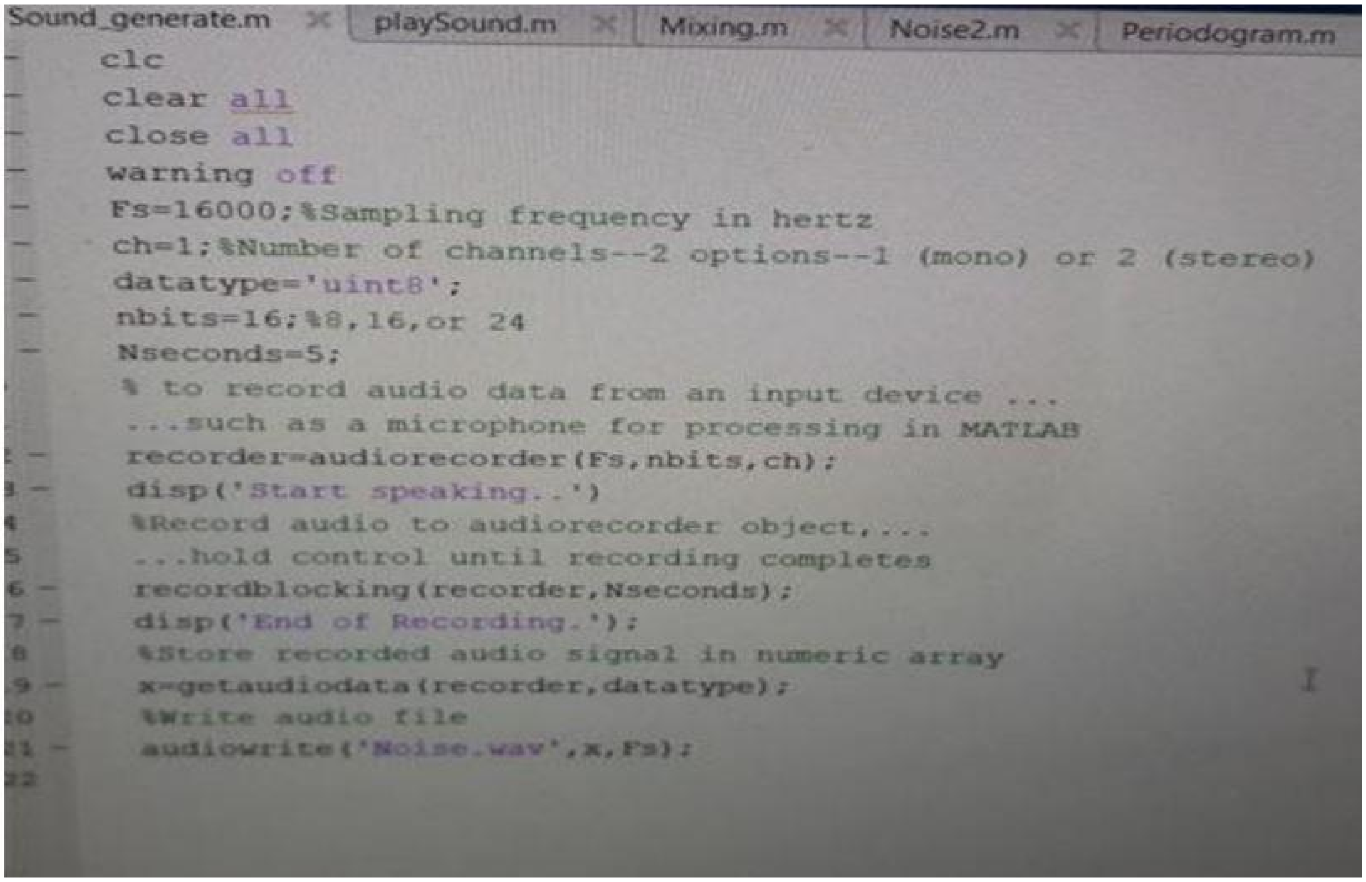
Programmable Embedded Systems
Assignment 6

Submitted by
Pratyush Jaiswal
18EE35014

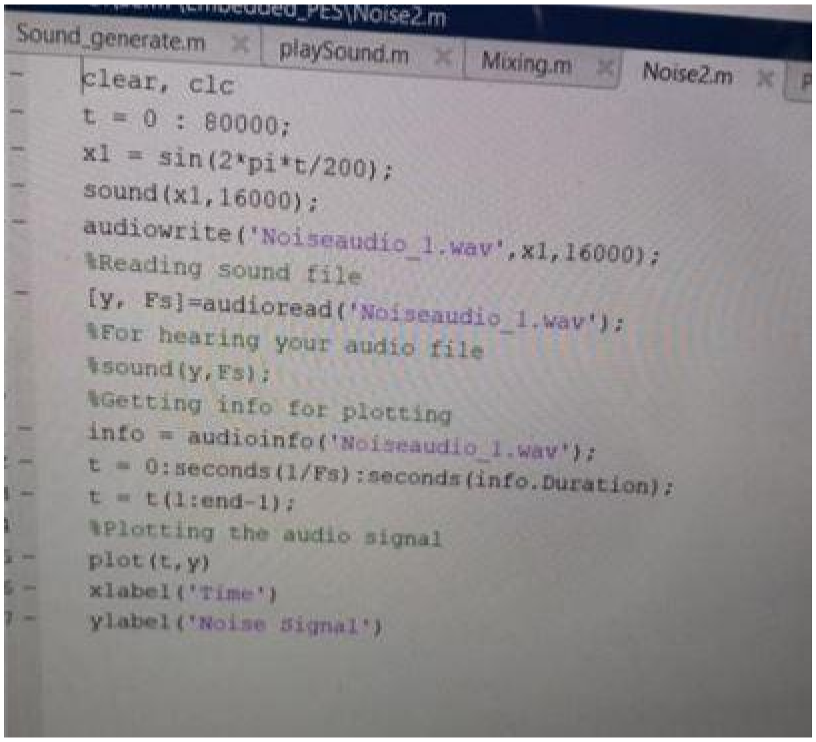
Record separately the AC noise v(n) and your speech s(n) at 16 kHz and 16-bit maximum. Then add both the signals to get x(n) = s(n)+v(n). Plot the PSD of signals x(n), s(n), and v(n) using pwelch or periodogram. Design a comb filter to produce the filtered output y(n) to get rid of the AC noise from x(n). Implement the filter using both MATLAB/python and STM32. Then compare the PSD of y(n), x(n), and s(n). Also, compare the PSD of filtered outputs from MATLAB/python and STM32.

Steps followed
(1)
First, the audio was recorded in Matlab at 16KHz,16 bit maximum.
The Matlab code for the same is given below:



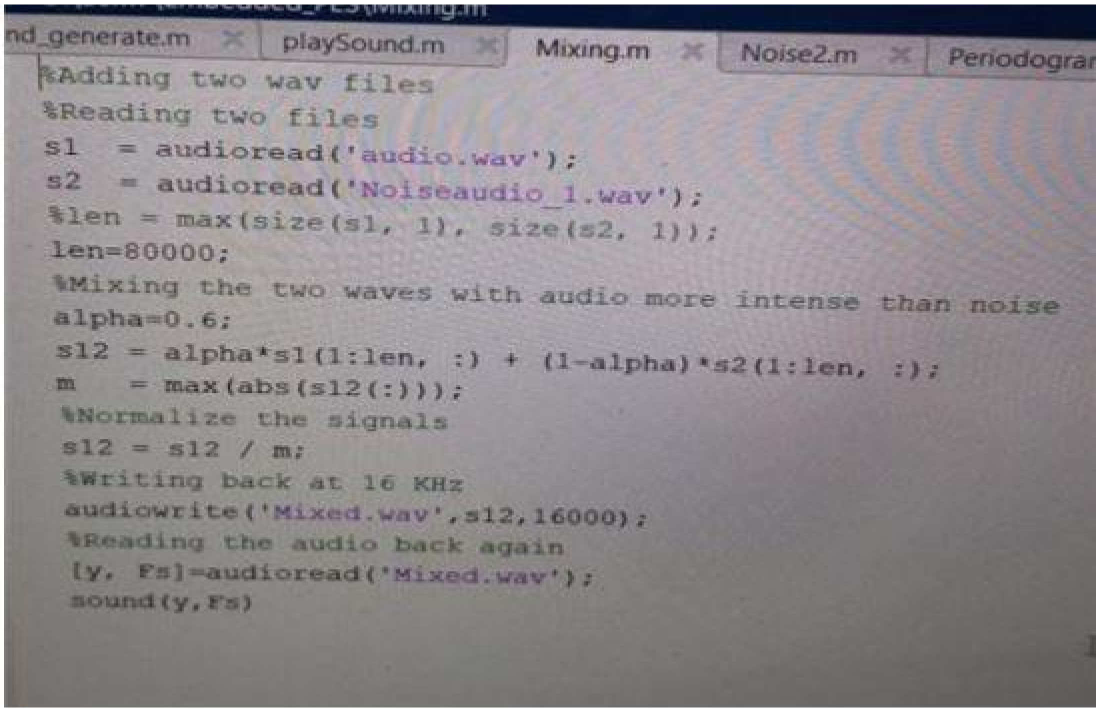
```
clc
clear all
close all
warning off
Fs=16000;%Sampling frequency in hertz
ch=1;%Number of channels--2 options--1 (mono) or 2 (stereo)
datatype='uint8';
nbits=16;%8,16,or 24
Nseconds=5;
% to record audio data from an input device ...
...such as a microphone for processing in MATLAB
recorder=audiorecorder(Fs,nbits,ch);
disp('Start speaking..')
%Record audio to audiorecorder object,...
...hold control until recording completes
recordblocking(recorder,Nseconds);
disp('End of Recording.');
%Store recorded audio signal in numeric array
x=getaudiodata(recorder,datatype);
%Write audio file
audiowrite('Noise.wav',x,Fs);
```

(2)
Now we have to generate the noise that will be coming into the audio.
It is an ac sinusoidal noise with angular frequency pi/100. It is also recorded at 16 kHz.
The Matlab code for the same is given below:



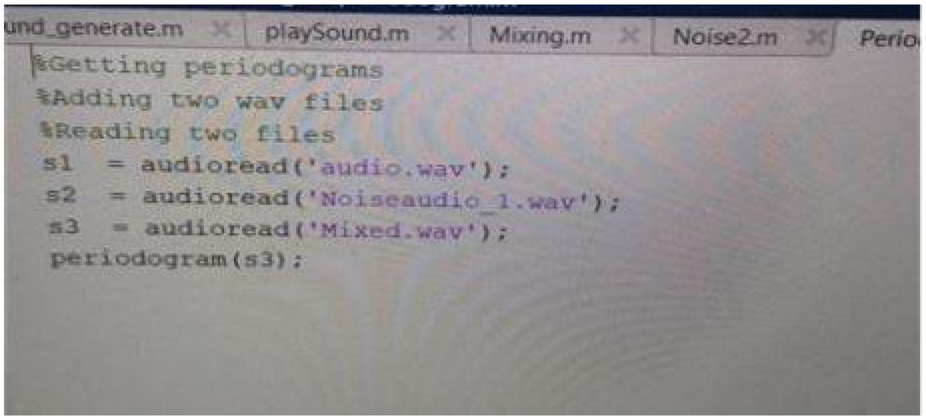
```
clear, clc
t = 0 : 80000;
x1 = sin(2*pi*t/200);
sound(x1,16000);
audiowrite('Noiseaudio_1.wav',x1,16000);
%Reading sound file
[y, Fs]=audioread('Noiseaudio_1.wav');
%For hearing your audio file
%sound(y,Fs);
%Getting info for plotting
info = audioinfo('Noiseaudio_1.wav');
t = 0:seconds(1/Fs):seconds(info.Duration);
t = t(1:end-1);
%Plotting the audio signal
plot(t,y)
xlabel('Time')
ylabel('Noise Signal')
```

(3)
Now both the above generated audio files (in ***wav*** format) are mixed to generate the noisy audio data. The noise can be felt in the background while the audio plays.
The Matlab code for the same is given below:



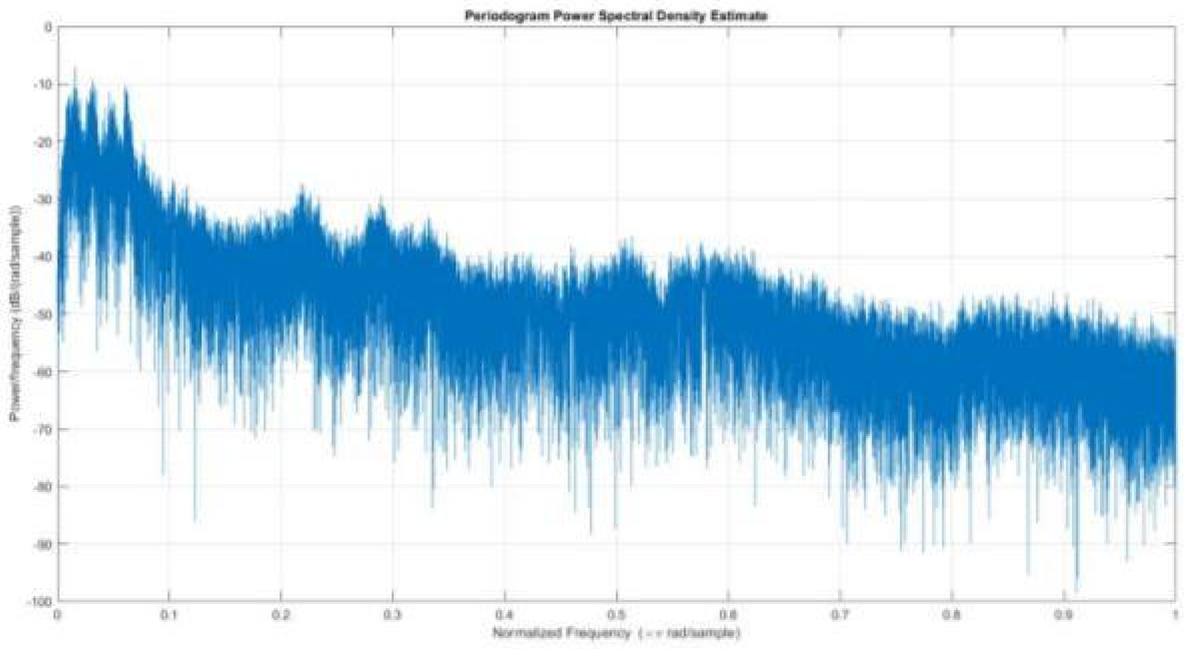
```
%Adding two wav files
%Reading two files
s1  = audioread('audio.wav');
s2  = audioread('Noiseaudio_1.wav');
%len = max(size(s1, 1), size(s2, 1));
len=80000;
%Mixing the two waves with audio more intense than noise
alpha=0.6;
s12 = alpha*s1(1:len, :) + (1-alpha)*s2(1:len, :);
m   = max(abs(s12(:)));
%Normalize the signals
s12 = s12 / m;
%Writing back at 16 KHz
audiowrite('Mixed.wav',s12,16000);
%Reading the audio back again
[y, Fs]=audioread('Mixed.wav');
sound(y,Fs)
```

(4)
After this, the periodograms of each signal are being plotted, which is the power spectral density estimate used for spectral analysis.

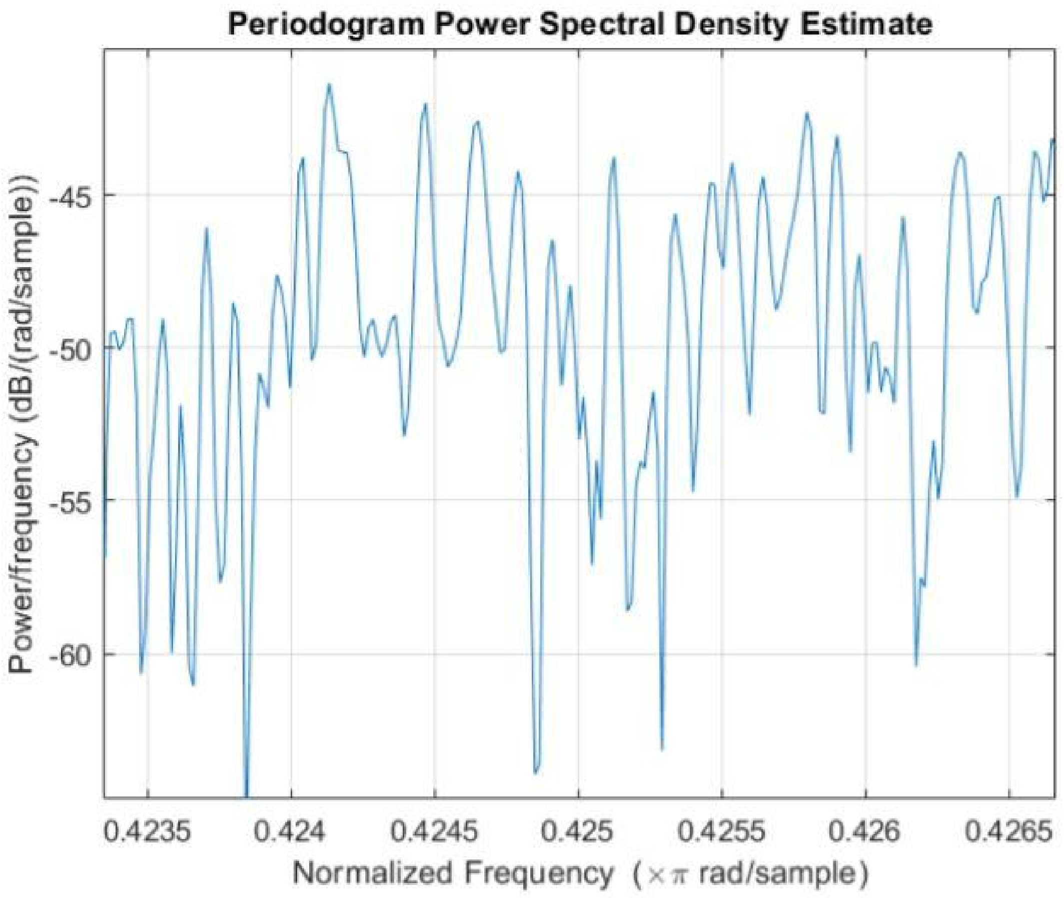


```
%Getting periodograms
%Adding two wav files
%Reading two files
s1  = audioread('audio.wav');
s2  = audioread('Noiseaudio_1.wav');
s3  = audioread('Mixed.wav');
periodogram(s3);
```

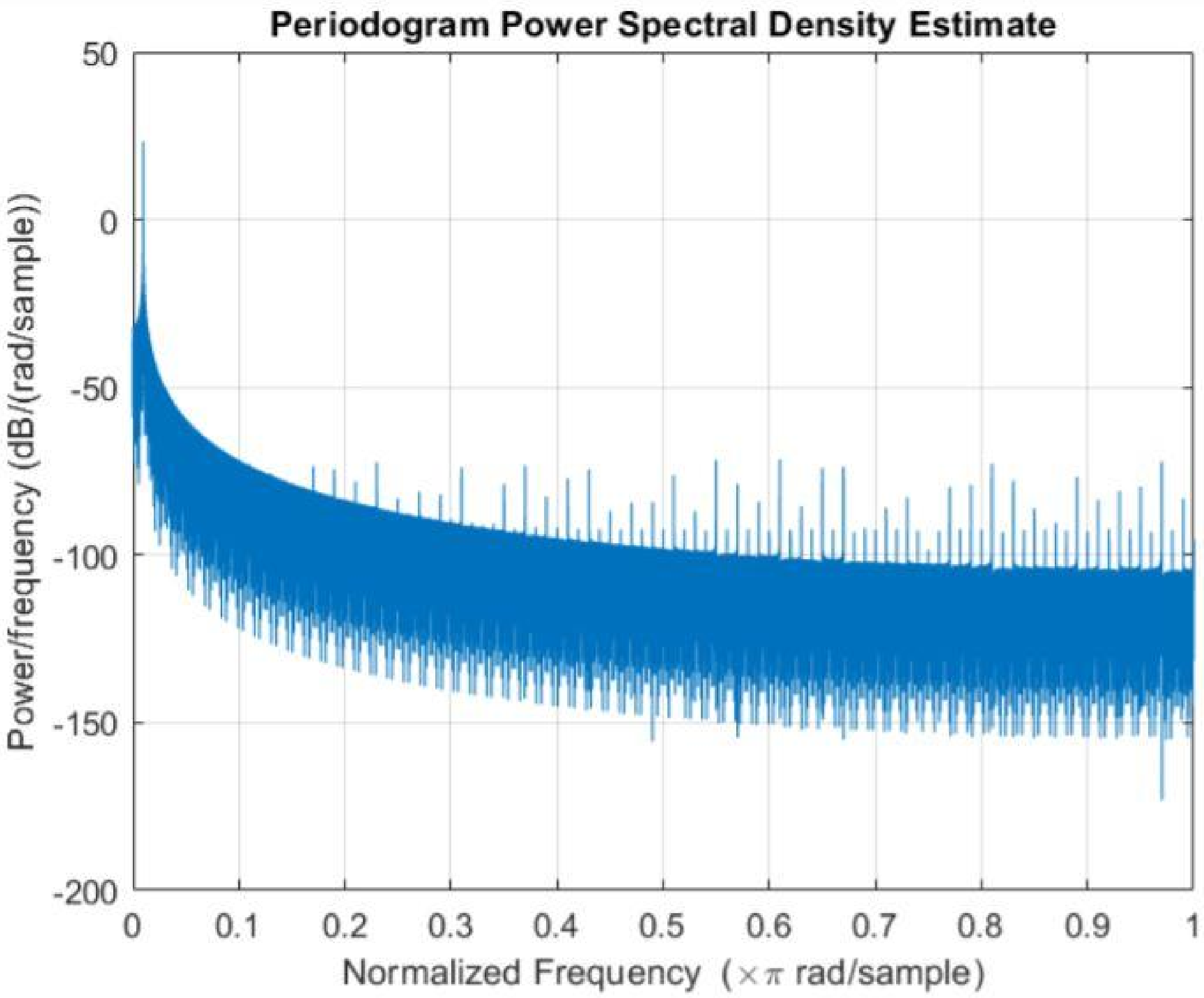
(5)
Now the last code reads the audio files and generate the periodograms for each signal. The first two images are those of the audio signal, the second one being the zoomed in version.

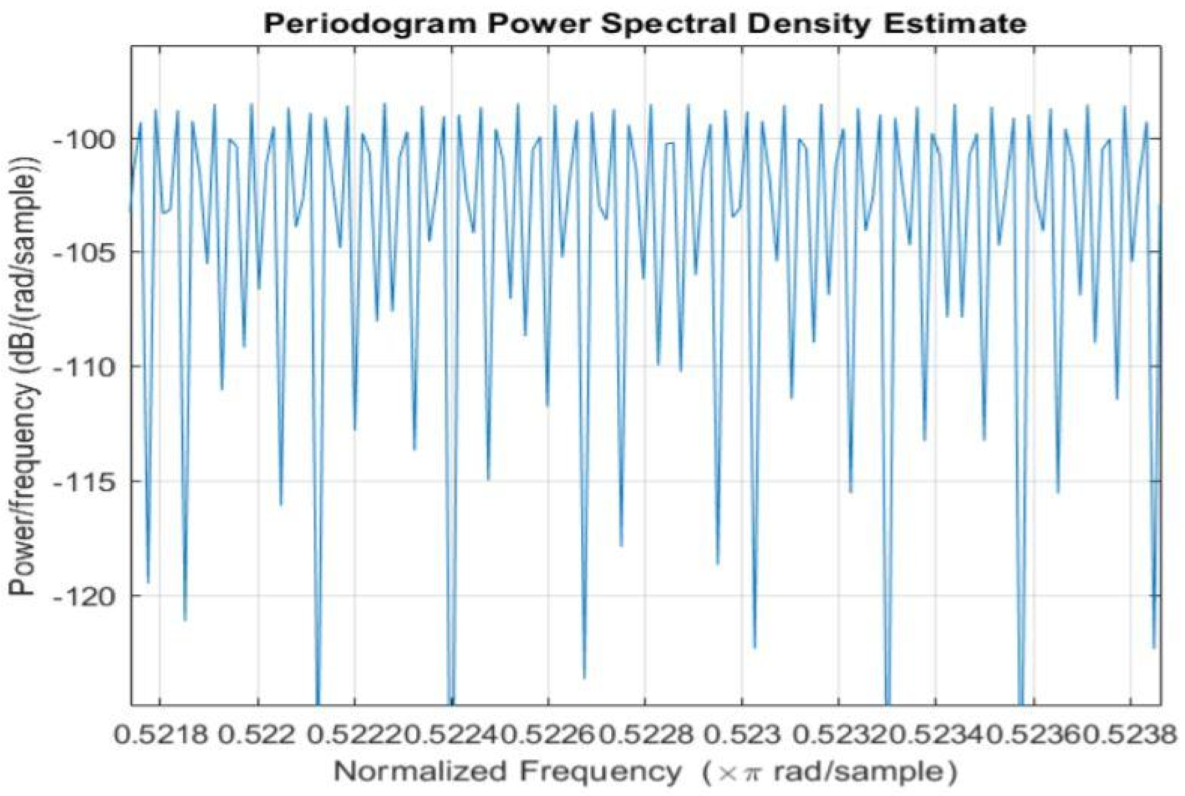


Periodogram Power Spectral Density Estimate
Power/frequency (dB/(rad/sample))
Normalized Frequency (×π rad/sample)
0
-10
-20
-30
-40
-50
-60
-70
-80
-90
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

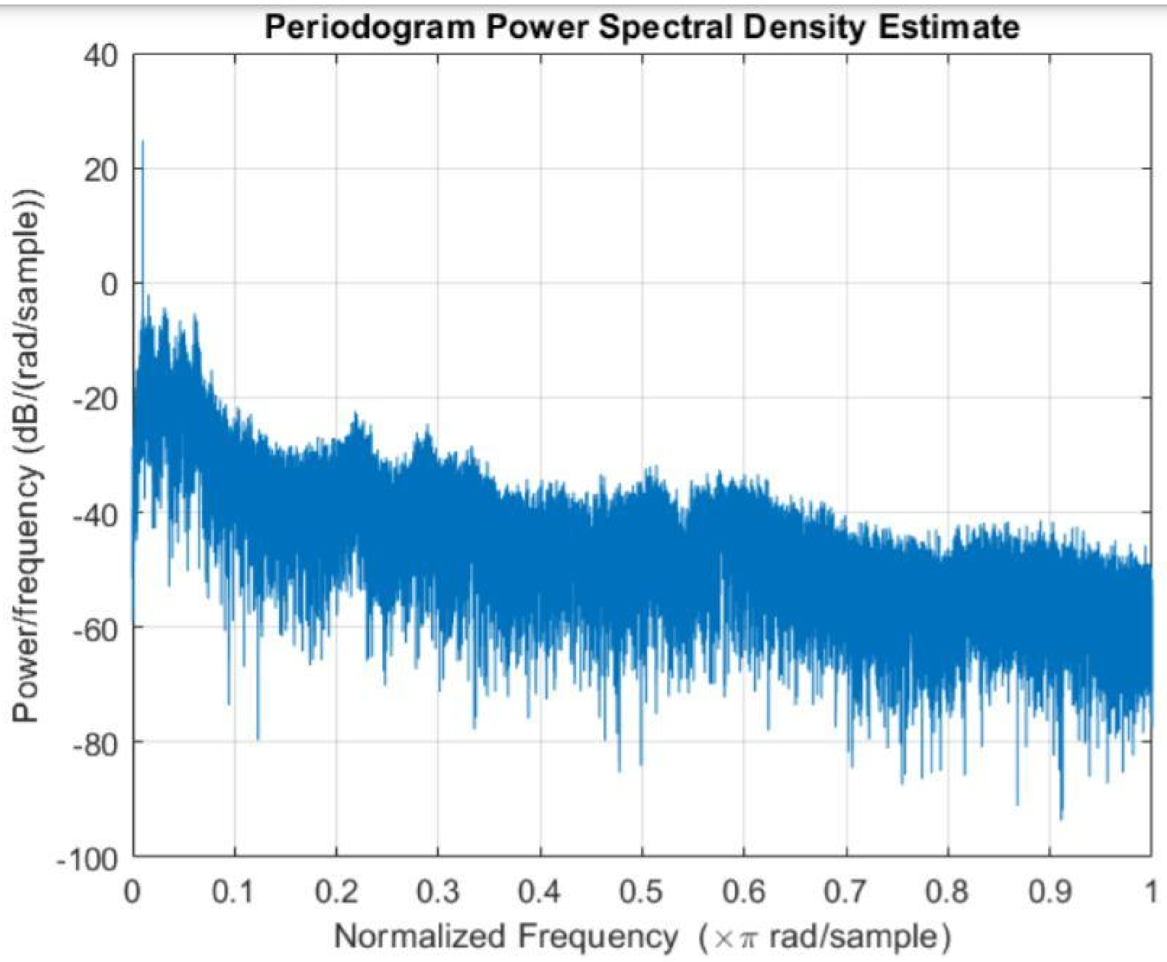


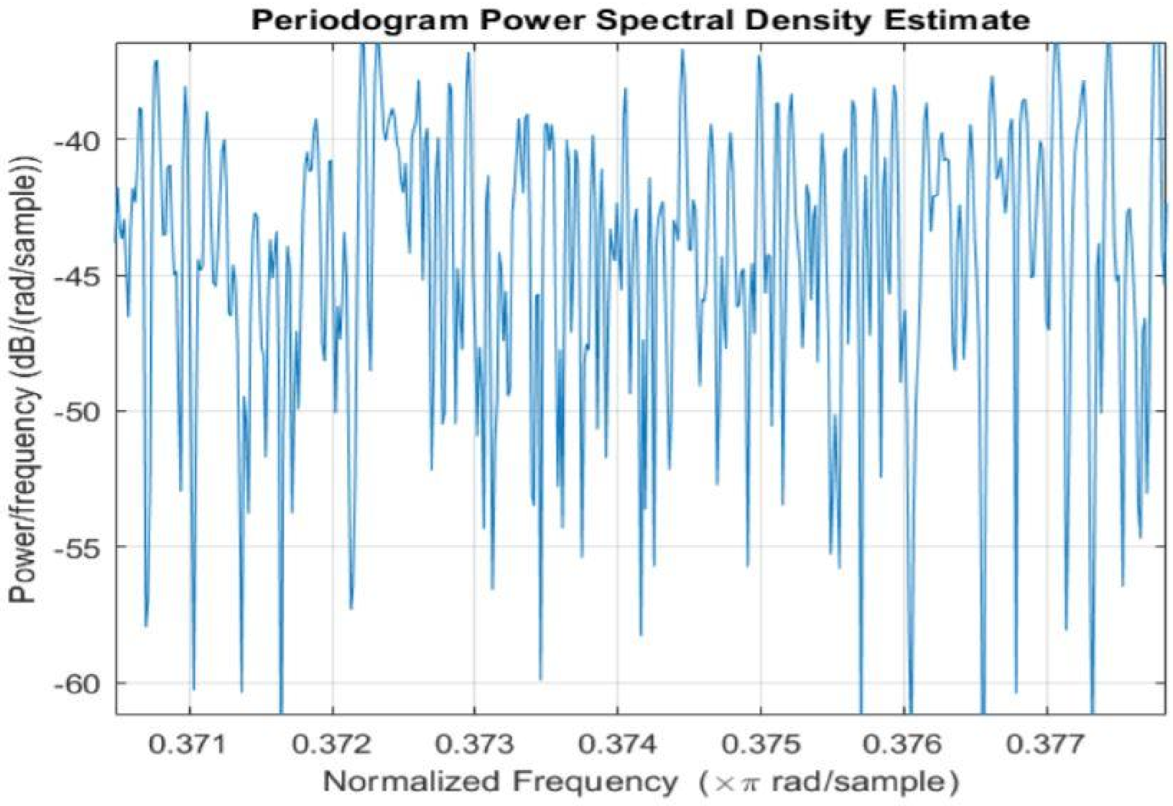
The following two images are the periodogram PSD estimates of the Noise that has been added. The second one zooms into the first one.


Periodogram Power Spectral Density Estimate
Power/frequency (dB/(rad/sample))
Normalized Frequency (×π rad/sample)
50
0
-50
-100
-150
-200
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

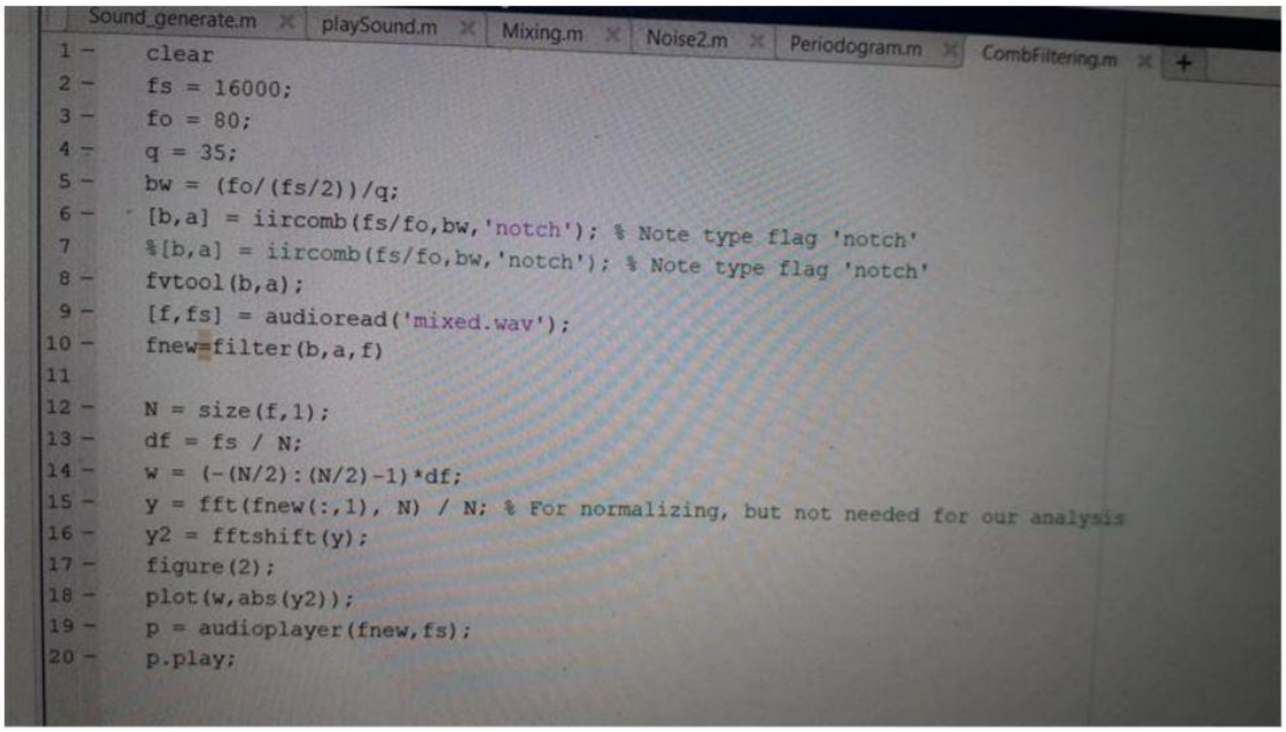


The final two images are the PSD periodogram estimates of the mixed-signal which has the audio with the noise imposed.


Periodogram Power Spectral Density Estimate
Power/frequency (dB/(rad/sample))
Normalized Frequency (×π rad/sample)
40
20
0
-20
-40
-60
-80
-100
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

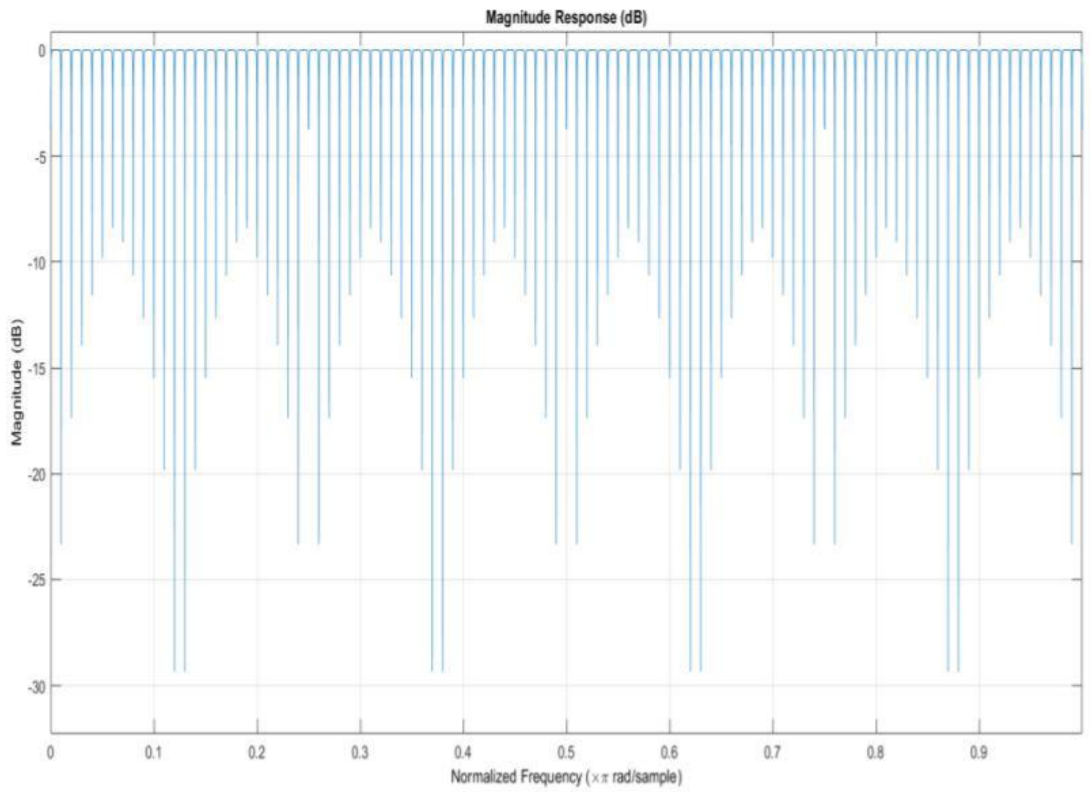


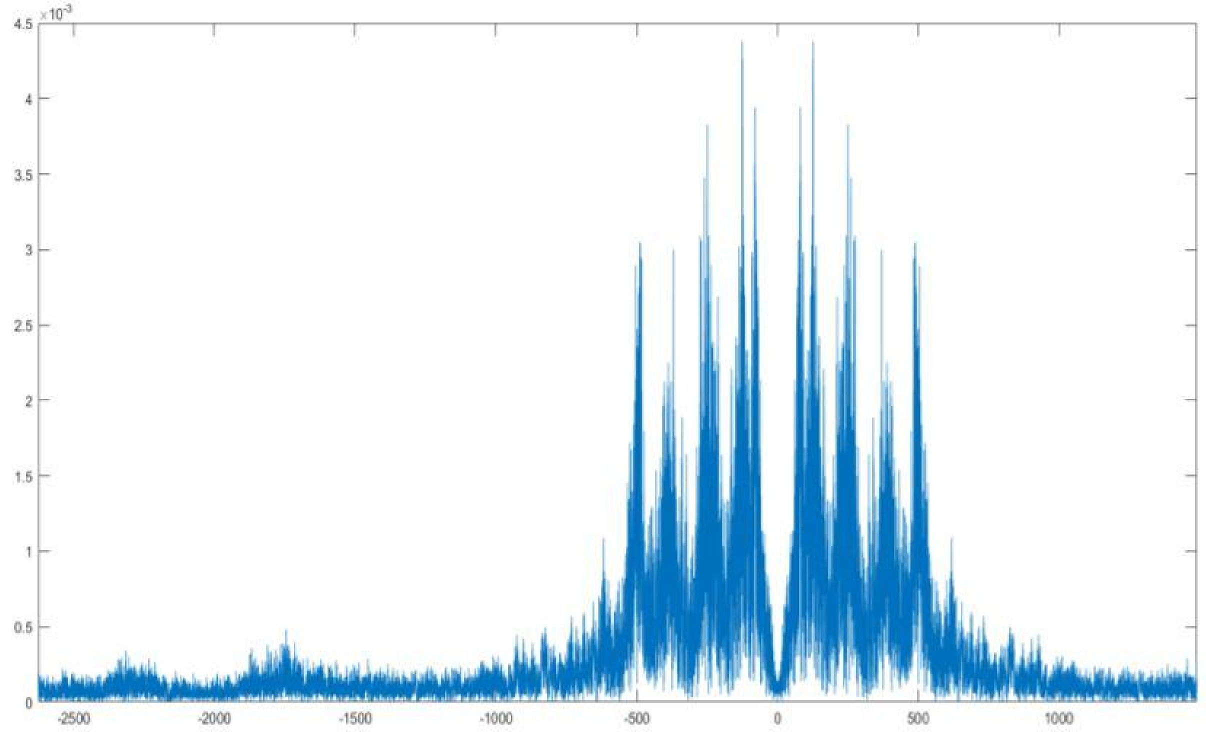
Now the signals are there we need to pass the signals through comb filter to get back the signal free from noise. The Matlab Code for applying the comb filter is given here in this code. It is to be noted that the noise signal is approximately taken to be spaced at 80 Hz and the sampling frequency of the signal to be 16 kHz and the sound after filtering has slight noise but it has been attenuated a lot.



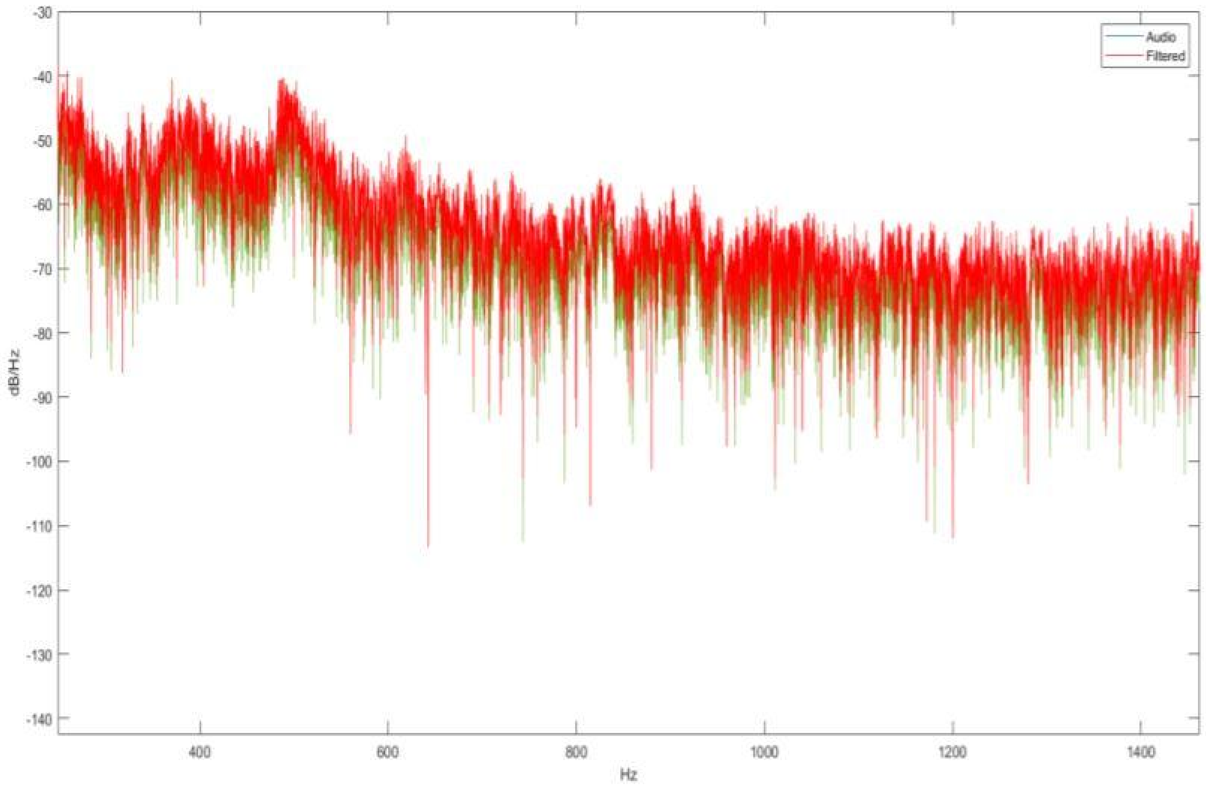
```
clear
fs = 16000;
fo = 80;
q = 35;
bw = (fo/(fs/2))/q;
[b,a] = iircomb(fs/fo,bw,'notch'); % Note type flag 'notch'
%[b,a] = iircomb(fs/fo,bw,'notch'); % Note type flag 'notch'
fvtool(b,a);
[f,fs] = audioread('mixed.wav');
fnew=filter(b,a,f)

N = size(f,1);
df = fs / N;
w = (-(N/2):(N/2)-1)*df;
y = fft(fnew(:,1), N) / N; % For normalizing, but not needed for our analysis
y2 = fftshift(y);
figure(2);
plot(w,abs(y2));
p = audioplayer(fnew,fs);
p.play;
```

The comb filter obtained using the fdatool is given below

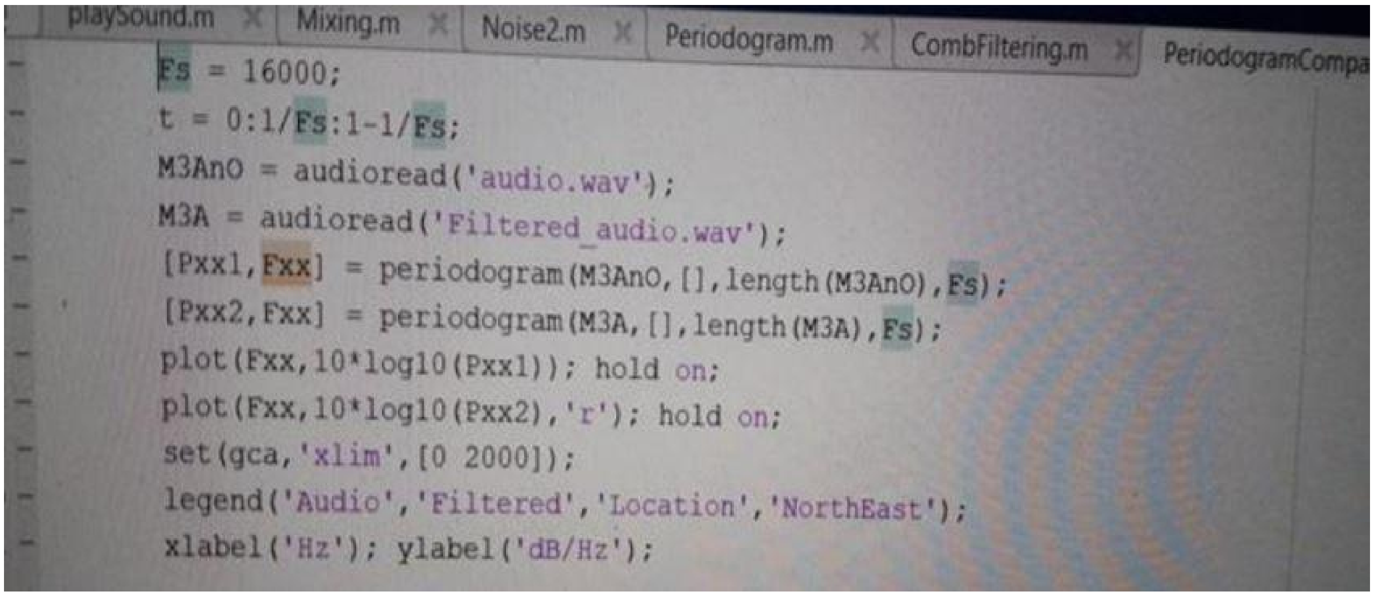




A comparison of the audio signal and filtered audio signal is given below as follows



The red signal is the periodogram of the filtered audio signal and the greenish-yellow color is the periodogram of the original signal. The code generating this plot is given by



```
Fs = 16000;
t = 0:1/Fs:1-1/Fs;
M3AnO = audioread('audio.wav');
M3A = audioread('Filtered_audio.wav');
[Pxx1,Fxx] = periodogram(M3AnO,[],length(M3AnO),Fs);
[Pxx2,Fxx] = periodogram(M3A,[],length(M3A),Fs);
plot(Fxx,10*log10(Pxx1)); hold on;
plot(Fxx,10*log10(Pxx2),'r'); hold on;
set(gca,'xlim',[0 2000]);
legend('Audio','Filtered','Location','NorthEast');
xlabel('Hz'); ylabel('dB/Hz');
```

It can be seen that the filtering takes place under the allowed margin error. The noise gets almost attenuated and the audio can be heard.