WMM 25L - Laboratorium 1 - Sprawozdanie

Analiza częstotliwościowa sygnałów czasu dyskretnego

Bartosz Żelazko i Amadeusz Lewandowski

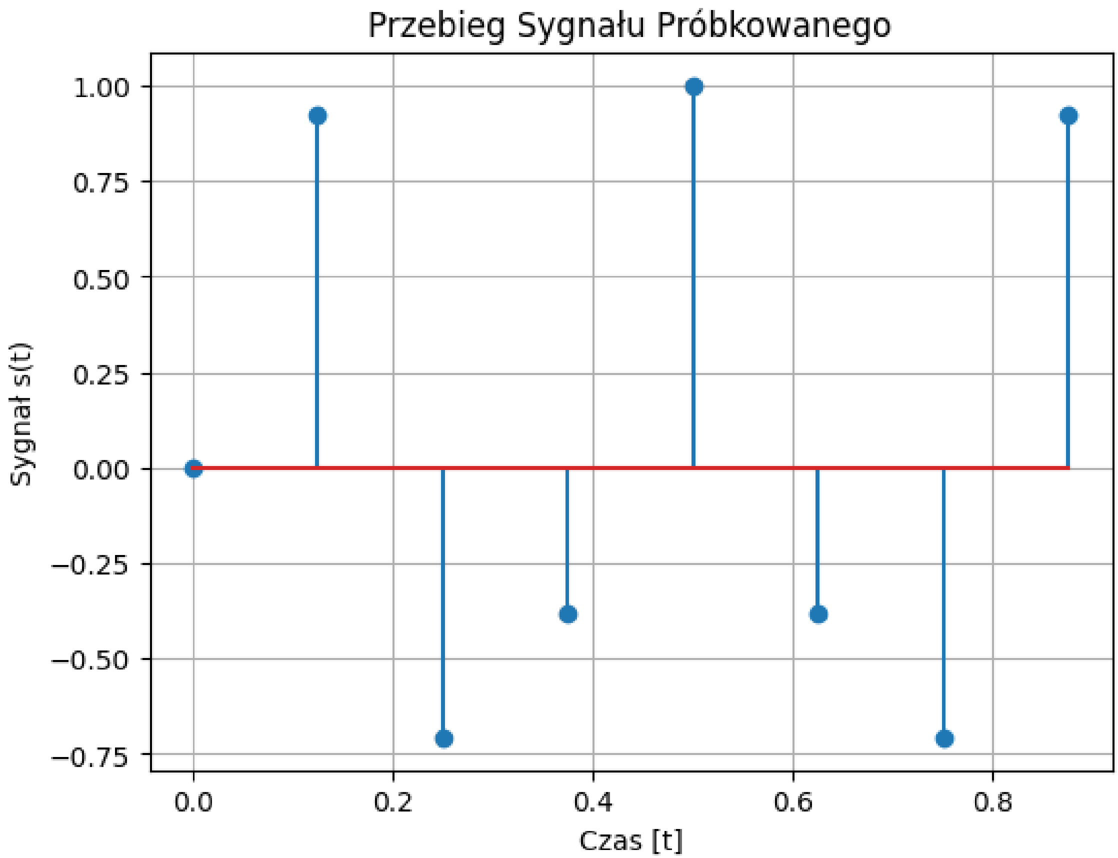
Zadanie 1

Liczba próbek (w jednym okresie) sygnału rzeczywistego $s(t) = sin(5\pi t)$ wynosi N, gdzie N jest potęgą 2.

a. Przyjmując N = 8 wykreślić przebieg sygnału spróbkowanego, widmo amplitudowe i fazowe oraz zweryfikować eksperymentalnie słuszność twierdzenia Parsevala.

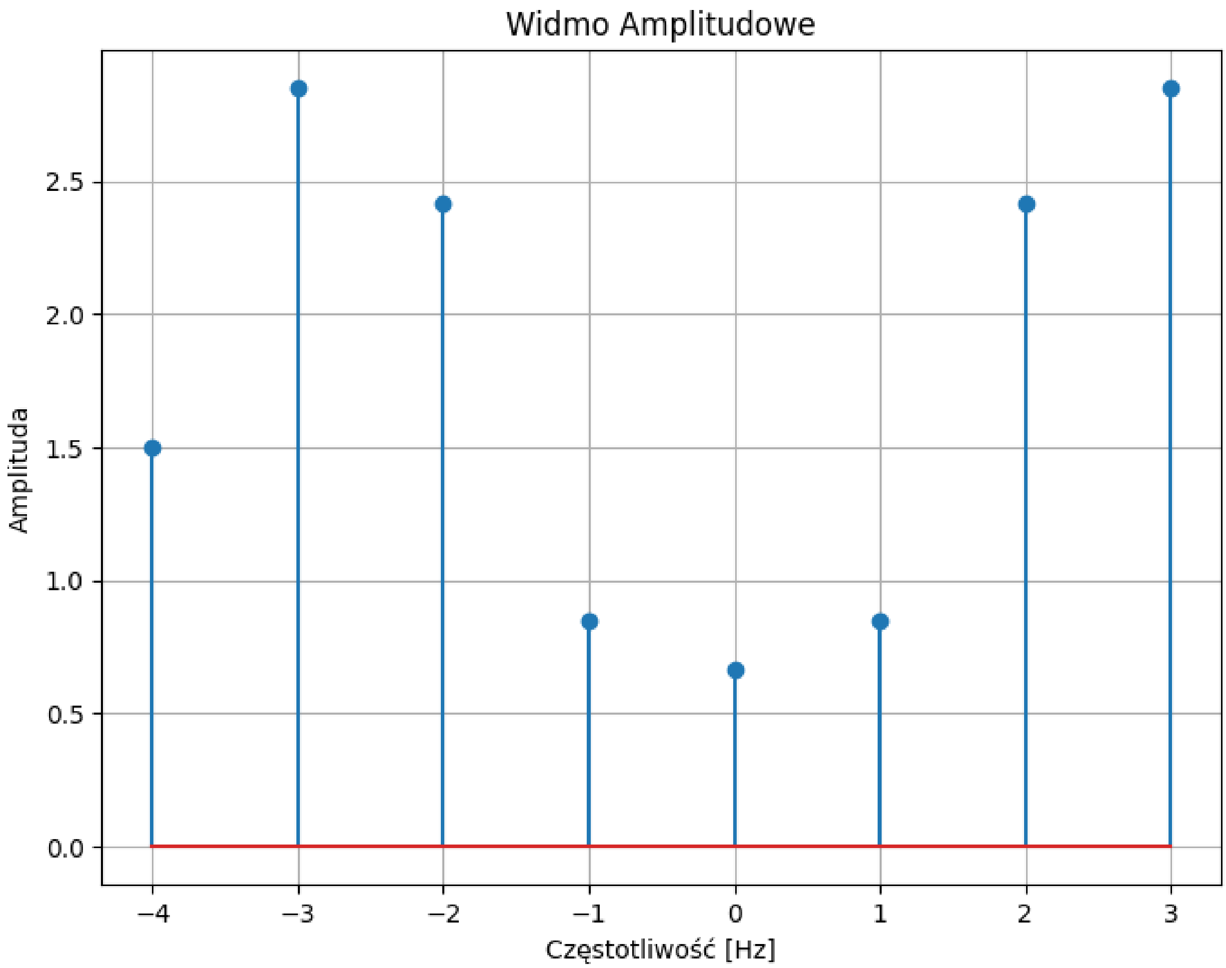
b. Wykreślić wykres przedstawiający czas wyznaczania widma sygnału dyskretnego za pomocą algorytmu FFT w funkcji liczby próbek $N = 2$^l, $l \in \mathbb{N}$. Skomentować kształt otrzymanego wykresu odnosząc się do teoretycznej złożoności obliczeniowej algorytmu FFT.

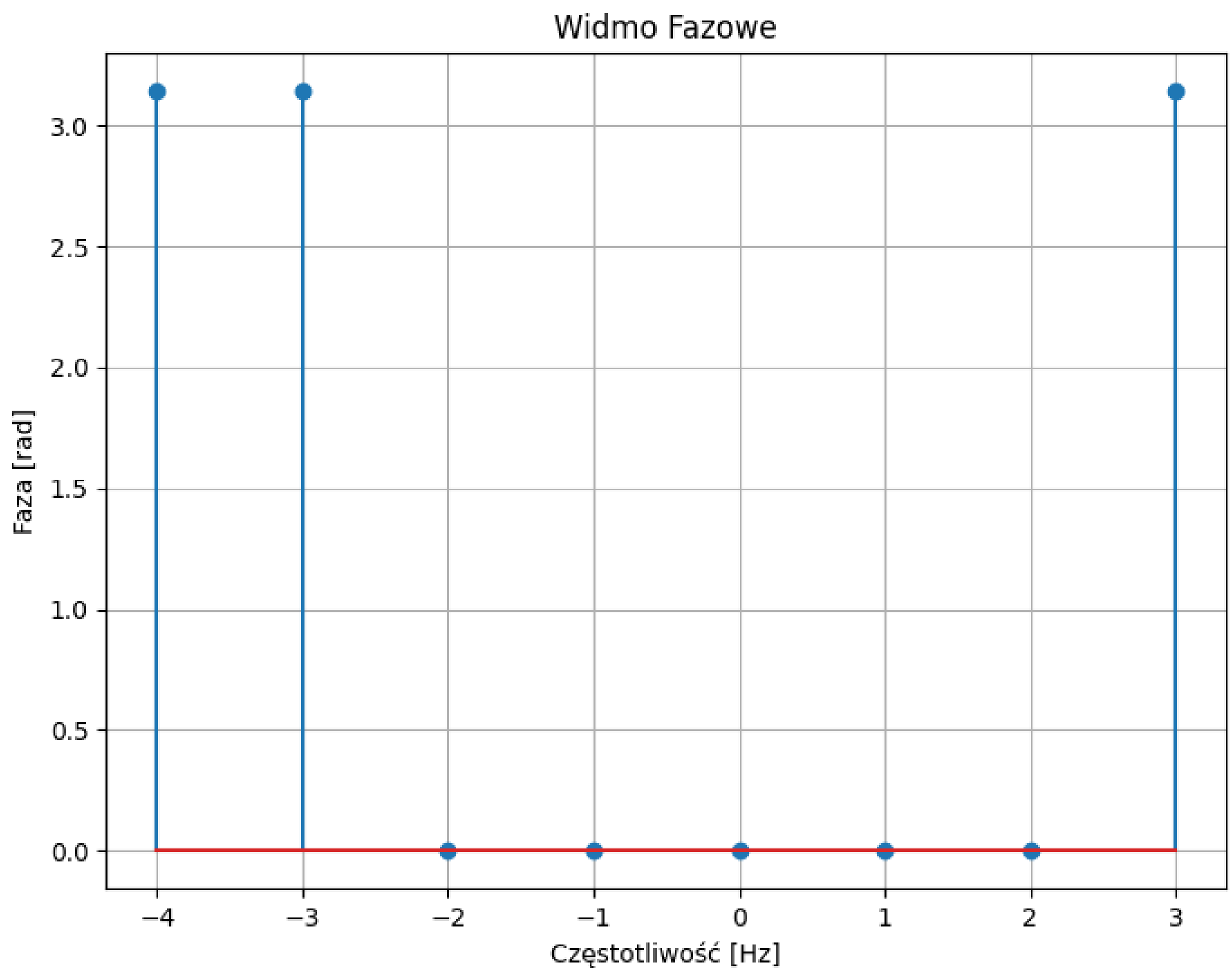
```
import numpy as np
import matplotlib.pyplot as plt

num_samples = 8 # Próbki
t = np.arange(num_samples) / num_samples # Równomierne rozłożenie
próbek w czasie
x = np.sin(5 * np.pi * t) # Sygnał

plt.stem(t, x)
plt.title("Przebieg Sygnału Próbkowanego")
plt.xlabel("Czas [s]")
plt.ylabel("Sygnał s(t)")
plt.grid(True)
plt.show()
```




```
X = np.fft.fft(x) # Dyskretna transformata Fouriera sygnału s(t)
epsilon = 1e-10

# Usuwawnie niepotrzebnych części urojonych
X.imag[np.abs(X.imag) < epsilon] = 0

amp = np.abs(X) # Widmo Amplitudowe
phase = np.angle(X) # Widmo Fazowe

freq = np.fft.fftfreq(num_samples, 1 / num_samples) # Częstotliwości

plt.figure(figsize=(8, 6))
plt.stem(freq, amp)
plt.title("Widmo Amplitudowe")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Amplituda")
plt.grid(True)
plt.show()

plt.figure(figsize=(8, 6))
plt.stem(freq, phase)
plt.title("Widmo Fazowe")
plt.xlabel("Częstotliwość [Hz]")
```

```
plt.ylabel("Faza [rad]")
plt.grid(True)
plt.show()
```





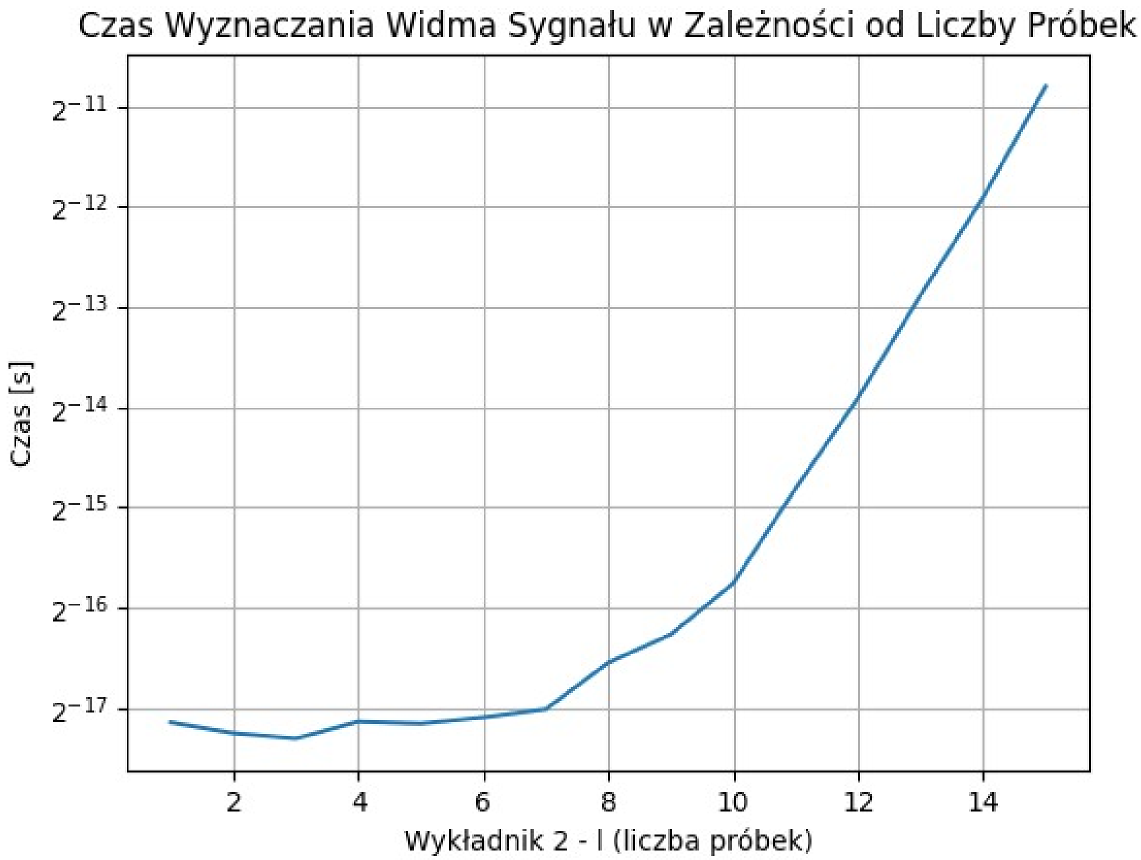

```
signal = round(np.sum(np.abs(x) ** 2), 5)
spectrum = round(np.sum(np.abs(X) ** 2) / num_samples, 5)

print("Weryfikacja Twierdzenia Parsevala:")
print(f"Sygnał: {signal}")
print(f"Widmo: {spectrum}")
print(f"Sygnał == Widmo: {signal == spectrum}")
```

```
Weryfikacja Twierdzenia Parsevala:
Sygnał: 4.0
Widmo: 4.0
Sygnał == Widmo: True
```

Twierdzenie Persevala zostało spełnione, co widać w powyższych wynikach.

```
import time
import gc

def measure_fft_time(l: int, iterations = 5000) -> float:
    N = 2 ** l
```

```python
    t = np.arange(N) / N
    x = np.sin(5 * np.pi * t)

    gc_old = gc.isenabled()
    gc.disable()

    start = time.process_time()
    for _ in range(iterations):
        np.fft.fft(x)
    end = time.process_time()

    if gc_old:
        gc.enable()

    return (end - start) / iterations

# Pomiar
measurements = [measure_fft_time(l) for l in range(1, 16)]

plt.plot(range(1, 16), measurements)
plt.title("Czas Wyznaczania Widma Sygnału w Zależności od Liczby 
Próbek")
plt.xlabel("Wykładnik 2 - l (liczba próbek)")
plt.ylabel("Czas [s]")
plt.yscale("log", base=2)
plt.grid(True)
plt.show()
```



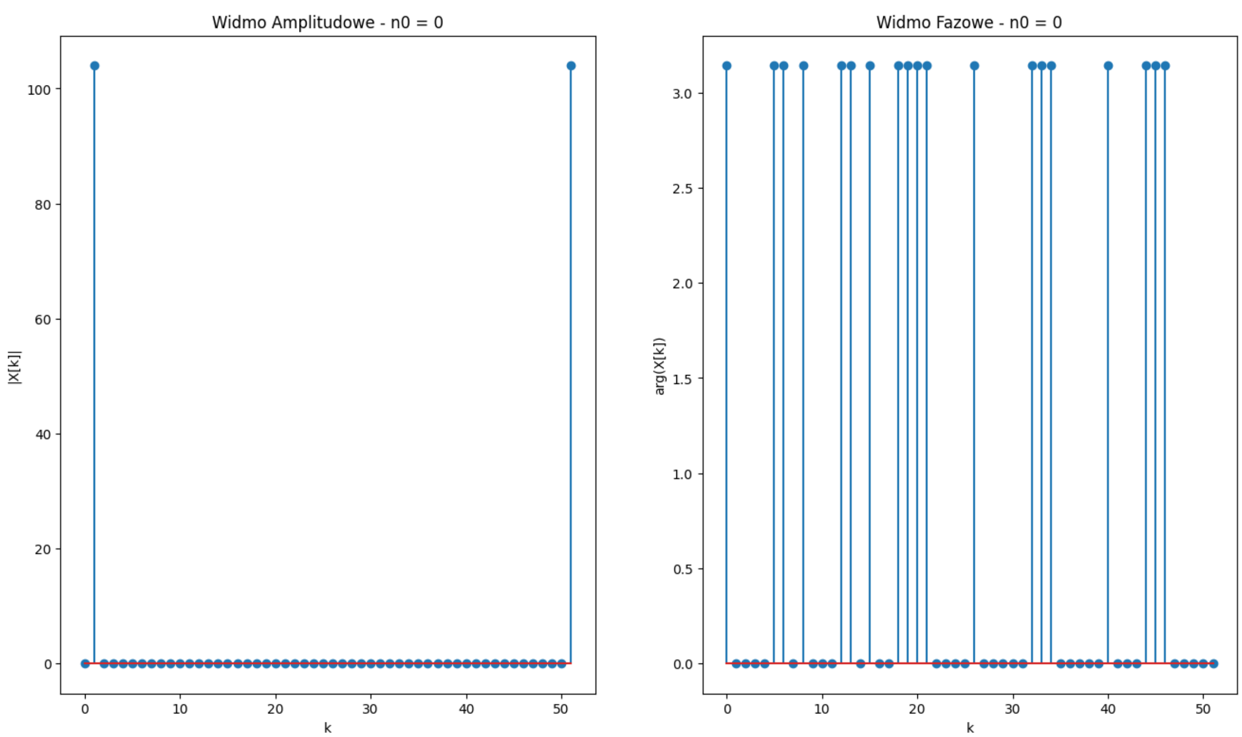
Teoretyczna złożoność obliczeniowa algorytmu FFT to O(nlog(n)), a kształt otrzymanego wyżej wykresu przypomina kształt tej złożoności obliczeniowej.

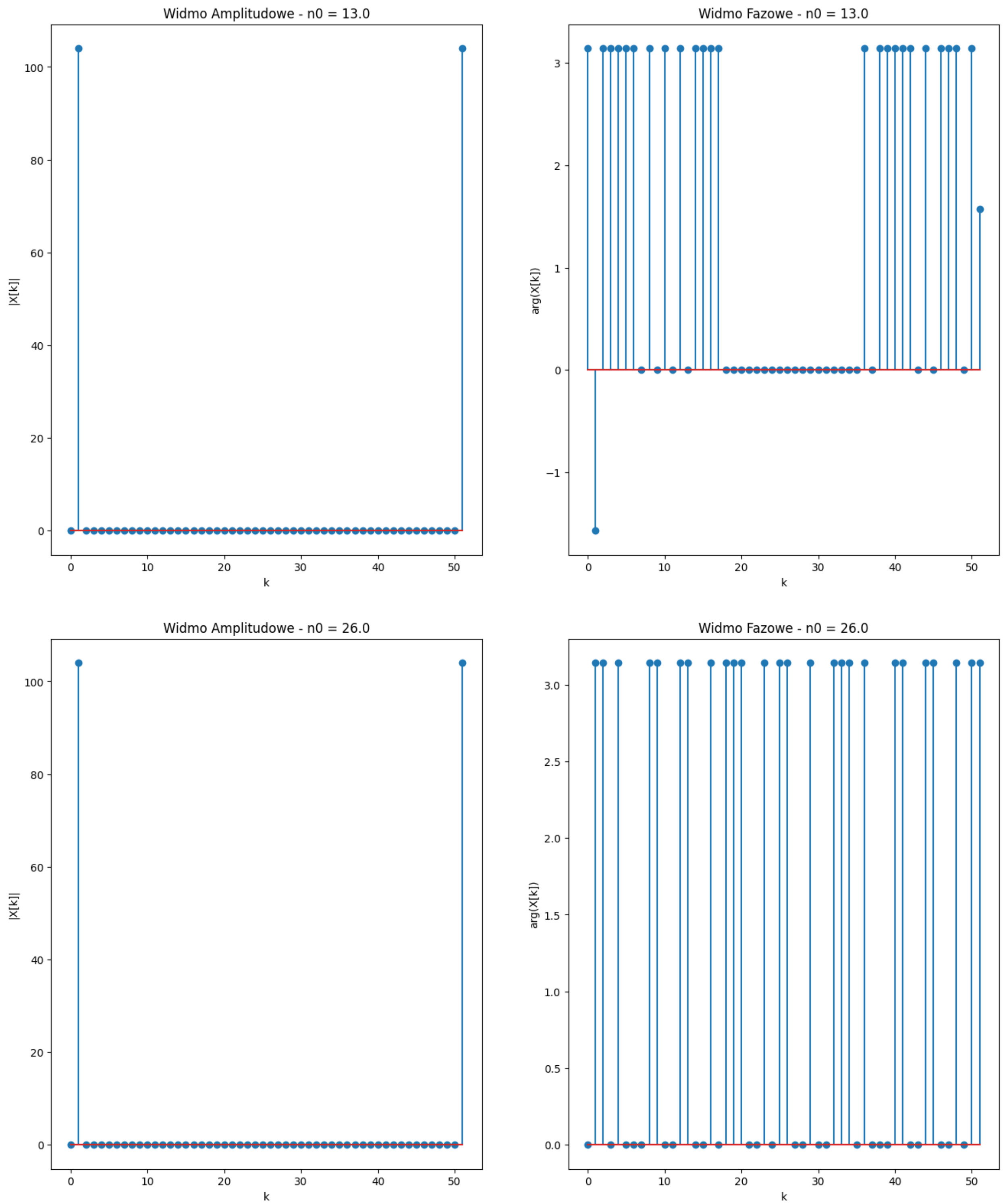
Zadanie 2

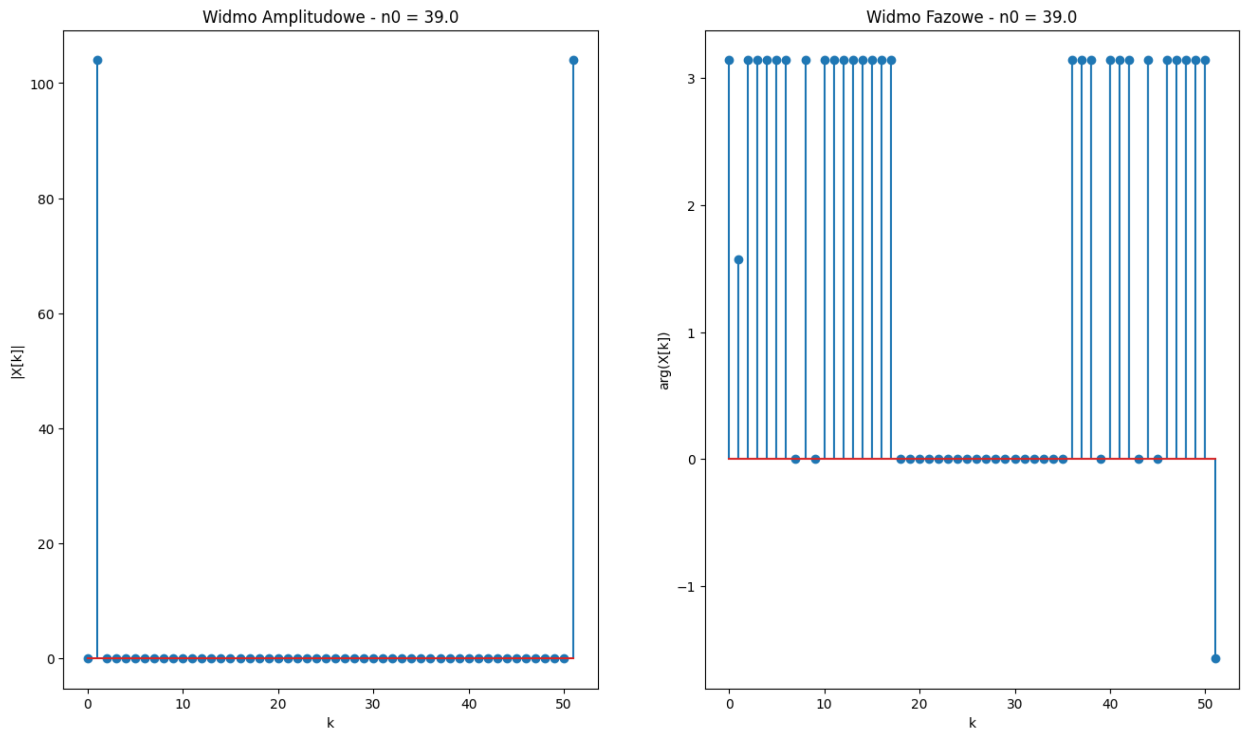
Zbadać wpływ przesunięcia w czasie na postać widma amplitudowego i widma fazowego dyskretnego sygnału harmonicznego $s[n] = A\,cos\,(2\pi*(n/N))$ o amplitudzie $A = 4$ i okresie podstawowym $N = 52$. W tym celu dla każdej wartości $n0 \in \{0, N/4, N/2, 3N/4\}$ wykreślić widmo amplitudowe i fazowe przesuniętego sygnału $s[n - n0]$. Skomentować otrzymane wyniki.

```
def signal(n: np.array, n0: int, A=4, N=52) -> float:
    return A * np.cos(2 * np.pi* (n - n0) / N)

# Parametry z treści zadania
A = 4
N = 52
n = np.arange(0, N)
n0_values = [0, N/4, N/2, 3*N/4]
epsilon = 1e-10

for n0 in n0_values:
    x = signal(n, n0)
```

```python
X = np.fft.fft(x) # Dyskretna transformata Fouriera sygnału s(N)

# Usuwawnie niepotrzebnych części urojonych
X.imag[np.abs(X.imag) < epsilon] = 0

amp = np.abs(X) # Widmo Amplitudowe
phase = np.angle(X) # Widmo Fazowe

plt.figure(figsize=(16, 9))

plt.subplot(1, 2, 1)
plt.stem(n, amp)
plt.title(f"Widmo Amplitudowe - n0 = {n0}")
plt.xlabel("k")
plt.ylabel("|X[k]|")

plt.subplot(1, 2, 2)
plt.stem(n, phase)
plt.title(f"Widmo Fazowe - n0 = {n0}")
plt.xlabel("k")
plt.ylabel("arg(X[k])")

plt.show()
```




Widmo Amplitudowe - n0 = 13.0
|X[k]|
0
20
40
60
80
100
0
10
20
30
40
50
k
Widmo Fazowe - n0 = 13.0
arg(X[k])
−1
0
1
2
3
0
10
20
30
40
50
k
Widmo Amplitudowe - n0 = 26.0
|X[k]|
0
20
40
60
80
100
0
10
20
30
40
50
k
Widmo Fazowe - n0 = 26.0
arg(X[k])
0.0
0.5
1.0
1.5
2.0
2.5
3.0
0
10
20
30
40
50
k



Widmo amplitudowe pozostaje niezmienne bez względu na przesunięcie, natomiast fazowe ulega zmianie, co widać na powyższych wykresach.

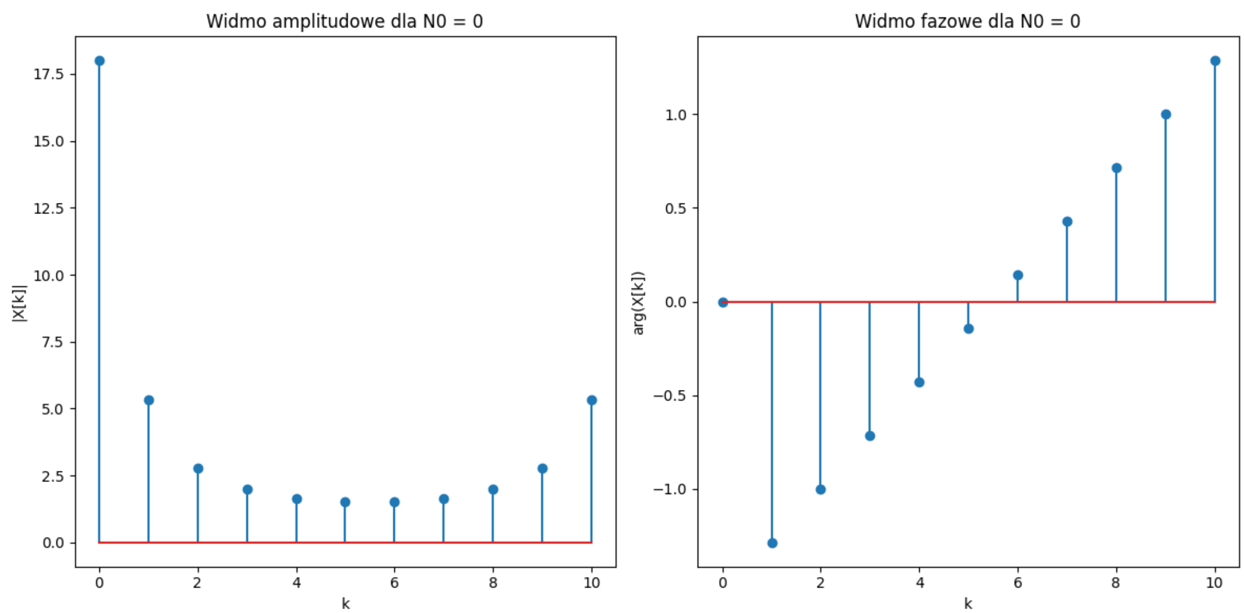
Jest to zgodne z właściwościami transformaty Fouriera, które dotyczą przesunięcia czasowego.

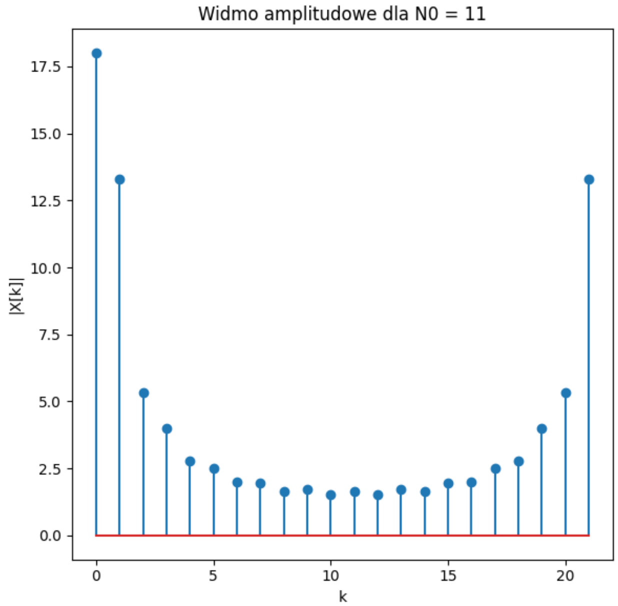
Zadanie 3

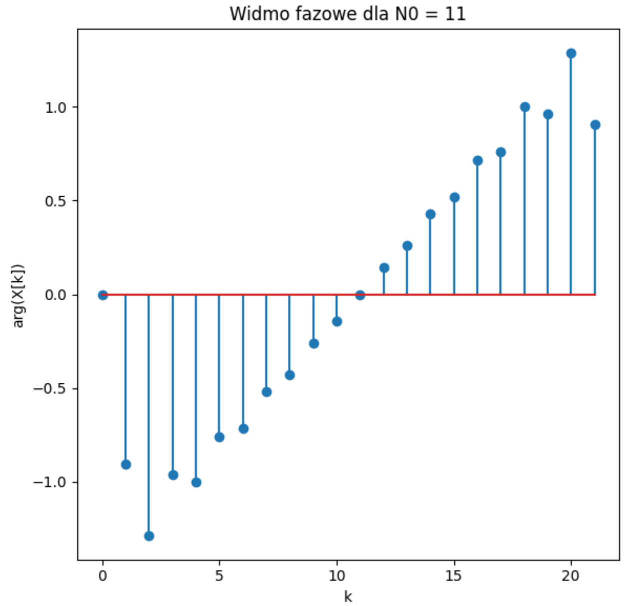
Zbadać wpływ dopełnienia zerami na postać widma amplitudowego i widma fazowego dyskretnego sygnału $s[n] = A\,(1 - (n \bmod N)/\,N\,)$ o amplitudzie $A = 3$ i okresie podstawowym $N = 11$. W tym celu dla każdej wartości $N0 \in \{0, 1N, 4N, 9N\}$ wykreślić widmo amplitudowe i fazowe sygnału $s[n]$ dopełnionego $N0$ zerami. Skomentować otrzymane wyniki.

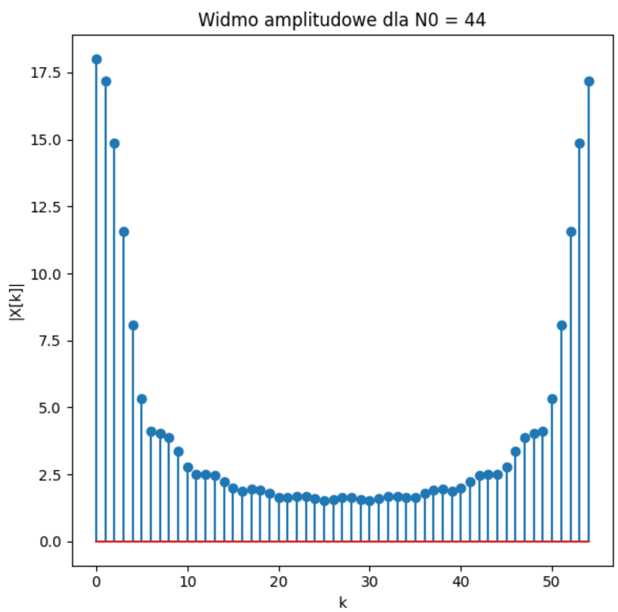
```
import numpy as np
import matplotlib.pyplot as plt
#funkcja calculate odopenia sygna zerami
def calculate(n, zero_count, A, N):
    signal = A * (1 - ((n % N) / N))
    signal_zeros = np.concatenate((signal, np.zeros(zero_count)))
    return signal_zeros

def main():
    #ustalenie wartosi zmiennych zgodnie z parametrami zadania
    A = 3
    N = 11
    N0 = [0, N, 4*N, 9*N]
    n = np.arange(0, N)
    for zero_count in N0:
```

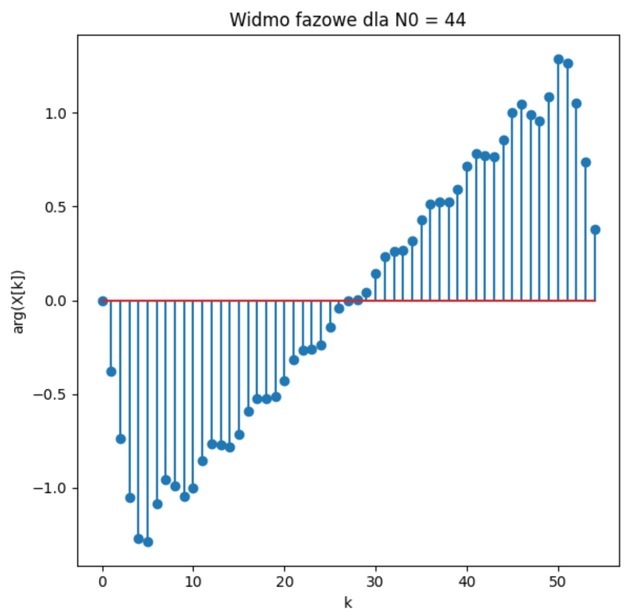
```python
        x = calculate(n, zero_count, A, N)
        X = np.fft.fft(x)
        #filtorwanie malych liczb
        for xi in X:
            if (xi.imag < 1e-10 or xi.real < 1e-10):
                xi = 0

        plt.figure(figsize=(12, 6))

        #wykres widma amplitudowego
        plt.subplot(1, 2, 1)
        plt.stem(np.arange(0, N+zero_count), np.abs(X))
        plt.title(f"Widmo amplitudowe dla N0 = {zero_count}")
        plt.xlabel("k")
        plt.ylabel("|X[k]|")

        #wykres widma fazowego
        plt.subplot(1, 2, 2)
        plt.stem(np.arange(0, N+zero_count), np.angle(X))
        plt.title(f"Widmo fazowe dla N0 = {zero_count}")
        plt.xlabel("k")
        plt.ylabel("arg(X[k])")

        plt.tight_layout()
        plt.show()

main()
```

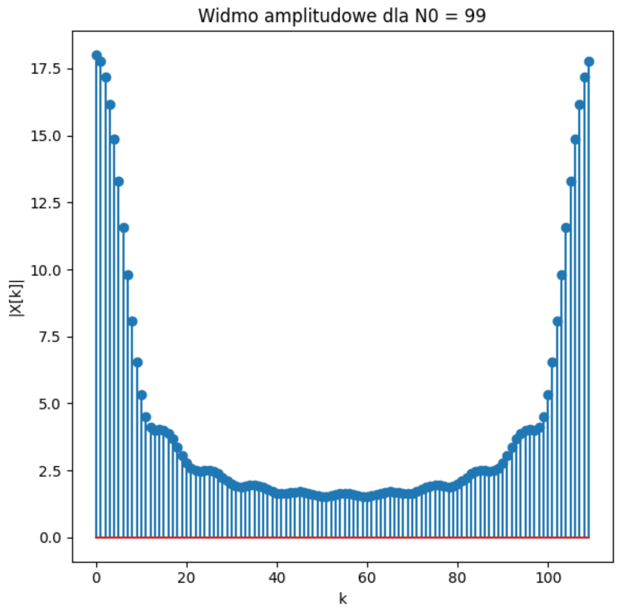


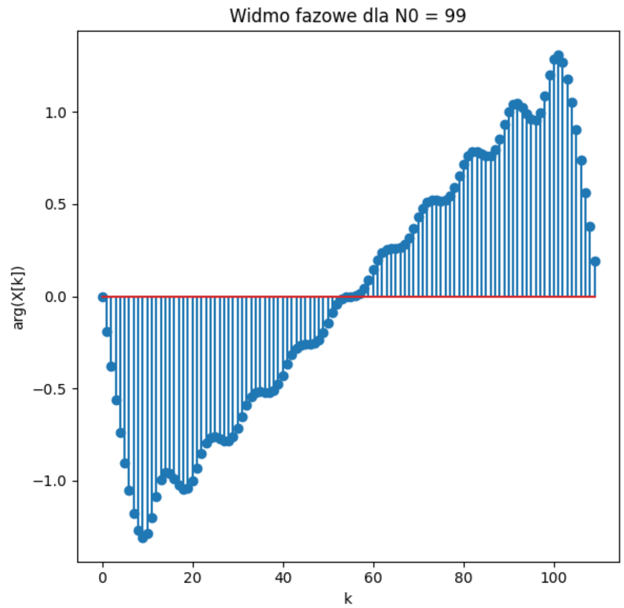

Widmo amplitudowe dla N0 = 11
|X[k]|
k


Widmo fazowe dla N0 = 11
arg(X[k])
k


Widmo amplitudowe dla N0 = 44
|X[k]|
k


Widmo fazowe dla N0 = 44
arg(X[k])
k





Wnioski

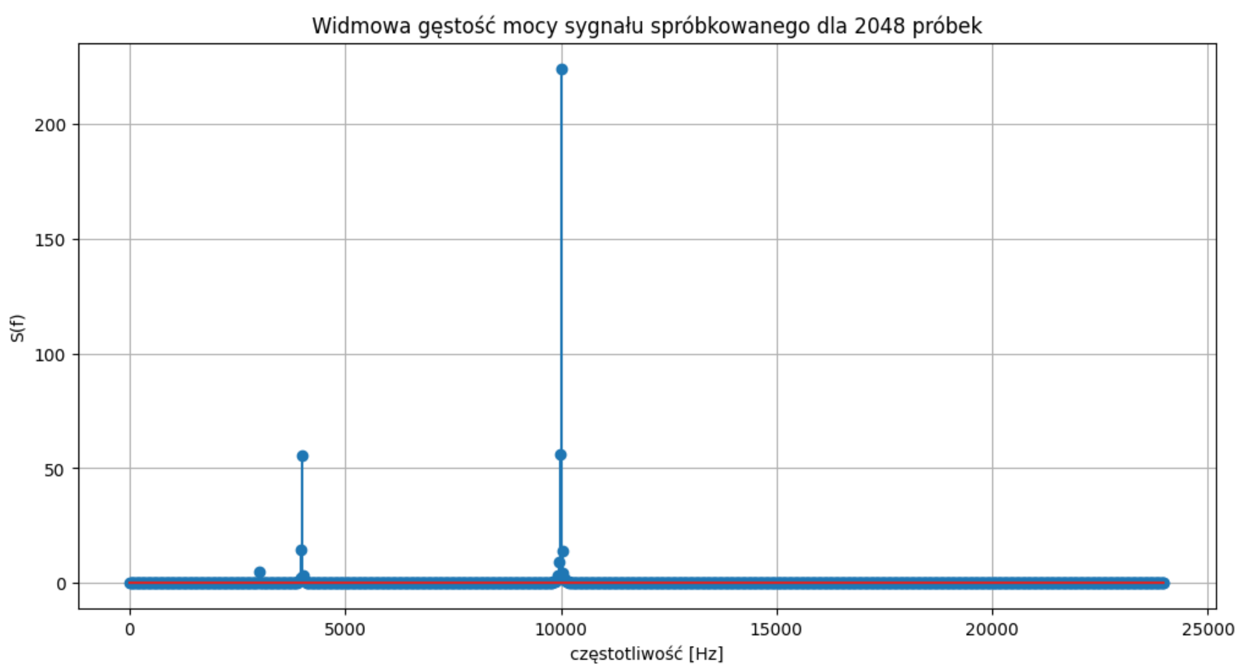
Jak możemy zauważyć na powyższych wykresach dopełnianie zerami nie zmienia kształtu wykresu widm. Natomiast w miarę wzrostu ilości zer rozdzielczość wykresów wzrasta gdyż pojawia się więcej punktów. Tak więc w ogólności można stwierdzić że dopełnienie zerami zwiększa dokładność otrzymanych widm.

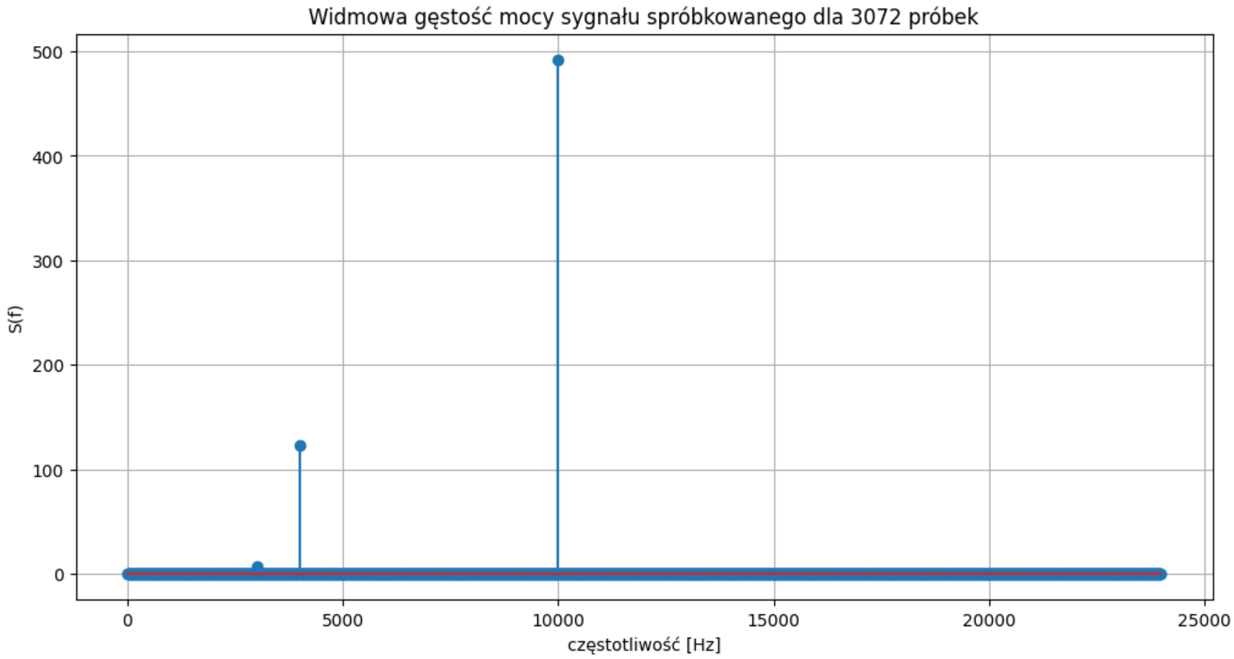
Zadanie 4

Dany jest sygnał rzeczywisty $s(t) =$ $ $A1\, sin(2\pi f1t) + A2\, sin(2\pi f2t) + A3\, sin(2\pi f3t)$ $, gdzie $A1 = 0.1$, $f1 = 3000$ Hz, $A2 = 0.4$, $f2 = 4000$ Hz, $A3 = 0.8$, $f3 = 10000$ Hz. Przy założeniu, że częstotliwość próbkowania wynosi $fs = 48000$ Hz, a liczba próbek sygnału wynosi $N1 = 2048$,przedstawić wykres widmowej gęstości mocy sygnału spróbkowanego. Czy dla podanej liczby próbek mamy do czynienia ze zjawiskiem przecieku widma? Czy sytuacja uległaby zmianie dla liczby próbek $N2 = 3/2N1$? Odpowiedź uzasadnić.

```
#deklaracja stałych zgodnie z treścią zadania
a1 = 0.1
a2 = 0.4
a3 = 0.8
f1 = 3000
f2 = 4000
f3 = 10000
fs = 48000
N1 = 2048
#funkcja signal generuje sygnał spróbkowany
def signal(N):
    time = np.arange(0, N)/fs
    signal = a1 * np.sin(2*np.pi*time*f1) + a2 *
np.sin(2*np.pi*time*f2) + a3 * np.sin(2*np.pi*time*f3)
    return(signal)
```

```python
#funkcja plot rysuje wykres widmowej gęstość mocy sygnału
spróbkowanego dla zadanego n
def plot(N):
    s = signal(N)
    X = np.fft.fft(s)
    #filtorwanie malych liczb
    for xi in X:
        if (xi.imag < 1e-10 or xi.real < 1e-10):
            xi = 0
    density = abs(X ** 2 / N) # widmowa gęstość mocy
    freq = np.fft.fftfreq(N, 1 / fs)
    plt.figure(figsize=(12, 6))
    plt.stem(freq[:N//2], density[:N//2])
    plt.title(f"Widmowa gęstość mocy sygnału spróbkowanego dla {N}
próbek")
    plt.xlabel("częstotliwość [Hz]")
    plt.ylabel("S(f)")
    plt.grid(True)

plot(N1)
plot(int(N1*3/2))
```





Wnioski

Jak możemy zauważyć na powyższych wykresach w przypadku gdy liczba próbek wynosi 2048 mamy do czynienia ze zjawiskiem przecieku widma. Na wykresie dla tej liczby próbek wyraźnie widać 3 wzrosty wartości odpowiadające częstotliwością $ f1, f2, f3 $, natomiast na wykresie widoczne są mniejsze wzrosty które są wynikiem " rozlania się " widma. Właśnie te wzrosty świadczą o występowaniu zjawiska przecieku. W przypadku zwiększenia liczby próbek do 3072 wyraźnie widać, że poboczne piki zniknęły i na wykresie obserwujemy 3 piki. Tak więc przy zwiększeniu liczby próbek zjawisko przecieku znika. Dla mniejszej liczby próbek występuje zjawisko przecieku gdyż częstotliwości składowych sygnału nie są całkowitą wielokrotnością rozdzielczości częstotliwościowej DFT ($\Delta f = fs/N$). Zwiększenie liczby próbek poprawia rozdzielczość częstotliwościową a co za tym idzie eliminuje efekt przecieku widma.