Linear Convolution

Aim:

To find linear convolution of following sequences with and without built in function.

- 1. x(n) = [1 2 1 1] h(n) = [1 1 1 1]
- 2. x(n) = [1 2 1 2]
  h(n) = [3 2 1 2]

Theory:

Linear convolution is a mathematical operation used to combine two signals to produce a third signal. It's a fundamental operation in signal processing and systems theory.

Mathematical Definition:

Given two signals, *x(t)* and *h(t)*, their linear convolution is defined as:

$$\mathbf{y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\,d\tau}$$

Applications:

Filtering: Convolution is used to filter signals, removing unwanted frequencies or noise.

System Analysis: The impulse response of a system completely characterizes its behaviour, and convolution can be used to determine the output of the system given a known input.

Image Processing: Convolution is used for tasks like edge detection, blurring, and sharpening images.

Program:

1. With built-in function:

```
clc; clear all; close all; x1 =
input("Enter first Sequence"); h1 =
input("Enter second Sequence"); y1 =
conv(x1,h1); disp("The convoluted
sequence is: "); disp(y1); l =
length(x1); m = length(h1); k = l+m-
1; n1 = 0:1:l-1; n2 = 0:1:m-1; n3 =
0:1:k-1; subplot(1,3,1);
```

```
stem(n1,x1,"o"); xlabel("n");
ylabel("Amplitude"); title("x(n)");
grid on xlim([-1 l+1]); ylim([0
max(x1)+2]);  subplot(1,3,2);
stem(n2,h1,"o"); xlabel("n");
ylabel("Amplitude"); title("h(n)");
grid on xlim([-1 m+1]); ylim([0
max(h1)+2]);  subplot(1,3,3);
stem(n3,y1,"o"); xlabel("n");
ylabel("Amplitude"); title("y(n)");
grid   on   xlim([-1
k+1]);       ylim([0
max(y1)+2]);
```

2.Without built-in function:

```
clc; clear all; close all; x1 =
input("Enter first Sequence"); h1 =
input("Enter second Sequence"); l =
length(x1); m = length(h1); k =
l+m-1; y1 = zeros(1,k); for i=1:l
for j=1:m
y1(i+j-1) = y1(i+j-1) + x1(i)*h1(j);
end end
disp("The convoluted sequence is: ");
disp(y1);  n1 = 0:1:l-1; n2 = 0:1:m-
1; n3 = 0:1:k-1; subplot(1,3,1);
stem(n1,x1,"o"); xlabel("n");
ylabel("Amplitude"); title("x(n)");
```

```
grid   on   xlim([-1
l+1]);       ylim([0
max(x1)+2]);
 subplot(1,3,2);
stem(n2,h1,"o");
xlabel("n");
ylabel("Amplitude");
title("h(n)"); grid
on xlim([-1 m+1]);
ylim([0 max(h1)+2]);
 subplot(1,3,3);
stem(n3,y1,"o");
xlabel("n");
ylabel("Amplitude");
title("y(n)"); grid
on xlim([-1 k+1]);
ylim([0 max(y1)+2]);
```

Result:

Performed Linear Convolution using with and without built-in function.

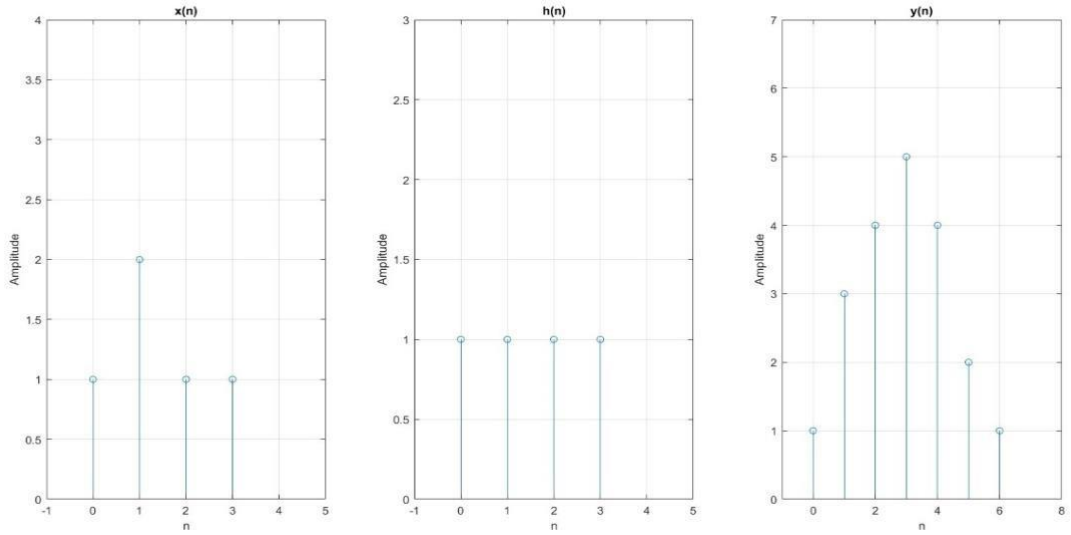
Observation:

a) Enter first Sequence [1 2 1 1]

Enter second Sequence [1 1 1 1] The

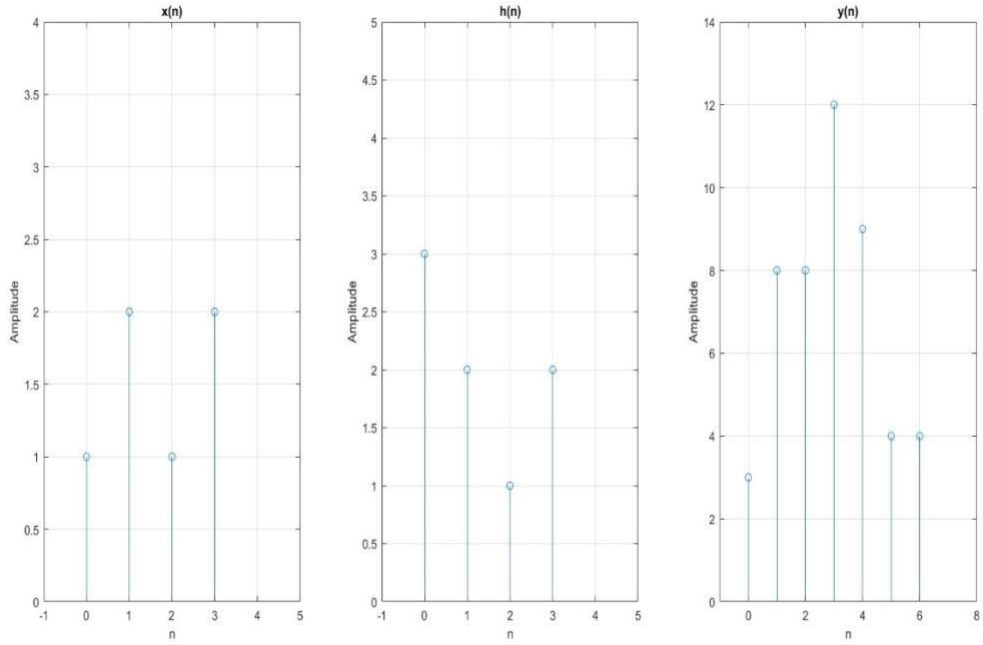
convoluted sequence is:

1 3 4 5 4 2 1



b) Enter first Sequence [1 2 1 2] Enter second Sequence [3 2 1 2] The convoluted sequence is:

3 8 8 12 9 4 4



Experiment No: 4 Date: 03/09/24

Circular Convolution

Aim:

To find circular convolution

a. Using FFT and IFFT.
b. Using Concentric Circle Method.
c. Using Matrix Method.

Theory:

Circular convolution is a mathematical operation that is like linear convolution but is performed in a periodic or circular manner. This is particularly useful in discrete-time signal processing where signals are often represented as periodic sequences.

Mathematical Definition:

Given two periodic sequences *x[n]* and *h[n]*, their circular convolution is defined as:

$y[n] = (x[n] \circledast h[n]) = \sum_{k=0}^{N-1} x[k]h[(n-k) \bmod N]$ Applications:

- Discrete-Time Filtering: Circular convolution is used for filtering discrete-time signals.
- Digital Signal Processing: It's a fundamental operation in many digital signal processing algorithms.
- Cyclic Convolution: In certain applications, such as cyclic prefix OFDM, circular convolution is used to simplify the implementation of linear convolution.

Program:

a. Using FFT and IFFT.

```
clc; close all;

clear all; x1

= [1 2 1 2]; x2

= [1 2 3 4];

X1_k = fft(x1);

X2_k = fft(x2);

Y1_k = X1_k.*X2_k;
y1      =      ifft(Y1_k);

disp("Using FFT and IFFT:")

disp(y1); clc; close all;

clear all; x1 = input("1st

seq"); x2 = input("2nd

seq"); l = length(x1); m =

length(x2); k = max(l,m); x1
```