Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Report

on

"Lab Works"

COMP 407

(For partial fulfillment of 4th Year/ 1st Semester in Computer Engineering)

Submitted by:

Babin Khatri (Roll No. 26)

Submitted to:

Mr. Satyendra Nath Lohani

Department of Computer Science and Engineering

Submission Date:

17th November, 2022

Table of Contents

Lab 1 : Introduction to Octave

1.1 Introduction

Octave is an interactive programming language specifically suited for vectorizable numerical calculations. It provides a high level interface to many standard libraries of numerical mathematics, e.g. LAPACK or BLAS.

The syntax of Octave resembles that of Matlab. An Octave program usually runs unmodified on Matlab. Matlab, being commercial software, has a larger function set, and so the reverse does not always work, especially when the program makes use of specialized add-on toolboxes for Matlab

1.2 Implementation

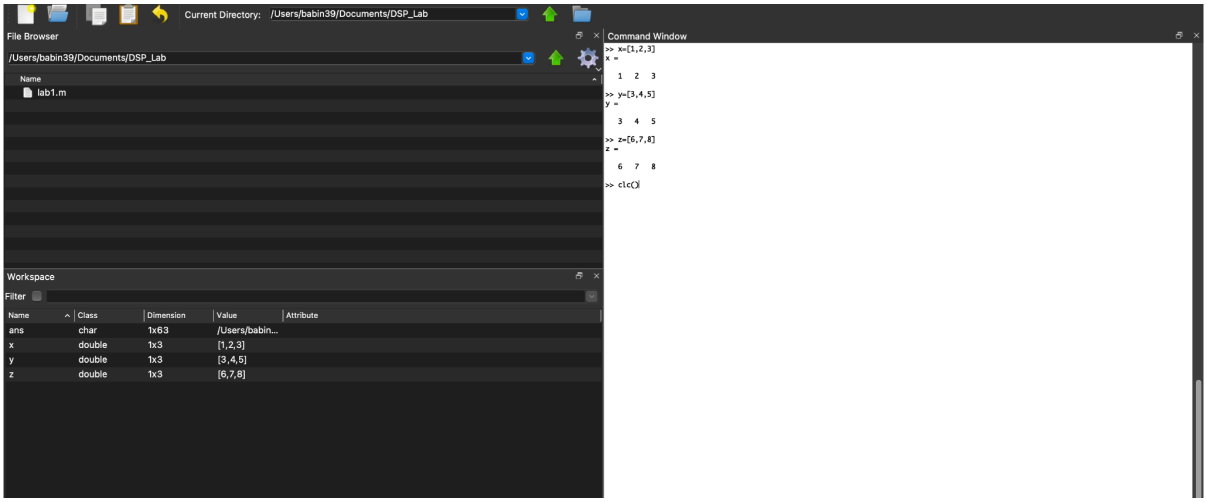
1.2.1 To study important commands of MATLAB software

- clc, close, xlabel, ylabel, zlabel, title, figure, subplot, linspace, stem, bar, plot, Colon Operator.

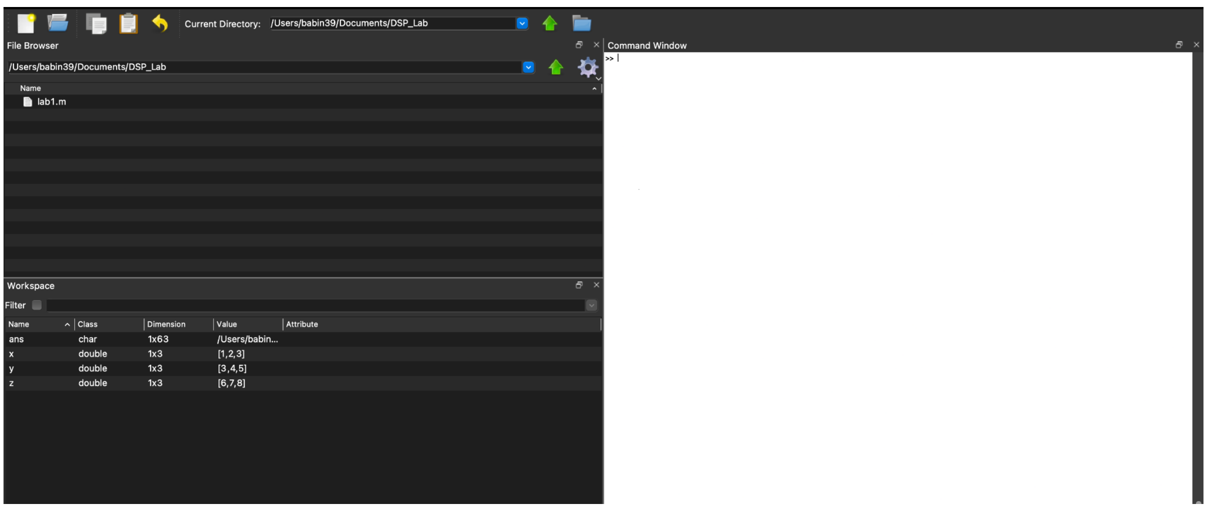
1.2.1.1. clc

It clears the terminal screen and moves the cursor to the upper left corner.

Suppose we have a command window with the following variables then if we want to clear the command window we use clc()



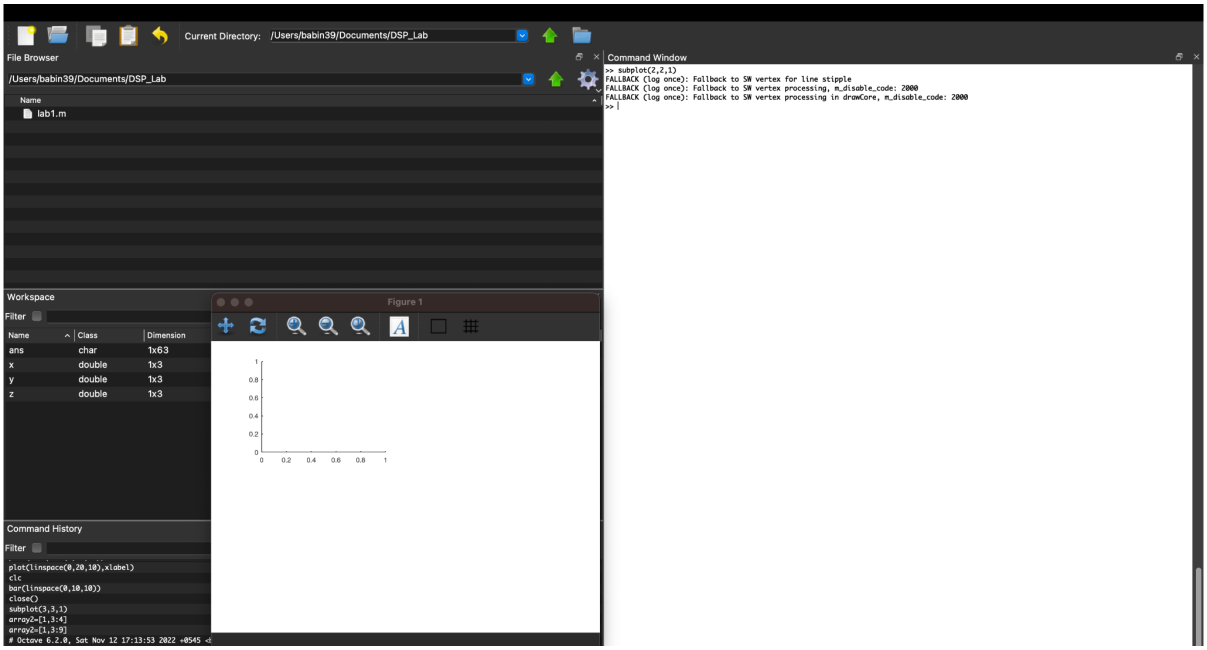
When clc() is entered the window will be as shown below:



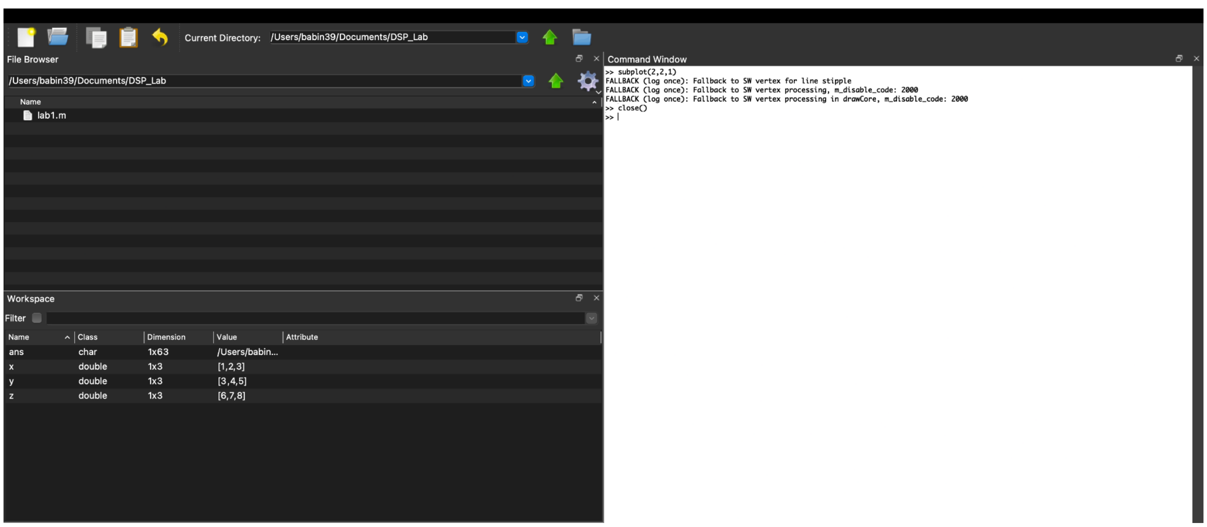
1.2.1.2. close

It closes the figure window(s).

If we have a figure window open then we can use close() to close the figure window as shown below:

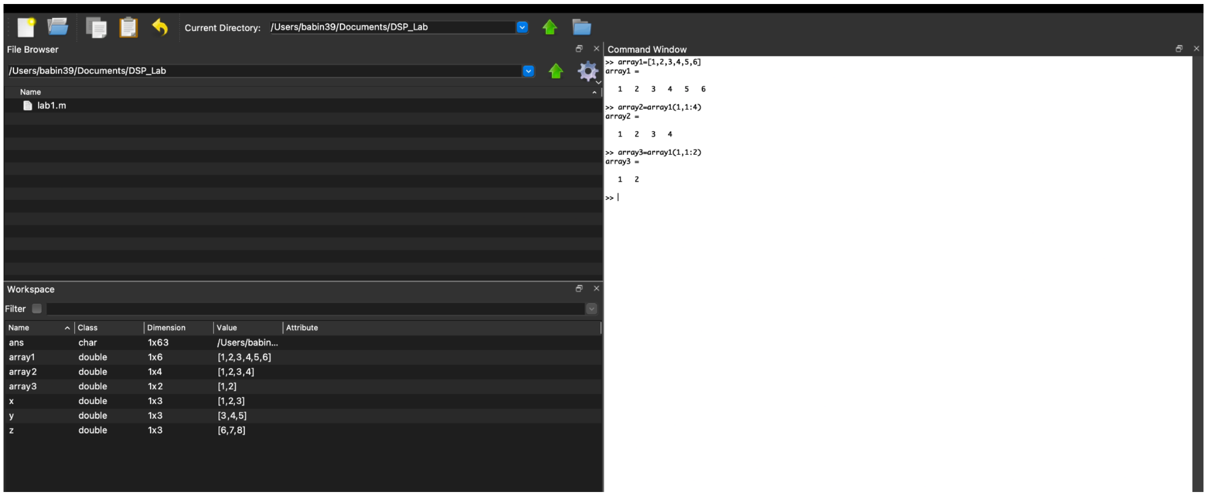


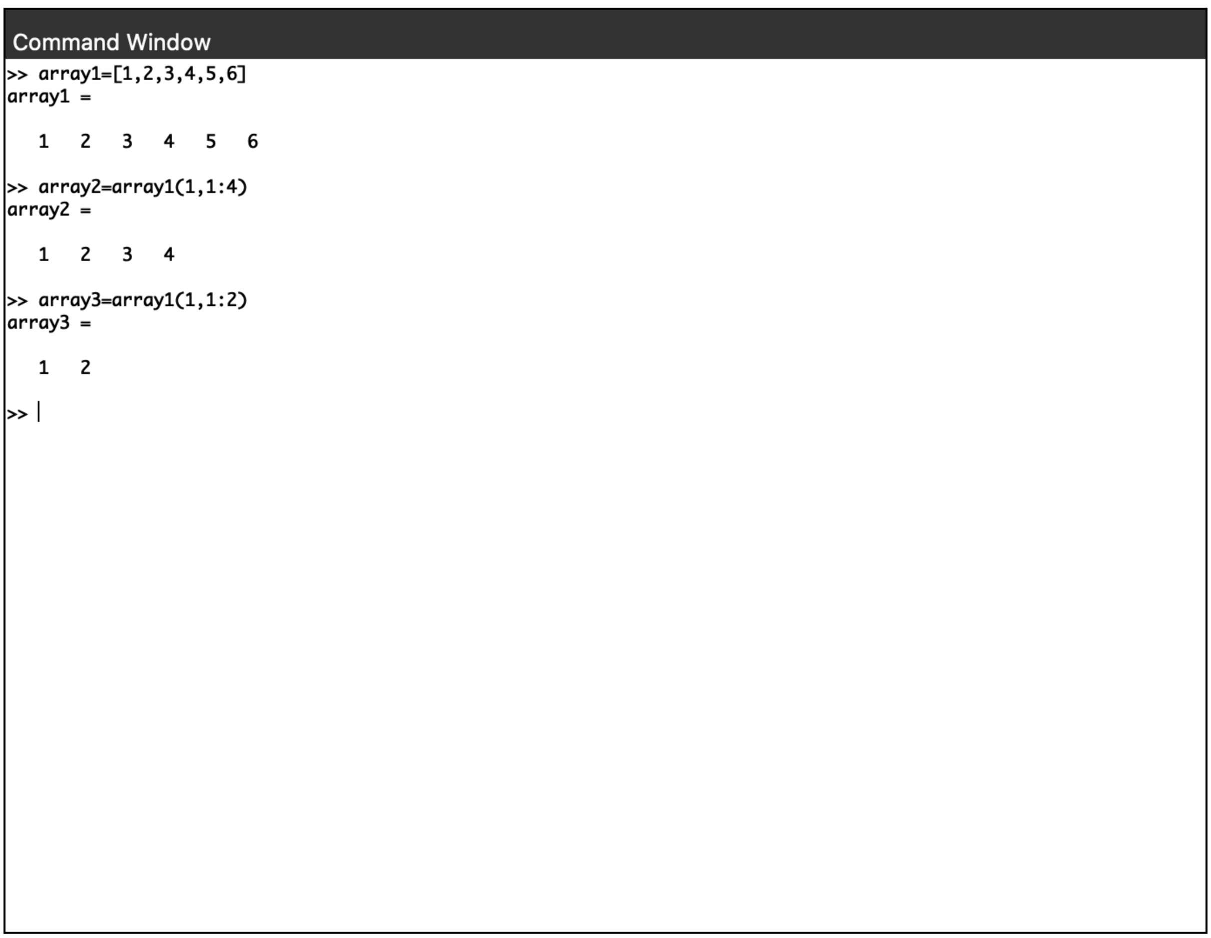
After using close()



1.2.1.3. Colon Operator

The colon : tells Octave to create a (row) vector of numbers starting from the first number and counting up to (and including) the second number, incrementing by 1 each time.





For the remaining commands we created a file named lab1.m and written the commands as shown below:

```
%%lab1

%% Study important commands

x=-2:0.5:2;
y=x.^2;


subplot(2,2,1)
plot(x,y)
title("Plot 1")
xlabel("x-axis")
ylabel("y-axis")

subplot(2,2,2)
z=linspace(1,5,5)
x=linspace(0,10,5)
stem(x,z)
title("Plot 2")
xlabel("x-axis")
ylabel("y-axis")

subplot(2,2,3)
z=linspace(1,5,5)
x=linspace(0,10,5)
bar(x,z)
title("Plot 3")
xlabel("x-axis")
ylabel("y-axis")


subplot(2,2,4)
z = [0:0.05:5];
plot3 (cos (2*pi*z), sin (2*pi*z), z);
legend ("helix");

xlabel("x-axis")
ylabel("y-axis")
zlabel("z-axis")
title ("plot3() of a helix");
```

The output obtained after execution is as shown below:

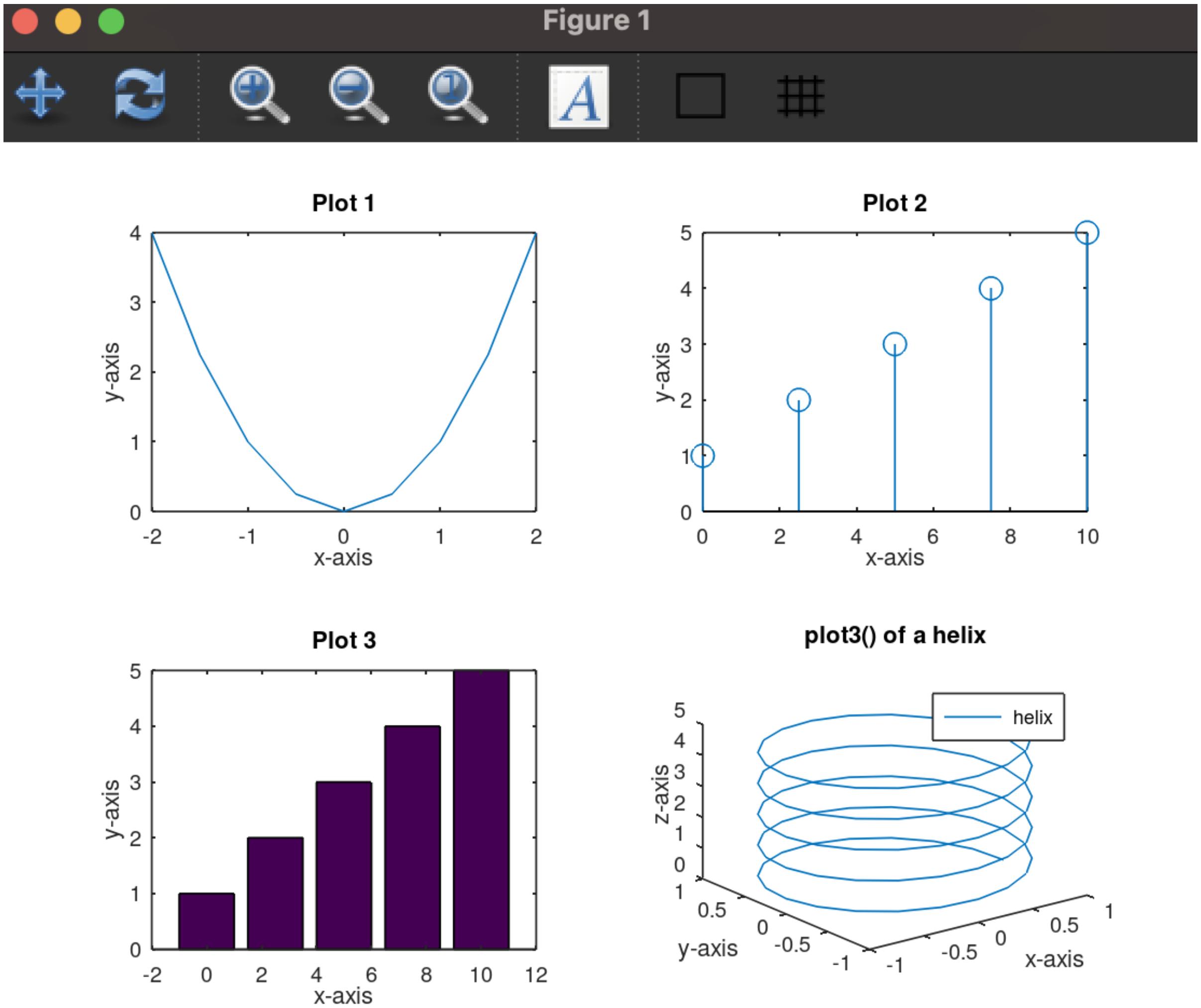


Figure 1
Plot 1
y-axis
x-axis
Plot 2
y-axis
x-axis
Plot 3
y-axis
x-axis
plot3() of a helix
helix
z-axis
y-axis
x-axis

1.2.2. Familiarization with the Octave environment.

- Create a matrix, A of size 3*4. Create another matrix, B of size 4*3.

```
%% Familiarization with MATLAB environment.

%% Matrix A of 3 x 4
A=[1,2,3,4;
  5,6,7,8;
  9,10,11,12]

%% Matrix B of 4 x 3
B=[1,2,3;
  5,6,7;
  9,10,11;
  4,8,12
  ]
```

- Add Matrix A and B. Subtract A from B.

```
>> A + B
error: operator +: nonconformant arguments (op1 is 3x4, op2 is 4x3)
>>
```

```
>> A - B
error: operator -: nonconformant arguments (op1 is 3x4, op2 is 4x3)
>>
```

- Multiply A and B. Multiply B and A [Errors Reason ?].

```
>> A * B
ans =

    54    76    98
   130   180   230
   206   284   362

>>
```

```
>> B * A
ans =

    38    44    50    56
    98   116   134   152
   158   188   218   248
   152   176   200   224

>>
```

- Transpose matrix A and B. Multiply the transposed matrices.

Filter

Name	Class	Dimension	Value	Attribute
A	double	3x4	[1,2,3,4; 5,6,...	
B	double	4x3	[1,2,3; 5,6,7;...	
ans	double	4x4	[38,98,158,...	
transposeA	double	4x3	[1,5,9; 2,6,1...	
transposeB	double	3x4	[1,5,9,4; 2,6,...	
x	double	1x5	[0,2.5000,5,...	
y	double	1x9	[4,2.2500,1,...	
z	double	1x101	[0,0.050000...	

```
>> transposeA=A'
transposeA =

     1     5     9
     2     6    10
     3     7    11
     4     8    12

>> transposeB= B'
transposeB =

     1     5     9     4
     2     6    10     8
     3     7    11    12

>> transposeA * transposeB
ans =

    38    98   158   152
    44   116   188   176
    50   134   218   200
    56   152   248   224

>>
```

```
>> transposeA=A'
transposeA =

     1     5     9
     2     6    10
     3     7    11
     4     8    12

>> transposeB= B'
transposeB =

     1     5     9     4
     2     6    10     8
     3     7    11    12

>> transposeA * transposeB
ans =

    38    98   158   152
    44   116   188   176
    50   134   218   200
    56   152   248   224

>>
```

Lab 2 : Signal

2.1 Introduction

The signal is a detectable quantity that contains specified information in it. In our perspective i.e., in engineering, it is a description of how one parameter is varying with the change in another parameter. There are some elementary signals that feature regularly in the analysis of signals and systems. They are namely, impulse signal, unit step signal, sinusoidal signal, exponential signal, ramp signal, and parabolic signal. Along with these, we use some signals derived from these basic signals like sinc.

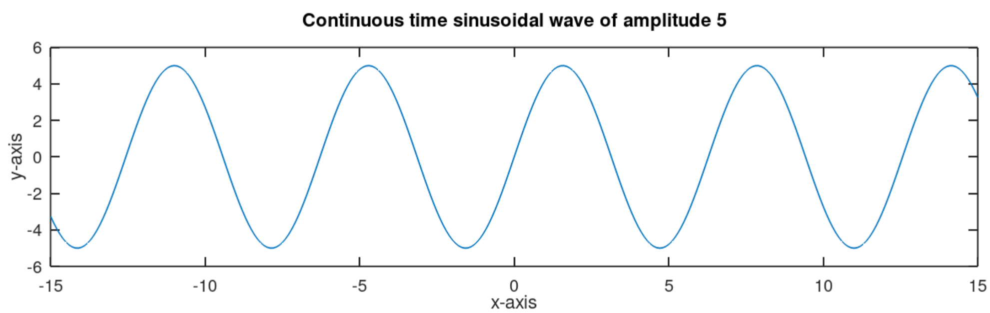
2.2 Implementation

- Generate a continuous time sinusoidal wave of amplitude 5.

Source Code:

```
%Generate a continuous time sinusoidal wave of amplitude 5.
x=[-15:0.1:15];
y=5*sin(x);
subplot(4,2,1)
plot(x,y);
title('Continuous time sinusoidal wave of amplitude 5');
xlabel('x-axis');
ylabel('y-axis');
```

Output:

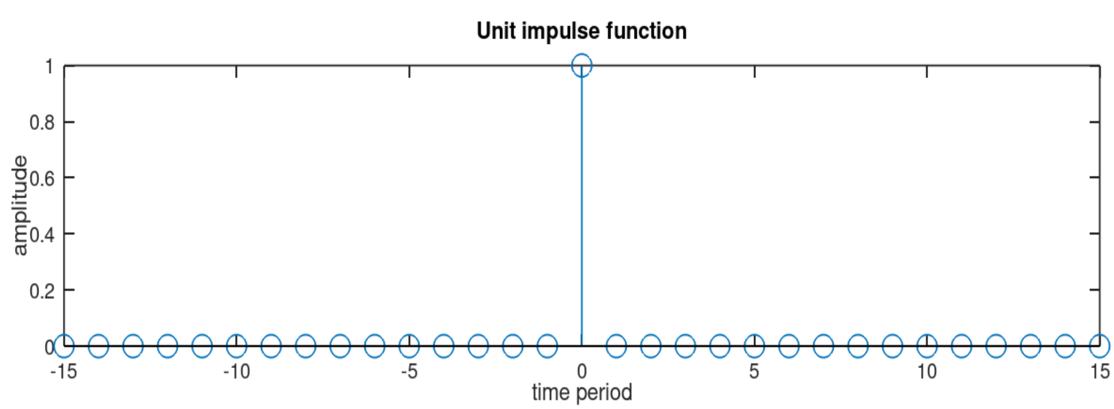


- Generate a unit impulse function.

Source Code:

```
%Generate a unit impulse function
time_period = [-15:1:15];
amplitude = [zeros(1,15), ones(1,1), zeros(1,15)];
subplot(4,2,2)
stem(time_period,amplitude);
title('Unit impulse function');
xlabel('time period');
ylabel('amplitude');
```

Output:

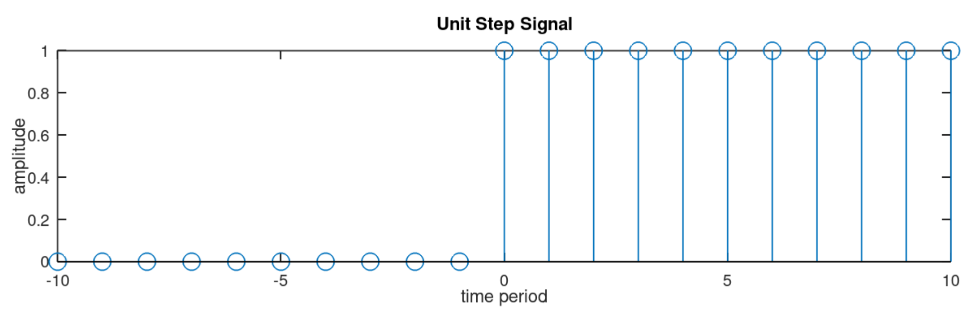


- Generate a unit step function.

Source Code:

```
%Generate a unit step function.

stepx=[-10:1:10];
stepy= stepx>=0;
subplot(4,2,3)
stem(stepx,stepy);
title('Unit Step Signal');
xlabel('time period');
ylabel('amplitude');
```

Output:

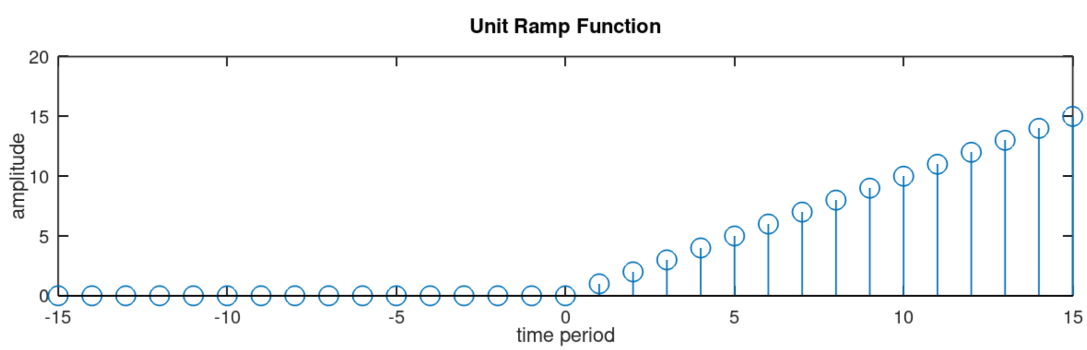


- Generate a unit ramp function

Source Code:

```
%Generate a unit ramp function.
rampx=[-15:1:15];
rampy= (rampx>=0).*rampx;
subplot(4,2,4)
stem(rampx,rampy);
title('Unit Ramp Function');
xlabel('time period');
ylabel('amplitude')
```

Output:

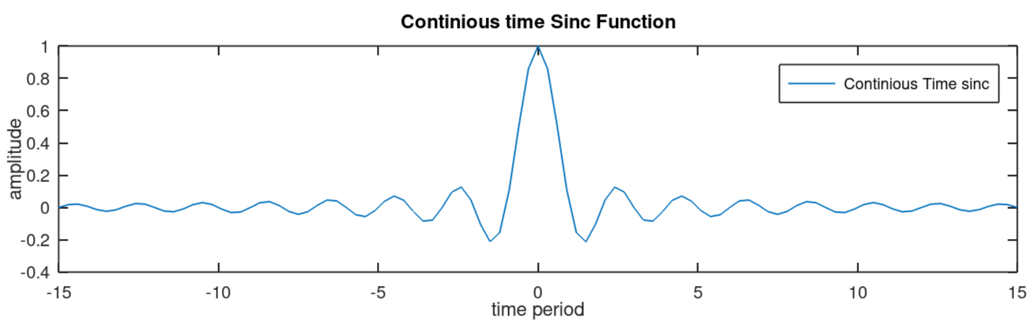


- Generate a continuous time sinc function

Source Code:

```
%Generate a continuous time sinc function
X = [-15:0.3:15];
y = sinc(X);
subplot(4,2,5)
plot(X,y);
title('Continious time Sinc Function');
xlabel('time period');
ylabel('amplitude');
legend('Continious Time sinc');
```

Output:

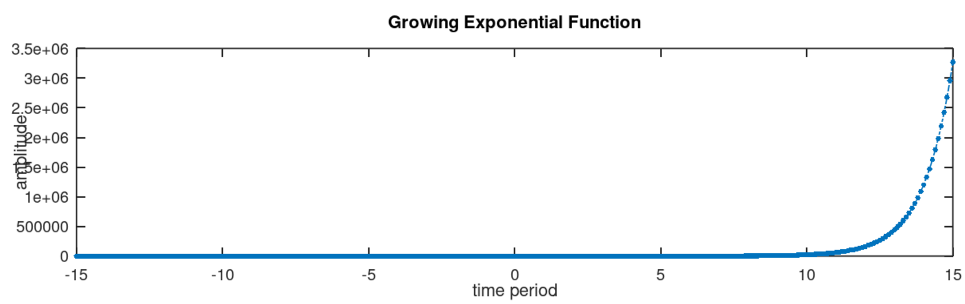


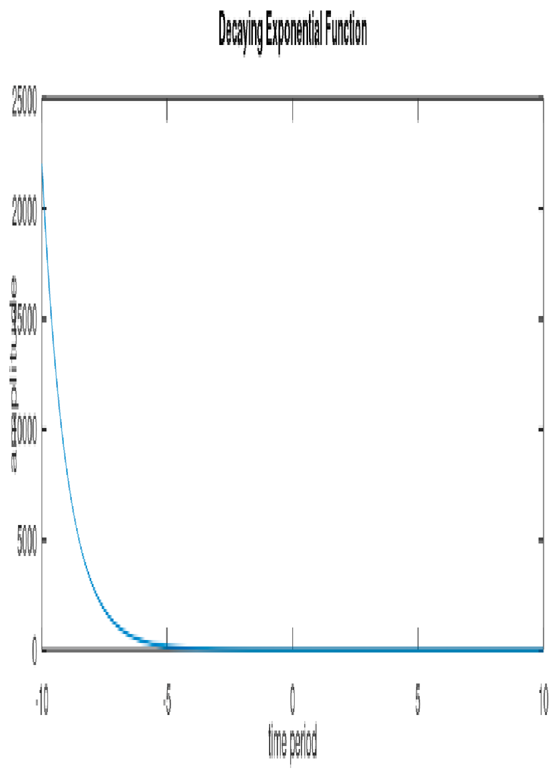
- Generate a continuous time exponential (growing, decaying, DC signal)

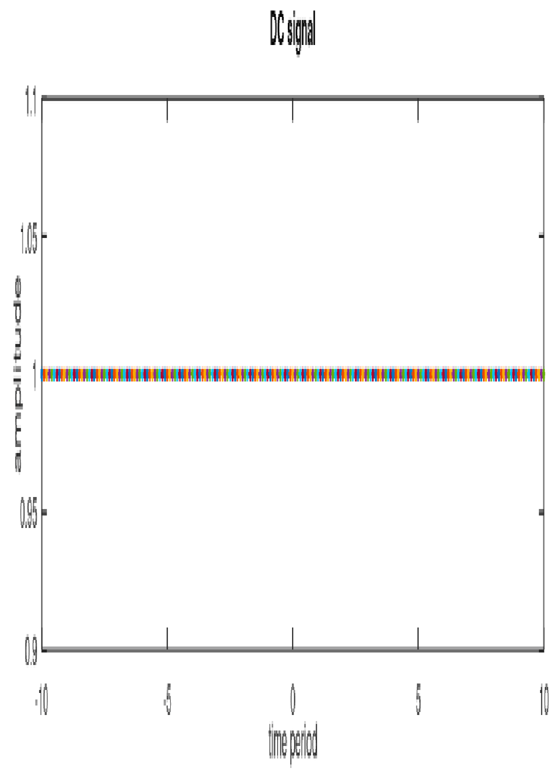
Source Code

```
%Generate a continuous time exponential(growing, decaying, DC signal)

%growing signal
subplot(4,2,6);
x=[-15:0.1:15];
y=exp(x);
plot(x,y,'.-.');
title('Growing Exponential Function');
xlabel('time period');
ylabel('amplitude');

%decaying signal
subplot(4,2,7);
x=[-10:0.1:10];
y=exp(-x);
plot(x,y);
title('Decaying Exponential Function');
xlabel('time period');
ylabel('amplitude');

%DC Signal
subplot(4,2,8);
x=[-10:0.1:10];
y=1;
plot(x,y,'.-.');
title('DC signal');
xlabel('time period');
ylabel('amplitude');
```

Output:




Decaying Exponential Function
amplitude
time period
25000
20000
15000
10000
5000
0
-10
-5
0
5
10

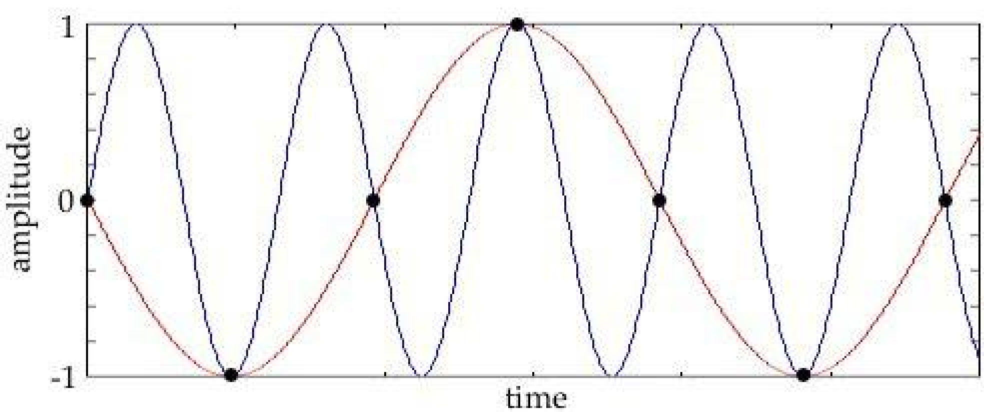

DC signal
amplitude
time period
1.1
1.05
1
0.95
0.9
-10
-5
0
5
10

Lab 3: Sampling of Signal

3.1 Introduction

3.1.1 Sampling

Sampling is the process of recording an analog signal, such as a continuous time sinusoid and converting into a discrete time sinusoid (digital). The way the signal is recorded differs depending on the type of analog signal (sound, pressure, light, etc..). Sampling theory tells us how to sample a signal. For example Nyquist's theorem tells us that the rate at which we sample an analog signal must be at least twice the maximum frequency of the sample in order to reconstruct the same signal. If the sample rate is too low then there might be multiple frequencies of continuous time sinusoids that match our discrete time sinusoid. This is called aliasing.



In the image shown above, the dots represent samples of a continuous time sinusoid over time

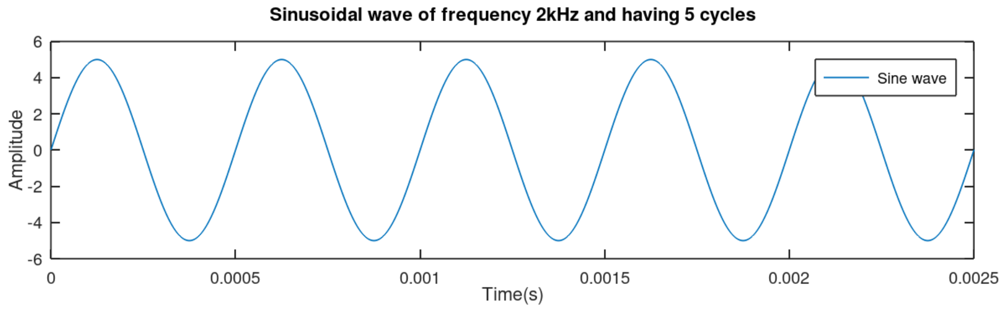
3.2 Implementation

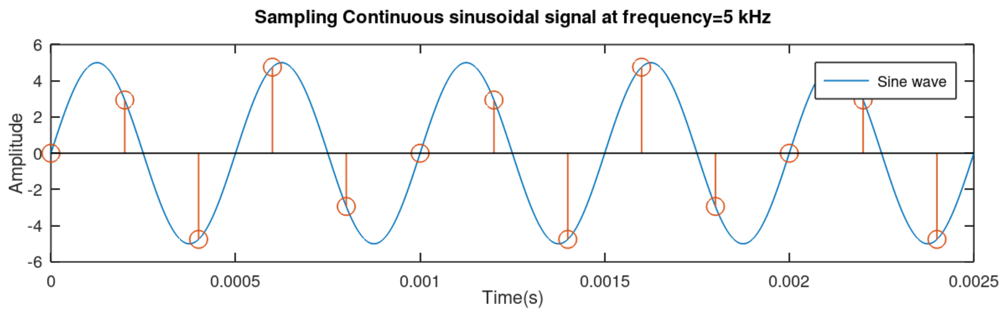
3.2.1 Generate the signal x = 5sin(2 pi f t) with 5 cycles, where f = 2 kHz signal and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz. (Title and label each figure)

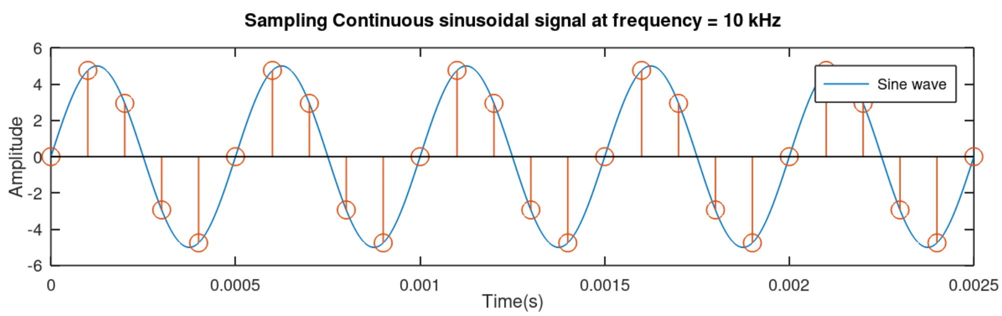
Source Code:

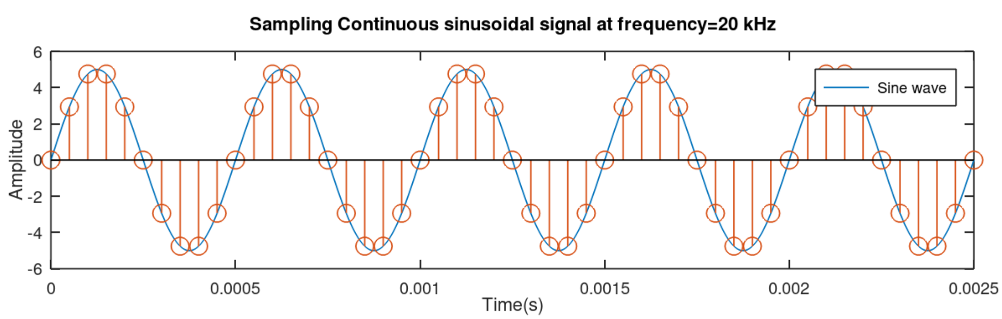
```
%Lab 3

%Question 1: Generate the signal x = 5sin(2 pi f t) with 5 cycles, where f = 2 kHz signal and sample the signal with
%frequency 5 KHz, 10 Khz, 20 KHz. (Title and label each figure)

cycles=5;
f=2000
t=0:0.000001:cycles*1/f;
x = 5*sin(2*pi*f*t);
subplot(4,2,1)
plot(t,x);
title('Sinusoidal wave of frequency 2kHz and having 5 cycles');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Sine wave');

%Sampling the signal with frequency 5 KHz
srate1=5000;
t1=0:1/srate1:cycles*1/f;
x1 = 5*sin(2*pi*f*t1);
subplot(4,2,2)
plot(t,x);
hold on
stem(t1,x1);
title('Sampling Continuous sinusoidal signal at frequency=5 kHz');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Sine wave');

%Sampling the signal with frequency 10 KHz
fs2=10000;
t2=0:1/fs2:cycles*1/f;
x2=5*sin(2*pi*f*t2);
subplot(4,2,3)
plot(t,x);
hold on
stem(t2,x2);
title('Sampling Continuous sinusoidal signal at frequency = 10 kHz');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Sine wave');

%Sampling the signal with frequency 20 KHz
srate3=20000
t3=0:1/srate3:cycles*1/f;
x3 = 5*sin(2*pi*f*t3);
subplot(4,2,4)
plot(t,x);
hold on
stem(t3,x3);
title('Sampling Continuous sinusoidal signal at frequency=20 kHz');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Sine wave');
```

Output:


Sinusoidal wave of frequency 2kHz and having 5 cycles
Amplitude
Time(s)
Sine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002
0.0025


Sampling Continuous sinusoidal signal at frequency=5 kHz
Amplitude
Time(s)
Sine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002
0.0025


Sampling Continuous sinusoidal signal at frequency = 10 kHz
Amplitude
Time(s)
Sine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002
0.0025

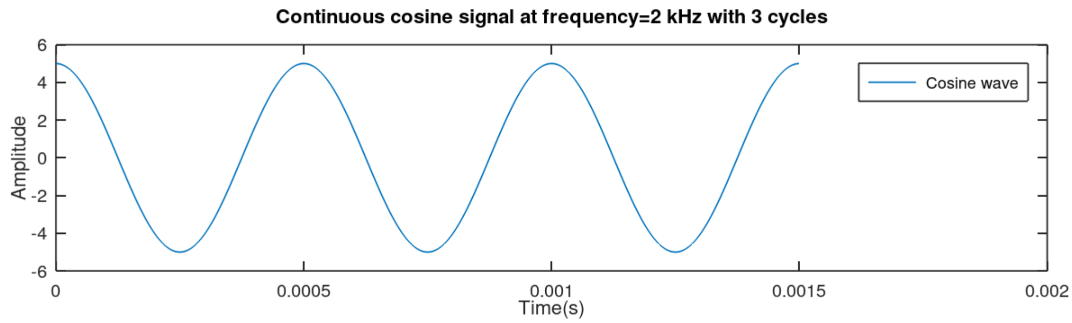

Sampling Continuous sinusoidal signal at frequency=20 kHz
Amplitude
Time(s)
Sine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002
0.0025

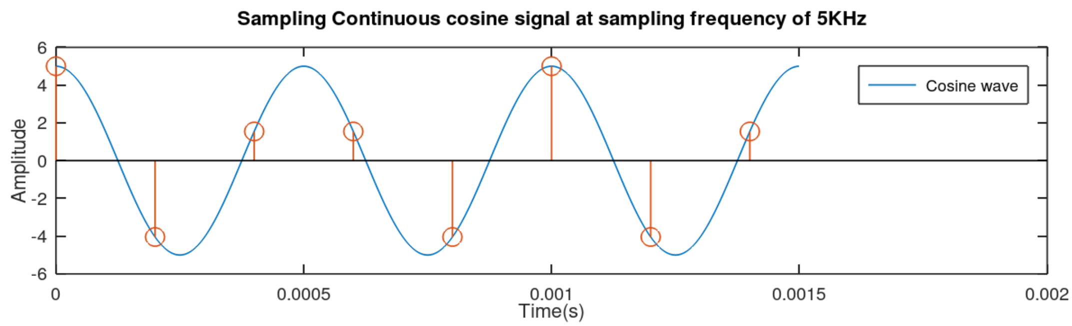
3.2.2 Generate the signal x = 5cos(2 pi f t) with 3 cycles, where f = 2 kHz signal and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz. (Title and label each figure)

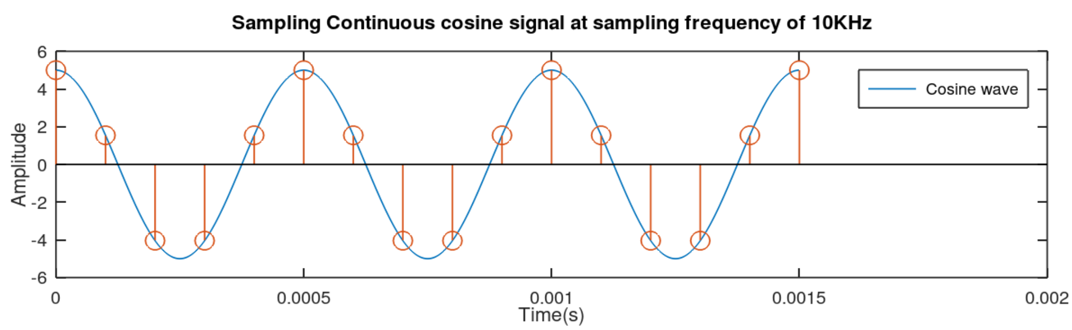
Source Code:

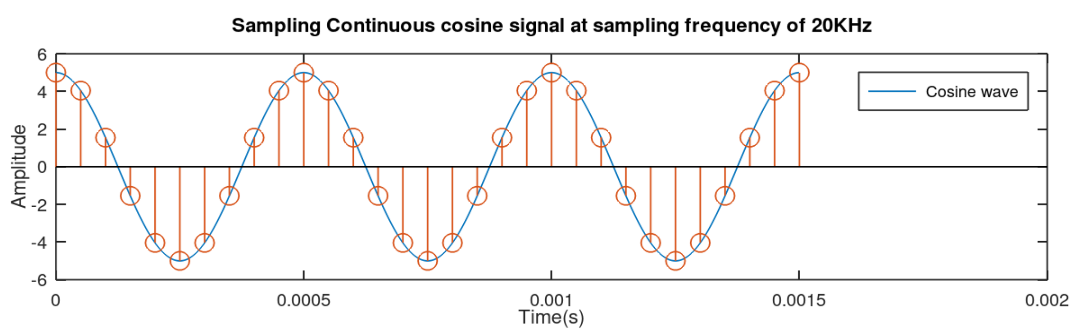
```
% Question 2: Generate the signal x = 5cos(2 pi f t) with 3 cycles, where f = 2 kHz signal and sample the signal with
%frequency 5 KHz, 10 Khz, 20 KHz. (Title and label each figure)

%Cosine wave
cycles=3;
f=2000
t=0:0.000001:cycles*1/f;
x = 5*cos(2*pi*f*t);
subplot(4,2,2)
plot(t,x);
title('Continuous cosine signal at frequency=2 kHz with 3 cycles');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Cosine wave');

%Sampling the signal with frequency 5 KHz
fs1=5000;
t1=0:1/fs1:cycles*1/f
x1 = 5*cos(2*pi*f*t1);
subplot(4,2,4)
plot(t,x);
hold on
stem(t1,x1);
title('Sampling Continuous cosine signal at sampling frequency of 5KHz');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Cosine wave');

%Sampling the signal with frequency 10 KHz
fs2=10000
t2=0:1/fs2:cycles*1/f;
x2 = 5*cos(2*pi*f*t2);
subplot(4,2,6)
plot(t,x);
hold on;
stem(t2,x2);
title('Sampling Continuous cosine signal at sampling frequency of 10KHz');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Cosine wave');


%Sampling the signal with frequency 20 KHz
fs3=20000
t3=0:1/fs3:cycles*1/f
x3 = 5*cos(2*pi*f*t3);
subplot(4,2,8)
plot(t,x);
hold on;
stem(t3,x3);
title('Sampling Continuous cosine signal at sampling frequency of 20KHz');
xlabel('Time(s)');
ylabel('Amplitude');
legend('Cosine wave');
```

Output:


Continuous cosine signal at frequency=2 kHz with 3 cycles
Amplitude
Time(s)
Cosine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002


Sampling Continuous cosine signal at sampling frequency of 5KHz
Amplitude
Time(s)
Cosine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002


Sampling Continuous cosine signal at sampling frequency of 10KHz
Amplitude
Time(s)
Cosine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002


Sampling Continuous cosine signal at sampling frequency of 20KHz
Amplitude
Time(s)
Cosine wave
6
4
2
0
-2
-4
-6
0
0.0005
0.001
0.0015
0.002

Lab 4 :Fourier Series

4.1. Introduction

In mathematics, a Fourier series is a periodic function composed of harmonically related sinusoids, combined by a weighted summation. With appropriate weights, one cycle of the summation can be made to approximate an arbitrary function in that interval. Basically, the Fourier Series is a function that breaks down any periodic function into a simple series of sine & cosine waves. Equation of fourier series:

$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \{a_n \cos nx + b_n \sin nx\}$$

Here, f(x) (left-side) is the target function we're attempting to approximate through the Fourier Series (right-side)

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\,dx, \quad a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx, \quad b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx.$$

Where a0, an, and bn are coefficients of Fourier series.

Even and Odd Function

A function f(x) is said to be even if f(−x) = f(x).

The function f(x) is said to be odd if f(−x) = −f(x).

Graphically, even functions have symmetry about the y-axis, whereas odd functions have symmetry around the origin.

4.2 Implementation

4.2.1 Tasks

- Fourier series expansion of odd signal for different N.(N= 3, 9, 100).
- Fourier series expansion of even signal for different N. (N=3,9,100).

Source Code:

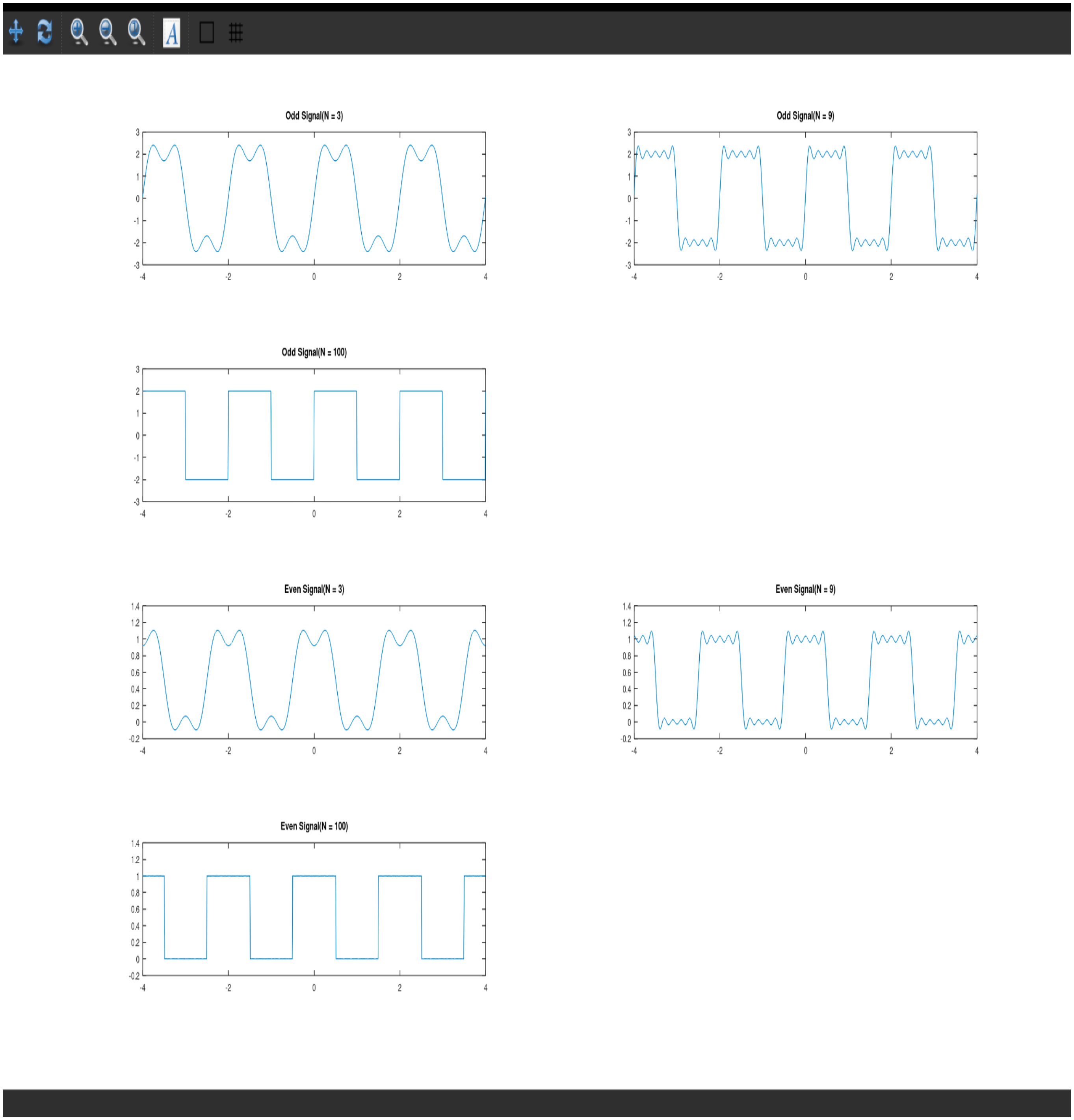
Editor

File Edit View Debug Run Help

lab4.m

```
% Lab 4


%1. Fourier series expansion of odd signal for different N.(N= 3, 9, 100).
subplot(4,2,1)
FourierSeries(3,1);
title("Odd Signal(N = 3)")

subplot(4,2,2)
FourierSeries(9,1);
title("Odd Signal(N = 9)")

subplot(4,2,3)
FourierSeries(100,1);
title("Odd Signal(N = 100)")

%2. Fourier series expansion of even signal for different N. (N=3,9,100).
subplot(4,2,5)
FourierSeries(3,2);
title("Even Signal(N = 3)")

subplot(4,2,6)
FourierSeries(9,2);
title("Even Signal(N = 9)")

subplot(4,2,7)
FourierSeries(100,2);
title("Even Signal(N = 100)")


function void = FourierSeries(N,evorod)
  Ts = 0.01;
  T = 2;
  t=0;
  if evorod == 1
    t= 0:Ts:T-Ts;
    f(t < T/2) = 2
    f(t >= T/2) = -2
  endif
  if evorod == 2
      t = -T/2:Ts:T/2;
      f(t < -T/4) = 0;
      f((t >= -T/4) & (t <= T/4)) = 1;
      f(t > T/4) = 0;
  endif

  a = zeros(1, N+1);
  b = zeros(1, N+1);
  for n = 0:N
      a(n+1) = (2 * Ts / T) * sum(f .* cos(2 * pi * n * t / T));
      b(n+1) = (2 * Ts / T) * sum(f .* sin(2 * pi * n * t / T));
  end
  t = -2*T:Ts:2*T;
  fs = (a(1)/2) * ones(size(t));
  for n = 1:N
      fs = fs + (a(n + 1) * cos(2*pi*n*t/T)) + (b(n + 1) * sin(2*pi*n*t/T)) ;
  end
  plot(t,fs)
end
```

Output:


Odd Signal(N = 3)
Odd Signal(N = 9)
Odd Signal(N = 100)
Even Signal(N = 3)
Even Signal(N = 9)
Even Signal(N = 100)

Lab 5 : Convolution of two signals (Linear and Circular Convolution).

5.1 Introduction

5.1.1. Convolution

Convolution is a mathematical way of combining two signals to form a third signal. It is the single most important technique in Digital Signal Processing. Using the strategy of impulse decomposition, systems are described by a signal called the impulse response. Convolution is important because it relates the three signals of interest: the input signal, the output signal, and the impulse response.

5.1.2. Linear Convolution

Linear convolution is the basic operation to calculate the output of any linear time invariant system given its input and its impulse response. It is more than multiplication and has one signal multiplied to multiple delays and advancements of another signal from -∞ to ∞. The size of output in linear convolution is L+M-1 where L is the size of the first input signal and M is the size of the second input signal.

Linear Convolution states that

$$y(n) = x(n) * h(n)$$

$$y(n) = x(n) * h(n)$$

$$y(n) = \sum_{k=-\infty}^{\infty} x(k)\ h(n-k) \qquad = \sum_{k=-\infty}^{\infty} x(k)\ h[-(k-n)]$$

5.1.3. Circular Convolution

Circular convolution, also known as cyclic convolution, is a special case of periodic convolution, which is the convolution of two periodic functions that have the same period. Periodic convolution arises, for example, in the context of the discrete-time Fourier transform (DTFT). Circular convolution is similar to linear convolution but works with periodic signals. The size of both the signals should be the same for circular convolution. The size of the output signal will be the same as of the input signals.

$$x_3(n) = \sum_{m=0}^{N-1} x_1(m) x_2[((n-m))_N] \quad m = 0, 1, 2 \ldots N-1$$

5.2 Implementation

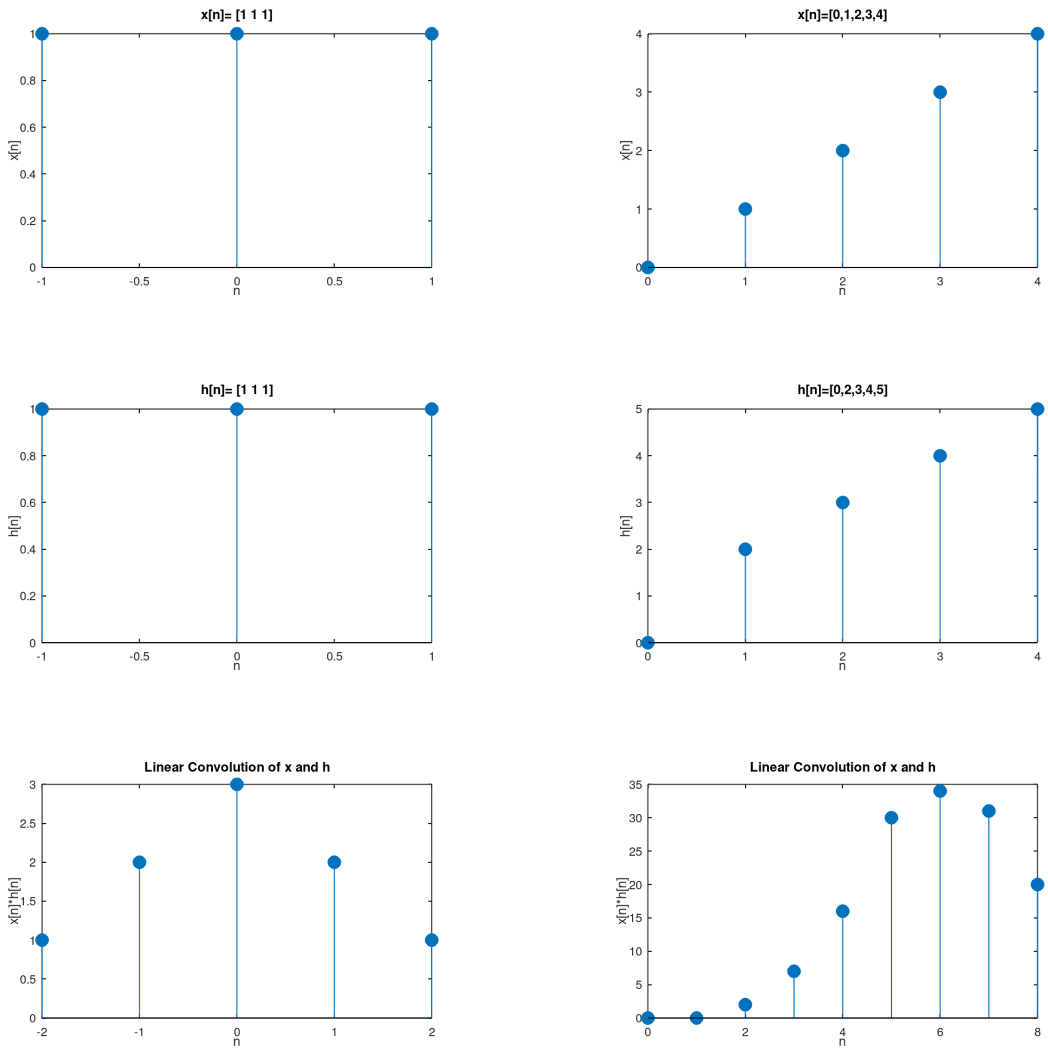
5.2.1 Perform Linear Convolution

1. $x[n]=\{1,\underset{\uparrow}{1},1\}$ & $h[n]=\{1,\underset{\uparrow}{1},1\}$.
2. $x[n]=\{\underset{\uparrow}{0},1,2,3,4\}$ & $h[n]=\{\underset{\uparrow}{0},2,3,4,5\}$

Source Code:

```
%Linear Convolution
x_axis=[-1,0,1]
x = [1 1 1];
h = [1 1 1];
linear_convolution = conv(x,h);

%Part 1
subplot(3,3,1)
stem(x_axis,x,'filled')
xlabel('n')
ylabel('x[n]')
title('x[n]= [1 1 1]')

subplot(3,3,4)
stem(x_axis,h,'filled')
xlabel('n')
ylabel('h[n]')
title('h[n]= [1 1 1]')

subplot(3,3,7)
x_axis=x_axis(:,1,1)*2: x_axis(end,end,1)*2
stem(x_axis,linear_convolution,'filled')
xlabel('n')
ylabel('x[n]*h[n]')
title('Linear Convolution of x and h')

%Part 2
x=[0,1,2,3,4]
h=[0,2,3,4,5]
x_axis=[0,1,2,3,4]

linear_convolution=conv(x,h)

subplot(3,3,2)
stem(x_axis,x,'filled')
xlabel('n')
ylabel('x[n]')
title('x[n]=[0,1,2,3,4]')

subplot(3,3,5)
stem(x_axis,h,'filled')
xlabel('n')
ylabel('h[n]')
title('h[n]=[0,2,3,4,5]')

subplot(3,3,8)
x_axis=x_axis(:,1,1)*2: x_axis(end,end,1)*2
stem(x_axis,linear_convolution,'filled')
xlabel('n')
ylabel('x[n]*h[n]')
title('Linear Convolution of x and h')

```

Output:



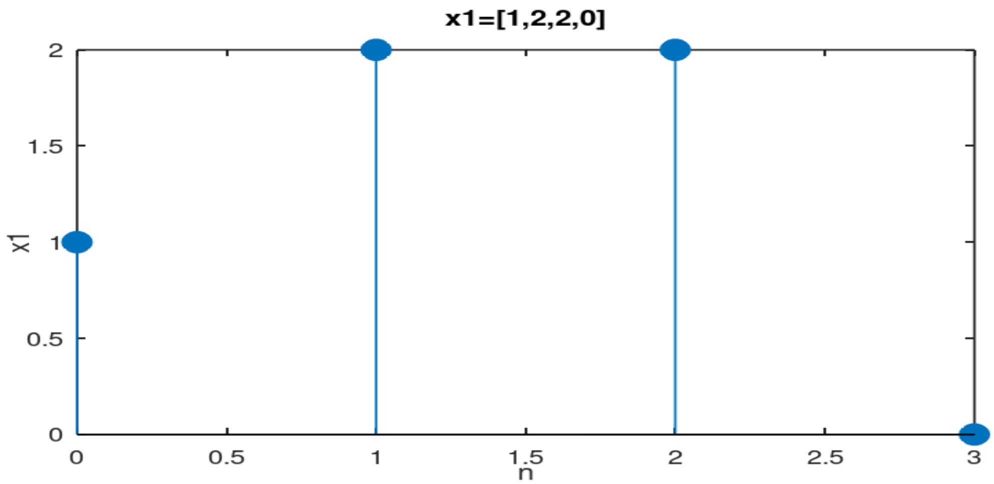
5.2.2 Perform Circular Convolution

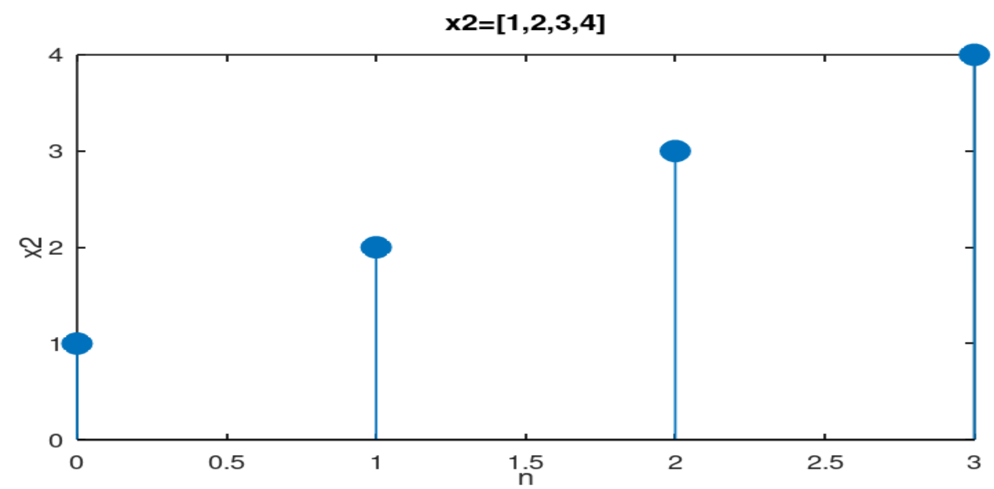
x1=[1,2,2,0] and x1=[1,2,3,4]

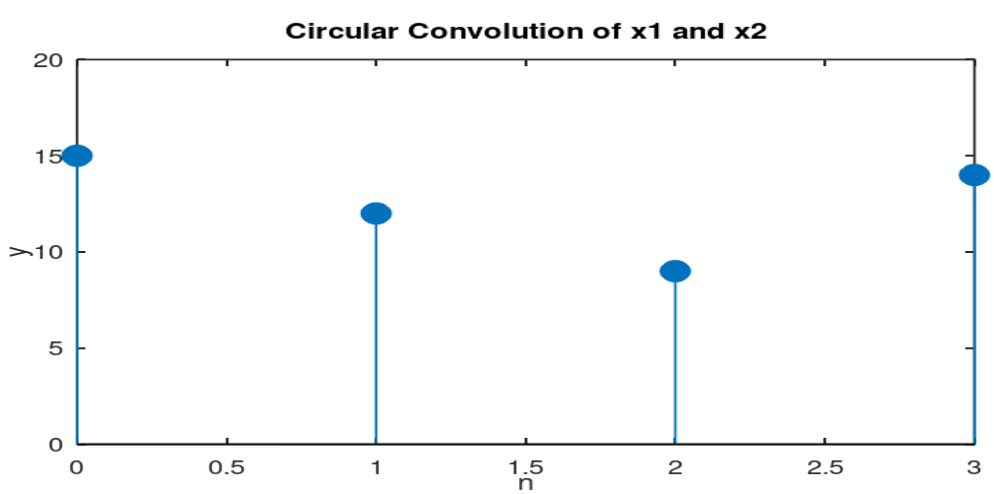
Source Code:

```
% Circular Convolution
x1=[1,2,2,0]
x2=[1,2,3,4]
x_axis=[0,1,2,3]
subplot(3,3,3)
stem(x_axis,x1,'filled')
xlabel('n')
ylabel('x1')
title('x1=[1,2,2,0]')

subplot(3,3,6)
stem(x_axis,x2,'filled')
xlabel('n')
ylabel('x2')
title('x2=[1,2,3,4]')
l1 = length(x1)
l2 = length(x2)
n = max(l1, l2)
y = zeros(1,n);

if l1 > l2
  x2 = [x2, zeros(1, l1 - l2)]
elseif l1 < l2
  x1 = [x1, zeros(1, l2 - l1)]
end
for m = 0:n-1
  for o=0:n-1
    z=mod(m-o,n);
    y(m+1)=y(m+1)+x1(o+1).*x2(z+1);

  endfor
endfor



subplot(3,3,9)
stem(x_axis,y,'filled')
xlabel('n')
ylabel('y')
title('Circular Convolution of x1 and x2')
```

Output:


x1=[1,2,2,0]
x1
n
0
0.5
1
1.5
2
2.5
3


x2=[1,2,3,4]
x2
n
0
1
2
3
4
0.5
1.5
2.5


Circular Convolution of x1 and x2
y
n
0
5
10
15
20
0.5
1
1.5
2
2.5
3

6. Conclusion

From the lab work done for the digital signal processing using Octave we got to know how to use the Octave / Matlab. We got the knowledge about how to properly visualize the signals using Octave. From the visualization we got to understand how convolution is done as well as how sampling is done. We got to change different data and visualize how the signal changes which positively impacted our understanding of the subject Digital Signal Processing.

7. Github

https://github.com/Babin6139/DSP_LAB