Medical Informatics-Signal Correletion
Assignment 1

Vasileios Nastos
MSc in department of Informatics
and Telecommunications
University of Ioannina
Arta, 47100, Greece
Id:137

April 5, 2023

1 Introduction

In signal processing, correlation refers to the process of comparing two signals and measuring the degree to which they are similar. Signal correlation is often used in applications such as digital communications, speech processing, and image processing. The most common type of correlation used in signal processing is cross-correlation. Cross-correlation is a measure of similarity between two signals as a function of the time shift applied to one of the signals. It is defined as the integral of the product of two signals, one of which is shifted with respect to the other. Mathematically, the cross-correlation of two signals x(t) and y(t) is given Equation 1:

$$R_{xy}(\tau) = \int_{-\infty}^{\infty} x(t)y(t-\tau)dt \tag{1}$$

where τ is the time shift, and Rxy(τ) is the resulting correlation function. The cross-correlation function Rxy(τ) indicates the similarity between the two signals at different time shifts τ. A high value of Rxy(τ) indicates that the two signals are similar, while a low value indicates that they are dissimilar. Cross-correlation is often used in applications such as speech recognition, where it is used to compare an input speech signal with a set of pre-recorded speech signals to identify the closest match. It is also used in image processing to align two images, by measuring the degree of similarity between them at different spatial shifts.

2 ECG Data

ECG (Electrocardiogram) data refers to recordings of the electrical activity of the heart over time. ECG recordings are obtained by placing electrodes on the skin of a person's chest, arms, and legs, which measure the electrical activity of the heart and transmit it to a recording device. The resulting ECG data typically consists of a time series of voltage measurements that reflect the electrical activity of the heart during each heartbeat. The ECG waveform is characterized by a series of distinct peaks and valleys, each of which corresponds to a different phase of the cardiac cycle. ECG data is commonly used in medical applications to diagnose and monitor heart conditions, such as arrhythmias, heart attacks, and other cardiovascular diseases. ECG data can also be used in research to study the electrical activity of the heart and to develop new methods for analyzing and interpreting ECG recordings. The analysis of ECG data involves various techniques such as signal processing, feature extraction, and machine learning. These techniques are used to extract meaningful information from the ECG waveform, such as heart rate variability, QRS complex duration, and ST segment changes, which can be used to diagnose and monitor various heart conditions.

For this assignment we use a subset of the data that are single column(Specififc frequency). Descriptive analytics for the data are presented in table 1

Statistic Meter	Value
Samples	65529
Mean	0.123
Median	-0.062
Std	0.567
Iqr	0.126
Skewness	2.574
Kurtosis	9.166

Table 1: Dataset descriptive analytics

3 Code and Results

For the implementaion stage why use python and more specifically the numpy llibrary that contains a correlation bultin function. Also we use the matplotlib module for the visualization stage. Results are presented in Figures 1,2, while the code for the first assignment is presented in Listing 1 and the code for the second assignment is presented in Listing 2. The full code is presented in Listing 3.

Listing 1: code for assingment 1

```
class Signal:
    def __init__(self) -> None:
        pass

    def sread(self,path_to_signal:str):
        self.ecg_signal=np.loadtxt(path_to_signal)
        self.random_noisy_ecg=0.5*np.random.randn(len(self.ecg_signal))
        self.noisy_ecg_signal=self.ecg_signal+self.random_noisy_ecg

        # Numpy correlation calculation
        self.corr_ecg_signal_with_noise=np.correlate(self.ecg_signal,self.noisy_ecg_signal,mode='same')
        self.corr_random_signal=np.correlate(self.ecg_signal,self.random_noisy_ecg,mode='same')

    def plotall(self):
        fig,axs=plt.subplots(nrows=3,ncols=2,figsize=(8,8))

        for i in range(2):
            axs[0,i%2].set_title('ECG')
            axs[0,i%2].plot(np.arange(len(self.ecg_signal)),self.ecg_signal,color='blue',linewidth=2)
            axs[0,i%2].spines['top'].set_visible(False)
            axs[0,i%2].spines['right'].set_visible(False)

        axs[1,0].set_title('ECG + Noise')
        axs[1,0].plot(np.arange(len(self.noisy_ecg_signal)),self.noisy_ecg_signal,color='blue',linewidth
            =2)
        axs[1,0].spines['top'].set_visible(False)
        axs[1,0].spines['right'].set_visible(False)

        axs[1,1].set_title('Random Noise')
        axs[1,1].plot(np.arange(len(self.random_noisy_ecg)),self.random_noisy_ecg,color='blue',linewidth
            =2)
        axs[1,1].spines['top'].set_visible(False)
        axs[1,1].spines['right'].set_visible(False)

        axs[2,0].set_title('Correletion of ECG + Noise')
        axs[2,0].plot(np.arange(len(self.corr_ecg_signal_with_noise)),self.corr_ecg_signal_with_noise,
            color='blue',linewidth=2)
        axs[2,0].spines['top'].set_visible(False)
        axs[2,0].spines['right'].set_visible(False)

        axs[2,1].set_title('Corretion Ecg with Random Noise')
        axs[2,1].plot(np.arange(len(self.corr_random_signal)),self.corr_random_signal,color='blue',
            linewidth=2)
        axs[2,1].spines['top'].set_visible(False)
        axs[2,1].spines['right'].set_visible(False)

        fig.tight_layout()
```

```
    fig.savefig(fname=os.path.join('','ecg_askisi1.png'),dpi=300)
    plt.show()
```

Listing 2: Assignment 2 code

```
class Signal2:
def __init__(self) -> None:
    pass

def sread(self,path_to_signal:str):
    self.ecg_signal=np.loadtxt(path_to_signal)
    f1=17
    f2=10

    self.sine1 = np.sin(2*np.pi*f1*np.arange(len(self.ecg_signal))*(1/len(self.ecg_signal)))
    self.sine2 = np.sin(2*np.pi*f2*np.arange(len(self.ecg_signal))*(1/len(self.ecg_signal)))

    self.random_noisy_ecg=0.5*np.random.randn(len(self.ecg_signal))
    self.noisy_ecg_signal=self.ecg_signal+self.random_noisy_ecg

    # Numpy correlation calculation
    self.corr_ecg_signal_sine1=np.correlate(self.ecg_signal,self.sine1,mode='same')
    self.corr_ecg_signal_sine2=np.correlate(self.ecg_signal,self.sine2,mode='same')

def plotall(self):
    fig,axs=plt.subplots(nrows=3,ncols=2,figsize=(8,8))

    for i in range(2):
        axs[0,i%2].set_title('ECG')
        axs[0,i%2].plot(np.arange(len(self.ecg_signal)),self.ecg_signal,color='blue',linewidth=2)
        axs[0,i%2].spines['top'].set_visible(False)
        axs[0,i%2].spines['right'].set_visible(False)

    axs[1,0].set_title('Sin Wave, f=17')
    axs[1,0].plot(np.arange(len(self.sine1)),self.sine1,color='blue',linewidth=2)
    axs[1,0].spines['top'].set_visible(False)
    axs[1,0].spines['right'].set_visible(False)

    axs[1,1].set_title('Sin Wave, f=10')
    axs[1,1].plot(np.arange(len(self.sine2)),self.sine2,color='blue',linewidth=2)
    axs[1,1].spines['top'].set_visible(False)
    axs[1,1].spines['right'].set_visible(False)

    axs[2,0].set_title('Correletion with Sin, f=17')
    axs[2,0].plot(np.arange(len(self.corr_ecg_signal_sine1)),self.corr_ecg_signal_sine1,color='blue',
        linewidth=2)
    axs[2,0].spines['top'].set_visible(False)
    axs[2,0].spines['right'].set_visible(False)

    axs[2,1].set_title('Correletion with Sin, f=10')
    axs[2,1].plot(np.arange(len(self.corr_ecg_signal_sine2)),self.corr_ecg_signal_sine2,color='blue',
        linewidth=2)
    axs[2,1].spines['top'].set_visible(False)
    axs[2,1].spines['right'].set_visible(False)

    fig.tight_layout()
    fig.savefig(fname=os.path.join('','ecg_askisi2.png'),dpi=300)
    plt.show()
```

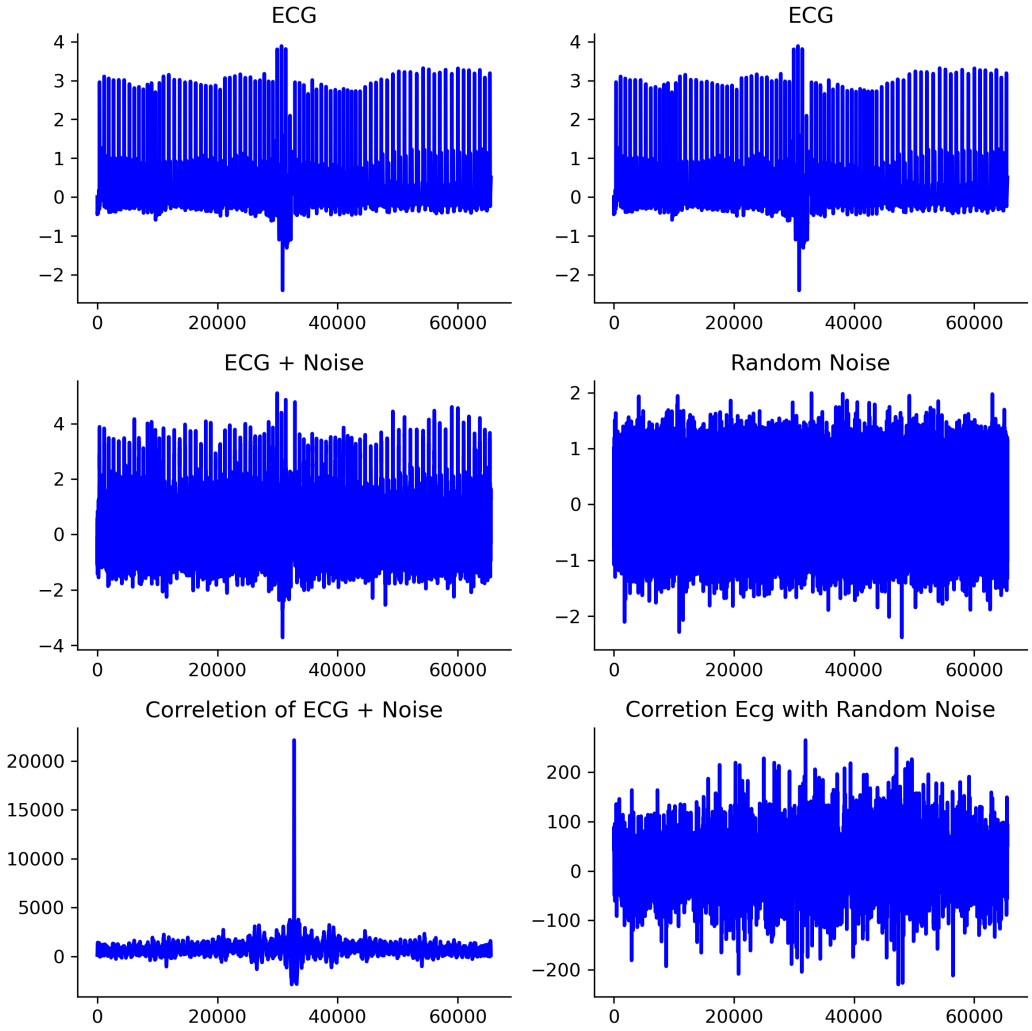

Figure 1: Assignment 1 results

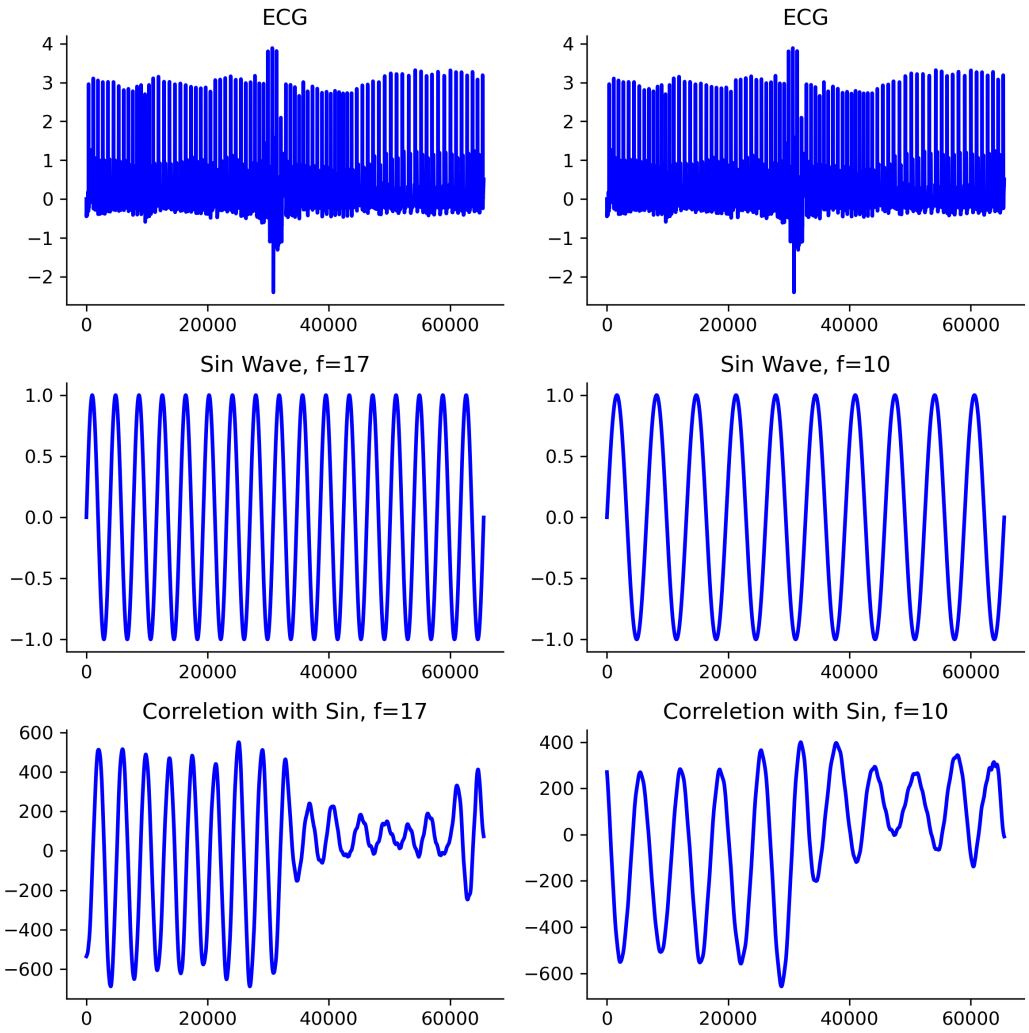

Figure 2: Assignment 2 results

Listing 3: Full assingment code

```
import os,numpy as np
import matplotlib.pyplot as plt
import statistics
from scipy.stats import skew,kurtosis
from tabulate import tabulate

class Signal:
    def __init__(self) -> None:
        pass

    def sread(self,path_to_signal:str):
        self.ecg_signal=np.loadtxt(path_to_signal)
        self.random_noisy_ecg=0.5*np.random.randn(len(self.ecg_signal))
        self.noisy_ecg_signal=self.ecg_signal+self.random_noisy_ecg

        # Numpy correlation calculation
        self.corr_ecg_signal_with_noise=np.correlate(self.ecg_signal,self.noisy_ecg_signal,mode='same')
        self.corr_random_signal=np.correlate(self.ecg_signal,self.random_noisy_ecg,mode='same')

    def statistics(self):
        q1,q3,_=statistics.quantiles(data=self.ecg_signal,n=4)
        rows=[
            ['Samples',self.ecg_signal.shape[0]],
```

```
            ['Mean',statistics.mean(self.ecg_signal)],
            ['Median',statistics.median(self.ecg_signal)],
            ['Std',statistics.stdev(self.ecg_signal)],
            ['Iqr',q3-q1],
            ['Skewness',skew(self.ecg_signal)],
            ['Kurtosis',kurtosis(self.ecg_signal)]
        ]
        print(tabulate(tabular_data=rows,headers=['Statistic Meter','Value'],tablefmt='fancy_grid',
            floatfmt='.3f'))
        with open(os.path.join('stats.tex'),'w') as writer:
            writer.write(tabulate(tabular_data=rows,headers=['Statistic Meter','Value'],tablefmt='latex',
                floatfmt='.3f'))

    def plotall(self):
        fig,axs=plt.subplots(nrows=3,ncols=2,figsize=(8,8))

        for i in range(2):
            axs[0,i%2].set_title('ECG')
            axs[0,i%2].plot(np.arange(len(self.ecg_signal)),self.ecg_signal,color='blue',linewidth=2)
            axs[0,i%2].spines['top'].set_visible(False)
            axs[0,i%2].spines['right'].set_visible(False)

        axs[1,0].set_title('ECG + Noise')
        axs[1,0].plot(np.arange(len(self.noisy_ecg_signal)),self.noisy_ecg_signal,color='blue',linewidth
            =2)
        axs[1,0].spines['top'].set_visible(False)
        axs[1,0].spines['right'].set_visible(False)

        axs[1,1].set_title('Random Noise')
        axs[1,1].plot(np.arange(len(self.random_noisy_ecg)),self.random_noisy_ecg,color='blue',linewidth
            =2)
        axs[1,1].spines['top'].set_visible(False)
        axs[1,1].spines['right'].set_visible(False)

        axs[2,0].set_title('Correletion of ECG + Noise')
        axs[2,0].plot(np.arange(len(self.corr_ecg_signal_with_noise)),self.corr_ecg_signal_with_noise,
            color='blue',linewidth=2)
        axs[2,0].spines['top'].set_visible(False)
        axs[2,0].spines['right'].set_visible(False)

        axs[2,1].set_title('Corretion Ecg with Random Noise')
        axs[2,1].plot(np.arange(len(self.corr_random_signal)),self.corr_random_signal,color='blue',
            linewidth=2)
        axs[2,1].spines['top'].set_visible(False)
        axs[2,1].spines['right'].set_visible(False)

        fig.tight_layout()
        fig.savefig(fname=os.path.join('','ecg_askisi1.png'),dpi=300)
        plt.show()

class Signal2:
    def __init__(self) -> None:
        pass

    def sread(self,path_to_signal:str):
        self.ecg_signal=np.loadtxt(path_to_signal)
        f1=17
        f2=10

        self.sine1 = np.sin(2*np.pi*f1*np.arange(len(self.ecg_signal))*(1/len(self.ecg_signal)))
        self.sine2 = np.sin(2*np.pi*f2*np.arange(len(self.ecg_signal))*(1/len(self.ecg_signal)))

        self.random_noisy_ecg=0.5*np.random.randn(len(self.ecg_signal))
        self.noisy_ecg_signal=self.ecg_signal+self.random_noisy_ecg

        # Numpy correlation calculation
        self.corr_ecg_signal_sine1=np.correlate(self.ecg_signal,self.sine1,mode='same')
        self.corr_ecg_signal_sine2=np.correlate(self.ecg_signal,self.sine2,mode='same')

    def statistics(self):
        q1,q3,_=statistics.quantiles(data=self.ecg_signal,n=4)
```

```
        rows=[
            ['Samples',self.ecg_signal.shape[0]],
            ['Mean',statistics.mean(self.ecg_signal)],
            ['Median',statistics.median(self.ecg_signal)],
            ['Std',statistics.stdev(self.ecg_signal)],
            ['Iqr',q3-q1],
            ['Skewness',skew(self.ecg_signal)],
            ['Kurtosis',kurtosis(self.ecg_signal)]
        ]
        print(tabulate(tabular_data=rows,headers=['Statistic Meter','Value'],tablefmt='fancy_grid',
             floatfmt='.3f'))
        with open(os.path.join('stats.tex'),'w') as writer:
            writer.write(tabulate(tabular_data=rows,headers=['Statistic Meter','Value'],tablefmt='latex',
                 floatfmt='.3f'))


    def plotall(self):
        fig,axs=plt.subplots(nrows=3,ncols=2,figsize=(8,8))

        for i in range(2):
            axs[0,i%2].set_title('ECG')
            axs[0,i%2].plot(np.arange(len(self.ecg_signal)),self.ecg_signal,color='blue',linewidth=2)
            axs[0,i%2].spines['top'].set_visible(False)
            axs[0,i%2].spines['right'].set_visible(False)

        axs[1,0].set_title('Sin Wave, f=17')
        axs[1,0].plot(np.arange(len(self.sine1)),self.sine1,color='blue',linewidth=2)
        axs[1,0].spines['top'].set_visible(False)
        axs[1,0].spines['right'].set_visible(False)

        axs[1,1].set_title('Sin Wave, f=10')
        axs[1,1].plot(np.arange(len(self.sine2)),self.sine2,color='blue',linewidth=2)
        axs[1,1].spines['top'].set_visible(False)
        axs[1,1].spines['right'].set_visible(False)

        axs[2,0].set_title('Correletion with Sin, f=17')
        axs[2,0].plot(np.arange(len(self.corr_ecg_signal_sine1)),self.corr_ecg_signal_sine1,color='blue',
             linewidth=2)
        axs[2,0].spines['top'].set_visible(False)
        axs[2,0].spines['right'].set_visible(False)

        axs[2,1].set_title('Correletion with Sin, f=10')
        axs[2,1].plot(np.arange(len(self.corr_ecg_signal_sine2)),self.corr_ecg_signal_sine2,color='blue',
             linewidth=2)
        axs[2,1].spines['top'].set_visible(False)
        axs[2,1].spines['right'].set_visible(False)

        fig.tight_layout()
        fig.savefig(fname=os.path.join('','ecg_askisi2.png'),dpi=300)
        plt.show()

if __name__=='__main__':
    import argparse
    parser=argparse.ArgumentParser("Argument Parser")
    parser.add_argument("--scenario",required=True,help="Select scenario(Available options(1|2))")
    args=parser.parse_args()

    if int(args.scenario)==1:
        signal_handler=Signal()
        signal_handler.sread(os.path.join('','ecg.txt'))
        signal_handler.statistics()
        signal_handler.plotall()
    elif int(args.scenario)==2:
        signal_handler2=Signal2()
        signal_handler2.sread(os.path.join('','ecg.txt'))
        signal_handler2.statistics()
        signal_handler2.plotall()
    else:
        raise ValueError("Not available option")
```