The American University in Cairo

School of Sciences and Engineering

Computer Science and Engineering

CSCE3611 - Digital Signal Processing

Project : Analyzing ECG Signals using Autocorrelation and Identifying Atrial Fibrillation

Spring 2023

Dr. Seif ELdawlatly

Nermien Elassy 900196006

Mariam ElAwadly 900191854

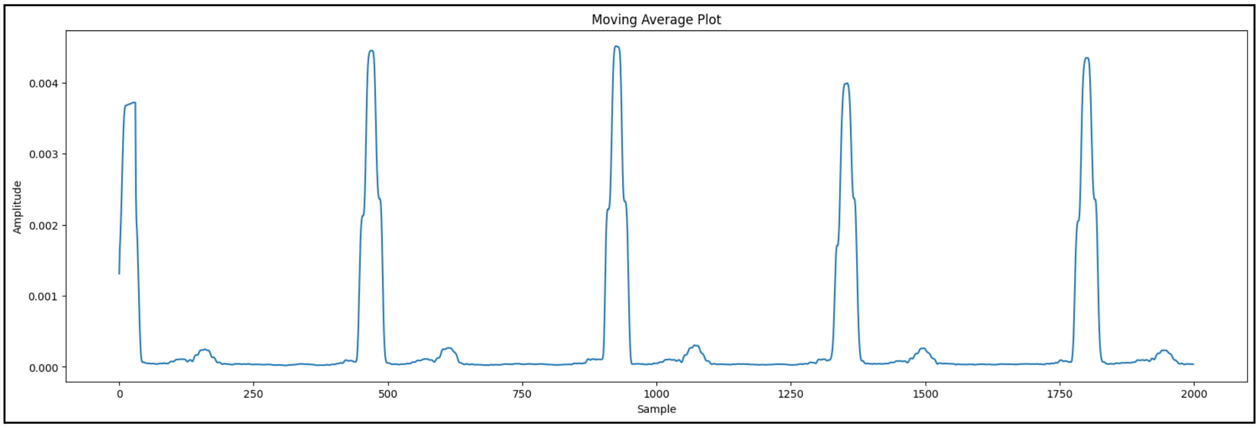
Ahmad Mohamed 900194153

Description of Approach:

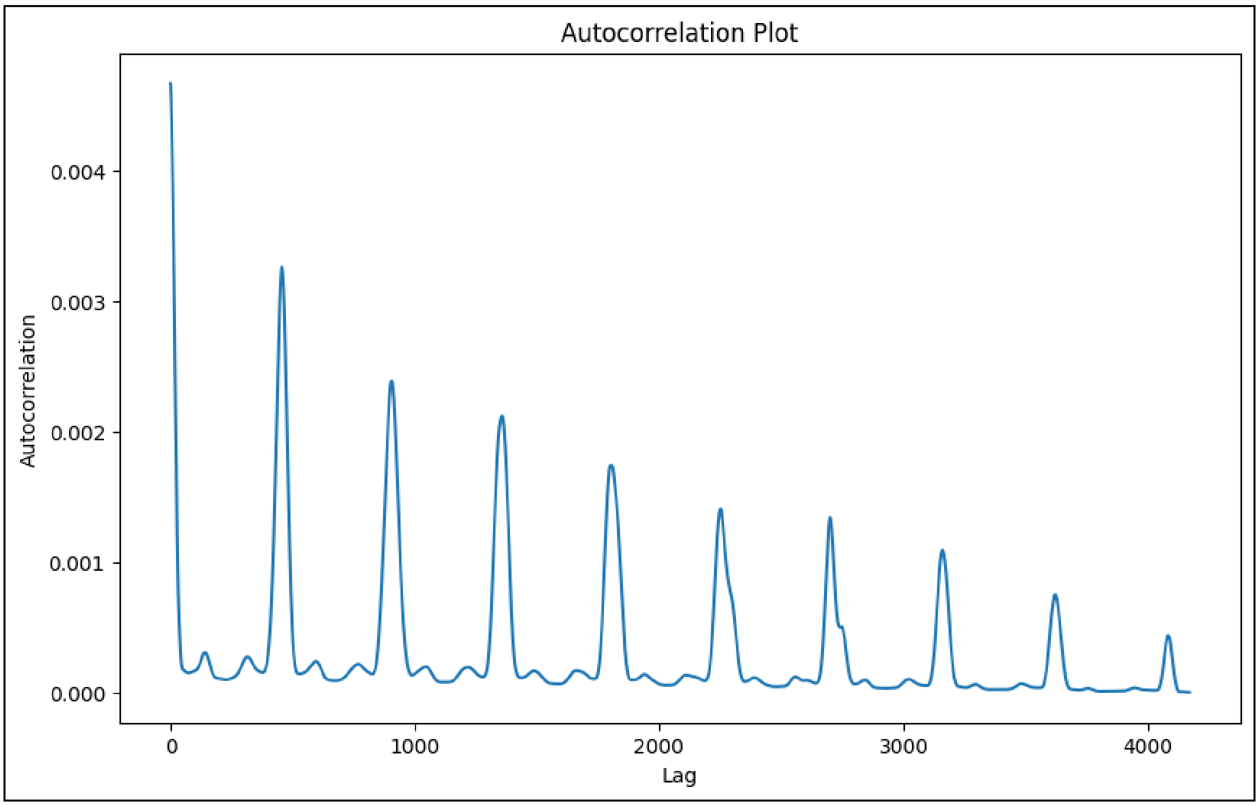
- A function **Read_originalsignal(filename)** was created to read the signal from the given text file and plot it
- A function def **NPointDiff(input_array, N)** was created to find the derivative of the ECG signal that we read from the text file using the 5-point difference equation
$$y(nT) = (\frac{1}{8}T)[-x(nT-2T) - 2x(nT-T) = 2x(nT+T) + x(nT+2T)]$$
- A function **square_the_derivative(arr)** that takes the derivative of the ECG signal and squares it using this equation
$$y(nT) = [x(nT)]^2$$
- A function **smooth_signal_squared(signal_squared)** takes the squares signal and smooth it using a moving average window of size 31 samples using this equation
$$y(nT) = \frac{1}{N}[x(nT-(N-1)T) + x(nT-(N-2)T + ... + x(nT)]$$
- A function **autocors(smoothed)** was created which takes the smoothed signal and applies Autocorrelation to it and plots it showing the lag on the x-axis
- Also a function **get_heart_rate(auto_correlated_signal, fs)** takes the autocorrelated signal and fs which is 512 HZ, and computes the heart rate 60 / (peaks[1] / fs) where peak[1] is the highest peak neglecting the peak at [0]
- Finally, a function **ECG(filename)** which combines all the aforementioned functions
- Then we run the function on the two given text files which are Data1.txt and Data2.txt to find the output stated in the next sections.
- Based on the results we obtained from the two signals, the measures we suggested that can be computed from the autocorrelation that increases as the amount of Atrial Fibrillation increases is as follows:
  We noticed that what differentiates the signal of someone who has an atrial fibrillation and someone who does not have is the peaks where we can see irregularity where more peaks an be noticed and the distance between them is very close thus we made these two functions based on the peaks as our metric
  - **compute_autocorrelation_peak_width(autocorr)**
    - In this function, we find the peaks using find_peaks function. Then, we apply np.diff function on the peaks to get the difference between each peak and the peak before it and divide it by the sampling frequency. Afterward, we get the mean of the values obtained in the previous step. This mean is the average difference between the peaks
  - **count_peaks(signal, threshold)**
    - In this function, we pass the autocorrelated signal along with a threshold we set, we set our threshold to 0.005 by observation of the graph. Then we check all the points in the signal and compare it to its neighbors. If it is higher, then we check if the value of the point is higher than the threshold. If yes, we increment the counter, named peaks, else we ignore the point and do nothing.

Output of part 1:

A. The first 2000 samples of the ECG signal after step 3:



B. A plot of the autocorrelation showing the lag on the x-axis and the autocorrelation value on the y-axis:

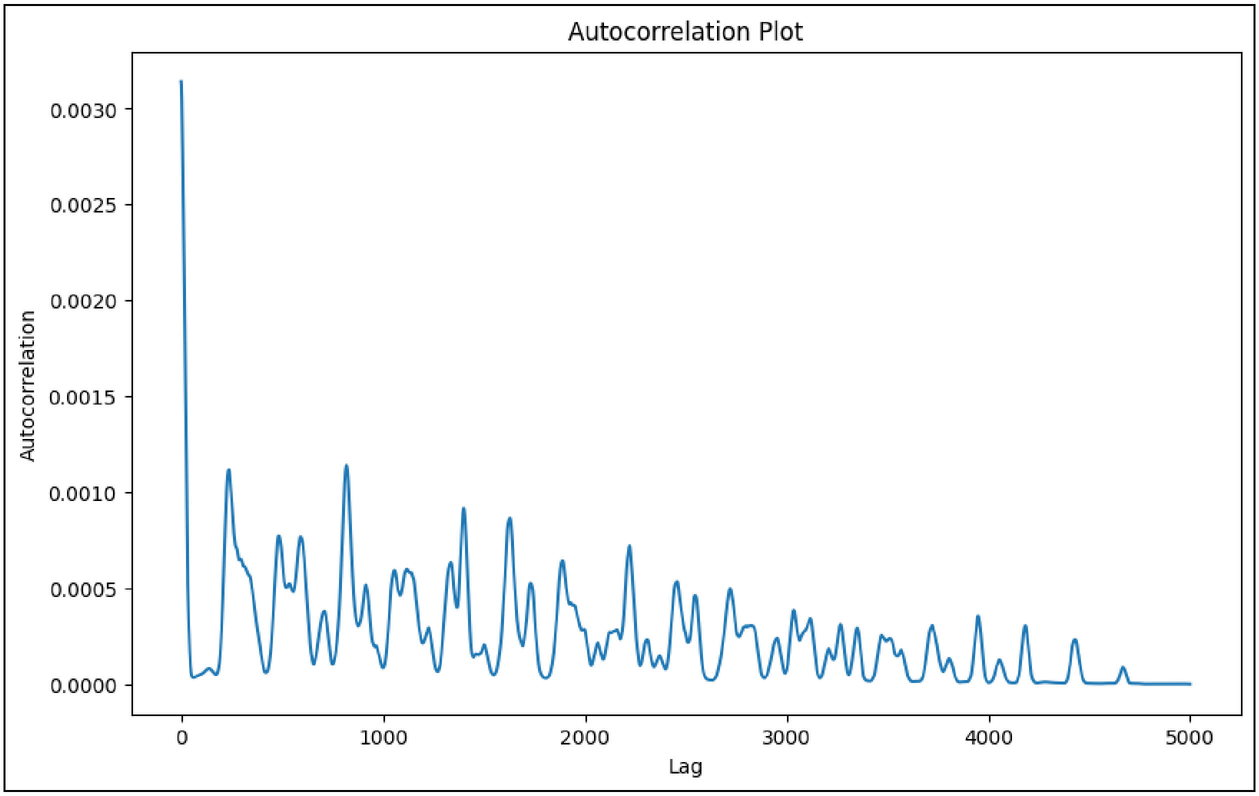


C. The Average Heart Rate is

`51.63025210084034 beats/min`

Output of part 2:

A. A plot of the autocorrelation showing the lag on the x-axis and the autocorrelation value on the y-axis:



B. The value of the suggested measure:
   a. Number of peaks
      i. Part 1: **9** peaks
      ii. Part 2: **18** peaks
   b. Average width between the peaks:
      i. Part 1: `0.22960707720588236`
      ii. Part 2: `0.15445696721311475`