Bogumił Wierzchowski 291498

Analiza sygnału w dziedzinie czasu i częstości

Ćwiczenie nr. 1

1. **Wstęp**

Celem ćwiczenia jest zbadanie sygnału nagranego w pliku *chord.wav* w dziedzinie czasu i częstości, określenie jego typu oraz wykonanie wykresów w funkcji czasu i widma mocy, a także stworzenie metody identyfikującej podane nagranie.

Użyte oprogramowanie: *Python ver. 3.9.7*

Użyte biblioteki: *numpy, scipy, matplotlib, pandas*

1. **Kod źródłowy**

import numpy as np

from scipy.io import wavfile

import scipy.io

import matplotlib.pyplot as plt

import librosa.display

import scipy.signal as sig

from scipy.signal import find\_peaks

import pandas as pd

wav\_fname = 'data/acc1/acc1/chord.wav'

samplerate, data = wavfile.read(wav\_fname)

plt.rcParams["figure.figsize"] = [15, 8]

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

length = data.shape[0] / samplerate

time = np.linspace(0., length, data.shape[0])

plt.plot(time,data)

plt.xlabel("Czas [s]")

plt.ylabel("Amplituda")

plt.title('Sygnał w dziedzinie czasu')

plt.show()

window = np.hamming(len(data))

widmo\_amp = np.abs(np.fft.rfft(data \* window)) / (len(data)/2)

f = np.fft.rfftfreq(len(data), 1/samplerate)

plt.plot(f, widmo\_amp)

plt.xlabel('częstotliwość [Hz]')

plt.ylabel('amplituda widma')

plt.title('Transformata Fouriera z oknem Hamminga')

plt.xlim(-30,4000)

#plt.ylim(0, 1000)

plt.show()

widmo\_dB = 20 \* np.log10(widmo\_amp \*\* 2)

plt.plot(f, widmo\_dB)

plt.xlabel('częstotliwość [Hz]')

plt.ylabel('moc [dB]')

plt.title('Widmo mocy sygnału')

#plt.xlim(-10,350)

#plt.ylim(0, 1000)

plt.show()

plt.plot(f, widmo\_dB)

plt.xlabel('częstotliwość [Hz]')

plt.ylabel('moc [dB]')

plt.title('Widmo mocy sygnału dla przedziału 16Hz - 4kHz')

plt.xlim(16,4000)

plt.ylim(-120, 120)

plt.show()

plt.plot(f, widmo\_dB)

plt.xlabel('częstotliwość [Hz]')

plt.ylabel('moc [dB]')

plt.title('Skala log-log')

plt.xscale("log")

#plt.xlim(16,4000)

#plt.ylim(-120, 120)

plt.show()

plt.plot(f, widmo\_dB)

plt.xlabel('częstotliwość [Hz]')

plt.ylabel('moc [dB]')

plt.title('Skala log-log dla przedziału 100Hz - 1kHz i powyżej 50dB')

plt.xscale("log")

plt.xlim(100,1000)

plt.ylim(50, 110)

plt.show()

table = scipy.io.loadmat('data/tones\_data.mat') # Importujemy częstotliwości i nazwy nut

f\_list = table['f'].tolist()[0] # Tworzymy listę częstotliwości

n\_list = [table['n'][0].tolist()[i][0].replace(" ", "") for i in range(0,len(f\_list))] # Tworzymy listę nut + usuwamy spacje

tones\_dict = dict(zip(f\_list,n\_list)) # Otrzymujemy słownik

start\_freq = 100

stop\_freq = 1000

def znajdz\_piki(f, widmo\_dB, start\_freq, stop\_freq, height, distance, tones\_dict, top):

    start = int(start\_freq \* len(widmo\_dB)/max(f))

    stop = int(stop\_freq \* len(widmo\_dB)/max(f))

    widmo\_db\_cropped = widmo\_dB[start:stop]

    f\_cropped = f[start:stop]

    peaks, \_ = find\_peaks(widmo\_db\_cropped, height=height, distance=distance)

    plt.plot(f\_cropped, widmo\_db\_cropped)

    plt.plot(f\_cropped[peaks], widmo\_db\_cropped[peaks], "x")

    plt.axhline(y = height, linestyle='-', color="gray")

    plt.xlabel('częstotliwość [Hz]')

    plt.ylabel('moc [dB]')

    plt.title(f'Znalezione piki dla przedziału {start\_freq}Hz - {stop\_freq}Hz i powyżej {height}dB')

    plt.xscale("log")

    plt.show()

    dzwieki = list(tones\_dict.values())

    czestosci = list(tones\_dict.keys())

    df = pd.DataFrame(list(zip(f\_cropped[peaks], widmo\_db\_cropped[peaks])), columns=['czestotliwosc\_piku', 'widmo\_dB'])

    df.sort\_values(by = 'widmo\_dB', ascending=False, inplace=True)

    df = df.iloc[:top,:]

    fitted\_freq = []

    fitted\_sounds = []

    for frequency in df.iloc[:,0]:

        fitted\_freq.append(czestosci[min(range(len(czestosci)), key=lambda i: abs(czestosci[i]-frequency))])

        fitted\_sounds.append(dzwieki[min(range(len(czestosci)), key=lambda i: abs(czestosci[i]-frequency))])

    df = df.assign(dop\_czestotliwosc = fitted\_freq)

    df = df.assign(nazwa\_dzwieku = fitted\_sounds)

    df.sort\_values(by = 'czestotliwosc\_piku', ascending=True, inplace=True)

    return(df)

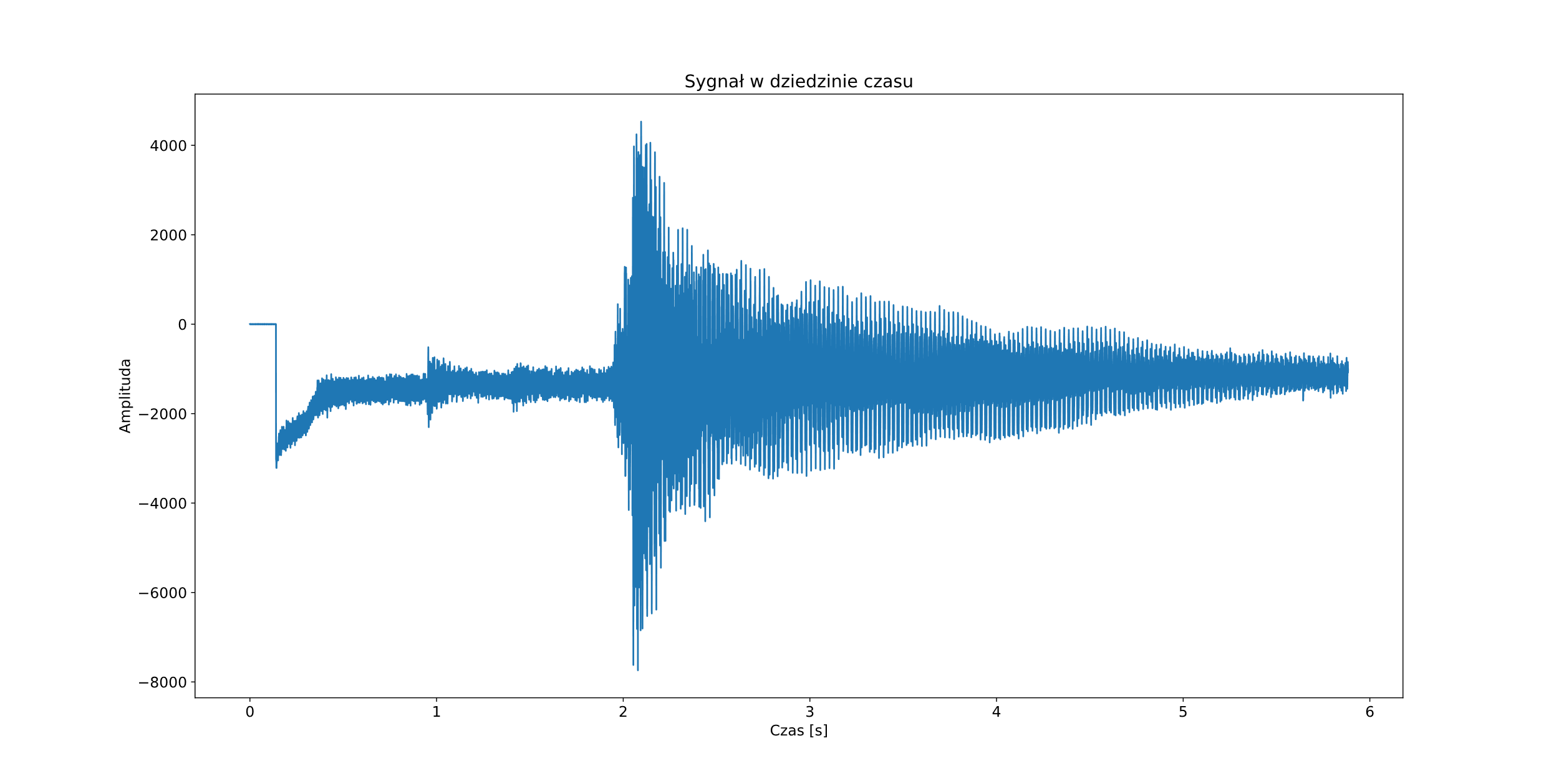
piki\_df = znajdz\_piki(f, widmo\_dB, start\_freq, stop\_freq, 50, 50, tones\_dict, 6)

piki\_df.reset\_index(inplace=True, drop=True)

piki\_df.to\_csv('data/piki.csv')

1. **Wyniki**

Wykres na Rys. 1 przedstawia sygnał w dziedzinie czasu:



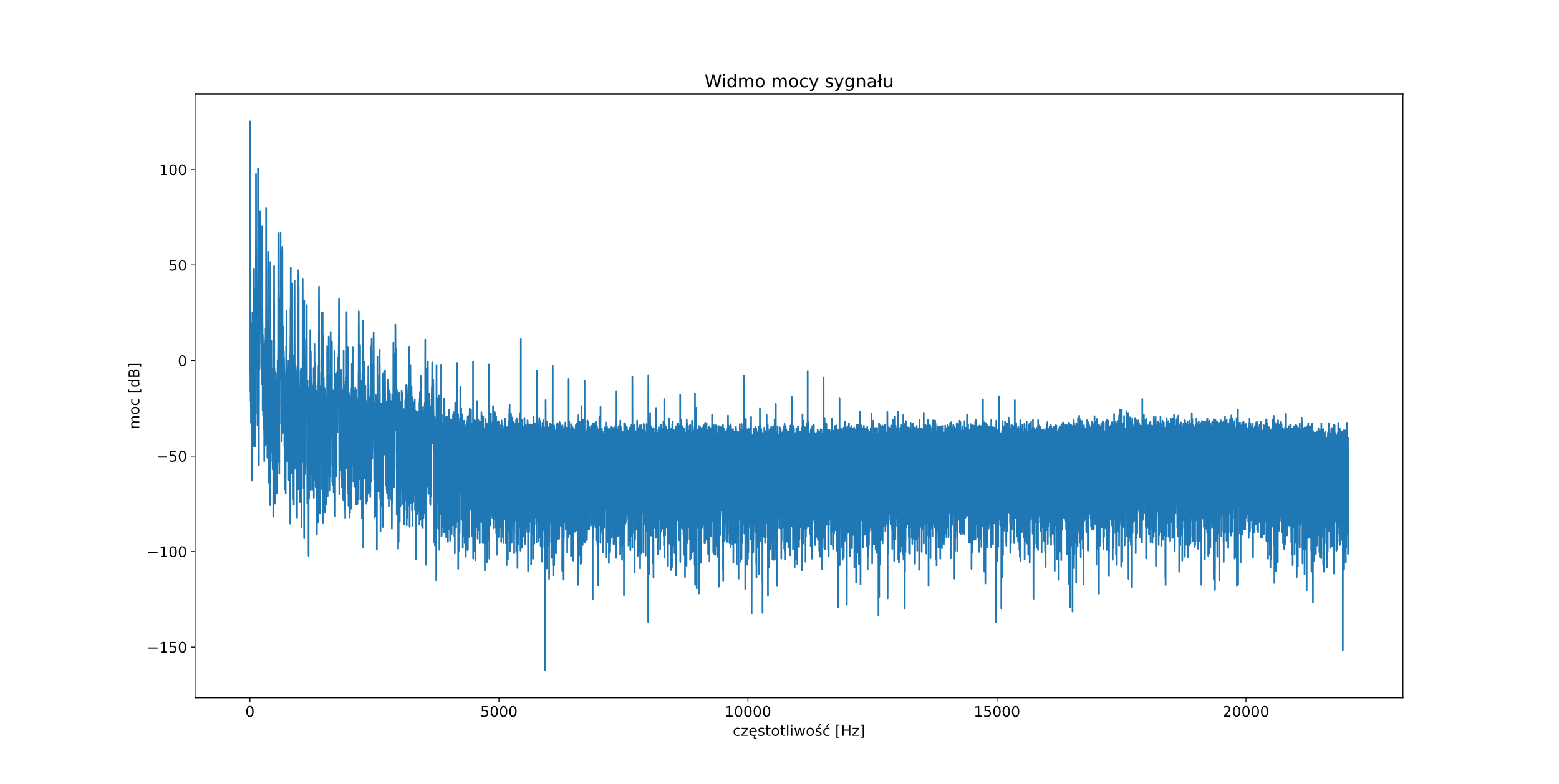
Rys.

Wykres na Rys. 1 przedstawia transformatę Fouriera z oknem Hamminga sygnału uciętą do częstotliwości 4kHz:



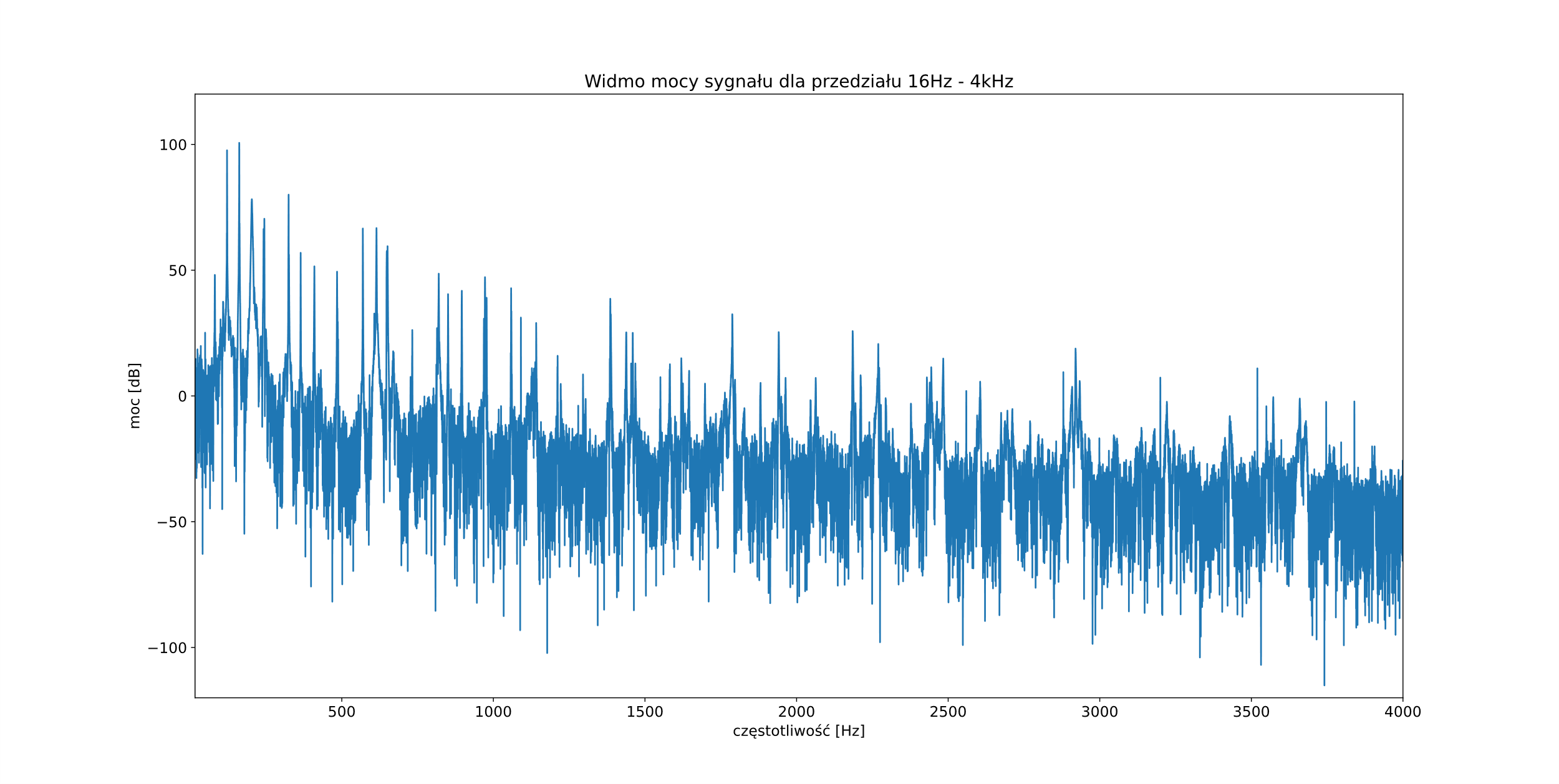
Rys.

Wykres na Rys. 1 przedstawia widmo mocy całego sygnału:



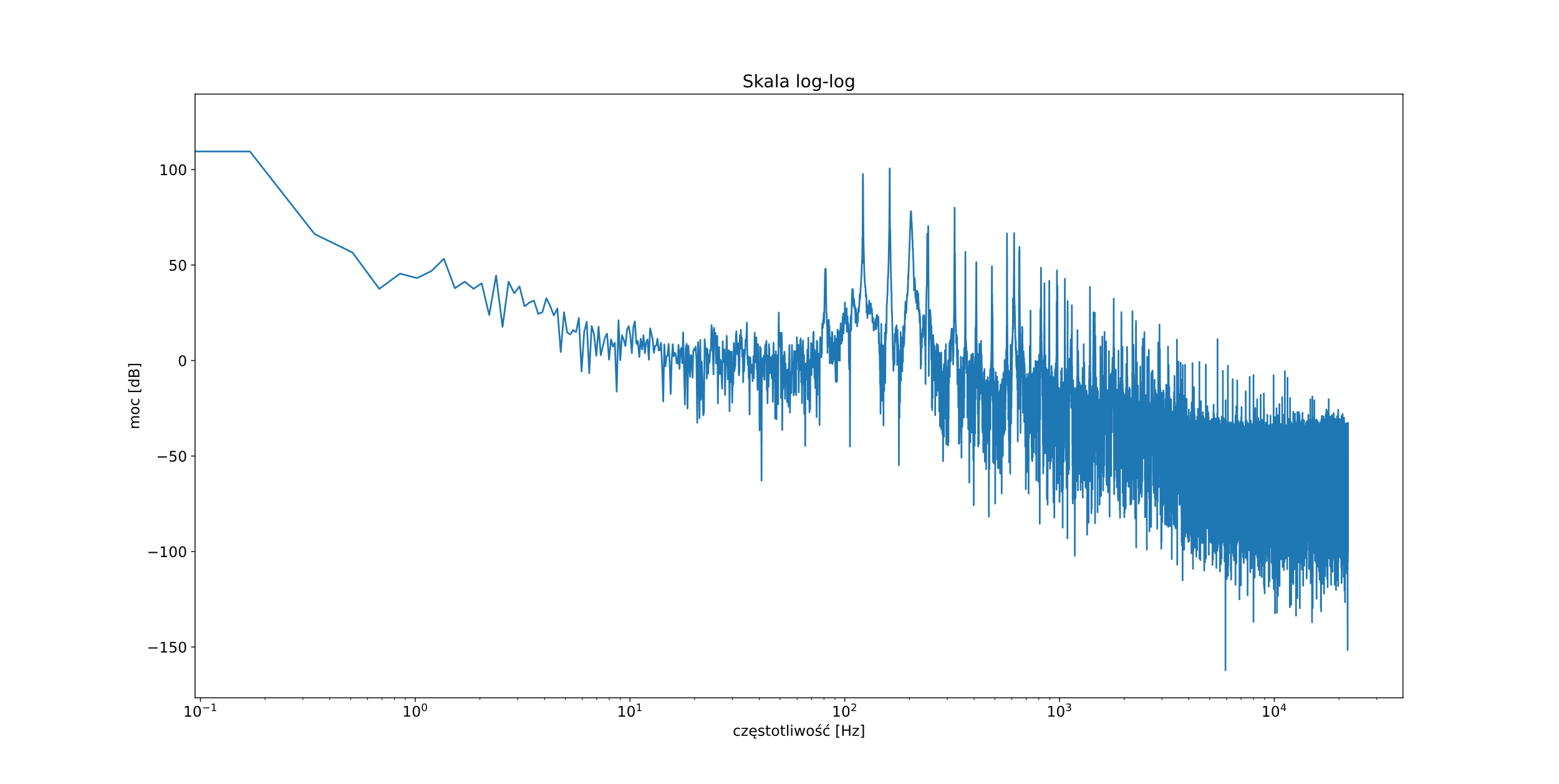
Rys.

Wykres na Rys. 1 przedstawia widmo mocy sygnału dla przedziału 16Hz – 4kHz:



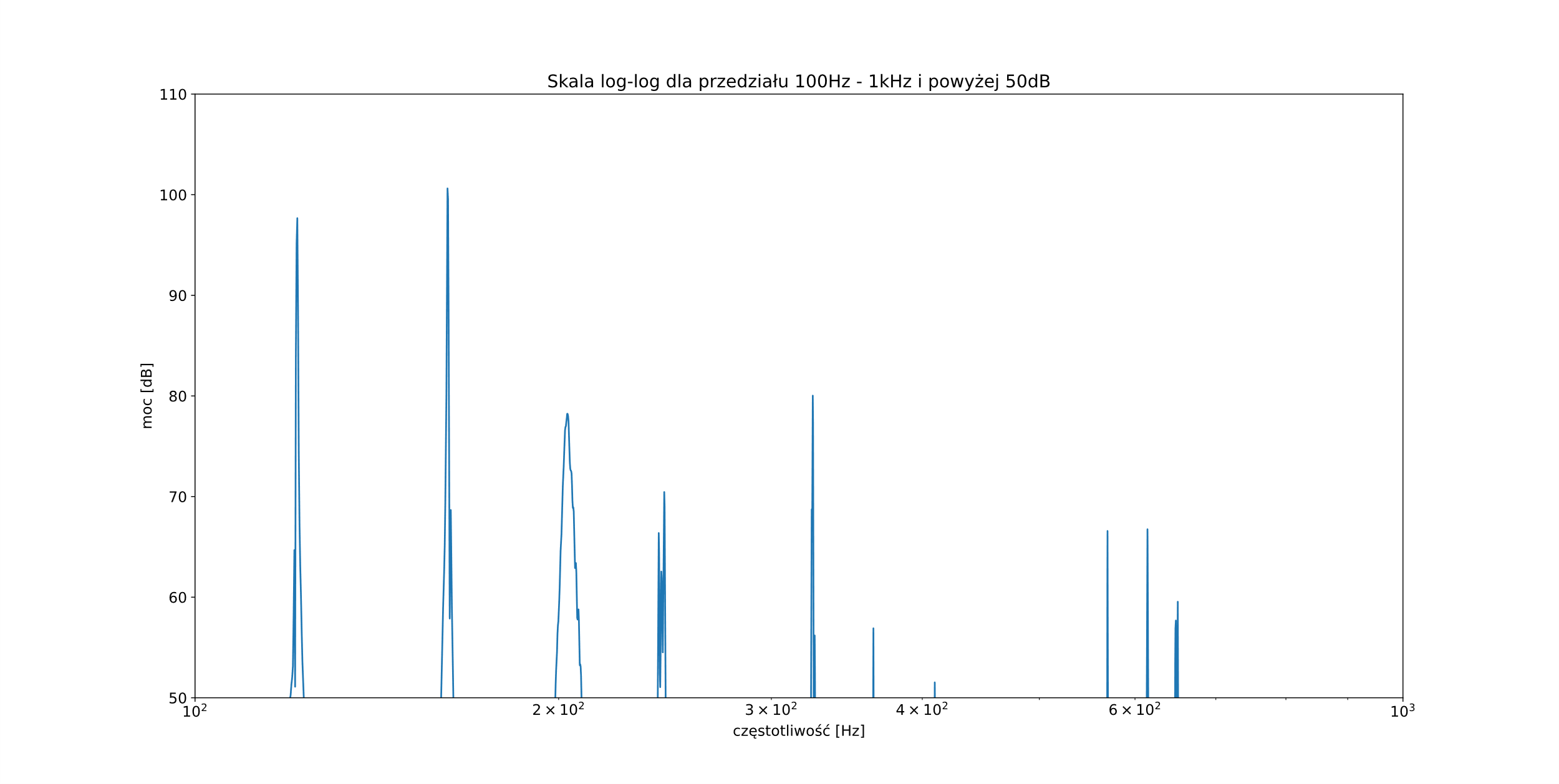
Rys.

Wykres na Rys. 1 przedstawia widmo mocy sygnału w skali log-log:



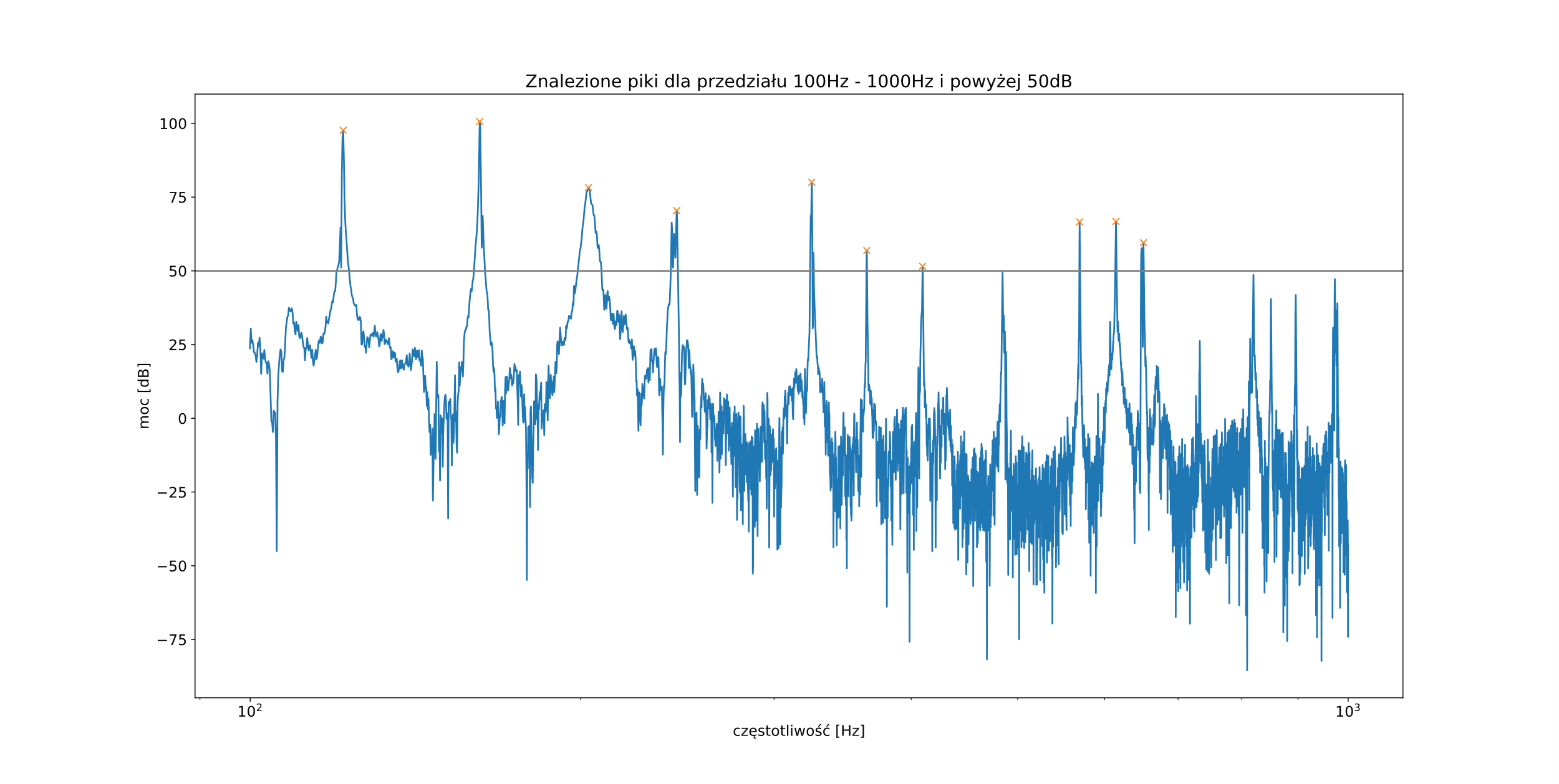
Rys.

Wykres na Rys. 1 przedstawia widmo mocy dla przedziału 100Hz - 1kHz z wartościami powyżej 50dB:



Rys. 6

Za pomocą funkcji *znajdz\_piki* wyznaczono położenia położenia pików i zaznaczono na wykresie znakiem ‘x’. Wykres zwracany przez powyższą funkcję jest przedstawiony na Rys. 6. Szara linia oznacza próg wartości, powyżej którego znajdowane były piki (w tym przypadku 50dB).



Rys. 7

Znalezione częstości (ograniczone do 6 najwyższych pików) oraz dopasowane do nich nazwy dźwięków (przy pomocy pliku *tones\_data.mat*) przedstawione są w tabeli poniżej:

|  |  |  |  |
| --- | --- | --- | --- |
| **Częstotliwość piku [Hz]** | **Widmo mocy [dB]** | **Dopasowana częstotliwość [Hz]** | **Nazwa dźwięku** |
| 121,53 | 97,67 | 123,47 | B2 |
| 161,81 | 100,63 | 164,81 | E3 |
| 203,29 | 78,23 | 207,65 | G#3 |
| 244,59 | 70,45 | 246,94 | B3 |
| 324,64 | 80,03 | 329,62 | E4 |
| 614,44 | 66,75 | 622,25 | D#5 |

1. **Wnioski**

* Analizowana melodia jest akordem.
* Akord ten zawiera dźwięki (6 najwyższych pod względem mocy): B2, E3, G#3, B3, E4, D#5.