Image Convolution and FFT

Problem 1a) 2-D Convolution of Image with Kernel:

The 2D convolution function has been implemented in Matlab with image, kernel, padding type as input arguments. Padding type is chosen from a list of 1 to 4 as

1.Clip/zero 2.Copy Edge 3.Reflect across edge 4.Wrap Around

The function call along with input is given by following code:

```
clc; clear
all; close
all;

%inputs to the function
f=imread('C:\Users\venkatesh\Desktop\wolves.png'); w=input('Enter
kernal values:'); pad=input('Enter type of
padding:\n1.Zero/Clip\n2.wrap around\n3.Copy edge\n4.Reflect across
edge');

%function call
g=myconvolve_2D(f,w,pad);

%display image
imshow(g,'InitialMagnification','fit'); title('Filtered
Image');
```

(Above code uses RGB image as input, hence output is an filtered RGB image. To give greyscale image as input we use: f=rgb@gray(f); before passing to the function and output is a filtered grayscale image.) Giving Input :

```
Enter kernal values:[1/9 1/9 1/9;1/9 1/9 1/9;1/9 1/9 1/9]
Enter type of padding:
1.Zero/Clip
2.Copy edge
3.Reflect across edge
4.wrap aroundl
>>
```

Visualization of Zero padding:

To visualize the effect of kinds of padding, let us consider image as a small matrix eg. [1 2 3 4;1 2 3 4;1 2 3 4;1 2 3 4]. The following are the matrices after padding and before convolution.

Zero/Clip: (input 1)

0	0	0	0	0	0
0	1	2	3	4	0
0	1	2	3	4	0
0	1	2	3	4	0
0	1	2	3	4	0
0	0	0	0	0	0

Copy edge: (input 2)

1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4

Reflect across edge: (input 3)

1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4
1	1	2	3	4	4

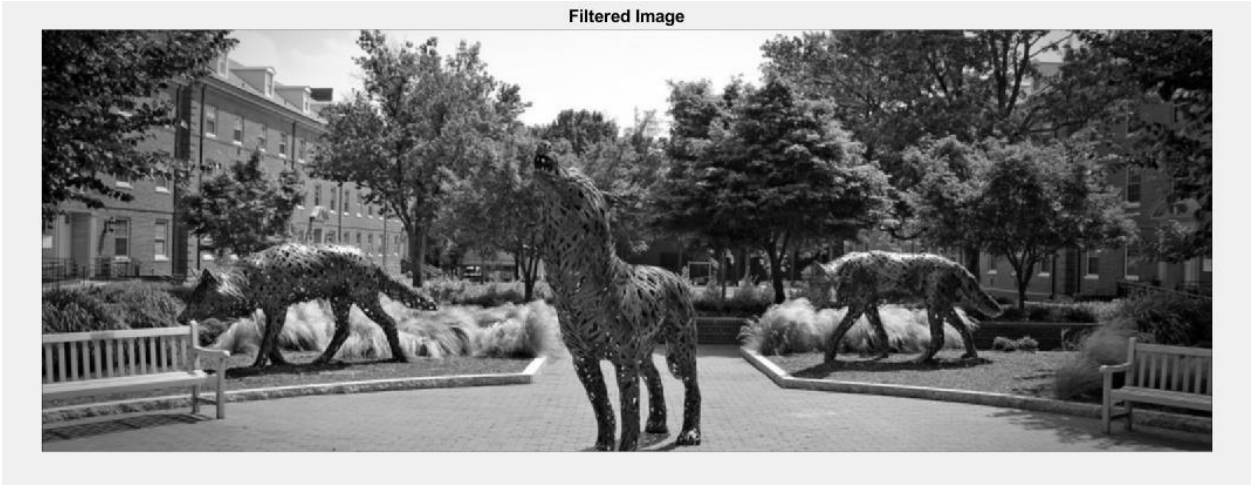
Wrap around: (input 4)

4	1	2	3	4	1
4	1	2	3	4	1
4	1	2	3	4	1
4	1	2	3	4	1
4	1	2	3	4	1
4	1	2	3	4	1

Note: for given 3x3 input, copy and reflect are similar Filtering Results :

Following are the sample filtering results for the different standard filters for the given 2 images as grayscale images.(considering clip/zero padding.

a.) Box filter or blur filter=[1/9 1/9 1/9; 1/9 1/9 1/9; 1/9 1/9 1/9]



b.) 1st order derivative filter=[-1 1] in terms of 3x3, [0 0 0; -1 1 0; 0 0 0]

c.) Prewitt Filter My= [1 1 1; 0 0 0; -1 -1 -1]

1st order derivative filter [-1; 1] in terms of 3x3, [0 -1 0; 0 1 0; 0 0 0]

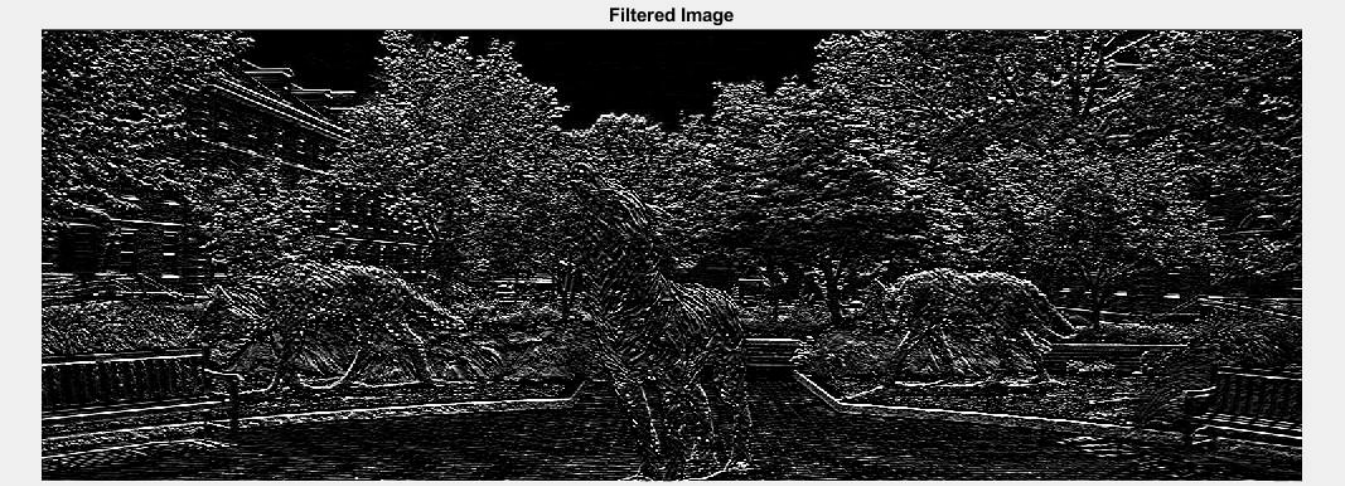
d.) Prewitt Filter Mz=[-1 0 1; -1 0 1; -1 0 1]

e.) Prewitt Filter My= [1 1 1; 0 0 0; -1 -1 -1]

f.) Sobel Filter Mx=[-1 0 1; -2 0 2; -1 0 1]

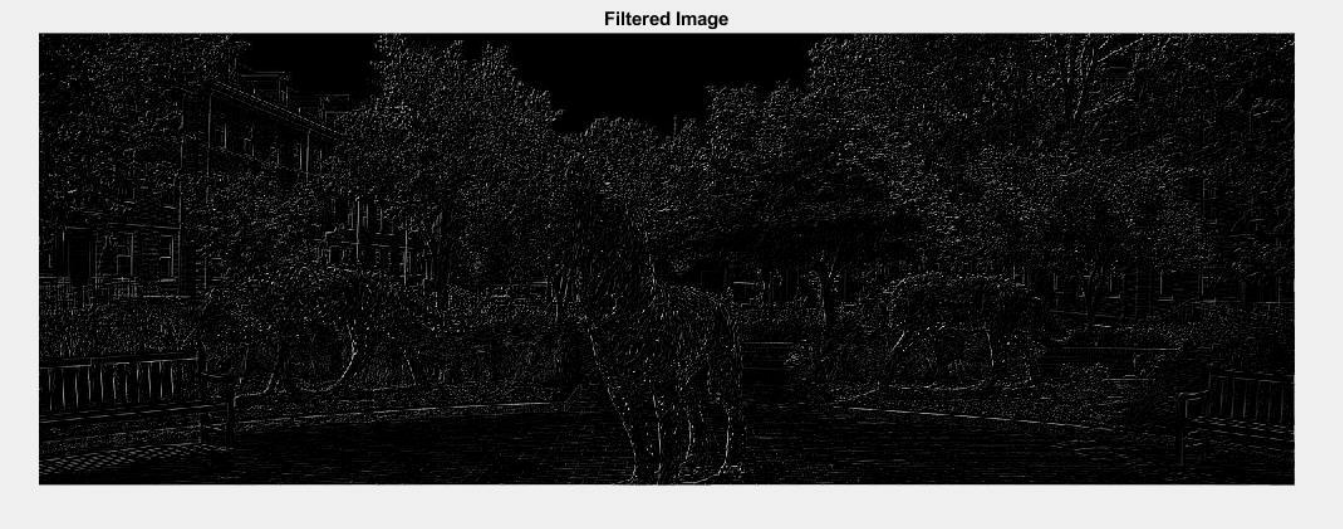
Sobel filter My=[1 2 1; 0 0 0; -1 -2 -1]

g.) Prewitt Filter My= [1 1 1; 0 0 0; -1 -1 -1]

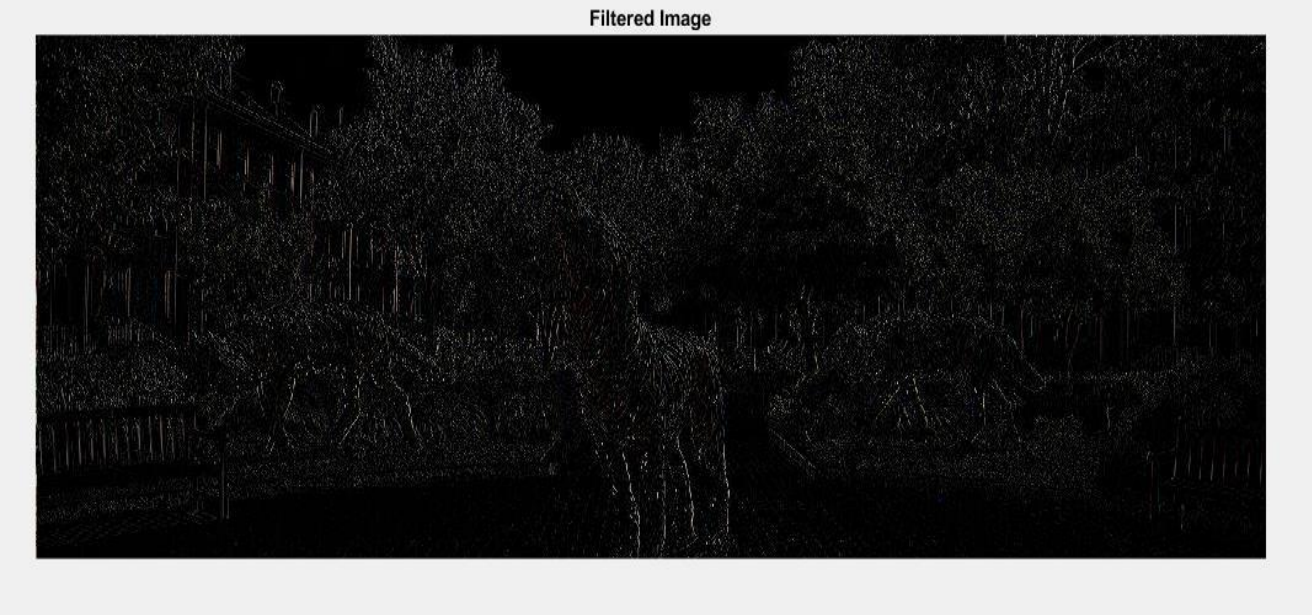


h.) Roberts Filter Mx=[0 1; -1 0] in terms of 3x3=[0 0 0; 0 1 0; -1 0 0]

i.) Roberts Filter My=[1 0; 0 -1] in terms of 3x3=[0 0 0; 0 1 0; 0 0 -1]



The code has been designed for both grey scale and rgb images hence following is an example of derivative filter on the 2 images. Other filters can also be implemented in the same way



Problem 1b) Testing the Convolution

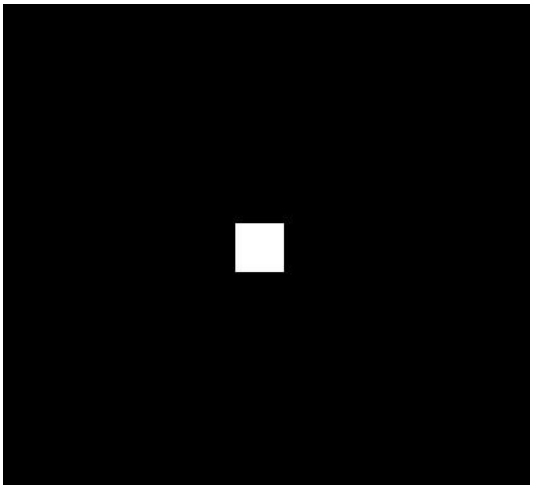
Creating test image:

```
check=zeros(1024,1024); check(512,512)=255;
```

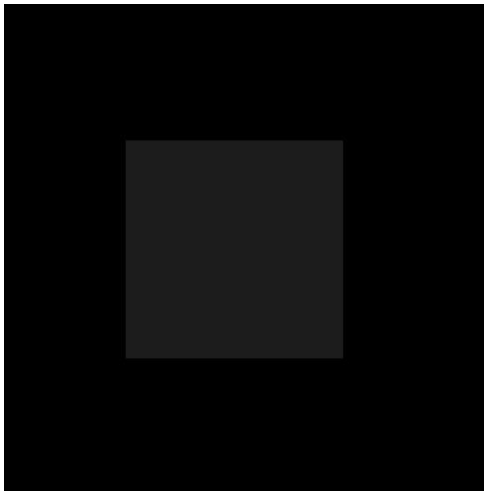
Checking convolution with blur filter:

```
w=input('Enter kernal values:'); pad=input('Enter type of
padding:\n1.Zero/Clip\n2.Copy edge\n3.Reflect across edge\n4.wrap around');
g=imfilter(mat2gray(check),w,0); imshow(g);
```

The input image is a blank image with a white spot at the centre(single pixel), after applying blur filter we get a gray patch in the centre(square region). the zoomed centre region is given here for visibility.



Before filtering



After filtering

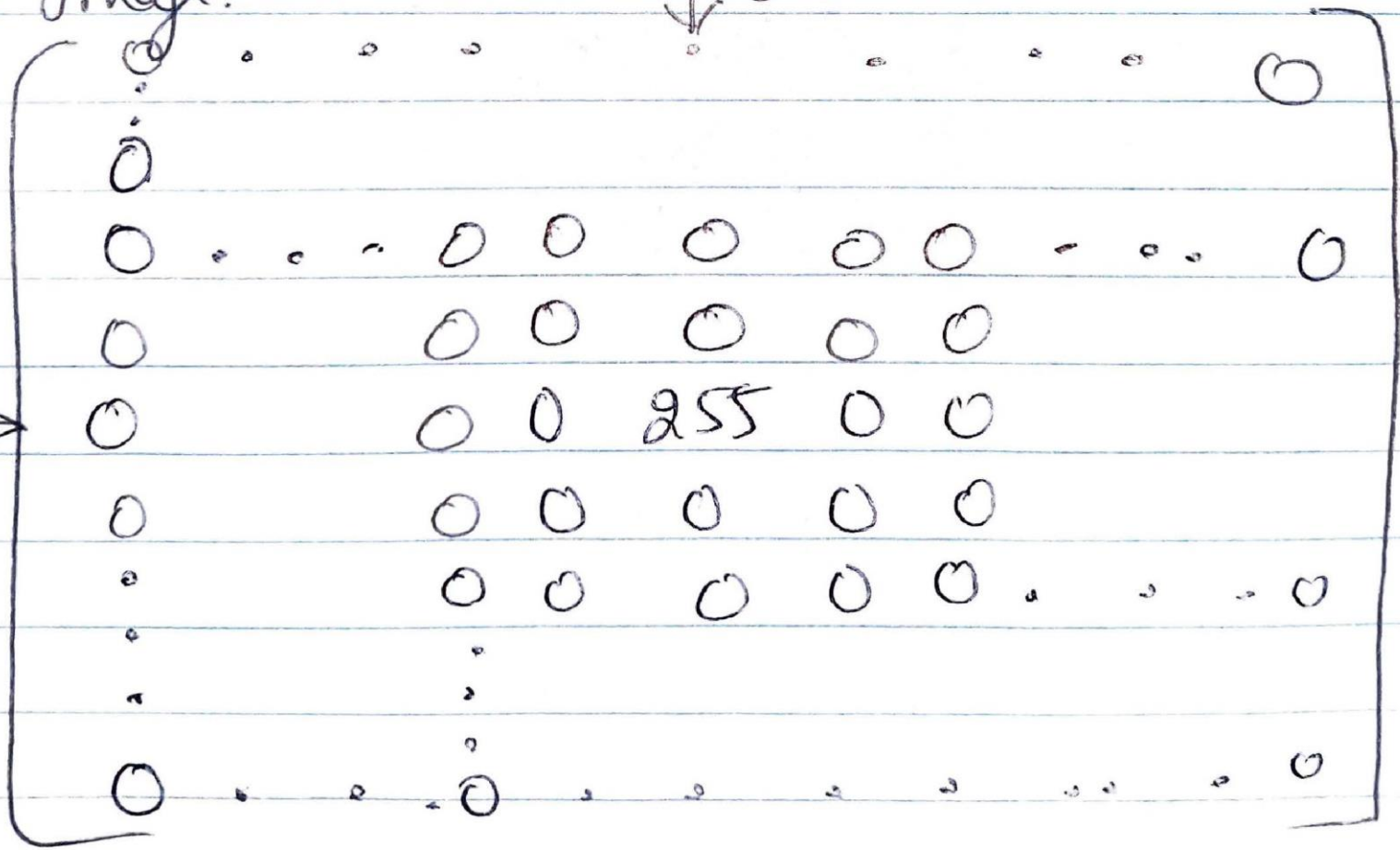
(Note: Zoomed image, not the original, original is attached as matlab fig file)

The explanation for blurring in the centre region is given as follows . Hence we see that the code performs convolution based on our requirements.

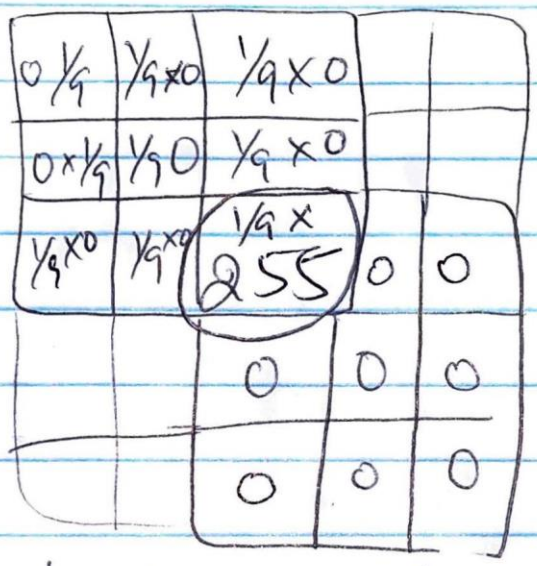
Convolution with central Impulse:

Consider Box filter : $[1/9, 1/9, 1/9; 1/9, 1/9, 1/9; 1/9, 1/9, 1/9]$

Test Image:



After convolution, only the neighbours of 255 are affected as other elements are zero.



This kind of operation happens for all 8 neighbours of 255 and the value becomes $255/9 = 28.33$ for box filter

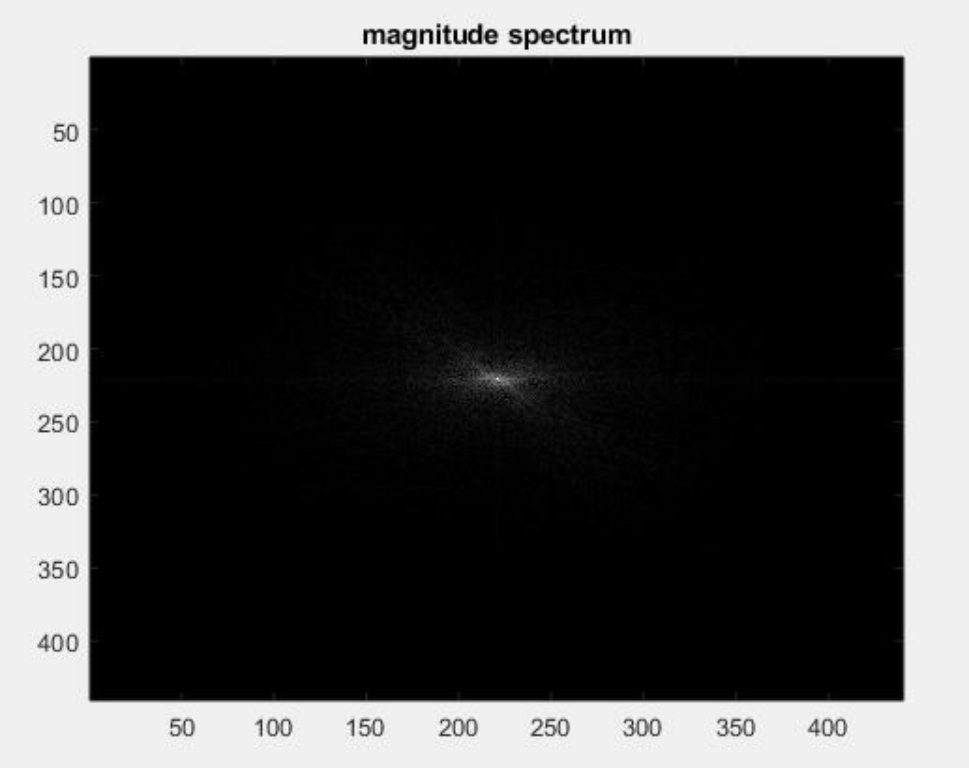
0	0	.	.	.	.	0
.	0	0	0	0	0	.
.	0	28.33	28.33	28.33	0	.
.	0	28.33	28.33	28.33	0	.
.	0	28.33	28.33	28.33	0	.
.	0	0	0	0	0	.
0	.	0	.	.	.	0

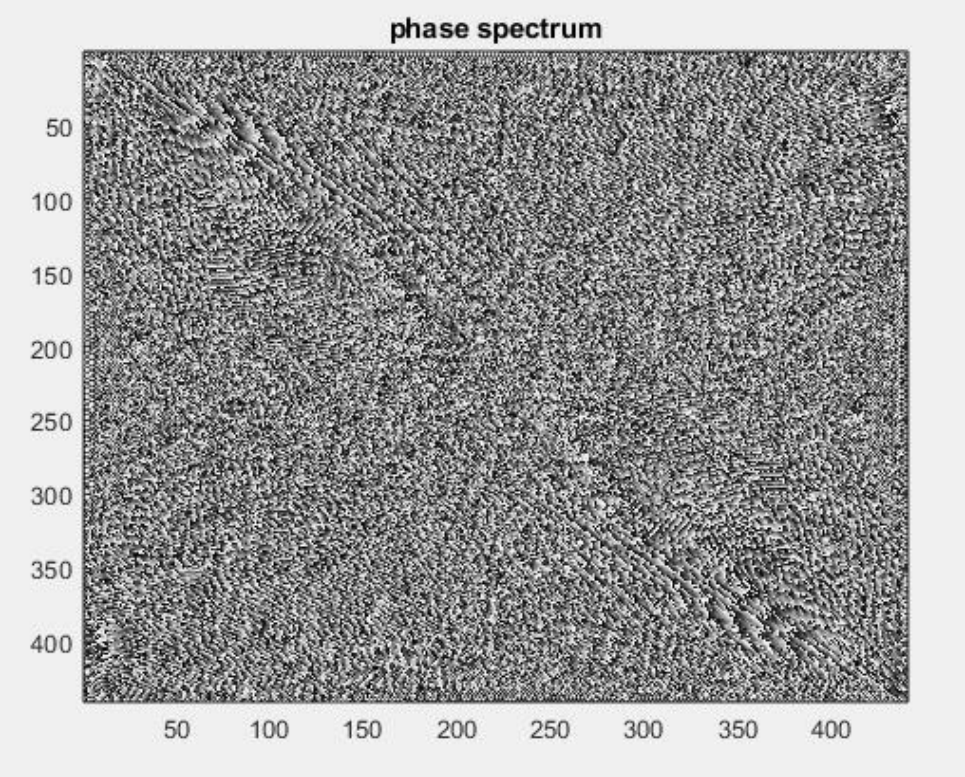
1024 x 1024

Problem2a) Implementation of 2D fft

- For 1D FFT we use the fft() function in Matlab
- To obtain 2D fft we take 1D fft for individual rows of the given matrix and then perform 1D fft column wise for the resultant matrix which is implemented in the attached code.
- The code is implemented using wolves.png an lena.png. to convert to grey image rgb2gray() function is used.
- Before performing dft we scale the image to [0,1]. This is perform using the function new=(L/255)*old. Here L=1, hence we divide image matrix by 255.
- To plot magnitude spectrum, we perform fftshift, transformation log(1+ abs(F)) and plot in 2D using imagesc() function.

Results for lena.png:





b.) Implementing Inverse DFT:

To obtain inverse dft using the 2D-DFT function,

- We pass conjugate of obtained fft to the function.
- Divide the result by product M,N. Where M,N are dimensions of the matrix.

Checking the results:

To check the working of the function, we find 2D-DFT and 2-D IDFT and we get the same image as input to fft.

(note: we multiply the idft by 255 to get original image as it was scaled to[0,1])

We can also check the results by taking difference of the original image and image after dft and idft, we obtain a black image since they are identical and difference is 0.This is shown below: