F1TENTH Autonomous Racing

Theme: Control

Introduction to Optimal Control

Ahmad Amine
University of Pennsylvania
aminea@seas.upenn.edu

Rahul Mangharam
University of Pennsylvania
rahulm@seas.upenn.edu

F1 TENTH

Safe Autonomy Lab
University of Pennsylvania

Penn Engineering

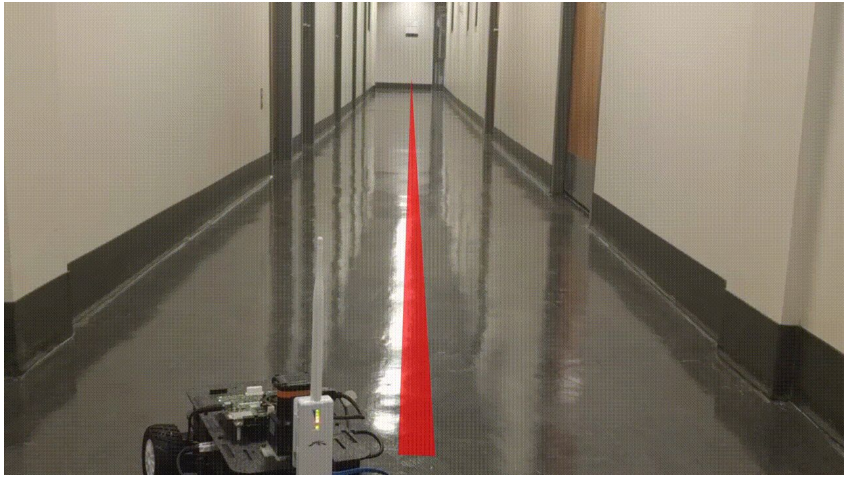
Lecture Content

1. Control Systems Basics
2. Introduction to Optimal Control
3. Introduction to Convex Optimization

Control Systems Basics

Recap - Controller's We've Seen So Far

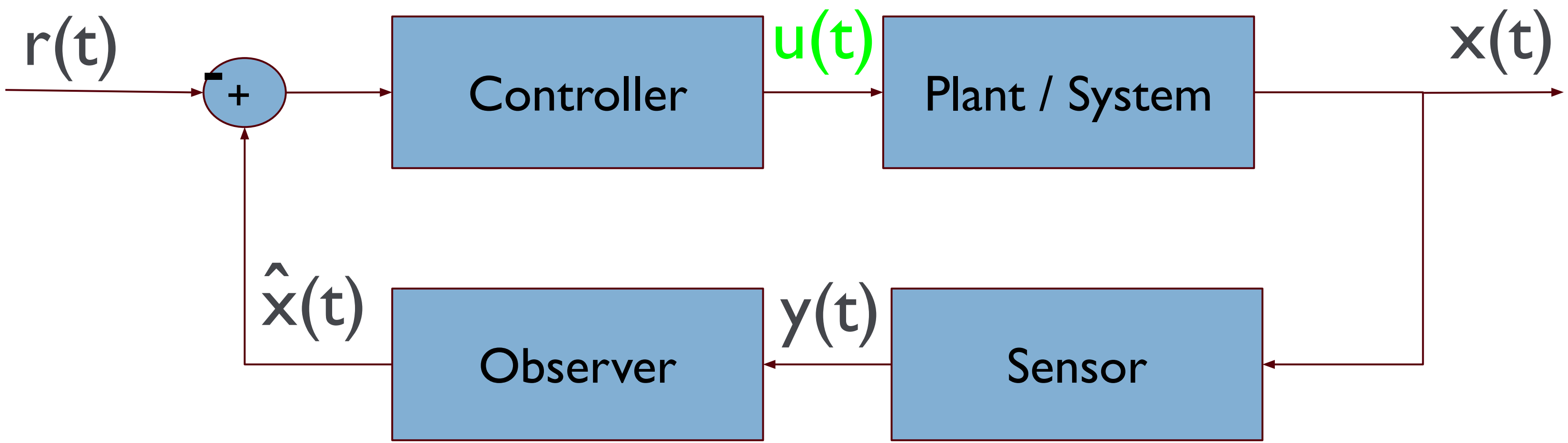
Pure Pursuit - Geometric



PID Control - Model Free

Common Elements Between Controllers?

Control Systems Design



"Control governs, or regulates, how the system behaves or functions."
~ IEEE Control Systems Society

The Goal as a Control Theorist / Engineer: Design a **SYSTEM** to track a desired reference signal => Output of Control Design is the System that produces u(t), not u(t)

Looks Familiar?

Inputs
(e.g. steering wheel angle,motor torque)

Real Vehicle

Outputs
(e.g. speed, accelerations)

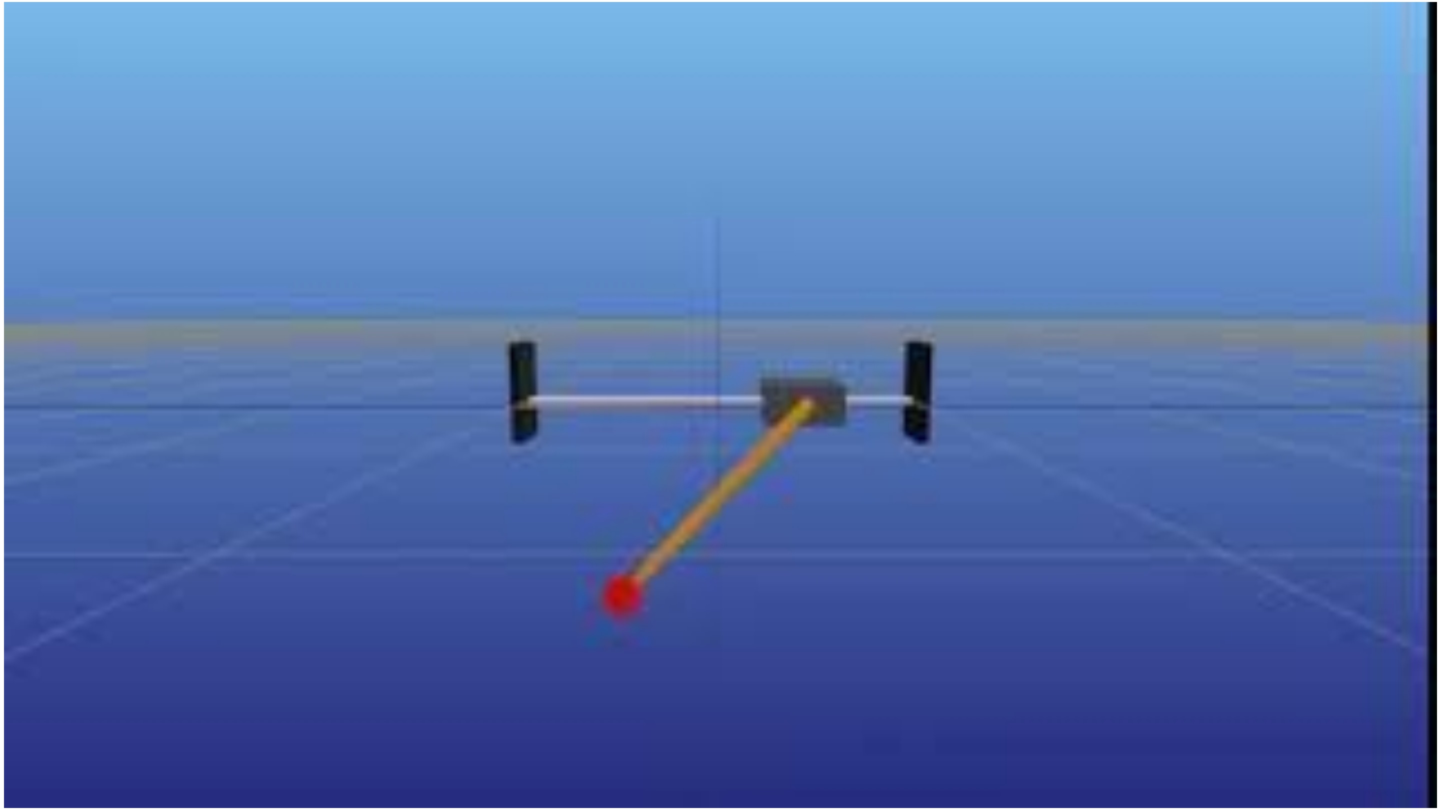
Simplified Representation

Model Accuracy and Mismatch

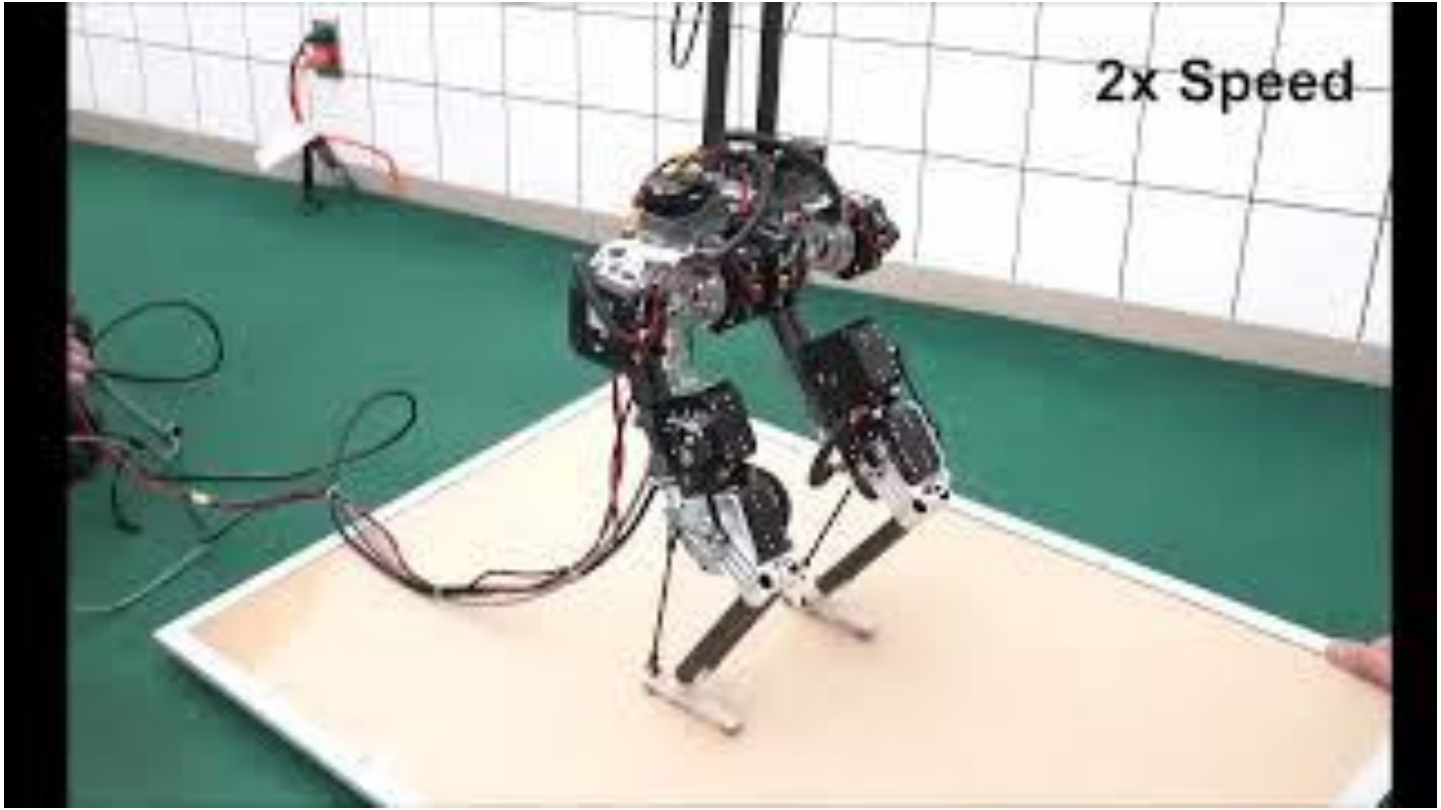
Vehicle dynamics model

- Differential equations
- Characteristic curve correlations

Control Systems Examples

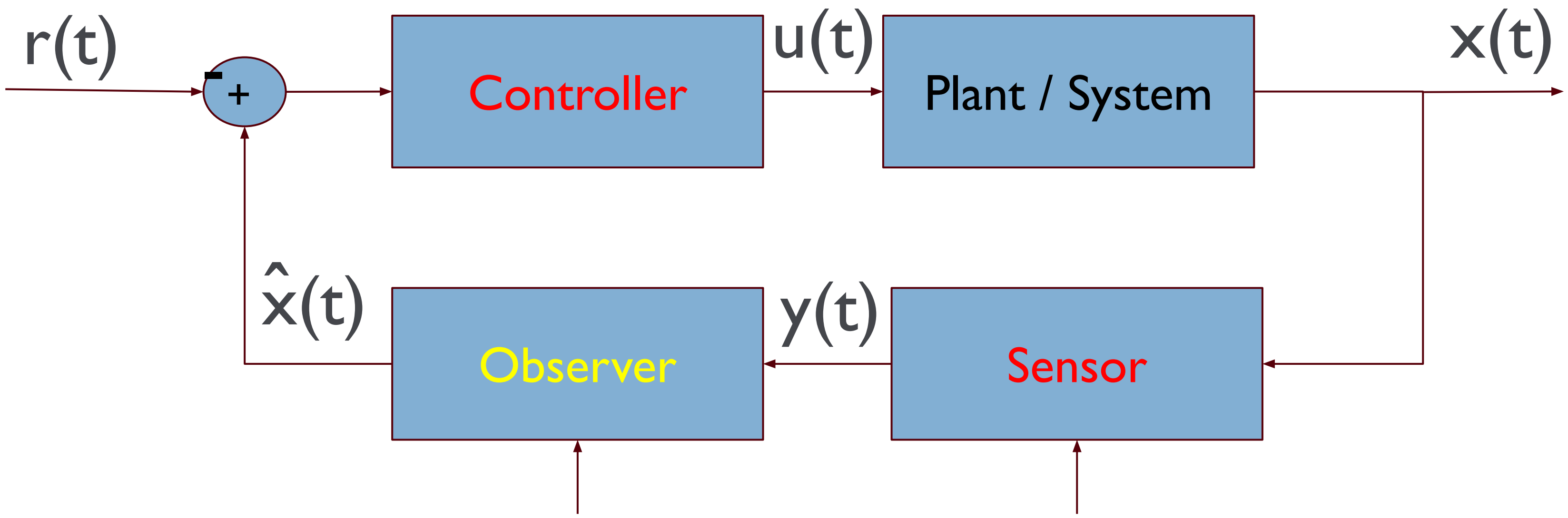


Control Systems Examples



Control Systems Examples

What is Needed for Control?

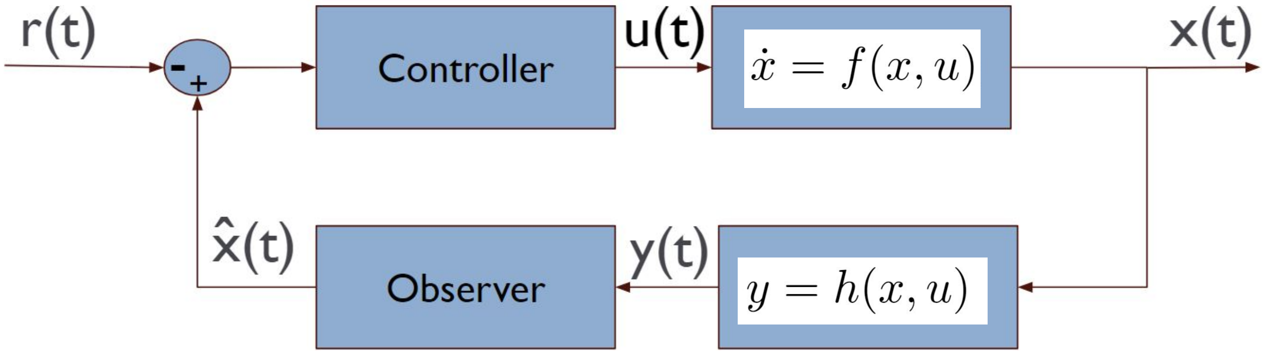


Optional if we can measure all states of a system

Needed for close-loop control

Penn Engineering

Terminology



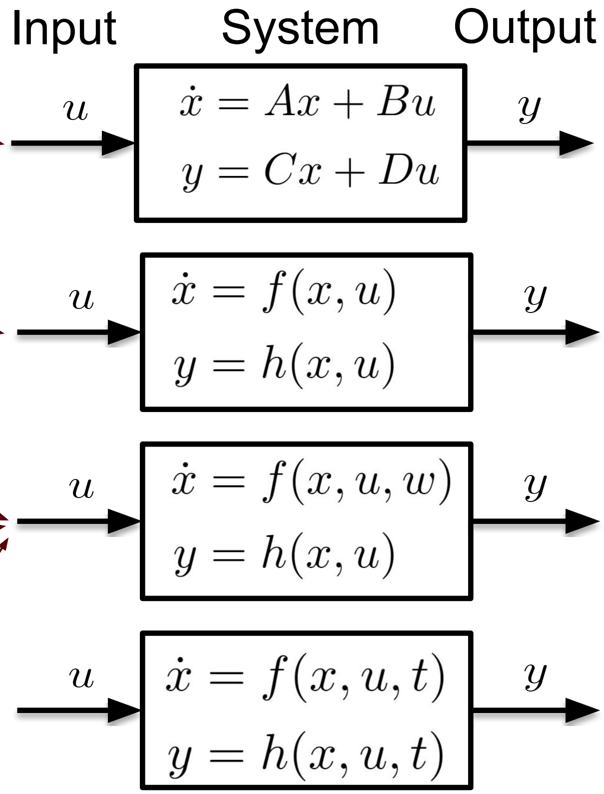
- System Dynamics: $f(x, u)$
- (System) Control Input: $u(t)$
- Sensor (Observation) Dynamics: $h(x, u)$
- Reference Signal: $r(t)$
- State Estimate: $\hat{x}(t)$

We went over this in Lecture 7

Penn Engineering

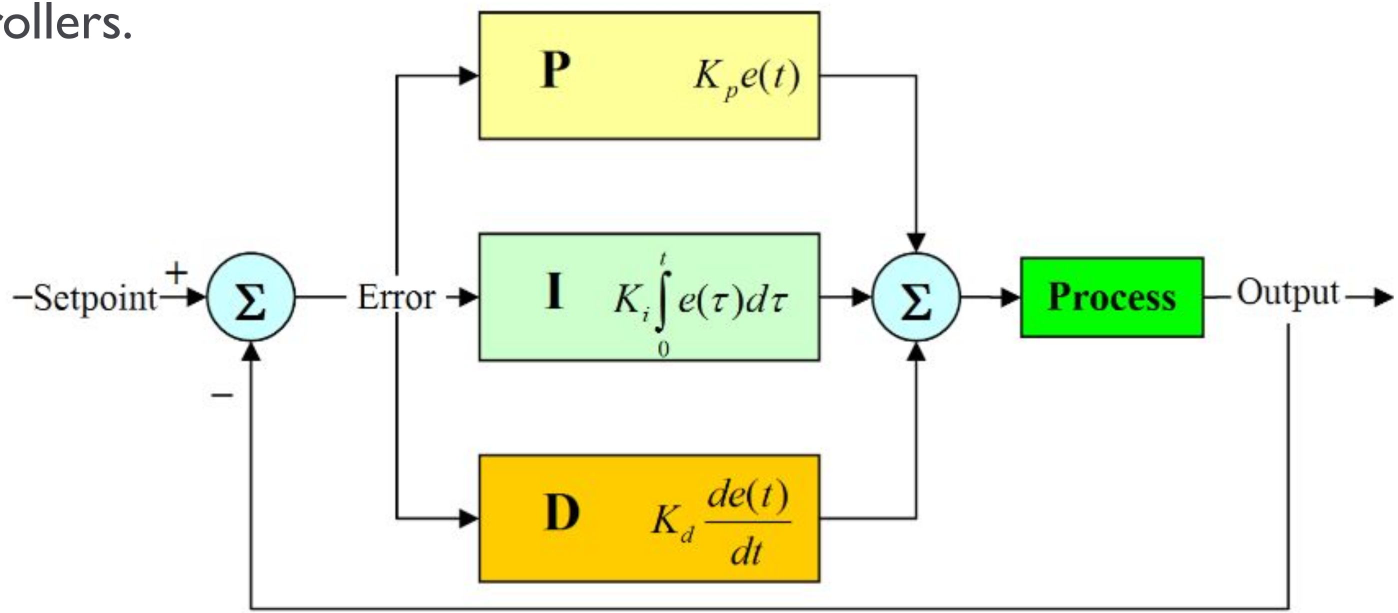
Types of Controllers

- **Multivariable Control:** Multiple Inputs and / or Outputs
- **Nonlinear Control:** Nonlinear dynamics, harder to ensure analytic results
- **Stochastic Control:** Minimize variance of output for stochastic systems
- **Robust Control:** Ensure specification satisfaction under noise / disturbances
- **Adaptive Control:** Real-time adaptation of controller parameters or system model.



Where does PID control fit?

- PID controllers belong to the class of linear single-input single-output controllers.



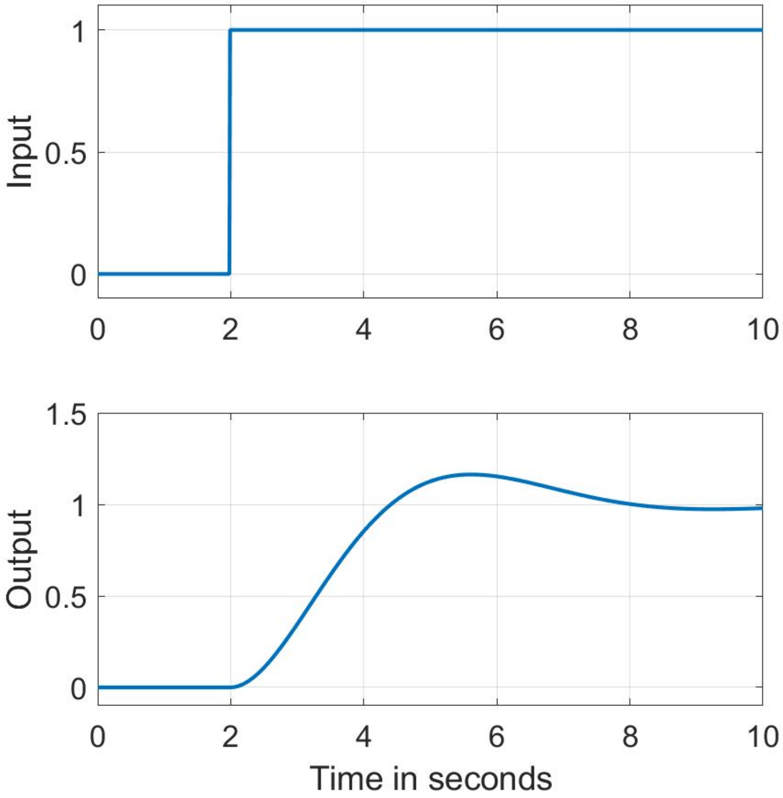
$$u(t) = K_p e(t) + K_I \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

Penn Engineering

PID Control Design Steps

1. Identify the system:
   - As the system is linear, single-input single-output, then the system can be easily identified using a step-response graph
   - Usually identified in frequency domain (laplace transforms)

$$H(s) = \frac{Y(s)}{X(s)}$$



PID Control Design Steps

2. List out specifications:

   a. Percentage Overshoot

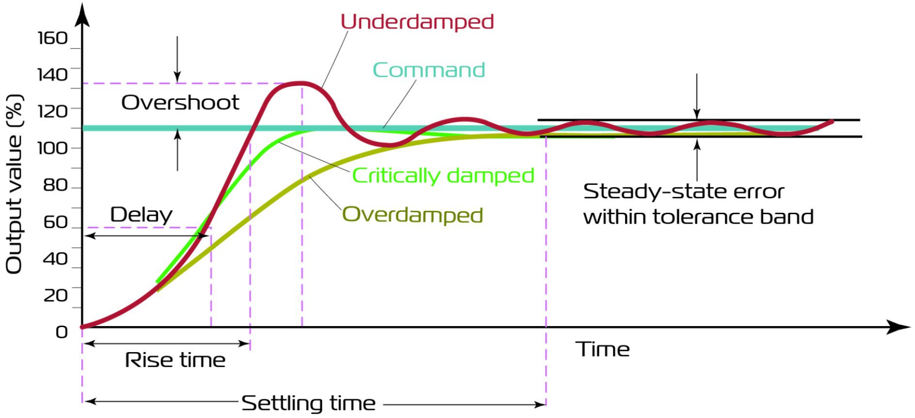
   b. Percentage Undershoot

   c. Percentage Steady-State Error

   d. Rise Time

   e. Settling Time

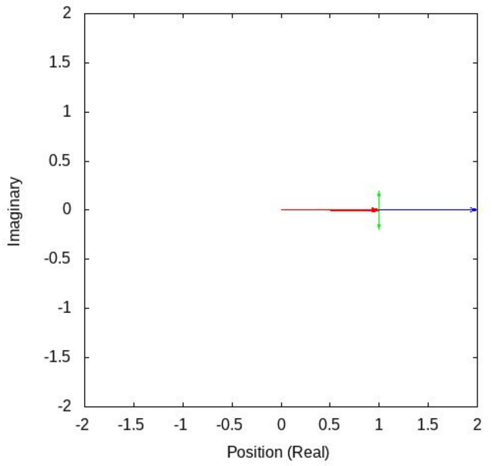
   etc…



PID Control Design Steps

3. Design the Controller:
    a. PID Tuning
    b. Pole Placement
    c. Self-tuning Regulators

    etc…



4. Deploy and Validate
    ○ Check to meet specifications

Effects of *increasing* a parameter independently[18]

Parameter	Rise time	Overshoot	Settling time	Steady-state error	Stability[14]
K_p	Decrease	Increase	Small change	Decrease	Degrade
K_i	Decrease	Increase	Increase	Eliminate	Degrade
K_d	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

Why not use simple PID rather than more complex Controllers?

PID Drawbacks

$$u(t) = K_p e(t) + K_i \int_0^t e(t')\,dt' + K_d \frac{de(t)}{dt}$$

- Handles **only a single input (e(t)) and a single output (u(t)) (SISO systems)**. E.g. angle error ⟶ steering angle input

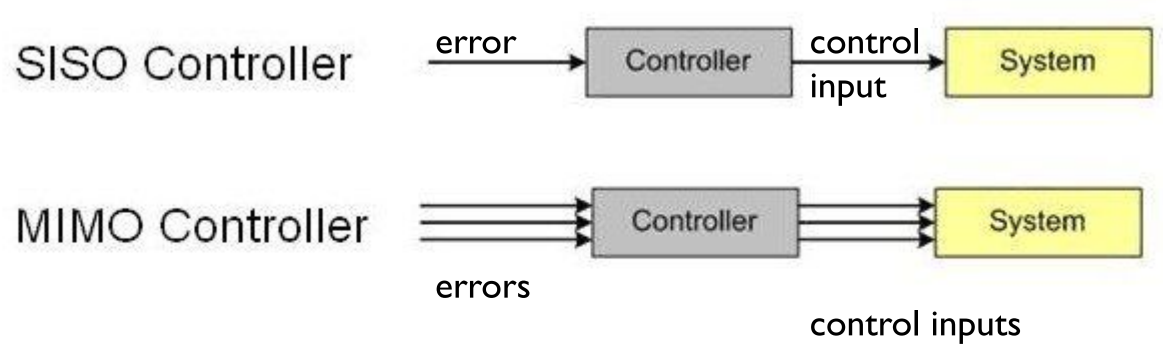


image: quora

PID Drawbacks

- A car takes multiple inputs (steering angle, acceleration).
- Independent PID controllers *may* give conflicting control commands, e.g., car may flip over.
- E.g. angle = steering angle = $\pi/3$, velocity = 70mph

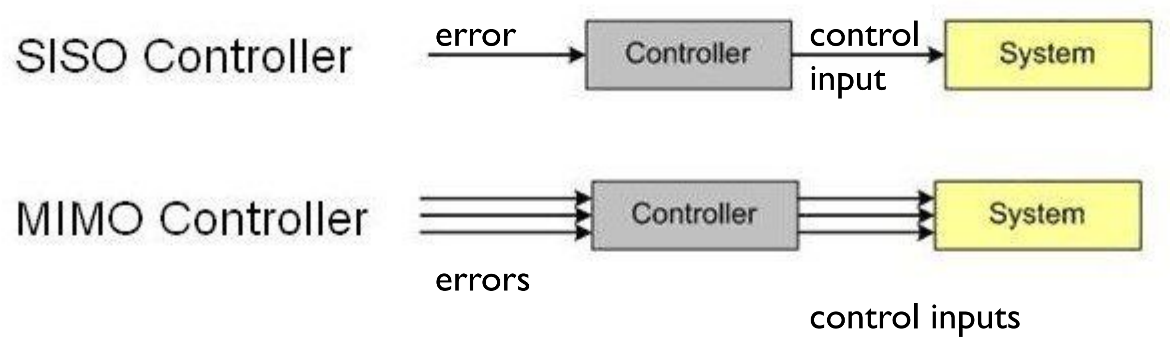
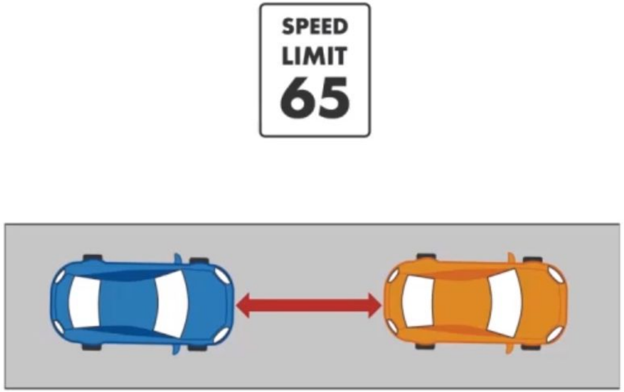


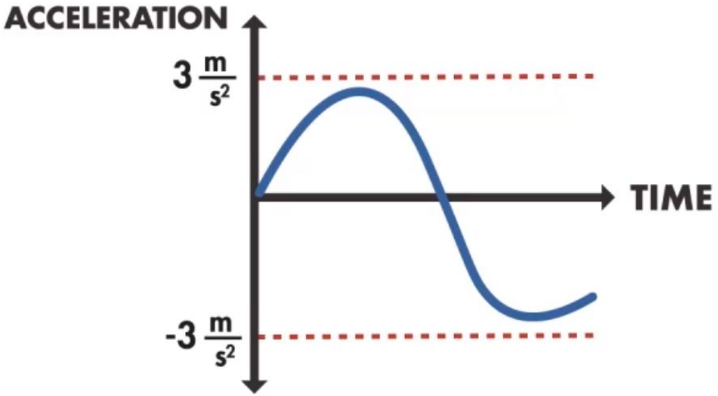
image: quora

PID Drawbacks

$$u(t) = K_\mathrm{p} e(t) + K_\mathrm{i} \int_0^t e(t')\,dt' + K_\mathrm{d} \frac{de(t)}{dt}$$

- **Cannot deal with constraints**. May generate impossible control inputs (steering angle = $\pi/2$) for the car to follow.





MIMO Control vs PID

- MIMO (Multi-Input Multi-Output) VS SISO with PID

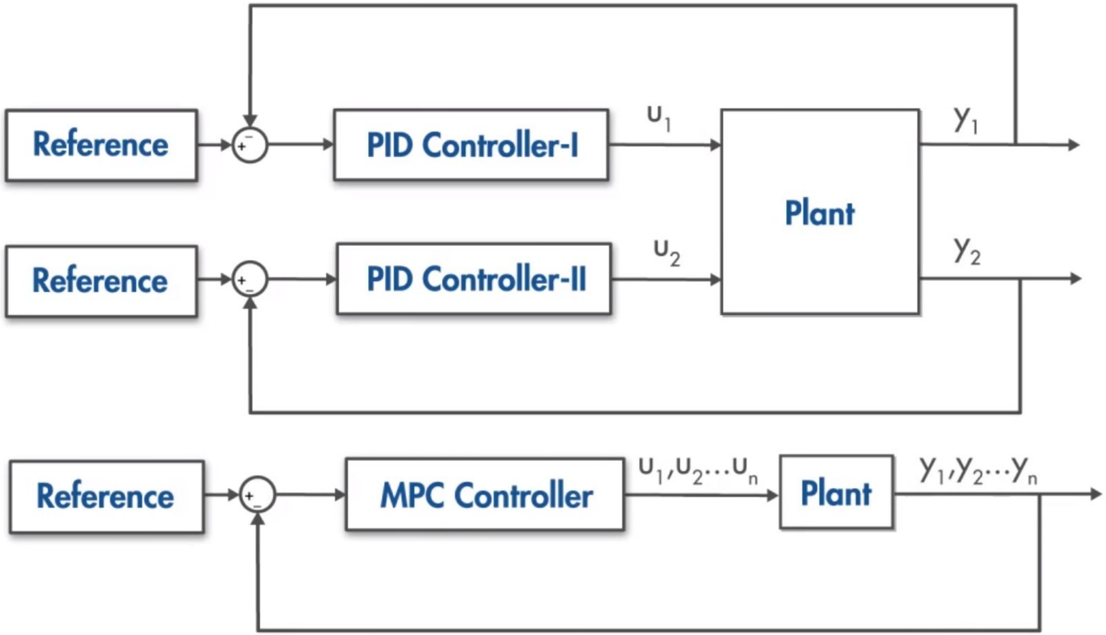
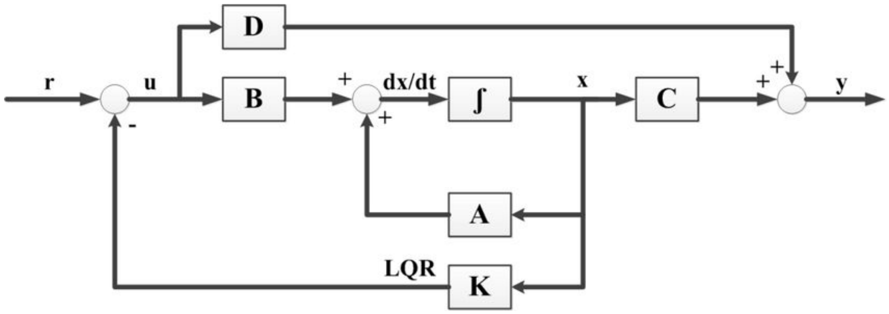


image: matlab

What does MIMO Control Look Like?

- Controller is now a multivariable (vector) function mapping from (at least) $R^N \rightarrow R^M$ where N is the number of system states, M is the number of control inputs
- Resulting control is dynamically feasible for full-state dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t)$$
$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t)$$



Introduction to Optimal Control

Penn Engineering

Optimal Control

- A branch of control concerned with deriving the "best" controllers given some definition of "best".
- More formally, optimal control is concerned with deriving controllers that minimize some defined cost function.
- Example: Optimal Unconstrained Linear Control - LQR:

$$\underset{u}{\operatorname{argmin}} \quad \int_0^\infty x^T Q x + u^T R u + x_\infty^T P x_\infty$$

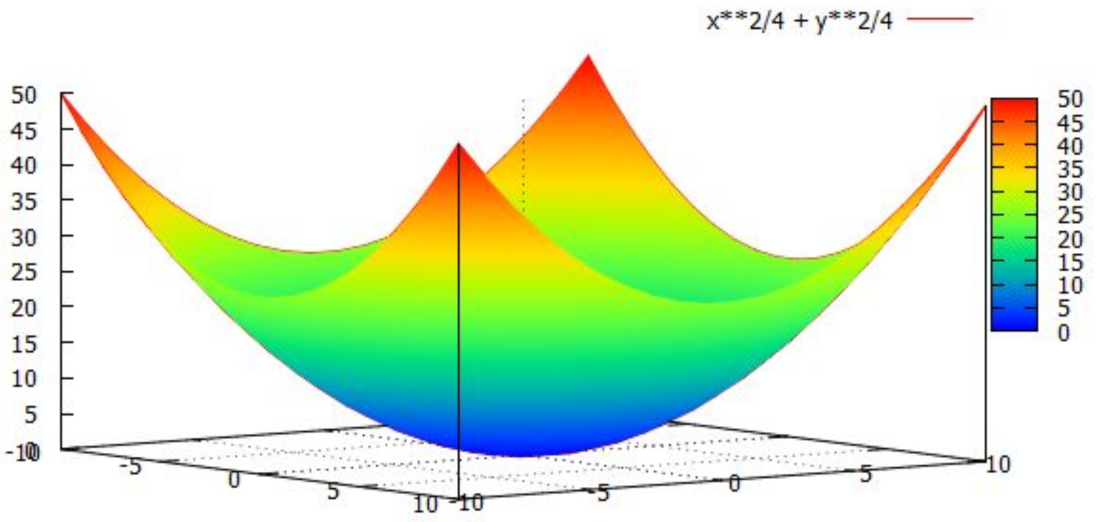
$$\text{given that} \quad \dot{x} = Ax + Bu$$

$$u(t) = R^{-1} B^T P$$

$$0 = A^T P + PA - PBR^{-1} B^T P + Q$$

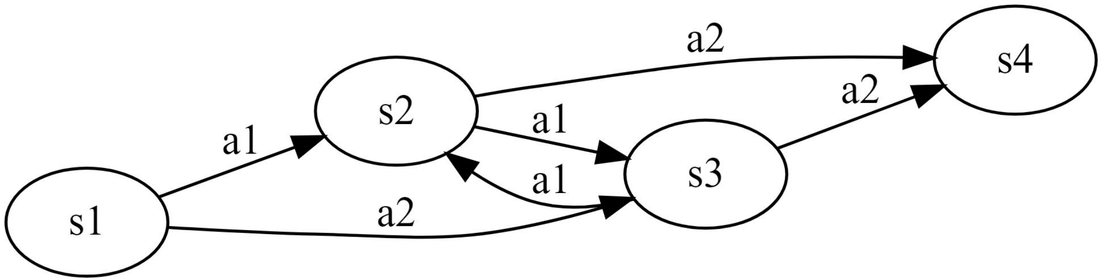
Formulating Optimal Control Problems

- How do we formulate and solve such a problem?
  - Start of by defining a cost function with "desirable" properties, denote by $\ell(\mathbf{x}, \mathbf{u})$:



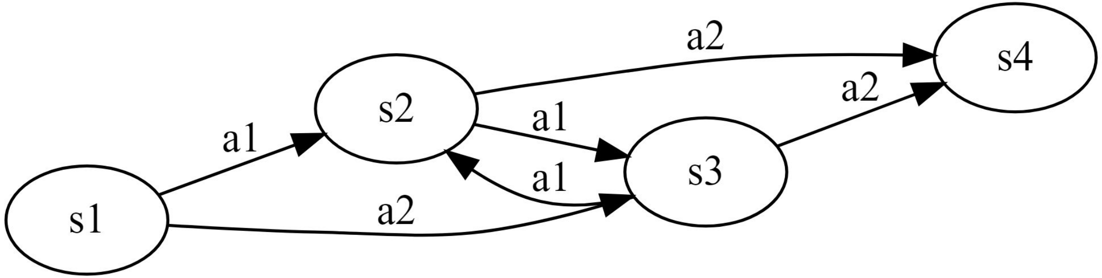
Solving Optimal Control Problems

- Minimize our "loss" function $\ell(\mathbf{x}, \mathbf{u})$
  - Find $\pi^*(\mathbf{x})$ that minimizes $\int_0^\infty \ell(\mathbf{x}, \mathbf{u})dt$ subject to our dynamics.
- Hard to solve, infinite-dimensional problem!
- Simplify:
  - Consider the following state-action graph



Discrete Optimal Control

- Problem is now to finding the shortest path on the graph

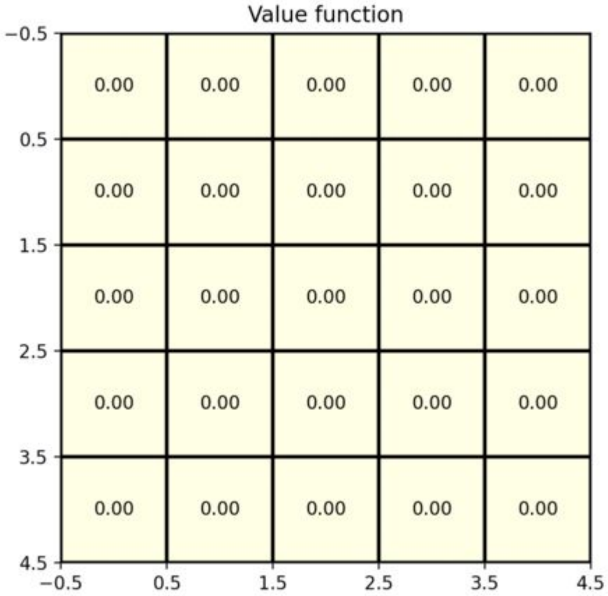


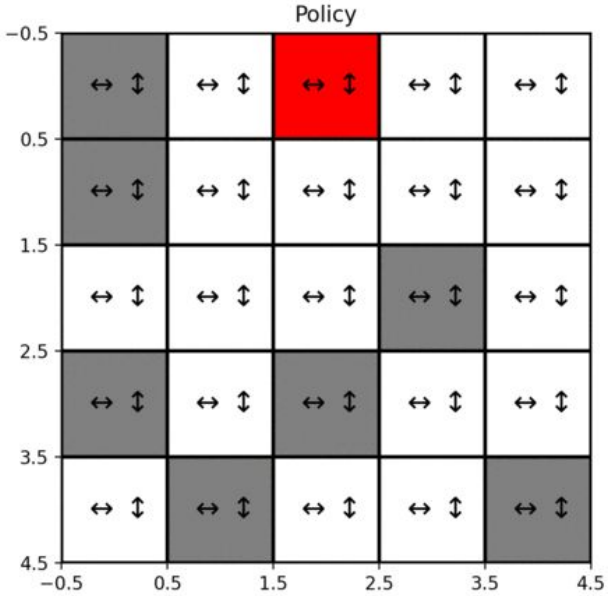
- Redefine infinite cost by the cost to the goal $J^*(\boldsymbol{s}_i)$
  - The optimal "cost-to-go"
  - Problem is now to minimize $J^*(\boldsymbol{s}_i)$:

$$\forall i \quad \hat{J}^*(\boldsymbol{s}_i) \Leftarrow \min_{a \in A} \left[\ell(\boldsymbol{s}_i, a) + \hat{J}^*\left(f(\boldsymbol{s}_i, a)\right) \right]$$

Discrete Optimal Control

- Iteratively updating the policy $\pi^*(\mathbf{x})$ to minimize $J^*(s_i)$ is called **Policy Iteration**:





Policy Iteration Algorithm

Policy Iteration (using iterative policy evaluation) for estimating $\pi \approx \pi_*$

1. Initialization
   $V(s) \in \mathbb{R}$ and $\pi(s) \in \mathcal{A}(s)$ arbitrarily for all $s \in \mathcal{S}$

2. Policy Evaluation
   Loop:
   $\quad \Delta \leftarrow 0$
   $\quad$ Loop for each $s \in \mathcal{S}$:
   $\quad\quad v \leftarrow V(s)$
   $\quad\quad V(s) \leftarrow \sum_{s',r} p(s',r|s,\pi(s))\big[r + \gamma V(s')\big]$
   $\quad\quad \Delta \leftarrow \max(\Delta, |v - V(s)|)$
   until $\Delta < \theta$ (a small positive number determining the accuracy of estimation)

3. Policy Improvement
   policy-stable $\leftarrow$ *true*
   For each $s \in \mathcal{S}$:
   $\quad$ *old-action* $\leftarrow \pi(s)$
   $\quad \pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s',r|s,a)\big[r + \gamma V(s')\big]$
   $\quad$ If *old-action* $\neq \pi(s)$, then *policy-stable* $\leftarrow$ *false*
   If *policy-stable*, then stop and return $V \approx v_*$ and $\pi \approx \pi_*$; else go to 2

Continuous Optimal Control Problems

- We can extend $J^*(s_i)$ to continuous space
  - Recursively minimize $\int_0^\infty \ell(\mathbf{x}, \mathbf{u})dt$ by minimizing $\ell(\mathbf{x}, \mathbf{u})$ plus the cost-to-go $J(x)$

$$\pi^*(\mathbf{x}) = \operatorname{argmin}_{\mathbf{u}} \left[\ell(\mathbf{x}, \mathbf{u}) + \frac{\partial J^*}{\partial \mathbf{x}} f(\mathbf{x}, \mathbf{u}) \right]$$

Optimal MIMO Control Example

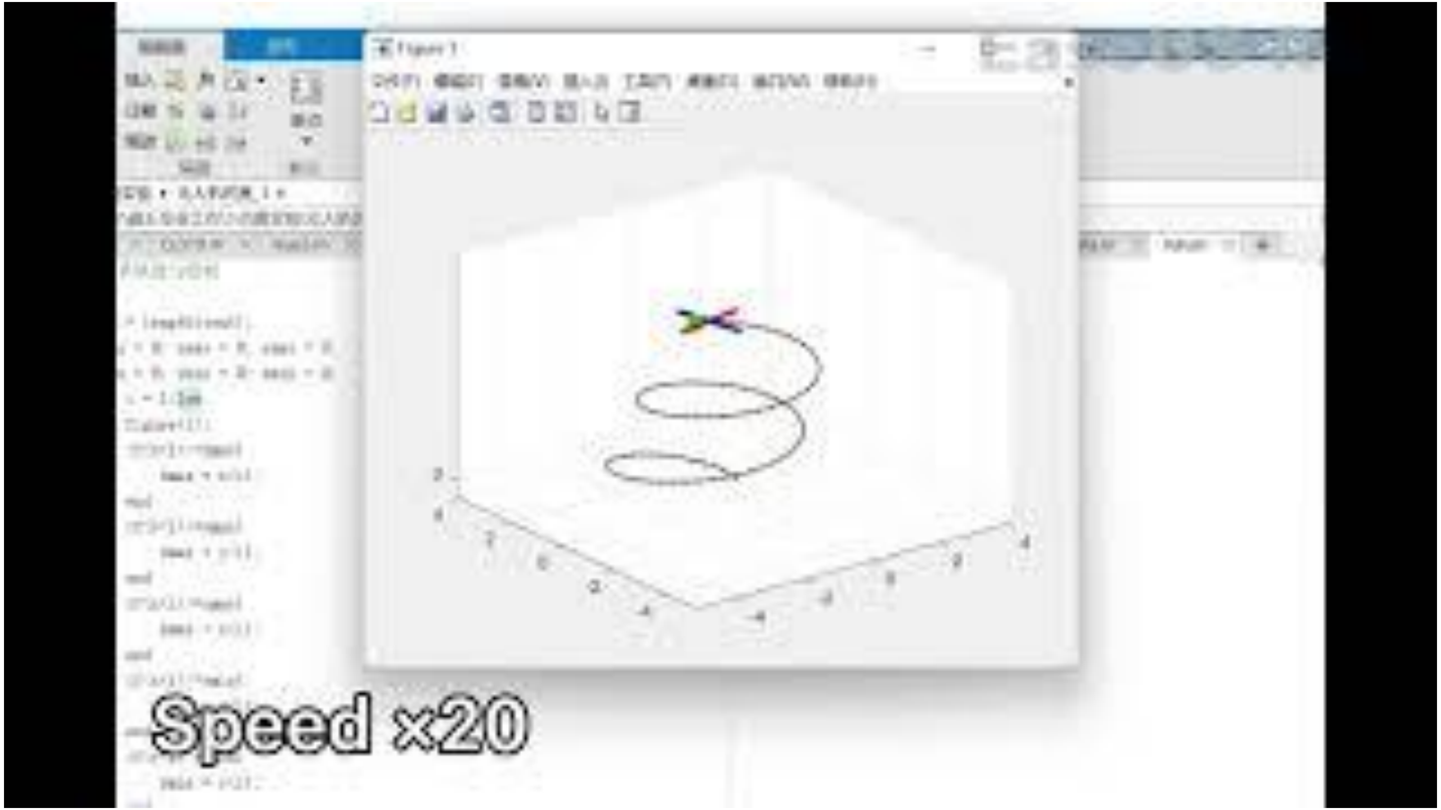
- Define $f(\mathbf{x}, \mathbf{u})$ as $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$
- Define $\ell(\mathbf{x}, \mathbf{u})$ as $\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u}$
  - To ensure "desired properties", enforce:

$$\mathbf{Q} = \mathbf{Q}^T \succeq 0, \mathbf{R} = \mathbf{R}^T \succ 0$$

- Derive $J^*(\mathbf{x})$ as $\mathbf{x}^T\mathbf{S}\mathbf{x}$ subject to $\mathbf{S} = \mathbf{S}^T \succeq 0$
  - Derived by solving HJB equation

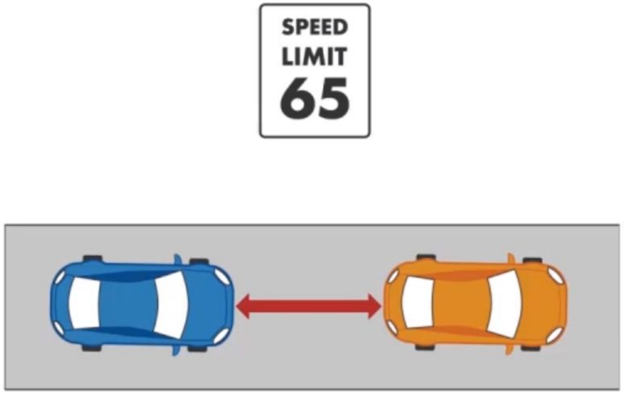
$$\forall \mathbf{x}, \quad 0 = \min_{\mathbf{u}} \left[\mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{u}^T\mathbf{R}\mathbf{u} + \frac{\partial J^*}{\partial \mathbf{x}}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}) \right]$$

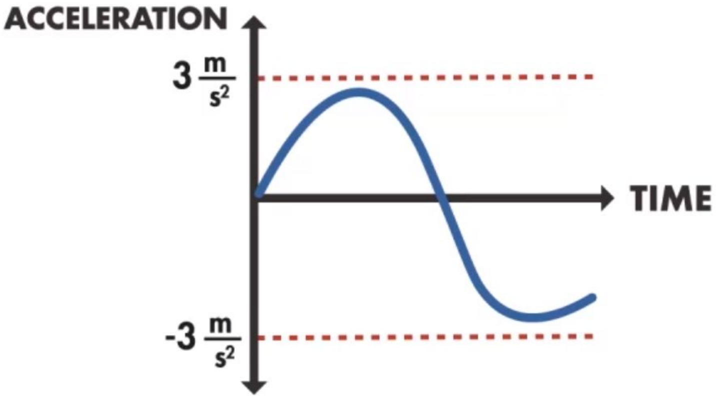
LQR Quadrotor Demo



Adding More Constraints?

- We often care about state or control constraints beyond just the dynamics
  - Example:





Adding More Constraints?

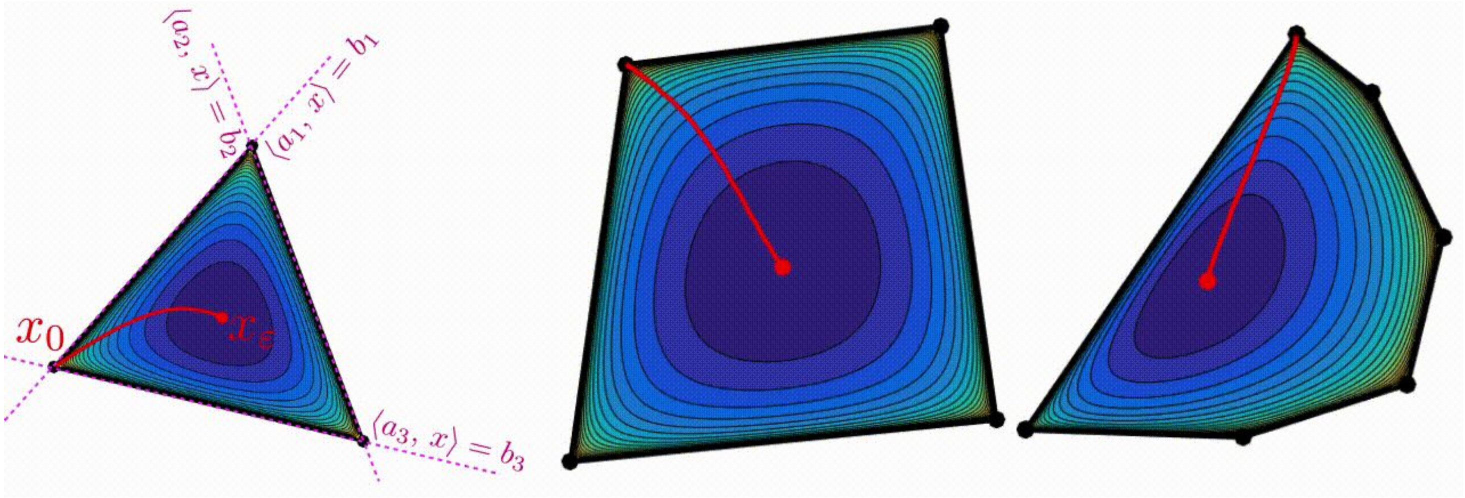
- Can we still use HJB to solve this optimal control problem?
  - Derived controller would no longer be linear (next lecture)

$$
\begin{aligned}
U_t^*(x(t)) := \underset{U_t}{\operatorname{argmin}} \; & \sum_{k=0}^{N-1} q(x_{t+k}, u_{t+k}) \\
\text{subj. to } \; & x_t = x(t) && \text{measurement} \\
& x_{t+k+1} = A x_{t+k} + B u_{t+k} && \text{system model} \\
& x_{t+k} \in \mathcal{X} && \text{state constraints} \\
& u_{t+k} \in \mathcal{U} && \text{input constraints} \\
& U_t = \{u_0, u_1, \ldots, u_{N-1}\} && \text{optimization variables}
\end{aligned}
$$

Penn Engineering

Numerical Optimization

- Alternatively, we can solve the problem numerically
  - **IF** problem can be formulated in **solvable form** (i.e compatible with solver)



Introduction to Convex Optimization

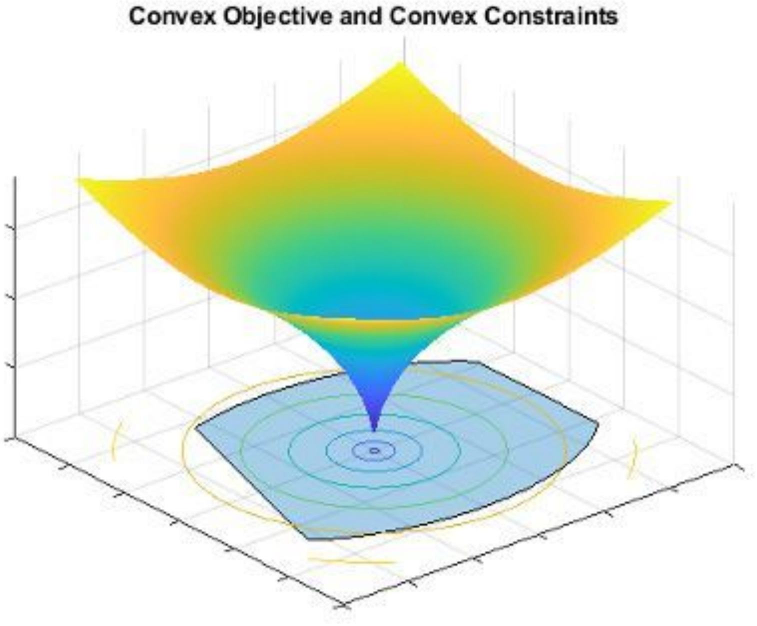
Penn Engineering

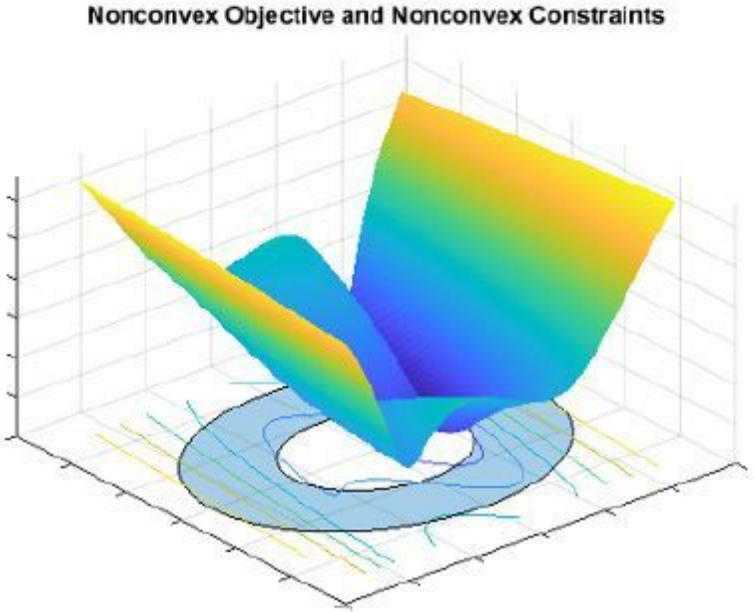
What is Optimization?

- “Maximizing or minimizing some function relative to some set, often representing a range of choices available in a certain situation”
- Example Optimization Problem:
  Maximize Value of Bag
  Subject to Weight <= 15

Why Convex Optimization?

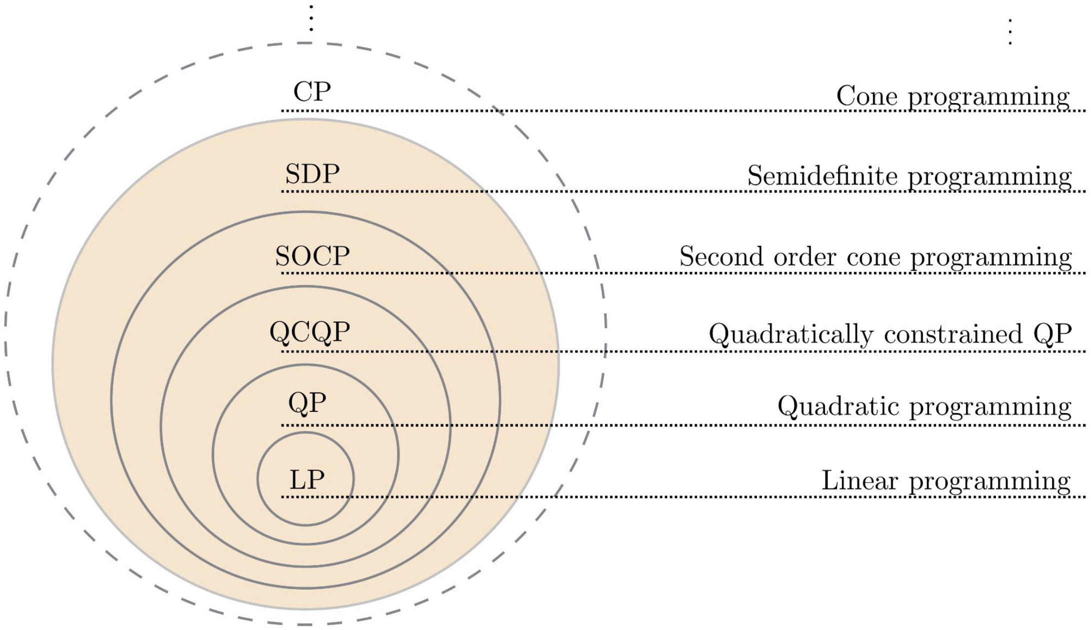
- Guarantees local optimality is global optimality





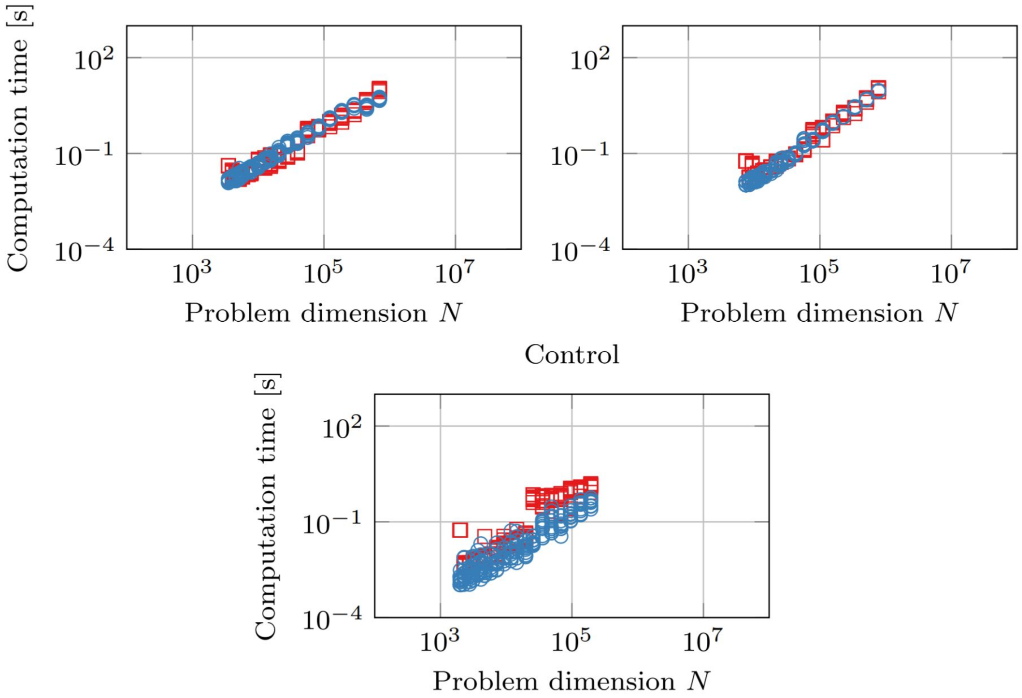
Why Convex Optimization?

- Comprises a large family of problems, with matching solvers



Why Convex Optimization?

- Low computation time allows solvers to be used in real-time

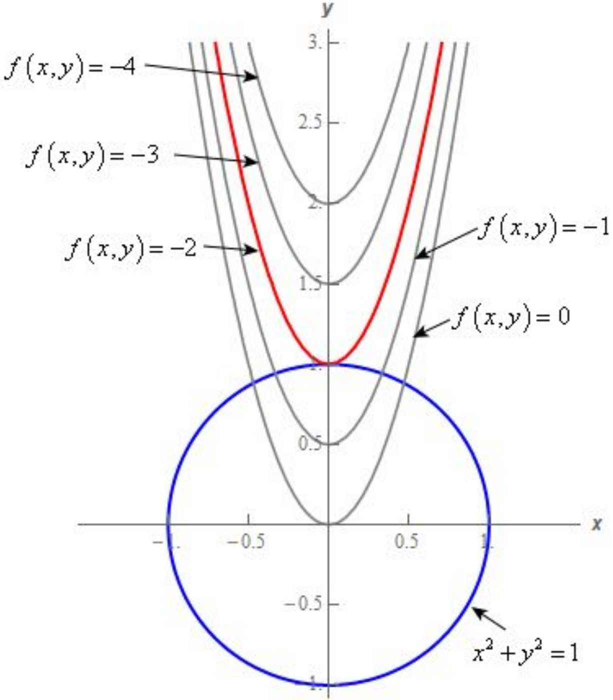


Convex Optimization

- A special class of mathematical optimization problems where the objective function and constraints are convex functions
- Example:

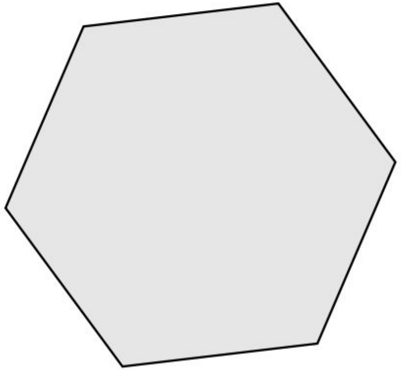
$$\min_{x,y} \quad f(x,y) = 8x^2 - 2y$$

$$\text{subject to} \quad x^2 + y^2 = 1$$

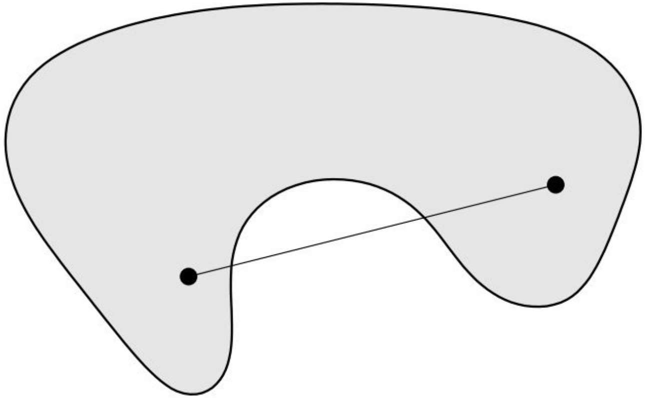


Convexity

- A convex set is one where all points along a line inside the set are also elements of the set



Convex

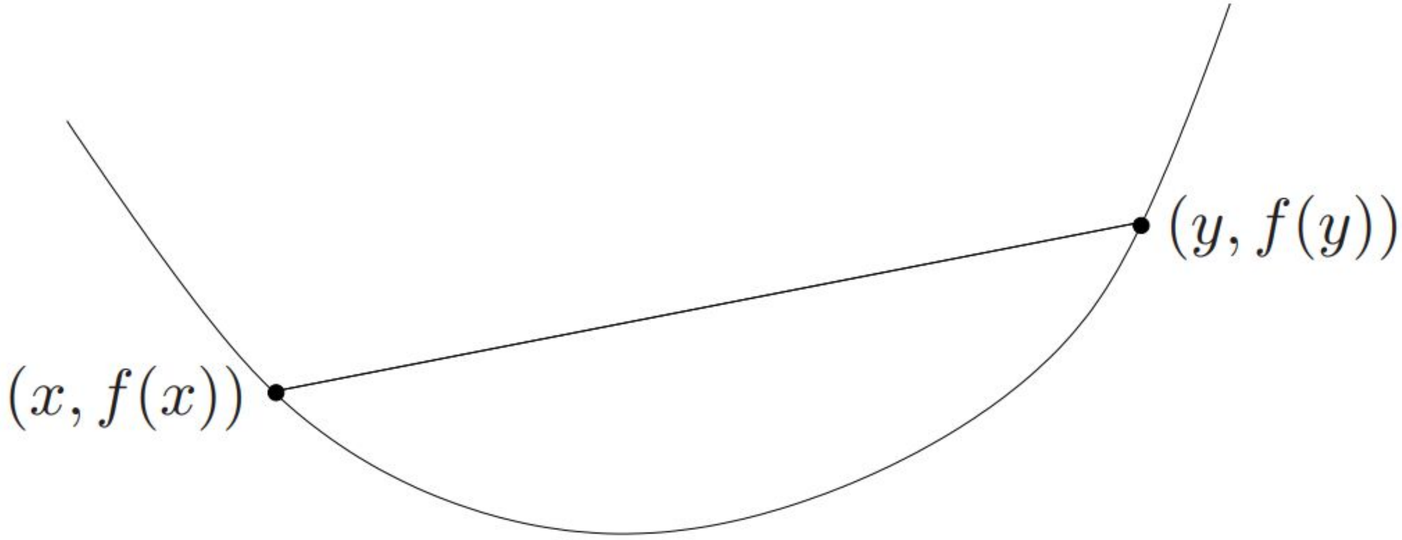


Non-Convex

Non-Convex

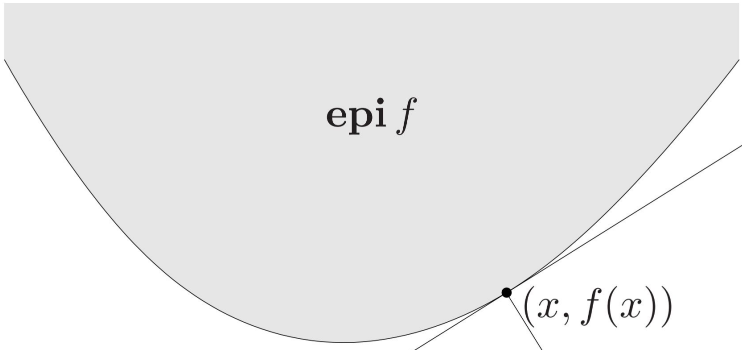
Convex Functions

- A convex function is one where all points along a line connecting two points of the function lie above the function

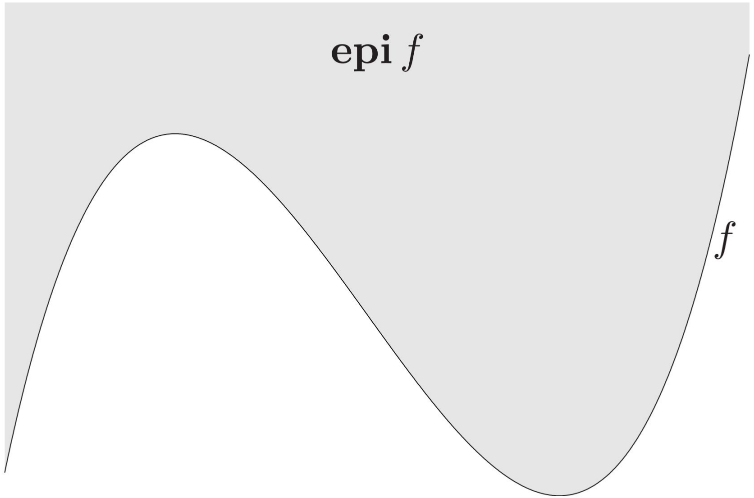


Relation to Convex Sets

- The epigraph of a convex function forms a convex set, where the epigraph is defined by $\textbf{epi}\, f = \{(x,t) \mid x \in \textbf{dom}\, f,\ f(x) \le t\}$



Convex



Non-Convex

Solving Convex Optimization

- The most popular unconstrained convex optimization solver is gradient descent

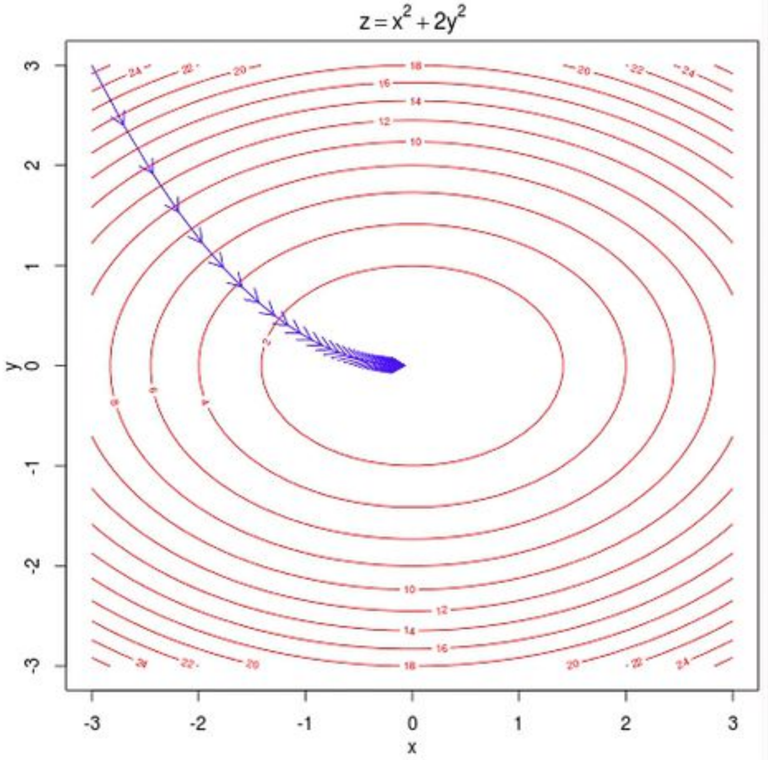
Algorithm 9.3 *Gradient descent method.*

given a starting point $x \in \textbf{dom}\, f$.

repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.



Penn Engineering

Constrained Solvers

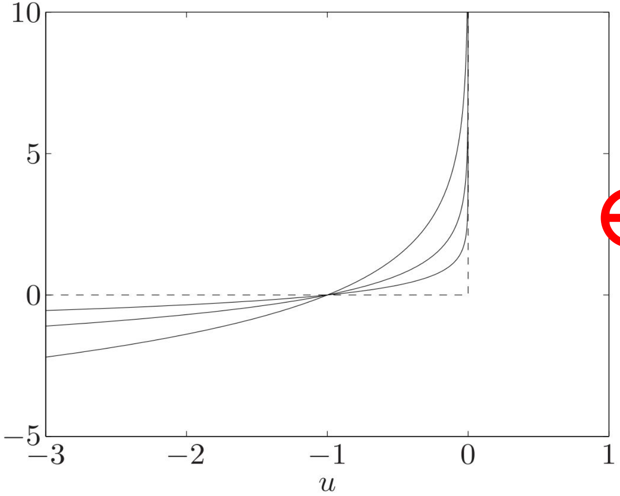
- The most popular constraint solvers use Interior-point methods
  - Idea: Structure problem such that an interior central path drives the iterative updates to the optimal point

$$\begin{array}{ll} \text{minimize} & f_0(x) + \sum_{i=1}^{m} I_-(f_i(x)) \\ \text{subject to} & Ax = b, \end{array}$$

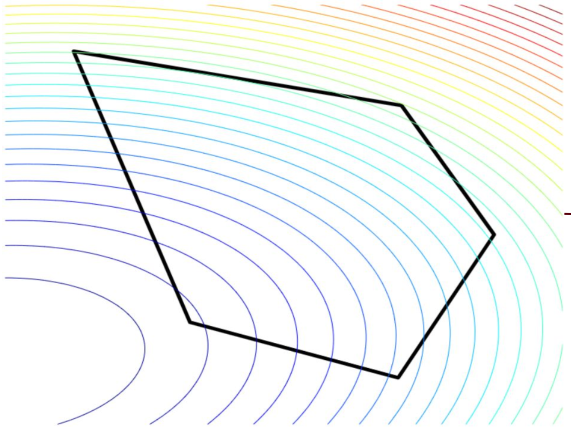
$$I_-(u) = \begin{cases} 0 & u \le 0 \\ \infty & u > 0. \end{cases}$$

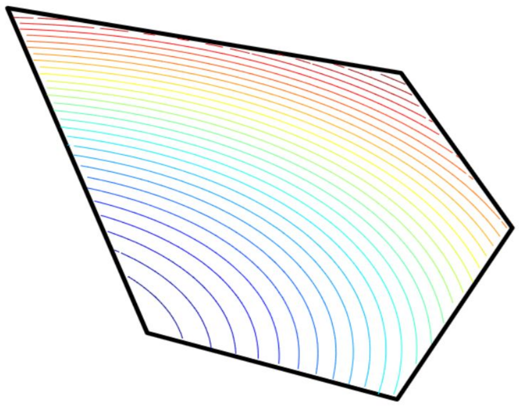
Constrained Solvers

- In practice, log-barrier functions are used
  - differentiable and numerically "friendly"



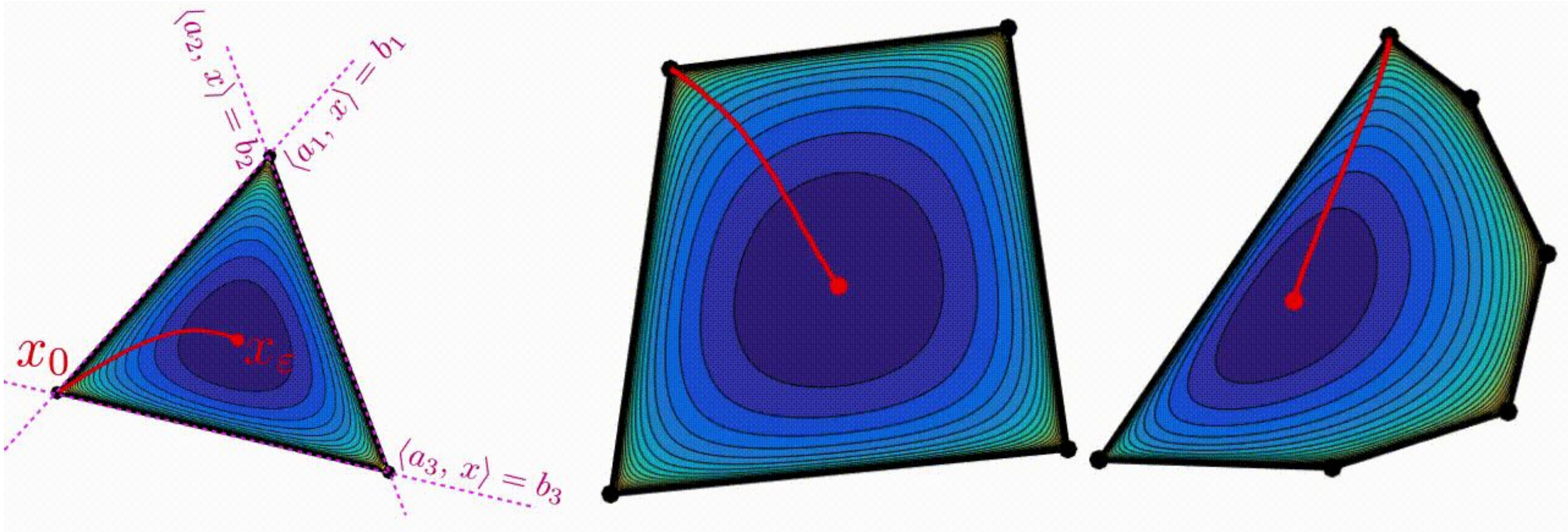
$\widehat{I}_-(u) = -(1/t)\log(-u)$, for $t = 0.5,\ 1,\ 2.$





Interior-Point Method

- Resulting optimization with central path



Take-aways

Summary

- Control design as a field and typical control workflow
- Comparison of PID control with other controllers
  - Most notably MIMO control
- Optimal Control as a field
  - HJB equation for deriving LQR controller
- Convex Optimization as a powerful mathematical tool to solving certain problems
  - Different solvers and handling constraints

Next Lecture

- Combine convex optimization with optimal control to derive a constrained optimal controller
  - How to formulate optimal control problem as convex problem?
- Practical implementation of controller
  - Efficient implementation through mathematical formulation