Modulation and Coding Practical Assignment

Name: Injy Nashaat Mahmoud

ID: 49-6748

Communications

Code:

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

# Function to generate PCM signal based on given binary sequence
def generate_pcm_signal(binary_sequence, ts):
    pcm_signal = []
    for bit in binary_sequence:
        if bit == '0':
            pcm_signal.extend([-1]*int(ts/2))
            pcm_signal.extend([1]*int(ts/2))
        elif bit == '1':
            pcm_signal.extend([1]*int(ts/2))
            pcm_signal.extend([-1]*int(ts/2))
    return np.array(pcm_signal)

# Define the matched filter
def matched_filter(pulse_shape):
    return np.flip(pulse_shape)  # Matched filter: g(t-T), where T is the pulse duration

# Perform convolution
def perform_convolution(signal, filter):
    return np.convolve(signal, filter, 'valid')

# Apply thresholding
def apply_threshold(signal, threshold):
    return (signal > threshold).astype(int)

# Parameters for first simulation
binary_sequence_1 = '0110'
ts_1 = 100  # Total time for one bit (arbitrary units)
```

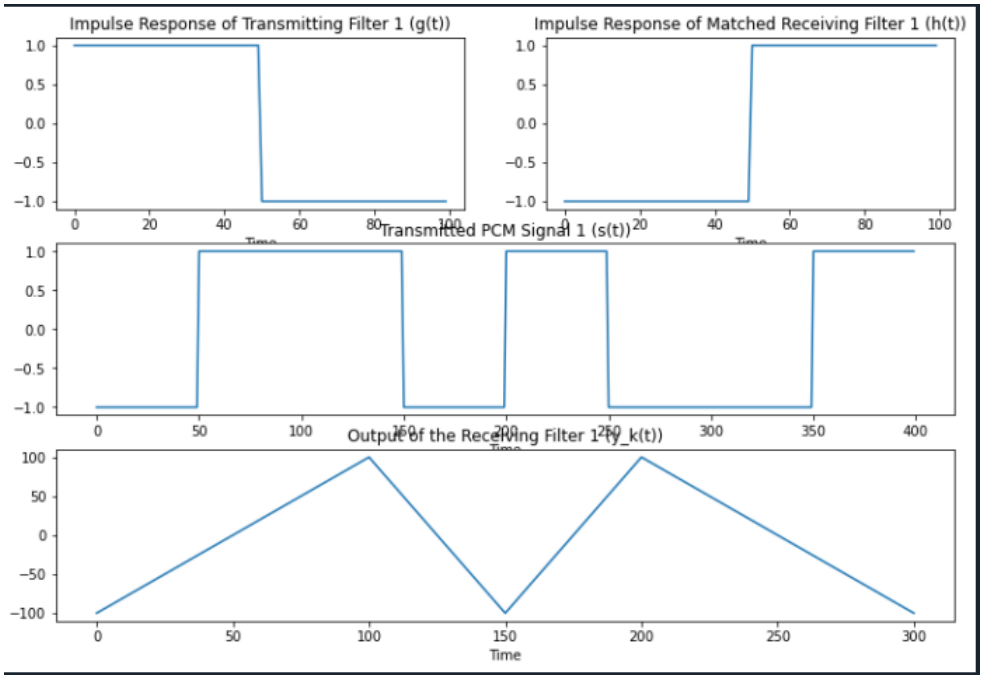
```
# Generate PCM signal for first simulation
pcm_signal_1 = generate_pcm_signal(binary_sequence_1, ts_1)

# Define the matched filter for first simulation
pulse_shape_1 = np.array([1]*int(ts_1/2) + [-1]*int(ts_1/2))  # Impulse response of transmitting filter
matched_filter_response_1 = matched_filter(pulse_shape_1)

# Perform convolution for first simulation
filtered_signal_1 = perform_convolution(pcm_signal_1, matched_filter_response_1)

Sampled_filter1=[]
for i in range(0,len(filtered_signal_1),ts_1):
    Sampled_filter1.append(filtered_signal_1[i])

# Apply thresholding for first simulation
threshold_1 = 0.5 * np.max(filtered_signal_1)
detected_signal_1 = apply_threshold(filtered_signal_1, threshold_1)

# Sample the detected signal for first simulation (at each Ts)
sampled_signal_1 = detected_signal_1[::int(ts_1/2)]

# Plot results for first simulation
plt.figure(figsize=(12, 8))

# Plot the impulse response of the transmitting filter for first simulation
plt.subplot(3, 2, 1)
plt.plot(pulse_shape_1)
plt.title('Impulse Response of Transmitting Filter 1 (g(t))')
plt.xlabel('Time')
```

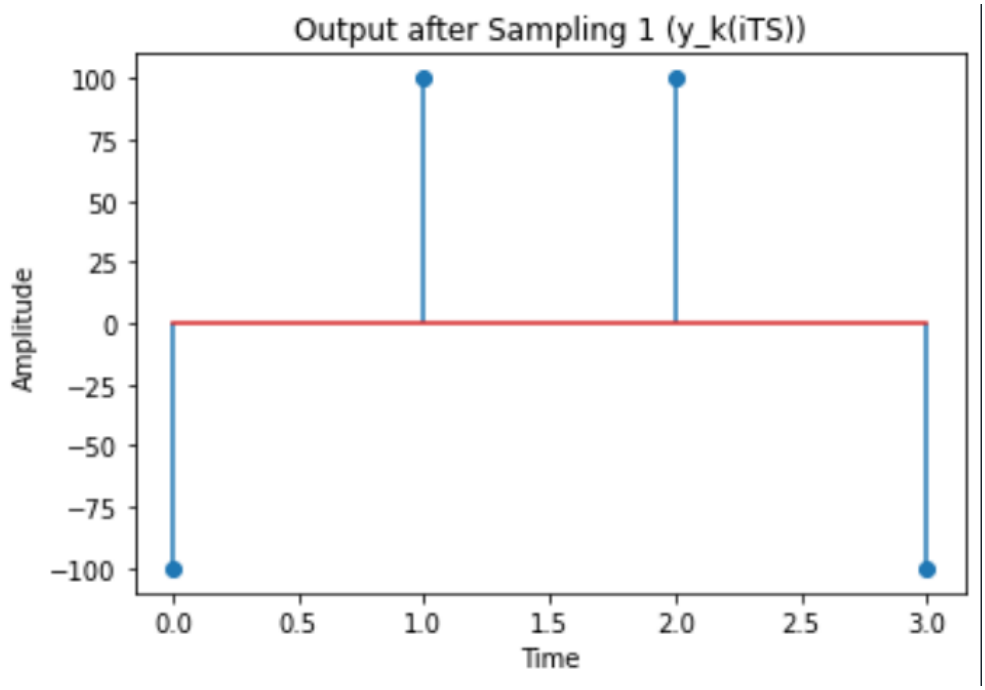
```
# Plot the impulse response of the matched receiving filter for first simulation
plt.subplot(3, 2, 2)
plt.plot(matched_filter_response_1)
plt.title('Impulse Response of Matched Receiving Filter 1 (h(t))')
plt.xlabel('Time')

# Plot the received signal for first simulation
plt.subplot(3, 1, 2)
plt.plot(pcm_signal_1)
plt.title('Transmitted PCM Signal 1 (s(t))')
plt.xlabel('Time')

# Plot the filtered signal for first simulation
plt.subplot(3, 1, 3)
plt.plot(filtered_signal_1)
plt.title('Output of the Receiving Filter 1 (y_k(t))')
plt.xlabel('Time')


plt.figure()
plt.stem( Sampled_filter1)
plt.title('Output after Sampling 1 (y_k(iTS))')
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.show()


# Parameters for second simulation
binary_sequence_2 = '1001'
ts_2 = 200  # Total time for one bit (arbitrary units)
```

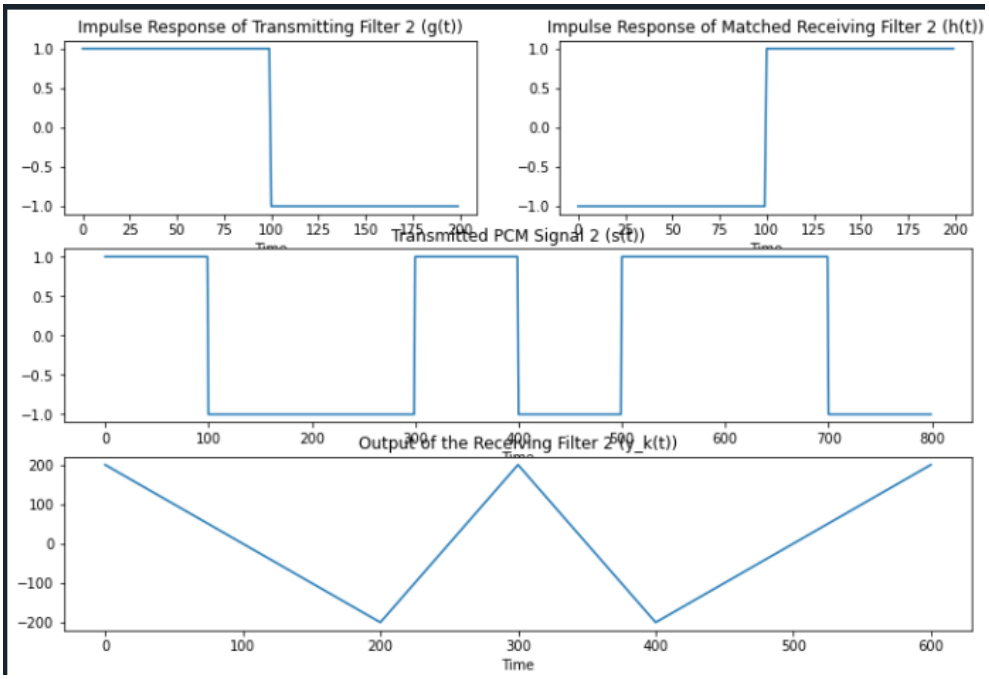
```
# Generate PCM signal for second simulation
pcm_signal_2 = generate_pcm_signal(binary_sequence_2, ts_2)

# Define the matched filter for second simulation
pulse_shape_2 = np.array([1]*int(ts_2/2) + [-1]*int(ts_2/2))  # Impulse response of transmitting filter
matched_filter_response_2 = matched_filter(pulse_shape_2)

# Perform convolution for second simulation
filtered_signal_2 = perform_convolution(pcm_signal_2, matched_filter_response_2)

Sampled_filter2=[]
for i in range(0,len(filtered_signal_2),ts_2):
    Sampled_filter2.append(filtered_signal_2[i])

# Apply thresholding for second simulation
threshold_2 = 0.5 * np.max(filtered_signal_2)
detected_signal_2 = apply_threshold(filtered_signal_2, threshold_2)

# Sample the detected signal for second simulation (at each Ts)
sampled_signal_2 = detected_signal_2[::int(ts_2/2)]

# Plot results for second simulation
plt.figure(figsize=(12, 8))

# Plot the impulse response of the transmitting filter for second simulation
plt.subplot(3, 2, 1)
plt.plot(pulse_shape_2)
plt.title('Impulse Response of Transmitting Filter 2 (g(t))')
plt.xlabel('Time')

# Plot the impulse response of the matched receiving filter for second simulation
```

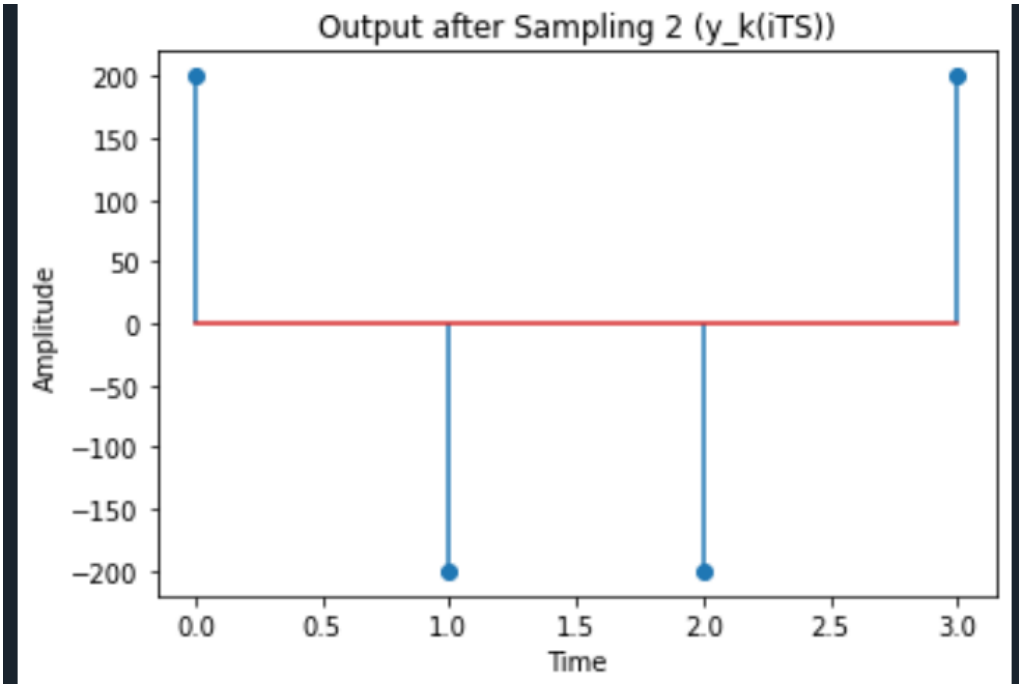
```
plt.subplot(3, 2, 2)
plt.plot(matched_filter_response_2)
plt.title('Impulse Response of Matched Receiving Filter 2 (h(t))')
plt.xlabel('Time')

# Plot the received signal for second simulation
plt.subplot(3, 1, 2)
plt.plot(pcm_signal_2)
plt.title('Transmitted PCM Signal 2 (s(t))')
plt.xlabel('Time')

# Plot the filtered signal for second simulation
plt.subplot(3, 1, 3)
plt.plot(filtered_signal_2)
plt.title('Output of the Receiving Filter 2 (y_k(t))')
plt.xlabel('Time')

plt.figure()
plt.stem( Sampled_filter2)
plt.title('Output after Sampling 2 (y_k(iTS))')
plt.xlabel('Time')
plt.ylabel('Amplitude')

plt.show()
```

Plots:


Impulse Response of Transmitting Filter 1 (g(t))
Impulse Response of Matched Receiving Filter 1 (h(t))
Transmitted PCM Signal 1 (s(t))
Output of the Receiving Filter 1 (y_k(t))
Time


Output after Sampling 1 (y_k(iTS))
Amplitude
Time





Challenges Faced:

- One challenge might be ensuring the correct alignment of signals during convolution and sampling, especially when dealing with different time durations (ts) for different simulations.

- Another challenge could be determining an appropriate threshold for thresholding the filtered signal to accurately detect the transmitted symbols.

Summary:

- The code simulates two scenarios of transmitting binary sequences using Pulse Code Modulation (PCM). It generates PCM signals, defines matched filters, performs convolution, applies thresholding, and samples the detected signals. The simulations are then plotted to visualize various components of the communication system, including the transmitted signal, the impulse response of filters, the filtered signal, and the sampled output.

- Through this implementation, one can study the impact of different parameters such as the binary sequence, the duration of each bit (ts), and the choice of matched filters on the communication system's performance, including aspects like noise resilience and detection accuracy.