Digital Signal Processing Lab

Laboratory report submitted for the partial fulfillment
of the requirements for the degree of

Bachelor of Technology
in
Electronics and Communication Engineering

by

Mohit Akhouri - 19ucc023

Course Coordinator
Dr. Divyang Rawal

Department of Electronics and Communication Engineering
The LNM Institute of Information Technology, Jaipur

September 2021

Contents

Chapter *3*

Experiment - 3

3.1 Aim of the Experiment

- Linear Convolution and DFT matrix Generation
- Simulink based convolution

3.2 Software Used

- MATLAB
- Simulink

3.3 Theory

3.3.1 About Linear Time invariant (LTI) system :

In system analysis, among other fields of study, a **linear time-invariant system** (LTI system) is a system that produces an output signal from any input signal subject to the constraints of **linearity** and **time-invariance**. These properties apply (exactly or approximately) to many important physical systems, in which case the response **y(t)** of the system to an arbitrary input **x(t)** can be found directly using convolution: **y(t) = x(t) h(t)** where **h(t)** is called the system's impulse response and represents convolution. A good example of an LTI system is any **electrical circuit** consisting of resistors, capacitors, inductors and linear amplifiers. Linear time-invariant system theory is also used in **image processing**.

LTI systems can also be characterized in the frequency domain by the system's transfer function, which is the **Laplace transform** of the system's impulse response (or Z transform in the case of discrete-time systems). As a result of the properties of these transforms, the output of the system in the frequency domain is the product of the transfer function and the transform of the input. In other words, **convolution** in the **time domain** is equivalent to **multiplication** in the frequency domain.

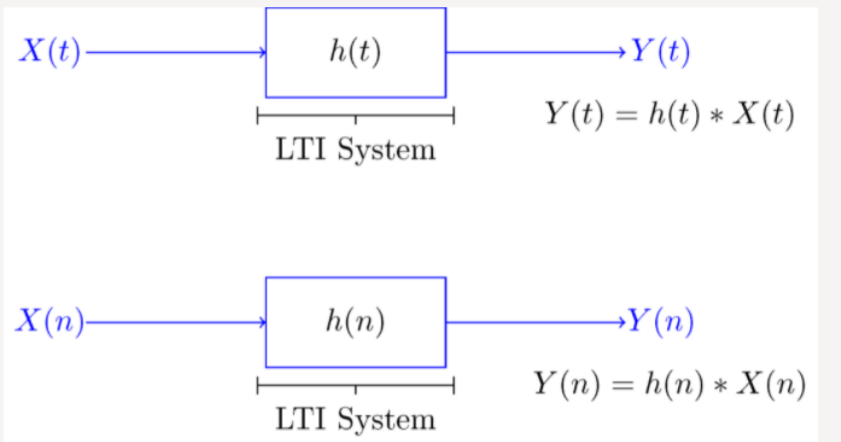

Figure 3.1 LTI system characterization

3.3.2 About Linear Convolution :

Linear convolution is a mathematical operation done to calculate the output of any Linear-Time Invariant (LTI) system given its **input** and **impulse response**. In linear convolution, both the sequences (input and impulse response) may or may not be of equal sizes. It is possible to find the response of a filter using linear convolution. The equation for Linear Convolution of **Discrete Time Sequence** is given as :

$$y[n] = x[n] * h[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] \tag{3.1}$$

In the above equation , **x[n]** is the **input sequence** and **h[n]** is the **Impulse response**.

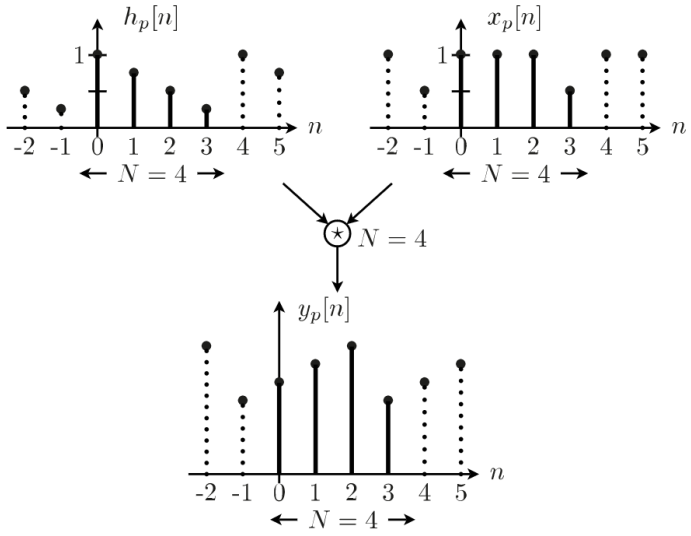

Figure 3.2 Linear Convolution of two finite length sequences

3.3.3 About Discrete Fourier Transform (DFT) :

The **discrete Fourier transform** (DFT) is one of the most important tools in digital signal processing. The DFT can calculate a signal's **frequency spectrum**. This is a direct examination of information encoded in the frequency, phase, and amplitude of the component sinusoids. For example, **human speech** and hearing use signals with this type of encoding. Second, the DFT can find a system's frequency response from the system's impulse response, and vice versa.

The DFT of a discrete-time signal x[n] with total **N samples** is given as :

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{\frac{-j2\pi nk}{N}} \tag{3.2}$$

3.3.3.1 About DFT matrix :

A **DFT matrix** is an expression of a discrete Fourier transform (DFT) as a **transformation matrix**, which can be applied to a signal through matrix multiplication.

DFT matrix can be constructed from **Twiddle factor** which is given as :

$$W_N = e^{\frac{-j2\pi}{N}} \tag{3.3}$$

The DFT can be calculated from DFT matrix and the equation for the same is given as:

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{3.4}$$

In the above equation , k ranges from 0 to N-1.

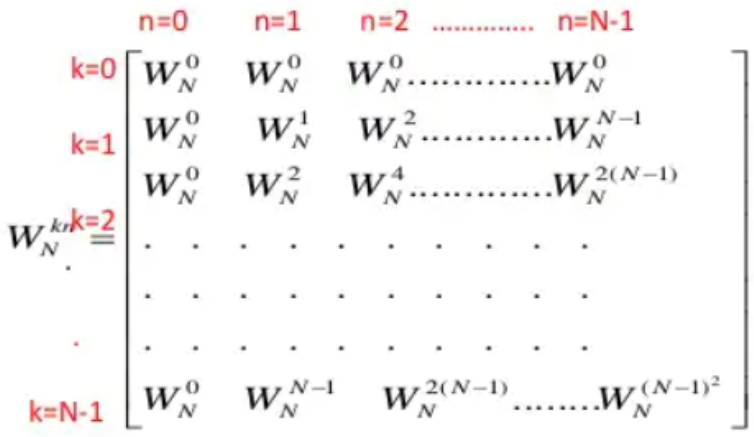

Figure 3.3 Twiddle Factor Matrix

3.4 Code and results

3.4.1 Linear Convolution using Convolution matrix M :

```
% 19ucc023
% Mohit Akhouri
% Experiment 3 - Observation 1 and Observation 2

% This code will calculate the convolution of x[n] and h[n]
% for different frequencies and impulse responses
% both by calling the user defined functions myLinConvMat and myConv
% and using inbuilt function 'conv'

% ALGORITHM : First we calculate the convolution matrix M and
% calculate the convolution using M and input sequence x[n]

clc;
clear all;
close all;

n = 0:1:99; % initializing the number of samples
h1 = [1 1]; % initializing impulse response h1[n]
h2 = [1 -1]; % initializing impulse response h2[n]
h3 = (1/3)*[1 1 1]; % initializing impulse response h3[n]
h4 = (1/4)*[1 1 -4 1 1]; % initializing impulse response h4[n]

freq = [0 1/10 1/5 1/4 1/2]; % initializing the frequency array

% Convolution for x and h1

h = h1; % temporary variable to store the different impulse responses
ind = 1; % index for correct printing of 'number of impulse response'
for i = 1:length(freq)
    xi = cos(2*pi*freq(i)*n); % initializing x[n] for different
 frequencies

    M = myLinConvMat(h,length(xi)); % storing convolution matrix in M
    y_conv = myConv(xi,M); % convolution of x[n] and h[n] using myConv
 function
    y_conv_inbuilt = conv(xi,h); % convolution of x[n] and h[n] using
 INBUILT function conv

    % plotting x[n], h[n]
    figure;
    subplot(2,2,1);
    stem(xi,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]->',i));
    title(sprintf('x_{%d}[n] = cos(2*%s*(%0.2f)*n)',i,'\pi',freq(i)));
    grid on;
    subplot(2,2,2);
    stem(h,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('h_{%d}[n]->',ind));
    title(sprintf('Impulse response h_{%d}[n]',ind));
    grid on;
```

Figure 3.4 Part 1 of the code for observation 1 and 2

```
    % plotting convolution using myConv and INBUILT FUNCTION conv
    subplot(2,2,3);
    stem(y_conv,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using myConv',i,ind,freq(i)));
    grid on;
    subplot(2,2,4);
    stem(y_conv_inbuilt,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using INBUILT function conv',i,ind,freq(i)));
    grid on;
    sgtitle('19ucc023 - Mohit Akhouri');
end

% Convolution for x and h2

h = h2; % temporary variable to store the different impulse responses
ind = 2; % index for correct printing of 'number of impulse response'
for i = 1:length(freq)
    xi = cos(2*pi*freq(i)*n); % initializing x[n] for different
 frequencies

    M = myLinConvMat(h,length(xi)); % storing convolution matrix in M
    y_conv = myConv(xi,M); % convolution of x[n] and h[n] using myConv
 function
    y_conv_inbuilt = conv(xi,h); % convolution of x[n] and h[n] using
 INBUILT function conv

    % plotting x[n], h[n]
    figure;
    subplot(2,2,1);
    stem(xi,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]->',i));
    title(sprintf('x_{%d}[n] = cos(2*%s*(%0.2f)*n)',i,'\pi',freq(i)));
    grid on;
    subplot(2,2,2);
    stem(h,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('h_{%d}[n]->',ind));
    title(sprintf('Impulse response h_{%d}[n]',ind));
    grid on;
    % plotting convolution using myConv and INBUILT FUNCTION conv
    subplot(2,2,3);
    stem(y_conv,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using myConv',i,ind,freq(i)));
    grid on;
```

Figure 3.5 Part 2 of the code for observation 1 and 2

```
    subplot(2,2,4);
    stem(y_conv_inbuilt,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using INBUILT function conv',i,ind,freq(i)));
    grid on;
    sgtitle('19ucc023 - Mohit Akhouri');
end

% Convolution for x and h3

h = h3; % temporary variable to store the different impulse responses
ind = 3; % index for correct printing of 'number of impulse response'
for i = 1:length(freq)
    xi = cos(2*pi*freq(i)*n); % initializing x[n] for different
 frequencies

    M = myLinConvMat(h,length(xi)); % storing convolution matrix in M
    y_conv = myConv(xi,M); % convolution of x[n] and h[n] using myConv
 function
    y_conv_inbuilt = conv(xi,h); % convolution of x[n] and h[n] using
 INBUILT function conv

    % plotting x[n], h[n]
    figure;
    subplot(2,2,1);
    stem(xi,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]->',i));
    title(sprintf('x_{%d}[n] = cos(2*%s*(%0.2f)*n)',i,'\pi',freq(i)));
    grid on;
    subplot(2,2,2);
    stem(h,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('h_{%d}[n]->',ind));
    title(sprintf('Impulse response h_{%d}[n]',ind));
    grid on;
    % plotting convolution using myConv and INBUILT FUNCTION conv
    subplot(2,2,3);
    stem(y_conv,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using myConv',i,ind,freq(i)));
    grid on;
    subplot(2,2,4);
    stem(y_conv_inbuilt,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using INBUILT function conv',i,ind,freq(i)));
    grid on;
    sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 3.6 Part 3 of the code for observation 1 and 2

```
end

% Convolution for x and h4

h = h4; % temporary variable to store the different impulse responses
ind = 4; % index for correct printing of 'number of impulse response'
for i = 1:length(freq)
    xi = cos(2*pi*freq(i)*n); % initializing x[n] for different
 frequencies

    M = myLinConvMat(h,length(xi)); % storing convolution matrix in M
    y_conv = myConv(xi,M); % convolution of x[n] and h[n] using myConv
 function
    y_conv_inbuilt = conv(xi,h); % convolution of x[n] and h[n] using
 INBUILT function conv

    % plotting x[n], h[n]
    figure;
    subplot(2,2,1);
    stem(xi,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]->',i));
    title(sprintf('x_{%d}[n] = cos(2*%s*(%0.2f)*n)',i,'\pi',freq(i)));
    grid on;
    subplot(2,2,2);
    stem(h,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('h_{%d}[n]->',ind));
    title(sprintf('Impulse response h_{%d}[n]',ind));
    grid on;
    % plotting convolution using myConv and INBUILT FUNCTION conv
    subplot(2,2,3);
    stem(y_conv,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using myConv',i,ind,freq(i)));
    grid on;
    subplot(2,2,4);
    stem(y_conv_inbuilt,'Linewidth',1.3);
    xlabel('samples(n)->');
    ylabel(sprintf('x_{%d}[n]*h_{%d}[n]->',i,ind));
    title(sprintf('convolution of x_{%d}[n] and h_{%d}[n] for
 frequency = %0.2f Hz using INBUILT function conv',i,ind,freq(i)));
    grid on;
    sgtitle('19ucc023 - Mohit Akhouri');
end
```

Published with MATLAB® R2020b

Figure 3.7 Part 4 of the code for observation 1 and 2

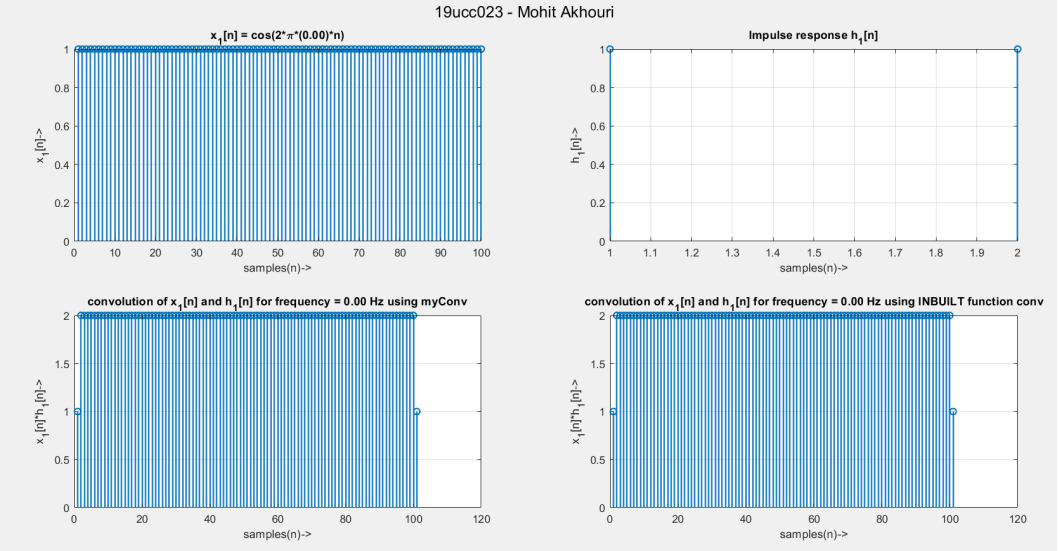

Figure 3.8 Plot of the convolution between $x_1[n]$ and $h_1[n]$

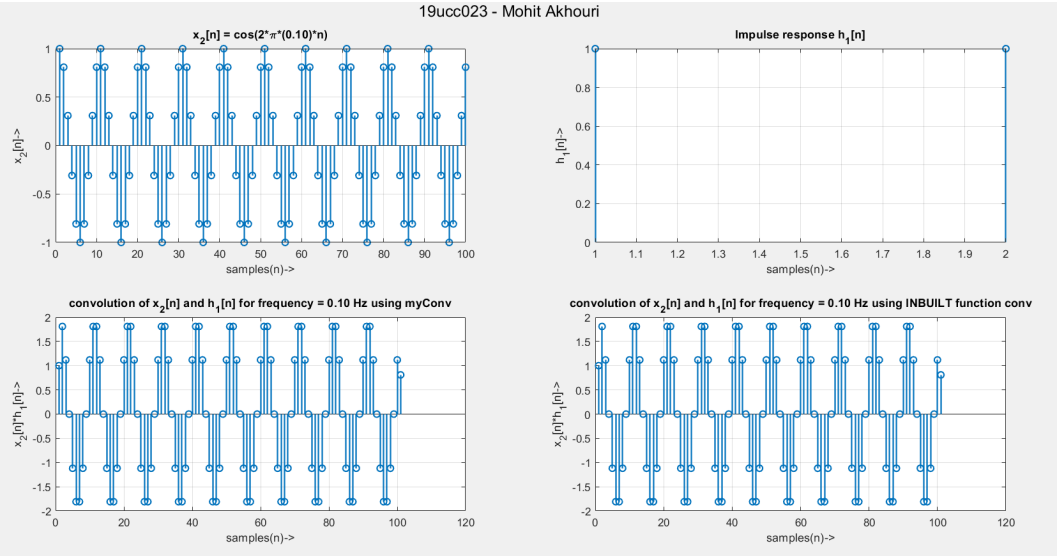

Figure 3.9 Plot of the convolution between $x_2[n]$ and $h_1[n]$

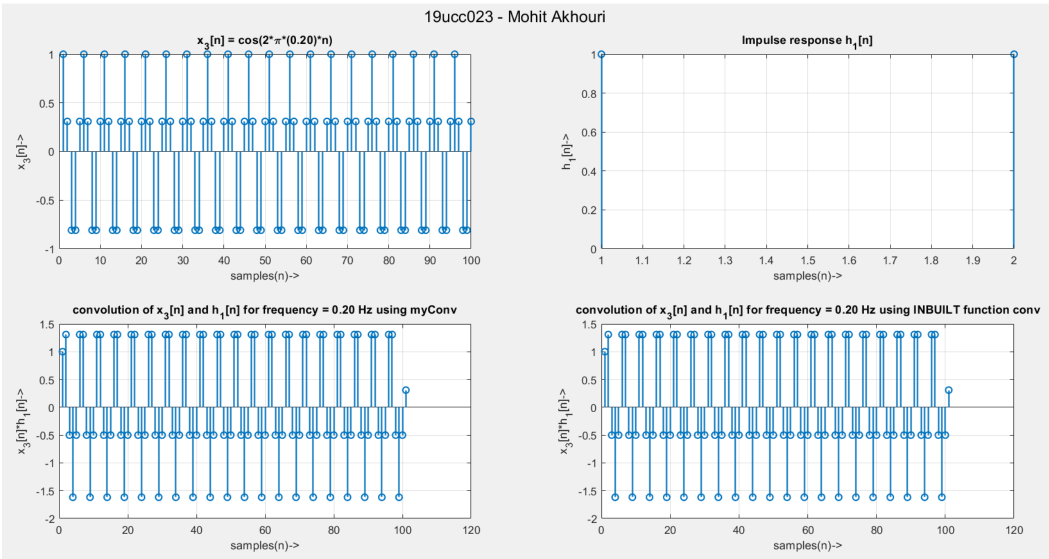

Figure 3.10 Plot of the convolution between $x_3[n]$ and $h_1[n]$

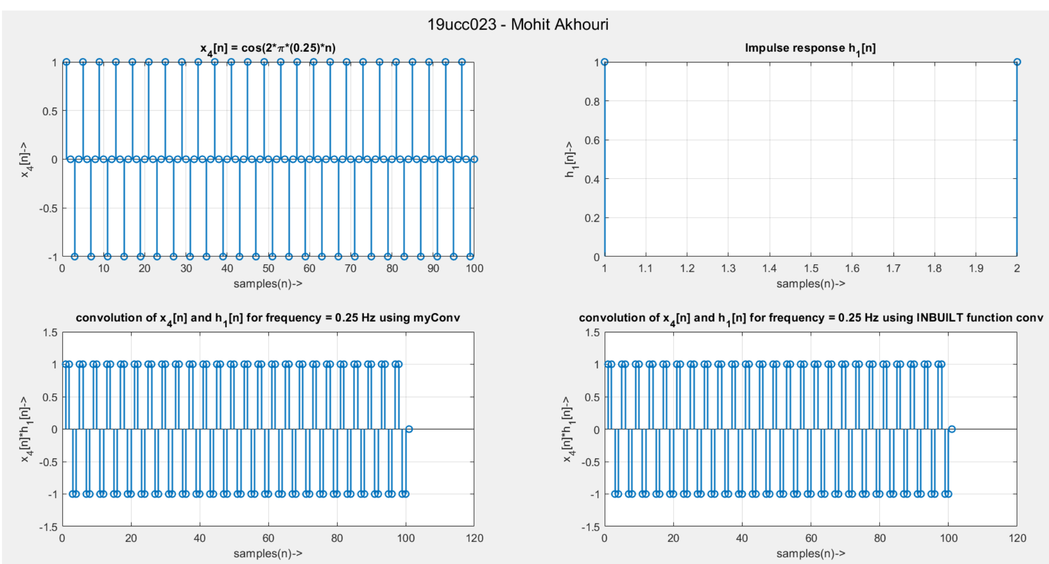

Figure 3.11 Plot of the convolution between $x_4[n]$ and $h_1[n]$

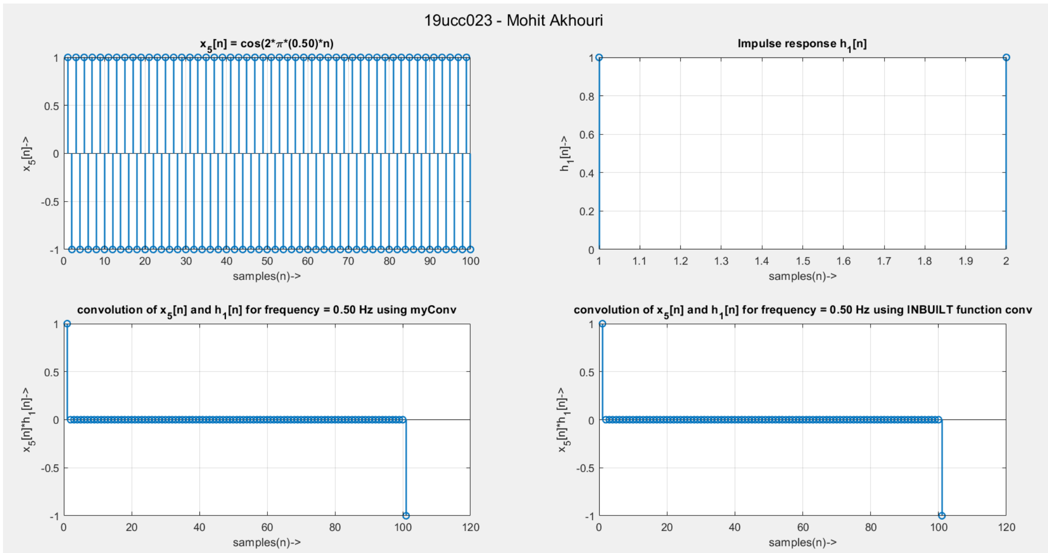

Figure 3.12 Plot of the convolution between $x_5[n]$ and $h_1[n]$

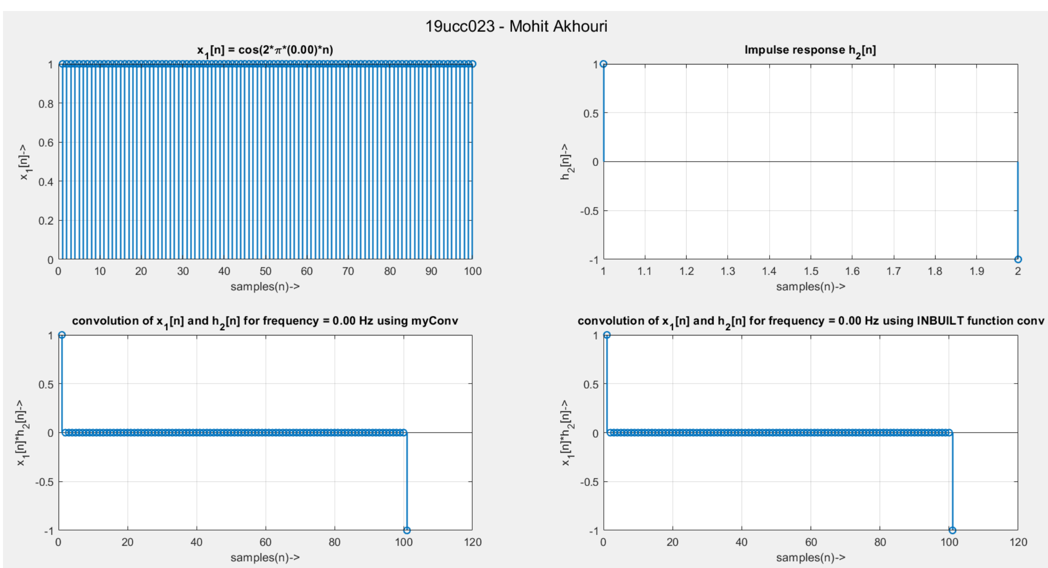

Figure 3.13 Plot of the convolution between $x_1[n]$ and $h_2[n]$

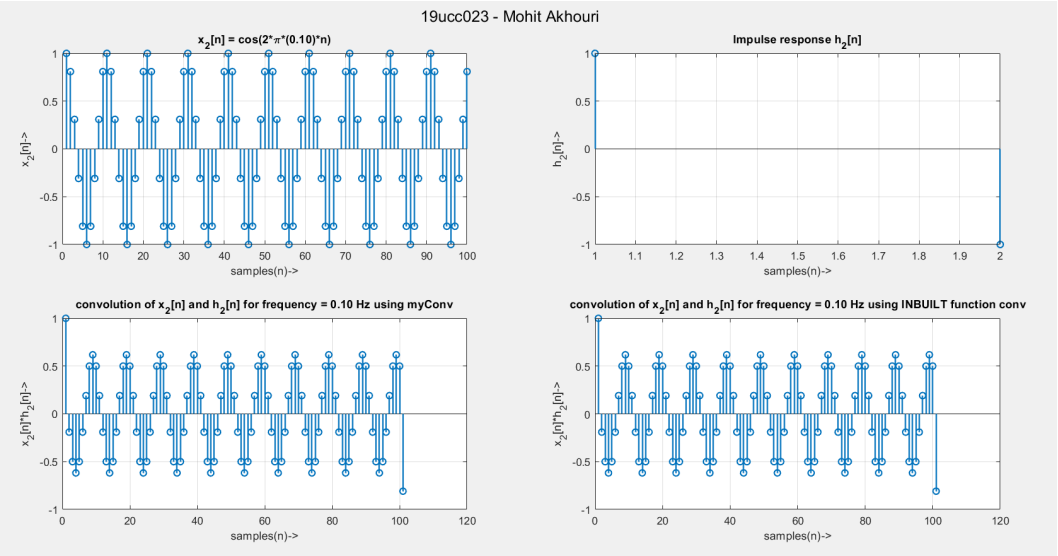

Figure 3.14 Plot of the convolution between $x_2[n]$ and $h_2[n]$

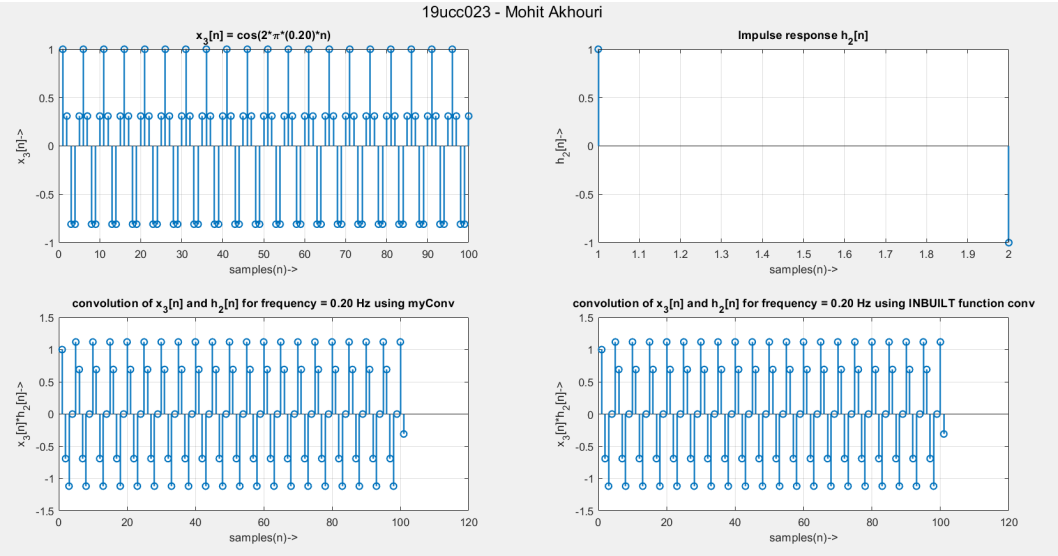

Figure 3.15 Plot of the convolution between $x_3[n]$ and $h_2[n]$

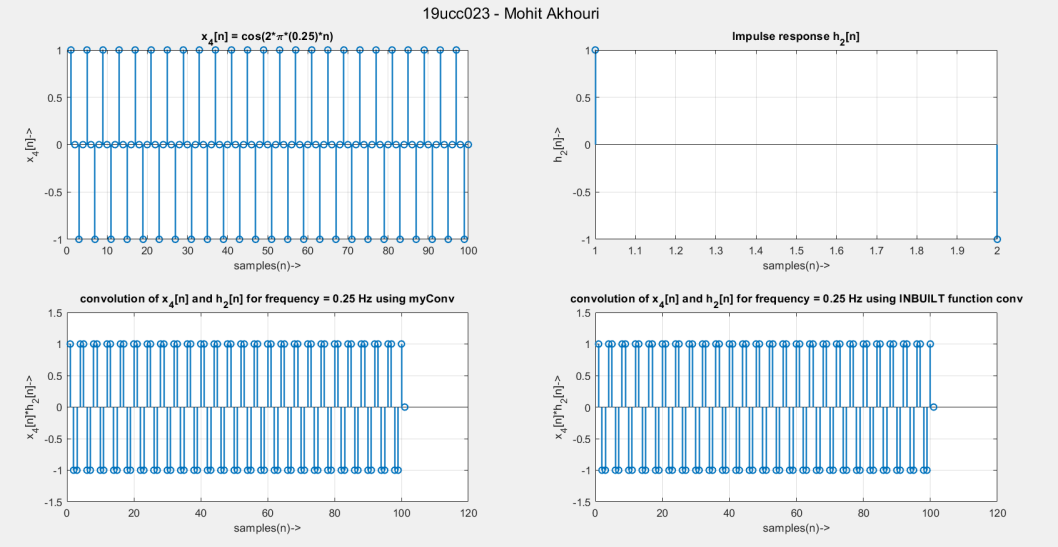

Figure 3.16 Plot of the convolution between $x_4[n]$ and $h_2[n]$

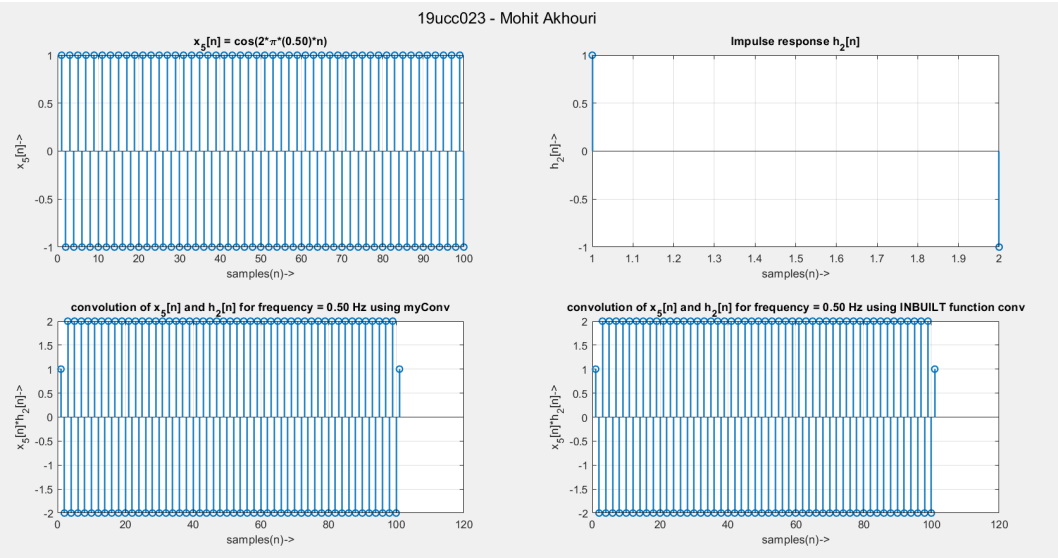

Figure 3.17 Plot of the convolution between $x_5[n]$ and $h_2[n]$

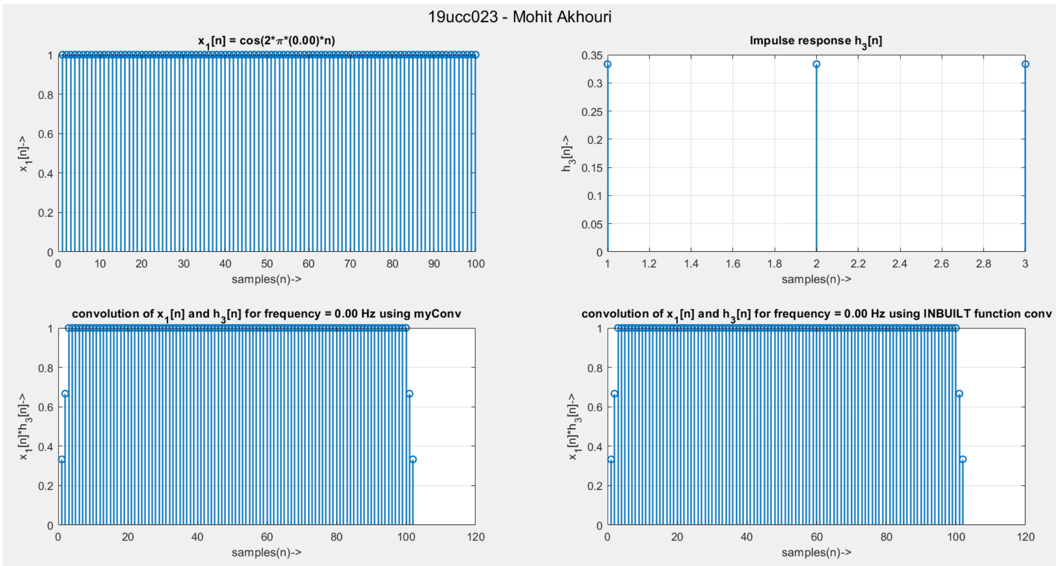

Figure 3.18 Plot of the convolution between $x_1[n]$ and $h_3[n]$

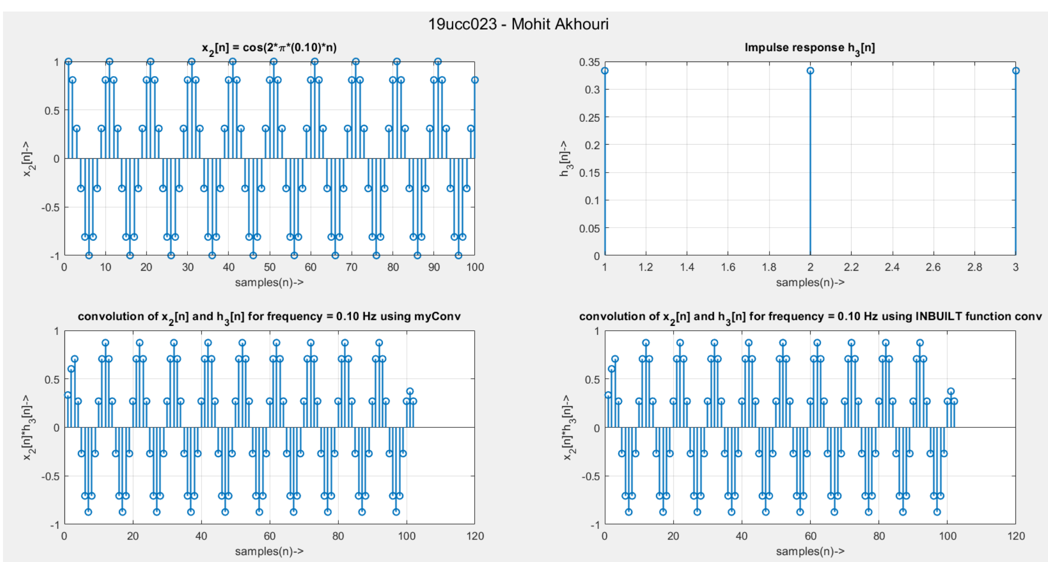

Figure 3.19 Plot of the convolution between $x_2[n]$ and $h_3[n]$

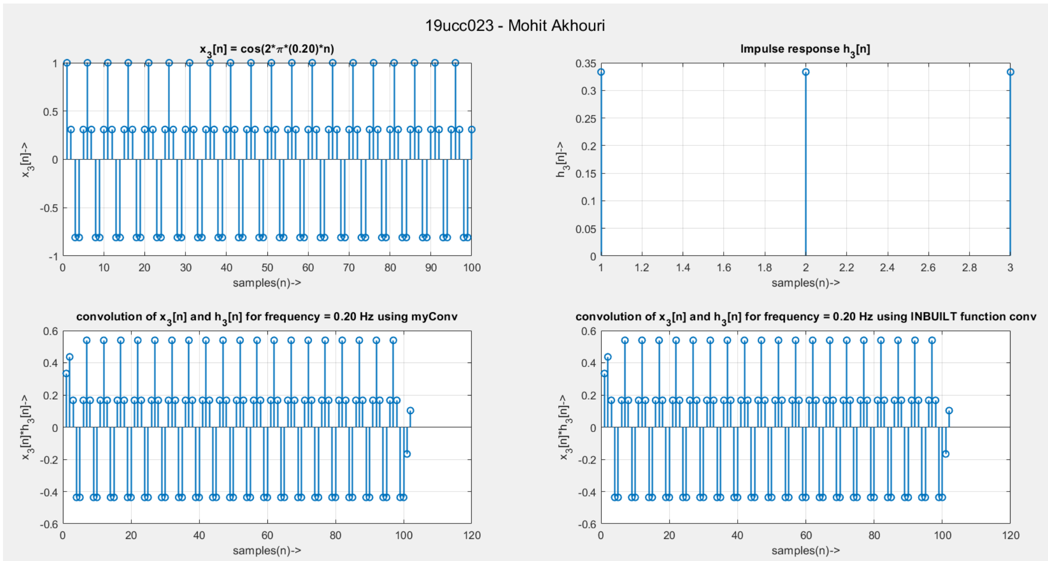

Figure 3.20 Plot of the convolution between $x_3[n]$ and $h_3[n]$

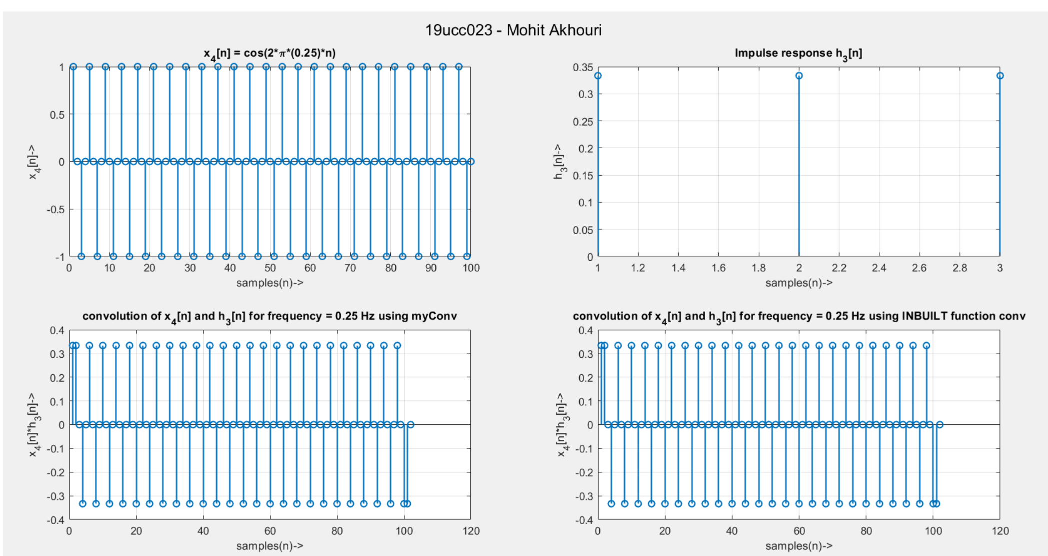

Figure 3.21 Plot of the convolution between $x_4[n]$ and $h_3[n]$

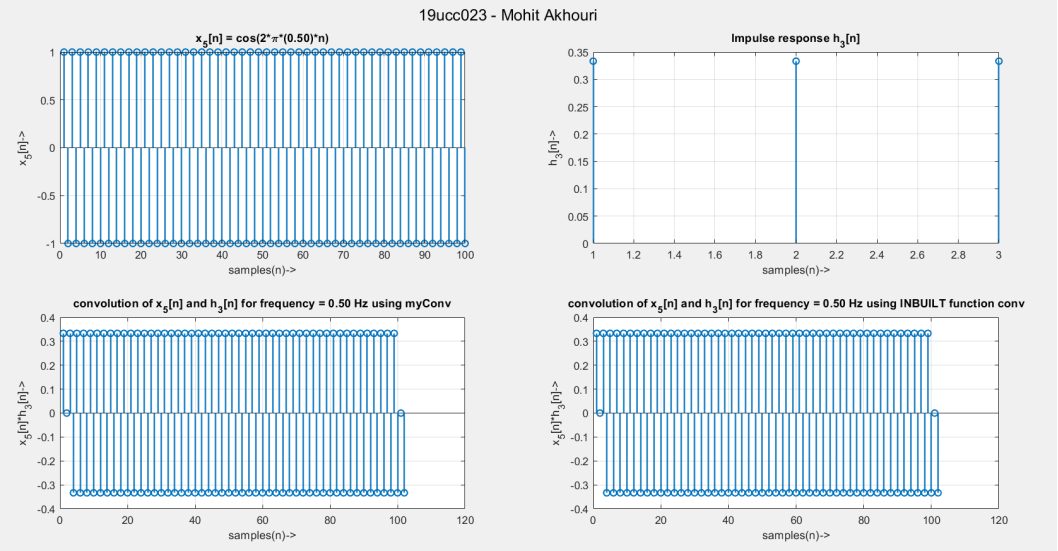

Figure 3.22 Plot of the convolution between $x_5[n]$ and $h_3[n]$

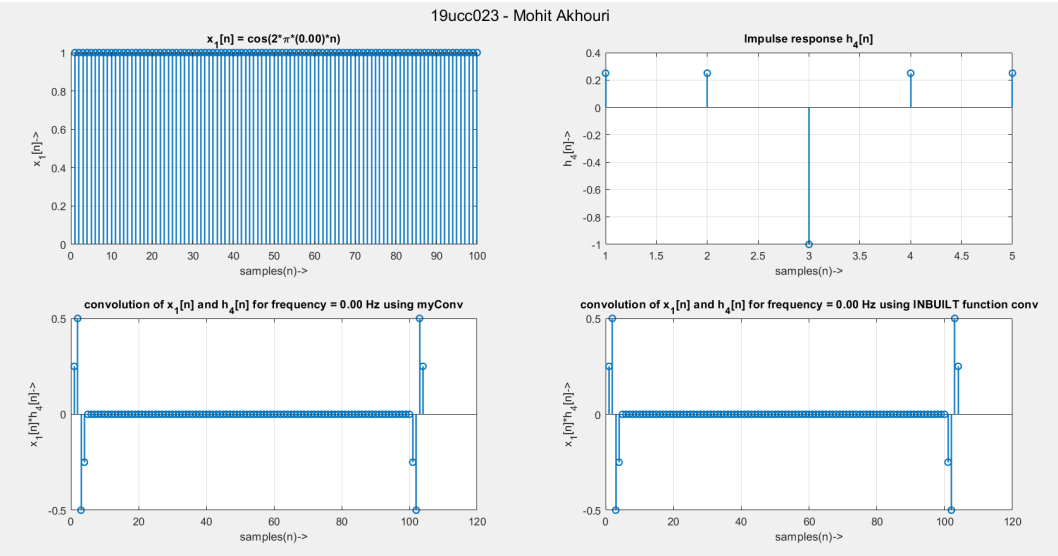

Figure 3.23 Plot of the convolution between $x_1[n]$ and $h_4[n]$

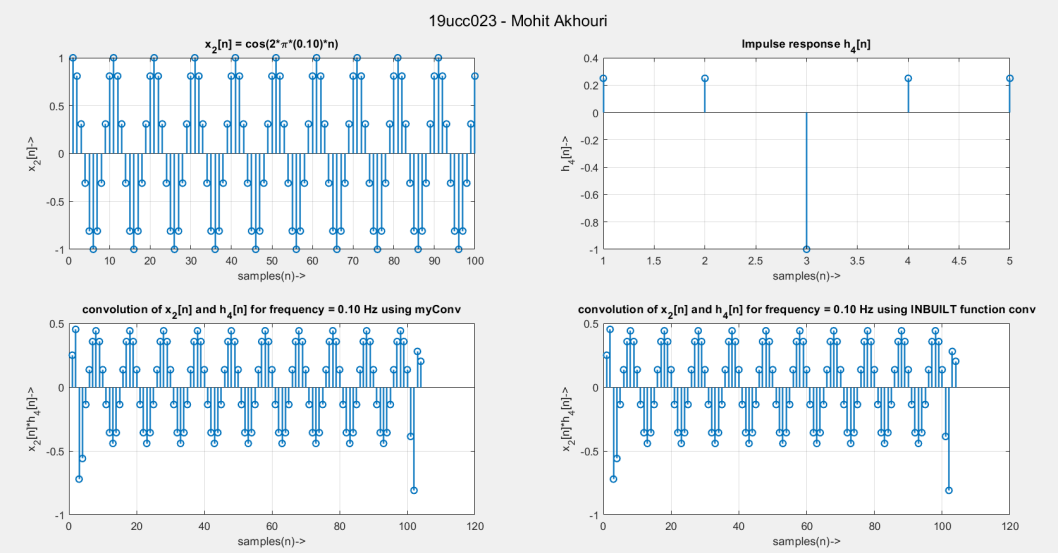

Figure 3.24 Plot of the convolution between $x_2[n]$ and $h_4[n]$

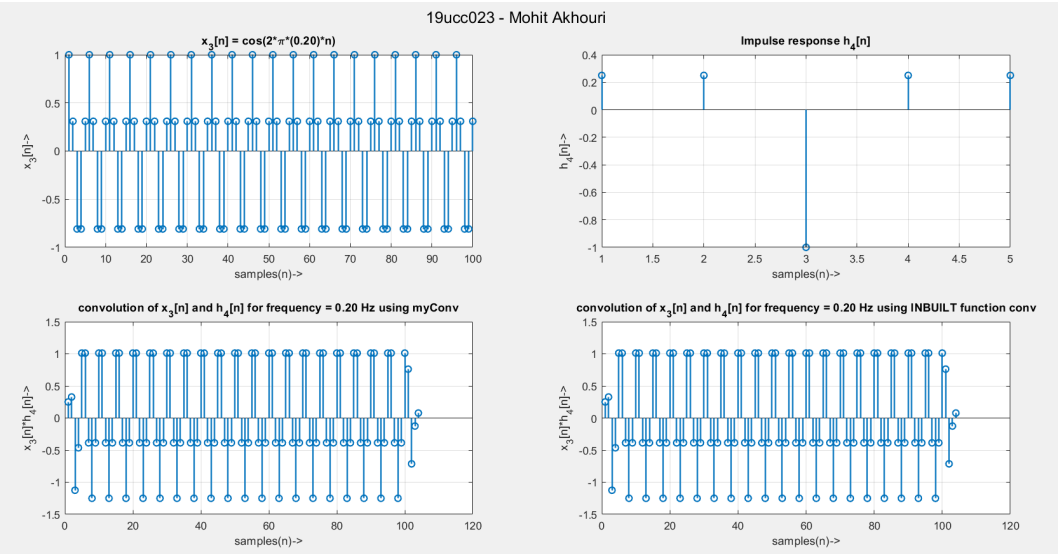

Figure 3.25 Plot of the convolution between $x_3[n]$ and $h_4[n]$

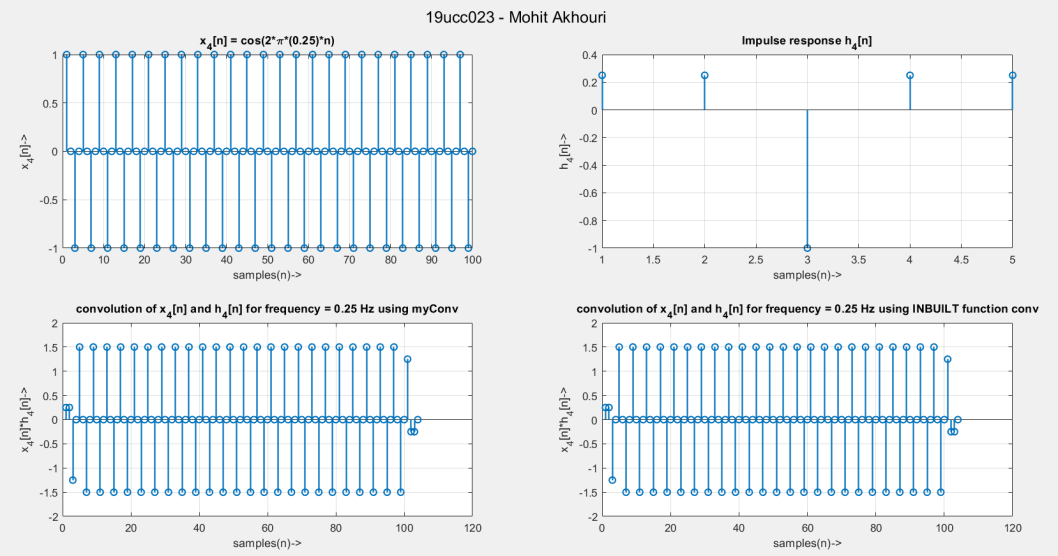

Figure 3.26 Plot of the convolution between $x_4[n]$ and $h_4[n]$

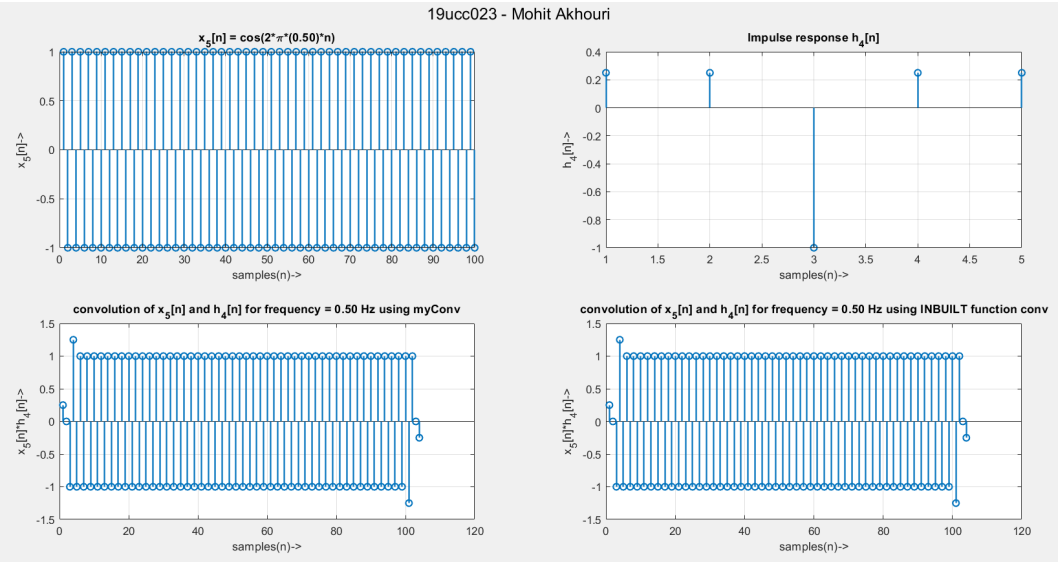

Figure 3.27 Plot of the convolution between $x_5[n]$ and $h_4[n]$

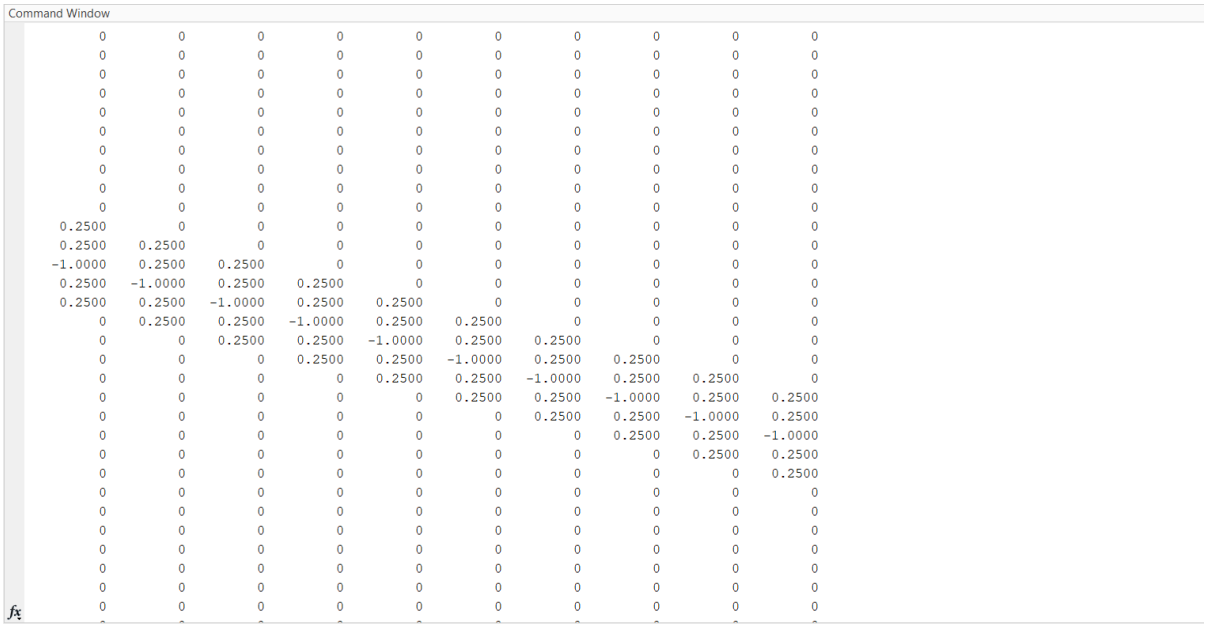

Figure 3.28 Screenshot of Convolution matrix generated by **myLinConvMat** function

3.4.2 DFT matrix generation and N-point DFT of x[n] for N=8,16,32,64:

```
% 19ucc023
% Mohit Akhouri
% Experiment 3 - Observation 3 and Observation 4

% This code will utilize the myDft function to calculate N-point DFT
% of any random sequence and compare the results with in-built fft
 function

% ALGORITHM : First we calculate the N-point DFT matrix and multiply
 the
% DFT matrix with input sequence x[n] to obtain N-point DFT

clc;
clear all;
close all;

x_8_point = rand(1,8); % A random 1 row * 8 columns array
x_16_point = rand(1,16); % A random 1 row * 16 columns array
x_32_point = rand(1,32); % A random 1 row * 32 columns array
x_64_point = rand(1,64); % A random 1 row * 64 columns array

% DFT of sequence x_8_point
dft_8_user_defined = myDft(x_8_point,8); % DFT through function myDft
dft_8_inbuilt = fft(x_8_point,8); % DFT through INBUILT function fft

% DFT of sequence x_16_point
dft_16_user_defined = myDft(x_16_point,16); % DFT through function
 myDft
dft_16_inbuilt = fft(x_16_point,16); % DFT through INBUILT function
 fft

% DFT of sequence x_32_point
dft_32_user_defined = myDft(x_32_point,32); % DFT through function
 myDft
dft_32_inbuilt = fft(x_32_point,32); % DFT through INBUILT function
 fft

% DFT of sequence x_64_point
dft_64_user_defined = myDft(x_64_point,64); % DFT through function
 myDft
dft_64_inbuilt = fft(x_64_point,64); % DFT through INBUILT function
 fft

% plotting of various signal x[n] and X(w) for different N-point
figure;
subplot(3,1,1);
stem(x_8_point,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('x[n]->');
title('RANDOM x[n] sequence for 8-point DFT');
grid on;
subplot(3,1,2);
```

Figure 3.29 Part 1 of the code for observation 3 and 4

```
stem(dft_8_user_defined,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using USER-DEFINED
 myDft function');
grid on;
subplot(3,1,3);
stem(dft_8_inbuilt,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using INBUILT fft
 function');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

figure;
subplot(3,1,1);
stem(x_16_point,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('x[n]->');
title('RANDOM x[n] sequence for 16-point DFT');
grid on;
subplot(3,1,2);
stem(dft_16_user_defined,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using USER-DEFINED
 myDft function');
grid on;
subplot(3,1,3);
stem(dft_16_inbuilt,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using INBUILT fft
 function');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

figure;
subplot(3,1,1);
stem(x_32_point,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('x[n]->');
title('RANDOM x[n] sequence for 32-point DFT');
grid on;
subplot(3,1,2);
stem(dft_32_user_defined,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using USER-DEFINED
 myDft function');
grid on;
subplot(3,1,3);
stem(dft_32_inbuilt,'Linewidth',1.5);
```

Figure 3.30 Part 2 of the code for observation 3 and 4

```
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using INBUILT fft
 function');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

figure;
subplot(3,1,1);
stem(x_64_point,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('x[n]->');
title('RANDOM x[n] sequence for 64-point DFT');
grid on;
subplot(3,1,2);
stem(dft_64_user_defined,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using USER-DEFINED
 myDft function');
grid on;
subplot(3,1,3);
stem(dft_64_inbuilt,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('X[\omega]->');
title('Discrete fourier transform (DFT) of x[n] using INBUILT fft
 function');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 3.31 Part 3 of the code for observation 3 and 4

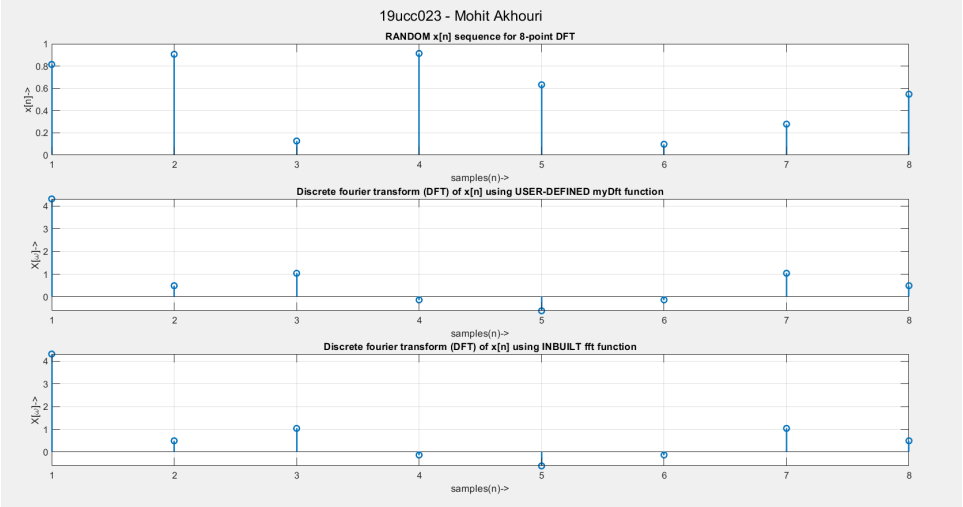

Figure 3.32 Plot of the 8-point DFT of random sequence x[n]

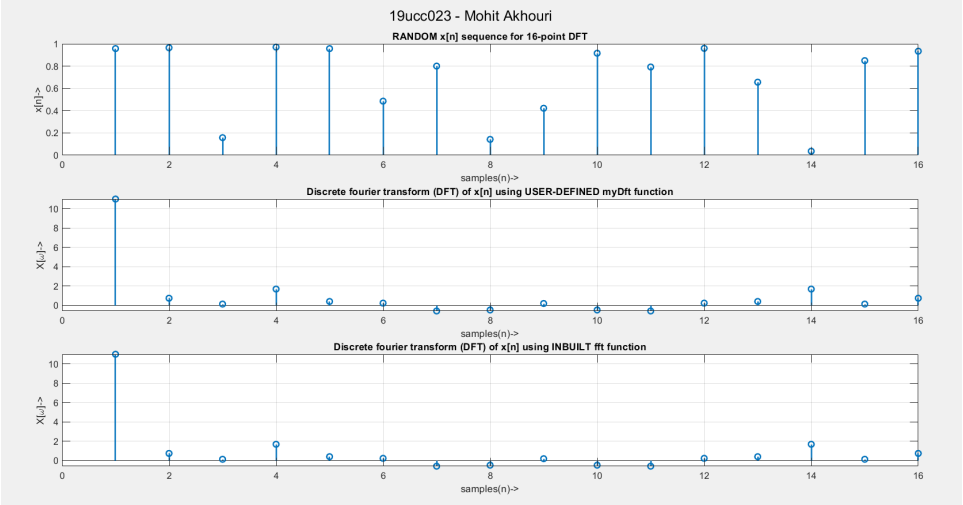

Figure 3.33 Plot of the 16-point DFT of random sequence x[n]

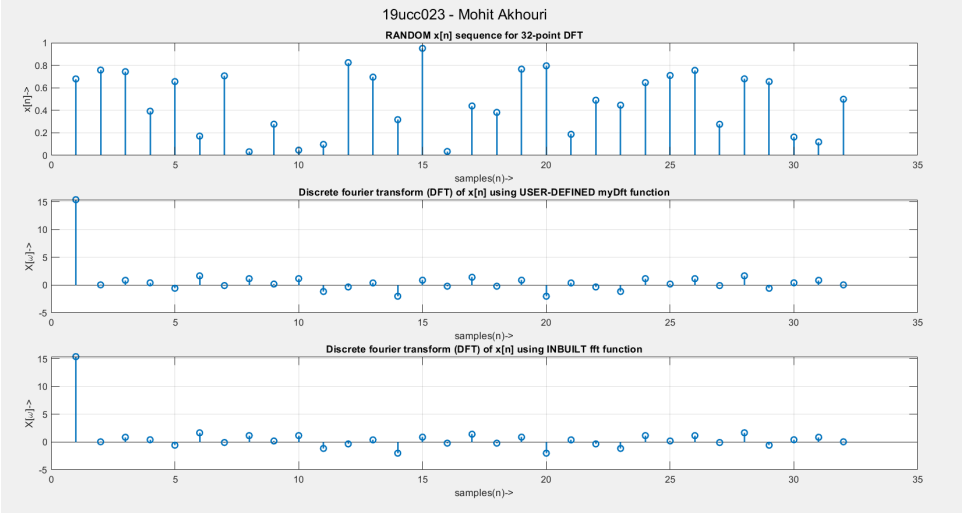

Figure 3.34 Plot of the 32-point DFT of random sequence x[n]

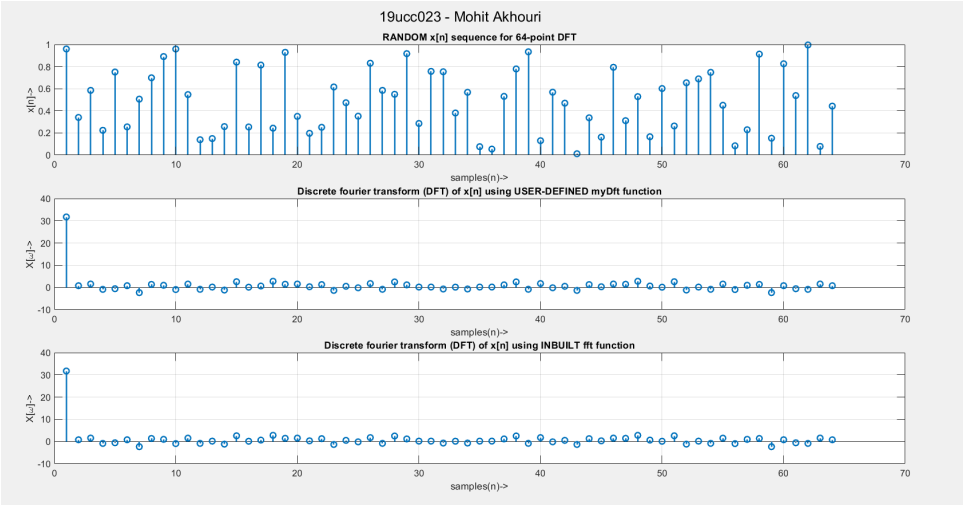

Figure 3.35 Plot of the 64-point DFT of random sequence x[n]

```
The DFT matrix is given as :
  Columns 1 through 5

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.7071 - 0.7071i   0.0000 - 1.0000i  -0.7071 - 0.7071i  -1.0000 - 0.0000i
   1.0000 + 0.0000i   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i  -0.7071 - 0.7071i  -0.0000 + 1.0000i   0.7071 - 0.7071i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i  -0.7071 + 0.7071i   0.0000 - 1.0000i   0.7071 + 0.7071i  -1.0000 - 0.0000i
   1.0000 + 0.0000i  -0.0000 + 1.0000i  -1.0000 - 0.0000i   0.0000 - 1.0000i   1.0000 + 0.0000i
   1.0000 + 0.0000i   0.7071 + 0.7071i  -0.0000 + 1.0000i  -0.7071 + 0.7071i  -1.0000 - 0.0000i

  Columns 6 through 8

   1.0000 + 0.0000i   1.0000 + 0.0000i   1.0000 + 0.0000i
  -0.7071 + 0.7071i  -0.0000 + 1.0000i   0.7071 + 0.7071i
   0.0000 - 1.0000i  -1.0000 - 0.0000i  -0.0000 + 1.0000i
   0.7071 + 0.7071i   0.0000 - 1.0000i  -0.7071 + 0.7071i
  -1.0000 - 0.0000i   1.0000 + 0.0000i  -1.0000 - 0.0000i
   0.7071 - 0.7071i  -0.0000 + 1.0000i  -0.7071 - 0.7071i
  -0.0000 + 1.0000i  -1.0000 - 0.0000i  -0.0000 - 1.0000i
  -0.7071 - 0.7071i  -0.0000 - 1.0000i   0.7071 - 0.7071i
```

Figure 3.36 Screenshot of 8-point DFT matrix generated by the function **myDft**

3.4.3 Convolution in Simulink :

```
% 19ucc023
% Mohit Akhouri
% Experiment 3 - Observation 5

% This code will deploy the simulink model to calculate convolution of
 two
% random sequences x[n] and h[n]

sim('Simulink_Observation_5'); % calling the simulink model
y_conv_inbuilt = conv(out.xn.data,out.hn.data); % Convolution from
 INBUILT function for cross-checking

% plotting of signals x[n],h[n] and convolved signal y[n]

figure;
subplot(2,2,1);
stem(out.xn.data,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('x[n]->');
title('random input signal x[n] from Simulink Model');
grid on;
subplot(2,2,2);
stem(out.hn.data,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('h[n]->');
title('random impulse response h[n] from Simulink Model');
grid on;
subplot(2,2,3);
stem(out.yn.data,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('y[n] = x[n] * h[n]');
title('Convolved signal y[n] obtained from Simulink Model');
grid on;
subplot(2,2,4);
stem(y_conv_inbuilt,'Linewidth',1.5);
xlabel('samples(n)->');
ylabel('y[n] = x[n] * h[n]');
title('Convolved signal y[n] obtained from INBUILT function conv');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 3.37 Code for the observation 5

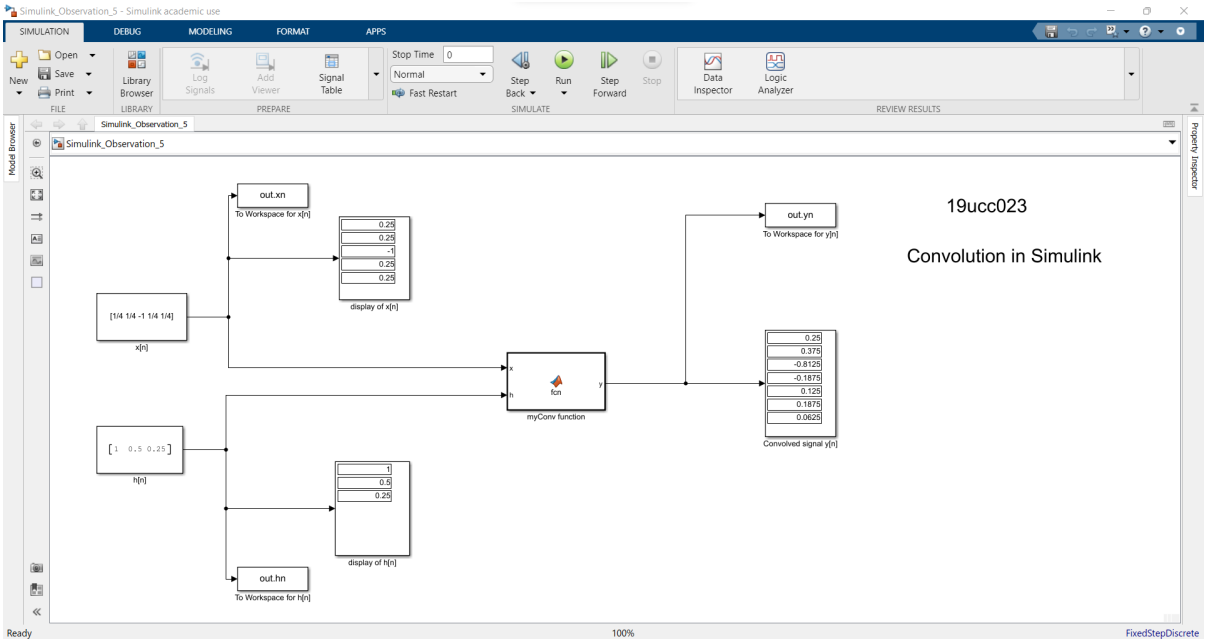

Figure 3.38 Simulink Model used for Convolution

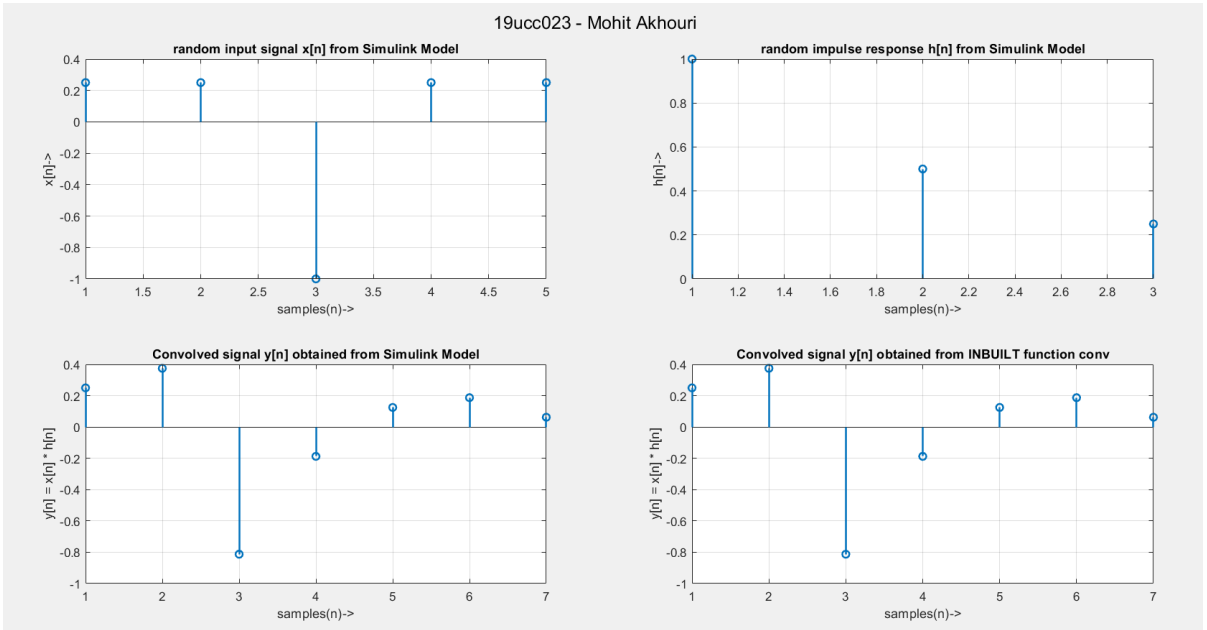

Figure 3.39 Plot of the convolution obtained between x[n] and h[n] from Simulink Model

3.4.4 Functions used in main codes for Convolution and DFT :

3.4.4.1 myLinConvMat.m function code :

```
function [M] = myLinConvMat(h,n)
% This function computes the convolution matrix M for impulse response
 h(n)
% with 'n' being the length of the input sequence 'x'

% 19ucc023
% Mohit Akhouri

rows = n; % initializing the number of rows of Convolution matrix M
length_h = length(h); % declaring the length of 'h'
columns = length(h) + n - 1; % declaring the number of columns of
 matrix M
M = zeros(rows,columns); % initializing the convolution matrix with
 zeros

% generating the convolution matrix (ALGORITHM)

for i=1:rows
    offset = i; % offset in the matrix from where 'h' needs to be
 stored
    for j=1:length_h
        M(i,offset) = h(j);
        offset = offset + 1; % offset incremented
    end
end
disp(M);
end
```

Published with MATLAB® R2020b

Figure 3.40 myLinConvMat function for computation of Convolution matrix

3.4.4.2 myConv function code :

```
function [y] = myConv(x,M)
% This function computes the convolution of 'x' and 'h'
% using convolution matrix M where x = input sequence

% 19ucc023
% Mohit Akhouri

rows = size(M,1); % storing number of rows of matrix M
columns = size(M,2); % storing number of columns of matrix M

y = zeros(1,columns); % initializing the variable to store convolution
 of sequences
sum = 0; % temporary variable to store sum

% ALGORITHM for calculation of convolution of sequences is as follows
for i=1:columns
    sum = 0;
    for j=1:rows
       sum = sum + (M(j,i)*x(j));
    end
    y(i) = sum;
end
end
```

Published with MATLAB® R2020b

Figure 3.41 myConv function for computation of Convolution from Convolution matrix M

3.4.4.3 myDft function code :

```
function [X] = myDft(x,N)
% This function will calculate the discrete fourier transform
% of input sequence x[n] , this function calculates N-point DFT

% 19ucc023
% Mohit Akhouri

% calculating the DFT matrix 'D'
D = zeros(N,N); % DFT matrix to store the values of twiddle factor
twd_factor = 0; % to store the value of twiddle factor

for n=1:N
    for k=1:N
        twd_factor = exp(-1j*2*pi*(k-1)*(n-1)/N);
        D(n,k) = twd_factor;
    end
end

disp('The DFT matrix is given as :');
disp(D);

% The ALGORITHM for calculation of DFT is as follows
X = zeros(1,N);
for i=1:N
    sum = 0;
    for j=1:N
        sum = sum+(D(i,j)*x(j));
    end
    X(i)=sum;
end

end
```

Published with MATLAB® R2020b

Figure 3.42 myDft function for computation of DFT matrix and N-point DFT

3.5 Conclusion

In this experiment , we learnt the concepts of **Linear Convolution** and **Discrete Fourier Transform (DFT)** of Digital Signal Processing. We learnt about convolution matrix and DFT matrix and how to utilize them to compute convolution and N-point DFT. We also compared the results with inbuilt functions like **fft** and **conv**. We also learnt about **Twiddle factor** utilization in the construction of the DFT matrix. We also performed the convolution in Simulink through various blocks and compared the results obtained from the Simulink model with results obtained from MATLAB code.