Lab Report: *05*

Title: *Quantization and PCM of Baseband Signal*

Course Title: *Data and Telecommunication Laboratory*

Course Code: *CSE-260*

2nd Year 2nd Semester Examination 2023

Submitted to :-

Sarnali Basak
Associate Professor

Dr. Md. Imdadul Islam
Professor

Dr. Md. Abul Kalam Azad
Professor

Department of Computer Science and Engineering
Jahangirnagar University
Savar, Dhaka-1342

Class Roll	Name
371	Md. Ahad Siddiki

Date of Submission: 29/12/2024

Lab Report: Quantization and PCM of Baseband Signal

Objective: The goal of this lab was to design a quantizer for sampled signals, determine the corresponding PCM data, and analyze quantization errors for both sinusoidal and speech signals. The experiments also aimed to observe the effects of quantization and error distribution

Introduction: In modern digital communication systems, the conversion of analog signals into digital form is a critical step to ensure compatibility with digital processing and storage technologies. This process, which involves sampling, quantization, and encoding, lies at the heart of systems such as audio recording, image processing, and telecommunications.

Quantization is the process of mapping a continuous range of signal values to a finite set of discrete levels. This is an essential step in Pulse Code Modulation (PCM), a widely used method for digital signal representation. PCM transforms sampled analog signals into a binary format, making them suitable for digital processing and transmission. However, this transformation introduces quantization errors due to the loss of information during mapping.

This lab focuses on the design and analysis of quantizers for sampled signals. It includes practical applications of quantization and PCM encoding, as well as the investigation of errors introduced during the process. Using MATLAB and Python, signals such as baseband and sinusoidal waves, along with speech signals, were quantized and analyzed. The experiment aims to provide insights into the trade-offs between quantization accuracy, complexity, and the impact of errors on signal fidelity.

Theory: Quantization is the process of mapping a range of values to discrete levels, essential in digital signal processing. Pulse Code Modulation (PCM) involves quantizing and encoding sampled signals into binary form. Errors arise due to differences between the original and quantized values

Experimental Setup:

1. **Software Tools:** MATLAB or Python for numerical computation and signal visualization.
2. **Signal Parameters:** Define the rectangular pulse train with a period and pulse width.
3. **Procedure:**
    - Compute the Fourier coefficients for the rectangular pulse train.
    - Reconstruct the signal using a finite number of terms (e.g).
    - Plot the original signal and the reconstructed signal for comparison.
    - Analyze the overshoot and its behavior as increases.

Codes in MATLAB

Q1. A sinusoidal wave 4sin(πt), as the baseband signal and plot the results.

```
>> t=0:0.1:2*pi;
S=4*sin(pi*t); % sampled signal
quantization = [ -3.25, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.25 ];
partition = [-3, -2, -1, 0, 1, 2, 3];
[I, Q] = quantiz(S, partition, quantization);
% Q gives the quantized value
stem(t ,S, 'b')
hold on
stem(t, Q, 'r*')
legend('Baseband','Sampled Value')
xlabel('time')
ylabel('Amplitude')
grid on
```

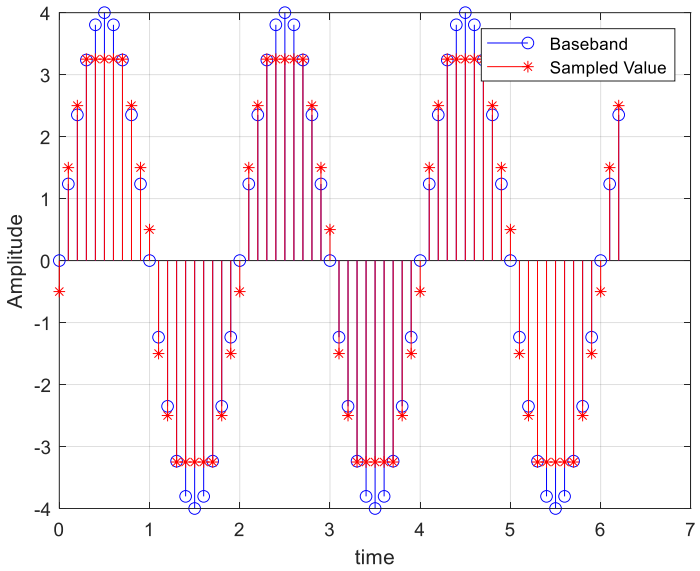


Fig 1

Q2. Determine maximum error and standard deviation of error hence plot error vs. time.

```
>> t=0:0.01:2*pi;
S=4*sin(pi*t); %sampled signal
quantization = [ -3.25, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.25 ];
partition = [-3, -2, -1, 0, 1, 2, 3];
[I, Q] = quantiz(S, partition, quantization);
% Q gives the quantized value
%SD=sqrt(sum((Q-S).^2)/length(Q))
%standards deviation of Q and S
%E=max(abs(Q-S)) %maximum value error
Er=S-Q;
plot(t,Er, 'k')
xlabel('time')
ylabel('Error')
```

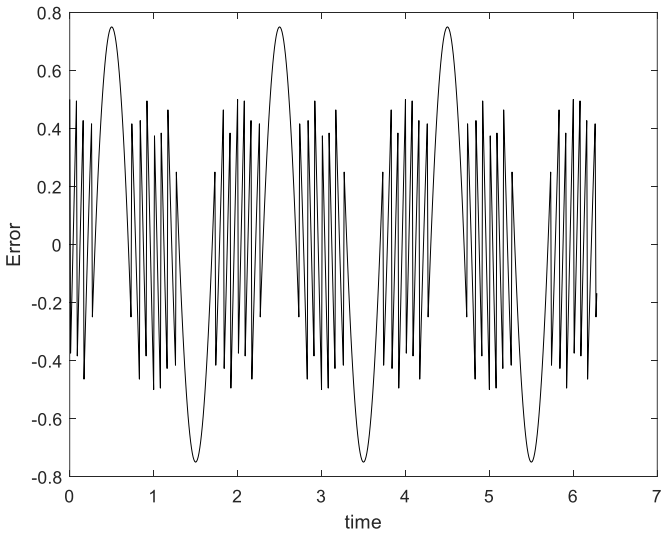


Fig 2

Q3. Following code shows the quantization of a speech signal and the corresponding result in fig 3

```
>> load mtlb;
X=mtlb;
S=X (1200:1300); %Taking 100 samples of speech
quantization = [ -3.25, -2.5, -1.5, -0.5, 0.5, 1.5, 2.5, 3.25 ];
partition = [-3, -2, -1, 0, 1, 2, 3];
[I, Q] = quantiz(S, partition, quantization);
stem(Q, 'r')
hold on
stem(S, 'b*')
legend('Baseband','Sampled Value')
xlabel('Index of samples')
ylabel('Amplitude')
grid on
```

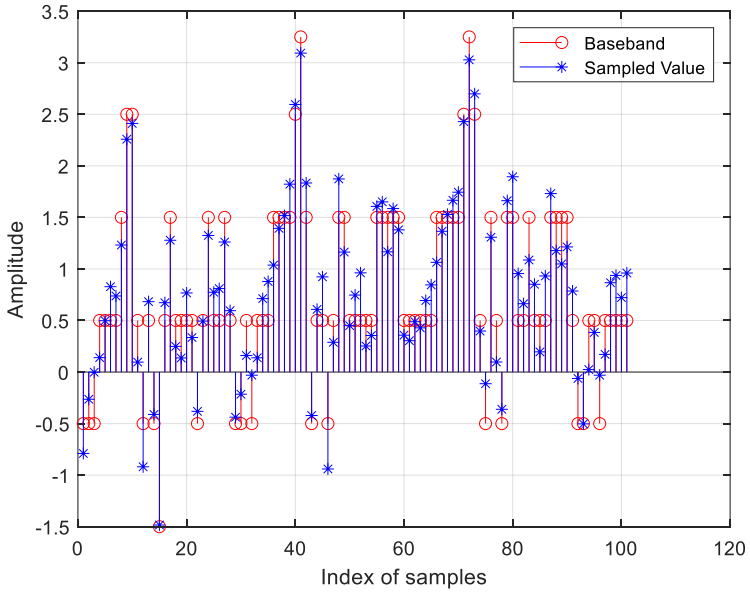


Fig 3

- plot Error signal of speech

```
>> Er=S-Q';
plot(Er, 'k')
xlabel('time')
ylabel('Error')
grid on
```

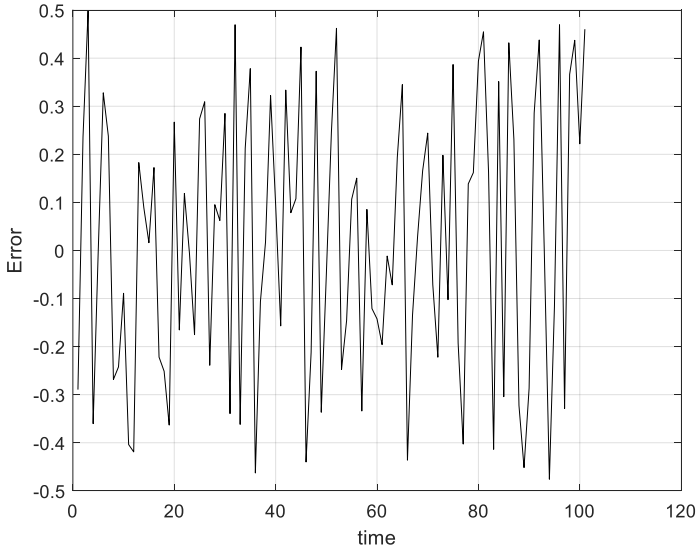
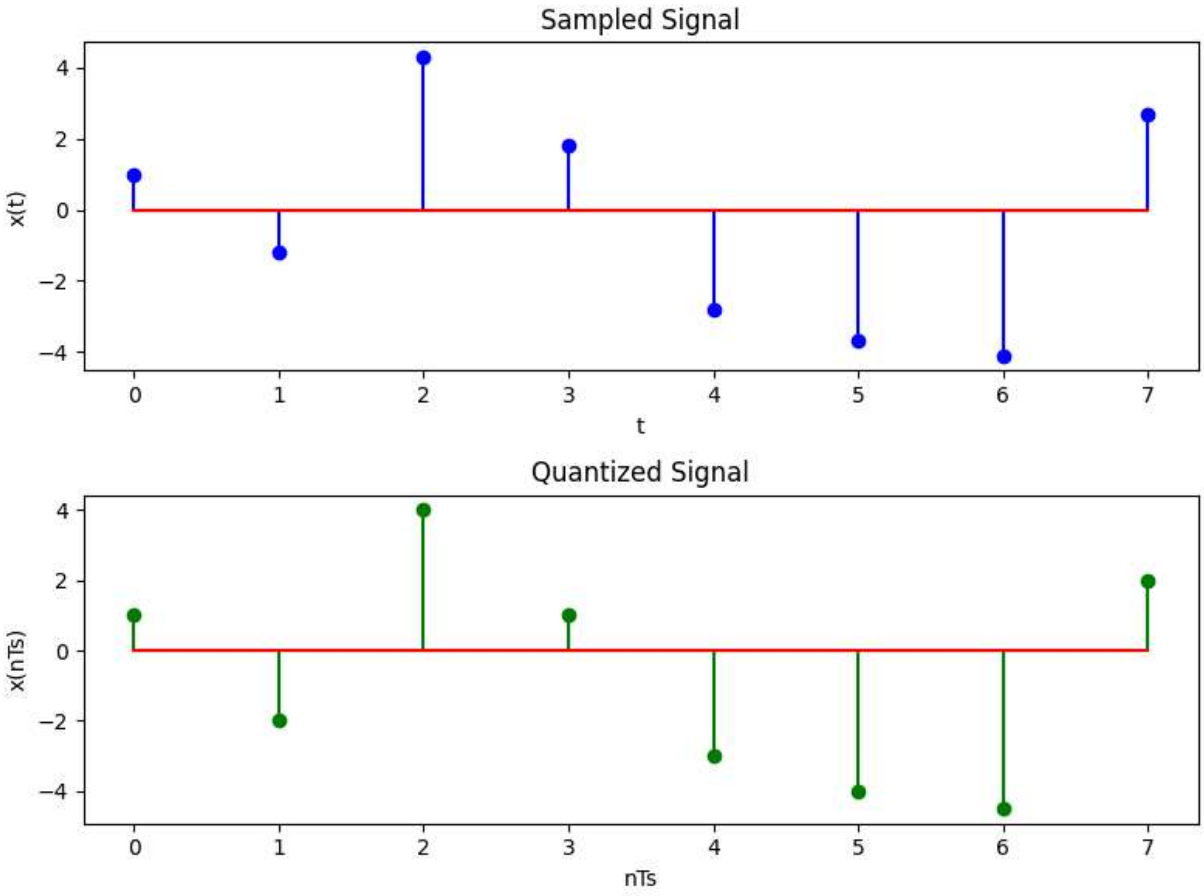


Fig 4

Codes in Python

Sampling and quantization in Python

In [8]:
```python
import matplotlib.pyplot as plt

# Sampled signal and quantization levels
x = [1, -1.2, 4.3, 1.8, -2.8, -3.7, -4.1, 2.7]  # Sampled signal
q = [-4.5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 4.5]  # Quantized values

# Initialize quantized array
quantiz = [0] * len(x)
for k in range(len(x)):
    for j in range(len(q) - 1):
        if x[k] >= q[j] and x[k] <= q[j + 1]:
            quantiz[k] = q[j]

# Create the plots
n = range(len(x))
plt.figure(figsize=(8, 6))  # Set figure size

# Plot the sampled signal
plt.subplot(211)
plt.stem(n, x, linefmt='b-', markerfmt='bo', basefmt='r-')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.title('Sampled Signal')

# Plot the quantized signal
plt.subplot(212)
plt.stem(n, quantiz, linefmt='g-', markerfmt='go', basefmt='r-')
plt.xlabel('nTs')
plt.ylabel('x(nTs)')
plt.title('Quantized Signal')

# Adjust layout to prevent overlapping
plt.tight_layout()
plt.show()
```



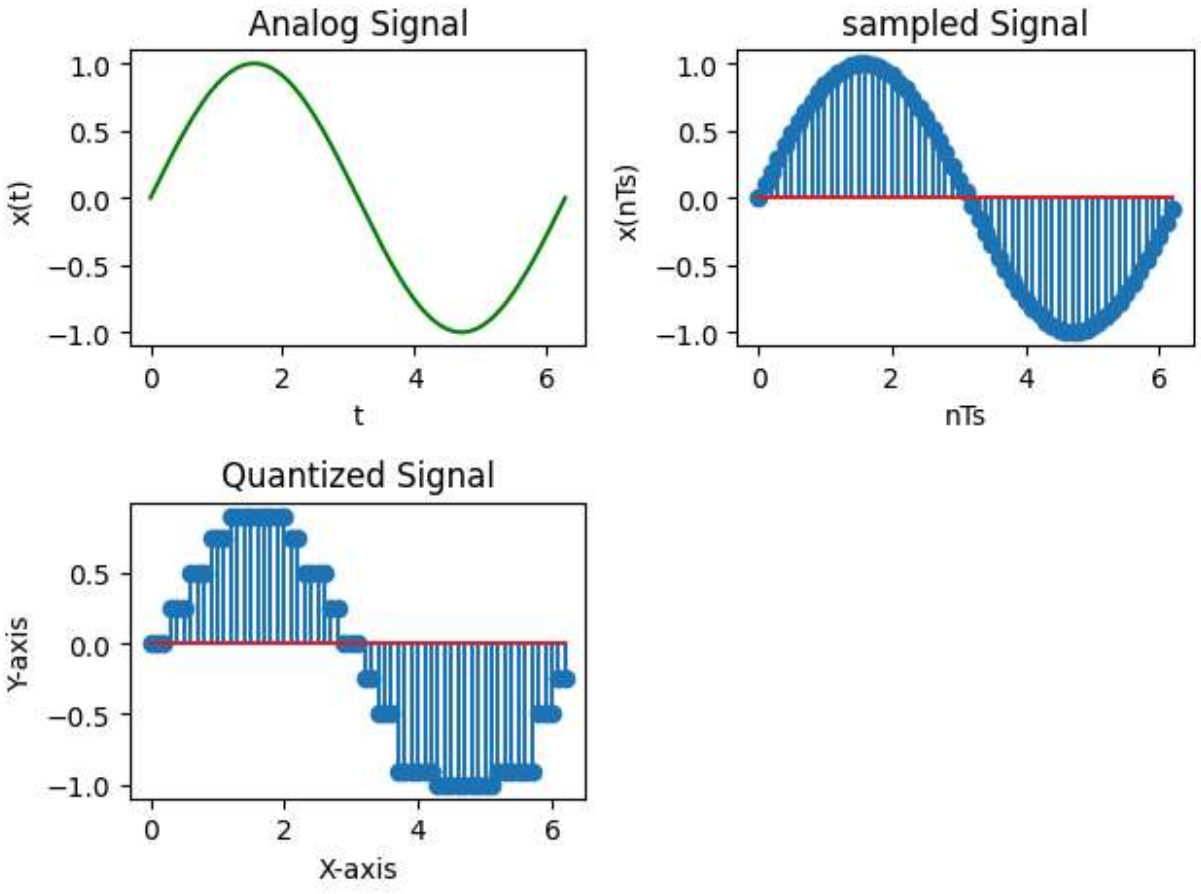
Observations:

1. **Sampled Signal:**
   - The first subplot shows the discrete sampled signal `x`, with positive and negative values of varying amplitudes.
2. **Quantized Signal:**
   - The second subplot displays the quantized signal `quantiz`, where sampled values are mapped to the nearest quantization levels in `q`.
3. **Quantization Effect:**
   - Quantization reduces the resolution, creating a simplified "stair-step" version of the original signal.
4. **Clarity:**
   - Proper titles, labels, and spacing ensure the plots are clear and non-overlapping.

Sampling and quantization of Sinusoidal wave

In [9]:
```
import matplotlib.pyplot as plt
import numpy as np
x = np.arange(0, 2*(np.pi), 0.001)
```

```
y = np.sin(x)
fs=10
s = np.arange(0, 2*(np.pi), 1/fs)
xs = np.sin(s)
plt.subplot(221)
plt.plot(x, y,'g')
plt.xlabel('t')
plt.ylabel('x(t)')
plt.title('Analog Signal')
plt.subplot(222)
plt.stem(s, xs)
plt.xlabel('nTs')
plt.ylabel('x(nTs)')
plt.title('sampled Signal')
quan=[0]*len(s) #initialization of array quan
q = [-1,-0.75,-0.9, -0.5,-0.25, 0,0.25, 0.5,0.75,0.9, 1,1]
for k in range(1, len(xs)):
  for j in range(0, len(q)-1):
    if xs[k]>=q[j] and xs[k]<=q[j+1]:
      quan[k]=q[j]
plt.subplot(223)
plt.stem(s,quan)
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.title('Quantized Signal')
plt.tight_layout()
plt.show()
```



Observations:

1. **Analog Signal (First Plot):**

- The first subplot shows a continuous sine wave (`sin(x)`), representing the analog signal with a time range from 0 to 2π and small steps (0.001).
- The plot is green, with axes labeled as `t` (time) and `x(t)` (amplitude).

2. **Sampled Signal (Second Plot):**

- The second subplot displays the signal sampled at a rate of 10 Hz (`fs=10`), producing discrete points.
- The signal is sampled at intervals of Ts = 1/10 seconds, and the plot shows these points using the `stem()` function.

3. **Quantized Signal (Third Plot):**

- The third subplot represents the quantized version of the sampled signal.
- The `xs` values are quantized to the closest levels in the `q` array, which ranges from -1 to 1 with 12 levels.
- The quantized values are stored in the array `quan` and plotted with `stem()` .

4. **Plot Layout:**

- The plots are arranged in a 2x2 grid, with each signal shown in separate quadrants.
- Axis labels are provided for clarity, and `plt.show()` displays the final result.

5. **Quantization:**

- Each sampled value is mapped to the nearest level in the quantization array, leading to a loss of precision.

Improvements:

- Using more quantization levels could improve signal accuracy.
- Adjusting the sampling frequency or the quantization method could yield better representations.

Conclusion: This experiment successfully demonstrated the process of quantization and PCM encoding for various signal types, including baseband signals, sinusoidal waves, and speech signals. The results highlighted the critical role of partitioning and quantization levels in accurately representing sampled signals in digital form.

The analysis revealed that higher quantization levels improve signal fidelity by reducing the quantization error, albeit at the cost of increased computational complexity and data size. Additionally, the error profile analysis provided valuable insights into the distribution and magnitude of errors, particularly at signal extremes where quantization inaccuracies were most pronounced.

Overall, this lab reinforced the foundational concepts of digital signal processing, emphasizing the trade-offs inherent in the quantization process. These findings are vital for designing efficient and reliable digital communication systems, where balancing accuracy and resource constraints is paramount. Future work could explore adaptive or non-uniform quantization techniques to enhance performance in scenarios involving dynamic signal ranges.