Information Technology Course Module Computational Intelligence
by Dr. Andreas Pech

Automated Robust First Peak Detection in a Time Signal using Computational Intelligence

MD FATHIR AHMED SHISHIR
1344477
md.shishir@stud.fra-uas.de

KHAN MUSHFIQUR RAHMAN
1347995
khan.rahman@stud.fra-uas.de

AKASH SAHA
1345829
akash.saha@stud.fra-uas.de

MD MARUF HOSSAIN
1390272
MARUF:HOSSAIN@stud.fra-uas.de

Abstract*— *Many computational problems in engineering and data science fields rely on the ability to detect and analyze peaks in a time-series noisy periodic signal which is useful in a variety of applications. Identifying peaks can be extremely beneficial to simply comprehend sudden increases or decreases in any kind of data. In this paper, we propose our deep research using two approaches; firstly, using signal processing, we find or detect the first peak of a signal with the help of libraries. Secondly, by using Machine Learning (ML) approaches such as Linear Regression and Support Vector Machine (SVM) to train, predict and detect the peaks of the signal. The effectiveness and comparison of these two algorithms are shown in our research.

Here is our Project Github Repository: Link

Keywords – peak detection, local maxima and minima, Machine Learning, Linear Regression, Support Vector Machine

I. INTRODUCTION

Peak detection in signals is a crucial task in many signal processing applications. The peaks of any data can provide valuable information to society, such as sudden increments/decrements in stock markets, heart monitoring machines, traffic data, fuel prices, and many more. While identifying peaks in a multivariable time series is simple, it is necessary to fully implement the concept of a peak in order to prevent subjectivity and to create algorithms that can identify peaks in any given time series. Presently, the existing peak detection algorithms dissatisfy the need to work effortlessly, which promotes the development of a simple, and useful peak detection algorithm for noisy signal analysis.

The broad scope of applications of any algorithms can be maintained by some steps such as detecting all local maxima points, applying intelligence techniques to detect the real peaks manually, and experiencing a peak detection efficiency that is fairly robust against high and low-frequency noise [1].

Detecting peaks in a signal and measuring their positions, threshold, altitudes, widths, or locations is a necessary function in data processing applications. One method is to take advantage of the fact that the first derivative of a peak has a zero-crossing at the peak maximum. One way to detect the first peak of a time-series signal is to include the scientific computation library functions such as SciPy, which means Scientific Python, which comes under NumPy.

Many different methods have been developed up to this point, such as Hilbert transform [2], Kalman filtering [3], wavelet transforms [4], a combination of Hilbert, adaptive thresholding, and wavelet transforms [5], non-linear filtering [6-9], K-Means Clustering [10], Gabor filtering [11], Gaussian second derivative filtering [12], histogram or cumulative distribution function [13], linear prediction analysis [14]. Besides all these signal prediction algorithms, there are two more methods that work great with noise filtering and detecting the first peak using the ML approach. They are Linear regression and Support Vector Machine. In our model, we are using these methods to filter the noise so that it can predict the first peak of a given time signal.

In this paper, we are going to discuss our workflow of the model, where we started with creating our own algorithm to detect the

first peak of the given test data files using signal processing applications. After that, we investigated another approach using ML to determine an automated, robust technique to detect the peak signal, and using scientific computational libraries, we implemented and detected the first peaks in both test data files. In Section II, we will discuss the implementation of the project where we elaborated the workflow of our project for peak detection in the signal formations, then steps to set up the environment, GUI (Graphical User Interface), and packages that need to be installed to work on the project. Section III consists of deep information about the implementation of the approaches that we acquired for data pre-processing on both signal processing and ML (Machine Learning) algorithms to implement in our model. Along with that, the results and outputs are briefly discussed in this section. Finally, we conclude our study with a deep understanding of our work on peak detection of noisy signals.

I. PROJECT IMPLEMENTATION

A. *Background Studies*

The ability to identify and indeed work with signal peaks is critical in a variety of fields ranging from engineering to economics. The peaks define or determine the maxima and minima points of any functional values. This is an important implementation to realize the position and values of any statistical values so that it can be easily predictable about any information provided.

A peak or local maximum is defined in this function as any sample with two main neighbors with smaller amplitudes. When there are multiple samples of equivalent amplitude width, the index of the middle sample is returned. Peak locations may differ in noisy signals even though noise can change the position of local maxima. Consider smoothing the signal before searching for peaks in those cases, or employ other peak finding and fitting methods [15].

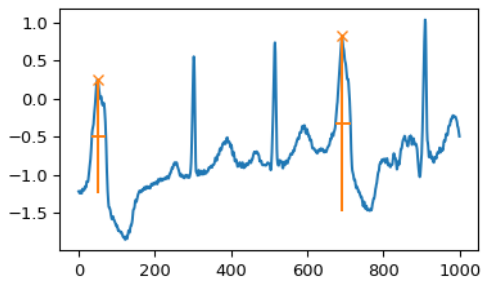
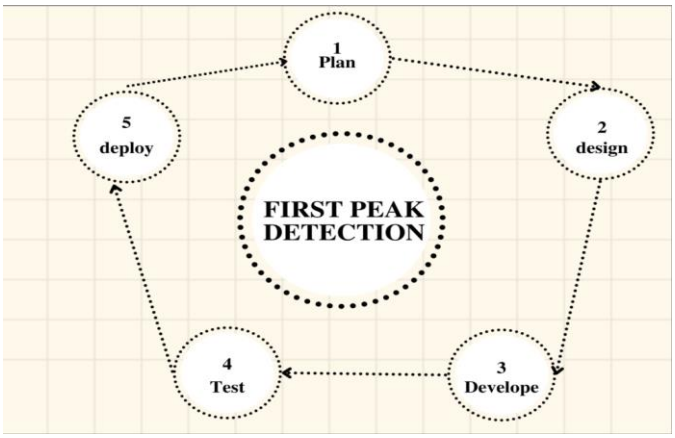


Fig. 1. Finding Peak using SciPy function [15].

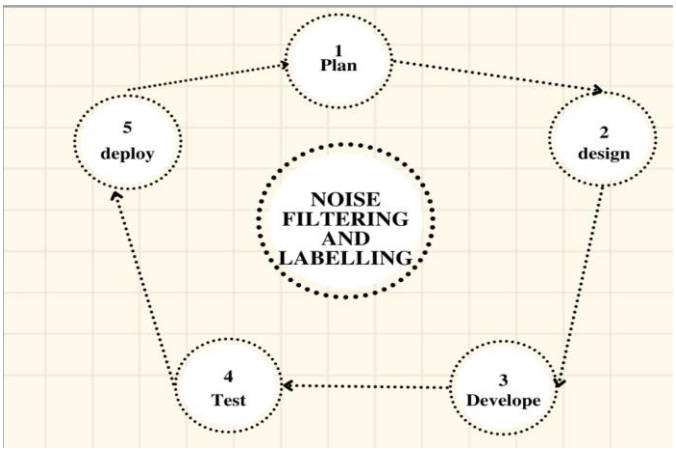
Beginning with the provided test dataset, the first thing we do is to create a function which can anticipate the peaks. For that process, we did our research in some steps which are described in detail as below.

B. *Software Engineering Workflow*

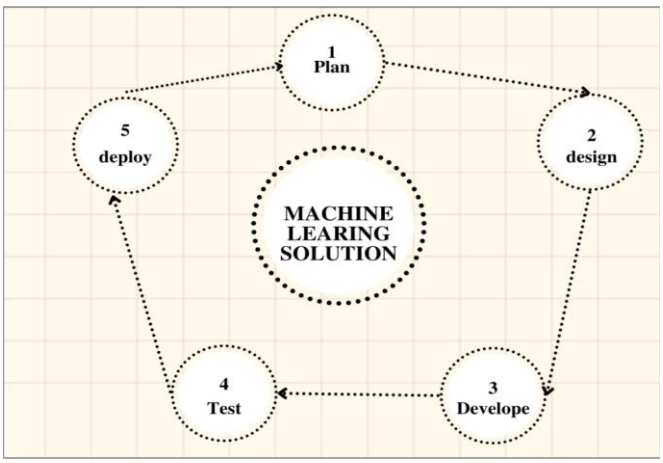
Here, we tried to work with an iterative and agile software development process. It happened in three cycles. First, for signal processing for first peak detection.



Second, for noise filtering and labeling.



Third, for machine learning approaches.



C. *Environment Setup*

To start with our experiment, the foremost thing is to build and run the project on our machine. We set up the environment. To build our project we choose the Python scripting language to create an

algorithm that can generate the expected output. Python's extensive technology enables it to provide a diverse set of libraries for computational intelligence. Python is compatible with a wide range of programming paradigms, including object-oriented and functional programming.

The machine must first have Python 3.8 or a newer version installed. The Anaconda environment or even PyCharm community edition should then be installed on the system. This console simplifies package management and deployment.

D. User Interface

Tkinter, or Tk, is a GUI (graphical user interface) toolkit that is installed and used to create graphical interaction. Tk is bound to Python. Running a specific command from the prompt could perhaps showcase a simple Tk interface, indicating that TKinter is properly installed on your machine [16]. From Tkinter, we imported "file dialog" modules, which can easily allow users to specify a file that needs to be opened or uploaded.

```
python -m tkinter
```

E. Installed Packages or Libraries

When the previous setup is all done, we can start installing the individual packages to run our project. The mentioned packages below can be directly installed by running those commands. As AI is more complicated to work with, these packages are more developed algorithms and flexible to work with. We have to implement packages like Matplotlib (v 3.6.0), NumPy (v 1.23.3), openpyxl (v 3.0.10), Pandas (v 1.5.0), SciPy (v 1.9.1), and Tkinter to work with our model so that it can generate robust signal peaks with noise filtration using an artificial intelligence approach.

Libraries like matplotlib are used for plotting the signal in a graphical structure, where we are importing the function "find_peaks", so that it can find the first local maxima of the provided data. Secondly, the pandas package is used to convert the provided test datasheets into readable form and can be accessed in excel format after data pre-processing. Scipy package is being installed for signal processing applications.

For our ML approach, we are using the Scikit-learn library. It showcases classification, regression, and clustering algorithms, such as for our work, we tried our experiment with linear regression and SVM (Support Vector Machine) models for training and testing the data.

```
from scipy.signal import find_peaks
from matplotlib import pyplot as plt
import numpy as np
import tkinter as tk
import pandas as pd
from tkinter import filedialog
```

F. Python IDE

Python scripts are written using an IDE (Integrated Development Environment). The python script for this project was written in a Pycharm tool. Also, under Anaconda, we can install Pycharm. Pycharm is an extremely simple to use web-based or machine application tool. Once installed, run the jupyter notebook by typing jupyter notebook at the anaconda command prompt. This IDE offers a variety of critical Python developer tools that are deeply intertwined to offer a pleasant environment for efficient Python, web, and data science research.

II. IMPLEMENTATION

A. Signal Processing Approach

1. Pre-Processing

It is necessary to filter for noise removal as data was collected in a noisy environment. To do that, we tried two different processes. The first one is with "Scipy.signal.butter()" and "Scipy.signal.filtfilt()" functions. We ran the latter function two times.

```
b, a = butter(3, 0.1, 'highpass')
filtered2 = filtfilt(b, a, dev_y)
filtered3 = filtfilt(b, a, filtered2)
```

Another approach that we tried is with "numpy.hannig()" which created a **Hanning Window** whose size was set to 50, and then used "numpy.convolve()" on the window and the signal. It creates a noise filter. Later, for simplicity, we chose to work with only "Scipy.signal.filtfilt()".

2. Peak Detection

To find peaks, we chose to use the "scipy.signal.find_peaks()" function, which finds all the peaks in a signal. But we only need the first peak. Also, the definition of "peak" in this function is rather different from our problem. So, to get that right, we added a small processing of these peaks to find the first peak.

```
indexes, _ = find_peaks(filtered3,
distance=1000)
      for vals in range(len(indexes)):
         peaks.append(filtered3[indexes[vals]])
         print(peaks)

      mean_indexes = np.average(peaks)
      print(mean_indexes)

      for ind in range(0, len(indexes)-1):
         if (ind == 0) & (filtered3[indexes[ind]] >
filtered3[indexes[ind+1]]):
            first_peak = filtered3[indexes[ind]]
            location = indexes[ind]
         elif (ind > 0) & (filtered3[indexes[ind]] >
mean_indexes) & \
               (filtered3[indexes[ind]] >
filtered3[indexes[ind+1]]) &
(filtered3[indexes[ind]] > filtered3[indexes[ind-
1]]):
            first_peak = filtered3[indexes[ind]]
```

```
        location = indexes[ind]
    print(first_peak)
    print(location)
```

What we did is, as the "find_peaks" function returns the indexes, we used them to generate an array of values. Then we took an average of that. After that, all the peaks that are above average and greater than the previous and next peaks are chosen. These are the desired peaks for our solution. And as we needed the first peak, we chose the value as **"First Peak".**

B. Machine Learning Approach

1. *Pre-Processing*

As given, data sheets are not processed for machine learning, and we have to label the data to do the work. After marking the desired peak, we added this peak value as a "**Class**" label and added this to the datasheet. We took the signal (X) as data and classified it as the output (y).

```
for r in range(row):
   for c in range(col):
      val = data.iloc[r, c]
      val = val.astype(float)
      new_list[r][c] = val

new_dataframe = pd.DataFrame(new_list)

new_dataframe.to_excel("Wand_000_changed.x
lsx")
```

1. *Predicting Peaks*

We used the popular machine learning library for Python, "Scikit_learn", for data training and processing. We had to get continuous peak values, so we used regression algorithms to train and test. Specifically, we applied

"Sklearn.linearregression" and "sklearn.SVM". After comparing the error score, we chose to work with "SVR" from "sklearn.SVM". We also used "GridSearchCV()" to look for the best parameter option.

```
y_train = train.Class  # if y is
only one column
train.drop(['Class'], axis=1,
inplace=True)
X_train = train

y_test = test.Class
test.drop(['Class'], axis=1,
inplace=True)
X_test= test

DT = SVR()
cross_val =
GridSearchCV(estimator=DT,
```

```
param_grid={'max_iter':[-1, 100,
500, 750, 1000]},
                              cv=4)
cross_val.fit(X_train, y_train)
pred = cross_val.predict(X_test)

print("The first five prediction
{}".format(pred[525:]))
print("The real first five labels
{}".format(y_test[525:]))

print(mean_absolute_error(y_test,
pred))
```

Results

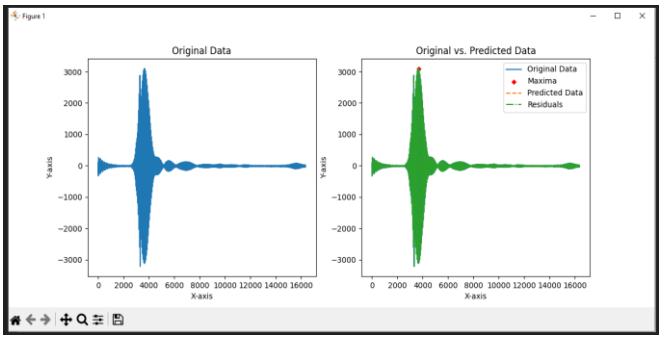
1.

```
The last five prediction
[3508.73619569 3479.73645961 3495.67847291 3505.65778555 3482.81882914]
The real last five labels
166    4948
365    2727
176    3614
179    4936
180    3125
Name: Class, dtype: int64
918.2239334974915
##################################################
```

Generic signal processing:

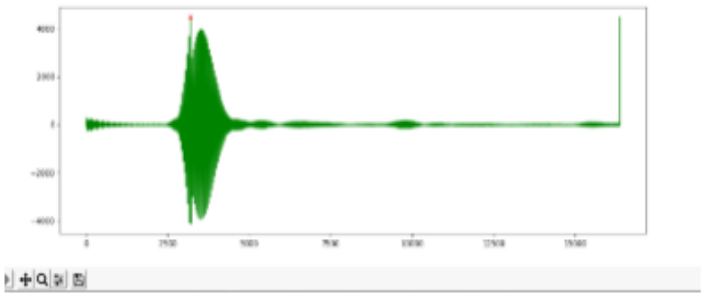
From the datasheet, we get 50 signals. Before noise filtering, it looked like the below figure.

After noise filtering, the output signals are generated as shown below.

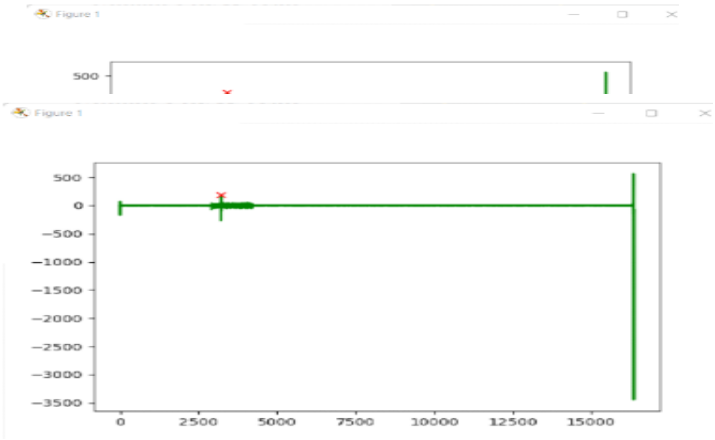


Here, the red dot on the signal is the first calculated peak. For test data two, these are like,

Before noise filtering(signal from the second data sheet)-



After noise filtering-



The green line at the end is for added 0.00. It is added while testing with “SVR()”.

2. Machine learning solution:

In machine learning solution, when data is trained with “sklearn.Linear_Model.linearRegression()” it shows Mean_absolute_error 1675.0146

But when we used model SVR() from Support Vector Machine it showed “MEAN-ABSOLUTE_ERROR - 918.2239”

	mean_fit_time	std_fit_time	...	std_test_score	rank_test_score
0	3.645571	0.146199	...	0.016387	2
1	2.363227	0.025431	...	0.011486	1
2	3.690223	0.095773	...	0.016387	2
3	3.750158	0.063970	...	0.016387	2
4	3.979616	0.154393	...	0.016387	2

Also the grid search cross validation shows when the output is best.

Problems encountered and solutions:

Initially, we attempted to develop a custom algorithm tailored to our specific needs. Unfortunately, this approach did not yield the desired results. Consequently, we transitioned to utilizing Python's default libraries.During the initial stages of noise filtering, we encountered issues due to coding errors, including the inadvertent selection of an incorrect function. To address these challenges, we introduced a Hamming window, followed by attempting cross-correlation with the signal. However, this approach did not produce the expected results. To overcome this, we turned to the NumPy convolve() function for signal convolution. Later, we opted for simplicity by adopting the scipy.filtfilt() function.

When venturing into machine learning, we encountered a significant challenge. The absence of labeled data sheets posed difficulties in our initial attempts. Ultimately, we resolved this issue by manually identifying the desired outcomes and obtaining a second labeled datasheet, enabling us to pursue a more intelligent approach.

Discussion

In this paper, we have provided a formal explanation of the concept of finding the first peak in a noisy periodic signal using both signal processing and machine learning approaches. We have presented two approaches that have some limitations for any simulated or real-time data.

In the signal processing approach, first, we did the data pre-processing by using two functions for noise filtering. After that, we used another package and function to examine the first and exact peak that returns the index positions of the local maxima of a periodic signal. Although we detected the first peak of the noisy signal, it was not robust. To get robustness in the peak formation, we studied that a lot of conditions need to be added for each signal variation; this can reduce the performance of high and low-frequency signals, and this is considered a limitation by using signal processing methods.

In the Machine Learning approach, we again performed the data pre-processing to avoid noise signals, by adding the first peak value that we obtained from the signal processing approach to a specified Class label. In the next step, after preparing the labeled data, we performed training and processing, which was executed using Linear Regression and SVM algorithms. We found out that, during training the big signal datasets, the detected or predicted peak is closer to the first peak but not the exact one that was encountered in the signal processing approach.

As a result, we can conclude that by using a signal processing algorithm, the first peak is detected at the accurate position, but it is not precise with every other signal type. Nevertheless, in the ML approach, the first peak can be found in any type of data, but it can be at the factually incorrect position or nearer to the exact first peak.

REFERENCES

[1] Felix Scholkmann, Jens Boss, Martin Wolf.; "An Efficient Algorithm for Automatic Detection in Noisy Periodic and Quasi-Periodic signal.", Algorithms, 2012, 5, www.mdpi.com/journal/algorithms

[2] Benitez, D.; Gaydecki, P.A.; Zaidi, A.; Fitzpatrick, A.P. "The use of the Hilbert transform in ECG signal analysis." Comput. Biol. Med. 2001, 31, 399–406.

[3] Tzallas, A.T.; Oikonomou, V.P.; Fotiadis, D.I. "Epileptic spike detection using a Kalman filter based approach." In Proceedings of the 28th IEEE EMBS Annual International Conference, New York, NY, USA, 2006; pp. 501–504.

[4] Singh, O.; Sunkaria, R.K. "A robust R-peak detection algorithm using wavelet packets." Int. J. Comput. Appl. 2011, 36, 37–43.

[5] Rabbani, H.; Mahjoob, M.P.; Farahabadi, E.; Farahabadi, "A. R peak detection in electrocardiogram signal based on an optimal combination of wavelet transform, Hilbert transform, and adaptive thresholding." J. Med. Signals Sens. 2011, 1, 91–98.

[6] Sun, Y.; Suppappola, S.; Wrublewski, T.A. "Microcontroller-Based real-time QRS detection." Biomed. Instrum. & Technol. 1992, 26, 477–484.

[7] 24. Ferdi, Y.; Herbeuval, J.P.; Charaf, A.; Boucheham, B. "R wave detection using fractional digital differentiation." ITBM-RBM 2003, 24, 273–280.

[8] 25. Aboy, M.; McNames, J.; Thong, T.; Tsunami, D.; Ellenby, M.S.; Goldstein, B. "An automatic beat detection algorithm for pressure signals." IEEE Trans. Biomed. Eng. 2005, 52, 1662–1670.

[9] 26. Shim, B.; Min, H.; Yoon, S. "Nonlinear preprocessing method for detecting peaks from gas chromatograms." BMC Bioinf. 2009, doi:10.1186/1471-2105-10-378.

[10] Mehta, S.S.; Sheta, D.A.; Lingayat, N.S.; Chouhan, V.S. "K-Means algorithm for the detection and delineation of QRS-complexes in electrocardiogram." IRBM 2010, 31, 48–54.

[11] Nguyen, N.; Huang, H.; Oraintara, S.; Vo, "A. Peak detection in mass spectrometry by Gabor filters and envelope analysis." J. Bioinf. Comput. Biol. 2009, 7, 547–569.

[12] Fredriksson, M.J.; Petersson, P.; Axelsson, B.O.; Bylund, D. "An automatic peak finding method for LC-MS data using Gaussian second derivative filtering." J. Seperation Sci. 2009, 32, 3906–3918.

[13] Sezan, M.I. "A peak detection algorithm and its application to histogram-based image data reduction." Comput Vis. Graph. Image Proc. 1990, 49, 36–51.

[14] Lin, K.-P. "QRS feature extraction using linear prediction." IEEE Trans. Biomed. Eng. 1989, 36, 1050–1055.

[15] https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html

TEAM MEMBER	TASK PERFORMED
MD FATHIR AHMED SHISHIR	CODE: ● First Peak detection ● Machine learning(Linear Regressing)
	REPORT: ● Implementation (Signal Processing and Regression approach) ● Results & Output Section ● Conclusion & Discussion
MD MARUF HOSSAIN	CODE: ● Data labeling ● Machine learning(SVM)
	REPORT ● Abstract & Conclusion ● Project Implementation ● SE Workflow
AKASH SAHA **&** **KHAN MUSHFIQUR RAHMAN**	CODE: ● GUI Creation ● Noise filtering ● Plotting the output
	REPORT: ● Environment setup ● Problems & solution ● Presentation PPT ● Background studies