



**Netaji Subhas University  
of Technology**

**LAB REPORT**

# DATA COMMUNICATIONS

Name	<b>Kushagra Lakhwani</b>
Roll No	<b>2021UCI8036</b>
Semester	<b>4th</b>
Course	<b>CICPC12</b>

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

March 19, 2023

---

### **Abstract**

The practical lab report "*Data Communications*" is the original and unmodified content submitted by *Kushagra Lakhwani* (Roll No. 2021UCI8036).

The report is submitted to *Mr. Pattetti*, Department of Computer Science and Engineering, NSUT, Delhi, for the partial fulfillment of the requirements of the course (CICPC12).

# Index

<b>1</b>	<b>Fourier Transform</b>	<b>3</b>
1.1	Matlab Code . . . . .	3
1.2	Output . . . . .	4
<b>2</b>	<b>Random Density Function</b>	<b>4</b>
2.1	Matlab Code . . . . .	4
2.2	Output . . . . .	5
<b>3</b>	<b>Quantization: Uniform</b>	<b>6</b>
3.1	Matlab Code . . . . .	6
3.2	Output . . . . .	7
<b>4</b>	<b>Quantization: Non-Uniform</b>	<b>7</b>
4.1	Matlab Code . . . . .	7
4.2	Output . . . . .	8

# 1 Fourier Transform

We plot a Rectangular Pulse Signal  $x(t)$  in *Matlab* and explore its magnitude and phase spectrum of its Fourier Transform.

## 1.1 Matlab Code

```
close all;

% parameters of a rectangular pulse signal
w = 10;           % width
A = 1;           % amplitude
t = -10:0.01:10; % time vector
xt = A * rectpuls(t, w); % rectangular pulse signal

% plot the rectangular pulse signal in the first subplot
subplot(2, 2, 1)
plot(t, xt)
xlabel('Time')
ylabel('Amplitude')
title('Rectangular pulse')

% define a range of frequencies and compute the Fourier transform at each frequency
w = -8 * pi:0.01:8 * pi; % range of frequencies
for i = 1:length(w)
    xw(i) = trapz(t, xt .* exp(-1i * w(i) .* t)); % Fourier transform
end

% plot the Fourier transform in the second subplot
subplot(2, 2, 2)
plot(w, xw)
title('Fourier transform of rect pulse: Sampling signal')
xlabel('Frequency')
ylabel('Amplitude')

% plot the magnitude spectrum of the Fourier transform in the third subplot
subplot(2, 2, 3)
plot(w, abs(xw))
title('Magnitude spectrum')
xlabel('Frequency')
ylabel('Amplitude')

% plot the phase spectrum of the Fourier transform in the fourth subplot
subplot(2, 2, 4)
plot(w, angle(xw))
title('Phase spectrum')
xlabel('Frequency')
ylabel('Amplitude')
```

## 1.2 Output

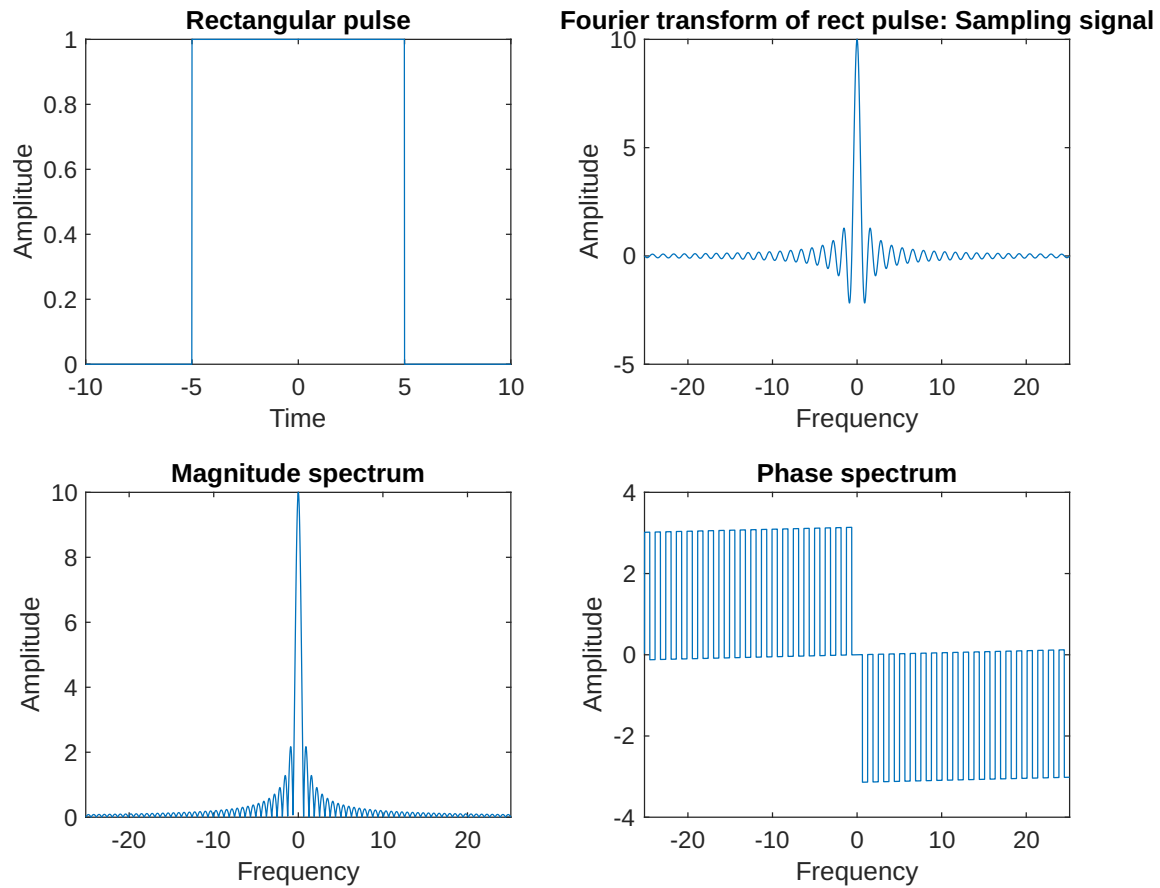


Figure 1: Fourier Transform

## 2 Random Density Function

Using the Gaussian random numbers we find the mean and variance.

### 2.1 Matlab Code

```
% Generate random numbers
data = randn(1000, 1);

% Create histogram
histogram(data, 20, 'Normalization', 'pdf');
hold on;

% Calculate mean and standard deviation
mu = mean(data);
sigma = std(data);
```

```

% Define x values for Gaussian curve
x = linspace(min(data), max(data), 100);

% Calculate y values for Gaussian curve
y = normpdf(x, mu, sigma);

% Overlay Gaussian curve
plot(x, y, 'LineWidth', 2);

% Add title and labels
title('Histogram of Random Data with Gaussian Fit');
xlabel('Data Value');
ylabel('Probability Density');

% Turn off hold
hold off;

```

## 2.2 Output

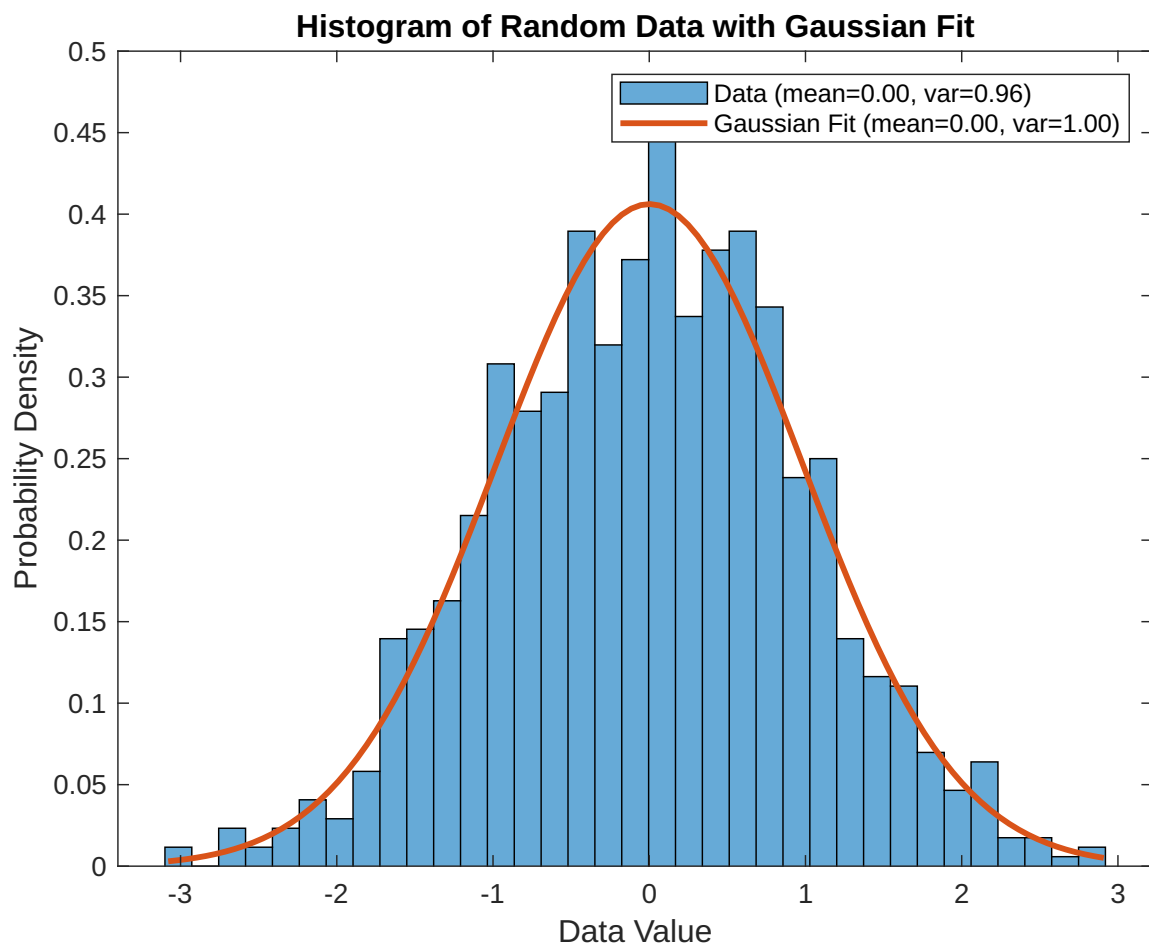


Figure 2: Gaussian Distribution

### 3 Quantization: Uniform

Computing the Signal to quantization Noise ratio of Uniform Quantization. Plot SNQR vs. Quantization levels.

#### 3.1 Matlab Code

*% Program to Compute SQNR of Uniform Quantization and Plot the SQNR vs. Quantization Levels*  
`close all; clc;`

*% Signal Parameters*

`N = 10000;` *% Number of samples in the signal*  
`f = 1;` *% Signal frequency*  
`Fs = 1000;` *% Sampling frequency*  
`t = (0:N - 1) / Fs;` *% Time vector*  
`x = sin(2 * pi * f * t);` *% Signal*

*% Quantization Parameters*

`L = 2:20;` *% Number of quantization levels to try*  
`b = log2(L);` *% Number of bits to represent each level*  
`Delta = 2 ./ (L - 1);` *% Step size of the quantization levels*  
`SQNR = zeros(length(L), 1);` *% To store the Signal to Quantization Noise Ratio (SQNR) for each quantization level*

*% Uniform Quantization*

`for i = 1:length(L)`  
`q = round(x / Delta(i)) * Delta(i);` *% Quantize the signal*  
`% Compute the SQNR`  
`noise = x - q;`  
`signal_power = sum(x .^ 2) / N;`  
`noise_power = sum(noise .^ 2) / N;`  
`SQNR(i) = 10 * log10(signal_power / noise_power);`  
`end`

*% Plot the SQNR vs. Quantization Levels*

`figure;`  
`plot(b, SQNR, 'b-o', 'LineWidth', 2);`  
`xlabel('Number of Bits');`  
`ylabel('Signal to Quantization Noise Ratio (dB)');`  
`grid on;`

### 3.2 Output

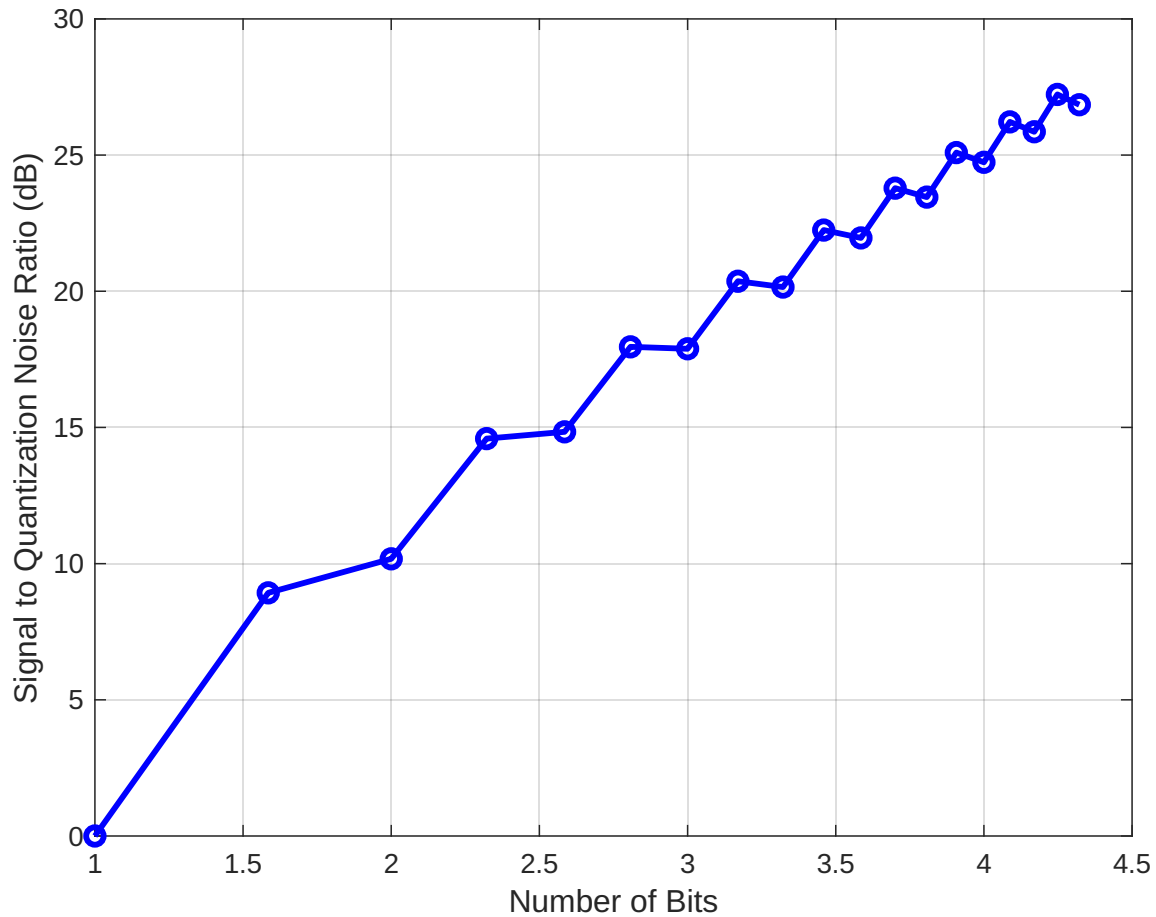


Figure 3: SQNR vs Quantization

## 4 Quantization: Non-Uniform

Computing SNR of Non-Uniform Quantization and Plot SNR vs. Quantization Levels

### 4.1 Matlab Code

```
% Program to Compute SNR of Non-Uniform Quantization and Plot the SNR vs. Quantization Levels
close all; clc;

% Signal Parameters
N = 10000;           % Number of samples in the signal
f = 1;               % Signal frequency
Fs = 1000;           % Sampling frequency
t = (0:N - 1) / Fs;  % Time vector
x = sin(2 * pi * f * t); % Signal
```



```
% Quantization Parameters
L = 2:20;                                % Number of quantization levels to try
b = log2(L);                             % Number of bits to represent each level
Delta = 2 ./ (L - 1);                    % Step size of the quantization levels
SQNR = zeros(length(L), 1);              % To store the Signal to Quantization Noise Ratio (SQNR) for each qu

% Non-Uniform Quantization
for i = 1:length(L)
    q = zeros(size(x));
    % Compute quantization levels
    V = [- (L(i) - 1) / 2 : 1 : (L(i) - 1) / 2] * Delta(i);
    % Quantize the signal
    for j = 1:N
        [val, index] = min(abs(x(j) - V));
        q(j) = V(index);
    end

    % Compute the SQNR
    noise = x - q;
    signal_power = sum(x.^2) / N;
    noise_power = sum(noise.^2) / N;
    SQNR(i) = 10 * log10(signal_power / noise_power);
end

% Plot the SNR vs. Quantization Levels
figure;
plot(b, SQNR, 'b-o', 'LineWidth', 2);
xlabel('Number of Bits');
ylabel('Signal to Quantization Noise Ratio (dB)');
grid on;
```

## 4.2 Output

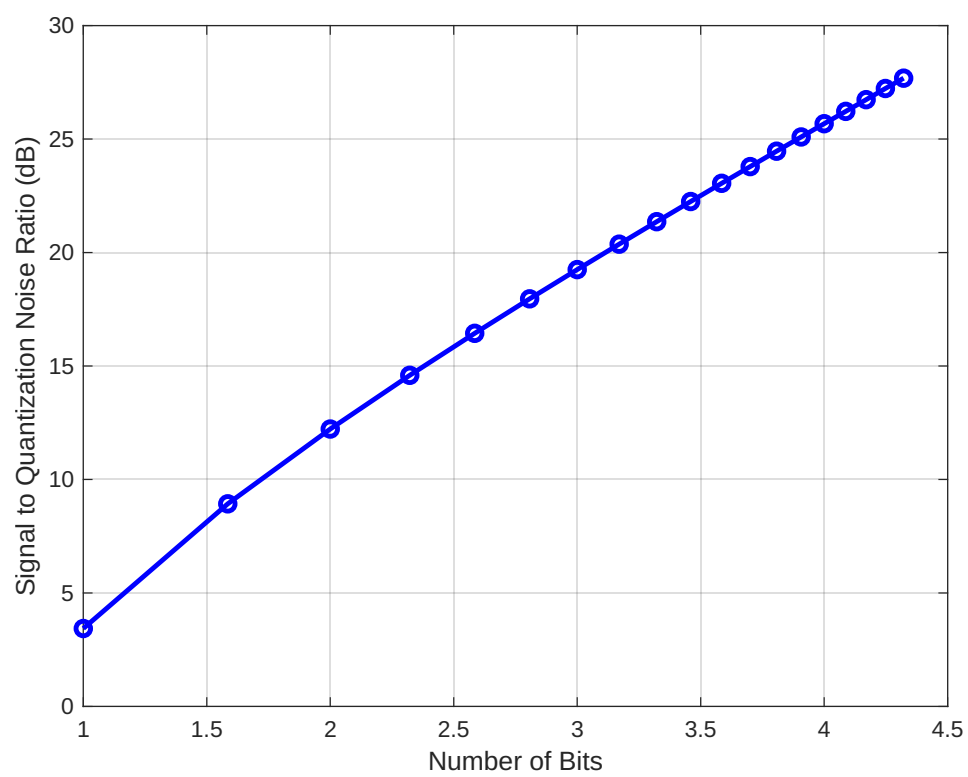


Figure 4: SQNR vs Quantization (non-uniform)