Chapter 10

PID Control

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

Desborough Honeywell, 2000, see [DM02].

PID control is by far the most common way of using feedback in natural and man-made systems. PID controllers are commonly used in industry and a large factory may have thousands of them, in instruments and laboratory equipment. In engineering applications the controllers appear in many different forms: as a stand alone controller, as part of hierarchical, distributed control systems, or built into embedded components. Most controllers do not use derivative action. In this chapter we discuss the basic ideas of PID control and the methods for choosing the parameters of the controllers. Many aspects of control can be understood based on linear analysis. However, there is one nonlinear effect, that has to be considered in most control systems namely that actuators saturate. In combinations with controllers having integral actions saturations give rise to an effect called *integral windup*. This phenomenon that occurs in practically all control systems will be discussed in depth for PID controllers. Methods to avoid windup will also be presented. Finally we will also discuss implementation of PID controllers, similar methods can be used to implement many other controllers.

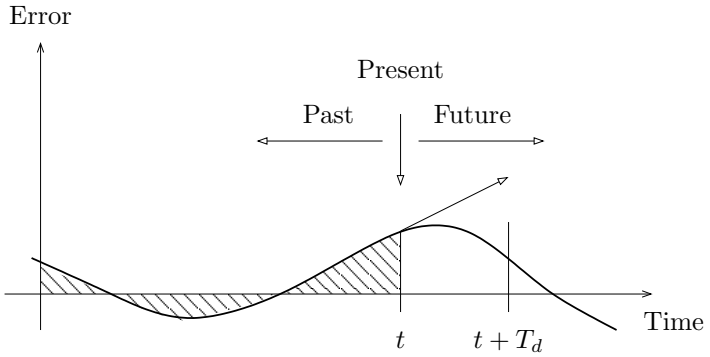

Figure 10.1: A PID controller takes control action based on past, present and prediction of future control errors.

10.1 The Controller

The ideal version of the PID controller is given by the formula

$$u(t) = k_p e(t) + k_i \int_0^t e(\tau)\, d\tau + k_d \frac{de}{dt}, \tag{10.1}$$

where u is the control signal and e is the control error ($e = r - y$). The reference value, r, is also called the *setpoint*. The control signal is thus a sum of three terms: a proportional term that is proportional to the error, an integral term that is proportional to the integral of the error, and a derivative term that is proportional to the derivative of the error. The controller parameters are proportional gain k_p, integral gain k_i and derivative gain k_d. The controller can also be parameterized as

$$u(t) = k_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right), \tag{10.2}$$

where T_i is the integral time constant and T_d the derivative time constant. The proportional part acts on the present value of the error, the integral represents an average of past errors and the derivative can be interpreted as a prediction of future errors based on linear extrapolation, as illustrated in Figure 10.1. Note that the control signal u is formed entirely from the error e, there is no feedforward term (which would correspond to $k_r r$ in the state feedback case). In Section 10.5 we will introduce a modification which also uses feedforward.

We begin our analysis by considering pure proportional feedback. Figure 10.2a shows the responses of the output to a unit step in the command

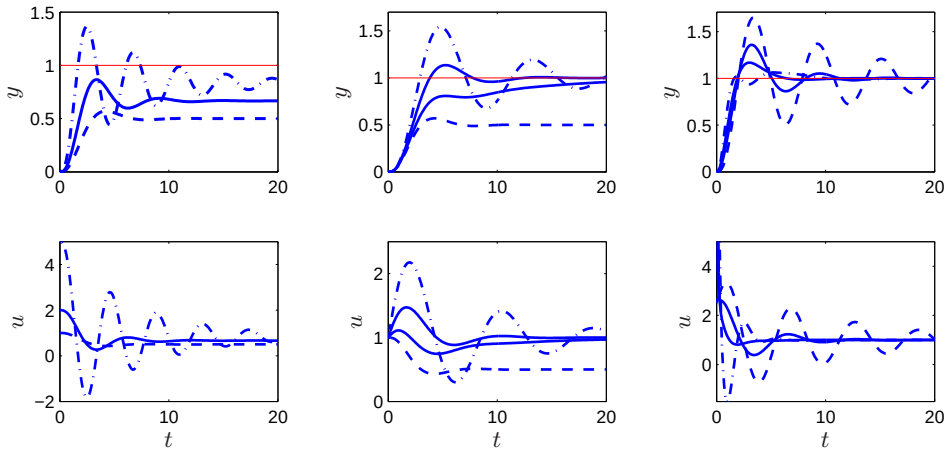

Figure 10.2: Responses to step changes in the command signal for a proportional controller (left), PI controller (center) and PID controller (right). The process has the transfer function $P(s) = 1/(s+1)^3$, the proportional controller (left) had parameters $k_p = 1$ (dashed), 2 and 5 (dash-dotted), the PI controller has parameters $k_p = 1$, $k_i = 0$ (dashed), 0.2, 0.5 and 1 (dash-dotted), and the PID controller has parameters are $k_p = 2.5$, $k_i = 1.5$ and $k_d = 0$ (dashed), 1, 2, 3 and 4 (dash-dotted).

signal 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 non-zero steady state error. Letting the process and the controller have transfer functions $P(s)$ and $C(s)$, the transfer function from reference to output is

$$G_{yr} = \frac{PC}{1+PC}. \tag{10.3}$$

The zero frequency gain with proportional control $C(s) = k_p$ is

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

and thus the steady state error for a unit step is $1 - G_{yr}(0) = 1/(1+k_pP(0))$. For the system in Figure 10.2a with gains $k_p = 1$, 2 and 5, the steady state error is 0.5, 0.33 and 0.17. The error decreases with increasing gain, but the system also becomes more oscillatory. Notice in the figure that the initial value of the control signal equals the controller gain.

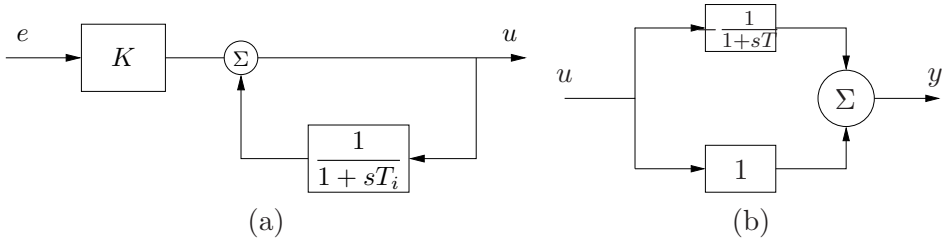

Figure 10.3: Implementation of integral action (left) and derivative action (right) by combining simple blocks.

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

$$u(t) = k_p e(t) + u_d, \tag{10.4}$$

where u_d is a feedforward term that is adjusted to give the desired steady state value. If we choose $u_d = r/P(0) = k_r r$, then the output will be exactly equal to the reference value, as it was in the state space case. However, this requires exact knowledge of the process dynamics, which is usually not available. In early controllers the term u_d, which was also called the reset, was adjusted manually to obtain the desired steady state.

As we saw in Sections 1.5 and 6.4, integral action guarantees that the process output agrees with the reference in steady state and provides an alternative to including a feedforward term. To see the effect of integral action in the frequency domain, we consider a proportional-integral (PI) controller, which has a transfer function

$$C(s) = k_p e + \frac{k_i}{s}.$$

We see that the controller has infinite zero frequency gain ($C(0) = \infty$) and it then follows from equation (10.3) that $G_{yr}(0) = 1$, which implies that there is no steady-state error.

Integral action can also be viewed as a method for generating the feedforward term u_d in the proportional controller (10.4) automatically. An alternative way to represent this action is shown in Figure 10.3a, where the low pass part of the control action of a proportional controller is filtered and feed back with positive gain. This implementation, called *automatic reset*, was one of the early inventions of integral control. Integral action is often realized in this way in biological systems.

The transfer function of the system in Figure 10.3 is obtained by loop tracing: assuming exponential signals, we have

$$u = k_p e + \frac{1}{1+sT}u,$$

and solving for u gives

$$u = k_p \frac{1+sT}{sT} e = k_p + \frac{k_p}{sT},$$

which is the transfer function for a PI controller.

The properties of integral action are illustrated in Figure 10.2b. 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 removed when integral gain action is used. The response creeps slowly towards the reference for small values of k_i, but faster for larger integral gains, but the system also becomes more oscillatory.

We now return to the general PID controller and consider the use of the derivative term, k_d. Recall that the original motivation for derivative feedback was to provide predictive action. The input-output relation of a controller with proportional and derivative action is

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

where $T_d = k_d/k_p$ is the derivative time constant. The action of a controller with proportional and derivative action can be interpreted as if the control is made proportional to the *predicted* process output, where the prediction is made by extrapolating the error T_d time units into the future using the tangent to the error curve (see Figure 10.1).

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

$$C(s) = \Big(1 - \frac{1}{1+sT}\Big) = \frac{sT}{1+sT}U(s).$$

The system thus has the transfer function $G(s) = sT/(1+sT)$, which approximates a derivative for low frequencies. Notice that this implementation gives filtering automatically.

Figure 10.2c illustrates the behavior of a system with a PID controller: the system is oscillatory when no derivative action is used and it becomes

more damped as derivative gain is increased. A comparison of the systems with P, PI and PID control in Figure 10.2 shows that the steady-state error is removed by introducing integral action and that the response speed can be improved by introducing derivative action.

10.2 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 way to do this is to go through the steps of modeling and control design. Since the PID controller has so few parameters a number of special empirical methods have also been developed. A simple idea is to connect a controller, increase the gain until the the system starts to oscillate, and then reduce the gains by an appropriate factor. Another is to measure some features of the open loop response and to determine controller parameters based on these features. We will present the Ziegler-Nichols methods which are the most celebrated tuning rules.

Ziegler-Nichols' Tuning

Ziegler and Nichols develped two techniques for controller tuning in the 1940s. The idea was to tune the controller based on the following idea: Make a simple experiment, extract some features of process dynamics from the experimental data, and determine controller parameters from the features.

One method is based on direct adjustment of the controller parameters. A controller is connected to the process, integral and derivative gain 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. The controller parameters are then given by Table 10.1. The values in the table were obtained based on many simulations and experiments on processes that are normally encountered in process industry. There are many variations of the method which are widely used in industry.

Another method proposed by Ziegler and Nichols is based on determination of the open loop step response of the process, as shown Figure 10.4a. The step response is measured by applying a step input to the process and recording the response. The response is scaled to correspond to a unit step input and characterized by parameters a and T_{del}, which are the intercepts of the steepest tangent of the step response with the coordinate axes. The parameter T_{del} is an approximation of the time delay of the system and

Table 10.1: Controller parameters for the Ziegler-Nichols frequency response method which gives controller parameters in terms of critical gain k_c and critical period T_c. Parameter T_p is an estimate of the period of damped oscillations of the closed loop system.

Controller	k_p/k_c	T_i/T_c	T_d/T_c	T_p/T_c
P	0.5			1.0
PI	0.4	0.8		1.4
PID	0.6	0.5	0.125	0.85

a/T_{del} is the steepest slope of the step response. Notice that it is not necessary to wait until steady state to find the parameters, it suffices to wait until the response has had an inflection point. The controller parameters are given in Table 10.2. The parameters were obtained by extensive simulation of a range of representative processes.

Improved Ziegler-Nichols Rules

There are two drawbacks with the Ziegler-Nichols rules: too little process information is used and the closed loop systems that are obtained lack robustness. Substantially better tuning is obtained by fitting the model

$$P(s) = \frac{K}{1+sT} e^{-sT_{\text{del}}} \tag{10.5}$$

to the step response. A simple way to do this is illustrated in Figure 10.4b. The zero frequency gain of the process K is determined from the steady state value of the step response. The time delay T_{del} is determined from the

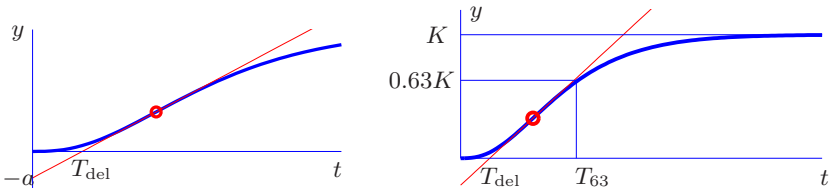

Figure 10.4: Characterization of the unit step response by two (left) and three parameters (right). The point where the tangent is steepest is marked with a small circle.

Table 10.2: Controller parameters for the Ziegler-Nichols step response method. Parameter T_p is an estimate of the period of damped oscillations of the closed loop system.

Controller	ak_p	T_i/T_{del}	T_d/T_{del}	T_p/T_{del}
P	1			4
PI	0.9	3		5.7
PID	1.2	2	$T_{\rm del}/2$	3.4

intercept of the steepest tangent to the step response and the time T_{63} is the time where the output has reached 63% of its steady state value. The parameter T is then given by $T = T_{63} - T_{\rm del}$. Notice that the experiment takes longer time than the experiment in Figure 10.4a because it is necessary to wait until the steady state has been reached. The following tuning formulas have been obtained by tuning controllers to a large set of processes typically encountered in process control

$$\begin{aligned} k_p K &= \min\left(0.4\ T/L,\ 0.25\right) \\ T_i &= \max\left(T,\ 0.5T_{\rm del}\right). \end{aligned} \tag{10.6}$$

Notice that the improved formulas typically give lower controller gain than the Ziegler-Nichols method, and that integral gain is higher, particularly for systems with dynamics that are delay dominated, i.e. $T_{\rm del} > 2T$.

Relay Feedback

The experiment used in the Ziegler-Nichols frequency response method, where the gain of a proportional controller is increased until the system reaches instability, gives the frequency ω_{180} where the process has a phase lag of 180° and the process gain K_{180}at that frequency. Another way to obtain this information is to connect the process in a feedback loop with a relay as shown in Figure 10.5a. For many systems there will then be an oscillation, as shown in Figure 10.5b, where the relay output u is a square wave and the process output y is close to a sinusoid. Notice that the process input and output have opposite phase and that an oscillation with constant period is established quickly.

To provide some analysis, we assume that the relay output is expanded in a Fourier series and that the process attenuates higher harmonics effectively. It is then sufficient to consider only the first harmonic component of the

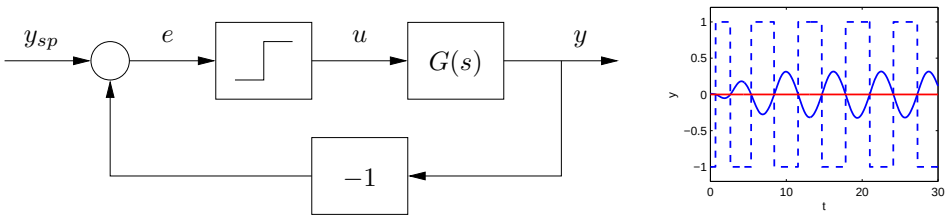

Figure 10.5: Block diagram of a process with relay feedback (left) and typical signals (right). The process output y is solid and the relay output u is dashed. Notice that the signals u and y are out of phase.

input. The input and the output then have opposite phase, which means that the frequency of the oscillation ω_{180} is such that the process has a phase lag of 180°. If d is the relay amplitude, the first harmonic of the square wave input has amplitude $4d/\pi$. Let a be the amplitude of the process output. The process gain at ω_{180} is then given by

$$K_{180} = |P(i\omega_{180}| = \frac{\pi a}{4d}. \tag{10.7}$$

The relay experiment is easily automated. Since the amplitude of the oscillation is proportional to the relay output, it is easy to control it by adjusting the relay output. The amplitude and the period can be determined after about 20 s, in spite of the fact that the system is started so far from the equilibrium that it takes about 8 s to reach the correct level. The settling time for the step response of the system is 40 s.

Automatic tuning based on relay feedback is used in many commercial PID controllers. Tuning is accomplished simply by pushing a button which activates relay feedback. The relay amplitude is automatically adjusted to keep the oscillations sufficiently small, the relay feedback is switched to a PID controller as soon as the tuning is accomplished.

10.3 Modeling and Control Design

Parameters of PID controllers can also be determined by modeling process dynamics and applying some method for control design. Since the complexity of the controller is directly related to the complexity of the model it is necessary to have models of low order.

To illustrate the ideas we will consider the case where a process dynamics

is approximated by a first order transfer function

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

The approximation is reasonable for systems where storage of mass, momentum and energy can be captured by one state variable. Typical examples are the velocity of a car on the road, control of the velocity of a rotating system, electric systems where energy is essentially stored in one component, incompressible fluid flow in a pipe, level control of a tank, pressure control in a gas tank and temperature in a body with uniform temperature distribution.

A PI controller has the transfer function

$$C(s) = k_p + \frac{k_i}{s},$$

and the transfer function of the closed loop system from reference to output is

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

The closed loop system has the characteristic polynomial

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

Assuming that the desired characteristic polynomial is

$$s^2 + 2\zeta\omega_0 s + \omega_0^2, \tag{10.8}$$

we find that the controller parameters are given by

$$\begin{aligned} k_p &= \frac{2\zeta\omega_0 - a}{b} = \frac{2\zeta\omega_0 T - 1}{K} \\ k_i &= \frac{\omega_0^2}{b} = \frac{\omega_0^2 T}{K}. \end{aligned} \tag{10.9}$$

The parameter ω_0 determines the response speed and ζ determines the damping. Notice that controller gain becomes negative if $\zeta\omega_0 < 0.5$, which gives a closed loop system with bad properties, as will be discussed in Section 12.5.

The same approach can be used to find the parameters of a PID controller for a process with dynamics of second order (Exercise 1).

Example 10.1 (Cruise control design)**.** Consider the problem of maintaining the speed of a car as it goes up a hill. In Example 5.14we found that there was very little difference between the linear and nonlinear models when investigating PI control provided that the throttle does not reach the saturation limits. We will now use a linear model to design a controller and to investigate the effects of design parameters. A simple linear model of a car was given in Example 5.10:

$$\frac{d(v - v_e)}{dt} = a(v - v_e) + b(u - u_e) - g\theta, \tag{10.10}$$

where v is the velocity of the car, u is the input from the engine and θ is the slope of the hill. The parameters were $a = -0.101$, $b = 1.3203$, $g = 9.8$, $v_e = 20$, and $u_e = 0.1616$. This model will be used to find suitable parameters of a vehicle speed controller. To investigate the behavior of the closed loop system we start with the linearized process model (10.10) and we assume that the cruise controller is PI controller is described by

$$u = k_p(v_e - v) + k_i \int_0^t (v_e - v(\tau))d\tau. \tag{10.11}$$

To compute the error dynamics, introduce the error $e = v_e - v$, differentiate equations (10.10) and (10.11), and eliminate u. The error is then given by the differential equation

$$\frac{d^2e}{dt^2} + (-a + bk_p)\frac{de}{dt} + bk_ie = 9.8\frac{d\theta}{dt}. \tag{10.12}$$

We notice that the steady state error is zero if θ and e are constant, which is no surprise since the controller has integral action.

To understand the effects of the controller parameters K and k_i we can compare equation (10.12) with the standard second order system

$$\ddot{q} + 2\zeta\omega_0\dot{q} + \omega_0^2 q = ku.$$

This gives

$$k_p = \frac{a + 2\zeta\omega_0}{b} \qquad k_i = \frac{\omega_0^2}{b}$$

where ζ denotes relative damping and ω_0 is the undamped natural frequency. The parameter ω_0 gives response speed and ζ determines the shape of the response. Since it is desirable that a cruise control system should respond to changes smoothly without oscillations we choose $\zeta = 1$, which corresponds to critical damping.

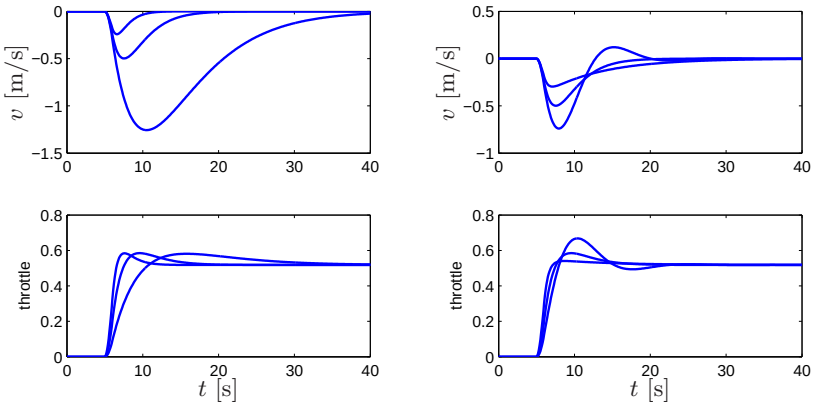

Figure 10.6: Illustrates the effect of parameters ω_0 (left) and ζ_0 (right) on the response of a car with cruise control.

The consequences of different choices of ω_0 and ζ_0 are illustrated in Figure 10.6. The figure shows the velocity and throttle for a car that first moves on a horizontal road and encounters a hill with slope 4° at time 6. Between time 5 and 6 the slope increases linearly. The choice of ω_0 is a compromise between response speed and control actions. The compromise is illustrated in Figure 10.6, which shows the velocity error and the control signal for a simulation where the slope of the road suddenly changes by 4°. The largest velocity error decreases with increasing ω_0, but the control signal also changes more rapidly. In the simple model (10.10) it was assumed that the force responds instantaneously to throttle commands. For rapid changes there may be additional dynamics that has to be accounted for. There are also physical limitations to the rate of change of the force, which also restrict the admissible value of ω_0. A reasonable choice of ω_0 is in the range of 0.2 to 1.0. Notice in Figure 10.6 that even with $\omega_0 = 0.1$ the largest velocity error is only 1 m/s.

Another interpretation of the effect of the integral action can be given by returning to the basic force balance model of the car

$$m\frac{dv}{dt} = F - F_d,$$

where m is the mass of the car, F is the applied force (from the engine) and F_d is the disturbance force (aerodynamic drag and force of gravity). Since zero steady state error implies that v is constant, we see that the PI controller generates an output force F that in steady state is equal to the

drag force F_d. Since the error is zero in steady state the controller output equals the output of the integrator of the PI controller. The output of the integrator in the PI controller can thus be interpreted as an estimator of the drag force. ∇

10.4 Integrator Windup

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

Example 10.2 (Cruise control)**.** The windup effect is illustrated in Figure 10.7, 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. The torque required is however 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 the gravity. The error is large and the integral continues to build up until the error reaches zero at time 30, but but the controller output is still much larger than the saturation limit and the actuator remains saturated. The integral term starts to decrease and at time 45 and the velocity settles to quickly to the desired value. Notice that it takes considerable time before the controller output comes in the range where it does not saturate resulting in a large overshoot. ∇

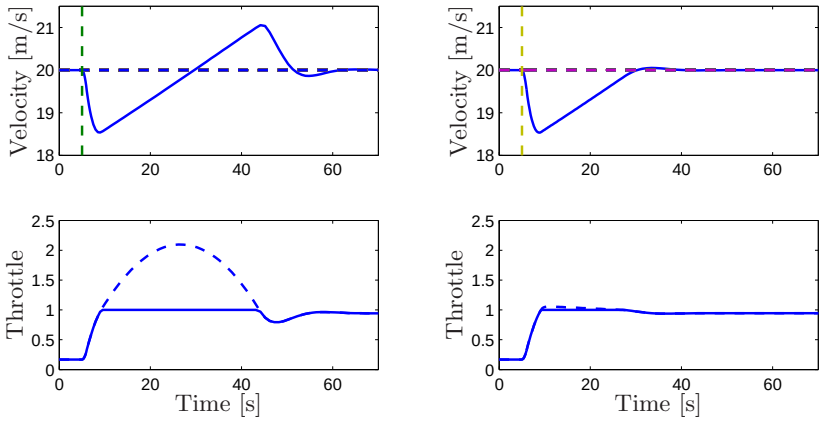

Figure 10.7: Simulation of windup (left) and anti-windup (right). The figure shows the speed v and the throttle u for a car that encounters a slope that is so steep that the throttle saturates. The controller output is dashed. The controller parameters are $k_p = 0.5$ and $k_i = 0.1$.

Avoiding Windup

There are many ways to avoid windup. One method is illustrated in Figure 10.8: 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 feedback to the integrator in such a way that e_s goes towards zero. This implies that controller output is kept close to the saturation limit. The controller output will then change as soon as the error changes sign and integral windup is avoided.

The rate at which the controller output is reset is governed by the feedback gain, k_t, a large value of k_t give a short reset time. The parameter k_t can, however, not be too large because measurement error can then cause an undesirable reset. A reasonable compromise is to choose $k_t \approx 1/T_i$ for PI control and as $k_t \approx 1/\sqrt{T_i T_d}$ for PID control. We illustrate how integral windup can be avoided by investigating the cruise control system.

Example 10.3 (Cruise control)**.** Figure 10.7b shows what happens when

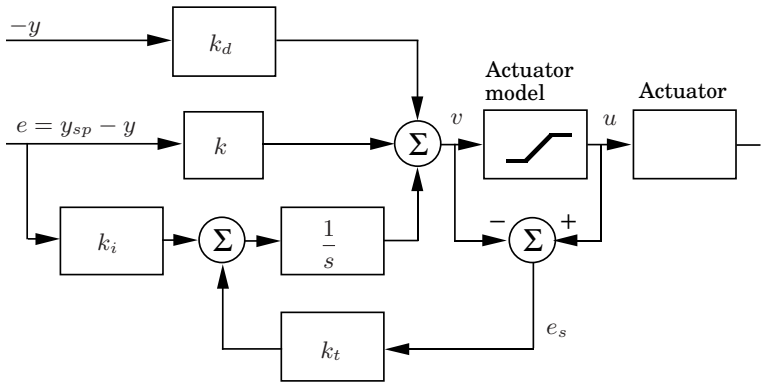

Figure 10.8: PID controller with anti-windup.

a controller with anti-windup is applied to the system simulated in Figure 10.7a. Because of the feedback from the actuator model the output of the integrator is quickly reset to a value such that the controller output is at the saturation limit. The behavior is drastically different from that in Figure 10.7a and the large overshoot is avoided. The tracking gain $k_t = 2$ in the simulation. ∇

10.5 Implementation

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

Filtering the Derivative

A drawback with derivative action is that an ideal derivative has very high gain for high frequency signals. This means that high frequency measurement noise will generate large variations of the control signal. The effect of measurement noise be reduced by replacing the term $k_d s$ by

$$D_a = -\frac{k_d s}{1 + sT_f}. \tag{10.13}$$

This can be interpreted as an ideal derivative that is filtered using a first-order system with the time constant T_f. For small s the transfer function is

approximately $k_d s$ and for large s it is equal to k_d/T_f. The approximation acts as a derivative for low-frequency signals and as a constant gain for the high frequency signals. The filtering time is chosen as $T_f = (k_d/k)/N$, with N in the range of 2 to 20. Filtering is obtained automatically if the derivtive is implemented by taking difference between the signal and a its filtered version as shown in Figure 10.2.

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

$$C(s) = k_p \left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d/N}\right). \tag{10.14}$$

The high-frequency gain of the controller is $K(1 + N)$. Instead of filtering just the derivative it is also possible to use an ideal controller and filter the measured signal. The transfer function of such a controller with the filter is then

$$C(s) = k_p \left(1 + \frac{1}{sT_i} + sT_d\right) \frac{1}{(1 + sT_f)^2}. \tag{10.15}$$

where a second order filter is used.

Setpoint Weighting

The control system in equation (10.1) is called a system with *error feedback* because the controller acts on the error, which is the difference between the reference and the output. In the simulation of PID controllers in Figure 10.1 there is a large initial peak of the control signal, which is caused by the derivative of the reference signal. The peak can be avoided by modifying the controller equation (10.1) to

$$u = k_p\big(\beta r - y\big) + k_i \int_0^\infty \big(r(\tau) - y(\tau)\big)\, d\tau + k_d\Big(\gamma \frac{dr}{dt} - \frac{dy}{dt}\Big) \tag{10.16}$$

In this controller, 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*.

Figure 10.9 illustrates the effects of setpoint weighting on the step response. The figure shows clearly the effect of changing β. The overshoot for reference changes is smallest for $\beta = 0$, which is the case where the reference is only introduced in the integral term, and increases with increasing

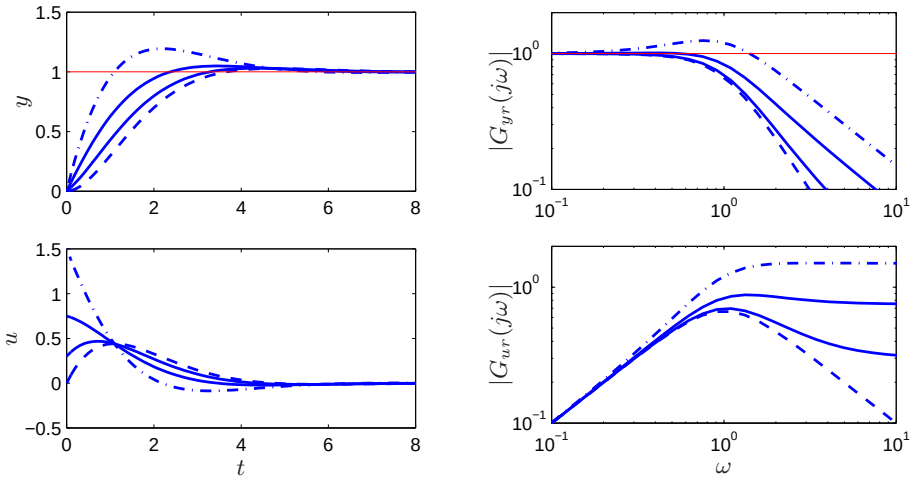

Figure 10.9: Time and frequency responses for system with PI controller and setpoint weighting. The curves on the left show responses in process output y and control signal and the curves on the right show the gain curves for the transfer functions $G_{yr}(s)$ and $G_{ur}(s)$. The process transfer function is $P(s) = 1/s$, the controller gains are $k = 1.5$ and $k_i = 1$, and the setpoint weights are $\beta = 0$ (dashed) 0.2, 0.5 and 1 (dash dotted).

β. Parameter β it typically in the range of 0 to 1 and γ is normally zero to avoid large transients in the control signal when the reference is changed.

The controller given by equation (10.16) is a special case of controller with two degrees which will be discussed in Section 11.1.

Implementation based Operational Amplifiers

PID controllers have been implemented in many different technologies. Figure 10.10 shows how they can be implemented by feedback around operational amplifiers.

To show that the circuit in Figure 10.10b is a PID controller we will use the approximate relation between the input voltage e and the output voltage u of an operational amplifier in Section 3.3:

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

where Z_0 is the impedance between the negative input of the amplifier and the input voltage e, and Z_1 is the impedance between the zero input of the

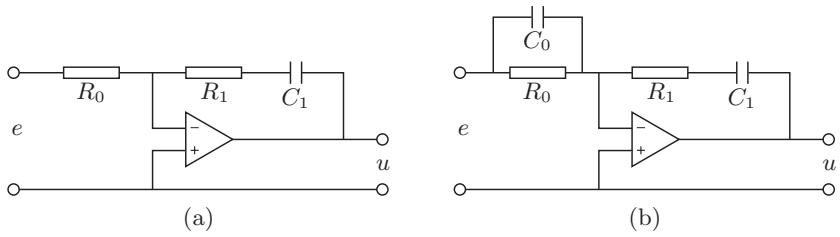

Figure 10.10: Schematic diagram of an electronic PI (left) and PID controllers (right) based on feedback around an operational amplifier.

amplifier and the output voltage u. The impedances are given by

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

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

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

This is the input-output relation for a PID controller on the form (10.2) with parameters

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

The corresponding results for a PI controller is obtained by setting $C_0 = 0$.

Computer Implementation

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

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

6. Repeat

Notice that an output is sent to the actuators as soon as it is available. The time delay is minimized by making the calculations in Step 3 as short as possible and performing all updates after the output is commanded.

As an illustration we consider the PID controller in Figure 10.8, which has a filtered derivative, setpoint weighting and protection against integral windup. The controller is a continuous time dynamical system. To implement it using a computer, the continuous time system has to be approximated by a discrete time system.

The signal v is the sum of the proportional, integral and derivative terms

$$v(t) = P(t) + I(t) + D(t) \tag{10.17}$$

and the controller output is $u(t) = \text{sat}(v(t))$ where sat is the saturation function that models the actuator. The proportional term is

$$P = k_p(\beta y_{sp} - y).$$

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

$$P(t_k) = k_p\left(\beta y_r(t_k) - y(t_k)\right), \tag{10.18}$$

where $\{t_k\}$ denotes the sampling instants, i.e., the times when the computer reads its input. The integral term is

$$I(t) = k_i \int_0^t e(s)ds + \frac{1}{T_t}\big(\text{sat}(v) - v\big)$$

and approximating the integral by a sum gives

$$I(t_{k+1}) = I(t_k) + k_i h\, e(t_k) + \frac{h}{T_t}\big(\text{sat}(v) - v\big). \tag{10.19}$$

The derivative term D is given by the differential equation

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

Approximating this equation with a backward difference we find

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

which can be rewritten as

$$D(t_k) = \frac{T_f}{T_f + h} D(t_{k-1}) - \frac{k_d}{T_f + h} (y(t_k) - y(t_{k-1})) . \tag{10.20}$$

The advantage of using a backward difference is that the parameter $T_f/(T_f + h)$ is nonnegative and less than one for all $h > 0$, which guarantees that the difference equation is stable.

Reorganizing equations (10.17)–(10.20), the PID controller can be described by the following pseudo code:

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

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

Precomputation of the coefficients `bi`, `ad`, `bd` and `br` saves computer time in the main loop. These calculations have to be done only when controller parameters are changed. The main loop is executed once every sampling period. The program has three states: `yold`, `I`, and `D`. One state variable can be eliminated at the cost of a less readable code. Notice that the code includes computing the derivative of the process output, proportional action on a portion of the error ($b \neq 1$), and modeling of the actuator saturation in the integral computation to give protection against windup.

10.6 Further Reading

The history of PID control is a very rich one and stretches back to the beginning of the foundation of control theory. A very readable treatment is

given by Mindel [Min02]. A comprehensive presentation of PID control is given in [ÅH95].

10.7 Exercises

1. Consider a second order process with transfer function

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

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

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

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

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