

Laboratorium 1

Autorzy: Krzysztof Zalewa 273032, Michał Pakuła 272828

Data: 24 Marca 2025

Ćwiczenie 1

Celem ćwiczenia było:

1. Napisanie skryptu w Pythonie umożliwiającego wczytywanie i wizualizację badanych sygnałów.
 - ekg1.txt – 12 kolumn odpowiada odprowadzeniom, fs = 1000 Hz
 - ekg100.txt – 1 kolumna, fs = 360 Hz
 - ekg_noise.txt – 1 kolumna: czas, 2 kolumna: wartości amplitud EKG, fs = 360 Hz
2. Umożliwienie obserwacji wycinka sygnału

```
In [21]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
In [22]: # Ładowanie wybranego pliku
def loadFile(file_name):
    file_path = "./src/"+file_name

    return pd.read_csv(file_path, sep="\\s+", header=None, engine="python")
```

Po załadowaniu pliku .txt są trzy możliwości

1. Plik ma 12 kolumn z danymi
2. Plik ma 2 kolumny z danymi
3. Plik ma 1 kolumnę z danymi

```
In [23]: # Sprawdź liczbę kolumn
def displayEKG(data_frame, start, end):
    num_rows = len(data_frame)
    tick_rate = 72000
    if start == "":
        start = 0
    if end == "":
        end = num_rows

    x = list(range(int( start ),int( end )))
    new_data = data_frame.iloc[int( start ):int( end )].copy()
    font = {'size':20}
    num_rows = len(new_data)
    minutes = num_rows/tick_rate
    x_labels = np.linspace(0,minutes,int( minutes ))

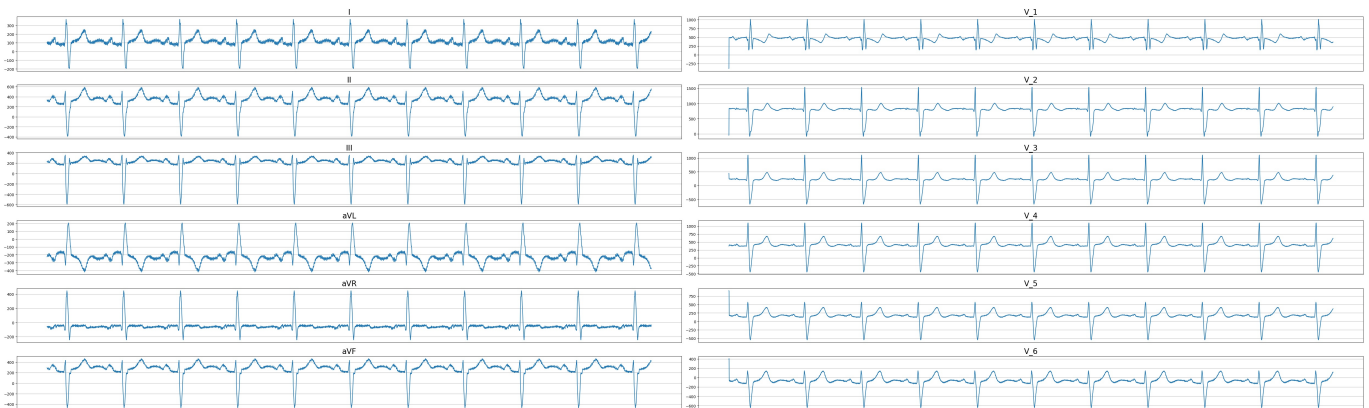
    if len( data_frame.columns ) == 12:
        column_names = ['I', 'II', 'III', 'aVL', 'aVR', 'aVF', 'V_1', 'V_2', 'V_3', 'V_4', 'V_5', 'V_6']
        new_data.columns = column_names
        plt.figure(figsize=(50,15))
        j=0
        for i in range(1,12,2):
            plt.subplot(6,2,i)
            plt.plot(x,new_data[column_names[j]])
            plt.grid(True)
            plt.xticks(ticks=x_labels*tick_rate,labels=x_labels)
            plt.title(column_names[j],fontdict=font)
            j += 1
        for i in range(2,13,2):
            plt.subplot(6,2,i)
            plt.plot(x,new_data[column_names[j]])
            plt.xticks(ticks=x_labels*tick_rate,labels=x_labels)
            plt.grid(True)
            plt.title(column_names[j],fontdict=font)
            j += 1
        plt.tight_layout()
        plt.show()

    if len(new_data.columns) == 1:
        new_data.columns = ['data']
        font = {'size':20}
        plt.figure(figsize=(20,5))
        plt.plot(x,new_data['data'])
```

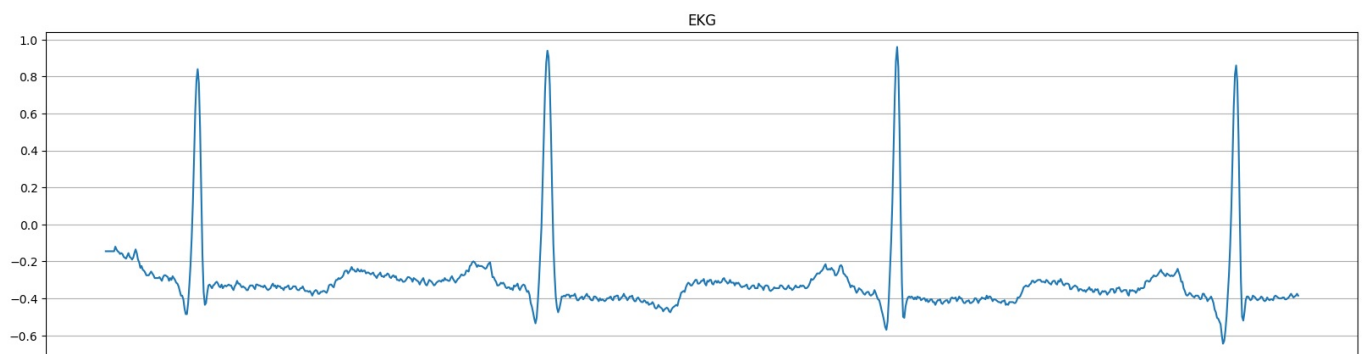
```
plt.xticks(ticks=x_labels*tick_rate, labels=x_labels)
plt.grid(True)
plt.title("EKG")
plt.show()

if len(new_data.columns) == 2:
    new_data.columns = ['I', 'data']
    font = {'size': 20}
    plt.figure(figsize=(20, 5))
    plt.plot(x, new_data['data'])
    plt.xticks(ticks=x_labels*tick_rate, labels=x_labels)
    plt.grid(True)
    plt.title("EKG")
```

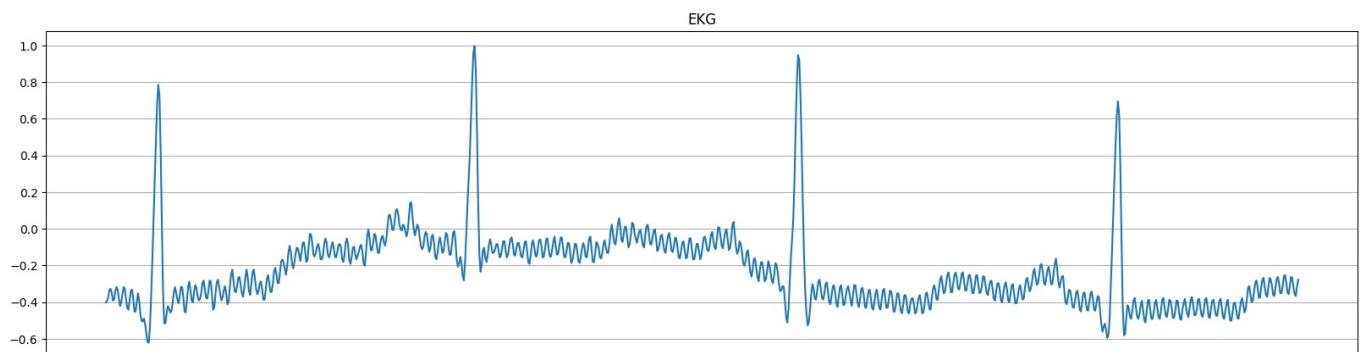
```
In [25]: data_frame = loadFile(input("Podaj nazwe pliku z danymi: "))
start = input("Początek zakresu (Minimalnie 0): ")
num_rows = len(data_frame)
end = input("Koniec zakresu (Maksymalnie "+str( num_rows )+"): ")
displayEKG(data_frame, start, end)
```



```
In [ ]: data_frame = loadFile("ekg100.txt")
displayEKG(data_frame, 0, 1000)
```



```
In [26]: data_frame = loadFile("ekg_noise.txt")
displayEKG(data_frame, 100, 1100)
```



Sygnały zapisane w plikach ekg_noise.txt oraz ekg100.txt zostały ograniczone do 1000 próbek. W przypadku próby wyświetlenia całego sygnału otrzymany obraz jest nie czytelny.

Ćwiczenie 2

Celem ćwiczenia było:

1. Wygeneruj ciąg próbek odpowiadający fali sinusoidalnej o częstotliwości 50 Hzi długości 65536.
2. Wyznacz dyskretną transformatę Fouriera tego sygnału i przedstaw jego widmo amplitudowe na wykresie w zakresie częstotliwości $[0, fs/2]$, gdzie fs oznacza częstotliwość próbkowania.
3. Wygeneruj ciąg próbek mieszaniny dwóch fal sinusoidalnych (tzn. ich kombinacji liniowej) o częstotliwościach 50 i 60 Hz. Wykonaj zadanie z punktu 2 dla tego sygnału.
4. Powtórz eksperymenty dla różnych czasów trwania sygnałów, tzn. dla różnych częstotliwości próbkowania.
5. Wyznacz odwrotne transformaty Fouriera ciągów wyznaczonych w zadaniu 2 i porównaj z ciągami oryginalnymi.

```
In [31]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

```
In [ ]: #Generacja widma sygnału sin_wave1 na podstawie fft

def displayWave(wave,fs,length,name):
    t = np.arange(length) / fs
    plt.figure(figsize=(20,5))
    plt.subplot(4,1,1)
    plt.plot(wave)
    plt.grid(True)
    plt.title(f"Sygnał {name} przed transformacją")

    fourier1 = np.fft.fft(wave)

    widmo = np.abs(fourier1)
    abs_widmo = widmo / np.max(widmo)
    freq = np.fft.fftfreq(len(t),1/fs)
    pos_freq = freq[:len(freq)//2]
    pos_widmo = abs_widmo[:len(abs_widmo)//2]

    plt.subplot(4,1,2)
    plt.plot(widmo[:length//2])
    plt.xlim(right = fs/2)
    plt.grid(True)
    plt.title(f"Sygnał {name} po transformacji")

    fourier2 = np.fft.ifft(fourier1)

    plt.subplot(4,1,3)
    plt.plot(fourier2)
    plt.grid(True)
    plt.title(f"Sygnał {name} po odwrotnej transformacji")

    plt.subplot(4,1,4)
    plt.plot(pos_freq,pos_widmo)
    plt.xlim(right = fs/2)
    plt.grid(True)
    plt.title("Widmo amplitudowe")
    plt.xlabel('Częstotliwość (Hz)')
    plt.ylabel('Amplituda')

    plt.subplots_adjust(hspace=0.75)
    plt.show()
```

Zadanie nr1

Jednocześnie generowana jest fala sinusoidalna do zadania 2 oraz mieszanina próbek do zadania 3.

```
In [34]: fs = 44100
freq1 = 50
freq2 = 60
length = 65536

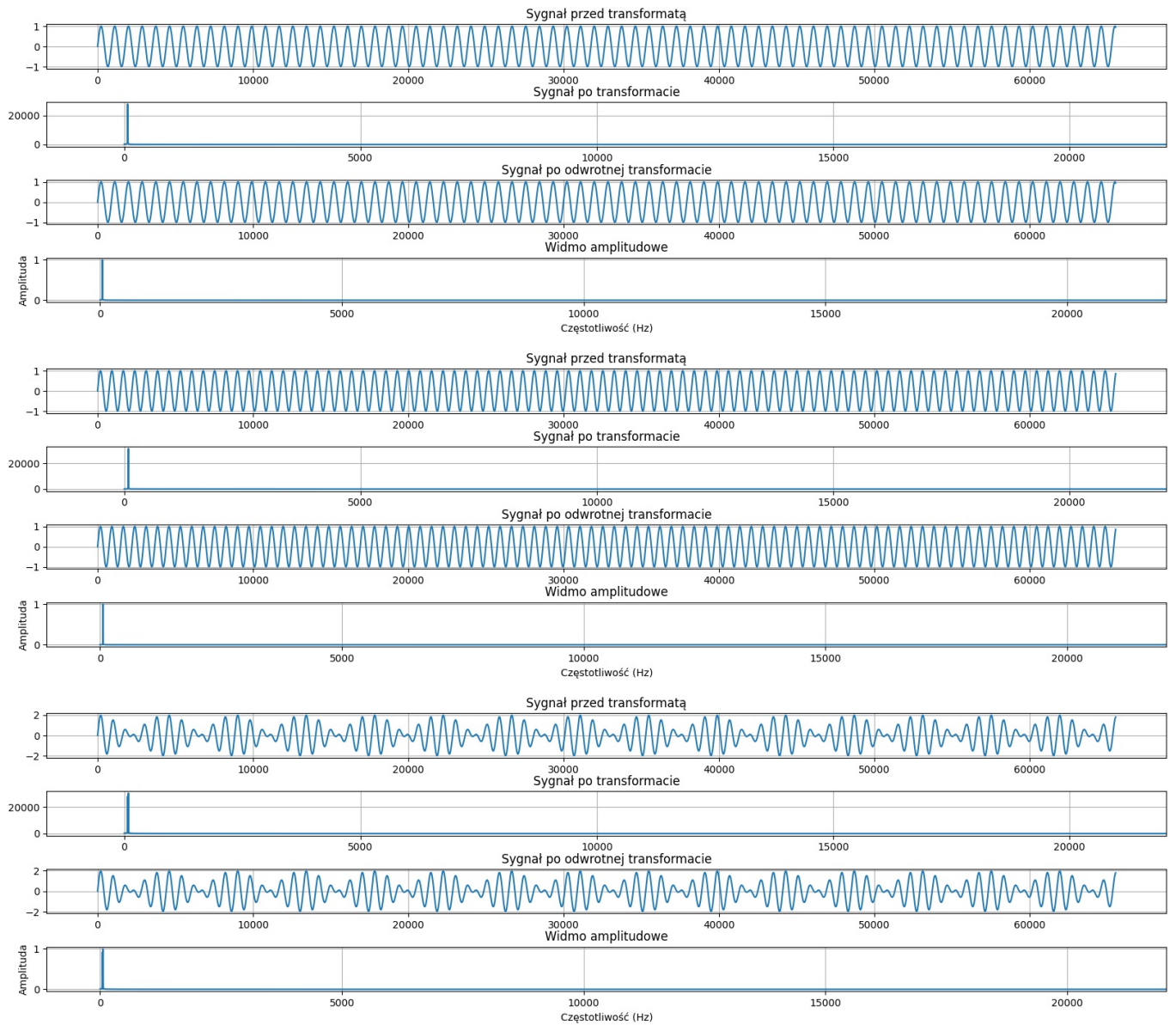
t = np.arange(length) / fs
sin_wave1 = np.sin(2 * np.pi * freq1 * t)

t = np.arange(length) / fs
sin_wave2 = np.sin(2 * np.pi * freq2 * t)

mixed_wave = sin_wave1 + sin_wave2

displayWave(sin_wave1,fs,length)
```

```
displayWave(sin_wave2,fs,length)
displayWave(mixed_wave,fs,length)
```



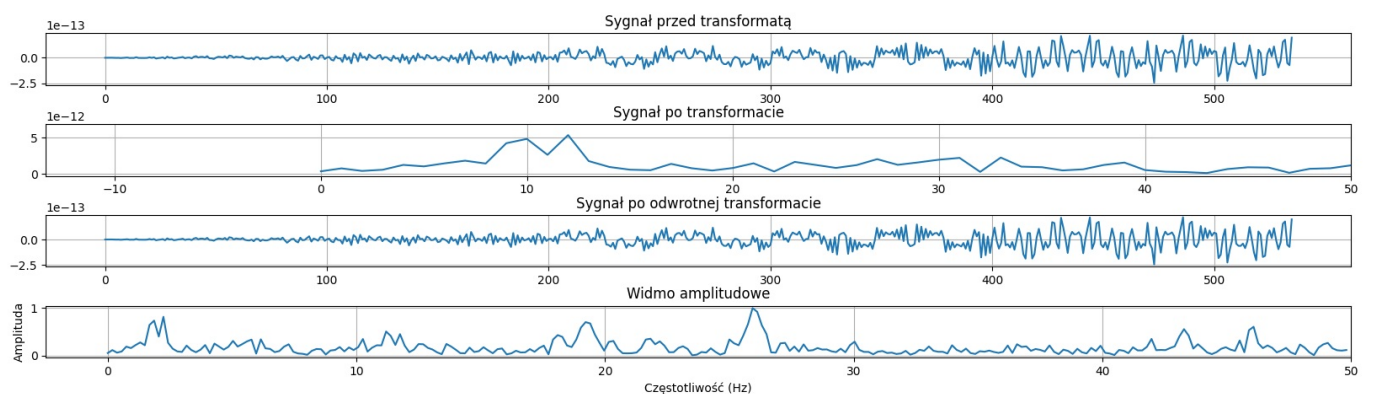
Do dalszych badań wybrano częstotliwości próbkowania 100Hz oraz 120Hz (Dwukrotności częstotliwości sygnału)

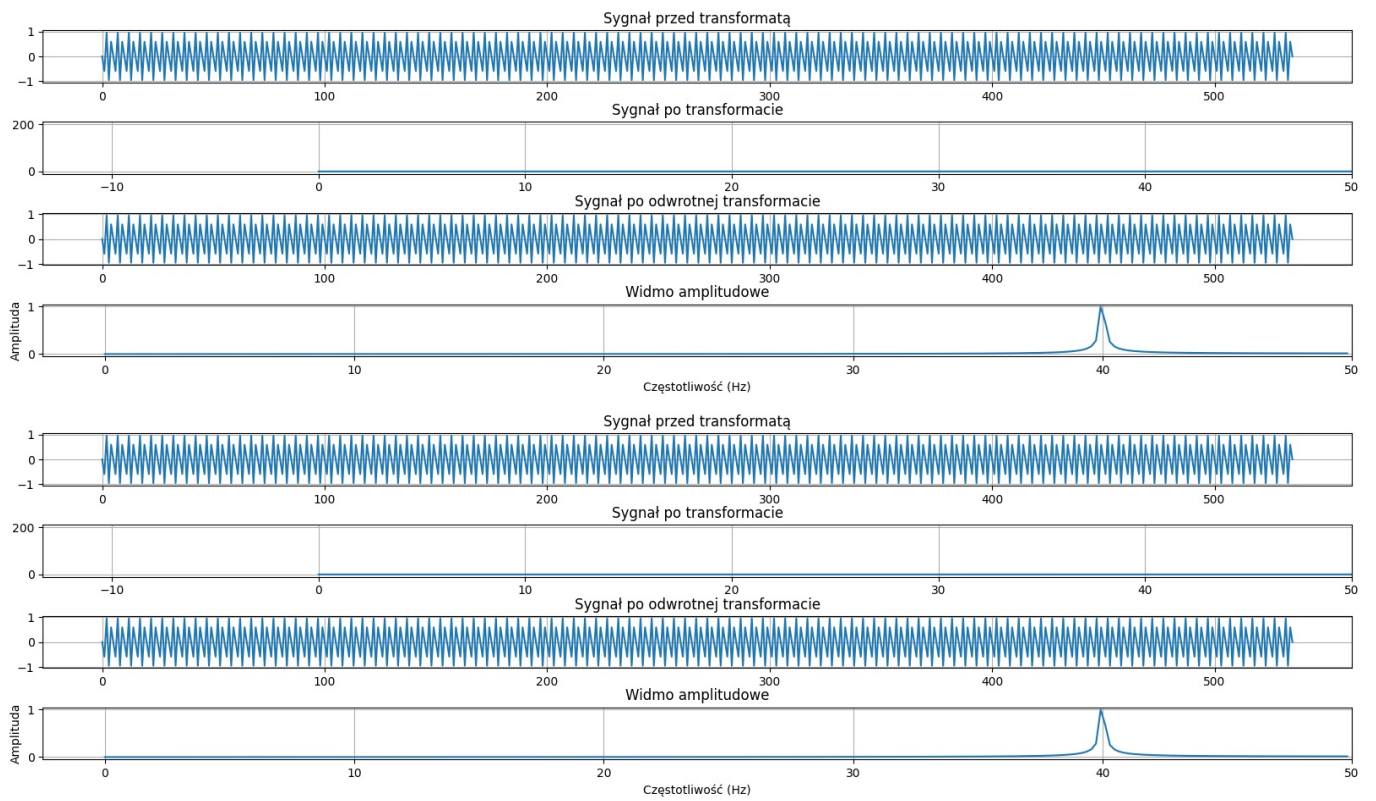
```
In [ ]: fs = 100
freq1 = 50
freq2 = 60
length = 536

t = np.arange(length) / fs
sin_wave1 = np.sin(2 * np.pi * freq1 * t)

t = np.arange(length) / fs
sin_wave2 = np.sin(2 * np.pi * freq2 * t)

displayWave(sin_wave1,fs,length)
displayWave(sin_wave2,fs,length)
```



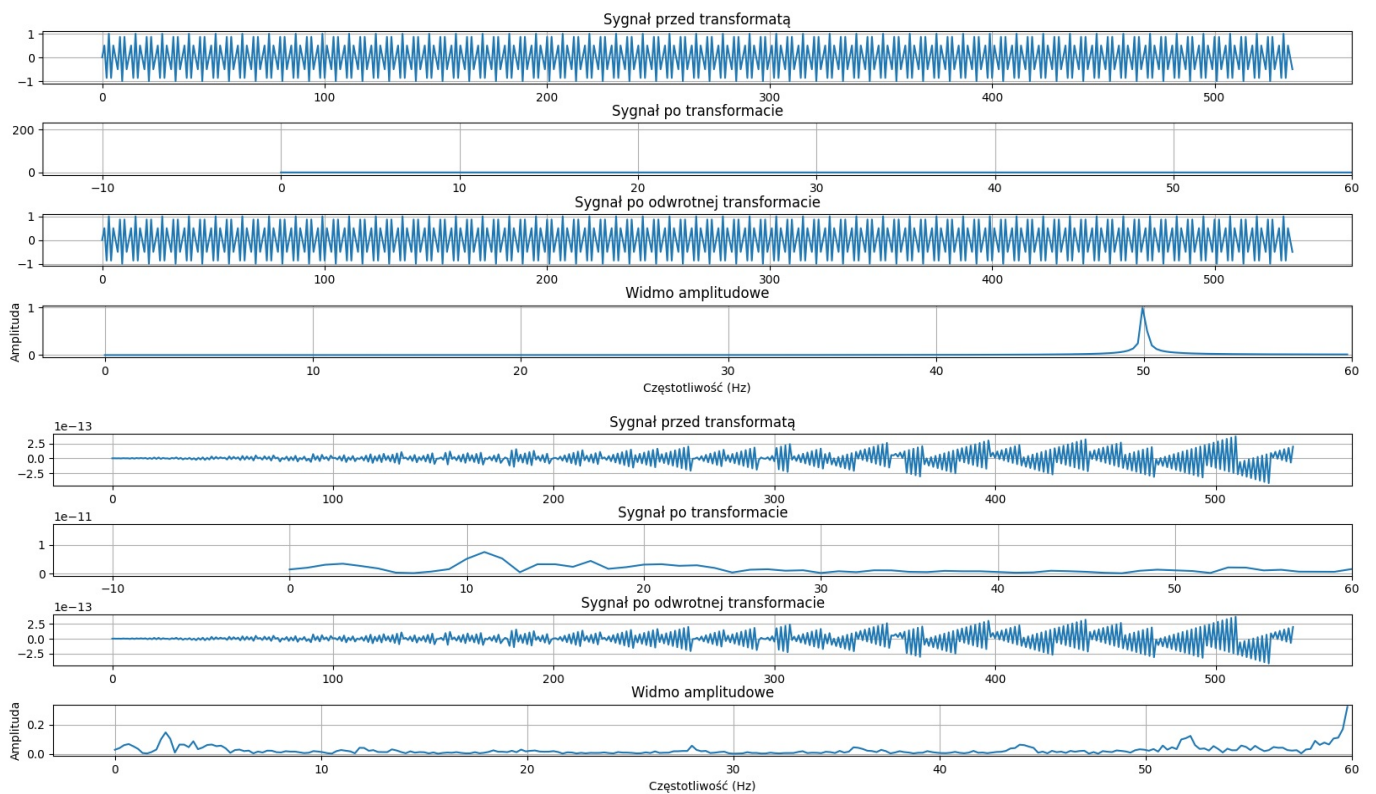


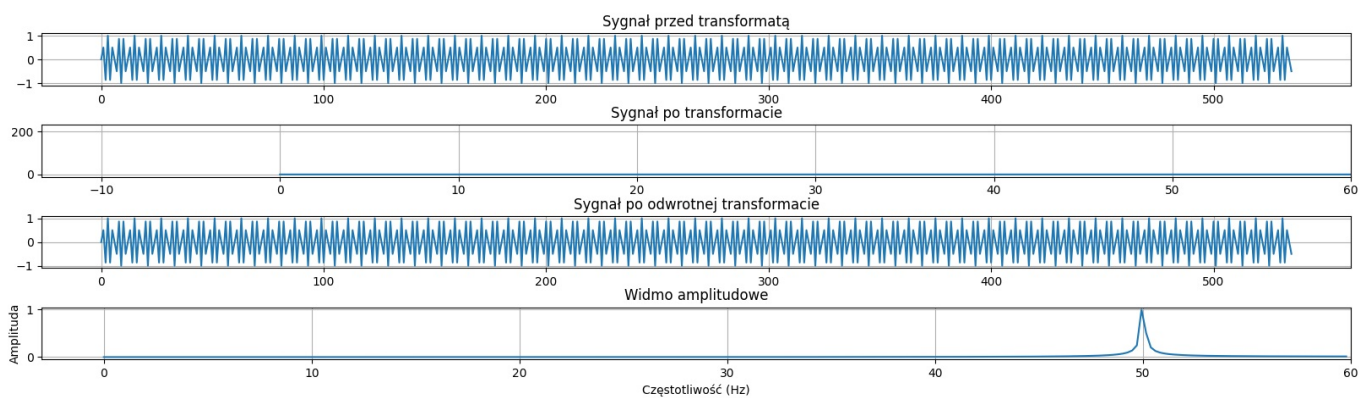
```
In [ ]: fs = 120
freq1 = 50
freq2 = 60
length = 536

t = np.arange(length) / fs
sin_wave1 = np.sin(2 * np.pi * freq1 * t)

t = np.arange(length) / fs
sin_wave2 = np.sin(2 * np.pi * freq2 * t)

displayWave(sin_wave1, fs, length)
displayWave(sin_wave2, fs, length)
```





Ćwiczenie 3.

Celem ćwiczenia jest obserwacja widma sygnału EKG.

1. Wczytać sygnał ecg100.txt i ocenić go wizualnie na wykresie
2. Wyznaczyć jego dyskretną transformatę Fouriera i przedstawić widmo amplitudowe sygnału w funkcji częstotliwości w zakresie $[0, fs/2]$, gdzie fs oznacza częstotliwość próbkowania.
3. Wyznaczyć odwrotną dyskretną transformatę Fouriera ciągu wyznaczonego w punkcie 2 i porównać otrzymany ciąg próbek z pierwotnym sygnałem ecg100 (można wyznaczyć różnicę sygnałów).

```
In [20]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

```
In [ ]: file_name = "./src/"+input("Podaj nazwę pliku z danymi: ")
data_frame = pd.read_csv(file_name, sep="\s+", header=None, engine="python")
```

```
<>:2: SyntaxWarning: invalid escape sequence '\s'
<>:2: SyntaxWarning: invalid escape sequence '\s'
/tmp/ipykernel_24246/4076881288.py:2: SyntaxWarning: invalid escape sequence '\s'
data_frame = pd.read_csv(file_name, sep="\s+", header=None, engine="python")
```

```
0
0      -0.145
1      -0.145
2      -0.145
3      -0.145
4      -0.145
...
649995 -0.075
649996 -0.445
649997 -0.675
649998 -0.765
649999 -1.280
```

[650000 rows x 1 columns]

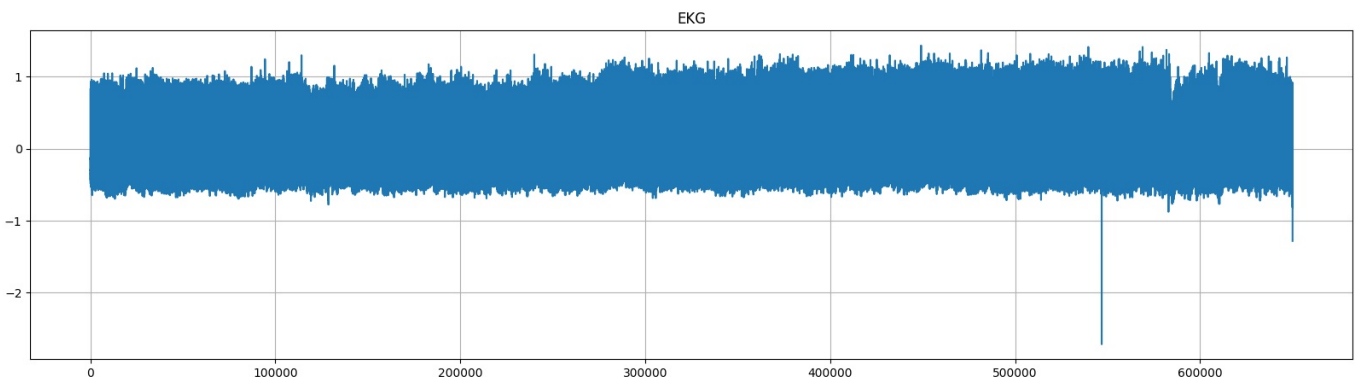
Zadanie nr1

Wczytano plik ekg100.txt zgodnie z treścią ćwiczenia.

W celu oceny wizualnej wykorzystano bibliotekę matplotlib w celu wizualizacji funkcji wynikającej z treści pliku tekstowego.

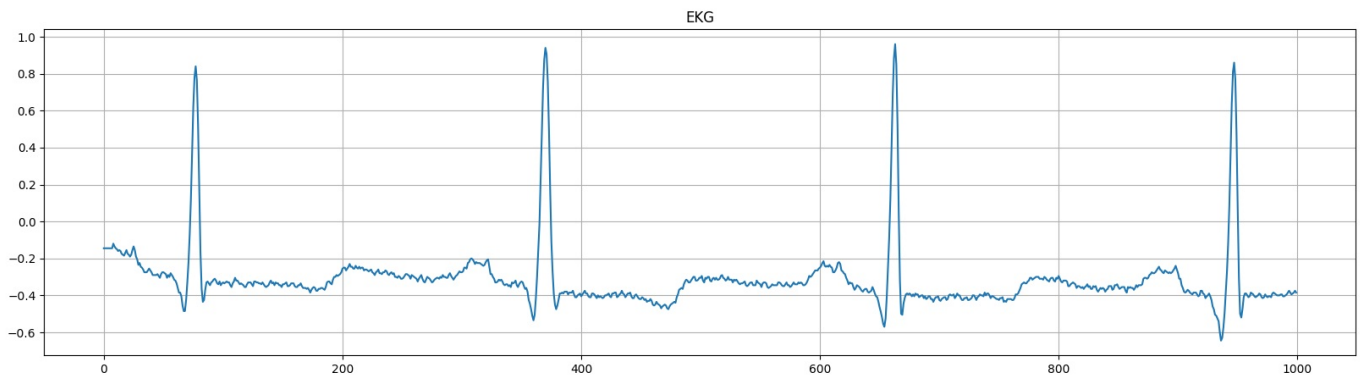
```
In [22]: #Wykres EKG (Na podstawie pobranych danych) w zakresie start : end
def displayEKG(start,end):
    data_frame.columns = ['data']
    new_data = data_frame.iloc[ start :end ].copy()
    plt.figure(figsize=(20,5))
    plt.plot(new_data['data']) #rysowanie wykresu
    plt.grid(True)
    plt.title("EKG")
    plt.show()
```

```
In [23]: displayEKG(0, len(data_frame))
```



Przy próbie wyświetlenia całego sygnału widać że jest on mało czytelny. Można więc ograniczyć wyświetlany sygnał do 1000 próbek

```
In [24]: displayEKG(0,1000)
```



Po ograniczeniu zakresu widać że pobrany sygnał wyświetla się poprawnie i jest czytelny.

Zadanie nr2

Wyznaczono transformantę Fouriera korzystając z biblioteki numpy oraz funkcji fft. Na podstawie transformanty przedstawiono widmo amplitudowe w funkcji częstotliwości.

Korzystając z zewnętrznych źródeł wykonano dodatkowe kroki w celu prawidłowego wyznaczenia widma amplitudowego:

- Wyprowadzono amplitudę transformanty Fouriera,
- Znormalizowano zakres amplitudy aby maksymalną wartością było 1,
- Wygenerowało prawidłowe częstotliwości dla zadanego zakresu,
- Wyznaczono widmo w dodatnim zakresie częstotliwości,

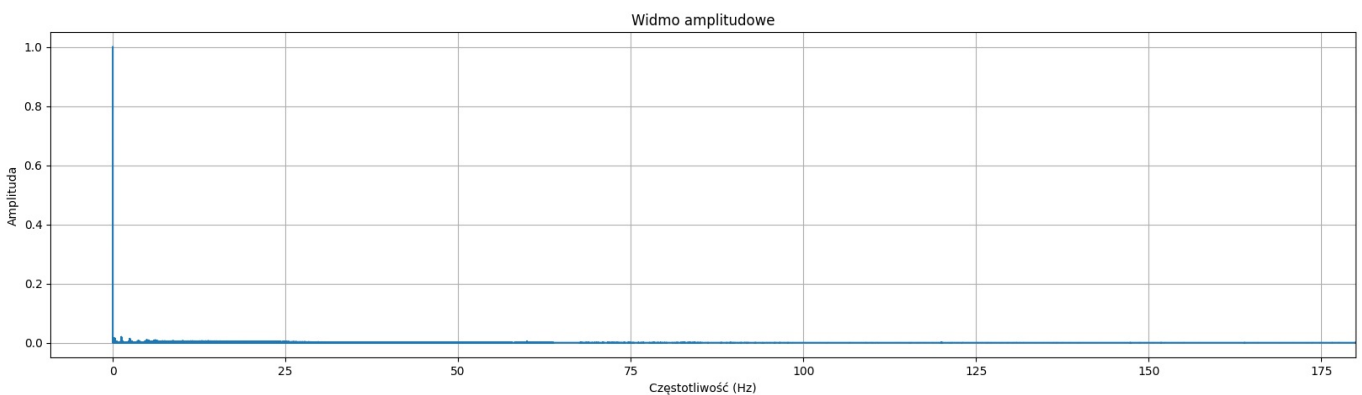
```
In [25]: #Generacja i wyświetlenie widma sygnału
fs = 360
t = len(data_frame)

fourier1 = np.fft.fft(data_frame['data']) #transformata fouriera

#dodatkowe kroki
widmo = np.abs(fourier1)
abs_widmo = widmo / np.max(widmo)
freq = np.fft.fftfreq(t,1/fs)
pos_freq = freq[:len(freq)//2] #częstotliwości w zakresie [0,fs/2]
pos_widmo = abs_widmo[:len(abs_widmo)//2] #widmo amplitudowe (część dodatnia)
#koniec dodatkowych kroków

plt.figure(figsize=(20,5))
plt.plot(pos_freq,pos_widmo)
plt.xlim(right = fs/2)
plt.grid(True)
plt.title("Widmo amplitudowe")
plt.xlabel('Częstotliwość (Hz)')
plt.ylabel('Amplituda')

plt.show()
```



Wyznaczono odwrotną tranformantę Fouriera oraz porównano otrzymany ciąg z pierwotnym sygnałem.

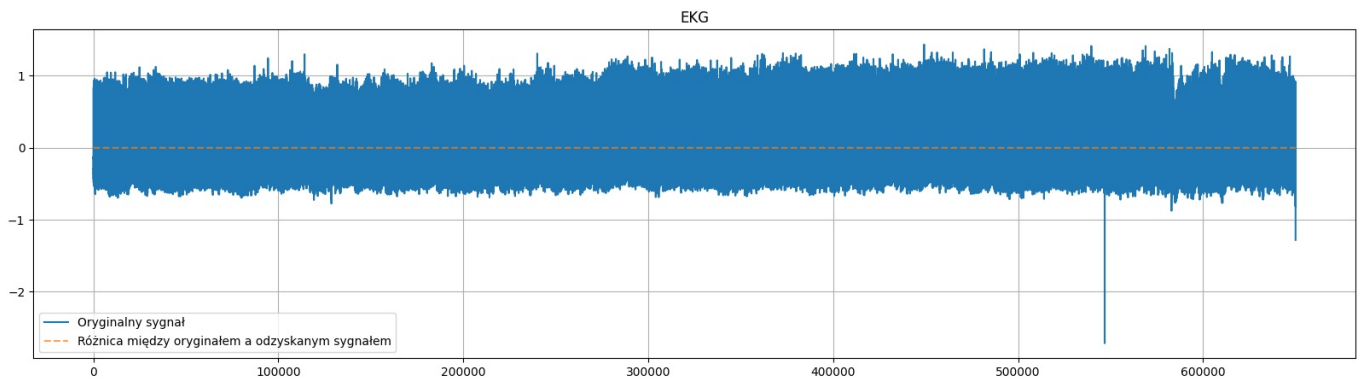
Wyznaczoną różnicę zaznaczono pomarańczową linią przerywaną.

```
In [29]: #Wizualizacja różnicy odwrotnej transformaty(Utworzonej na podstawie widma) i sygnału
def displayFourier(start,end):
    inv_fourier = np.fft.ifft(fourier1).real
    inv_fourier = data_frame['data'] - inv_fourier
    inv_fourier = inv_fourier.values[start:end]
    plt.figure(figsize=(20,5))
```



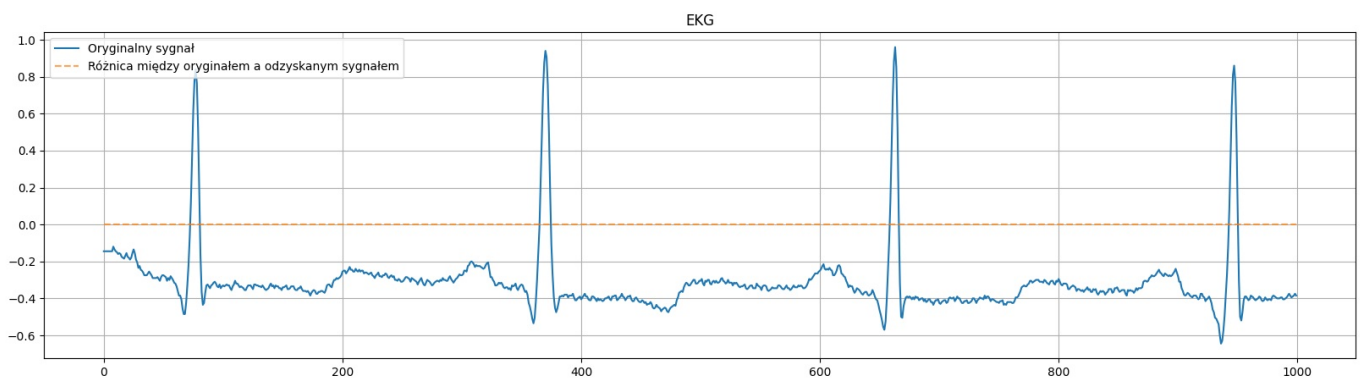
```
plt.plot(data_frame['data'][start:end], label="Oryginalny sygnał")
plt.plot(inv_fourier, linestyle="dashed", alpha=0.7, label="Różnica między oryginałem a odzyskanym sygnałem")
plt.grid(True)
plt.legend()
plt.title("EKG")
plt.show()
```

In [27]: `displayFourier(0,len(data_frame))`



Podobnie jak w poprzednim wypadku cały sygnał jest nie czytelny więc ograniczamy go do 1000 próbek.

In [30]: `displayFourier(0,1000)`



Można łatwo zauważyć że różnica sygnałów nie istnieje lub jest bardzo znikoma i wręcz nieodczytywalna.

Ćwiczenie 4

Celem ćwiczenia jest praktyczne wypróbowanie działania filtrów w celu wyeliminowania niepożądanych zakłóceń z sygnału EKG. Proszę wybrać rodzaj filtra do eksperymentowania, np. Butterwortha lub Czebyszewa. Do filtracji wykorzystać gotowe funkcje z biblioteki `scipy.signal`. Biblioteka posiada również funkcje wspomagające projektowanie filtrów, które można zastosować.

1. Wczytaj sygnał `ekg_noise.txt` i zauważ zakłócenia nałożone na sygnał. Wykreślić częstotliwościową charakterystykę amplitudową sygnału.
2. Zbadaj filtr dolnoprzepustowy o częstotliwości granicznej 60 Hz w celu redukcji zakłóceń pochodzących z sieci zasilającej. Wyznacz parametry filtra, wykreśl jego charakterystykę (zależność tłumienia od częstotliwości), przebieg sygnału po filtracji oraz jego widmo. Można też wyznaczyć różnicę między sygnałem przed i po filtracji i widmo tej różnicy.
3. Zastosuj następnie, do sygnału otrzymanego w punkcie 2, filtr górnoprzepustowy o częstotliwości granicznej 5 Hz w celu eliminacji wpływu linii izoelektrycznej. Sporządź wykresy sygnałów jak w punkcie 2.

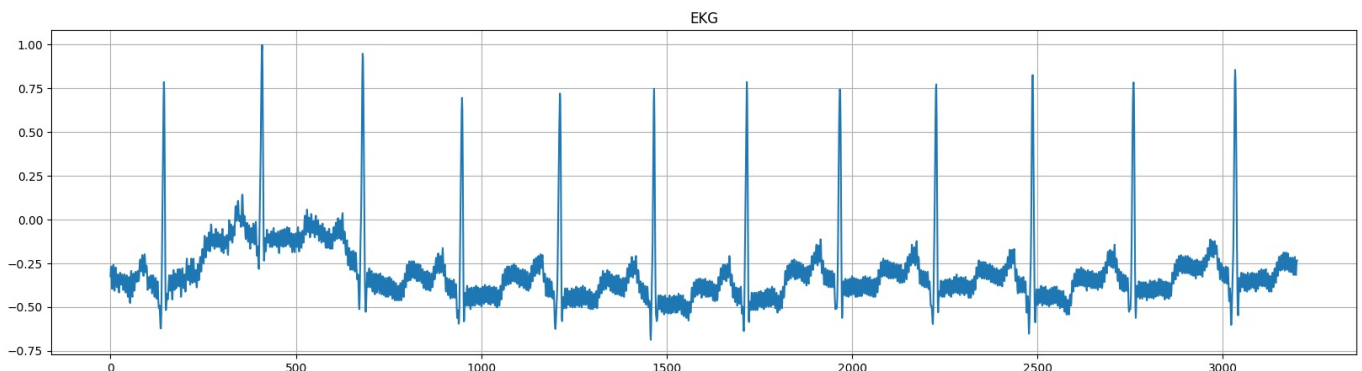
```
In [1]: import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.signal as sci
import os
```

```
In [ ]: if os.name == 'nt':
    file_name = "../src/"+input("Podaj nazwe pliku z danymi: ")
elif os.name == 'posix':
    file_name = "../src//"+input("Podaj nazwe pliku z danymi: ")
else:
    print("Nieznany system")
data_frame = pd.read_csv(file_name, sep="\s+", header=None, engine="python")

print(data_frame)
```

```
In [21]: #Wykres EKG (Na podstawie pobranych danych) w zakresie start : end
num_rows = len(data_frame)
start = input("Początek zakresu (Minimalnie 0): ")
end = input("Koniec zakresu (Maksymalnie "+str( num_rows )+"): ")
if start == "":
    start = 0
if end == "":
    end = num_rows
start = int(start)
end = int(end)

data_frame.columns = ['I', 'data']
new_data = data_frame.iloc[int( start ):int( end )].copy()
font = {'size':20}
plt.figure(figsize=(20,5))
plt.plot(new_data['data'])
plt.grid(True)
plt.title("EKG")
plt.show()
```



Zadanie 1

Wyznaczono widmo amplitudowe, na którego podstawie będą weryfikowane działania filtrów.

```
In [14]: #Generacja i wyświetlenie widma sygnału
fs = 360
t = len(data_frame)

fourier1 = np.fft.fft(data_frame['data'])
```

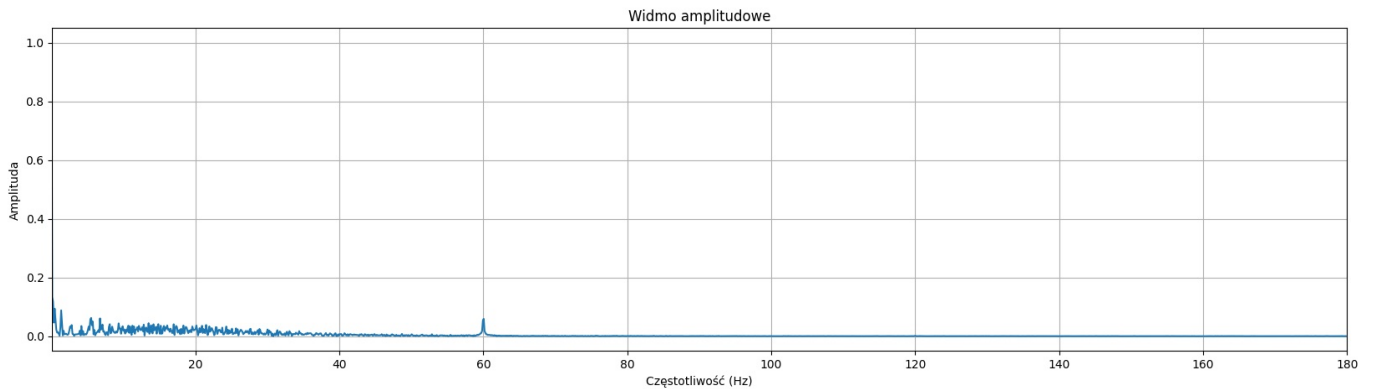
```

widmo = np.abs(fourier1)
abs_widmo = widmo / np.max(widmo)
freq = np.fft.fftfreq(t,1/fs)
pos_freq = freq[:len(freq)//2]
pos_widmo = abs_widmo[:len(abs_widmo)//2]

plt.figure(figsize=(20,5))
plt.plot(pos_freq,pos_widmo)
plt.xlim(left = 0.1, right = fs/2)
plt.grid(True)
plt.title("Widmo amplitudowe")
plt.xlabel('Częstotliwość (Hz)')
plt.ylabel('Amplituda')

plt.show()

```



Zadanie 2

Przedstawiono graficznie działanie filtru dolnoprzepustowego o częstotliwości granicznej 60Hz

Następnie przedstawiono dane pierwotne sygnału oraz dane które zostały przetworzone przy pomocy filtru na jednym wykresie w celu łatwego zidentyfikowania różnic w przebiegu sygnału

Wyświetlono również widmo przetworzonego sygnału w celu dalszej analizy

```

In [9]: #Zastosowanie dolnoprzepustowego filtru Butterwortha
order = 6
fs = 360
cutoff = 60

b,a = sci.butter(order,cutoff,fs=fs,btype='low',analog=False)

w, h = sci.freqz(b, a, fs=fs, worN=8000)

plt.figure(figsize=(20,5))
plt.subplot(3, 1, 1)
plt.plot(w, np.abs(h), 'b')
plt.plot(cutoff, 0.5*np.sqrt(2), 'ko')
plt.axvline(cutoff, color='k')
plt.xlim(0, 0.5*fs)
plt.title("Charakterystyka filtru")
plt.xlabel('Częstotliwość [Hz]')
plt.grid()

y = sci.lfilter(b, a, data_frame['data'])

plt.subplot(3, 1, 2)
plt.plot(data_frame['data'][start:end] , 'b-', label='Dane')
plt.plot( y[start:end], 'g-', linewidth=2, label='Przefiltrowane dane')
plt.legend()
plt.xlabel('Czas [sec]')
plt.grid()

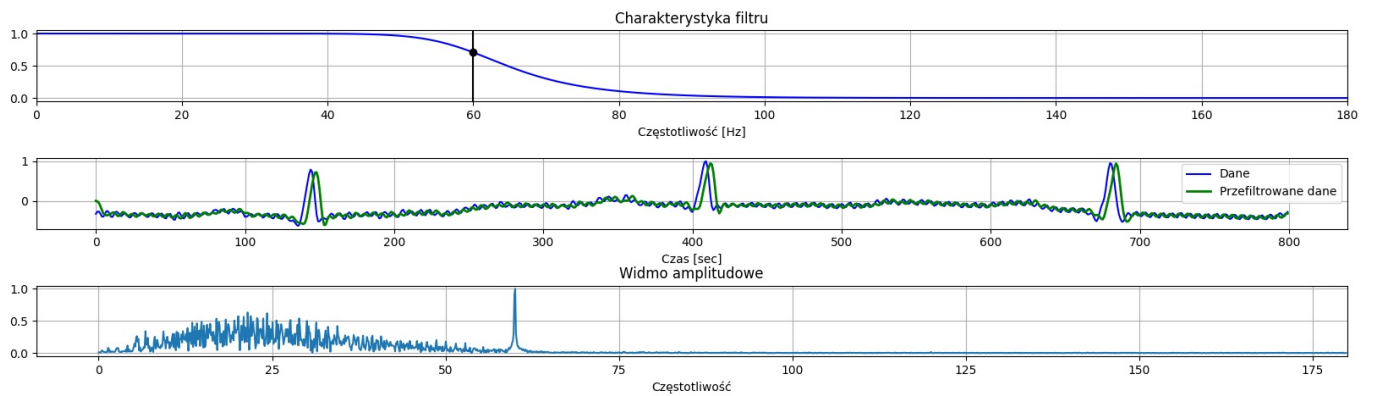
fourier1 = np.fft.fft(data_frame['data']-y)

widmo = np.abs(fourier1)
abs_widmo = widmo / np.max(widmo)
freq = np.fft.fftfreq(t,1/fs)
pos_freq = freq[:len(freq)//2]
pos_widmo = abs_widmo[:len(abs_widmo)//2]

plt.subplot(3,1,3)
plt.plot(pos_freq,pos_widmo)
plt.xlim(right = fs/2)
plt.grid(True)
plt.title("Widmo amplitudowe")
plt.xlabel('Częstotliwość')

```

```
plt.subplots_adjust(hspace=0.8)
plt.show()
```



Zadanie 3

Po wyznaczeniu przebiegu sygnału po zastosowaniu filtru dolnoprzepustowego, wyznaczono charakterystykę filtru

górnoprzepustowego o wartości granicznej 5Hz

Sporządzono wykres, porównujący różnice z przebiegiem z zadania 2 oraz zwizualizowano widmo amplitudowe po zastosowaniu drugiego filtra.

```
In [20]: #Zastosowanie górnoprzepustowego filtru Butterwortha
order = 3
fs = 360
cutoff = 5

b,a = sci.butter(order,cutoff,fs=fs,btype='hp',analog=False)

w, h = sci.freqz(b, a, fs=fs, worN=8000)
hy = sci.lfilter(b, a, y)

plt.figure(figsize=(20,5))

plt.subplot(3, 1, 1)
plt.plot(w, np.abs(h), 'b')
plt.plot(cutoff, 0.5*np.sqrt(2), 'ko')
plt.axvline(cutoff, color='k')
plt.xlim(0, 0.5*fs)
plt.title("Charakterystyka filtru")
plt.xlabel('Częstotliwość [Hz]')
plt.grid()

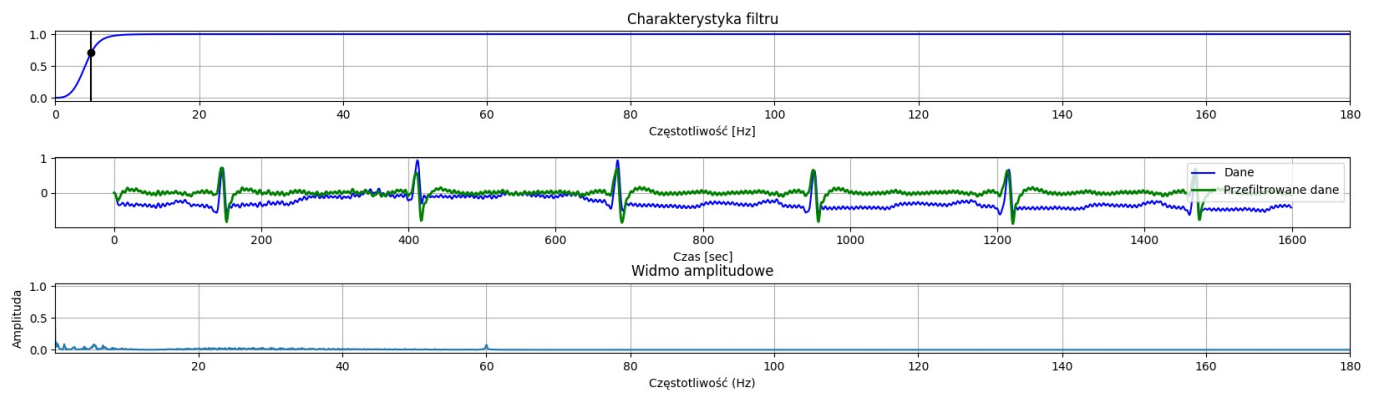
plt.subplot(3, 1, 2)
plt.plot(y[start:end], 'b-', label='Dane')
plt.plot(hy[start:end], 'g-', linewidth=2, label='Przefiltrowane dane')
plt.legend()
plt.xlabel('Czas [sec]')
plt.grid()

fourier1 = np.fft.fft(data_frame['data']-hy)

widmo = np.abs(fourier1)
abs_widmo = widmo / np.max(widmo)
freq = np.fft.fftfreq(t,1/fs)
pos_freq = freq[:len(freq)//2]
pos_widmo = abs_widmo[:len(abs_widmo)//2]

plt.subplot(3,1,3)
plt.plot(pos_freq,pos_widmo)
plt.xlim(left= 0.1, right = fs/2)
plt.grid(True)
plt.title("Widmo amplitudowe")
plt.xlabel('Częstotliwość (Hz)')
plt.ylabel('Amplituda')

plt.subplots_adjust(hspace=0.8)
plt.show()
```



Otrzymany sygnał końcowy znacząco różni się od początkowego, ze względu na nałożenie filtrów.
Najlepiej różnicę widać patrząc na widma amplitudowe kolejnych wykresów.