In [2]:

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

margin=0.05

def readFile(filename, frequency, hasFirstTime):
    """
    Odczyt danych z pliku tekstowego o nazwei filename
    Dane były próbkowane z częstotliwością frequency
    Parametr hasFirstTime wskazuje, czy pierwsza kolumna zawiera wartości amplitudy
    """
    timePeriod = 1 / frequency

    xarray.clear()
    valuesArray.clear()
    try:
        with open(filename, 'r') as file:
            for line in file:
                columns = line.split()
                if hasFirstTime and len(columns) > 1:
                    xarray.append(float(columns[0]))
                    columns = columns[1:]  # Usun istniejaca kolumne
                else:
                    xarray.append(len(xarray) * timePeriod)

                for i, value in enumerate(columns):
                    if len(valuesArray) <= i:
                        valuesArray.append([])  #Stworz nowa liste
                    valuesArray[i].append(float(value))

        numberOfCases = len(valuesArray)
        return (True, numberOfCases)
    except Exception as e:
        with out:
            print("Błąd odczytu pliku", str(e))
        return (False, -1)

def fourierIOdwrotny(x, time,fs, nazwa, k=1000, eliminate0=True):
    """
    Funkcja odpowiedzialna za wizualizajcę funkcji, jej transformaty Fouriera i jej
    """
    #sygnal
    T=1/fs
    plt.figure()
    plt.plot(time[:k],x[:k])
    plt.title(nazwa)
    plt.xlabel('Czas [s]')
    plt.ylabel('Amplituda')
    plt.tight_layout()
    plt.grid(True)
    plt.show()

    #fourier
    n=len(x)
```

```
    fourier=np.fft.fft(x)
    freq_axis = np.fft.fftfreq(n, d=1/fs)
    modul=abs(fourier)
    #modul2=modul
    modul[0]=modul[1]
    plt.figure()
    plt.plot(freq_axis, modul)
    plt.title('Widmo amplitudowe sygnału - '+ nazwa)
    plt.xlabel('Częstotliwość [Hz]')
    plt.ylabel('Amplituda')
    plt.tight_layout()
    plt.grid(True)
    xmin, xmax = 0, fs/2  # Desired domain
    xmargin = (xmax - xmin) * margin
    plt.xlim(xmin - xmargin, xmax + xmargin)
    plt.show()

    #transformata odwrotna
    odwrotna= np.fft.ifft(fourier)
    modulOdwrotna=abs(odwrotna)
    modulFunkcji=[abs(ele) for ele in x]
    roznica=modulOdwrotna-modulFunkcji

    plt.subplot(1, 2, 1)
    plt.plot(time[:k],modulFunkcji[:k])
    plt.title(nazwa)
    plt.xlabel('Czas [s]')
    plt.ylabel('Amplituda')
    plt.title('Oryginalny')
    plt.subplot(1, 2, 2)
    plt.plot(time[:k],modulOdwrotna[:k])
    plt.title(nazwa)
    plt.xlabel('Czas [s]')
    plt.ylabel('Amplituda')
    plt.title('Po transformacie')

    plt.figure()
    plt.plot(time[:k], roznica[:k])
    plt.title('Roznica między oryginalem a transformata odwrotna - '+  nazwa)
    plt.xlabel('Czas [s]')
    plt.ylabel('Amplituda')
    plt.grid(True)
    plt.show()


"""
x- sygnal oryginalny
t- wektro czasu
fs- czestotliwosc probkowania
cutoffFrequency - czestotliwosc odciecia
w i h parametry filtra
"""
def ilustrujFiltrowanie(x, t, fs , cutoffFrequency, w, h, nazwaFiltra ):
    fourier=np.fft.fft(x)
    fourier[0]=fourier[1]
    n=len(x)
```

```
    h=np.array([(0.001 if (i==0) else i) for i in h])
    freq_axis=np.fft.fftfreq(n, d=1/fs)
    plt.plot(w,20*np.log10(abs(h)))
    plt.title('Charakterystyka: '+nazwaFiltra)
    plt.xlabel('Czestotliwosc [radian / s]')
    plt.ylabel('Amplituda [dB]')
    plt.margins(0, 0.1)
    plt.grid(which='both', axis='both')
    plt.axvline(cutoffFrequency, color='green', label='Czestoliwosc odciecia') # cu
    plt.legend()
    plt.xlim(0-margin, )
    plt.show()

    filtering=fourier*h
    filtered=np.fft.ifft(filtering)

    plt.figure()
    plt.plot(freq_axis, abs(fourier), '-r', label='Widmo oryginalne')
    plt.plot(freq_axis,abs(filtering), '-b', label='Widmo filtrowane')
    plt.legend()
    xmin, xmax = 0, fs/2  # Desired domain
    xmargin = (xmax - xmin) * margin
    plt.xlim(xmin - xmargin, xmax + xmargin)
    plt.xlabel("Czestotliwosc [Hz]")
    plt.ylabel("Amplituda")
    plt.title("Porównanie widma")
    plt.show()

    """
    plt.figure()
    plt.subplot(1,2,1)
    plt.plot(freq_axis, abs(fourier))
    plt.xlabel('Czestotliwosc [Hz]')
    plt.ylabel('Amplituda')
    xmin, xmax = 0, fs/2  # Desired domain
    xmargin = (xmax - xmin) * margin
    plt.xlim(xmin - xmargin, xmax + xmargin)
    plt.title('Widmo oryginalne')
    plt.subplot(1,2,2)
    plt.plot(freq_axis,abs(filtering))
    plt.xlabel('Czestotliwosc [Hz]')
    plt.ylabel('Amplituda')
    xmin, xmax = 0, fs/2  # Desired domain
    xmargin = (xmax - xmin) * margin
    plt.xlim(xmin - xmargin, xmax + xmargin)
    plt.title('Widmo wyfiltrowane')
    """

    plt.figure()
    plt.plot(t, x , 'r',label='oryginalny')
    plt.plot(t, filtered, 'b',label='wyfiltrowany')
    plt.legend()
    plt.title("Porównanie sygnału w dziedzinie czasu")
    plt.show()
```

```
    return filtered
```

In [ ]: