# Derivative-Free / Black-box Optimization

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

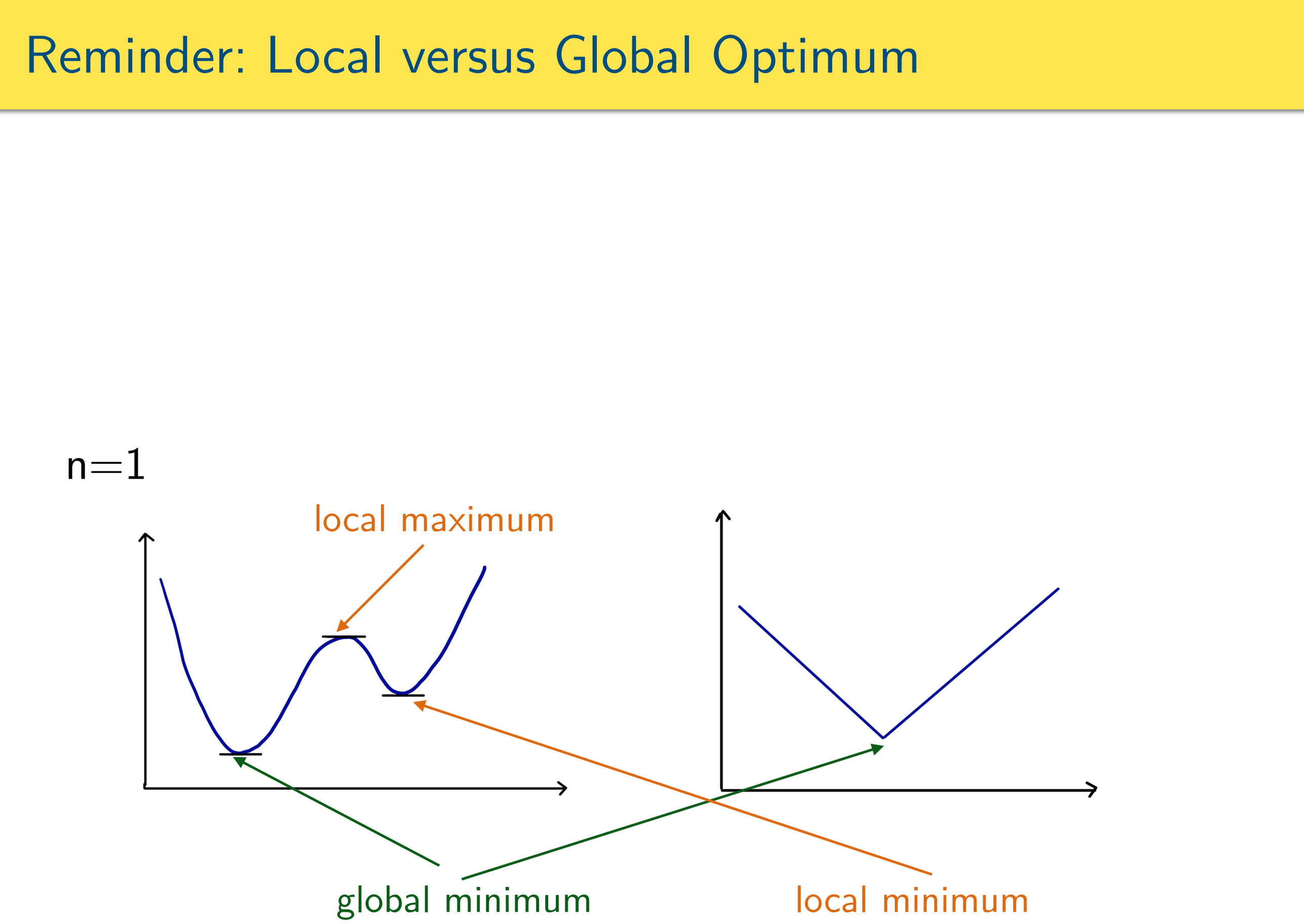
$$f : \Omega \subset \mathbb{R}^n \to \mathbb{R}, x \mapsto f(x) \in \mathbb{R}$$

without derivatives (gradient). $\Omega$: search space, $n$ :dimension of the search space

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

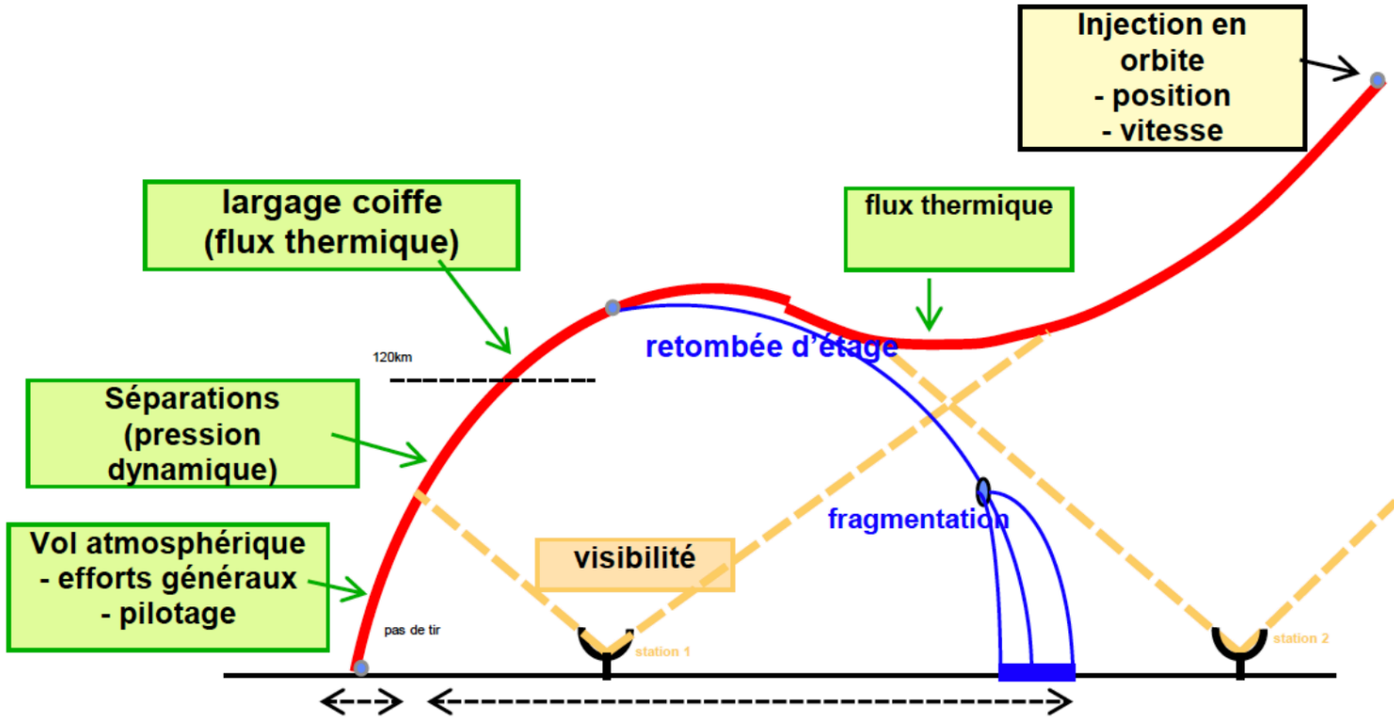

The function is seen by the algorithm as a zero-order **oracle** [a first order oracle would also return gradients] that can be queried at points and the oracle returns an answer

# Reminder: Local versus Global Optimum



# Examples: Optimization of the Design of a Launcher


Poppy




copyright by Astrium

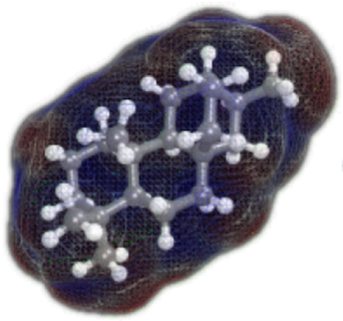
- Scenario: multi-stage launcher brings a satellite into orbit
- Minimize the overall cost of a launch
- Parameters: propellant mass of each stage / diameter of each stage / flux of each engine / parameters of the command law

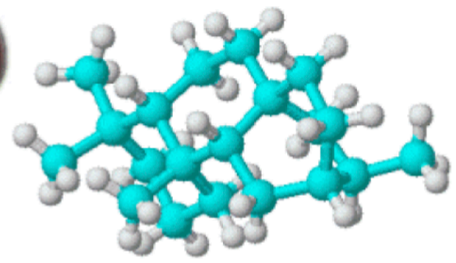
*23 continuous parameters to optimize*

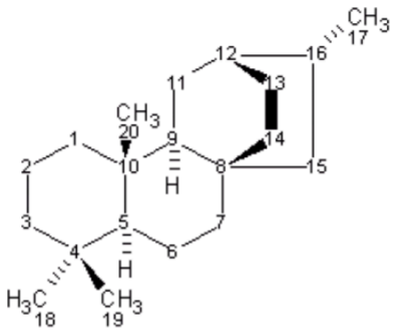
*+ constraints*

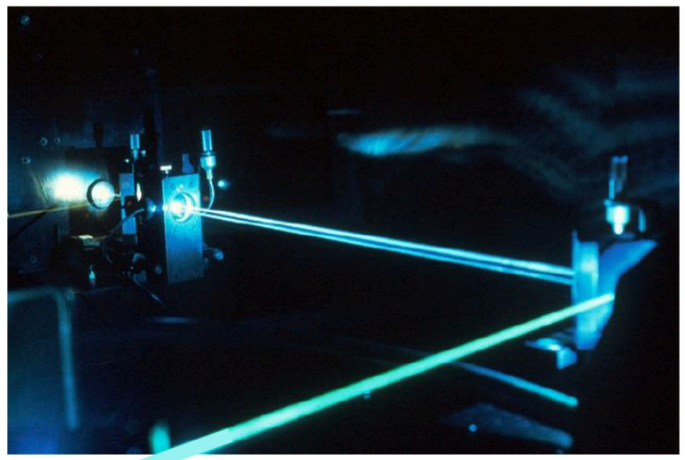
# Control of the Alignement of Molecules

*application domain: quantum physics or chemistry*

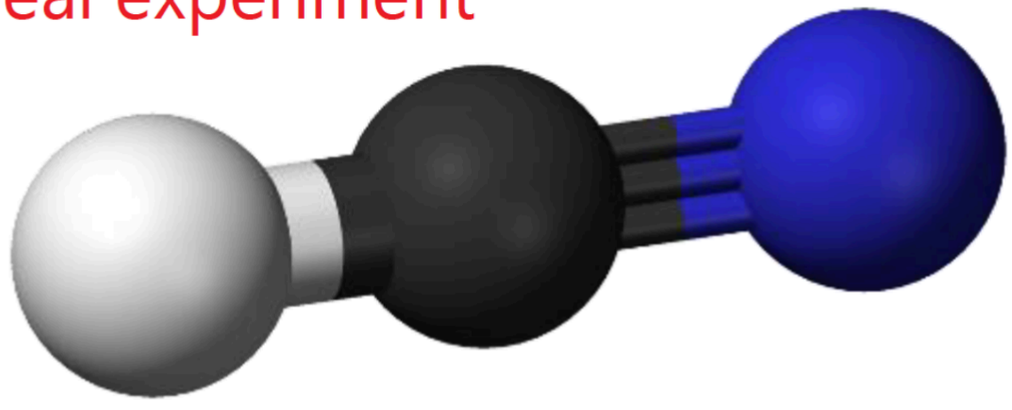






**Objective function:**
via numerical simulation
or a real experiment



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)

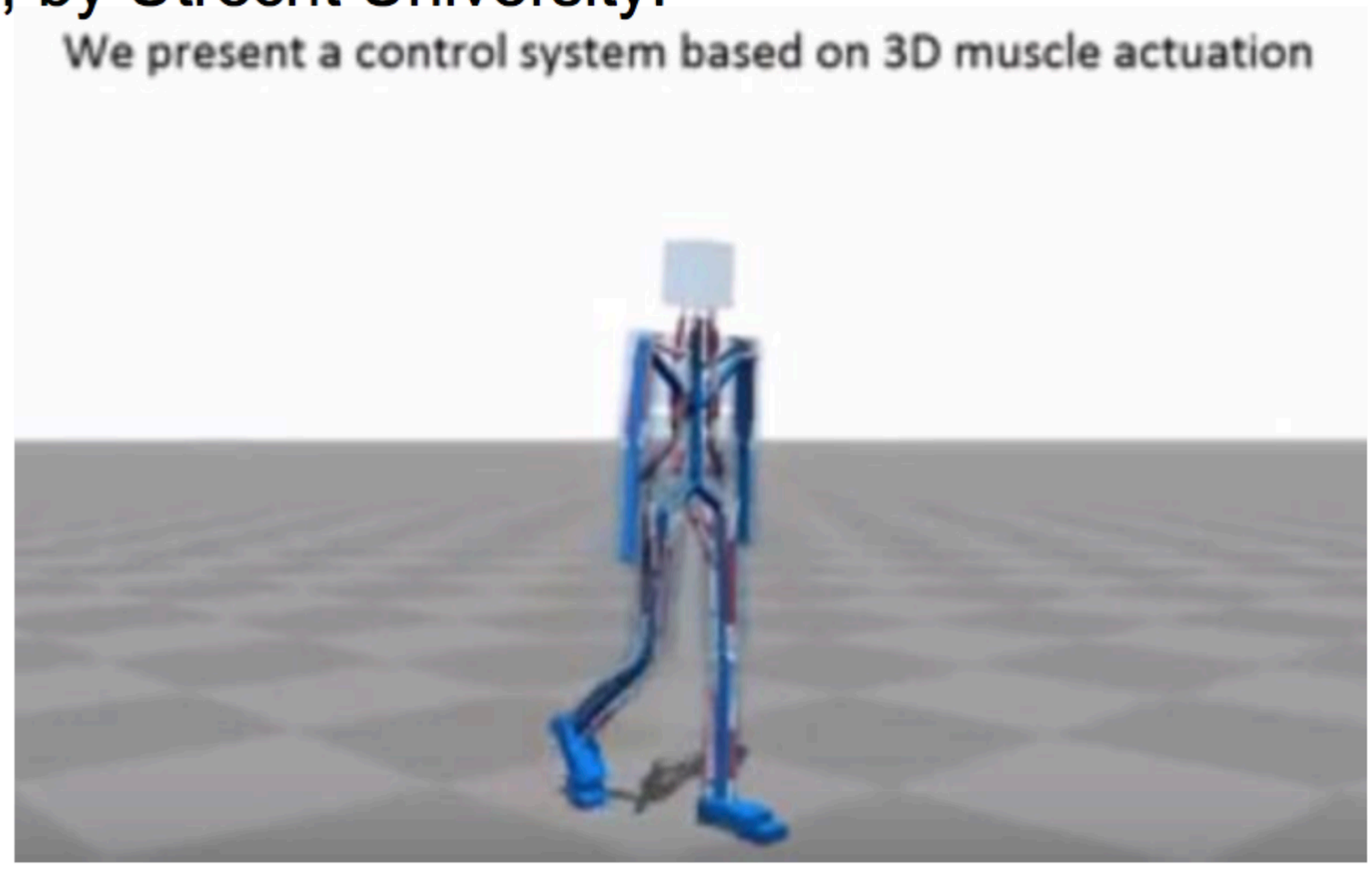
## Coffee Tasting Problem

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

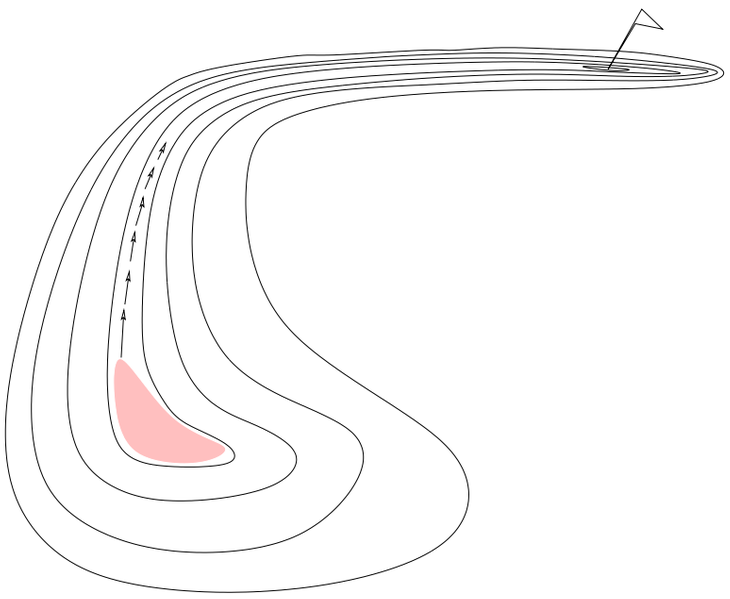
# Level Sets of a Function

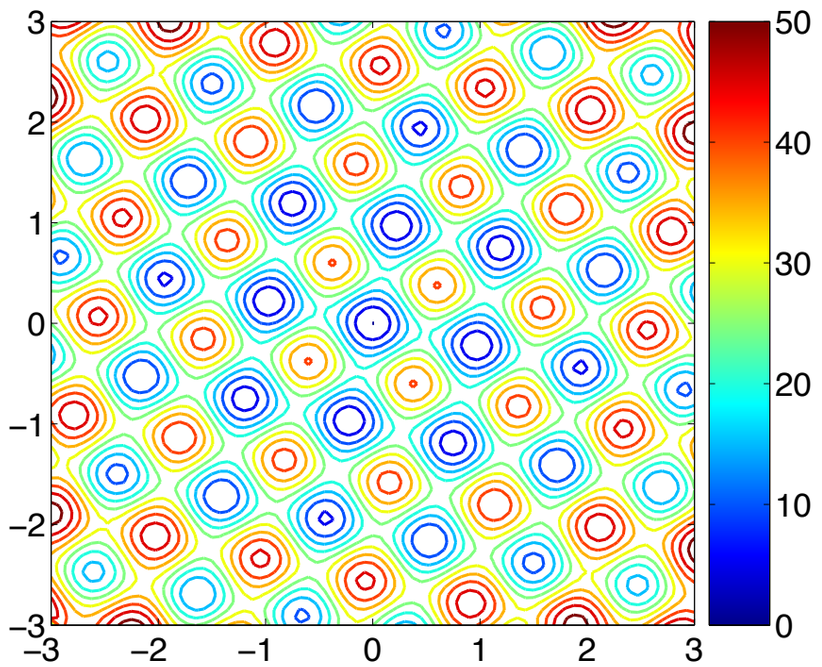
# Level Sets: Visualization of a Function

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

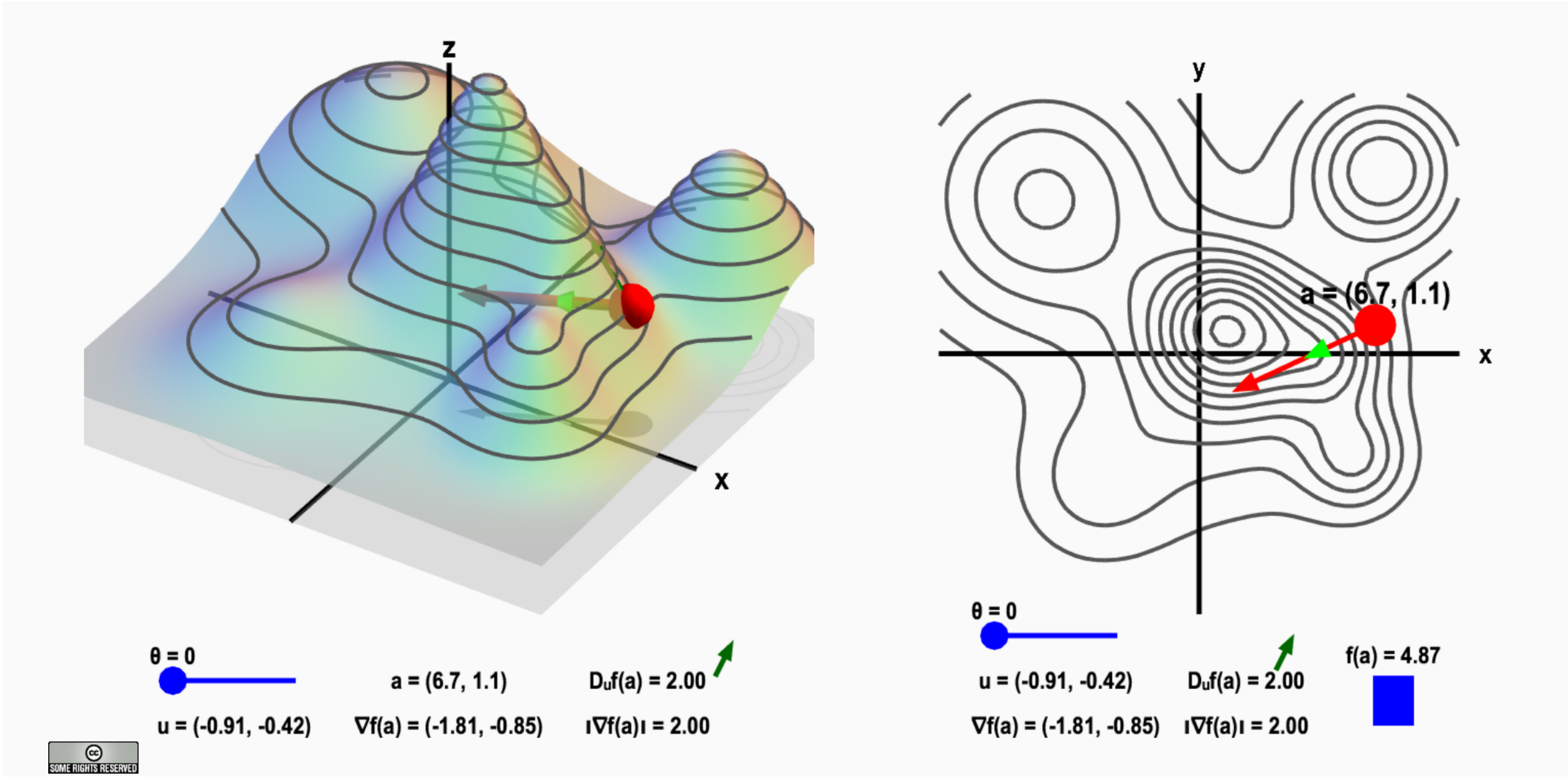
$$\mathscr{L}_c = \{x \in \mathbb{R}^n \,|\, f(x) = c,\}, c \in \mathbb{R}$$

## Examples of level sets in 2D





# Level Sets: Visualization of a Function

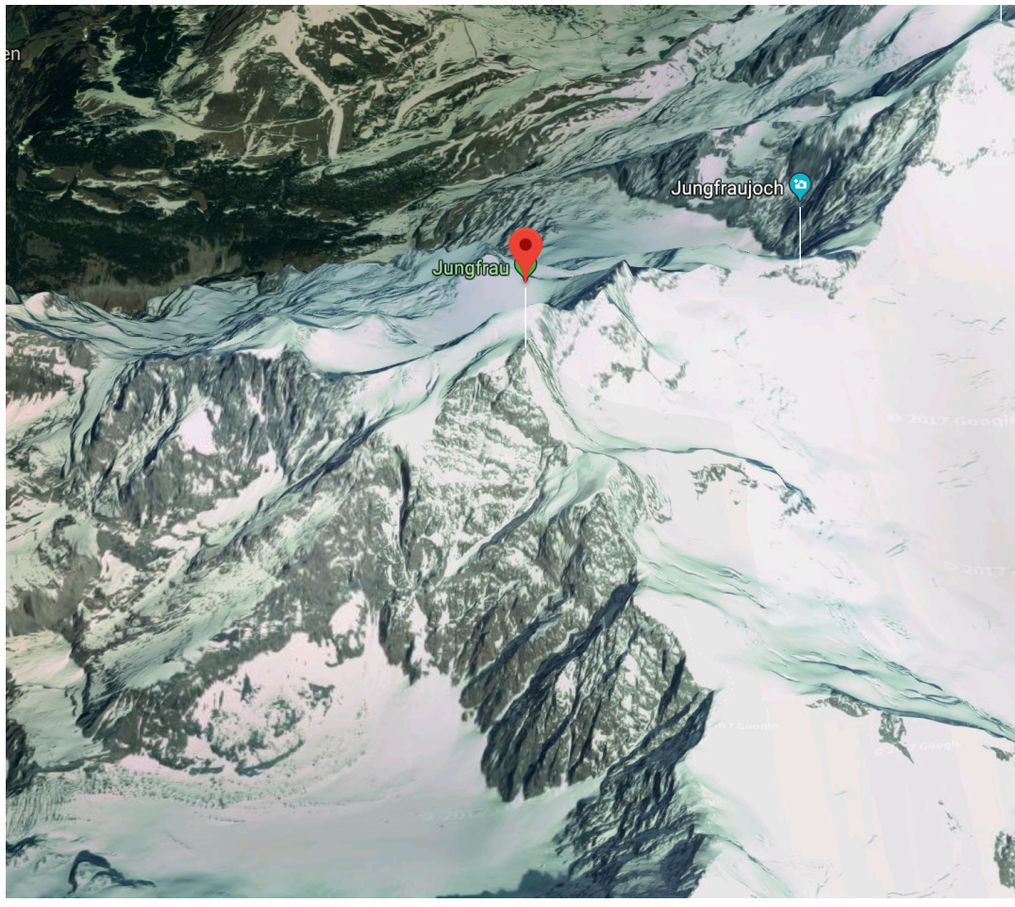




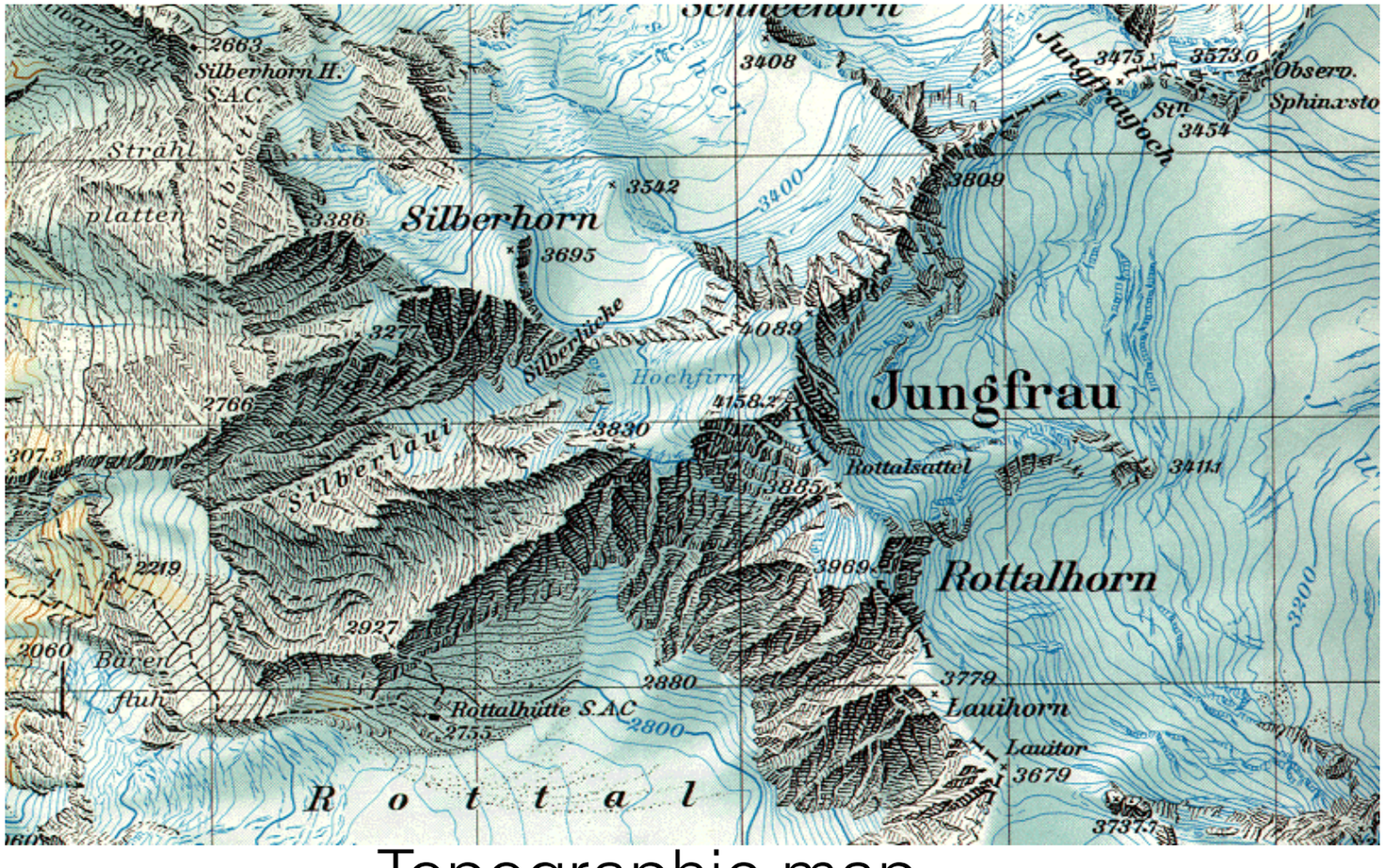

Source: Nykamp DQ, “Directional derivative on a mountain.” From *Math Insight*. http://mathinsight.org/applet/directional_derivative_mountain

# Level Sets: Topographic Map

The function is the altitude




3-D picture




Topographic map

# Level Set: Exercice

Consider a convex-quadratic function

$$f : x \mapsto \frac{1}{2}(x - x^*)^T H(x - x^*) = \frac{1}{2}\sum_i h_{i,i}\,(x_i - x_i^*)^2 + \frac{1}{2}\sum_{i \neq j} h_{i,j}\,(x_i - x_i^*)(x_j - x_j^*)$$

with $H$ a symmetric, positive, definite matrix

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

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

3. 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

# 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

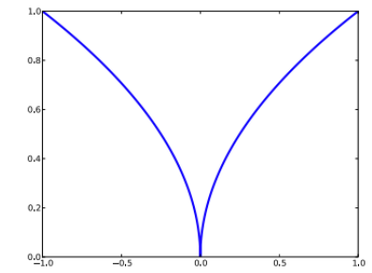
- dimensionality (size of search space)

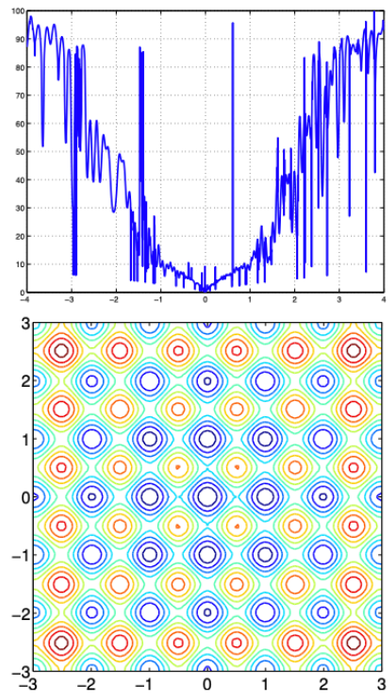
  (considerably) larger than three

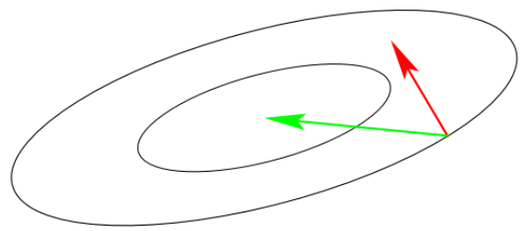
- non-separability

  dependencies between the objective variables

- ill-conditioning

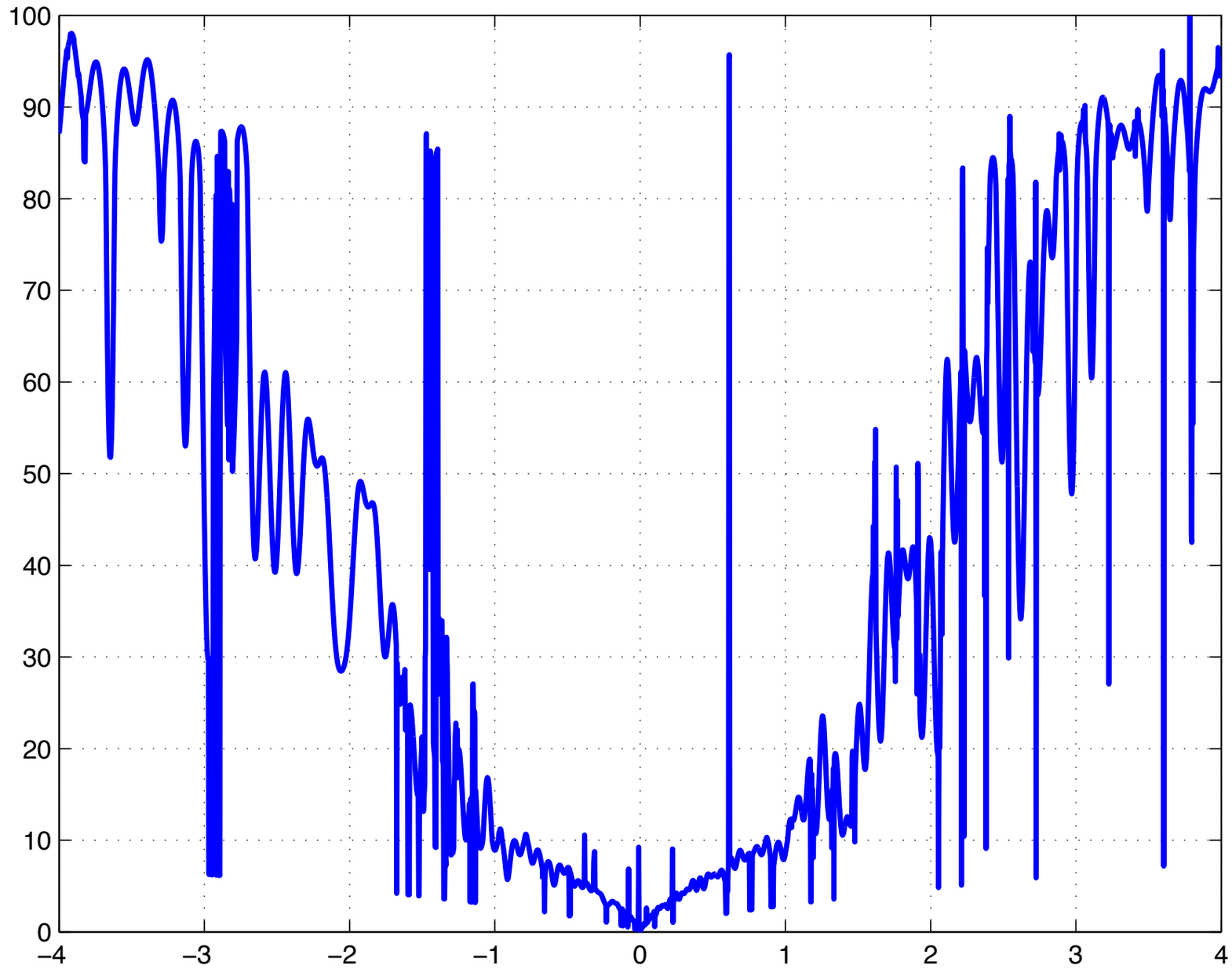






gradient direction Newton directio

# Ruggedness




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

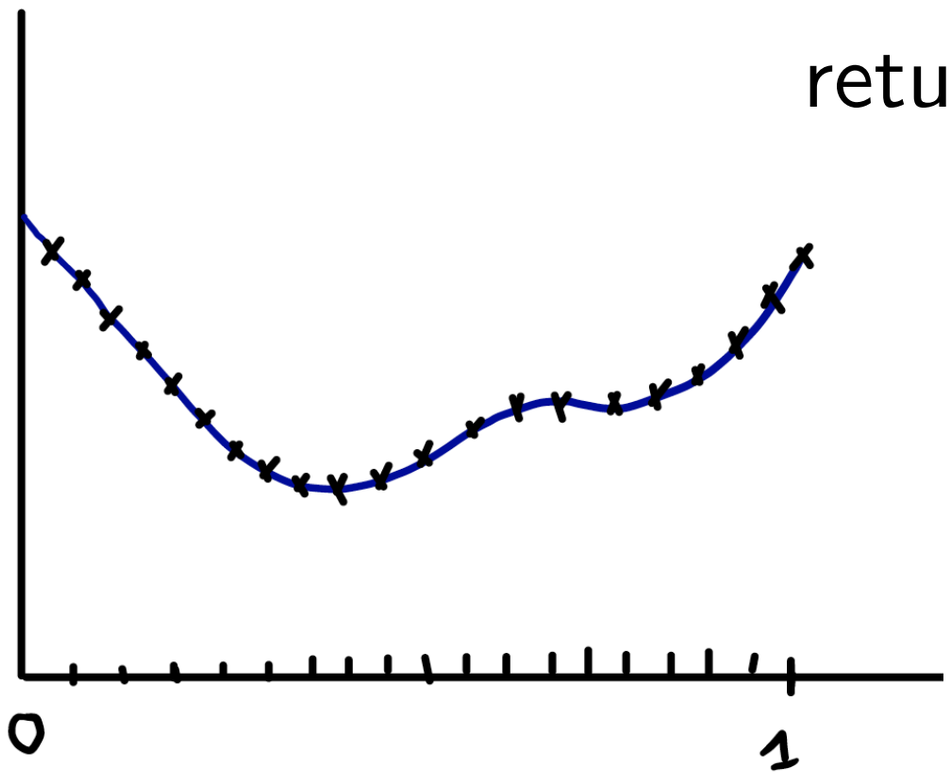
if n=1, which simple approach could you use to minimize:

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

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

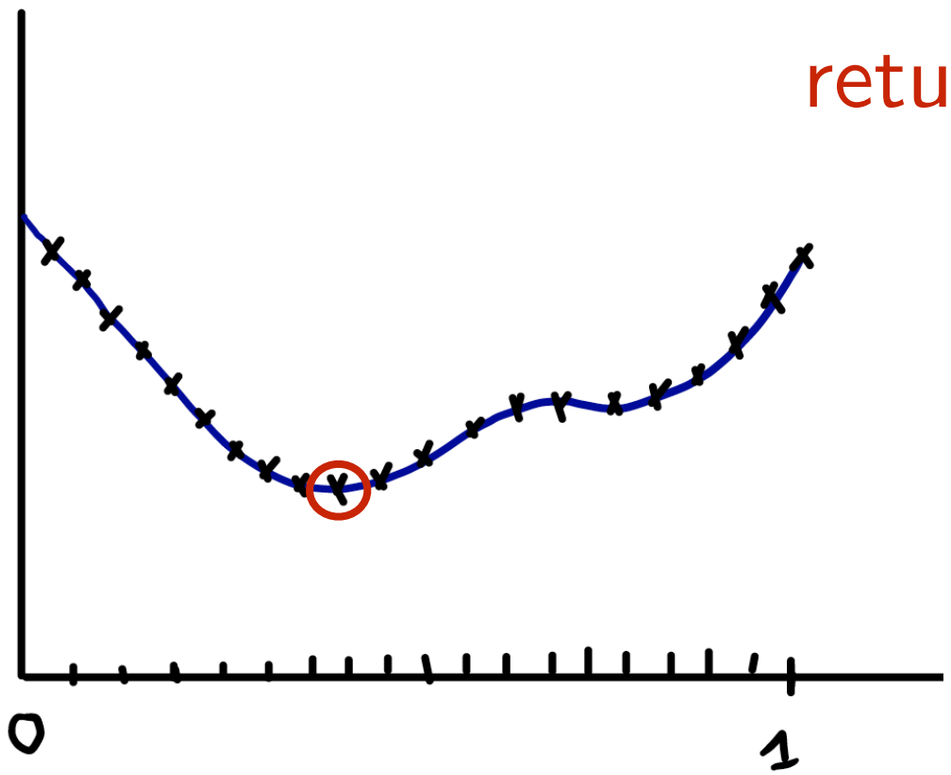
if n=1, which simple approach could you use to minimize:

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

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

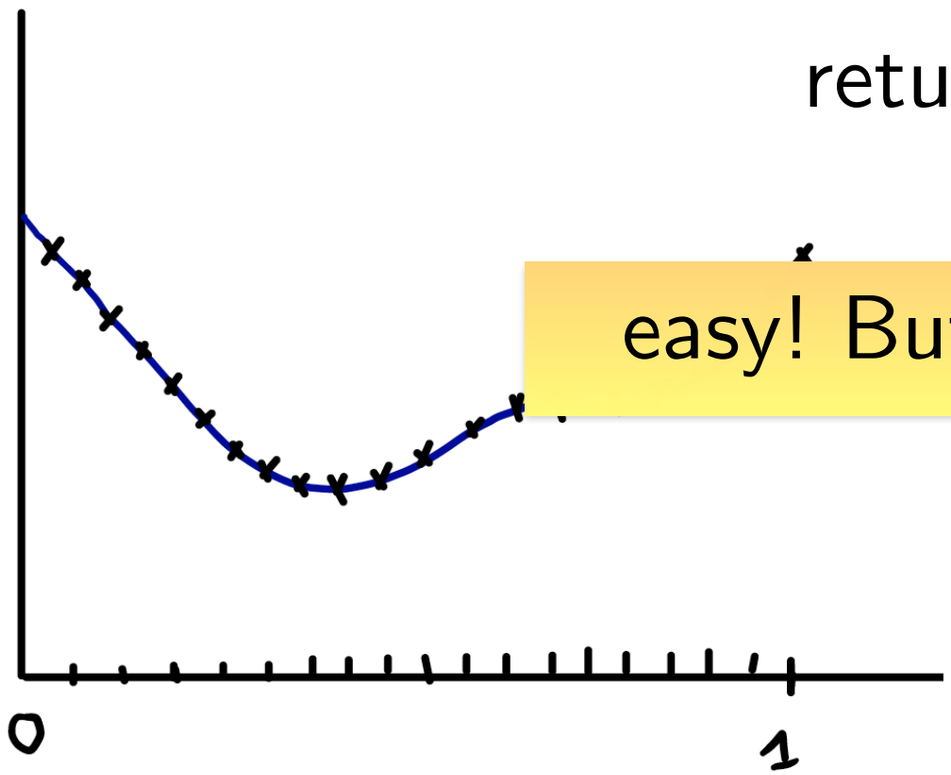
if n=1, which simple approach could you use to minimize:

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

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.

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 $f : x = (x_1, \ldots, x_n) \in \mathbb{R}^n \mapsto f(x) \in \mathbb{R}$, let us define the 1-D functions that are cuts of $f$ along the different coordinates:

$$f^i_{(x^i_1,\ldots,x^i_n)}(y) = f(x^i_1, \ldots, x^i_{i-1}, y, x^i_{i+1}, \ldots, x^i_n)$$

for $(x^i_1, \ldots, x^i_n) \in \mathbb{R}^{n-1}$, with $(x^i_1, \ldots, x^i_n) = (x^i_1, \ldots, x^i_{i-1}, x^i_{i+1}, \ldots, x^i_n)$

**Definition:** A function $f$ is **separable** if for all i, for all $(x^i_1, \ldots, x^i_n) \in \mathbb{R}^{n-1}$, for all $(\hat{x}^i_1, \ldots, \hat{x}^i_n) \in \mathbb{R}^{n-1}$

$$\text{argmin}_y f^i_{(x^i_1,\ldots,x^i_n)}(y) = \text{argmin}_y f^i_{(\hat{x}^i_1,\ldots,\hat{x}^i_n)}(y)$$

*a weak definition of separability*

# Separability (cont)

**Proposition:** Let $f$ be a **separable** then for all $x_i^j$

$$\operatorname{argmin} f(x_1, \ldots, x_n) = \left( \operatorname{argmin} f^1_{(x_2^1, \ldots, x_n^1)}(x_1), \ldots, \operatorname{argmin} f^n_{(x_1^n, \ldots, x_{n-1}^n)}(x_n) \right)$$

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

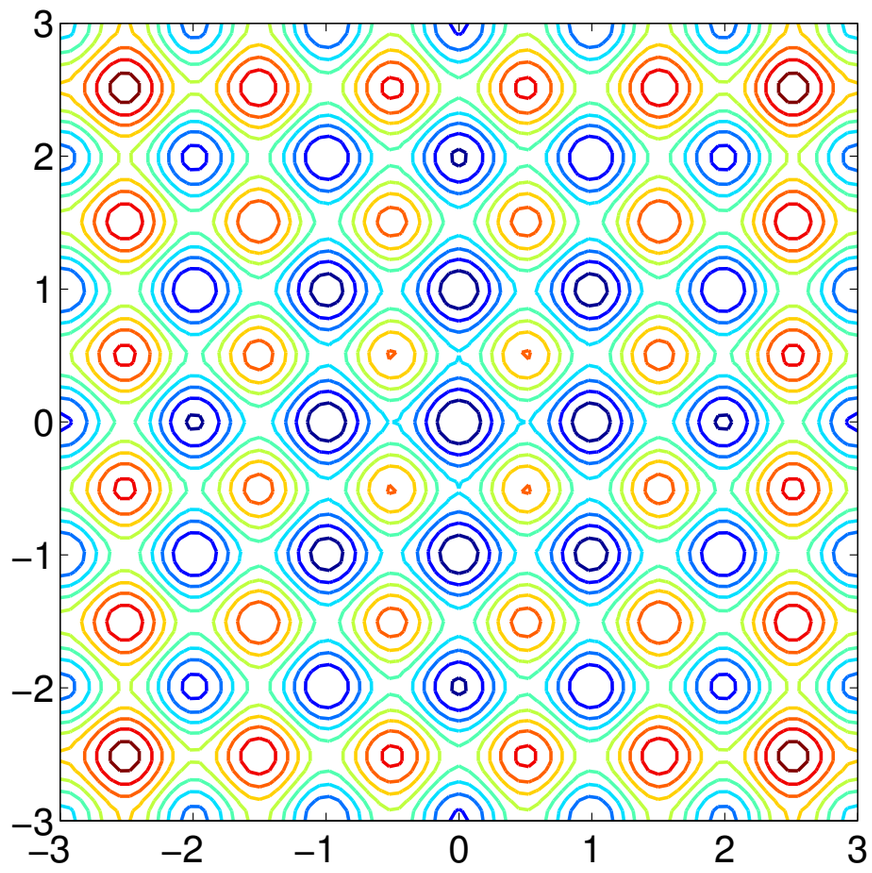
**Exercice:** prove the previous proposition

# Example: Additively Decomposable Functions

**Exercice:** Let $f(x_1, \ldots, x_n) = \sum_{i=1}^{n} h_i(x_i)$ for $h_i$ having a unique argmin. Prove that $f$ is separable. We say in this case that $f$ is additively decomposable.

**Example:** Rastrigin function

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



# 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)

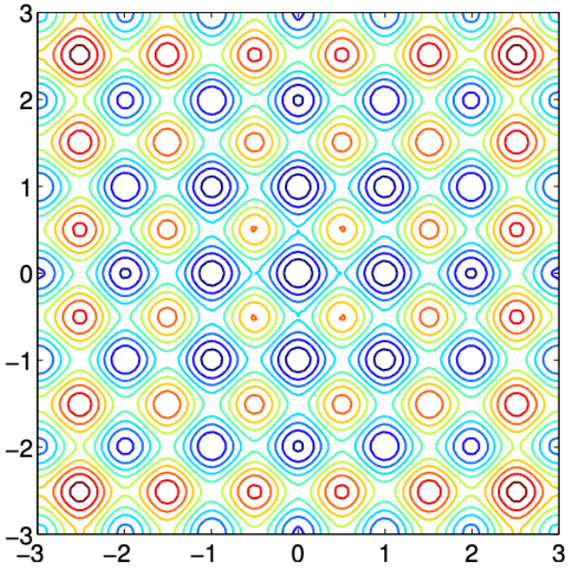
# Non-separable Problems

*Building a non-separable problem from a separable one*

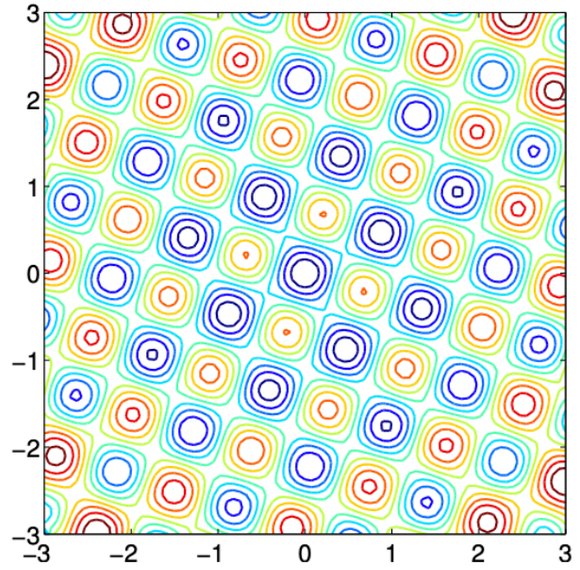
## Rotating the coordinate system

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

**R** rotation matrix




**R**




[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

**Exercice:** Consider a convex-quadratic function $f(x) = \frac{1}{2}(x - x^\star)H(x - x^\star)$ with $H$ a symmetric, positive, definite (SPD) matrix.

**1.** why is it called a convex-quadratic function? What is the H matrix of $f$ ?

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 Problems

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

**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 to you think it is the case?

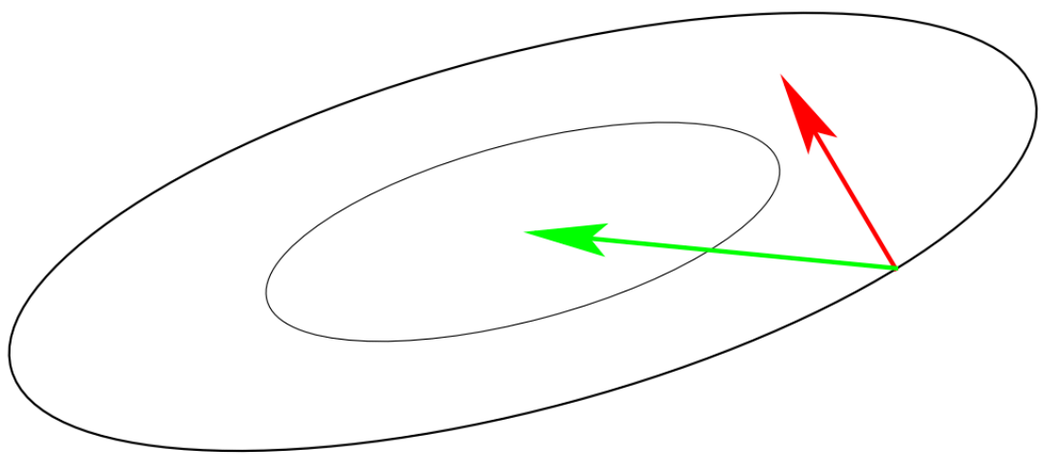
**5.** why are ill-conditioned problems difficult?

(see also **Exercice 2.5**)

# 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 $10^{10}$ are not unusual in real world problems.