**Batch: A-3 Roll No.: 16010122104**

**Experiment No. 4**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

|  |
| --- |
| **Title:** Write a program to Compute linear and circular convolution of two discrete time signal sequences using MATLAB. |

**Objective:** To familiarize the beginnerto MATLAB by introducing the basic features and commands of the program.

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| **CO3** | To understand the concept of convolution and perform different convolution operations on the given input signals. |

**Books/ Journals/ Websites referred:**

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html
3. www.mccormick.northwestern.edu/docs/efirst/matlab.pdf
4. A.Nagoor Kani “Digital Signal Processing”, 2nd Edition, TMH Education.

**Pre Lab/ Prior Concepts:**

**Convolution**

Discrete time convolution is a method of finding response of linear time invariant system. It is based on the concepts of linearity and time invariance and assumes that the system information

is known in terms of its impulse response h[n].

Convolution is defined as

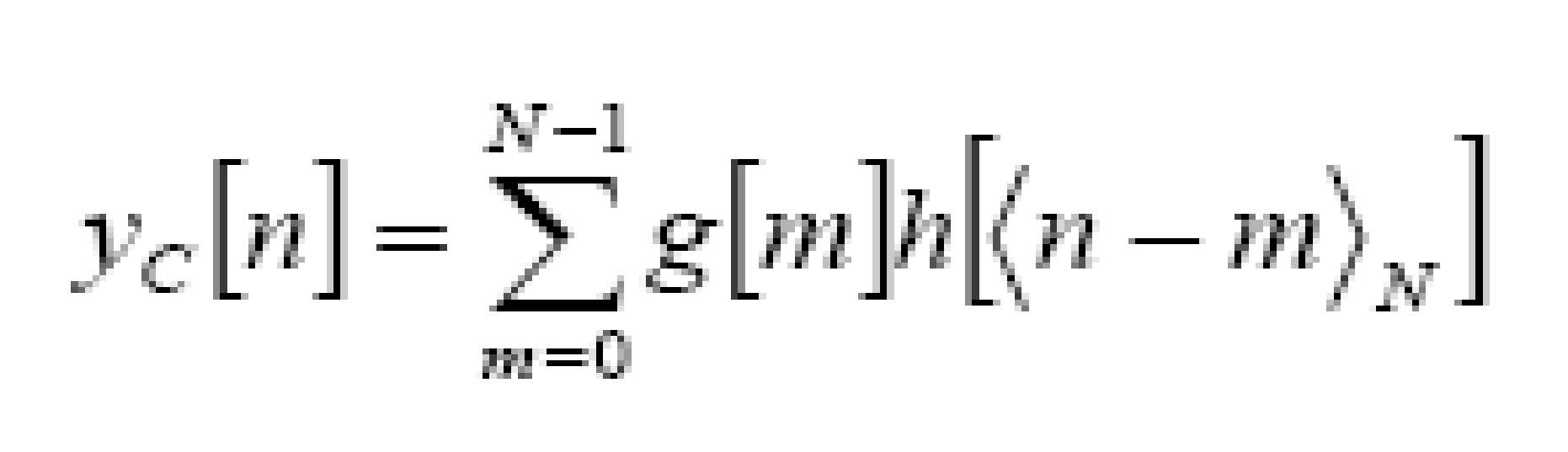
∞

Y[n] = Σ h[k]x [n-k] =h[n]\*x[n] k=-∞

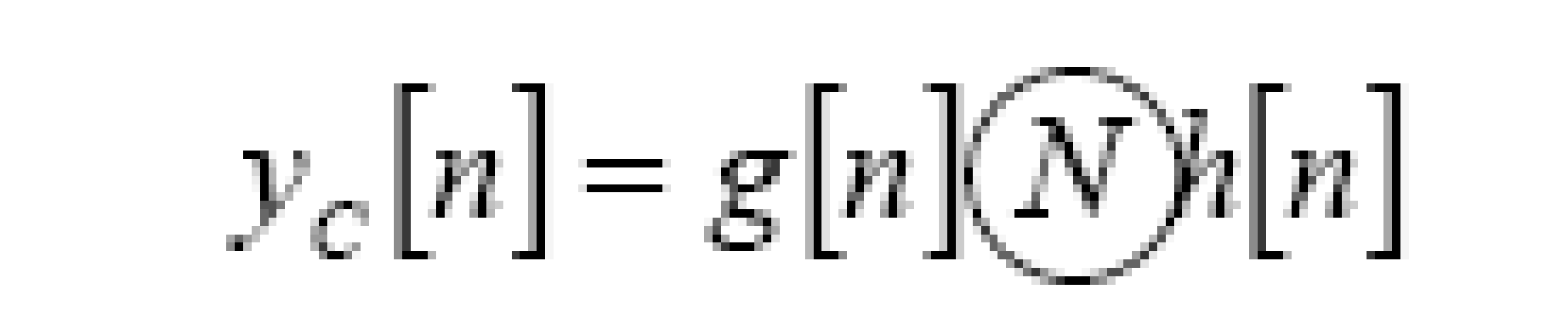
Convolution consists of folding, shifting, Multiplication and summation operations.

**Circular Convolution**

Circular convolution between two length N sequences can be carried out as shown by the expression below:



Since the above operation involves two length-N sequences it is referred to as the N-point circular convolution and denoted by:



As in linear convolution circular convolution is commutative.

i.e.



**Example Of Linear Convolution:**

Let x[n] = {1, 2, 3} and h[n] = {1, 1, 1}

Step 1: Determine the length of the output (N + M - 1 = 3 + 3 - 1 = 5)  
Step 2: Align and multiply the sequences  
Step 3: Sum the products

n = 0: y = 1 \* 1 = 1  
n = 1: y1 = 1 \* 2 + 2 \* 1 = 4  
n = 2: y = 1 \* 3 + 2 \* 1 + 3 \* 1 = 8  
n = 3: y = 2 \* 1 + 3 \* 1 = 5  
n = 4: y = 3 \* 1 = 3

The result of linear convolution is y[n] = {1, 4, 8, 5, 3}

**Example Of Circular Convolution:**

Let x[n] = {1, 2, 3, 4} and h[n] = {2, 1, 2, 1}

Step 1: Determine the length of the output (N = 4)  
Step 2: Circularly shift h[n] and multiply with x[n]  
Step 3: Sum the products for each shift

n = 0: y = 12 + 21 + 32 + 41 = 13  
n = 1: y1 = 11 + 22 + 31 + 42 = 14  
n = 2: y = 12 + 21 + 32 + 41 = 13  
n = 3: y = 11 + 22 + 31 + 42 = 14

The result of circular convolution is y[n] = {13, 14, 13, 14}

**Implementation details along with screenshots:**

**Linear Convolution:**

% Prompt the user for input signals x(n) and h(n)

x = input('Enter the signal x(n) as a vector (e.g., [1, 2, 3, 4]): ');

h = input('Enter the signal h(n) as a vector (e.g., [2, 1, 0.5]): ');

% Lengths of the signals

Lx = length(x);

Lh = length(h);

% Length of the output (circular convolution result)

N = max(Lx, Lh);

% Pad sequences to equal length N

x = [x, zeros(1, N - Lx)];

h = [h, zeros(1, N - Lh)];

% Initialize the output signal y(n) to zero

y = zeros(1, N);

% Step-by-step circular convolution

for n = 1:N

% Create a new figure for each step

figure;

% Plot x(n) in the first subplot

subplot(3, 1, 1);

stem(0:N-1, x, 'filled', 'r');

title(['x(n) = ', mat2str(x)]);

axis([-1 N 0 max(x)\*1.2]);

xlabel('n');

ylabel('Amplitude');

% Plot h(n) at the current step (circularly shifted version) in the second subplot

h\_shifted = circshift(h, -(n-1));

subplot(3, 1, 2);

stem(0:N-1, h\_shifted, 'filled', 'b');

title(['Shifted h(n) at n = ', num2str(n-1)]);

axis([-1 N 0 max(h)\*1.2]);

xlabel('n');

ylabel('Amplitude');

% Element-wise multiplication (x(n) and shifted h(n)) in the third subplot

multiplication = x .\* h\_shifted;

subplot(3, 1, 3);

stem(0:N-1, multiplication, 'filled', 'g');

title(['Element-wise multiplication at n = ', num2str(n-1)]);

axis([-1 N min(multiplication)\*1.2 max(multiplication)\*1.2]);

xlabel('n');

ylabel('Product');

% Compute the sum for the current output y(n)

y(n) = sum(multiplication);

% Pause to view the results

pause(1);

end

% Create a final figure for the complete circular convolution result

figure;

stem(0:N-1, y, 'filled', 'g');

title('Final Circular Convolution Result');

axis([-1 N-1 min(y)\*1.2 max(y)\*1.2]);

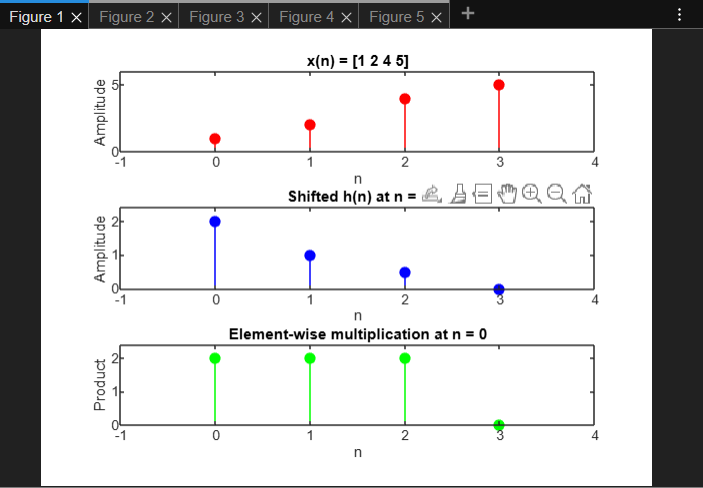
xlabel('n');

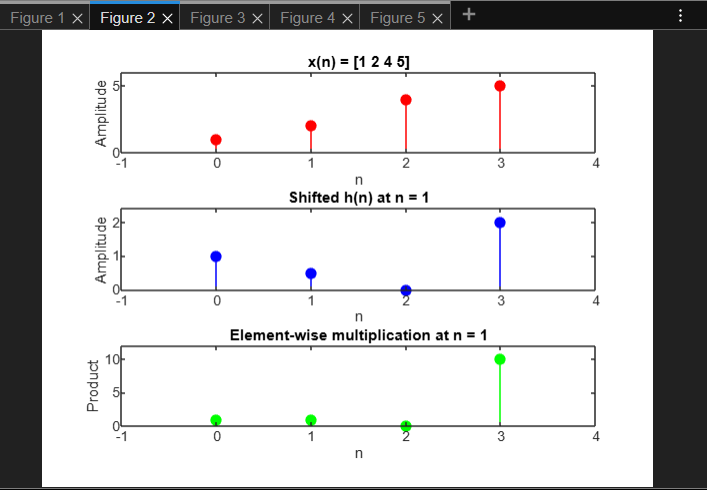
ylabel('y(n)');

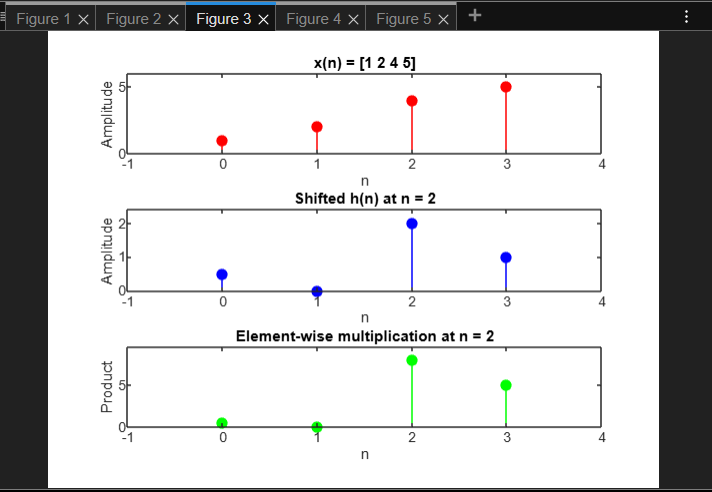
% Display the final circular convolution result

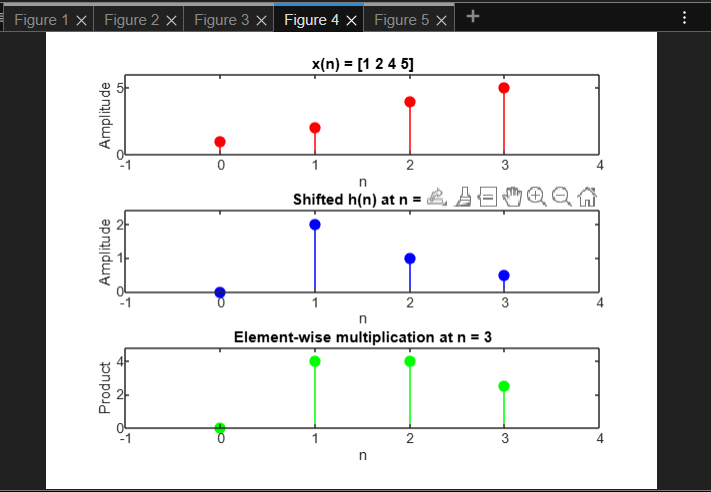
disp('Final Circular Convolution Result y(n):');

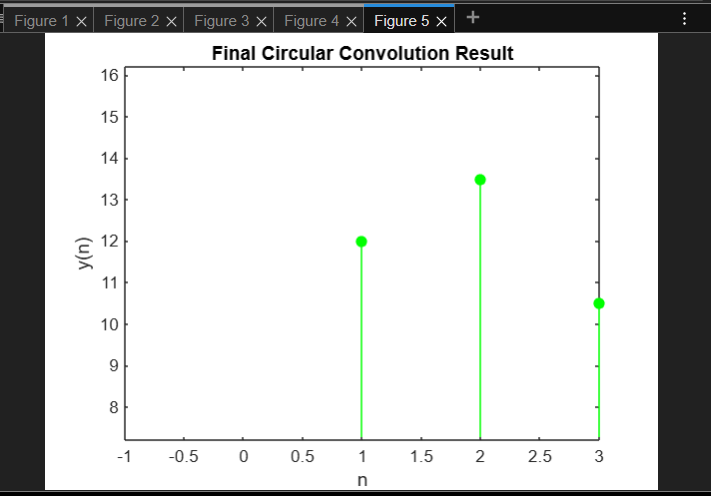
disp(y);

****

****

****

****

****

**Circular Convolution:**

% Prompt the user for input signals x(n) and h(n)

x = input('Enter the signal x(n) as a vector (e.g., [1, 2, 3, 4]): ');

h = input('Enter the signal h(n) as a vector (e.g., [2, 1, 0.5]): ');

% Lengths of the signals (assuming both signals are of the same length)

N = length(x);

% Initialize the output signal y(n) to zero

y = zeros(1, N);

% Step-by-step circular convolution

for n = 1:N

% Create a new figure for each step

figure;

% Plot x(n) in the first subplot

subplot(3, 1, 1);

stem(0:N-1, x, 'filled', 'r'); hold on;

title(['x(n) = ', mat2str(x)]);

axis([-1 N 0 max(x)\*1.2]);

xlabel('n');

ylabel('Amplitude');

% Plot h(n) at the current step (shifted version) in the second subplot

h\_shifted = circshift(h, -(n-1));

subplot(3, 1, 2);

stem(0:N-1, h\_shifted, 'filled', 'b');

title(['Shifted h(n) at n = ', num2str(n-1)]);

axis([-1 N 0 max(h)\*1.2]);

xlabel('n');

ylabel('Amplitude');

% Element-wise multiplication (x(n) and shifted h(n)) in the third subplot

multiplication = x .\* h\_shifted;

subplot(3, 1, 3);

stem(0:N-1, multiplication, 'filled', 'g');

title(['Element-wise multiplication at n = ', num2str(n-1)]);

% Ensure that the axis limits are valid for the multiplication plot

y\_min = min(multiplication) \* 1.2;

y\_max = max(multiplication) \* 1.2;

if y\_min == y\_max

y\_min = y\_min - 1;

y\_max = y\_max + 1;

end

axis([-1 N-1 y\_min y\_max]);

xlabel('n');

ylabel('Product');

% Compute the sum for the current output y(n)

y(n) = sum(multiplication);

% Pause to view the results

pause(1);

end

% Create a final figure for the complete circular convolution result

figure;

stem(0:N-1, y, 'filled', 'g');

title('Final Circular Convolution Result');

axis([-1 N-1 min(y)\*1.2 max(y)\*1.2]);

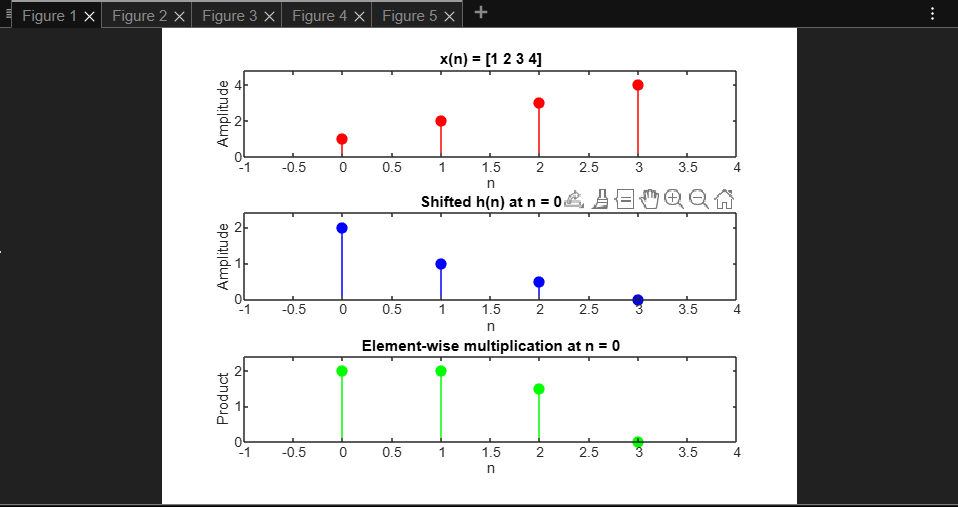
xlabel('n');

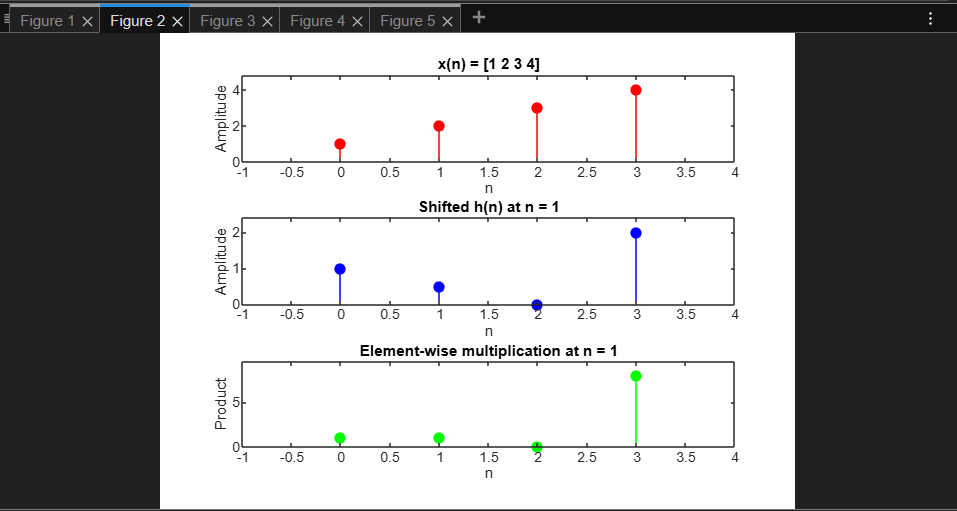
ylabel('y(n)');

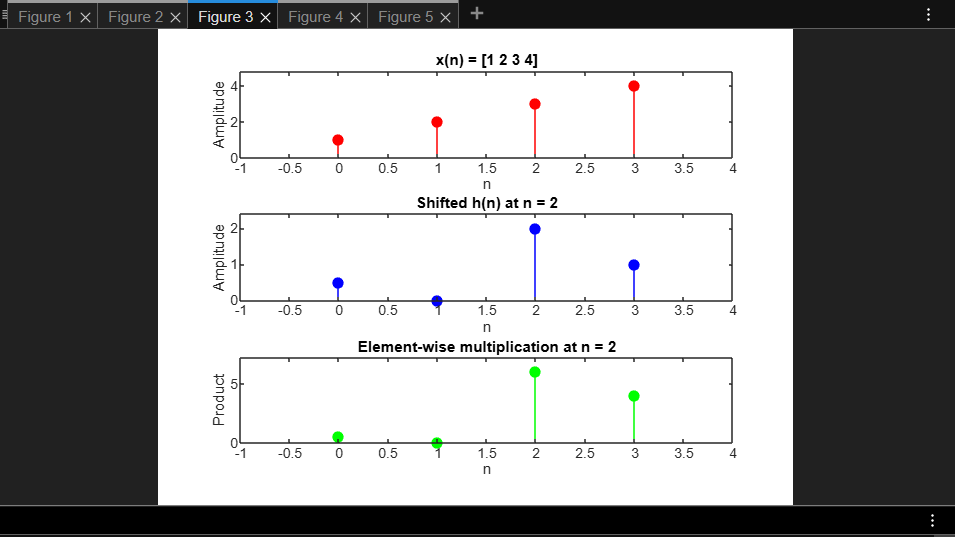
% Display the final circular convolution result

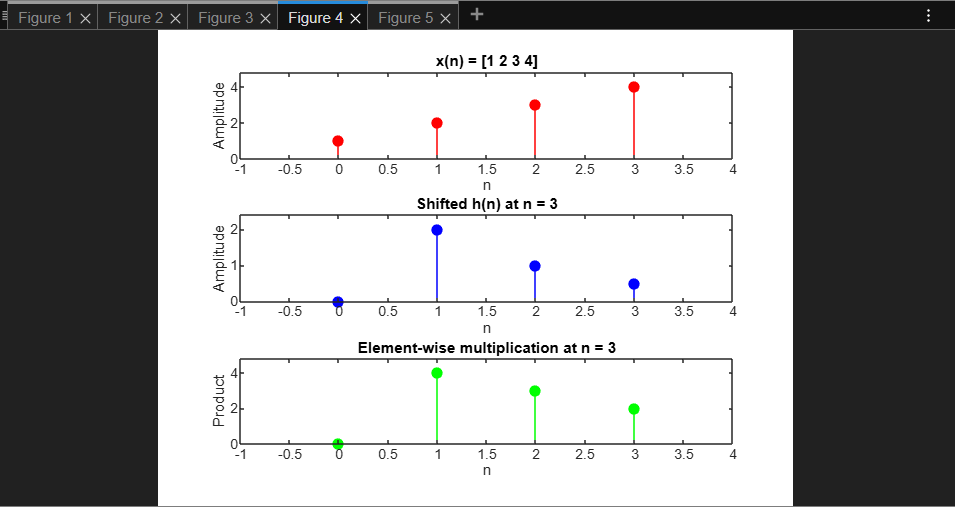
disp('Final Circular Convolution Result y(n):');

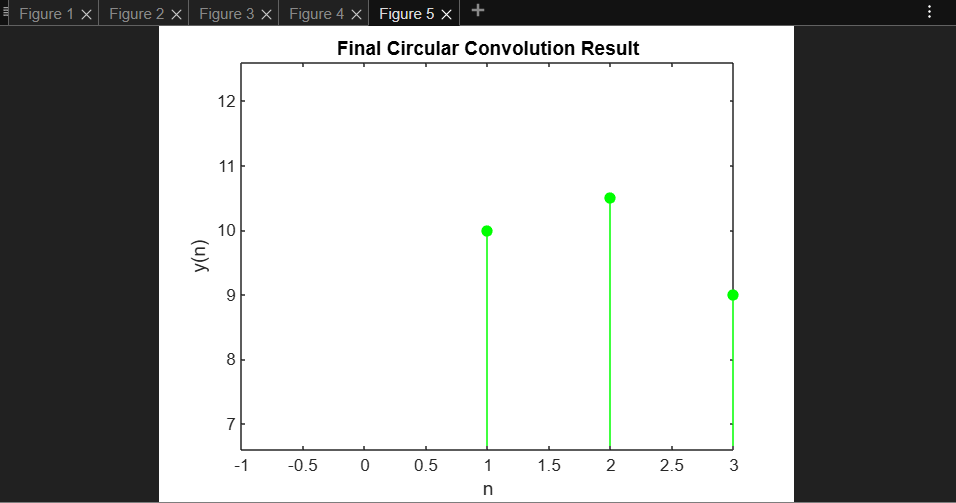
disp(y);

****

****

****

****

****

**Conclusion:-**

Write a program to Compute linear and circular convolution of two discrete time signal sequences using MATLAB.

**Date: 18/02/2025 Signature of faculty in-charge**

**Post Lab Descriptive Questions**

* 1. Explain the role of convolution in signal processing.

Ans:

Convolution plays a crucial role in signal processing as it allows us to determine the output of a linear time-invariant (LTI) system for any given input signal. Its key functions include:

1. System response calculation: Convolution helps compute the output of an LTI system when given an input signal and the system's impulse response.
2. Filtering: It is fundamental in implementing digital filters, enabling the removal of unwanted frequencies or enhancement of specific signal components.
3. Feature extraction: Convolution can be used to detect or extract specific features from signals.
4. Signal analysis: It aids in analyzing signals by breaking them down into simpler components.
   1. Explain the difference between linear and circular convolution?

Ans:

Linear convolution and circular convolution differ in several aspects:

1. Periodicity: Linear convolution operates on non-periodic sequences, resulting in a non-periodic output. Circular convolution assumes the input sequences are periodic and produces a periodic output.
2. Length of output: For linear convolution of sequences with lengths N and M, the output length is N+M-1. In circular convolution, the output length is equal to the length of the longer input sequence.
3. Computation method: Linear convolution involves sliding one sequence past the other. Circular convolution involves wrapping the sequences around themselves119.
4. Application: Linear convolution is used in general signal processing tasks. Circular convolution is particularly useful in frequency domain analysis and fast convolution algorithms.
   1. Explain with the help of an example the steps required to transform linear convolution with circular convolution and vice-versa.

Ans:

To transform linear convolution to circular convolution:

1. Determine the length of linear convolution output (N+M-1).
2. Zero-pad both input sequences to this length.
3. Perform circular convolution on the zero-padded sequences.

Example:  
x[n] = {1, 2, 3}, h[n] = {1, 1, 1}  
Linear convolution output length: 3 + 3 - 1 = 5  
Zero-padded x[n] = {1, 2, 3, 0, 0}  
Zero-padded h[n] = {1, 1, 1, 0, 0}  
Perform circular convolution on these padded sequences.

To transform circular convolution to linear convolution:

1. Determine the length of desired linear convolution output (N+M-1).
2. Perform circular convolution with this length.
3. Take the first N+M-1 samples of the circular convolution output.

Example:  
x[n] = {1, 2, 3}, h[n] = {1, 1, 1}  
Desired linear convolution output length: 3 + 3 - 1 = 5  
Perform circular convolution with length 5  
Take the first 5 samples of the circular convolution output.