Feedback Systems

An Introduction for Scientists and Engineers
SECOND EDITION

Karl Johan Åström
Richard M. Murray

Version v3.0g (1 Nov 2015)

This is the electronic edition of *Feedback Systems* and is available from http://www.cds.caltech.edu/~murray/FBS. Hardcover editions may be purchased from Princeton University Press, http://press.princeton.edu/titles/8701.html.

PRINCETON UNIVERSITY PRESS
PRINCETON AND OXFORD

Chapter Eleven
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.

L. Desborough and R. Miller, 2002 [DM02].

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 describe 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.

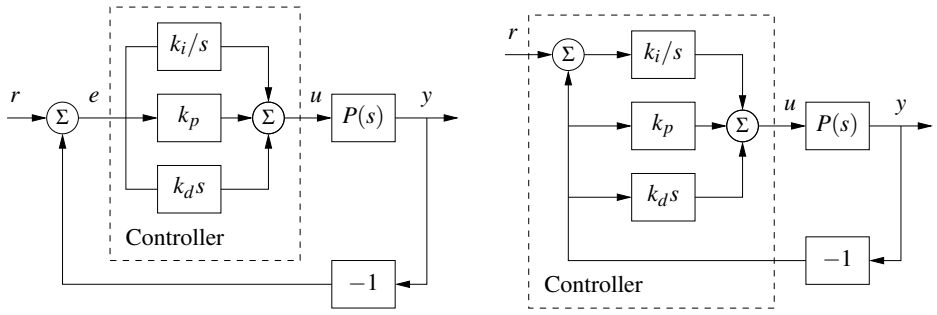
11.1 Basic Control Functions

PID control, which was introduced 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 stand-alone controllers, as part of hierarchical, distributed control systems and 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 [YHSD00].

Block diagrams of closed loop systems with PID controllers are shown in Figure 11.1. The control signal u for the system in Figure 11.1a 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 11.1b; combinations of the schemes will be discussed in Section 11.5. The command signal r is called the reference signal in regulation problems, or the *setpoint* in the literature of PID control. The input/output relation for an ideal PID controller with error feedback is

$$u = k_p e + k_i \int_0^t e(\tau)\,d\tau + k_d \frac{de}{dt} = k_p\Big(e + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de}{dt}\Big). \qquad (11.1)$$

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



(a) PID using error feedback (b) PID using two degrees of freedom

Figure 11.1: Block diagrams of closed loop systems with ideal PID controllers. Both controllers 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 act on the process output y.

k_p, the integral gain k_i and the derivative gain k_d. The controller can also be parameterized with the time constants $T_i = k_p/k_i$ and $T_d = k_d/k_p$, called integral time (constant) and derivative time (constant).

The controller (11.1) represents an idealized controller. It is a useful abstraction for understanding the PID controller, but several modifications must be made to obtain a controller that is practically useful. Before discussing these practical issues we will develop some intuition about PID control.

We start by considering pure proportional feedback. Figure 11.2a 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. 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 transfer function be $P(s)$, with proportional feedback we have $C(s) = k_p$ and the transfer function from reference to error is

$$G_{er}(s) = \frac{}{1 + k_p P(s)}. \tag{11.2}$$

Assuming that the closed loop is stable the steady-state error for a unit step is

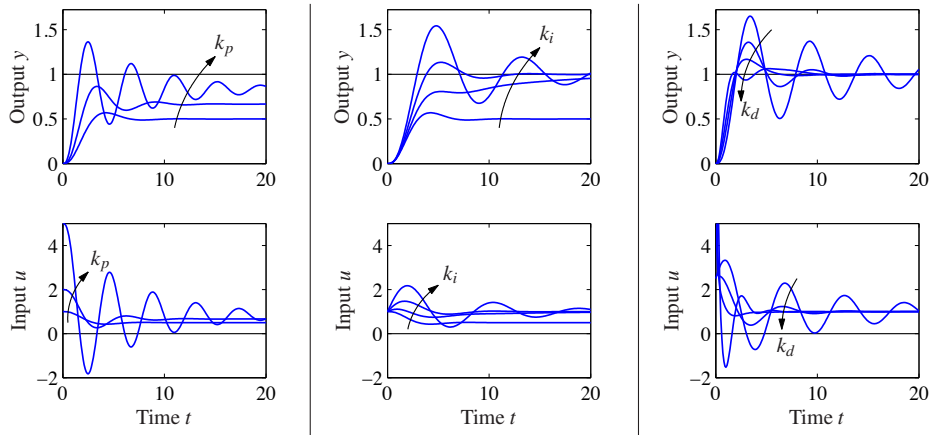
$$G_{er}(0) = \frac{1}{1 + k_p P(0)}.$$

For the system in Figure 11.2a 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. The system becomes unstable for $k_p = 8$. 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

$$u(t) = k_p e(t) + u_{\text{ff}}, \tag{11.3}$$

where u_{ff} is a feedforward term that is adjusted to give the desired steady-state



(a) Proportional control (b) PI control (c) PID control

Figure 11.2: Responses to step changes in the reference value for a 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 has 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 $k_p = 2.5$, $k_i = 1.5$ and $k_d = 0$, 1, 2 and 4.

value. If r is constant and we choose $u_{\text{ff}} = r/P(0) = k_r r$, then the steady-state 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.

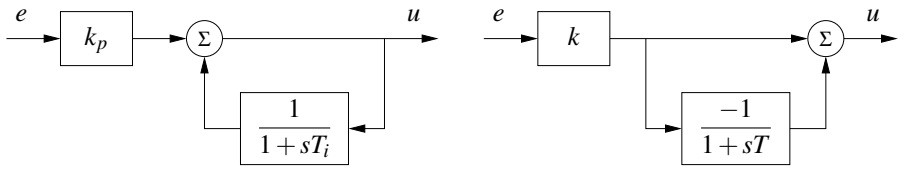
Another alternative to avoid a steady state error is to multiply the reference by $1 + k_p P(0)$, but this also requires precise knowledge of the steady state gain of the process.

As we saw in Section 7.4, 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 (11.1) with $k_i \neq 0$. Assume that $u(t)$ and $e(t)$ converge to steady-state values $u = u_0$ and $e = e_0$. It then follows from equation (11.1) that

$$u_0 = k_p e_0 + k_i \lim_{t\to\infty} \int_0^t e(t)dt$$

The limit of the right hand side is not finite unless $e(t)$ goes to zero which implies that $e_0 = 0$. We can thus conclude that with integral action the error will be zero if it reaches a steady state. Notice that we have not assumed that the process is linear or time invariant. We have, however, assumed that an equilibrium reached. Analysis based on transfer functions require that the closed loop system is linear and time invariant.

Using integral action to achieve zero steady-state error is much better than using feedforward, which requires a precise knowledge of process parameters.



(a) Automatic reset (b) Derivative action

Figure 11.3: The block diagram in (a) shows how integral action is implemented using *positive feedback* with a first-order system, sometimes called automatic reset. The block diagram in (b) shows how derivative action can be implemented by taking differences between a static system and a first-order system.

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

$$C(s) = k_p + \frac{k_i}{s} + k_d s. \tag{11.4}$$

The controller has infinite gain at zero frequency ($C(0) = \infty$), and it then follows from equation (**??**) 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 (11.3) automatically. This is shown in Figure 11.3a, 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 11.3a 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 11.2b 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 toward the reference for small values of k_i and goes faster for larger integral gains, but the system also becomes more oscillatory.

The integral gain k_i is a useful measure for attenuation of load disturbances. Consider a closed loop system under PID control, like the one in Figure 9.1. Assume that the system is stable and initially at rest with all signals being zero. Apply a unit step load 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. Since $e(t)$ goes to zero as $t \to \infty$ it follows from (11.1) that

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

The integrated error $\int_0^\infty e(t)dt$ is thus inversely proportional to the 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 or anticipatory 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 11.3b. The transfer function for the system is

$$G_{ue}(s) = k\Big(1 - \frac{1}{1+sT}\Big) = k\frac{sT}{1+sT} = \frac{kTs}{1+sT}. \tag{11.5}$$

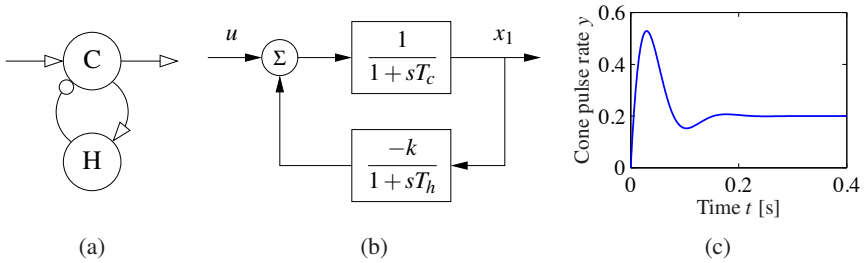
The transfer function $G_{ue}(s)$, approximates a derivative for low frequencies, because for $|s| \ll 1/T$ we have $G(s) \approx kTs$. The transfer function G_{ue} acts like a differentiator for signals with frequencies low frequencies and as a constant gain k for high frequency signals.

Figure 11.2c illustrates the effect of derivative action: the system is oscillatory when no derivative action is used, and it becomes more damped as the derivative gain is increased. Performance deteriorates if the 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 11.2c. The impulse can be avoided by using the controller configuration shown in Figure 11.1b.

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 9.14, where it is said that the eye adapts to changing light intensity. Analogously, feedback with integral action is called perfect adaptation [YHSD00]. In biological systems proportional, integral and derivative action is generated by combining subsystems with dynamical behavior similarly to what is done in engineering systems. For example, PI action can be generated by the interaction of several hormones [ESGK02].

Example 11.1 PD action in the retina

The response of cone photoreceptors 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, which 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 11.4a. The system



(a) (b) (c)

Figure 11.4: Schematic diagram of cone photoreceptors (C) and horizontal cells (H) in the retina. In the schematic diagram in (a), excitatory feedback is indicated by arrows and inhibitory feedback by circles. A block diagram is shown in (b) and the step response in (c).

can be modeled by ordinary differential equations by representing neuron signals as continuous variables representing the average pulse rate. In [Wil99] 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 Figure 11.4b. The step response of the system shown in Figure 11.4c 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 $T_h = 0.08$. ∇

11.2 Simple Controllers for Complex Systems

Many of the design methods discussed in previous chapters have the property that the complexity of the controller is a direct reflection of the complexity of the model. When designing controllers by output feedback in Chapter 8, 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 these design methods to PID control will require that the models have to be of first or second order; see Exercise **??**.

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 the 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 system, the level in a tank and the concentration in a volume with good mixing. System dynamics are of second order if the storage of mass, energy and momentum can be captured by two state variables; typical examples are the position of a car on the road (position and velocity), the stabilization of stiff satellites (angular orientation and angular velocity), the levels in two connected tanks and two-compartment models. A wide range of techniques

for model reduction are also available. In this chapter we will focus on design techniques where 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 integral 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 simply $s + Kk_i$. Specifying performance by the desired time constant T_{cl} of the closed loop system, we find that the integral gain is given by

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

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

For systems that are not well represented by a constant gain, we can obtain a better approximation by using the 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 13.5. The controller gain is then given by

$$k_i = -\frac{1}{2P'(0)}, \tag{11.6}$$

and the expected closed loop time constant is $T_{cl} \approx -P'(0)/P(0)$.

Example 11.2 Integral 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 10.2. The transfer function for the system dynamics is

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

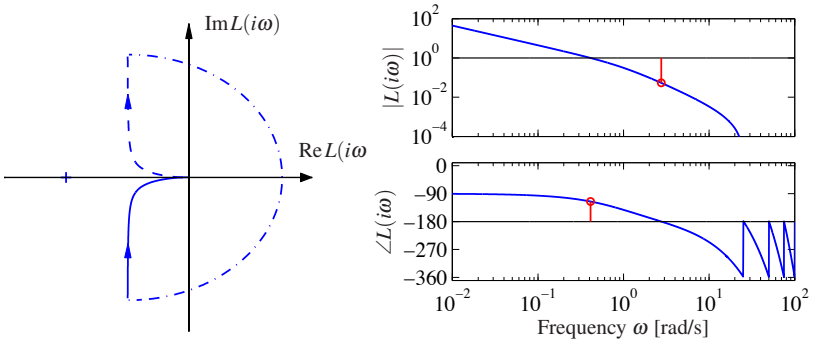
where $a = \zeta\omega_0$, $\tau = 2\pi n/\omega_0$ and the gain has been normalized to 1. We have $P(0) = 1$ and $P'(0) = -\tau/2 - 1/a$, and it follows from (11.6) that the integral gain can be chosen as $k_i = a/(2 + a\tau)$. Nyquist and Bode plots for the resulting loop transfer function are shown in Figure 11.5. ∇

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

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



(a) Nyquist plot (b) Bode plot

Figure 11.5: 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.

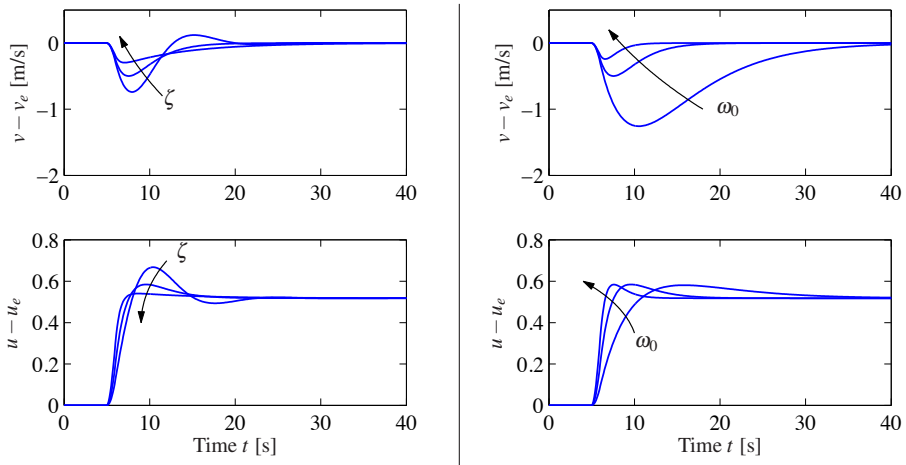
The closed loop poles can thus be assigned arbitrary values by proper choice of the controller gains k_p and k_i. We illustrate the by an example.

Example 11.3 Cruise control using PI feedback
Consider the problem of maintaining the speed of a car as it goes up a hill. In Example 6.14 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 6.11:

$$\frac{d(v-v_e)}{dt} = -a(v-v_e) + b(u-u_e) - g\theta, \tag{11.7}$$

where v is the velocity of the car, u is the input from the engine throttle 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 a first-order system. Since the open loop dynamics is so slow ($1/a \approx 100$ sec.) , it is natural to specify a faster closed loop system by requiring that the closed loop system be of second-order with damping ratio ζ and undamped natural frequency ω_0. The controller gains are given by equation (**??**).

Figure 11.6 shows the velocity and the throttle for a car that initially moves on a horizontal road and encounters a hill with a slope of 4° at time $t = 6$ s. To design a PI controller we choose $\zeta = 1$ to obtain a response without overshoot, as shown in Figure 11.6a. 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 illustrated in Figure 11.6b. The largest velocity error decreases with increasing ω_0, but the control signal also changes more rapidly. In the simple model (11.7) 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



(a) $\omega_0 = 0.5$, $\zeta = 0.5, 1, 2$ (b) $\zeta = 1$, $\omega_0 = 0.2, 0.5, 1$

Figure 11.6: Cruise control using PI feedback. The step responses for the error and input illustrate the effect of parameters $\zeta = 1$ and ω_0 on the response of a car with cruise control. The slope of the road changes linearly from 0° to 4° between $t = 5$ and 6 s. (a) Responses for $\omega_0 = 0.5$ and $\zeta = 0.5$, 1 and 2. Choosing $\zeta = 1$ gives no overshoot in v. (b) Responses for $\zeta = 1$ and $\omega_0 = 0.2$, 0.5 and 1.0.

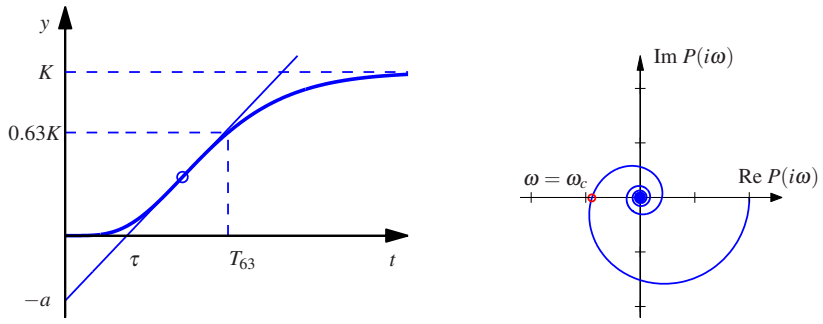
force, which also restricts the admissible value of ω_0. A reasonable choice of ω_0 is in the range 0.5–1.0. Notice in Figure 11.6 that even with $\omega_0 = 0.2$ the largest velocity error is only 1 m/s. ∇

A PI controller can also be used for a process with second-order dynamics, but there will be restrictions on the possible locations of the closed loop poles. 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 11.2.

Instead of finding a low-order model and designing controllers for them, we can also use a high-order model and attempt to place only a few dominant poles. An integral 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 integral controller is $L(s) = k_i P(s)/s$. The roots of the closed loop characteristic polynomial are the roots of $s + k_i P(s) = 0$. Requiring that $s = -a$ be a root, the controller gain should be chosen as

$$k_i = \frac{a}{P(-a)}. \tag{11.8}$$

The pole $s = -a$ will be a dominant closed-loop pole if a is small. A similar approach can be applied to PI and PID controllers; see Exercise 11.3.



(a) Step response method (b) Frequency response method

Figure 11.7: Ziegler–Nichols step and frequency response experiments. The open-loop unit step response in (a) is characterized by the parameters a and τ. The frequency response method (b) characterizes process dynamics by the point where the Nyquist curve of the process transfer function first intersects the negative real axis and the frequency ω_c where this occurs.

11.3 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 [ZN42]. Their idea was to perform a simple experiment, extract some features of process dynamics from the experiment and determine the controller parameters from the features.

Ziegler–Nichols' Tuning

In the 1940s, Ziegler and Nichols developed two methods for controller tuning 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 11.7a. The step response is measured by a *bump test*. The process is first brought to steady-state, the input is then changed by a suitable amount, the output is measured and scaled to correspond to a unit step input. Newton and Ziegler characterized the step response by only two parameters a and τ, which are the intercepts of the steepest tangent of the step response with the coordinate axes. The parameter 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 be able to determine the parameters, it suffices to wait until the response has had an inflection point. The suggested controller parameters are given in Table 11.1. They were obtained by extensive simulation of a range of representative processes. A controller was

Table 11.1: 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	k_p	T_i	T_d
P	1/a		
PI	0.9/a	3τ	
PID	1.2/a	2τ	0.5τ

(a) Step response method

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

(b) Frequency response method

tuned manually for each process, and an attempt was then made 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 at 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 11.7b.

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 unfortunately have 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 the FOTD (First Order and Time Delay) model

$$P(s) = \frac{K}{1+sT} e^{-\tau s}. \tag{11.9}$$

This model is commonly used to approximate the step response of systems with monotone step responses. The parameters of the model are be obtained from a bump test as indicated in Figure 11.7a. The zero frequency gain K is the steady-state value of the unit step response. The time delay τ is the intercept of the steepest tangent with the time axis, as in the Ziegler-Nichols method. The time T_{63} is the time where the output has reached 63% of its steady-state value and T is then given by $T = T_{63} - \tau$. Notice that it takes longer time to find an FOTD model than the Ziegler-Nichols model (a and τ) because to determine K it is necessary to wait until the steady state has been reached, see Figure 11.7a

The frequency response method can be improved by measuring more points on the Nyquist curve, e.g., the zero frequency gain K or the point where the process has a 90° phase lag. This latter point can be obtained by connecting an integral

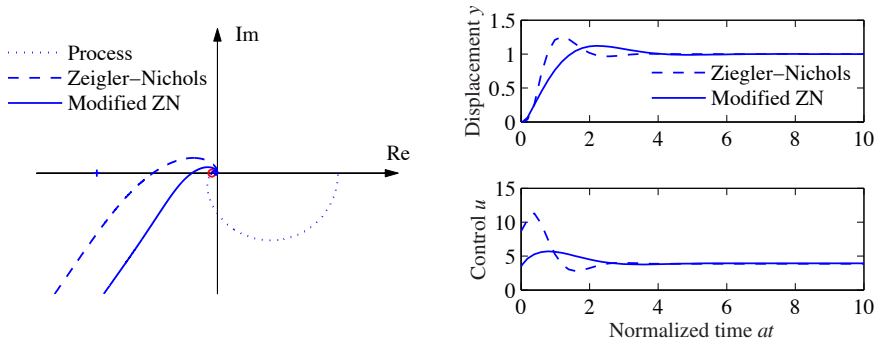


Figure 11.8: PI control of an AFM in tapping mode. Nyquist plots (a) and step responses (b) for PI control of the vertical motion of an atomic force microscope in tapping mode. Results with Ziegler–Nichols tuning are shown by dashed lines, and modified Ziegler–Nichols tuning is shown by solid lines. The Nyquist plot of the process transfer function is shown by dotted lines.

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, based on [ÅH05]:

$$k_p = \frac{0.15\tau + 0.35T}{K\tau} \quad \left(\frac{0.9T}{K\tau}\right), \qquad k_i = \frac{0.46\tau + 0.02T}{K\tau^2} \quad \left(\frac{0.3T}{K\tau^2}\right), \tag{11.10a}$$

$$k_p = 0.22k_c - \frac{0.07}{K} \quad \left(0.4k_c\right), \qquad k_i = \frac{0.16k_c}{T_c} + \frac{0.62}{KT_c} \quad \left(\frac{0.5k_c}{T_c}\right). \tag{11.10b}$$

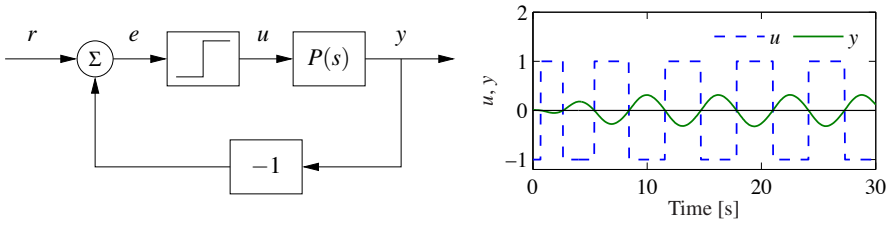
The values for the Ziegler–Nichols rule from Table 11.2a are given in parentheses. Notice that the improved formulas typically give lower controller gains than the Ziegler–Nichols method. The integral gain is higher for systems where the dynamics are delay-dominated, $\tau \gg T$.

Example 11.4 Atomic force microscope 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 11.2. The transfer function is normalized by choosing $1/a$ as the time unit, yielding

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

where $T_n = 2n\pi a/\omega_0 = 2n\pi\zeta$. The Nyquist plot of the transfer function is shown (dotted line) in Figure 11.8a for $\zeta = 0.002$ and $n = 20$. The first intersection with the real axis occurs at $\mathrm{Re}\, s = -0.0461$ for $\omega_c = 13.1$. The critical gain is thus $k_c = 21.7$ and the critical period is $T_c = 0.48$. Using the 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.



(a) Relay feedback (b) Oscillatory response

Figure 11.9: Block diagram of a process with relay feedback (a) and typical signals (b). The process output y is a solid line, and the relay output u is a dashed line. Notice that the signals u and y have opposite phases.

With this controller the stability margin is $s_m = 0.31$, which is quite small. The step response of the controller is shown (dashed lines) in Figure 11.8. Notice in particular that there is a large overshoot in the control signal.

The modified Ziegler–Nichols rule (11.10b) gives the controller parameters $k_p = 3.47$ 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 11.8. 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 11.2 that a pure integral controller has the normalized gain $k_i = 1/(2 + T_n) = 0.44$ which is more than an order of magnitude smaller than the integral gain of the PI controller.. ∇

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 critical 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 11.9a. For many systems there will then be an oscillation, as shown in Figure 11.9b, where the relay output u is a square wave and the process output y is close to a sinusoid. Moreover, the fundamental sinusoidal components of the input and the output are 180° out of phase, which means that the system oscillates with the critical period T_c. Notice that an oscillation with constant period is established quickly.

To determine the critical gain k_c we expand the square wave relay output in a Fourier series. Notice 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 $|P(i\omega_c)| = \pi a/(4d)$ and the critical gain is

$$k_c = \frac{4d}{a\pi}. \tag{11.11}$$

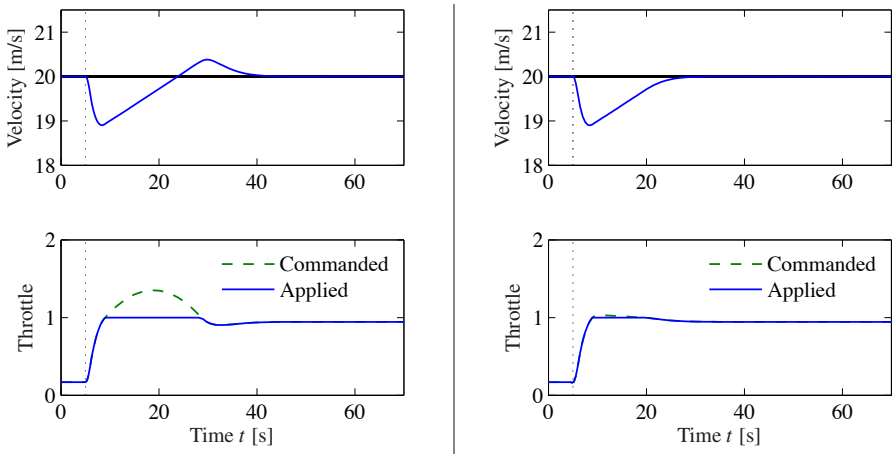
Having obtained the critical gain k_c and the critical period T_c, the controller parameters can then be determined using the Ziegler–Nichols rules. Improved tuning can be obtained by fitting a model to the data obtained from the relay experiment.

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 replaced by a PID controller as soon as the tuning is finished.

11.4 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 remains 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 11.5 Cruise control
The windup effect is illustrated in Figure 11.10a, 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 required to compensate for 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 the velocity settles to the desired value at time $t = 40$. Also notice the large overshoot. ∇



(a) Windup (b) Anti-windup

Figure 11.10: Simulation of PI cruise control with windup (a) and anti-windup (b). 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 a dashed line. The controller parameters are $k_p = 0.5$ and $k_i = 0.1$. The anti-windup compensator eliminates the overshoot by preventing the error for building up in the integral term of the controller.

There are many methods to avoid windup. One method is illustrated in Figure 11.11: 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 toward 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 noise can then cause an undesirable reset. A reasonable choice is to choose k_t as a multiple of $1/T_i = k_i/k_p$.

The anti-windup scheme in Figure 11.11 assumes that the actuator output is available to the controller, when this is not the case we will instead use a model of the actuator as will be done in Section 11.5.

We illustrate how integral windup can be avoided by investigating the cruise control system.

Example 11.6 Cruise control with anti-windup

Figure 11.10b shows what happens when a controller with anti-windup is applied to the system simulated in Figure 11.10a. Because of the feedback from the ac-

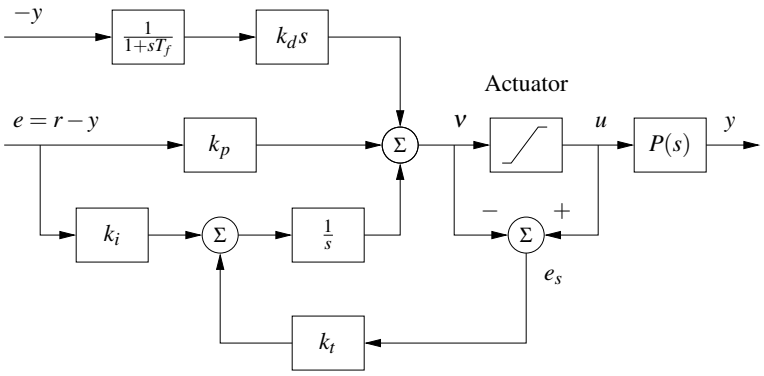


Figure 11.11: PID controller with a filtered derivative and anti-windup. The input to the integrator $(1/s)$ consists of the error term plus a "reset" based on input saturation. If the actuator is not saturated, then $e_s = u - v$, otherwise e_s will decrease the integrator input to prevent windup.

tuator 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 11.10a and the large overshoot is avoided. The tracking gain used in the simulation is $k_t = 2$ which is an order of magnitude larger than $1/T_i = k_i/k_p = 0.2$. ∇

Manual Control

Automatic control is often combined with manual control. This can be accomplished by providing the controller with a switch from manual to automatic. Manual control is typically actuated using buttons for increasing and decreasing the control signal. The control signal increases at constant rate when pushing the increase button and it decreases at constant rate when the decrease button is pushed. There are more sophisticated schemes where the rate increases when the switch is pushed a longer time, like the search mechanism in an iPod.

Care has to be taken to avoid transients when switching modes. This can be accomplished by the arrangement shown in Figure 11.12 where the integrator for the PI controller is used both for manual and automatic control. In automatic mode (A) the switch S connects the error signal to the input of the integrator. In manual mode (M) the switch disconnects the error signal and connects the integral to the increase-decrease buttons in manual mode. The switching transients will be small since the output of the integrator is the same in manual and automatic mode. There proportional term will create a switching transient if the error e is not zero when the mode is changed.

Controller with a Tracking Mode

When continuous control is combined with logic it is important that the controller state is handled properly when switching modes. In the controller with manual

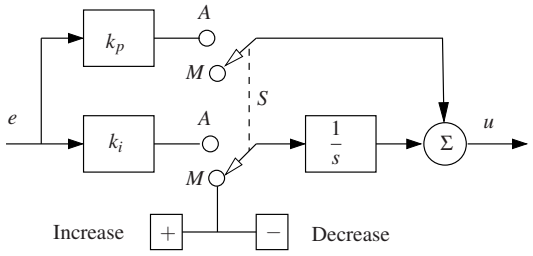


Figure 11.12: Block diagram of a PI controller that can switch between manual and automatic control. Automatic mode is indicated by *A* and manual mode by *M*. Manual control is executed by the buttons marked + and − which increase or decrease the control signal.

and automatic control in Figure 11.12 the problem was solved by using the same integrator for manual and automatic control. This ensures that there will be no switching transients. A general approach is to introduce a *tracking mode* in the controller.

A block diagram of a controller with a tracking mode is shown in Figure 11.13. The controller has modes for automatic control and tracking, selected by a switch. In the automatic control mode (A) the controller works as a normal controller which generates the control signal u from the reference r and the process output y. The controller has two degrees of freedom and is governed by the differential equation

$$a(p)u(t) = b(p)r(t) - c(p)y(t). \tag{11.12}$$

In tracking mode (T) the output u of the controller is equal to the tracking signal w and the behavior is governed by the differential equation

$$a_o(p)\,z(t) = b(p)\,r(t) - c(p)\,y(t) + \big(a_o(p) - a(p)\big)\,u(t), \tag{11.13}$$

where the polynomial $a_o(s)$ is stable, monic, and of the same degree as $a(s)$. If the controller has integral action we have

$$a(0) = 0, \qquad b(0) = c(0), \qquad a_o(0) \neq 0.$$

In steady state where all signals are constant we have $w = u$. The controller can

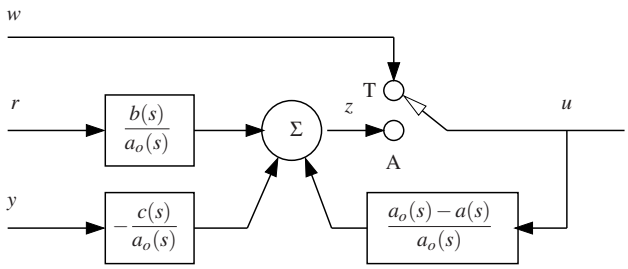


Figure 11.13: Block diagram of a controller with a tracking mode. The controller has three continuous input signals: reference r, process output y and tracking signal w, and a switch for selecting automatic A control or tracking T.

then be switched to automatic mode without transients. The polynomial $a_o(s)$ determines how quickly the state is reached. By continuity we have $w(t) \approx u(t)$ for slowly varying signals.

When the controller is switched to automatic we set $z = u$ and equation (11.13) becomes

$$a_o(p)\,u(t) = b(p)\,r(t) - c(p)\,y(t) + \big(a_o(p) - a(p)\big)\,u(t),$$

which is identical to (11.12). We will illustrate with an example.

Example 11.7 PI controller with tracking mode
A PI controller with two degrees of freedom has the input/output relation

$$u(t) = k_p\big(\beta r(t) - y(t)\big) + k_i \int_0^t \big(r(\tau) - y(\tau)\big)\,d\tau,$$

which is equivalent to the differential equation

$$\frac{du}{dt} = \left(k_p\beta\frac{dr}{dt} + k_i r\right) - \left(k_p\frac{dy}{dt} + k_i y\right).$$

In tracking mode the controller is described by

$$\frac{dz}{dt} + a_o z = \left(k_p\beta\frac{dr}{dt} + r\right) - \left(k_p\frac{dy}{dt} + k_i y\right) + a_o u$$

(see Figure 11.13). A convenient way to combine automatic and tracking mode is to represent the controller as

$$\frac{dz}{dt} = \left(k_p\beta\frac{dr}{dt} + r\right) - \left(k_p\frac{dy}{dt} + k_i y\right) + a_o(u - z)$$
$$u = \begin{cases} z & \text{automatic mode} \\ w & \text{tracking mode} \end{cases}.$$

In this representation the controller is a first order system with one state z whose dynamics has the characteristic polynomial $s + a_o$. In automatic mode the controller output is equal to the state z. In the tracking mode the output is equal to the external tracking signal w and the controller state z tracks the signal w. ∇

11.5 Implementation

There are many practical issues that have to be considered when implementing PID controllers. They have been developed over time based on practical experience. 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 in 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. The time constant of the filter is typically chosen as $T_f = (k_d/k_p)/N = T_d/N$, with N in the range 5–20. Filtering is obtained automatically if the derivative is implemented by taking the difference between the signal and its filtered version as shown in Figure 11.3b; see also equation (11.5)).

Instead of filtering just the derivative, it is also possible to use an ideal controller and filter the measured signal. Choosing a second order Butterworth filter the transfer function the controller with the filter becomes

$$C(s) = k_p\left(1+\frac{1}{sT_i}+sT_d\right)\frac{1}{1+sT_f+(sT_f)^2/2}. \tag{11.14}$$

Setpoint Weighting

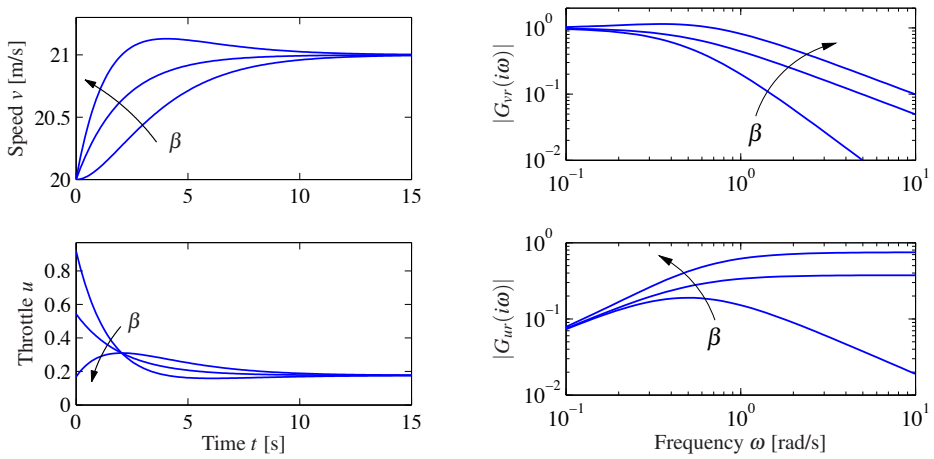
Figure 11.1 shows two configurations of a PID controller. The system in Figure 11.1a has a controller with *error feedback* where proportional, integral and derivative action acts on the error. In the simulation of PID controllers in Figure 11.2c there is a large initial peak in the control signal, which is caused by the derivative of the reference signal. The peak can be avoided by using the controller in Figure 11.1b, where proportional and derivative action acts only on the process output. An intermediate form is given by

$$u = k_p\big(\beta r - y\big) + k_i\int_0^t \big(r(\tau)-y(\tau)\big)\,d\tau + k_d\Big(\gamma\frac{dr}{dt}-\frac{dy}{dt}\Big), \tag{11.15}$$

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 11.8 Cruise control with setpoint weighting
Consider the PI controller for the cruise control system derived in Example 11.3. Figure 11.14 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 0–1, and γ is normally zero to avoid large transients in the control signal when the reference is changed. ∇

The controller given by equation (11.15) is a special case of the general controller structure having two degrees of freedom, which was discussed in Section 8.5.



(a) Step response (b) Frequency responses

Figure 11.14: 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 $\gamma = 0$.

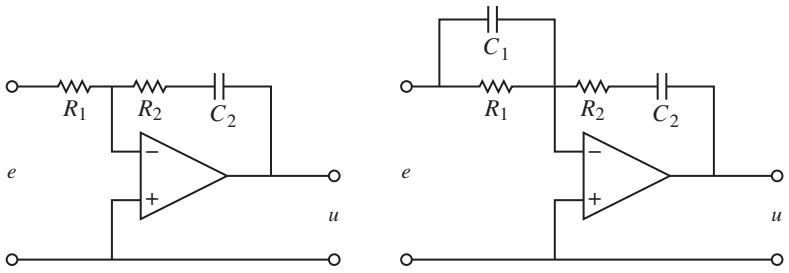
Implementation Based on Operational Amplifiers

PID controllers have been implemented in different technologies. Figure 11.15 shows how PI and PID controllers can be implemented by feedback around operational amplifiers.

To show that the circuit in Figure 11.15b is a PID controller we will use the approximate relation between the input voltage e and the output voltage u of the operational amplifier derived in Example 9.3,

$$u = -\frac{Z_2}{Z_1}e.$$

In this equation Z_1 is the impedance between the negative input of the amplifier



(a) PI controller (b) PID controller

Figure 11.15: Schematic diagrams for PI and PID controllers using op amps. The circuit in (a) uses a capacitor in the feedback path to store the integral of the error. The circuit in (b) adds a filter on the input to provide derivative action.

and the input voltage e, and Z_2 is the impedance between the zero input of the amplifier and the output voltage u. The impedances are given by

$$Z_1(s) = \frac{R_1}{1+R_1C_1s}, \qquad Z_2(s) = R_2 + \frac{1}{C_2s},$$

and we find the following relation between the input voltage e and the output voltage u:

$$u = -\frac{Z_2}{Z_1}e = -\frac{R_2}{R_1}\frac{(1+R_1C_1s)(1+R_2C_2s)}{R_2C_2s}e.$$

This is the input/output relation for a PID controller of the form (11.1) with parameters

$$k_p = \frac{R_1C_1+R_2C_2}{R_1C_2}, \qquad T_i = R_1C_1+R_2C_2, \qquad T_d = \frac{R_1R_2C_1C_2}{R_1C_1+R_2C_2}.$$

The corresponding results for a PI controller are obtained by setting $C_1 = 0$ (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, and 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 state
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 11.11, which has a filtered derivative, setpoint weighting and protection against integral windup (anti-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.

In Figure 11.11, 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 $P = k_p(\beta r - y)$ is implemented simply by replacing the continuous variables with their sampled versions. Hence

$$P(t_k) = k_p\left(\beta r(t_k) - y(t_k)\right), \tag{11.16}$$

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,

$$I(t_{k+1}) = I(t_k) + k_i h\, e(t_k) + \frac{h}{T_t}\big(\mathrm{sat}(v) - v\big), \tag{11.17}$$

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

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

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

$$D(t_k) = \frac{T_f}{T_f + h} D(t_{k-1}) - \frac{k_d}{T_f + h}\left(y(t_k) - y(t_{k-1})\right). \tag{11.18}$$

The advantage of using a backward difference is that the parameter $T_f/(T_f + h)$ is nonnegative and less than 1 for all $h > 0$, which guarantees that the difference equation is stable. Reorganizing equations (11.16)–(11.18), the PID controller can be described by the following pseudocode:

```
% 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)        % compute 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 state
  yold=y                    % update derivative state
  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 four multiplications, four 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 setpoint weighting and anti-windup protection.

11.6 Further Reading

The history of PID control is very rich and stretches back to the beginning of the foundation of control theory. Very readable treatments are given by Bennett [Ben79, Ben93] and Mindel [Min02]. The Ziegler–Nichols rules for tuning PID controllers, first presented in 1942 [ZN42], were developed based on extensive experiments with pneumatic simulators and Vannevar Bush's differential analyzer at MIT. An interesting view of the development of the Ziegler–Nichols rules is given in an interview with Ziegler [Bli90]. An industrial perspective on PID control is given in [Bia95], [Shi96] and [YH91] and in the paper [DM02] cited in the beginning of this chapter. A comprehensive presentation of PID control is given in [ÅH05]. Interactive learning tools for PID control can be downloaded from http://www.calerga.com/contrib.

Exercises

11.1 (Ideal PID controllers) Consider the systems represented by the block diagrams in Figure 11.1. Assume that the process has the transfer function $P(s) = b/(s+a)$ and show that the transfer functions from r to y are

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

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

Pick some parameters and compare the step responses of the systems.

11.2 Consider a second-order process with the 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

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

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

and judge if the pole can be considered dominant. Compare with the value of the integral gain given by equation (11.6).

11.4 (Ziegler–Nichols tuning) Consider a system with transfer function $P(s) = e^{-s}/s$. Determine the 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.

11.5 (Vehicle steering) Design a proportional-integral controller for the vehicle steering system that gives the closed loop characteristic polynomial

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

11.6 (Congestion control) A simplified flow model for TCP transmission is derived in [HMTG00, LPD02]. The linearized dynamics are modeled by the transfer function

$$G_{qp}(s) = \frac{b}{(s+a_1)(s+a_2)} e^{-s\tau_e},$$

which describes the dynamics relating the expected queue length q to the expected packet drop p. The parameters are given by $a_1 = 2N^2/(c\tau_e^2)$, $a_2 = 1/\tau_e$ and $b = c^2/(2N)$. The parameter c is the bottleneck capacity, N is the number of sources feeding the link and τ_e is the round-trip delay time. Use the parameter values $N = 75$ sources, $C = 1250$ packets/s and $\tau_e = 0.15$ and find the 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.

11.7 (Motor drive) Consider the model of the motor drive in Exercise 3.10. Develop an approximate second-order model of the system and use it to design an ideal PD controller that gives a closed loop system with eigenvalues in $\zeta\omega_0 \pm i\omega_0\sqrt{1-\zeta^2}$. Add low-pass filtering as shown in equation (11.14) and explore how large ω_0 can be made while maintaining a good stability margin. Simulate the closed loop system with the chosen controller and compare the results with the controller based on state feedback in Exercise 7.12.

11.8 Consider the system in Exercise 11.7 investigate what happens if the second-order filtering of the derivative is replace by a first-order filter.

11.9 (Tuning rules) Apply the Ziegler–Nichols and the modified tuning rules to design PI controllers for systems with the transfer functions

$$P_1 = \frac{e^{-s}}{s}, \qquad P_2 = \frac{e^{-s}}{s+1}, \qquad P_3 = e^{-s}.$$

Compute the stability margins and explore any patterns.

11.10 (Windup and anti-windup) Consider a PI controller of the form $C(s) = 1 + 1/s$ for a process with input that saturates when $|u| > 1$, and whose linear dynamics are given by the transfer function $P(s) = 1/s$. Simulate the response of

the system to step changes in the reference signal of magnitude 1, 2 and 3. Repeat the simulation when the windup protection scheme in Figure 11.11 is used.

11.11 (Windup protection by conditional integration) Many methods have been proposed to avoid integrator windup. One method called *conditional integration* is to update the integral only when the error is sufficiently small. To illustrate this method we consider a system with PI control described by

$$\frac{dx_1}{dt} = u, \qquad u = \text{sat}_{u_0}(k_p e + k_i x_2), \qquad \frac{dx_2}{dt} = \begin{cases} e & \text{if } |e| < e_0 \\ 0 & \text{if } |e| \geq e_0, \end{cases}$$

where $e = r - x$. Plot the phase portrait of the system for the parameter values $k_p = 1$, $k_i = 1$, $u_0 = 1$ and $e_0 = 1$ and discuss the properties of the system. The example illustrates the difficulties of introducing ad hoc nonlinearities without careful analysis.