Discrete-Time PID Control

1.0 Learning Outcomes……1
2.0 Introduction……1
3.0 Discrete-Time PID Difference Equations……...2
4.0 Summary……..……...8
Recommended Citation……8

1.0 Learning Outcomes

After studying this ACE-Lab learning guide, you should be able to demonstrate the Learning Outcomes listed in Table 1. On completion of the ACE-Lab learning guide, revisit Table 1 and confirm whether each Learning Outcome has been achieved.

Table 1: *Learning Outcomes from the Discrete-Time PID Control learning guide*

ID	Learning Outcome	Achieved
1.	Describe how proportional, integral and derivative actions contribute to a discrete-time PID controller.	☐
2.	Express the proportional, integral and derivative control terms using discrete-time difference equations.	☐
3.	Combine the individual control terms to formulate a discrete-time PID controller suitable for implementation on a microcontroller.	☐

2.0 Introduction

A discrete-time PID (proportional, integral and derivative) controller is widely used in embedded and digital control systems to regulate a measured output in real time. The controller operates at a fixed sampling interval, T_s, and calculates a new control signal at each sample using the current and previous error values.

In practical implementations, the three-term PID (i.e., proportional, integral and derivative) actions are expressed using difference equations so they can be executed directly on a microcontroller, as illustrated in Figure 1. The integral term can be implemented using the Trapezoidal Rule, while the derivative term can be approximated using Backward Euler (also referred to as the backward-difference approximation). These methods are well suited to embedded control because they require only a small number of stored values and relatively little computation, with their practical implementation explored in the following section.

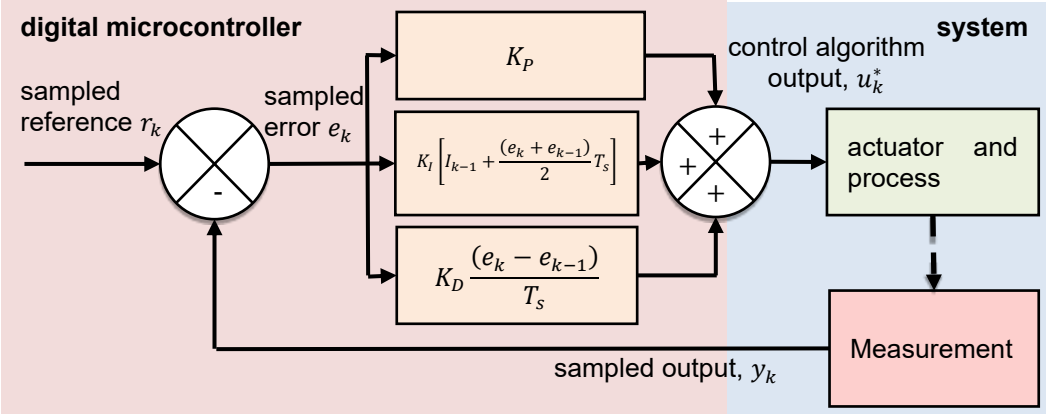


Figure 1: Discrete-Time *Proportional, Integral and Derivative (PID) Control*

3.0 Discrete-Time PID Difference Equations

For implementation on a microcontroller, the PID controller can be expressed in a discrete-time difference equation form. This allows the control signal to be calculated at each sampling instant using the current and previously stored error values.

Proportional (P) Control

The proportional (P) controller calculates the control output from the current error between the desired reference value and the measured system output. At sampling instant, k, the error is given by:

$$e_k = r_k - y_k \tag{1}$$

where r_k denotes the desired reference value and y_k denotes the measured system output. The proportional contribution to the controller output, $u_{P,k}$, is then given by:

$$u_{P,k} = K_P e_k \tag{2}$$

where K_P is the proportional gain. The proportional action therefore responds directly to the magnitude of the current error, with its operation illustrated in block diagram form in Figure 2. A larger error produces a larger control action, providing an immediate response that drives the measured output towards the desired reference.

The value of K_P influences how strongly the controller responds to the error and, consequently, the resulting closed-loop system response. A value that is too low may result in a slow response and persistent steady-state error, whereas a value that is too high may produce excessive overshoot, oscillation or, in some systems, instability.

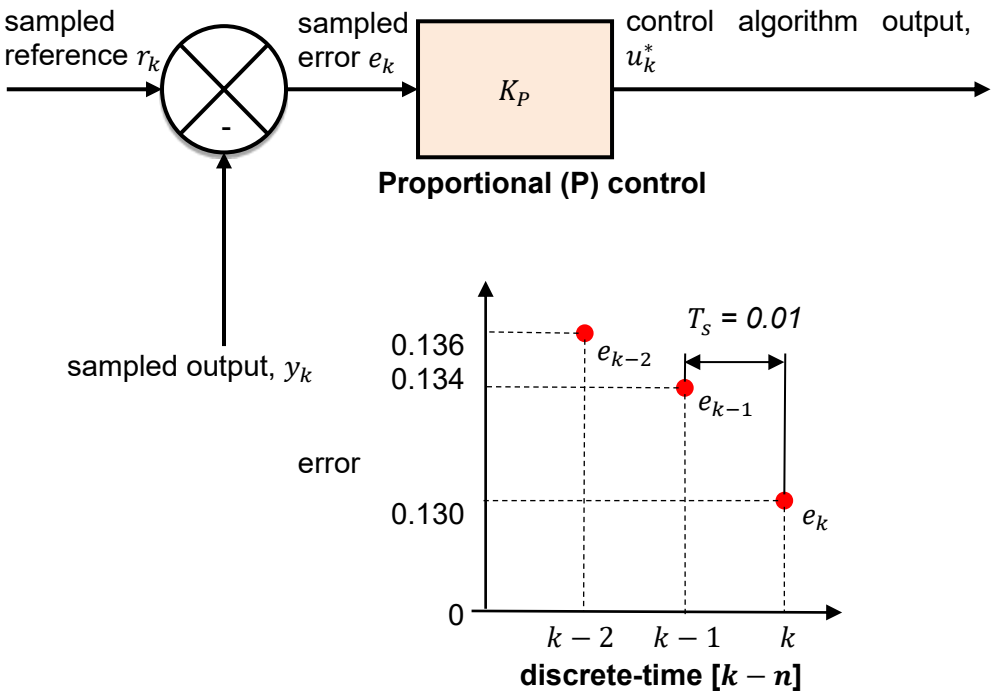


Figure 2: *Proportional (P) Control*

Integral (I) Control

The integral action accounts for the accumulation of error over time. In a discrete-time implementation, this accumulated error can be approximated using the Trapezoidal Rule, which calculates the area between consecutive error samples, as illustrated in Figure 3.

For the current sampling interval, the incremental area is given by:

$$\Delta I_k = \frac{(e_k + e_{k-1})}{2} T_s \tag{3}$$

where e_k and e_{k-1} denote the error values at the current and previous sampling instants, respectively, and T_s denotes the sampling period. Equation (3) determines the mean of the two error values and multiplies it by the width of the sampling interval, T_s.

At each sampling instant, this new incremental area is added to the accumulated integral value from the previous sampling instant:

$$I_k = I_{k-1} + \frac{(e_k + e_{k-1})}{2} T_s \tag{4}$$

where I_k represents the accumulated integral of the error and I_{k-1} is the previously accumulated integral of the error.

The area calculated during the current sampling interval represents only the latest contribution to the integral. To represent the integral of the error over time,

this area must be added to the accumulated value from the previous sampling instant.

For example, if the error remains constant at $e = 1$ and the sampling period is $T_s = 0.1s$, using Equation (3), the incremental area added during each sampling interval is given by:

$$\Delta I_k = \frac{(1+1)}{2}(0.1) = 0.1$$

Using Equation (4), the accumulated integral therefore increases at each sampling instant:

$$I_1 = 0.1, \quad I_2 = 0.2, \quad I_3 = 0.3,\ldots$$

If only the area from the current sampling interval were used, the value would remain at 0.1 and would not represent the accumulation of error over time. The recursive summation therefore allows a persistent error to progressively increase the integral value.

As the accumulated integral value increases, the integral contribution to the control signal also increases. Once the error has been reduced to zero, the accumulated integral value can remain at the level required to provide the control effort needed to hold the system at the desired reference. This is what allows integral action to eliminate persistent steady-state error.

In a practical implementation, I_{k-1} is stored in memory and fed back into the calculation at the next sampling instant. This recursive form means that the complete history of the error does not need to be stored or recalculated. Only the previous accumulated integral value, the current error and the previous error are required, making the method well suited to real-time implementation on a microcontroller.

The integral contribution to the controller output is therefore calculated as:

$$u_{I,k} = K_I I_k \tag{5}$$

where K_I is the integral gain. As error persists over successive sampling intervals, the accumulated integral value increases, allowing the integral action to progressively increase its contribution to the control signal. This is particularly useful for reducing or eliminating persistent steady-state error, as illustrated in Figure 3.

After each controller update, the newly calculated value, I_k is stored and used as I_{k-1}, during the next sampling interval.

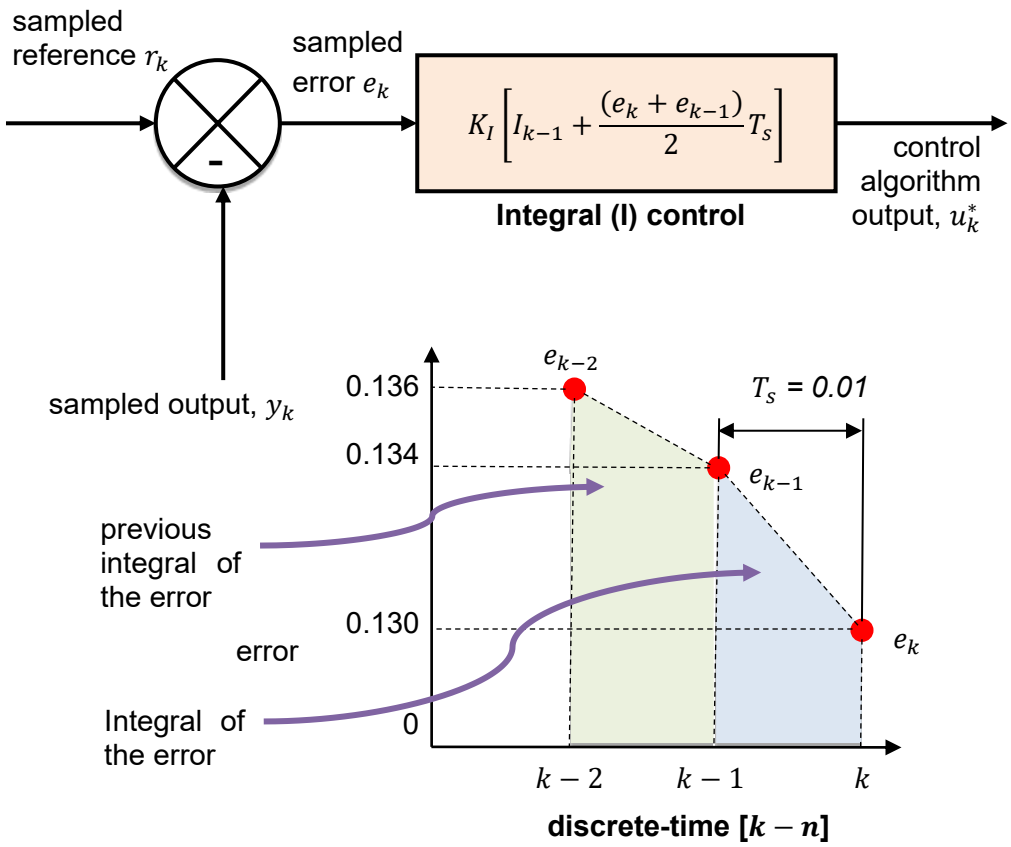


Figure 3: *Integral (I) Control*

Derivative (D) Control

The derivative action responds to the rate at which the error is changing. In a discrete-time implementation, this rate of change can be approximated using the backward-difference approximation (also referred to as the Backward Euler approximation), obtained using the current and previous error samples over the sampling period:

$$\frac{de}{dt} \approx \frac{(e_k - e_{k-1})}{T_s} \tag{6}$$

Using this approximation, the derivative contribution to the controller output is given by:

$$u_{D,k} = K_D \frac{(e_k - e_{k-1})}{T_s} \tag{7}$$

where K_D is the derivative gain, see Figure 4.

The derivative action therefore responds to how quickly the error is changing. A rapid change in error produces a larger derivative contribution, whereas little or no change in error produces a small or zero derivative contribution.

If the error remains constant between sampling instants, $e_k - e_{k-1}$, the derivative contribution is zero. Derivative action therefore responds only to

changes in the error and does not provide a control contribution for a constant error.

The rate of change of the error is useful because it provides an indication of how quickly the system is approaching or moving away from the desired reference. For example, if the error is positive but decreasing rapidly, the system output is approaching the reference quickly. In this case, $e_k - e_{k-1}$ is negative and the derivative contribution therefore becomes negative. This can oppose part of the proportional control action and reduce the overall control effort as the system approaches the reference.

For example, consider:

$$e_{k-1} = 4, \quad e_k = 2, \quad T_S = 0.1s$$

Using Equation (6), the rate of change of the error is:

$$\frac{(e_k - e_{k-1})}{T_s} = \frac{2-4}{0.1} = -20$$

If the derivative gain K_D is 0.05, Equation (7) gives:

$$u_{D,k} = 0.05(-20) = -1$$

The negative derivative contribution indicates that the error is decreasing. In this case, the derivative action reduces the overall control effort as the system approaches the reference.

A positive derivative contribution occurs when the error is increasing between sampling instants, such that $e_k > e_{k-1}$. From Equation (6), the rate of change of the error is then positive and, for a positive derivative gain K_D, Equation (7) produces a positive derivative contribution. This increases the control signal in the positive direction in response to the changing error.

The sign of the derivative contribution therefore depends on the direction in which the error is changing:

$$e_k - e_{k-1} > 0 \quad \Longrightarrow \quad u_{D,k} > 0$$

$$e_k - e_{k-1} < 0 \quad \Longrightarrow \quad u_{D,k} < 0$$

where $K_D > 0$.

When the system is approaching the reference rapidly, the derivative contribution can oppose part of the existing proportional action and provide a damping effect. This can help to limit overshoot and reduce oscillation in the system response.

As differentiation is particularly sensitive to high-frequency measurement noise, small fluctuations in the measured signal can produce relatively large changes in the derivative output. In practical implementations, filtering is therefore

commonly applied to reduce the influence of high-frequency noise. The derivative gain, K_D, should also be tuned carefully, as an excessively large value can amplify noise and produce excessive or rapidly changing control effort, whereas a value that is too small may have little influence on the system response.

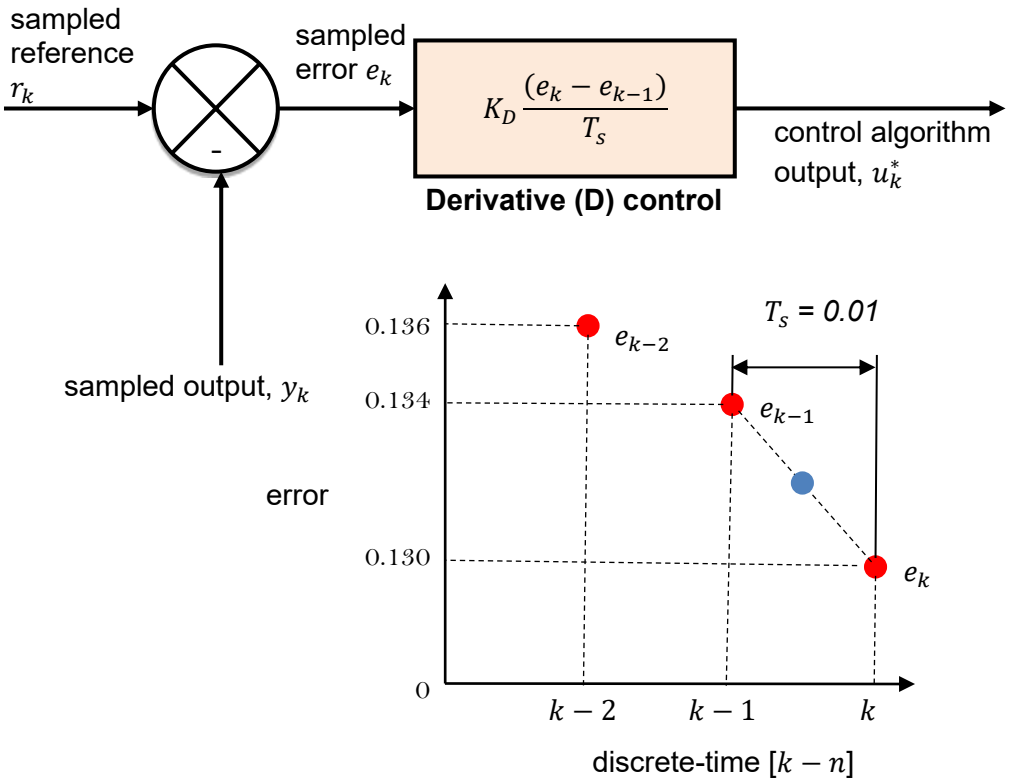


Figure 4: *Derivative (D) Control*

Proportional, Integral and Derivative (PID) Control

The three-term discrete-time PID controller combines the proportional, integral and derivative contributions to form the overall control signal, as illustrated in Figure 5, with this represented by:

$$u_k^* = u_{P,k} + u_{I,k} + u_{D,k} \tag{8}$$

Substituting the individual proportional, integral and derivative contributions gives:

$$u_k^* = K_P e_k + K_I \left[I_{k-1} + \frac{(e_k + e_{k-1})}{2} T_s \right] + K_D \frac{(e_k - e_{k-1})}{T_s} \tag{9}$$

In practice, these gains are tuned to achieve the required system response while also considering the limitations of the physical implementation. These may include actuator saturation, PWM resolution, sensor resolution and noise, the selected sampling period, and the numerical precision and computational capability of the microcontroller. Appropriate filtering and anti-windup may also be required to ensure reliable controller operation.

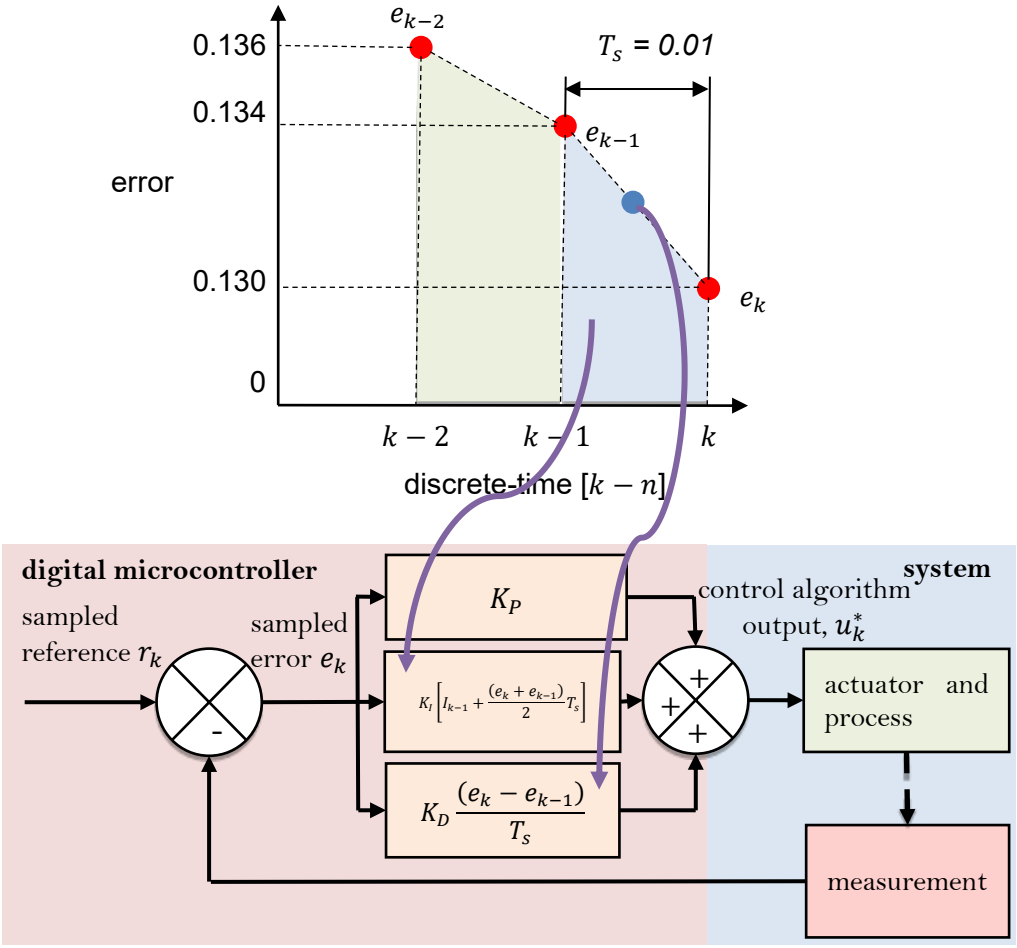


Figure 5: *PID Control*

4.0 Summary

This learning guide introduced proportional, integral and derivative control in discrete-time form for real-time execution on a microcontroller. The proportional action was calculated directly from the current error, while the integral and derivative actions were implemented using the Trapezoidal Rule and backward-difference approximation, respectively.

Return to the Learning Outcomes at the beginning of this learning guide and consider whether you can now confidently demonstrate each one.

Recommended Citation

If you have used this practical exercise in your teaching, learning or research, please cite it using the reference below, adapting the format as required by your preferred referencing style.

Pickering, J. E. (2026). Discrete-Time PID Control. ACE-Lab. Teaching Notes: ACE-Lab Practical Exercises, Version 1.0. https://zenodo.org/records/23086025