*In the Name of God*

University of Tehran

School of Electrical and
Computer Engineering

# Digital Signal Processing

# Computer Assignment #2

**Amirhossein Mohammadi**
**810197689**

Spring 2021

# Table of Contents

# Abstract

Goal of the computer assignment is to get familiar with Filter Designer Tool of matlab. Designing filters in this environment is the main goal. It takes look at frequency response of images and voices either. At last we get familiar with kernels and what is the result of using them.

# Question one

## Part A & B

Sampling frequency is equal to 11025.

$$f_s = 11025$$

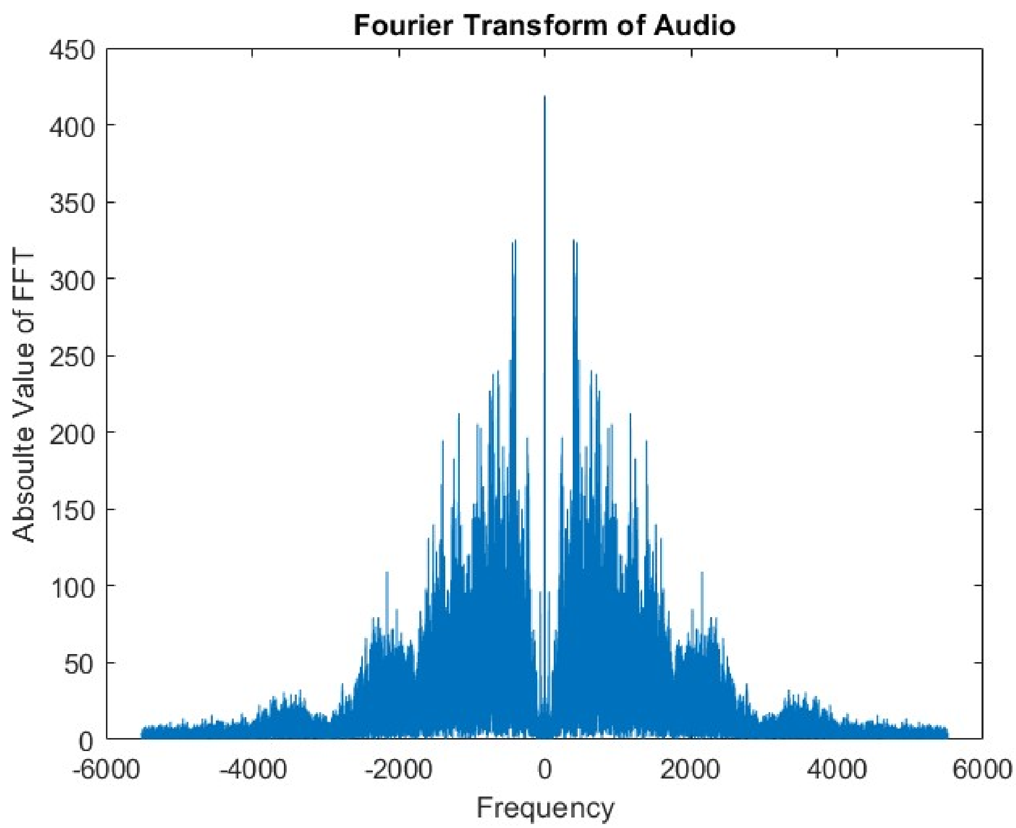


Figure 1, Fourier Transform of Audio

As We can see bandwidth of signal is:

$$BW = 5500\ hz$$

## Part C & D & E

From definition of autocorrelation function, we have:

$$R_y[m] = \sum y[n]y[n-m] =$$
$$\sum (x[n] + \alpha x[n-k_1] + \beta x[n-k_2])(x[n-m\,] + \alpha x[n-m-k_1] + \beta x[n-m-k_2])$$
$$= R_x + \alpha^2 R_x + \beta^2 R_x + \alpha R_x[m+k_1] + \alpha R_x[m-k_1] + \beta R_x[m-k_2] + \beta R_x[m+k_2] + \alpha\beta R_x[m-k_2+k_1] + \alpha\beta R_x[m+k_2-k_1]$$

So, from eq. above and xcorr function output which shown in Figure 2 We can understand that:

$$\begin{aligned} &at\ m = k_1 = 5500 \rightarrow \alpha R_x[0] = 306.9 \\ &at\ m = k_2 = 9000 \rightarrow \beta R_x[0] = 211.4 \\ &at\ m = k_2 - k_1 = 3500 \rightarrow \alpha\beta\ R_x[0] = 66.26 \end{aligned}$$

After solving eq:

$$\alpha = \frac{66.26}{211.4} = 0.31 \approx 0.3;\ k_1 = 5500$$
$$\beta = \frac{66.26}{306.9} = 0.22\ \approx 0.2;\ k_2 = 9000$$

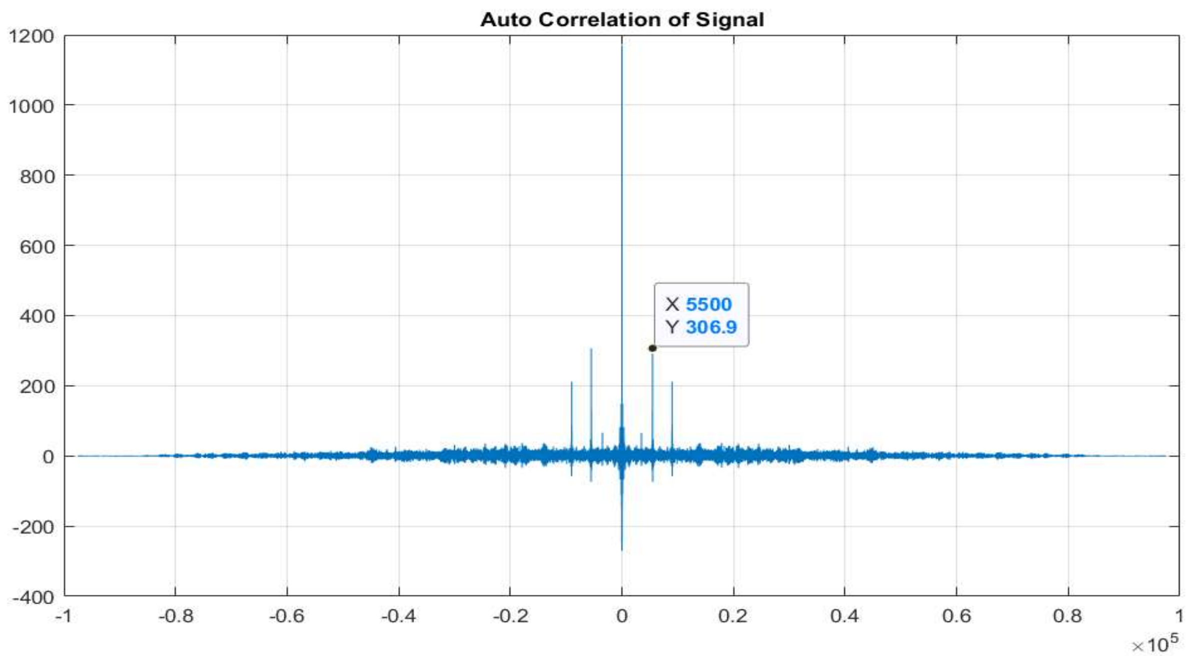


Figure2 , Autocorrelation of audio sample

## Part F, Finding Impulse Response of Echo System

$$we\ know\ that: x * h = y$$

$and\ from\ eq\ we\ have:$

$$y\,[n]\ = x[n] +\ 0.3\, x[n-5.5k] + 0.2\, x[n-9k]$$

$$\rightarrow h\,[n] = \delta[n] + 0.3\delta[n-5.5k] + 0.2\,\delta[n-k]$$

## Part G, Removing Echo from Audio Signal

Filter function in MATLAB requires coefficient of transfer function to filter signal, therefore we find z-transform of Echo System impulse response then, inverse the response to remove Echo From audio.

$$Z - Transform\ \ of\ impulse\ respone: 1 + 0.2\, z^{-9k} + 0.3z^{-5k}$$

$$Inverse\ form\ of\ singal = \frac{1}{1 + 0.2\, z^{-9k} + 0.3z^{-5k}}$$

$So\ coefficient\ for\ filter\ function\ is:$

$b\ =\ 1;$

$a\ =\ [1,\quad zeros(1,5499),\quad 0.3,\quad zeros(1,3499),\quad 0.2\,];$

After performing filter function and listening to audio we can understand that Echo has been completely removed from audio.

The audio has been recorded as: "*y_Without Echo.wav*" and it can be found with other files.

## Part H, Kaiser Filter in Filter and Design App MATLAB

Kaiser Filter has been designed as figure 4, and its frequency response has been shown in figure 3.

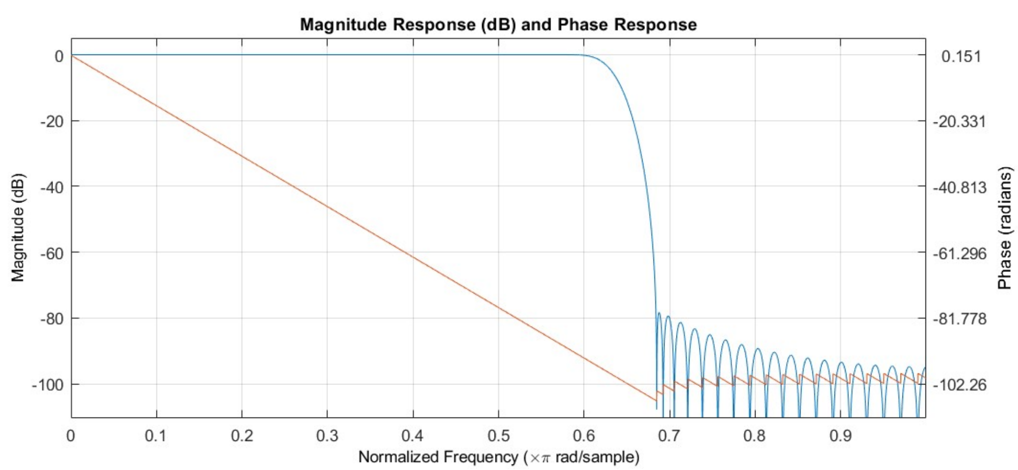


Figure 3, Frequency Response of Kaiser Filter

For better noise limitation, parameter $F_C$ has been changed to 3500

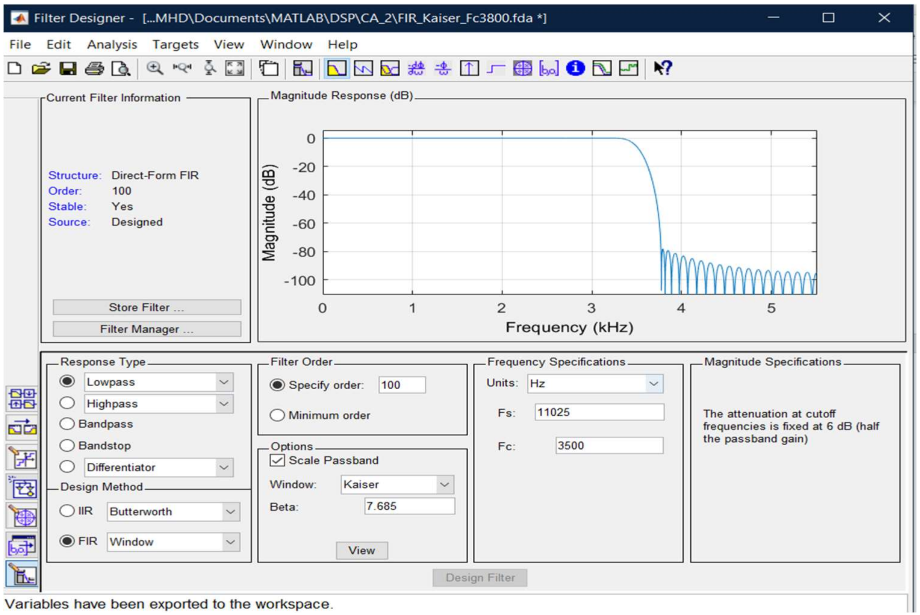


Figure 4, Kaiser Filter Designing

## Part I

As we can see in figure 5 Hd filter has limited the noise and reduced bandwidth of signal to $(f_c = 3500)$

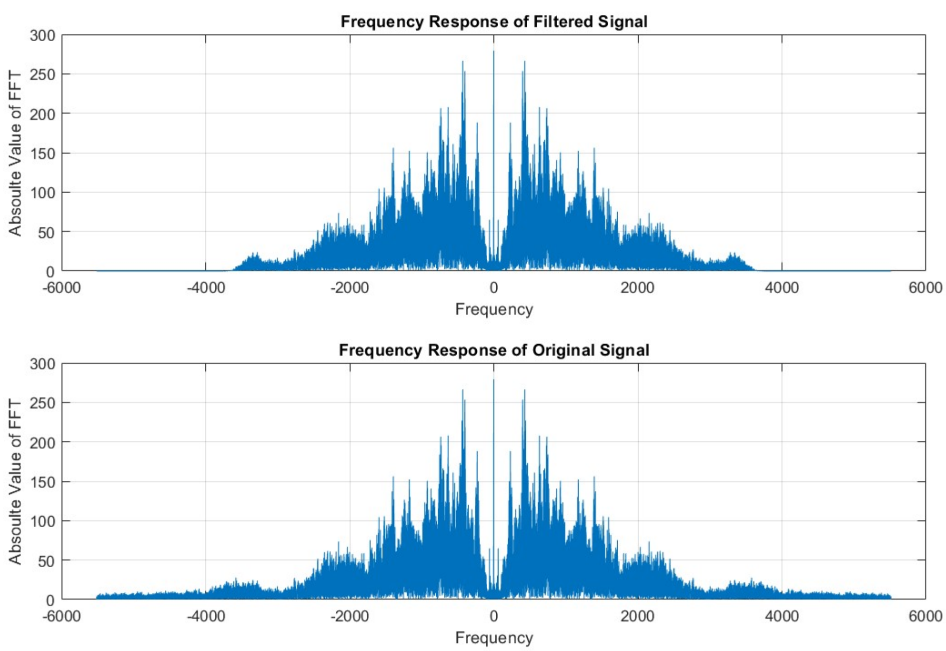


Figure 5, Frequency Response Original and Filtered Signal

## Part J

Here is Signal in Time Domain, for having a closer look time has been limited in figure 6 to understand difference between filters, as we can see high frequencies has been cut from Filtered Signal and it looks smoother.

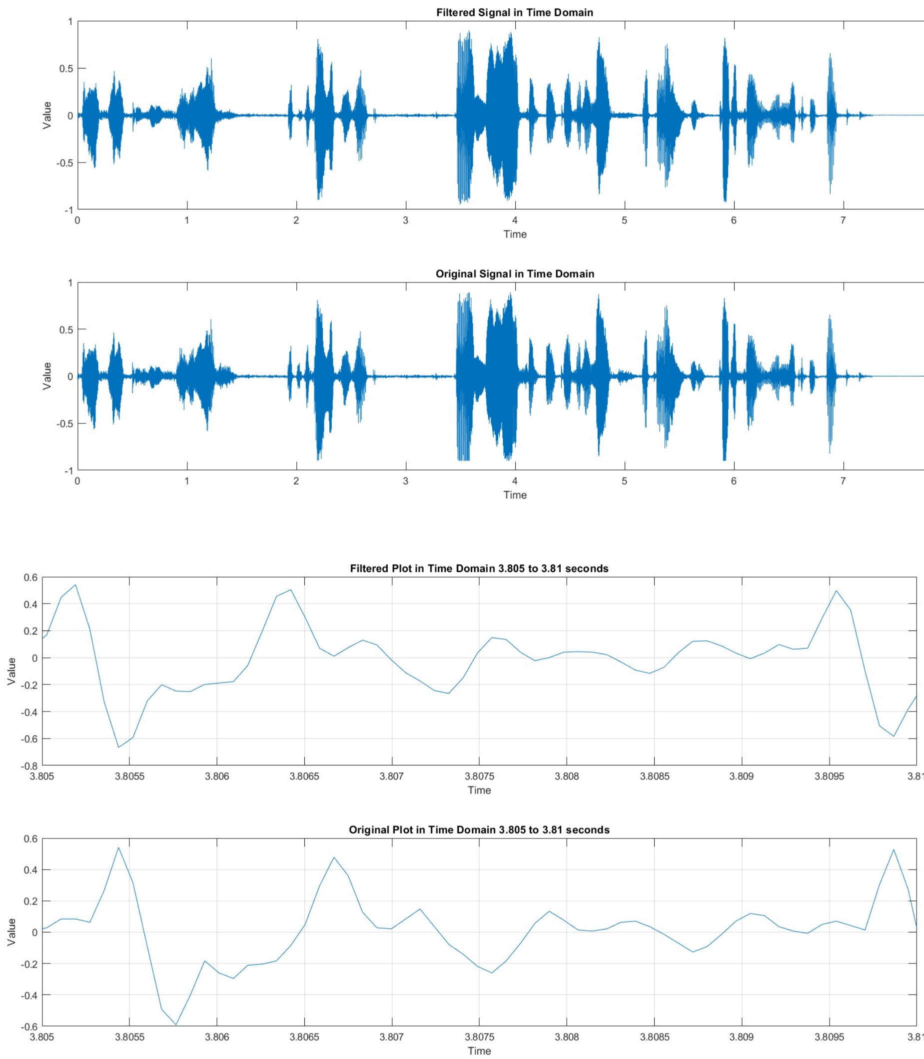


Figure 6, Signals in Time Domain

## Part K, Designing another FIR Filter

Chebyshev filter is used and its frequency response is as below.

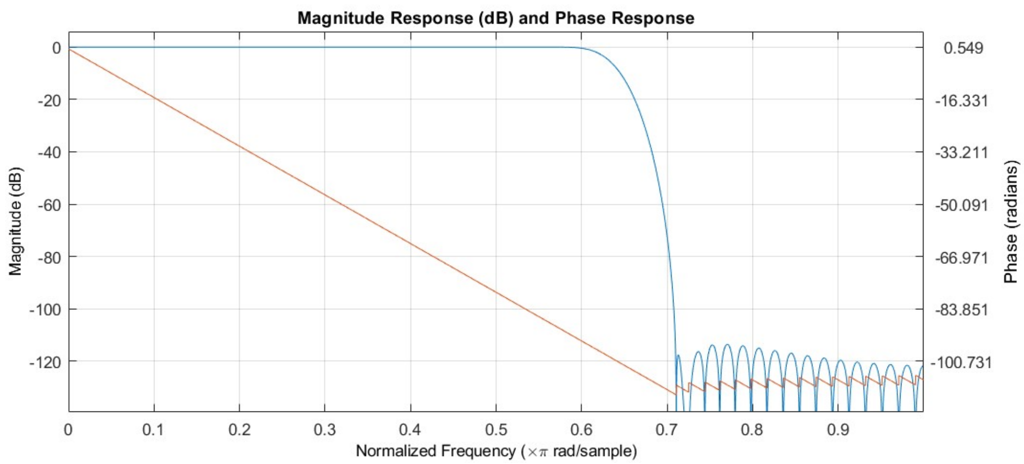


Figure 7, Chebyshev Magnitude and Phase Response

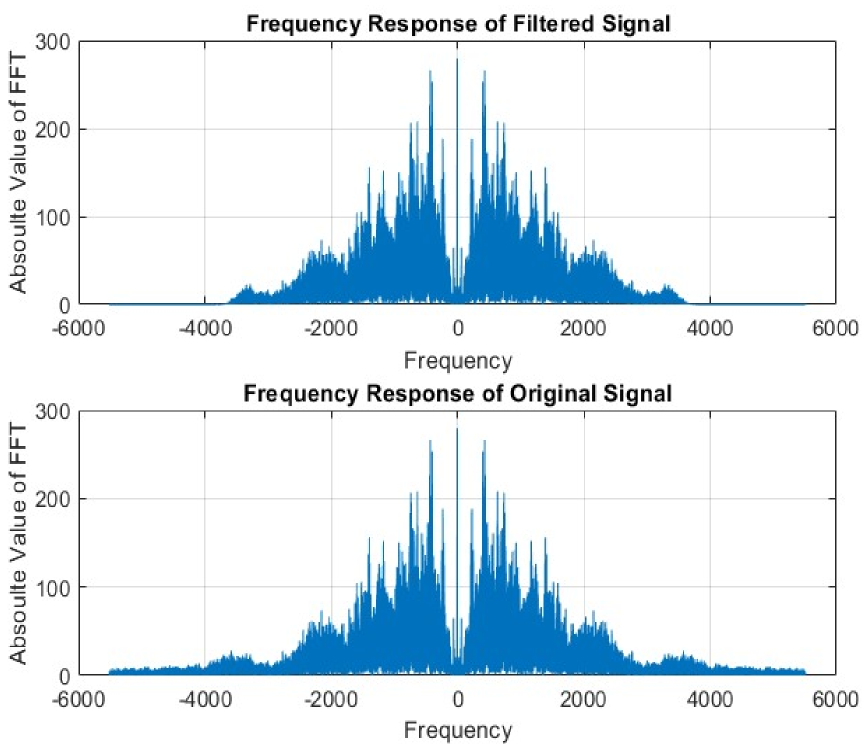


Figure 8, Frequnecy Response of Signal

As We can see Kaiser filter is a little better than Chebyshev in same order.

## Question Two

The transfer function of previous part is used to creat echo and it has been removed with same method.

$$Z - Transform\ \ of\ impulse\ respone : 1 + 0.2\, z^{-9k} + 0.3z^{-5k}$$

$$Inverse\ form\ of\ singal = \frac{1}{1 + 0.2\, z^{-9k} + 0.3z^{-5k}}$$

The Voices Has been recorded and saved, The file Will Be Uploaded With Project.

## Question Three

In This question the goal is to implement different filters on pictures, the original image has shown below

Figure 9 , Original Grayscale Image

Analyzing Different Filters:

**Gaussian**: returns a rotationally symmetric Gaussian lowpass filter. It has two parameters, sigma which indicates power and mean which indicates offset.

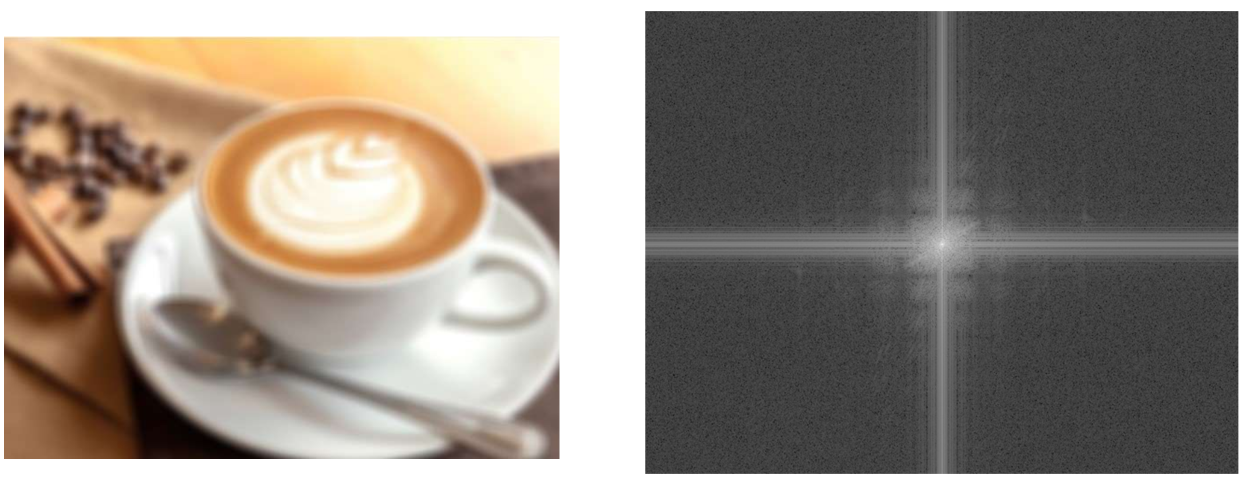

Figure10 Guassian Filter Output, With Frequency Response

**Average:** returns an averaging filter, the parameter indicates average window size as it increases picture tends to be more blurry.

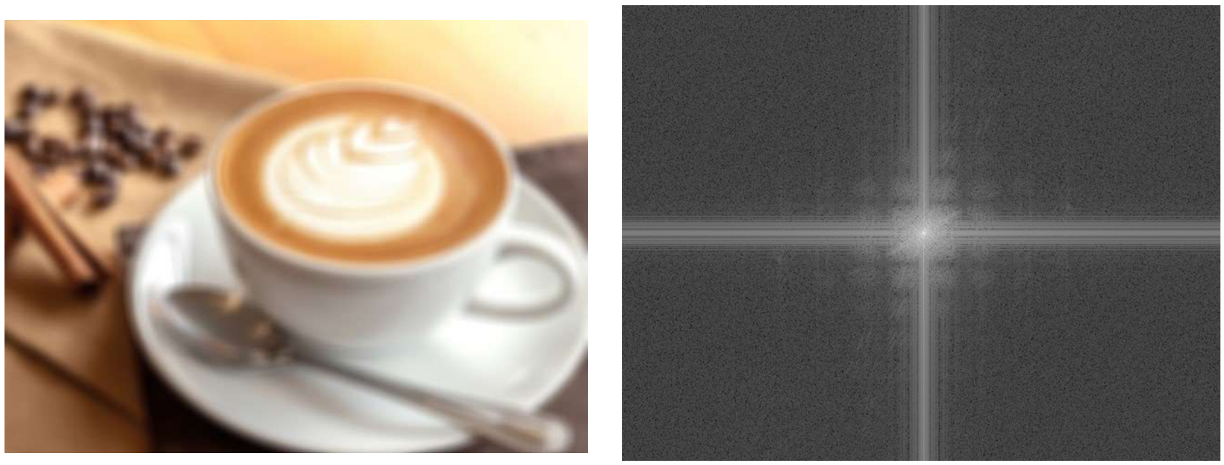

Figure11 , Average filter

The output is very similar to Guassian

**Disk:** returns a circular averaging filter, it is same as before but average is circular. The parameter R shows radius of circle which is being averages as it increases picture is more blurry.

Figure12 , Disk Filter

**These three filters (disk, average, guassian) have lowpass noise as it is clear from pictures .**

**Sobel**: emphasizes horizontal edges using the smoothing effect by a vertical gradient. it has no other parameter. It just keep the edges and other parts will be removed.

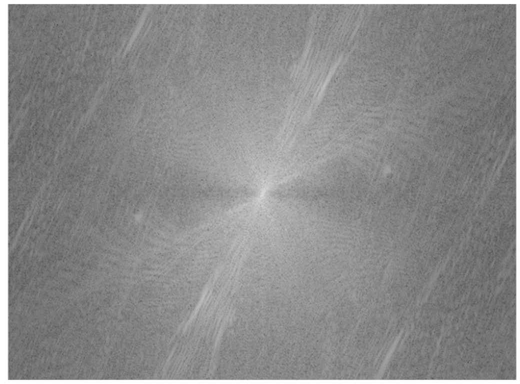

Figure13 , Sobel Filter

**Laplacian:** It is same as Laplacian operator, returns a 3-by-3 filter approximating the shape of the two-dimensional Laplacian operator, alpha controls the shape of the Laplacian. As it increases white pixel reduces. Laplacian differentiate picture two times and find the edges of picture

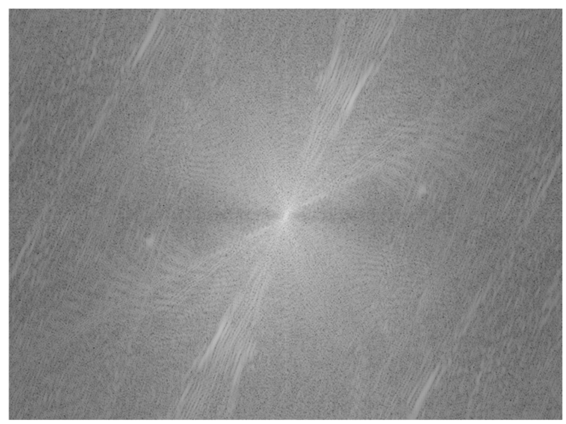

Figure14 , Laplacian Filter

**These two filters find edges, they are high pass and its clear from their frequency response**

## Question 4

### Part A

An image kernel is a small matrix used to apply effects such as blurring, sharpening, edge detection and more.

Here is a glimpse to how they work, the kernel convolves with image for instance I show mechanism for sharpen kernel:

Sharpener Kernel:

$$\begin{pmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Below, for each 3x3 block of pixels in the image on the left, we multiply each pixel by the corresponding entry of the kernel and then take the sum. That sum becomes a new pixel in the image on the right. Hover over a pixel on either image to see how its value is computed.[1]

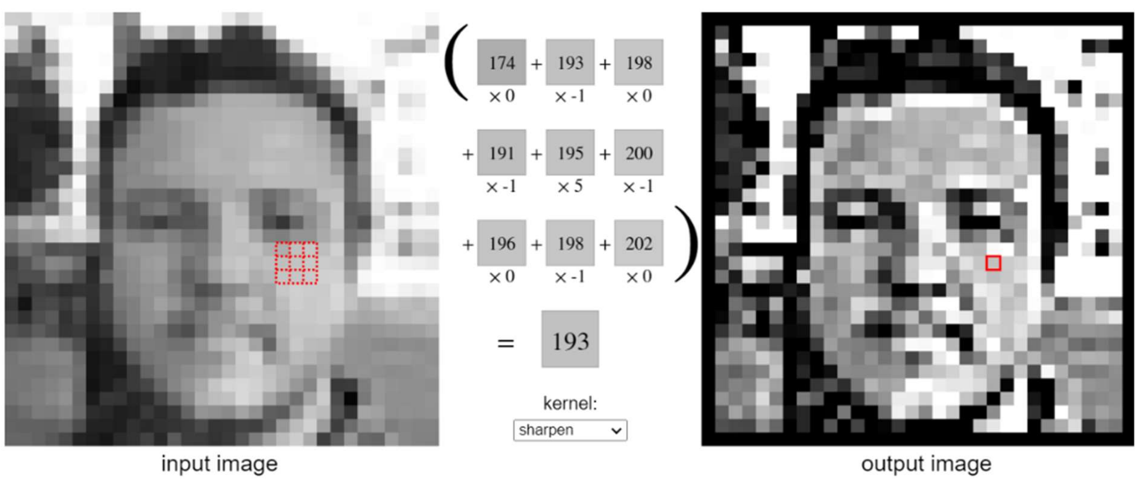


Figure 15

[1] https://setosa.io/

## Part B [1]

Edge Detection: It emphasis on edges and set other pixels to zero.

Figure16 , Edge Detector Kernel

Sharpen: Same as previous, This filter's focus is on edges of picture but it keeps other pixels and it just make image sharper.

Figure17 , Edge Detector

[1] Two methods has been used to show the result: conv2 and imfilter, Here imfilter results are shown. Output of the other method can be seen in code.

Gaussian Blur: it blurs image by applying gaussian function, it makes picture more blurry.

Figure18 , Guassian Blur

Identity: Does not change the image as we can see in figure 19.

Figure19 , Identity

## Part C

Kernels are also used in machine learning for 'feature extraction', a technique for determining the most important portions of an image. By this technique image will be moved in higher dimension for better analysing.

## Appendix: Instruction in Running Code

Each Question has its own name in code documentary.

For question 1 it is necessary to import Kaiser filter which is designed as an object to workspace.

Necessary explanation has been written in comments.