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 *4*

Experiment - 4

4.1 Aim of the Experiment

- Circular Convolution and DFT Multiplication for two sequences
- Simulink based circular convolution

4.2 Software Used

- MATLAB
- Simulink

4.3 Theory

4.3.1 About Circular Convolution :

Circular convolution, also known as **cyclic convolution**, is a special case of **periodic convolution**, which is the convolution of two periodic functions that have the same period. Periodic convolution arises, for example, in the context of the discrete-time Fourier transform (DTFT). In particular, the DTFT of the product of two discrete sequences is the periodic convolution of the DTFTs of the individual sequences. And each DTFT is a periodic summation of a continuous Fourier transform function. Although DTFTs are usually continuous functions of frequency, the concepts of periodic and circular convolution are also directly applicable to discrete sequences of data. In that context, **circular convolution** plays an **important role** in **maximizing the efficiency of a certain kind of common filtering operation**.

Circular convolution can also be obtained through **change of basis**. In the first method, by multiplying **DFT of two sequences** we obtain their circular convolution if we take **Inverse DFT** of the product. In the second method , We can obtain Circular Convolution by product of the **Convolution matrix** , **DFT matrix** and **inverse DFT matrix**.

The circular convolution for a system with **system response** h[n] and **input sequence** x[n] (both are padded with zeros to make them equal in size) is as follows :

$$y(n) = \sum_{i=0}^{N-1} h(i).x((n-i))_N \tag{4.1}$$

In the above equation , **N** is the **larger sequence length** and it is taken as a multiple of 2.N = 2^i where i can be 2,3,4 etc. In the equation the **term** $x((n-i))_N$ represents the **index** $N - i + n^{'}$.

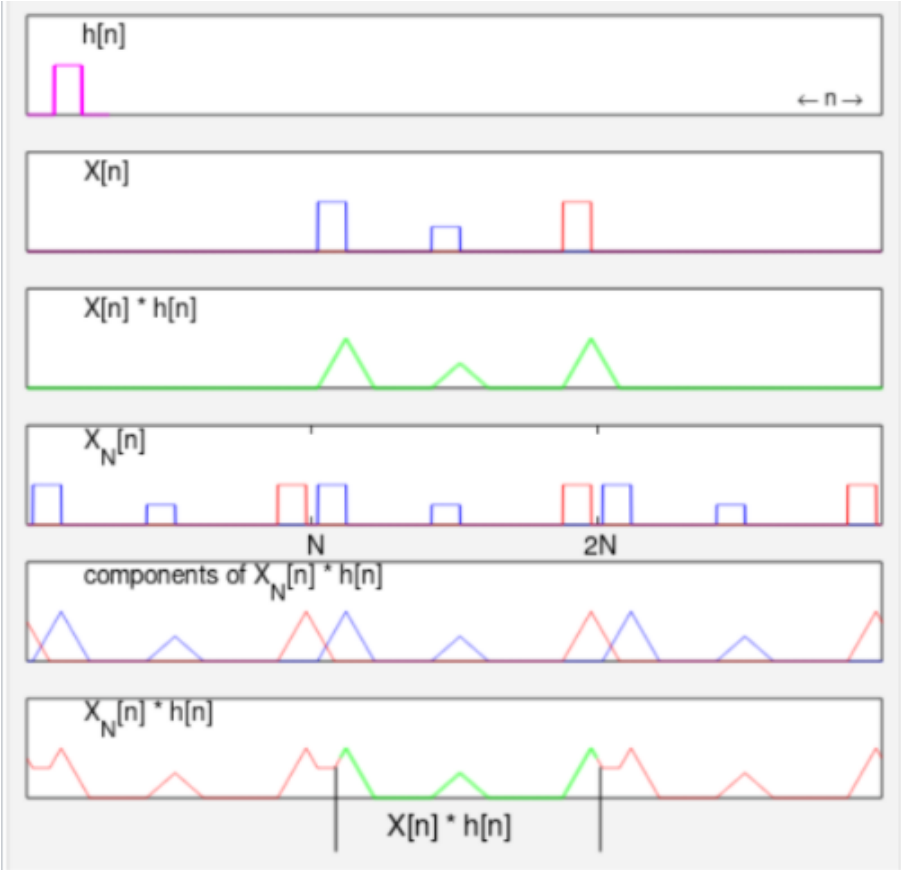

Figure 4.1 Example of Circular Convolution

4.3.1.1 Change of Basis

Circular Convolution can also be obtained for two sequences x[n] and h[n] by :

- Taking Inverse DFT of product of DFT of x[n] and h[n]
- Through Matrix Multiplication method

4.4 Code and results

4.4.1 Circular Convolution of two pair of sequences using Circular Convolution matrix :

```
% 19ucc023
% Mohit Akhouri
% Experiment 4 - Observation 1 and Observation 2

% This code will calculate circular convolution between two sets
% of x[n] and h[n] and compare them with inbuilt command
% for circular convolution 'cconv(x,y)'

% ALGORITHM : call the my_Circular_Convolution function

clc;
clear all;
close all;

h = [2 1 2 1]; % initializing h1
x1 = [1 2 3 4]; % initializing x1 (first input sequence)

% initializing second input sequence x2
n = 0:1:9; % initializing samples for x2[n]
x2 = zeros(1,length(n)); % initializing x2[n] with zeros

for i=1:length(n)
    x2(i) = (0.5)^(n(i)); % calculating the sample values of x2[n]
end

length_h = length(h); % length of impulse response h[n]
length_x1 = length(x1); % length of input sequence x1[n]
length_x2 = length(x2); % length of input sequence x2[n]

% calling my_Circular_Convolution to calculate circular convolution
% of x1[n] and h[n] , x2[n] and h[n] and compare the results obtained
% with inbuilt command for circular convolution 'cconv(x,y,n)'

% circular convolution from USER_DEFINED function
 my_Circular_Convolution
circ_conv_x1h = my_Circular_Convolution(x1,h);
circ_conv_x2h = my_Circular_Convolution(x2,h);

% circular convolution from INBUILT function cconv(x,y,n)
circ_conv_x1h_inbuilt = cconv(x1,h,max(length_x1,length_h));
circ_conv_x2h_inbuilt = cconv(x2,h,max(length_x2,length_h));

% plotting the results obtained for circular convolution

% circular convolution between x1[n] and h[n]
figure;
subplot(2,2,1);
stem(x1,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x_{1}[n] ->');
title('plot of input sequence x_{1}[n]');
grid on;
```

Figure 4.2 Part 1 of the code for observation 1 and 2

```
subplot(2,2,2);
stem(h,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('h[n] ->');
title('plot of impulse response h[n]');
grid on;
subplot(2,2,3);
stem(circ_conv_x1h,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel(' x_{1}[n] \otimes h[n]->');
title('Circular convolution of x_{1}[n] and h[n] using my\_Circular
\_Convolution');
grid on;
subplot(2,2,4);
stem(circ_conv_x1h_inbuilt,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel(' x_{1}[n] \otimes h[n]->');
title('Circular convolution of x_{1}[n] and h[n] using INBUILT
 function cconv');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% circular convolution between x2[n] and h[n]
figure;
subplot(2,2,1);
stem(x2,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x_{2}[n] ->');
title('plot of input sequence x_{2}[n]');
grid on;
subplot(2,2,2);
stem(h,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('h[n] ->');
title('plot of impulse response h[n]');
grid on;
subplot(2,2,3);
stem(circ_conv_x2h,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel(' x_{2}[n] \otimes h[n]->');
title('Circular convolution of x_{2}[n] and h[n] using my\_Circular
\_Convolution');
grid on;
subplot(2,2,4);
stem(circ_conv_x2h_inbuilt,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel(' x_{2}[n] \otimes h[n]->');
title('Circular convolution of x_{2}[n] and h[n] using INBUILT
 function cconv');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 4.3 Part 2 of the code for observation 1 and 2

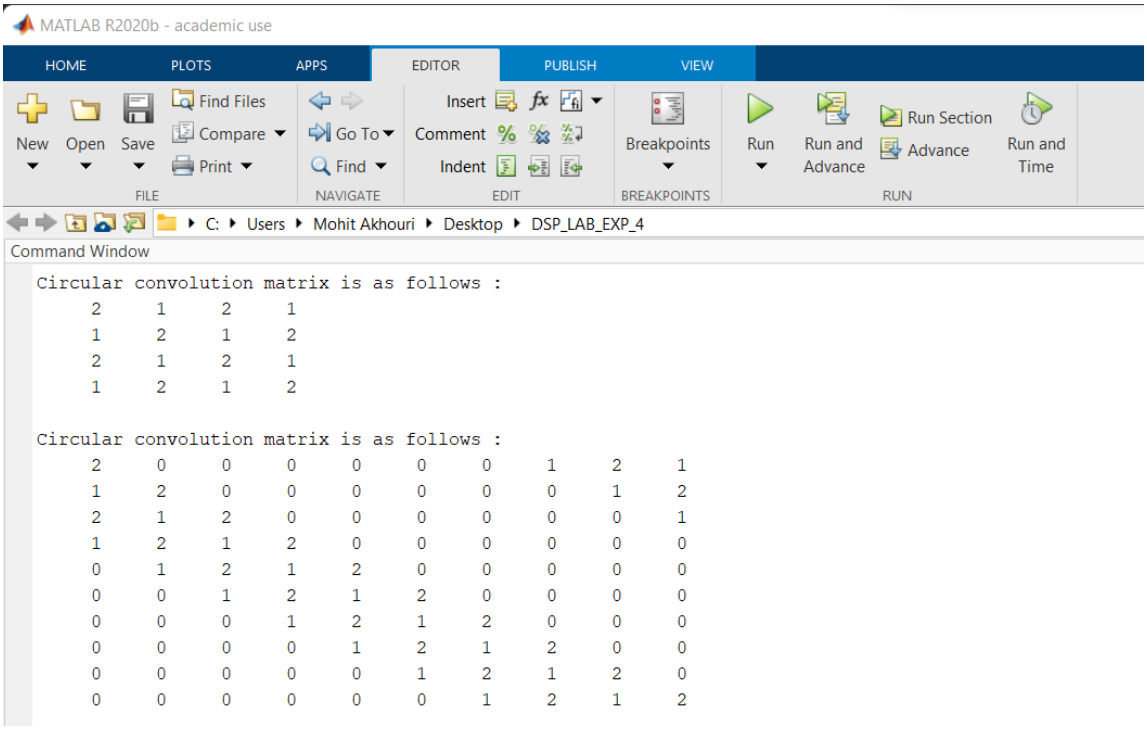

Figure 4.4 Displaying Circular Convolution matrices of **h[n]** for $x_1[n]$ and $x_2[n]$

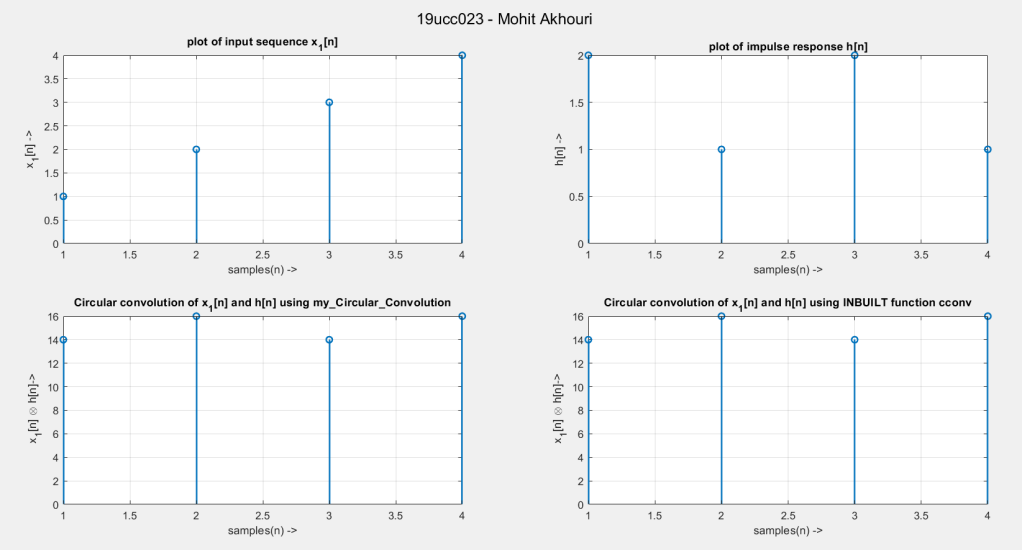

Figure 4.5 Plot of Circular Convolution between $x_1[n]$ and h[n]

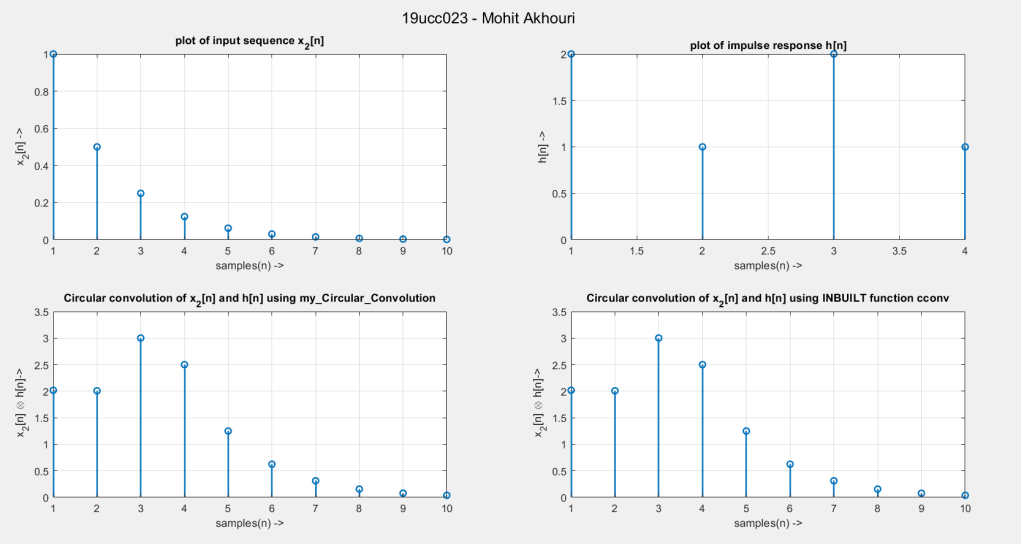

Figure 4.6 Plot of Circular Convolution between $x_2[n]$ and h[n]

4.4.2 Change of Basis :

4.4.2.1 Verify Circular Convolution $\Longleftrightarrow$ DFT multiplication of Fourier Transform Pair :

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

% This code will verify the property that :
% if x[n] and h[n] are sequences then CIRCULAR CONVOLUTION
% of x[n] and h[n] is equal to IDFT(X(k)*H(k))

clc;
clear all;
close all;

% ALGORITHM : Initialize x[n] and h[n]
% First calculate DFT of x[n] and h[n] , let it be X(k) and H(k)
% Now , multiply X(k) and H(k) point by point and finally pass them
% thtough myIDFT function to calculate IDFT. IDFT obtained can now be
% compared with the circular convolution of x[n] and h[n]

% randperm(N,K) : returns a random permutation of K integers from 1 to
 N

x = randperm(8,8); % taking a random sequence x[n]
h = randperm(8,8); % taking a random sequence h[n]

% calculating the circular convolution of x[n] and h[n]

circ_conv_xh = my_Circular_Convolution(x,h);

% Finding the 8-point DFT sequence of x[n] and h[n]

[X,Dx] = myDFT(x,8); % DFT of input sequence x[n] , Dx is DFT matrix
 of x[n]
[H,Dh] = myDFT(h,8); % DFT of impulse response h[n] , Dh is DFT matrix
 of h[n]

% Multiplying X(k) and H(k) sample by sample ( point-by-point )

Y = X .* H; % point-by-point multiplication of X(k) and H(k)

% Finding the IDFT of Y(k)

y = myIDFT(Y,8); % IDFT of Y(k) to be compared with circ_conv_xh

% Plotting the results and comparing the IDFT y[n] with circular
% convolution circ_conv_xh

% plotting x[n],h[n],X(k) and H(k)
figure;
subplot(2,2,1);
stem(x,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] ->');
```

Figure 4.7 Part 1 of the code for observation 3

```
title('plot of input sequence x[n]');
grid on;
subplot(2,2,2);
stem(h,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('h[n] ->');
title('plot of impulse response h[n]');
grid on;
subplot(2,2,3);
stem(X,'Linewidth',1.5);
xlabel('samples(k) ->');
ylabel('X(k) ->');
title('DFT of input sequence x[n]');
grid on;
subplot(2,2,4);
stem(H,'Linewidth',1.5);
xlabel('samples(k) ->');
ylabel('H(k) ->');
title('DFT of impulse response h[n]');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting Y(k) = X(k) .* H(k)
figure;
stem(Y,'Linewidth',1.5);
xlabel('samples(k) ->');
ylabel('Y(k) ->');
title('19ucc023 - Mohit Akhouri','Y(k) = X(k)*H(k) : MULTIPLICATION of
 X(k) and H(k) point-by-point in frequency domain');
grid on;

% plotting y[n] = IDFT(Y(k)) and circ_conv_xh for comparison of plots
figure;
subplot(2,1,1);
stem(y,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('y[n] ->');
title('IDFT of seqeunce Y(k)');
grid on;
subplot(2,1,2);
stem(circ_conv_xh,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] \otimes h[n] ->');
title('Circular Convolution of x[n] and h[n] using my\_Circular
\_Convolution');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 4.8 Part 2 of the code for observation 3

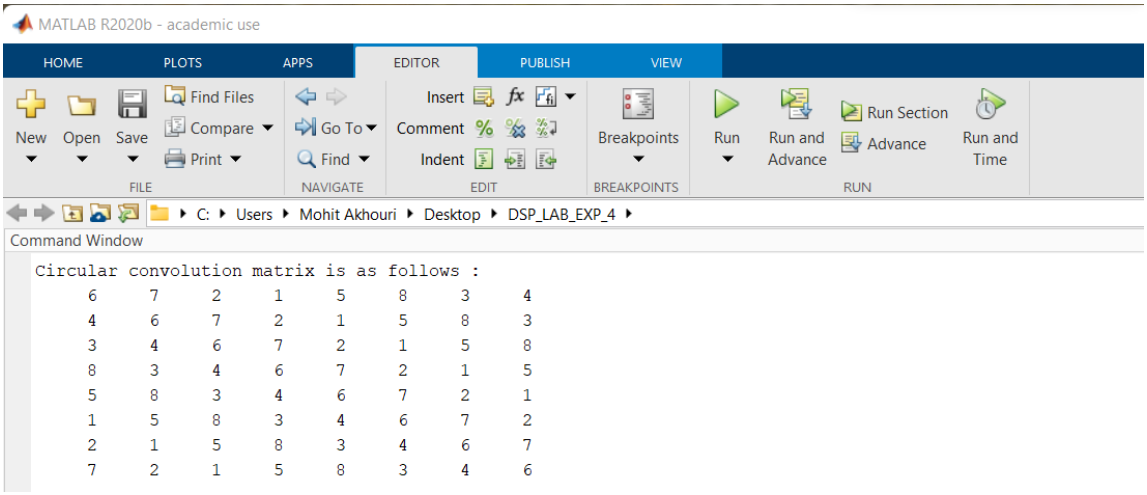

Figure 4.9 Display of Circular Convolution matrix for impulse response h[n]

```
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 4.10 Display of DFT matrix for Impulse response h[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 4.11 Display of DFT matrix for Input sequence x[n]

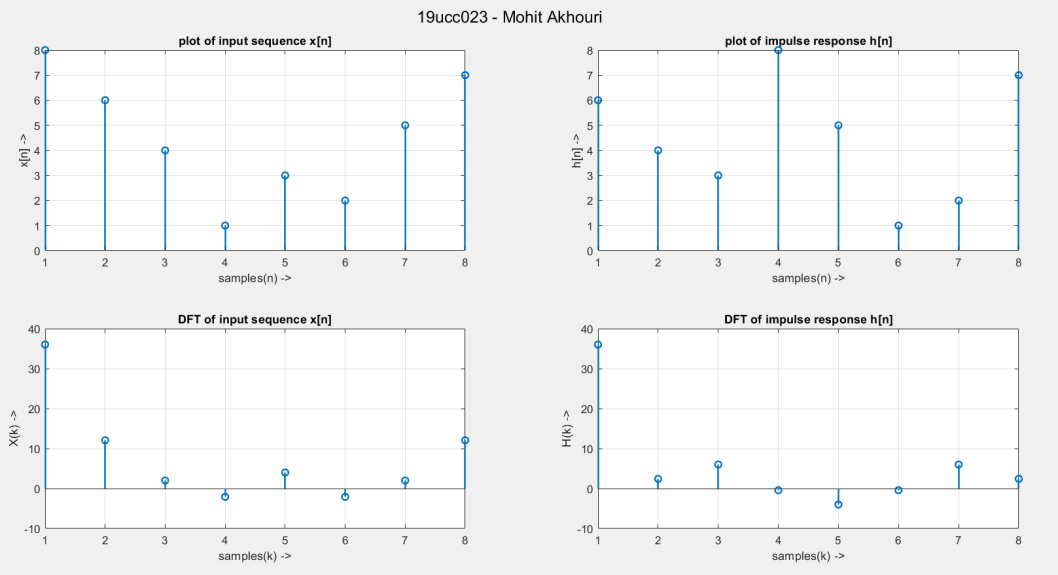

Figure 4.12 Plots of x[n], h[n], X(k) and H(k)

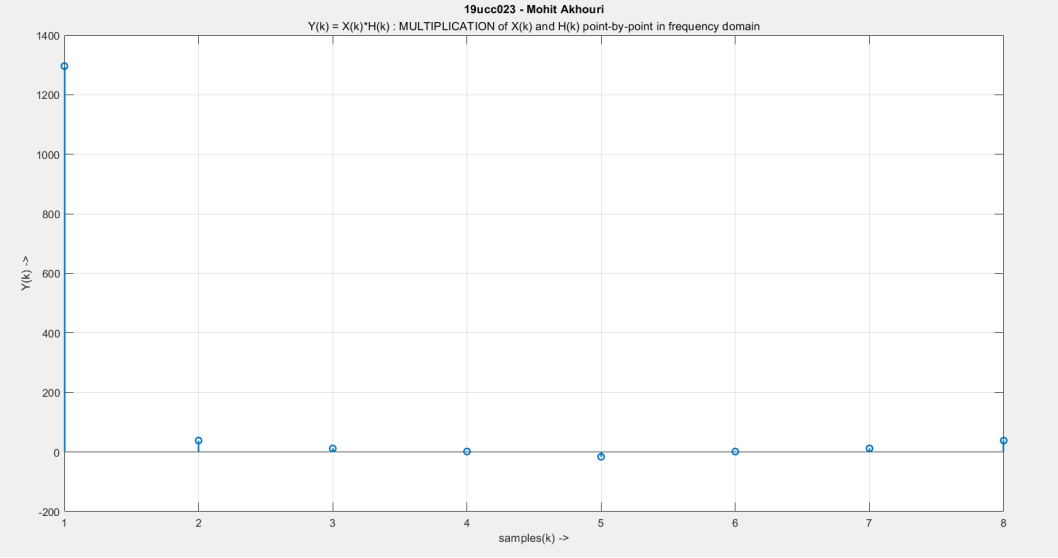

Figure 4.13 Plot of multiplication of X(k) and H(k) in frequency domain

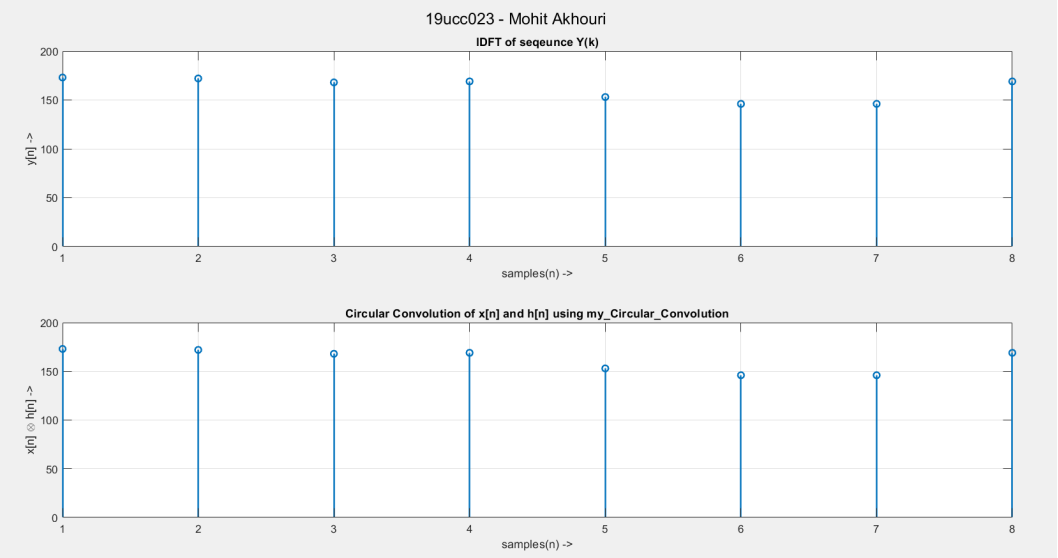

Figure 4.14 Comparison of Plots of IDFT sequence and Circular Convolution (USER-DEFINED)

4.4.2.2 Verify Circular Convolution through MATRIX Multiplication :

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

% In this code , we will perform matrix multiplication of Convolution
% matrix, DFT matrix and inverse of DFT matrix, next we will compare
 the
% results obtained with circular convolution of x[n] and h[n]

clc;
clear all;
close all;

% ALGORITHM : Initialize 8-point x[n] and h[n]
% First calculate the convolution matrix H and DFT matrix D
% Now calculate 8-point DFT of x[n]
% Then we perform the matrix multiplication and compare the results
% with the circular convolution of x[n] and h[n]

x = randperm(8,8); % input sequence x[n]
h = randperm(8,8); % impulse response h[n]
circ_conv_xh = my_Circular_Convolution(x,h); % Circular convolution of
 x[n] and h[n]

length_x = length(x); % length of input sequence x[n]
length_h = length(h); % length of impulse response h[n]

rows = 8; % rows of matrix ( both convolution matrix and DFT matrix )
columns = 8; % columns of matrix ( both convolution matrix and DFT
 matrix )
n = 8; % parameter for passing to myDFT function

H = myCirConvMat(h,length_x); % Circular convolution matrix H
[Xf , D_mat_x] = myDFT(x,8); % Calculating DFT matrix D_mat_x and DFT
 (Xf) of input sequence x[n]

D_mat_x_inverse = inv(D_mat_x); % Inverse of DFT matrix D_mat_x of
 x[n]

Hf = (D_mat_x * H)*(D_mat_x_inverse); % calculating Hf matrix

% calculating Yf = Hf * Xf
Yf = zeros(1,n); % initializing Yf vector

for i=1:rows
    sum = 0;
    for j=1:columns
        sum = sum + (Hf(i,j)*Xf(j));
    end
    Yf(i) = sum;
end
```

Figure 4.15 Part 1 of the code for observation 4

```
% calculating y = D8(-1) . Yf to be compared with circ_conv_xh
y = zeros(1,n); % initializing vector 'y'
for i=1:rows
    sum = 0;
    for j=1:columns
        sum = sum + (D_mat_x_inverse(i,j)*Yf(j));
    end
    y(i) = sum;
end

% plotting x[n] and h[n]
figure;
subplot(2,1,1);
stem(x,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] ->');
title('plot of input sequence x[n]');
grid on;
subplot(2,1,2);
stem(h,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('h[n] ->');
title('plot of impulse response h[n]');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');

% plotting y[n] and circ_conv_xh for comparison of Circular
 convolution
figure;
subplot(2,1,1);
stem(y,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('y[n] ->');
title('y[n] = D_{8}^{-1}.Y_{F} : circular convolution obtained via
 MATRIX MULTIPLICATION');
grid on;
subplot(2,1,2);
stem(circ_conv_xh,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] \otimes h[n] ->');
title('Circular Convolution of x[n] and h[n] using my\_Circular
\_Convolution');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 4.16 Part 2 of the code for observation 4

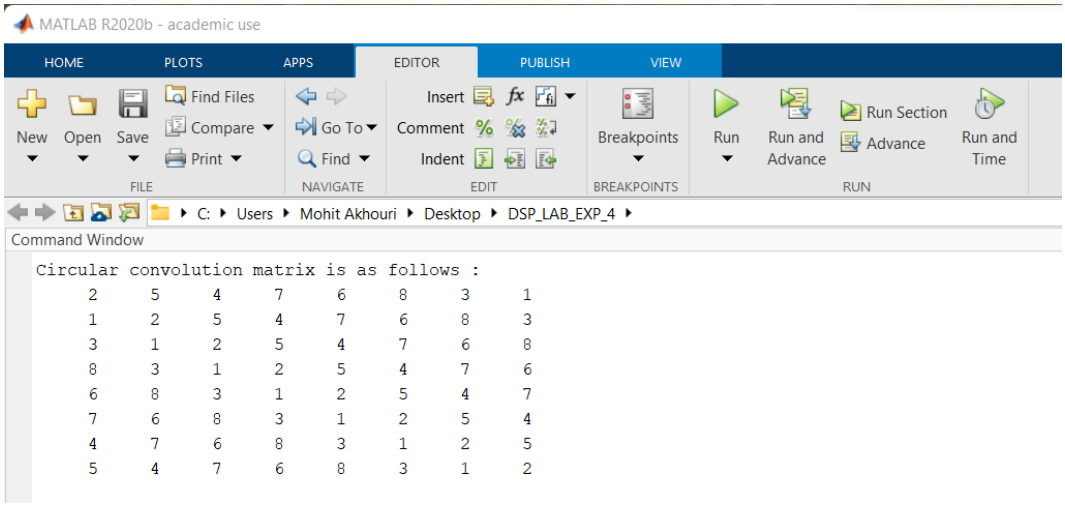

Figure 4.17 Display of Circular Convolution matrix of impulse response h[n]

```
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 4.18 Display of DFT matrix of Input sequence x[n]

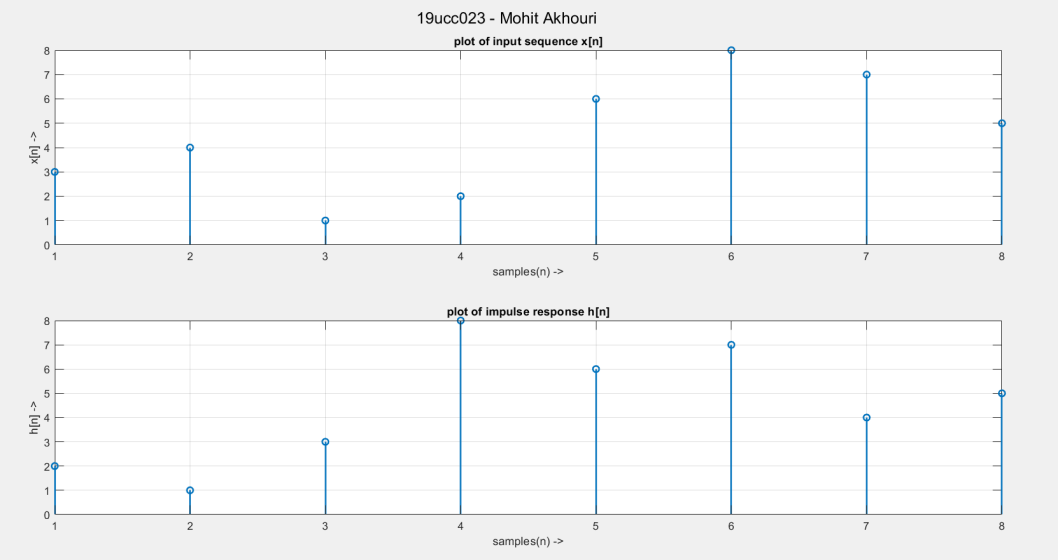

Figure 4.19 Plot of input sequence x[n] and impulse response h[n]

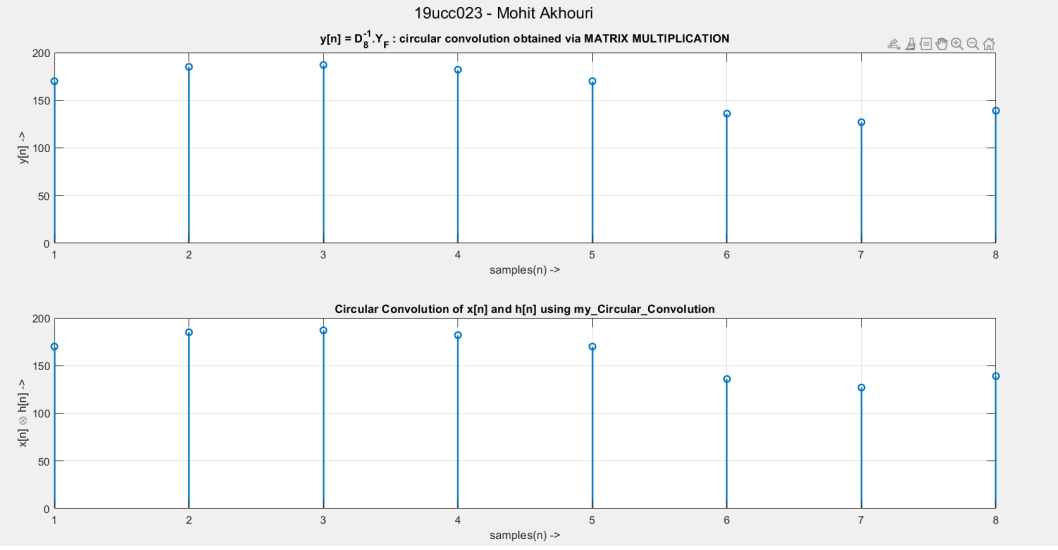

Figure 4.20 Comparison of Plots of MATRIX MULTIPLICATION OUTPUT and circular Convolution obtained (USER-DEFINED)

4.4.3 Circular Convolution in Simulink :

```
% 19ucc023
% Mohit Akhouri
% Experiment 4 - Observation 5

% This code will call the simulink model 'Simulink_Observation_5' for
% calculation of circular convolution between x[n] and h[n]

sim('Simulink_Observation_5'); % calling the simulink model

% plotting the two circular convolutions obtained via Simulink Model

figure;
subplot(2,1,1);
stem(out.circ_conv_xh_1.data,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] \otimes h[n] ->');
title('Circular Convolution between x[n] and h[n] obtained via
 SIMULINK MODEL through my\_Circular\_Convolution function');
grid on;
subplot(2,1,2);
stem(out.circ_conv_xh_2.data,'Linewidth',1.5);
xlabel('samples(n) ->');
ylabel('x[n] \otimes h[n] ->');
title('Circular Convolution between x[n] and h[n] obtained via
 SIMULINK MODEL through DFT multiplication');
grid on;
sgtitle('19ucc023 - Mohit Akhouri');
```

Figure 4.21 Code for the observation 5

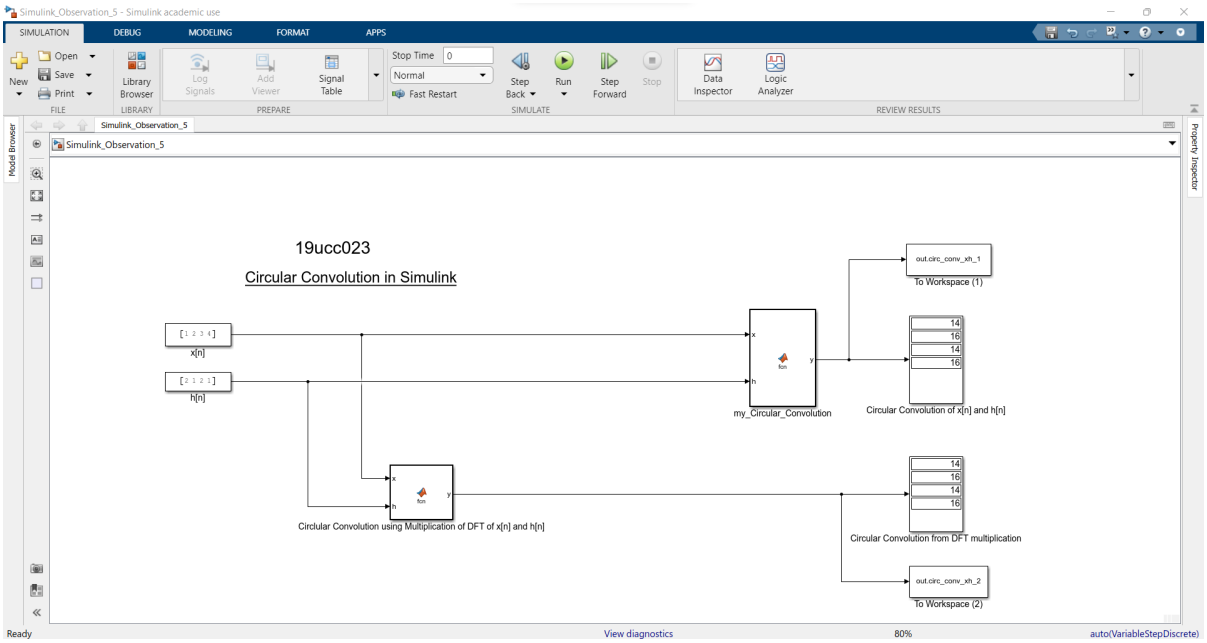

Figure 4.22 Simulink Model used for Circular Convolution

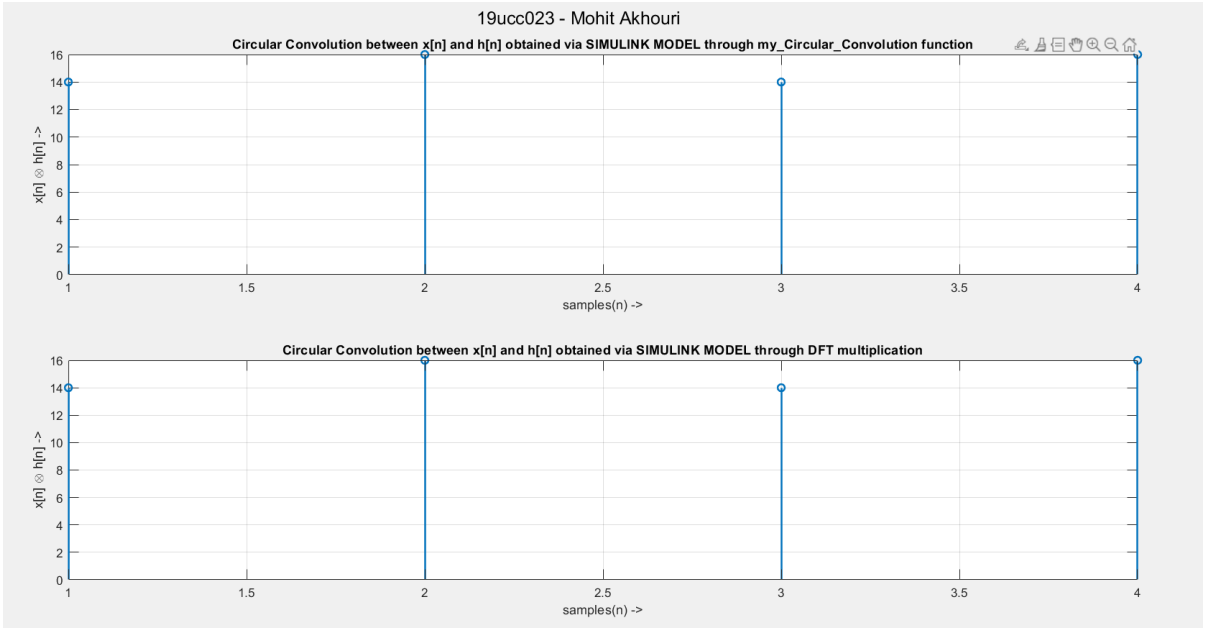

Figure 4.23 Plot of Circular Convolution (through 2 methods) obtained via Simulink Model

4.4.4 Functions used in main codes for Circular Convolution , DFT and IDFT :

4.4.4.1 myCirConvMat.m function code :

```
function [H] = myCirConvMat(h,n)
% This function calculates the circular convolution matrix
% here 'h' is the impulse response and 'n' is the length of the input
% sequence x[n]

% 19ucc023
% Mohit Akhouri

h = flip(h); % flipping the impulse response h[n]
rows = n; % rows of Circular convolution matrix
columns = n; % columns of Circular convolution matrix
H = zeros(rows,columns); % initializing the Circular convolution
 matrix with zeros

length_h = length(h); % length of impulse response h[n]
row_ind = 1; % row index for storing in matrix

% ALGORITHM : for calculation of circular convolution matrix
% Flip the h[n]
% run a loop from i to last index of h[n] and store the values in H
% run a loop from 1 to i-1 ( this is the cyclic part ) of h[n] and
 store
% values in H

for i=length_h:-1:1
    offset = 1; % for column index ( for storage of values in H )

    % loop from 'i' to 'last index' of h[n]
    for j=i:length_h
        H(row_ind,offset) = h(j);
        offset = offset + 1;
    end

    % loop from 1 to 'i-1' of h[n]
    for j=1:i-1
        H(row_ind,offset) = h(j);
        offset = offset + 1;
    end

    row_ind = row_ind + 1; % incrementing row index

end

disp('Circular convolution matrix is as follows :');
disp(H); % displaying the matrix

end
```

Published with MATLAB® R2020b

Figure 4.24 myCirConvMat function to calculate **Circular Convolution matrix** of input given to it

4.4.4.2 my_Circular_Convolution.m function code :

```
function [y] = my_Circular_Convolution(x,h)
% This function will calculate the circular convolution
% of input sequence x[n] and impulse response h[n]

% 19ucc023
% Mohit Akhouri

% ALGORITHM : First make the length of both sequences equal , then
 call
% myCirConvMat function to calculate the circular convolution matrix.
% Now multiply the Matrix with 'column-vector' x[n] to get the final
 result

% Making the length of the two sequences equal by 'PADDING WITH ZEROS'

length_x = length(x); % length of input sequence x[n]
length_h = length(h); % length of impulse response h[n]

if(length_x > length_h)
    h = [h zeros(1,length_x - length_h)]; % padding 'h[n]' with zeros
    length_h = length(h);
else
    x = [x zeros(1,length_h - length_x)]; % padding 'x[n]' with zeros
    length_x = length(x);
end

n = length_x; % length of input sequence x[n] to be passed to function

% Calling the myCirConvMat function to calculate the circular
 convolution
% matrix and storing it in variable H

H = myCirConvMat(h,n); % calling the function for Circular Conv.
 matrix

% Algorithm for calculation of circular convolution is as follows

y = zeros(1,n); % initializing the output vector

for i=1:n
    sum = 0;
    for j=1:n
       sum = sum + (H(i,j)*x(j));
    end
    y(i) = sum;
end

end
```

Published with MATLAB® R2020b

Figure 4.25 my_Circular_Convolution function to calculate the **Circular Convolution** of x[n] and h[n]

4.4.4.3 myDFT.m function code :

```
function [X,D] = 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 4.26 myDFT function to calculate the **Discrete Fourier Transform** of input given to it

4.4.4.4 myIDFT.m function code :

```
function [x] = myIDFT(X,N)
% This function will calculate the inverse discrete fourier transform
% (IDFT) of N-point DFT sequence X(k)

% 19ucc023
% Mohit Akhouri

x=zeros(1,N); % initializing output

% main loop algorithm to calculate IDFT
for n=1:N
    sum=0;
    for k=1:N
        sum=sum+(X(k).*(exp(1j*2*pi*(k-1)*(n-1)/N)));
    end
    sum=sum/N;
    x(n)=sum;
end

x=abs(x);

end
```

Published with MATLAB® R2020b

Figure 4.27 myIDFT function to calculate the **Inverse Discrete Fourier Transform** of input given to it

4.5 Conclusion

In this experiment , we learnt the concepts of **Circular Convolution** and **change of Basis** of Digital Signal Processing. We learnt about **circular convolution matrix** and how it can be used to computer circular convolution of two sequences. We also learnt about the concept of **Change of basis**. In change of basis , we learnt about the relationship between Circular Convolution and **DFT multiplication**. We learnt that after taking IDFT of product of DFT of two sequences x[n] and h[n] , we get their circular convolution. We also learnt about the relationship between Circular Convolution and **Matrix Multiplication** . In Matrix multiplication , we generated **Circular Convolution matrix** , **DFT matrix** and **inverse of DFT matrix** and through their multiplication , we obtained the circular convolution . We implemented the techniques in MATLAB . We also built model for Circular Convolution in Simulink and compared the results obtained from MATLAB coding.