Derivative Free Optimization

joint course between
Optimization Master Paris Saclay - AMS Master
2024/2025

Anne Auger - Dimo Brockhoff
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

29/11/2024	room 1314
06/12/2024	room 1314
13/12/2024	room 1314
20/12/2024	room 1213
10/01/2025	room 1213
17/01/2025	room 1314
24/01/2025	room 1314
31/01/2025	room 1314
07/02/2025	room 1314
14/02/2025 [EXAM]	TBA

Evaluation

Written exam on 14/02/2025) 60%

Project (in group) around benchmarking of algorithms

- oral presentation to the class

) 40%

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 [taught D. Brockhoff]
- Benchmarking (partly taught by D. Brockhoff)

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

Derivative-Free / Black-box Optimization

Task: minimize a numerical **objective** function (also called *fitness* function or *loss* function)

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

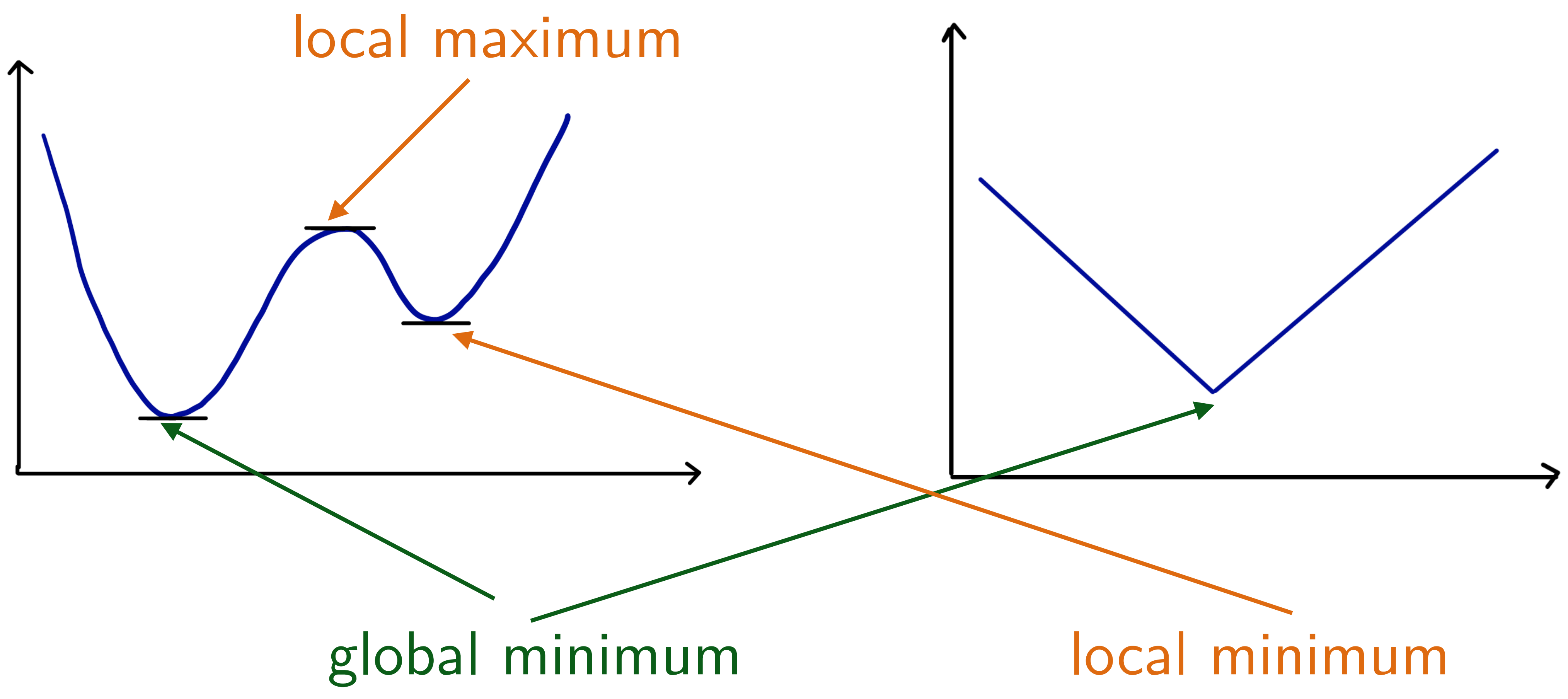
without derivatives (gradient). Ω: search space, n :dimension of the search space

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

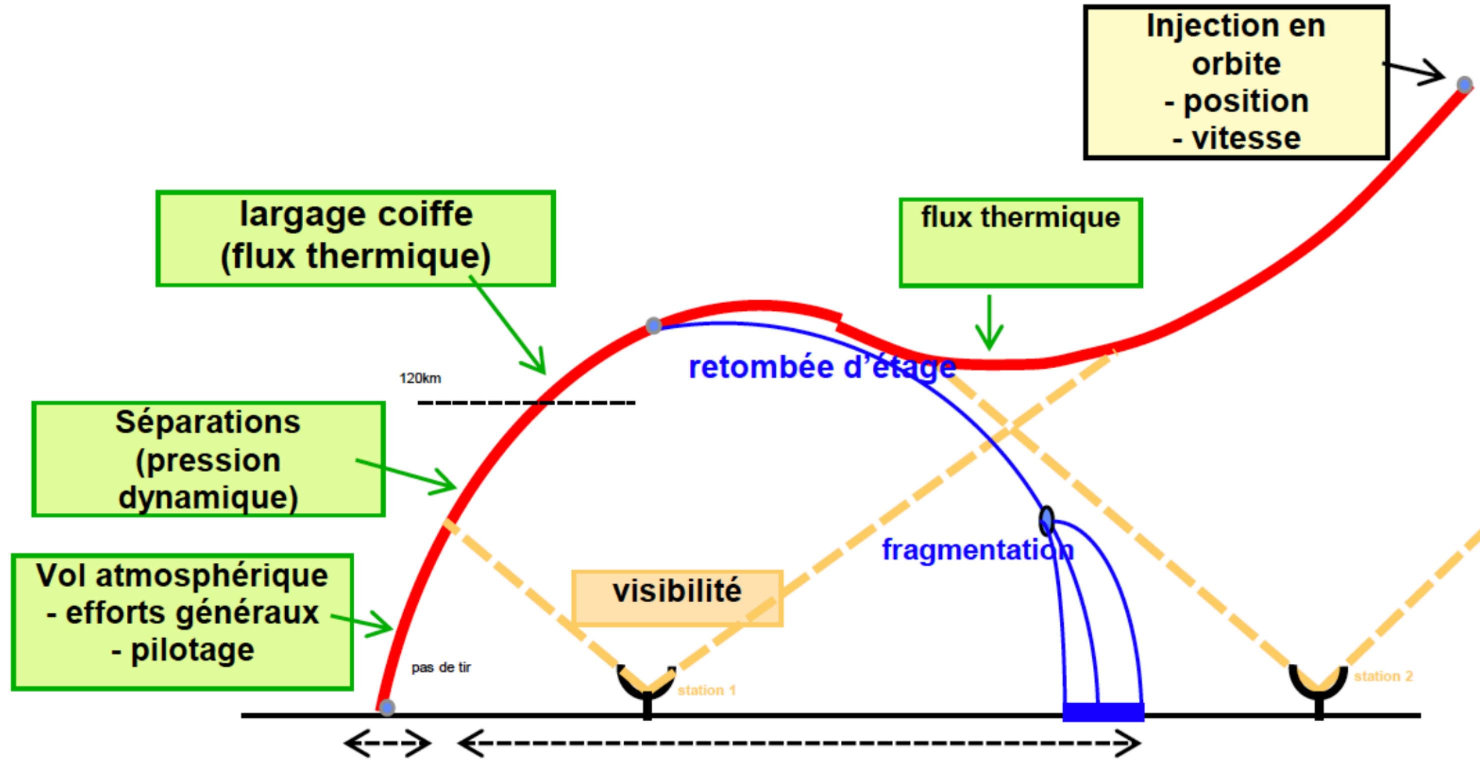
Reminder: Local versus Global Optimum

n=1



Examples: Optimization of the Design of a Launcher

Poppy



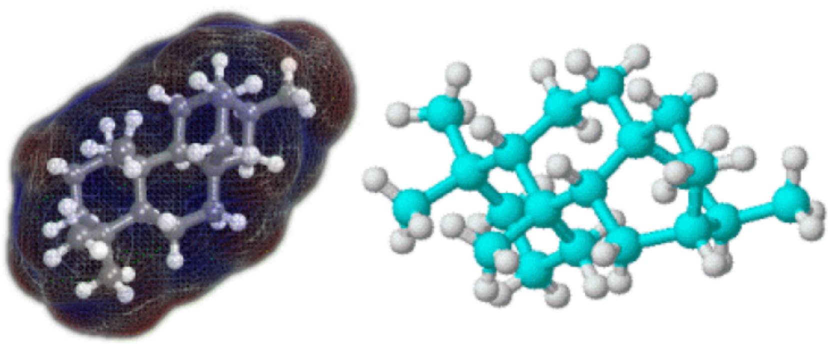
- Scenario: multi-stage launcher brings a satellite into orbit
- Minimize the overall cost of a launch
- Parameters: propellant mass of each stage / diameter of each stage / flux of each engine / parameters of the command law

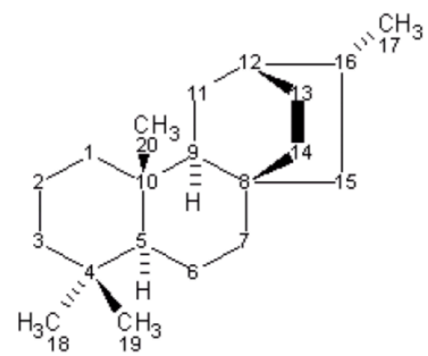
23 continuous parameters to optimize

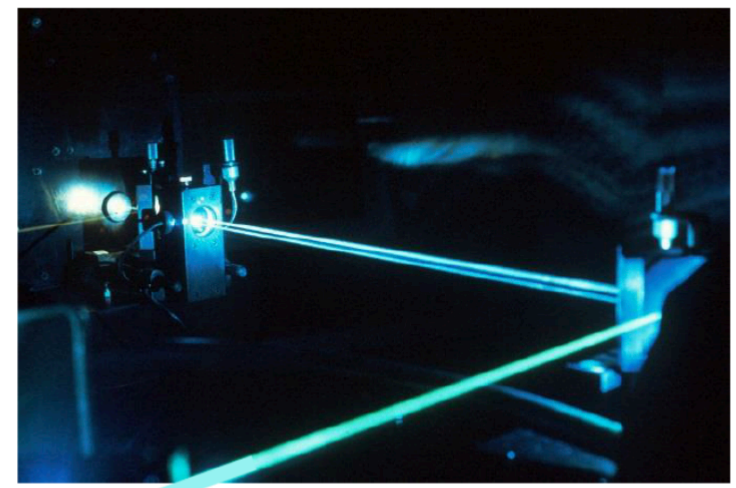
+ constraints

Control of the Alignement of Molecules

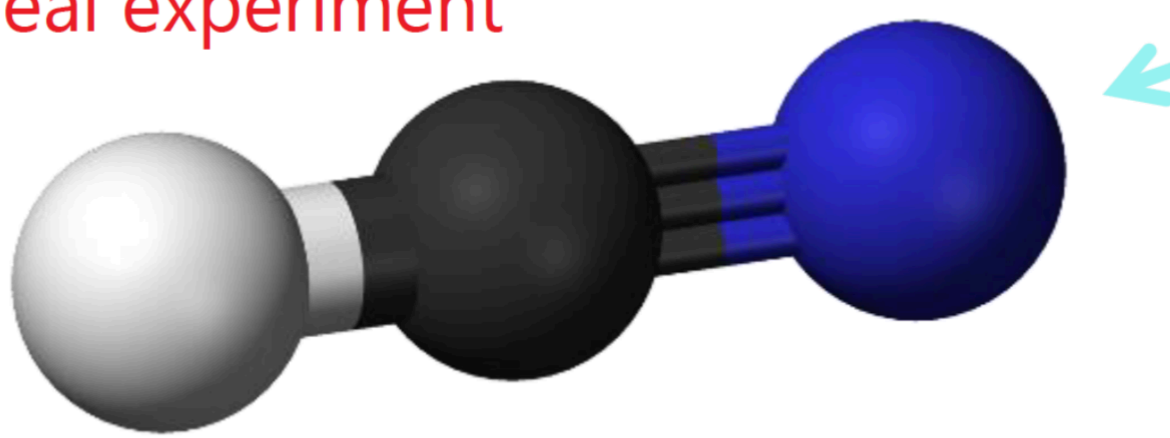
application domain: quantum physics or chemistry







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

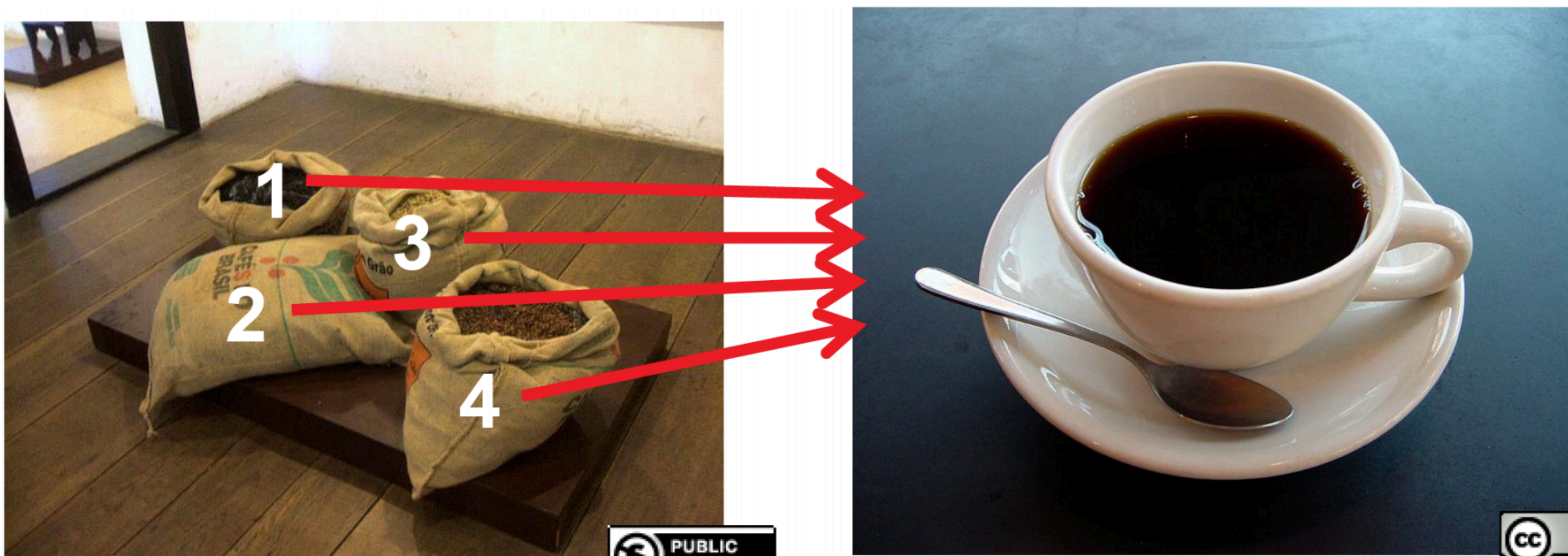
Coffee Tasting Problem

$x_i \geq 0$
$\sum x_i = 1$

- Find a mixture of coffee in order to keep the coffee taste from one year to another

$(x_1, x_2, x_3, x_4) \longrightarrow$ Taste

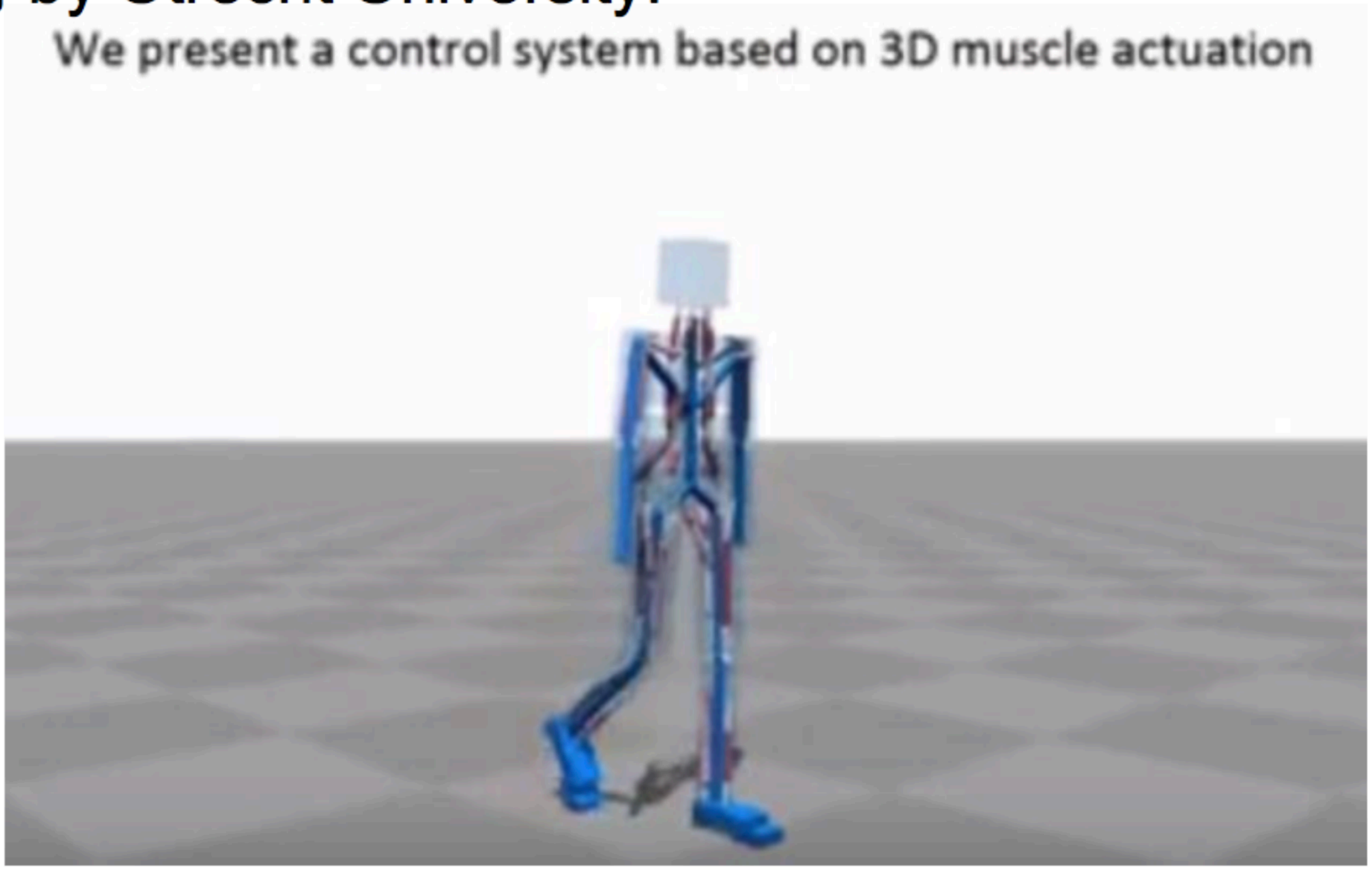
- Objective function = opinion of one expert



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

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

- 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

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

- 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

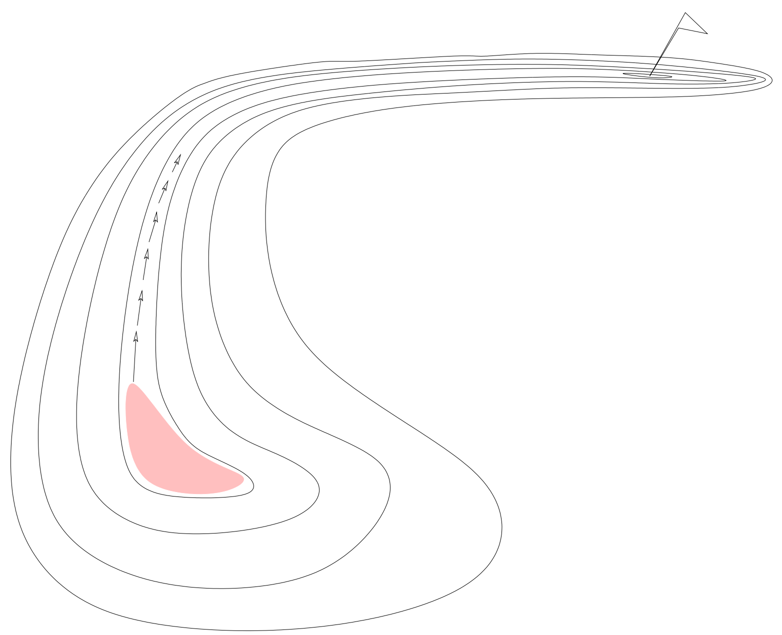
Level Sets of a Function

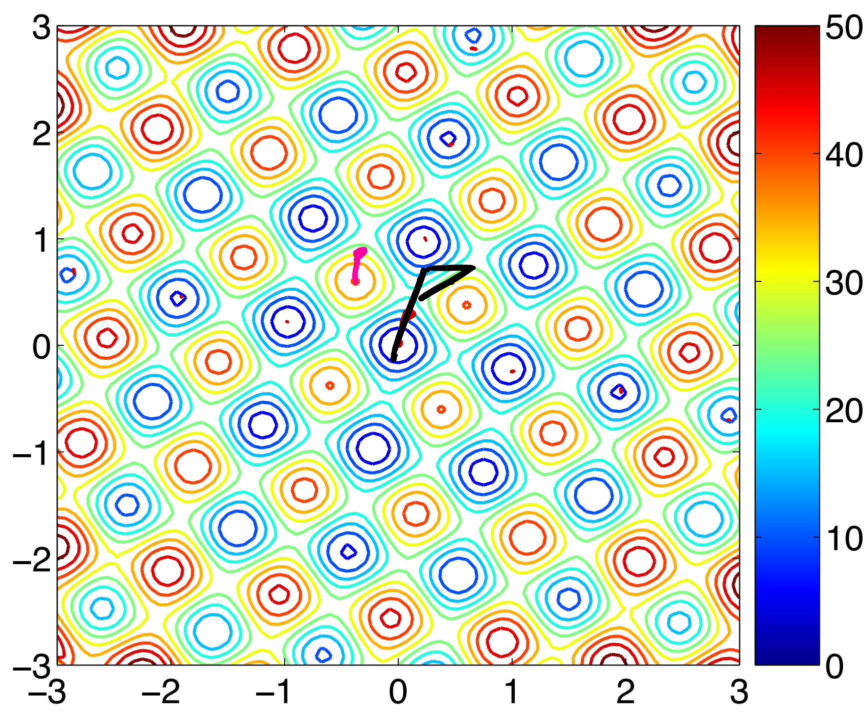
Level Sets: Visualization of a Function

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

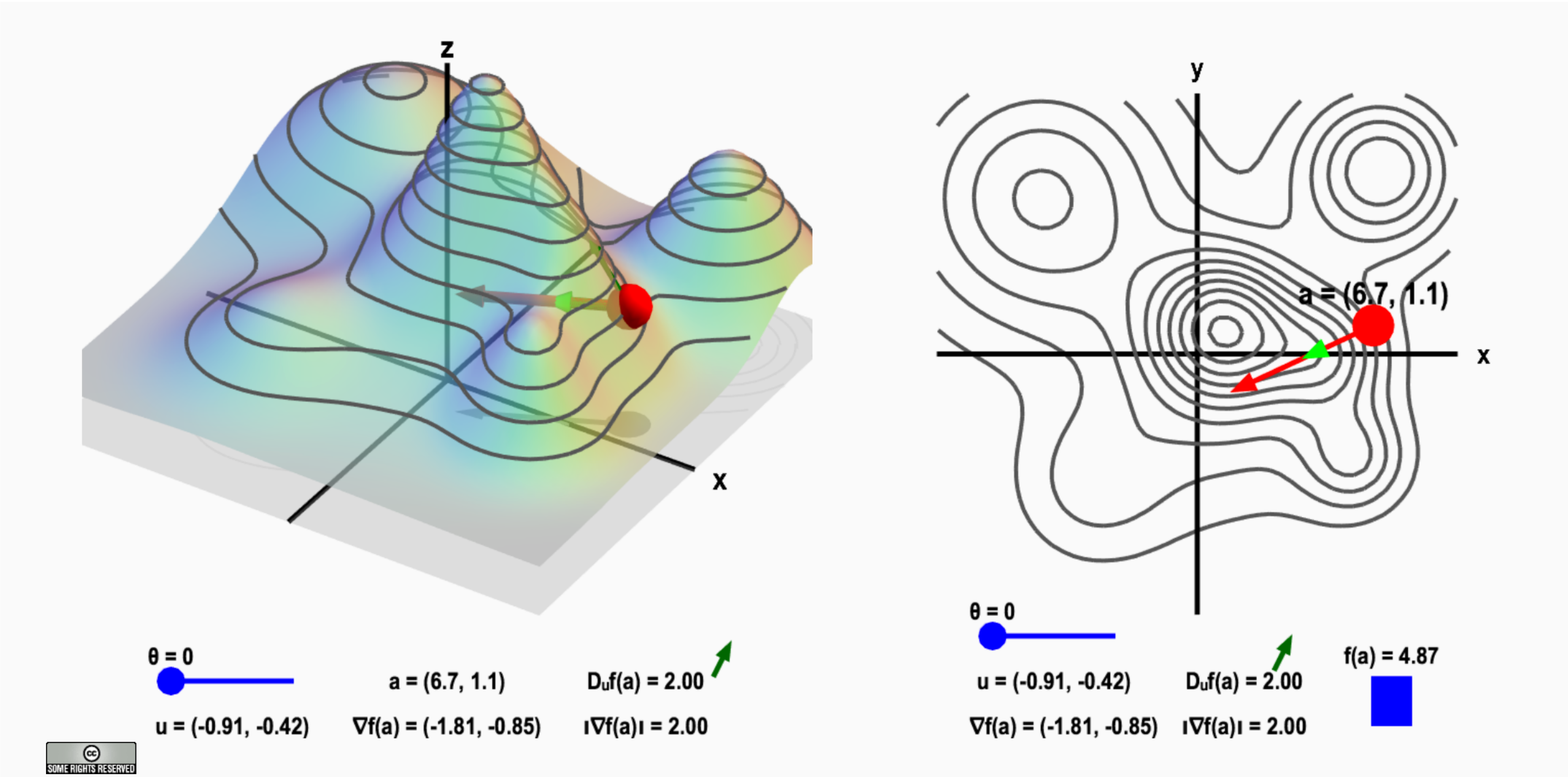
$$\mathcal{L}_c = \{x \in \mathbb{R}^n \,|\, f(x) = c\}, c \in \mathbb{R}$$

Examples of level sets in 2D





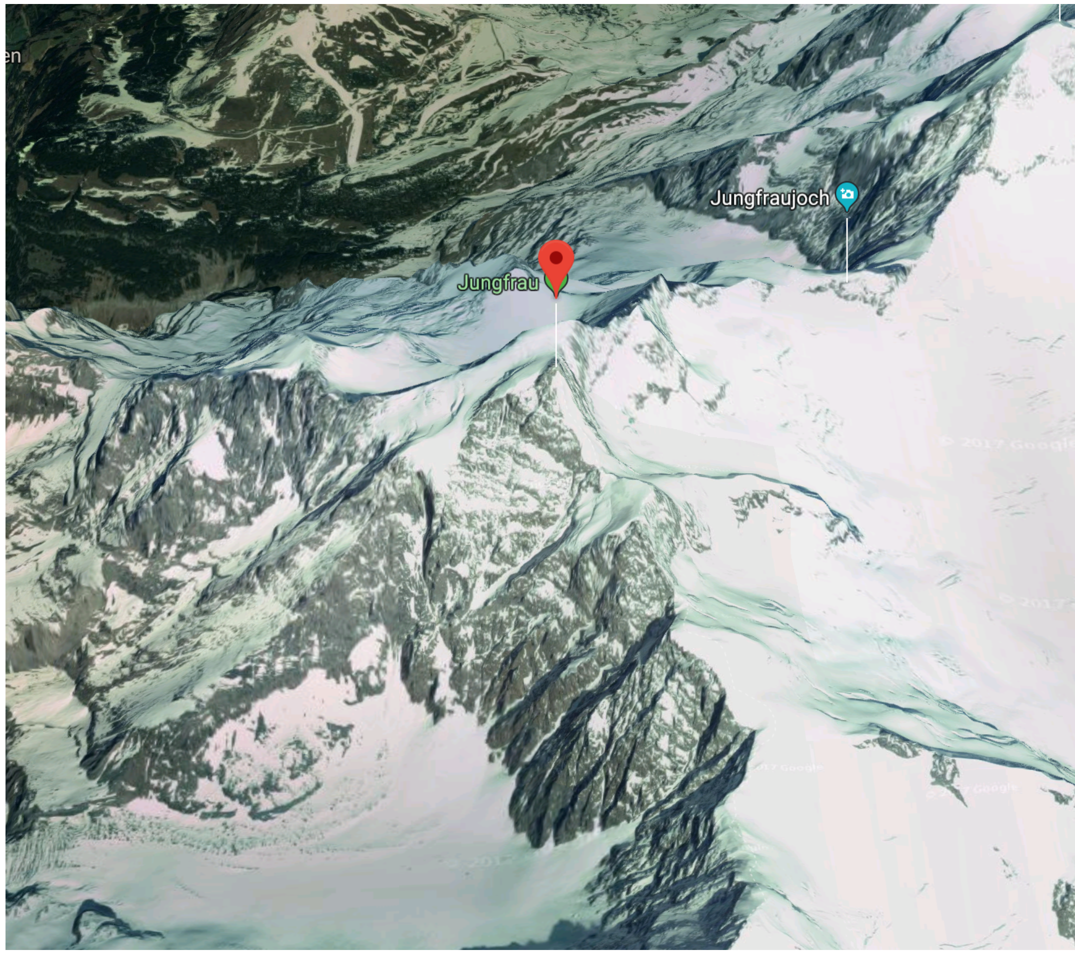
Level Sets: Visualization of a Function



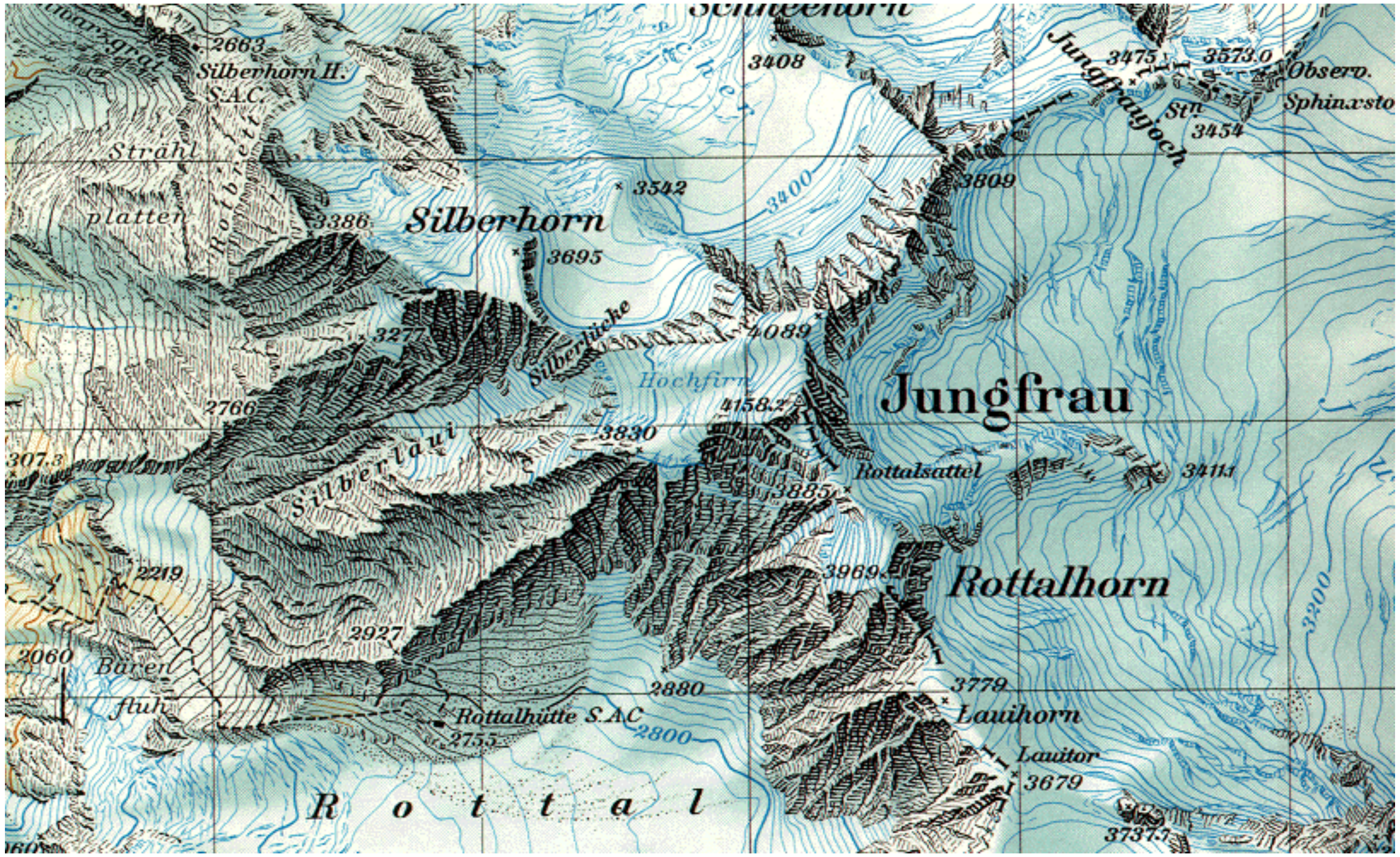
Source: Nykamp DQ, "Directional derivative on a mountain." From *Math Insight*. http://mathinsight.org/applet/directional_derivative_mountain

Level Sets: Topographic Map

The function is the altitude



3-D picture



Topographic map

Level Set: Exercice

Consider a strictly convex-quadratic function

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

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

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

strictly convex.

$f(x) \geq 0$ because $H \succ 0$

$f(x) = 0 \iff x - x^* = 0 \iff x = x^*$

$H = \nabla^2 f(x)$ Hessian matrix

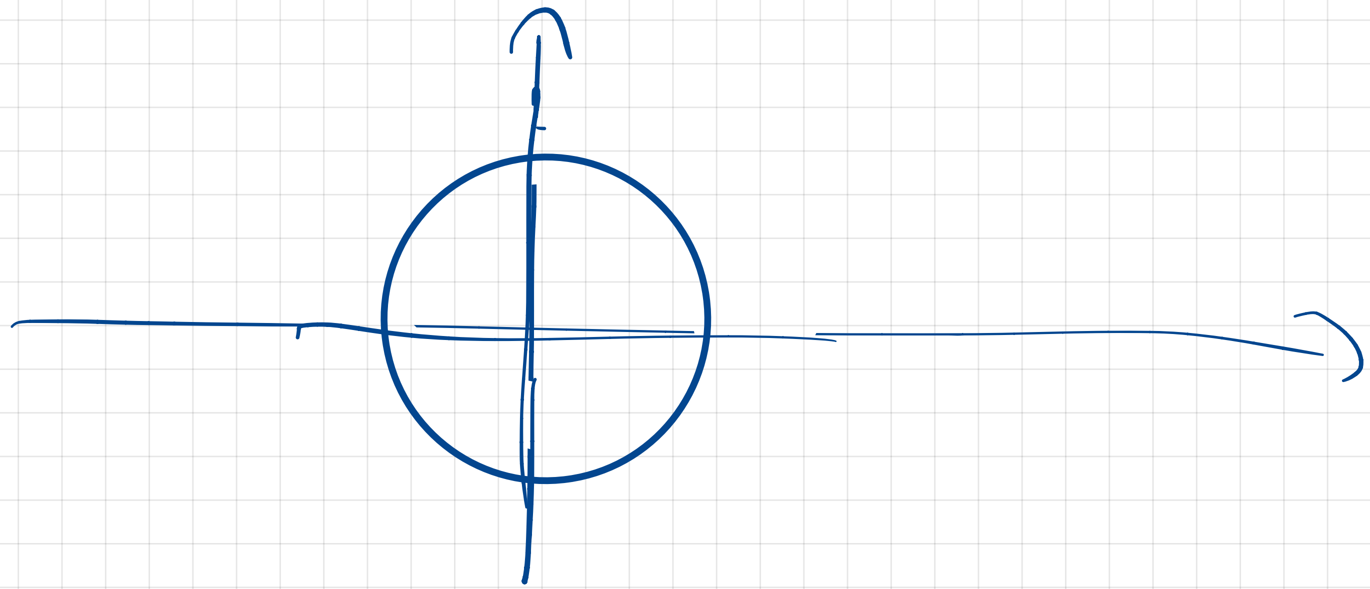
WLG $x^* = 0$

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

$$f(x) = \frac{1}{2}(x_1^2 + x_2^2)$$

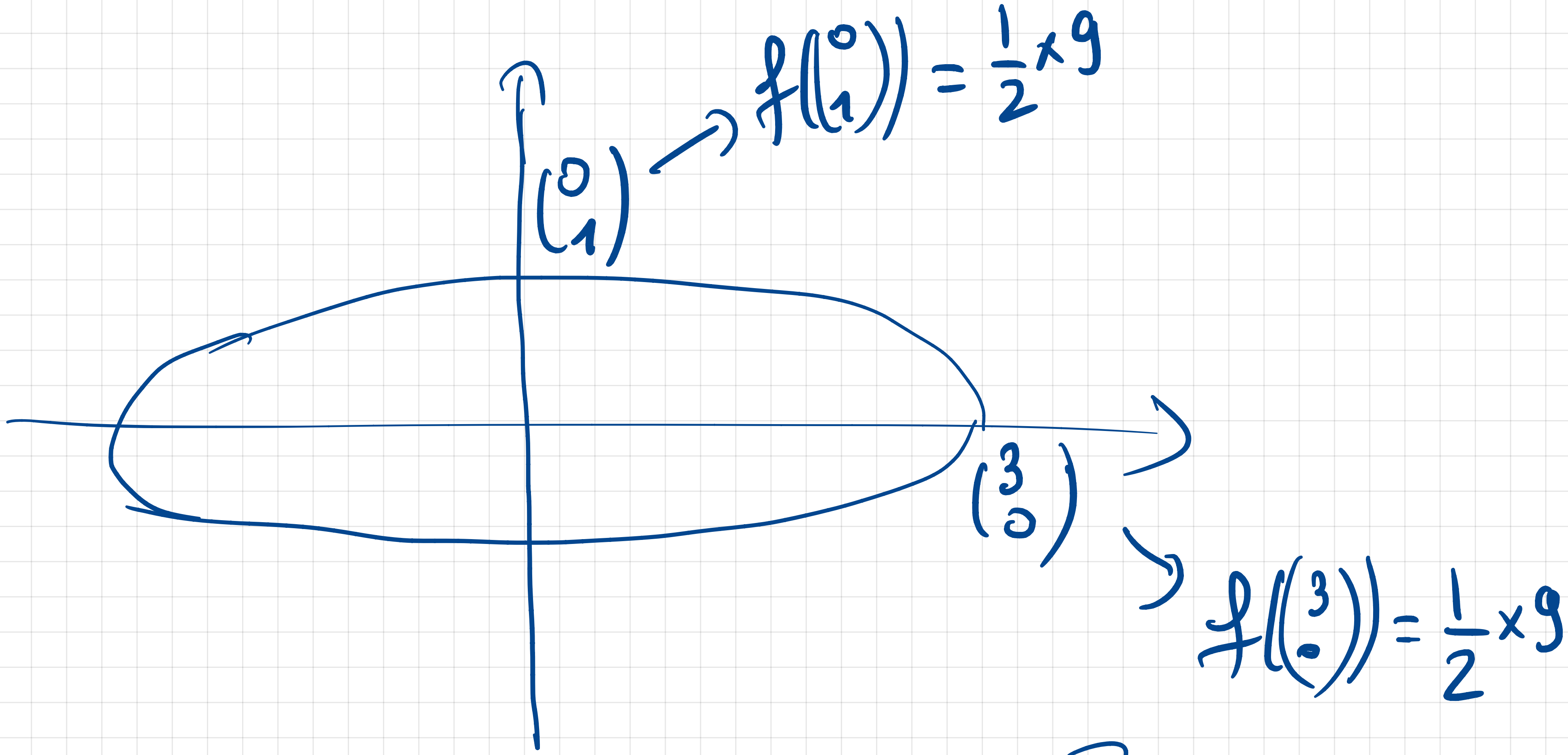
$$L_c = \{ x \mid \frac{1}{2}(x_1^2 + x_2^2) = c \}$$

$c > 0$



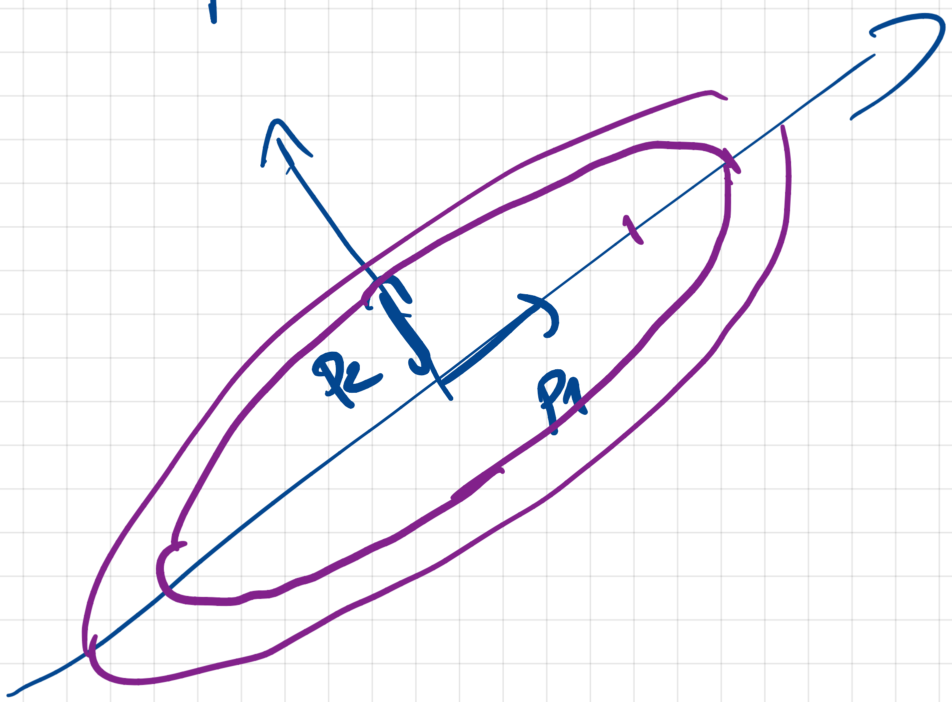
$H = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$ $\quad L_c = \{ x = (x_1, x_2) \mid \frac{1}{2}(x_1^2 + 9x_2^2) = c \}$

$c > 0$, ellipsoid.



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

$P = (p_1, p_2)$ eigenvectors of H

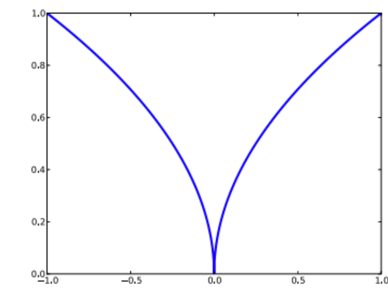


What Makes an Optimization Problem Difficult?

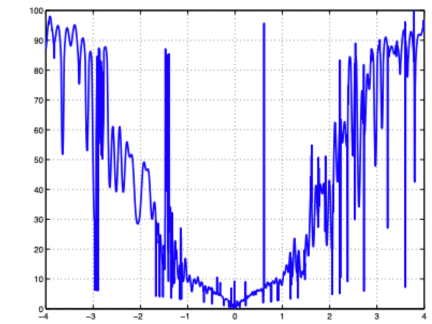
What Makes a Function Difficult to Solve?

Why stochastic search?

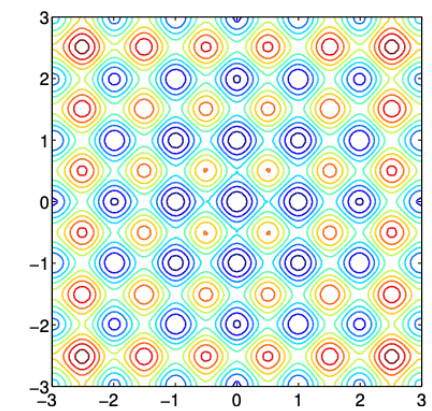
- non-linear, non-quadratic, non-convex
  on linear and quadratic functions much better search policies are available



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

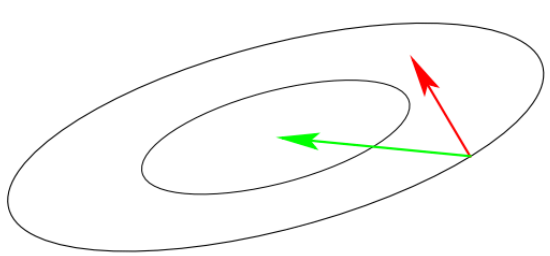


- dimensionality (size of search space)
  (considerably) larger than three



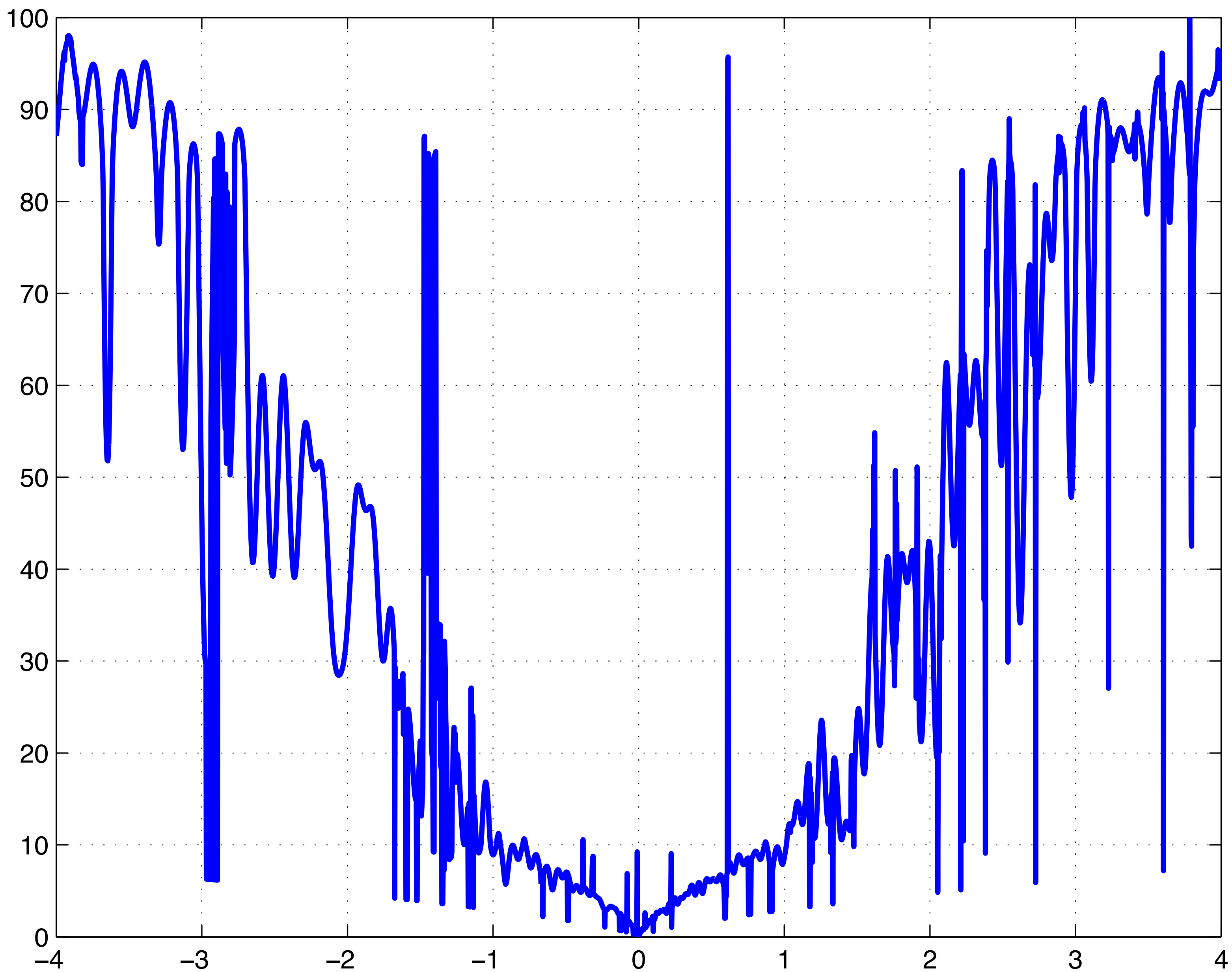
- non-separability
  dependencies between the objective variables

- ill-conditioning



gradient direction Newton directio

Ruggedness



$f: x \in \mathbb{R}^4 \rightarrow \mathbb{R}$

$d \in \mathbb{R}^4$

$t \in \mathbb{R} \rightarrow f(td)$

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

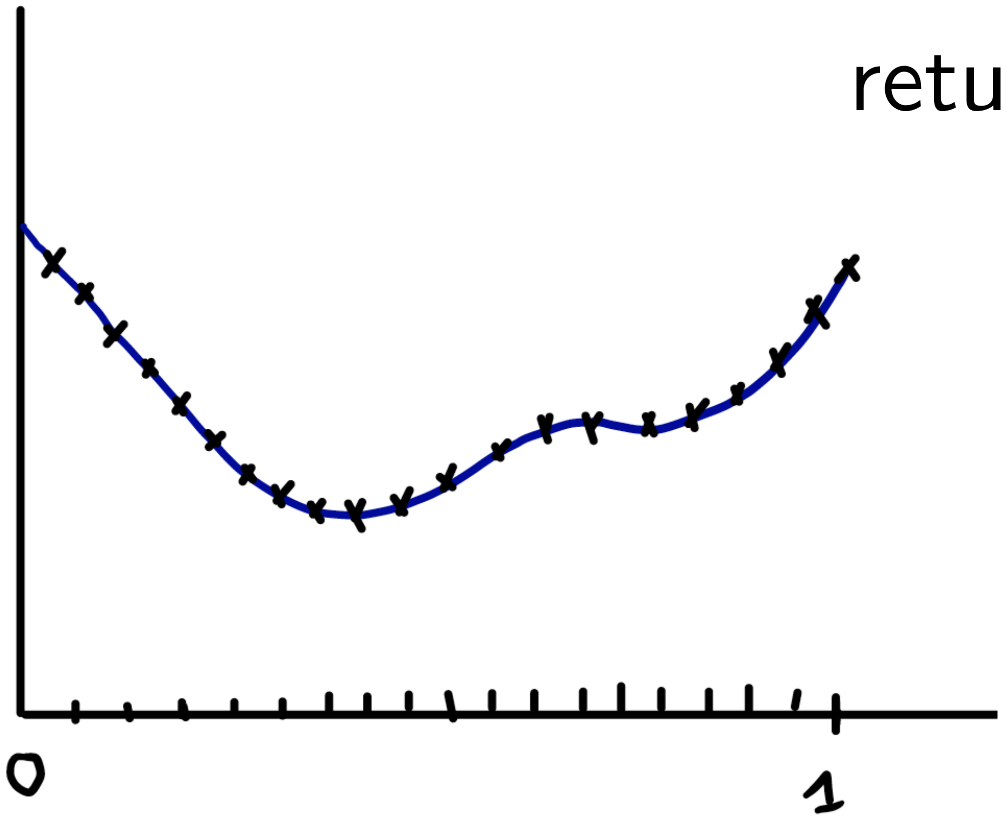
if n=1, which simple approach could you use to minimize:

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

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

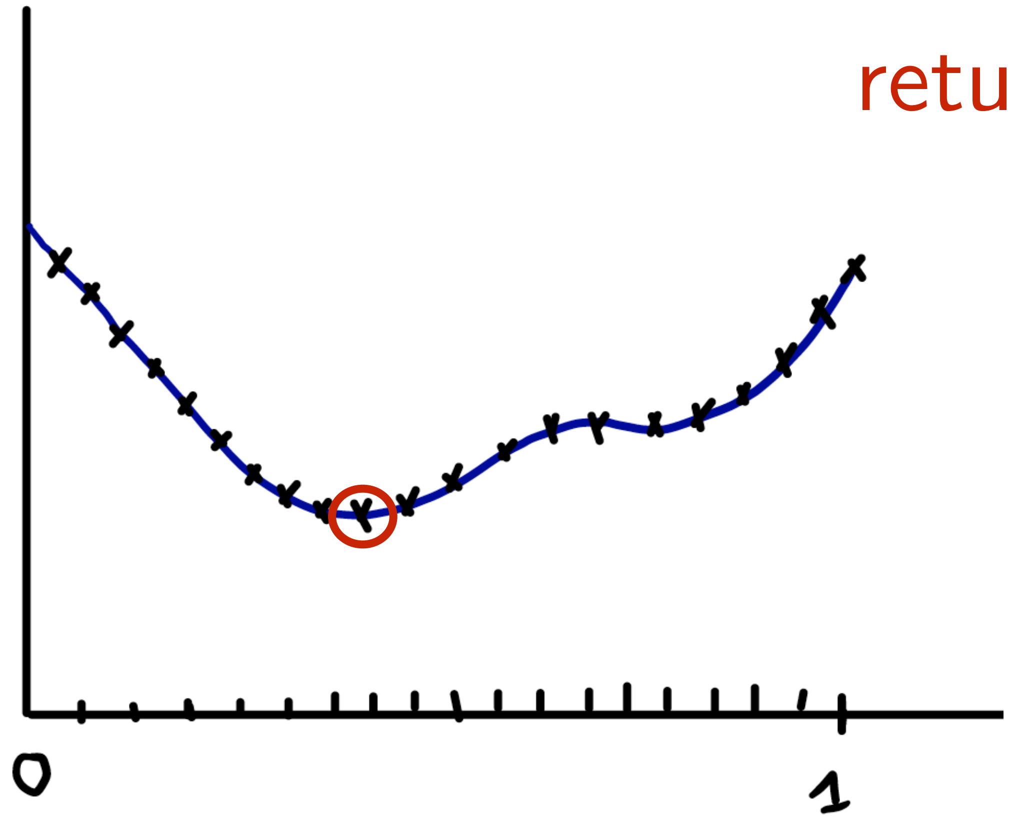
if n=1, which simple approach could you use to minimize:

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

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

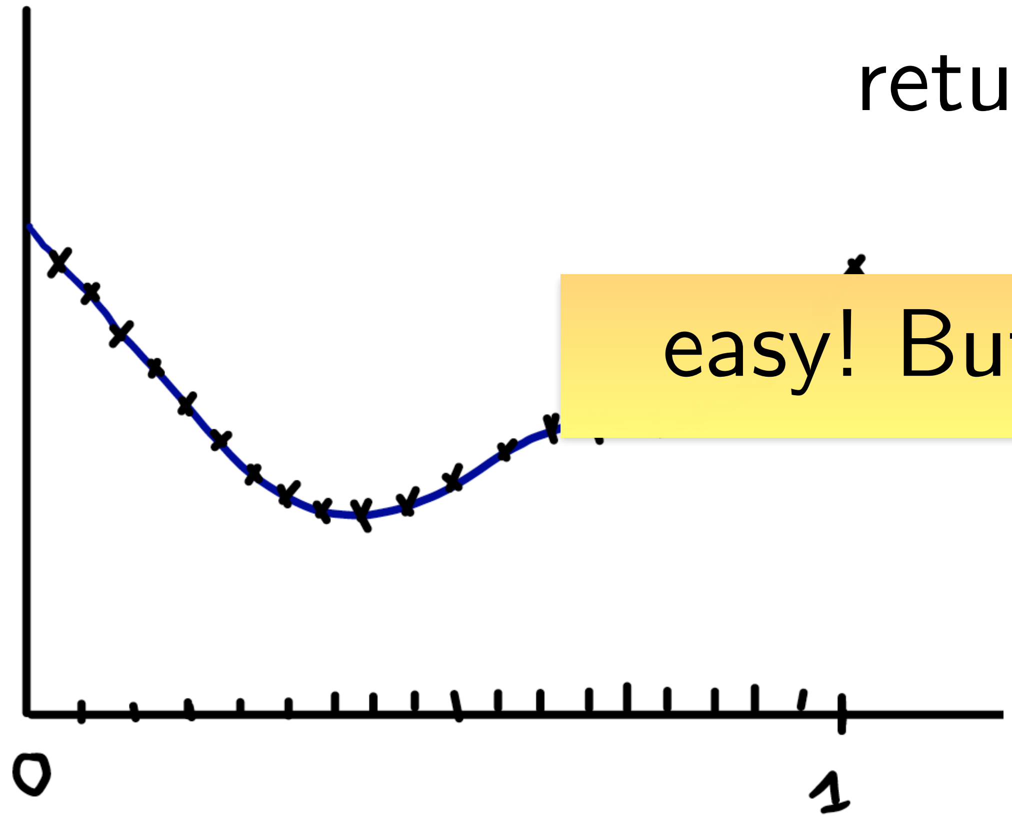
if n=1, which simple approach could you use to minimize:

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

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

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]. To get similar coverage, in terms of distance between adjacent points, of the 10-dimensional space $[0,1]^{10}$ would require $100^{10} = 10^{20}$ points. A 100 points appear now as isolated points in a vast empty space.

grid search

Consequence: a search policy (e.g. exhaustive search) that is valuable in small dimensions might be useless in moderate or large dimensional search spaces.

Curse of Dimensionality

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

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
```

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.

Proof: $\mathbb{R} \ni y \overset{h}{\longmapsto} h(y) \in \mathbb{R}$. Let $g : \mathrm{Im}(h) \to \mathbb{R}$ strict increasing

$$\boxed{\operatorname*{argmin}_y h(y) = \operatorname*{argmin}_y g \circ f(y)}$$

Let $\bar{x} \in \operatorname*{argmin}_y h(y)$

$h(\bar{x}) \leq h(y) \quad \forall y$

Since g strict increasing

$g \circ h(\bar{x}) \leq g \circ h(y) \quad \forall y$

$\Rightarrow \bar{x} \in \operatorname{argmin} g \circ h$

Let $\bar{x} \in \underset{y}{\operatorname{argmin}}\ g \circ h(y)$

$g \circ h(\bar{x}) \leq g \circ h(y) \quad \forall y$

$\overset{g^{-1}}{\Longrightarrow} \quad g^{-1}(g \circ h(\bar{x})) \leq g^{-1}(g \circ h(y)) \quad \forall y$

generalized inverse

$\Rightarrow \quad h(\bar{x}) \leq h(y) \quad \forall y$

$\Rightarrow \quad \bar{x} \in \operatorname{argmin}\ h$

Since the argmin is preserved when composing with g strict increasing to the left, then if f is separable, gof is separable.

Example

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

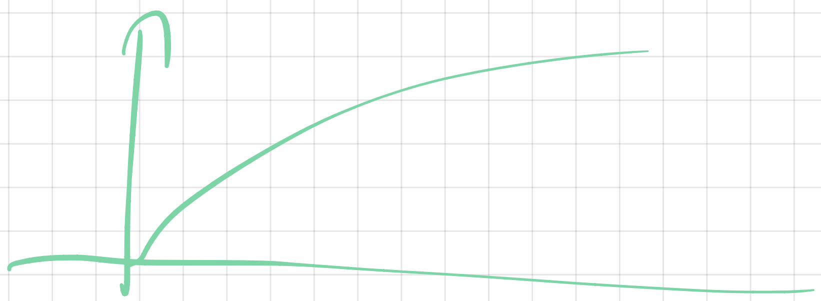
$\tilde{f}(x) = \left(\sum_{i=1}^{m} x_i^2 \right)^{1/4}$

$\parallel$

$g \circ f(x)$

$g(x) = \begin{cases} \operatorname{Im} f \to \mathbb{R} \\ x \to x^{1/4} \end{cases}$

$\operatorname{Im} f = \mathbb{R}_{\geq 0}$



$g \rightarrow$ $\quad \operatorname{argmin} h = \operatorname{argmax} g \circ f$

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

Let us prove that $(\text{argmin}_y f_{x\neg 1}(y), \ldots, \text{argmin}_y f_{x\neg n}(y)) \subset \text{argmin} f$

$$\alpha_i \in \underset{\alpha}{\text{argmin}}\, f(x_1, \ldots, x_{i-1}, \alpha, x_{i+1}, \ldots, x_n) \quad i = 1, \ldots, n$$

$$f(x) = f(x_1, \ldots, x_n) \underset{\substack{\uparrow \\ \text{by def of } \alpha_1}}{\geq} f(\alpha_1, x_2, \ldots, x_n) \underset{\substack{\uparrow \\ \text{by def of } \alpha_2}}{\geq} f(\alpha_1, \alpha_2, x_3, \ldots, x_n)$$

$$f(x) \geq \dots \geq f(\alpha_1, \dots, \alpha_n) \quad \forall x$$

$$(\alpha_1, \dots, \alpha_n) \in \operatorname*{argmin}_{x \in \mathbb{R}^n} f.$$

The other inclusion is immediate:

$$\operatorname*{argmin}_{x} f \subset \left(\operatorname*{argmin}_{x} f(y), \dots, \right)$$

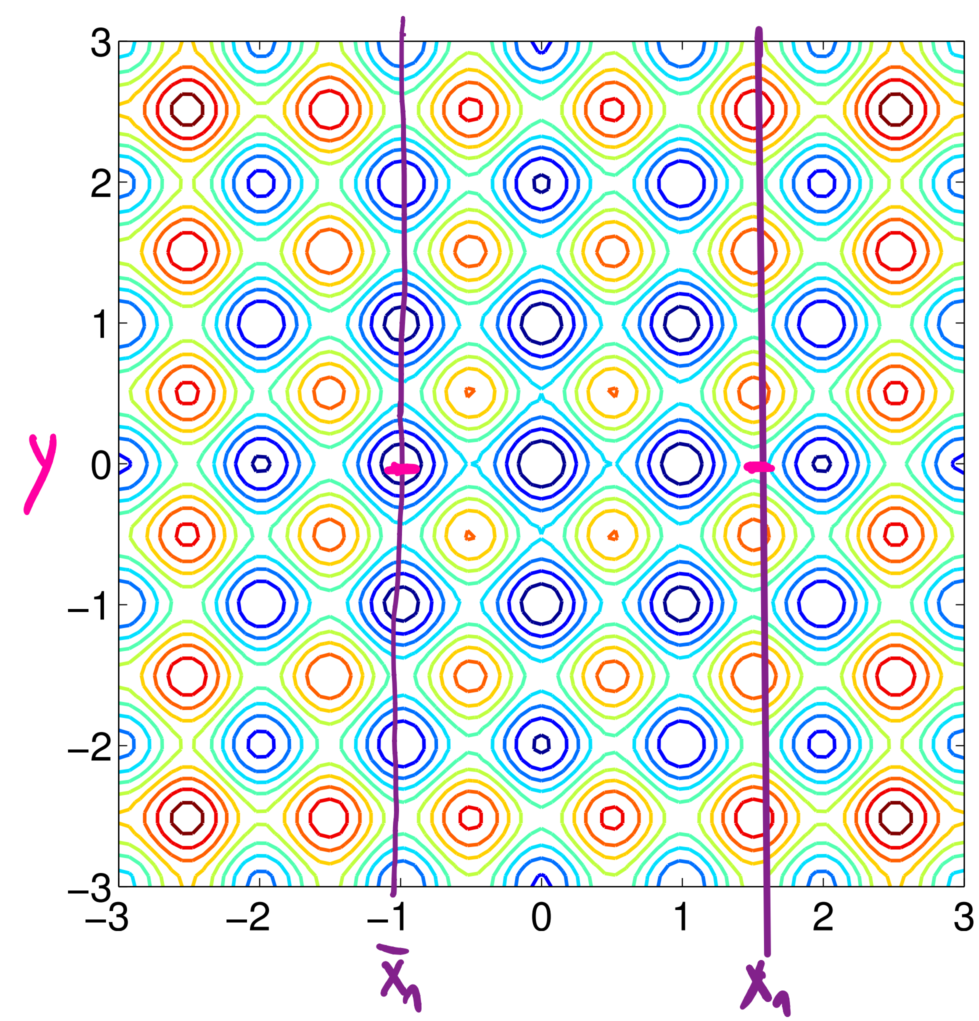
Example: Additively Decomposable Functions

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.

Proof:

$$f(x) = \exp\left(\ln \prod_{i=1}^{n} h_i(x_i)\right)$$

$$= \exp\left(\sum_{i=1}^{n} \ln h_i(x_i)\right)$$

$= g \circ$ additively decomposable

$g(x) = \exp(x)$ strict inc

$\tilde{f}(x) = \sum_{i=1}^{n} \ln h_i(x_i)$: additively decomposable

↳ 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)

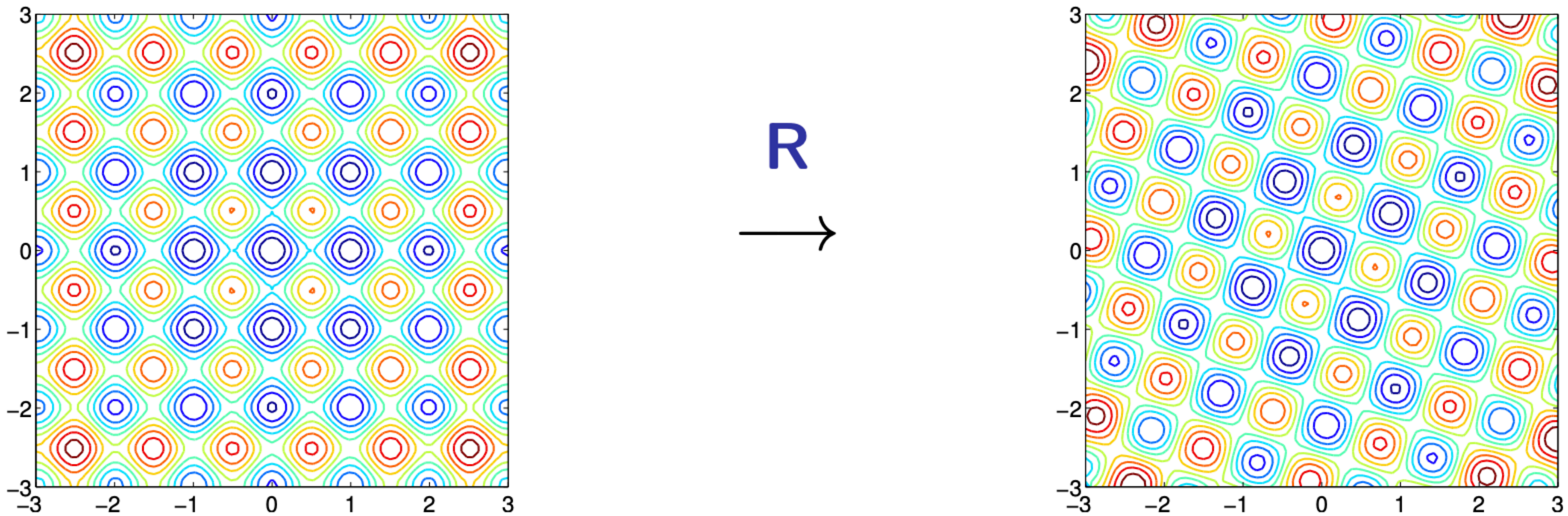
Non-separable Problems

Building a non-separable problem from a separable one

Rotating the coordinate system

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

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

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

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

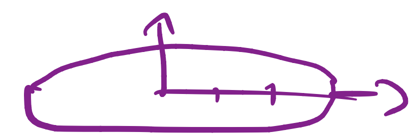
Ill-conditioned means a high condition number of the Hessian matrix H.

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

1. Compute its Hessian matrix, its condition number $H = \begin{pmatrix} 1 & 0 \\ 0 & 9 \end{pmatrix}$

cond(H) = 9

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



3. Generalize to a general convex-quadratic function

Real-world problems are often ill-conditioned.

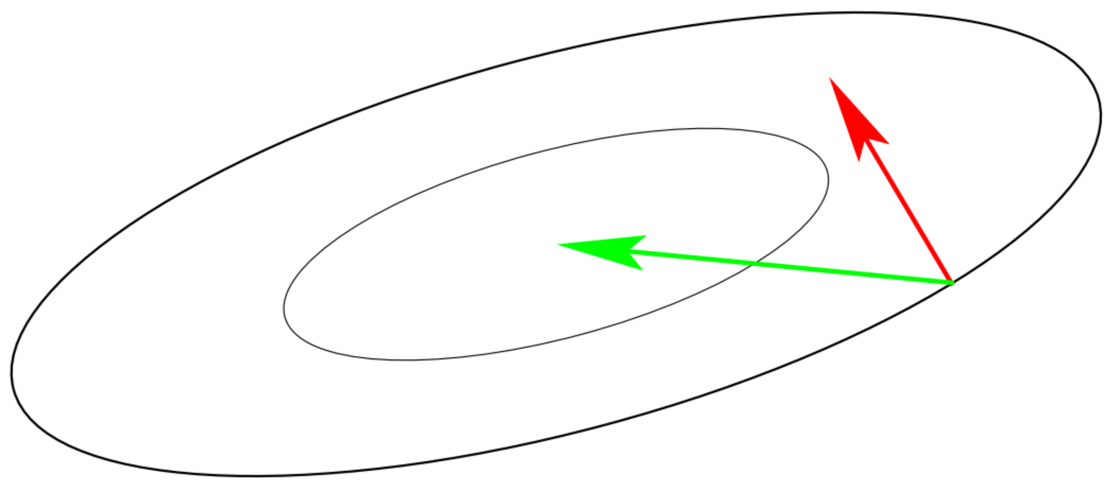
4. Why do you think it is the case? → physical variables optimized can live on different scales.

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

Condition number equals nine here. Condition numbers up to 10^{10} | 10^6
are not unusual in real world problems.