Wstęp do multimediów (WMM)

Laboratorium #1: Analiza częstotliwościowa sygnałów czasu dyskretnego

Grupa 103, 13 marca 2025 r., godz. 14.15
Kacper Górski
Jakub Bagiński

Zadanie 1

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

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

```
from matplotlib import pyplot as plt
from scipy.fft import fft, fftfreq
import numpy as np
import time
```

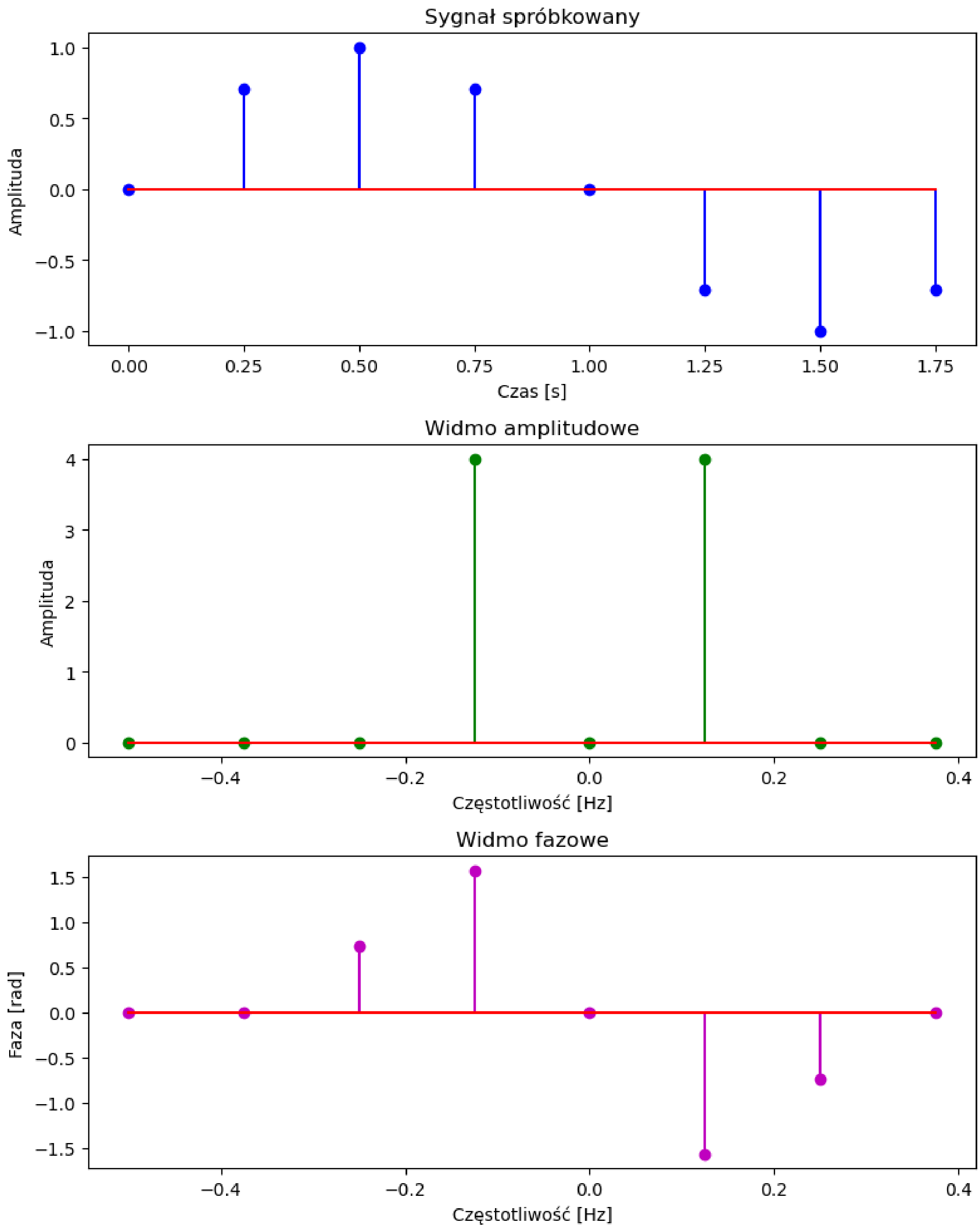
Przyjmując N = 8 wykreślić przebieg sygnału spróbkowanego, widmo amplitudowe i fazowe oraz zweryfikować eksperymentalnie słuszność twierdzenia Parsevala.

```
N = 8                                         # liczba próbek
t = np.linspace(0.0, 2, N, endpoint=False)  # czas próbkowania, liczba 
próbek N
signal = np.sin(np.pi * t)                    # sygnał s(t) = sin(πt)

def calc_spectrum(signal):
    return fft(signal)

def calc_amplitude_spectrum(signal):
    return np.abs(calc_spectrum(signal))

def calc_phase_spectrum(signal):
    angles = np.angle(calc_spectrum(signal))

    zero_angles = abs(angles) < 0.01
    angles *= ~zero_angles

    return angles
```

```python
def calc_parseval_left(signal):
    return np.sum(np.abs(signal) ** 2)

def calc_parseval_right(spectrum):
    return np.sum(np.abs(spectrum) ** 2) / N

def calc_freq_power(signal):
    spectrum = calc_spectrum(signal)
    return (1/N**2) * np.sum(np.abs(spectrum) ** 2)

def prove_parseval(signal):
    spectrum = calc_spectrum(signal)
    print("Parseval")
    print("left: ", calc_parseval_left(signal))
    print("right: ", calc_parseval_right(spectrum))
    return round(calc_parseval_left(signal), 4) ==
round(calc_parseval_right(spectrum), 4)

print('Parseval proven? ', prove_parseval(signal))

# plot
spectrum = calc_spectrum(signal)
amplitude_spectrum = calc_amplitude_spectrum(signal)
phase_spectrum = calc_phase_spectrum(signal)
freqs = fftfreq(N)

fig, axs = plt.subplots(3, 1, figsize=(8, 10))

axs[0].stem(t, signal, linefmt='b-', markerfmt='bo', basefmt="r-")
axs[0].set_title("Sygnał spróbkowany")
axs[0].set_xlabel("Czas [s]")
axs[0].set_ylabel("Amplituda")

axs[1].stem(freqs, amplitude_spectrum, linefmt='g-', markerfmt='go',
basefmt="r-")
axs[1].set_title("Widmo amplitudowe")
axs[1].set_xlabel("Częstotliwość [Hz]")
axs[1].set_ylabel("Amplituda")

axs[2].stem(freqs, phase_spectrum, linefmt='m-', markerfmt='mo',
basefmt="r-")
axs[2].set_title("Widmo fazowe")
axs[2].set_xlabel("Częstotliwość [Hz]")
axs[2].set_ylabel("Faza [rad]")

plt.tight_layout()
plt.show()
```

```
Parseval
left:  4.0
```

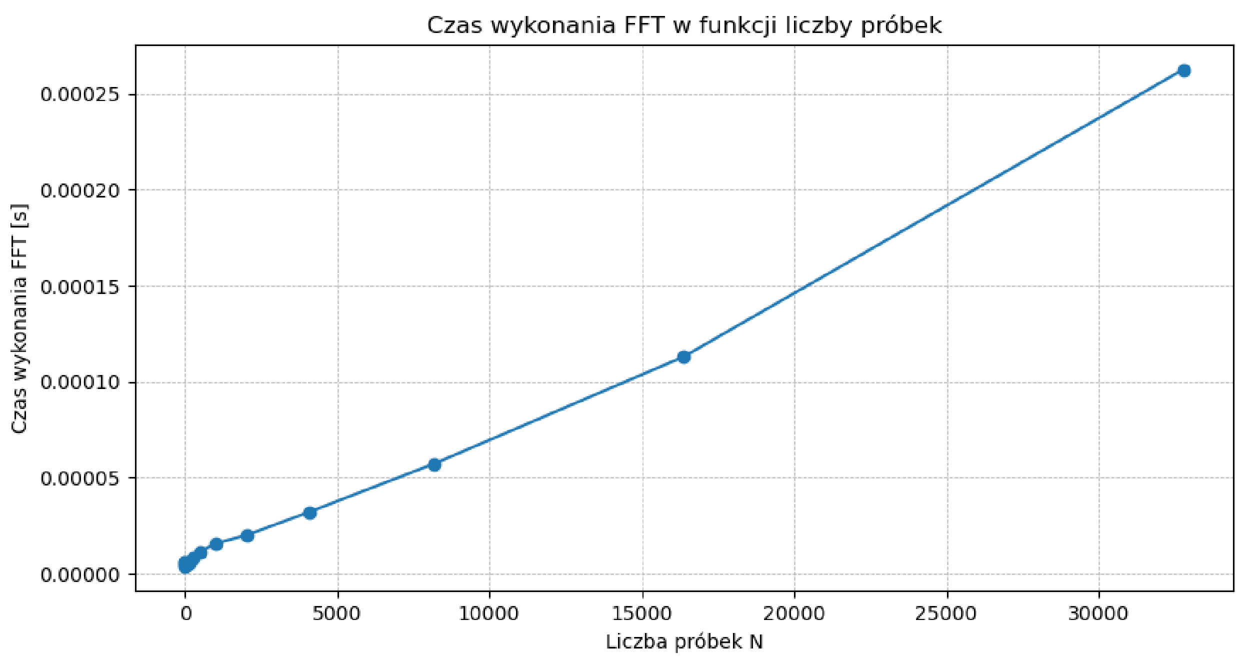
```
right:  4.0
Parseval proven?  True
```



Wniosek: Wyniki potwierdzają słuszność twierdzenia Parsevala.

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

```
iter = 1000
l = np.arange(1, 16)
N_l = 2 ** l
exec_times = []

for n in N_l:
    mean_time = 0
    for _ in range(iter):
        signal = np.random.rand(n)

        start = time.perf_counter()
        fft(signal)
        stop = time.perf_counter()

        mean_time += (stop - start)
    exec_times.append(mean_time / iter)

plt.figure(figsize=(10, 5))
plt.plot(N_l, exec_times, marker='o', linestyle='-')
plt.xlabel("Liczba próbek N")
plt.ylabel("Czas wykonania FFT [s]")
plt.title("Czas wykonania FFT w funkcji liczby próbek")
plt.grid(True, which="both", linestyle="--", linewidth=0.5)
plt.show()
```



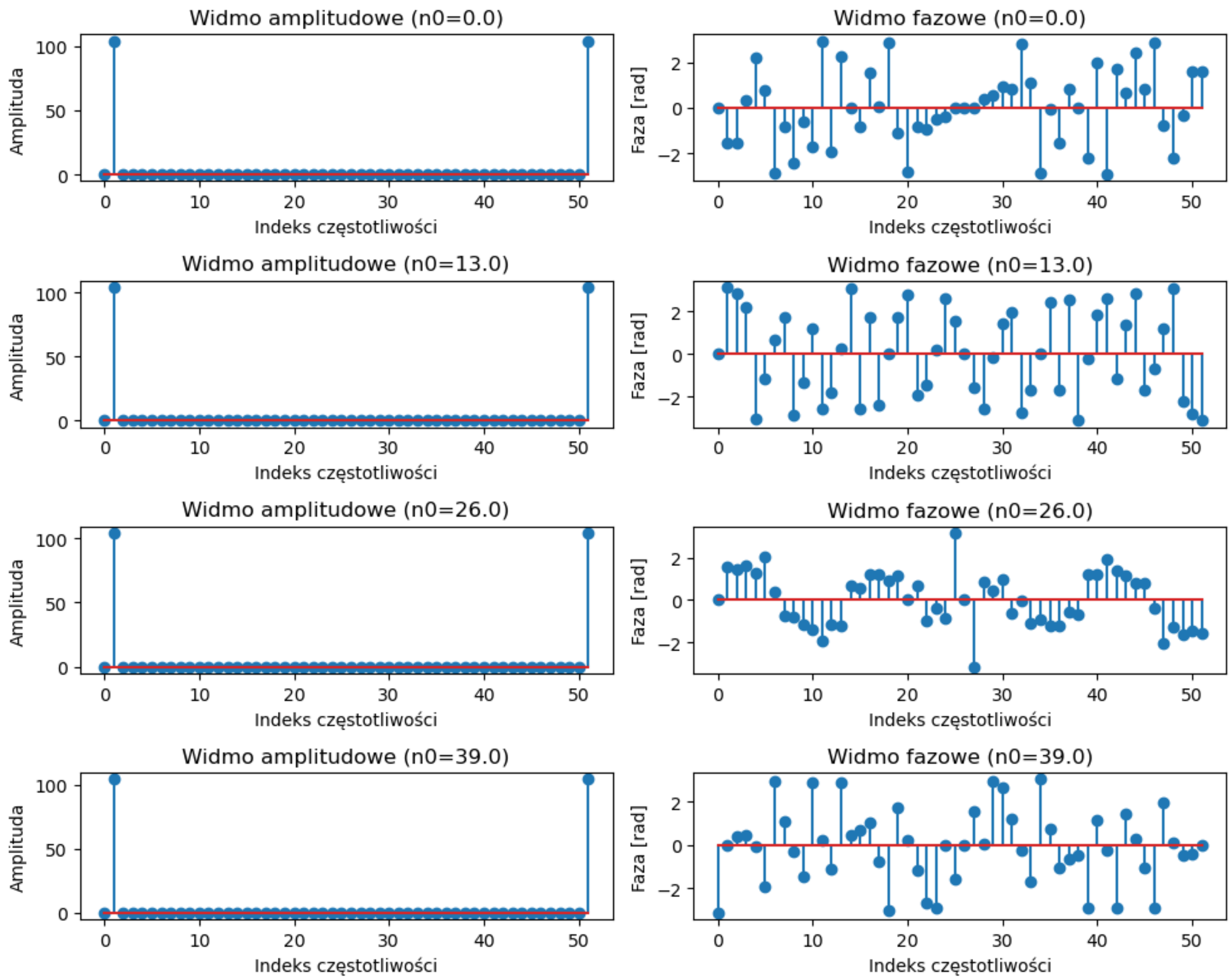
Wniosek: Wykres ma kształt zbliżony do liniowego, jednak zauważalna jest tendencja do wzrostu szybszego niż liniowy. Stąd można założyć, że kształt wykresu asymptotycznie opisuje krzywa N * log N, co potwierdza teoretyczną złożoność obliczeniową algorytmu. Czynnik log N będzie wraz ze wzrostem N miał coraz mniejsze znaczenie, dlatego dla dużych N kształt przypomina linię prostą.

Zadanie 2

Zbadać wpływ przesunięcia w czasie na postać widma amplitudowego i widma fazowego dyskretnego sygnału harmonicznego $s[n] = A\ sin\ (2\pi n\ /\ N)$ 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.

```
A = 4
N = 52.

signal = lambda n, bias=0: A * np.sin(2 * np.pi * (n - bias) / N)

n = np.arange(N)
n0_array = np.array([0, N/4, N/2, 3 * N/4])

def calc_spectrum(signal):
    return fft(signal)

def calc_amplitude_spectrum(signal):
    return np.abs(calc_spectrum(signal))


def calc_phase_spectrum(signal):
    angles = np.angle(calc_spectrum(signal))

    zero_angles = abs(angles) < 0.01
    angles *= ~zero_angles

    return angles


fig, axes = plt.subplots(len(n0_array), 2, figsize=(10, 8))

for i, n0 in enumerate(n0_array):
    shifted_signal = signal(n, bias=n0)
    amplitude_spectrum = calc_amplitude_spectrum(shifted_signal)
    phase_spectrum = calc_phase_spectrum(shifted_signal)

    axes[i, 0].stem(amplitude_spectrum)
    axes[i, 0].set_title(f"Widmo amplitudowe (n0={n0})")
    axes[i, 0].set_xlabel("Indeks częstotliwości")
    axes[i, 0].set_ylabel("Amplituda")
```

```
    axes[i, 1].stem(phase_spectrum)
    axes[i, 1].set_title(f"Widmo fazowe (n0={n0})")
    axes[i, 1].set_xlabel("Indeks częstotliwości")
    axes[i, 1].set_ylabel("Faza [rad]")

plt.tight_layout()
plt.show()
```



Wnioski:

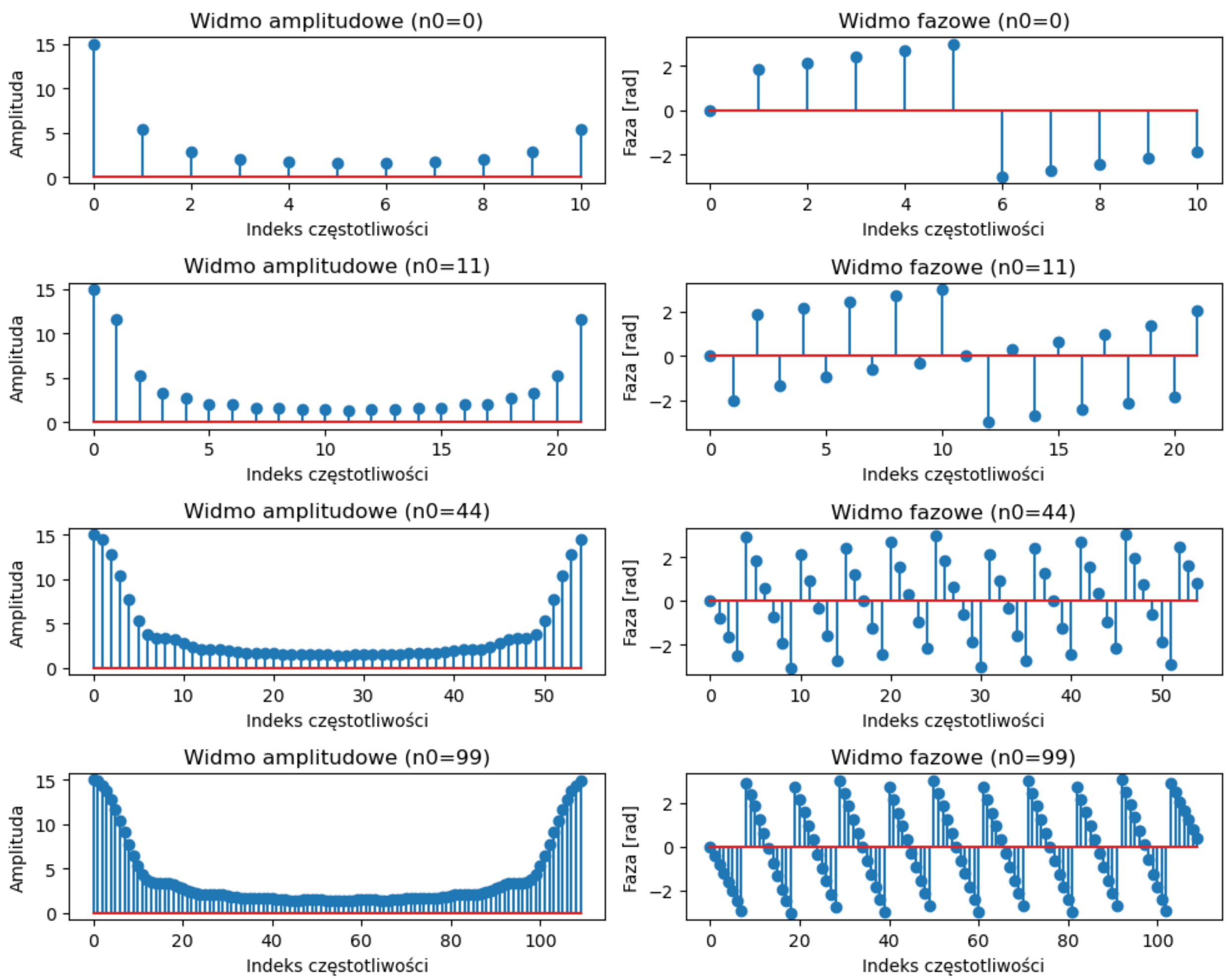
Dodanie przesunięcia w czasie nie miało wpływu na przbieg widma amplitudowego, ponieważ dany punkt z widma obracamy jedynie wokół środka zespolonego układu współrzędnych, zatem amplituda czyli odległość tego punktu od środka się nie zmienia. Jednak znacząco zmienia kształt i przebieg widma fazowego, co widać na powyższych diagramach.

Zadanie 3

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

```
import numpy as np
from scipy.fft import fft
import matplotlib.pyplot as plt

N = 11
A = 3
signal = lambda n: A * (n % N) / N

n = np.arange(N)
N0 = [0, 1 * N, 4 * N, 9 * N]

def calc_spectrum(signal):
    return fft(signal)

def calc_amplitude_spectrum(signal):
    return np.abs(calc_spectrum(signal))


def calc_phase_spectrum(signal):
    angles = np.angle(calc_spectrum(signal))

    zero_angles = abs(angles) < 0.01
    angles *= ~zero_angles

    return angles


fig, axes = plt.subplots(len(N0), 2, figsize=(10, 8))


for i, n0 in enumerate(N0):
    calculated_signal = signal(n)
    for _ in range(n0):
        calculated_signal = np.append(calculated_signal, 0)

    amplitude_spectrum = calc_amplitude_spectrum(calculated_signal)
    phase_spectrum = calc_phase_spectrum(calculated_signal)


    axes[i, 0].stem(amplitude_spectrum)
    axes[i, 0].set_title(f"Widmo amplitudowe (n0={n0})")
    axes[i, 0].set_xlabel("Indeks częstotliwości")
    axes[i, 0].set_ylabel("Amplituda")
```

```
    axes[i, 1].stem(phase_spectrum)
    axes[i, 1].set_title(f"Widmo fazowe (n0={n0})")
    axes[i, 1].set_xlabel("Indeks częstotliwości")
    axes[i, 1].set_ylabel("Faza [rad]")

plt.tight_layout()
plt.show()
```



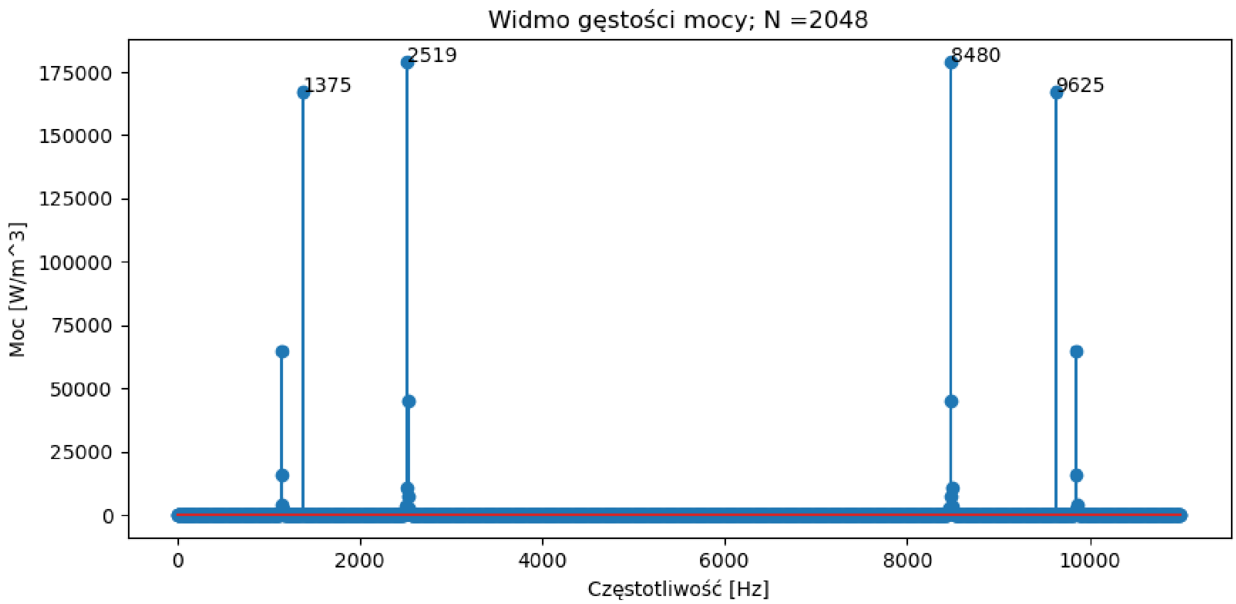
Wniosek: Dopełnienie zerami poprawia rozdzielczość widma, ale nie wpływa na zmianę charakterystyki częstotliwościowej. Poprawa rozdzielczości może ukazać odmienny kształt wykresu, co widać po widmach fazowych. Początkowo zagęszczenie słupków na wykresie było małe, co w połączeniu z funkcją modulo, a zatem pewną okresowością, może nie ukazywać pełni wykresu.

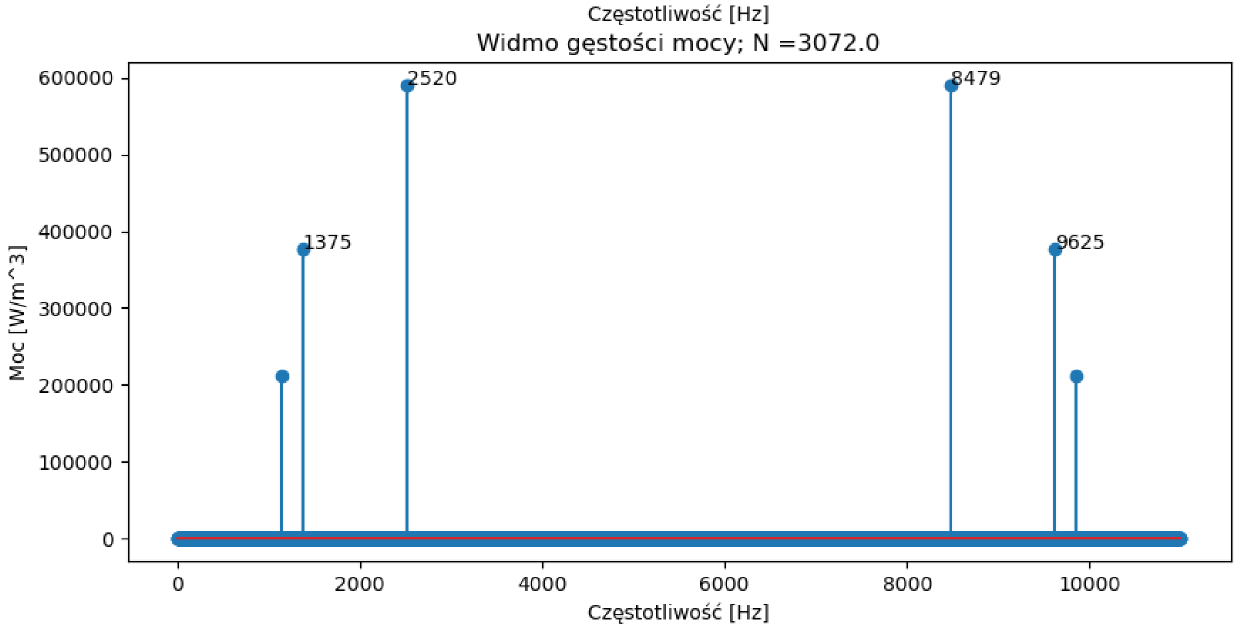
Zadanie 4

Dany jest sygnał rzeczywisty $s(t) = A1\ sin(2\pi f1t) + A2\ sin(2\pi f2t) + A3\ sin(2\pi f3t)$, gdzie $A1 = 0.3$, $f1 = 5000$ Hz, $A2 = 0.4$, $f2 = 6000$ Hz, $A3 = 0.5$, $f3 = 11000$ 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 $N2 = (3/2) * N1$? Odpowiedź uzasadnić.

```
def sample_signal(signal, period, num_samples,
sampling_frequency=None):
    freq = 1 / period
    if sampling_frequency is not None:
        delta_t = 1 / sampling_frequency
        period = num_samples * delta_t
    else:
        delta_t = period / num_samples
    dirac_d = np.arange(0, period, delta_t)
    delta_f = freq / num_samples
    buckets = np.arange(0, freq, delta_f)
    return signal(dirac_d), dirac_d, buckets


make_discrete_signal = lambda signal, num_samples: sample_signal(
    signal, num_samples, num_samples
)[0]


A_list = [0.3, 0.4, 0.5]
f_list = [5000, 6000, 11000]
N = 2048
sampling_frequency = 48000
largest_frequency = f_list[-1]
sampling_period = 1 / sampling_frequency
n_labels = 4

signal = lambda x: sum([A * np.sin(2 * np.pi * f * x) for A, f in
zip(A_list, f_list)])


samples, probing_signal, buckets = sample_signal(
    signal, 1 / largest_frequency, N, sampling_frequency
)
spectrum = np.fft.fft(samples)
spectral_power_density = np.abs(spectrum) ** 2

_, ax = plt.subplots(2, 1, figsize=(10, 10))

ax[0].stem(buckets, spectral_power_density)
```

```
ax[0].set_title("Widmo gęstości mocy; N =" + str(N))
ax[0].set_xlabel("Częstotliwość [Hz]")
ax[0].set_ylabel("Moc [W/m^3]")

largest_indices = np.argsort(spectral_power_density)[-n_labels:]

for i in largest_indices:
    ax[0].annotate(f"{int(buckets[i])}", xy=(buckets[i],
spectral_power_density[i]))


N = (3 / 2) * N

samples, probing_signal, buckets = sample_signal(
    signal, 1 / largest_frequency, N, sampling_frequency
)
spectrum = np.fft.fft(samples)
spectral_power_density = np.abs(spectrum) ** 2

ax[1].stem(buckets, spectral_power_density)
ax[1].set_title("Widmo gęstości mocy; N =" + str(N))
ax[1].set_xlabel("Częstotliwość [Hz]")
ax[1].set_ylabel("Moc [W/m^3]")

largest_indices = np.argsort(spectral_power_density)[-n_labels:]

for i in largest_indices:
    ax[1].annotate(f"{int(buckets[i])}", xy=(buckets[i],
spectral_power_density[i]))

plt.show()
```





Czy dla podanej liczby próbek mamy do czynienia ze zjawiskiem przecieku widma?
W przypadku N = 2048 widać na wykresie przeciek widma.

Czy sytuacja uległaby zmianie dla liczby próbek N2 = 3 * N1 / 2?
Tak, sytuacja ulega zmianie. Jest to związane z końcem okresu próbkowania, jeśli skonczymy próbkowanie w miejscu, w którym kończy się okres funkcji, to nie będzie przecieku widma. Częstotliwość próbkowania (48kHz) jest 48 razy większa niż 1kHz - NWD częstotliwości podstawowych składowych sygnału (5kHz, 6kHz i 11kHz). 48 dzieli liczbę 3072, a nie 2048, stąd koniec próbkowania przypada na koniec okresu funkcji.