Contents

1 | Experiment N0 02

2 | Experiment Name

1. Study of Circular Convolution and Implementation using MATLAB code.
2. Plotting the figure of given two signals& summation and subtraction of them using MATLAB code.
3. Plotting the given two signal in one figure MATLAB code.

3 | Introduction

Circular convolution is "the fundamental operation to compute discrete time signals". Considerring two finite sequence periodic discrete time signals with length N are x1[n], x2[n], the periodic convolution of these two discrete signals is, . [1] The operation relates the output sequence y(n) of a linear-time invariant (LTI) system, with the input sequence x(n) and the unit sample sequence h(n), as shown in Fig. 1. In this experiment, we found convolution manually instead using built-in function 'ifft' in ***matlab***.
Formula for convolution is:

$$y(n) = \sum_{m=1}^{N-1} x1(k) * x2(n-m)$$

The shifting in this operation is a circular shifting. In this experiment we had implemented circular convolution using ***matlab*** code. There are two methods to determine circular convolution. They are,

- Concentric circle method
- Matrix multiplication method.

4 | Objectives

The main objectives of this experiments were:

- To find the convolution of the input sequence
- To develop an algorithm to find convolution without using 'ifft' function

5 | Equipment Required

MATLAB

6 | Algorithm

6.1 | Matrix Multiplication Method:

- **Step 1** The NXN matrix is formed by repeating one of the sequences. This is then achieved by making a circular shift of one sample.
- **Step 2** Then the Second sequence forms a column matrix.
- **Step 3** Finally the result of circular convolution is calculated by multiplying these two matrices.

7 | Matlab Code

Here is the input code of convolution-

7.1 | Circular convolution using matrix method

```
a=[1 2 3 4]
b=[]
c=[zeros(1,4)]
d=[zeros(1,4)]
k=4;
b(1)=a(k);
for i=1:1:3
    b(i+1)=a(i);
end
c(1)=b(k);
for i=1:1:3
    c(i+1)=b(i);
end
d(1)=c(k);
for i=1:1:3
    d(i+1)=c(i);
end
A=[a;b;c;d]
A=A.'
h=[1 1 1 1]
conv=h*A
stem(conv);
```

7.2 | Summation-subtraction

```
clc;
clear all;
n1=[0 0 0 2 2 2 1 1 1 0 2];
n2=[2 2 0 1 1 1 0 0 0 0 3];
m=length(n1);
subplot(4,1,1)
stem(n1)
title('n1')
subplot(4,1,2)
stem(n2)
title('n2')
subplot(4,1,3)
stem(n1+n2)
title('n3')
subplot(4,1,4)
stem(n1-n2)
title('n4')
```

7.3 | Two signal's figure in one plot

```
h = 1 ;  % height
a = 2 ;  % top side
b = 4 ;   % base
h2 = 2 ;  % height
%%Frame vertices
A = [0 0] ;
B=[3*b,0];
C = [0.5*(b-a)+a+2*b h] ;
D = [2*b+0.5*(b-a) h] ;
```

```
E=[2*b-0.5*(b-a) h2];
F=[b+0.5*(b-a) h2];
G=[b-0.5*(b-a) h]
H=[0.5*(b-a) h]
coor=[A;B;C;D;E;F;G;H;0 0];
coor1=[A;4 0;3 1;1 1;0 0]
subplot(2,1,1)
plot(coor1(:,1), coor1(:,2))
%patch(coor1(:,1), coor1(:,2),'w')
subplot(2,1,2)
plot(coor(:,1), coor(:,2))
```

8 | Result & Discussion

Here is the outcome of above code-

8.1 | Circular convolution

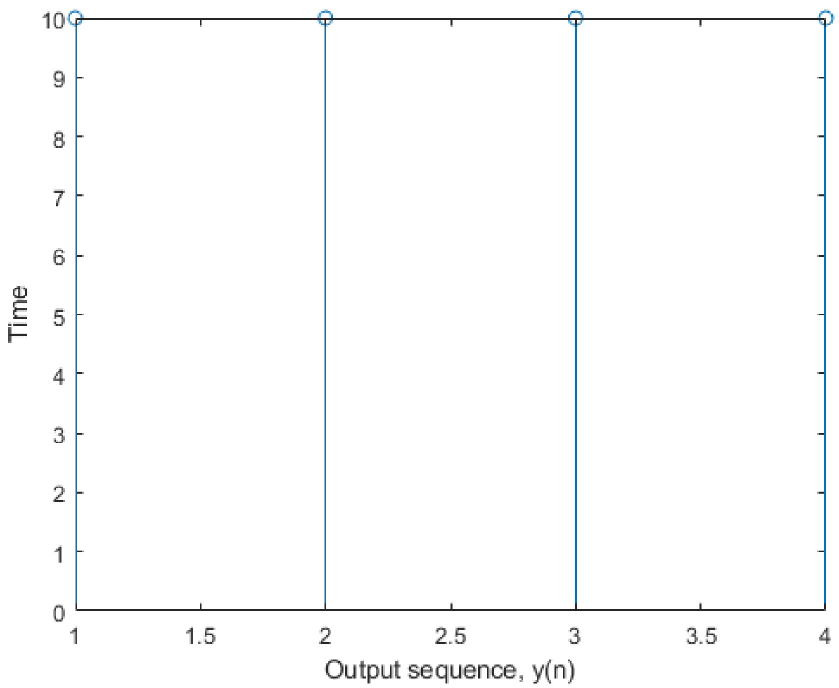

Figure 8.1: Graphical Plot of output sequence y(n).

8.2 | Summation subtraction of signals

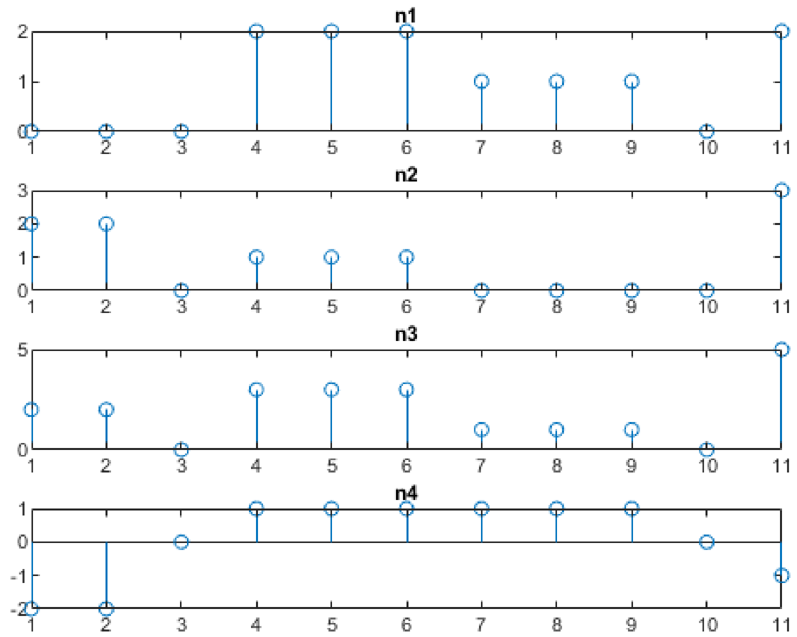

Figure 8.2: Graphical Plot of n1, n2, n1+n2, n1-n2 signals.

8.3 | Plotting two signal in one figure

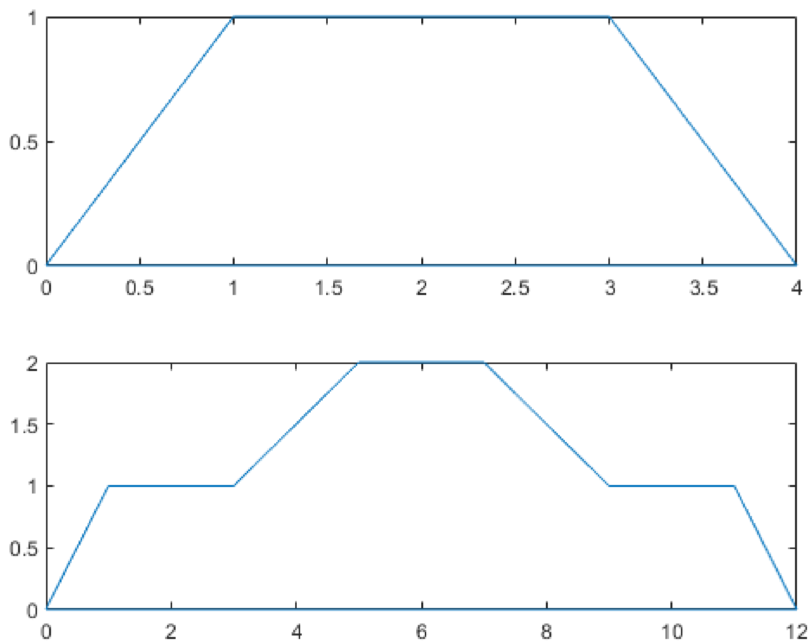

Figure 8.3: Graphical Plot signal1 & signal2

The outcomes of this experiment were achieved as desired i.e. the circular convolution of input signals using the matrix method. The matrix multiplication approach handled periodic input signal by circular

shifting method. For example, for N length input, the shifting was done N-1 times. After each shifting, the shifted values are stored in another array. Thereby, there are N number of arrays where each array has N number of elements. The matrix formed using this array is of N*N size. By transposing them this matrix, the actual matrix needed was formed. And finally using matrix multiplication of this matrix with another column matrix, the result obtained.
To draw the trapezoidal signal all the angular points were determined and then plotted.

9 | Conclusion

The experiment was successful & we did not encounter any error while running the ***Matlab*** code.

10 | References

[1] Libretexts. 4.3: Discrete Time Convolution. *Engineering LibreTexts*, 5 2022.