6.270 Lecture

Control Systems

Steven Jorgensen
Massachusetts Institute of Technology
January 2014

Overview of Lecture

- Feed Forward Open Loop Controller
  - Pros and Cons
- Bang-Bang Closed Loop Controller
- Intro to PID Closed Loop Control
  - Proportional Control
  - Proportional-Integral Control
  - Proportional-Derivative Control
  - Proportional-Integral-Derivative Control
- Personal Tips and Suggestions

Feed Forward / Open Loop Controller

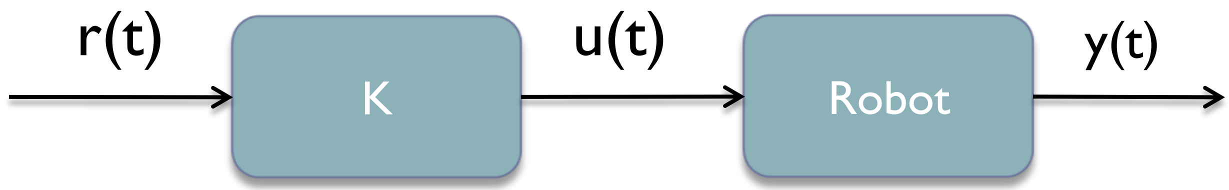
Examples

- Room Lightswitch
- Water Faucet
- Time Based Toaster

Robot Examples

- Set Stock Servo Motor Position
- Setting PWM on motor

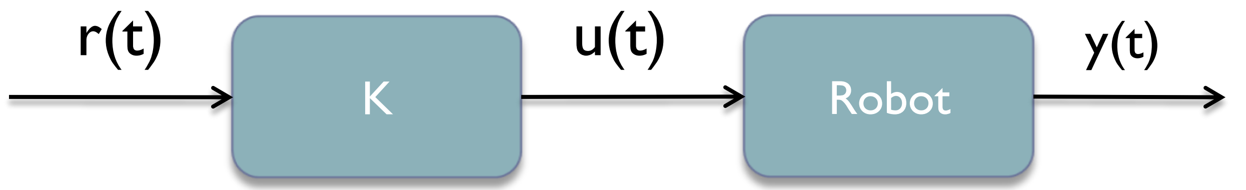
Feed Forward / Open Loop Controller



r(t) – Reference Input (eg: Voltage, Desired Angle/Speed)
K – Gain (Some Units)
u(t) – Controller Output
y(t) - Plant Output (eg: Actual Angle/Speed)

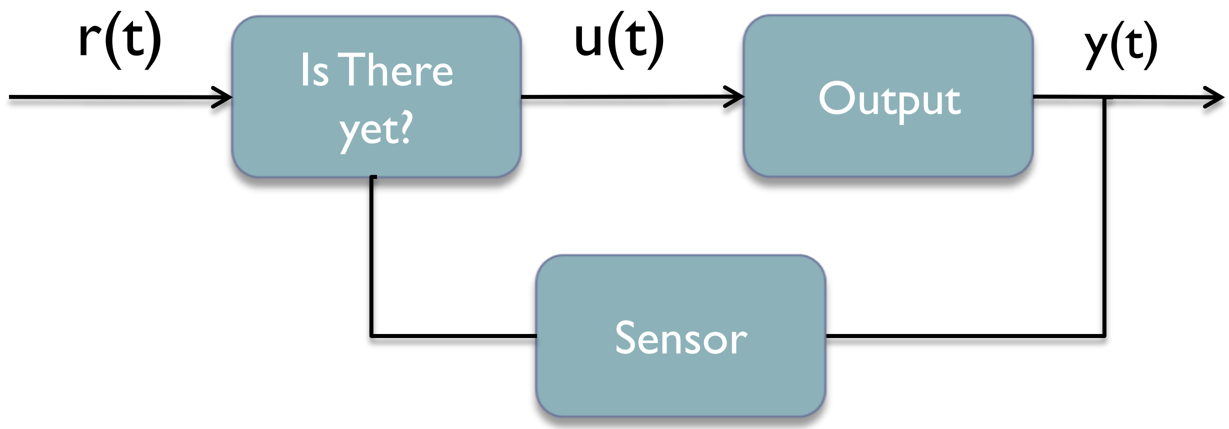
- Given a reference input, multiply it by a gain, K.
- Apply controller output to robot

Feed Forward / Open Loop Controller



- Pros
  - Simple to Implement
  - If everything is known, controller can be reliable
- Cons:
  - Robot does not know if the desired output is reached
  - Not robust under variable loading

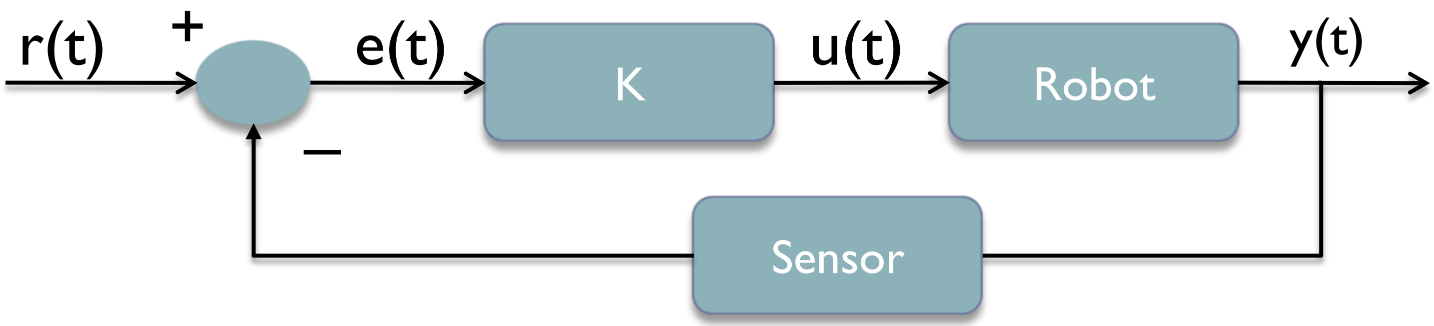
Bang-Bang Closed Loop Controller



u(t) – On/OFF

- Example: Thermostat (ON until warm temperature is reached)
- Sensor Information Dictates Action
- Not good for navigation

Closed Loop Control: Proportional Controller

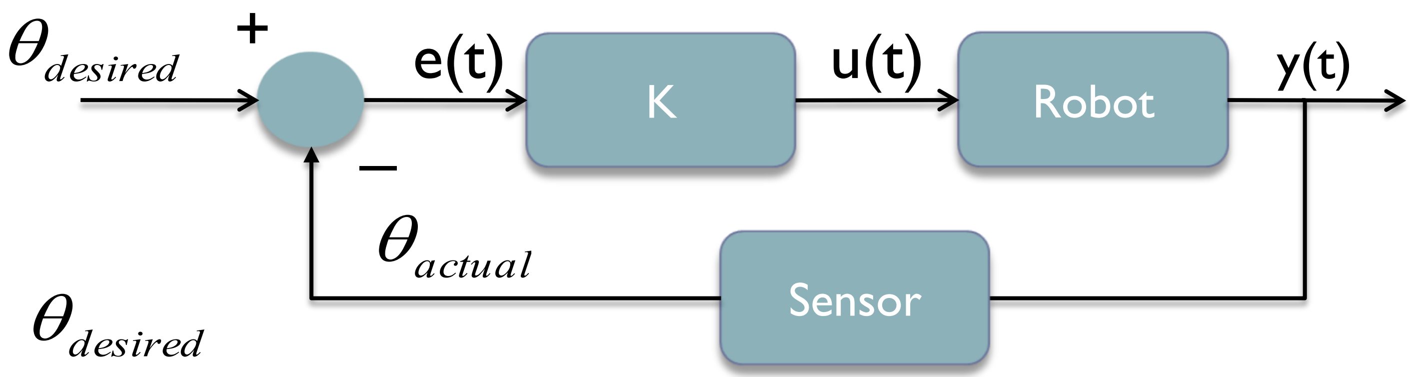


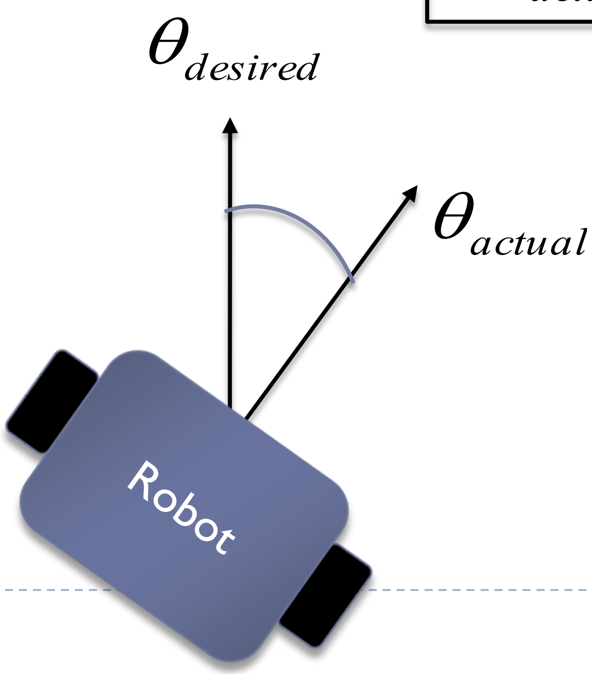
$$e(t) = r(t) - y(t)$$

- Use sensor information to dictate action
- Use negative feedback to minimize error e(t)

Proportional Controller Example

- Suppose we want a robot to turn to a particular angle





$$r(t) = \theta_{desired}$$

$$y(t) = \theta_{actual}$$

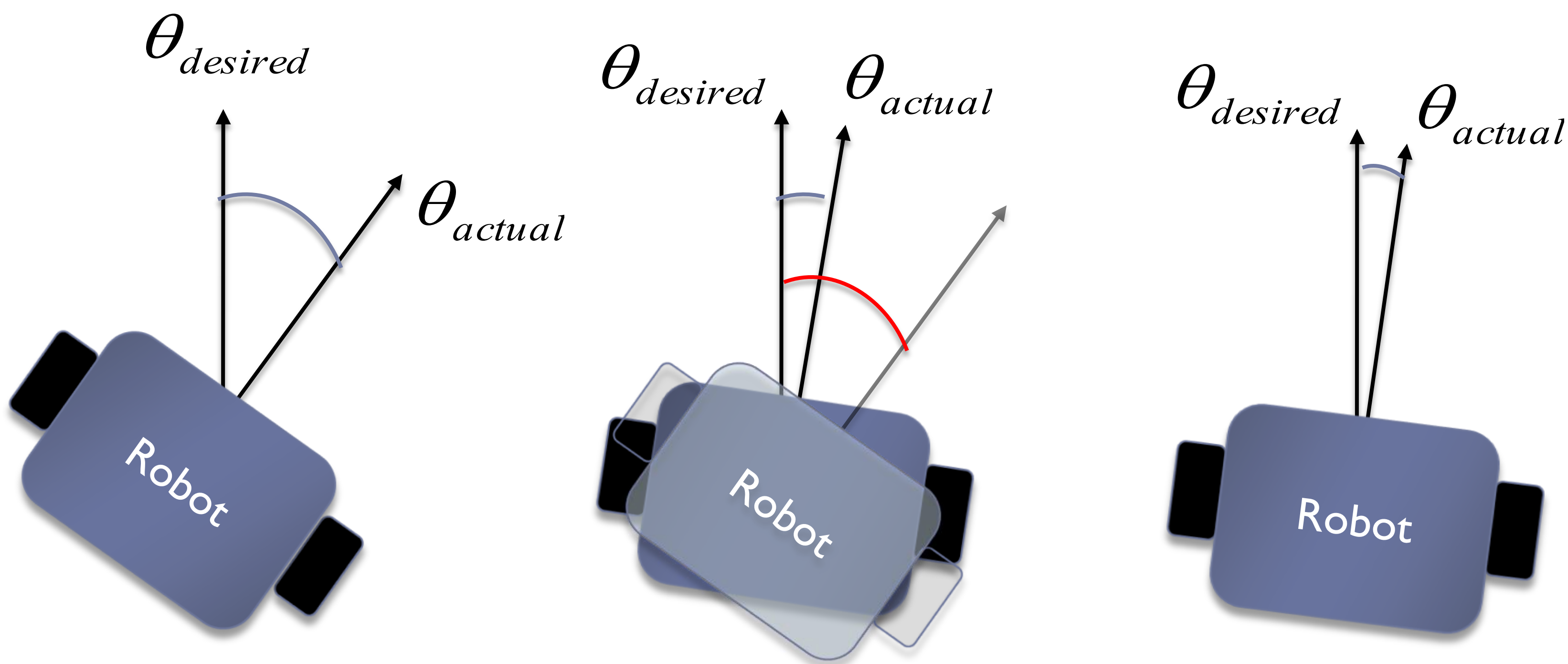
$$e(t) = \theta_{desired} - \theta_{actual}$$

$$u(t) = k(\theta_{desired} - \theta_{actual})$$

Proportional Controller Example

- Remember, we are trying to minimize e(t). Suppose k = 10
- As error increases, u(t) puts more effort to achieve desired angle
- As error decreases, u(t) puts less effort to achieve desired angle

$$e(t) = \theta_{desired} - \theta_{actual} \qquad u(t) = k(\theta_{desired} - \theta_{actual})$$



$$e = (0 - \pi/4) \qquad u = 10(-\pi/4) \qquad u = 10(-\pi/6)$$

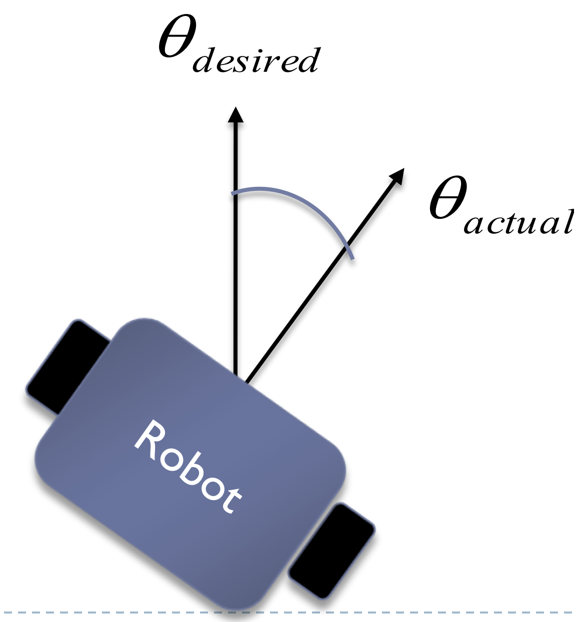
Issues with Proportional Controller

- Tuning the gain K:
  - We want high K to reach desired output but…
  - K can't be too "stiff" (Overshoots like an undamped spring)
  - K can't be too "soft" (K is too small to move the robot)
- Thus, desired output is **never** reached

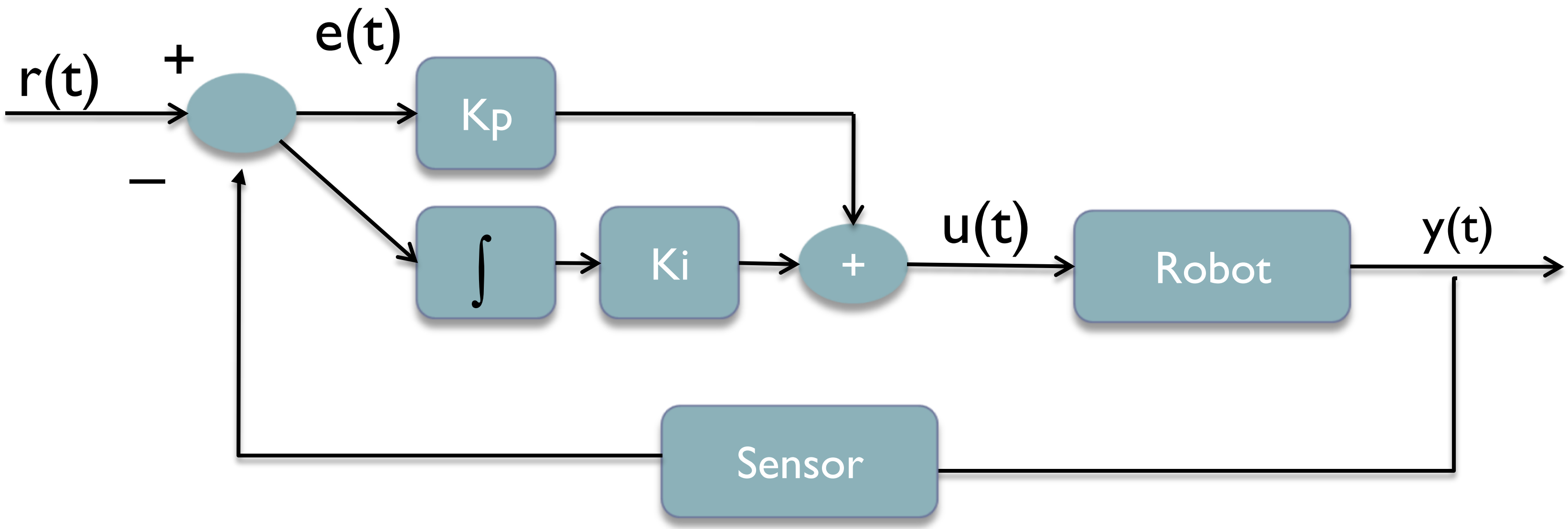
$$u(t) = k(\theta_{desired} - \theta_{actual})$$

$$u(t) = k\Delta x$$

Looks like Hooke's Law…



Proportional-Integral (PI) Controller

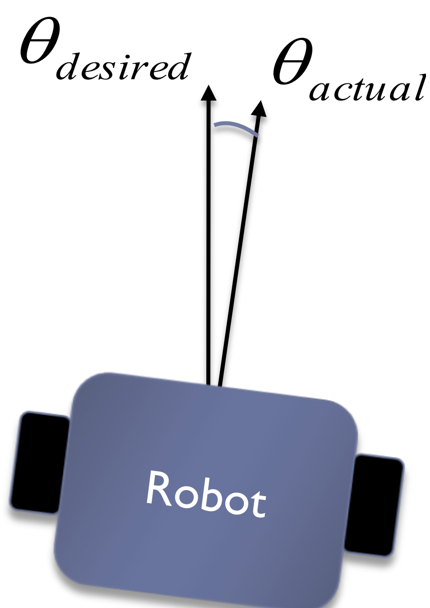


$$u(t) = k_p e(t) + k_i \int e(t)$$

- *"Integral" Essential Concept:* **Accumulate all error and get rid of it**
- Add a integral term to get rid of error

PI Controller Example

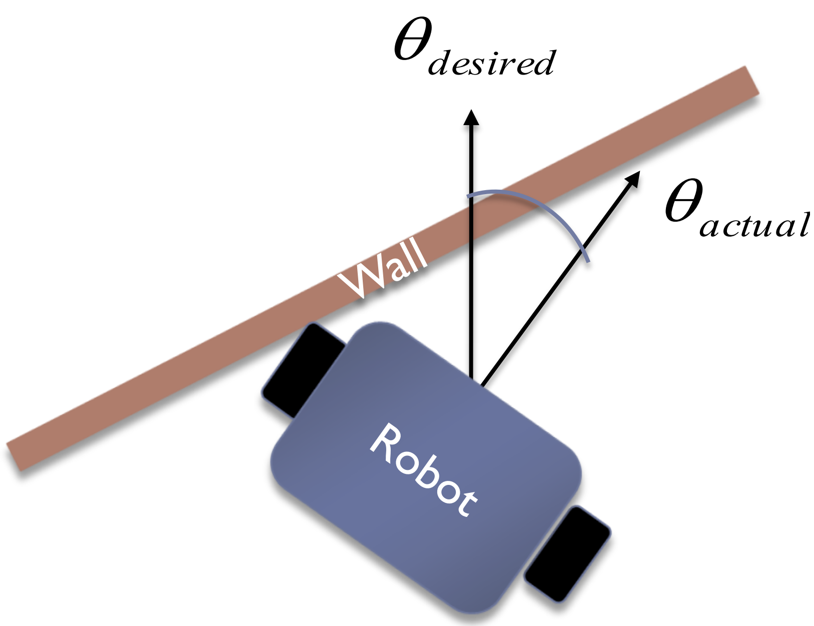
- *"Integral" Essential Concept:* **Accumulate all error and get rid of it**
- Integral will accumulate both positive and negative error
  - This slowly disappears given appropriate values of ki



$$u(t) = k_p e(t) + k_i \int e(t)$$

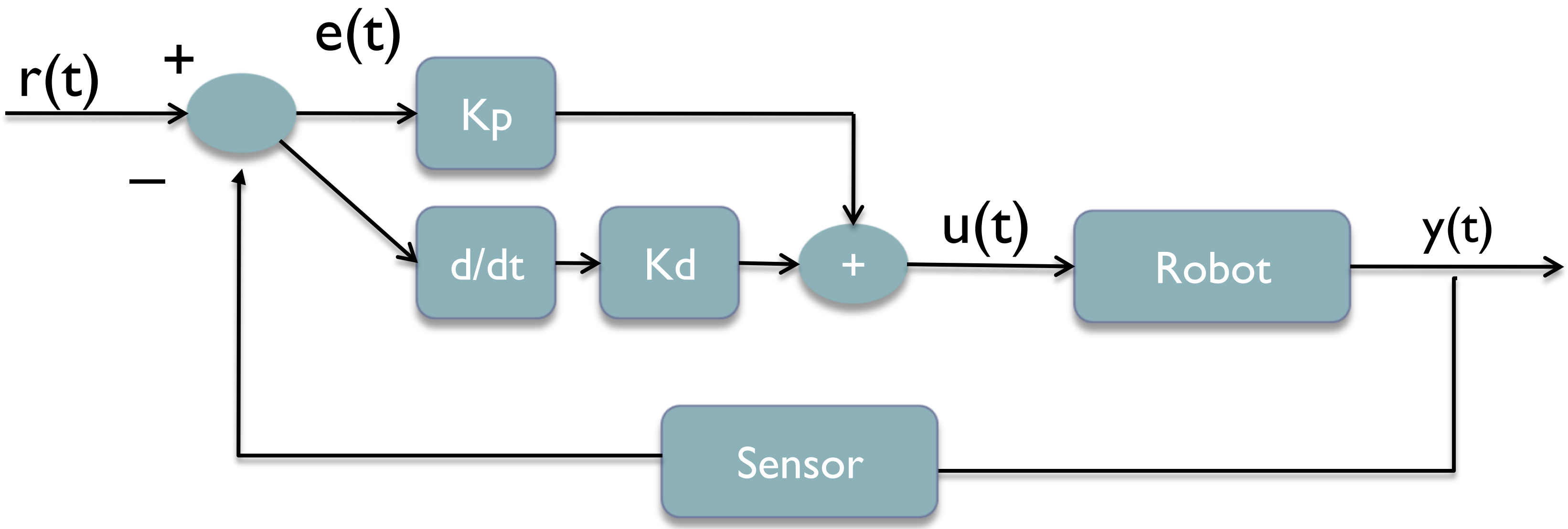
PI Controller Issue

- Integral term can **wind up** forever.
- Suppose robot wants to reach a certain desired angle, but something is blocking it.
  - What happens to integral term? How might you solve this?



$$u(t) = k_p e(t) + k_i \int e(t)$$

Proportional-Derivative (PD) Controller



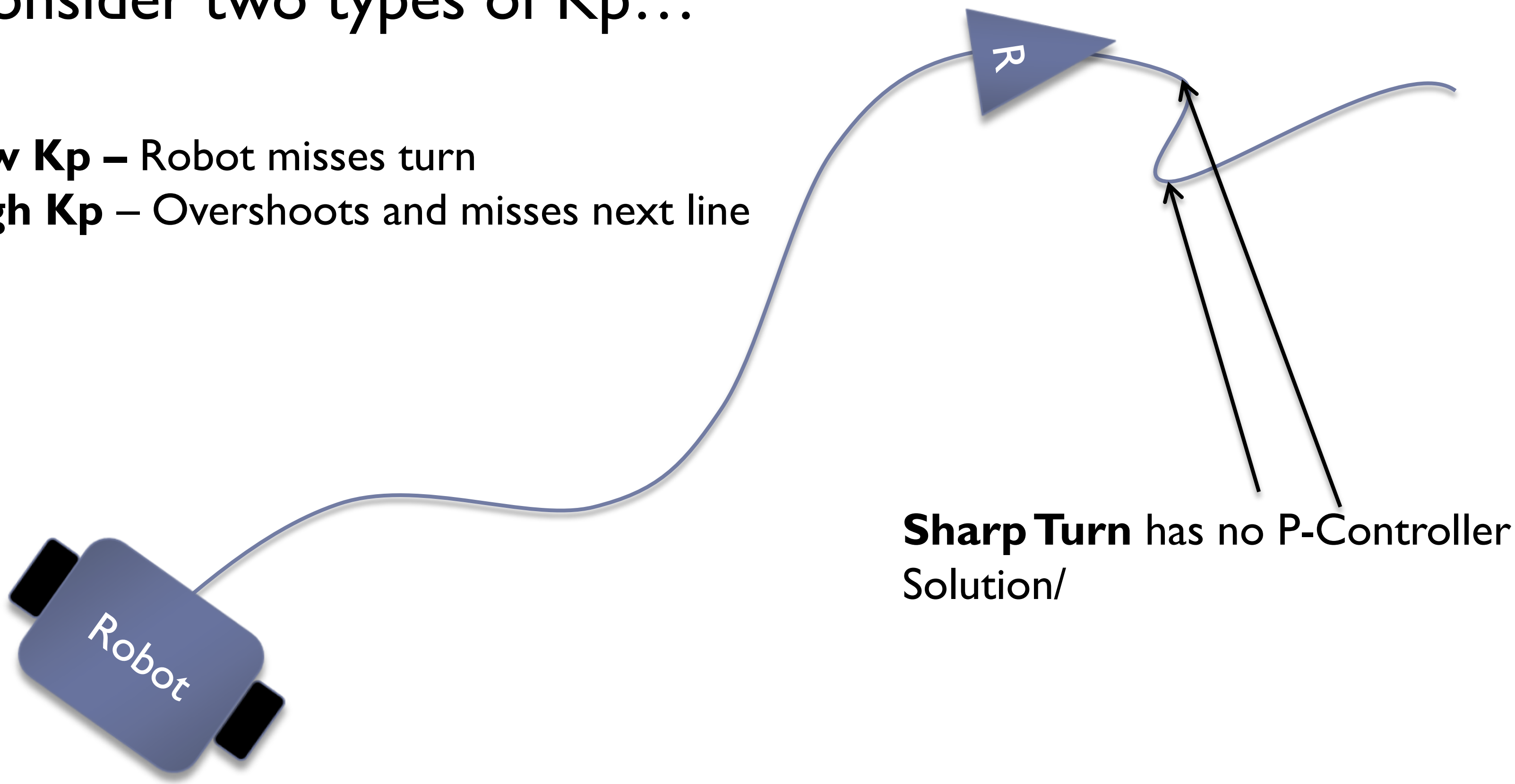
$$u(t) = k_p e(t) + k_d \frac{d}{dt} e(t)$$

- *"Derivative" Essential Concept:* **If controller didn't change fast enough, apply some extra effort**
- Add a derivative term to act as a damper

PD Control Example

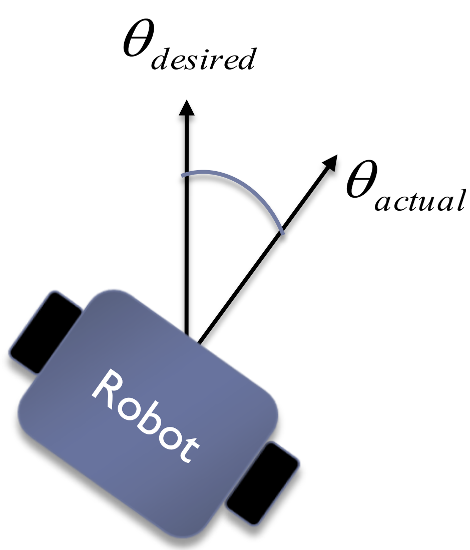
- Suppose you are making a line-follower robot:
- Consider two types of Kp…

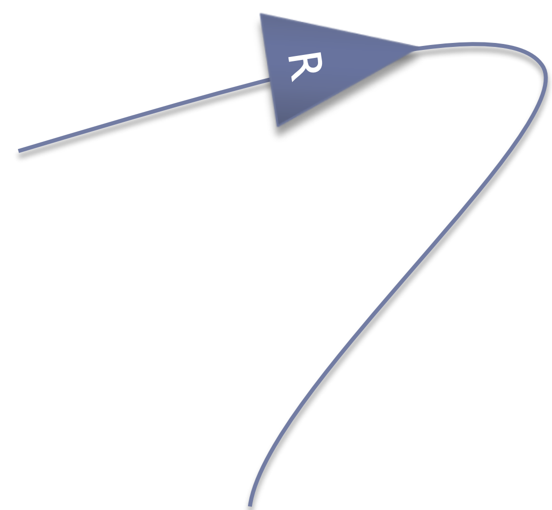
Low Kp – Robot misses turn
High Kp – Overshoots and misses next line



PD Control Example

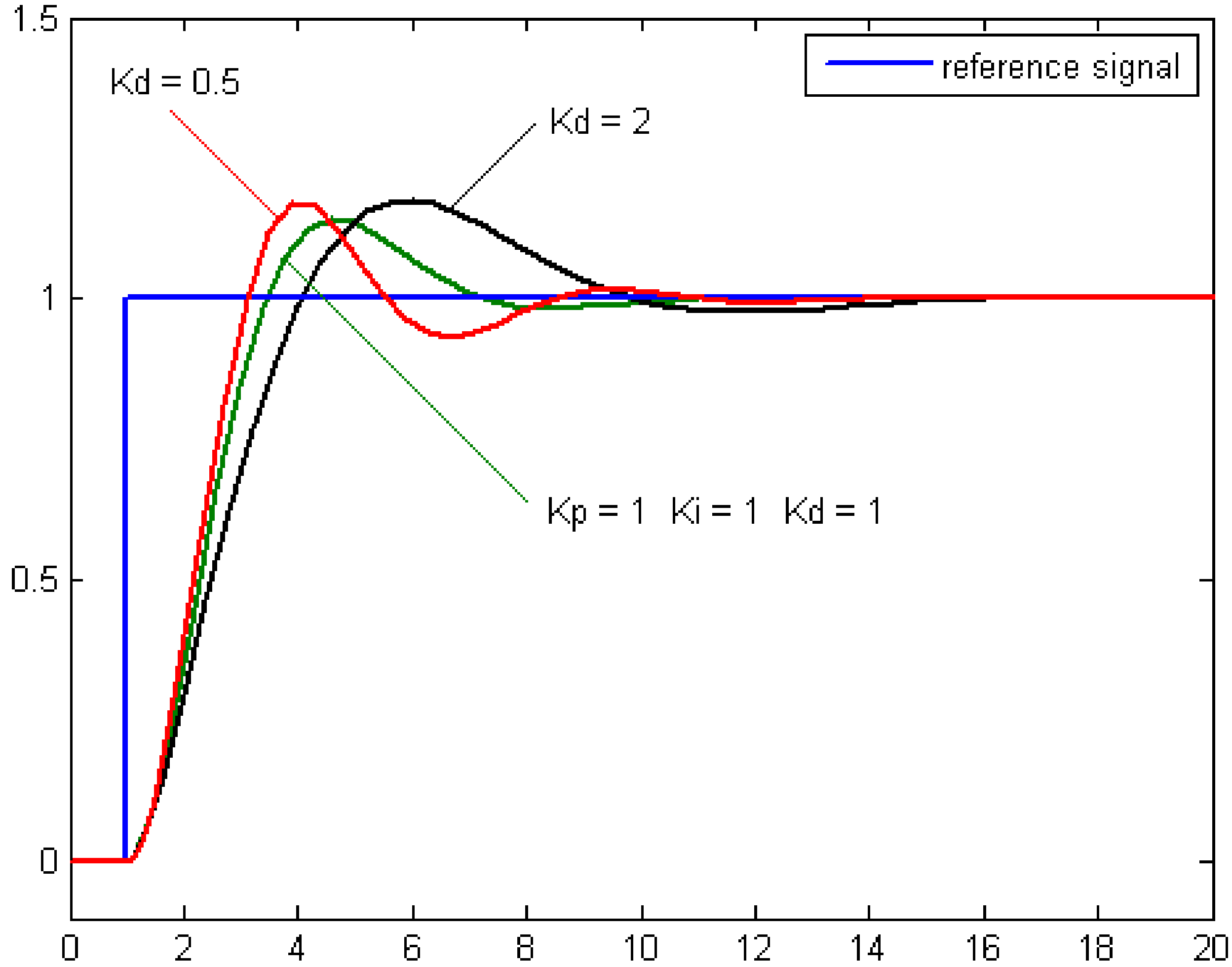
- Problem: We want a really high Kp to reach desired output
- Issue: Robot Overshoots
- Using "D-controller" as Damper





$$u(t) = k_p e(t) + k_d \frac{d}{dt} e(t)$$

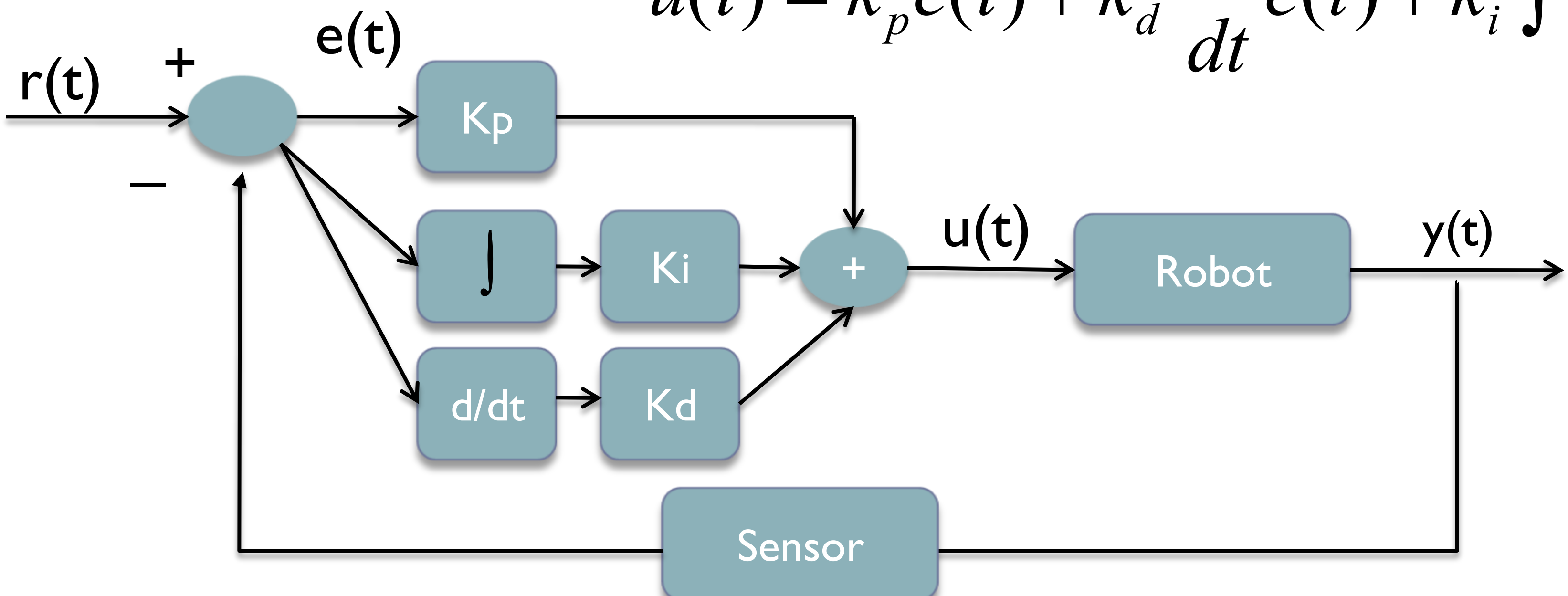
Solution: Use "D" on top of "P"



- Image from wikipedia: https://en.wikipedia.org/wiki/File:Change_with_Kd.png

Complete PID Controller

$$u(t) = k_p e(t) + k_d \frac{d}{dt} e(t) + k_i \int e(t)$$



- P – Proportional (Spring) term to reach goal
- I – Integral term to remove residual error
- D – Derivative term to add damping

Complete PID Tuning

- Do this smartly.
- Start with only P. Increase P until it Oscillates
- Include D until the system is critically damped (no oscillation)
- Include I to remove residual

For more tips on PID tuning check:

Ziegler-Nichols Tuning Method

https://en.wikipedia.org/wiki/Ziegler%E2%80%93Nichols_method

Personal Tips/Suggestion

- Robot has inherent damping
  - Don't bother with D
- Design with tolerances in mind
  - Don't bother with I.
  - In most cases I is overkill for simple applications
- P-Control is good enough.
- Tune P controller using potentiometer (frob knob) as gain