Laboratorium nr 1

April 26, 2022

1 Laboratorium nr 1 - Analiza sygnału EKG

Autorzy:

Michał Droń 248832

Filip Panek 252794

1.1 1. Platforma testowa

1.1.1 Konfiguracja

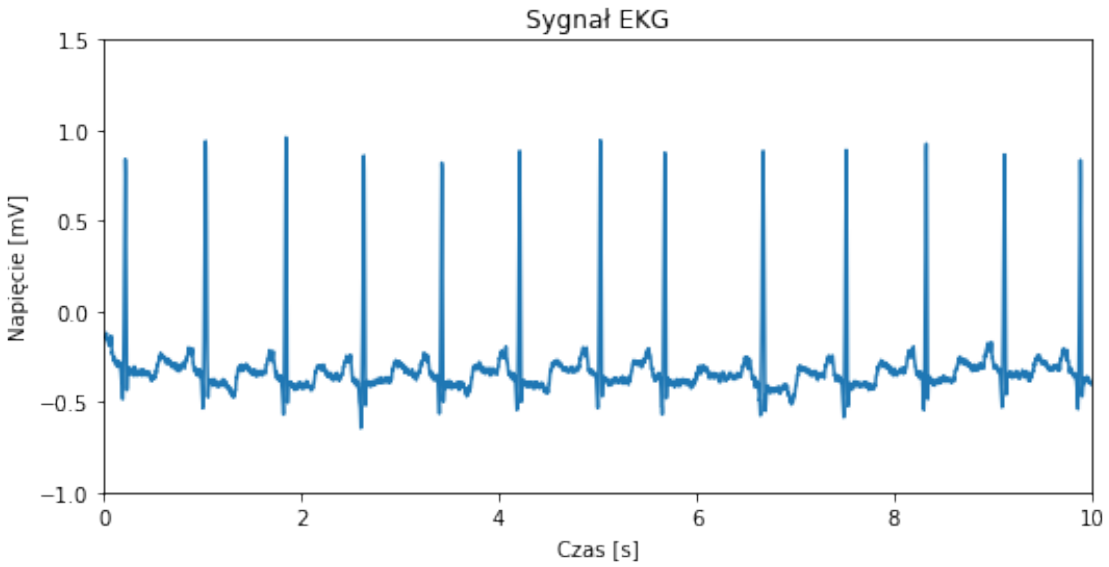
[2]:
```
%matplotlib inline

import matplotlib.pyplot as plt
import numpy as np

# rozmiar wykresu
plt.rcParams["figure.figsize"] = (8.5,4)

# funkcja do tworzenia wykresów
def drawGraph(number, title):
    fig = plt.figure(number)
    ax = fig.add_subplot(111)
    ax.set_title(title)
    return ax

# wyciszenie powiadomień o odrzucaniu części zespołowej
import warnings
warnings.filterwarnings('ignore', message='.*ComplexWarning.*')
```

1.1.2 Pojedynczy wykres

Komórka poniżej pokazuje dane z pliku `ekg100.txt`.

[3]:
```
# ładowanie danych
data = np.fromfile('data/ekg100.txt', sep=' ')

# podane Fs = 360 Hz
SAMPLING_RATE = 360

X_range = np.arange(stop=(len(data)))
X = X_range / SAMPLING_RATE

ax1 = drawGraph(1, "Sygnał EKG")
ax1.set_xlabel("Czas [s]")
ax1.set_ylabel("Napięcie [mV]")
# zakresy
ax1.set_xlim([0, 10])
ax1.set_ylim([-1, 1.5])
ax1.plot(X, data)
```

[3]:
```
[<matplotlib.lines.Line2D at 0x1180c8b20>]
```

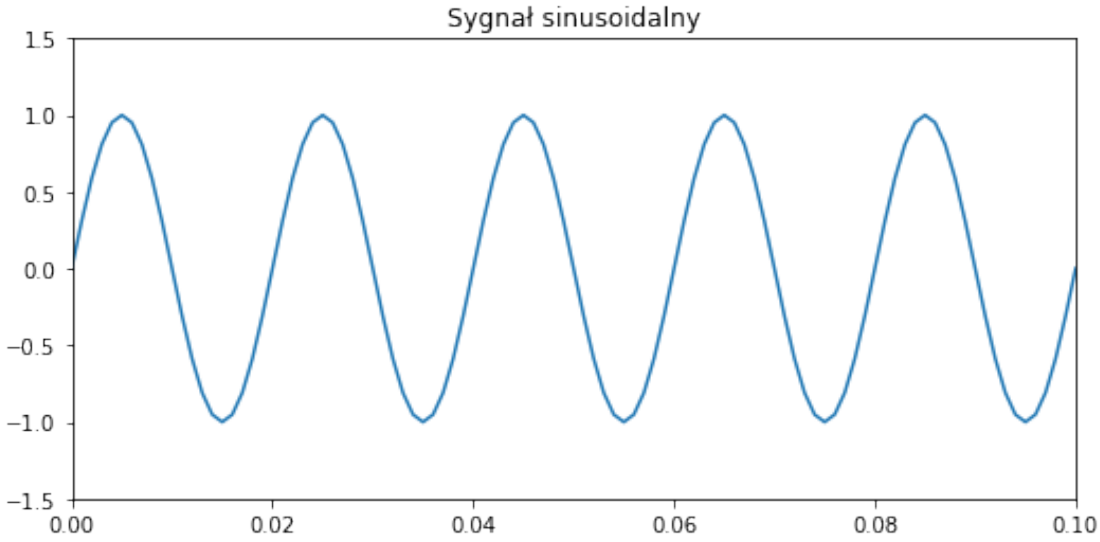


1.2 2. Analiza sygnału okresowego

1.2.1 Sinusoida dla Fs = 1000 Hz

[4]:
```
FS = 1000
F = 50
N = 65536
SAMPLES = np.linspace(0.0, N/FS, N)
```

```
SINE = np.sin(2*np.pi*F*SAMPLES)

fig2 = plt.figure(2)
ax2 = fig2.add_subplot(111)
ax2.set_title("Sygnał sinusoidalny")
ax2.set_xlim([0, 0.1])
ax2.set_ylim([-1.5, 1.5])
ax2.plot(SAMPLES, SINE)

fig2.canvas.draw()
```



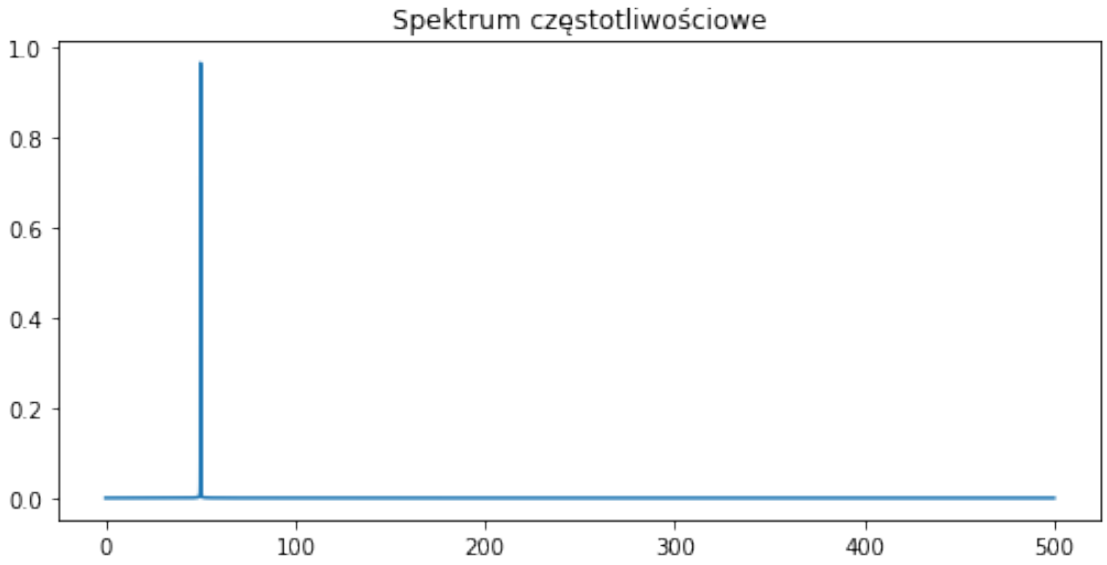
Powyżej wygenerowano sygnał sinusoidalny.

1.2.2 Transformata Fouriera sygnału

Wykorzystujemy tylko połowę transformaty, ponieważ druga połowa jest odbiciem lustrzanym. Oprócz tego wykorzystujemy wartość bezwzględną aby poprawnie obliczyć częstotliwość

[5]:
```
FFT = np.fft.fft(SINE)
Y = 2/N * np.abs(FFT[:N//2])
X = np.linspace(0.0, FS/2, N//2)

ax3 = drawGraph(3, "Spektrum częstotliwościowe")
ax3.plot(X, Y)
```

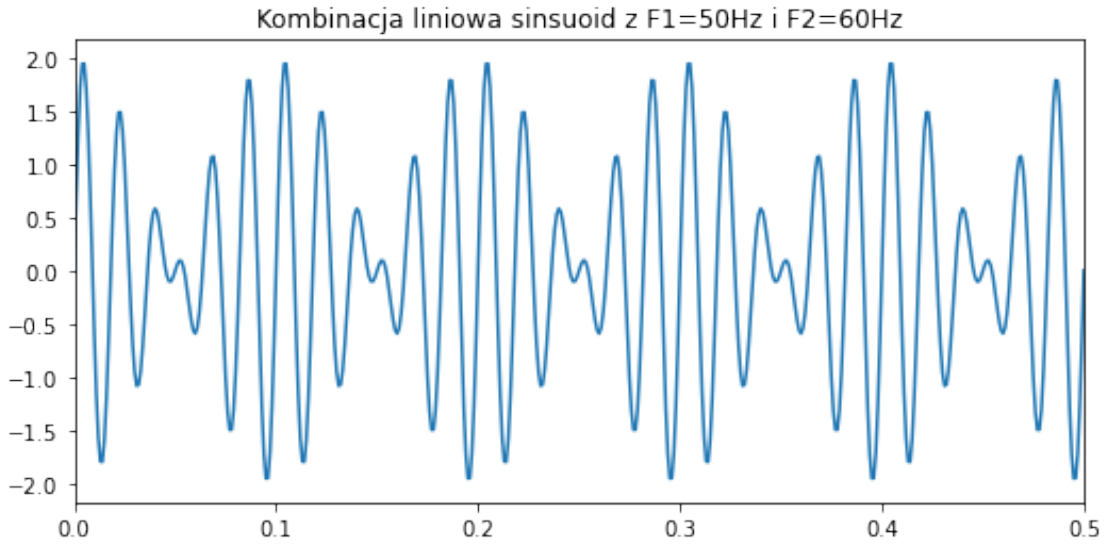
[5]:
```
[<matplotlib.lines.Line2D at 0x118224fa0>]
```

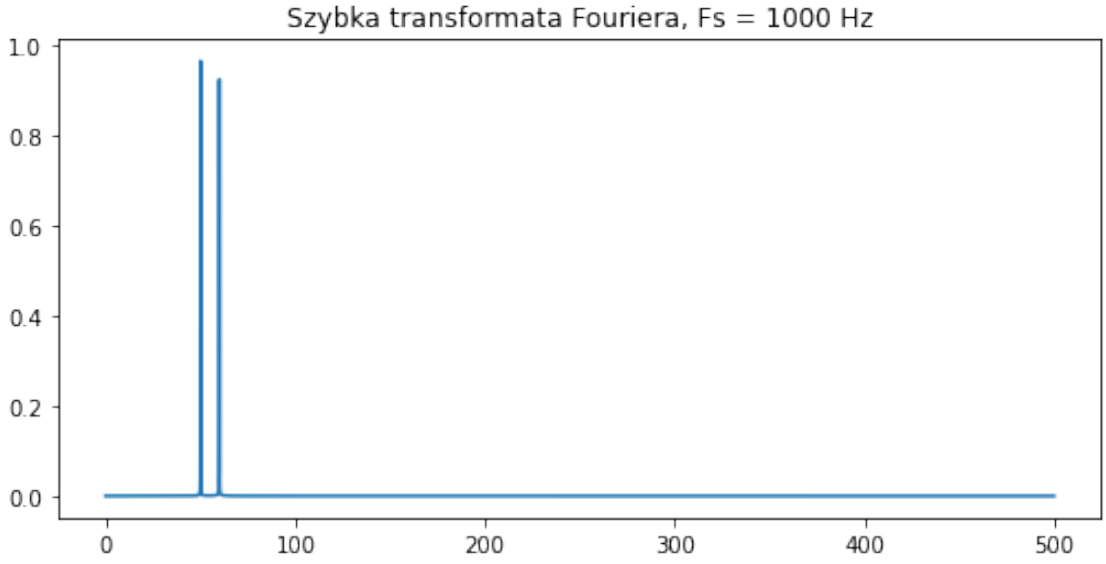


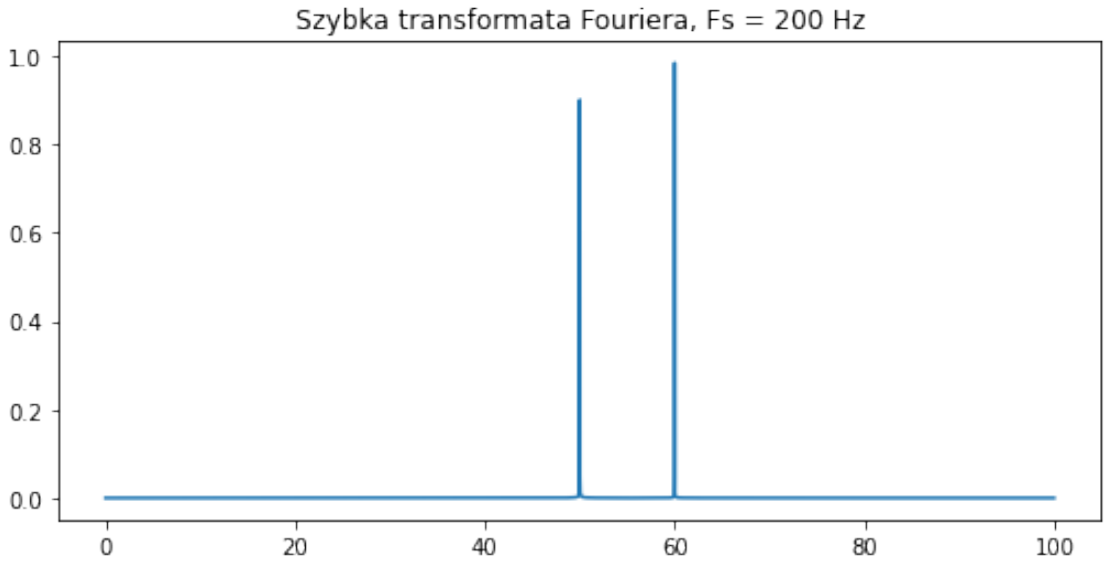
1.2.3 Kombinacja liniowa sinusoid

[6]:
```
F1 = 50
F2 = 60
SINE_2 = np.sin(2*np.pi*F1*SAMPLES) + np.sin(2*np.pi*F2*SAMPLES)

ax4 = drawGraph(4, "Kombinacja liniowa sinsuoid z F1=50Hz i F2=60Hz")
ax4.set_xlim(0, 0.5)
ax4.plot(SAMPLES, SINE_2)

FFT_2 = np.fft.fft(SINE_2)
Y = 2/N * np.abs(FFT_2[:N//2])
ax5 = drawGraph(5, "Szybka transformata Fouriera, Fs = 1000 Hz")
ax5.plot(X, Y)

# i dla Fs=200Hz
FS = 200
SAMPLES_2 = np.linspace(0.0, N/FS, N)
SINE_3 = np.sin(2*np.pi*F1*SAMPLES_2) + np.sin(2*np.pi*F2*SAMPLES_2)
FFT_3 = np.fft.fft(SINE_3)
Y = 2/N * np.abs(FFT_3[:N//2])
X = np.linspace(0.0, FS/2, N//2)
ax6 = drawGraph(6, "Szybka transformata Fouriera, Fs = 200 Hz")
ax6.plot(X, Y)
```

[6]: [<matplotlib.lines.Line2D at 0x11831a670>]


Kombinacja liniowa sinsuoid z F1=50Hz i F2=60Hz
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.1
0.2
0.3
0.4
0.5


Szybka transformata Fouriera, Fs = 1000 Hz
1.0
0.8
0.6
0.4
0.2
0.0
0
100
200
300
400
500



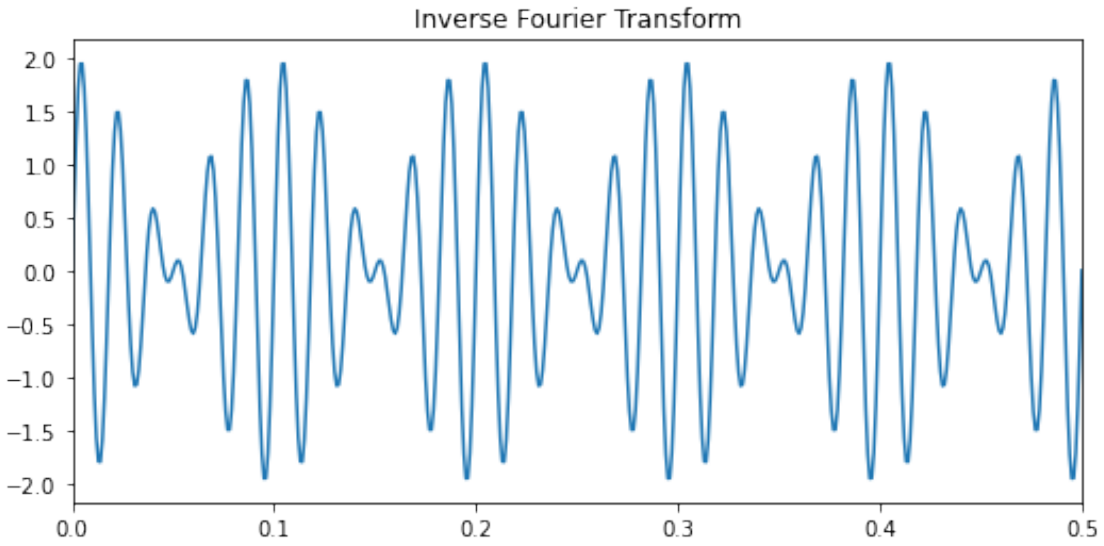
Jak widać na wykresach, różnią się one jedynie dziedziną. Wyniki jeśli nie są identyczne to są mocno do siebie zbliżone.

1.2.4 Odwrotne transformaty Fouriera

[7]:
```
INVERSE  = np.fft.ifft(FFT_2)
ax7 = drawGraph(7, "Inverse Fourier Transform")
ax7.set_xlim(0, 0.5)
ax7.plot(SAMPLES, INVERSE)
```

```
/Users/droniu/.pyenv/versions/3.9.7/envs/cpsio-lab/lib/python3.9/site-
packages/matplotlib/cbook/__init__.py:1298: ComplexWarning: Casting complex
values to real discards the imaginary part
  return np.asarray(x, float)
```

[7]:
```
[<matplotlib.lines.Line2D at 0x1183c8e50>]
```



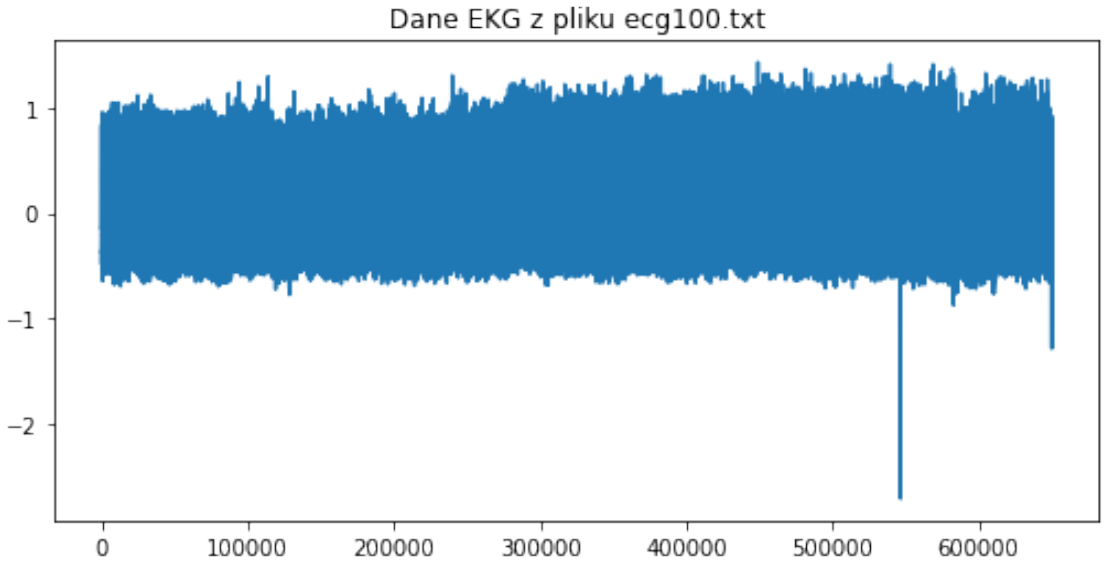
Jak można zauważyć, odwrotna transformata Fouriera pozwala nam odtworzyć sygnał mocno zbliżony do tego z wykresu nr 4, różnice są minimalne.

1.3 3. Analiza sygnału EKG

1.3.1 Cały sygnał

[8]:
```
ax8 = drawGraph(8, "Dane EKG z pliku ecg100.txt")
ax8.plot(data)
```

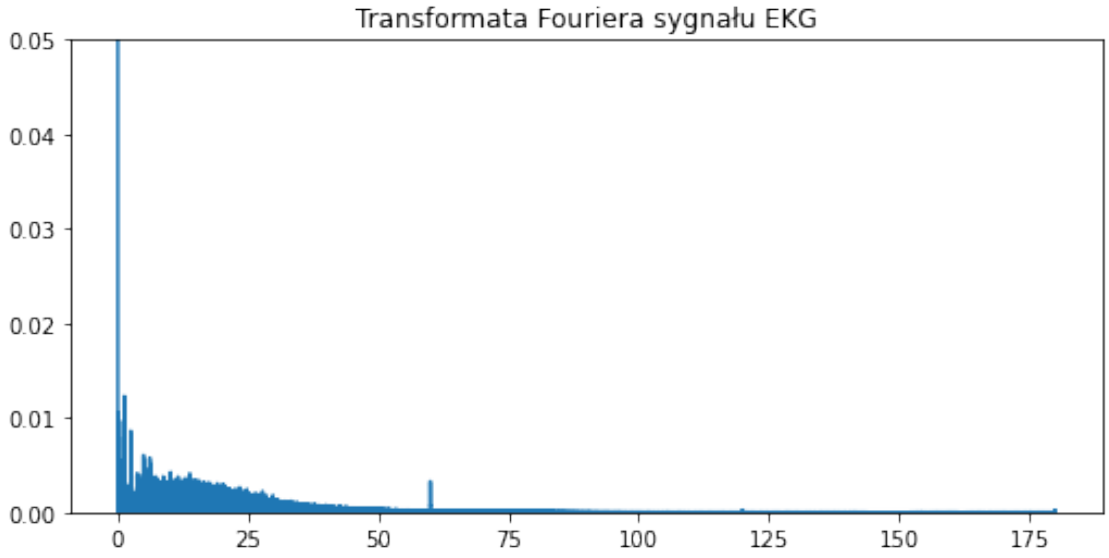
[8]:
```
[<matplotlib.lines.Line2D at 0x1184607f0>]
```



1.3.2 Transformata Fouriera

[9]:
```
FS_ECG = 360
N_ECG = len(data)
FFT_ECG = np.fft.fft(data)

X = np.linspace(0.0, FS_ECG/2, N_ECG//2)
Y = 2/N_ECG * np.abs(FFT_ECG[:N_ECG//2])
ax9 = drawGraph(9, "Transformata Fouriera sygnału EKG")

ax9.set_ylim(0, 0.05)
ax9.plot(X, Y)
```

[9]:
```
[<matplotlib.lines.Line2D at 0x1184bdaf0>]
```



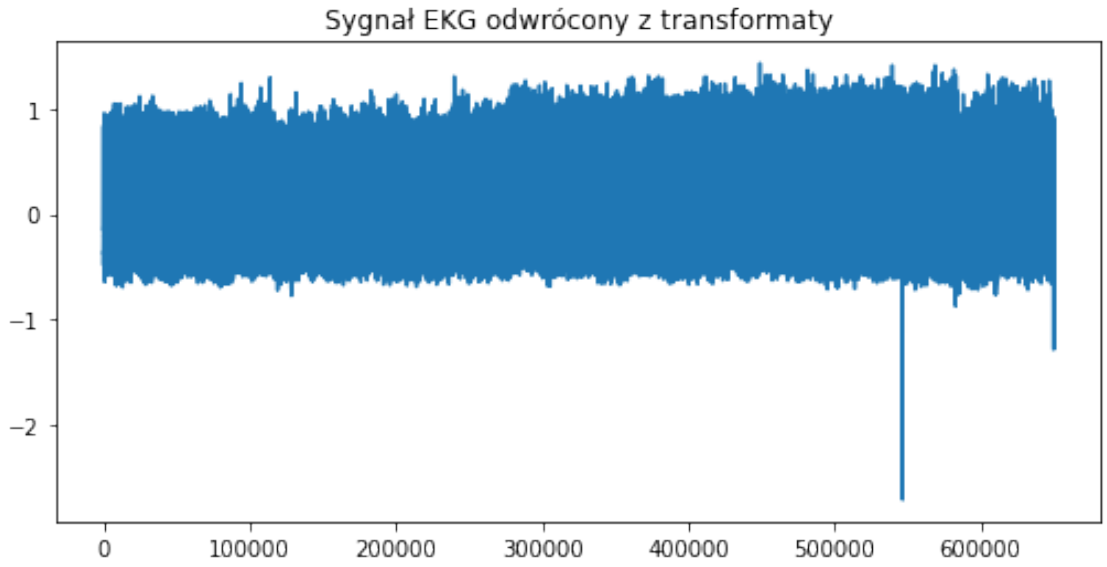
Jak można zauważyć, sygnał EKG składa się z częstotliwości głównie pomiędzy 0 a 50 Hz. Widać wyraźny spike na samym początku oraz w okolicach 60 Hz - pierwszy wynika najprawdoopdobniej z “pływania” sygnału, natomiast drugi z zakłóceń z sieci elektrycznej

1.3.3 Odwrotna transformata Fouriera

[10]:
```
INVERTED_ECG = np.fft.ifft(FFT_ECG)
ax10 = drawGraph(10, "Sygnał EKG odwrócony z transformaty")
ax10.plot(INVERTED_ECG)
```

```
/Users/droniu/.pyenv/versions/3.9.7/envs/cpsio-lab/lib/python3.9/site-
packages/matplotlib/cbook/__init__.py:1298: ComplexWarning: Casting complex
values to real discards the imaginary part
  return np.asarray(x, float)
```

[10]:
```
[<matplotlib.lines.Line2D at 0x1185236a0>]
```

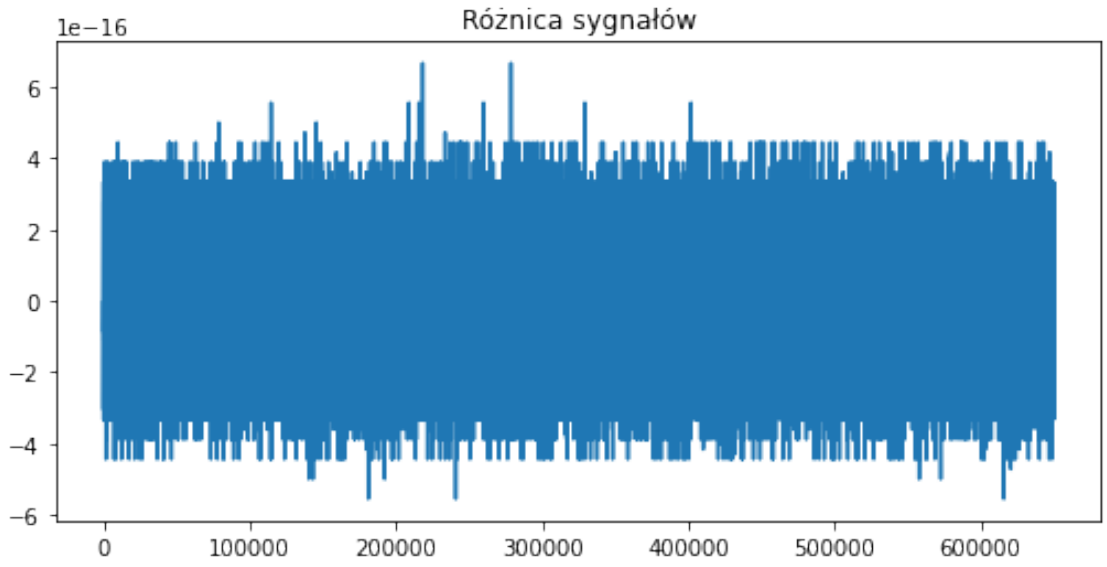


1.3.4 Różnica w sygnałach

[11]:
```
SIGNAL_DIFF = np.subtract(data, INVERTED_ECG)
ax11 = drawGraph(11, "Różnica sygnałów")
ax11.plot(SIGNAL_DIFF)
```

```
/Users/droniu/.pyenv/versions/3.9.7/envs/cpsio-lab/lib/python3.9/site-
packages/matplotlib/cbook/__init__.py:1298: ComplexWarning: Casting complex
values to real discards the imaginary part
  return np.asarray(x, float)
```

[11]:
```
[<matplotlib.lines.Line2D at 0x118582f40>]
```

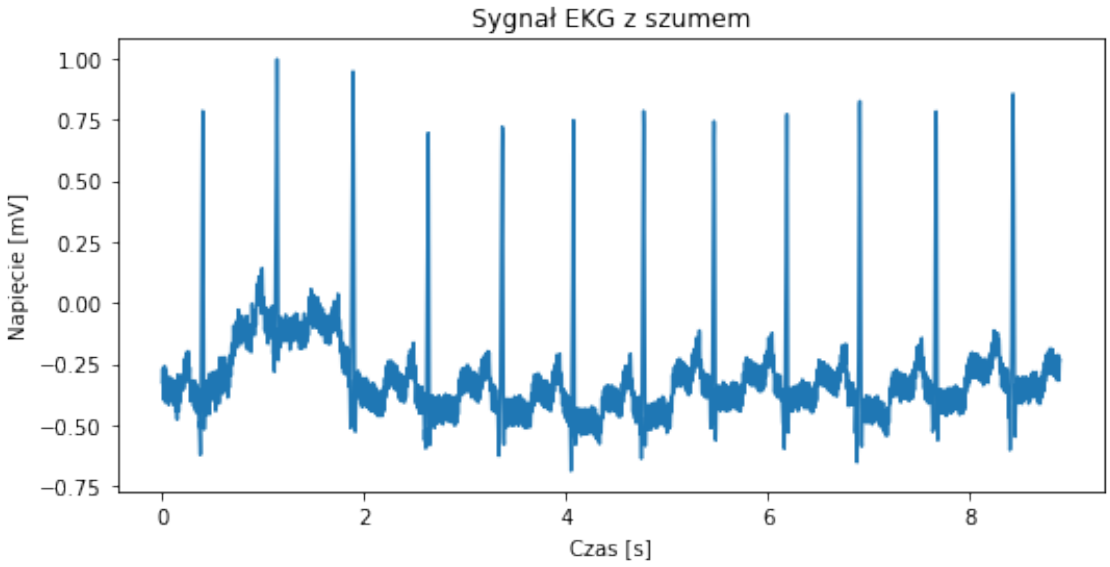


Na wykresie można zauważyć że różnica w sygnałach jest nieistotna statycznie, rzędu 10-16

1.4 4. Filtrowanie sygnału EKG

[12]:
```
data = np.genfromtxt('./data/ekg_noise.txt')
time, values = data.transpose()
ax12 = drawGraph(12, "Sygnał EKG z szumem")
ax12.plot(time, values)
ax12.set_xlabel("Czas [s]")
ax12.set_ylabel("Napięcie [mV]")
```

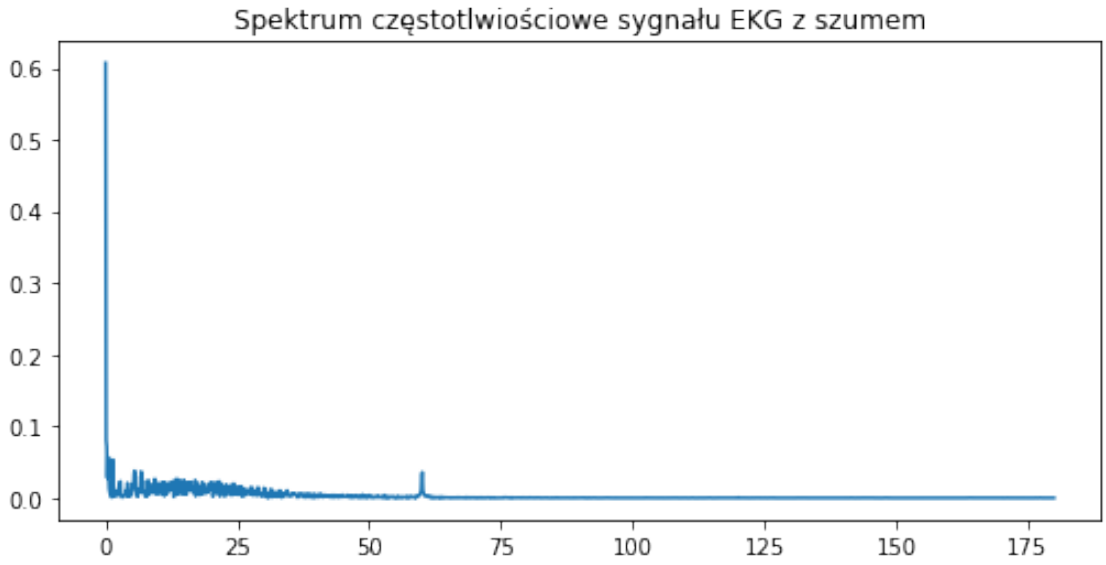
[12]:
```
Text(0, 0.5, 'Napięcie [mV]')
```



1.4.1 Spektrum częstotliwościowe

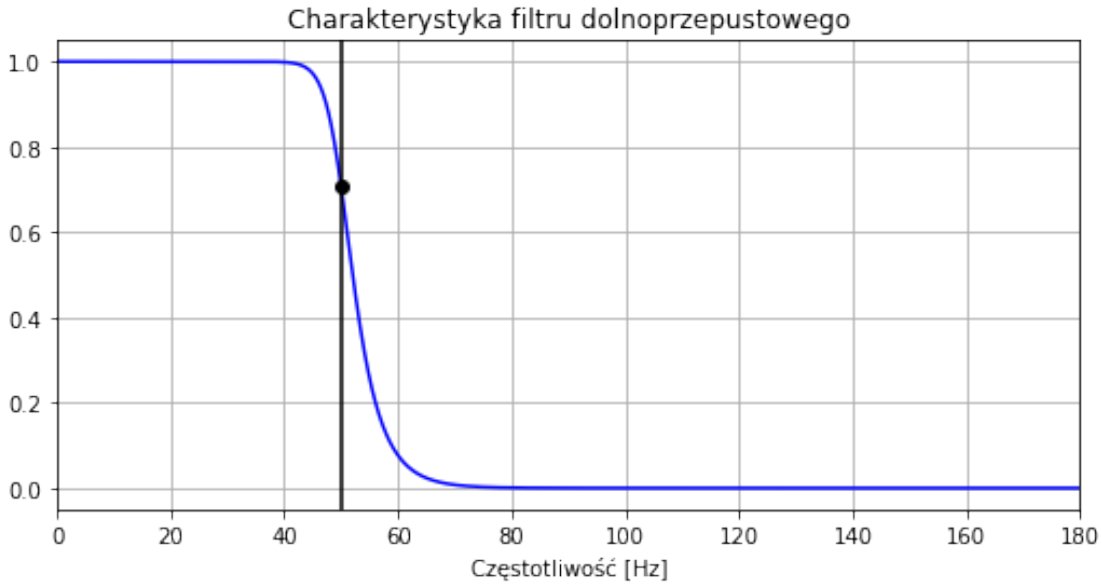
[13]:
```
# częstotliwość próbkowania (FS) pozostaje taka sama
N_ECG = len(values)
FFT_ECG = np.fft.fft(values)

X = np.linspace(0.0, FS_ECG/2, N_ECG//2)
Y_pre = 2/N_ECG * np.abs(FFT_ECG[:N_ECG//2])
ax13 = drawGraph(13, "Spektrum częstotlwiościowe sygnału EKG z szumem")
ax13.plot(X, Y_pre)
```

[13]:
```
[<matplotlib.lines.Line2D at 0x11867fac0>]
```



1.4.2 Filtr dolnoprzepustowy Butterwortha

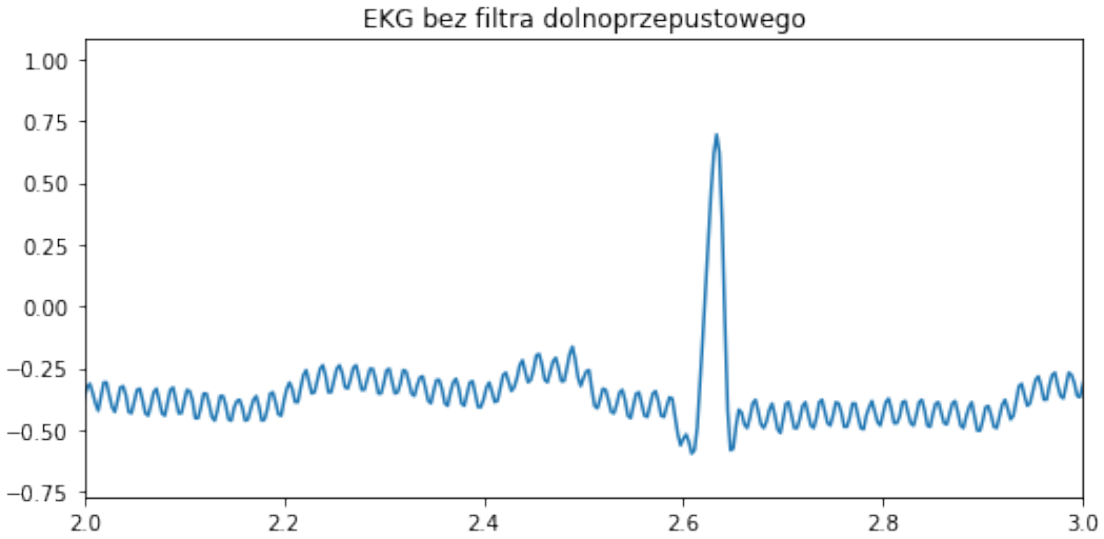
[14]:
```
from scipy.signal import butter, lfilter, freqz

def butter_lowpass(cutoff, fs, order=5):
    return butter(order, cutoff, fs=fs, btype='low', analog=False)

def butter_lowpass_filter(data, cutoff, fs, order=5):
    b, a = butter_lowpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

# Wymagania filtru
order = 12
cutoff = 50  # częstotliwość odcięcia, Hz

# Zmienne filtra
b, a = butter_lowpass(cutoff, FS_ECG, order)
# Wykres
w, h = freqz(b, a, fs=FS_ECG, worN=8000)
ax14 = drawGraph(14, "Charakterystyka filtru dolnoprzepustowego")
ax14.plot(w, np.abs(h), 'b')
ax14.plot(cutoff, 0.5*np.sqrt(2), 'ko')
ax14.axvline(cutoff, color='k')
ax14.set_xlim(0, 0.5*FS_ECG)
ax14.set_xlabel('Częstotliwość [Hz]')
ax14.grid()
```

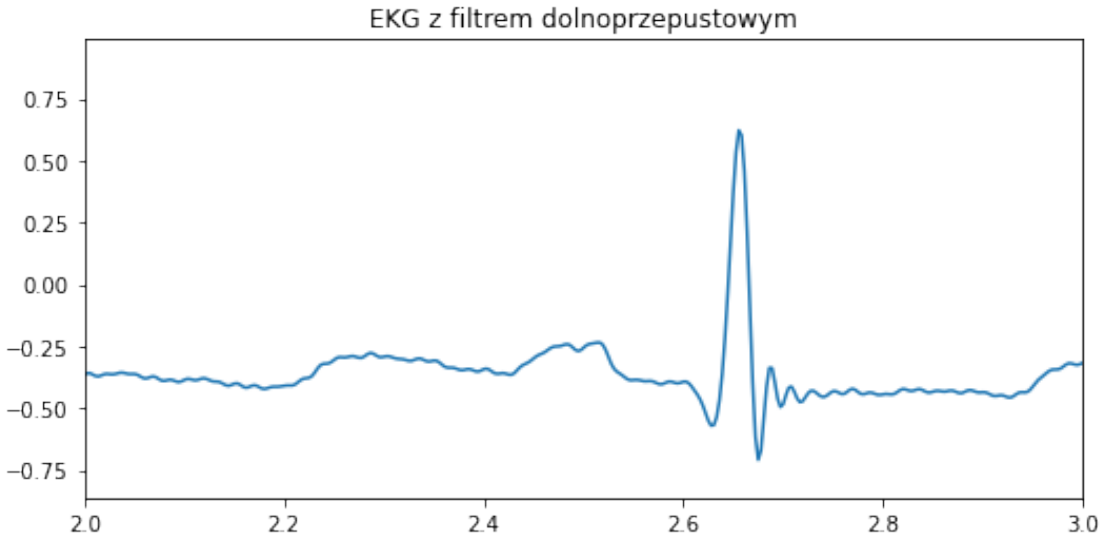


1.4.3 Użycie filtra

[15]:
```
Y_filtered = butter_lowpass_filter(values, cutoff, FS_ECG, order)
ax15 = drawGraph(15, "EKG bez filtra dolnoprzepustowego")
ax15.plot(time, values)
ax15.set_xlim(2, 3)

ax16 = drawGraph(16, "EKG z filtrem dolnoprzepustowym")
ax16.plot(time, Y_filtered)
ax16.set_xlim(2, 3)
```

[15]: (2.0, 3.0)

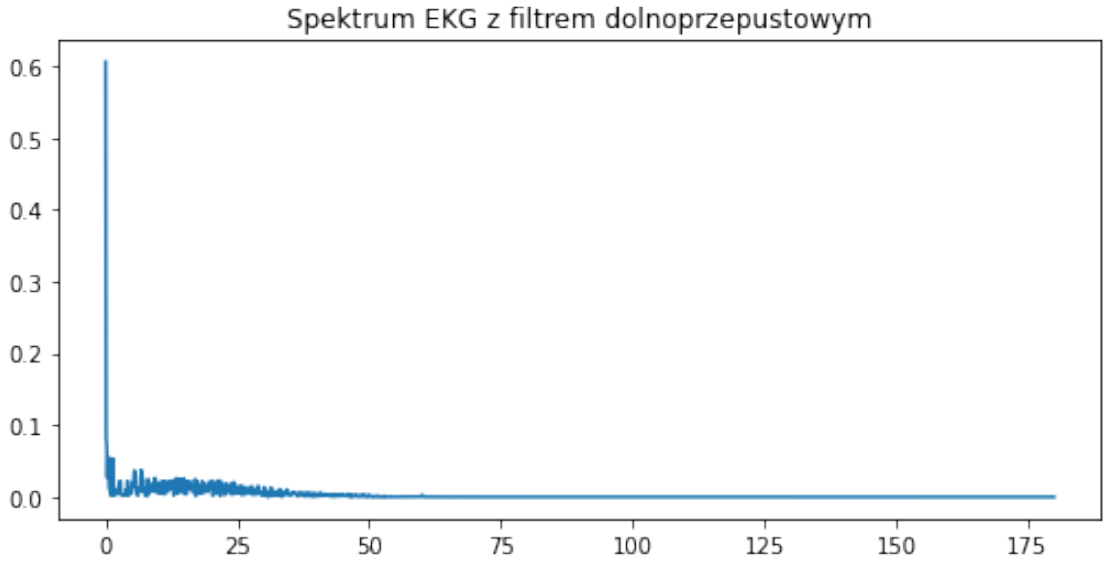




W tym przypadku widać wyraźną redukcję zakłóceń wynikających z częstotliwości w sieci elektrycznej.

[16]:
```
FFT_ECG = np.fft.fft(Y_filtered)

X = np.linspace(0.0, FS_ECG/2, N_ECG//2)
Y_post = 2/N_ECG * np.abs(FFT_ECG[:N_ECG//2])

ax17 = drawGraph(17, "Spektrum EKG z filtrem dolnoprzepustowym")
ax17.plot(X, Y_post)
```

[16]:
```
[<matplotlib.lines.Line2D at 0x125fa8fd0>]
```



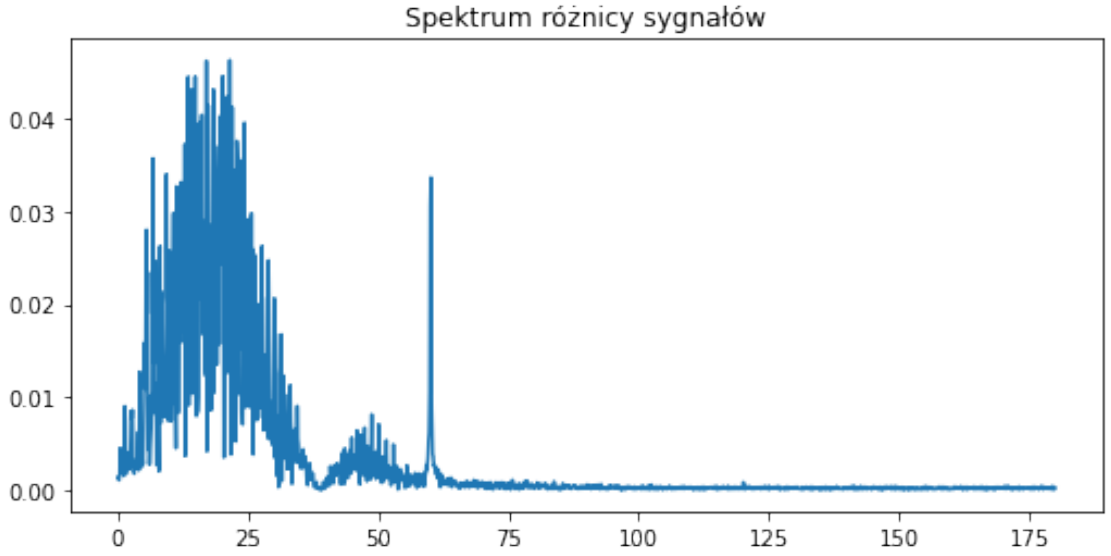
1.4.4 Różnica sygnałów

[17]:
```
Y_diff = np.subtract(values, Y_filtered)
ax18 = drawGraph(18, "Różnica sygnałów")
ax18.plot(time, Y_diff)
```

[17]:
```
[<matplotlib.lines.Line2D at 0x12601d220>]
```

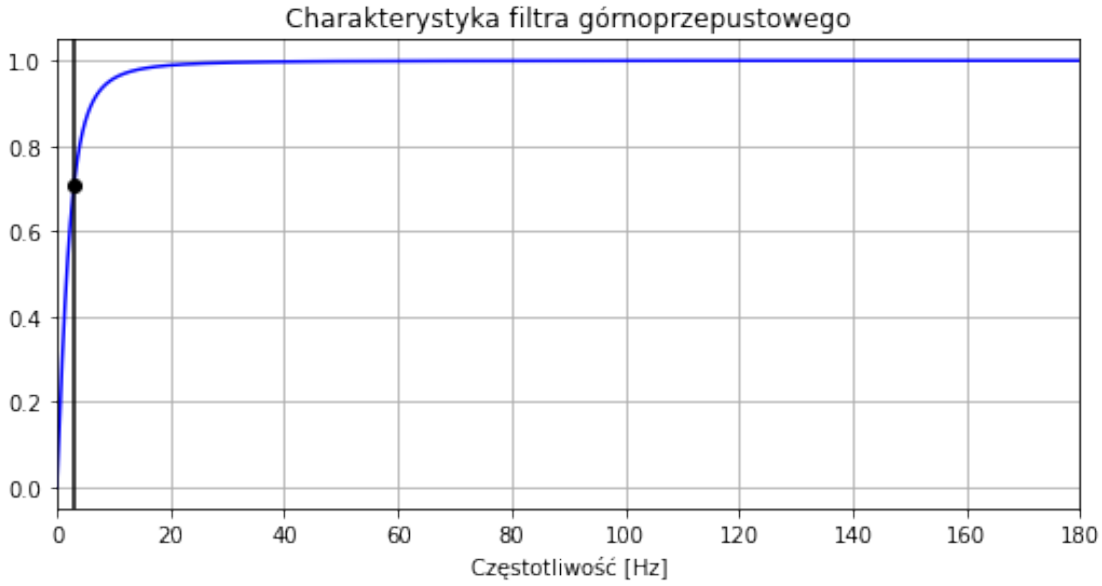
[18]:
```
FFT_DIFF = np.fft.fft(Y_diff)
X = np.linspace(0.0, FS_ECG/2, N_ECG//2)
Y_FFT_DIFF = 2/N_ECG * np.abs(FFT_DIFF[:N_ECG//2])

ax19 = drawGraph(19, "Spektrum różnicy sygnałów")
ax19.plot(X, Y_FFT_DIFF)
```

[18]:
```
[<matplotlib.lines.Line2D at 0x12606bee0>]
```



1.4.5 Filtr górnoprzepustowy

[19]:
```
def butter_highpass(cutoff, fs, order=5):
    return butter(order, cutoff, fs=fs, btype='high', analog=False)

def butter_highpass_filter(data, cutoff, fs, order=5):
    b, a = butter_highpass(cutoff, fs, order=order)
    y = lfilter(b, a, data)
    return y

# Wymagania filtru
order = 1
cutoff = 3  # desired cutoff frequency of the filter, Hz

# Zmienne filtra
b, a = butter_highpass(cutoff, FS_ECG, order)

# Wykres
w, h = freqz(b, a, fs=FS_ECG, worN=8000)
ax20 = drawGraph(20, "Charakterystyka filtra górnoprzepustowego")
ax20.plot(w, np.abs(h), 'b')
ax20.plot(cutoff, 0.5*np.sqrt(2), 'ko')
ax20.axvline(cutoff, color='k')
ax20.set_xlim(0, 0.5*FS_ECG)
ax20.set_xlabel('Częstotliwość [Hz]')
ax20.grid()
```

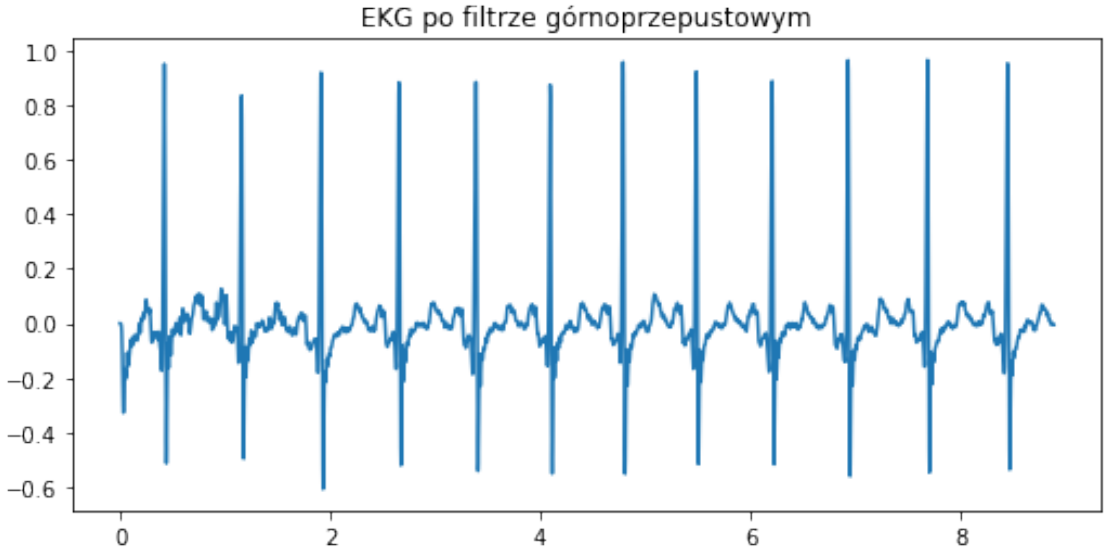


Parametry zostały dobrane w ten sposób, aby zminimalizować "pływanie" sygnału, jednocześnie

nie tracąc zbyt dużo informacji o sygnale.

[20]:
```
Y_final = butter_highpass_filter(Y_filtered, cutoff, FS_ECG, order)
ax21 = drawGraph(21, "EKG po filtrze górnoprzepustowym")
ax21.plot(time, Y_final)
```

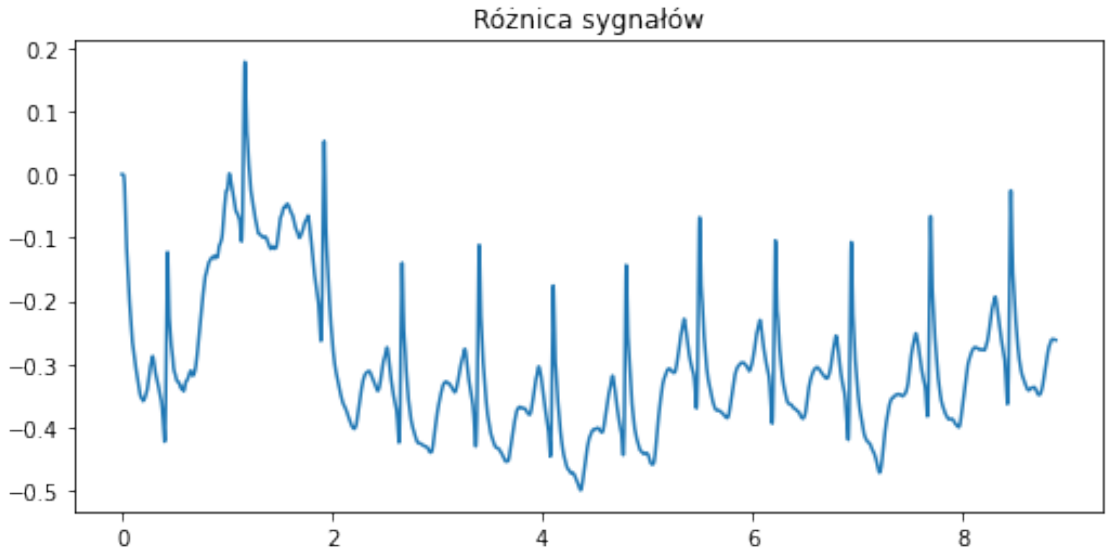
[20]: [<matplotlib.lines.Line2D at 0x1261531f0>]



1.4.6 Post high-pass spectrum

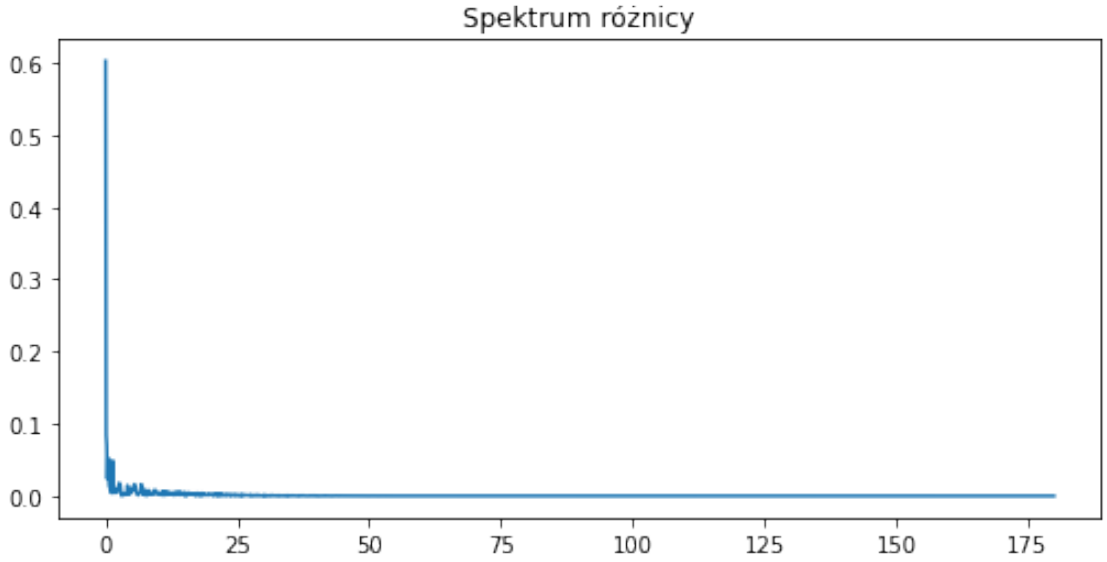
[21]:
```
Y_final_diff = np.subtract(Y_filtered, Y_final)
ax22 = drawGraph(22, "Różnica sygnałów")
ax22.plot(time, Y_final_diff)
```

[21]: [<matplotlib.lines.Line2D at 0x1261bd1c0>]



[22]:
```
FFT_FINAL_DIFF = np.fft.fft(Y_final_diff)
X = np.linspace(0.0, FS_ECG/2, N_ECG//2)
Y_FFT_FINAL_DIFF = 2/N_ECG * np.abs(FFT_FINAL_DIFF[:N_ECG//2])

ax23 = drawGraph(23, "Spektrum różnicy")
ax23.plot(X, Y_FFT_FINAL_DIFF)
```

[22]: [<matplotlib.lines.Line2D at 0x12621cca0>]



1.4.7 Wnioski

W wykresie 22 otrzymaliśmy sygnał EKG przepuszczony przez filtr o charakterze pasmoprzepustowym. Zminimalizowany został efekt “pływania” oraz zakłócenia z sieci elektrycznej.