Sampling and Quantization

Laboratory Report

Prepared by:
ENG-219-058/2021

Lecturer:
Mr Martin Wafula

Multimedia Universirty of Kenya
Electrical Engineering

November 27, 2024

Sampling and Quantization

1. Introduction

Digital signal processing relies on fundamental principles such as the Sampling Theorem and quantization to effectively convert and reconstruct signals for digital transmission and storage. The Nyquist-Shannon Sampling Theorem states that a continuous-time signal can be perfectly represented and reconstructed from its discrete samples if it is band-limited and the sampling frequency is at least twice the maximum frequency present in the signal (the Nyquist rate). Band-limiting ensures that the signal is restricted to a range of frequencies, preventing any frequency components higher than the specified maximum frequency. This theorem enables the seamless conversion of continuous-time signals into discrete-time signals without losing critical information, provided the sampling rate satisfies the conditions.

Reconstruction from sampled signals is a crucial operation that ensures the original continuous-time signal can be accurately recovered. By applying interpolation methods and ideal low-pass filters (sinc functions in the frequency domain), it is possible to retrieve the original signal from its sampled version. The quality of reconstruction is heavily influenced by the sampling rate, and undersampling leads to aliasing, which distorts the reconstructed signal.

Quantization is another vital step in digital signal processing, where continuous analog signals are approximated by discrete levels for digital representation. While this process facilitates efficient storage and transmission, it introduces quantization error, which depends on the number of quantization levels. Quantization error impacts signal fidelity, Signal-to-Noise Ratio (SNR), and bitrate, creating trade-offs between quality and efficiency.

This report investigates these core principles through MATLAB simulations, focusing on both the verification of the Sampling Theorem and the effects of quantization. The experiment explores the impact of varying sampling rates on signal reconstruction and examines the relationship between quantization levels, quantization error, SNR, and bitrate. By analyzing these processes, the report highlights the interplay between signal fidelity and efficiency in digital communication systems.

2. Objective

- To analyze and verify the Sampling Theorem.
- To reconstruct the original signal from sampled data.
- To perform quantization.

3. Theoretical Background

The Sampling Theorem can be mathematically described as follows:

$$f_s \geq 2B$$

where:

- f_s: Sampling frequency (samples per second).
- B: The highest frequency component of the continuous-time signal.

When a continuous-time signal $x(t)$ is sampled at a rate f_s, the discrete signal $x[n]$ is obtained by:

$$x[n] = x(nT), \quad \text{where} \quad T = \frac{1}{f_s}$$

To perfectly reconstruct the signal from its samples, f_s must be at least twice the signal's bandwidth. If the sampling rate is lower than the Nyquist rate, aliasing occurs, distorting the reconstructed signal.

The reconstruction of a sampled signal relies on the ideal interpolation formula:

$$x(t) = \sum_{n=-\infty}^{\infty} x[n] \cdot \operatorname{sinc}\left(\frac{t - nT}{T}\right)$$

where:

- $x(t)$: The reconstructed continuous-time signal.
- $x[n]$: The discrete samples of the signal.
- T: The sampling period, related to the sampling frequency by $f_s = \frac{1}{T}$.
- $\operatorname{sinc}(t) = \frac{\sin(\pi t)}{\pi t}$: The sinc function, acting as an ideal low-pass filter.

In practice, sinc interpolation is often approximated by simpler methods like linear or spline interpolation, though sinc interpolation provides theoretically perfect reconstruction when sampling above the Nyquist rate.

4. Methodology

4.1 Sampling Theorem

1. Define the message signal with 1 Hz and 3 Hz sinusoidal components.
2. Plot the message signal in the time domain.
3. Compute and plot its frequency spectrum using FFT.
4. Sample the signal with a sampling period, e.g., 0.02 seconds (50 Hz).
5. Plot the sampled signal in the discrete-time domain.
6. Compute and plot the spectrum of the sampled signal.

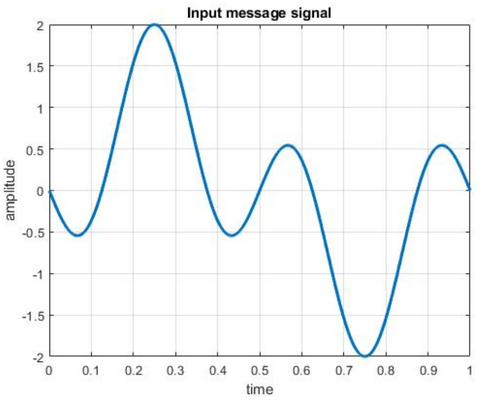
MATLAB Code

Below is a simplified version of the MATLAB code used to verify the Sampling Theorem:

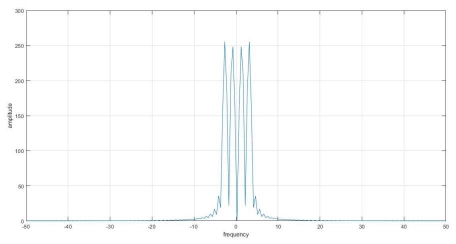
Listing 1: Analysis of Sampling Theorem

```
%% Copyright @ Dr Sudip Mandal

%% ANALYSIS OF SAMPLING THEOREM

clear all;
close all;
clc;

% Define the message signal
tot = 1;
td = 0.002;
t = 0:td:tot;
L = length(t);
x = sin(2*pi*t) - sin(6*pi*t);

% Plot the message signal in time domain
figure(1);
plot(t, x, 'linewidth', 2);
xlabel('time'); ylabel('amplitude');
grid;
title('Input message signal');

% Plot the signal in frequency domain
Lf = length(x);
Lfft = 2^ceil(log2(Lf) + 1);
fmax = 1 / (2 * td);
Faxis = linspace(-fmax, fmax, Lfft);
xfft = fftshift(fft(x, Lfft));

figure(2);
plot(Faxis, abs(xfft));
xlabel('frequency'); ylabel('amplitude');
axis([-50 50 0 300]);
grid;

% Sample the message signal
ts = 0.02;
Nfactor = round(ts / td);
xsm = downsample(x, Nfactor);
tsm = 0:ts:tot;

% Plot the sampled signal (discrete time version)
figure(3);
stem(tsm, xsm, 'linewidth', 2);
```

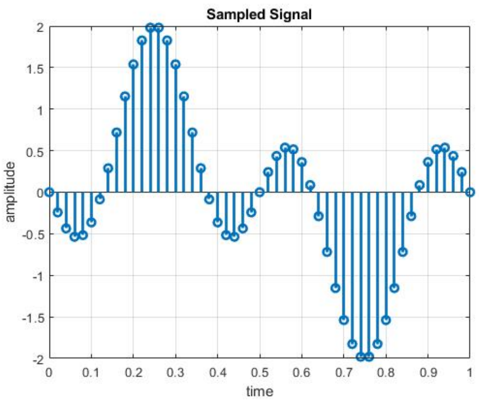
```
xlabel('time'); ylabel('amplitude');
grid;
title('Sampled Signal');

% Compute the spectrum of sampled signal
xsmu = upsample(xsm, Nfactor);
Lfu = length(xsmu);
Lffu = 2^ceil(log2(Lfu) + 1);
fmaxu = 1 / (2 * td);
Faxisu = linspace(-fmaxu, fmaxu, Lffu);
xfftu = fftshift(fft(xsmu, Lffu));

% Plot the spectrum of the sampled signal
figure(4);
plot(Faxisu, abs(xfftu));
xlabel('frequency'); ylabel('amplitude');
title('Spectrum of Sampled signal');
grid;
```



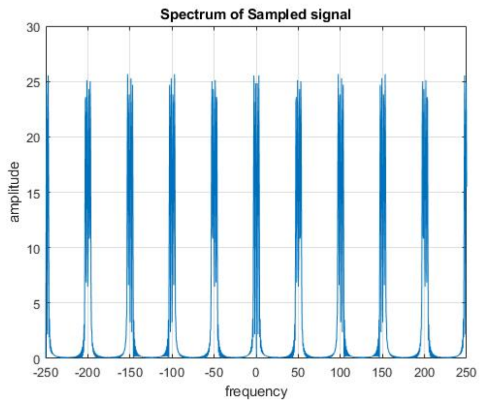
(a) Input message signal



(b) Fast Fourier Transform



(c) Sampled signal



(d) Spectrum of Sampled signal

Figure 1: Sampling Theorem

4.2 Reconstruction from the Sampled Signal

1. Define parameters and generate the signal.
2. Upsample the sampled signal by inserting zeros.
3. Analyze the frequency spectrum of the upsampled signal using FFT.
4. Design a low-pass filter to retain frequencies between $-10\,\text{Hz}$ and $10\,\text{Hz}$.
5. Filter the upsampled signal using the LPF.
6. Apply inverse FFT to convert the filtered signal to the time domain and compare it with the original signal.

MATLAB Code

The following MATLAB code is used to implement the reconstruction process:

Listing 2: Reconstruction from Sampled Signal

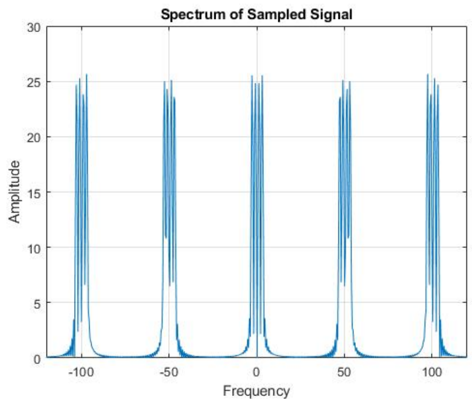
```matlab
%% Copyright @ Dr Sudip Mandal

%% Reconstruction from Sampled Signal

clear all;
close all;
clc;

% Define Parameters and Generate Signal
tot = 1;
td = 0.002;
t = 0:td:tot;
L = length(t);
x = sin(2*pi*t) - sin(6*pi*t);

ts = 0.02;

% Upsample and zero fill the sampled signal
Nfactor = round(ts / td);

xsm = downsample(x, Nfactor);

xsmu = upsample(xsm, Nfactor);

% Frequency Spectrum of Sampled Signal
Lfu = length(xsmu);
Lffu = 2^ceil(log2(Lfu) + 1);
fmaxu = 1 / (2 * td);
Faxisu = linspace(-fmaxu, fmaxu, Lffu);
xfftu = fftshift(fft(xsmu, Lffu));
```

```
% Plot the spectrum of the Sampled Signal
figure(1);
plot(Faxisu, abs(xfftu));
xlabel('Frequency'); ylabel('Amplitude');
axis([-120 120 0 300 / Nfactor]);
title('Spectrum of Sampled Signal');
grid;

% Design a Low Pass Filter
BW = 10;
H_lpf = zeros(1, Lffu);
H_lpf(Lffu / 2 - BW : Lffu / 2 + BW - 1) = 1;

figure(2);
plot(Faxisu, H_lpf);
xlabel('Frequency'); ylabel('Amplitude');
title('Transfer function of LPF');
grid;

% Filter the Sampled Signal
x_recv = Nfactor * (xfftu) .* H_lpf;

figure(3);
plot(Faxisu, abs(x_recv));
xlabel('Frequency'); ylabel('Amplitude');
axis([-120 120 0 300]);
title('Spectrum of LPF output');
grid;

% Inverse FFT for Time domain representation
x_recv1 = real(ifft(fftshift(x_recv)));
x_recv2 = x_recv1(1:L);

figure(4);
plot(t, x, 'r', t, x_recv2, 'b--', 'linewidth', 2);
xlabel('Time'); ylabel('Amplitude');
title('Original vs. Reconstructed Message Signal');
grid;
```

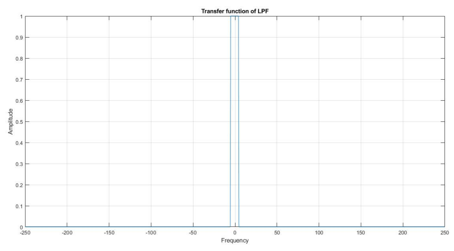
4.3 Quantization

1. Define quantization levels (e.g., 8, 16, 32).
2. Quantize the signal by mapping samples to the nearest levels.
3. Plot the quantized signal and calculate quantization error.

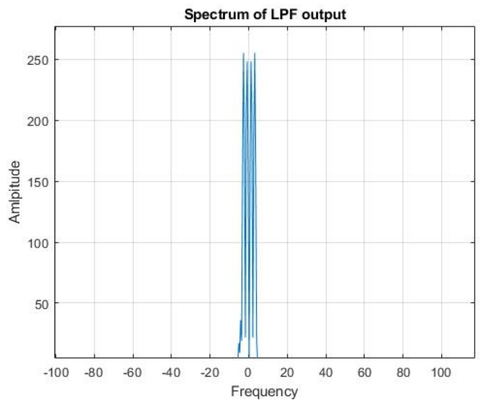
MATLAB Code Outline



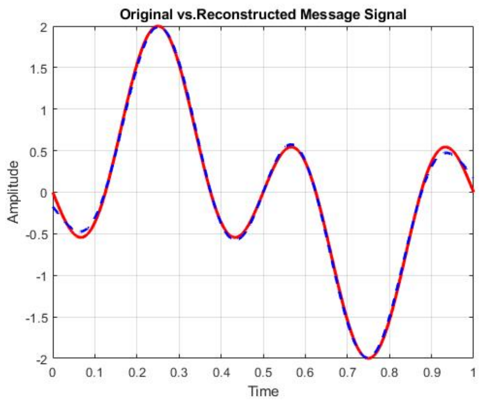
(a) Spectrum of Sampled Signal



(b) Transfer Function



(c) Spectrum of Low Pass Filter



(d) Output vs Original

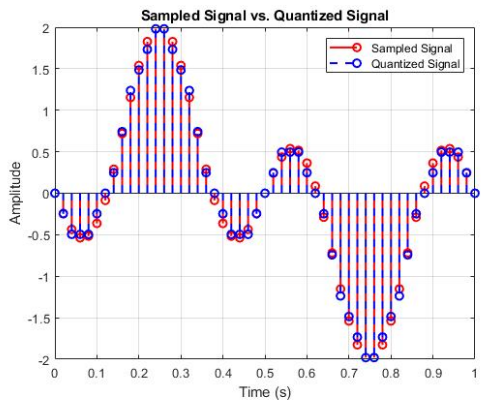
Figure 2: Reconstruction of Sampled Signal

Listing 3: Quantization Process in MATLAB

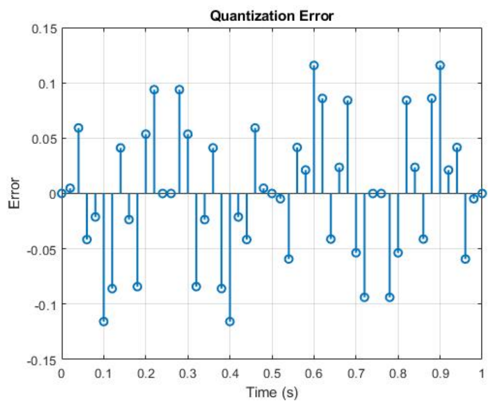
```
%% Copyright @ Dr Sudip Mandal
%% Quantization

clear all;
close all;
clc;

% Define the message signal
tot = 1;
td = 0.002;
t = 0:td:tot;
L = length(t);
x = sin(2*pi*t) - sin(6*pi*t);

% Plot the message signal in time domain
figure(1);
plot(t, x, 'linewidth', 2);
xlabel('time'); ylabel('amplitude');
grid;
title('Input␣message␣signal');
```

```
% Plot the signal in frequency domain
Lf = length(x);
Lfft = 2^ceil(log2(Lf) + 1);
fmax = 1 / (2 * td);
Faxis = linspace(-fmax, fmax, Lfft);
xfft = fftshift(fft(x, Lfft));

figure(2);
plot(Faxis, abs(xfft));
xlabel('frequency'); ylabel('amplitude');
axis([-50 50 0 300]);
grid;

% Sample the message signal
ts = 0.02;
Nfactor = round(ts / td);
xsm = downsample(x, Nfactor);
tsm = 0:ts:tot;

% Plot the sampled signal (discrete time version)
figure(3);
stem(tsm, xsm, 'linewidth', 2);
xlabel('time'); ylabel('amplitude');
grid;
title('Sampled Signal');

% Compute the spectrum of sampled signal
xsmu = upsample(xsm, Nfactor);
Lfu = length(xsmu);
Lffu = 2^ceil(log2(Lfu) + 1);
fmaxu = 1 / (2 * td);
Faxisu = linspace(-fmaxu, fmaxu, Lffu);
xfftu = fftshift(fft(xsmu, Lffu));

% Plot the spectrum of the sampled signal
figure(4);
plot(Faxisu, abs(xfftu));
xlabel('frequency'); ylabel('amplitude');
title('Spectrum of Sampled signal');
grid;

% Quantization process
levels = 16;
x_min = min(xsm);
x_max = max(xsm);
step = (x_max - x_min) / levels;
```

```
% Quantize the sampled signal
x_quantized = step * round((xsm - x_min) / step) + x_min;

% Plot quantized vs. sampled signal
figure(5);
stem(tsm, xsm, 'r', 'LineWidth', 1.5); hold on;
stem(tsm, x_quantized, 'b--', 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Amplitude');
title('Sampled Signal vs. Quantized Signal');
legend('Sampled Signal', 'Quantized Signal');
grid on;

% Quantization error
quantization_error = xsm - x_quantized;
figure(6);
stem(tsm, quantization_error, 'LineWidth', 1.5);
xlabel('Time (s)');
ylabel('Error');
title('Quantization Error');
grid on;
```



(a) Sampled vs Quantized



(b) Quantization Error

Figure 3: Quantization

5. Discussion

In digital signal processing, accurate sampling and quantization are essential for efficient and distortion-free representation of analog signals. The Nyquist rate, defined as twice the maximum frequency of a signal, ensures precise reconstruction without distortion. Sampling below this rate causes **aliasing**, where higher frequencies are misrepresented as lower ones. This is evident in the frequency spectrum as overlapping spectral replicas. To avoid aliasing, the sampling rate must exceed the Nyquist rate, and an anti-aliasing filter should band-limit the signal before sampling.

At or above the Nyquist rate, the frequency spectrum replicates without overlapping, ensuring accurate reconstruction. Practical systems often sample at rates higher than the Nyquist rate to account for filter imperfections, reduce quantization noise, and improve performance. Low-pass filters (LPFs) play a critical role in reconstruction by removing high-frequency components and aliasing artifacts. However, improper filter bandwidth can either distort the desired signal (if too narrow) or allow aliasing artifacts (if too wide).

Quantization converts continuous signals into discrete levels but introduces **quantization error**, which decreases as the number of quantization levels increases. Higher levels improve signal fidelity and reduce distortion. This relationship is reflected in the Signal-to-Noise Ratio (SNR), which increases with quantization levels. The theoretical SNR (in dB) is:

$$\text{SNR} = 6.02N + 1.76,$$

where $N = \log_2 L$ is the number of bits per sample.

The bitrate required for digital communication is calculated as:

$$\text{Bitrate} = \text{Sampling Rate} \times \text{Bits per Sample}.$$

Increasing the sampling rate or quantization levels raises the bitrate, demanding more bandwidth for transmission.

Sampling and quantization are critical in practical systems such as **digital telephony** (8 kHz sampling with 8-bit quantization) and **audio CDs** (44.1 kHz sampling with 16-bit quantization). These processes enable efficient and accurate signal representation, storage, and transmission.

System designers must balance these parameters based on application requirements. Higher sampling rates improve fidelity but increase data size and processing needs, while more quantization levels enhance quality at the cost of higher storage and transmission demands. Optimal choices depend on the desired balance between quality and efficiency.

6. Conclusion

The experiments successfully verified the sampling theorem, reconstruction of original signal from sampled data and quantization. Effective signal representation in digital systems relies on balancing sampling and quantization. The Nyquist rate ensures accurate sampling, while low-pass filters and anti-aliasing prevent distortion. Increasing sampling rates and quantization levels improves quality but demands more resources. Achieving this balance is crucial for reliable performance in systems like telephony and audio recording, aligning efficiency with quality.

7. References

1. Proakis, J.G., & Salehi, M. (2007). *Digital Communications.* McGraw-Hill.

2. Oppenheim, A.V., & Schafer, R.W. (2010). *Discrete-Time Signal Processing.* Pearson.

3. Shannon, C.E. (1949). "Communication in the Presence of Noise." *Proceedings of the IRE*, 37(1), 10–21.