Bogumił Wierzchowski 291498

Analiza sygnału w dziedzinie czasu i częstości

Ćwiczenie 2: FFT-logia

1. Wstęp

Celem ćwiczenia jest wprowadzenie do analizy sygnału, zapoznanie się z pojęciami: częstość próbkowania, częstość Nyquista, aliasing, zero padding, upsampling oraz zapoznanie z podstawowymi własnościami transformaty Fouriera w krótkim oknie czasowym.

Użyte oprogramowanie: *Python ver. 3.9.7*

Użyte biblioteki: *numpy, scipy, matplotlib*

2. Kod źródłowy

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.signal as sig
from scipy.io import wavfile

plt.rcParams["figure.figsize"] = [15, 8]
plt.rcParams['font.size'] = '13'

N = 1024
A1 = 5
A2 = 3
f1 = 10
f2 = 20
fs = 1000
dt = 1/fs

t = np.arange(N)*dt
x1 = A1 * np.sin(2 * np.pi * f1 * t)
x2 = A2 * np.sin(2 * np.pi * f2 * t)
x = x1 + x2

widmo = 20 * np.log10(np.abs(np.fft.rfft(x))) / (N/2)
freq = np.fft.rfftfreq(N, d=dt)
```

```
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału nieskończonego oraz jego widma ')
ax1.plot(t, x)
ax1.set(xlabel='czas [s]', ylabel='sygnał x(t)', xlim = (0,0.5))
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]', xlim =
(0,100))
plt.show()

boxsize = 200
shift = 300
window = sig.windows.boxcar(boxsize)
rect = np.concatenate((np.zeros(int(N/2 - shift) - int(window.size/2)),
window))
rect = np.concatenate((rect, (np.zeros(int(N/2 + shift) -
int(window.size/2)))))
widmo = 20 * np.log10(np.abs(np.fft.rfft(rect))) / (N/2)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału prostokątnego oraz jego widma ')
ax1.plot(t, rect)
ax1.set(xlabel='czas [s]', ylabel='sygnał x(t)', xlim = (0,0.5))
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]', xlim =
(0,100))
plt.show()

signal = x * rect
widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie złożenia syg. prost. z syg niesk. oraz jego widma ')
ax1.plot(t, signal)
ax1.set(xlabel='czas [s]', ylabel='sygnał x(t)', xlim = (0,0.5))
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]', xlim =
(1,100), ylim=(-0.025, 0.125))
plt.show()

window = np.hamming(N/2)
hamm = np.concatenate((window, np.full(N - window.size, min(window))))

widmo = 20 * np.log10(np.abs(np.fft.rfft(hamm))) / (N/2)
fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie okna Hamminga oraz jego widma ')
ax1.plot(t, hamm)
ax1.set(xlabel='czas [s]', ylabel='sygnał x(t)', xlim = (0,0.5))
ax2.plot(freq, widmo)
```

```
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]', xlim =
(0,100), ylim=(-0.18, 0.125))
plt.show()

signal = x * hamm
widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie złożenia okna Hamminga z syg. niesk. oraz jego widma
')
ax1.plot(t, signal)
ax1.set(xlabel='czas [s]', ylabel='sygnał x(t)', xlim = (0,0.5))
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]', xlim =
(0,100), ylim=(-0.005, 0.12))
plt.show()

wav_fname = 'chord.wav'
samplerate, data = wavfile.read(wav_fname)

length = data.shape[0] / samplerate
N = data.shape[0]
t = np.linspace(0., length, data.shape[0])
freq = np.fft.rfftfreq(len(data), 1/samplerate)

signal = data
widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału $\it{chord.wav}$ przy częstości próbkowania '
+ f'fs = {samplerate}Hz')
ax1.plot(t, signal)
ax1.set(xlabel='t [s]', ylabel='sygnał x(t)')
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]')
plt.show()

dec = 4
signal = data[::dec]
widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)
fs = samplerate/dec
freq = np.fft.rfftfreq(len(signal), 1/fs)
time = t[::dec]

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału $\it{chord.wav}$ przy częstości próbkowania '
+ f'fs = {fs}Hz')
ax1.plot(time, signal)
ax1.set(xlabel='t [s]', ylabel='sygnał x(t)')
```

```
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]')
plt.show()

dec = 32
signal = data[::dec]
widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)
fs = samplerate/dec
freq = np.fft.rfftfreq(len(signal), 1/fs)
time = t[::dec]

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału $\it{chord.wav}$ przy częstości próbkowania '
+ f'fs = {fs}Hz')
ax1.plot(time, signal)
ax1.set(xlabel='t [s]', ylabel='sygnał x(t)')
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]')
plt.show()

dec = 128
signal = data[::dec]
widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)
fs = samplerate/dec
freq = np.fft.rfftfreq(len(signal), 1/fs)
time = t[::dec]

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału $\it{chord.wav}$ przy częstości próbkowania '
+ f'fs = {fs}Hz')
ax1.plot(time, signal)
ax1.set(xlabel='t [s]', ylabel='sygnał x(t)')
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]')
plt.show()

zero_array = np.zeros(4*len(data))
signal = np.concatenate((data, zero_array))
length = signal.shape[0] / samplerate
time = np.linspace(0., length, signal.shape[0])

widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)
fs = samplerate
freq = np.fft.rfftfreq(len(signal), 1/fs)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału $\it{chord.wav}$ oraz jego widma z
wykorzystaniem $\it{zero-padding}$')
ax1.plot(time, signal)
```

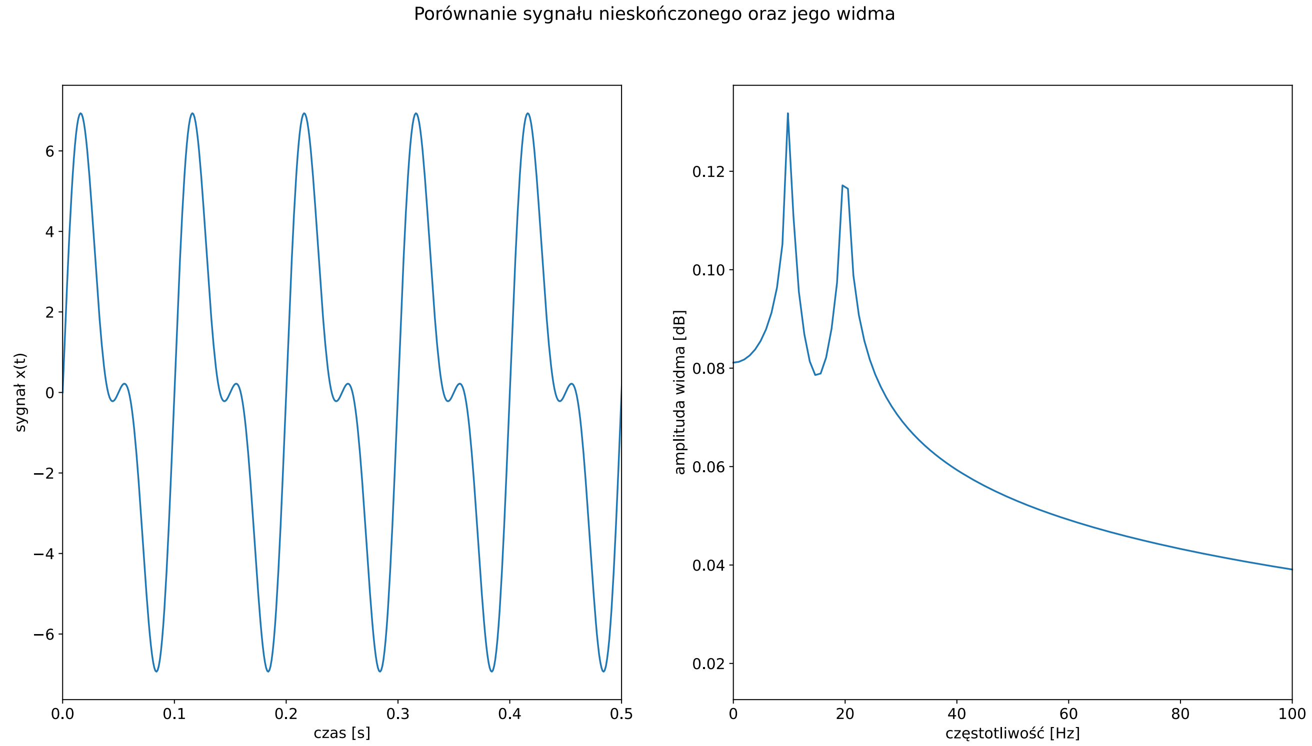
```
ax1.set(xlabel='t [s]', ylabel='sygnał x(t)')
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]')
plt.show()

signal = np.zeros(2*len(data))

for index, value in enumerate(data):
    signal[index*2] = value

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

widmo = 20 * np.log10(np.abs(np.fft.rfft(signal))) / (N/2)
fs = samplerate * 2
freq = np.fft.rfftfreq(len(signal), 1/fs)

fig, (ax1, ax2) = plt.subplots(1, 2)
fig.suptitle('Porównanie sygnału $\it{chord.wav}$ oraz jego widma z
wykorzystaniem $\it{upscalingu}$')
ax1.plot(time, signal)
ax1.set(xlabel='t [s]', ylabel='sygnał x(t)')
ax2.plot(freq, widmo)
ax2.set(xlabel='częstotliwość [Hz]', ylabel='amplituda widma [dB]')
plt.show()
```

3. Wyniki

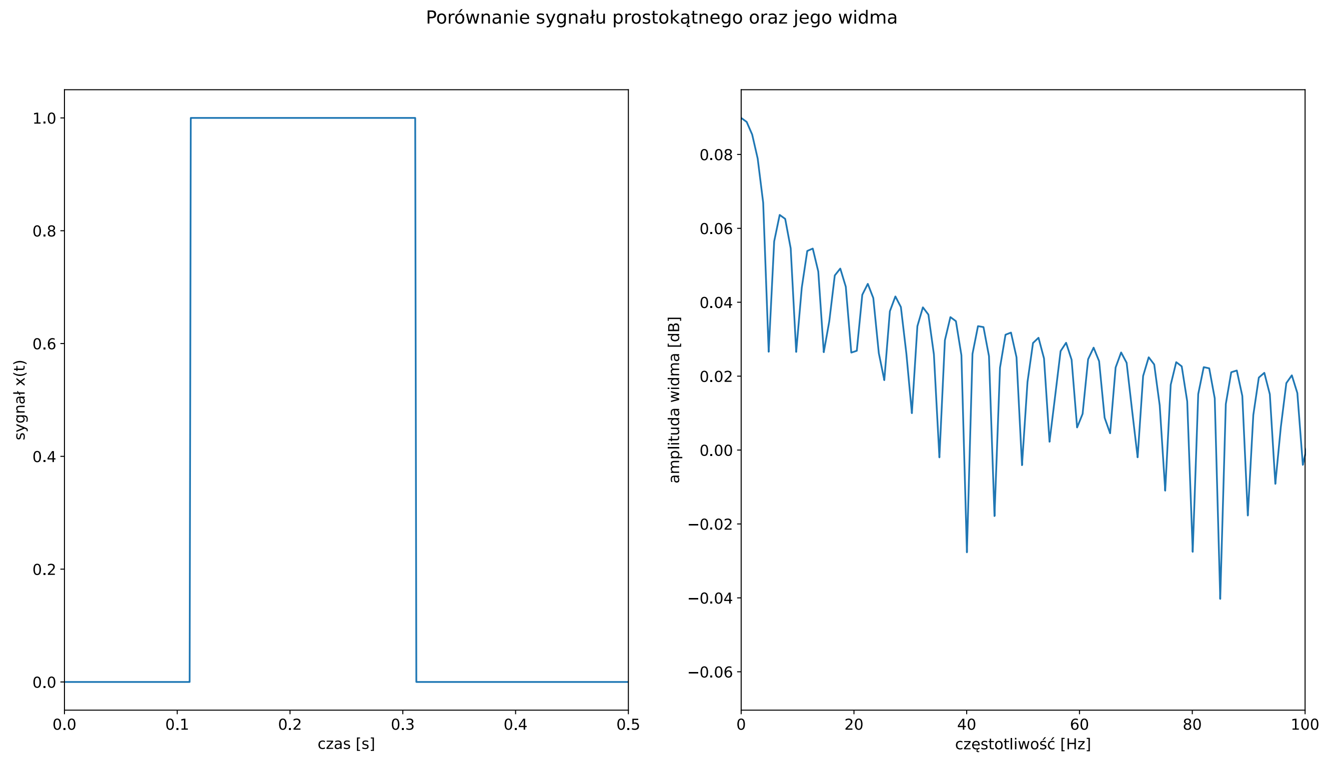
3.1 Porównanie sygnałów w funkcji czasu oraz widm

Poniższy wykres przedstawia sygnał nieskończony w dziedzinie czasu oraz jego widmo:

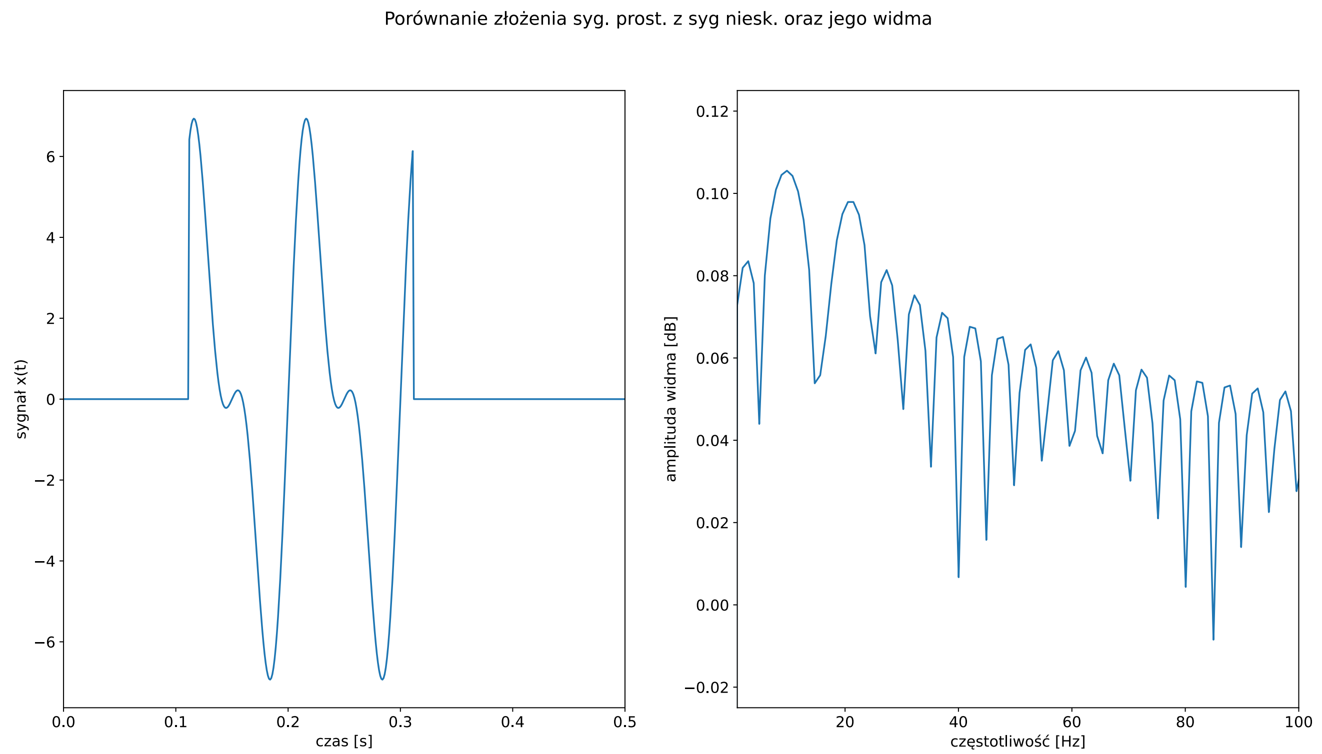
Wzór na sygnał: $x(t) = 5\sin(20\pi t) + 5\sin(40\pi t)$



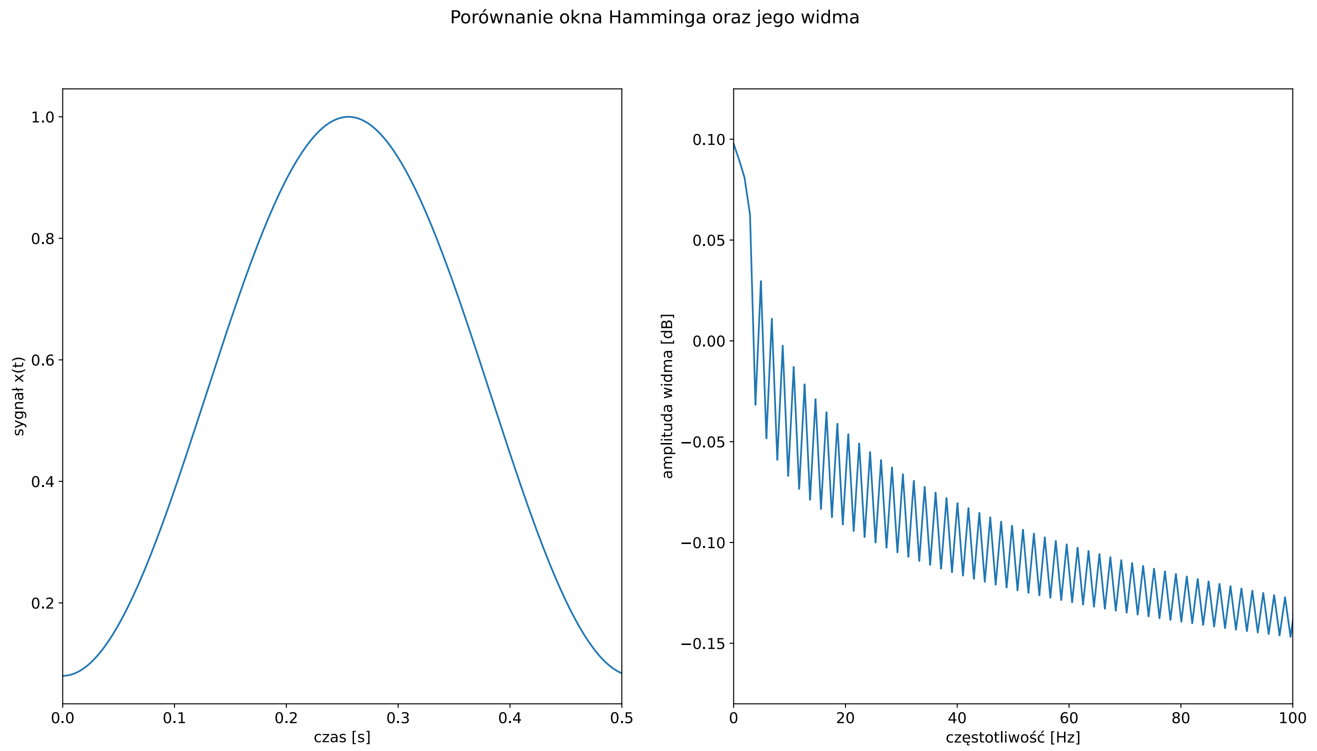
Poniższy wykres przedstawia sygnał prostokątny oraz jego widmo:



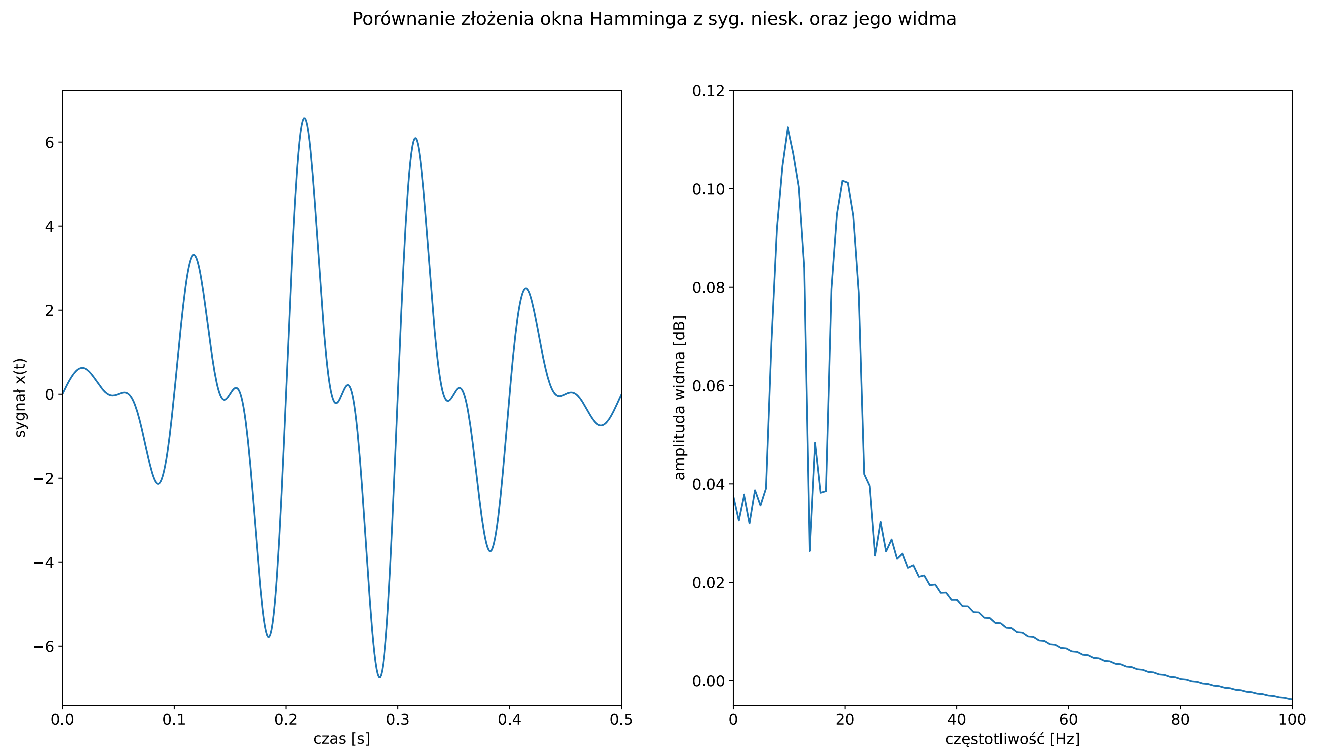
Poniższy wykres przedstawia iloczyn sygnału prostokątnego z sygnałem niesk. oraz jego widmo:



Poniższy wykres przedstawia okno Hamminga oraz jego widmo:

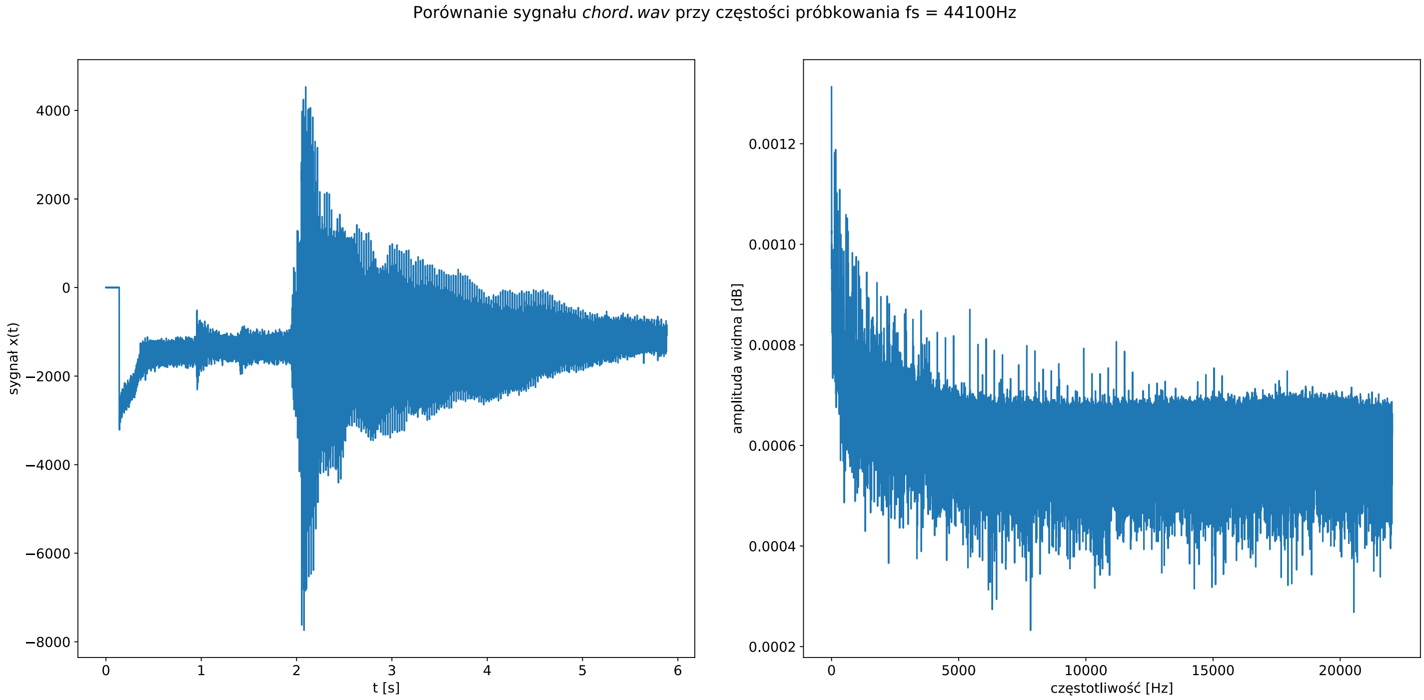


Poniższy wykres przedstawia iloczyn okna Hamminga i sygnału niesk. oraz jego widmo:

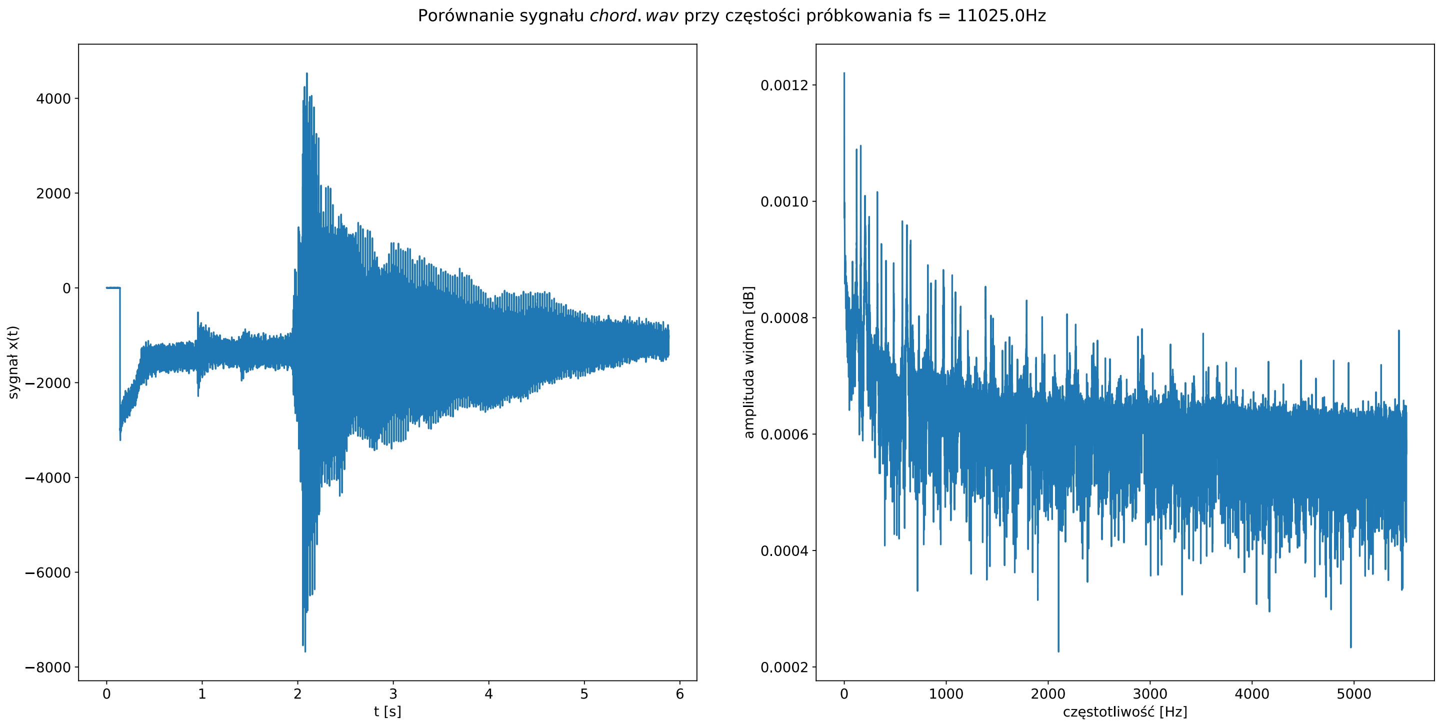


3.2 Twierdzenie o próbkowaniu. Aliasing.

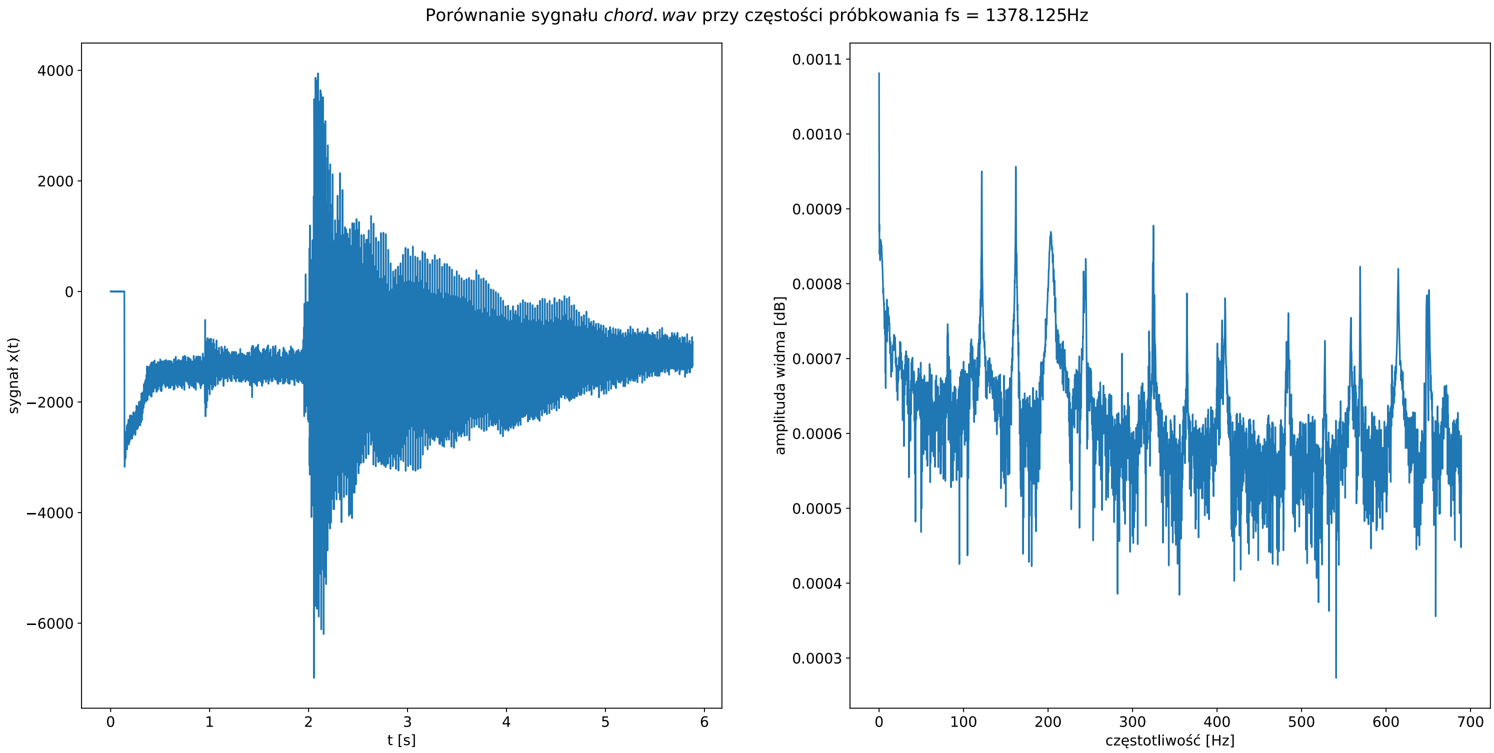
Porównanie sygnału *chord.wav* oraz jego widmo dla częstotliwości próbkowania $f_s = 44\ 100 Hz$ (2-krotna częstotliwość Nyquista) przedstawia poniższy wykres:



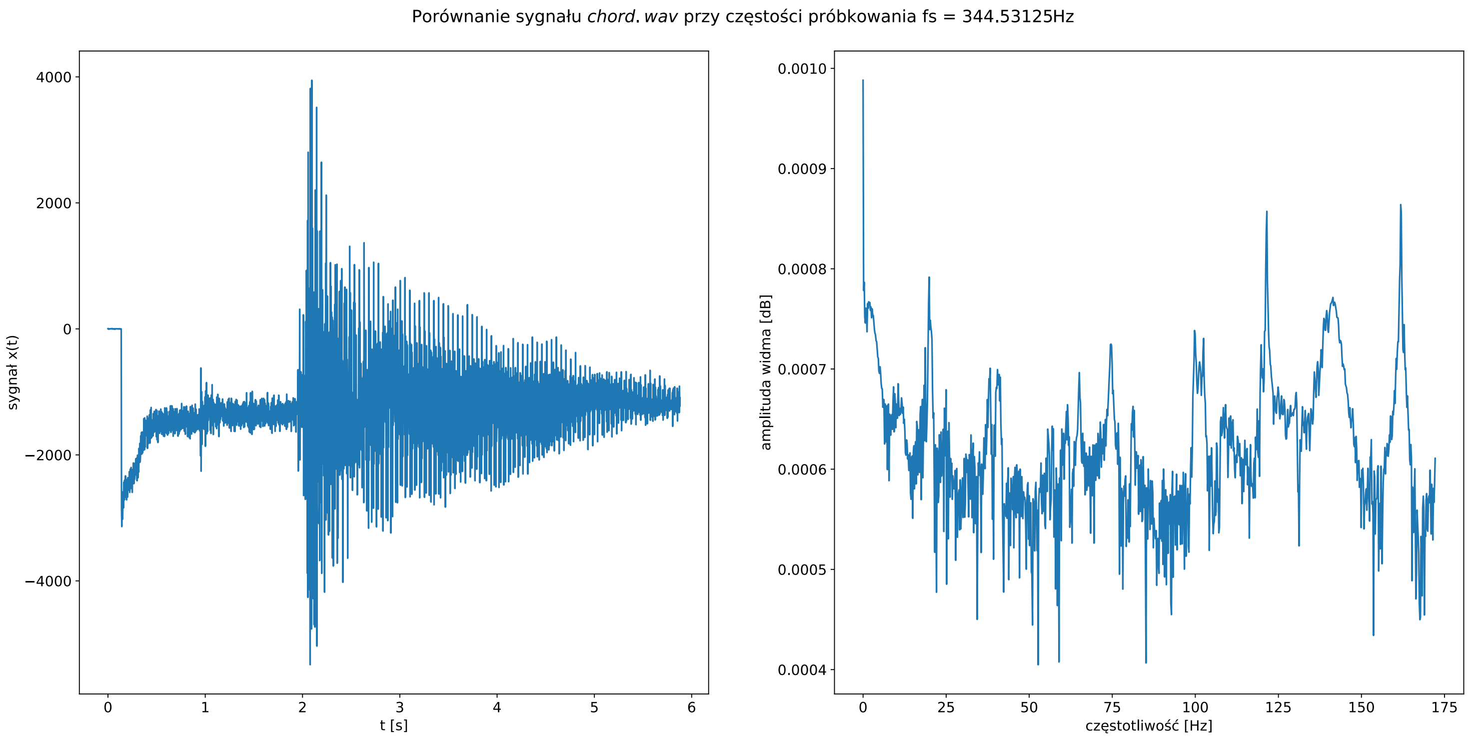
Porównanie sygnału *chord.wav* oraz jego widmo dla częstotliwości próbkowania $f_s = 11\ 025 Hz$ (4-krotna decymacja) przedstawia poniższy wykres:



Porównanie sygnału *chord.wav* oraz jego widmo dla częstotliwości próbkowania $f_s = 1378{,}125Hz$ (32-krotna decymacja) przedstawia poniższy wykres:



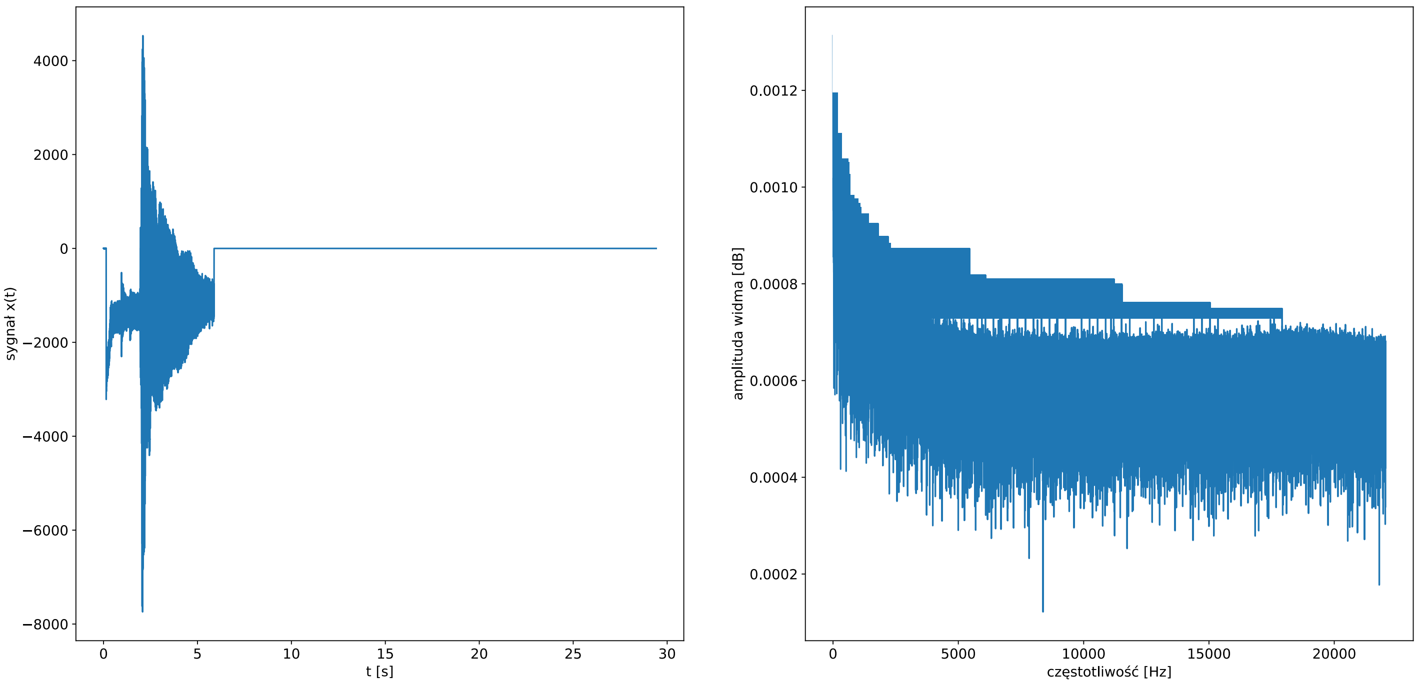
Porównanie sygnału *chord.wav* oraz jego widmo dla częstotliwości próbkowania $f_s = 344{,}53125Hz$ (128-krotna decymacja) przedstawia poniższy wykres:



3.3 Zero padding

Porównanie sygnału *chord.wav* oraz jego widmo dla częstotliwości próbkowania $f_s = 44\ 100Hz$ z wykorzystaniem zero paddingu (4x długość próbki) przedstawia poniższy wykres:

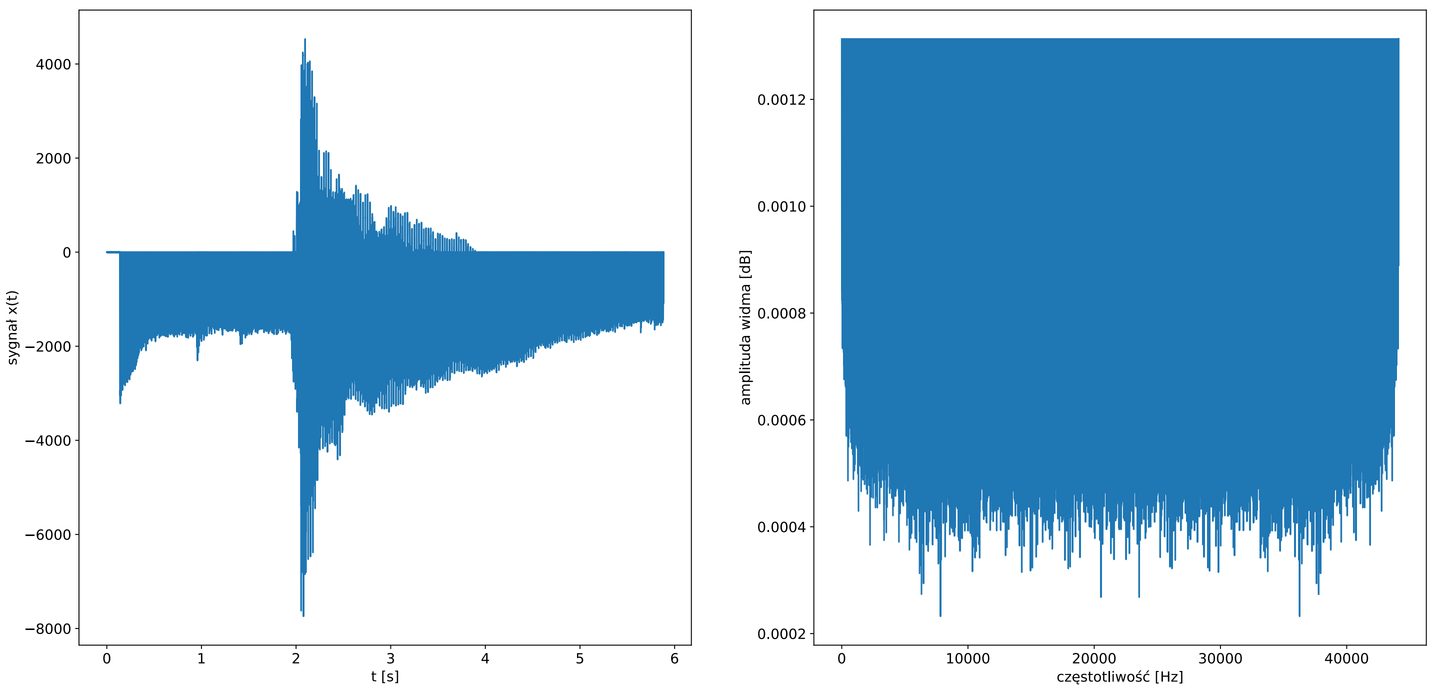
Porównanie sygnału $chord.wav$ oraz jego widma z wykorzystaniem $zero - padding$



3.4 Upsampling

Porównanie sygnału *chord.wav* oraz jego widmo z wykorzystaniem upsamplingu (wstawiono 0 przed co 2-gą próbką) przedstawia poniższy wykres:

Porównanie sygnału $chord.wav$ oraz jego widma z wykorzystaniem $upscalingu$



4. Wnioski

- Analiza widma Fouriera pozwala określić składowe częstotliwościowe sygnału.
- Okno Hamminga pozwala lepiej uwypuklić składowe częstotliwościowe sygnału.
- Nie da się jednocześnie zwiększyć rozdzielczości czasowej i częstotliwościowej.
- Aby zdobyć informację o składowej o częstotliwości f, potrzebujemy próbkować sygnał z częstotliwością równą co najmniej $2f$.
- Możemy zwiększyć częstotliwość próbkowania poprzez *zero-padding*, lub *upsampling,* lecz jest to jedynie pozorny zysk gdyż dodatkowe próbki są jedynie interpolowane na podstawie rzeczywistych danych.