PABNA UNIVERSITY OF SCIENCE AND TECHNOLOGY

Department of Information & Communication Engineering

Faculty of Engineering & Technology

Practical Lab Report

Course Code: ICE-3110

Course Title: Sessional based on Signal and Systems

Submitted By-

Fahomida Ashraf Tani

Roll: 140620

3rd year 1st Semester

Session: 2013-2014

Date of submission: 22-04-17

Submitted To-

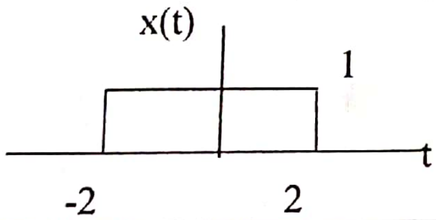
Md. Imran Hossain,

Assistant Professor

Department of Information & Communication Engineering.

Signature:..............

INDEX

EXPERIMENT NO.	NAME OF THE EXPERIMENT	REMARKS	PAGE NO
01	Explain and implement Discrete Fourier transform (DFT) and Inverse Discrete Fourier Transform (IDFT) using MATLAB.		
02	Let $x(n) = \{1,2, 3, 4, 5, 6, 7, 6, 5, 4, 3, 2, 1\}$ Determine and Plot the following sequences, $xl(n) = 2x(n-5) - 3x(n+4)$		
03	Write a Matlab Program to perform following operation –i) Sampling ii) Quantization iii) Coding.		
04	Determine and plot the following sequences. $X(n)= 2\delta(n+2)- \delta(n-4), -5\leq n\leq 5$		
05	Plot following signal operations using user defined function-i) Addition ii)folding.		
06	Plot following signal operations using user defined function i) Signal Multiplication ii) Signal Shifting		
07	Using MATLAB to Plot the Fourier Transform of a Time Function the aperiodic pulse shown below  		
08	To find the amplitude spectrum of the two frequency saignal : $x(t) = Cos(2\pi\, 100t)+cos(2\pi\, 500t)$ and also find approximate the Fourier trans form integral for $0\leq f\leq 800Hz$		
09	Explsin and generate sinusoidal wave with different frequency using MATLAB		
10	Explain and Implementation of following Elementary Discrete Signals Using MATLAB The unit sample sequence ii) the unit step signal iii) the unit ramp signal		
11	Generate the signal $x(n)=K.exp[c.n]$ where $K=2$, $c=-\frac{1}{12}+\frac{j\pi}{6}$, n=0 to 50.		
12	convolution 2 Sequences		

2)

Name of The Program: Explain and implement Discrete Fourier transform (DFT) and Inverse Discrete Fourier Transform (IDFT) using matlab.

Theory:

Discrete Fourier Transform:

mathematics, the discrete. Fourier transform (DFT) converts a finite set list of equally spaced samples of a function into the list of coefficients of a time combination of complex sinusoids ordered by their frequencies, that has there sample values. It can be said to convert the sampled function from its original domain to the frequency domain.

The DFT of $x(K)$ is denoted $X(i) = DFT \{x(K)\}$ and defined

$$X(i) = \sum_{K=0}^{N-1} x(K) W_N^{ik} , \quad 0 \leq i \leq N$$

Drawback of DFT:

The DTFT of a signal $x(k)$ is defined as follows:

$$X(f) = \sum_{k=0}^{\infty} x(k)\, e^{-jk2\pi fT} \quad ; \quad -f_s/2 < f \le f_s/2$$

It has two drawback. One drawback is that a direct evaluation of $X(f)$ using above equation requires an infinite number of floating point operations of FLOPS. This is compounded by the second computional drawback namely that the transform, itself must be evaluated at an infinite number of frequencies.

To address the second limitations, we evaluate $X(f)$ at N discrete values f. In particular consider the following discrete values of frequencies equally spaced over one period of $X(f)$.

$$f_i = \frac{i f_s}{N}, \quad 0 \le i < N$$

It is of interest to view the complex points, $z_i = \exp(j2\pi f_i T)$ corresponding to the discrete frequencies. Then $z_i = \exp(ji2\pi/N)$.

The first limitations can be removed by restructing our considerations to signals of finite duration

$$X(f) = \sum_{K=0}^{N-1} x(K) \exp(-j 2\pi f T)$$

Inverse Discrete Fourier Transform (IDFT):

The continuous time - fourier transform has an inverse whose is always identical to the original signal. The inverse of the DFT, which is denoted $x(K) = \text{IDFT}\{X(i)\}$ is computed as follow:

$$x(K) = \frac{1}{N} \sum_{i=0}^{N-1} X(i) W_N^{-Ki}, \quad 0 \le K < N$$

Here, W_N has been replaced by in the complex conjugate $W_N^* = W_N^{-1}$

To compute the DFT of a sequence by using MATLAB

```
%DFT of a sequence
clc;
clear all;
close all;
N= input('Enter the length of sequence= ');
x= input('Enter the sequence= ');
```

```
n=[0:1:N-1];
K=[0:1:N-1];
WN=exp(-j*2*pi/N);
nK=n*K;
WNnK=WN.^nK;
XK=x*WNnK;
disp('XK=');
disp(XK);
mag =abs(XK)
subplot(2,1,1);
stem(K,mag);
grid on;
xlabel('--->K');
title('magnitude of fourier transform');
ylable('magnitude');
phase =angle(XK)
subplot(2,1,2);
stem(K,phase);
grid on;
xlabel('--->K');
title('phase of fourier transform');
ylable('phase');
```

Input:

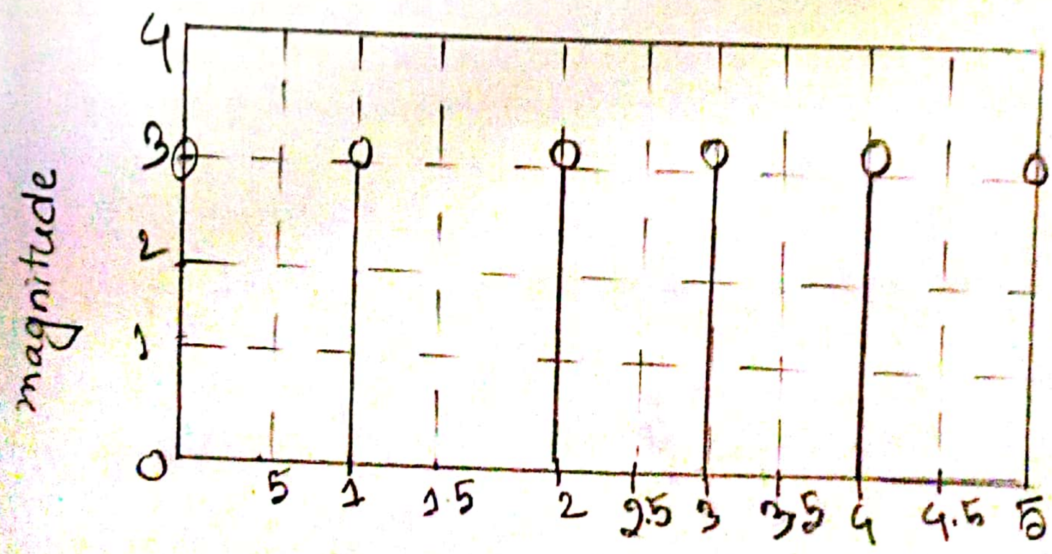
Enter the length of sequence = 6

Enter the sequence = 3

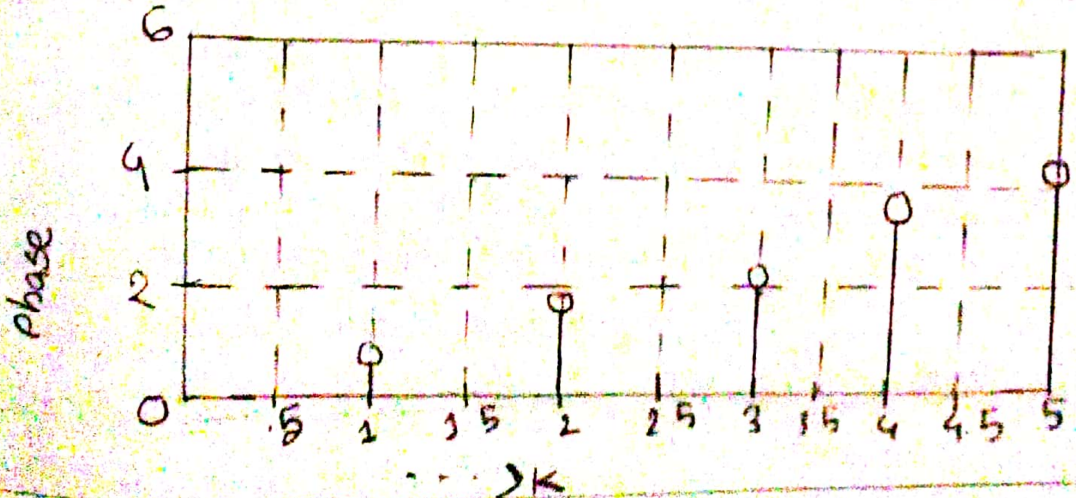
output:

XK = 3·0000+ 0·0000i 3·0000+0·0000i 3·0000+1000i
3·0000+ 0·0000i 3·0000+0·0000i 3·0000+000i

magnitude of Fourier transform



phase of Fourier transform



To compute the IDFT of a Sequence using MATLAB:

programme

```
%.IDFT of a sequence
XK = input('Enter X(K) = ');
[N,M] = size (XK);
if M~=1;
XK = XK;
N=M;
end
xn= zeros (N,1);
K = 0:N-1
for n=0:N-1;
xn(n+1) =exp(j*2*pi*K*n/N)*XK;
end;
Xn =xn/N;
disp('x(n) = ');
disp( xn);
plot( xn);
grid on;
plot (xn);
stem(K,xn);
xlabel('--> ');
ylabel('--> magnitude ');
```

title('IDFT of a Sequence');

outputs

enter X(k)= [4, 3+3i, 2, 3-3i]

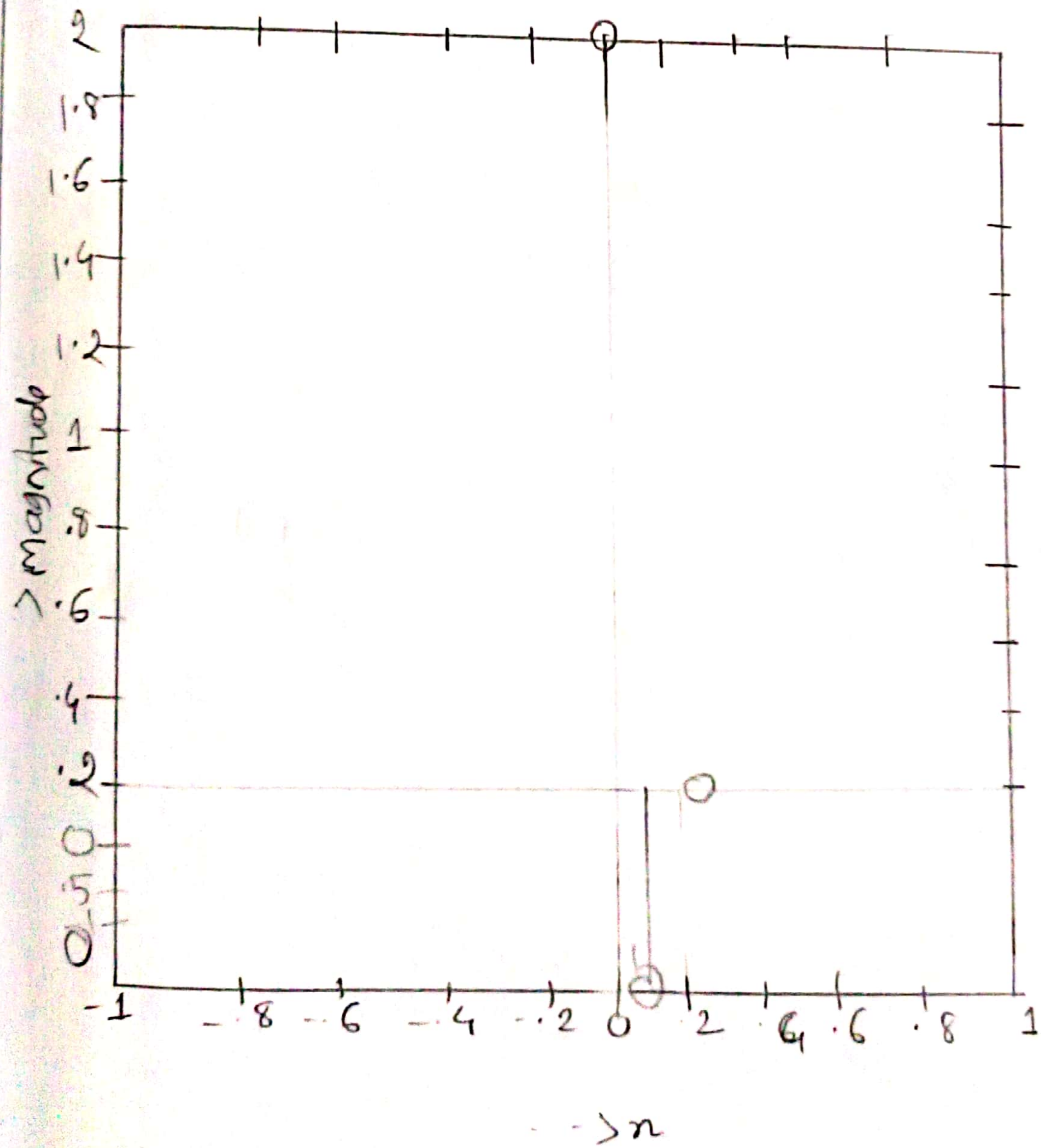
x(n)= 3.000 + .000i

-1.000 + 1.000i

0.000 + .000i

2 .000

IDFT of a Sequence.



Name Of The Programme: Let $x(n) = \{1, 2, 3, 4, 5, 6, 7, 5, 4, 3, 2, 1\}$. Determine and plot the following sequence

$$x_1(n) = 2x(n-5) - 3x(n+4)$$

Theory:

Signal: A signal is defined as a function of one or more variables which conveys information

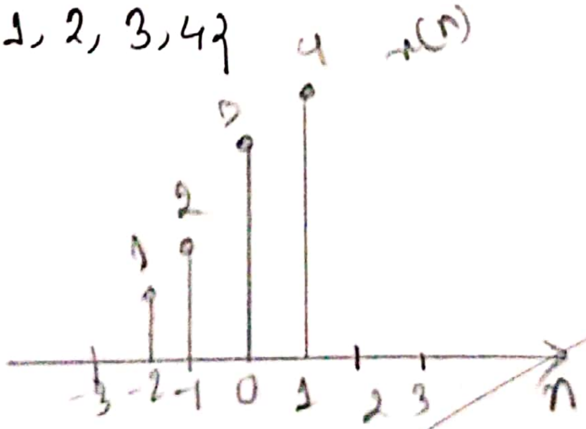
Shifting: Let us consider a discrete time signal $x(n)$. Let $y(n)$ is a signal denote to obtain by shifting the signal $x(n)$ by $(n-n_0)$ that is $y(n) = x(n-n_0)$.

Example:

Suppose $x(n) = \{1, 2, \underset{\uparrow}{3}, 4\}$

determine $y(n) = x(n-1)$

Here $x(n) = \{1, 2, 3, 4\}$



Here $n = +1$

for $n = -2$

$x(-2-1) = x(-2) = 1$

for $n = -1$

$x(-1-1) = x(-2) = 2$

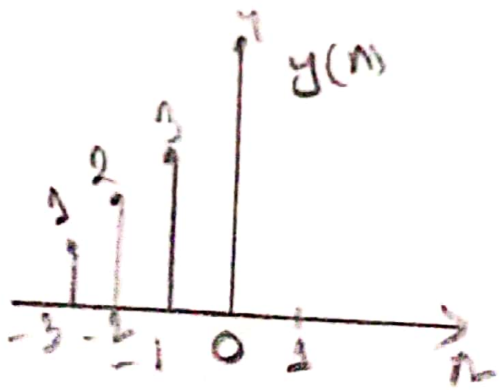
for $n = 0$

$x(0-1) = x(-1) = 3$

for $n = 1$

$x(1-1) = x(0) = 4$

So, $y(n) = \{1, 2, 3, \underset{\uparrow}{4}\}$



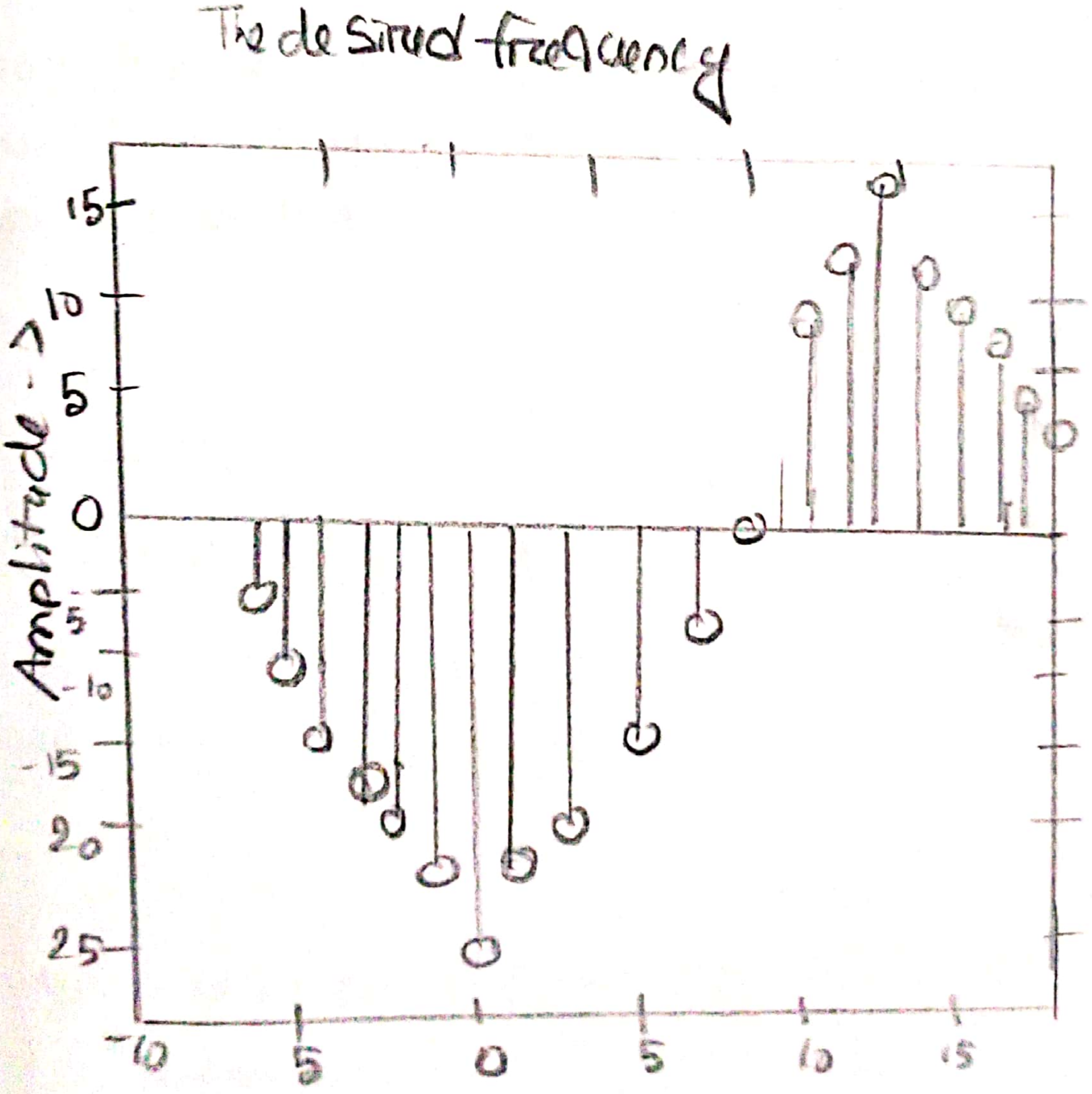
To compute the plot Sequence:

```
function [y,n] = sigadd(x1,n1,x2,n2)
n = min(min(n1), min(n2)): max(max(n2),
    max(n2));

y1 = zeros(1, length(n));
y2 = y1;
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
y=y1+y2;
function [y,n] = sigshift(x,m,n0)
n=m+n0;
y=x;
clc;
clear all;
close all;
n=-2:10;
x=[1:7, 6:-1:1];
[x1,n1]= sigshift(x,n,5);
[x2,n2] = sigshift(x,n,-4);
```

```
[x3,n3]=sigadd(2*x1,n1,(-3)*x2,n1);
stem(n3,x3);
title('The desired sequences');
xlabel('n-->');
ylabel('Amplitude-->');
```

output:



Name of The problem: write a matlab program to perform following operation.

(i) Sampling (ii) Quantization (iii) coding

Theory:

Sampling: This is the conversion of a continuous time signal into a discrete time signal obtained by taking samples of the continuous time signal at discrete time instants. Thus if $x_a(t)$ is the input to the the sampler, the output is $x_a(nT) = x(n)$. where T is called the sampling interval.

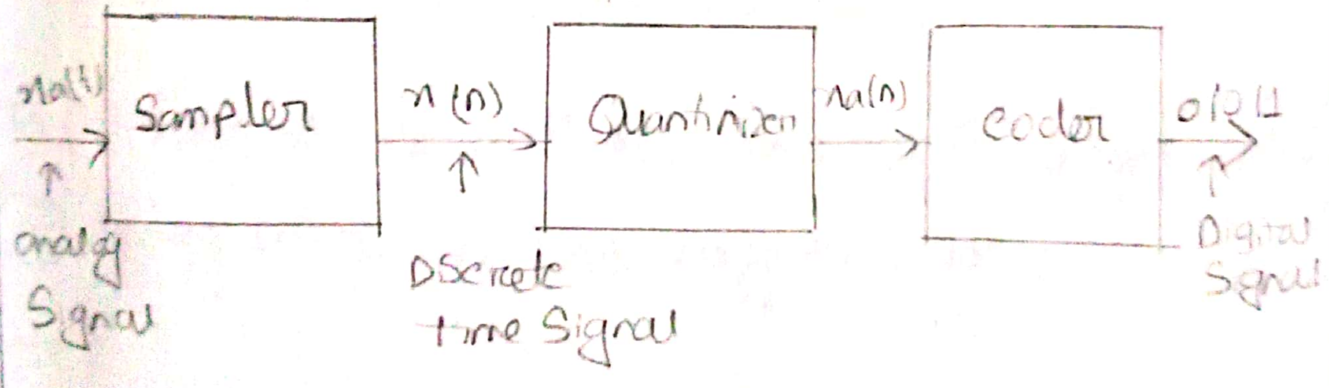


Figure: Basic parts of analog to digital converter

Quantinization: Quantinization in mathematics and digital signal processing is the process of maping a large set of input values to a smaller set.

Rounding and truncation are typical examples of quantinization processes.

The difference between an input value and its quantinized value (such as round-off error) is referred to as quantinization error. A device or algorithmic function that performs quantinization is called a quantizer.

coding: A system of signals used to represent letters or numbers in transmitting messages. The instruction in a computer program.

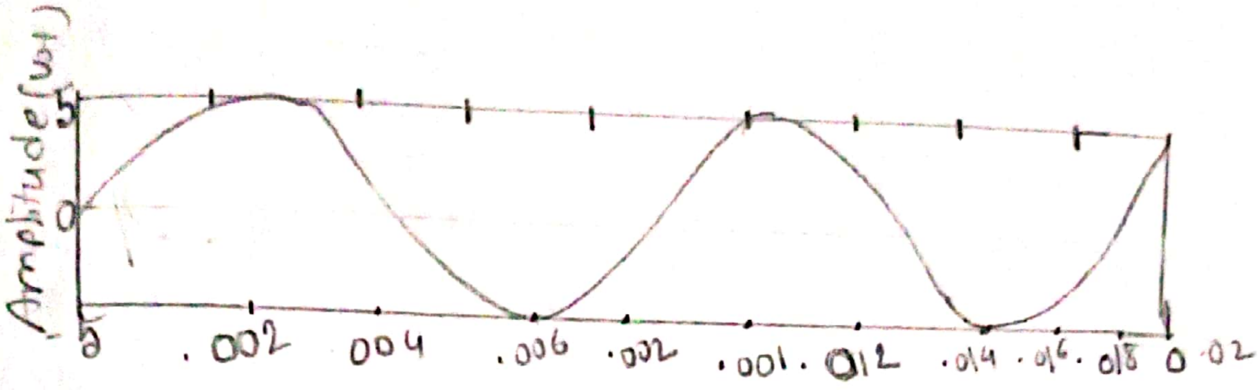
To perform this operation (i) sampling (ii) Quantinization (iii) coding using MATLAB are given bellow:

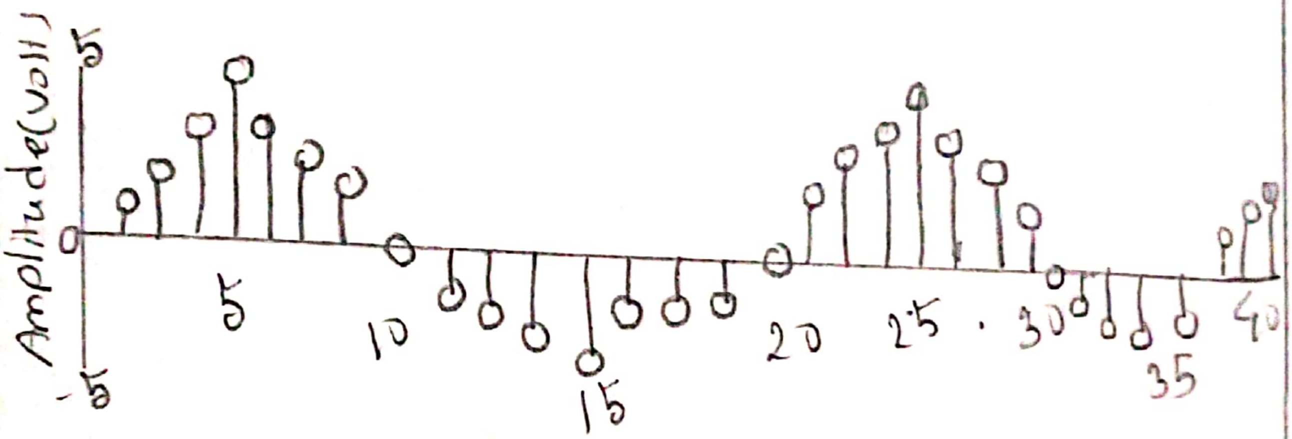
```
clc;
clear all;
close all;
%disp('Input function');
% input function
fprintf('\n\n');
% disp('Transmitting function');
```

```
% transmitting signal generation
f= 100;
T= 1/f;
t = 0:T/100: 2*T;
y = A*sin(2*pi*f*t);
% figure 1)
Subplot( 5,1,1);
plot (t,y,'line width', 3);
ylabel( 'Amplitude (volt)');
xlabel( 'time (sec)');
title( 'Transmitting Signal');
% Sampling
Ts = T/20;
Fs = 1/Ts;
y1 = A*sin(2*pi*f*n*Ts);
% figure 2
Subplot(5,1,2);
Stem( n,y1);
ylabel ('Amplitude (volt)');
xlabel('discrete time ');
title('Discrete time Signal after Sampling');
% additional of DC level
```

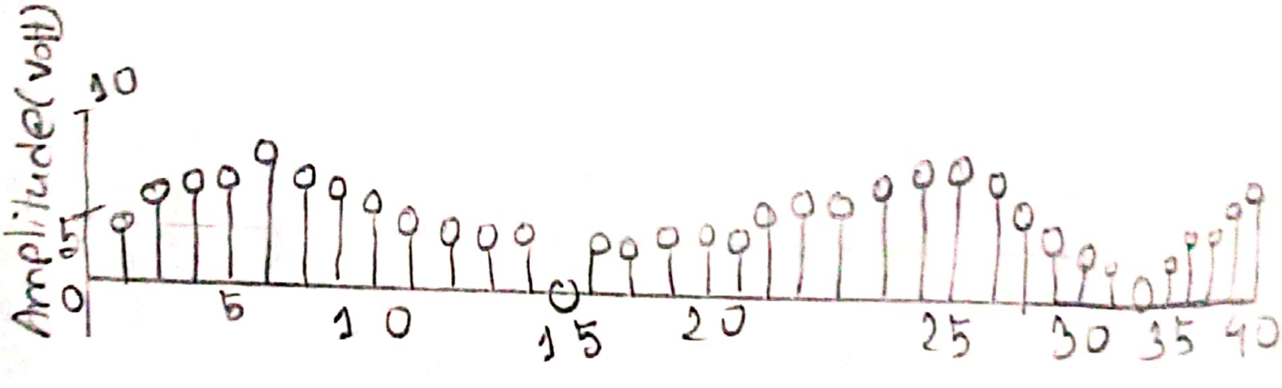
```
y2=A+y1;
%figure 3
Subplot(5,1,3);
Stem(n,y2);
ylabel('Amplitude (volt)');
xlabel('discrete time');
title('Dc level+ Discrete time Signal');
% quantization
y3=round(y2);
% figure(4);
Subplot(5,1,4);
Stem(n,y3);
ylabel('Amplitude (volt)');
xlabel('discrete time');
title('Quantized Signal');
% Binary information(coding)generation
y4=dec2bin(y3);
disp('Binary Information(coding):');
Binary=y4;
disp(Binary);
```

outputs



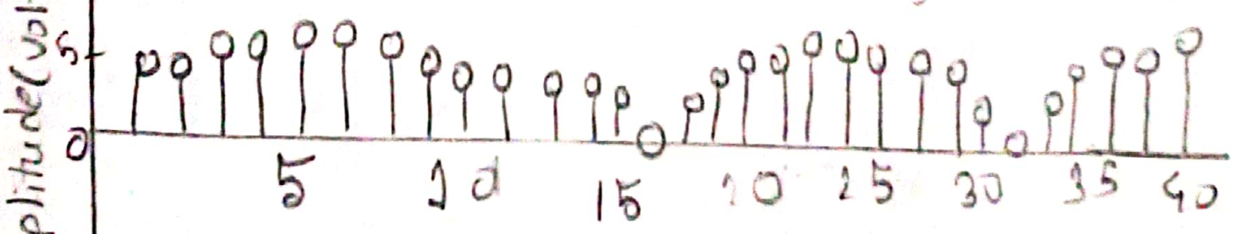


Dc level + Discrete time Signal



discrete time

Quantized Signal



discrete time

Binary information (coding):

0111
1000
1001
1010
1010
1010
1001
1000
0111
0101
0011
0010
0001
0000
0000
0000
0001
0010
0011
0101
0111
1000
1001
1010
1010
1010
1000
0111
0101
0011
0010
0001
0000
0000

0000
0001
0010
0011
0101

Name of The problem: Determine and plot the following sequence, $x(n) = 2\delta(n+2) - \delta(n-4)$, $-5 \leq n \leq 5$

Theory:

Discrete time Signal: A Signal $x(n)$ is said to be discrete time signal if it can be defined for a discrete instant of time (say n).

unit Sample: It is definited in two ways:

(I) Discrete time (II) Continuous time

(a) Discrete time unit impulse: It is defined as

$$\delta[n] = \begin{cases} 0; & n \neq 0 \\ 1; & n = 0 \end{cases}$$

(b) Continuous time unit impulse: It is defined by

$$\delta(t) = \begin{cases} 1; & t = 0 \\ 0; & t \neq 0 \end{cases}$$

the unit Sample sequence is a signal that is zero everywhere; except at n=0 where t=0 its value is unity.

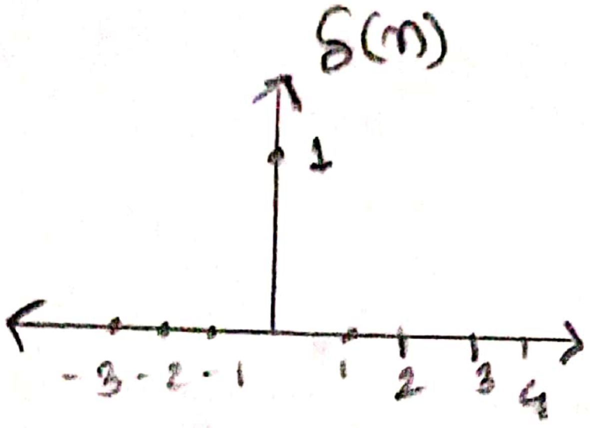


Figure: Graphical representation of the unit Sample signal.

Ex: Suppose,

$$x(n) = \delta(n+2), \quad -2 < n \leq 2$$

we know that,

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$

For $n = -2$

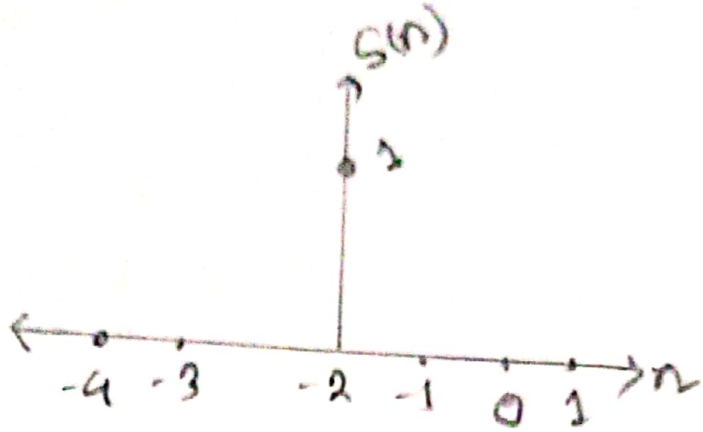
$$\delta(-2+2) = \delta(0) = 1$$

For $n = -1$, $\delta(-1+2) = \delta(1) = 0$

For $n = 0$, $\delta(0+2) = \delta(2) = 0$

For $n = 2$, $\delta(2+2) = \delta(4) = 0$

Graphical representation of $x(n)$:



Plot sequence using MATLAB:

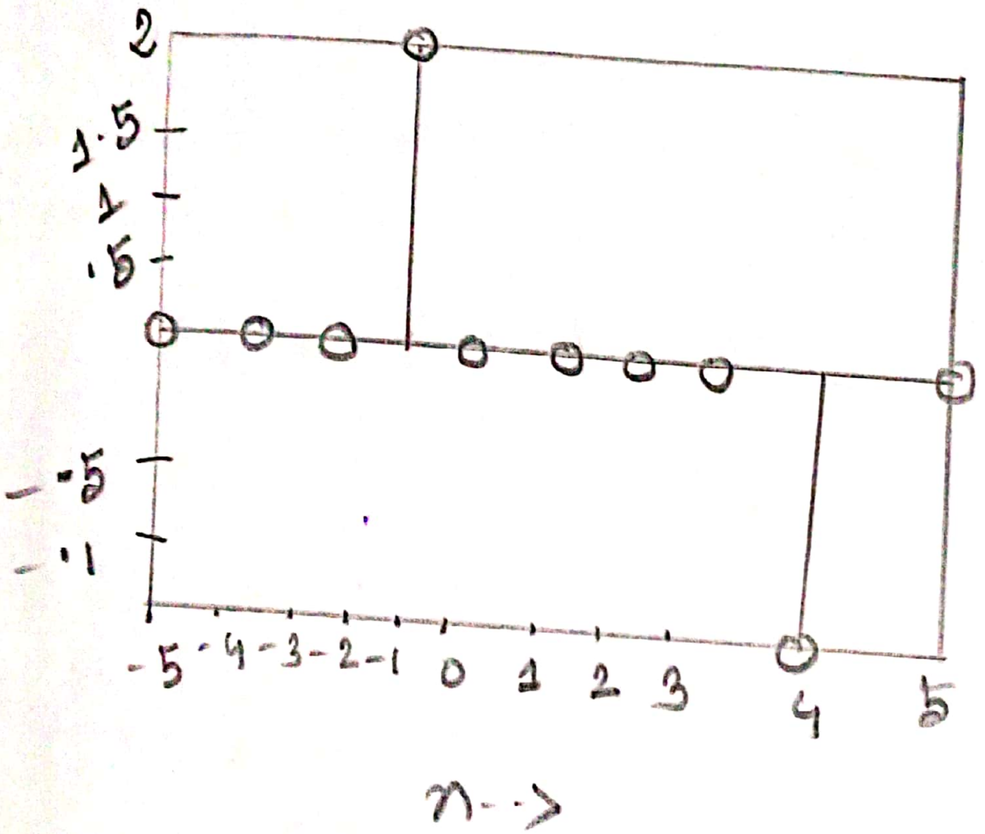
```
function [x,n] = imSeq(n0,n1,n2)
n=[n1:n2];
x=[(n-n0)==0];
end
```

Then.

```
clc;
clearall;
close all;
n=-5:5
x= 2*imSeq(-2,-5,5)-imSeq(4,-5,5)
figure(1);
stem(n,x);
title('The desired sequence');
```

```
xlabel('n-->');
ylabel('Amplitude-->');
```

output:

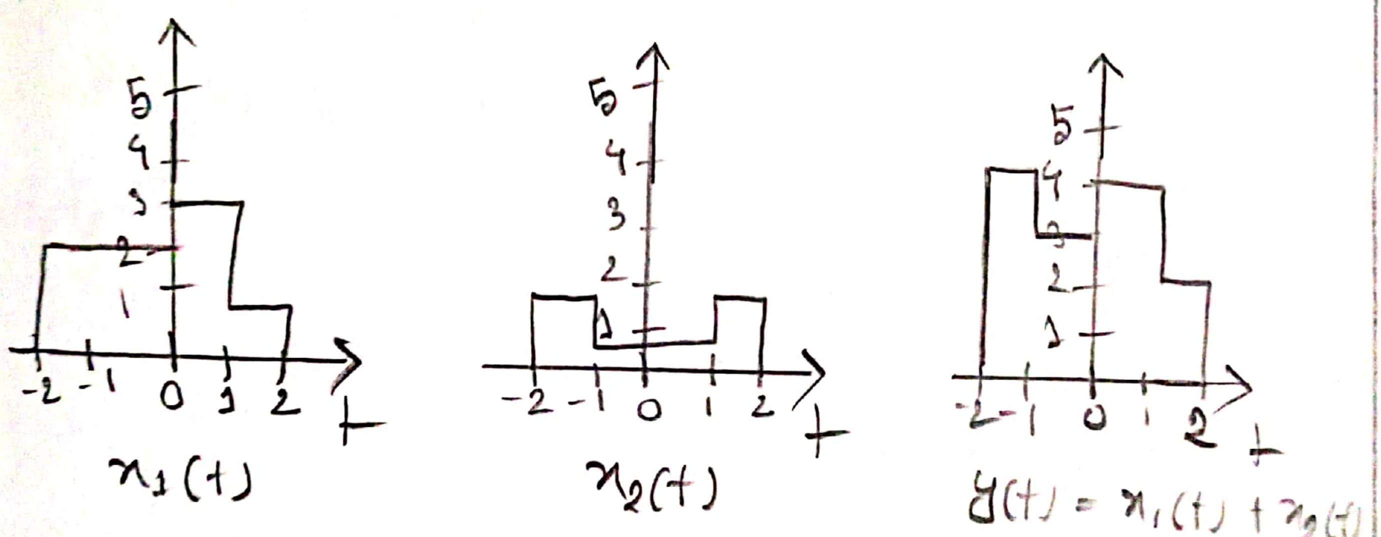


Name of The Program: plot following Signal operations using user defined function (I) Addition (II) Folding

Theory:

Addition of Signals: Consider a pair of continuous time signal $x_1(t)$ and $x_2(t)$. Adding these two signals $x_1(t)$ and $x_2(t)$ result in a signal $y(t)$. The period of the signal is unaltered.

$$y(t) = x_1(t) + x_2(t)$$



(a) Addition of signals

Consider a pair of discrete time signals $x_1(n)$ and $x_2(n)$ these two signal $x_1(n)$ and $x_2(n)$ in a signal produced $y(n)$.

The period of $y(n)$ is unchanged.

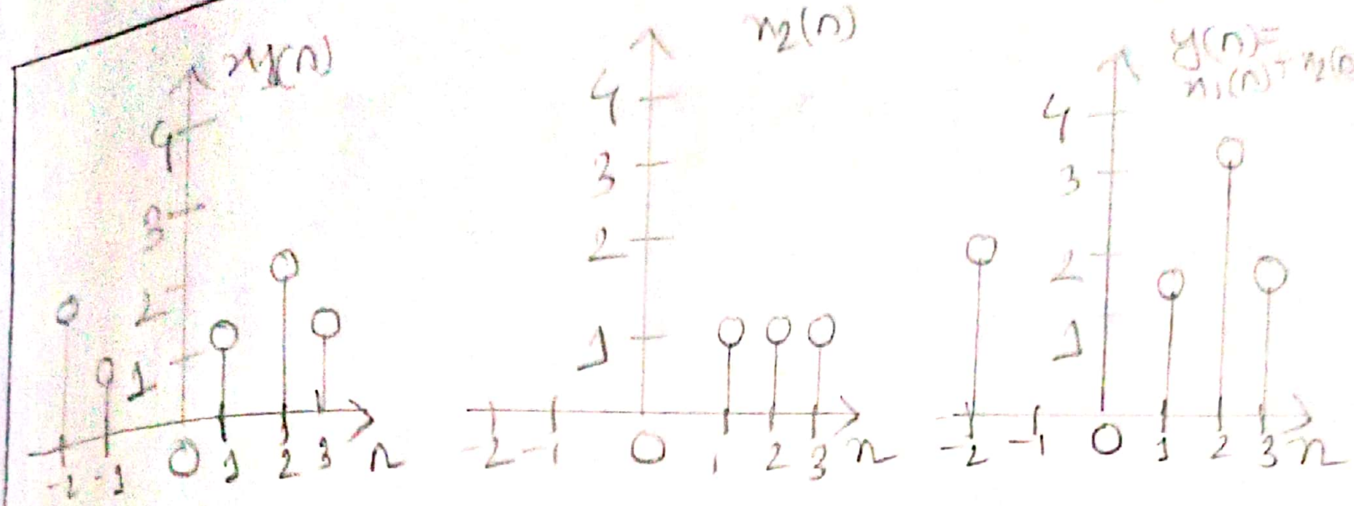


Figure: Addition of Signals.

Folding of Signals: Consider a discrete time signal $x(n)$, Folding means converting the position in positive to negative and negative to positive. The period of $x(n)$ is unchanged. So $x(n) = x(-n)$.

Example:

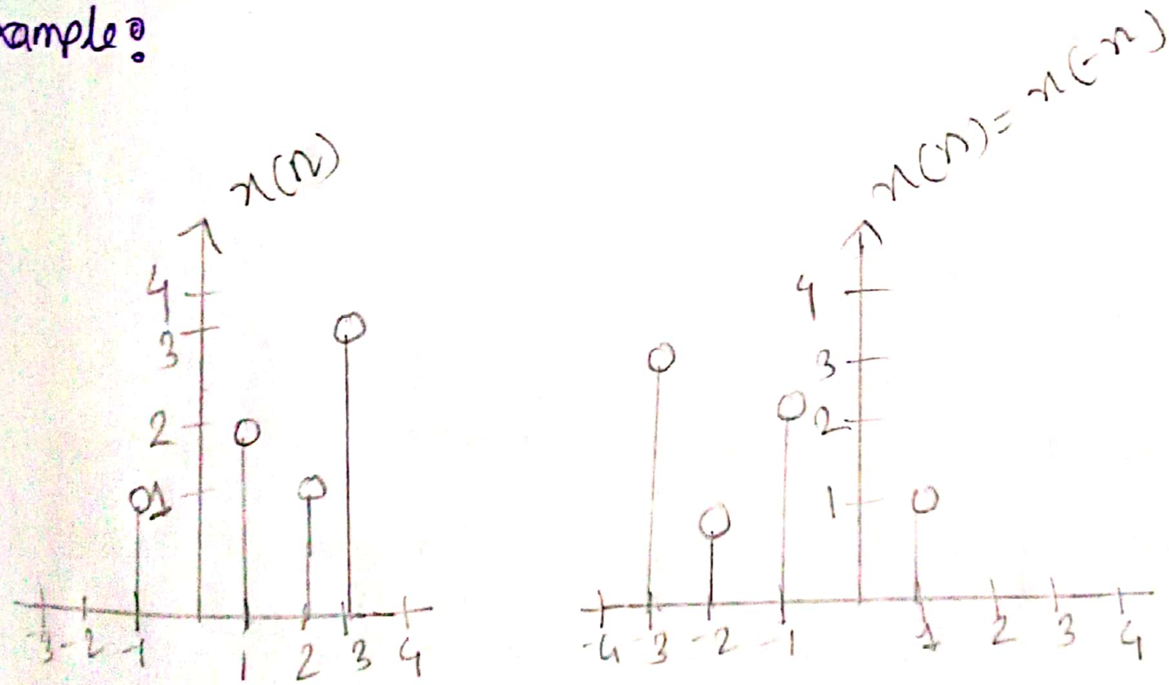


Fig: Folding of Signals.

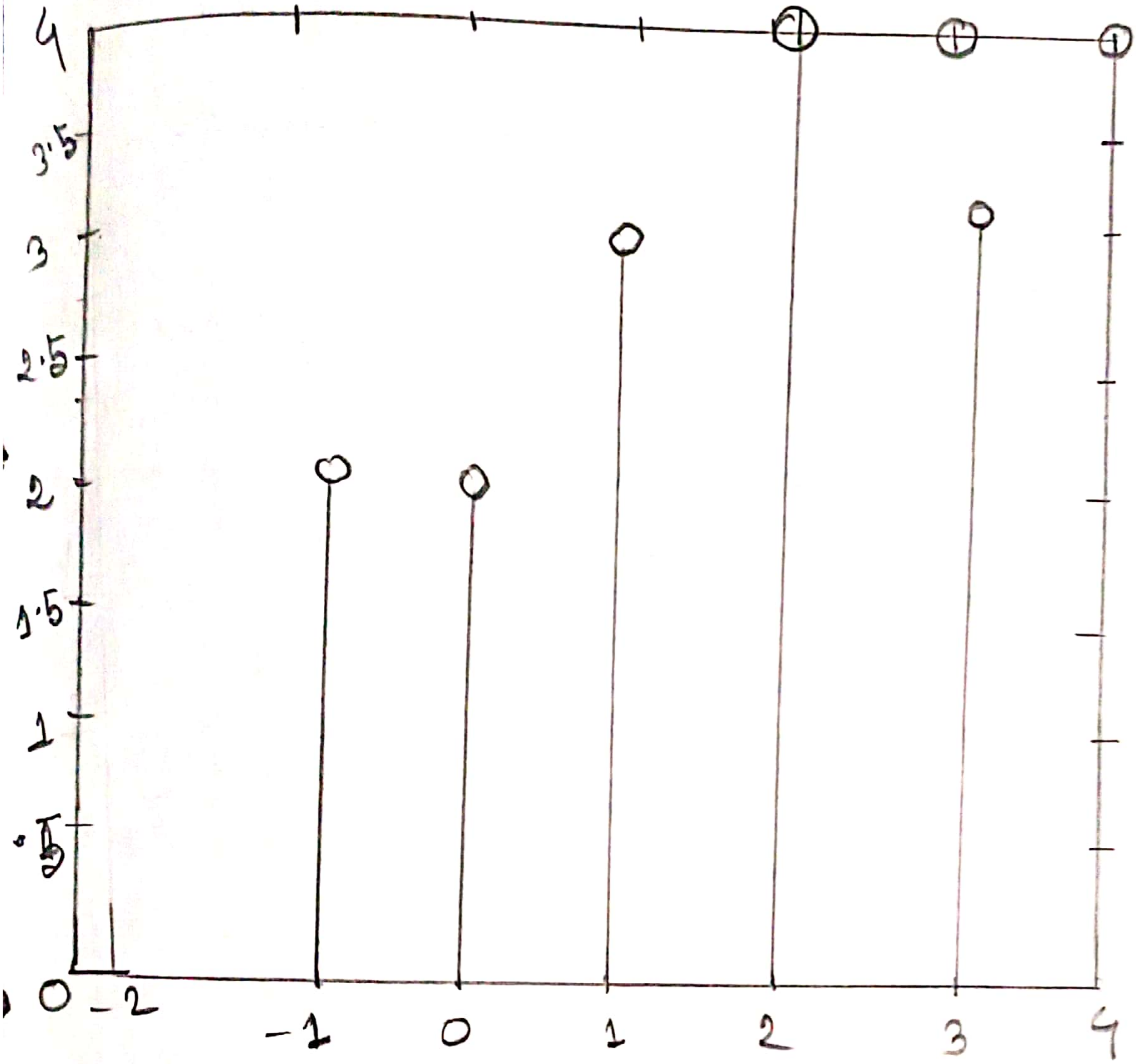
IN MATLAB:

Addition of Signals by using MATLAB:

```
clc;
clear all;
close all;
% x1=[4 2 6 1 4 9];
% n1 = -2:3;
% x2 =[4 3 5 1 1 10];
% n2 = -1:3;
% [y n] = sigadd(x1,n1,x2,n2);
% stem (n,y);
n1 = 0:4;
x1 =[0 1 2 3 4];
n2 = -2:2;
x2 =[2 2 2 2 2];
n = min(min(n1),min(n2)):max(max(n1),max(n2));
y1 = zeros(1, length(n)); y2 = y1;
y1(find((n>=min(n1))&(n<=max(n1))==1))=x1;
y2(find((n>=min(n2))&(n<=max(n2))==1))=x2;
y = y1+y2;
stem(n,y)
```

outputs:

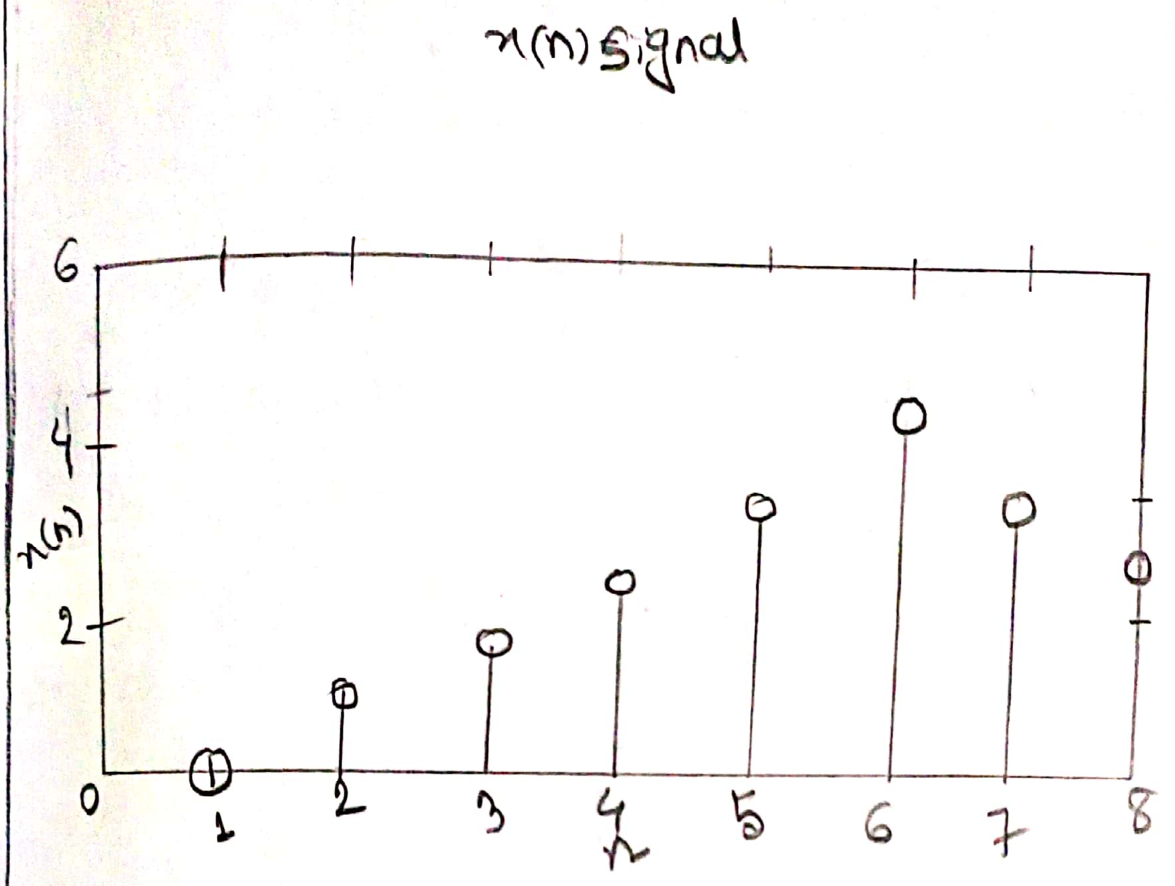
Figure 1

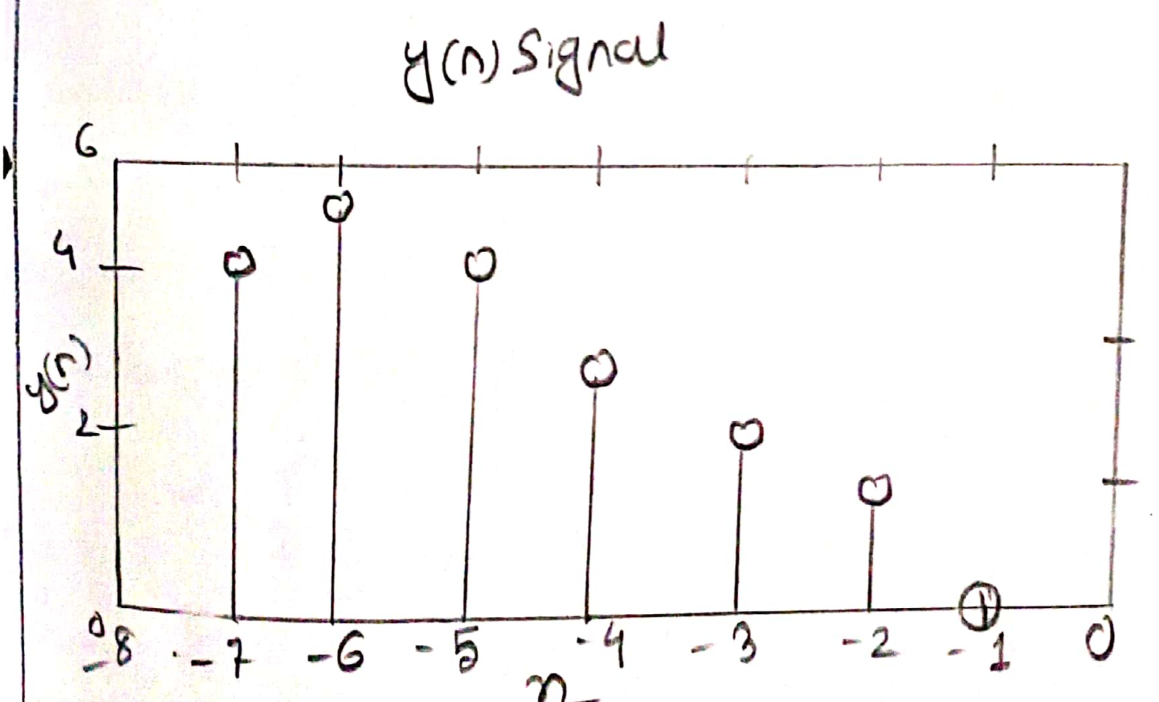


Folding of Signals using MATLAB:

```
% function [y,n] = sigfold (x,n),
% n = - fliplr(n);
% y = fliplr (x);

clc;
clear all;
close all;
% folding a discrete time signal
n = 0:8;
x = [0 0 1 2 3 4 5 4 3];
Subplot(2,1,1);
Stem(n,x);
title('x(n) signal');
xlabel('n');
ylabel('x(n)');
m = -fliplr (n); y = fliplr (x);
Subplot (2,1,2);
stem(m,y);
title ('y(n) = x(-n)');
xlabel('n');
ylabel ('y(n)');
```


x(n) Signal
6
4
2
0
x(n)
1
2
3
4
5
6
7
8
n


y(n) Signal
6
4
2
0
y(n)
-8
-7
-6
-5
-4
-3
-2
-1
0
n

Name of the program: plot following signal operations using defined function.

(i) Signal multiplication (ii) Signal shifting

Theory:

Signal: A signal is defined as a function of one or more variables which conveys information. It is also a physical quantity with time or any other independent variable.

Example: ECG Signal. speech signal.

multiplication of signals: Multiplication of signal is the basic operation on signals.

consider a pair of discrete time signal $x_1(n)$ and $x_2(n)$. multiplication of these two discrete time signals, $x_1(n)$ and $x_2(n)$ and resulted output signal $y(n)$ this, $y(n) = x_1(n) \cdot x_2(n)$

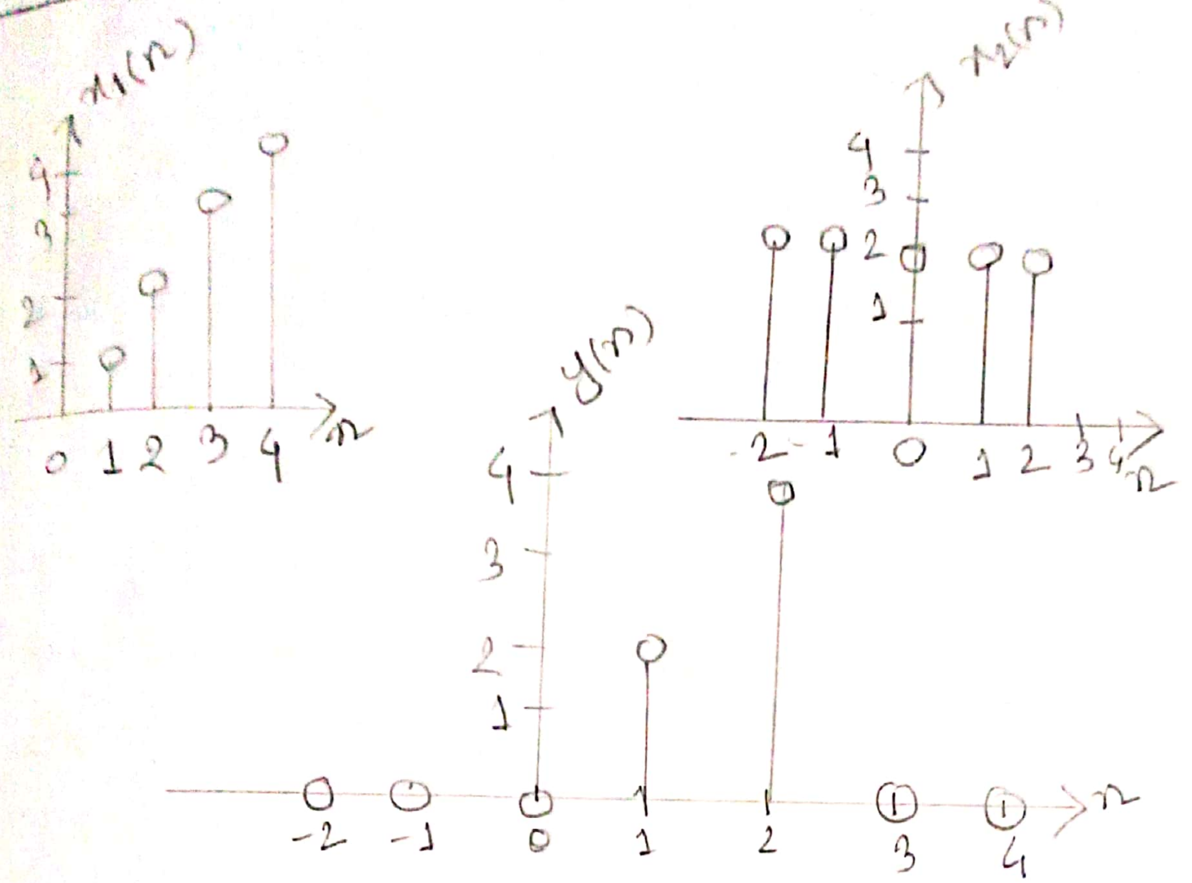


Fig: multiplication of signal.

Shifting of signal: Let us consider a discrete time signal $x(n)$. Let $y(n)$ is a signal denote to obtain by shifting the signal $x(n)$ by $(n-n_0)$ that is $y(n)=x(n-n_0)$

Shift $h(-k)$ by n_0 to the right (left) if n_0 is positive (negative) to obtain $h(n_0-k)$

Ex:

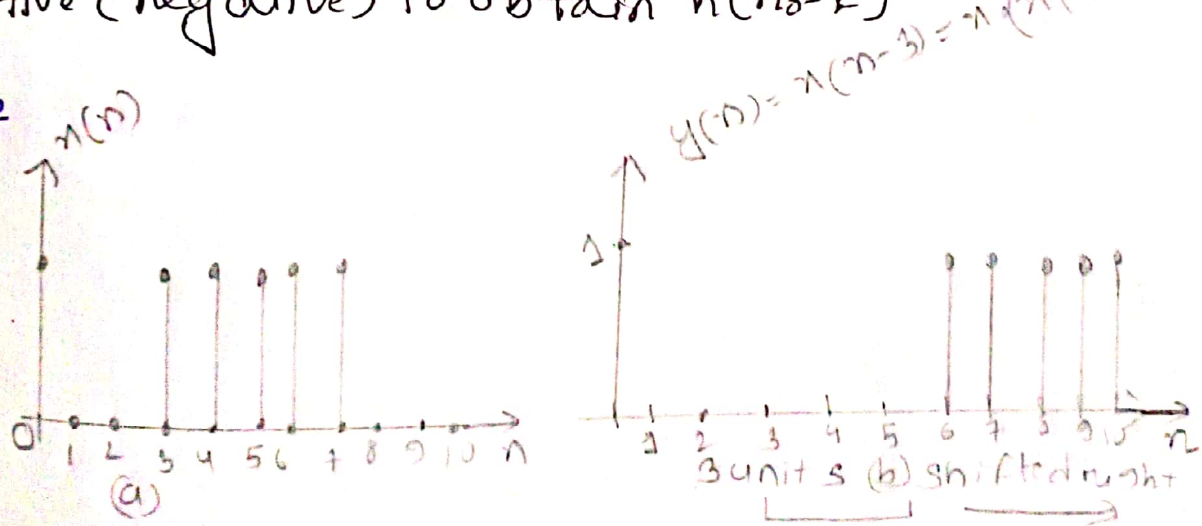
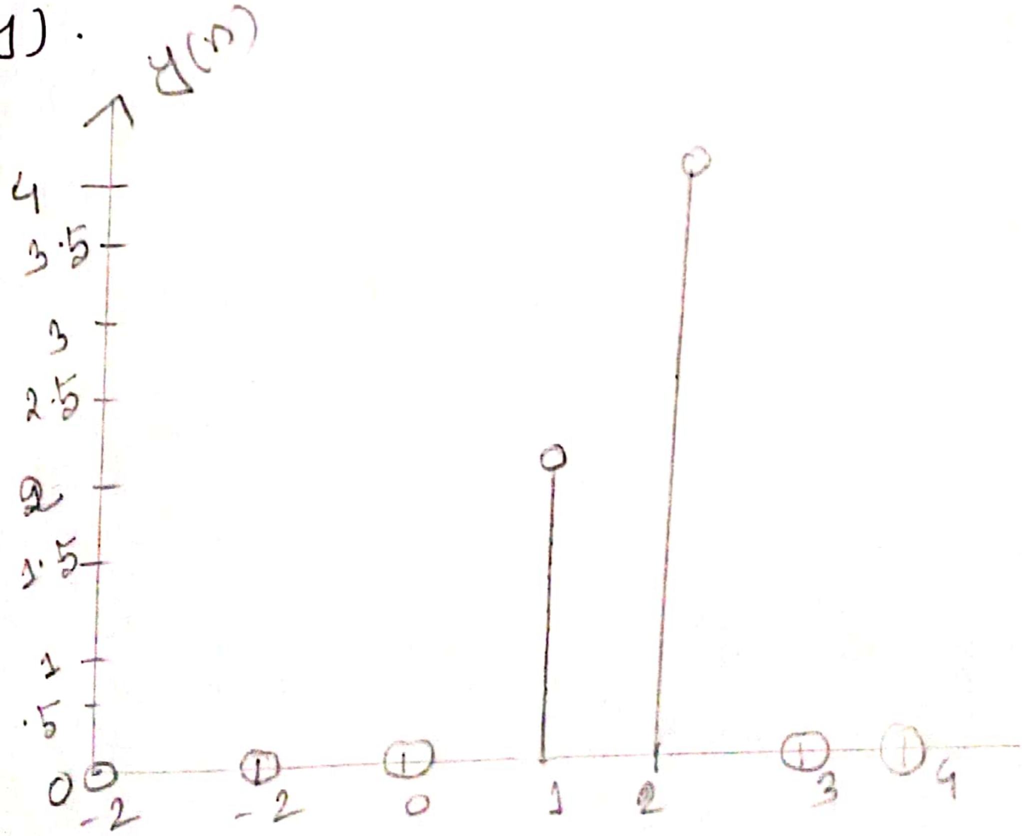


Fig: (a) original positive signal $x(n)$ Fig (b): Right shifted signal $x(n-3)$

Signal multiplication using MATLAB:

```
clc;
clear all;
close all;
% function [y,n] = sigmult(x1,x2,n1,n2)
n1 =0:4
n2 =-2:2
x1 =[0 1 2 3 4];
x2 =[2 2 2 2 2];
n = min(min(n1),min(n2)): max(max(n1)), max(n2));
y1 = zeros(1, length(n));
y2= y1;
y1(find(n>=min(n1)) & (n<=max(n1))==1)) = x1;
y2(find((n>min(n2)) & (n<=max(n2))==1)) = x2;
y = y1.*y2;
stem(n,y).
```

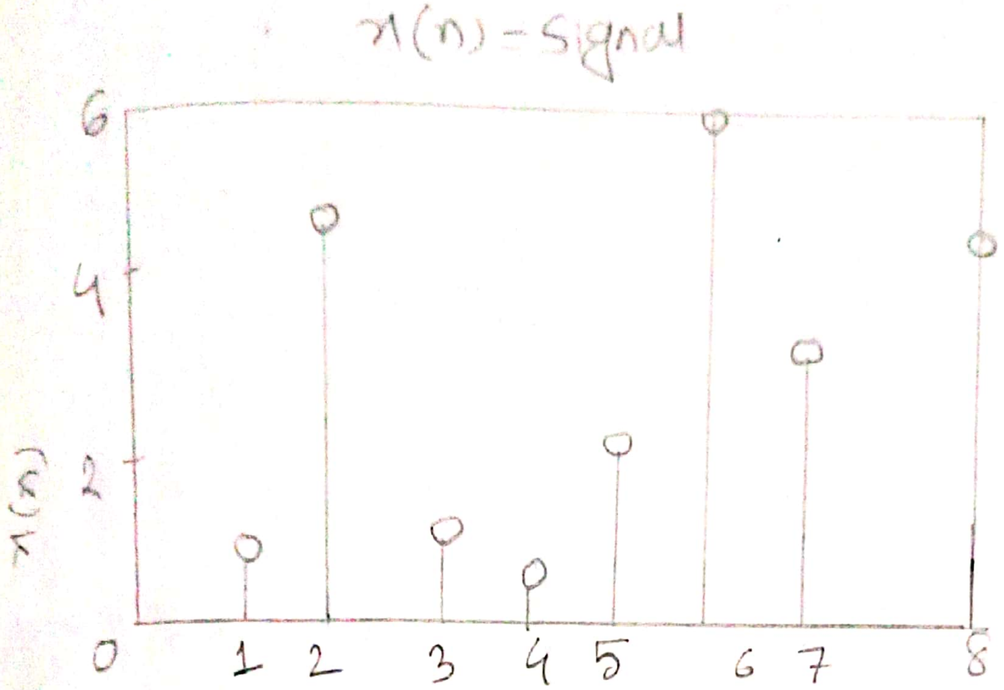
output:

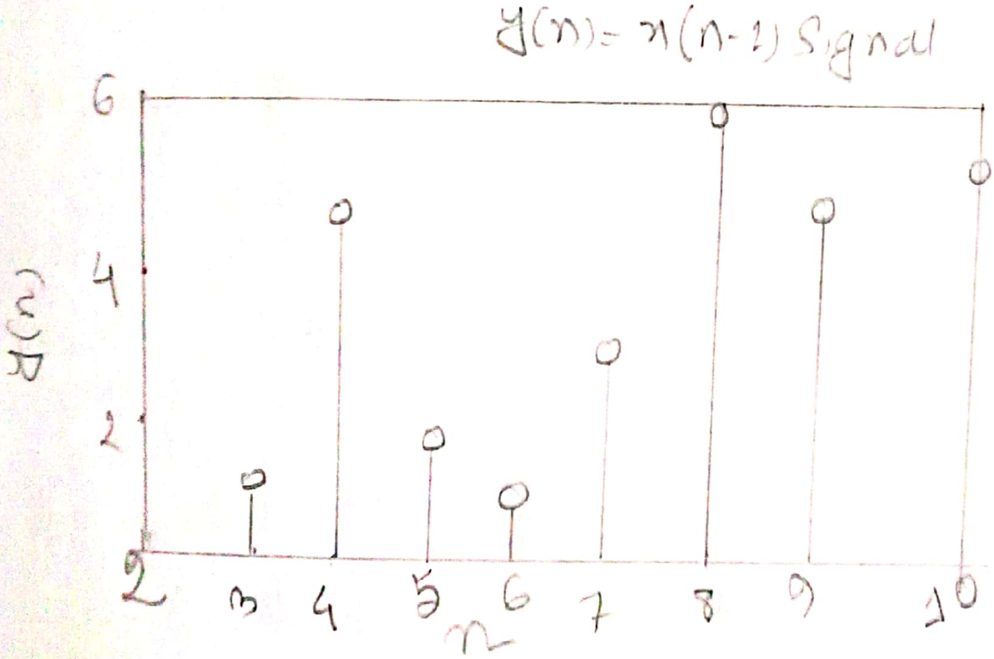


Signal shifting using MATLAB:

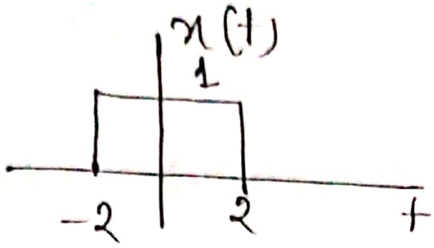
```
% Shifting a non-function Discrete time signal.
clc;
clear all;
close all;
n = 0:8;
x=[0 1 5 2 1 3 6 9 5];
subplot(2,1,1);
stem(n,x);
title('x(n) signal');
xlabel('n');
ylabel('x(n)');
m = n+2; y =x;
subplot(2,1,2);
stem(m,y);
title('y(n)=x(n-2) Signal');
xlabel('n');
ylabel('y(n)');
```

outputs:





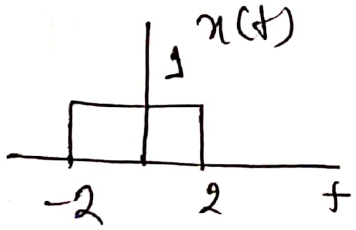
Name of the problem: Using MATLAB to plot the Fourier Transform of a Time function the aperiodic pulse shown bellow.



Theory:

Fourier Transform: A function derived from a given function and representing it by a series of sinusoidal functions.

The aperiodic pulse shown below.



has a Fourier transform:

$$X(if) = 4\,\mathrm{Sinc}(4\pi f)$$

This can be found using the table of Fourier Transforms. We can use MATLAB to plot this transform. MATLAB has a built in sinc function. However the definition of the MATLAB sinc function is slightly different than the one used in class and on the Fourier transform table.

$$\text{Sinc}(x) = \frac{\text{Sin}(\pi x)}{\pi x}$$

Thus in MATLAB we write the transform X, using Sinc(4f); Since the π factor is built into the function.

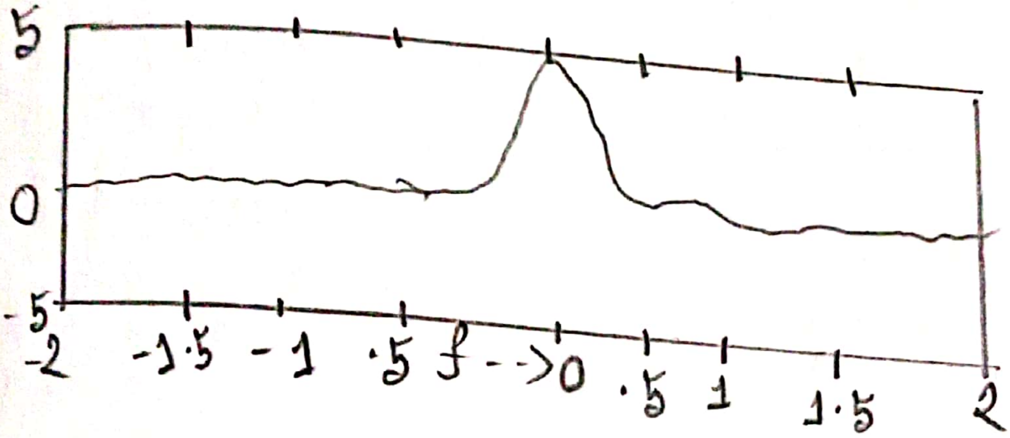
In MATLAB:

```
clc;
clear all;
close all;
f=-2:.01:2;
X=4*Sinc(4*f);
figure(1);
Subplot(3,1,1);
Plot(f,real(X));
title('Real part');
Xlabel('f-->');
Subplot(3,1,2);
Plot(f,abs(X));
title('magnitude part');
xlabel('f-->');
Subplot(3,1,3);
Plot(f,angle(X));
```

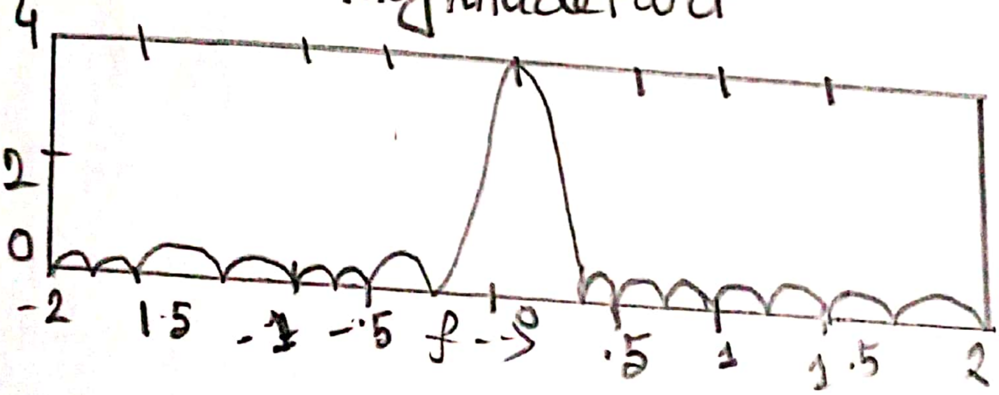
```
title ('phase part');
xlabel('f - >');
```

output:

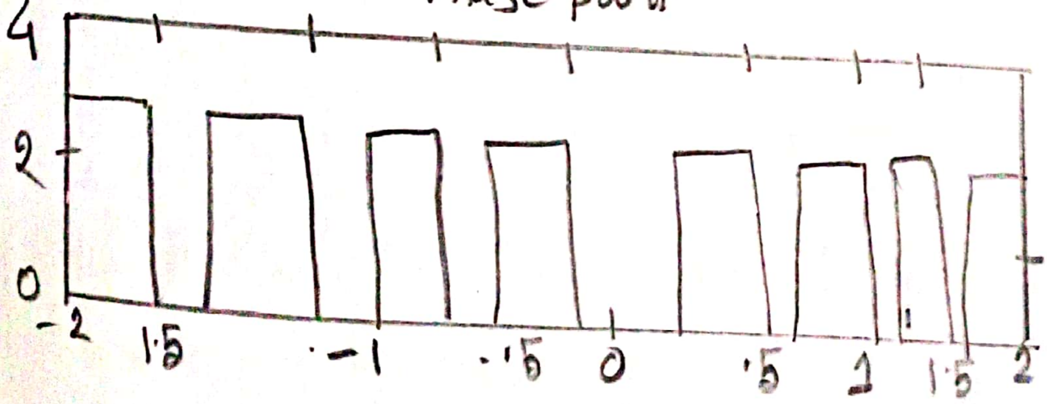
Real part



Magnitude part



Phase part



Name of The Problem: Explain and generate Sinusoidal wave with different frequency using MATLAB

Theory: A Sine wave or Sinusoid is a mathematical curve that describes a smooth repetitive oscillation. It is named after the function Sine of which it is the graph. It occurs often in pure and applied mathematics as well as physics, engineering, signal processing and many other fields.

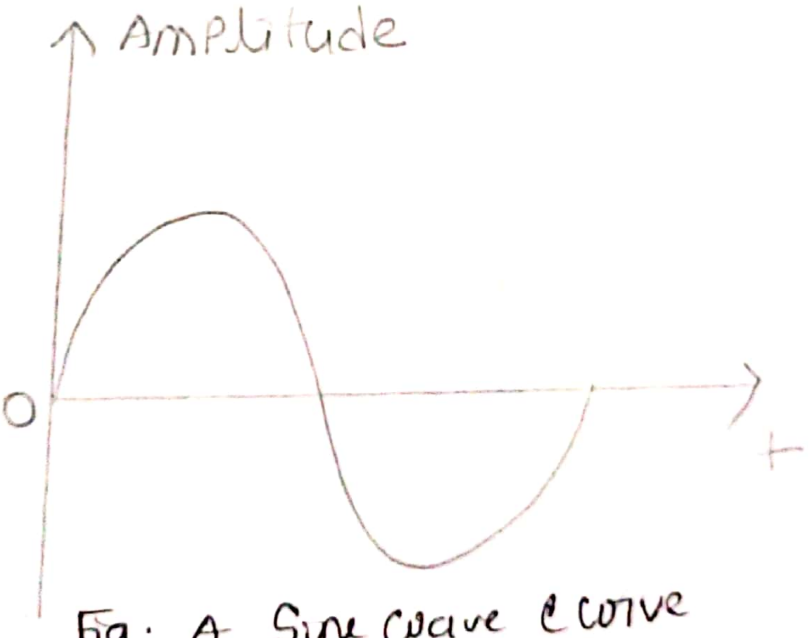


Fig: A Sine wave curve

The equation of the Sinusoidal wave is,

$$z = x + iy = r(\cos\phi + i\sin\phi)$$

$$x = r\cos\phi$$

$$y = r\sin\phi$$

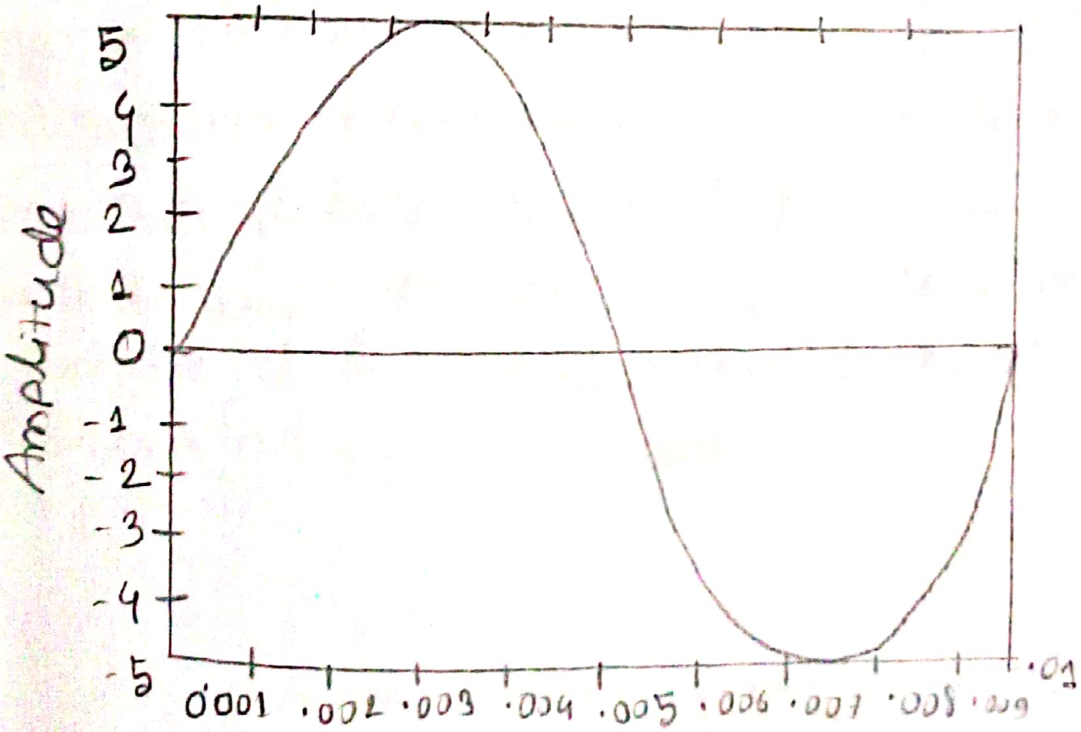
$$\varphi = \omega t$$

Sinusoidal wave with frequency using MATLAB:

```
clc;
clear all;
close all;
A=5;
f=100;
T=1/f;
t=0:T/100:T;
y=A*sin(2*pi*f*t);
plot(t,y);
xlabel('t');
ylabel('Amplitude');
title('Sinusoidal wave');
```

output:

Sinusoidal wave



Name of The Program: Explain and Implementation of following Elementary Discrete Signals using MAT-LAB.

(i) The unit Sample Sequence (ii) The unit Step Signal.

(iii) The unit ramp Signal.

Theory:

The unit Sample Sequence: The unit Sample Sequence is denoted as $\delta(n)$ and is defined as.

$$\delta(n) = \begin{cases} 1, & \text{for } n = 0 \\ 0 & \text{for } n \neq 0 \end{cases}$$

In words the unit sample sequence is a signal that is zero everywhere, except at $n=0$. where its value is unity. The signal is sometimes reffered to as a unit impulse. In contrast to ' . _ Signal $\delta(t)$, which is also called a unit impulse and is defined to be zero everywhere except at $t=0$ and has unit area the unit sample sequence is much less mathematically complicated. The graphical representation is shown in fig:

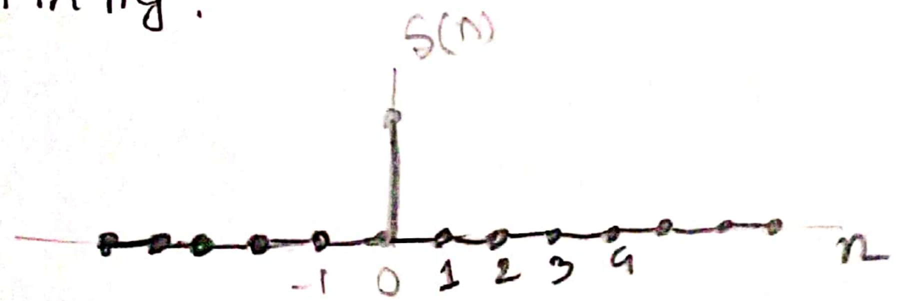


fig: Graphical representation of the unit Sample Signal

The unit Step Signal: The unit Step Signal is denoted as $u(n)$ and is defined as.

$$u(n) = \begin{cases} 1, & \text{for } n \geq 0 \\ 0, & \text{for } n \leq 0 \end{cases}$$

The graphical representation of the unit Step signal is given bellow:

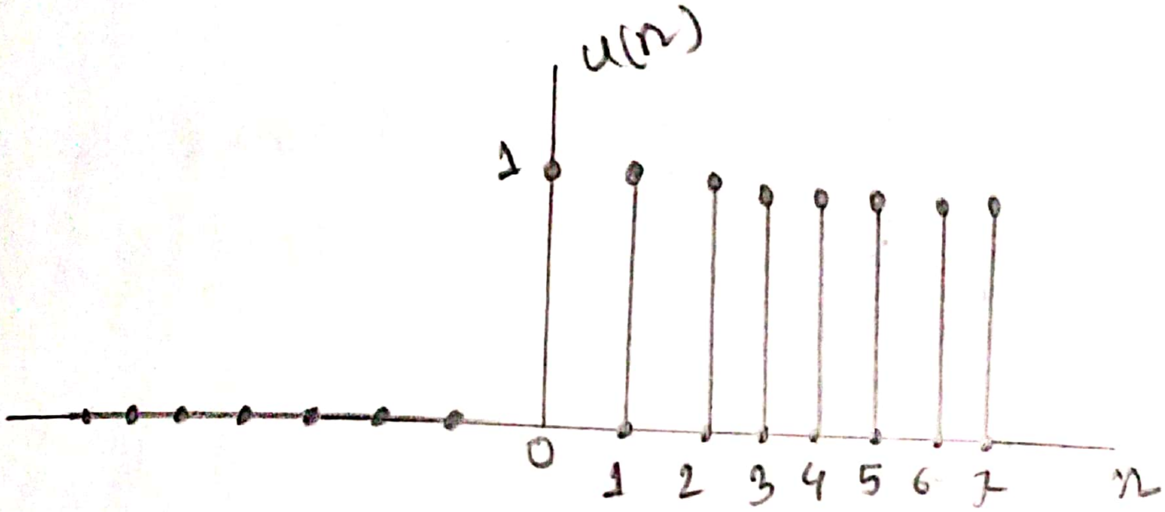


Fig: Graphical representation of the unit Step Signal.

The unit ramp Signal: The unit ramp Signal is denoted as $u_r(n)$ and is defined as,

$$u_r(n) = \begin{cases} n, & \text{for } n \geq 0 \\ 0, & \text{for } n < 0 \end{cases}$$

The graphical representation of unit ramp signal is given bellow

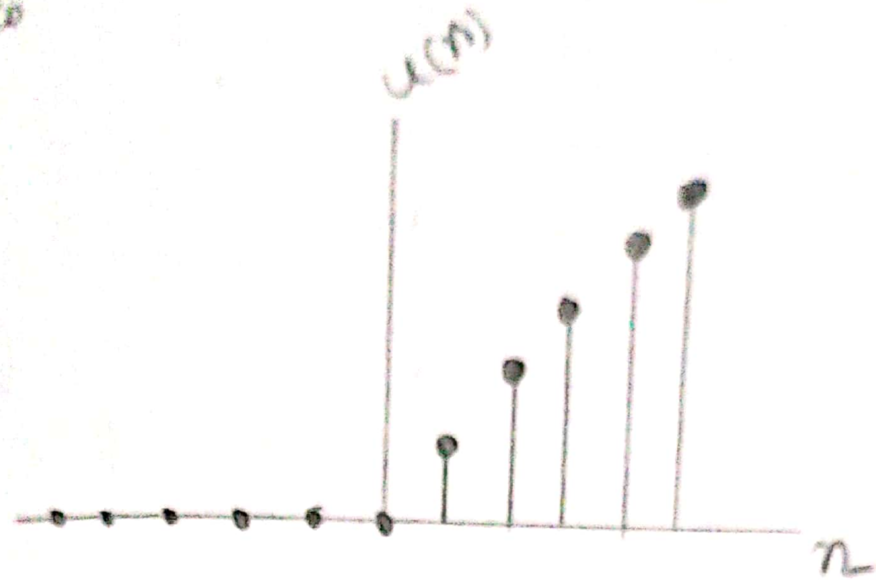


Fig: Graphical representation of unit ramp signal

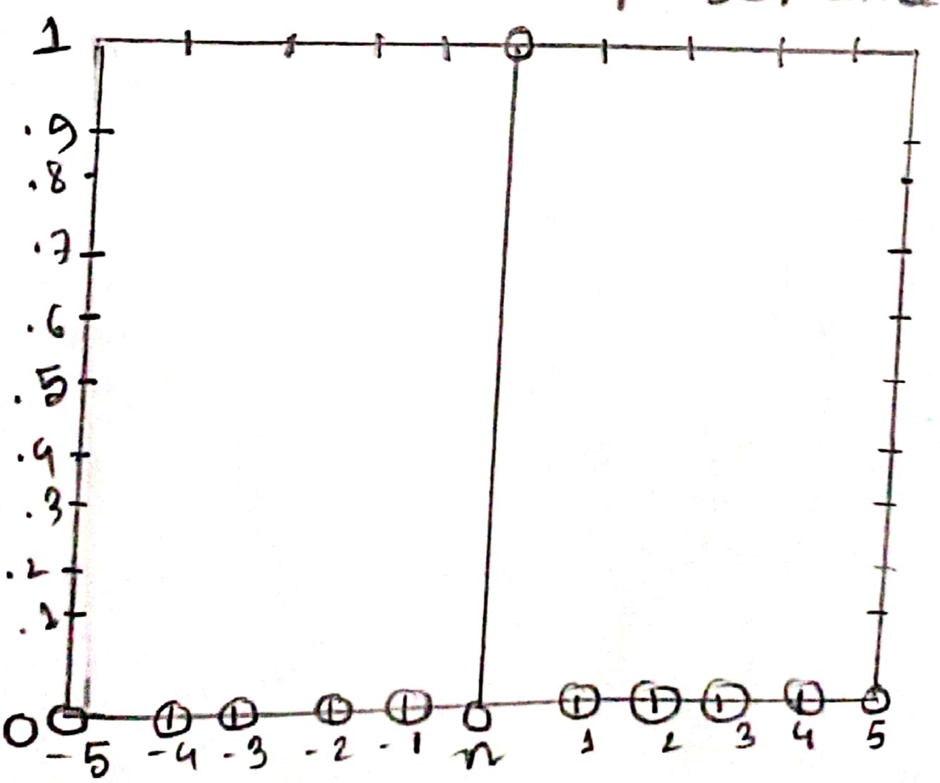
In MATLAB:

The unit Sample Sequence using MATLAB:

```
clc;
clear all;
close all;
%n = [-5 : 5];
% x = 2 * imSeq - (-2, -5, 5);
% stem (n, x);
n = - 5 : 5;
x = [n = = 0];
Stem (n, x)
title ('unit sample Sequence');
xlabel 1 ('n');
ylabel 2 ('x');
```

output:

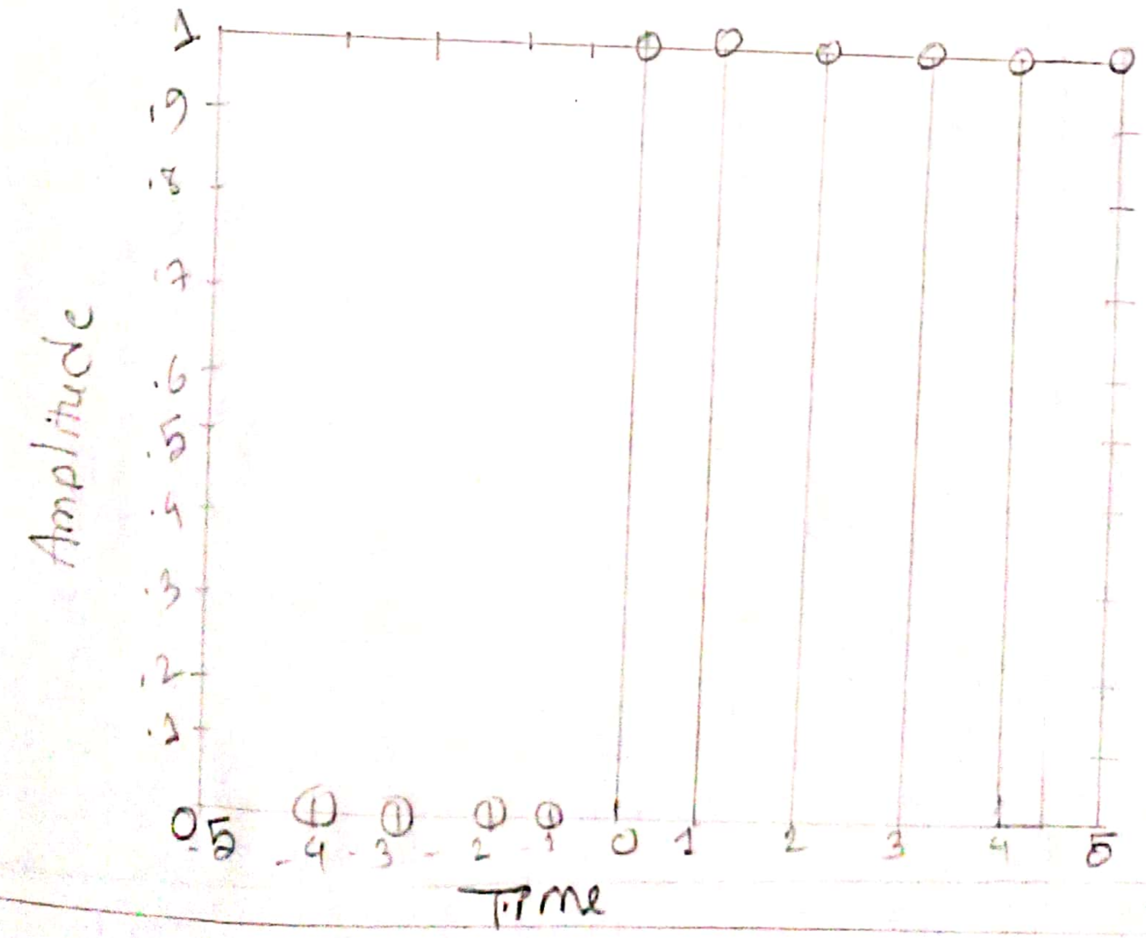
unit Sample sequence



The unit Step Signal using MATLAB:

```
% unit step function
clc
clear all;
close all;
N=5; % No of Sample
n=-N:1:N
x=[zeros(1,N) 1 ones(1,N)]
stem(n,x);
xlabel('Time');
ylabel('Amplitude');
title('unit Step');
```

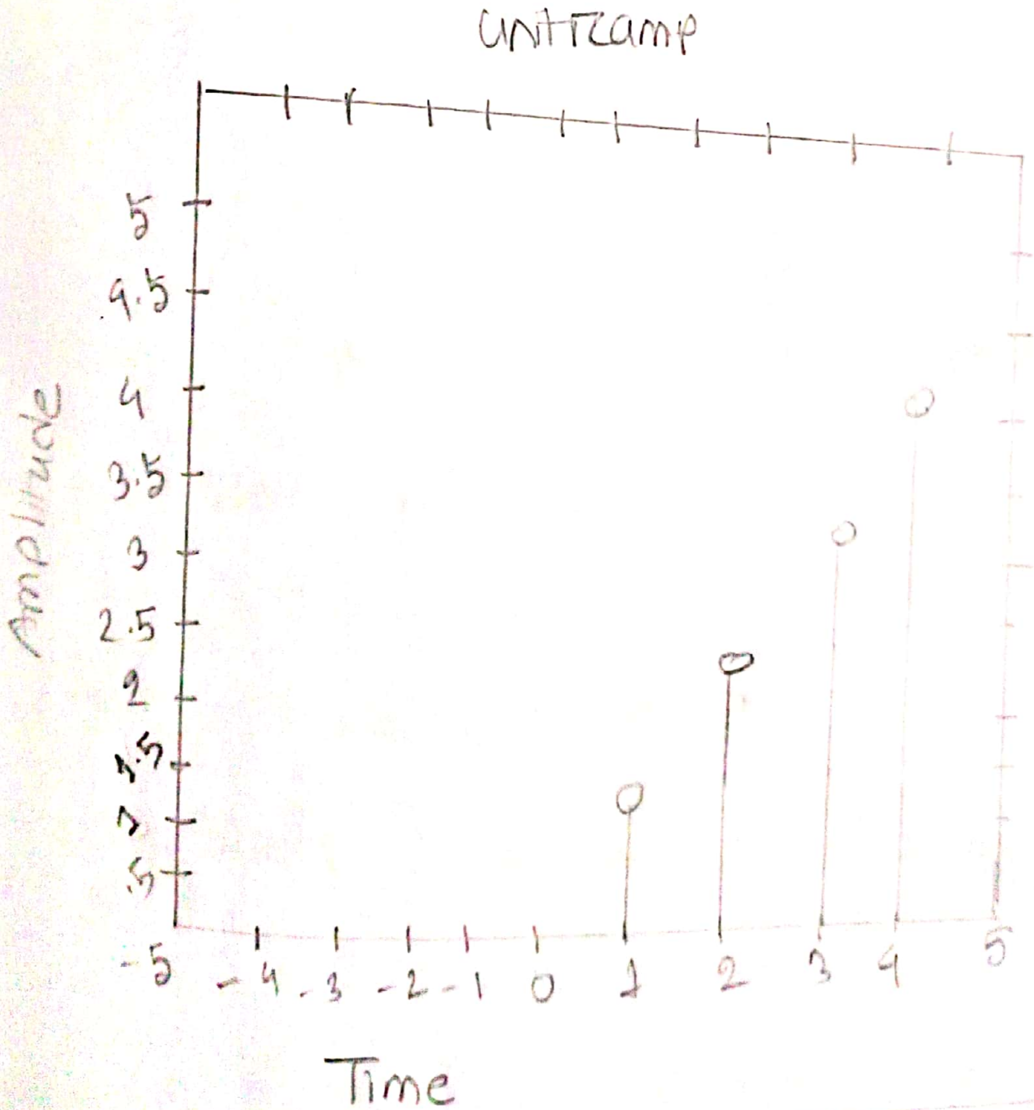
output:



The unit ramp signal using MATLAB:

```
% UnitRamp function
clc;
clear all;
close all;
n= -5:5;
x=n.* [n>=0];
stem (n,x)
xlabel1('Time');
ylabel1 ('Amplitude');
title ('UnitRamp');
```

Output:



Name of The program: Generate the signal $x(n) = k \cdot \exp[cn]$ where $k = 2$, $c = -1/12 + j\pi/6$, $n = 0 to 50$.

Theory:

complex number:

A complex number is a number that can be expressed in the form a+ib, where a and b are real numbers and i is the imaginary unit satisfying the equation $i^2 = -1$

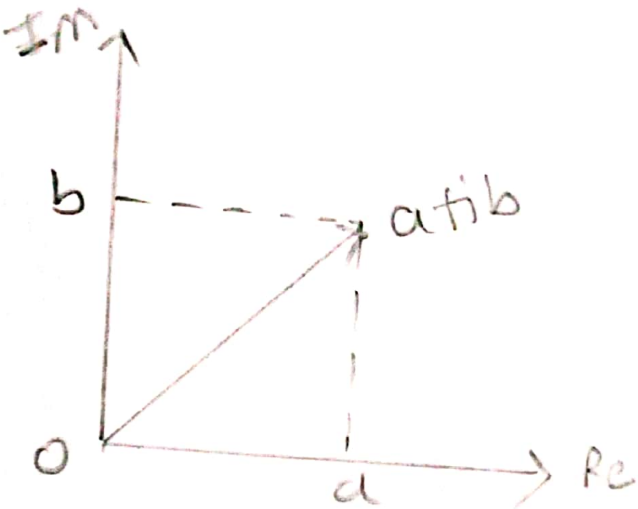


Figure: complex number representation

Real part: Here complex number

$$z = a + ib$$

Here, a is the real part.

imaginary part: Here ib is the imaginary part.

polar form:

Absolute value:

The absolute value of a complex number $z = x+iy$ is $r = |x| = \sqrt{x^2+y^2}$

argument; we know that,

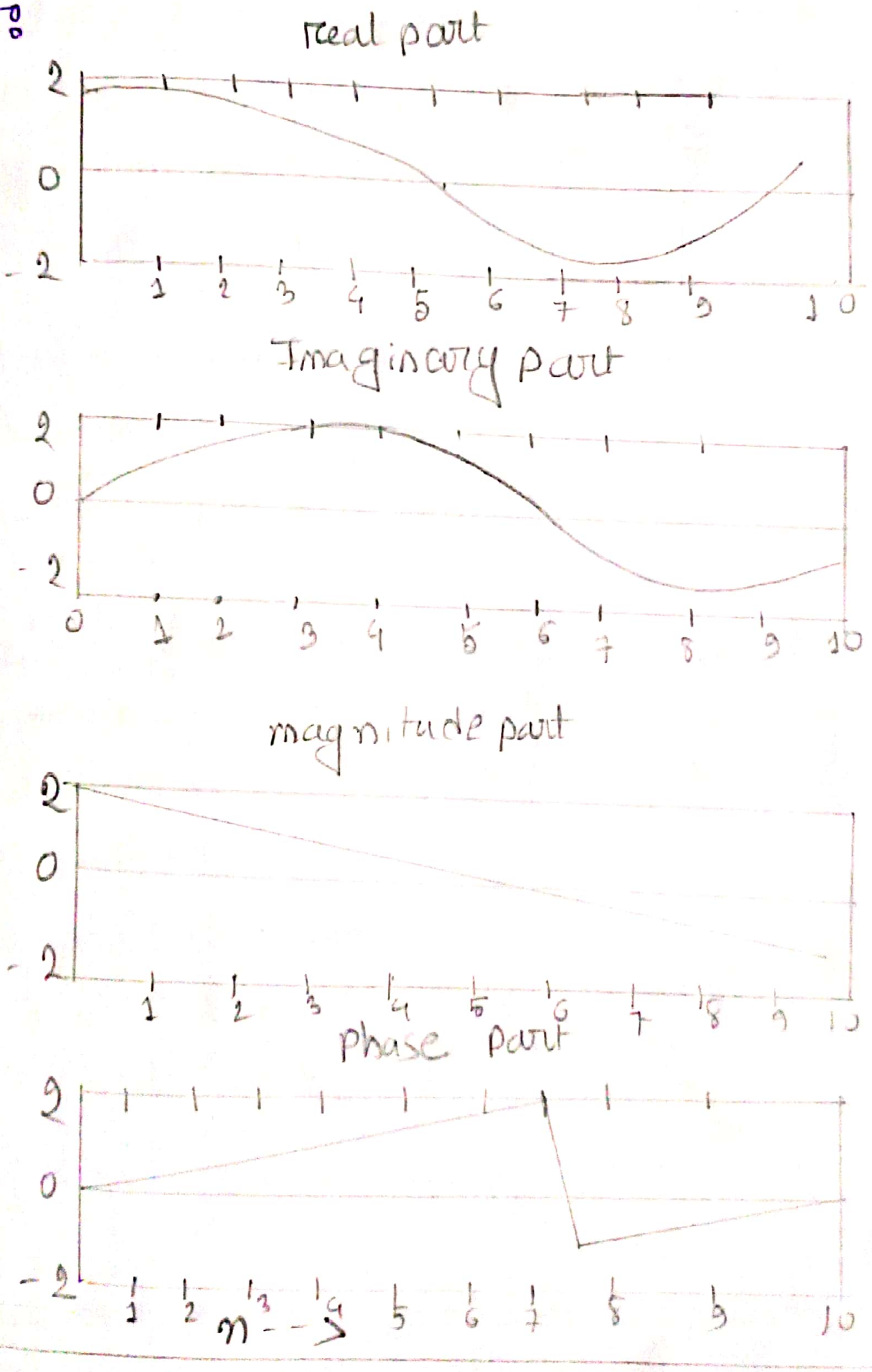
$$\theta = \tan^{-1} y/x$$

Plot sequence using MATLAB:

```
clc;
clear all;
close all;
n=0:50;
K=2
c=(-1/12)+ j*(pi/6);
x =K * exp(c*n);
figure(1):
Subplot(4,1,1);
Plot(n, real(x));
title('real part');
xlabel('n.-->');
Subplot(4,1,3);
Plot(n, abs(x));
title('magnitude part');
```

```
xlabel('n-->');
subplot(4,1,4);
plot(n, angle(x));
title('phase part');
xlabel('n-->');
```

output:



Name of The Problem: Find the convolution of 2 sequence using MATLAB

Theory:

Convolution: Convolution is a mathematical way of combining two signals to form a third signal. For a moment, suppose the input signal to a discrete LTI system with transfer function h(n) is a delta impulse $\delta(n-k)$

Convolution Sum discrete time LTI System:

The equation of the discrete time LTI System is,

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

This equation gives that, the response y(n) of the LTI system as function of the input signal x(n) and the input sample (impulse) response h(n) is called convolution Sum.

Steps of convolution Sum:

The process of computing the convolution between x(k) and h(k) involves the following four Steps.

Folding: Fold $h(k)$ about $k=0$, to obtain $h(-k)$

Shifting: Shift $h(-k)$ by n_0 to the right (left) if n_0 positive (negative) to obtain $h(n_0-k)$

multiplication: multiply $x(k)$ by $h(n_0-k)$ to obtain the product sequence $V_{n_0}(k) = x(k) \cdot h(n_0-k)$.

Summation: Sum all the values of product sequence $V_{n_0}(k)$ to obtain the value of the output at time $n=n_0$.

convolution of 2 sequence using MATLAB:

```
clc;
clear all;
close all;
a=input('Enter the input sequence : ');
b=input('Enter the input sequence: ');
n1 = length(a);
n2 = length(b);
n=0:1:n1-1;
subplot(2,2,1);
title('Input sequence');
```

```
xlabel('n');
ylabel('a(n)');
grid on;
y=0:1:n2-1;
subplot(2,2,2);
stem(n,b);
xlabel('n');
ylabel('b(n)');
% problem here
c=conv(a,b);
n3=0:1:n1+n2-2;
subplot(2,1,2);
stem(n3,c);
title('convolution of two sequence');
xlabel('n');
ylabel('c(n)');
```

Input Sequence