Lec 12: Constrained Optimization

15-369/669/769: Numerical Computing

Instructor: Minchen Li

Table of Content

Table of Content

Motivation and Examples

Constrained Optimization: General Form

- **General problem:**

$$\min_{\mathbf{x}} f(\mathbf{x})$$
$$\text{s.t. } g(\mathbf{x}) = 0, \quad h(\mathbf{x}) \geq 0.$$

- $f : \mathbb{R}^n \to \mathbb{R}$ — objective function; $g : \mathbb{R}^n \to \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^p$ — constraints.
- Includes many special cases:
  - $f(\mathbf{x}) = h(\mathbf{x}) \equiv 0 \to$ root-finding.
  - $g(\mathbf{x}) = h(\mathbf{x}) \equiv 0 \to$ unconstrained optimization.
- Practical goals: Find feasible $\mathbf{x}$ improving f even if global optimality is hard.

Motivation and Examples

Applications and Examples

- Constrained optimization appears in nearly every applied field:
    - Engineering: equilibrium, dynamics.
    - AI: machine learning, computer vision.
    - Graphics: geometry processing, deformation.
- Example — eigenvalue problem:

$$\min_{\mathbf{x}} \mathbf{x}^\top A\mathbf{x} \quad \text{s.t. } \|\mathbf{x}\|_2 = 1.$$

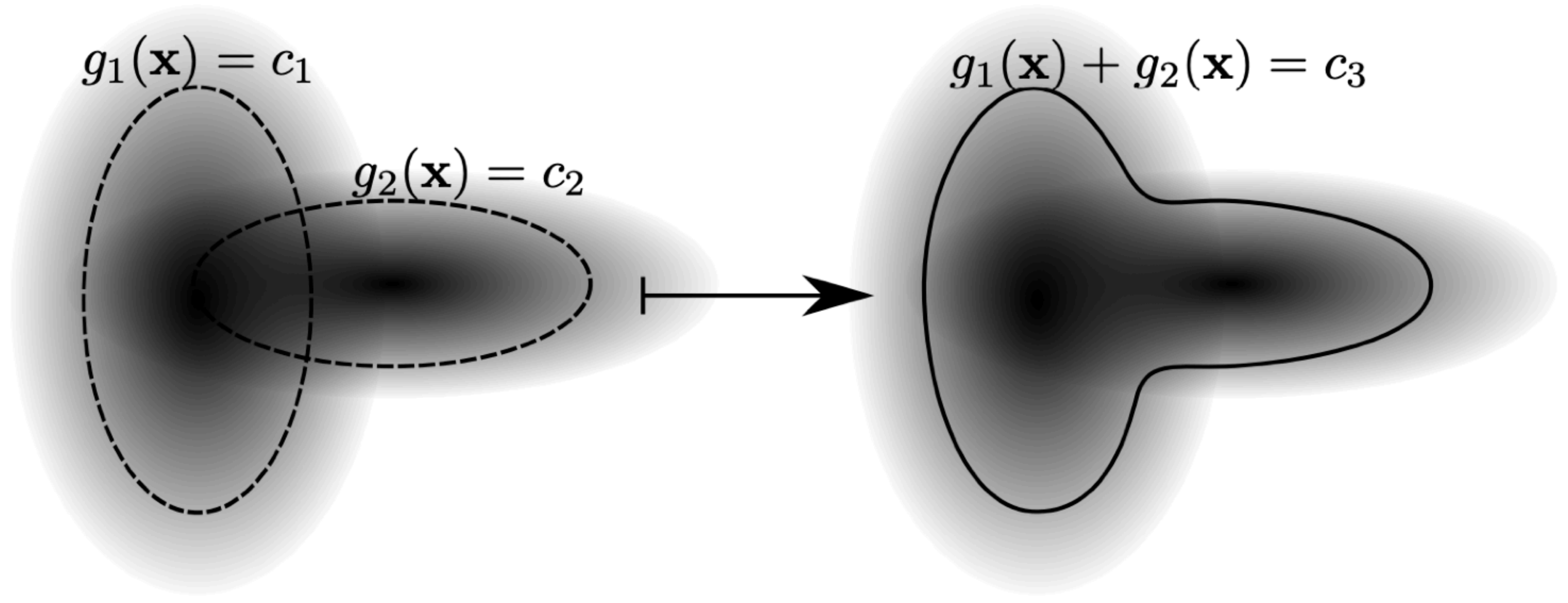
  Constrains the solution to the unit sphere.

- Other examples below illustrate different classes of constraints.

Motivation and Examples

Example 1: Geometric Projection

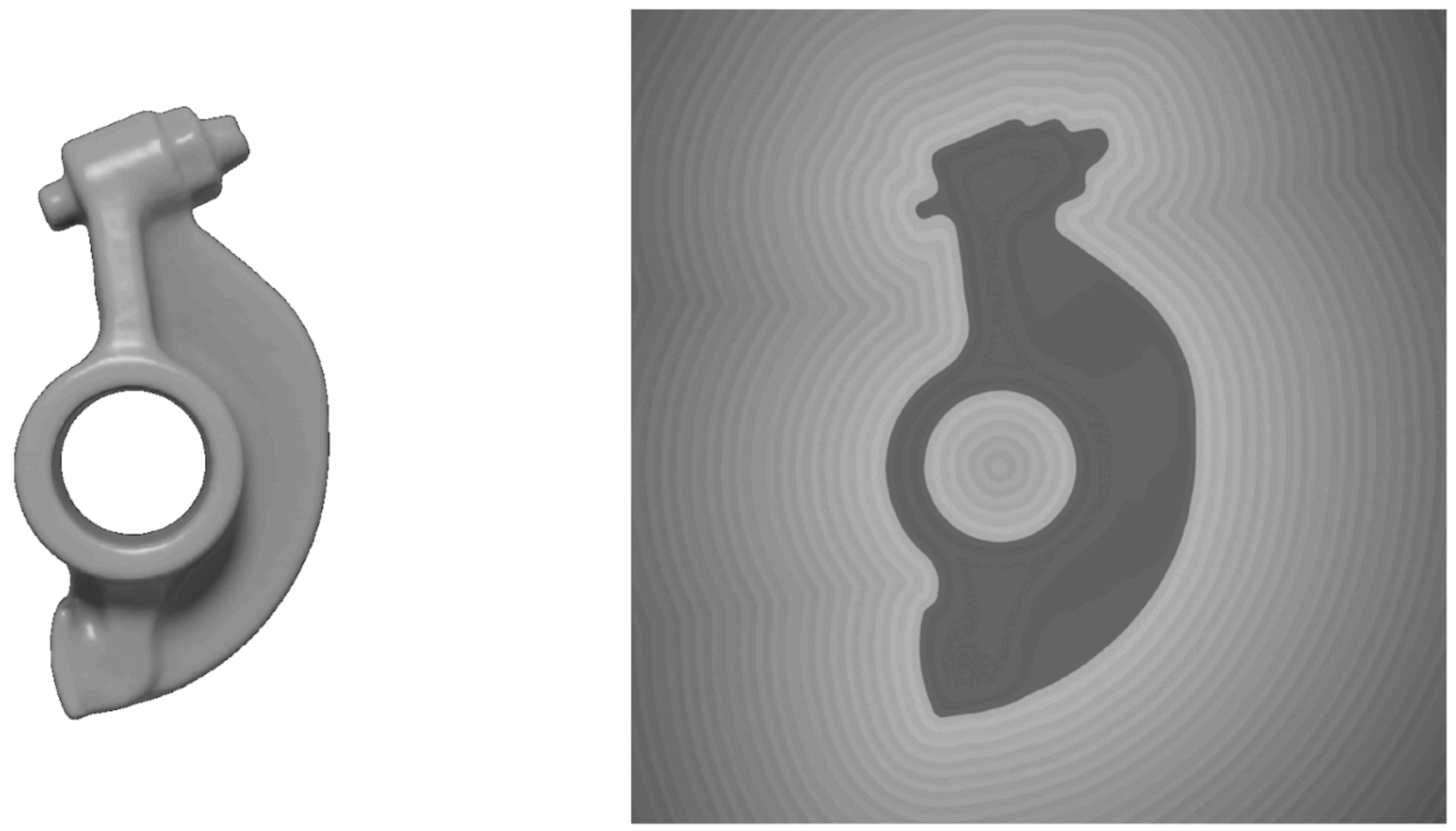
- Many shapes $S \subset \mathbb{R}^n$ can be represented implicitly as $g(\mathbf{x}) = 0$.
- Examples:
  - Sphere: $g(\mathbf{x}) = \|\mathbf{x}\|_2^2 - 1$.
  - Cube: $g(\mathbf{x}) = \|\mathbf{x}\|_1 - 1$.
  - Blobby shapes: $g(\mathbf{x}) = c + \sum_i a_i e^{-b_i\|\mathbf{x}-\mathbf{x}_i\|_2^2}$.



- To project a point $\mathbf{y}$ onto S:

$$\min_{\mathbf{x}} \|\mathbf{x} - \mathbf{y}\|_2 \quad \text{s.t. } g(\mathbf{x}) = 0.$$

- Modern use: neural implicit representations of 3D shapes.



3D model   Neural signed distance function

Motivation and Examples

Example 2: Manufacturing (Linear Programming)

- **Goal:** Maximize total profit by deciding production plan subject to resource constraints:.

$$\max_{\mathbf{x}} \sum_{j=1}^{k} p_j x_j \quad \text{s.t.} \quad x_j \geq 0, \quad \sum_{j=1}^{k} c_{ij} x_j \leq s_i, \ \forall i.$$

  - $\mathbf{x} = (x_1, \ldots, x_k)$: number of units to produce for each product.
  - p_j: profit per unit of product j.
  - c_{ij}: amount of resource i consumed by one unit of product j.
  - s_i: total available amount of resource i.
- This is a **linear program (LP)** with linear objective and constraints.
- **Applications:** Resource allocation, supply chain optimization, and factory scheduling.

Motivation and Examples

Example 3: Bundle Adjustment (Computer Vision)

- Reconstruct 3D points $\mathbf{y}_j$ and camera matrices P_i from 2D observations $\mathbf{x}_{ij}$:

$$\min_{\mathbf{y}_j, P_i} \sum_{i,j} \|P_i \mathbf{y}_j - \mathbf{x}_{ij}\|_2^2 \quad \text{s.t. } P_i \in \mathcal{S} \; \forall i,$$

  where $\mathcal{S}$ denotes valid projection matrices.

- Applications:
  - 3D reconstruction and SLAM.
  - Camera calibration and motion recovery.

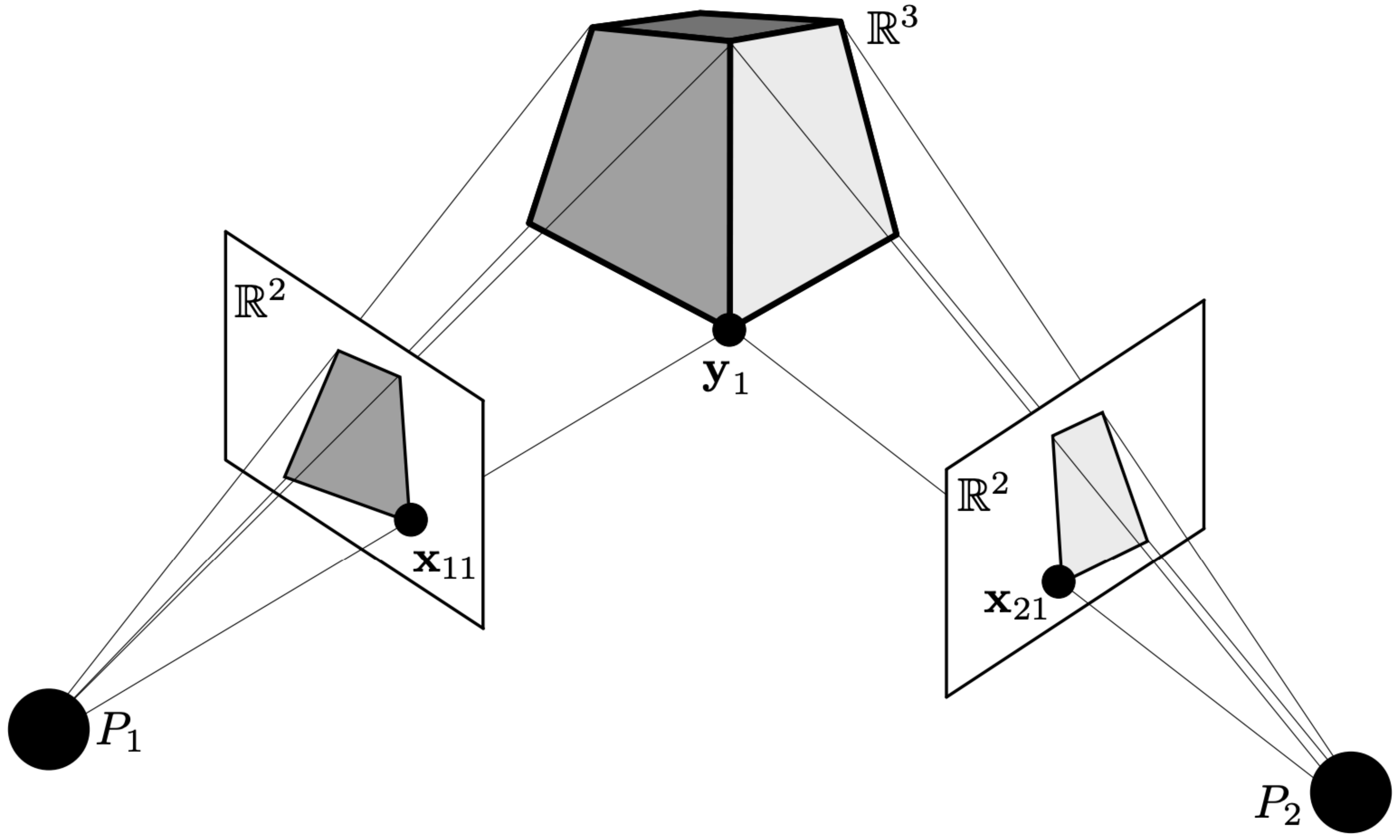


Table of Content

- Motivation and Examples
- Optimality Conditions
- Optimization Algorithms
- Convex Programming Problems

Optimality Conditions

Lagrange Multipliers (Dual Variables)

- We assume f, g, h are differentiable.
- The general constrained optimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t. } g(\mathbf{x}) = 0,\ h(\mathbf{x}) \geq 0.$$

- **Lagrange multipliers:** ignoring $h(\mathbf{x})$ for now, we can introduce auxiliary variables λ to convert the equality constraints into an unconstrained problem.

$$\Lambda(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda \cdot g(\mathbf{x})$$

  Critical points of f subject to $g(\mathbf{x}) = 0$ are given by stationary points of Λ w.r.t. both $\mathbf{x}$ and λ.

- But how to deal with the inequality constraints?

Optimality Conditions

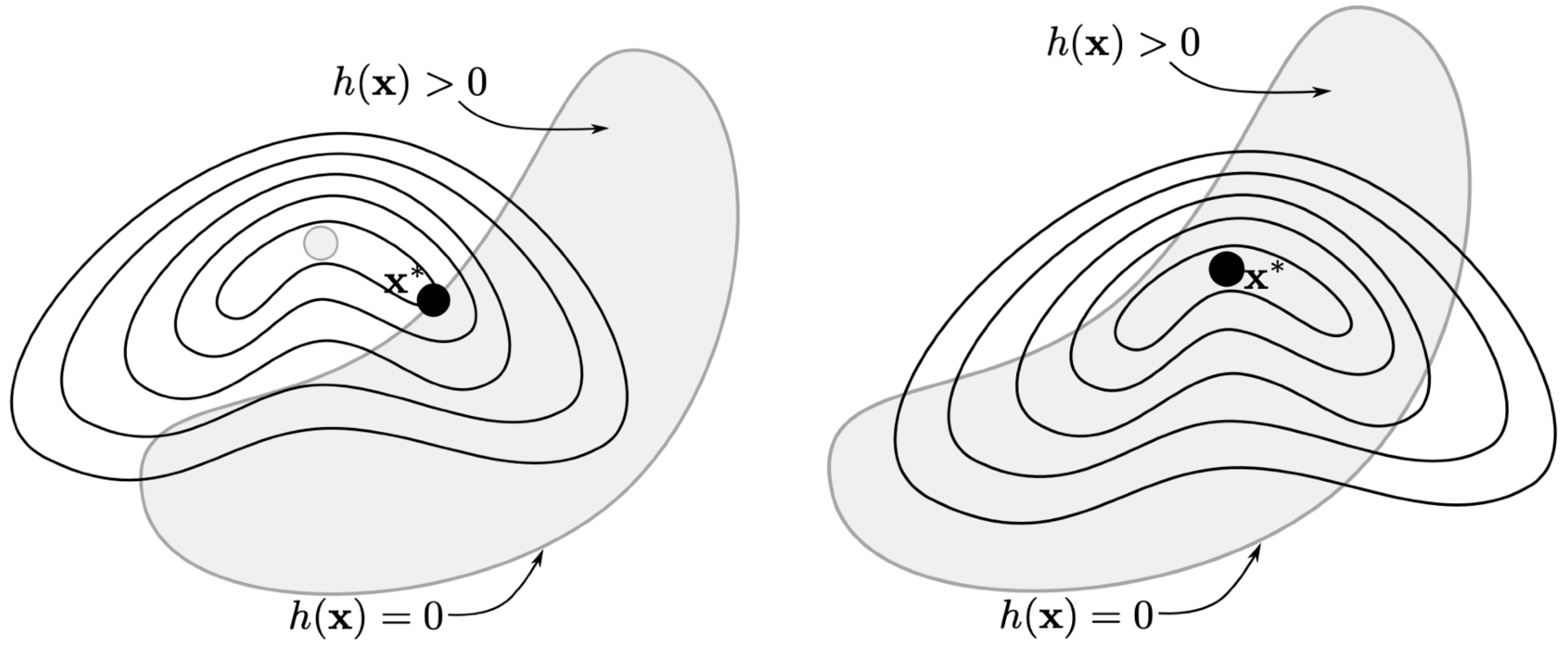
Feasibility and Critical Points

- **Definition 10.1 (Feasible point and feasible set)**: A feasible point $\mathbf{x}$ satisfies $g(\mathbf{x}) = 0$ and $h(\mathbf{x}) \geq 0$. The feasible set is the set of all such $\mathbf{x}$.
- **Definition 10.2 (Critical point of constrained optimization)** A critical point satisfies the constraints and is a local minimum, maximum, or saddle point of f within the feasible set.
- Finding a feasible $\mathbf{x}$ can already be difficult before optimizing f.
- Equality-constrained critical points can be found via the Lagrangian $\Lambda(\mathbf{x}, \lambda)$.

Optimality Conditions

KKT Conditions: Motivation

- Constrained problems combine:
  - Root-finding ($g(\mathbf{x}) = 0$),
  - Feasibility ($h(\mathbf{x}) \geq 0$),
  - Minimization ($f(\mathbf{x})$).



Active constraint
$h(\mathbf{x}^*) = 0$

Inactive constraint
$h(\mathbf{x}^*) > 0$

- Active constraint: $h_j(\mathbf{x}^*) = 0$; Inactive constraint: $h_j(\mathbf{x}^*) > 0$.
- Equality constraints are always active, while inequality constraints can be active or inactive at optimality, which we do not know a priori.
- If all inequality constraints are active, optimality can be found by finding the critical point of:

$$\Lambda(\mathbf{x}, \lambda, \mu) = f(\mathbf{x}) - \lambda \cdot g(\mathbf{x}) - \mu \cdot h(\mathbf{x}).$$

Optimality Conditions

Complementary Slackness and Dual Feasibility

- Add complementary slackness condition to allow inactive inequality constraints:

$$\mu_j h_j(\mathbf{x}^*) = 0.$$

- This ensures either:
    - $h_j(\mathbf{x}^*) = 0 \Rightarrow$ constraint is active,
    - or $\mu_j = 0 \Rightarrow$ constraint is inactive, and thus ignored in $\Lambda(\mathbf{x}, \lambda, \mu)$.

- Recall $\nabla h_j(\mathbf{x}^*)$ points in the direction of steepest increase of $h_j(\cdot)$ at $\mathbf{x}^*$, so infinitesimal displacements δ from $\mathbf{x}^*$ move into the feasible set when $\nabla h_j(\mathbf{x}^*) \cdot \delta > 0$.

- So for $\mathbf{x}^*$ to be optimal, any δ moving into the feasible set should increase f locally ($\nabla f(\mathbf{x}^*) \cdot \delta > 0$).

- Then from $0 = \nabla_{\mathbf{x}}(\mathbf{x}^*, \lambda, \mu) \Rightarrow \nabla f(\mathbf{x}^*) = \mu_j \nabla h_j(\mathbf{x}^*)$, we have $\mu_j > 0$ — dual feasibility.

Optimality Conditions

Karush–Kuhn–Tucker (KKT) Conditions

- **Theorem 10.1**: Under suitable regularity conditions, a local optimum $\mathbf{x}^*$ for

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t. } g(\mathbf{x}) = 0,\; h(\mathbf{x}) \geq 0$$

  satisfies the existence of multipliers $\lambda \in \mathbb{R}^m,\; \mu \in \mathbb{R}^p$ such that:

$$\begin{cases} 0 = \nabla f(\mathbf{x}^*) - \sum_i \lambda_i \nabla g_i(\mathbf{x}^*) - \sum_j \mu_j \nabla h_j(\mathbf{x}^*) & \text{(stationarity)} \\ g(\mathbf{x}^*) = 0,\; h(\mathbf{x}^*) \geq 0 & \text{(primal feasibility)} \\ \mu_j h_j(\mathbf{x}^*) = 0,\; \forall j & \text{(complementary slackness)} \\ \mu_j \geq 0,\; \forall j & \text{(dual feasibility)} \end{cases}$$

- When h is absent, this reduces to the standard Lagrange multiplier condition.

Optimality Conditions

Example: Applying KKT Conditions

- **Problem:** $\max_{x,y} \; xy \quad \text{s.t. } x + y^2 \leq 2, \quad x, y \geq 0.$
- Here $f(x, y) = -xy$ (converted to minimization), with constraints:
  $h_1(x, y) = 2 - x - y^2, \quad h_2(x, y) = x, \quad h_3(x, y) = y.$
- **KKT Conditions:**

$$
\begin{aligned}
0 &= -y + \mu_1 - \mu_2 && \text{(stationarity in } x\text{)},\\
0 &= -x + 2\mu_1 y - \mu_3 && \text{(stationarity in } y\text{)},\\
x + y^2 \leq 2, &\quad x, y \geq 0 && \text{(primal feasibility)},\\
\mu_1(2 - x - y^2) = 0, &\quad \mu_2 x = 0, \quad \mu_3 y = 0 && \text{(complementary slackness)},\\
\mu_1, \mu_2, \mu_3 &\geq 0 && \text{(dual feasibility)}.
\end{aligned}
$$

- This system characterizes all optimal candidates under KKT.

Optimality Conditions

2nd-Order Conditions

- KKT is only a set of 1st-order conditions — $\mathbf{x}^*$ satisfying the KKT conditions can be local minimum, maximum, or saddle points.

- Similar to unconstrained optimization, one can further perform 2nd-order checks on

$$\nabla^2_{\mathbf{x}} \Lambda(\mathbf{x}^*, \lambda^*, \mu^*) = \nabla^2 f(\mathbf{x}^*) - \sum_i \mu_i^* \cdot \nabla^2 h_i(\mathbf{x}^*) - \sum_j \lambda_j^* \nabla^2 g_j(\mathbf{x}^*)$$

  to classify the critical point.

- If $\nabla^2_{\mathbf{x}} \Lambda(\mathbf{x}^*, \lambda^*, \mu^*)$ is SPD/SND/Indefinite, $\mathbf{x}^*$ is a local minimum/maximum/saddle point.

- But in practice, this is often not needed.

Table of Content

Optimization Algorithms
Overview

- Many robust methods for constrained optimization exist in modern software libraries.
- We can treat these as “clients” instead of implementing them from scratch.
- Nonetheless, understanding common strategies helps interpret solver behavior.
- Two major families:
  - **Sequential Quadratic Programming (SQP)**
  - **Barrier Methods**

Optimization Algorithms

Sequential Quadratic Programming (SQP)

- SQP approximates nonlinear constrained problems by simpler **quadratic subproblems**.
- Given a current guess $\mathbf{x}_k$, use a Taylor expansion:

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \arg\min_{\mathbf{d}} \left[\frac{1}{2}\mathbf{d}^\top H_f(\mathbf{x}_k)\mathbf{d} + \nabla f(\mathbf{x}_k)^\top \mathbf{d} + f(\mathbf{x}_k)\right]$$

  subject to:

$$g_i(\mathbf{x}_k) + \nabla g_i(\mathbf{x}_k)^\top \mathbf{d} = 0, \quad h_i(\mathbf{x}_k) + \nabla h_i(\mathbf{x}_k)^\top \mathbf{d} \geq 0.$$

- Each step solves a **quadratic program (QP)**: quadratic objective, linear constraints.
- Works best near a good initial point $\mathbf{x}_0$; far from optimum, may fail.
- Uses second-order model for f, first-order for g, h.

Optimization Algorithms

Equality-Constrained SQP

- When only equality constraints are present, define:

$$\Lambda(\mathbf{d}, \boldsymbol{\lambda}) = \frac{1}{2}\mathbf{d}^\top H_f(\mathbf{x}_k)\mathbf{d} + \nabla f(\mathbf{x}_k)^\top \mathbf{d} + \boldsymbol{\lambda}^\top (g(\mathbf{x}_k) + Dg(\mathbf{x}_k)\mathbf{d}) \,.$$

- Setting derivative to zero gives: $0 = H_f(\mathbf{x}_k)\mathbf{d} + \nabla f(\mathbf{x}_k) + [Dg(\mathbf{x}_k)]^\top \boldsymbol{\lambda}$.
- Combined system for $\mathbf{d}$ and $\boldsymbol{\lambda}$: $\begin{pmatrix} H_f(\mathbf{x}_k) & [Dg(\mathbf{x}_k)]^\top \\ Dg(\mathbf{x}_k) & 0 \end{pmatrix} \begin{pmatrix} \mathbf{d} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} -\nabla f(\mathbf{x}_k) \\ -g(\mathbf{x}_k) \end{pmatrix} .$
- Each iteration solves this linear system to obtain $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{d}$.
- Extensions (e.g., BFGS) can approximate H_f to avoid inverting large matrices.

Optimization Algorithms

Inequality Constraints and Active-Set Methods

- SQP for inequalities uses quadratic programs (QP) with linearized inequality constraints.
- To solve inequality-constrained QP, an **active-set strategy** maintains constraints currently "active" at the estimated solution, and so equality-constrained QP solvers can be applied.
- Violated constraints are added, and h_j with $\nabla f \cdot \nabla h_j \leq 0$ are removed dynamically.

```
// General SQP method:
While not converged:
   Quadratically approximate f, linearize g and h
   // solve general QP:
   For k = 1, 2, …: // can be inexact, e.g. only run 1 iteration
      Update active set
      Solve equality-constrained QP
```

Optimization Algorithms

Barrier (Penalty) Methods for Equality Constraints

- Replace constraints with penalty terms in the objective:

$$f_\rho(\mathbf{x}) = f(\mathbf{x}) + \rho \|g(\mathbf{x})\|_2^2.$$

- As $\rho \to \infty$, violations of $g(\mathbf{x}) = 0$ are penalized, forcing feasibility.
- **Barrier method:** iteratively solve unconstrained problems for increasing ρ.
- Steps:
    1. Optimize f_ρ as unconstrained problem.
    2. Check feasibility tolerance.
    3. If constraints not satisfied, increase ρ and repeat.
- Pros: simple to implement; Cons: as ρ increases, Hessian becomes ill-conditioned.

Optimization Algorithms

Barrier Methods for Inequality Constraints

- For inequality constraints $h_i(\mathbf{x}) \geq 0$, add a barrier term preventing infeasibility:

$$f_{\text{inv}}(\mathbf{x}) = f(\mathbf{x}) + \rho \sum_i \frac{1}{h_i(\mathbf{x})} \qquad \text{(inverse barrier)},$$

$$f_{\text{log}}(\mathbf{x}) = f(\mathbf{x}) - \rho \sum_i \log h_i(\mathbf{x}) \qquad \text{(logarithmic barrier)}.$$

- Barrier terms go to $+\infty$ as $h_i(\mathbf{x}) \to 0$, keeping iterates feasible.

- When solving the "unconstrained" proxy problem using gradient-based methods, needs **filtered** line search to avoid infeasibility.

- Accuracy increases as $\rho \to 0$, but the problem becomes worse-conditioned.

Optimization Algorithms

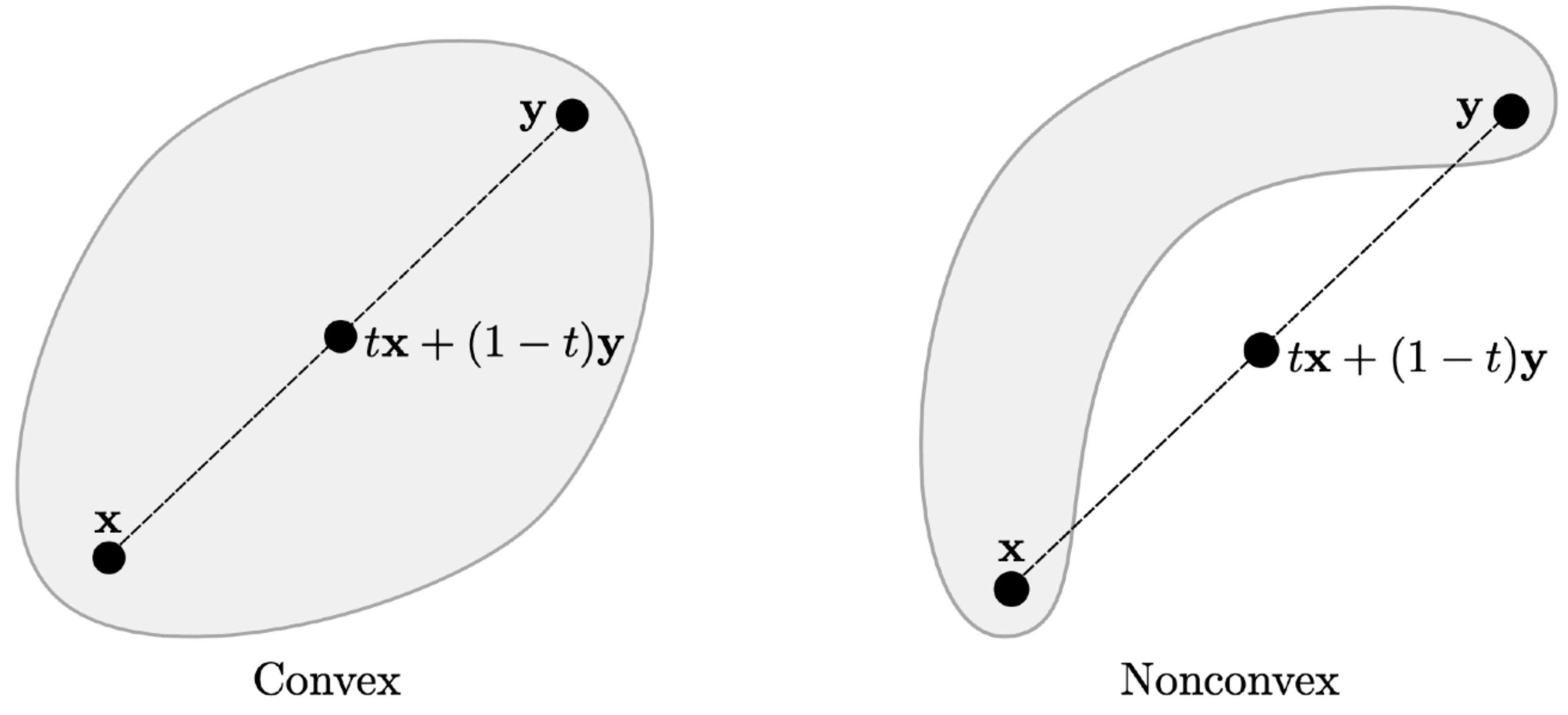
Comparison Between SQP and Barrier Methods

- SQP
  - [-] No convergence guarantee (line search cannot be applied).
- Barrier Methods
  - [+] Guarantees convergence with line search.
  - [-] Can become ill-conditioned when requesting high accuracy.
  - [-] Needs to maintain feasibility for inequality constraints (when diverging barriers are applied).

Table of Content

Convex Programming Problems

Convex Programming



- **Convex programming** guarantees a unique global minimum if both the objective f and the feasible set are convex.

- **Definition 10.3 (Convex set):** A set $S \subseteq \mathbb{R}^n$ is convex if for any $\mathbf{x}, \mathbf{y} \in S$,

$$t\mathbf{x} + (1-t)\mathbf{y} \in S \quad \forall t \in [0, 1].$$

- **Example 10.9:** The disc $\{\mathbf{x} : \|\mathbf{x}\|_2 \leq 1\}$ is convex, but the circle $\{\mathbf{x} : \|\mathbf{x}\|_2 = 1\}$ is not.

- **Key theorem:** A convex function cannot have suboptimal local minima, even on a convex domain. Thus, convex optimization guarantees convergence to a **global minimum**.
  - If a convex function had two local minima, all points between them would have smaller or equal objective values – contradicting local minimality.

Convex Programming Problems

Convexity

- Always check convexity – it greatly improves robustness and interpretability.
- **Disciplined Convex Programming (DCP):**
  - Provides compositional rules for combining convex objectives and constraints.
  - Ensures the resulting problem remains convex.
- Example rules:
  - Intersection of convex sets is convex.
  - Sum of convex functions is convex.
  - $h(\mathbf{x}) = \max\{f(\mathbf{x}), g(\mathbf{x})\}$ is convex if f, g are convex.
  - Sublevel set $\{\mathbf{x} : f(\mathbf{x}) \leq c\}$ is convex if f is convex.

Convex Programming Problems

Convex Programming Applications

- **Nonnegative least squares:**

$$\min_{\mathbf{x} \geq 0} \|A\mathbf{x} - \mathbf{b}\|_2^2$$

  Both objective and feasible set are convex.

- **Linear programs:** Linear objectives + linear constraints $\Rightarrow$ convex.

- **Including $\|\mathbf{x}\|_1$ in convex objectives:**
  - Introduce auxiliary variable $\mathbf{y}$ s.t. $y_i \geq x_i$, $y_i \geq -x_i$.
  - Then $\|\mathbf{x}\|_1 = \sum_i y_i$ at optimum.

- Convex libraries like CVX automatically handle such substitutions.

Convex Programming Problems

More Examples

- **Second-Order Cone Programming (SOCP)**

$$\text{minimize}_{\mathbf{x}} \ \mathbf{b} \cdot \mathbf{x}$$
$$\text{subject to } \|A_i\mathbf{x} - \mathbf{b}_i\|_2 \leq d_i + \mathbf{c}_i \cdot \mathbf{x} \text{ for all } i = 1, \ldots, k.$$

- **Semidefinite Programming (SDP)**
    - Matrix variable, with semi-definiteness constraints.

- *__Integer programming__ problems can be relaxed to convex programming with continuous variables, which can be more conveniently solved to construct the original solution.

Table of Content

- Motivation and Examples
- Optimality Conditions
- Optimization Algorithms
- Convex Programming Problems