

**School of Engineering and Applied Science (SEAS)
Ahmedabad University**

BTech(ICT) Semester VI:Digital Signal Processing

Laboratory Assignment- 3

Enrollment No:AU1841131

Name:Mansi Dobariya

AIM :: Lab3 helps to revise the concept of signal and systems. Solving the properties of signal like circular convolution,convn and conv2 functions in MATLAB .Also,3x3 kernel matrix convolution with image which includes imread,rgb2gray and imshow functions. That's clearly shown in plotted image.

1. Explore command conv2 in Matlab. Take input of 2 Matrix from user and Find 2D convolution of the same. Also explore the properties of conv2 command and analyze the result.

Solution Problem-1

(a) Matlab Script:

```
1 clc;
2 clear all;
3 %First Matrix
4 rc_A=input('Enter no. Row and Column value of Matrix A')
5 for i = 1:rc_A
6     for j = 1:rc_A
7         A(i,j) = input(sprintf('Enter value (%d, %d) of Matrix A >> ', i, j))
8     end
9 end
10 %Second Matrix
11 rc_B=input('Enter no. Row and Column value of Matrix B')
12 for i = 1:rc_B
13     for j = 1:rc_B
14         B(i,j) = input(sprintf('Enter value (%d, %d) of Matrix B >> ', i, j))
15     end
16 end
17 display(A)
18 display(B)
19 %2D convolution
20 convolution2D=conv2(A,B)
21
```

(b) Approach:

taking input of matrix using for loop row-wise which is shown in below image.Having 2 matrices put into conv2 in-built function which returns the two-dimensional convolution of matrices A and B.

(c) Simulation Output:

Enter no. Row and Column value of Matrix A
5

rc_A =

5

Enter value (1, 1) of Matrix A >>
17

A =

17

Enter value (1, 2) of Matrix A >>
24

A =

17 24

Enter value (1, 3) of Matrix A >>
1

A =

17 24 1

A =

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

B =

1	3	1
0	5	0
2	1	2

convolution2D =

17	75	90	35	40	53	15
23	159	165	45	105	137	16
38	198	120	165	205	197	52
56	95	160	200	245	184	35
19	117	190	255	235	106	53
20	89	160	210	75	90	6
22	47	90	65	70	13	18

2. Application of 2D convolution on image processing applications

a) Take a standard test image “Lenna.png” from shared folder. Explore following commands and apply for given image

Solution Problem-2a

(a) Matlab Script:

```
1 clc;
2 close all;
3
4 % Original image
5 subplot(1,2,1);
6 image=imread("lenna.png"); % reads an image file
7 imshow(image); % display the image in the graphics file
```

```

8 title('Original Image');
9
10 % Scaling to gray
11 subplot(1,2,2);
12 GrayScale=rgb2gray(image); %The rgb2gray function converts RGB images to grayscale
    by eliminating the hue and saturation information while retaining the
    luminance
13 imshow(GrayScale); % display the image in the graphics file
14 title('Gray Scale of Original Image');
15

```

(b) Approach:

Using imread function, read the image and imshow function for show the image. rgb2gray function for converting colour image into gray scale.

(c) Simulation Output:



b) Now in order to perform 2D convolution, Given image becomes first input as matrix and second input will be a set of kernel matrix performing different operations on image which are mentioned below

Solution Problem-2b1 Average(blur,smooth)

(a) Matlab Script:

```

1 clc;
2 clear all;
3
4 % The original image
5 subplot(1,3,1);
6 image=imread('lenna.png'); % reads an image file
7 imshow(image); % display the image in the graphics file
8 title('Original Image');
9
10 % Scaling to gray
11 subplot(1,3,2);

```

```

12 GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
13 imshow(GrayScale); % display the image in the graphics file
14 title('Gray Scale ');
15
16 %average(blur,smooth)
17 A=zeros(3,3)
18 A(1,:)= [1/9 1/9 1/9];
19 A(2,:)= [1/9 1/9 1/9];
20 A(3,:)= [1/9 1/9 1/9]; ;
21 subplot(1,3,3);
22 con=convn(A,GrayScale) %for N-Dimension conv
23 imshow(con,[]);
24 title('A conv with GrayScale image')
25

```

(b) Approach:

Using imread function, read the image and imshow function for show the image. rgb2gray function for converting colour image into gray scale. And take convolution of image and kernel matrix.

(c) Simulation Output:



Solution Problem-2b2 Sharpen 3x3 convolution kernel

(a) Matlab Script:

```

1 clc;
2 clear all;
3
4 image=imread("lenna.png"); % reads an image file
5 imshow(image); % display the image in the graphics file
6
7 GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
8 imshow(GrayScale); % display the image in the graphics file
9
10 %Sharpen 3x3 kernel
11 subplot(1,2,1);
12 S=zeros(3,3)
13 S(1,:)= [0 -1 0];
14 S(2,:)= [-1 5 -1];
15 S(3,:)= [0 -1 0];
16 con=convn(S,image) %for N-Dimension conv
17 imshow(con)

```

```

18 title('Sharpen Kenel conv with Original image')
19 subplot(1,2,2);
20 con=convn(S,GrayScale) %for N-Dimention conv
21 imshow(con)
22 title(' with GrayScale image')

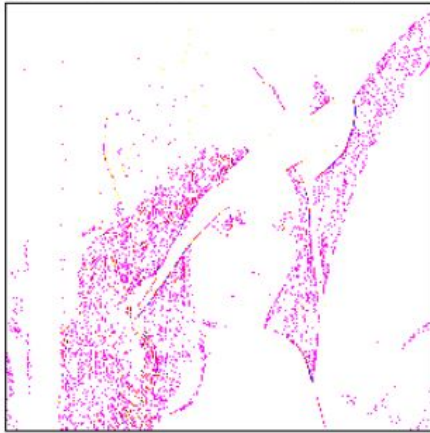
```

(b) Approach:

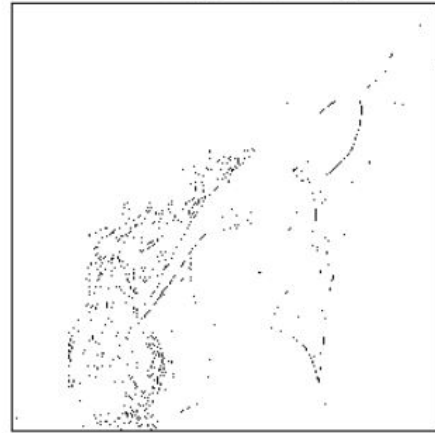
Using imread function, read the image and imshow function for show the image. rgb2gray function for converting colour image into gray scale. And take convolution of image and kernel matrix.

(c) Simulation Output:

Sharpen Kenel conv with Original image



with GrayScale image



Solution Problem-2b3 Edge detection 3x3 convolution kernel

(a) Matlab Script:

```

1 clc;
2 clear all;
3 image=imread("lenna.png"); % reads an image file
4 imshow(image); % display the image in the graphics file
5 %scaling to gray
6 GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
7 imshow(GrayScale); % display the image in the graphics file
8
9 %Edge detection kernel
10 subplot(2,3,1);
11 E=zeros(3,3)
12 E(1,:)=[0 -1 0];
13 E(2,:)=[-1 4 -1];
14 E(3,:)=[0 -1 0];
15 con=convn(E,image) %for N-Dimention conv
16 imshow(con)
17 title('E with Original image')
18 %E_h
19 subplot(2,3,2);
20 Eh=zeros(3,3)
21 Eh(1,:)=[0 0 0];
22 Eh(2,:)=[-1 2 -1];
23 Eh(3,:)=[0 0 0];
24 con=convn(Eh,image) %for N-Dimention conv
25 imshow(con)

```

```

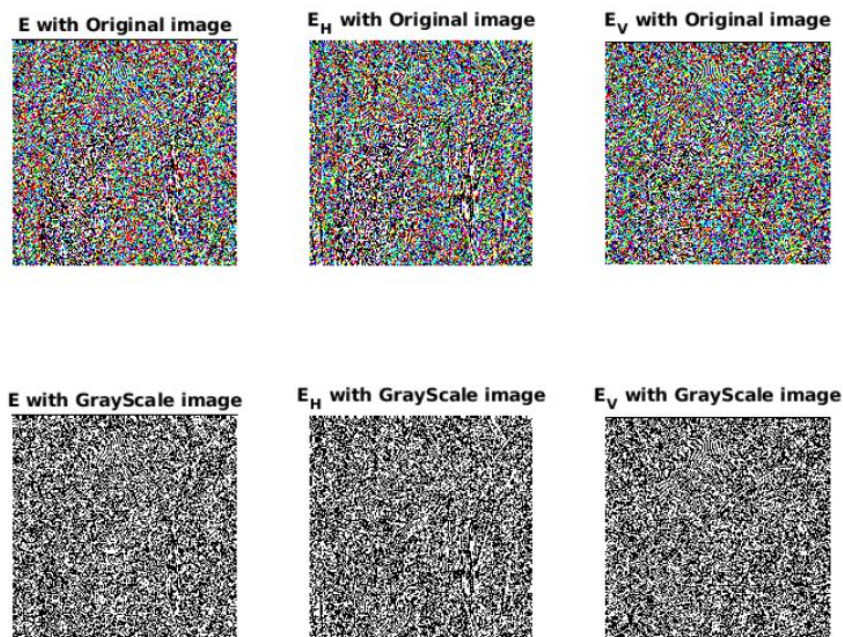
26 title('E_H with Original image')
27 %E_v
28 subplot(2,3,3);
29 Ev=zeros(3,3)
30 Ev(1,:)= [0 -1 0];
31 Ev(2,:)= [0 2 0];
32 Ev(3,:)= [0 -1 0];
33 con=convn(Ev,image) %for N-Dimension conv
34 imshow(con)
35 title('E_V with Original image')
36 %with grayscale image
37 subplot(2,3,4);
38 con=convn(E,GrayScale) %for N-Dimension conv
39 imshow(con)
40 title('E with GrayScale image')
41
42 subplot(2,3,5);
43 con=convn(Eh,GrayScale) %for N-Dimension conv
44 imshow(con)
45 title('E_H with GrayScale image')
46
47 subplot(2,3,6);
48 con=convn(Ev,GrayScale) %for N-Dimension conv
49 imshow(con)
50 title('E_V with GrayScale image')

```

(b) Approach:

Using imread function, read the image and imshow function for show the image. rgb2gray function for converting colour image into gray scale. And take convolution of image and kernel matrix.

(c) Simulation Output:



Solution Problem-2b4 Gradient detection 3x3 convolution kernel

(a) Matlab Script:

```

1 clc;
2 clear all;
3

```



```

4 image=imread("lenna.png"); % reads an image file
5 imshow(image); % display the image in the graphics file
6
7 % Scaling to gray
8 GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
9 imshow(GrayScale); % display the image in the graphics file
10
11 %Gradient 3x3 kernel
12 %G_H
13 subplot(2,2,1);
14 Gh=zeros(3,3)
15 Gh(1,:)=[-1 -1 -1];
16 Gh(2,:)=[0 0 0];
17 Gh(3,:)=[1 1 1];
18 con=convn(Gh,image) %for N-Dimension conv
19 imshow(con)
20 title('G_H with Original image')
21 %with grayscale image
22 subplot(2,2,2);
23 con=convn(Gh,GrayScale) %for N-Dimension conv
24 imshow(con)
25 title('G_H with GrayScale image')
26
27 %G_V
28 subplot(2,2,3);
29 Gv=zeros(3,3)
30 Gv(1,:)=[-1 0 1];
31 Gv(2,:)=[-1 0 1];
32 Gv(3,:)=[-1 0 1];
33 con=convn(Gv,image) %for N-Dimension conv
34 imshow(con)
35 title('G_V with Original image')
36
37 subplot(2,2,4);
38 con=convn(Gv,GrayScale) %for N-Dimension conv
39 imshow(con)
40 title('G_V with GrayScale image')

```

(b) Approach:

Using imread function, read the image and imshow function for show the image. rgb2gray function for converting colour image into gray scale. And take convolution of image and kernel matrix.

(c) Simulation Output:

G_H with Original image



G_H with GrayScale image



G_V with Original image



G_V with GrayScale image



Solution Problem-2b5 Sobel Operator 3x3 convolution kernel

(a) Matlab Script:

```
1 clc;
2 clear all;
3
4 image=imread("lenna.png"); % reads an image file
5 imshow(image); % display the image in the graphics file
6
7 % Scaling to gray
8 GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
9 imshow(GrayScale); % display the image in the graphics file
10
11 %Sobel 3x3 kernel
12 %S_H
13 subplot(2,2,1);
14 Sh=zeros(3,3)
15 Sh(1,:)= [1 2 1];
16 Sh(2,:)= [0 0 0];
17 Sh(3,:)= [-1 -2 -1];
18 con=convn(Sh,image) %for N-Dimension conv
19 imshow(con)
20 title('S_H with Original image')
21 %with grayscale image
22 subplot(2,2,2);
23 con=convn(Sh,GrayScale) %for N-Dimension conv
24 imshow(con)
25 title('S_H with GrayScale image')
26
27 %S_V
28 subplot(2,2,3);
29 Sv=zeros(3,3)
30 Sv(1,:)= [1 0 -1];
```

```

31 Sv(2,:)= [2 0 -2];
32 Sv(3,:)= [1 0 -1];
33 con=convn(Sv,image) %for N-Dimension conv
34 imshow(con)
35 title('S_V with Original image')
36
37 subplot(2,2,4);
38 con=convn(Sv,GrayScale) %for N-Dimension conv
39 imshow(con)
40 title('S_V with GrayScale image')

```

(b) Approach:

Using imread function, read the image and imshow function for show the image. rgb2gray function for converting colour image into gray scale. And take convolution of image and kernel matrix.

(c) Simulation Output:

S_H with Original image



S_H with GrayScale image



S_V with Original image



S_V with GrayScale image



3. Develop a MATLAB function to obtain circular convolution of two sequences. Verify the function for following sequence. Write a script file to use the developed function.

Solution Problem- 3 a & b

(a) Matlab Script:

```

1 clc;
2 close all;
3 x1=input('Enter x1 :: ');
4 x2=input('Enter x2 :: ');
5
6 y_convofunc=circular_convo(x1,x2);
7

```

```

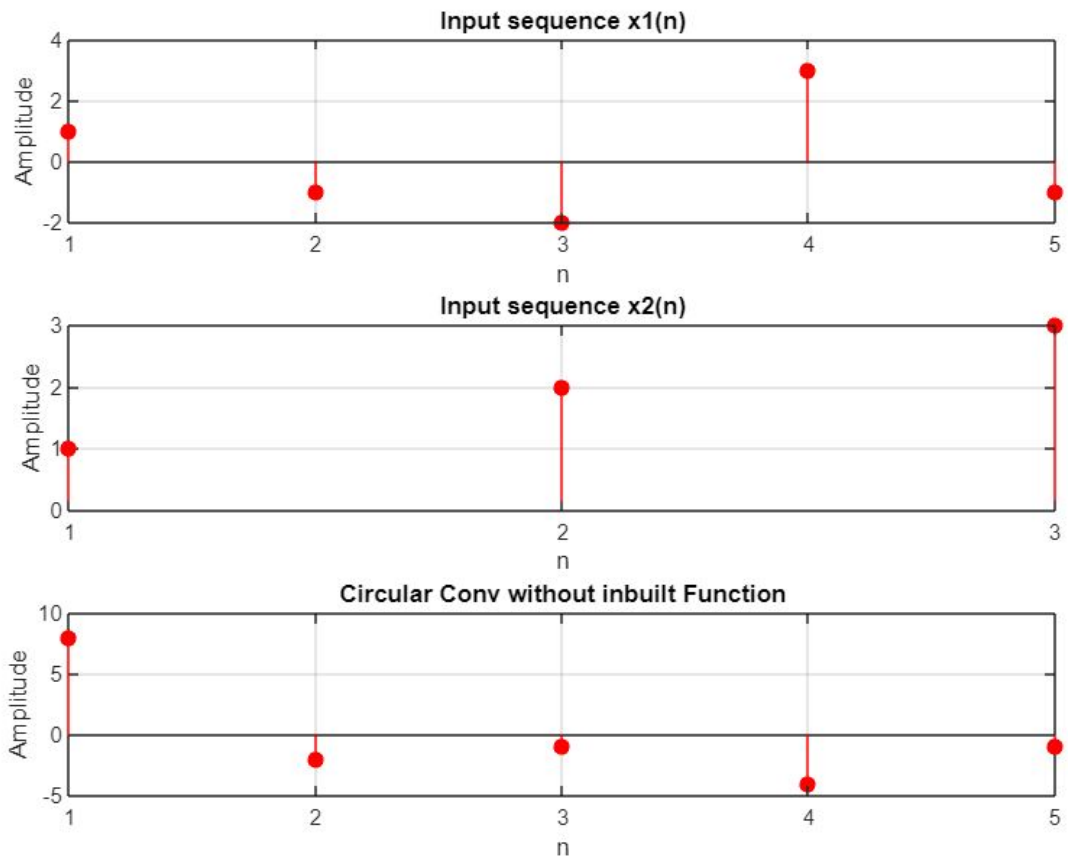
8 subplot(3,1,1);
9 stem(x1,'r');
10 hold on;
11 grid on;
12 title('Input sequence x1(n)');
13 xlabel('n');
14 ylabel('Amplitude');
15 xticks(1:1:length(x1));
16
17 subplot(3,1,2);
18 stem(x2,'r');
19 hold on;
20 grid on;
21 title('Input sequence x2(n)');
22 xlabel('n');
23 ylabel('Amplitude');
24 xticks(1:1:length(x2));
25
26 subplot(3,1,3);
27 stem(y_convofunc,'r');
28 hold on;
29 grid on;
30 title('Circular Conv without inbuilt Function');
31 xlabel('n');
32 ylabel('Amplitude');
33 xticks(1:1:length(y_convofunc));
34
35 %calculate
36 function ccov=circular_convo(x1, x2)
37     if (length(x1) >= length(x2))
38         x1padding=x1;
39         x2padding=[x2 zeros(1, length(x1)-length(x2))];
40     else
41         x2padding=x2;
42         x1padding=[x1 zeros(1, length(x2)-length(x1))];
43     end
44     ccov = ifft(fft(x1padding).*fft(x2padding));
45     ccov = ccov(1:max(length(x1), length(x2)));
46 end
47

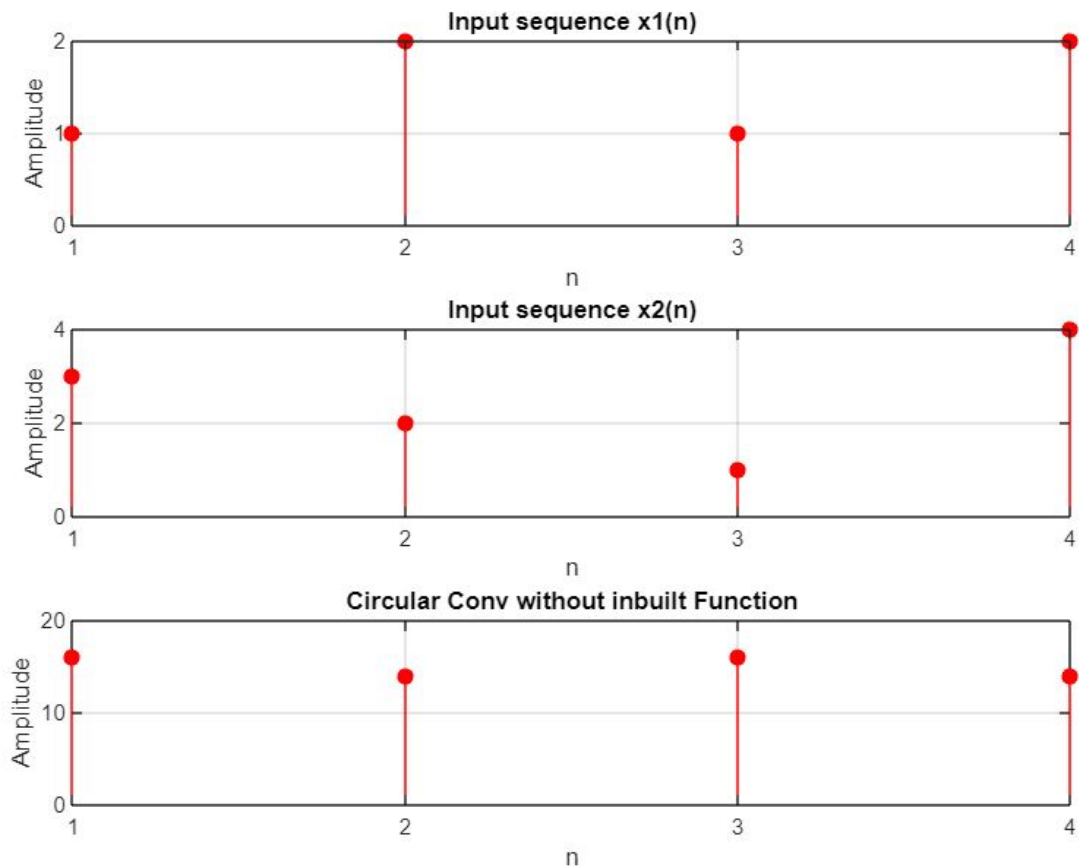
```

(b) Approach:

taking input x1 and x2 for the circular convolution which are passed own func. For the circular convolution of x and y should be same length, the arrays must be padded with zeros to length $\min(N(\text{length of } x1) + L(\text{length of } x2) - 1)$ before you take the DFT. then inverting the product of the DFTs, retain only the first $N + L - 1$ elements. so getting max length of padding in convo result,

(c) Simulation Output a and b:





Solution Problem- 3 c

(a) Matlab Script:

```

1
2 clc;
3 close all;
4 N=8;
5 dn=0.1;
6 n=0:dn:N-1;
7
8 x1=cos((2*pi/N).*n);
9 x2=sin((2*pi/N).*n);
10
11 y_convofunc=circular_convo(x1,x2);
12
13 subplot(3,1,1);
14 stem(x1,'r');
15 hold on;
16 grid on;
17 title('Input sequence x1(n) ');
18 xlabel('n');
19 ylabel('Amplitude');
20
21 subplot(3,1,2);
22 stem(x2,'r');
23 hold on;
24 grid on;
25 title('Input sequence x2(n) ');
26 xlabel('n');
27 ylabel('Amplitude');
28
29 subplot(3,1,3);

```

```

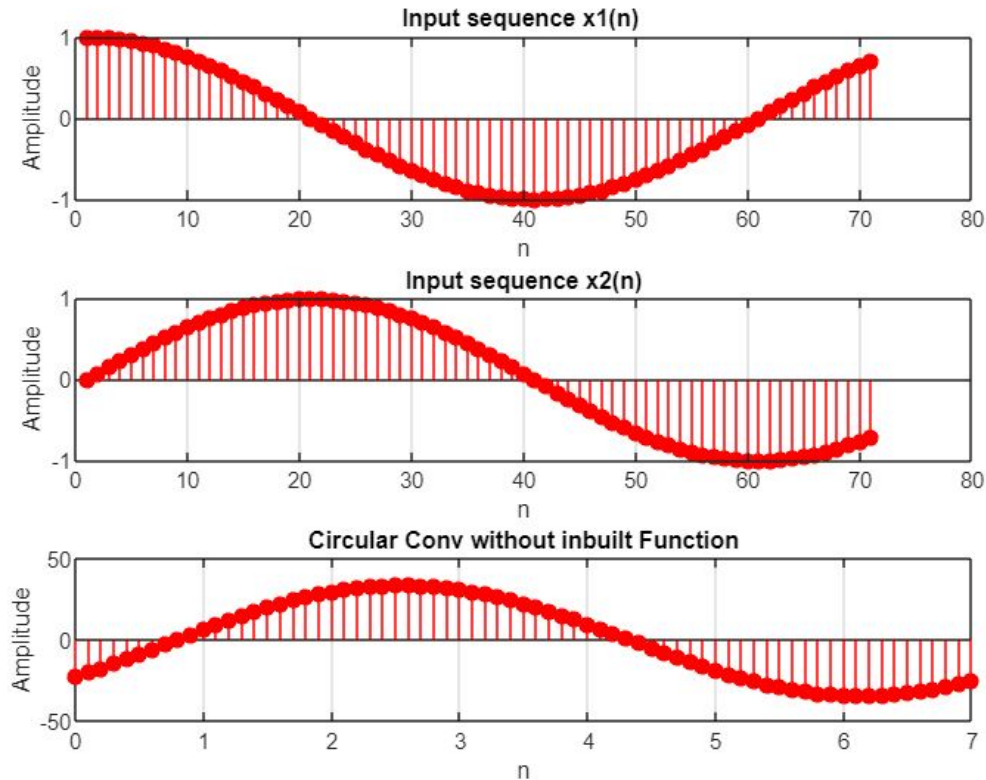
30 stem(n, y_convofunc, 'r');
31 hold on;
32 grid on;
33 title('Circular Conv without inbuilt Function');
34 xlabel('n');
35 ylabel('Amplitude');
36
37
38 function ccov=circular_convo(x1, x2)
39     if (length(x1) >= length(x2))
40         x1padding=x1;
41         x2padding=[x2 zeros(1, length(x1)-length(x2))];
42     else
43         x2padding=x2;
44         x1padding=[x1 zeros(1, length(x2)-length(x1))];
45     end
46     ccov = ifft(fft(x1padding).*fft(x2padding));
47     ccov = ccov(1:max(length(x1), length(x2)));
48 end
49

```

(b) Approach:

here x_1 and x_2 will be sinusoidal signal so taking input x_1 and x_2 for the circular convolution which are passed own func. For the circular convolution of x and y should be same length, the arrays must be padded with zeros to length $\min(N(\text{length of } x_1) + L(\text{length of } x_2) - 1)$ before you take the DFT. then inverting the product of the DFTs, retain only the first $N + L - 1$ elements. so getting max length of padding in convo result,

(c) Simulation Output c:



4. Write a MATLAB program to find circular convolution of two sequences using Matrix

Multiplication method.

Solution Problem- 4 a & b

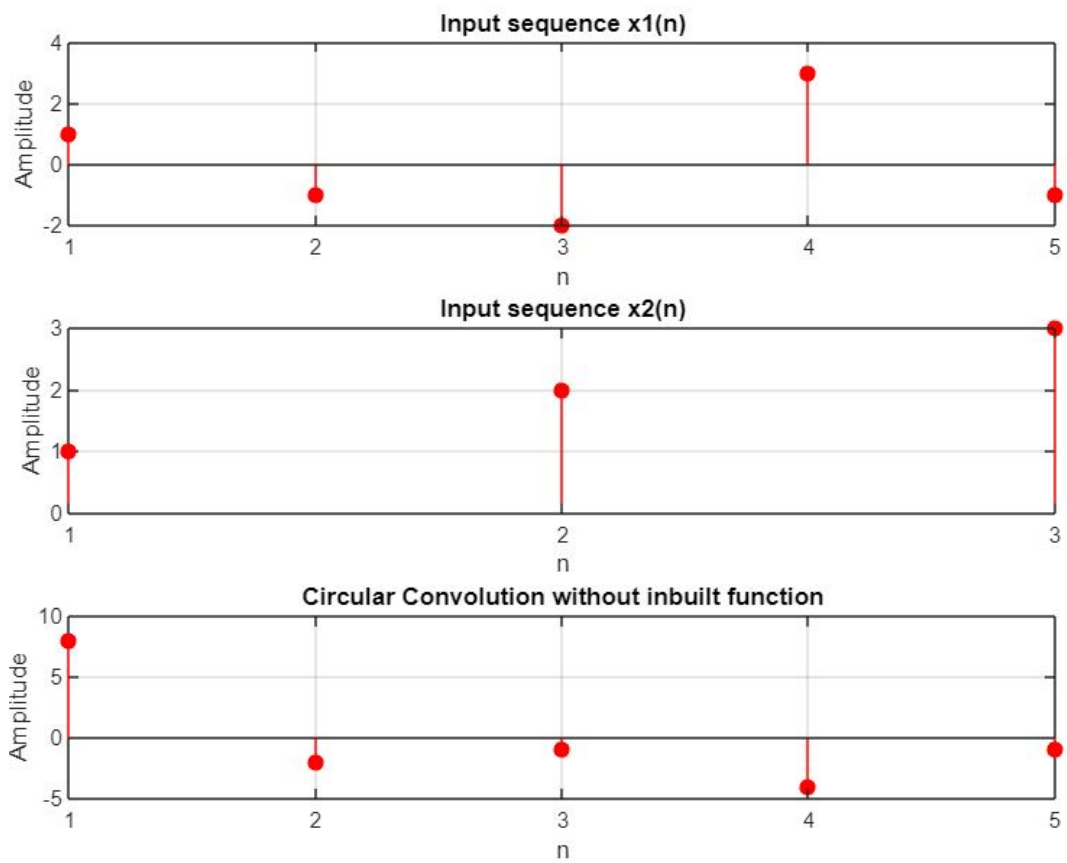
(a) Matlab Script:

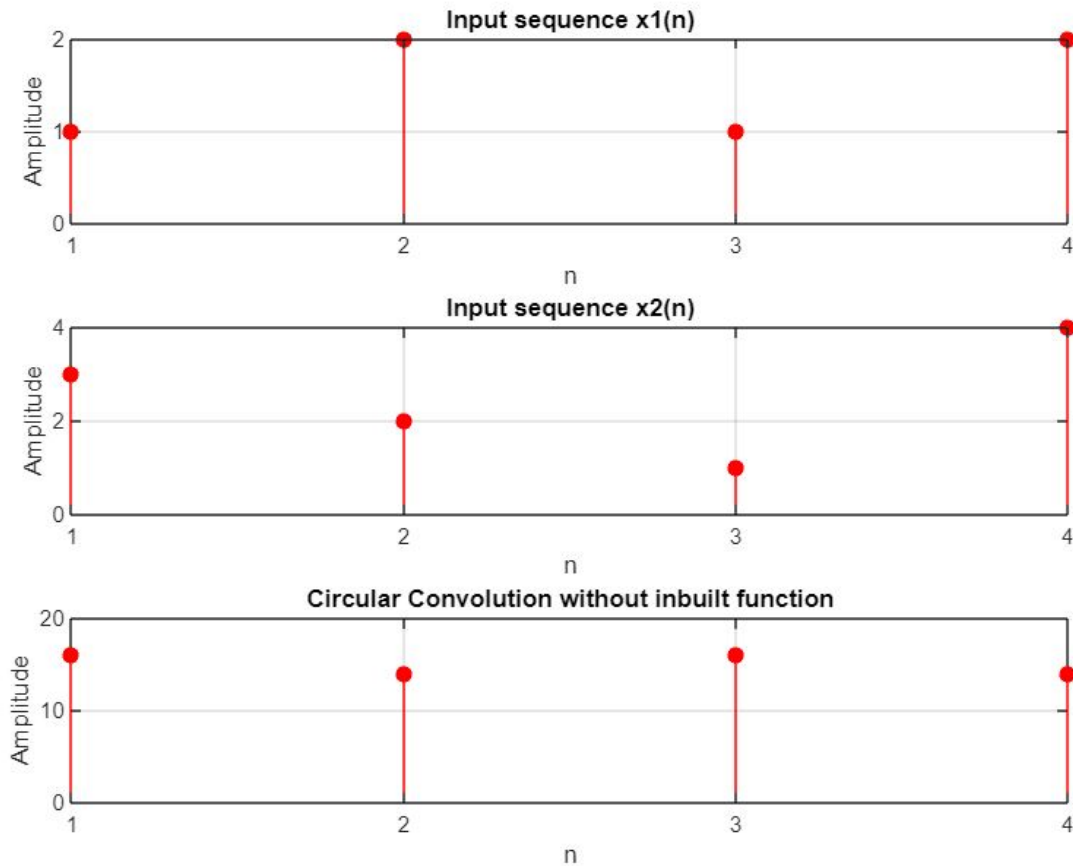
```
1  clc;
2  close all;
3  x1=input('Enter x1 :: ');
4  x2=input('Enter x2 :: ');
5
6  if (length(x1) >= length(x2))
7      [y_convofunc, matrix, x3]=circular_convo(x1,x2);
8  else
9      [y_convofunc, matrix, x3]=circular_convo(x2,x1);
10 end
11
12 subplot(3,1,1);
13 stem(x1,'r');
14 hold on;
15 grid on;
16 title('Input sequence x1(n)');
17 xlabel('n');
18 ylabel('Amplitude');
19 xticks(1:1:length(x1));
20
21 subplot(3,1,2);
22 stem(x2,'r');
23 hold on;
24 grid on;
25 title('Input sequence x2(n)');
26 xlabel('n');
27 ylabel('Amplitude');
28 xticks(1:1:length(x2));
29
30 subplot(3,1,3);
31 stem(y_convofunc,'r');
32 hold on;
33 grid on;
34 title('Circular Convolution without inbuilt function');
35 xlabel('n');
36 ylabel('Amplitude');
37 xticks(1:1:length(y_convofunc));
38
39
40 function [ccov, matrix, x3]=circular_convo(x1, x2)
41     matrix=zeros(length(x1),length(x1));
42     matrix(:, 1)=x1;
43     for count=2:length(x1)
44         arr=circshift(matrix(:, count-1), 1);
45         matrix(:,count)=arr;
46     end
47     x3=zeros(1, length(x1));
48     x3(1,1:length(x2))=x2;
49     ccov=transpose(matrix*transpose(x3));
50 end
51
```

(b) Approach:

taking input x1 and x2 for the circular convolution which are passed own func, creating the matrix from x1 then multiplied with column vector x2. first created matrix column-wise.so first column is x1 and second is the circular shift of x1,.....So created a for-loop for all columns,did a circular shift of the previous column .Taking the transpose of that product .finally got convolution Matrix.

(c) Simulation Output a and b:





Solution Problem- 4 c

(a) Matlab Script:

```

1  clc;
2  close all;
3
4
5  N=8;
6  dn=0.1;
7  n=0:dn:N-1;
8
9  x1=cos((2*pi/N).*n);
10 x2=sin((2*pi/N).*n);
11
12 if (length(x1) >= length(x2))
13     [y_convofunc, matrix, x3]=circular_convo(x1,x2);
14 else
15     [y_convofunc, matrix, x3]=circular_convo(x2,x1);
16 end
17
18 subplot(3,1,1);
19 stem(x1,'r');
20 hold on;
21 grid on;
22 title('Input sequence x1(n)');
23 xlabel('n');
24 ylabel('Amplitude');
25
26 subplot(3,1,2);
27 stem(x2,'r');
28 hold on;
29 grid on;
30 title('Input sequence x2(n)');

```

```

31 xlabel('n');
32 ylabel('Amplitude');
33
34 subplot(3,1,3);
35 stem(n, y_convofunc,'r');
36 hold on;
37 grid on;
38 title('Circular Convolution without inbuilt Function');
39 xlabel('n');
40 ylabel('Amplitude');
41
42
43 function [ccov, matrix, x3]=circular_convo(x1, x2)
44     matrix=zeros(length(x1),length(x1));
45     matrix(:, 1)=x1;
46     for count=2:length(x1)
47         arr=circshift(matrix(:, count-1), 1);
48         matrix(:,count)=arr;
49     end
50     x3=zeros(1, length(x1));
51     x3(1,1:length(x2))=x2;
52     ccov=transpose(matrix*transpose(x3));
53 end
54

```

(b) Approach:

here x1 and x2 will be sinusoidal signal so taking input x1 and x2 for the circular convolution which are passed own func, creating the matrix from x1 then multiplied with column vector x2. first created matrix column-wise.so first column is x1 and second is the circular shift of x1,.....So created a for-loop for all columns,did a circular shift of the previous column .Taking the transpose of that product .finally got convolution Matrix.

(c) Simulation Output c:

