# Course introduction

## Lecture 1

Course of:
**Signal and imaging acquisition and modelling in environment**

06/03/2024

*Federico De Guio - Matteo Fossati*

# General info about the course

- The course is organized in **frontal lectures** and **laboratory sessions**
  - Even during the lectures, you are often required to write some code
  - The lecture part will be recorded and made available the week after
- All the exercises will be in **Python**
  - No need to be an expert, we will start with a Python introduction
  - We will use Colab to code → shared jupyter notebooks, access to GPUs
- **Please answer the survey on the eLearning page** if you didn't do it yet.
- Requisites: **bring your laptop**
  - Groups of **2 persons** can be formed, but it is better if everybody has its own laptop

## Exam

- During the course we will propose some **extended exercises**
  - Most likely the time in the class won't be enough to complete them. In case you can work on them at home and ask questions if you need help.
  - You are required to **complete all the exercises**

- For the exam we ask you to:
  - **pick one extended exercise**
  - **complete it** and prepare a **written report** explaining all the details about implementation and results
  - send us the report in advance

- The exam will be **oral** and will start from the discussion of the report

- People belonging to the same group must pick different exercises for the written report

# Course organization

## The course is divided into few main chapters

1. Introduction / refresher  (today)
2. 1D signals
3. From 1D to 2D: images
4. Analysis of telescope data
5. Remote sensing

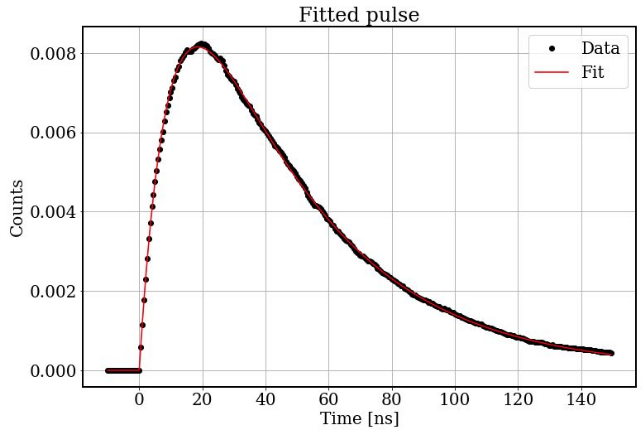
- **All topics are interconnected!**

## Introduction / refresher [today]

- Intro to python
  - **Numpy, matplotlib, scipy**
- Reminder of simple notions of statistics
- How to fit a function to a dataset

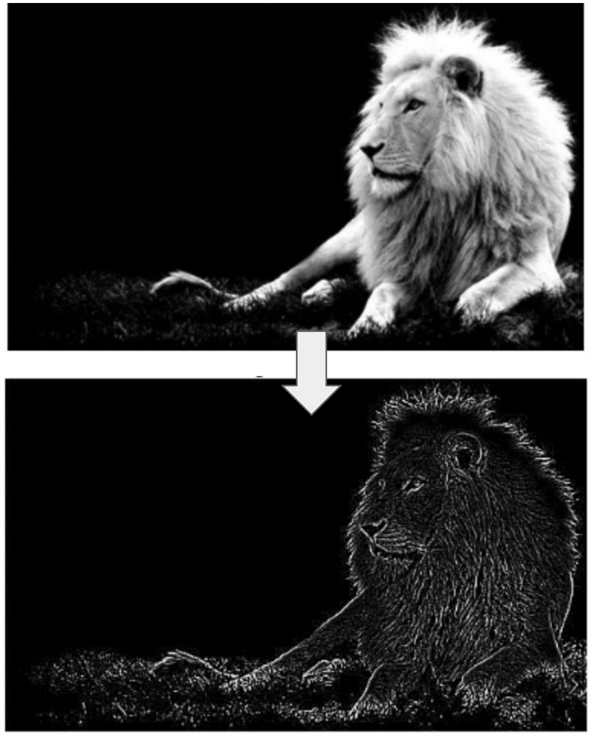
# 1D signals

- Signals from Silicon Photomultipliers (**SiPMs**)
- Generation of a pulse library
  - The **“hit or miss” monte carlo method**
  - Simulate the impact of noise
- Write a **denoising DNN**
  - Discussion of the approach
  - Build a DNN to remove the noise from the generated samples



## From 1D to 2D: images

- Intro to **OpenCV**
  - Code and test basic filters for the manipulation of the images
- Basic operations with images
  - Dynamic range
  - Picture equalization
  - Convolution of images
  - etc…

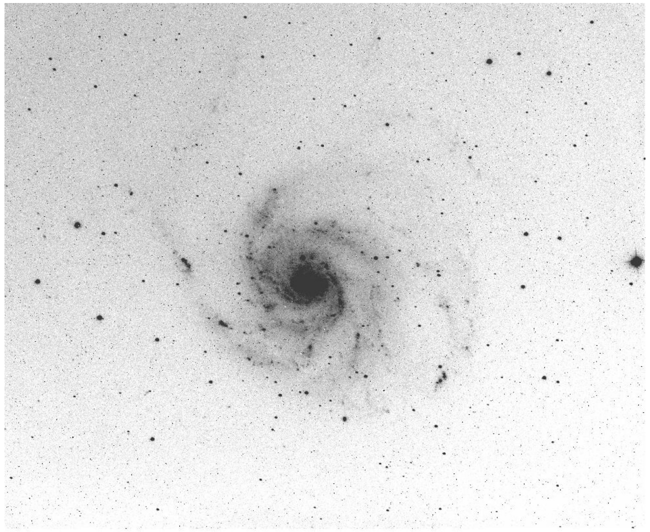


## Telescope

- Experiments with images and spectra collected via the **Bicocca Optical Telescope** (on top of U9 building)

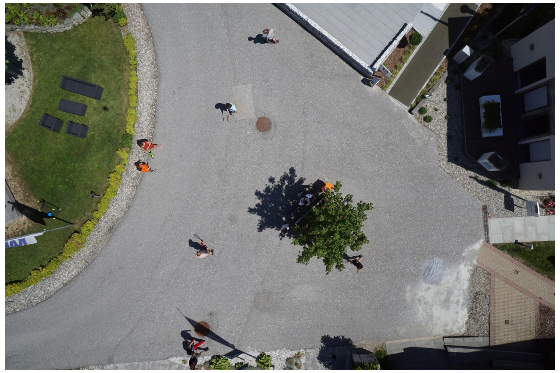
- Telescope images:
    - Characterization of the detector properties, calibrations and application to scientific images
    - Segmentation and noise properties of the images
    - **CNN exercise for galaxy classification**

- Telescope spectra:
    - Understanding optical spectra, calibrations and application to stellar spectra (Black-Body emission)
    - **CNN exercise for the extraction of stellar temperatures from spectra**



# Remote sensing and segmentation

- Experiments with images collected via remote sensing
- **Drone images:**
    - Segmentation on terrain images acquired using a drone (close distance)
    - UNet architecture
- **Satellite images:**
    - Measure physical quantities on images from satellites



# Intro to Python - 1

## See notebooks on Colab

[Intro to Python]

[Intro to numpy]

# Exercise:

- Generate 1k events gaussian distributed
- Write functions to estimate the mean and the σ of the dataset

## The variance

- To measure the dispersion of the dataset, we could compute the total difference from the arithmetic mean (sum of the deviations). However this is zero by definition:

- Instead we use the **variance** defined as the **average squared sum of the deviations:**

$$s^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$

## How to compute the variance

$$
\begin{aligned}
N s^2 &= \sum_{i=1}^{N} (x_i - \bar{x})^2 \\
&= \sum_{i=1}^{N} ({x_i}^2 + \bar{x}^2 - 2\,\bar{x}\,x_i) \\
&= \sum_{i=1}^{N} {x_i}^2 + N\bar{x}^2 - 2\,\bar{x} \sum_{i=1}^{N} x_i \\
&= \sum_{i=1}^{N} {x_i}^2 + N\bar{x}^2 - 2N\bar{x}^2 \\
&= \sum_{i=1}^{N} {x_i}^2 - N\bar{x}^2
\end{aligned}
$$

$$
\boxed{s^2 = \frac{1}{N} \sum_{i=1}^{N} {x_i}^2 - \bar{x}^2}
$$

# Intro to Python - 2

See notebooks on Colab

[Intro to Matplotlib]
[Intro to SciPy]

# Exercise:

- Estimation of parameters

## What question do I want to answer?

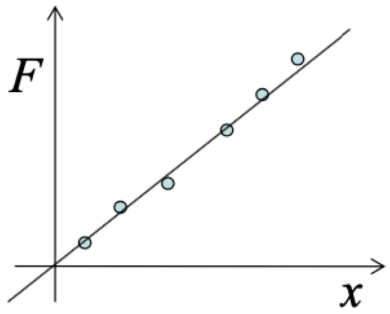
- Let’s consider two physical quantities (x, y)
- Let’s assume to know the law of physics that relates the two quantities $y = f(x; \boldsymbol{\alpha})$, but we don’t know the numerical value of the parameters ($\boldsymbol{\alpha}$)

**What curve best describes the data?**

- We want to **fit the function to our set of measures**

## Example 1

- We want to study the elastic properties of a spring
  - We fix one side of the spring and apply some tension on the other side
  - We measure how much the spring stretches as a function of the applied force
  - The Hool law predicts that the stretch is proportional to the force: **F = kx** (with $x = l - l_0$)

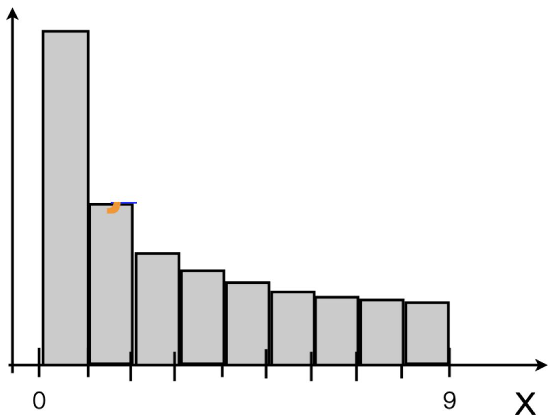



- Question: **what is the elastic constant of the spring?**

## Example 2

- We want to characterize a radioactive source
  - We measure the source activity with a Geiger counter
  - The phenomenon is well described by an exponential law:

$$P(t) = \frac{1}{\tau} e^{-t/\tau}$$




- Question: **what is the value of the $\tau$ constant?**

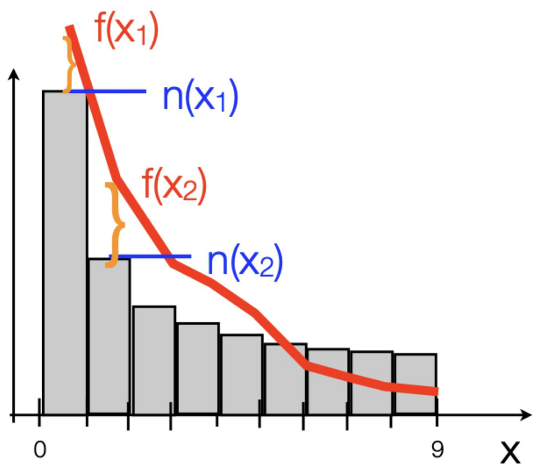
## Approach: minimize the weighted distance between data and function

- When considering the data collected in an histogram we can define the distance as:

$$\chi^2 \equiv \sum_{\text{all bins}} \frac{\left(n_{\text{meas}}(x_i) - f(x_i)\right)^2}{\sigma_i^2}$$

- The error at the denominator acts as the weight

**Minimization of the Chi2**



# Exercise: Radioactive decay source

- Generate 1k events distributed according to a **falling exponential** with
  - Tau decay constant = 1.5 ns
  - X range = [0, 4] ns
  - N bins = 30

- **Draw the histogram** of the generated events
- **Draw the PDF** used to generate the events

- Define the fitting function
- Define the distance between the histo and the function (**Chi2**)
  - What is the error in each bin?
- **Find the best parameter(s)** that describe the data
  - Perform a parameter scan and minimize the Chi2
- Compare with the result from `optimize.curve_fit`

## Reminder: the poissonian distribution

- It describes **events in a continuum**
    - Random rare events that happen at a well defined rate
- Look for the probability to observe ***r*** events in a given time interval. $\boldsymbol{\lambda}$ is the event rate.
- Example: thunderstorm
    - How many lightnings do we see in a given time interval?

$$P(r;\lambda) = e^{-\lambda}\,\frac{\lambda^r}{r!}$$

Lightning striking the Eiffel Tower, June 3, 1902, at 9:20 P.M. This is one of the earliest photographs of lightning in an urban setting In:"Thunder and Lightning", Camille Flammarion, translated by Walter Mostyn Published in 1906.

## Reminder: properties of the poissonian distribution

- It is described by a **single parameter $\lambda$**

| Mean | Standard deviation |
| --- | --- |
| $\lambda$ | $\sqrt{\lambda}$ |