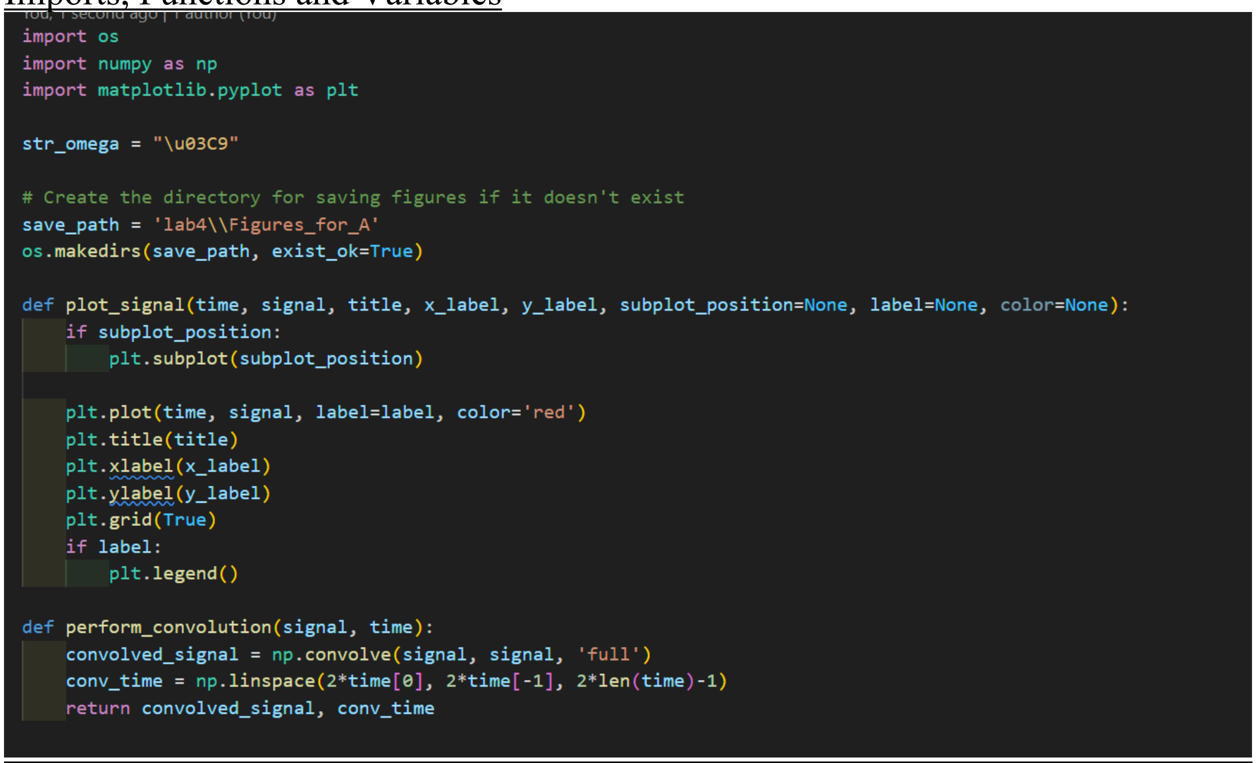
Lab Assignment

Imports, Functions and Variables

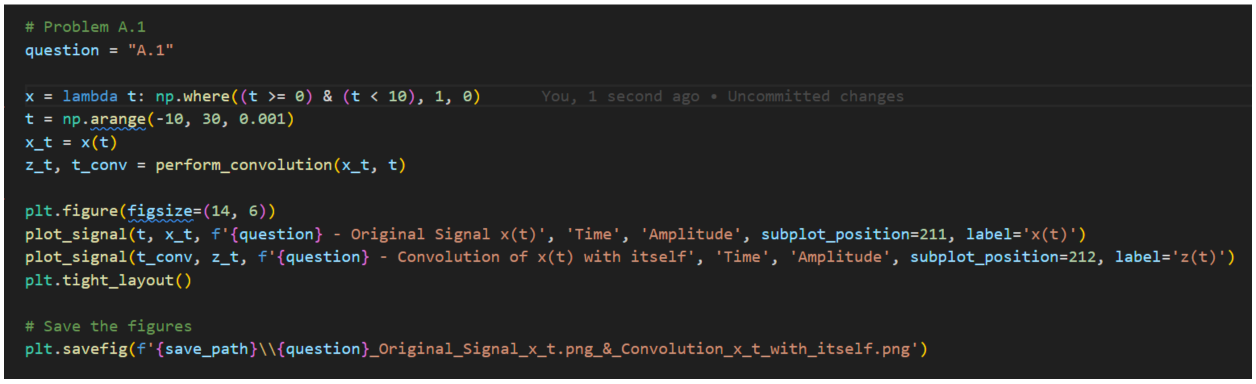


```python
You, 1 second ago | 1 author (You)
import os
import numpy as np
import matplotlib.pyplot as plt

str_omega = "\u03C9"

# Create the directory for saving figures if it doesn't exist
save_path = 'lab4\\Figures_for_A'
os.makedirs(save_path, exist_ok=True)

def plot_signal(time, signal, title, x_label, y_label, subplot_position=None, label=None, color=None):
    if subplot_position:
        plt.subplot(subplot_position)

    plt.plot(time, signal, label=label, color='red')
    plt.title(title)
    plt.xlabel(x_label)
    plt.ylabel(y_label)
    plt.grid(True)
    if label:
        plt.legend()

def perform_convolution(signal, time):
    convolved_signal = np.convolve(signal, signal, 'full')
    conv_time = np.linspace(2*time[0], 2*time[-1], 2*len(time)-1)
    return convolved_signal, conv_time
```

Part A

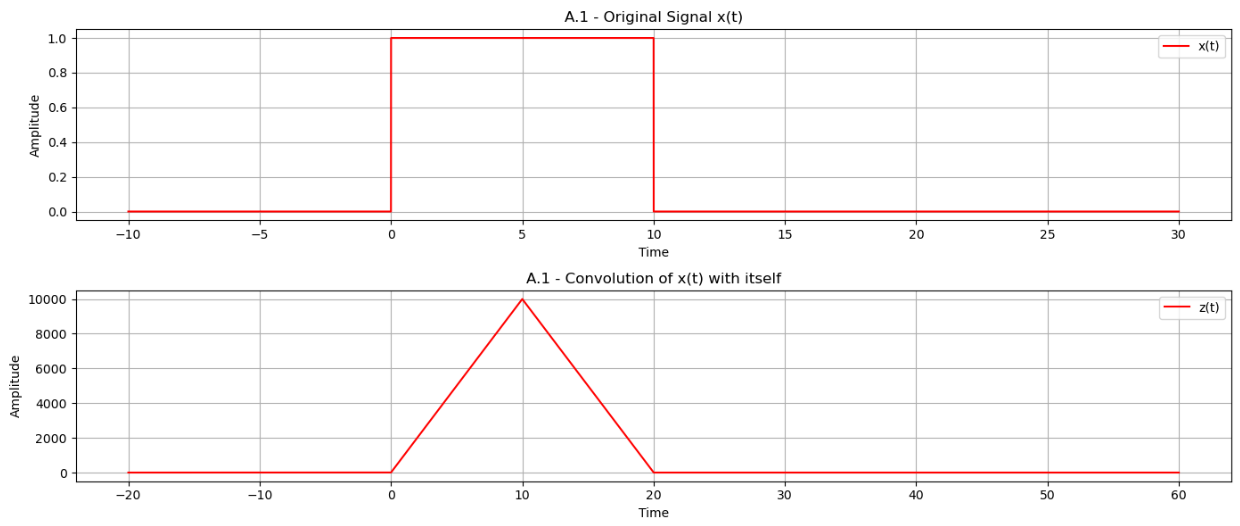
- **Problem A.1**

  Code:



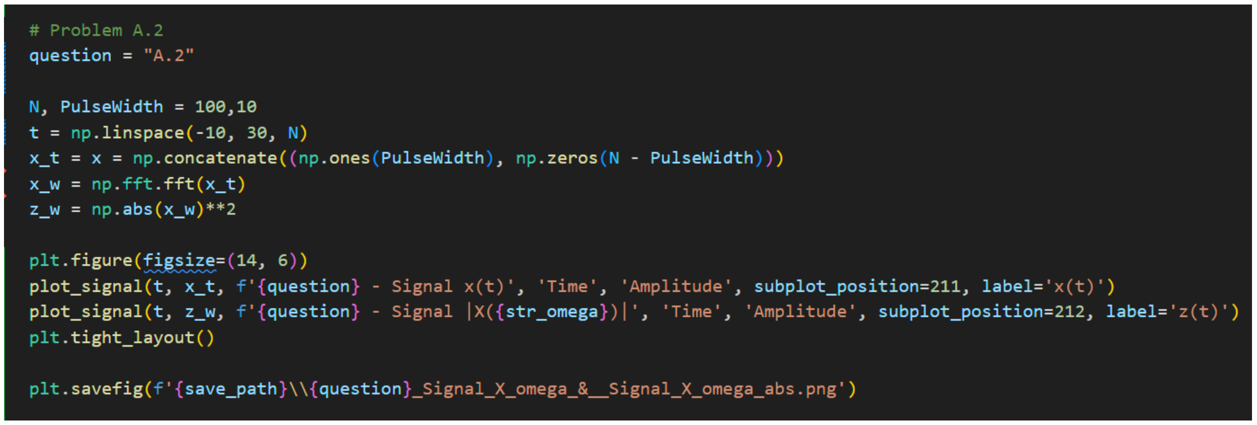
```python
# Problem A.1
question = "A.1"

x = lambda t: np.where((t >= 0) & (t < 10), 1, 0)        You, 1 second ago • Uncommitted changes
t = np.arange(-10, 30, 0.001)
x_t = x(t)
z_t, t_conv = perform_convolution(x_t, t)

plt.figure(figsize=(14, 6))
plot_signal(t, x_t, f'{question} - Original Signal x(t)', 'Time', 'Amplitude', subplot_position=211, label='x(t)')
plot_signal(t_conv, z_t, f'{question} - Convolution of x(t) with itself', 'Time', 'Amplitude', subplot_position=212, label='z(t)')
plt.tight_layout()

# Save the figures
plt.savefig(f'{save_path}\\{question}_Original_Signal_x_t.png_&_Convolution_x_t_with_itself.png')
```

  Results:



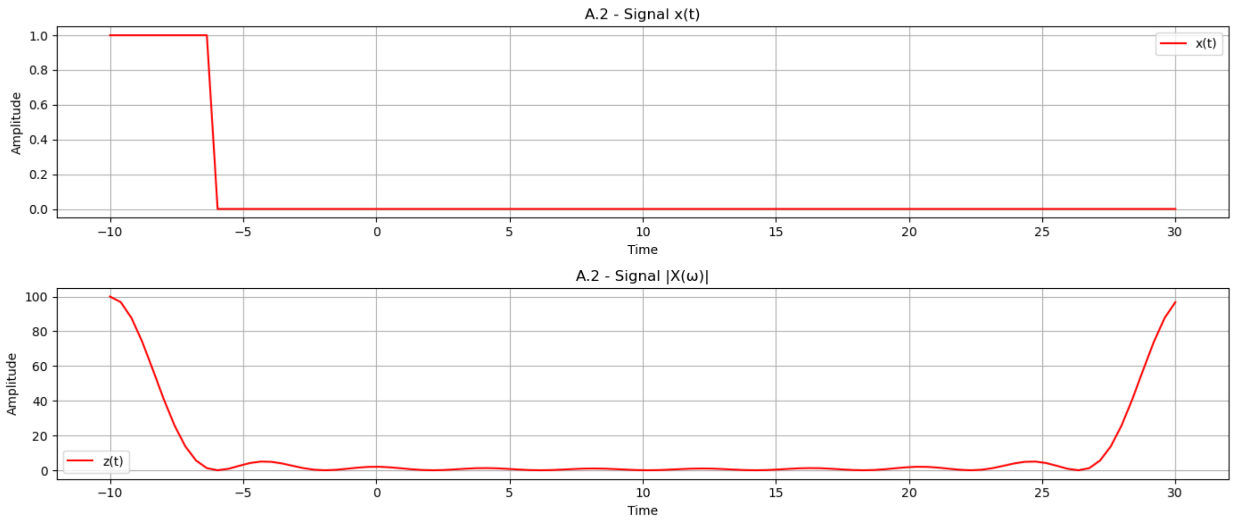
- **Problem A.2**

  Code:



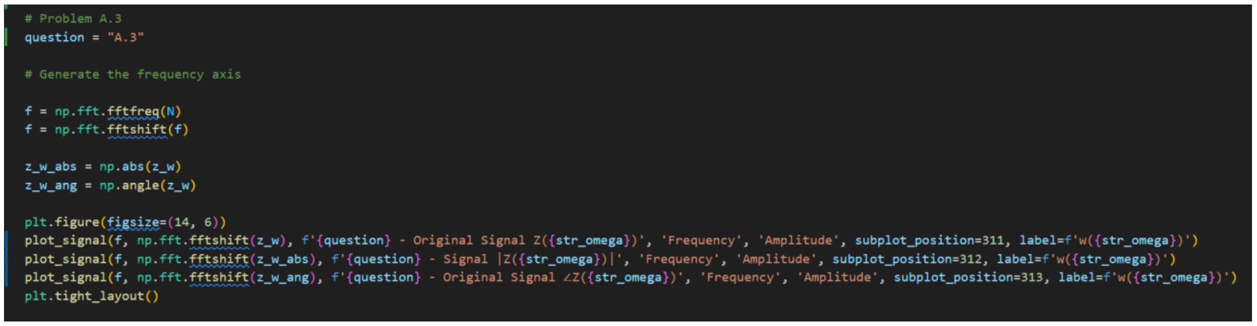
```
# Problem A.2
question = "A.2"

N, PulseWidth = 100,10
t = np.linspace(-10, 30, N)
x_t = x = np.concatenate((np.ones(PulseWidth), np.zeros(N - PulseWidth)))
x_w = np.fft.fft(x_t)
z_w = np.abs(x_w)**2

plt.figure(figsize=(14, 6))
plot_signal(t, x_t, f'{question} - Signal x(t)', 'Time', 'Amplitude', subplot_position=211, label='x(t)')
plot_signal(t, z_w, f'{question} - Signal |X({str_omega})|', 'Time', 'Amplitude', subplot_position=212, label='z(t)')
plt.tight_layout()

plt.savefig(f'{save_path}\\{question}_Signal_X_omega_&__Signal_X_omega_abs.png')
```

  Results:



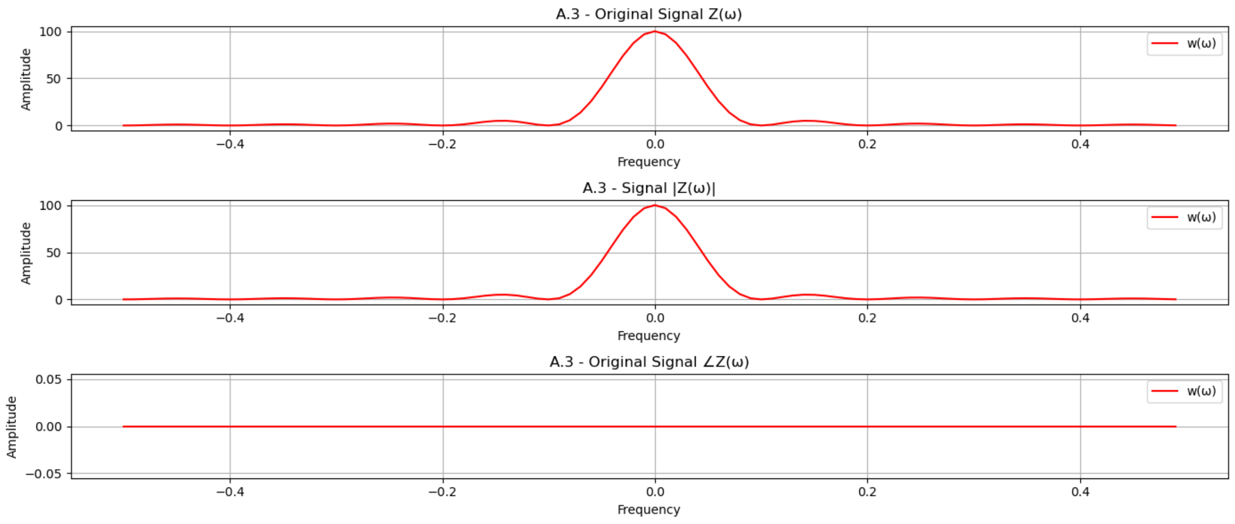
- **Problem A.3**

Code:



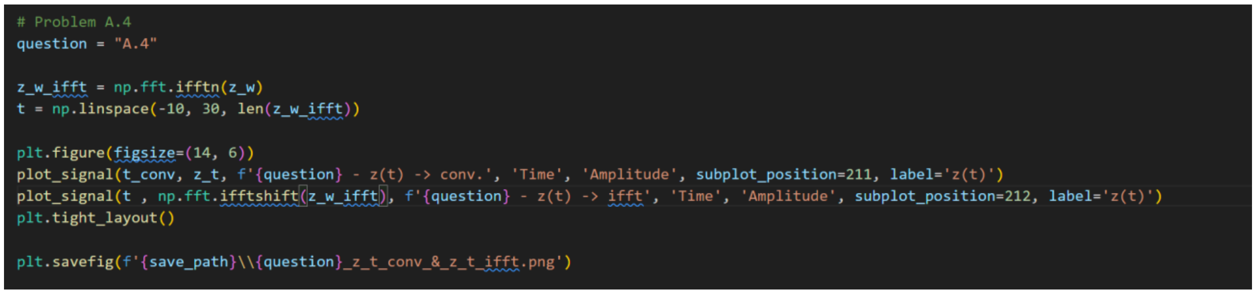
```
# Problem A.3
question = "A.3"

# Generate the frequency axis

f = np.fft.fftfreq(N)
f = np.fft.fftshift(f)

z_w_abs = np.abs(z_w)
z_w_ang = np.angle(z_w)

plt.figure(figsize=(14, 6))
plot_signal(f, np.fft.fftshift(z_w), f'{question} - Original Signal Z({str_omega})', 'Frequency', 'Amplitude', subplot_position=311, label=f'w({str_omega})')
plot_signal(f, np.fft.fftshift(z_w_abs), f'{question} - Signal |Z({str_omega})|', 'Frequency', 'Amplitude', subplot_position=312, label=f'w({str_omega})')
plot_signal(f, np.fft.fftshift(z_w_ang), f'{question} - Original Signal ∠Z({str_omega})', 'Frequency', 'Amplitude', subplot_position=313, label=f'w({str_omega})')
plt.tight_layout()
```

Results:



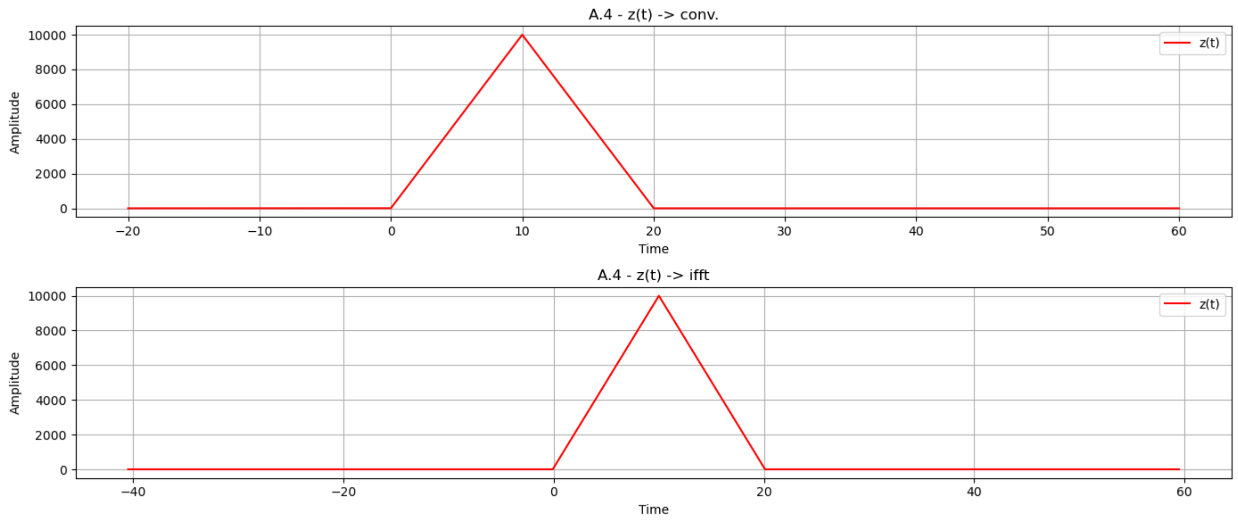
- **Problem A.4**

Code:



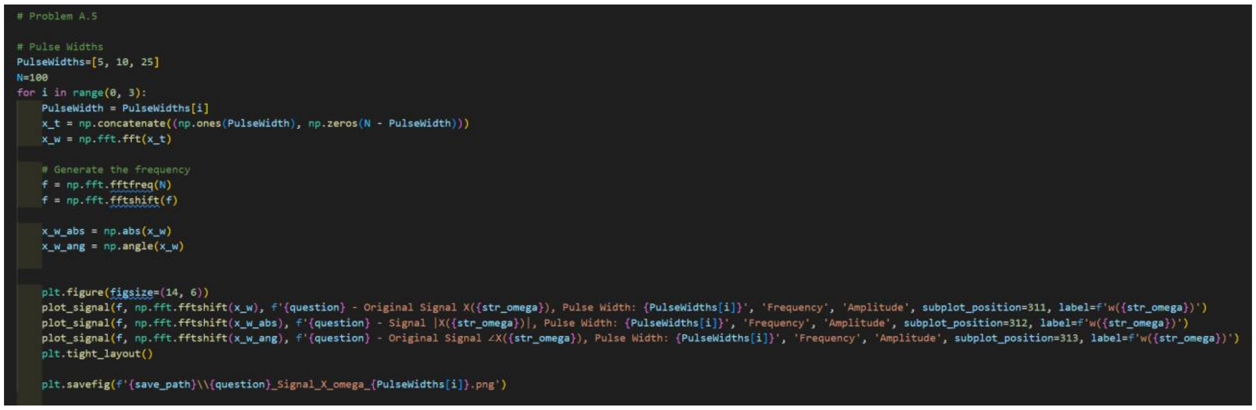
```
# Problem A.4
question = "A.4"

z_w_ifft = np.fft.ifftn(z_w)
t = np.linspace(-10, 30, len(z_w_ifft))

plt.figure(figsize=(14, 6))
plot_signal(t_conv, z_t, f'{question} - z(t) -> conv.', 'Time', 'Amplitude', subplot_position=211, label='z(t)')
plot_signal(t , np.fft.ifftshift(z_w_ifft), f'{question} - z(t) -> ifft', 'Time', 'Amplitude', subplot_position=212, label='z(t)')
plt.tight_layout()

plt.savefig(f'{save_path}\\{question}_z_t_conv_&_z_t_ifft.png')
```

Results:



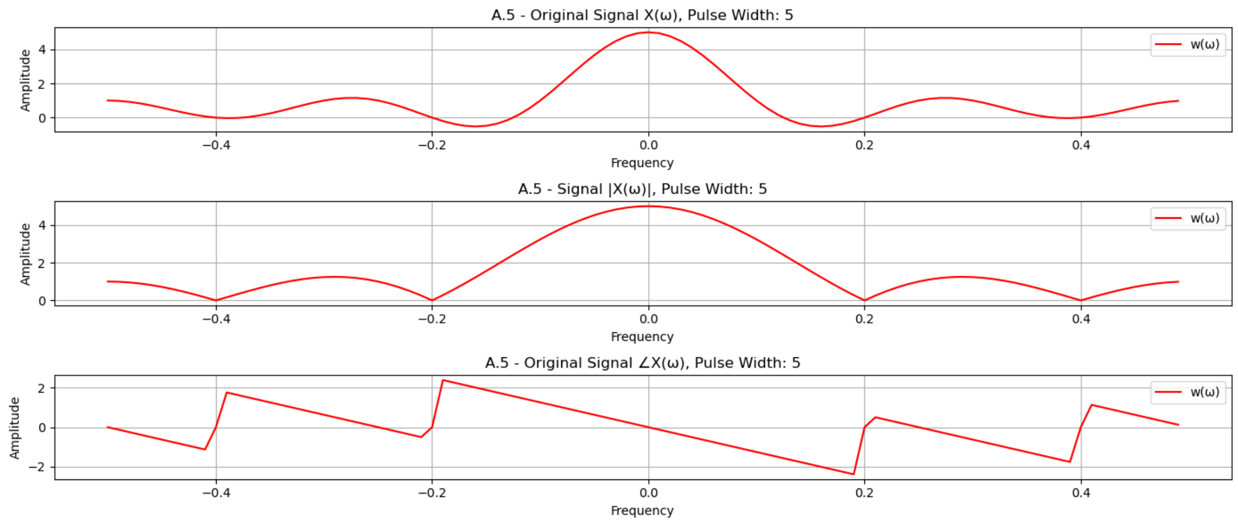
- **Problem A.5**

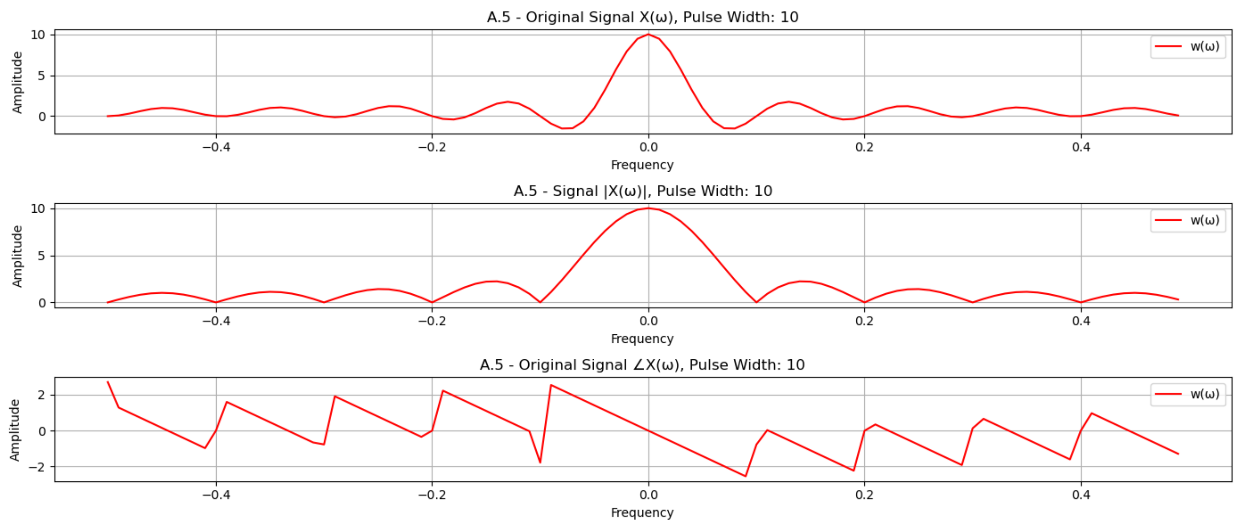
Code:

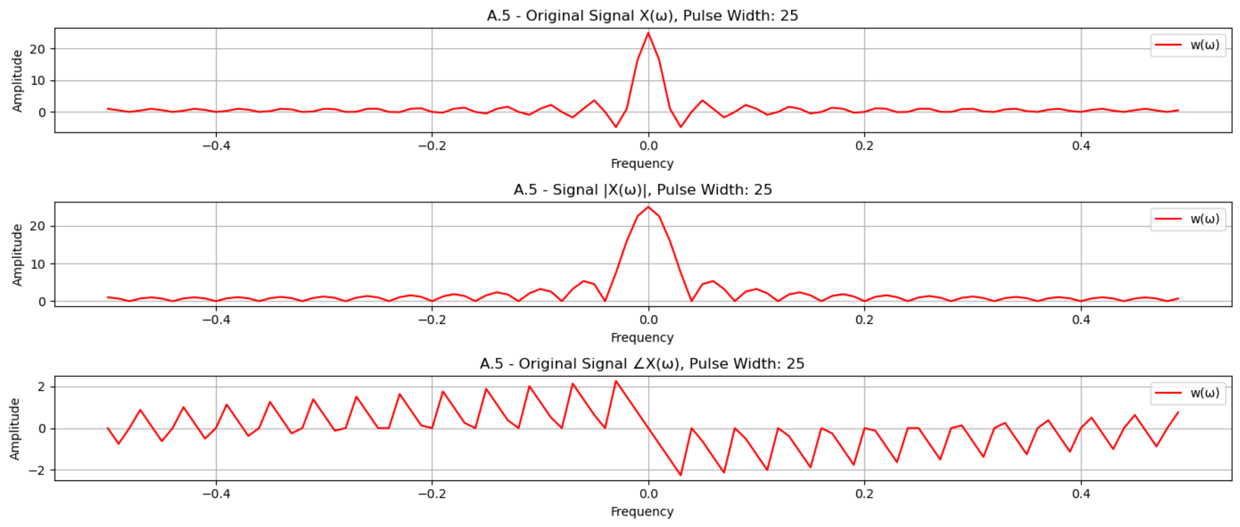


```
# Problem A.5

# Pulse Widths
PulseWidths=[5, 10, 25]
N=100
for i in range(0, 3):
    PulseWidth = PulseWidths[i]
    x_t = np.concatenate((np.ones(PulseWidth), np.zeros(N - PulseWidth)))
    x_w = np.fft.fft(x_t)

    # Generate the frequency
    f = np.fft.fftfreq(N)
    f = np.fft.fftshift(f)

    x_w_abs = np.abs(x_w)
    x_w_ang = np.angle(x_w)


    plt.figure(figsize=(14, 6))
    plot_signal(f, np.fft.fftshift(x_w), f'{question} - Original Signal X({str_omega}), Pulse Width: {PulseWidths[i]}', 'Frequency', 'Amplitude', subplot_position=311, label=f'w({str_omega})')
    plot_signal(f, np.fft.fftshift(x_w_abs), f'{question} - Signal |X({str_omega})|, Pulse Width: {PulseWidths[i]}', 'Frequency', 'Amplitude', subplot_position=312, label=f'w({str_omega})')
    plot_signal(f, np.fft.fftshift(x_w_ang), f'{question} - Original Signal ∠X({str_omega}), Pulse Width: {PulseWidths[i]}', 'Frequency', 'Amplitude', subplot_position=313, label=f'w({str_omega})')
    plt.tight_layout()

    plt.savefig(f'{save_path}\\{question}_Signal_X_omega_{PulseWidths[i]}.png')
```

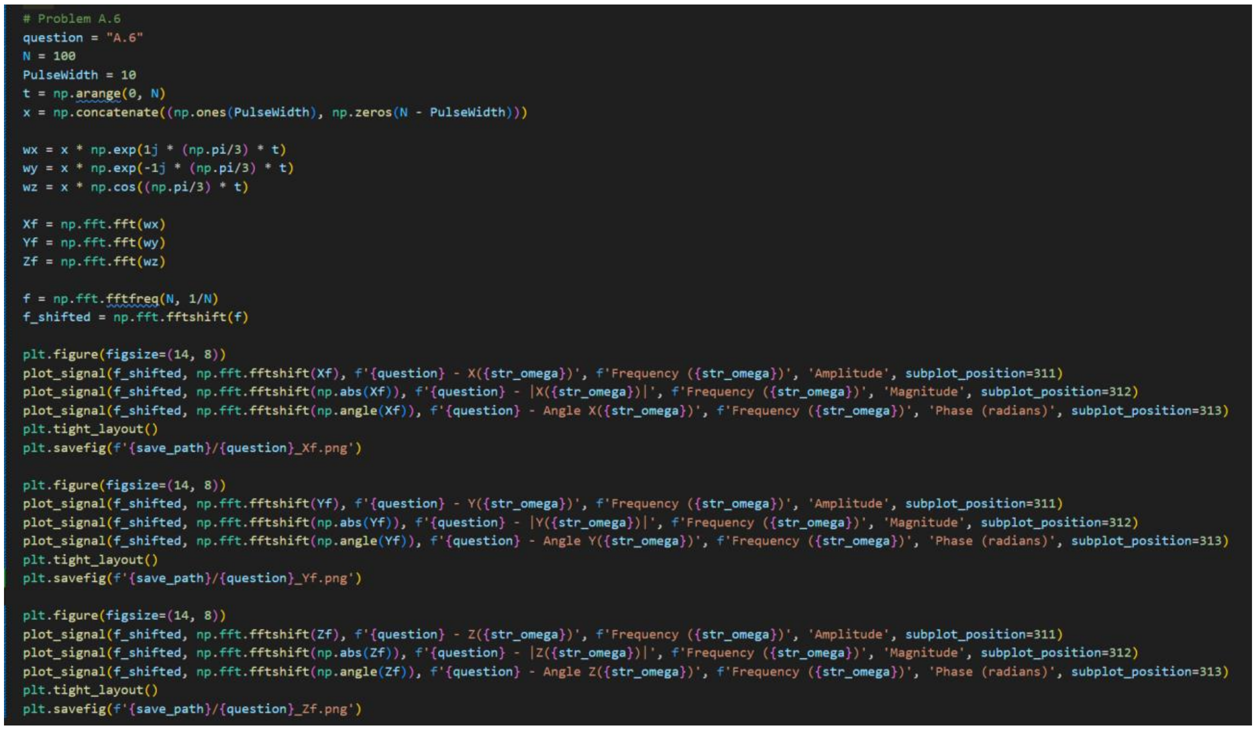
Results:







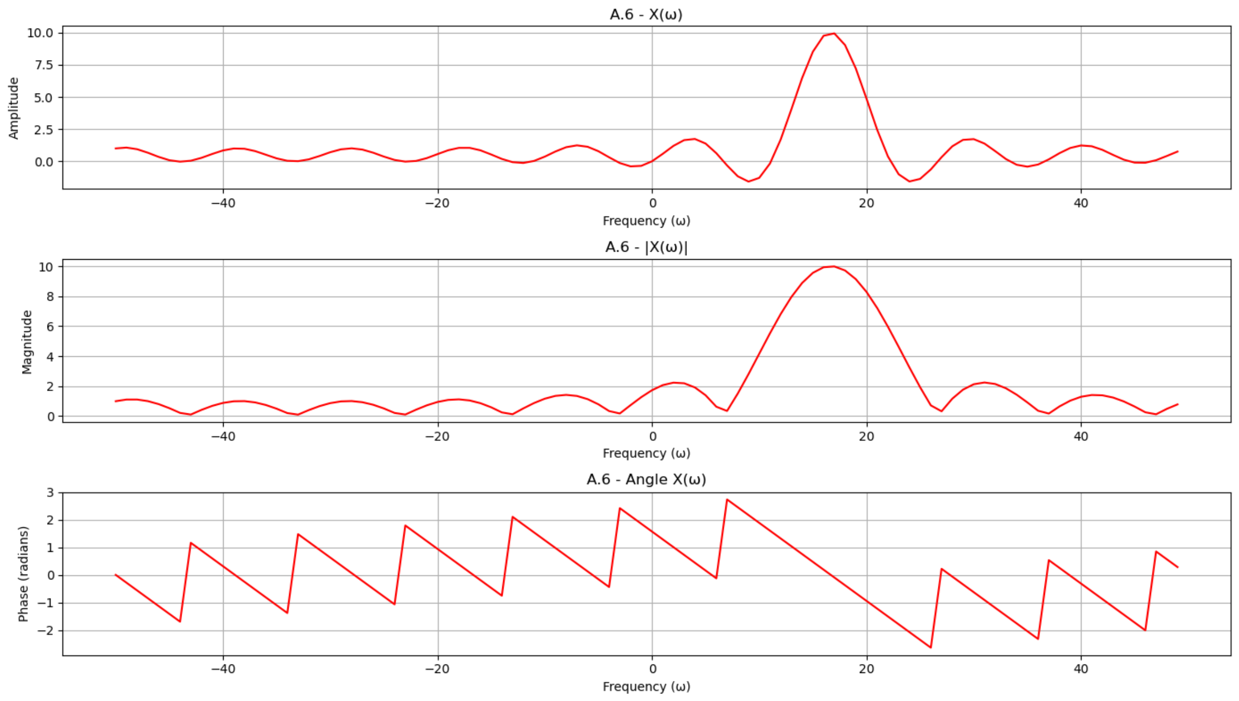
- **Problem A.6**
  Code:



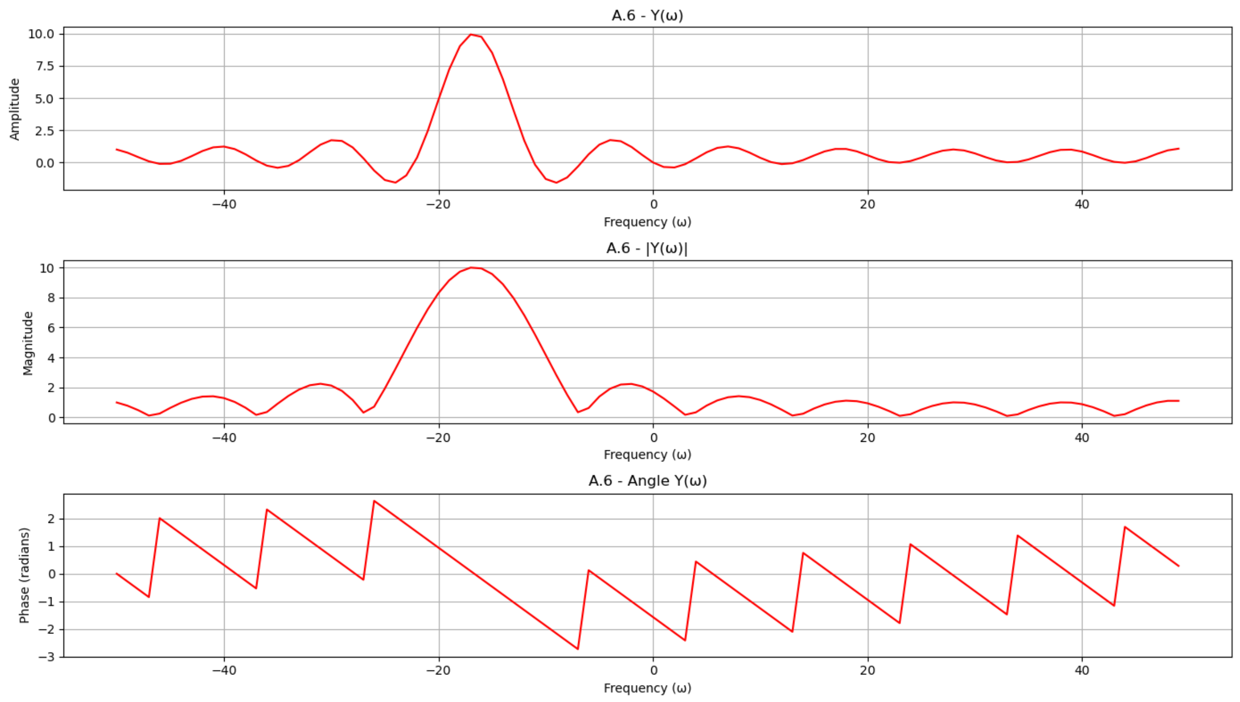
```
# Problem A.6
question = "A.6"
N = 100
PulseWidth = 10
t = np.arange(0, N)
x = np.concatenate((np.ones(PulseWidth), np.zeros(N - PulseWidth)))

wx = x * np.exp(1j * (np.pi/3) * t)
wy = x * np.exp(-1j * (np.pi/3) * t)
wz = x * np.cos((np.pi/3) * t)

Xf = np.fft.fft(wx)
Yf = np.fft.fft(wy)
Zf = np.fft.fft(wz)

f = np.fft.fftfreq(N, 1/N)
f_shifted = np.fft.fftshift(f)

plt.figure(figsize=(14, 8))
plot_signal(f_shifted, np.fft.fftshift(Xf), f'{question} - X({str_omega})', f'Frequency ({str_omega})', 'Amplitude', subplot_position=311)
plot_signal(f_shifted, np.fft.fftshift(np.abs(Xf)), f'{question} - |X({str_omega})|', f'Frequency ({str_omega})', 'Magnitude', subplot_position=312)
plot_signal(f_shifted, np.fft.fftshift(np.angle(Xf)), f'{question} - Angle X({str_omega})', f'Frequency ({str_omega})', 'Phase (radians)', subplot_position=313)
plt.tight_layout()
plt.savefig(f'{save_path}/{question}_Xf.png')

plt.figure(figsize=(14, 8))
plot_signal(f_shifted, np.fft.fftshift(Yf), f'{question} - Y({str_omega})', f'Frequency ({str_omega})', 'Amplitude', subplot_position=311)
plot_signal(f_shifted, np.fft.fftshift(np.abs(Yf)), f'{question} - |Y({str_omega})|', f'Frequency ({str_omega})', 'Magnitude', subplot_position=312)
plot_signal(f_shifted, np.fft.fftshift(np.angle(Yf)), f'{question} - Angle Y({str_omega})', f'Frequency ({str_omega})', 'Phase (radians)', subplot_position=313)
plt.tight_layout()
plt.savefig(f'{save_path}/{question}_Yf.png')

plt.figure(figsize=(14, 8))
plot_signal(f_shifted, np.fft.fftshift(Zf), f'{question} - Z({str_omega})', f'Frequency ({str_omega})', 'Amplitude', subplot_position=311)
plot_signal(f_shifted, np.fft.fftshift(np.abs(Zf)), f'{question} - |Z({str_omega})|', f'Frequency ({str_omega})', 'Magnitude', subplot_position=312)
plot_signal(f_shifted, np.fft.fftshift(np.angle(Zf)), f'{question} - Angle Z({str_omega})', f'Frequency ({str_omega})', 'Phase (radians)', subplot_position=313)
plt.tight_layout()
plt.savefig(f'{save_path}/{question}_Zf.png')
```

Results:

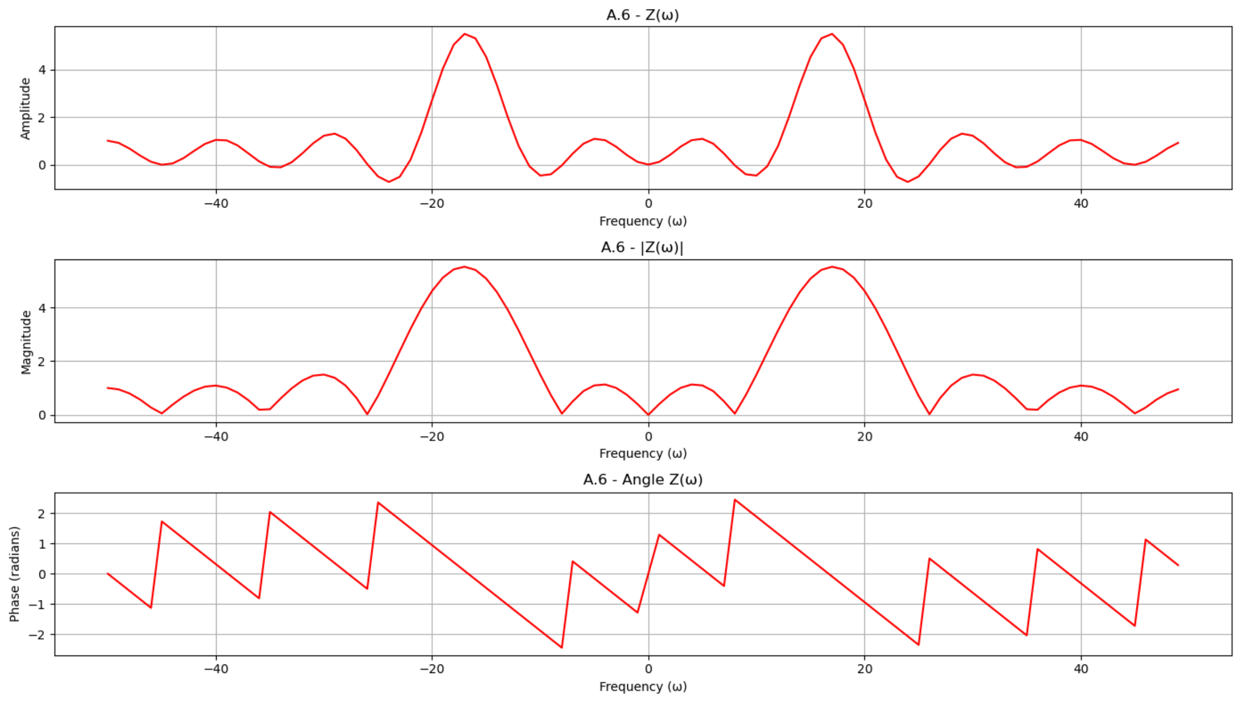
Xf=



Yf=


A.6 - Y(ω)
Amplitude
Frequency (ω)
A.6 - |Y(ω)|
Magnitude
Frequency (ω)
A.6 - Angle Y(ω)
Phase (radians)
Frequency (ω)

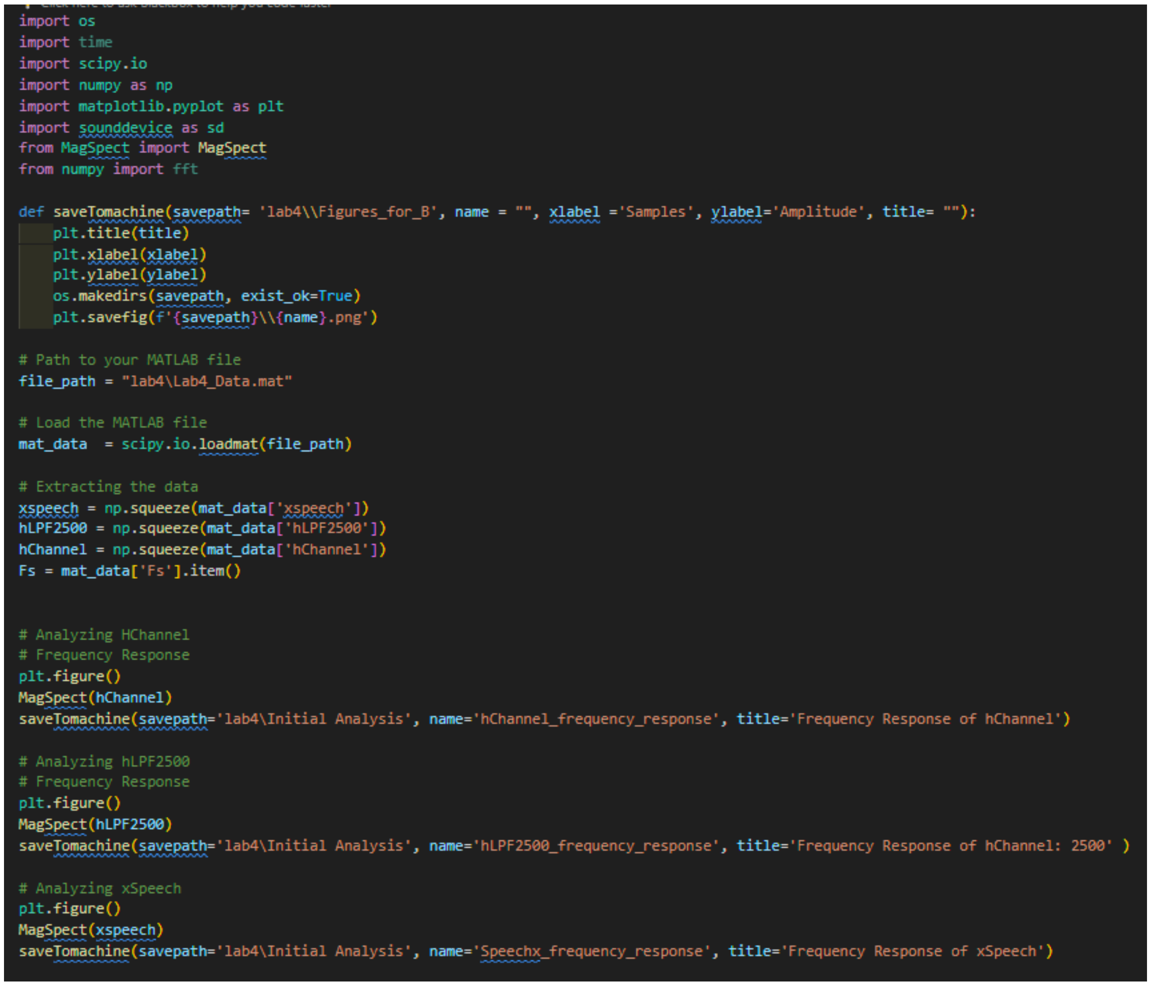
Zf=


A.6 - Z(ω)
Amplitude
Frequency (ω)
A.6 - |Z(ω)|
Magnitude
Frequency (ω)
A.6 - Angle Z(ω)
Phase (radians)
Frequency (ω)

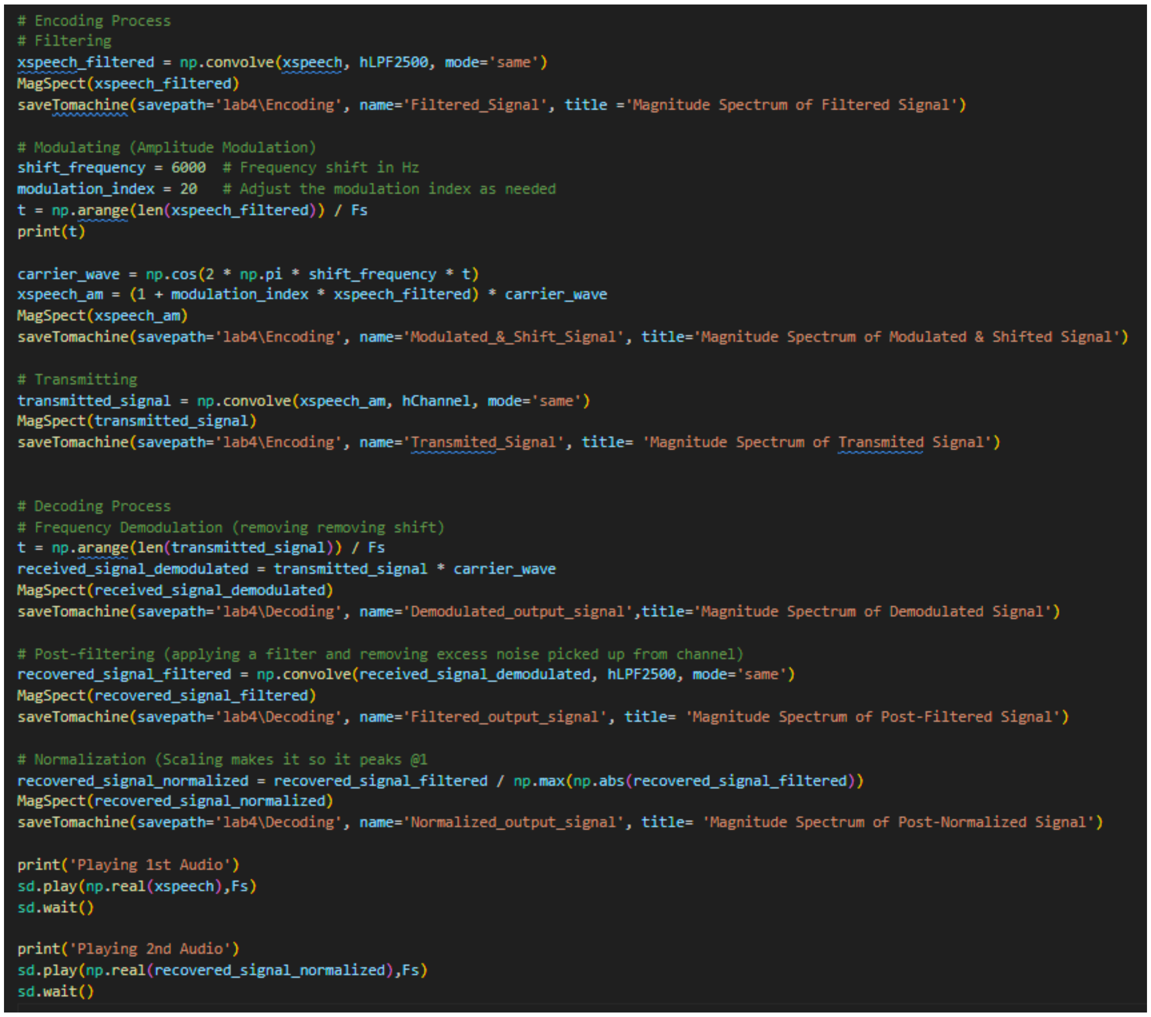
Part B. Application of the Fourier Transform

- **Problem B.1**

  Code:

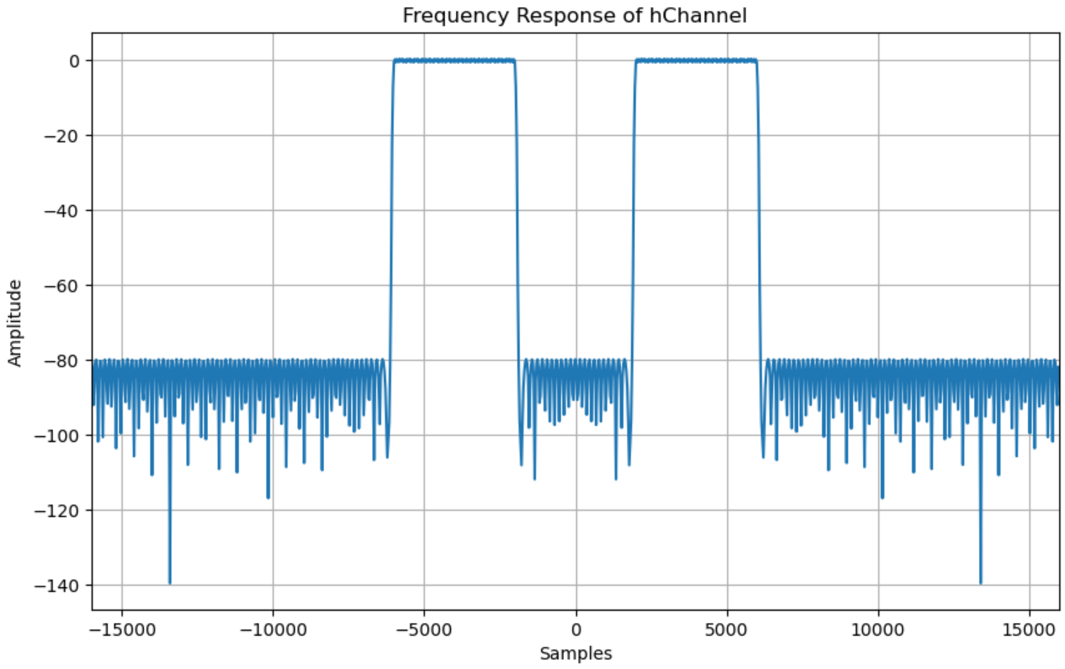


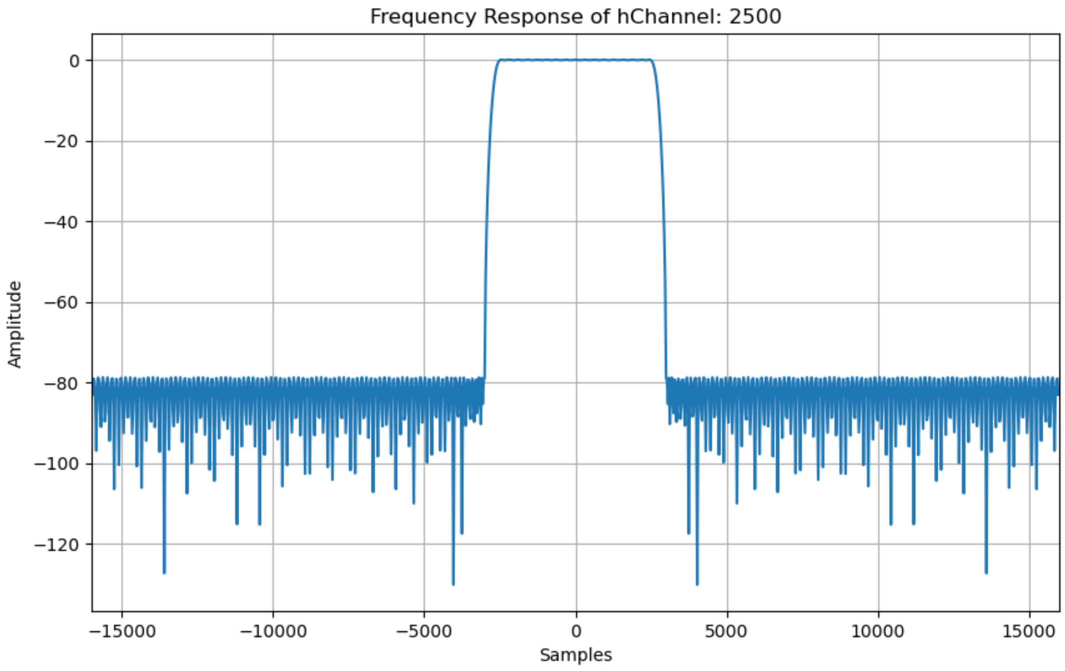
```
import os
import time
import scipy.io
import numpy as np
import matplotlib.pyplot as plt
import sounddevice as sd
from MagSpect import MagSpect
from numpy import fft

def saveTomachine(savepath= 'lab4\\Figures_for_B', name = "", xlabel ='Samples', ylabel='Amplitude', title= ""):
    plt.title(title)
    plt.xlabel(xlabel)
    plt.ylabel(ylabel)
    os.makedirs(savepath, exist_ok=True)
    plt.savefig(f'{savepath}\\{name}.png')

# Path to your MATLAB file
file_path = "lab4\Lab4_Data.mat"

# Load the MATLAB file
mat_data  = scipy.io.loadmat(file_path)

# Extracting the data
xspeech = np.squeeze(mat_data['xspeech'])
hLPF2500 = np.squeeze(mat_data['hLPF2500'])
hChannel = np.squeeze(mat_data['hChannel'])
Fs = mat_data['Fs'].item()


# Analyzing HChannel
# Frequency Response
plt.figure()
MagSpect(hChannel)
saveTomachine(savepath='lab4\Initial Analysis', name='hChannel_frequency_response', title='Frequency Response of hChannel')

# Analyzing hLPF2500
# Frequency Response
plt.figure()
MagSpect(hLPF2500)
saveTomachine(savepath='lab4\Initial Analysis', name='hLPF2500_frequency_response', title='Frequency Response of hChannel: 2500' )

# Analyzing xSpeech
plt.figure()
MagSpect(xspeech)
saveTomachine(savepath='lab4\Initial Analysis', name='Speechx_frequency_response', title='Frequency Response of xSpeech')
```

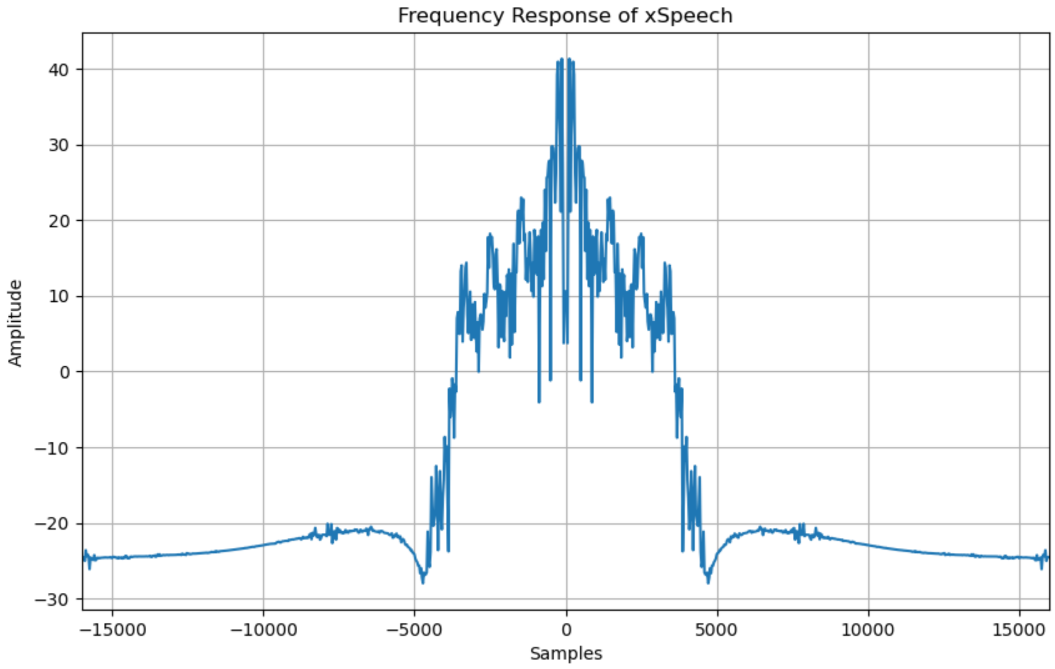

```python
# Encoding Process
# Filtering
xspeech_filtered = np.convolve(xspeech, hLPF2500, mode='same')
MagSpect(xspeech_filtered)
saveTomachine(savepath='lab4\Encoding', name='Filtered_Signal', title ='Magnitude Spectrum of Filtered Signal')

# Modulating (Amplitude Modulation)
shift_frequency = 6000  # Frequency shift in Hz
modulation_index = 20   # Adjust the modulation index as needed
t = np.arange(len(xspeech_filtered)) / Fs
print(t)

carrier_wave = np.cos(2 * np.pi * shift_frequency * t)
xspeech_am = (1 + modulation_index * xspeech_filtered) * carrier_wave
MagSpect(xspeech_am)
saveTomachine(savepath='lab4\Encoding', name='Modulated_&_Shift_Signal', title='Magnitude Spectrum of Modulated & Shifted Signal')

# Transmitting
transmitted_signal = np.convolve(xspeech_am, hChannel, mode='same')
MagSpect(transmitted_signal)
saveTomachine(savepath='lab4\Encoding', name='Transmited_Signal', title= 'Magnitude Spectrum of Transmited Signal')


# Decoding Process
# Frequency Demodulation (removing removing shift)
t = np.arange(len(transmitted_signal)) / Fs
received_signal_demodulated = transmitted_signal * carrier_wave
MagSpect(received_signal_demodulated)
saveTomachine(savepath='lab4\Decoding', name='Demodulated_output_signal',title='Magnitude Spectrum of Demodulated Signal')

# Post-filtering (applying a filter and removing excess noise picked up from channel)
recovered_signal_filtered = np.convolve(received_signal_demodulated, hLPF2500, mode='same')
MagSpect(recovered_signal_filtered)
saveTomachine(savepath='lab4\Decoding', name='Filtered_output_signal', title= 'Magnitude Spectrum of Post-Filtered Signal')

# Normalization (Scaling makes it so it peaks @1
recovered_signal_normalized = recovered_signal_filtered / np.max(np.abs(recovered_signal_filtered))
MagSpect(recovered_signal_normalized)
saveTomachine(savepath='lab4\Decoding', name='Normalized_output_signal', title= 'Magnitude Spectrum of Post-Normalized Signal')

print('Playing 1st Audio')
sd.play(np.real(xspeech),Fs)
sd.wait()

print('Playing 2nd Audio')
sd.play(np.real(recovered_signal_normalized),Fs)
sd.wait()
```

- Results:

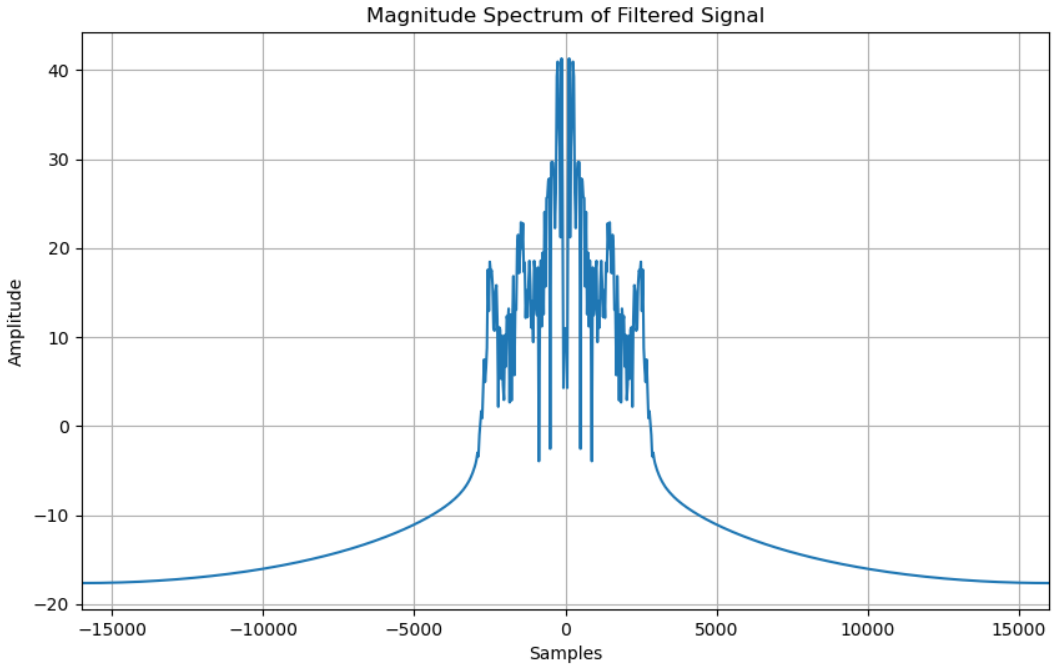
  Initial Analysis:

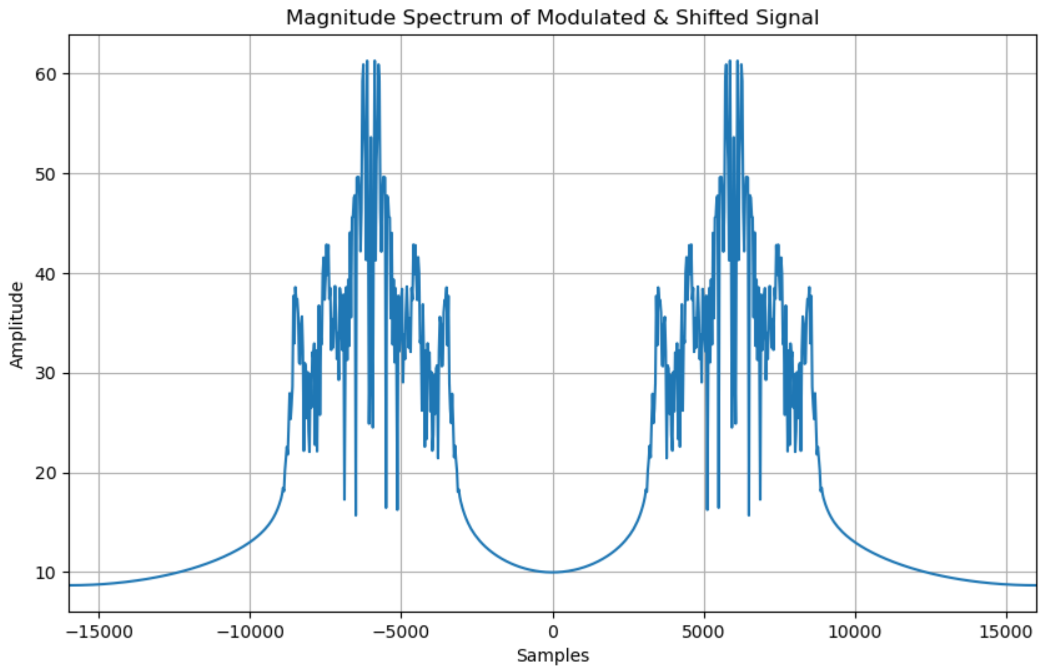

Frequency Response of hChannel
Amplitude
Samples
0
−20
−40
−60
−80
−100
−120
−140
−15000
−10000
−5000
0
5000
10000
15000

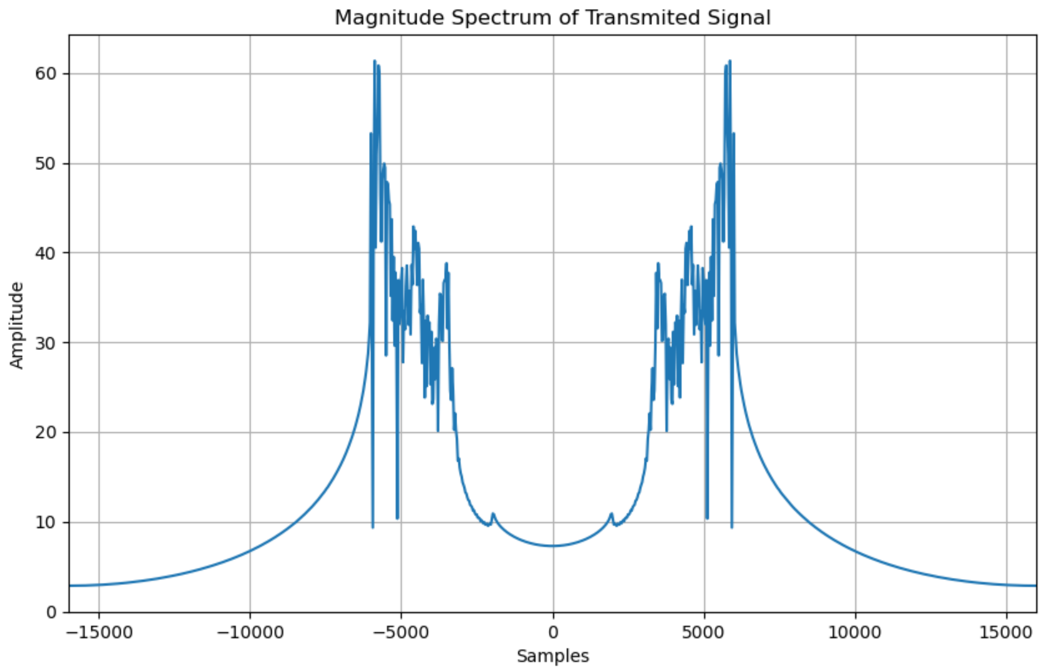

Frequency Response of hChannel: 2500
Amplitude
Samples
0
−20
−40
−60
−80
−100
−120
−15000
−10000
−5000
0
5000
10000
15000


Frequency Response of xSpeech
Amplitude
Samples
40
30
20
10
0
−10
−20
−30
−15000
−10000
−5000
0
5000
10000
15000

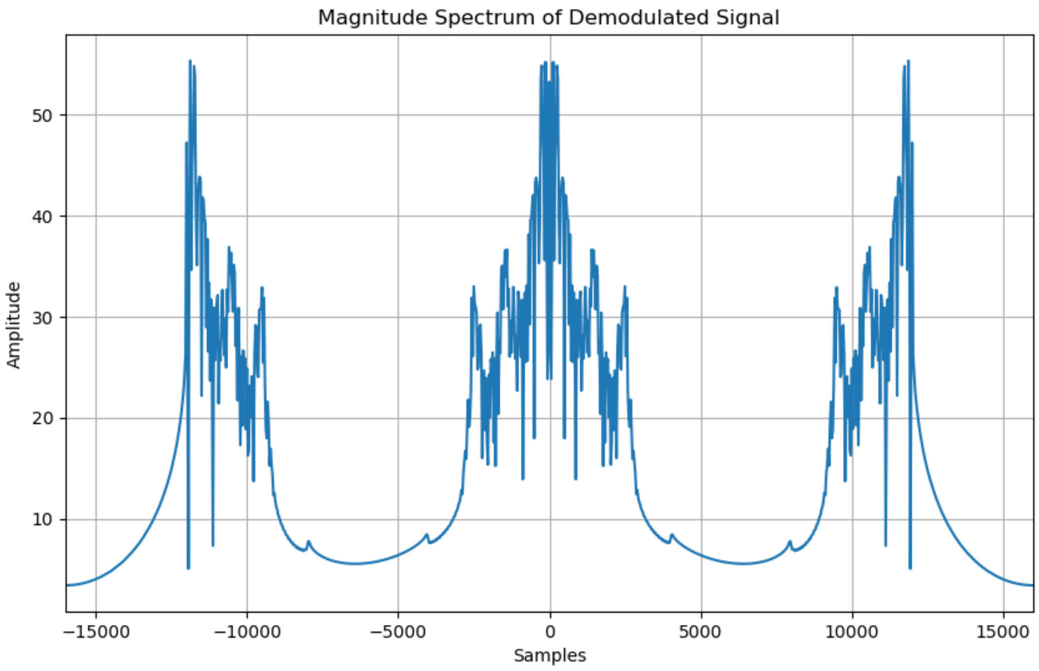
Encoding:

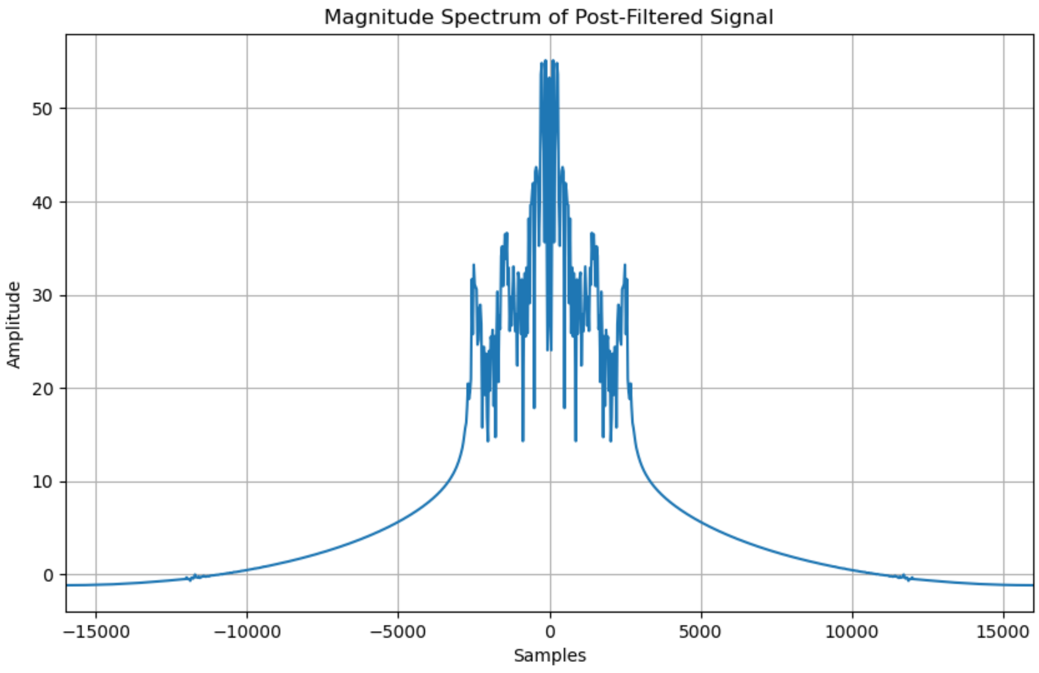

Magnitude Spectrum of Filtered Signal
Amplitude
Samples
40
30
20
10
0
−10
−20
−15000
−10000
−5000
0
5000
10000
15000

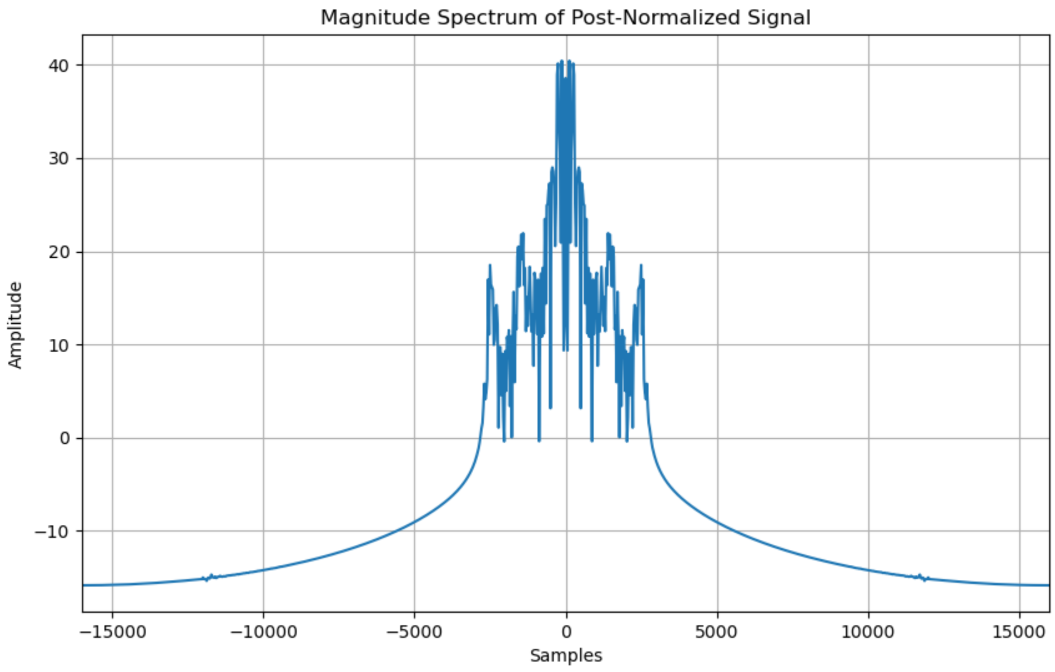

Magnitude Spectrum of Modulated & Shifted Signal
Amplitude
Samples
60
50
40
30
20
10
−15000
−10000
−5000
0
5000
10000
15000


Magnitude Spectrum of Transmited Signal
Amplitude
Samples
60
50
40
30
20
10
0
−15000
−10000
−5000
0
5000
10000
15000

Decoding:


Magnitude Spectrum of Demodulated Signal
Amplitude
Samples
−15000
−10000
−5000
0
5000
10000
15000
10
20
30
40
50


Magnitude Spectrum of Post-Filtered Signal
Amplitude
Samples
−15000
−10000
−5000
0
5000
10000
15000
0
10
20
30
40
50


Magnitude Spectrum of Post-Normalized Signal
Amplitude
Samples
40
30
20
10
0
−10
−15000
−10000
−5000
0
5000
10000
15000