13

Implementation

13.1 Introduction

PID controllers were originally implemented using analog techniques. Early systems used pneumatic relays, bellows, and needle-valve constrictions. Electric motors with relays and feedback circuits and operational amplifiers were used later. Many of the features like anti-windup and derivation of process output instead of control error were incorporated as “tricks” in these implementations.

It is now common practice to implement PID controllers using microprocessors, and some of the old tricks have been rediscovered. Several digital PID controllers in use today have features that are inherited from old techniques when the controllers were implemented using pneumatic devices. This is a typical example of the fact that ideas sometimes change at a much slower rate than hardware. Several additional issues must be considered in connection with digital implementations. The most important ones have to do with sampling, discretization, and quantization.

This chapter presents some implementation issues related to PID control. Section 13.2 gives a short overview of the early analog pneumatic and electronic implementations. Section 13.3 treats computer implementation aspects such as sampling, prefiltering, and discretization of the PID algorithm. Velocity algorithms, or incremental algorithms, are needed in applications where the integration is performed outside the controller. The most common application is electrical motors. These algorithms, which are shown to be useful even when the integration is performed inside the controller, are presented in Section 13.4. Operational aspects, such as bumpless transfers at mode switches and parameter changes, are presented in Section 13.5. A controller may have different outputs depending on which actuating device is used. Controller outputs are discussed in Section 13.6. The chapter ends with a summary and references.

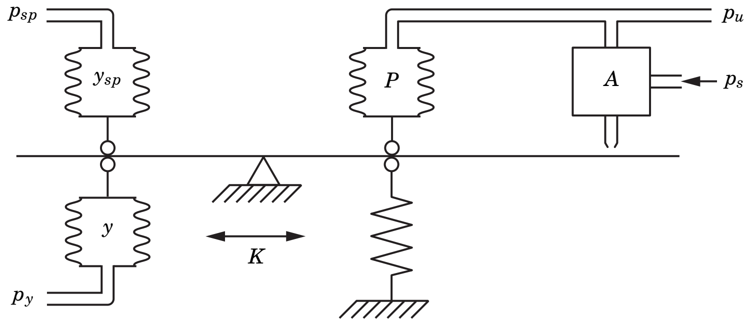

Figure 13.1 Schematic diagram of a pneumatic P controller based on the force balance principle.

13.2 Analog Implementations

The early implementations of PID controllers were all analog. This section presents the pneumatic controller implementation and the analog electronic implementation.

The Pneumatic Controller

This section presents the basic function of pneumatic controllers. To make this clear, lots of details have been removed from the presentation and the drawings. We refer to the references for details.

The Pneumatic P Controller A schematic diagram of a pneumatic P controller based on force balance is shown in Figure 13.1. The system consists of a beam that can rotate around a pivot point. The beam is provided with three bellows, a spring, a position sensor, and a pneumatic amplifier. The bellows can exert forces on the beam proportional to the pressure in the bellows. The position sensor is a flapper valve, which gives a pressure signal that is approximately inversely proportional to the distance between the nozzle and the beam. The pneumatic amplifier A can amplify pneumatic signals.

To understand the operation of the system it is assumed that the forces of all bellows are proportional to the air pressure in the bellows. The two left bellows receive pressures p_{sp} and p_y proportional to the set point and the measured variable, respectively. The pressure amplifier A receives supply pressure p_s and provides output pressure p_u, which is the controller output. The right bellow labeled P is the feedback bellow or the proportional bellow. In the P controller the pressure in this bellow, p_p, is equal to the output pressure p_u.

A torque balance gives the following relation between the pressures:

$$p_u - bias = K(p_{sp} - p_y). \tag{13.1}$$

The *bias* term is the force given by the spring. The gain K is determined by the position of the balance point, and can therefore be chosen by adjusting this point. Equation 13.1 is obviously the equation for a P controller.

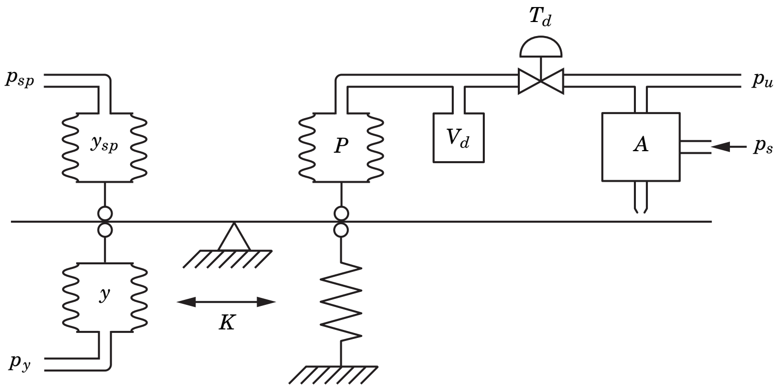

Figure 13.2 Schematic diagram of a pneumatic PD controller based on the force balance principle.

Suppose, for example, that the set-point pressure p_{sp} increases. The beam will then rotate in positive direction, leading to a decrease in the outflow from the nozzle valve. This will cause an increase of the output pressure p_u.

The Pneumatic PD Controller A pneumatic PD controller is shown in Figure 13.2. In this controller, a valve and a volume V_d is introduced between the amplifier A and the feedback bellow P. Because of this valve, it is no longer true that $p_p = p_u$, but the following dynamic relation between the two pressures holds:

$$P_p(s) = \frac{1}{1+sT_d} P_u(s). \tag{13.2}$$

The value of the time constant T_d can be adjusted by the valve position.

Since a counteraction caused by the feedback bellow P is delayed compared with the P controller, a change in p_y or p_{sp} will initially result in a larger reaction in the output pressure p_u.

A torque balance gives the following relations between the pressures:

$$p_p - bias = K(p_{sp} - p_y).$$

From (13.2) this gives the following output pressure;

$$P_u(s) = bias + K(1 + sT_d)(P_{sp} - P_y),$$

which is the equation of a PD controller with derivative time T_d.

The Pneumatic PID Controller A pneumatic PID controller is shown in Figure 13.3. In this controller, the spring is replaced by a bellow labeled I. This bellow is connected to the pressure p_p through a volume V_i and a valve labeled T_i. The pressure in the bellow I is

$$P_i(s) = \frac{1}{1+sT_i} P_p(s), \tag{13.3}$$

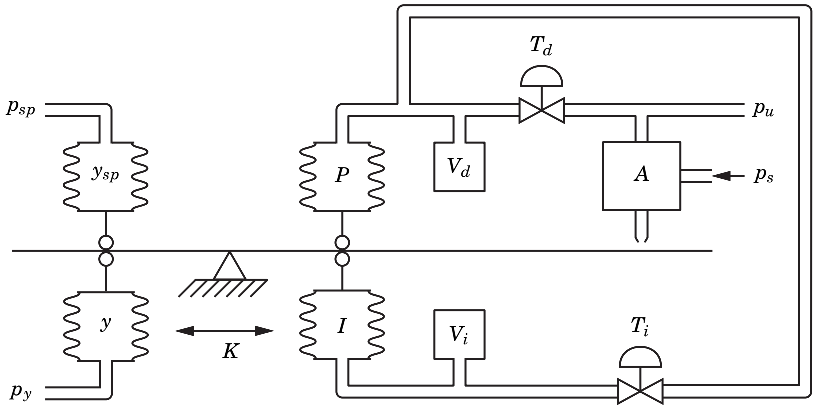

Figure 13.3 Schematic diagram of a pneumatic PID controller based on the force balance principle.

where the time constant T_i may be adjusted by the valve labeled T_i.

A torque balance gives the following relations between the pressures:

$$p_p - p_i = K(p_{sp} - p_y).$$

From (13.2) and (13.3) this gives the following output pressure:

$$P_u(s) = K\frac{(1+sT_i)(1+sT_d)}{sT_i}(P_{sp}(s) - P_y(s)). \tag{13.4}$$

This equation shows that the system is a PID controller on interacting form (see Section 3.2) with gain K, integral time T_i, and derivative time T_d.

The idea of using feedback in the controller was a major invention. Both the flapper valve and the pneumatic amplifier are strongly nonlinear. The arrangement with the feedback loop implies that the input-output relation of the controller does not change much even if the component changes, provided that the gain is sufficiently large. This idea, which is called force feedback, gave drastic improvements in the performance of the controllers. A typical example of the impact of feedback.

The Analog Electronic Controller

A PID controller may be implemented by analog electronic components in many ways. This section presents some basic implementations based on operational amplifiers. Lots of details have been left out for the sake of simplicity. As for the pneumatic controllers, we refer to the references for details.

The Electronic PI Controller An electronic PI controller is shown in Figure 13.4.

An approximate relation between the input voltage e and the output voltage u is obtained by

$$u = -\frac{Z_1}{Z_0}e,$$

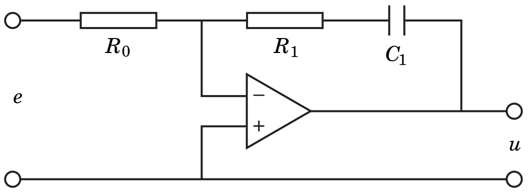

Figure 13.4 Schematic diagram of an electronic PI controller based on feedback around an operational amplifier.

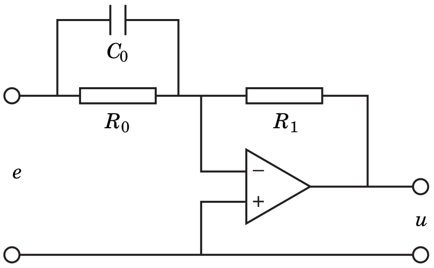

Figure 13.5 Schematic diagram of an electronic PD controller based on feedback around an operational amplifier.

where Z_0 is the impedance between the negative input of the amplifier and the input voltage e, and Z_1 is the impedance between the zero input of the amplifier and the output voltage u. These impedances are

$$\begin{aligned} Z_0 &= R_0 \\ Z_1 &= R_1 + \frac{1}{C_1 p}, \end{aligned}$$

where p is the differential operator. This gives the following relation between the input voltage e and the output voltage u:

$$u = -\frac{Z_1}{Z_0} e = -\frac{R_1}{R_0}\left(1 + \frac{1}{R_1 C_1 p}\right) e.$$

This is a PI controller with parameters

$$K = \frac{R_1}{R_0} \qquad T_i = R_1 C_1.$$

A P controller is obtained by removing the capacitor.

The Electronic PD Controller An electronic PD controller is shown in Figure 13.5.

The impedances between the negative input of the amplifier and the input and output voltages, respectively, become

$$\begin{aligned} Z_0 &= \frac{R_0}{1 + R_0 C_0 p} \\ Z_1 &= R_1. \end{aligned}$$

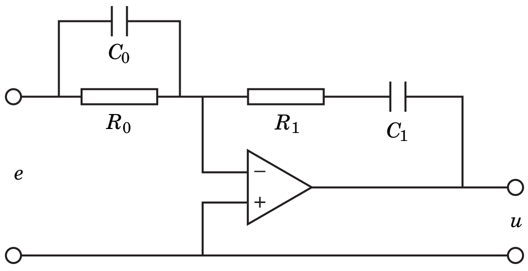

Figure 13.6 Schematic diagram of an electronic PID controller based on feedback around an operational amplifier.

This gives the following relation between the input voltage e and the output voltage u:

$$u = -\frac{Z_1}{Z_0} e = -\frac{R_1}{R_0} (1 + R_0 C_0 p)\, e.$$

This is a PD controller with parameters

$$K = \frac{R_1}{R_0} \qquad T_d = R_0 C_0.$$

A P controller is obtained by removing the capacitor.

The Electronic PID Controller An electronic PID controller may be obtained by combining the two previous schemes. This is shown in Figure 13.6.

The impedances between the negative input of the amplifier and the input and output voltages, respectively, become

$$Z_0 = \frac{R_0}{1 + R_0 C_0 p}$$
$$Z_1 = R_1 + \frac{1}{C_1 p}.$$

This gives the following relation between the input voltage e and the output voltage u:

$$u = -\frac{Z_1}{Z_0} e = -\frac{R_1}{R_0} \frac{(1 + R_0 C_0 p)(1 + R_1 C_1 p)}{R_1 C_1 p} e.$$

This is a PID controller on interacting form with parameters

$$K = \frac{R_1}{R_0} \qquad T_i = R_1 C_1 \qquad T_d = R_0 C_0.$$

13.3 Computer Implementations

Most controllers are implemented nowadays in computers. There are some topics that have to be considered due to the fact that the signals are sampled at discrete time instances. These topics are treated in this section.

Sampling

When the controller is implemented in a computer, the analog inputs are read, and the outputs are set with a certain sampling period. This is a drawback compared to the analog implementations, since the sampling introduces dead time in the control loop.

When a digital computer is used to implement a control law, the ideal sequence of operation is the following.

1. Wait for clock interrupt
2. Read analog input
3. Compute control signal
4. Set analog output
5. Update controller variables
6. Go to 1

With this implementation, the delay is minimized. If the analog input is read with a sampling period h, the average delay of the measurement signal is $h/2$. The computation time is often short compared to the sampling period. This means that the total delay is about $h/2$. However, most controllers and instrument systems do not organize the calculation in this way. Therefore, the delays introduced because of the sampling are often several sampling periods.

Aliasing

The sampling mechanism introduces some unexpected phenomena, which must be taken into account in a good digital implementation of a PID controller. To explain these, consider the signals

$$s(t) = \cos(n\omega_s t \pm \omega t)$$

and

$$s_a(t) = \cos(\omega t),$$

where $\omega_s = 2\pi/h$ [rad/s] is the sampling frequency. Well-known formulas for the cosine function imply that the values of the signals at the sampling instants $[kh, k = 0, 1, 2, ...]$ have the property

$$s(kh) = \cos(nkh\omega_s \pm \omega kh) = \cos(\omega kh) = s_a(\omega kh).$$

The signals s and s_a thus have the same values at the sampling instants. This means that there is no way to separate the signals if only their values at the sampling instants are known. Signal s_a is, therefore, called an *alias* of signal s. This is illustrated in Figure 13.7. A consequence of the aliasing effect is that a high-frequency disturbance after sampling may appear as a low-frequency signal. In Figure 13.7 the sampling period is 1 s, and the sinusoidal disturbance has a period of 6/5 s. After sampling, the disturbance appears as a sinusoid with the frequency

$$f_a = 1 - \frac{5}{6} = 1/6 \text{ Hz}.$$

This low-frequency signal with time period 6 s is seen in the figure.

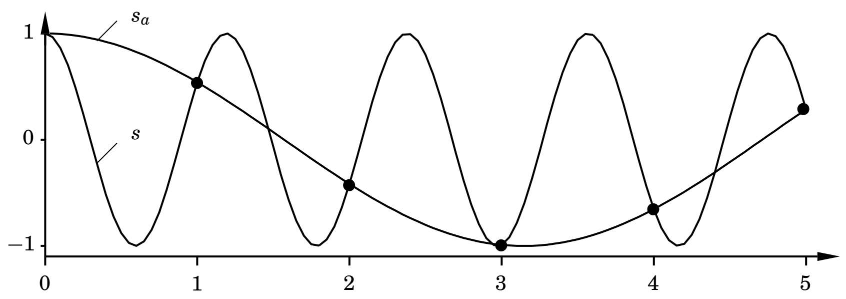

Figure 13.7 Illustration of the aliasing effect. The diagram shows signal s and its alias s_a.

Prefiltering

The aliasing effect can create significant difficulties if proper precautions are not taken. High frequencies, which in analog controllers normally are effectively eliminated by low-pass filtering, may, because of aliasing, appear as low-frequency signals in the bandwidth of the sampled control system. To avoid these difficulties, an analog prefilter (which effectively eliminates all signal components with frequencies above half the sampling frequency) should be introduced. Such a filter is called an antialiasing filter. A second-order Butterworth filter is a common antialiasing filter. Higher-order filters are also used in critical applications. An implementation of such a filter using operational amplifiers is shown in Figure 13.8. The selection of the filter bandwidth is illustrated by the following example.

EXAMPLE 13.1—SELECTION OF PREFILTER BANDWIDTH
Assume it is desired that the prefilter attenuate signals by a factor of 16 at half the sampling frequency. If the filter bandwidth is ω_b and the sampling frequency is ω_s, we get

$$(\omega_s/2\omega_b)^2 = 16.$$

Hence,

$$\omega_b = \frac{1}{8}\,\omega_s.$$

□

Notice that the dynamics of the prefilter will be combined with the process dynamics.

Discretization

To implement a continuous-time control law, such as a PID controller in a digital computer, it is necessary to approximate the derivatives and the integral that appear in the control law. A few different ways to do this are presented below.

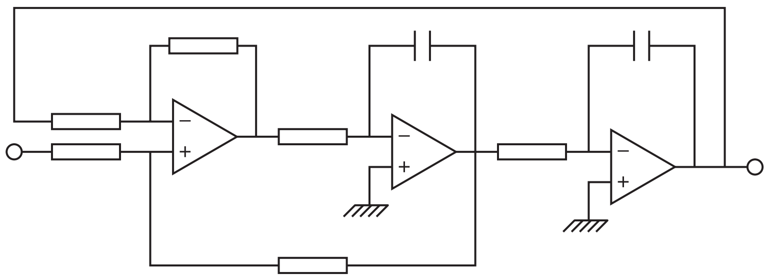

Figure 13.8 Circuit diagram of a second-order Butterworth filter.

Proportional Action The proportional term is

$$P = K(by_{sp} - y).$$

This term is implemented simply by replacing the continuous variables with their sampled versions. Hence,

$$P(t_k) = K\left(by_{sp}(t_k) - y(t_k)\right), \tag{13.5}$$

where $\{t_k\}$ denotes the sampling instants, i.e., the times when the computer reads the analog input.

Integral Action The integral term is given by

$$I(t) = \frac{K}{T_i}\int_0^t e(s)ds.$$

It follows that

$$\frac{dI}{dt} = \frac{K}{T_i}e. \tag{13.6}$$

There are several ways of approximating this equation. Approximating the derivative by a forward difference gives

$$\frac{I(t_{k+1}) - I(t_k)}{h} = \frac{K}{T_i}e(t_k).$$

This leads to the following recursive equation for the integral term

$$I(t_{k+1}) = I(t_k) + \frac{Kh}{T_i}e(t_k). \tag{13.7}$$

If the derivative in Equation 13.6 is approximated instead by a backward difference, the following is obtained:

$$\frac{I(t_k) - I(t_k - 1)}{h} = \frac{K}{T_i}e(t_k).$$

This leads to the following recursive equation for the integral term:

$$I(t_{k+1}) = I(t_k) + \frac{Kh}{T_i}\, e(t_{k+1}). \tag{13.8}$$

Another simple approximation method is due to Tustin. This approximation is

$$I(t_{k+1}) = I(t_k) + \frac{Kh}{T_i}\, \frac{e(t_{k+1}) + e(t_k)}{2}. \tag{13.9}$$

Yet another method is called ramp equivalence. This method gives exact outputs at the sampling instants if the input signal is continuous and piecewise linear between the sampling instants. The ramp equivalence method gives the same approximation of the integral term as the Tustin approximation, i.e., Equation 13.9.

Notice that all approximations have the same form, i.e.,

$$I(t_{k+1}) = I(t_k) + b_{i1}e(t_{k+1}) + b_{i2}e(t_k), \tag{13.10}$$

but with different values of parameters b_{i1} and b_{i2}.

Derivative Action The derivative term with the classical first-order filter is given by Equation 3.14, i.e.,

$$\frac{T_d}{N}\,\frac{dD}{dt} + D = -KT_d\,\frac{dy}{dt}. \tag{13.11}$$

This equation can be approximated in the same way as the integral term.

Approximating the derivative by a forward difference gives

$$\frac{T_d}{N}\,\frac{D(t_{k+1}) - D(t_k)}{h} + D(t_k) = -KT_d\,\frac{y(t_{k+1}) - y(t_k)}{h}.$$

This can be rewritten as

$$D(t_{k+1}) = \left(1 - \frac{Nh}{T_d}\right) D(t_k) - KN\left(y(t_{k+1}) - y(t_k)\right). \tag{13.12}$$

If the derivative in Equation 13.11 is approximated by a backward difference, the following equation is obtained:

$$\frac{T_d}{N}\,\frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -KT_d\,\frac{y(t_k) - y(t_{k-1})}{h}.$$

This can be rewritten as

$$D(t_k) = \frac{T_d}{T_d + Nh}\, D(t_{k-1}) - \frac{KT_dN}{T_d + Nh}\left(y(t_k) - y(t_{k-1})\right). \tag{13.13}$$

Using the Tustin approximation to approximate the derivative term gives

$$D(t_k) = \frac{2T_d - Nh}{2T_d + Nh}\, D(t_{k-1}) - \frac{2KT_dN}{2T_d + Nh}\left(y(t_k) - y(t_{k-1})\right). \tag{13.14}$$

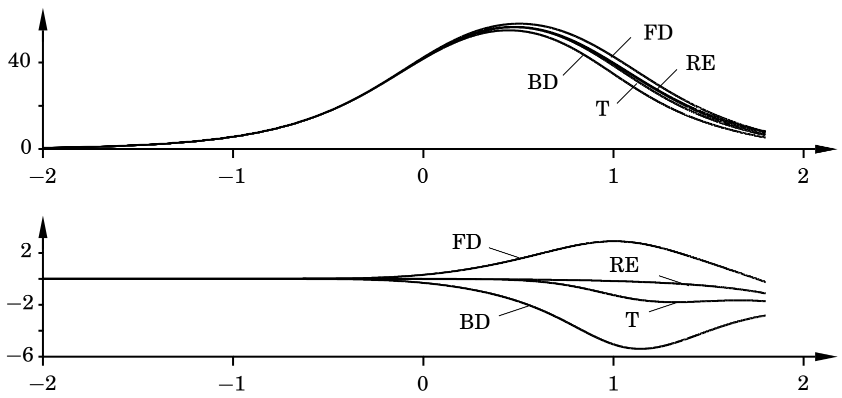

Figure 13.9 Phase curves for PD controllers obtained by different discretizations of the derivative term $sT_d/(1+sT_d/N)$ with $T_d = 1, N = 10$ and a sampling period 0.02. The discretizations are forward differences (FD), backward differences (BD), Tustin's approximation (T), and ramp equivalence (RE). The lower diagram shows the differences between the approximations and the true phase curve.

Finally, the ramp equivalence approximation is

$$D(t_k) = e^{-Nh/T_d} D(t_{k-1}) - \frac{KT_d(1-e^{-Nh/T_d})}{h}\,(y(t_k) - y(t_{k-1})). \qquad (13.15)$$

All approximations have the same form,

$$D(t_k) = a_d D(t_{k-1}) - b_d\,(y(t_k) - y(t_{k-1}))\,, \qquad (13.16)$$

but with different values of parameters a_d and b_d.

The approximations of the derivative term are stable only when $|a_d| < 1$. When using the forward difference approximation stability requires that $T_d > Nh/2$. The approximation becomes unstable for small values of T_d. The other approximations are stable for all values of T_d. Notice, however, that Tustin's approximation and the forward difference approximation give negative values of a_d if T_d is small. This is undesirable because the approximation will then exhibit ringing. The backward difference approximation give good results for all values of T_d, including $T_d = 0$.

For reasonable fast sampling there are only small differences between the approximations as long as they are stable. There are, however, practical differences. In a general-purpose controller it is desirable that derivative action can be switched off. A natural way to do this is to set $T_d = 0$. This can easily be accomplished when the derivative is approximated by a backward difference. All other methods will either give instability or overflow for $T_d = 0$. The backward difference is therefore a reasonable choice for approximating the derivative.

Figure 13.9 shows the phase curves for the different discrete time approximations. Tustin's approximation and the ramp equivalence approximation give the best agreement with the continuous time case, the backward approximation gives less phase advance, and the forward approximation gives more phase advance. The forward approximation is seldom used because of the problems with instability for small values of derivative time T_d. Tustin's algorithm is used quite frequently because of its simplicity and its close agreement with the continuous time transfer function. The backward difference is used when an algorithm that is well behaved for small T_d is needed.

All approximations of the PID controller can be represented as

$$R(q)u(kh) = T(q)y_{sp}(kh) - S(q)y(kh), \tag{13.17}$$

where q is the forward shift operator, and the polynomials R, S, and T are of second order. The polynomials R, S, and T have the forms

$$\begin{aligned} R(q) &= (q-1)(q-a_d) \\ S(q) &= s_0q^2 + s_1q + s_2 \\ T(q) &= t_0q^2 + t_1q + t_2, \end{aligned} \tag{13.18}$$

which means that Equation 13.17 can be written as

$$\begin{aligned} u(kh) = {} & t_0y_{sp}(kh) + t_1y_{sp}(kh-h) + t_2y_{sp}(kh-2h) \\ & - s_0y(kh) - s_1y(kh-h) - s_2y(kj-2h) \\ & + (1+a_d)u(kh-h) - a_du(kh-h). \end{aligned}$$

The coefficients in the S and T polynomials are

$$\begin{aligned} s_0 &= K + b_{i1} + b_d \\ s_1 &= -K(1+a_d) - b_{i1}a_d + b_{i2} - 2b_d \\ s_2 &= Ka_d - b_{i2}a_d + b_d \\ t_0 &= Kb + b_{i1} \\ t_1 &= -Kb(1+a_d) - b_{i1}a_d + b_{i2} \\ t_2 &= Kba_d - b_{i2}a_d. \end{aligned} \tag{13.19}$$

The coefficients in the polynomials for different approximation methods are given in Table 13.1.

Controller with Second Order Filter

A nice implementation of a PID controller is to combine a second order filtering of the measured signal with an ideal PID controller; see Section 3.3. We will now discuss how such controllers can be implemented. Let y be the measured signal and y_f the filtered signal. We have

$$Y_f(s) = G_f(s)Y(s) = \frac{1}{1 + sT_f + (sT_f)^2/2}Y(s). \tag{13.20}$$

Introducing the state variables $x_1 = y_f$ and $x_2 = T_f dy_f/dt$ the filter can be represented as

$$\begin{aligned} T_f\frac{dx_1}{dt} &= x_2 \\ T_f\frac{dx_2}{dt} &= 2(-x_1 - x_2 + y). \end{aligned} \tag{13.21}$$

Table 13.1 Coefficients in different approximations of the continuous time PID controller.

	Forward	Backward	Tustin	Ramp equivalence
b_{i1}	0	$\frac{Kh}{T_i}$	$\frac{Kh}{2T_i}$	$\frac{Kh}{2T_i}$
b_{i2}	$\frac{Kh}{T_i}$	0	$\frac{Kh}{2T_i}$	$\frac{Kh}{2T_i}$
a_d	$1-\frac{Nh}{T_d}$	$\frac{T_d}{T_d+Nh}$	$\frac{2T_d-Nh}{2T_d+Nh}$	e^{-Nh/T_d}
b_d	KN	$\frac{KT_dN}{T_d+Nh}$	$\frac{2KT_dN}{2T_d+Nh}$	$\frac{KT_d(1-e^{-Nh/T_d})}{h}$

The filtered derivative $dy_f/dt = x_2/T_f$ can be extracted from the filter and the controller is then given by

$$u = k(by_{sp} - y_f) + k_i \int_0^t (y_{sp}(\tau) - y_f(\tau))d\tau + k_d \frac{dy_f}{dt}. \tag{13.22}$$

If the PID controller (13.22) is implemented digitally, both $x_1 = y_f$ and $x_2 = T_f dy_f/dt$ have to be converted to digital form. This implementation is suitable for special-purpose systems. For general-purpose systems the filter can be implemented digitally. Assume that the sampling has period h and let the sampling instants be t_k. Approximating the derivative in (13.21) with a backward difference we find

$$\begin{aligned} x_1(t) &= x_1(t-h) + \frac{hT_f}{T_f^2 + 2hT_f + 2h^2} x_2(t-h) \\ &\quad + \frac{2h^2}{T_f^2 + 2hT_f + 2h^2}(y(t) - x_1(t-h)) \\ x_2(t) &= \frac{T_f^2}{T_f^2 + 2hT_f + 2h^2} x_2(t-h) + \frac{2hT_f}{T_f^2 + 2hT_f + 2h^2}(y(t) - x_1(t-h)). \end{aligned}$$

To obtain an algorithm which permits the parameter T_f to be zero we introduce the state variables

$$\begin{aligned} y_1 &= x_1 \\ y_2 &= \frac{h}{T_f} x_2. \end{aligned}$$

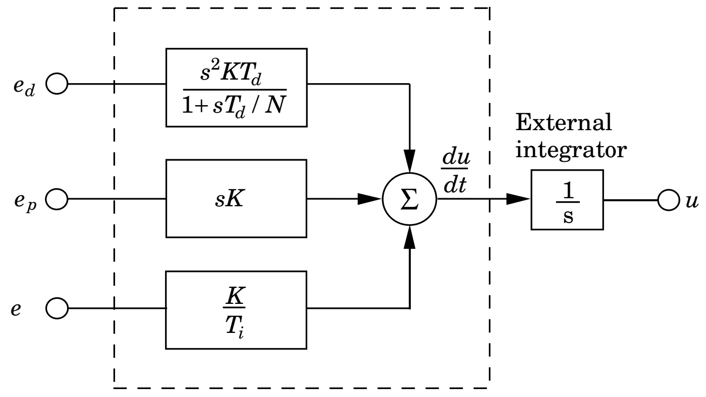

Figure 13.10 Block diagram of a PID algorithm in velocity form.

The equation for the controller can then be written as

$$\begin{aligned}
y_1(t) &= y_1(t-h) + p_1 y_2(t-h) + p_2\big(y(t) - y_1(t-h)\big) = y_1(t-h) + y_2(t)\\
y_2(t) &= p_1 y_2(t-h) + p_2\big(y(t) - y_1(t-h)\big)\\
v(t) &= K(by_{sp} - y_1) - p_4 y_2(t) + I(t-h)\\
&= K(by_{sp} - y_1) - p_2(K+p_4)y(t)\\
&\quad + \big(p_2(K+p_4) - K\big)y_1(t-h) + \big(K - p_1(k+p_4)\big)y_2(t-h) + I(t-h)\\
u(t) &= sat(v)\\
I(t) &= I(t-h) + p_3(y_{sp}(t) - y_1(t)) + p_5(u(t) - v(t)).
\end{aligned} \tag{13.23}$$

where the integral term has been approximated by a forward difference and protection for windup has been introduced. The parameters of the controller are given by

$$\begin{aligned}
p_1 &= \frac{T_f^2}{T_f^2 + 2hT_f + 2h^2} \quad p_2 = \frac{2h^2}{T_f^2 + 2hT_f + 2h^2}\\
p_3 &= \frac{Kh}{T_i} \quad p_4 = \frac{KT_d}{h} \quad p_5 = \frac{h}{T_t}
\end{aligned} \tag{13.24}$$

13.4 Velocity Algorithms

The algorithms described so far are called positional algorithms because the output of the algorithms is the control variable. In certain cases the control system is arranged in such a way that the control signal is driven directly by an integrator, e.g., a motor. It is then natural to arrange the algorithm in such a way that it gives the velocity of the control variable. The control variable is then obtained by integrating its velocity. An algorithm of this type is called a velocity algorithm. A block diagram of a velocity algorithm for a PID controller is shown in Figure 13.10.

Velocity algorithms were commonly used in many early controllers that were built around motors. In several cases, the structure was retained by the manufacturers when technology was changed in order to maintain functional compatibility with older equipment. Another reason is that many practical issues, like wind-up protection and bumpless parameter changes, are easy to implement using the velocity algorithm. This is discussed further in Sections 3.5 and 13.5. In digital implementations velocity algorithms are also called incremental algorithms.

Incremental Algorithm

The incremental form of the PID algorithm is obtained by computing the time differences of the controller output and adding the increments

$$\Delta u(t_k) = u(t_k) - u(t_{k-1}) = \Delta P(t_k) + \Delta I(t_k) + \Delta D(t_k).$$

In some cases integration is performed externally. This is natural when a stepper motor is used. The output of the controller should then represent the increments of the control signal, and the motor implements the integrator. The increments of the proportional part, the integral part, and the derivative part are easily calculated from Equations 13.5, 13.10, and 13.16:

$$\begin{aligned}
\Delta P(t_k) &= P(t_k) - P(t_{k-1}) = K\left(b y_{sp}(t_k) - y(t_k) - b y_{sp}(t_{k-1}) + y(t_{k-1})\right)\\
\Delta I(t_k) &= I(t_k) - I(t_{k-1}) = b_{i1}\, e(t_k) + b_{i2}\, e(t_{k-1})\\
\Delta D(t_k) &= D(t_k) - D(t_{k-1}) = a_d \Delta D(t_{k-1}) - b_d \left(y(t_k) - 2y(t_{k-1}) + y(t_{k-2})\right).
\end{aligned}$$

One advantage with the incremental algorithm is that most of the computations are done using increments only. Short word-length calculations can often be used. It is only in the final stage where the increments are added that precision is needed.

Velocity Algorithms for Controllers without Integral Action

A velocity algorithm cannot be used directly for a controller without integral action because such a controller cannot keep the stationary value. This can be understood from the block diagram in Figure 13.11A, which shows a proportional controller in velocity form. Stationarity can be obtained for any value of the control error e, since the output from the derivation block is zero for any constant input. The problem can be avoided with the modification shown in Figure 13.11B. Here, stationarity is only obtained when $u = Ke + u_b$, where u_b is the bias term.

If a sampled PID controller is used, a simple version of the method illustrated in figure 13.11B is obtained by implementing the P controller as

$$\Delta u(t) = u(t) - u(t-h) = Ke(t) + u_b - u(t-h),$$

where h is the sampling period.

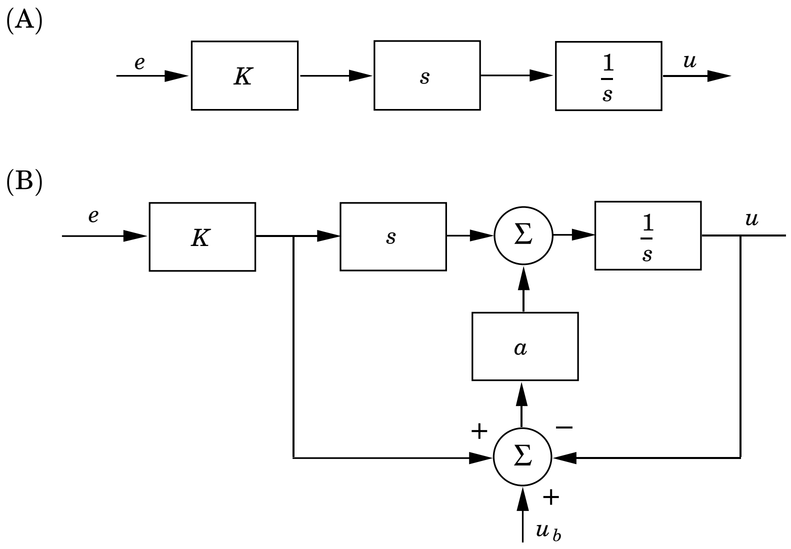

Figure 13.11 Illustrates the difficulty with a proportional controller in velocity form (A) and a way to avoid it (B).

Feedforward Control

Feedforward control was discussed in Chapter 5. In feedforward control, the control signal is composed of two terms,

$$u = u_{fb} + u_{ff}.$$

Here u_{fb} is the feedback component and u_{ff} is the feedforward component, either from a measurable disturbance or from the set point.

To avoid integrator windup, it is important that the antiwindup mechanism acts on the final control signal u, and not only on the feedback component u_{fb}.

Unfortunately, many of the block-oriented instrument systems available today have the antiwindup mechanisms inside the feedback controller blocks, without any possibility to add feedforward signals to these blocks. Hence, the feedforward signals must be added after the controller blocks. This may lead to windup. Because of this, several tricks, like feeding the feedforward signal through high-pass filters, are used to reduce the windup problem. These strategies do, however, lead to a less effective feedforward.

Incremental algorithms are efficient for feedforward implementation. By first adding *the increments* of the feedback and feedforward components,

$$\Delta u = \Delta u_{fb} + \Delta u_{ff}$$

and then forming the control signal as

$$u(t) = u(t-h) + \Delta u(t),$$

windup is avoided. This requires that the feedback control blocks have inputs for feedforward signals.

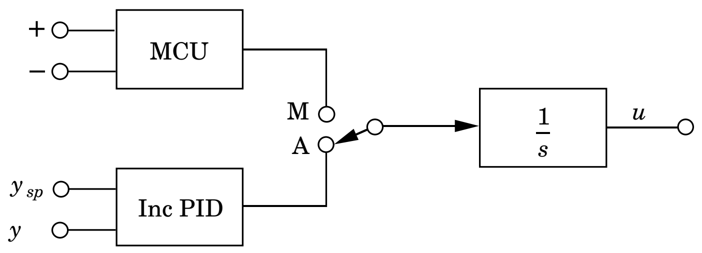

Figure 13.12 Bumpless transfer in a controller with incremental output. MCU stands for manual control unit.

13.5 Operational Aspects

Practically all controllers can be run in two modes: manual or automatic. In manual mode the controller output is manipulated directly by the operator, typically by pushing buttons that increase or decrease the controller output. A controller may also operate in combination with other controllers, such as in a cascade or ratio connection, or with nonlinear elements, such as multipliers and selectors. This gives rise to more operational modes. The controllers also have parameters that can be adjusted in operation. When there are changes of modes and parameters, it is essential to avoid switching transients. The way the mode switchings and the parameter changes are made depends on the structure chosen for the controller.

Bumpless Transfer Between Manual and Automatic

Since the controller is a dynamic system, it is necessary to make sure that the state of the system is correct when switching the controller between manual and automatic mode. When the system is in manual mode, the control algorithm produces a control signal that may be different from the manually generated control signal. It is necessary to make sure that the two outputs coincide at the time of switching. This is called *bumpless transfer*.

Bumpless transfer is easy to obtain for a controller in incremental form. This is shown in Figure 13.12. The integrator is provided with a switch so that the signals are either chosen from the manual or the automatic increments. Since the switching only influences the increments there will not be any large transients.

A similar mechanism can be used in the series, or interacting, implementation of a PID controller shown in Figure 3.3, see Figure 13.13. In this case there will be a switching transient if the output of the PD part is not zero at the switching instant.

For controllers with parallel implementation, the integrator of the PID controller can be used to add up the changes in manual mode. The controller shown in Figure 13.14 is such a system. This system gives a smooth transition between manual and automatic mode provided that the switch is made when the output of the PD block is zero. If this is not the case, there will be a switching transient.

It is also possible to use a separate integrator to add the incremental

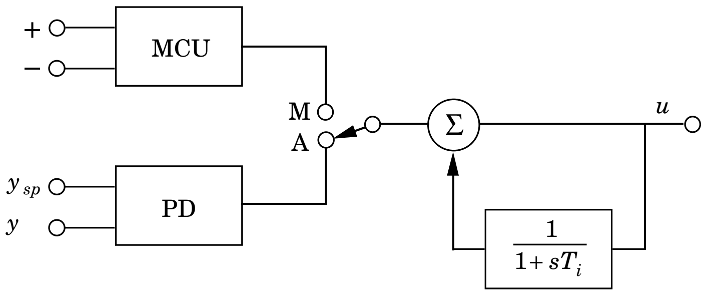

Figure 13.13 Bumpless transfer in a PID controller with a special series implementation.

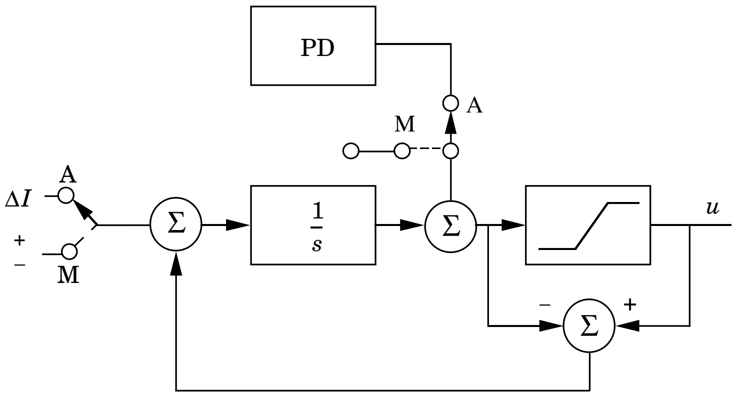

Figure 13.14 A PID controller where one integrator is used both to obtain integral action in automatic mode and to sum the incremental commands in manual mode.

changes from the manual control device. To avoid switching transients in such a system, it is necessary to make sure that the integrator in the PID controller is reset to a proper value when the controller is in manual mode. Similarly, the integrator associated with manual control must be reset to a proper value when the controller is in automatic mode. This can be realized with the circuit shown in Figure 13.15. With this system the switch between manual and automatic is smooth even if the control error or its derivative is different from zero at the switching instant. When the controller operates in manual mode, as is shown in Figure 13.15, the feedback from the output v of the PID controller tracks the output u. With efficient tracking the signal v will thus be close to u at all times. There is a similar tracking mechanism that ensures that the integrator in the manual control circuit tracks the controller output.

Bumpless Parameter Changes

A controller is a dynamical system. A change of the parameters of a dynamical system will naturally result in changes of its output. Changes in the output can be avoided, in some cases, by a simultaneous change of the state of the system. The changes in the output will also depend on the chosen realization. With a PID controller it is natural to require that there be no drastic changes in the

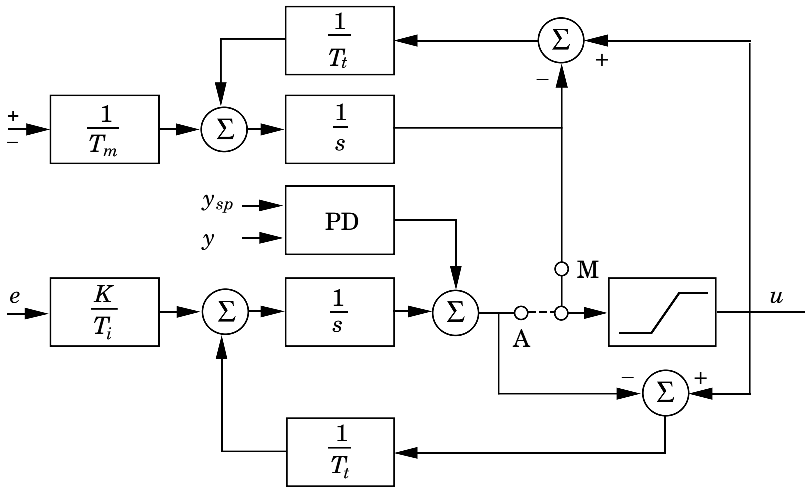

Figure 13.15 PID controller with parallel implementation that switches smoothly between manual and automatic control.

output if the parameters are changed when the error is zero. This will hold for all incremental algorithms because the output of an incremental algorithm is zero when the input is zero, irrespective of the parameter values. For a position algorithm it depends, however, on the implementation.

Assume that the state is chosen as

$$x_I = \int^t e(\tau)d\tau$$

when implementing the algorithm. The integral term is then

$$I = \frac{K}{T_i} x_I.$$

Any change of K or T_i will then result in a change of I. To avoid bumps when the parameters are changed, it is essential that the state be chosen as

$$x_I = \int^t \frac{K(\tau)}{T_i(\tau)} e(\tau)d\tau$$

when implementing the integral term.

With sensible precautions, it is easy to ensure bumpless parameter changes if parameters are changed when the error is zero. There is, however, one case where special precautions have to be taken, namely, if set-point weighting is used. To have bumpless parameter changes in such a case it is necessary that the quantity $P + I$ be invariant to parameter changes. This means that when parameters are changed, the state I should be changed as follows:

$$I_{\text{new}} = I_{\text{old}} + K_{\text{old}}(b_{\text{old}}\, y_{sp} - y) - K_{\text{new}}(b_{\text{new}}\, y_{sp} - y). \qquad (13.25)$$

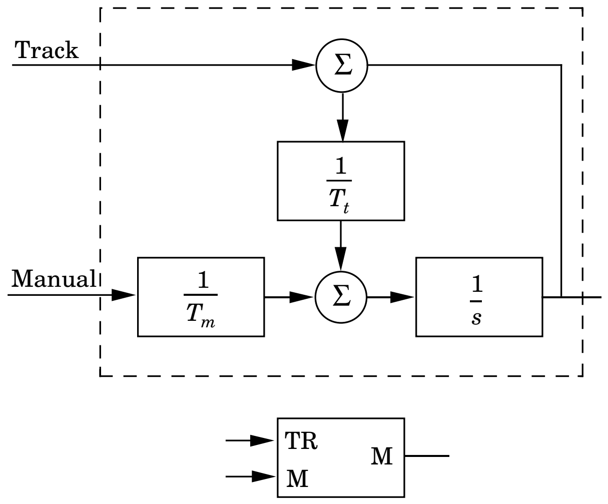

Figure 13.16 Manual control module.

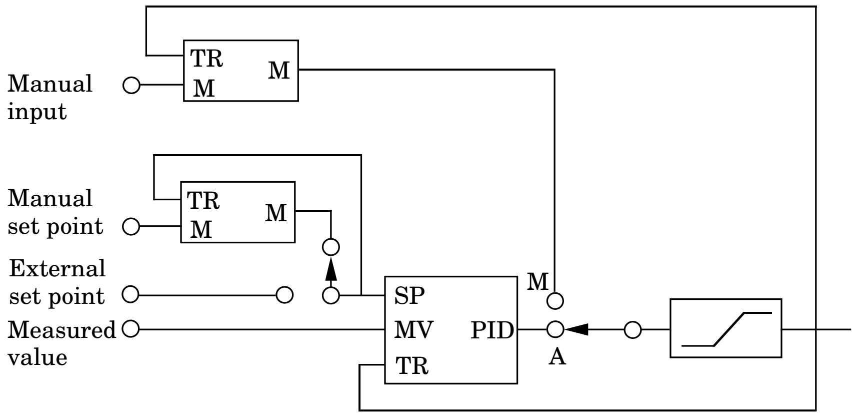

Figure 13.17 A reasonable complete PID controller with antiwindup, automatic-manual mode, and manual and external set point.

To build automation systems it is useful to have suitable modules. Figure 13.16 shows the block diagram for a manual control module. It has two inputs: a tracking input and an input for the manual control commands. The system has two parameters: the time constant T_m for the manual control input and the reset time constant T_t. In digital implementations it is convenient to add a feature so that the command signal accelerates as long as one of the increase-decrease buttons is pushed. Using the module for PID control and the manual control module in Figure 13.16, it is straightforward to construct a complete controller. Figure 13.17 shows a PID controller with internal or external set points via increase-decrease buttons and manual automatic mode. Notice that the system only has two switches.

Computer Code

As an illustration we will give computer codes for two PID controllers. A PID controller with first order filtering of the derivative term where the derivative term is approximated by backward differences is described by Equations 13.5, 13.7,13.9, and 13.13. Anti-windup is provided using the scheme described in Section 3.5. A skeleton code for the controller is given in Figure 13.18. The main loop has two states, the integral term `I`, and `x` which is used to implement derivative action. The parameters $p_1, \ldots, p_6$ are precomputed to save computing time in the main loop. These parameters have to be computed only when parameters are changed. The integral term is also reset as described by (13.25) to avoid transients when parameters are changed. The main loop in the control algorithm requires eight additions and six multiplications. Notice that the calculations are structured so that there are only three additions and two multiplications between reading the analog inputs are setting the digital output. The states are updated after setting the digital output.

A PID controller with second order filtering of the process variable is described by Equation 13.23, where the filter is implemented using backward differences and the integral term is approximated using forward differences. Anti-windup is obtained by the scheme shown in Figure 3.13. The algorithm has three states `y1`, `y2`, and `I`, which represent the states of the measurement filter and the integral term. The main loop in the control algorithm requires ten additions and seven multiplications. Using a second order filter only requires a marginal increase of computing time. The time between reading the analog inputs and setting the digital output can be reduced by changing the coordinates of the representation of the filter.

13.6 Controller Outputs

Analog Outputs

The inputs and outputs of a controller are normally analog signals, typically 0–20 mA or 4–20 mA. The main reason for using 4 mA instead of 0 mA as the lower limit is that many transmitters are designed for two-wire connection. This means that the same wire is used for both driving the sensor and transmitting the information from the sensor. It would not be possible to drive the sensor with a current of 0 mA. The main reason for using current instead of voltage is to avoid the influence of voltage drops along the wire due to resistance in the (perhaps long) wire. In pneumatic controllers, the standard range is 3–15 psi.

Thyristors and Triacs

In temperature controllers it is common practice to integrate the power amplifier with the controller. The power amplifier could be a thyristor or a triac. With a thyristor, an AC voltage is switched to the load at a given angle of the AC voltage. Since the relation between angle and power is nonlinear, it is crucial to use a transformation to maintain a linear relationship. A triac is

```
"Compute controller coefficients
    p1=K*b                              "set-point gain
    p2=K+K*Td/(Tf+h)                    "PD gain
    p3=Tf/(Tf+h)                        "filter constant
    p4=K*Td*h/((Tf+h)*(Tf+h))           "derivative gain
    p5=K*h/Ti                           "integral gain
    p6=h/Tt                             "anti-windup gain

"Bumpless parameter changes
    I=I+Kold*(bold*ysp-y)-Knew*(bnew*ysp-y)

"Control algorithm
    adin(ysp)                           "read set point
    adin(y)                             "read process variable
    v=p1*ysp-p2*y+x+I                   "compute nominal output
    u=sat(v,ulow,uhigh)                 "saturate output
    daout(u)                            "set analog output
    x=p3*x+p4*y                         "update derivative
     I=I+p5*(ysp-y)+p6*(u-v)            "update integral
```

Figure 13.18 Skeleton code for implementing a PID controller with first order filtering of the derivative term.

```
"Compute controller coefficients
    den=Tf*Tf+2*h*Tf+2*h*h              "denominator
    p1=Tf*Tf/den                        "filter constant
    p2=2*h*h/den                        "filter constant
    p3=K*h/Ti                           "integral gain
    p4=K*Td/h                           "derivative gain
    p5=h/Tt                             "anti-windup gain

"Bumpless parameter changes
    I=I+Kold*(bold*ysp-y1)-Knew*(bnew*ysp-y1)

"Control algorithm
    r=adin(ysp)                         "read set point
    y=adin(y)                           "read process variable
    x2=p1*y2+p2*(y-y1)                  "update filter state x2
    y1=y1+y2                            "update filter state x1
    v=K*(b*ysp-y1)-p4*y2+I              "compute nominal output
    u=sat(v,ulow,uhigh)                 "saturate output
    daout(u)                            "set analog output
    I=I+p3*(ysp-y1)+p5*(u-v)            "update integral
```

Figure 13.19 Skeleton code for implementing a PID controller with second order filtering of the measured signal.

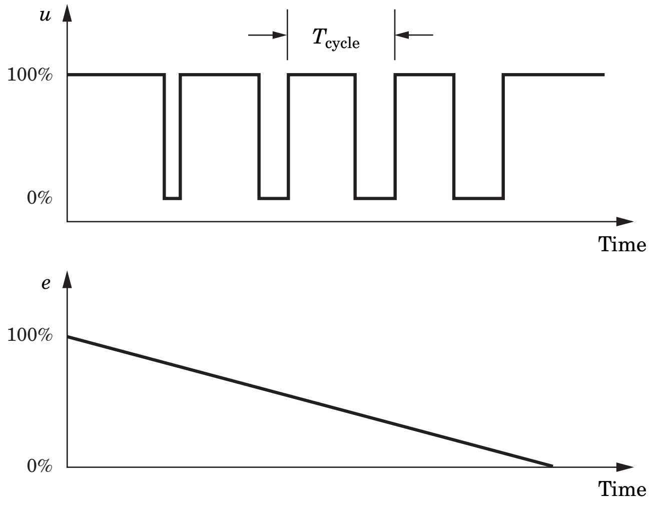

Figure 13.20 Illustration of controller output based on pulse width modulation.

also a device that implements switching of an AC signal, but only at the zero crossing. Such a device is similar to a pulse output.

Pulse Width Modulation

In some cases, such as with the triac, there is an extreme quantization in the sense that the actuator only accepts two values, on or off. In such a case, a cycle time T_{cycle} is specified, and the controller gives a pulse with width

$$T_{\text{pulse}}(t) = \frac{u(t) - u_{\min}}{u_{\max} - u_{\min}} T_{\text{cycle}}. \tag{13.26}$$

A similar, but slightly different, situation occurs when the actuator has three levels: max, min, and zero. A typical example is a motor-driven valve where the motor can stand still, go forward, or go backward.

Figure 13.20 illustrates the pulse width modulation. The figure shows the output from a P controller with pulse width modulation for different values of the control error.

Three-Position Pulse Output

If a valve is driven by a constant-speed electrical motor, the valve can be in three states: "increase," "stop," and "decrease." Control of valves with electrical actuators is performed with a controller output that can be in three states. Three-position pulse output is performed using two digital outputs from the controller. When the first output is conducting, the valve position will increase. When the second output is conducting, the valve position will decrease. If none of the outputs are conducting, the valve position is constant. The two outputs must never be conducting at the same time.

There is normally both a dead zone and a dead time in the controller to ensure that the change of direction of the motor is not too frequent and not too

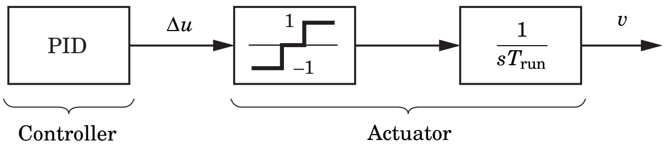

Figure 13.21 A PID controller with three-position pulse output combined with an electrical actuator.

fast. It means that the controller output is constant as long as the magnitude of the control error is within the dead zone and that the output is stopped for a few seconds before it is allowed to change direction.

A servo-motor is characterized by its running time T_{run}, which is the time it takes for the motor to go from one end position to the other. Since the servo-motor has a constant speed, it introduces an integrator in the control loop, where the integration time is determined by T_{run}. A block diagram describing a PID controller with three-position pulse output combined with an electrical actuator is shown in Figure 13.21. Suppose that we have a steady-state situation, where the output from the PID controller u is equal to the position v of the servo-motor. Suppose further that we suddenly want to increase the controller output by an amount Δu. As long as the increase-output is conducting, the output v from the servo-motor will increase according to

$$\Delta v = \frac{1}{T_{\text{run}}} \int_0^t 1\,dt = \frac{t}{T_{\text{run}}}.$$

To have Δv equal to Δu, the integration must be stopped after time

$$t = \Delta u T_{\text{run}}.$$

In a digital controller, this means that the digital output corresponding to an increasing valve position is to be conducting for n sampling periods, where n is given by

$$n = \frac{\Delta u T_{\text{run}}}{h},$$

where h is the sampling period of the controller.

To be able to perform a correct three-position pulse output, two buffers (`Buff_increase` and `Buff_decrease`) must be used to hold the number of pulses that should be sent out. A computer code for three-position pulse output is given in Figure 13.22. For the sake of simplicity, details such as dead zone and dead time are omitted in the code.

According to Figure 13.21, the controller output is Δu instead of u in the case of three-position pulse output. The integral part of the control algorithm is outside the controller, in the actuator. This solution causes no problems if the control algorithm really contains an integral part. P and PD control can not be obtained without information of the valve position, see Figure 13.11.

```
if delta_u > 0 then
    if valve_is_increasing then
        Buff_increase = Buff_increase + n;
    else
    Buff_decrease = Buff_decrease - n;
    if Buff_decrease < 0 then
        Buff_increase = - Buff_decrease;
        Buff_decrease = 0;
        valve_is_decreasing = false;
        valve_is_increasing = true;
        end;
    end;
else if delta_u < 0 then
    if valve_is_increasing then
        Buff_decrease = Buff_decrease + n;
    else
    Buff_increase = Buff_increase - n;
    if Buff_increase < 0 then
        Buff_decrease = - Buff_increase;
        Buff_increase = 0;
        valve_is_increasing = false;
        valve_is_decreasing = true;
        end;
    end;
end;
if Buff_increase > 0 then
    Increaseoutput = 1;
    Decreaseoutput = 0;
    Buff_increase = Buff_increase - 1;
else if Buff_decrease > 0 then
    Increaseoutput = 0;
    Decreaseoutput = 1;
    Buff_decrease = Buff_decrease - 1;
end;
```

Figure 13.22 Skeleton code for three-position pulse output.

13.7 Summary

In this chapter we have described implementation of PID controllers. We have followed the historical development starting with pneumatic and electronic implementation of analog controllers. Computer implementation are presented in detail including skeleton code. The reason for doing this is that many features of modern implementation have inherited several features of the old analog computers; the preference for the series form is one example.

It is interesting to consider the development of the controllers. During each phase of the development the technology has matured and improved, but knowledge has often been lost in the technology shifts. For example, it took quite a while before the importance of measurement filtering and anti-windup were appreciated in the computer implementations. One reason for it is that many details were not well documented and thus easily forgotten when technology changed. Another was that some good features were obtained automatically because of particular features of the technology. The important issues of operational aspects and human-machine interfaces have also discussed in this chapter.

13.8 Notes and References

The book [Holzbock, 1958] presents many early implementations of PID controllers using pneumatic, hydraulic, and electric technologies. Implementation of pneumatic controllers are discussed in [Lloyd and Anderson, 1971; Pavlik and Machei, 1960]. Electronic implementations are discussed by [Anderson, 1972]. It in interesting that all books mentioned above are written by equipment vendors. The paper by [Goff, 1966b] describes early efforts in digital implementation of PID controllers. Digital implementations are treated in detail in [Clarke, 1984; Hanselmann, 1987; Åström and Wittenmark, 1997]. The paper [Turnbull, 1988] gives a broad description of the development of Eurotherm's temperature controller spanning a period of more than two decades and technologies from electronic to digital. The book [Dote, 1972] describes implementation of controllers for motion control. Code for implementation on signal processors that admits very fast sampling is found in [Åström and Steingrímsson, 1991].