# Curve Clustering Project Documentation

# Overview

This documentation will explain the logic and file structure of the project. Overall, the input of a project is an matrix where is the number of sample *types* (for instance, each voltage applied in an I(V)) measurement and is the number of samples taken (for instance, if we have 1000 I(V) curves measured).

We will use the fact that Python is object oriented, therefore we will have an *object* which will be modified. An *object* in Python can have *attributes* and *methods*. This will be clearer later. An object in python is realized calling a new *class*.

Thus, the first thing we will do is to define our object.

# Main class: data\_2\_cluster

## \_\_init\_\_\_

This is the first thing that happens when you create an object. To create an object, you need to specify a folder where all the samples reside. Then the object is created with 4 new attributes.

*Input:* the source folder

*Result:* 4 new attributes:

* data\_2\_cluster.X, data\_2\_cluster.X\_norm: Two matrices where is the number of samples *types* and is the number of samples taken.
* data\_2\_cluster.X\_df, data\_2\_cluster.X\_norm\_df: Same as above but the shape of pandas dataframe.

The normalization is done according to z-score normalization:

– the mean of the measurement , – the standard deviation.

## Data\_2\_cluster.plot()

This is the plotting method of the object, for plotting its’ attributes.

*Input:* The user is prompted to enter what he chooses to plot

*Output:* The data is plotted according to the user selections.

## Pca(self, shiftData=False)

This method calculates the principal components of the data.

Usually, the mean of each measurement *type* is calculated (the mean of every *row* in the data matrix). Then, the mean is subtracted. Then the covariance matrix is calculated according to:

However, this means that the original I(V) curves are being distorted, or, in other words, we treat each voltage measurement to be independent to one another. Since we wish to avoid that, we will use a different matrix, the correlation matrix:

Another advantage that is normalized in the range [-1,1] so we can use the unnormalized data with it.

**Note:** if there is a row of zeroes in the data matrix, than the covariance matrix is calculated:

After the matrix is calculated, it is diagonalized and the values are being sorted according to: <https://arxiv.org/abs/1404.1100>.

*Input:* the user is prompted to choose which data to perform PCA upon. It is also possible to shift the data to zero means by choose shiftData=True.

*Output:* 4 attributes are added to the object: the (correlation) covariance matrix, the principal values, the principal component in the data space and the data vector in PC space, all with names according to the user selection.

## 2.4 create\_histogram(norm=True)

This function calculates the histogram of each measurement (column in the data matrix). Proper binning is an issue in curves with plateau (or energy gap), since most of the data is located in a certain position and increasing too much the bins bumber results in very long computation time – a different approach was chosen.

Kernel Density Estimation create a histogram by place a block exactly on the point it represents and sum the total height at each location. Where how many nearest neighbors to choose from is chosen by defining the width and kernel function. More on the subject can be found here: <https://scikit-learn.org/stable/modules/density>

Here we chose gaussian kernel and width which 1/150 of the sample region.

Resources:

<https://jakevdp.github.io/PythonDataScienceHandbook/05.13-kernel-density-estimation.html>

<https://chartio.com/learn/charts/histogram-complete-guide/>

<https://en.wikipedia.org/wiki/Histogram#Number_of_bins_and_width>

## 

מה לעשות הלאה:

1. לעשות פלוט יפה יותר
2. לכתוב PCA
3. לסיים דוקומנטציה
4. להכין מצגת
5. את השרטוט של ה-PCA לכתוב בצורה יותר קומפקטית