# Digital Signal Processing Lab

## Name: Deep C. Patel

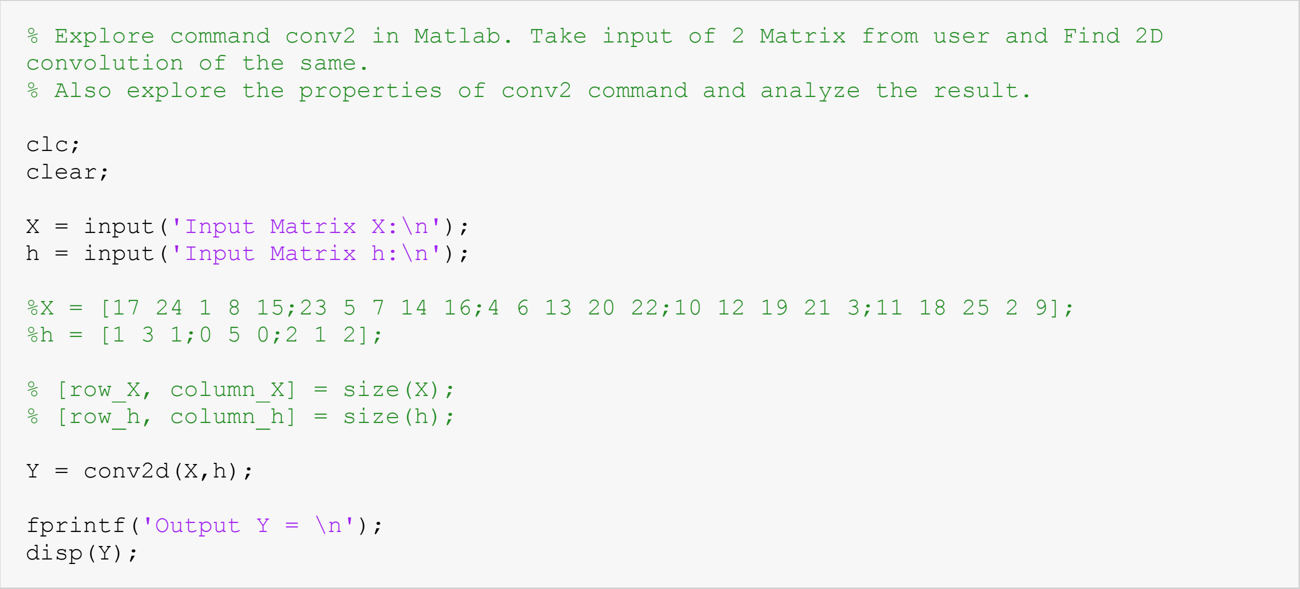
## Roll No: 1401010

# Lab Report

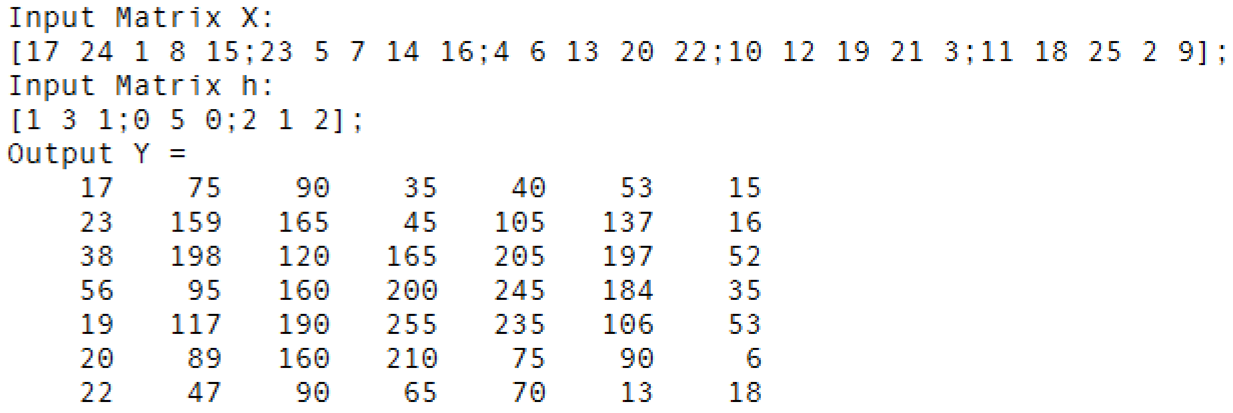
## Lab Work:-

## Lab – 3

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.

clc;
clear;

X = input('Input Matrix X:\n');
h = input('Input Matrix h:\n');

%X = [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];
%h = [1 3 1;0 5 0;2 1 2];

% [row_X, column_X] = size(X);
% [row_h, column_h] = size(h);

Y = conv2d(X,h);

fprintf('Output Y = \n');
disp(Y);
```



```
Input Matrix X:
[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];
Input Matrix h:
[1 3 1;0 5 0;2 1 2];
Output Y =
    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
```

## CONV2D.M

```
% Custom 2D Convolution Function

function Y = conv2d(X,h)

    [row_X, column_X] = size(X);
    [row_h, column_h] = size(h);

    n1 = 0:1:column_X;
    n2 = 0:1:column_h;

    row_Y = row_X + row_h - 1;
    column_Y = column_X + column_h - 1;

    Y = zeros(row_Y, column_Y);

    % Flipping and Padding h

    h = flip(h);

    h = [zeros(row_X-1,column_h);h];
    new_r_h = size(h,1);

    i=0;

    while i ~= column_Y

        if(i >= row_X)
            k = row_X - 1;
            new_r_h = new_r_h - 1;
        else
            k = i;
        end

        j = new_r_h;

        for n = k+1:-1:1

            Y(i+1,:) = Y(i+1,:) + convi(X(n,:),h(j,:));
            j = j - 1;

        end

        i = i + 1;

    end
end
```

# CONVI.M

```
% 1D Convolution Function

function [y, n] = convi(x, n1, h, n2)

    lenX = length(x);
    lenH = length(h);

    h = flip(h);

    y = zeros(1,lenH+lenX-1);

    h = [zeros(1,lenX) h zeros(1,lenX-1)];
    newlenH = length(h);

    m = lenH+lenX;
    i=1;

    while m ~= 1
        y(1,i) = sum(x.*h(1,m:newlenH-(i-1)));

        m = m-1;
        i = i+1;
    end

    c = min(n2) + min(n1);
    n = c:1:c+length(y)-1;
end
```

# 2).

## A

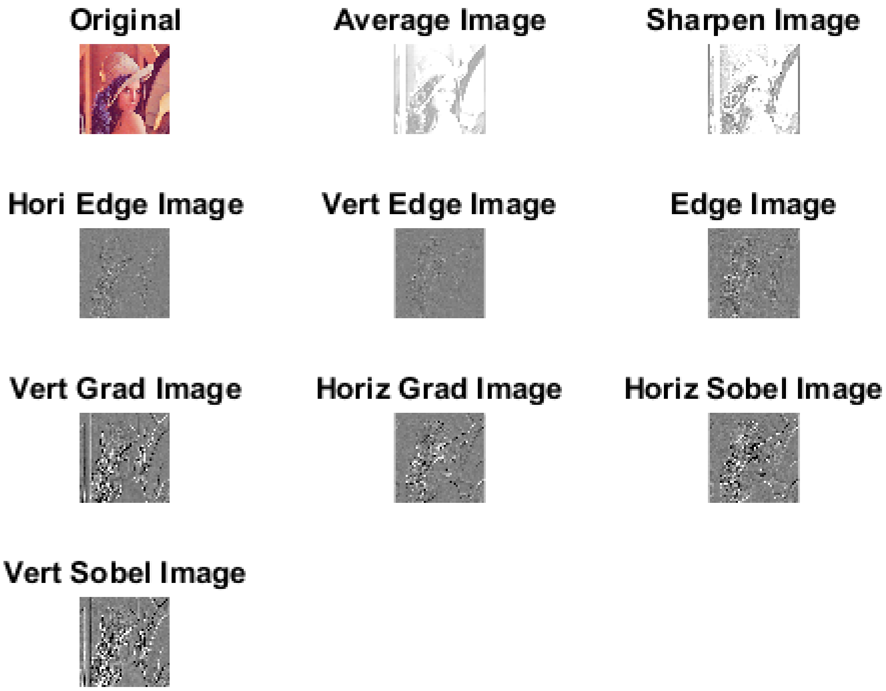
```
% Application of 2D convolution on image processing applications
% (A).Take a standard test image “lenna.png” from shared folder.
% Explore following commands and apply for given image.
% -> imread
% -> rgb2gray
% -> imshow

clc;
clear;

image = imread('lenna.png');
image = rgb2gray(image);

figure;
imshow(image);
```

## B

```
% Application of 2D convolution on image processing applications
% (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


clc;
clear;

I = imread('lenna.png');
I_gray = rgb2gray(I);
image = double(I_gray);

average = [1/9 1/9 1/9;...
           1/9 1/9 1/9;...
           1/9 1/9 1/9];

sharp = [0 -1 0;...
         -1 5 -1;...
         0 -1 0];

edge_detection = [0 -1 0;...
                  -1 4 -1;...
                  0 -1 0];
```

```
edge_detection_horizontal = [0 0 0;...
                             -1 2 -1;...
                             0 0 0];

edge_detection_vertical = [0 -1 0;...
                           0 2 0;...
                           0 -1 0];

gradient_horizontal = [-1 -1 -1;...
                       0 0 0;...
                       1 1 1];

gradient_vertical = [-1 0 1;...
                     -1 0 1;...
                     -1 0 1];

sobel_horizontal = [1 2 1;...
                       0 0 0;...
                       -1 -2 -1];

sobel_vertical = [1 0 -1;...
                     2 0 -2;...
                     1 0 -1];

image_average = conv2d(image,average);
image_sharp = conv2d(image,sharp);
image_edge_detection_horiz = conv2d(image,edge_detection_horizontal);
image_edge_detection_vert = conv2d(image,edge_detection_vertical);
image_edge_detection = conv2d(image,edge_detection);
image_gradient_vert = conv2d(image,gradient_vertical);
image_gradient_horiz = conv2d(image,gradient_horizontal);
image_sobel_horiz = conv2d(image,sobel_horizontal);
image_sobel_vert = conv2(image,sobel_vertical);

figure;

subplot(4,3,1);
imshow(I);
title('Original');

subplot(4,3,2);
imshow(int8(image_average));
title('Average Image');

subplot(4,3,3);
imshow(int8(image_sharp));
title('Sharpen Image');

subplot(4,3,4);
imshow(int8(image_edge_detection_horiz));
```

```
title('Hori Edge Image');

subplot(4,3,5);
imshow(int8(image_edge_detection_vert));
title('Vert Edge Image');

subplot(4,3,6);
imshow(int8(image_edge_detection));
title('Edge Image');

subplot(4,3,7);
imshow(int8(image_gradient_vert));
title('Vert Grad Image');

subplot(4,3,8);
imshow(int8(image_gradient_horiz));
title('Horiz Grad Image');

subplot(4,3,9);
imshow(int8(image_sobel_horiz));
title('Horiz Sobel Image');

subplot(4,3,10);
imshow(int8(image_sobel_vert));
title('Vert Sobel Image');
```



Original

Average Image

Sharpen Image

Hori Edge Image

Vert Edge Image

Edge Image

Vert Grad Image

Horiz Grad Image

Horiz Sobel Image

Vert Sobel Image

## CONV2D.M

```
% Custom 2D Convolution Function

function Y = conv2d(X,h)

    [row_X, column_X] = size(X);
    [row_h, column_h] = size(h);

    n1 = 0:1:column_X;
    n2 = 0:1:column_h;

    row_Y = row_X + row_h - 1;
    column_Y = column_X + column_h - 1;

    Y = zeros(row_Y, column_Y);

    % Flipping and Padding h

    h = flip(h);

    h = [zeros(row_X-1,column_h);h];
    new_r_h = size(h,1);

    i=0;

    while i ~= column_Y

        if(i >= row_X)
            k = row_X - 1;
            new_r_h = new_r_h - 1;
        else
            k = i;
        end

        j = new_r_h;

        for n = k+1:-1:1

            Y(i+1,:) = Y(i+1,:) + convi(X(n,:),h(j,:));
            j = j - 1;

        end

        i = i + 1;

    end
end
```

## CONVI.M

```
% 1D Convolution Function

function [y, n] = convi(x, n1, h, n2)

    lenX = length(x);
    lenH = length(h);

    h = flip(h);

    y = zeros(1,lenH+lenX-1);

    h = [zeros(1,lenX) h zeros(1,lenX-1)];
    newlenH = length(h);

    m = lenH+lenX;
    i=1;

    while m ~= 1
        y(1,i) = sum(x.*h(1,m:newlenH-(i-1)));

        m = m-1;
        i = i+1;
    end

    c = min(n2) + min(n1);
    n = c:1:c+length(y)-1;
end
```

## 3).

```
% Develop a MATLAB function to obtain circular convolution of two sequences.
% Use definition of circular convolution to obtain the output sequence.
% Verify the function for following sequence. Write a script file to use
% the developed function.

clc;
clear;

x1 = [1,-1,-2,3,-1];
h1 = [1,2,3];

y1 = circ_conv(x1,h1);
y1mat = cconv(x1,h1);
```

```
x2 = [1, 2, 1, 2];
h2 = [3,2,1,4];

y2 = circ_conv(x2,h2);
y2mat = cconv(x2,h2,length(x2));

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

x3 = cos(2*pi*n/N);
h3 = sin(2*pi*n/N);

y3 = circ_conv(x3,h3);
y3mat = cconv(x3,h3,length(x3));

fprintf('Y1             = ');
disp(y1);

fprintf('Y1 by Inbuilt = ');
disp(y1mat);

fprintf('Y2             = ');
disp(y2);

fprintf('Y2 by Inbuilt = ');
disp(y2mat);

fprintf('Y3             = ');
disp(y3);


fprintf('Y3 by Inbuilt = ');
disp(y3mat);
```

```
Y1             =      8    -2    -1    -4    -1


Y1 by Inbuilt =     1.0000    1.0000   -1.0000   -4.0000   -1.0000    7.0000   -3.0000


Y2             =     16    14    16    14


Y2 by Inbuilt =     16    14    16    14
```

```
Y3           =

-0.0000    2.8284    4.0000    2.8284    0.0000   -2.8284   -4.0000   -2.8284



Y3 by Inbuilt =

-0.0000    2.8284    4.0000    2.8284    0.0000   -2.8284   -4.0000   -2.8284
```

## CIRC_CONV.M

```
% Circular Convolution

function y = circ_conv(x, h)

    lenX = length(x);
    lenH = length(h);

    if lenX == lenH

        y = zeros(1,lenX);
        h = flip(h);
        for i = 1:lenX
            y(1,i) = sum(x.*h);
            h = circshift(h,1,2);
        end

        y = circshift(y,length(y)-1,2);

    elseif lenX>lenH

        h = [h zeros(1,lenX-lenH)];

        y = zeros(1,lenX);
        h = flip(h);
        for i = 1:lenX
            y(1,i) = sum(x.*h);
            h = circshift(h,1,2);
        end

        y = circshift(y,length(y)-1,2);

    else

        x = [x zeros(1,lenH-lenX)];

        y = zeros(1,lenH);
        h = flip(h);
        for i = 1:lenH
            y(1,i) = sum(x.*h);
            h = circshift(h,1,2);
        end

        y = circshift(y,length(y)-1,2);
    end
end
```

4).

```
% Write a MATLAB program to find circular convolution of two sequences using Matrix
% Multiplication method.

clc;
clear;

x1 = [1,-1,-2,3,-1];
h1 = [1,2,3];

y1 = circ_conv_matrix(x1,h1);
y1mat = cconv(x1,h1);

x2 = [1, 2, 1, 2];
h2 = [3, 2, 1, 4];

y2 = circ_conv_matrix(x2,h2);
y2mat = cconv(x2,h2,length(x2));

N = 8;

n = 0:1:N-1;

x3 = cos(2*pi*n/N);
h3 = sin(2*pi*n/N);

y3 = circ_conv_matrix(x3,h3);
y3mat = cconv(x3,h3,length(x3));

fprintf('Y1            = ');
disp(y1);

fprintf('Y1 by Inbuilt = ');
disp(y1mat);

fprintf('Y2            = ');
disp(y2);

fprintf('Y2 by Inbuilt = ');
disp(y2mat);
```

```
fprintf('Y3            = ');
disp(y3);

fprintf('Y3 by Inbuilt = ');
disp(y3mat);
```

```
Y1            =      8    -2    -1    -4    -1


Y1 by Inbuilt =     1.0000    1.0000   -1.0000   -4.0000   -1.0000    7.0000   -3.0000


Y2            =     16    14    16    14


Y2 by Inbuilt =     16    14    16    14


Y3            =

 -0.0000    2.8284    4.0000    2.8284    0.0000   -2.8284   -4.0000   -2.8284


Y3 by Inbuilt =

 -0.0000    2.8284    4.0000    2.8284    0.0000   -2.8284   -4.0000   -2.8284
```

## CIRC_CONV_MATRIX.M

```
% Circular Convolution using Matrix Method

function y = circ_conv_matrix(x, h)

    lenX = length(x);
    lenH = length(h);

    if lenX == lenH

        x = [x(1,1) flip(x(1,2:end))];
        c = zeros(lenX,lenX);

        for i = 1:lenX
            c(i,:) = x;
            x = circshift(x,1,2);
        end

        % disp(c);

        y = c*h';
        y = y';

    elseif lenX>lenH

        h = [h zeros(1,lenX-lenH)];

        x = [x(1,1) flip(x(1,2:end))];

        c = zeros(lenX,lenX);

        for i = 1:lenX
            c(i,:) = x;
            x = circshift(x,1,2);
        end

        % disp(c);

        y = c*h';
        y = y';

    else

        x = [x zeros(1,lenH-lenX)];

        x = [x(1,1) flip(x(1,2:end))];

        c = zeros(lenX,lenX);

        for i = 1:lenX
            c(i,:) = x;
            x = circshift(x,1,2);
        end

        % disp(c);

        y = c*h';
        y = y';
    end
end
```

**-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-x-**