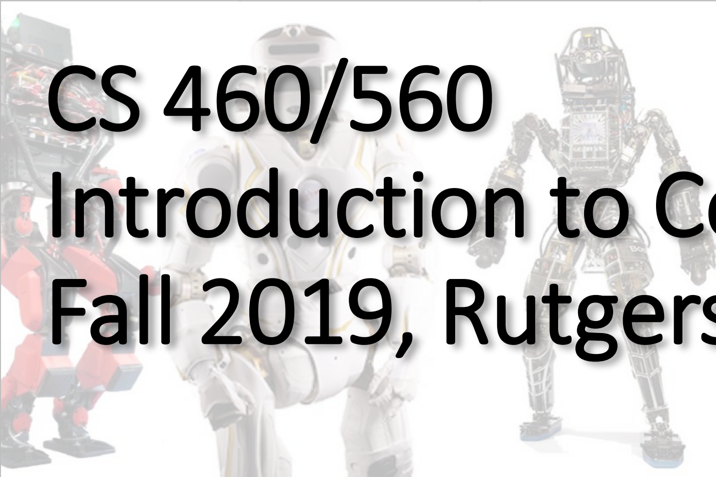
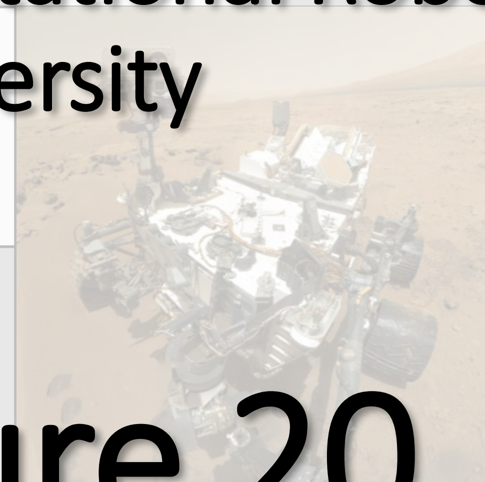
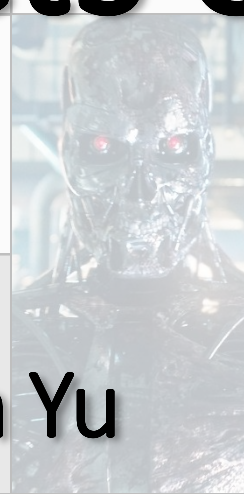
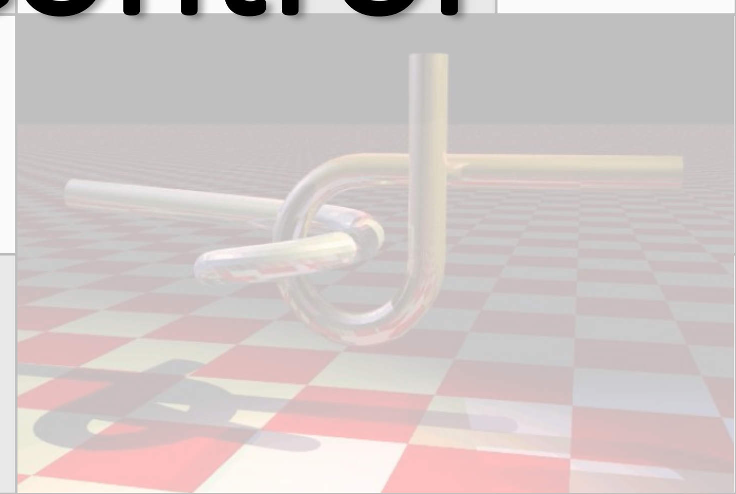
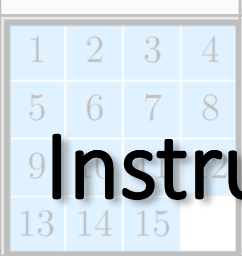
CS 460/560
Introduction to Computational Robotics
Fall 2019, Rutgers University






Lecture 20
Aspects of Control

Instructor: Jingjin Yu

Outline

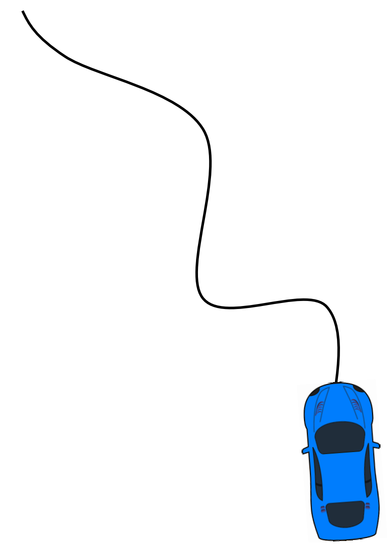
Feedback (closed-loop) control

- ⇨Mathematical models of dynamical systems
- ⇨Concept: open-loop v.s. closed-loop control
- ⇨Ubiquity of feedback control systems
- ⇨History of modern feedback control system: Watt's flyball governor
- ⇨PID control
  - ⇨ Basic concepts
  - ⇨ Behavior of individual terms
  - ⇨ Tuning
- ⇨Pure pursuit for controlling differential drive robots (DDR)

⇨Optimal control

- ⇨The Hamilton-Jacobi-Bellman equation, dynamic programming
- ⇨The maximum principle
- ⇨Time optimal trajectory of Dubin's car and DDR, bang-bang control

Modeling Dynamical Systems



A **dynamical system** (e.g., a car) is often modeled as

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

- x: the **state** of the system, $x(t)$ yields the **trajectory**
  - E.g., for a car, $x(t) = (x_1(t), x_2(t), \theta(t))$
  - $\dot{x} = \frac{dx(t)}{dt}$ is the time derivative, i.e., the velocity of the system
  - For a car, $\dot{x} = (\dot{x_1}, \dot{x_2}, \dot{\theta})$
- u: the **control input**
  - E.g., for a real car, approximately, $u = (\theta, v)$
    - θ is the front wheel bearing
    - v is the forward speed (for a 2-wheel drive, assuming no slippage)
  - u may be speed, acceleration, and so on…
- f: system **evolution** function
  - How do x, u determine $\dot{x}$

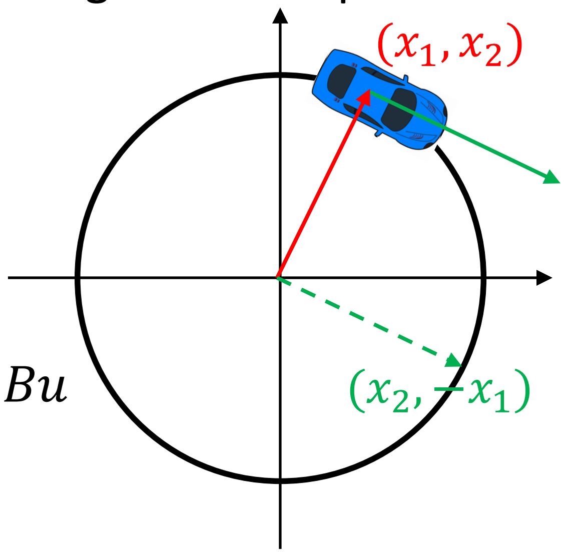
In **discrete** settings, often written as $x_t = f(x_{t-1}, u_{t-1})$

- May view this as integration of the continuous model: $x_t = x_{t-1} + \int_{t-1}^{t} \dot{x} dt$
- Often written as $x_k = f(x_{k-1}, u_{k-1})$

Modeling Dynamical Systems, Continued

Examples

- ⇨A car going at fixed speed along x_1-axis: $\dot{x_1} = 1$
  - ⇨ In this case, $f(x, u) = 1$ is a constant
- ⇨An accelerating car along x_1-axis with acceleration a: $\dot{x_1} = at$
  - ⇨ $u = a$, the acceleration, $f(x, u) = at$, does not depend on x
- ⇨A car going clockwise along the unit circle around the origin at unit speed
  - ⇨ $\dot{x} = (\dot{x}_1, \dot{x}_2, \dot{\theta}) = (x_2, -x_1, -1)$
  - ⇨ Initial condition: $x_1 = 1, x_2 = 0$
  - ⇨ The car will keep circling the unit circle at unit speed
  - ⇨ So it takes 2π time to go one round



Linear and non-linear systems

- ⇨Linear systems: f is a linear function, e.g., $\dot{x} = Ax + Bu$
- ⇨Non-linear systems: f is non-linear

What to grasp from the last two slides?

- ⇨Dynamical systems may be modeled as we have described
- ⇨In particular, given x_{k-1}, u_{k-1}, and $f(x, u)$, we can **predict** x_k

Open-Loop versus Closed-Loop Control

Open-loop control: the control input does not consider the current state of the system

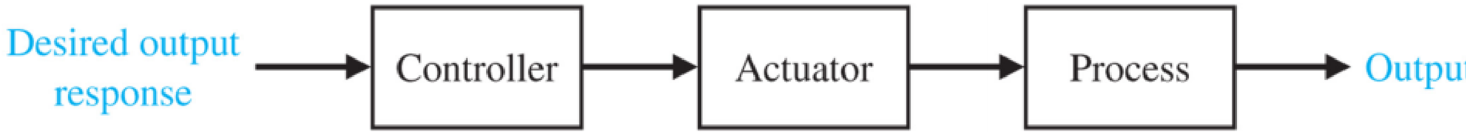
- ⇨ Roughly speaking, this is saying u is independent of x in $\dot{x} = f(x, u)$
- ⇨ Example: $u = \dot{x} = 0$
- ⇨ Example: $u = (\dot{x_1}, \dot{x_2}) = (1,1)$
  - ⇨ For the car we work with, give constant input signals to the two wheels

Feedback (closed-loop) control: the control input takes into account the (observed) system state $\tilde{x}$

- ⇨ Example: damping $u = \dot{x} = -\tilde{x}$
  - ⇨ What does this system do in one dimension?
  - ⇨ If $\tilde{x}$ is positive, $\dot{x}$ is then negative, causing x to decrease. So it takes the system to the origin
- ⇨ Example: regulator $u = \dot{x} = c - (\tilde{x} - ct)$
  - ⇨ What does this system do in one dimension?
  - ⇨ If $\tilde{x} - ct$ is positive, meaning we have gone too fast, the system will slow down
  - ⇨ Otherwise, the system will speed up
  - ⇨ Overall, system goes at a speed of c

System Block Diagram

Open-loop control



Feedback (closed-loop) control

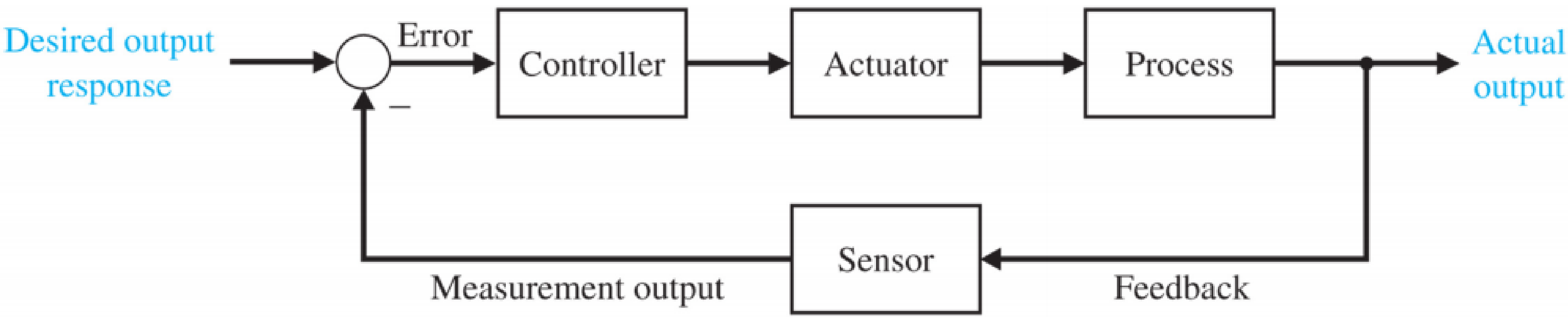


Image source: http://www.ece.mcmaster.ca/~davidson/

Real World Examples

Is a cannon open-loop or feedback-based?

⇨Open-loop: we do not maintain control after shooting the shell

What about a missile?

⇨Closed-loop: it tracks a target and is a form of PID control

In general, open-loop systems are cheaper than feedback-based system

Image sources: www.army.mil, wikipedia.org

Ubiquity of Feedback Control Systems

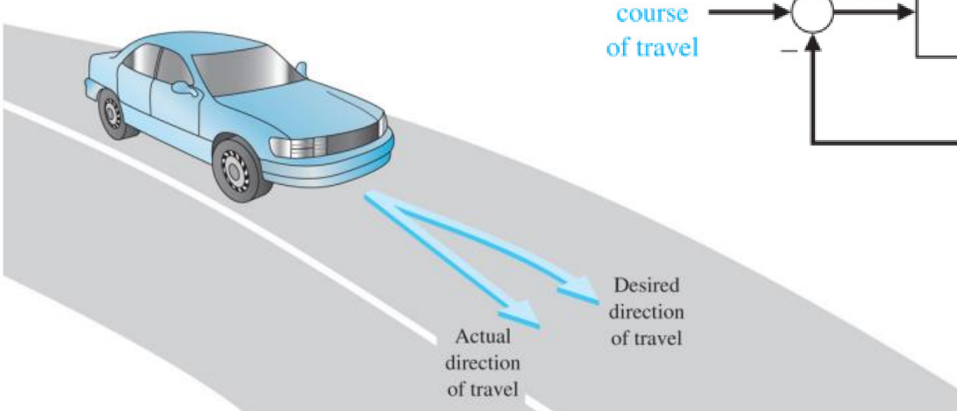
Feedback control systems are everywhere!

Example: ourselves

- ⇨ Walking with eyes open or closed
- ⇨ Reaction to electrical shock
- ⇨ Regulation of glucose (blood sugar) level with insulin (pancreas) and glycogen
  - ⇨ Sugar crash happens if you overload the system

Example: room temperature control (thermostat)

Example: driving a car



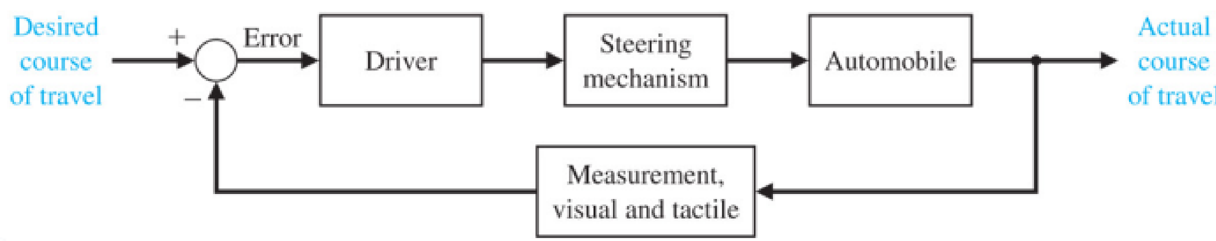


Image source: http://www.ece.mcmaster.ca/~davidson/

A Little History on Modern Feedback Control

After steam engine was invented, how to control its running speed is a problem of major interest

A successful design was Watt's flyball governor

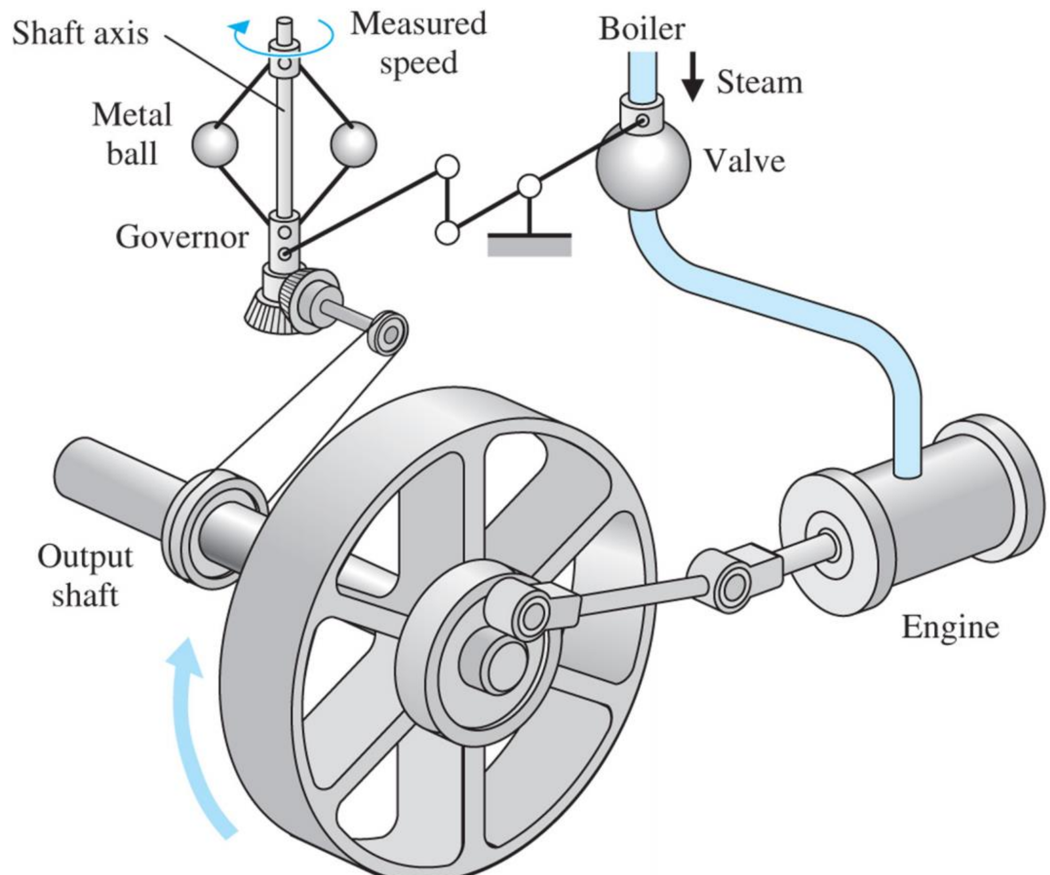


Image source: http://www.ece.mcmaster.ca/~davidson/

PID Controller

PID controller stands for **proportional-integral-derivative controller**

⇨There are many different "theoretical" feedback controllers

⇨However, the final implementation often uses some form of PID control

General form:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

Block diagram

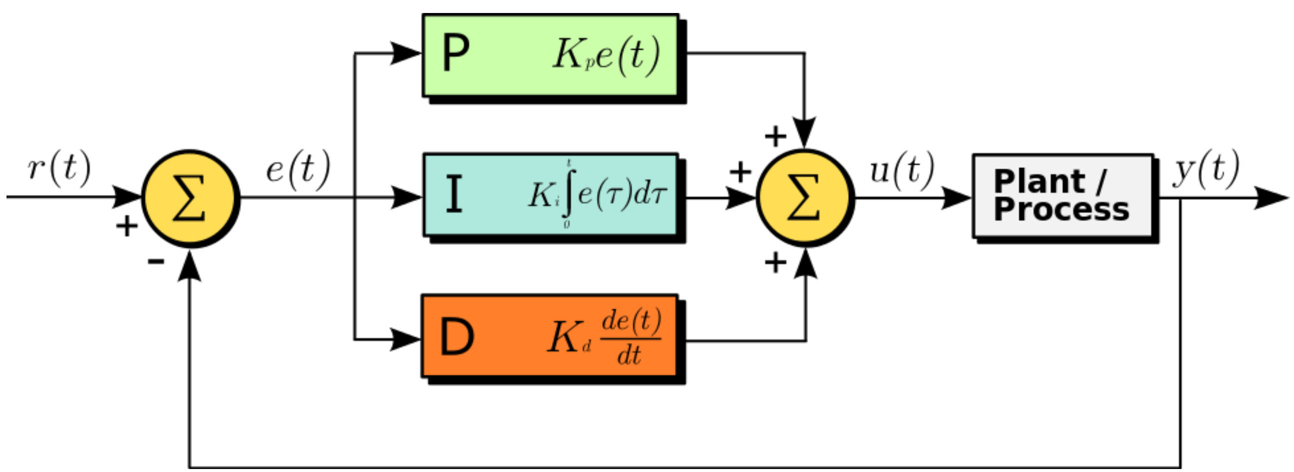
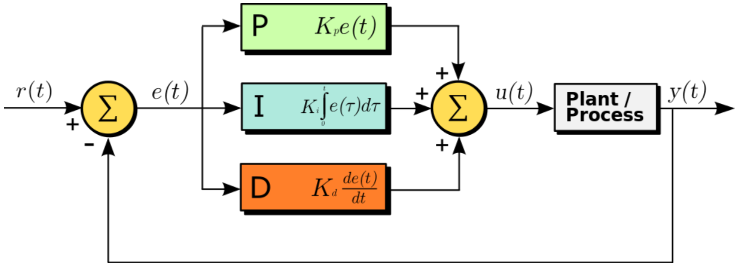


Image source: wikipedia.org

PID Controller Breakdown



$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de(t)}{dt}$$

The error term $e(t)$

- ⇨ $e(t) = Set\ Point\ - Current\ Location$
- ⇨ E.g., if we want to go to the origin, may set $e(t) = (0,0) - (x_1(t), x_2(t))$

The proportional term, $K_p e(t)$

- ⇨ Adjust control based on **instant position error**
- ⇨ K_p: proportional gain, usually a positive constant
- ⇨ $e(t)$ large, then $u(t)$ is large

The integral term, $K_i \int_0^t e(\tau)d\tau$

- ⇨ Adjust system behavior based on **cumulative error**, slow response
- ⇨ Accelerates convergence to set point
- ⇨ K_i: integral gain

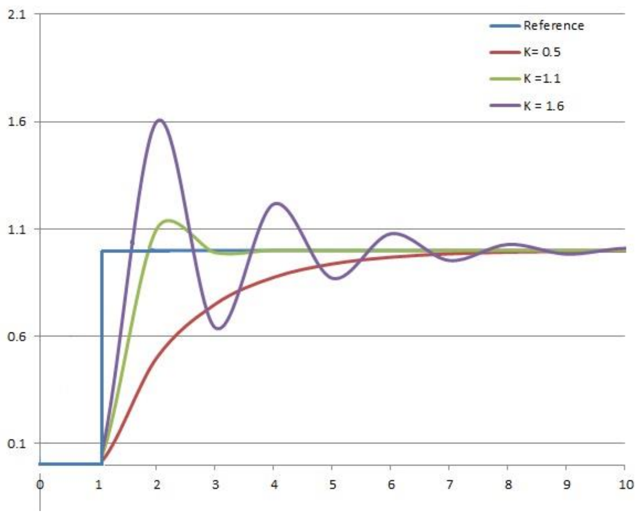
The derivative term, $K_d \frac{de(t)}{dt}$

- ⇨ Adjust system behavior based on **differential error**, fast response (predictive)
- ⇨ Generally provides damping (preventing overshot), improves stability
- ⇨ K_d: derivative gain

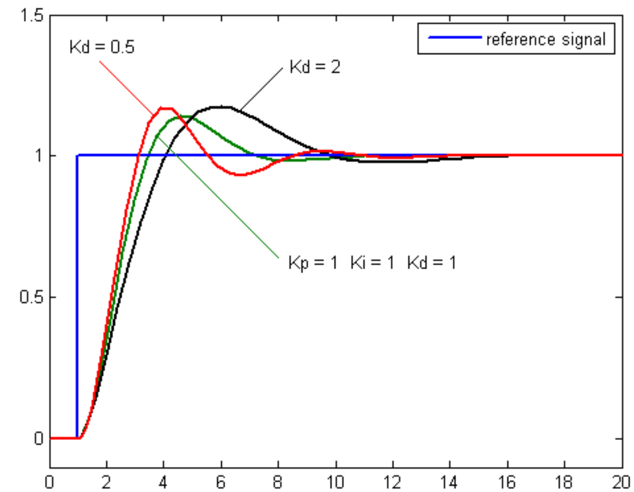
Image source: wikipedia.org

PID Controller Breakdown Example

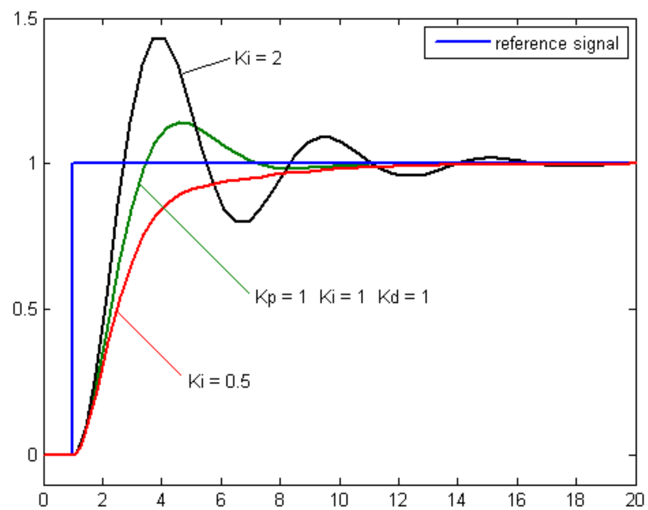
Effects of varying the gains in a 1D system



Varying K_i



Varying K_p



Varying K_d

Image source: wikipedia.org

PID Controller Tuning

PID control applies easily to many systems since $e(t)$ can often be computed easily

However, the process of tuning a PID controller can be tricky

General (manual) method

- ⇨Set $K_i = K_d = 0$ and tune K_p until the output oscillates
- ⇨Adjust K_i so that the system converges to set point
- ⇨Adjust K_d to remove oscillation until acceptable

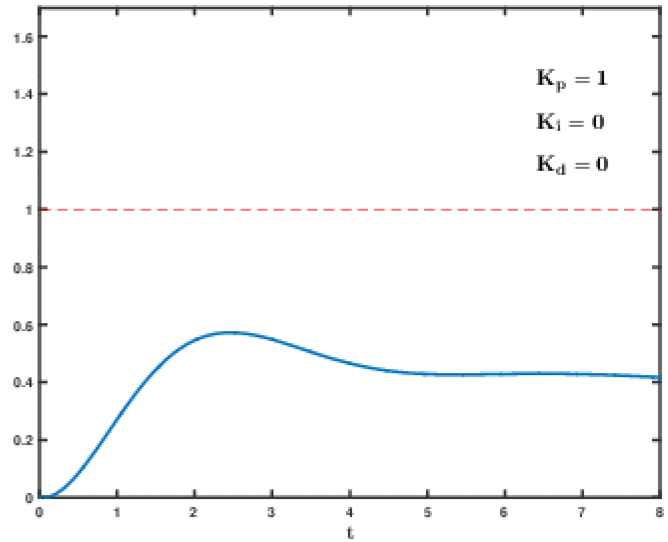
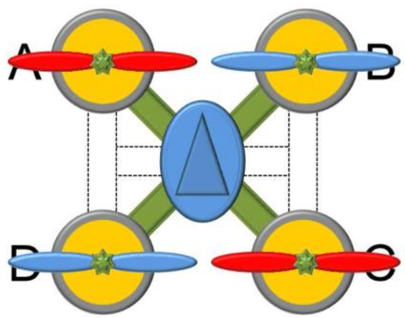
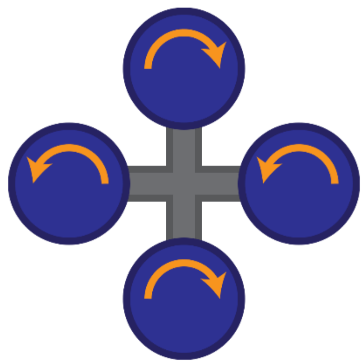


Image source: wikipedia.org

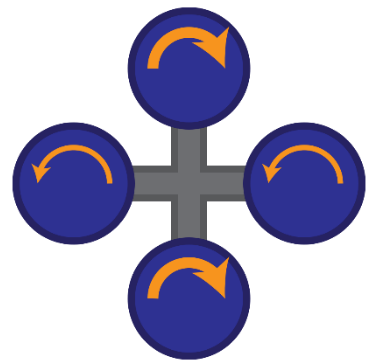
Controlling a Quadcopter

How does a quadcopter work?

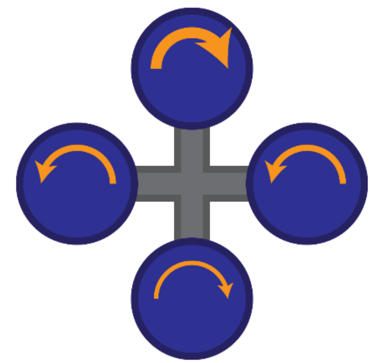




Up/down



Yaw change



Pitch/roll change

Quadcopter control

- ⇨Hover
- ⇨Basic trajectory following
  - ⇨ Doing a sequence of “hovering” with new set points

Image source: wikipedia.org

Pure Pursuit for Differential Drive Robots

Most two wheeled robots can be viewed as a **differentially driven robot** (DDR)

⇨ Two wheel inputs in the range of $[-1, 1]$

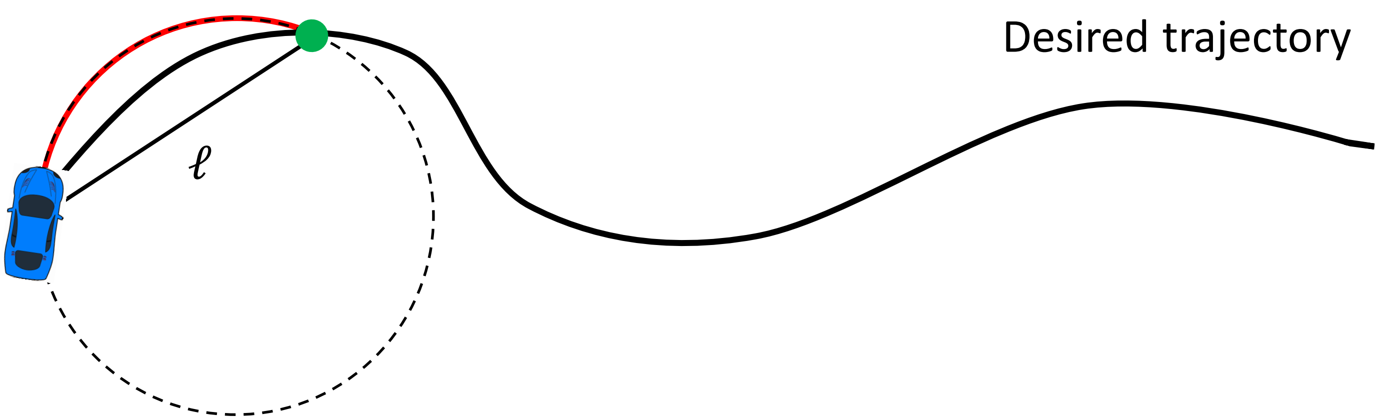
Pure pursuit path following algorithm

⇨ From the current location of car, locate a waypoint of distance ℓ (some constant) on the desired trajectory

⇨ Compute the required curvature to the waypoint

⇨ Adjust wheel speeds to follow the computed arc

⇨ Note: the car's direction is tangential to the computed arc



Value Function and Principle of Optimality

Cost of trajectories can be captured with the **functional**

$$J(t,x,u) = \int_t^{t_1} L\big(s, x(s), u(s)\big)ds + K(x(t_1))$$

The **value function** is defined as the infimum of J over control u

$$V(t,x) := \inf_{u_{[t,t_1]}} J(t,x,u)$$

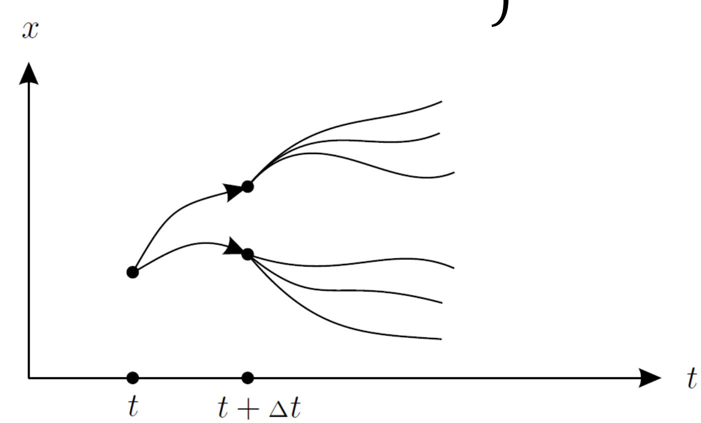
Principle of optimality states: For every $(t,x) \in [t_0, t_1) \times R^n$ and every $\Delta t \in (0, t_1 - t]$, the value function satisfies

$$V(t,x) = \inf_{u_{[t,t+\Delta t]}} \left\{ \int_t^{t+\Delta t} L\big(s, x(s), u(s)\big)ds + V\big(t+\Delta t, x(t+\Delta t)\big) \right\}$$

Intuitive (provable)

But important

Yes, related to dynamic programming in CS



The Hamilton-Jacobi-Bellman (HJB) equation

From the principle of optimality, HJB equation can be derived

$$-\widehat{V}_t(t,x) = \inf_{u \in U}\{L(t,x,u) + \langle \widehat{V}_x(t,x), f(t,x,u)\rangle\}$$

⇨A PDE providing **necessary** and **sufficient** conditions for optimal control u^*

⇨From u^* one can also obtain the optimal trajectory x^*

⇨From HJB, can derive conditions for optimal solutions

E.g., a simple integrator $\dot{x} = u$ with $L(x,u) = x^2 + u^3$. HJB

$$-V_t(t,x) = \inf_{u \in \mathbb{R}}\{x^2 + u^3 + V_x(t,x)u\}$$

Optimal control: $$u^*(t) = -\sqrt{\frac{1}{3}V_x(t,x)}$$

PDE: $$-V_t(t,x) = x^2 - 2\left(\frac{1}{3}V_x(t,x)\right)^{\frac{3}{2}}$$

In general, closed form solutions for HJB are difficult to come by

Pontryagin's Maximum Principle

HJB provides necessary and sufficient conditions for optimality

- ⇨But does so requiring the value function $V(t, x)$ be C^1
- ⇨That is, first order partial derivatives must be continuous

Pontryagin's maximum principle address this.

- ⇨The (fixed end point problem) setup: $J(u) = \int_{t_0}^{t_f} L(x, u)dt + K(t_f, x_f)$ and $\dot{x} = f(x, u)$
- ⇨Hamiltonian: $H(x, u, p, p_0) \coloneqq \langle p, f(x, u)\rangle + p_0 L(x, u)$
- ⇨Maximum principle says that
  - ⇨ There exist $(p^*, p_0) \neq (0,0)$ such that $\dot{x}^* = H_p(x^*, u^*, p^*, p_0^*)$, $\dot{x}^* = H_p(x^*, u^*, p^*, p_0^*)$
  - ⇨ There exists global maximum $H(x^*(t), u^*(t), p^*(t), p_0^*) \geq H(x^*(t), u(t), p^*(t), p_0^*)$
  - ⇨ $H(x^*(t), u^*(t), p^*(t), p_0^*) = 0$ for all $t \in [t_0, t_f]$

A bit of history

- ⇨HJB equation: 1957, from US
- ⇨Maximum principle: 1956, from USSR
- ⇨The cold war era…

Optimal Trajectories for Dubins Car, DDR

Bang-bang control

- ⇨Both HJB and the maximum principle lead to **bang-bang** control
- ⇨In general, optimal trajectory uses extreme control inputs

E.g., moving from $x = 0$ to $x = 1$ with $\dot{x} = u \in [-1, 1]$

- ⇨What is the time optimal strategy?
- ⇨Move with $\dot{x} = 1$
- ⇨What if $\ddot{x} = u \in [-1, 1]$?
- ⇨Move with $\ddot{x} = 1$ halfway, then $\ddot{x} = -1$

E.g. Dubins car

- ⇨Three types of moves: L, S, R
- ⇨A distance optimal solution uses at most a sequence of three
- ⇨Similar results applies to DDR, e.g. extremal

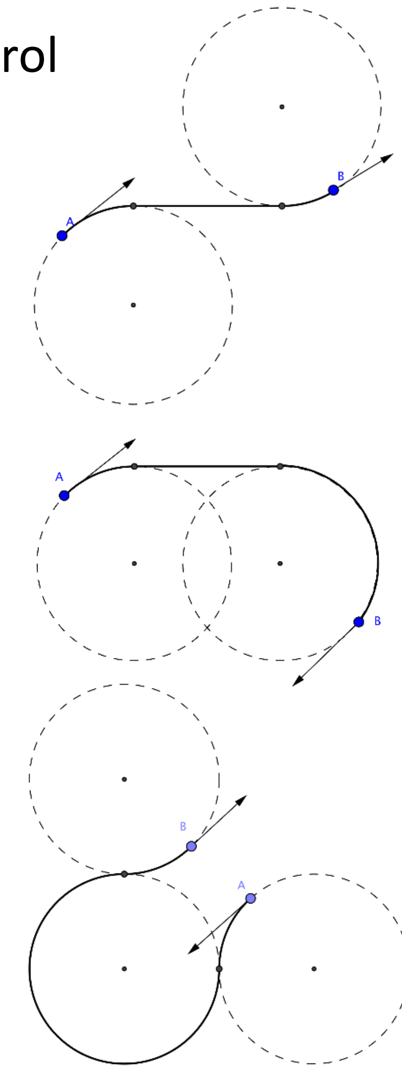


Image source: wikipedia.org