Introduction to Robotics

Manipulation and Programming

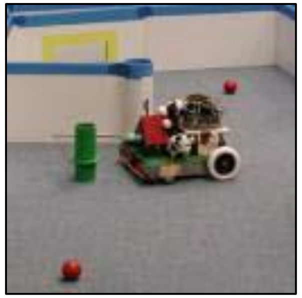
Unit 4: Motion Control

MOBILE ROBOT PART 4 – SYSTEM BUILDING: LOW LEVEL CONTROL LOOP

DR. ERIC CHOU IEEE SENIOR MEMBER

Objectives

- High-level control system paradigms
  - Model-Plan-Act Approach
  - Behavioral Approach
  - Finite State Machine Approach
- **Low-level control loops**
  - PID controller for motor velocity
  - PID controller for robot drive system


Low Level Control Loop

SECTION 1

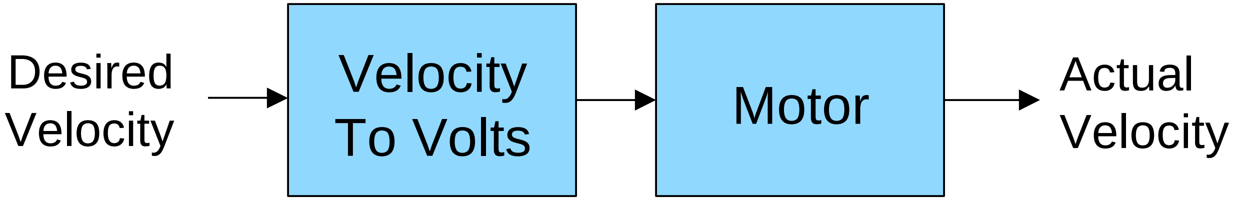
Control Loops

- Open Loops
- Closed Loops
- Proportional Controller
- Proportional-derivative Controller
- PID Controller

Problem: How do we set a motor to a given velocity?

Open Loop Controller

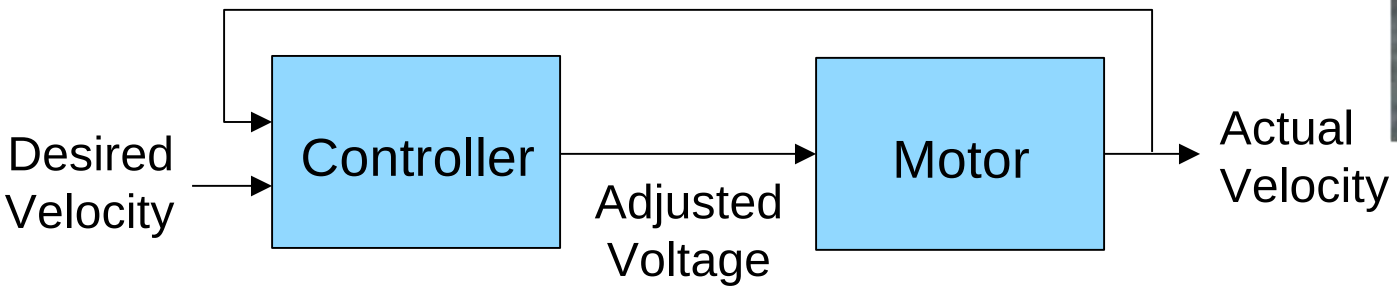
- Use trial and error to create some kind of relationship between velocity and voltage
- Changing supply voltage or drive surface could result in incorrect velocity



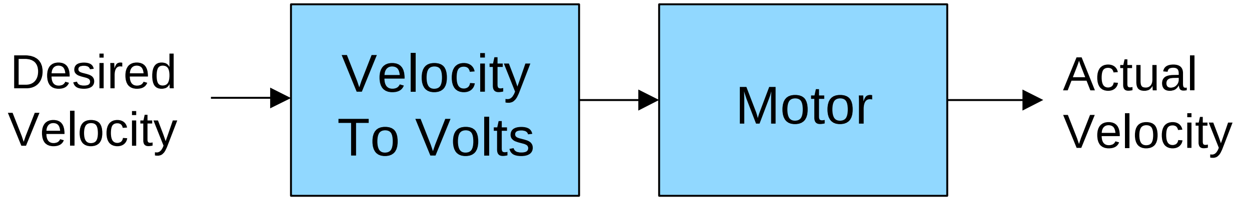
Problem: How do we set a motor to a given velocity?

Closed Loop Controller

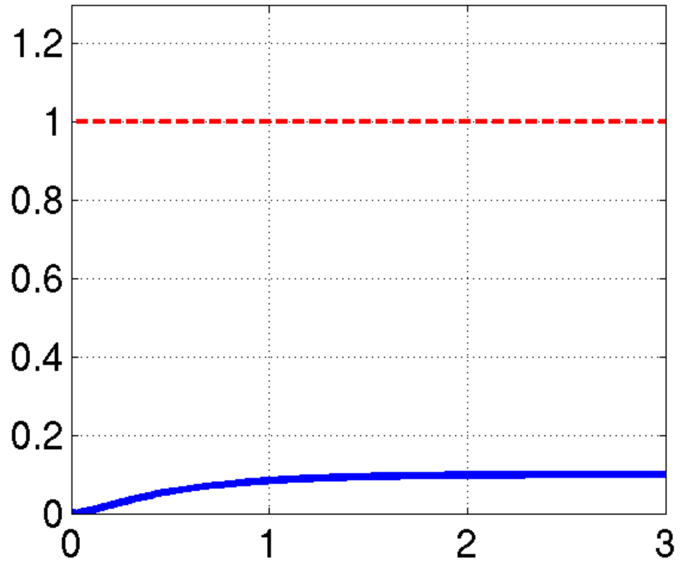
- Feedback is used to adjust the voltage sent to the motor so that the actual velocity equals the desired velocity
- Can use an optical encoder to measure actual velocity



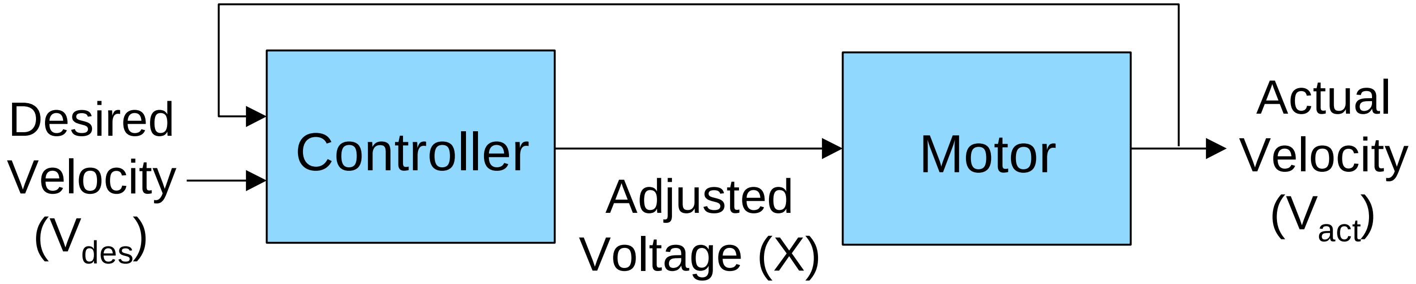
Step response with no controller



- Naive velocity to volts
- Model motor with several differential equations
- Slow rise time
- Stead-state offset

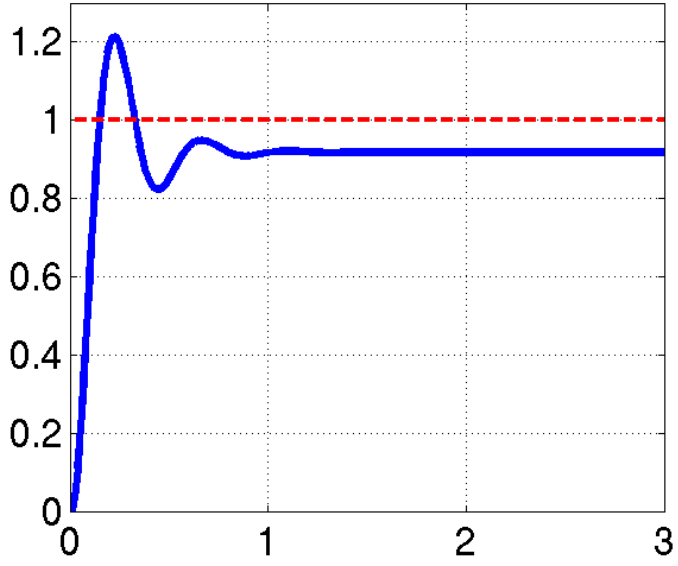


Step response with Proportional Controller

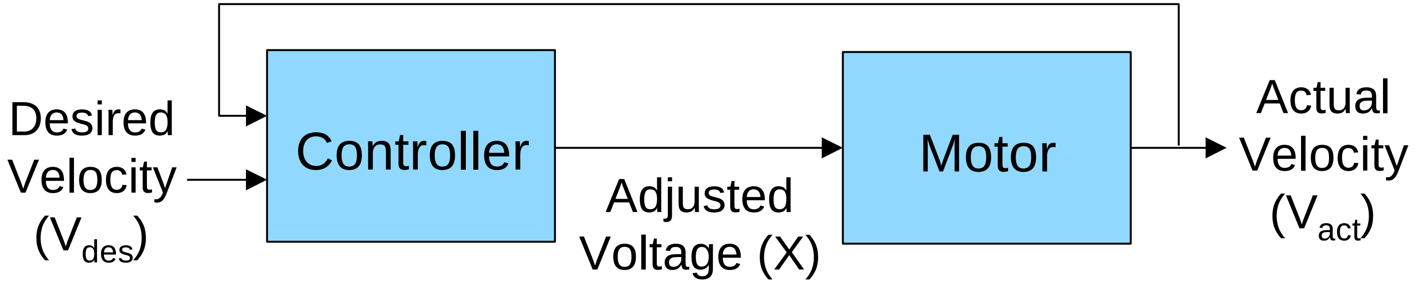


$$X = V_{des} + K_P \cdot (V_{des} - V_{act})$$

- Big error big = big adj
- Faster rise time
- Overshoot
- Stead-state offset

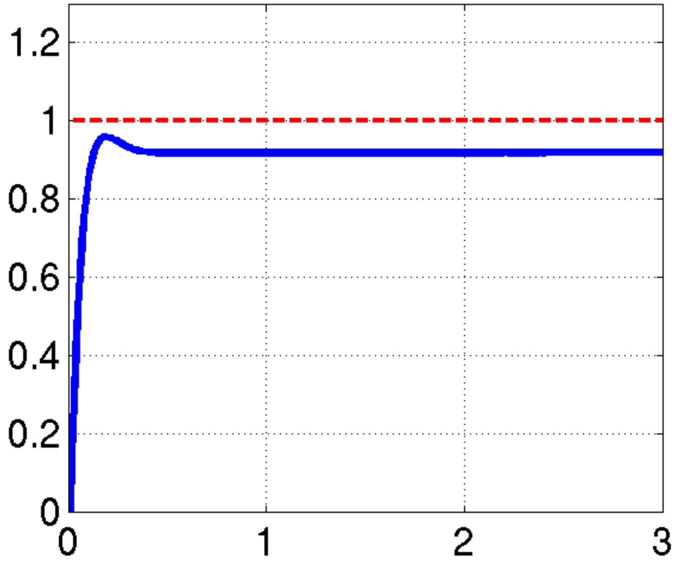


Step response with Proportional-derivative Controller

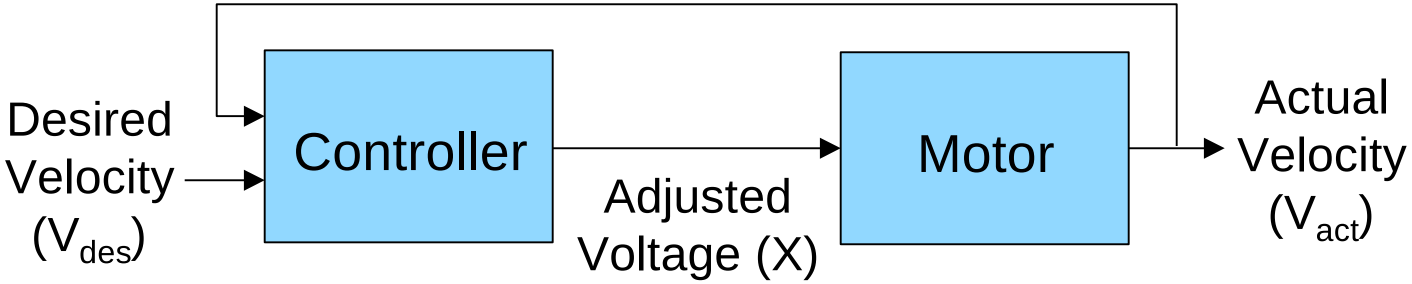


$$X = V_{des} + K_P e(t) - K_D \frac{de(t)}{dt}$$

- When approaching desired velocity quickly, de/dt term counteracts proportional term slowing adjustment
- Faster rise time
- Reduces overshoot

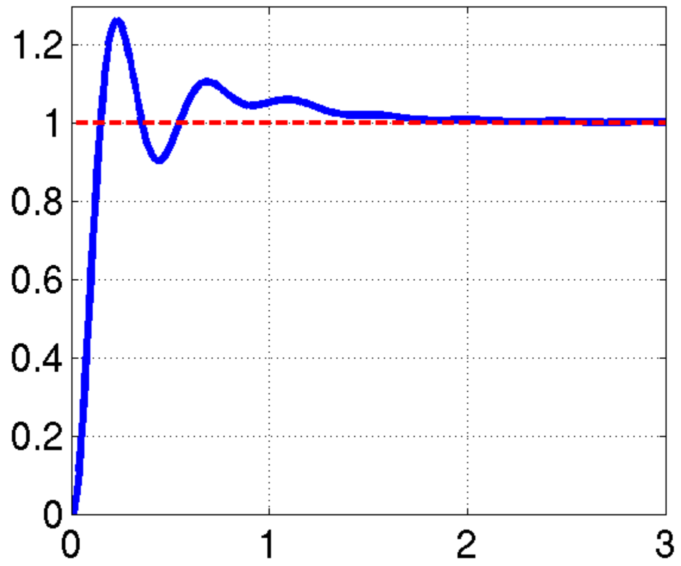


Step response with Proportional-Integral Controller (PID)

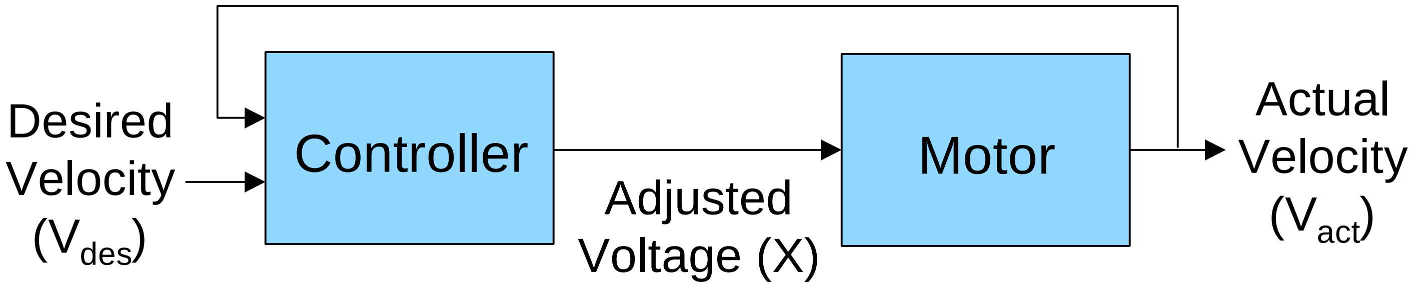


$$X = V_{des} + K_P e(t) - K_I \int e(t)\, dt$$

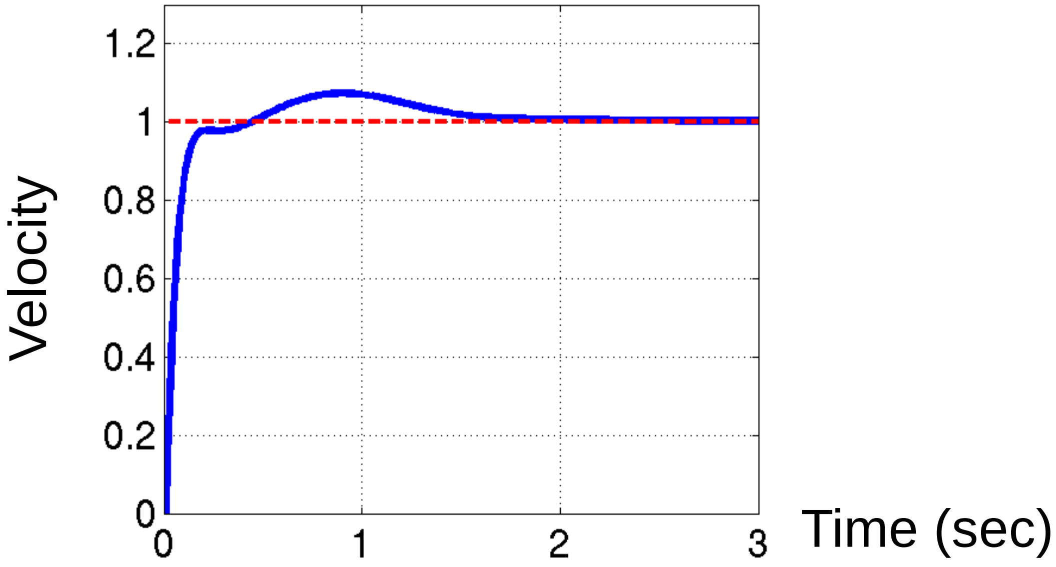
- Integral term eliminates accumulated error
- Increases overshoot



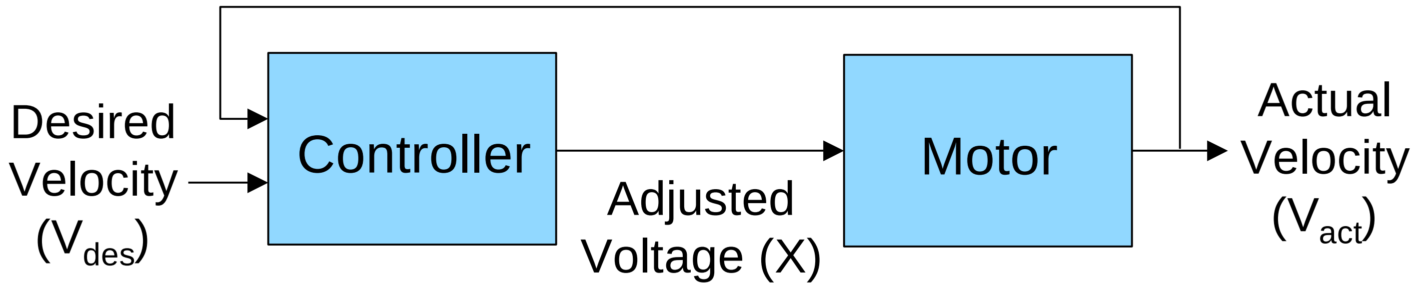
Step response with PID controller



$$X = V_{des} + K_P e(t) + K_I \int e(t)\, dt - K_D \frac{de(t)}{dt}$$



Choosing and tuning a controller

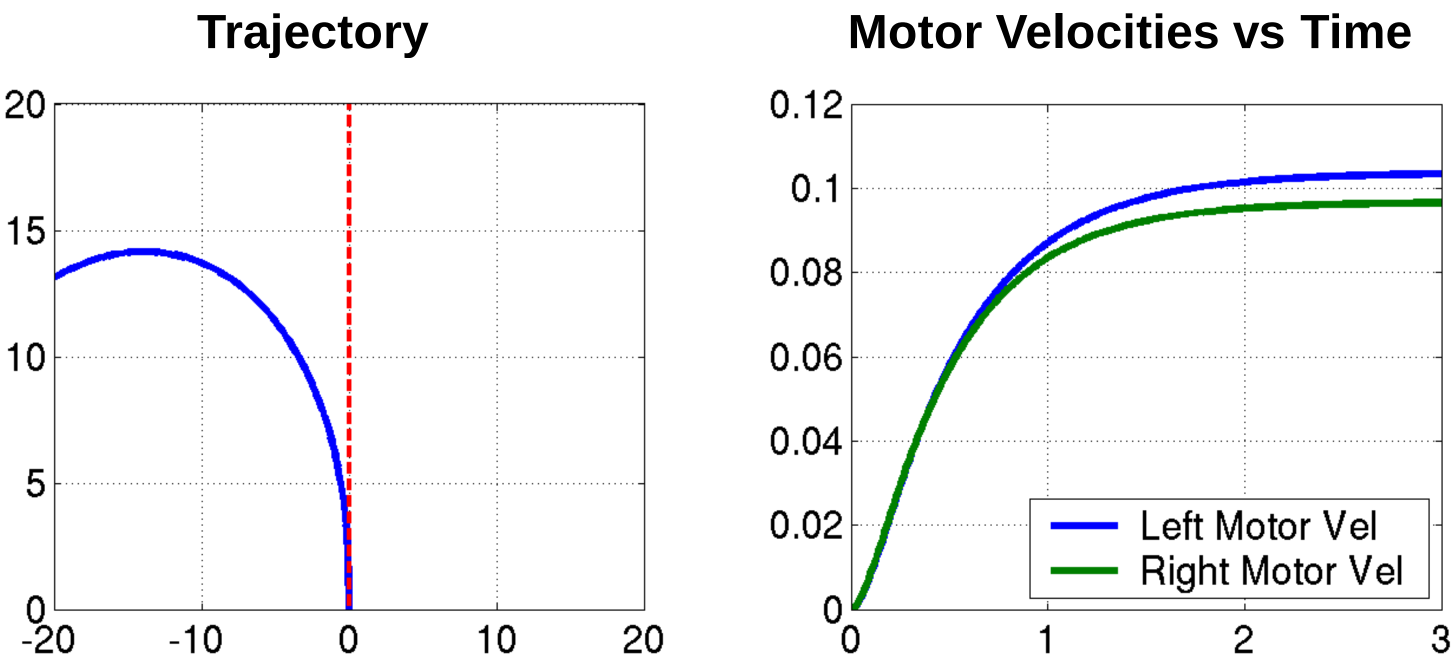


- Use the simplest controller which achieves the desired result
- Tuning PID constants is very tricky, especially for integral constants
- Consult the literature for more controller tips and techniques

Keep in a Straight Line

SECTION 2

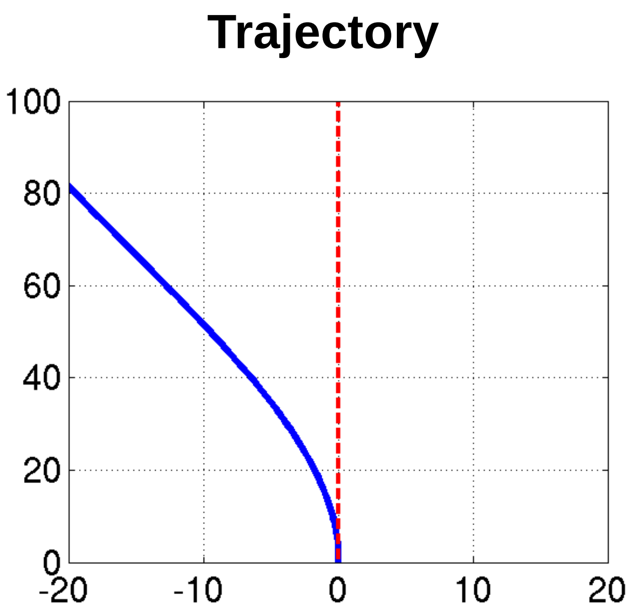
Problem: How do we make our robots go in a nice straight line?

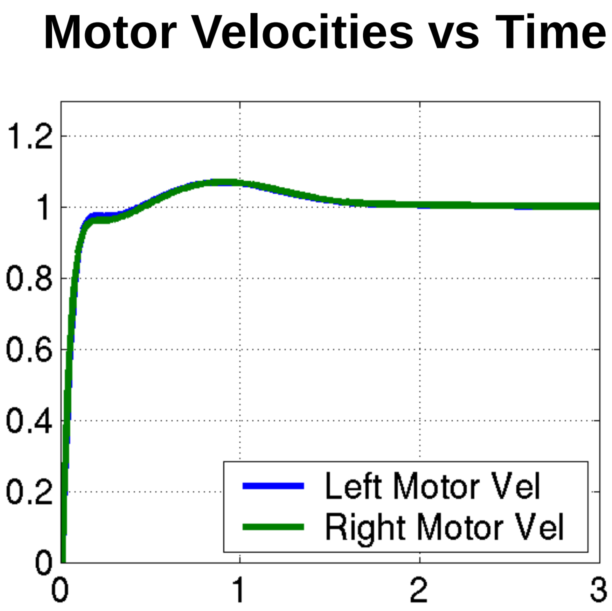


Model differential drive with slight motor mismatch

- With an open loop controller, setting motors to same velocity results in a less than straight trajectory

Problem: How do we make our robots go in a nice straight line?

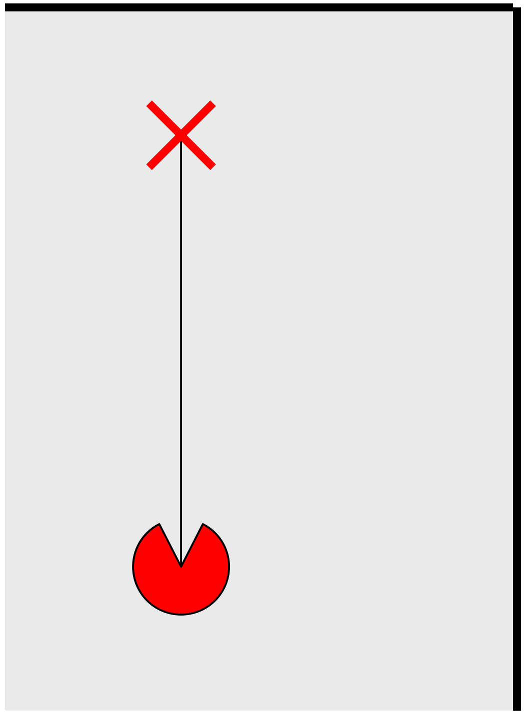




- With an independent PID controller for each motor, setting motors to same velocity results in a straight trajectory but not necessarily **straight ahead**!

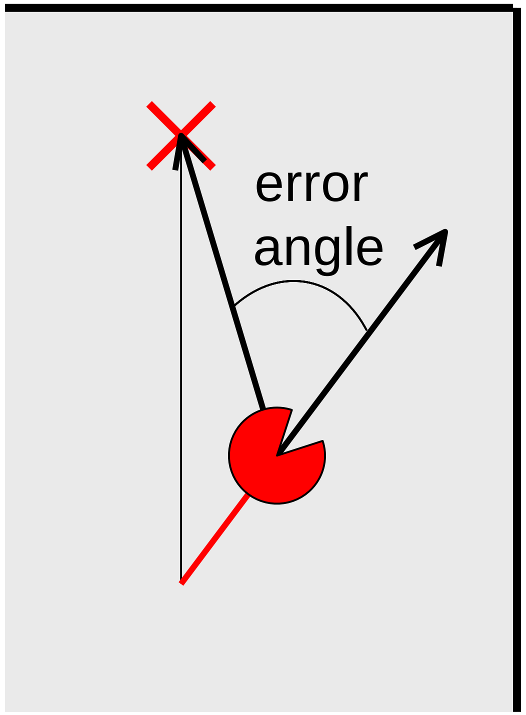
Problem: How do we make our robots go in a nice straight line?

- Need to couple drive motors
  - Use low-level PID controllers to set motor velocity and a high-level PID controller to couple the motors
  - Use one high-level PID controller which uses odometry or even image processing to estimate error



Problem: How do we make our robots go in a nice straight line?

- Need to couple drive motors
  - Use low-level PID controllers to set motor velocity and a high-level PID controller to couple the motors
  - Use one high-level PID controller which uses odometry or even image processing to estimate error



Summary

SECTION 3

Summary

- Integrating **feedback** into your control system "closes the loop" and is essential for creating robust robots
- Simple **finite state machines** make a solid starting point for your Python control systems
- Spend time this weekend **designing behaviors** and deciding how you will **integrate** these behaviors to create your control system