JOMO KENYATTA UNIVERSITY OF AGRICULTURE AND TECHNOLOGY

TELECOMMUNICATION SYSTEMS LAB REPORT

COURSE OF STUDY: BSc. TELECOMMUNICATION AND INFORMATION ENGINEERING

GROUP WORK: TELECOMMUNICATION LAB REPORT ETI 2506

LAB 2

STUDENT NAME	REGISTRATION NUMBER
1. PATIENCE SIMULI	ENE221-0110/2018
2. LEVIN WASIKE	ENE221-0300/2016
3. BENSON K. CHEGE	ENE221-0283/2016
4. WYCLIFFE WASONGA	ENE221-0122/2017

INTRODUCTION

Digital signal processing is a crucial field in modern technology that deals with the manipulation, analysis, and representation of signals in a discrete form. MATLAB, a widely-used programming environment, provides an excellent platform for signal processing tasks due to its comprehensive toolset and user-friendly interface. In this lab session, we delve into the fundamental concepts of digital signal generation and manipulation using MATLAB.

The primary objective of this lab session is to gain hands-on experience in generating and processing basic digital signals. We will explore various signal types, such as unit responses, step functions, sign functions, rectangles, sine waves, and sine cardinals. These exercises aim to familiarise us with MATLAB's signal plotting functions, such as stem and plot, and impart knowledge of signal characteristics such as frequency, duration, and amplitude.

This laboratory session also explores aliasing and quantization. Aliasing is a phenomenon that occurs when a continuous signal is sampled at a rate that is too low to accurately represent its original characteristics. This undersampling leads to false representations of high-frequency components in the signal, resulting in distortion and ambiguity in the reconstructed signal. Quantization is a process in signal processing and digital systems where continuous values are approximated or represented by discrete values from a finite set. It involves converting an analog signal into a digital format by assigning discrete numerical values to the continuous amplitude levels of the signal.

METHODOLOGY AND FINDINGS

Exercise 2.1: Basic digital signals Generation

a. **Write a MATLAB program to generate and display (using the stem function) the signals defined in Table 1.**

i. The Dirac function

Dirac.m
/MATLAB Drive/Dirac.m

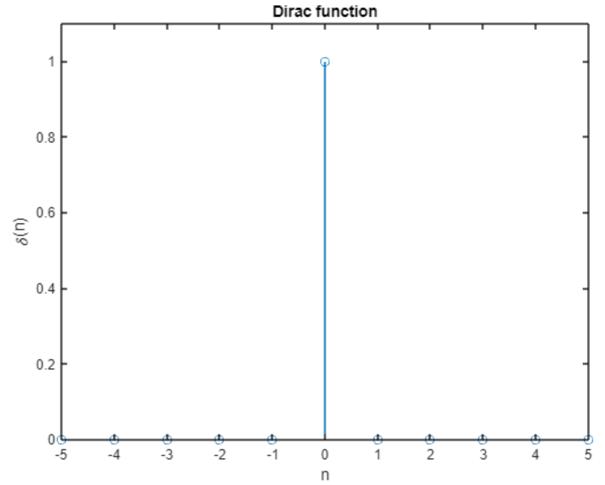
```matlab
% Dirac (Unit Impulse)
N = 10; % Given number of samples
n = -N/2:N/2; % vector

d = [zeros(1,N/2) 1 zeros(1,N/2)];

figure; % display

stem(n,d);
xlabel('n');
ylabel('\delta(n)');
title('Dirac function');
axis([-N/2 N/2 0 1.1]);
```

Figure 1



ii. Unit Step (Heaviside)

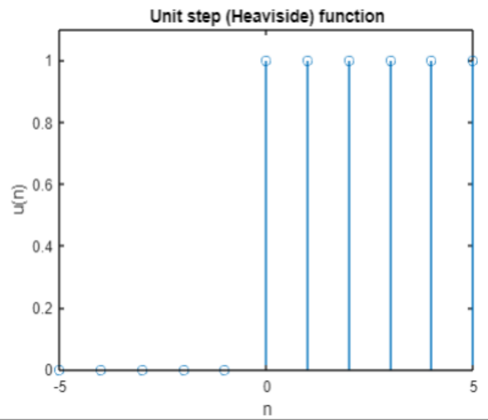
UnitStep.m
/MATLAB Drive/UnitStep.m

```matlab
% Unit Step (Heaviside) function
N = 10; % number of samples
n = -N/2:N/2;

unit_step = [zeros(1,N/2) 1 ones(1,N/2)];

figure; % displaying the graph
stem(n,unit_step);
xlabel('n');
ylabel('u(n)');
title('Unit step (Heaviside) function');
axis([-N/2 N/2 0 1.1]);
```

Figure 1



iii. Sign function

Sign_Function.m

/MATLAB Drive/Sign_Function.m

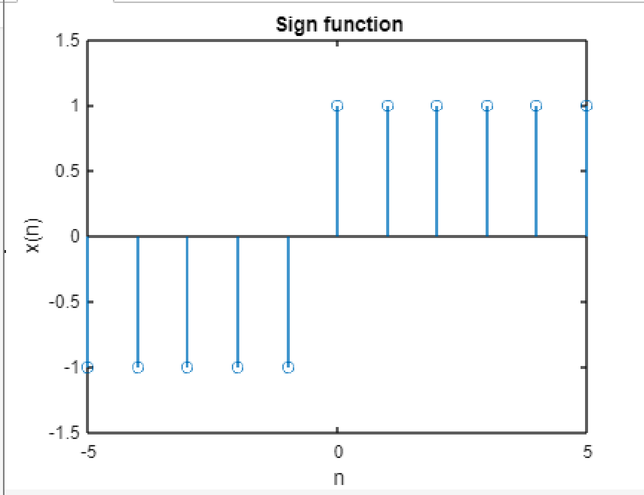
```
% Sign function
N = 10; % number of samples
n = -N/2:N/2;

u_n = [zeros(1,N/2) 1 ones(1,N/2)]; % Unit step function

x_n = 2.*u-1;

figure;
stem(n,x);
xlabel('n');
ylabel('x(n)');
title('Sign function');
axis([-N/2 N/2 -1.5 1.5]);
```

Figure 1



iv. Rectangle function

Rectangle_Function.m

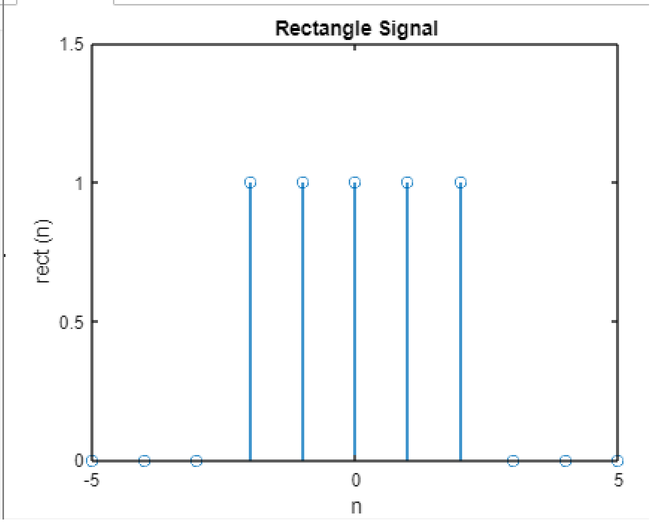
/MATLAB Drive/Rectangle_Function.m

```
N = 10; % number of samples
n = -N/2:N/2; % vector
M = 2;

rectangle = [zeros(1,N/2-M) ones(1,2*M+1) zeros(1,N/2-

stem(n,rectangle);
title('Rectangle Signal');
xlabel('n');
ylabel('rect (n)');
axis([-N/2 N/2 0 1.5]);
```

Figure 1



v. Sine function

Sine_Function.m

/MATLAB Drive/Sine_Function.m

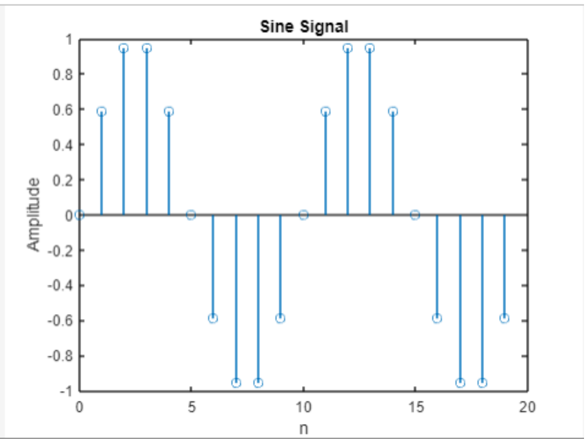
```
L = 20;
n = 0:L-1;
f0 = 100; % initial frequency in Hz
fs = 1000; % sampling frequency in Hz
Ts = 1/fs; % sampling interval in s

x = 1;
x_n = x*sin(2*pi*f0/fs*n);

figure; % display

stem(n,x_n);
xlabel('n');
ylabel('Amplitude');
title('Sine Signal');
```

Figure 1



vi. Sine Cardinal Function

Sine_Cardinal.m

/MATLAB Drive/Sine_Cardinal.m

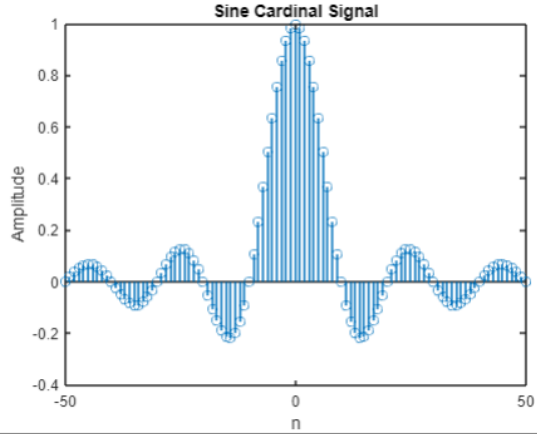
```
L = 50;
n = -L:L;
Ts = 0.1; % Sampling interval (seconds)

% Defining at n not = to 0
x(n~=0) = sin(pi*n(n~=0)*Ts)./(pi*n(n~=0)*Ts);

% Defining at n = 0
x(n==0) = 1;

stem(n,x);
title ('Sine Cardinal Signal');
xlabel('n');
ylabel('Amplitude');
```

Figure 1



(b) Write a MATLAB function [x, t] = sin_NU(f0, fs, T) to generate a sine signal. The output parameters x and t are the signal and time vectors, respectively. The input parameters are f0 (signal frequency in Hz), fs (sampling frequency in Hz), T (signal duration in seconds).

sin_NU.m

/MATLAB Drive/sin_NU.m

```
function [x,t]=sin_NU(fs,f0,T)

    t = 0:1/fs:T;

    x = sin (2*pi*f0*t);

end
```

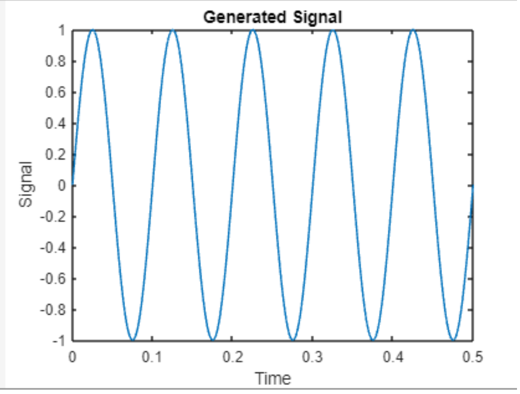
(c) Test your sin_NU function with the input parameter values below and display the result using the plot function.

{f0 = 10, fs = 1000, T = 0.5}

```
[x,t]=sin_NU(1000,10,0.5);


plot(t,x);
title('Generated Signal');

xlabel('Time');
ylabel('Signal');
```



Exercise 2.2: Audio aliasing

To illustrate the aliasing phenomenon, let's perform two simple experiments allowing us to "hear" it. Using the sin_NU function of Exercise 2.1:

a) Generate two 1 kHz sine signals (2 seconds duration), first signal at 20 kHz sample frequency and second signal at 1.5 kHz sample frequency;

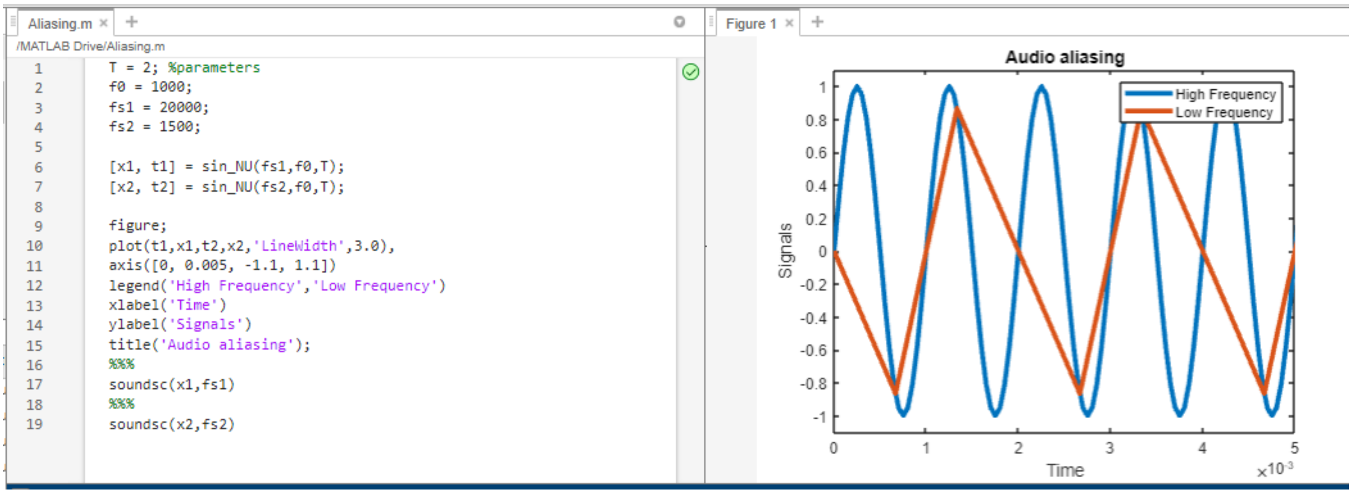
```
T = 2; %parameters
f0 = 1000;
fs1 = 20000;
fs2 = 1500;

[x1, t1] = sin_NU(fs1,f0,T);
[x2, t2] = sin_NU(fs2,f0,T);

figure;
plot(t1,x1,t2,x2,'LineWidth',3.0),
axis([0, 0.005, -1.1, 1.1])
legend('High Frequency','Low Frequency')
xlabel('Time')
ylabel('Signals')
title('Audio aliasing');
%%%
soundsc(x1,fs1)
%%%
soundsc(x2,fs2)
```

(b)) On the same graph, use the plot function to display the two signals versus t in the range below:

$0 \leq t \leq 5$ msec



(c) Listen to the two signals one after another using the function soundsc(x, fs); and (d) Give your interpretation of this listening.

When listening to the two signals, it was noticeable that the first signal, which is sampled at 20 kHz, sounds like a pure 1 kHz tone. However, the second signal, sampled at 1.5 kHz, wasn't a pure 1kHz tone. The sampling frequency of 1.5 kHz is less than twice the signal frequency. This means that the higher frequency components of the signal fold back into the lower frequency range during sampling, causing aliasing.

<u>Exercise 2.3: Quantization</u>

Quantization is done by replacing each value of an analog signal x(t) by the value of the nearest quantization level. To exemplify this operation, let's simulate a unipolar ADC (Analog to Digital Converter) having the technical specifications: R = 10 Volts (full-scale range) and B = 3 (number of bits).

(a) Write a MATLAB function y = adc_NU(x, R, B) where x and y are vectors containing the input signal and the quantized signal, respectively;

```
function y = adc_NU(x, R, B)
level = [0:R/(2^B):R-R/(2^B)];
temp = [-Inf,(level(2:end)-R/(2^(B+1))),Inf];
y = zeros(1,length(x));
for i = 1:length(level)
y = y + (x >= temp(i)).*(x < temp(i+1)).*level(i);
end
```

(b) Test your function with an input ramp signal ranging from -5 to 15 Volts (1 volt per step).

The code below was used for the test:

adc_NU.m ADC_NU_Test.m

/MATLAB Drive/ADC_NU_Test.m

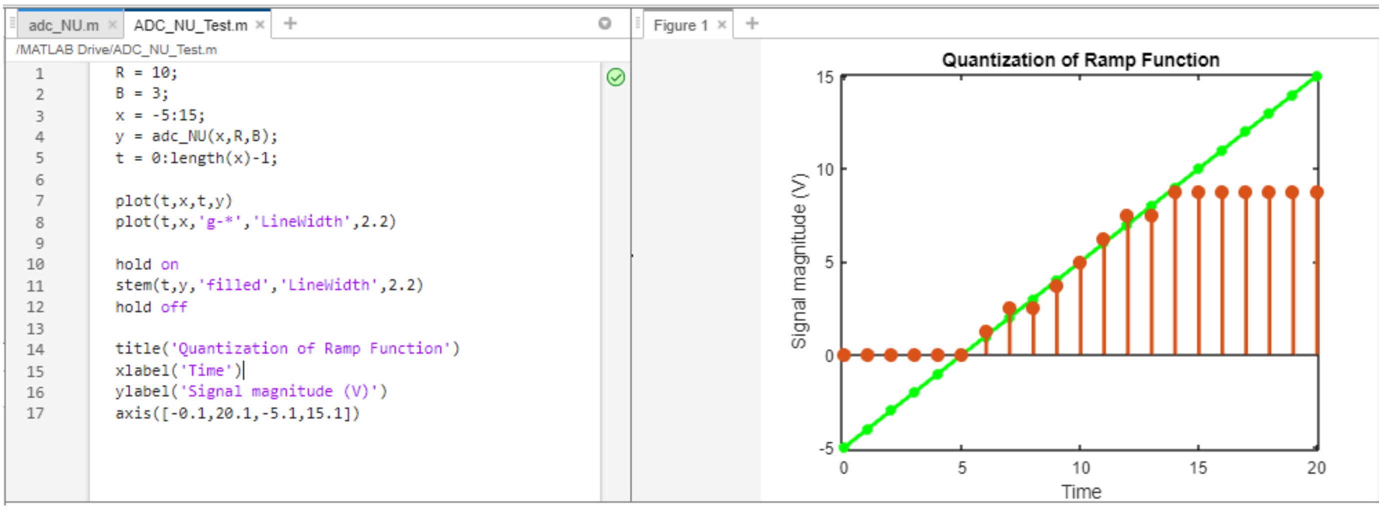
```
R = 10;
B = 3;
x = -5:15;
y = adc_NU(x,R,B);
t = 0:length(x)-1;

plot(t,x,t,y)
plot(t,x,'g-*','LineWidth',2.2)

hold on
stem(t,y,'filled','LineWidth',2.2)
hold off

title('Quantization of Ramp Function')
xlabel('Time')
ylabel('Signal magnitude (V)')
axis([-0.1,20.1,-5.1,15.1])
```

(c) On the same graph, use the plot and stem functions to display the input signal and quantized signal, respectively.



Conclusion

By actively participating in these lab exercises, we acquired practical skills in generating, visualising, and manipulating digital signals using MATLAB. These foundational concepts are essential for anyone seeking to delve into advanced topics in signal processing, communications, and related fields. The insights gained from this lab session will serve as a solid foundation for more complex signal processing tasks and applications in real-world scenarios.