ECG Signal Processing and Analysis

Sofia Noemi Crobeddu
student number: 032410554
International Master of Biometrics and Intelligent Vision - Université Paris-Est Cretéil (UPEC)
Assignment 1

1 Introduction

Congestive heart failure (i.e. CHF) is a problematic condition and it is about the heart that struggles to pump blood efficiently. This can lead to various complications such as fluid buildup and reduced oxygen supply to the organs of a person. Patients affected by a severe CHF often require constant monitoring and treatment. An alternative to the conventional therapies is the long-term oral milrinone one, an oral inotropic agent, that was well tolerated and provided relief in many cases in a study conducted by Baim et al. and described in the article [1]. This study involved 100 patients with serious CHF were treated with oral milrinone during 2.5 years. These people showed that milrinone provided early improvements in hemodynamic and clinical conditions for 51% of the patients and it did not significantly reduce the high baseline mortality connected to this disorder. Life table analysis showed, in fact, a 39% mortality rate at 6 months and a 63% mortality rate at 1 year of therapy.

Beside these results, to manage and have an overall idea of CHF, electrocardiography (i.e. ECG) is very important to monitoring the heart function. ECG is essential since it allows for real-time observation of electrical activity of the heart, central also to control arrhythmias for example.

In this report we will analyze data that describe just a part of people from this big study conducted by Baim et al.: the BIDMC Congestive Heart Failure Database (see the article [2]) from PhysioNet. This database includes long-term ECG recordings from 15 subjects (11 men, aged 22 to 71, and 4 women, aged 54 to 63), with severe CHF. These patients were subjected preliminarily to conventional medical therapy before receiving milrinone. The recordings had been sampled at 250 Hz and are about 20 hours in duration. Moreover, the original analog recordings were performed with a recording bandwidth from 0.1 Hz to 40 Hz. It is important also to specify that for each recording there are two ECG signals with 12-bit resolution over a range of ± 10 millivolts.
In this specific report we will concentrate just on the first record $chf01$, analyzing some key cardiovascular parameters and we will also focus on the detection of R-peaks. The analysis is made through the programming language Python.

2 Signal Acquisition process

The original format of the data is the WFDB format. It is composed by .dat files containing the raws' values of the ECG signal and .hea files with metadata such as signal names, sampling frequency, and signal units.
The files with extentions .ecg, .ecg- and .hea- were not considered since the first two are respectively the annotation files and their copies, while the files .hea- are the copies of .hea.

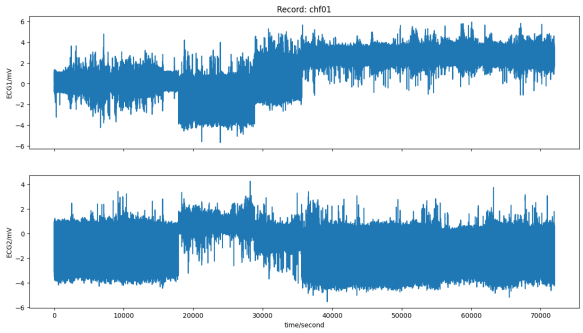

Fig. 1: ECG signals.

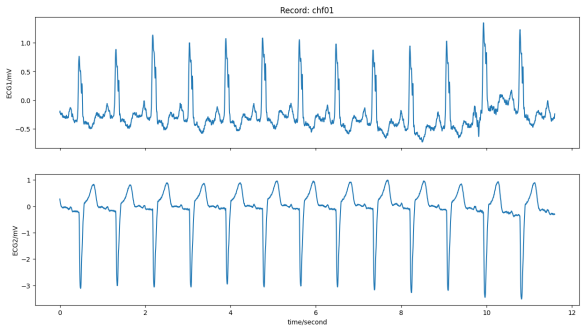

Fig. 2: Part of ECG signals.

To load these ECG data of the first patient, it was necessary the *wfdb* package in Python, to read and extract data from the signal. In particular, the function *rdrecord* was used to catch the Record object containing the signal and metadata, while the function p_signal outputs a bidimensional array where each row is a sample and the columns correspond to the two ECG signals.

After converting the signal into a dataframe, we transform this dataset into a file with a common format .csv.

Below it is showed the code performed for this starting step in the analysis:

```
import wfdb
import pandas as pd

#Loading of ECG signal of record chf01
record = wfdb.rdrecord('C:/Users/sofyc/OneDrive/Desktop/UPEC/Data capture and processing/
    assignment 1/bidmc-congestive-heart-failure-database-1.0.0/files/chf01')

#Extracting data from signal
signal = record.p_signal

#Conversion into DataFrame
df = pd.DataFrame(signal, columns=record.sig_name)

#Saving in CSV
df.to_csv('output.csv', index=False)
#path for output.csv: "C:\Users\sofyc\OneDrive\Desktop\UPEC\Data capture and processing\assignment
    1\output.csv"
```

The dataset *df* we created before is really big, in fact we have 17994491 rows for each signal. After checking there are not NA in the dataset, to have an idea of how the signals are, we can see their visualizations in Figure 1.

We can also look just at a specific part of the signal, taking for example just a range of samples from 100 to 3000. The zoomed parts of the ECG signals are shown in Figure 2.

Note that the graphs are created through the library *matplotlib.pyplot* of Python.

3 Basic Signal Characteristics and Methodologies

In this section we calculate some basic signal characteristics to have a general overview of the signal.

- **Average signal value (mean of the signal)**:
  it is the mean of the signal within time. The calculation is performed using the function for a dataframe from the library *pandas*.
  The code used is the following:

```
mean_ECG1 = df['ECG1'].mean()
mean_ECG2 = df['ECG2'].mean()
```

  The outputs are a mean of 1.183 for the ECG1 and a mean of -0.346 for ECG2. As we can see the first signal is positive on average, while the second one is negative on average. Ideally the mean of a signal should be around 0, and as we saw from Figure 1, the first one has a bigger offset (i.e. the mismatch of the signal from 0).

- **Energy of the signal**:
  for its calculation we need to perform the sum of the squared values of the signal, transforming it in an array before, through the sum's function of the library *Numpy*. It represents the overall intensity of the signal.
  The code used is the following:

```
import numpy as np
energy_ECG1 = np.sum(df['ECG1']**2)
energy_ECG2 = np.sum(df['ECG2']**2)
```

  The outputs are an energy of 58371197.127 for the ECG1 and an energy of 13065603.215 for ECG2. As we can see the first signal has a higher energy compared to the second one.

- **Variance**:
  it is the variability of the signal within time. As for the mean, the calculation is performed using the function for a dataframe from the library *pandas*.
  The code used is the following:

```
var_ECG1 = df['ECG1'].var()
var_ECG2 = df['ECG2'].var()
```

  The outputs are a variance of 1.845 for the ECG1 and a variance of 0.606 for ECG2. As we can see the first signal has a higher variability. This can be due to noise or to physiological phenomenons such as irregular beats.

- **Signal-to-noise ratio (SNR):**
  it is a measure of the quality of the signal, and it is defined as the ratio of the signal power to the noise power. It is often expressed in decibel and it compares the level of a suitable signal to the level of noise. A ratio greater than 0 dB represents more signal than noise. In formula it can be written as:

$$SNR = \frac{P_{signal}}{P_{noise}}$$

  and in the case of signal and noise expressed through the logarithmic decibel scale, the formula can be rewritten as:

$$P_{signal,dB} = 10log_{10}(P_{signal})$$

  and

$$P_{noise,dB} = 10log_{10}(P_{noise}).$$

  Hence, the final equation is:

$$SNR_{dB} = 10log_{10}(\frac{P_{signal}}{P_{noise}})$$

  The code used is the following:

```
SNR_ECG1 = 10 * np.log10(energy_ECG1/var_ECG1)
SNR_ECG2 = 10 * np.log10(energy_ECG2/var_ECG2)
```

The outputs are a SNR of 75.001 dB for the ECG1 and a SNR of 73.334 dB for ECG2. As we can see we have a value greater than 0 for both, and this suggests that data registered have a good quality compared to the noise component.

4 R-peak Detection Method

In this section we analyze the R-peaks of the signals. They are the peaks of the R wave and are associated to the QRS complex, the combination between Q, R and S waves. These three are the graphical deflections of a usual electrocardiogram. QRS complex is connected to the depolarization of the ventricles of the heart, a process that leads to the ventricular contraction of muscles and thus, to the expulsion of blood into the circulatory system.
To detect the R-peaks the first thing to do is apply a filter to reduce noise frequencies. In this case we know that the original analog recordings were already restricted to a bandwidth between 0.1 Hz and 40 Hz that is commonly suitable. To improve the accuracy without causing a distort analysis, we can apply a Band-pass Filter taking the range $[0.5; 40$ Hz$]$, a common used range to have the principal information. In this way we can be sure that the frequencies detected are really up to 40 Hz, and in the same time fix a low cutoff for the high-pass filter at 0.5 Hz. Going more deeply with the definitions, a band-pass filter is an algorithm that maintains the same information of frequency's values at frequencies inside the range, and putting to 0 the ones outside. It is basically a combination between the low-pass filter (a filter that puts to 0 high frequencies and maintains the low frequencies) and high-pass (a filter that puts to 0 low frequencies and maintains the high frequencies).

To implement this algorithm we use two functions:

- Function for Butterworth filter, that is a digital filter which doesn't introduce variation in the signal of the band-pass (frequencies we want to maintain). Out of the band, the filter reduces gradually the width of frequencies. This "lower" answer of the filter doesn't introduce distortions in the filtering of the signal. *butter* function calculates the coefficients of Butterworth filter that will be used to filter the signal. Note also that, before applying *butter*, we fix the equation regarding the maximum frequency of the signal based on Nyquist-Shannon Theorem such that

$$f_s \geq 2f_{max}$$

  where f_s is the sampling frequency equal to 250 Hz. Hence by this constraint we have that:

$$f_{max} \leq \frac{1}{2} f_s.$$

- Function of band-pass filter, where we use *filtfilt* to apply the filter to the signal. This function does the filtering for two times (one forward and one backward), to avoid distortions.

The code used is the following:

```
from scipy.signal import butter, filtfilt

fs = 250 #sampling frequency of 250 Hz
low_cutoff = 0.5  #cutoff frequency of high-pass filter
high_cutoff = 40  #cutoff frequency of low-pass filter

#Function for Butterworth filter
def bandpass_butterworth(lowcut, highcut, fs, order=4):
    #higher order means a narrower transition between the maintained (i.e. passed) band, and the chnged band, but this can cause distortions
    #analog=False since it is a digital filter
    nyquist = 0.5 * fs #Shannon-Nyquist Theorem
    low = lowcut / nyquist #normalized frequency
    high = highcut / nyquist #normalized frequency
    b, a = butter(order, [low, high], btype='band') #coefficients of Butterworth filter
    return b, a

#Function for band-pass filter
def bandpass_filter(data, lowcut, highcut, fs, order=4):
```

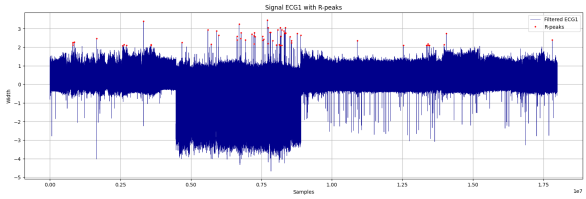

Fig. 3: ECG1 R-peaks.

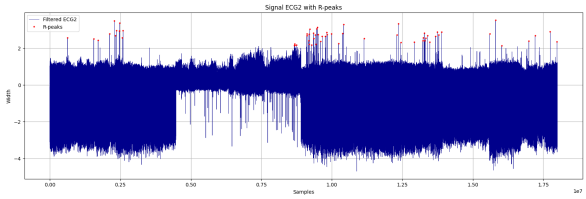

Fig. 4: ECG2 R-peaks.

```
    b, a = bandpass_butterworth(lowcut, highcut, fs, order=order)
    y = filtfilt(b, a, data)
    return y

#Application to data
filtered_ecg1 = bandpass_filter(df['ECG1'].values, low_cutoff, high_cutoff, fs)
filtered_ecg2 = bandpass_filter(df['ECG2'].values, low_cutoff, high_cutoff, fs)
```

After the filtering of the signals, we can detect the R-peaks. The method used for this step is the **Thresholding technique**: for each of the two ECG signals, we calculated a threshold fixed at the 60% of the maximum value of the signal. The R waves with a value higher than the threshold represents our R-peaks. After this, we used the function $find_peaks$from the python library $SciPy$ to identify the positions the picks in the filtered signals. In particular, this function search for the local maximum greater than the threshold, and ensures that the distance between the peaks is at least of a certain amount of samples. In this case we used 150 samples as distance, since they are equivalent to 0.6, in fact we have that the sampling frequency is defined as:

$$f_s = \frac{1}{T_s}$$

where T_s is the sampling period between two successive samples. Hence, remembering that in our case $f_s = 250$ Hz, we have that:

$$T_s = \frac{\text{Number of samples}}{f_s} = \frac{150}{250} = 0.6.$$

The code used to implement this technique is the following:

```
#Threshold: the 60% of the maximum
t_ecg1 = 0.6 * max(filtered_ecg1)
t_ecg2 = 0.6 * max(filtered_ecg2)

#Extracting the peaks from the second derivatives
peaks_ecg1, _ = find_peaks(filtered_ecg1, height=t_ecg1, distance=150)
peaks_ecg2, _ = find_peaks(filtered_ecg2, height=t_ecg2, distance=150)
```

The Figures 3 and 4 show the plots of the two ECG signals in dark-blue with detected R-peaks in red. As we can see, even if we applied the filters, we have still higher peaks in the first ECG signal compared to the second one.

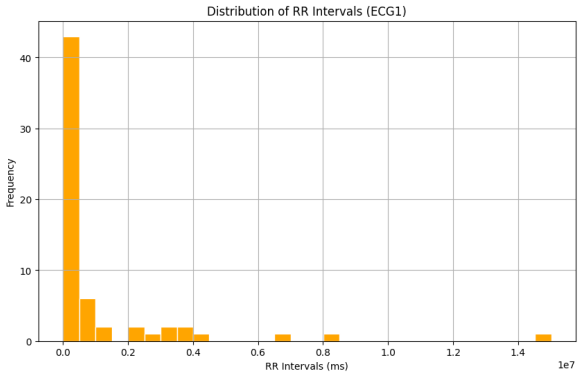

Fig. 5: ECG1's histogram for RR intervals.

5 Cardiovascular Parameter Calculation

In this final section we compute some key cardiovascular parameters for the two ECG signals and in particular RR intervals, Heart rate and its variability. They are really important since they provide insights into the functioning of the heart and through them, it is possible to assess heart's rhythm and see if there are potential irregularities.

5.1 Presentation of the results

Let's now look at the parameters and their results:

- RR intervals: they measure the time between two consecutive R-peaks in a ECG signal. These intervals reflects heart rhythm and they are calculated as the distance between the picks R in samples (measure unit).
  The code used is the following:

```
rr_intervals_ecg1 = np.diff(peaks_ecg1)
rr_intervals_ecg2 = np.diff(peaks_ecg2)
```

  The output of the result is shown in the Appendix (see Figure 9). To have a clearer vision of this output, we can also look at the representations in Figures 5 and 6 that were made transforming the RR intervals into milliseconds in this way:

```
rr_intervals_ms_ecg1 = rr_intervals_ecg1 / fs x 1000
rr_intervals_ms_ecg2 = rr_intervals_ecg2 / fs x 1000
```

  As we can notice, for the ECG2 we have an higher frequency of RR intervals between 0 and 0.5 ms, compared to the first ECG, but in the meantime we have also, a higher frequency of RR larger intervals (around 2.3 or 2.4 ms) in the ECG2 while in the ECG1 the largest RR interval is close to 1.4 ms.
- Heart rate: it is the frequency between the RR intervals and it is expressed in beats per minute (bpm).
  The code used is the following:

```
fs = 250
heart_rate_ecg1 = 60 / (rr_intervals_ecg1 / fs)
heart_rate_ecg2 = 60 / (rr_intervals_ecg2 / fs)
```

  The output of the result is shown in the Appendix (see Figure 10). From these results, performing the mean calculation, we have that the ECG1 has a heart rate mean of 5.57 bpm, while ECG2 has a heart rate of 1.85 bpm. In addiction, as we did for the previous parameter, we can look at the representations in Figures 7 and 8. As we can see the Heart rate of ECG1 seems to have a larger variability compared to the second one, and it also reaches a higher value, i.e. 70 bmp.
  Beside this, the graphs seem not very realistic since they represent heart rates really close to 0 for long periods. This is probably not a normal physiological behaviour of a patient with severe CHF, even if the condition of sever CHF can cause bradycardia and arrhythmias. In the case of the Figures in fact, the patient would had been in cardiac arrest. We can justify these graphs

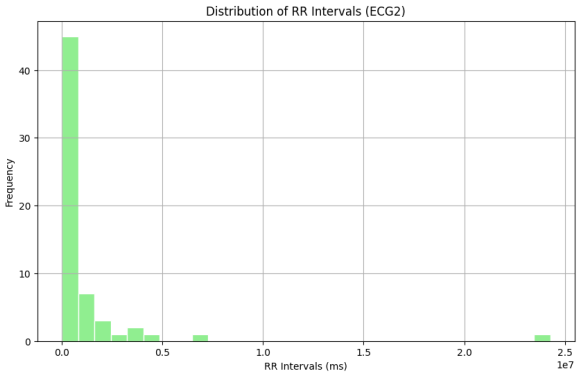

Fig. 6: ECG2's histogram for RR intervals

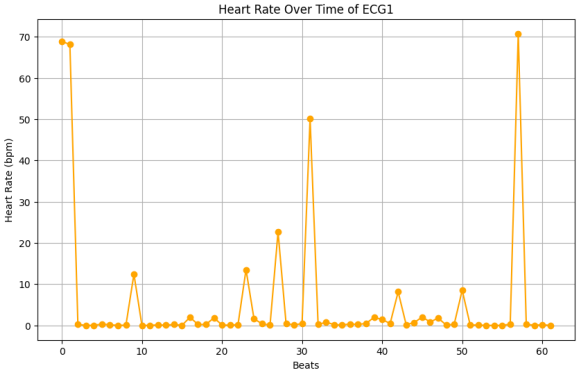

Fig. 7: Heart Rate of ECG1.

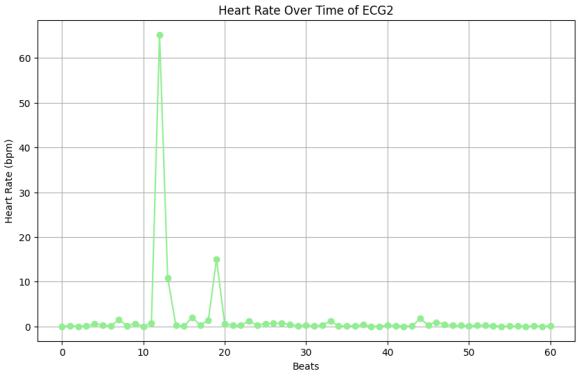

Fig. 8: Heart Rate of ECG2.

saying that, first of all, the two ECGs presents 0 values (in ECG1 19934 and in ECG2 53085). This is probably due to the fact that the recording lasted 20m hours and there could had been interruptions during the detection or measurement errors.

- Heart rate variability (HRV): it is an index of variability of RR intervals within time. It is useful to comprehend the adaptability of cardiovascular system. It is usually preferable to have a higher HRV since it indicates a good adaptability of the heart to changes or stress, and it is computed as the standard deviation of the RR intervals.
  The code used is the following:

```
hrv_ecg1 = np.std(rr_intervals_ecg1)
hrv_ecg2 = np.std(rr_intervals_ecg2)
```

The output for the Heart rate variability was of 593558.47 ms for ECG1 and of 808970.12 ms for ECG2. These numbers are really high, but as explained before, there could have been a problem during the detection.

5.2 Significance of the parameters and relation with cardiovascular health

As mentioned in the previous paragraph, RR intervals reflect the time between consecutive heartbeats. In the case of patients with arrythmias for example, the heartbeats are irregular.
Heart rate, instead, provides a measure of the number of heartbeats per minute. Hence, it is a fundamental for cardiovascular health. Moreover, in adults with a regular health condition, heart rate should be between 60 and 100 bpm.
Finally, HRV quantifies the autonomic nervous system's regulation of heart.

6 Conclusions

We can conclude that from this analysis ECG1 has a higher energy compared to the second signal, then in the first channel we have more activity or maybe it is more sensitive to movements of muscles. Moreover, since the SNR is elevate in both signals, we can say that, apparently, we have good quality of signals. This is contradicted by the calculation of the means, in fact we have slight distances from 0, and also by the last section's analysis since we have heart beat close to 0 for long periods. These results suggest that maybe, during the recording, there could have been a problem.

7 Appendix

```
RR intervals ECG1:  [    218     220   58827  789909  918292   44564   97457  592777  271368
    1208 1095477  919199  116383  195009   67216  654136    7568   65831
   63226    8106  156397  228262   71981    1110    9549   35483  266750
     661   33026  121413   33923     299   66569   19394   71896  137272
   41069   59359   31732    7588   10486   32468    1839   84361   21586
    7452   17869    8453  155510   51516    1756  203333  123357 2003688
 1643003  814503   50228     212   49694  520250   85261 3755269]
 RR intervals ECG2:  [ 929682  155005  406935  172839   25754   48466   97246   10247   99232
   22953 6069549   21542     230    1376   52701  372656    7310   57993
   10464    1001   25438   63839   69309   12031   56400   25291   21062
   22358   32431   86572   54631  111476   52706   12520  154376  249020
  142729   40474  718906 1164737   59441   78781  482523  300877    8595
   43420   15580   32300   73470   75927  198397   68228   49243   93989
 1712206  207324  217147  963026  220385  534566  244370]
```

Fig. 9: RR intervals outputs.

```
Heart Rate ECG1:  [6.88073394e+01 6.81818182e+01 2.54984956e-01 1.89895292e-02
 1.63346735e-02 3.36594561e-01 1.53914034e-01 2.53046255e-02
 5.52754931e-02 1.24172185e+01 1.36926654e-02 1.63185556e-02
 1.28884803e-01 7.69195268e-02 2.23161152e-01 2.29310113e-02
 1.98202960e+00 2.27856177e-01 2.37244172e-01 1.85048113e+00
 9.59097681e-02 6.57139603e-02 2.08388325e-01 1.35135135e+01
 1.57084511e+00 4.22737649e-01 5.62324274e-02 2.26928896e+01
 4.54187610e-01 1.23545255e-01 4.42177873e-01 5.01672241e+01
 2.25330109e-01 7.73435083e-01 2.08634695e-01 1.09272102e-01
 3.65238988e-01 2.52699675e-01 4.72708937e-01 1.97680548e+00
 1.43047873e+00 4.61993347e-01 8.15660685e+00 1.77807281e-01
 6.94894839e-01 2.01288245e+00 8.39442610e-01 1.77451792e+00
 9.64568195e-02 2.91171675e-01 8.54214123e+00 7.37706127e-02
 1.21598288e-01 7.48619546e-03 9.12962423e-03 1.84161384e-02
 2.98638210e-01 7.07547170e+01 3.01847306e-01 2.88322922e-02
 1.75930378e-01 3.99438762e-03]
 Heart Rate ECG2:  [1.61345492e-02 9.67710719e-02 3.68609237e-02 8.67859684e-02
 5.82433797e-01 3.09495316e-01 1.54247990e-01 1.46384308e+00
 1.51160916e-01 6.53509345e-01 2.47135331e-03 6.96314177e-01
 6.52173913e+01 1.09011628e+01 2.84624580e-01 4.02515993e-02
 2.05198358e+00 2.58651906e-01 1.43348624e+00 1.49850150e+01
 5.89668999e-01 2.34966087e-01 2.16422110e-01 1.24677915e+00
 2.65957447e-01 5.93096358e-01 7.12183079e-01 6.70900796e-01
 4.62520428e-01 1.73266183e-01 2.74569384e-01 1.34558111e-01
 2.84597579e-01 1.19808307e+00 9.71653625e-02 6.02361256e-02
...
 2.04164965e-01 1.97558181e-01 7.56059819e-02 2.19851088e-01
 3.04611823e-01 1.59593144e-01 8.76062810e-03 7.23505238e-02
 6.90776294e-02 1.55759035e-02 6.80627084e-02 2.80601460e-02
 6.13823301e-02]
```

Fig. 10: Heart Rate outputs.

References

[1] Baim, D.S., Colucci, W.S., Monrad, E.S., Smith, H.S., Wright, R.F., Lanoue, A., Gauthier, D.F., Ransil, B.J., Grossman, W., Braunwald, E.: Survival of patients with severe congestive heart failure treated with oral milrinone. Journal of the American College of Cardiology **7**(3), 661–670 (1986) https://doi.org/10.1016/s0735-1097(86)80478-8 . PMID: 3950244

[2] PhysioNet: BIDMC Congestive Heart Failure Database. https://physionet.org/content/chfdb/1.0.0/. Published: Oct. 14, 2000. Version: 1.0.0 (2000)

[3] Goldberger, A.L., Amaral, L.A.N., Glass, L., Hausdorff, J.M., Ivanov, P.C., Mark, R.G., Mietus, J.E., Moody, G.B., Peng, C.-K., Stanley, H.E.: Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. Circulation **101**(23), 215–220 (2000) https://doi.org/10.1161/01.CIR.101.23.e215 . [Online]

[4] Wikipedia contributors: Signal-to-noise ratio. https://en.wikipedia.org/wiki/Signal-to-noise_ratio (2024)

[5] antimattercorrade: Pan_Tompkins_QRS_Detection. https://github.com/antimattercorrade/Pan_Tompkins_QRS_Detection/blob/main/Pan_Tompkins.ipynb (2024)

[6] MIT Lab for Computational Physiology: wfdb. https://wfdb-python.readthedocs.io/en/latest/wfdb.html. © Copyright 2018, MIT Lab for Computational Physiology. (2018)

[7] SciPy API — Signal processing (scipy.signal) — find_peaks. https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.find_peaks.html. © Copyright 2008-2024, The SciPy community. (2024)

[8] QRS complex. https://en.wikipedia.org/wiki/QRS_complex

[9] Nyquist–Shannon sampling theorem. https://en.wikipedia.org/wiki/Nyquist%E2%80%93Shannon_sampling_theorem

[10] Heart rate. https://en.wikipedia.org/wiki/Heart_rate#:~:text=The%20American%20Heart%20Association%20states,below%2060%20bpm%20at%20rest.

[11] Nait-Ali, A.: Session: Spectral Analysis. Lecture slides, Université Paris-Est Créteil (UPEC) - Faculté des Sciences et Technologie (2024)

[12] Nait-Ali, A.: Session: Digital Signal Processing. Lecture slides, Université Paris-Est Créteil (UPEC) - Faculté des Sciences et Technologie (2024)