Bogumił Wierzchowski 291498

Analiza sygnału w dziedzinie czasu i częstości

Ćwiczenie 4: Filtry

1. **Wstęp**

Celem ćwiczenia jest zapoznanie się z rodzajami oraz działaniem filtrów FIR oraz IIR.

Użyte oprogramowanie: *Python ver. 3.9.7*

Użyte biblioteki: *numpy, scipy, matplotlib*

1. **Kod źródłowy**

import numpy as np

import matplotlib.pyplot as plt

import scipy.signal as sig

import scipy.io

import sys

# cd .\semestr2\ASwDCiC\lab4\

plt.rcParams['font.size'] = '13'

def signaltonoiseScipy(a, axis=0, ddof=0):

    a = np.asanyarray(a)

    m = a.mean(axis)

    sd = a.std(axis=axis, ddof=ddof)

    return np.where(sd == 0, 0, m/sd)

N = 2000

A = 5

f = 5

fs = 1000

dt = 1/fs

t = np.arange(N) \* dt

x = A \* np.sin(2 \* np.pi \* f \* t) + A \*np.random.normal(0, 1, N)

widmo\_sygnalu = 20 \* np.log10(np.abs(np.fft.rfft(x \* np.hamming(N))) / N/2)

f = np.fft.rfftfreq(N, 1 / fs)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(x)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,x)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title= 'Zaszumiony sygnał x(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_sygnalu)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo zaszumionego sygnał')

plt.show()

dp\_100 = sig.firwin(100, 10, fs=fs)

przefiltrowany = sig.lfilter(dp\_100, 1, x)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany DP sygnał x(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego DP sygnału')

plt.show()

gp = sig.firwin(101, 100, pass\_zero=False, fs=fs)

przefiltrowany = sig.lfilter(gp, 1, x)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany GP sygnał x(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego GP sygnału')

plt.show()

b, a = sig.iirfilter(8, 10 / (fs / 2), ftype='butter', btype='lowpass')

przefiltrowany = sig.lfilter(b, a, x)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany DP sygnał x(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego DP sygnału')

plt.show()

b, a = sig.iirfilter(8, 100 / (fs / 2), ftype='butter', btype='highpass')

przefiltrowany = sig.lfilter(b, a, x)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany GP sygnał x(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego GP sygnału')

plt.show()

f1 = 5

f2 = 50

y = A \* np.sin(2\*np.pi\*f1\*t)+2\*np.sin(10\*np.pi\*f2\*t)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(y)))

widmo\_sygnalu = 20 \* np.log10(np.abs(np.fft.rfft(y \* np.hamming(N))) / N/2)

f = np.fft.rfftfreq(N, 1 / fs)

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,y)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Złożenie 2 sinusów y(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_sygnalu)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo złożenia 2 sinusów')

plt.show()

dp\_10 = sig.firwin(100, 10, fs=fs)

przefiltrowany = sig.lfilter(dp\_10, 1, y)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany DP sygnał y(t), SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego sygnał')

plt.show()

gp\_200 = sig.firwin(101, 200, pass\_zero=False, fs=fs)

przefiltrowany = sig.lfilter(gp\_200, 1, y)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany GP sygnał y(t), SNR: %.2f db' % SNRscipy, xlim=(0.5, 0.6))

ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego sygnał')

plt.show()

mat = scipy.io.loadmat('ecg.mat')

ecg = mat['ecg']

N = ecg.shape[0]

ecg = ecg.reshape(N)

fs = 500

dt = 1/fs

t = np.arange(N) \* dt

widmo\_ecg = 20 \* np.log10(np.abs(np.fft.rfft(ecg \* np.hamming(N))) / N/2)

f = np.fft.rfftfreq(N, 1 / fs)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(ecg)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,ecg)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Wczytany sygnał $\it{ecg.mat}$, SNR: %.2f db' % SNRscipy)

ax2.plot(f, widmo\_ecg)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo sygnału')

plt.show()

dp\_40 = sig.firwin(100, 40, fs=fs)

przefiltrowany = sig.lfilter(dp\_40, 1, ecg)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

gp\_8 = sig.firwin(101, 8, pass\_zero=False, fs=fs)

przefiltrowany = sig.lfilter(gp\_8, 1, przefiltrowany)

widmo\_przefiltrowany = 20 \* np.log10(np.abs(np.fft.rfft(przefiltrowany \* np.hamming(N))) / N/2)

SNRscipy = 20\*np.log10(abs(signaltonoiseScipy(przefiltrowany)))

fig, (ax1, ax2) = plt.subplots(2, 1)

ax1.plot(t,przefiltrowany)

ax1.set(xlabel='Czas [s]', ylabel='amplituda [a.u.]', title='Przefiltrowany sygnał $\it{ecg.mat}$, SNR: %.2f db' % SNRscipy)

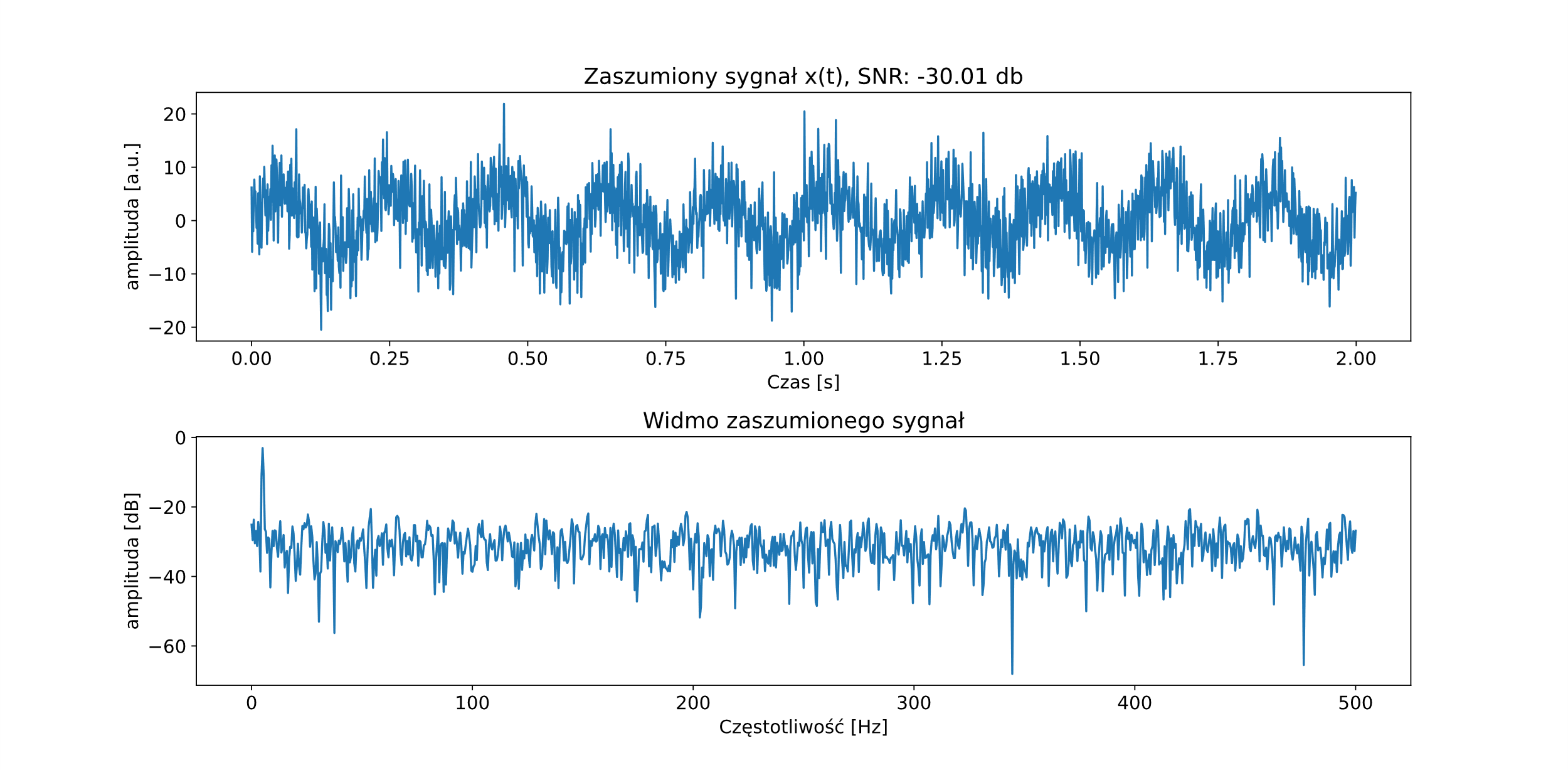
ax2.plot(f, widmo\_przefiltrowany)

ax2.set(xlabel='Częstotliwość [Hz]', ylabel='amplituda [dB]', title='Widmo przefiltrowanego sygnał')

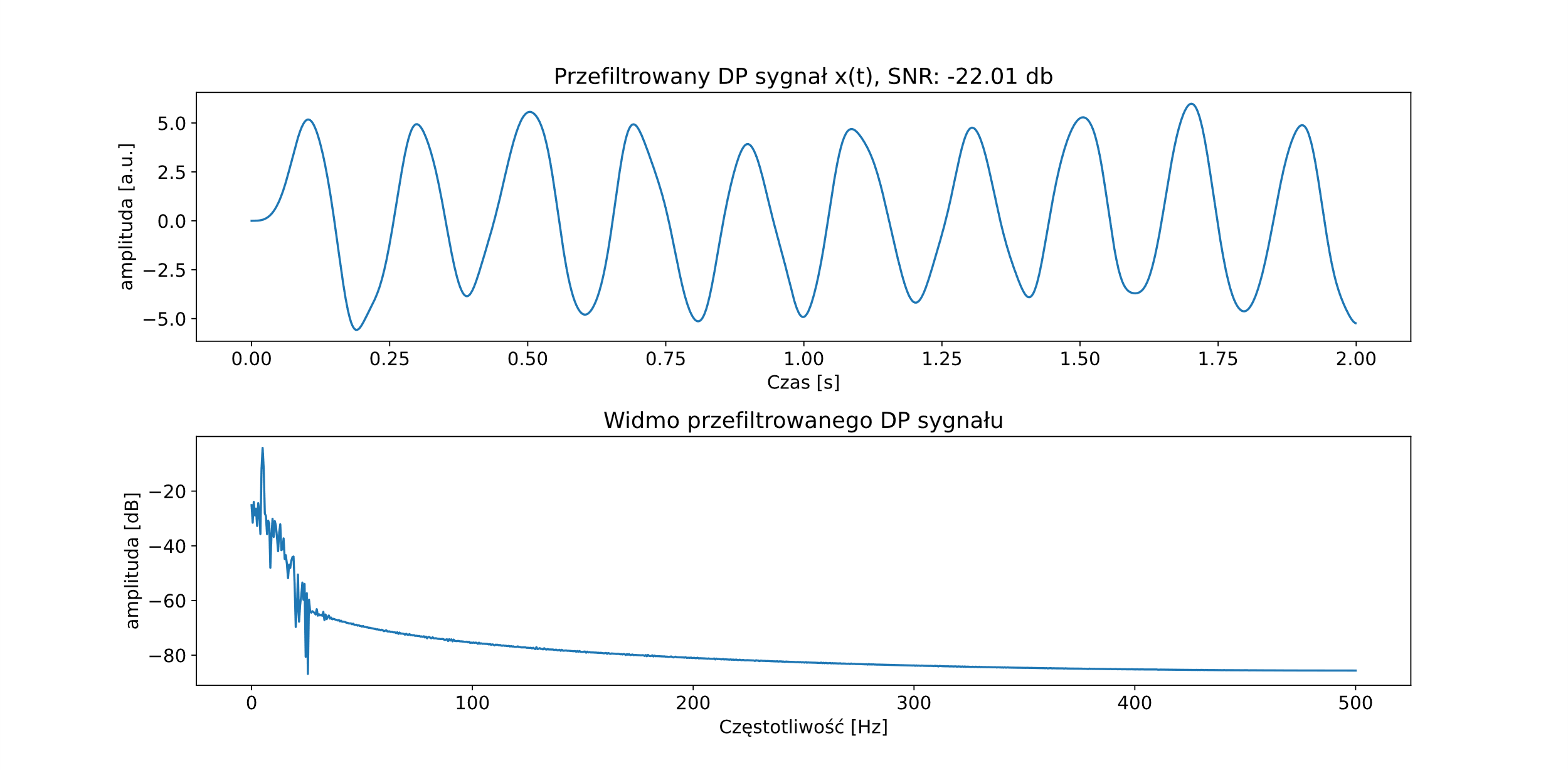
plt.show()

1. **Wyniki**

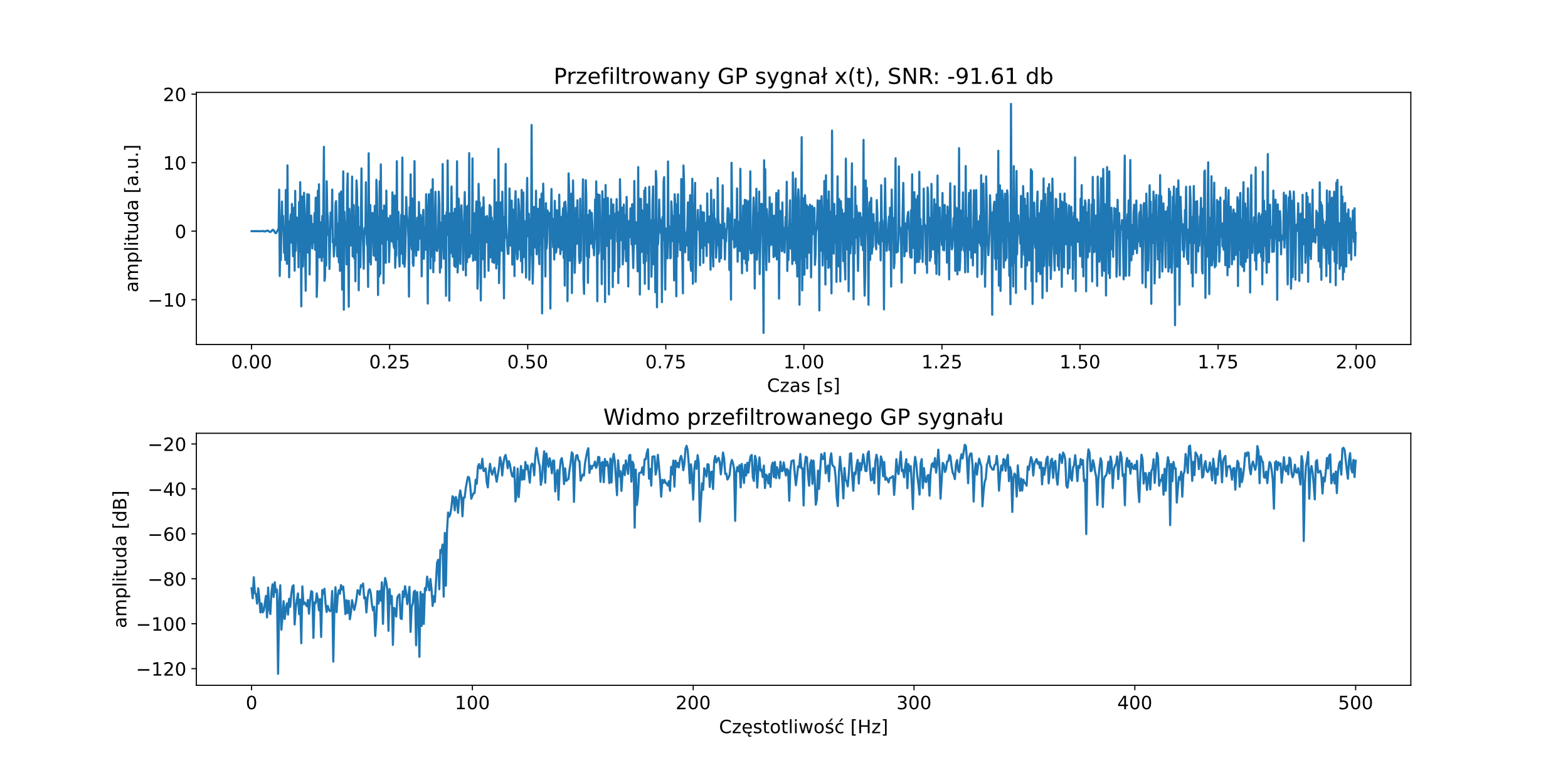
Zaszumiony sygnał postaci: oraz jego widmo, gdzie oznacza losową wartość w chwili t wylosowaną ze standardowego rozkładu normalnego, przedstawia wykres poniżej:



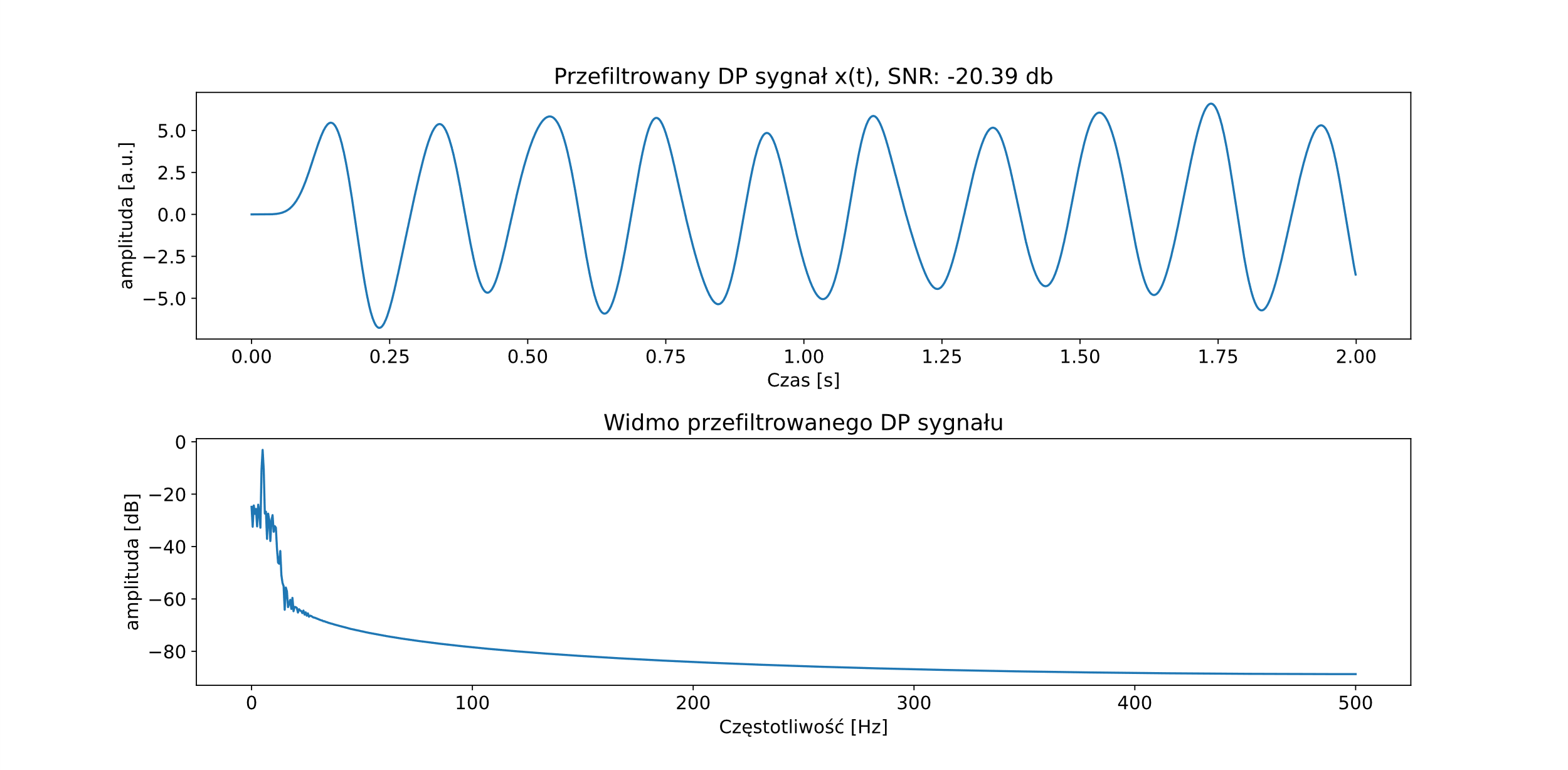
Przefiltrowany sygnał oraz jego widmo) filtrem FIR dolnoprzepustowym o długości i częstotliwości granicznej 10Hz, przedstawia wykres poniżej:



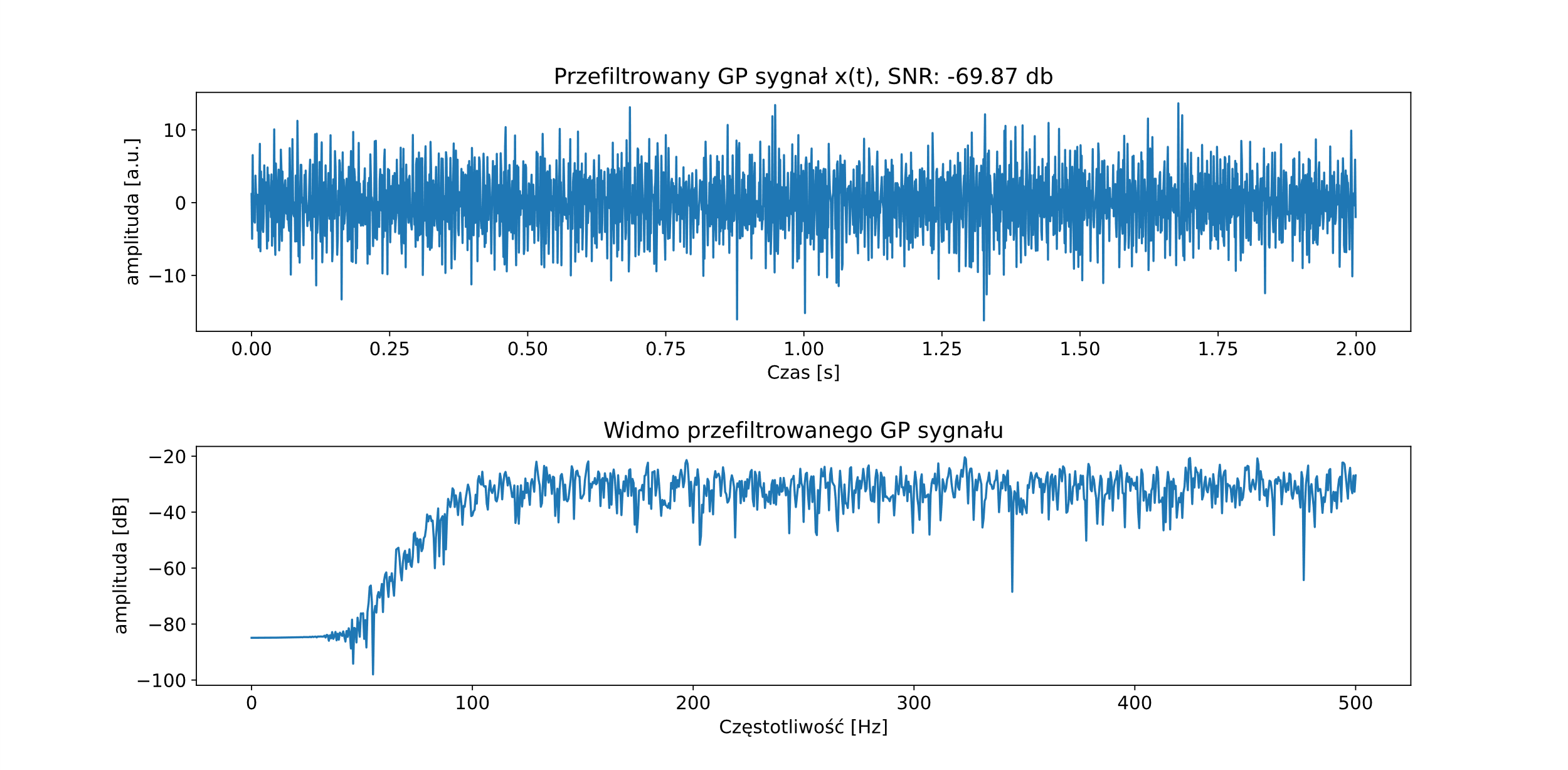
Przefiltrowany sygnał oraz jego widmo) filtrem FIR górnoprzepustowym o długości i częstotliwości granicznej 100Hz, przedstawia wykres poniżej:



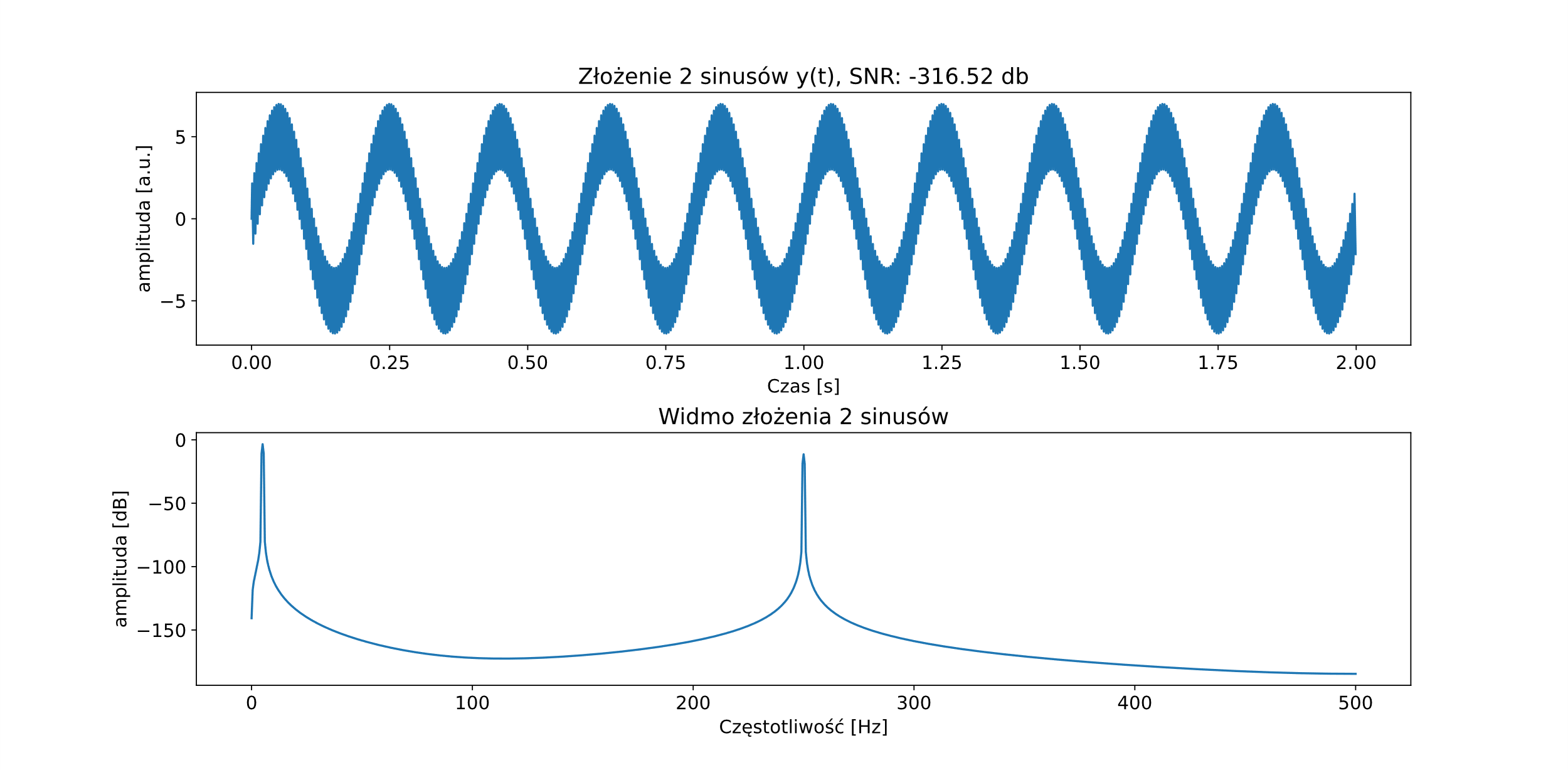
Przefiltrowany sygnał oraz jego widmo) filtrem IIR (Butterwortha) dolnoprzepustowym o rzędzie i częstotliwości granicznej 10Hz, przedstawia wykres poniżej:



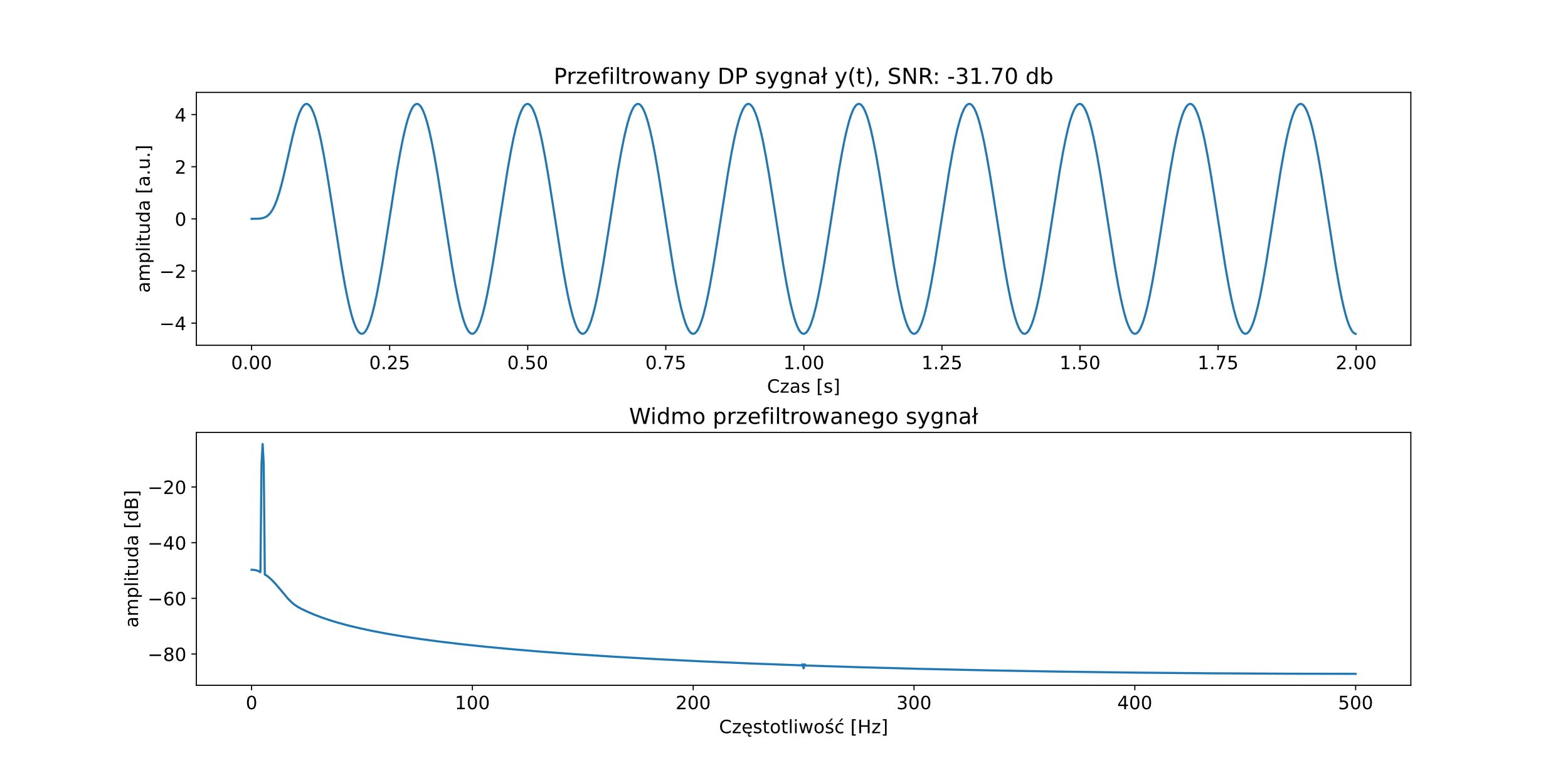
Przefiltrowany sygnał oraz jego widmo) filtrem IIR (Butterwortha) górnoprzepustowym o rzędzie i częstotliwości granicznej 100Hz, przedstawia wykres poniżej:



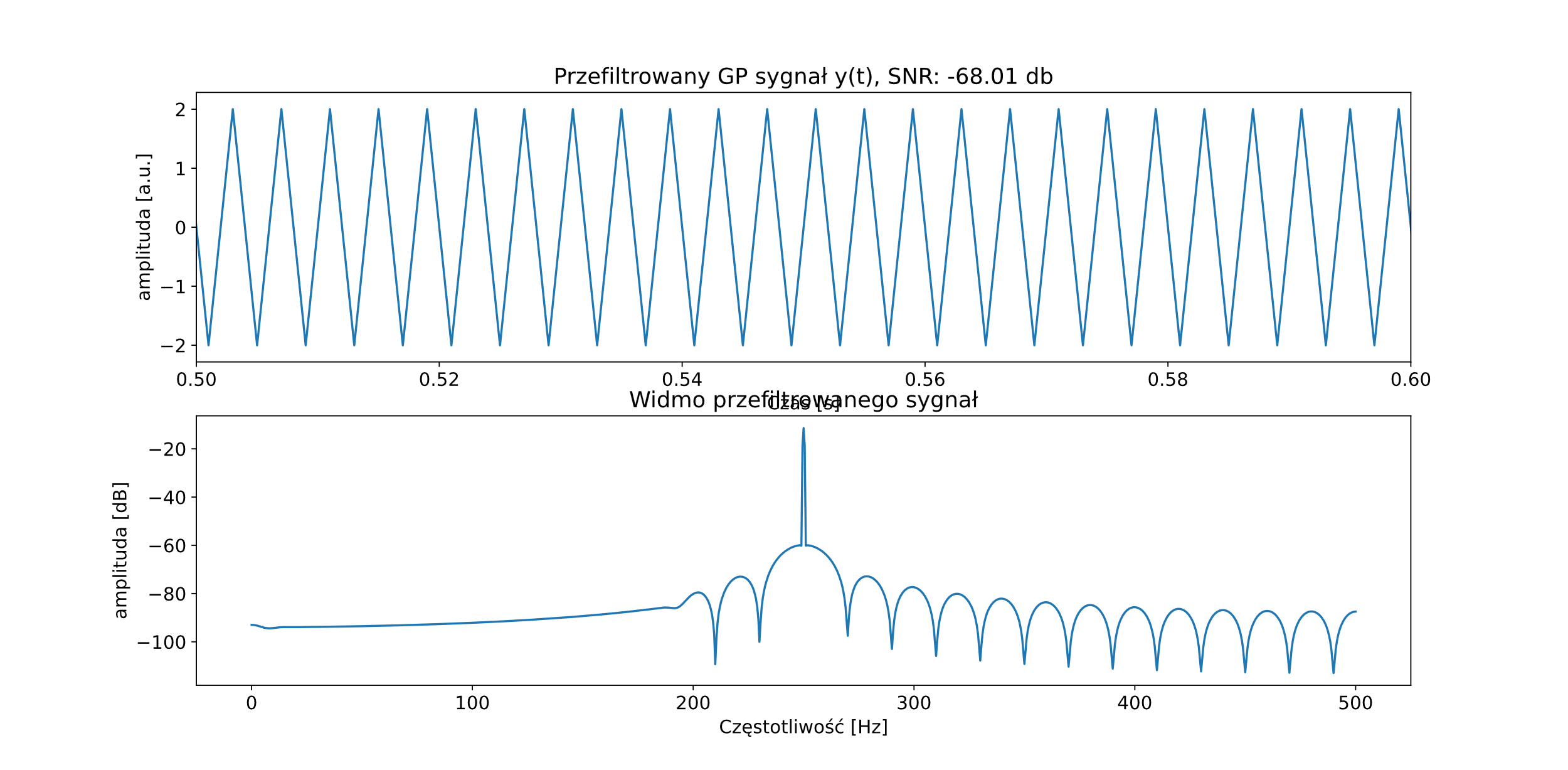
Złożenie 2 sinusów postaci: oraz ich widmo, przedstawia wykres poniżej:



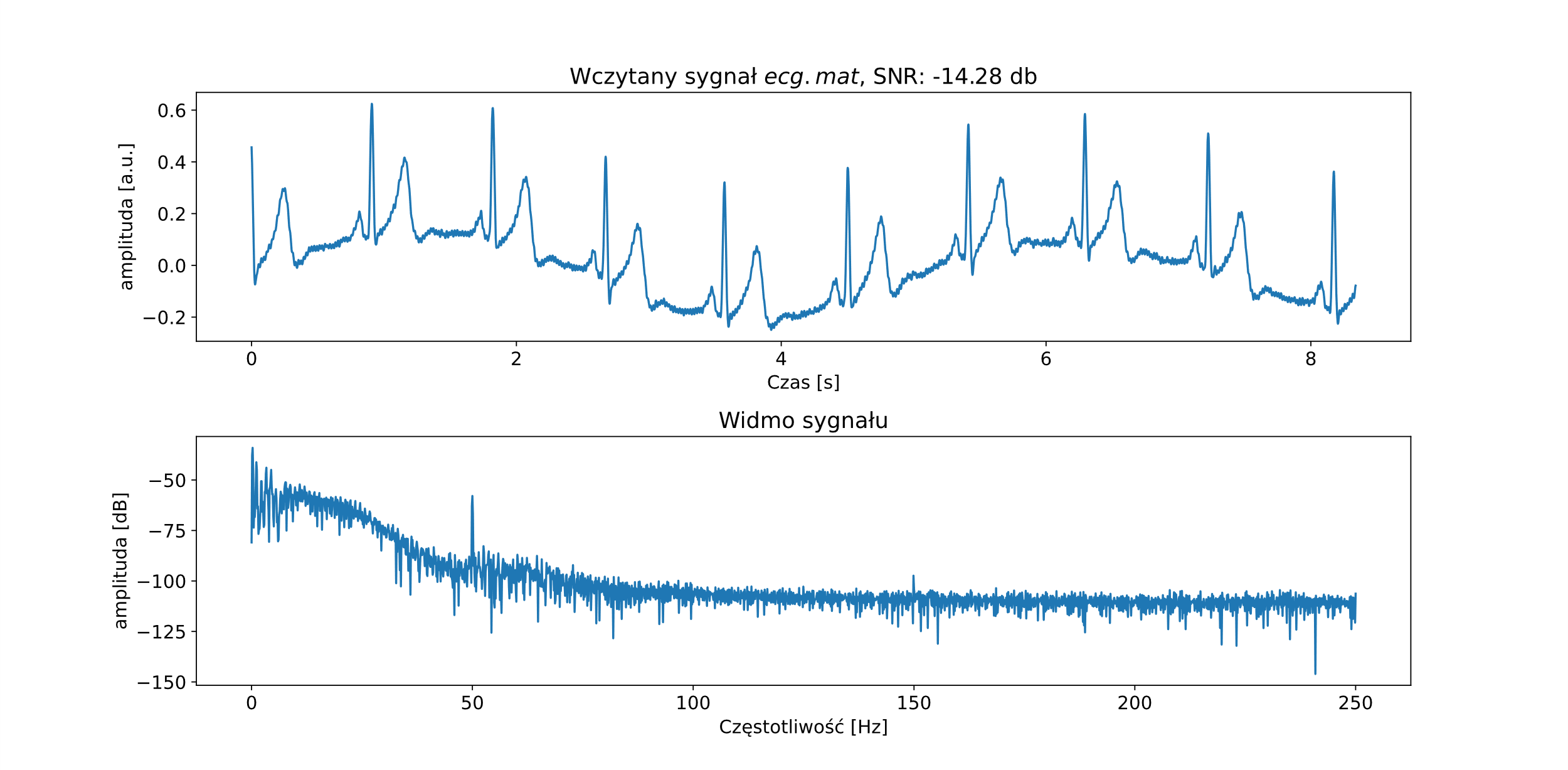
Przefiltrowany sygnał dolnoprzepustowym filtrem FIR o częstotliwości odcięcia 10Hz oraz jego widmo, przedstawia wykres poniżej:



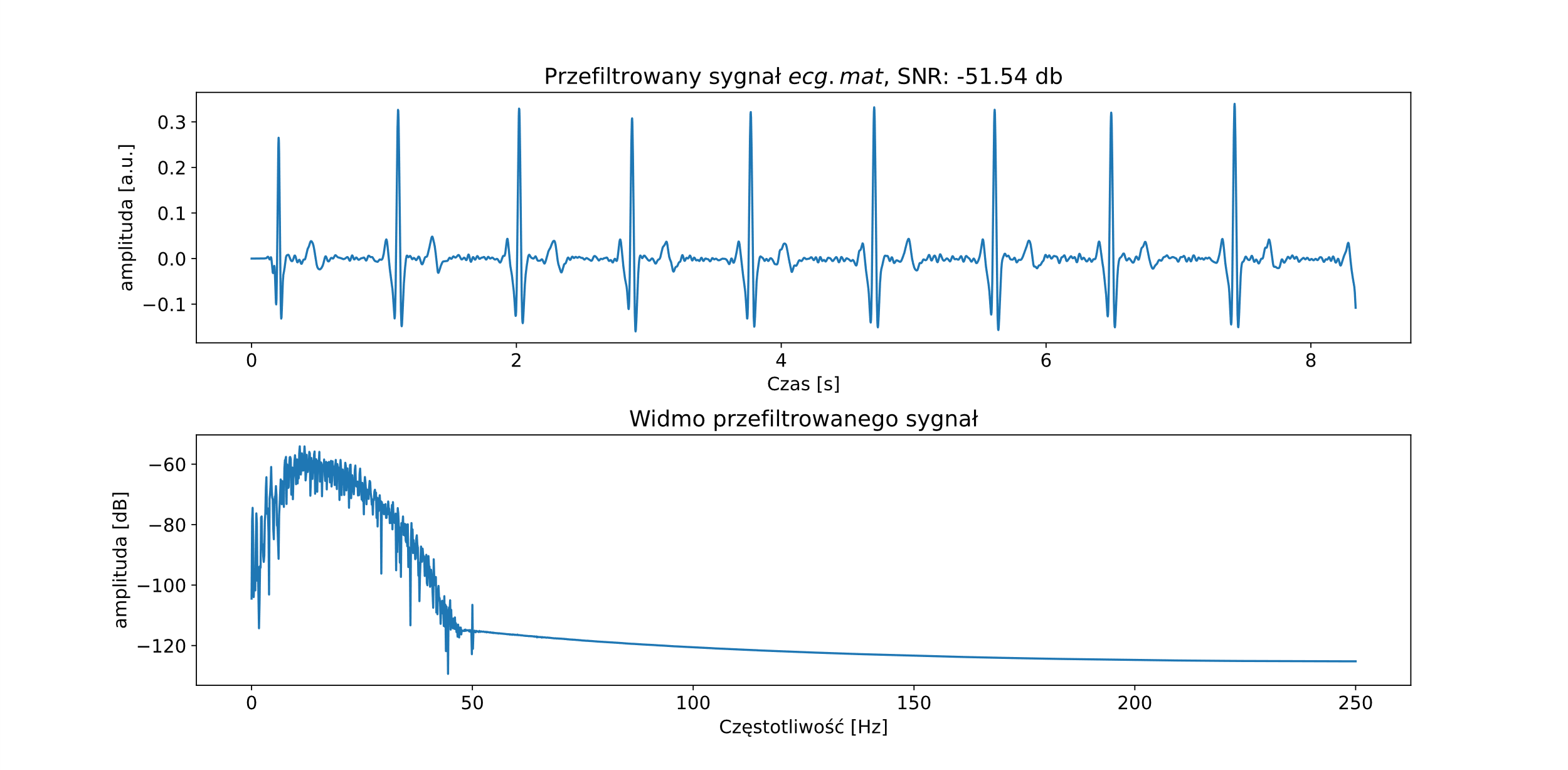
Przefiltrowany sygnał górnoprzepustowym filtrem FIR o częstotliwości odcięcia 200Hz oraz jego widmo, przedstawia wykres poniżej:



Wczytany sygnał *ecg.mat* oraz jego widmo przedstawia wykres poniżej:



Przefiltrowany najpierw dolnoprzepustowo () a następnie górnoprzepustowo () sygnał *ecg.mat* oraz jego widmo przedstawia wykres poniżej (oba filtry FIR):



Filtry zostały dobrane tak aby wyciąć zakłócenia sieci elektrycznej o częstotliwości 50Hz oraz falowanie sieci bazowej o częstotliwości < 8Hz.

1. **Wnioski**

* Filtry typu FIR oraz IIR można wykorzystać do analizy sygnału EKG.
* Filtry górnoprzepustowe tłumią częstotliwości niże od granicznej a przepuszczają te wyższe.
* Filtry dolnoprzepustowe tłumią częstotliwości wyższe od granicznej a przepuszczają te niższe.
* Zarówno filtry FIR jak i IIR nie są idealne, nie zapewniają idealnego tłumienia powyżej częstotliwości odcięcia lecz istnieje pewne pasmo przejściowe.
* Sygnały analityczne lepiej poddają się filtracji niż te rzeczywiste.