read from Equation 3.5. First assume that n and u_b are zero. Then the loop gain should be high in order to ensure that process output x is close to set point y_{sp}. A high value of controller gain K also makes the system insensitive to load disturbance d. However, if n is nonzero, it follows from Equation 3.5 that measurement noise n influences the process output in the same way as set point y_{sp}. To avoid making the system sensitive to measurement noise, the loop gain should not be made too large. Further, the controller bias u_b influences the system in the same way as a load disturbance. It is, therefore, obvious that the design of the loop gain is a trade-off between different control objectives and that there is no simple answer to what loop gain is the best. This will depend on which control objective is the most important.

It also follows from Equation 3.5 that there will normally be a steady-state error with proportional control. This can be deduced intuitively from the observation following from Equation 3.4 that the control error is zero only when $u = u_b$ in stationarity. The error, therefore, can be made zero at a given operating condition by a proper choice of the controller bias u_b.

The static analysis given above is based on the assumption that the process can be described by a static model. This leaves out some important properties of the closed-loop system dynamics. The most important one is that the closed-loop system will normally be unstable for high-loop gains if the process dynamics are considered. In practice, the maximum loop gain is thus determined by the process dynamics. One way to describe process dynamics leads to descriptions like Equation 3.3 where the process gain is frequency-dependent. This was discussed in Chapter 2.

A typical example of proportional control is illustrated in Figure 3.2. The figure shows the behavior of the process output and the control signal after

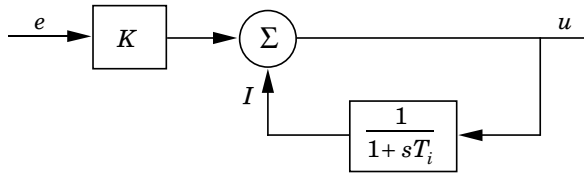


Figure 3.3 Implementation of integral action as positive feedback around a lag.

a step change in the set point. The steady-state error can be computed from Equation 3.5. The bias term u_b, the load d, and the noise n are all zero in the simulation. With a controller gain $K = 1$ and a static process gain $K_p = 1$, the error is therefore 50 percent. The figure shows that the steady-state error decreases with increasing controller gain as predicted by Equation 3.5. Notice also that the response becomes more oscillatory with increasing controller gain. This is due to the process dynamics.

Integral Action

The main function of the integral action is to make sure that the process output agrees with the set point in steady state. With proportional control, there is normally a control error in steady state. With integral action, a small positive error will always lead to an increasing control signal, and a negative error will give a decreasing control signal no matter how small the error is.

The following simple argument shows that the steady-state error will always be zero with integral action. Assume that the system is in steady state with a constant control signal (u_0) and a constant error (e_0). It follows from Equation 3.1 that the control signal is then given by

$$u_0 = K\left(e_0 + \frac{e_0}{T_i}\,t\right).$$

As long as $e_0 \neq 0$, this clearly contradicts the assumption that the control signal u_0 is constant. A controller with integral action will always give zero steady-state error.

Integral action can also be visualized as a device that automatically resets the bias term u_b of a proportional controller. This is illustrated in the block diagram in Figure 3.3, which shows a proportional controller with a reset that is adjusted automatically. The adjustment is made by feeding back a signal, which is a filtered value of the output, to the summing point of the controller. This was actually one of the early inventions of integral action, or "automatic reset," as it was also called. The implementation shown in Figure 3.3 is still used by many manufacturers.

Simple calculations show that the controller in Figure 3.3 gives the desired results. The following equation is obtained from the block diagram:

$$u = Ke + I = T_i\,\frac{dI}{dt} + I.$$

Hence,

$$T_i\,\frac{dI}{dt} = Ke,$$

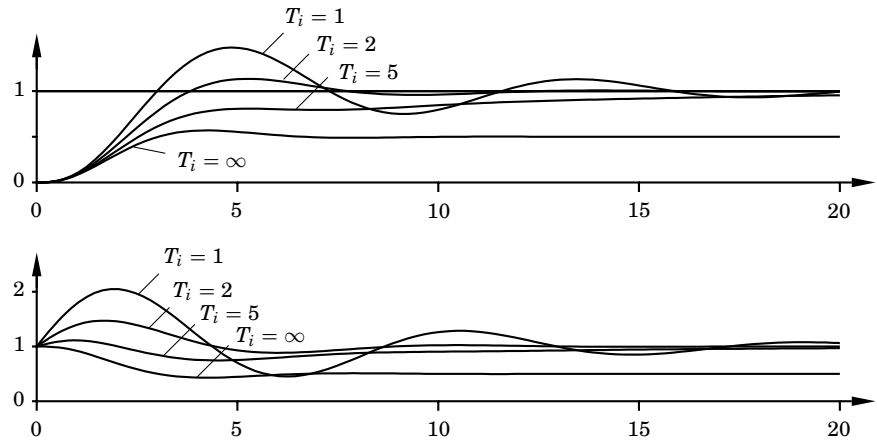


Figure 3.4 Simulation of a closed-loop system with proportional and integral control. The process transfer function is $G(s) = (s+1)^{-3}$, and the controller gain is $K = 1$. The upper diagram shows set point $y_{sp} = 1$ and process output y for different values of integral time T_i. The lower diagram shows control signal u for different integral times.

and we find that

$$u = K\left(e + \frac{1}{T_i}\int e(\tau)d\tau\right),$$

which is a PI controller.

The properties of integral action are illustrated in Figure 3.4, which shows a simulation of a system with PI control. The proportional gain is constant, $K = 1$, and the integral time is changed. The case $T_i = \infty$ corresponds to pure proportional control. This case is identical to the case $K = 1$ in Figure 3.2, where the steady-state error is 50 percent. The steady-state error is removed when T_i has finite values. For large values of the integration time, the response creeps slowly towards the set point. The approach is approximately exponential with time constant T_i/KK_p. The approach is faster for smaller values of T_i but is also more oscillatory.

Derivative Action

The purpose of the derivative action is to improve the closed-loop stability. The instability mechanism can be described intuitively as follows. Because of the process dynamics, it will take some time before a change in the control variable is noticeable in the process output. Thus, the control system will be late in correcting for an error. The action of a controller with proportional and derivative action may be interpreted as if the control is made proportional to the *predicted* process output, where the prediction is made by extrapolating the error by the tangent to the error curve (see Figure 3.5). The basic structure of a PD controller is

$$u(t) = K\left(e(t) + T_d\frac{de(t)}{dt}\right).$$

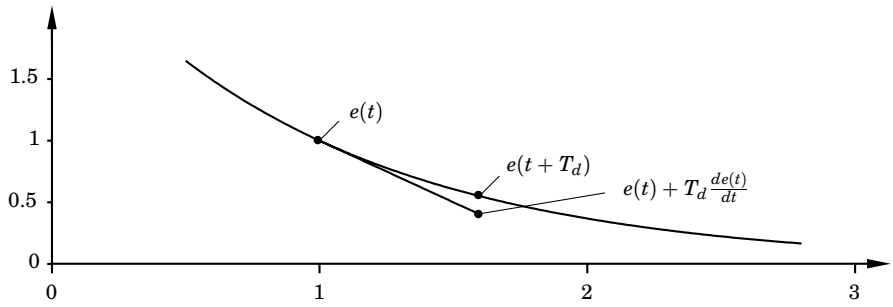


Figure 3.5 Interpretation of derivative action as predictive control, where the prediction is obtained by linear extrapolation.

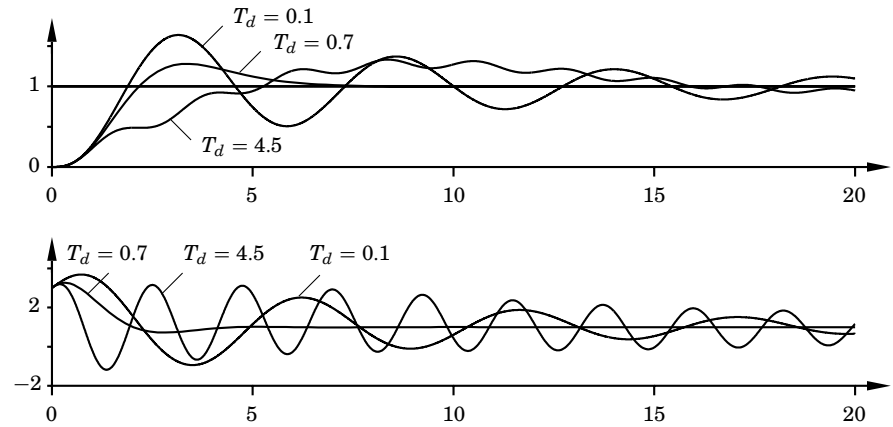


Figure 3.6 Simulation of a closed-loop system with proportional, integral, and derivative control. The process transfer function is $G(s) = (s+1)^{-3}$, the controller gain is $K = 3$, and the integral time is $T_i = 2$. The upper diagram shows set point $y_{sp} = 1$ and process output y for different values of derivative time T_d. The lower diagram shows control signal u for different derivative times.

A Taylor series expansion of $e(t + T_d)$ gives

$$e(t + T_d) \approx e(t) + T_d \frac{de(t)}{dt}.$$

The control signal is thus proportional to an estimate of the control error at time T_d ahead, where the estimate is obtained by linear extrapolation. The properties of derivative action are illustrated in Figure 3.6, which shows a simulation of a system with PID control.

Controller gain and integration time are kept constant, $K = 3$ and $T_i = 2$, and derivative time T_d is changed. For $T_d = 0$ we have pure PI control. The closed-loop system is oscillatory with the chosen parameters. Initially damping increases with increasing derivative time but decreases again when derivative time becomes too large.

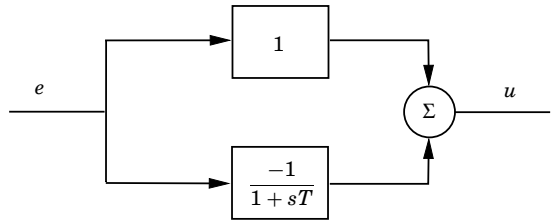


Figure 3.7 Classical implementation of derivative action.

Classical Implementation of Derivative Action

In Figure 3.3 it was shown that integral action originally was implemented by positive feedback around a first-order lag. Derivative action was also originally implemented using a first-order lag as is shown by the block diagram in Figure 3.7. The Laplace transform of the output is given by

$$U(s) = \left(1 - \frac{1}{1+sT}\right)E(s) = \frac{sT}{1+sT}E(s). \tag{3.6}$$

The system thus has the transfer function $G(s) = sT/(1+sT)$. Notice that filtering is obtained automatically with this implementation.

Alternative Representations

The PID algorithm given by Equation 3.1 can be represented by the transfer function

$$C(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right). \tag{3.7}$$

A slightly different version is most common in commercial controllers. This controller is described by

$$C'(s) = K'\left(1 + \frac{1}{sT_i'}\right)(1 + sT_d'). \tag{3.8}$$

The two controller structures are presented in block diagram form in Figure 3.8. The controller given by Equation 3.7 is called non-interacting, and the one given by Equation 3.8 interacting. The reason for this nomenclature is that in the controller (3.7) the integral time T_i does not influence the derivative part, and the derivative time T_d does not influence the integral part. The parts are thus non-interacting. In the interacting controller, the derivative time T_d' does influence the integral part. Therefore, the parts are interacting.

The interacting controller (3.8) can always be represented as a non-interacting controller whose coefficients are given by

$$\begin{aligned} K &= K'\frac{T_i' + T_d'}{T_i'} \\ T_i &= T_i' + T_d' \\ T_d &= \frac{T_i'T_d'}{T_i' + T_d'}. \end{aligned} \tag{3.9}$$

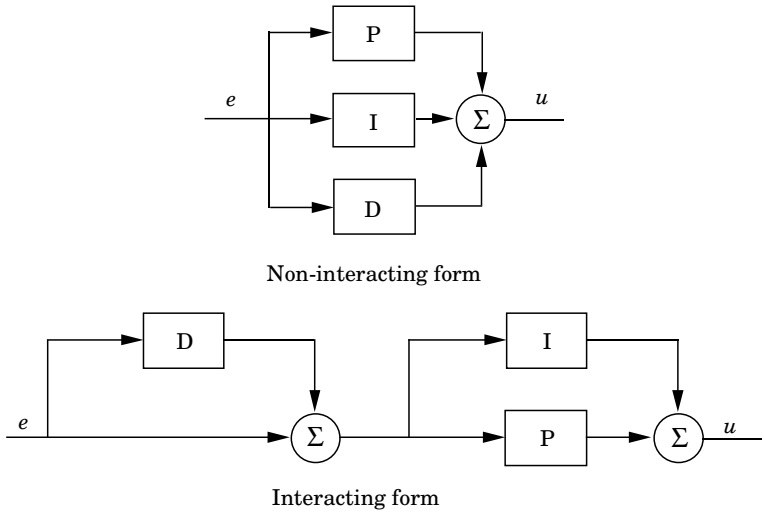


Figure 3.8 Interacting and non-interacting form of the PID algorithm.

An interacting controller of the form (3.8) that corresponds to a non-interacting controller can be found only if

$$T_i \geq 4T_d.$$

Then,

$$\begin{aligned} K' &= \frac{K}{2}\left(1+\sqrt{1-4T_d/T_i}\right) \\ T_i' &= \frac{T_i}{2}\left(1+\sqrt{1-4T_d/T_i}\right) \\ T_d' &= \frac{T_i}{2}\left(1-\sqrt{1-4T_d/T_i}\right). \end{aligned} \tag{3.10}$$

The non-interacting controller given by Equation 3.7 is more general, and we will use that in the future. It is, however, sometimes claimed that the interacting controller is easier to tune manually.

There is also a historical reason for preferring the interacting controller. Early pneumatic controllers were easier to build using the interacting form. See Chapter 13. When the controller manufacturers changed technology from pneumatic to analog electric and, finally, to digital technique, they kept the interactive form. Therefore, the interacting form is most common among single-loop controllers.

It is important to keep in mind that different controllers may have different structures. It means that if a controller in a certain control loop is replaced by another type of controller, the controller parameters may have to be changed. Note, however, that the interacting and the non-interacting forms are different only when both the I and the D parts of the controller are used. If we only use the controller as a P, PI, or PD controller, the two forms are equivalent. Yet

another representation of the PID algorithm is given by

$$C''(s) = k + \frac{k_i}{s} + sk_d. \tag{3.11}$$

The parameters are related to the parameters of standard form through

$$k = K \qquad k_i = \frac{K}{T_i} \qquad k_d = KT_d. \tag{3.12}$$

The representation (3.11) is equivalent to the standard form, but the parameter values are quite different. This may cause great difficulties for anyone who is not aware of the differences, particularly if parameter $1/k_i$ is called integral time and k_d derivative time. The form given by Equation 3.11 is often useful in analytical calculations because the parameters appear linearly. The representation also has the advantage that it is possible to obtain pure proportional, integral, or derivative action by finite values of the parameters.

Summarizing, we have thus found that there are three different forms of the PID controller.

- The standard or non-interacting form given by Equation 3.7.
- The series or interacting form given by Equation 3.8.
- The parallel form given by Equation 3.11.

The standard form is sometimes called the ISA algorithm, or the ideal algorithm. The proportional, integral, and derivative actions are non-interacting in the time domain. This algorithm admits complex zeros, which is useful when controlling systems with oscillatory modes.

The series form is also called the classical form. This representation is obtained naturally when a controller is implemented as an analog device based on a pneumatic force balance system. The name classical reflects this. The series form has an attractive interpretation in the frequency domain because the zeros correspond to the inverse values of the derivative and integral times. All zeros of the controller are real. Pure integral or proportional action cannot be obtained with finite values of the controller parameters.

The parallel form is the most general form because pure proportional or integral action can be obtained with finite parameters. The controller can also have complex zeros. In this way it is the most flexible form. However, it is also the form where the parameters have little physical interpretation.

Summary

The PID controller has three terms. The proportional term P corresponds to proportional control. The integral term I gives a control action that is proportional to the time integral of the error. This ensures that the steady-state error becomes zero. The derivative term D is proportional to the time derivative of the control error. This term allows prediction of the future error. There are many variations of the basic PID algorithm that will substantially improve its performance and operability. They are discussed in the following sections.

3.3 Filtering the Derivative

A drawback with derivative action is that an ideal derivative has very high gain for high-frequency signals. This means that high-frequency measurement noise will generate large variations of the control signal. To see this we consider a measured output

$$y = \sin t + a \sin \omega t,$$

where the first term is the useful signal and the second term represents noise. The ratio of noise to signal is thus a. The derivative term of the controller is then

$$D = KT_d \frac{dy}{dt} = KT_d\big(\cos t + a\omega \cos \omega t\big). \tag{3.13}$$

The amplitude of the signal is KT_d, and the amplitude of the noise is $KT_d a\omega$. The ratio of noise and signal is $a\omega$. This can be arbitrarily large even if a is small if the frequency is sufficiently high. The effect of measurement noise can to some extent be eliminated by implementing the derivative term as

$$D = -\frac{sKT_d}{1 + sT_d/N}\, Y. \tag{3.14}$$

This can be interpreted as an ideal derivative that is filtered using a first-order system with the time constant T_d/N. For small s the transfer function is approximately sKT_d, and for large s it is equal to KN. The approximation acts as a derivative for low-frequency signal components, and the high-frequency gain is limited to KN. High-frequency measurement noise is amplified at most by a factor KN. Typical values of N are 2 to 20. Notice that the implementation of the derivative given in Figure 3.7 automatically gives a limitation of the high-frequency gain; see Equation 3.6.

The transfer function of a PID controller with a filtered derivative is

$$C(s) = K\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d/N}\right). \tag{3.15}$$

The high-frequency gain of the controller is $K(1 + N)$. High frequency measurement noise can thus generate significant variations in the control signal. It is therefore advantageous to use heavier filtering.

Instead of filtering only the derivative it is possible to filter the measured signal and apply the filtered signal to an ideal PID controller. The equivalent controller transfer function is

$$C_{eq} = C(s)G_f(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right)\frac{1}{1 + sT_f + (sT_f)^2/2}, \tag{3.16}$$

when a second-order filter with relative damping $\zeta = 1/\sqrt{2}$ is used. The filter-time constant T_f is typically chosen as T_i/N for PI control or as T_d/N for PID control, where N ranges from 2 to 20.

It follows from (3.16) that the controller gain goes to zero for high frequencies. This property, which is called *high frequency roll-off*, guarantees that high-frequency measurement noise will not generate large control signals. High-frequency roll-off also increases the robustness of the closed loop system.

3.4 Set-Point Weighting

The control system in Figure 3.1 is called a system with *error feedback* because the controller acts on the error, which is the difference between the set point and the output. A more flexible structure is obtained by treating the set point and the process output separately. A PID controller of this form is given by

$$u(t) = K\left(e_p + \frac{1}{T_i}\int_0^t e(s)ds + T_d\,\frac{de_d}{dt}\right), \tag{3.17}$$

where the error in the proportional part is

$$e_p = by_{sp} - y, \tag{3.18}$$

and the error in the derivative part is

$$e_d = cy_{sp} - y. \tag{3.19}$$

The error in the integral part must be the true control error

$$e = y_{sp} - y,$$

in order to avoid steady-state control errors. The controllers obtained for different values of b and c will respond to load disturbances and measurement noise in the same way. The response to set-point changes will depend on the values of b and c, which are called *set-point weights*.

The properties of a system where the controller has set-point weighting is illustrated in Figure 3.4, which shows the response of a PID controller to set-point changes, load disturbances, and measurement errors for different values of b. The figure shows clearly the effect of changing b. The overshoot for set-point changes is smallest for $b = 0$, which is the case where the reference is only introduced in the integral term, and increases with increasing b. Notice that a simulation like the one in Figure 3.4 is useful in order to give a quick assessment of the responses of a closed-loop system to set-point changes, load disturbances, and measurement errors.

The parameter c is normally chosen equal to zero to avoid large transients in the control signal due to sudden changes in the set point. An exception is when the controller is the secondary controller in a cascade coupling (see Section 12.4). In this case, the set-point changes smoothly because it is given by the primary controller output. Notice that if the integral action is implemented with positive feedback around a lag as in Figure 3.3, the parameter b is equal to one.

The controller with $b = 0$ and $c = 0$ is sometimes called an I-PD controller, and the controller with $b = 1$ and $c = 0$ is sometimes called a PI-D controller. We prefer to stick to the generic use of PID and give the parameters b and c, thereby making a small contribution towards reduction of three-letter abbreviations.

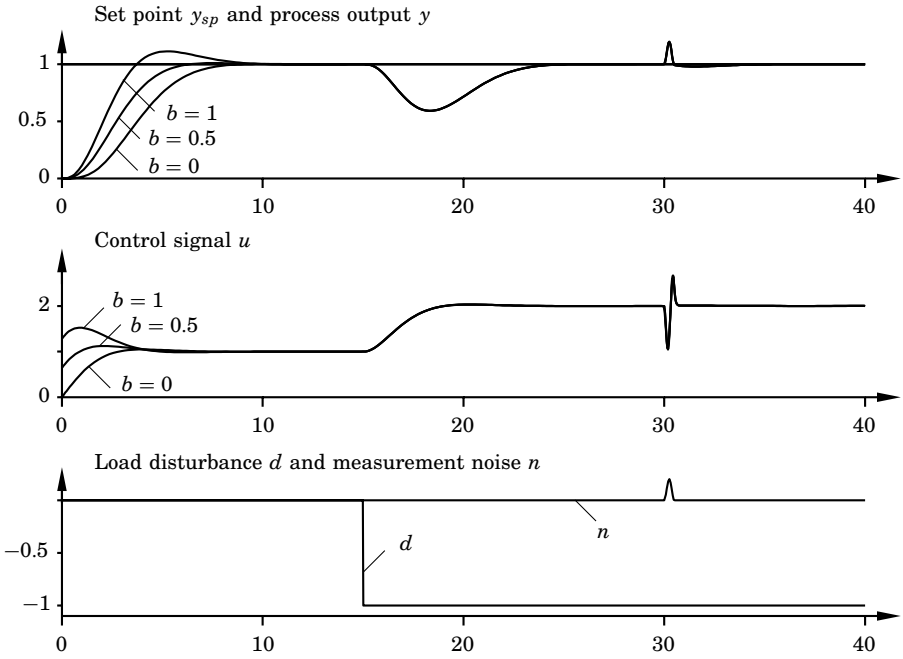


Figure 3.9 The response to set-point changes, load disturbances, and measurement errors for different values of set-point weighting b.

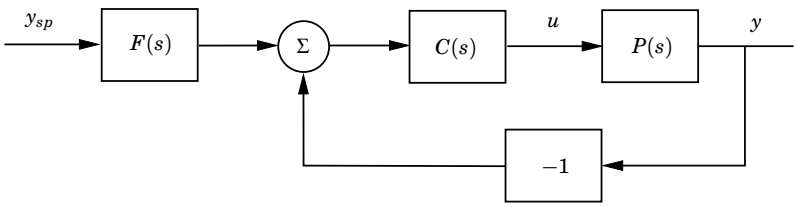


Figure 3.10 Block diagram of a simple feedback loop with a PID controller having a two-degree-of-freedom structure. The transfer function $P(s)$ is the process transfer function.

In the block diagram in Figure 3.1, the controller output is generated from the error $e = y_{sp} - y$. Notice that this diagram is no longer valid when the control law given by Equation 3.17 and the error definitions (3.18) and (3.19) are used. A block diagram for a system with PID control is instead given by Figure 3.10 where the transfer function $C(s)$ is given by (3.7) and

$$F(s) = \frac{b + \frac{1}{sT_i} + scT_d}{1 + \frac{1}{sT_i} + sT_d} = \frac{cT_iT_ds^2 + bsT_i + 1}{T_iT_ds^2 + sT_i + 1}. \tag{3.20}$$

System with Two Degrees of Freedom

In general, a control system has many different requirements. It should have good transient response to set-point changes, and it should reject load disturbances and measurement noise. For a system with error feedback only, an attempt is made to satisfy all demands with the same mechanism. Such systems are called one-degree-of-freedom systems.

The system shown in Figure 3.10 is said to have two degrees of freedom because the signal path from the set point to the control signal is different from the signal path from measured value to control signal.

There are many possible configurations of systems with two degrees of freedom. The system shown in Figure 3.10 is only one alternative. An extended use of structures with two degrees of freedom is a very natural extension of the PID controller. The key idea is to let the controller C be a PI or a PID controller but to use more flexible feedforward than the standard PID controller permits. This will be discussed more fully in Chapter 5.

3.5 Integrator Windup

Although many aspects of a control system can be understood based on linear theory, some nonlinear effects must be accounted for. All actuators have limitations: a motor has limited speed, a valve cannot be more than fully opened or fully closed, etc. For a control system with a wide range of operating conditions, it may happen that the control variable reaches the actuator limits. When this happens the feedback loop is broken and the system runs as an open loop because the actuator will remain at its limit independently of the process output. If a controller with integrating action is used, the error may continue to be integrated if the algorithm is not properly designed. This means that the integral term may become very large or, colloquially, it "winds up." It is then required that the error has opposite sign for a long period before things return to normal. The consequence is that any controller with integral action may give large transients when the actuator saturates.

EXAMPLE 3.1—ILLUSTRATION OF INTEGRATOR WINDUP

Figure 3.11 shows the control signal, the measurement signal, and the set point in a case where the control signal becomes saturated. After the first set-point change, the control signal increases to its upper limit $u_{\max}$. This control signal is not large enough to eliminate the control error. Therefore, the integral of the control error, and the integral part of the control signal, increases. Since the desired control signal u increases, there is a difference between the desired control signal and the true control signal u_{out}.

Figure 3.11 shows what happens when after a certain time the set point is lowered to a level where the controller is able to eliminate the control error. Since the sign of the control error becomes negative, the control signal starts to decrease, but since the desired control signal u is above the limit $u_{\max}$, the true control signal u_{out} is stuck at the limit for a while and the response becomes delayed. □

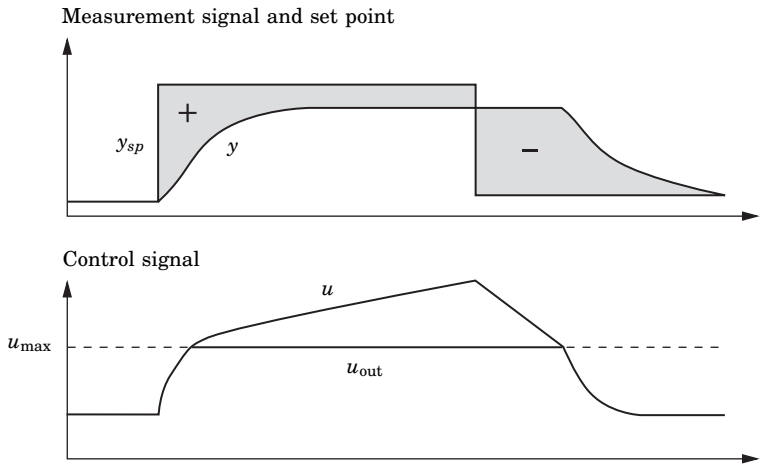


Figure 3.11 Illustration of integrator windup.

The following example shows some other effects that may occur due to integrator windup when the process is unstable.

EXAMPLE 3.2—ILLUSTRATION OF INTEGRATOR WINDUP
The windup phenomenon is illustrated in Figure 3.12, which shows control of an integrating process with a PI controller. The initial set-point change is so large that the actuator saturates at the high limit. The integral term increases initially because the error is positive; it reaches its largest value at time $t = 10$ when the error goes through zero. The output remains saturated at this point because of the large value of the integral term. It does not leave the saturation limit until the error has been negative for a sufficiently long time to let the integral part come down to a small level. Notice that the control signal bounces between its limits several times. The net effect is a large overshoot and a damped oscillation where the control signal flips from one extreme to the other. The output finally comes so close to the set point that the actuator does not saturate. The system then behaves linearly and settles. □

Integrator windup may occur in connection with large set-point changes, or it may be caused by large disturbances or equipment malfunctions. Windup can also occur when selectors are used so that several controllers are driving one actuator. In cascade control, windup may occur in the primary controller when the secondary controller is switched to manual mode, uses its local set point, or if its control signal saturates. See Section 12.4.

The phenomenon of windup was well known to manufacturers of analog controllers, who invented several tricks to avoid it. They were described under labels like preloading, batch unit, etc. Although the problem was well understood, there were often limits imposed because of the analog implementations. The ideas were often kept as trade secrets and not much spoken about. The problem of windup was rediscovered when controllers were implemented digitally and several methods to avoid windup were presented in the literature. In the following section we describe several of the ideas.

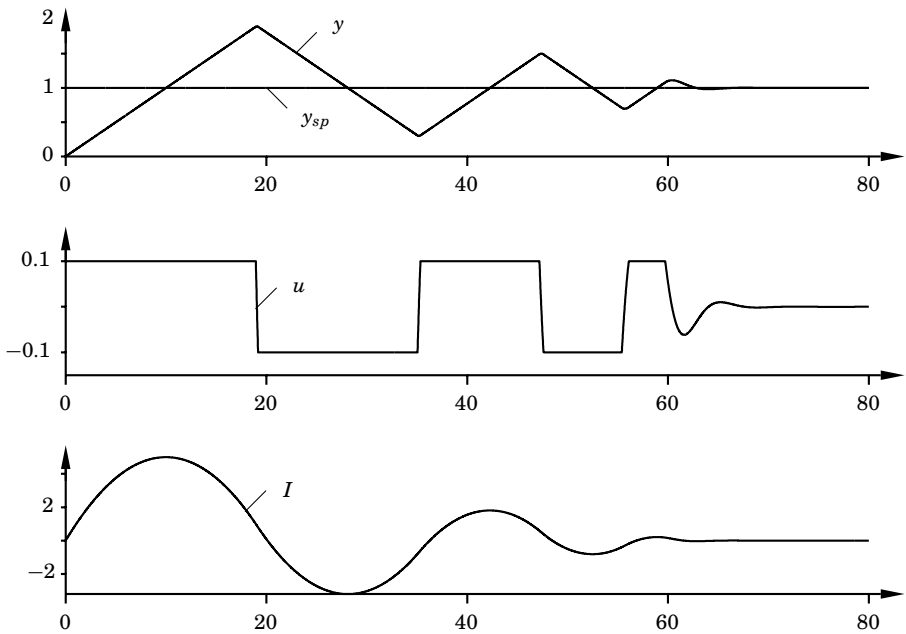


Figure 3.12 Illustration of integrator windup. The diagrams show process output y, set point y_{sp}, control signal u, and integral part I.

Set-Point Limitation

One way to try to avoid integrator windup is to introduce limiters on the set-point variations so that the controller output will never reach the actuator bounds. This often leads to conservative bounds and limitations on controller performance. Further, it does not avoid windup caused by disturbances.

Incremental Algorithms

In the early phases of feedback control, integral action was integrated with the actuator by having a motor drive the valve directly. In this case, windup is handled automatically because integration stops when the valve stops. When controllers were implemented by analog techniques, and later with computers, many manufacturers used a configuration that was an analog of the old mechanical design. This led to the so-called velocity algorithms, discussed in Chapter 13. In this algorithm the rate of change of the control signal is first computed and then fed to an integrator. In some cases this integrator is a motor directly connected to the actuator. In other cases the integrator is implemented internally in the controller. With this approach it is easy to handle mode changes and windup. Windup is avoided by inhibiting the integration whenever the output saturates. This method is equivalent to back-calculation, which is described below. If the actuator output is not measured, a model that computes the saturated output can be used. It is also easy to limit the rate of change of the control signal.

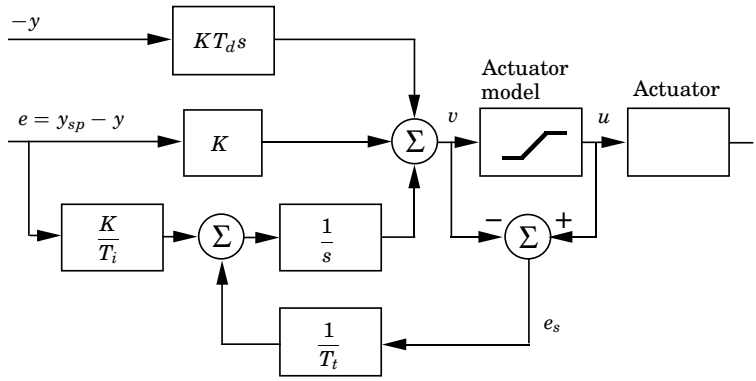


Figure 3.13 PID controller with anti-windup.

Back-Calculation and Tracking

Back-calculation works as follows. When the output saturates, the integral term in the controller is recomputed so that its new value gives an output at the saturation limit. It is advantageous not to reset the integrator instantaneously but dynamically with a time constant T_t.

Figure 3.13 shows a block diagram of a PID controller with anti-windup based on back-calculation. The system has an extra feedback path that is generated by measuring the actual actuator output, or the output of a mathematical model of the saturating actuator, and forming an error signal (e_s) as the difference between the output of the controller (v) and the actuator output (u). The signal e_s is fed to the input of the integrator through gain $1/T_t$. The signal is zero when there is no saturation. Thus, it will not have any effect on the normal operation when the actuator does not saturate. When the actuator saturates, the signal e_s is different from zero. The normal feedback path around the process is broken because the process input remains constant. There is, however, a feedback path around the integrator. Because of this, the integrator output is driven towards a value such that the integrator input becomes zero. The integrator input is

$$\frac{1}{T_t}\, e_s + \frac{K}{T_i}\, e,$$

where e is the control error. Hence,

$$e_s = -\frac{KT_t}{T_i}\, e$$

in steady state. Since $e_s = u - v$, it follows that

$$v = u_{\text{lim}} + \frac{KT_t}{T_i}\, e,$$

where u_{lim} is the saturating value of the control variable. Since the signals e and u_{lim} have the same sign, it follows that v is always larger than u_{lim} in

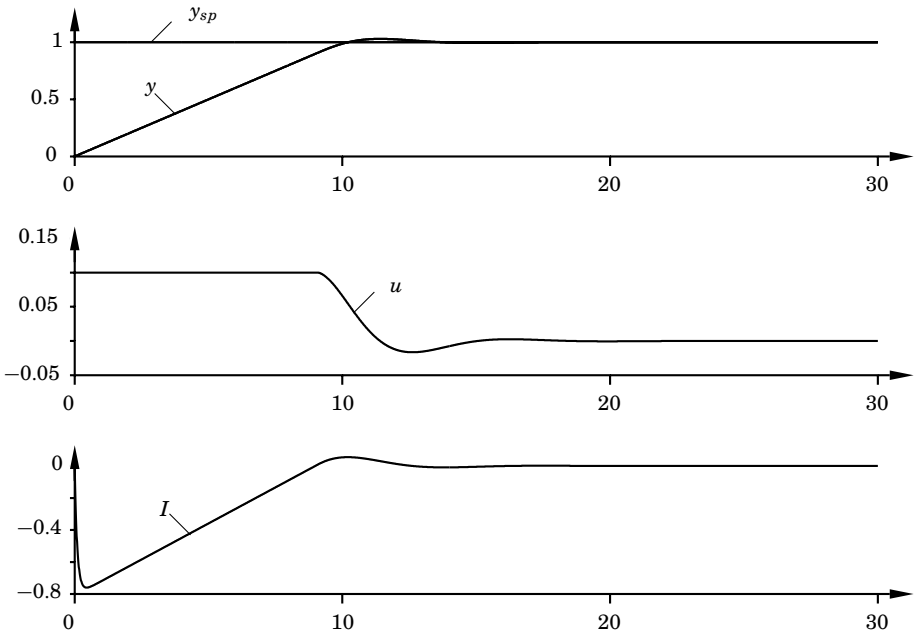


Figure 3.14 Controller with anti-windup applied to the system of Figure 3.12. The diagrams show process output y, set point y_{sp}, control signal u, and integral part I.

magnitude. This prevents the integrator from winding up. The rate at which the controller output is reset is governed by the feedback gain, $1/T_t$, where T_t can be interpreted as the time constant, which determines how quickly the integral is reset. We call this the tracking time constant.

Figure 3.14 shows what happens when a controller with anti-windup is applied to the system simulated in Figure 3.12. Notice that the output of the integrator is quickly reset to a value such that the controller output is at the saturation limit and the integral has a negative value during the initial phase when the actuator is saturated. This behavior is drastically different from that in Figure 3.12, where the integral has a positive value during the initial transient. Also notice the drastic improvement in performance compared to the ordinary PI controller used in Figure 3.12.

The effect of changing the values of the tracking time constant is illustrated in Figure 3.15. From this figure, it may thus seem advantageous to always choose a very small value of the time constant because the integrator is then reset quickly. However, some care must be exercised when introducing anti-windup in systems with derivative action. If the time constant chosen is too small, spurious errors can cause saturation of the output, which accidentally resets the integrator. The tracking time constant T_t should be larger than T_d and smaller than T_i. A rule of thumb that has been suggested is to choose $T_t = \sqrt{T_i T_d}$.

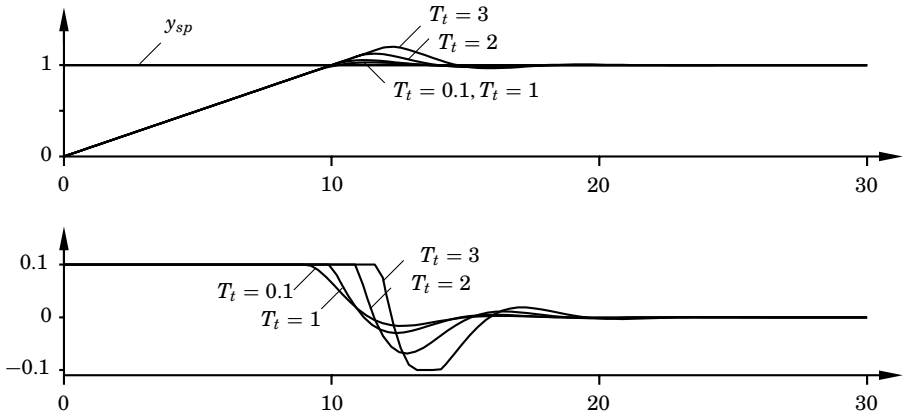


Figure 3.15 The step response of the system in Figure 3.12 for different values of the tracking time constant T_t. The upper curve shows process output y and set point y_{sp}, and the lower curve shows control signal u.

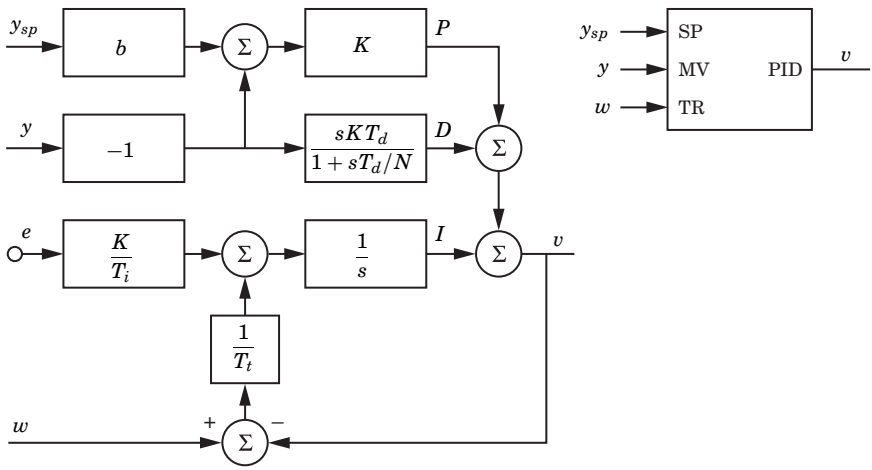


Figure 3.16 Block diagram and simplified representation of PID module with tracking signal.

Controllers with a Tracking Mode

A controller with back-calculation can be interpreted as having two modes: the normal *control mode*, when it operates like an ordinary controller, and a *tracking mode*, when the controller is tracking so that it matches given inputs and outputs. Since a controller with tracking can operate in two modes, we may expect that it is necessary to have a logical signal for mode switching. However, this is not necessary, because tracking is automatically inhibited when the tracking signal w is equal to the controller output. This can be used with great advantage when building up complex systems with selectors and cascade control.

Figure 3.16 shows a PID module with a tracking signal. The module has

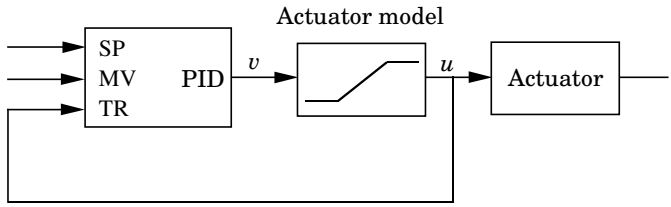


Figure 3.17 Representation of the controller with anti-windup in Figure 3.13 using the basic control module with tracking shown in Figure 3.16.

three inputs: the set point, the measured output, and a tracking signal. The new input TR is called a tracking signal because the controller output will follow this signal. Notice that tracking is inhibited when $w = v$. Using the module the system shown in Figure 3.13 can be presented as shown in Figure 3.17.

The Proportional Band

The notion of proportional band is useful in order to understand the windup effect and to explain schemes for anti-windup. The *proportional band* is an interval such that the actuator does not saturate if the instantaneous value of the process output or its predicted value is in the interval. For PID control without derivative gain limitation, the control signal is given by

$$u = K(by_{sp} - y) + I - KT_d \frac{dy}{dt}. \tag{3.21}$$

Solving for the predicted process output

$$y_p = y + T_d \frac{dy}{dt},$$

gives the proportional band (y_l, y_h) as

$$\begin{aligned} y_l &= by_{sp} + \frac{I - u_{\max}}{K} \\ y_h &= by_{sp} + \frac{I - u_{\min}}{K}. \end{aligned} \tag{3.22}$$

and $u_{\min}$, $u_{\max}$ are the values of the control signal for which the actuator saturates. The controller operates in the linear mode, if the predicted output is in the proportional band. The control signal saturates when the predicted output is outside the proportional band. Notice that the proportional band can be shifted by changing the integral term.

To illustrate that the proportional band is useful in understanding windup, we show the proportional band in Figure 3.18 for the system discussed in Example 3.2. (Compare with Figure 3.12.) The figure shows that the proportional band starts to move upwards because the integral term increases. This implies that the output does not reach the proportional band until it is much larger than the set point. When the proportional band is reached the control signal

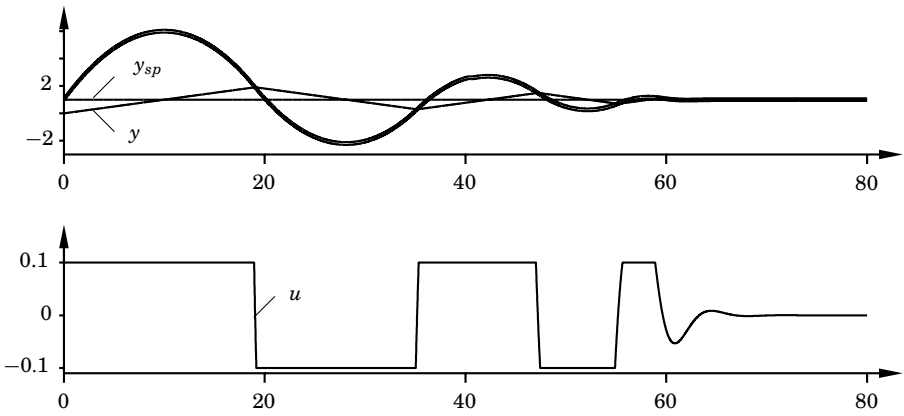


Figure 3.18 The proportional band for the system in Example 3.2. The upper diagram shows process output y and the proportional band. The lower diagram shows control signal u.

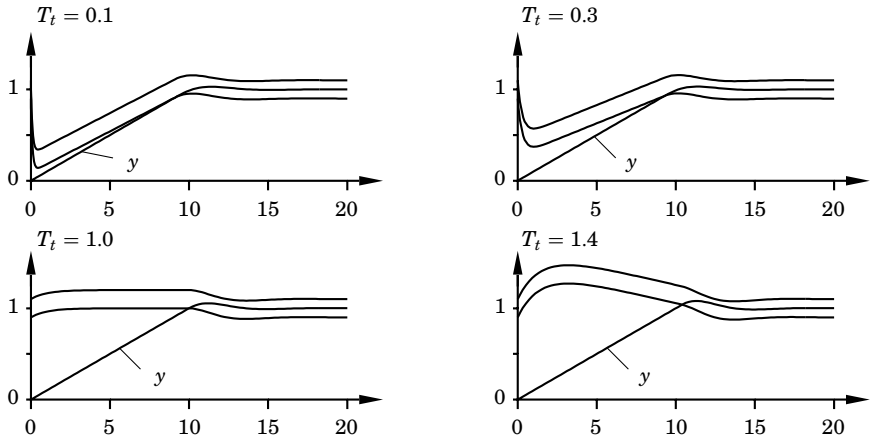


Figure 3.19 The proportional band and the process output y for a system with conditional integration and tracking with different tracking time constants T_t.

decreases rapidly. The proportional band changes so rapidly, however, that the output very quickly moves through the band, and this process repeats several times.

The notion of proportional band helps us to understand several schemes for anti-windup. Figure 3.19 shows the proportional band for the system with tracking for different values of the tracking time constant T_t. The figure shows that the tracking time constant has a significant influence on the proportional band. Because of the tracking, the proportional band is moved closer to the process output. How rapidly it does this is governed by the tracking time constant T_t. Notice that there may be a disadvantage in moving it too rapidly, since the predicted output may then move into the proportional band because of noise and cause the control signal to decrease unnecessarily.

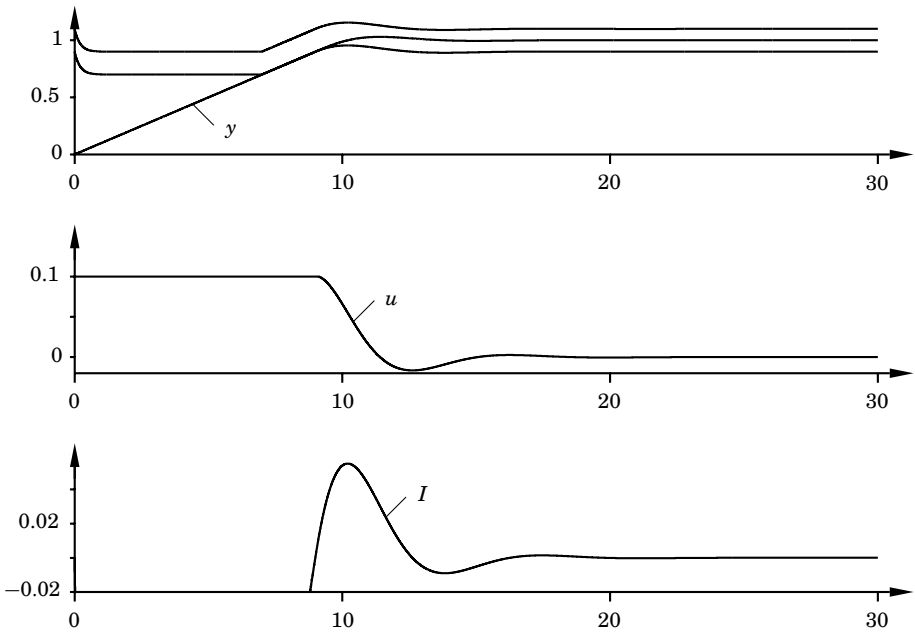


Figure 3.20 Simulation of the system in Example 3.2 with conditional integration. The diagrams show the proportional band, process output y, control signal u, and integral part I.

Conditional Integration

Conditional integration is an alternative to back-calculation or tracking. In this method integration is switched off when the control is far from steady state. Integral action is thus only used when certain conditions are fulfilled; otherwise the integral term is kept constant. The method is also called integrator clamping.

The conditions when integration is inhibited can be expressed in many different ways. Figure 3.20 shows a simulation of the system in Example 3.2 with conditional integration such that the integral term is kept constant during saturation. A comparison with Figure 3.19 shows that, in this particular case, there is very little difference in performance between conditional integration and tracking. The different wind-up schemes do, however, move the proportional bands differently.

A few different switching conditions are now considered. One simple approach is to switch off integration when the control error is large. Another approach is to switch off integration during saturation. Both these methods have the disadvantage that the controller may get stuck at a non-zero control error if the integral term has a large value at the time of switch-off.

A method without this disadvantage is the following. Integration is switched off when the controller is saturated *and* the integrator update is such that it causes the control signal to become more saturated. Suppose, for example, that the controller becomes saturated at the upper saturation. Integration is then

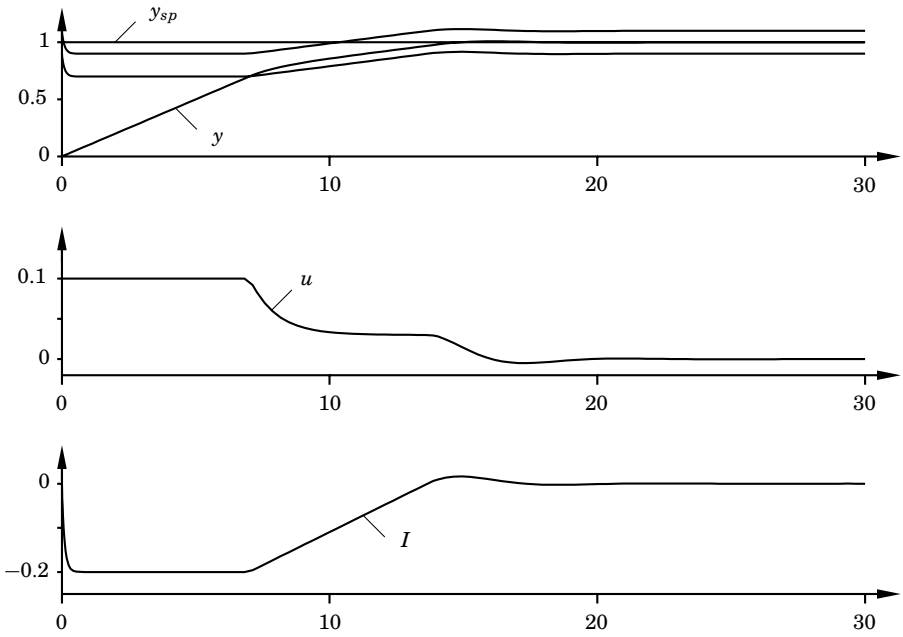


Figure 3.21 Adjustment of the proportional band using cut-back parameters. The diagrams show the proportional band, set point y_{sp}, process output y, control signal u, and integral part I.

switched off if the control error is positive, but not if it is negative.

Some conditional integration methods are intended mainly for startup of batch processes, when there may be large changes in the set point. One particular version, used in temperature control, sets the proportional band outside the set point when there are large control deviations. The offset can be used to adjust the transient response obtained during startup of the process. The parameters used are called cutback or preload (see Figure 3.21). In this system the proportional band is positioned with one end at the set point and the other end towards the measured value when there are large variations. These methods may give windup during disturbances.

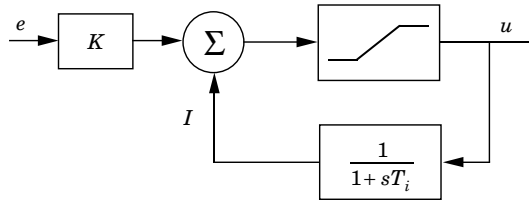
Series Implementation

In Figure 3.3, we showed a special implementation of a controller in interacting form. To avoid windup in this controller we can incorporate a model of the saturation in the system as shown in Figure 3.22a. Notice that in this implementation the tracking time constant T_t is the same as the integration time T_i. This value of the tracking time constant is often too large.

In Figure 3.22a, the model of the saturation will limit the control signal directly. It is important, therefore, to have a good model of the physical saturation. Too hard a limitation will cause an unnecessary limitation of the control action. Too weak a limitation will cause windup.

More flexibility is provided if the saturation is positioned as in Figure 3.22b.

a)



b)

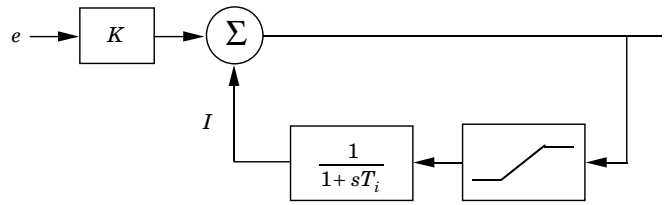


Figure 3.22 Two ways to provide anti-windup in the controller in Figure 3.3 where integral action is generated as automatic reset.

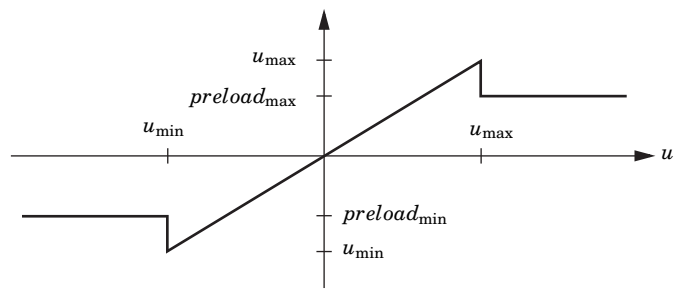


Figure 3.23 A "batch unit" used to provide anti-windup in the controller in Figure 3.3.

In this case, the saturation will not influence the proportional part of the controller. With this structure it is also possible to force the integral part to assume other preload values during saturation. This is achieved by replacing the saturation function by the nonlinearity shown in Figure 3.23. This anti-windup procedure is sometimes called a "batch unit" and may be regarded as a type of conditional integration. It is mainly used for adjusting the *overshoot* during startup when there is a large set point change. In early single-loop controllers the batch unit was supplied as a special add-on hardware.

Combined Schemes

Tracking and conditional integration can also be combined. In [Howes, 1986] it is suggested to manipulate the proportional band explicitly for batch control. This is done by introducing so-called *cutback points*. The high cutback is above the set point, and the low cutback is below. The integrator is clamped when the predicted process output is outside the cutback interval. Integration

is performed with a specified tracking time constant when the process output is between the cutback points. The cutback points are considered as controller parameters that are adjusted to influence the response to large set-point changes. A similar method is proposed in [Dreinhofer, 1988], where conditional integration is combined with back-calculation. In [Shinskey, 1988], the integrator is given a prescribed value $i = i_0$ during saturation. The value of i_0 is tuned to give satisfactory overshoot at startup. This approach is also called preloading.

3.6 When Can PID Control Be Used?

There are many requirements on a controlled system. It should respond well to set-point changes, it should attenuate load disturbances, measurement noise should not give excessive control actions, and the system should be insensitive to process variations. Design of a control system also involves aspects of process dynamics, actuator saturation, and disturbance characteristics. It may seem surprising that a controller as simple as the PID controller can work so well. The general empirical observation is that most industrial processes can be controlled reasonably well with PID control provided that the demands on the performance of the control are not too high. In the following paragraphs we delve further into this issue by first considering cases where PID control is sufficient and then discussing some generic problems where more sophisticated control is advisable.

When Is PI Control Sufficient?

All stable processes can be controlled by an I controller if the performance requirements are modest. Proportional action gives additional performance enhancements. It is therefore not surprising that the PI controller is the most common controller. Disregarding saturations a process with first-order dynamics can be given any desired performance using a PI controller. PI control can also be used for processes with integral action.

Derivative action is frequently not used. It is an interesting observation that many industrial controllers only have PI action and that in others the derivative action can be (and frequently is) switched off. It can be shown that PI control is adequate for all processes where the dynamics are essentially of the first order (level controls in single tanks, stirred tank reactors with perfect mixing, etc). It is fairly easy to find out if this is the case by measuring the step response or the frequency response of the process. If the step response looks like that of a first-order system or, more precisely, if the Nyquist curve lies in the first and the fourth quadrants only, then PI control is sufficient. Another reason is that the process has been designed so that its operation does not require tight control. Then, even if the process has higher-order dynamics, what it needs is an integral action to provide zero steady-state offset and an adequate transient response by proportional action.

When Is Derivative Action Useful?

A double integrator cannot be controlled by a PI controller. The reason is that the process has a phase lag of 180° and that a PI controller also has a phase

lag. Derivative action is needed for such a process. Disregarding saturations a process with second-order dynamics can be given any desired performance using a PID controller.

Similarly, PID control is sufficient for processes where the dominant dynamics are of the second order. For such processes there are no benefits gained by using a more complex controller. A typical case of derivative action improving the response is when the dynamics are characterized by time constants that differ in magnitude. Derivative action can then profitably be used to speed up the response. Temperature control is a typical case. Derivative control is also beneficial when tight control of a higher-order system is required. The higher-order dynamics would limit the amount of proportional gain for good control. With a derivative action, improved damping is provided, hence, a higher proportional gain can be used to speed up the transient response.

Many processes encountered in process control have dynamics with essentially monotone step responses, often with time delay. If the dynamics is delay dominated derivative action gives modest performance improvements compared with PI control, but derivative action gives significant improvements for processes that are lag dominated. This is discussed further in Chapter 7.

When Is More Sophisticated Control Needed?

The benefits of using a more sophisticated controller than the PID is demonstrated by some examples below.

Higher-Order Processes When the system is of an order higher than two, the control can be improved by using a more complex controller than the PID controller. This is illustrated by the following example.

EXAMPLE 3.3—SYSTEMS OF HIGH ORDER
Consider a third-order process described by the following transfer function

$$P(s) = \frac{1}{(s+1)^3}.$$

Figure 3.24 shows the control obtained using a PID controller and a more complex controller of higher order.

The PID controller has the parameters $K = 3.4$, $T_i = 2.0$, and $T_d = 0.6$. The PID controller is compared with a controller of the form

$$R(s)u(t) = -S(s)y(t) + T(s)y_{sp}(t),$$

with the following controller polynomials

$$\begin{aligned} R(s) &= s(s^2 + 11.5s + 57.5) \\ S(s) &= 144s^3 + 575s^2 + 870s + 512 \\ T(s) &= 8s^3 + 77s^2 + 309s + 512. \end{aligned}$$

The benefits of using a more complex controller in the case of higher-order dynamics is clearly demonstrated in the figure. □

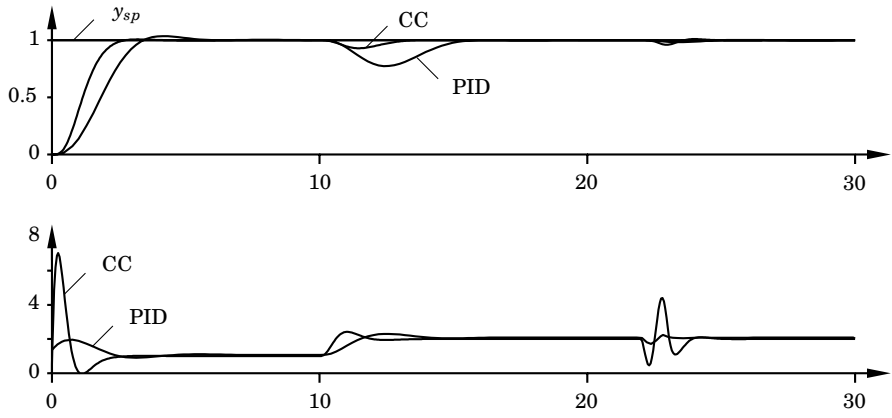


Figure 3.24 Control of the third-order system in Example 3.3 using a PID controller (PID) and a more complex controller (CC). The figure shows responses to a set-point change, a load disturbance, and finally a measurement disturbance. The upper diagram shows set point y_{sp} and measurement signal y, and the lower diagram shows control signal u.

Systems with Long Time Delay Control of systems with a dominant time delay are notoriously difficult. It is also a topic on which there are many different opinions concerning the merit of PID control. There seems to be general agreement that derivative action does not help much for processes with dominant time delays. For open-loop stable processes, the response to command signals can be improved substantially by introducing dead-time compensation. The load disturbance rejection can also be improved to some degree because a dead-time compensator allows a higher loop gain than a PID controller. Systems with dominant time delays are thus candidates for more sophisticated control.

EXAMPLE 3.4—COMPENSATION FOR TIME DELAYS
Consider a process with the transfer function

$$P(s) = \frac{1}{1+2s}e^{-4s},$$

which has a significant time delay. Figure 3.25 shows a simulation of the closed-loop system obtained with a PI controller with a gain $K = 0.2$ and an integral time $T_i = 2.5$. For comparison the figure also shows the performance with a Smith predictor, which is a special controller for a system with time delays. This controller will be discussed in detail in Chapter 8. The response to set-point changes is much better with the Smith predictor, but the improvement in response to load disturbances is less. □

Systems with Oscillatory Modes Systems with oscillatory modes occur in applications such as flexible robot arms, disk drives and optical memories,

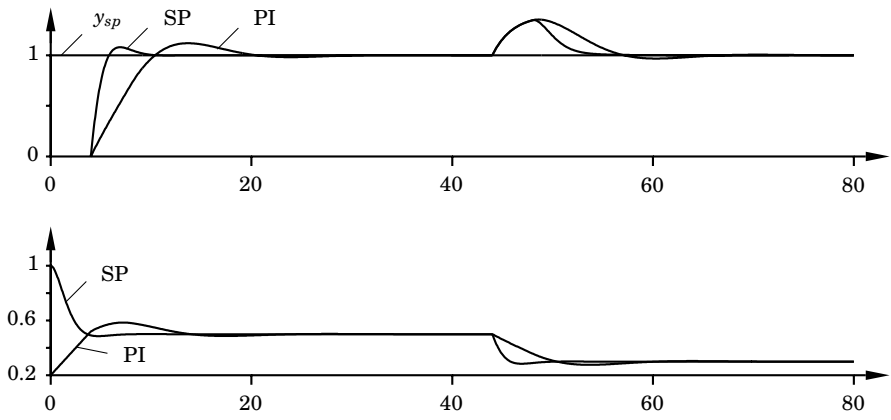


Figure 3.25 Control of the system in Example 3.4 with PI control (PI) and with a Smith predictor (SP). The upper diagram shows set point y_{sp} and measurement signal y, and the lower diagram shows control signal u.

atomic force microscopes (AFMs), micro-mechanical systems (MEMS), flexible space structures, and combustion systems. There are particular difficulties when the damping is very low so that the system is highly resonant. Typical applications are in micro-mechanical systems and in atomic force microscopes. Systems with resonant modes are not so common in process control applications. Derivative action can give drastic improvements for oscillatory systems as is illustrated by the following example.

EXAMPLE 3.5—AN OSCILLATORY SYSTEM WITH LOW DAMPING
Consider a system with the normalized transfer function

$$P(s) = \frac{a^2}{s^2 + 2\zeta as + a^2}.$$

We will consider systems with very low relative damping $\zeta = 0.005$. The performance obtained with a PI controller is severely limited by the low relative damping of the process. Since a PI controller cannot provide any phase lead the damping of the oscillatory modes of the closed-loop system will be smaller than those of the open-loop system. A key requirement is that the PI controller does not excite the high-frequency modes.

A pure integrating controller is a reasonable choice. The stability condition for such a controller is $k_i < 2\zeta a^3$, which implies that $k_i = \zeta a^3$ is a good value of the controller gain. With this choice the closed-loop system has the same settling time as the open-loop system. The response time can only be improved a little by adding proportional action. Figure 3.26 shows the input and the output for a step change in the set point for a controller with these parameters. The step response has a settling time of about 1500 s. The reason why the system has to be so slow is that the oscillatory motion cannot be damped by the PI controller, and it is therefore necessary to have a slow controller so that

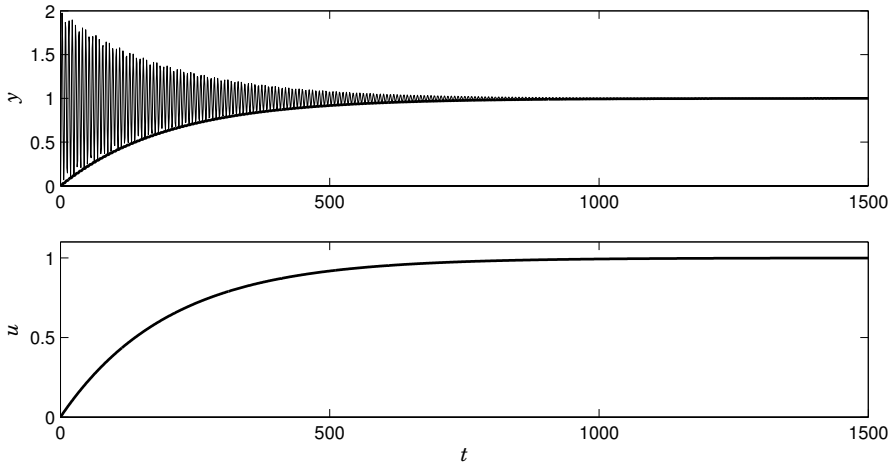


Figure 3.26 Output and control signals for PI control of the oscillatory system. The oscillating signal is the open-loop response.

the oscillatory modes are not excited by the controller. In the figure we have also shown the step response of the process.

The performance can be improved drastically by using a PID controller. One possibility is to use a PID controller with parameters

$$\begin{aligned} k &= (1 + 2\alpha_0\zeta_0)\frac{\omega_0^2}{a^2} - 1 \\ k_i &= \frac{\alpha_0\omega_0^3}{a^2} \\ k_d &= \frac{(\alpha_0 + 2\zeta_0)\omega_0 - 2\zeta a}{a^2}. \end{aligned} \tag{3.23}$$

Here ω_0, α_0, and ζ_0 are design parameters that give the properties of the closed-loop system. A reasonable choice is $\omega_0 = 3a$, $\alpha_0 = 1$, $\zeta_0 = 0.5$. The rationale for the formulas and the parameter choices will be given later in Section 6.4. Figure 3.27 shows the responses of the output and the controller to a step change in the set point. In this case the system has a settling time of about 2 s. This is about three orders of magnitude better than with PI control! The reason for this is that by using derivative action it is possible to damp the oscillations. This is indicated in the figure by showing the open-loop response of the process in dashed line. Also notice the drastic differences in the control signals for PI and PID control. It is also important to use set-point weighting with $b = 0$ to avoid a rapid change of the control variable. Such a change will excite the poorly damped oscillatory modes. □

Summary

When the dynamics of the process to be controlled are simple, a PID controller is sufficient. When the dynamics become more complicated, the performance

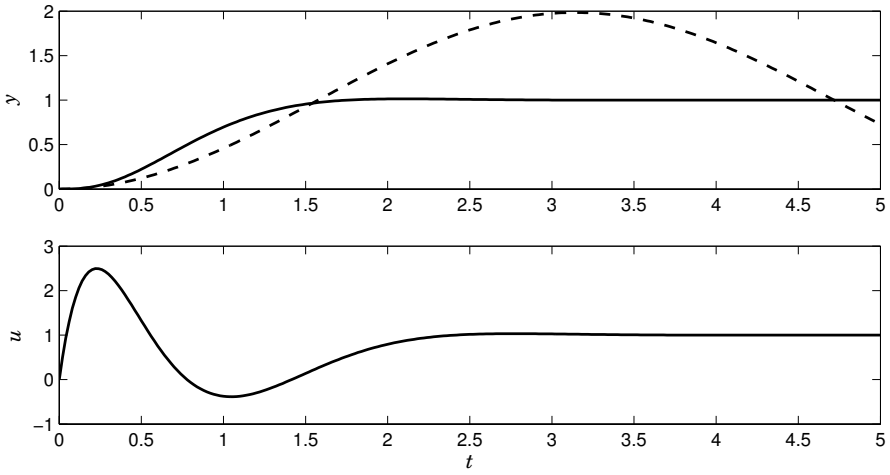


Figure 3.27 Output and control signals for PID control of the oscillatory system. The dashed curve is the open-loop response.

can be improved by using a more sophisticated controller structure than the PID. Examples of such processes have been given above.

For some systems with large parameter variations it is possible to design linear controllers that allow operation over a wide parameter range. Such controllers are, however, often of high order.

The control of process variables that are closely related to important quality variables may be of a significant economic value. In such control loops it is frequently necessary to select the controller with respect to the disturbance characteristics. This often leads to strategies that are not of the PID type. These problems are often associated with time delays.

A general controller attempts to model the disturbances acting on the system. Since a PID controller has limited complexity, it cannot model complex disturbance behavior in general nor periodic disturbances in particular.

3.7 Summary

A detailed presentation of the PID algorithm has been given. Several modifications of the "textbook" version must be made to obtain a practical, useful controller. Problems that must be handled are, for example, integral windup and introduction of set-point values. A discussion of the limitations of the PID algorithm and a characterization of processes in which the PID controller manages to perform the control have also been given.

3.8 Notes and References

An interesting summary of the development of the PID controller is given in [Bennett, 2000]. Proportional feedback in the form of a centrifugal governor was used to regulate the speed of windmills around 1750. James Watt used a similar system for speed control of steam engines in 1788. Integral action was discovered later by several authors. It is explained analytically by [Maxwell, 1868] and [Vyshnegradskii, 1876]. Feedback control with proportional and integral action was rediscovered many times after that. In the early stages, the development of controllers was closely related to development of sensors and actuators. Sensing, actuation, and control were often combined in the same device. There was also confusion about integral and derivative action because some controllers acted through motors that had integral action.

The PID controller in the form we know it today emerged in the period from 1915 to 1940. The major development was made in legendary instrument companies such as Bristol, Fisher, Foxboro, Honeywell, Leeds & Northrup, Mason-Neilan, and Taylor Instrument. Integral action was called automatic reset because it replaced a manual reset that was used in proportional controllers to obtain the correct steady-state value. The potential of a controller that could anticipate future control errors was discussed in the 1920s. However, it took some time before the idea could be implemented. A controller with derivative action was introduced by Ralph Clarridge of the Taylor Instrument Company in 1935. At that time the function was called "pre-act." An interesting overview of the early history of PID controllers is given by [Stock, 1988]. There is also much interesting material in publications from the instrument companies. The interview with Ziegler, who is one of the pioneers in our field, in [Blickley, 1990], gives a perspective on the early development; other interesting material is found in [Bennett, 1993].

It is interesting to observe that feedback was crucial for the construction of the controller itself. The early pneumatic systems used the idea that an essentially linear controller can be obtained by a feedback loop composed of linear passive components and a nonlinear amplifier, the flapper valve. Similar ideas were used in electronic controllers with electric motors and relays.

Many of the practically useful modifications of the controller first appeared as special hardware functions. They were not expressed in mathematical form. An early mathematical analysis of a steam engine with a governor was made independently by [Maxwell, 1868] and [Vyshnegradskii, 1876]. This analysis clearly showed the difference between proportional and integral control. The papers [Minorsky, 1922; Küpfmüller, 1928; Nyquist, 1932; Hazen, 1934] were available at the time when the PID controller was developed. However, there is little evidence that the engineers in the process control field knew about them. Process control therefore developed independently. Two of the early papers [Grebe *et al.*, 1933] and [Ivanoff, 1934] were written by engineers at the Dow Chemical Company. There were also contributions from university researchers [Callender *et al.*, 1936] and [Hartree *et al.*, 1937].

The PID controller has gone through an interesting development because of the drastic technology changes that have happened since 1930. The pneumatic controller improved drastically by making systematic use of the force balance

principle. Pneumatics was replaced by electronics when the operational amplifier appeared in the 1950s. The emergence of computer control in the 1960s was an important development. In the early computer control systems the computer commanded the set points of analog controllers. The next stage of the development was direct digital control (DDC), where the computer was controlling the actuator directly; see [Webb, 1967]. A digital computer was then used to implement many PID controllers. This development led to a reconsideration of much of the fundamentals of PID control; see [Goff, 1966b], [L&N, 1968], [Moore *et al.*, 1970], and [Palmor and Shinnar, 1979]. The appearance of microprocessors in the 1970s made it possible to use digital control for single-loop controllers. It also led to the development of distributed control systems for process control, where the PID controller was a key element; see [Lukas, 1986]. As the computing power of the microprocessors increased it was possible to introduce tuning, adaptation, and diagnostics in the single-loop controllers. This development started in the 1980s and has accelerated in the 1990s; see [Åström *et al.*, 1993].

It is interesting to observe that many facts about PID control were rediscovered in connection with the shifts in technology. One reason being that many practical aspects of PID control were considered as proprietary information that was not easily accessible in public literature. Much useful information was also scattered in the literature.

Two different approaches were used to deal with set-point changes in early controllers. Some controllers used error feedback but others introduced the set point only in the integral part. The effect of this is that the overshoot that occurs with set-point changes can be reduced. The idea that a separation of the responses to set points and load disturbances can be accomplished by using a controller with two degrees of freedom was introduced in [Horowitz, 1963]. The application to PID control was introduced in [Araki, 1984]. An early industrial application is described in [Shigemasa *et al.*, 1987], see also [Araki and Taguchi, 1998; Taguchi and Araki, 2000]. Set-point weighting where an adjustable fraction of the set point is introduced in proportional and derivative parts is now a common feature of PID controllers.

The phenomenon of integral windup was well known in the early analog implementations. The controller structures used were often such that windup was avoided. The anti-windup schemes were rediscovered in connection with the development of direct digital control. This is discussed in [Fertik and Ross, 1967]. Much work on avoiding windup has been done since then, and windup has now made its way into some textbooks of control; see [Åström and Wittenmark, 1997]. There are many papers written on the windup phenomenon; see [Kramer and Jenkins, 1971; Glattfelder and Schaufelberger, 1983; Krikelis, 1984; Gallun *et al.*, 1985; Kapasouris and Athans, 1985; Glattfelder and Schaufelberger, 1986; Howes, 1986; Åström, 1987b; Hanus *et al.*, 1987; Chen and Wang, 1988; Glattfelder *et al.*, 1988; Hanus, 1988; Zhang and Evans, 1988; Åström and Rundqwist, 1989; Rundqwist, 1990; Walgama and Sternby, 1990]. A detailed treatment of the windup problem is given in the book [Glattfelder and Schaufelberger, 2003]. Mode switching is treated in the paper [Åström, 1987b].