AIMS: Optimization

Paul Goulart

Control Group

Department of Engineering Science

University of Oxford

Day 1: Optimization and Convexity

References

Convex Optimization & Duality Theory:

Boyd & Vandenberghe (2004)

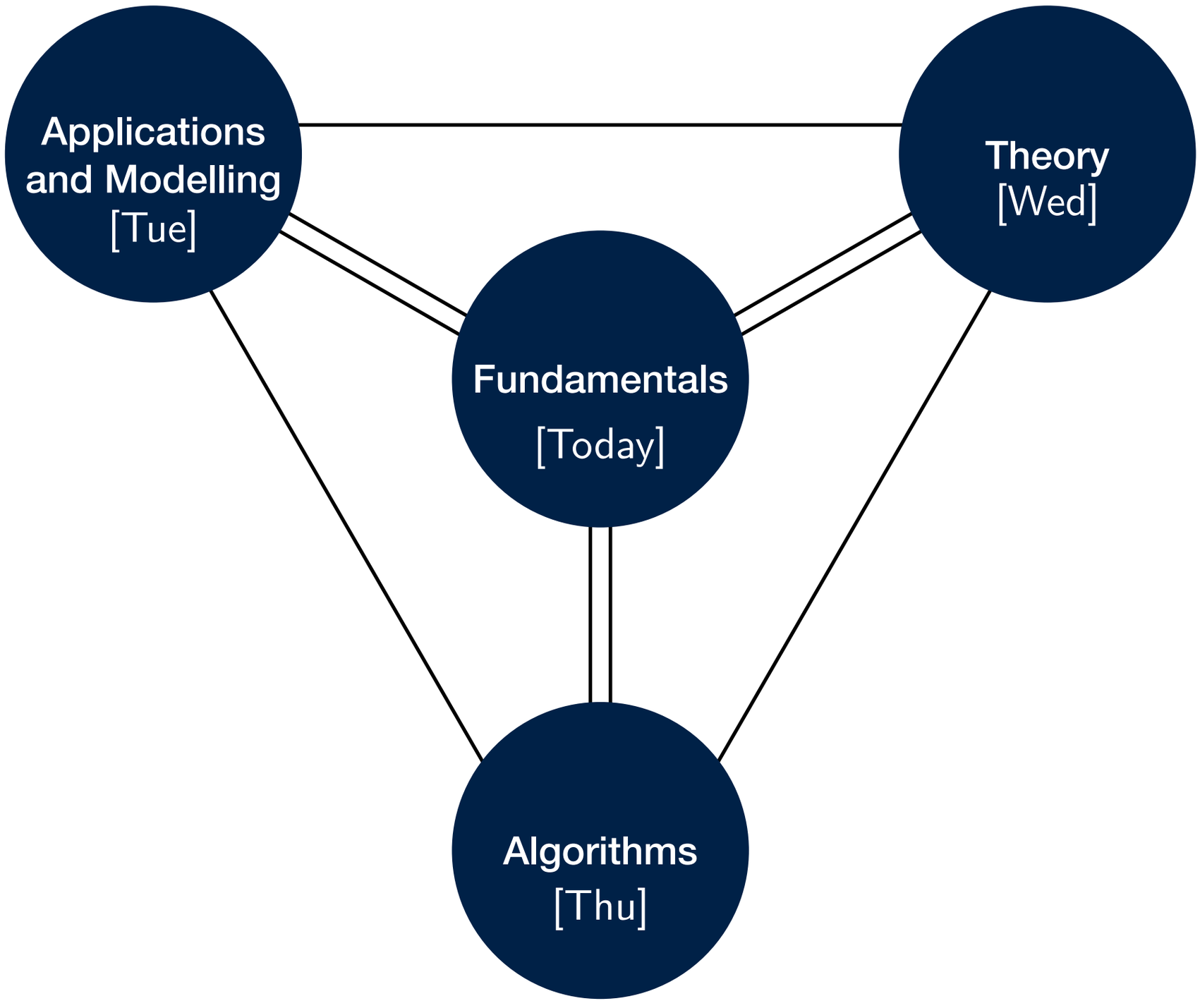
Convex Optimization, Cambridge University Press.

Who cares about optimization?

Optimization is about making **good decisions** in a principled way, often subject to **constraints**. Applications are everywhere in science, mathematics, and business:

- Managing an investment portfolio
- Training neural networks
- Scheduling public transport
- Optimizing supply chains
- Designing electronic circuit layouts
- Choosing worker shift patterns
- Real-time optimal control

This course:


Applications and Modelling [Tue]
Theory [Wed]
Fundamentals [Today]
Algorithms [Thu]

Describing an optimization problem

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x) \\ \text{subject to:} \quad & x \in X \end{aligned}$$

Key ingredients:

- A vector x collects the **decision variables**
- A set $\mathcal{X}$ is the **domain** of the decision variables
- A set $X \subseteq \mathcal{X}$ is the **constraint set**.
  Describes the **feasible** decisions.
- An **objective** function $f : \mathcal{X} \mapsto \mathbb{R}$.
  Assigns a **cost** $f(x)$ to each decision x.

Describing an optimization problem

A more common following format:

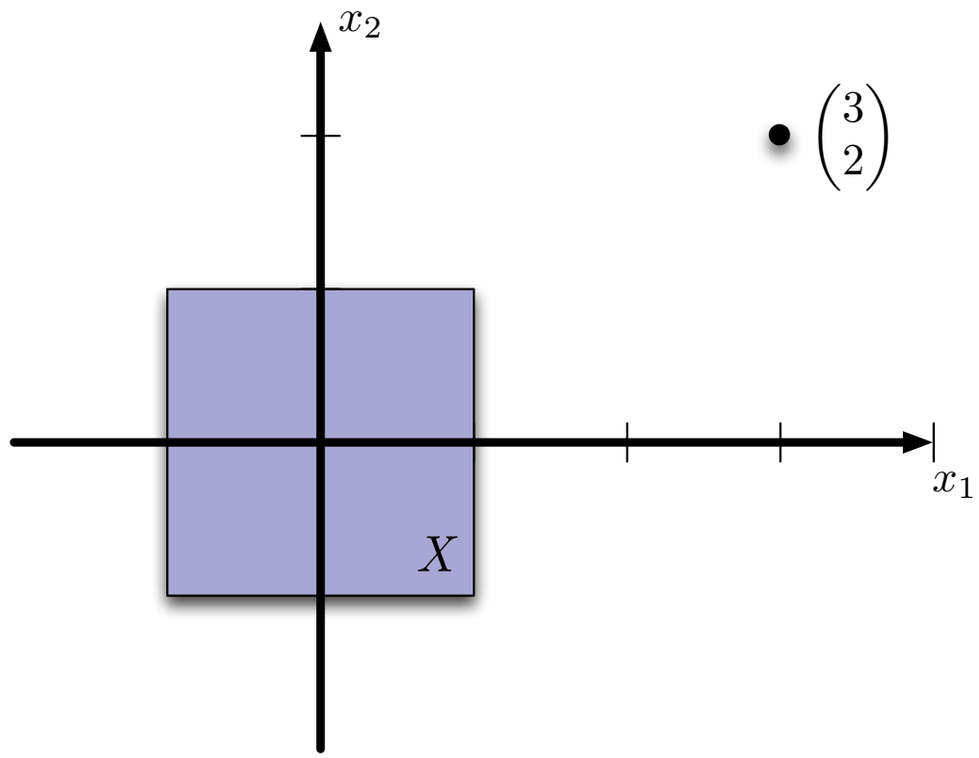
$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f_0(x) \\ \text{subject to:} \quad & f_i(x) \leq 0 \quad i = 1, \ldots, m \\ & h_i(x) = 0 \quad i = 1, \ldots, p \end{aligned}$$

Defined by the following **problem data**:

- **Objective function** $f_0 : \mathcal{X} \to \mathbb{R}$
- **Domain** $\mathcal{X} \subseteq \mathbb{R}^n$ of the objective function.
- Optional **inequality constraints** $f_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, m$
- Optional **equality constraints** $h_i : \mathbb{R}^n \to \mathbb{R}$, for $i = 1, \ldots, p$

A simple example

Problem: Find the point in the unit box X closet to the point (3,2)



Same problem in standard format:

$$\begin{aligned} \min_{(x_1,x_2)\in\mathbb{R}^2} \quad & (x_1-3)^2+(x_2-2)^2 \\ \text{subject to:} \quad & x_1 \le 1 \\ & -x_1 \le 1 \\ & x_2 \le 1 \\ & -x_2 \le 1 \end{aligned}$$

Properties of optimization problems

Consider a general **Nonlinear Program** (NLP):

$$J^* = \min_{x \in \mathcal{X}} f(x)$$

Notation:

- If $J^* = -\infty$, then the problem is **unbounded below**.
- If $\mathcal{X}$ is empty, then the problem is **infeasible** $(J^* = +\infty)$.
- If $\mathcal{X} = \mathbb{R}^n$, the problem is **unconstrained**.
- There might be more than one solution. The solution set is:

$$\arg\min_{x \in \mathcal{X}} f(x) := \{x \in \mathcal{X} \mid f(x) = J^*\}$$

What can go wrong?

- If the constraints are inconsistent, then the problem is **infeasible**. Example:

$$\begin{aligned} \min_{x \in \mathbb{R}} \quad & x^2 \\ \text{subject to:} \quad & x \leq -1 \\ & x \geq 1 \end{aligned}$$

- It might be possible to make $f_0(x)$ arbitrarily negative without violating any of the constraints. The problem is **unbounded**. Example:

$$\begin{aligned} \min_{x \in \mathbb{R}} \quad & x \\ \text{subject to:} \quad & x \leq 0 \end{aligned}$$

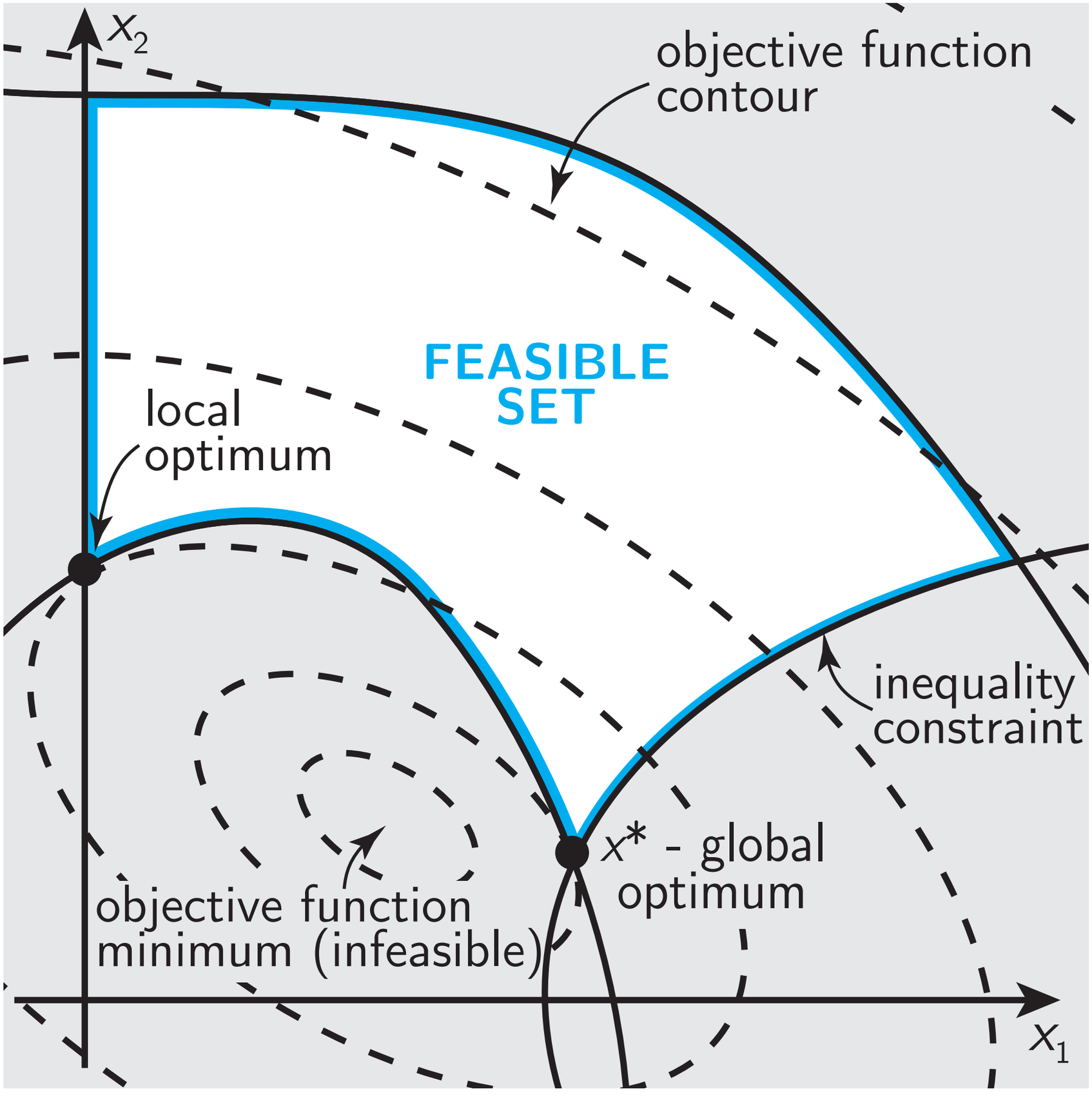
What can go wrong?

- The value J^* might be finite, but there is no x that achieves it. Example:

$$\begin{aligned} \inf_{x \in \mathbb{R}} \quad & e^{-x} \\ \text{subject to:} \quad & x \geq 0 \end{aligned}$$

The optimal value $J^* = 0$ exists, but there are no optimal solutions.

Geometry of an optimization problem

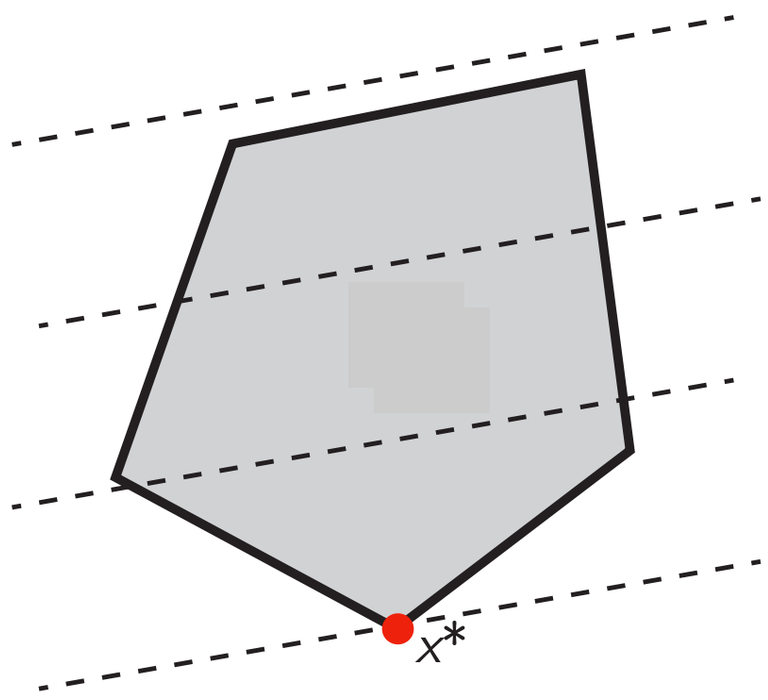


Common problem types

“Easy” problems: Linear and (convex) quadratic programs

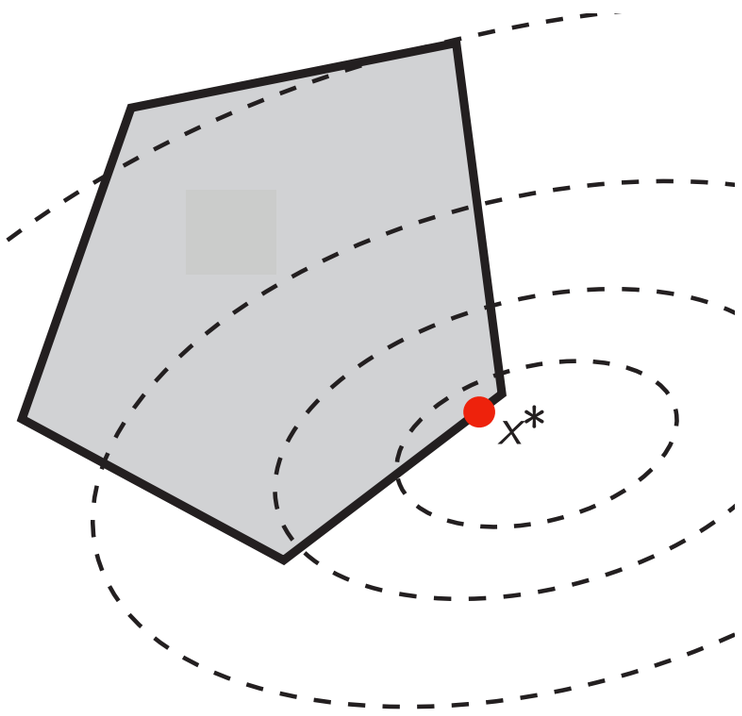
Linear Program (LP): Linear cost and constraint functions; feasible set is a polyhedron.

$$\begin{aligned} \min_{x} \quad & c^\top x \\ \text{subject to:} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$



Convex Quadratic Program (QP): Quadratic cost and linear constraint functions; feasible set is a polyhedron. Convex if $P \succeq 0$.

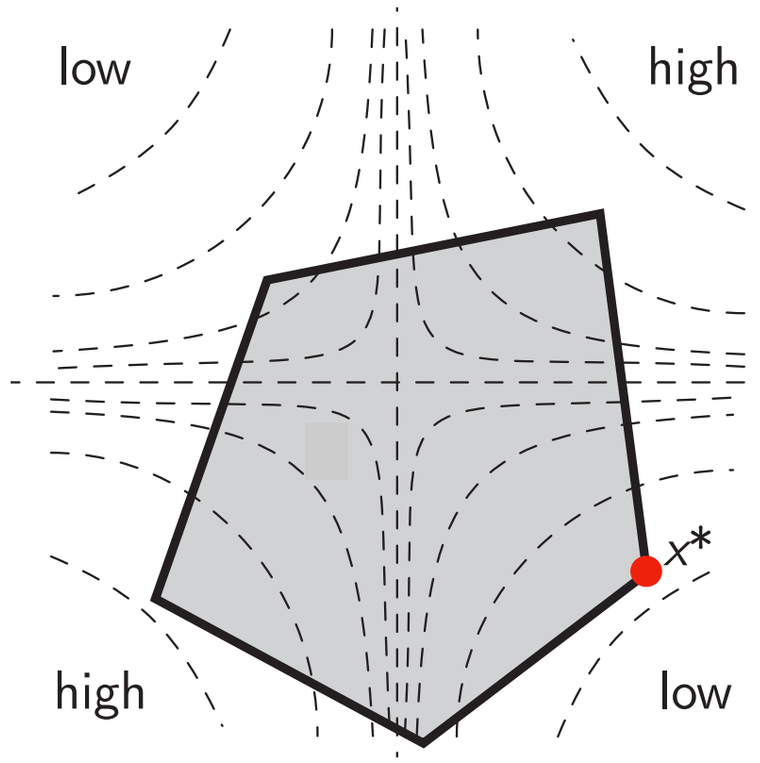
$$\begin{aligned} \min_{x} \quad & \tfrac{1}{2} x^\top P x + q^\top x \\ \text{subject to:} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$



“Hard” problems: Nonconvex and integer programs

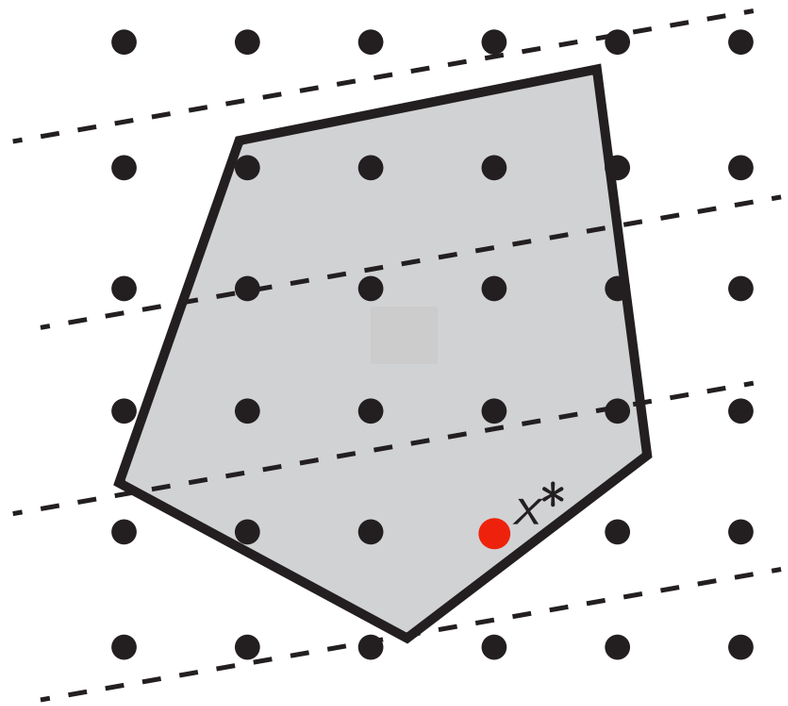
Nonconvex Quadratic Program: QP with $P \not\succeq 0$.

$$\begin{aligned} \min_x \quad & \tfrac{1}{2} x^\top P x + q^\top x \\ \text{subject to:} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$



Mixed Integer Linear Program (MILP): Linear program with binary or integer constraints.

$$\begin{aligned} \min_x \quad & c^\top x \\ \text{subject to:} \quad & Gx \leq h \\ & Ax = b \\ & x \in \{0,1\}^n \text{ or } x \in \mathbb{Z}^n \end{aligned}$$



Software tools for optimization

A simple optimization problem:

$$\begin{aligned} \min_{x_1,x_2} \quad & |x_1 + 5| + |x_2 - 3] \\ \text{subject to:} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

This problem is a **linear program.**

- Huge variety of tools for solving LPs and QPs (and other standard forms).
  - **Examples:** Matlab (linprog), OSQP, qpOASES, ECOS, SDPT3, Sedumi, MOSEK, Gurobi, IPOPT
- **No standard interface** to solvers. They are almost all different.
- General purpose modelling tools allow easy switching between solvers.
  - CVX / Yalmip (Matlab). CvxPy (Python). JuMP (Julia), GAMS, AMPL..

Software tools for optimization (Matlab)

A simple optimization problem:

$$\begin{aligned} \min_{x_1,x_2} \quad & |x_1 + 5| + |x_2 - 3] \\ \text{subject to:} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

The YALMIP toolbox :

```
%make variables
sdpvar x1 x2;

%define cost function
f = abs(x1 + 5) + abs(x2 - 3);

%define constraints
X = (2.5 <= x1 <= 5) & ...
    ( -1 <= x2 <= 1);

%solve
optimize(X,f);
```

Software tools for optimization (Matlab)

A simple optimization problem:

$$\begin{aligned} \min_{x_1,x_2} \quad & |x_1 + 5| + |x_2 - 3] \\ \text{subject to:} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

The CVX toolbox for Matlab:

```
cvx_begin
    %make variable
variables x1 x2
    %define problem
minimize(abs(x1 + 5) + abs(x2 - 3))
subject to
2.5 <= x1 <= 5
 -1 <= x2 <= 1

cvx_end    %solves automatically
```

Software tools for optimization (Python)

A simple optimization problem:

$$\begin{aligned} \min_{x_1,x_2} \quad & |x_1 + 5| + |x_2 - 3] \\ \text{subject to:} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

The CvxPy toolbox for Python :

```
import cvxpy as cp

#make variable
x = cp.Variable(2)
#define objective
f = cp.abs(x[0] + 5) + cp.abs(x[1] - 3)

#define constraints
constr = [2.5 <= x[0], x[0] <= 5,
          -1 <= x[1], x[1] <= 1]

#make problem and solve
prob = cp.Problem(cp.Minimize(f),constr)
prob.solve()
```

Software tools for optimization (Julia)

A simple optimization problem:

$$\begin{aligned} \min_{x_1,x_2} \quad & |x_1 + 5| + |x_2 - 3] \\ \text{subject to:} \quad & 2.5 \leq x_1 \leq 5 \\ & -1 \leq x_2 \leq 1 \end{aligned}$$

The JuMP toolbox for Julia :

```
using JuMP, OSQP
model = Model(OSQP.Optimizer)

#make variable
@variable(model, x[1:2])

#define objective
@NLobjective(model,Min, abs(x[1] + 5) + abs(x[2] - 3) )

#define constraints
@constraint(model, 2.5 <= x[1] <= 5)
@constraint(model,  -1 <= x[2] <= 1)

#solve
optimize!(model)
```

Convex sets

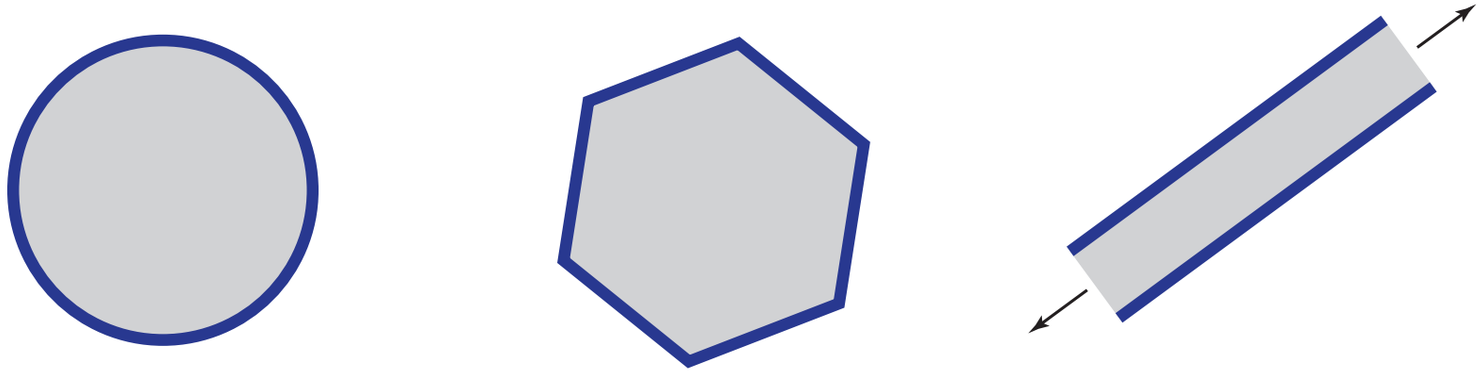
Convex sets

A set $\mathcal{X}$ is **convex** *if and only if* for any pair of points x and y in $\mathcal{X}$, any **convex combination** of x and y lies in $\mathcal{X}$:

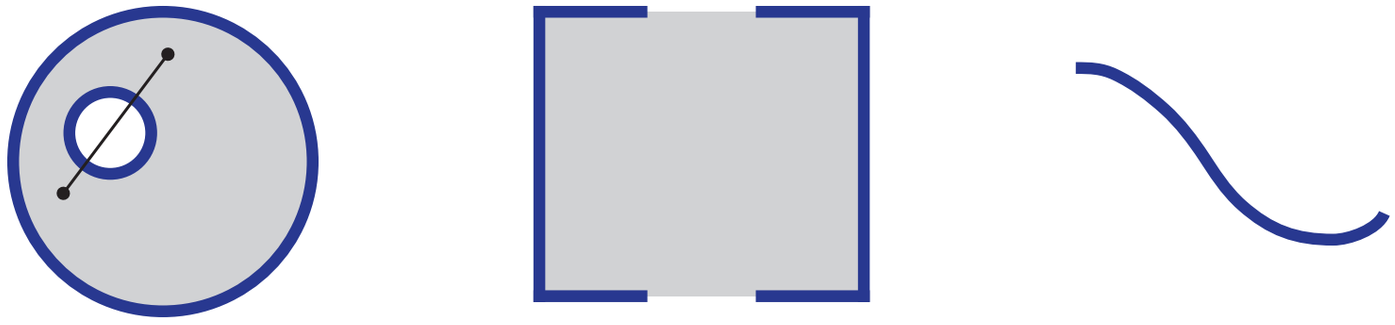
$$\mathcal{X} \text{ is convex} \quad \Leftrightarrow \quad \lambda x + (1-\lambda)y \in \mathcal{X},\ \forall \lambda \in [0,1],\ \forall x, y \in \mathcal{X}$$

Easier version : All line segments starting and ending in $\mathcal{X}$ stay in $\mathcal{X}$

Convex:



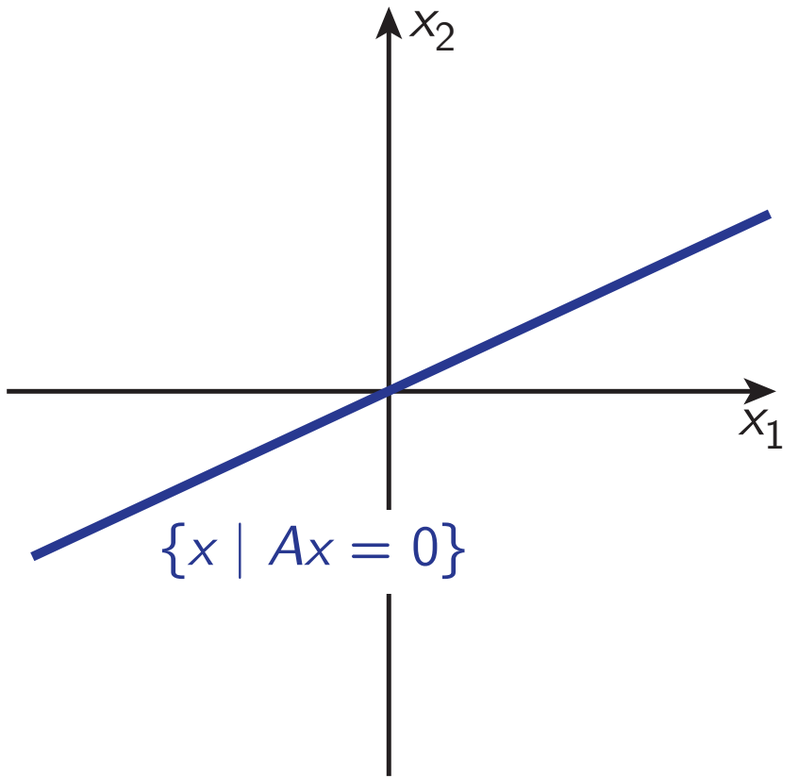
Non-convex:



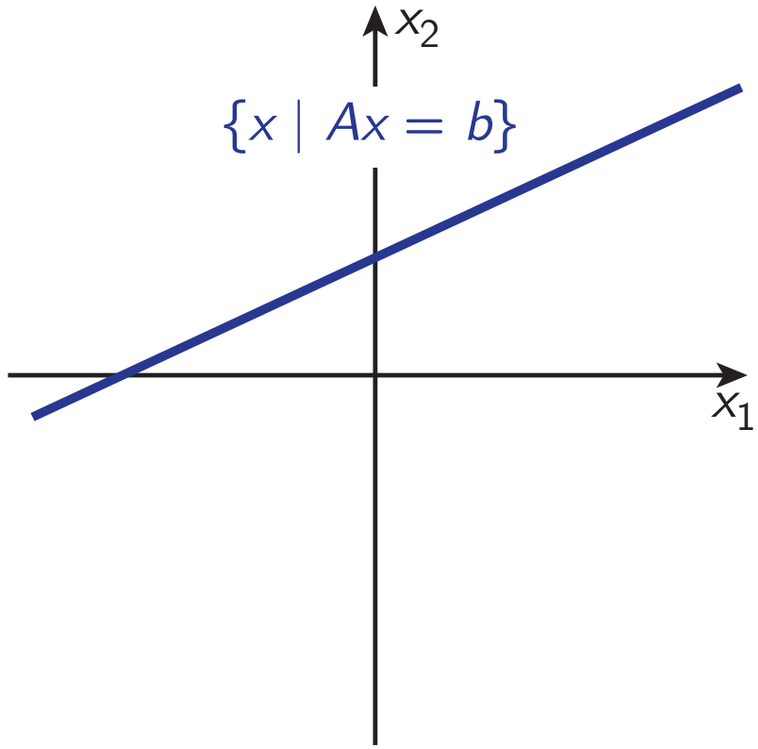
Affine sets

An **affine set** is a convex set defined by $\mathcal{X} = \{x \in \mathbb{R}^n \mid Ax = b\}$.

This definition covers lines, planes and single points.



A 1-d subspace in $\mathbb{R}^2$



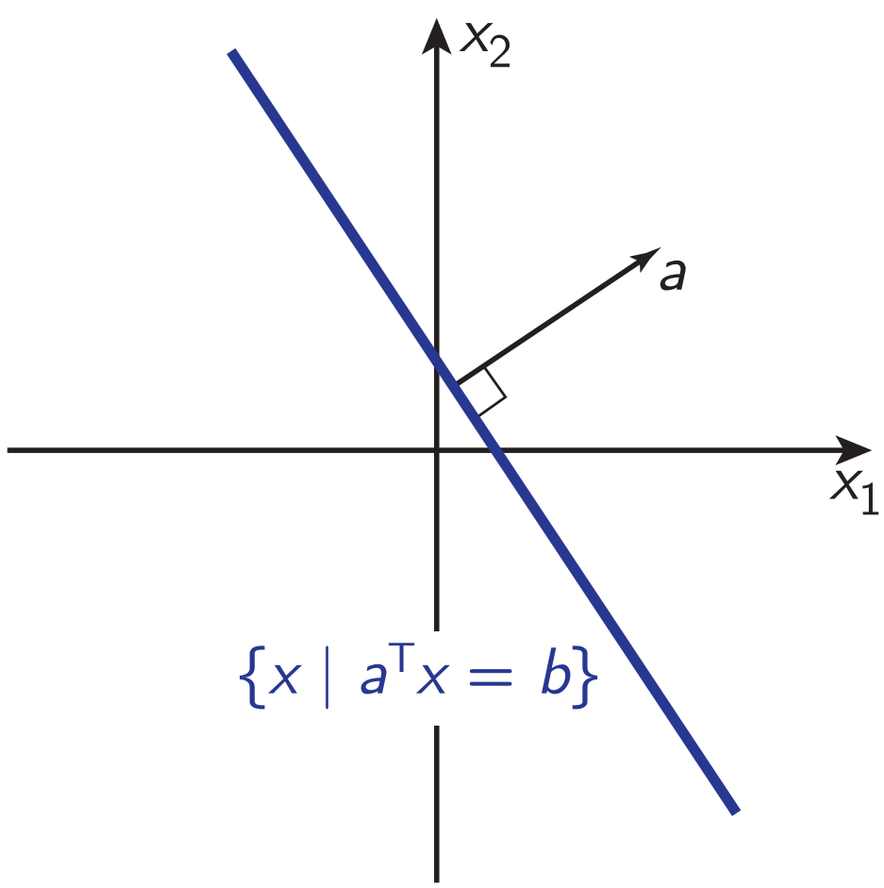
An affine set in $\mathbb{R}^2$

Hyperplanes and half spaces

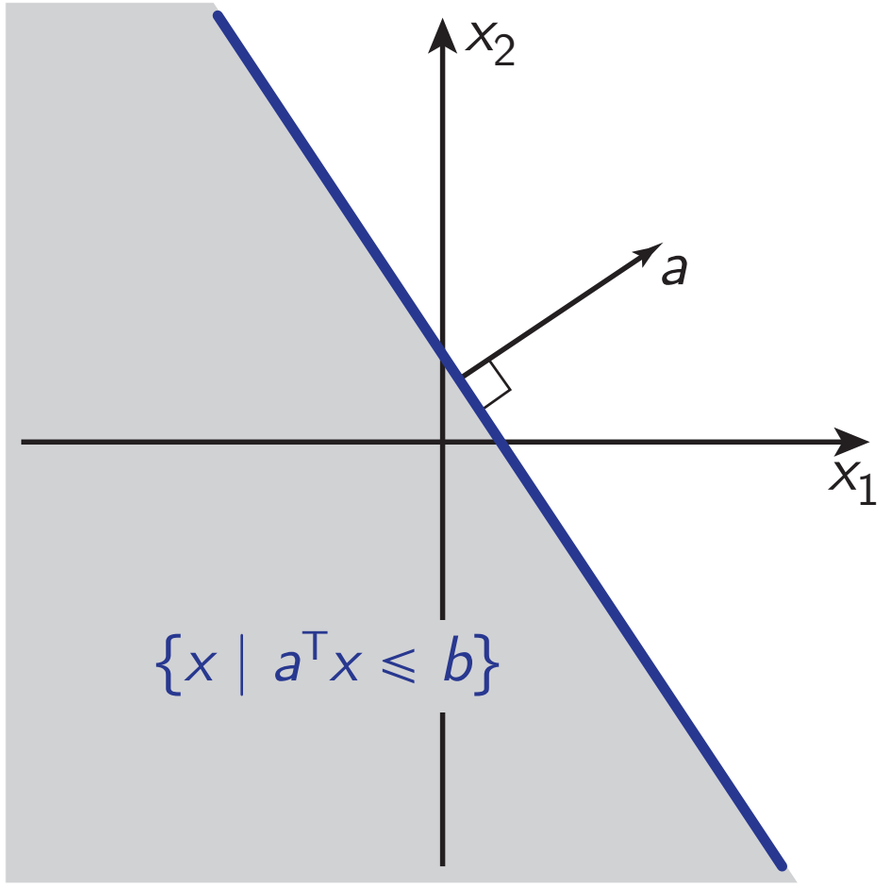
A **hyperplane** is defined by $\{x \in \mathbb{R}^n \mid a^\top x = b\}$ for $a \neq 0$.

A **halfspace** is everything on one side of a hyperplane, i.e. $\{x \in \mathbb{R}^n \mid a^\top x \leq b\}$.

Hyperplanes and halfspaces are always convex.



A hyperplane

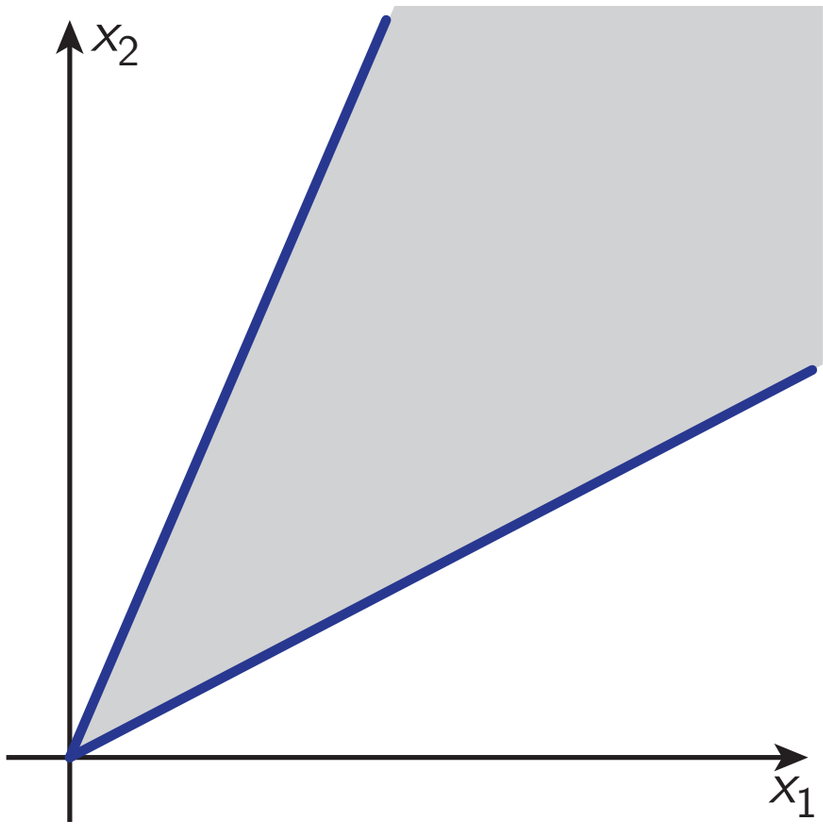


A halfspace

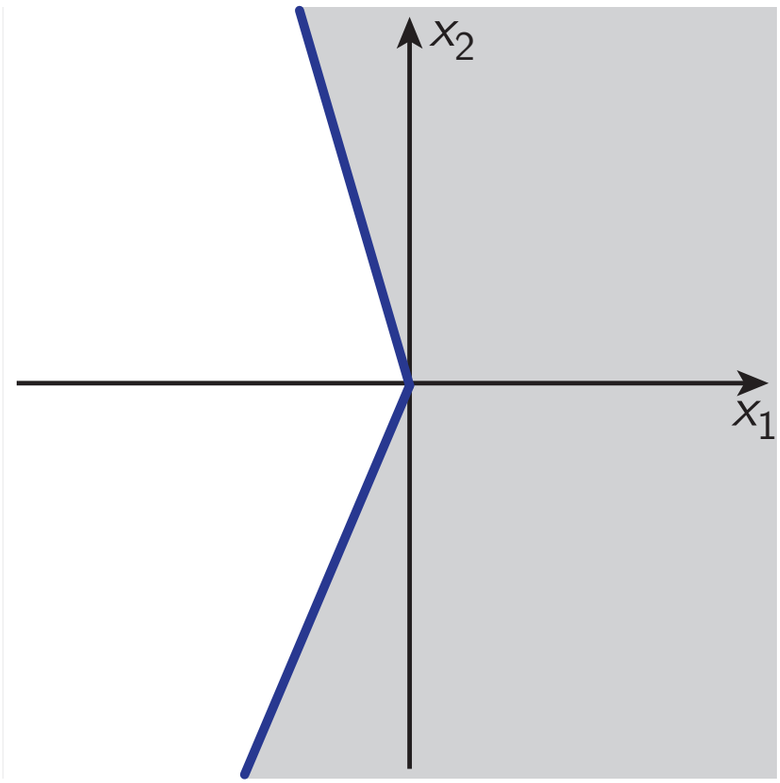
Cones

A set $\mathcal{X}$ is a **cone** if for all $x \in \mathcal{X}$, and for all $\theta > 0$, $\theta x \in \mathcal{X}$.

A cones is **not necessarily** convex.



A convex cone



A non-convex cone

A **convex cone** is 1) convex and 2) a cone.

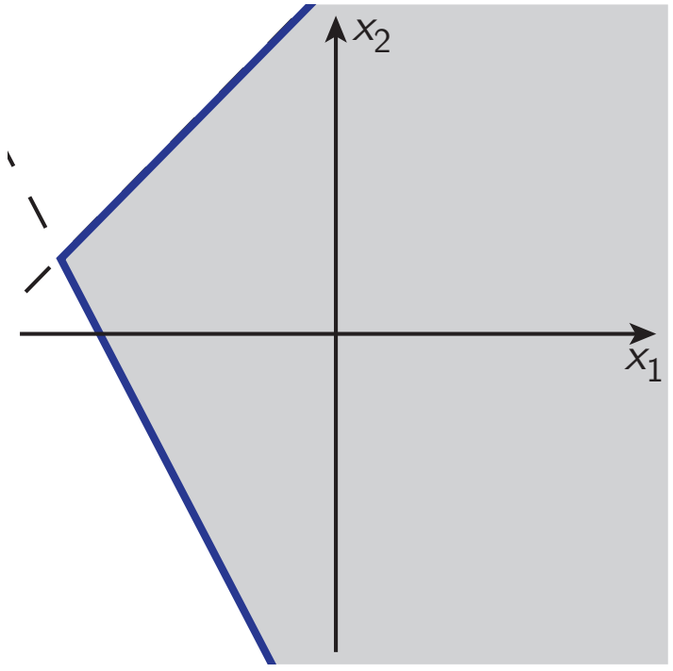
Polyhedra and polytopes

A **polyhedron** is the intersection of a *finite* number of halfspaces:

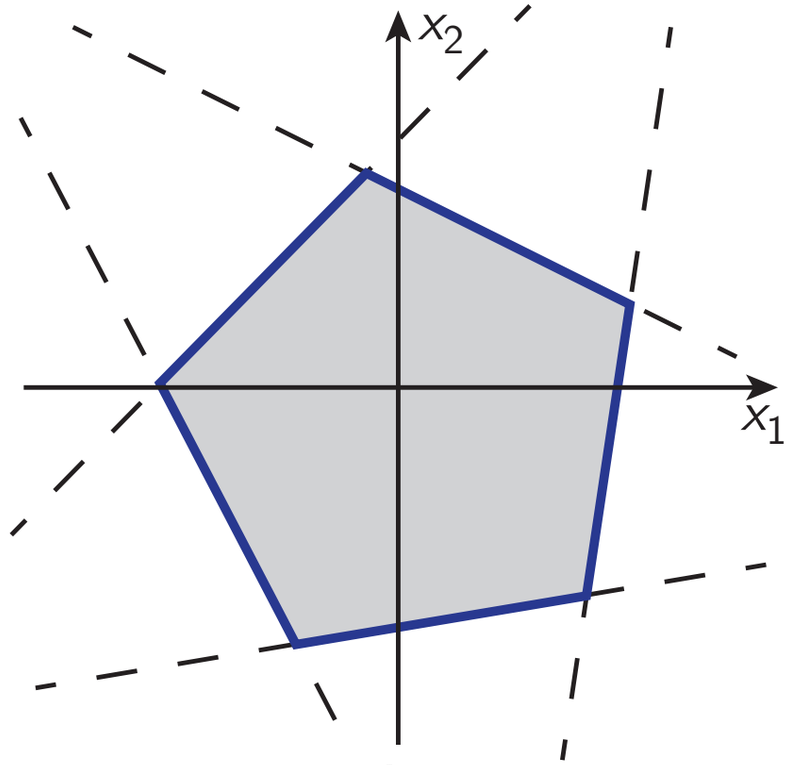
$$\begin{aligned}\mathcal{X} &= \{x \,|\, a_1^\top x \leq b_1,\, a_2^\top x \leq b_2, \ldots, a_m^\top \leq b_m\} \\ &= \{x \,|\, Ax \leq b\}\end{aligned}$$

where $A := [a_1, a_2, \ldots, a_m]^\top$ and $b := [b_1, b_2, \ldots, b_m]^\top$.

A **polytope** is a *bounded* polyhedron.



An (unbounded) polyhedron



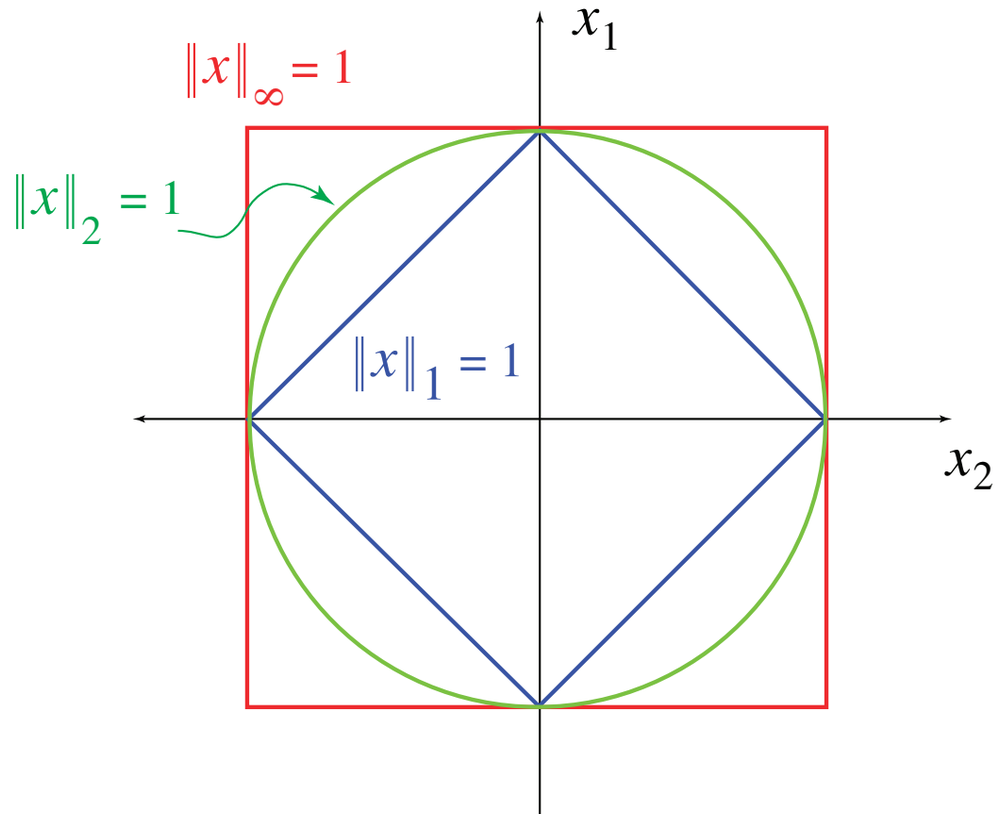
A polytope

Norm balls and cones

Norm ball

A norm ball with center x_c is radius r is the set $\{x \mid \|x - x_c\| \leq r\}$.

The 'shape' of the ball depends on the norm.

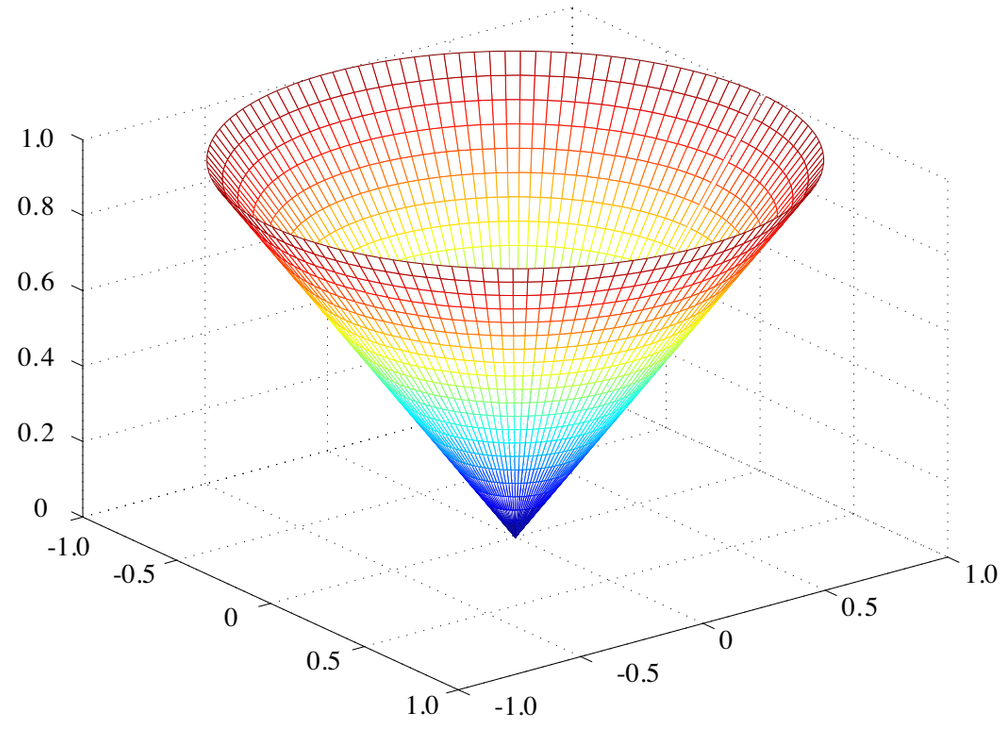


Norm cone

A norm cone is a set

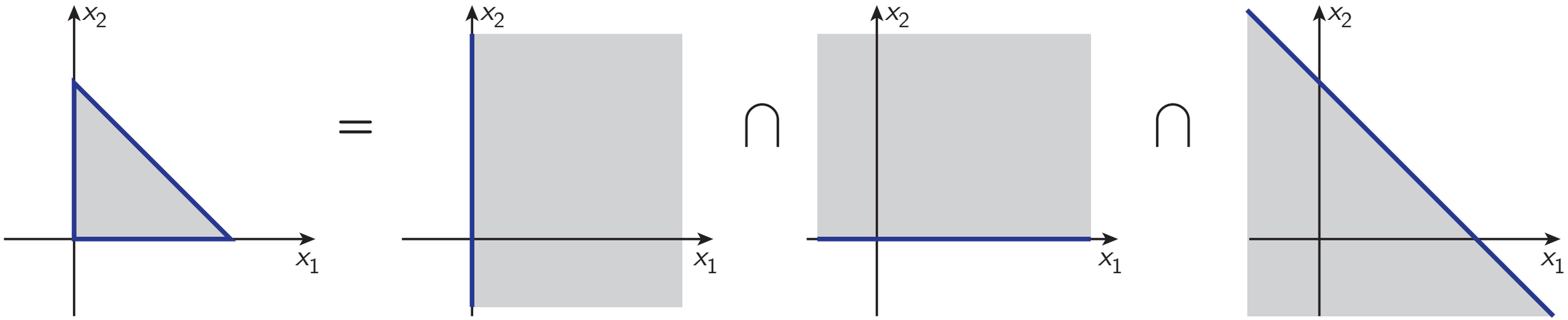
$$\{(t, x) \mid \|x\| \leq t\}$$

Norm balls and norm cones are convex sets.



Set intersections

The intersection of two or more convex sets is convex.



- Many sets can be written as the intersection of simpler convex elements, and are therefore convex.

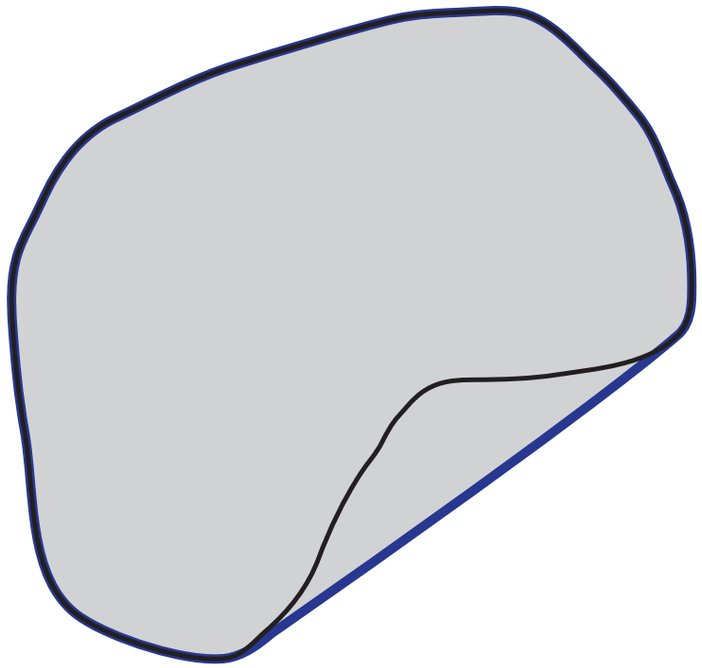
- Any convex set can be written as a (possibly infinite) intersection of halfspaces.

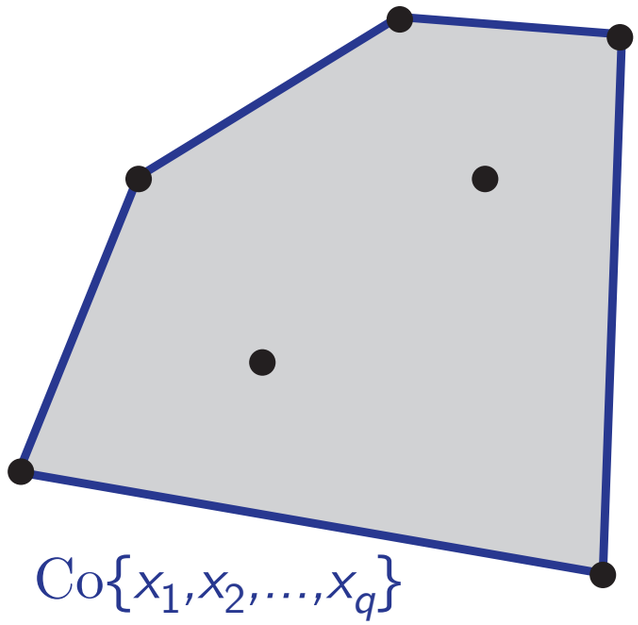
Convex hull

The **convex hull** of a set $\mathcal{X}$ is the set of all convex combinations of points in $\mathcal{X}$:

$$\mathrm{Co}(\mathcal{X}) := \{x \,|\, x = \lambda a + (1-\lambda)b,\ \lambda \in [0,1],\ a, b \in \mathcal{X}\}$$

It is the smallest convex set that contains $\mathcal{X}$: for all convex sets $\mathcal{Y} \supseteq \mathcal{X}$, $\mathrm{Co}(\mathcal{X}) \subseteq \mathcal{Y}$.



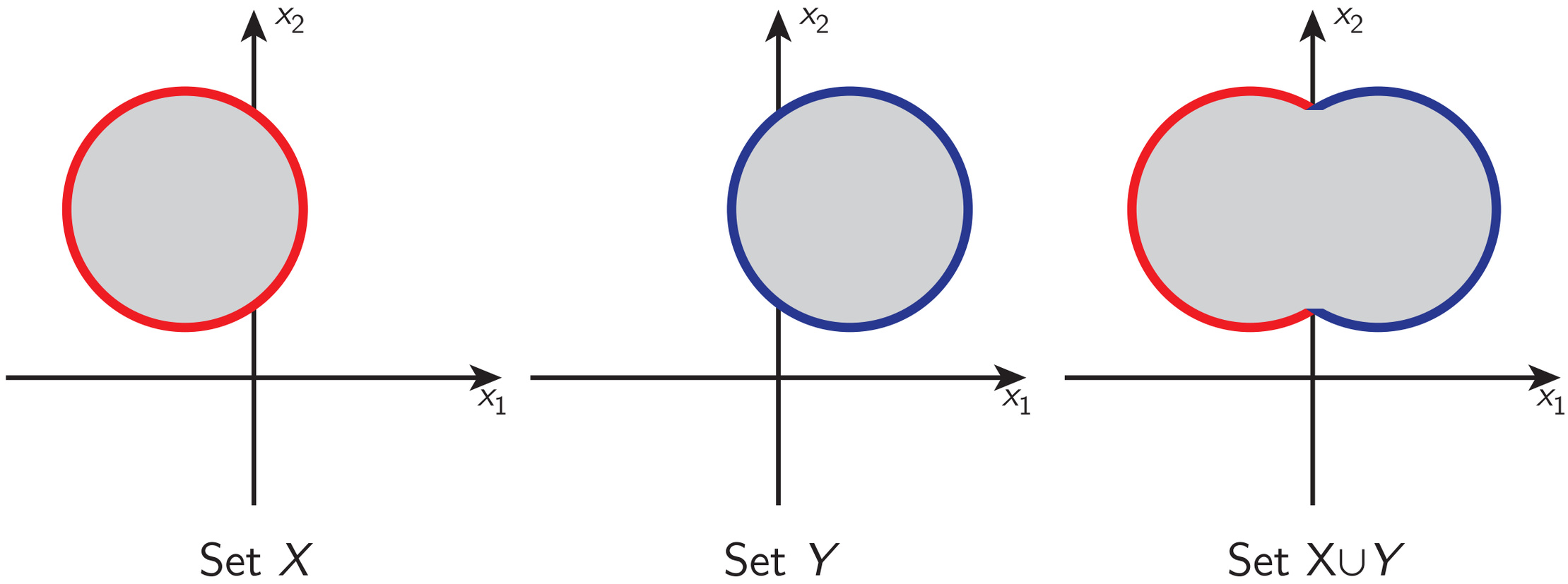


For a set $\mathcal{X} = \{x_1, x_2, \ldots, x_q\}$ comprising q points, the convex hull can be written

$$\mathrm{Co}(\mathcal{X}) = \left\{ \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_q x_q \;\middle|\; \lambda_i \geq 0,\ i = 1, \ldots, q,\ \sum_{i=1}^{q} \lambda_i = 1 \right\}$$

Set unions

the **union** of two sets is **not** convex in general, regardless of whether the original sets were convex!



Implication : logical OR conditions are usually not convex.

Convex functions

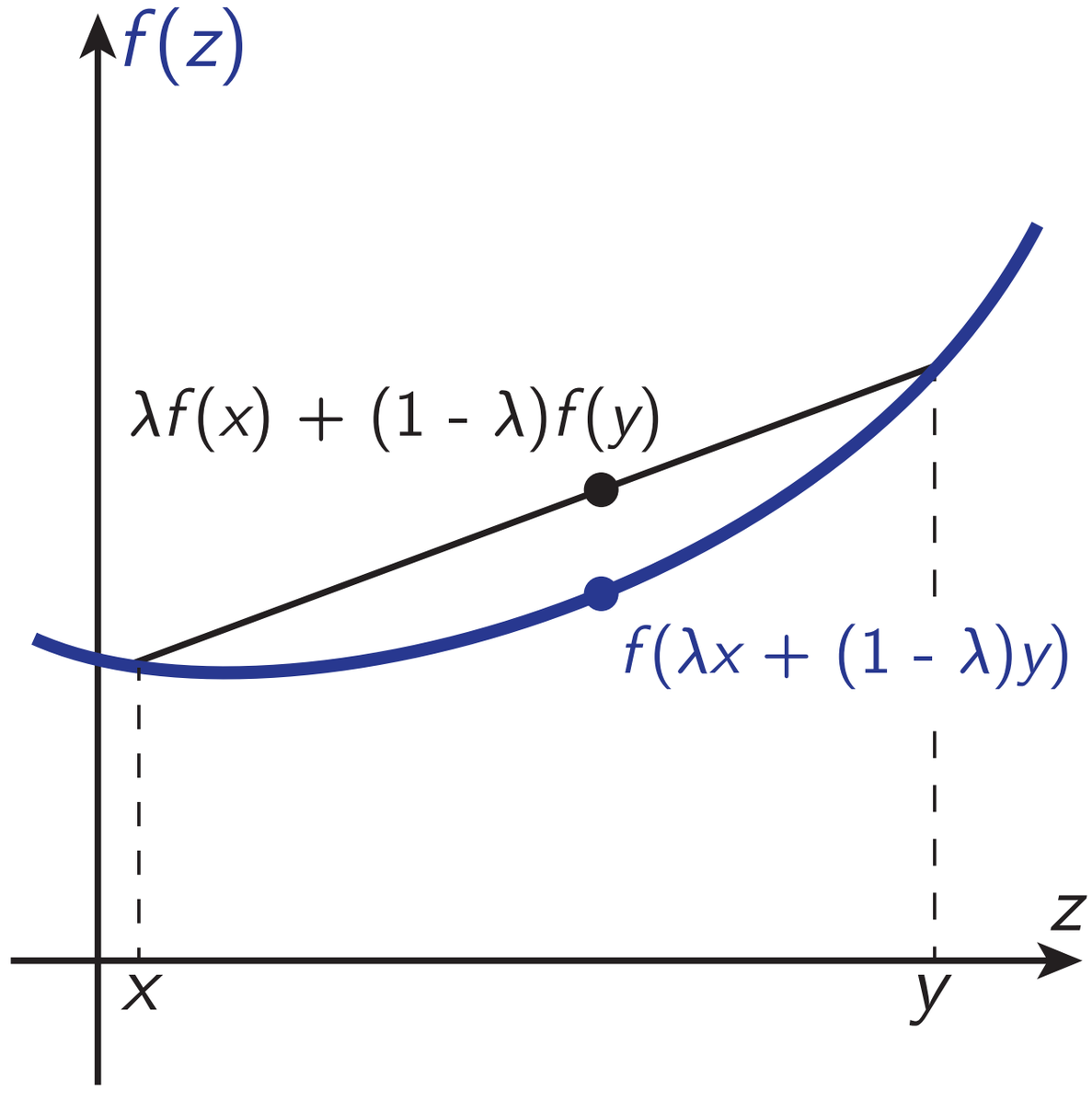
Convex functions

A function $f : \mathrm{dom}(f) \to \mathbb{R}$ is **convex** iff its domain $\mathrm{dom}(f)$ is convex *and*

$$f(\lambda x+(1-\lambda)y) \leq \lambda f(x)+(1-\lambda)f(y), \quad \forall \lambda \in (0,1), \quad \forall x, y \in \mathrm{dom}(f)$$

- **Stricly** convex if the inequality is strict

- **Concave** if -f is convex

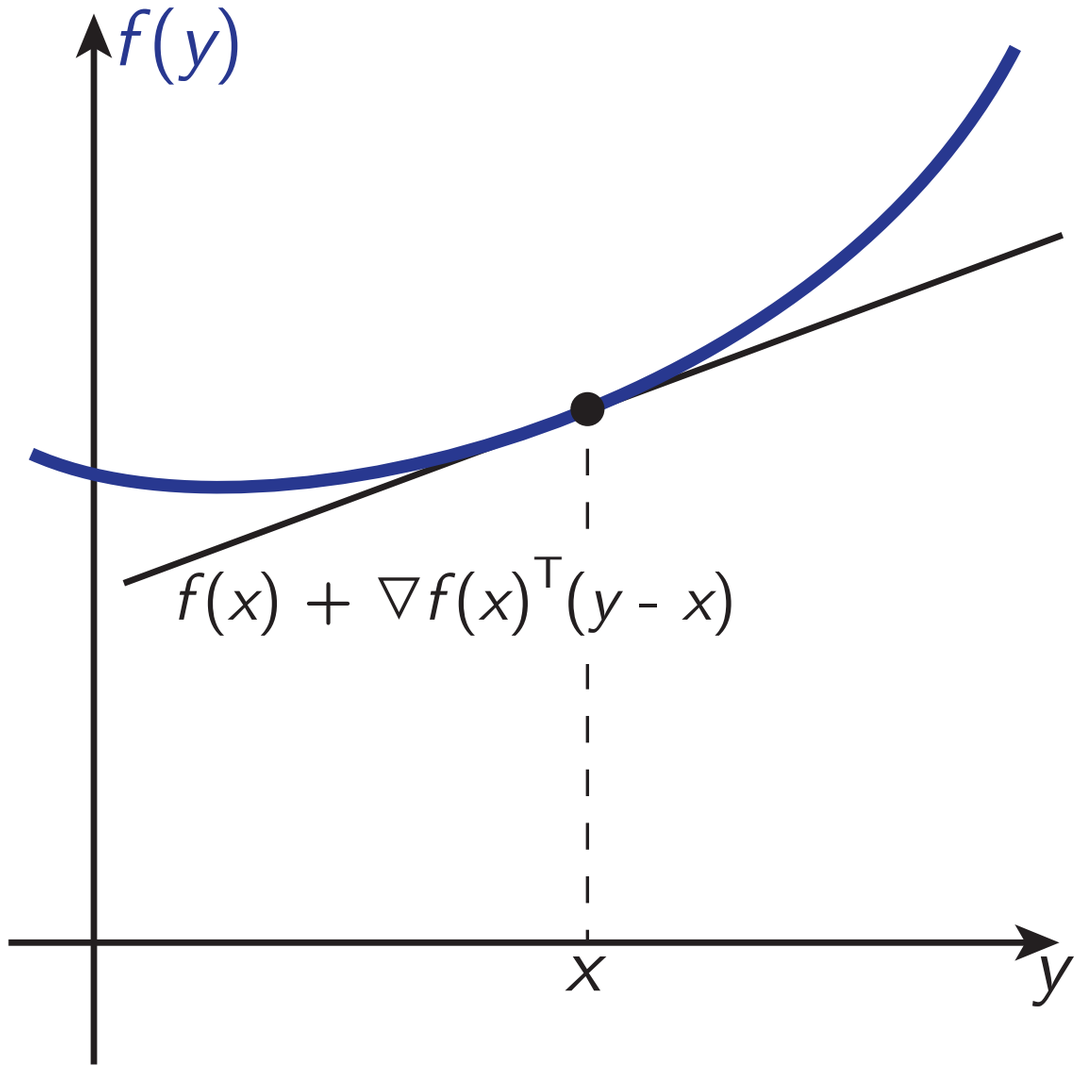


1st order conditions for convexity

A differentiable function $f : \mathrm{dom}(f) \to \mathbb{R}$ with a convex domain is **convex** *iff*

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x), \quad \forall x, y \in \mathrm{dom}(f)$$

First order approximation of f around any x is a global underestimator of f.



2nd-order condition for convexity

A twice-differentiable function $f : \mathrm{dom}(f) \to \mathbb{R}$ is **convex** *iff* its domain $\mathrm{dom}(f)$ is convex *and*

$$\nabla^2 f(x) \succeq 0, \quad \forall x \in \mathrm{dom}(f)\,,$$

where the **Hessian** $\nabla^2 f(x)$ is defined by

$$\nabla^2 f(x)_{ij} = \frac{\partial^2 f(x)}{\partial x_i \partial x_j}$$

If $\mathrm{dom}(f)$ is convex *and* $\nabla^2 f(x) \succ 0$ for all $x \in \mathrm{dom}(f)$, then f is **strictly convex**.

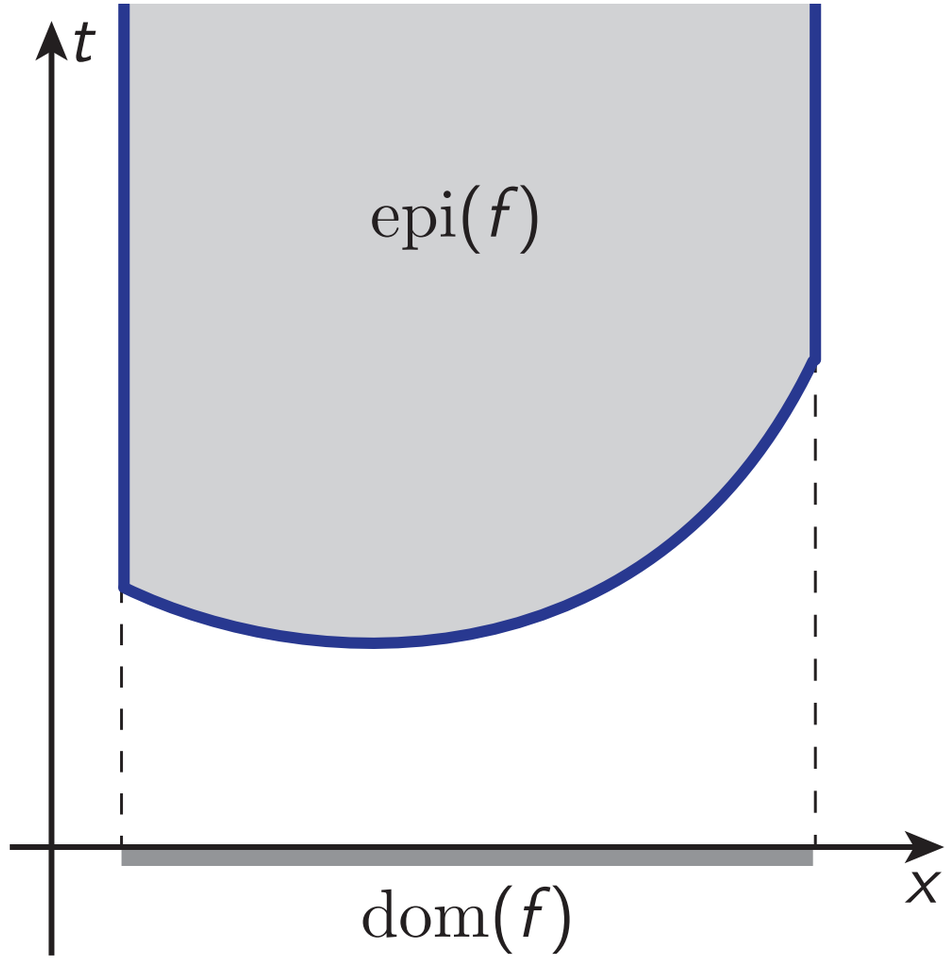
Epigraph of a function

The **epigraph** of a function $f : \mathrm{dom}(f) \to \mathbb{R}$ is the **set**

$$\mathrm{epi}(f) = \left\{ \begin{bmatrix} x \\ t \end{bmatrix} \,\middle|\, x \in \mathrm{dom}(f),\ f(x) \le t \right\} \subseteq \mathrm{dom}(f) \times \mathbb{R}$$

It has dimension one higher than the domain of f.

A function is convex *iff* its epigraph is a convex set.



Examples of convex functions $\mathbb{R} \to \mathbb{R}$

The following functions are **convex** (on domain $\mathbb{R}$ unless otherwise stated):

- Affine: $ax + b$ for any $a, b \in \mathbb{R}$
- Exponential: e^{ax} for any $a \in \mathbb{R}$
- Powers: x^α on domain $\mathbb{R}_{++}$, for $\alpha \geq 1$ or $\alpha \leq 0$
- Powers of absolute value: $|x|^p$, for $p \geq 1$

The following functions are **concave** (on domain $\mathbb{R}$ unless otherwise stated):

- Affine: $ax + b$ for any $a, b \in \mathbb{R}$
- Powers: x^α on domain $\mathbb{R}_{++}$, for $0 \leq \alpha \leq 1$
- Logarithm: $\log x$ on domain $\mathbb{R}_{++}$
- Entropy: $-x \log x$ on domain $\mathbb{R}_{++}$

Examples of convex functions $\mathbb{R}^n \to \mathbb{R}$

Affine functions on $\mathbb{R}^n$ are both convex and concave:

- On $\mathbb{R}^n$, for some $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$:

$$f(x) = a^\top x + b$$

Vector Norms on $\mathbb{R}^n$ are all convex:

- On $\mathbb{R}^n$, ℓ_p norms have the form, for $p \geq 1$,

$$||x||_p = \left(\sum_{i=1}^{n} |x|^p \right)^{1/p}, \quad \text{with } ||x||_\infty = \max_i |x_i|$$

Examples of convex functions $\mathbb{R}^{m\times n}\to\mathbb{R}$

Affine functions on $\mathbb{R}^{m\times n}$ are both convex and concave:

- On $\mathbb{R}^{m\times n}$, for some $A\in\mathbb{R}^{m\times n}$, $b\in\mathbb{R}$:

$$f(X)=\mathrm{trace}(A^\top X)+b=\sum_{i=1}^{m}\sum_{j=1}^{n}A_{ij}X_{ij}+b$$

Matrix Norms on $\mathbb{R}^{m\times n}$ are all convex:

- On $\mathbb{R}^{m\times n}$ the **spectral**, or **maximum singular value** norm is

$$||X||_2=\sigma_{\max}(X)=[\lambda_{\max}(X^\top X)]^{1/2}.$$

Convexity preserving operations

Certain operations preserve the convexity of functions:

- Non-negative weighted sum
- Composition with affine function
- Pointwise maximum and supremum
- Partial minimization

and many other possibilities...

Convexity preserving operations

Non-negative weighted sum

If f is a function convex, then αf is convex for $\alpha \geq 0$. For several convex functions f_i, $\sum_i \alpha_i f_i$ is convex if all $\alpha_i \geq 0$.

Composition with affine function

If f is a convex function, then $f(Ax + b)$ is convex.

Example: $||Ax - b||$ is convex for any norm.

Pointwise maximum

If $f_1, \ldots, f_m$ are convex functions, then $f(x) = \max\{f_1(x), \ldots, f_m(x)\}$ is convex.

Example: Piecewise linear functions $\max_{i=1,\ldots,m}\{a_i^\top x + b\}$ are convex.

Convexity preserving operations (cont'd)

Pointwise supremum

If $f(x, y)$ is convex in x for every $y \in \mathcal{Y}$, then $g(x) = \sup_{y \in \mathcal{Y}} f(x, y)$ is convex.

Examples

- Support function of a set $\mathcal{A}$:

$$S_{\mathcal{A}}(x) = \sup_{y \in \mathcal{A}} y^\top x$$

- Distance to farthest point in a (possibly non-convex) set $\mathcal{B}$:

$$f(x) = \sup_{y \in \mathcal{B}} ||x - y||$$

- Maximum eigenvalue of a matrix $X \in \mathbb{S}^n$:

$$\lambda_{\max}(X) = \sup_{||y||_2 \leq 1} y^\top X y$$

Convexity preserving operations (cont'd)

Parametric Minimization

If $f(x, y)$ is convex in (x, y) and the set $\mathcal{C}$ is convex, then

$$g(x) = \min_{y \in \mathcal{C}} f(x, y)$$

is convex.

Example: $\operatorname{dist}(x, \mathcal{S}) = \inf_{y \in \mathcal{S}} ||x - y||$ is convex if $\mathcal{S}$ is convex.

Convexity preserving operations (cont'd)

Composition with scalar functions

For $g : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R} \to \mathbb{R}$, $f(x) = h(g(x))$ is convex if:

- g is convex, h is convex, $\tilde{h}$ is non-decreasing
- g is concave, h is convex, $\tilde{h}$ is non-increasing

Examples

- $\exp g(x)$ for convex g
- $1/g(x)$ for concave positive g

Convex optimization

Standard form convex problems

A standard form **convex** optimization problem:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f_0(x) \\ \text{subject to:} \quad & f_i(x) \leq 0 \quad i = 1, \ldots, m \\ & a_i^\top x = b_i \quad i = 1, \ldots, p \end{aligned}$$

This problem is convex if:

- The domain $\mathcal{X}$ is a convex set.
- The objective function f_0 is a convex function.
- The inequality constraint functions f_i are all convex.
- The equality constraint functions $h_i(x) = a_i^\top x$ are all affine.

Standard form convex problems

The affine constraints are typically gathered into matrix form:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f_0(x) \\ \text{subject to:} \quad & f_i(x) \leq 0 \quad i = 1, \ldots, m \\ & Ax = b \quad A \in \mathbb{R}^{p \times m} \end{aligned}$$

Crucial Fact!

For a convex problem, **every** locally optimal solution is globally optimal.

NB: Writing or rewriting an optimization problem in convex form can be tricky, and is not always possible. It is always worth trying though.

Local and Global Optimality for Convex Problems

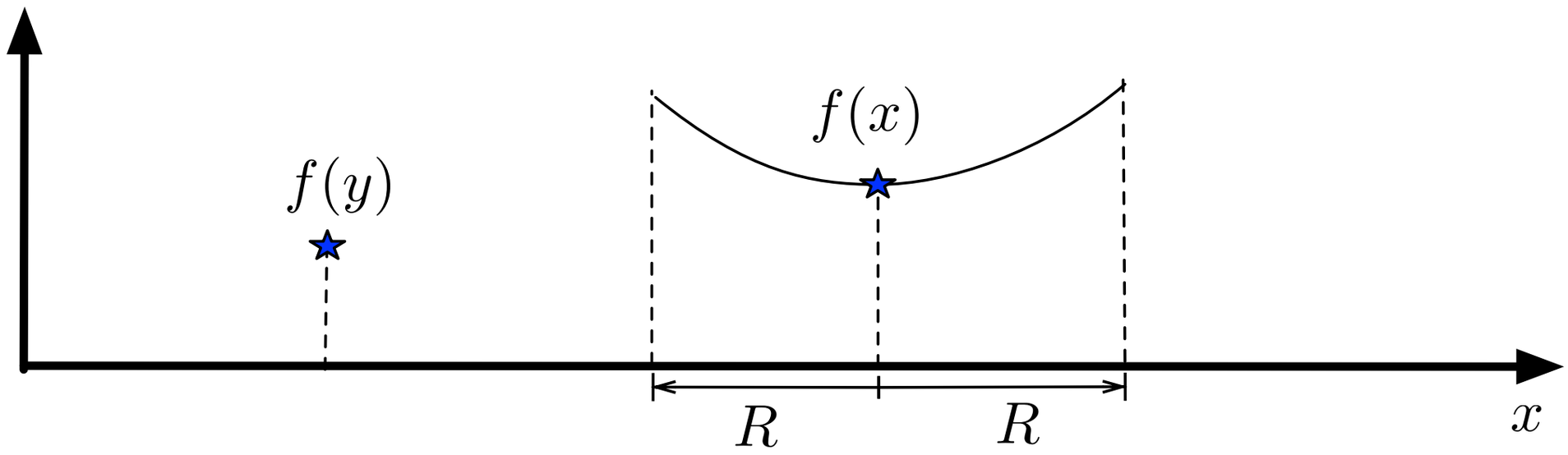
For a convex problem, **every** locally optimal solution is globally optimal.

Proof:

- Assume that x is locally optimal, but not globally optimal.
- Therefore there is some other point y such that $f(y) < f(x)$.
- x locally optimal implies that there is some $R > 0$ such that

$$\|z - x\|_2 \leq R \Rightarrow f(x) \leq f(z)$$

- The problem can't be convex.



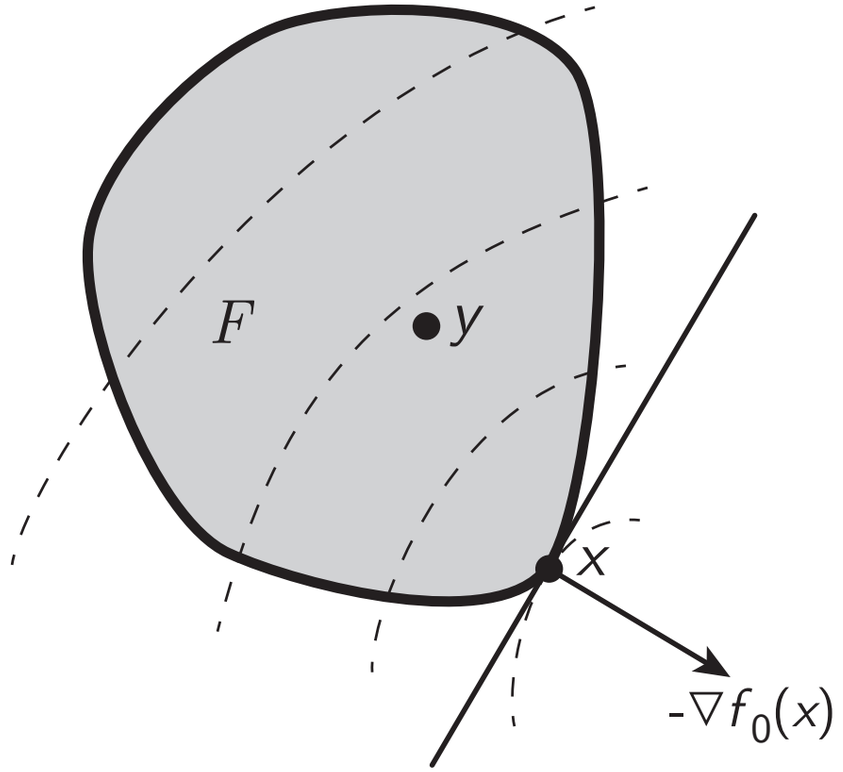
Optimality Criterion for Differentiable f_0

For a convex problem with a differentiable objective function f_0, x is optimal *iff* it is feasible, *and*

$$\nabla f_0(x)^\top (y - x) \geq 0, \quad \text{for all feasible } y.$$

The condition $\nabla f_0(x) = 0$ is the familiar sufficient first-order optimality condition.

The expression above states that the gradient may be non-zero, as long as all other feasible points are not "downhill" from the optimum.



Optimality Criterion for Differentiable f_0

Unconstrained problem:

$$\min_x \quad f_0(x)$$

x is optimal *iff* $\quad x \in \mathrm{dom}(f_0), \quad \nabla f_0(x) = 0.$

Optimality Criterion for Differentiable f_0

Equality constrained problem:

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{subject to:} \quad & Ax = b \end{aligned}$$

x is optimal *iff* $\quad x \in \mathrm{dom}(f_0), \quad Ax = b, \quad \exists \nu : \nabla f_0(x) + A^\top \nu = 0.$

Optimality Criterion for Differentiable f_0

Minimization over non-negative orthant:

$$\begin{aligned} \min_x \quad & f_0(x) \\ \text{subject to:} \quad & x \geq 0 \end{aligned}$$

x is optimal *iff* $x \in \mathrm{dom}(f_0), \quad x \geq 0$

and

$$\begin{aligned} \nabla f_0(x)_i > 0 \quad &\Rightarrow \quad x_i = 0 \\ \nabla x_i > 0 \quad &\Rightarrow \quad f_0(x)_i = 0 \end{aligned}$$

Equivalent optimization problems

Two problems are (informally) called **equivalent** if the solution to one can be (easily) inferred from the solution to the other, and vice versa.

- **Introducing slack variables for linear inequalities:**

$$\begin{aligned} \min_{x} \quad & f_0(x) \\ \text{subject to:} \quad & A_i x \leq b_i \quad i = 1, \ldots, m \end{aligned}$$

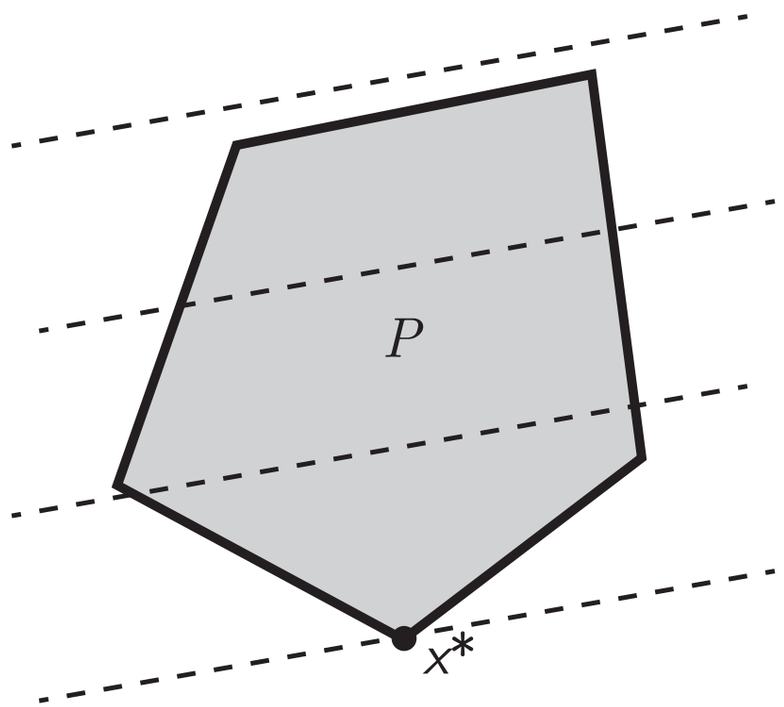
is equivalent to

$$\begin{aligned} \min_{x, s_i} \quad & f_0(x) \\ \text{subject to:} \quad & A_i x + s_i = b_i \quad i = 1, \ldots, m \\ & s_i \geq 0 \quad i = 1, \ldots, m \end{aligned}$$

A general linear program (LP)

Affine cost and constraint functions:

$$\begin{aligned} \min_{x} \quad & c^\top x + d \\ \text{subject to:} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$



An alternative format:

$$\begin{aligned} \min_{x} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Many problems can be rewritten (with some effort!) into LPs.

Example LPs

Constrained ℓ_∞ (Chebyshev) minimization:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & ||x||_\infty \\ \text{subject to:} \quad & Fx \leq g \end{aligned}$$

How can this be written as a linear program?

Example LPs

Constrained ℓ_1 minimization:

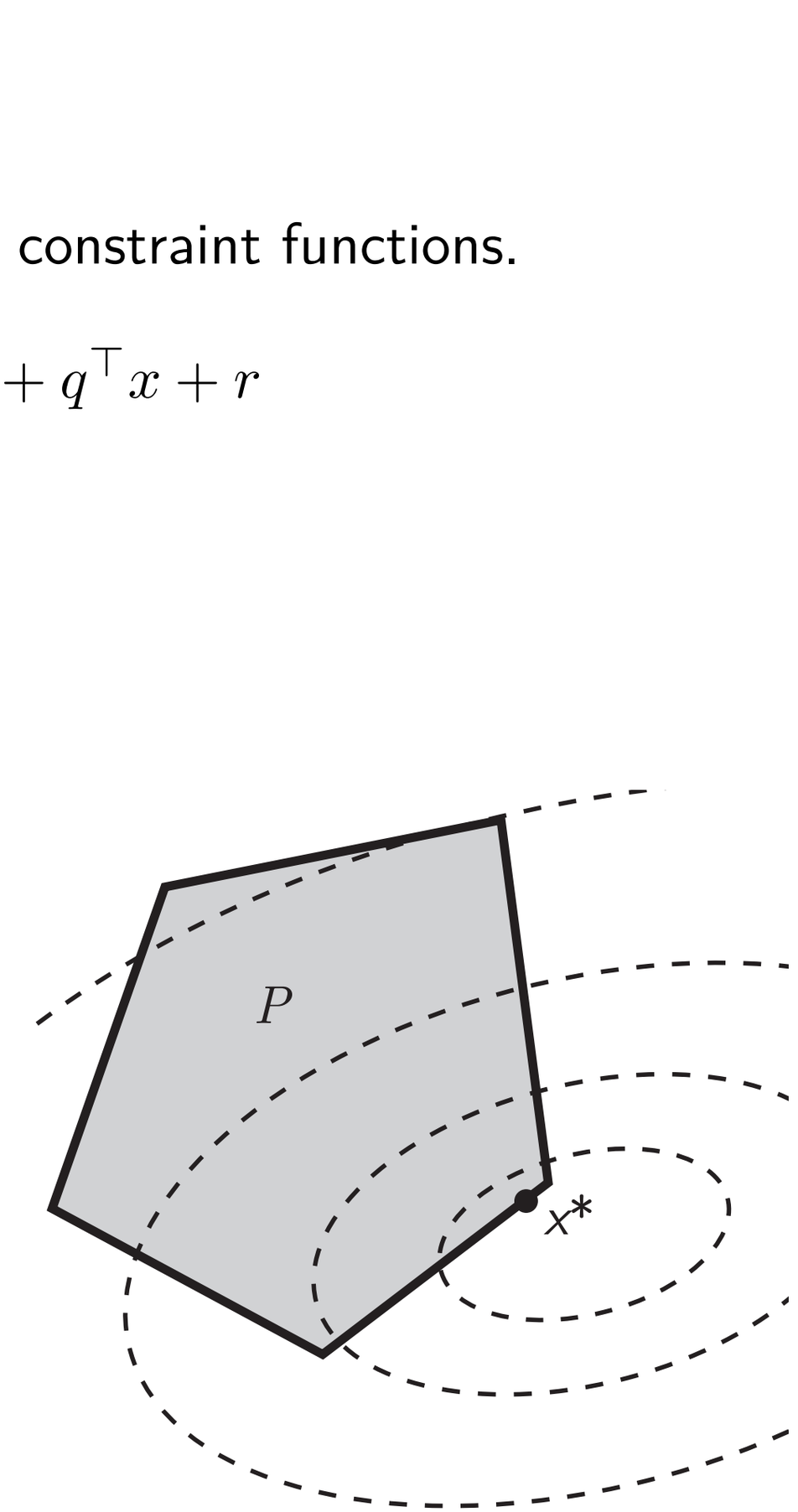
$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & ||Ax - b||_1 \\ \text{subject to:} \quad & Fx \leq g \end{aligned}$$

How can this be written as a linear program?

A general quadratic program (LP)

Quadratic cost function with $P \in \mathbb{S}^n_+$, affine constraint functions.

$$\begin{aligned} \min_x \quad & \tfrac{1}{2} x^\top P x + q^\top x + r \\ \text{subject to:} \quad & Gx \leq h \\ & Ax = b \end{aligned}$$



Example QPs

Linear program with random cost:

$$\begin{aligned} \min_x \quad & \mathbb{E}[c^\top x] + \gamma \mathrm{var}(c^\top x) \\ \text{subject to:} \quad & Gx \geq h \\ & Ax = b \end{aligned}$$

- Random cost function vector c with mean $\overline{c}$ and covariance Σ
- Objective is expected cost plus a "risk premium" γ on the variance.
- Large γ means large risk aversion — we prefer a small variance to the lowest expected cost.

How can this be written as a quadratic program?

Example QPs

Tikhonov Regularization: Least squares with extra penalty for nonzero terms.

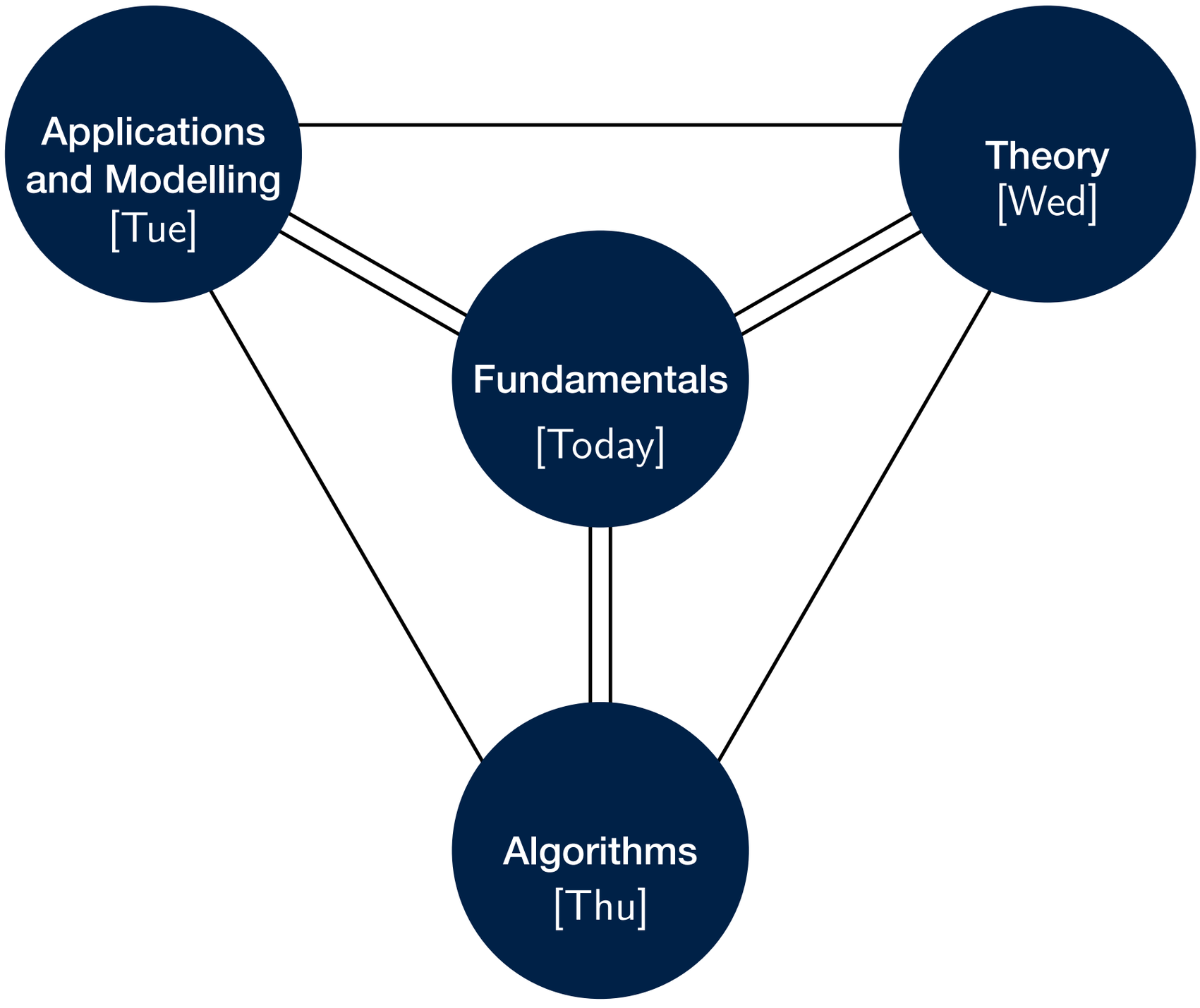
$$\min_{x \in \mathbb{R}^n} ||Ax - b||_2^2 + \gamma \cdot ||x||_1$$

- A larger penalty γ will tend to produce sparser solutions.
- Must convert an *unconstrained* problem into a larger *constrained* one to get it into standard QP form.
- Requires $\gamma \geq 0$ for convexity.

How can this be written as a quadratic program?

Day 2 : Applications

This course:


Applications and Modelling [Tue]
Theory [Wed]
Fundamentals [Today]
Algorithms [Thu]

Data Fitting

(Approximately) solving linear equations

Let's look at solving the system of equations

$$Ax \approx b$$

when **no exact solution** is possible. This the as the basis of the standard **least squares fitting problem.**

Usually (but not always) there are more equations than variables.

Least squares approach

First idea : minimise the sum of the squares of the errors:

$$\min \|Ax - b\|^2$$

The solution is

$$x^* = (A^T A)^{-1} A^T b$$

Least squares approach

Why is least squares a good idea?

- **Geometric** : Ax^* is the point closest to b in the image of A.

- **Model fitting** : Given measurements $y = Ax + w$, with y the measurements and w measurement noise, provides best guess for x.

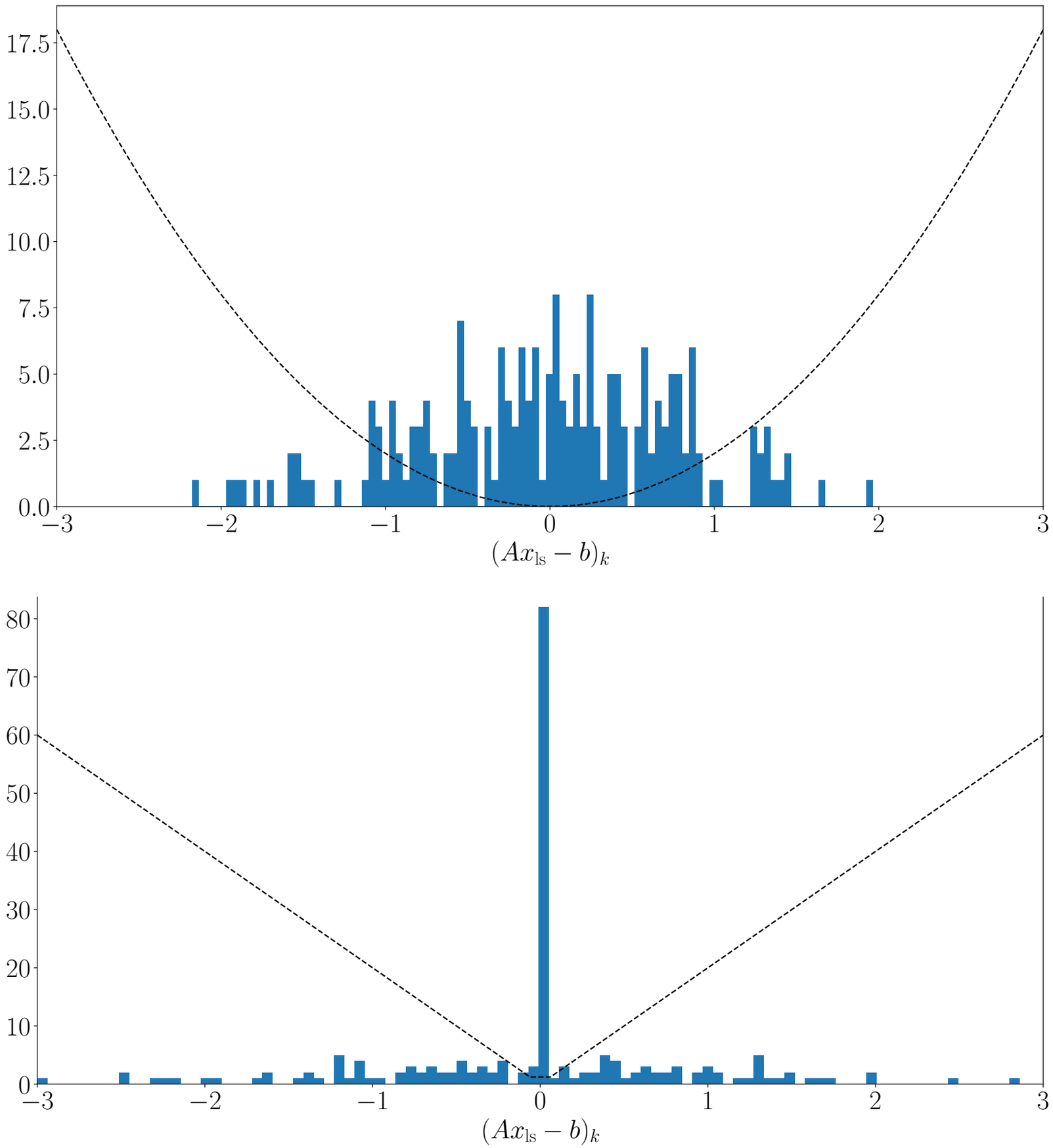
- **Optimal design** : If x are decision variables, Ax is the outcome of a set of design choices, and b is a desired outcome, then $x*$ is the design that best approximates the result.

What happens if we minimise a different norm?

$$\min \|Ax - b\|_1$$

This is now a **linear program** (after rewriting). There is no analytical solution anymore.

Residual minimisation in different norms



Many other possibilities…

$$\text{min} \;\; \phi(Ax - b)$$

- ℓ_1 **norm:**

$$\phi(y) = \sum_i |y_i|$$

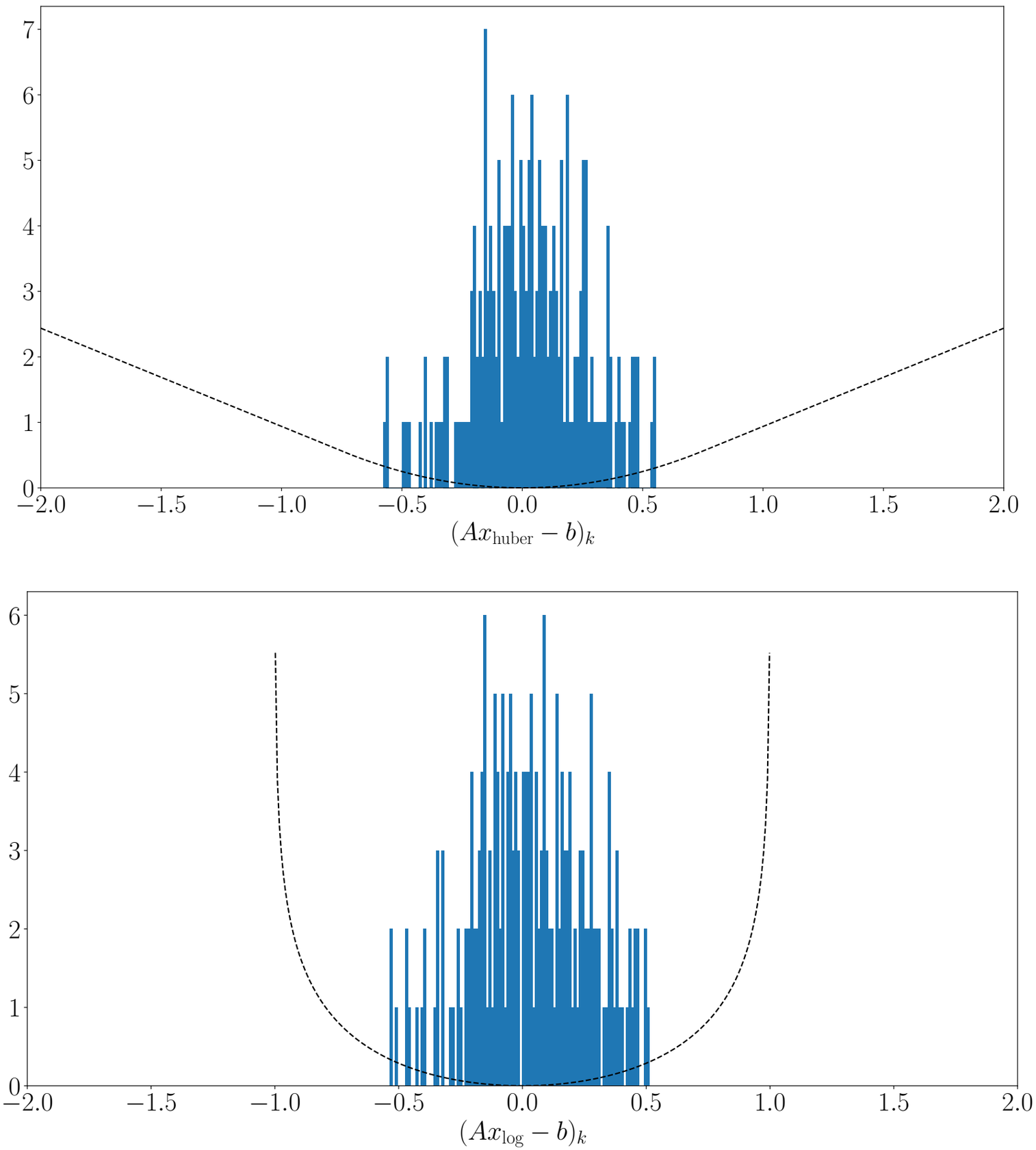
- **Huber penalty:**

$$\phi(y) = \sum_i h(y_i), \qquad h(u) = \begin{cases} u^2 & |u| \leq M \\ M(2|u| - M) & |u| \geq M \end{cases}$$

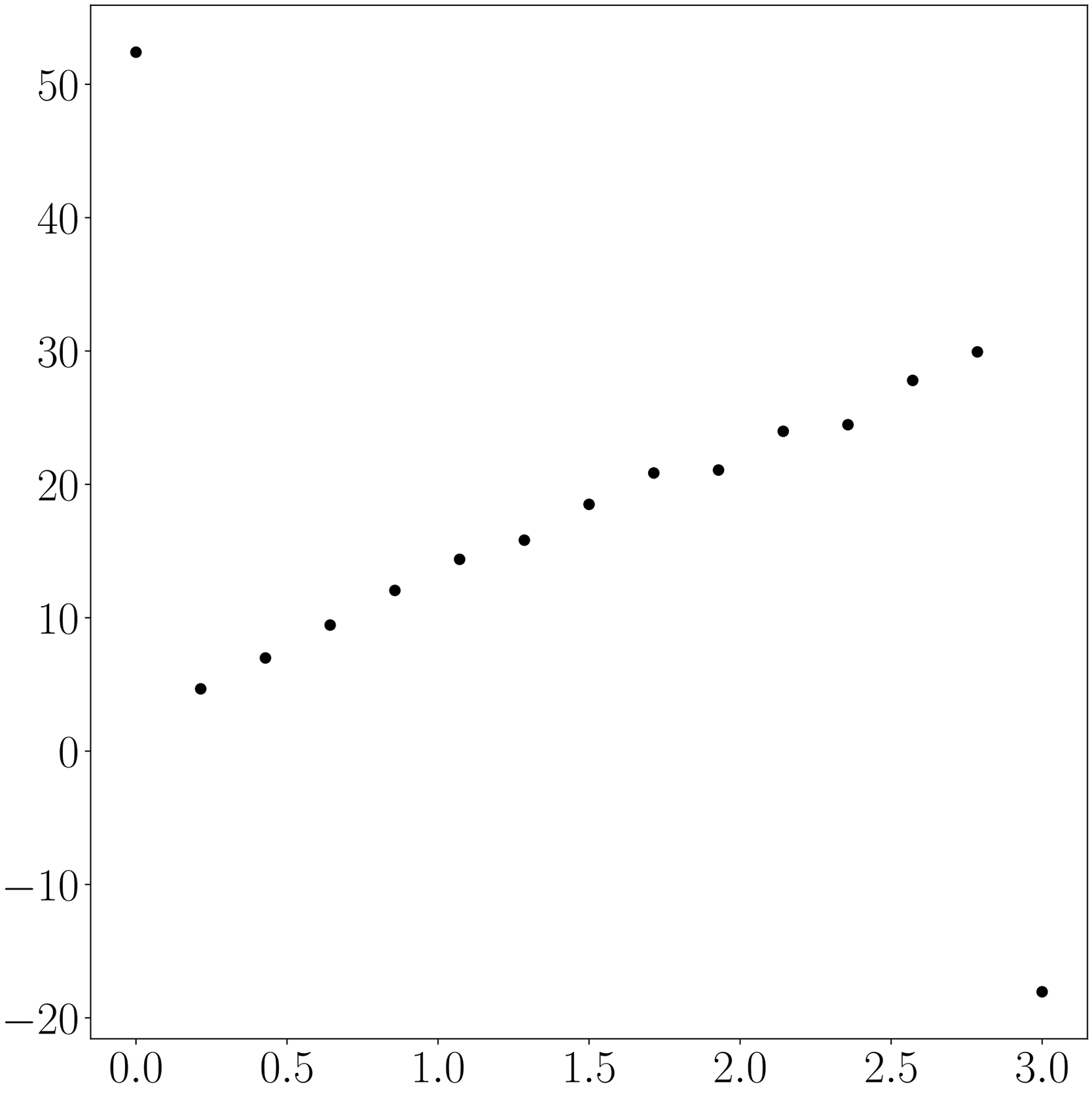
- **Log-barrier penalty:**

$$\phi(y) = \sum_i -\log(1 - y_i^2)$$

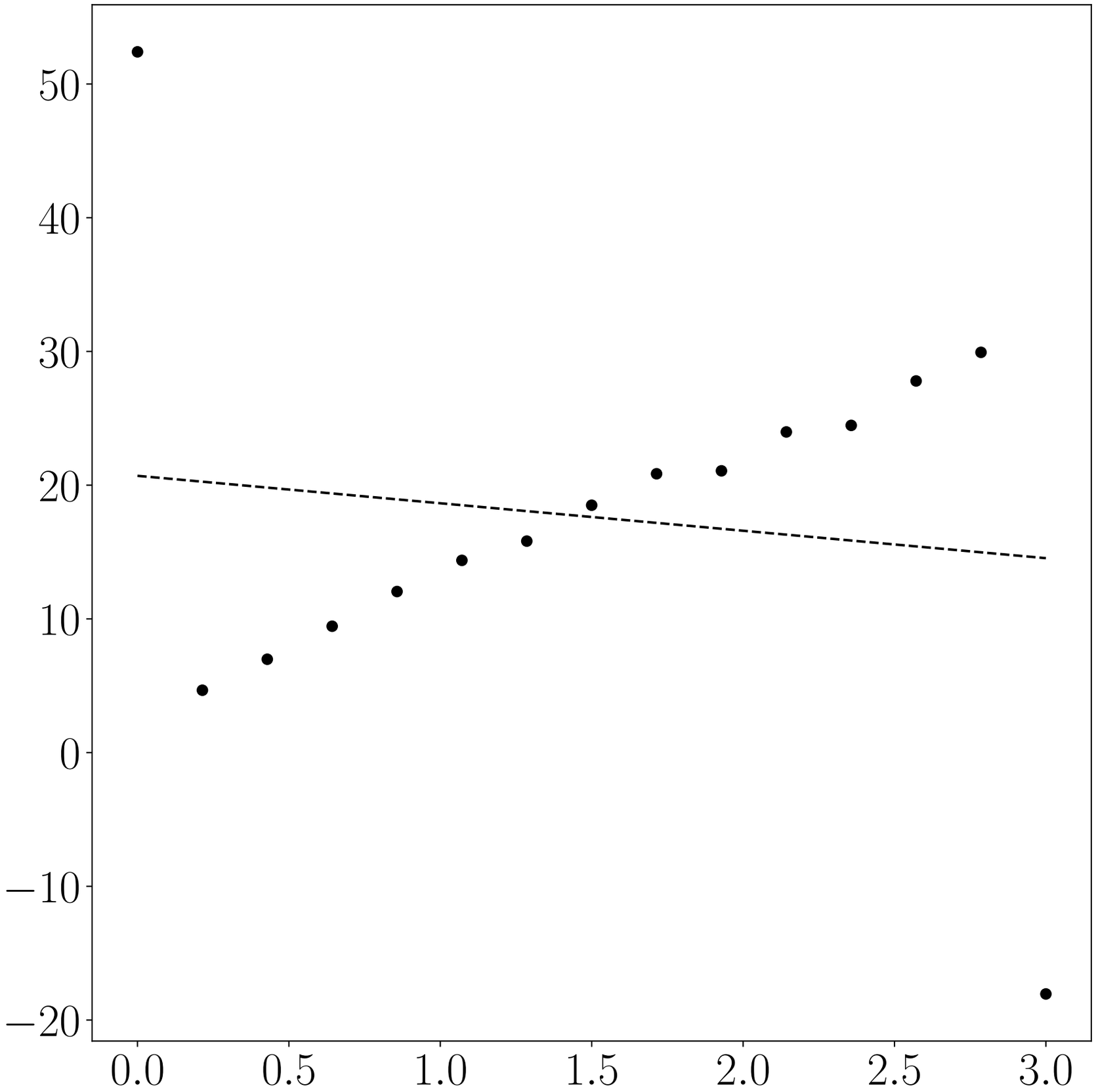
Residual minimisation with different penalties



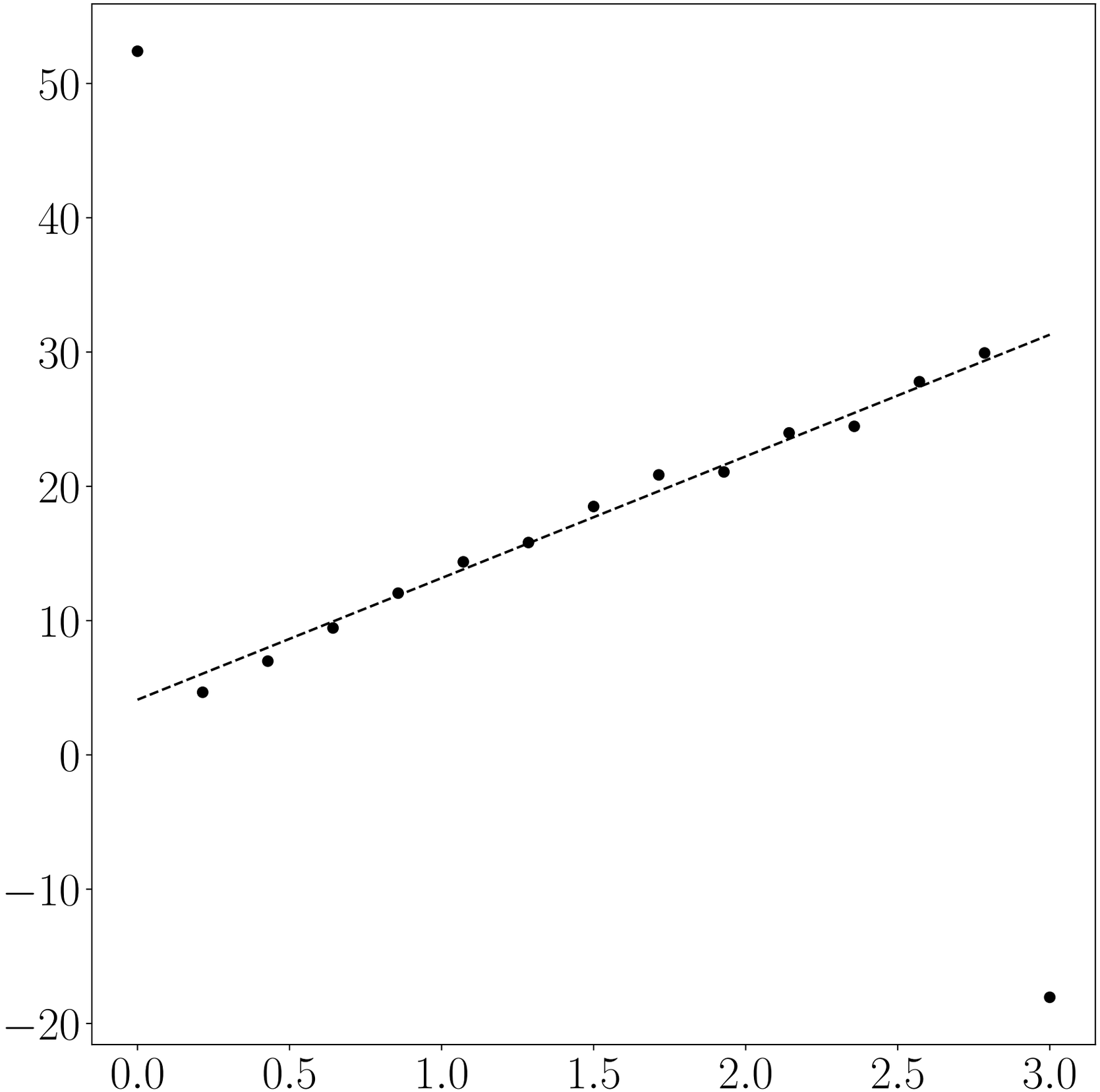
Protecting against outliers



Protecting against outliers : least squares



Protecting against outliers : Huber fitting



Nonnegative least squares

What if I only want to make non-negative design choices?

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|Ax - b\|^2 \\ \text{subject to:} \quad & x \geq 0 \end{aligned}$$

This is now a **constrained quadratic program (QP)**. There is *no analytical solution* anymore.

Sparse signal recovery ("fat" version)

Suppose that $\hat{x}$ is an unknown signal, known to be sparse.

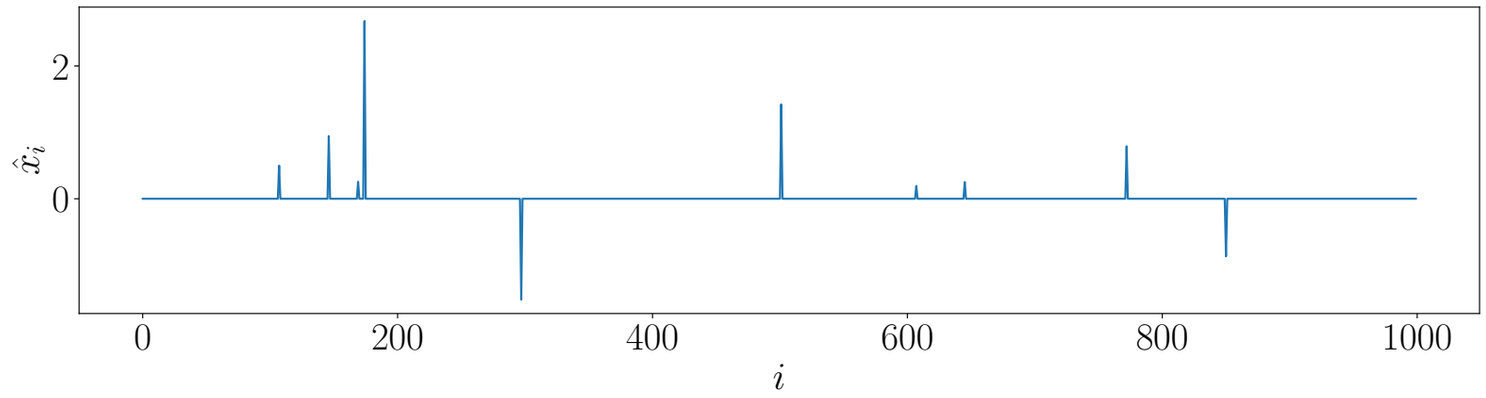
We make linear measurements $y = A\hat{x}$ with $A \in \mathbb{R}^{m \times n}$ and $m < n$.

Compute the signal with the smallest ℓ_1 norm, consistent with the measurements:

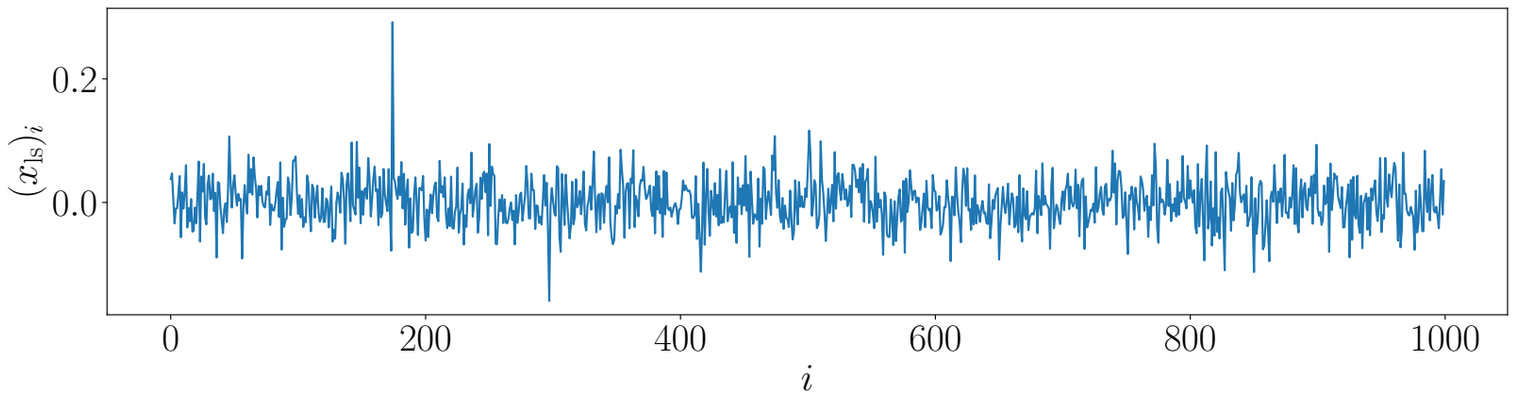
$$\begin{aligned} \text{min} \quad & \|x\|_1 \\ \text{subject to:} \quad & Ax = y \end{aligned}$$

Sparse signal recovery (“fat” version)

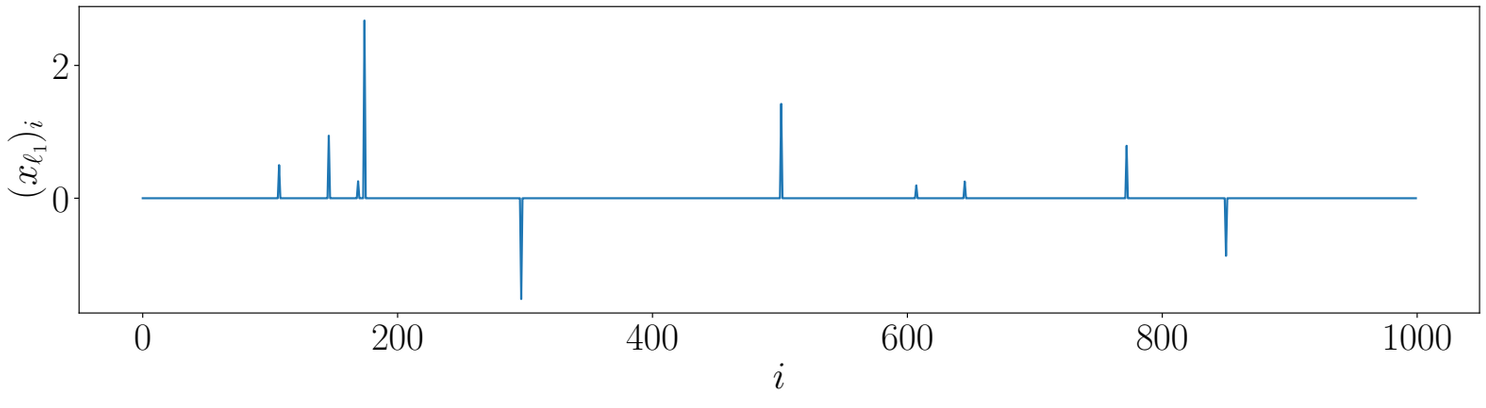
Exact signal : 10 nonzero components out of 1000. A is 100x1000



Least squares : can’t recover the sparse signal



1-norm estimate is **exact**



Sparse signal recovery ("tall" version)

What if I have more measurements than decisions, but still want a sparse decision?

$$\min \|Ax - b\|^2 + \lambda \|x\|_1$$

We can rewrite it as a QP:

$$\begin{aligned} \min_{x \in \mathbb{R}^n, \gamma t \in \mathbb{R}^n} \quad & \|Ax - b\|^2 + \mathbf{1}^T t \\ \text{subject to:} \quad & -t \leq x \leq t \end{aligned}$$

This tends to produce progressively sparser solutions as $\gamma \to \infty$.

Linear Classifiers / SVM

Support Vector Machines

Problem : Given a collection of data points in $\mathbf{R}^n$ with associated labels, find the separating hyperplane (i.e. a linear classifier) between two types.

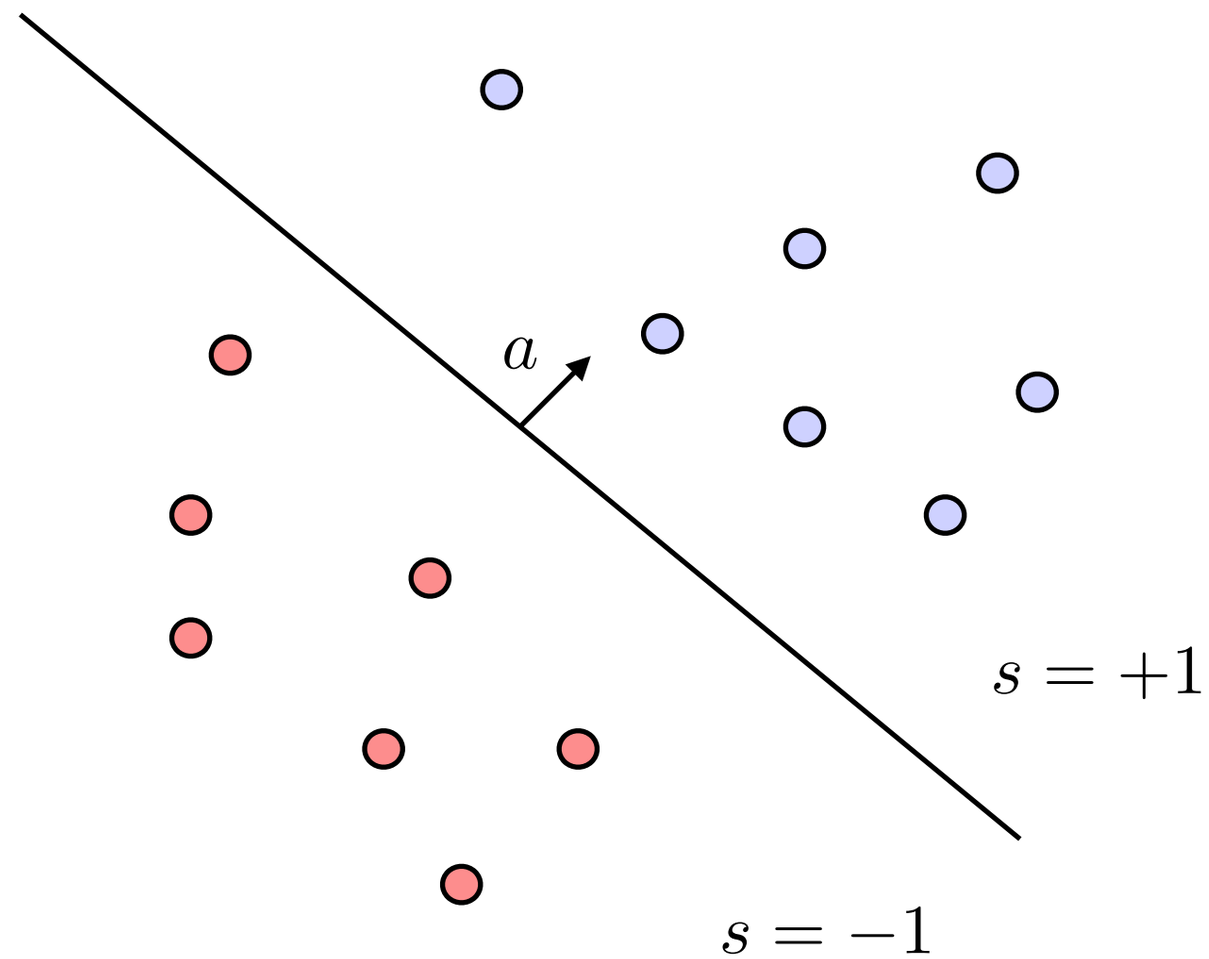
Each point should be on the side of the plane dictated by its label:

$$a^T v_i + b > 0 \quad \text{if} \quad s_i = 1$$
$$a^T v_i + b < 0 \quad \text{if} \quad s_i = -1$$

$\Leftrightarrow$

Every point should satisfy:

$$s_i(a^T v_i + b) \geq 1$$



Support Vector Machines (perfectly separable)

Problem : Given a collection of data points in $\mathbf{R}^n$ with associated labels, find the separating hyperplane (i.e. a linear classifier) between two types.

If the points are **completely separable**:

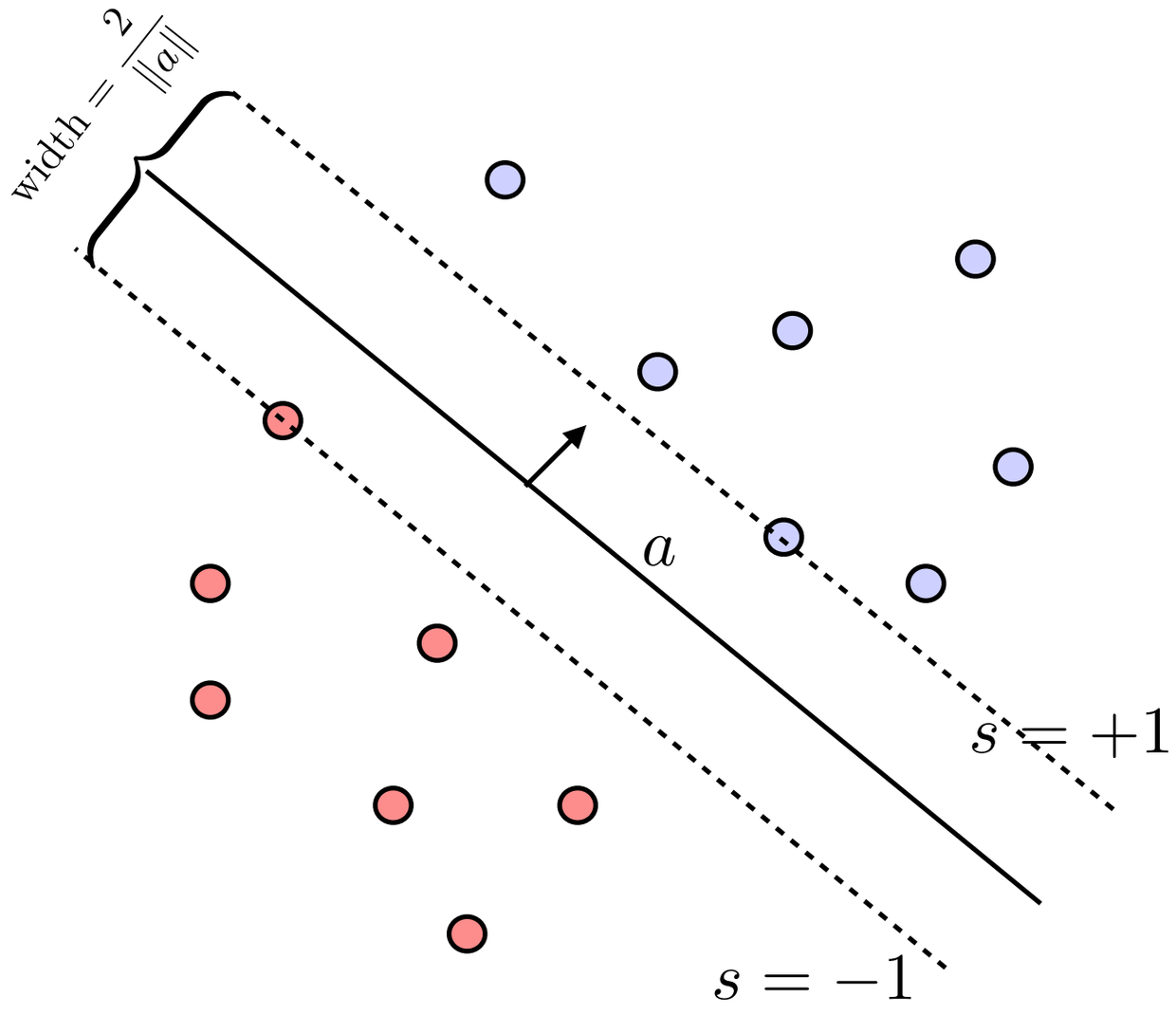
Every point should satisfy:

$$s_i(a^T v_i + b) \geq 1$$

Maximize the margin:

$$\min_{a,b} \quad \|a\|^2$$

subject to: $s_i(A^T v_i + b) \geq 1, \quad \forall i$



Support Vector Machines (not separable)

Problem : Given a collection of data points in $\mathbf{R}^n$ with associated labels, find the separating hyperplane (i.e. a linear classifier) between two types.

If the points are **not** completely separable:

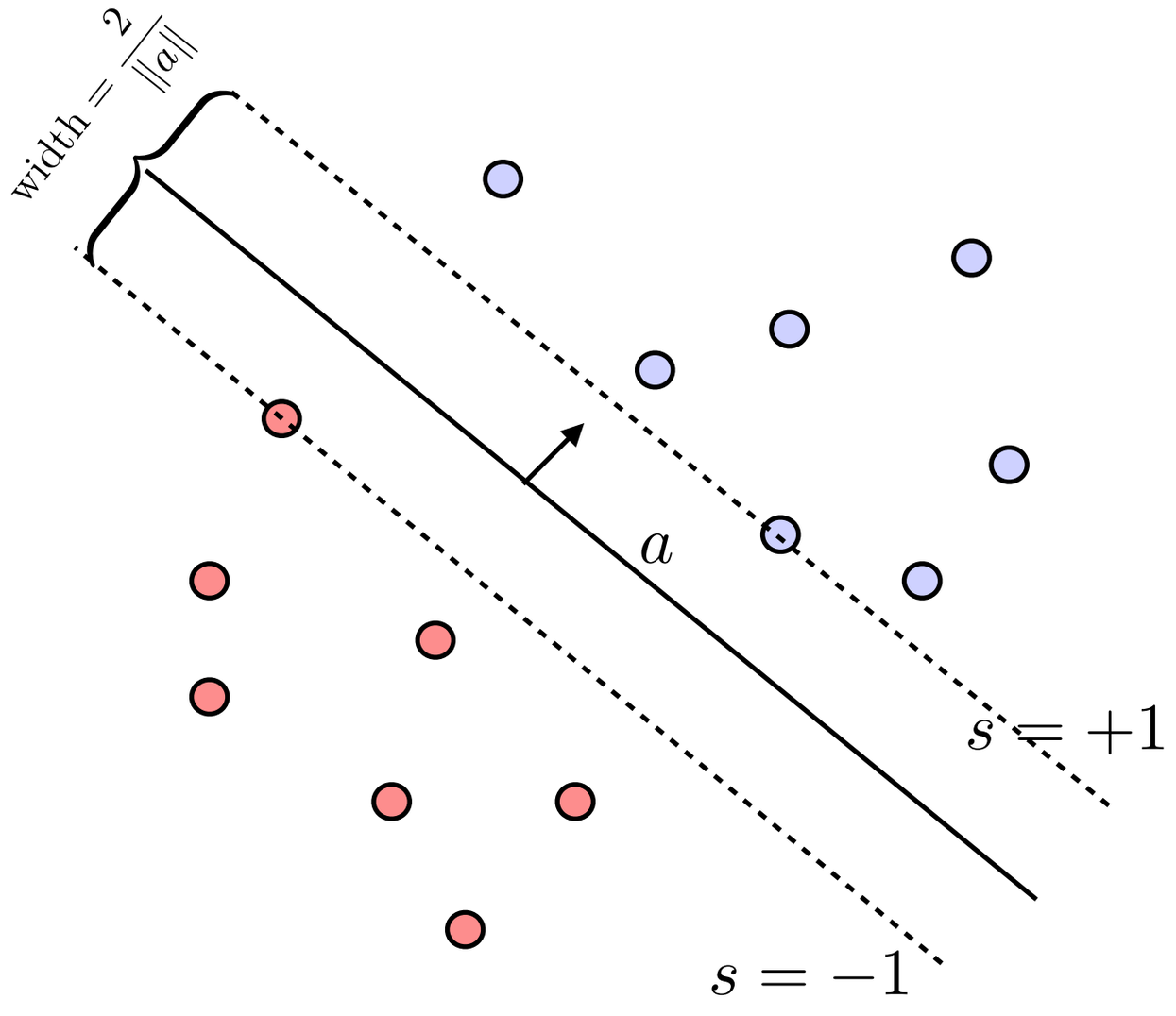
The violation of each point is :

$$\max\{0, 1 - s_i(a^T v_i + b)\}$$

This is the **hinge loss** function.

Minimise the violations:

$$\min_{a,b} \left[\sum_i \max\{0, 1 - s_i(a^T v_i + b)\} \right]$$



Support Vector Machines (soft margin)

Problem : Given a collection of data points in $\mathbf{R}^n$ with associated labels, find the separating hyperplane (i.e. a linear classifier) between two types.

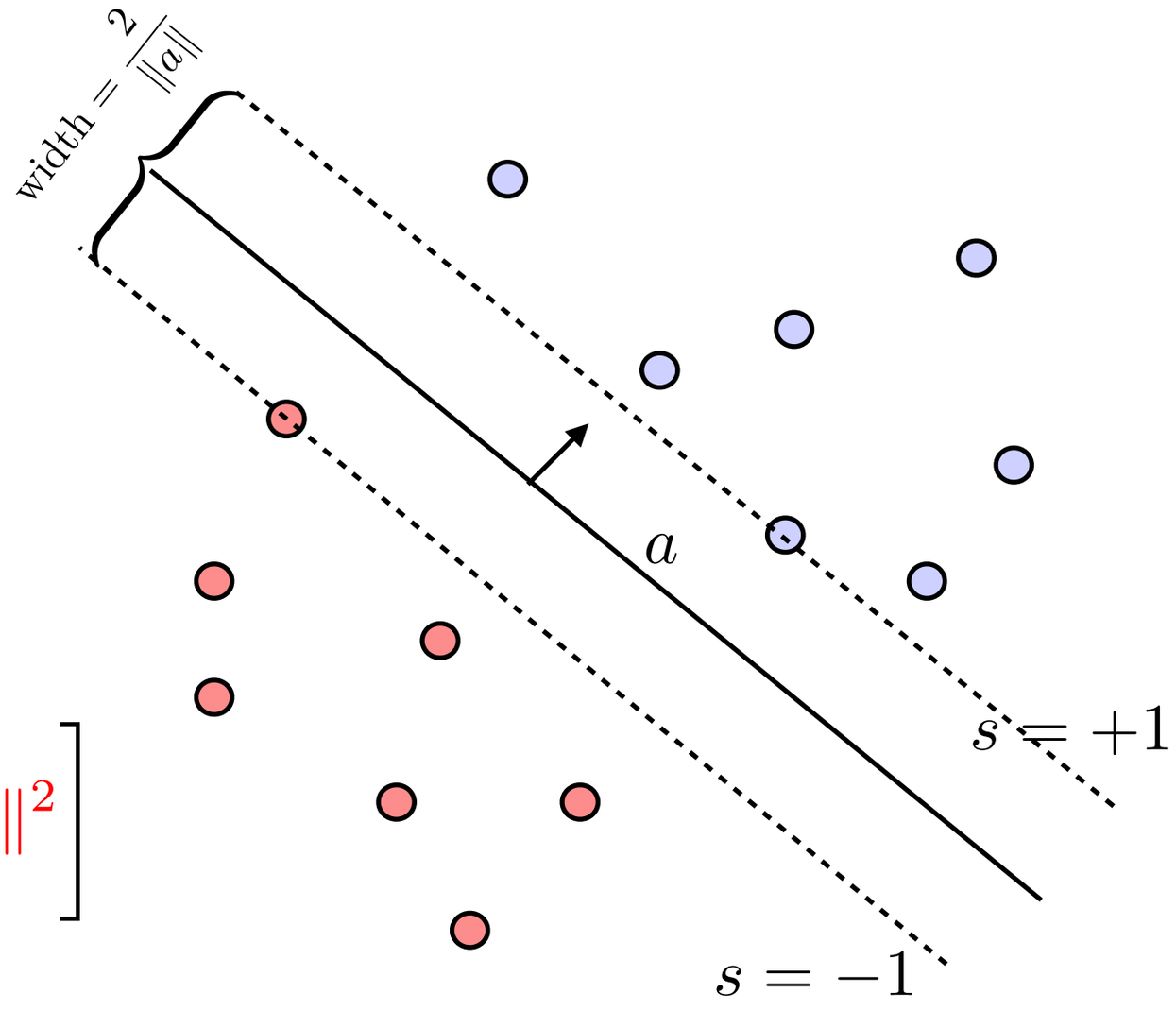
If the points are **not** completely separable:

The violation of each point is :

$$\max\{0, 1 - s_i(a^T v_i + b)\}$$

Minimise the violations:

$$\min_{a,b} \left[\sum_i \max\{0, 1 - s_i(a^T v_i + b)\} + \lambda \|a\|^2 \right]$$

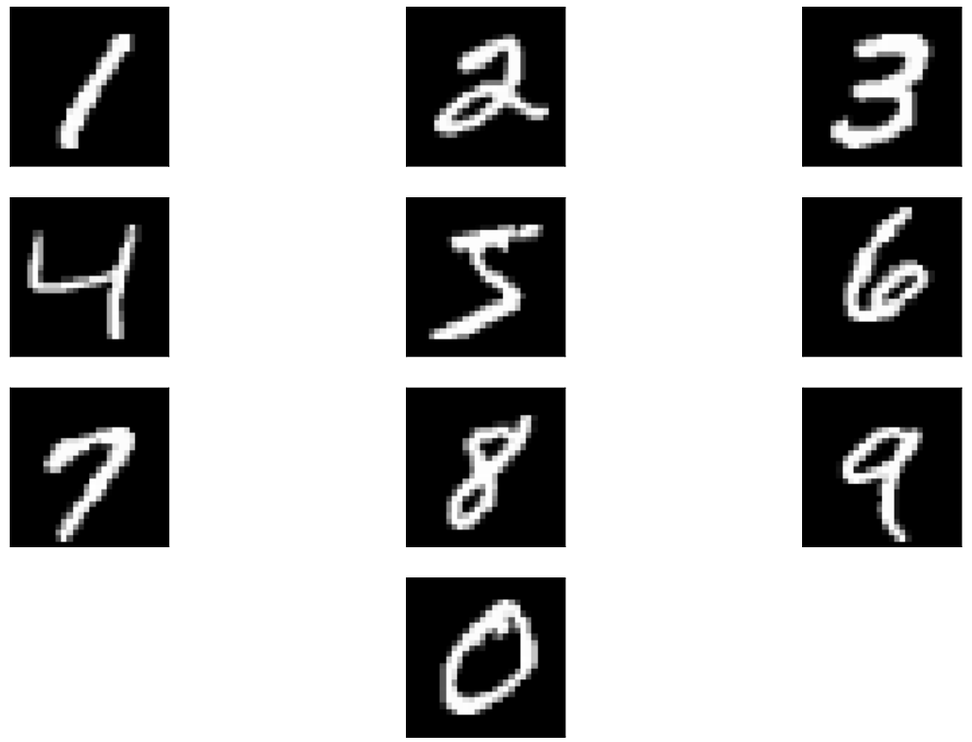


SVM Example

Character Recognition

MNIST data set of handwritten numerals

- Each character is 28 x 28 pixels
- 60k example images
- 10k further testing images
- Each sample comes with a label 0 — 9



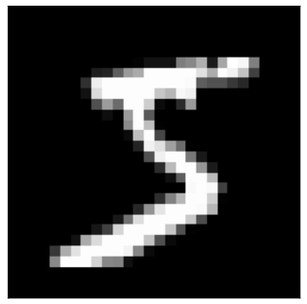
Goal

Use linear classification to identify handwritten numbers

Monochrome image representation

Images represented as an $m \times n$ matrix X

Each value X_{ij} represents a pixel's intensity ($0 =$ black, and $255 =$ white)

$X =$ 

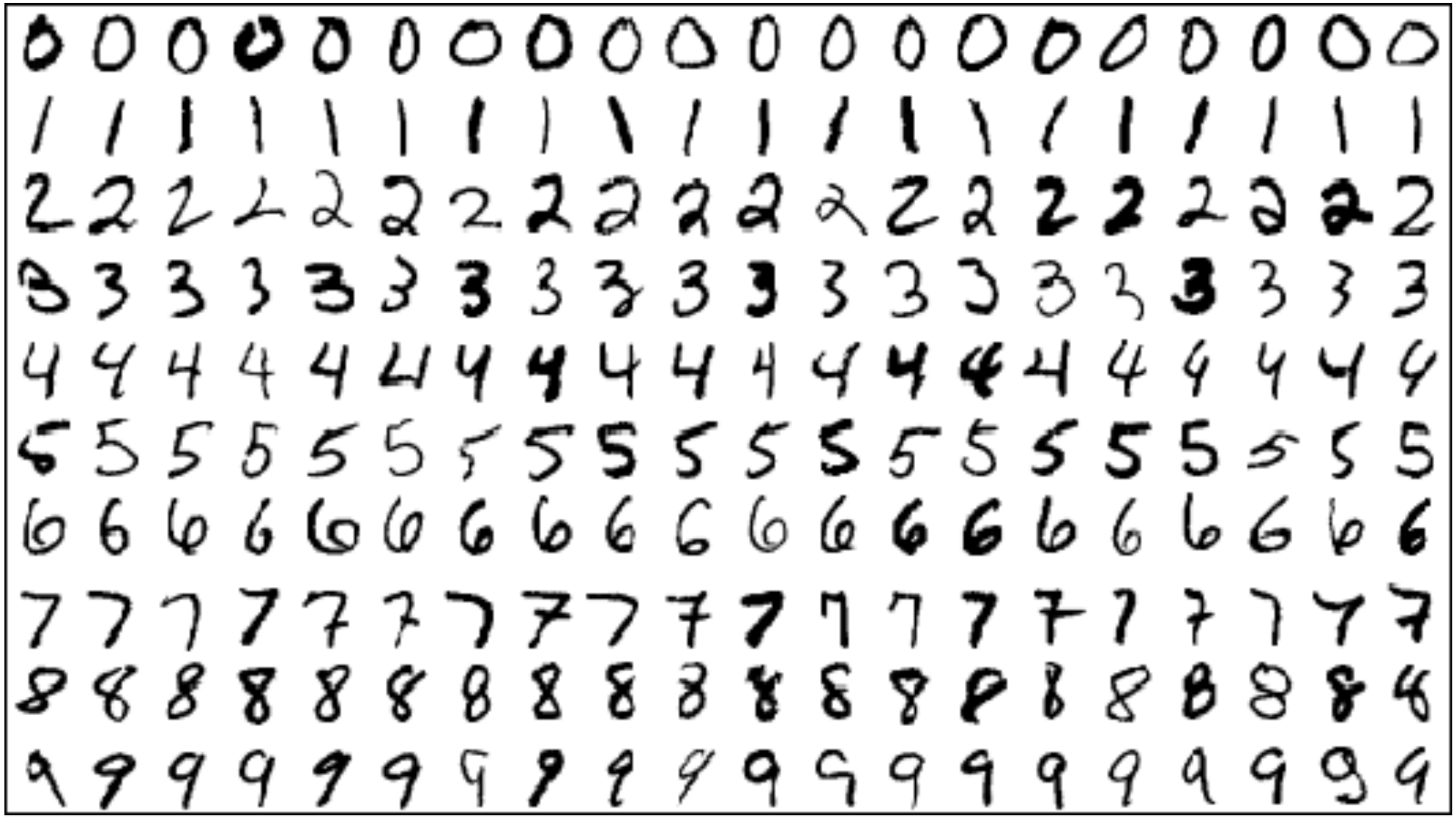
(in MNIST, $m = n = 28$)

We can represent an $m \times n$ matrix X by a single vector $x \in \mathbf{R}^{mn}$

$$X_{ij} = x_k, \qquad k = m(j-1) + i$$

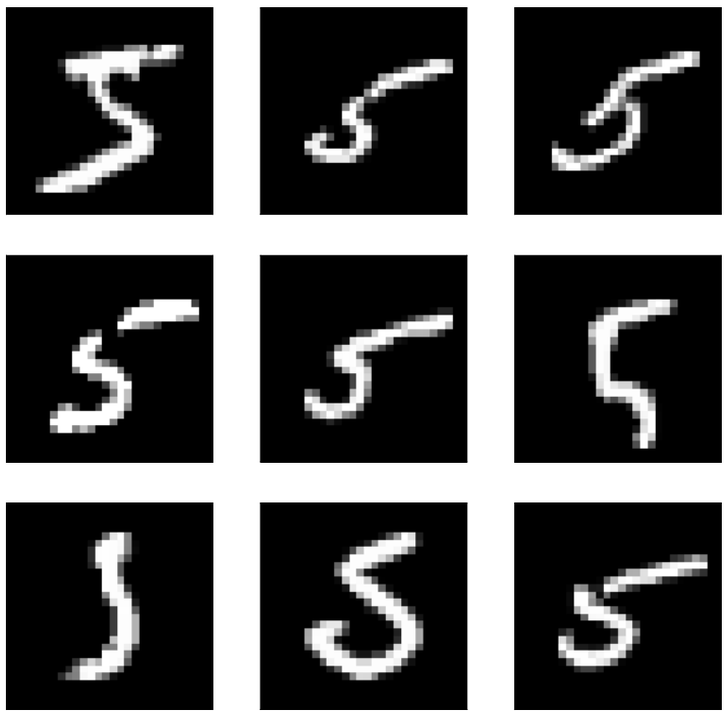
$x =$

MNIST data set

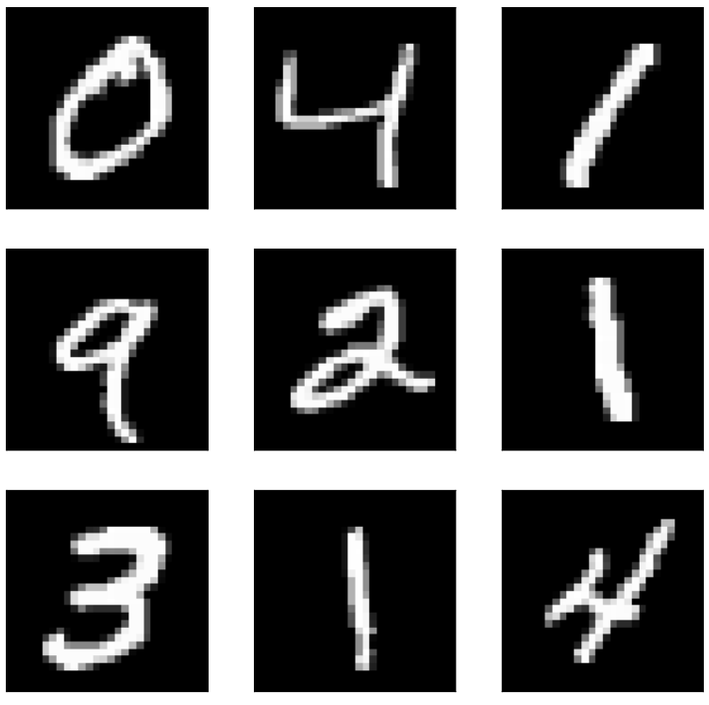


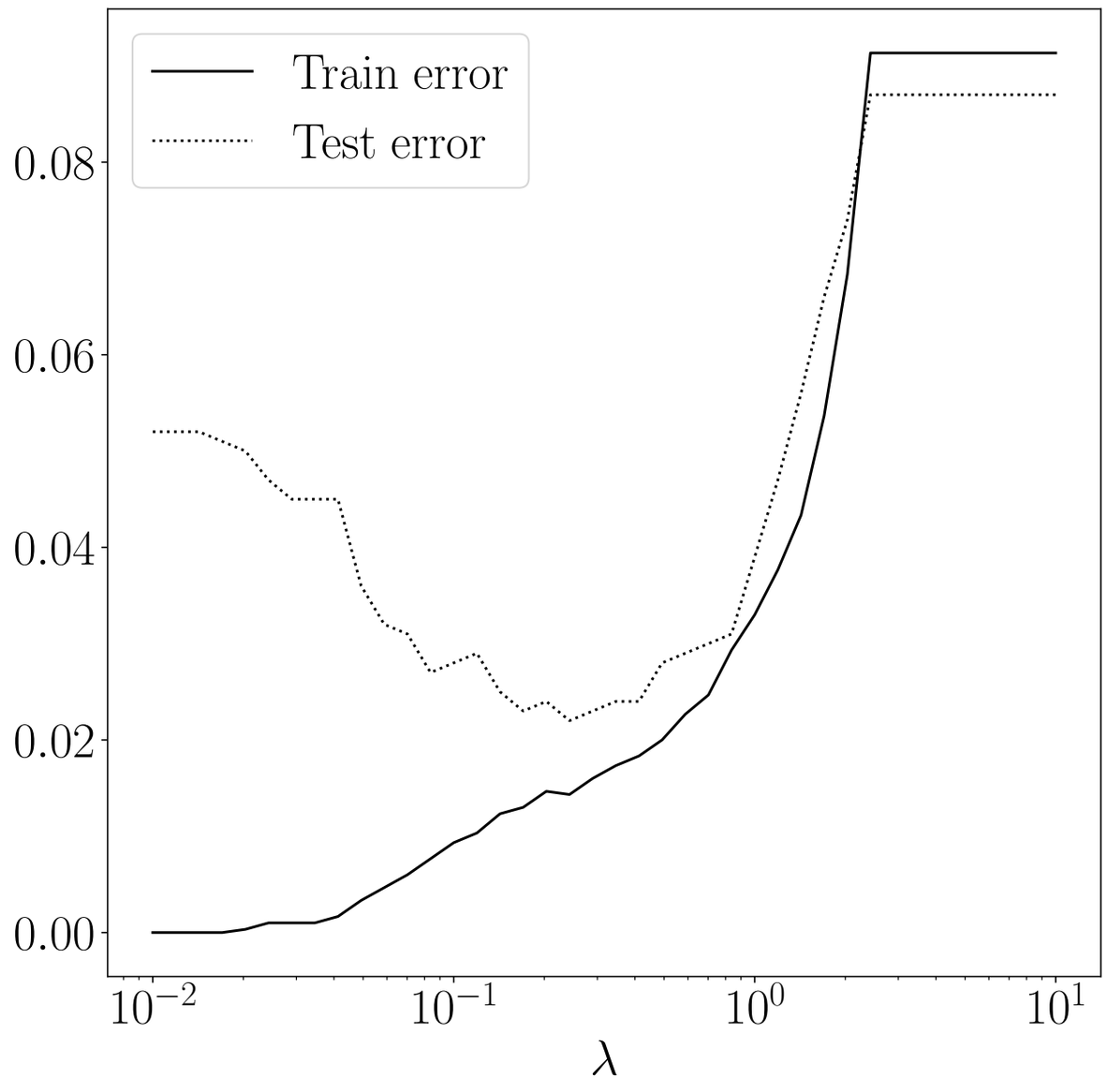
Learn to classify “5”

5



Not 5



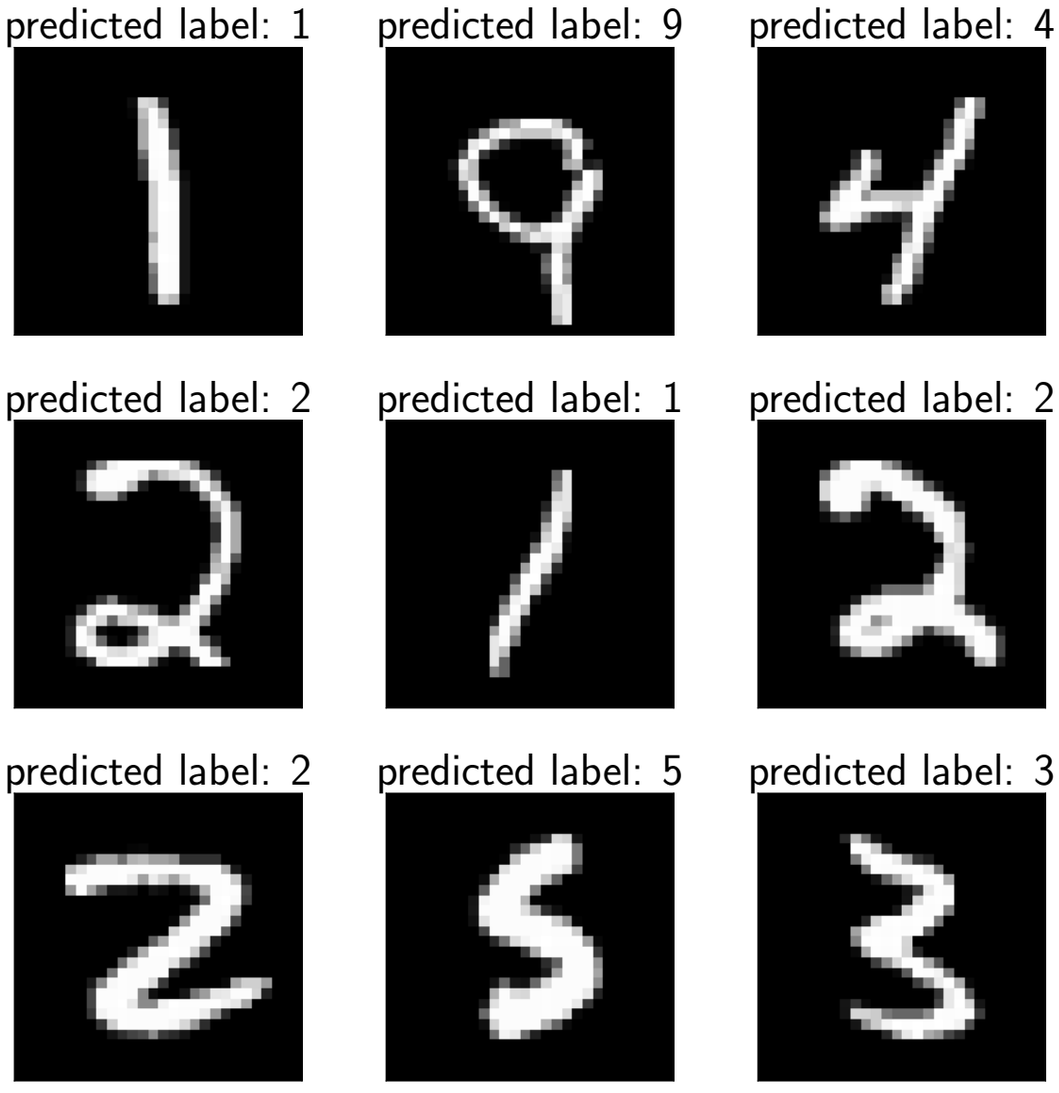


Multiclass classification

1. Train one classifier per label k
   (e.g., k vs anything else), obtaining (a_k, b_k)

2. Predict all results and take the maximum

$$\hat{y}^{(i)} = \text{argmax}_k \;\; a_k^T v^{(i)} + b_k$$



Optimal Control

Optimal Control Problems

Linear dynamical system

$$x_{t+1} = Ax_t + Bu_t, \quad t = 1, 2, \ldots$$
$$y_t = Cx_t, \quad t = 1, 2, \ldots$$

- x_t is the *state* at time t
- u_t is the *input*
- y_t is the *output*
- A is the *dynamics matrix*

The problem

- The *initial state* $x_1 = x^{\text{init}}$ is given

- **Goal.** Choose $u_1, u_2, \ldots, u_{T-1}$ to achieve some goals, e.g.,
  - Get to desired final state $x_T = x^{\text{des}}$
  - Minimize the input effort (make $\|u_t\|$ small for all t)
  - Track desired output y_t^{des} (make $\|y_t - y_t^{\text{des}}\|$ small)

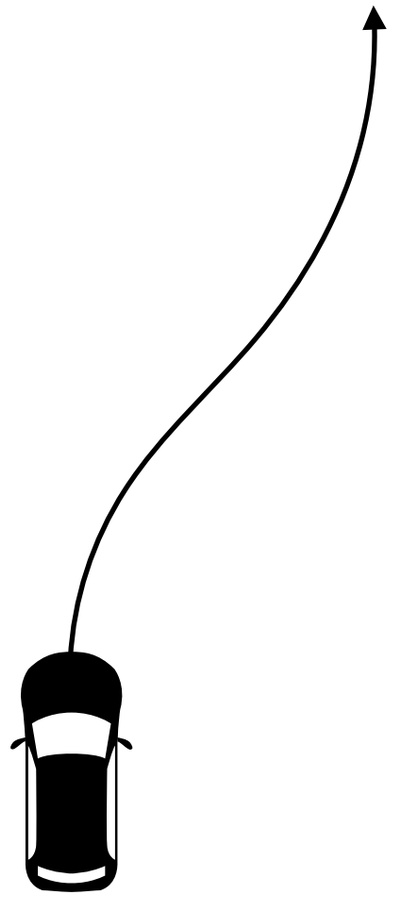
Vehicle trajectory optimization

Sample position and velocity at times $\tau = 0, h, 2h, \ldots$

Vehicle with mass m

Vehicle with mass m

- 2-vector p_t is the position at time ht
- 2-vector v_t is the velocity at time ht
- 2-vector u_t is the force applied at time ht
- $-\eta v_t$ is the friction force applied at ht



Small time interval h

$$\frac{p_{t+1} - p_t}{h} \approx v_t$$
$$m\frac{v_{t+1} - v_t}{h} \approx -hv_t + u_t$$

$$\longrightarrow$$

$$p_{t+1} = p_t + hv_t$$
$$v_{t+1} = (1 - h\eta/m)v_t + (h/m)u_t$$

Classical optimal control

$$
\begin{array}{ll}
\text{minimize} & \sum_{t=1}^{T} \|y_t - y_t^{\text{des}}\|^2 + \rho \sum_{t=1}^{T-1} \|u_t\|^2 \\
\text{subject to} & x_{t+1} = Ax_t + Bu_t, \quad t = 1, \ldots, T-1 \\
& y_t = Cx_t, \quad t = 1, \ldots, T \\
& x_1 = x^{\text{init}}
\end{array}
$$

Remarks

- The variables are $x_2, \ldots, x_T$, $y_2, \ldots, y_T$, and $u_t \ \ldots, u_{T-1}$
- $\rho > 0$ trade offs between control effort and tracking error
- No inequalities means **analytical** solution is available

Constrained optimal control

$$
\begin{array}{ll}
\text{minimize} & \sum_{t=1}^{T} \|y_t - y_t^{\text{des}}\|^2 + \rho \sum_{t=1}^{T-1} \|u_t\|^2 \\
\text{subject to} & x_{t+1} = Ax_t + Bu_t, \quad t = 1, \ldots, T-1 \\
& y_t = Cx_t, \quad t = 1, \ldots, T \\
& \|u_t\|_\infty \leq u^{\text{max}}, \quad t = 1, \ldots, T-1 \\
& \|u_t - u_{t-1}\|_1 \leq s^{\text{max}}, \quad t = 1, \ldots, T-1 \\
& x_1 = x^{\text{init}}
\end{array}
$$

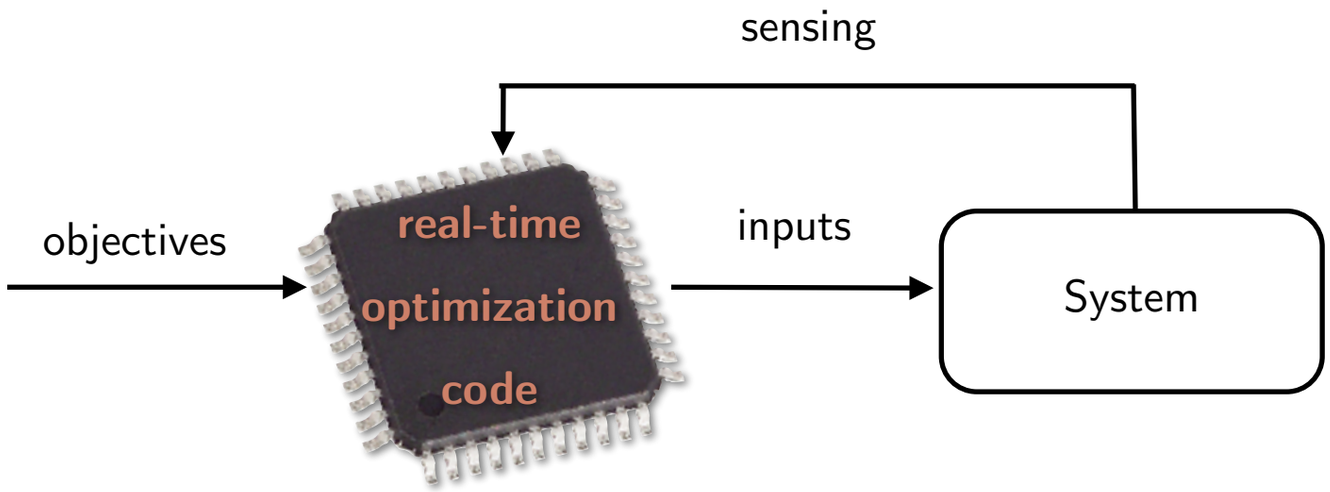
max-input

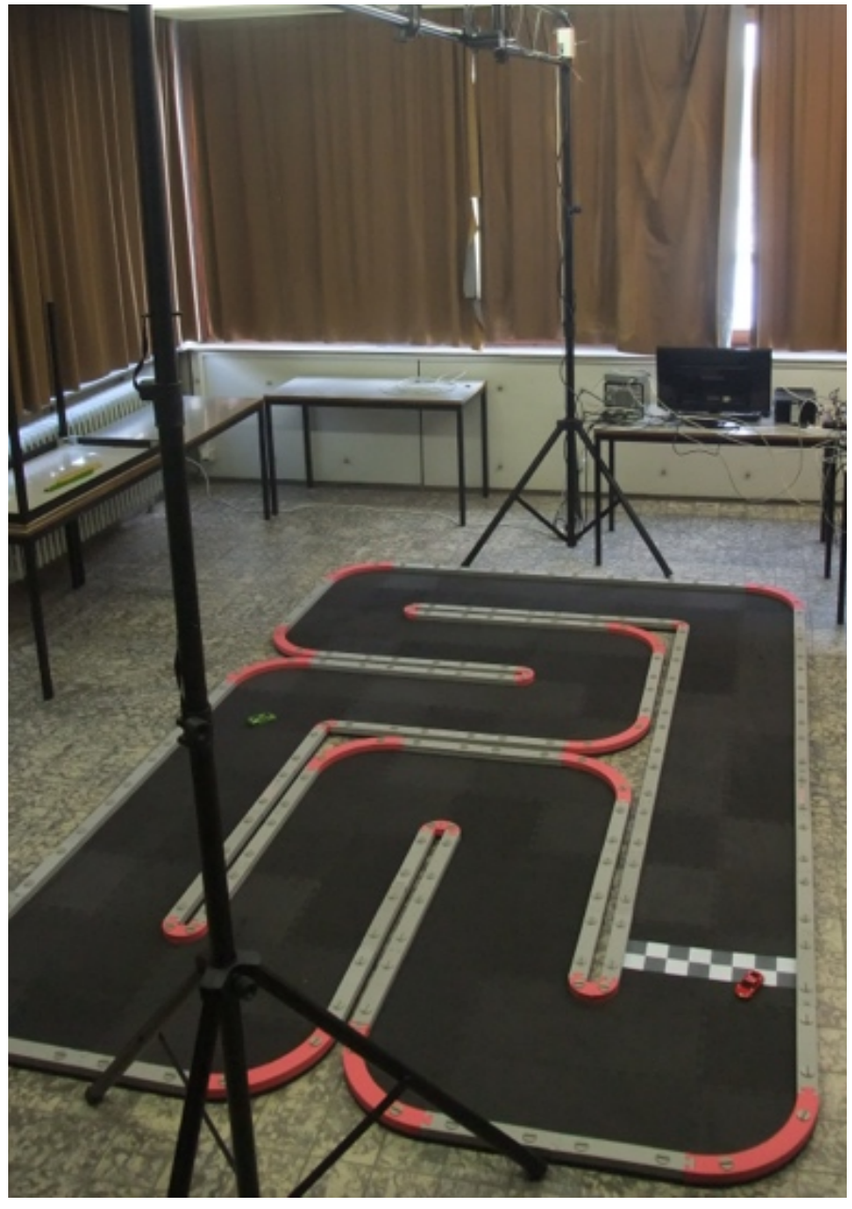
max-input variation

Remarks

- Could add state constraints (e.g. don't hit anything....)
- Could add energy management constraints for EVs
- Solution is re-computed and applied **in real-time**

Optimization-in-the-loop for real-time control systems





- Project : automatic racing on 1:43 scale race track
- Computing Platform : ARM Cortex A9

Autonomous Racing : Drift Control

Autonomous Racing : Obstacle Avoidance

Portfolio Optimization

Sudoku

Sudoku as an integer programming problem

Sudoku is a **constraint satisfaction** problem.

Has **binary** decisions :

$$x_{v,r,c} \in \{0,1\} \text{ for}$$

$$v \in 1 \ldots 9$$

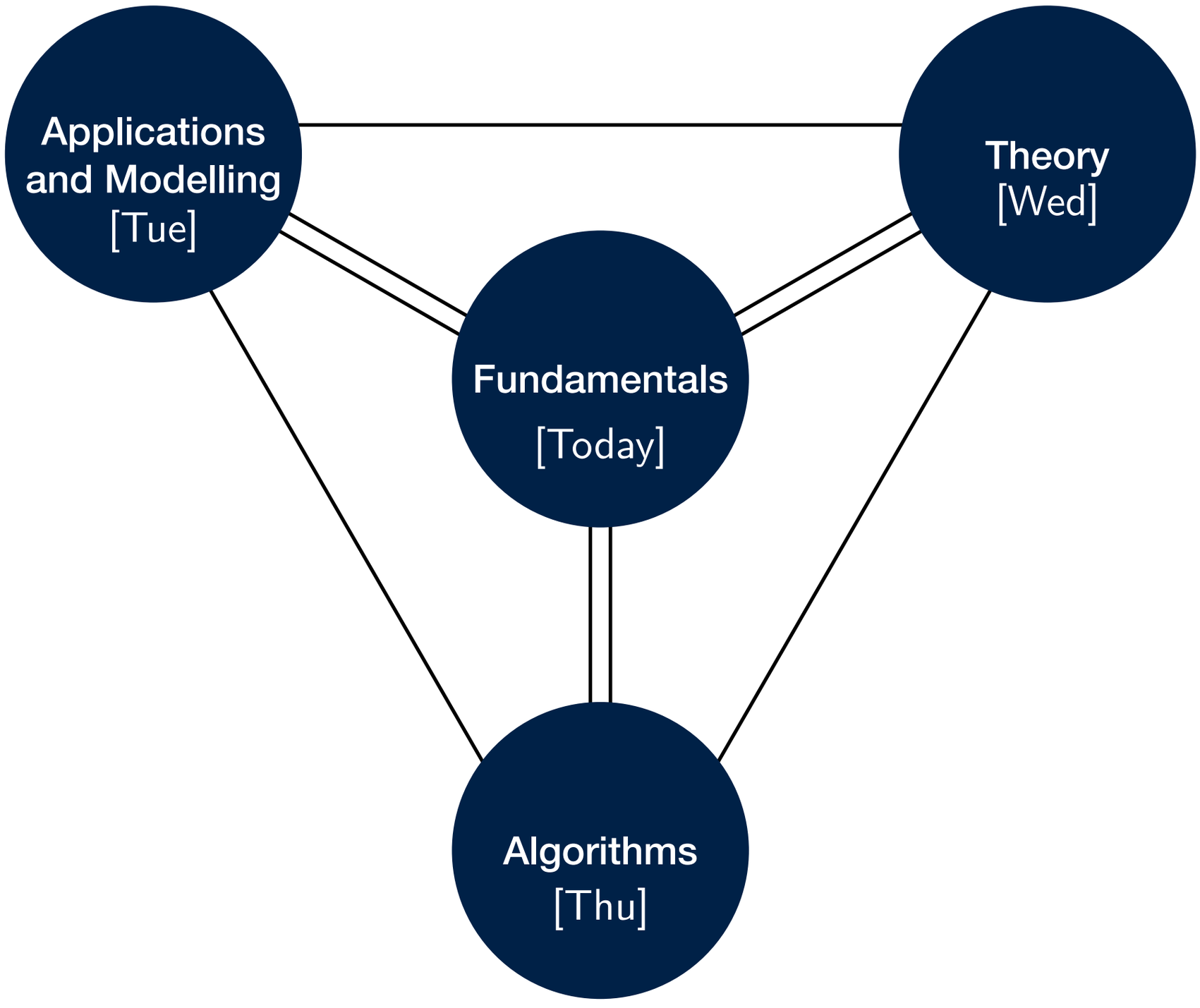
$$r = 1 \ldots 9$$

$$c = 1 \ldots 9$$

	1						8	
8		4				7		6
	3			9			1	
			7		8			
		5		1		8		
			3		4			
	4			2			6	
5		2				3		9
	9						5	

Day 3 : Duality

This course:


Applications and Modelling [Tue]
Theory [Wed]
Fundamentals [Today]
Algorithms [Thu]

Lagrangian function

Recall our standard optimization problem:

$$(\mathcal{P}): \quad \begin{array}{lll} \min\limits_{x \in \mathcal{X}} & f_0(x) & \\ \text{subject to:} & f_i(x) \leq 0 & i = 1 \dots m \\ & h_i(x) = 0 & i = 1 \dots p \end{array}$$

with optimal value p^*.

Lagrangian Function: $L : \mathcal{X} \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$

$$L(x, \lambda, \nu) = f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x)$$

- λ_i : inequality Lagrange multiplier for $f_i(x) \leq 0$.
- ν_i : equality Lagrange multiplier for $h_i(x) = 0$.

Langrange dual function

The **dual function** $g : \mathbb{R}^m \times \mathbb{R}^p$ is

$$g(\lambda, \nu) = \inf_{x \in \mathcal{X}} L(x, \lambda, \nu)$$
$$= \inf_{x \in \mathcal{X}} \left[f_0(x) + \sum_{i=1}^{m} \lambda_i f_i(x) + \sum_{i=1}^{p} \nu_i h_i(x) \right]$$

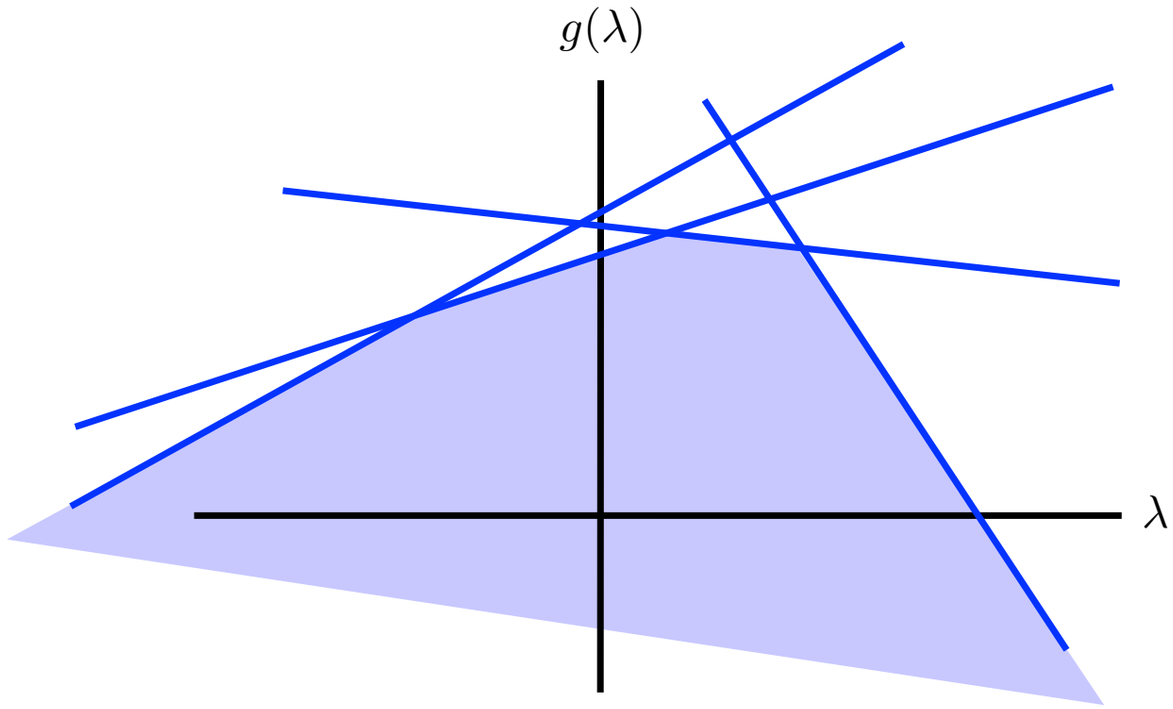
The dual function $g(\lambda, \nu)$:

- is always **concave**.
- generates lower bounds for p^*:

$$g(\lambda, \nu) \leq p^*, \qquad \forall (\lambda \geq 0, \nu \in \mathbb{R}^p)$$

- might be $-\infty$:

$$\operatorname{dom} g := \{\lambda, \nu \mid g(\lambda, \nu) > -\infty\}$$



Example: least-norm solution to linear equations

$$(\mathcal{P}): \quad \begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & x^\top x \\ \text{subject to:} & Ax = b \end{array}$$

The **Lagrangian** is $L(x, \nu) = x^\top x + \nu^\top (Ax - b)$.

Example: least-norm solution to linear equations

$$(\mathcal{P}): \quad \begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & x^\top x \\ \text{subject to:} & Ax = b \end{array}$$

The **Lagrangian** is $L(x, \nu) = x^\top x + \nu^\top (Ax - b)$.

Dual function:
Minimize the Lagrangian $L(x, \nu)$ by setting the gradient to zero:

$$\nabla_x L(x, \nu) = 2x + A^\top \nu = 0 \quad \Rightarrow \quad x = -\frac{1}{2} A^\top \nu.$$

Substitute back into L to get the dual function:

$$g(\nu) = -\frac{1}{4} \nu^\top A A^\top \nu - b^\top \nu \qquad \text{[a concave function]}$$

Lower bound property: $-\frac{1}{4} \nu^\top A A^\top \nu - b^\top \nu \leq p^*$ for every ν.

The dual problem

Every $\nu \in \mathbb{R}^p$, $\lambda \geq 0$ produces a lower bound for p^* using the dual function.

Which is the best?

$$(\mathcal{D}): \quad \begin{array}{ll} \max\limits_{\lambda,\nu} & g(\lambda, \nu) \\ \text{subject to:} & \lambda \geq 0 \end{array}$$

- Problem $(\mathcal{D})$ is **convex**, even if $(\mathcal{P})$ is not.
- Problem $(\mathcal{D})$ has optimal value $d^* \leq p^*$.
- The point (λ, ν) is **dual feasible** if $\lambda \geq 0$ and $(\lambda, \nu) \in \operatorname{dom} g$.
- Can often impose the constraint $(\lambda, \nu) \in \operatorname{dom} g$ explicitly in $(\mathcal{D})$.

Example : Dual of a Linear Program (LP)

$$(\mathcal{P}): \quad \begin{aligned} \min_{x \in \mathbb{R}^n} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b \\ & Cx \leq d \end{aligned}$$

The **dual function** is

$$g(\lambda, \nu) = \min_{x \in \mathbb{R}^n} \left[c^\top x + \nu^\top (Ax - b) + \lambda^\top (Cx - d) \right]$$

$$= \min_{x \in \mathbb{R}^n} \left[(A^\top \nu + C^\top \lambda + c)^\top x - b^\top \nu - d^\top \lambda \right]$$

$$= \begin{cases} -b^\top \nu - d^\top \lambda & \text{if } A^\top \nu + C^\top \lambda + c = 0 \\ -\infty & \text{otherwise} \end{cases}$$

Lower bound property: $-b^\top \nu - d^\top \lambda \leq p^*$ whenever $A^\top \nu + C^\top \lambda + c = 0$ and $\lambda \geq 0$.

Example : Dual of a Linear Program (LP)

$$(\mathcal{P}): \quad \begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & c^\top x \\ \text{subject to:} & Ax = b \\ & Cx \leq d \end{array}$$

The **dual problem** is

$$(\mathcal{D}): \quad \begin{array}{ll} \max\limits_{\lambda, \nu} & -b^\top \nu - d^\top \lambda \\ \text{subject to:} & A^\top \nu + C^\top \lambda + c = 0 \\ & \lambda \geq 0 \end{array}$$

The dual of a linear program is also a linear program.

Example : Norm minimization with equality constraint

$$(\mathcal{P}): \quad \begin{array}{ll} \min\limits_{x} & \|x\|_2 \\ \text{subject to:} & Ax = b \end{array}$$

The **dual function** is

$$g(\lambda) = \min_x \left[\|x\| - (A^\top \nu)^\top x + b^\top \nu\right]$$

$$= \begin{cases} b^\top \nu & \text{if } \|A^\top \nu\|_2 \leq 1 \\ -\infty & \text{otherwise} \end{cases}$$

The **dual problem** is

$$(\mathcal{D}): \quad \begin{array}{ll} \max\limits_{\nu} & b^\top \nu \\ \text{subject to:} & \|A^\top \nu\|_2 \leq 1 \end{array}$$

Lower bound property: $b^\top \nu \leq p^*$ whenever $\|A^\top \nu\|_2 \leq 1$.

Example: Dual of a quadratic program

A quadratic program (QP) with $Q \succ 0$:

$$(\mathcal{P}): \quad \begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \tfrac{1}{2} x^\top Q x + c^\top x \\ \text{subject to:} \quad & Cx \leq d \end{aligned}$$

The **dual function** is

$$g(\lambda) = \min_{x \in \mathbb{R}^n} \left[\frac{1}{2} x^\top Q x + c^\top x + \lambda^\top (Cx - d) \right]$$

$$= \min_{x \in \mathbb{R}^n} \left[\frac{1}{2} x^\top Q x + (c + C^\top \lambda)^\top x - d^\top \lambda \right]$$

The unconstrained minimization over x is convex for every λ. If $Q \succ 0$, then the optimal x satisfies

$$Qx + c + C^\top \lambda = 0$$

Example: Dual of a quadratic program (cont'd)

Substitute $x = -Q^{-1}(c + C^\top \lambda)$ into the dual function:

$$g(\lambda) = -\frac{1}{2}\left(c + C^\top \lambda\right)^\top Q^{-1}\left(c + C^\top \lambda\right) - d^\top \lambda$$

Dual of a QP:

The dual problem is to maximize $g(\lambda)$ over $\lambda \geq 0$, or equivalently,

$$(\mathcal{D}): \quad \begin{aligned} \min_{\lambda} \quad & \tfrac{1}{2}\lambda^\top C^\top Q^{-1} C\lambda + \left(CQ^{-1}c + d\right)^\top \lambda + \tfrac{1}{2}c^\top Q^{-1}c \\ \text{subject to:} \quad & \lambda \geq 0 \end{aligned}$$

NB: Dual of a QP is another QP.

Example : Dual of a Mixed-Integer Linear Program (MILP)

$$(\mathcal{P}): \quad \begin{aligned} \min_{x \in \mathcal{X}} \quad & c^\top x \\ \text{subject to:} \quad & Ax \leq b \\ & \mathcal{X} = \{-1, 1\}^n \end{aligned}$$

The **dual function** is

$$\begin{aligned} g(\lambda) &= \min_{x_i \in \{-1,1\}} \left[c^\top x + \lambda^\top (Ax - b) \right] \\ &= -\|A^\top \lambda + c\|_1 - b^\top \lambda \end{aligned}$$

The **dual problem** is

$$(\mathcal{D}): \quad \begin{aligned} \max_{\lambda} \quad & -\|A^\top \lambda + c\|_1 - b^\top \lambda \\ \text{subject to:} \quad & \lambda \geq 0 \end{aligned}$$

The dual of an MILP is an LP (without integers).

Weak and strong duality

Weak Duality

- It is **always** true that $d^* \leq p^*$.
- Sometimes the dual is much easier to solve than the primal (or vice-versa).
- Example: The dual of an MILP (difficult) is a standard LP (easy).

Strong Duality

- It is **sometimes** true that $d^* = p^*$.
- Strong duality usually holds for convex problems.
- Strong duality usually does not hold for non-convex problems.
- Can impose conditions on convex problems to guarantee that $d^* = p^*$.

Strong duality for convex problems

An optimization problem with f_0 and all f_i convex:

$$(\mathcal{P}): \quad \begin{aligned} \text{min} \quad & f_0(x) \\ \text{subject to:} \quad & f_i(x) \leq 0 \quad i = 1 \dots m \\ & Ax = b \quad A \in \mathbb{R}^{p \times n} \end{aligned}$$

Slater Condition

If there is at least one **strictly feasible point**, i.e.

$$\left\{ x \;\middle|\; Ax = b,\ f_i(x) < 0,\ \forall i \in \{1, \dots, m\} \right\} \neq \emptyset$$

Then $p^* = d^*$.

- Stronger version: Only nonlinear inequalities must be strictly feasible.
- Strong duality always holds for LPs (unless primal/dual both infeasible!)

Many other **constraint qualification** conditions can be used.

A geometric interpretation

Assume one inequality constraint only:

$$\mathcal{G} := \{(u,t) \mid t = f_0(x),\ u = f_1(x),\ x \in \mathcal{X}\}$$

Primal problem:

$$p^* = \min\{t \mid (u,t) \in \mathcal{G},\ u \leq 0\}$$

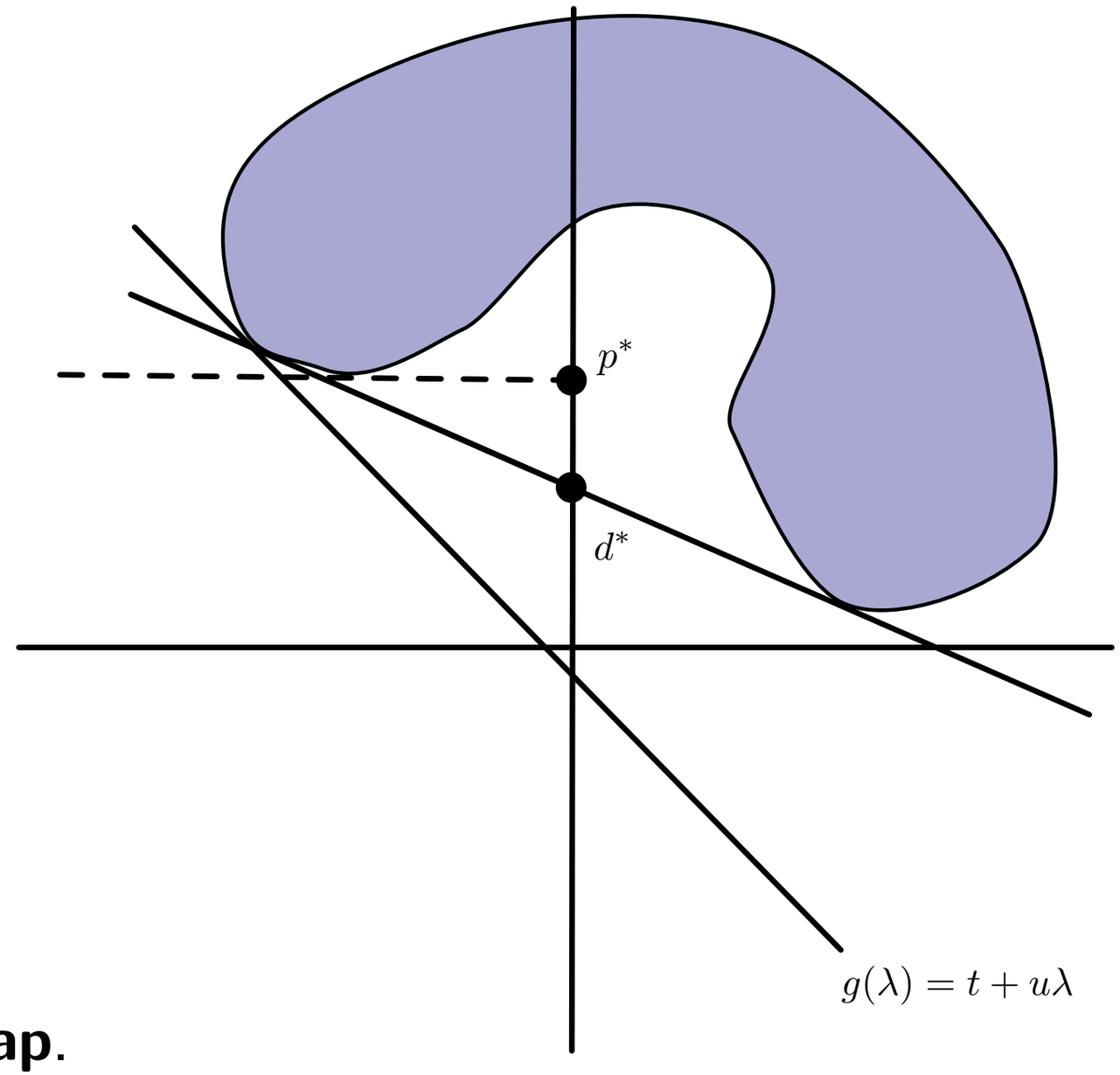
Dual function:

$$g(\lambda) = \min_{(u,t)\in\mathcal{G}} (t + \lambda u)$$

Dual problem:

$$d^* = \max_{\lambda \geq 0} g(\lambda)$$

The quantity $p^* - d^*$ is the **duality gap**.



Primal and dual solution properties

Assume that strong duality holds, with optimal solution x^* and (λ^*, ν^*).

From strong duality, $d^* = p^* \;\; \Rightarrow \;\; g(\lambda^*, \nu^*) = f_0(x^*)$.

Karush-Kuhn-Tucker Conditions

Assume f_i and h_i all differentiable. **Necessary** conditions for optimality:

- **Primal Feasibility**:

$$f_i(x^*) \leq 0 \quad i = 1, \ldots, m$$
$$h_i(x^*) = 0 \quad i = 1 \ldots, p$$

- **Dual Feasibility**:

$$\lambda^* \geq 0$$

- **Complementary Slackness**:

$$\lambda_i f_i(x^*) = 0 \quad i = 1, \ldots, m$$

- **Stationarity**:

$$\nabla_x L(x^*, \lambda^*, \nu^*) = \nabla f_0(x^*) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x^*) + \sum_{i=1}^{p} \nu_i \nabla h_i(x^*) = 0$$

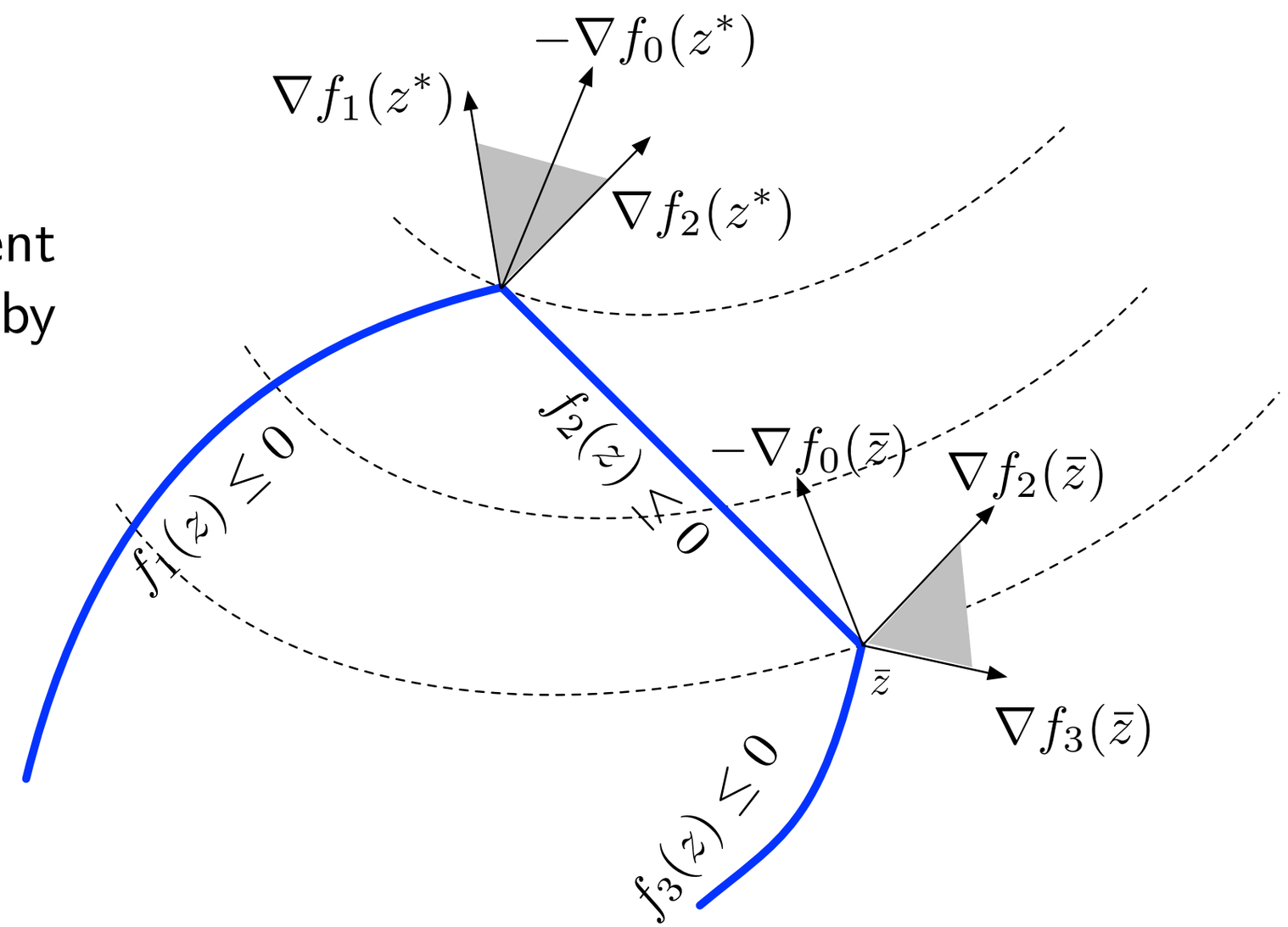
Geometric interpretation of KKT conditions

Assume inequality constraints only.

Rewrite stationarity condition as:

$$-\nabla f_0(x) = \sum_{i=1}^{m} \lambda_i \nabla f_i(x)$$

- Direction of steepest descent is in convex cone spanned by constraint gradients ∇g_i



KKT conditions for convex problems

For a convex optimization problem:

- If (x^*, λ^*, ν^*) satisfy the KKT conditions, then $p^* = d^*$.
  - $p^* = f_0(x^*) = L(x^*, \lambda^*, \nu^*)$ (due to complementary slackness)
  - $d^* = g(\lambda^*, \nu^*) = L(x^*, \lambda^*, \nu^*)$ (due to convexity of the functions and stationarity)
- If the Slater condition holds, then
  - x^* is optimal **if and only if** there exist (λ^*, ν^*) satisfying the KKT conditions.

KKT conditions for a QP

Consider a (convex) quadratic program with $Q \succeq 0$:

$$(\mathcal{P}): \quad \begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \tfrac{1}{2} x^\top Q x + c^\top x \\ \text{subject to:} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

The **Lagrangian** is $L(x, \lambda, \nu) = \frac{1}{2} x^\top Q x + c^\top x + \nu^\top (Ax - b) - \lambda^\top x$.

The KKT conditions are:

$$\begin{aligned} \nabla_x L(x, \lambda, \nu) = Qx + A^\top \nu - \lambda + c = 0 \qquad & \text{[stationarity]} \\ Ax = b \qquad & \text{[primal feasibility]} \\ x \geq 0 \qquad & \text{[primal feasibility]} \\ \lambda \geq 0 \qquad & \text{[dual feasibility]} \\ x_i \lambda_i = 0 \quad i = 1 \ldots n \qquad & \text{[complementarity]} \end{aligned}$$

The final three conditions are often written together as $0 \leq x \perp \lambda \geq 0$.

Example: implicit vs explicit constraints

Example: Box-constrained LP (method 1)

$$(\mathcal{P}): \quad \begin{aligned} \min_{x\in\mathbb{R}^n} \quad & c^\top x \\ \text{subject to:} \quad & Ax = b \\ & -\mathbf{1} \le x \le \mathbf{1} \end{aligned}$$

Dual function:

$$g(\bar{\lambda}, \underline{\lambda}, \nu) = \min_{x\in\mathbb{R}^n} \left[c^\top x + (Ax-b)^\top \nu + (-\mathbf{1}-x)^\top \underline{\lambda} + (-\mathbf{1}+x)^\top \bar{\lambda}\right]$$

$$= \begin{cases} -b^\top \nu - \mathbf{1}^\top \bar{\lambda} - \mathbf{1}^\top \underline{\lambda} & \text{if } c + A^\top \nu - \underline{\lambda} + \bar{\lambda} = 0 \\ -\infty & \text{otherwise} \end{cases}$$

Dual Problem:

$$(\mathcal{D}): \quad \begin{aligned} \max_{\nu,\bar{\lambda},\underline{\lambda}} \quad & -b^\top \nu - \mathbf{1}^\top \bar{\lambda} - \mathbf{1}^\top \underline{\lambda} \\ \text{subject to:} \quad & c + A^\top \nu - \underline{\lambda} + \bar{\lambda} = 0 \\ & \bar{\lambda} \ge 0,\ \underline{\lambda} \ge 0 \end{aligned}$$

Example: implicit vs explicit constraints

Example: Box-constrained LP (method 2)

$$(\mathcal{P}): \quad \begin{array}{ll} \min\limits_{\|x\|_\infty \le 1} & c^\top x \\ \text{subject to:} & Ax = b \end{array}$$

Dual function:

$$g(\nu) = \min_{\|x\|_\infty \le 1} \left[c^\top x + (Ax - b)^\top \nu \right]$$

$$= -b^\top \nu - \|A^\top \nu + c\|_1$$

Dual Problem:

$$(\mathcal{D}): \quad \begin{array}{ll} \max\limits_{\nu} & -b^\top \nu - \|A^\top \nu + c\|_1 \\ \text{subject to:} & \text{[no constraints]} \end{array}$$

NB: same as the previous dual!

Sensitivity analysis

A general optimization problem and its dual:

$$\begin{array}{rll} \min\limits_{x} & f_0(x) & \\ \text{subject to:} & f_i(x) \leq 0 & i = 1 \dots m \\ & h_i(x) = 0 & i = 1 \dots p, \end{array} \quad \Bigg| \quad \begin{array}{rl} \max\limits_{\nu,\lambda} & g(\nu, \lambda) \\ \text{subject to:} & \lambda \geq 0 \end{array}$$

A perturbed optimization problem and its dual:

$$\begin{array}{rll} \min\limits_{x} & f_0(x) & \\ \text{subject to:} & f_i(x) \leq u_i & i = 1 \dots m \\ & h_i(x) = v_i & i = 1 \dots p, \end{array} \quad \Bigg| \quad \begin{array}{rl} \max\limits_{\nu,\lambda} & g(\nu, \lambda) - u^\top \lambda - v^\top \nu \\ \text{subject to:} & \lambda \geq 0 \end{array}$$

- x is the primal decision variable. (λ, ν) are the dual decision variables.
- u and v are parameters representing perturbations to the constraints.
- $p^*(u, v)$ is the optimal value as a function of (u, v).

Sensitivity and Lagrange ultipliers

Assume strong duality holds for the unperturbed problem with (ν^*, λ^*) dual optimal.

Weak duality for the perturbed problem implies

$$\begin{aligned} p^*(u,v) &\geq g^*(\nu^*,\lambda^*) - u^\top\lambda^* - v^\top\nu^* \\ &= p^*(0,0) - u^\top\lambda^* - v^\top\nu^* \end{aligned}$$

Global Sensitivity Analysis

- λ_i^* large and $u_i < 0$ $\Rightarrow$ $p^*(u,v)$ increases greatly.
- λ_i^* small and $u_i > 0$ $\Rightarrow$ $p^*(u,v)$ does not decrease much.
- $\left.\begin{array}{l}\nu^* \text{ large and positive and } v_i < 0 \\ \nu^* \text{ large and negative and } v_i > 0\end{array}\right\}$ $\Rightarrow$ $p^*(u,v)$ increases greatly.
- $\left.\begin{array}{l}\nu^* \text{ small and positive and } v_i > 0 \\ \nu^* \text{ small and negative and } v_i < 0\end{array}\right\}$ $\Rightarrow$ $p^*(u,v)$ does not decrease much.

Note results are **not** symmetrical because we only have a lower bound on $p^*(u,v)$.

Sensitivity and Lagrange multipliers

Assume strong duality holds for the unperturbed problem with (ν^*, λ^*) dual optimal.

Weak duality for the perturbed problem implies

$$\begin{aligned} p^*(u,v) &\geq g^*(\nu^*,\lambda^*) - u^\top\lambda^* - v^\top\nu^* \\ &= p^*(0,0) - u^\top\lambda^* - v^\top\nu^* \end{aligned}$$

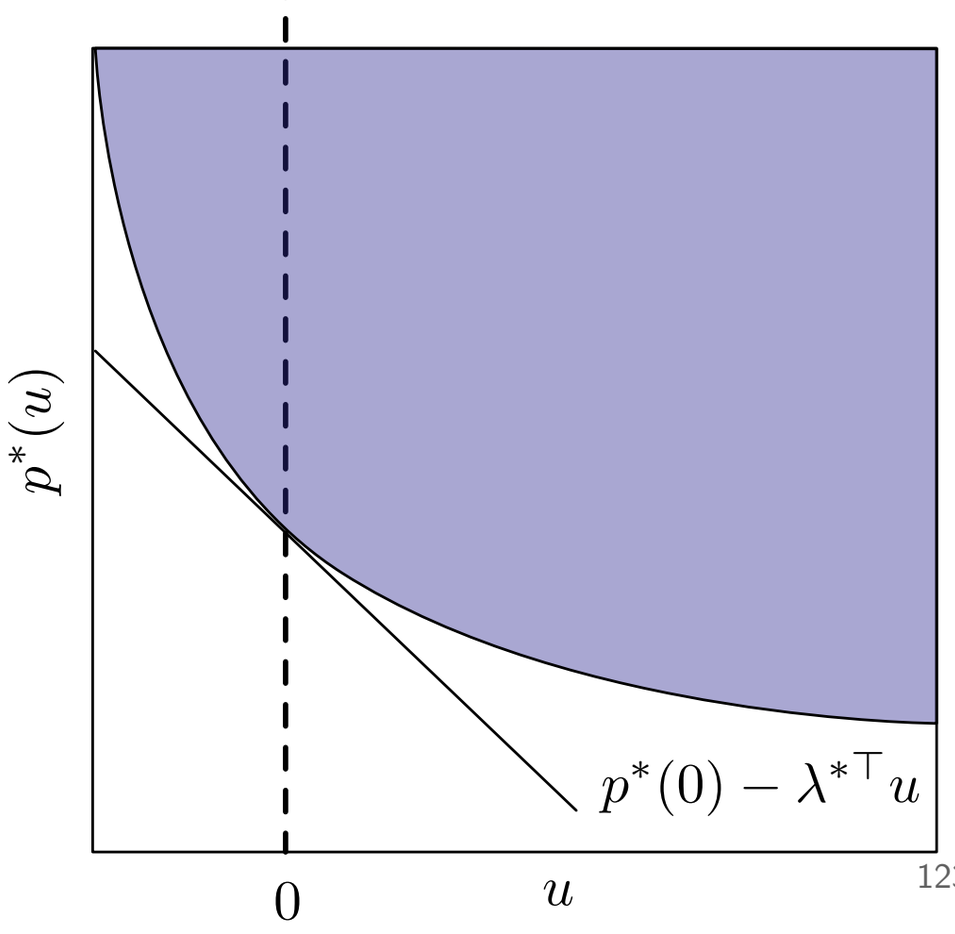
$$\begin{aligned} p^*(u,v) &\geq g^*(\nu^*,\lambda^*) - u^\top\lambda^* - v^\top\nu^* \\ &= p^*(0,0) - u^\top\lambda^* - v^\top\nu^* \end{aligned}$$

Local Sensitivity Analysis

If in addition $p^*(u,v)$ is differentiable at $(0,0)$, then

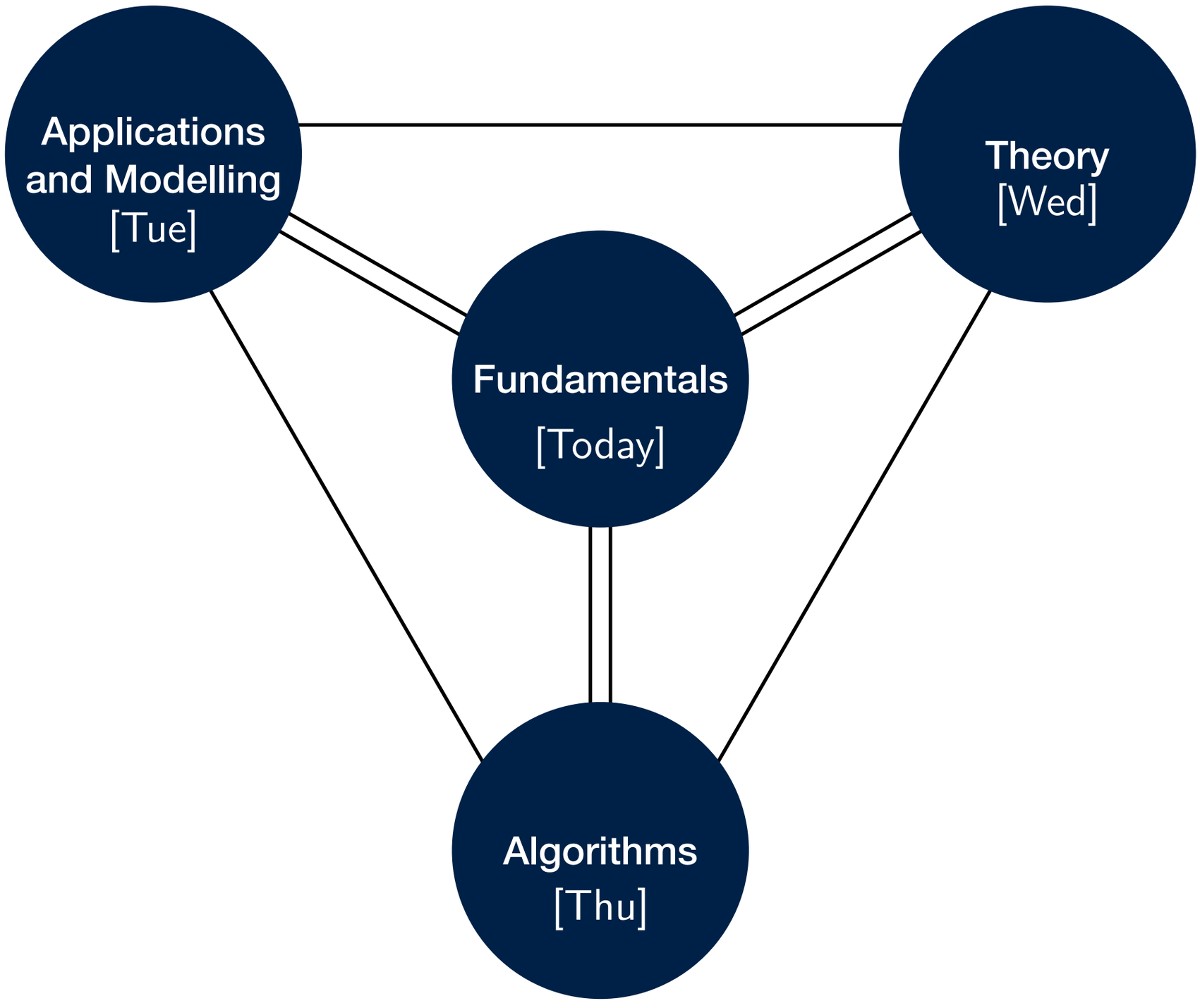
$$\lambda_i^* = -\frac{\partial p^*(0,0)}{\partial u_i}, \quad \nu_i^* = -\frac{\partial p^*(0,0)}{\partial v_i}$$

- λ_i^* is the sensitivity of p^* relative to i^{th} inequality.
- ν_i^* is the sensitivity of p^* relative to i^{th} equality.



Day 4 : Algorithms

This course:


Applications and Modelling [Tue]
Theory [Wed]
Fundamentals [Today]
Algorithms [Thu]

Second order methods

Newton's method

Newton's root finding method

Goal : find a root (i.e. a zero) of the equation

$$h(x) = 0$$

Method

1. Make a guess x^k and a linear approximation

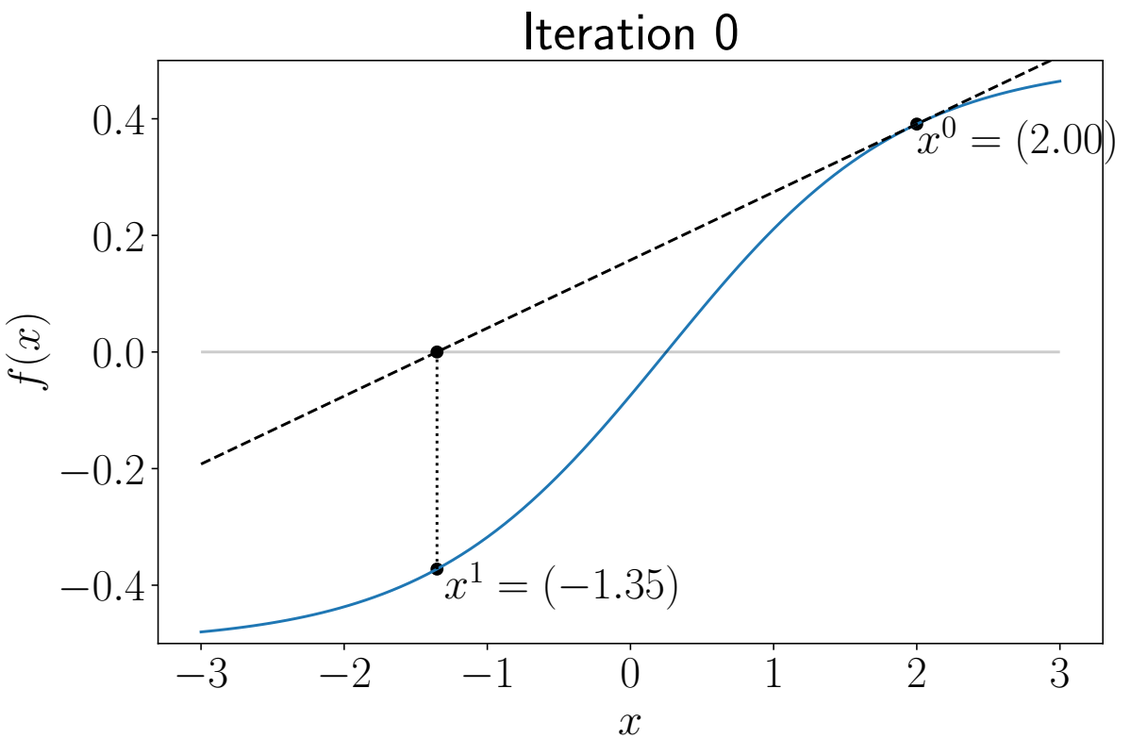
$$h(x) \approx h(x^k) + \frac{\partial h}{\partial x^k}(x^k)(x - x^k)$$

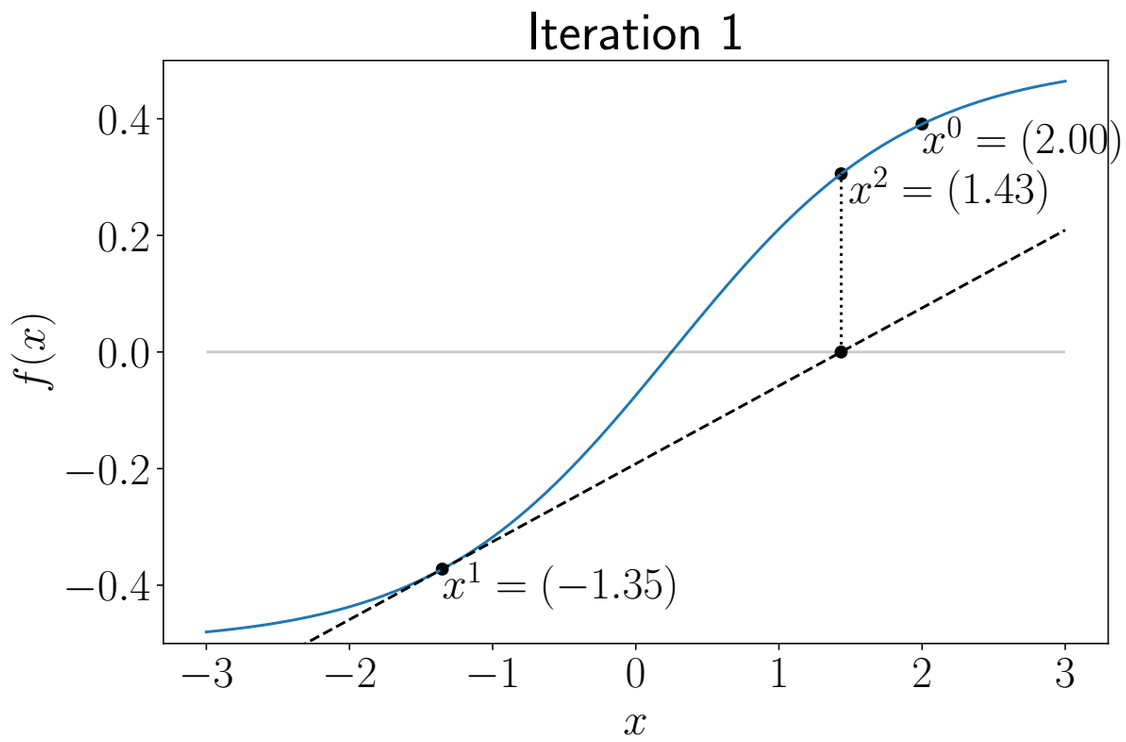
2. Iteratively set $h(x^k)$ to 0

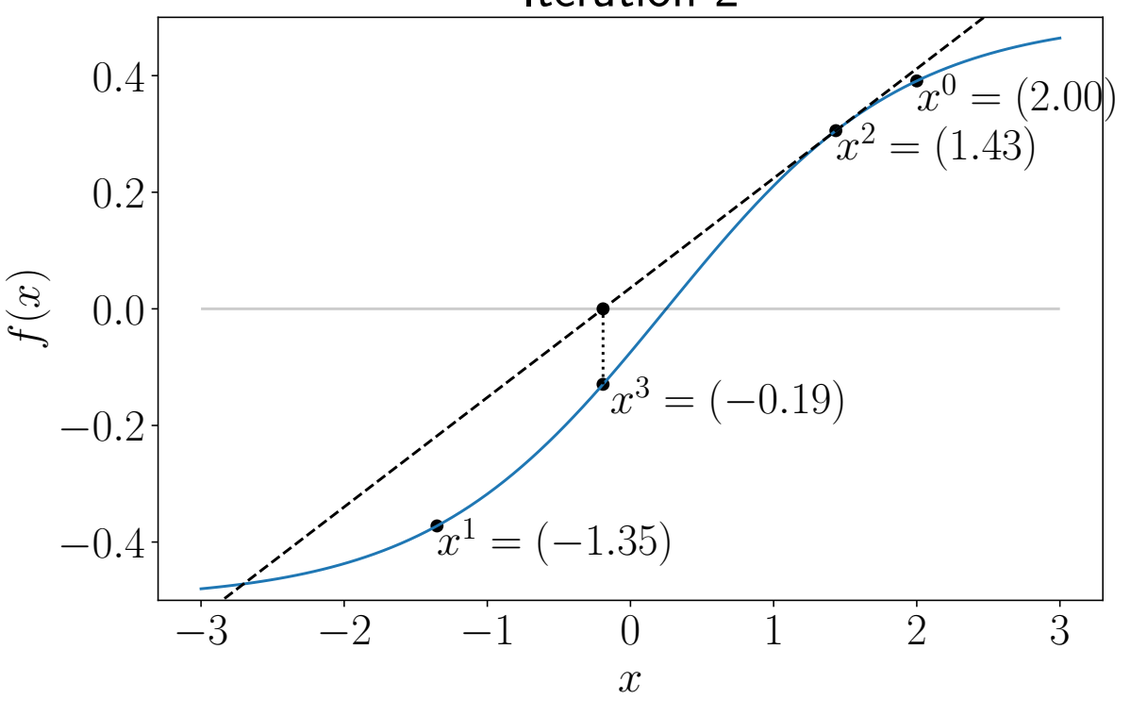
$$h(x^k) + \frac{\partial h}{\partial x^k}(x^k)(x^{k+1} - x^k) = 0$$

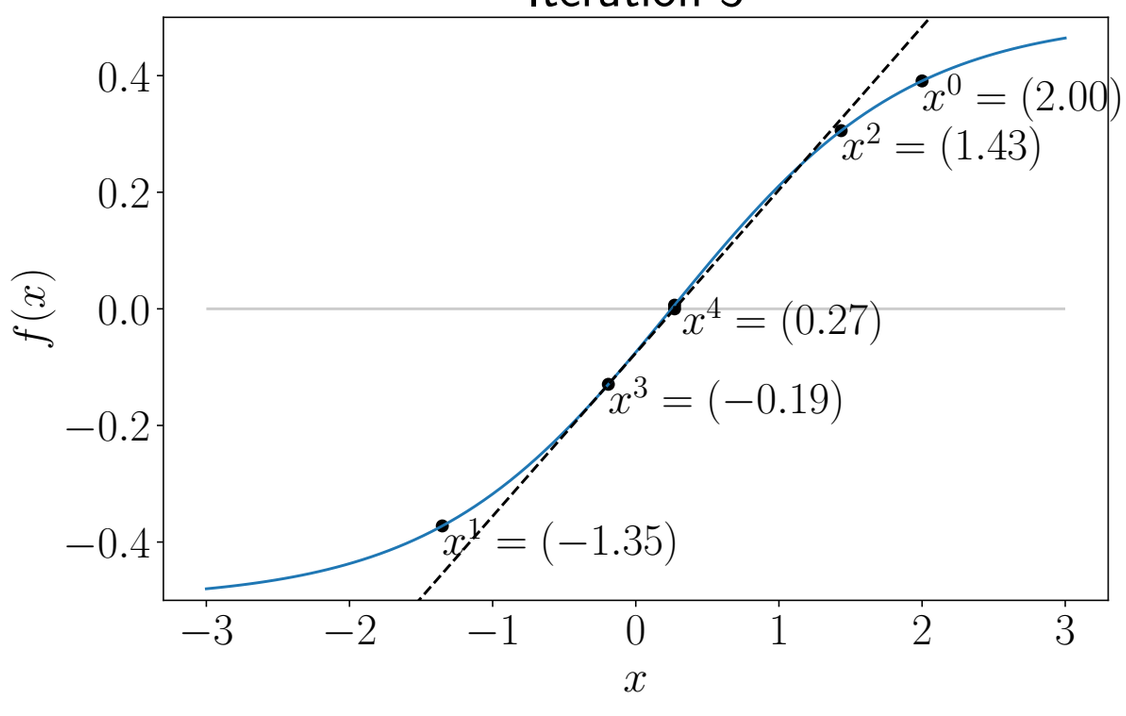
Newton's method example

$$f(x) = \frac{1}{1 + e^{-1.2x+0.3}} - 0.5 = 0 \qquad \Longrightarrow \qquad x^* = 0.3$$









Newton's root finding method (multivariable)

Goal : find a root (i.e. a zero) of the set of equations

$$h(x) = 0$$

where $h : \mathbb{R}^m \to \mathbb{R}^n$ is a **vector valued** function.

Method

1. Make a guess x^k and a linear approximation

$$h(x) \approx h(x^k) + Dh(x^k)(x - x^k)$$

2. Iteratively set $h(x^k)$ to 0

$$h(x^k) + Dh(x^k)(x^{k+1} - x^k) = 0$$

Derivative

$$Dh = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}$$

Newton's method iterations

Update equation is the same as:

$$h(x^k) + Dh(x^k)\underbrace{(x^{k+1} - x^k)}_{\Delta x} = 0$$

Algorithm

1. Solve $Dh(x^k)\Delta x = -h(x^k)$
2. $x^{k+1} \leftarrow x^k + \Delta x$

- Iterations can be **expensive** (factor and solve with $Dh(x^k)$)
- **Fast convergence** close to the solution $x^\star$

The interior point method

Convex optimization as a root finding problem

We will look at the simplest case — the linear program:

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b \end{aligned}$$

Primal

$$\begin{aligned} \text{minimize} \quad & c^T x \\ \text{subject to} \quad & Ax + s = b \\ & s \geq 0 \end{aligned}$$

Dual

$$\begin{aligned} \text{maximize} \quad & -b^T y \\ \text{subject to} \quad & A^T y + c = 0 \\ & y \geq 0 \end{aligned}$$

KKT conditions

$$\left.\begin{aligned} Ax + s - b &= 0 \\ A^T y + c &= 0 \\ s_i y_i &= 0, \quad i = 1, \ldots, m \\ s, y &\geq 0 \end{aligned}\right\} \longrightarrow \text{A root finding problem!}$$

Convex optimization as a root finding problem

$$
\begin{aligned}
Ax + s - b &= 0 \\
A^T y + c &= 0 \\
s_i y_i &= 0, \quad i = 1, \ldots, m \\
s, y &\geq 0
\end{aligned}
$$

Diagonalize complementary slackness

$$
S = \mathbf{diag}(s) = \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_m \end{bmatrix}
\qquad
SY\mathbf{1} = \mathbf{diag}(s)\mathbf{diag}(y)\mathbf{1} = \begin{bmatrix} s_1 y_1 & & & \\ & s_2 y_2 & & \\ & & \ddots & \\ & & & s_m y_m \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 y_1 \\ s_2 y_2 \\ \vdots \\ s_m y_m \end{bmatrix}
$$

$$
Y = \mathbf{diag}(y) = \begin{bmatrix} y_1 & & & \\ & y_2 & & \\ & & \ddots & \\ & & & y_m \end{bmatrix}
$$

$$
\downarrow
$$

$$
s_i y_i = 0, \quad i = 1, \ldots, m \qquad \Longleftrightarrow \qquad SY\mathbf{1} = 0
$$

Main idea

Optimality conditions

$$h(y,x,s) = \begin{bmatrix} Ax + s - b \\ A^T y + c \\ SY\mathbf{1} \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ SY\mathbf{1} \end{bmatrix} = 0 \qquad \begin{aligned} S &= \mathbf{diag}(s) \\ Y &= \mathbf{diag}(y) \end{aligned}$$

$$s, y \geq 0$$

- Apply variants of Newton's method to solve $h(x,s,y) = 0$
- Enforce $s, y > 0$ (strictly) at every iteration
- **Motivation** avoid getting stuck in "corners"

Newton's method for optimality conditions

Optimality conditions

$$h(y,x,s) = \begin{bmatrix} Ax+s-b \\ A^Ty+c \\ SY\mathbf{1} \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ SY\mathbf{1} \end{bmatrix} = 0$$

$$s, y \geq 0$$

Derivative

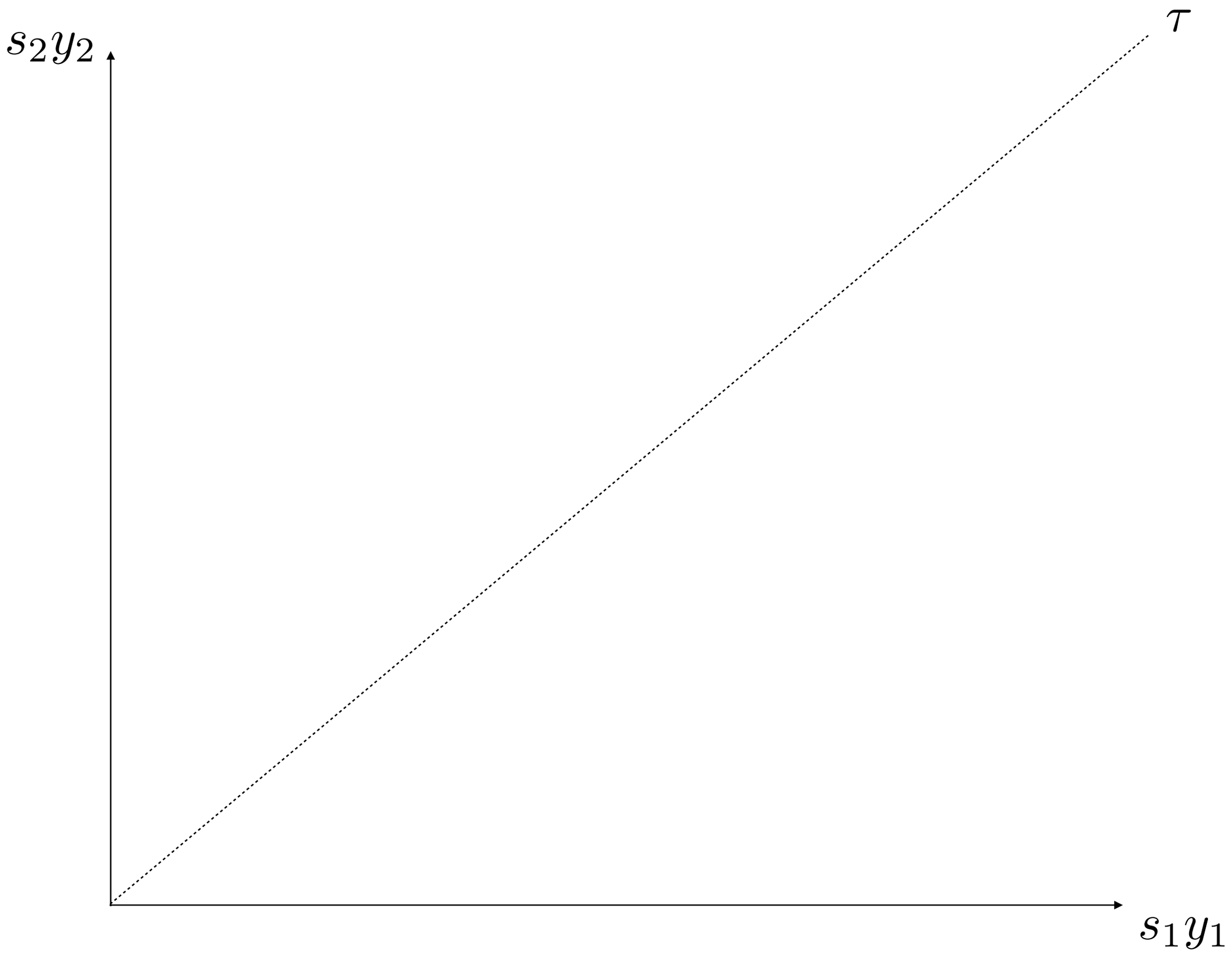
$$Dh(y,x,s) = \begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix}$$

Iterations

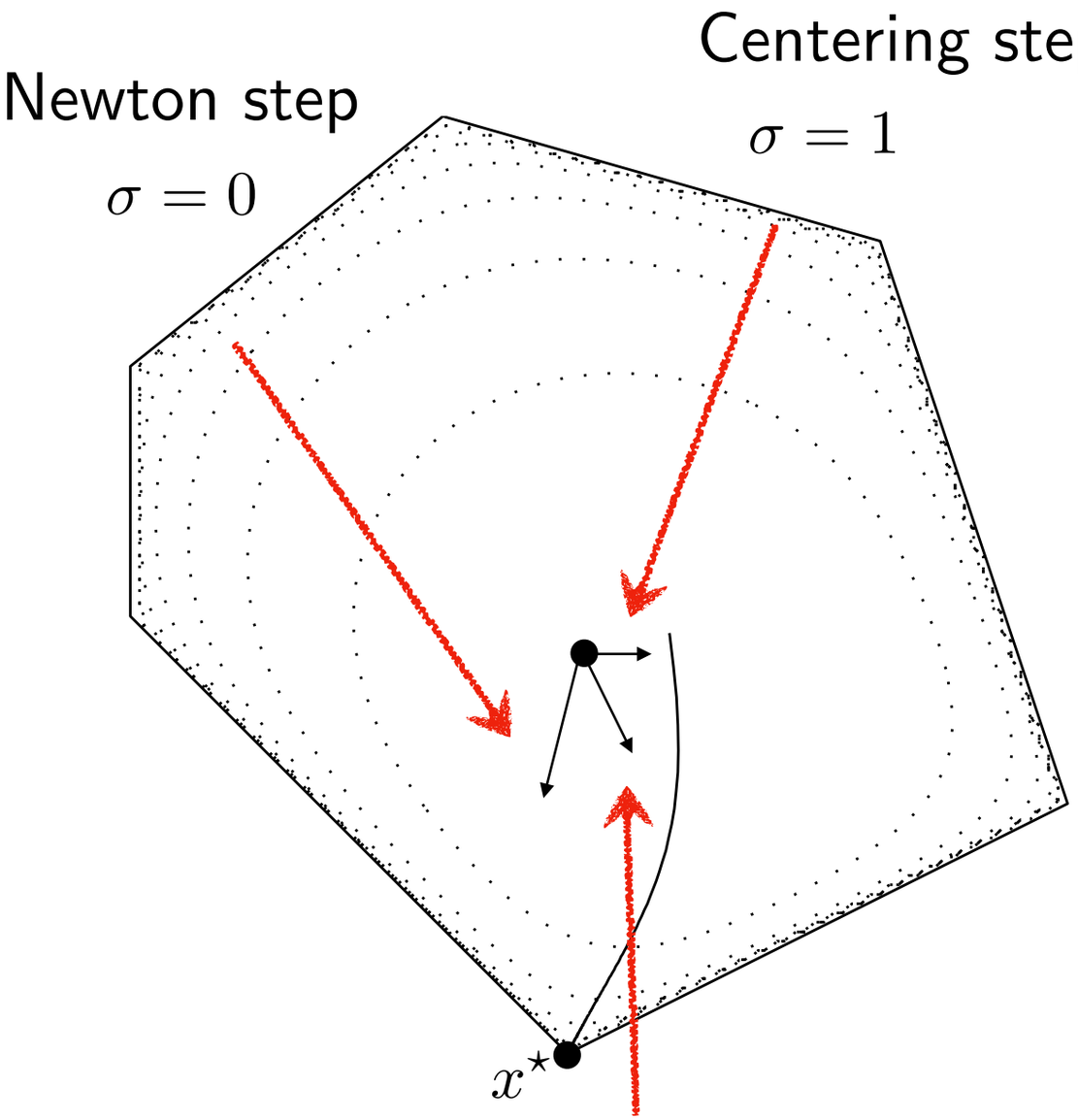
- Solve $Dh(y^k,x^k,s^k)\Delta(y^k,x^k,s^k) = -h(y^k,x^k,s^k)$
- $$\begin{bmatrix} y^{k+1} \\ x^{k+1} \\ s^{k+1} \end{bmatrix} \leftarrow \begin{bmatrix} y^k \\ x^k \\ s^k \end{bmatrix} + \Delta(y^k,x^k,s^k)$$

Caution! This might make (s,y) negative!

The central path



Path following methods



Combined step

Centering step

Brings iterate towards the **central path** and usually biased towards $s, y > 0$.
No progress on the duality measure μ.

Newton step

It brings towards the **zero duality measure** μ. Quickly violates $s, y > 0$.

Combined step

Best of both worlds with longer steps

Primal-dual path-following algorithm

Initialization

1. Given (x_0, s_0, y_0) such that $s_0, y_0 > 0$

Iterations

1. Choose $\sigma \in [0, 1]$
2. Solve $\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} + \sigma\mu\mathbf{1} \end{bmatrix}$ where $\mu = s^T y/m$
3. Find maximum α such that $y + \alpha\Delta y > 0$ and $s + \alpha\Delta s > 0$
4. Update $(y, x, s) \leftarrow (y, x, s) + \alpha(\Delta y, \Delta x, \Delta s)$

Working towards optimality conditions

Optimality conditions satisfied **only at convergence**

Stopping criteria

Primal residual

$$r_p = Ax + s - b \to 0 \qquad \|r_p\| \leq \epsilon_{\text{pri}}$$

Dual residual

$$r_d = A^T y + c \to 0 \qquad \|r_d\| \leq \epsilon_{\text{dua}}$$

Complementary slackness

$$s^T y \to 0 \qquad s^T y \leq \epsilon_{\text{gap}}$$

Smoothed optimality conditions

Optimality conditions

$$
\begin{aligned}
Ax + s - b &= 0 \\
A^T y + c &= 0 \\
s_i y_i &= \tau \quad \longleftarrow \text{ Same } \tau \text{ for every pair} \\
s, y &\geq 0
\end{aligned}
$$

Same optimality conditions for a "smoothed" version of our problem

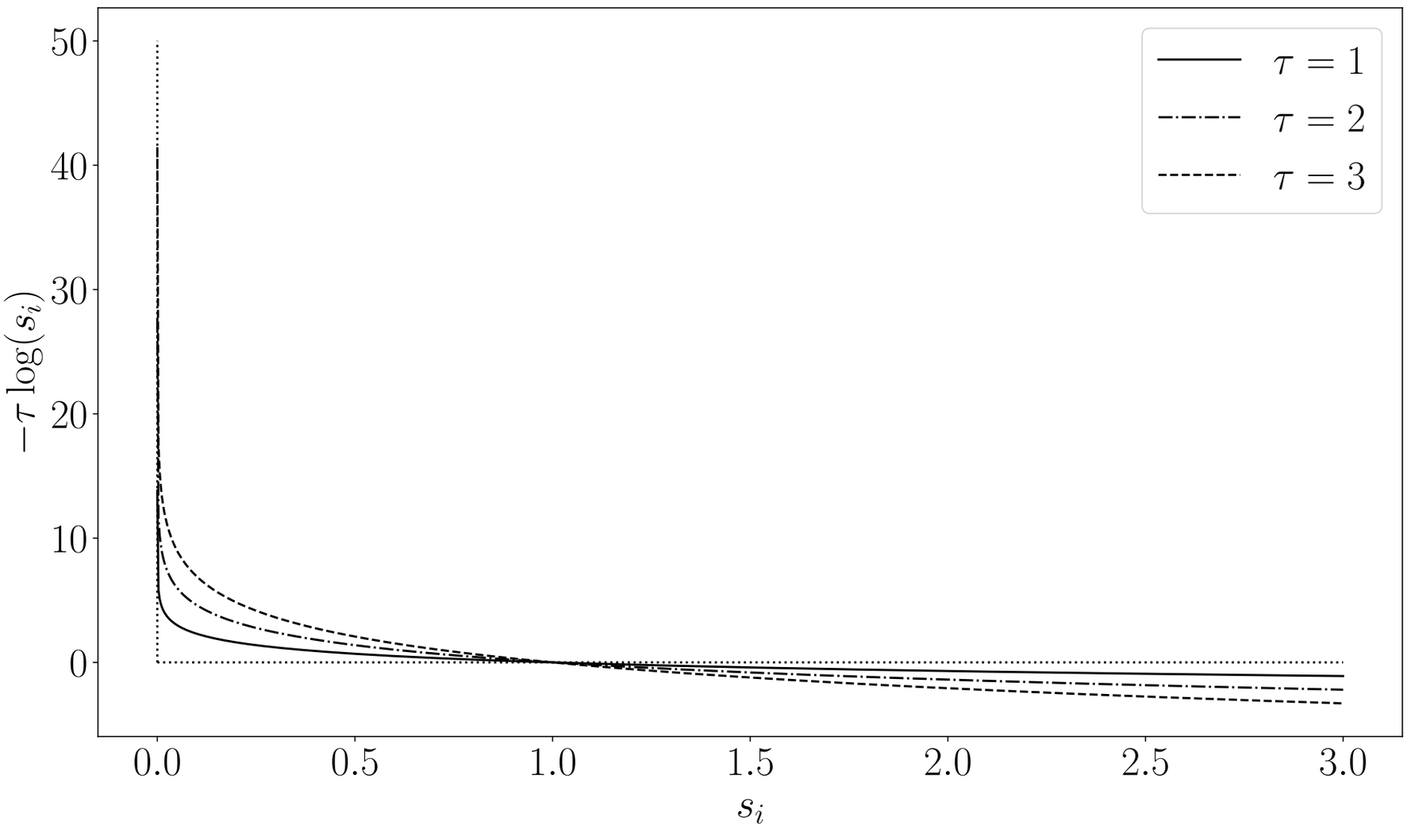
Do solutions actually exist*?

What do they represent?

(*spoiler: yes)

Log barrier functions

$$\phi(s) = -\tau \sum_{i=1}^{m} \log(s_i) \quad \text{on domain} \quad s_i > 0$$



As $\tau \to 0$ it approximates

$$\mathcal{I}_{s_i \geq 0} = \begin{cases} 0 & \text{if } s_i \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

Smoothed Problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax + s = b \\ & s \geq 0 \end{array} \longrightarrow \begin{array}{ll} \text{minimize} & c^T x + \phi(s) = c^T x - \tau \sum_{i=1}^{m} \log(s_i) \\ \text{subject to} & Ax + s = b \end{array}$$

Lagrangian function

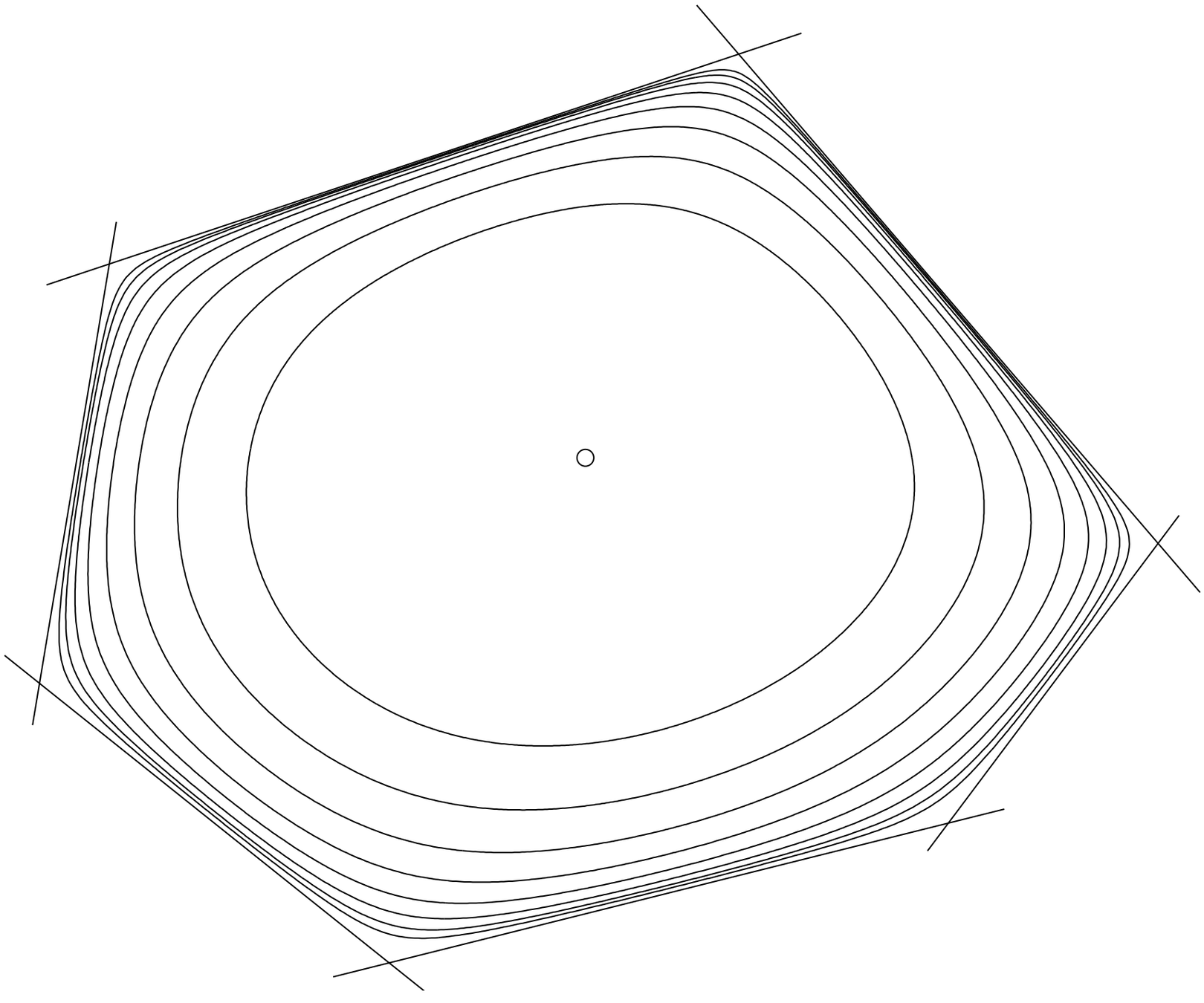
$$L(x, s, y) = c^T x - \tau \sum_{i=1}^{m} \log(s_i) + y^T (Ax + s - b)$$

$$\frac{\partial L}{\partial x} = A^T y + c = 0$$

$$\frac{\partial L}{\partial s_i} = -\tau \frac{1}{s_i} + y_i = 0 \quad \Longrightarrow s_i y_i = \tau$$

Smoothed feasible region

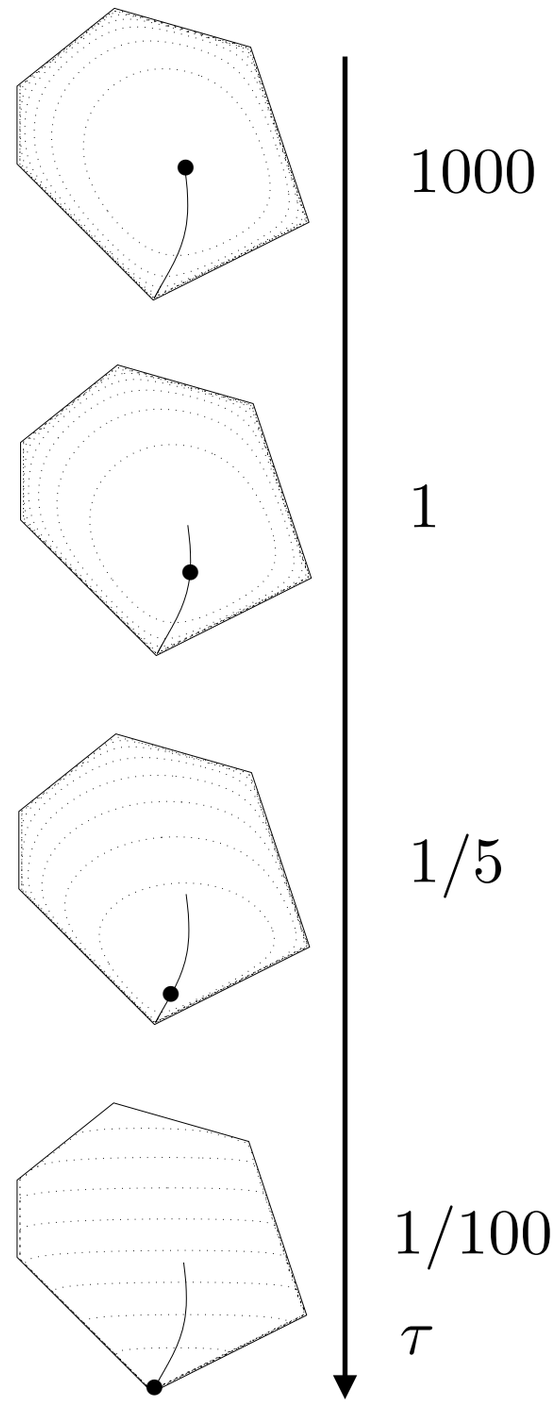
$$\phi(s) = -\tau \sum_{i=1}^{m} \log(s_i) \quad \text{on domain} \quad s_i > 0$$



The central path

minimize $c^T x - \tau \sum_{i=1}^m \log(s_i)$
subject to $Ax + s = b$

Analytic Center
$\tau \to \infty$



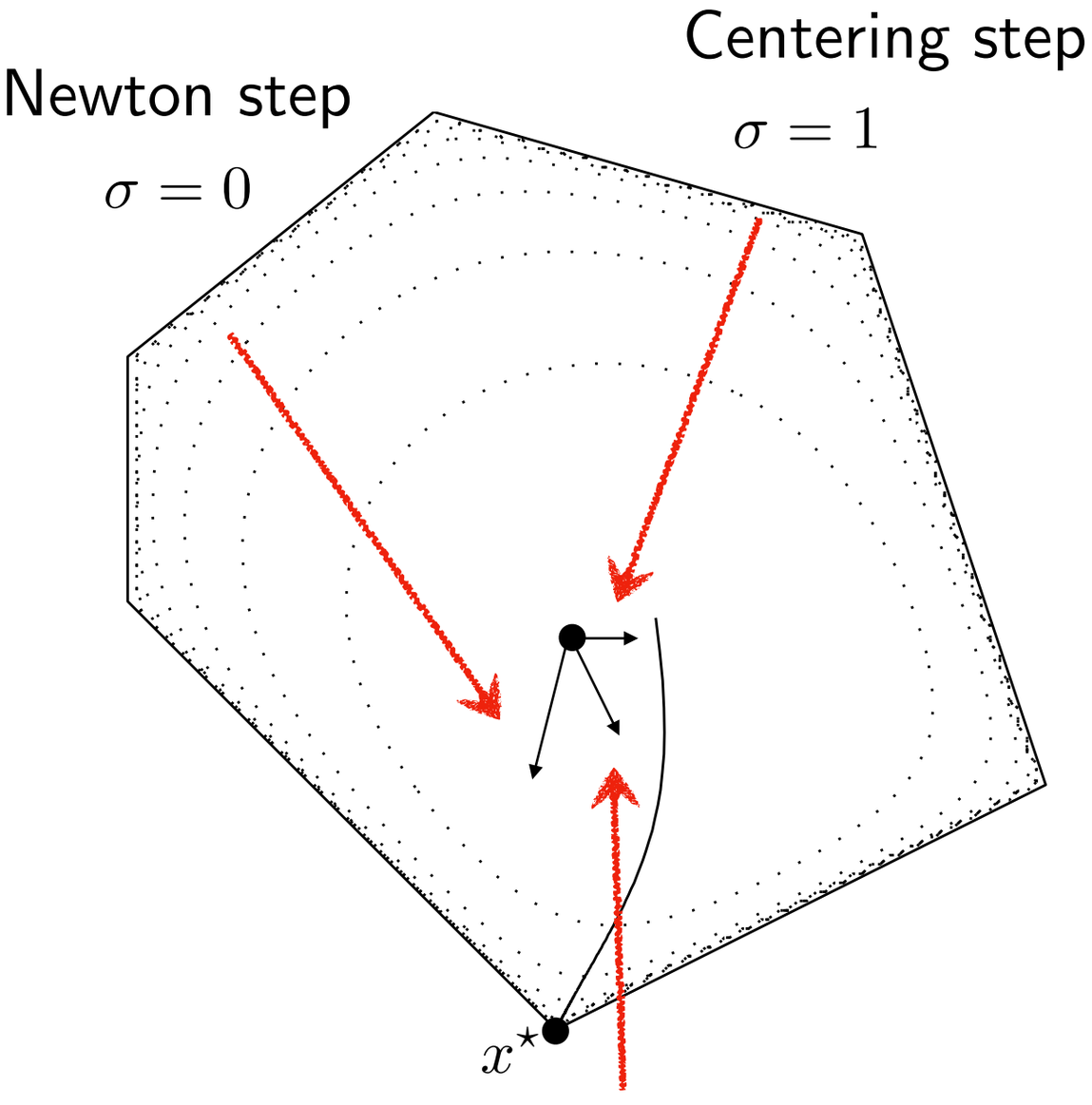
Central path
Set of points $(x^\star(\tau), s^\star(\tau), y^\star(\tau))$ with $\tau > 0$ such that

$$\begin{aligned} Ax + s - b &= 0 \\ A^T y + c &= 0 \\ s_i y_i &= \tau \\ s, y &\geq 0 \end{aligned}$$

Main idea

Follow central path as $\tau \to 0$

Predictor-corrector methods



Combined step

Predict

Compute the Newton direction

Estimate

How good is the Newton step / how much does μ decrease?

Choose a centering parameter

Rough guideline:

Pick $\sigma \approx 0$ if Newton step is good

Pick $\sigma \approx 1$ if Newton step is bad

How good is the Newton step?

Newton step

$$(\Delta x_a, \Delta s_a, \Delta y_a)$$

Maximum step-size

$$\alpha_p = \max\{\alpha \in [0,1] \mid s + \alpha \Delta s_a \geq 0\}$$

$$\alpha_d = \max\{\alpha \in [0,1] \mid y + \alpha \Delta y_a \geq 0\}$$

Two issues

- The new points might will not produce much improvement: $(s + \alpha_p \Delta s_a)_i (y + \alpha_d \Delta y_a)_i$ much larger than 0
- The complementarity error depends on the step lengths α_p and α_d

Choosing the centering parameter to get good improvement

Newton step

$$(\Delta x_a, \Delta s_a, \Delta y_a)$$

Maximum step-size

$$\alpha_p = \max\{\alpha \in [0,1] \mid s + \alpha \Delta s_a \geq 0\}$$

$$\alpha_d = \max\{\alpha \in [0,1] \mid y + \alpha \Delta y_a \geq 0\}$$

Duality measure candidate
(after Newton step)

$$\mu_a = \frac{(s + \alpha_p \Delta s_a)^T (y + \alpha_d \Delta y_a)}{m}$$

Centering parameter heuristic

$$\sigma = \left(\frac{\mu_a}{\mu}\right)^3$$

Correcting for complementarity error

Newton step

$$\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y_a \\ \Delta x_a \\ \Delta s_a \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} \end{bmatrix} \longrightarrow \quad s_i(\Delta y_a)_i + y_i(\Delta s_a)_i + s_i y_i = 0$$

Complementarity error

$$(s_i + (\Delta s_a)_i)(y_i + (\Delta y_a)_i) = (\Delta s_a)_i(\Delta y_a)_i \neq 0$$

Complementarity violation
depends on step length

Corrected / centering step

$$\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} - \Delta S_a \Delta Y_a \mathbf{1} + \sigma\mu\mathbf{1} \end{bmatrix}$$

$$\Delta S_a = \mathbf{diag}(\Delta s_a)$$
$$\Delta Y_a = \mathbf{diag}(\Delta y_a)$$

Mehrotra predictor-corrector algorithm

Initialization

Given (x, s, y) such that $s, y > 0$

1. **Termination conditions**

   $r_p = Ax + s - b, \quad r_d = A^T y + c, \quad \mu = (s^T y)/m$

   If $\|r_p\|, \|r_d\|, \mu$ are small, **break** Optimal solution $(x^\star, s^\star, y^\star)$

2. **Newton step** (affine scaling)

$$\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y_a \\ \Delta x_a \\ \Delta s_a \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} \end{bmatrix}$$

Mehrotra predictor-corrector algorithm

3. **Barrier parameter**

$$\alpha_p = \mathsf{max}\{\alpha \in [0,1] \mid s + \alpha \Delta s_a \geq 0\}$$

$$\alpha_d = \mathsf{max}\{\alpha \in [0,1] \mid y + \alpha \Delta y_a \geq 0\}$$

$$\mu_a = \frac{(s + \alpha_p \Delta s_a)^T (y + \alpha_d \Delta y_a)}{m}$$

$$\sigma = \left(\frac{\mu_a}{\mu} \right)^3$$

4. **Corrected direction**

$$\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} - \Delta S_a \Delta Y_a \mathbf{1} + \sigma \mu \mathbf{1} \end{bmatrix}$$

Mehrotra predictor-corrector algorithm

5. **Update iterates**

$$\alpha_p = \mathsf{max}\{\alpha \geq 0 \mid s + \alpha \Delta s \geq 0\}$$
$$\alpha_d = \mathsf{max}\{\alpha \geq 0 \mid y + \alpha \Delta y \geq 0\}$$

$$(x, s) = (x, s) + \mathsf{min}\{1, \eta\alpha_p\}(\Delta x, \Delta s)$$
$$y = y + \mathsf{min}\{1, \eta\alpha_d\}\Delta y$$

Nearly all modern interior point solvers are based on this method!

First order methods

Newton's method revisited

Goal : find a root (i.e. a zero) of the set of equations

$$h(x) = 0$$

where $h : \mathbb{R}^m \to \mathbb{R}^n$ is a **vector valued** function.

Method

1. Make a guess x^k and a linear approximation

$$h(x) \approx h(x^k) + Dh(x^k)(x - x^k)$$ Differentiation

2. Iteratively set $h(x^k)$ to 0

We differentiated only one time. What is "second order" about it?

Newton's method revisited

Goal : find a minimiser of a function:

$$\min f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ maps a vector to a scalar.

Method

1. Differentiate once to get a **stationarity condition**.

$$g(x) = \nabla f(x) = 0$$ Differentiation

2. Make a guess x^k and a linear approximation

$$g(x) \approx g(x^k) + Dg(x^k)(x - x^k) = \nabla f(x^k) + \nabla^2 g(x^k)(x - x^k)$$ Differentiation again!

A general descent scheme

Goal : find a minimiser of a function:

$$\min f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable.

Method

1. Choose a **descent direction** d^k, i.e. $\nabla f(x^k)^T d^k < 0$
2. Choose a step size $\alpha^k > 0$
3. Set $x^{k+1} = x^k + \alpha^k d^k$

NB: Now we only need the gradient, and the iterations are **very cheap**.

Gradient descent

Goal : find a minimiser of a function:

$$\min f(x)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is differentiable.

Choose $d^k = -\nabla f(x^k)$

Method

1. Choose a **descent direction** d^k, i.e. $\nabla f(x^k)^T d^k < 0$
2. Choose a step size $\alpha^k > 0$
3. Set $x^{k+1} = x^k + \alpha^k d^k \ = x^k - \alpha^k \nabla f(x^k)$

This is the **steepest descent** strategy. It just says "go directly downhill".

Steepest descent as quadratic approximation

1. Approximate function around the current iterate:

$$f(x) \approx f(x^k) + \nabla f(x^k)^T (x - x^k) + \frac{1}{2\alpha_k} \|x - x^k\|_2^2$$

Important : not quite a Taylor series approximation.

We replaced $\nabla^2 f$ with $\frac{1}{\alpha_k} I$.

2. Set the gradient of our approximation to zero w.r.t. x:

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k)$$

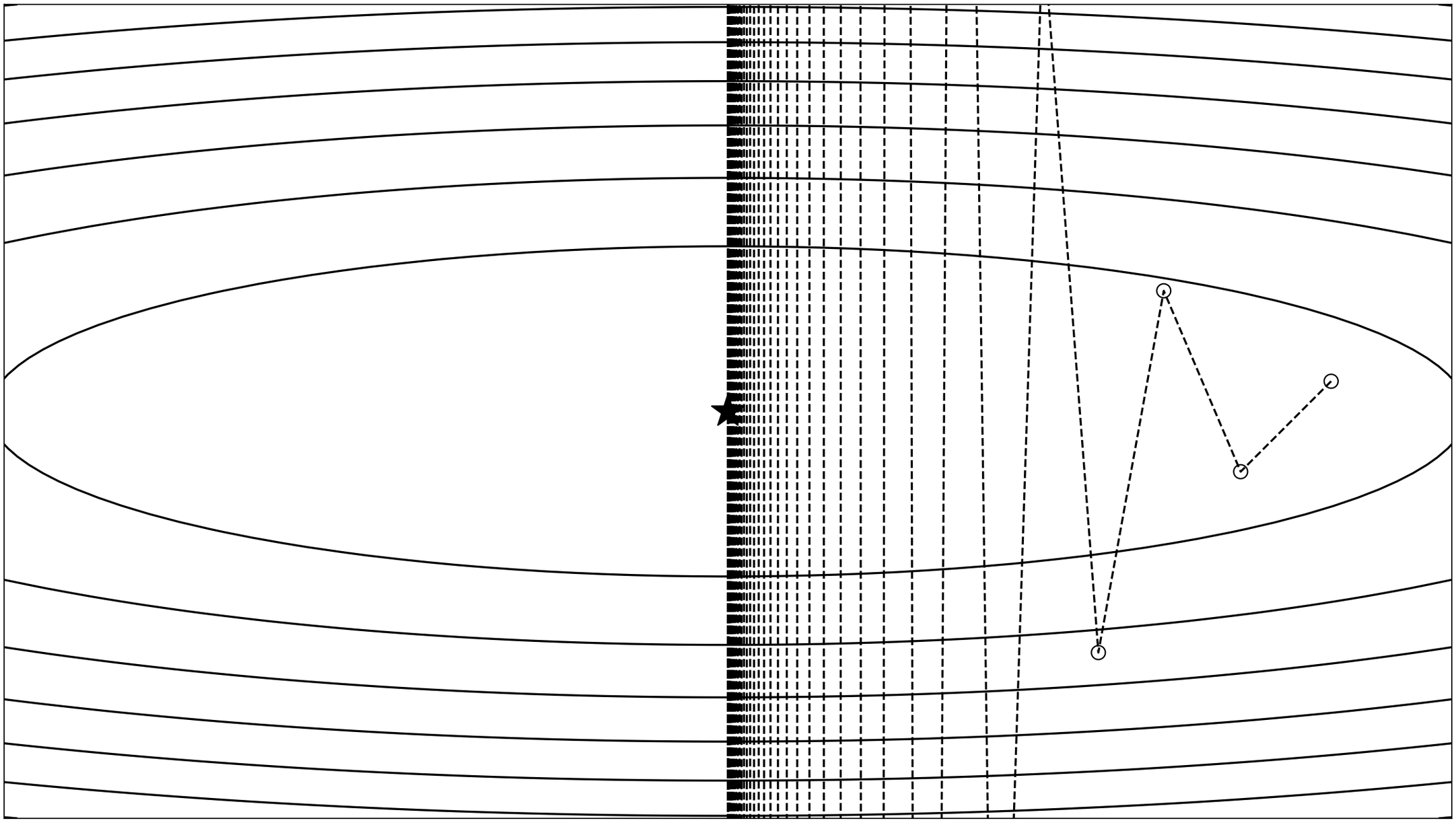
Fixed step sizes

$$f(x) = (x_1^2 + 20x_2^2)/2$$

$$x^0 = (20, 1)$$

$$t = 0.15$$

Diverges



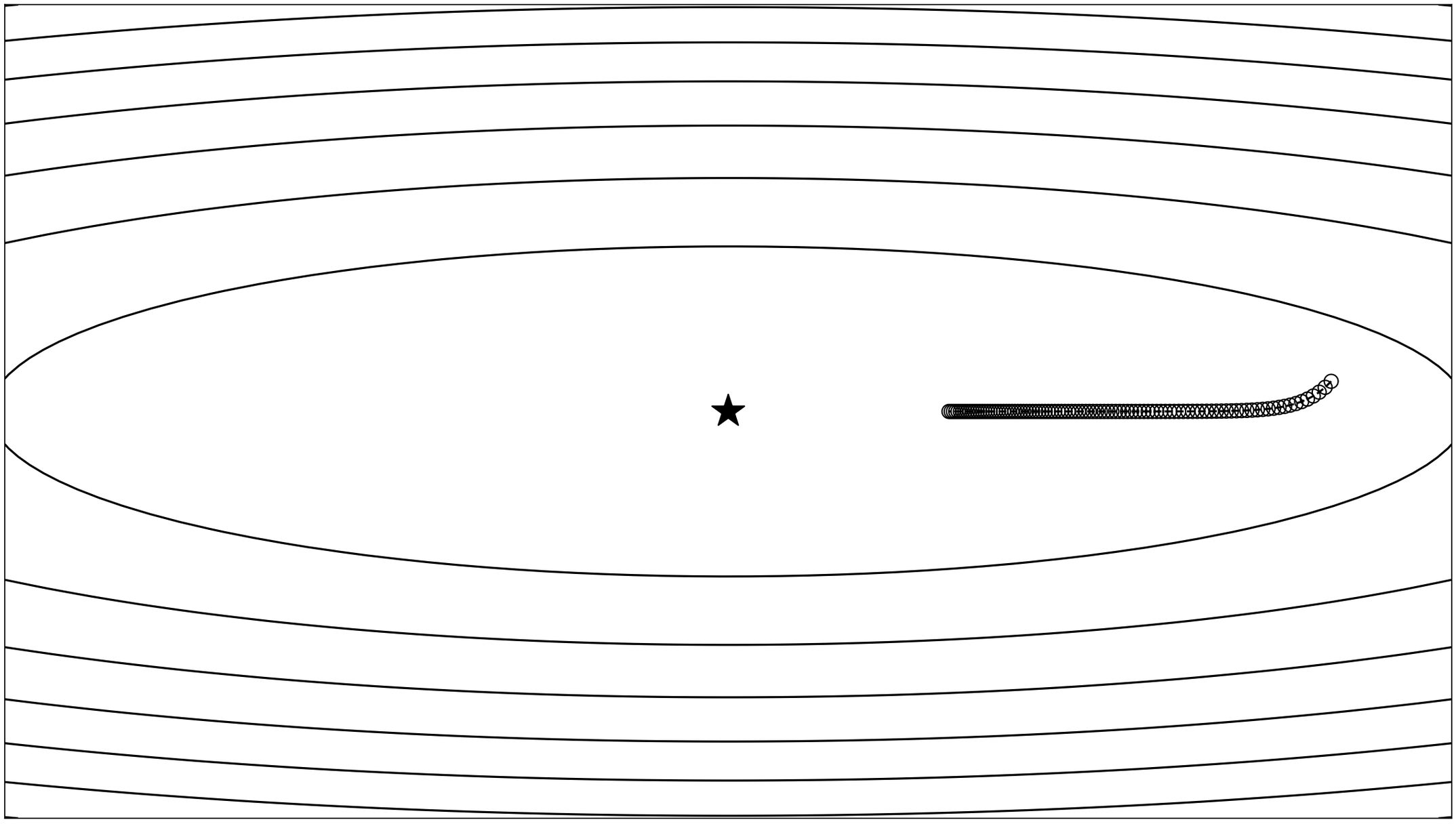
Fixed step sizes

$$f(x) = (x_1^2 + 20x_2^2)/2$$

$$x^0 = (20, 1)$$

$$t = 0.01$$

Very slow



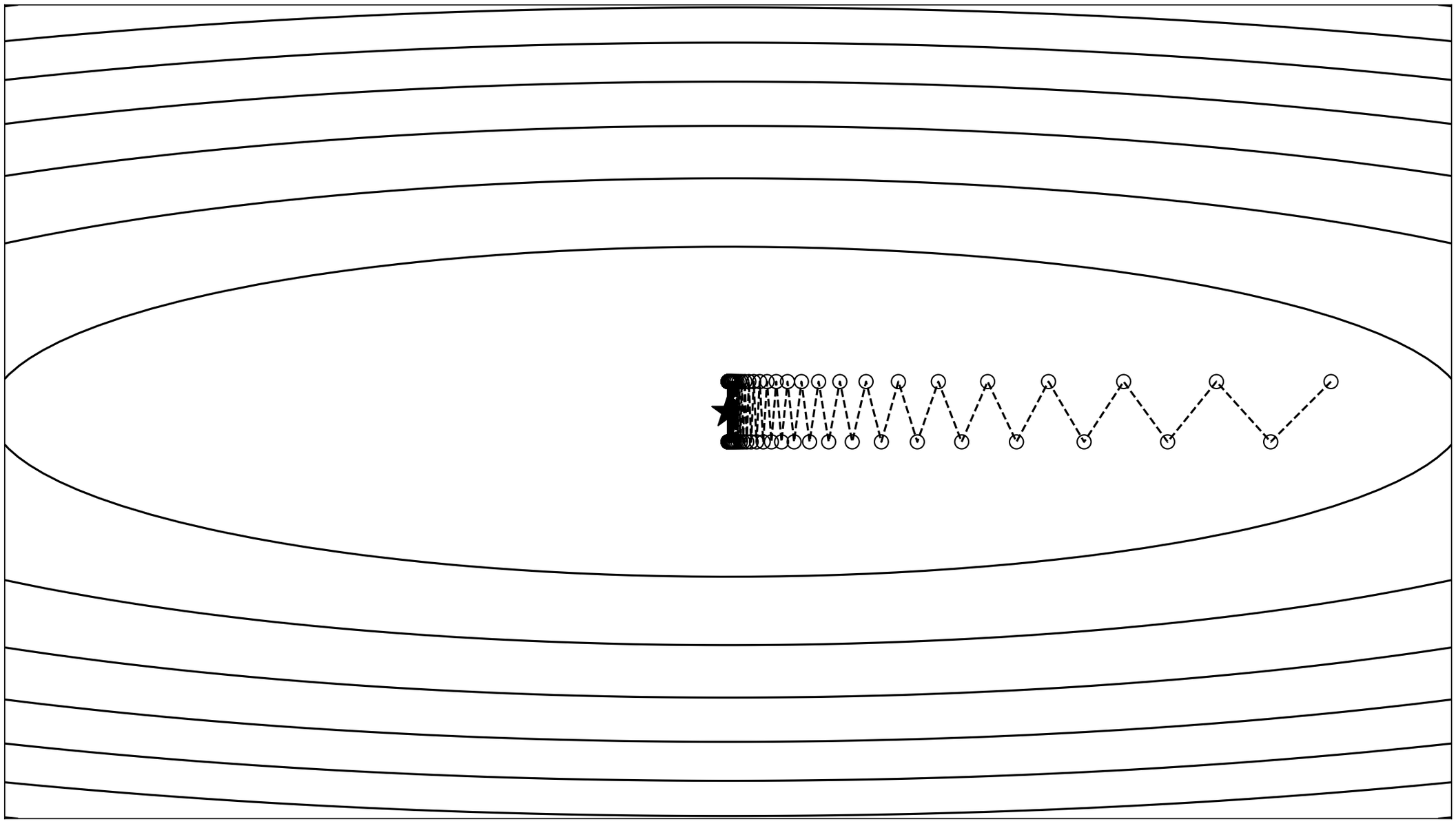
Fixed step sizes

$$f(x) = (x_1^2 + 20x_2^2)/2$$

$$x^0 = (20, 1)$$

$$t = 0.10$$

Oscillates



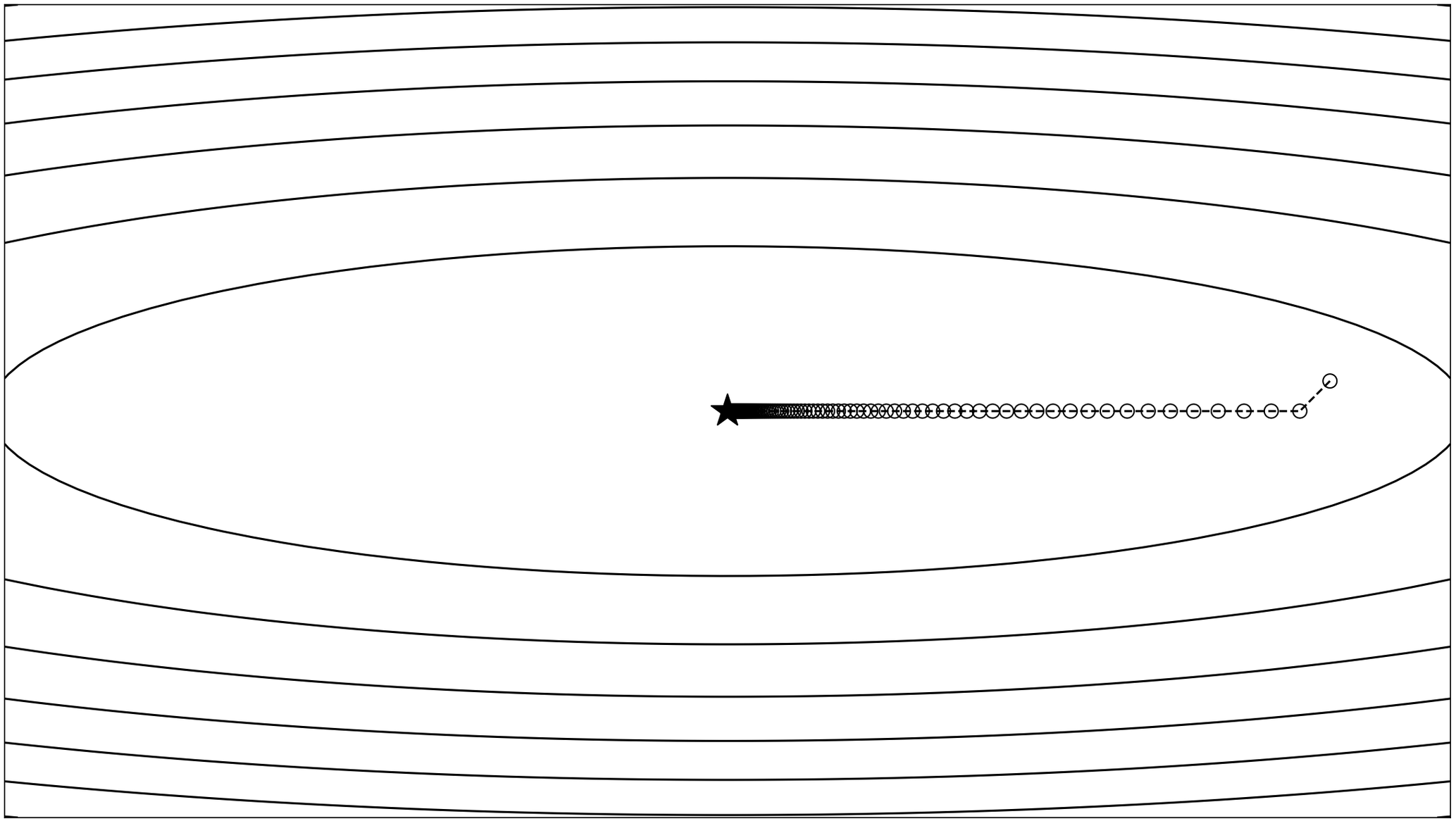
Fixed step sizes

$$f(x) = (x_1^2 + 20x_2^2)/2$$

$$x^0 = (20, 1)$$

$$t = 0.05$$

Just right?



Gradient descent for a quadratic function

$$\text{minimize} \quad f(x) = \frac{1}{2}(x - x^\star)^T P(x - x^\star)$$

Gradient descent method becomes:

$$x^{k+1} = x^k - \alpha^k \nabla f(x^k) = x^k - \alpha^k P(x^k - x^*)$$

Theorem

If $t_k = t = \dfrac{2}{\lambda_{\min}(P) + \lambda_{\max}(P)}$, then

$$\|x^k - x^\star\|_2 \leq \left(\frac{\mathbf{cond}(P) - 1}{\mathbf{cond}(P) + 1}\right)^k \|x^0 - x^\star\|_2$$

This (optimal) rate depends on the c**ondition number**.

Gradient descent for more general (convex) functions

Strong convexity and smoothness properties

$$0 \preceq \mu I \preceq \nabla^2 f(x) \preceq LI, \quad \forall x$$

- The function is L-**smooth**.
- The function is μ-**strongly convex**.

Theorem

If $t_k = t = \dfrac{2}{\mu + L}$, then

If $t_k = t = \dfrac{2}{\lambda_{\min}(P) + \lambda_{\max}(P)}$, then

$$\|x^k - x^\star\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|x^0 - x^\star\|_2, \qquad \kappa = L/\mu$$

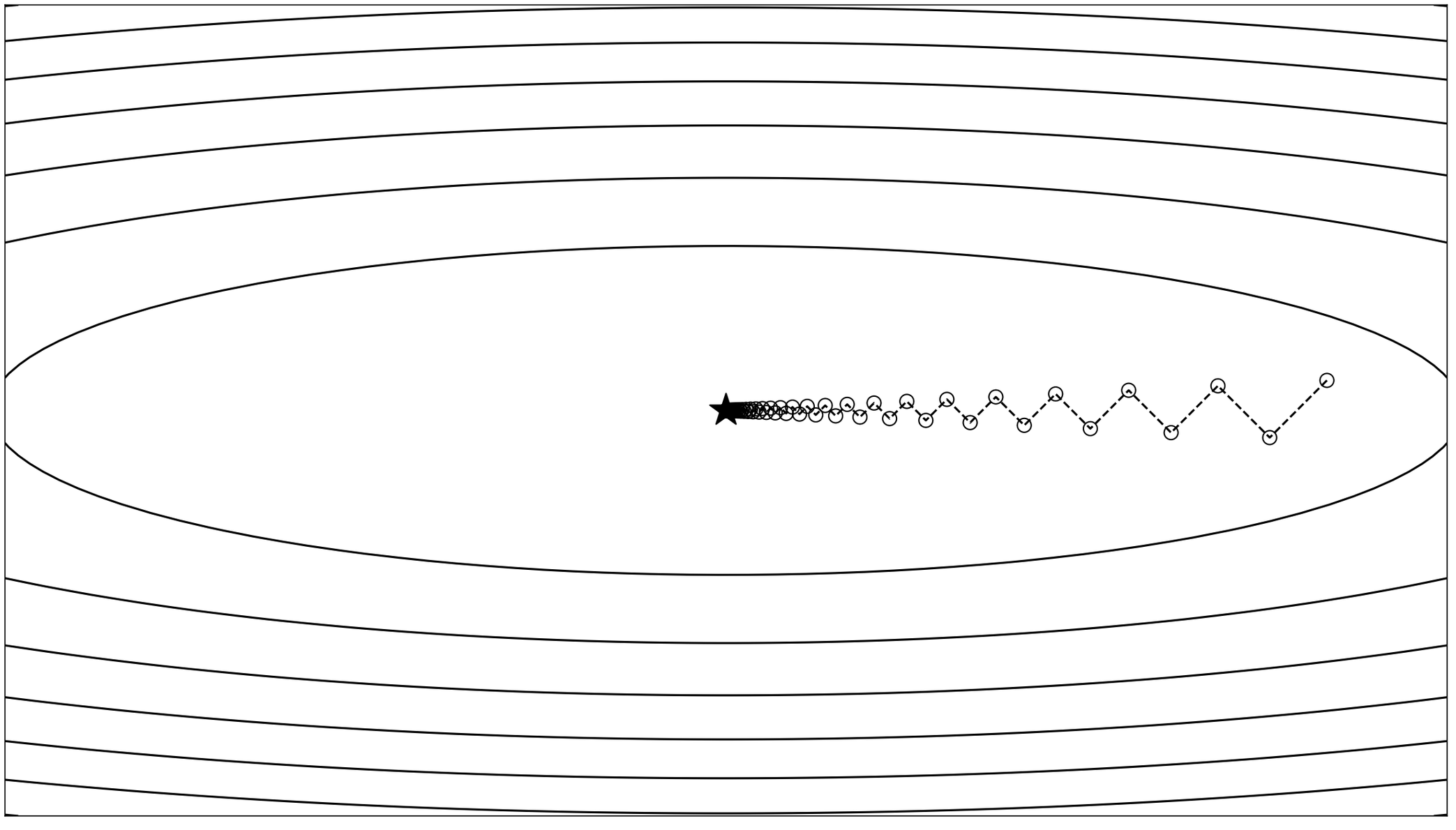
Fixed step sizes

$$f(x) = (x_1^2 + 20x_2^2)/2$$

$$x^0 = (20, 1)$$

$$t = 2/(1 + 20) = 0.0952$$

Optimal



Projected gradient descent

Goal : find a minimiser of a function:

$$\min f(x) + \mathcal{I}_C(x)$$

Where C is the **indicator function** of a set.

$$\mathcal{I}_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{otherwise} \end{cases}$$

Same thing as:

$$\begin{aligned} \min \quad & f(x) \\ \text{subject to:} \quad & x \in C \end{aligned}$$

Can we still use gradient descent?

Projected gradient descent

Goal : find a minimiser of a function:

$$\min f(x) + \mathcal{I}_C(x)$$

A simple modification can be applied....

Gradient descent

$$x^{k+1} = x^k - \alpha \nabla f(x^k)$$

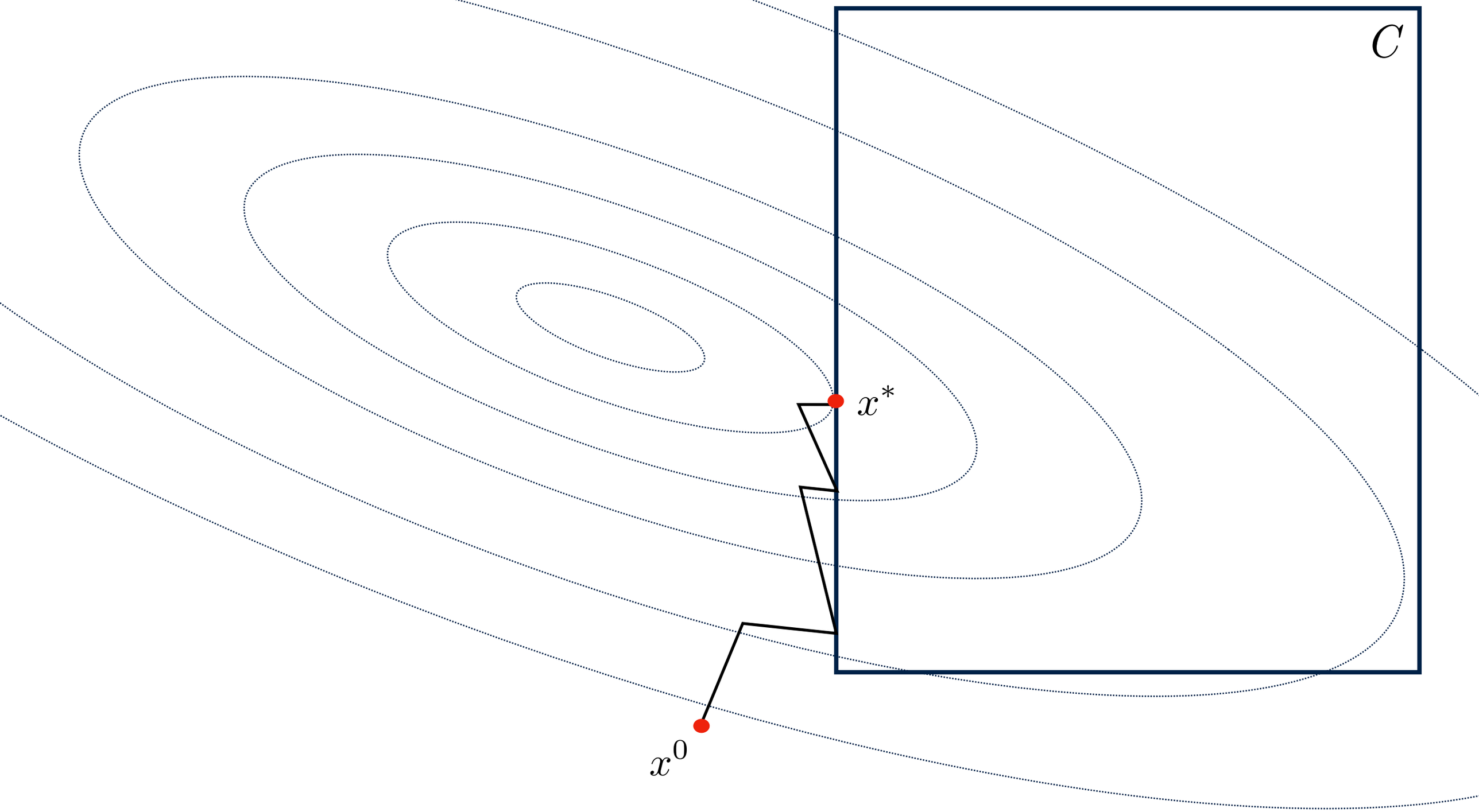
Projected gradient descent

$$x^{k+1} = \Pi_C \left(x^k - \alpha \nabla f(x^k) \right)$$

The projection operator

$$\Pi_C(z) = \arg\min_{x \in C} \|x - z\|^2$$

Projected gradient descent



Operator Splitting Methods

Optimisation using operator splitting

Many optimisation problems can be written like this:

$$\text{minimize} \quad f(w) + g(w)$$

$$\Downarrow$$

$$\begin{aligned} \text{min} \quad & f(\tilde{w}) + g(w) + \frac{\rho}{2}\|w - \tilde{w}\|^2 \\ \text{subject to} \quad & \tilde{w} = w \end{aligned}$$

$$\Downarrow$$

$$\max_{y} \left[\min_{(w,\tilde{w})} \; f(\tilde{w}) + g(w) + \frac{\rho}{2}\|w - \tilde{w}\|^2 + \langle y, w - \tilde{w} \rangle \right]$$

Optimisation using operator splitting

Solving the dual problem :

$$\max_{y} \left[\min_{(w,\tilde{w})} \; f(\tilde{w}) + g(w) + \frac{\rho}{2}\|w - \tilde{w}\|^2 + \langle y, w - \tilde{w}\rangle \right]$$

Gradient ascent (difficult way):

$$(w^{k+1}, \tilde{w}^{k+1}) \leftarrow \arg\min_{(w,\tilde{w})} \left(f(\tilde{w}) + g(w) + \frac{\rho}{2} \left\| \tilde{w} - w + \frac{y^k}{\rho} \right\|^2 \right)$$

$$y^{k+1} \leftarrow y^k + \rho \left(\tilde{w}^{k+1} - w^{k+1} \right)$$

Alternating Direction Method of Multipliers (ADMM)

Solving the dual problem :

$$\max_{y} \left[\min_{(w,\tilde{w})} \; f(\tilde{w}) + g(w) + \frac{\rho}{2}\|w - \tilde{w}\|^2 + \langle y, w - \tilde{w}\rangle \right]$$

Gradient ascent (easy way / ADMM):

1. $$\tilde{w}^{k+1} \leftarrow \arg\min_{\tilde{w}} \left(f(\tilde{w}) + \frac{\rho}{2} \left\| \tilde{w} - w^k + \frac{y^k}{\rho} \right\|^2 \right)$$

2. $$w^{k+1} \leftarrow \arg\min_{w} \left(g(w) + \frac{\rho}{2} \left\| w - \tilde{w}^{k+1} - \frac{y^k}{\rho} \right\|^2 \right)$$

3. $$y^{k+1} \leftarrow y^k + \rho \left(\tilde{w}^{k+1} - w^{k+1} \right)$$

A standard-form convex optimisation problem

Consider convex conic problems in the following form:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2} x^\top P x + q^\top x \\ \text{subject to} \quad & Ax \in \mathcal{C} \end{aligned}$$

- P is positive semidefinite (could even be zero)
- $\mathcal{C}$ is convex in the form $\mathcal{C} = \mathcal{B} \times \mathcal{K}_b$
- Include linear/quadratic programs, second-order cone problems, SDPs

- **QP** : $\mathcal{C}$ is a (translated) box or positive orthant.
- **SDP** : $\mathcal{C}$ is the (translated) positive semidefinite cone.

A standard-form convex optimisation problem

$$
\begin{aligned}
&\text{minimize} && \tfrac{1}{2}x^\top P x + q^\top x \\
&\text{subject to} && Ax = z, \quad z \in \mathcal{C}
\end{aligned}
$$

$\Downarrow$

$f(\tilde{w})$ $\qquad$ $g(w)$

$$
\begin{aligned}
\min \quad & \tfrac{1}{2}\tilde{x}^\top P \tilde{x} + q^\top \tilde{x} + \mathcal{I}_{A\tilde{x}=\tilde{z}}(\tilde{x},\tilde{z}) + \mathcal{I}_{\mathcal{C}}(z) \\
\text{s.t.} \quad & (\tilde{x},\tilde{z}) = (x,z)
\end{aligned}
$$

$$\tilde{w} = w$$

ADMM in a nutshell

We solve this problem via ADMM:

$$\begin{aligned} \text{minimize} \quad & \tfrac{1}{2} x^\top P x + q^\top x \\ \text{subject to} \quad & Ax = z, \quad z \in \mathcal{C} \end{aligned}$$

Inner QP

1 $(x^{k+1}, \tilde{z}^{k+1}) \leftarrow \underset{(x,z):Ax=\tilde{z}}{\arg\min} \frac{1}{2} x^T P x + q^T x + \frac{\sigma}{2} \left\| x - x^k \right\|^2 + \frac{\rho}{2} \left\| \tilde{z} - z^k + \frac{y^k}{\rho} \right\|^2$

2 $z^{k+1} \leftarrow \Pi_{\mathcal{C}} \left(\tilde{z}^{k+1} + \frac{y^k}{\rho} \right)$

Projection onto $\mathcal{C}$

3 $y^{k+1} \leftarrow y^k + \rho \left(\tilde{z}^{k+1} - z^{k+1} \right)$

Optimization projects in EngSci

OSQP solver documentation

Join our forum for any questions related to the solver!

The OSQP (Operator Splitting Quadratic Program) solver is a numerical optimization package for solving convex quadratic programs in the form

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + q^T x \\ \text{subject to} & l \leq Ax \leq u \end{array}$$

where x is the optimization variable and $P \in \mathbf{S}^n_+$ a positive semidefinite matrix.

Code available on GitHub.

Citing OSQP

If you are using OSQP for your work, we encourage you to

- Cite the related papers
- Put a star on GitHub Star 1.1k

We are looking forward to hearing your success stories with OSQP! Please share them with us.

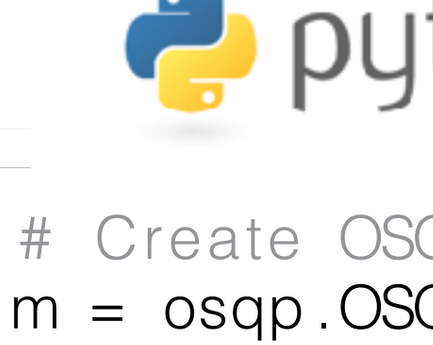
Features

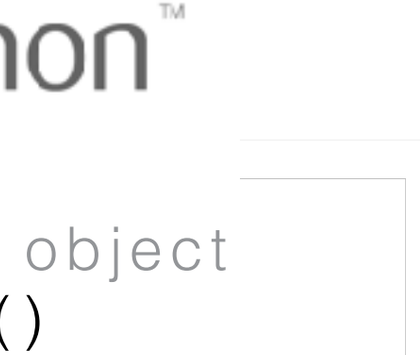
Efficient

It uses a custom ADMM-based first-order method requiring only a single matrix factorization in the setup phase. All the other operations are extremely cheap. It also implements custom sparse linear algebra routines exploiting structures in the problem data.

Optimisation on embedded systems

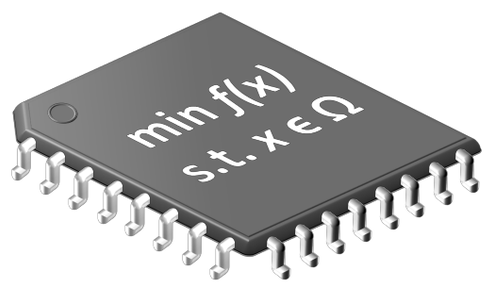
```
# Create OSQP object
m = osqp.OSQP()

# Initialize solver
m.setup(P, q, A, l, u,
        settings)

# Generate C code
m.codegen('folder_name')
```





Optimized C code

Embedded hardware



OSQP Users

Google
Massachusetts Institute of Technology
Stanford University
UC DAVIS UNIVERSITY OF CALIFORNIA
University of Stuttgart Germany

LinkedIn
Berkeley UNIVERSITY OF CALIFORNIA
UC San Diego
Duke UNIVERSITY
LUND UNIVERSITY

Adobe
Baidu 百度
Los Alamos NATIONAL LABORATORY
diamond
OPTIMAL

mobike
XILINX
APTIV
CITADEL

TOYOTA
lyft
BLUE CANYON TECHNOLOGIES
MACQUARIE
BlackRock

Clarabel jl/rs

Search docs

Home
- Features
- Credits
- License

User Guides
- Julia
- Rust
- Python

Examples
- Basic QP Example
- Basic SOCP Example

Citing Clarabel

Contributing

API Reference
- Solver Settings
- Julia API
- Rust API

Version v0.3.0

Clarabel is an interior point numerical solver for convex optimization problems using a novel homogeneous embedding. The Clarabel package solves the following problem:

$$\begin{aligned} \text{minimize} \quad & \frac{1}{2}x^T P x + q^T x \\ \text{subject to} \quad & Ax + s = b \\ & s \in \mathcal{K} \end{aligned}$$

with decision variables $x \in \mathbb{R}^n$, $s \in \mathbb{R}^m$ and data matrices $P = P^\top \succeq 0, q \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, and $b \in \mathbb{R}^m$. The convex set $\mathcal{K}$ is a composition of convex cones.

Clarabel is available in either a native Julia or a native Rust implementation, with a Python interface also available for the Rust version.

Features

- Versatile: Clarabel solves linear programs (LPs), quadratic programs (QPs), second-order cone programs (SOCPs), and problems with exponential and power cone constraints. The Julia version also solves semidefinite programs (SDPs).
- Quadratic objectives: Unlike interior point solvers based on the standard homogeneous self-dual embedding (HSDE) model, Clarabel handles quadratic objective without requiring any epigraphical reformulation of its objective function. It can therefore be significantly faster than other HSDE-based solvers for problems with quadratic objective functions.
- Infeasibility detection: Infeasible problems are detected using using a homogeneous embedding technique.
- Arbitrary precision types: You can solve problems with any floating point precision, e.g. Float32 or Julia's BigFloat type in Julia and f32 or f64 types in Rust.
- Open Source: Our code is available on GitHub and distributed under the Apache 2.0 License. The Julia implementation is here. The Rust implementation and Python interface is here.

Credits

The following people are involved in the development of Clarabel:

- Paul Goulart (main development, maths and algorithms)
- Yuwen Chen (maths and algorithms)

All contributors are affiliated with the Control Group of the Department of Engineering Science at the University of

Thanks