PID CONTROL

EECS 467: Autonomous Robotics Laboratory

Today's Plan

- Simple controllers
  - Bang-bang
  - PID

- Pure Pursuit

Control

- Suppose we have a plan:
  - "Hey robot! Move north one meter, the east one meter, then north again for one meter."

- How do we execute this plan?
  - How do we go exactly one meter?
  - How do we go exactly north?

Open Loop (Feed forward)

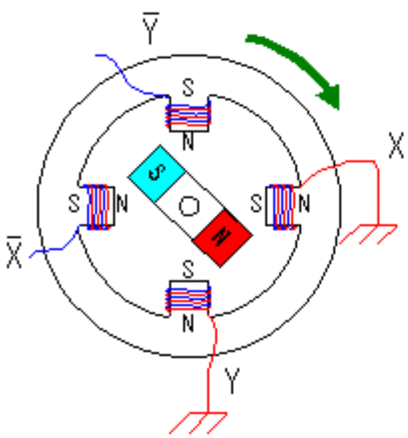
- Idea: Know your system.

  - If I command the motors to "full power" for three seconds, I'll go forward one meter.

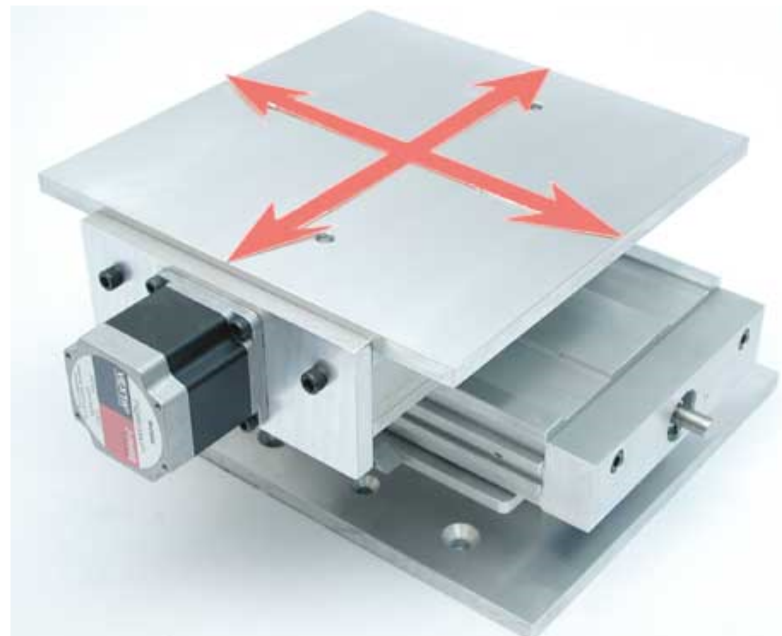
  - Is this a good idea?

Open Loop: XYZ Positioning table

- Physical construction of stepper motors allows precise open-loop positioning



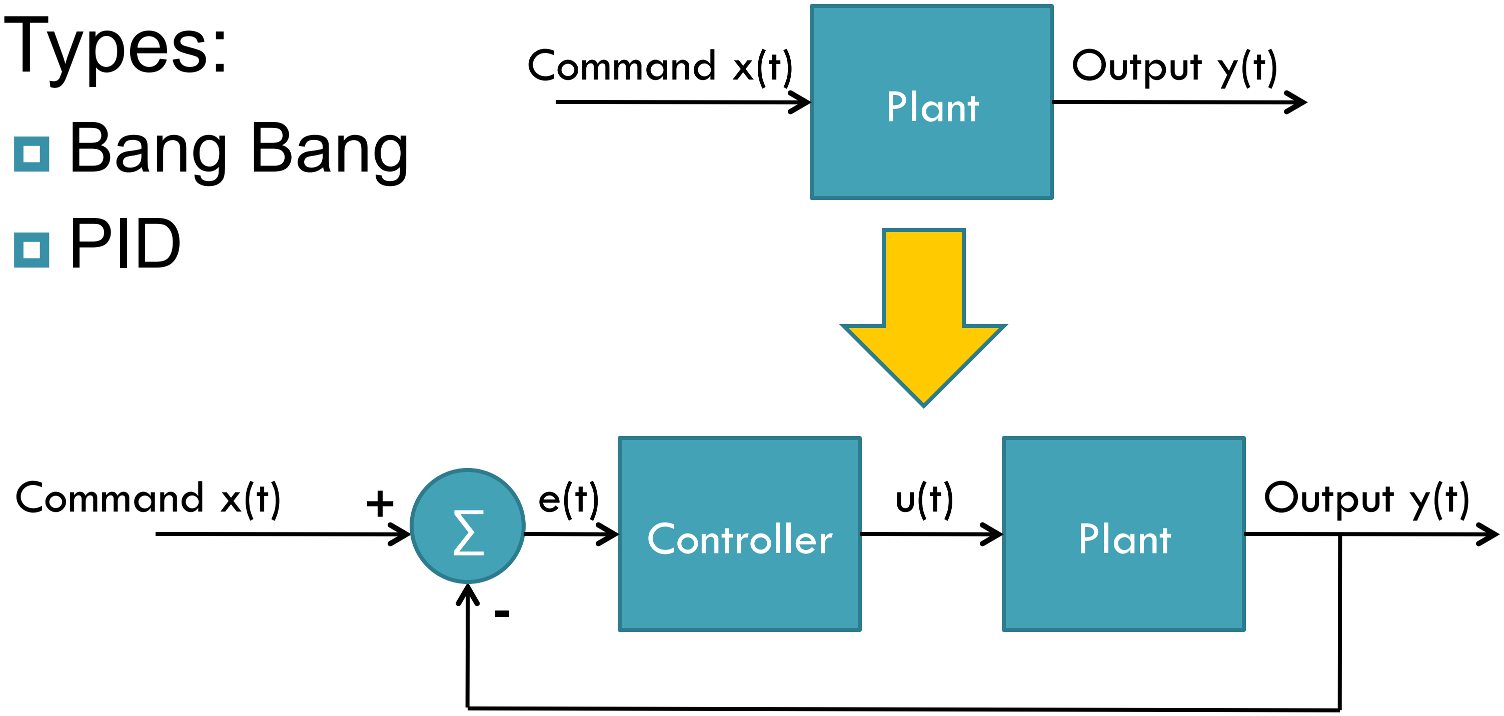
X	$\overline{X}$	Y	$\overline{Y}$
0	1	0	1
1	0	0	1
1	0	1	0
0	1	1	0



Credit: electricsteppermotors.com

Closed Loop

- Use real-time information about system performance to improve system performance.
- Types:
  - Bang Bang
  - PID



Bang Bang Control

- Actuator is always at one of its limits

Bang-Bang:

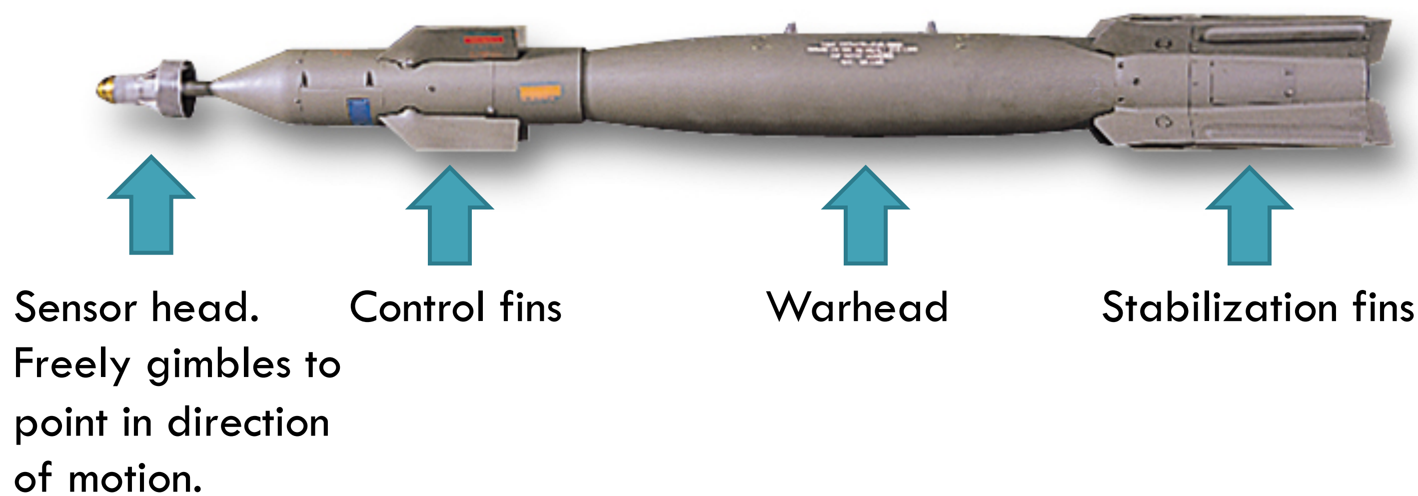
```
while (true)
    if (error < 0)
        Command(maximum value)
    else
        Command(minimum value)
end
```

This is stupid. No one would do this.

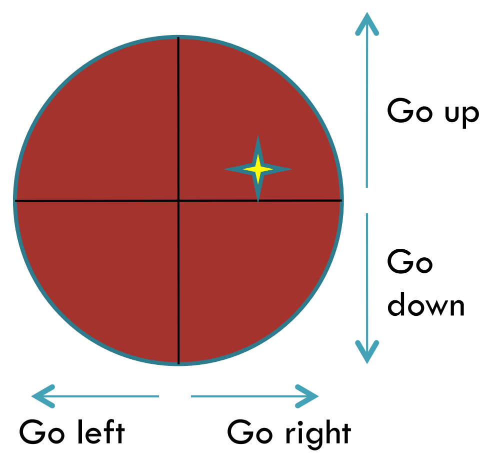
Especially for something important....

Bang Bang… Bang.

- GBU-12 Paveway II Laser Guided Bomb

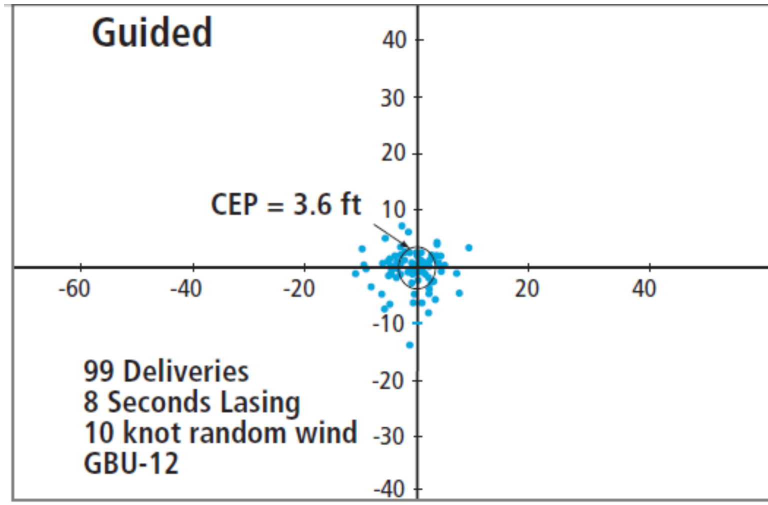


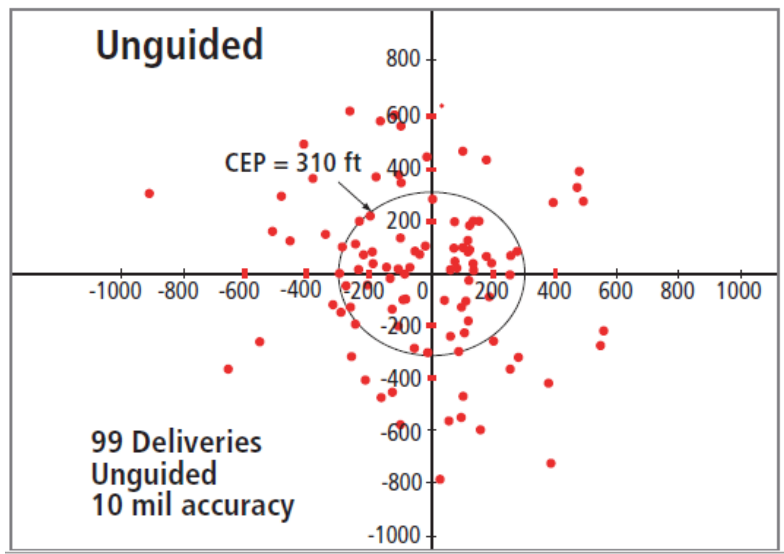
- Sensor head detects laser spot in one of four quadrants.



Bang Bang Control (Continued)

- Pros:
  - Simple/cheap to implement
  - Hugely better performance than open loop
  - Needs only primitive actuators
- Cons:
  - Performance (higher drag)





Proportional Control

- Obvious improvement to Bang-Bang control: allow intermediate control values
- $u(t) = K_p\ e(t)$
- Intuition: If $e(t) > 0$, goal position is larger than current position. So, command a larger position.

Proportional Control

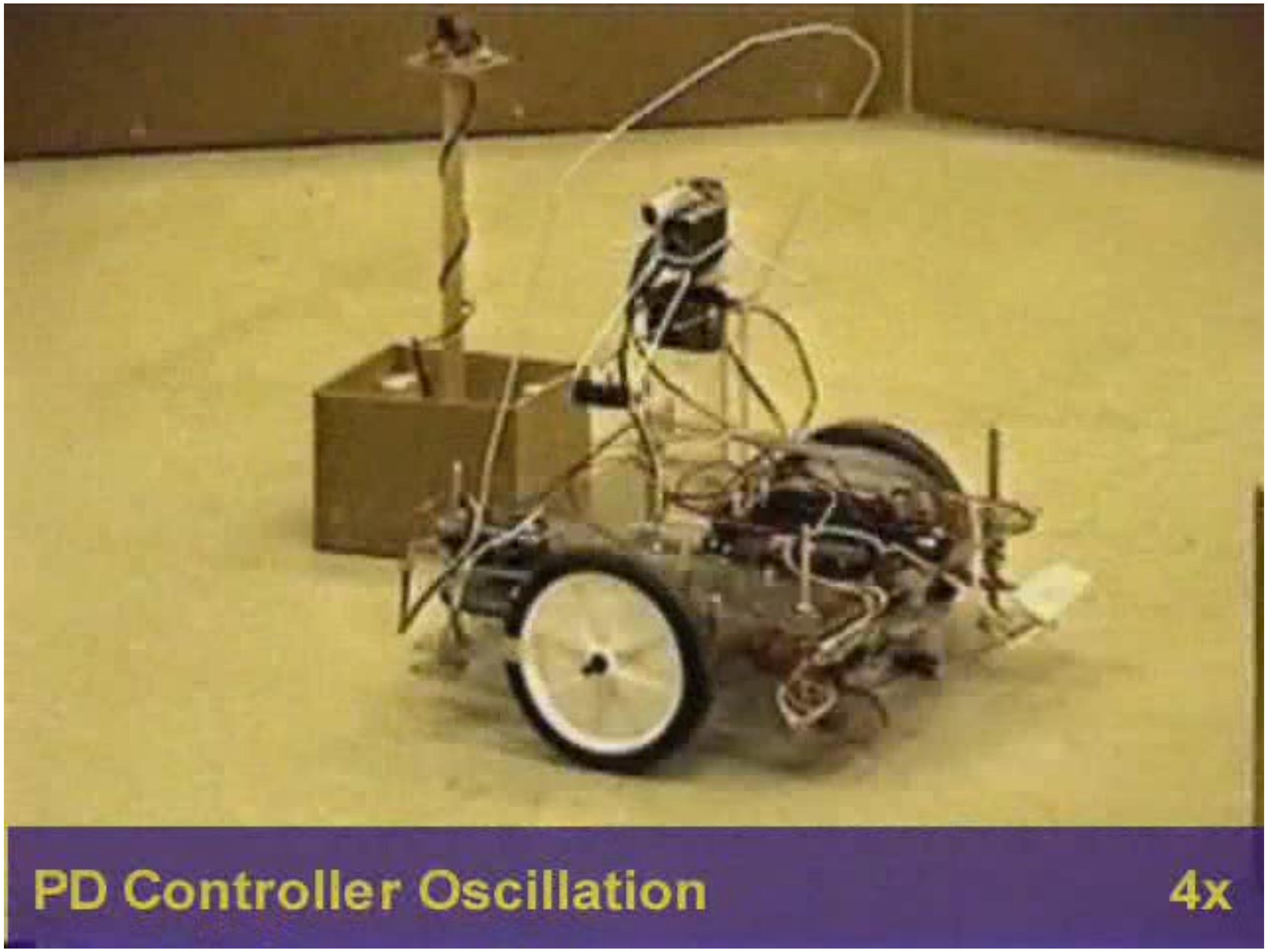
- We want to drive error to zero quickly
  - This implies large gains

- We want to get rid of steady-state error
  - If we're close to desired output, proportional output will be small. This makes it hard to drive steady-state error to zero.
  - This implies large gains.

- Really large gains?
  - Bang-bang control.

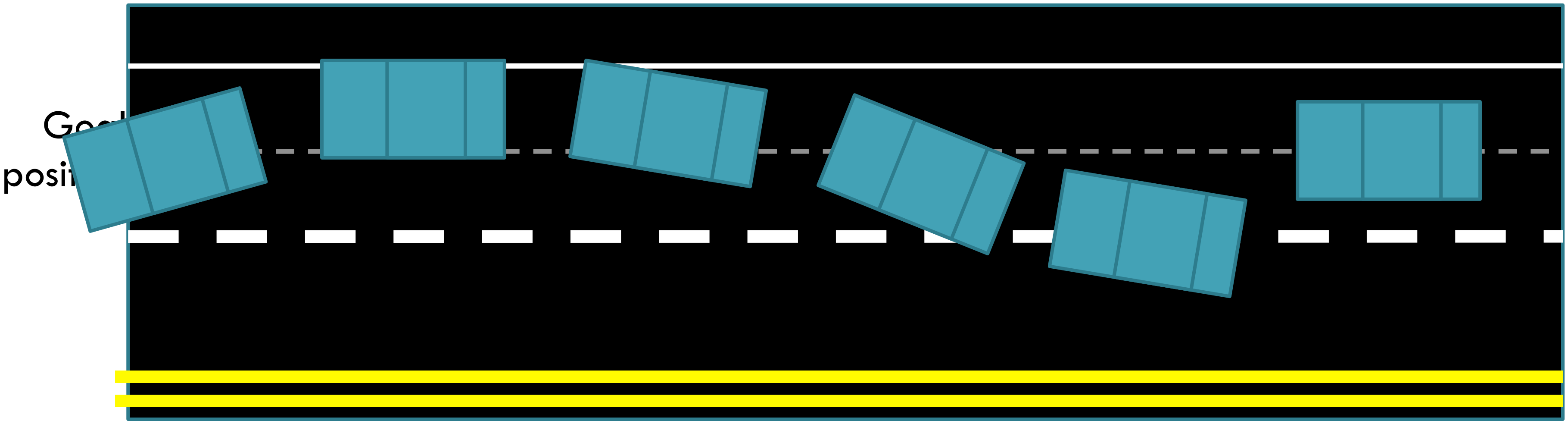
- What's wrong with really large gains?
  - Oscillations. (We'll come back to this)

Proportional Control: Oscillation



Intuition: P

- Suppose we observe lateral position of car driving down road



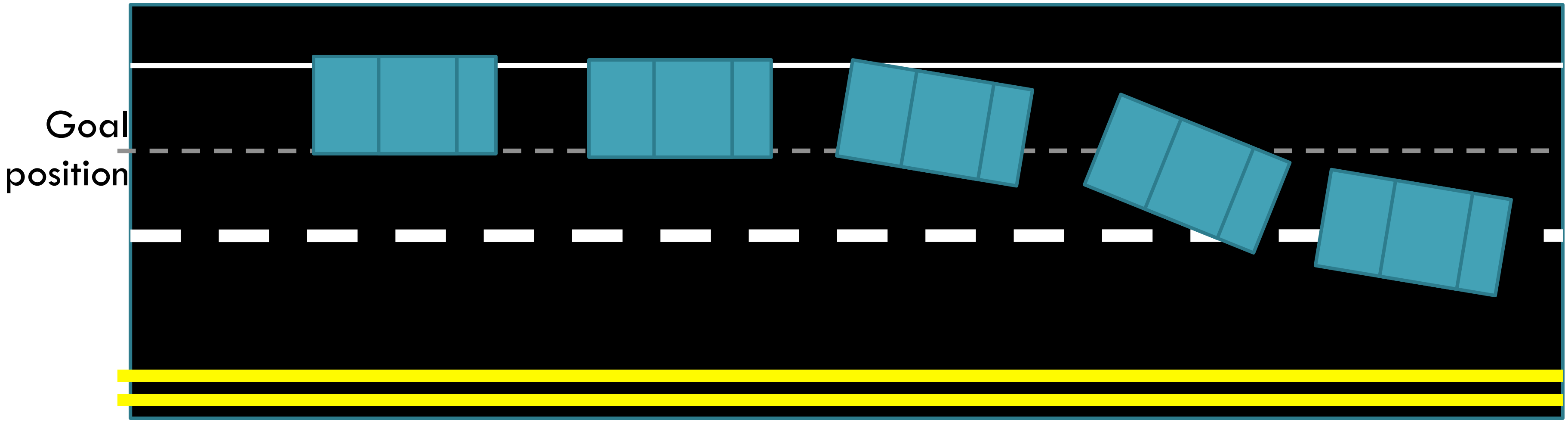
- P control is “happy” when car is centered in lane
  - Even if we’re pointed away from the center.

Derivative Control

- Our vehicle doesn't respond immediately to our control inputs.
  - From the controller's perspective, there's a delay.
- We need to "dampen" the behavior of the system.
  - When we're getting close to our desired value, slow down a bit!
- Problem: computing derivatives is very sensitive to noise!

Intuition: D

- Derivative control is "happy" when we're driving parallel to desired path.
  - Things not getting better, but not getting worse either.



PD Controller

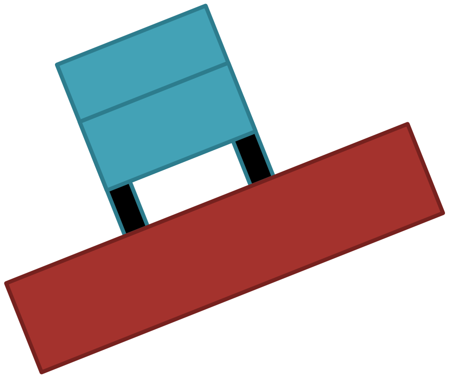
- Combine P and D terms
  - P seeks error = 0
  - D seeks d/dt error = 0

  - D term helps us avoid oscillation, allowing us to have bigger P terms
    - Faster response
    - Less oscillation

Integral Control

- Suppose we're in steady state, close to desired value.
  - D term is zero
  - P term is nearly zero

- P term may not be strong enough to force error to zero
  - Perhaps the car is on a hill
  - Perhaps the actuator is misaligned
    - We're not commanding what we think



Integral Control

- If we have error for a long period of time, it argues for additional correction.
- Integrate error over time, add to command signal.
- Force *average* error to zero (in steady state)

PID Control

- Combine all three types together, different gains for each type:

$$u(t) = K_p e(t) + K_d \frac{d}{dt} e(t) + K_i \int e(t)$$

- Note: we *often* won't use all three terms.
  - Each type of term has downsides
  - Use only the terms you need for good performance
    - Avoid nasty surprises

Computing Gains

- Where do PID gains come from?
  - Analysis
    - Carefully model system in terms of underlying physics and PID controller gains.
    - Compute values of PID controller so that system is 1) stable and 2) performs well
  - Empirical experimentation
    - Hard to make models accurate enough: many parameters
    - Often, easy to tune by hand.

PID Tuning

- Very simple PID tuning procedure:
  1. Increase P term until performance is adequate or oscillation begins
  2. Increase D term to dampen oscillation
  3. Go to 1 until no improvements possible.
  4. Increase I term to eliminate steady-state error.

- Better procedure
  - Ziegler-Nichols Tuning Method

Integrator Gotchas

- Integrator wind-up:
  - Suppose it takes a large command to eliminate steady state error. (I.e., the hill is VERY steep)
  - If desired command changes, it can take a long time to “drain” the integrator. ➔ bad system performance
- Solutions
  - Clamp integrator