The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

Project Title: Finding Heart Rate from ECG Signals and Identifying Missing Beats for Sinus Arrest Patients

This project consists of two parts as follows:

Part 1: Detecting ECG R-waves

Implement the QRS detection method given in the attached File (ECG Processing.pdf). Your function should take as inputs the ECG signal to process and the moving average window size *N* mentioned on slide 12. The function should return a vector that contains the time stamps of the R wave and a vector that contains the corresponding RR intervals (Slide 14). Apply your function to the ECG signal provided in the file "DataN.txt". The sampling rate of this ECG signal is 256 Hz. You will need to suggest a method to compute the threshold needed for detection.

Deliverables

- Your code

```
function [RWaveTimestamps, RRIntervals] = DetectQRS(file, N)
%N is the moving average window size

% The sampling rate of the ECG signal is 256 Hz
fs = 256;
T = 1/fs;
fid = fopen(file);
ECGSignal = cell2mat(textscan(fid, '%f'));
B = ECGSignal;
fclose(fid);
```

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

```
% A figure showing the first 1500 samples of the ECG signal before noise
% filtering.
figure(1);
n = 0:1:1499;
plot(n, ECGSignal(1:1500));
xlabel("n");
ylabel("Amplitude");
numberOfSamples = length(ECGSignal);

% A typical noisy ECG signal will appear along with a 50Hz (or 60 Hz)
% mains noise
% A notch filter with a filtered frequency of 50Hz can be used to remove
% the baseline 50Hz
% stop frequency of 50 Hz
stopFrequency = 50;
w0 = stopFrequency/(fs/2);
bw = w0/35;
[b, a] = iirnotch(w0, bw);
ECGSignal = filtfilt(b, a, ECGSignal);
% The ECG signal is further filtered using a Band-pass filter with a
% bandwidth of 0.1-45 Hz
% All frequency values are in Hz.
order = 6;  % Order
Fc1 = 0.1;  % First Cutoff Frequency
Fc2 = 45;   % Second Cutoff Frequency

% Construct an FDESIGN object and call its BUTTER method.
h  = fdesign.bandpass('N,F3dB1,F3dB2', order, Fc1, Fc2, fs);
Hd = design(h, 'butter');
ECGSignal = filter(Hd, ECGSignal);

% A figure showing the first 1500 samples of the ECG signal after noise
% filtering.
```

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

```
figure(2);
plot(n, ECGSignal(1:1500));
xlabel("n");
ylabel("Amplitude");

% One way to overcome baseline drifts is to differentiate the ECG signal
% since we are looking for large slopes in the signal
x = ECGSignal;
y = zeros(numberOfSamples, 1);
y(1) = (1/(8*T))*(2*x(2)+x(3));
y(2) = (1/(8*T))*(-2*x(1)+2*x(3)+x(4));
for n = 3:1:numberOfSamples-2
    y(n) = (1/(8*T))*(-x(n-2)-2*x(n-1)+2*x(n+1)+x(n+2));
end
y(numberOfSamples-1) = (1/(8*T))*(-2*x(numberOfSamples-2)-x(numberOfSamples-
3)+2*x(numberOfSamples));
y(numberOfSamples) = (1/(8*T))*(-2*x(numberOfSamples-1)-x(numberOfSamples-2));

% The next step is to square the derivative. This makes all data points
% positive and does nonlinear amplification of the output of the derivative
% emphasizing the higher frequencies (i.e., predominantly the ECG
% frequencies)
x = y;
for n = 1:1:numberOfSamples
    y(n) = (x(n))^2;
end

% The next step is to smooth the squared signal using a moving-average
% window
% y(nT) = (1/N)[x(nT-(N-1)T)+x(nT-(N-2)T)
% +...+x(nT)]
% where N is the number of samples in the width of the moving average
% window
% N should be approximately the same as the widest possible QRS complex
```

```
x = y;
y = zeros(numberOfSamples, 1);
for n = 1:1:numberOfSamples
    for i = 1:1:N
        if(n >= N)
            y(n) = y(n) + x(n-N+i);
        else
            y(n) = x(n);
        end
    end
    y(n) = y(n)/N;
end
ECGSignal = y;

% Zero padding
for i = 1:1:fs/2
    ECGSignal(i) = 0;
end
for i = numberOfSamples-fs/2:1:numberOfSamples
    ECGSignal(i) = 0;
end

% The final step is to set a threshold on the moving average output. The
% peak value above the threshold within the moving average window is
% approximately the R wave
A = zeros(fs/2, 1);
for i = fs:1:fs*2
    if(floor(ECGSignal(i)) <= 190)
        A(i) = ECGSignal(i);
    end
end
threshold = 15*std(A);
[pks, locs] = findpeaks(ECGSignal, "MinPeakHeight", threshold, "MinPeakProminence",
N);
```

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

```
RWaveTimestamps = zeros(length(pks), 1);
for i = 1:1:length(RWaveTimestamps)
    RWaveTimestamps(i) = locs(i)*T;
end

% The sequence of RR intervals - that is, all intervals between adjacent QRS
% complexes resulting from sinus node depolarizations - forms the RR
% interval time series or RR tachogram
RRIntervals = zeros(length(pks)-1, 1);
for i = 2:1:length(RWaveTimestamps)
    RRIntervals(i-1) = (RWaveTimestamps(i)-RWaveTimestamps(i-1))*10^3;
end

% A corresponding sequence of instantaneous heart rate is defined as ff_i =
% 1/RR_i
heartRate = zeros(length(RRIntervals), 1);
for i = 1:1:length(heartRate)
    heartRate(i) = 1/RRIntervals(i);
end

% A figure showing the first 1500 samples of the ECG signal with an "*"
% marking the detected R waves.
n = 0:1:1499;
figure(3);
plot(n, ECGSignal(1:1500));
hold on
plot(n(locs(1:5)), ECGSignal(locs(1:5)), "*");
xlabel("n");
ylabel("Amplitude");

% A figure showing the first 1500 samples of the ECG signal with an "*" marking the
detected
% R waves for N = 25 but without noise filtering.
figure(4);
```

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

```
plot(n, B(1:1500));
hold on
plot(n(locs(1:5)), B(locs(1:5)), "*");

% A plot of the RR intervals with Beat number on the x-axis and RR interval
% in msec on the y-axis in the case of N = 25.
figure(5);
x = 0:1:length(RRIntervals)-1;
plot(x, RRIntervals);
xlabel("Beat number");
ylabel("RR interval (msec)");
end
```

- A figure showing the first 1500 samples of the ECG signal before and after noise filtering.

The American University in Cairo

Computer Science and Engineering Department

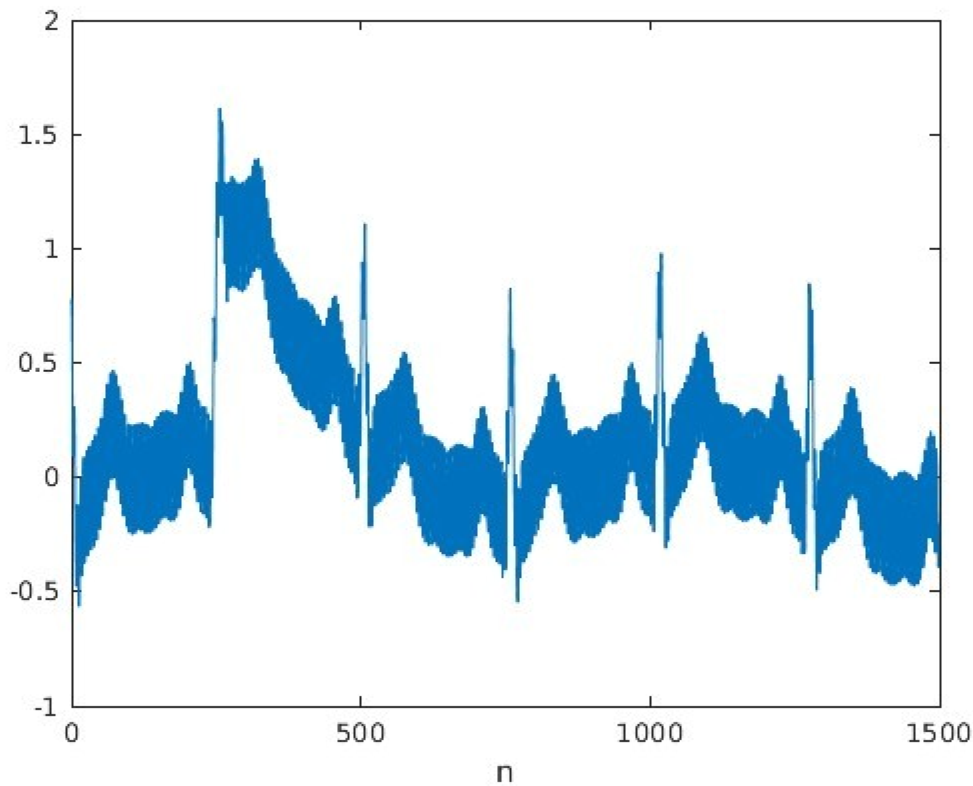
Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)



The American University in Cairo

Computer Science and Engineering Department

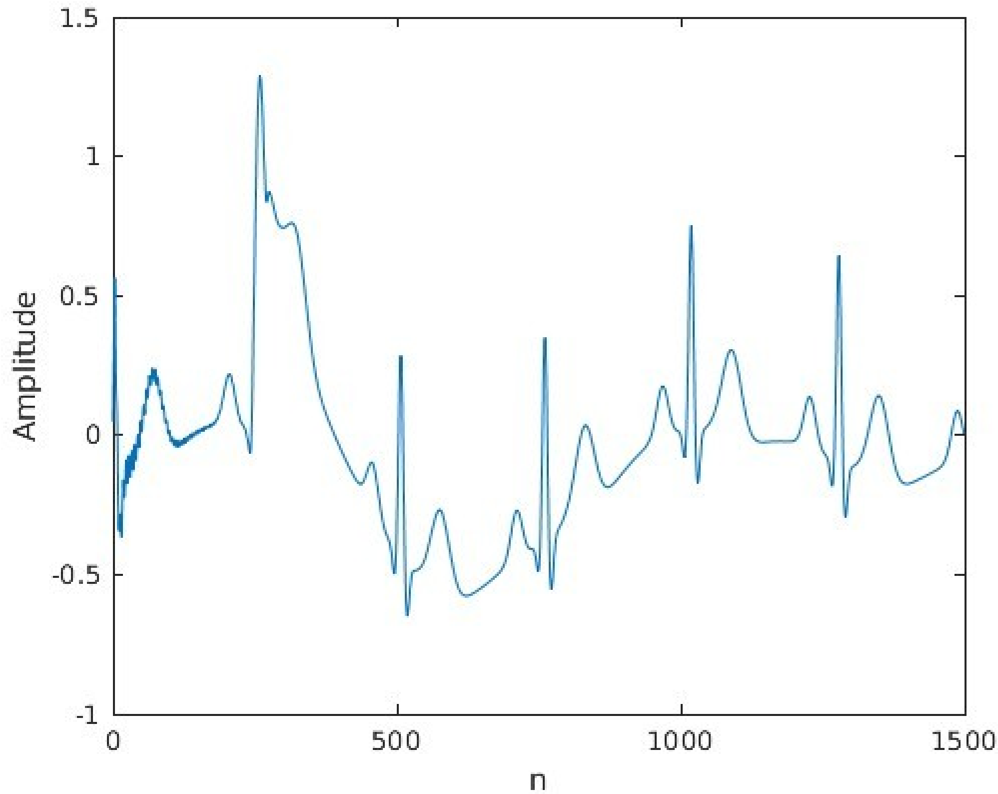
Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)



The American University in Cairo

Computer Science and Engineering Department

Fall 2022

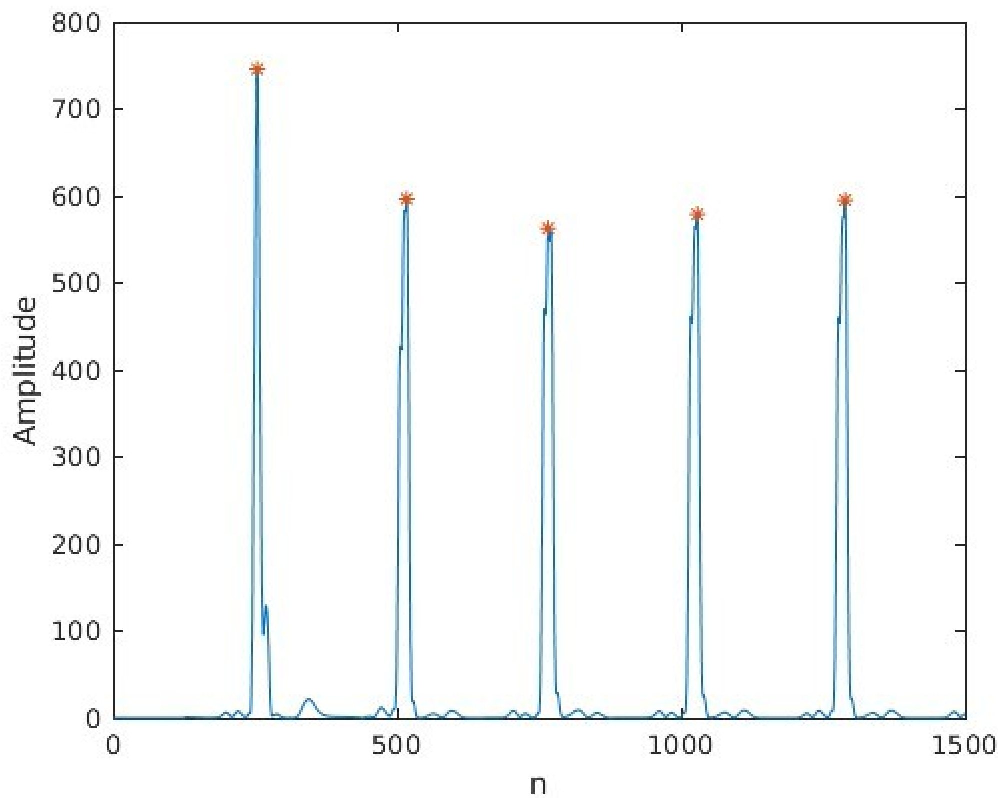
CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

- A figure showing the first 1500 samples of the ECG signal with an “*” marking the detected R waves for N = 10.



The American University in Cairo

Computer Science and Engineering Department

Fall 2022

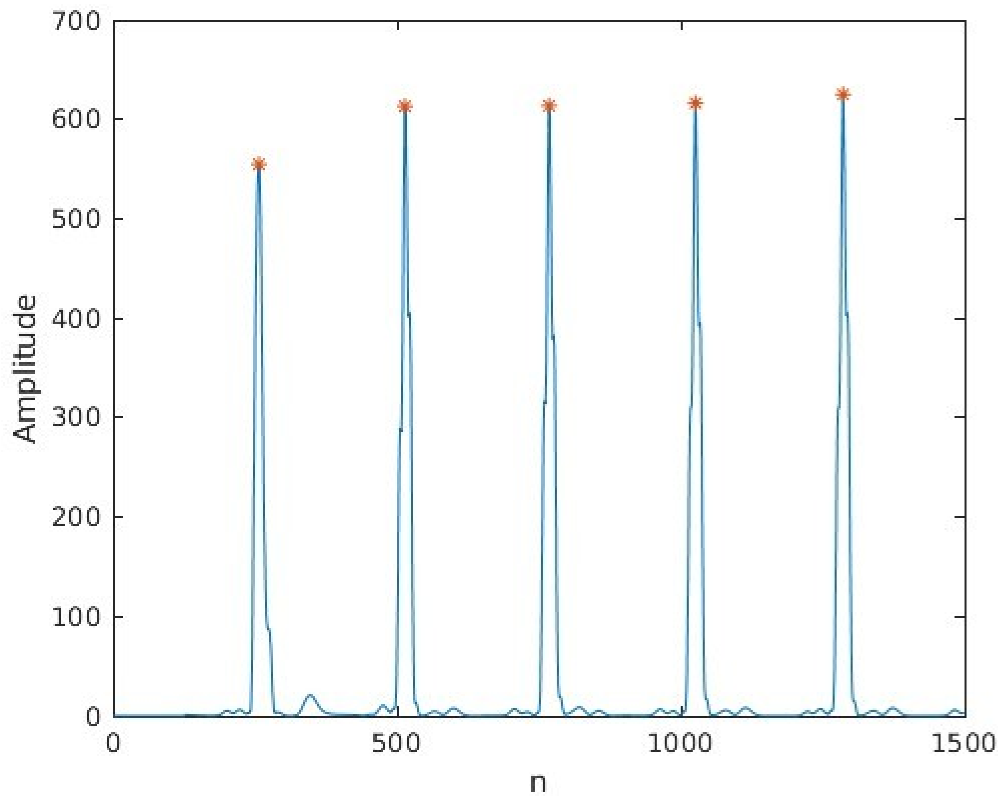
CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

- A figure showing the first 1500 samples of the ECG signal with an "*" marking the detected R waves for N = 15.



The American University in Cairo

Computer Science and Engineering Department

Fall 2022

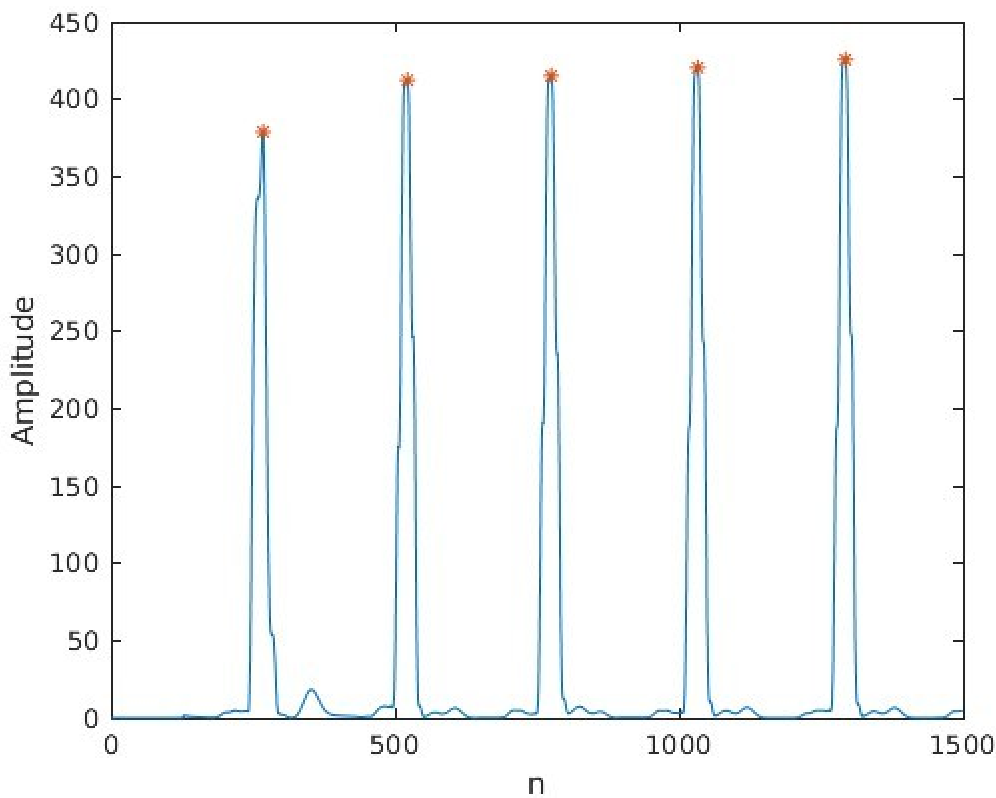
CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

- A figure showing the first 1500 samples of the ECG signal with an "*" marking the detected R waves for N = 25.



The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

- What can you conclude about the optimal setting of *N*? Explain your answer.

  We can conclude that the optimal setting of *N* is 25. The moving average window size *N* determines the how smooth the signal is. Therefore, predicting the moving average window size determines how well we can tease out the trend-cycle component. Choosing the wrong moving average window size causes pollution in the trend-cycle estimates due to the periodicity of the data. Too small moving average window sizes like 10 and 15 do not eliminate noise as effective as a moving average window size of 25.

- A figure showing the first 1500 samples of the ECG signal with an "*" marking the detected R waves for N = 25 but without noise filtering.

The American University in Cairo

Computer Science and Engineering Department

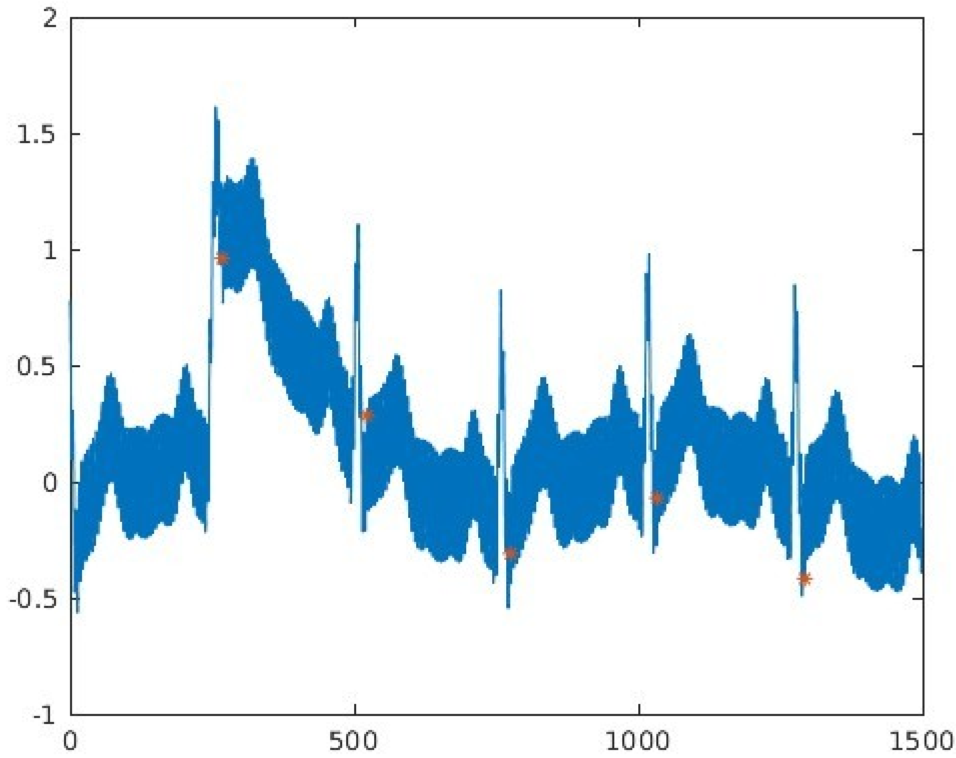
Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)



The American University in Cairo

Computer Science and Engineering Department

Fall 2022

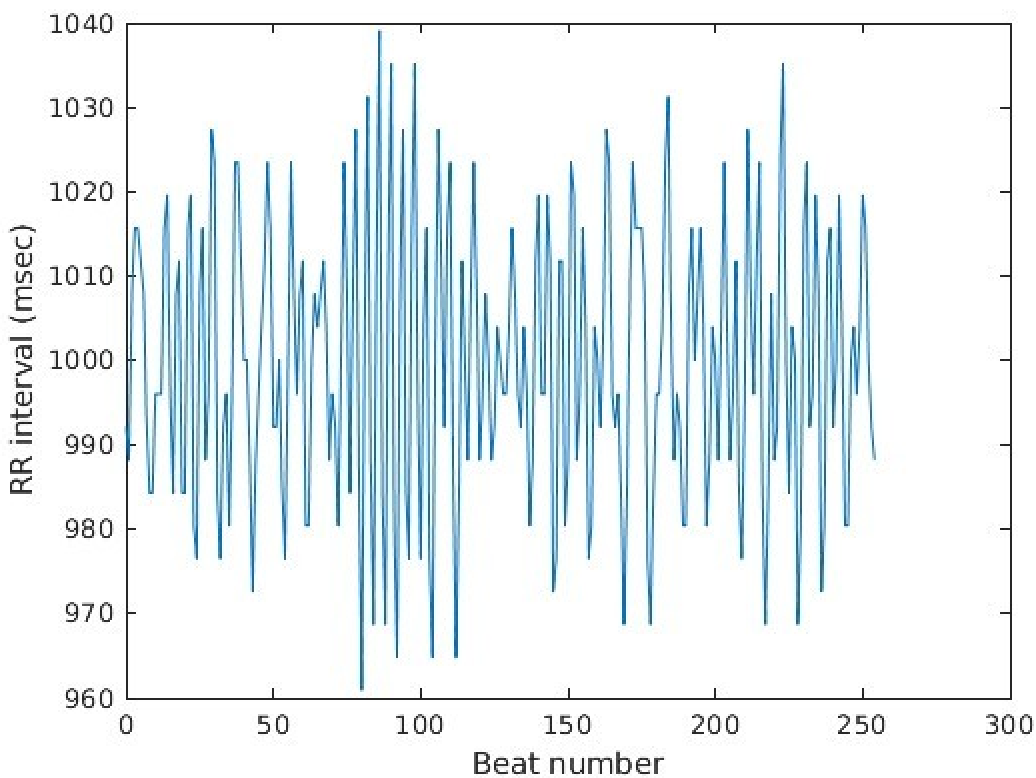
CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

- A plot of the RR intervals with Beat number on the x-axis and RR intervals on the y-axis in the case of N = 25.



The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

Description of the approach used

The function *DetectQRS* takes two formal parameters file and N. The first formal parameter, file, is the name of the file that contains the samples of the ECG signal, while the second formal parameter, N, is the moving average window size. *DetectQRS* returns *RWaveTimestamps* and *RRIntervals*. *RWaveTimestamps* is a vector that contains the sample number at which R waves are located, and *RRIntervals* is another a vector that contains the difference in the number of samples between successive R waves. The function starts by setting the sampling frequency *fs* to a value of 256. Then, it sets the sampling interval to the reciprocal of the sampling frequency. The function *fopen* is used to open the text file "DataN.txt". Then the input from the file is read into a vector called *ECGSignal*. Then the file is closed using the function *fclose*. The first 1500 sample values of *ECGSignal* are plotted on the first figure. *ECGSignal* is then passed through a notch filter. The stop frequency is represented as a variable called *stopFrequency*. *stopFrequency* is assigned a value of 50 Hz. *stopFrequency* is then normalized to be in the range between 0 to 1 by dividing it by half the sampling frequency. The bandwidth is obtained by dividing the normalized value by a quality factor of 35. second-order IIR notch filter is then used to obtain the numerator and denominator which are used in zero phase digital filtering. The output of the notch filter is then passed through the band-pass filter. The band-pass filter has a low cut-off frequency of 0.1 Hz, a high cutoff frequency of 45 Hz, and an order of 6. The first 1500 sample values of the output of the band-pass and notch filters are then plotted on the second figure. To bypass the baseline drifts the output of the band-pass and notch filters is differentiated using this formula . Then the output of the differentiation process is squared to make all values positive and to non-linearly magnify the output of differentiation and stress on the bigger frequencies. The output of the squaring process is then passed through a moving average filter that basically smooths the signal using the moving average window of size N. The signal is padded with zeros at the start and end with zeros due to the undesirable effect generated by the moving average filter. Then, the threshold is calculated based on the standard deviation of the values of the noise which lies between the first pair of successive R waves. The factor that was suitable to multiply the standard deviation by was 15. The R wave timestamps are obtained using the function *findpeaks* and multiplying the indices by the sampling interval T. The RR intervals are obtained by iterating over all the elements of *RWaveTimestamps* and using the formula . The heart rate is also calculated by reciprocating the elements of *RRIntervals*. The third figure which shows first 1500 samples of the processed ECG signal and the marked R waves is then plotted against the sample number n. The fourth figure that illustrates the first 1500 samples of the original ECG signal is then plotted against the sample number n. Finally, the fifth figure which shows the RR intervals plotted against the sample number n.

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

Part 2: Sinus Arrest Missing Beats Detection

The ECG signal of a patient with Sinus Arrest is provided in file “Data2.txt”. Implement a function that you can use to find the timestamps at which a beat should have been recorded (missing beats). The function should take as inputs the ECG signal to process and the moving average window size *N*. Your function should make use of the function you implemented in Part 1. The function should return the timestamps of missing beats. Apply your function to the ECG signal provided in the file “Data2.txt” with the best *N* obtained from Part 1. The sampling rate is 256 Hz.

Deliverables:

- Your code

```
function [missingBeatsTimestamps] = FindMissingBeats(file, N)

%file is the ECG of a patient with Sinus Arrest
%N is the moving average window size

% The sampling rate is 256 Hz.
fs = 256;
T = 1/fs;

% Your function could make use of the function you implemented in Part 1.
[RWaveTimestamps , RRIntervals] = DetectQRS(file, N);
average = sum(RRIntervals)/length(RRIntervals);
standardDeviation = std(RRIntervals);
count = sum(RRIntervals < average + standardDeviation);
total = sum(RRIntervals(RRIntervals < average + standardDeviation));
mean = (total/count)*10^(-3);
```

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

```
I = find(RRIntervals > average + standardDeviation);

numberOfMissingBeats = sum(RRIntervals > average + standardDeviation);
missingBeatsTimestamps = zeros(numberOfMissingBeats, 1);
for i = 1:1:length(missingBeatsTimestamps)
    missingBeatsTimestamps(i) = RWaveTimestamps(I(i))+mean;
end
missingBeatsTimestamps = missingBeatsTimestamps/T;
end
```

- The timestamps of missing beat(s) (the sample number at which the R wave should have been present).

  1.0e+03 *

  1.1213

  2.4233

  5.2853

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

Description of the approach used

The function *DetectQRS* is further used in detecting the missing beats. The timestamps of the signal as well as the time interval between each beat are stored in a vector. First, the standard deviation of the intervals is calculated. Then the average value of all intervals is calculated. Those two values are then used to calculate the mean of the RR intervals that are less than the standard deviation plus the average, the intervals which were greater than the standard deviation plus the average in that case represent a missing beat. To calculate the timestamp of the missing beat, we get the stamp of the previous beat and add to it the average interval we calculated in the previous step.

The American University in Cairo

Computer Science and Engineering Department

Fall 2022

CSCE 363/3611 Digital Signal Processing

Project

(Due on: December 11, 2022 at mid-night)

(Submit on Blackboard as one .zip file)

Submission:

- Your MATLAB or Python code to be submitted on Blackboard on **December 11 at mid-night**
- A report (to be submitted on Blackboard on **December 11 at mid-night**) that includes the following:
  - Description of the approach used
  - Output figures as described in Part 1
  - Timestamp of missing beats in Part 2
- Submission of the above items should be as one .zip file by the deadline.

Guidelines:

- This is a group project. A maximum of 3 students per group is allowed.
- Each team must send an e-mail by **Sunday, November 20 at mid-night** specifying the members of the team.
- **Changing teams will not be allowed.**
- Project evaluation will occur in the class of **December 12**
- Project grading will be as follows (out of 15):
  - 5 points of the code will be submitted
  - 5 points on the submitted report
  - 5 points on the evaluation and discussion