Feedback Systems

An Introduction for Scientists and Engineers

SECOND EDITION

Karl Johan Åström
Richard M. Murray

Version v3.1.5 (2020-07-24)

Chapter 11

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 a PID feedback control algorithm.

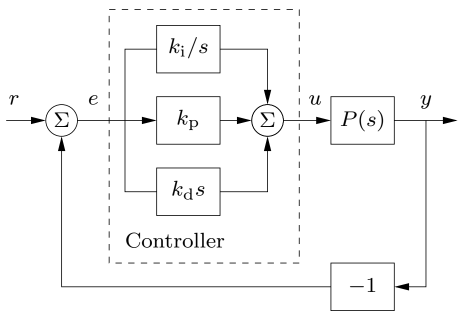
L. Desborough and R. Miller, 2002 [DM02a].

Proportional-integral-derivative (PID) control is by far the most common way of using feedback in engineering systems. In this chapter we present the basic properties of PID control and the methods for choosing the parameters of the controllers. We also analyze the effects of actuator saturation, an important feature of many feedback systems, and describe methods for compensating for it. Finally, we discuss the implementation of PID controllers as an example of how to implement feedback control systems using analog or digital computation.

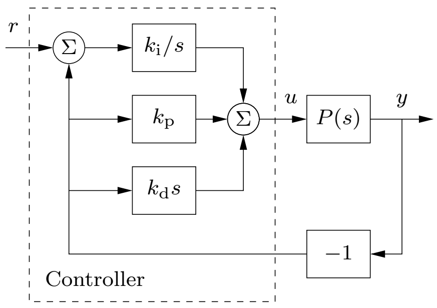
11.1 Basic Control Functions

The PID controller was introduced in Section 1.6, where Figure 1.15 illustrates that control action is composed of three terms: the proportional term (P), which depends on the present error; the integral term (I), which depends on past errors; and the derivative term (D), which depends on anticipated future errors. A major difference between a PID controller and an advanced controller based on feedback from estimated states (see Section 8.5) is that the observer-based controller predicts the future state of the system using a mathematical model, while the PID controller makes use of linear extrapolation of the measured output. A PI controller does not make use of any prediction of the future state of the system.

A survey of controllers for more than 100 boiler-turbine units in the Guangdong Province in China is a typical illustration of the prevalence of PID-based control: 94.4% of all controllers were PI, 3.7% PID, and 1.9% used advanced control [SLL16]. The reasons why derivative action is used in only 4% of all controllers are that the benefits of prediction are significant primarily for processes that permit large controller gains. For many systems, prediction by linear extrapolation can generate large undesired control signals because measurement noise is amplified. In addition care must be taken to find a proper prediction horizon. Temperature control is a



(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 only the process output y.

typical case where derivative action can be beneficial: sensors have low noise levels and controllers can have high gain.

PID control appears in simple dedicated systems and in large factories with thousands of controllers: as stand-alone controllers, as elements of hierarchical, distributed control systems, and as components of embedded systems. Advanced control systems are implemented as hierarchical systems, where high-level controllers give setpoints to PID controllers in a lower layer. The PID controllers are directly connected to the sensors and actuators of the process. The importance of PID controllers thus has not decreased with the adoption of advanced control methods, because the performance of the system depends critically on the behavior of the PID controllers [DM02a]. 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 command signal r is called the reference signal in regulation problems or the *setpoint* in the literature of PID control. 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 $k_f r$ 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 input/output relation for an ideal PID controller with error feedback is

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

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 k_p, the integral gain k_i, and the derivative gain k_d. The gain k_p is sometimes expressed in terms of the

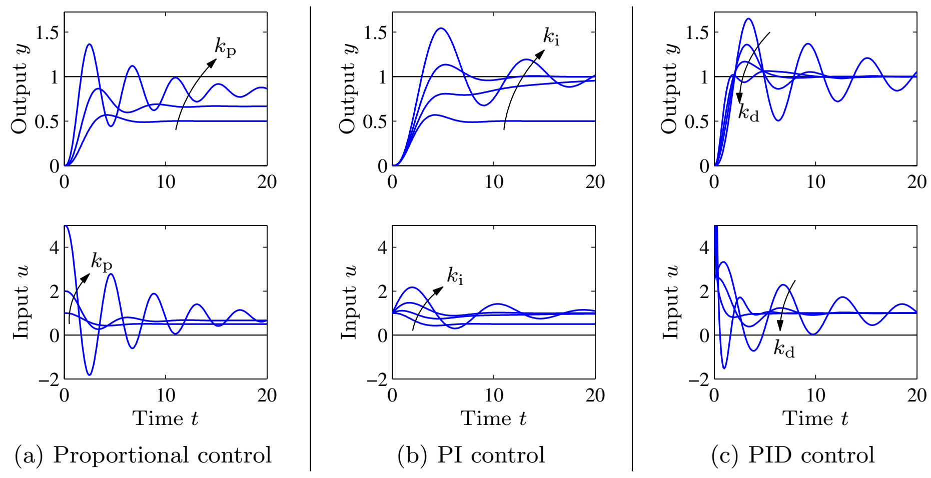


Figure 11.2: Responses to step changes in the reference value for a system with a proportional controller (a), PI controller (b), and PID controller (c). The process has the transfer function $P(s) = 1/(s + 1)^3$, the proportional controller has parameters $k_p = 1$, 2, and 5, the PI controller has parameters $k_p = 1$, $k_i = 0$, 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.

proportional band, defined as $\text{PB} = 100/k_p$. A proportional band of 10% thus implies that the controller operates linearly for only 10% of the span of the measured signal. The controller can also be parameterized with the time constants $T_i = k_p/k_i$ and $T_d = k_d/k_p$, called the integral time (constant) and the derivative time (constant). The parameters T_i and T_d have dimensions of time and can naturally be related to the time constants of the controller.

The controller (11.1) is an idealized representation. 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

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

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

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

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 $k_\mathrm{p} = 8$. 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

$$u(t) = k_\mathrm{p} e(t) + u_\mathrm{ff}, \tag{11.3}$$

where u_ff is a feedforward term that is adjusted to give the desired steady-state value. If the reference value r is constant and we choose $u_\mathrm{ff} = r/P(0) = k_\mathrm{f} r$, 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 zero frequency gain $P(0)$, which is usually not available. The parameter u_ff, called the *reset* value, was adjusted manually in early controllers. Another alternative to avoid a steady-state error is to multiply the reference by $1 + k_\mathrm{p} P(0)$, but this also requires precise knowledge of $P(0)$.

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 $k_\mathrm{i} \neq 0$. 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

$$u_0 = k_\mathrm{p} e_0 + k_\mathrm{i} \lim_{t\to\infty} \int_0^t e(t)dt.$$

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 integral control has the property that if a steady state exists, the error will always be zero. This property is sometimes called the *magic of integral action*. Notice that we have not assumed that the process is linear or time-invariant. We have, however, assumed that there is an equilibrium point. It is much better to achieve zero steady-state error by integral action than by feedforward, which requires a precise knowledge of process parameters.

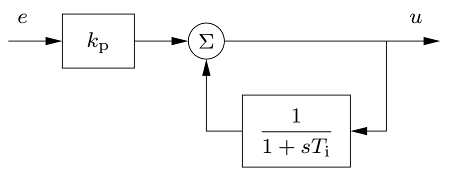
The effect of integral action can also be understood from frequency domain analysis. The transfer function of the PID controller is

$$C(s) = k_\mathrm{p} + \frac{k_\mathrm{i}}{s} + k_\mathrm{d} s. \tag{11.4}$$

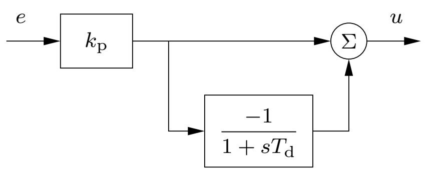
The controller has infinite gain at zero frequency ($C(0) = \infty$), and it then follows from equation (11.2) that $G_{er}(0) = 0$, which implies that there is no steady-state error for a step input.

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 (it was much easier to implement a low-pass filter than to implement an integrator). The transfer function of the system in Figure 11.3a is obtained by block diagram algebra: we have

$$G_{ue} = k_\mathrm{p} \frac{1 + sT_\mathrm{i}}{sT_\mathrm{i}} = k_\mathrm{p} + \frac{k_\mathrm{p}}{sT_\mathrm{i}},$$



(a) Integral action (automatic reset)



(b) Derivative action

Figure 11.3: Implementation of integral and derivative action. 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.

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 converges more quickly 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 11.1. 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 equation (11.1) that

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

The *integrated error*, IE, for a unit step load disturbance $\text{IE} = \int_0^\infty e(t)dt$ is thus inversely proportional to the integral gain k_i and hence serves as 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 derivative action. 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.*

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

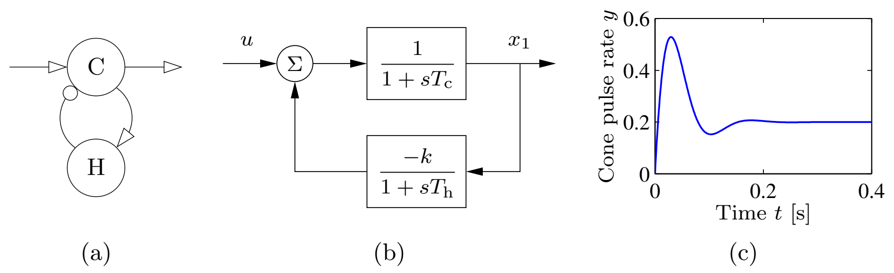


Figure 11.4: Schematic diagram of cone photoreceptors (C) and horizontal cells (H) in the retina. In the schematic diagram in (a), excitatory feedback is indicated by arrows and inhibitory feedback by circles. A block diagram is shown in (b) and the step response in (c).

for the system is

$$G_{ue}(s) = k_p\Big(1 - \frac{1}{1+sT_d}\Big) = k_p\frac{sT_d}{1+sT_d} = \frac{k_d s}{1+sT_d}. \tag{11.5}$$

The transfer function $G_{ue}(s)$ approximates a derivative for low frequencies because for $|s| \ll 1/T_d$ we have $G(s) \approx k_pT_ds = k_ds$. The transfer function G_{ue} acts like a differentiator for signals with low frequencies and as a constant gain k_p for high-frequency signals, so we can regard this as a filtered derivative.

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. 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 9.18, 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 are 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 [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 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 given 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 requires that the models must be of first or second order.

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, the 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 and velocity of a car on the road, the orientation and angular velocity of satellites, the levels in two connected tanks, and the concentrations in two-compartment models. A wide range of techniques for model reduction are also available. In this section 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. Any stable system can be controlled by an integral controller provided that the requirements on the closed loop system are modest. To design a controller we approximate the transfer function of the process by a constant $K = P(0)$, which will be reasonable for any stable system at sufficiently low frequencies. 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 can be chosen as $k_i = 1/(T_{cl}P(0))$.

This simplified analysis requires that T_{cl} be sufficiently large that the process transfer function can indeed be approximated by a constant. A reasonable criterion is that $T_{cl} > T_{ar}$, where $T_{ar} = -P'(0)/P(0)$ is known as the *average residence time* of the open loop system.

To obtain controllers with higher performance we approximate the process dynamics by a first-order system (rather than a constant):

$$P(s) \approx \frac{P(0)}{1 + sT_{ar}}.$$

A reasonable design criterion is to obtain a step response with small overshoot and reasonable response time. An integral controller with gain

$$k_\mathrm{i} = \frac{1}{2P(0)T_\mathrm{ar}}, \tag{11.6}$$

gives the loop transfer function

$$L(s) = P(s)C(s) \approx \frac{P(0)}{1 + sT_\mathrm{ar}}\frac{k_\mathrm{i}}{s} = \frac{1}{2sT_\mathrm{ar}(1 + sT_\mathrm{ar})},$$

and the closed loop poles become $s = (-0.5 \pm 0.5i)/T_\mathrm{ar}$. Using the approximations in Table 7.1 on page 7-20, we see that this controller has $\omega_0 = 1/(T_{ar}\sqrt{2})$, which gives a rise time of $3.1\,T_\mathrm{ar}$, a settling time of $7.9\,T_\mathrm{ar}$, and overshoot of 4%.

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. This transfer function is unusual since there is a time-delay term in the numerator.

To design a controller, we focus on the low-frequency dynamics of the system. We have $P(0) = 1$ and $P'(0) = -\tau/2 - 1/a = -(2 + a\tau)/(2a)$. For low frequencies the loop transfer function can then be approximated by

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

Using the design rule (11.6) we set $k_\mathrm{i} = -1/(2P'(0))$, Nyquist and Bode plots for the resulting loop transfer function are shown in Figure 11.5. We see that the controller provides good performance at low frequency and has good stability margins. Note that even though the system dynamics include a time-delay term, we were able to obtain good performance using a simple integral controller and a simple set of calculations. ∇

Another approach to designing simple controllers is to use the gains of the controller to set the location of the closed loop poles. PI controllers give two gains with which to tune the closed loop dynamics, and for simple models the closed loop poles can be set using these gains.

Consider a first-order system with the transfer function

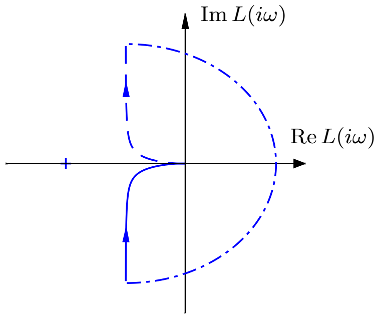
$$P(s) = \frac{b}{s + a}.$$

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

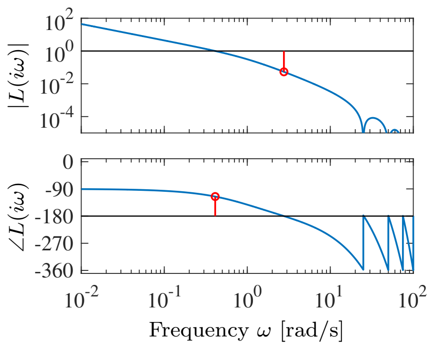
$$s(s + a) + bk_\mathrm{p}s + bk_\mathrm{i} = s^2 + (a + bk_\mathrm{p})s + bk_\mathrm{i}.$$

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

$$p(s) = s^2 + a_1 s + a_2,$$



(a) Nyquist plot



(b) Bode plot

Figure 11.5: Nyquist and Bode plots of the loop transfer function for integral control of an AFM in tapping mode. The integrating controller gives good robustness properties based on a simple analysis. At high frequencies the Nyquist plot has an infinite number of small loops with decreasing amplitude in the left half-plane. These loops are not visible in the Nyquist plot but they show up clearly in the Bode plot.

we find that the controller parameters are

$$k_\mathrm{p} = \frac{a_1 - a}{b}, \qquad k_\mathrm{i} = \frac{a_2}{b}. \tag{11.7}$$

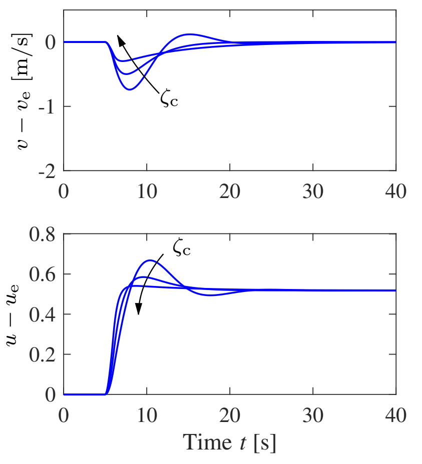
If we require a response of the closed loop system that is slower than that of the open loop system, a reasonable choice is $a_1 = a + \alpha$ and $a_2 = \alpha a$, where $\alpha < a$ determines the closed loop response. If a response faster than that of the open loop system is required, a possible choice is $a_1 = 2\zeta_\mathrm{c}\omega_\mathrm{c}$ and $a_2 = \omega_\mathrm{c}^2$, where ω_c and ζ_c are the undamped natural frequency and damping ratio of the dominant mode.

The choice of ω_c has a significant impact on the robustness of the system and will be discussed in Section 14.5. An upper limit to ω_c is given by highest frequency where the model is valid. Large values of ω_c 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.

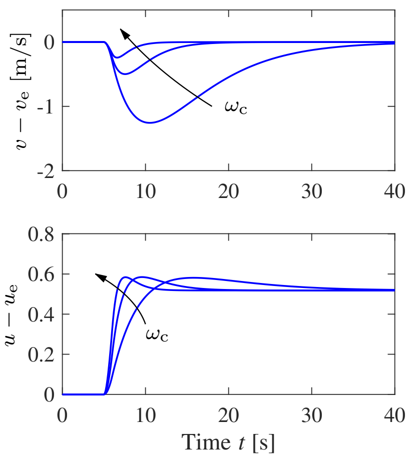
Example 11.3 Cruise control using PI feedback
Consider the problem of maintaining the speed of a car as it goes up a hill. In Example 6.11 we found that there was little difference between the linear and nonlinear models when investigating PI control, provided that the throttle did not reach the saturation limits. A simple linear model of a car was given in Example 6.11:

$$\frac{d(v - v_\mathrm{e})}{dt} = -a(v - v_\mathrm{e}) - b_\mathrm{g}(\theta - \theta_\mathrm{e}) + b(u - u_\mathrm{e}), \tag{11.8}$$

where v is the velocity of the car, u is the input to the engine (throttle), and θ is the slope of the hill. The parameters were $a = 0.01$, $b = 1.32$, $b_\mathrm{g} = 9.8$, $v_\mathrm{e} = 20$, $\theta_\mathrm{e} = 0$, and $u_\mathrm{e} = 0.1687$. This model will be used to find suitable parameters of a vehicle speed controller. The transfer function from throttle to velocity is a



(a) $\omega_c = 0.5$, $\zeta_c = 0.5$, 1, 2



(b) $\zeta_c = 1$, $\omega_c = 0.2$, 0.5, 1

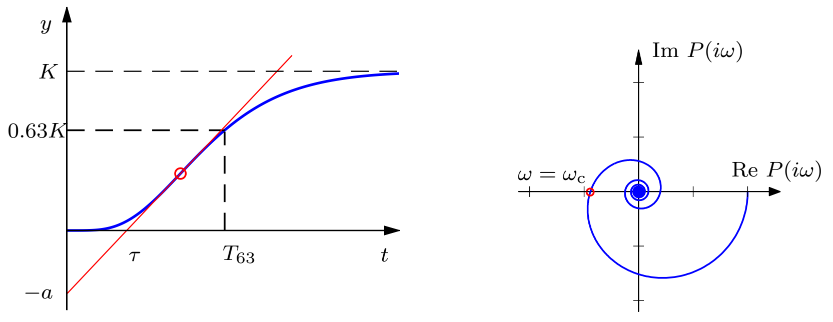
Figure 11.6: Cruise control using PI feedback. The step responses for the error and input illustrate the effect of parameters ζ_c and ω_c on the response of a car with cruise control. The slope of the road changes linearly from 0° to 4° between $t = 5$ and 6 s. (a) Responses for $\omega_c = 0.5$ and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

first-order system. Since the open loop dynamics are quite slow ($1/a \approx 100$ s), 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_c and undamped natural frequency ω_c. The controller gains are given by equation (11.7).

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 $t = 5$ s. To design a PI controller we choose $\zeta_c = 1$ to obtain a response without overshoot, as shown in Figure 11.6a. The choice of ω_c 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_c, but the control signal also changes more rapidly. In the simple model (11.8) 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 limits to the rate of change of the force, which also restricts the admissible value of ω_c. A reasonable choice of ω_c is in the range 0.5–1.0. Notice in Figure 11.6 that even with $\omega_c = 0.2$ the largest velocity error is only about 1.3 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 (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



(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.

integral controller has one parameter, and it is possible to position one pole. To see this, 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 $k_i = a/P(-a)$. The pole $s = -a$ will be a dominant closed loop pole if a is smaller than the magnitude of the other closed loop process poles. A similar approach can be applied to PI and PID controllers (Exercise 11.3).

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. A typical process may have thousands of PID controllers. Since the PID controller has so few parameters a number of special empirical methods have been developed for direct adjustment of the controller parameters.

Ziegler–Nichols' Tuning

The first tuning rules were developed by Ziegler and Nichols [ZN42] in the 1940s. Their idea was to perform a simple experiment on the process and extract features 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, and finally the output is measured and scaled to correspond to a unit step input. Ziegler and Nichols characterized the step

Table 11.1: Original Ziegler–Nichols tuning rules. (a) The step response method gives 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	$0.9/a$	$\tau/0.3$	
PID	$1.2/a$	$\tau/0.5$	0.5τ

(a) Step response method

Type	k_p	T_i	T_d
P	$0.5k_c$		
PI	$0.45k_c$	$T_c/1.2$	
PID	$0.6k_c$	$T_c/2$	$T_c/8$

(b) Frequency response method

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 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 k_c of the proportional gain is observed together with the period of oscillation T_c. It follows from Nyquist's stability criterion that the Nyquist contour for the loop transfer function $L = k_cP(s)$ passes through 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 $P(s)$ where the phase lag is 180°, as shown in Figure 11.7b. The suggested controller parameters are then given by Table 11.1b.

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.

Tuning Based on the FOTD Model

The Ziegler–Nichols methods use only two parameters to characterize process dynamics, a and τ for the step response method and k_c and T_c for the frequency domain method. Tuning of PID controllers can be improved if we characterize the process by more parameters. The first-order and time-delay (FOTD) model

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

is commonly used to approximate the step response of systems with essentially monotone step responses. The parameter τ_n, which has values between 0 and 1, is called the *relative time delay* or the *normalized time delay.* The dynamics are characterized as being *lag dominated* if τ_n is close to zero, *delay dominated* if τ_n is close to one, and *balanced* for intermediate values.

The parameters of the FOTD model can be determined 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 a 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.

There are many versions of improved tuning rules for the model (11.9). As an illustration we give the following rules for PI control, based on [ÅH06]:

$$k_\mathrm{p} = \frac{0.15\tau + 0.35T}{K\tau} \quad \Big(\frac{0.9T}{K\tau}\Big), \qquad k_\mathrm{i} = \frac{0.46\tau + 0.02T}{K\tau^2} \quad \Big(\frac{0.27T}{K\tau^2}\Big), \tag{11.10a}$$

$$k_\mathrm{p} = 0.16k_\mathrm{c} \quad \Big(0.45k_\mathrm{c}\Big), \qquad k_\mathrm{i} = \frac{0.16k_\mathrm{c} + 0.72/K}{T_\mathrm{c}} \quad \Big(\frac{0.54k_\mathrm{c}}{T_\mathrm{c}}\Big). \tag{11.10b}$$

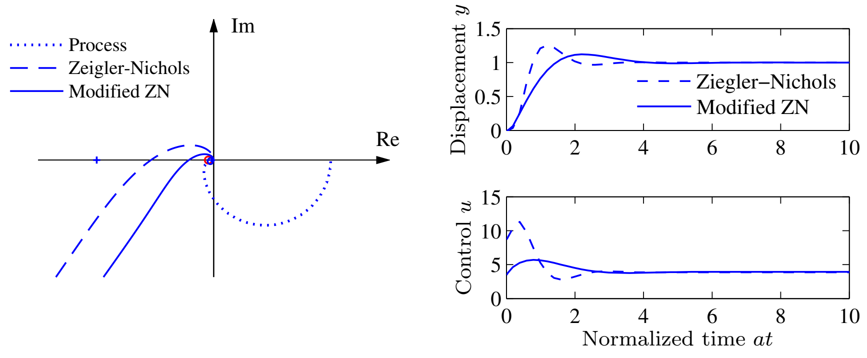
The values for the Ziegler–Nichols rule from Table 11.1 are given in parentheses. Notice that the improved formulas typically give lower controller gains than the original Ziegler–Nichols method.

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 $P(s)$ is shown as a dotted line in Figure 11.8a for $\zeta = 0.002$ and $n = 20$. The first intersection with the real axis occurs at $\operatorname{Re} s = -0.0461$ for $\omega_\mathrm{c} = 13.1$. The critical gain is thus $k_\mathrm{c} = 21.7$ and the critical period is $T_\mathrm{c} = 0.48$. Using the Ziegler–Nichols tuning rule, we find the parameters $k_\mathrm{p} = 8.67$ and $k_\mathrm{i} = 22.6$ ($T_\mathrm{i} = 0.384$) for a PI controller. With this controller the stability margin is $s_\mathrm{m} = 0.31$, which is quite small. The step response of the controller is shown using 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 $k_\mathrm{p} = 3.47$ and $k_\mathrm{i} = 8.73$ ($T_\mathrm{i} = 0.397$) and the stability margin becomes $s_\mathrm{m} = 0.61$. The step response with this controller is shown using solid lines 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 $k_\mathrm{i} = 1/(2 + T_n) = 0.44$, which is



(a) (b)

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.

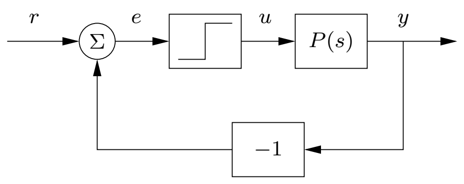
more than an order of magnitude smaller than the integral gain of the PI controller. ∇

The tuning rules based on the FOTD model work well for PI controllers. Derivative action has little effect on processes with delay-dominated dynamics, but can give substantial performance for processes with lag-dominated dynamics. Tuning of PID controllers for processes with lag-dominated dynamics cannot, however, be based on the FOTD model; see [ÅH06].

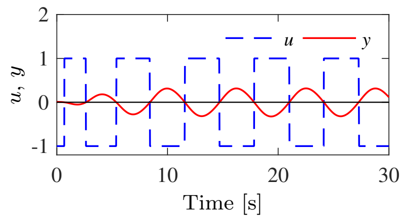
Relay Feedback

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. It is then sufficient to consider only the first harmonic component of the input. Letting d be the relay amplitude, this component 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 $k_c = 4d/(\pi a)$. Having obtained the critical gain k_c and the critical



(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.

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 when the tuning is finished. The main advantage of relay tuning is that a short experiment for identification of process dynamics is generated automatically. The original relay autotuner can be improved significantly by using an asymmetric relay, which admits determination of more parameters [BHÅ16].

11.4 Integral Windup

Many aspects of a control system can be understood from linear models. However, there are 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

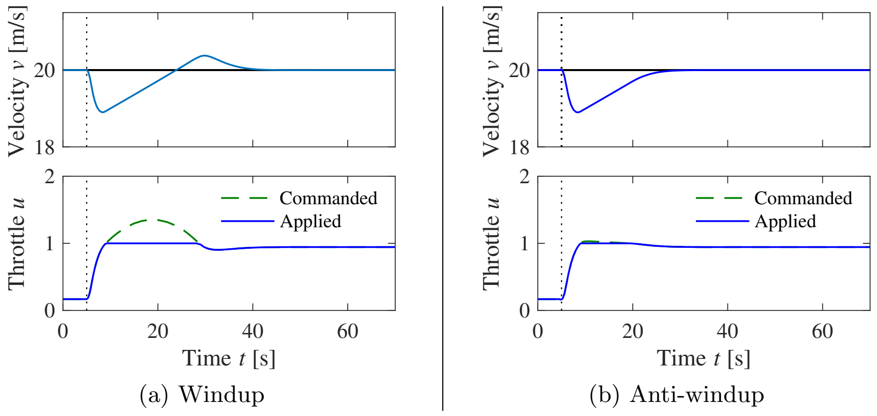


Figure 11.10: Simulation of PI cruise control with windup (a) and anti-windup (b). The figure shows the speed v and the throttle u for a car that encounters a slope that is so steep that the throttle saturates. The controller output is a dashed line. The controller parameters are $k_\text{p} = 0.5$, $k_\text{i} = 0.1$ and $k_\text{aw} = 2.0$. The anti-windup compensator eliminates the overshoot by preventing the error from building up in the integral term of the controller.

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 25, 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. ∇

Avoiding Windup

Windup can occur in any controller with integral action. There are many methods to avoid windup. One method for PID control is illustrated in Figure 11.11: the system has an extra feedback path that is generated from a mathematical model of the saturating actuator. The signal e_s is the difference between the outputs of the controller u_a and the actuator model u. It is fed to the input of the integrator through the gain k_aw. 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_aw; a large value of k_aw gives a short reset time. The parameter k_aw cannot be too large because measurement noise can then cause an undesirable reset. A reasonable choice is to choose k_aw as a multiple of the integral gain k_i.

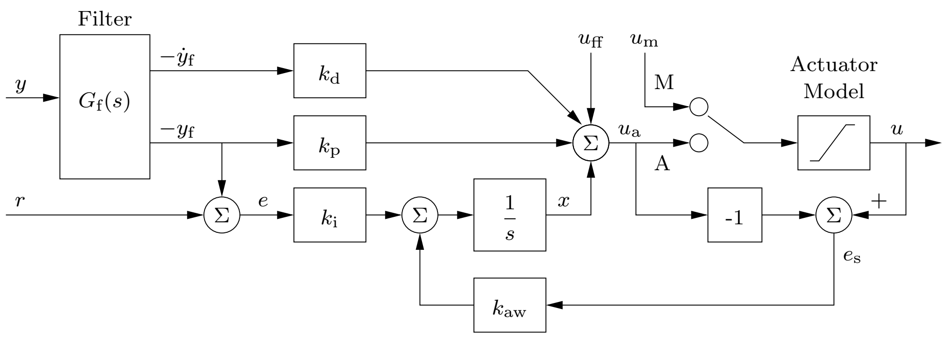


Figure 11.11: PID controller with filtering, anti-windup, and manual control. The controller has filtering of the measured signal, an input u_{ff} for feedforward signal, and another input u_m for direct control of the output. The switch is in position A for normal operation; if it is set to M the control variable is manipulated directly. The input to the integrator ($1/s$) has a "reset" term that avoids integrator windup in addition to the normal P, I, and D terms. Notice that the reference r only enters in the integral term.

The controller also has an input u_{ff} for feedforward control. By entering the feedforward signal as shown in Figure 11.11, the basic anti-windup scheme also deals with saturation caused by the feedforward signal.

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 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 11.10a and the large overshoot is avoided. The tracking gain used in the simulation is $k_{aw} = 2$ which is an order of magnitude larger than the integral gain $k_i = 0.2$. ∇

To explore if windup protection improves stability, we can redraw the block diagram so that the nonlinearity is isolated. The closed loop system then consists of a linear block and a static nonlinearity. With an ideal saturation, the nonlinearity is a sector-bounded nonlinearity modeled by equation (10.17) with $k_{low} = 0$ and $k_{high} = 1$, and the linear part has the transfer function

$$H(s) = \frac{sP(s)C(s) - k_{aw}}{s + k_{aw}} \tag{11.11}$$

(Exercise 11.12). We can use the circle criterion in Section 10.5 to check stability of the closed loop system. We first observe that the special form of the nonlinearity implies that the circle reduces to the line Re $s = -1$. Applying the circle criterion, we find that the system with windup protection is stable if the Nyquist curve of the transfer function $H(s)$ is to the right of the line Re $s = -1$. If we use describing functions we find that oscillations may occur if the Nyquist curve $H(i\omega)$ intersects the negative real axis to the left of the critical point -1.

Manual Control and Tracking

Automatic control is often combined with manual control, where the operation modes are selected by a switch as illustrated in Figure 11.11. The switch is normally in the position A (automatic). Manual control is selected by moving the switch to position M (manual) and the control variable is then manipulated directly, often by buttons for increasing and decreasing the control signal. For example, in a cruise control system such as that shown in Figure 1.16a, the control signal increases at constant rate when pushing the increase speed (accel) button and it decreases at constant rate when the decrease speed (decel) button is pushed. In Figure 11.11 the manipulated variable is denoted by u_m.

Care has to be taken to avoid transients when switching modes. This can be accomplished by the arrangement shown in Figure 11.11. When the controller is in manual mode the feedback through the gain k_aw adjusts the input to the integrator so that the controller output u_a tracks the manual input u_m, resulting in no transient when switching to automatic control.

To see how the controller in Figure 11.11 is implemented, let the integrator output be z. The controller is then described by

$$\frac{dx}{dt} = k_\mathrm{i}(r - y_\mathrm{f}) + k_\mathrm{aw}(u - u_\mathrm{a}), \quad u_\mathrm{a} = z - k_\mathrm{p} y_\mathrm{f} - k_\mathrm{d}\dot y_\mathrm{f}, \quad u = \begin{cases} F(u_\mathrm{a}) & \text{automatic,} \\ F(u_\mathrm{m}) & \text{manual,} \end{cases}$$

where $F(u)$ is the function that represents the actuator model. The parameter k_aw is typically larger than k_i and it then follows that the controller output u tracks u_m in manual mode (tracking would be ideal if the term $k_\mathrm{i}(r - y_\mathrm{f})$ is zero).

Anti-Windup for General Controllers

Anti-windup can also be extended to general control architectures such as the state space-based designs studied in Chapters 7 and 8. For the case of an output feedback controller with integral action via state augmentation (see Example 8.9), we modify the anti-windup compensation to adjust the entire controller state instead of just the integrator state. The approach is particularly easy to understand for controllers based on state feedback and an observer, like the one shown in Figure 8.11. Without modification, when a saturation occurs then the wrong information is sent to the observer (the commanded input instead of the saturated input). To address this, we simply introduce a model for the saturating actuator and feed its output to the observer, as illustrated in Figure 11.12.

To investigate the stability of the controller with anti-windup, we observe that if the observer model is designed so that the process actuator never saturates, the block diagram of the closed loop system can be redrawn so that it consists of a nonlinear static block representing the actuator model $F(x)$ and a linear block representing the observer and the process. We can again make use of the circle criterion described in Section 10.5 to provide conditions for stability. The linear block has the transfer function

$$H(s) = K\bigl(sI - A + LC\bigr)^{-1}\bigl(B + LC[sI - A]^{-1}B\bigr), \tag{11.12}$$

where A, B, and C are the matrices of the state space model, K is the feedback gain matrix, and L is the gain matrix of the Kalman filter. With a simple satu-

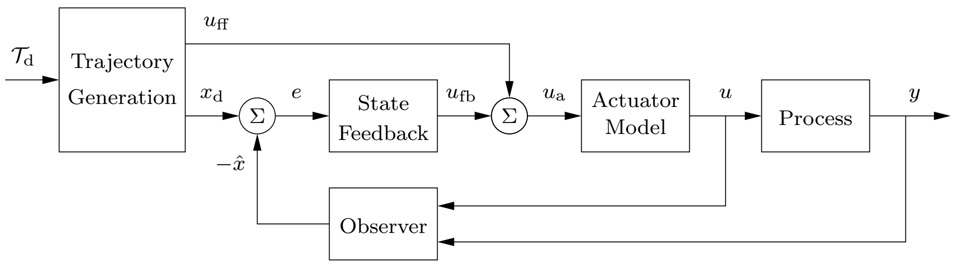


Figure 11.12: Anti-windup for a general controller architecture. Compare with the corresponding controller without anti-windup in Figure 8.11.

rating actuator, the nonlinearity is sector-bounded with $k_{\text{low}} = 0$ and $k_{\text{high}} = 1$ in equation (10.17). It then follows from the circle criterion that the closed loop is stable if the Nyquist plot of $L(i\omega)$ is to the right of the line Re $z = -1/k_{\text{high}} = -1$, and the winding number condition is satisfied.

Facilities for manual control and tracking with observers and state augmentation can be done in the same way as for the PID controller in Figure 11.11.

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). Note that in the implementation in Figure 11.3b, the filter time constant T_f is equal to the derivative time constant T_d ($N = 1$).

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

$$C(s) = k_p \left(1 + \frac{1}{sT_i} + sT_d\right) \frac{1}{1 + sT_f + (sT_f)^2/2}. \tag{11.13}$$

For the system in Figure 11.11, filtering is done in the box marked $G_f(s)$, which has the dynamics

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ -2T_\mathrm{f}^{-2} & -2T_\mathrm{f}^{-1} \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 2T_\mathrm{f}^{-2} \end{pmatrix} y. \qquad (11.14)$$

The states are $x_1 = y_\mathrm{f}$ and $x_2 = dy_\mathrm{f}/dt$. The filter thus gives filtered versions of the measured signal and its derivative. The second-order filter also provides good high-frequency roll-off, which improves robustness.

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

$$u = k_\mathrm{p}\big(\beta r - y\big) + k_\mathrm{i}\int_0^t \big(r(\tau) - y(\tau)\big)\,d\tau + k_\mathrm{d}\Big(\gamma\frac{dr}{dt} - \frac{dy}{dt}\Big), \qquad (11.15)$$

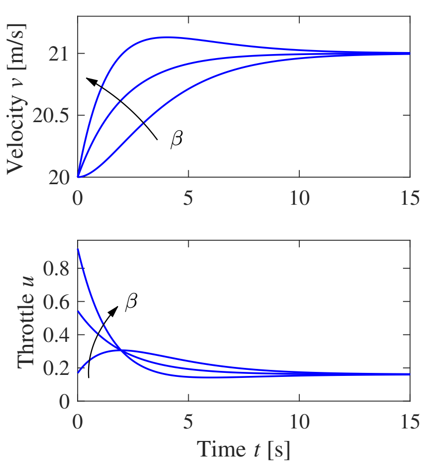
where the proportional and derivative actions act on fractions β and γ of the reference. Integral action has to act on the error to make sure that the error goes to zero in steady state. The closed loop systems obtained for different values of β and γ respond to load disturbances and measurement noise 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*. Setpoint weighting is a simple way to obtain two degree-of-freedom action in a PID controller. A controller with $\beta = \gamma = 0$ is sometimes called an *I-PD controller*, as seen Figure 11.1b. We illustrate the effect of setpoint weighting by an example.

Example 11.7 Cruise control with setpoint weighting
Consider the PI controller for the cruise control system derived in Example 11.3. Figure 11.13 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. ∇

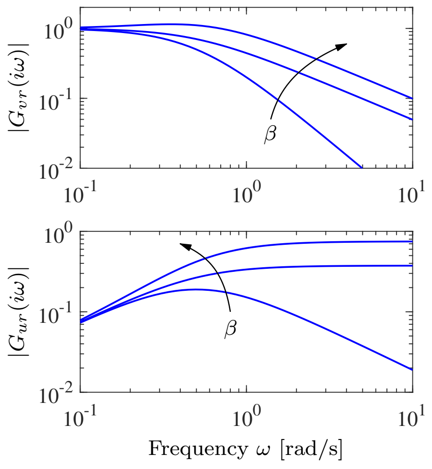
The controller given by equation (11.15) is a special case of the general controller structure having two degrees of freedom, which was discussed in Section 8.5.

Implementation Based on Operational Amplifiers

PID controllers have been implemented in different technologies. Figure 11.14 shows how PI and PID controllers can be implemented by feedback around operational amplifiers.



(a) Step response



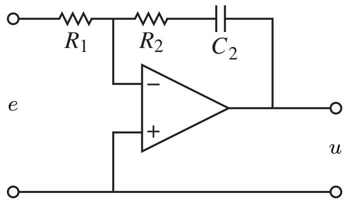
(b) Frequency responses

Figure 11.13: Step 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 $k_\text{p} = 0.74$ and $k_\text{i} = 0.19$. The setpoint weights are $\beta = 0$, 0.5, and 1, and $\gamma = 0$.

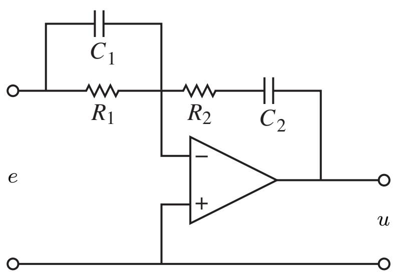
To show that the circuit in Figure 11.14b is a PID controller we will use the approximate relation given by equation (4.14), which is valid when resistances R_i are replaced by impedances Z_i (Exercise 10.1). This gives the transfer function $-Z_2/Z_1$ for the closed loop op amp circuit, noting that the gain of the operational amplifier is negative. For the PI control in Figure 11.14a the impedances are

$$Z_1 = R_1, \quad Z_2 = R_2 + \frac{1}{sC_2} = \frac{1 + R_2C_2s}{sC_2}, \quad \frac{Z_2}{Z_1} = \frac{1 + R_2C_2s}{sR_1C_2} = \frac{R_2}{R_1} + \frac{1}{R_1C_2s},$$

which shows that the circuit is an implementation of a PI controller with gains $k_\text{p} = R_2/R_1$ and $k_\text{i} = 1/(R_1C_2)$.



(a) PI controller



(b) PID controller

Figure 11.14: 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.

A similar calculation for the PID controller in Figure 11.14b gives

$$Z_1(s) = \frac{R_1}{1 + R_1C_1s}, \quad Z_2(s) = R_2 + \frac{1}{C_2s}, \quad \frac{Z_2}{Z_1} = \frac{(1 + R_1C_1s)(1 + R_2C_2s)}{R_1C_2s},$$

which shows that the circuit is an implementation of a PID controller with the parameters

$$k_\mathrm{p} = \frac{R_1C_1 + R_2C_2}{R_1C_2}, \qquad T_\mathrm{i} = R_1C_1 + R_2C_2, \qquad T_\mathrm{d} = \frac{R_1R_2C_1C_2}{R_1C_1 + R_2C_2}.$$

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 output
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 u_a is the sum of the proportional, integral, and derivative terms, and the controller output is $u = \mathrm{sat}(u_\mathrm{a})$, where sat is the saturation function that models the actuator. The proportional term $P = k_\mathrm{p}(\beta r - y)$ is implemented simply by replacing the continuous variables with their sampled versions. Hence

$$P(t_k) = k_\mathrm{p}\big(\beta r(t_k) - y(t_k)\big), \tag{11.16}$$

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,

$$I(t_{k+1}) = I(t_k) + k_\mathrm{i}h\, e(t_k) + \frac{h}{T_\mathrm{aw}}\big(\mathrm{sat}(u_\mathrm{a}) - u_\mathrm{a}\big), \tag{11.17}$$

where $T_\mathrm{aw} = h/k_\mathrm{aw}$ represents the anti-windup term. The filtered derivative term D is given by the differential equation

$$T_\mathrm{f}\frac{dD}{dt} + D = -k_\mathrm{d}\dot{y}.$$

Approximating the derivative with a backward difference gives

$$T_{\mathrm{f}}\frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -k_{\mathrm{d}}\,\frac{y(t_k) - y(t_{k-1})}{h},$$

which can be rewritten as

$$D(t_k) = \frac{T_{\mathrm{f}}}{T_{\mathrm{f}} + h}\,D(t_{k-1}) - \frac{k_{\mathrm{d}}}{T_{\mathrm{f}} + h}\,(y(t_k) - y(t_{k-1}))\,. \tag{11.18}$$

The advantage of using a backward difference is that the parameter $T_{\mathrm{f}}/(T_{\mathrm{f}} + h)$ 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/Taw

 % Initalize variables
 I = 0, yold = adin(ch2)

 % 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
   ua = P + I + D               % compute temporary output
   u = sat(ua, ulow, uhigh)     % simulate actuator saturation
   daout(ch1)                   % set analog output ch1
   I = I + bi*(r-y) + br*(u-ua) % 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 and implemented on a programmable logical controller (PLC). Notice that the code computes the filtered derivative of the process output and that it has setpoint weighting and anti-windup protection. Note also that in this code we apply the actuator saturation inside the controller, rather than measuring the actuator output as in Figure 11.11.

11.6 Further Reading

The history of PID control is very rich and stretches back to the early uses of feedback. Good presentations are given by Bennett [Ben79, Ben93] and Mindel [Min02]. Industrial perspectives on PID control are given in [Bia95], [Shi96], and [YH91], which all mention that a significant fraction of PID controllers are poorly tuned. PID algorithms have been used in many fields; an unconventional application is to explain popular monetary policy rules [HSH15]. 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]. The book [O'D06] lists more than 1730 tuning rules. A detailed discussion of methods for avoiding windup is given in [ZT11], and a comprehensive treatment of PID control is given in Åström and Hägglund [ÅH06]. Advanced relay autotuners are presented in Berner *et al.* [BSÅH17]. 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) $G_{yr}(s) = \dfrac{bk_\text{d}s^2 + bk_\text{p}s + bk_\text{i}}{(1+bk_\text{d})s^2 + (a+bk_\text{p})s + bk_\text{i}}$,

(b) $G_{yr}(s) = \dfrac{bk_\text{i}}{(1+bk_\text{d})s^2 + (a+bk_\text{p})s + bk_\text{i}}$.

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

$$(s+\alpha_\text{c})(s^2 + 2\zeta_\text{c}\omega_\text{c}s + \omega_\text{c}^2).$$

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 (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.

11.5 (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.

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

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

11.7 (Average residence time with PID control) The average residence time is a measure of the response time of the system. For a stable system with impulse response $h(t)$ and transfer function $P(s)$ it can be defined as

$$T_{\text{ar}} = \int_0^\infty t h(t)\, dt = -\frac{P'(0)}{P(0)}.$$

Consider a stable system with $P(0) \neq 0$ and a PID controller having integral gain $k_\text{i} = k_\text{p}/T_\text{i}$. Show that the average residence time of the closed loop system is given by $T_{\text{ar}} = T_\text{i}/(P(0)k_\text{p})$.

11.8 (Web server control) Web servers can be controlled using a method known as *dynamic voltage frequency scaling* in which the processor speed is regulated by changing its supply voltage. A typical control goal is to maintain a given service rate, which is approximately equal to maintaining a specified queue length. The queue length x can be modeled by equation (3.32),

$$\frac{dx}{dt} = \lambda - \mu,$$

where λ is the arrival rate and μ is the service rate, which is manipulated by changing the processor voltage. A PI controller for keeping queue length close to x_r is given by

$$\mu = k_\text{p}(x - \beta x_\text{r}) + k_\text{i} \int_0^t (x - x_\text{r})\, dt.$$

Choose the controller parameters k_p and k_i so that the closed loop system has the characteristic polynomial $s^2 + 1.6s + 1$, then adjust the setpoint weight β so that the response to a step in the reference signal has 2% overshoot.

11.9 (Motor drive) Consider the model of the motor drive in Exercise 3.7 with the parameter values given in Exercise 7.11. 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 $-\zeta\omega_0 \pm i\omega_0\sqrt{1-\zeta^2}$. Add low-pass filtering as shown in equation (11.13) 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 7.11.

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 10. 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

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

where $e = r - x$. Plot the phase portrait of the system for the parameter values $k_\mathrm{p} = 1$, $k_\mathrm{i} = 1$, $u_0 = 1$, and $e_0 = 1$ and discuss the properties of the system. The example illustrates the difficulties of introducing *ad hoc* nonlinearities without careful analysis.

11.12 (Windup stability) Consider a closed loop system with controller transfer function $C(s)$ and process transfer function $P(s)$. Let the controller have windup protection with the tracking constant k_aw. Assume that the actuator model in the anti-windup scheme is chosen so that the process never saturates.

(a) Use block diagram transformations to show that the closed loop system with anti-windup can be represented as a connection of a linear block with transfer function (11.11) and a nonlinear block representing the actuator model.

(b) Show that the closed loop system is stable if the Nyquist plot of the transfer function (11.11) has the property $\operatorname{Re} H(i\omega) > -1$.

(c) Assume that $P(s) = k_\mathrm{v}/s$ and $C(s) = k_\mathrm{p} + k_\mathrm{i}/s$. Show that the system with windup protection is stable if $k_\mathrm{aw} > k_\mathrm{i}/k_\mathrm{p}$.

(d) Use describing function analysis to show that without the anti-windup protection, the system may not be stable and estimate the amplitude and frequency of the resulting oscillation.

(e) Build a simple simulation that verifies the results from part (d).

11.13 Consider the system in Exercise 11.9 and investigate what happens if the second-order filtering of the derivative is replaced by a first-order filter.