Controls and Signals

Maslab IAP 2009

Ellen Yi Chen

Agenda

- What do we mean by controls?
- Simple PID Controller
- Robot Drive Controller
- Examples
- Extensions

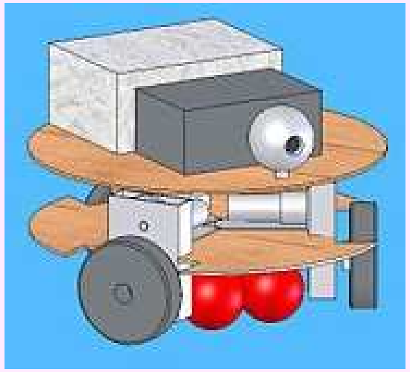
What are Controls?

- “High” Level Control Paradigms
  - Model/Plan/Act
  - Emergent
  - FSM (Finite State Machine)
- “Low” Level Control Loops
  - Motor Velocity
  - Robot Angular Position
  - Etc…

Why can’t we just tell the robot to go at 0.2m/s in a straight line?

What are Controls?

Sensors Information



Actuator Command

Sensors are far from perfect

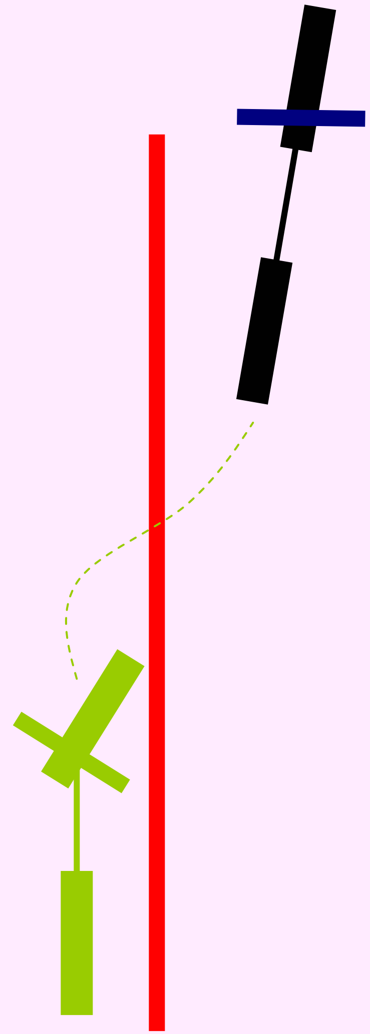
- Camera white balance
- Encoder quantization error
- Ultrasound reflections
- Infrared sensors noisy
- Etc…

Actuators are far from perfect

- Motor velocity changes with time/terrain/torque
- Wheels/gears slip
- Servos get stuck
- Etc…

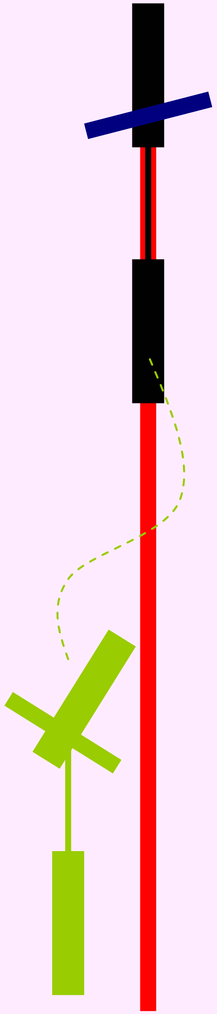
Example: Bike in straight line

- Steer the bike in a straight line blindfolded
- Open loop → no sensor feedback
- What if you hit a rock?
- What if the handle bars aren't perpendicular to the wheels?



Example: Bike in a straight line

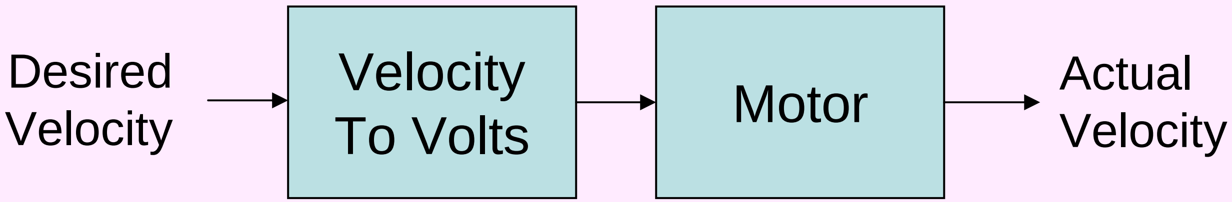
- **If you can see the pavement → Closed Loop Approach**
- **Control based on error: PID**
- **Proportional** : Change handle angle proportional to the current error
- **Derivative** : Large handle corrections when error is changing slowly, and small handle corrections when error is changing quickly
- **Integral** : Handle corrections based on the cumulative error



Problem: Set Motor Velocity

Open Loop Controller

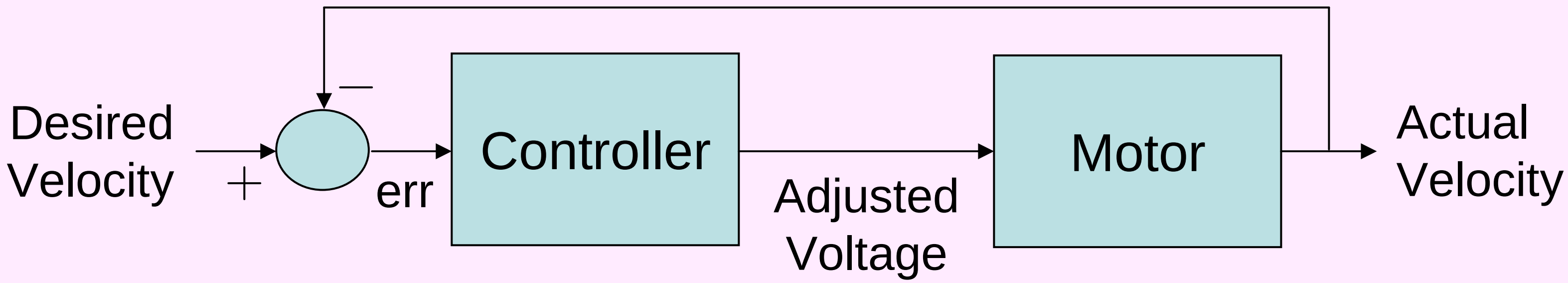
- Use trial and error to create relationship between velocity and voltage
- Problems
  - Supply voltage change
  - Bumps in carpet
  - Motor Transients



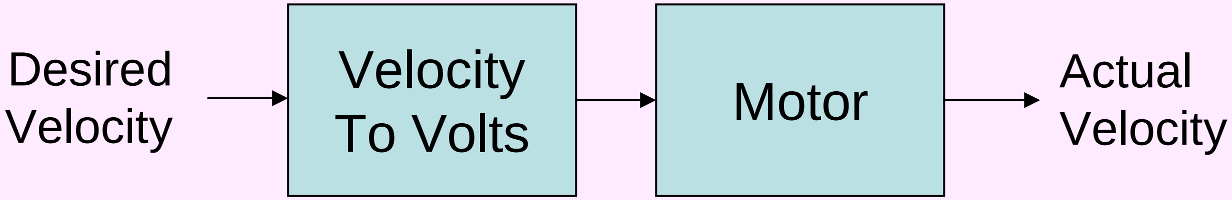
Problem: Set Motor Velocity

Closed Loop Controller

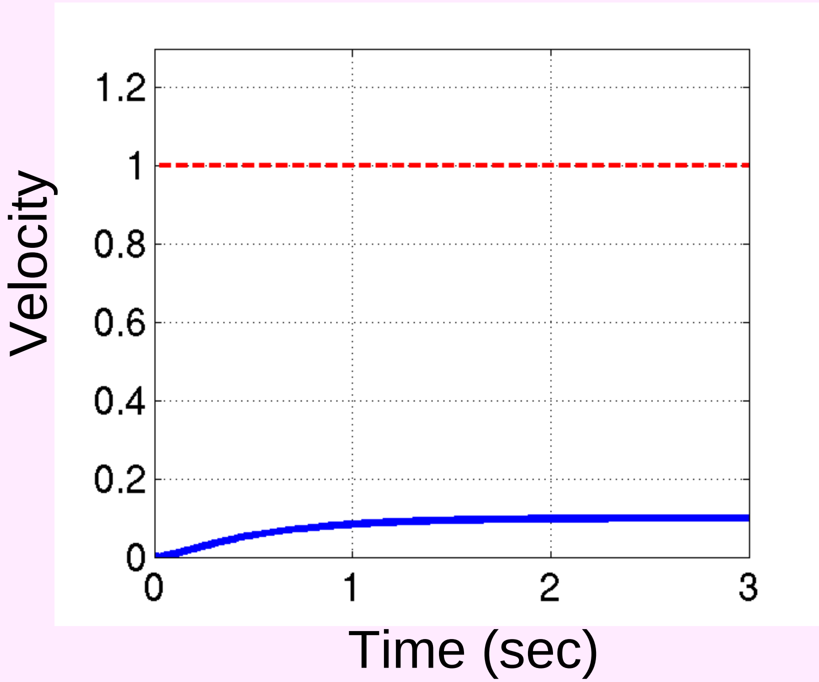
- Feedback is used so that the actual velocity equals the desired velocity
- Can use an optical encoder to measure actual velocity



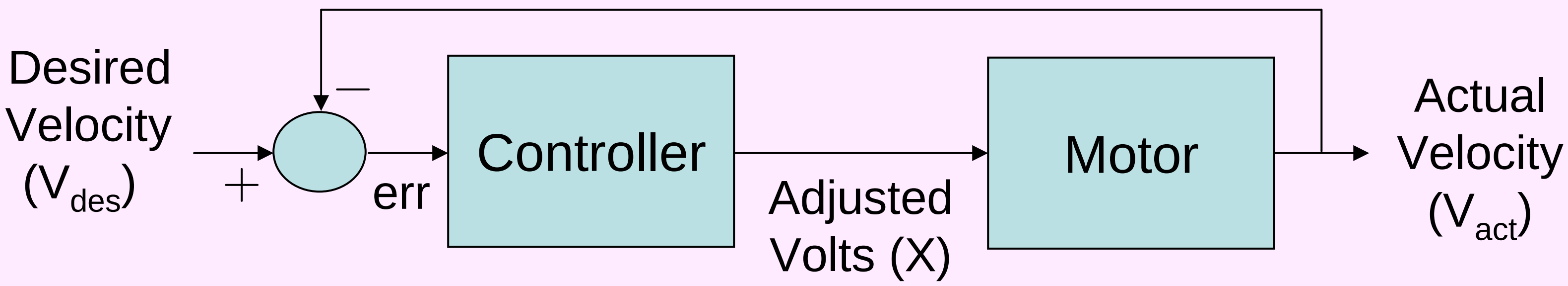
Step response with no controller



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

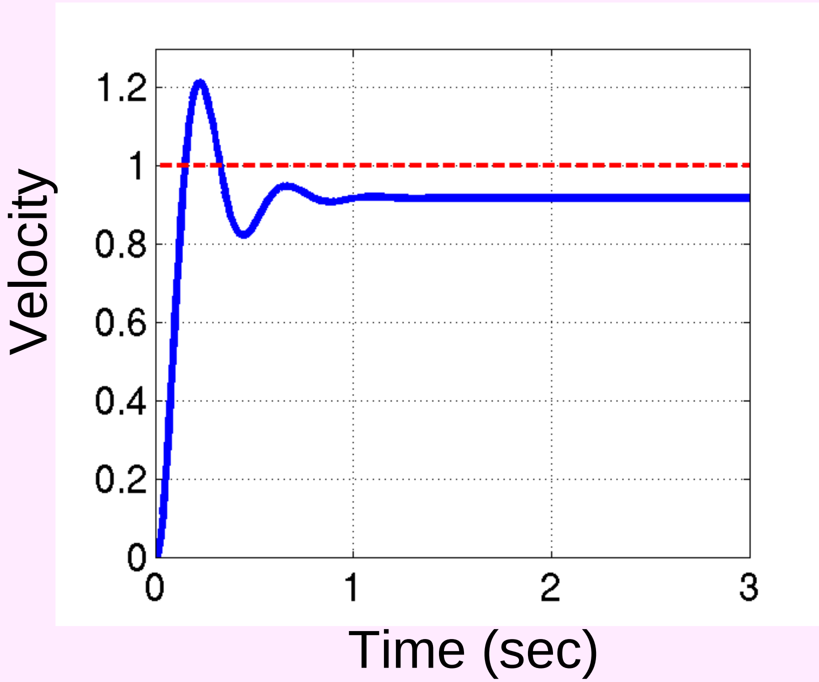


Step response with proportional controller

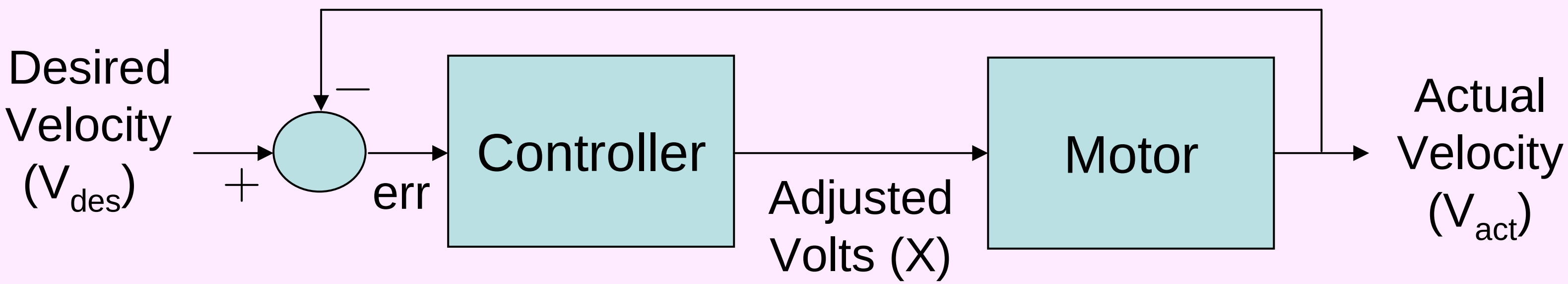


$$X = V_{des} + K_P \cdot (V_{des} - V_{act})$$

- Big error big = big adj
- Faster rise time
- Overshoot
- Stead-state offset (there is still an error but it is not changing!)

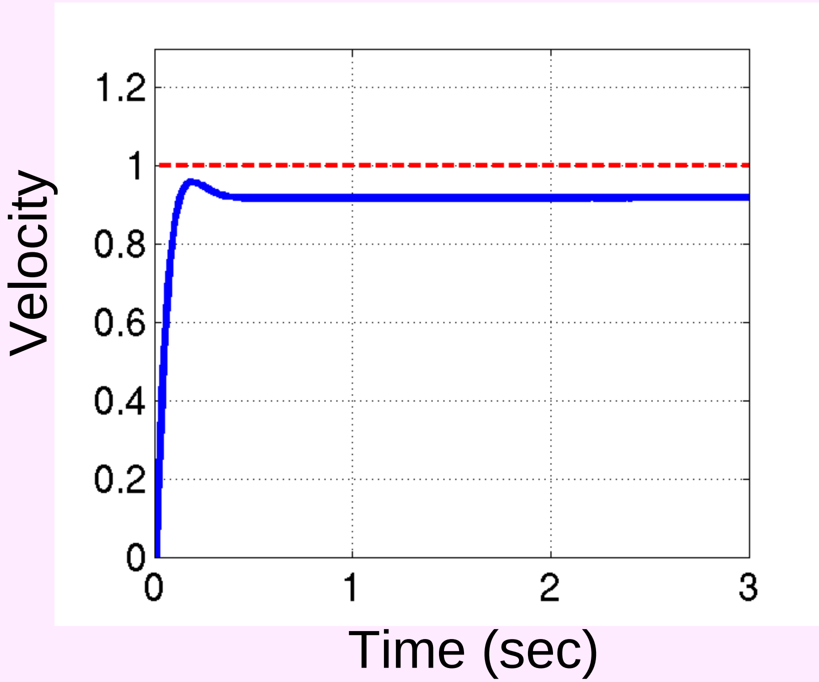


Step response with PD controller

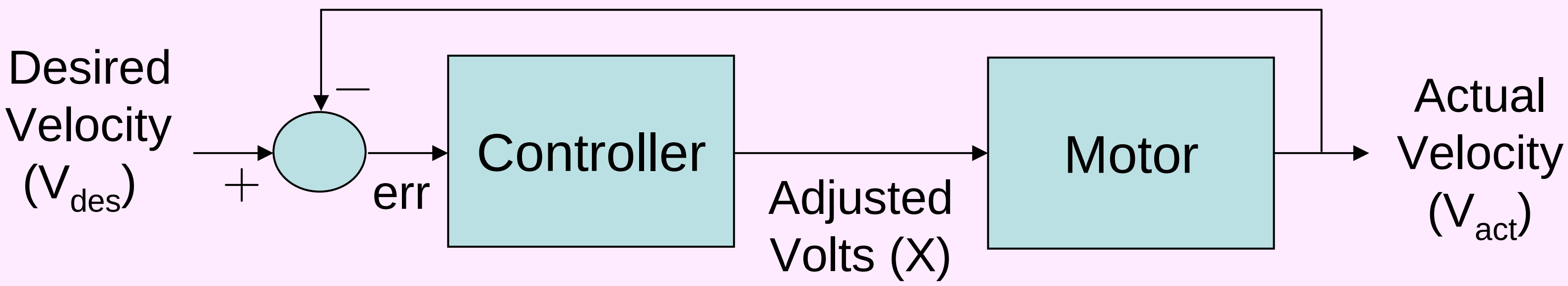


$$X = V_{des} + K_P e(t) - K_D \frac{de(t)}{dt}$$

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

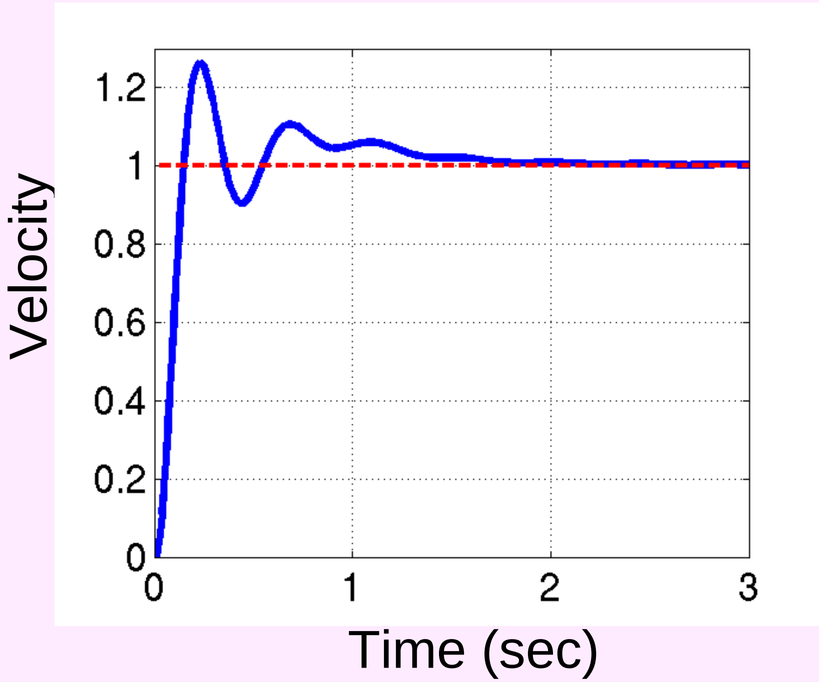


Step response with PI controller

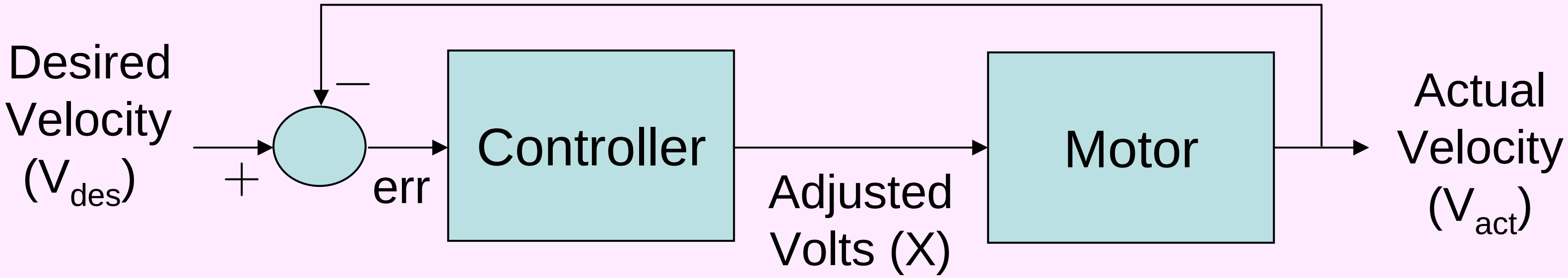


$$X = V_{des} + K_P e(t) - K_I \int e(t)\ dt$$

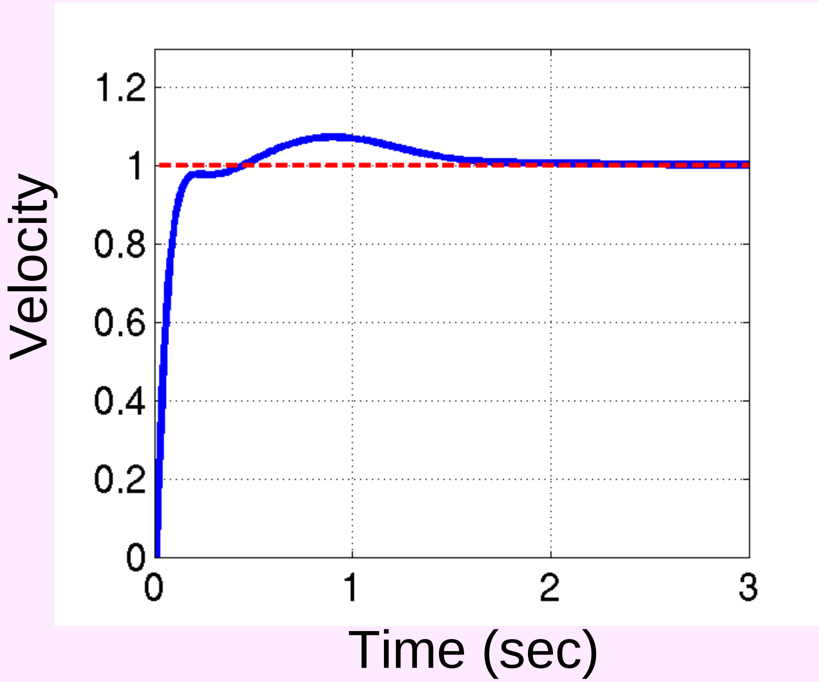
- Integral term eliminates accumulated error
- Increases overshoot



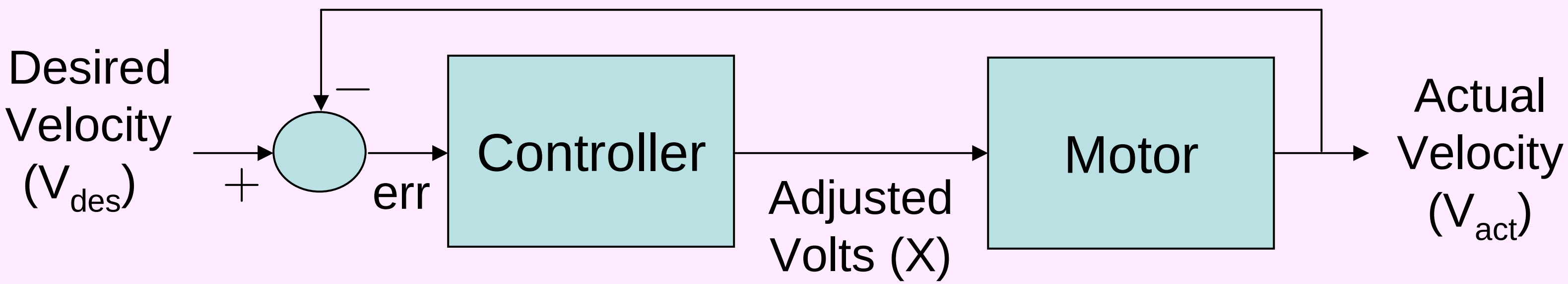
Step response with PID controller



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



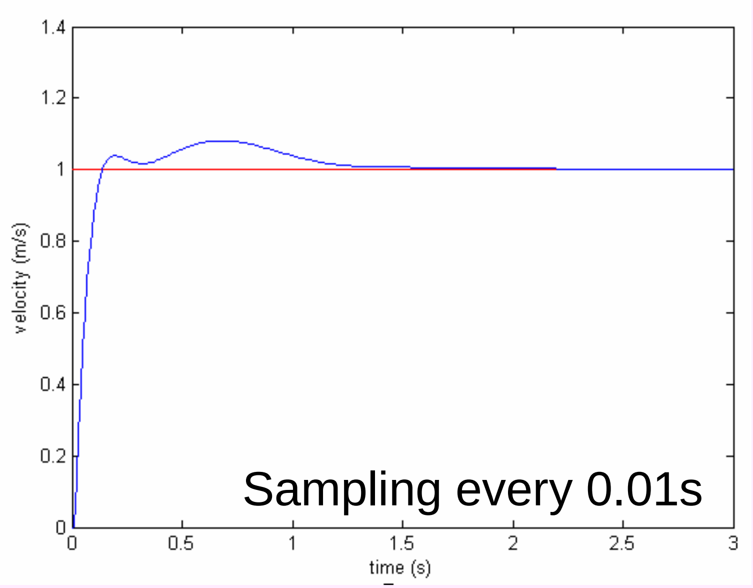
Choosing and tuning a controller

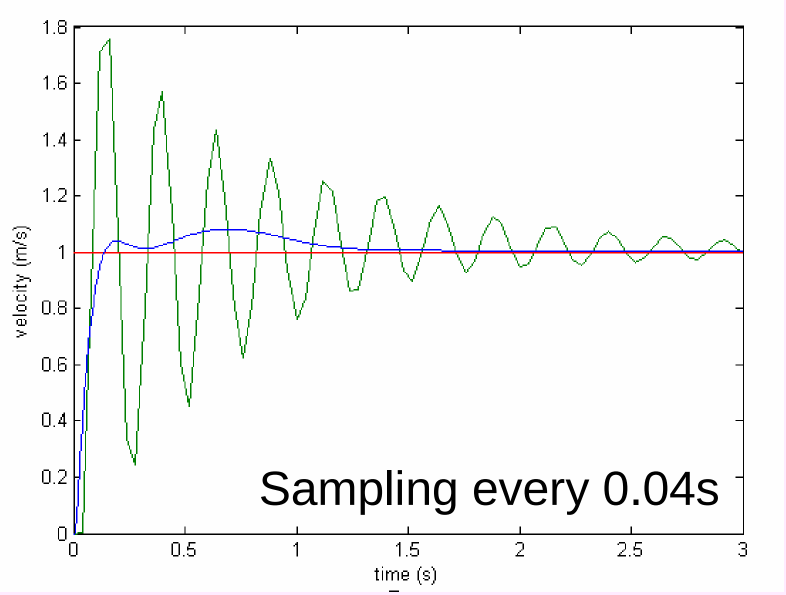


	Rise Time	Overshoot	SS Error
Proportional	Decrease	Increase	Decrease
Integral	Decrease	Increase	Eliminate
Derivative	~	Decrease	~

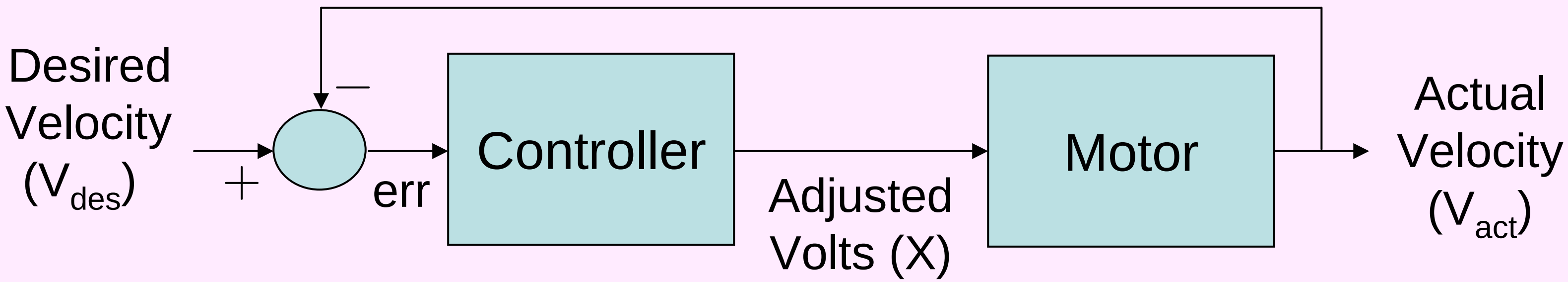
Effect of sampling time

- When you learn PID in 2.004, you learn it in continuous models
- For the discrete world, sampling time is another variable!
- Say you tune your PID and you sample every 0.01 seconds
- Then you write more code, add more threads
- At the end, you sample every 0.04 seconds. This affects your system and you may have to retune your PID!



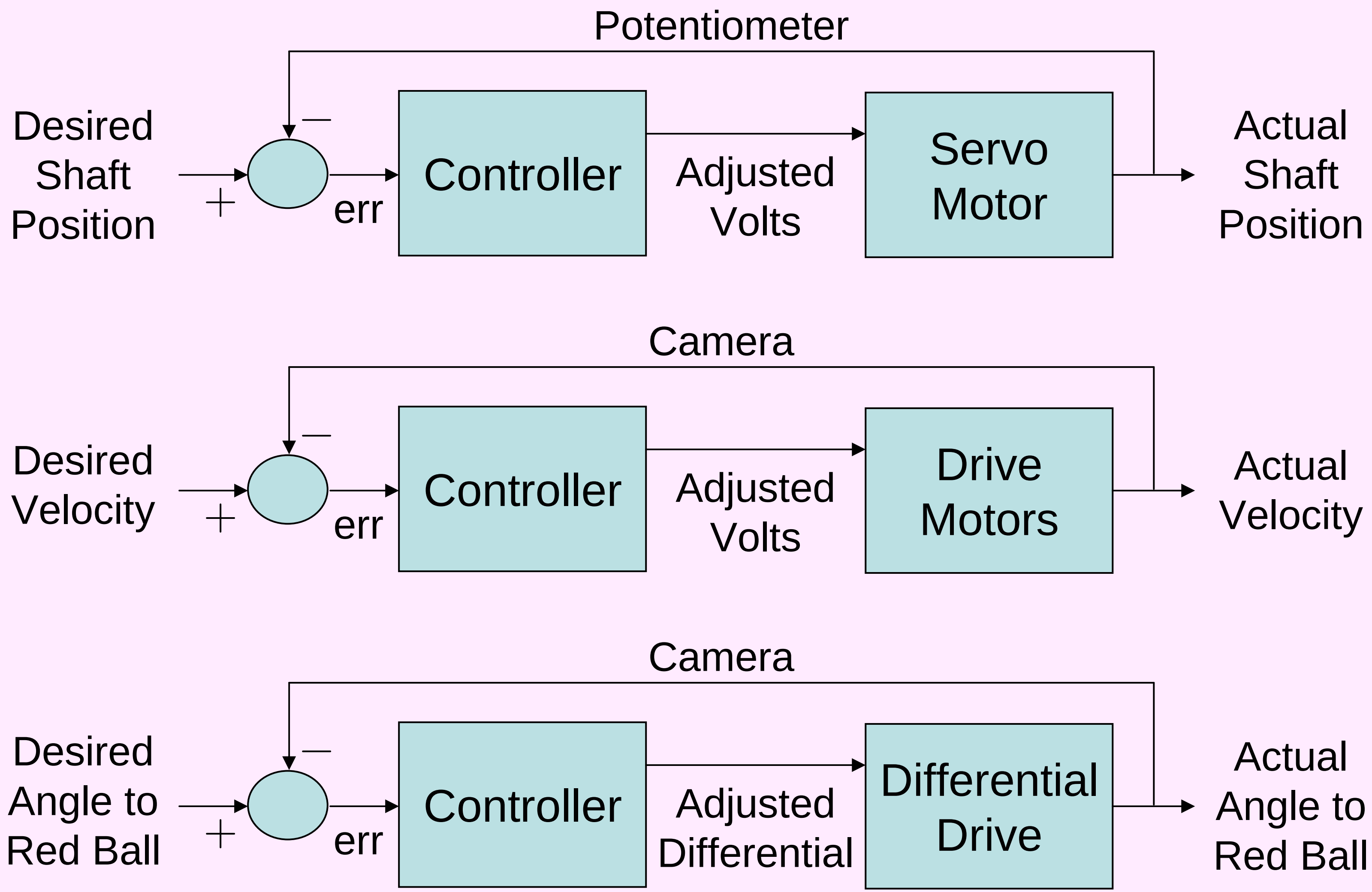


Choosing and tuning a controller



- 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

Other Control Loop Uses

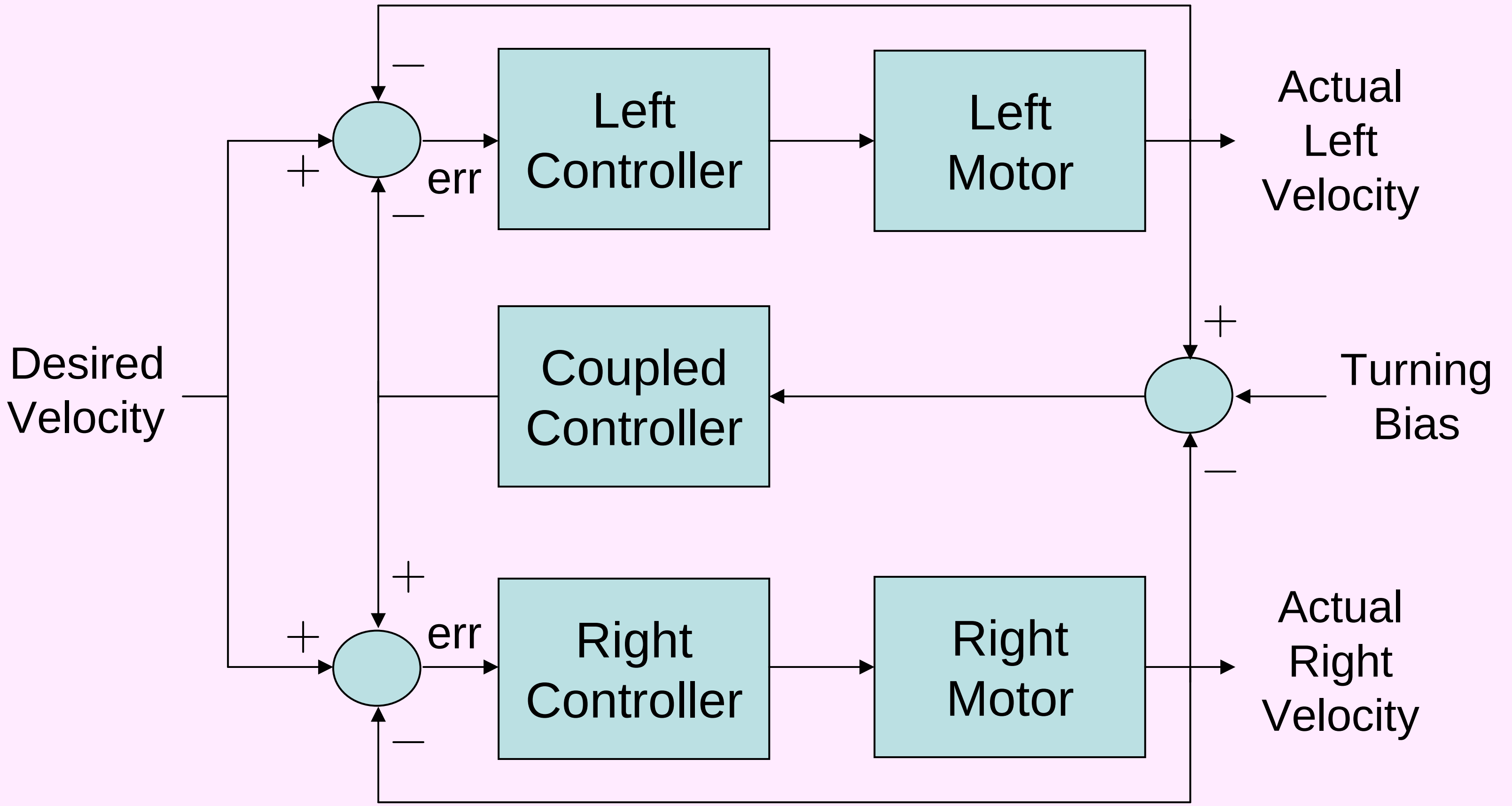


Matlab Examples

- motorContructor → Create a basic motor structure
- motorSetVoltage → Set the motor voltage
- motorStepResponse → Find unit step response for a motor
- motorPID → Find unit step response for a motor with PID
- robotPID → differential drive robot with two independent PID loops
- plotRobotTrajectory → plot the trajectory of robotPID

*Thanks to Christopher Batten for the code

We can synchronize the motors with a third PID controller



Inspired from “Mobile Robots”, Jones, Flynn, and Seiger, 1999

We can synchronize the motors with a third PID controller

What should the coupled controller use as its error input?

Velocity Differential

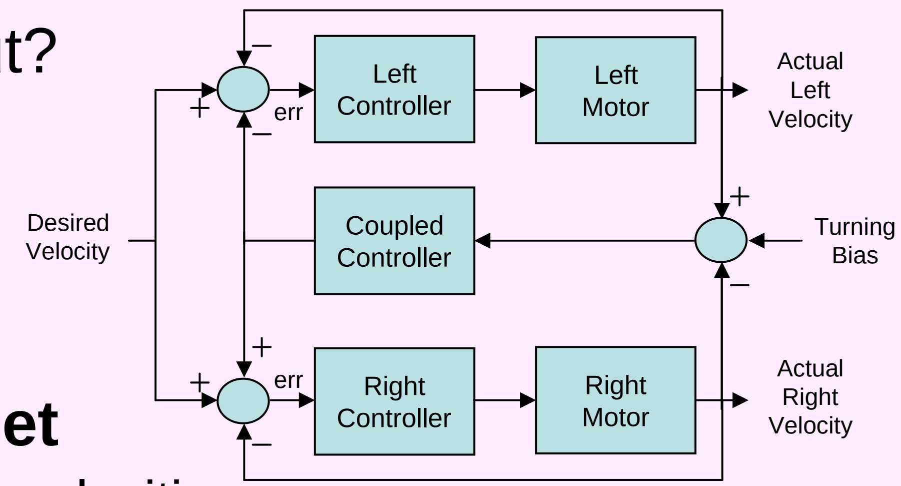
- Will simply help the robot go straight but not necessarily straight ahead

Cumulative Centerline Offset

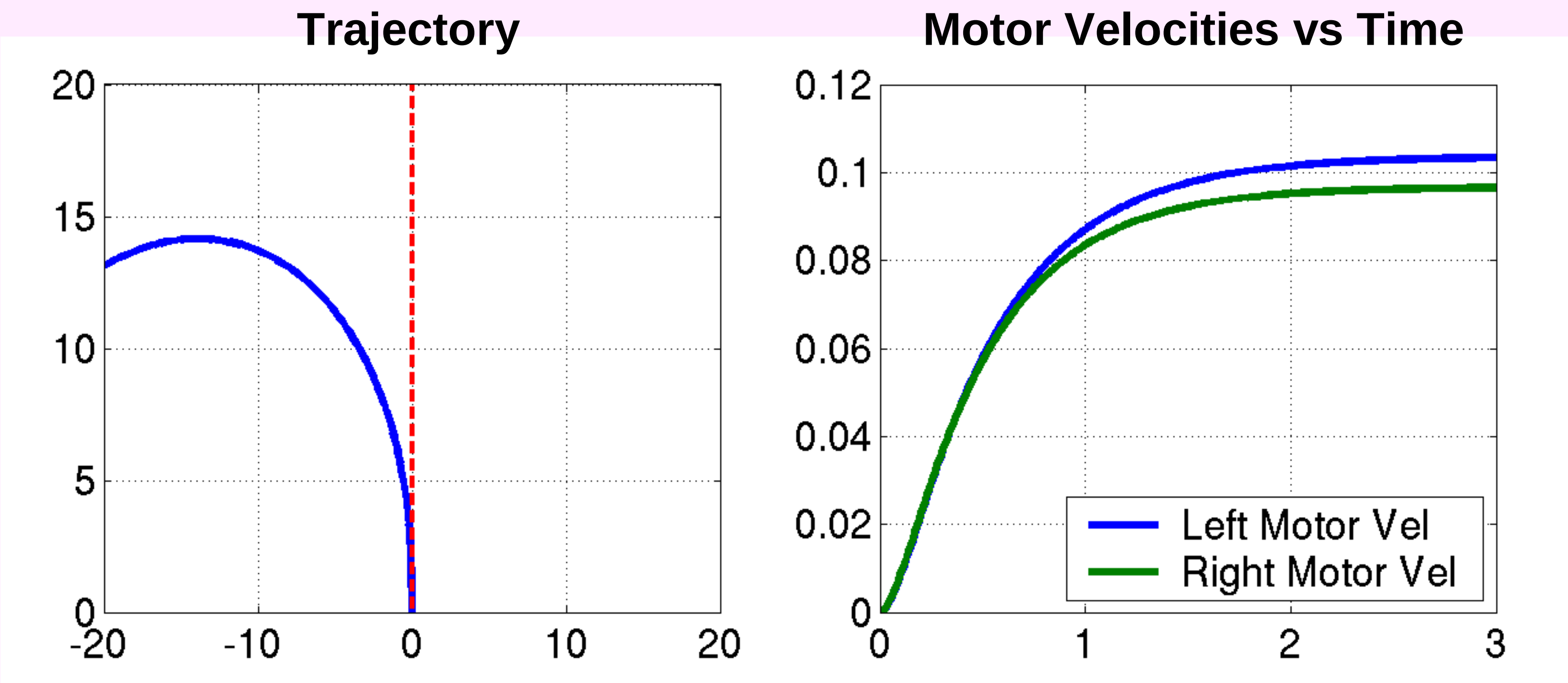
- Calculate by integrating motor velocities and assuming differential steering model for the robot
- Will help the robot go straight ahead

Alternatives:

- Gyro
- Camera



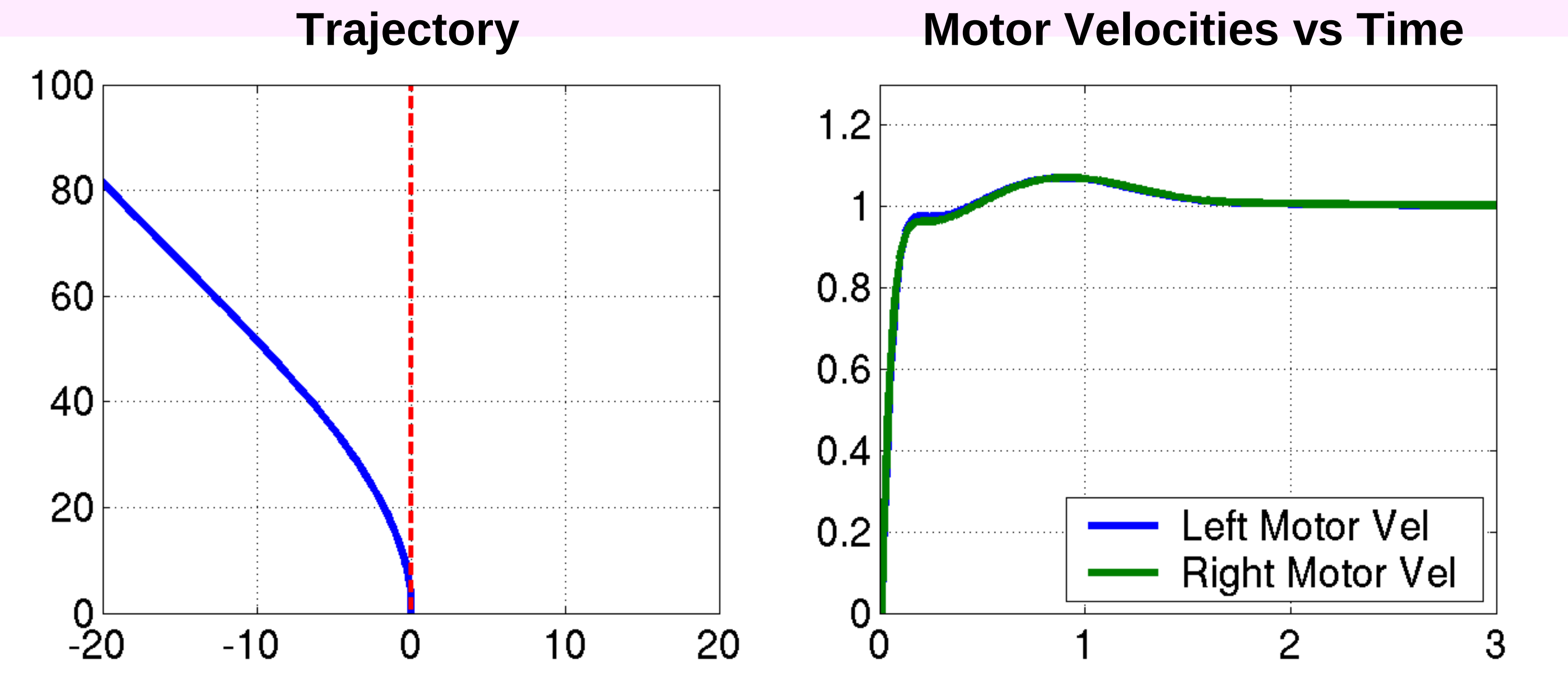
Robot driving in a straight line



Model differential drive with slight motor mismatch

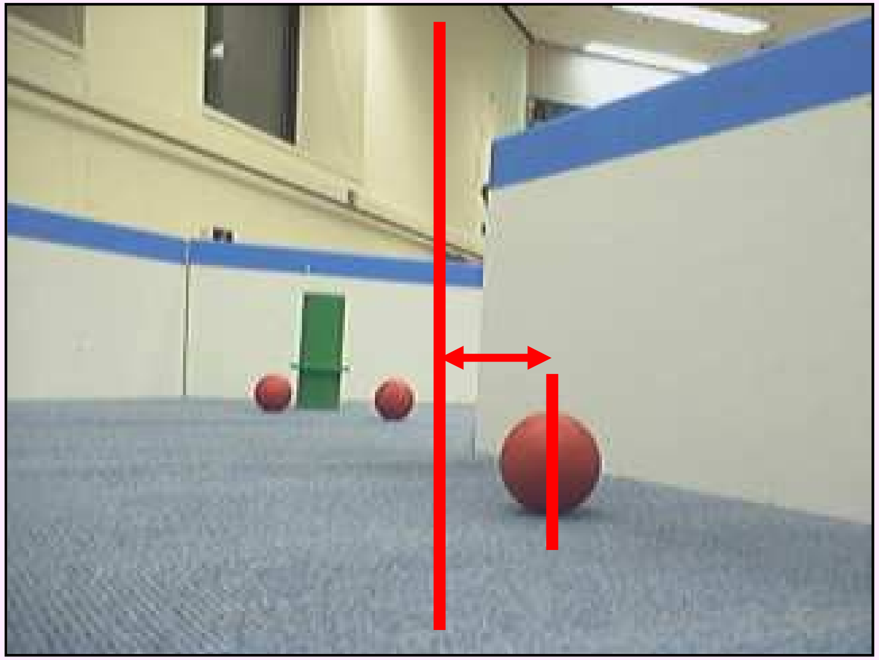
With an open loop controller, setting motors to same velocity results in a less than straight trajectory

Robot driving in a 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!

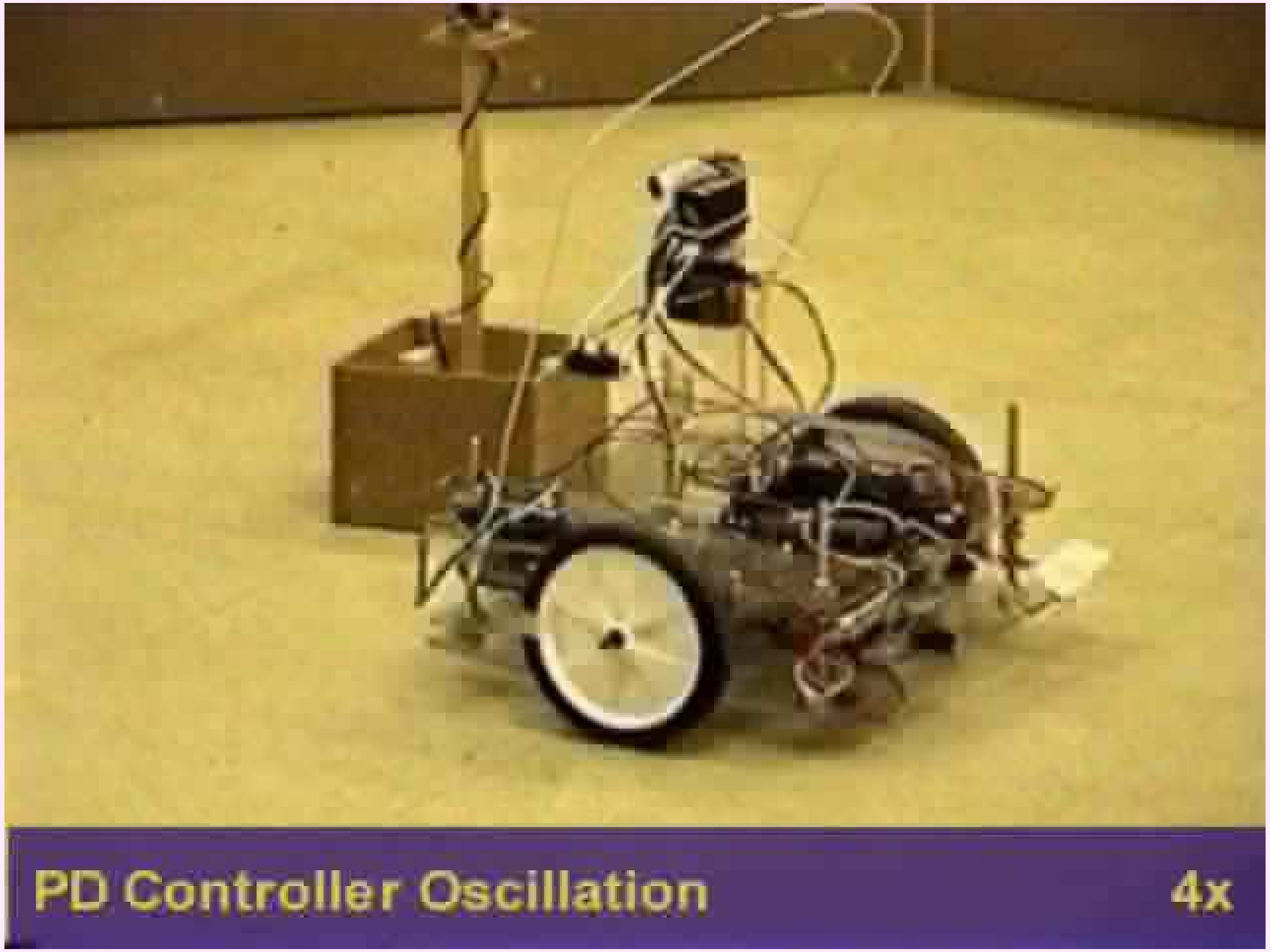
Alternatives: Gyro or Camera



– Track how far ball center is from center of image
– Use analytical model of projection to determine an orientation error
– Push error through PID controller

What if we just used a simple proportional controller? Could lead to steady-state error if motors are not perfectly matched!

Example Videos



Java Examples

Wall Following without PID

http://web.mit.edu/6.186/2008/lectures/pid/wallfollow/index.html

Wall Following with PID

http://web.mit.edu/6.186/2008/lectures/pid/wallfollowpid/index.html

Driving Straight without PID

http://web.mit.edu/6.186/2008/lectures/pid/towardball/index.html

Driving Straight with PID

http://web.mit.edu/6.186/2008/lectures/pid/towardballpid/index.html

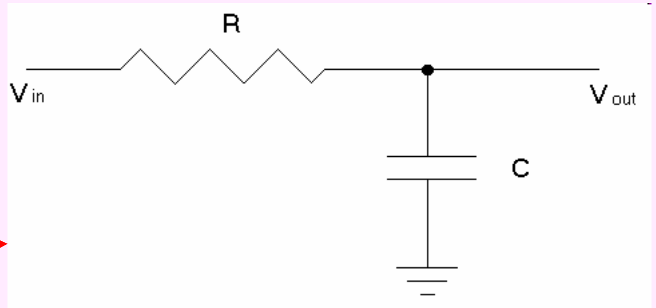
*Thanks to Dany Qumsiyeh for the code

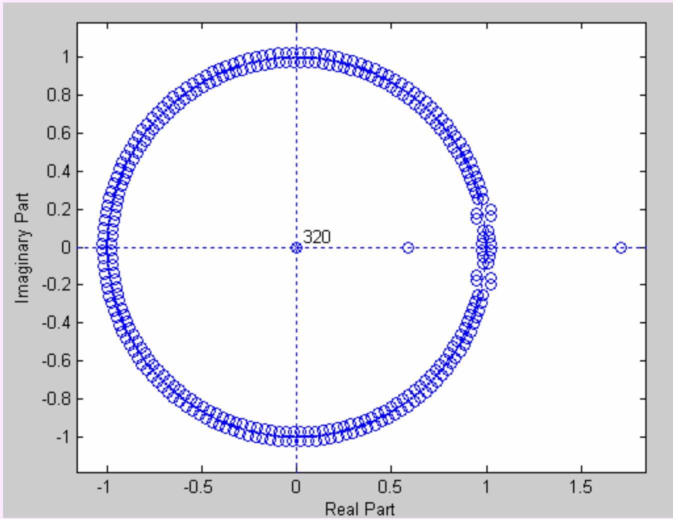
Extensions

- Controls and signal processing are powerful tools (6.003, 2.004, etc…)
  - Modeling of physical systems
    - Given parameters of a system, how do we determine how it will act to a given input
    - Etc…
  - Control schemes
    - Deterministic control schemes
    - PID controllers
    - Fuzzy logic controllers
    - Etc…
  - Signal processing
    - Discrete and continuous methods
    - Filters: Low-pass, high-pass, band-pass, notch
    - Frequency domain techniques
    - Echo removal
    - Autocorrelation techniques
    - Etc…
  - System identification
    - For an unknown black box system, how do we find the transfer function?
    - Impulse invariant, swept sine, stochastic methods
    - Parametric techniques, nonparametric techniques
    - Etc…

Filter Design

- Continuous Filters
  - In the real world, time is continuous.
  - We are constantly getting inputs and giving outputs
  - Analog circuits
- Discrete Filters
  - When using computers, we get discrete samples at a given sampling rate
  - FIR Filters (Finite Impulse Response)
  - IIR Filters (Infinite Impulse Response)
- Filter Types
  - Low Pass –allows low frequencies to pass through
  - High Pass – allows high frequencies
  - Band Pass- allows a bands of frequencies to pass

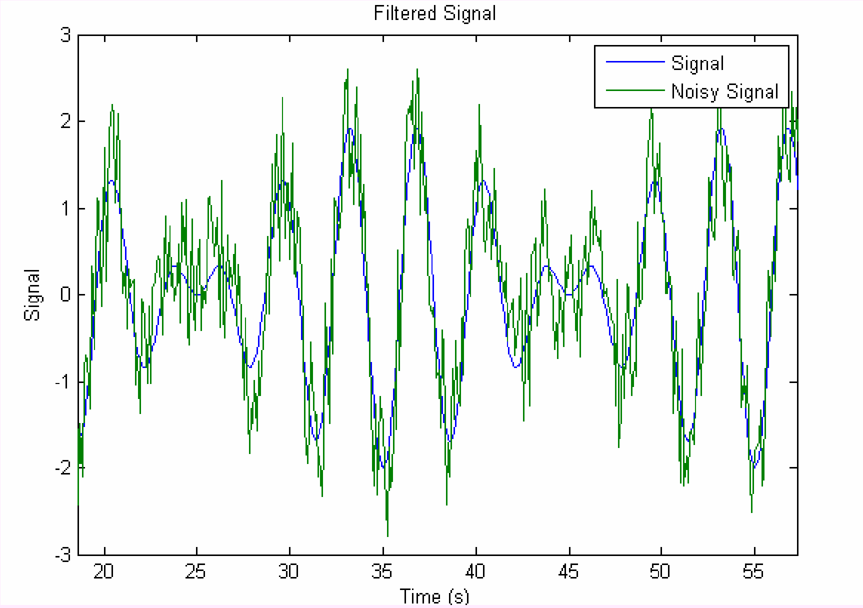




Pole/Zero plot for FIR filter

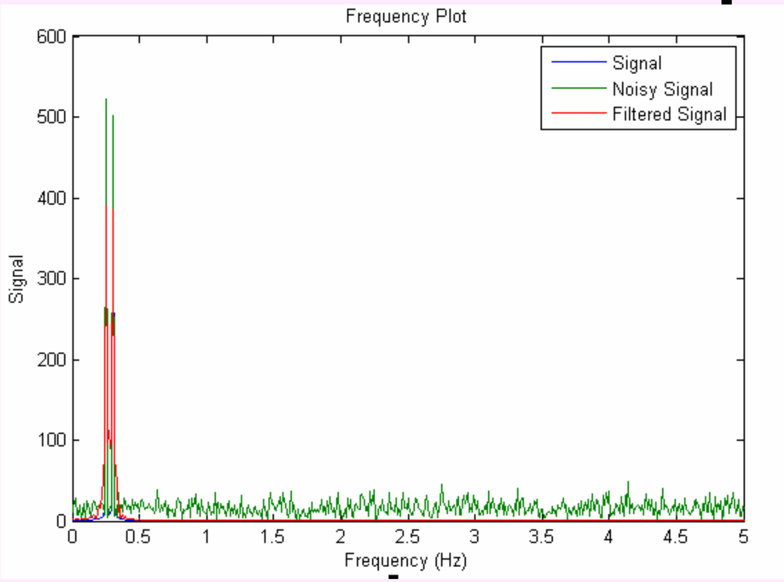
Example: FIR Filter

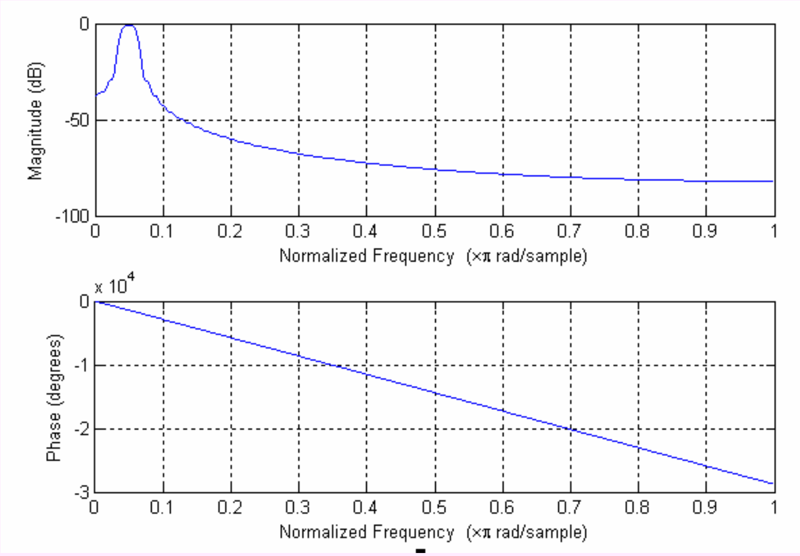
- Lets say you have a signal and your sensor is very noisy
- Could be IR sensor, ultrasound, or even an image
- How do you separate actual signal from the noise?
- Use an FIR digital filter (in your code)
- y(n) → filter output at time n
- x(n-k) → sensor input at time n-k
- b → weighting constants given by Matlab
- N → filter order given by Matlab

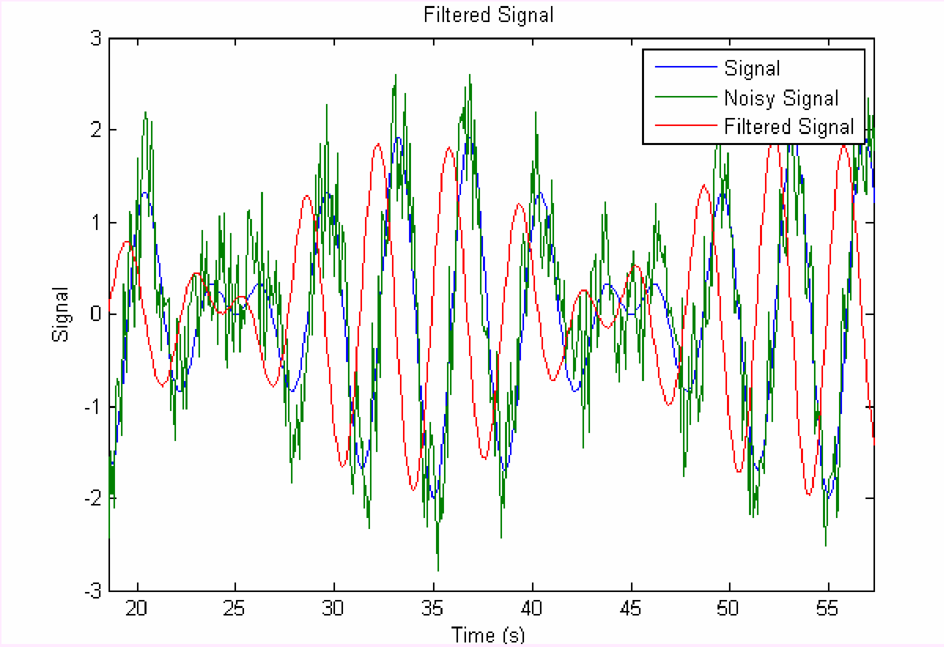


$$y[n] = b_0 x[n] + b_1 x[n-1] + \cdots + b_N x[n-N]$$

Example: FIR filter







- Create band pass filter
- Recover the band of frequencies where the actual signal is
- Special Notes
  - The better the filter, the higher the order (N)
  - The lag in the filter is approximately N/2 samples

Take Aways

- Why do we need PID?
  - Motors are not matched
  - Your center of mass is not in the middle of your robot
  - Signals are noisy
- Use a PID Controller to simplify driving code
  - Motor Speed: Encoders
  - Robot angle: Gyro
  - Robot trajectory: Gyro and Camera

References

- **Christopher Batten, "Controls for Mobile Robot," 2007, http://maslab.mit.edu/2007/wiki/Control_lecture.**
- **Dany Qumsiyeh, Controls scripts 2008, http://maslab.mit.edu/2008/wiki/PID.**