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:

```
clc;
clear all;
%First Matrix
rc_A=input('Enter no. Row and Column value of Matrix A')
for i = 1:rc_A
  for j = 1:rc_A
    A(i,j) = input(sprintf('Enter value (%d, %d) of Matrix A >> ', i, j))
  end
end
%Second Matrix
rc_B=input('Enter no. Row and Column value of Matrix B')
for i = 1:rc_B
   for j = 1:rc_B
     B(i,j) = input(sprintf('Enter value (%d, %d) of Matrix B >> ', i, j))
   end
end
display(A)
display(B)
%2D convolution
convolution2D=conv2(A,B)

```

(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:

```
clc;
close all;

% Original image
subplot(1,2,1);
image=imread("lenna.png"); % reads an image file
imshow(image); % display the image in the graphics file
```

```
title('Original Image');

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

(c) Simulation Output:

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

Solution Problem-2b1 Average(blur,smooth)

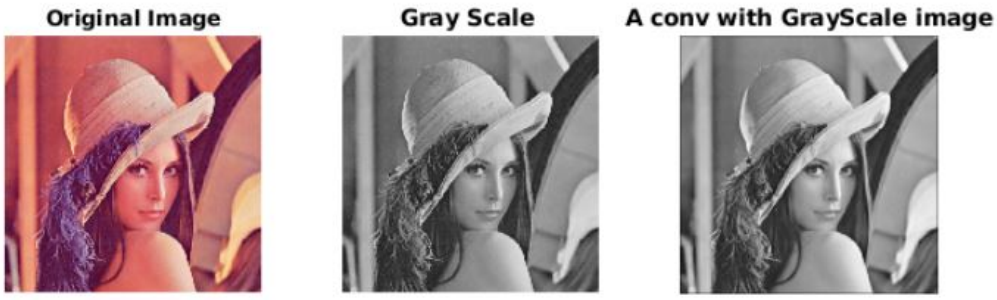
(a) Matlab Script:

```
clc;
clear all;

% The original image
subplot(1,3,1);
image=imread("lenna.png"); % reads an image file
imshow(image); % display the image in the graphics file
title('Original Image');

% Scaling to gray
subplot(1,3,2);
```

```
GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
imshow(GrayScale); % display the image in the graphics file
title('Gray Scale ');

%average(blur,smooth)
A=zeros(3,3)
A(1,:)=[1/9 1/9 1/9];
A(2,:)=[1/9 1/9 1/9];
A(3,:)=[1/9 1/9 1/9]; ;
subplot(1,3,3);
con=convn(A,GrayScale)  %for N-Dimention conv
imshow(con,[]);
title('A conv 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-2b2 Sharpen 3x3 convolution kernel

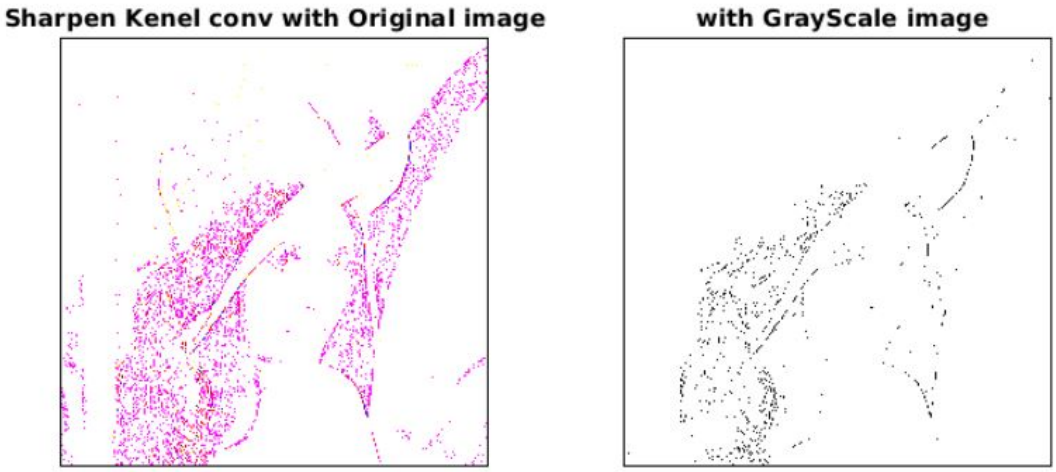
(a) Matlab Script:

```
clc;
clear all;

image=imread("lenna.png"); % reads an image file
imshow(image); % display the image in the graphics file

GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
imshow(GrayScale); % display the image in the graphics file

%Sharpen 3x3 kernel
subplot(1,2,1);
S=zeros(3,3)
S(1,:)=[0 -1 0];
S(2,:)=[-1 5 -1];
S(3,:)=[0 -1 0];
con=convn(S,image)  %for N-Dimention conv
imshow(con)
```

```
title('Sharpen Kenel conv with Original image')
subplot(1,2,2);
con=convn(S,GrayScale)  %for N-Dimention conv
imshow(con)
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:



Solution Problem-2b3 Edge detection 3x3 convolution kernel

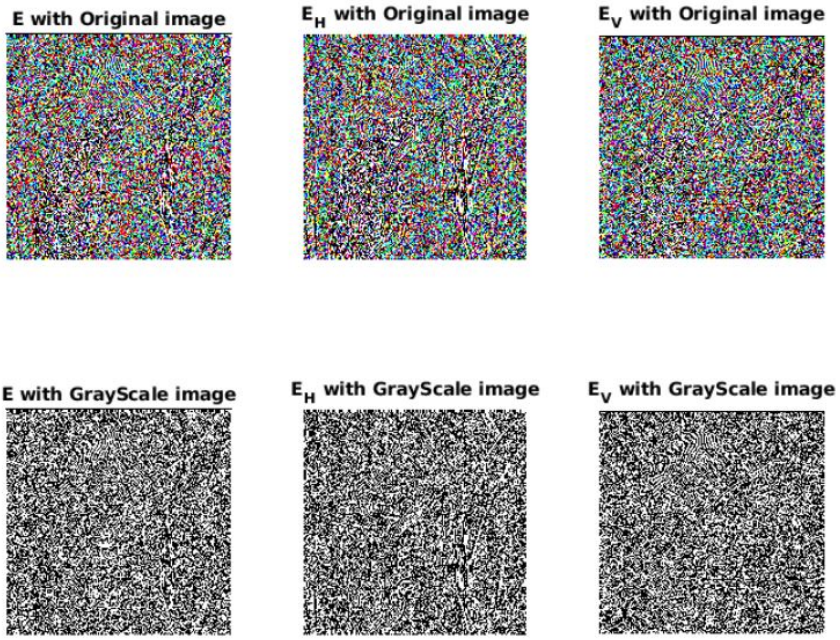
(a) Matlab Script:

```
clc;
clear all;
image=imread("lenna.png"); % reads an image file
imshow(image); % display the image in the graphics file
%scaling to gray
GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
imshow(GrayScale); % display the image in the graphics file

%Edge detection kernel
subplot(2,3,1);
E=zeros(3,3)
E(1,:)=[0 -1 0];
E(2,:)=[-1 4 -1];
E(3,:)=[0 -1 0];
con=convn(E,image)  %for N-Dimention conv
imshow(con)
title('E with Original image')
%E_h
subplot(2,3,2);
Eh=zeros(3,3)
Eh(1,:)=[0 0 0];
Eh(2,:)=[-1 2 -1];
Eh(3,:)=[0 0 0];
con=convn(Eh,image)  %for N-Dimention conv
imshow(con)
```

```
title('E_H with Original image')
%E_v
subplot(2,3,3);
Ev=zeros(3,3)
Ev(1,:)=[0 -1 0];
Ev(2,:)=[0 2 0];
Ev(3,:)=[0 -1 0];
con=convn(Ev,image)  %for N-Dimention conv
imshow(con)
title('E_V with Original image')
%with grayscale image
subplot(2,3,4);
con=convn(E,GrayScale)  %for N-Dimention conv
imshow(con)
title('E with GrayScale image')

subplot(2,3,5);
con=convn(Eh,GrayScale)  %for N-Dimention conv
imshow(con)
title('E_H with GrayScale image')

subplot(2,3,6);
con=convn(Ev,GrayScale)  %for N-Dimention conv
imshow(con)
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:

```
clc;
clear all;

```

```
image=imread("lenna.png"); % reads an image file
imshow(image); % display the image in the graphics file

% Scaling to gray
GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
imshow(GrayScale); % display the image in the graphics file

%Gradient 3x3 kernel
%G_H
subplot(2,2,1);
Gh=zeros(3,3)
Gh(1,:)=[-1 -1 -1];
Gh(2,:)=[0 0 0];
Gh(3,:)=[1 1 1];
con=convn(Gh,image)  %for N-Dimention conv
imshow(con)
title('G_H with Original image')
%with grayscale image
subplot(2,2,2);
con=convn(Gh,GrayScale)  %for N-Dimention conv
imshow(con)
title('G_H with GrayScale image')

%G_V
subplot(2,2,3);
Gv=zeros(3,3)
Gv(1,:)=[-1 0 1];
Gv(2,:)=[-1 0 1];
Gv(3,:)=[-1 0 1];
con=convn(Gv,image)  %for N-Dimention conv
imshow(con)
title('G_V with Original image')

subplot(2,2,4);
con=convn(Gv,GrayScale)  %for N-Dimention conv
imshow(con)
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:

Solution Problem-2b5 Sobel Operator 3x3 convolution kernel

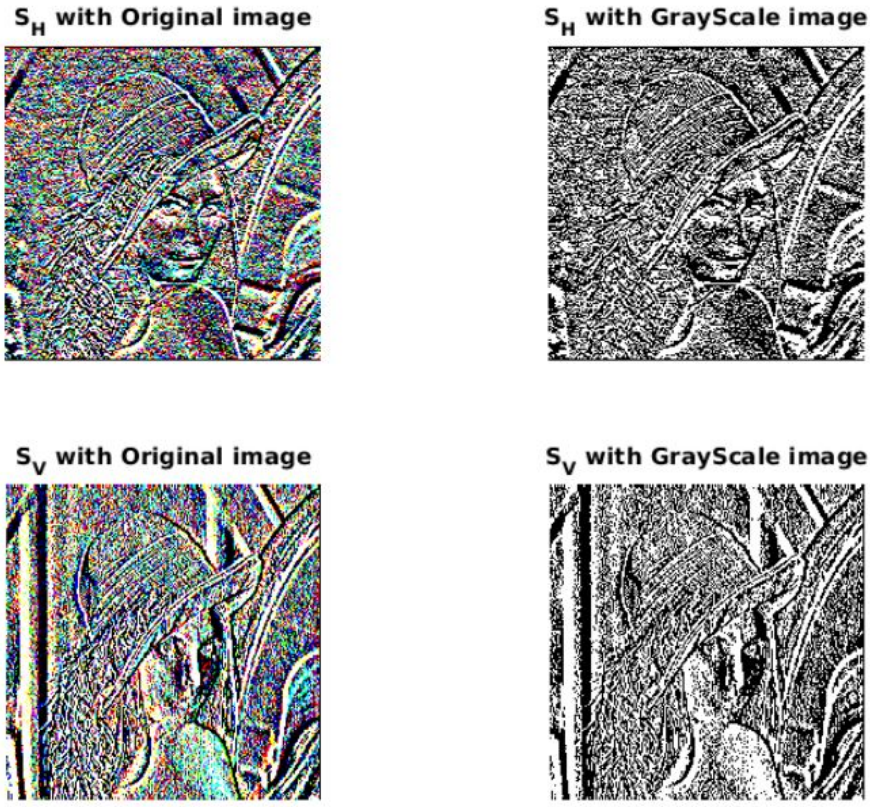
(a) Matlab Script:

```
clc;
clear all;

image=imread("lenna.png"); % reads an image file
imshow(image); % display the image in the graphics file

% Scaling to gray
GrayScale=rgb2gray(image); % The rgb2gray function converts RGB images to
    grayscale by eliminating the hue and saturation information while retaining
    the luminance
imshow(GrayScale); % display the image in the graphics file

%Sobel 3x3 kernel
%S_H
subplot(2,2,1);
Sh=zeros(3,3)
Sh(1,:)=[1 2 1];
Sh(2,:)=[0 0 0];
Sh(3,:)=[-1 -2 -1];
con=convn(Sh,image)  %for N-Dimention conv
imshow(con)
title('S_H with Original image')
%with grayscale image
subplot(2,2,2);
con=convn(Sh,GrayScale)  %for N-Dimention conv
imshow(con)
title('S_H with GrayScale image')

%S_V
subplot(2,2,3);
Sv=zeros(3,3)
Sv(1,:)=[1 0 -1];
```

```
Sv(2,:)=[2 0 -2];
Sv(3,:)=[1 0 -1];
con=convn(Sv,image)  %for N-Dimention conv
imshow(con)
title('S_V with Original image')

subplot(2,2,4);
con=convn(Sv,GrayScale)  %for N-Dimention conv
imshow(con)
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:



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

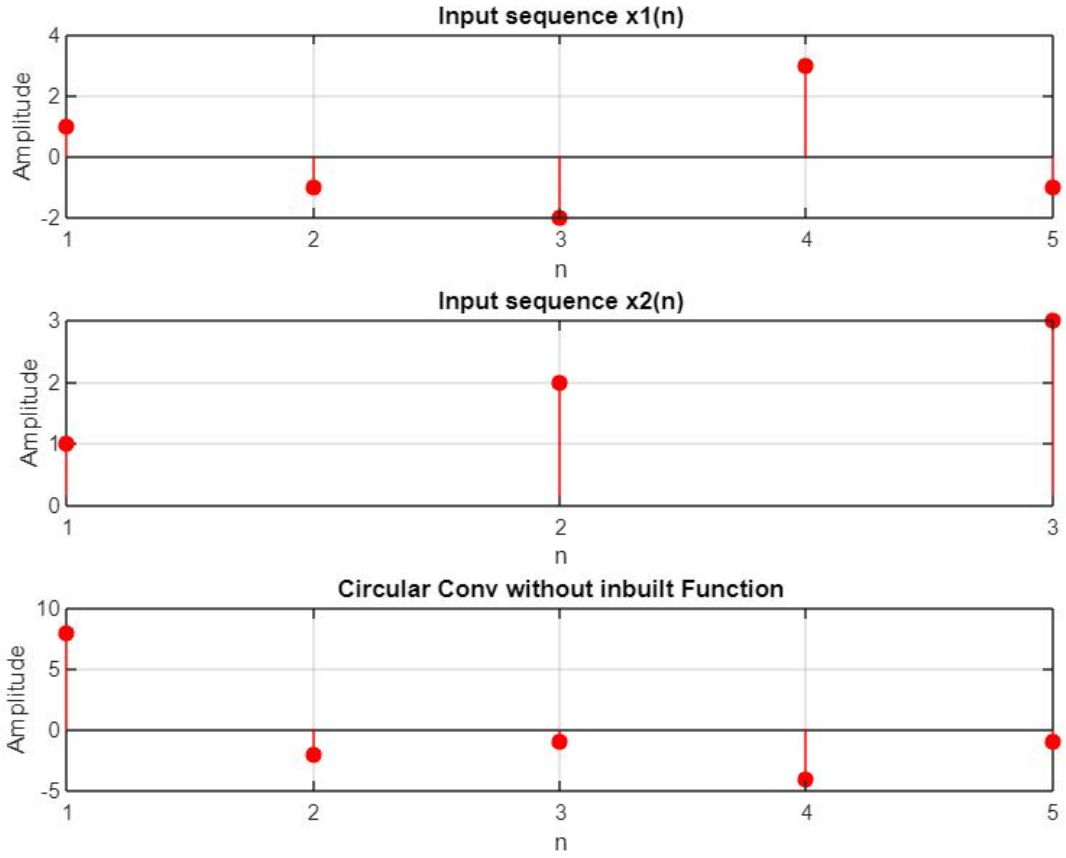
(a) Matlab Script:

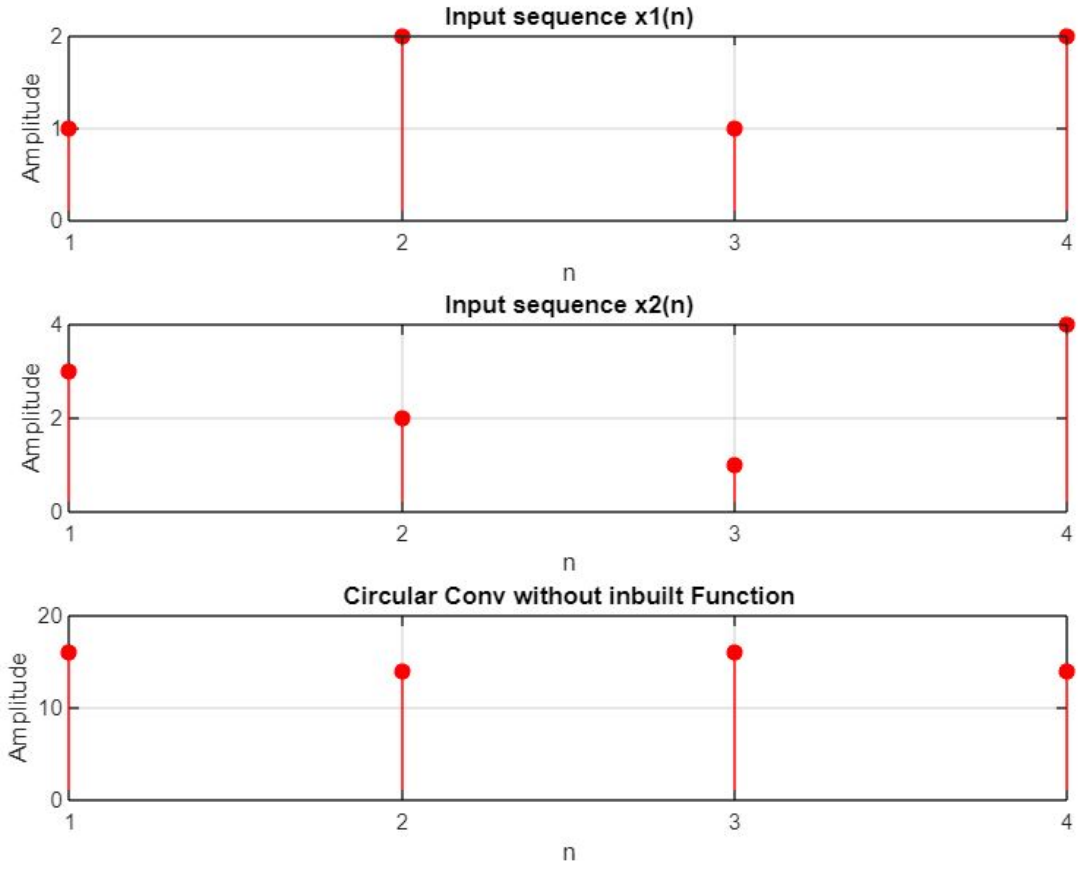
```
clc;
close all;
x1=input('Enter x1 :: ');
x2=input('Enter x2 :: ');

y_convofunc=circular_convo(x1,x2);

```

```
subplot(3,1,1);
stem(x1,'r');
hold on;
grid on;
title('Input sequence x1(n)');
xlabel('n');
ylabel('Amplitude');
xticks(1:1:length(x1));

subplot(3,1,2);
stem(x2,'r');
hold on;
grid on;
title('Input sequence x2(n)');
xlabel('n');
ylabel('Amplitude');
xticks(1:1:length(x2));

subplot(3,1,3);
stem(y_convofunc,'r');
hold on;
grid on;
title('Circular Conv without inbuilt Function');
xlabel('n');
ylabel('Amplitude');
xticks(1:1:length(y_convofunc));

%calculate
function ccov=circular_convo(x1, x2)
    if (length(x1) >= length(x2))
        x1padding=x1;
        x2padding=[x2 zeros(1, length(x1)-length(x2))];
    else
        x2padding=x2;
        x1padding=[x1 zeros(1, length(x2)-length(x1))];
    end
    ccov = ifft(fft(x1padding).*fft(x2padding));
    ccov = ccov(1:max(length(x1), length(x2)));
end

```

(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(length of x1) + L(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:


Input sequence x1(n)
Amplitude
n
Input sequence x2(n)
Amplitude
n
Circular Conv without inbuilt Function
Amplitude
n



Solution Problem- 3 c

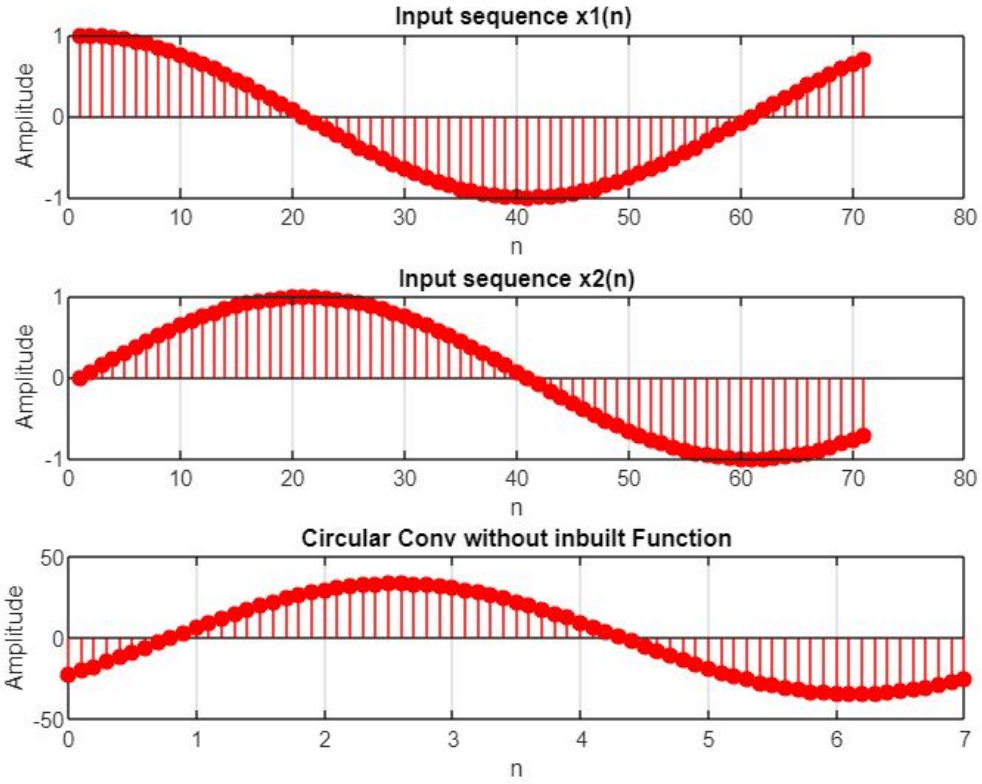
(a) Matlab Script:

```
clc;
close all;
N=8;
dn=0.1;
n=0:dn:N-1;

x1=cos((2*pi/N).*n);
x2=sin((2*pi/N).*n);

y_convofunc=circular_convo(x1,x2);

subplot(3,1,1);
stem(x1,'r');
hold on;
grid on;
title('Input sequence x1(n) ');
xlabel('n');
ylabel('Amplitude');

subplot(3,1,2);
stem(x2,'r');
hold on;
grid on;
title('Input sequence x2(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,1,3);
```

```
stem(n, y_convofunc,'r');
hold on;
grid on;
title('Circular Conv without inbuilt Function');
xlabel('n');
ylabel('Amplitude');


function ccov=circular_convo(x1, x2)
    if (length(x1) >= length(x2))
        x1padding=x1;
        x2padding=[x2 zeros(1, length(x1)-length(x2))];
    else
        x2padding=x2;
        x1padding=[x1 zeros(1, length(x2)-length(x1))];
    end
    ccov = ifft(fft(x1padding).*fft(x2padding));
    ccov = ccov(1:max(length(x1), length(x2)));
end

```

(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.For the circular convolution of x and y should be same length, the arrays must be padded with zeros to length min N(length of x1) + L(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 c:



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

Multiplication method.

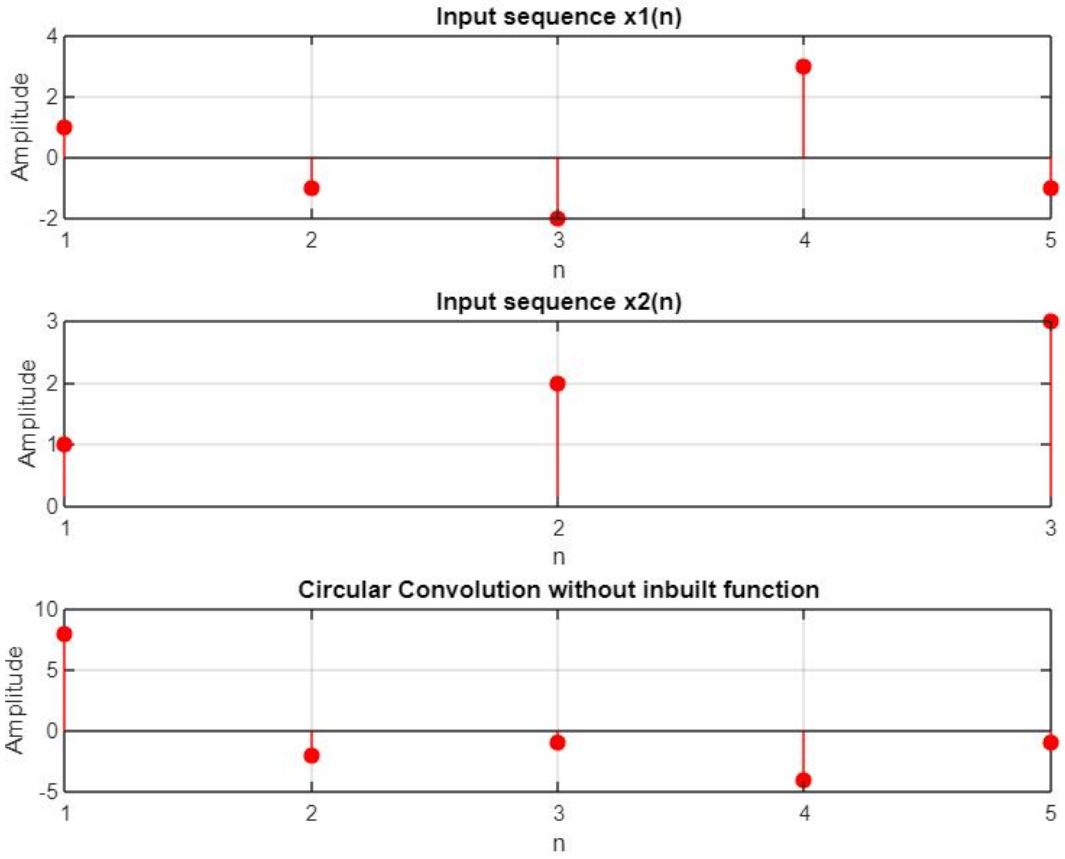
Solution Problem- 4 a & b

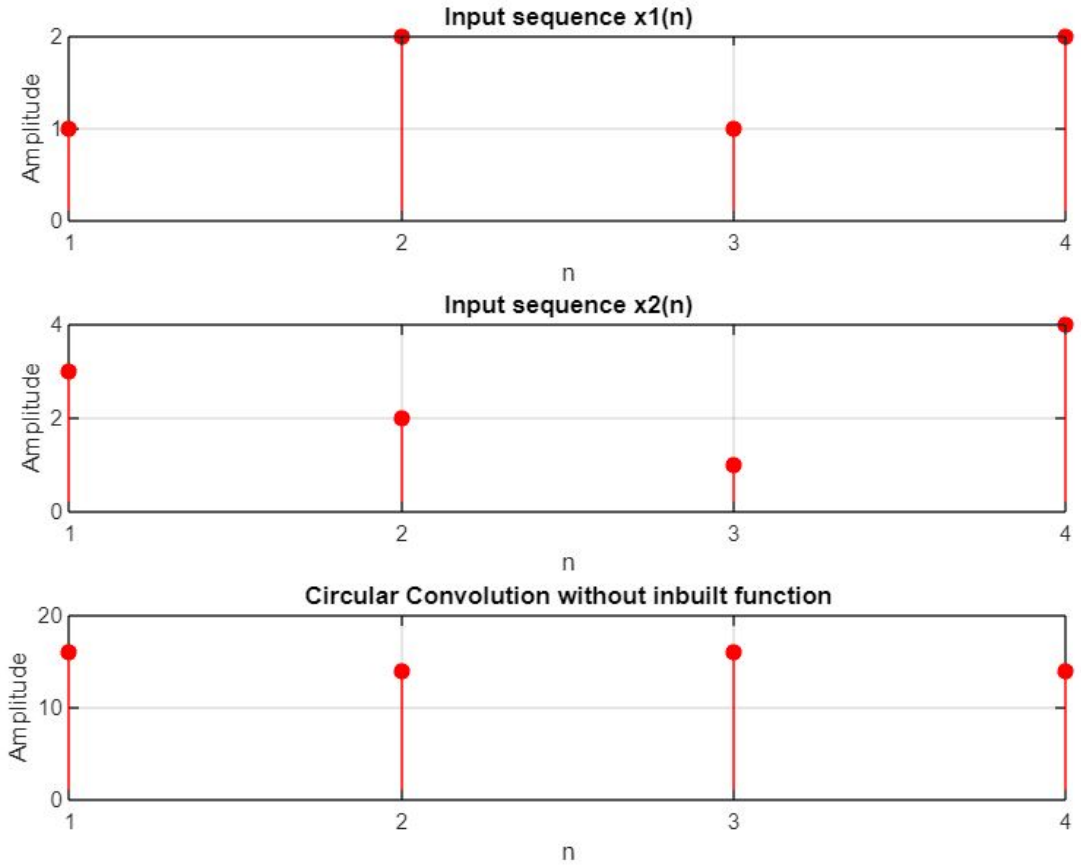
(a) Matlab Script:

```
  clc;
close all;
x1=input('Enter x1 :: ');
x2=input('Enter x2 :: ');

if (length(x1) >= length(x2))
    [y_convofunc, matrix, x3]=circular_convo(x1,x2);
else
    [y_convofunc, matrix, x3]=circular_convo(x2,x1);
end

subplot(3,1,1);
stem(x1,'r');
hold on;
grid on;
title('Input sequence x1(n)');
xlabel('n');
ylabel('Amplitude');
xticks(1:1:length(x1));

subplot(3,1,2);
stem(x2,'r');
hold on;
grid on;
title('Input sequence x2(n)');
xlabel('n');
ylabel('Amplitude');
xticks(1:1:length(x2));

subplot(3,1,3);
stem(y_convofunc,'r');
hold on;
grid on;
title('Circular Convolution without inbuilt function');
xlabel('n');
ylabel('Amplitude');
xticks(1:1:length(y_convofunc));


function [ccov, matrix, x3]=circular_convo(x1, x2)
    matrix=zeros(length(x1),length(x1));
    matrix(:, 1)=x1;
    for count=2:length(x1)
       arr=circshift(matrix(:, count-1), 1);
       matrix(:,count)=arr;
    end
    x3=zeros(1, length(x1));
    x3(1,1:length(x2))=x2;
    ccov=transpose(matrix*transpose(x3));
end

```

(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:


Input sequence x1(n)
Amplitude
n
Input sequence x2(n)
Amplitude
n
Circular Convolution without inbuilt function
Amplitude
n



Solution Problem- 4 c

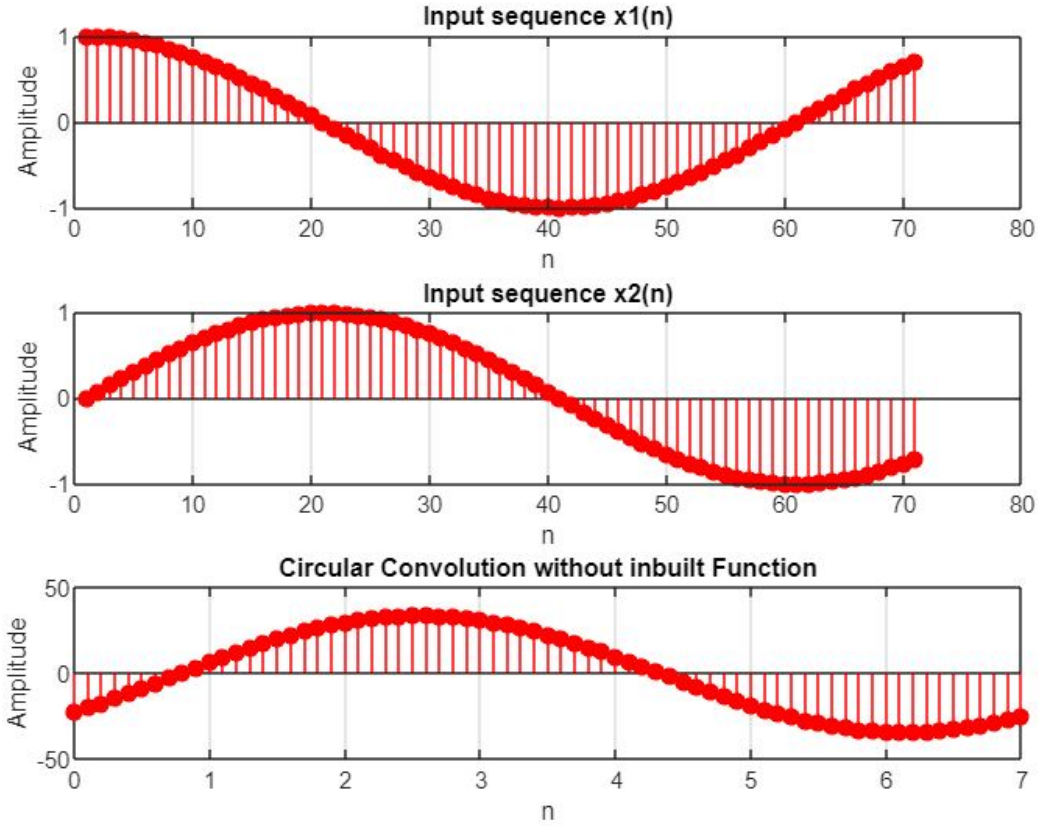
(a) Matlab Script:

```
clc;
close all;

N=8;
dn=0.1;
n=0:dn:N-1;

x1=cos((2*pi/N).*n);
x2=sin((2*pi/N).*n);

if (length(x1) >= length(x2))
    [y_convofunc, matrix, x3]=circular_convo(x1,x2);
else
    [y_convofunc, matrix, x3]=circular_convo(x2,x1);
end

subplot(3,1,1);
stem(x1,'r');
hold on;
grid on;
title('Input sequence x1(n)');
xlabel('n');
ylabel('Amplitude');

subplot(3,1,2);
stem(x2,'r');
hold on;
grid on;
title('Input sequence x2(n)');
```

```
xlabel('n');
ylabel('Amplitude');

subplot(3,1,3);
stem(n, y_convofunc,'r');
hold on;
grid on;
title('Circular Convolution without inbuilt Function');
xlabel('n');
ylabel('Amplitude');


function [ccov, matrix, x3]=circular_convo(x1, x2)
    matrix=zeros(length(x1),length(x1));
    matrix(:, 1)=x1;
    for count=2:length(x1)
       arr=circshift(matrix(:, count-1), 1);
       matrix(:,count)=arr;
    end
    x3=zeros(1, length(x1));
    x3(1,1:length(x2))=x2;
    ccov=transpose(matrix*transpose(x3));
end

```

(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:


Input sequence x1(n)
Amplitude
n
Input sequence x2(n)
Amplitude
n
Circular Convolution without inbuilt Function
Amplitude
n