Derivative Free Optimization

joint course between
Optimization Master Paris Saclay - AMS Master
2025/2026

Anne Auger
RandOpt team
Inria and CMAP, Ecole Polytechnique, IP Paris
anne.auger@inria.fr

Organization of the class

When: Friday afternoon - 2pm - 5:15pm at ENSTA

28/11/2025	room 1322
05/12/2025	room 1322
12/12/2025	room 1322
19/12/2025	room 1322
09/01/2026	room 1322
16/01/2026	room 1322
23/01/2026	room 1322
30/01/2026	room 1322
06/02/2026	room 1322
13/02/2026 [EXAM]	TBA

Evaluation

Written exam on 14/02/2025

Project (in group) to be discussed

Syllabus

Topics covered

Derivative Free Optimization / Black-box optimization

- Single-objective optimization
  - what makes a problem difficult
  - algorithm to solve those difficulties (mostly stochastic)
- Multi-objective optimization
- Benchmarking

Practical Exercices

practical exercices: implement/manipulate algorithms

Python / Matlab / ...

ultimate goal: optimize a (real) black-box problem on your own

- understand and visualize convergence / adaptation / invariance
- experience numerics — numerical errors, finite machine precision

Objectives

- understand how numerical optimization works when derivatives are unavailable
- identify the main challenges, typical failure modes
- learn how they are addressed in state-of-the-art methods
- teach about scientific / research approach in optimization

Disclaimer & Focus

- This course does not aim at exhaustive coverage
- It emphasizes a major class of derivative-free methods: Evolution Strategies, with a strong focus on CMA-ES

CMA-ES is:

one widely used derivative-free solvers worldwide
supported by Python implementations exceeding 80 million downloads
one of which will be used hands-on during the course

Derivative-Free / Black-box Optimization

Task: minimize a numerical **objective** function (also called *fitness* function or *loss* function) $K \in \mathbb{N}$

$$f : \Omega \subset \mathbb{R}^n \to \mathbb{R}^{K}, x \mapsto f(x) \in \mathbb{R}$$

$K=1$ single-objective optimization

$K>1$: multi-objective optimization.

without derivatives (gradient). Ω: search space, n :dimension of the search space

Also called **zero-order black-box** optimization

The function is seen by the algorithm as a zero-order **oracle** [a first order oracle would also return gradients] that can be queried at points and the oracle returns an answer

First order (black-box): methods that are using ∇f.

- → Gradient descent
- → Conjugate gradient
- → quasi-Newton (BFGS, ...)

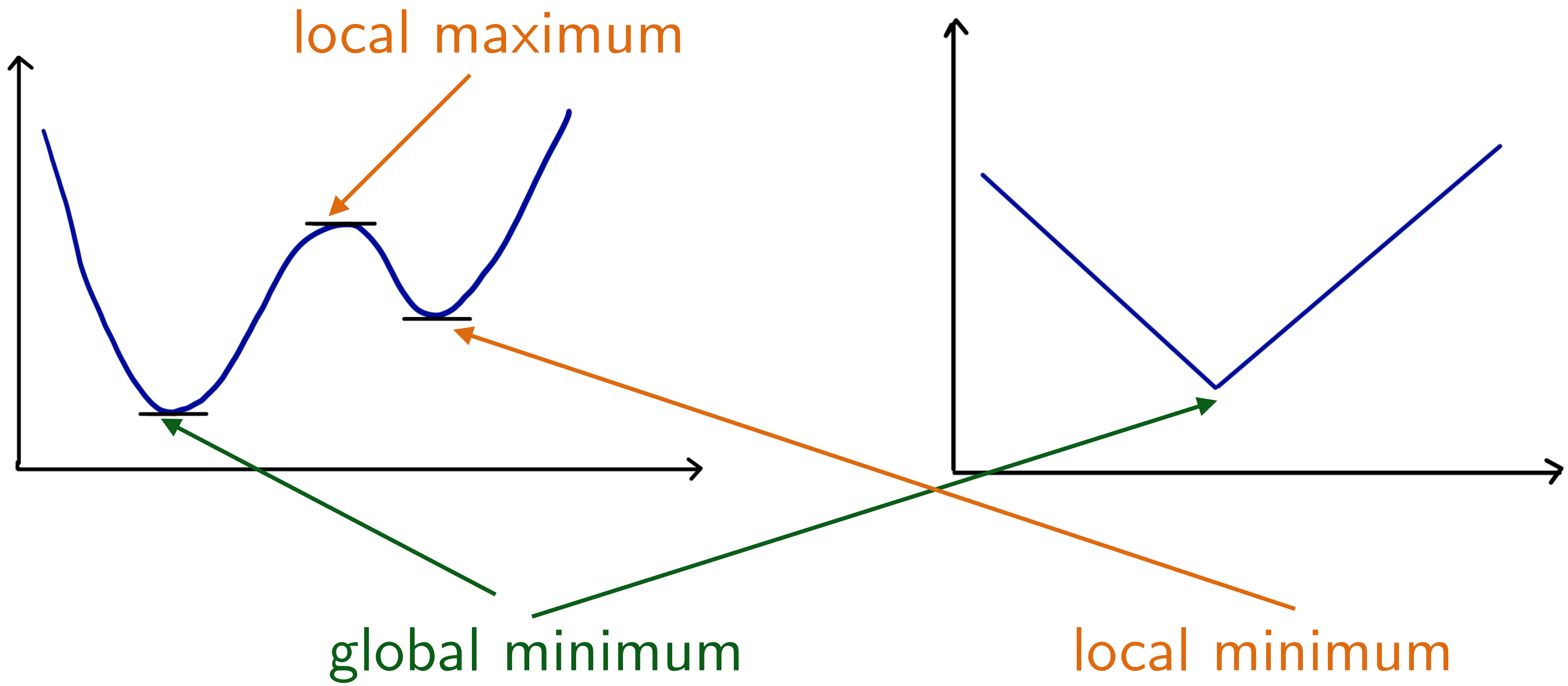
Second order: $\nabla^2 f$, ∇f Newton

Newton direction: $-\nabla^2 f(x) \nabla f(x)$

(followed by Newton-method)

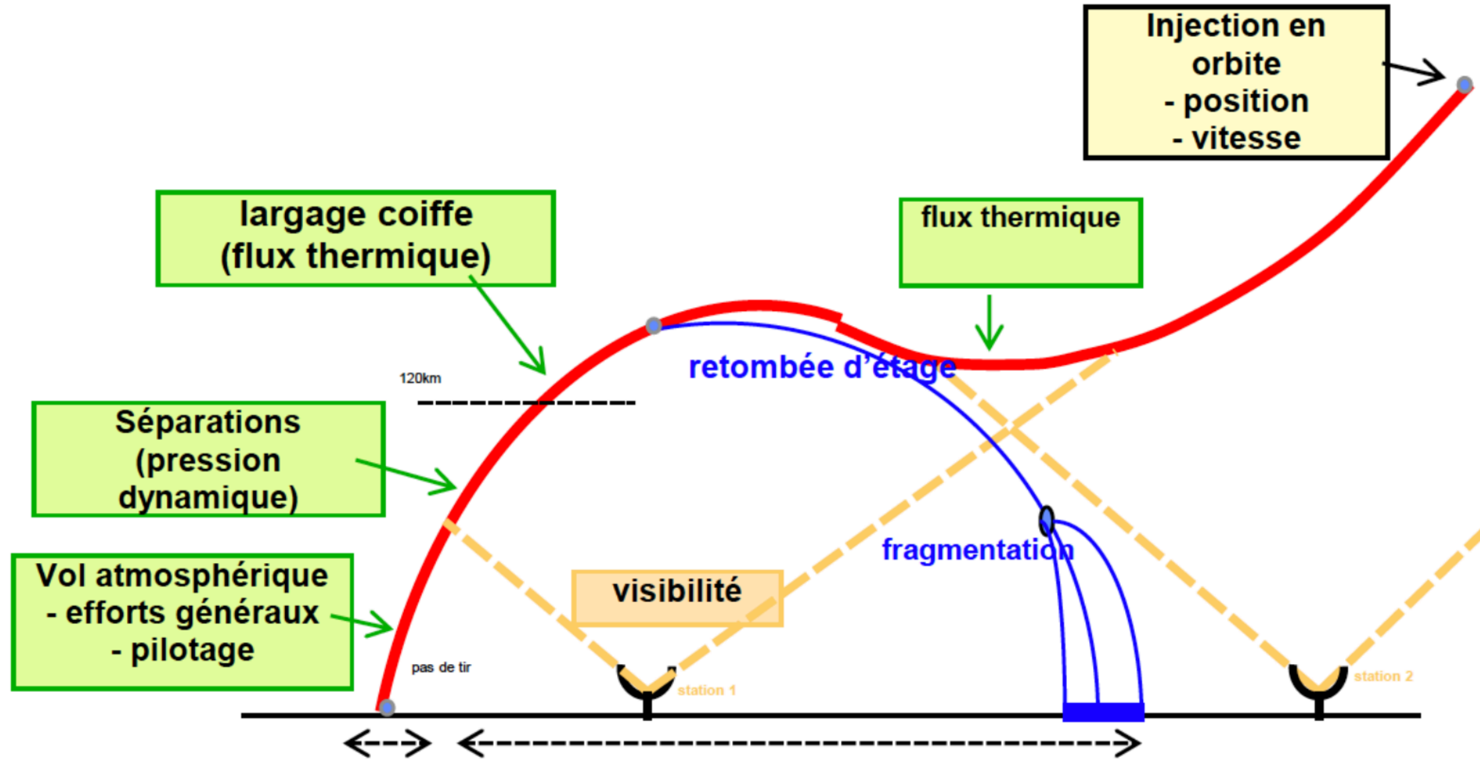
Reminder: Local versus Global Optimum

n=1



Examples: Optimization of the Design of a Launcher

Poppy



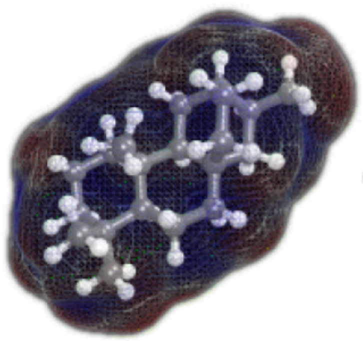
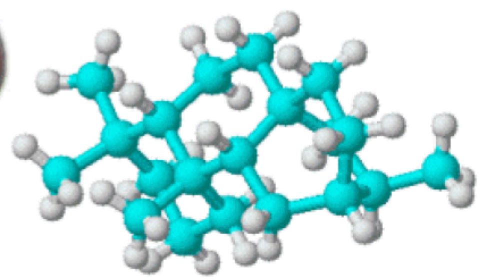
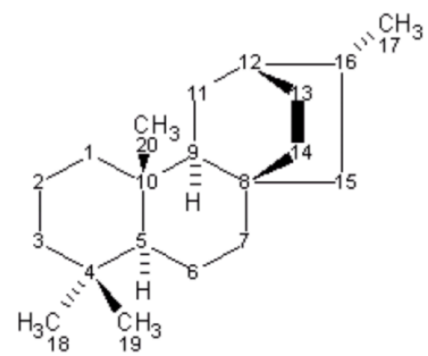
- Scenario: multi-stage launcher brings a satellite into orbit
- Minimize the overall cost of a launch
- Parameters: propellant mass of each stage / diameter of each stage / flux of each engine / parameters of the command law

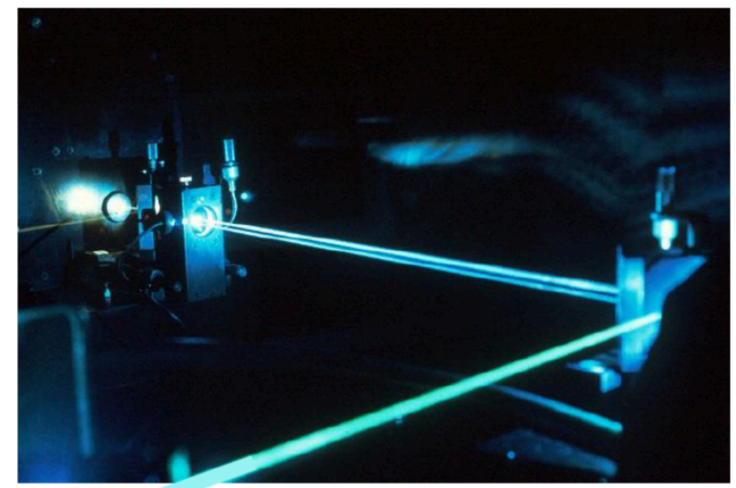
23 continuous parameters to optimize

+ constraints

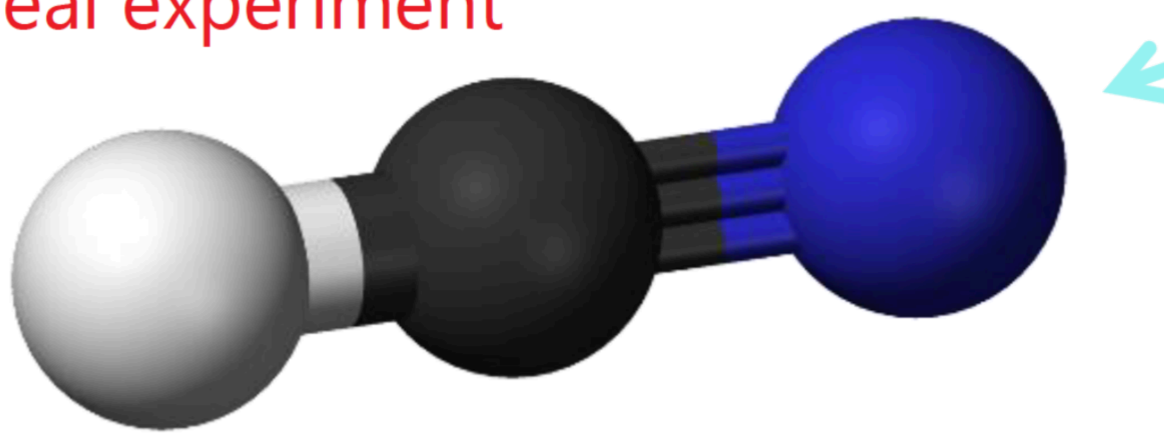
Control of the Alignement of Molecules

application domain: quantum physics or chemistry



Objective function:
via numerical simulation
or a real experiment



possible application in drug design

In the case of a real lab experiment: the objective function is a real black-box

Coffee Tasting Problem (A real Black-box)

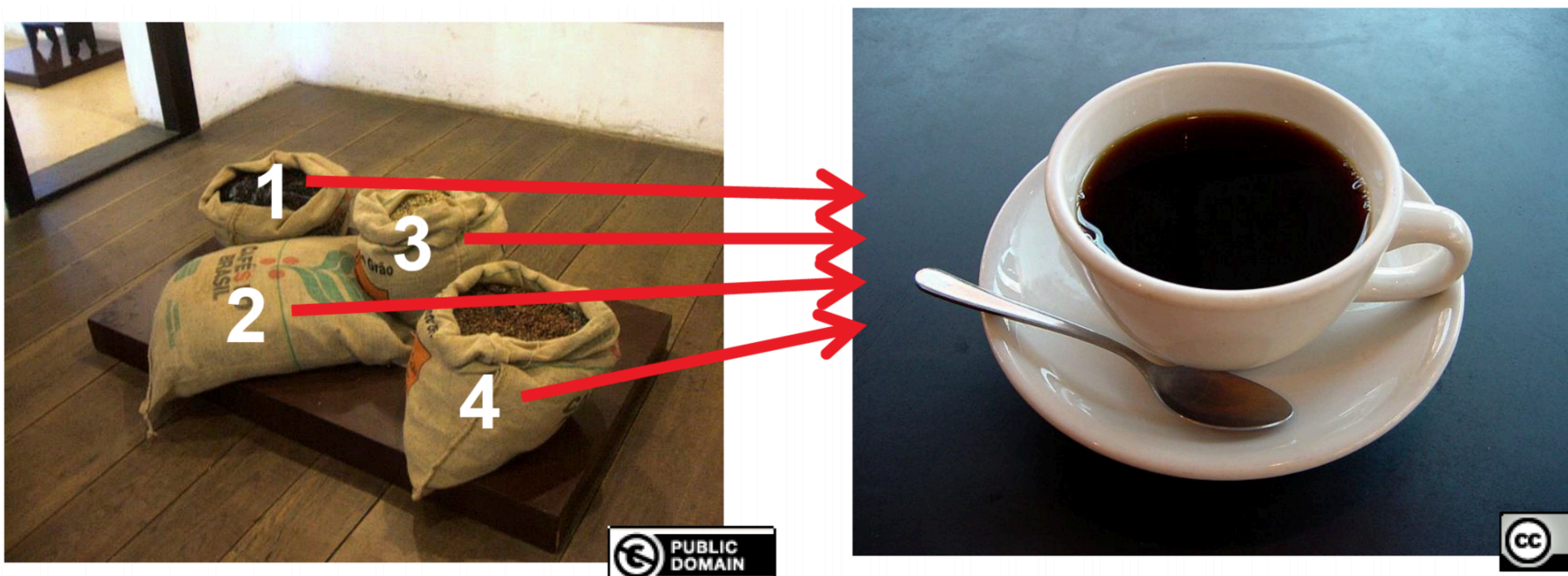
Coffee Tasting Problem

$x_i \geq 0$ $\quad$ x_i : quantity of coffee i

$\sum x_i = 1$

- Find a mixture of coffee in order to keep the coffee taste from one year to another
- Objective function = opinion of one expert

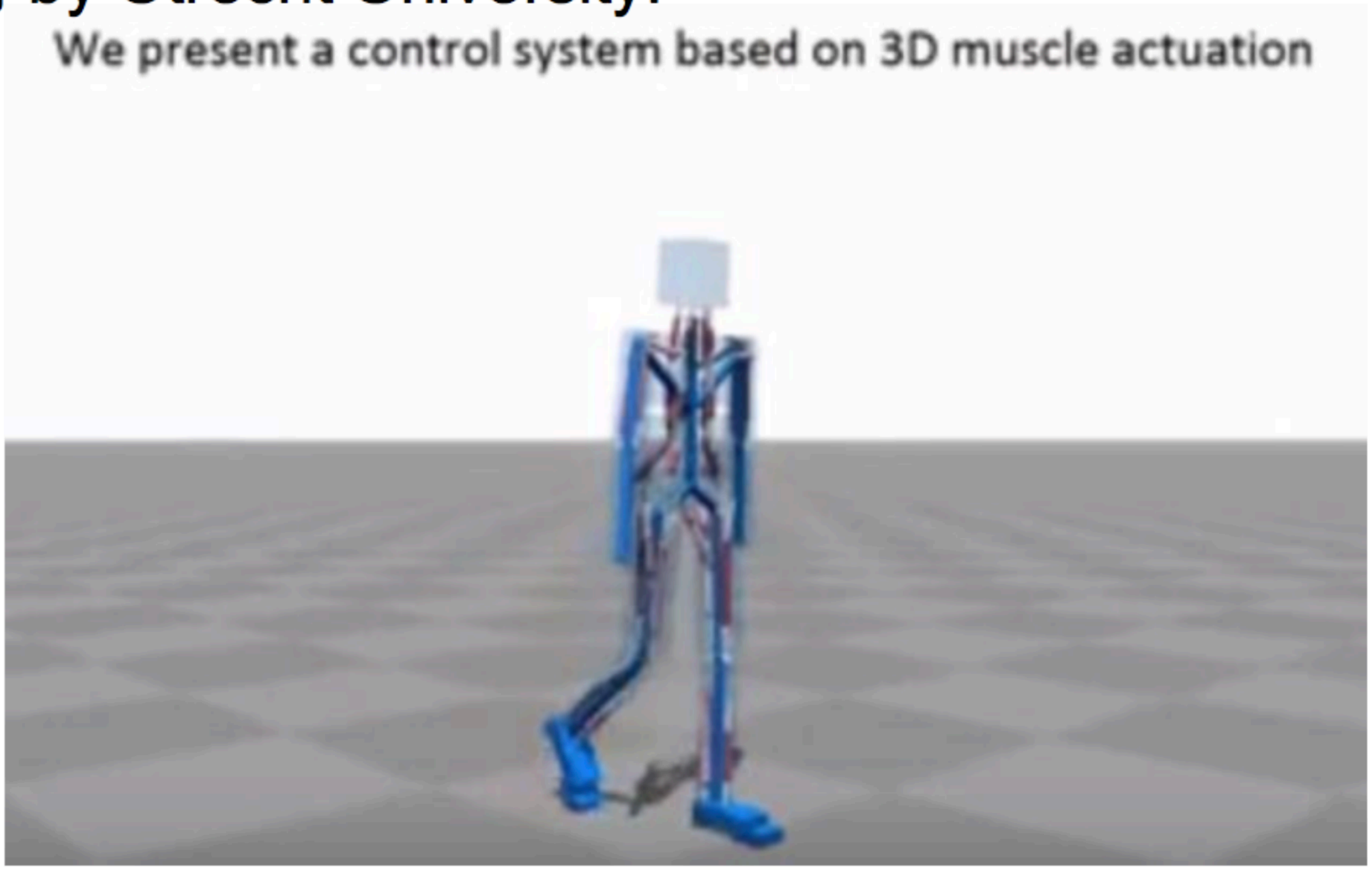
$(x_1, \dots, x_n) \rightarrow (\text{Taste}(x) - \text{Taste prev. year})^2$



M. Herdy: *"Evolution Strategies with subjective selection"*, 1996

A last Application

Computer simulation teaches itself to walk upright (virtual robots (of different shapes) learning to walk, through stochastic optimization (CMA-ES)), by Utrecht University:



https://www.youtube.com/watch?v=yci5FuI1ovk

T. Geitjtenbeek, M. Van de Panne, F. Van der Stappen: "Flexible Muscle-Based Locomotion for Bipedal Creatures", SIGGRAPH Asia, 2013.

What is the Goal?

- We want to find $x^\star$ such that $f(x^\star) \leq f(x)$ for all x

↳ global minimizer

$x^\star \in \text{argmin}_x f(x)$

- In general we will never find $x^\star$

why?

What is the Goal?

- We want to find $x^\star$ such that $f(x^\star) \leq f(x)$ for all x

$$x^\star \in \text{argmin}_x f(x)$$

- In general we will never find $x^\star$

- Because of the numerical/continuous nature of the search space we typically never hit exactly $x^\star$, we instead converge to a solution:

we want to find $x_t \in \mathbb{R}^n$ such that $\lim_{t \to \infty} f(x_t) = \min f$

of course we want ***fast*** convergence

Level Sets of a Function

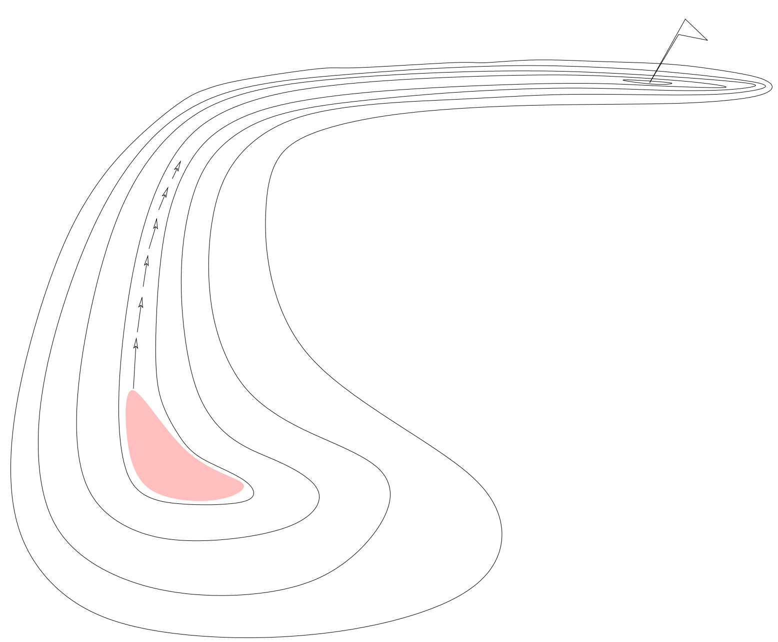
Level Sets: Visualization of a Function

One-dimensional (1-D) representations are often misleading (as 1-D optimization is "trivial", see slides related to curse of dimensionality), we therefore often represent level-sets of functions

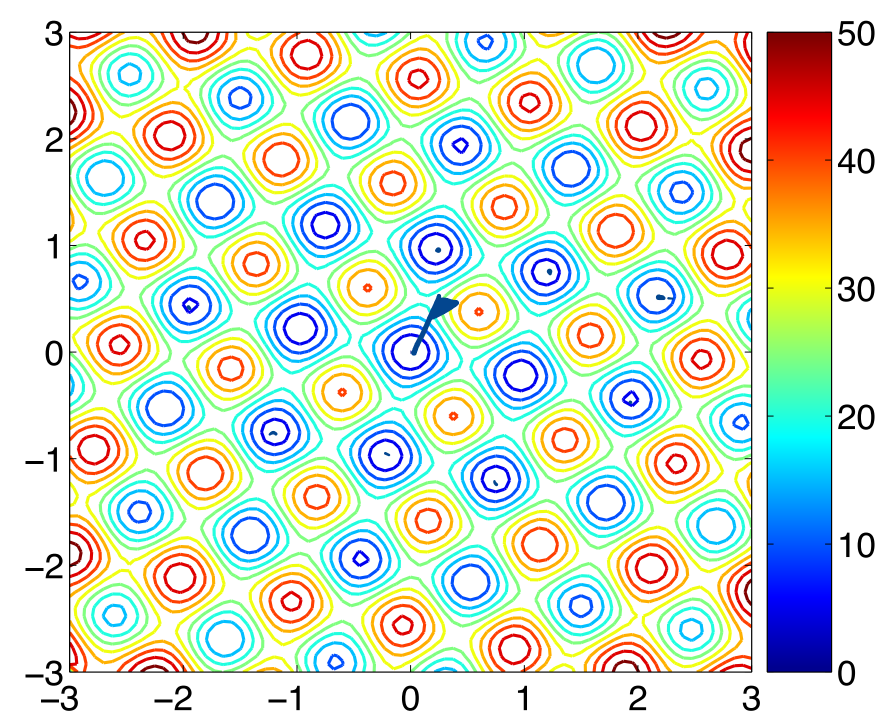
$$\mathcal{L}_c = \{x \in \mathbb{R}^n \,|\, f(x) = c,\}, c \in \mathbb{R}$$

Sublevel sets $\text{Lev}_c^{\leq} = \{x \in \mathbb{R}^n \mid f(x) \leq c\}, c \in \mathbb{R}$

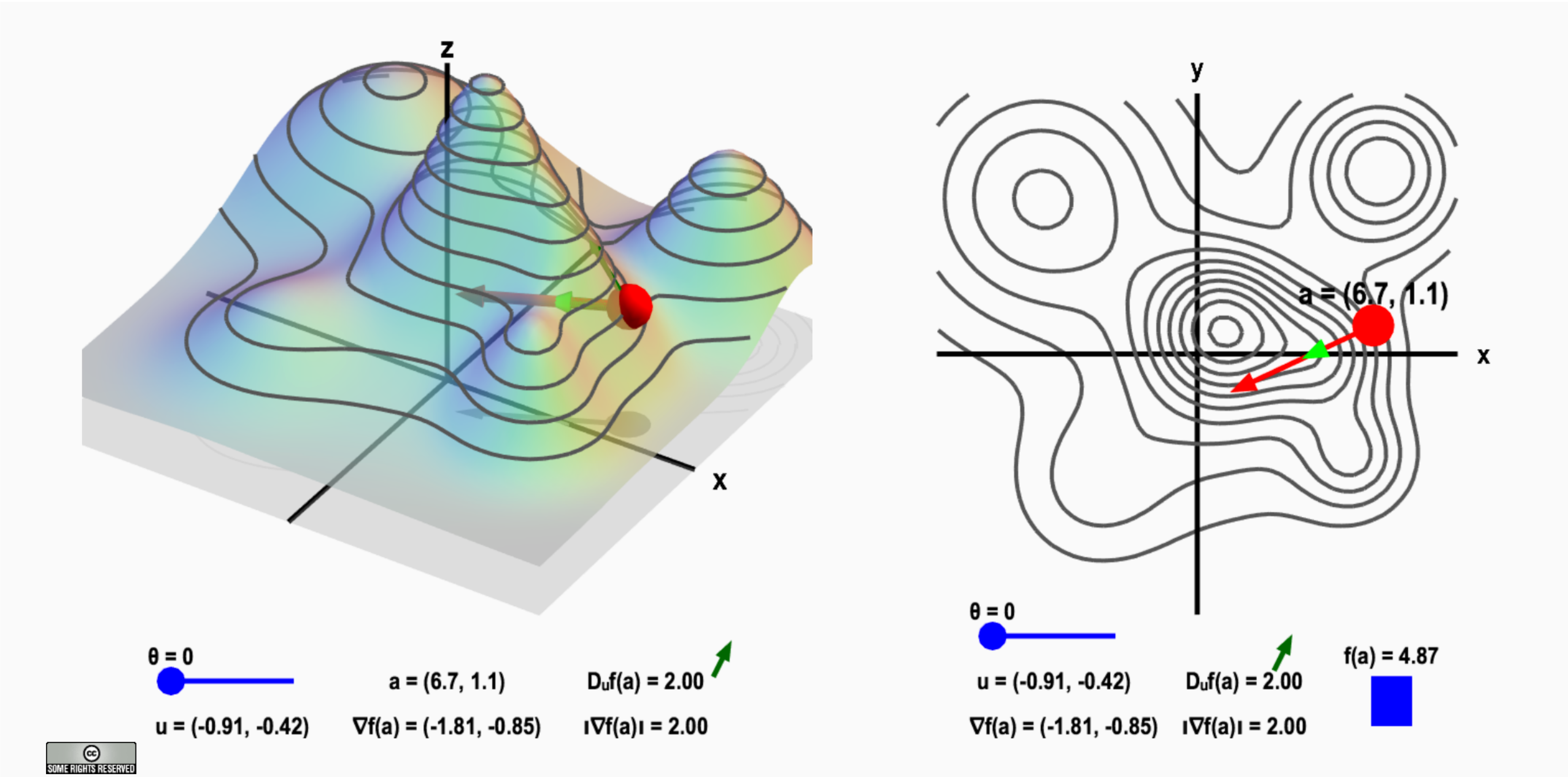
Examples of level sets in 2D



function with plenty local minima



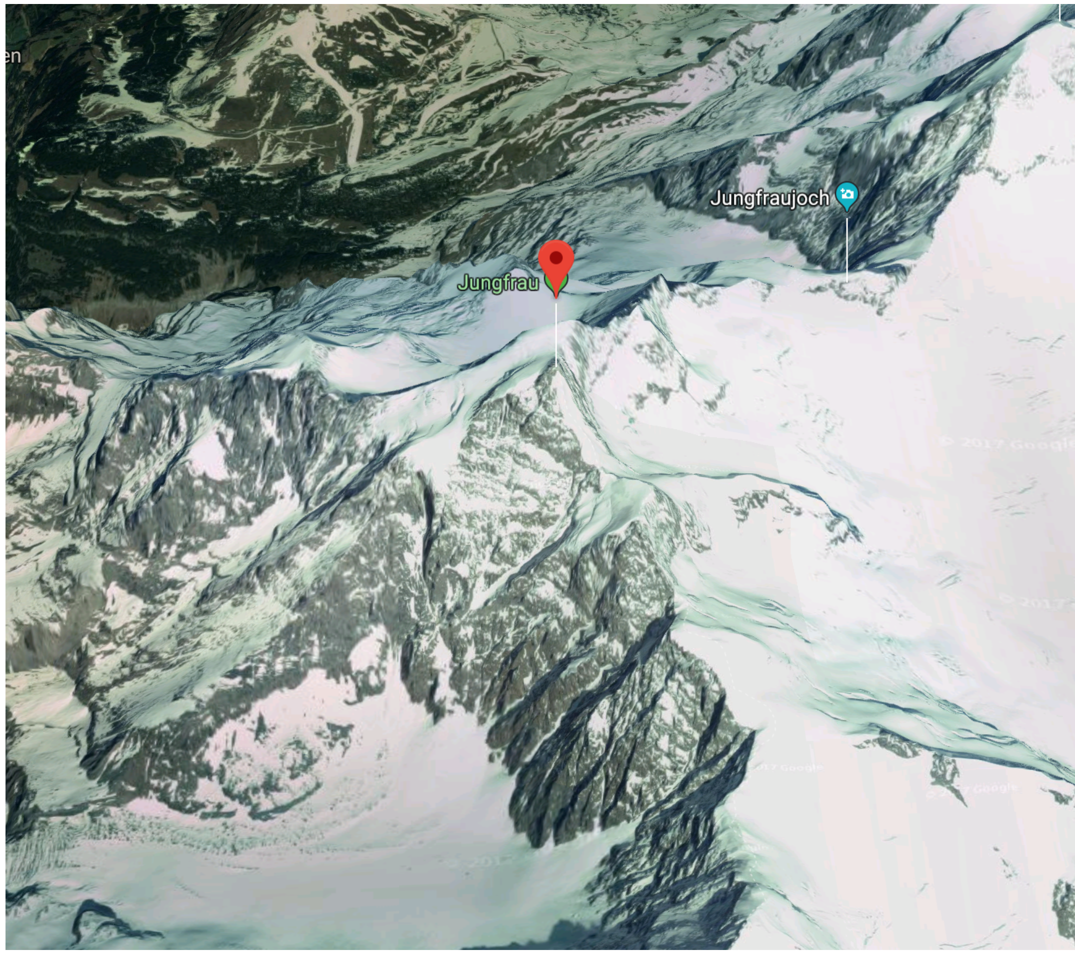
Level Sets: Visualization of a Function



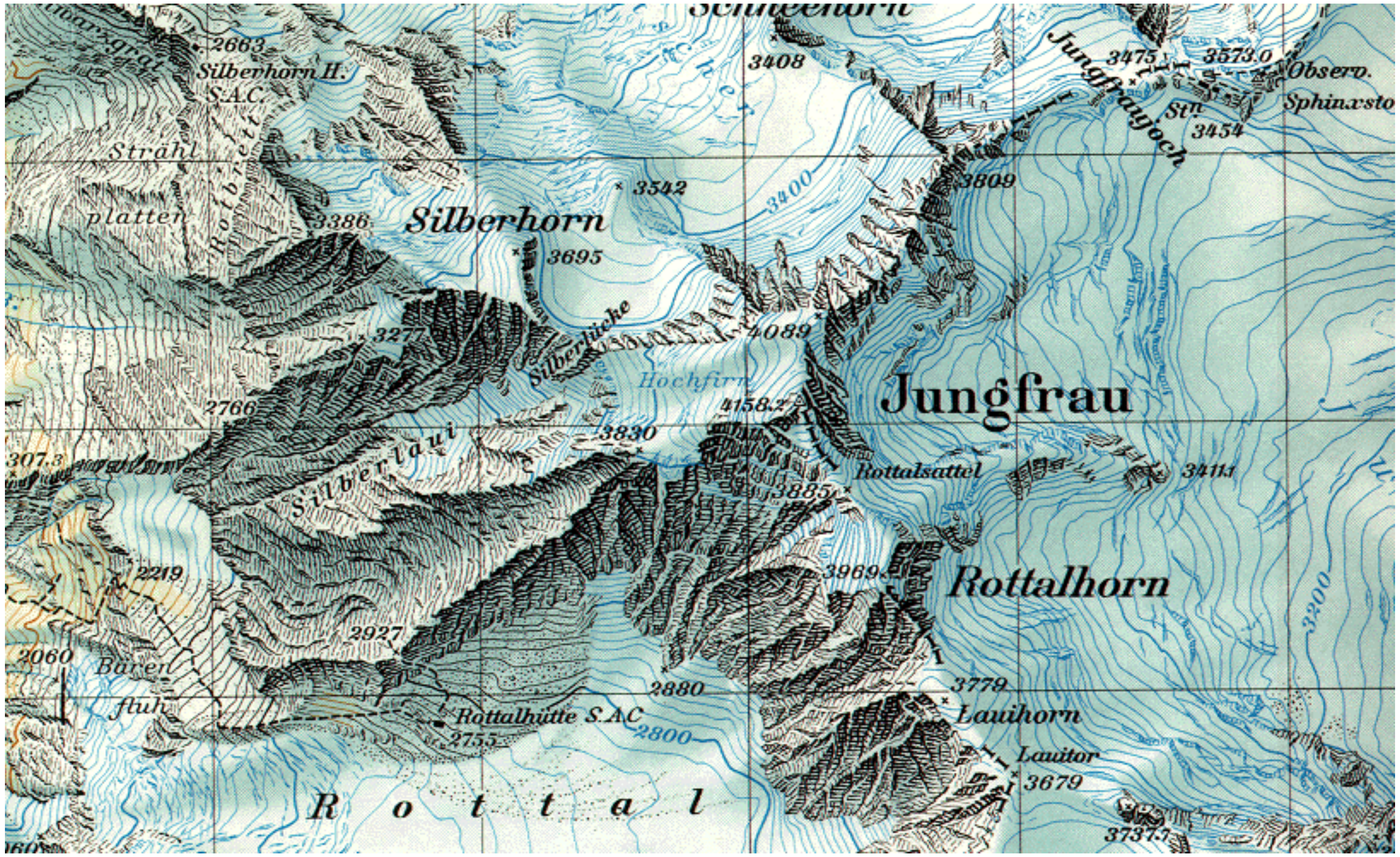
Source: Nykamp DQ, "Directional derivative on a mountain." From *Math Insight*. http://mathinsight.org/applet/directional_derivative_mountain

Level Sets: Topographic Map

The function is the altitude



3-D picture



Topographic map

Level Set: Exercice

Consider a strictly convex-quadratic function

$$f(x) = \frac{1}{2}(x - x^\star)^\top H(x - x^\star) = \frac{1}{2}\sum_i h_{ii}(x_i - x_i^\star) + \frac{1}{2}\sum_{i \neq j} h_{ij}(x_i - x_i^\star)(x_j - x_j^\star)$$

with H a symmetric, positive, definite matrix ($H \succ 0$).

1. What is/are the optima of f ? What does H represent for the function ?

2. Assume n=2, $H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ plot the level sets of f

3. Same question with $H = \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix}$

4. Same question with $H = P \begin{bmatrix} 1 & 0 \\ 0 & 9 \end{bmatrix} P^T$ with $P \in \mathbb{R}^{2\times 2}$

   P orthogonal

1/ $f(x) = \frac{1}{2}(x - x^*)^T H (x - x^*) \geq 0$ since $H \succ 0$

$f(x^*) = 0$, thus $f(x^*) \leq f(x) \quad \forall x$

$\rightarrow x^*$ is the unique minimizer of f.

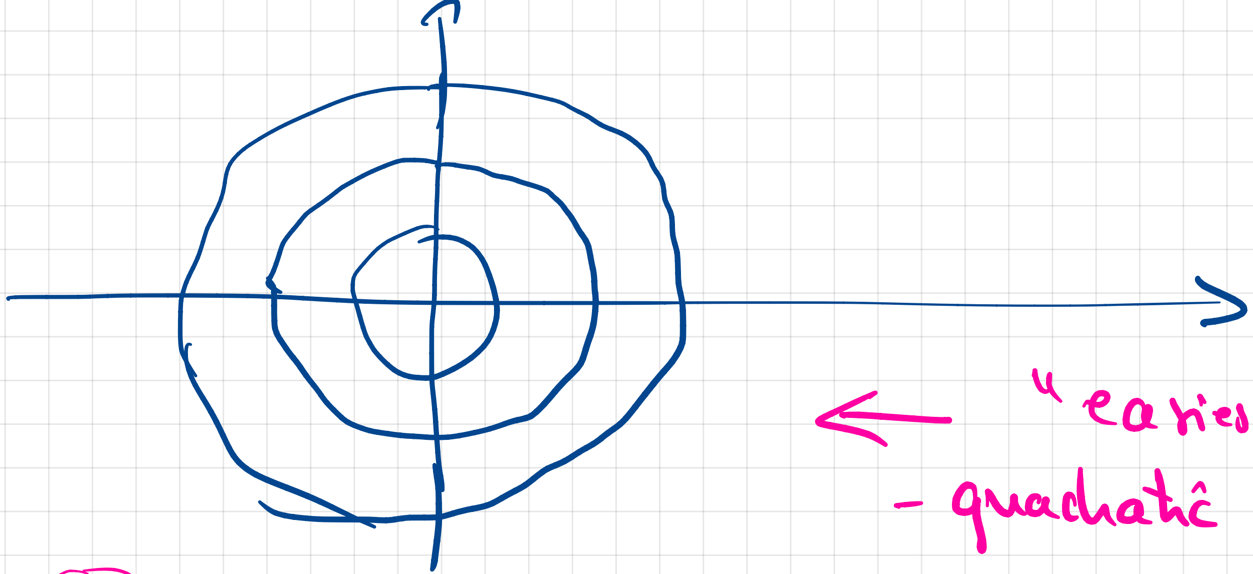
f strictly convex.

H is the Hessian matrix. $D^2 f(x) = H \quad \forall x \in \mathbb{R}^n$

2/ $n = 2$, $f(x) = \frac{1}{2}\left\{(x_1 - x_1^*)^2 + (x_2 - x_2^*)^2\right\}$

$\{x \mid f(x) = c\} \qquad c > 0$

$= \{(x_1, x_2) \mid (x_1 - x_1^*)^2 + (x_2 - x_2^*)^2 = c\}$



"easiest" convex
- quadratic problem

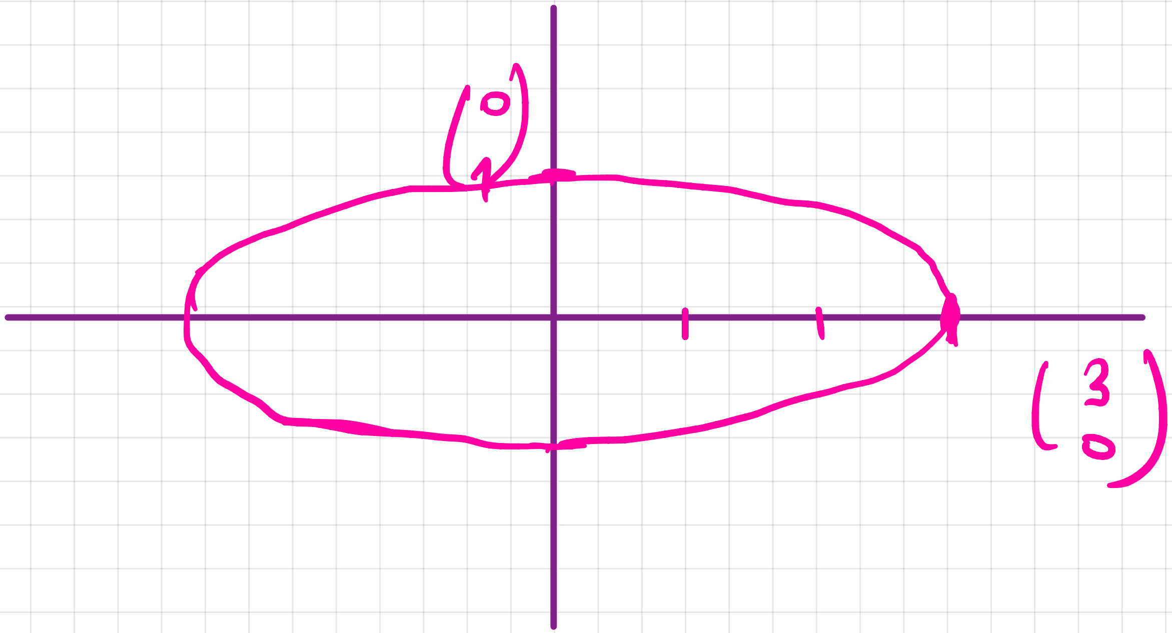
3) If $H = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$

$$\text{Lev}_c^= f = \left\{ (x_1, x_2) \mid \frac{1}{2}\left((x_1 - x_1^*)^2 + 9\,(x_2 - x_2^*)^2 \right) = c \right\}$$

$c > 0$

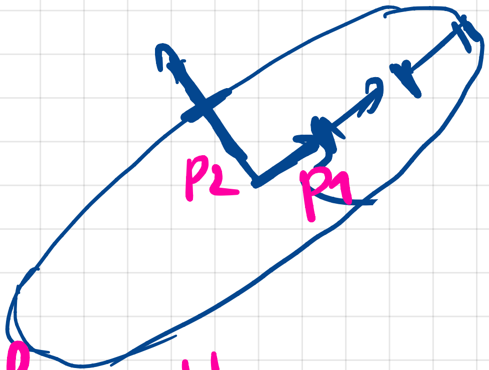
Assume <u>WLG</u> $x^* = 0$

$$\left\{ x = (x_1, x_2) \mid \frac{1}{2}\left(x_1^2 + 9\,x_2^2 \right) = c \right\}$$



4) $H = P \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix} \underline{P}^T$ → Rotated ellipsoid.

$\underline{P} = \left(p_1, \; p_2 \right)$



Main-axis are the eigenvectors of H.

What Makes an Optimization Problem Difficult?

What makes a numerical optimization problem difficult?

1/ Non-convexity

2) Non-smooth

3) High-dimension more than 100

4/ Steep : ill-conditionned

5) Constraints

6) "Expensive" to evaluate → can sometimes take days (crash-simulations)
↳ parallelize
↳ "Bayesian optimization"

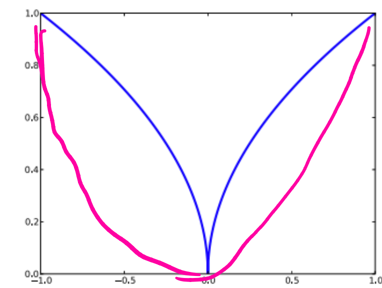
7/ If we do not know if there are optima
↳ It's typically the case: we do not ensure we found the global minimizer but we are happy with "good" local optimum.

What Makes a Function Difficult to Solve?

Why stochastic search?

- non-linear, non-quadratic, non-convex

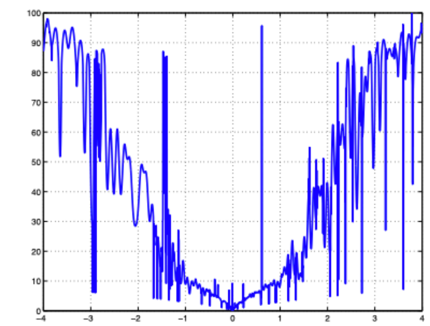
  on linear and quadratic functions much better search policies are available

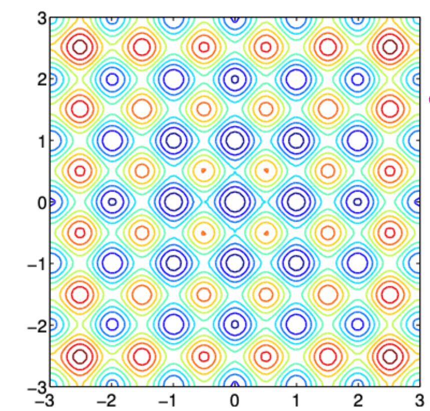


- ruggedness

  non-smooth, discontinuous, multimodal, and/or noisy function

  more than one local optimum





many local optima

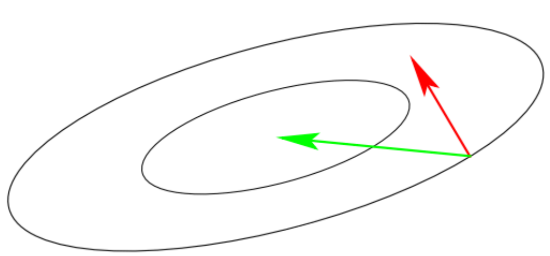
- dimensionality (size of search space)

  (considerably) larger than three

- non-separability

  dependencies between the objective variables

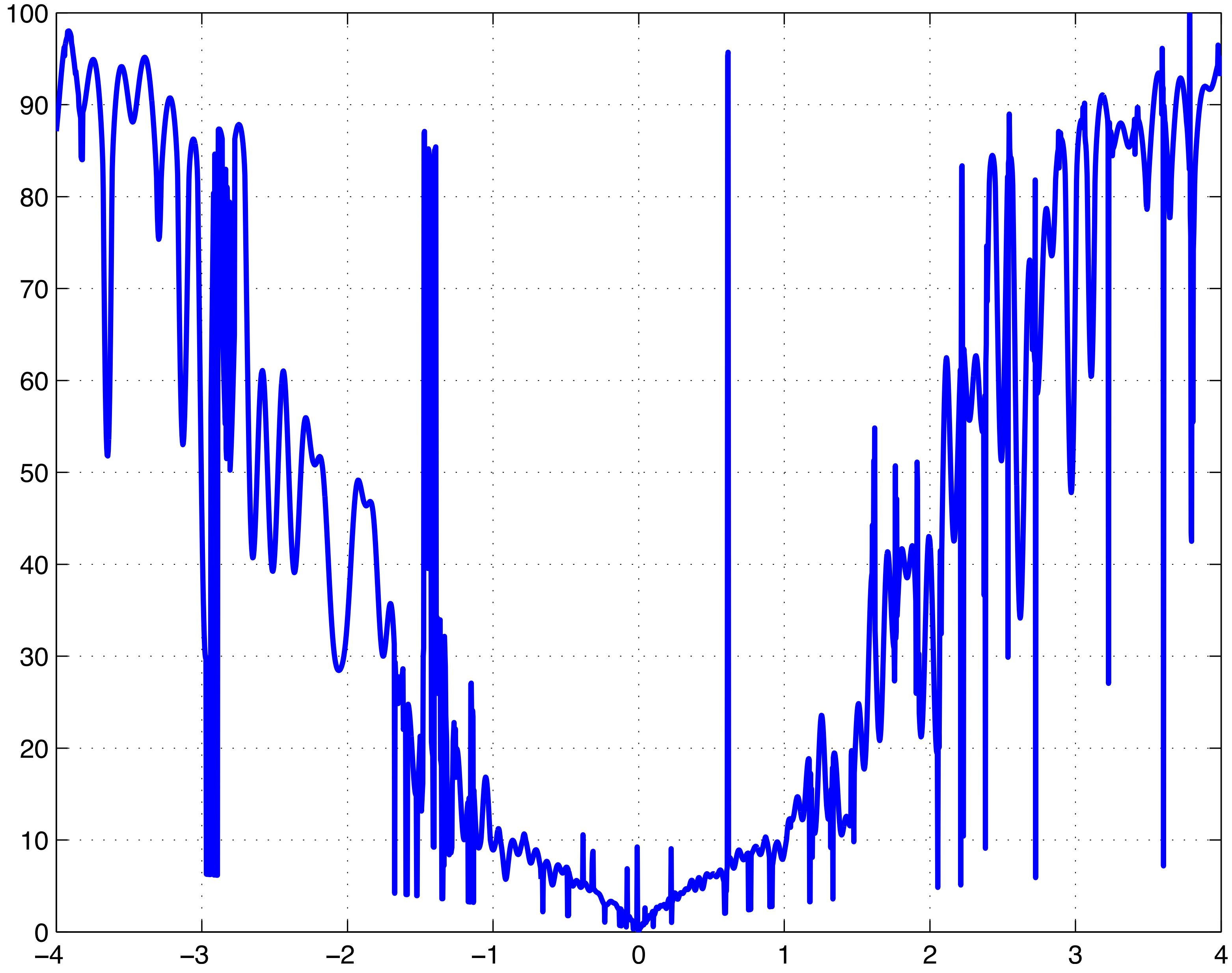
- ill-conditioning



gradient direction Newton directio

In this class:
strong focus on methods to address difficult black-box problems,
that can typically address the difficulties of the previous slide.

Ruggedness



A cut of a 4-D function that can easily be solved with the CMA-ES algorithm

Why is Optimization a non-trivial Problem?

Curse of dimensionality

if n=1, which simple approach could you use to minimize:

$$f : [0,1] \to \mathbb{R} \quad ?$$

Why is Optimization a non-trivial Problem?

Curse of dimensionality

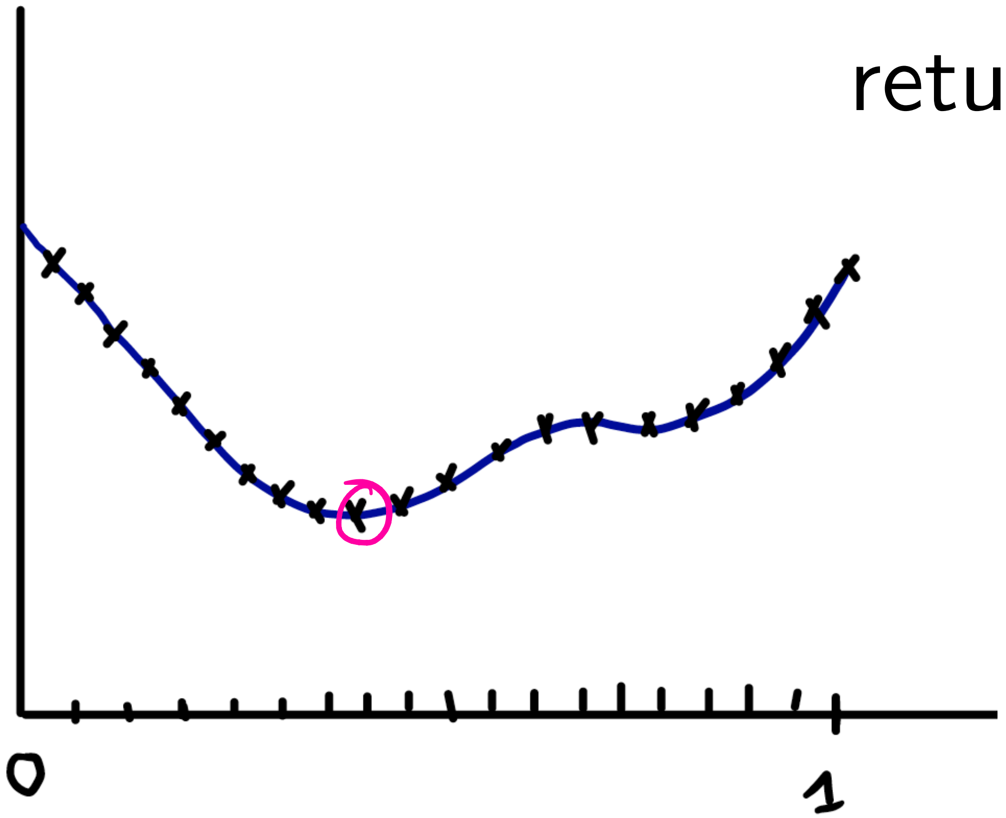
if n=1, which simple approach could you use to minimize:

$$f : [0, 1] \to \mathbb{R} \quad ?$$

set a regular grid on [0,1]

evaluate on f all the points of the grid

return the lowest function value



Why is Optimization a non-trivial Problem?

Curse of dimensionality

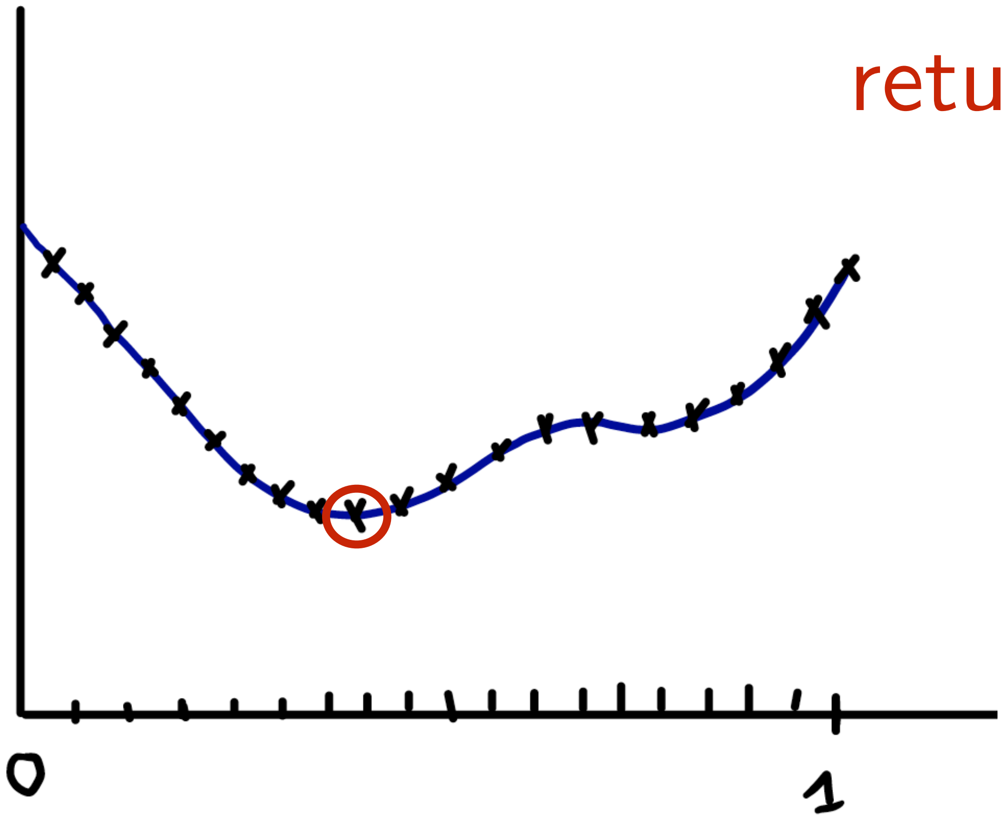
if n=1, which simple approach could you use to minimize:

$$f : [0,1] \to \mathbb{R} \quad ?$$

set a regular grid on [0,1]

evaluate on f all the points of the grid

return the lowest function value



Why is Optimization a non-trivial Problem?

Curse of dimensionality

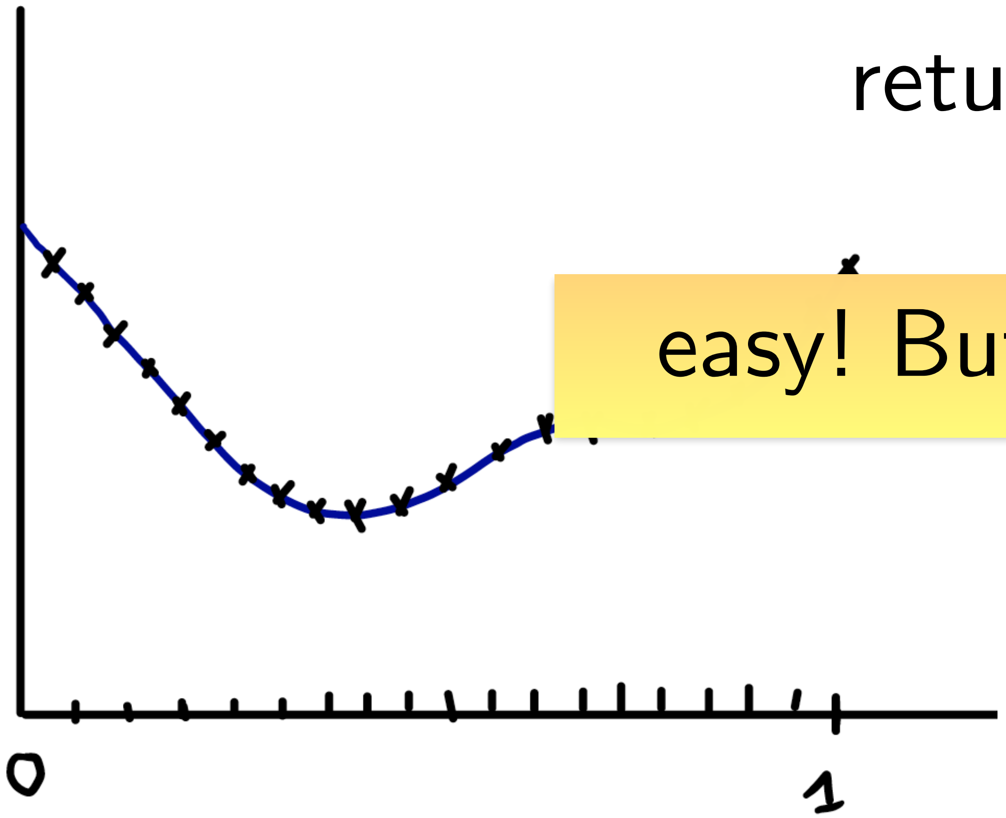
if n=1, which simple approach could you use to minimize:

$$f : [0,1] \to \mathbb{R} \quad ?$$

set a regular grid on [0,1]

evaluate on f all the points of the grid

return the lowest function value



easy! But how does it scale when n increases?

1-D optimization is trivial

Curse of Dimensionality

The term curse of dimensionality (Richard Bellman) refers to problems caused by the rapid increase in volume associated with adding extra dimensions to a (mathematical) space.

Example: Consider placing 100 points onto a real interval, say [0,1].

How many points would you need to get a similar coverage (in terms of distance between adjacent points) in dimension 10?

Curse of Dimensionality

How long would it take to evaluate 10^{20} points?

$$f(x) = \sum_{i=1}^{10} x_i^2$$

Curse of Dimensionality

How long would it take to evaluate 10^{20} points?

```
import timeit
timeit.timeit('import numpy as np ;
np.sum(np.ones(10)*np.ones(10))', number=1000000)
> 7.0521080493927
```

$\sum_{i=1}^{n} x_i^2$ $\quad x = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}$

7 seconds for 10^6 evaluations of $f(x) = \sum_{i=1}^{10} x_i^2$

We would need more than 10^8 days for evaluating 10^{20} points

[As a reference: origin of human species: roughly 6 x 10^8 days]

Separability

Given $x = (x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots x_n)$ denote

$$x^{\neg i} = (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \mathbb{R}^{n-1}$$

$$f_{x^{\neg i}}(y) = f(x_1, \ldots, x_{i-1}, y, x_{i+1}, \ldots, x_n)$$

The function $f_{x^{\neg i}}(y)$ is a 1-D function which is a cut of f along the coordinate i.

Definition: A function f is **separable** if for all i, for all $x, \bar{x}$

$$\mathrm{argmin}_y f_{x^{\neg i}}(y) = \mathrm{argmin}_y f_{\bar{x}^{\neg i}}(y)$$

→ *the optimum along the coordinate i, does not depend on how the other coordinates are fixed.*

a weak definition of separability

Lemma: Given $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathrm{Im}(f) \to \mathbb{R}$ strictly increasing. If f is **separable** then $g \circ f$ is separable.

Proposition: Let f be a **separable** then for all x

$$\text{argmin} f(x_1, \ldots, x_n) = \left(\text{argmin}_y f_{x\neg 1}(y), \ldots, \text{argmin}_y f^n_{x\neg n}(y) \right)$$

and f can be optimized using n minimization along the coordinates.

Exercice: prove the proposition

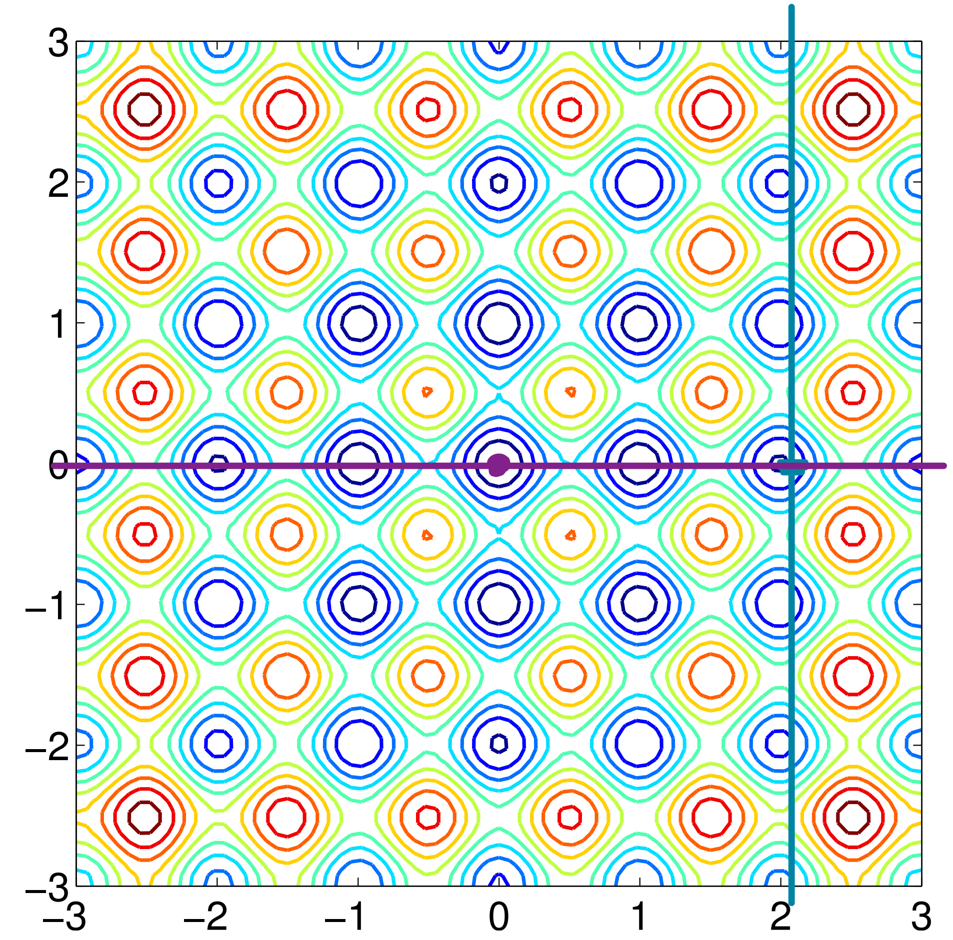
Example: Additively Decomposable Functions

Lemma: Let $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} h_i(x_i)$ for h_i having a unique argmin.

Then f is separable. We say in this case that f is additively decomposable.

Example: Rastrigin function

$$f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$$



Consequence

Consider $f(x) = \prod_{i=1}^{n} h_i(x_i)$ with $h_i(x_i) > 0$. Then it is separable.

Non-separable Problems

Separable problems are typically easy to optimize. Yet **difficult real-word problems are non-separable**.

One needs to be careful when evaluating optimization algorithms that not too many test functions are separable and if so that the *algorithms do not exploit separability*.

Otherwise: *good performance on test problems will not reflect good performance of the algorithm to solve difficult problems*

Algorithms known to exploit separability:

Many Genetic Algorithms (GA), Most Particle Swarm Optimization (PSO)

Non-separable Problems

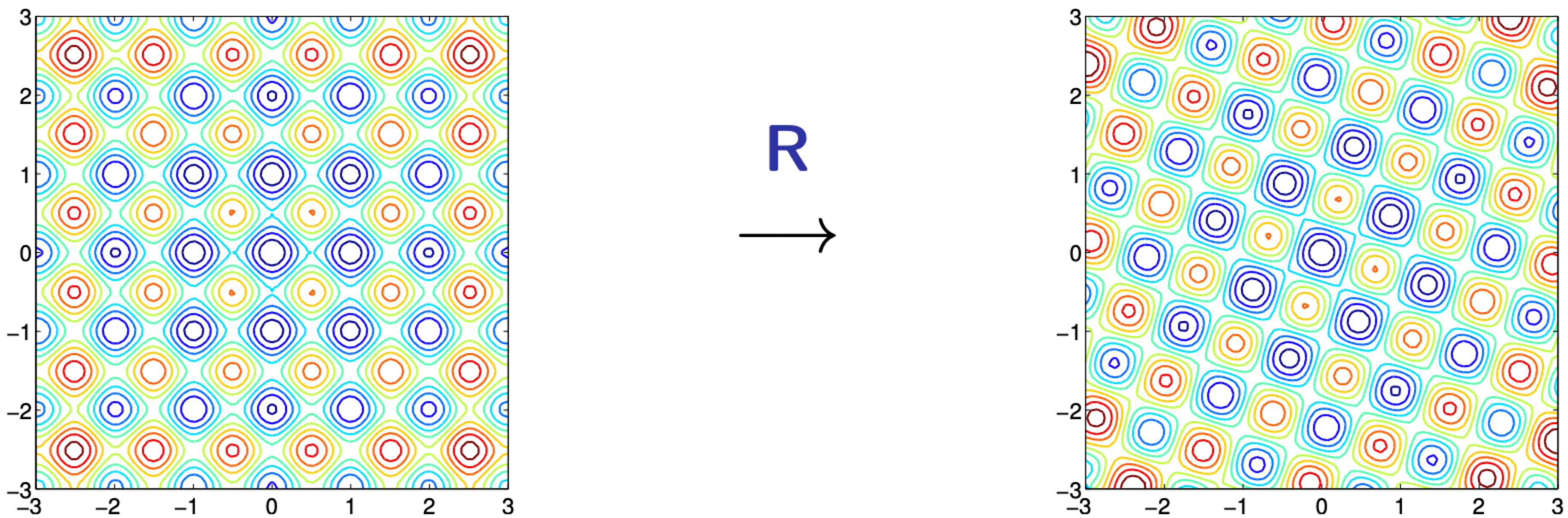
Building a non-separable problem from a separable one

Rotating the coordinate system

- $f : \mathbf{x} \mapsto f(\mathbf{x})$ separable
- $f : \mathbf{x} \mapsto f(\mathrm{R}\mathbf{x})$ non-separable

orthogonal matrix

R rotation matrix



[1] Hansen, Ostermeier, Gawelczyk (1995). On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. Sixth ICGA, pp. 57-64, Morgan Kaufmann

[2] Salomon (1996). "Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms." BioSystems, 39(3):263-278

Ill-conditioned Problems - Case of Convex-quadratic functions

Consider a strictly convex-quadratic function $f(x) = \frac{1}{2}(x - x^\star)^\top H(x - x^\star)$ for $x = (x_1, \ldots, x_n)^\top \in \mathbb{R}^n$ and $x^\star \in \mathbb{R}^n$ with H a symmetric, positive, definite (SPD) matrix.

Remember that $H = \nabla^2 f(x)$.

The condition number of the matrix H (with respect to the Euclidean norm) is defined as

$$\text{cond}(H) = \frac{\lambda_{\text{max}}(H)}{\lambda_{\text{min}}(H)}$$

with $\lambda_{\text{max}}()$ and $\lambda_{\text{min}}()$ being respectively the largest and smallest eigenvalues.

Ill-conditioned means a high condition number of the Hessian matrix H.

Consider now the specific case of the function $f(x) = \frac{1}{2}(x_1^2 + 9x_2^2)$

$H = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$

1. Compute its Hessian matrix, its condition number

2. Plots the level sets of f, relate the condition number to the axis ratio of the level sets of f

cond$(H) = 9$ axis-ratio $= 3 = \sqrt{\text{cond}(H)}$

3. Generalize to a general convex-quadratic function

Real-world problems are often ill-conditioned.

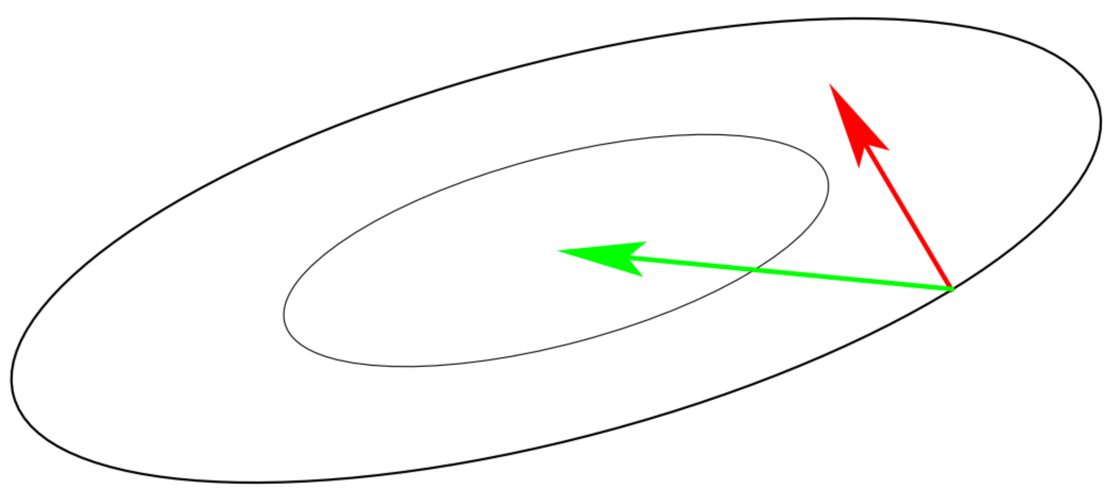
4. Why do you think it is the case? → physical variables optimized can live on different scales.

5. why are ill-conditioned problems difficult?

Ill-conditioned Problems

consider the curvature of the level sets of a function

ill-conditioned means "squeezed" lines of equal function value (high curvatures)



gradient direction $-f'(\boldsymbol{x})^{\mathrm{T}}$

Newton direction $-\boldsymbol{H}^{-1}f'(\boldsymbol{x})^{\mathrm{T}}$

Ill-conditionned : cond $\sim 10^4/10^6$

Condition number equals nine here. Condition numbers up to 10^{10} are not unusual in real world problems.

Part II: Algorithms

A simple randomized / stochastic Derivative Free Optimization algorithm:

PURE RANDOM SEARCH (PRS)

[Objective: minimize $f : [-1,1]^n \to \mathbb{R}$
X_t is the estimate of the optimum at iteration t
`Input` $(U_t)_{t\geq 1}$ independent identically distributed each $U_t \sim \mathcal{U}_{[-1,1]^n}$ (unif. distributed in $[-1,1]^n$)]

```
1. Initialize t = 1, X_1 = U_1
2. while not terminate
3.        t = t + 1
4.        If f(U_t) ≤ f(X_{t-1})
5.              X_t = U_t
6.        Else
8.              X_t = X_{t-1}
```

1. Show that for all $t \geq 1$

$$f(X_t) = \min\{f(U_1), \ldots, f(U_t)\}$$

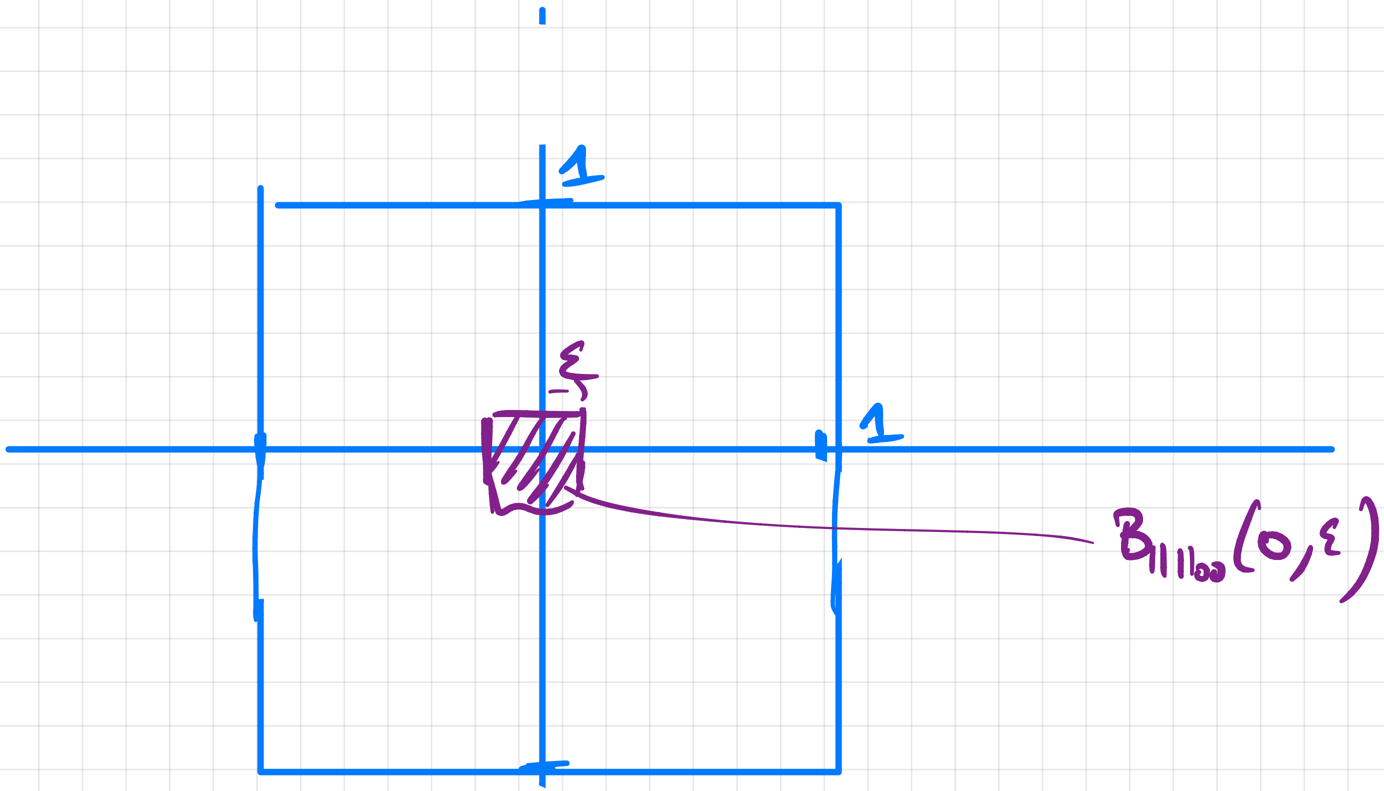
BY INDUCTION

2. We consider the simple case where $f(x) = \|x\|_\infty$ (we remind that $\|x\|_\infty := \max(|x_1|, \ldots, |x_n|)$). Show the convergence in probability of the PRS algorithm towards the optimum of f, that is prove that for all $\epsilon > 0$

$$\lim_{t\to\infty} \Pr\left(\|X_t\|_\infty \geq \epsilon\right) = 0$$

Hint: Prove and use the equality

$$\{\|X_t\|_\infty \geq \epsilon\} = \cap_{k=1}^{t}\{\|U_k\|_\infty \geq \epsilon\}$$



We want to prove that $\Pr(\|x_t\| \geq \varepsilon) \xrightarrow[t\to+\infty]{} 0$

$$\{\|X_t\|_\infty \geq \varepsilon\} \underset{(1)}{=} \{\min\{\|U_1\|_\infty, \dots, \|U_t\|_\infty\} \geq \varepsilon\}$$

$$= \bigcap_{k=1}^{t} \{\|U_k\|_\infty \geq \varepsilon\}$$

$$\Pr(\{\|X_t\|_\infty \geq \varepsilon\}) = \Pr\left(\bigcap_{k=1}^{t} \{\|U_k\|_\infty \geq \varepsilon\}\right)$$

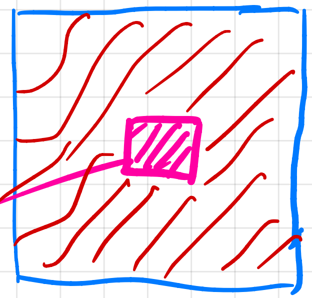
$$\underset{\text{By ind}}{=} \prod_{k=1}^{t} \Pr(\|U_k\|_\infty \geq \varepsilon)$$

$$\underset{\text{Since the RV are identically dis}}{=} \Pr(\|U_1\|_\infty \geq \varepsilon)^t$$

$$\Pr(\|U_1\|_\infty \geq \varepsilon) = 1 - \Pr(\|U_1\|_\infty \leq \varepsilon)$$

$= \Pr(\text{to be in red set})$

$$= 1 - \frac{\mathrm{Vol}(\{x \mid \|x\|_\infty \leq \varepsilon\})}{\mathrm{Vol}(\{x \mid \|x\|_\infty \leq 1\})}$$



$$= 1 - \frac{(2\varepsilon)^n}{2^n}$$

$$= 1 - \varepsilon^n$$

$$\mathbb{P}r\left(\{ \|X_t\|_\infty \geq \varepsilon \}\right) = (1 - \varepsilon^n)^t \xrightarrow[t \to \infty]{} 0$$

Hence the PRS converges in probability to the the minimizer of $f(x) = \|x\|_\infty$.

Let's quantify how fast it converges.

↳ for this we look at hitting time of the optimum.

$T_\varepsilon = \inf \{ t , \|X_t\|_\infty \leq \varepsilon \}$ Hitting time of a ball of radius ε around the optimum.

T_ε is a random variable

We can try to estimate the expected hitting time

$$\mathbb{E}(T_\varepsilon).$$

Let us assume we play a random game with 2 outcomes:

win with probability p

lose ——— $1-p$

(Game Head/Tail → proba to win $\frac{1}{2}$)

and the outcome is sampled randomly and independently.

We play till we win.

T is how many times I need to play to see the first win.

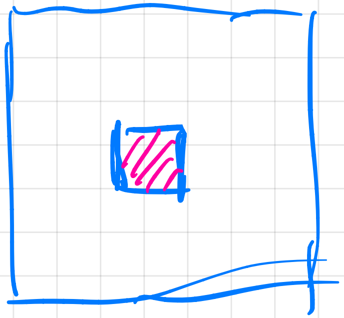
→ T follows a geometric distribution with parameter p

On "average" $\mathbb{E}(T) = \frac{1}{p}$

We can make a parallel between PRS and a game:

win = if I reach $B(0, \varepsilon)$

lose otherwise.



T_ε = Hitting time of $B(0, \varepsilon)$

= time it takes to win in the game.

$T_\varepsilon \sim$ geometric distribution with parameter

$$p = \mathbb{P}r\left(\|U_1\|_\infty \leq \varepsilon\right) = \varepsilon^n$$

$$\mathbb{E}(T_\varepsilon) = \frac{1}{\varepsilon^n}$$

This is slow, as a comparison, if we get linear (geometric) convergence then

$$\mathbb{E}(T_\varepsilon) \simeq \log(1/\varepsilon)$$

<u>OBJECTIVE:</u> Explain how to do better than PRS.