Georgia Institute of Technology

Dr. Magnus Egerstedt
Professor
School of Electrical and
Computer Engineering

Control of Mobile Robots

Module 1
Introduction to Controls

How make mobile robots move in effective, safe, predictable, and collaborative ways using modern control theory?

Lecture 1.1 – Control of Mobile Robots

- Magnus Egerstedt
  - Professor, School of Electrical and Computer Engineering, Georgia Institute of Technology
  - Two research thrusts:
    - Control theory
    - Robotics
  - Research domains
    - Swarm robotics
    - Behavior-based control
    - Field robotics

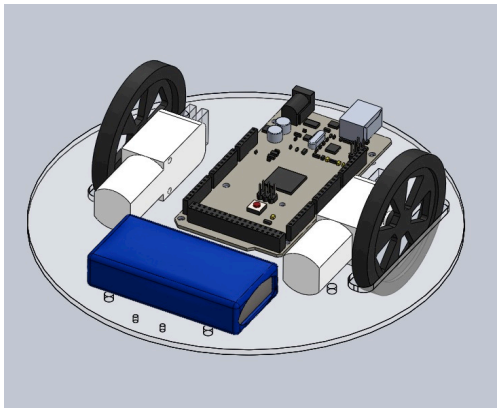
http://www.youtube.com/user/GRITSlab

Control of Mobile Robots

- What's in the course?
  - Control theory
  - Robot models
  - Mobility controllers
  - Applications
- What's not in the course?
  - AI
  - Perception
  - Mechanical engineering

Course Mechanics

- Lectures (8 each week, 7 weeks)
- Weekly quizzes + glue lectures
- Additional material
  - Sim.I.am MATLAB robot simulator
  - Build you own robot



A Disclaimer

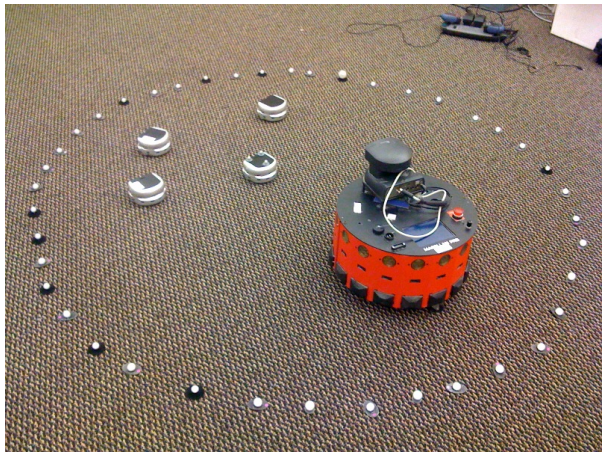
$$3(m_w + m_b)\dot{v} - m_b d\cos\phi\ddot{\phi} + m_b d\sin\phi\dot{\phi}^2 = \frac{1}{R}(\tau_L + \tau_R)$$

$$(3b^2 + \frac{1}{2}m_w r^2 + I_2)\ddot{\psi} + m_b d^2 \sin\phi\cos\phi\dot{\psi}\dot{\phi} = \frac{L}{R}(\tau_L - \tau_R)$$

$$-m_b d\cos\phi\ddot{v} + (-m_b d^2 - I_3)\ddot{\phi} + m_b d^2 \sin\phi\cos\phi\dot{\phi}^2 - m_b gd\sin\phi = \tau_L + \tau_R$$

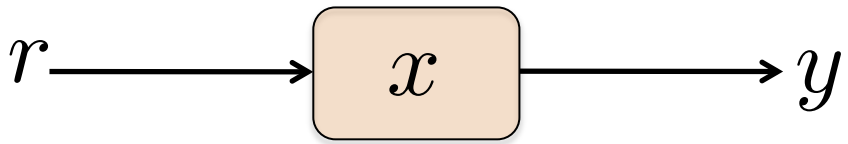
Lecture 1.2 – What's Control Theory, Anyway?

- System = Something that changes over time
- Control = Influence that change
- Examples:
  - **Robots**
  - Epidemics
  - Stock markets
  - Thermostats
  - Circuits
  - Engines
  - Power grids
  - Autopilots



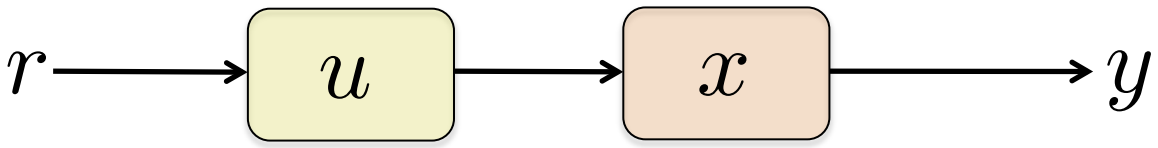
The Basic Building Blocks

- ***State*** = Representation of what the system is currently doing
- ***Dynamics*** = Description of how the state changes
- ***Reference*** = What we want the system to do
- ***Output*** = Measurement of (some aspects of the) system



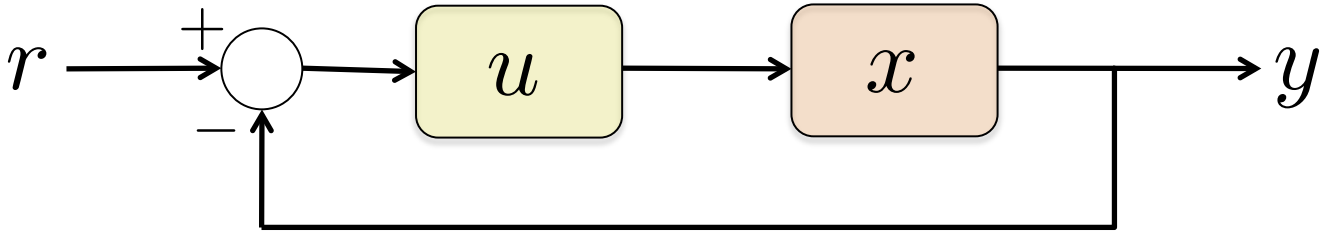
The Basic Building Blocks

- ***State*** = Representation of what the system is currently doing
- ***Dynamics*** = Description of how the state changes
- ***Reference*** = What we want the system to do
- ***Output*** = Measurement of (some aspects of the) system
- ***Input*** = Control signal



The Basic Building Blocks

- ***State*** = Representation of what the system is currently doing
- ***Dynamics*** = Description of how the state changes
- ***Reference*** = What we want the system to do
- ***Output*** = Measurement of (some aspects of the) system
- ***Input*** = Control signal
- ***Feedback*** = Mapping from outputs to inputs



Back to the Examples

- **Robots**
- Epidemics
- Stock markets
- Thermostats
- Circuits
- Engines
- Power grids
- Autopilots

Lecture 1.3 – On the Need for Models

- Control Theory = How pick the input signal u?
- Objectives:
  - Stability
  - Tracking
  - Robustness
  - Disturbance rejection
  - Optimality

Dynamical Models

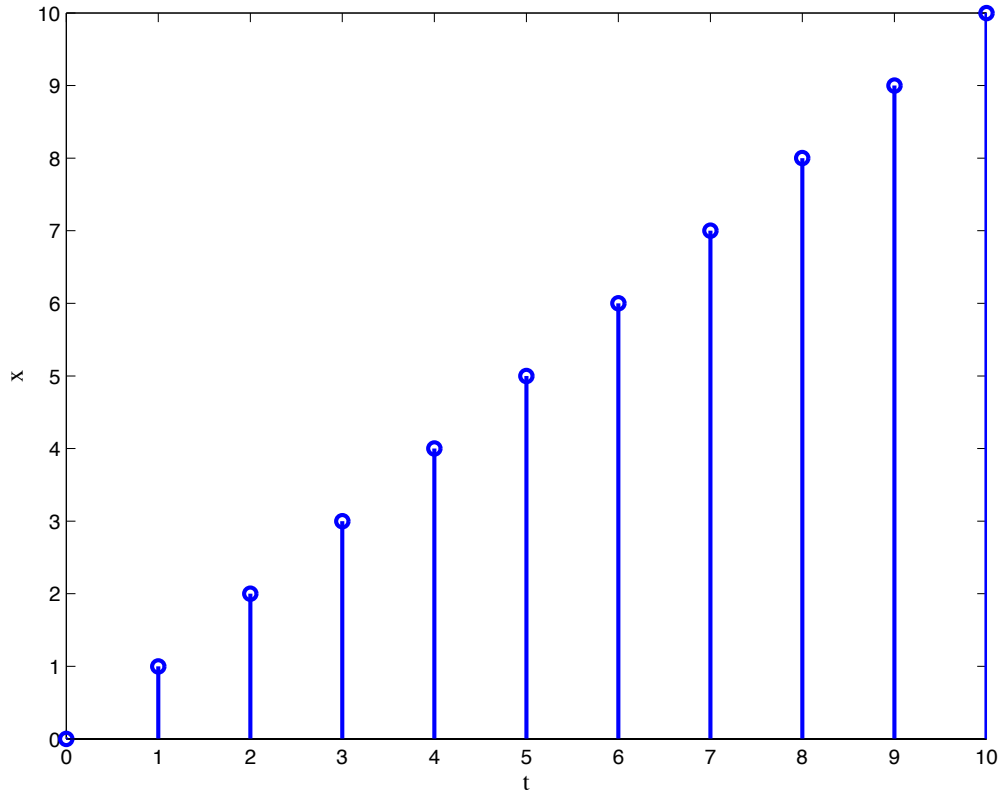
- Effective control strategies rely on predictive models
- Discrete time:

$$x_{k+1} = f(x_k, u_k)$$ ← Difference equation

Example: Clock

$$x_{k+1} = x_k + 1$$

Discrete Time Clock



Dynamics = Change Over Time

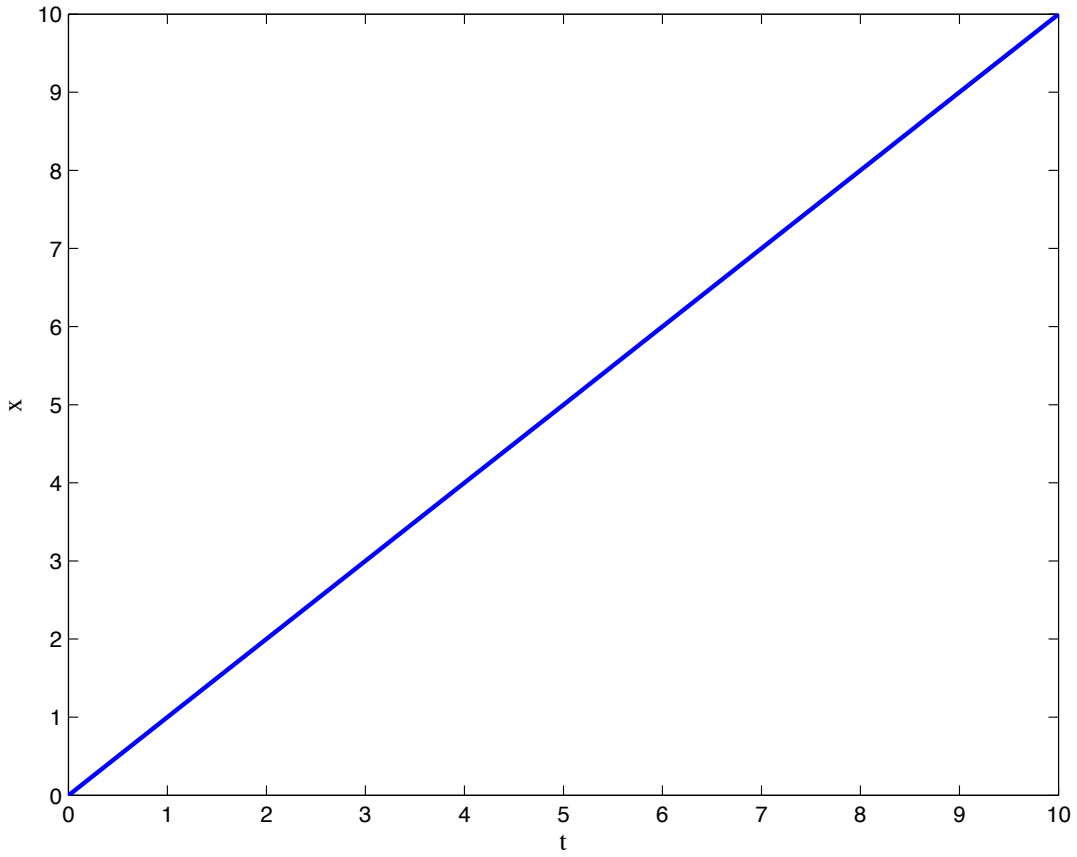
- Laws of Physics are all in continuous time: Instead of "next" state, we need derivatives with respect to time
- Continuous time:

$$\frac{dx}{dt} = f(x,u) \quad \sim \quad \dot{x} = f(x,u) \longleftarrow \text{Differential equation}$$

- Clock:

$$\dot{x} = 1$$

Continuous Time Clock



From Continuous to Discrete

- But in implementation, everything is discrete/sampled!

$$\dot{x} = f(x, u)$$

- Sample time: δt

$$x_k = x(k\delta t) \;\Rightarrow\; x_{k+1} = x((k+1)\delta t) = ???$$

$$x(k\delta t + \delta t) \approx x(k\delta t) + \delta t \dot{x}(k\delta t)$$

$$\boxed{x_{k+1} = x_k + \delta t f(x_k, u_k)}$$

Lecture 1.4 - Cruise-Controllers

- Make a car drive at a desired, reference speed (r)
- Newton's Second Law:

$$F = ma$$

- State: velocity (x)
- Input: gas/brake (u) $\quad F = cu$ (c=electro-mechanical transmission coefficient)
- Dynamics:

$$\dot{x} = a \;\Rightarrow\; m\dot{x} = cu \;\Rightarrow\; \dot{x} = \frac{c}{m}u$$


tgdaily.com
Urban Challenge
Sting Racing Crash

Control Design

- Assume that we measure the velocity ($y=x$)
- The control signal should be a function of r-y ($=e$)

- What properties should the control signal have?
  - Small e gives small u
  - u should not be "jerky"
  - u should not depend on us knowing c and m exactly

Lecture 1.5 – Control Design Basics

- Car Model:

$$\dot{x} = \frac{c}{m} u$$

- Want

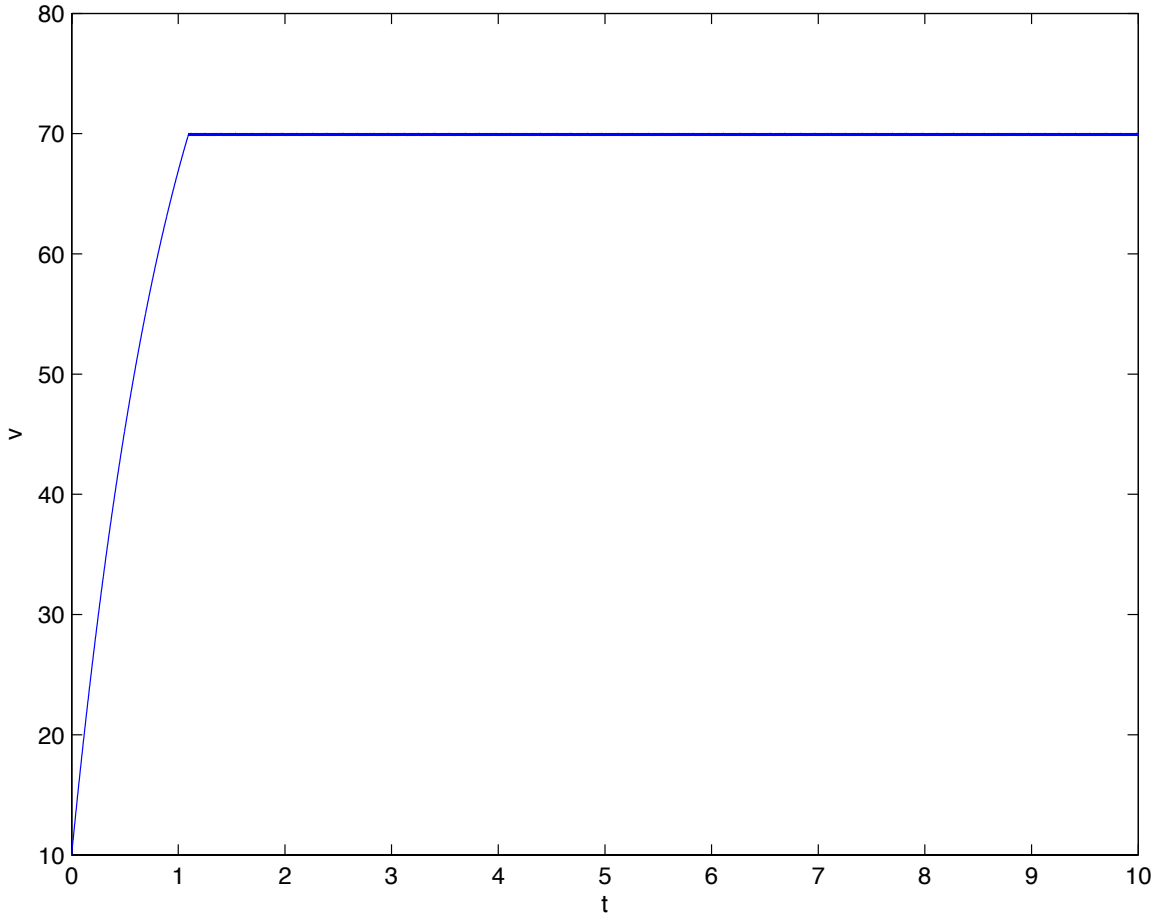
$$x \to r \text{ as } t \to \infty \quad (e = r - x \to 0)$$

- *Attempt 1*

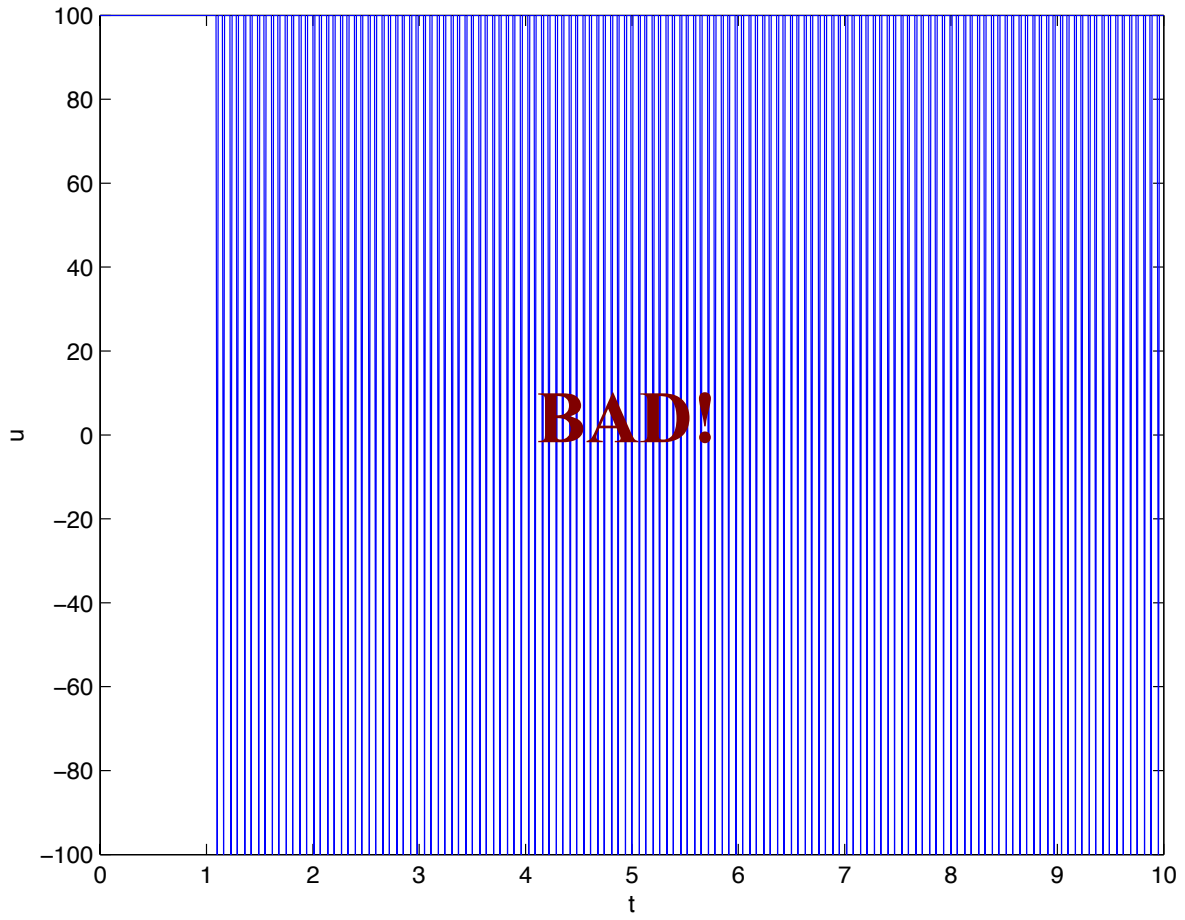
$$u = \begin{cases} u_{\max} & \text{if } e > 0 \\ -u_{\max} & \text{if } e < 0 \\ 0 & \text{if } e = 0 \end{cases}$$

Bang-Bang Control

r=70



Bang-Bang Control



Bumpy ride
Burns out actuators
….

P-Regulators

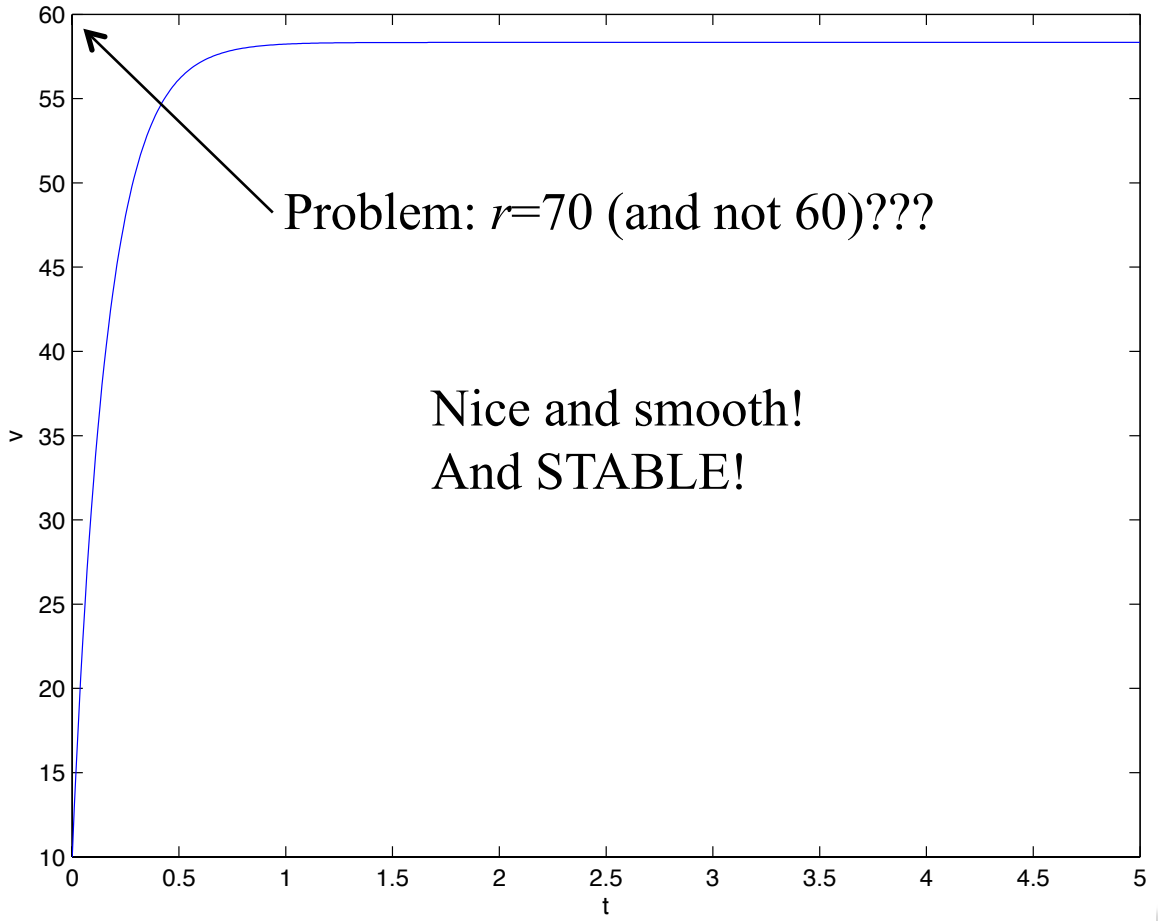
- Problem: The controller over-reacts to small errors

- *Attempt 2*

$$u = ke$$

  - Small error yields small control signals
  - Nice and smooth
  - So-called proportional regulation (P-regulator)

P-Regulator


Problem: r=70 (and not 60)???
Nice and smooth!
And STABLE!
v
t
60
55
50
45
40
35
30
25
20
15
10
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

Stability But Not Tracking

- Caveat: The "real" model is augmented to include a wind-resistance term:

$$\dot{x} = \frac{c}{m}u - \gamma x$$

- This is the model used for simulating the controller
- At steady-state (x does not change any more)

$$\dot{x} = 0 = \frac{c}{m}u - \gamma x = \frac{c}{m}k(r - x) - \gamma x \;\Rightarrow\; (ck + m\gamma)x = ckr$$

$$x = \frac{ck}{ck + m\gamma}r < r$$

Lecture 1.6 – Performance Objectives

- A controller should:
  - Stability (BIBO)
  - Tracking
  - Robustness

Problem: All of a sudden we have to know all these physical parameters that we typically do not know – *Not robust!*

$$\dot{x} = \frac{c}{m}u - \gamma x$$

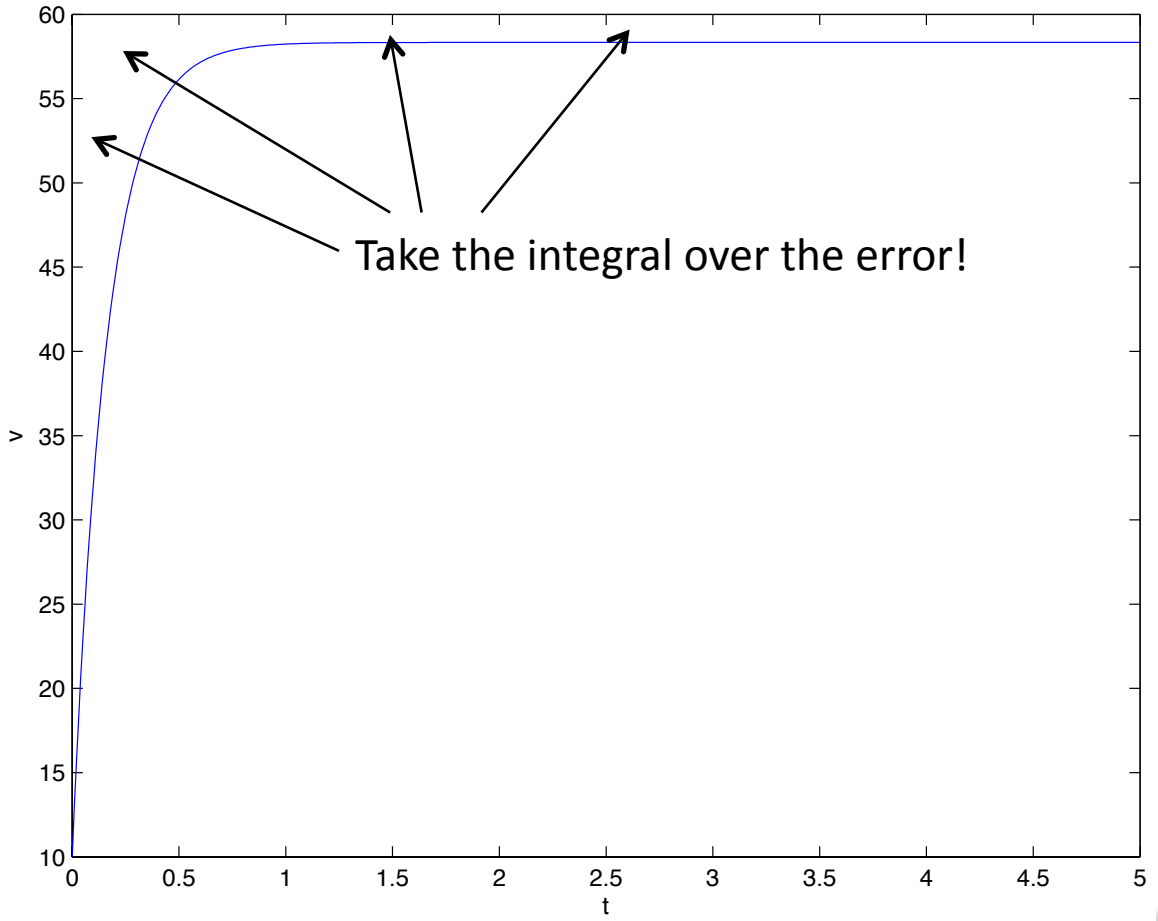
- *Attempt 3*

$$u = ke + \gamma\frac{m}{c}x$$

$$\dot{x} = 0 = \frac{c}{m}k(r - x) + \gamma x - \gamma x \;\Rightarrow\; x = r$$

Tracking!

P-Regulator


Take the integral over the error!
v
t
60
55
50
45
40
35
30
25
20
15
10
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5

PI-Regulators

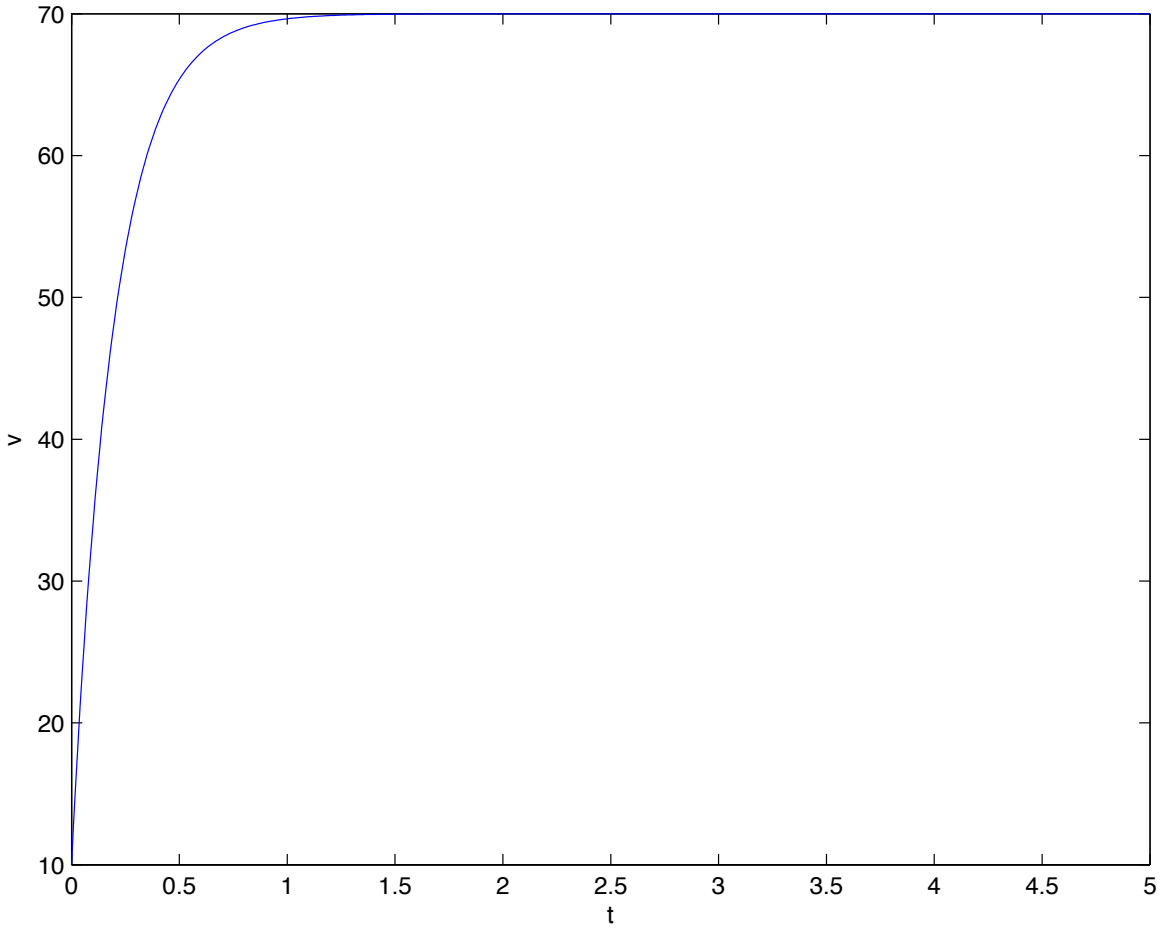
- *Attempt 4*

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau$$

- Or, why not a PID-Regulator?

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau + k_D \frac{de(t)}{dt}$$

PI-Regulator



Lecture 1.7 – PID Control

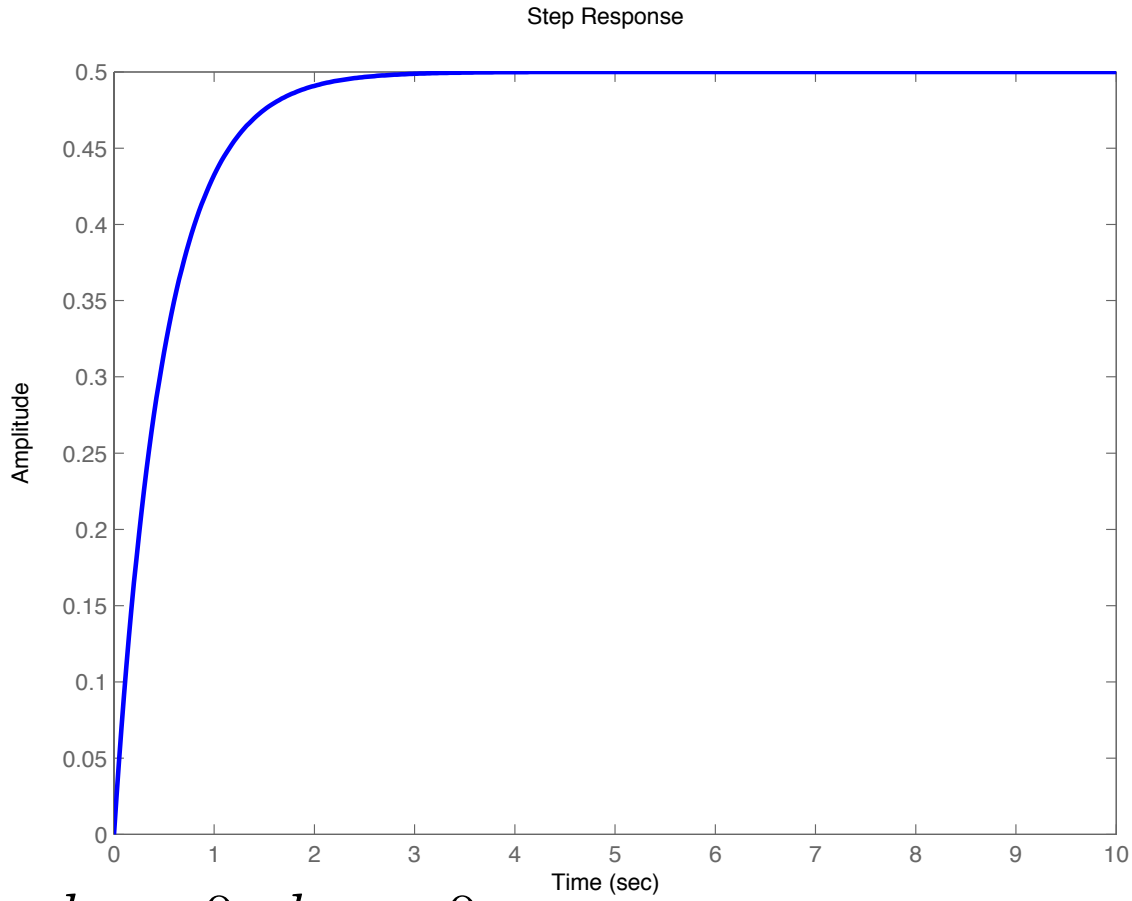
$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau + k_D \frac{de(t)}{dt}$$

- P: Contributes to stability, medium-rate responsiveness
- I: Tracking and disturbance rejection, slow-rate responsiveness. May cause oscillations
- D: Fast-rate responsiveness. Sensitive to noise
- PID: By far the most used low-level controller.
  - Note: Stability is not guaranteed.
- Feedback has a remarkable ability to fight uncertainty in model parameters!

Cruise-Controller (Again)

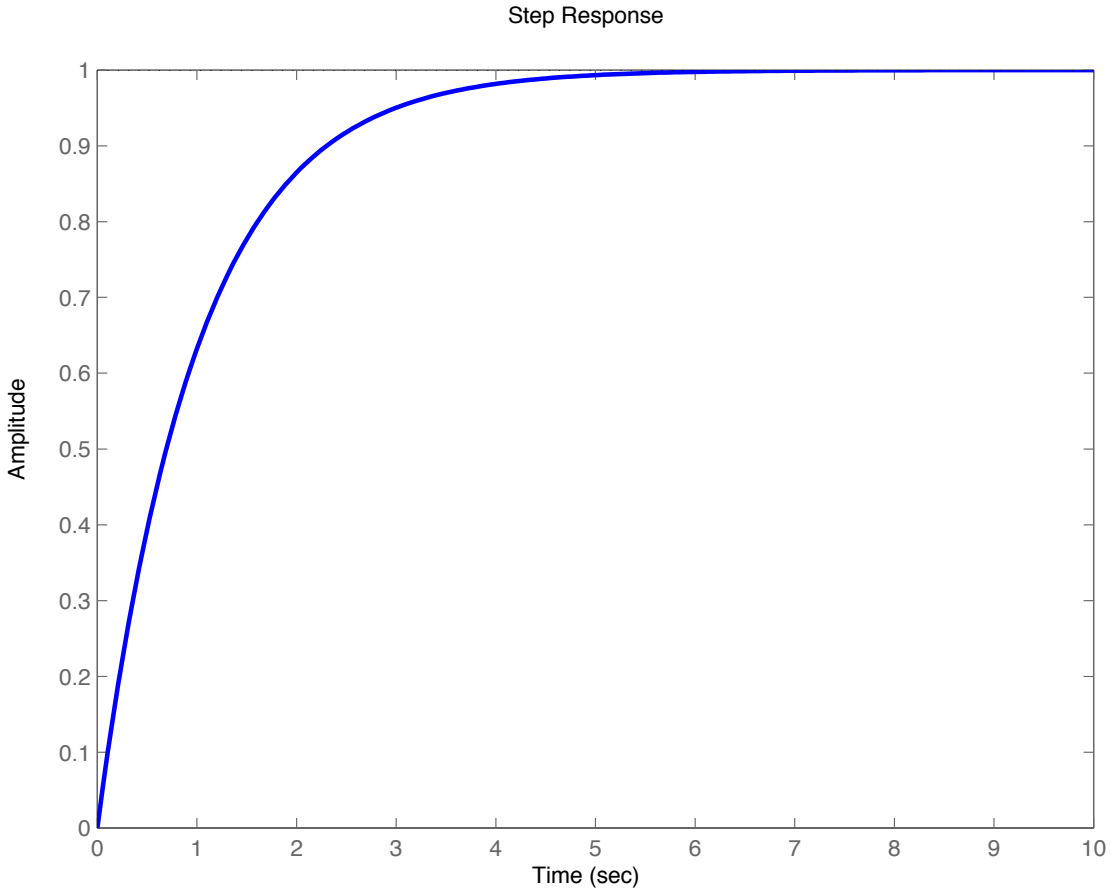
$$\dot{x} = \frac{c}{m}u - \gamma x \qquad\qquad c = 1,\ m = 1,\ \gamma = 0.1,\ r = 1$$

Cruise-Controller (Again)



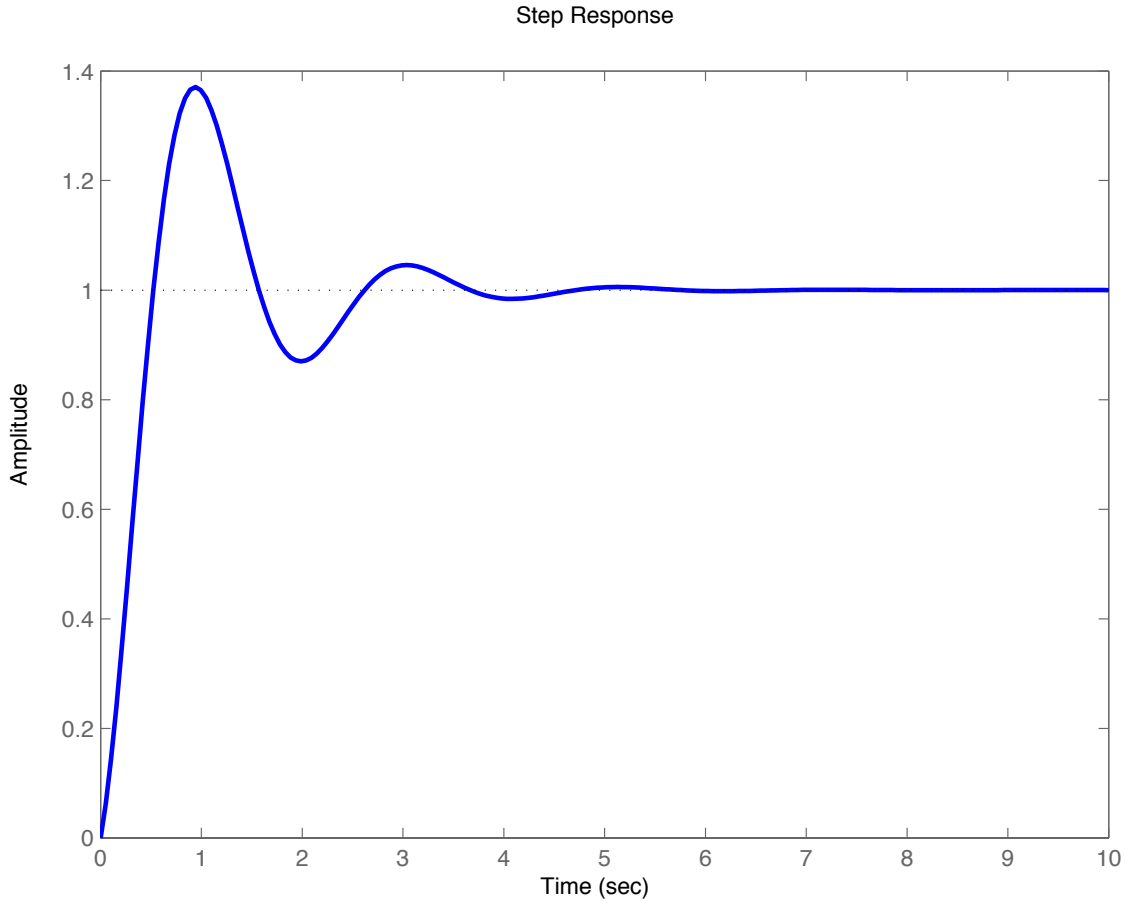
$$k_P = 1,\ k_I = 0,\ k_D = 0$$

Cruise-Controller (Again)



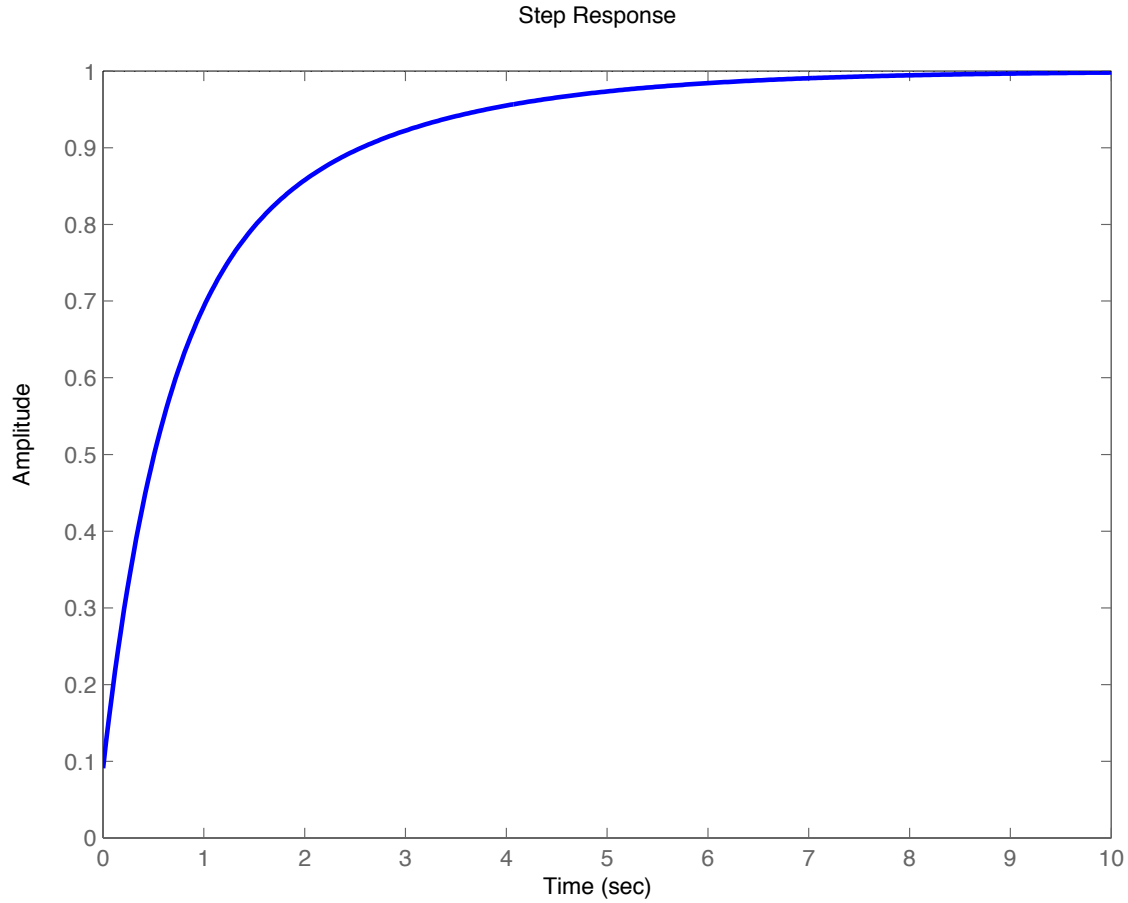
$$k_P = 1,\ k_I = 1,\ k_D = 0$$

Cruise-Controller (Again)



$$k_P = 1,\ k_I = 10,\ k_D = 0$$

Cruise-Controller (Again)



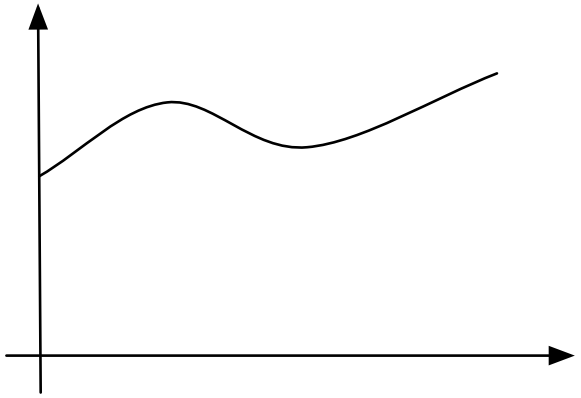
$$k_P = 1,\ k_I = 1,\ k_D = 0.1$$

Lecture 1.8 – Implementation

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau + k_D \frac{de(t)}{dt}$$

- How turn something as strange as an integral into something implementable?

Δt (sample time) $\quad \dot{e} \approx \frac{e_{new} - e_{old}}{\Delta t}$

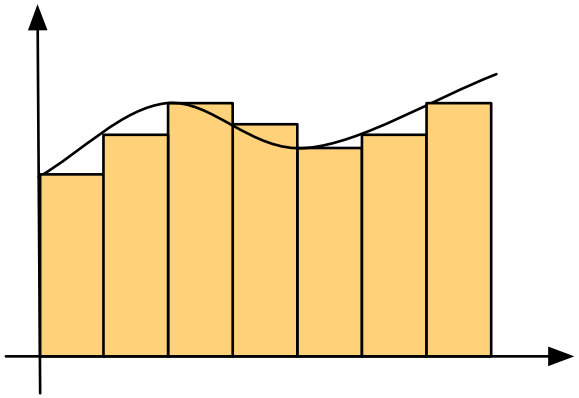


Lecture 1.8 – Implementation

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau + k_D \frac{de(t)}{dt}$$

- How turn something as strange as an integral into something implementable?

Δt (sample time) $\quad \dot{e} \approx \frac{e_{new} - e_{old}}{\Delta t}$

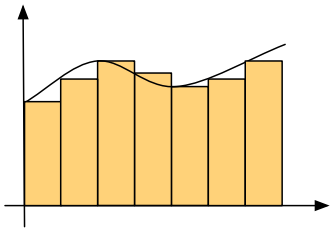


Lecture 1.8 – Implementation

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau + k_D \frac{de(t)}{dt}$$

- How turn something as strange as an integral into something implementable?

Δt (sample time) $\quad \dot{e} \approx \frac{e_{new} - e_{old}}{\Delta t}$



$$\int_0^t e(\tau) d\tau \approx \sum_{k=0}^{N} e(k\Delta t)\Delta t = \Delta t E$$

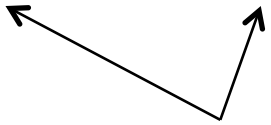
$$\Delta t E_{new} = \Delta t \sum_{k=1}^{N+1} e(k\Delta t) = \Delta t e((N+1)\Delta t) + \Delta t E_{old}$$

$$E_{new} = E_{old} + e$$

Implementation

- Each time the controller is called

```
read e;
e_dot=e-old_e;
E=E+e;
u=kP*e+kD*e_dot+kI*E;
old_e=e;
```



Note: The coefficients now include the sample time and must be scaled accordingly

Module 1: Introduction to Control

- That (almost) concludes Module 1
- Module 2: Mobile robots
- Module 3: What did we really do in Module 1?

Example: Quadrotor Altitude Control

$$\ddot{x} = cu - g$$

$$u = k_P e + k_I \int e d\tau + k_D \dot{e}$$