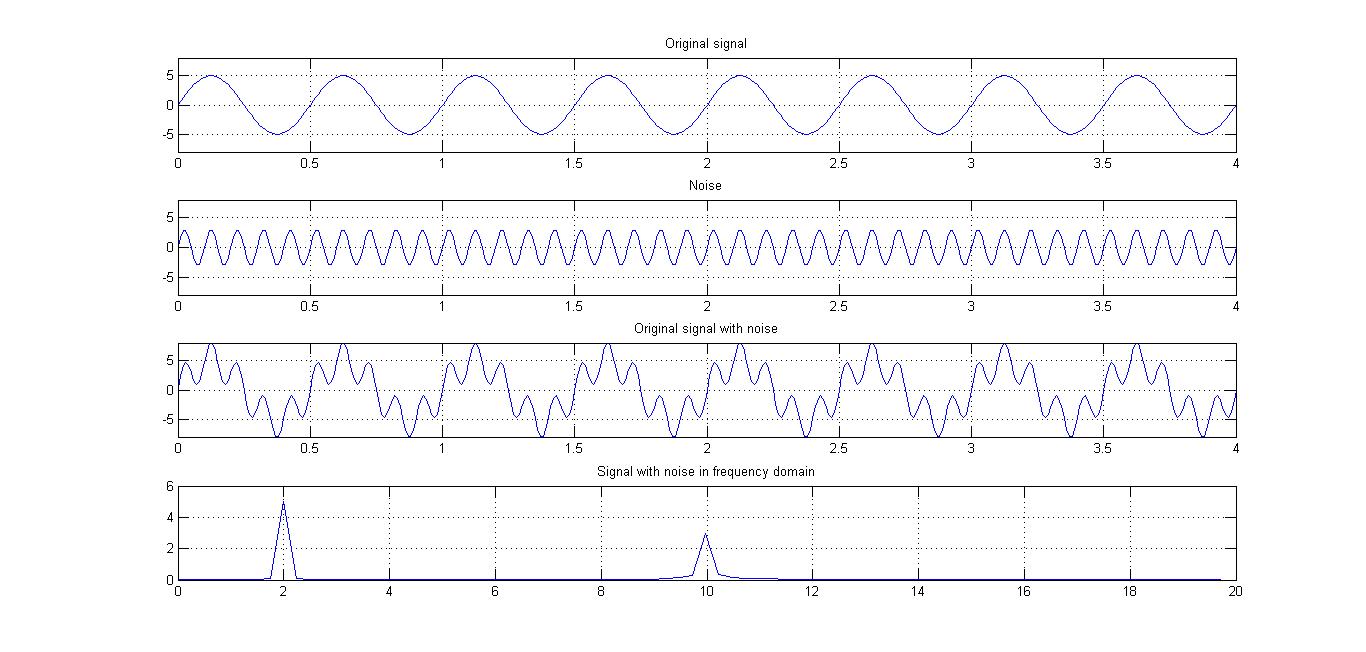
## Verify (experimentally) the basic properties of the Butterworth system. Develop a program that implements the classical Butterworth filter and use the program for signal filtering. Display the results

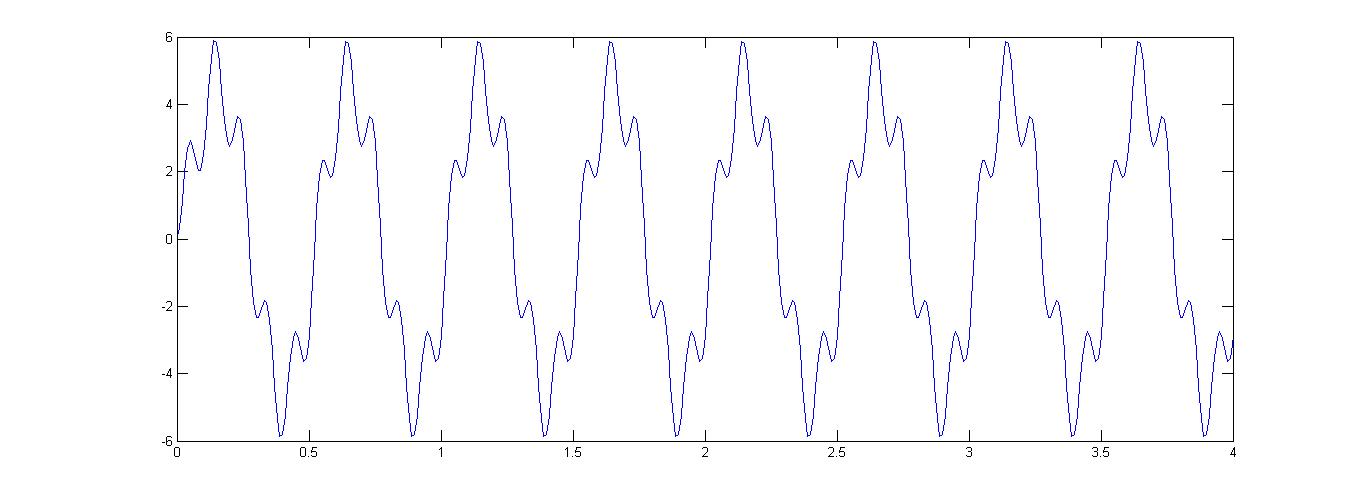
1. Sin wave with noise



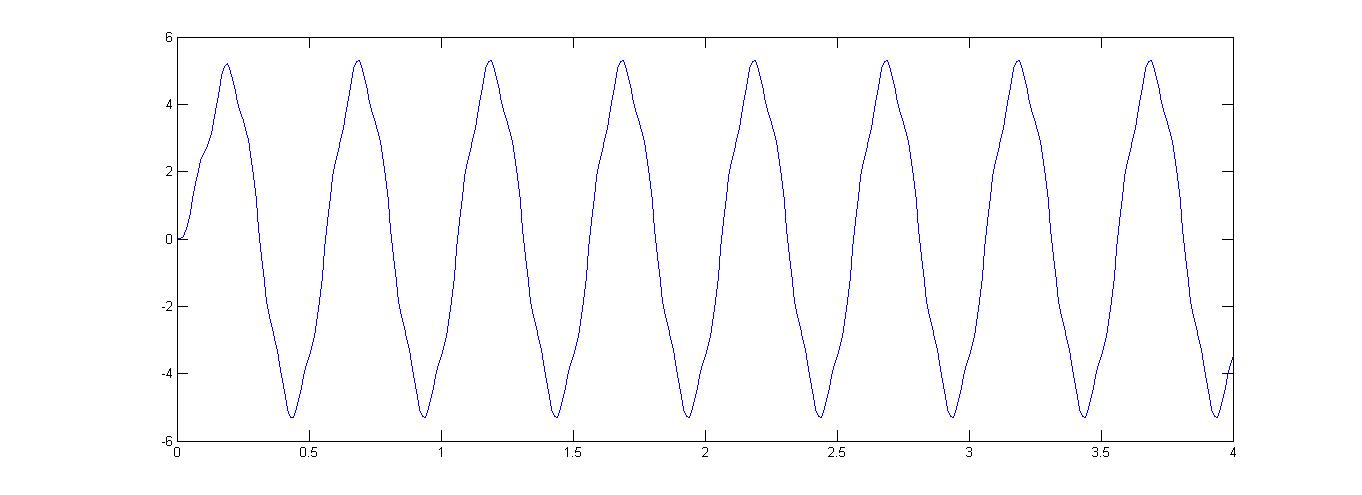
Butterworth filter design using inbuilt functions.

Cutoff frequency: 0.1 (normalized)

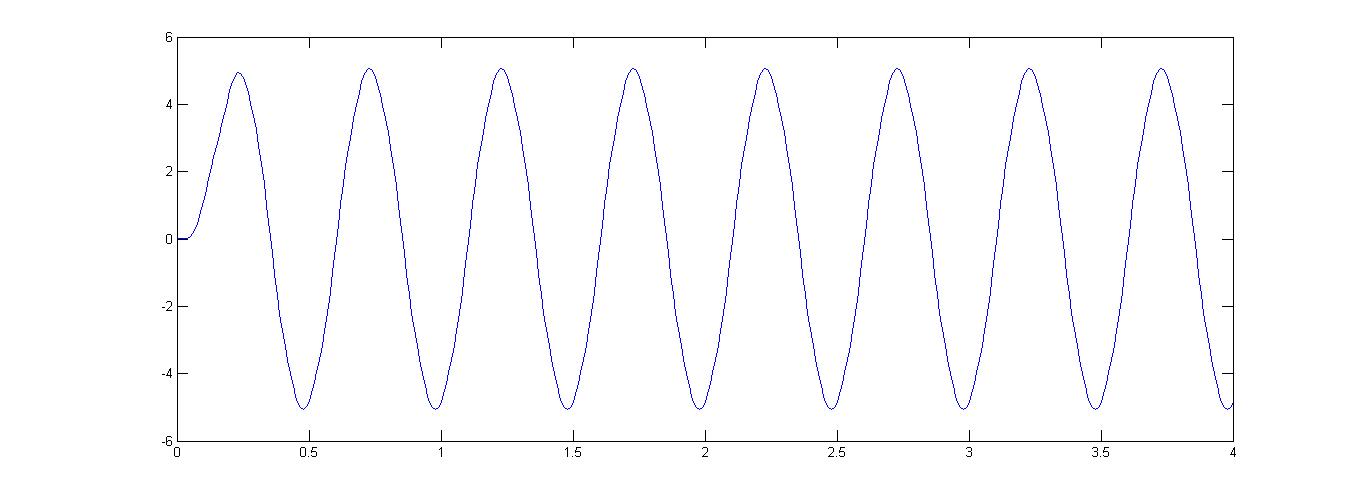
Order = 1



Order = 3



Order = 5



MATLAB code

dt = 1/100; % sampling frequency

et = 4; % length of series

t = 0:dt:et;% x axis points

y\_orig = 5\*sin(2\*2\*pi\*t); % signal

noise = 3\*sin(10\*2\*pi\*t); % noise

y = 5\*sin(2\*2\*pi\*t) + 3\*sin(10\*2\*pi\*t); % signal with noise

figure(1);

subplot(411),plot(t, y\_orig),title('Original signal'); grid on; axis([0 et -8 8])

subplot(412),plot(t,noise),title('Noise'); grid on; axis([0 et -8 8])

subplot(413),plot(t,y),title('Original signal with noise'); grid on; axis([0 et -8 8])

Y = fft(y);

n = size(y,2)/2;

amp\_spec = abs(Y)/n;

subplot(414);

freq = (0:79)/(2\*n\*dt);

plot(freq,amp\_spec(1:80)),title('Signal with noise in frequency domain');grid on

[a, b] = butter(5,0.1);

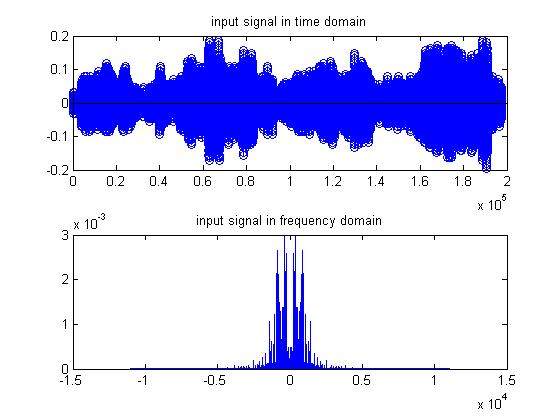
output = filter(a, b, y);

figure(2);

plot(t,output);

AUDIO INPUT

Input signal in time domain and frequency domain

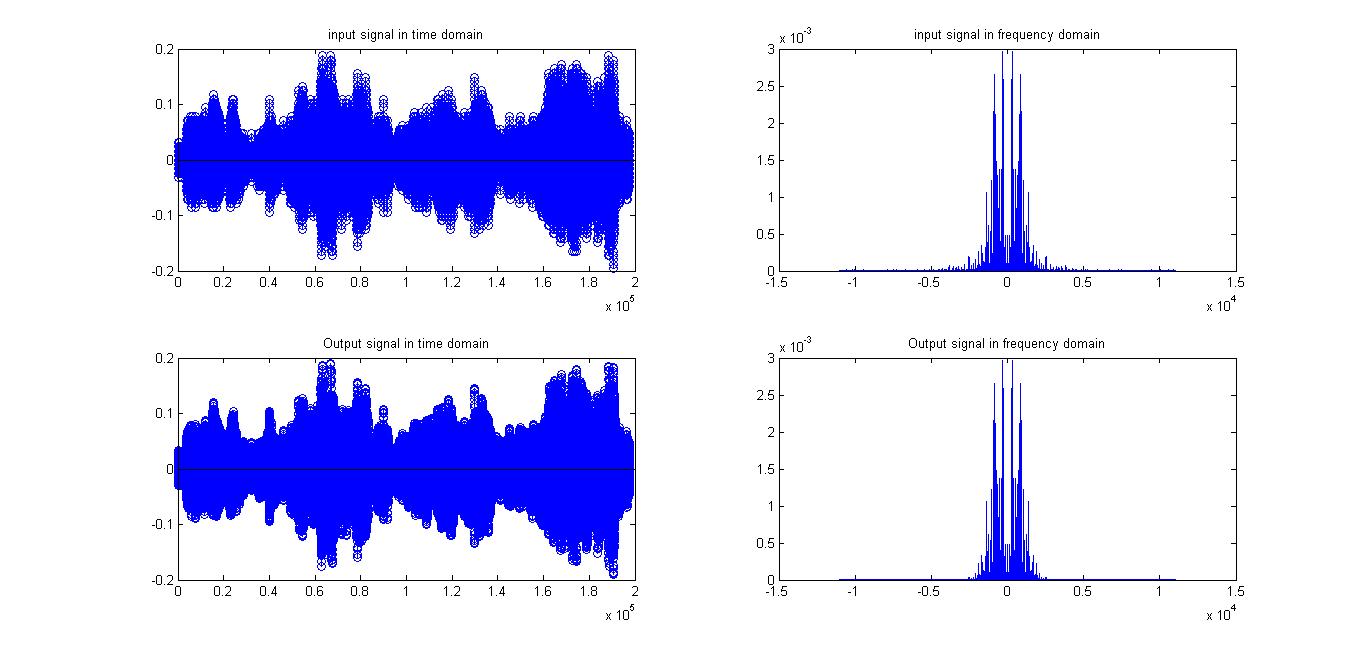


Designed butterworth low pass filter at cutoff frequency = 2000 hz

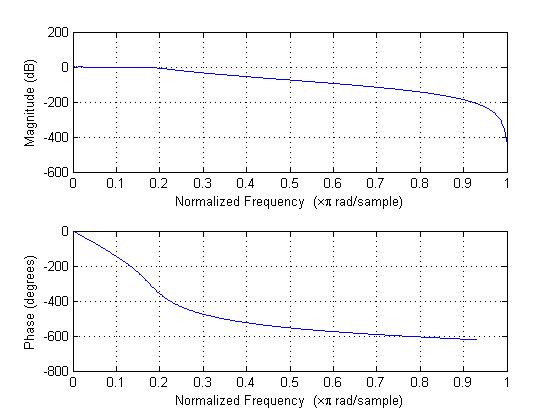
Which gets normalized to 2000/ nyquist frequency = 2000 / (fs/2) = 2000 / 22050\*2 = 0.1814

This cutoff value was estimated after trial and error.

Order of the filter = 7;



Filter plot



Matlab code

[f, fs] = audioread('test.wav');

N = length(f);

%% Playing the audio

sound(f,fs);

pause(10);

%% Plot both audio channels

figure;

subplot(221);

stem(1:N, f(:,1));

title('input signal in time domain');

%% Plot the spectrum

N = size(f,1);

df = fs / N;

w = (-(N/2):(N/2)-1)\*df;

y = fft(f(:,1), N) / N; %//For normalizing, but not needed for our analysis

y2 = fftshift(y);

subplot(222);

plot(w,abs(y2)),title('input signal in frequency domain');

%% Design a lowpass filter that filters out anything after 2000 Hz

order = 7;

cutFreq = 2000 / (fs/2);

[b,a] = butter(order, cutFreq);

%% Filter the signal

fOut = filter(b, a, f);

%% Construct audioplayer object and play

p = audioplayer(fOut, fs);

p.play;

subplot(223),stem(1:N,fOut(:,1)),title('Output signal in time domain');

y = fft(fOut(:,1), N) / N;

y2 = fftshift(y);

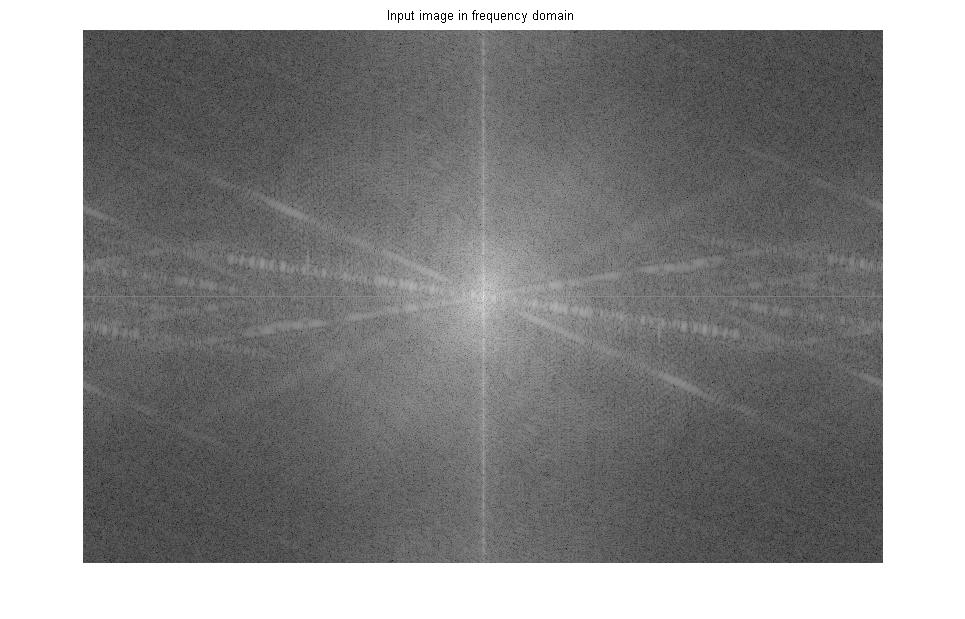
subplot(224),plot(w, abs(y2)),title('Output signal in frequency domain');

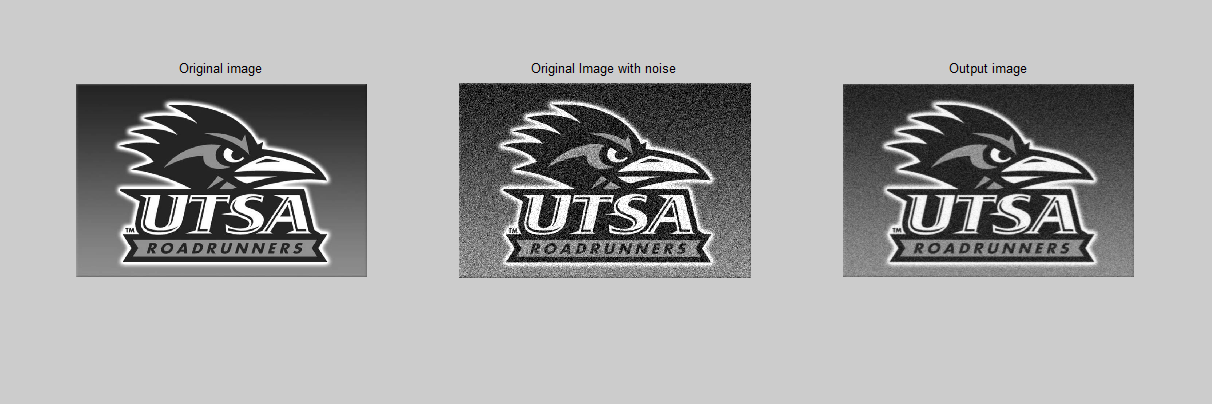
IMAGE INPUT

Selected image



After grayscaling, the image in frequency domain looks like





MATLABCode

%% Low-Pass Filter

clear all;

clc;

close all;

%% reading image (grayscale)

I\_orig = imread('logo.jpg');

I\_orig = rgb2gray(I\_orig);

[M,N] = size(I\_orig);

%% Image in frequency domain

F = fft2(I\_orig);

F = fftshift(F); % Center FFT

F = abs(F); % Get the magnitude

F = log(double(F+1)); % Use log, for perceptual scaling, and +1 since log(0) is undefined

F = mat2gray(F); % Use mat2gray to scale the image between 0 and 1

imshow(F,[]),title('Input image in frequency domain'); % Display the result

figure

%% adding noise to the image

I = imnoise(I\_orig,'gaussian');

%% Finding D(u,v)

D = zeros(size(I));

for u = 1:M

for v = 1:N

D(u,v) = ((u-(M/2))^2 + (v-(N/2))^2)^(1/2);

end

end

%% finding H(u,v)

distance = 40;

order = 1/20;

H = zeros(size(I));

for u = 1:M

for v = 1:N

H(u,v) = 1/(1 + (D(u,v)/distance)^2\*order);

end

end

mesh(H);

figure

%% Processing

F = fft2(I);

F = fftshift(F);

Y = F.\*H;

Y1 = ifftshift(Y);

y = ifft2(Y1);

%% Displaying results

subplot(131),imshow(I\_orig),title('Original image');

subplot(132),imshow(I),title('Original Image with noise');

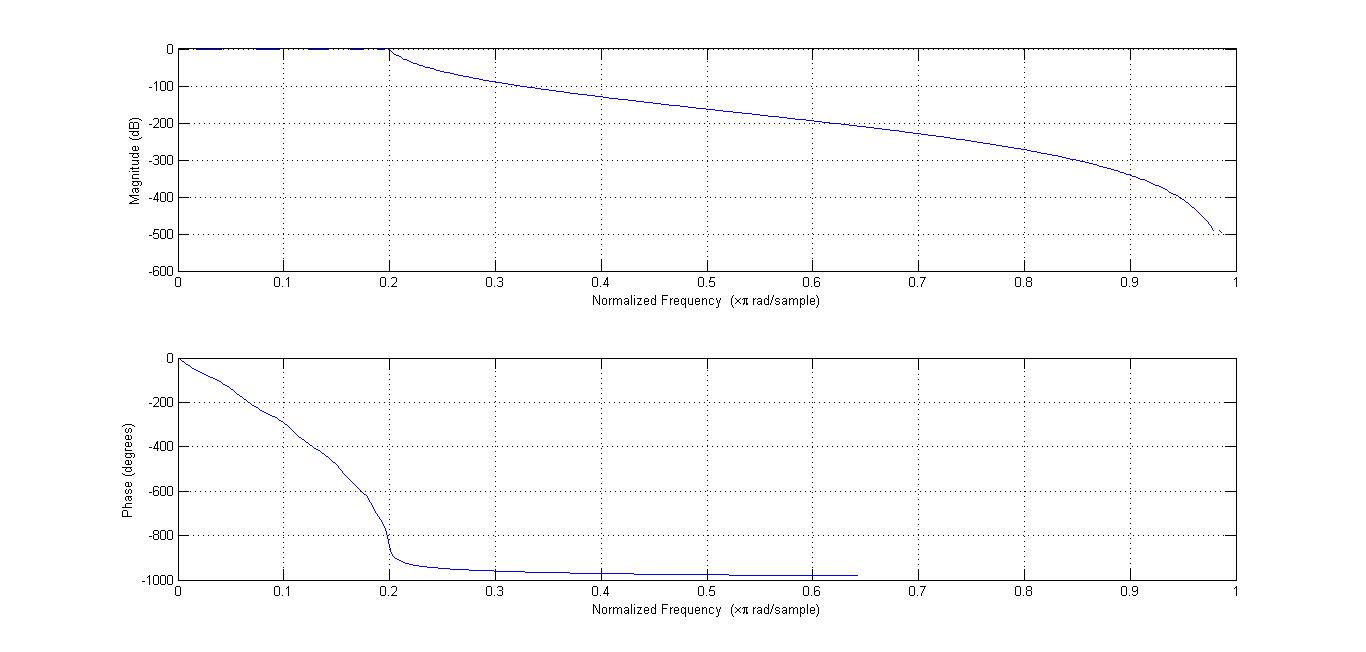
subplot(133),imshow(uint8(y)),title('Output image');

CHEBYSHEF FILTER

Input to output, when applied on images



Filter used



Matlab code

%% Chebyshef

I = imread('logo.jpg');

I = rgb2gray(I);

I\_noise = imnoise(I,'salt & pepper');

order = 11;

p2p\_ripple = 2;

cut = 0.2;

[n, d] = cheby1(order, p2p\_ripple, cut);

fvtool(n,d);

I\_output = filter(n, d, double(I\_noise));

subplot(131),imshow(I),title('Original Image');

subplot(132),imshow(I\_noise),title('Original Image with gaussian noise');

subplot(133),imshow(uint8(I\_output)),title('Filtered image with chebyshef filter');