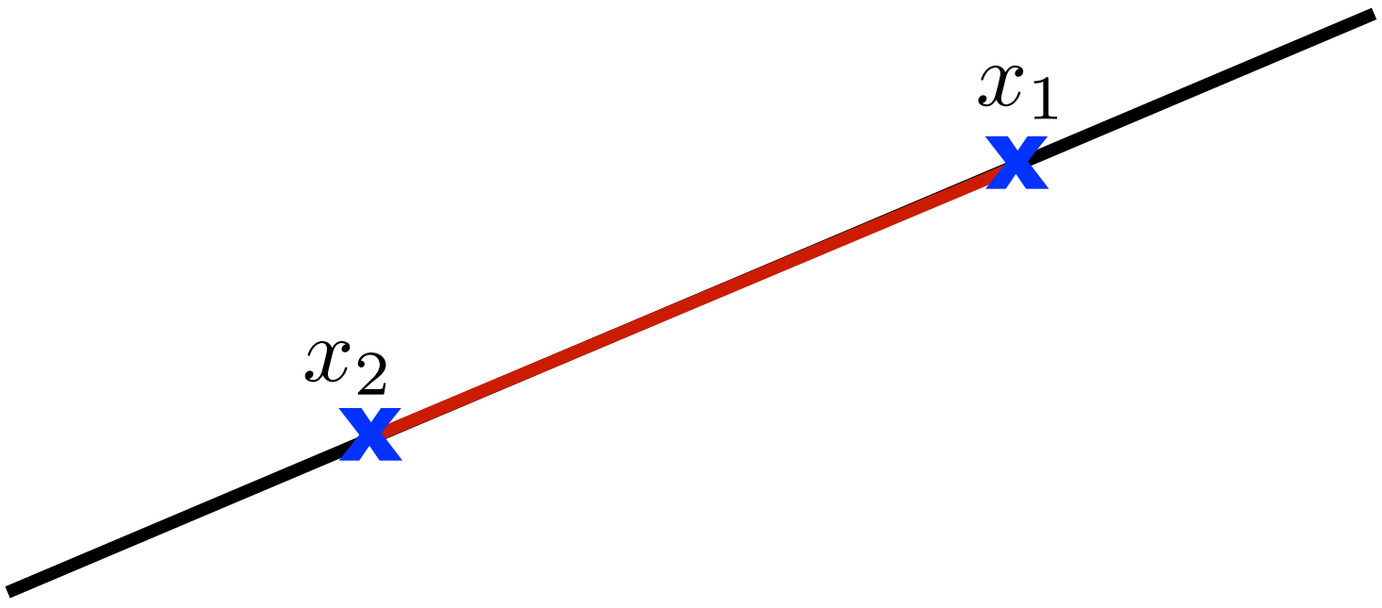
ECE-GY 9243 / ME-GY 7973

Optimal and Learning Control for Robotics

Ludovic Righetti

Lecture 11
Minimizing functions - Bellman Principle of Optimality

Convex Sets



$x_1, x_2 \in \mathbb{R}^n$

Points of the form $y = \theta x_1 + (1 - \theta)x_2$ **with** $\theta \in \mathbb{R}$

form the line passing through x_1 and x_2

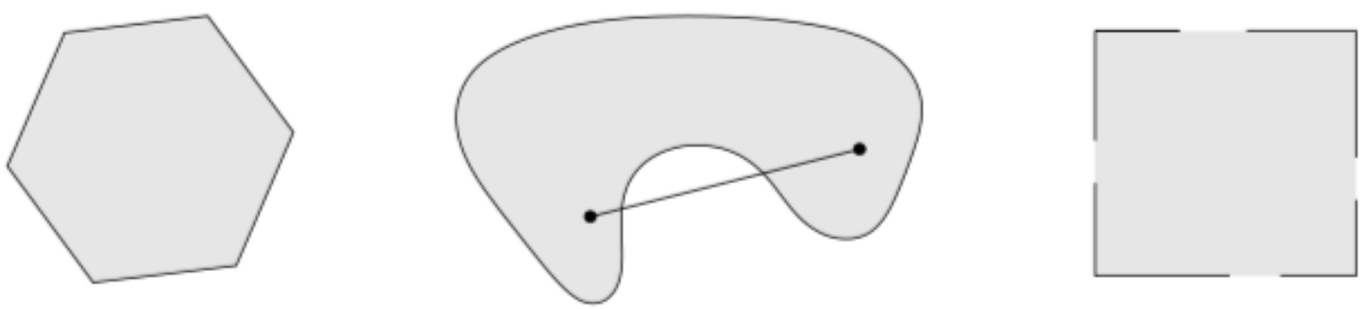
$$y = x_1 \text{ when } \theta = 1 \qquad y = x_2 \text{ when } \theta = 0$$

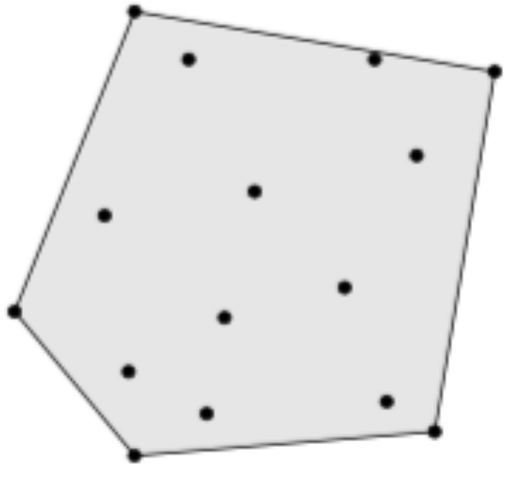
Points of the form $y = \theta x_1 + (1 - \theta)x_2$ **with** $\theta \in [0, 1]$

form the **closed line segment** between x_1 and x_2

Convex Sets

A set C is convex if the line segment between any two points in C lies in C, i.e. if for any x_1, $x_2 \in$ C and any $\theta \in [0, 1]$, we have $\theta x_1 + (1 - \theta)x_2 \in C$

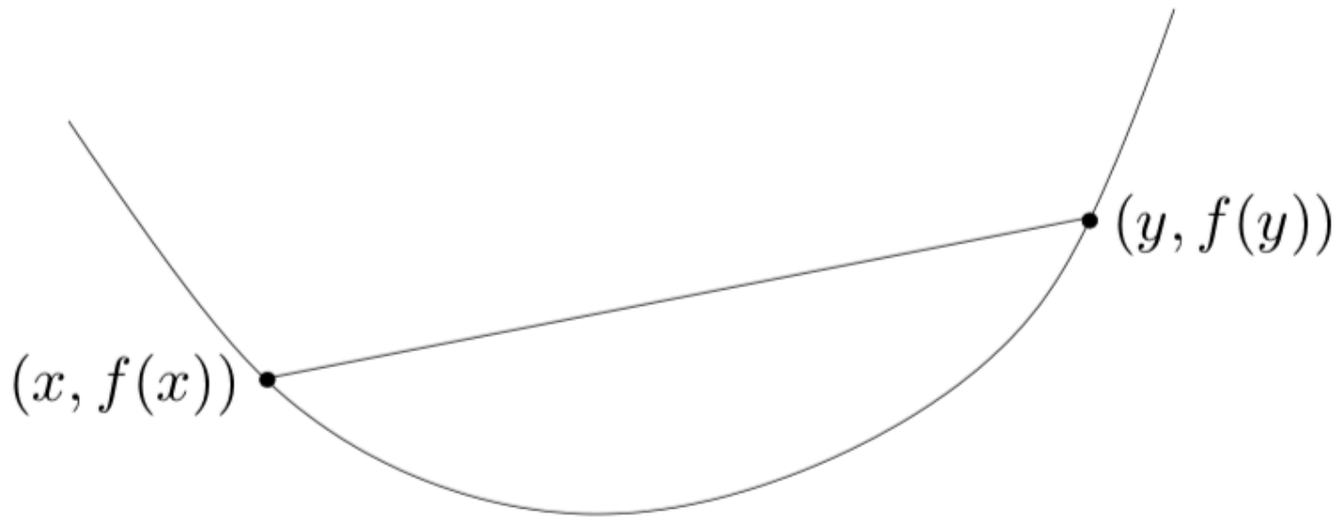




Convex Functions

A function $f : \mathbf{R}^n \to \mathbf{R}$ is *convex* if $\mathbf{dom}\, f$ is a convex set and if for all x, $y \in \mathbf{dom}\, f$, and θ with $0 \leq \theta \leq 1$, we have

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y)$$

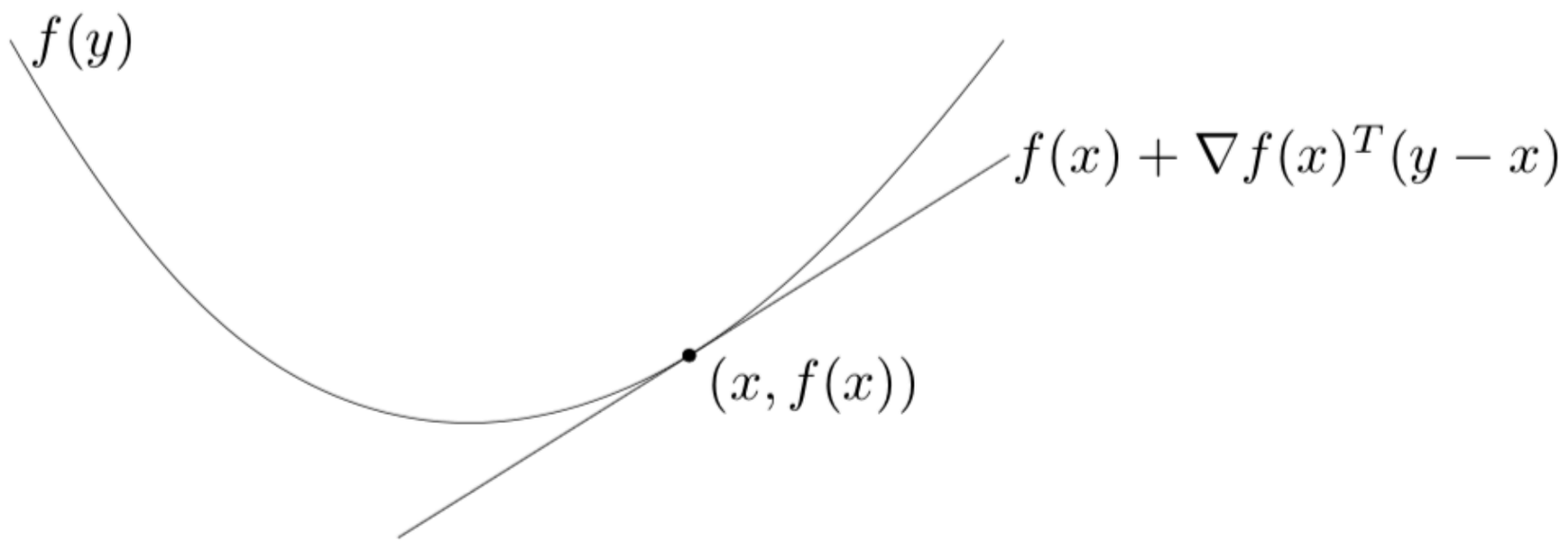


Convex Functions

First order condition for convexity

If f is differentiable (i.e. if its gradient ∇f exists at each point in **dom** f). Then f is convex if and only if

1. **dom** f is convex
2. $f(x) + \nabla f(x)^T (y - x) \leq f(y)$



Convex Functions

Second order condition for convexity

If f is twice differentiable (i.e. if its Hessian H exists at each point in **dom** f). Then f is convex if and only if

1. **dom** f is convex
2. $H(x) \geq 0$ (i.e. the Hessian is positive definite at each point)

$$\mathbf{H} = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Minimizing Convex Functions

$$\min_x f(x)$$

Any local minima of a convex function is also a global minima

If f is convex, a necessary and sufficient condition for x to be a minimum of f is that $\nabla f(x) = 0$

Minimizing Convex Functions with convex equality constraint

$$\min_x f(x)$$

$$\text{subject to } Ax = b$$

We introduce the Lagrangian of the optimization problem

$$L = f(x) + \lambda^T (Ax - b)$$

Necessary and sufficient condition for optimality

$$\begin{aligned} &\frac{\partial L}{\partial x} = 0 \\ &\frac{\partial L}{\partial \lambda} = 0 \end{aligned} \quad \Leftrightarrow \quad \begin{aligned} \nabla f(x) + A^T \lambda &= 0 \\ Ax &= b \end{aligned}$$

Convex optimization problem with inequality constraints

$$\min_x f(x)$$

$$\text{subject to} \ : \quad Ax = b$$

$$g(x) \leq 0 \qquad \text{where g(x) is convex}$$

We introduce the Lagrangian of the optimization problem
(Method of Lagrange Multipliers)

$$L = f(x) + \lambda^T (Ax - b) + \mu^T g(x)$$

Necessary and sufficient conditions for optimality
(Karush–Kuhn–Tucker conditions - KKT conditions)

$$\frac{\partial L}{\partial x} = 0 \qquad Ax = b \quad g(x) \leq 0$$

$$\mu_i \geq 0 \quad \mu^T g(x) = 0$$

Two easy-to-solve convex programs

Linear program (LP)

$$\begin{aligned} &\text{minimize } \mathbf{c}^T\mathbf{x} + \mathbf{d} \\ &\text{subject to } \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ &\mathbf{A}\mathbf{x} = b \end{aligned}$$

Efficient algorithms to solve large LPs
(e.g. simplex algorithm, interior-point methods)

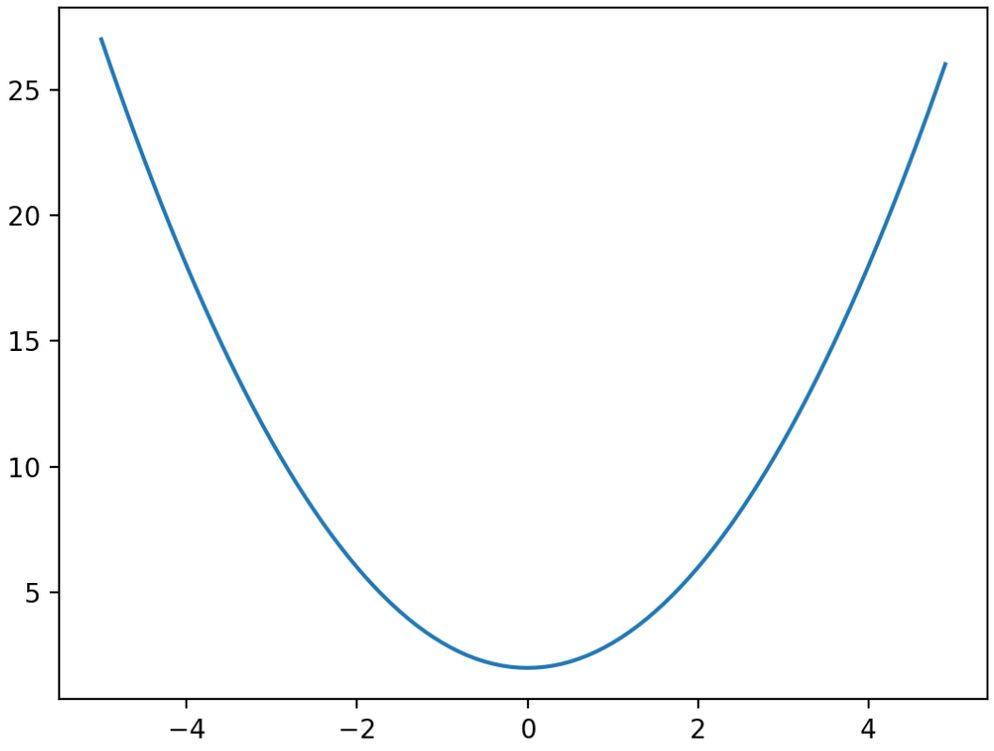
Quadratic program (QP)

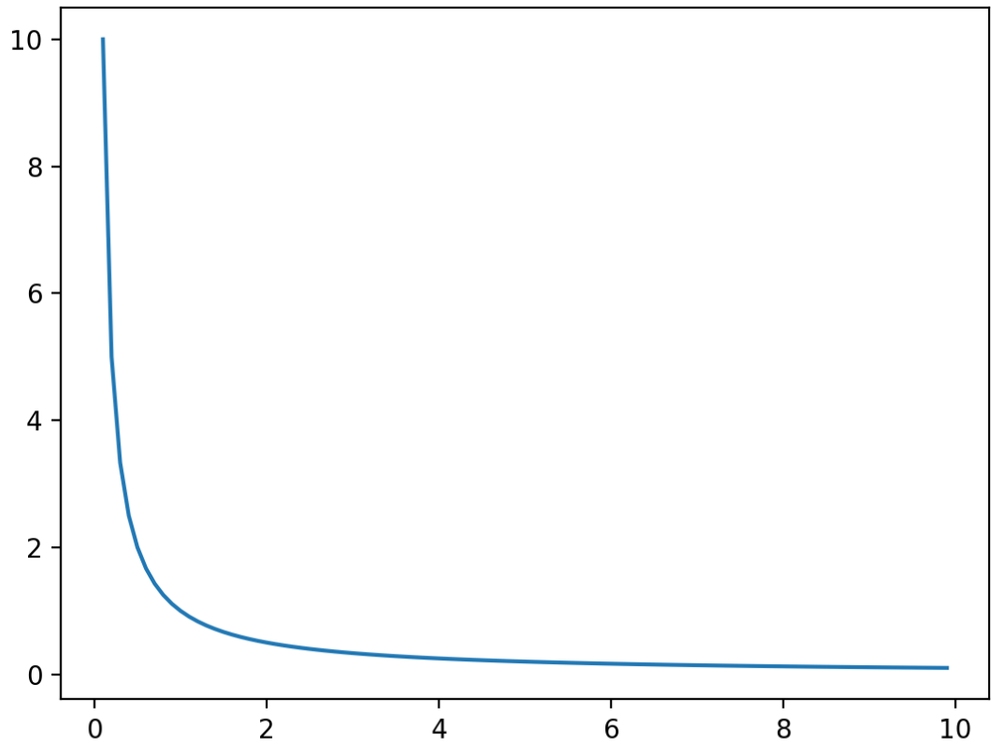
$$\begin{aligned} &\text{minimize } (1/2)\mathbf{x}^T\mathbf{P}\mathbf{x} + \mathbf{q}^T\mathbf{x} + \mathbf{r} \\ &\text{subject to } \mathbf{G}\mathbf{x} \leq \mathbf{h} \\ &\mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

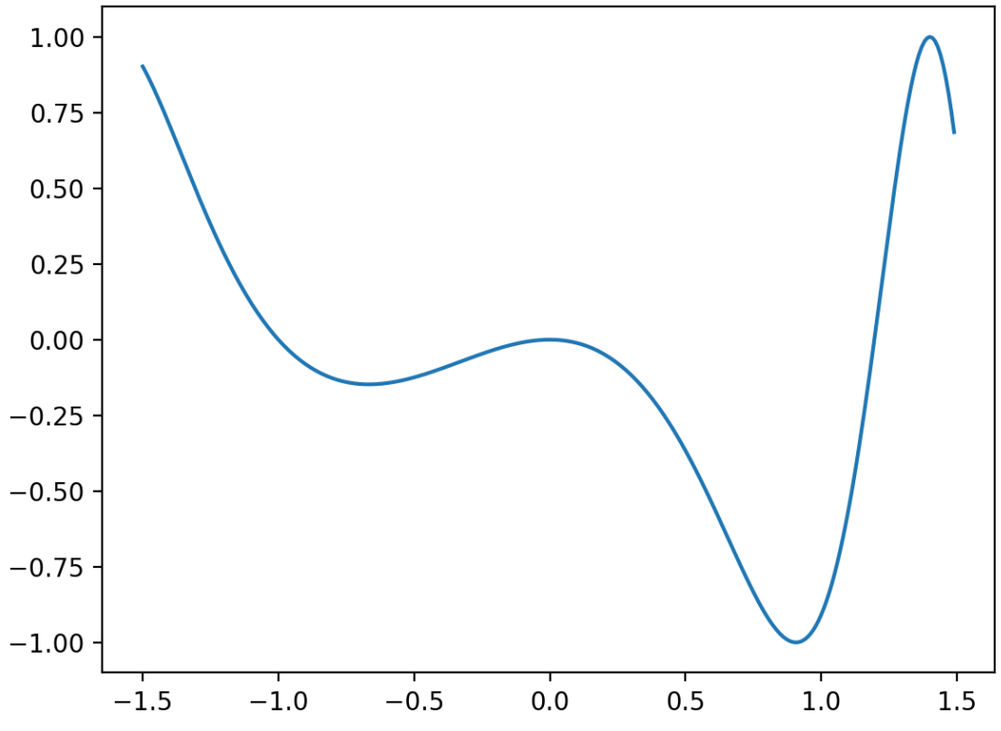
With P>0

Again easy to solve problem
(active set methods, interior-point methods)

Minimizing a function







Non convex functions?

$$\begin{aligned} & \min_x f(x) \\ \text{subject to} \ : \quad & h(x) = 0 \\ & g(x) \leq 0 \end{aligned}$$

where f(x), g(x) and h(x) are arbitrary

Necessary (but not necessarily sufficient) conditions for optimality (Karush–Kuhn–Tucker conditions - KKT conditions)

$$\frac{\partial L}{\partial x} = 0 \qquad \begin{array}{ll} h(x) = 0 & g(x) \leq 0 \\ \mu_i \geq 0 & \mu^T g(x) = 0 \end{array}$$

Numerical methods for unconstrained minimization

$$\min_x f(x)$$

Idea: start with a guess x and then find a "descent direction" Δx to update x such that makes f(x) decrease

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}$$

General descent method.

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

repeat

1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied. ← Typically when the gradient is close to 0

Line search methods

Exact line search: just find t that leads to the best update possible

$$t = \text{argmin}_{s \geq 0}\ f(x + s\Delta x)$$

Backtracking line search: start with a t and then make it smaller until we can reduce f(x)

Backtracking line search.

given a descent direction Δx for f at $x \in \textbf{dom}\, f$, $\alpha \in (0, 0.5)$, $\beta \in (0, 1)$.
$t := 1$.
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x, \quad t := \beta t.$

Gradient Descent

Gradient descent method.

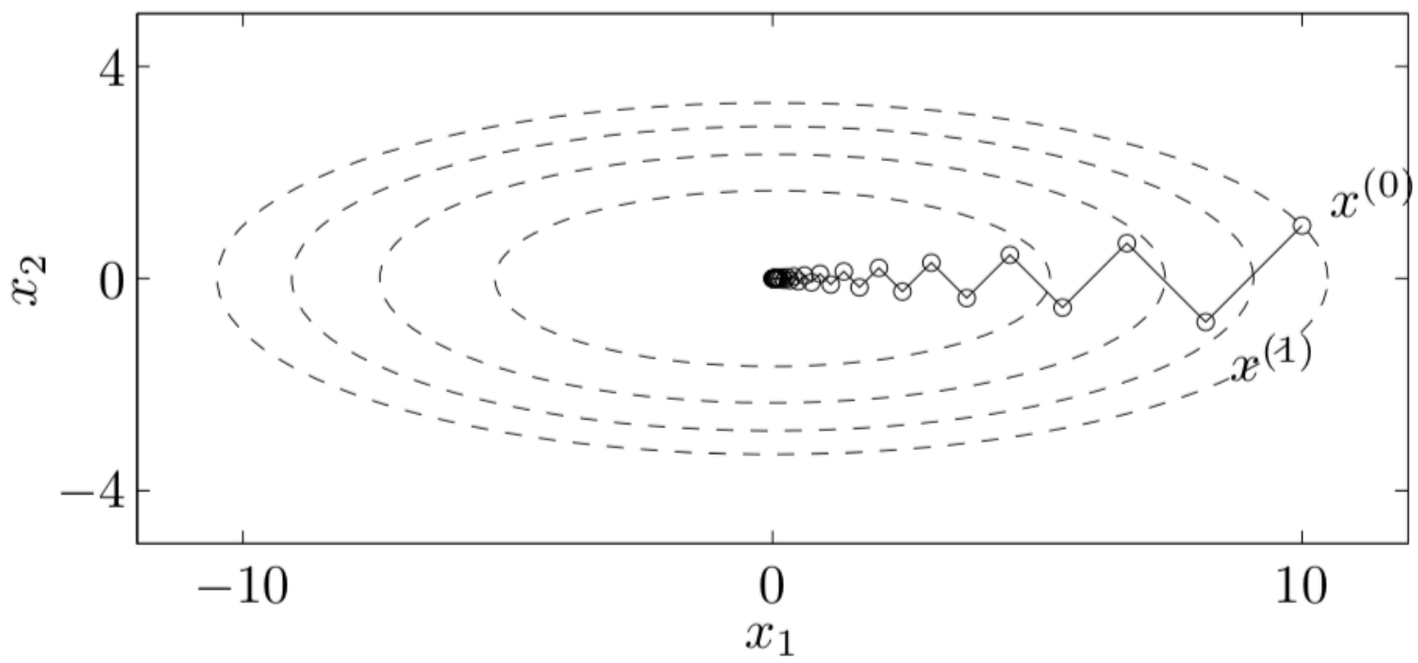
given a starting point $x \in \mathbf{dom}\, f$.

repeat

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

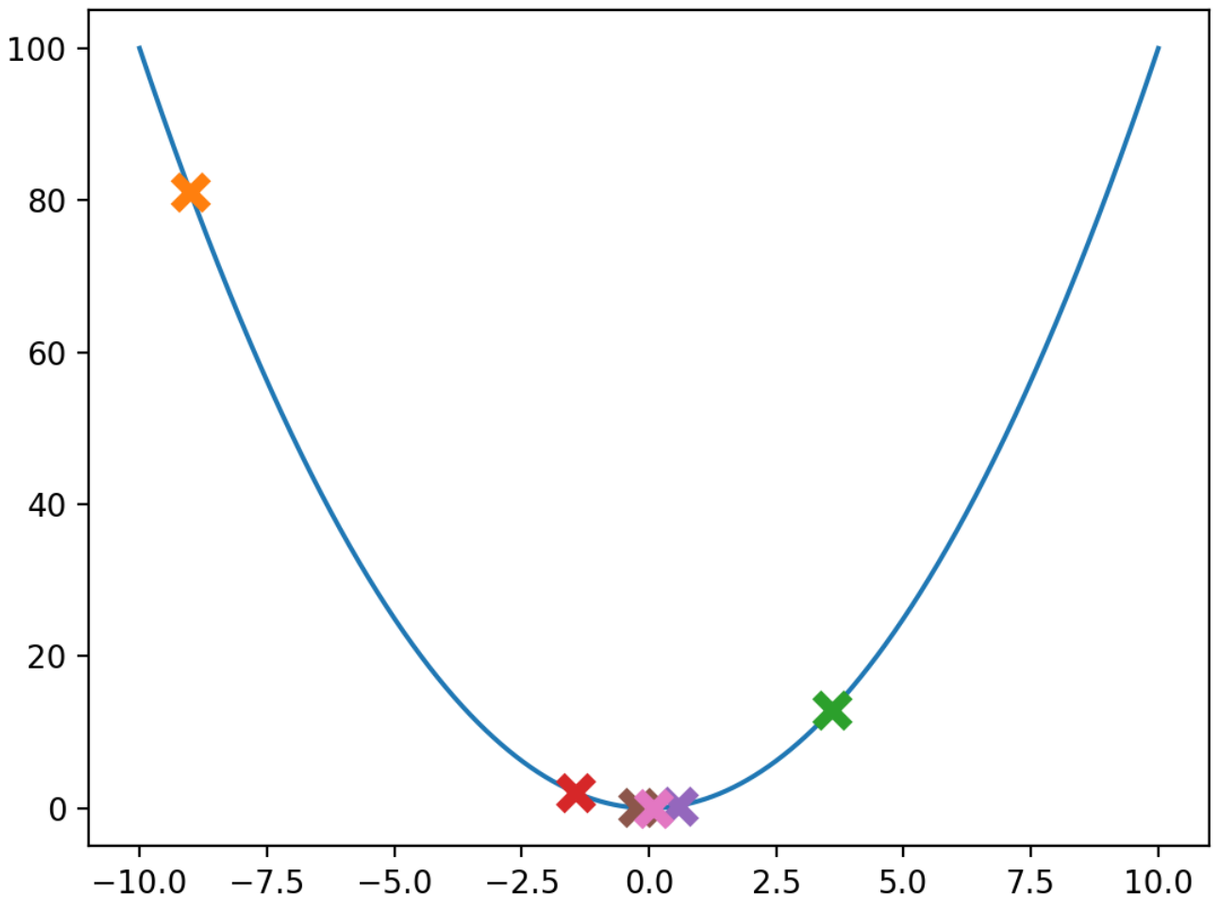
until stopping criterion is satisfied.

Convergence is linear $f(x^{(k)}) - p^\star \leq c^k(f(x^{(0)}) - p^\star)$



$$f(x) = (1/2)(x_1^2 + 10x_2^2)$$

Gradient Descent



$$\alpha = 0.7$$

$$\beta = 0.1$$

$$f(x) = x^2$$

Newton-Raphson Method

Newton's method.

given a starting point $x \in \mathbf{dom}\, f$, tolerance $\epsilon > 0$.

repeat

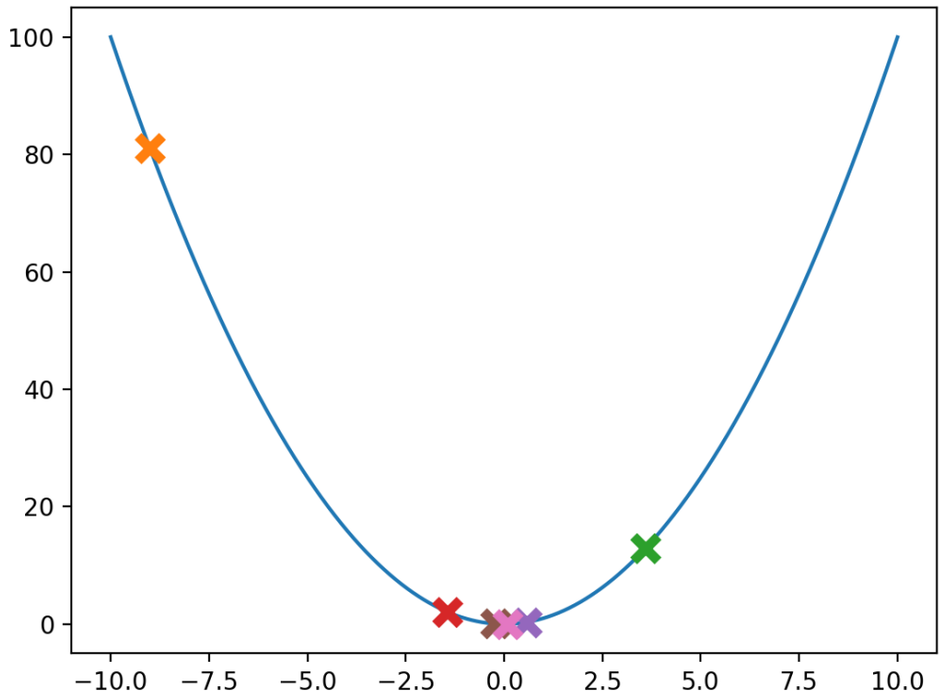
1. *Compute the Newton step and decrement.*
$$\Delta x_{\mathrm{nt}} := \mathsf{H}^{-1}\nabla f(x); \quad \lambda^2 := \nabla f(x)^T \mathsf{H}^{-1}\nabla f(x)$$
2. *Stopping criterion.* **quit** if $\lambda^2/2 \leq \epsilon$.
3. *Line search.* Choose step size t by backtracking line search.
4. *Update.* $x := x + t\Delta x_{\mathrm{nt}}$.

Convergence is quadratic near the minimum

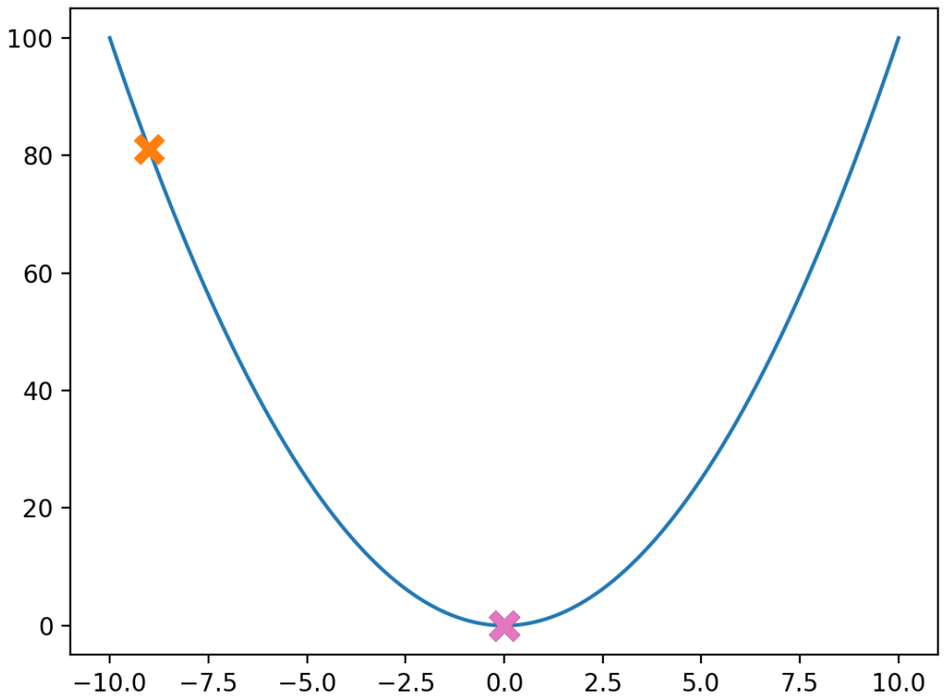
Problem is that computing the Hessian can be costly!

Quasi-Newton methods seek to approximate the Hessian computation: Gauss-Newton method, BFGS, etc

Gradient Descent



Newton Method

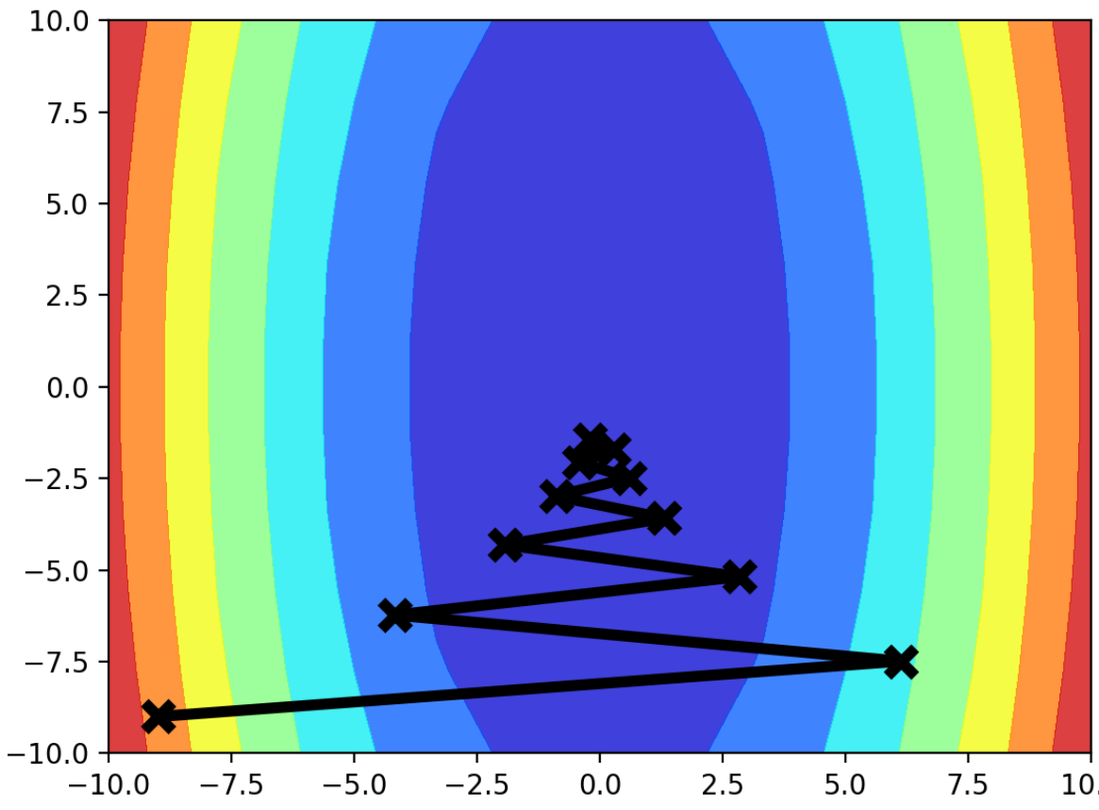


$$\alpha = 0.7$$

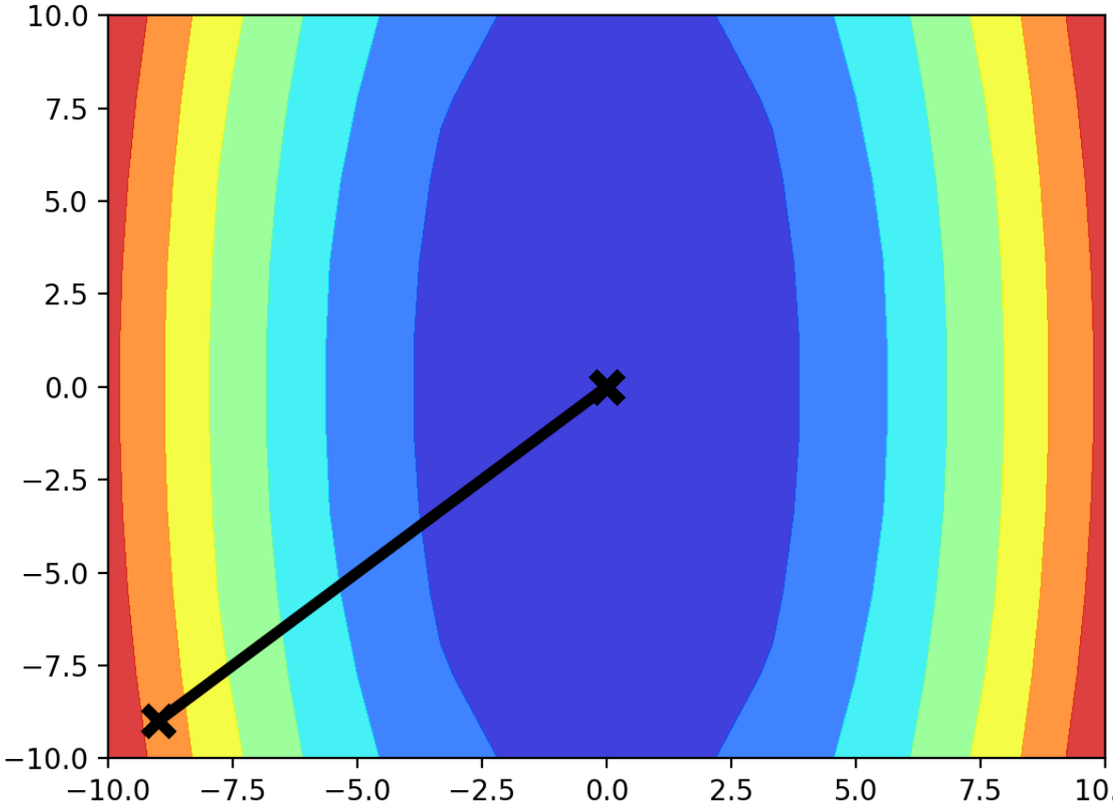
$$\beta = 0.1$$

$$f(x) = x^2$$

Gradient Descent



Newton Method



$$\alpha = 0.7$$

$$\beta = 0.1$$

$$f(x) = x^T \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} x$$

Summary

Convex optimization problems:

- When a local minimum is found => global minimum!
- KKT conditions are also sufficient conditions
- Linear and quadratic programs can be solved very efficiently
- Many out-of-the-box algorithms to solve convex optimization problems (e.g. Gurobi, CVXOpt, etc)

Non-convex optimization problems

- When a local minimum is found => no way to know if it is global
- KKT conditions can be used to find local minima (need a good first guess x0)
- Out-of-the-box algorithms can solve general optimization problems, only local minima can be found

Optimal control

Continuous time

$$\min_{\mathbf{u}(t)} \int_{t=0}^{t=T} J(\mathbf{x}(t), \mathbf{u}(t))$$

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

$$\mathbf{b}(\mathbf{x}, \mathbf{u}, t) \leq \mathbf{c}(t)$$

Discrete time

$$\min_{\mathbf{u}_n} \Sigma_{n=0}^{n=N} J(\mathbf{x}_n, \mathbf{u}_n)$$

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n)$$

$$\mathbf{b}_n(\mathbf{x}_n, \mathbf{u}_n, n) \leq \mathbf{c}_n$$

Finite horizon

Optimization over finite time

Infinite horizon

Optimization in the limit $t \rightarrow \infty$

Discrete time optimal control
Dynamic Programming

Markov Decision Process (MDP)

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, \boldsymbol{\omega}_n)$$

Markov property knowledge of state n is sufficient to predict n+1 => there is no need to remember previous states or actions

Note that $\mathbf{x}_n$ is now a random variable.
It is a normal vector if the noise ω_n is 0.

Deterministic example

We need to schedule 4 different operations A,B,C and D (e.g. during a manipulation task)

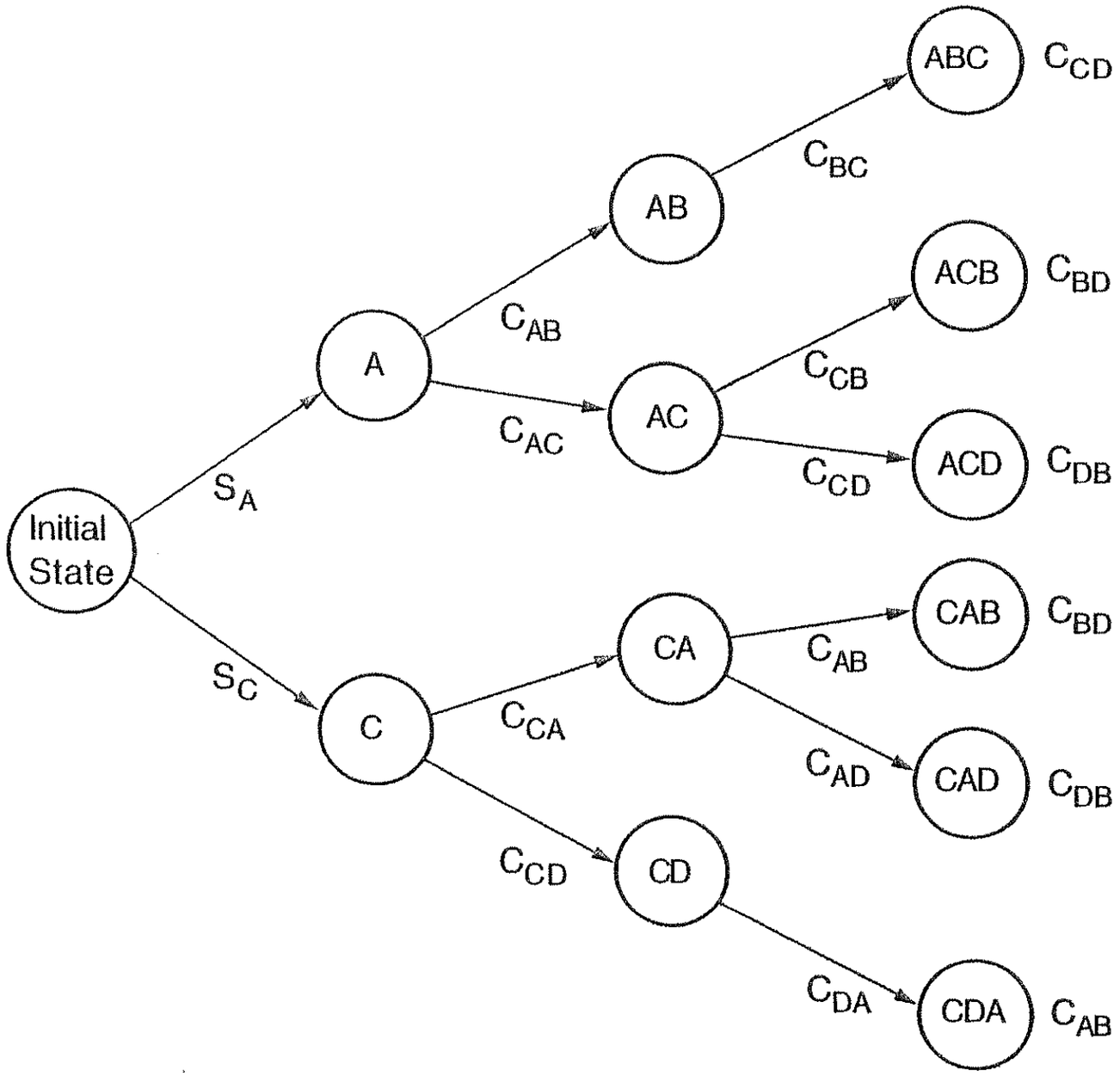
B can only be done after A
D can only be done after C

Cost of first action taken S_A or S_C

Cost to transition from one action to another Cij

Taken from [Bertsekas, 2005]

Deterministic example

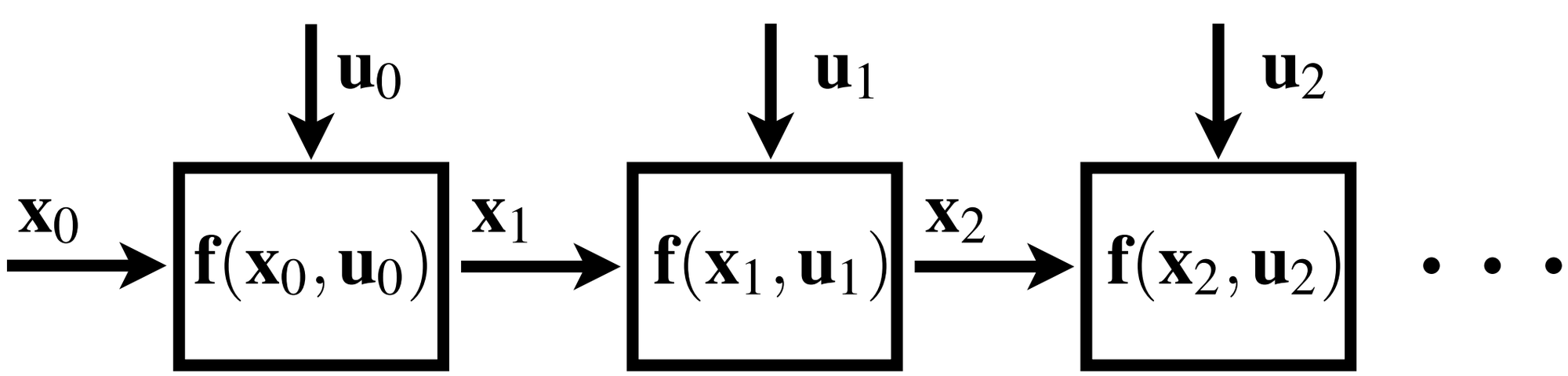


Taken from [Bertsekas, 2005]

Open-loop control

No disturbance $\mathbf{w}_k = 0$

We have a deterministic problem (planning problem)
It is sufficient to know the initial state to predict all future states and compute the optimal controls



Stochastic example: inventory problem

We are selling coffee and need to order a certain quantity of coffee to meet the demand while we don't want to keep too much stock because it is costly to store
How can we plan coffee orders for the next N weeks?

x_n Quantity of coffee at week n

u_n Amount of ordered coffee at week n

ω_n Client demand => random variable with know distribution

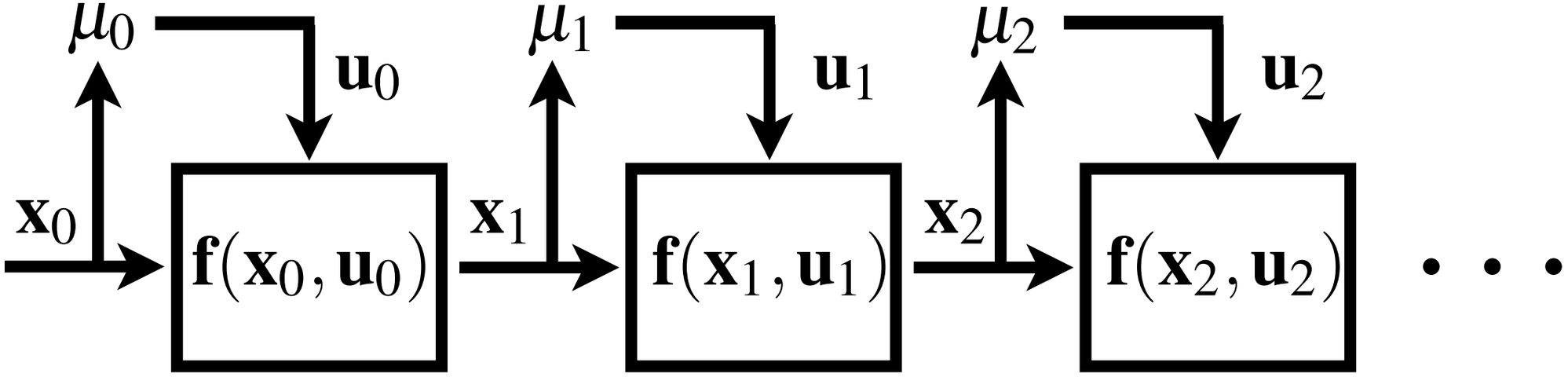
$$x_{n+1} = x_n + u_n - \omega_n \qquad J = \mathbb{E}\left(R(x_N) + \Sigma_{n=0}^{N-1} x_k^2 + cu_k\right)$$

Taken from [Bertsekas, 2005]

Closed-loop control

$\mathbf{w}_k \neq 0$

Decisions can only be made when stage k is attained. We need to measure the state at stage k and then use it to compute the optimal control. It is a closed loop system



The need for a policy

$\mu_k(\mathbf{x}_k) = \mathbf{u}_k$ maps a state into a control at stage k

$\pi = \{\mu_0, \cdots, \mu_{N-1}\}$ a policy or a control law

Expected cost of a policy

$$J_\pi(x_0) = E\left\{ g_N(x_n) + \sum_{k=0}^{N-1} g_k(x_k, \mu_n(x_n), w_k) \right\}$$

The optimal policy π^* $J_{\pi^*} = \min_\pi J_\pi(x_0)$

Discrete time optimal control

Additive cost function $$J = E\left\{ g_N(x_n) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\}$$

Discrete-time system $$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, \mathbf{u}_n, \boldsymbol{\omega}_n)$$

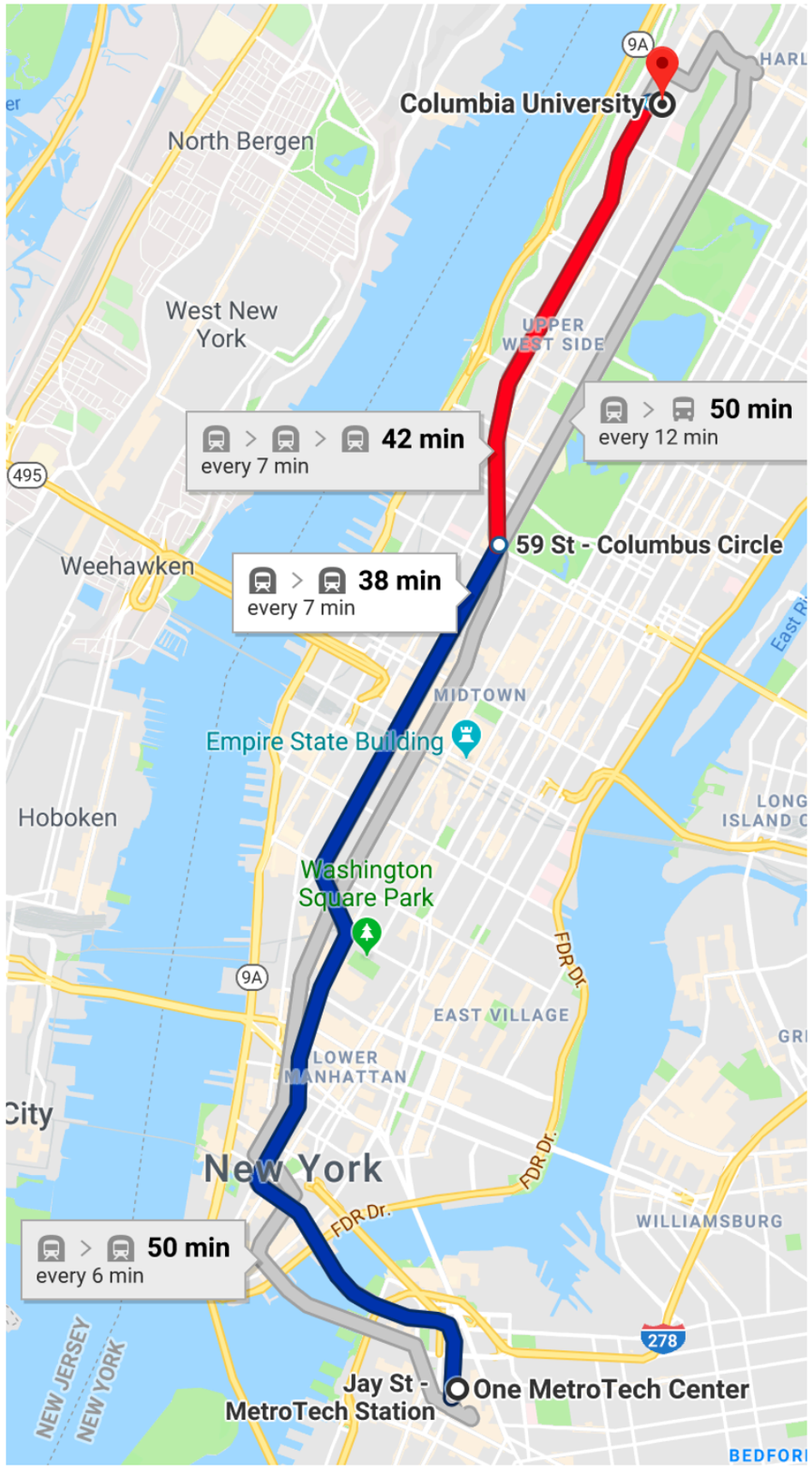
Random parameters $$w_n \sim P(\cdot \mid x_n, u_n)$$

Control or state constraints $$x_n \in S_{x_n} \qquad u_n \in S_{u_n}$$

Open-loop control is ok for deterministic systems (prepare a plan a then execute it)

Closed-loop control is needed for stochastic systems (adapt the behavior to current state of the system)

Bellman's Principle of optimality



If the fastest path from Tandon to Columbia is with the blue line with a change to the red line at Columbus Circle, then the fastest way from Columbus Circle to Columbia is with the red line as well

Subpath of optimal paths are also optimal for then own subproblem

Bellman's Principle of optimality

Proposition Let $\pi^* = \{\mu_0^*, \cdots, \mu_{N-1}^*\}$ be an optimal policy for the basic problem and assume that when using π^*, a given state x_i occurs at the time i with positive probability. Consider the subproblem whereby we are at x_i at time i and wish to minimize the "cost-to-go" from time i to time N

$$E\left\{g_N(x_n) + \sum_{k=i}^{N-1} g_k(x_k, u_k, w_k)\right\}$$

Then the truncated policy $\{\mu_i^*, \cdots, \mu_{N-1}^*\}$ is optimal for this subproblem. □

[Taken from Bertsekas, 2005]

Dynamic Programming

[Taken from Bertsekas, 2005]

Dynamic Programming

Proposition For every initial state x_0, the optimal cost $J^*(x_0)$ of the basic problem is equal to $J_0(x_0)$, given by the last step of the following algorithm, which proceeds backward in time from period $N-1$ to period 0:

$$J_N(x_N) = g_N(x_N) \tag{1}$$

$$J_n(x_n) = \min_{u_n} \underset{w_n}{E} \left\{ g_n(x_n, u_n, w_n) + J_{n+1}(f_n(x_n, u_n, w_n)) \right\}, \; n = 0, \cdots, N-1 \tag{2}$$

where the expectation is taken with respect to the probability distribution of w_n, which depends on x_n and u_n. Furthermore, if $u_n^* = \mu_n^*(x_n)$ minimizes the right side of Equation (2) for each x_n and n, the policy $\pi^* = \{\mu_0^*, \cdots, \mu_{N-1}^*\}$ is optimal. □

[Taken from Bertsekas, 2005]

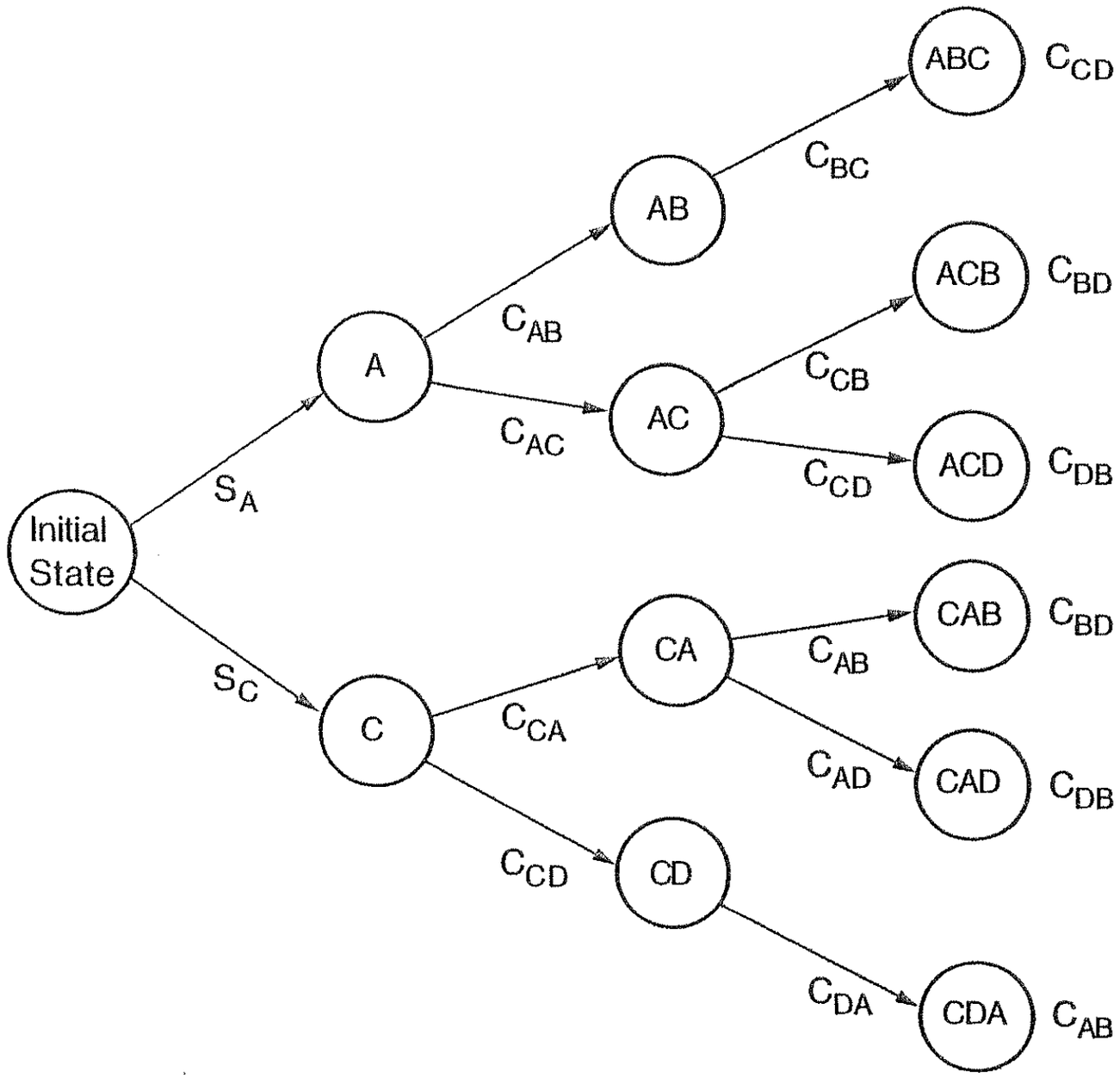
Dynamic Programming

Dynamic programing is the algorithm that finds the global optimal solution to discrete time optimal control problems.

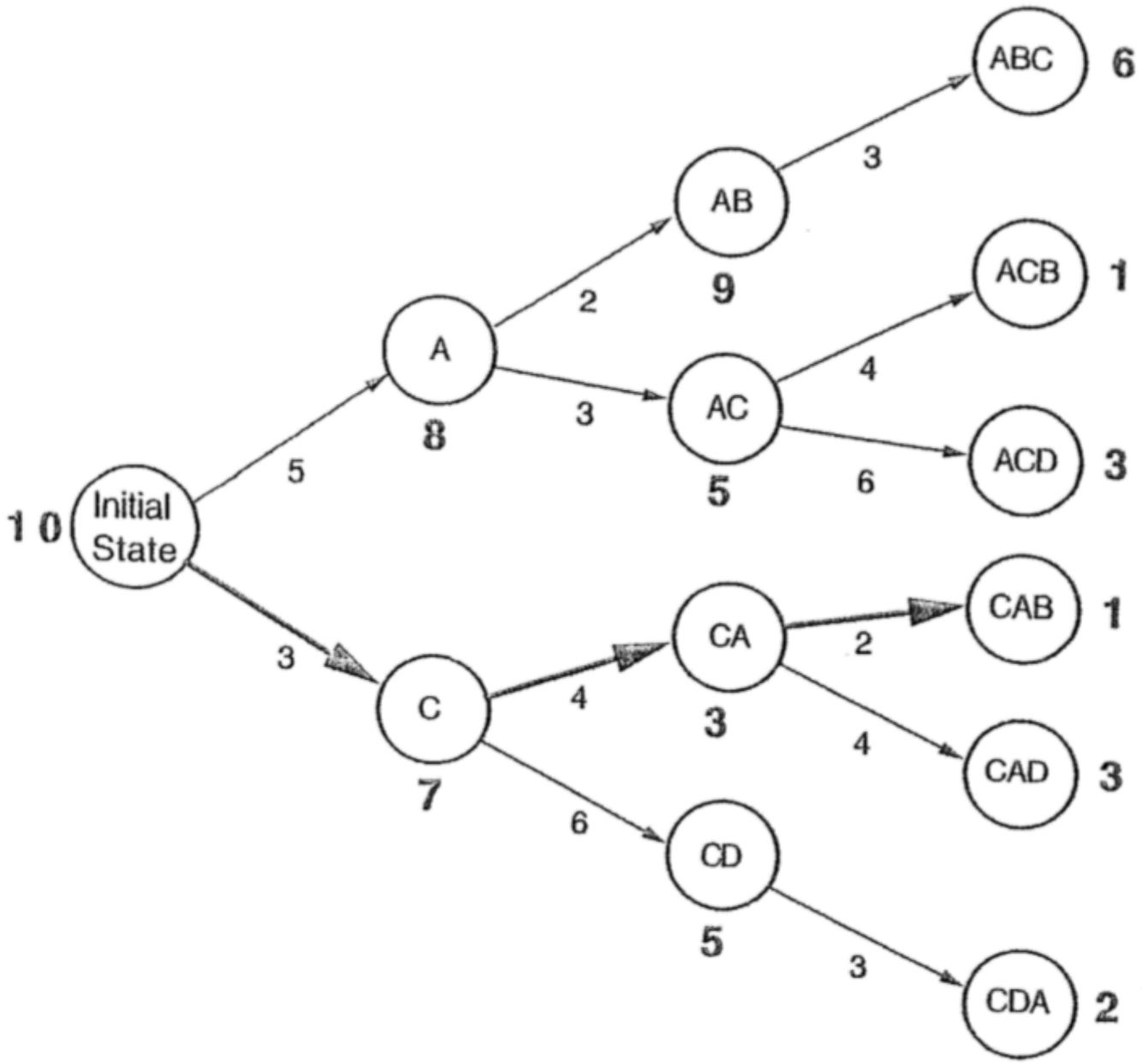
It was "invented" by Richard Bellman in 1953.

You already know dynamic programming:
Dijkstra's shortest path algorithm, Vitterbi algorithm, A* algorithms, etc... can be viewed as special cases of DP.

Dynamic Programming



Dynamic Programming



Example

$$\min_{u_k} \; x_N^2 + \sum_{k=0}^{N-1} \left(x_k^2 + u_k^2 \right) \qquad\qquad x_{k+1} = x_k + u_k$$

$$N = 2 \qquad x_0 = 3$$

Analytic solutions

Problems with quadratic costs and linear dynamics can be solved analytically.

They are called Linear-Quadratic problems and they are fundamental tools for the control of robots
(to be studied in 3 lectures from today)

In general, it is not possible to find analytic solutions for optimal control problems. We need to apply the DP algorithm numerically

Numerical solution

For stage N compute $J_N(x_N) = g_N(x_N)$ for every state

For k = N-1 to 0:

compute for every state x_n the cost to go

$$J_n(x_n) = \min_{u_n} \underset{w_n}{E} \{g_n(x_n, u_n, w_n) + J_{n+1}(f_n(x_n, u_n, w_n))\}$$

Complexity is generally polynomial in the number of states and controls

Number of possible policies is exponential in the number of states and stages

Curse of dimensionality

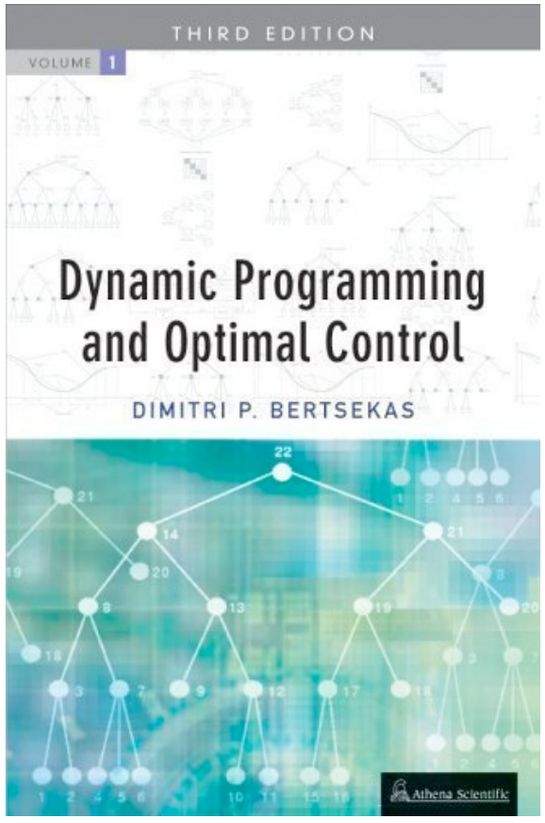
We often need to discretize the states and the controls

Quality of the solution will depend on the discretization (if the discretization is too rough, the solutions might not be useable)

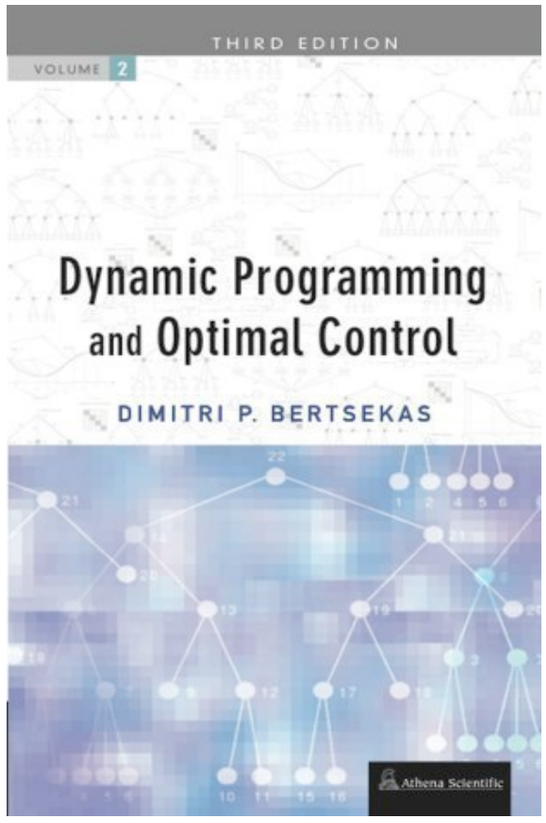
The number of states grows exponentially fast with respect to the number of dimensions!!

Infinite horizon problems

Recommended literature



Vol. 1 Finite horizon problems



Vol. 2 Infinite horizon problems

This lecture: vol. 1 (chapter 1 and 3) and vol. 2 (chapter 1)