Sprawozdanie z laboratorium

Przetwarzanie Sygnałów i Obrazów

Ćwiczenie numer: 2

Temat: Sygnały dyskretne. Próbkowanie.

Wykonujący ćwiczenie:

- Zaborowska Magda
- Wójtowicz Patryk

Studia dzienne I stopnia

Kierunek: Informatyka

Semestr: III Grupa zajęciowa: PS 12

Prowadzący ćwiczenie:
mgr inż. Dawid Najda

.....

OCENA

Data wykonania ćwiczenia

22.10.2021r.

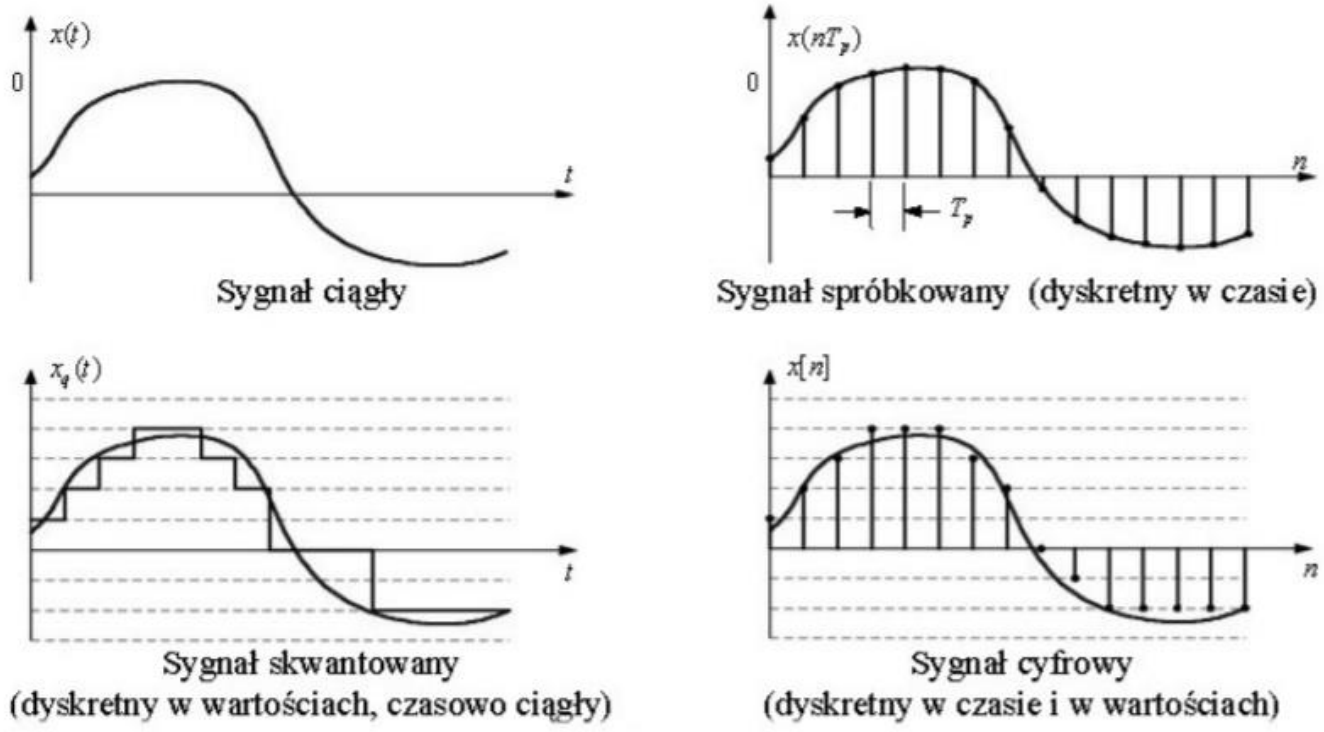
.....

Data i podpis prowadzącego

Teoria

Sygnał jest nośnikiem informacji o tym jak dowolna, mierzalna wielkość zmienia się w czasie.
Sygnał cyfrowy można określić jako ten, którego dziedziną i zbiorem wartości są wartości dyskretne. Natomiast **sygnał analogowy** ma ciągłą dziedzinę i zbiór wartości. **Sygnał dyskretny** powstaje przez próbkowanie sygnału ciągłego. Nie jest on funkcją, lecz ciągiem składającym się z próbek. W odróżnieniu od sygnału cyfrowego, próbki mogą przyjmować wartości z nieograniczonego lub ograniczonego zbioru. Aby zmienić sygnał analogowy na cyfrowy należy posłużyć się kwantyzacją. Są to przekształcenia, które zmniejszają precyzję sygnału tak, aby mogły zostać przetworzone przez jakieś urządzenie.
Twierdzenie Nyquista-Kotielnikova (twierdzenie o próbkowaniu) głosi, że jeśli sygnał ciągły S1 poddamy próbkowaniu z częstotliwością ponad dwukrotnie większą niż częstotliwość pierwotnego sygnału, wówczas za pomocą tych próbek będzie można odwzorować sygnał S1.[1-3]

$$\mathbf{fs > 2 \times f}$$



Rysunek 1 Zamiana sygnału analogowego w cyfrowy z użyciem kwantyzacji. [4]

Zadanie 1

Treść zadania:

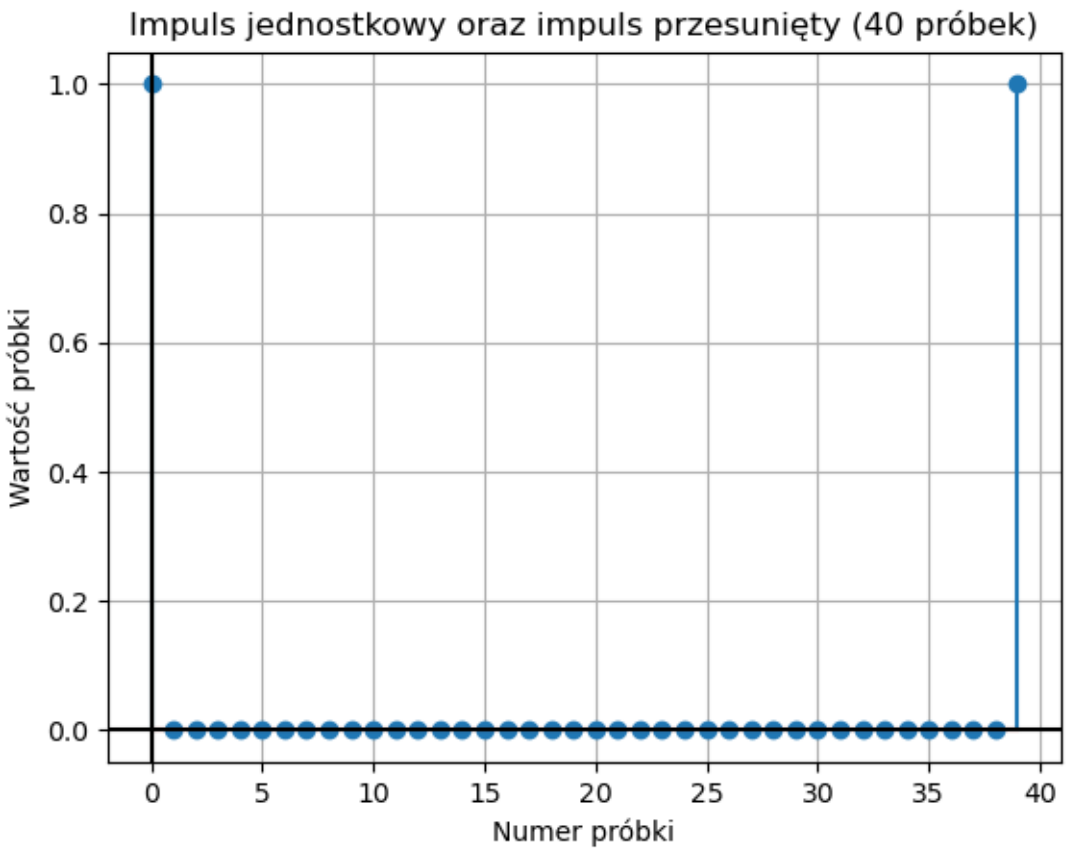
Wygenerować następujące sygnały dyskretne:
a) Impuls jednostkowy oraz impuls przesunięty o 40 próbek.
b) Przebiegi periodyczne (po 5 okresów, łącznie 200 próbek): sinusoidalny, piłokształtny, prostokątny. Wygenerowane sygnały przedstawić na wykresach w tym samym oknie. Wskazówka: użyć funkcji sawtooth, square z pakietu scipy.signal.
c) Szum gaussowski (200 próbek) dla parametrów: $\mu = 0$ oraz $\sigma 2 = 0.5$. Wskazówka: użyć funkcji random.normal z pakietu numpy.

A) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
from scipy import signal as si # Biblioteka scipy importowanie signala jako si

x=np.zeros(40); # Tworzy tablice o 40 wartościach z domyślnie wpisanymi zerami
x[0]=1 # Dla pierwzej wartości w tablicy przypisanie jedynki
x[39]=1 # Dla ostatniej wartośći w tablicy przypisanie jedynki

plt.stem(x) # Rysuje linie pod punktami (w tym przypadku widoczne dla ostatniej wartości)
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Impuls jednostkowy oraz impuls przesunięty (40 próbek)') # Tytuł wykresu
plt.xlabel("Numer próbki") # Podpis osi x
plt.ylabel("Wartość próbki") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 2 Kod do zadania 1.

Wynik:



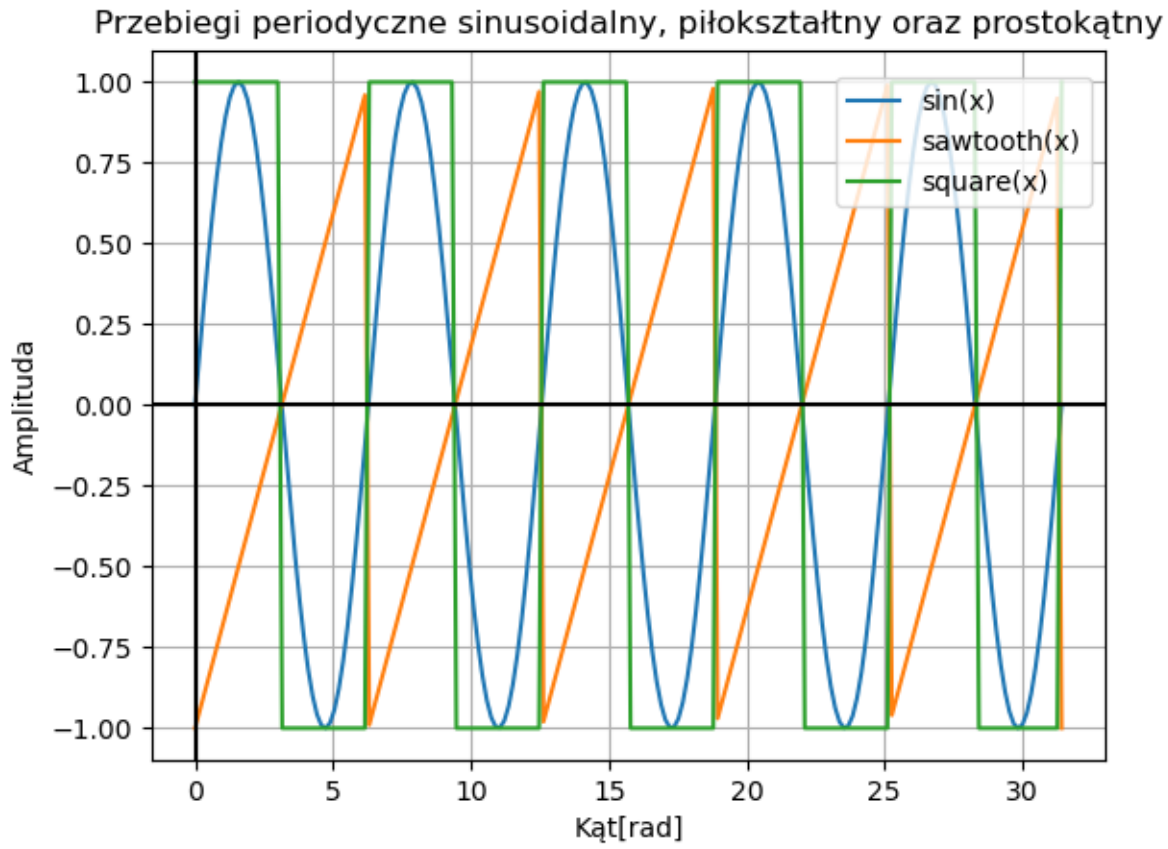
Wykres 1 Impuls jednostkowy oraz impuls przesunięty o 40 próbek.

B) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
from scipy import signal as si # Biblioteka scipy importowanie signala jako si

x = np.linspace(0,5*2*np.pi,200) # Określenie dziedziny <0,10π>, 200 pomiarów w tym zakresie
y1 = np.sin(x) # Wzór funkcji y1 (sinusoidalny)
y2 = si.sawtooth(x) # Wzór funkcji y2 (piłokształtny)
y3 = si.square(x) # Wzór funkcji y3 (prostokątny)

plt.plot(x,y1) # Rysowanie w zakresie zmiennej x funkcji y1
plt.plot(x,y2) # Rysowanie w zakresie zmiennej x funkcji y2
plt.plot(x,y3) # Rysowanie w zakresie zmiennej x funkcji y3
plt.legend(['sin(x)','sawtooth(x)','square(x)'],bbox_to_anchor=(1, 0.87),loc="center right")
# Stworzenie legendy i ustalenie jej lokalizacji na wykresie
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Przebiegi periodyczne sinusoidalny, piłokształtny oraz prostokątny') # Tytuł wykresu
plt.xlabel("Kąt[rad]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 3 Kod do zadania 1B.

Wynik:



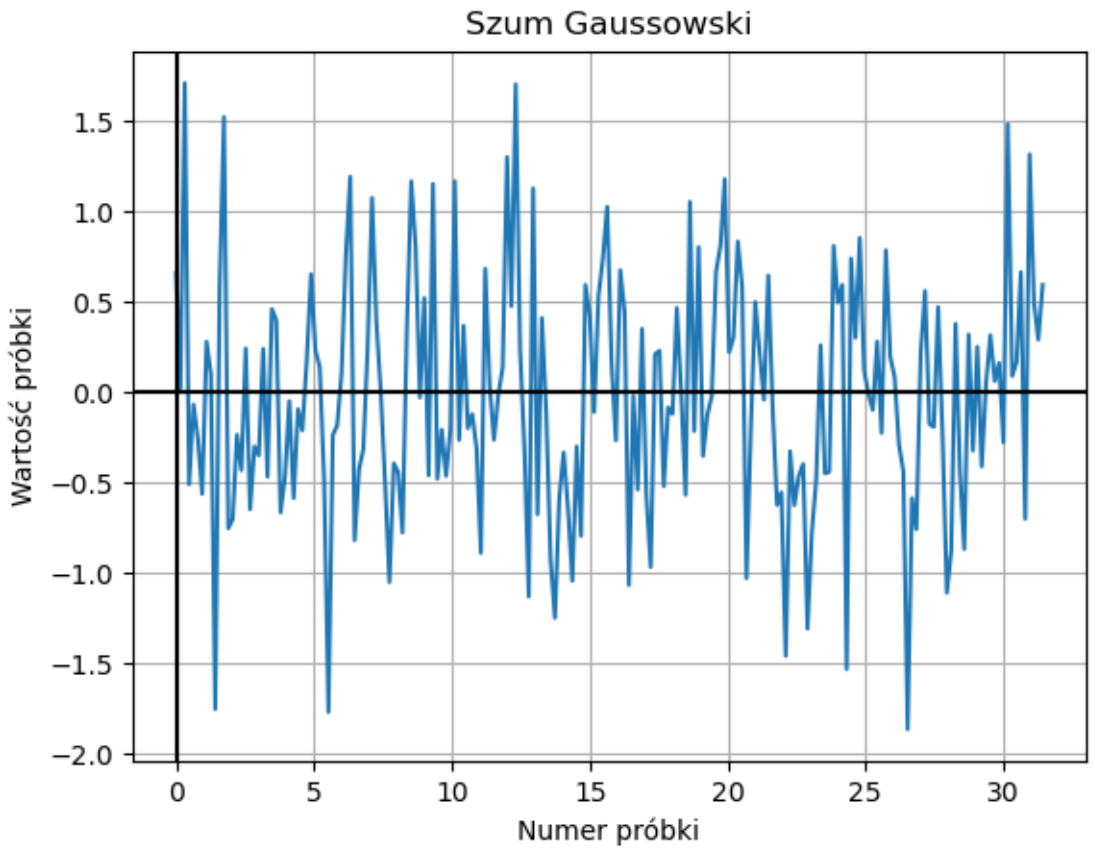
Wykres 2 Przebiegi periodyczne: sinusoidalny, piłokształtny, prostokątny.

C) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
from scipy import signal as si # Biblioteka scipy importowanie signala jako si

x = np.linspace(0,5*2*np.pi,200) # Określenie dziedziny <0,10π>, 200 pomiarów w tym zakresie
y = np.random.normal(0,np.sqrt(0.5),size = (200)) # Wzór funkcji y

plt.plot(x,y) # Rysowanie w zakresie zmiennej x funkcji y
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Szum Gaussowski') # Tytuł wykresu
plt.xlabel("Numer próbki") # Podpis osi x
plt.ylabel("Wartość próbki") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 4 Kod do zadania 1C.

Wynik:



Wykres 3 Szum gaussowski (200 próbek) dla parametrów: μ = 0 oraz σ=sqrt(0,5)

Zadanie 2

Treść zadania:

Wygeneruj dyskretny sygnał sinusoidalny o amplitudzie A, częstotliwości f, przesunięciu fazowym φ dla częstotliwości próbkowania fs.

a) Dla każdego ze wspomnianych parametrów sporządzić po trzy przykładowe przebiegi zmieniając odpowiednio ich wartości.

b) Przeprowadzić testy odsłuchowe.

c) Jaki wpływ na wrażenia słuchowe mają amplituda, częstotliwość i faza?

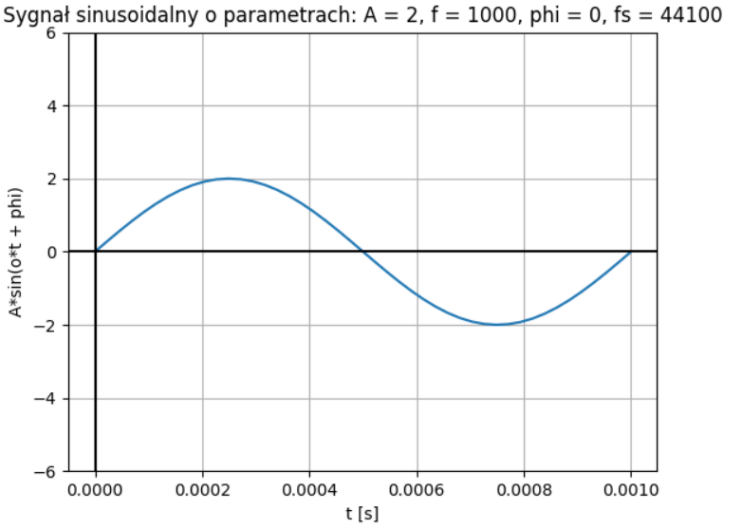
Realizacja w kodzie:

W kodzie dla poszczególnych zadań zmieniano odpowiednio wartość amplitudy, częstotliwości i przesunięcia fazowego

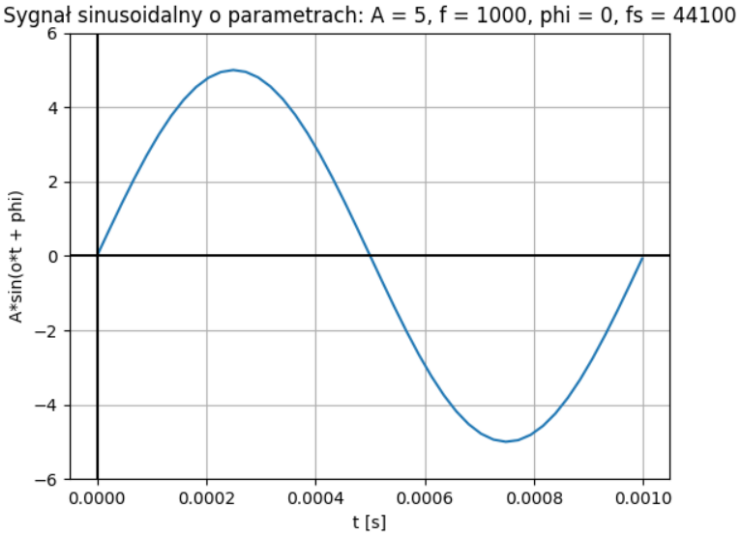
```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
import soundfile as sf # Biblioteka soundfile jak sf

A=1 # Wartość amplitudy
f=1000  # Wartość częstotliwości
T=1/f   # wartość okresu
phi=0   # Wartość przesunięcia Phi
fs=44100  # Wartość częstotliwości próbkowania
Ts = 1/fs   # Wartość okresu próbkowania
o = 2*np.pi*f   # Wartość omegi

t = np.arange(0,T,Ts)   #dziedzina <0 , 0.001> próbkowanie co wartość Ts
y=A * np.sin(o*t + phi) #Wzór funkcji sinuzoidalnej
plt.plot(t,y) # Rysowanie w zakresie zmiennej x funkcji y
plt.ylim(-6,6)  #Skalowanie osi OY
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Sygnał sinusoidalny o parametrach: A = 1, f = 1000, phi = 0, fs = 44100') # Tytuł wykresu
plt.xlabel("t [s]") # Podpis osi x
plt.ylabel("A*sin(o*t + phi") # Podpis osi y
sf.write('wynik.wav',y,fs) # Eksportowanie do pliku wav funkcji y o próbkowaniu Fs
plt.show() # Wywołanie wykresu
```

Rysunek 5 Kod do zadania 2.

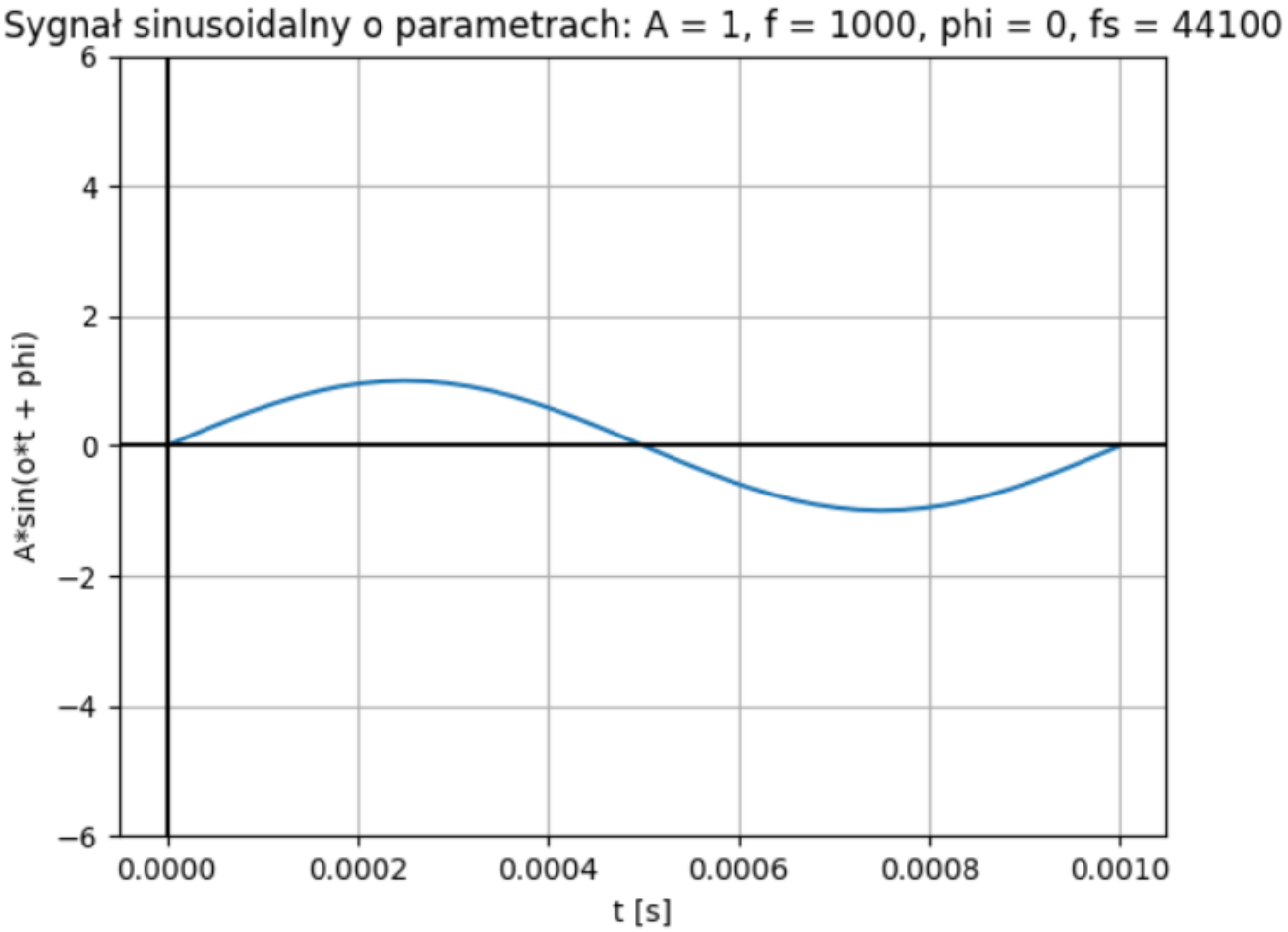
Wyniki zmian amplitudy:



*Wykres 6 Sygnał sinusoidalny o parametrach: **A = 2**, f = 1000, phi = 0, fs = 44100*

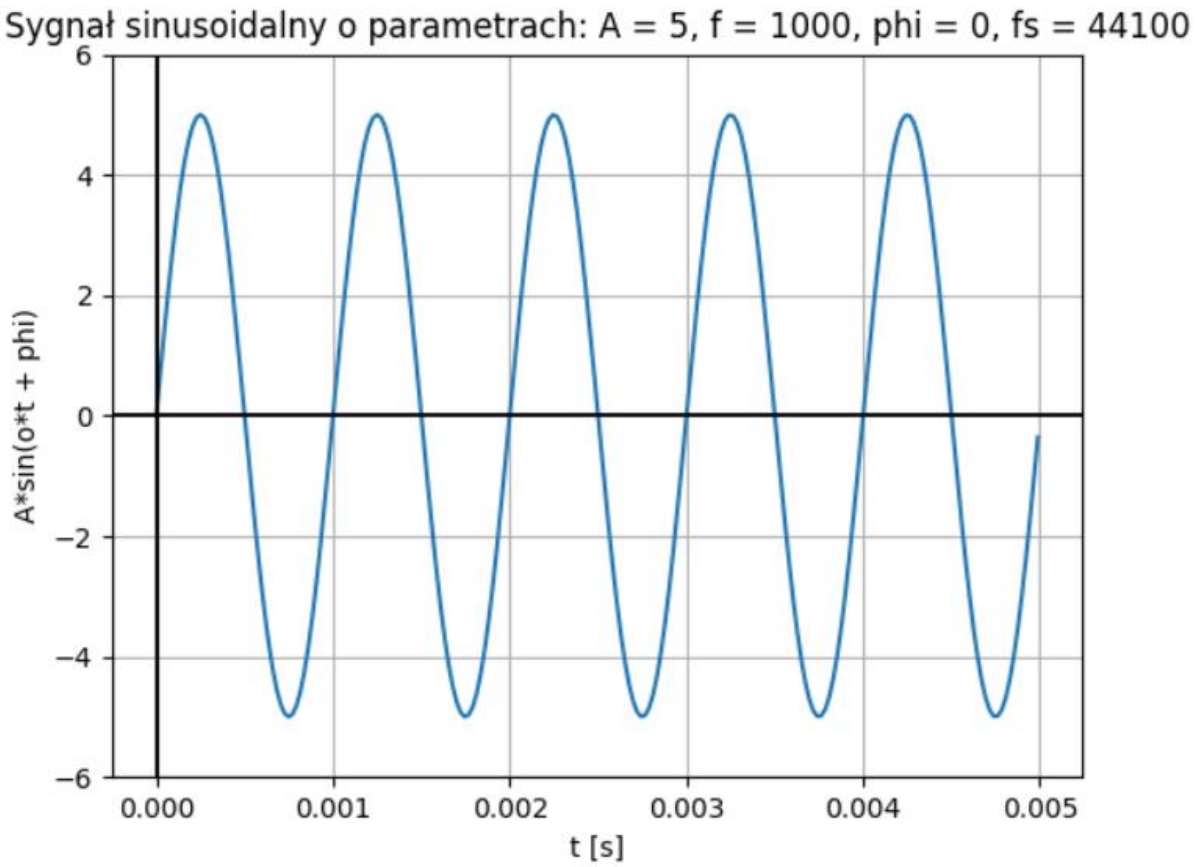


*Wykres 5 Sygnał sinusoidalny o parametrach: **A = 5**, f = 1000, phi = 0, fs = 44100*

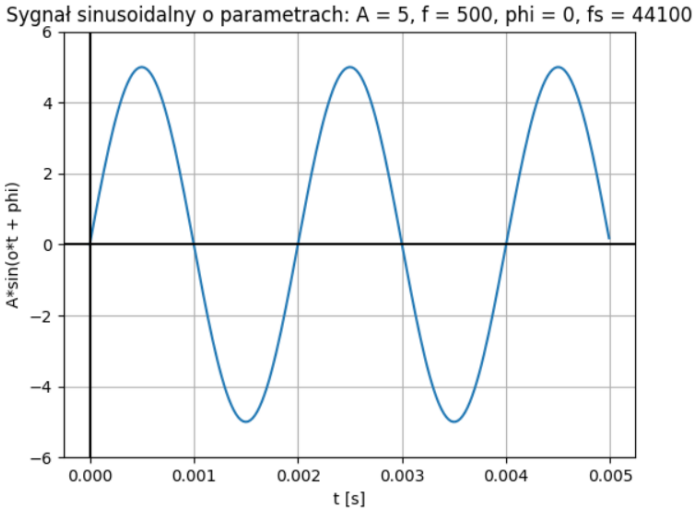


*Wykres 4 Sygnał sinusoidalny o parametrach**: A = 1**, f = 1000, phi = 0, fs = 44100*

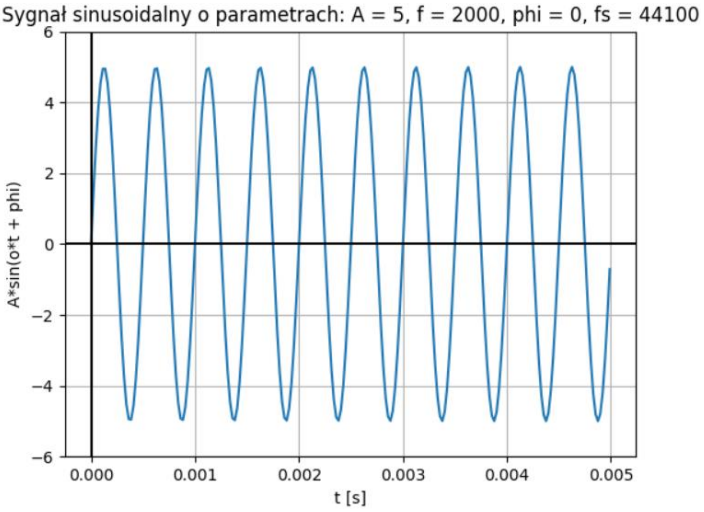
Wyniki zmian częstotliwości:



*Wykres 7 Sygnał sinusoidalny o parametrach: A = 5, **f = 1000**, phi = 0, fs = 44100*

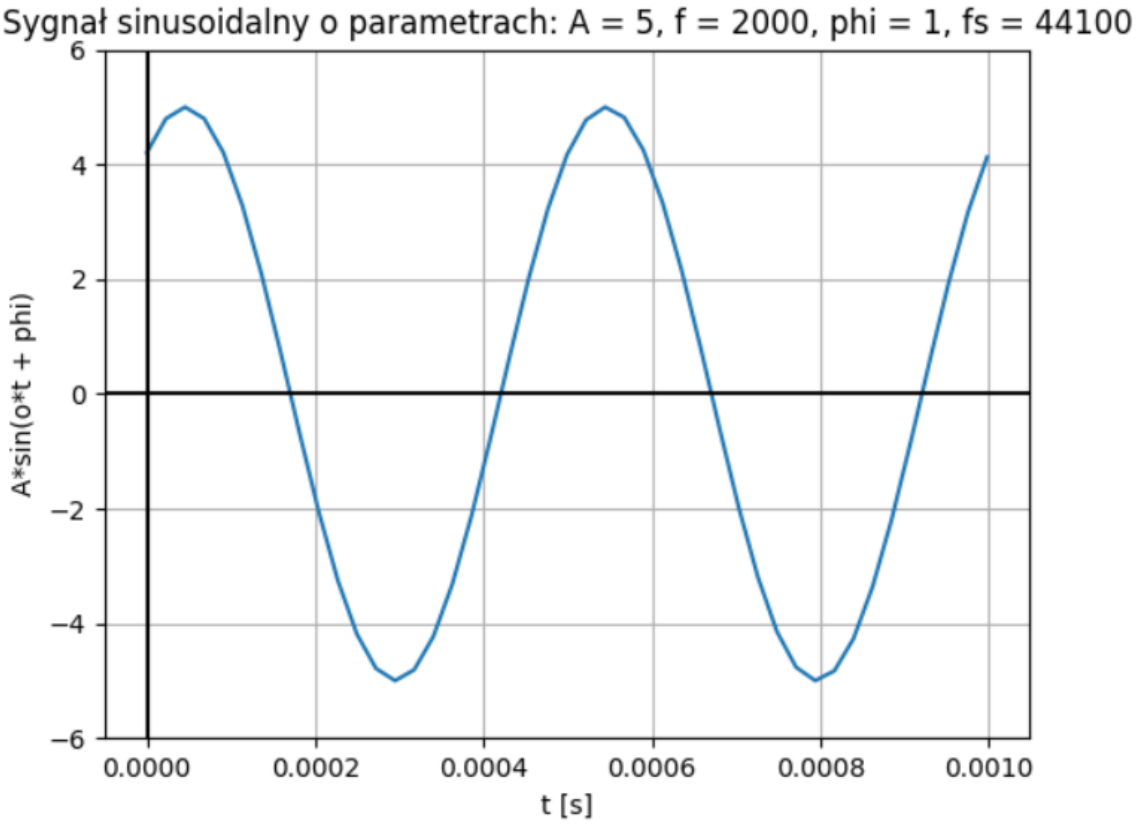


*Wykres 8 Sygnał sinusoidalny o parametrach: A = 5, **f = 500**, phi = 0, fs = 44100*

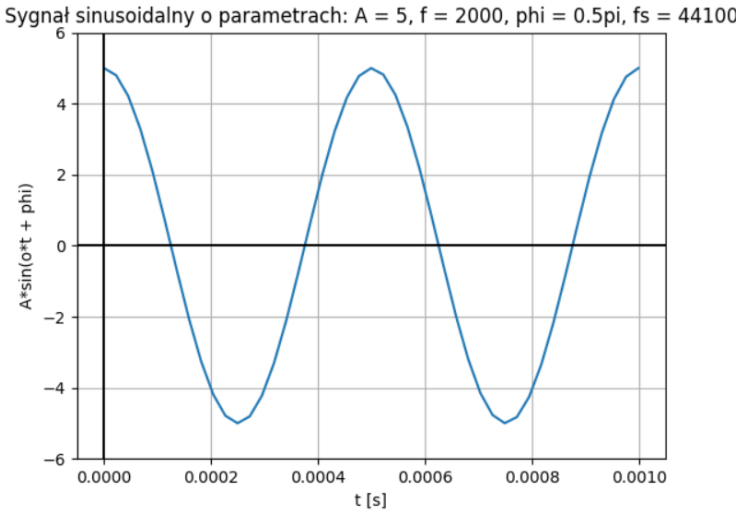


*Wykres 9 Sygnał sinusoidalny o parametrach: A = 5, **f = 2000**, phi = 0, fs = 44100*

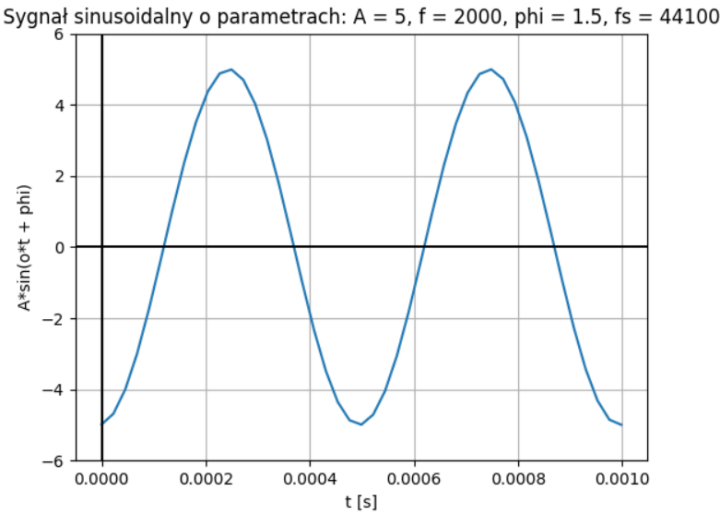
Wyniki zmian przesunięcia fazowego:



Wykres 10 Sygnał sinusoidalny o parametrach: A = 5, f = 2000, ***phi = 1****, fs = 44100*



Wykres 11 Sygnał sinusoidalny o parametrach: A = 5, f = 2000, ***phi = 0.5pi****, fs = 44100*



Wykres 9 Sygnał sinusoidalny o parametrach: A = 5, f = 2000, ***phi = 1.5****, fs = 44100*

Zadanie 3

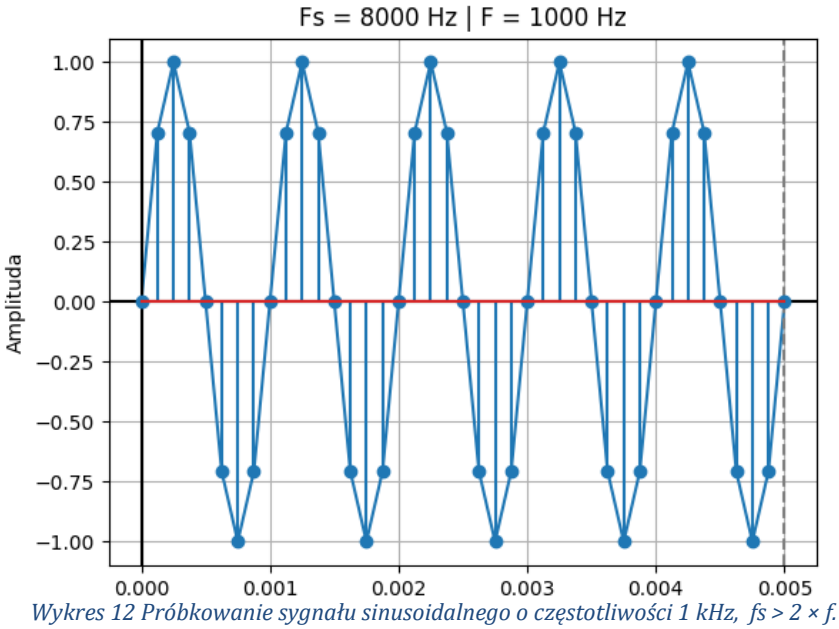
Treść zadania:
Zasymulować próbkowanie sygnału sinusoidalnego o częstotliwości 1 kHz, zastosować różne wartości tempa próbkowania:
a) fs > 2 × f
b) fs = 2 × f
c) fs < 2 × f
Porównać przebiegi na wykresach (stosując tę samą skalę czasu). W którym z powyższych trzech przypadków warunek Nyquista był spełniony? Zakładając ustaloną wartość fs, jakie zakresy zmienności parametrów f oraz φ wystarczają do spróbkowania wszystkich możliwych sygnałów o zadanej amplitudzie?

A) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt

A = 1 # Amplituda
F = 1000 # Częstotliwość
Fs = 8000 # Częstotliwość próbkowania
Ts = 1/Fs # Częstotliwość próbkowania w sekundach
O = 2*np.pi*F # Omega
Phi = 0 # Przesunięcie fazowe
x = np.arange(0,0.0051,Ts) # Określenie dziedziny <0,0.0051>, Ts pomiarów w tym zakresie
y = A * np.sin(O*x + Phi) # Wzór funkcji y

plt.plot(x,y) # Rysowanie w zakresie zmiennej x funkcji y
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
plt.grid(True,which='both') # Włączenie siatki
plt.stem(x,y,'-',use_line_collection="true") # Włączenie lini punktów wykresu
plt.title('Fs = 8000 Hz | F = 1000 Hz') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 6 Kod do zadania 3A.

Wynik:



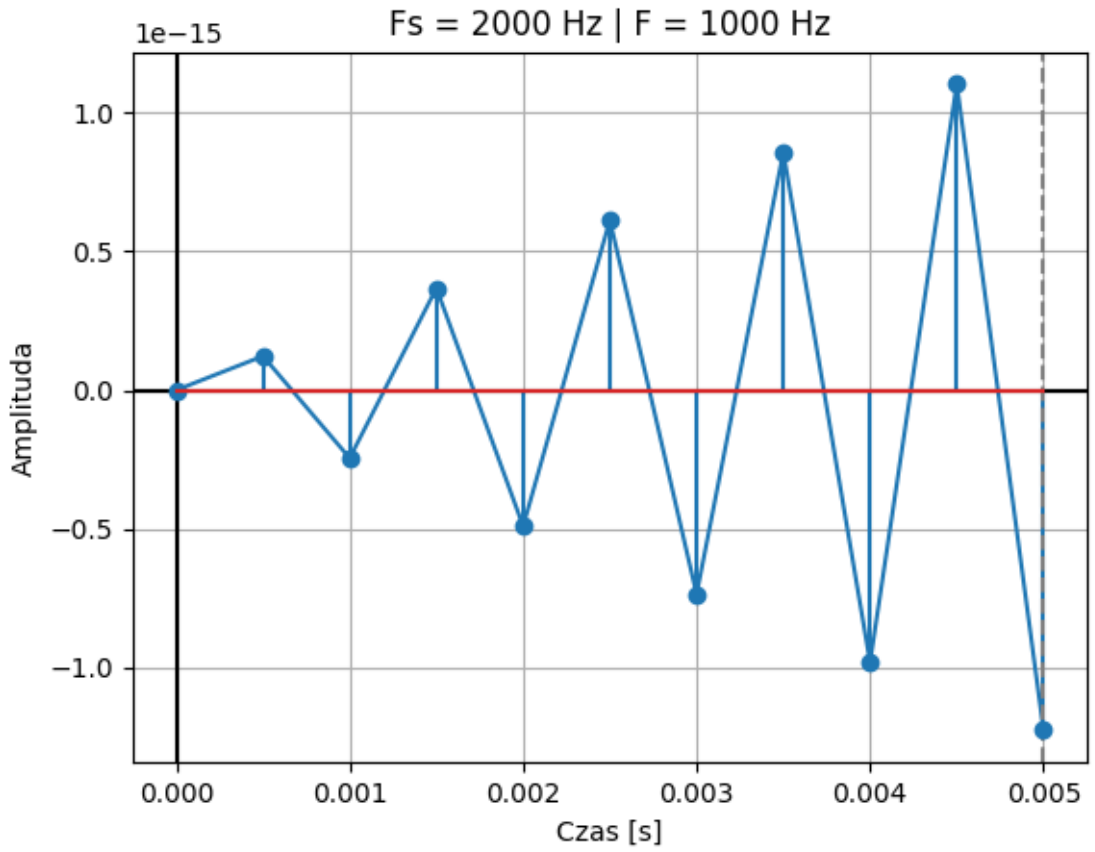
Wykres 12 Próbkowanie sygnału sinusoidalnego o częstotliwości 1 kHz, fs > 2 × f.

B) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt

A = 1 # Amplituda
F = 1000 # Częstotliwość
Fs = 2000 # Częstotliwość próbkowania
Ts = 1/Fs # Częstotliwość próbkowania w sekundach
O = 2*np.pi*F # Omega
Phi = 0 # Przesunięcie fazowe
x = np.arange(0,0.0051,Ts) # Określenie dziedziny <0,0.0051>, Ts pomiarów w tym zakresie
y = A * np.sin(O*x + Phi) # Wzór funkcji y

plt.plot(x,y) # Rysowanie w zakresie zmiennej x funkcji y
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.axvline(x=0.005,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.005 kolorem szarym linią przerywaną
plt.grid(True,which='both') # Włączenie siatki
plt.stem(x,y,'-',use_line_collection="true") # Włączenie lini punktów wykresu
plt.title('Fs = 2000 Hz | F = 1000 Hz') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 7 Kod do zadania 3B

Wynik:



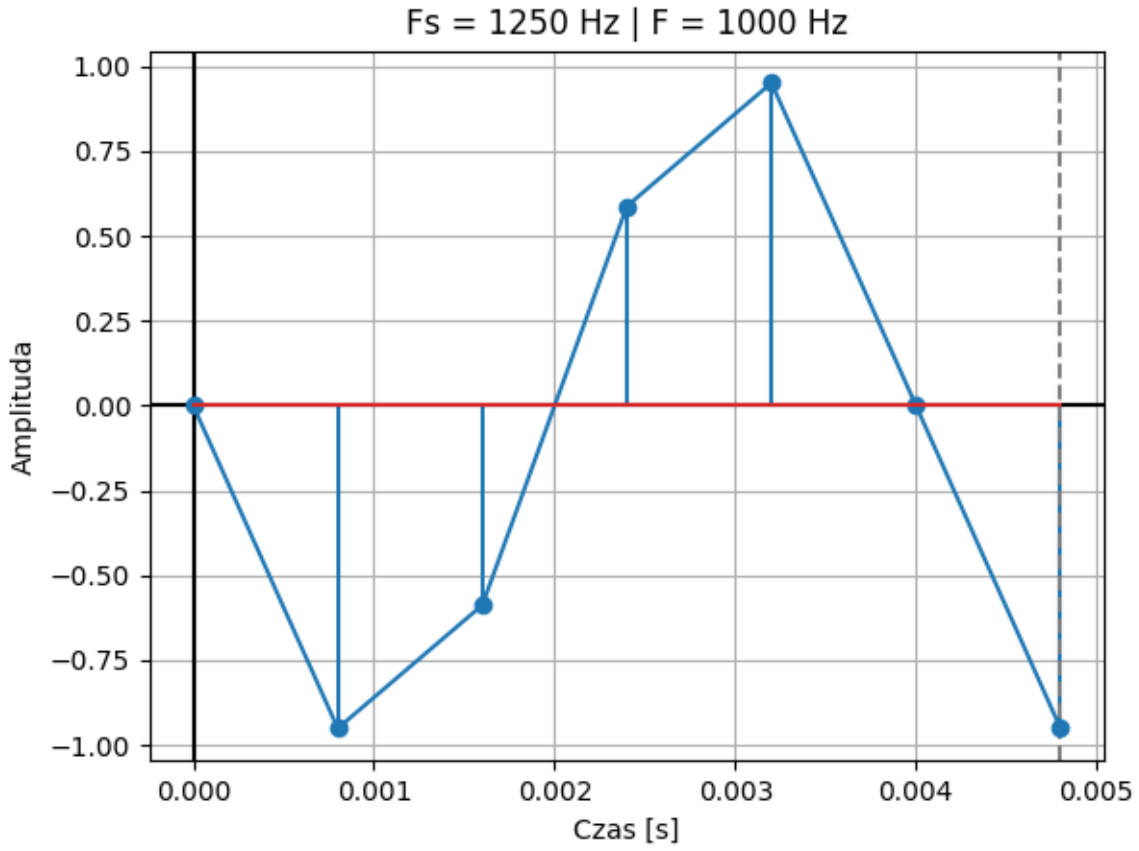
Wykres 13 Próbkowanie sygnału sinusoidalnego o częstotliwości 1 kHz, fs = 2 × f.

C) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt

A = 1 # Amplituda
F = 1000 # Częstotliwość
Fs = 1250 # Częstotliwość próbkowania
Ts = 1/Fs # Częstotliwość próbkowania w sekundach
O = 2*np.pi*F # Omega
Phi = 0 # Przesunięcie fazowe
x = np.arange(0,0.0051,Ts) # Określenie dziedziny <0,0.0051>, Ts pomiarów w tym zakresie
y = A * np.sin(O*x + Phi) # Wzór funkcji y

plt.plot(x,y) # Rysowanie w zakresie zmiennej x funkcji y
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.axvline(x=0.0048,color = "grey",linestyle = "--") # Zaznaczeni lini x=0.0048 kolorem szarym linią przerywaną
plt.grid(True,which='both') # Włączenie siatki
plt.stem(x,y,'-',use_line_collection="true") # Włączenie lini punktów wykresu
plt.title('Fs = 1250 Hz | F = 1000 Hz') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu
```

Rysunek 8 Kod do zadania 3C.

Wynik:



Wykres 14 Próbkowanie sygnału sinusoidalnego o częstotliwości 1 kHz, $fs < 2 \times f$.

Zadanie 4

Treść zadania:
Wykorzystując mikrofon oraz dowolną aplikację do rejestracji dźwięków (np. Audacity, Rejestrator Windows, etc.) zarejestrować krótką wypowiedź np. odliczanie do trzech. Zarejestrowany sygnał zapisać do pliku wav (format mono, częstotliwość próbkowania 44.1kHz). Otworzyć plik w wybranym środowisku, a następnie odwrócić w czasie kolejność zarejestrowanych próbek.
a) Wykreślić oryginalnie zrejestrowany sygnał oraz sygnał odwrócony w czasie,
b) Dokonać superpozycji oryginalnego sygnału mowy z sygnałem szumu gaussowskiego (μ = 0, σ 2 = 1). Sporządzić wykresy czasowe sygnału oryginalnego i zakłóconego. Porównać słuchowo sygnał oryginalny z sygnałem zakłóconym.

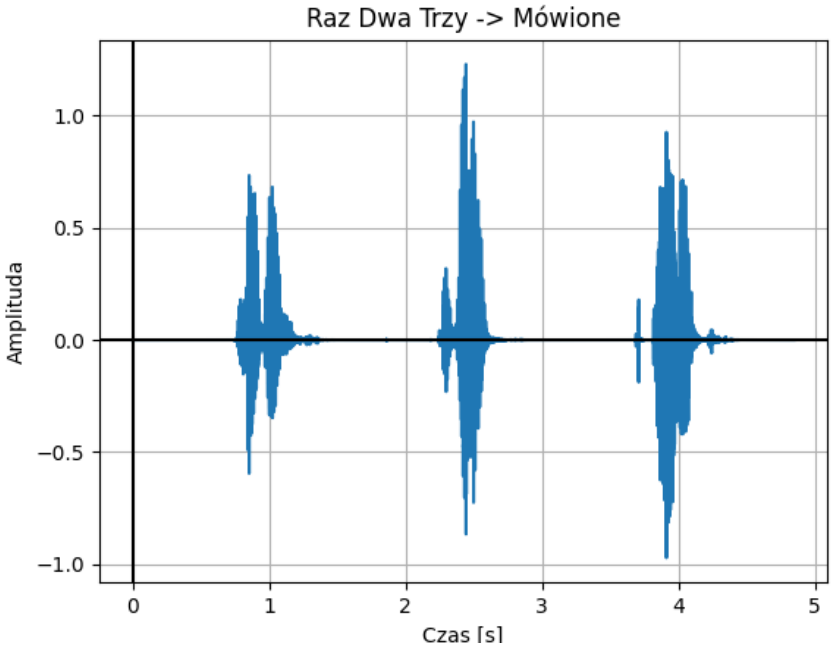
A) Realizacja w kodzie:

- Sygnał zwykły:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
import soundfile as sf # Biblioteka soundfile jak sf
import wave as w # Biblioteka wave jak w

sound = w.open("Razdwatrzy.wav") # Otwarcie pliku audio w zmiennej
frames_frequency = sound.getframerate()*2 # Wyliczanie częstotliwości klatek
record = np.frombuffer(sound.readframes(-1),dtype="int16")/2000 # Wczytanie dźwięku do zmiennej z zmiennej z plikiem audio
x = np.arange(0, record.size/frames_frequency, 1/frames_frequency) # Stworzenie zakresu o długości <0;record.size/frames_frequency>
#z badaniem próbek 1/frames_frequency

plt.plot(x,record) # Rysowanie w zakresie zmiennej x funkcji record
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Raz Dwa Trzy -> Mówione') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu

sf.write('Razdwatrzy.wav',record,frames_frequency) # Eksportowanie do pliku wav funkcji record o próbkowaniu frames_frequency
```

Rysunek 9 Kod do zadania 4A.

Wynik:



Wykres 15 Sygnał zarejestrowany (mówione "raz dwa trzy").

- Sygnał odwrócony:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
import soundfile as sf # Biblioteka soundfile jak sf
import wave as w # Biblioteka wave jak w

sound = w.open("Razdwatrzy.wav") # Otwarcie pliku audio w zmiennej
frames_frequency = sound.getframerate()*2 # Wyliczanie częstotliwości klatek
record = np.frombuffer(sound.readframes(-1),dtype="int16")/2000 # Wczytanie dźwięku do zmiennej z zmiennej z plikiem audio
record = np.flip(record) # Odwrócenie nagrania
x = np.arange(0, record.size/frames_frequency, 1/frames_frequency) # Stworzenie zakresu o długości <0;record.size/frames_frequency>
#z badaniem próbek 1/frames_frequency

plt.plot(x,record) # Rysowanie w zakresie zmiennej x funkcji record
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Raz Dwa Trzy -> Mówione') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu

sf.write('Razdwatrzy - od tyłu.wav',record,frames_frequency) # Eksportowanie do pliku wav funkcji record o próbkowaniu
# frames_frequency
```

Rysunek 10 Kod do zadania 4A.

Wynik:

Wykres 16 Sygnał odwrócony (mówione "raz dwa trzy").

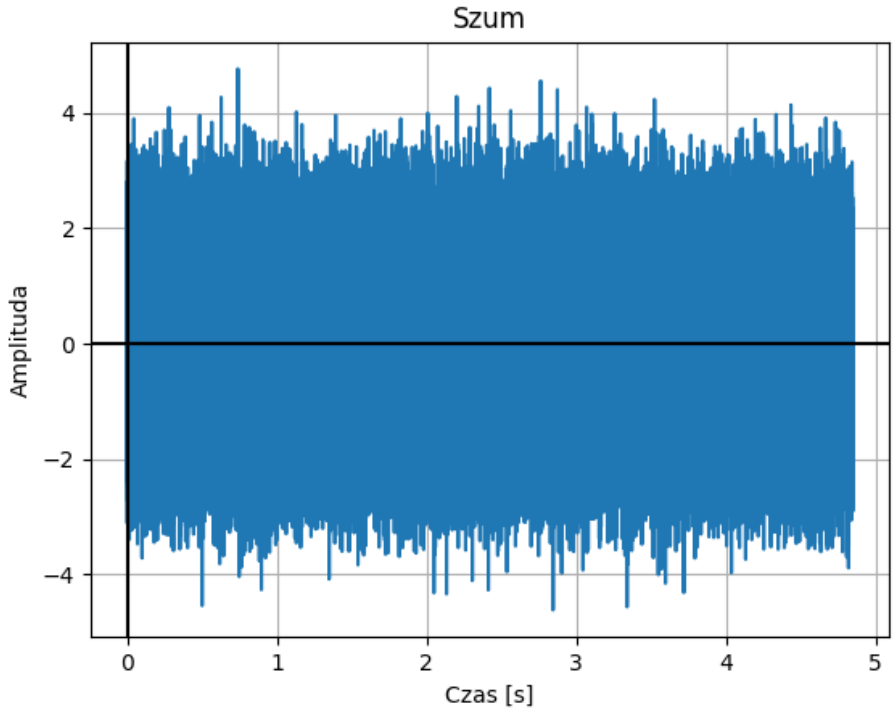
B) Realizacja w kodzie:

```python
import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
import soundfile as sf # Biblioteka soundfile jak sf
import wave as w # Biblioteka wave jak w

SNR = 3 # Współczynnik sygnał-szum z treści zadania

sound = w.open("Razdwatrzy.wav") # Otwarcie pliku audio w zmiennej
frames_frequency = sound.getframerate()*2 # Wyliczanie częstotliwości klatek
record = np.frombuffer(sound.readframes(-1),dtype="int16")/2000 # Wczytanie dźwięku do zmiennej z zmiennej z plikiem audio
x = np.arange(0, record.size/frames_frequency, 1/frames_frequency) # Stworzenie zakresu o długości <0;record.size/frames_frequency>
noise = np.random.normal(0,1,record.size) # Stworzenie randomowego szumu do połączenia z audio

plt.plot(x,noise) # Rysowanie w zakresie zmiennej x funkcji noise
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Szum') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu

sf.write('Razdwatrzy - szum.wav',noise,frames_frequency) # Eksportowanie do pliku wav funkcji noise o próbkowaniu frames_frequency
record = record + noise/(5 * SNR)

plt.plot(x,record) # Rysowanie w zakresie zmiennej x funkcji record
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Nagranie "Razdwatrzy" + Szum') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
plt.show() # Wywołanie wykresu

sf.write('Razdwatrzy - nagranie+szum.wav',record,frames_frequency) # Eksportowanie do pliku wav funkcji noise o próbkowaniu
# frames_frequency
```

Rysunek 11 Kod do zadania 4B.

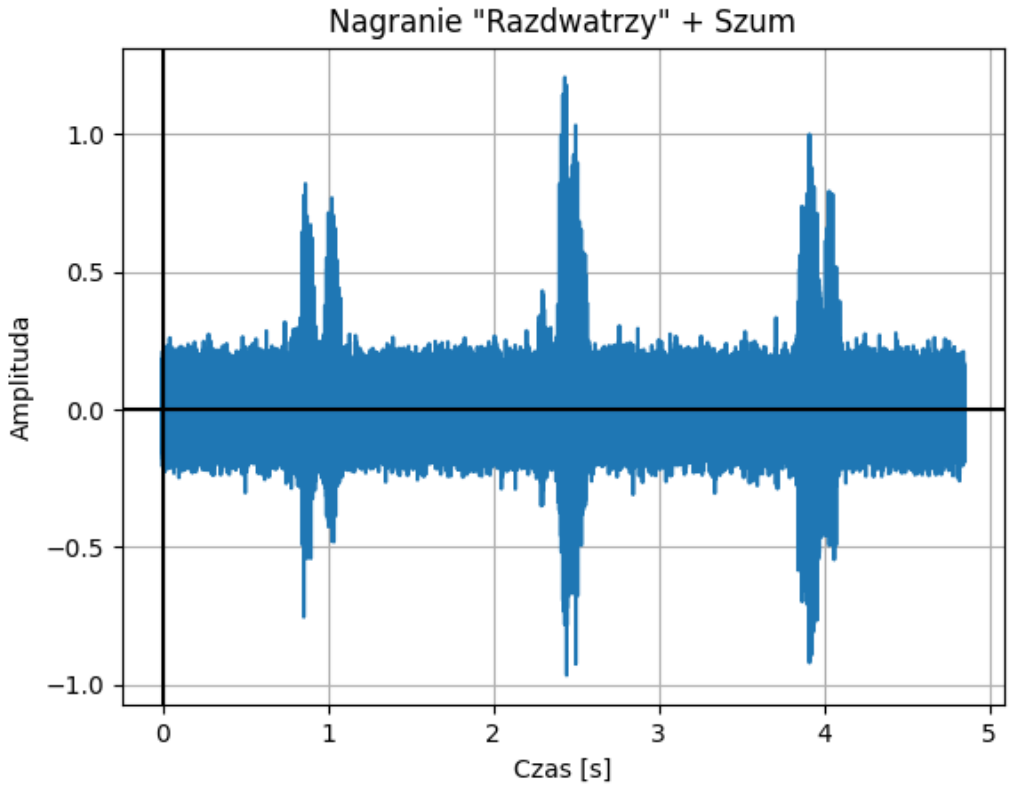
Wyniki:

- Wykres samego szumu:



Wykres 17 Sygnał szumu.

- Wykres nagrania z szumem:



Wykres 18 Sygnał szumu i nagranej mowy.

Zadanie 5(dodatkowe)

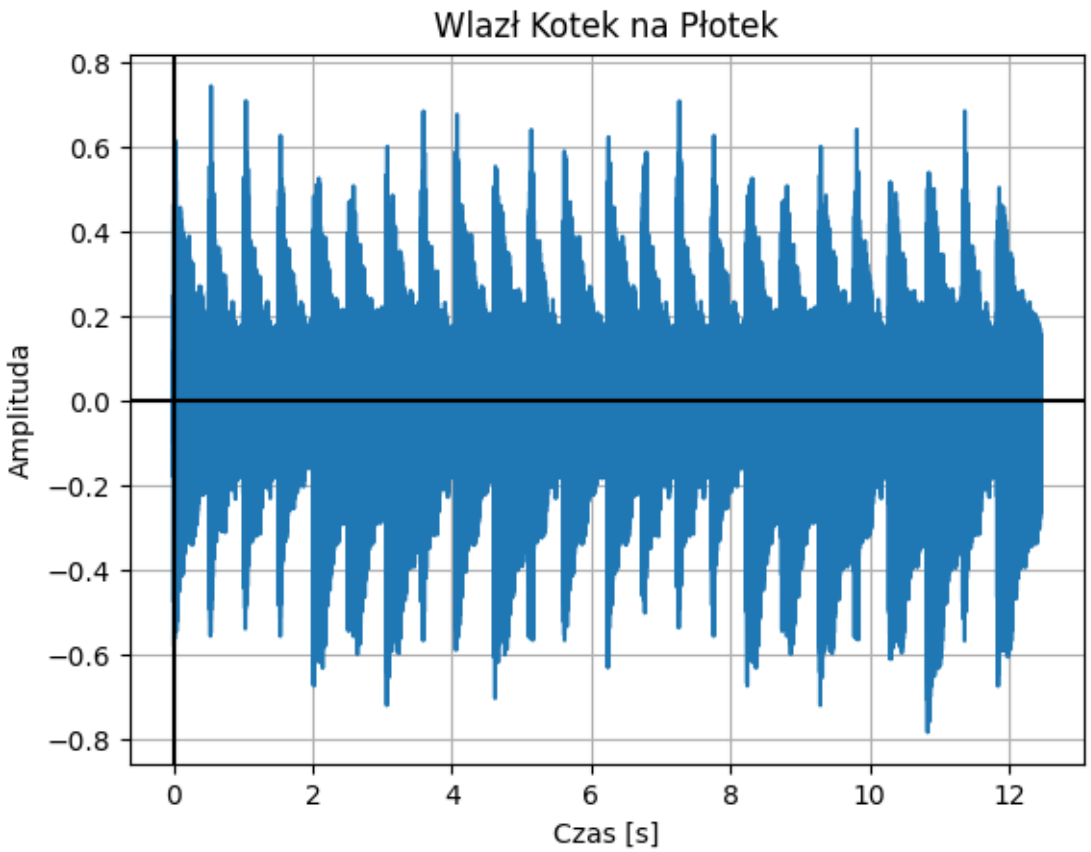
Realizacja w kodzie

```python
from music21 import note, stream # Z biblioteki music21 importujemy note oraz stream

#Tworzenie nut do "wlazł kotek na płotek"
n1 = note.Note('G', quarterLength=1)
n2 = note.Note('E', quarterLength=1)
n3 = note.Note('E', quarterLength=1)
n4 = note.Note('F', quarterLength=1)
n5 = note.Note('D', quarterLength=1)
n6 = note.Note('D', quarterLength=1.1)

n7 = note.Note('C', quarterLength=1)
n8 = note.Note('E', quarterLength=1)
n9 = note.Note('G', quarterLength=1.1)

n10 = note.Note('C', quarterLength=1)
n11 = note.Note('E', quarterLength=1)
n12 = note.Note('G', quarterLength=1.25)

n13 = note.Note('G', quarterLength=1)
n14 = note.Note('E', quarterLength=1)
n15 = note.Note('E', quarterLength=1)
n16 = note.Note('F', quarterLength=1)
n17 = note.Note('D', quarterLength=1)
n18 = note.Note('D', quarterLength=1.1)

n19 = note.Note('C', quarterLength=1)
n20 = note.Note('E', quarterLength=1)
n21 = note.Note('C', quarterLength=1.1)

n22 = note.Note('C', quarterLength=1)
n23 = note.Note('E', quarterLength=1)
n24 = note.Note('C', quarterLength=1.25)
```

```
s = stream.Stream() # Rozpoczęcie sklejania nut
s.append([n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20,n21,n22,n23,n24])
# Ciąg nut razem sklejony

s.write('midi',fp='kotek') # Zapis do pliku midi

from midi2audio import FluidSynth # Z biblioteki midi2audio improtuje FluidSynth

fs = FluidSynth() # Rozpoczęcie konwertera
fs.midi_to_audio('kotek.midi','my_melody.wav') # Konwertowanie z MIDI do WAV

import numpy as np  # Biblioteka numpy jako np
import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
import soundfile as sf # Biblioteka soundfile jak sf
import wave as w # Biblioteka wave jak w

sound = w.open("kotek.wav") # Otwarcie pliku audio w zmiennej
frames_frequency = sound.getframerate()*2 # Wyliczanie częstotliwości klatek
record = np.frombuffer(sound.readframes(-1),dtype="int16")/2000 # Wczytanie dźwięku do zmiennej z zmiennej z plikiem audio
x = np.arange(0, record.size/frames_frequency, 1/frames_frequency) # Stworzenie zakresu o długości <0;record.size/frames_frequency>

plt.plot(x,record) # Rysowanie w zakresie zmiennej x funkcji record
plt.axhline(y=0,color = "k") # Zaznacznie lini y=0 kolorem czarnym
plt.axvline(x=0,color = "k") # Zaznacznie lini x=0 kolorem czarnym
plt.grid(True,which='both') # Włączenie siatki
plt.title('Wlazł Kotek na Płotek') # Tytuł wykresu
plt.xlabel("Czas [s]") # Podpis osi x
plt.ylabel("Amplituda") # Podpis osi y
```

Rysunek 12 Kod do zadania 5.

Wynik

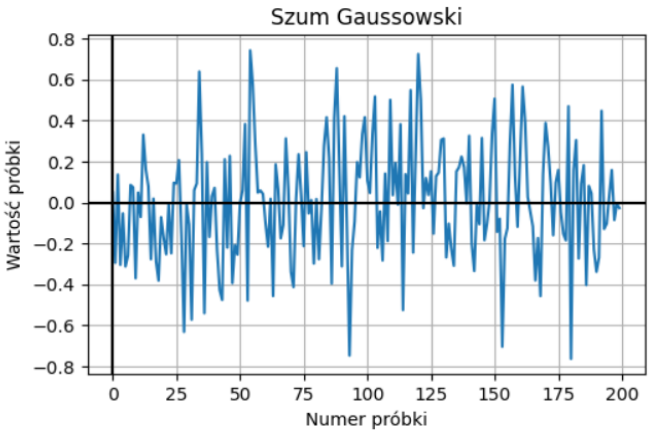


Wykres 19 Sygnał wygenerowany w zadaniu dodatkowym.

Podsumowanie i wnioski

Powyższe zadania pozwoliły bliżej przyjrzeć się pojęciom sygnału dyskretnego, próbkowania i wpływu różnych parametrów na dźwięk. W zadaniu pierwszym zauważono, że generowanie sygnału dyskretnego jest niemożliwe bez próbkowania. Wartości kolejnych próbek można zapisać w tablicy, co potwierdza teorię, iż sygnał dyskretny nie jest funkcją, lecz ciągiem. Dodatkowo zapoznano się z szumem Gaussowskim, który jest sygnałem nieokresowym. Jego częstotliwości i amplituda losowo zmieniają się w czasie. W zadaniu 2.1c zauważono, że nie trzeba definiować dziedziny funkcji generującej szum Gaussowski w zmiennej „x". Można ją zdefiniować w trzecim parametrze funkcji *random.normal()* z pakietu NumPy, o ile nasz przedział zaczyna się od zera.

```
1   import numpy as nu;
2   import matplotlib.pyplot as plt;
3
4   u, sigma = 0, nu.sqrt(0.1)          # Deklaracja zmiennych i ich wartości
5   s = nu.random.normal(u, sigma, 200) # Wzór funkcji
6   plt.plot(s)
7   plt.axhline(y=0,color = "k")        # Zaznacznie lini y=0 kolorem czarnym
8   plt.axvline(x=0,color = "k")        # Zaznacznie lini x=0 kolorem czarnym
9   plt.grid(True,which='both')         # Włączenie siatki
10  plt.title('Szum Gaussowski')        # Tytuł wykresu
11  plt.xlabel("Numer próbki")          # Podpis osi x
12  plt.ylabel("Wartość próbki")        # Podpis osi y
13  plt.show()
14
15
```



Rysunek 13 Realizacja zadanie 2.1c bez definiowania dziedziny za pomocą zmiennej x.

W zadaniu drugim badano wpływ amplitudy, częstotliwości i przesunięcia fazowego na wrażenia słuchowe. Po wykonaniu kilku testów jesteśmy w stanie stwierdzić, że:

-Amplituda wpływa na natężenie dźwięku. Oznacza to, że wraz ze wzrostem amplitudy rośnie natężenie (głośność), której jednostką są decybele.

-Wraz ze wzrostem częstotliwości, rośnie wysokość dźwięku.

-Zmiana przesunięcia fazowego nie wpływa na dźwięk, natomiast przesuwa go w czasie.

W zadaniu trzecim należało zasymulować próbkowanie sygnału w różnych częstotliwościach próbkowania. Od razu zauważono jak istotny jest wybór wartości częstotliwości próbkowania. W przypadkach, gdy jest ona mniejsza lub równa dwukrotnej częstotliwości zadanego sygnału, z pozyskanych próbek nie będziemy w stanie odwzorować pierwotnego sygnału. Zadanie potwierdziło słuszność twierdzenia Nyquista-Kotielnikova. Dzięki zadaniu czwartemu zapoznano się z Python'owymi bibliotekami Wave oraz Soundfile. Okazało się, że w łatwy sposób można odwrócić dźwięk lub modyfikować go na przykład dodając do niego szum.

Zadania były inspiracją do przeprowadzenia dodatkowego testu. Jak wiadomo częstotliwości fal, słyszalnych dla człowieka, zawierają się od ok 16Hz do ok 20kHz. Napisano program generujący sygnał o amplitudzie równej 1, częstotliwości próbkowania równej 44,1kHz i stopniowo zwiększano częstotliwość sygnału. W naszym przypadku granica słyszanego dźwięku to około 18,5kHz.

```
1   import numpy as np  # Biblioteka numpy jako np
2   import matplotlib.pyplot as plt # Biblioteka matplotlib od python plot jako plt
3   import soundfile as sf # Biblioteka soundfile jak sf
4
5   A = 1 # Amplituda
6   F = 18500 # Częstotliwość
7   Fs = 44100 # Częstotliwość próbkowania
8   Ts = 1/Fs # Częstotliwość próbkowania w sekundach
9   O = 2*np.pi*F # Omega
10  Phi = 0 # Przesunięcie fazowe
11  x = np.arange(0,10,Ts) # Określenie dziedziny <0,10>, Ts pomiarów w tym zakresie
12  y = A * np.sin(O*x + Phi) # Wzór funkcji y
13
14  plt.plot(x,y) # Rysowanie w zakresie zmiennej x funkcji y
15  sf.write('wynik18_5khz.wav',y,Fs) # Eksportowanie do pliku wav funkcji y o próbkowaniu Fs
```

Rysunek 14 Kod do badania granicy słyszalnego dźwięku.

Źródła lub bibliografia lub podobnie

[1]. http://home.agh.edu.pl/~rad/wiki/images/5_Sygnaly.pdf
[2]. http://iele.polsl.pl/iele/dydaktyka/zasoby/materialy/ptc-uk/lekcja1.html
[3]. https://pl.wikipedia.org
[4]. https://r.pawliczek.po.opole.pl/dydaktyka/mtr_06.pdf