WARSAW UNIVERSITY OF TECHNOLOGY

DIGITAL SIGNAL PROCESSING

Group 1

Nikolai Golovan
Debaleena Misra

FREQUENCY-BASED FILTERING

TABLE OF CONTENTS

FREQUENCY-BASED FILTERING

Problem Statement

An audio signal is given that contains 2 types of noises:

- A constant factory noise
- An occasional "useful" signal.

Apply windowed FFT, analyze the spectrogram of signal and propose an algorithm for extraction of the useful signal, preferably based on band-pass filtering. Make visualization of the time-domain signal and its spectrogram, of intermediate and final processing results.

Signal Given

- Glass signal contaminated with noise
- Original Noise signal

Methodology

(1) Time-domain representation of the signals

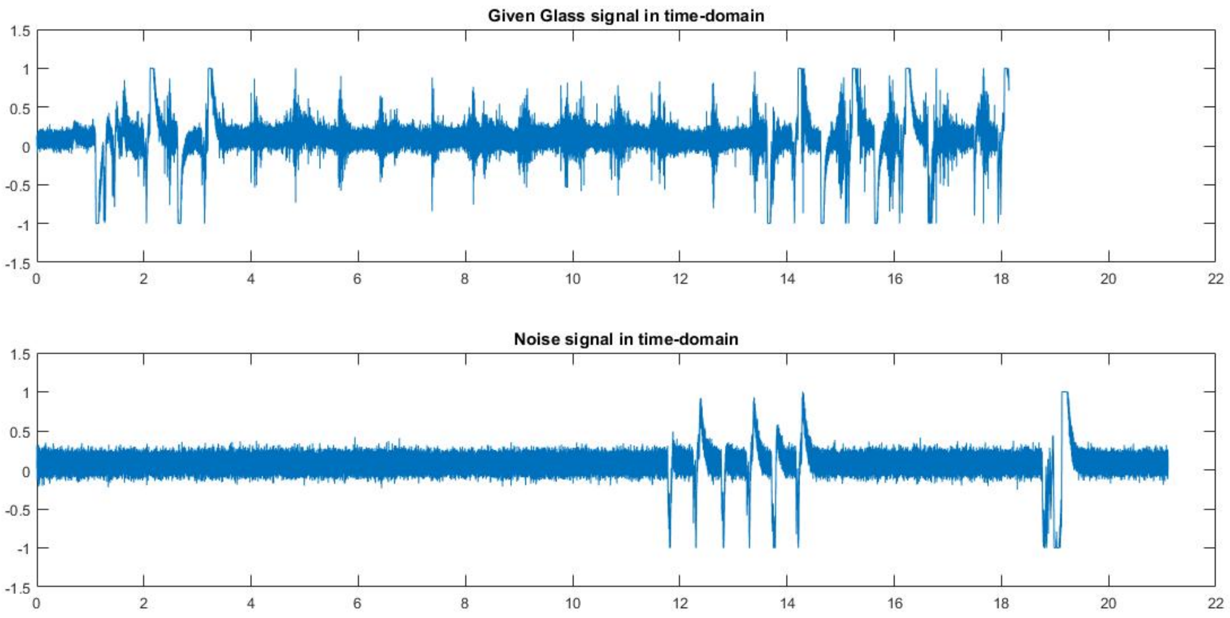


Fig 1: Time-domain representation of Signals

Plotting the two time-domain representations of the given Glass signal (mixed with noise) and the original noise signal, we find that the signal length is smaller than that of the noise, so our main focus will be the signal length (around 18 seconds) only.

(2) Frequency-domain representation of the signals

Now we apply windowed FFT (Fast-Fourier transform) to our signals and view them in the frequency domain. A window size of 1024 points is taken.

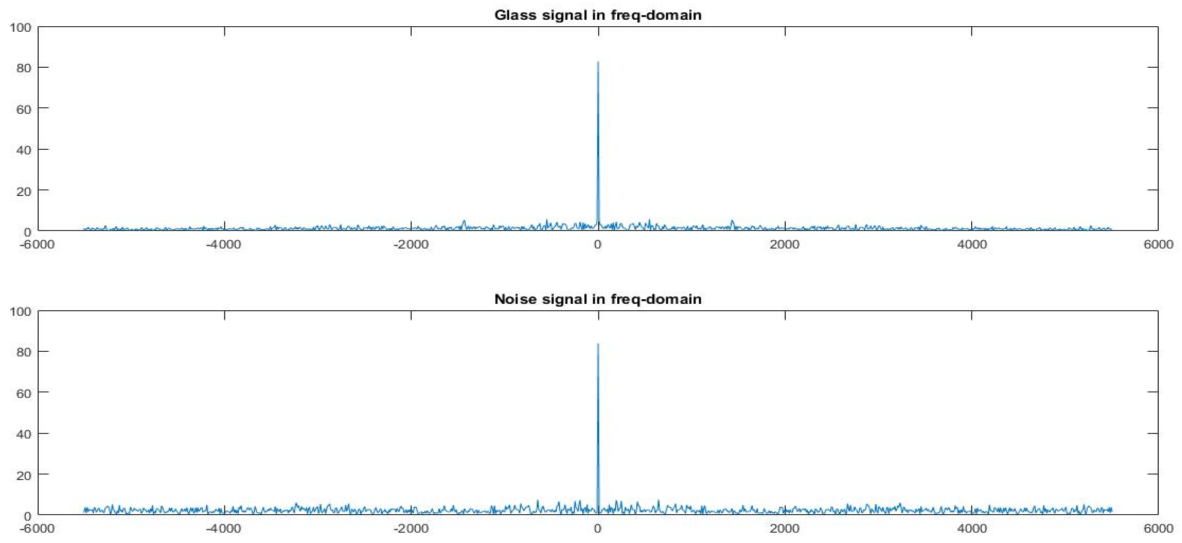


Fig 2: Frequency-domain representations of the signals

The presence of the DC component is evident from the above plots. So we apply FFT again, now on the signals after having removed their DC components and their frequency-domain representations are

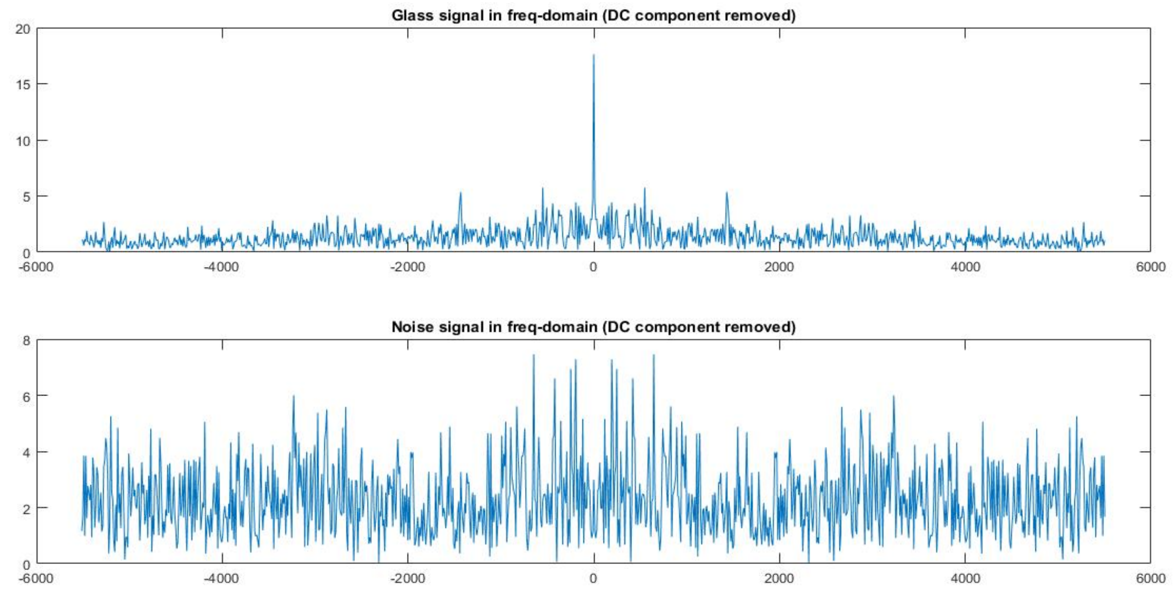


Fig 3: Frequency-domain representation of signals (DC components removed)

The glass signal still has very high amplitude at 0Hz, but it is visibly reduced due to the removal of DC component. From the FFT for glass, we can observe that the primary amplitude peaks are all in the lower range within 2000 Hz, which gives us an intuition of possible choices of frequency for the useful signal, discussed later in the report.

(3) Spectrograms of the signals

Next, we find the spectrogram of the given signals (using Short-Time Fourier Transform). Here, our audio files of contaminated glass and original noise are shown on a graph according to time and frequency, with brightness indicating amplitude

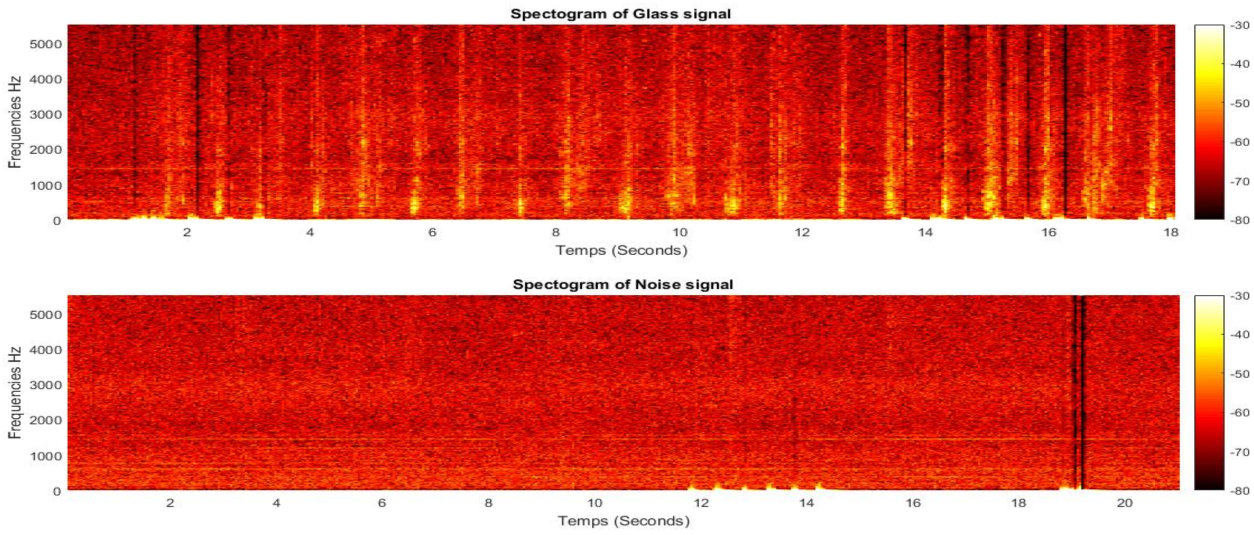


Fig 4: Spectrogram of signals

The amplitude of the frequency components (which in spectra is indicated on the y-axis) is expressed by means of the degree of whiteness (more energy, more whiteness). As formants are frequency regions with a high energy (due to the filter resonances), in the spectrogram they are displayed as light bands. Visually from the heat map distribution we can see that since the energy comes out to be uniform in most part of the spectrum, so we'd now focus only on the higher energy parts (which are our primary areas of interest that we will try to successfully filter later). The vertical yellow lines on the spectrogram are also of consideration to be able to detect the onset of attacks (which can be heard as recurring burst-like events in the audio files). Zooming to the higher energy areas :

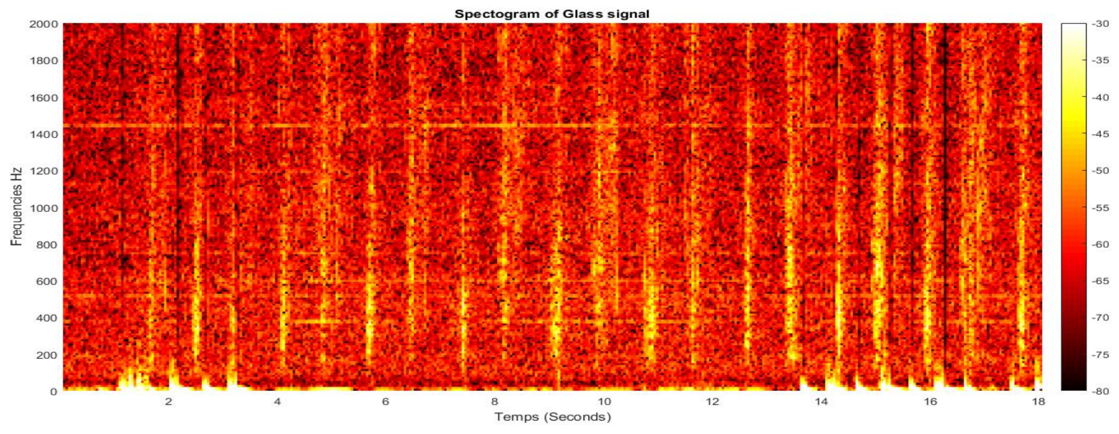


Fig 5: Spectrogram of Glass signal (zoomed)

We find that the main energy content (bright spots of yellow) is concentrated in the lower frequency values (range till 2000Hz seen here) which corroborates our intuition from (section 2) that most of the useful signal is contained in the lower frequency ranges. In the next sections, we propose an algorithm to explore these ranges in more details.

(4) Proposed Algorithm

From the previous sections on energy distributions, we can assume that our primary area of interest is in the lower range of frequencies. However, noise amplitude peaks are also located in that range, so we try to remove them to keep only the useful signal. So we now propose an algorithm to find the right range of frequencies. Our algorithm is based on the method of **Power Spectral Densities (PSD).** Since the PSD describes how the power of a time signal is distributed with frequency, we can approximate the useful frequency values by selecting the bands of frequencies that have useful power content. The power values can be read directly from the power spectrum without having to integrate over an interval. Since PSD and power spectrum are real, they do not contain any phase information.

a) Plot the Power Spectral Density for Glass signal (PSD_g)
b) Plot the Power Spectral Density for Noise Signal (PSD_n)
c) Plot the difference of Powers between the two signal (using the difference of PSDs), which gives us an idea of the Signal-to-Noise-Ratio (SNR) with frequency
d) From (c), identify frequency bands where the SNR values are acceptable (PSDs of Noise > PSDs of Glass) and where the SNR values are dropping (PSDs of Noise > PSDs of Glass)
e) Create filters to successfully retain the bands having acceptable frequencies and block bands having low SNRs

Plots

- Power Spectral Densities

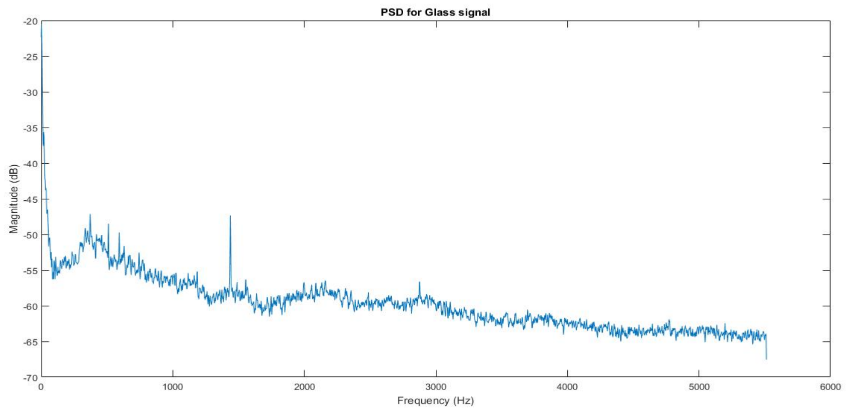


Fig 6: PSD for Glass Signal

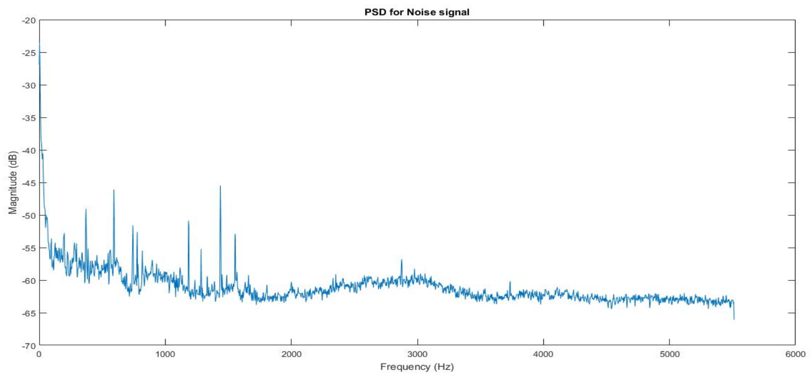


Fig 7: PSD for Noise Signal

- Comparison of PSD for both signals

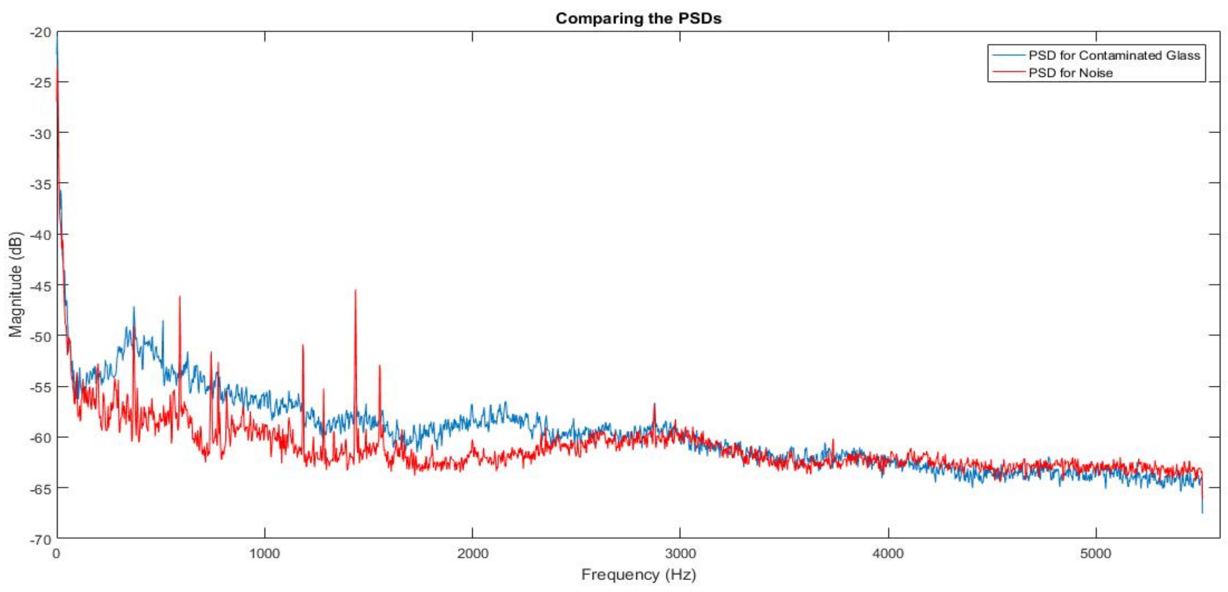


Fig 8: Comparison of PSD of both Glass (in blue) and Noise (in red) signals

- Plotting of the SNR as difference of powers of the two signals

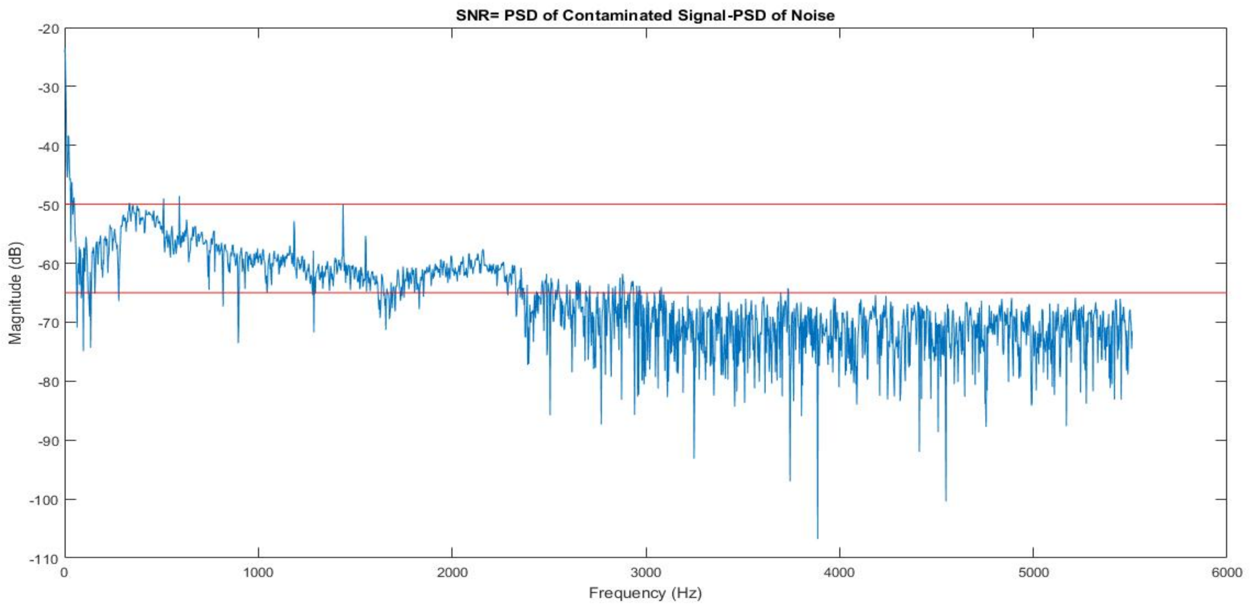


Fig 9: SNR (dB) versus Freq (Hz) using the difference of PSDs of the Glass and Noise signals

We choose a desired range of SNR values as acceptable, as indicated by the reference lines (in red). So our next aim is to find the corresponding frequency ranges for which the SNR values would lie within that area. We take the help of the PSD comparison plots to be able to identify the approximate values of frequency ranges.

(5) Design of Filters

From the PSD graphs one can determine desired filtering frequencies. We view the comparison plot (figure 8) in different ranges of frequency to identify where the power of signal (in blue) is visibly higher than the power of noise (in red). Some of the magnified plots are given below:

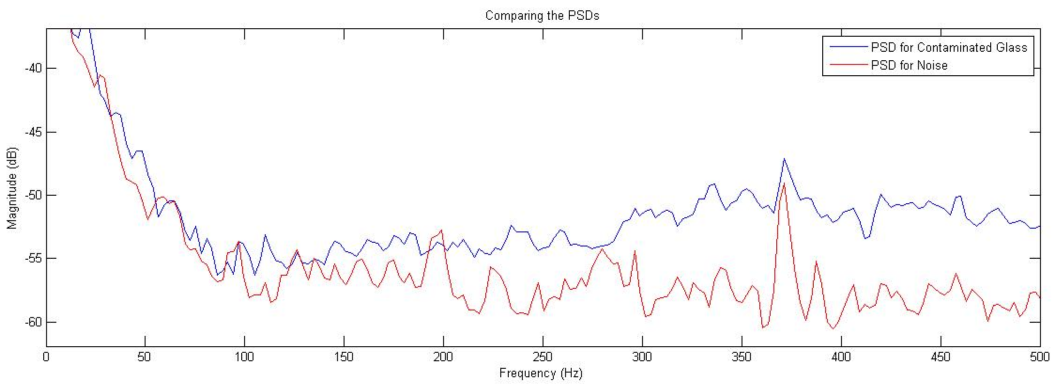
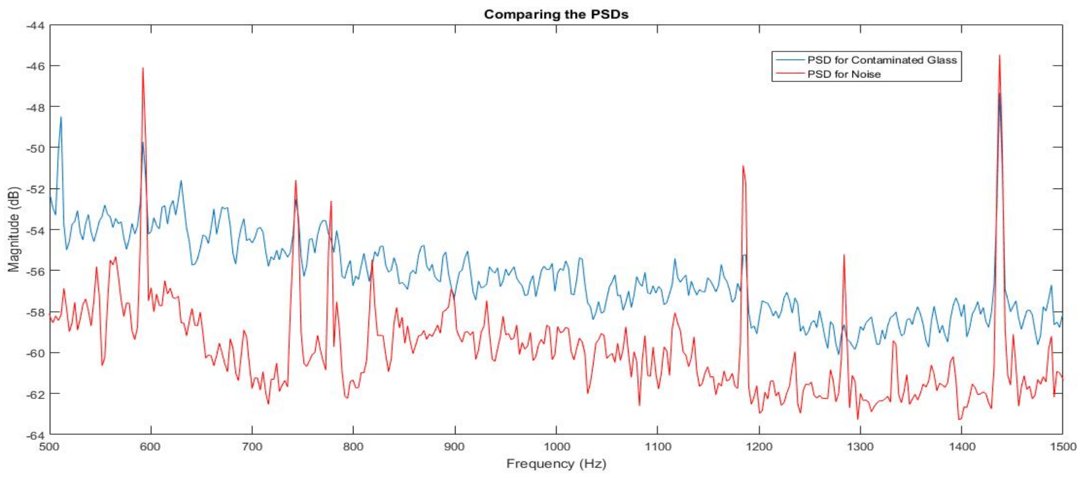


Fig 10: Comparison of PSD in 0-500Hz range



* Fig 11: Comparison of PSD in 500-1500Hz range

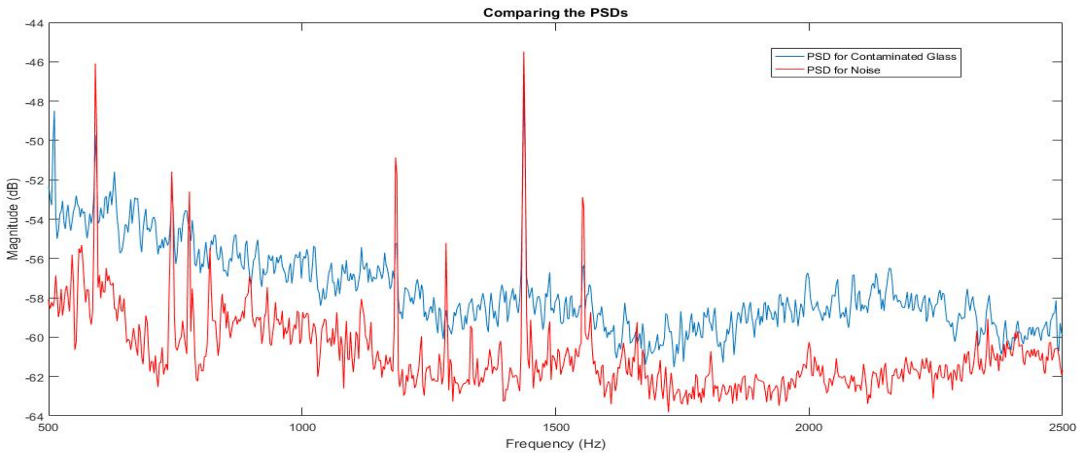


Fig 12: Comparison of PSD in 1500-2500Hz range

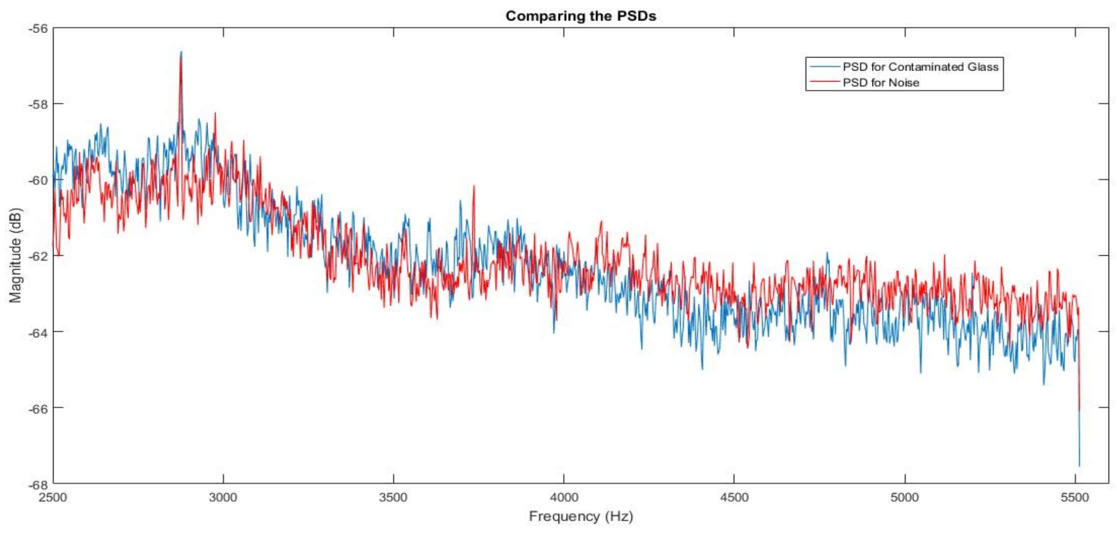


Fig 13: Comparison of PSD from 2500Hz till end

Having identified the useful frequency ranges from the above plots, we next use a combination of low-pass, high-pass and band-stop filters to remove, retain and block frequencies in the bands given below:

1. Low-Pass Filter 2300Hz
2. Band-stop 180-210Hz
3. Band-stop 350-37Hz
4. Band-stop 560-600Hz
5. Band-stop 725-825Hz
6. Band-stop 1160-1570Hz

Choice of filter

It was decided to use a Butterworth IIR filter. With the built-in function using MATLAB, applying different types of filters is a very efficient and time- saving.

The frequency response of the Butterworth filter is maximally flat (i.e. has no ripples) in the pass-band and rolls off towards zero in the stop- band. When viewed on a logarithmic Bode plot, the response slopes off linearly towards negative infinity. A first-order filter's response rolls off at −6 dB per octave (−20 dB per decade) (all first-order low- pass filters have the same normalized frequency response). A second-order filter decreases at −12 dB per octave, a third-order at −18 dB and so on. Butterworth filters have a monotonically changing magnitude function, unlike other filter types that have non-monotonic ripple in the pass- band and/or the stop- band.

Compared with a Chebyshev Type I/Type II filter or an elliptic filter, the Butterworth filter has a slower roll-off, and thus will require a higher order to implement a particular stop- band specification, but Butterworth filters have a more linear phase response in the pass-band than Chebyshev Type I/Type II and elliptic filters can achieve. The Butterworth filter rolls off more slowly around the cutoff frequency than the Chebyshev filter or the Elliptic filter, but without ripple

Applying filters and evaluation of efficiency using a modified SNR metric

It was decided to use a 4th-order filter butter-worth filter because in analog filtering it is difficult to implement high- order filters, so exactly 4th- order filter is considered to be a perfect match between digital and analog implementations.

Calculation of SNR

To determine, how good or bad the filtering is the SNR value is calculated for the filtered signal. This calculation can be done using two different algorithms:

1) Signal-to-noise ratio is defined as the ratio of the power of a signal (meaningful information) and the power of background noise (unwanted signal). Expressed in decibels (dB)

$$P_{signal} = 10log_{10}\left(Amplitude_{signal}{}^{2}\right) dB$$

$$P_{noise} = 10log_{10}\left(Amplitude_{noise}{}^{2}\right) dB$$

$$SNR = 10log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) dB$$

2) If noise power can't be measured by any reason, it is possible to estimate SNR via FFT, but one needs to know the signal's band. The procedure is the same, except FFT frequency indexes should be used, instead of sample numbers while estimating the power. In this case, the power spectral density should be evaluated at first.

However, in our task, the original useful signal has not been provided, so it could not be calculated in the traditional way. So we adopted a modified definition of the SNR – it is expressed as the ratio of the power of the given signal to power of the noise present in it. So, we first find the residual noise present in our filtered signal.

Residual Noise - Filtering is applied to both the contaminated glass signal and the pure noise signal; this will equally get rid of unwanted frequency ranges both the signals. If the original noise signal is also passed through the same stages of filtering as the contaminated glass signal, then it gives us the residual noise and we can compute the residual noise power (having removed all unwanted frequencies) and compute SNR of filtered signal by finding a ratio using power of this residual noise.

And finally the SNR for the filtered signal is compared against the SNR of the given contaminated signal, to be able to comment on whether the noise content has decreased or not.

$$SNR_{start} = \frac{Power_{\ given\ glass\ signal}}{Power_{original\ noise}}$$

$$SNR_{finis} = \frac{Power_{\ filtered\ glass\ signal}}{Power_{residual\ filtered\ noise}}$$

Results

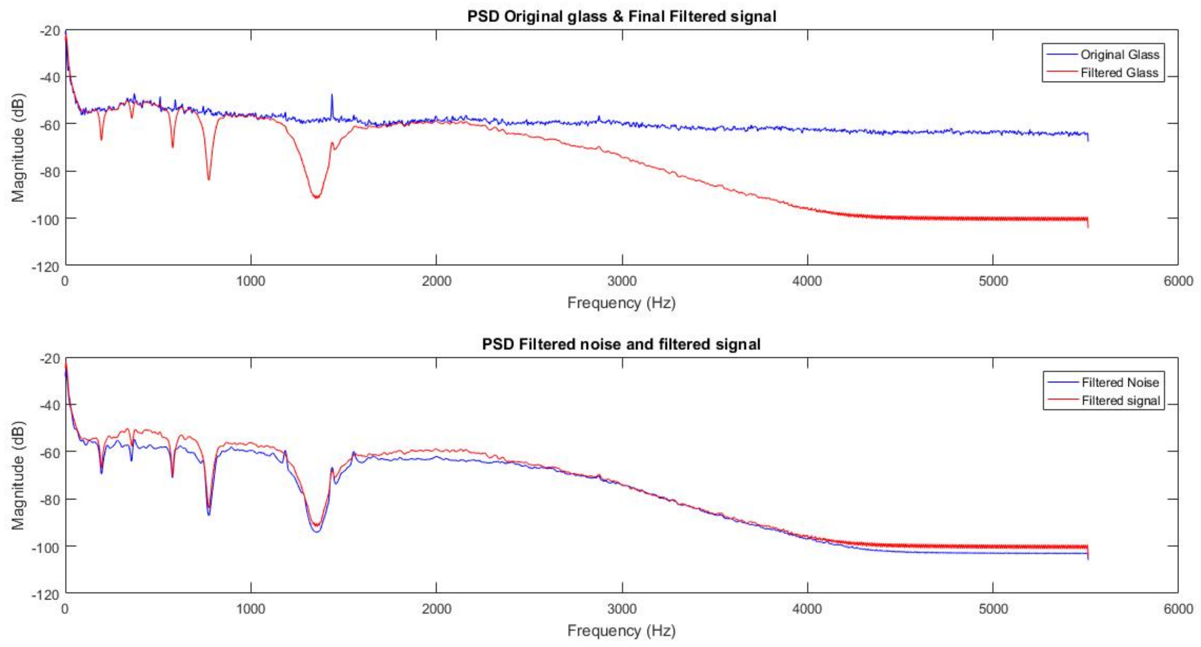


Fig: PSD comparison of filtered signals with original signals

SNR Values

SNR at start = 2.99 dB

SNR after last stage of filtering = 3.34 dB

Comments

From the results we can see that there is an improvement of SNR, compared to the original signals. The PSD comparison also indicates that power content of the noise has been effectively suppressed.

(6) MATLAB Program

For finding the power spectral densities, the **pwelch** function in MATLAB is used. This function takes a large data vector, break it into smaller segments of a specified length, compute as many period-grams as there are segments, and then averages them. As the number of available segments increases, the **pwelch** function will yield a smoother power spectrum (less variance) with power values closer to the expected values.

For the filtering, in-built Butterworth functions have been used.

Code

```
clearvars;
close all;

[g,fg] = audioread('glas-11025.wav'); %Glass signal contaminated with noise
g2=g-mean(g); %Remove DC
fs=fg;
[n,fn] = audioread('noise-11025.wav'); %Original Noise signal
n2=n-mean(n); %Remove DC

%Plot in time-domain
figure;
subplot(2,1,2);
tn=linspace(0,length(n)/fn,length(n)); %time-vector
plot(tn,n);
title('Noise signal in time-domain');

subplot(2,1,1);
tg=linspace(0,length(g)/fg,length(g)); %time-vector
plot(tg,g);
title('Given Glass signal in time-domain');

%Plot in frequency-domain
figure;
Nfft=1024;
f1=linspace(-fn/2,fn/2,Nfft+1); %Frequency-vector
f1 = f1(1:end-1);

G1=fftshift(abs(fft(n,Nfft)));
subplot(2,1,2);
plot(f1(1:end),G1(1:end));
title('Noise signal in freq-domain');

f2=linspace(-fg/2,fg/2,Nfft+1); %Frequency-vector
f2 = f2(1:end-1);

G2=fftshift(abs(fft(g,Nfft)));
subplot(2,1,1);
plot(f2(1:end),G2(1:end));
title('Glass signal in freq-domain');

figure;
G3=fftshift(abs(fft(n2,Nfft)));
subplot(2,1,2);
plot(f1(1:end),G3(1:end));
title('Noise signal in freq-domain (DC component removed)');

G4=fftshift(abs(fft(g2,Nfft)));
subplot(2,1,1);
```

```
plot(f2(1:end),G4(1:end));
title('Glass signal in freq-domain (DC component removed)');

%Proceeding with the DC components removed

% PLOTTING THE SPECTROGRAM
Nfft=1024;
figure;
subplot(2,1,1);
window=hamming(Nfft); %window with a size of N
noverlap=Nfft/2; %no. of points for repeating the window
[Sg,F,T,Pg]=spectrogram(g2,window,noverlap,Nfft,fg,'yaxis');
surf(T,F,10*log10(Pg),'edgecolor','none');
axis tight;
view(0,90);
colormap(hot);
set(gca,'clim',[-80,-30]); %setting limits of axes color
xlabel('Temps (Seconds)');
ylabel('Frequencies Hz');
title ('Spectogram of Glass signal');
colorbar

subplot(2,1,2);
[Sn,F,T,Pn]=spectrogram(n2,window,noverlap,Nfft,fn,'yaxis');
surf(T,F,10*log10(Pn),'edgecolor','none');
axis tight;
view(0,90);
colormap(hot);
set(gca,'clim',[-80,-30]); %setting limits of axes color
xlabel('Temps (Seconds)');
ylabel('Frequencies Hz');
title ('Spectogram of Noise signal');
colorbar

%Power Spectral Density
Nfft2=4096;
window2=hamming(Nfft2); %window with a size of N
noverlap2=Nfft2/2; %no. of points for repeating the window

%Plotting Power spectral densities
figure;
[Pxx_glass,fgpsd] = pwelch(g2,window2,noverlap2,Nfft2,fg);
subplot(2,2,1);
plot(fgpsd,real(10*log10(Pxx_glass)));
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('PSD for Glass signal');

[Pxx_noise,fnpsd] = pwelch(n2,window2,noverlap2,Nfft2,fn);
subplot(2,2,2);
plot(fnpsd,real(10*log10(Pxx_noise)));
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('PSD for Noise signal');

subplot(2,2,3);
plot(fgpsd,real(10*log10(Pxx_glass)));
hold on;
plot(fnpsd,real(10*log10(Pxx_noise)),'r');title('Comparing the PSDs');
legend('PSD for Contaminated Glass','PSD for Noise');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
```

```
pdiff=Pxx_glass-Pxx_noise;
subplot(2,2,4);
plot(fgpsd,real(10*log10(pdiff)));
title ('SNR= PSD of Contaminated Signal-PSD of Noise');
m1=-65;
hline = refline([0 m1]);
hline.Color = 'r';
m2=-50;
hline2 = refline([0 m2]);
hline2.Color = 'r';
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
hold on;

%Filters
%1. LPF 2300Hz
%2. bandstop 180-210Hz
%3. bandstop 350-37Hz
%4. bandstop 560-600Hz
%5. bandstop 725-825Hz
%6. bandstop 1160-1570Hz

%LPF
n=4;
fc=2300;
[b,a]=butter(n,fc/(fg/2)); %Glass
Op1=filter(b,a,g2);
[bn,an]=butter(n,fc/(fn/2)); %Noise
Op1n=filter(bn,an,n2);

%Band-stop 180-210Hz
lc2=180/(fg/2);
uc2=210/(fg/2);
[b2,a2]= butter(2,[lc2 uc2],'stop'); %Glass
Op2 = filter(b2,a2,Op1);
lc2n=180/(fn/2);
uc2n=210/(fn/2);
[b2n,a2n]= butter(2,[lc2n uc2n],'stop'); %Noise
Op2n = filter(b2n,a2n,Op1n);

%Band-stop 350-370
lc3=350/(fg/2);
uc3=370/(fg/2);
[b3,a3]= butter(2,[lc3 uc3],'stop'); %Glass
Op3 = filter(b3,a3,Op2);
lc3n=350/(fn/2);
uc3n=370/(fn/2);
[b3n,a3n]= butter(2,[lc3n uc3n],'stop'); %Noise
Op3n = filter(b3n,a3n,Op2n);

%Band-stop 560-600
lc4=560/(fg/2);
uc4=600/(fg/2);
[b4,a4]= butter(2,[lc4 uc4],'stop'); %Glass
Op4 = filter(b4,a4,Op3);
lc4n=560/(fn/2);
uc4n=600/(fn/2);
[b4n,a4n]= butter(2,[lc4n uc4n],'stop'); %Noise
Op4n = filter(b4n,a4n,Op3n);

%Band-stop 725-825
lc5=725/(fg/2);
uc5=825/(fg/2);
```

```
[b5,a5]= butter(2,[lc5 uc5],'stop'); %Glass
Op5 = filter(b5,a5,Op4);
lc5n=725/(fn/2);
uc5n=825/(fn/2);
[b5n,a5n]= butter(2,[lc5n uc5n],'stop'); %Noise
Op5n = filter(b5n,a5n,Op4n);

%Band-stop 1160-1570Hz
lc6=1160/(fg/2);
uc6=1570/(fg/2);
[b6,a6]= butter(2,[lc6 uc6],'stop'); %Glass
Op6 = filter(b6,a6,Op5);
lc6n=1160/(fn/2);
uc6n=1570/(fn/2);
[b6n,a6n]= butter(2,[lc6n uc6n],'stop'); %Noise
Op6n = filter(b6n,a6n,Op5n);

%Power spectral density of the filtered glass and filtered noise signal
figure;
[Pxx_BS6,Fbs6] = pwelch(Op6,window,noverlap,Nfft2,fs); %Filtered Glass
subplot(2,1,1);
plot(fgpsd,10*log10(Pxx_glass),'b');
title('PSD Original glass & Final Filtered signal');
hold on;
plot(Fbs6,10*log10(Pxx_BS6),'r');
legend('Original Glass', 'Filtered Glass');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');

[Pxx_Bn6,Fbn6] = pwelch(Op6n,window,noverlap,Nfft2,fn); %Filtered Noise
subplot(2,1,2);
plot(Fbn6,10*log10(Pxx_Bn6),'b');
hold on;
plot(Fbs6,10*log10(Pxx_BS6),'r');
title ('PSD Filtered noise and filtered signal');
legend('Filtered Noise', 'Filtered signal');
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');

%SNR calculations
noisy_signal = audioread('glas-11025.wav');
noise = audioread('noise-11025.wav');
snr_start = 10*log10(mean(noisy_signal.^2) / mean(noise.^2))

snr_final = 10*log10(mean(Op6.^2) / mean(Op6n.^2))

%Listen to i/p and o/p -- Please uncomment the next two lines to play the sounds

%soundsc(Op6,fs); %listen to output
%soundsc(g,fs); %listen to input
```

(7) Conclusion

This particular algorithm is not very effective in practice, because it doesn't extract the useful signal very well. This can be easily seen according to the SNR values. One reason could be because of the fact that the noise is present over all frequencies, so in order to block the frequencies for noise, we also have to remove major chunks of the signal, thereby resulting in poor SNR. Improvement in SNR less than one means that this algorithm performs poorly in this particular task. Nevertheless, the theoretical intuition outlined in the frequency-based filtering for the task is done correctly, which is shown by improvement of SNR.