Spring 2015
EE 4377
Filter Testing Using an Audio File

Dr. Aslan

Project Introduction

I assume you have a noisy sound file that you want to eliminate the noise using a High Pass Filter.

STEP I

I will use an audio file (project.wav) and later add a noise into that file. MATLAB opens .wav file. If you have .mp3, .wma, etc. format you need to convert them to .wav file. We need to load project.wav file and listen it. This file does not have any noise.

Listing 1: Playing project.wav file in MATLAB - `File_01.m`

```
[file_no_noise,fs]=wavread('project.wav'); %loads test.wav file as vector ...
    (file_no_noise) and sampling frequency fs.
size(file_no_noise) %this will show the size of the sound file. If you see this file has two ...
    columns (stereo has two channels). We just need one. I will only get one of the channels. fs ...
    is 44100 Hz (most recording will be sampled this value).
sound(file_no_noise,fs) %This will play the audio file based on fs.
file_mono=file_no_noise(:,1);  %This will only take one channel.
sound(file_mono,fs) %There is almost no difference. For voice stereo or mono are almost same. ...
    That is one reason most of talk shows are AM stations.
Nos1=cos(2*pi*3*fs/40*(0:length(file_mono)-1)/fs)'; %Create the first noise (what is the ...
    frequency of the signal).
sound(Nos1,fs)  %Listen the noise you have just generated.
figure (1)
subplot 211 %We can plot sound and noise using spectrogram. It will be.
spectrogram(Nos1,512,[],[],fs);colorbar %time vs frequency plot. It is useful to see the
subplot 212 %frequencies.
spectrogram(file_mono,512,[],[],fs);colorbar
figure (2)  %The second figure for to see magnitude vs time for sound
subplot 211 %and noise.
plot(file_mono)
subplot 212
plot(Nos1)
Nos2=rand(1,length(file_mono)); %Create a second noise. This is now a noise with wide range
sound(Nos2,fs)  %of frequencies. Play and plot this noise. You will see its
figure (3)  %frequency values.
subplot 211
plot(Nos2)
subplot 212
spectrogram(Nos2,512,[],[],fs);
```

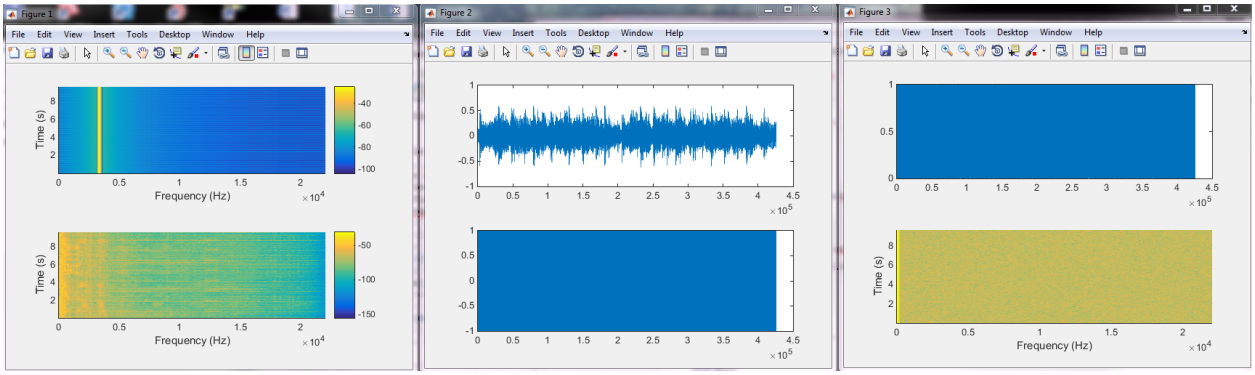

Figure 1: File_01.m plots

Now we can see the frequency components in this file using FFT.

Listing 2: Frequency components - `File_02.m`

```
X_audio=fft(file_mono); % FFT of the audio file
X_Nos1=fft(Nos1);   % FFT of the first noise
f=((-length(X_audio)/2):1:(length(X_audio)/2-1))*(fs/length(X_audio));  % [-fs/2, fs/2] range
plot(f/1000,fftshift(abs(X_audio)/max(abs(X_audio))))   % Plot of first FFT (normalized)
hold on
plot(f/1000,fftshift(abs(X_Nos1)/max(abs(X_Nos1))),'r') % Plot of second FFT (normalized)
X_Nos2=fft(Nos2);
figure(2)
plot(f/1000,fftshift(abs(X_Nos2)/max(abs(X_Nos2))),'r') %Plot of second FFT (normalized)The ...
     second noise has high DC value and wide spectrum. I will stop low frequency values. This ...
     will create second noise   as   high    frequency noise
```

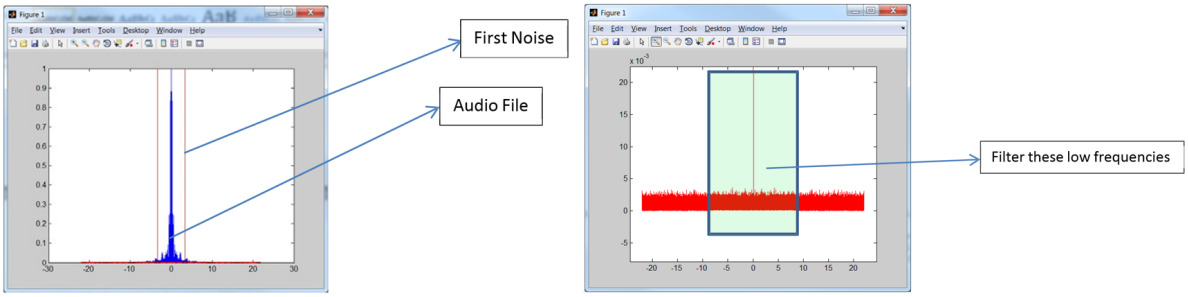

Figure 2: File_02.m plots

Listing 3: High Pass Filter - `File_03.m`

```
% HIGH PASS FILTER DESIGN (I Will use one of the filter we used in the class):
fs=44100;
N=151;  %change this with your N
omg=0:pi/1000:pi;
n=-(N-1)/2:1:(N-1)/2; n((N-1)/2+1)=0.0000001;
h1=sin(n*0.75*pi)./(n.*pi);
w=0.54+0.46*cos(2*pi*n/(N-1));  % Change it for your window function omg_f0=pi;
omg_f0=pi; % For High Pass Filter
c=cos(omg_f0*n);
h=w.*h1.*c; syms z;
h_z=ones(1,length(h));
h_z=h_z*z;
m=0:1:(length(h)-1); h_z=h_z.^m; h_z=h_z.*h; h_z=sum(h_z);
h_w=subs(h_z,exp(-j*omg)); plot(0.5*fs*omg/(1000*pi),20*log10(abs(h_w))); grid on
% From this figure I can say that Band pass filter only passes f> 5 kHz. From figure
% above we want to pass 10 kHz and above. We can adjust our filter to do that. I want
% to first apply this filter to my Nos2 file and apply FFT and see the result first.
% We apply filter to our Noise using convolution.
```

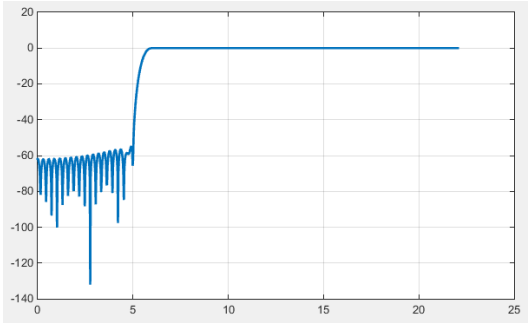

Figure 3: File_03.m plot

Listing 4: Filtered Noise - `File_04.m`

```
Nos2_filtered=conv(Nos2,h); % This is the output of the noise file with the filter above is applied.
size(Nos2)  % The convolution changes the number of samples. The Nos2 has 337920 but
size(Nos2_filtered) % The Nos2_filtered has 338070 samples.
sound(Nos2_filtered,fs);    % Listen unfiltered and filtered noises to see the diffrence.
sound(Nos2,fs);
```

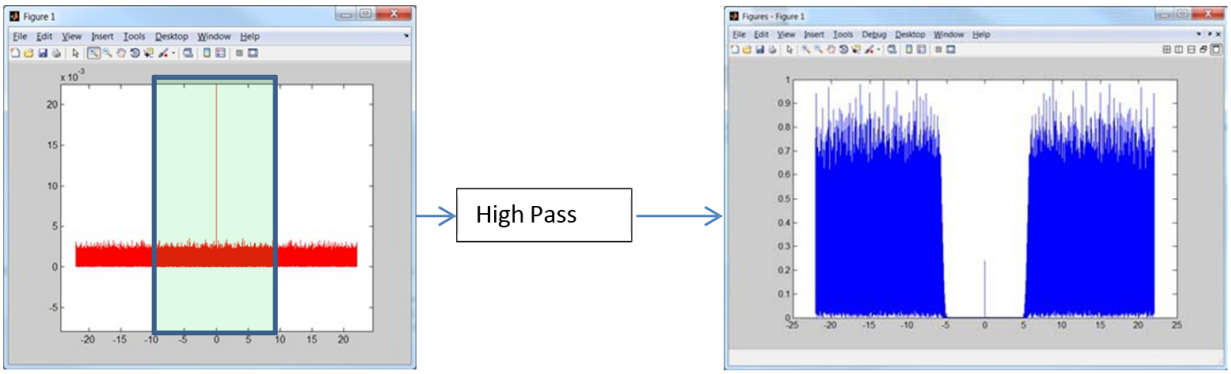

Figure 4: HP filtered noise signal

Listing 5: Writing Sound File

```
%If you want to change the fc to 10 kHz or other values you need to change your Om1 to proper
%value.(Try the same exercise and make the filter filters 10 kHz and lower)
%Now finally I can create our noisy audio.
Nos2_filtered= Nos2_filtered(1:337920); %Adjust the filtered noise size (changed due convolution)
Noisy_Audio=file_mono+Nos2'+Nos1;   %Add file and noise all together.
sound(Noisy_Audio,fs)   %Listen to sound with noise.
spectrogram(Noisy_Audio,512,[],[],fs);colorbar  %See the noisy Audio. You may realize the where ...
    the problem
wavwrite(Noisy_Audio,fs,32,'Noisy_file_1.wav')  % Save file as .wav file. You can use this file ...
    later.
sound(Noisy_Audio,fs)
```

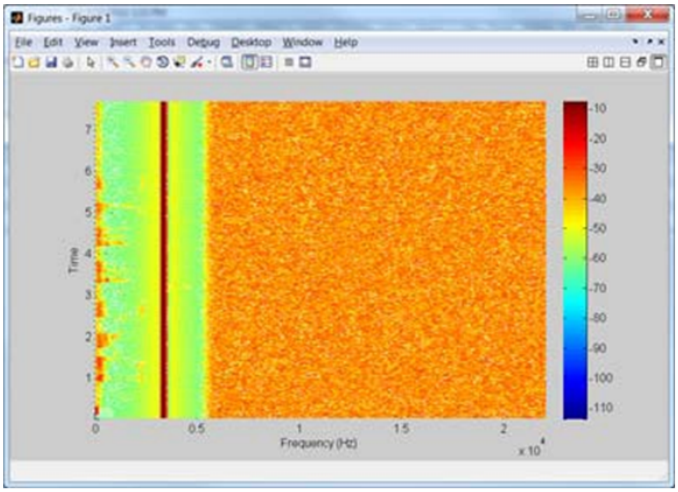

Figure 5: Spectrogram of Noisy File

STEP II

Listing 6: Noisy File with Different Sampling Rate

```
% Now we can filter noise from the file and get the message. We know that noise mostly high ...
    frequency and that cause problem. The original message frequency looks less than 2000 Hz. We ...
    will design a low pass filter that passes frequencies less than 5000 and later we will ...
    change that frequency. After that we will increase the order of filter for better response ...
    (cascade couple of times to increase stop band response).
[Noisy_file,fs]=wavread('Noisy_file_1.wav'); % Open the file noise. This File is the one we ...
    created in step 1.
sound(Noisy_file,fs) % Now play the sound. This is really noisy.
sound(Noisy_file,2*fs)  % Play the sound by changing sampling frequency.
sound(Noisy_file,10*fs) % Explain what happened.
sound(Noisy_file,20*fs)
sound(Noisy_file,fs,4)  % change bits. What happened?
sound(Noisy_file,fs,2)
sound(Noisy_file,fs,16)
% Low Pass Filter < 5 kHz
f=fs; N=33; %change this with your N
omg=0:pi/100:pi;
n=-(N-1)/2:1:(N-1)/2;
n((N-1)/2+1)=0.0000001;
h1=sin(n*0.250*pi)./(n.*pi);
w=0.5+0.5*cos(2*pi*n/(N-1)); % Change it for your window function
h=h1.*w;
syms z;
h_z=ones(1,length(h));
h_z=h_z*z;
m=0:1:(length(h)-1);
h_z=h_z.^m; h_z=h_z.*h; h_z=sum(h_z);
h_w=subs(h_z,exp(-j*omg));
plot(0.5*f*omg/pi,20*log10(abs(h_w))); grid on
```

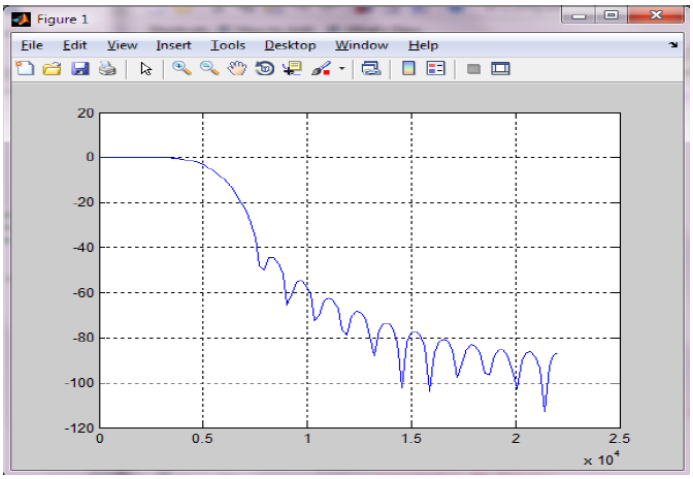

Figure 6: Low Pass Filter

Listing 7: Low Pass Filtered Signal

```
% Now we can apply this FIR low pass filter to our noisy sound and check the result.
sound_filtered=conv(Noisy_file,h); % This is how you apply your impulse response to input
size(sound_filtered) % The convolution changes the number of samples. The Noisy_file has 337920
size(Noisy_file _filtered) % The sound_filtered_has 337952 samples.
sound(Noisy_file,fs); % Listen unfiltered and filtered sounds to see the difference.
sound(sound_filtered,fs);
spectrogram(sound_filtered,512,[],[],fs);colorbar
```

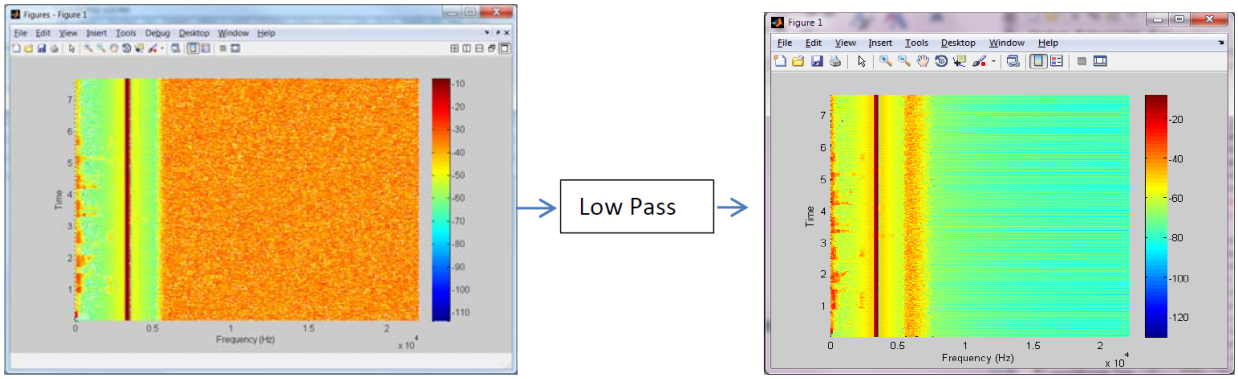

Figure 7: Low Pass Filtered Signal

Listing 8: Better Low Pass Filtered Signal

```
% Low Pass Filter < 2 kHz
f=fs;
N=33; %change this with your N
omg=0:pi/100:pi;
n=-(N-1)/2:1:(N-1)/2;
n((N-1)/2+1)=0.0000001;
h1=sin(n*0.1*pi)./(n.*pi);
w=0.5+0.5*cos(2*pi*n/(N-1)); % Change it for your window function
h=h1.*w;
syms z;
h_z=ones(1,length(h));
h_z=h_z*z;
m=0:1:(length(h)-1);
h_z=h_z.^m;
h_z=h_z.*h;
h_z=sum(h_z);
h_w=subs(h_z,exp(-j*omg));
plot(0.5*f*omg/pi,20*log10(abs(h_w))); grid on
% Above code just before (syms z); code enter following (h=conv(h,h);) and re run the code. ...
    Compare two graphs.
```

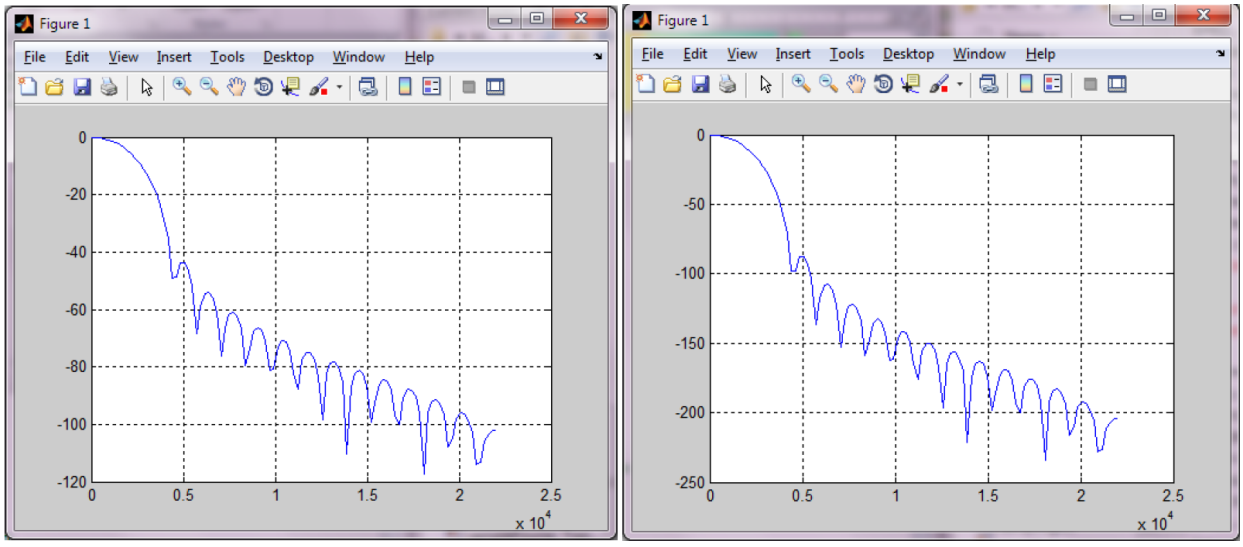

Figure 8: Better Low Pass Filtered Signal

Listing 9: Better Low Pass Filtered Signal

```
% Now apply the filter again with a better filtering response
sound_filtered1=conv(Noisy_file,h); % This is how you apply your impulse response to input
size(sound_filtered1) % The convolution changes the number of samples. The Noisy_file has 337920
size(Noisy_file) % The sound_filtered_has 337952 samples.
sound(Noisy_file,fs); % Listen unfiltered and filtered sounds to see the difference.
sound(sound_filtered1,fs);
spectrogram(sound_filtered1,512,[],[],fs);colorbar
```

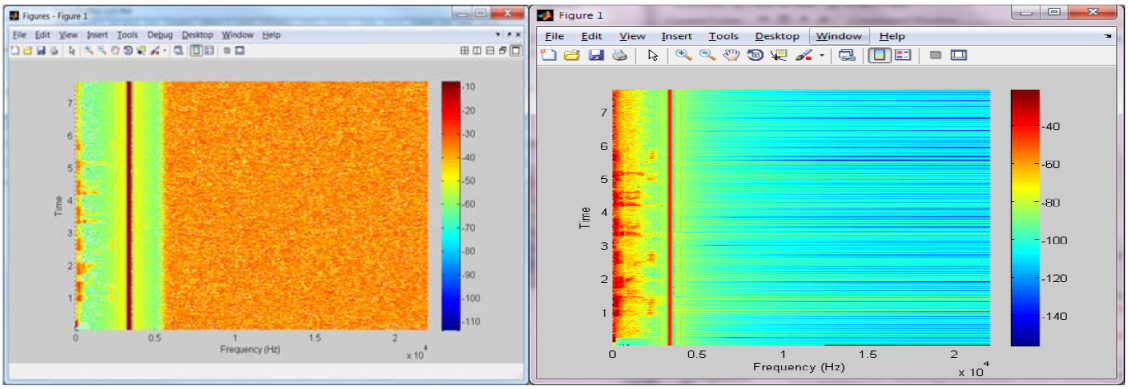

Figure 9: Better Low Pass Filtered Signal

Listing 10: Low Pass Filter < 1.5 kHz

```
omg=0:pi/100:pi;f=fs;N=33; %change this with your N
n=-(N-1)/2:1:(N-1)/2;
n((N-1)/2+1)=0.0000001;
h1=sin(n*0.21)./(n.*pi);
w=0.5+0.5*cos(2*pi*n/(N-1)); % Change it for your window function
h=h1.*w; h=conv(h,h); h=conv(h,h);
syms z;
h_z=ones(1,length(h));
h_z=h_z*z;
m=0:1:(length(h)-1);
h_z=h_z.^m; h_z=h_z.*h;f; h_z=sum(h_z);
h_w=subs(h_z,exp(-j*omg));
plot(0.5*f*omg/pi,20*log10(abs(h_w))); grid on
```

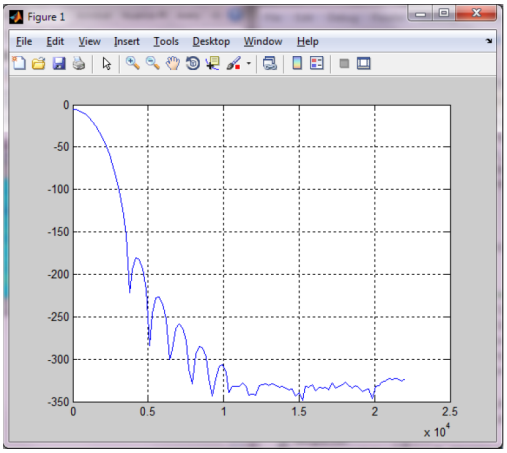

Figure 10: Low Pass Filter < 1.5 kHz

Listing 11: Low Pass Filter < 1.5 kHz

```
% Now apply the filter again with a better filtering response
sound_filtered1=conv(Noisy_file,h); % This is how you apply your impulse response to input
size(sound_filtered1) % The convolution changes the number of samples. The Noisy_file has 337920
size(Noisy_file) % The sound_filtered_has 337952 samples.
sound(Noisy_file,fs); % Listen unfiltered and filtered sounds to see the difference.
sound(sound_filtered1,fs);
spectrogram(sound_filtered1,512,[],[],fs);colorbar
```

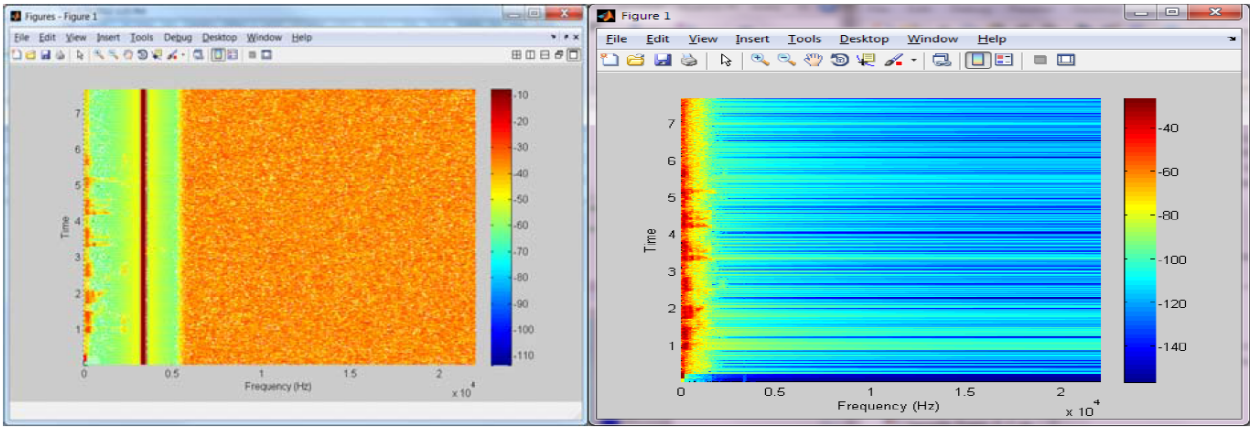

Figure 11: Low Pass Filter < 1.5 kHz