Integral Controllers

- create a restoring force that is proportional to the sum of all past errors multiplied by time,

$$Output_I = K_I K_P \Sigma(E\ \Delta t)$$

Where,

$Output_I$ = controller output due to integral control

K_I = integral gain constant (sometimes expressed as $1/T_I$)

K_p = proportional gain constant

$\Sigma(E\Delta t)$ = sum of all past errors (multiplied by the time they existed)

- For a constant value of error, the value of $\Sigma(E\Delta t)$ will increase with time, causing the restoring force to get larger and larger.
- Eventually, the restoring force will get large enough to overcome friction and move the controlled variable in a direction to eliminate the error.
- The introduction of **integral control** in a control system can reduce the steady-state error to zero.

Integral Controllers: Steady-State-Error Problem

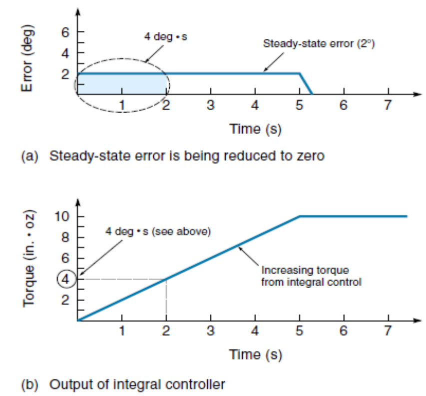


Figure 6.8 Graph showing integral control eliminating a steady-state error.

Integral Controllers:

The proportional feedback system of (*KP* = 10 in. · oz/deg) has been modified to include integral feedback. The arm has been at rest (at the 30° position) when a weight is placed on the end of the arm, causing a downward torque of 40 in. · oz [Figure 6.9(a)].

- Describe how the control system responds to the weight.

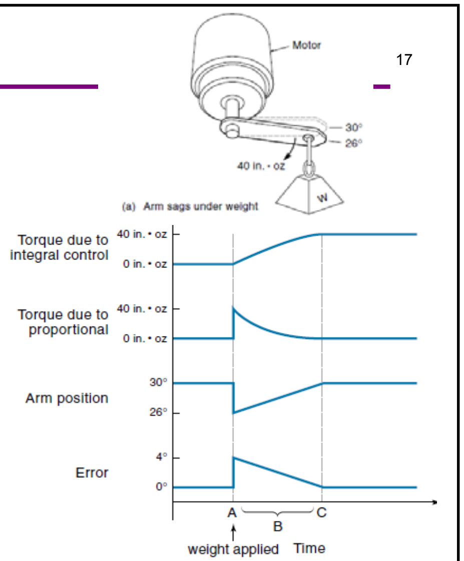


Figure 6.9 The system response of proportional plus integral control.

Integral Controllers: Drawback

- Although the addition of integral feedback eliminates the steady-state-error problem, it reduces the overall stability of the system.
- The problem occurs because integral feedback tends to make the system overshoot, which may lead to oscillations.
- Also, the response of integral feedback is relatively slow because it takes a while for the 'error.time' area to build up.

Integral Controllers: Drawback

- All mechanical systems have friction, and friction is nonlinear—that is, it takes more force to overcome friction when the object is at rest than it does to keep an object moving.
- At time = 0, the system in Figure 6.10 has just moved to a new position and stopped, leaving a steady-state error. The restoring force is equal to the contribution from proportional control plus the increasing force from the integral control.
- For a while the object doesn't move, but finally the combined restoring force overcomes friction and the object "breaks loose."
- Once moving, the friction force immediately drops so some force is "left over," which goes to accelerating the object.
- This may cause it to overshoot, and the whole process starts again from the other side.

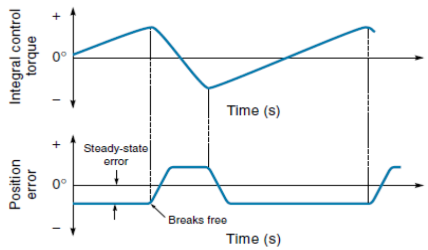


Figure 6.10 The system response of proportional plus integral control.

Derivative Controllers

- One solution to the overshoot problem is to include derivative control.
- **Derivative controller** "applies the brakes," slowing the controlled variable just before it reaches its destination.
- Mathematically, the contribution from derivative controller is expressed by the following equation:

$$\text{Output}_D = K_D K_P \frac{\Delta E}{\Delta t}$$

where

Output_D = controller output due to derivative control
K_D = derivative gain constant (sometimes expressed T_D, unit is time)
K_P = proportional gain constant
$\frac{\Delta E}{\Delta t}$ = error rate of change (slope of error curve)

Derivative Controllers

- Assume the controlled variable is initially at 0°.
- At time A the set point moves rapidly to 30°.
- Because of mechanical inertia, it takes time for the object to get up to speed.
- Notice that the position error (E) is increasing (positive slope) during this time period (A to B).
- Therefore, derivative control, which is proportional to error slope, will have a positive output, which gives the object a boost, to help get it moving.
- As the controlled variable closes in on the set-point value (B to C), the position error is decreasing (negative slope), so derivative feedback applies a negative force that acts like a brake, helping to slow the object.

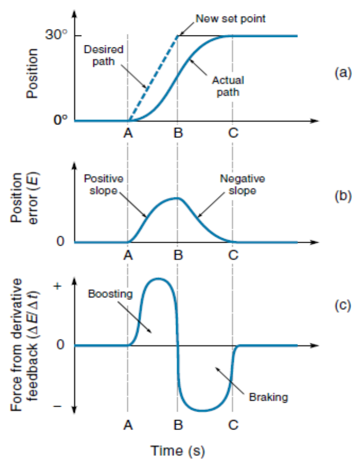


Figure 6.11 Contribution of derivative control, showing boosting and braking.

Derivative Controllers

- derivative control improves system performance in two ways.
  - first, it provides an extra boost of force at the beginning of a change to promote faster action;
  - second, it provides for braking when the object is closing in on the new set point. This braking action not only helps reduce overshoot but also tends to reduce steady-state error.
- *Derivative control has no influence on the accuracy of the system, just the response time, so it is never used by itself.*

PID Controllers

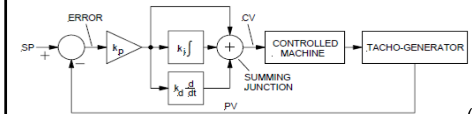
- Many control systems use a combination of the three types of feedback already discussed: **P**roportional + **I**ntegral + **D**erivative (**PID**) Control.
- The foundation of the system is proportional control.
- Adding integral control provides a means to eliminate steady-state error but may increase overshoot.
- Derivative control is good for getting sluggish systems moving faster and reduces the tendency to overshoot.
- The response of the PID system can be described by the following Equation, which simply adds together the three components required:

$$\text{Output}_{\text{PID}} = K_P E + K_I K_P \Sigma(E\Delta t) + K_D K_P \frac{\Delta E}{\Delta t}$$

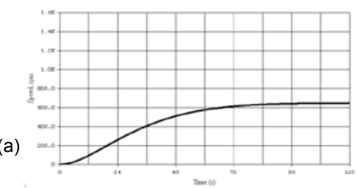
where

$\text{Output}_{\text{PID}}$ = output of the PID controller
K_P = proportional control gain
K_I = integral control gain (often seen as $1/TI$)
K_D = derivative control gain (often seen as TD)
E = error (deviation from the set point)
$\Sigma(E\Delta t)$ = sum of all past errors (area under the error · time curve)
$\Delta E/\Delta t$ = rate of change of error (slope of the error curve)

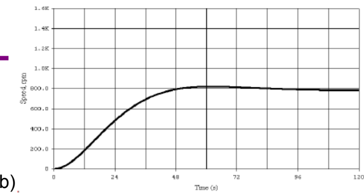
PID Controllers: DC Motor speed control



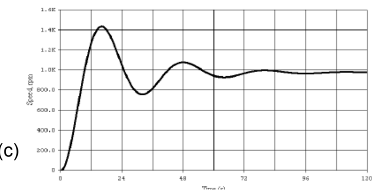
- Closed Loop Control System with Ideal PID



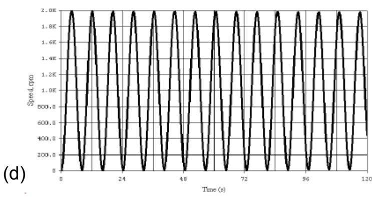
(a)



(b)



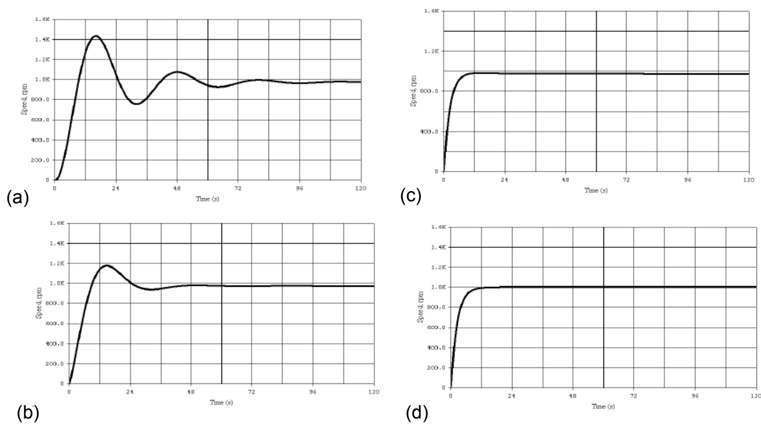
(c)



(d)

DC Motor Speed Control Response with
(a) Low k_p Only, (b) Moderate k_p,
(c) High K_p, (d) very High k_p

PID Controllers: DC Motor speed control



DC Motor Speed Control Response (a) with High k_p Only, (b) with High k_p and Low Derivative Gain k_d, (c) with High k_p and Moderate Derivative Gain k_d, (d) with High k_p, Moderate Derivative Gain k_d, and Low Integral Gain k_i.

PIP Controllers

- In a dynamic system, such as a robot arm, the desired position is a moving target, in which case we are concerned with path control.
- Further, the desired path between two points may not be a straight line.
- For example, a welding robot needs to follow the path of the seam.
- The **feedforward,** or PIP approach is a way to implement path control.

  PIP = Proportional + **I**ntegral + **P**review
- PIP controller that incorporates information of the future path in its current output.
- Many systems have this information available — either the entire path is stored in memory or the system is equipped with a preview sensor as illustrated in Figure 6.12 for a welding robot.

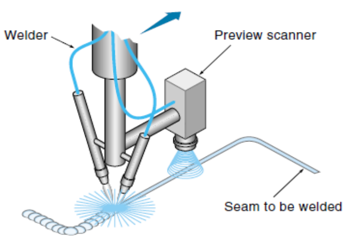


Figure 6.12 A welder using a preview sensor for PIP control.

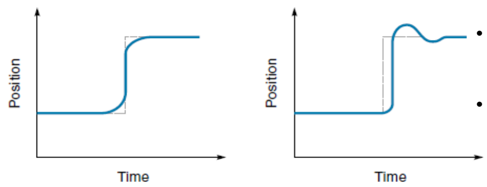
PIP Controllers

Output = $K_P E + K_I K_P \Sigma(E\,\Delta t) + K_F (P_{T+1} - P_T)$

K_F = feedforward gain constant

P_T = position it should be in now

P_{T+1} = position it should be in, in the future (at T + 1)



(a) With feedforward (b) Without feedforward

Figure 6.13 Improved path control with feedforward.

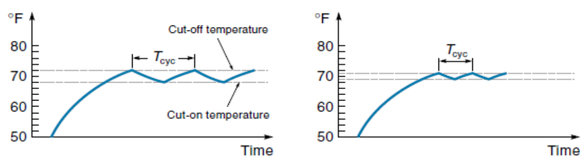
- feedforward term, $K_F (P_{T+1} - P_T)$, is proportional to the difference between where the controlled object is and where it must be in the future.
- If this number is large, the system has a long way to go and should speed up.
- If the number is small or zero, the system will be stopping and so should begin slowing down.
- In each case, the value of the feedforward term is added to the controller output so that the controlled object is pushed with more or less force, depending on where it has to be in the near future.
- By anticipating the change in direction, the PIP system can begin slowing ahead of time and minimize overshoot.

On-Off Controllers: Two-Point Control

- **Two-point control** (also called **on–off control**) is the simplest type of closed-loop control strategy.
- The actuator can push the controlled variable with only full force or no force.
- When the actuator is off, the controlled variable settles back to some rest state.
- A good example of two-point control is a thermostatically controlled heating system.
- Consider a house sitting for a long time with the heat turned off and an outside temperature of 50°F. Eventually, the inside temperature would drop to 50°. This is its rest-state temperature.
- Now suppose the heat is turned on and the thermostat is set for an average temperature of 70°.
- As Figure 6.14(a) shows, the inside temperature begins to climb, rapidly at first, and then more slowly (as the heat losses increase). When the temperature reaches the 72° cutoff point, the furnace shuts down.
- The house temperature immediately starts to decline toward its rest state of 50°; but long before it gets there, it reaches the cut-on point of 68°, and the furnace comes back on.
- Notice that the temperature curve is like a charging and discharging capacitor.

On-Off Controllers: Two-Point Control

- Note: there is a cycle time (T_{cyc}) associated with two-point control.
- This cycle time is affected by the capacity of the furnace and the house, as well as the temperature difference between the cut-on and cut-off points.
- If the limits were moved closer together—say, 69° and 71°, the temperature would be maintained closer to 70°, but the cycle frequency would increase, as illustrated in Figure 6.14(b).
- Generally, a high cycle rate is undesirable because of wear on motors and switches.
- two-point control has only limited applications, mostly on slow-moving systems where it is acceptable for the controlled variable to move back-and-forth between the two limit points.



(a) Cut-on = 68°; cut-off = 72° (b) Cut-on = 69°; cut-off = 71°

Figure 11.4 Temperature curve of a two-point heating system.