CS122A: Intermediate Embedded and Real Time Operating Systems

Jeffrey McDaniel

University of California, Riverside

Introduction to Control Systems

- A **control system** is an embedded system that regulates the behavior of a physical device

Introduction to Control Systems

- A **control system** is an embedded system that regulates the behavior of a physical device
- It attempts to match a system value to a desired value

Introduction to Control Systems

- A **control system** is an embedded system that regulates the behavior of a physical device
- It attempts to match a system value to a desired value
- Cruise control and water heaters are common examples

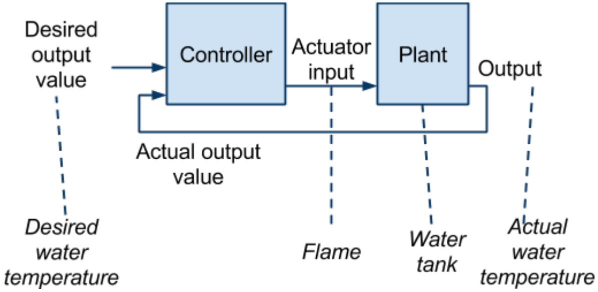
Introduction to Control Systems

- A **control system** is an embedded system that regulates the behavior of a physical device
- It attempts to match a system value to a desired value
- Cruise control and water heaters are common examples
- The **plant** is the device being controlled

Introduction to Control Systems

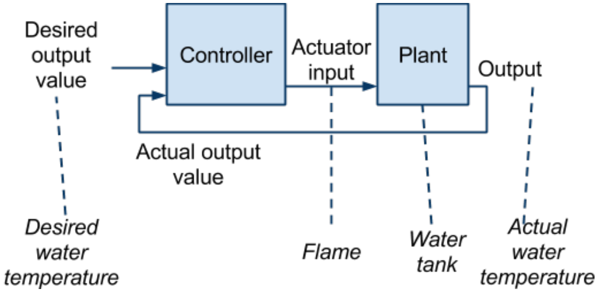
- A **control system** is an embedded system that regulates the behavior of a physical device
- It attempts to match a system value to a desired value
- Cruise control and water heaters are common examples
- The **plant** is the device being controlled
- The **controller** takes feedback and controls the plant

Introduction to Control Systems



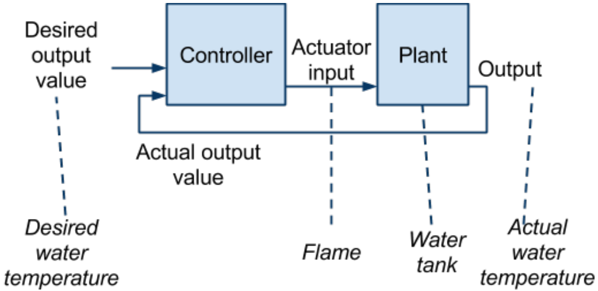
-

Introduction to Control Systems



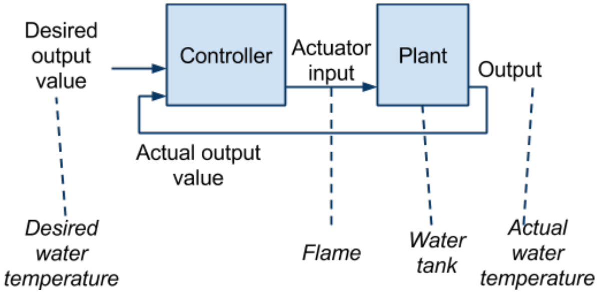
- The plant (*water tank*) outputs the actual value (*actual water temperature*) which is the value to be regulated

Introduction to Control Systems



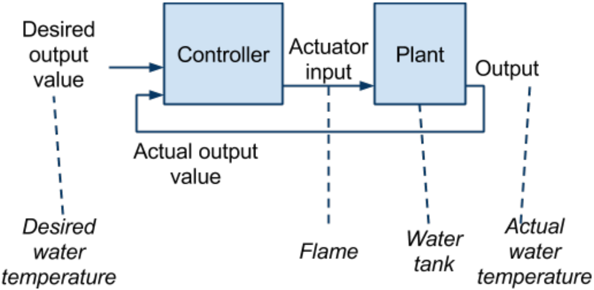
- The plant (*water tank*) outputs the actual value (*actual water temperature*) which is the value to be regulated
- This actual value is used as **feedback** to the controller

Introduction to Control Systems



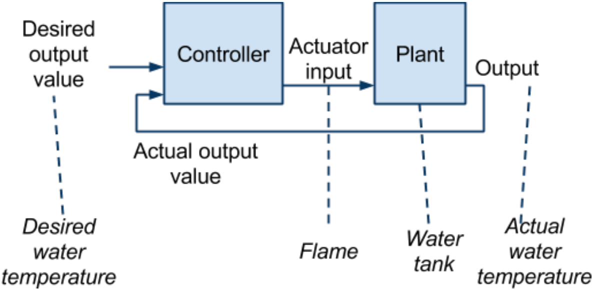
- The plant (*water tank*) outputs the actual value (*actual water temperature*) which is the value to be regulated
- This actual value is used as **feedback** to the controller
- The controller uses the actuator input (flame) to alter the output

Introduction to Control Systems



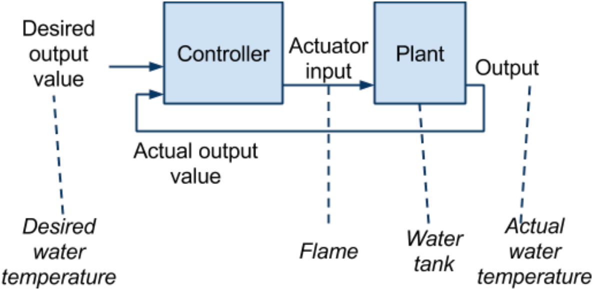
- The plant (*water tank*) outputs the actual value (*actual water temperature*) which is the value to be regulated
- This actual value is used as **feedback** to the controller
- The controller uses the actuator input (flame) to alter the output
- The controller also has a desired output value (*desired water temperature*) as input

Introduction to Control Systems



- The **error** is the difference between the desired and actual value (*desired* - *actual*)

Introduction to Control Systems



- The **error** is the difference between the desired and actual value (*desired - actual*)
- The job of the controller is to reduce this error to zero by altering the actuator input based on the feedback

Water Heater Example

Water Heater Controller

- Actuator (gas valve): 8 bit value (0-200) (B)
- Desired temperature: 4 bit value (0-100°C/ 10) (A3:0)
- Actual temperature: 4 bit value (0-100°C/ 10) (A7:4)

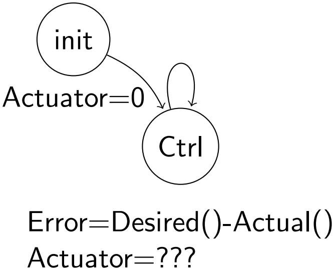
```c
#define Actuator B
unsigned char Desired(){
    return (A & 0x0F);
}
unsigned char Actual(){
    return (A & 0xF0) >> 4;
}
signed char Error;
```

Water Heater Example

Water Heater Controller

- Actuator (gas valve): 8 bit value (0-200) (B)
- Desired temperature: 4 bit value (0-100°C/ 10) (A3:0)
- Actual temperature: 4 bit value (0-100°C/ 10) (A7:4)

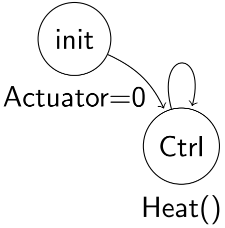
```
#define Actuator B
unsigned char Desired(){
    return (A & 0x0F);
}
unsigned char Actual(){
    return (A & 0xF0) >> 4;
}
signed char Error;
```



Water Heater Example

Water Heater Controller

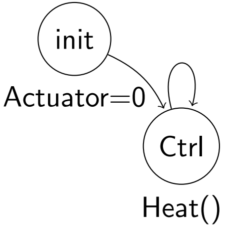
- Actuator (gas valve): 8 bit value (0-200) (B)
- Desired temperature: 4 bit value (0-100°C/ 10) (A3:0)
- Actual temperature: 4 bit value (0-100°C/ 10) (A7:4)



Water Heater Example

Water Heater Controller

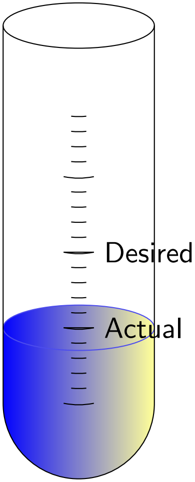
- Actuator (gas valve): 8 bit value (0-200) (B)
- Desired temperature: 4 bit value (0-100°C/ 10) (A3:0)
- Actual temperature: 4 bit value (0-100°C/ 10) (A7:4)




```
unsigned char Heat(){
  if (Desired() < Actual()) {
    Actuator = 200;
  }
}
```

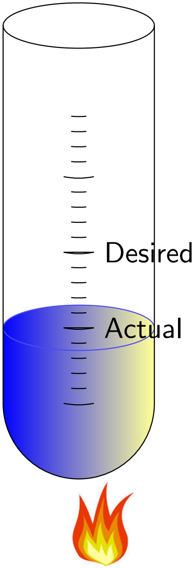
Control Systems

- **On-off control** is the simplest method



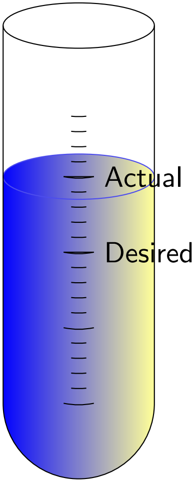
Control Systems

- **On-off control** is the simplest method
- Turn on actuator if `Desired() < Actual()`



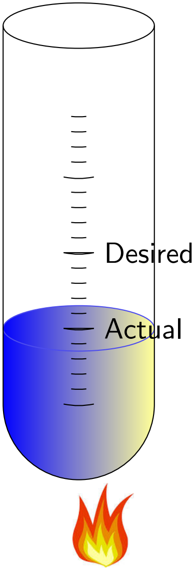
Control Systems

- **On-off control** is the simplest method
- Turn on actuator if `Desired() < Actual()`
- Actual may **overshoot** desired



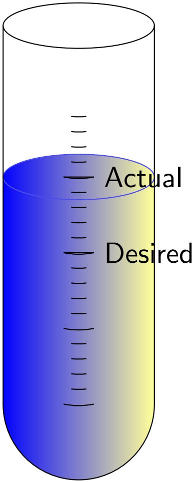
Control Systems

- **On-off control** is the simplest method
- Turn on actuator if `Desired() < Actual()`
- Actual may **overshoot** desired
- Turning off actuator lowers the actual temperature



Control Systems

- **On-off control** is the simplest method
- Turn on actuator if `Desired() < Actual()`
- Actual may **overshoot** desired
- Turning off actuator lowers the actual temperature
- Actual temperature will **oscillate** around desired temperature



Control Systems

- The time it takes to bring the lower actual value to the desired value is the **rise time**

Control Systems

- The time it takes to bring the lower actual value to the desired value is the **rise time**
- Too slow a rise time is not desirable, but overshooting is also not desirable

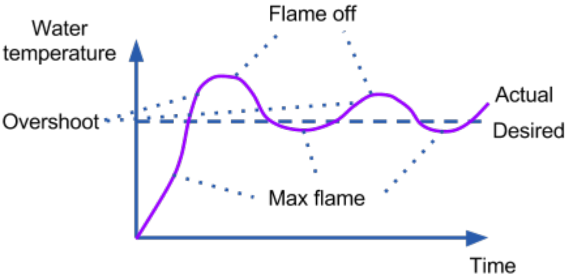
Control Systems

- The time it takes to bring the lower actual value to the desired value is the **rise time**
- Too slow a rise time is not desirable, but overshooting is also not desirable
- Oscillation is also not desirable, although it is inevitable

Control Systems

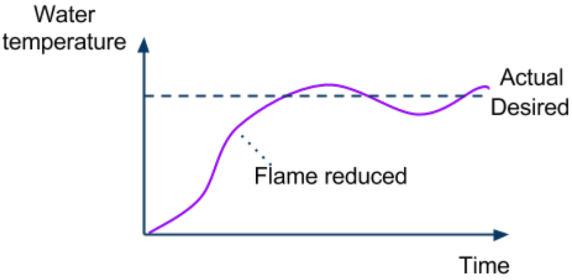
- The time it takes to bring the lower actual value to the desired value is the **rise time**
- Too slow a rise time is not desirable, but overshooting is also not desirable
- Oscillation is also not desirable, although it is inevitable
- A system with oscillation amplitude *increasing* over time is said to be **unstable**

Proportional Control



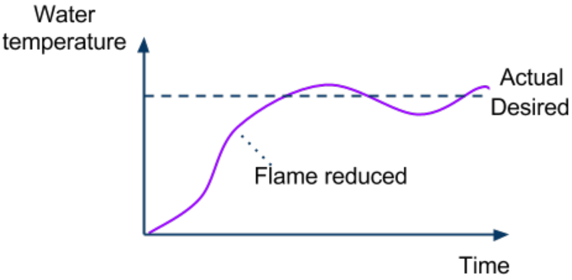
- Overshoot needs to be avoided, rise time needs to be quick, and oscillation needs to be minimized

Proportional Control



- Overshoot needs to be avoided, rise time needs to be quick, and oscillation needs to be minimized
- In a **proportional controller** the actuator (flame) is set *proportionally* to the error.
  $Actuator = Kp * Error$

Proportional Control



- Overshoot needs to be avoided, rise time needs to be quick, and oscillation needs to be minimized
- In a **proportional controller** the actuator (flame) is set *proportionally* to the error.
  $Actuator = Kp * Error$
- Actuator is reduced as actual approaches desired value

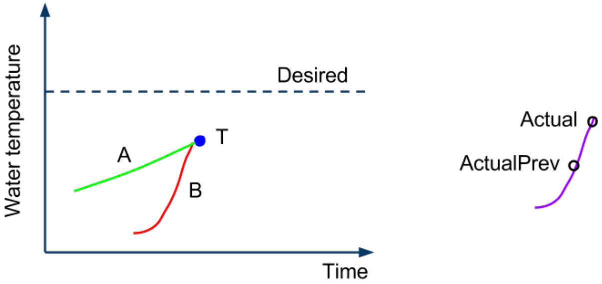
Proportional-Derivative (PD) Control

- Proportional control does not take into account the *rate of change* of the output value.

Proportional-Derivative (PD) Control

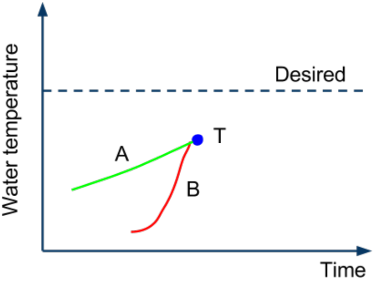
- Proportional control does not take into account the *rate of change* of the output value.
- Problems of overshoot, oscillation, and slow rise time can be overcome even further using a **Proportional-Derivative (PD) controller**

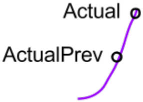
Proportional-Derivative (PD) Control



- The rate of change is used to calculate the actuator value

Proportional-Derivative (PD) Control

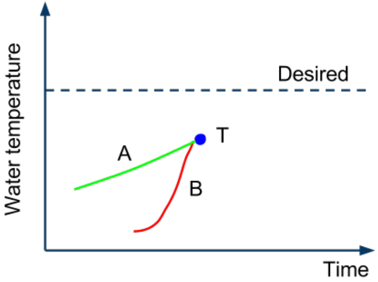


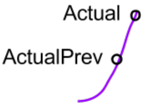


- The rate of change is used to calculate the actuator value

$$Actuator = K_p * Error - K_d * Deriv$$

Proportional-Derivative (PD) Control



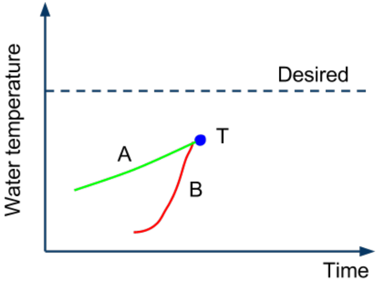


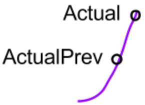
- The rate of change is used to calculate the actuator value

$$Actuator = K_p * Error - K_d * Deriv$$

$$f(x) = e^x + log_{10^{x^2}} + x^2$$

Proportional-Derivative (PD) Control





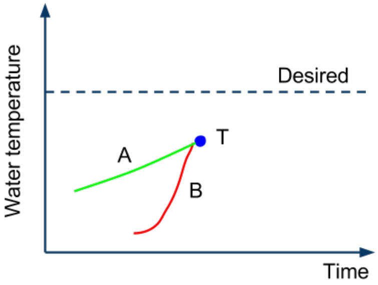
- The rate of change is used to calculate the actuator value

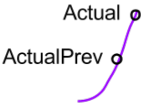
$$Actuator = K_p * Error - K_d * Deriv$$

$$f(x) = e^x + log_{10^{x^2}} + x^2$$

$$Deriv = f'(x) = ?$$

Proportional-Derivative (PD) Control





- The rate of change is used to calculate the actuator value

$$Actuator = K_p * Error - K_d * Deriv$$

$$Deriv = Actual - ActualPrev$$

Proportional-Integral-Derivative (PID) Control

- Control systems can suffer from **steady-state error** where the actual output value never reaches the desired value

Proportional-Integral-Derivative (PID) Control

- Control systems can suffer from **steady-state error** where the actual output value never reaches the desired value
- Steady-state error can be approximated using the sum of past error values

Proportional-Integral-Derivative (PID) Control

- Control systems can suffer from **steady-state error** where the actual output value never reaches the desired value
- Steady-state error can be approximated using the sum of past error values
- This sum can be computed as the area under the curve, or the plots integral

Proportional-Integral-Derivative (PID) Control

- Control systems can suffer from **steady-state error** where the actual output value never reaches the desired value
- Steady-state error can be approximated using the sum of past error values
- This sum can be computed as the area under the curve, or the plots integral

$$Actuator = K_p * Error + K_i * Integ - K_d * Deriv$$

With $K_i << K_p$ typically

Proportional-Integral-Derivative (PID) Control

- Control systems can suffer from **steady-state error** where the actual output value never reaches the desired value
- Steady-state error can be approximated using the sum of past error values
- This sum can be computed as the area under the curve, or the plots integral

$$Actuator = K_p * Error + K_i * Integ - K_d * Deriv$$

  With $K_i << K_p$ typically
- Limits must be set on these values based on the actuator abilities

Proportional-Integral-Derivative (PID) Control

- Control systems can suffer from **steady-state error** where the actual output value never reaches the desired value
- Steady-state error can be approximated using the sum of past error values
- This sum can be computed as the area under the curve, or the plots integral

  $$Actuator = K_p * Error + K_i * Integ - K_d * Deriv$$

  With $K_i << K_p$ typically
- Limits must be set on these values based on the actuator abilities
- These values are typically calculated using floating point numbers, which are computationally intensive, integer approximations can sometimes be acceptable