|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **FAKULTI TEKNOLOGI KEJURUTERAAN**  **ELEKTRIK DAN ELEKTRONIK**  **UNIVERSITI TEKNIKAL MALAYSIA MELAKA** | | | | | |
| **DIGITAL SIGNAL PROCESSING** | | | | | |
| **BEET3373** | | | **SEMESTER 1** | **SESI 2021/2022** | |
| **LAB 1: INTRODUCTION TO SIGNAL ANALYSIS** | | | | | |
| **NO.** | **STUDENTS' NAME** | | | | **MATRIC. NO.** |
| **1.** | **AHMAD IRFAN BIN HARMAN** | | | | **B081910068** |
| **2.** | **MOHAMED HAZEEM BIN HASHAINI** | | | | **B081910350** |
| **3.** | **MUHAMAD AZIM HAMZI BIN AZAHA** | | | | **B081910086** |
| **4.** | **RAHMAN KAZI ASHIKUR** | | | | **B081910450** |
| **PROGRAMME** | | **3 BEEC** | | | |
| **SECTION /**  **GROUP** | | **3BEEC S1/1** | | | |
| **DATE** | | **05/1/2021** | | | |
| **NAME OF**  **INSTRUCTOR(S)** | | **1. DR.jamil** | | | |
| **2.** | | | |
| **EXAMINER’S COMMENT(S)** | | | | **TOTAL MARKS** | |

**LAB 1: INTRODUCTION TO SIGNAL**

# 1.0 OBJECTIVES

Student able to:

1. To learn to use simulation tools, MATLAB, in signal analysis.
2. To understand basic digital signal time characteristics and operations.

# 2.0 EQUIPMENT

|  |  |
| --- | --- |
| **Hardware** | **Type/Version** |
| 1. Workstation (Computer) | Windows 7 |
| 2. MATLAB | R2013 |

# 3.0 SYNOPSIS & THEORY

MATLAB is a simulation tools and used extensively in both academic and industrial environments for numerical computation and data visualization. MATLAB allows matrix manipulation, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs in other languages. In this lab, MATLAB is used for signal analysis.

The signals arising in DSP are basically discrete-time signals. Discrete-time sequence, *x*[*n*] is developed by uniformly sampling a continuous-time signal, *xa*(*t*). *x*[*n*] *xa* *t**t**nTs*  *xa* *nTs* , *n* ,2, 1, 0, 1, 2,

*Ts*is a sampling time and sampling frequency, *Fs = 1/Ts*.

The Shannon sampling theorem says that a signal can be completely recovered from a set of samples if the sampling frequency Fs is greater than two times the maximum frequency, *Fs > 2Fm.* This maximum frequency *Fm*is known as the *Nyquist frequency*. If the sampling frequency is not greater than two times the *Nyquist* frequency, the signal cannot be uniquely recovered and *aliasing* occurs.

**4.0 PROCEDURE**

# 1. Familiarizing with Simulation Tools

1. Launch the MATLAB software.
2. MATLAB is a ‘matrix-based’ simulation tool in which discrete-time signals are represented in the form of matrices. Type the following code using the MATLAB editor and rename the file as **mylab1\_sig1.m**.

|  |
| --- |
| %Common signals clear; close;  %Signal x1 - Sinusoidal signal  L1 = 60; %Signal length (no. of samples)  fs1 = 20; %Sampling frequency for discrete-time signal  f1 = 3;%Frequency of 3 Hz  t1 = [0:L1-1]/fs1; %Signal sampling instants  x1 = cos(2\*pi\*f1\*t1); %Sinusoidal signal  x1 stem(t1, x1); %Plotting x1 with discrete samples  xlabel('Time (s)'); ylabel('Amplitude'); title('Cosine signal (3Hz) sampled at 20Hz'); |

Chart

Description automatically generated

## Exercises

1. What is the sampling period used in generating the signal x1 in **mylab1\_sig1.m**?

 sampling period = 1/fs = **0.05 s**



we have aliasing here, changing the frequency of x1 from 3 Hz to 17 Hz has no effect on the signal. As a result, the two signals will be regarded to have the same frequency even if their frequencies are different. This is due to the absence of the Nyquist criterion (fs>=2f) in this case.

The Nyquist requirements are not violated when the frequency is changed to 4 Hz, 5 Hz, or 7 Hz, but they are violated when the frequency is changed to 16 Hz, 15 Hz, or 13 Hz (fs is not 2 times of signal original frequency).

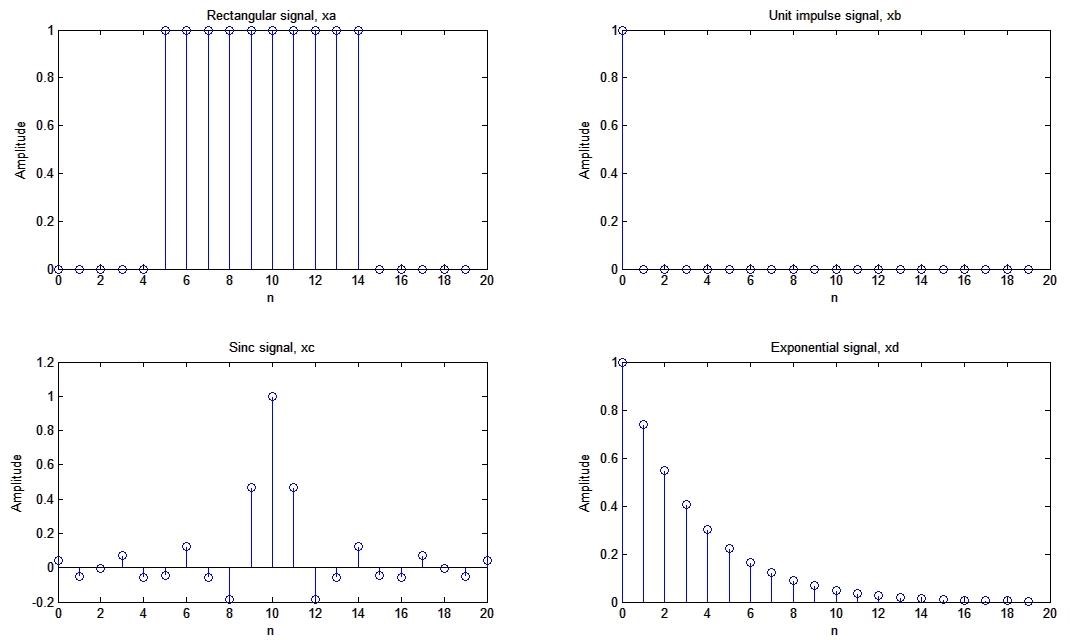
1. In *Exercise (b)*, which of the signals generated were **not compliant** with the Nyquist’s Sampling Theorem?

Nyquist's Sampling Theorem is violated by the signals 16Hz, 15Hz, and 13Hz. Because the fs>=2f criterion does not apply to them.

1. What is the maximum frequency of f1 for the sinusoidal signal x1 that we can adopt in **mylab1\_sig1.m**? What are the consequences if we exceed the maximum value? Relate your answer to the observation made in *Exercise (c)*.

The maximum frequency is limited to 10 hertz. If this value is surpassed, aliasing occured

3. Complete the code given below to generate common signals used in DSP as in **Figure 1**. Understand the coding and determine what type of signals generated (Rectangular/Unit impulse/Sinc/Exponential).



**Figure 1 – Common Signals Used in DSP**

|  |
| --- |
| % Common signals clear; % Signal xa  La = 20; %Signal length (no. of samples) na = [0:La-1]; %Signal sample index  xa = [zeros(1, La/4) ones(1,La/2), zeros(1,La/4)];  <ADD YOUR CODE HERE> %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 1 for  %plotting  stem(na, xa); %Plotting of signal  xlabel('n'); ylabel('Amplitude'); title('<Add Signal Type>, xa'); %Plotting of signal    % Signal xb  Lb = 20; %Signal length (no. of samples) nb = [0:Lb-1]; %Signal sample index xb = [1 zeros(1, Lb-1)]; %signal  <ADD YOUR CODE HERE> %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 2 for %plotting  <ADD YOUR CODE HERE> %Use stem %Plotting of signal with discrete samples  <ADD YOUR CODE HERE> %Use xlabel,ylabel,title %Plotting of signal – label your graph    % Signal xc  Lc = 21; %Signal length (no. of samples) nc = [0:Lc-1]; %Signal sample index  ang\_d = 0.2\*pi; %Phase angle per sampling period  n\_temp = [1:floor(Lc/2)]; x\_temp = sinc(ang\_d\*n\_temp); %Half side of signal x\_temp1 = fliplr(x\_temp); %Flip x\_temp left-right to generate the other half xc = [x\_temp1 1 x\_temp]; %combining both halves and 1 at  %centre  <ADD YOUR CODE HERE> %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 2 for %plotting  <ADD YOUR CODE HERE> %Use stem %Plotting of signal with discrete samples  <ADD YOUR CODE HERE> %Use xlabel,ylabel,title %Plotting of signal – label your graph    % Signal xd  Ld = 20; %Signal length (no. of samples) nd = [0:Ld-1]; %Signal sample index alpha = -0.3; %constant xd = exp(alpha\*nd); %signal |

<ADD YOUR CODE HERE> %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 2 for

%plotting

<ADD YOUR CODE HERE> %Use stem %Plotting of signal with discrete samples

<ADD YOUR CODE HERE> %Use xlabel,ylabel,title %Plotting of signal – label your graph

**Exercise**

% Common signals

clear;

%% Signal xa

La = 20; %Signanl length (no. of samples)

na = [0:La-1]; %Signal sample index

xa = [zeros(1, La/4) ones(1,La/2), zeros(1,La/4)];

figure(1);

subplot(2,2,1);

%plotting

stem(na, xa); %Plotting of signal

xlabel('n'); ylabel('Amplitude'); title('Rectangular signal, xa'); %Plotting of signal

%% Signal xb

Lb = 20; %Signal length (no. of samples)

nb = [0:Lb-1]; %Signal sample index

xb = [1 zeros(1, Lb-1)]; %signal

subplot(2,2,2); %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 2 for plotting

stem(nb, xb); %Plotting of signal

xlabel('n'); ylabel('Amplitude'); title('Unit impulse signal, xb'); %Plotting of signal

%% Signal xc

Lc = 21; %Signal length (no. of samples)

nc = [0:Lc-1]; %Signal sample index

ang\_d = 0.2\*pi; %Phase angle per sampling period

n\_temp = [1:floor(Lc/2)]; x\_temp = sinc(ang\_d\*n\_temp); %Half side of signal

x\_temp1 = fliplr(x\_temp); %Flip x\_temp left-right to generate the other half

xc = [x\_temp1 1 x\_temp]; %combining both halves and 1 at centre

subplot(2,2,3); %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 3 for plotting

stem(nc, xc); %Use stem %Plotting of signal with discrete samples

xlabel('n'); ylabel('Amplitude'); title('Sinc signal, xc'); %Plotting of signal

%% Signal xd

Ld = 20; %Signal length (no. of samples)

nd = [0:Ld-1]; %Signal sample index

alpha = -0.3; %constant

xd = exp(alpha\*nd); %signal

subplot(2,2,4);

stem(nd, xd);

xlabel('n');

ylabel('Amplitude');

title('Exponential signal, xd');

outputGraphical user interface, chart

Description automatically generated

# 2. Basic Signal Operations in Time Domain

1. Complete the code below to generate sinusoidal signals x2a and x2b, with frequencies f2a = 300 Hz and f2b = 500 Hz and sampling frequencies of fs2a = 5\*f2a and fs5b = 5\*f2b respectively. Each signal is to be 0.1 s in length.

## Exercise

a. List the code and plot the signals. Save 5-second versions of the signals into .wav files as ‘sig1.wav’ and ‘sig2.wav’. (Hint: Use the function **wavwrite***.* Type *help wavwrite* in the Command Window for more information on this function.)

|  |
| --- |
| % Lab 1, Section 2, Q1 clear;    % Signal x2a of 300Hz  <ADD YOUR CODE HERE>; %Freqeuncy of signal in Hz  <ADD YOUR CODE HERE>; %Sampling frequency for discrete-time signal  L2a = 0.1\*fs2a; %Signal length (no. of samples)- 0.1s t2a = [0:L2a-1]/fs2a; %Signal sampling instants x2a = cos(2\*pi\*f2a\*t2a); %Sinusoidal signal      % Signal x2b of 500Hz  <ADD YOUR CODE HERE>; %Freqeuncy of signal in Hz  <ADD YOUR CODE HERE>; %Sampling frequency for discrete-time signal  L2b = 0.1\*fs2b; %Signal length (no. of samples) – 0.1s t2b = [0:L2b-1]/fs2b; %Signal sampling instants x2b = cos(2\*pi\*f2b\*t2b); %Sinusoidal signal    %Plotting  <ADD YOUR CODE HERE> Plot x2a & x2b;Use subplot,stem,xlabel,ylabel,title;  %Wave files  % Signal x2a of 300Hz  <ADD YOUR CODE HERE>; %Freqeuncy of signal in Hz  <ADD YOUR CODE HERE>; %Sampling frequency for discrete-time signal  <ADD YOUR CODE HERE>; %Signal length (no. of samples)- change to 5 seconds t2a = [0:L2a-1]/fs2a; %Signal sampling instants x2a = cos(2\*pi\*f2a\*t2a); %Sinusoidal signal  <ADD YOUR CODE HERE>; %Save into wav file (‘sig1.wav’) using wavwrite function;use nbits=8    % Signal x2b of 500Hz  <ADD YOUR CODE HERE>; %Freqeuncy of signal in Hz  <ADD YOUR CODE HERE>; %Sampling frequency for discrete-time signal  <ADD YOUR CODE HERE>; %Signal length (no. of samples) - change to 5 seconds t2b = [0:L2b-1]/fs2b; %Signal sampling instants x2b = cos(2\*pi\*f2b\*t2b); %Sinusoidal signal  <ADD YOUR CODE HERE>; %Save into wav file (‘sig2.wav’) using wavwrite function; use nbits=8 |

1. Write the complete code to mix the two signals (x2a and x2b) above.

%% Signal x2a of 300Hz

f2a = 300 ; %Freqeuncy of signal in Hz

fs2a = 5\*f2a ; %Sampling frequency for discrete-time signal

L2a = 0.1\*fs2a; %Signal length (no. of samples)- 0.1s

t2a = [0:L2a-1]/fs2a; %Signal sampling instants

x2a = cos(2\*pi\*f2a\*t2a); %Sinusoidal signal

% Signal x2b of 500Hz

f2b = 500 ; %Freqeuncy of signal in Hz

fs2b = 5\*f2b; %Sampling frequency for discrete-time signal

L2b = 0.1\*fs2b; %Signal length (no. of samples) - 0.1s

t2b = [0:L2b-1]/fs2b; %Signal sampling instants

x2b = cos(2\*pi\*f2b\*t2b); %Sinusoidal signal

%Plotting

figure(1);

% Plotting xa

subplot(2,1,1);

stem(t2a,x2a);

xlabel('Time (s)');

ylabel('Amplitude');

title('xa signal');

% Plotting xb

subplot(2,1,2);

stem(t2b,x2b);

xlabel('Time (s)');

ylabel('Amplitude');

title('xb signal');

%% Signal x2a of 300Hz

f2a = 300 ; %Freqeuncy of signal in Hz

fs2a = 5\*f2a ; %Sampling frequency for discrete-time signal

L2a = 0.1\*fs2a; %Signal length (no. of samples)- 0.1s

t2a = [0:L2a-1]/fs2a; %Signal sampling instants

x2a = cos(2\*pi\*f2a\*t2a); %Sinusoidal signal

% Signal x2b of 500Hz

f2b = 500 ; %Freqeuncy of signal in Hz

fs2b = 5\*f2b; %Sampling frequency for discrete-time signal

L2b = 0.1\*fs2b; %Signal length (no. of samples) - 0.1s

t2b = [0:L2b-1]/fs2b; %Signal sampling instants

x2b = cos(2\*pi\*f2b\*t2b); %Sinusoidal signal

%% Signal x2a of 300Hz

f2a = 300 ; %Freqeuncy of signal in Hz

fs2a = 5\*f2a ; %Sampling frequency for discrete-time signal

L2a = 5\*fs2a; %Signal length (no. of samples)- 5 s

t2a = [0:L2a-1]/fs2a; %Signal sampling instants

x2a = cos(2\*pi\*f2a\*t2a); %Sinusoidal signal

wavwrite(x1b,fs1b,8,'hall.wav');

% Signal x2b of 500Hz

f2b = 500 ; %Freqeuncy of signal in Hz

fs2b = 5\*f2b; %Sampling frequency for discrete-time signal

L2b = 5\*fs2b; %Signal length (no. of samples) - 5 s

t2b = [0:L2b-1]/fs2b; %Signal sampling instants

x2b = cos(2\*pi\*f2b\*t2b); %Sinusoidal signal

wavwrite(x2b,fs2b,8,'man.wav');

output

Timeline

Description automatically generated with medium confidence

## Exercises

1. What are the two parameters of the signals that we need to adjust to be equivalent before we can mix them?

Sampled data, s1 and also sample rate data, Fs1

1. Coding below shows how to apply the necessary adjustments to the two signals stored in the .wav files above. Complete the code to read in the signals from the .wav files and mix the signals appropriately.

% Lab1, Section 2, Q2 clear;

% Read in signals

|  |
| --- |
| [s1, fs1, nbits1] = <ADD YOUR CODE HERE>; % Use wavread function [s2, fs2, nbits2] = <ADD YOUR CODE HERE>; % Use wavread function  % Sampling rate adjustment fs = lcm(fs1, fs2); if fs > fs1  fs\_fac1 = floor(fs / fs1);  s1\_fs = zeros(1, fs\_fac1\*length(s1)); s1\_fs(1:fs\_fac1:end) = s1; else  s1\_fs = s1; end if fs > fs2  fs\_fac2 = floor(fs / fs2);  s2\_fs = zeros(1, fs\_fac2\*length(s2)); s2\_fs(1:fs\_fac2:end) = s2; else  s2\_fs = s2; end    % Length adjustment  L1 = length(s1\_fs); L2 = length(s2\_fs); if L1 > L2  s2\_fs = [s2\_fs zeros(1, L1 - L2)]; else  s1\_fs = [s1\_fs zeros(1, L2 - L1)]; end    %Mixing of signals s\_m = s1\_fs + s2\_fs;    %Play back of sounds  <ADD YOUR CODE HERE>; %Play sound of signal 1 (s1 & fs1) – use sound function input('Press ...');  <ADD YOUR CODE HERE>; %Play sound of signal 2 (s2 & fs2) – use sound function input('Press ...');  <ADD YOUR CODE HERE>; %Play sound of mixed signal (s\_m & fs) – use sound function |

1. Listen to the playback of the output compared to the constituent signals. Has the operation achieved its purpose of signal mixing?

Yes .sound 1 and sound2 we here properly. only mixed sound we can not here properly.

**Code Output:**

[s1, fs1] = audioread('hall.wav');% Use wavread function

[s2, fs2] = audioread('man.wav'); % Use wavread function

% Sampling rate adjustment

fs = lcm(fs1, fs2);

if fs > fs1

fs\_fac1 = floor(fs / fs1);

s1\_fs = zeros(1, fs\_fac1\*numel(s1));

s1\_fs(1:fs\_fac1:end) = s1;

else

s1\_fs = s1;

end

if fs > fs2

fs\_fac2 = floor(fs / fs2);

s2\_fs = zeros(1, fs\_fac2\*length(s2));

s2\_fs(1:fs\_fac2:end) = s2;

else

s2\_fs = s2;

end

% Length adjustment

L1 = length(s1\_fs); L2 = length(s2\_fs);

if L1 > L2

s2\_fs = [s2\_fs zeros(1, L1 - L2)];

else

s1\_fs = [s1\_fs zeros(1, L2 - L1)];

end

%Mixing of signals

s\_m = s1\_fs + s2\_fs;

fs=fs1+fs2;

%Play back of sounds

sound(s1,fs1) %Play sound of signal 1 (s1 & fs1) – use sound function input('Press ...');

plot(s1)

sound(s2,fs2) %Play sound of signal 2 (s2 & fs2) – use sound function input('Press ...');

plot(s2)

sound(s\_m,fs); %Play sound of mixed signal (s\_m & fs) – use sound function

plot(s\_m)

**Matlab File:**



3. Signal convolutions are often performed in DSP systems. For example, the output signal y[n] of a system with the unit impulse response h[n] due to an input signal x[n] can be obtained from the convolution between x[n] and h[n].

## Exercises

1. Write a complete code to use the function **conv** to generate the convolution between x[n] and h[n].

x[n] and h[n] are rectangular signals and sinc signals respectively (refer Section 1 Q3). Type *help conv* in the Command Window for more information on this function.

|  |
| --- |
| %Lab 1, Section 2, Q3a    % Rectangular signal  <ADD YOUR CODE HERE>; % Refer to the code given earlier  %Plotting of signal  <ADD YOUR CODE HERE>; % Use subplot, stem, xlabel, ylabel, title function    % Sinc signal  <ADD YOUR CODE HERE>; % Refer to the code given earlier  %Plotting of signal  <ADD YOUR CODE HERE>; % Use subplot, stem, xlabel, ylabel, title function  %Convolution between x[n] and h[n]  y = <ADD YOUR CODE HERE>; % Use conv function L\_y = length(y); n\_y = [0:L\_y-1]; %Plotting of signal  <ADD YOUR CODE HERE>; % Use subplot, stem, xlabel, ylabel, title function |

% Common signals

clear;

%% Signal xa

La = 20; %Signanl length (no. of samples)

na = [0:La-1]; %Signal sample index

xa = [zeros(1, La/4) ones(1,La/2), zeros(1,La/4)];

figure(1);

subplot(2,2,1);

%plotting

stem(na, xa ,'filled'); %Plotting of signal

xlabel('n'); ylabel('Amplitude'); title('Rectangular signal, xa'); %Plotting of signal

%% Signal xc

Lc = 21; %Signal length (no. of samples)

nc = [0:Lc-1]; %Signal sample index

ang\_d = 0.2\*pi; %Phase angle per sampling period

n\_temp = [1:floor(Lc/2)]; x\_temp = sinc(ang\_d\*n\_temp); %Half side of signal

x\_temp1 = fliplr(x\_temp); %Flip x\_temp left-right to generate the other half

xc = [x\_temp1 1 x\_temp]; %combining both halves and 1 at centre

subplot(2,2,2); %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 3 for plotting

stem(nc, xc ,'filled'); %Use stem %Plotting of signal with discrete samples

xlabel('n'); ylabel('Amplitude'); title('Sinc signal, xc'); %Plotting of signal

%Convolution between x[n] and h[n]

y = conv(xa,xc);

tc = linspace(-5,5,length(y)); %Time vector for convoluted signal

subplot(2,2,3)

stem(tc,y,'filled')

xlabel('Time')

ylabel('Covolution Result y')

title('Signal y=x[n]\*h[n]')

Chart, scatter chart

Description automatically generated

1. What is the length of the convolution output? How is the convolution output length L\_y related to input signal length L\_x and the system impulse response length L\_h?

The length of the convolution output is double to length of the signal x(n) and h(n) - 1. The Output length of Convoluted output = L+M-1

**Where L = length of x(n)**

**M = length of h(n)**

**Here above L = 20**

**M = 21**

**So Length of y = 20 + 21 - 1 = 40**

1. What is the convolution output for unit impulse signals and sinc signals? What can we conclude about unit impulses in convolution?

**We see above that impulse function when convoluted to sinc function produces same sinc function output. In convolution when input signal is impulse function the output is same as of  h(n).**

%Lab 1, Section 2, Q3c

% Unit impulse signal

|  |
| --- |
| <ADD YOUR CODE HERE>; % Refer to the code given earlier  %Plotting of signal  <ADD YOUR CODE HERE>; % Use subplot, stem, xlabel, ylabel, title function  % Sinc signal  <ADD YOUR CODE HERE>; % Refer to the code given earlier  %Plotting of signal  <ADD YOUR CODE HERE>; % Use subplot, stem, xlabel, ylabel, title function  %Convolution between x[n] and h[n]  y = <ADD YOUR CODE HERE>; % Use conv function L\_y = length(y); n\_y = [0:L\_y-1]; %Plotting of signal  <ADD YOUR CODE HERE>; % Use subplot, stem, xlabel, ylabel, title function |

% Common signals

clear;

%% Signal xb

Lb = 20; %Signal length (no. of samples)

nb = [0:Lb-1]; %Signal sample index

xb = [1 zeros(1, Lb-1)]; %signal

subplot(2,2,1); %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 2 for plotting

stem(nb, xb ,'filled'); %Plotting of signal

xlabel('n'); ylabel('Amplitude'); title('Unit impulse signal, xb'); %Plotting of signal

%% Signal xc

Lc = 21; %Signal length (no. of samples)

nc = [0:Lc-1]; %Signal sample index

ang\_d = 0.2\*pi; %Phase angle per sampling period

n\_temp = [1:floor(Lc/2)]; x\_temp = sinc(ang\_d\*n\_temp); %Half side of signal

x\_temp1 = fliplr(x\_temp); %Flip x\_temp left-right to generate the other half

xc = [x\_temp1 1 x\_temp]; %combining both halves and 1 at centre

subplot(2,2,2); %Use subplot %Sub-divide figure into 2X2 = 4 plots and set plot 3 for plotting

stem(nc, xc ,'filled'); %Use stem %Plotting of signal with discrete samples

xlabel('n'); ylabel('Amplitude'); title('Sinc signal, xc'); %Plotting of signal

%Convolution between x[n] and h[n]

y = conv(xb,xc);

tc = linspace(-5,5,length(y)); %Time vector for convoluted signal

subplot(2,2,3)

stem(tc,y,'filled')

xlabel('Time')

ylabel('Covolution Result y')

title('Signal y=x[n]\*h[n]')

output

Graphical user interface, chart

Description automatically generated

1. Copy the ‘hall.wav’ and ‘man.wav’ file to working directory. Try out below coding at your command window. Listen to the sounds and explain what happens. Any differences with 11250, 22500 and 44100 Hz?

**With 11250Hz**

**Both the sounds produce sound waves in which hall.wav is a small blurry sound and man.wav produces long unclear sound.**

**With 11250 Hz, both the sounds are blurry and unclear sounds. Output sound is only an echoed version of  fast blowing air**

**With 22500 Hz,**

**the hall sound becomes clear to some extent , and man sound can be heard but the words of man are unclear. The man's voice is heavier due to the small sampling frequency used. With this sampling frequency the output sound is like a long echoes sound of fast blowing air.**

**With 44100 Hz**

**as fs is increased the voice becomes more clear and sharp. The man voice is perfectly clear and all the words said by a man in this sound can been heard easily and the hall voice is also perfect and clear. Hall sound is an echo sound of air in hall, which makes the man sound echoed after convolution. The output voice after convolution can be heard clearly with this sampling frequency.**

%Lab 1, Section 2, Q3d

hall=wavread('hall.wav'); plot(hall); sound(hall,11250) in=wavread('man.wav'); plot(in); sound(in,11250) out=conv2(in,hall); plot(out); sound(out,11250)



hall.wav man.wav

**11250Hz**

%Lab 1, Section 2, Q3d

hall=audioread('hall.wav');

subplot(2,2,1);

plot(hall);

title('hall sound plot')

sound(hall,11250)

in=audioread('man.wav');

subplot(2,2,2);

plot(in);

title('man sound plot')

sound(in,11250)

out=conv2(in,hall);

subplot(2,2,3);

plot(out);

title('After conv')

sound(out,11250)

**22500Hz**

%Lab 1, Section 2, Q3d

hall=audioread('hall.wav');

subplot(2,2,1);

plot(hall);

title('hall sound plot')

sound(hall,22500)

in=audioread('man.wav');

subplot(2,2,2);

plot(in);

title('man sound plot')

sound(in,22500)

out=conv2(in,hall);

subplot(2,2,3);

plot(out);

title('after conv sound plot')

sound(out,22500)

**44100 Hz**

%Lab 1, Section 2, Q3d

hall=audioread('hall.wav');

subplot(2,2,1);

plot(hall);

title('hall sound plot')

sound(hall,44100)

in=audioread('man.wav');

subplot(2,2,2);

plot(in);

title('man sound plot')

sound(in,44100)

out=conv2(in,hall);

subplot(2,2,3);

plot(out);

title('after conv sound plot')

sound(out,44100)

Chart

Description automatically generated

**5.0 EXPERIMENT RESULT**

*(Please attached all the graphs obtained from the Procedure)*

# Familiarizing with Simulation Tools

**section1 mylab sig1**

Chart

Description automatically generated

**section1 question1(b)**

Change the **frequency** of x1 to 17 Hz. What is the signal plot that you observe

Timeline

Description automatically generated

Observe the signal plots for frequencies such as 4 Hz, 5 Hz, 7 Hz, 16 Hz, 15 Hz, and 13 Hz. What can you conclude from the observation?

Chart, diagram

Description automatically generated

**section1 question3**

**Graphical user interface, chart, histogram

Description automatically generated**

**Section2\_q1**

Timeline

Description automatically generated with medium confidence

**Section-2 question3(a)**

Chart, box and whisker chart

Description automatically generated

**Section-2 question3(c)**

Chart

Description automatically generated

**Section-2 question3(d)**

For 11250 Hz

Chart

Description automatically generated

For 22500 Hz

Graphical user interface, chart

Description automatically generated

For 44100 Hz

Chart

Description automatically generated

## 6.0 QUESTION & DISCUSSION

Answer all Questions in Procedure (Exercise) and discuss each task based on Experiment Data and Experiment Result.

1. **What is the sampling period used in generating the signal x1 in mylab1\_sig1.m?**

The sampling period used in generating the signal x1 is 0.05 seconds. As we can see that t1 variable is used in generating x1 signal:

**fs1** = **20**; %Sampling frequency for discrete-time signal

**t1** = [0: L1-1]/fs1; %Signal sampling instants

**x1** = cos(2\*pi\***f1**\*t1); %Sinusoidal signal x1

So, the step size in t1 will be 1 but by dividing by fs(i.e. 20) will be 0.05 seconds. Thus, the sampling period will be 0.05 seconds.

1. **Change the frequency of x1 to 17 Hz. What is the signal plot that you observe? Observe the signal plots for frequencies such as 4 Hz, 5 Hz, 7 Hz, 16 Hz, 15 Hz, and 13 Hz. What can you conclude from the observation?**

Changing the frequency of x1 from 3 Hz to 17 Hz makes no change in the signal, because we are getting aliasing here. So, the two signals will be considered as having same frequency while they are having different. This is because the Nyquist criteria(fs>=2f) is not followed here.

Timeline, box and whisker chart

Description automatically generated

Changing the frequency to 4 Hz, 5 Hz, 7 Hz doesn't yield any problem because the Nyquist criteria is not disobeyed here but it is violated for 16 Hz, 15 Hz, and 13 Hz(fs is not 2 times of signal original frequency).

So, the signals we will be having for 16 Hz, 15 Hz, and 13 Hz will not be at their original frequencies (as observable from the output).

Instead, a pair of signals(4Hz&16Hz, 5Hz&15Hz, 7Hz&13Hz) at different frequencies will appear at only one frequency as observable below.

Chart, histogram

Description automatically generated

1. **In *Exercise (b)*, which of the signals generated were not compliant with the Nyquist’s Sampling Theorem?**

The signals 16Hz, 15Hz and 13Hz are not compliant with Nyquist's Sampling Theorem. As fs>=2f criteria doesn't hold for them.

The 3Hz signal looks same as 7Hz signal, 16Hz signal appears to be same as 4Hz signal, 15Hz is same as 5Hz signal while 13 Hz signal appears to same as 7 Hz signal.

1. **What is the maximum frequency of f1 for the sinusoidal signal x1 that we can adopt in mylab1\_sig1.m? What are the consequences if we exceed the maximum value? Relate your answer to the observation made in *Exercise (c)*.**

The maximum frequency can only be 10 HZ.

If exceeded than this, the same effect will happen (i.e., signals at different frequencies will be appearing to have the same frequency.

1. **Complete the code given below to generate common signals used in DSP as in Figure 1. Understand the coding and determine what type of signals generated (Rectangular/Unit impulse/Sinc/Exponential).**

Sgnal xa

We can see from line **xa = [zeros(1, La/4) ones(1,La/2), zeros(1,La/4)];** that first five entries of the signal are zero, ten entries are ones while remaining five entries will still remain zero. Thus it will be a rectangular plot.

Signal xb

We can see from line **xb = [1 zeros(1, Lb-1)];** that only first entry will be one while remaining will be zero. So, we will be having only a step signal.

Signal xc

From line **xc = [x\_temp1 1 x\_temp]; %combining both halves and 1 at centre**, it is observable that 1 will be at the origin while on left as well as right hand, we will be having a flipped signal.

Signal Xd

This is the easiest case of all, we can easily conclude that it is going to an exponential signal from line **xd = exp(alpha\*nd); %signal**.

## 7.0 CONCLUSION

From this lab we learn to use simulation tools, MATLAB, in signal analysis.

And understand basic digital signal time characteristics and operations.