Chapter Ten
PID Control

Based on a survey of over eleven thousand controllers in the refining, chemicals and pulp and paper industries, 97% of regulatory controllers utilize PID feedback.

Desborough Honeywell, 2000 [58].

This chapter treats the basic properties of proportional-integral-derivative (PID) control and the methods for choosing the parameters of the controllers. We also analyze the effects of actuator saturation and time delay, two important features of many feedback systems, and methods for compensating for these effects. Finally, we will discuss the implementation of PID controllers as an example of how to implement feedback control systems using analog or digital computation.

10.1 BASIC CONTROL FUNCTIONS

PID control, which was introduced already in Section 1.5 and has been used in several examples, is by far the most common way of using feedback in engineering systems. It appears in simple devices and in large factories with thousands of controllers. PID controllers appear in many different forms: as a standalone controller, as part of hierarchical, distributed control systems or built into embedded components. Most PID controllers do not use derivative action so they should strictly speaking be called PI controllers; we will however use PID as a generic term for this class of controller. There is also growing evidence that PID control appears in biological systems [201].

Block diagrams of closed loop systems with PID controllers are shown in Figure 10.1. The control signal u for the system in Figure 10.1a is formed entirely from the error e; there is no feedforward term (which would correspond to $k_r r$ in the state feedback case). A common alternative in which proportional and derivative action do not act on the reference is shown in Figure 10.1; combinations of the schemes will be discussed in Section 10.5. The command signal r is called the reference value in regulation problems, or the *setpoint* in literature of PID control. The input-output relation for an ideal PID controller with error feedback is

$$u = k_p e + k_i \int_0^t e(\tau)\,d\tau + k_d\ fracdedt = k_p\Big(e + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de}{dt}\Big). \quad (10.1)$$

The control action is thus the sum of three terms: proportional feedback, the integral term, and derivative action. For this reason PID controllers were originally called *three term controllers*. The controller parameters are the proportional gain

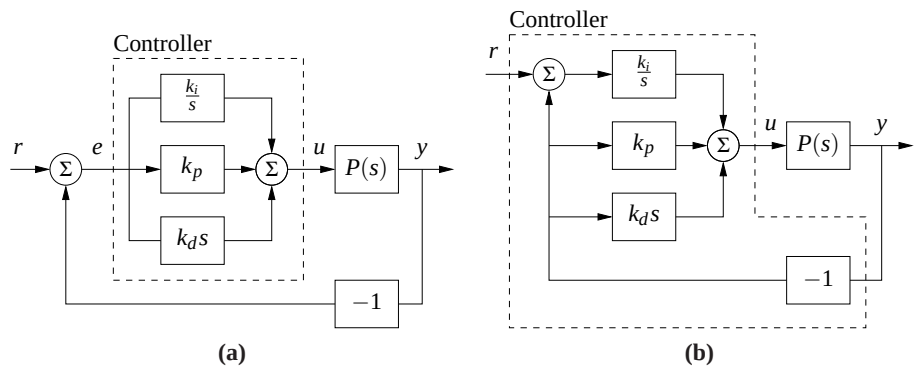


Figure 10.1: Block diagrams of closed loop systems with ideal PID controllers. Both controller have one output the control signal u. The controller in (a), which is based on error feedback, has one input, the control error $e = r - y$. For this controller proportional, integral and derivative action acts on the error $e = r - y$. The two-degree-of-freedom controller in (b), has two inputs, the reference r and the process output y. Integral action acts on the error, but proportional and derivative action acts on the process output y.

k_p, the integral gain k_i and the derivative gain k_d. The time constants T_i and T_d, called integral time (constant) and derivative time (constant), are sometimes used instead of the integral and derivative gains.

The controller (10.1) represents an idealized controller. It is a useful abstraction for understanding the PID controller, but several modifications must be made in order to obtain a controller that is practically useful. Before discussing these practical issues we we will develop some intuition about PID control.

We start by considering pure proportional feedback. Figure 10.2a shows the responses of the process output to a unit step in the reference value for a system with pure proportional control at different gain settings. All of the systems have error feedback. In the absence of a feedforward term, the output never reaches the reference and hence we are left with nonzero steady state error. Letting the process and the controller have transfer functions $P(s)$ and $C(s)$, the transfer function from reference to output is

$$G_{yr} = \frac{PC}{1+PC}. \tag{10.2}$$

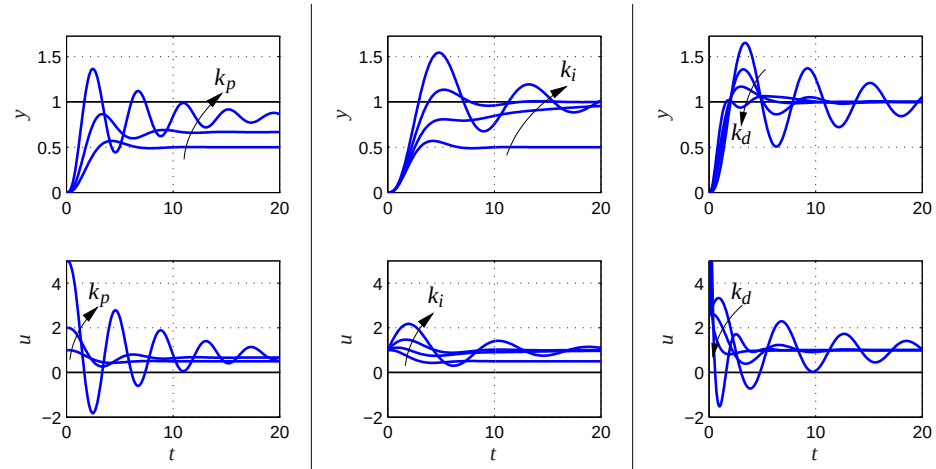
and thus the steady state error for a unit step is

$$1 - G_{yr}(0) = \frac{1}{1+k_p P(0)}.$$

For the system in Figure 10.2a with gains $k_p = 1$, 2 and 5, the steady state error is 0.5, 0.33 and 0.17. The error decreases with increasing gain, but the system also becomes more oscillatory. Notice in the figure that the initial value of the control signal equals the controller gain.

To avoid having a steady state error, the proportional term can be changed to

$$u(t) = k_p e(t) + u_{\text{ff}}, \tag{10.3}$$



(a) Proportional control

(b) PI control

(c) PID control

Figure 10.2: Responses to step changes in the reference value for system with a proportional controller (a), PI controller (b) and PID controller (c). The process has the transfer function $P(s) = 1/(s+1)^3$, the proportional controller (left) had parameters $k_p = 1$, 2 and 5, the PI controller has parameters $k_p = 1$, $k_i = 0$, 0.2, 0.5 and 1, and the PID controller has parameters are $k_p = 2.5$, $k_i = 1.5$ and $k_d = 0$, 1, 2 and 4.

where u_{ff} is a feedforward term that is adjusted to give the desired steady state value. If we choose $u_{\text{ff}} = r/P(0) = k_r r$, then the output will be exactly equal to the reference value, as it was in the state space case, provided that there are no disturbances. However, this requires exact knowledge of the process dynamics, which is usually not available. The parameter u_{ff}, called *reset* in the PID literature, must therefore be adjusted manually.

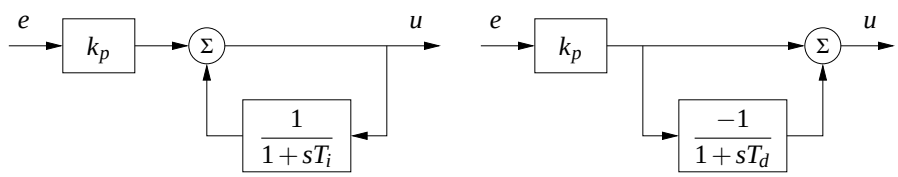
As we saw in Sections 6.4, integral action guarantees that the process output agrees with the reference in steady state and provides an alternative to the feedforward term. Since this result is so important we will provide a general proof. Consider the controller given by equation (10.1). Assume that there exist a steady state with $u = u_0$ and $e = e_0$. It then follows from equation (10.1) that

$$u_0 = k_p e_0 + k_i e_0 t,$$

which is a contradiction unless e_0 or k_i are zero. We can thus conclude that with integral action the error will be zero if it reaches a steady state. Notice that we have not made any assumptions about linearity of the process or the disturbances. We have, however assumed that an equilibrium exists. Using integral action to achieve zero steady state error is much better than using feedforward, which requires precise knowledge of process parameters.

The effect of integral action can also be understood from frequency domain analysis. The transfer function of the PID controller is

$$C(s) = k_p + \frac{k_i}{s} + k_d s. \tag{10.4}$$



(a) Automatic reset **(b)** Proportional-derivative

Figure 10.3: Implementation of PI and PD controllers. The block diagram on the left shows how integral action is implemented using *positive feedback* with a first order system, sometimes called automatic reset. The block diagram on the right shows how derivative action can be implemented by taking differences between a static system and a first order system.

The controller has infinite gain at zero frequency ($C(0) = \infty$) and it then follows from equation (10.2) that $G_{yr}(0) = 1$, which implies that there is no steady state error for a step input.

Integral action can also be viewed as a method for generating the feedforward term u_{ff} in the proportional controller (10.3) automatically. One way to do this is shown in Figure 10.3a, where the controller output is low-pass filtered and fed back with positive gain. This implementation, called *automatic reset*, was one of the early inventions of integral control. The transfer function of the system in Figure 10.3a is obtained by block diagram algebra; we have

$$G_{ue} = k_p \frac{1+sT_i}{sT_i} = k_p + \frac{k_p}{sT_i},$$

which is the transfer function for a PI controller.

The properties of integral action are illustrated in Figure 10.2b for a step input. The proportional gain is constant, $k_p = 1$, and the integral gains are $k_i = 0$, 0.2, 0.5 and 1. The case $k_i = 0$ corresponds to pure proportional control, with a steady state error of 50%. The steady state error is eliminated when integral gain action is used. The response creeps slowly towards the reference for small values of k_i and goes faster for larger integral gains, but the system also becomes more oscillatory.

Integral gain k_i is a useful measure for attenuation of load disturbances. Consider a closed loop system under PID control and assume that the system is stable and initially at rest with all signals being zero. Apply a unit step disturbance at the process input. After a transient the process output goes to zero and the controller output settles at a value that compensates for the disturbance. It follows from (10.1) that

$$u(\infty) = k_i \int_0^\infty e(t)dt.$$

The integrated error is thus inversely proportional to integral gain k_i. The integral gain is thus a measure of the effectiveness of disturbance attenuation. A large gain k_i attenuates disturbances effectively but too large a gain gives oscillatory behavior, poor robustness and possibly instability.

We now return to the general PID controller and consider the effect of the derivative term, k_d. Recall that the original motivation for derivative feedback was

to provide predictive action. Notice that the combination of the proportional and the derivative terms can be written as

$$u = k_p e + k_d \frac{de}{dt} = k_p\big(e + T_d \frac{de}{dt}\big) = k_p e_p,$$

where $e_p(t)$ can be interpreted as a prediction of the error at time $t + T_d$ by linear extrapolation. The prediction time $T_d = k_d/k_p$ is the derivative time constant of the controller.

Derivative action can be implemented by taking the difference between the signal and its low-pass filtered version as shown in Figure 10.3b. The transfer function for the system is

$$G_{ue}(s) = k_p\Big(1 - \frac{1}{1+sT_d}\Big) = k_p \frac{sT_d}{1+sT_d}. \tag{10.5}$$

The system thus has the transfer function $G(s) = sT_d/(1+sT_d)$, which approximates a derivative for low frequencies ($|s| < T_d$).

Figure 10.2c illustrates the effect of derivative action: the system is oscillatory when no derivative action is used and it becomes more damped as derivative gain is increased. Performance deteriorates if derivative gain is too high. When the input is a step the controller output generated by the derivative term will be an impulse. This is clearly visible in Figure 10.2c. The impulse can be avoided by using the controller configuration shown in Figure 10.1b.

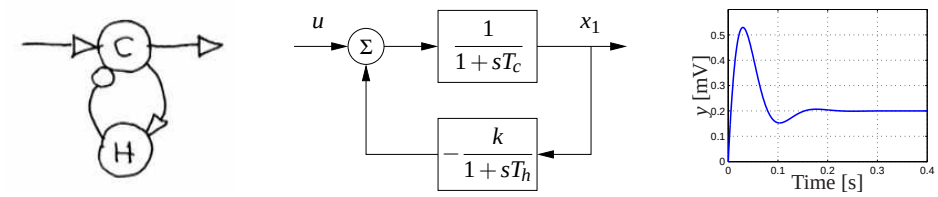
Although PID control was developed in the context of engineering applications, it also appears in nature. Disturbance attenuation by feedback in biological systems is often called adaptation. A typical example is the pupillary reflex discussed in Example 8.11 where it is said that the eye adapts to changing light intensity. Analogously, feedback with integral action is called perfect adaptation [201]. In biological systems proportional, integral and derivative action is generated by combining subsystems with dynamical behavior similar to what is done in engineering systems. For example, PI action can be generated by the interaction of several hormones [69].

Example 10.1 PD action in the retina

The response of cone photo receptors in the retina is an example where proportional and derivative action is generated by a combination of cones and horizontal cells. The cones are the primary receptors stimulated by light, the cones in turn stimulate the horizontal cells and the horizontal cells give inhibitory (negative) feedback to the cones. A schematic diagram of the system is shown in Figure 10.4a. The system can be modeled by ordinary differential equations by representing neuron signals by continuous variables representing the average pulse rate. In [199] it is shown that the system can be represented by the differential equations

$$\frac{dx_1}{dt} = \frac{1}{T_c}(-x_1 - kx_2 + u), \qquad \frac{dx_2}{dt} = \frac{1}{T_h}(x_1 - x_2),$$

where u is the light intensity and x_1 and x_2 are the average pulse rates from the cones and the horizontal cells. A block diagram of the system is shown in Fig-



(a) **(b)** **(c)**

Figure 10.4: Schematic diagram of cone photo receptors (C) and horizontal cells (H) in the retina. Excitatory feedback is indicated by arrows and inhibitory feedback by circles in the schematic diagram in (a). A block diagram is shown in (b) and the step response in (c).

ure 10.4b. The step response of the system shown in Figure 10.4c shows that the system has a large initial response followed by a lower constant steady state response typical of proportional and derivative action. The parameters used in the simulation are $k = 4$, $T_c = 0.025$ and $T_c = 0.08$. ∇

10.2 SIMPLE CONTROLLERS FOR COMPLEX SYSTEMS

Many of the design methods discussed in previous chapters have the property that the complexity of the controller is directly reflected by the complexity of the model. When designing controllers by output feedback in Chapter 7 we found for single-input single-output systems that the order of the controller was the same as the order of the model, possibly one order higher if integral action was required. Applying similar design methods for PID control will require that we have have low order models of the processes to be able to easily analyze the results.

Low order models can be obtained from first principles. Any stable system can be modeled by a static system if its inputs are sufficiently slow. Similarly a first order model is sufficient if storage of mass, momentum or energy can be captured by only one variable; typical examples are the velocity of a car on a road, angular velocity of a stiff rotational systems, level in a tank and concentration in a volume with good mixing. System dynamics are of second order if storage of mass, energy and momentum can be captured by two state variable; typical examples are position of a car on the road, stabilization of stiff satellites, levels in two connected tanks and two compartment models. A wide range of techniques for model reduction also available. In this chapter we will focus on design techniques were we simplify the models to capture the essential properties that are needed for PID design.

We begin by analyzing the case of integral control. A stable system can be controlled by an integrating controller provided that the requirements on the closed loop system are modest. To design the controller we assume that the transfer function of the process is a constant $K = P(0)$. The loop transfer function under integral control then becomes Kk_i/s and the closed loop characteristic polynomial is sim-

ply $s+Kk_i$. Specifying performance by the desired time constant T_{cl} of the closed loop system we find that the integral gain is given by

$$k_i = 1/(T_{cl}P(0)).$$

The analysis requires that T_{cl} is large enough that the process transfer function can be approximated by a constant.

For systems that are not well-represented by a constant gain, another way to find a suitable value of integral gain is to make a Taylor series expansion of the loop transfer function

$$L(s) = \frac{k_i P(s)}{s} \approx \frac{k_i(P(0)+sP'(0))}{s} = k_i P'(0) + \frac{k_i P(0)}{s}.$$

Choosing $k_i P'(0) = -0.5$ gives a system with good robustness as will be discussed in Section **??**. The controller gain is then given by

$$k_i = -\frac{1}{2P'(0)} \tag{10.6}$$

and the expected closed loop time constant is $T_{cl} \approx -2P'(0)/P(0)$. This approach is useful when the process has a pole at the origin, as illustrated in the following example.

Example 10.2 Integrating control of AFM in tapping mode
A simplified model of the dynamics of the vertical motion of an atomic force microscope in tapping mode was discussed in Exercise 9.5. The transfer function for the system dynamics is

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

where $a = \zeta\omega_0$, and $\tau = 2\pi n/\omega_0$ and the gain has been normalized to 1. We have $P(0) = 1$ and $P'(0) = -\tau/2 - 1/a$, and it follows from (10.6) that the integral gain is $k_i = a/(2+a\tau)$. A Nyquist plot and Bode plot for the resulting loop transfer function are shown in Figure 10.5. ∇

A first order system has the transfer function

$$P(s) = \frac{b}{s+a}.$$

With a PI controller the closed loop system has the characteristic polynomial

$$s(s+a) + bk_p s + bk_i s = s^2 + (a+bk_p)s + bk_i.$$

The closed loop poles can thus be assigned arbitrary values by proper choice of the controller gains. Requiring that the closed loop system has the characteristic polynomial

$$s^2 + a_1 s + a_2^2,$$

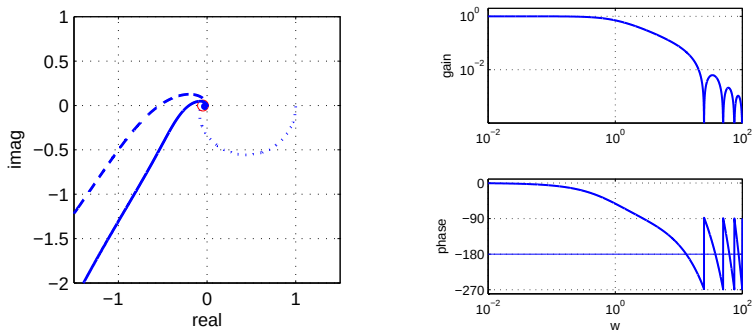


Figure 10.5: Integral control for AFM in tapping mode. An integral controller is designed based on the slope of the process transfer function at 0. The controller gives good robustness properties based on a very simple analysis.

we find that the controller parameters are

$$k_p = \frac{a_1 - a}{b}, \qquad k_i = \frac{a_2}{b}. \tag{10.7}$$

If we require a response of the closed loop system that is slower than that of the open loop system, a reasonable choice is $a_1 = a + \alpha$ and $a_2 = \alpha a$. If a response that is faster that the open loop system is required, it is reasonable to choose $a_1 = 2\zeta_0\omega_0$ and $a_2 = \omega_0^2$, where ω_0 and ζ_0 are undamped natural frequency and relative damping of the dominant mode. These choices have significant impact on the robustness of the system and will be discussed in Section 12.4. An upper limit to ω_0 is given by the validity of the model. Large values of ω_0 will require fast control actions and actuators may saturate if the value is too large. A first order model is unlikely to represent the true dynamics for high frequencies. We illustrate the design by an example.

Example 10.3 Cruise control using PI feedback

Consider the problem of maintaining the speed of a car as it goes up a hill. In Example 5.14 we found that there was little difference between the linear and nonlinear models when investigating PI control provided that the throttle did not reach the saturation limits. A simple linear model of a car was given in Example 5.11:

$$\frac{d(v - v_e)}{dt} = -a(v - v_e) + b(u - u_e) - g\theta, \tag{10.8}$$

where v is the velocity of the car, u is the input from the engine and θ is the slope of the hill. The parameters were $a = 0.0101$, $b = 1.3203$, $g = 9.8$, $v_e = 20$, and $u_e = 0.1616$. This model will be used to find suitable parameters of a vehicle speed controller. The transfer function from throttle to velocity is thus a first order system. Since the open loop dynamics is so slow it is natural to specify a faster closed loop system by requiring that the closed loop system is of second order with relative damping ζ and undamped natural frequency ω_0. The controller gains are given by (10.7).

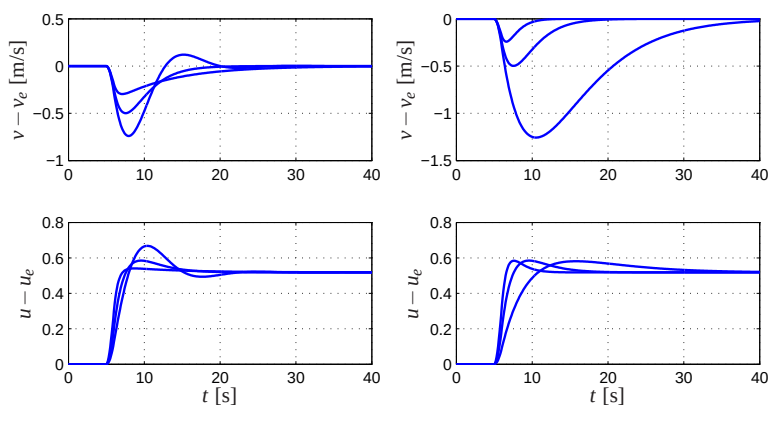


Figure 10.6: Cruise control using PI feedback. The step responses for the error and input illustrate the effect of parameters $\zeta_0 = 1$ and ω_0 on the response of a car with cruise control. A change in road slope from 0° to 4° is applied between $t = 5$ and 6 s. The plot on the left shows the response for $\omega_0 = 0.5$ and $\zeta_0 = 0.5$, 1 and 2. Choosing $\zeta_0 = 1$ gives no overshoot. The plot on the right shows the response for $\zeta_0 = 1$ and $\omega_0 = 0.2$, 0.5 and 1.0.

Figure 10.6 shows the velocity and the throttle for a car that initially moves on a horizontal road and encounters a hill with slope 4° at time $t = 6$ sec. To design a PI controller we choose $\zeta_0 = 1$ to obtain a response without overshoot, as shown in Figure 10.6a. The choice of ω_0 is a compromise between response speed and control actions: a large value gives a fast response but it requires fast control action. The trade-off is is illustrated in Figure 10.6b. The largest velocity error decreases with increasing ω_0, but the control signal also changes more rapidly. In the simple model (10.8) it was assumed that the force responds instantaneously to throttle commands. For rapid changes there may be additional dynamics that have to be accounted for. There are also physical limitations to the rate of change of the force, which also restricts the admissible value of ω_0. A reasonable choice of ω_0 is in the range of 0.5 to 1.0. Notice in Figure 10.6 that even with $\omega_0 = 0.2$ the largest velocity error is only 1 m/s.

∇

A PI controller can also be used for a process with second order dynamics, but there will be restrictions on possible locations of closed loop poles, as shown in Exercise 10.2. Using a PID controller it is possible to control a system of second order in such a way that the closed loop poles have arbitrary locations, see Exercise 10.3.

Instead of finding a low order model and designing controllers for them we can also use a high order model and only attempt to place a few dominant poles. An integrating controller has one parameter and it is possible to position one pole. Consider a process with the transfer function $P(s)$. The loop transfer function with an integrating controller is $L(s) = k_i P(s)/s$. The roots of the closed loop characeristic polynomial are the roots of $s + k_i P(s) = 0$. Requiring that $s = -a$ is

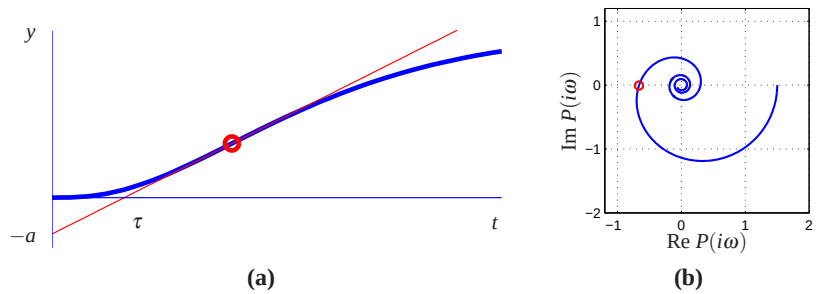


Figure 10.7: Illustration of Ziegler-Nichols step and frequency response experiments. The unit step response in (a) is characterized by the parameters a and T_{del}. The frequency response method characterizes process dynamics by the point of the Nyquist curve of the process transfer function first intersects the negative real axis and the frequency ω_c where this occurs.

a root, we find that the controller gain should be chosen as

$$k_i = \frac{a}{P(-a)}. \tag{10.9}$$

The pole $s = -a$ will be dominant if a is small. A similar approach can be applied to PI and PID controllers.

10.3 PID TUNING

Users of control systems are frequently faced with the task of adjusting the controller parameters to obtain a desired behavior. There are many different ways to do this. One approach is to go through the conventional steps of modeling and control design as described in the previous section. Since the PID controller has so few parameters, a number of special empirical methods have also been developed for direct adjustment of the controller parameters. The first tuning rules were developed by Ziegler and Nichols [204]. Their idea was to make a simple experiment, extract some features of process dynamics from the experiment and determine controller parameters from the features.

Ziegler-Nichols' Tuning

Ziegler and Nichols developed two methods for controller tuning in the 1940s based on simple characterization of process dynamics in the time and frequency domains.

The time domain method is based on a measurement of part of the open loop unit step response of the process, as shown in Figure 10.7a. The step response is measured by applying a unit step input to the process and recording the response. The response is characterized by parameters a and τ, which are the intercepts of the steepest tangent of the step response with the coordinate axes. The parame-

Table 10.1: Ziegler-Nichols tuning rules. (a) The step response methods give the parameters in terms of the intercept a and the apparent time delay τ. (b) The frequency response method gives controller parameters in terms of critical gain k_c and critical period T_c.

Type	ak_p	T_i/τ	T_d/τ
P	1		
PI	0.9	3	
PID	1.2	2	0.5

(a) Step response method

Type	k_p/k_c	T_i/T_c	T_d/T_c
P	0.5		
PI	0.4	0.8	
PID	0.6	0.5	0.125

(b) Frequency response method

ter τ is an approximation of the time delay of the system and a/τ is the steepest slope of the step response. Notice that it is not necessary to wait until steady state is reached to find the parameters, it suffices to wait until the response has had an inflection point. The controller parameters are given in Table 10.1. The parameters were obtained by extensive simulation of a range of representative processes. A controller was tuned manually for each process and it was then attempted to correlate the controller parameters with a and τ.

In the frequency domain method a controller is connected to the process, the integral and derivative gains are set to zero, and the proportional gain is increased until the system starts to oscillate. The critical value of the proportional gain k_c is observed together with the period of oscillation T_c. It follows from Nyquist's stability criterion that the loop transfer function $L = k_c P(s)$ intersects the critical point for the frequency $\omega_c = 2\pi/T_c$. The experiment thus gives the point on the Nyquist curve of the process transfer function where the phase lag is 180°, as shown in Figure 10.7b.

The Ziegler-Nichols methods had a huge impact when they were introduced in the 1940s. The rules were simple to use and gave initial conditions for manual tuning. The ideas were adopted by manufacturers of controllers for routine use. The Ziegler-Nichols tuning rules have unfortunately two severe drawbacks: too little process information is used and the closed loop systems that are obtained lack robustness.

The step response method can be improved significantly by characterizing the unit step response by parameters K, τ and T in the model

$$P(s) = \frac{K}{1+sT} e^{-\tau s}. \tag{10.10}$$

The parameters can be obtained by fitting the model to a measured step response. Notice that the experiment takes longer time than the experiment in Figure 10.7a because to determine K it is necessary to wait until the steady state has been reached. Also notice that the intercept a in the Ziegler-Nichols rule is given by $a = K\tau/T$.

The frequency response method can be improved by measuring more points

on the Nyquist curve, for example the zero frequency gain K or the point where the process has 90° phase lag. This latter point can be obtained by connecting an integrating controller and increasing its gain until the system reaches the stability limit. The experiment can also be automated by using relay feedback, as will be discussed later in this section.

There are many versions of improved tuning rules. As an illustration we give the following rules for PI control from [18]:

$$\begin{aligned} k_p &= \frac{0.17\tau + 0.28T}{K\tau} \quad \left(\frac{0.9T}{K\tau}\right), & k_i &= \frac{0.46\tau + 0.02T}{K\tau^2} \quad \left(\frac{0.3}{K\tau}\right) \\ k_p &= 0.16k_c \quad \left(0.4k_c\right), & k_i &= \frac{0.16k_c + 0.72K}{T_c} \quad \left(\frac{0.5k_c}{T_c}\right) \end{aligned} \tag{10.11}$$

The values for the Ziegler-Nichols rule are given in parentheses. Notice that the improved formulas typically give lower controller gains than the Ziegler-Nichols method. The integral gain is higher for systems whose dynamics are delay dominated, $\tau \gg T$.

Example 10.4 PI control of AFM in tapping mode

A simplified model of the dynamics of the vertical motion of an atomic force microscope in tapping mode was discussed in Example 10.2. The transfer function is normalized by choosing $1/a$ as the time unit. The normalized transfer function is

$$P(s) = \frac{1 - e^{-sT_n}}{sT_n(s+1)}.$$

where $T_n = 2n\pi a/\omega_0 = 2n\pi\zeta$. The Nyquist plot of the transfer function is shown in Figure 10.8a for $z = 0.002$ and $n = 20$. The leftmost intersection of the Nyquist curve with the real axis occurs at $\operatorname{Re} s = -0.0461$ for $\omega = 13.1$. The critical gain is thus $k_c = 21.7$ the critical period is $T_c = 0.48$. Using Ziegler-Nichols tuning rule we find the parameters $k_p = 8.87$ and $k_i = 22.6$ ($T_i = 0.384$) for a PI controller. With this controller the stability margin is $s_m = 0.31$, which is quite small. The step response of the controller is shown in Figure 10.8. Notice in particular that there is a large overshoot in the control signal.

The modified Ziegler-Nichols rule (10.11) gives the controller parameters $k = 3.47$ and $k_i = 8.73$ ($T_i = 0.459$) and the stability margin becomes $s_m = 0.61$. The step response with this controller is shown in Figure 10.8. A comparison of the responses obtained with the original Ziegler Nichols rule shows that the overshoot has been reduced. Notice that the control signal reaches its steady state value almost instantaneously. It follows from Example 10.2 that a pure integrating controller has the normalized gain $k_i = 1/(2 + T_n) = 0.44$. Comparing this with the gains of a PI controller we can conclude that a PI controller gives much better performance than a pure integrating controller. ∇

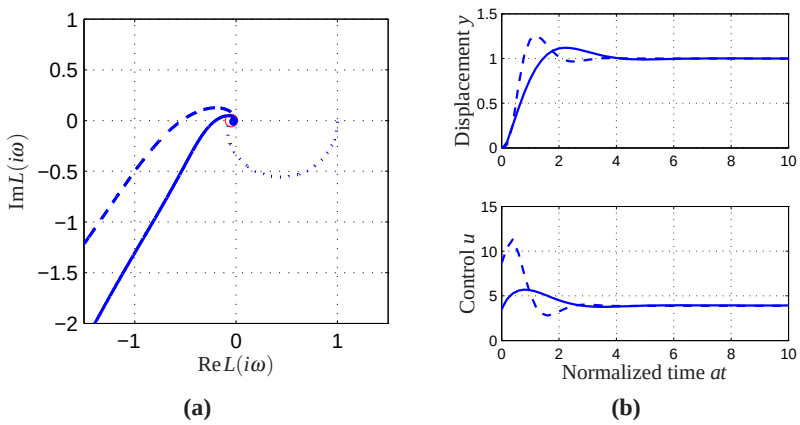


Figure 10.8: PI control of an AFM in tapping mode. Nyquist plots (a) and step responses (b) for PI control of the vertical motion of an atomic force microscope in tapping mode. The averaging parameter is $n = 20$. Results with Ziegler-Nichols tuning are shown in dashed lines, and modified Ziegler-Nichols tuning is shown in full lines. The Nyquist plot of the process transfer function is shown in dotted lines.

Relay Feedback

The Ziegler-Nichols frequency response method increases the gain of a proportional controller until oscillation to determine the critical gain k_c and the corresponding period T_c or equivalently the point where the Nyquist curve intersects the negative real axis. One way to obtain this information automatically is to connect the process in a feedback loop with a nonlinear element having a relay function as shown in Figure 10.9a. For many systems there will then be an oscillation, as shown in Figure 10.9b, where the relay output u is a square wave and the process output y is close to a sinusoid. Moreover the input and the output are out of phase, which means that the system oscillates with the critical period T_c, where the process has a phase lag of 180°. Notice that an oscillation with constant period is established quickly.

The critical period is simply the period of the oscillation. To determine the critical gain we expand the square wave relay output in a Fourier series. Notice

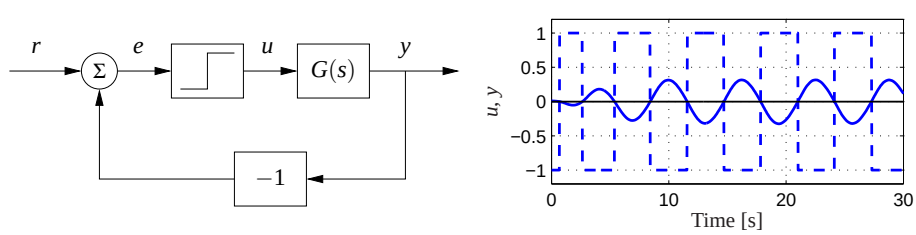


Figure 10.9: Block diagram of a process with relay feedback (left) and typical signals (right). The process output y is solid and the relay output u is dashed. Notice that the signals u and y have opposite phase.

in the figure that the process output is practically sinusoidal because the process attenuates higher harmonics effectively. It is then sufficient to consider only the first harmonic component of the input. Letting d be the relay amplitude, the first harmonic of the square wave input has amplitude $4d/\pi$. If a is the amplitude of the process output, the process gain at the critical frequency $\omega_c = 2\pi/T_c$ is $|P(i\omega_c)| = \frac{\pi a}{4d}$ and the critical gain is

$$K_c = \frac{4d}{a\pi}. \tag{10.12}$$

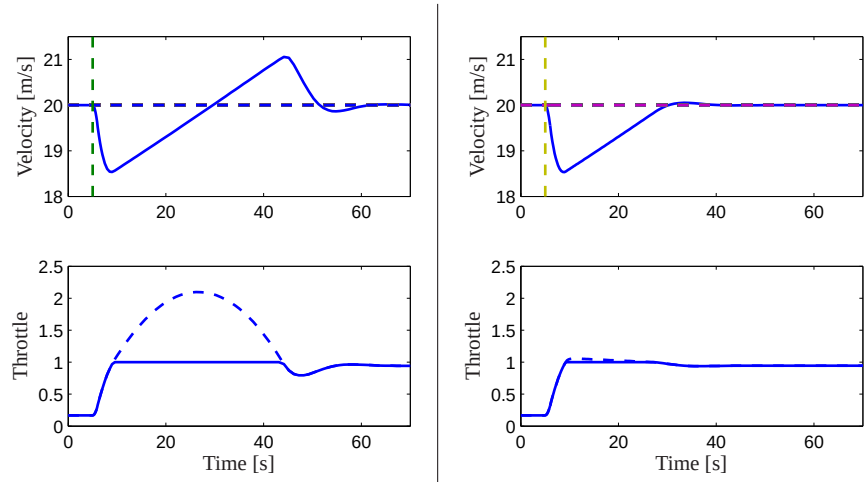
Having obtained the critical gain K_c and the critical period T_c the controller parameters can then be determined using the Ziegler-Nichols rules. Improved tuning can be obtained by fitting a model to the data obtained from the relay experiment.

The relay experiment can be automated. Since the amplitude of the oscillation is proportional to the relay output, it is easy to control it by adjusting the relay output. Automatic tuning based on relay feedback is used in many commercial PID controllers. Tuning is accomplished simply by pushing a button that activates relay feedback. The relay amplitude is automatically adjusted to keep the oscillations sufficiently small and the relay feedback is switched to a PID controller as soon as the tuning is finished.

10.4 INTEGRATOR WINDUP

Many aspects of a control system can be understood from linear models. There are, however, some nonlinear phenomena that must be taken into account. These are typically limitations in the actuators: a motor has limited speed, a valve cannot be more than fully opened or fully closed, etc. For a system that operates over a wide range of conditions, it may happen that the control variable reaches the actuator limits. When this happens the feedback loop is broken and the system runs in open loop because the actuator will remain at its limit independently of the process output as long as the actuator remains saturated. The integral term will also build up since the error is typically nonzero. The integral term and the controller output may then become very large. The control signal will then remain saturated even when the error changes and it may take a long time before the integrator and the controller output come inside the saturation range. The consequence is that there are large transients. This situation is referred to as *integrator windup*, illustrated in the following example.

Example 10.5 Cruise control
The windup effect is illustrated in Figure 10.10, which shows what happens when a car encounters a hill that is so steep (6°) that the throttle saturates when the cruise controller attempts to maintain speed. When encountering the slope at time $t = 5$ the velocity decreases and the throttle increases to generate more torque. However, the torque required is so large that the throttle saturates. The error decreases slowly because the torque generated by the engine is just a little larger than the torque



(a) Windup **(b)** Anti-windup

Figure 10.10: Simulation of PI cruise control with windup (left) and anti-windup (right). The figure shows the speed v and the throttle u for a car that encounters a slope that is so steep that the throttle saturates. The controller output is dashed. The controller parameters are $k_p = 0.5$ and $k_i = 0.1$.

required to compensate for the gravity. The error is large and the integral continues to build up until the error reaches zero at time 30, but the controller output is still larger than the saturation limit and the actuator remains saturated. The integral term starts to decrease and at time 45 and the velocity settles quickly to the desired value. Notice that it takes considerable time before the controller output comes into the range where it does not saturate, resulting in a large overshoot. ∇

There are many ways to avoid windup. One method is illustrated in Figure 10.11: 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 k_t. The signal e_s is zero when there is no saturation and the extra feedback loop has no effect on the system. When the actuator saturates, the signal e_s is fed back to the integrator in such a way that e_s goes towards zero. This implies that controller output is kept close to the saturation limit. The controller output will then change as soon as the error changes sign and integral windup is avoided.

The rate at which the controller output is reset is governed by the feedback gain, k_t; a large value of k_t gives a short reset time. The parameter k_t cannot be too large because measurement error can then cause an undesirable reset. A reasonable choice is to choose k_i as a fraction of $1/T_i$. We illustrate how integral windup can be avoided by investigating the cruise control system.

Example 10.6 Cruise control

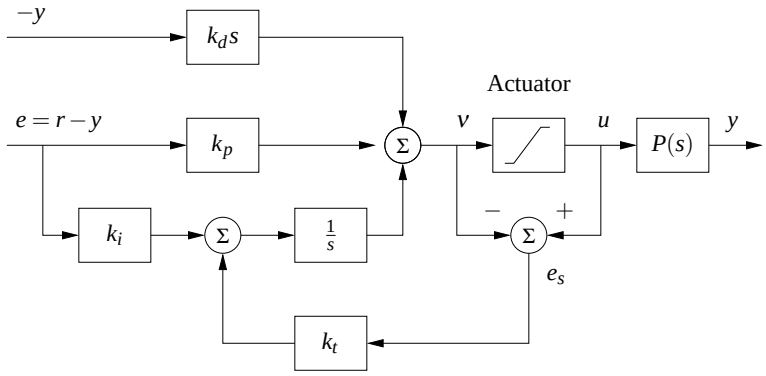


Figure 10.11: PID controller with filtered derivative and anti-windup. The input to the integrator $(1/s)$ consists of the error term plus a "reset" based on input saturation. If the actuator is not saturated then $e_s = u - v$, otherwise e_s will decrease the integrator input to prevent windup.

Figure 10.10b shows what happens when a controller with anti-windup is applied to the system simulated in Figure 10.10a. Because of the feedback from the actuator model, the output of the integrator is quickly reset to a value such that the controller output is at the saturation limit. The behavior is drastically different from that in Figure 10.10a and the large overshoot is avoided. The tracking gain is $k_t = 2$ in the simulation. ∇

10.5 IMPLEMENTATION

There are many practical issues that have to be considered when implementing PID controllers. They have been developed over time based on practical experiences. In this section we consider some of the most common. Similar considerations also apply to other types of controllers.

Filtering the Derivative

A drawback with derivative action is that an ideal derivative has high gain for high frequency signals. This means that high frequency measurement noise will generate large variations of the control signal. The effect of measurement noise may be reduced by replacing the term $k_d s$ by $k_d s/(1 + sT_f)$, which can be interpreted as an ideal derivative of a low-pass filtered signal. For small s the transfer function is approximately $k_d s$ and for large s it is equal to k_d/T_f. The approximation acts as a derivative for low-frequency signals and as a constant gain for the high frequency signals. The filtering time is chosen as $T_f = (k_d/k)/N$, with N in the range of 2 to 20. Filtering is obtained automatically if the derivative is implemented by taking the difference between the signal and its filtered version as shown in Figure 10.3b (see equation (10.5)). Instead of filtering just the derivative it is also possible to use an ideal controller and filter the measured signal. The transfer function of such

a controller with a filter is then

$$C(s) = k_p \left(1 + \frac{1}{sT_i} + sT_d\right) \frac{1}{1 + sT_f + (sT_f)^2/2}. \tag{10.13}$$

where a second order filter is used.

Setpoint Weighting

Figure 10.1 on page 304 shows two configurations of a PID controller. The system in Figure 10.1a shows a controller with *error feedback* where proportional, integral and derivative action acts on the error. In the simulation of PID controllers in Figure 10.2 there is a large initial peak of the control signal, which is caused by the derivative of the reference signal. The peak can be avoided by using the controller in Figure 10.1b where proportional and derivative action acts only on the process output. An intermediate form is given by

$$u = k_p\big(\beta r - y\big) + k_i \int_0^\infty \big(r(\tau) - y(\tau)\big)\, d\tau + k_d\Big(\gamma \frac{dr}{dt} - \frac{dy}{dt}\Big), \tag{10.14}$$

where the proportional and derivative actions act on fractions β and γ of the reference. Integral action has to act on the error to make sure that the error goes to zero in steady state. The closed loop systems obtained for different values of β and γ respond to load disturbances and measurement noise in the same way. The response to reference signals is different because it depends on the values of β and γ, which are called *reference weights* or *setpoint weights*. We illustrate the effect of setpoint weighting by an example.

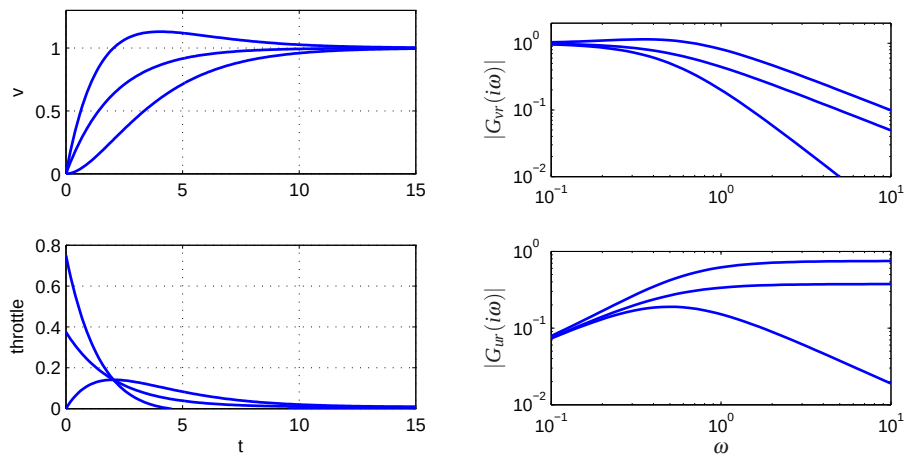
Example 10.7 Cruise control
Consider the PI controller for the cruise control system derived in Example 10.3. Figure 10.12 shows the effect of setpoint weighting on the response of the system to a reference signal. With $\beta = 1$ (error feedback) there is an overshoot in velocity and the control signal (throttle) is initially close to the saturation limit. There is no overshoot with $\beta = 0$ and the control signal is much smaller, clearly a much better drive comfort. The frequency responses gives another view of the same effect. The parameter β is typically in the range of 0 to 1 and γ is normally zero to avoid large transients in the control signal when the reference is changed. ∇

The controller given by equation (10.14) is a special case of controller with two degrees of freedom, which will be discussed in more detail in Section 11.2.

Implementation Based on Operational Amplifiers

PID controllers have been implemented in different technologies. Figure 10.13 shows how PI and PID controllers can be implemented by feedback around operational amplifiers.

To show that the circuit in Figure 10.13b is a PID controller we will use the the approximate relation between the input voltage e and the output voltage u of an



(a) Step response **(b)** Frequency responses

Figure 10.12: Time and frequency responses for PI cruise control with setpoint weighting. Step responses are shown in (a) and the gain curves of the frequency responses in (b). The controller gains are $k_p = 0.74$ and $k_i = 0.19$. The setpoint weights are $\beta = 0$, 0.5 and 1 and $\gamma = 0$.

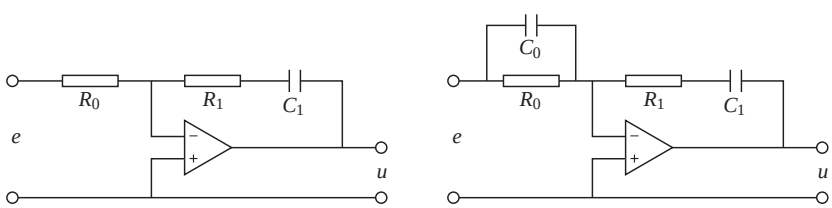
operational amplifier derived in Example 8.3,

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

In this equation 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. The impedances are given by

$$Z_0(s) = \frac{R_0}{1 + R_0 C_0 s} \qquad Z_1(s) = R_1 + \frac{1}{C_1 s},$$

and we find the following relation between the input voltage e and the output volt-



(a) PI controller **(b)** PID controller

Figure 10.13: Schematic diagrams for PI and PID controllers using op amps. The circuit on the left uses a capacitor in the feedback path to store the integral of the error. The circuit on the right ads a filter on the input to provide derivative action.

age u:

$$u = -\frac{Z_1}{Z_0}e = -\frac{R_1}{R_0}\frac{(1+R_0C_0s)(1+R_1C_1s)}{R_1C_1s}e.$$

This is the input-output relation for a PID controller of the form (10.1) with parameters

$$k_p = \frac{R_1}{R_0} \qquad T_i = R_1C_1 \qquad T_d = R_0C_0.$$

The corresponding results for a PI controller is obtained by setting $C_0 = 0$ (removing the capacitor).

Computer Implementation

In this section we briefly describe how a PID controller may be implemented using a computer. The computer typically operates periodically, with signals from the sensors sampled and converted to digital form by the A/D converter, the control signal computed and then converted to analog form for the actuators. The sequence of operation is as follows:

1. Wait for clock interrupt
2. Read input from sensor
3. Compute control signal
4. Send output to the actuator
5. Update controller variables
6. Repeat

Notice that an output is sent to the actuators as soon as it is available. The time delay is minimized by making the calculations in Step 3 as short as possible and performing all updates after the output is commanded. This simple way of reducing the latency is, unfortunately, seldom used in commercial systems.

As an illustration we consider the PID controller in Figure 10.11, which has a filtered derivative, setpoint weighting and protection against integral windup. The controller is a continuous time dynamical system. To implement it using a computer, the continuous time system has to be approximated by a discrete time system.

A block diagram of a PID controller with anti-windup is shown in Figure 10.11. The signal v is the sum of the proportional, integral and derivative terms, and the controller output is $u = \mathrm{sat}(v)$ where sat is the saturation function that models the actuator. The proportional term $k_p(\beta r - y)$ is implemented simply by replacing the continuous variables with their sampled versions. Hence

$$P(t_k) = k_p\left(\beta r(t_k) - y(t_k)\right), \tag{10.15}$$

where $\{t_k\}$ denotes the sampling instants, i.e., the times when the computer reads its input. We let h represent the sampling time, so that $t_{k+1} = t_k + h$. The integral term is obtained by approximating the integral with a sum

$$I(t_{k+1}) = I(t_k) + k_i h\, e(t_k) + \frac{h}{T_t}\big(\mathrm{sat}(v) - v\big), \tag{10.16}$$

where $T_t = h/k_t$ represents the anti-windup term. The filtered derivative term D is given by the differential equation

$$T_f \frac{dD}{dt} + D = -k_d y.$$

Approximating the derivative with a backward difference gives

$$T_f \frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -k_d \frac{y(t_k) - y(t_{k-1})}{h},$$

which can be rewritten as

$$D(t_k) = \frac{T_f}{T_f + h} D(t_{k-1}) - \frac{k_d}{T_f + h} \left(y(t_k) - y(t_{k-1}) \right). \tag{10.17}$$

The advantage of using a backward difference is that the parameter $T_f/(T_f + h)$ is non-negative and less than one for all $h > 0$, which guarantees that the difference equation is stable. Reorganizing equations (10.15)–(10.17), the PID controller can be described by the following pseudo code:

```
% Precompute controller coefficients
bi=ki*h
ad=Tf/(Tf+h)
bd=kd/(Tf+h)
br=h/Tt

% Control algorithm - main loop
while (running) {
  r=adin(ch1)               % read setpoint from ch1
  y=adin(ch2)               % read process variable from ch2
  P=kp*(b*r-y)              % compute proportional part
  D=ad*D-bd*(y-yold)        % update derivative part
  v=P+I+D                   % compute temporary output
  u=sat(v,ulow,uhigh)       % simulate actuator saturation
  daout(ch1)                % set analog output ch1
  I=I+bi*(r-y)+br*(u-v)     % update integral
  yold=y                    % update old process output
  sleep(h)                  % wait until next update interval
}
```

Precomputation of the coefficients `bi`, `ad`, `bd` and `br` saves computer time in the main loop. These calculations have to be done only when controller parameters are changed. The main loop is executed once every sampling period. The program has three states: `yold`, `I`, and `D`. One state variable can be eliminated at the cost of less readable code. The latency between reading the analog input and setting the analog output consists of 4 multiplications, 4 additions and evaluation of the `sat` function. All computations can be done using fixed point calculations if necessary. Notice that the code computes the filtered derivative of the process output, and that it has set-point weighting and anti-windup protection.

10.6 FURTHER READING

The history of PID control is very rich and stretches back to the beginning of the foundation of control theory. A very readable treatment is given by Bennett [28, 29] and Mindel [148]. The Ziegler-Nichols rules for tuning PID controllers, first presented in 1942 [204], were developed based on extensive experiments with pneumatic simulators and Vannevar Bush's differential analyzer at MIT. An interview with Nichols gives an interesting view of the development of the Ziegler-Nichols rules is given in an interview with Ziegler [40]. An industrial perspective on PID control is given in [34], [176] and [200] and in the paper [58] cited in the beginning of this chapter. A comprehensive presentation of PID control is given in [17] and [18]. Interactive learning tools for PID control can be downloaded from `http://www.calerga.com/contrib`.

EXERCISES

10.1 Consider the systems represented by the block diagrams in Figure **??**. Assume that the process has the transfer function $P(s) = b/(s+a)$ show that the transfer functions from r to y are

$$(a)G_{yr}(s) = \frac{bk_d s^2 + bk_p s + bk_i}{(1+bk_d)s^2 + (a+bk_d)s + bk_i}$$
$$(b)G_{yr}(s) = \frac{bk_i}{(1+bk_d)s^2 + (a+bk_d)s + bk_i}$$

10.2 Consider a second order process with transfer function

$$P(s) = \frac{b}{s^2 + a_1 s + a_2}.$$

The closed loop system with a PI controller is a third order system. Show that it is possible to position the closed loop poles as long as the sum of the poles is $-a_1$. Give equations for the parameters that give the closed loop characteristic polynomial

$$(s+\alpha_0)(s^2 + 2\zeta_0\omega_0 s + \omega_0^2).$$

10.3 Consider a second order process with transfer function

$$P(s) = \frac{b}{s^2 + a_1 s + a_2}.$$

Find the gains for a PID controller that gives the closed loop system the characteristic polynomial

$$(s+\alpha_0)(s^2 + 2\zeta_0\omega_0 s + \omega_0^2).$$

10.4 Consider a system with the transfer function $P(s) = (s+1)^{-2}$. Find an integrating controller that gives a closed loop pole at $s = -a$ and determine the value of a that maximizes integral gain. Determine the other poles of the system

and judge if the pole can be considered to be dominant. Compare with the value of the integral gain given by (10.6).

10.5 Compartment models and many systems encountered in industry have the property that their impulse responses are positive or equivalently that their step responses are monotone. Consider such a system with the transfer function $P(s)$. Show that the impulse response $h_n(t)$ of the normalized system $P(s)/P(0)$ has the properties $h_n(t) \geq 0$ and $\int_0^\infty h_n(t)dt = 1$. The function $h_n(t)$ can be interpreted as a probability density function - the probability that a particle entering the system at time 0 will exit at time t. Let

$$T_{ar} = \int_0^\infty t h_n(t) dt$$

be the average residence time. Show that $T_{ar} = -P'(0)/P(0)$ and that the tuning formula (10.6) can be written as $k_i = 1/(T_{ar}P(0))$.

10.6 Consider a system with the transfer function $P(s) = e^{-s}/s$. Determine parameters of P,PI and PID controllers using Ziegler-Nichols step- and frequency response methods. Compare the parameter values obtained by the different rules and discuss the results.

10.7 (Vehicle steering) Design a proportion-integral controller for the vehicle steering system that gives closed loop characteristic equation

$$s^3 + 2\omega_0 s^2 + 2\omega_0 s + \omega_0^3.$$

10.8 (Congestion control) A simplified flow model for TCP transmission is derived in [134, 101]. The linearized dynamics are modeled by the transfer function

$$G_{qp}(s) = \frac{b}{(s+a_1)(s+a_2)} e^{-s\tau^*}.$$

which describes the dynamics relating expected queue length q to expected packet drop p. The parameters are given by where $a_1 = 2N^2/(c\tau^{*2})$, $a_2 = 1/\tau^*$ and $b = c^2/(2N)$. The parameter c is the bottleneck capacity, N the number sources feeding the link and τ^* is the round trip delay time. Use the parameter values $N = 75$ sources, $C = 1250$ packets/s and $\tau^* = 0.15$ and find parameters of a PI controller using one of the Ziegler-Nichols rules and the corresponding improved rule. Simulate the responses of the closed loop systems obtained with the PI controllers.