


Wydział Informatyki Katedra Mediów Cyfrowych i Grafiki Komputerowej Pracownia Specjalistyczna Przetwarzanie sygnałów i obrazów	Data: 2.04.2025
Ćwiczenie nr 3 Temat: Dyskretne przekształcenie Fouriera Grupa nr Konrad Łupiński Kacper Żebrowski	Prowadzący: Mgr inż. Aleksander Sawicki Ocena: 

1. Wstęp:

Zapoznanie się transformatą Fouriera oraz bibliotekami `scipy.fft()` i `scipy.ifft()` w języku Python w jupyter notebook.

2. Zadanie 3.1:

Wygeneruj dokładnie 1 okres fali sinusoidalnej (32 lub 64 próbki). Sporządź wykresy sinusoidy i jej transformaty Fouriera w jednym oknie (część rzeczywistą, urojoną, moduł i kąt). Co możesz powiedzieć o symetrii widma zespolonego?

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

N = 64
f = 5
fs = 64
A = 1
phi = 0

n = np.arange(N)
y = A * np.sin(2 * np.pi * f * n / fs + phi)

Y = np.fft.fft(y)
frequencies = np.fft.fftfreq(N, d=1/fs)

real_part = np.real(Y)
imag_part = np.imag(Y)
magnitude = np.abs(Y)
phase = np.angle(Y)
```

```
plt.figure(figsize=(10, 8))

plt.subplot(3, 2, 1)
plt.plot(n, y, marker='o')
plt.title("Sygnał sinusoidalny")
plt.xlabel("Numer próbek")
plt.ylabel("Amplituda")
plt.grid()

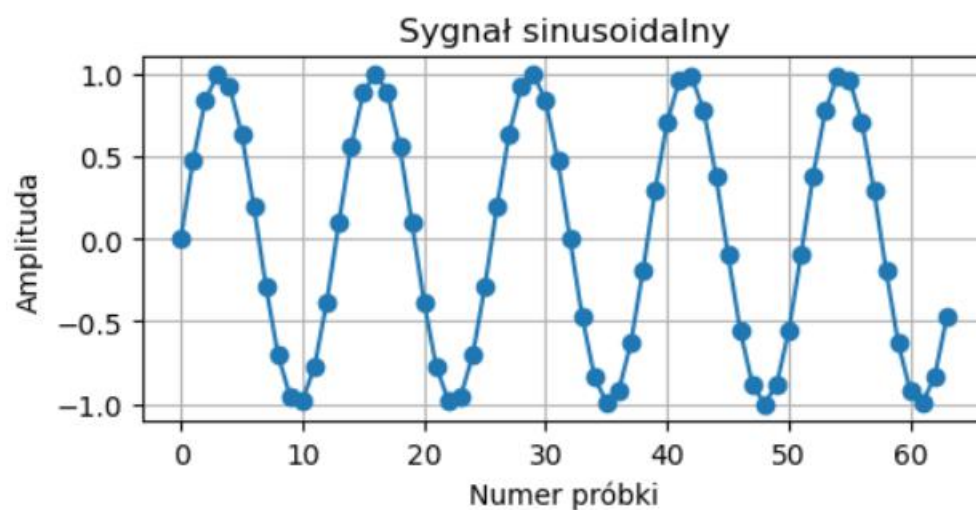
plt.subplot(3, 2, 2)
plt.stem(frequencies, real_part)
plt.title("Część rzeczywista FFT")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Re(X[k])")
plt.grid()

plt.subplot(3, 2, 3)
plt.stem(frequencies, imag_part)
plt.title("Część urojona FFT")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Im(X[k])")
plt.grid()

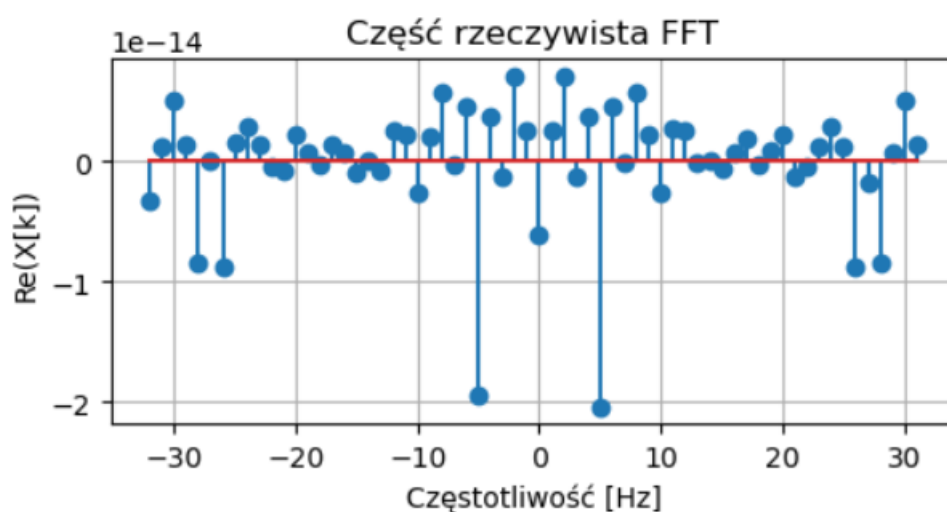
plt.subplot(3, 2, 4)
plt.stem(frequencies, magnitude)
plt.title("Moduł FFT")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("|X[k]|")
plt.grid()

plt.subplot(3, 2, 5)
plt.stem(frequencies, phase)
plt.title("Faza FFT")
plt.xlabel("Częstotliwość [Hz]")
plt.ylabel("Kąt (rad)")
plt.grid()

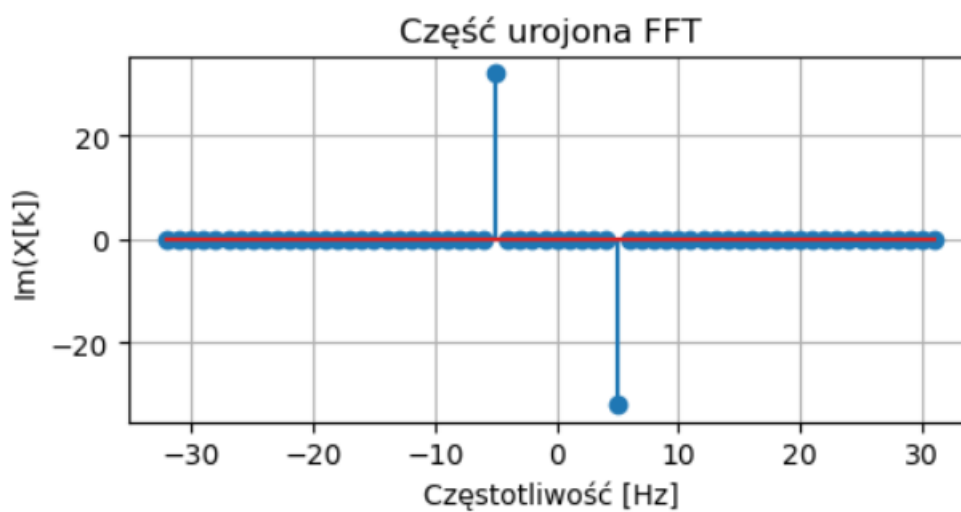
plt.tight_layout()
plt.show()
```



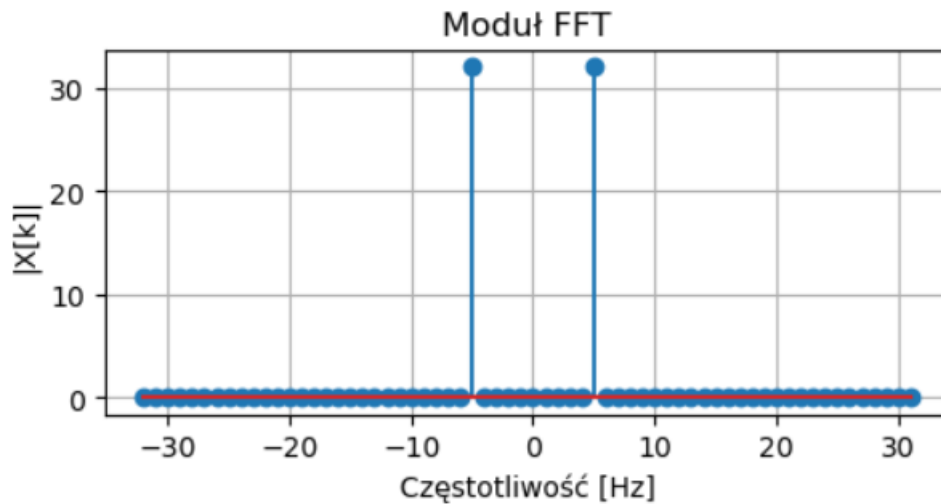
Rys1. Graficzna reprezentacja sygnału sinusoidalnego



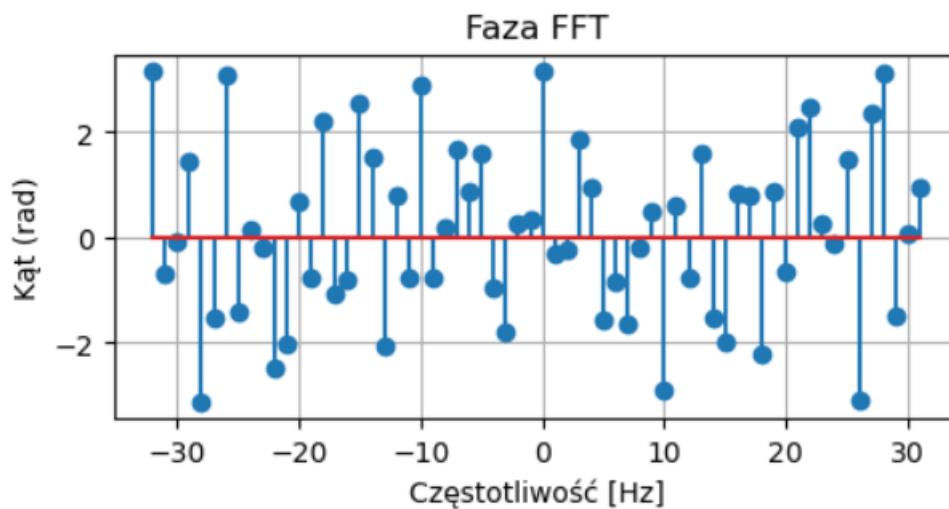
Rys2. Graficzna reprezentacja części rzeczywistej widma sygnału



Rys3. Graficzna reprezentacja części urojonej widma sygnału



Rys4. Graficzna reprezentacja modułu z części urojonej widma sygnału



Rys5. Graficzna reprezentacja przesunięcia fazowego dla widma sygnału

Widmo zespolone sinusoidy ma symetrię lustrzaną względem częstotliwości zerowej, gdzie część rzeczywista jest symetryczna, część urojona jest antysymetryczna, a moduł jest identyczny dla częstotliwości dodatnich i ujemnych. Jest to tzw. symetria hermitowska, charakterystyczna dla transformaty Fouriera sygnałów rzeczywistych.

3. Zadanie 3.2:

Wygeneruj następujące sygnały: $y_1[n] = \cos(2\pi n/N + \pi/4)$, $y_2[n] = 0.5\cos(4\pi n/N)$ oraz $y_3[n] = 0.25\cos(8\pi n/N + \pi/2)$, gdzie N - liczba próbek sygnału. Wyznacz ich transformaty Fouriera. Obliczenia powtórz dla sygnału $y_4 = y_1 + y_2 + y_3$. Jaki jest związek pomiędzy amplitudą, fazą, liczbą okresów poszczególnych sygnałów a

wartościami widma zespolonego? Jak zachowuje się funkcja fft w stosunku do sumy sygnałów?

```
N = 64
n = np.arange(N)

y1 = np.cos(2 * np.pi * n / N + np.pi / 4)
y2 = 0.5 * np.cos(4 * np.pi * n / N)
y3 = 0.25 * np.cos(8 * np.pi * n / N + np.pi / 2)
y4 = y1 + y2 + y3

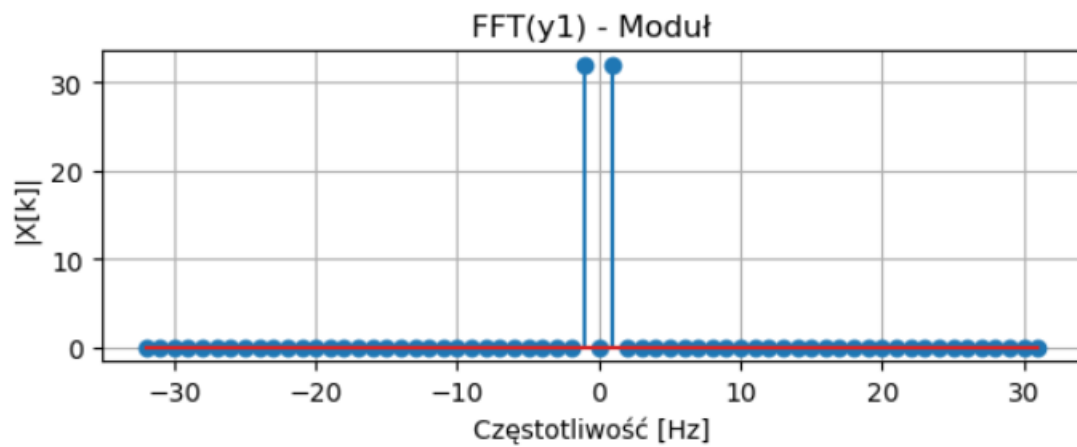
Y1 = np.fft.fft(y1)
Y2 = np.fft.fft(y2)
Y3 = np.fft.fft(y3)
Y4 = np.fft.fft(y4)
frequencies = np.fft.fftfreq(N, d=1 / N)

def plot_fft(Y, title, subplot_index):
    plt.subplot(4, 2, subplot_index)
    plt.stem(frequencies, np.abs(Y))
    plt.title(f"{title} - Moduł")
    plt.xlabel("Częstotliwość [Hz]")
    plt.ylabel("|X[k]|")
    plt.grid()

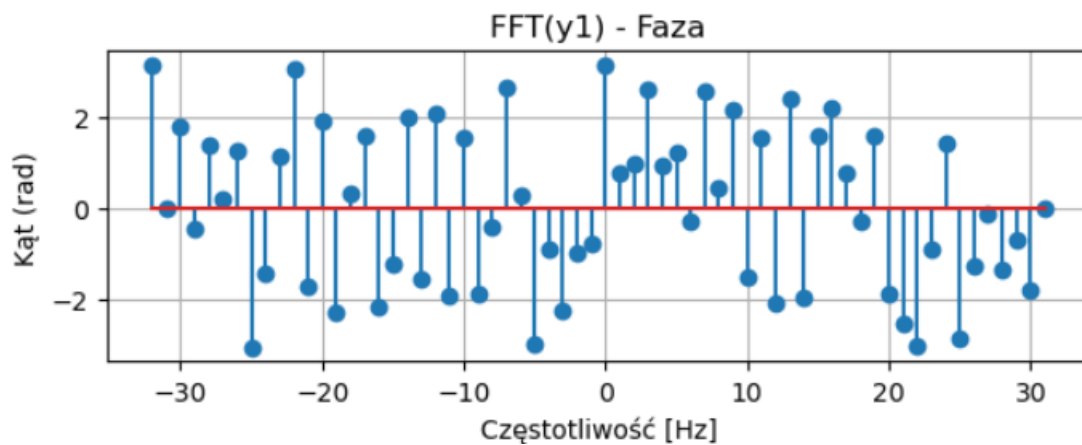
    plt.subplot(4, 2, subplot_index + 1)
    plt.stem(frequencies, np.angle(Y))
    plt.title(f"{title} - Faza")
    plt.xlabel("Częstotliwość [Hz]")
    plt.ylabel("Kąt (rad)")
    plt.grid()

plt.figure(figsize=(12, 10))
plot_fft(Y1, "FFT(y1)", 1)
plot_fft(Y2, "FFT(y2)", 3)
plot_fft(Y3, "FFT(y3)", 5)
plot_fft(Y4, "FFT(y4)", 7)
```

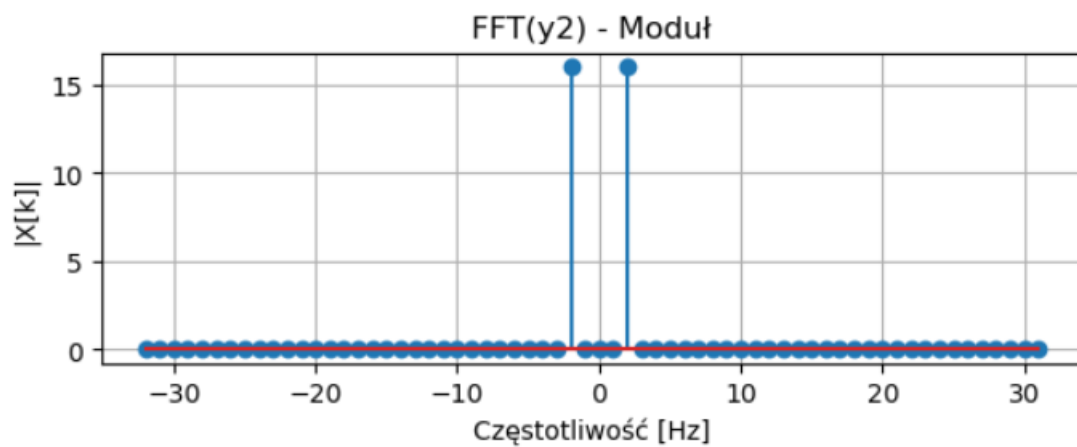
```
plt.tight_layout()
plt.show()
```



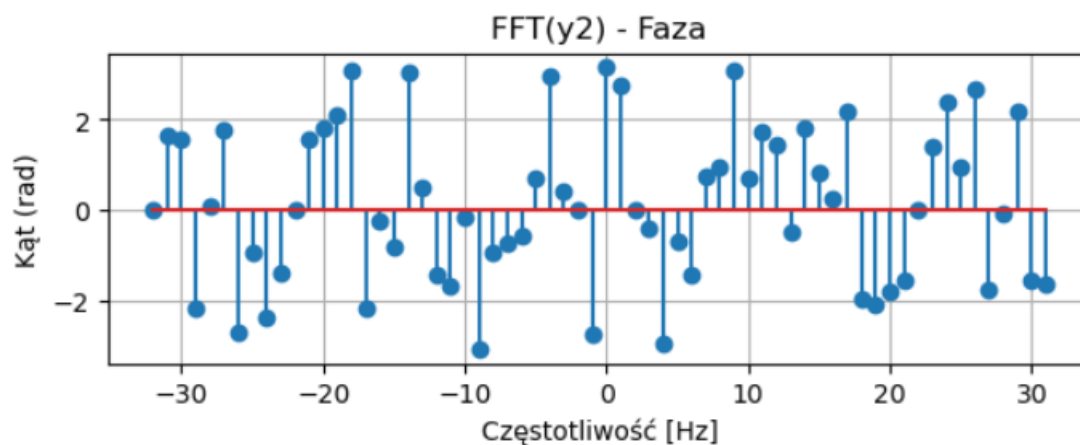
Rys6. Graficzna reprezentacja modułu z części urojonej widma sygnału o wzorze $y_1[n] = \cos(2\pi n/N + \pi/4)$



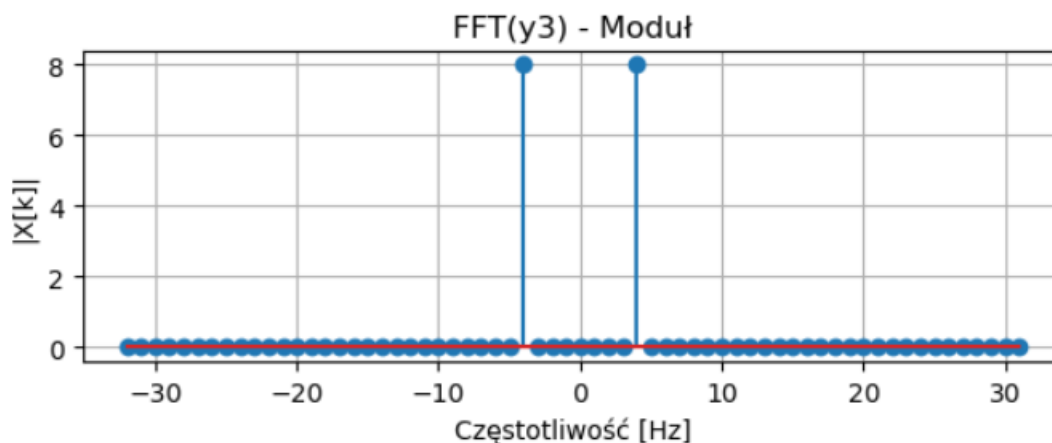
Rys7. Graficzna reprezentacja przesunięcia fazowego dla widma sygnału o wzorze $y_1[n] = \cos(2\pi n/N + \pi/4)$



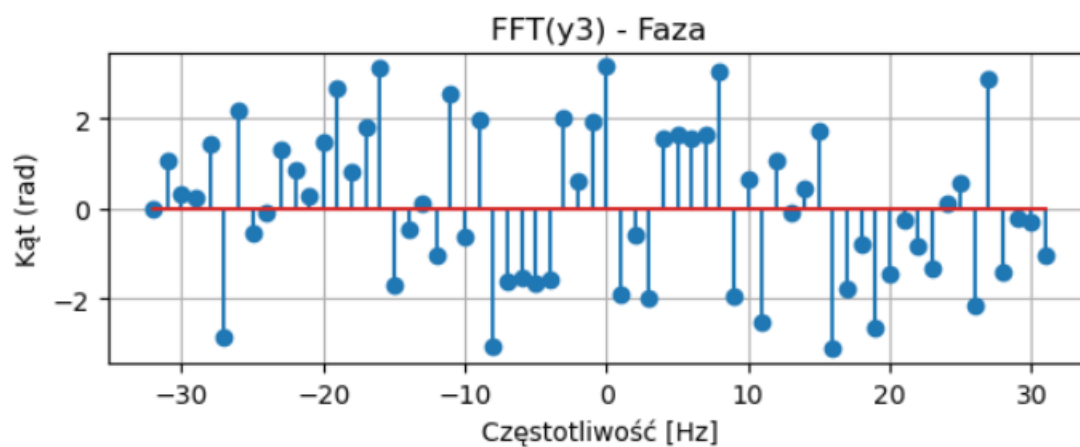
Rys8. Graficzna reprezentacja modułu z części urojonej widma sygnału o wzorze $y_2[n] = 0.5\cos(4\pi n/N)$



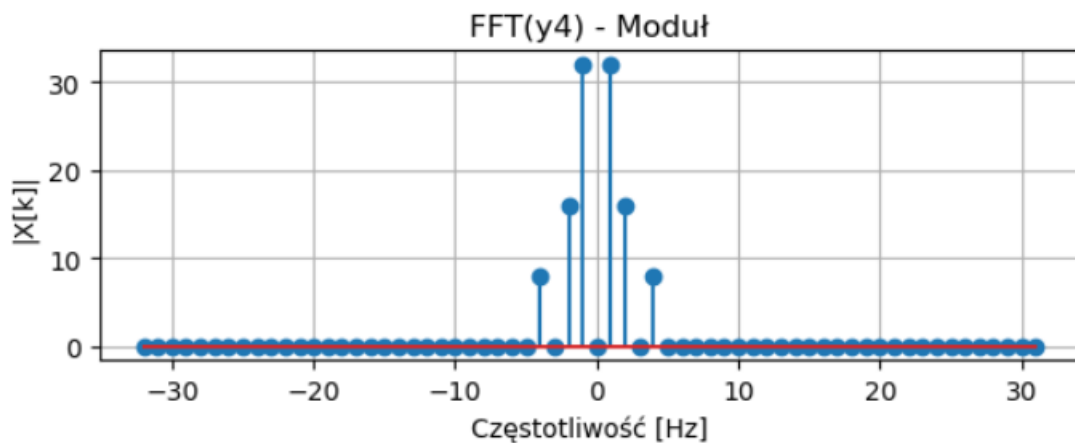
Rys9. Graficzna reprezentacja przesunięcia fazowego dla widma sygnału o wzorze $y_2[n] = 0.5\cos(4\pi n/N)$



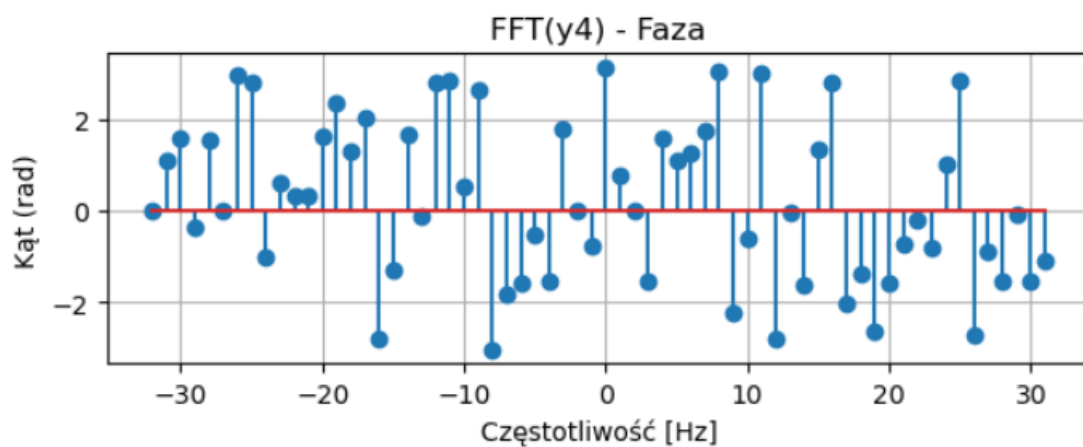
Rys10. Graficzna reprezentacja modułu z części urojonej widma sygnału o wzorze $y_3[n] = 0.25\cos(8\pi n/N + \pi/2)$



Rys11. Graficzna reprezentacja przesunięcia fazowego dla widma sygnału o wzorze $y_3[n] = 0.25\cos(8\pi n/N + \pi/2)$



Rys12. Graficzna reprezentacja modułu z części urojonej widma sygnału y4



Rys13. Graficzna reprezentacja przesunięcia fazowego dla widma sygnału y4

Amplituda sygnału wpływa na wysokość pików w widmie im większa, tym wyższy pik. Faza sygnału decyduje o kącie w widmie zespolonym, ale nie zmienia jego częstotliwości. Liczba okresów sygnału w oknie analizy wpływa na „czystość” widma im bardziej całkowita liczba okresów, tym ostrzejsze i bardziej wyraźne piki. Funkcja FFT zastosowana do sumy sygnałów działa w sposób liniowy, co oznacza, że wynikowe widmo jest efektem nałożenia się widm poszczególnych sygnałów składowych.

4. Zadanie 3.3:

Wyznacz odwrotną transformatę Fouriera dla widm zespolonych FFT otrzymanych w zadaniach 3.1 i 3.2. Jak zinterpretujesz otrzymane wyniki?

```
y1_reconstructed = np.fft.ifft(Y1).real
y2_reconstructed = np.fft.ifft(Y2).real
y3_reconstructed = np.fft.ifft(Y3).real
```



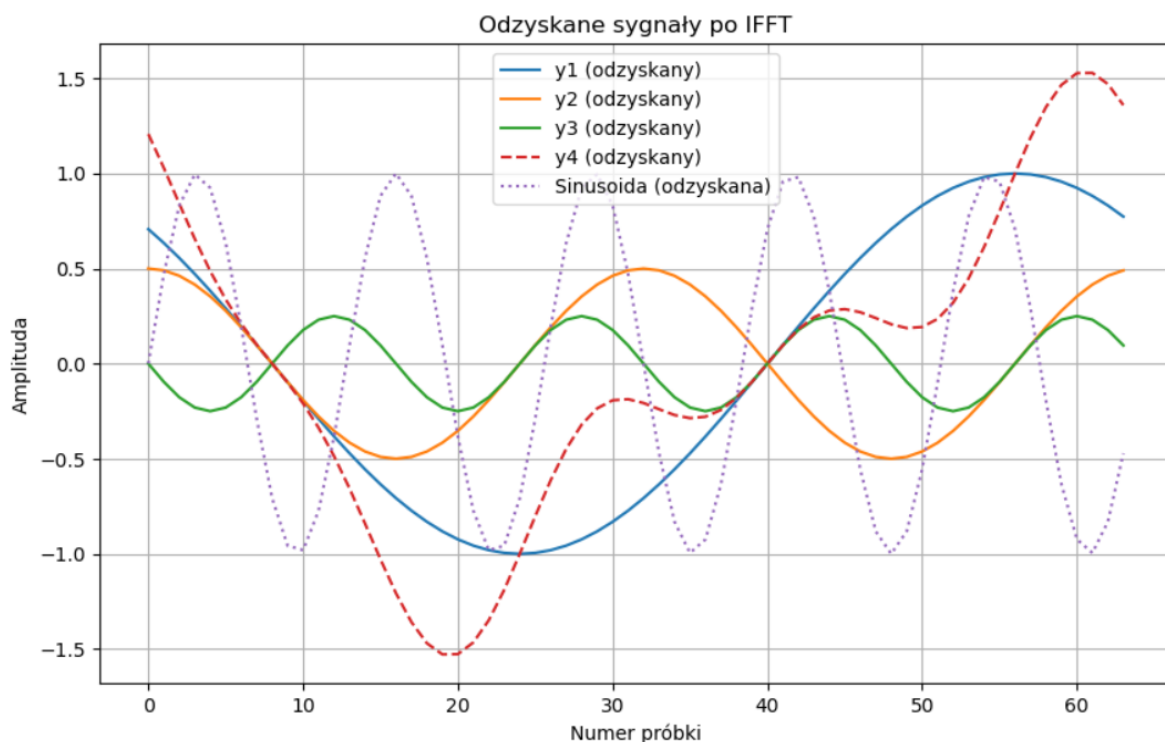
```

y4_reconstructed = np.fft.ifft(Y4).real

Y_sin = np.fft.fft(y)
y_sin_reconstructed = np.fft.ifft(Y_sin).real

plt.figure(figsize=(10, 6))
plt.plot(n, y1_reconstructed, label='y1 (odzyskany)')
plt.plot(n, y2_reconstructed, label='y2 (odzyskany)')
plt.plot(n, y3_reconstructed, label='y3 (odzyskany)')
plt.plot(n, y4_reconstructed, label='y4 (odzyskany)',
linestyle='dashed')
plt.plot(n, y_sin_reconstructed, label='Sinusoida (odzyskana)',
linestyle='dotted')
plt.legend()
plt.title("Odzyskane sygnały po IFFT")
plt.xlabel("Numer próbek")
plt.ylabel("Amplituda")
plt.grid()
plt.show()

```



Rys14. Porównanie graficznych reprezentacji odzyskanych sygnałów

Otrzymane wyniki pokazują, że dzięki odwrotnej transformacie Fouriera można skutecznie odzyskać sygnały czasowe z ich widm częstotliwościowych. Odtworzone sygnały są praktycznie identyczne z oryginałami, co potwierdza poprawność przeprowadzonych obliczeń. Niewielkie różnice mogą wynikać z błędów numerycznych związanych z reprezentacją zespoloną i precyzją obliczeń.

5. Zadanie 3.4:

Powtórz zadanie 3.1 dla sinusoidy zespolonej tj. $y[n] = \exp(i(\omega n + \phi))$, gdzie ω - pulsacja unormowana oraz ϕ - przesunięcie fazowe. UWAGA, przyjąć, że $\omega = 2k/N$, gdzie N - liczba próbek sygnału oraz k - dowolna liczba całkowita. Jakie są różnice (w symetrii) w stosunku do widma sygnału rzeczywistego?

```
N = 64
k = 2
phi = np.pi / 4
w = 2 * np.pi * k / N

n = np.arange(N)
y = np.exp(1j * (w * n + phi))
Y = np.fft.fft(y)

fig, axs = plt.subplots(3, 2, figsize=(14, 7))

axs[0, 0].stem(n, y.real)
axs[0, 0].set_title('sinusoida zespolona część rzeczywista')
axs[0, 0].set_xlabel('liczba próbek N')
axs[0, 0].set_ylabel('amplituda')

axs[0, 1].stem(n, y.imag)
axs[0, 1].set_title('sinusoida zespolona część urojona')
axs[0, 1].set_xlabel('liczba próbek N')
axs[0, 1].set_ylabel('amplituda')

axs[1, 0].stem(n, Y.real)
axs[1, 0].set_title('Wykres - czesc rzeczywista fft dla funkcji y')
axs[1, 0].set_xlabel('Liczba próbek N')
axs[1, 0].set_ylabel('Amplituda')

axs[1, 1].stem(n, Y.imag)
```

```

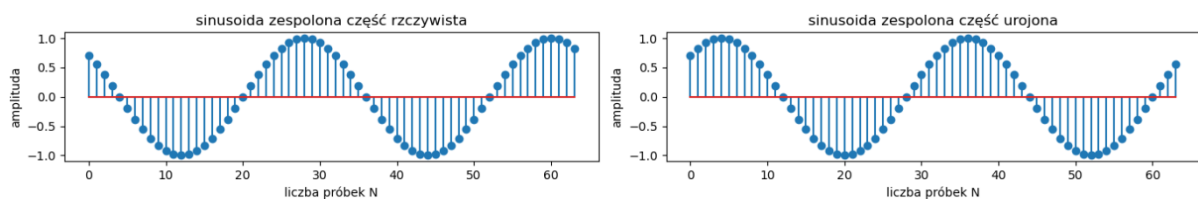
axs[1, 1].set_title('Wykres - czesc urojona fft dla funkcji y')
axs[1, 1].set_xlabel('Liczba próbek N')
axs[1, 1].set_ylabel('Amplituda')

axs[2, 0].stem(n, np.abs(Y))
axs[2, 0].set_title('Wykres - modul fft dla funkcji y')
axs[2, 0].set_xlabel('Liczba próbek N')
axs[2, 0].set_ylabel('Amplituda')

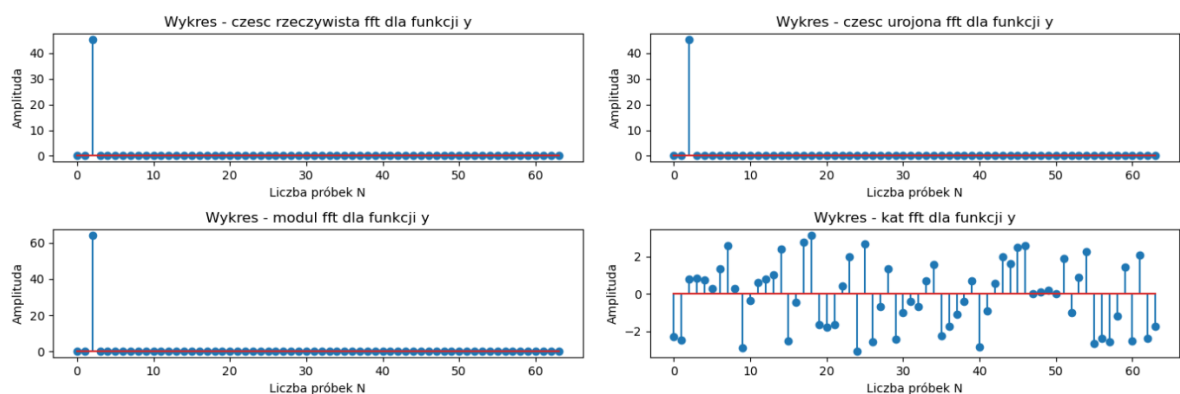
axs[2, 1].stem(n, np.angle(Y))
axs[2, 1].set_title('Wykres - kat fft dla funkcji y')
axs[2, 1].set_xlabel('Liczba próbek N')
axs[2, 1].set_ylabel('Amplituda')

plt.tight_layout()
plt.show()

```



Rys15. Graficzna reprezentacja części rzeczywistej i urojonej zespolonego sygnału sinusoidalnego



Rys16. Analiza widmowa sygnału zespolonego

Widmo sygnału rzeczywistego jest symetryczne. Widmo sygnału zespolonego nie jest symetryczne. W widmie sygnału rzeczywistego moduł wykazuje symetryczność, a faza antysymetryczność, jednak w widmie sygnału zespolonego ani moduł, ani faza nie wykazują takiej zależności.

6. Zadanie 3.5:

Wygeneruj 2.5 okresu sinusoidy. Sprawdź jak wygląda moduł widma zespolonego FFT sygnału. Skąd biorą się zniekształcenia? Porównaj z FFT sygnału wymnożonego przez funkcję okna. Zastosuj funkcję window oraz okna: bartlett, blackman, hamming, hann, kaiser, triang.

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

f = 1
fs = 32
t_end = 2.5 / f
t = np.arange(0, t_end, 1/fs)

signal = np.sin(2 * np.pi * f * t)

windows = {
    "Bartlett": np.bartlett(len(t)),
    "Blackman": np.blackman(len(t)),
    "Hamming": np.hamming(len(t)),
    "Hann": np.hanning(len(t)),
    "Kaiser": np.kaiser(len(t), beta=14),
    "Triang": np.bartlett(len(t))
}

n_samples = len(t)

fft_signal = np.fft.fft(signal)
magnitude_spectrum = np.abs(fft_signal)

plt.figure(figsize=(12, 18))

plt.subplot(9, 1, 1)
plt.stem(np.arange(n_samples), signal, basefmt=" ")
plt.title('Sygnał sinusoidalny')
plt.xlabel('Indeks próbki')
plt.ylabel('Amplituda')
plt.xlim(0, 80)
plt.grid(True)

plt.subplot(9, 1, 2)
plt.stem(np.arange(len(magnitude_spectrum)), magnitude_spectrum,
```

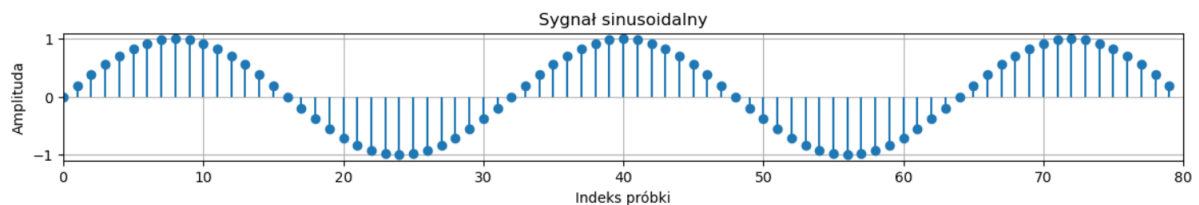
```

basefmt=" ")
plt.title('Moduł widma FFT oryginalnego sygnału')
plt.xlabel('Indeks próbki')
plt.ylabel('Amplituda')
plt.xlim(0, 80)
plt.grid(True)

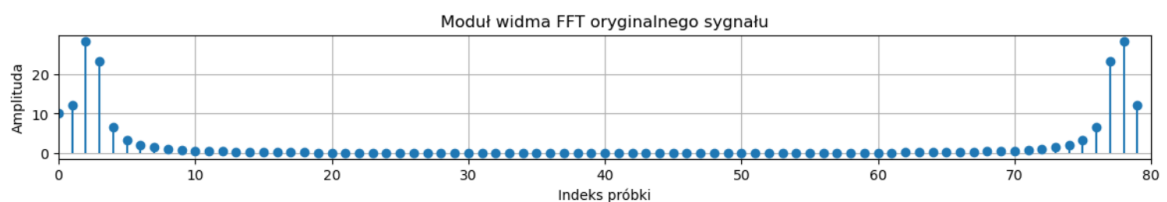
for idx, (window_name, window) in enumerate(windows.items(),
start=3):
    signal_windowed = signal * window
    plt.subplot(9, 1, idx)
    plt.stem(np.arange(n_samples), signal_windowed, basefmt=" ")
    plt.title(f'Sygnał okna {window_name}')
    plt.xlabel('Indeks próbki')
    plt.ylabel('Amplituda')
    plt.xlim(0, 80)
    plt.grid(True)

plt.tight_layout()
plt.show()

```

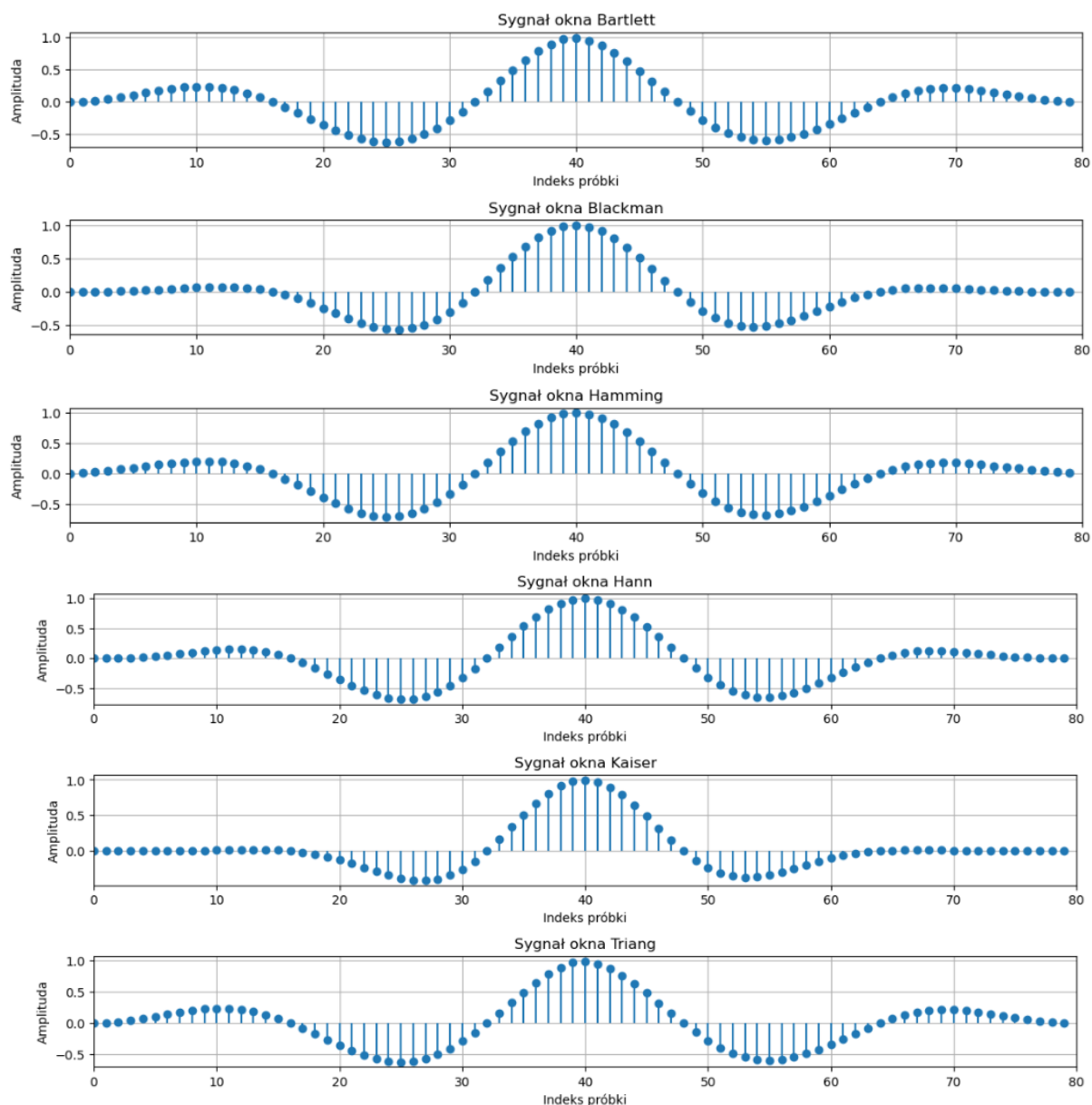


Rys17. Próbką sygnału sinusoidalnego



Rys18. Widmo podstawowego sygnału sinusoidalnego

Zniekształcenia w widmie FFT wynikają z faktu analizowania 2.5 okresu sinusoidy jest to niepełna ilość okresów co doprowadza do nieściśności w sygnale.



Rys19. Amplitudy wykresów dla różnych okien

7. Podsumowanie:

W ramach ćwiczenia zapoznano się z działaniem transformaty Fouriera oraz wykorzystaniem funkcji `scipy.fft()` i `scipy.ifft()` w języku Python. Przeprowadzone analizy pokazały, że sygnały rzeczywiste mają widmo symetrii Hermitowskiej, a sygnały zespolone tej symetrii nie wykazują. Transformata Fouriera jest operacją liniową, co oznacza, że widmo sumy sygnałów odpowiada sumie ich widm. Odwrotna transformata pozwala odtworzyć sygnał bez utraty informacji. Zauważono również, że dla niepełnookesowych sygnałów występuje efekt wycieku widma, które można ograniczyć poprzez zastosowanie funkcji okna. Całość ćwiczenia pozwoliła lepiej zrozumieć

~~analizę częstotliwościową sygnałów i praktyczne aspekty jej realizacji w środowisku Python.~~