Name : Sahil Yadav
Roll No. : 21ECE1039
Course : Digital Communication

Experiment No. -3

Assignment 3 - Simulation of PCM-TDM

Aim :-
Transmission of Pulse Code Modulation (PCM) output waveform using Time Division Multiplexing (TDM) with and without additive noise.

Tools Used :-
Anaconda's Spyder Python Simulation Software.

Theory :-
Time-Division Multiplexing
Time-division multiplexing is the interleaving of sampled data from different sources. Each signal occupies a discrete time slot while occupying the total frequency spectrum. Time-division multiplexing can be performed on both analog and digital signals.
The interleaving of analog signals can be visualized by a rotating commutator that samples the bandlimited signal. All three input signals are drawn exactly the same, allowing for easier visualization of the interleaved sampled pulses in (b). The sampled signals are then digitized (c) and transmitted as a digital PCM/TDM signal. Also, the PCM/TDM signal can be interleaved 4 with other digital data (d). The widespread Bell System's T1 carrier handles 24 multiplexed voice channels and transmits at 1.544 Mbps (24 channels x 64 kbps plus 8 kbps for synchronization). TDM requires synchronization in order to extract the desired signal at the proper time slot

Code :

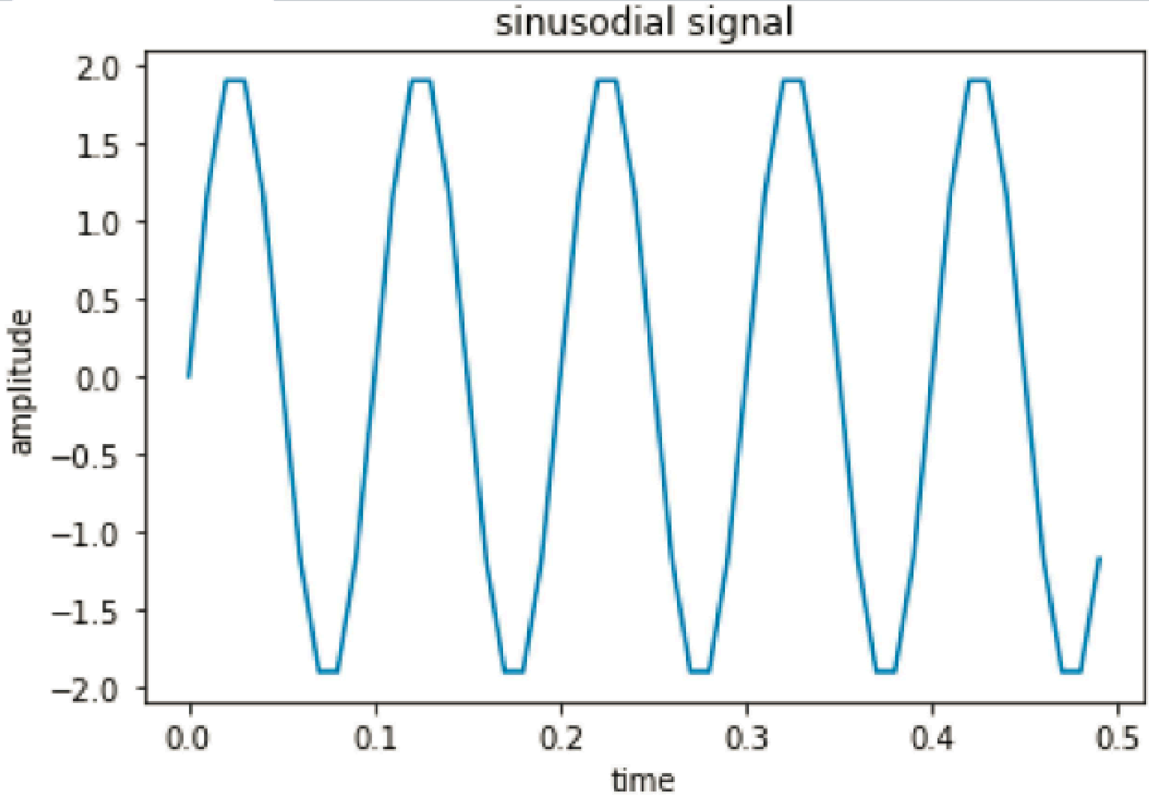
```
import numpy as np
import matplotlib.pyplot  as plt

a=2              #amplitude
frequency=10
f_s=100   #sampling Frequency
dur=5/frequency

t = np.arange(0, dur,1/f_s)

#signal
sin_signal=a*np.sin(2*np.pi*frequency*t)


plt.title("sinusodial signal")
plt.plot(t,sin_signal)
plt.xlabel("time")
plt.ylabel("amplitude")
frequency = 10  # Frequency of the triangular wave in Hz
amplitude = 2   # Amplitude of the triangular wave
cycles = 5      # Number of cycles to plot
sampling_frequency = 100

duration = cycles / frequency

t = np.arange(0, duration,1/sampling_frequency)

# Generate the triangular wave
triangle_wave = amplitude * np.abs(2 * (t * frequency - np.floor(t * frequency +
0.5)))-amplitude/2



# Plot the triangular wave
plt.plot(t, triangle_wave)
plt.title('Triangular Waveform (Amplitude = 2, Frequency = 10 Hz)')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.grid(True)
plt.show()
# Quantization levels
quantization_levels = np.array([-1, 1])  # Bipolar levels

# Uniform PCM Encoding
def uniform_pcm_encoding(signal, levels):
        encoded_signal = []
        for sample in signal:
        if sample >= 0:
        encoded_signal.append(1)
        else:
        encoded_signal.append(0)
        return encoded_signal
```

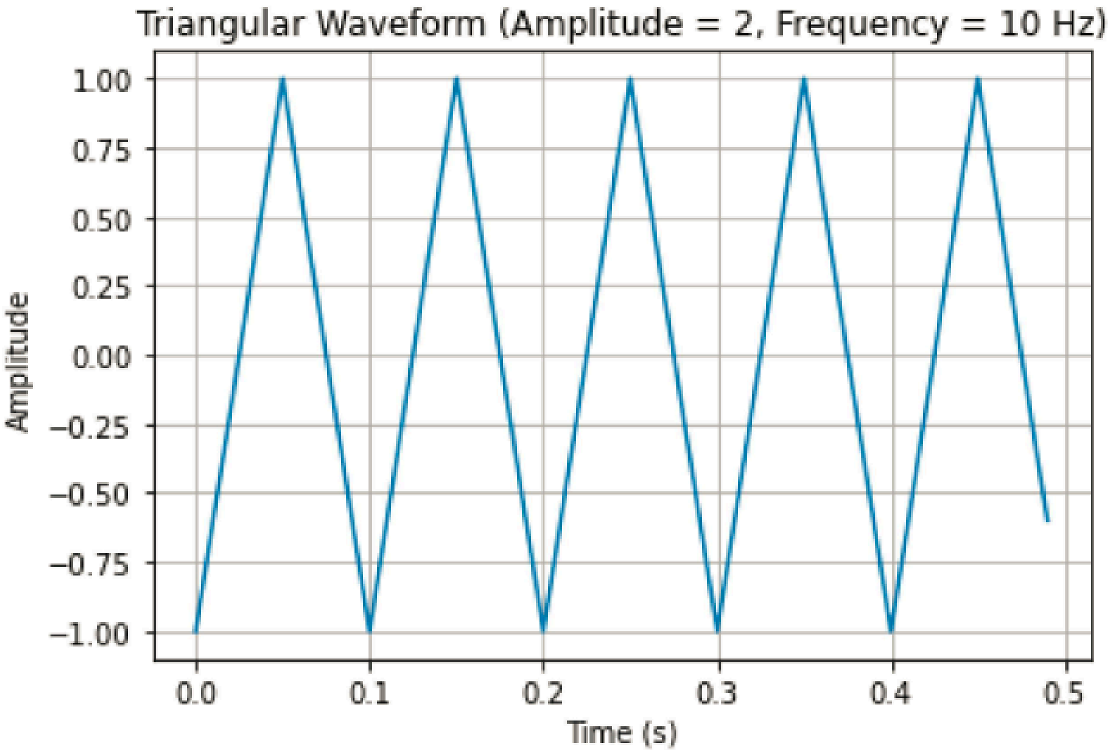
```
# Encode the analog signal
encoded_signal = uniform_pcm_encoding(sin_signal, quantization_levels)
encoded_signal1 = uniform_pcm_encoding(triangle_wave, quantization_levels)

#plot for NRZ of triangular

plt.figure(figsize=(12, 6))
plt.subplot(2, 1, 1)
plt.step(t, encoded_signal1, where='post', color='blue')
plt.title('Bipolar NRZ of Tringular wave')
plt.xlabel('Time')
plt.ylabel('Amplitude(Binary)')
plt.ylim(-0.5, 1.5)
plt.yticks([0, 1], ['0', '1'])

#plot
plt.subplot(2, 1, 2)
plt.step(t, encoded_signal, where='post', color='red')
plt.title('Bipolar NRZ of sine wave ')
plt.xlabel('Time')
plt.ylabel('Amplitude (Binary)')
plt.ylim(-0.5, 1.5)
plt.yticks([0, 1], ['0', '1'])

plt.tight_layout()
plt.show()
# Time multiplexing the signals
multiplexed_signal = np.zeros(len(triangle_wave) * 2)
multiplexed_signal[::2] = triangle_wave
multiplexed_signal[1::2] = sin_signal[:len(triangle_wave)]  # Adjust the length of
sin_signal

# Plotting
plt.figure()
plt.subplot(3, 1, 1)
plt.plot(t_tri, triangle_wave, color='blue')
plt.title('Triangular Wave Signal')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.grid(True)

plt.subplot(3, 1, 2)
plt.plot(t[:len(triangle_wave)], sin_signal[:len(triangle_wave)], color='red')
plt.title('Sinusoidal Wave Signal')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.grid(True)

plt.subplot(3, 1, 3)
plt.step(np.arange(len(multiplexed_signal)), multiplexed_signal, where='post',
color='green')
plt.title('Multiplexed Signal')
```

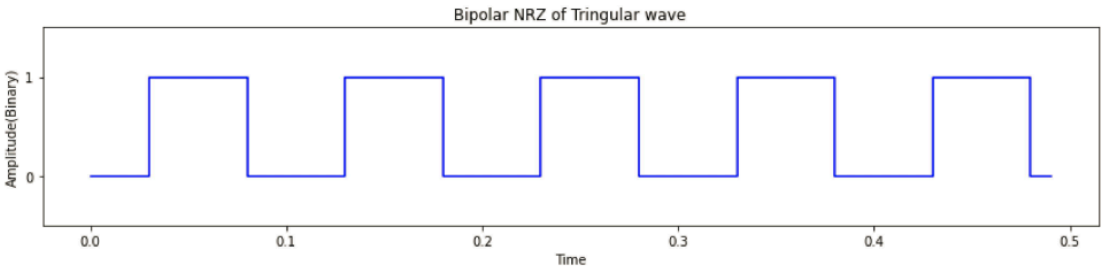
```
plt.xlabel('Time (samples)')
plt.ylabel('Amplitude')
plt.grid(True)

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

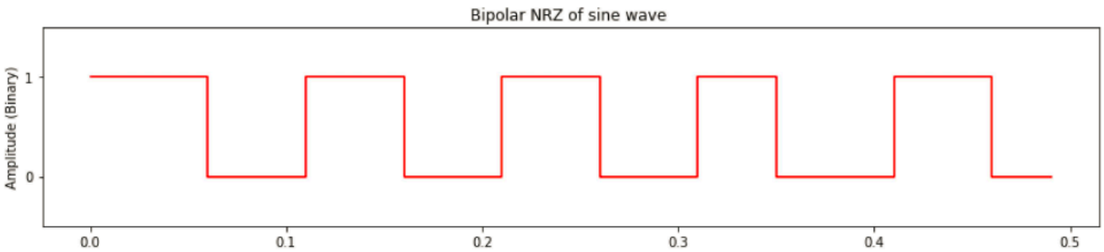
Observation

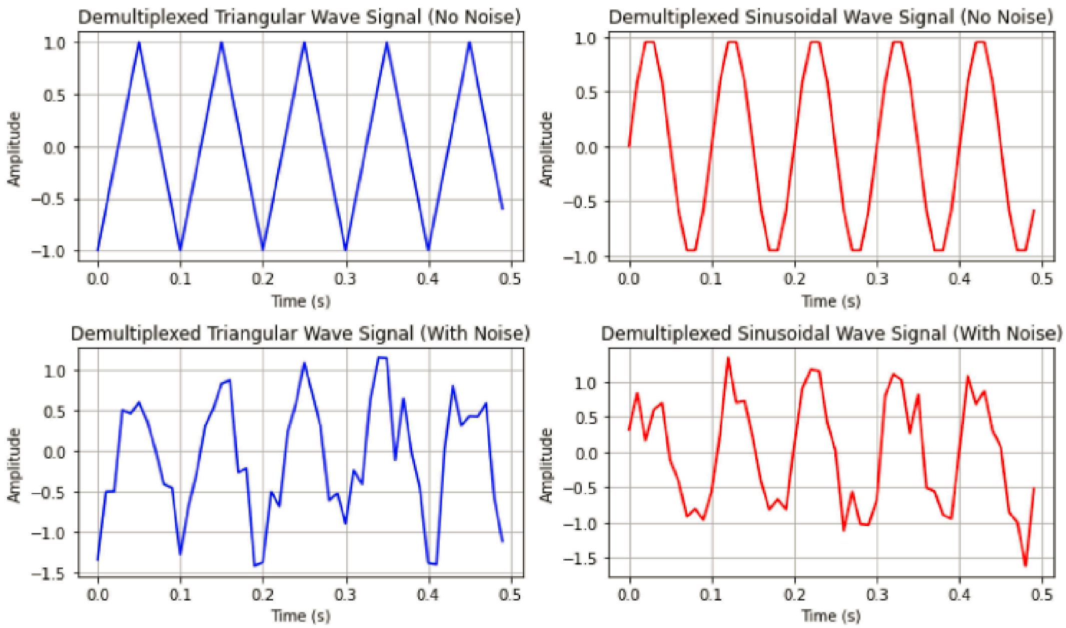
1. **Sinusoidal and Triangular Signals Generation:**
   - Two signals are generated: a sinusoidal signal and a triangular signal.
   - The sinusoidal signal has an amplitude of 2 and a frequency of 10 Hz.
   - The triangular signal has an amplitude of 2, a frequency of 10 Hz, and is plotted for 5 cycles.
2. **Uniform PCM Encoding:**
   - Quantization levels are set to [-1, 1], representing bipolar levels.
   - Uniform Pulse Code Modulation (PCM) encoding is applied to both the sinusoidal and triangular signals.
   - For each sample in the signal, if the sample is non-negative, it is encoded as 1; otherwise, it is encoded as 0.
3. **Bipolar NRZ Plots:**
   - Bipolar Non-Return-to-Zero (NRZ) encoding is visualized for both signals.
   - The triangular wave's binary representation is shown in blue, and the sinusoidal wave's binary representation is shown in red.
   - Each step in the plot represents a sample in the signal, where '0' is represented by a lower level and '1' by a higher level.
4. **Time Multiplexing:**
   - The two signals are time-multiplexed into a single signal for transmission.
   - The multiplexed signal alternates between samples of the triangular and sinusoidal signals.
5. **Plotting Multiplexed Signal:**
   - Three subplots are created:
     - The first subplot shows the triangular wave.
     - The second subplot shows the sinusoidal wave.
     - The third subplot shows the time-multiplexed signal.
   - The multiplexed signal alternates between the triangular and sinusoidal waveforms.


sinusodial signal
amplitude
time
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0
0.1
0.2
0.3
0.4
0.5

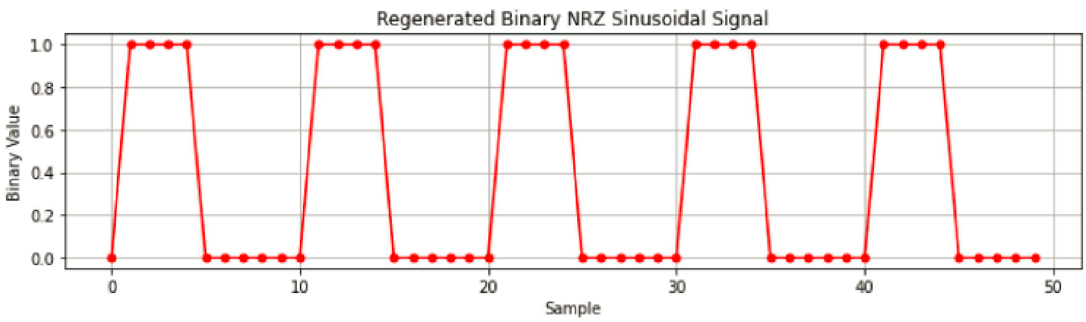

Triangular Waveform (Amplitude = 2, Frequency = 10 Hz)
Amplitude
Time (s)


Bipolar NRZ of Tringular wave
Amplitude(Binary)
Time

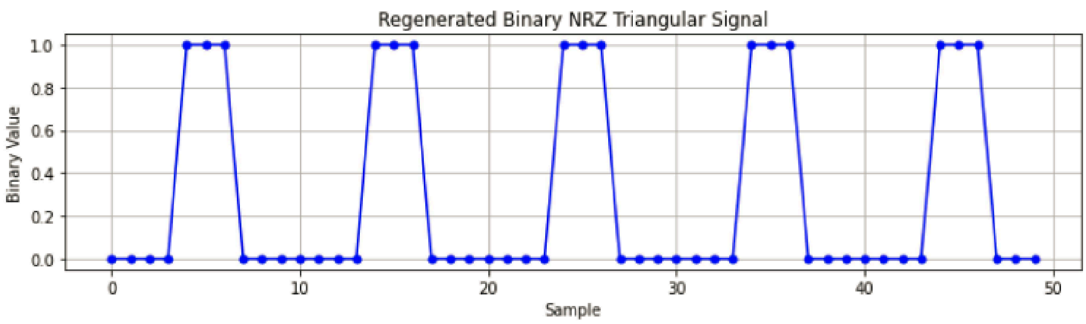

Bipolar NRZ of sine wave
Amplitude (Binary)



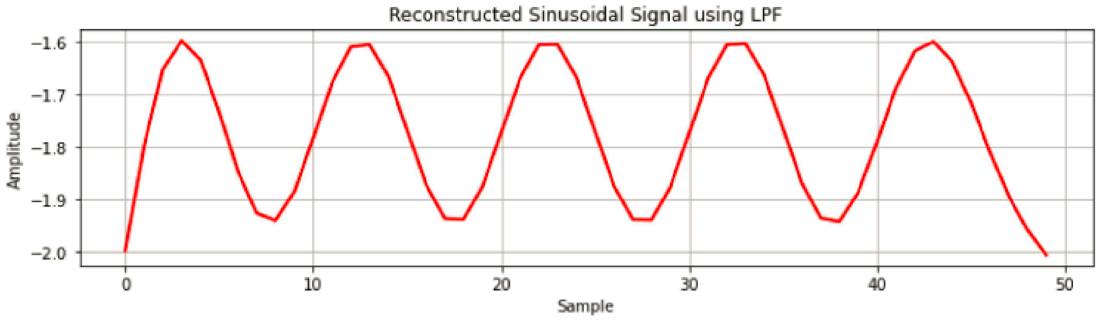
Demultiplexed Triangular Wave Signal (No Noise)
Demultiplexed Sinusoidal Wave Signal (No Noise)
Demultiplexed Triangular Wave Signal (With Noise)
Demultiplexed Sinusoidal Wave Signal (With Noise)
Amplitude
Time (s)

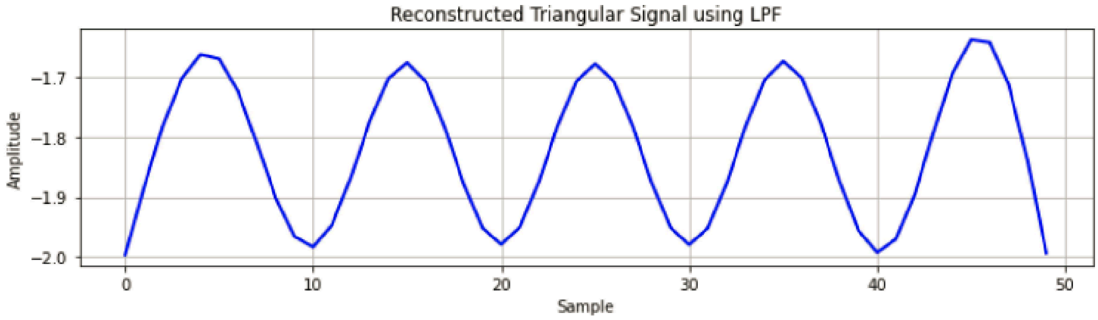


Regenerated Binary NRZ Sinusoidal Signal
Binary Value
Sample



Regenerated Binary NRZ Triangular Signal
Binary Value
Sample


Reconstructed Sinusoidal Signal using LPF
Amplitude
−1.6
−1.7
−1.8
−1.9
−2.0
0
10
20
30
40
50
Sample


Reconstructed Triangular Signal using LPF
Amplitude
−1.7
−1.8
−1.9
−2.0
0
10
20
30
40
50
Sample