Information and Communication Engineering

Noakhali-3814

Course Title: Signals and Systems LAB

Course Code: ICE-3108

Remarks

Submitted to:	**Submitted by:**
Mohammad Amzad Hossain	Name: Aoishwarya Mojumder
Associate Professor	Roll No: BFH2011010F
Department of ICE	Session: 2019-20,
Noakhali Science and Technology University	Year-3, Term-1

Submission Date:20/08/2023

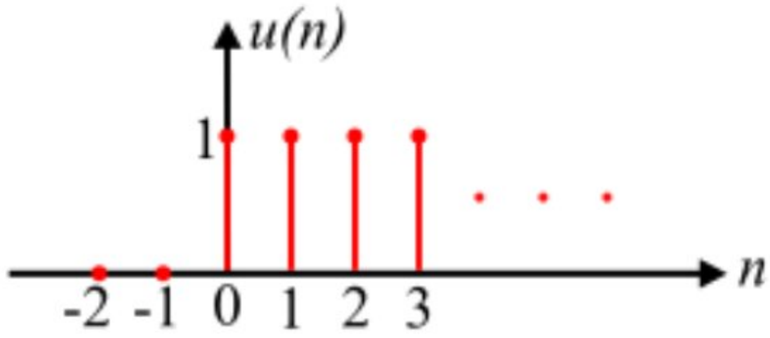
1.

Name of the experiment: Write a MATLAB program to represent the generation of different basic signals(Unit step signal,Unit impulse signal,Exponential signal,Unit ramp signal,Unit parabolic signal,Unit rectangle pulse,Triangular signal,Sinusoidal signal,Symmetric (Even) and Antisymmetric (Odd) signals).

Theory:

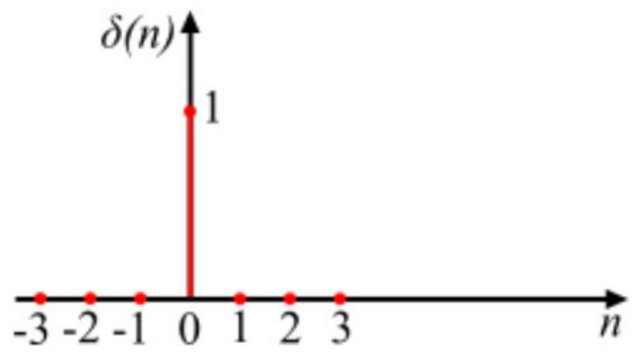
UNIT step signal : The unit step signal which is defined only at discrete instants of time is known as discrete-time unit step signal. It is denoted by u(n). Mathematically, the discrete-time unit step signal or sequence u(n) is defined as follows –

u(n)={1for n≥0, 0for n<0

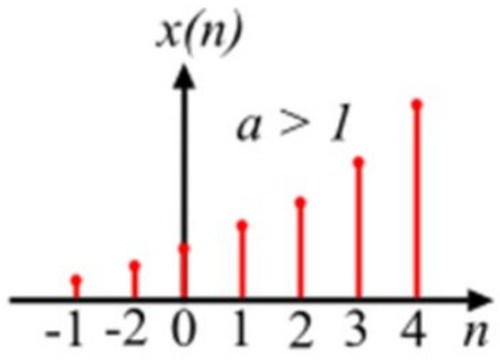


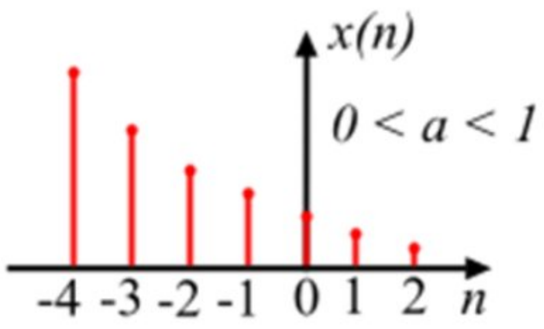
Unit impulse signal: The discrete-time unit impulse signal is denoted by δ(n) and is defined as :

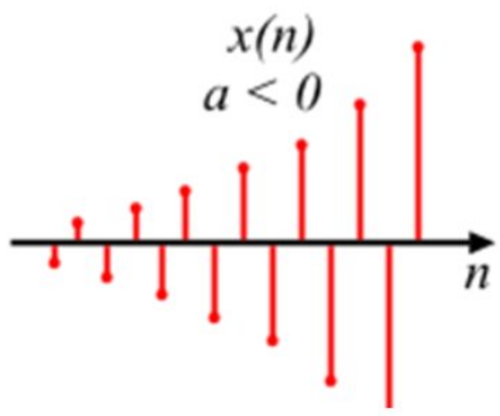
δ(n)={1for n=0 ,0for otherwise



Exponential signal: A discrete-time real exponential sequence is defined as :$x(n) = a^n$ for all n
Depending upon the value of a the discrete time real exponential signal may be of following type −
When a > 1, the exponential sequence x(n) grows exponentially.
When 0 < a < 1, the exponential signal x(n) decays exponentially.
When a < 0, the exponential sequence x(n) takes alternating signs.

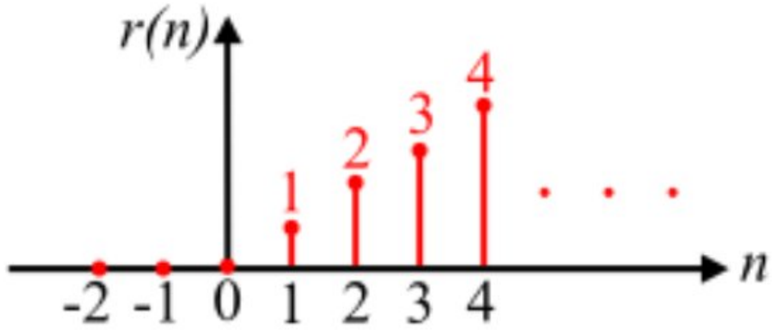






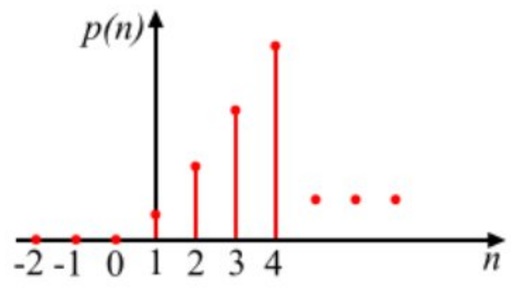
Unit ramp signal: The discrete time unit ramp signal is that function which starts from n = 0 and increases linearly. It is denoted by r(n). It is signal whose amplitude varies linearly with time n. mathematically, the discrete time unit ramp sequence is defined as −

r(n)={n for n≥0 ,0for n<0

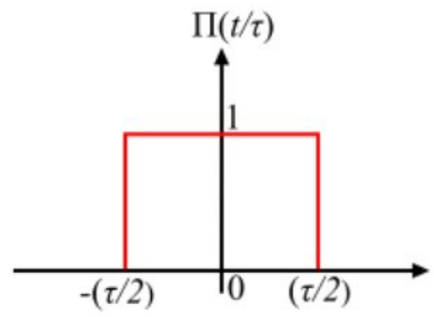


Unit parabolic signal: The discrete-time unit parabolic sequence is a unit parabolic signal which is defined only at discrete instants of positive time n. It is denoted by p(n). Mathematically, p(n) is given as,

p(n) ={ $\frac{n^2}{2}$for n≥0 ,0 for n<0 .Also, p(n)= $\frac{n^2}{2}$u(n)

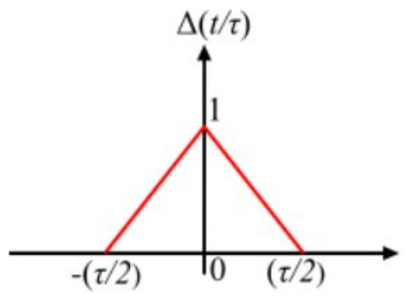


Unit rectangle pulse:A signal that produces a rectangular shaped pulse with a width of τ (where $\tau = 1$ for the unit rectangular function) centred at $t = 0$ is known as rectangular signal. Mathematically, the unit rectangular signal is defined as,∏(t/τ)= {1 for |t|≤(τ/2) ,0 otherwise .



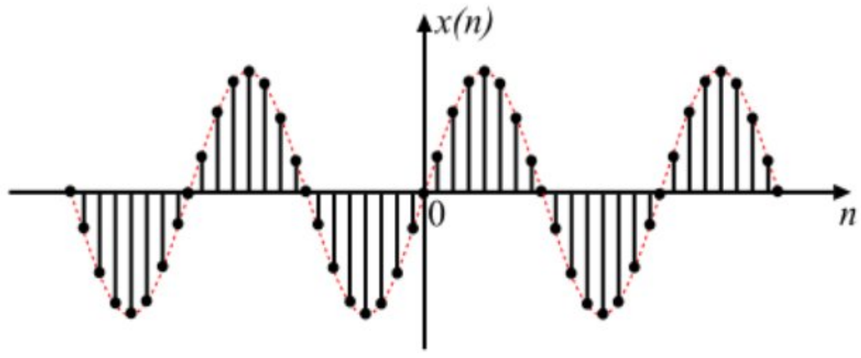
Triangular signal:A function whose graph takes the shape of a triangle is known as triangular signal. The triangular signal is also known as hat function or tent function. Mathematically, the unit triangular pulse signal Δ(t/τ) is defined as,

Δ(t/τ)={1−(2|t|/τ)for|t|<(τ/2) 0for|t|>(τ/2)



Sinusoidal sequence/signal:A sinusoidal sequence/signal which is defined only at discrete instants of time is called discrete-time sinusoidal signal. It is given as follows:

$$x(n) = A \sin(\omega n + \varphi) = A \sin(2\pi f n + \varphi)$$



Symmetric (Even) and Antisymmetric (Odd) signals:

A signal x(t) or x[n] is referred to as an even signal if

x(-t) = x(t), x[-n] = x[n]

A signal x(t) or x[n] is referred to as an odd signal if

x(-t) = -x(t), x[-n] = -x[n]

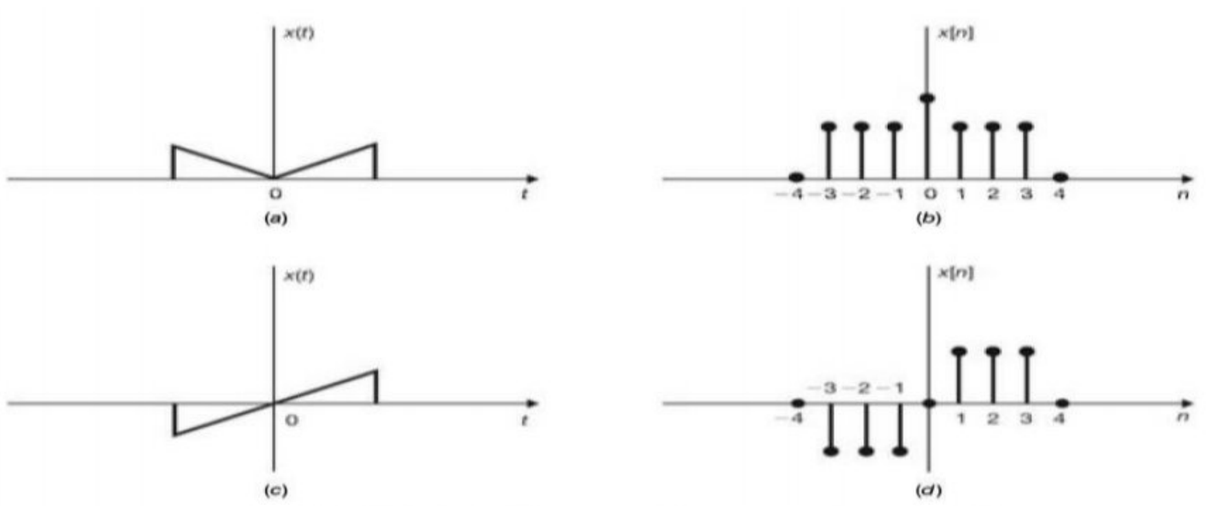


Fig. 1-2 Examples of even signals (*a* and *b*) and odd signals (*c* and *d*).

Source code:

```
clc;
clear all;
close all;
% Unit step signal:
N=10;t=1:10;
x=ones(1,N);
subplot(5,2,1);
stem(t,x,'b');
xlabel('time');ylabel('amplitude');
title('unit step');

%Unit impulse signal
t=-10:1:10;x=(t==0);
subplot(5,2,2);stem(t,x,'b');
xlabel('time');ylabel('amplitude');
title('unit impulse');

%Exponential signal
n=5:15;x_n=(0.9).^n;
subplot(5,2,3);stem(n,x_n,'b');
xlabel('Time sample');ylabel('Amplitude');
title('unit exp');

%Unit ramp signal
t=-20:20;ramp_n=(n>=0).*n;
subplot(5,2,4);stem(n, ramp_n);
xlabel('Time Sample');ylabel('Amplitude')
title('unit ramp');

%Unit parabolic signal
t=-6:0.01:6; parabola = 0.5 *(t.^2);
subplot(5,2,5);plot(t, parabola);
xlabel('Time Sample');ylabel('Amplitude')
title('unit para');

%Unit rectangle pulse
n=-5:0.01:5;width=2;
x_n=rectpuls(n,width);subplot(5,2,6);plot(n,x_n);
xlabel('Time sample');ylabel('Amplitude');
title('unit rec');

%Triangular signal
x = -5:0.01:5;width=2;
y=tripuls(x,width);subplot(5,2,7);plot(x,y);
title(['Triangular Pulse']);
xlabel('Time');ylabel('Amplitude');

%Sinusoidal signal
t=-0.05:0.001:0.05;f=25;a=2;
y =a* sin (2*pi*f*t);subplot(5,2,8);plot(t, y);
xlabel('Time Sample');ylabel('Amplitude')
title('Sinusoidal signal');

%Symmetric (Even) and Antisymmetric (Odd) signals
n1=0:6;x1=[1 1 1 1 1 1 0];n2=-fliplr(n1);
n=min(min(n1),min(n2)):max(max(n1),max(n2));
y1=zeros(1,length(n));y1((n>=min(n1))&(n<=max(n1)))=x1();x=y1;
xe=0.5*(x+fliplr(x));xo=0.5*(x-fliplr(x));
subplot(5,2,9),stem(n,xe);
title('Even Signal');
subplot(5,2,10),stem(n,xo);
title('Odd Signal');
```

Output:

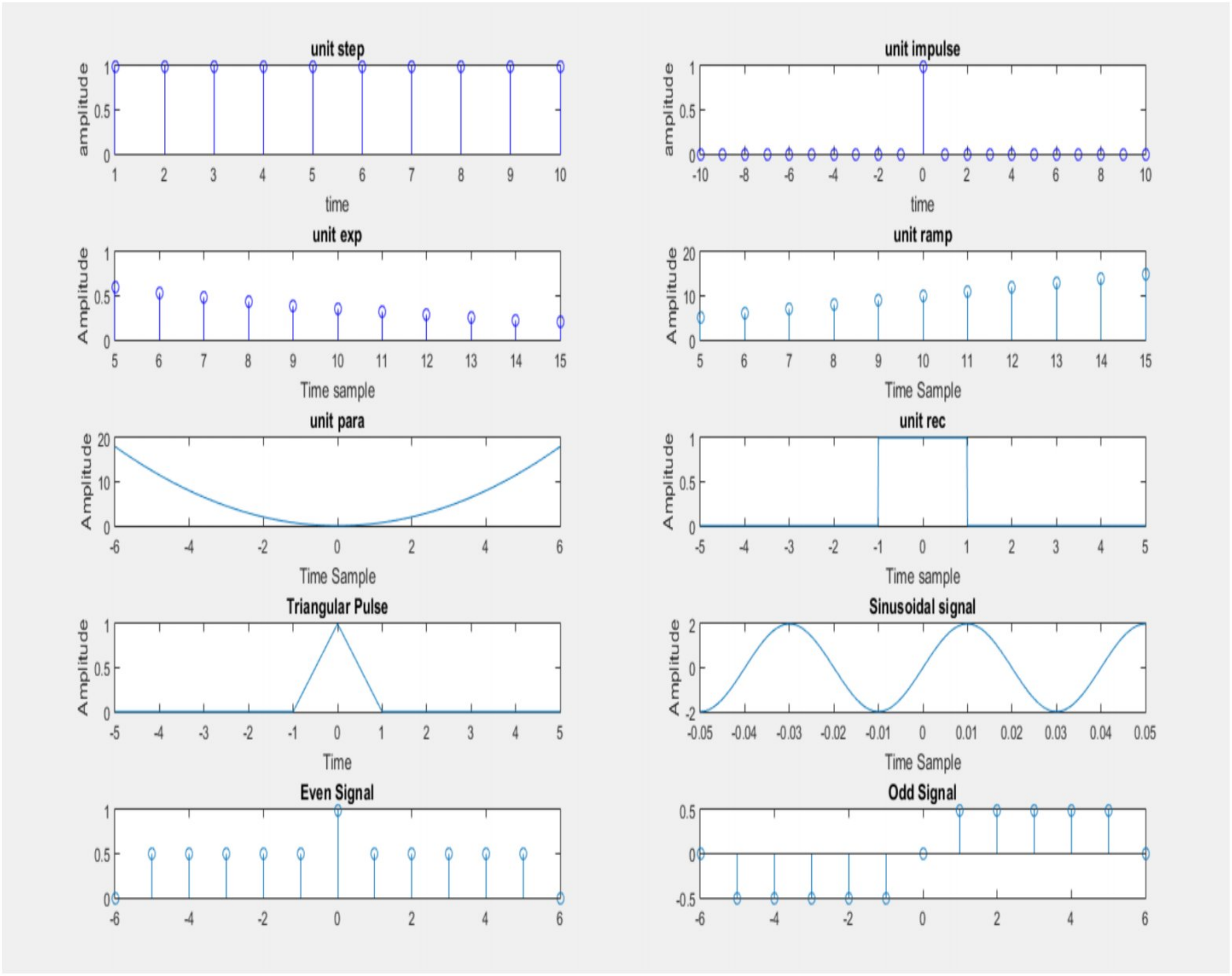
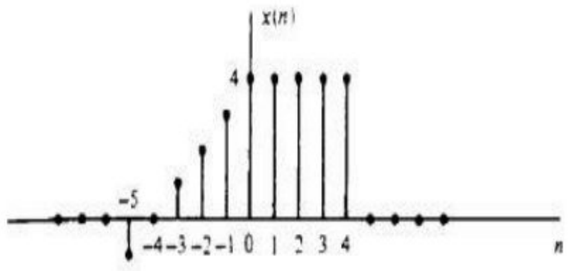


Fig: Different Basic Discreate Time Signals

2.

Name of the experiment: Write a MATLAB program to show a graphical representation of the signals x[n-3] and x[n+3].



Theory:

The time shifting operation of a discrete time signal x(n) is represented as ,x$(n) = x(n - n_0)$.

This equation shows that the signal y(n) can be obtained by time-shifting the signal x(n) by n_0 units. If the value of n_0 is positive, then the shift of the signal is to the right and the resulting signal is the delayed version. Whereas, if the value of n_0 is negative, then the shift of the signal is to the left and it is the time advanced version of the signal.

Source code:

```
n1=3;n2=3;n=-5:4;
x=[-1 0 1 2 3 4 4 4 4 4];
subplot(3,1,1);
stem(n,x);
title('Signal x(n)');
m=n+n1;
y=x;
subplot(3,1,2);
stem(m,y);
title('signal x(n-3)');
t=n-n2;
z=x;
subplot(3,1,3);
stem(t,z);
title('signal x(n+3)');
```

Output:

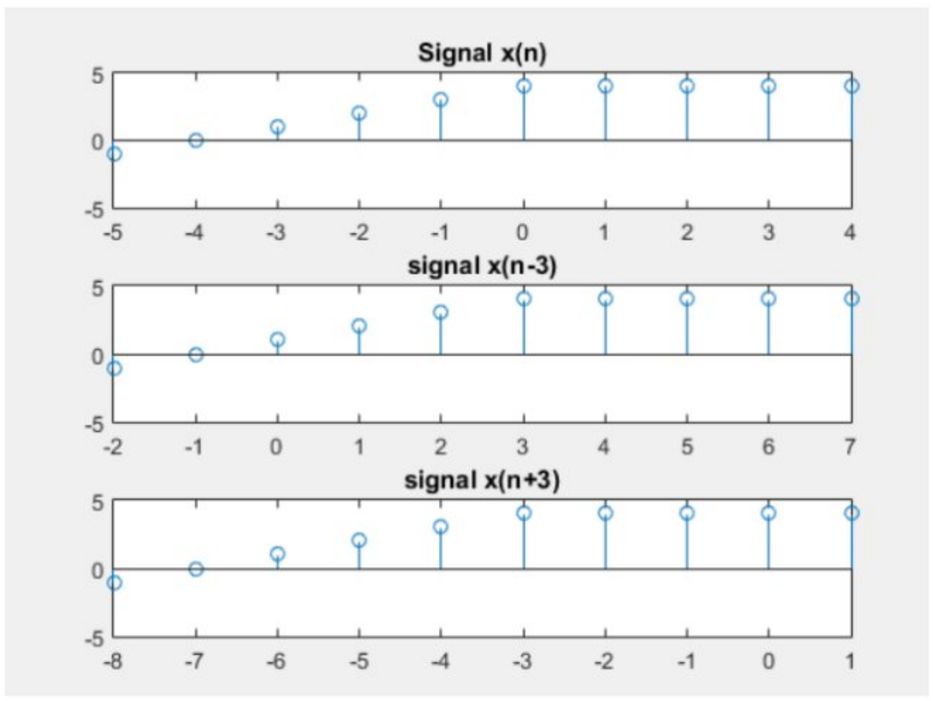


Figure:Time Shifting Signal

3.

Name of the experiment: Write a MATLAB program to verify the time invariant characteristics of discrete time system.

Theory:

In the discrete-time case, the time-invariance property is known as shift invariance.Let,x(n) is the input and x(n-k) is the delayed input to the given discrete time system. Then, the output of the system corresponding to the x(n) is given by $x(n) \rightarrow y(n) = T[x(n)]$

And the output for the delayed input if $(n-k) \rightarrow (n,k) = [(n-k)] = (n)|_{x(n)=x(n-k)}$

Also, the output of the system delayed by k units is $(n-k) = (n)|_{n=(n-k)}$,if $y(n,k) = y(n-k)$

That is, when the delayed output of the system is equal to the output due to delayed input for all possible values of k, then the given system is a time invariant system.

Source code:

```
n = 0:40; D = 5;a = 3.0;b = -2;
x = a*cos(2*pi*0.1*n) + b*cos(2*pi*0.4*n);
xd = [zeros(1,D) x];
num = [2.2403 2.4908 2.2403];den = [1 -0.4 0.75];
y = filter(num,den,x);yd = filter(num,den,xd);
d = y - yd(1+D:41+D);
subplot(3,1,1)stem(n,y);ylabel('Amplitude');
title('Output y[n]'); grid;
subplot(3,1,2)stem(n,yd(1:41));
ylabel('Amplitude');
title(['Output due to Delayed Input x[n-',
num2str(D),']']); grid;
subplot(3,1,3);stem(n,d);
xlabel('Time index n'); ylabel('Amplitude');
title('Difference Signal'); grid;
```

Output:

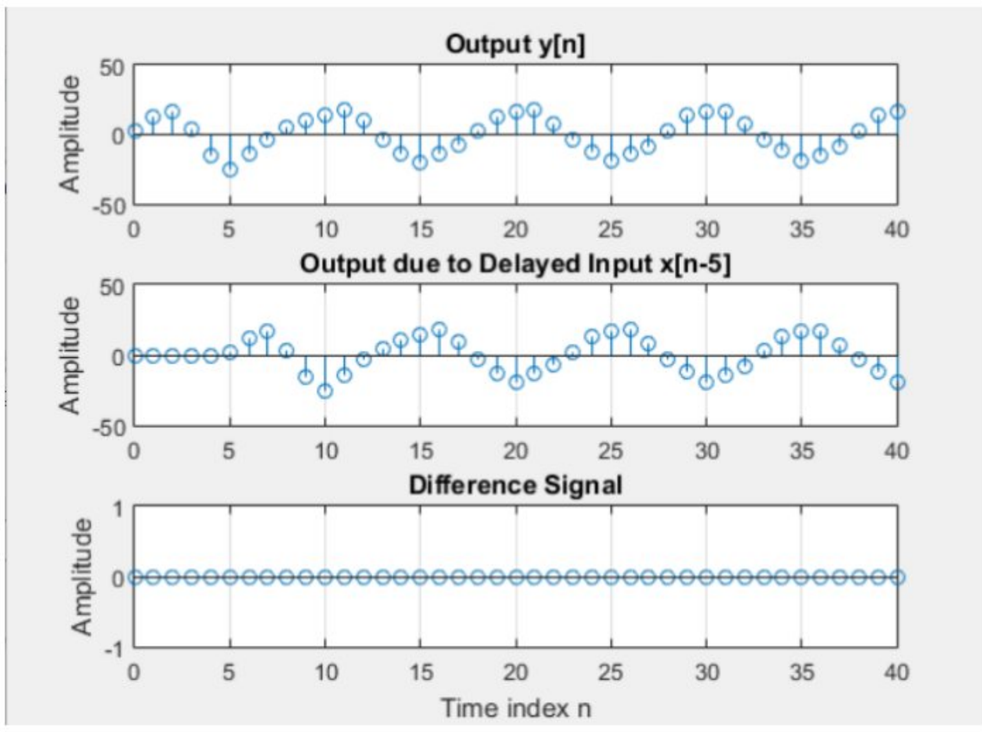


Figure:Time invariant Signal

4.

Name of the experiment: Write a MATLAB program to verify the causality of discrete time system.

Theory:

A causal system is the one in which the output y(n) at time n depends only on the current input x(n) at time n, and its past input sample values such as x(x−1), x(x−2),…. Otherwise, if a system output depends on future input values such as x(x+1), x(x+2), …, the system is noncausal.

$y(n) = x(n-1) + x(n), y(n) = nx(n)$

Source code:

```
h = [0.1, 0.2, 0.3, 0.2, 0.1];
x = [1, 2, 3, 4, 5, 4, 3, 2, 1];
y = conv(x, h);
subplot(3,1,1);
stem(0:length(y)-1, y, 'r',
'LineWidth', 2); grid on;
title('Causal Output
Signal');xlabel('Sample
Index');ylabel('Amplitude');
subplot(3,1,2);
stem(0:length(x)-1, x, 'b',
'LineWidth', 2); grid on;
title('Input Signal');
xlabel('Sample
Index');ylabel('Amplitude');
subplot(3,1,3);
stem(0:length(h)-1, h, 'g',
'LineWidth', 2); grid on;
title('Impulse Response');
xlabel('Sample
Index');ylabel('Amplitude');
title('Causal System Response');
```

Output:

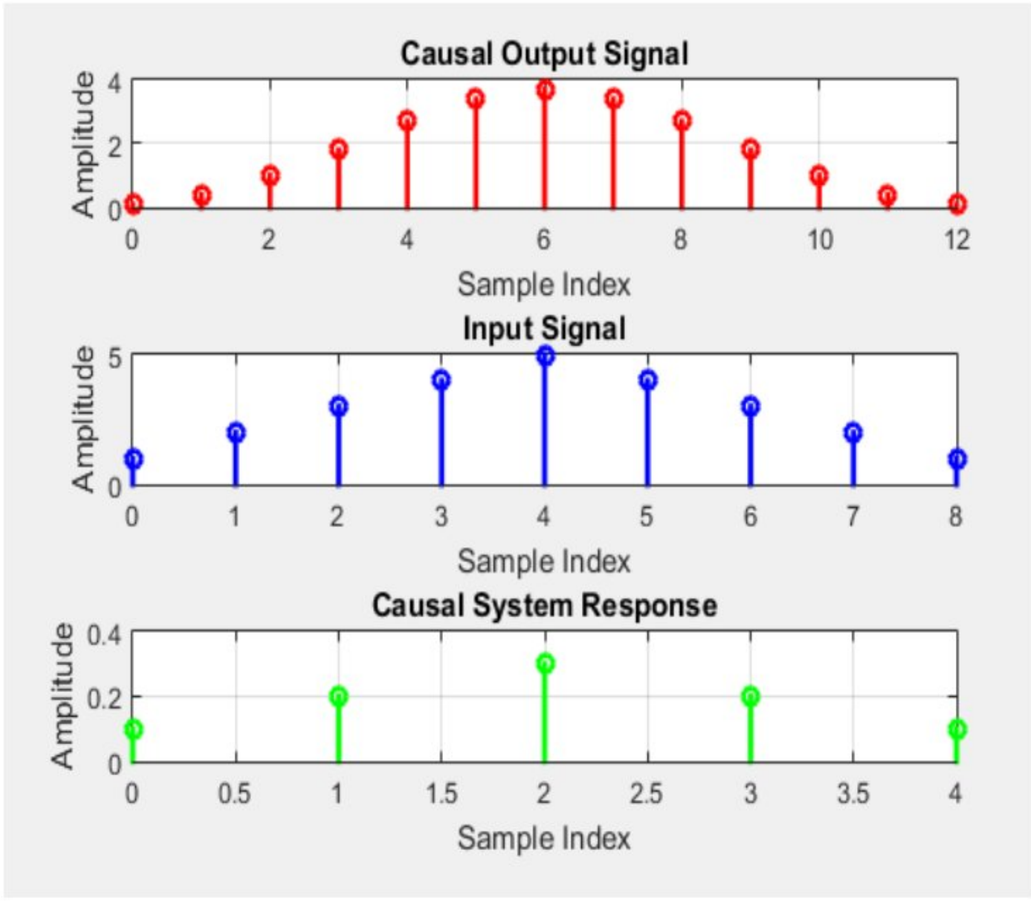


Figure:CausalSignal

5.

Name of the experiment: Write a MATLAB program to verify the linearity of discrete time system.

Theory:

The term linear defines a special class of systems where the output is the superposition, or sum, of the individual outputs had the individual inputs been applied separately to the system.Let,Input $x_1(n)$ to a system results in an output $y_1(n)$. We symbolize this situation with the following expression:

$$x_1(n) \xrightarrow{\text{results in}} y_1(n). \quad x_2(n) \xrightarrow{\text{results in}} y_2(n). \quad x_1(n)+x_2(n) \xrightarrow{\text{results in}} y_1(n)+y_2(n).$$

if the inputs are scaled by constant factors c_1 and c_2, then the output sequence parts are also scaled by those factors as $c_1x_1(n)+c_2x_2(n) \xrightarrow{\text{results in}} c_1y_1(n)+c_2y_2(n).$

this proportionality attribute of linear systems is sometimes called the homogeneity property.It is the concept of system linearity.

Source code:

```
clc; close all; clear all;
n=0:40; a=2; b=1;
x1=cos(2*pi*0.1*n);
x2=cos(2*pi*0.4*n);x=a*x1+b*x2;
y=n.*x;y1=n.*x1;y2=n.*x2;
yt=a*y1+b*y2;d=y-yt;
d=round(d)
if d
disp('Given system is not satisfy
linearity property');
else
disp('Given system is satisfy linearity
property');end
subplot(3,1,1), stem(n,y);grid;
subplot(3,1,2), stem(n,yt);grid;
subplot(3,1,3), stem(n,d);grid;
```

Output:

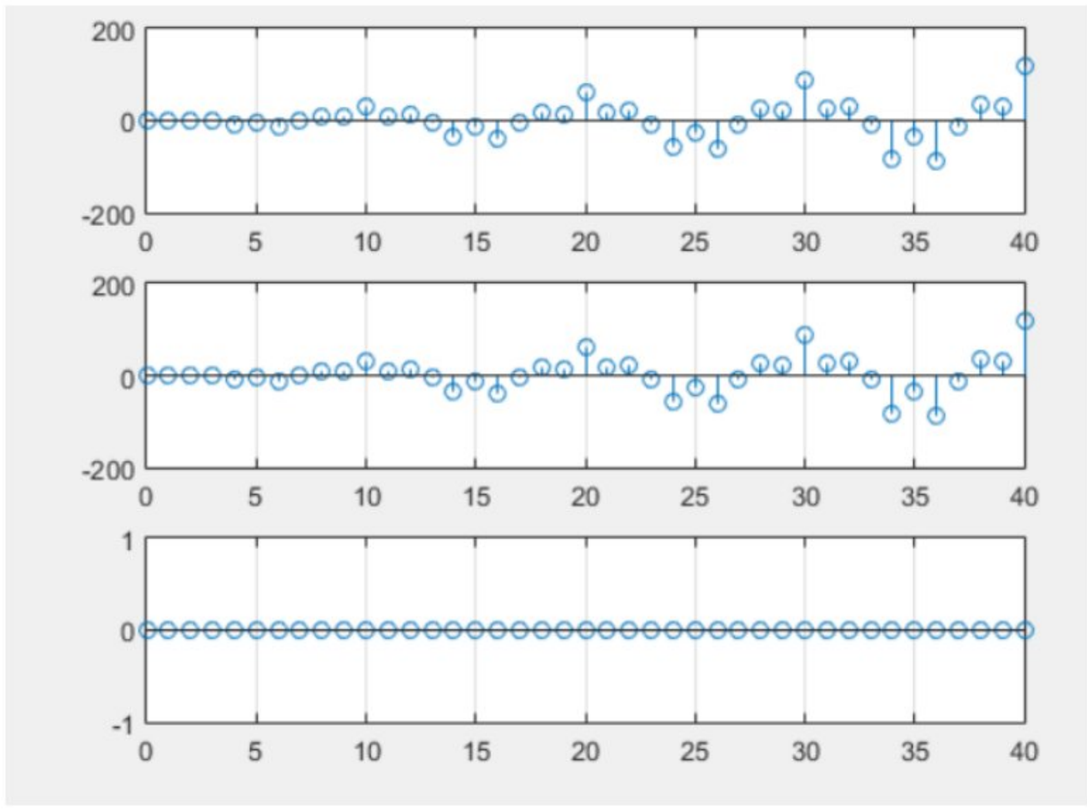


Figure:Linearity Check

6.

Name of the experiment: Write a MATLAB program to verify the stability of discrete time system.

Theory:

BIBO stability stands for bounded input, bounded output stability. BIBO stability is the system property that any bounded input yields a bounded output. This is to say that as long as we input a signal with absolute value less than some constant, we are guaranteed to have an output with absolute value less than some other constant. Bounded input bounded output stability, also known as BIBO stability, is an important and generally desirable system characteristic. A system is BIBO stable if every bounded input signal results in a bounded output signal, where boundedness is the property that the absolute value of a signal does not exceed some finite constant. In terms of time domain features, a discrete time system is BIBO stable if and only if its impulse response is absolutely summable.

Source code:

```
N=40;
num=[0.9 -0.45 0.35 0.002];
den = [1.0 0.71 -0.46 -0.62];
n=0:N-1;

x=(n==0);

y=filter(num,den,x);

stem(n,y);
title('Verification of Stability');
```

Output:

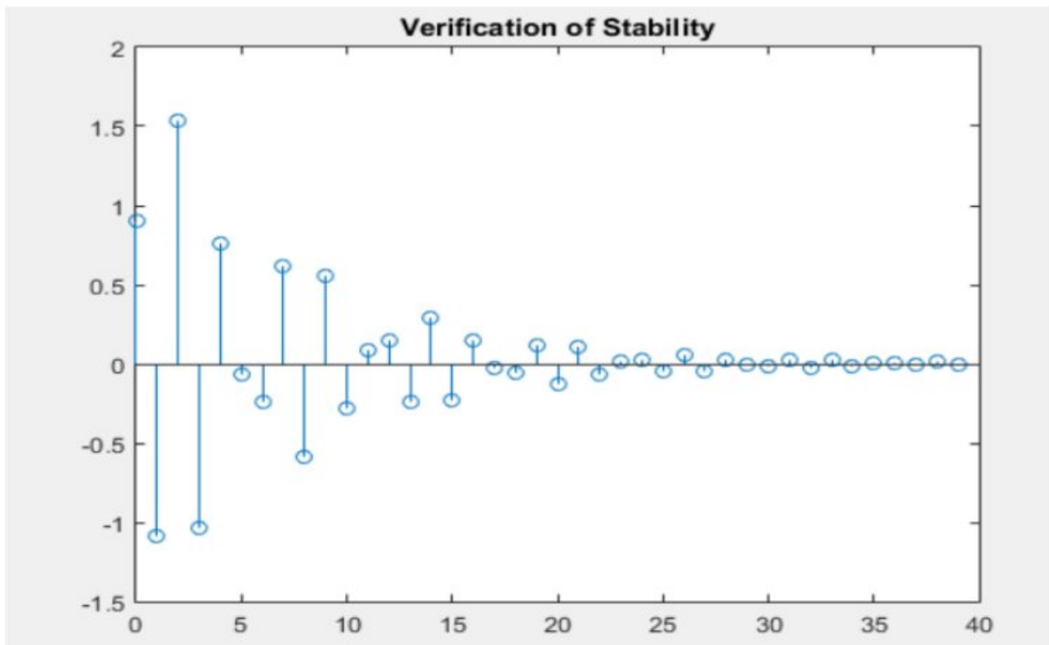


Figure: Stability Check

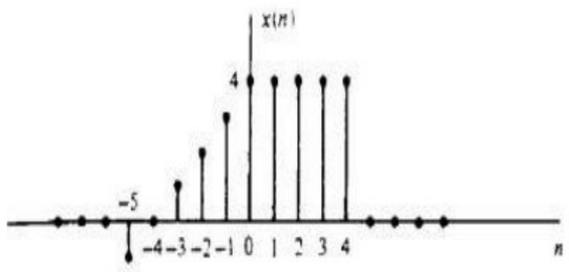
7.

Name of the experiment: Write a MATLAB program to verify the shifting operation of discrete time system.

Theory:

The time shifting operation of a discrete time signal x(n) is represented as ,

$x(n) = (n - n_0)$.



This equation shows that the signal y(n) can be obtained by time-shifting the signal x(n) by n_0 units. If the value of n_0 is positive, then the shift of the signal is to the right and the resulting signal is the delayed version. Whereas, if the value of n_0 is negative, then the shift of the signal is to the left and it is the time advanced version of the signal.

Source code:

```
n1=3;n2=3;n=-5:4;
x=[-1 0 1 2 3 4 4 4 4 4];
subplot(3,1,1);stem(n,x);
title('Signal x(n)');
m=n+n1;y=x;
subplot(3,1,2);stem(m,y);
title('signal x(n-3)');
t=n-n2;z=x;
subplot(3,1,3);
stem(t,z);
title('signal x(n+3)');
```

Output:

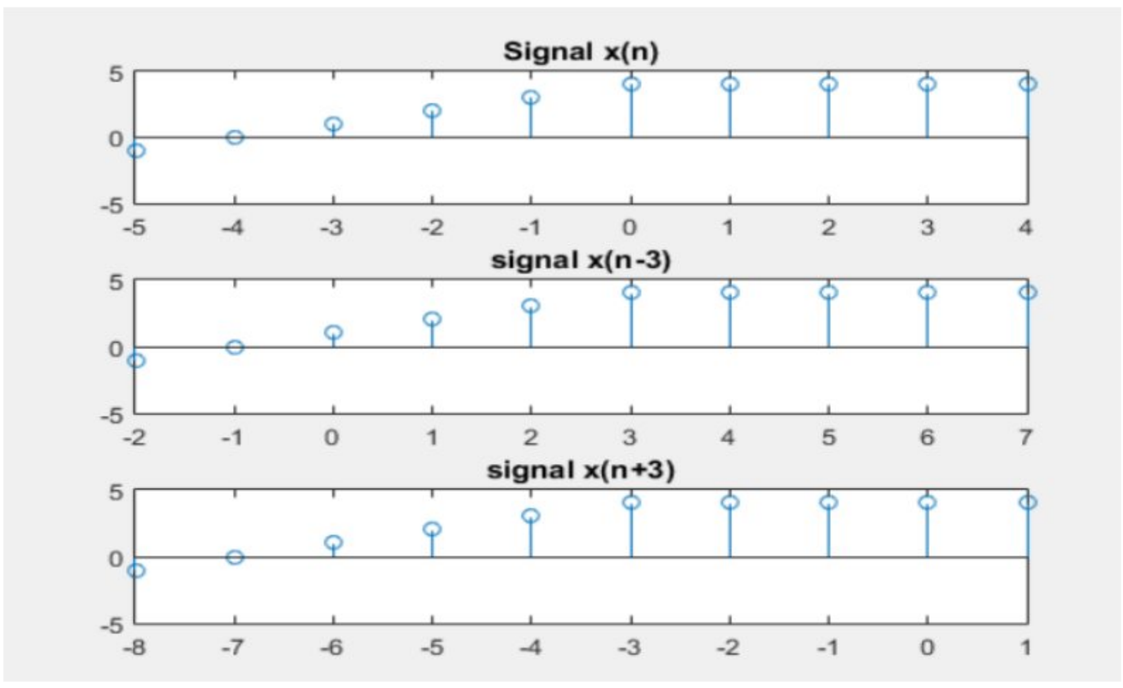


Figure:Shifting Signal

8.

Name of the experiment: Write a MATLAB and Python program to generate Fourier series of a Square Wave.

Theory: A Fourier series is an expansion of a periodic function f(x) in terms of an infinite sum of sines and cosines. Fourier Series makes use of the orthogonality relationships of the sine and cosine functions.
The formula for the fourier series of the function f(x) in the interval [-L, L], i.e. $-L \le x \le L$ is given by:

$$f(x) = A_0 + \sum_{n=1}^{\infty} A_n \cdot \cos\left(\frac{n\pi x}{L}\right) + \sum_{n=1}^{\infty} B_n \cdot \sin\left(\frac{n\pi x}{L}\right)$$

Here,

$$A_0 = \frac{1}{2L} \cdot \int_{-L}^{L} f(x)\,dx$$

$$A_n = \frac{1}{L} \cdot \int_{-L}^{L} f(x)\cos\left(\frac{n\pi x}{L}\right)dx, \quad n > 0$$

$$B_n = \frac{1}{L} \cdot \int_{-L}^{L} f(x)\sin\left(\frac{n\pi x}{L}\right)dx, \quad n > 0$$

Source code:

```
f=500;C=4/pi;
dt=5.0e-05;tpts=(4.0e-3/5.0e-5)+1;
for n=1:12;
for m=1:tpts;
s1(n,m)=(4/pi)*(1/(2*n-1))*sin((2*n-
1)*2*pi*f*dt*(m-1));
end end
for m=1:tpts;
a1=s1(:,m);
a2(m)=sum(a1);
end
f1=a2;
t=0.0:5.0e-5:4.0e-3;
plot(t,f1);xlabel('Time,s');ylabel('Amplitude,V');
title('FourierSeriesExpansion');
```

Output:

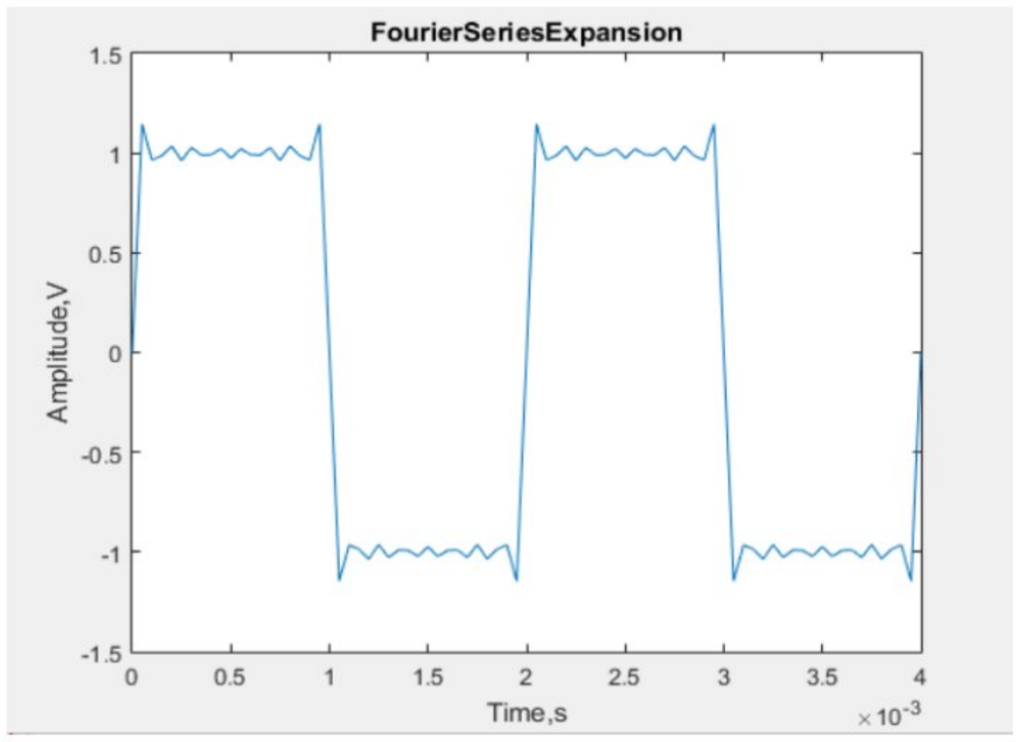


Figure:FourierSeries

Python:

```
import numpy as np
import matplotlib.pyplot as plt
amplitude = 1.0
frequency = 1.0
period = 1 / frequency
num_terms = 20
t = np.linspace(0, 3 * period, 1000)
fourier_series = np.zeros_like(t)
for n in range(1, num_terms + 1):
term = (4 / (n * np.pi)) * np.sin(2 * np.pi * n * frequency * t)
    fourier_series += term
fourier_series *= amplitude
plt.figure(figsize=(10, 6))

plt.plot(t, amplitude * np.sign(np.sin(2 * np.pi * frequency * t)), label="Original Square Wave")
plt.plot(t, fourier_series, label=f"Fourier Series ({num_terms} terms)")
plt.title("Fourier Series of a Square Wave")
plt.xlabel("Time")
plt.ylabel("Amplitude")
plt.legend()
plt.grid()
plt.show()
```

9.

Name of the experiment: Write a MATLAB and Python program to generate Discrete-Time Fourier Transform (DTFT) of given signal.

$$X(e^{j\omega}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

Theory:

A discrete-time signal can be represented in the frequency domain using discrete-time Fourier transform. Therefore, the Fourier transform of a discretetime sequence is called the discrete-time Fourier transform (DTFT).Mathematically, if x(n) is a discrete-time sequence, then its discrete-time Fourier transform is defined as −

$F[x(n)]=X(\omega)=\sum_{n=-\infty}^{\infty} x(n)e^{jwn}$

The discrete-time Fourier transform X(ω) of a discrete-time sequence x(n) represents the frequency content of the sequence x(n). Therefore, by taking the Fourier transform of the discrete-time sequence, the sequence is decomposed into its frequency components. For this reason, the DTFT X(ω) is also called the signal spectrum.

Source code:

```
x=ones(1,3);

w=-2*pi:0.01:4*pi;
h=freqz(x,1,w);
subplot(2,1,1);
plot(w/pi,abs(h));
grid;
title('Magnitude Response');

xlabel('\omega/\pi');

subplot(2,1,2);
plot(w/pi,angle(h));
grid;

title('Phase Response');

xlabel('\omega/\pi');
```

Output:

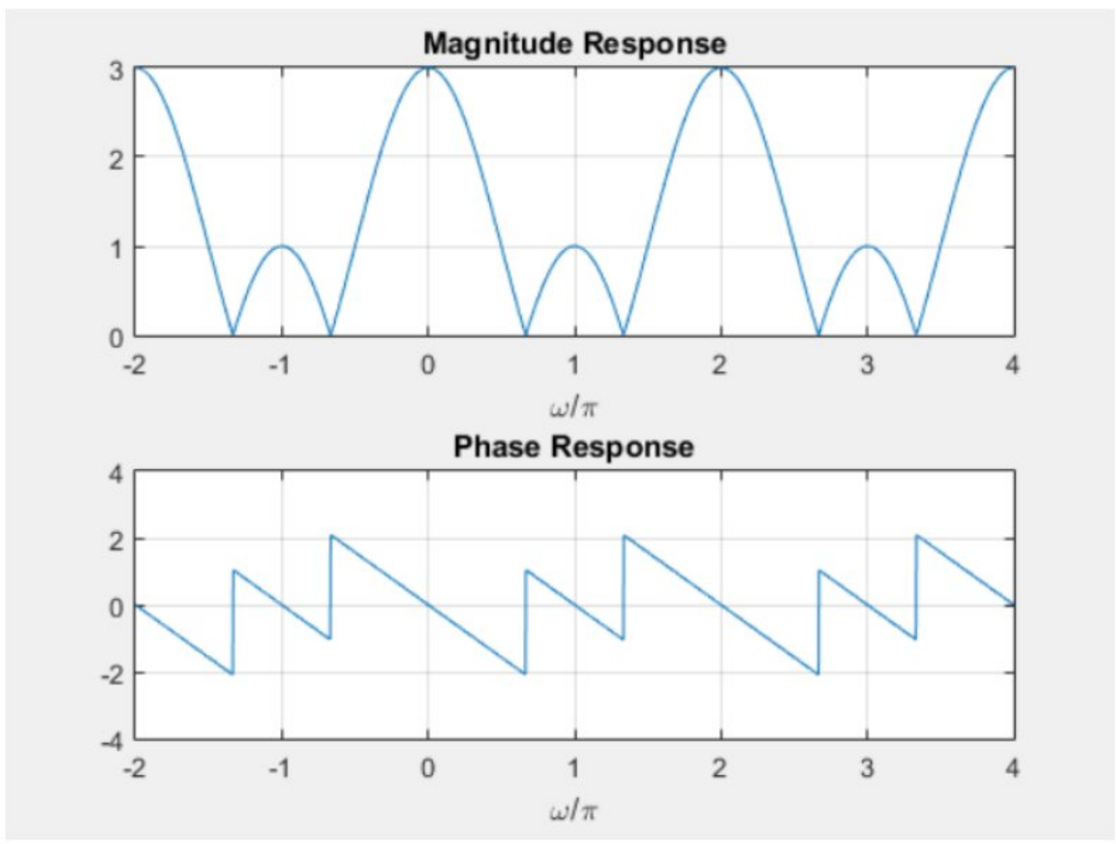


Figure:DiscreatetimeFourierTransform

Python:

```
import numpy as np
import matplotlib.pyplot as plt
def dtft(signal, omega):
N = len(signal)
result = np.zeros_like(omega, dtype=np.complex128)
for k in range(N):
        result += signal[k] * np.exp(-1j * omega * k)
```

```
    return result
N = 64
n = np.arange(N)
frequency = 0.2
signal = np.exp(1j * 2 * np.pi * frequency * n)
num_omega = 1000
omega = np.linspace(-np.pi, np.pi, num_omega)
dtft_result = dtft(signal, omega)
plt.figure(figsize=(10, 6))
plt.subplot(2, 1, 1)
plt.plot(omega, np.abs(dtft_result))
plt.title('Magnitude of DTFT')
plt.xlabel('Frequency (radians)')
plt.ylabel('|X(omega)|')
plt.grid()
plt.subplot(2, 1, 2)
plt.plot(omega, np.angle(dtft_result))
plt.title('Phase of DTFT')
plt.xlabel('Frequency (radians)')
plt.ylabel('Phase (radians)')
plt.grid()
plt.tight_layout()
plt.show()
```

10.

Name of the experiment: Write a MATLAB and Python program for determining the response of the LIT system for input signal is x[n] = [1, 2, 3, 6], where impulse response of LIT system is h[n] = [1, 2, 1, -2].

Theory: Convolution is a mathematical tool to combining two signals to form a third signal. Therefore, in signals and systems, the convolution is very important because it relates the input signal and the impulse response of the system to produce the output signal from the system. In other words, the convolution is used to express the input and output relationship of an LTI system. The mathematical shorthand notation for the convolution operation is to use the * symbol as follows:

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

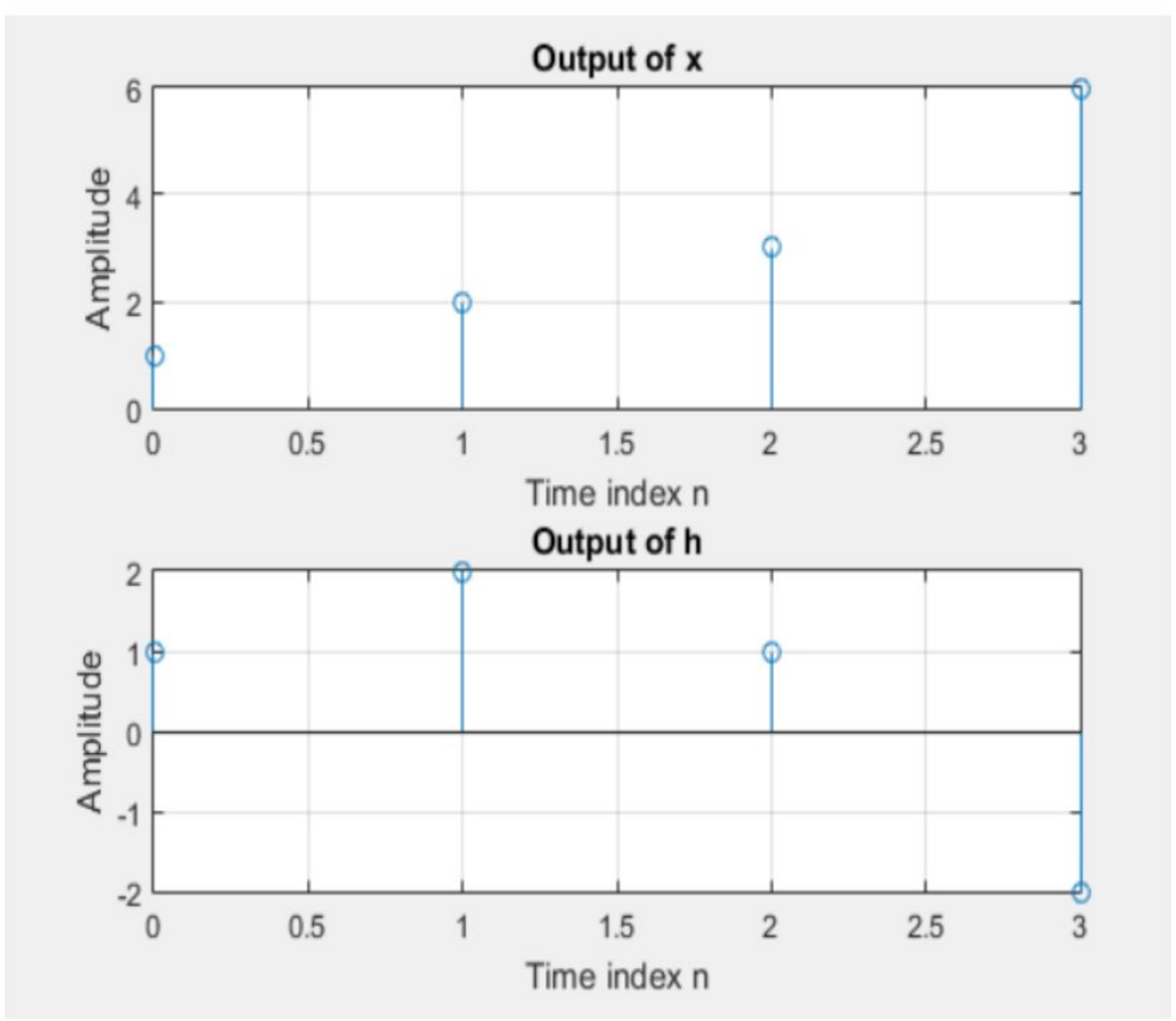
Source code:

```
h = [1, 2, 1, -2];nh=[0 1 2 3];% impulse response
x = [1, 2, 3, 6]; nx=[0 1 2 3];% input sequence
y = conv(h,x);
ny = 0:length(y)-1;
subplot(3,1,1);
stem(nx,x);
xlabel('Time index n');
ylabel('Amplitude');
title('Output of x');grid;
subplot(3,1,2);
stem(nh,h);
xlabel('Time index n');ylabel('Amplitude');
title('Output of h');grid;
subplot(3,1,3);stem(ny,y);
xlabel('Time index n');
ylabel('Amplitude');
title('Output of y');grid;
```

Python:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.array([1, 2, 3, 6])
h = np.array([1, 2, 1, -2])
result_length = len(x) + len(h) - 1
y = np.convolve(x, h)
n_result = np.arange(result_length)
plt.figure(figsize=(10, 6))
plt.stem(np.arange(len(x)), x, basefmt='b',
linefmt='b', markerfmt='bo', label='Input Signal')
plt.stem(np.arange(len(h)), h, basefmt='g',
linefmt='g', markerfmt='go', label='Impulse
Response')
plt.stem(n_result, y, basefmt='r', linefmt='r',
markerfmt='ro', label='Response')
plt.title('Response of LIT System')
plt.xlabel('Time Index (n)')
plt.ylabel('Amplitude')
plt.legend()
plt.grid()
plt.show()
```

Output:



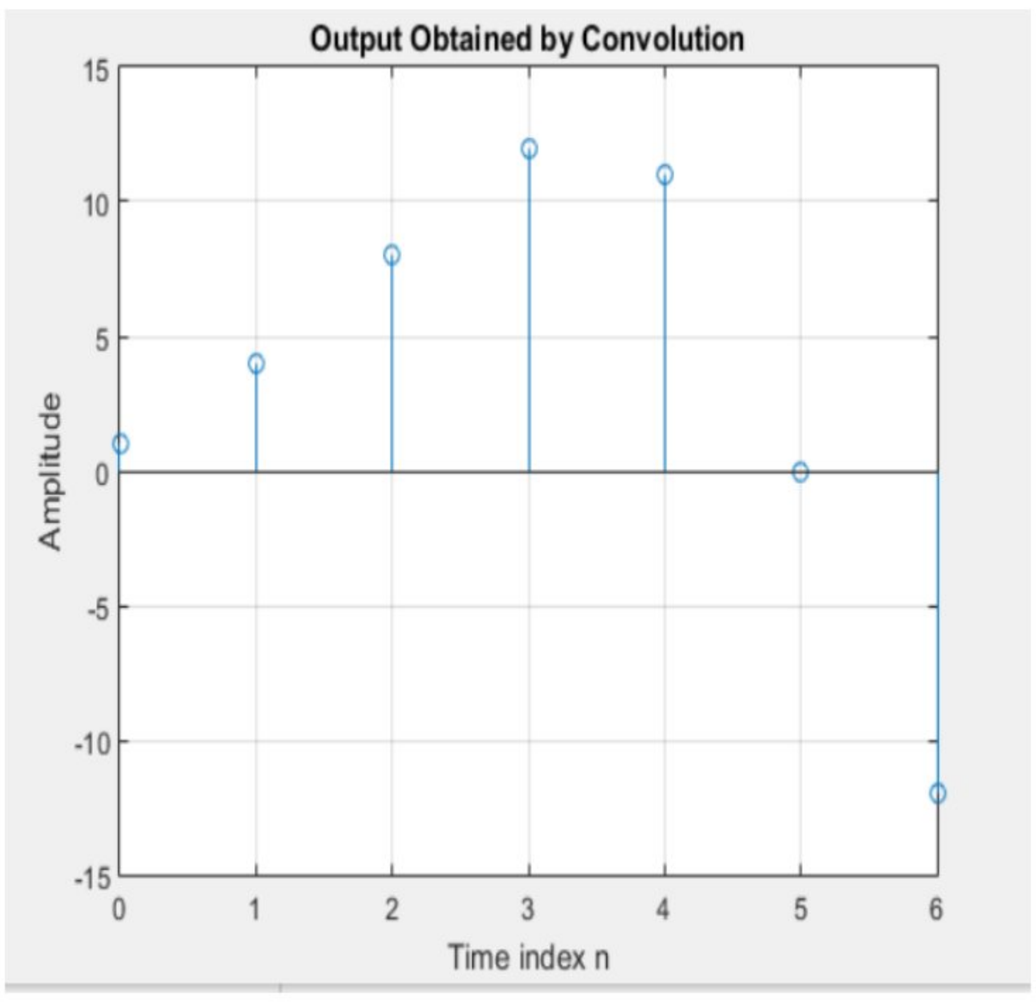


Figure:Response of LTI System

11.

Name of the experiment: Write a MATLAB and Python program to obtain Cross-correlation of sequence x[n] = [… 0, 0, 2, -1, 3, 7, $\underset{\uparrow}{1}$, 2, -3, 0, 0,…] and y[n] = […,0, 0, 1, -1, 2, -2, $\underset{\uparrow}{4}$, 1, -2, 5, 0, 0,… .] autocorrelation of a sequence x [n] of the given sequences & verify the property.

Theory:

The correlation of two functions or signals or waveforms is defined as the measure of similarity between those signals. There are two types of correlations :Cross-correlation,Autocorrelation.

Cross-correlation: The cross-correlation between two different signals indicates the degree of relatedness between one signal and the time-delayed version of another signal.The cross-correlation of energy (or aperiodic) signals and power (or periodic) signals is defined separately.Let, two complex signals $x_1(t)$ and $x_2(t)$ of finite energy. Then, the cross-correlation of these two energy signals is defined as

$R_{12}(\tau)= \int_{-\infty}^{\infty} x1(t)x*2(t-\tau)dt = \int_{-\infty}^{\infty} x1(t+\tau)x*2(t)dt$, $R_{xy}(l)=\sum_{=\infty}^{\infty} x(l)y(l-k)$

Autocorrelation: The autocorrelation function is defined as the measure of similarity or coherence between a signal and its time delayed version. Therefore, the autocorrelation is the correlation of a signal with itself.Like cross-correlation, autocorrelation is also defined separately for energy (or aperiodic) signals and power (periodic) signals. The autocorrelation of an energy or aperiodic signal x(t) is defined as −

$R_{11}(\tau)=R(\tau)=\int_{-\infty}^{\infty} x(t)x*(t-\tau)dt$

Source code:

Cross-correlation:

```
clc;clear all;close all;
x=[2 -1 3 7 1 2 -3];nx=[-4 -3 -2 -1 0 1 2 ];
y=[1 -1 2 -2 4 1 -2 5];ny=[-4 -3 -2 -1 0 1 2
3];a=xcorr(x,y);na = 0:length(a)-1;
subplot(3,1,1);stem(nx,x,'g');
xlabel('time');ylabel('amplitude');title('1stsequence
');
subplot(3,1,2);stem(ny,y,'r');xlabel('time');ylabel('
amplitude');title('2nd sequence');
subplot(3,1,3);stem(na,a,'r');xlabel('time');ylabel('
amplitude');title('cross correlation sequence');
```

Auto-correlation:

```
x=[2 -1 3 7 1 2 -3];
nx=[-4 -3 -2 -1 0 1 2 ];
a=xcorr(x);
na = 0:length(a)-1;
subplot(2,1,1);stem(x,'b');
xlabel('time');
ylabel('amplitude');title('input sequence');
subplot(2,1,2);stem(a,'r');
xlabel('time');
ylabel('amplitude');
title('auto correlation sequence');
```

Output:

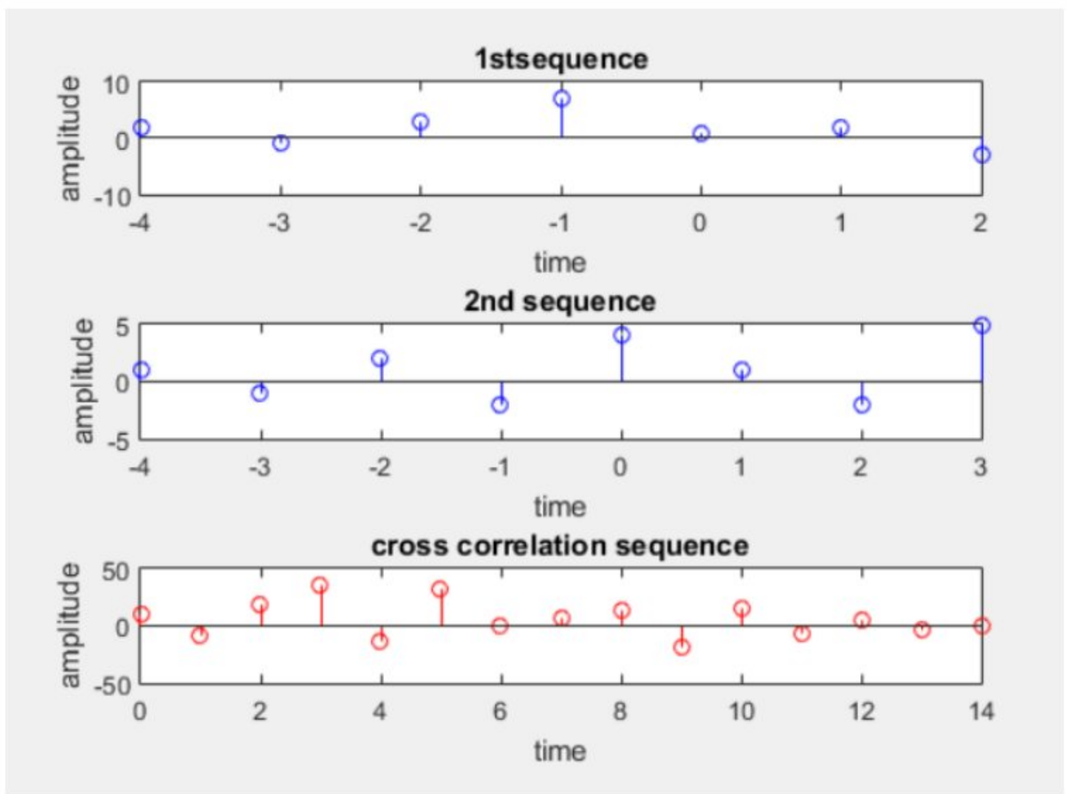


Figure:Cross-correlation sequence

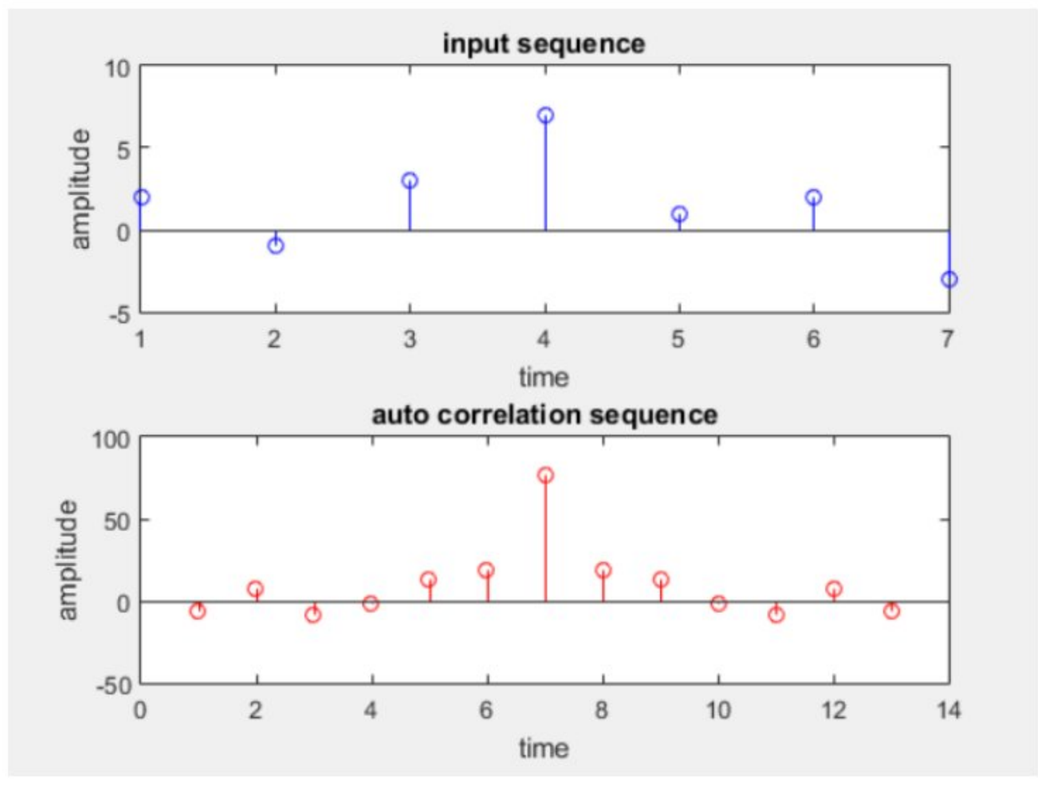


Figure:Auto-correlation sequence

Python:

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import correlate as cr
x = np.array([2, -1, 3, 7, 1, 2, -3])
y = np.array([1, -1, 2, -2, 4, 1, -2, 5])
cross_corr = cr(x, y, mode='full')
plt.stem(lag, cross_corr)
plt.title('Cross-Correlation of x[n] and y[n]')
plt.show()
auto_corr = cr(x, x, mode='full')
x = np.array([2, -1, 3, 7, 1, 2, -3])
la = np.arange(1,auto_corr.size+1)
plt.stem(la, auto_corr)
plt.title('Autocorrelation of x[n]')
plt.show()
```

Discussion:

From this experiment we see the properties of cross-correlation and auto-correlation.Auto-correlation is a mathematical tool to understand the characteristics of a signal and its self-similarity. It's commonly used in fields like signal processing, time series analysis, and communication systems to extract useful information about the underlying data. Cross-correlation is closely related to convolution. Specifically, the cross-correlation of signals x[n] and y[n] is equivalent to the convolution of signal x[n] with the time-reversed version of signal y[-n].The peak at a time delay of zero indicates that the two sequences are most similar when they are aligned. The decrease in the correlation and autocorrelation as the time delay increases indicates that the two sequences become less similar as they are shifted apart. The results can also be used to design and optimize algorithms that use correlation and autocorrelation.