Aim ↬ To understand and implement the linear convolution and circular convolution of signals using MATLAB.

Software Required ↬ MATLAB

Theory ↬

Convolution is a mathematical operation that combines two signals to produce a third signal. It is a fundamental tool in signal processing, especially in the analysis of linear time-invariant (LTI) systems. The convolution of two signals provides a way to determine how the shape of one signal is modified by another.

Linear Time-Invariant (LTI) System ↴

A Linear Time-Invariant (LTI) system is a system that satisfies two key properties:

1. **Linearity**: The principle of superposition applies. If the system's response to $x_1(t)$ is $y_1(t)$ and the response to $x_2(t)$ is $y_2(t)$, then the response to $a \cdot x_1(t)+b \cdot x_2(t)$ is $a \cdot y_1(t)+b \cdot y_2(t)$, where a and b are constants.
2. **Time-Invariance**: The system's behavior does not change over time. If the response to x(t) is y(t), then the response to $x(t-t_0)$ is $y(t-t_0)$ for any time shift t_0.

Linear Convolution ↴

Linear convolution is used for signals that are not necessarily periodic, and it represents the way a system's output is generated from an input signal and its impulse response.

Continuous-Time Convolution ↴

For continuous-time signals, the convolution of two signals x(t) as the input signal and h(t) as the impulse response of the system is defined as:

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

Discrete-Time Convolution ↴

For discrete-time signals, the convolution of two signals x[n] as the input signal and h[n] as the impulse response of the system is defined as:

$$x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

Circular Convolution ↴

Circular convolution applies to periodic signals, where the signals repeat after a fixed period. This is especially useful in digital signal processing for periodic or finite-length sequences. In discrete-time, circular convolution is defined as:

$$x[n]©h[n] = \sum_{k=0}^{N-1} x[k]h[(n-k)\% \, N]$$

Here N is the period, and indices are taken modulo N to enforce periodicity.

Properties of Convolution ↴

1. Commutativity: $x[n] * h[n] = h[n] * x[n]$
2. Associativity: $x[n] * (h[n] * g[n]) = (x[n] * h[n]) * g[n]$
3. Distributivity: $x[n] * (h[n] + g[n]) = x[n] * h[n] + x[n] * g[n]$
4. Identity: Convolution with a delta function δ(t) or δ[n] yields the original signal: $x(t) * \delta(t) = x(t)$ or $x[n] * \delta[n] = x[n]$.

Code ↬

```
%Linear Convolution
clc;
clear;

x_n = [3, 2.5, -1, 1.4];
y_n = [-0.2, 2.1, 0.7, 1.2];
z_n = [1, -0.6, 2.8];

%Using built-in function
f_n = conv(x_n,y_n);

figure;
subplot(2,2,1);
stem(x_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("x_n");

subplot(2,2,2);
stem(y_n,'filled');
```

```
xlabel("n");
ylabel("Amplitude");
title("y_n");

subplot(2,2,3);
stem(f_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("z_n = x_n * y_n");

%Using Formula
l_x = length(x_n);
l_y = length(y_n);
l_k = l_x + l_y - 1;

c1_n = zeros(1,l_k);

for n=1:l_k
    for k=1:l_x
        if n-k+1>0 && n-k+1<=l_y
            c1_n(n) = c1_n(n) + x_n(k)*y_n(n-k+1);
        end
    end
end

subplot(2,2,4);
stem(c1_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("k_n = x_n * y_n");

%Commutative property
c1_n = conv(x_n,y_n);
c2_n = conv(y_n,x_n);

%Associative property
a1_n = conv(conv(x_n,y_n),z_n);
a2_n = conv(x_n,conv(y_n,z_n));

%Distributive property
```

```
d1_n = conv(z_n,x_n+y_n);
d2_n = conv(z_n,x_n)+conv(z_n,y_n);

figure;
subplot(3,2,1);
stem(c1_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("c1_n = x_n * y_n");

subplot(3,2,2);
stem(c2_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("c2_n = y_n * x_n");

subplot(3,2,3);
stem(a1_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("a1_n = (x_n * y_n) * z_n");

subplot(3,2,4);
stem(a2_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("a2_n = x_n * (y_n * z_n)");

subplot(3,2,5);
stem(d1_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("d1_n = z_n * (x_n + y_n)");

subplot(3,2,6);
stem(d2_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("d2_n = (z_n * x_n) + (z_n * y_n)");
```

```
%Circular Convolution
clc;
clear;

x_n = [3, 2.5, -1, 1.4, -0.7];
y_n = [-0.2, 2.1, 0.7];

%Using built-in function
N = max(length(x_n),length(y_n));

z_n = cconv(x_n,y_n,N);

figure;
subplot(2,2,1);
stem(x_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("x_n");

subplot(2,2,2);
stem(y_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("y_n");

subplot(2,2,3);
stem(z_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("z_n = x_n * y_n");

%Using Formula
x_n_f = [x_n,zeros(1,N-length(x_n))];
y_n_f = [y_n,zeros(1,N-length(y_n))];
k_n = zeros(1,N);

for n=1:N
    for k=1:N
        k_n(n) = k_n(n) + x_n_f(k)*y_n_f(mod(n-k,N)+1);
    end
```

```
end

subplot(2,2,4);
stem(k_n,'filled');
xlabel("n");
ylabel("Amplitude");
title("k_n = x_n * y_n");
```

Output ↬

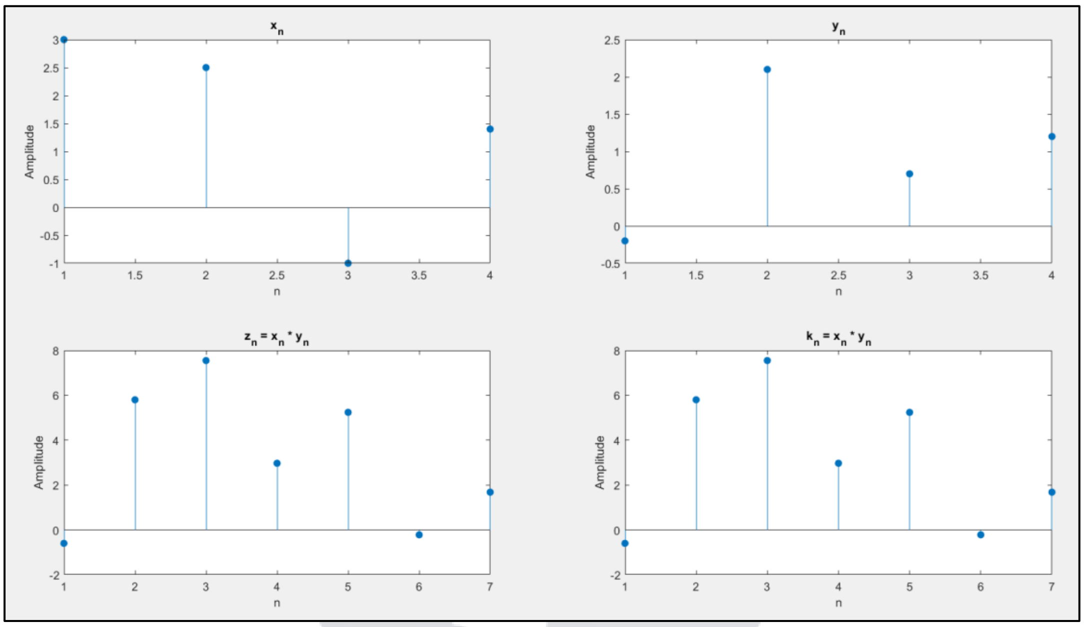

Fig. i) Plots for Linear Convolution

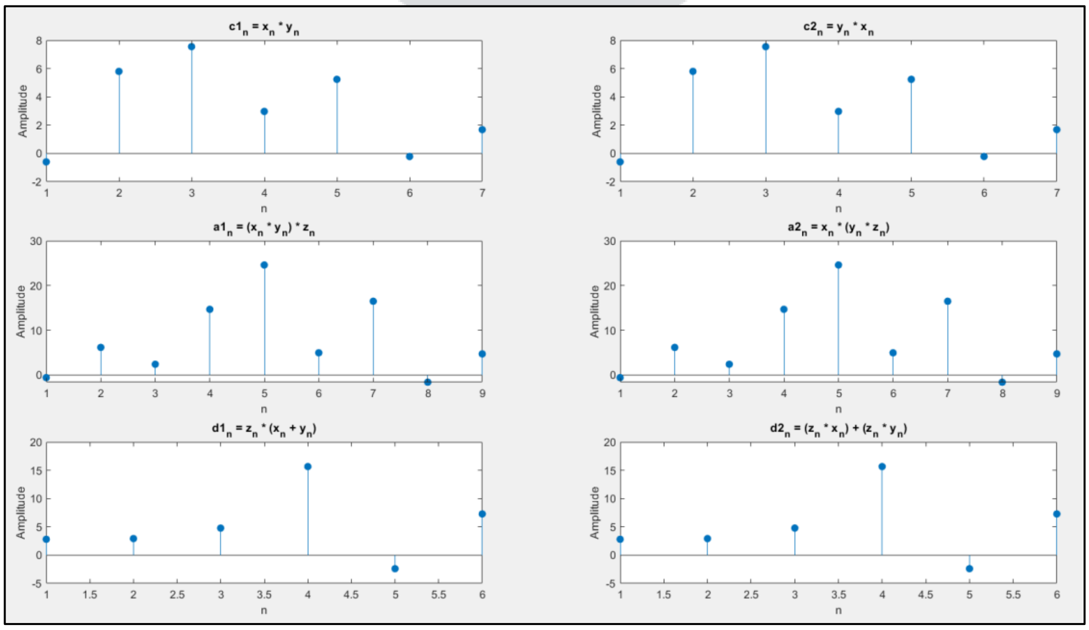

Fig. ii) Plots for properties of Convolution

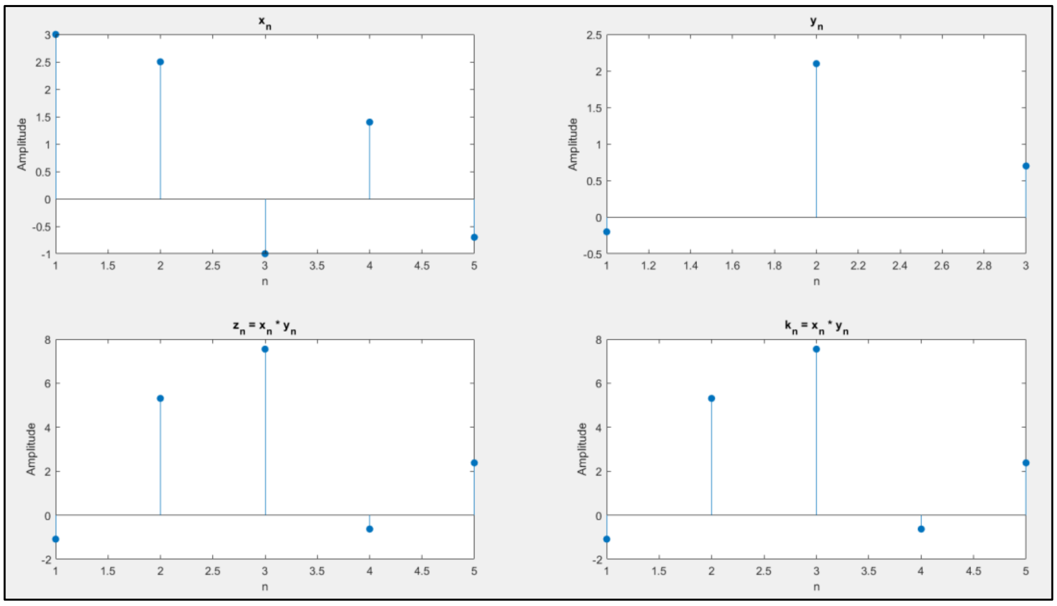

Fig. iii) Plots for Circular Convolution

Result ↬

The experiment was successfully performed by implementing the linear convolution and circular convolution of signals. It showed how the impulse response modifies the input. Circular convolution involves periodic sequences, producing outputs that wrap around after a set period.

Conclusion ↬

Linear convolution shows input-output interaction, with continuous time approximated by discrete sampling. Circular convolution handles periodic signals, emphasizing time invariance and linearity.

Precautions ↬

- Verify sequence lengths are suitable for convolution to avoid artifacts.
- Ensure correct interpretation of the convolution output about input sequences.
- Double-check input parameters for accuracy.
- Confirm that plots accurately represent the convolution results.