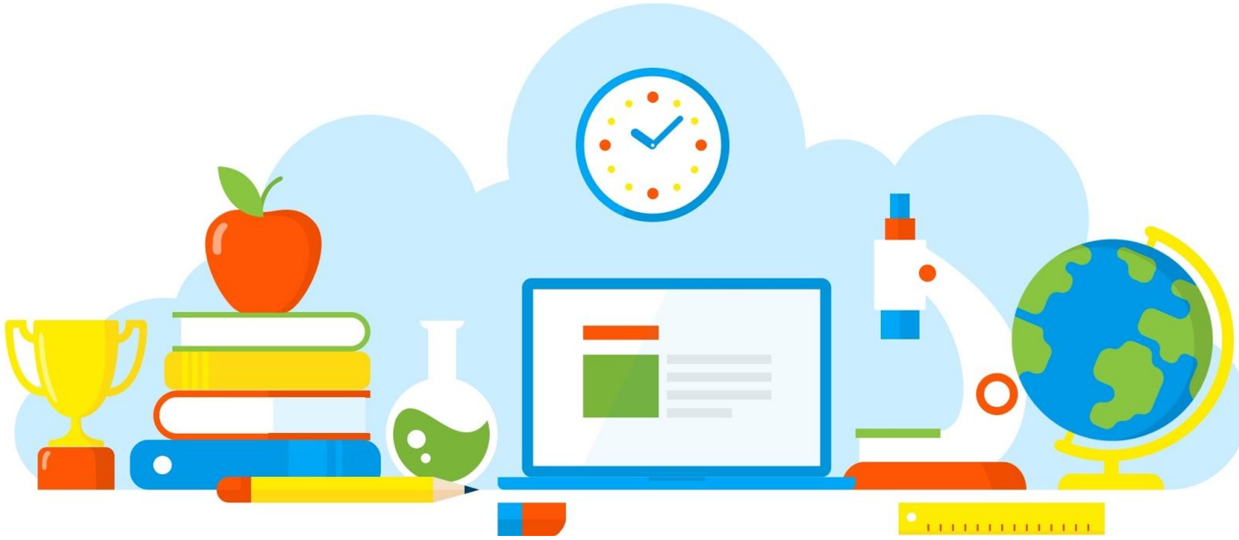


# DSAA Assignment 2

*Roll Number - 2018121004*

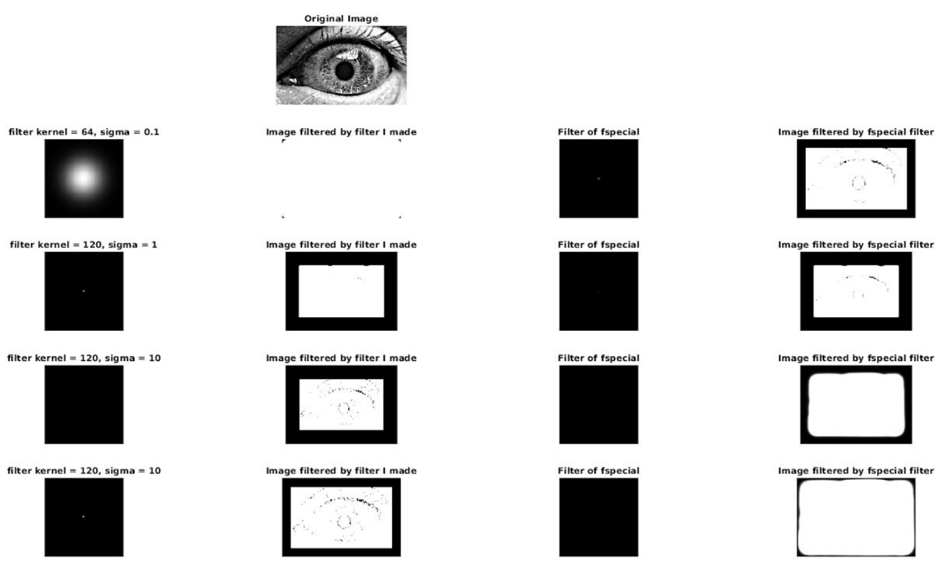
Anurag Sahu
25/02/8018

# Q1 part 1

## Code

```
function ret = gauss_filter(N, sigma)
    ret = single(zeros(N));
    for row = 1 : N
        for col = 1 : N
            x = row - N/2;
            y = col - N/2;
            ret(row, col) = (1/2*pi*sigma*sigma)*exp(-(x*x + y*y)/2*sigma*sigma);
        end
    end
    ret = mat2gray(ret);
end
```

## OutPut of code:-




## Observation:-

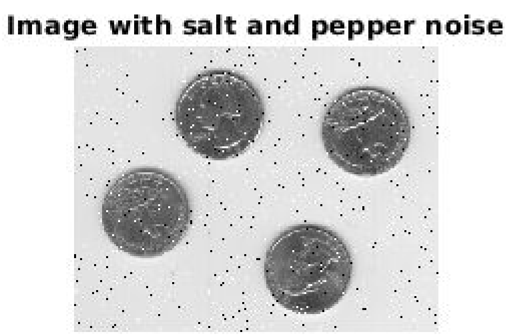
From the above images, it can be clearly deduced that the filters which have a lesser value of sigma have more white space in the middle and the image is totally whitened by the use of those type of filters, on the other side(in build functions) the same effect is reversed, the with larger value of sigma gives a more whitened image.

# Q1 part 2

## Code

```
function mean_image = median_filter(I, N)
    set = im2col(I,[N N]);
    replace_with = (median(set));
    mean_image = col2im(replace_with, [N N],size(I));
end
```

## OutPut





## Observation

The median filters are the most useful filter for removing the salt and pepper noise, in salt and pepper noise the image signal some low frequency and high-frequency noise and the can be removed by replacing the whole region be the median values for smaller pixel areas depending upon the spread of noise.

Ideally, the kernel size taken is between 2 and 4.

# Q1 part 3

## Code

```
image = imread("cameraman.tif");
imshow(image);
kernel_size = 3;
sigma = 0.1;
g_filter = gauss_filter(kernel_size, sigma);
g_filtered = imfilter(image,g_filter);
N = 3;
median_filtered = median_filter(g_filtered,N);
imshow(median_filtered);
```

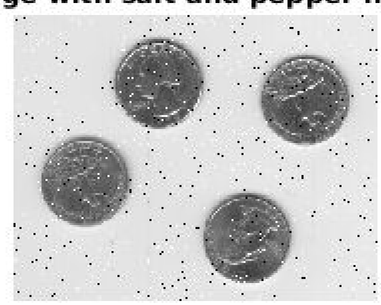
## OutPut

## Observation

The Image we can see here is the output of applying both gauss_filter and then median filter back to back, the image has lost the details, but it can be clearly seen that the image has gained sharp edges all the sharpness is increased, the regions with larger area with almost the same color have developed sharp boundaries, this can be segmentation.

# Q1 part 4

**Image with salt and pepper noise**



**median Filter Applied**

## Observation

The median filters are the most useful filter for removing the salt and pepper noise, in salt and pepper noise the image signal some low frequency and high-frequency noise and the can be removed by replacing the whole region be the median values for smaller pixel areas depending upon the spread of noise.

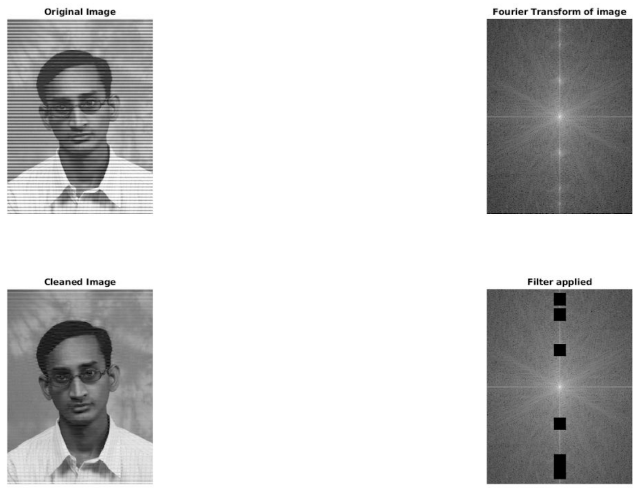
Ideally, the kernel size taken is between 2 and 4.

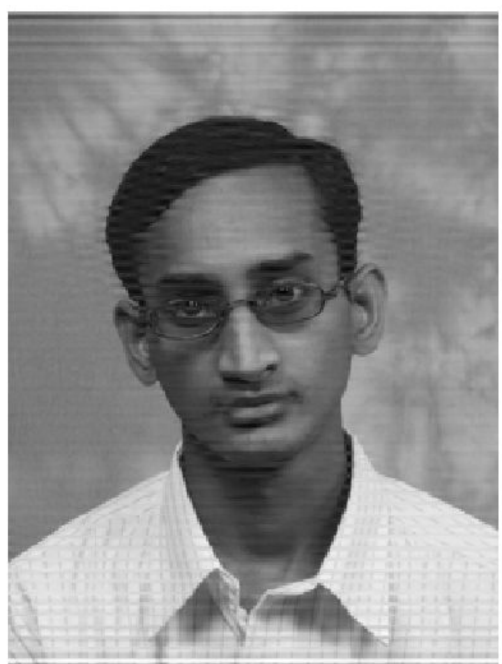
# Q1 part 5

## Code

```
image = imread("inp2.png");
subplot(2,2,1), imshow(image), title("Original Image");
freq_abs = abs((fft2(double(image))));
s=log(1+fftshift(freq_abs));
freq_angle = angle((fft2(double(image))));
subplot(2,2,2);   imshow(s,[]),  title("Fourier Transform of image");
fft_img = fftshift((fft2(double(image))));
diff = 10;

for j = 120-diff:120+diff
    for n = 100-diff:100+diff
        fft_img(n,j) = 0;
        s(n,j) = 0;
    end
    for n = 220-diff:220+diff
        fft_img(n,j) = 0;
        s(n,j) = 0;
    end
    for n = 280-diff:280+diff
        fft_img(n,j) = 0;
        s(n,j) = 0;

    end
    for n = 300-diff:300+diff
        fft_img(n,j) = 0;
        s(n,j) = 0;

    end
    for n = 17-diff:17+diff
        fft_img(n,j) = 0;
        s(n,j) = 0;
    end
    for n = 42-diff:42+diff
        fft_img(n,j) = 0;
        s(n,j) = 0;
    end
end


final_image = real(ifft2(ifftshift(fft_img)));
subplot(2,2,3); imshow(final_image,[]), title("Cleaned Image");
subplot(2,2,4), imshow(s,[]),  title("Filter applied");
```

## OutPut

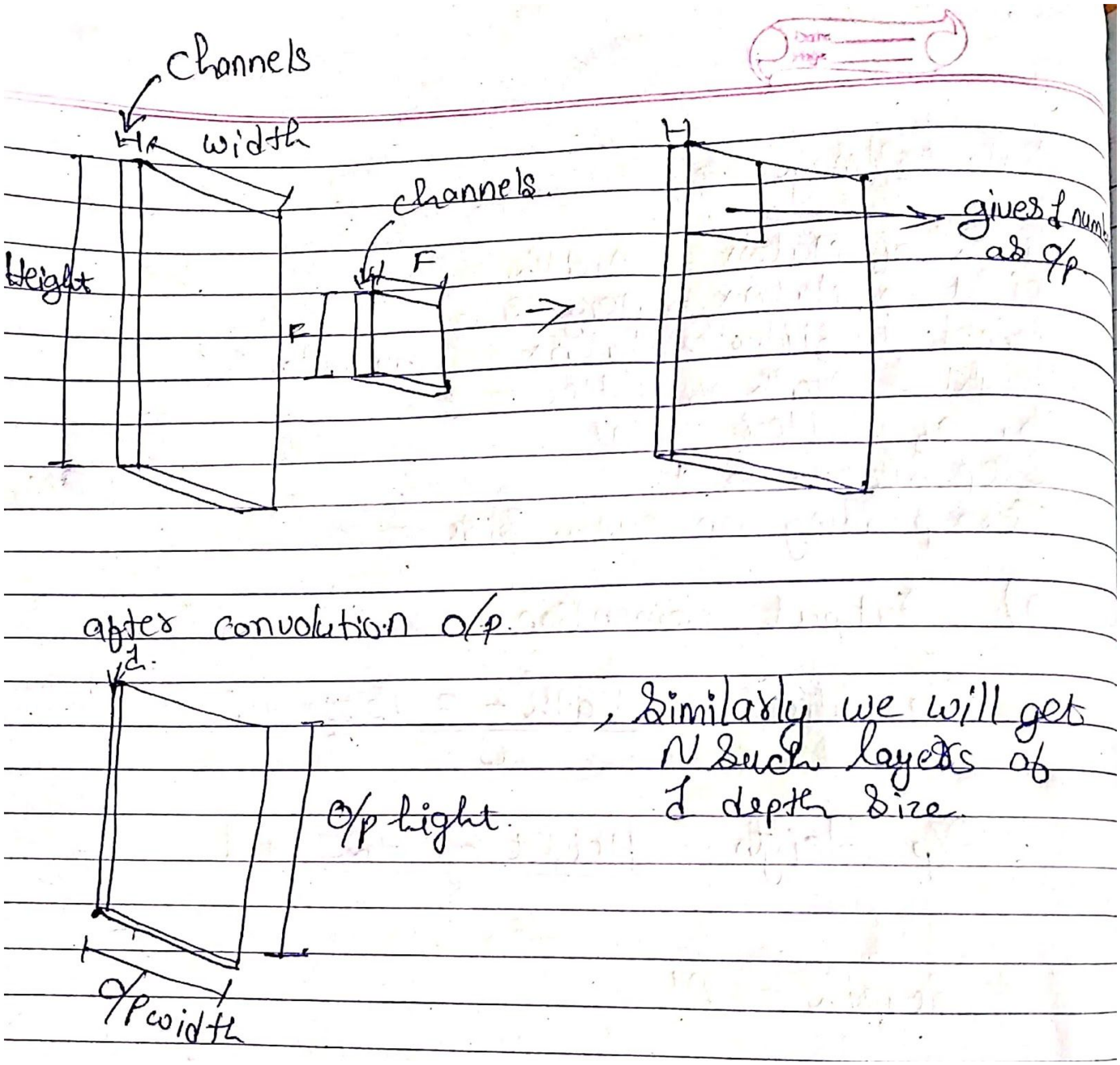

Original Image
Fourier Transform of image
Cleaned Image
Filter applied



## Observation:-

The noise, in this case, was very periodic one, hence when we looked at the frequency image of the image we can clearly pinpoint the places where the image had a lot of unusual frequency, therefore, we can just mask the frequency overheads and the image is now appearing a lot clear.

# Q2



Width of Matrix = Width
Height of Matrix = Heights
Channels in filter & Matrix = Channels
Height & Width of Filter = F
No. of Filters = N
Step size = S
Zero padding on each side = Z.

a) Output dimention =

$$O/p \text{ Width} = \frac{\text{Width} - F + 2Z}{S} + 1$$

$$O/p \text{ Heigth} = \frac{\text{Height} - F + 2Z}{S} + 1$$

deapth = N

The final dimention of O/p will be.

$$\left(\frac{\text{Width} - F + 2Z}{S} + 1\right) * \left(\frac{\text{Height} - F + 2Z}{S} + 1\right) * N$$

Part-2.

Total Number of additions

$= \text{final dimention} * F * F * \text{channels} - 1$

Total Number of Multiplications

$= \text{final dimention} * F * F * \text{channels}.$

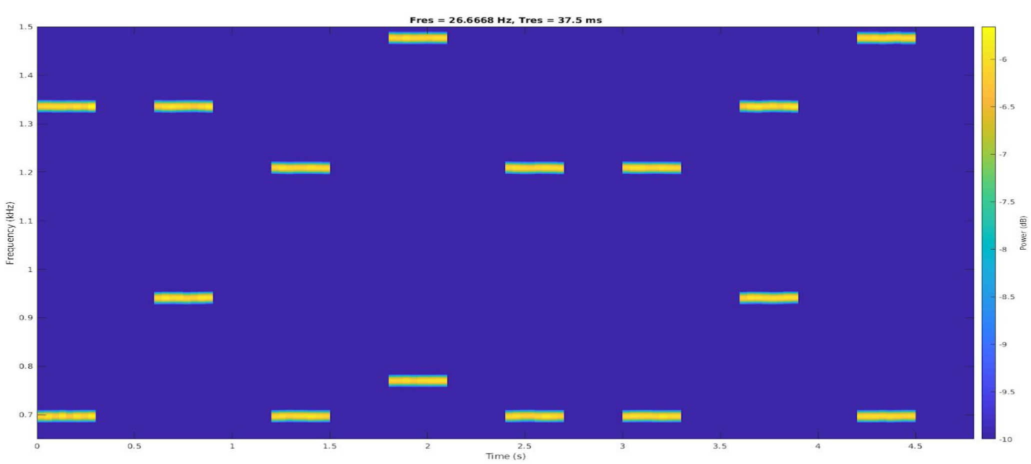
$$= \left\lceil \frac{\text{width} - F + 2Z}{S} \right\rceil * \left\lceil \frac{\text{Height} - F + 2Z}{S} \right\rceil * N * F * F * \text{channels}.$$

# Q3

## Code

```
close all;
clear all;
clc;
[audio_0,fs_0] = audioread('tone.wav');
pspectrum(audio_0,fs_0,'spectrogram','Leakage',1,'OverlapPercent',0,'MinThreshold',-10,'FrequencyLimits',[650,1500]);

%% answer = 20161103
```

## OutPut



## Observation

The Answer is -> **20161103**. The dialed tones have two frequency components added to them, so in order to find the dialed numbers we have to make a short time Fourier transform of the input signal and then it will be clearly visible which all frequencies are released at which point of time approximately, the frequencies corresponding to each number are given below.

**Dual-Tone Multi-Frequency (DTMF) table of frequency combinations**

"High Group" frequencies [Hz]

| "Low Group" frequencies [Hz] | 1209 | 1336 | 1477 | 1633 | |
|---|---|---|---|---|---|
| 697 | 1 | 2 | 3 | A | (Row 1) |
| 770 | 4 | 5 | 6 | B | (Row 2) |
| 852 | 7 | 8 | 9 | C | (Row 3) |
| 941 | * | 0 | # | D | (Row 4) |
| | (Column 1) | (Column 2) | (Column 3) | (Column 4) | |

# Q4

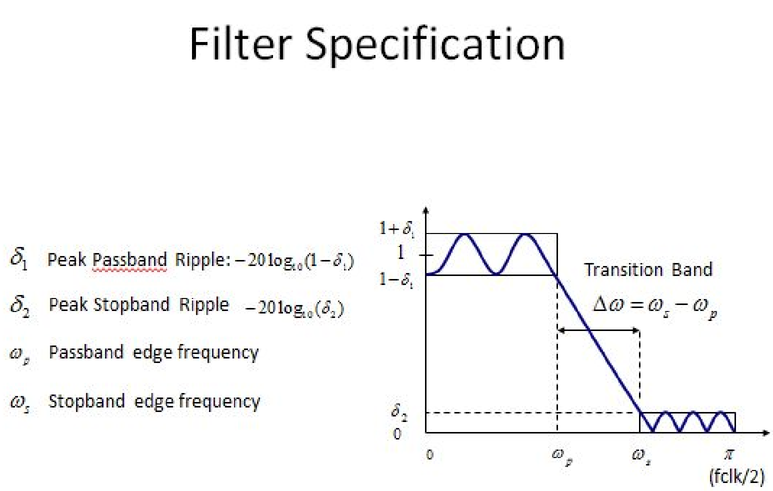
## Code

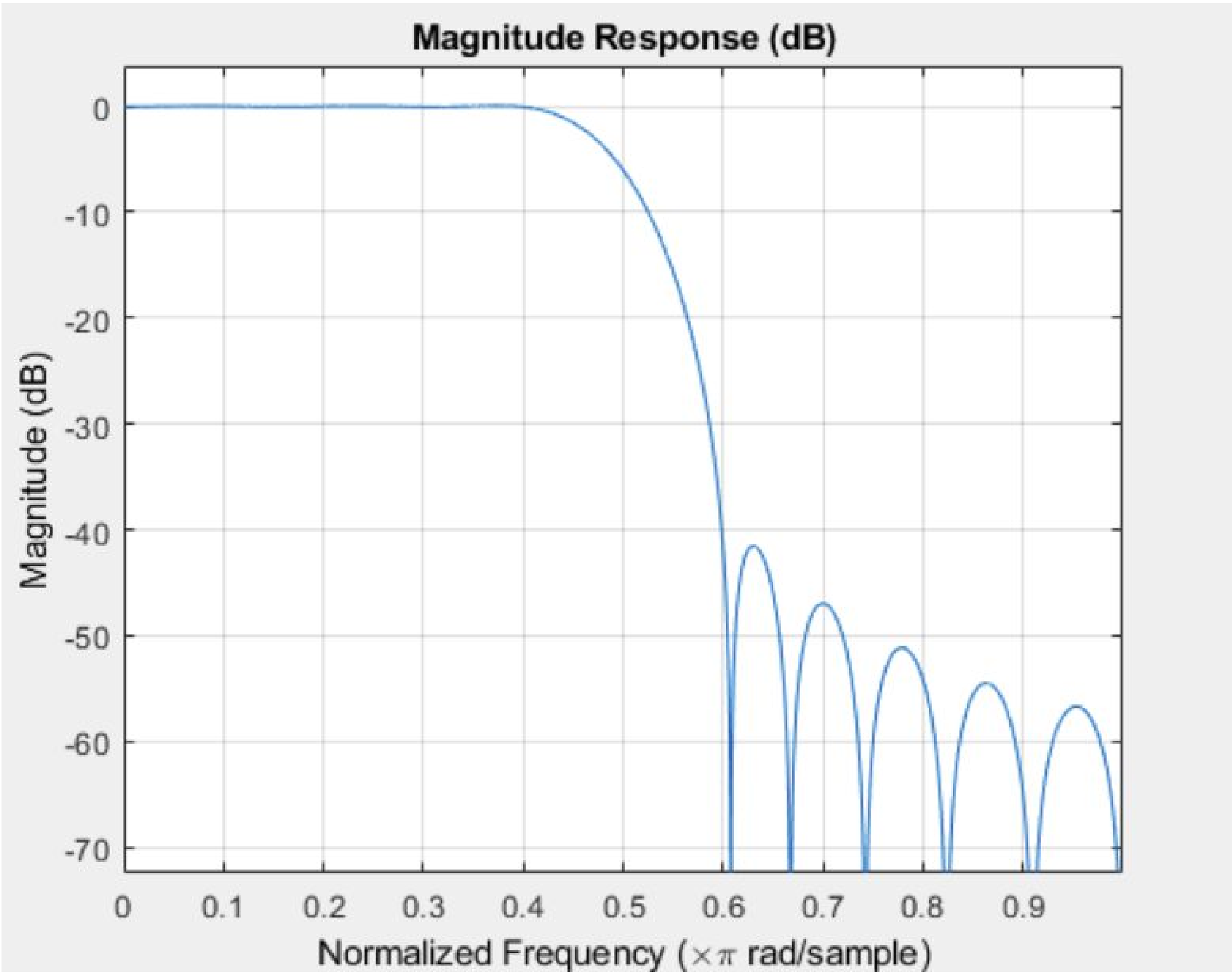
```
[audio,Fs] = audioread('signal_4.wav');
%sound(audio);  pause(40);
filterDesign= designfilt('lowpassfir','PassbandFrequency',0.15,...
                         'StopbandFrequency',0.3, ...
                         'PassbandRipple',1, ...
                         'StopbandAttenuation',160, ...
                         'DesignMethod','kaiserwin');

f_op = filtfilt(filterDesign,audio);
f_op = f_op .* 2;
sound(f_op);
pspectrum(f_op(1:length(f_op)),Fs,'spectrogram');
```

## Observation.

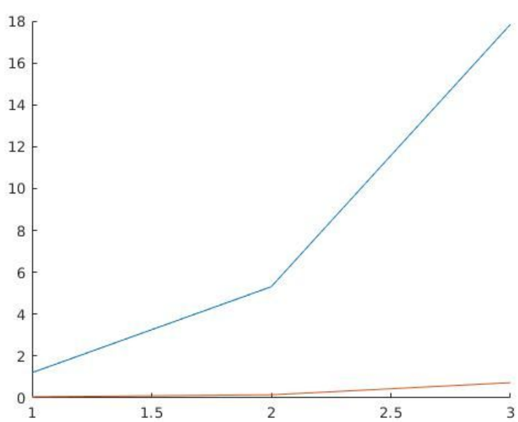
The audio sample provided was having a lot of high-frequency noise it was like broadband noise, so in order to remove the noise we can make the signal pass through low pass filter, So fo that we have designed a filter,

The filter has a Passband frequency of 0.15 * pi rad/sec, stopband frequency 0.3 * pi rad/sec, and stopband attenuation of 160 although designing the filter in the analog circuit will be difficult because of the extreme values but it can still work on Matlab.





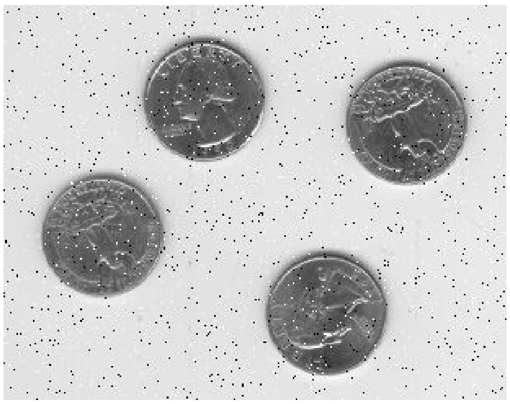
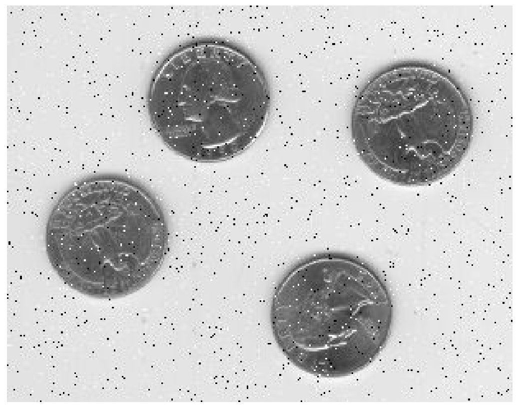

## Q5



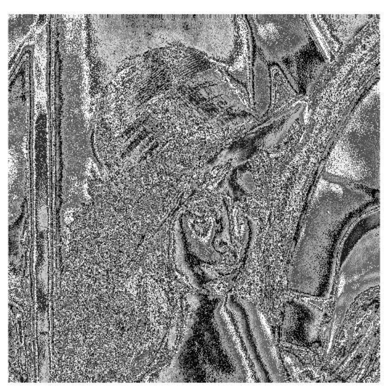

Here the red line is for DFT and the blue line is for FFT.

## Observation.

1. The DFT here shows better performance than the FFT because FFT is being called recursively the FFT is taking a lot of time, for larger values FFT is better than DFT.

The image on the right is the ifft(fft(image)) and the left one is ifft(FFT2_func));

# Q6

## The Code of the Question looks like this

```
image = imread('cameraman.tif');
subplot(1,3,1);     imshow(image,[]); title('Original Image');
freq_domain = ((fft2(double(image))));
image_recon = abs(fft2(double(freq_domain)));
subplot(1,3,2); imshow(image_recon,[]); title('Reconstructed Image');
fully_recover = abs(fft2(flipud(fliplr(freq_domain))));
subplot(1,3,3); imshow(fully_recover,[]); title('Fully Recovered Image');
```

## The output of the code


## The Observations:-

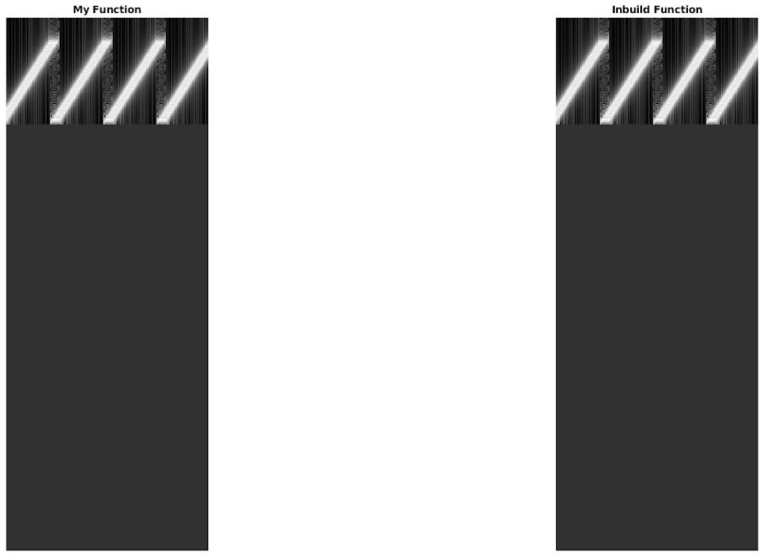
1. The image as it can be seen is mirrored along x-axis and y-axis. This happens because it's the property of DFT that $x[n] \rightarrow X[n]$ (Applying N DFT) And then $X[n] \rightarrow Nx[-n]$ (Applying N DFT) And this operation is applied on both rows and columns hence the image is mirrored along x-axis and y-axis, The original image can be retained if we reverse the frequency domain coordinates of the matrix.

# Q7 part 1

## The Code of the Question looks like this

```
clc;
clear all;
close all;

[audio] = double(audioread('chirp.wav'));
window_size = 200;
stride  = 20;
subplot(1,2,1); result = spectrogram(audio,window_size,stride);
title("My Function");
subplot(1,2,2);  spectrogram(audio,window_size,stride);
title("Inbuild Function");


function result = spectrogram(audio,window_size,stride)
    number_of_fft = ceil((size(audio,1)-window_size)/stride);
    final_spectrogram = ones([500 number_of_fft]);
    for i = [0:number_of_fft-1]
        fft_done = fftshift(fft(double(audio(i*stride+1 : i*stride+window_size))));
        final_spectrogram(1:size(fft_done)/2,i+1) = abs(fft_done(1:size(fft_done)/2));
    end
    result = mat2gray(log(1 + final_spectrogram));
    imshow(result);
end
```

## The output of the code




## The Observations:-

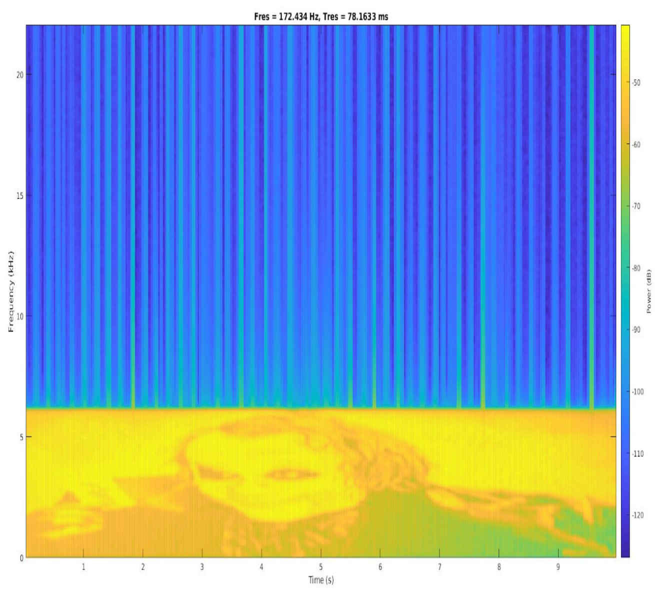
1. The output produced by the by the spectrogram I made and inbuild spectrogram is same.

## Q7 part 2

### The Code of the Question looks like this

```
[audio,Fs] = audioread('message.wav');
pspectrum(audio,Fs,'spectrogram');
```

### The output of the code




### The Observations:-

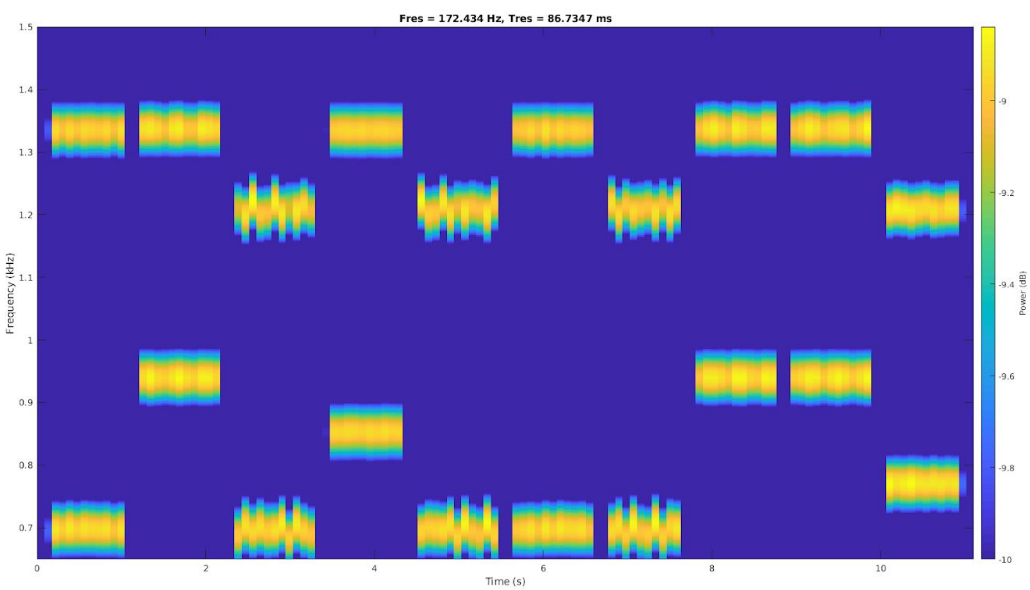
2. I have plotted the spectrum of the sound signal and from that, I can deduce that the password might be 'joker', as we can see the picture of joker the in the downside of the spectrogram.

# Q7 part 3

## The Code of the Question looks like this

```
roll_number = 2018121004;
dial_tone_op = dial_tone(roll_number);
%sound(dial_tone_op,44100*1.5);
pspectrum(dial_tone_op,44100,'spectrogram','Leakage',1,'OverlapPercent',0,'MinThreshold',-10,'FrequencyLimits',[650,1500]);

function dial_tone_op = dial_tone(roll_number)
    Fs = 44100;
    all_digits = dec2base(roll_number,10)-'0';
    pause = zeros(0.1*Fs,1);
    dial_tone_op = zeros(0.1*Fs,1);
    for n = 1:length(all_digits)
        [aud,~] = audioread(strcat('./q3/',int2str(all_digits(n)),'.ogg'));
        dial_tone_op = [dial_tone_op;aud;pause];
    end
end
```

## The output of the code




## The Observations:-

3. I have plotted the spectrum of the sound signal from the generated by the multi-tone dialing of my roll number = 2018121004.