Contents

1 PID Control

1.1 Introduction

The Proportional-Integral-Derivative (PID) controller is the most widely used control algorithm in industry. It governs the majority of feedback loops, either in its basic form or with minor variations. PID controllers are implemented in various ways, including as standalone devices, as part of Direct Digital Control (DDC) systems, or within hierarchical distributed process control systems.

This chapter provides an introduction to PID control. It details the fundamental algorithm and its different representations. An intuitive discussion of the controller's behavior in closed-loop systems is presented. The issue of integral windup, which arises when a controller with integral action is connected to a process with actuator saturation, is addressed, along with methods to prevent it.

Key aspects of implementing PID controllers in digital computers are discussed, including prefiltering, various digital approximation techniques, noise filtering, and best practices for coding. Operational considerations, such as smooth transitions (bumpless transfer) between manual and automatic modes and between different parameter sets, are also covered. The chapter concludes by examining the appropriate and inappropriate applications of PID control, highlighting examples where it is effective and where it may not be suitable.

1.2 The Principle of Feedback in Control Systems

Feedback is a fundamental concept in control systems engineering, enabling systems to maintain desired performance despite external disturbances and internal variations.

A simple feedback control loop is illustrated in Figure 1.1. The system comprises two primary components: the **process** and the **controller**. These components interact through a series of signals that establish a causal relationship between inputs and outputs.

Components of the Control Loop

- **Process:** The process represents the system or plant being controlled. It has one input, known as the *manipulated variable* or *control variable*, denoted by u. The output of the process is the *process variable* (PV), denoted by y, which is measured by a sensor.

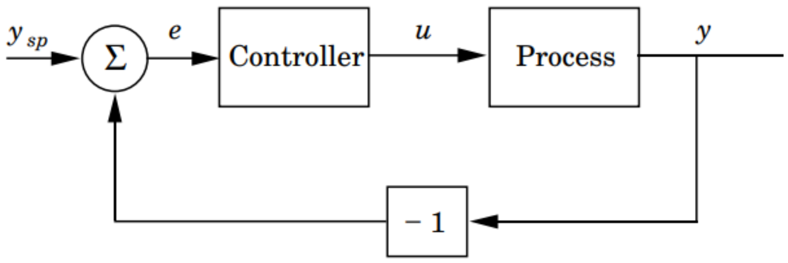

Figure 1.1: Block diagram of a process with a feedback controller.

- **Controller:** The controller receives the *control error* e as its input and produces the control variable u as its output. The control error is defined as the difference between the desired value of the process variable, known as the *setpoint* (SP) or *reference value*, denoted by y_{sp}, and the actual measured process variable:

$$e = y_{\text{sp}} - y$$

1.2.1 Closed Feedback Loop

In the depicted block diagram, the process and the controller are interconnected to form a *closed feedback loop.* This configuration allows the system to automatically adjust the control variable u based on the error e, thereby minimizing the discrepancy between the process variable y and the setpoint y_{sp}.

1.2.2 Operational Mechanism of Feedback

The primary objective of the feedback loop is to ensure that the process variable y remains as close as possible to the setpoint y_{sp}, even in the presence of disturbances. The operational mechanism can be described as follows:

1. **Equilibrium State:** Assume the system is initially in equilibrium, with the process variable y matching the setpoint y_{sp}.

2. **Disturbance Occurrence:** A disturbance d affects the process, causing the process variable y to deviate from the setpoint. For instance, y may increase above y_{sp}.

3. **Error Detection:** The sensor detects the change in y, and the controller computes the control error $e = y_{\text{sp}} - y$. In this case, e becomes negative since $y > y_{\text{sp}}$.

4. **Control Action:** Based on the negative error e, the controller adjusts the control variable u to counteract the disturbance. Specifically, the controller decreases u to reduce the process variable y.

5. **Restoration to Setpoint:** The reduction in u prompts the process to lower y back towards y_{sp}, thereby reducing the error e to zero.

This sequence of actions exemplifies *negative feedback*, where the controller's response acts in the opposite direction of the disturbance to restore equilibrium. Negative feedback is essential for stabilizing the system and ensuring accurate tracking of the setpoint.

1.2.3 Types of Feedback

Feedback can be categorized based on the direction of the controller's response relative to the process variable:

- **Negative Feedback:** As described earlier, negative feedback reduces the error by applying a control action opposite to the direction of the disturbance. It is the most commonly used feedback type due to its stabilizing effects.
- **Positive Feedback:** In contrast, positive feedback amplifies the error by applying a control action in the same direction as the disturbance. While less common, positive feedback can be useful in specific applications such as oscillators or in systems requiring amplification.

1.2.4 Mathematical Representation

The behavior of a feedback control system can be described mathematically using transfer functions or block diagram algebra. For a simple proportional controller, the relationship between the control variable u and the error e is given by:

$$u = K_p e$$

where K_p is the proportional gain. The closed-loop transfer function, which relates the setpoint y_{sp} to the process variable y, can be derived by analyzing the interconnections of the process and the controller.

1.2.5 Example: Temperature Control System

Consider a temperature control system where the goal is to maintain the temperature of a furnace at a desired setpoint y_{sp}. The process variable y is the actual temperature measured by a sensor. The controller adjusts the fuel flow rate u to the furnace based on the error $e = y_{\text{sp}} - y$.

1. **Setpoint Change:** If the setpoint y_{sp} is increased, the error e becomes positive if the current temperature y is below the new setpoint.
2. **Controller Response:** The controller increases the fuel flow rate u proportionally to the positive error e, raising the furnace temperature y towards the new setpoint.

3. **Disturbance Handling:** If an external disturbance, such as a cooler ambient temperature, causes the furnace temperature y to drop, the error e becomes positive, prompting the controller to increase u to compensate.

4. **Stabilization:** Once the temperature y returns to the setpoint y_{sp}, the error e approaches zero, and the controller reduces the fuel flow rate u accordingly.

This example demonstrates how a feedback control system dynamically adjusts the control variable to maintain the process variable at the desired setpoint, ensuring stable and accurate temperature regulation.

1.2.6 On-Off Control

Feedback can be implemented in various ways. A simple feedback mechanism is *on-off control*, mathematically described as:

$$u = \begin{cases} u_{\max} & \text{if } e > 0, \\ u_{\min} & \text{if } e < 0, \end{cases}$$

where $e = y_{\text{sp}} - y$ is the control error. This control law implies that maximum corrective action is always applied: the manipulated variable u is at its maximum $u_{\max}$ when the error is positive and at its minimum $u_{\min}$ when the error is negative. On-off control is straightforward with no parameters to adjust. While it often keeps the process variable near the setpoint, it typically results in oscillations.

Note that the control variable u is undefined when $e = 0$. To address this, modifications such as introducing a dead zone or hysteresis are commonly used (see Figure 1.2).

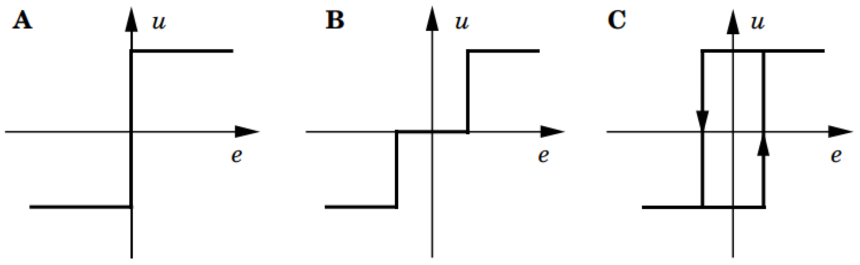

Figure 1.2: Controller characteristics for ideal on-off control (A), and modifications with dead zone (B) and hysteresis (C).

1.2.7 Proportional Control

The oscillations in on-off control often occur because the system overreacts—small changes in the error cause the manipulated variable to switch between its extreme values. This issue is mitigated in *proportional control*, where the controller's response is proportional to the control error for small errors. Figure 1.3 illustrates the characteristic of a proportional controller, represented by the nonlinear function $u = f_c(e)$.

To describe a proportional controller's characteristic, we specify the limits $u_{\max}$ and $u_{\min}$ of the control variable u. The linear range can be defined either by the slope of the characteristic (controller gain K) or by the range over which the characteristic is linear, known as the *proportional band* P_b. This range is typically centered around the setpoint. The relationship between the proportional band and the controller gain is given by:

$$u_{\max} - u_{\min} = KP_b.$$

Assuming $u_{\max} - u_{\min} = 100\%$, it follows that:

$$K = \frac{100}{P_b}.$$

For large errors, a proportional controller behaves like an on-off controller.

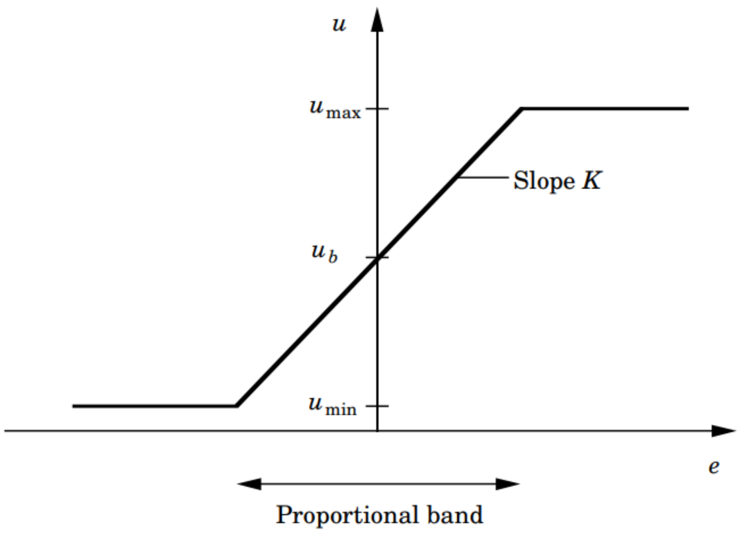

Figure 1.3: Characteristic of a proportional controller. The input is the control error e, and the output is the control signal u.

1.2.8 Static Analysis of Feedback Systems

Some properties of a control system can be understood through a simple static analysis. This involves introducing the *static process characteristic*, a curve that shows the steady-state value of the process output y as a function of the process input u (see Figure 1.4). This curve is meaningful only for stable processes.

The static process characteristic is crucial because it helps determine the range of control signals required to adjust the process output over the desired range, assists in sizing actuators, and aids in selecting appropriate sensor resolution. It can also help evaluate whether variations in static gain are significant enough to influence the control design.

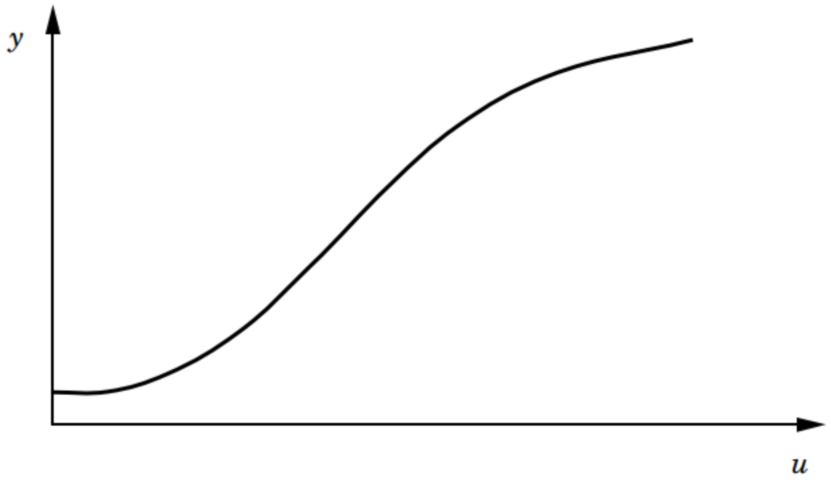

Figure 1.4: Static process characteristic showing process output y as a function of process input u under steady-state conditions.

1.2.9 Proportional Control Analysis

Consider a process under proportional control with the controller characteristic:

$$u = f_c(y_{\text{sp}} - y),$$

where y_{sp} is the setpoint and y is the process output. By introducing the inverse controller characteristic f_c^{-1}, we can rewrite this as:

$$y_{\text{sp}} - y = f_c^{-1}(u).$$

Using the static process characteristic $y = f_p(u)$, the equilibrium value of u satisfies:

$$y_{\text{sp}} - f_c^{-1}(u) = f_p(u).$$

This equation can be solved graphically by finding the intersection of the graphs of $f_p(u)$ and $y_{\text{sp}} - f_c^{-1}(u)$, as illustrated in Figure 1.5. If the static characteristics are monotonic, the intersection is unique. The equilibrium value of the process output y is the y-coordinate corresponding to this intersection. This graphical method makes it easy to see how the equilibrium is influenced by the setpoint and the controller gain.

The equilibrium matches the setpoint only if:

$$y_{\text{sp}} = y_0 \overset{\text{def}}{=} f_p(u_b),$$

where u_b is a bias value of u. For all other values of the setpoint, there will be a deviation.

If we approximate the process characteristic with a straight line of slope K_p and the controller gain is K, we can compute the deviation more easily. Introducing the parameter a as shown in Figure 1.5, we find:

$$y_{\text{sp}} - y_0 = \left(K_p + \frac{1}{K}\right) a,$$

$$y_{\text{sp}} - y = \frac{1}{K} a.$$

This leads to the steady-state error:

$$e = y_{\text{sp}} - y = \frac{1}{1 + K_p K}(y_{\text{sp}} - y_0).$$

Thus, the larger the loop gain $K_p K$, the smaller the steady-state deviation.

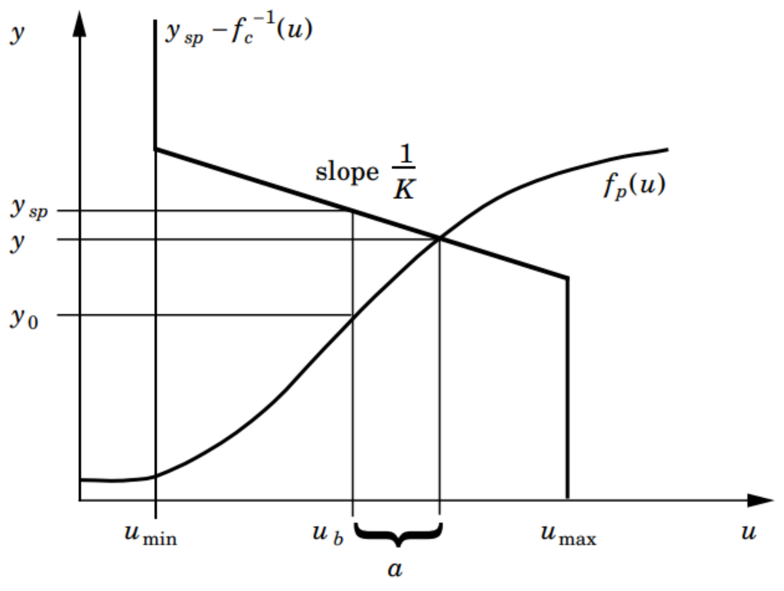

Figure 1.5: Determination of equilibrium from static process and controller characteristics.

1.3 PID Control

In the previous section, we observed that proportional control often results in a steady-state error. Therefore, practical control algorithms are typically more sophisticated than simple proportional controllers. Empirically, the *Proportional-Integral-Derivative (PID)* controller has been found to be highly effective. Within the proportional band, the behavior of the standard PID algorithm can be expressed as:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right], \tag{1.1}$$

where $u(t)$ is the control signal, $e(t) = y_{\text{sp}} - y(t)$ is the control error, K is the proportional gain, T_i is the integral time, and T_d is the derivative time. The control signal $u(t)$ comprises three components:

- **Proportional (P) term**: Proportional to the current error $e(t)$.
- **Integral (I) term**: Proportional to the integral of the error over time.
- **Derivative (D) term**: Proportional to the derivative (rate of change) of the error.

1.3.1 Proportional Action

For pure proportional control, the PID control law simplifies to:

$$u(t) = Ke(t) + u_b, \tag{1.2}$$

where u_b is a bias or reset value. When the control error is zero, the control signal equals the bias $u(t) = u_b$. The bias u_b is often set to the midpoint of the control signal range, $(u_{\max}+u_{\min})/2$, but can also be adjusted manually to eliminate steady-state error at a specific setpoint.

Static Analysis

Many characteristics of proportional control can be understood through static analysis. Consider the simple feedback loop depicted in Figure 1.6, consisting of a process and a controller.

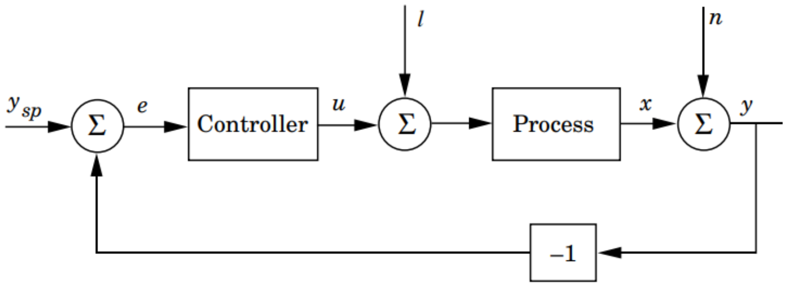

Figure 1.6: Block diagram of a simple feedback loop.

Assume the controller uses proportional action, and the process is modeled by the static relationship:

$$x = K_p(u + l), \tag{1.3}$$

where:

- x is the process variable.
- u is the control signal.
- l represents a load disturbance.
- K_p is the static process gain.

From the block diagram, we have:

$$y = x + n, \tag{1.4a}$$
$$x = K_p(u + l), \tag{1.4b}$$
$$u = K(y_{\text{sp}} - y) + u_b, \tag{1.4c}$$

where:

- y is the measured output.
- n is measurement noise.
- y_{sp} is the setpoint.
- K is the controller gain.

Eliminating intermediate variables yields the relationship between the process variable x, the setpoint y_{sp}, the load disturbance l, and the measurement noise n:

$$x = \frac{KK_p}{1 + KK_p}(y_{\text{sp}} - n) + \frac{K_p}{1 + KK_p}(l + u_b). \tag{1.5}$$

The product KK_p is a dimensionless quantity known as the *loop gain*. Several important properties of the closed-loop system can be inferred from Equation (1.5):

- **Setpoint Tracking**: With $n = 0$ and $u_b = 0$, a high loop gain KK_p ensures that the process output x closely follows the setpoint y_{sp}.
- **Disturbance Rejection**: A high loop gain reduces sensitivity to the load disturbance l.
- **Noise Sensitivity**: If measurement noise n is present, it affects the process output similarly to the setpoint. To prevent excessive sensitivity to noise, the loop gain should not be too large.
- **Bias Influence**: The controller bias u_b influences the system in the same way as a load disturbance.

Designing the loop gain involves a trade-off between different control objectives, and the optimal loop gain depends on which objectives are most critical.

Equation (1.5) also indicates that a steady-state error typically exists with proportional control. Intuitively, from Equation (1.4c), the control error $e(t)$ is zero only when $u = u_b$ at steady state. Thus, by appropriately choosing the controller bias u_b, the steady-state error can be eliminated at a specific operating point.

Note that the static analysis assumes the process can be described by a static model, which overlooks some dynamic properties of the closed-loop system. Importantly, considering process dynamics reveals that high loop gains can lead to instability. In practice, the maximum allowable loop gain is limited by the process dynamics.

1.3.2 Integral Action

The primary purpose of integral action is to eliminate the steady-state error between the process output and the setpoint. While proportional control typically leaves a residual steady-state error, integral action ensures that even a small error will continually adjust the control signal until the error is eliminated.

To demonstrate that the steady-state error is zero with integral action, assume the system reaches steady state with a constant control signal u_0 and a constant error e_0. According to the PID control law (Equation 1.1), the control signal is:

$$u_0 = K\left(e_0 + \frac{e_0}{T_i}t\right). \tag{1.6}$$

As long as $e_0 \neq 0$, the term $\frac{e_0}{T_i}t$ grows unbounded with time, contradicting the assumption that u_0 is constant. Therefore, in steady state, $e_0 = 0$.

Integral action can also be viewed as an automatic adjustment of the bias term u_b in a proportional controller. This concept is illustrated in the block diagram of Figure 1.7, where a proportional controller includes a reset mechanism.

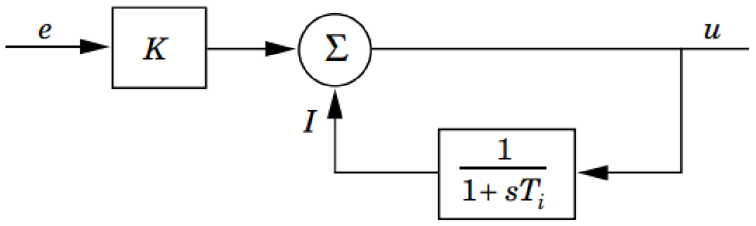

Figure 1.7: Implementation of integral action as positive feedback around a lag.

From the block diagram, we derive:

$$u = Ke + I, \tag{1.7a}$$

$$T_i \frac{dI}{dt} + I = u. \tag{1.7b}$$

Eliminating u between these equations yields:

$$T_i \frac{dI}{dt} = Ke, \tag{1.8}$$

which confirms that the controller functions as a PI controller.

1.3.3 Derivative Action

Derivative action aims to enhance the stability of the closed-loop system. The instability arises because process dynamics introduce a delay between a change in the control signal and its effect on the process output, causing the control system to lag in correcting errors. A controller with proportional and derivative action can be interpreted as making the control proportional to the predicted future error, obtained by extrapolating the current error using its tangent (see Figure 1.10).

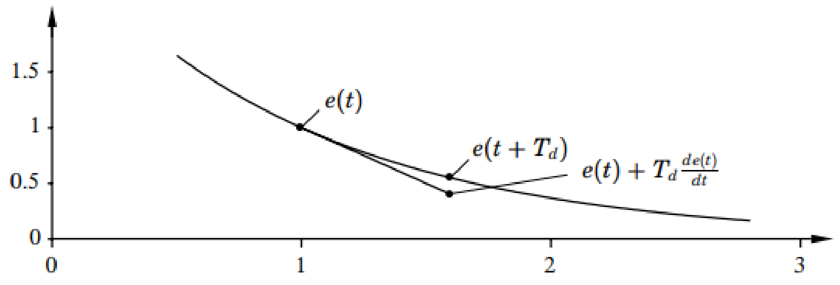

Figure 1.8: Interpretation of derivative action as predictive control, where the prediction is obtained by linear extrapolation.

The fundamental form of a PD controller is:

$$u(t) = K\left(e(t) + T_d \frac{de(t)}{dt}\right). \tag{1.9}$$

Using a Taylor series expansion:

$$e(t+T_d) \approx e(t) + T_d \frac{de(t)}{dt}, \tag{1.10}$$

the control signal becomes proportional to an estimate of the error T_d seconds into the future, obtained through linear extrapolation.

Summary

The PID controller consists of three components:

- **Proportional (P) term**: Control action proportional to the current error $e(t)$.
- **Integral (I) term**: Control action proportional to the integral of the error over time, ensuring zero steady-state error.
- **Derivative (D) term**: Control action proportional to the derivative of the error, allowing the controller to anticipate future errors based on current trends.

These components work together to improve control system performance. Various modifications of the basic PID algorithm can further enhance its effectiveness and usability, which will be discussed in the next section.

1.4 A Perspective on PID Control

While the basic PID equation (Equation 1.1) provides a foundational understanding, implementing it directly does not always yield an effective controller. To design a robust PID controller, several additional considerations are essential:

- **Noise Filtering and High-Frequency Roll-Off**: Reducing the impact of measurement noise and limiting the controller's response at high frequencies.
- **Set Point Weighting and Two Degrees of Freedom (2 DOF)**: Adjusting the controller's response to set point changes independently from disturbance rejection.
- **Windup Prevention**: Addressing the issue of integrator windup when actuators are saturated.
- **Tuning Methods**: Selecting appropriate parameters for the proportional, integral, and derivative terms.

Understanding and integrating these considerations into the PID controller design process leads to more robust and reliable control systems.

1.5 Filtering and Set Point Weighting

1.5.1 Filtering

Differentiation in control systems is highly sensitive to measurement noise. This sensitivity arises because the transfer function of an ideal differentiator, $G(s) = s$, increases without bound as frequency increases (i.e., as $s \to \infty$). To illustrate this phenomenon, consider the following example.

Example 1. Differentiation Amplifies High-Frequency Noise

Suppose we have a signal composed of a sinusoidal component and high-frequency noise:

$$y(t) = \sin t + n(t) = \sin t + a_n \sin(\omega_n t), \tag{1.11}$$

where $n(t)$ represents noise with amplitude a_n and angular frequency ω_n.

Taking the derivative of $y(t)$ yields:

$$\frac{dy(t)}{dt} = \cos t + n'(t) = \cos t + a_n\omega_n \cos(\omega_n t). \tag{1.12}$$

In this expression, the noise term is amplified by a factor of ω_n. The signal-to-noise ratio (SNR) of the original signal is $\frac{1}{a_n}$, whereas the SNR of the differentiated signal becomes $\frac{\omega_n}{a_n}$. As ω_n increases, the SNR deteriorates, meaning high-frequency noise is significantly amplified during differentiation.

To mitigate this issue in practical controllers employing derivative action, it is essential to limit the high-frequency gain of the derivative component. This can be achieved by implementing the derivative term with a first-order low-pass filter:

$$D(s) = -\frac{sKT_d}{1 + \dfrac{sT_d}{N}} Y(s), \tag{1.13}$$

instead of the ideal derivative term $D(s) = sKT_dY(s)$.

Equation (1.13) can be interpreted as filtering the ideal derivative with a first-order system having a time constant of $\dfrac{T_d}{N}$. This modification ensures that for low-frequency signals, the derivative action remains effective, while the high-frequency gain is limited to KN. Consequently, high-frequency measurement noise is amplified by at most a factor of KN. Typical values for N range between 8 and 20.

1.5.2 Further Limitation of High-Frequency Gain

In controllers employing derivative action with filtering, the transfer function from the measurement y to the controller output u is given by:

$$C(s) = -K\left(1 + \frac{1}{sT_i} + \frac{sT_d}{1 + \dfrac{sT_d}{N}}\right). \tag{1.14}$$

At high frequencies (as $s \to \infty$), the controller gain approaches a constant value:

$$\lim_{s\to\infty} C(s) = -K(1 + N). \tag{1.15}$$

This indicates that the gain does not diminish with increasing frequency, which can lead to issues with robustness against process variations and sensitivity to high-frequency noise. As discussed in earlier sections (see Section **??**), it is highly desirable to reduce the controller gain at high frequencies to improve robustness and reduce noise amplification.

Applying Additional Low-Pass Filtering

One effective method to limit the high-frequency gain is to apply additional low-pass filtering to the controller output. This can be implemented using a filter transfer function of the form:

$$F(s) = \frac{1}{(1 + sT_f)^n}, \tag{1.16}$$

where:

- T_f is the filter time constant.
- n is the order of the filter.

The choice of the filter time constant T_f is a trade-off between filtering effectiveness and control performance. A smaller T_f provides better noise attenuation but may introduce more phase lag, potentially destabilizing the system. To balance this, T_f can be related to the controller's time constants. For instance:

- If using derivative action, a suitable choice is $T_f = \dfrac{T_d}{N}$.
- For a PI controller without derivative action, setting $T_f = \dfrac{T_i}{N}$ may be appropriate.

Revised Controller Implementation

With the additional filtering, the controller can be reformulated as:

$$C(s) = -K\left(1 + \frac{1}{sT_i} + sT_d\right)\frac{1}{\left(1 + \dfrac{sT_d}{N}\right)^2}. \tag{1.17}$$

This structure offers several advantages:

1. **Design Flexibility**: It allows the use of design methods developed for the ideal PID controller.
2. **Iterative Design Procedure**: One can iteratively design the controller by first designing for the process $P(s)$ to obtain initial controller parameters, such as T_d.
3. **Performance-Filtering Trade-Off**: By considering the modified process $P_{\text{mod}}(s) = \dfrac{P(s)}{\left(1 + \dfrac{sT_d}{N}\right)^2}$, the designer can observe how filtering affects performance and adjust T_d accordingly.

Iterative Design Procedure

The iterative design approach involves the following steps:

1. **Initial Design**:
   - Design an ideal PID controller for the original process $P(s)$.
   - Determine initial values for the controller parameters, including the derivative time T_d.
2. **Modify Process Model**:
   - Adjust the process model to include the effect of the filter: $P_{\text{mod}}(s) = \dfrac{P(s)}{\left(1 + \dfrac{sT_d}{N}\right)^2}$.
3. **Redesign Controller**:
   - Redesign the PID controller using the modified process model $P_{\text{mod}}(s)$.
   - Update the controller parameters based on this design.
4. **Iteration**:
   - Repeat the modification and redesign steps until the controller performance meets the desired specifications.

1.5.3 Set Point Weighting

When utilizing the standard PID control law as given by Equation (1.21), a step change in the reference signal $r(t)$ can result in an impulse in the control signal $u(t)$, primarily due to the derivative action on the reference. This sudden spike is often undesirable in practical systems. To prevent this, derivative action is typically not applied to the reference signal directly.

An effective method to address this issue is through **set point weighting**, where the proportional and derivative actions act on modified versions of the reference signal. The PID controller equation becomes:

$$u(t) = K \left[b\, r(t) - y(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \left(c\, \frac{dr(t)}{dt} - \frac{dy(t)}{dt} \right) \right], \tag{1.18}$$

where:

- K is the proportional gain,
- T_i is the integral time constant,
- T_d is the derivative time constant,
- $y(t)$ is the process output,
- $e(t) = r(t) - y(t)$ is the tracking error,

- b and c are weighting parameters (typically between 0 and 1).

The integral term must rely solely on the error $e(t)$ to ensure correct steady-state behavior. The introduction of b and c creates a controller with **two degrees of freedom** (2-DOF), as the paths from $r(t)$ and $y(t)$ to $u(t)$ are different.

Transfer Functions

The transfer function from the reference input $r(t)$ to the control signal $u(t)$ is:

$$C_r(s) = K\left(b + \frac{1}{sT_i} + c\,sT_d\right). \tag{1.19}$$

The transfer function from the process output $y(t)$ to the control signal $u(t)$ is:

$$C_y(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right). \tag{1.20}$$

This shows that the controller responds differently to changes in $r(t)$ and $y(t)$, hence providing two degrees of freedom.

Impact on System Response

The choice of b and c affects how the system responds to set point changes, without altering its response to load disturbances or measurement noise. This is illustrated in Figure 1.9, which compares the system's output $y(t)$ and control signal $u(t)$ for different values of b.

In the simulation:

- The process transfer function is $P(s) = \dfrac{1}{(s+1)^3}$.
- The controller parameters are $K = 3$, $T_i = 2$, and $T_d = 1.5$.
- The set point weighting parameter c is set to zero to avoid large transients in $u(t)$ due to derivative action on the reference.

Figure 1.9 shows that:

- For $b = 0$ (dashed line), the proportional action does not act on the reference signal. This results in the smallest overshoot in the output $y(t)$ but a slower response.
- For $b = 0.5$ (solid line), the proportional action partially acts on the reference, providing a balance between overshoot and response speed.
- For $b = 1.0$ (dash-dotted line), the proportional action fully acts on the reference signal, leading to a faster response but with a larger overshoot.

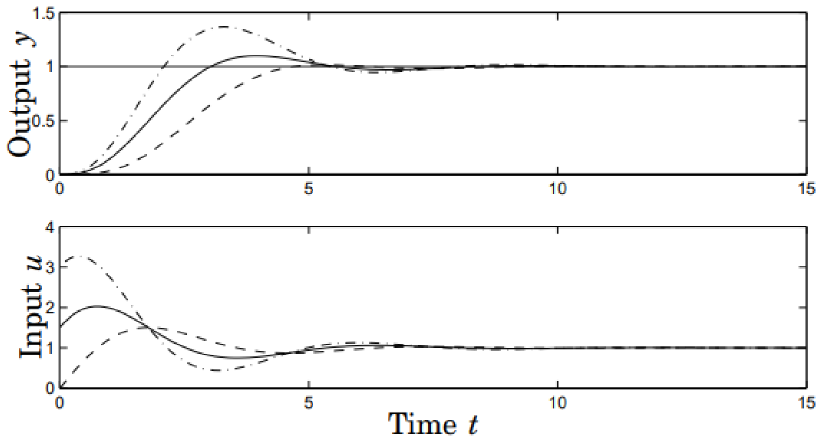

Figure 1.9: Response to a step change in the reference for systems with different set point weights: $b = 0$ (dashed line), $b = 0.5$ (solid line), and $b = 1.0$ (dash-dotted line). The process has the transfer function $P(s) = \frac{1}{(s+1)^3}$, and the controller parameters are $K = 3$, $T_i = 2$, and $T_d = 1.5$.

Choosing the Weighting Parameters

The parameter b allows tuning of the system's response to set point changes:

- $b = 0$: Minimizes overshoot, suitable for systems where overshoot is undesirable.
- $b = 1$: Maximizes responsiveness, suitable when a quick response is needed and overshoot is acceptable.
- $0 < b < 1$: Provides a trade-off between response speed and overshoot.

The parameter c is generally set to zero:

- Setting $c = 0$ avoids applying derivative action to the reference signal, preventing large spikes in the control signal $u(t)$ when the set point changes abruptly.
- Derivative action is then solely based on the process output $y(t)$, enhancing disturbance rejection without causing excessive control actions due to set point changes.

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right], \tag{1.21}$$

where $e(t) = r(t) - y(t)$.

1.6 PID Controller Variations

The standard form of the PID controller, as previously presented, can be expressed using the transfer function:

$$G(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right), \tag{1.22}$$

where:

- K is the proportional gain,
- T_i is the integral time constant,
- T_d is the derivative time constant,
- s is the Laplace variable.

This form is often referred to as the **non-interacting** or **parallel** PID controller.

1.6.1 Interacting (Series) Form

A slightly different version, commonly used in commercial controllers, is the **interacting** or **series** form. Its transfer function is given by:

$$G_{\mathrm{P}}(s) = K_{\mathrm{P}}\left(1 + \frac{1}{sT_i^{\mathrm{P}}}\right)\left(1 + sT_d^{\mathrm{P}}\right). \tag{1.23}$$

Expanding Equation (1.23) yields:

$$G_{\mathrm{P}}(s) = K_{\mathrm{P}}\left(1 + \frac{T_d^{\mathrm{P}}}{T_i^{\mathrm{P}}} + \frac{1}{sT_i^{\mathrm{P}}} + sT_d^{\mathrm{P}}\right). \tag{1.24}$$

Here:

- K_{P} is the proportional gain of the interacting controller,
- T_i^{P} is the integral time constant,
- T_d^{P} is the derivative time constant.

1.6.2 Conversion Between Forms

The interacting controller (Equation (1.23)) can be represented as a non-interacting controller (Equation (1.22)) with equivalent parameters:

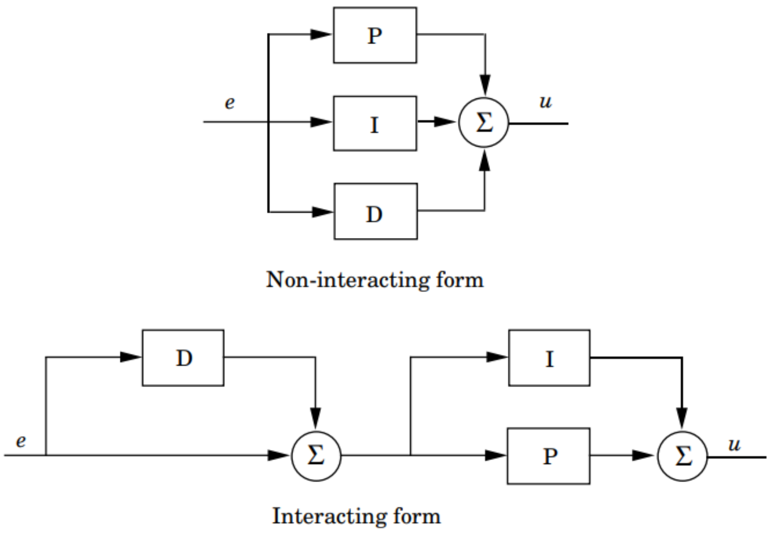

Figure 1.10: Interacting and non-interacting form of the PID algorithm.

$$K = K_{\mathrm{P}}\left(\frac{T_i^{\mathrm{P}} + T_d^{\mathrm{P}}}{T_i^{\mathrm{P}}}\right), \tag{1.25}$$

$$T_i = T_i^{\mathrm{P}} + T_d^{\mathrm{P}}, \tag{1.26}$$

$$T_d = \frac{T_i^{\mathrm{P}} T_d^{\mathrm{P}}}{T_i^{\mathrm{P}} + T_d^{\mathrm{P}}}. \tag{1.27}$$

Conversely, to find an interacting controller that corresponds to a given non-interacting controller, the following condition must be satisfied:

$$T_i \geq 4T_d. \tag{1.28}$$

If this condition holds, the parameters of the interacting controller can be calculated as:

$$K_{\mathrm{P}} = \frac{K}{2}\left(1 + \sqrt{1 - \frac{4T_d}{T_i}}\right), \tag{1.29}$$

$$T_i^{\mathrm{P}} = \frac{T_i}{2}\left(1 + \sqrt{1 - \frac{4T_d}{T_i}}\right), \tag{1.30}$$

$$T_d^{\mathrm{P}} = \frac{T_i}{2}\left(1 - \sqrt{1 - \frac{4T_d}{T_i}}\right). \tag{1.31}$$

1.6.3 Comparison of Controller Forms

The non-interacting controller (Equation (1.22)) is more general and will be the primary focus in subsequent discussions. However, the interacting controller is sometimes preferred for manual tuning due to its structure.

It's important to recognize that different controller forms may necessitate different parameter values. When replacing one controller type with another, parameter adjustments are often required to maintain equivalent performance.

Notably, the interacting and non-interacting forms yield the same behavior when only two of the three actions (P, I, D) are used—such as in P, PI, or PD controllers. Differences arise only when all three actions are present.

1.6.4 Alternative PID Representation

Another common representation of the PID controller is:

$$G_{\mathrm{PP}}(s) = k + \frac{k_i}{s} + sk_d, \tag{1.32}$$

where:

- k is the proportional gain,
- k_i is the integral gain,
- k_d is the derivative gain.

The relationships between this form and the standard non-interacting form are:

$$k = K, \tag{1.33}$$

$$k_i = \frac{K}{T_i}, \tag{1.34}$$

$$k_d = KT_d. \tag{1.35}$$

While mathematically equivalent to the standard form, the parameter values differ significantly. This can lead to confusion, especially if $1/k_i$ is mistakenly referred to as the integral time or if k_d is called the derivative time. Misnaming k_i as the integration time adds to the potential misunderstanding.

Despite these challenges, the form in Equation (1.32) is advantageous in analytical calculations because the parameters appear linearly. Additionally, it allows for the implementation of pure proportional, integral, or derivative action by assigning finite values to the respective parameters while setting the others to zero.

1.6.5 The PIPD Controller

The PID controller with set point weighting can be effectively represented using a PI-PD (Proportional-Integral-Proportional-Derivative) architecture, commonly referred to as the **PIPD Controller**. This subsection explores the structure, parameter relationships, and advantages of the PIPD controller.

Block Diagram Representation

Figure 1.11 illustrates the block diagram of a PI-PD controller. This configuration is equivalent to a conventional PID controller with set point weighting.

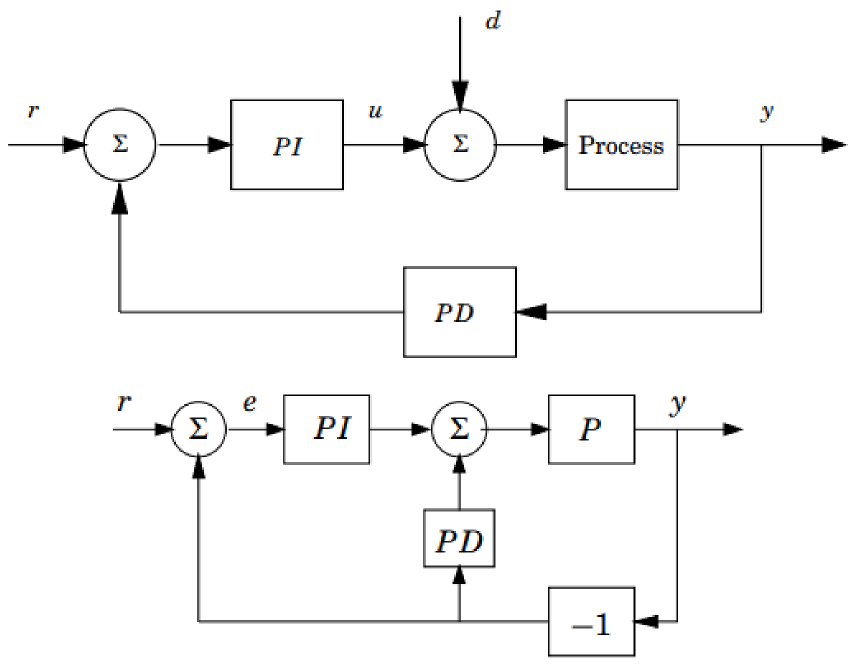

Figure 1.11: Block diagram of a PI-PD controller. This controller is equivalent to a conventional PID controller with set point weighting.

Transfer Functions of PI and PD Blocks

To understand the equivalence between the PI-PD controller and the PID controller with set point weighting, we introduce the transfer functions of the individual blocks:

$$G_{\mathrm{PI}}(s) = k_{\mathrm{P}} + \frac{k_{\mathrm{P}i}}{s}, \tag{1.36}$$

$$G_{\mathrm{PD}}(s) = 1 + k_{\mathrm{P}}ds. \tag{1.37}$$

- k_P is the proportional gain of the PI block.
- $k_{\mathrm{P}i}$ is the integral gain of the PI block.
- d is the derivative time constant of the PD block.

Input-Output Relationship

The input-output relationship of the complete PI-PD controller can be derived by combining the transfer functions of the PI and PD blocks. The controller output $U(s)$ in response to the reference input $R(s)$ and the process output $Y(s)$ is given by:

$$U(s) = k_\mathrm{P} R(s) + \frac{k_{\mathrm{P}i}}{s}(R(s) - Y(s)) - (k_\mathrm{P} + k_{\mathrm{P}i} d)\, Y(s) - k_\mathrm{P} d s Y(s). \tag{1.38}$$

This equation demonstrates that the PI-PD controller is identical to the PID controller with set point weighting as described in Equation (1.18). The parameters of the PIPD controller relate to the standard PID parameters as follows:

$$k = k_\mathrm{P} + \frac{k_{\mathrm{P}i} d k_\mathrm{P}}{k_{\mathrm{P}i}}, \tag{1.39}$$

$$T_i = \frac{k_\mathrm{P} + \frac{k_{\mathrm{P}i} d k_\mathrm{P}}{k_{\mathrm{P}i}}}{k_{\mathrm{P}i}} = k_\mathrm{P} + \frac{k_\mathrm{P} d}{k_{\mathrm{P}i}}, \tag{1.40}$$

$$T_d = \frac{k_\mathrm{P} k_{\mathrm{P}i} d}{k_{\mathrm{P}i}(k_\mathrm{P} + \frac{k_{\mathrm{P}i} d k_\mathrm{P}}{k_{\mathrm{P}i}})} = \frac{k_\mathrm{P} d}{k_\mathrm{P} + k_{\mathrm{P}i} d}. \tag{1.41}$$

Special Cases: I-PD and PI-D Controllers

Following the same parameterization pattern, specific configurations of the PI-PD controller can be defined based on the values of the weighting parameters b and c:

- **I-PD Controller**: Achieved by setting $b = 0$ and $c = 0$. In this configuration, the reference signal influences only the integral term, minimizing overshoot and reducing sensitivity to set point changes.
- **PI-D Controller**: Achieved by setting $b = 1$ and $c = 0$. Here, the proportional action fully acts on the reference signal, enhancing responsiveness but potentially increasing overshoot.

Advantages of the PIPD Controller

The PIPD controller offers several benefits over the conventional PID controller:

1. **Enhanced Tuning Flexibility**: The separation of PI and PD blocks allows for independent tuning of integral and derivative actions, facilitating more precise control.

2. **Improved Robustness**: By decoupling the set point weighting from the disturbance rejection, the controller can maintain robustness against load disturbances and measurement noise.

3. **Simplified Design Process**: The structure of the PIPD controller aligns with standard PID tuning methods, enabling the use of established design techniques for determining k, T_i, and T_d.

Complexity of Controller Parameters

While the PIPD controller structure offers enhanced flexibility, it introduces complexity in the relationship between the controller parameters k, T_i, T_d and the PI-PD parameters k_{P}, $k_{\mathrm{P}i}$, d. This complexity necessitates careful parameter management to ensure consistent and predictable controller behavior.

1.7 Handling Integrator Windup

Integrator windup is a phenomenon that occurs in PID control systems, particularly those with integral action, when the controller output exceeds its operational limits. This usually happens when there are large disturbances, setpoint changes, or system constraints, such as actuator saturation. When the output hits its maximum or minimum limits, the feedback loop is effectively broken, leading to an accumulation of error in the integrator component.

Since the integral term continues to accumulate error while the actuator remains at its limit, the control action can overshoot significantly once the actuator returns to its normal operating range. This "winding up" of the integral term can result in large transients, prolonged settling times, and reduced control accuracy, often causing the system to oscillate excessively as it tries to stabilize.

1.7.1 Illustration of integrator windup

The figure above provides an illustration of integrator windup in a control system with a PI controller managing an integrating process. This example demonstrates how integrator windup affects system behavior, particularly when there is a large setpoint change.

Initially, the setpoint increases sharply, creating a substantial error. In response, the control signal u rises to its maximum limit, causing the actuator to saturate at this high level. The integral term I, which accumulates over time to address persistent errors, continues to increase because the error remains positive. This leads to a steady buildup in the integral action.

At time $t = 10$, the error finally goes through zero, and the integral term I reaches its peak value. However, the system remains saturated due to the large accumulated value of I. The control signal u does not leave the saturation limit until the error stays negative long enough for the integral term to reduce. During this period, the control signal oscillates between its limits, bouncing back and forth several times. This

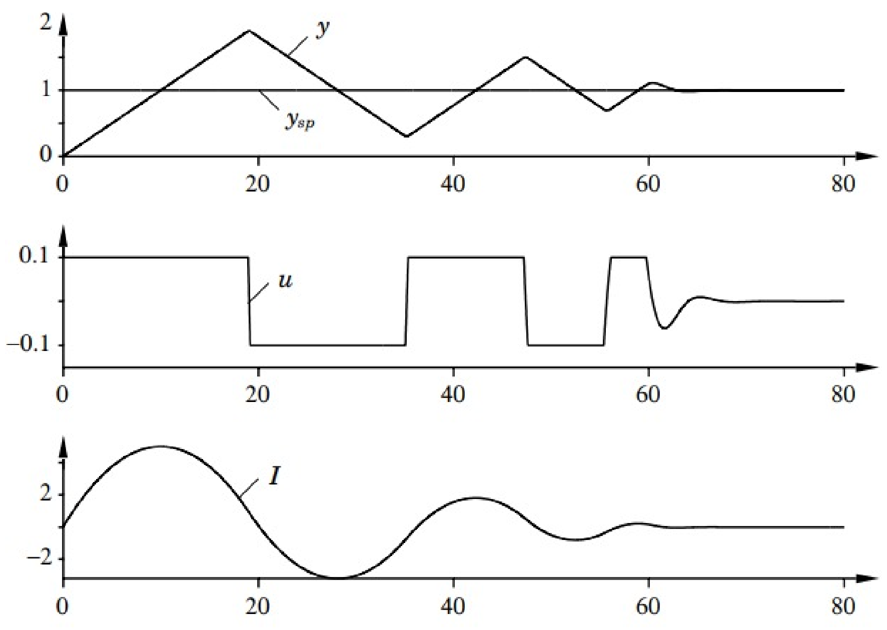

Figure 1.12: Illustration of Integrator Windup in a PID-Controlled Integrating Process.

oscillatory behavior, driven by the accumulated integral action, is similar to what is seen in relay oscillation.

As the system output y finally nears the setpoint y_{sp}, the actuator moves out of saturation, allowing the control system to regain stability. At this point, the system transitions to linear behavior and gradually settles near the setpoint.

This example highlights the effect of integrator windup: prolonged overshoot, oscillations, and delayed settling. These effects are common in systems with large setpoint changes, disturbances, or actuator limits, and they emphasize the importance of anti-windup mechanisms to manage the integral action effectively and ensure smooth system responses.

1.7.2 Problem of Integrator Windup

The primary issue with integrator windup is that the system accumulates error in the integral term when the actuator is saturated, causing the system to overshoot or oscillate after saturation ends. In cases where the control error is substantial, the integrator can reach large values, prolonging the time required to bring the system back to the desired setpoint. This effect is particularly noticeable in systems with significant time delays or nonlinearities.

Integrator windup is especially problematic in systems requiring fast responses to sudden changes, as it can compromise the stability and effectiveness of the control process. For instance, in systems with rapid setpoint changes or those subject to frequent disturbances, integrator windup can lead to performance degradation, excessive oscillations, and increased energy consumption.

1.7.3 Anti-Windup Mechanisms

To prevent integrator windup, several anti-windup mechanisms have been developed:

- **Integrator Clamping:** In this approach, the integrator is temporarily "clamped" or held at a certain value when saturation is detected, preventing further accumulation of error. This simple yet effective method helps prevent the integrator from building up excessive error during periods when the control output is constrained.
- **Back-Calculation Method:** This method adjusts the integrator value based on the difference between the controller's desired output and the actuator's actual output when saturation occurs. An additional feedback loop, often with a "tracking" time constant, allows the integrator to decay toward a value that matches the actuator limit. Back-calculation effectively resets the integrator in a controlled manner, allowing the system to quickly recover once saturation ends.
- **Incremental Algorithms:** Here, the controller is structured so that only incremental changes in control output are applied, reducing the likelihood of integrator windup. This method is effective when combined with actuators that directly implement control signals without the risk of saturation. The incremental approach can be implemented as a velocity algorithm, which minimizes the integrator's response time after the saturation period.
- **Tracking Mode in Digital Controllers:** Many digital controllers incorporate a tracking mode, where the integral term adjusts dynamically based on external conditions or a model of the actuator's behavior. By monitoring the difference between the control signal and the actuator's output, the integrator adjusts automatically to prevent windup. This mode is particularly effective in complex systems with varying operational limits.

Anti-windup techniques are critical in ensuring the stability and responsiveness of PID controllers, especially in systems with high saturation risks or strict performance requirements. By limiting integrator windup, these mechanisms improve control performance, reduce oscillations, and help maintain system stability.

1.8 Incremental Algorithms

In the early development of feedback control systems, integral action was integrated directly with the actuator. This integration was typically achieved by using a motor to drive the valve, ensuring automatic windup handling by ceasing integration when the valve reached its limit. As control systems transitioned from analog to digital implementations, manufacturers often retained analog-like configurations, leading to the emergence of velocity algorithms.

1.8.1 Back-Calculation and Tracking

With the advent of computer-based controllers, many manufacturers adopted velocity algorithms analogous to earlier mechanical designs. In a velocity algorithm, the rate of change of the control signal is calculated first and then fed into an integrator. Depending on the implementation, this integrator may be a motor directly connected to the actuator or an internal component within the controller itself. This approach prevents windup by inhibiting integration whenever the actuator output saturates, effectively implementing back-calculation, as detailed below.

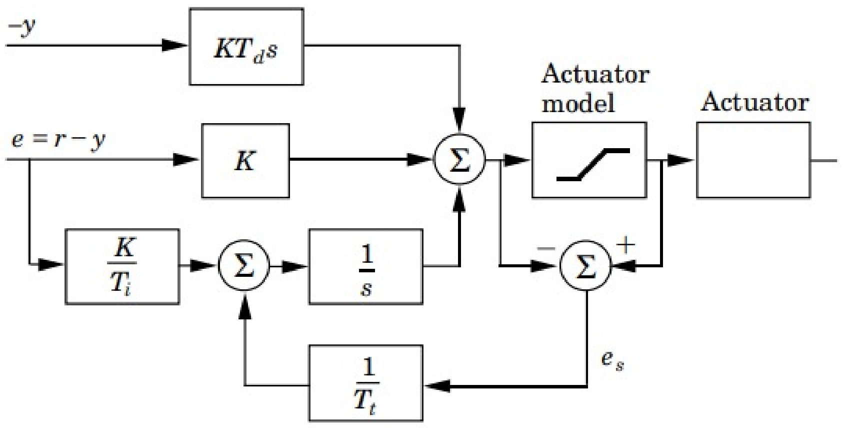

Figure 1.13: Controller with Anti-Windup Using Back-Calculation. The actuator output is estimated using a mathematical model.

When the actuator output cannot be directly measured, a mathematical model of the saturating actuator can estimate the saturated output, as shown in Figure 1.13. This estimation allows the controller to apply anti-windup measures effectively.

Back-Calculation Mechanism

The back-calculation mechanism adjusts the controller's integral term when the output saturates. Rather than resetting the integrator instantaneously, back-calculation dynamically adjusts it using a time constant T_t. This adjustment is controlled by the feedback gain $\frac{1}{T_t}$, which determines the rate at which the controller output resets.

Figure 1.14 illustrates anti-windup application in a PID controller. In this setup, an additional feedback path is established by measuring the actual actuator output and forming an error signal e_s as the difference between the controller output v and the actuator output u. This error signal is fed into the integrator through a gain of $\frac{1}{T_t}$.

Steady-State Behavior In steady-state conditions, where $e_s = 0$, the controller operates normally without affecting the integral term. When the actuator saturates, e_s

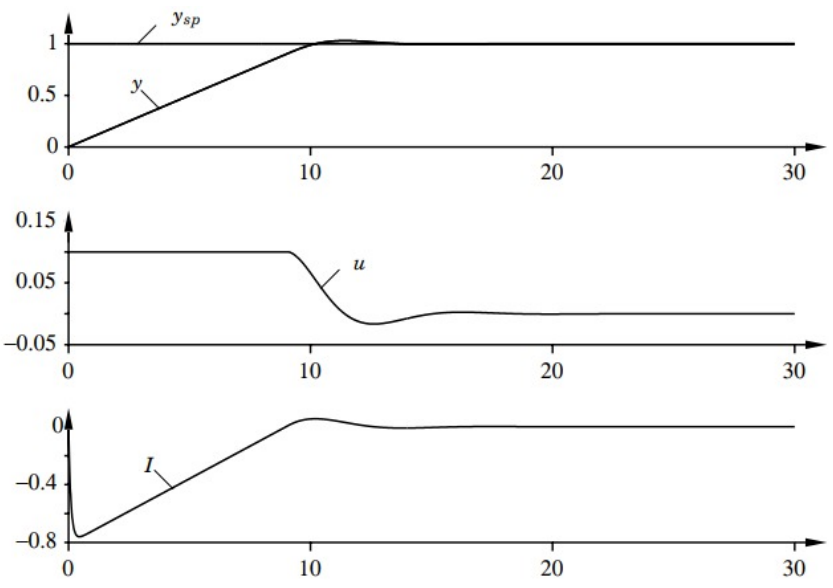

Figure 1.14: Controller with Anti-Windup Applied to the System. The diagrams show process output y, setpoint y_{sp}, control signal u, and integral part I.

becomes non-zero, causing the integrator to adjust dynamically. The integrator then drives towards a value that ensures the controller output v stays at the saturation limit u_{lim}. In steady-state, this relationship is defined as:

$$e_s = -\frac{KT_t}{T_i}e, w \tag{1.42}$$

where e is the control error $e = r - y$. This dynamic adjustment prevents the integrator from winding up by resetting it at a controlled rate determined by T_t, called the tracking time constant.

Figure 1.15 demonstrates the system's step response under different tracking time constants T_t. A smaller T_t allows the integrator to reset more quickly, preventing windup more effectively. However, an excessively small T_t can lead to undesired effects, such as spurious errors causing output saturation. A practical guideline is to select T_t within the range:

$$T_d < T_t < T_i, \tag{1.43}$$

with a commonly suggested value being $T_t = \sqrt{T_i T_d}$.

1.8.2 Controllers with a Tracking Mode

Controllers using back-calculation operate in two distinct modes:

1. **Normal Control Mode**: Functions as a standard PID controller, responding to control errors based on the setpoint and process output.

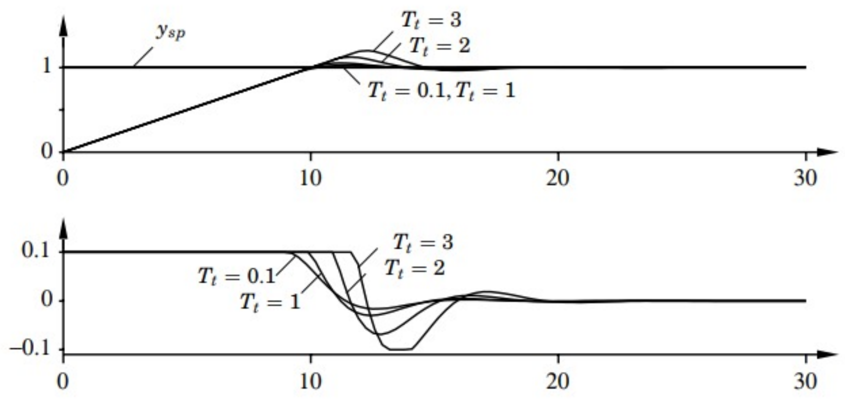

Figure 1.15: Step Response of the System for Different Tracking Time Constants T_t. The upper curve shows process output y and setpoint y_{sp}, while the lower curve displays the control signal u.

2. **Tracking Mode**: Activates when the controller needs to align with specific input and output signals, ensuring that the controller output matches a desired tracking signal.

Automatic Inhibition of Tracking The tracking mechanism is inherently self-regulating. Tracking is inhibited when the tracking signal w equals the controller output v. This automatic inhibition eliminates the need for explicit mode-switching logic, simplifying the integration of complex control systems, such as those involving selectors and cascade control configurations.

Figure **??** shows how the anti-windup controller from Figure 1.13 can be represented using the basic control module with tracking shown in Figure **??**. This representation highlights the versatility and adaptability of the tracking-based approach in managing controller behavior under different conditions.

Benefits of Tracking Mode Incorporating a tracking mode offers several advantages:

- **Enhanced Stability**: By automatically inhibiting tracking when not needed, the controller maintains stability without requiring additional control logic.
- **Simplified Integration**: Facilitates the combination of multiple control strategies, such as cascade control, by seamlessly managing different control modes.
- **Improved Performance**: Allows the controller to adjust its behavior dynamically based on the tracking signal, optimizing performance across various scenarios.

Practical Implementation Considerations When implementing controllers with tracking mode, consider several practical aspects:

- **Seamless Mode Transition**: The controller should transition smoothly between normal and tracking modes without introducing instability or unwanted transients, especially in systems with frequent setpoint changes.
- **Accuracy of the Tracking Signal**: The tracking signal w should accurately reflect the desired controller output during tracking mode. Misalignment between w and the controller output v can lead to unnecessary switching or oscillatory behavior.
- **Dynamic Adjustment of Tracking**: The controller's internal logic should effectively blend normal control actions with tracking behavior, particularly during transient events.
- **Avoiding Excessive Resets**: Rapid resetting of the integrator, especially with a small tracking time constant T_t, may inadvertently lead to integrator windup or saturation. A carefully selected T_t value within $T_d < T_t < T_i$ enables effective tracking without excessive resets.

Scenario: Temperature Control System

- **Objective:** Maintain furnace temperature at the setpoint y_{sp}.
- **Components:**
  - **Process Variable (PV):** Current temperature y.
  - **Control Signal (u):** Fuel flow rate.
  - **Setpoint (SP):** Desired temperature y_{sp}.
  - **Disturbance (d):** External cooler ambient temperature.

Step-by-Step Operation

1. **Setpoint Change:**
   - Increase y_{sp}.
   - Positive error $e = y_{\text{sp}} - y$ if $y < y_{\text{sp}}$.
2. **Controller Response:**

$$\Delta u(t) = K_p e(t) + \frac{K_p h}{T_i} e(t) + K_d \frac{de(t)}{dt}$$

3. **Control Signal Update:**

$$u(t) = u(t-h) + \Delta u(t)$$

4. **Disturbance Handling:**

- Cooler d causes y to drop.
- Error e increases, prompting controller to increase u.

5. **Stabilization:**
   - As y returns to y_{sp}, e approaches zero.
   - Controller reduces u accordingly.

Control Signal Dynamics

$$u(t) = K_p e(t) + \frac{K_p h}{T_i} \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

- **Proportional Term ($K_p e(t)$):** Responds to current error.
- **Integral Term ($\frac{K_p h}{T_i} \int e(\tau) d\tau$):** Accumulates past errors.
- **Derivative Term ($K_d \frac{de(t)}{dt}$):** Predicts future errors.

Practical Considerations

- **Tracking Time Constant (T_t):**
$$T_t = \sqrt{T_i T_d}$$
- **Guideline for T_t:**
$$T_d < T_t < T_i$$
- **Benefits of Proper T_t:**
  - Prevents integrator windup.
  - Balances responsiveness and stability.

Applications of Tracking Mode The tracking mode is particularly useful in complex control architectures, such as cascade control or systems with selector logic, where the controller must respond dynamically to multiple inputs. In such configurations, tracking allows the controller to smoothly integrate with auxiliary control loops or external signals, facilitating enhanced performance under varying operational demands.

Key applications include:

- **Cascade Control Systems**: In cascade systems, an inner loop controller can operate in tracking mode, following the output of an outer loop controller, enhancing system response by enabling quick adjustments to changes in the outer loop.

- **Override Control**: In override control, the tracking mode can prioritize specific objectives, such as safety limits, by adjusting the controller output based on predefined tracking signals.
- **Setpoint Scheduling**: In processes requiring gradual setpoint changes, tracking mode allows the controller to follow an externally provided trajectory, minimizing abrupt changes in the control signal.

Summary of Incremental Algorithms Incremental algorithms, particularly those involving back-calculation and tracking modes, are essential in modern PID control for managing integrator windup and enhancing controller flexibility. By implementing back-calculation, the controller avoids excessive accumulation in the integrator during saturation. Tracking mode further adds versatility by allowing the controller to follow external signals or match specific input-output relationships as needed. Together, these techniques enable PID controllers to maintain optimal performance across diverse applications and operational scenarios.

1.9 Tuning

All general control design methods can be applied to PID control. Additionally, several specialized methods, known as **tuning methods**, have been developed specifically for PID controllers. Regardless of the chosen method, it is crucial to consider key control elements such as load disturbances, sensor noise, process uncertainty, and reference signals.

Among the most well-known tuning methods are those developed by Ziegler and Nichols. These methods have significantly influenced PID control practices for over half a century. They are based on characterizing process dynamics with a few parameters and applying simple equations to determine controller parameters. It is noteworthy that despite their limited applicability, Ziegler-Nichols methods remain widely referenced due to their simplicity and suitability for educational purposes in basic control courses.

1.9.1 The Step Response Method

One of the tuning methods proposed by Ziegler and Nichols is the **Step Response Method**. This method relies on process information derived from the open-loop step response, typically obtained through a bump test. It can be seen as a traditional model-based approach, where a simple process model is assumed.

The step response is characterized by two primary parameters: a and L, as illustrated in Figure 1.16.

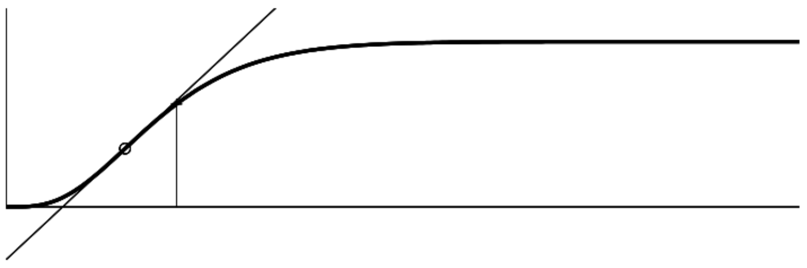

Figure 1.16: Characterization of a step response in the Ziegler-Nichols step response method.

The procedure involves identifying the point on the step response curve where the slope is maximal. A tangent is drawn at this point, and its intersections with the coordinate axes provide the parameters a and L.

Once a and L are determined, the PID controller parameters can be obtained using the guidelines provided in Table 1.1.
Additionally, an estimate of the closed-loop period T_p is provided in the table.

Procedure for the Step Response Method

The Step Response Method involves the following steps:

Table 1.1: Controller Parameters

Controller	K	T_i	T_d	T_p
P	$1/a$	-	-	$4L$
PI	$0.9/a$	$3L$	-	$5.7L$
PID	$1.2/a$	$2L$	$L/2$	$3.4L$

1. **Perform a Step Test**: Apply a step input to the open-loop system and record the process output response.
2. **Identify Parameters a and L**:
   - Determine the point on the step response where the slope is maximum.
   - Draw a tangent at this point.
   - Identify the intercepts of the tangent with the time axis (to find L) and the output axis (to find a).
3. **Determine Controller Parameters**: Use the values of a and L in Table 1.1 to obtain the PID controller parameters K, T_i, and T_d.
4. **Estimate Closed-Loop Period T_p**: Use the relationship provided in the table to estimate T_p for the closed-loop system.

1.9.2 The Frequency Response Method

Another pivotal tuning method developed by Ziegler and Nichols is the **Frequency Response Method**. This approach is grounded in a straightforward characterization of the process dynamics' frequency response. The design process relies on identifying a specific point on the Nyquist plot of the process transfer function $P(s)$, namely the point where the Nyquist curve intersects the negative real axis. This intersection point is characterized by two critical parameters: the frequency ω_{180} and the gain at that frequency $K_{180} = |P(i\omega_{180})|$.

Historically, this intersection point is referred to as the **ultimate point** and is characterized by the parameters:

- $K_u = \frac{1}{K_{180}}$ (Ultimate Gain)
- $T_u = \frac{2\pi}{\omega_{180}}$ (Ultimate Period)

Determining K_u and T_u

The parameters K_u and T_u can be determined through the following empirical procedure:

1. **Initial Controller Setup**: Connect a PID controller to the process and configure it to act purely as a proportional controller by setting the integral time $T_i = \infty$ and the derivative time $T_d = 0$.

2. **Gain Adjustment**: Gradually increase the proportional gain K until the system output begins to exhibit sustained oscillations.

3. **Parameter Identification**:
   - The gain at which the oscillations occur is recorded as the **ultimate gain** K_u.
   - The period of these oscillations is measured as the **ultimate period** T_u.

This method is empirical, relying on direct experimentation with the process to determine the necessary controller parameters.

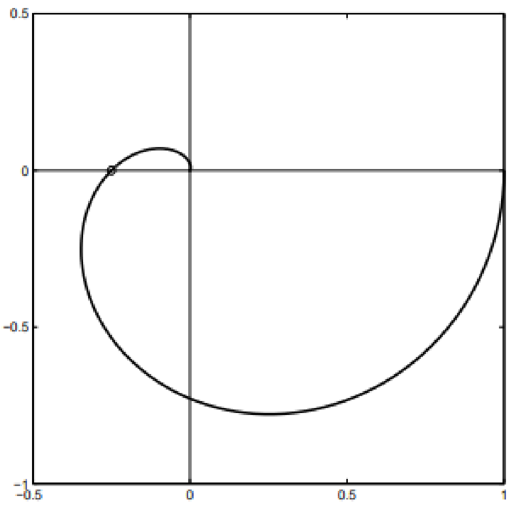

Figure 1.17: Characterization of a step response in the Ziegler-Nichols step response method

Controller Parameter Calculation

Once K_u and T_u are determined, the PID controller parameters can be set using the guidelines provided in Table 1.2.

Table 1.2: Controller Parameters

Controller	K	T_i	T_d	T_p
P	$0.5K_u$	-	-	T_u
PI	$0.4K_u$	$0.8T_u$	-	$1.4T_u$
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$0.85T_u$

Additionally, an estimate of the period T_p of the dominant dynamics of the closed-loop system is provided in the table.

Procedure for the Frequency Response Method

The Frequency Response Method involves the following steps:

1. **Setup Controller as Proportional Control**: Configure the PID controller to operate in proportional mode by setting $T_i = \infty$ and $T_d = 0$.
2. **Increment Gain Until Oscillation**: Slowly increase the proportional gain K until the system output begins to oscillate with a constant amplitude.
3. **Record Ultimate Gain and Period**:
   - The gain at which sustained oscillations occur is noted as K_u.
   - The period of these oscillations is measured as T_u.
4. **Determine PID Parameters**: Use the values of K_u and T_u in Table 1.2 to obtain the PID controller parameters K, T_i, and T_d.
5. **Estimate Closed-Loop Period**: Use the provided relationships to estimate T_p, the period of the dominant dynamics in the closed-loop system.

Example Application

Consider a process with an unknown transfer function $P(s)$. To tune a PID controller using the Frequency Response Method:

1. Configure the PID controller as a proportional controller ($T_i = \infty$, $T_d = 0$).
2. Incrementally increase K until sustained oscillations are observed.
3. Suppose the ultimate gain K_u is found to be 2, and the ultimate period T_u is 5 seconds.
4. Refer to Table 1.2 to set the PID parameters:
   - $K = 0.6 \times 2 = 1.2$
   - $T_i = 0.5 \times 5 = 2.5$ seconds
   - $T_d = 0.125 \times 5 = 0.625$ seconds
5. The estimated period of the dominant closed-loop dynamics is $T_p = 0.85 \times 5 = 4.25$ seconds.

1.9.3 Assessment of the Ziegler-Nichols Tuning Methods

The Ziegler-Nichols tuning rules were developed to achieve closed-loop systems with effective attenuation of load disturbances. Based on extensive simulations, these methods utilize a design criterion known as the **quarter amplitude decay ratio**, which aims to reduce the amplitude of oscillations by a factor of four per period. This corresponds to a closed-loop damping ratio of approximately $\zeta = 0.2$, which is relatively low and can lead to underdamped systems.

1.10 Digital Implementation of PID Controllers

1.10.1 Discretization

To implement a continuous-time control law, such as a PID controller, in a digital computer, it is essential to approximate the derivatives and the integral terms that are intrinsic to the control law. Various methods for this approximation are discussed below.

1.10.2 Proportional Action

The proportional term of the PID controller is defined as:

$$P = K(b\, y_{sp} - y)$$

where K is the proportional gain, y_{sp} is the setpoint, and y is the process variable. To implement this in a discrete system, we replace the continuous variables with their sampled versions, yielding:

$$P(t_k) = K\left(b\, y_{sp}(t_k) - y(t_k)\right)$$

where $\{t_k\}$ represents the sampling instants, i.e., the moments when the digital computer reads the analog input.

1.10.3 Integral Action

The integral term of the controller is expressed as:

$$I(t) = \frac{K}{T_i} \int_0^t e(s)\, ds$$

where T_i is the integral time constant, and $e(s) = y_{sp}(s) - y(s)$ is the error term. By differentiating this term, we obtain:

$$\frac{dI}{dt} = \frac{K}{T_i} e$$

To discretize this term, we use a forward difference approximation, yielding:

$$\frac{I(t_{k+1}) - I(t_k)}{h} = \frac{K}{T_i} e(t_k)$$

where h is the sampling period. This leads to the recursive equation for the integral term:

$$I(t_{k+1}) = I(t_k) + \frac{K\, h}{T_i} e(t_k)$$

1.10.4 Derivative Action

The derivative term, as described in Equation (6.2), is given by:

$$T_d N \frac{dD}{dt} + D = -K T_d \frac{dy}{dt}$$

where T_d is the derivative time constant, and N is a filter coefficient to smooth the derivative term. To discretize this term, we approximate the derivatives using a backward difference:

$$T_d N \frac{D(t_k) - D(t_{k-1})}{h} + D(t_k) = -K T_d \frac{y(t_k) - y(t_{k-1})}{h}$$

which can be rearranged as:

$$D(t_k) = \frac{T_d}{T_d + Nh} D(t_{k-1}) - \frac{K T_d N}{T_d + Nh} (y(t_k) - y(t_{k-1}))$$

This backward difference approximation ensures stability, as the parameter $\frac{T_d}{T_d+Nh}$ remains in the range $[0, 1]$ for all valid parameter values.

1.10.5 Summary of Discrete PID Controller Equations

In summary, the PID controller can be discretized as follows:

$$\begin{aligned}
p(t_k) &= K \left(b\, r(t_k) - y(t_k)\right) \\
e(t_k) &= r(t_k) - y(t_k) \\
d(t_k) &= \frac{T_d}{T_d + Nh} d(t_{k-1}) - \frac{KN}{T_d + Nh} (y(t_k) - y(t_{k-1})) \\
i(t_{k+1}) &= i(t_k) + \frac{Kh}{T_i} e(t_k) \\
u(t_k) &= p(t_k) + i(t_k) + d(t_k)
\end{aligned}$$

where $p(t_k)$, $i(t_k)$, and $d(t_k)$ represent the discrete proportional, integral, and derivative terms, respectively, and $u(t_k)$ is the control signal at sampling instant t_k.

1.11 Velocity Algorithms

The control algorithms discussed so far are known as *positional algorithms*, where the output is the absolute control signal. In some systems, however, the control signal directly drives an integrator, such as in a motor-driven system. For these cases, it is more natural for the control algorithm to generate the *velocity* or rate of change of the control variable rather than the control variable itself. By integrating this velocity, the control signal can be obtained. An algorithm designed in this way is referred to as a *velocity algorithm*. Figure 1.18 shows a block diagram of a PID controller using a velocity algorithm.

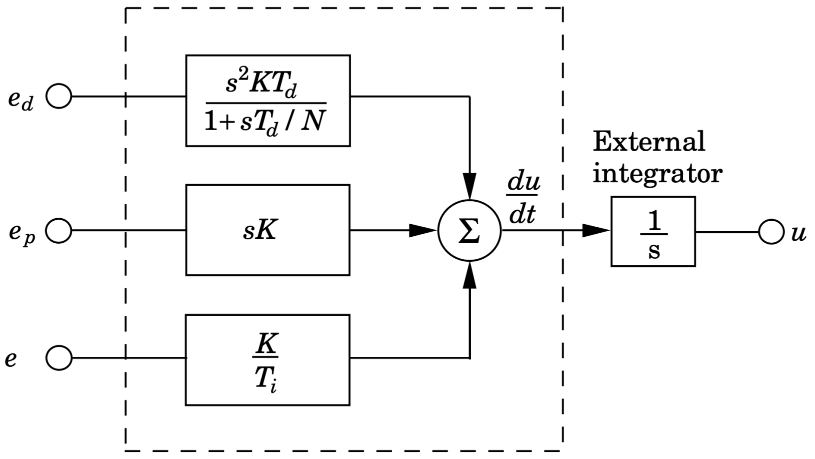

Figure 1.18: Block diagram of a PID algorithm in velocity form.

Velocity algorithms were popular in early controllers, particularly those employing motors. Even as technology evolved, this structure was often retained to ensure compatibility with legacy systems. Another advantage of velocity algorithms is that certain practical functions, such as *wind-up protection* and *bumpless parameter changes*, are simpler to implement, as discussed in Sections 6.5 and 6.7. When implemented digitally, velocity algorithms are also known as *incremental algorithms.*

1.11.1 Incremental Algorithm

The incremental form of the PID algorithm is derived by calculating the time difference of the controller output and expressing it as increments of the control signal:

$$\Delta u(t_k) = u(t_k) - u(t_{k-1}) = \Delta P(t_k) + \Delta I(t_k) + \Delta D(t_k)$$

In systems where integration is handled externally—such as when a stepper motor performs the integration—the controller output can directly represent the increments needed by the motor. The increments for each component (proportional, integral, and derivative) can be computed as follows, based on equations derived in previous sections:

$$\begin{aligned}
\Delta P(t_k) &= P(t_k) - P(t_{k-1}) = K\left(by_{sp}(t_k) - y(t_k) - by_{sp}(t_{k-1}) + y(t_{k-1})\right)\\
\Delta I(t_k) &= I(t_k) - I(t_{k-1}) = bi_1 e(t_k) + bi_2 e(t_{k-1})\\
\Delta D(t_k) &= D(t_k) - D(t_{k-1}) = a_d \Delta D(t_{k-1}) - b_d\left(y(t_k) - 2y(t_{k-1}) + y(t_{k-2})\right)
\end{aligned}$$

One of the primary advantages of using an incremental algorithm is computational efficiency, as most operations involve increments rather than absolute values. This allows for shorter word-length calculations, requiring higher precision only when summing the increments to form the control signal.

1.11.2 Velocity Algorithms in Controllers without Integral Action

A velocity algorithm cannot be directly applied to a controller without integral action, as such controllers cannot hold a steady output. This limitation is illustrated in Figure ??, where a proportional controller in velocity form fails to maintain stationarity because the output of the derivative block is zero for any constant input, meaning the control error e could vary without affecting the output.

To address this, a modification can be applied as shown in Figure ??. In this approach, stationarity is achieved when the control output u satisfies $u = Ke + u_b$, where u_b serves as an offset term.

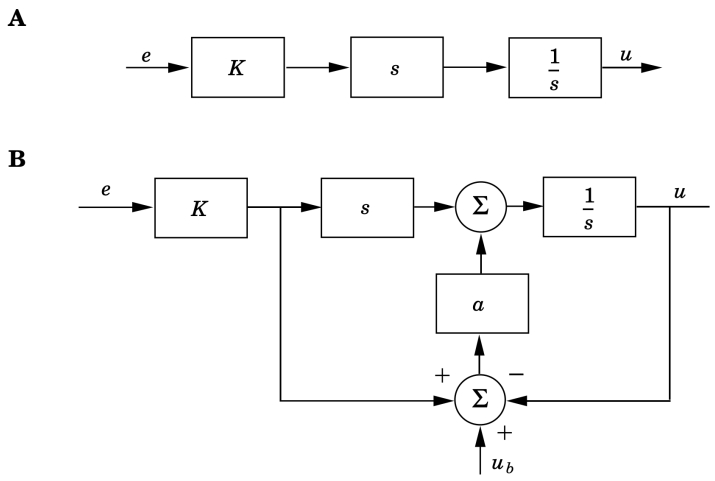

Figure 1.19: (A) A proportional controller in velocity form with no integral action, showing loss of stationarity. (B) Modified structure to restore stationarity.

For a sampled PID controller, a similar approach can be implemented by defining the proportional controller as:

$$\Delta u(t) = u(t) - u(t-h) = Ke(t) + u_b - u(t-h)$$

where h is the sampling period.

This structure ensures that the control system can retain a steady state by holding the last known control value. This is particularly useful in digital implementations where maintaining a stationary output is crucial.

1.12 Feedforward Control

In control systems, *feedforward control* offers a proactive way to enhance system performance by compensating for known disturbances or setpoint changes before they impact the output. Feedforward control works in tandem with traditional feedback control, creating a combined control signal expressed as:

$$u = u_{\mathrm{FB}} + u_{\mathrm{FF}}$$

where u_{FB} represents the feedback component, which adjusts based on system response errors, and u_{FF} represents the feedforward component, which anticipates and counteracts disturbances or setpoint changes.

1.12.1 Components of Feedforward Control

The feedforward term u_{FF} is calculated using information about the process or any measurable disturbances. This approach helps mitigate the impact of disturbances more quickly than feedback alone, as it acts before errors develop. The feedback term u_{FB}, on the other hand, compensates for any unpredicted deviations by continuously correcting based on the error between the desired setpoint and the actual output.

An effective feedforward system must carefully balance these two terms to avoid issues such as *integrator windup*, which occurs when the feedback controller's integrator accumulates excessive error. To prevent this, an *anti-windup mechanism* should operate on the total control signal u, rather than just the feedback component u_{FB}.

Figure 1.20: Block diagram illustrating feedforward and feedback control components with anti-windup applied to the total control signal.

1.12.2 Implementation Challenges and Solutions

In many modern control systems, anti-windup mechanisms are integrated within the feedback controller blocks. However, if the anti-windup feature acts only on the feedback control block, it cannot account for any feedforward signals added outside the block. This limitation often forces the feedforward signal to be injected *after* the feedback control block, which can lead to windup issues.

Various techniques have been developed to mitigate this windup problem, although they come with trade-offs. One common method is to pass the feedforward signal through a high-pass filter before adding it to the control signal. This approach reduces windup but may compromise the effectiveness of the feedforward action, especially for slow or constant disturbances.

1.12.3 Incremental Algorithms in Feedforward Control

Incremental algorithms are particularly effective for implementing feedforward control because they manage control changes in terms of increments, helping to avoid windup altogether. By calculating the incremental changes in both the feedback and feedforward terms separately and then summing them, we obtain:

$$\Delta u = \Delta u_{\mathrm{FB}} + \Delta u_{\mathrm{FF}}$$

The overall control signal can then be constructed incrementally as:

$$u(t) = u(t-h) + \Delta u(t)$$

where h is the sampling interval. This approach allows the system to update the control output smoothly and prevents windup by ensuring that both the feedback and feedforward contributions are managed within the same framework.

For incremental algorithms to be fully effective in feedforward control, it is essential that the feedback control blocks have inputs for feedforward signals. This setup ensures that the control signal can be updated as a cohesive whole, maintaining stability and responsiveness even in the presence of disturbances.

1.12.4 Summary

Feedforward control, when combined with feedback, improves system performance by counteracting disturbances in advance. However, special care must be taken to avoid integrator windup. By using incremental algorithms and designing control blocks to incorporate both feedback and feedforward components, feedforward control can be effectively implemented in digital systems. This approach ensures that the total control signal remains stable, allowing the feedforward component to operate without negatively impacting the feedback controller's performance.

1.13 Applications and Limitations of PID Control

PID controllers are widely adopted in industrial settings due to their simplicity and effectiveness in managing a variety of control systems. They excel in maintaining the process variable close to the desired setpoint by addressing key control requirements such as responsiveness to command signals, robustness against measurement noise and process variations, and effective rejection of load disturbances. Despite their broad applicability, PID controllers may not be suitable for all scenarios, particularly in systems with complex dynamics or significant delays.

1.13.1 When Is PI Control Sufficient?

In numerous industrial applications, the integral (I) and proportional (P) actions of a PID controller are adequate for achieving desired control performance. PI control is particularly effective in systems where the process dynamics are predominantly first-order.

To determine the suitability of PI control, analyze the system's step or frequency response. If the step response resembles that of a first-order system or if the Nyquist plot remains within the first and fourth quadrants, PI control is generally sufficient. Additionally, in processes designed to operate without stringent control precision, PI controllers can manage higher-order dynamics effectively by eliminating steady-state error through integral action and ensuring adequate transient response via proportional control.

1.13.2 When Is PID Control Sufficient?

PID control is ideal for systems where the dominant dynamics are second-order. In these cases, the derivative (D) action enhances the controller's ability to manage more complex behaviors without requiring a more sophisticated control strategy. Situations where PID control proves beneficial include:

- **Processes with Varying Time Constants:** Systems characterized by multiple time constants benefit from derivative action, which accelerates response and improves stability.
- **Temperature Control Systems:** These often exhibit second-order dynamics where derivative action helps dampen oscillations and achieve faster settling times.
- **Higher-Order Systems Requiring Tight Control:** In processes demanding precise regulation, derivative action provides additional damping, allowing for higher proportional gains and faster transient responses without inducing instability.

The derivative term acts as a predictive mechanism, anticipating future errors based on the current rate of change, thereby enhancing the overall stability and performance of the control system.

1.13.3 When PID Control May Not Be Adequate

While PID controllers are versatile, they are not universally applicable. Certain system characteristics can limit their effectiveness:

- **Higher-Order Processes:** Systems with dynamics exceeding second-order may exhibit complex behaviors that PID controllers cannot adequately manage, leading to suboptimal performance or instability.
- **Systems with Long Dead Time:** Processes that experience significant delays between control actions and observable effects on the process variable can cause the controller to react too late, resulting in oscillations or instability.
- **High-Frequency Noise Sensitivity:** The derivative term can amplify measurement noise, necessitating careful tuning and possibly additional noise filtering.

In such cases, more sophisticated control methodologies, such as Model Predictive Control (MPC) or adaptive control strategies, may be necessary to achieve the desired performance and stability.

1.14 Conclusion

PID control remains a foundational technique in control systems engineering due to its simplicity and effectiveness. Despite its limitations, enhancements and proper tuning allow PID controllers to meet the demands of various applications. Future trends include integrating machine learning for adaptive control and improving robustness in complex systems

Summary

This chapter provides a comprehensive exploration of **PID (Proportional-Integral-Derivative) Control** systems, which are fundamental to the field of control engineering. It begins by introducing the essential **principle of feedback** in control systems, explaining how feedback mechanisms are critical for maintaining system stability and achieving desired performance. The discussion includes an analysis of closed feedback loops, the operational mechanisms underlying feedback, and the various types of feedback utilized in different control scenarios. A practical example of a temperature control system is used to illustrate these concepts, alongside explanations of simpler control methods such as on-off and proportional control. The chapter further delves into the static analysis of feedback systems and provides an in-depth examination of proportional control analysis.

Building on this foundation, the chapter thoroughly examines the components of PID control. It breaks down the **Proportional Action**, which addresses the current error; the **Integral Action**, which accumulates past errors to eliminate steady-state error; and the **Derivative Action**, which predicts future errors based on the rate of change. This detailed analysis is complemented by a discussion on the broader perspective of PID control, including techniques like filtering and set point weighting that enhance controller performance. Various **PID Controller Variations** are explored, such as the interacting (series) form, methods for converting between different controller forms, and alternative representations like the PIPD controller.

A significant portion of the chapter is dedicated to addressing the challenge of **integrator windup**, a common issue in PID controllers where the integral term accumulates excessively, leading to performance degradation. The chapter illustrates the phenomenon, discusses the problems it causes, and presents effective anti-windup mechanisms to mitigate its effects. Additionally, the chapter covers **incremental algorithms**, including back-calculation and tracking methods, as well as controllers with a tracking mode, which enhance the robustness and adaptability of PID controllers.

The **tuning** of PID controllers is another critical topic addressed, with methodologies such as the step response method and the frequency response method being discussed in detail. The chapter also assesses the Ziegler-Nichols tuning methods, evaluating their effectiveness and applicability in various control scenarios. With the increasing prevalence of digital control systems, the chapter explores the **digital implementation of PID controllers**, covering discretization techniques and the implementation of proportional, integral, and derivative actions in discrete form. A summary of discrete PID controller equations is provided to facilitate practical applications.

Further, the chapter examines **velocity algorithms**, including incremental algorithms and velocity algorithms in controllers without integral action, which offer alternative approaches to PID control. The concept of **feedforward control** is introduced, detailing its components, implementation challenges, and solutions. The role of incremental algorithms in feedforward control is also discussed, highlighting their contribution to improved controller performance.

Finally, the chapter analyzes the **applications and limitations of PID control**, outlining scenarios where PI control is sufficient, where full PID control is necessary, and instances where PID control may not be adequate. This comprehensive overview underscores the versatility and foundational importance of PID control in various engineering applications while also acknowledging its limitations and the contexts in which alternative control strategies might be more appropriate. The chapter concludes by synthesizing the key insights, emphasizing the enduring relevance of PID controllers in modern control systems.

Bibliography

[1] Nagrath, I. J., & Gopal, M. (2017). *Control System Engineering.* PHI Learning Pvt. Ltd. Available: https://www.amazon.in/NAGRATH-J_CONTROL-SYSTEMS-ENG-699/dp/8195175589

[2] Nise, N. S. (2015). *Control Systems Engineering* (7th ed.). Wiley. Available: https://www.wiley.com/en-us/Control+Systems+Engineering%2C+7th+Edition-p-9781119361247

[3] Åström, K. J., & Murray, R. M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers.* Princeton University Press. Available: https://www.amazon.in/PID-Controllers-Theory-Design-Tuning/dp/1556175167

[4] Gopal, M. (2012). *Control Systems: Principles and Design* (4th ed.). Pearson Education. Available: https://www.pearson.com/store/p/control-systems-principles-and-design/P200000003135/9780131420340

[5] Wikipedia contributors. (n.d.). *PID controller. Wikipedia, The Free Encyclopedia.* Available: https://en.wikipedia.org/wiki/PID_controller

[6] Author(s). (2021). *A review of PID control, tuning methods and applications. Journal Name*, 9, 818–827. Available: https://link.springer.com/article/10.1007/s40435-020-00665-4

[7] National Instruments. (2024). *The PID Controller & Theory Explained.* Available: https://www.ni.com/en/shop/labview/pid-theory-explained.html

Problems on PID Controllers

Problem 1: Understanding the PID Controller Equation

The Proportional-Integral-Derivative (PID) controller is described by the following control law:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right] \tag{1.44}$$

where $u(t)$ is the control signal, $e(t)$ is the control error defined as $e(t) = y_{\mathrm{sp}}(t) - y(t)$, K is the proportional gain, T_i is the integral time constant, and T_d is the derivative time constant.

(a) Explain the role of each term in the PID controller equation and how they affect the control signal $u(t)$.

(b) Describe how each term contributes to the system response when there is a sudden change in the setpoint $y_{\mathrm{sp}}(t)$.

Problem 2: Transfer Function of PID Controller

Consider the PID controller described in Problem 1.

(a) Derive the transfer function $G_c(s)$ of the PID controller in the Laplace domain.

(b) Discuss how the proportional gain K, integral time T_i, and derivative time T_d influence the poles and zeros of the controller's transfer function.

Problem 3: Stability Analysis with Proportional Control

A process can be modeled by the first-order transfer function:

$$G_p(s) = \frac{1}{s+1} \tag{1.45}$$

A proportional controller with gain K_p is used in a unity feedback configuration.

(a) Derive the closed-loop transfer function of the system.

(b) Determine the range of values of K_p for which the closed-loop system is stable.

(c) If $K_p = 10$, calculate the steady-state error for a unit step input and explain why a steady-state error exists.

Problem 4: Ziegler-Nichols Step Response Method

A process has the following step response to a unit step input:
- The output reaches 28.3% of its final value at $t = 2$ seconds. - The output reaches 63.2% of its final value at $t = 5$ seconds.

Assuming that the process can be approximated by a first-order plus dead time (FOPDT) model, use the Ziegler-Nichols step response method to determine suitable PID controller parameters.

Problem 5: Ziegler-Nichols Frequency Response Method

An open-loop stable process is controlled using a proportional controller. When the proportional gain K is increased to $K_u = 15$, the closed-loop system exhibits sustained oscillations with a period of $T_u = 4$ seconds.

(a) Confirm that $K_u = 15$ is the ultimate gain and $T_u = 4$ seconds is the ultimate period.

(b) Use the Ziegler-Nichols frequency response method to calculate the PID controller parameters K, T_i, and T_d.

Problem 6: Anti-Windup Mechanism Design

Consider a PID controller implemented in a system where the actuator has saturation limits of $u_{\min} = -10$ and $u_{\max} = 10$.

(a) Explain what is meant by integrator windup and why it can be problematic in this system.

(b) Design an anti-windup mechanism using the back-calculation method. Derive the equations needed to implement it, including the selection of the tracking time constant T_t.

Problem 7: Digital Implementation of PID Controller

A PID controller is to be implemented in a digital control system with a sampling period of $h = 0.1$ seconds.

(a) Derive the discrete-time equations for the proportional, integral, and derivative terms using the backward difference approximation for the derivative and a rectangular approximation for the integral.

(b) Provide a step-by-step algorithm for implementing the PID controller in discrete time, including initialization and update equations.

Problem 8: Analysis and Design of a Temperature Control System

A temperature control system can be modeled by the transfer function:

$$G_p(s) = \frac{K_p}{(s+a)(s+b)} \tag{1.46}$$

where $K_p = 2$, $a = 1$, and $b = 0.5$.

(a) Analyze the open-loop response of the system to a unit step input. Determine the time constants and steady-state gain.

(b) Design a PID controller to achieve a desired closed-loop time constant of $T_c = 0.5$ seconds. Assume the controller can be approximated by:

$$G_c(s) = K\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{1.47}$$

Determine suitable values for K, T_i, and T_d.

(c) Simulate the closed-loop system response to a unit step change in the setpoint using the designed PID controller. Comment on the performance in terms of rise time, overshoot, and settling time.

Solution to Problem 1

(a) Explanation of Each Term in the PID Controller Equation

The PID controller is defined by the control law:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right] \tag{1.48}$$

where:

- $u(t)$ is the control signal applied to the process.
- $e(t) = y_{\text{sp}}(t) - y(t)$ is the control error, the difference between the setpoint $y_{\text{sp}}(t)$ and the process variable $y(t)$.
- K is the proportional gain.
- T_i is the integral time constant.
- T_d is the derivative time constant.

We will explain the role of each term in the PID controller and how they affect the control signal $u(t)$.

Proportional Term: $Ke(t)$ The proportional term is:

$$u_P(t) = Ke(t) \tag{1.49}$$

Role: The proportional term generates a control signal that is directly proportional to the current error $e(t)$. It provides an immediate response to the error.

Effect on $u(t)$: If the error increases, the proportional term increases the control signal proportionally. It acts to reduce the magnitude of the error by adjusting the control signal in the opposite direction.

Impact: The proportional term helps to reduce the rise time and reduces but does not eliminate the steady-state error. A higher proportional gain K leads to a larger control action for a given error, which can speed up the system response but may cause overshoot and potential instability if too large.

Integral Term: $K\frac{1}{T_i}\int_0^t e(\tau)\,d\tau$ The integral term is:

$$u_I(t) = K\frac{1}{T_i}\int_0^t e(\tau)\,d\tau \tag{1.50}$$

Role: The integral term accumulates the error over time. It considers the history of the error, summing up past errors to eliminate steady-state offset that the proportional control alone cannot remove.

Effect on $u(t)$: The integral term increases the control signal when the error has persisted over time. It adjusts the control action to eliminate any residual steady-state error by integrating the error until it reaches zero.

Impact: The integral action eliminates steady-state error, improving the accuracy of the system. However, too much integral action can lead to slow system response and can cause overshoot or oscillations due to the accumulated error, known as integrator windup.

Derivative Term: $KT_d\frac{de(t)}{dt}$ The derivative term is:

$$u_D(t) = KT_d\frac{de(t)}{dt} \tag{1.51}$$

Role: The derivative term predicts the future trend of the error by calculating its rate of change. It provides a control action proportional to the rate at which the error is changing.

Effect on $u(t)$: If the error is changing rapidly, the derivative term contributes a significant control action to counteract this change. It responds to the slope of the error curve, dampening the system response.

Impact: The derivative action improves the stability of the system by reducing overshoot and dampening oscillations. It acts as a brake, slowing down the controller output when the error is changing quickly. However, derivative action can amplify high-frequency noise in the error signal.

Combined Effect on $u(t)$ The total control signal is the sum of the three terms:

$$u(t) = u_P(t) + u_I(t) + u_D(t) \tag{1.52}$$

Each term contributes differently:

- The proportional term responds to the present error.
- The integral term responds to the accumulation of past errors.
- The derivative term anticipates future errors based on the current rate of change.

Together, they provide a control action that considers past, present, and future error behavior, aiming to improve both the transient and steady-state performance of the system.

(b) Contribution of Each Term to System Response During a Sudden Setpoint Change

When there is a sudden change in the setpoint $y_{\text{sp}}(t)$, the error $e(t)$ experiences an abrupt change. We will analyze how each term in the PID controller contributes to the system response in this scenario.

Proportional Term **Response to Sudden Setpoint Change:** The proportional term immediately responds to the change in error. Since the error increases abruptly, the proportional term produces a large immediate change in the control signal proportional to the magnitude of the error.

Contribution: It provides the initial corrective action to start driving the process variable toward the new setpoint. The magnitude of the response is determined by the proportional gain K.

Integral Term **Response to Sudden Setpoint Change:** The integral term begins to accumulate the error over time. Initially, its contribution is small because it depends on the integral of the error.

Contribution: As time progresses, the integral term increases, contributing more to the control signal. It ensures that any steady-state error resulting from the proportional control is eliminated. The integral action accelerates the system toward the setpoint over time.

Derivative Term **Response to Sudden Setpoint Change:** The derivative term reacts to the rate of change of the error. With a sudden setpoint change, the error changes rapidly, leading to a large derivative.

Contribution: The derivative term produces a significant control action that opposes the rapid change in error. It acts to dampen the system response, reducing overshoot and preventing excessive oscillations.

Overall System Response **Combined Effect:** The combined action of the three terms results in a balanced system response:

- *Speed of Response:* The proportional term provides a quick response to reduce the error.
- *Elimination of Steady-State Error:* The integral term ensures that any residual error is eliminated over time.
- *Stability and Overshoot Reduction:* The derivative term prevents the system from responding too aggressively, reducing overshoot and enhancing stability.

Practical Considerations:
In practice, sudden changes in the setpoint can cause large spikes in the control signal due to the derivative term. To mitigate this, derivative action is often applied only to the process variable $y(t)$ rather than the error $e(t)$. This modification prevents the derivative term from reacting to setpoint changes, focusing instead on changes in the process variable.

Mathematical Illustration Suppose there is an instantaneous increase in the setpoint at time $t = t_0$:

$$y_{\text{sp}}(t) = y_{\text{sp}}(t_0^-) + \Delta y_{\text{sp}} \tag{1.53}$$

This causes an instantaneous change in error:

$$e(t) = e(t_0^-) + \Delta y_{\text{sp}} \tag{1.54}$$

Proportional Term:

$$u_P(t) = Ke(t) = K\left(e(t_0^-) + \Delta y_{\text{sp}}\right) \tag{1.55}$$

Integral Term:

$$u_I(t) = K\frac{1}{T_i}\int_0^t e(\tau)\,d\tau \tag{1.56}$$

Initially, $u_I(t)$ changes slowly because the integral over a very short time is small.
Derivative Term:

$$u_D(t) = KT_d\frac{de(t)}{dt} \tag{1.57}$$

Since the change in $e(t)$ is instantaneous, $\frac{de(t)}{dt}$ approaches infinity, causing $u_D(t)$ to spike. To prevent this, derivative action is often modified in practical controllers.

Solution to Problem 2

(a) Derivation of the Transfer Function $G_c(s)$

The PID controller is given by:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right] \tag{1.58}$$

where $e(t) = y_{\text{sp}}(t) - y(t)$.

Taking the Laplace transform of both sides (assuming zero initial conditions):

$$\begin{aligned} U(s) &= K\left[E(s) + \frac{1}{T_i}\cdot\frac{1}{s}E(s) + T_d\cdot sE(s)\right] \\ &= K\left[\left(1 + \frac{1}{T_i s} + T_d s\right)E(s)\right] \end{aligned}$$

Therefore, the transfer function $G_c(s)$ of the PID controller is:

$$G_c(s) = \frac{U(s)}{E(s)} = K\left(T_d s^2 + s + \frac{1}{T_i}\right)\cdot\frac{1}{s} \tag{1.59}$$

Simplifying:

$$G_c(s) = K\left(\frac{T_d s^2 + s + \frac{1}{T_i}}{s}\right) \tag{1.60}$$

(b) Influence of K, T_i, and T_d on Poles and Zeros

The transfer function $G_c(s)$ has:

- **A pole at** $s = 0$ (from the denominator s).
- **Zeros** determined by the roots of the numerator polynomial:

$$T_d s^2 + s + \frac{1}{T_i} = 0 \tag{1.61}$$

Effect of Proportional Gain K The gain K scales the entire transfer function but does not affect the location of poles or zeros.

Effect of Integral Time T_i The integral time T_i influences the constant term $\frac{1}{T_i}$ in the numerator:

- **Decreasing** T_i (stronger integral action) increases $\frac{1}{T_i}$.
- This changes the constant term in the numerator, affecting the roots of the numerator polynomial and thus the zeros of $G_c(s)$.

Effect of Derivative Time T_d The derivative time T_d affects the coefficient of s^2 in the numerator:

- **Increasing** T_d (stronger derivative action) increases the coefficient of s^2.
- This alters the curvature of the numerator polynomial, shifting the zeros.

Analysis of Zeros To see how T_i and T_d influence the zeros, solve:

$$T_d s^2 + s + \frac{1}{T_i} = 0 \tag{1.62}$$

The roots are:

$$s = \frac{-1 \pm \sqrt{1 - \frac{4T_d}{T_i}}}{2T_d} \tag{1.63}$$

- The discriminant $\Delta = 1 - \frac{4T_d}{T_i}$ determines the nature of the zeros.
- **If** $\Delta > 0$: Real and distinct zeros.
- **If** $\Delta = 0$: Real and repeated zero.
- **If** $\Delta < 0$: Complex conjugate zeros.

Impact on Controller Dynamics

- **Integral Time** T_i:
  - Shorter T_i (stronger integral action) increases $\frac{1}{T_i}$, which can lead to complex zeros if $\frac{4T_d}{T_i} > 1$.
  - Complex zeros can introduce oscillatory behavior in the controller's response.
- **Derivative Time** T_d:
  - Larger T_d (stronger derivative action) increases the likelihood of having real zeros, potentially improving the system's phase margin and damping.

Summary The parameters T_i and T_d influence the zeros of the controller's transfer function, thereby affecting the frequency response and transient behavior of the control system:

- Adjusting T_i modifies the integral action, impacting steady-state error elimination and the position of zeros.
- Adjusting T_d changes the derivative action, influencing the system's damping and stability by altering the zeros.

The proportional gain K affects the overall gain but does not change the poles or zeros, thus scaling the controller's response without altering its dynamic characteristics.

Solution to Problem 3

(a) Derivation of the Closed-Loop Transfer Function

Given the process transfer function:

$$G_p(s) = \frac{1}{s+1} \tag{1.64}$$

The proportional controller is:

$$G_c(s) = K_p \tag{1.65}$$

In a unity feedback configuration, the closed-loop transfer function $G_{cl}(s)$ is:

$$G_{cl}(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)} \tag{1.66}$$

Substituting $G_c(s)$ and $G_p(s)$:

$$\begin{aligned} G_{cl}(s) &= \frac{K_p\left(\frac{1}{s+1}\right)}{1 + K_p\left(\frac{1}{s+1}\right)} \\ &= \frac{\frac{K_p}{s+1}}{\frac{s+1+K_p}{s+1}} \\ &= \frac{K_p}{s+1+K_p} \end{aligned}$$

Therefore, the closed-loop transfer function is:

$$G_{cl}(s) = \frac{K_p}{s + (1 + K_p)} \tag{1.67}$$

(b) Range of K_p for Stability

The characteristic equation is obtained from the denominator:

$$s + (1 + K_p) = 0 \tag{1.68}$$

Solving for the pole:

$$s = -(1 + K_p) \tag{1.69}$$

For the system to be stable, the pole must be in the left-half of the complex plane:

$$\text{Re}(s) < 0 \tag{1.70}$$

Since $1 + K_p > 0$ for all $K_p > -1$, we have:

$$s = -(1 + K_p) < 0 \quad \text{if} \quad K_p > -1 \tag{1.71}$$

Therefore, the closed-loop system is stable for all values of K_p satisfying:

$$K_p > -1 \tag{1.72}$$

(c) Steady-State Error for $K_p = 10$

With $K_p = 10$, the closed-loop transfer function becomes:

$$G_{cl}(s) = \frac{10}{s+11} \tag{1.73}$$

For a unit step input $R(s) = \frac{1}{s}$, the output is:

$$Y(s) = G_{cl}(s)R(s) = \frac{10}{(s+11)s} \tag{1.74}$$

The error signal is:

$$E(s) = R(s) - Y(s) = \frac{1}{s} - \frac{10}{(s+11)s} \tag{1.75}$$

Simplify $E(s)$:

$$\begin{aligned} E(s) &= \frac{(s+11)-10}{(s+11)s} \\ &= \frac{s+1}{(s+11)s} \end{aligned}$$

Using the Final Value Theorem to find the steady-state error $e(\infty)$:

$$e(\infty) = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s) = \lim_{s\to 0} \frac{s(s+1)}{(s+11)s} = \lim_{s\to 0} \frac{s+1}{s+11} \tag{1.76}$$

Evaluating the limit:

$$e(\infty) = \frac{0+1}{0+11} = \frac{1}{11} \approx 0.0909 \tag{1.77}$$

Explanation of Steady-State Error A steady-state error exists because the system is a **Type 0** system (no poles at the origin in the open-loop transfer function). Proportional control alone cannot eliminate steady-state error for a step input in such systems.

The presence of steady-state error is due to the finite gain of the system at zero frequency:

$$\lim_{s\to 0} G_{cl}(s) = \lim_{s\to 0} \frac{10}{s+11} = \frac{10}{11} \tag{1.78}$$

This indicates that the output will reach $\frac{10}{11}$ of the input step magnitude at steady state, resulting in a steady-state error of $\frac{1}{11}$.

To eliminate the steady-state error, an integral component would be required in the controller to increase the system type and provide zero steady-state error for a step input.

Solution to Problem 4

Given:

- The output reaches 28.3% of its final value at $t = 2$ seconds. - The output reaches 63.2% of its final value at $t = 5$ seconds.

Assuming the process can be approximated by a First-Order Plus Dead Time (FOPDT) model:

$$G_p(s) = \frac{K_p e^{-Ls}}{\tau s + 1} \tag{1.79}$$

Our goal is to determine the PID controller parameters using the Ziegler-Nichols step response method.

Estimating Process Parameters

First, we need to estimate the process parameters K_p, L, and τ.

Since we have a unit step input and percentage outputs, we can assume the process gain $K_p = 1$ (the final value of the output corresponds to $K_p \times$ input magnitude).

The step response of the FOPDT model is:

$$y(t) = K_p \left(1 - e^{-\dfrac{t-L}{\tau}}\right) u(t-L) \tag{1.80}$$

For $t \geq L$.

We are given two data points:

1. At $t_1 = 2$ s, $y(t_1) = 0.283\, K_p$ 2. At $t_2 = 5$ s, $y(t_2) = 0.632\, K_p$

Substituting these into the step response equation:

$$y(t_1) = K_p \left(1 - e^{-\dfrac{t_1-L}{\tau}}\right) = 0.283\, K_p \tag{1.81}$$

$$\implies 1 - e^{-\dfrac{t_1-L}{\tau}} = 0.283 \tag{1.82}$$

$$\implies e^{-\dfrac{t_1-L}{\tau}} = 1 - 0.283 = 0.717 \quad (1) \tag{1.83}$$

Similarly:

$$y(t_2) = K_p \left(1 - e^{-\dfrac{t_2 - L}{\tau}} \right) = 0.632\, K_p \tag{1.84}$$

$$\implies 1 - e^{-\dfrac{t_2 - L}{\tau}} = 0.632 \tag{1.85}$$

$$\implies e^{-\dfrac{t_2 - L}{\tau}} = 1 - 0.632 = 0.368 \quad (2) \tag{1.86}$$

Taking the natural logarithm of both equations:
From (1):

$$-\frac{t_1 - L}{\tau} = \ln(0.717) \tag{1.87}$$

$$\implies \frac{t_1 - L}{\tau} = -\ln(0.717) \approx 0.333 \quad (3) \tag{1.88}$$

From (2):

$$-\frac{t_2 - L}{\tau} = \ln(0.368) \tag{1.89}$$

$$\implies \frac{t_2 - L}{\tau} = -\ln(0.368) \approx 1.0 \quad (4) \tag{1.90}$$

Subtract equation (3) from equation (4):

$$\frac{t_2 - L}{\tau} - \frac{t_1 - L}{\tau} = 1.0 - 0.333 \tag{1.91}$$

$$\implies \frac{(t_2 - t_1)}{\tau} = 0.667 \tag{1.92}$$

$$\implies \tau = \frac{t_2 - t_1}{0.667} = \frac{5 - 2}{0.667} \approx \frac{3}{0.667} \approx 4.5\,\mathrm{s} \tag{1.93}$$

Now, substitute τ back into equation (3) to find L:

$$\frac{t_1 - L}{\tau} = 0.333 \tag{1.94}$$

$$\implies t_1 - L = 0.333\,\tau \tag{1.95}$$

$$\implies L = t_1 - 0.333\,\tau \tag{1.96}$$

$$\implies L = 2 - 0.333 \times 4.5 = 2 - 1.5 = 0.5\,\mathrm{s} \tag{1.97}$$

Applying Ziegler-Nichols Tuning Rules

For the PID controller, the Ziegler-Nichols step response method provides the following tuning formulas:

$$K_c = 1.2\frac{\tau}{K_p L} \tag{1.98}$$

$$T_i = 2L \tag{1.99}$$

$$T_d = 0.5L \tag{1.100}$$

Given $K_p = 1$, $\tau = 4.5$ s, and $L = 0.5$ s:

Calculating K_c

$$K_c = 1.2\frac{\tau}{K_p L} = 1.2\frac{4.5}{1 \times 0.5} = 1.2 \times 9 = 10.8 \tag{1.101}$$

Calculating T_i

$$T_i = 2L = 2 \times 0.5 = 1.0\,\text{s} \tag{1.102}$$

Calculating T_d

$$T_d = 0.5L = 0.5 \times 0.5 = 0.25\,\text{s} \tag{1.103}$$

Result

The suitable PID controller parameters are:

- Proportional gain: $K_c = 10.8$
- Integral time: $T_i = 1.0$ s
- Derivative time: $T_d = 0.25$ s

Solution to Problem 5

An open-loop stable process is controlled using a proportional controller. When the proportional gain is increased to $K_u = 15$, the closed-loop system exhibits sustained oscillations with a period of $T_u = 4$ seconds.

(a) Confirmation of Ultimate Gain and Ultimate Period

The ultimate gain K_u is defined as the gain at which the closed-loop system begins to oscillate with constant amplitude (marginally stable). The corresponding oscillation period is the ultimate period T_u.

Given that the system exhibits sustained oscillations when the proportional gain is $K = 15$ and the period of oscillation is $T = 4$ seconds, we can confirm that:

- The **ultimate gain** is $K_u = 15$.
- The **ultimate period** is $T_u = 4$ seconds.

This data corresponds to the conditions required for applying the Ziegler-Nichols frequency response method.

(b) Calculation of PID Controller Parameters Using Ziegler-Nichols Method

The Ziegler-Nichols frequency response method provides tuning formulas for PID controller parameters based on K_u and T_u.

The standard Ziegler-Nichols tuning rules for a PID controller are:

$$\begin{aligned} K &= 0.6\,K_u \\ T_i &= 0.5\,T_u \\ T_d &= 0.125\,T_u \end{aligned}$$

Calculating the Proportional Gain K

$$\begin{aligned} K &= 0.6 \times K_u \\ &= 0.6 \times 15 \\ &= 9 \end{aligned}$$

Calculating the Integral Time T_i

$$\begin{aligned} T_i &= 0.5 \times T_u \\ &= 0.5 \times 4 \\ &= 2 \text{ seconds} \end{aligned}$$

Calculating the Derivative Time T_d

$$\begin{aligned} T_d &= 0.125 \times T_u \\ &= 0.125 \times 4 \\ &= 0.5 \text{ seconds} \end{aligned}$$

Resulting PID Controller Parameters

Using the Ziegler-Nichols frequency response method, the PID controller parameters are:

- **Proportional Gain**: $K = 9$
- **Integral Time**: $T_i = 2$ seconds
- **Derivative Time**: $T_d = 0.5$ seconds

Conclusion

By applying the Ziegler-Nichols tuning formulas, we have calculated the PID controller parameters based on the ultimate gain and ultimate period obtained from the experimental data. These parameters can be used to configure the PID controller for the process to achieve a satisfactory control performance.

Solution to Problem 6

(a) Explanation of Integrator Windup and Its Problematic Nature

Integrator Windup Integrator windup occurs in PID controllers when the integral term accumulates excessively due to actuator saturation. When the control signal $u(t)$ reaches its saturation limits $u_{\min}$ or $u_{\max}$, the actuator cannot respond proportionally to further increases in the controller output. However, the integral term continues to integrate the error $e(t)$, causing it to "wind up" to large values.

Why It Is Problematic in This System In the given system with actuator saturation limits $u_{\min} = -10$ and $u_{\max} = 10$, integrator windup can lead to:

- **Overshoot and Oscillations**: Once the actuator comes out of saturation, the unwound integral term can cause a large control action, leading to overshoot and oscillatory behavior.
- **Delayed Settling Time**: The excessive integral action prolongs the time the system takes to settle at the desired setpoint.
- **Reduced Stability**: The system may become unstable due to the lag introduced by the accumulated integral term.

Therefore, it is essential to prevent integrator windup to maintain system performance and stability.

(b) Design of Anti-Windup Mechanism Using Back-Calculation Method

To prevent integrator windup, we implement an anti-windup scheme using the back-calculation method.

Standard PID Controller The PID controller output before saturation (unsaturated control signal $v(t)$) is:

$$v(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right) \tag{1.104}$$

Incorporating Actuator Saturation The actual control signal $u(t)$ after applying saturation limits is:

$$u(t) = \operatorname{sat}(v(t)) = \begin{cases} u_{\max}, & \text{if } v(t) > u_{\max} \\ v(t), & \text{if } u_{\min} \le v(t) \le u_{\max} \\ u_{\min}, & \text{if } v(t) < u_{\min} \end{cases} \tag{1.105}$$

Anti-Windup via Back-Calculation Modify the integral term to include a feedback correction based on the difference between $v(t)$ and $u(t)$:

$$\frac{dI(t)}{dt} = e(t) + \frac{1}{T_t}\left(u(t) - v(t)\right) \tag{1.106}$$

where:

- $I(t)$ is the integral term.
- T_t is the tracking (anti-windup) time constant.

Derivation 1. **Compute the Unsaturated Control Signal $v(t)$:**

$$v(t) = K\left(e(t) + \frac{1}{T_i}I(t) + T_d\frac{de(t)}{dt}\right) \tag{1.107}$$

2. **Apply Saturation Limits to Obtain $u(t)$:**

$$u(t) = \operatorname{sat}(v(t)) \tag{1.108}$$

3. **Update the Integral Term with Back-Calculation:**

$$\frac{dI(t)}{dt} = e(t) + \frac{1}{T_t}\left(u(t) - v(t)\right) \tag{1.109}$$

- The term $\frac{1}{T_t}(u(t) - v(t))$ corrects the integral action when saturation occurs ($u(t) \neq v(t)$).

Selection of the Tracking Time Constant T_t The choice of T_t affects the responsiveness of the anti-windup mechanism:

- **Smaller T_t:** Faster correction of windup but may introduce noise or instability.
- **Larger T_t:** Slower correction, windup may persist longer.

A common guideline is to set T_t proportional to the integral time T_i:

$$T_t = \alpha T_i \tag{1.110}$$

where α is a constant between 0.1 and 1. A typical choice is $\alpha = 0.1$ to 0.3:

$$T_t = 0.2 T_i \tag{1.111}$$

Alternatively, if derivative action is significant, T_t can be chosen as:

$$T_t = \sqrt{T_i T_d} \tag{1.112}$$

Implementation Equations The complete PID controller with anti-windup is:

$$v(t) = K\left(e(t) + \frac{1}{T_i} I(t) + T_d \frac{de(t)}{dt}\right) \tag{1.113}$$

$$u(t) = \text{sat}\,(v(t)) \tag{1.114}$$

$$\frac{dI(t)}{dt} = e(t) + \frac{1}{T_t}\left(u(t) - v(t)\right) \tag{1.115}$$

Summary The back-calculation method effectively prevents integrator windup by adjusting the integral term based on actuator saturation. By selecting an appropriate T_t, the integral action remains effective without causing instability or excessive overshoot.

Solution to Problem 7

(a) Derivation of Discrete-Time Equations

We are to derive the discrete-time equations for the proportional, integral, and derivative terms of a PID controller using:

- **Backward difference approximation** for the derivative.
- **Rectangular approximation** for the integral.

The continuous-time PID controller is given by:

$$u(t) = K\left[e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d \frac{de(t)}{dt}\right] \tag{1.116}$$

where:

- $u(t)$ is the control signal.
- $e(t) = y_{\text{sp}}(t) - y(t)$ is the control error.
- K is the proportional gain.
- T_i is the integral time constant.
- T_d is the derivative time constant.

Let h be the sampling period ($h = 0.1$ seconds) and $t_k = kh$. Define:

- $e_k = e(t_k)$
- $u_k = u(t_k)$

Proportional Term The proportional term in discrete time is straightforward:

$$P_k = Ke_k \tag{1.117}$$

Integral Term Using the rectangular approximation (specifically, the backward rectangular method) for the integral:

$$\begin{aligned}\int_0^{t_k} e(\tau)\,d\tau &\approx \int_0^{t_{k-1}} e(\tau)\,d\tau + e_k h \\ &= I_{k-1} + e_k h\end{aligned}$$

where I_{k-1} is the accumulated integral up to time t_{k-1}.

Therefore, the integral term is updated as:

$$I_k = I_{k-1} + K\frac{h}{T_i}e_k \tag{1.118}$$

Derivative Term Using the backward difference approximation for the derivative:

$$\left.\frac{de(t)}{dt}\right|_{t=t_k} \approx \frac{e_k - e_{k-1}}{h} \tag{1.119}$$

Thus, the derivative term is computed as:

$$D_k = KT_d\frac{e_k - e_{k-1}}{h} \tag{1.120}$$

Summary of Discrete-Time Equations The discrete-time PID controller equations are:

$$P_k = Ke_k \tag{1.121}$$

$$I_k = I_{k-1} + K\frac{h}{T_i}e_k \tag{1.122}$$

$$D_k = KT_d\frac{e_k - e_{k-1}}{h} \tag{1.123}$$

$$u_k = P_k + I_k + D_k \tag{1.124}$$

(b) Step-by-Step Algorithm for Implementing the PID Controller

Initialization Before starting the control loop, initialize the following variables:

- Set the initial integral term: $I_0 = 0$
- Set the previous error: $e_{-1} = 0$

Control Loop At each sampling instant $t_k = kh$, perform the following steps:

1. **Measure the process variable** y_k and **compute the error**:
$$e_k = y_{\text{sp},k} - y_k \tag{1.125}$$
2. **Compute the proportional term**:
$$P_k = Ke_k \tag{1.126}$$
3. **Update the integral term** using the rectangular approximation:
$$I_k = I_{k-1} + K\frac{h}{T_i}e_k \tag{1.127}$$
4. **Compute the derivative term** using the backward difference approximation:
$$D_k = KT_d\frac{e_k - e_{k-1}}{h} \tag{1.128}$$
5. **Compute the control signal**:
$$u_k = P_k + I_k + D_k \tag{1.129}$$
6. **Apply the control signal** u_k to the process (e.g., send u_k to the actuator).
7. **Update the previous error** for the next iteration:
$$e_{k-1} = e_k \tag{1.130}$$

Algorithm Summary The algorithm can be summarized in pseudocode as:

```
Initialize:
    I = 0
    e_prev = 0

For each sampling instant k:
    Measure y_k
    Compute e_k = y_sp_k - y_k
    P = K * e_k
    I = I + K * (h / T_i) * e_k
    D = K * T_d * (e_k - e_prev) / h
    u_k = P + I + D
    Output u_k to the actuator
    Update e_prev = e_k
```

Notes

- The integral term I_k accumulates the error over time to eliminate steady-state error.
- The derivative term D_k predicts the future trend of the error, improving system stability.
- Proper tuning of K, T_i, and T_d is essential for optimal controller performance.
- Anti-windup strategies may be needed if actuator saturation is possible.
- Initializing $e_{-1} = e_0$ can prevent a large derivative action at startup.

Solution to Problem 8

(a) Analysis of the Open-Loop Response

Given the process transfer function:

$$G_p(s) = \frac{K_p}{(s+a)(s+b)} \tag{1.131}$$

with $K_p = 2$, $a = 1$, and $b = 0.5$. Substituting these values:

$$G_p(s) = \frac{2}{(s+1)(s+0.5)} \tag{1.132}$$

Time Constants The poles of the system are at $s = -1$ and $s = -0.5$. The corresponding time constants are:

$$\begin{aligned}\tau_1 &= \frac{1}{|s_1|} = \frac{1}{1} = 1 \text{ second}\\ \tau_2 &= \frac{1}{|s_2|} = \frac{1}{0.5} = 2 \text{ seconds}\end{aligned}$$

Steady-State Gain The steady-state gain is found by evaluating $G_p(s)$ at $s = 0$:

$$G_p(0) = \frac{2}{(0+1)(0+0.5)} = \frac{2}{(1)(0.5)} = \frac{2}{0.5} = 4 \tag{1.133}$$

Open-Loop Response to a Unit Step Input For a unit step input $R(s) = \frac{1}{s}$, the open-loop output is:

$$Y(s) = G_p(s)R(s) = \frac{2}{(s+1)(s+0.5)} \cdot \frac{1}{s} \tag{1.134}$$

The inverse Laplace transform would show that the output will rise gradually and eventually reach the steady-state value of 4, with dynamics determined by the time constants $\tau_1 = 1$ s and $\tau_2 = 2$ s.

(b) Design of the PID Controller

Our goal is to design a PID controller to achieve a desired closed-loop time constant $T_c = 0.5$ seconds. The PID controller is given by:

$$G_c(s) = K\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{1.135}$$

Desired Closed-Loop Transfer Function Assuming unity feedback, the closed-loop transfer function is:

$$G_{cl}(s) = \frac{G_c(s)G_p(s)}{1 + G_c(s)G_p(s)} \tag{1.136}$$

We aim for the closed-loop system to have a dominant pole at:

$$s = -\frac{1}{T_c} = -2 \text{ (since } T_c = 0.5 \text{ s)} \tag{1.137}$$

To achieve a desired closed-loop characteristic equation of:

$$(s+2)^3 = 0 \implies s^3 + 6s^2 + 12s + 8 = 0 \tag{1.138}$$

Derivation of the Characteristic Equation First, write the open-loop transfer function $L(s) = G_c(s)G_p(s)$:

$$L(s) = K\left(1 + \frac{1}{T_i s} + T_d s\right) \cdot \frac{2}{(s+1)(s+0.5)} \tag{1.139}$$

Express the PID controller as a single fraction:

$$G_c(s) = K\left(\frac{T_i T_d s^2 + T_i s + 1}{T_i s}\right) \tag{1.140}$$

Therefore,

$$L(s) = K \cdot \frac{T_i T_d s^2 + T_i s + 1}{T_i s} \cdot \frac{2}{(s+1)(s+0.5)} \tag{1.141}$$

Simplify $L(s)$:

$$L(s) = \frac{2K(T_i T_d s^2 + T_i s + 1)}{T_i s(s+1)(s+0.5)} \tag{1.142}$$

The characteristic equation is:

$$1 + L(s) = 0 \tag{1.143}$$

Multiply both sides by $T_i s(s+1)(s+0.5)$:

$$T_i s(s+1)(s+0.5) + 2K(T_i T_d s^2 + T_i s + 1) = 0 \tag{1.144}$$

Expand the terms:

$$T_i s(s^2 + 1.5s + 0.5) = T_i s^3 + 1.5 T_i s^2 + 0.5 T_i s$$
$$2K(T_i T_d s^2 + T_i s + 1) = 2K T_i T_d s^2 + 2K T_i s + 2K$$

Combine the terms to form the characteristic equation:

$$T_i s^3 + (1.5T_i + 2KT_iT_d)s^2 + (0.5T_i + 2KT_i)s + 2K = 0 \tag{1.145}$$

Divide both sides by T_i to simplify:

$$s^3 + (1.5 + 2KT_d)s^2 + (0.5 + 2K)s + \frac{2K}{T_i} = 0 \tag{1.146}$$

Matching Coefficients Set the coefficients equal to those of the desired characteristic equation:

$$\begin{aligned}
\text{Coefficient of } s^3 &: 1 = 1 \\
\text{Coefficient of } s^2 &: 1.5 + 2KT_d = 6 \\
\text{Coefficient of } s^1 &: 0.5 + 2K = 12 \\
\text{Constant term} &: \frac{2K}{T_i} = 8
\end{aligned}$$

Solving for Controller Parameters 1. Solve for K from the s^1 coefficient:

$$\begin{aligned}
0.5 + 2K &= 12 \\
2K &= 11.5 \\
K &= \frac{11.5}{2} = 5.75
\end{aligned}$$

2. Solve for T_d from the s^2 coefficient:

$$\begin{aligned}
1.5 + 2KT_d &= 6 \\
2(5.75)T_d &= 6 - 1.5 \\
11.5T_d &= 4.5 \\
T_d &= \frac{4.5}{11.5} \approx 0.3913 \text{ seconds}
\end{aligned}$$

3. Solve for T_i from the constant term:

$$\begin{aligned}
\frac{2K}{T_i} &= 8 \\
\frac{2(5.75)}{T_i} &= 8 \\
\frac{11.5}{T_i} &= 8 \\
T_i &= \frac{11.5}{8} \approx 1.4375 \text{ seconds}
\end{aligned}$$

Resulting PID Controller Parameters The suitable PID controller parameters are:

- **Proportional Gain**: $K = 5.75$
- **Integral Time**: $T_i = 1.4375$ seconds
- **Derivative Time**: $T_d = 0.3913$ seconds

(c) Simulation of the Closed-Loop Response

Closed-Loop Characteristic Equation With the designed controller, the closed-loop characteristic equation is:

$$(s+2)^3 = 0 \tag{1.147}$$

which means the system has a triple pole at $s = -2$.

Time Response The inverse Laplace transform of the closed-loop transfer function gives the step response:

$$y(t) = 1 - e^{-2t}(1 + 2t + 2t^2) \tag{1.148}$$

Performance Metrics

- **Rise Time**: The rise time (t_r) is the time taken for the response to go from 10% to 90% of its final value. For this system, the response is monotonic and reaches 90% of the final value when:

  $$y(t_r) = 0.9 \tag{1.149}$$

  Solving numerically, $t_r \approx 1$ second.

- **Overshoot**: Since the response is monotonic and there are no oscillations, the overshoot is zero.

- **Settling Time**: The settling time (t_s) is the time taken for the response to stay within a certain percentage (typically 2% or 5%) of the final value. For a pole at $s = -2$, the settling time is approximately:

  $$t_s \approx \frac{4}{2} = 2 \text{ seconds (for 2\% criterion)} \tag{1.150}$$

Comments on Performance The designed PID controller achieves the following:

- **Fast Response**: The system responds quickly to setpoint changes, reaching near the final value within approximately 1 second.
- **No Overshoot**: The absence of overshoot ensures the temperature does not exceed the desired setpoint, which is important in temperature control applications.
- **Quick Settling**: The system settles within 2% of the final value by around 2 seconds.

Conclusion The PID controller designed achieves the desired closed-loop time constant of $T_c = 0.5$ seconds. The simulation indicates a stable and efficient system with satisfactory performance metrics suitable for temperature control applications.

Presentation

PPT Link