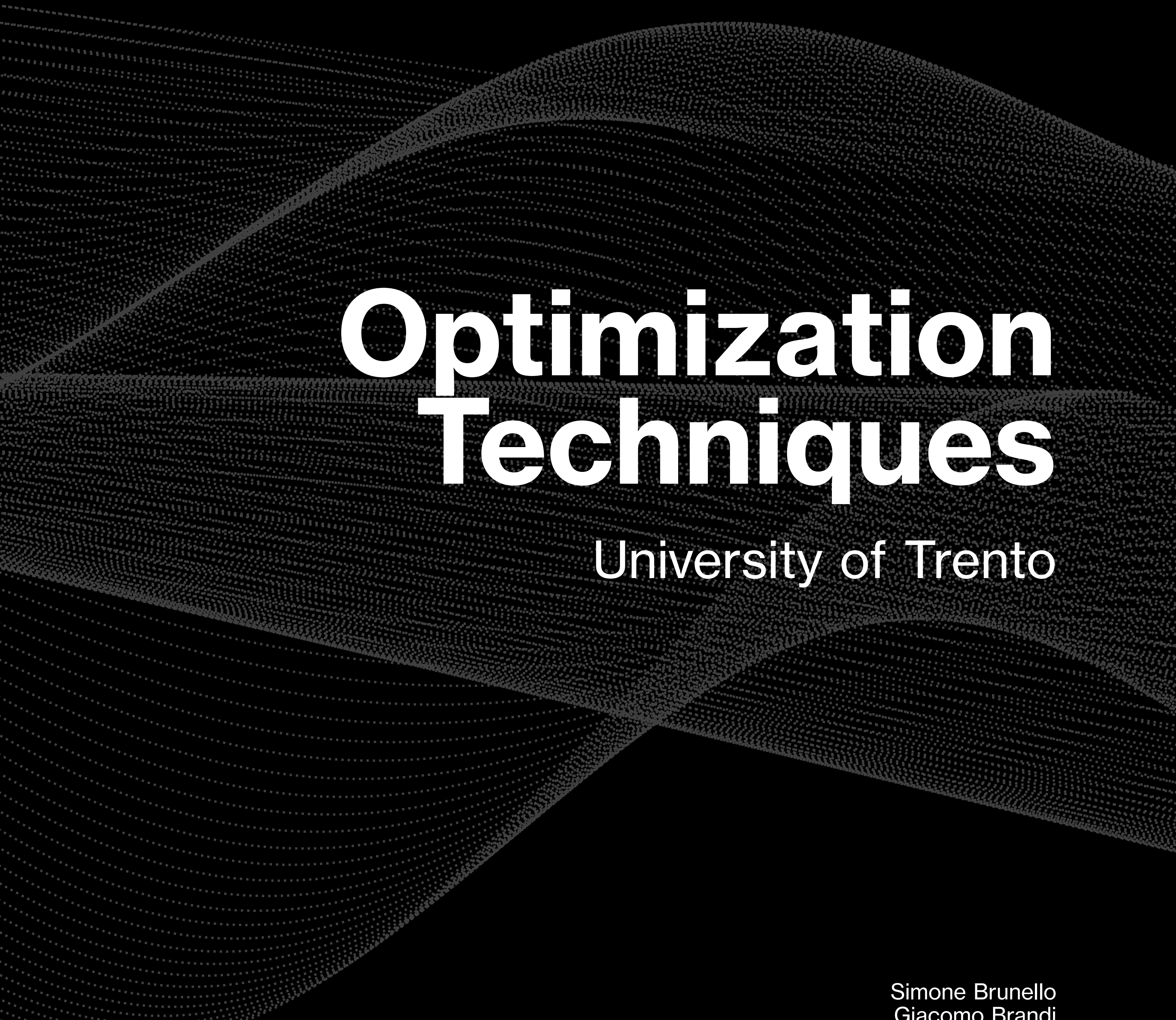


Optimization Techniques

University of Trento

Simone Brunello
Giacomo Brandi

1. OPTIMIZATION PROBLEMS

An optimization problem involves finding the best element among a set of possible alternatives that meet predetermined criteria. Optimization problems span many domains, including physics, business, and logistics.

Some examples of optimization problems include:

Rocket. You have to launch a rocket and land it at a certain point, you have to decide the angle with respect to the ground at which it is launched.

Travel Salesman's Problem. Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the starting city?

Knapsack. Given a set of items, each with a weight and a value, determine which items to include in the collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

Position of a Robot. We have a robot that needs to figure out where it is without a GPS antenna. Antennas have been installed nearby in known locations that can detect the angle between the line connecting the robot to the antenna and the line indicating the direction of the wheels. How can this data be used to calculate the robot's position?

Training of Neural Network. After all, neural networks are complicated optimization problems.

▶ **Simone B.** **TODO: mettere dei disegni per gli esempi precedenti.** ◀

Optimization consists of four steps:

- **Modeling**: in order to work on a problem with a computer, it is necessary to translate the problem into mathematical objects (variables, sets, functions, etc.) that represent it.
- **Solving**: Different problems have different characteristics and require analysis to determine the best technique. For example:
  - continuous/discrete: a deliveryman who has to decide how many pizzas to deliver each round must deliver an integer number of pizzas;
  - linear/nonlinear;
  - constrained/unconstrained: a car has a maximum speed;
  - numerical/analytical.

  After the analysis of the characteristics, it is necessary to find the best algorithm for the solution of the problem.
- **Verification**: every algorithm always return a result (in a certain sense, even a segmentation fault is a result). But is it correct? At this stage, the result is evaluated to see if it makes sense.
- **Sensitivity analysis**: the algorithm may be perfect and return the correct result for the desired problem, but the problem may have been inappropriately modeled, e.g. some approximations are inevitable, but if you go too far you will get useless results in reality.

There are no global solutions, but each problem is unique and requires different modeling, analysis, verification, and sensitivity analysis.

1.1. MODELING

Mathematics can represent physical, economic, social, and other phenomenon. These representations in mathematical objects are called *models*, and the process of creating a model is called *modeling*.

A good model must be:

- **accurate enought**: a good model describes phenomena with a high degree of accuracy;
- **sufficiently simple**: a model that is too complex would require more sophisticated algorithms and many computing resources to solve;
- **sufficiently robust/wide**: a good model will be flexible enough to fit a wide range of scenarios.

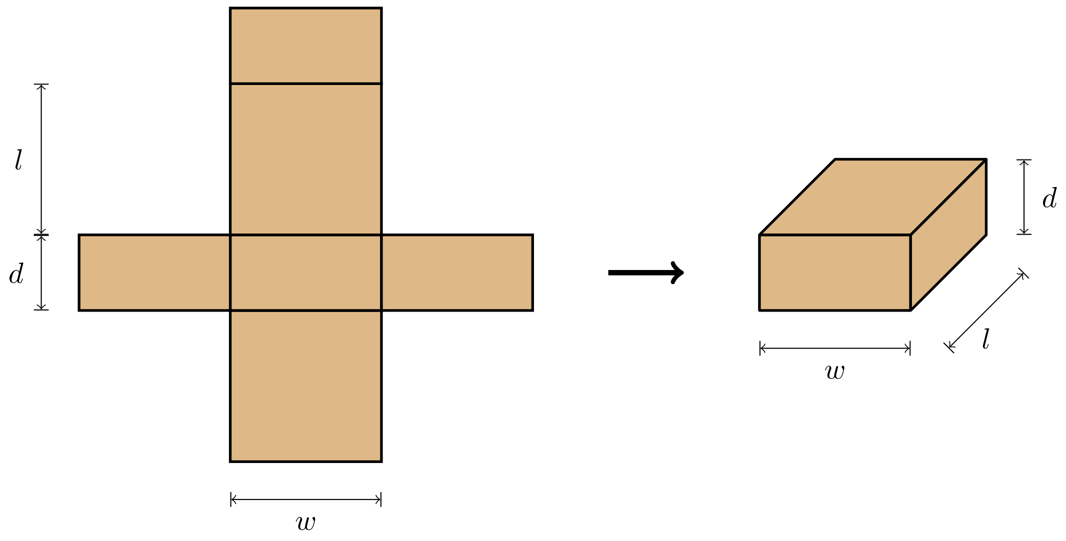
The modeling process involves three steps:

1. Identification of **state variables** (or decision variables): are things that can change in a problem (such as a robot's angle and a rocket's initial velocity). Decisione variables are represented as a column vector $\mathbb{R}^n$ denoted by x. For example, the variable of a car might be:

$$y = \begin{bmatrix} angle \\ velocity \\ distance \end{bmatrix}$$

2. Identification of a **ojective function** (or cost/gain function): is the function that evaluates how "good" a solution is. Formally it is a function $f : \mathbb{R}^n \to \mathbb{R}$. In the car example, an objective function is the distance to the specified destination.
3. Mathemarical description of the **constraints** (or circumstances): it specifies the constraints that the solution must have. For example, they can be given by physics (e.g. that the velocity must be positive). The set of all solutions satisfying the constraints is called **feasible set** and is denoted as Ω.

Example. Consider a box in which we want to maximize the volume by fixing an area.



The problem can be modeled as follow:

- **Decision variables**: l, h, w in which l is the length, h is the height, and w is the width of the box.
- **Constraints**: $2lv + 2wh + 2lw = A$ constant. In other words, the area of the box has to be fixed and has to be the same in all the solutions.
- **Objective function**: $V(l, h, w) = l \cdot h \cdot v$. Note: the goal is $\underset{l,h,w}{max}\ V(l, h, w)$.

1.2. CONSTRAINTS

As anticipated in Section 1.1, not all possible solutions are good. Depending on the actual problem we are solving, there may be limitations, for example:

- we need to calculate the optimal speed of a train, but we cannot choose a negative speed;
- to allocate a portfolio, we have to decide the percentage of stocks and bonds based on different parameters, the sum of bonds and stocks must be 100%;
- we have to find the best arrangement of packages in a van. The total weight must not exceed the permitted weight of the vehicle.

Constraints take the form of real-valued functions in two possible forms, let $x \in \mathbb{R}^2$ a solution, the contraints types are:

$$\begin{aligned} &h(x) = 0 && \text{equality constraints} \\ &g(x) \leq 0 && \text{inequality constraints} \end{aligned}$$

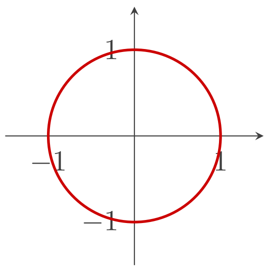
Formally, let m the number of *equality constraints* and p the number of *inequality contraints*, the *feasible set* $\Omega \subseteq \mathbb{R}^n$ is

$$\Omega = \left\{ x \in \mathbb{R}^n \,\middle|\, \begin{array}{ll} h_i(x) = 0, & i = 1, ..., m \\ g_j(x) \leq 0, & j = 1, ..., p \end{array} \right\}$$

Taking the above examples again, the train speed is x and the contraint is $g_1(x) = -x$. In the wallet example, x_1 reperesents the percentage of the bonds and x_2 reperesents the percentage of the stocks, the contraint is $h_1(x_1, x_2) = x_1 + x_2 - 100$. Finally, considering the packages in the van $x_1, ..., x_n$ are the weight of the n packages, and the constant w is the maximum weight, the contraint is $g_1(x1, ..., x_n) = x_1 + x_2 + ... + x_n - w = \sum_{i=1}^{n} x_n - w$.

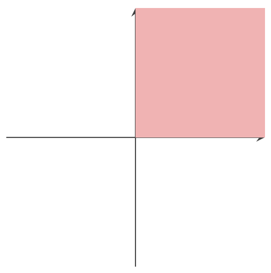
Circle. The choosen solution must be a point in the circumnference of radius 1.

$$\begin{aligned} radius = 1 \Rightarrow \\ radius - 1 = 0 \Rightarrow \\ h_1(x, y) = x^2 + y^2 - 1 \end{aligned}$$



First quadrant. We want to limit the solutions to be in the first quadrant. In order to do that, we introduce the following condition:

$$x \geq 0 \wedge y \geq 0 => \begin{cases} g_1(x, y) = -x \\ g_2(x, y) = -y \end{cases}$$



Colors. n primary colors are used, which are mixed to form the color required by the client. Of each color x_i, the operator must choose how much to put in and the sum must be exactly the liters requested by the user. To formalize that the sum of the quantities of each color must be exactly S, the constraint is:

$$h_1(x_1, ..., x_n) = x_1 + x_2 + ... + n_x - S$$

1.3. MINIMIZATION

To solve an optimization problem, it is necessary to introduce the concept of an objective function (see Section 1.1), which, given a solution, gives us a value that indicates how "good" it is. If we could find the solution where the function is maximum (or minimum, as the case may be), we would have solved the problem.

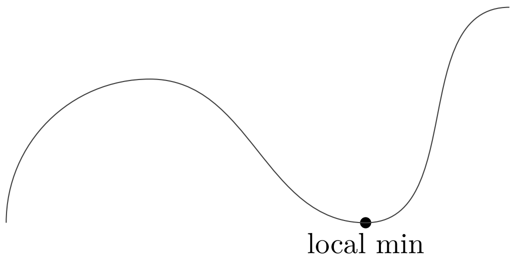
In the end, solving an optimization problem is nothing more than finding the minimum of a function in n variables, where n is the number of parameters that can vary (state variables). Conventionally in optimization problems we focus on finding the minimum, if the best solution is found at the maximum, we simply invert the function and still find the minimum.

Definition 1 (minimization problem). Find the $x \in \Omega$ that minimize f.

Definition 2 (local minimum). A point $x^* \in \Omega$ is a local minimum (or relative minimum) for f in Ω if $\exists \epsilon > 0$ such that

$$f(x^*) \leq f(x) \qquad \forall x \in \Omega \text{ such that } dist(x, x^*) \leq \epsilon$$

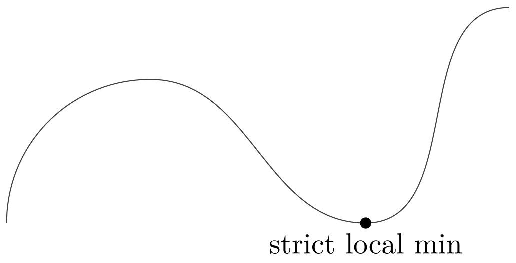
In other words, it is a point where the function has a lower value than all other nearby points.

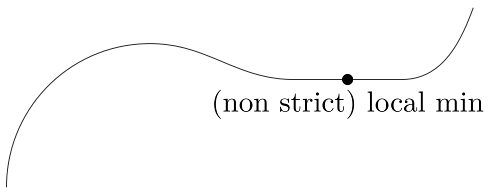


Definition 3 (strict local minimum). A point $x^* \in \Omega$ is a strict local minimum for f in Ω if $\exists \epsilon > 0$ such that

$$f(x^*) < f(x) \qquad \forall x \in \Omega \text{ such that } x^* \neq x \wedge dist(x, x^*) \leq \epsilon$$

That is, the function has a higher (not equal) value at nearby points. In other words, "plateaus" are not allowed.

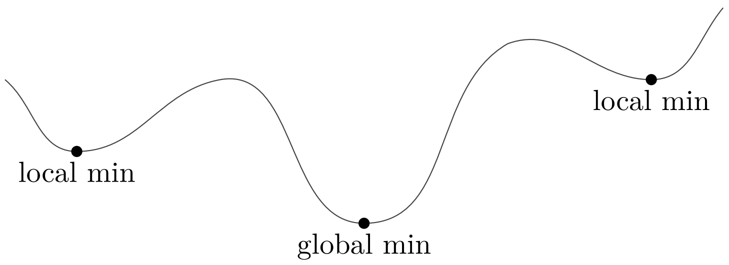




Definition 4 (global minimum). A point $x^* \in \Omega$ is a global minimum for f in Ω if

$$f(x^*) \leq f(x) \qquad \forall x \in \Omega$$

In other words, the smallest local minimum.



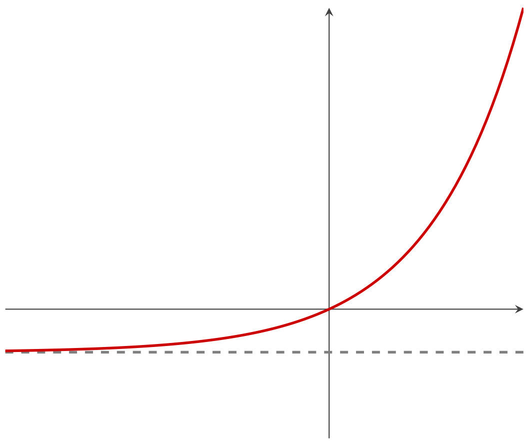
In the case of a global minimum, we write $f(x^*) = \underset{\Omega}{\texttt{min}} f$ and $x^* = \underset{\Omega}{\texttt{argmin}} f$.

Proceed in a similar manner to create the maximum (`max`).

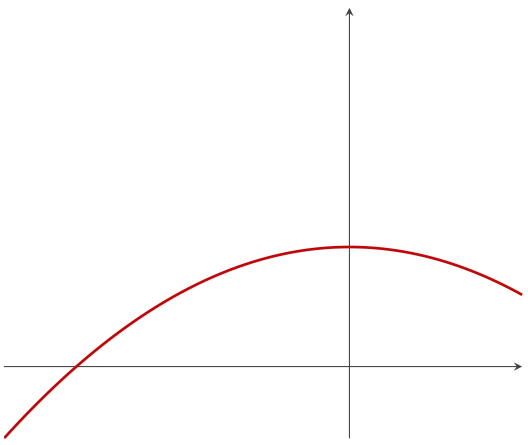
> Note: The words *minimum* and *maximum* refer to the coordinates of the point, not the value of the function.

> **Definition 5** (infimum)**.** The infinum of a function $f : \Omega \to \mathbb{R}$ is the **value** $\mathcal{M}$ such that:
> - $f(x) \geq \mathcal{M}$ for every $x \in \Omega$;
> - there is no $\mathcal{M}' > \mathcal{M}$ such that $f(x) \geq \mathcal{M}'$ for every $x \in \Omega$.

Every function has a infinum! It is not always a minimum. In fact, some functions have no minimum.



$\texttt{inf} f = -1$
$\nexists x$ such that $f(x) = -1$, then $\nexists \texttt{min} f$

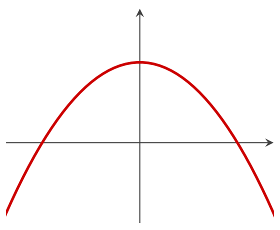


$\texttt{inf} f = -\infty$
$\nexists x$ such that $f(x) = -\infty$, then $\nexists \texttt{min} f$

When the infinum is $-\infty$ the function is unbounded.

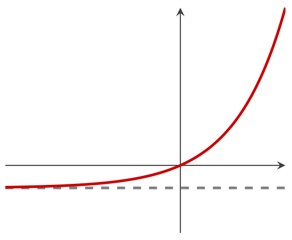
The minimum may not exist. In this case, the computer will produce a result that is meaningless. Before choosing an algorithm, it is necessary to verify that the minimum lies within the space of possible solutions Ω.

There are three reasons why the minimum may not exist:



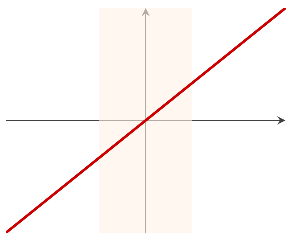
$\Omega = \mathbb{R}$

The function is not limited inferiorly. That is, it goes down to infinity.



$\Omega = \mathbb{R}$

The function is limited by -1, but there is no such number x so that $f(x) = -1$ (i.e., it never reaches it so it cannot be the minimum).



$\Omega = (-1, 1]$

-1 is the minimum, but it is not a viable solution because it does not meet all constraints ($-1 \notin \Omega$)

1.4. DERIVATIVE

Minimum? Huh. I learned how to do this in Calculus 1! I calculate the derivative and set it to $f = 0$. This way I get the points where the function “flattens out”. Then I calculate the second derivative and find the convexity, and if it remains positive both before and after the points identified with the first derivative, then that is a minimum. To find the global minimum, I go through all the minima to find the smallest one.

Sorry, but no. That is, the method is correct, but it does not take into account four problems:

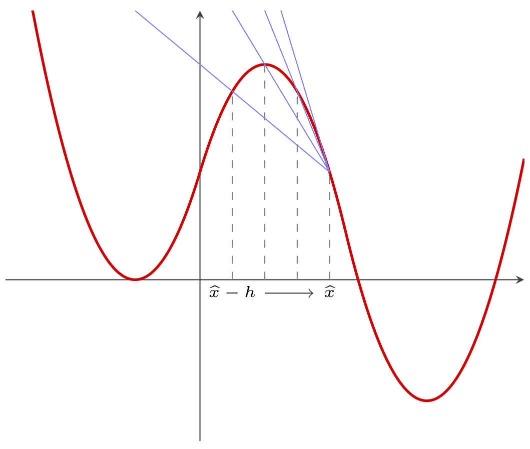
- How to calculate the derivative? The function is an objective function that models a problem in reality. Dream of the (relatively) easy to derive functions you find in Calculus 1, reality is not a polynomial. Calculating from the derivative is a difficult problem, even for a computer, if the function is complex.
- If the problem seems complex for one function in one variable, let alone thousands! Calculating the gradient, the equivalent of the derivative in more than one dimension, is a computationally time-consuming task.
- The function may not be derivable. Or the function may allow the first derivative, but not the second. Even then, it is not possible to follow this method.
- We do not have that function. Yes, we are asked to minimize a function that is not even given. It is difficult to model reality with a function with its own formula. In practice, we often have an external program that calculates the function at a given point, but tells us nothing else. So it is not possible to calculate the derivative.

The derivative does not solve the problem, but it does give us a very important piece of information: it tells us whether the function is going up or down. It is therefore worth reviewing the concept of the derivative and introducing the concept of the gradient (the equivalent of the derivative for functions in two or more variables).

Definition 6 (Derivative)**.** Let $f : \mathbb{R} \to \mathbb{R}$ and $\widehat{x} \in \mathbb{R}$, the derivative $f'(\widehat{x})$ is:

$$f'(\widehat{x}) = \lim_{h \to 0} \frac{f(\widehat{x} + h) - f(\widehat{x})}{h}$$

The derivative is the slope of the function, one point at a time.



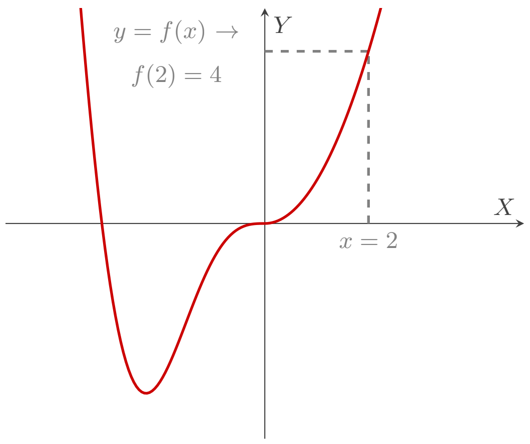
1.5. FROM 1 TO n DIMENSIONS

Calculus 1 covers functions in one dimension only. In this section, we first introduce functions in two dimensions and then extend the concept to an arbitrary number of dimensions. If you have already studied Calculus 2, the content of this section will probably be obvious.

1.5.1. 1-dimension

Functions in one dimension have $\mathbb{R}$ as domain and $\mathbb{R}$ as co-domain. It is written as $f : \mathbb{R} \to \mathbb{R}$. An example of a one-dimensional function is $f(x) = x^2 - sin(x)$.

Conventionally, we use the X-axis as the domain (the "input" of the function) and the Y-axis as the co-domain (the "output" of the function).



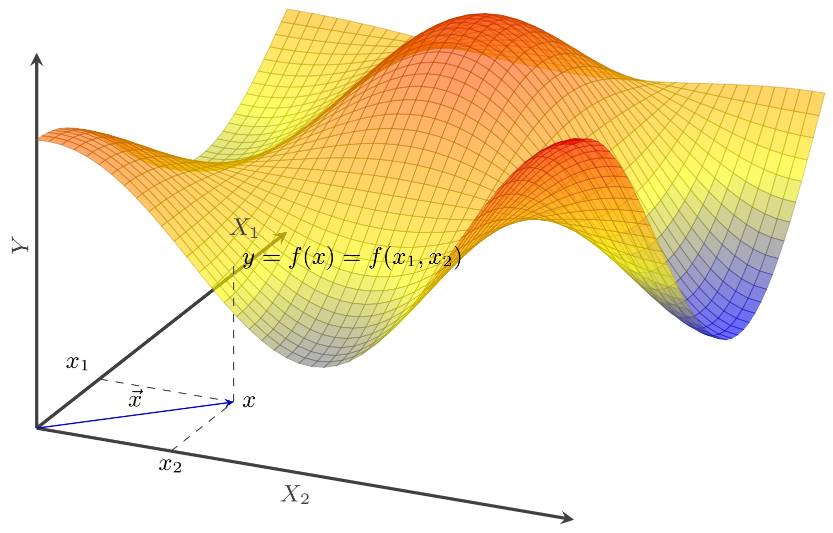
In one dimension, it is possible to introduce the concept of derivative, which indicates the slope of the function point by point (see **??**).

1.5.2. 2-dimentions

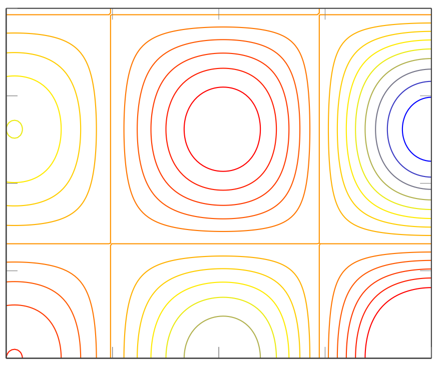
Remember the default wallpaper in Windows XP? The one with the grass hills? Imagine we want to represent the height of each point with a function, in other words, given a coordinate we want the function to tell us how high the terrain is at that point. Coordinates are the domain or input of the function, while height is the co-domain or output of the function. The coordinates cannot be represented by one real number, but two are needed.

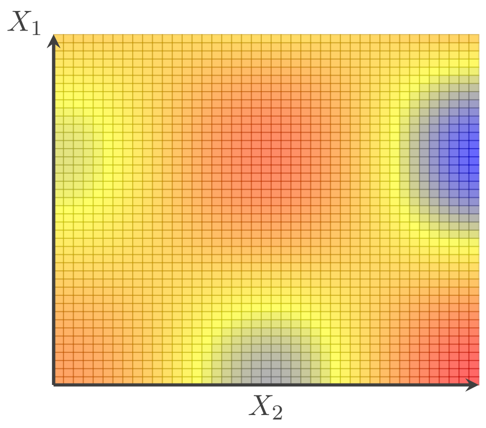
The domain consists of two real numbers in a tuple, e.g. $(\frac{1}{2}, -4)$. So it is $\mathbb{R} \times \mathbb{R}$. Instead, the co-domain remains a real number $\mathbb{R}$. A two-dimensional function is $f : (\mathbb{R} \times \mathbb{R}) \to \mathbb{R}$, which can be also written as $f : \mathbb{R}^2 \to \mathbb{R}$.

The two-element tuple used as input is represented as $x \in \mathbb{R}^2$ and its components are $x = (x_1, x_2)$. Then we will use $f(x)$ and $f(x_1, x_2)$ interchangeably. It can be seen that an element of the domain is nothing else than a vector $\vec{x} = (x_1, x_2)$ that, starting from the origin $\vec{O} = (0, 0)$, connects the point (x_1, x_2).



To visualize functions in two dimensions, the contour representation can be used; planes parallel to the plane passing through the X_1 and X_2 axes are introduced and the intersections are displayed. The color depends on the height of the plane. Another interesting representation in a plane of a function in two variables is that on the basis of color: the colder the color, the lower the value of the function, similarly, the warmer the higher the value. The previous function is represented by contour lines on the left and by color on the right.





Gradient. As with one-variable functions, we want to know the slope of the function at a given point; to do this, we introduced the concept of derivative. For functions with two (or more) dimensions, we use the concept of gradient $\nabla f(x_1, x_2)$.

To understand whether and how the function grows or decreases, we need to solve the problem one dimension at a time. We compute the derivative in dimensions X_1 and X_2 separately.

calculates how the value of the function changes when x_1 is

treat x_2 as a constant

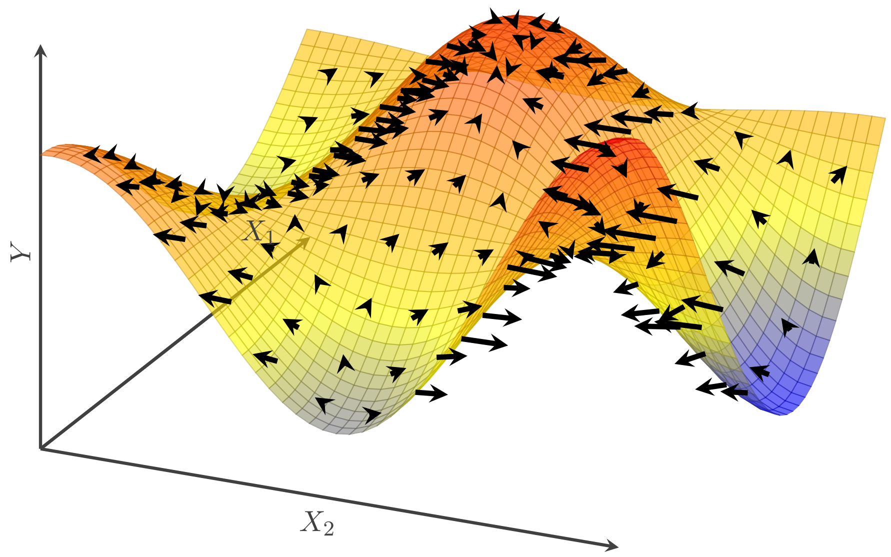
- $\frac{\partial f}{\partial x_1}(x_1, x_2) = \lim_{h \to 0} \frac{f(x_1+h, x_2) - f(x_1, x_2)}{h}$

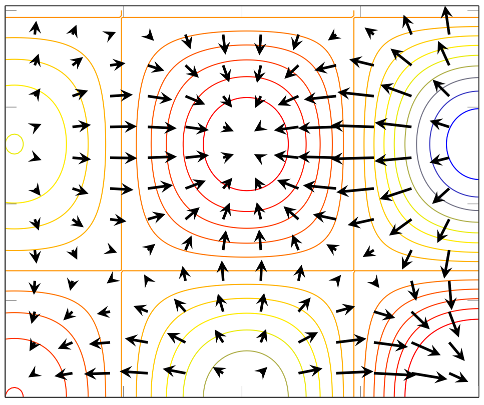
- $\frac{\partial f}{\partial x_2}(x_1, x_2) = \lim_{h \to 0} \frac{f(x_1, x_2+h) - f(x_1, x_2)}{h}$

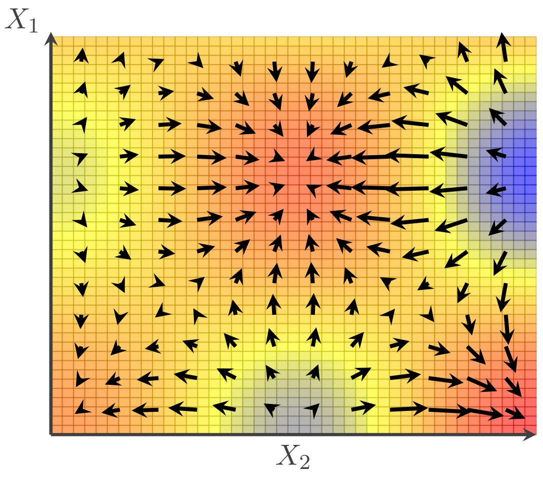
In order to obtain the gradient, we construct a vector in which each component represents how the function varies in one of the two dimensions; the variation is exactly the partial derivative calculated above.

$$\nabla f(x_1, x_2) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x_1, x_2) \\ \frac{\partial f}{\partial x_2}(x_1, x_2) \end{bmatrix} = \begin{bmatrix} \lim_{h \to 0} \frac{f(x_1+h, x_2) - f(x_1, x_2)}{h} \\ \lim_{h \to 0} \frac{f(x_1, x_2+h) - f(x_1, x_2)}{h} \end{bmatrix}$$

Going back to the hill example, the gradient represents the direction in which a drop of water would fall if it were dropped exactly at the point where the derivative is calculated.







1.5.3. n-dimension

Functions in more than two dimensions are difficult (if not impossible) to represent graphically, but it is still possible to extend the concepts seen in **??** for two dimensions to an arbitrary number of dimensions.

Let n be the dimensions, $\mathbb{R}^n$ the domain, and $\mathbb{R}$ the co-domain. An element of the domain is a tuple $x = (x_1, x_2, ..., x_n)$, where $x \in \mathbb{R}^n$. A function f in n dimensions is $f : \mathbb{R}^n \to \mathbb{R}$.

We can compute the partial derivative for all n dimensions:

- $\frac{\partial f}{\partial x_1}(x) = \lim_{h \to 0} \frac{f(x_1+h,\ldots,x_2)-f(x_1,\ldots,x_2)}{h}$
- $\frac{\partial f}{\partial x_2}(x) = \lim_{h \to 0} \frac{f(x_1,x_2+h,\ldots,x_n)-f(x_1,x_2,\ldots,x_n)}{h}$
- $\ldots$
- $\frac{\partial f}{\partial x_n}(x) = \lim_{h \to 0} \frac{f(x_1,\ldots,x_n+h)-f(x_1,\ldots,x_n)}{h}$

We can use partial derivatives to define the gradient.

> **Definition 7** (Gradient)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a function in n dimensions. The gradient is:
>
> $$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}$$

1.6. LABORATORY

This chapter is introductory and does not present any optimization algorithms, but it does introduce two very important concepts: the derivative and the gradient. The goals of this chapter are to become familiar with Python and numpy, learn how to write the pseudocode of the algorithms, and then convert it to Python.

1.6.1. Environment

Make sure you have Python installed. If you don't, follow the tutorial on the official site[1].

To make sure you have all the libraries installed, run:

```
pip install numpy, matplotlib
```

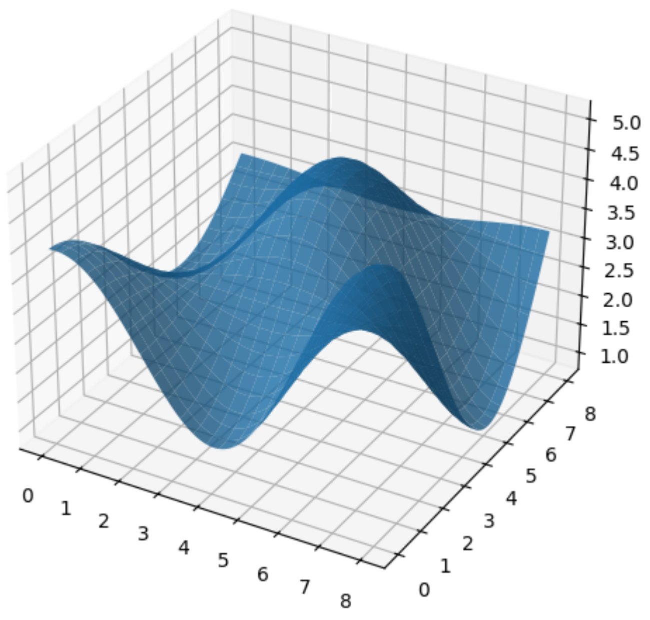
I recommend that you create a folder where you put all the lab files. To test how everything works, let's create a file called `test.py` and write into it:

```python
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np

def f(x):
    return np.cos(x[0], 2) + np.pow(x[1], 2) - 1.5 * x[0] * x[1]

def visualize(f):
```

[1] `https://www.python.org/`

```
    x = np.linspace(-10, 10, 100)
    y = np.linspace(-10, 10, 100)
    X,Y = np.meshgrid(x, y)
    Z = f([X, Y])

    fig = plt.figure(figsize=(14,6))

    ax = fig.add_subplot(1, 2, 1, projection='3d')
    ax.plot_surface(X, Y, Z, rstride=4, cstride=4, linewidth=0, alpha=.8)

    plt.show()

def main():
    visualize(f)

if __name__ == '__main__':
    main()
```

Run the file and if the following window appears, everything is working correctly!



1.6.2. Exercise: derivative

Write the pseudocode that calculates the derivative. The function must have three parameters: a function, a point x, and a value of h. h is optional and if not set takes the value of 0.01.

A possible solution is:

```
Function derivative(f, x, h ← 0.01):
    $fx \leftarrow f(x)$
    $fdx \leftarrow f(x+h)$
    return $(fdx - fx)/h$
```

Now, implement the function in Python. Create a file called `lab1.py`, then copy and complete the following function:

```python
import numpy as np

def derivative(f, x, h=0.01):
    # code
    return ...
```

Add a main function where you create a function, select a point, and call the derivative function to test if it works.

```python
def main():
    def f(x):
        return x**2 + 2*x + 5

    der = derivative(f, 3)
    print("The result is {}".format(der))
```

Finally, we need to invoke the main function when the program is executed.

```python
if __name__ == '__main__':
    main()
```

The result should be:

```python
import numpy as np

def derivative(f, x, h=0.01):
    # code
    return ...

def main():
    def f(x):
        return x**2 + 2*x + 5

    der = derivative(f, 3)
    print("The result is {}".format(der))

if __name__ == '__main__':
    main()
```

The output sould be:

```
The result is 8.009999999999806
```

1.6.3. Excercise: gradient

Write the pseudocode to compute the gradient in any number of dimensions, the function template is:

```
Function gradient($f, x = (x_1, ..., x_n), h \leftarrow 0.01$):
    // code
    return...
```

Then add a function named *gradient* to `lab1.py` with the following structure:

```
def gradient(f, x, h=0.01):
    # code
    return ...
```

and modify the main function to add:

```
def main():
    # previous code

    def f2(x):
        return np.cos(0.8 * x[0]) * np.cos(0.6 * x[1]) * np.exp(0.1 * x[0]) + 3

    x = [1, 4]
    grad = gradient(f2, x)
    print("The gradient of f2 in (1, 4) is ({}, {})".format(grad[0], grad[1]))
```

The output should be:

```
The gradient of f2 in (1, 4) is (0.41316061821610184, -0.3110320108611564)
```