Chapter 8

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.

This chapter describes the PID controller which unquestionably the most common way of solving practical control problem. Practical implementation issues are also discussed particularly mechanisms for avoiding integrator windup. Methods for automatic tuning of a PID controller are also discussed.

8.1 Introduction

The PID controller is by far the most common control algorithm. Most practical feedback loops are based on PID control or some minor variations of it. Many controllers do not even use derivative action. The PID controllers appear in many different forms, as a stand-alone controllers, they can also be part of a DDC (Direct Digital Control) package or a hierarchical distributed process control system or they are built into embedded systems. Thousands of instrument and control engineers worldwide are using such controllers in their daily work. The PID algorithm can be approached from many different directions. It can be viewed as a device that can be operated with a few empirical rules, but it can also be approached analytically.

This chapter gives an introduction to PID control. The basic algorithm and various representations are presented in detail. A description of the properties of the controller in a closed loop based on intuitive arguments is given. The phenomenon of reset windup, which occurs when a controller with integral action is connected to a process with a saturating actuator, is discussed, including several methods to avoid it. Filters to reduce noise influence and means to improve reference responses are also provided.

Implementation aspects of the PID controller are presented in Chapter **??**.

8.2 The PID Controller

The textbook version of the PID controller is

$$u(t) = ke(t) + k_i \int_0^t e(\tau)d\tau + k_d \frac{de}{dt}, \tag{8.1}$$

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

$$u(t) = k\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right), \tag{8.2}$$

where T_i is called integral time and T_d derivative time. The proportional part acts on the present value of the error, the integral represent and average of past errors and the derivative can be interpreted as a prediction of future errors based on linear extrapolation, see Figure 8.1.

Proportional Action

Figure 8.2 shows the response of the output to a unit step in the command signal for a system with pure proportional control. The output never reaches the steady state error. Let the process and the controller have transfer functions $P(s)$ and $C(s)$. The transfer function from reference to output is

$$G_{yr}(s) = \frac{P(s)C(s)}{1 + P(s)C(s)} \tag{8.3}$$

The steady state gain with proportional control $C(s) = k$ is

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

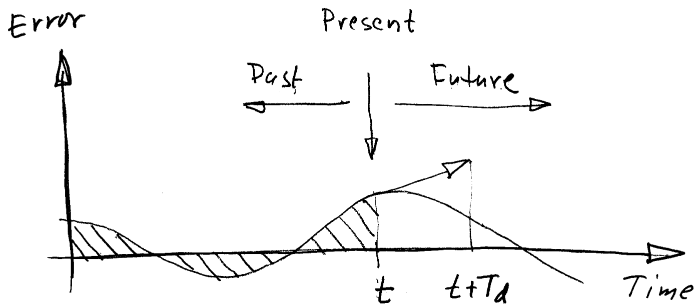

Figure 8.1: A PID controller takes control action based on past, present and prediction of future control errors.

The steady state error for a unit step is thus $1/(1+kP(0)$. For the system in Figure 8.2 with gains $k = 1$, 2 and 5 the steady state error is 0.5, 0.33 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 controller gain. To avoid having a steady state error the proportional controller can be change to

$$u(t) = Ke(t) + u_b. \tag{8.4}$$

where u_b is a bias or reset term which is adjusted to give the desired steady state value.

Integral Action

Integral action guarantees that the process output agrees with the reference in steady state. This can be shown as follows. Assume that the system is in steady state with a constant control signal (u_0) and a constant error $e_0 \neq 0$. It follows from Equation (8.1) that

$$u_0 = ke_0 + k_i e_0 t.$$

The left hand side is constant but the right hand side is a function of t. We thus have a contradiction and e_0 must be zero. Notice that in this argument the only assumption made is that there exist a steady state. Nothing specific is said about the process, it can for example be nonlinear.

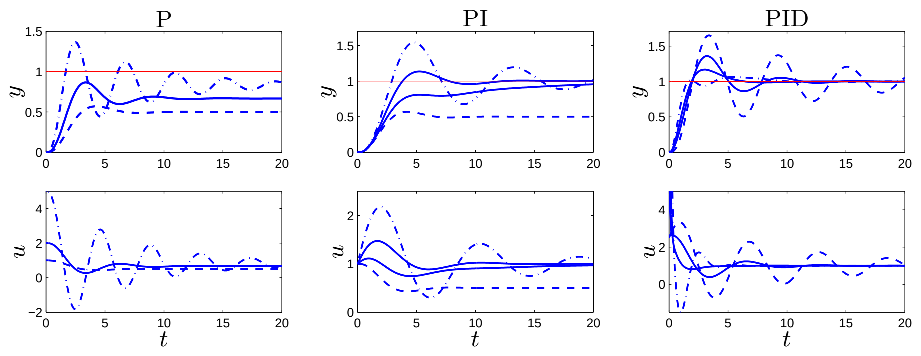

Figure 8.2: Responses to step changes in the command signal for proportional (left), PI (middle) and PID controllers (right). The process has the transfer function $P(s) = 1/(s+1)^3$, the controller parameters are $k = 1$ (dashed), 2 and 5 (dash-dotted) for the P controller, $k = 1$, $k_i = 0$ (dashed), 0.2, 0.5 and 1 (dash-dotted) for the PI controller and $k = 2.5$, $k_i = 1.5$ and $k_d = 0$ (dashed), 1, 2, 3 and 4 (dash-dotted) for the PID controller.

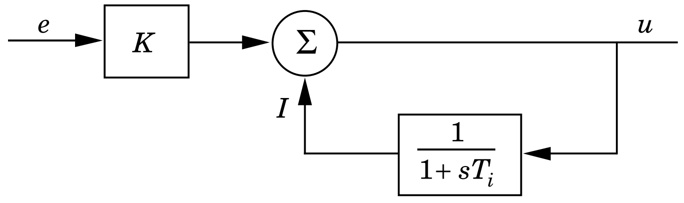

Figure 8.3: Implementation of integral action as automatic bias adjustment.

Another argument is that the transfer function of a controller with integral action has infinite gain at zero frequency ($C(0) = \infty$). It then follows from (8.3) that $G_{yr}(0) = 0$. This argument requires however that the system is linear.

Integral action can also be viewed as a method for generating the bias term u_b in the proportional controller (8.4) automatically. This is illustrated in Figure 8.3, where the bias u_b is generated by low pass filtering the output. This implementation, called *automatic reset*, was one of the early inventions of integral control. The transfer function of the system in Figure 8.3 is obtained by loop tracing. Assuming exponential signals and tracing them around the loop gives

$$u = ke + \frac{1}{1+sT}u.$$

Solving for u gives

$$u = k\frac{1+sT}{sT}e = \Big(k + \frac{k}{sT}\Big),$$

which is the transfer function of a PI controller.

The properties of integral action are illustrated in Figure 8.2. The proportional gain is constant, $k = 1$, and the integral gain is changed. The case $k_i = 0$ corresponds to pure proportional control, with a steady state error is 50%. The steady state error is removed when integral gain k_i is increased. The response creeps slowly towards the reference for small values of k_i. The approach is faster for larger integral gains but the system also becomes more oscillatory.

Derivative Action

Figure 8.2 shows that derivative action can improve the stability of the the closed-loop system. The input-output relation of a controller with proportional and derivative action is

$$u(t) = ke(t) + k_d\frac{de}{dt} = k\big(e(t) + T_d\frac{de}{dt}\big) = ke_p(t)$$

where $T_d = k_d/d$ is the derivative time. 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 8.1). Figure **??** illustrates the behaviour of a system with a PID controller. The system is oscillatory when not derivative action is used and it becomes more damped as derivative gain is increased.

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_ds by

$$D_a = -\frac{k_ds}{1+sT_f}. \tag{8.5}$$

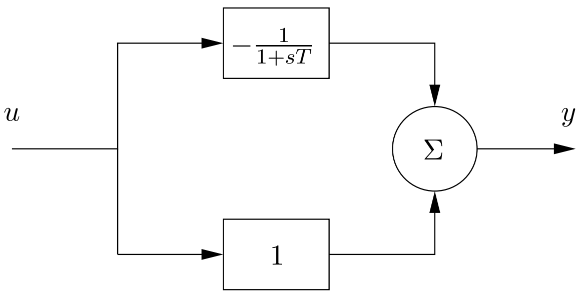

Figure 8.4: Implementation of the transfer function $sT/(1 + sT)$ which approximates derivative action.

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 high-frequency gain is k_d/Tf. The filtering time is chosen as $k_d/k/N$, with N in the range of 2 to 20. The transfer function of a PID controller with a filtered derivative is

$$C(s) = K\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + sT_d/N}\right). \tag{8.6}$$

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

$$C(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right)\frac{1}{(1 + sT_f)^2}. \tag{8.7}$$

where a second order filter is used.

An early implementation of derivative action is shown in Figure 8.4. In this system the derivative is shown as the difference between the signal and a filtered version of the signal. The transfer function for the system is

$$C(s) = \left(1 - \frac{1}{1 + sT}\right) = \frac{sT}{1 + sT}U(s). \tag{8.8}$$

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.

Set Point Weighting

The control system in (8.1) 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 8.1 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 (8.1) to

$$u(t) = k\big(\beta r(t) - y(t)\big) + k_i \int_0^\infty \big(r(\tau) - y(\tau)\big) d\tau + k_d\Big(\gamma \frac{dr(t)}{dt} - \frac{dy(t)}{dt}\Big) \tag{8.9}$$

In this controller proportional and derivative action only acts on a 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 be different because it depends on the values of β and γ, which are called *reference weights* or *setpoint weights*.

Figure 8.5 illustrates the effects of set point 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 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 (8.9) is a special case of the general controller with two degrees of freedom in Figure **??**. The transfer functions are

$$\begin{aligned} C(s) &= k + \frac{k_i}{s} + k_d s \\ F(s) &= \frac{\gamma k_d s^2 + \beta k s + k_i}{k_d s^2 + ks + k_i}. \end{aligned}$$

8.3 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. There 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 control system with a wide range of operating conditions, it may happen that the control variable reaches the actuator limits. When this happens

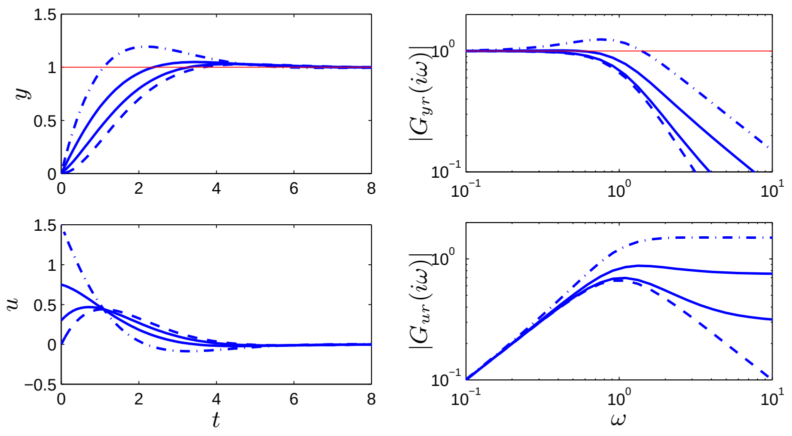

Figure 8.5: Time and frequency responses for system with PI controller and setpoint weighting. The curves on the left show responses in process output y and control signal and the curves on the right show the gain curves for the transfer functions $G_{yr}(s)$ and $G_{ur}(s)$. The process transfer function 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).

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. For a controller with integral action the integral term may become very large. When this happens the error must change sign for a long period before the integrator winds down. The consequence is that there may be large transients. This is colloquially referred to as *integrator wind up*

The wind-up effect is illustrated in Figure 8.6, which shows control of an integrating process with a PI controller. The initial reference signal is so large that the actuator saturates at the high limit. The integral term increases initially because the error is positive; it reaches its largest value at time $t = 10$ when the error goes through zero. The output remains saturated at this point because of the large value of the integral term. It does not leave saturation until the error has been negative for a sufficiently long time. Notice that the control signal bounces between its limits several times. The net effect is a large overshoot and a damped oscillation where the control signal flips from one extreme to the other as in relay oscillations. The output finally comes so close to the reference that the actuator does

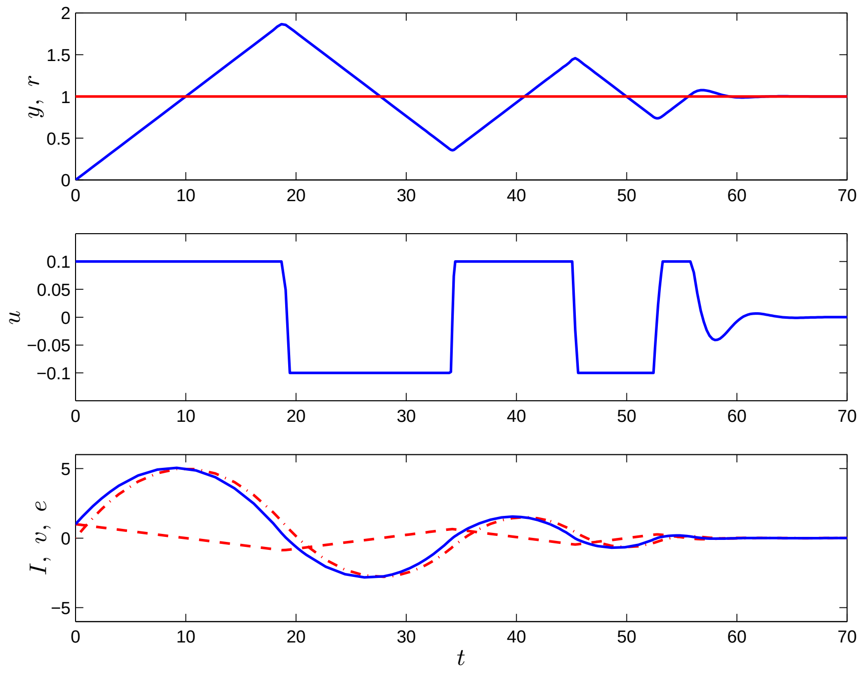

Figure 8.6: Illustration of integrator windup. The plots show process output y, reference r in the upper plot, control signal u in the middle plot, controller output u_o (full) and integral part I and control error e (dashed) in lower part. (dash dotted).

not saturate and the system then behaves linearly and settles quickly.

There are many ways to avoid windup, one method is illustrated in Figure 8.7. 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 $1/T_t$. 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 different from zero. The normal feedback path around the process is broken because the process input remains constant. The feedback around the integrator will act and it attempts to drive e_s to zero. This implies that controller output is kept close to the saturation limit and integral windup is avoided. The rate at which the controller output is reset is governed by the

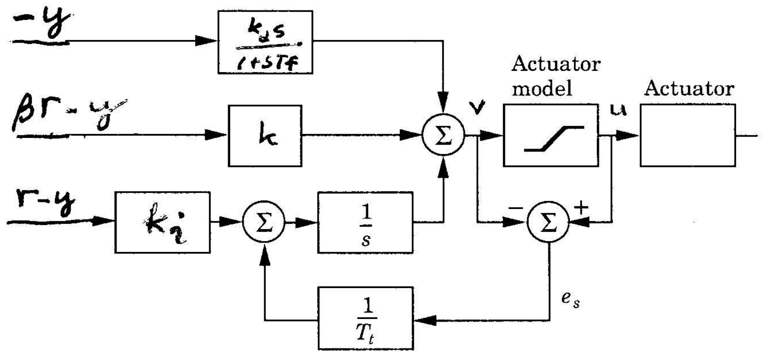

Figure 8.7: PID controller with anti-windup.

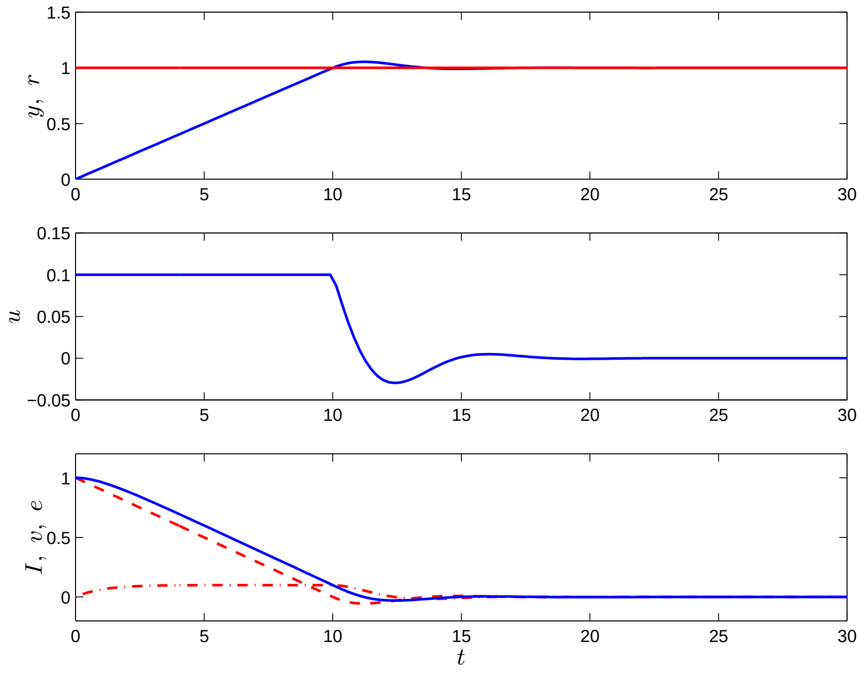

Figure 8.8: Controller with anti-windup applied to the system of Figure 8.6. The plots show process output y, reference r in the upper plot, control signal u in the middle plot, controller output u_o (full) and integral part I and control error e (dashed) in lower part. (dash dotted).

feedback gain, $1/T_t$, where the tracking time constant T_t can be interpreted as the time constant, which determines how quickly the integral is reset. A short long time constant gives a slow reset and a short time constant a short reset time. Measurement error can cause an undesirable reset if the time constant is too short. A reasonable compromise is to choose T_t as a fraction of T_i for proportional control and as $T_t = \sqrt{T_i T_d}$ for PID control.

Figure 8.8 shows what happens when a controller with anti-windup is applied to the system simulated in Figure 8.6. The output of the integrator is quickly reset to a value such that the controller output is at the saturation limit, and the integral has a negative value during the initial phase when the actuator is saturated. This behavior is drastically different from that in Figure 8.6, where the integral was positive during the initial transient. Also notice the drastic improvement in performance compared to the ordinary PI controller used in Figure 8.6.

8.4 Tuning

There are many ways to tune a PID controller. Traditional control techniques based on modeling and design can be used, but there are also special methods for direct tuning based on simple process experiments. A few methods are described in this section.

PI Control of First Order Systems

The dynamics of many systems can be approximated by a first order system with the 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 velocity of car on the road, control of velocity of 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, temperature in a body with essentially uniform temperature distribution.

A PI controller with set point weighting is described by

$$\begin{aligned} C(s) &= k + \frac{k_i}{s} \\ F(s) &= \frac{\beta k s k_i}{ks + k_i}, \end{aligned}$$

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

$$G_{yr}(s) = \frac{P(s)C(s)F(s)}{1+P(s)C(s)} = \frac{b(\beta k s + k_i)}{s^2 + (a+bk)s + bk_i}.$$

The closed loop system has the characteristic polynomial

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

Assuming that the desired characteristic polynomial is

$$s^2 + 2\zeta\omega_0 s + \omega_0^2, \tag{8.10}$$

we find that the controller parameters are given by

$$\begin{aligned} k &= \frac{2\zeta\omega_0 - a}{b} = \frac{2\zeta\omega_0 T - 1}{K} \\ k_i &= \frac{\omega_0^2}{b} = \frac{\omega_0^2 T}{K}. \end{aligned} \tag{8.11}$$

The parameter ω_0 determines the response speed and ζ determines the damping.

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

Loop Shaping

Since a PI controller has two parameters it is possible to shape the loop transfer function by specifying one point on the Nyquist curve. For example, we can choose controller gains to give a specified phase margin at a given crossover frequency ω_{gc}. To be specific let the process transfer function be $P(s)$. The frequency response of the loop transfer function with PI control is

$$\begin{aligned} L(i\omega) &= P(i\omega)\Big(k - i\frac{k_i}{\omega}\Big) = \big(a(\omega) + ib(\omega)\big)\Big(k - i\frac{k_i}{\omega}\Big) \\ &= a(\omega)k + \frac{b(\omega)k}{\omega} + i\Big(b(\omega)k - \frac{a(\omega)k_i}{\omega}\Big), \end{aligned}$$

where $a(\omega) = \mathrm{Re}P(i\omega) = r(\omega)\cos\varphi(\omega)$ and $b(\omega) = \mathrm{Im}P(i\omega) = r(\omega)\sin\varphi(\omega)$. Requiring that the phase margin is φ_m we get

$$\begin{aligned} a(\omega_{gc})k + \frac{b(\omega_{gc})k_i}{\omega_{gc}} &= -\cos\varphi_m \\ b(\omega_{gc})k - \frac{a(\omega_{gc})k_i}{\omega_{gc}} &= -\sin\varphi_m \end{aligned}$$

Solving this equation gives the controller parameters

$$\begin{aligned} k &= -\frac{a(\omega_{gc})\cos\varphi_m + b(\omega_{gc})\sin\varphi_m}{a^2(\omega_{gc}) + b^2(\omega_{gc})} = -\frac{\cos\left(\varphi(\omega) - \varphi_m\right)}{r(\omega)} \\ k_i &= \frac{\big(a(\omega_{gc})\sin\varphi_m - b(\omega_{gc})\cos\varphi_m\big)\omega_{gc}}{a^2(\omega_{gc}) + b^2(\omega_{gc})} = \frac{\sin\left(\varphi(\omega) + \varphi_m\right)}{r(\omega)}. \end{aligned} \tag{8.12}$$

We have

$$\arg P(i\omega_{gc}) + \arg C(i\omega_{gc}) \geq -\pi + \varphi_m.$$

since a PI controller has phase lag between 0 and $\pi/2$ we find that the gain crossover frequency must be chosen so that

$$-\pi + \varphi_m \leq \arg P(i\omega_{gc}) \leq -\pi/2 + \varphi_m \tag{8.13}$$

It follows from(8.12) that integral gain is zero at the lower limit and proportional gain is zero at the higher limit.

Figure 8.9 shows the Nyquist plots for the loop transfer function for different ω_{gc} for a system with the transfer function $P(s) = 1/(s+1)^4$. With a phase margin $\varphi_m = \pi/3$ (8.13) becomes

$$0.13 = \tan\pi/24 < \omega_{gc} < \tan\pi/6 = 0.58$$

The lower limit correspond to a purely integrating controller and the upper limit is a purely proportional controller.

In Section **??** it was shown that k_i is a good measure for load disturbance attenuation and that the stability margin $m_s = 1/M_s$ is a good robustness measure. These measures are shown in Figure 8.9. The largest value $k_i = 0.30$ is obtained for $\omega_{gc} = 0.36$, the largest stability margin $s_m = 0.48$ is obtained for $\omega_{gc} = 0.18$. Reasonable values of the gain crossover frequency are between 0.18 and 0.36. For $\omega_{gc} = 0.3$ we get $k = 0.71$, $k_i = 0.29$ and $s_m = 0.67$ and for $\omega_{gc} = 0.36$ we get $k = 0.96$, $k_i = 0.30$ and $s_m = 0.55$.

Ziegler-Nichols' Tuning Methods

Two special methods for tuning of PID controllers developed by Ziegler and Nichols in the 1940s are still commonly used. They are 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 the open-loop step response, which is characterized by two parameters a and T_{del}, as shown in Figure 8.10. The step

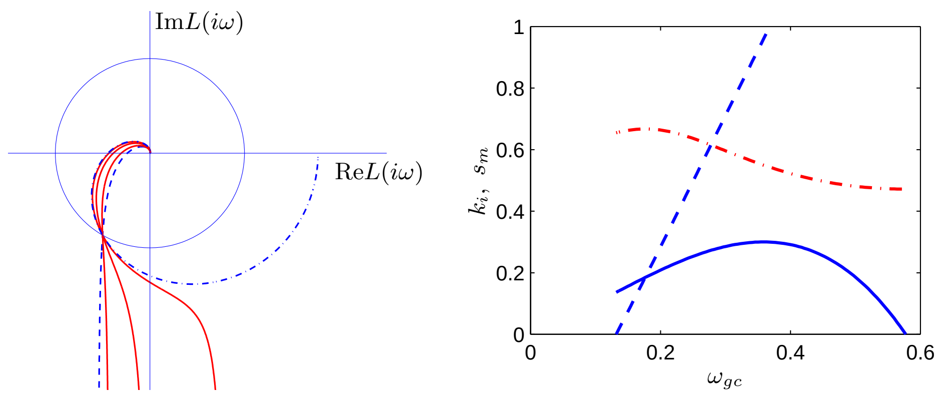

Figure 8.9: The left figure shows the loop transfer functions for PI controllers gain crossover frequencies are $\omega_{gc} = 0.13$ (dashed), 0.3, 0.4, 0.5 (full) and 0.58 (dash-dotted). The right figure shows controller gains k (full), k_i (dashed) and stability margin s_m (dash-dotted) as functions of the phase margin ω_{gc}.

response is characterized by parameters a and T_{del} which are the intercepts of the steepest tangent of the step response with the coordinate axes. Parameter T_{delat} is an approximation of the time delay of the system and 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 8.1.

Another method is based on frequency response features was also developed by Ziegler and Nichols. Process data is obtained by connecting

Table 8.1: 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	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 /2	3.4

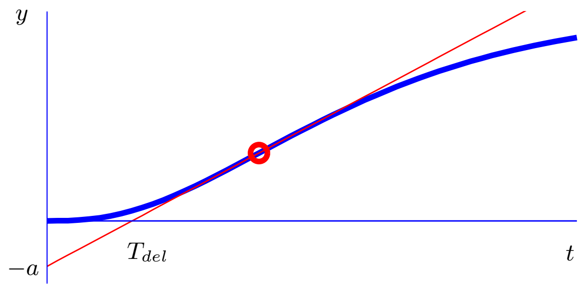

Figure 8.10: Characterization of the unit step response by parameters a and T_{del} which are the intercepts of the steepest tangent to the response with the coordinate axes. The point where the tangent is steepest is marked with a small circle.

Table 8.2: 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/k_c	T_i/T_c	T_d/T_c	T_p/T_u
P	0.5			1.0
PI	0.4	0.8		1.4
PID	0.6	0.5	0.125	0.85

a feedback loop with proportional control. The gain of the controller is increased until the system reaches the stability boundary, the gain of the controller k_c and the period T_c of the oscillation is observed. The controller parameters are then given by Table 8.2.

Improved Ziegler-Nichols Rules

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

$$P(s) = \frac{K}{1+sT}e^{-sT_d} \tag{8.14}$$

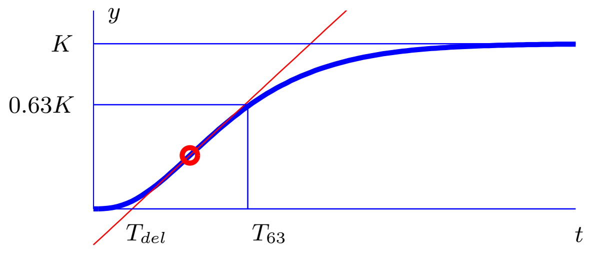

Figure 8.11: Finding the parameters of the model (8.14) from a unit step response.

to the step response. A simple way to do this is illustrated in Figure **??**. The steady state 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 intercept of the stepest tangent to the step response as in Figure 8.10 and the time T_{63} is the time where the output has reached 63% of its steady state value. Parameter T is given by $T = T_{63} - T_{del}$. Notice that the experiment takes longer time than the experiment in Figure 8.10 because it is necessary to way 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

$$\begin{aligned} kK &= \min\left(0.4\frac{T}{L},\ 0.25\right) \\ k_iKT_{del} &= \max\left(0.1\frac{T}{T_{del}},\ 0.5\right). \end{aligned} \tag{8.15}$$

Figure 8.12 illustrates the relations between the controller parameters and the process parameters. The controller gain is normalized by multiplying it either with the static process gain K or with the parameter $a = KT_{del}/T$. Integral gain is normalized by multiplication with KT_{del} and integration time by division by T_{del}. The controller parameters in Figure 8.12 are plotted as functions of the normalized time delay $\tau = T_{del}/(T_{del} + T)$.

$$\begin{aligned} kK &= \min\left(0.4\frac{T}{L},\ 0.25\right) \\ k_iKT_{del} &= \max\left(0.1\frac{T}{T_{del}},\ 0.5\right). \end{aligned} \tag{8.16}$$

Notice that the improved formulas typically give lower controller gain than the Ziegler-Nichols method, but that integral gain is higher particularly for

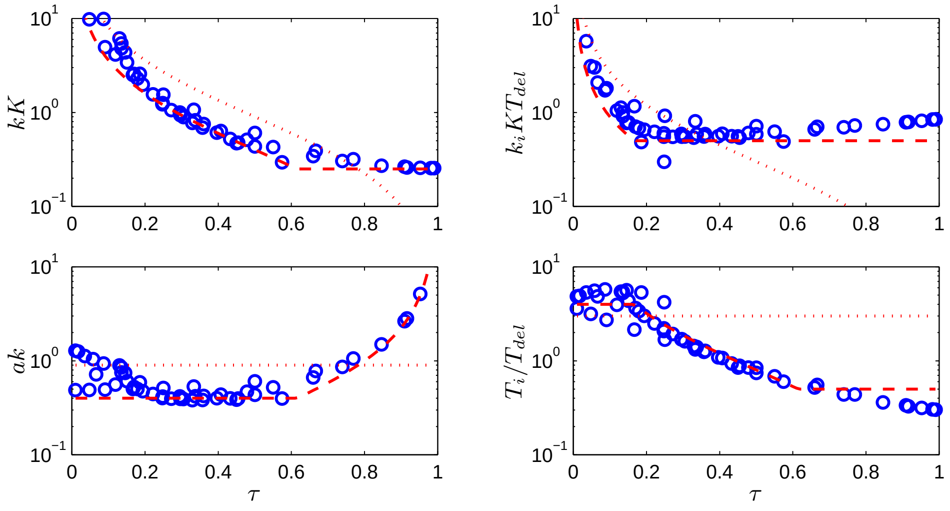

Figure 8.12: Proportional and integral gains for PI controllers given by the Ziegler-Nichols rule (dotted), the improved rule given by (**??**) (dashed) and controller designed for known transfer functions (∘).

systems with dynamics that is delay dominated. The figure also shows that significant improvements are obtained by characterizing dynamics by three parameters.

There are also improved tuning formulas for the frequency response method. In this case it is convenient to characterize the process by critical gain k_c, critical period T_c and static process gain K. One improved formula that is applicable for $k_cK > 0.2$ is

$$\begin{aligned} k &= 0.25k_c \\ k_i &= \frac{0.1k_c(1+4k_cK}{T_c} \end{aligned} \tag{8.17}$$

Relay Feedback

The experiment in Ziegler-Nichols frequency response method gives the frequency where the process has a phase lag of 180° and the gain of the process transfer function 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 8.13. This has been used to develop methods for automatic tuning of PID controllers. For many systems there will then be an oscillation, as

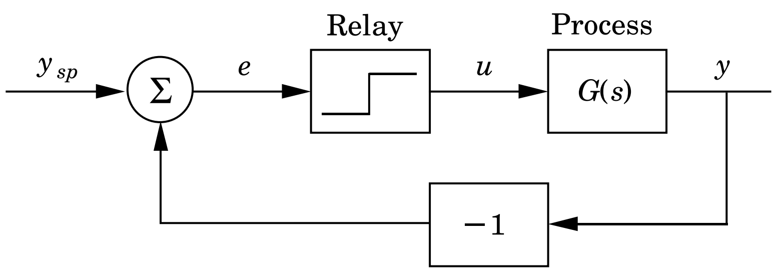

Figure 8.13: Block diagram of a process with relay feedback.

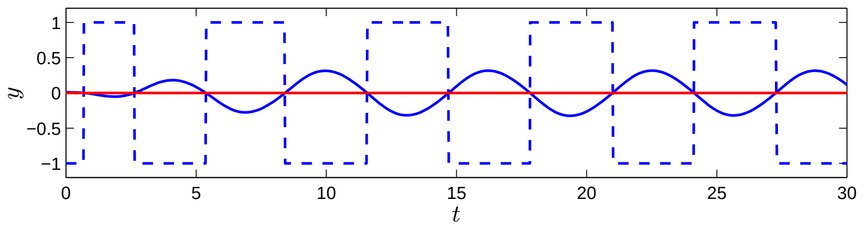

Figure 8.14: Process output y (solid) and relay output u (dashed) for a system under relay feedback. Notice that the signals are out of phase. The process has the transfer function $P(s) = (s + 1)^{-4}$.

shown in Figure 8.14, where the control signal is a square wave and the process output is close to a sinusoid. Notice that the process input and output have opposite phase and that a stable oscillation is established quickly.

To explain how the system works, 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 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

$$K_{180} = \frac{\pi a}{4d}. \tag{8.18}$$

Notice that 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. Notice in Figure 8.14 that a stable oscillation is

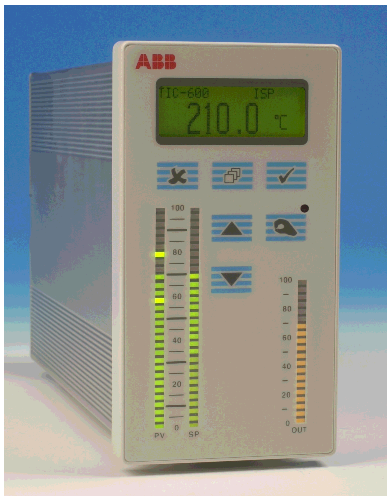

Figure 8.15: PID controller with automatic tuning.

established very quickly. The amplitude and the period can be determined after about 20 s only, 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 average residence time of the system is 12 s, which means that it would take about 40 s for a step response to reach steady state.

The idea of relay feedback has been used to implement PID controller with automatic tuning. An example of such a controller is shown in Figure 8.15. For this controller tuning is accomplished simply by pushing a button which activates relay feedback. The relay amplitude is adjusted automatically not to perturb the process too much and the controller automatically reverts to PID mode as soon as the tuning is accomplished.

8.5 Computer Control

In this section we will describe how a PID controller may be implemented using a digital computer. More material on implementation is given in Chapter 10. Most controllers are implemented in computers. The computer typically operates periodically, signals from the sensors are sampled and converted to digital form by the AD converter, the control signal is computed, converted to analog form for the actuators. The sequence of operation is as follows:

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

Notice that an analog output is done as soon as the output is available. The time delay is minimized by making the calculations in Step 3 as short as possible and to delay all updates until the analog output is commanded.

Discretization

As an illustration we consider the PID controller in Figure 8.7 which has a filtered derivative, set point 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

$$v(t) = P(t) + I(t) + D(t) \tag{8.19}$$

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

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

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

$$P(t_k) = k\left(\beta y_r(t_k) - y(t_k)\right) \tag{8.20}$$

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

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

Approximating the integral by a sum gives

$$I(t_{k+1}) = I(t_k) + k_i h\, e(t_k) + \frac{h}{T_t}\big(\mathrm{sat}(v) - v\big) \tag{8.21}$$

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

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

This can be rewritten as

$$D(t_k) = \frac{T_f}{T_f + h}\, D(t_{k-1}) - \frac{k_d}{T_f + h}\, \big(y(t_k) - y(t_{k-1})\big) \tag{8.22}$$

The advantage by 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.

Computer Code

Reorganizing Equations (8.19), (8.20), (8.21) and (8.22) 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
r=adin(ch1)                 "read setpoint from ch1
y=adin(ch2)                 "read process variable from ch2
P=K*(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
```

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 program must be called 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 derivation of the process output only, proportional action on part of the error only ($b \neq 1$), the last term in the updating of the integral gives protection against windup.

8.6 Further Reading

Ziegler-Nichols original paper. Some paper from industry (Honeywell) that describes industrial use. Bialkowsky?. A comprehensive presentation of PID control is given in [**?**]. Overviews of industrial use of adaptive control is found in [] and [].