Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

Lab Report (1 - 5)

of

"Digital Signal Processing"

COMP 407

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

Submitted by:

Neha Malla (27)

Submitted to:

Mr. Satyendra Nath Lohani

Department of Computer Science and Engineering

Submission Date: 5th November, 2020

LAB 1: Introduction to Matlab

Objective:

1. To study important commands of MATLAB software: clc, close, xlabel, ylabel, zlabel, title, figure, subplot, linspace, stem, bar, plot
2. For familiarization with the MATLAB environment
    2.1. Create a matrix, A of size 3*4. Create another matrix, B of size 4*3
    2.2. Add Matrix A and B. Subtract A from B
    2.3. Multiply A and B. Multiply B and A
    2.4. Transpose matrix A and B. Multiply the transposed matrices

Source Code:

```
# To study important commands of MATLAB software
x = [0, 1, 2, 3, 4, 5]
y = [0, 2, 4, 6, 8, 10]
subplot(2, 2, 1)
plot(x, y)
title("graph 1")
xlabel("x axis")
ylabel("y axis")

x1 = linspace(-10, 10, 100);
y1 = x1.^2;
subplot(2, 2, 2)
plot(x1, y1)
```

```
title("graph 2")
xlabel("x axis")
ylabel("y axis")

y2 = [75 91 105 123.5 131 150 179 203 226 249 281.5];
subplot(2, 2, 3)
bar(y2)
title("graph 3")
xlabel("x axis")
ylabel("y axis")

x3 = linspace(0, 10, 10)';
y3 = (exp(0.25*x3));
subplot(2, 2, 4)
stem(x3, y3)
title("graph 4")
xlabel("x axis")
ylabel("y axis")

# Familiarization with MATLAB environment
A = [1, 2, 3, 4;
     5, 6, 7, 8;
     9, 10, 11, 12];
B = [12, 11, 10;
     9, 8, 7;
     6, 5, 4;
     3, 2, 1];
sum = A + B;
sub = B - A;

mulAB = A*B;
mulBA = B*A;
```

```
transA = A'
transB = B'
mulTransAB = transA*transB
mulTransBA = transB*transA
```

Output:

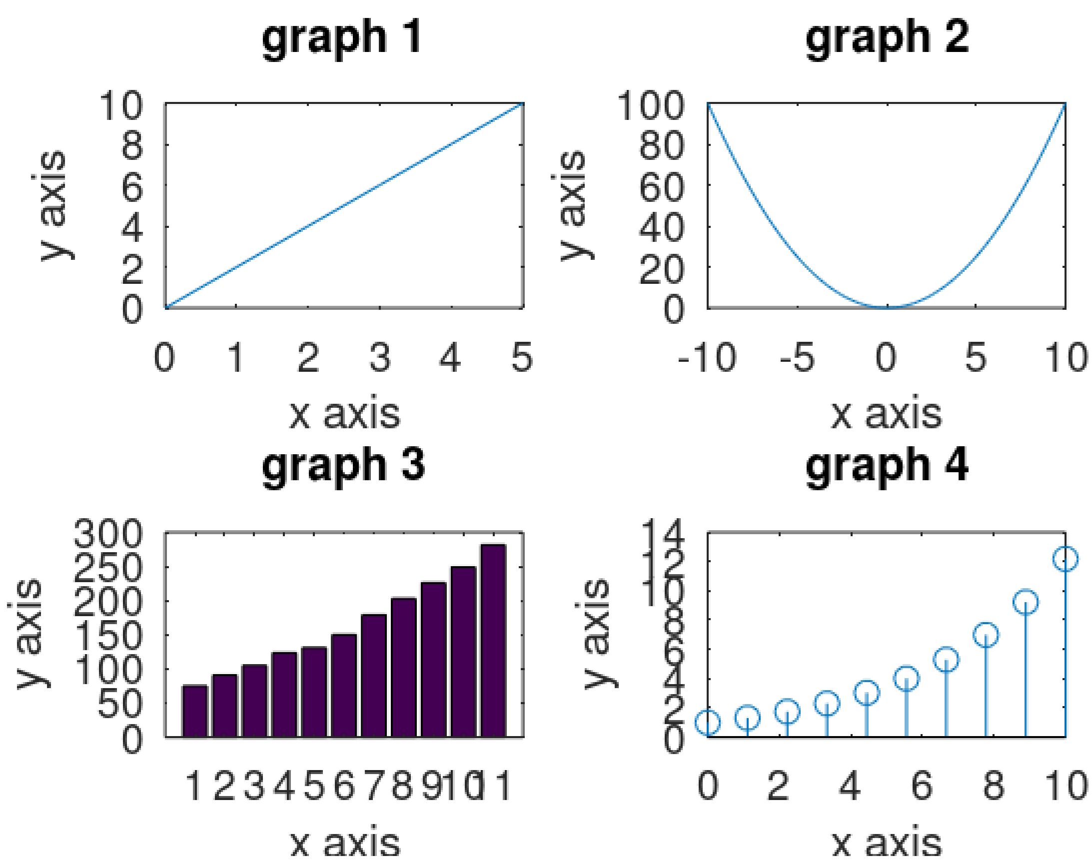


Figure 1.1: To study important commands of MATLAB software

Command Window

```
>> A = [1, 2, 3, 4;
     5, 6, 7, 8;
     9, 10, 11, 12];
>> B = [12, 11, 10;
     9, 8, 7;
     6, 5, 4;
     3, 2, 1];
>>
>> sum = A+B;
error: operator +: nonconformant arguments (op1 is 3x4, op2 is 4x3)
>> sub = B-A;
error: operator -: nonconformant arguments (op1 is 4x3, op2 is 3x4)
>>
>> A*B
ans =

    60    50    40
   180   154   128
   300   258   216

>> B*A
ans =

   157   190   223   256
   112   136   160   184
    67    82    97   112
    22    28    34    40

>> A'
ans =

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

>> B'
ans =

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

>> A'*B'
ans =

   157   112    67    22
   190   136    82    28
   223   160    97    34
   256   184   112    40

>> B'*A'
ans =

    60   180   300
    50   154   258
    40   128   216

>>
```

Figure 1.2: For familiarization with the MATLAB environment

Discussion:

Here we got familiar with the general commands to plot different types of graphs and then performed some operations on two matrices A and B of size 3*4 and 4*3 respectively. The addition and subtraction of these two matrices aren't possible as their size varies, there is an error. Both multiplications, A*B and B*A are possible here as it is verified by the multiplication rule of [n*m] . [m*p] = [n*p]. Similarly the multiplication of their transpose is possible.

LAB 2: Introduction to Signals

Objective:

1. To generate a continuous time sinusoidal wave of amplitude 5
2. To generate a unit impulse function.
3. To generate a unit step function.
4. To generate a unit ramp function.
5. To generate a continuous time sinc function
6. To generate a continuous time exponential (growing, decaying, DC signal)

Source Code:

```
# 1. continuous time sinusoidal wave of amplitude 5
x = [-10:0.1:10];
y  = 5*sin(x);
plot(x, y);
title('continuous time sinusoidal wave of amplitude 5');
xlabel('x axis');
ylabel('y axis');

# 2. unit impulse function
t = [-10:1:10];
impulse = t == 0;
stem(t, impulse);
title('unit impulse');
xlabel('time');
```

```
ylabel('amplitude');

# 3. unit step function
t = [-10:1:10];
unitstep = t >= 0;
stem(t, unitstep)
title('unit step');
xlabel('time');
ylabel('amplitude');

# 4. unit ramp function
t = [-10:1:10];
ramp = t.*unitstep;
stem(t, ramp);
title('unit ramp');
xlabel('time');
ylabel('amplitude')

# 5. continuous time sinc function
x = [-10:0.1:10];
y = sinc(x);
plot(x, y);
grid
title('continuous time sinc ');
xlabel('time');
ylabel('amplitude');

# 6. continuous time exponential
# growing
subplot(1, 3, 1);
x = [-10:0.1:10];
y = exp(x);
```

```
plot(x, y);
title('growing exponential');
xlabel('time');
ylabel('amplitude');
# decaying
subplot(1, 3, 2);
x = [-10:0.1:10];
y = exp(-x);
plot(x, y);
title('decaying exponential');
xlabel('time');
ylabel('amplitude');

# DC signal
t = linspace(-4*pi, 4*pi)';
x = square(t);
plot(t/pi , x, 'r');
grid
title('DC signal');
xlabel('t / \pi')
```

Output:

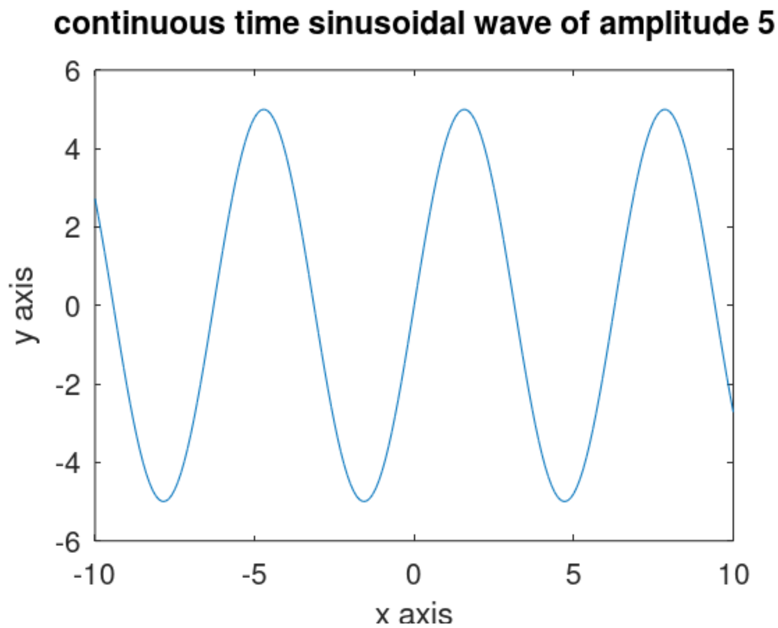


Figure 2.1: Continuous time sinusoidal wave of amplitude 5

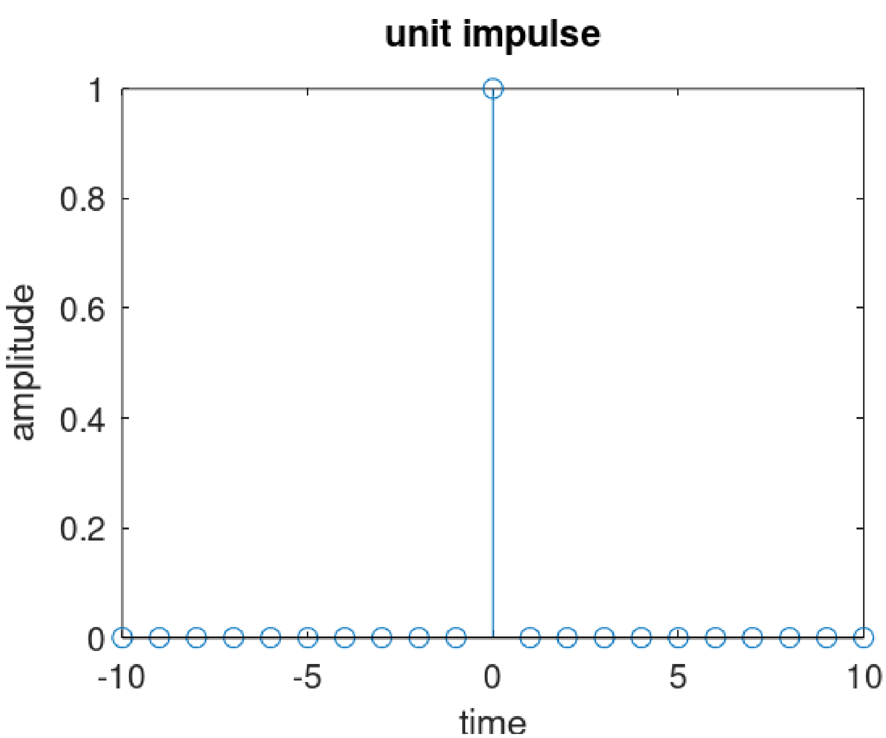


Figure 2.2: Unit impulse function

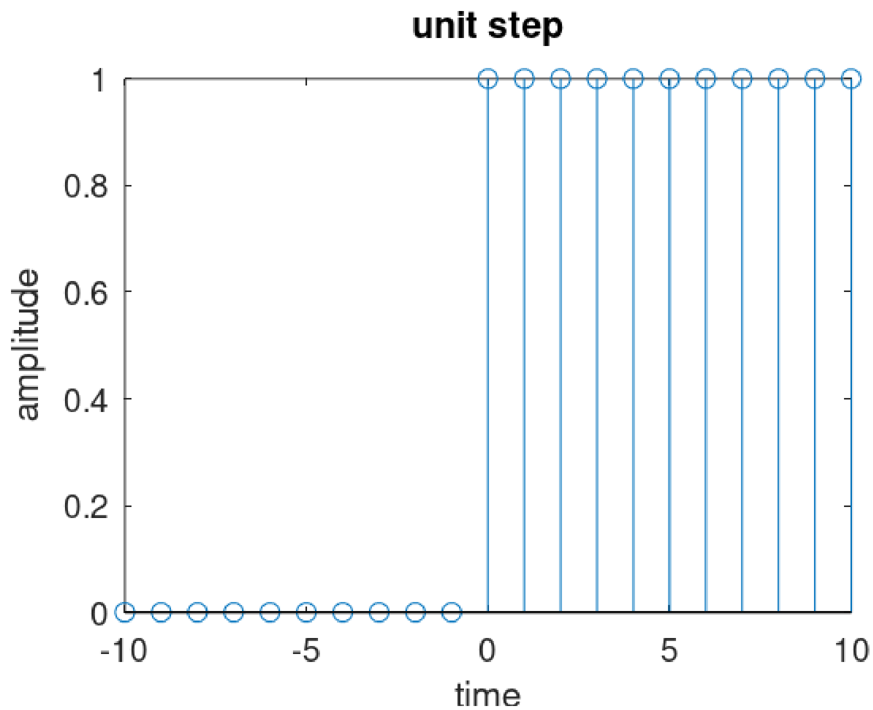


Figure 2.3: Unit step function

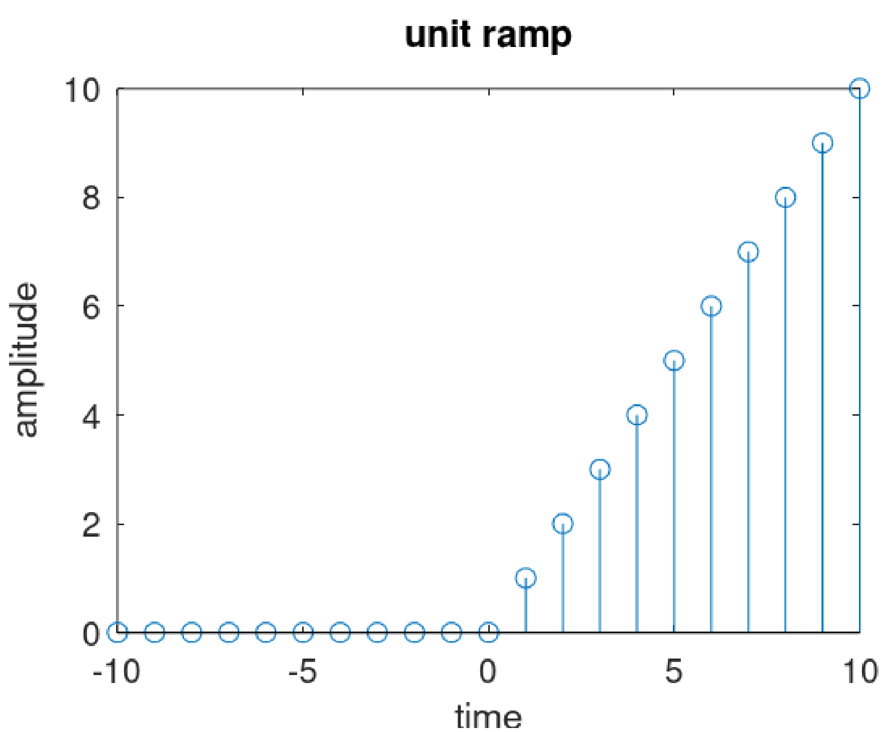


Figure 2.4: Unit ramp function

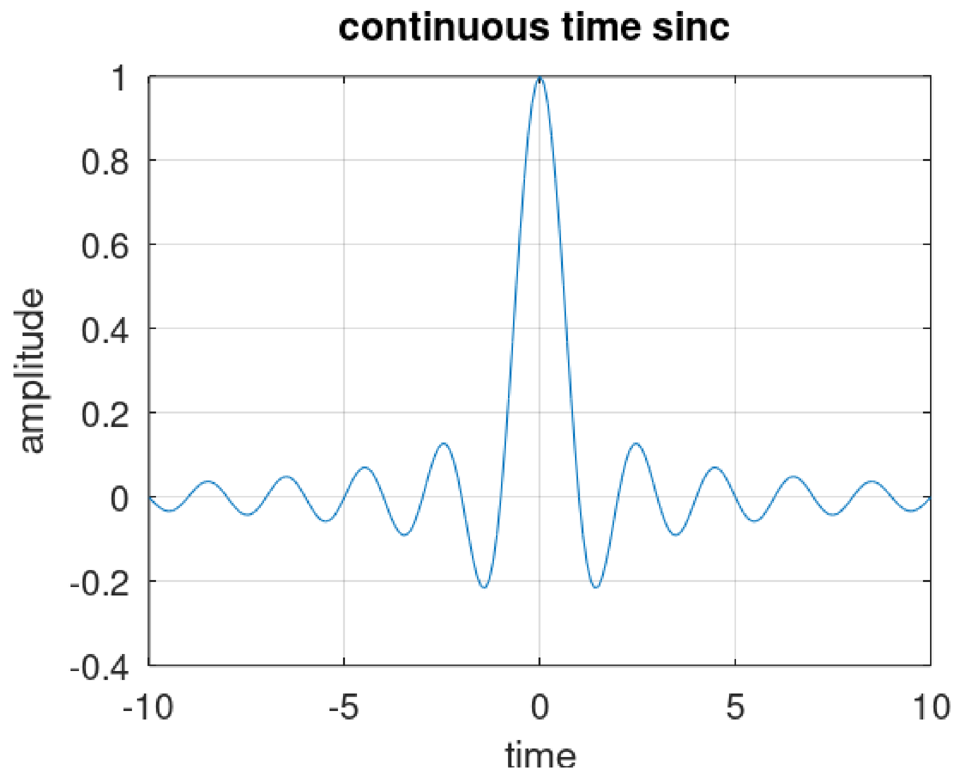


Figure 2.5: Continuous time sinc function

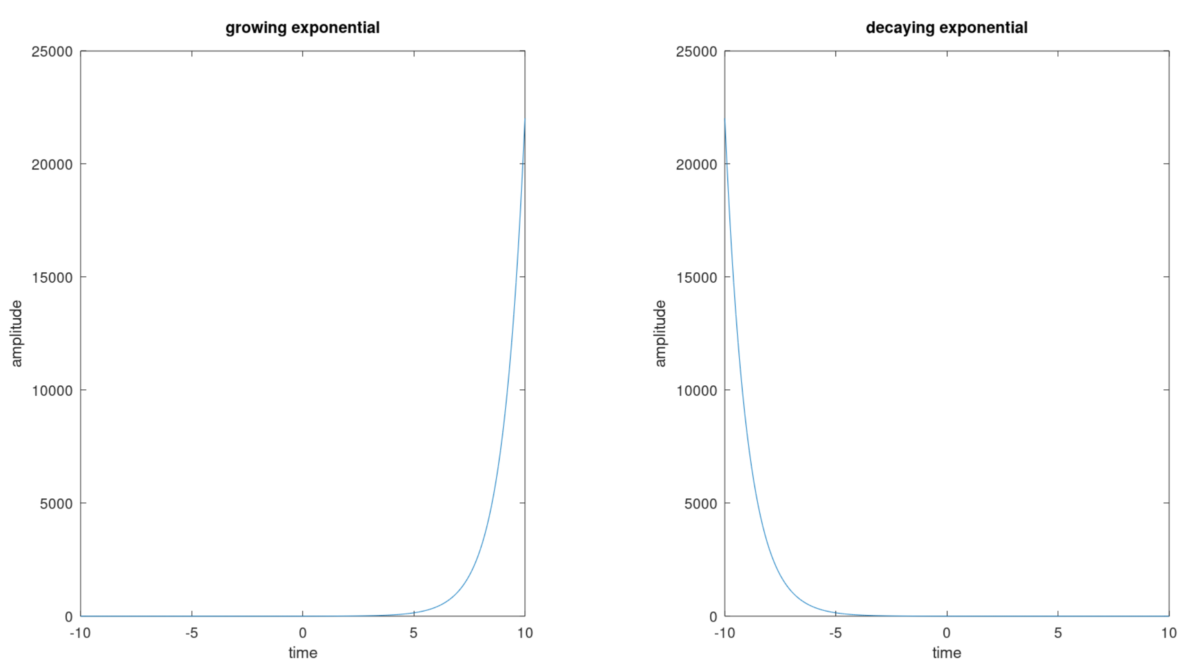


Figure 2.6.1: Continuous time exponential

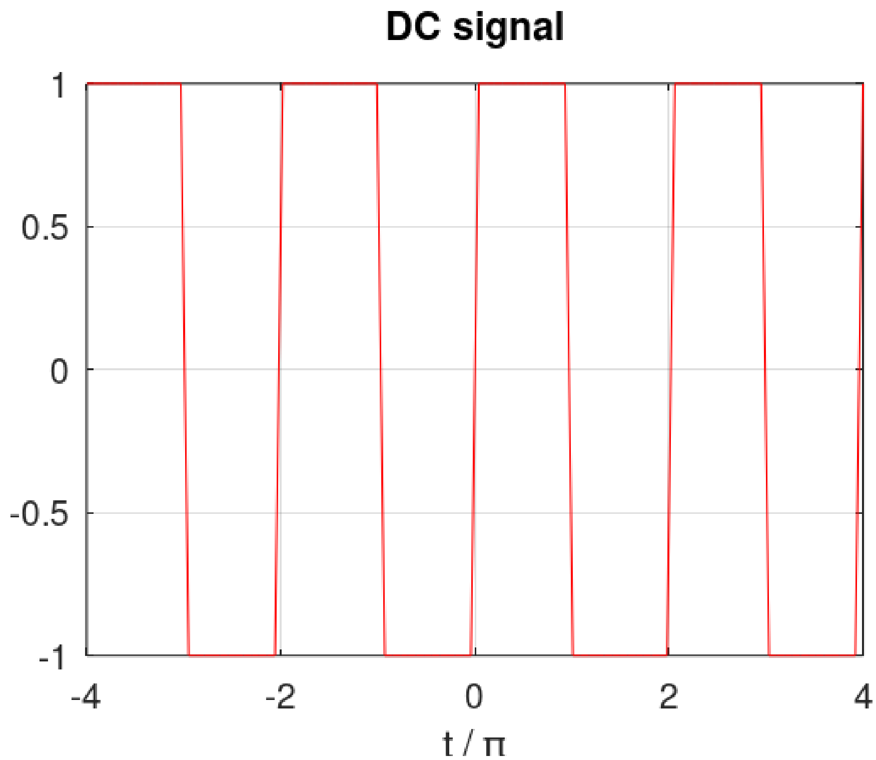


Figure 2.6.2: DC signal

LAB 3: Sampling of Signal

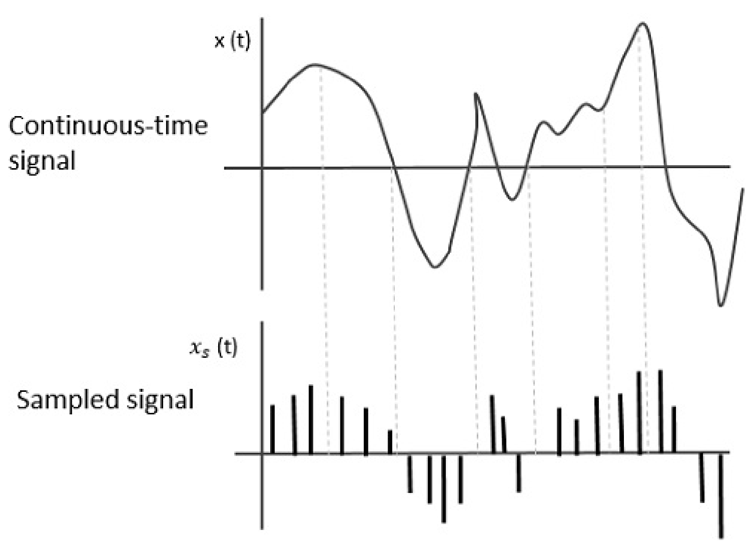
Objective:

1. To generate the signal x = 5sin(2 pi f t) with 5 cycles, where f = 2 kHz, and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz.
2. To generate the signal x = 5cos(2 pi f t) with 3 cycles, where f = 2 kHz, and sample the signal with frequency 5 KHz, 10 Khz, 20 KHz.

Description:

The process of measuring the instantaneous values of a continuous-time signal in a discrete form is defined as sampling. A sample is a piece of data taken from a larger data set that is continuous in the time domain. When a source generates an analog signal, the signal must be discretized in time if it is to be digitized as 1s and 0s, i.e., High or Low. Sampling is the discretization of an analog signal.

The diagram below depicts a continuous-time signal x t and a sampled signal xs t. The sampled signal xs t is obtained by multiplying x t by a periodic impulse train.



Source Code:

```
# 1. sinusoidal wave
     cycles = 5;
     f = 2000;
     t = 0:0.000001:cycles*1/f;
     x = 5*sin(2*pi*f*t);
     plot(t, x);
     title('Sinusoidal  wave  of  frequency  2kHz  and  having  5
     cycles');
     xlabel('time(sec)');
     ylabel('amplitude');

     # for 5 khz
     cycles = 5;
     f = 2000;
     t = 0:0.000001:cycles*1/f;
     x = 5*sin(2*pi*f*t);
     plot(t, x);
     hold on;
     sample1 = 5000;
     t1 = 0:1/sample1:cycles*1/f;
     x1 = 5*sin(2*pi*f*t1);
     stem(t1, x1);
     title('Sampling continuous sinusoidal signal at frequency =
     5KHz');
     xlabel('time(sec)');
     ylabel('amplitude');

     # for 10 khz
     cycles = 5;
     f = 2000;
```

```
    t = 0:0.000001:cycles*1/f;
    x = 5*sin(2*pi*f*t);
    plot(t, x);
    hold on;
    sample2 = 10000;
    t2 = 0:1/sample2:cycles*1/f;
    x2 = 5*sin(2*pi*f*t2);
    stem(t2, x2);
    title('Sampling continuous sinusoidal signal at frequency =
    10KHz');
    xlabel('time(sec)');
    ylabel('amplitude');

    # for 20 khz
    cycles = 5;
    f = 2000;
    t = 0:0.000001:cycles*1/f;
    x = 5*sin(2*pi*f*t);
    plot(t, x);
    hold on;
    sample3 = 20000;
    t3 = 0:1/sample3:cycles*1/f;
    x3 = 5*sin(2*pi*f*t3);
    stem(t3, x3);
    title('Sampling continuous sinusoidal signal at frequency =
    20KHz');
    xlabel('time(sec)');
    ylabel('amplitude');

# 2. cosine wave
    cycles = 3;
    f = 2000;
```

```
t = 0:0.000001:cycles*1/f;
x = 5*cos(2*pi*f*t);
plot(t, x);
title('Cosine wave of frequency 2kHz and having 3 cycles');
xlabel('time(sec)');
ylabel('amplitude');

# for 5 khz
cycles = 3;
f = 2000;
t = 0:0.000001:cycles*1/f;
x = 5*cos(2*pi*f*t);
plot(t, x);
hold on;
sample1 = 5000;
t1 = 0:1/sample1:cycles*1/f;
x1 = 5*cos(2*pi*f*t1);
stem(t1, x1);
title('Sampling continuous cosine signal at frequency = 
5KHz');
xlabel('time(sec)');
ylabel('amplitude');

# for 10 khz
cycles = 3;
f = 2000;
t = 0:0.000001:cycles*1/f;
x = 5*cos(2*pi*f*t);
plot(t, x);
hold on;
sample2 = 10000;
t2 = 0:1/sample2:cycles*1/f;
```

```
x2 = 5*cos(2*pi*f*t2);
stem(t2, x2);
title('Sampling continuous cosine signal at frequency = 10KHz');
xlabel('time(sec)');
ylabel('amplitude');

# for 20 khz
cycles = 3;
f = 2000;
t = 0:0.000001:cycles*1/f;
x = 5*cos(2*pi*f*t);
plot(t, x);
hold on;
sample3 = 20000;
t3 = 0:1/sample3:cycles*1/f;
x3 = 5*cos(2*pi*f*t3);
stem(t3, x3);
title('Sampling continuous cosine signal at frequency = 20KHz');
xlabel('time(sec)');
ylabel('amplitude');
```

Output:

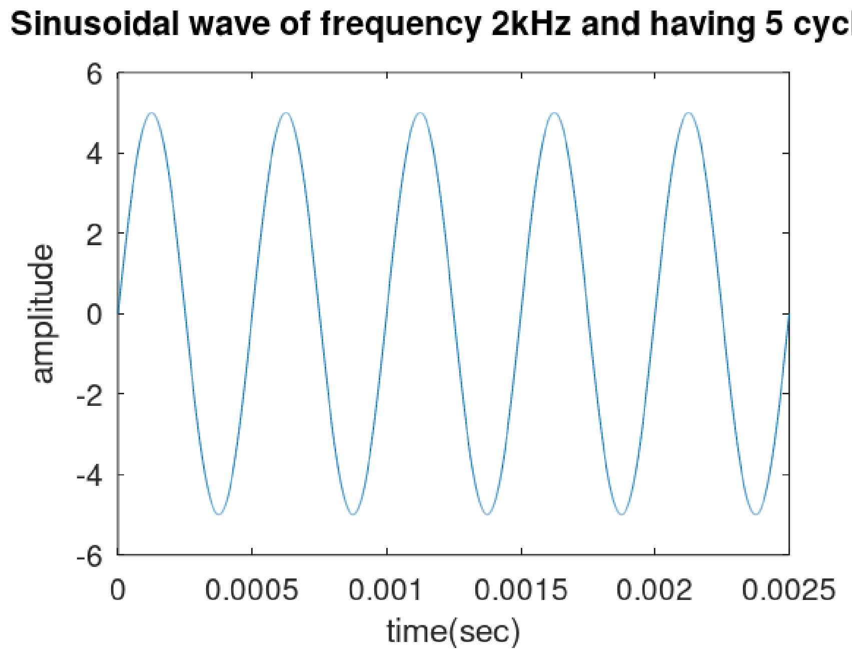


Figure 3.1: Sinusoidal wave of f = 2KHz with 5 cycles

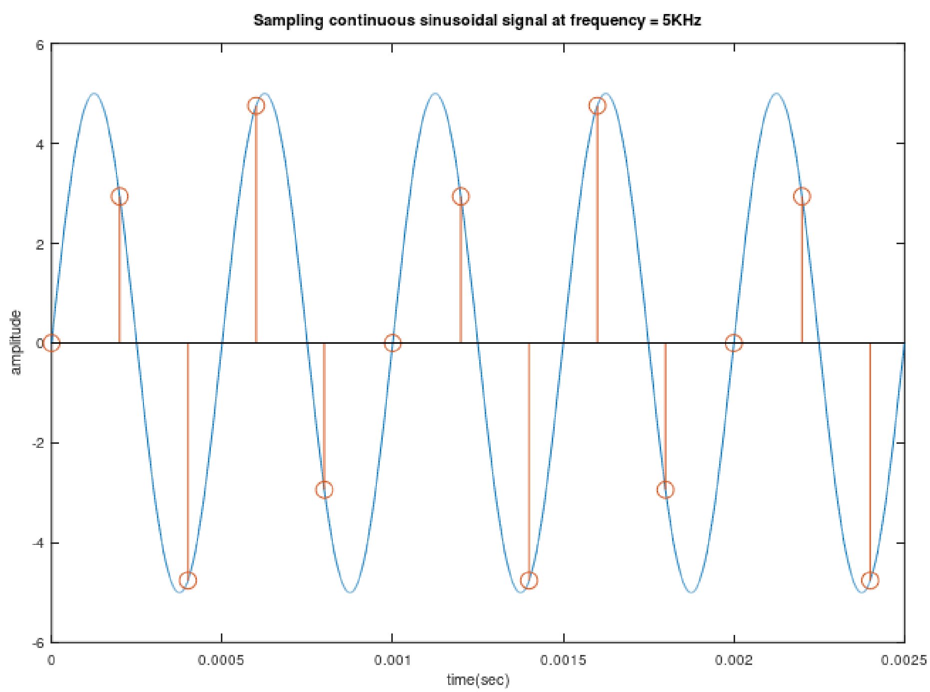


Figure 3.2: Sampling of sinusoidal wave at f = 5KHz

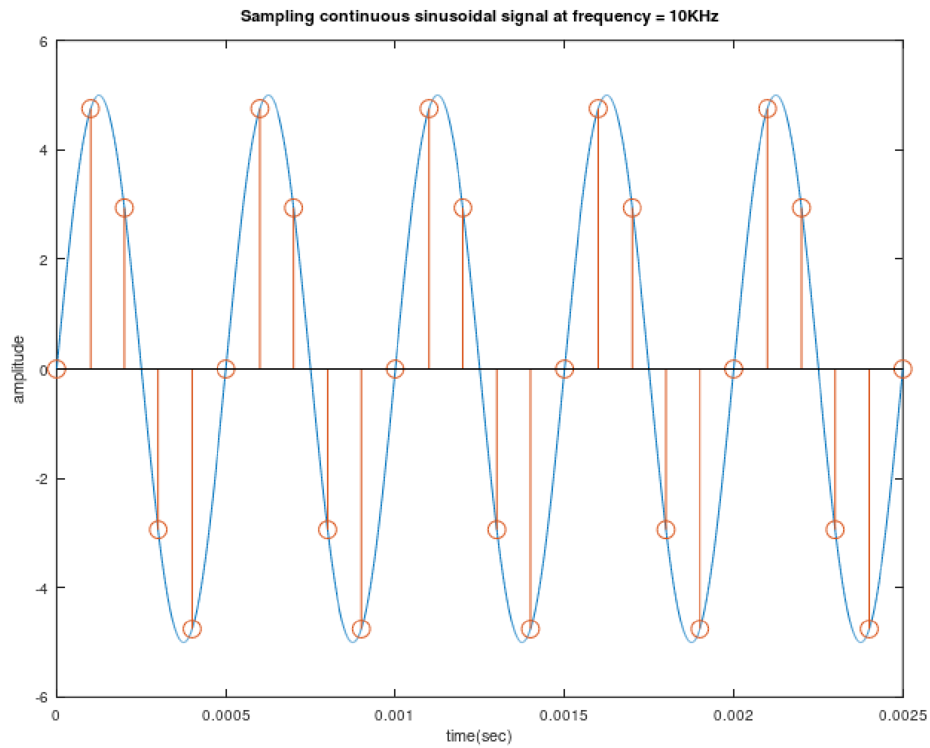


Figure 3.3: Sampling of sinusoidal wave at f = 10KHz

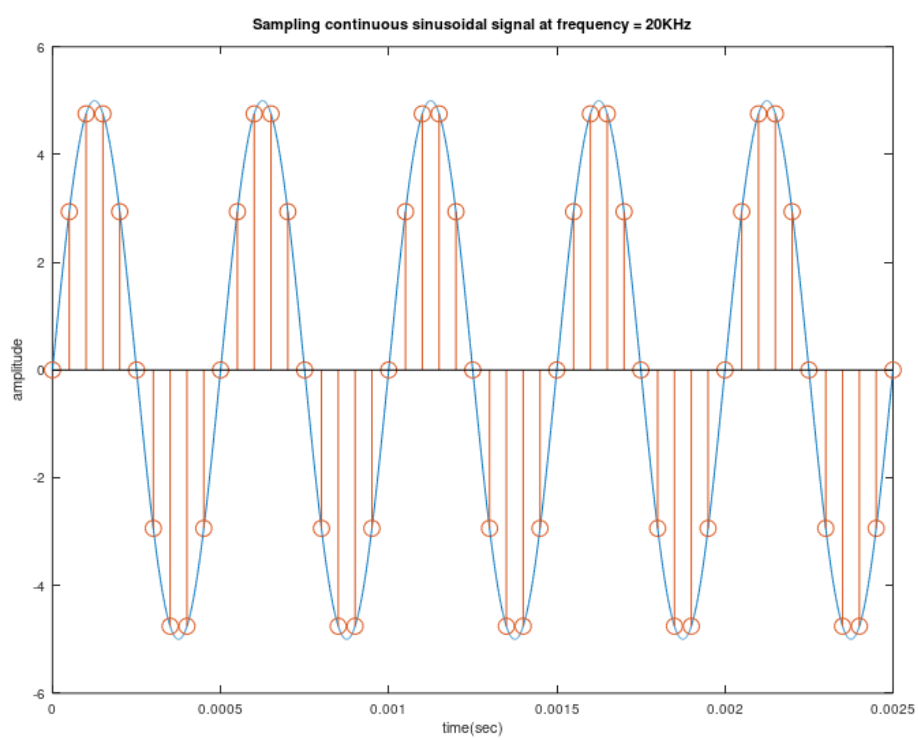


Figure 3.4: Sampling of sinusoidal wave at f = 20KHz

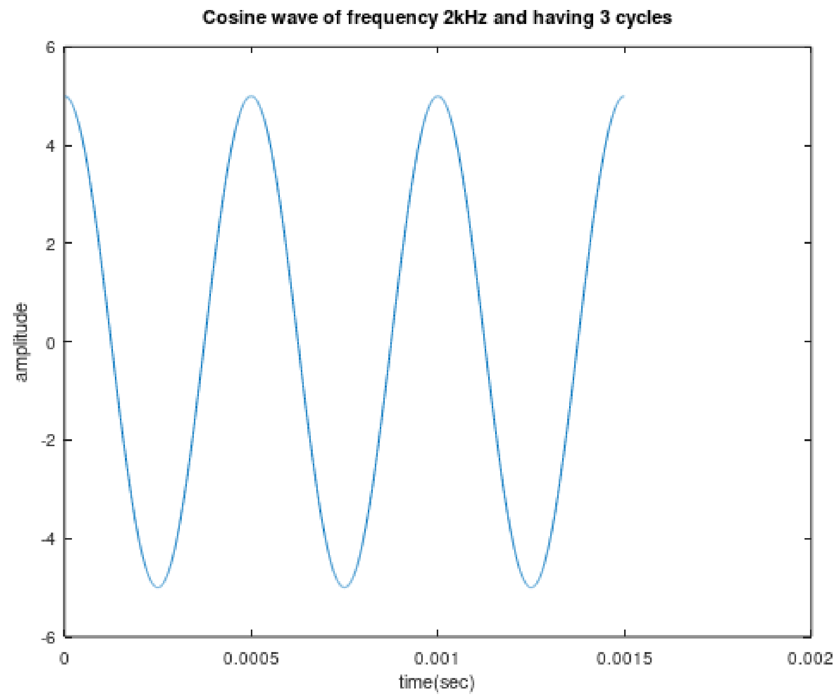


Figure 3.5: Cosine wave of f = 2KHz with 3 cycles

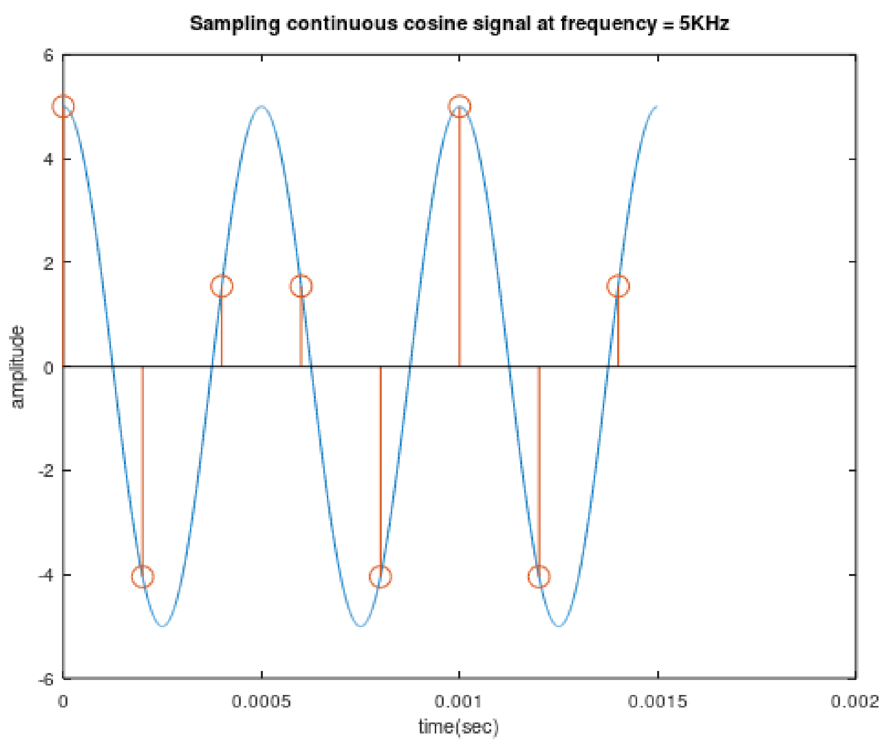


Figure 3.6: Sampling of cosine wave at f = 5KHz

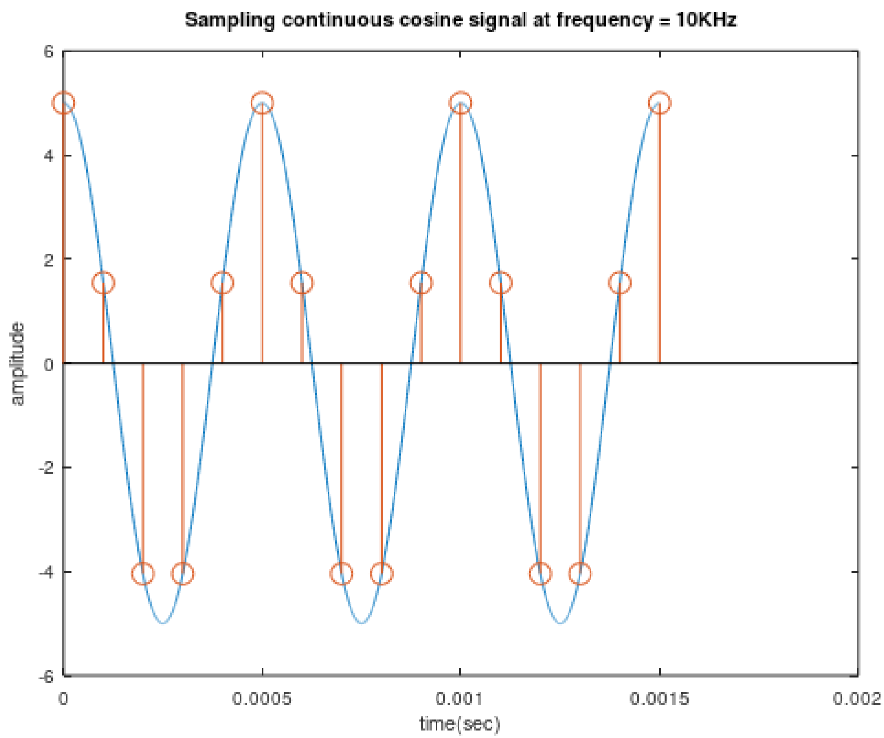


Figure 3.7: Sampling of cosine wave at f = 10KHz

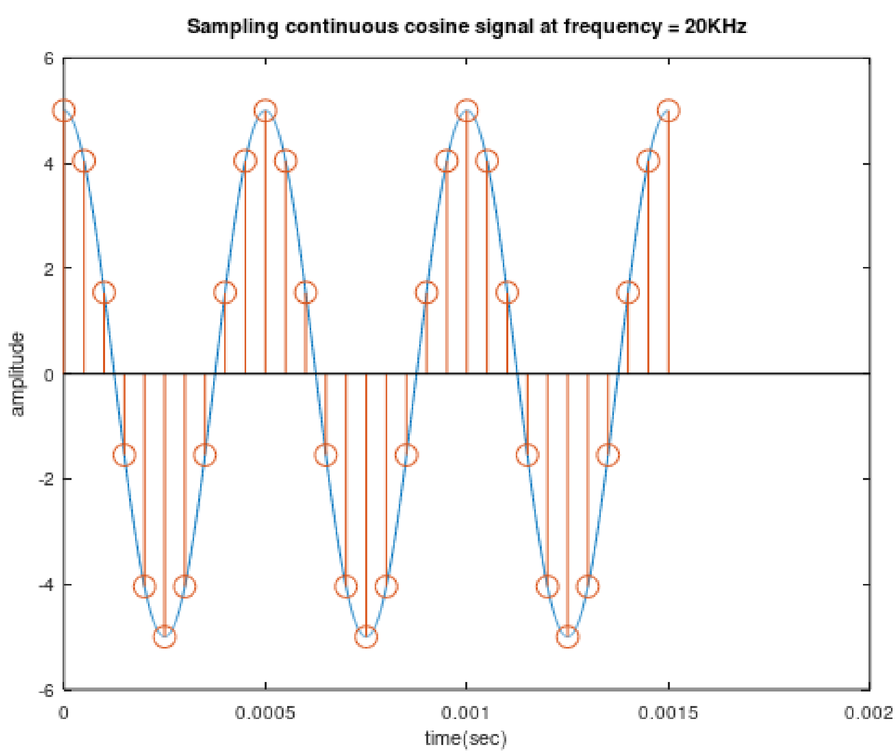


Figure 3.8: Sampling of cosine wave at f = 20KHz

LAB 4: Fourier Series

Objective:

1. To generate a fourier series expansion of odd signals for different N. (N = 3, 9, 100).
2. To generate a fourier series expansion of even signals for different N. (N = 3, 9, 100).

Description:

The Fourier series represents any periodic signal x(t) in terms of an infinite sum of sines and cosines

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

where

$$a_0 = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x)\, d x$$

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(n x)\, d x$$

$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(n x)\, d x$$

and n = 1, 2, 3, ...

Source Code:

```
# 1. Odd function: y = x
     clear all;clc;
     syms x n pi
     y = x;   # odd function
     a0 = (1/pi)*int(y, x, -pi, pi);
     sum = a0/2;
     N = 3;
     for n = 1:N
         #finding the coefficients
         an = (1/pi)*int(y*cos(n*x), x, -pi, pi)
         bn = (1/pi)*int(y*sin(n*x), x, -pi, pi)
         sum = sum + (an*cos(n*x) + bn*sin(n*x))
     end
     ezplot(x, y, [-pi, pi])
     grid on;hold on;
     ezplot(x, sum, [-pi, pi])
     title("y = x at N = 3")

     clear all;clc;
     syms x n pi
     y = x;   # odd function
     a0 = (1/pi)*int(y, x, -pi, pi);
     sum = a0/2;
     N = 3;
     for n = 1:N
         #finding the coefficients
         an = (1/pi)*int(y*cos(n*x), x, -pi, pi)
         bn = (1/pi)*int(y*sin(n*x), x, -pi, pi)
         sum = sum + (an*cos(n*x) + bn*sin(n*x))
     end
```

```
    ezplot(x, y, [-pi, pi])
    grid on;hold on;
    ezplot(x, sum, [-pi, pi])
    title("y = x at N = 3")

    clear all;clc;
    syms x n pi
    y = x;   # odd function
    a0 = (1/pi)*int(y, x, -pi, pi);
    sum = a0/2;
    N = 3;
    for n = 1:N
        #finding the coefficients
        an = (1/pi)*int(y*cos(n*x), x, -pi, pi)
        bn = (1/pi)*int(y*sin(n*x), x, -pi, pi)
        sum = sum + (an*cos(n*x) + bn*sin(n*x))
    end
    ezplot(x, y, [-pi, pi])
    grid on;hold on;
    ezplot(x, sum, [-pi, pi])
    title("y = x at N = 3")

# 2. Even function: y = x^2
    clear all;clc;
    syms x n pi
    y = abs(x);   # even function
    a0 = (1/pi)*int(y, x, -pi, pi);
    sum = a0/2;
    N = 3;
    for n = 1:N
        #finding the coefficients
        an = (1/pi)*int(y*cos(n*x), x, -pi, pi)
```

```
    bn = (1/pi)*int(y*sin(n*x), x, -pi, pi)
    sum = sum + (an*cos(n*x) + bn*sin(n*x))
end
ezplot(x, y, [-pi, pi])
grid on;hold on;
ezplot(x, sum, [-pi, pi])
title("y = |x| at N = 3")

clear all;clc;
syms x n pi
y = abs(x);   # even function
a0 = (1/pi)*int(y, x, -pi, pi);
sum = a0/2;
N = 9;
for n = 1:N
    #finding the coefficients
    an = (1/pi)*int(y*cos(n*x), x, -pi, pi)
    bn = (1/pi)*int(y*sin(n*x), x, -pi, pi)
    sum = sum + (an*cos(n*x) + bn*sin(n*x))
end
ezplot(x, y, [-pi, pi])
grid on;hold on;
ezplot(x, sum, [-pi, pi])
title("y = |x| at N = 9")

clear all;clc;
syms x n pi
y = abs(x);   # even function
a0 = (1/pi)*int(y, x, -pi, pi);
sum = a0/2;
N = 100;
for n = 1:N
```

```
    #finding the coefficients
    an = (1/pi)*int(y*cos(n*x), x, -pi, pi)
    bn = (1/pi)*int(y*sin(n*x), x, -pi, pi)
    sum = sum + (an*cos(n*x) + bn*sin(n*x))
end
ezplot(x, y, [-pi, pi])
grid on;hold on;
ezplot(x, sum, [-pi, pi])
title("y = |x| at N = 100")
```

Output:

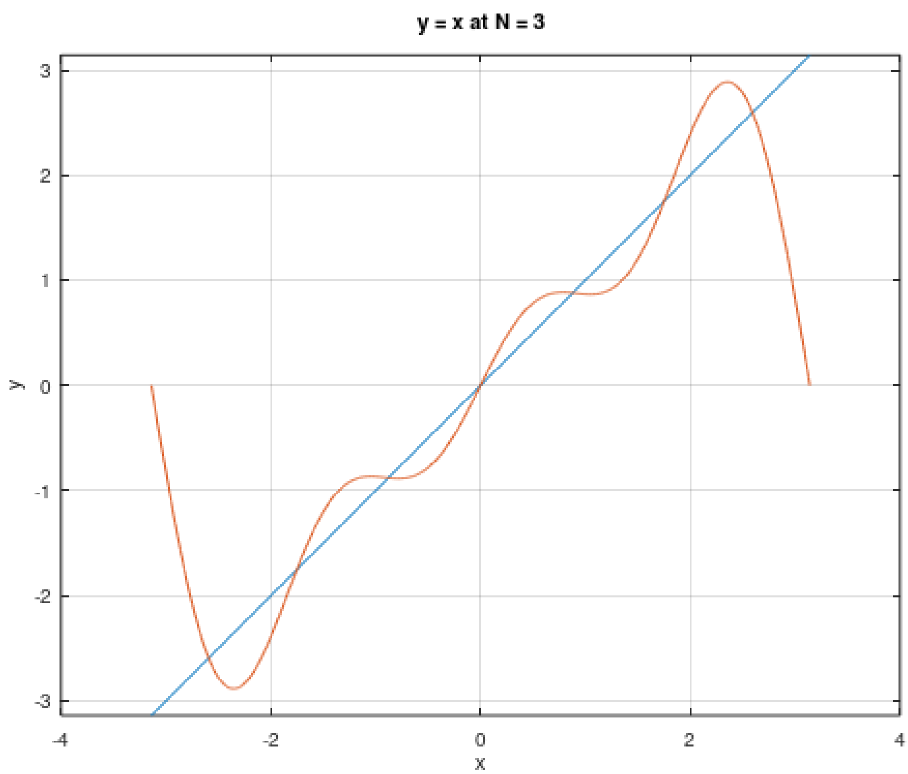


Figure 4.1: Fourier series odd function at N = 3

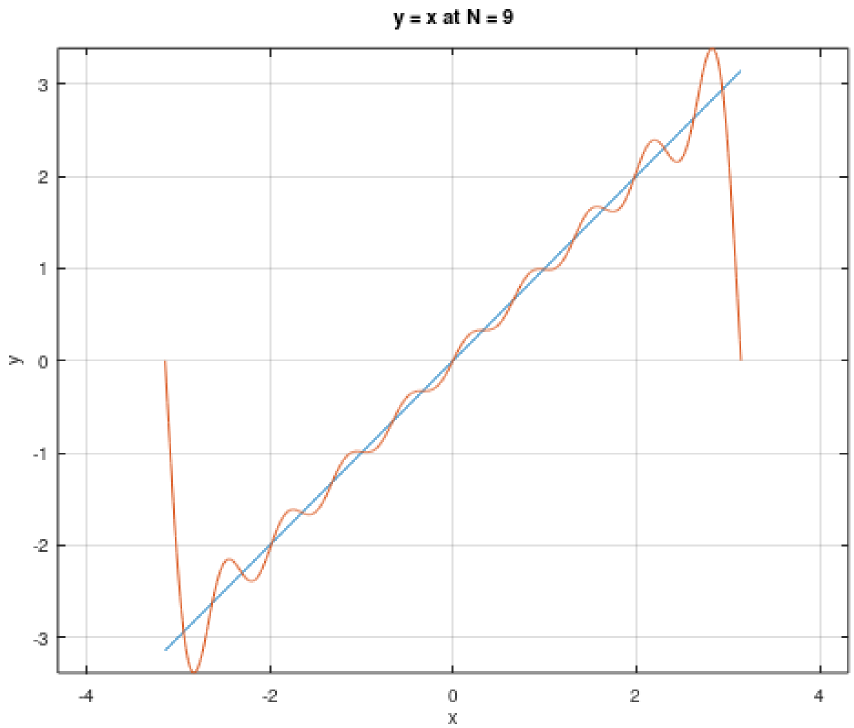


Figure 4.2: Fourier series odd function at N = 9

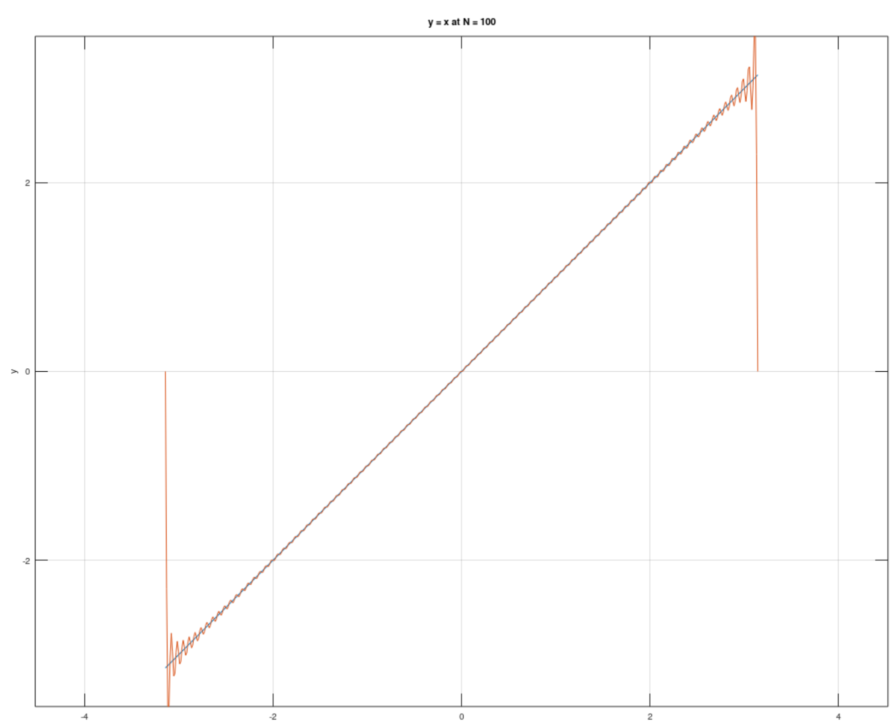


Figure 4.3: Fourier series odd function at N = 100

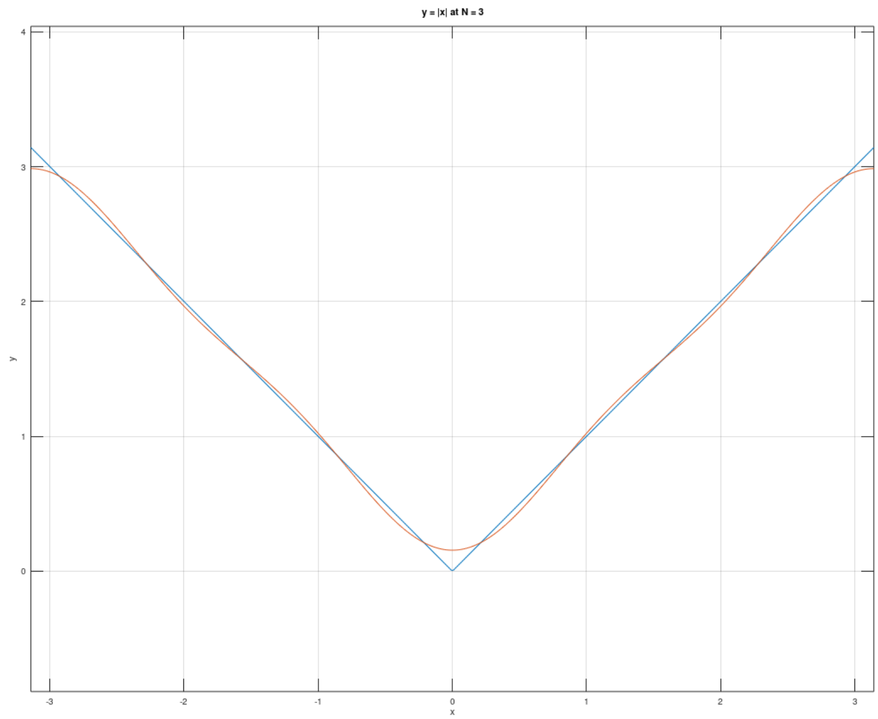


Figure 4.4: Fourier series even function at N = 3

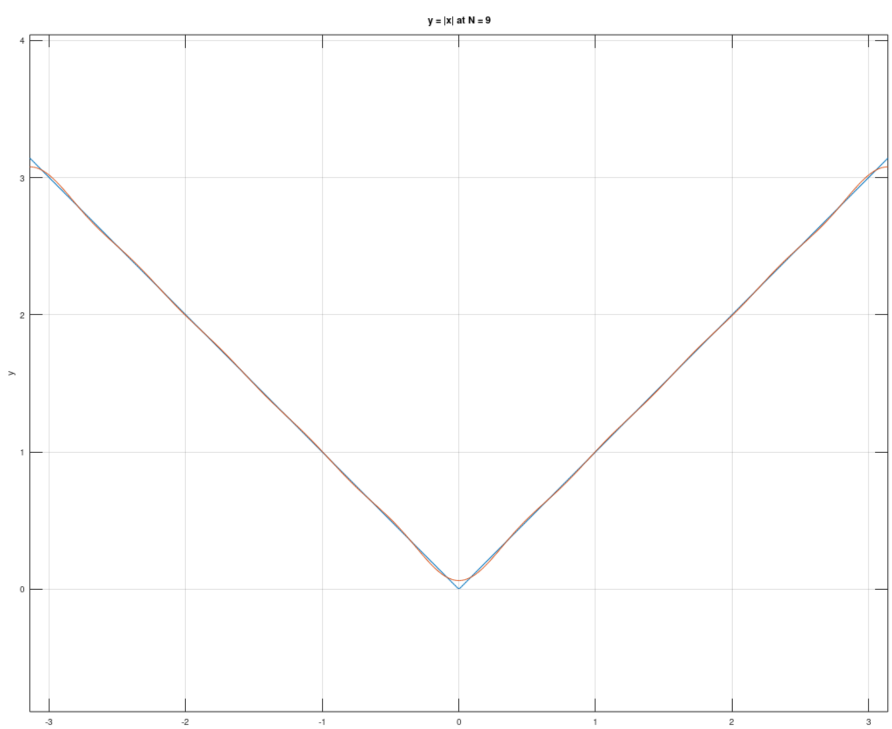


Figure 4.5: Fourier series even function at N = 9

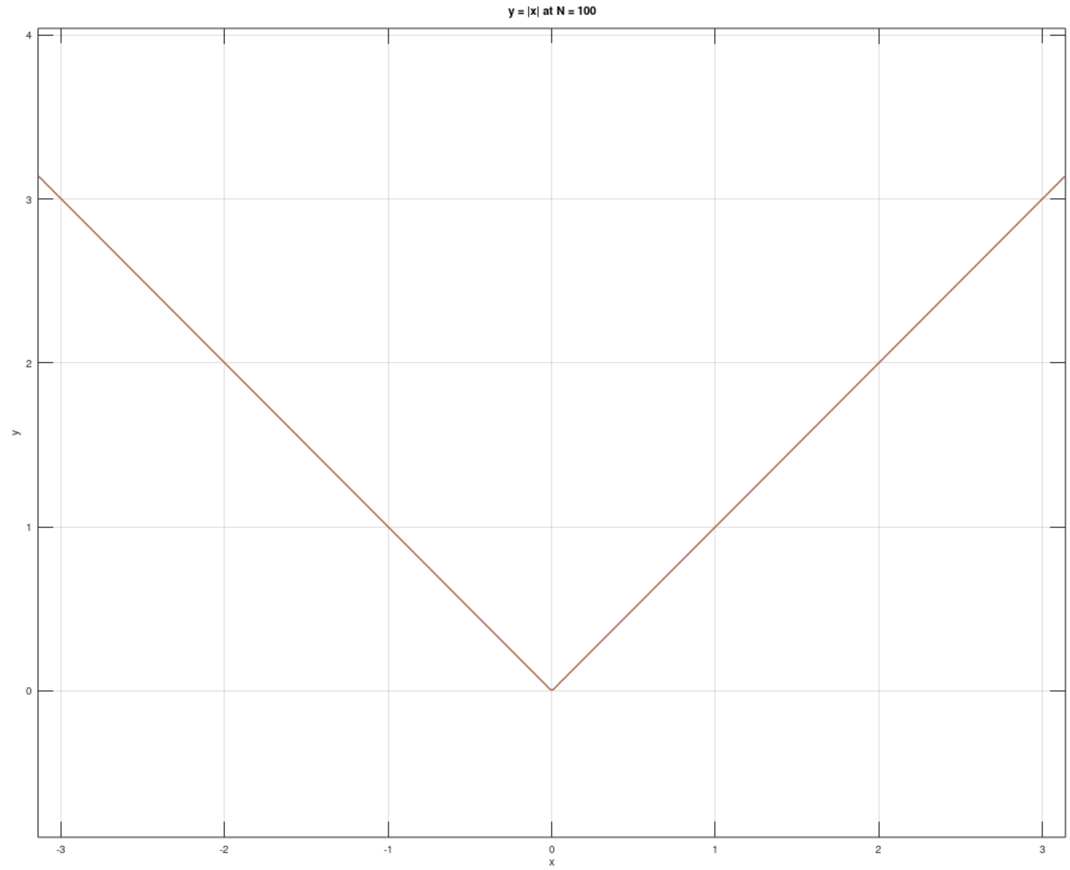


Figure 4.6: Fourier series even function at N = 100

LAB 5: Linear and Circular Convolution of two signals

Objective:

1. To perform Linear Convolution:
   a. x[n] = {1, 1, 1} & h[n] = {1, 1, 1}
   b. x[n] = {0, 1, 2, 3, 4} & h[n] = {0, 2, 3, 4, 5}
2. To perform Circular Convolution:
   a. x1 = [1 2 2 0] & x2 = [1 2 3 4]

Discussion:

The basic operation for calculating the output of any linear time invariant system given its input and impulse response is linear convolution. Even if the number of samples in the input and Impulse response signals is not the same. Still, linear convolution is possible. The number of samples in the output of linear convolution is given by L = M + N – 1, where M is the number of samples in x(n), N is the number of samples in h(n).

Circular convolution is the same as linear convolution, except that the signal's support is periodic. Circular convolution is possible only after modifying the signals via a method known as zero padding. In zero padding, zeroes are appended to the sequence that has a lesser size to make the sizes of the two sequences equal. And the output of the circular convolution will have the same number of samples.

Source Code:

```
# 1. Linear Convolution
     x = [1, 1, 1]
     h = [1, 1, 1]
     linCon = conv(x, h)

     subplot(3,1,1)
     stem(x)
     title('x')

     subplot(3,1,2)
     stem(h)
     title('h')

     subplot(3,1,3)
     stem(linCon)
     title('Linear Convolution of x and h')

     x = [0, 1, 2, 3, 4]
     h = [0, 2, 3, 4, 5]
     linCon = conv(x, h)
     subplot(3,1,1)
     stem(x)
     title('x')

     subplot(3,1,2)
     stem(h)
     title('h')

     subplot(3,1,3)
     stem(linCon)
```

```
    title('Linear Convolution of x and h')

# 2. Circular convolution
    x1 = [1, 2, 2, 0]
    x2 = [1, 2, 3, 4]
    cirCon = cconv(x1, x2, 4)

    subplot(3,1,1)
    stem(x1)
    title('x1')

    subplot(3,1,2)
    stem(x2)
    title('x2')

    subplot(3,1,3)
    stem(cirCon)
    title('Circular Convolution of x1 and x2')
```

Output:

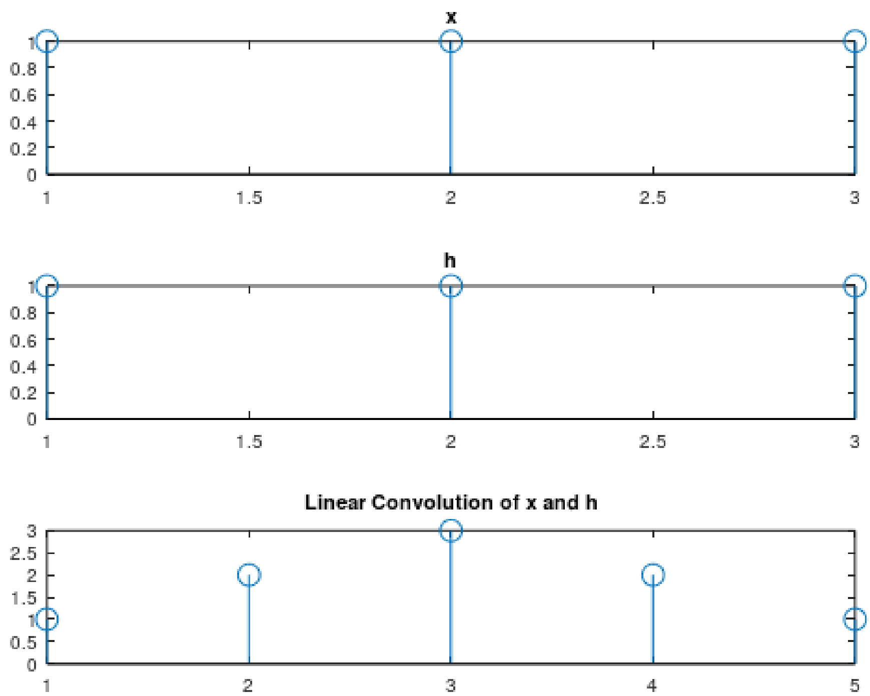


Figure 5.1.a: Linear Convolution

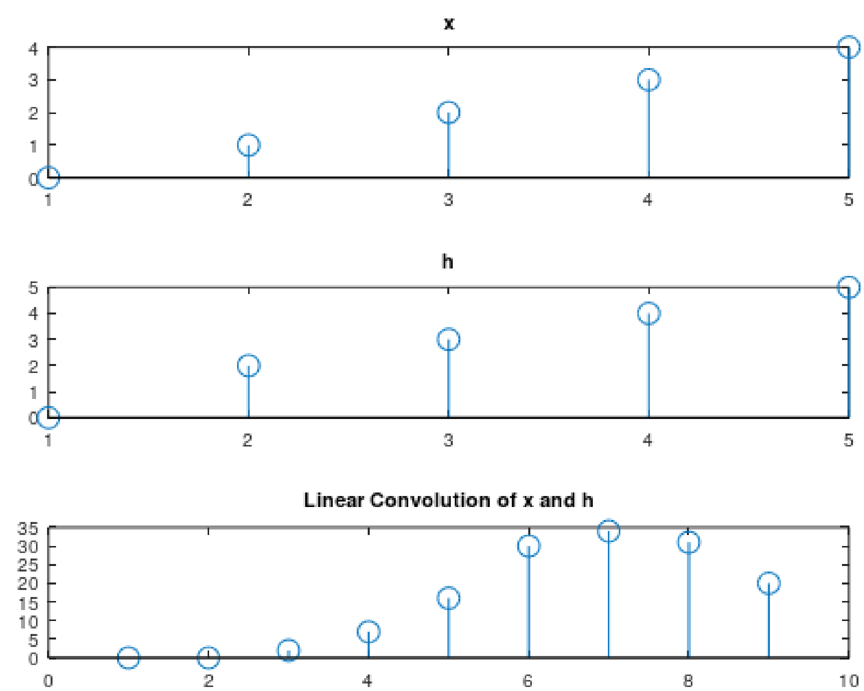


Figure 5.1.b: Linear Convolution

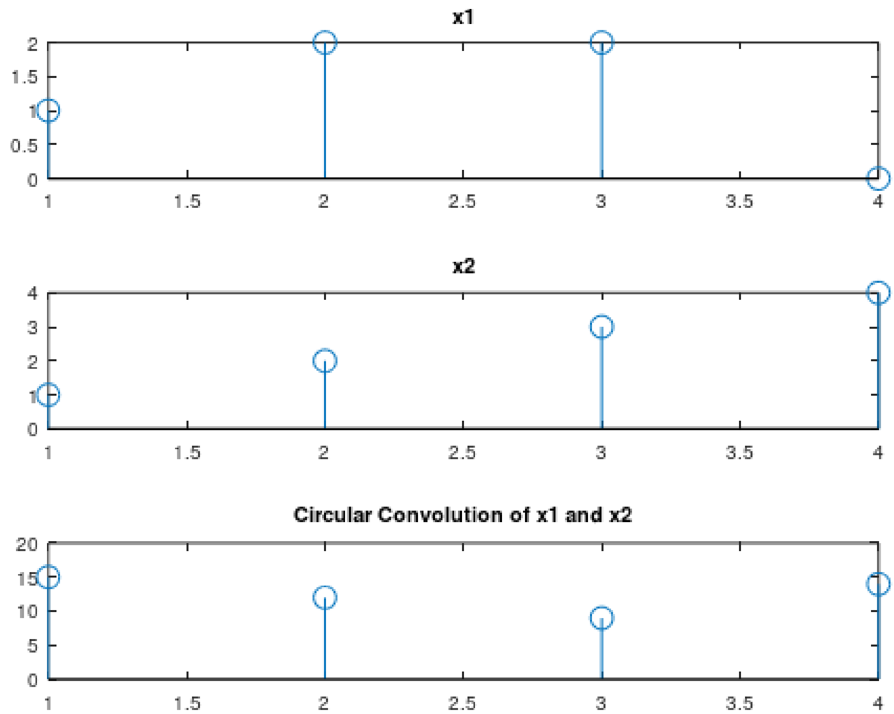


Figure 5.2.a: Circular Convolution