Proportional-Integral Control

ENCE361 Embedded Systems 1

Course Coordinator: Ciaran Moore (ciaran.moore@Canterbury.ac.nz)

Lecturer: Le Yang (le.yang@canterbury.ac.nz)

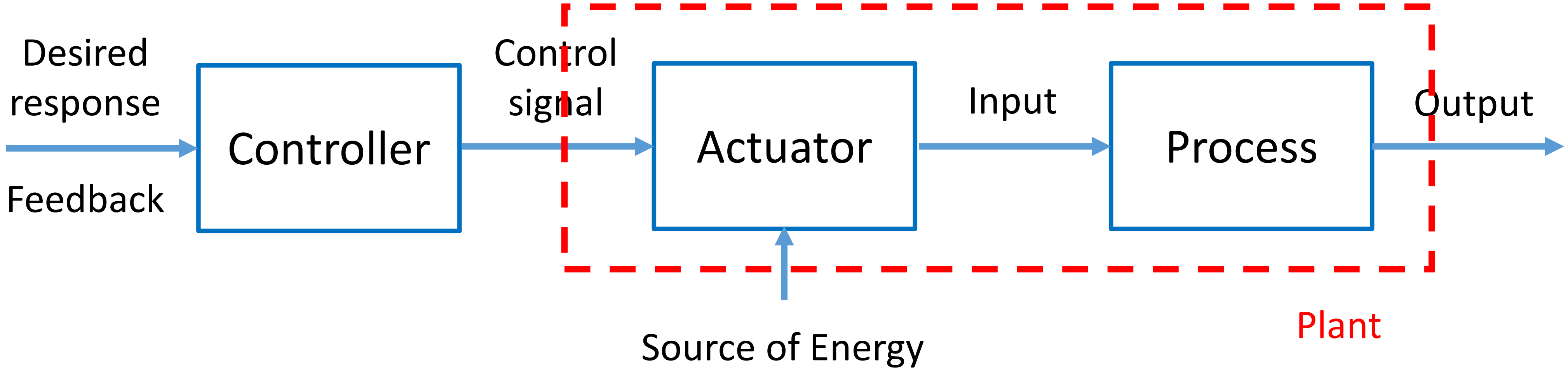
Department of Electrical and Computer Engineering

Where we’re going today

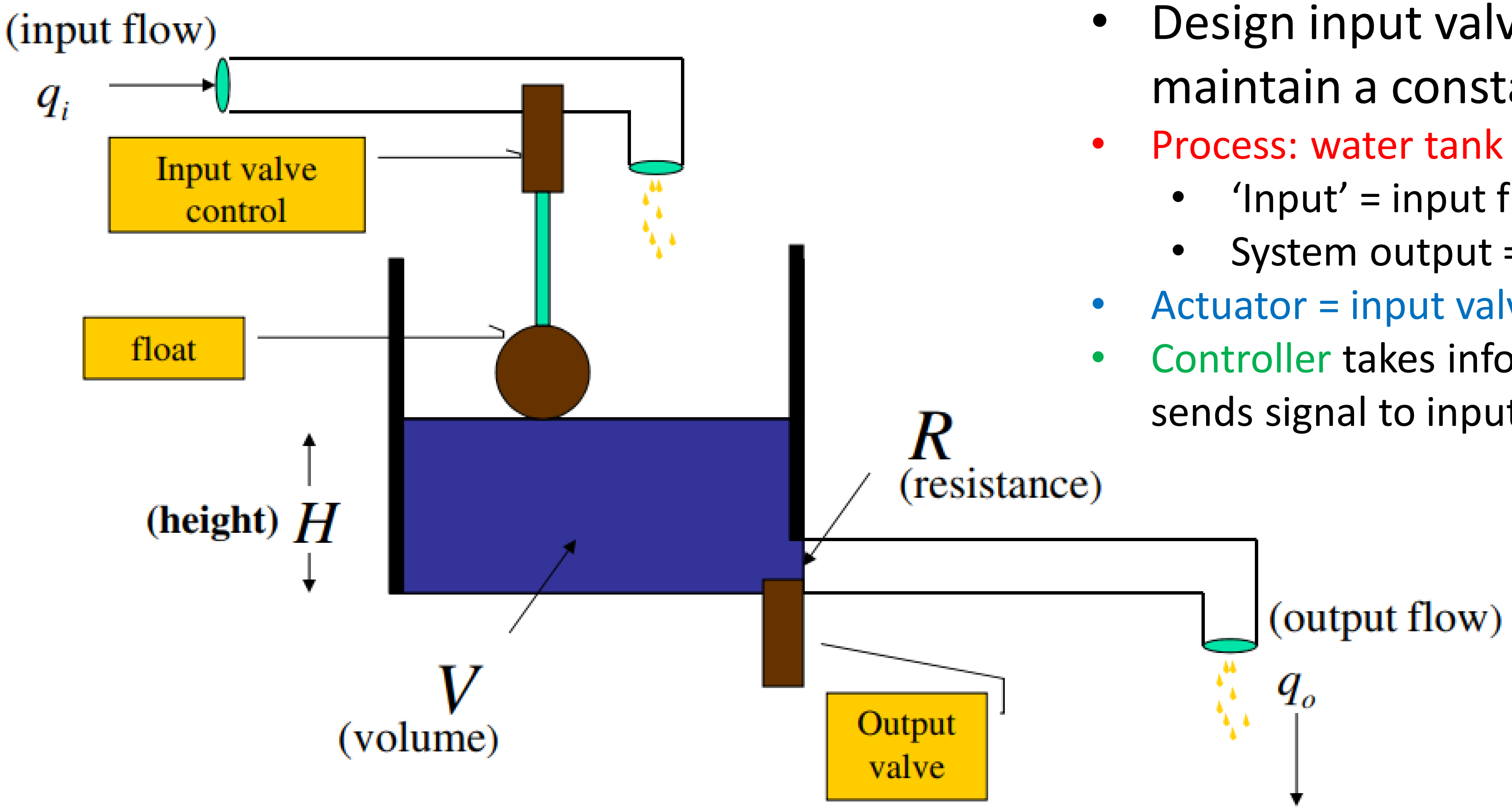
- **Introduction to control systems**
- Proportional control
- Integral control
- Digital PI control

Introduction to Control Systems

- Control system = interconnection of components forming a configuration that provides a desired system response
  - Process: a series of task together transforming inputs to outputs
  - Actuator (mover): take control signal and convert the source of energy into (mechanic) move
  - Controller: produce the control signal in a format suitable as input to actuator



Example: Liquid Level System

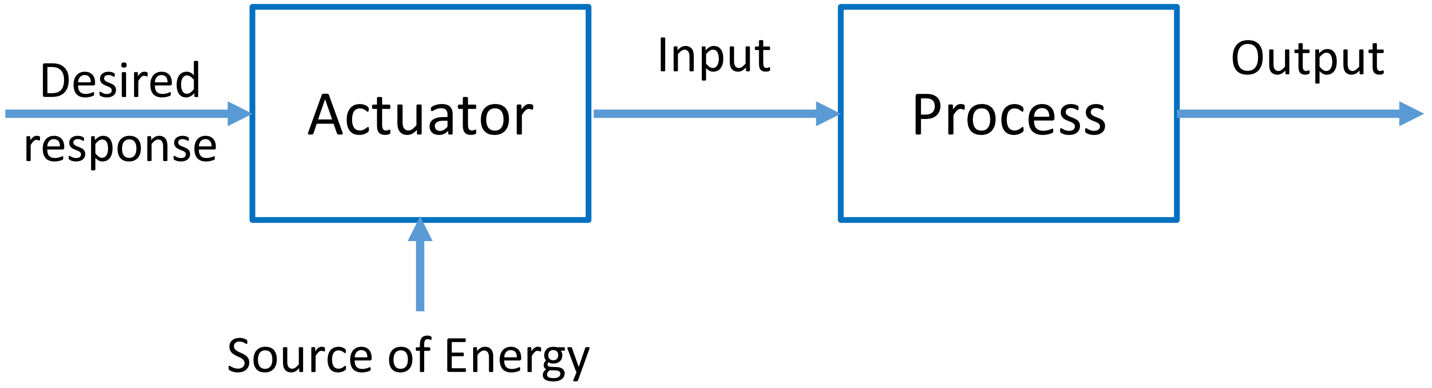


- Design input valve control to maintain a constant height
- Process: water tank
  - 'Input' = input flow + output flow
  - System output = water height
- Actuator = input valve
- Controller takes info from float and sends signal to input valve

From http://www.cs.wayne.edu/~hzhang/courses/8260b/Lectures/4-0%20-%20ControlTheory-Tutorial-for-CS-IBM.pdf

Open-Loop Control System

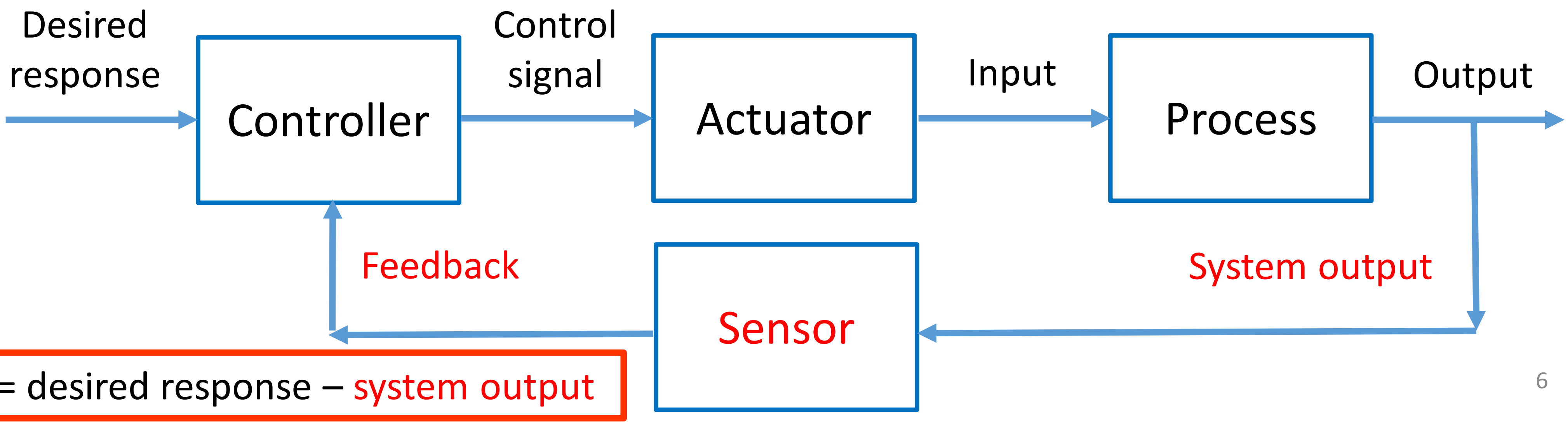
- Use a controller or actuator to directly control the process



- Conceptually simple and easy
  - Example: toaster, old microwave oven, washing machine
- Design an open-loop control system requires *accurate* knowledge on the plant (actuator + process)
  - Unreliable when there are unexpected variations in the system

Closed-Loop Control System

- Open-loop control system does not monitor the system output but simply assumes it works as expected
- Closed-loop control system uses a sensor (e.g., yourself ☺) to feed system output back to adjust controller behavior *adaptively*



Error signal = desired response – system output

Where we’re going today

- Introduction to control systems
- **Proportional control**
- Integral control
- Digital PI control

Proportional Control

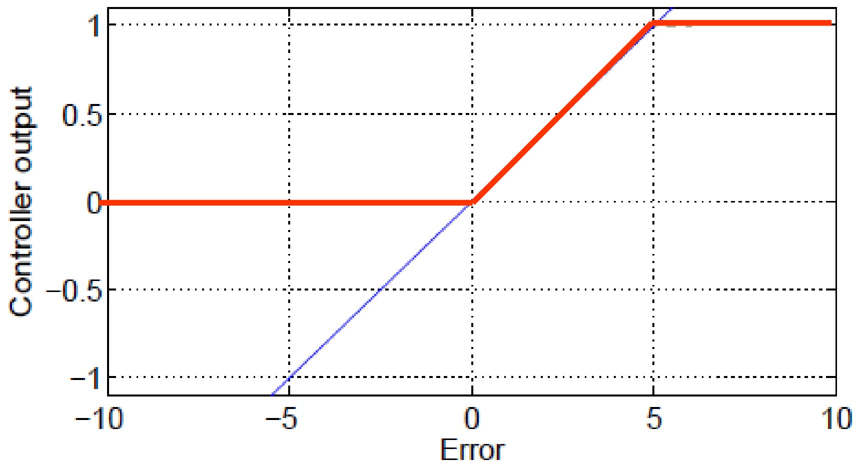
- Proportional control (P) monitors

  error signal e(t) = desired response x(t) – system output m(t)

- But let the magnitude of control signal depend on the error magnitude
- Proportional controller drives actuator in proportional to e(t)
  - Controller output is

    $$c(t) = k_p e(t) = k_p(x(t) - m(t))$$

  - k_p is the proportional control gain
  - $c(t) = \begin{cases} 1, & e(t) > T \\ 0, & e(t) < 0 \\ k_p e(t), & 0 < e(t) < T \end{cases}$



Transfer characteristic of a proportional control system.

Offset Error (1)

- Proportional controller drives actuator in proportional to e(t)
  - Controller output is

$$c(t) = k_p e(t) = k_p(x(t) - m(t))$$

  - At equilibrium, we have

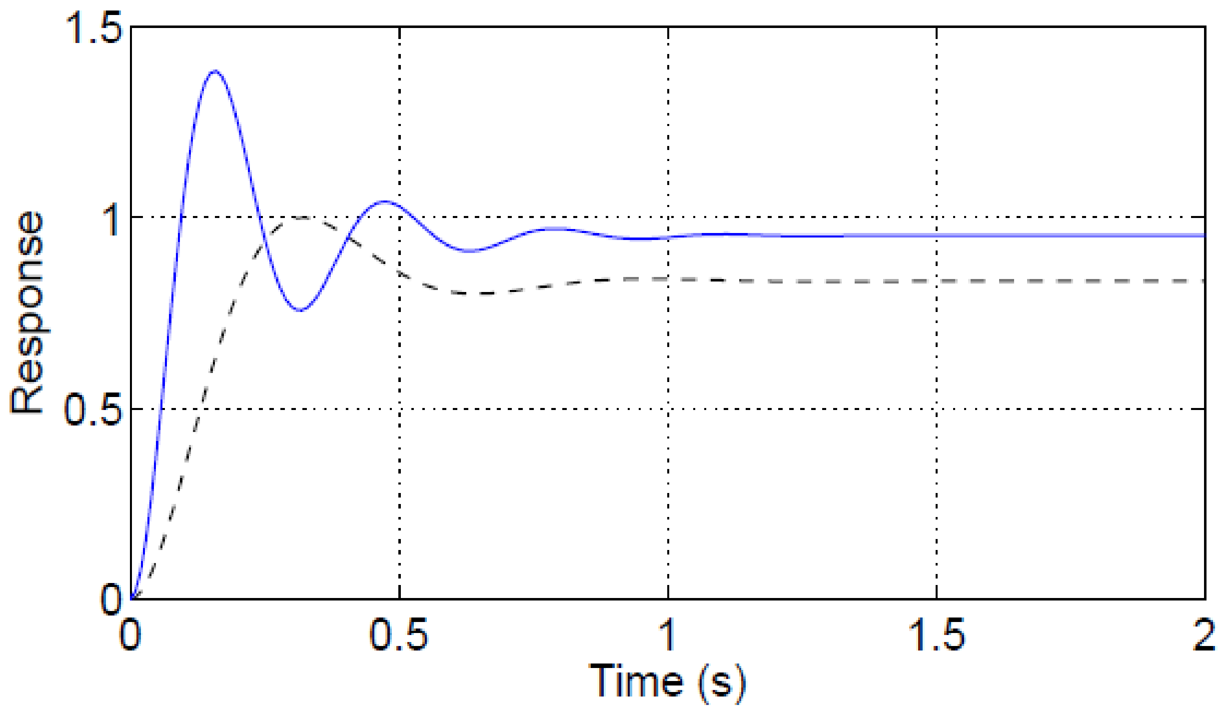
$$m(t) = k_p e(t) = k_p(x(t) - m(t))$$

$$\Downarrow$$

$$m(t) = \frac{k_p}{k_p + 1} x(t) \neq x(t)$$

Offset Error (2)

- Proportional control has offset error
  - Increasing gain k_p (blue)
    - Faster response
    - Decrease offset error
    - Increase overshoot
  - Decreasing gain k_p (dashed)
    - Slower response
    - Increase offset error
    - Decrease overshoot



Where we’re going today

- Introduction to control systems
- Proportional control
- **Integral control**
- Digital PI control

Integral Control (1)

- Problem of proportional control
  - Controller output solely depends on instantaneous error signal
    - Offset error always exists

$$m(t) = \frac{k_p}{k_p + 1}\, x(t) \neq x(t)$$

    - Increasing gain k_p reduces offset error at the cost of increased overshoot and possible instability

- Naïve solution: manual reset
  - To achieve $m(t) = x(t)$, provide a scaled response $x'(t) = \frac{k_p+1}{k_p} x(t)$ such that

$$m(t) = \frac{k_p}{k_p + 1}\, x'(t) = \frac{k_p}{k_p + 1} * \frac{k_p + 1}{k_p} x(t) = x(t)$$

Integral Control (2)

- Integral control: an automatic approach to correct offset error $e(t)$
  - Controller output depends on the integral of error signal e(t)

$$c(t) = k_i \int_{-\infty}^{t} e(\tau)d\tau$$

  - k_I is the integral control gain

- If there is an offset error (i.e., $e(t)$ is non-zero)
  - Integral control would increases $c(t)$ to correct it
- Even if $e(t)$ is zero, $c(t)$ can still be non-zero!
  - Proportional Integral (PI) control

- Integrating error signal $e(t) = x(t) - m(t)$ may reduce noise in measured system output $m(t)$
  - Recall digital signal conditioning in Lectures 4 & 5

Integral Control (3)

- Integral controller output depends on entire history of the error signal
  - It could introduce overshoot and even oscillation

  - Example: a positive $e(t)$ persists → increased integral control output

    $e(t)$ = 0 but positive integral control output persists → overshoot

    a negative $e(t)$ → reduced integral control output

    $e(t)$ = 0 but negative integral control output persists → oscillation

Digital Realization of Integral Control

- How to implement error integration in C code?
  - Sampling with is needed to approximate <u>numerically</u> the integral using summation

Error signal : $e(t) \rightarrow e(nT_s)$, T_s: sampling interval

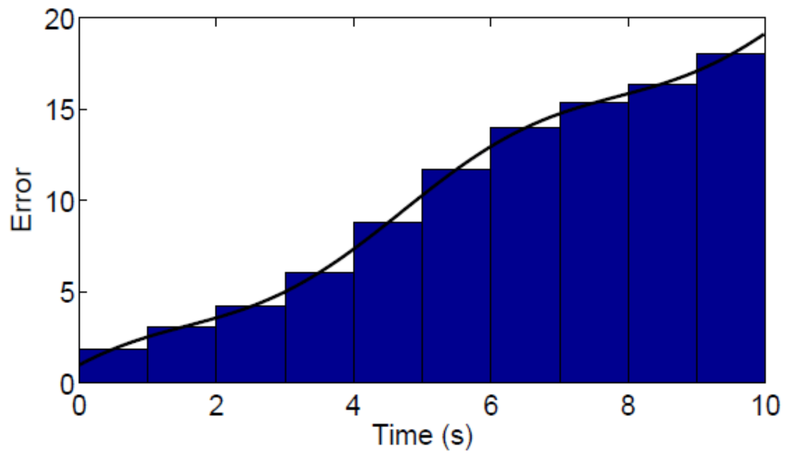
Control Signal: $c(t) \rightarrow c(nT_s)$, possible DAC needed before output

$$c(t) = k_i \int_{-\infty}^{t} e(\tau) d\tau$$

$$\Downarrow$$

$E(nT_s) = E((n-1)T_s) + T_s \cdot e(nT_s)$, Approximated signal integral

$c(nT_s) = k_i \cdot E(nT_s)$, Amplify control signal



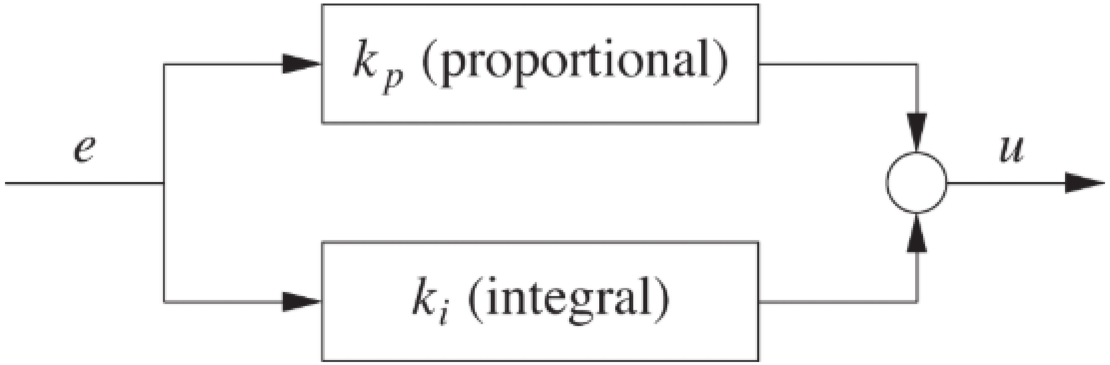
Numerical integration with ZOH

Where we’re going today

- Introduction to control systems
- Proportional control
- Integral control
- **Digital PI control**

Digital PI Control

Block diagram of PI controller



- PI controller output:

$$c(t) = k_p e(t) + k_i \int_{-\infty}^{t} e(\tau) d\tau$$

- Digital PI controller output:

$$c(nT_s) = k_p e(nT_s) + k_i \left(E\big((n-1)T_s\big) + T_s \cdot e(nT_s) \right)$$

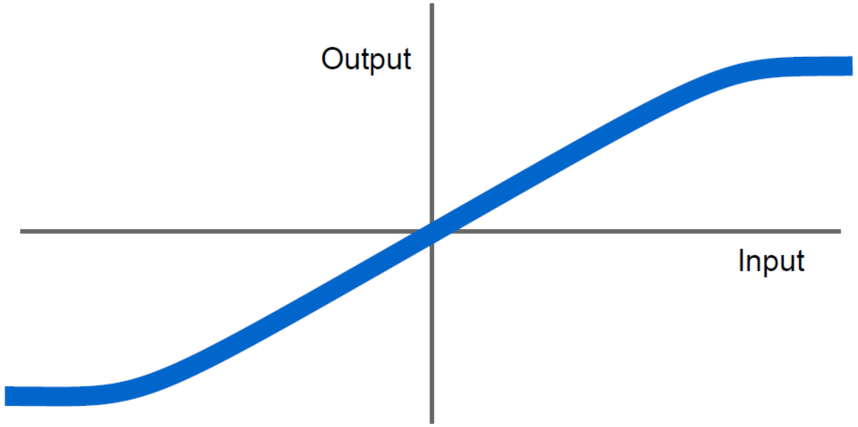
$$E(nT_s) = E\big((n-1)T_s\big) + T_s \cdot e(nT_s)$$

Digital Realization

```
static double error_integrated = 0.0;   // Error signal integration
static double error_previous = 0.0;     // Previous error sample


double pid_update (double error, double proportional_gain,
                   double integral_gain, double derivative_gain,
                   double delta_t)
{
    double error_derivative;
    double control;

    error_integrated += error * delta_t;         // Error signal integration
    error_derivative = (error - error_previous) / delta_t;  // Error signal time derivative

    control = error * proportional_gain          // Proportional control
        + error_integrated * integral_gain       // Integral control
        + error_derivative * derivative_gain;    // Derivative control

    error_previous = error;                      // Update previous error sample

    return control;                              // Control signal c(t)
}
```

Output Saturation & Integral Windup

- Controller output has no limits on its magnitude
- Actuator may not be able to “follow” controller output
  - Fundamental limitation, due to e.g., power and physical constraints
  - Further demands from controller output have no effect



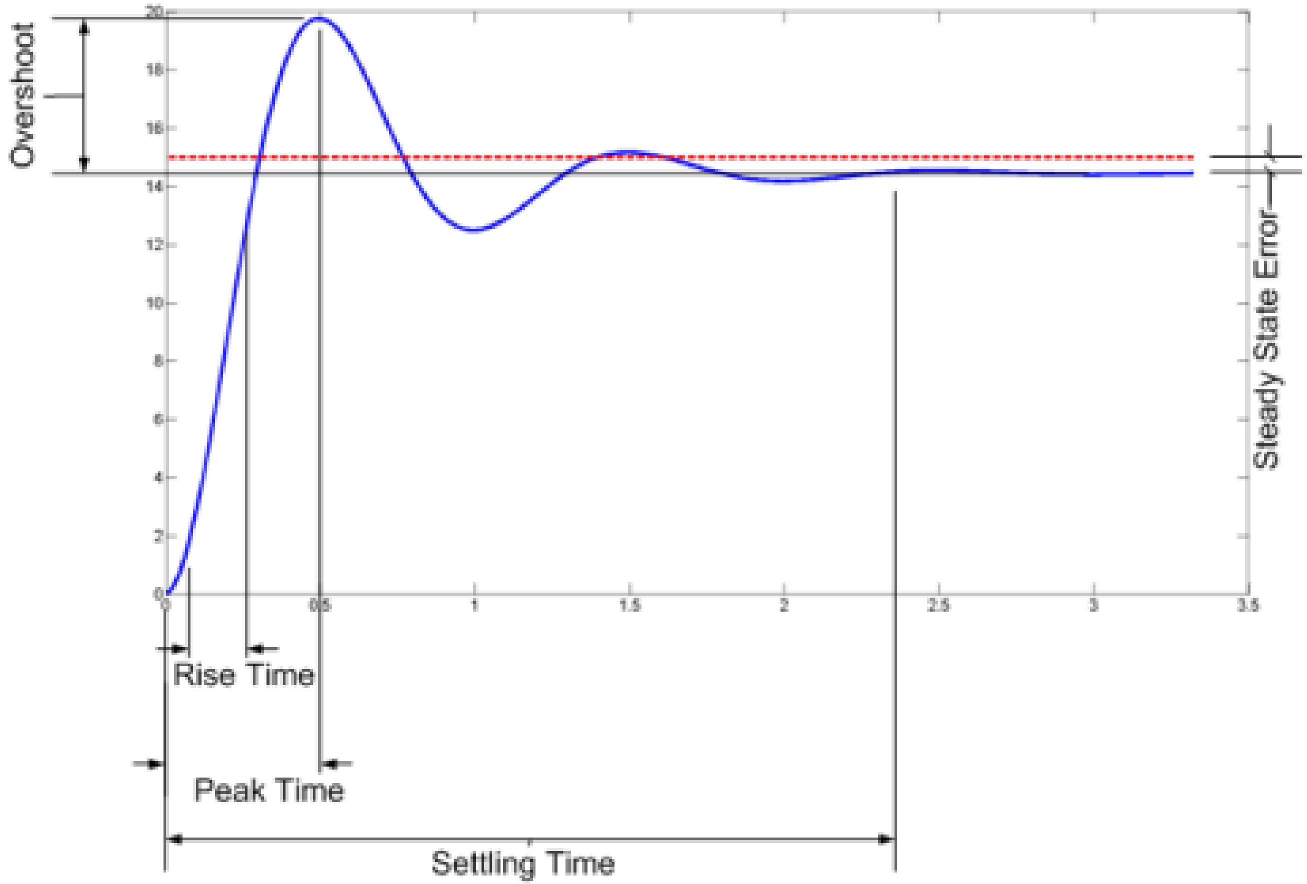
- With integral control, the accumulated error and control output can be very large, causing significant overshoot
  - Integral windup

More on https://www.oreilly.com/library/view/feedback-control-for/9781449362638/ch10.html

Digital Realization

```
P = Kp * error;                       // Proportional control
dI = Ki * error * T;                  // Integral control
D = (Kd/T)*(error - prev_error);      // Derivative control
control = P + (I + dI)  + D;          // PID control signal c(t)
prev_error = error;                   // Update previous error sample

// Enforce output limits
if  (control > OUTPUT_MAX)
    control = OUTPUT_MAX;
else if  (control < OUTPUT_MIN)
    control = OUTPUT_MIN;
else
    I += dI;                          // Accumulate error signal only if controller output
                                      // falls within [OUTPUT_MIN, OUTPUT_MAX]
```

PI Control Tuning (1)



- Rise time: time taken for m(t) to go from 10% to 90% of its steady value
- Overshoot:

(max value – steady value)/steady value *100%

- Settling time: time taken for m(t) to be bounded within a tolerance of say, 2% of its steady value
- Steady-state error: difference between the steady value of m(t) and desired response x(t)

From http://www.mee.tcd.ie/~corrigad/3c1/control_ho2_2012_students.pdf

PI Control Tuning (2)

Effects of Increasing Gains

	Rise Time	Overshoot	Settling Time	Steady-state Error
Proportional control gain k_p	Decrease	Increase	Small change	Decrease
Integral control gain k_i	Decrease	Increase	Increase	Eliminate