清華大學 交叉信息研究院
Tsinghua University Institute for Interdisciplinary Information Sciences

ISR Lab
Intelligent Systems and Robotics Lab

Advanced Topics for Robotics
智能机器人前沿探究

Lecture 4: Optimal Control and Planning

Jianyu Chen (陈建宇)

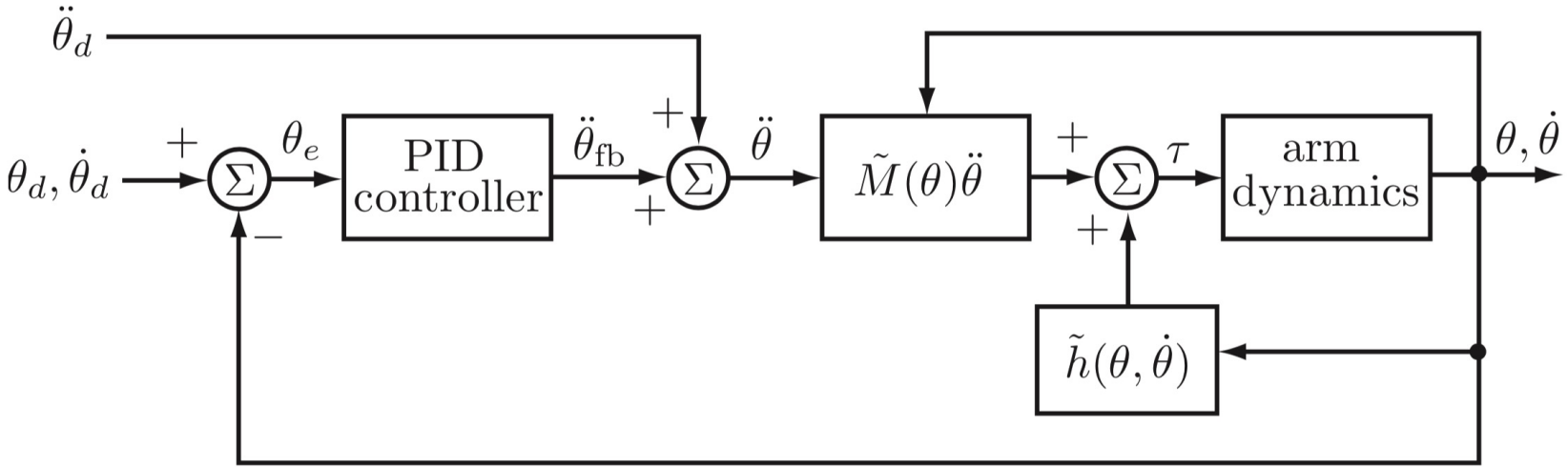
Institute for Interdisciplinary Information Sciences

Tsinghua University

Contents

About basic feedback control

- The basic feedback control we introduced is only able to track a desired point/trajectory
- This requires us to define a useful desired trajectory
- Let's first introduce a basic heuristic method to plan the trajectory



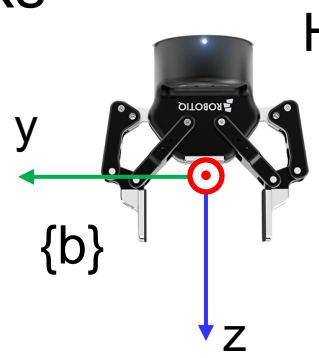
Pick and place

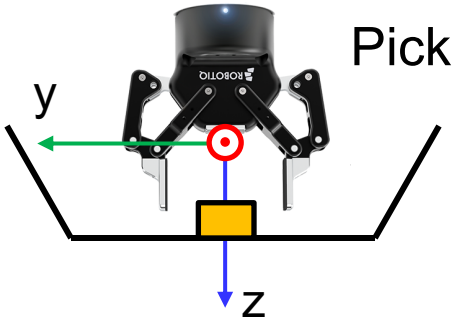
- Automatic **pick and place** is a core application for industrial robots

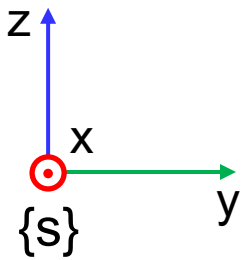
Basic pick and place

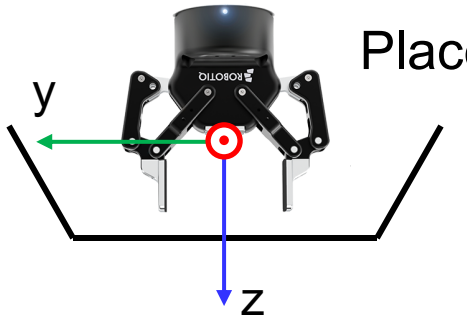
- Define **goal poses** of the robot's end-effector to perform the pick and place tasks

How to determine the SE(3) matrices? (xyz-rgb)







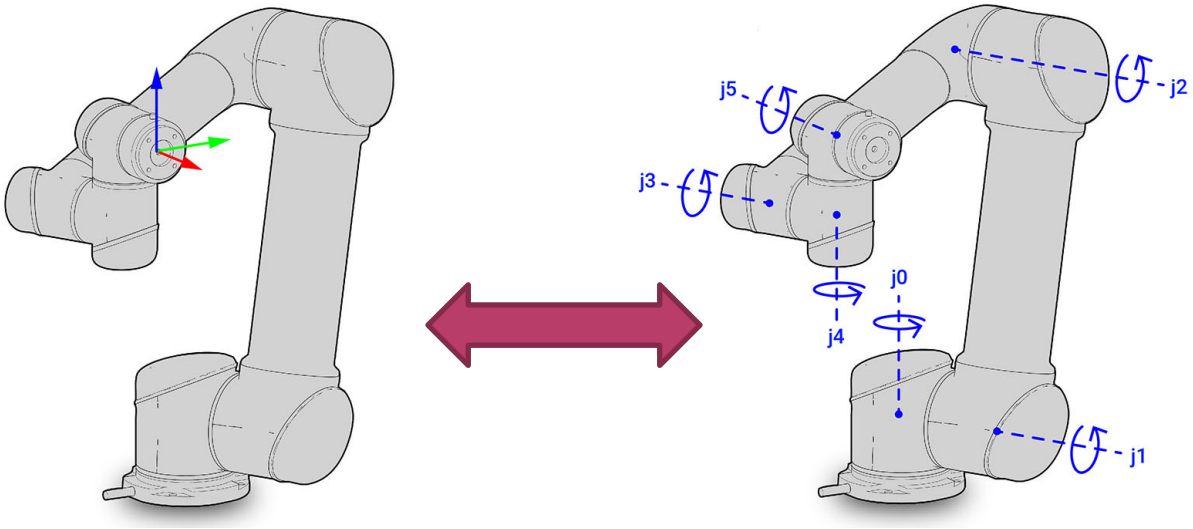


- Now the problem becomes: how to plan robot joint motions to move the end-effector to the goal poses?

Kinematic planning

- How to plan the **joint space** motion to generate desired end-effector motions?
- Forward kinematics?

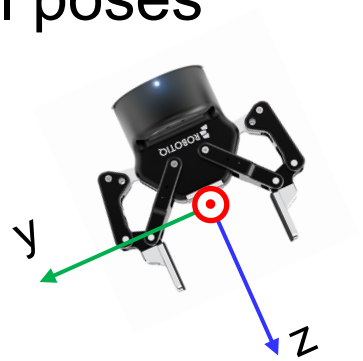
$$T = f(\theta)$$



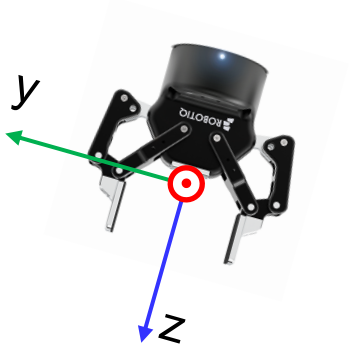
Task space C-space

Plan with inverse kinematics

- Use **inverse kinematics (IK)** to compute the corresponding **joint angles** of goal poses
- Plan a **joint space trajectory** to connect the start and goal poses



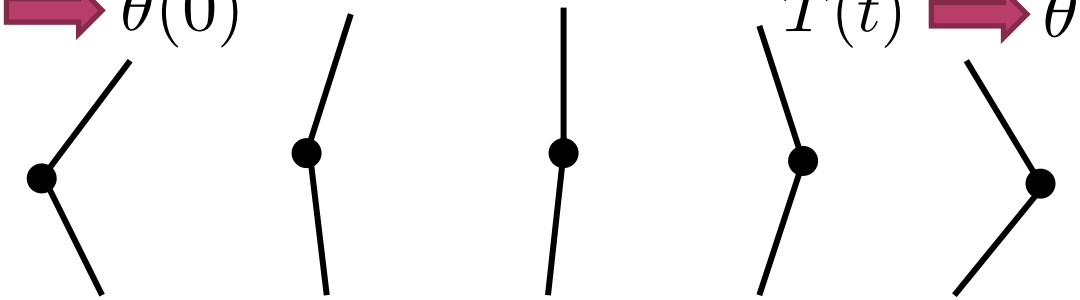
Start pose



Goal pose

$T(0)$ IK $\theta(0)$

$T(t)$ IK $\theta(t)$



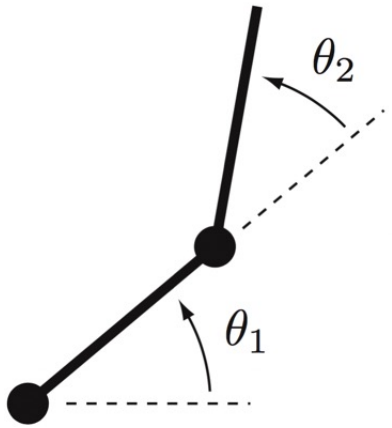
Joint space trajectory

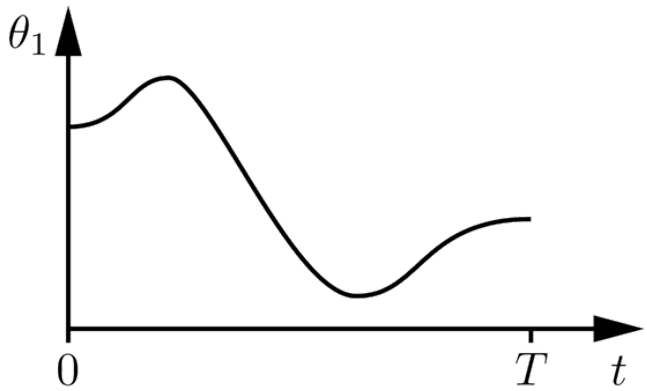
Trajectory definitions

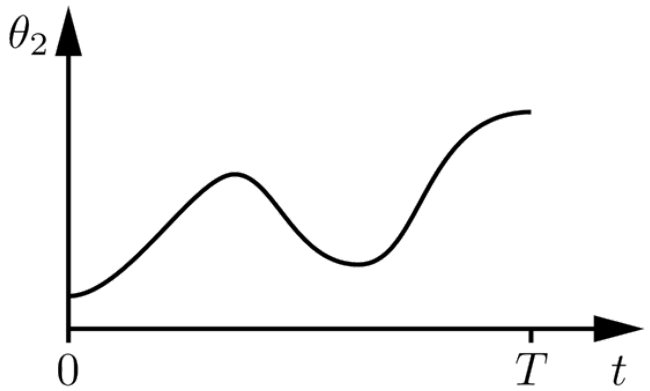
Trajectory

- A specification of the configuration as a function of time.

$$\theta(t),\ t \in [0, T]$$





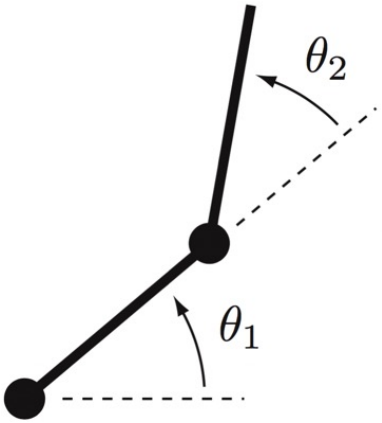


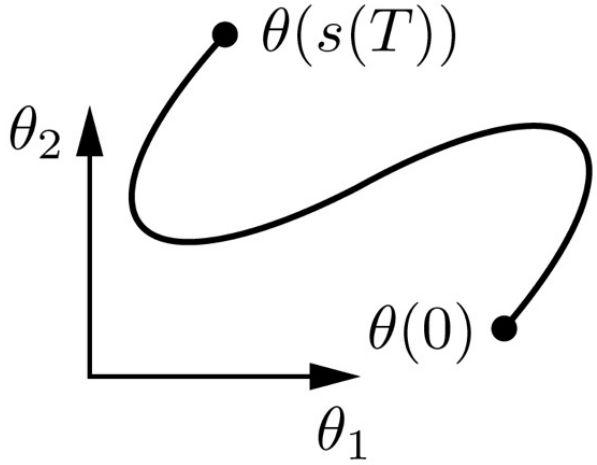
Trajectory definitions

- **Path**
  - A specification of the configuration as a function of a path parameter.

$$\theta(s),\ s \in [0, 1]$$

- **Time scaling**
  - A mapping s(t): [0, T] → [0, 1], from time to the path parameter.





Trajectory definitions

- Motion as a function of θ(s) and s(t):

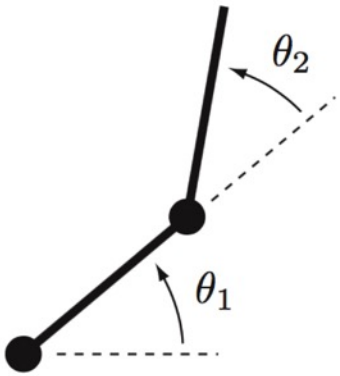
$$\dot{\theta} = \frac{d\theta}{ds}\dot{s},$$
$$\ddot{\theta} = \frac{d\theta}{ds}\ddot{s} + \frac{d^2\theta}{ds^2}\dot{s}^2$$

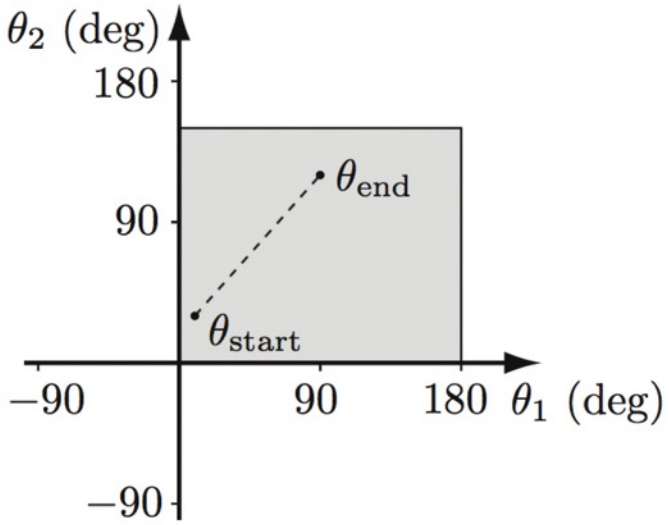
- Both $\theta(s)$ and $s(t)$ must be twice-differentiable.

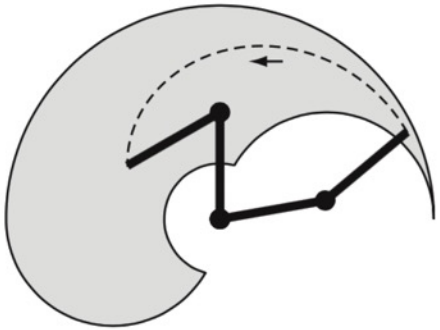
Path interpolation

- Straight line in joint space

$$\theta(s) = \theta_{\text{start}} + s(\theta_{\text{end}} - \theta_{\text{start}})$$







Third-order Polynomial Time scaling

- We can define the time scaling as a cubic polynomial of time

$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

- It should satisfy the initial and terminal constraints:

$$s(0) = \dot{s}(0) = 0 \qquad s(T) = 1 \qquad \dot{s}(T) = 0$$

- The solution is:

$$a_0 = 0, \quad a_1 = 0, \quad a_2 = \frac{3}{T^2}, \quad a_3 = -\frac{2}{T^3}$$

$$s = a_2 t^2 + a_3 t^3$$

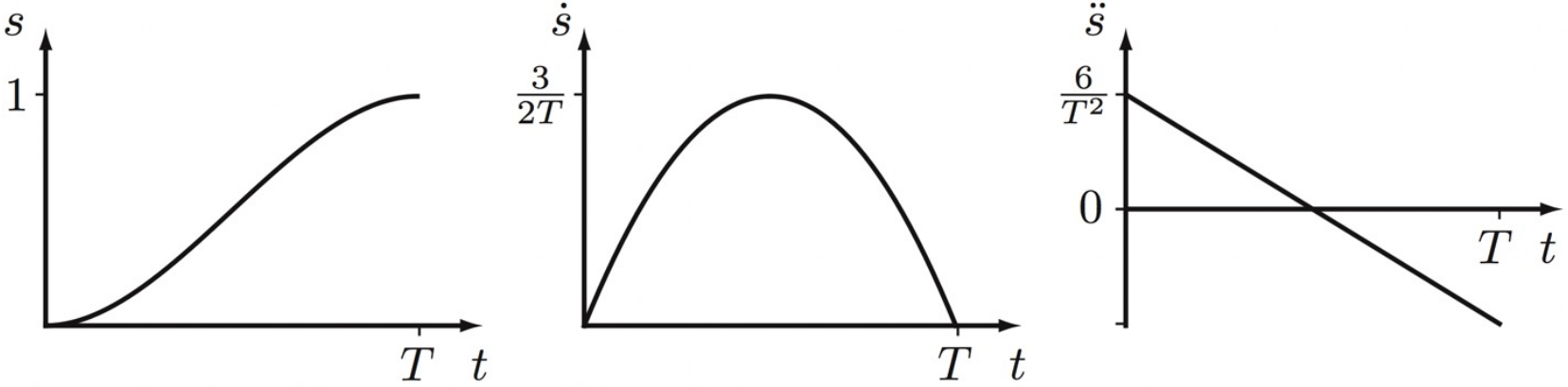
Third-order Polynomial Time scaling

- Applying this time scaling to the straight-line path, we have

$$\theta(t) = \theta_{\text{start}} + \left(\frac{3t^2}{T^2} - \frac{2t^3}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}})$$

$$\dot{\theta}(t) = \left(\frac{6t}{T^2} - \frac{6t^2}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}}),$$

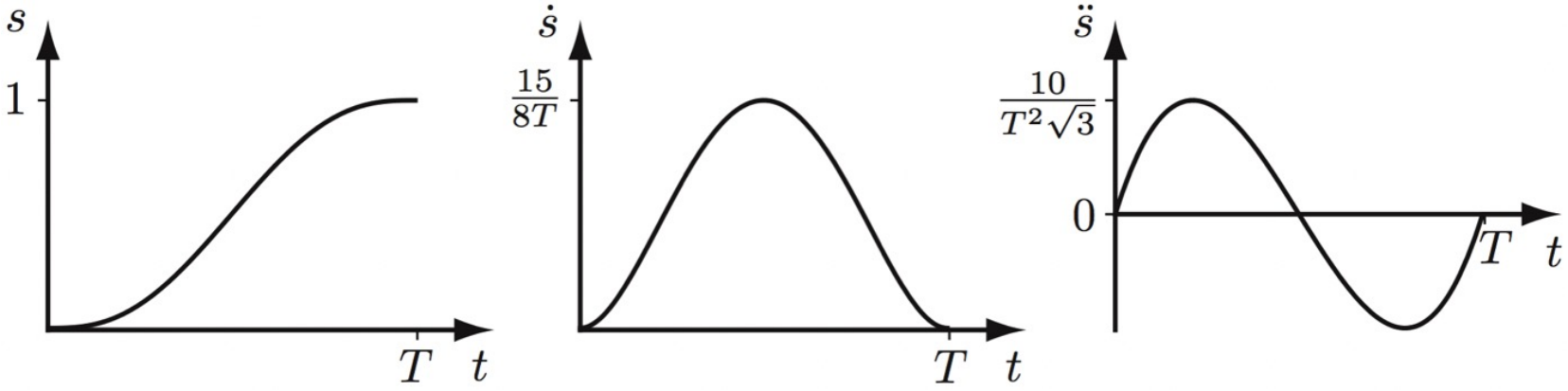
$$\ddot{\theta}(t) = \left(\frac{6}{T^2} - \frac{12t}{T^3}\right)(\theta_{\text{end}} - \theta_{\text{start}}).$$



Fifth-order Polynomial Time scaling

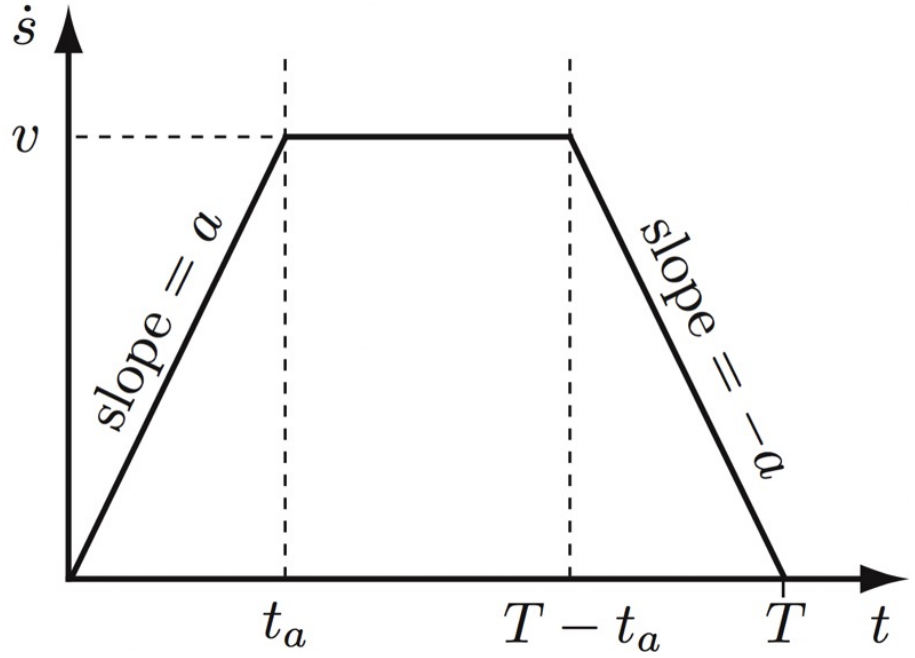
$$s(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5$$

$$s(0) = 0 \quad \dot{s}(0) = 0 \quad \ddot{s}(0) = 0$$
$$s(T) = 1 \quad \dot{s}(T) = 0 \quad \ddot{s}(T) = 0$$



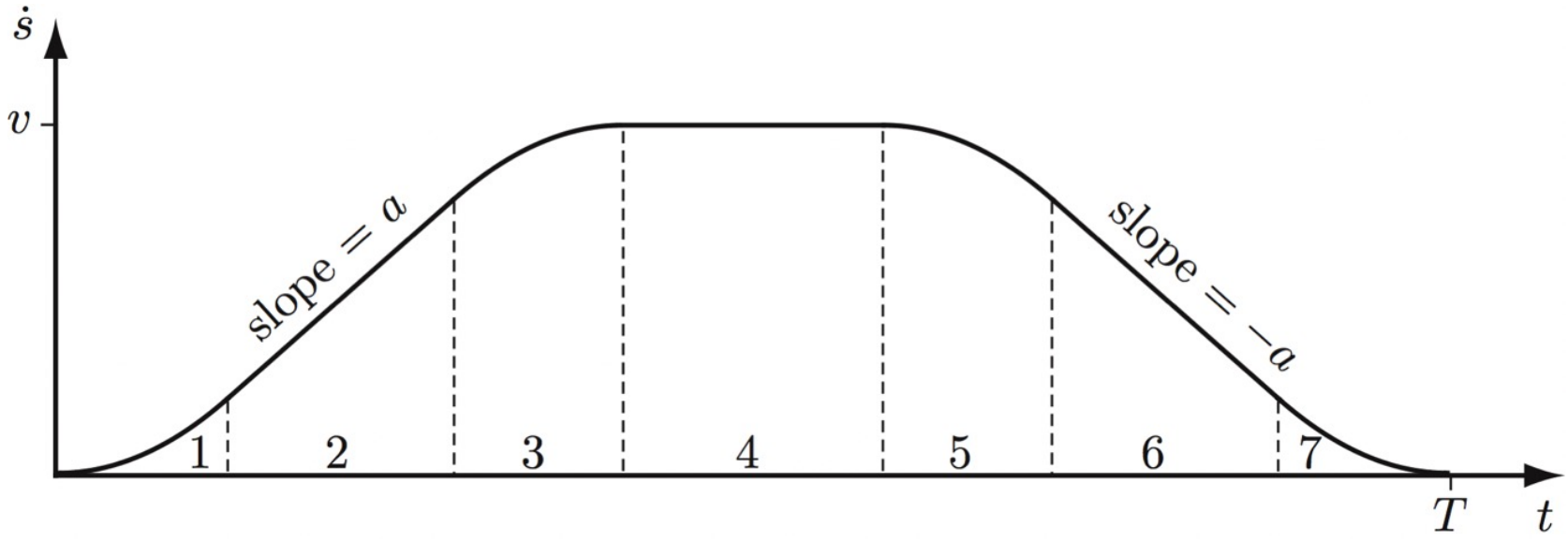
Trapezoidal time scaling

- Constant accelerations



S-curve time scaling

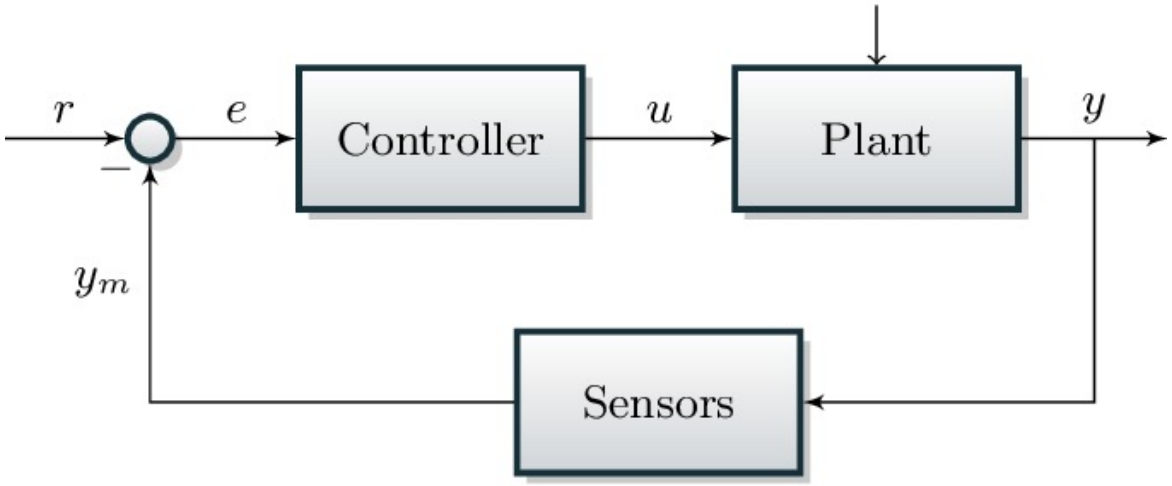
- Constant accelerations in 2,6
- Constant jerk in 1,3,5,7



Contents

Robot feedback control

- The robot control methods introduced in the last lecture are **global feedback control**

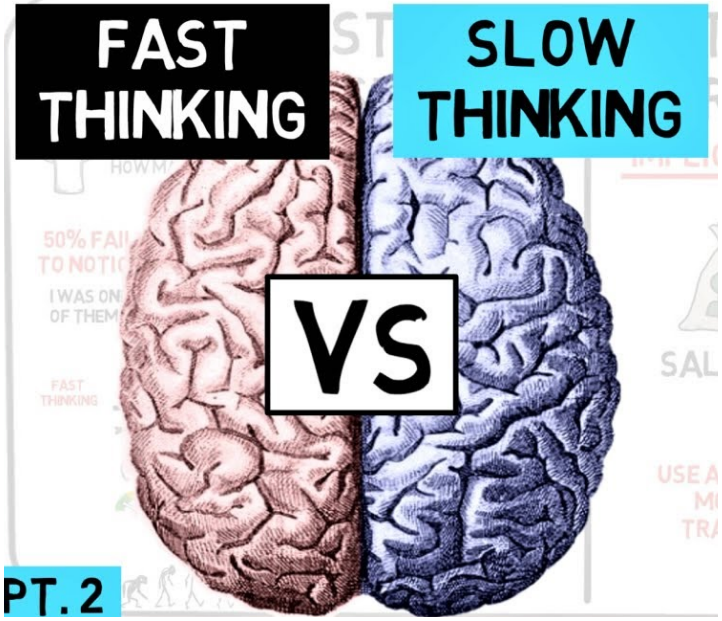


$$u = g(e)$$

Global Feedback vs Local Optimization

Global Feedback

Local Optimization



Local Optimization

- We can consider the control problem as solving the **local optimization** problem
- Due to inability to optimize infinite horizon, we use **finite horizon** to approximate

Final cost to approximate infinite horizon

$$u(\cdot), x(\cdot) = \underset{u(\cdot), x(\cdot)}{\operatorname{argmin}} \; l_T(x(T)) + \int_{t=0}^{T} l(x(t), u(t))$$

$$\text{s.t.} \quad \dot{x}(t) = f(x(t), u(t))$$

$$x(t) = x_0$$

Forward dynamics

Forward dynamics

- Original forward dynamics

$$\ddot{\theta} = ForwardDynamics(\theta, \dot{\theta}, \tau, \mathcal{F}_{\text{tip}})$$

- Let $q_1 = \theta,\ q_2 = \dot{\theta}$, we have

$$\begin{aligned} \dot{q}_1 &= q_2, \\ \dot{q}_2 &= ForwardDynamics(q_1, q_2, \tau, \mathcal{F}_{\text{tip}}). \end{aligned}$$

- Let $x = [q, \dot{q}]^T$ and $u = \tau$, we have

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

System Discretization

- Furthermore, we need the **discrete-time** formulation
- General form for **system discretization**

$$x[k+1] = x[k] + \int_{t[k]}^{t[k+1]} f(x[t], u[t])dt, \quad x(t[k]) = x[k]$$

- **Euler integration**

$$\begin{aligned} x[k+1] &= f_d(x[k], u[k]) \\ &\approx x[k] + f_c(x[k], u[k])dt \end{aligned}$$

- Discretization for linear systems

$$\begin{aligned} x[k+1] &= x[k] + \int_{t_n}^{t_n+h} [A(t) + Bu]dt \\ &= e^{Ah}x[k] + A^{-1}(e^{Ah} - I)Bu[k] \end{aligned}$$

Discrete-time local optimization

- Local optimization problem formulation

The planned trajectory

$$u[0:T], x[0:T] = \underset{u[0:T], x[0:T]}{\operatorname{argmin}} \sum_{k=0}^{T} l(x[k], u[k])$$

$$\text{s.t.} \quad x[k+1] = f(x[k], u[k])$$

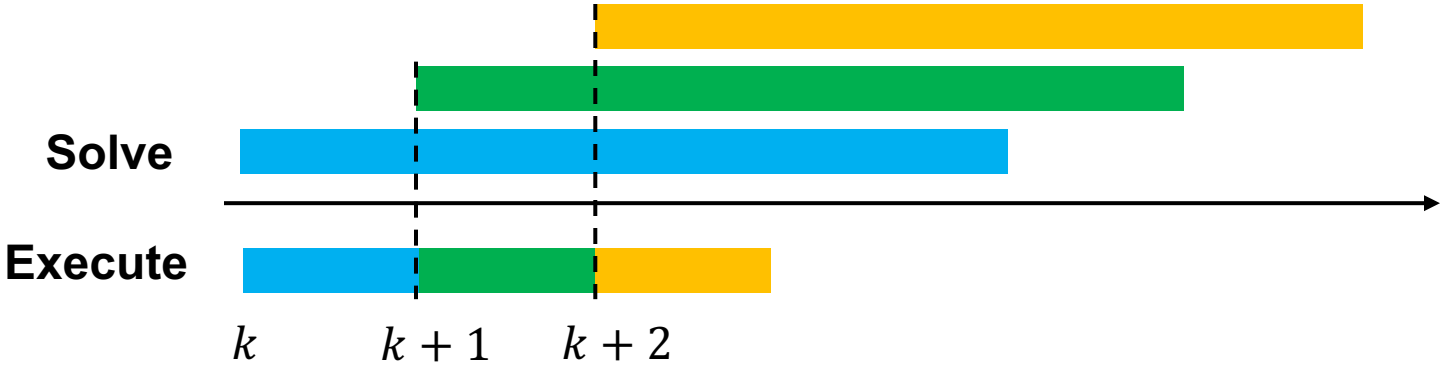
$$x[0] = x_0$$

Initial condition

+ **Additional constraints**
(Torque limits, safety)

Model Predictive Control

- After we got the planned trajectory $u[0:T], x[0:T]$, we can just take the action of **the first step** $u[0]$
- Then at the next step, solve a **new** local optimization problem initialized from **the next step's state**

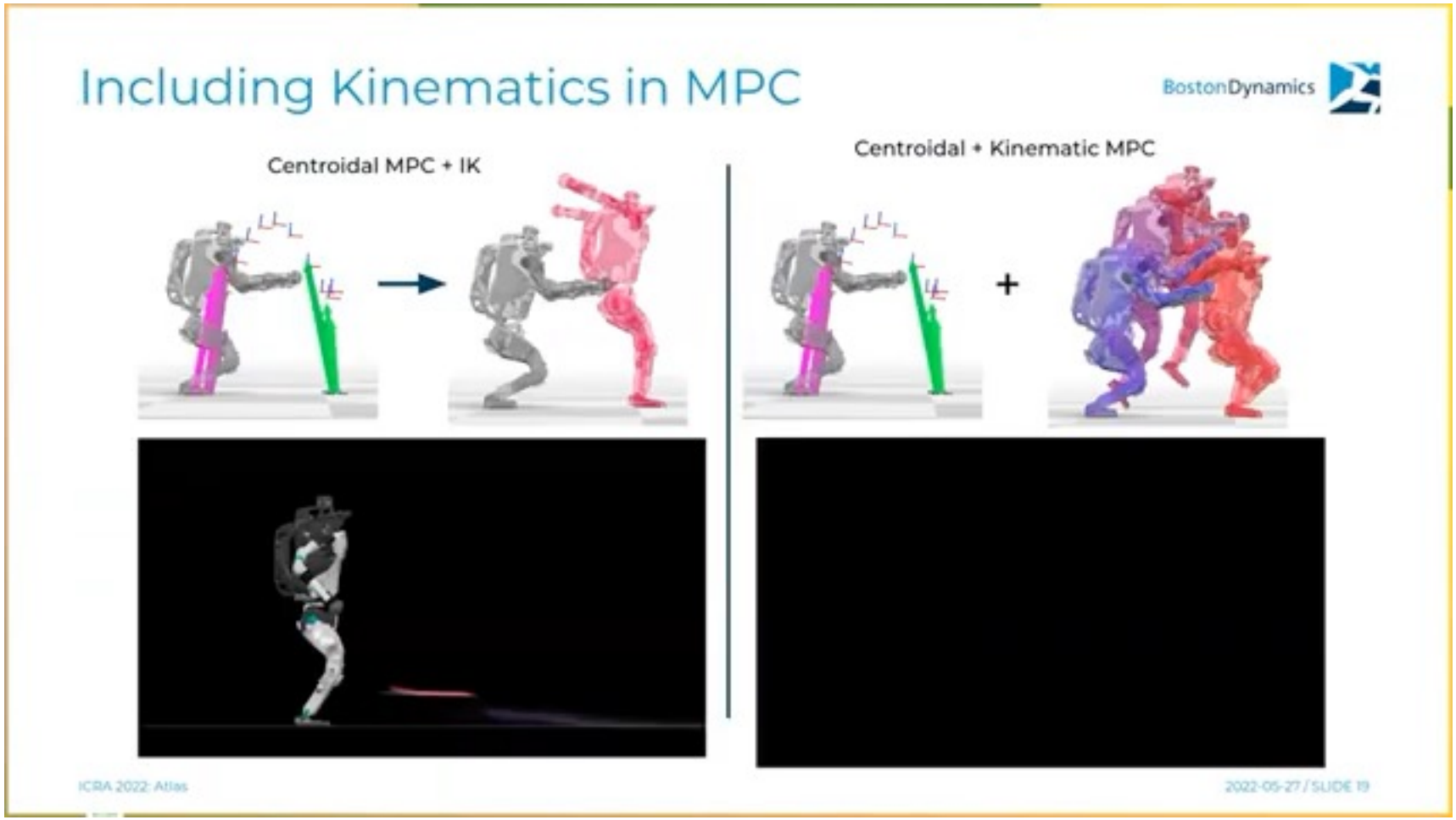


- This is actually a **feedback** of the initial conditions

$$u[0] = g(x[0])$$

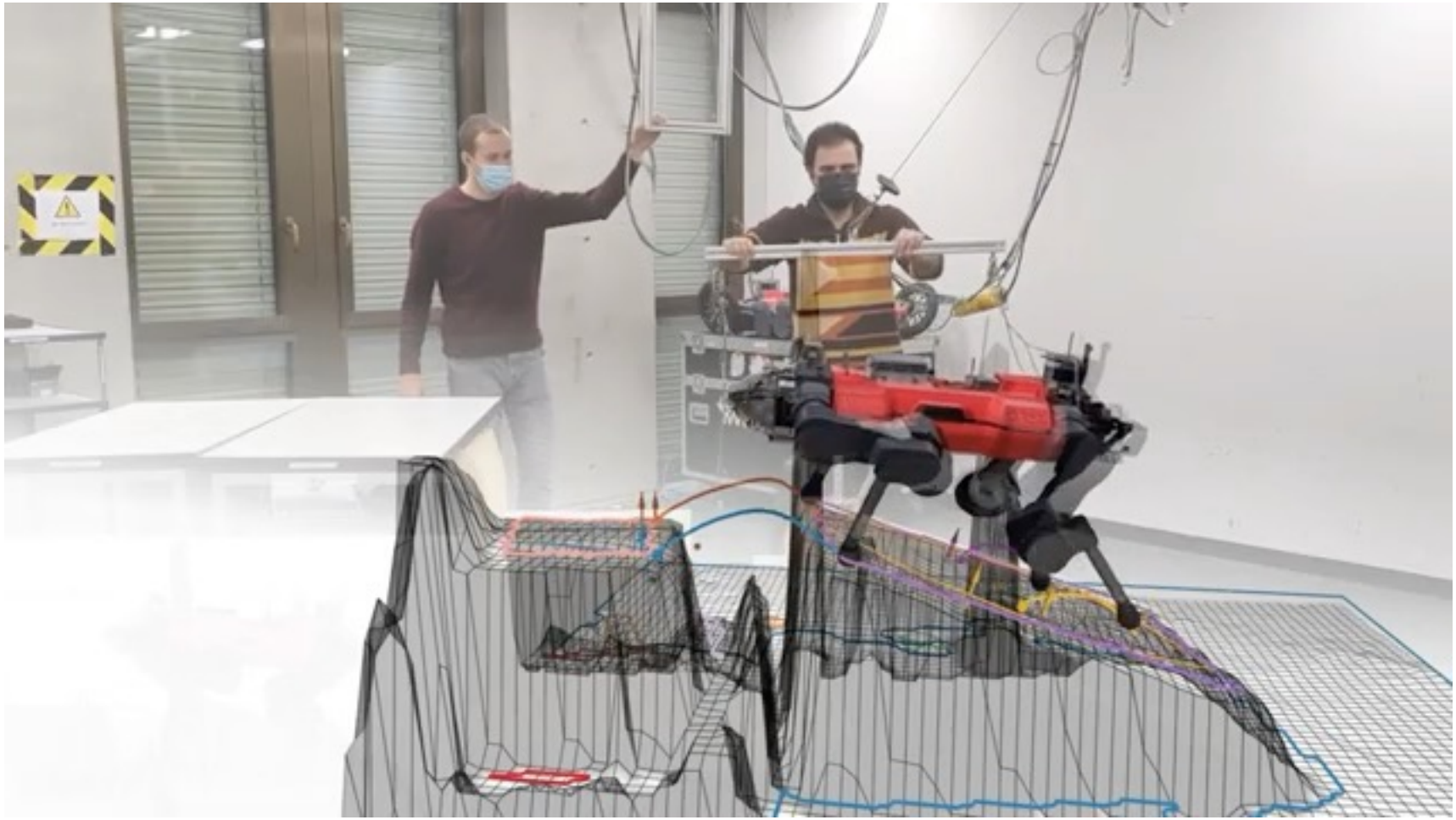
Applications of MPC

- Boston Dynamics Atlas



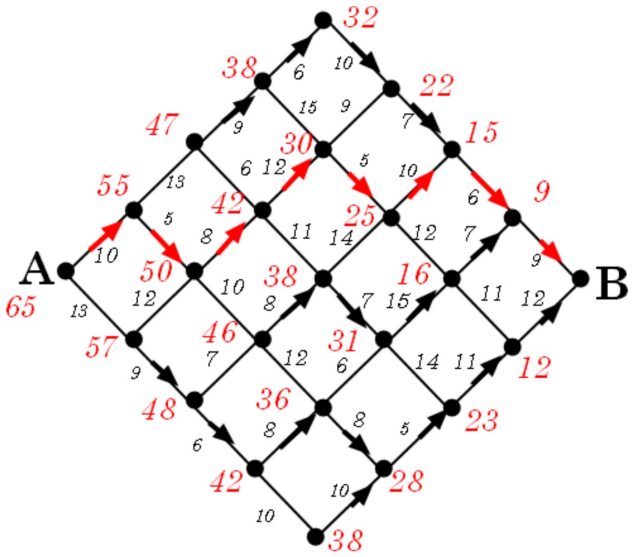
Applications of MPC

- ETH RSL Anymal

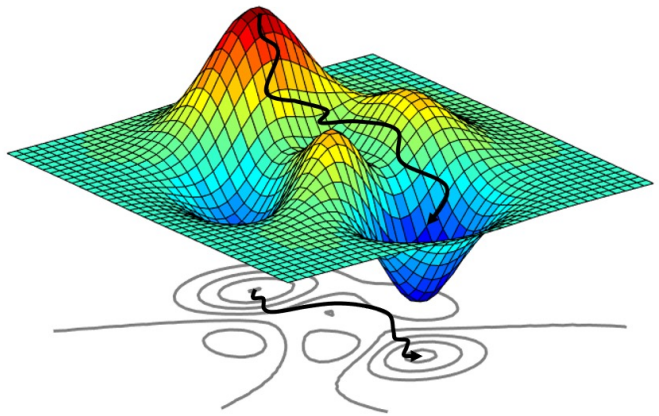


MPC solutions

- The core of MPC is to solve the inner-loop **local optimization problem**
- There are two main branches



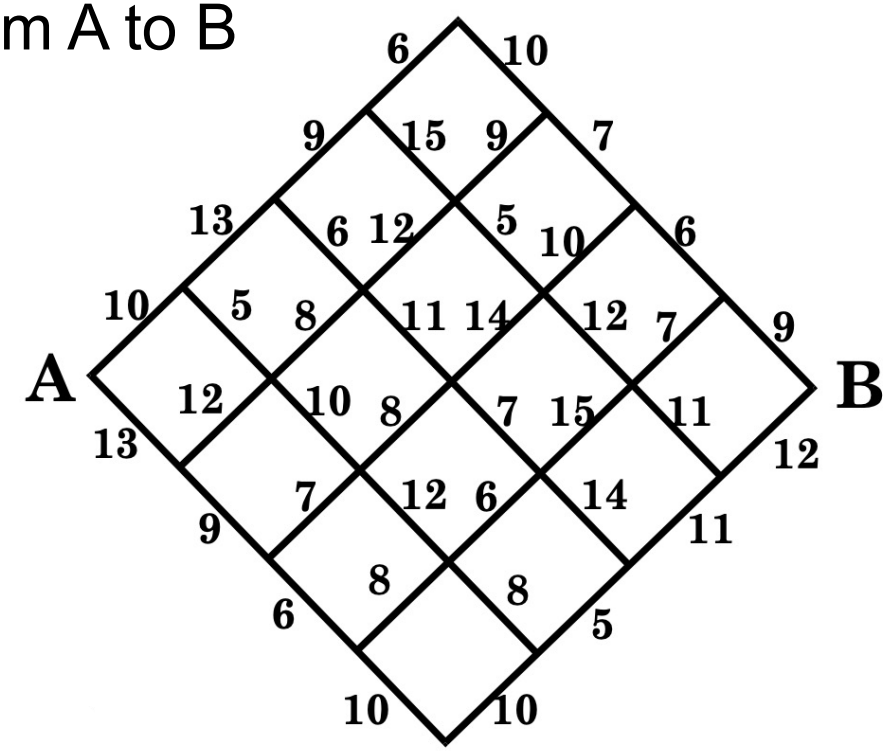
Optimal control

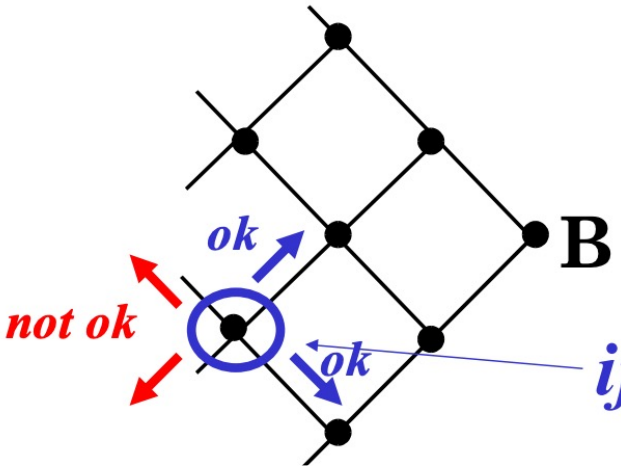


Trajectory optimization

Shortest Path Problem

- **Illustrative example:**
  - Find “optimal” path from A to B
- **Description:**
  - Moving only to the right

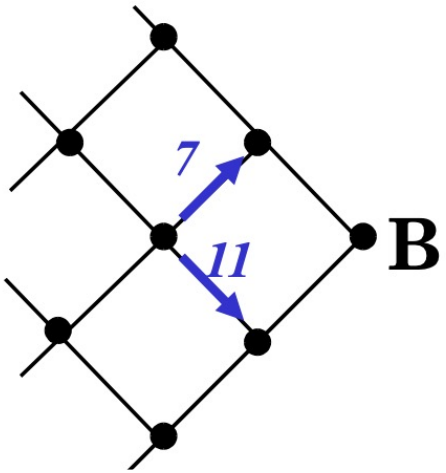


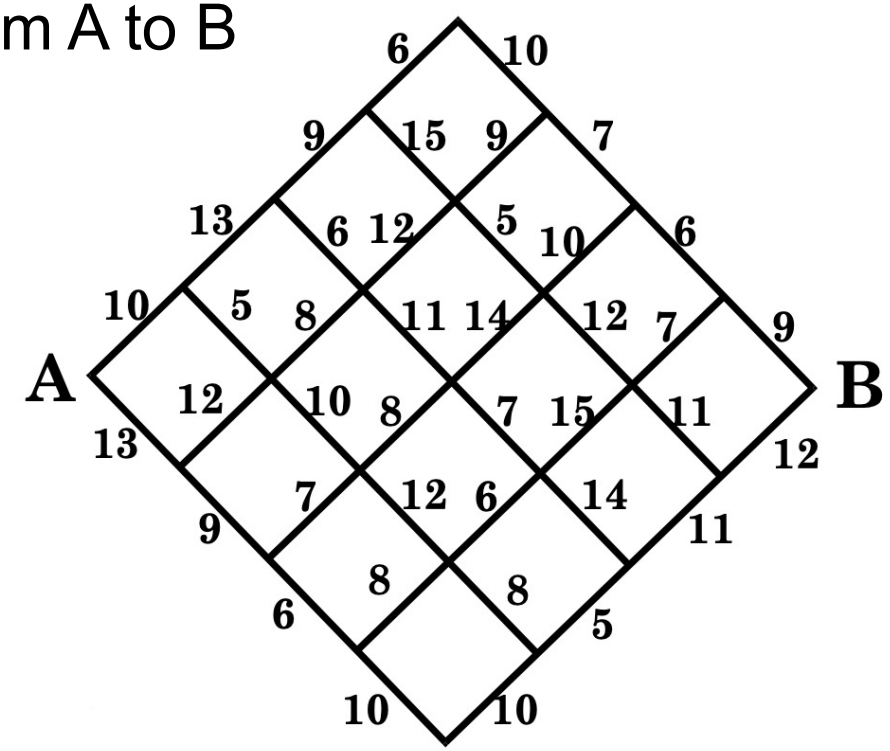


if we are here…

Shortest Path Problem

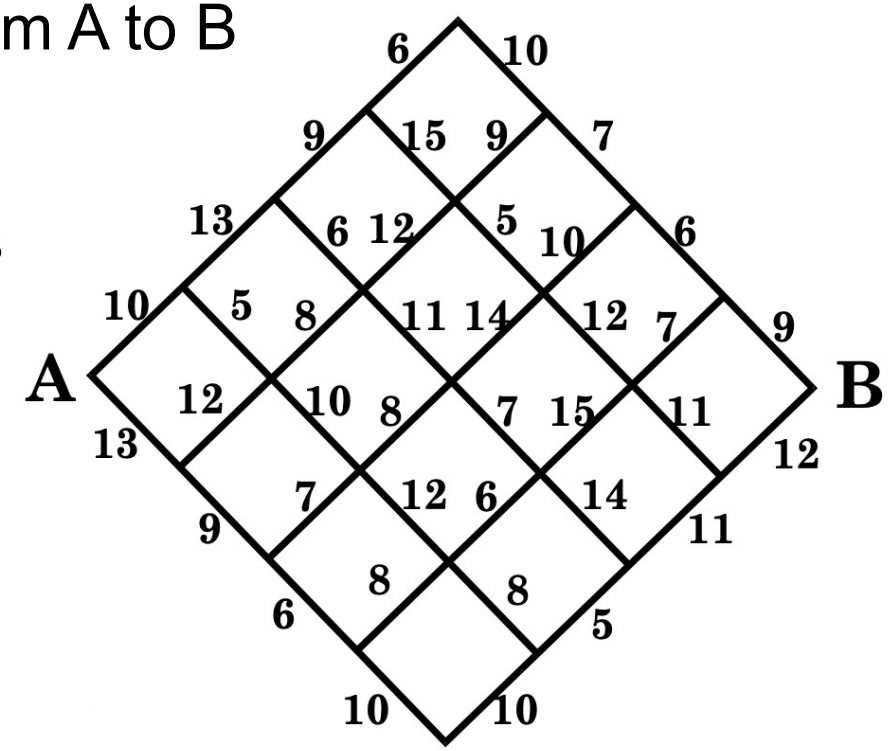
- **Illustrative example:**
  - Find "optimal" path from A to B
- **Description:**
  - Number next to line is the "cost" in going along a particular path





Shortest Path Problem

- **Illustrative example:**
  - Find "optimal" path from A to B
- **Description:**
  - Optimal path from A to B is the one with the smallest overall cost
  - There are 70 possible routes starting from A
- **How to solve efficiently?**



Dynamic Programming

Main idea (Principle of Optimality)

"From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point"

- R. Bellman (1956)

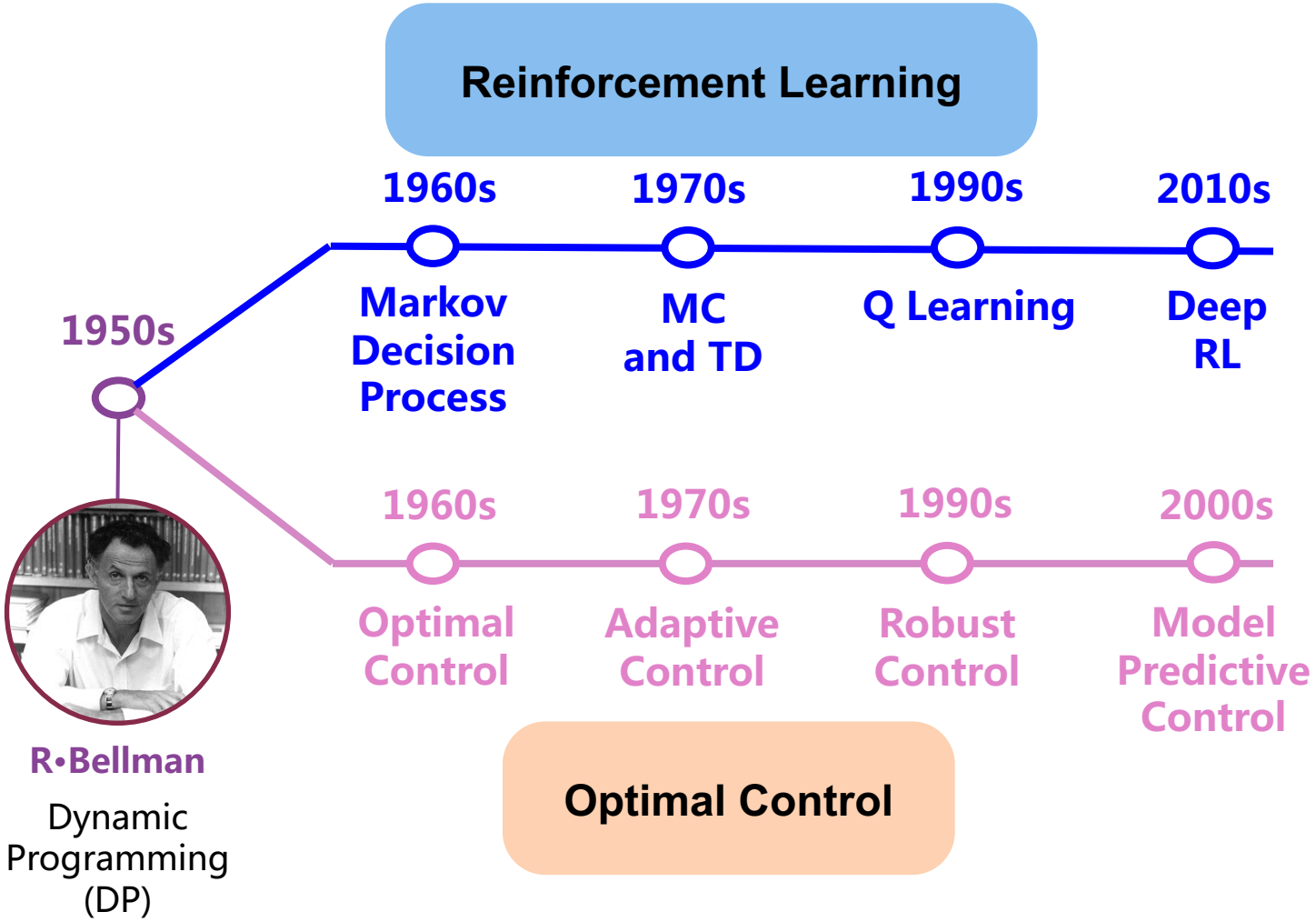
Developed by Richard Bellman in 1956

- Computer science
- Control theory
- Optimization
- Operation
- Economics
- Bioinformatics

R. Bellman
(1920-1984)

Reinforcement Learning and Optimal Control

- Dynamic programming is the foundation of both reinforcement learning and optimal control



Dynamic Programming

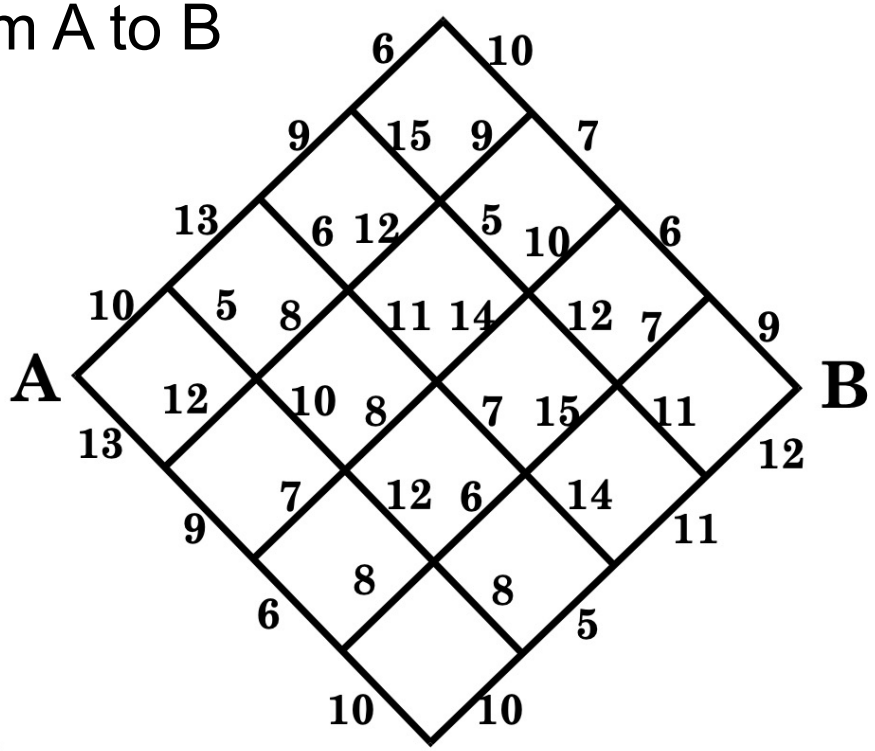
- Main idea **(Principle of Optimality)**

> "From any point on an optimal trajectory, the remaining trajectory is optimal for the corresponding problem initiated at that point"
>
> - R. Bellman (1956)

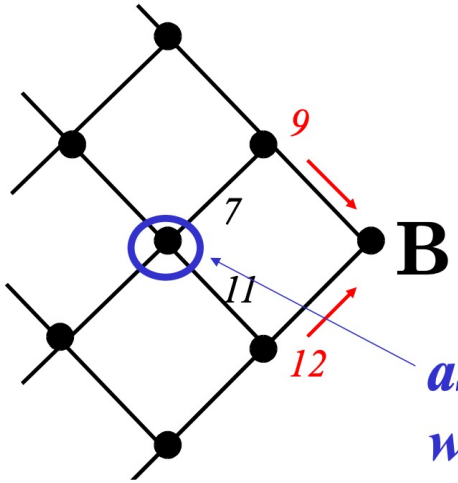
- How is this useful?
  - Only need to optimize the current one step if the optimal trajectory is already obtained for the next step
  - Multiple (infinite) step optimization problem breaks down to **one step optimization**

DP for Shortest Path Problem

- **Illustrative example:**
  - Find "optimal" path from A to B
- **Solution:**
  - Compute optimal total cost and path from each state



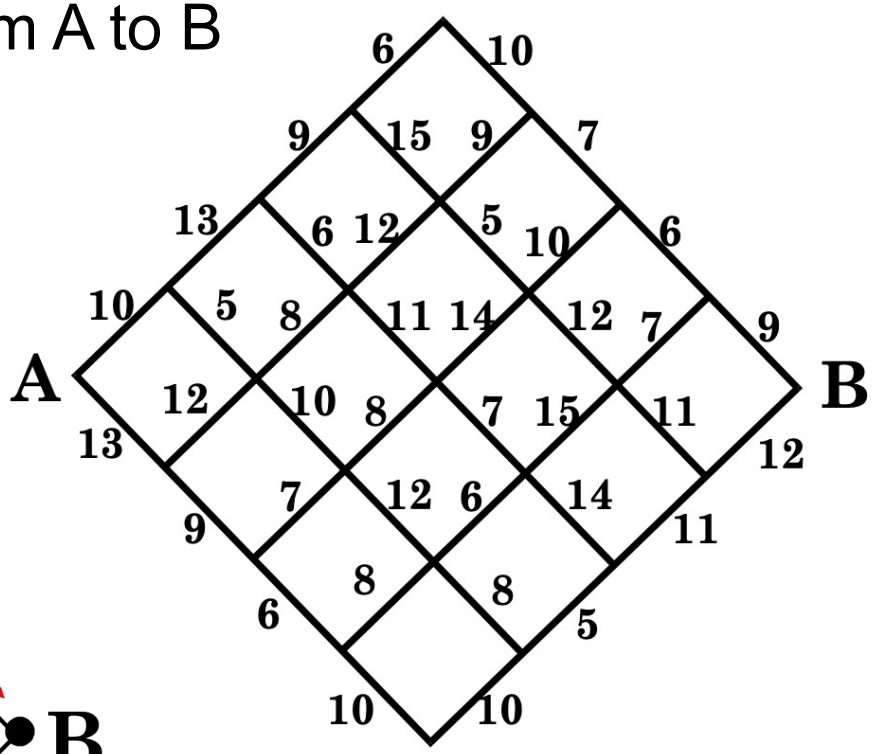
determine the optimal path from ⊙ *to* B



assume that we are here…

DP for Shortest Path Problem

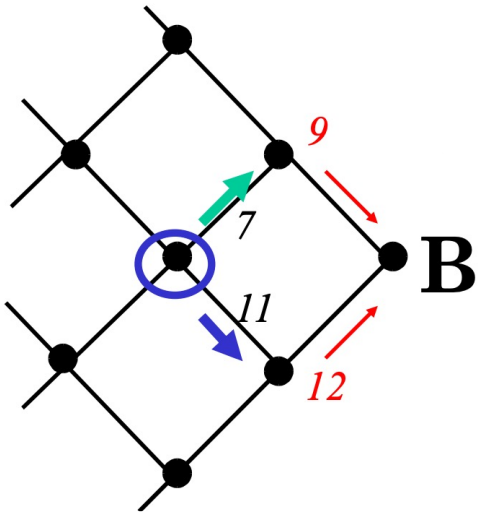
- **Illustrative example:**
  - Find "optimal" path from A to B
- **Solution:**
  - Compute optimal total cost and path from each state



two options:

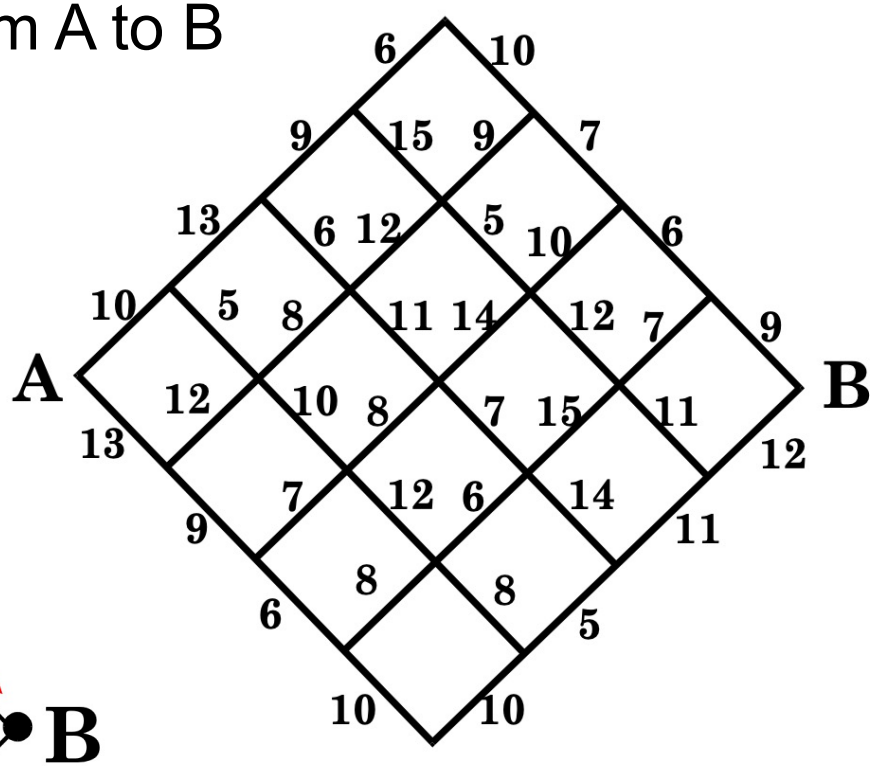
$7 + 9 = 16$

$11 + 12 = 23$



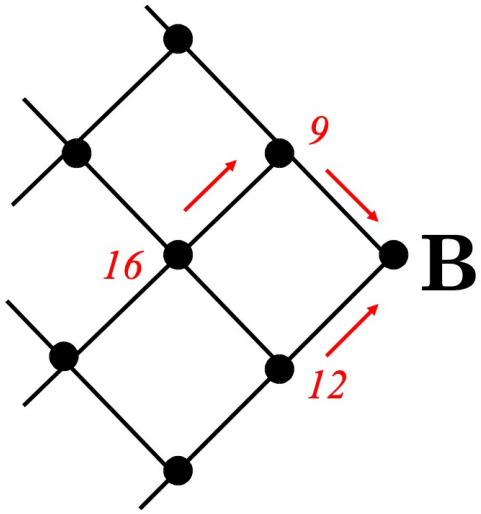
DP for Shortest Path Problem

- **Illustrative example:**
  - Find "optimal" path from A to B
- **Solution:**
  - Compute optimal total cost and path from each state



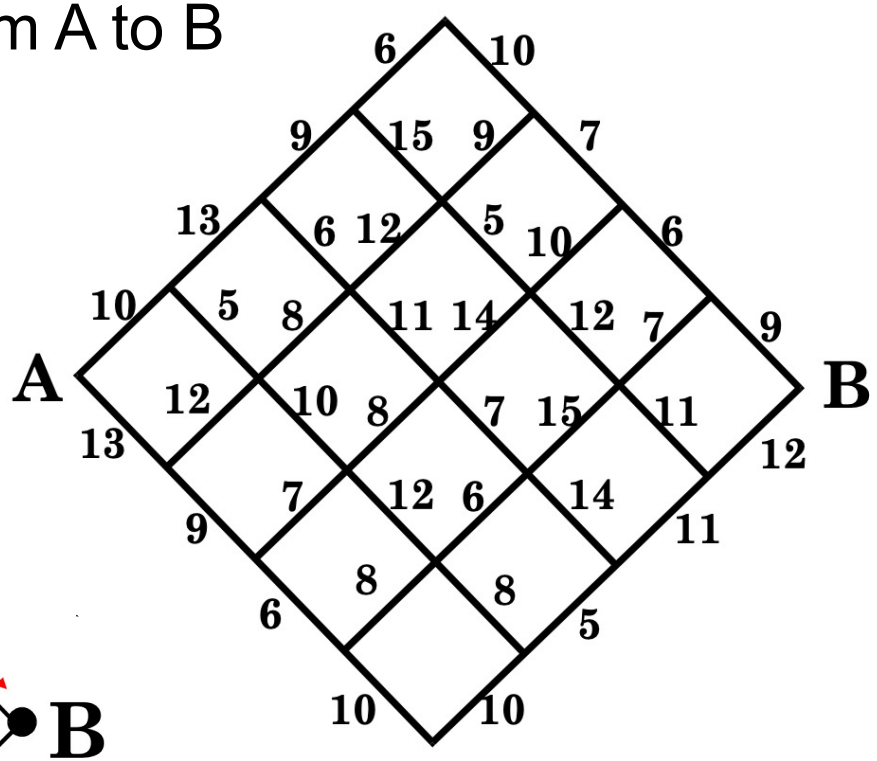
Assign:

- *optimal path*
- *optimal cost*



DP for Shortest Path Problem

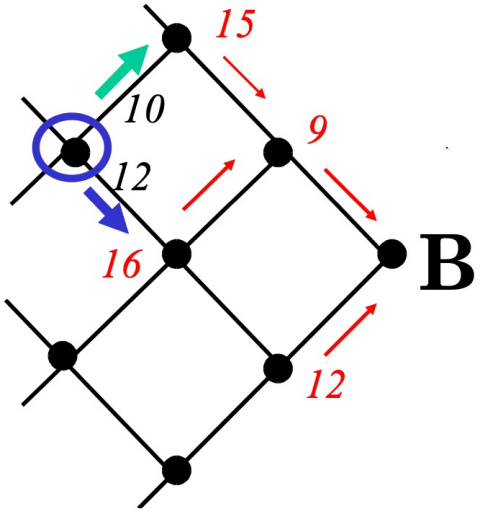
- **Illustrative example:**
  - Find "optimal" path from A to B
- **Solution:**
  - Compute optimal total cost and path from each state



Continue…

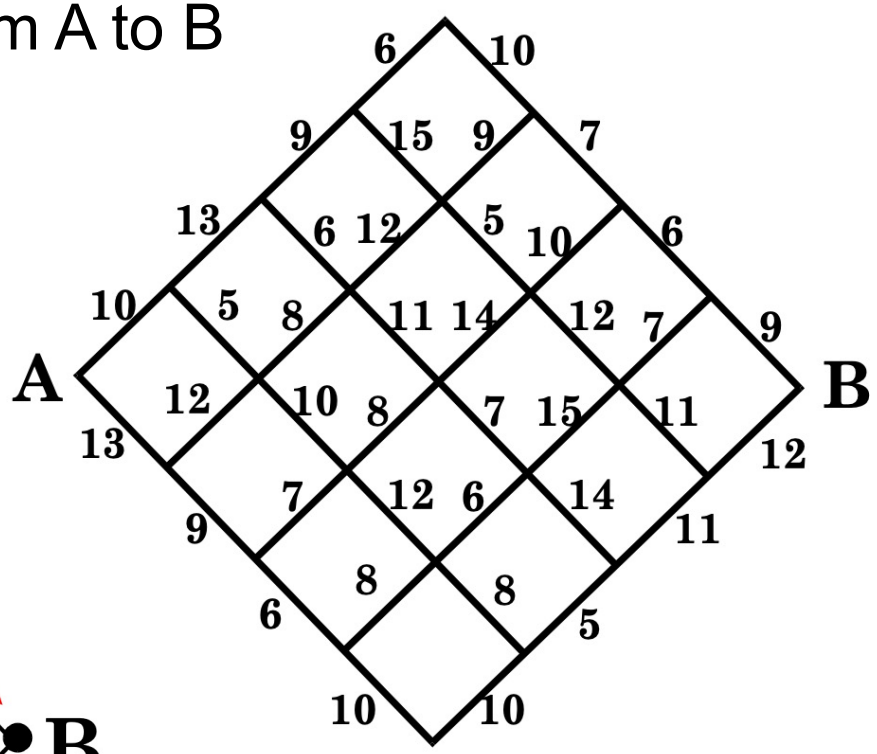
$10 + 15 = 25$

$12 + 16 = 28$



DP for Shortest Path Problem

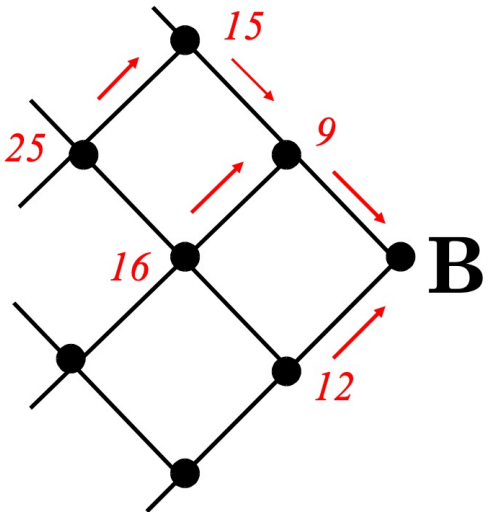
- **Illustrative example:**
  - Find "optimal" path from A to B
- **Solution:**
  - Compute optimal total cost and path from each state



Continue…

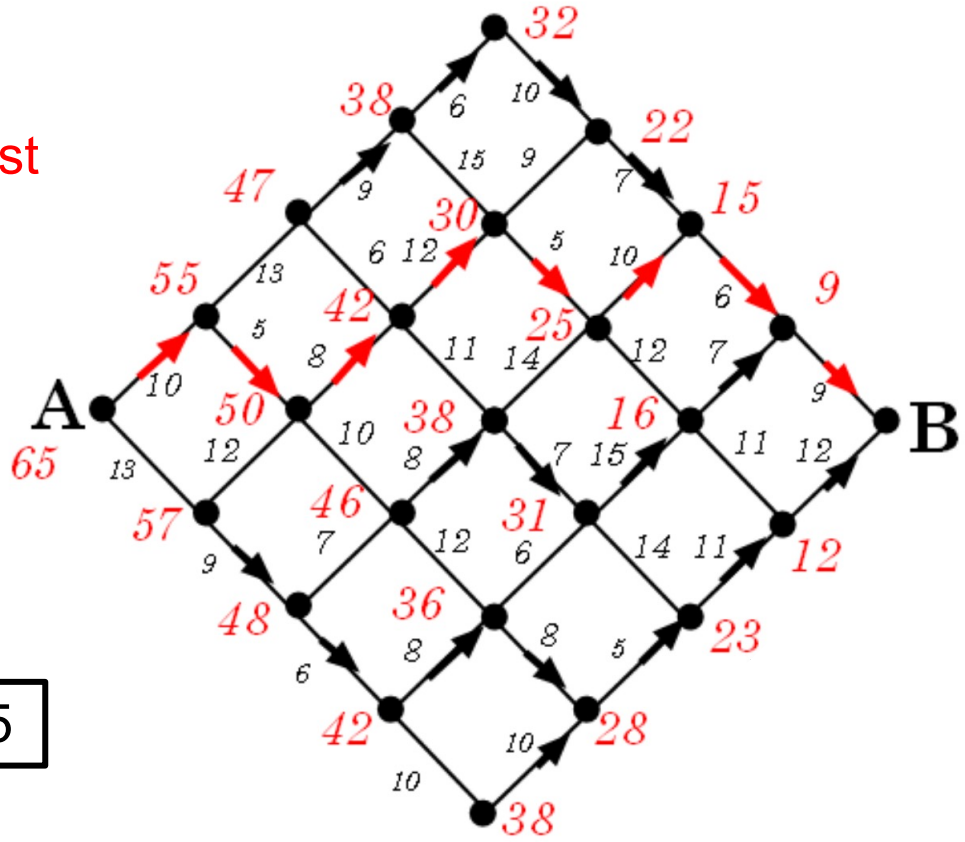
$10 + 15 = 25$

$12 + 16 = 28$



DP for Shortest Path Problem

- **Illustrative example:**
  - Find "optimal" path from A to B
- **Solution:**
  - Compute optimal total cost and path from each state



Optimal cost: 65

Bellman Equation

- **Optimal cost-to-go**

$$J^*(s_i) = \min_{a[\cdot]\in\mathcal{A}} \sum_{k=0}^{\infty} l(s[k], a[k]), \quad s[0] = s_i \in \mathcal{S}$$

cost-to-go
Similar to "value function"

Hard to search

- **Bellman Equation** via Dynamic Programming

$$J^*(s_i) = \min_{a\in\mathcal{A}} \{l(s_i, a) + J^*(f(s_i, a))\}, \quad \forall s_i \in \mathcal{S}$$

Easier to search

cost of the current step

Optimal cost-to-go starting from the next step

Bellman Operation

- How to solve for the Bellman Equation?

$$J^*(s_i) = \min_{a \in \mathcal{A}} \left\{ l(s_i, a) + J^*(f(s_i, a)) \right\}, \quad \forall s_i \in \mathcal{S}$$

- The right hand-side can be regard as an operation over a function:

$$J^* = \mathcal{B} J^*$$

- We can regard the cost-to-go as a point in the function space

$$J \in \mathcal{J}$$

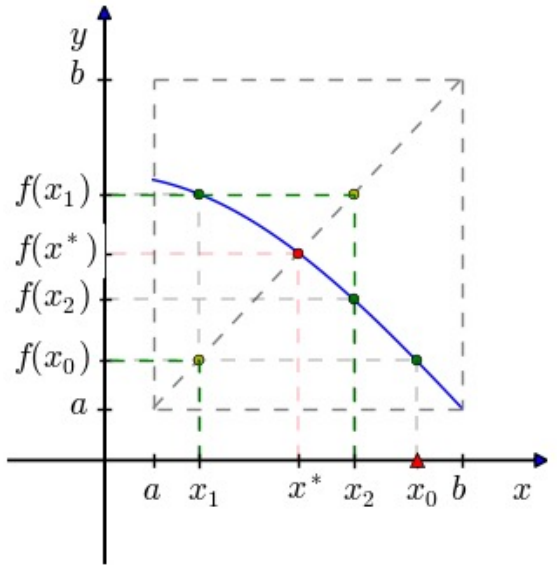
- J^* is then a fixed point of the Bellman operation

Fixed Point Iteration

- How to solve the fixed point for. $J^* = \mathcal{B}J^*$?
  - Fixed point iteration

$$x \leftarrow f(x)$$

$$x^* = f(x^*)$$



  - Fixed-point iteration for the cost-to-go function

$$J_{n+1}(s_i) \leftarrow \min_{a \in \mathcal{A}} \{l(s_i, a) + J_n(f(s_i, a))\}, \quad \forall s_i \in \mathcal{S}$$

Until $J_n \rightarrow J^*$

Value Iteration

Value Iteration algorithm

Initialize **cost-to-go** arbitrarily;

Repeat:

for $i = 1, ..., N$:

$$J(s_i) \leftarrow \min_{a \in \mathcal{A}} \{l(s_i, a) + J(f(s_i, a))\}$$

until J converges;

Extract **optimal policy**:

for $i = 1, ..., N$:

$$\pi(s_i) = \operatorname*{argmin}_{a \in \mathcal{A}} \{l(s_i, a) + J(f(s_i, a))\}$$

Extension to Continuous Space

- Can we extend to **continuous states and actions?**
  - Intractable for general nonlinear systems

$$J^*(s) = \min_{a \in \mathcal{A}} \left\{ l(s,a) + J^*(f(s,a)) \right\}, \quad \forall s \in \mathcal{S}$$

  - RL provides promising solutions, but still suffer to function approximation and sampling issues

$$J_\theta(s) \approx \min_{\pi_\phi} \left\{ l(s, \pi_\phi(s)) + J_\theta(f(s, \pi_\phi(s))) \right\}, \quad \forall s \in \mathcal{S}$$

- But, there are special types of continuous systems where **exact solutions exist**

Linear Quadratic Regulator

- Consider **linear system** with continuous states and controls, in discrete time

$$x[k+1] = Ax[k] + Bu[k]$$

$$\mathbb{R}^n \qquad \mathbb{R}^{n\times n} \qquad \mathbb{R}^{n\times m} \qquad \mathbb{R}^m$$

- With initial state

$$x(0) = x_0$$

- The goal is to find the optimal control to obtain the optimal cost-to-go

$$J^*(x[0]) = \min_{u[\cdot]} \sum_{k=0}^{\infty} l(x[k], u[k])$$

Linear Quadratic Regulator

- The cost is **quadratic**

$$l(x[k], u[k]) = x[k]^T Q x[k] + u[k]^T R u[k]$$

$$= \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}$$

$x[k]^T Q x[k]$ Penalize the **state deviation** from the origin

$u[k]^T R u[k]$ Penalize the **control effort**

- Require $Q \succeq 0$ $R \succ 0$

Linear Quadratic Regulator

- **A Linear Quadratic Regulator (LQR) problem is**
  - For a **linear system**

$$x[k+1] = Ax[k] + Bu[k] \qquad x(0) = x_0$$

  - And **quadratic cost**

$$l(x[k], u[k]) = x[k]^T Q x[k] + u[k]^T R u[k]$$

$$Q \succeq 0 \qquad R \succ 0$$

  - Find the optimal control that **minimize the cost-to-go**

$$J^*(x[0]) = \min_{u[\cdot]} \sum_{k=0}^{\infty} \left\{ \begin{bmatrix} x[k] \\ u[k] \end{bmatrix}^T \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \begin{bmatrix} x[k] \\ u[k] \end{bmatrix} \right\}$$

Dynamic Programming for LQR

- **How to solve the Bellman Equation for LQR?**

$$J^*(x) = \min_{u(x)} \left\{ l(x,u) + J^*(f(x,u)) \right\}, \quad \forall x$$

Feedback control law based on x

Solve for all states

- **The Value Iteration for LQR**

$$J_{n+1}(x) \leftarrow \min_{u(x)} \left\{ l(x,u) + J_n(f(x,u)) \right\}, \quad \forall x$$

Solution for LQR

- **By induction, we have that at each iteration**
  - The **cost-to-go** is

$$J_n(x) = x^T P_n x$$

  - The **control policy** is

$$u_n^*(x) = -\left[B^T P_n B + R\right]^{-1} B^T P_n A x$$

  - where

$$P_{n+1} = A^T P_n A + Q - A^T P_n B \left[B^T P_n B + R\right]^{-1} B^T P_n A$$

Solution for LQR

- **Recall value iteration for LQR**

$$J_{n+1}(x) \leftarrow \min_{u(x)} \left\{ l(x,u) + J_n(f(x,u)) \right\}, \quad \forall x$$

- **The solution at convergence?**

$$P_{n+1} = A^T P_n A + Q - A^T P_n B \left[B^T P_n B + R \right]^{-1} B^T P_n A$$

$$n \to \infty$$

$$P_\infty = A^T P_\infty A + Q - A^T P_\infty B \left[B^T P_\infty B + R \right]^{-1} B^T P_\infty A$$

Solution for LQR

- The **value iteration** for LQR

$$J_{n+1}(x) \leftarrow \min_{u(x)} \{l(x,u) + J_n(f(x,u))\}$$

- Solution for LQR
  - The **optimal cost-to-go** is

$$J^*(x) = x^T P x$$

  - The **optimal control policy** is

$$u^*(x) = -\left[B^T P B + R\right]^{-1} B^T P A x$$

  - The solution is dependent on the **Discrete Algebraic Riccati Equation (DARE)**

$$P = A^T P A + Q - A^T P B \left[B^T P B + R\right]^{-1} B^T P A$$

which can be solved as $P = \mathrm{dare}(A, B, Q, R)$

Iterative LQR

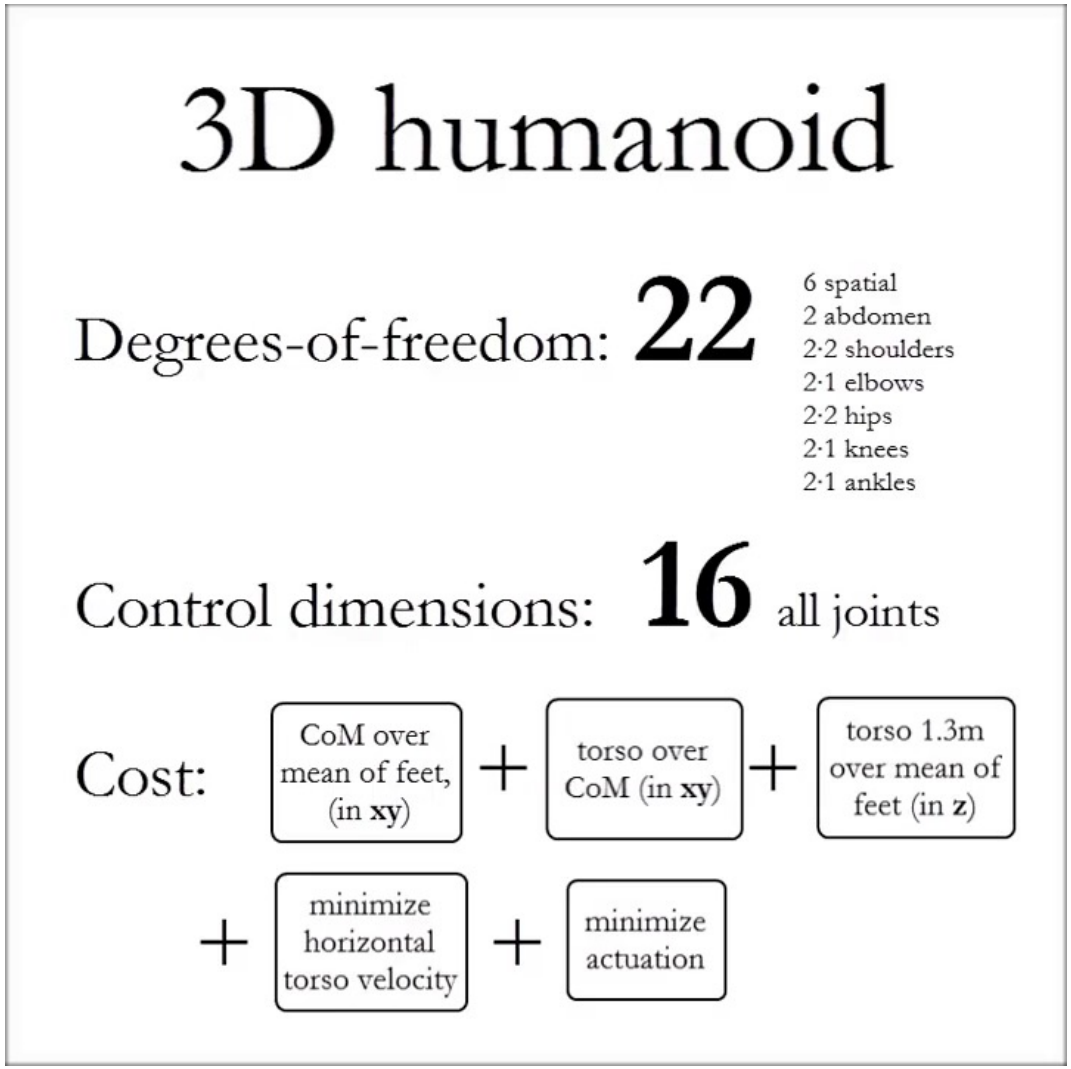
- The **Iterative LQR** algorithm
  - Given an **initial trajectory** $x_0(\cdot), u_0(\cdot)$
  - **Linearize** the dynamics
  - Using the **quadratic approximation** of the cost

$$l(x,u) \approx l(x_0,u_0) + \frac{\partial l}{\partial x}(x-x_0) + \frac{\partial l}{\partial u}(u-u_0)$$
$$+(x-x_0)^T \frac{\partial^2 l}{\partial x^2}(x-x_0) + \ldots$$

  - Solve the resulting **LQR** problem
  - **Simulate** a new trajectory with the **original dynamics**
  - Continue iteration

Example

- MPC with ILQR



Contents

Optimization Basics

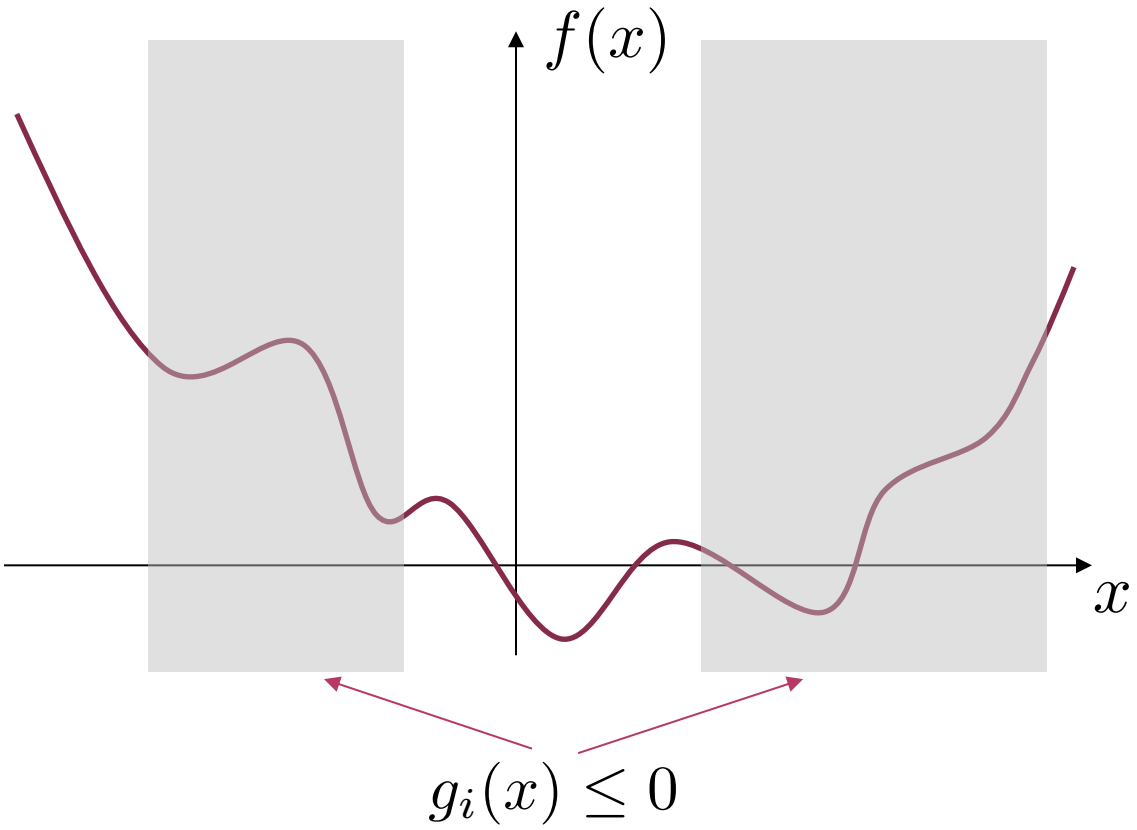
- Optimization problem formulation

$$\min_{x} f(x) \quad \text{s.t.} \quad \forall i, g_i(x) \leq 0$$

Decision variables

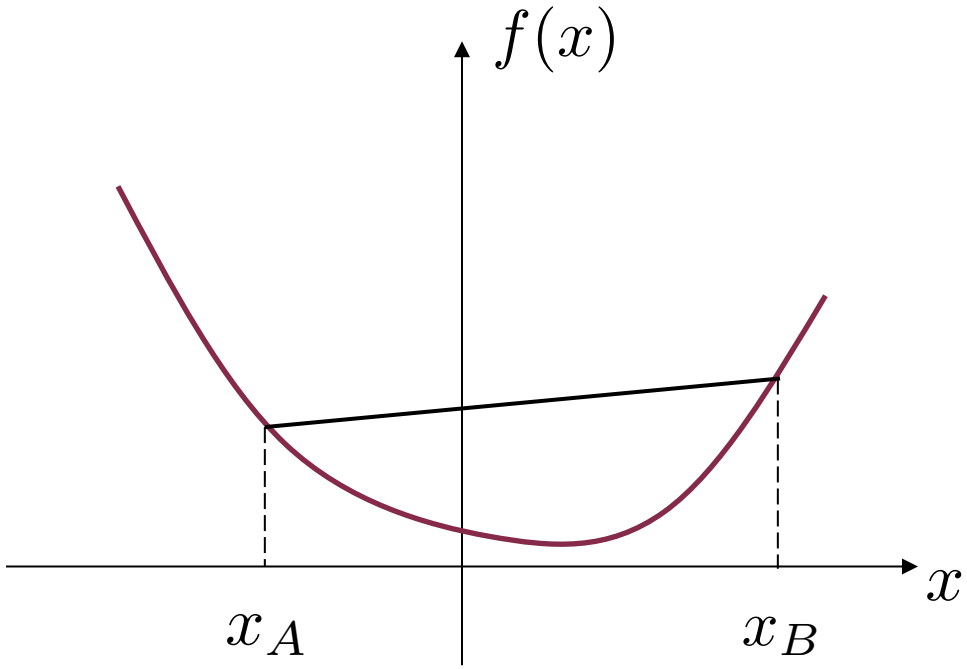
Scalar **objective**

Vector **constraints**



Optimization Basics

- A more special case is **convex optimization**
- **Convex function**

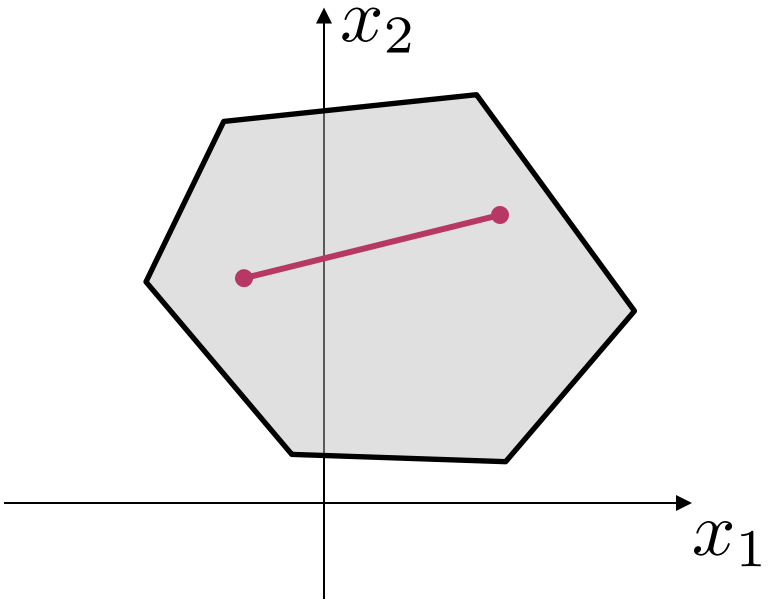


$$\forall x_A, x_B \qquad 0 \leq \alpha \leq 1$$

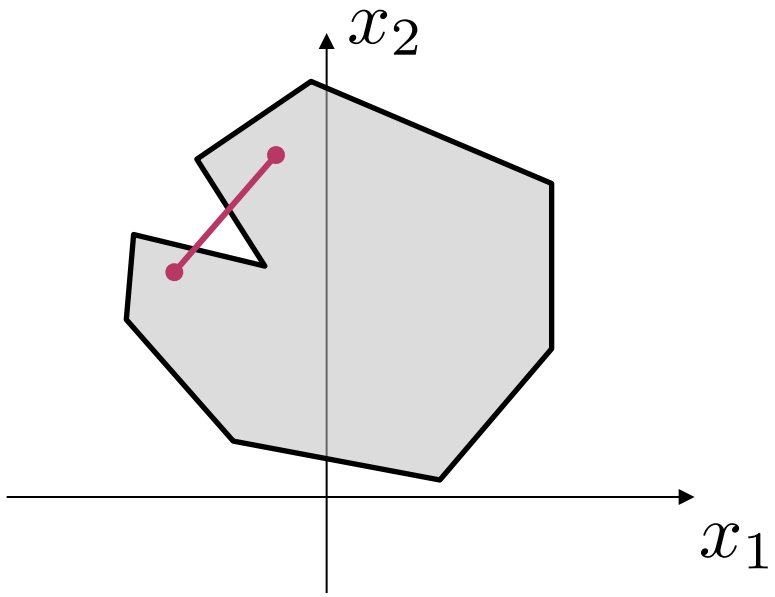
$$\alpha f(x_A) + (1-\alpha) f(x_B) \geq f(\alpha x_A + (1-\alpha) x_B)$$

Optimization Basics

- **Convex set**



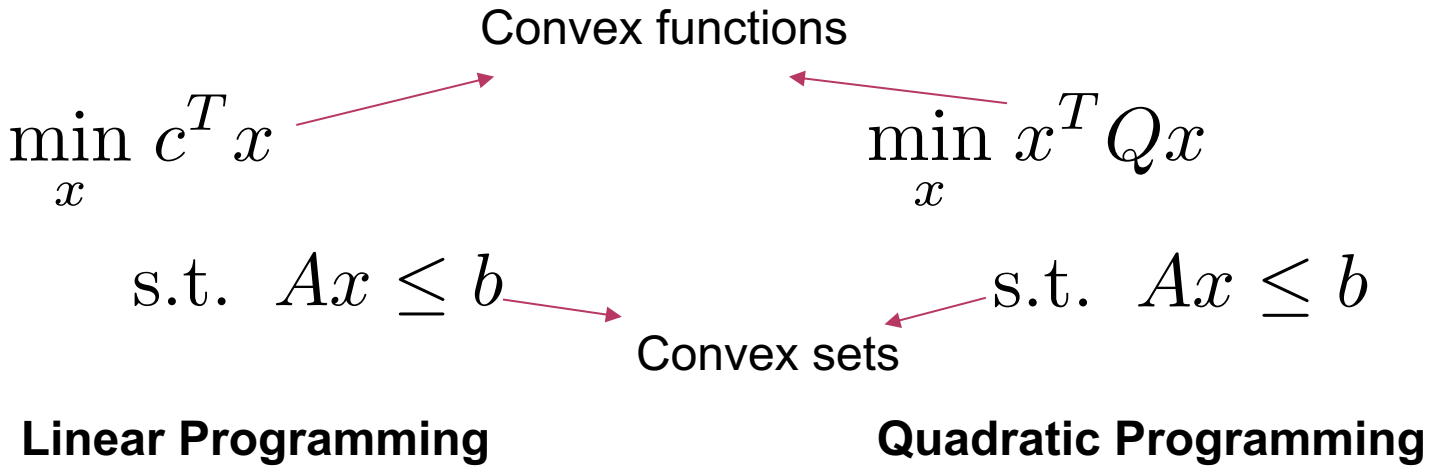
Convex



Non-Convex

Optimization Basics

- **Convex optimization problems**



- **Linear Programming (LP)** and **Quadratic Programming (QP)** are all **convex**
- It is relatively easy to reliably obtain the **global optimum** for LP and QP

Trajectory Optimization for LQ Problems

- Let's start with **linear dynamics** with **quadratic costs**

$$
\begin{aligned}
u[0:T], x[0:T] = & \underset{u[0:T], x[0:T]}{\operatorname{argmin}} \sum_{k=0}^{T} x^T Q x + u^T R u \\
\text{s.t.} \quad & x[k+1] = Ax[k] + Bu[k] \\
& x[0] = x_0 \\
& Cx + Du \leq b
\end{aligned}
$$

- This is a **Quadratic Programming (QP)** problem, where a mature global optimum solution is available

Direct Transcription

- **Direct transcription**
  - Both **x and u** are decision variables
  - Use **explicit constraint** for system dynamics

$$u[0:T], x[0:T] = \underset{u[0:T], x[0:T]}{\operatorname{argmin}} \sum_{k=0}^{T} x^T Q x + u^T R u$$

$$\text{s.t. } x[k+1] = Ax[k] + Bu[k]$$

$$x[0] = x_0$$

$$Cx + Du \leq b$$

Direct Shooting

- Alternatively, since we know the initial condition, and also the control inputs, we are able to obtain the states by propagating the dynamics
- If we substitute states with the control inputs and initial condition, then this is called the direct shooting method
  - States are **not** decision variables
  - **No** explicit dynamics constraints

$$x[1] = Ax[0] + Bu[0]$$
$$x[2] = A(Ax[1] + Bu[0]) + Bu[1]$$
$$\dots$$

Nonlinear Trajectory Optimization

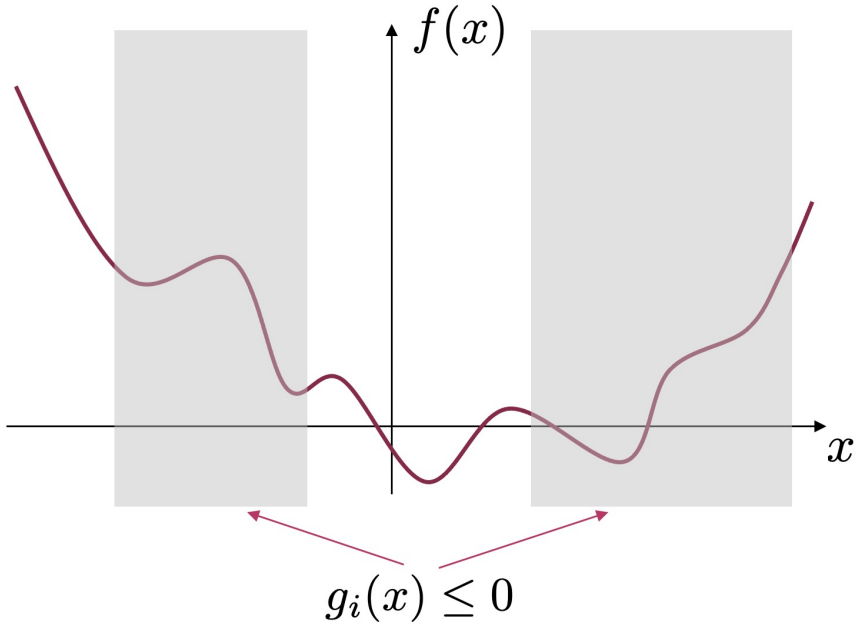
- Trajectory optimization for general **nonlinear systems**

$$\min_{u[0:T],x[0:T]} \sum_{k=0}^{T} l(x[k], u[k])$$

$$\text{s.t.} \quad x[k+1] = f(x[k], u[k])$$

$$x[0] = x_0$$

$$g_i(x[k], u[k]) \leq 0 \quad \forall i, k$$



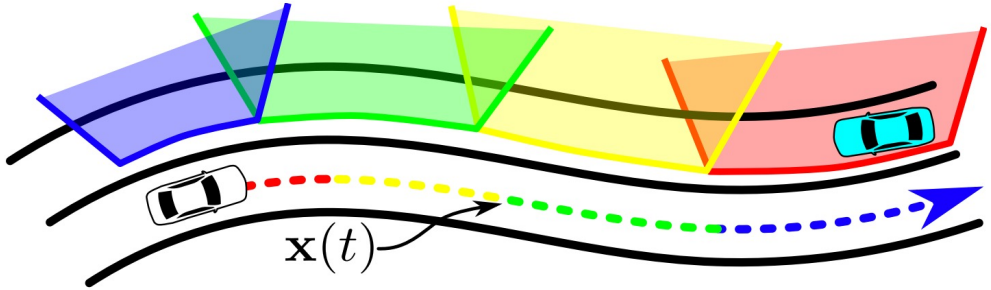
- This is a **non-convex optimization** problem, where only **local optimum** solutions are available

Nonlinear Trajectory Optimization

- **Local optimum**
  - Solution depends on initial trajectory

- **Avoid local optimum**

$t \in [0, 1)$
$t \in [1, 2)$
$t \in [2, 3)$
$t \in [3, 4)$



Nonlinear Trajectory Optimization

- There are several nonlinear trajectory optimization solvers
- **Sequential Quadratic Programming (SQP)**
  - Quadratic approximation of costs
  - Linear approximation of constraints
  - Solve the QP iteratively
- **Penalty based methods**
  - **Augmented Lagrangian**
  - **Interior points**

Examples

- Complex behavior generation

Example

- Boston Dynamics Atlas

https://www.youtube.com/watch?v=EGABAx52GKI

Optimal Control vs Trajectory Optimization

- **Optimal control**
  - Fast solving
  - Exact solution for linear quadratic case
  - Can extend to continuous time and infinite horizon
  - Approximate solution for nonlinear case
  - Hard to handle constraint
- **Trajectory optimization**
  - General solution
  - Computation efficiency issue
  - Only for discrete time and finite horizon

Thank you!

清華大學 交叉信息研究院

Institute for Interdisciplinary Information Sciences, Tsinghua University

ISR Lab

Intelligent Systems and Robotics Lab