Part 4: A Real World Case Study – Vinyl denoising

Sebastian Grubb (sg3510)

Contents

4.1 Noise Identification

Here we take the periodogram of the file with noise to try and identify it. Figure 1 and Figure 2 were made to assist in the decision.

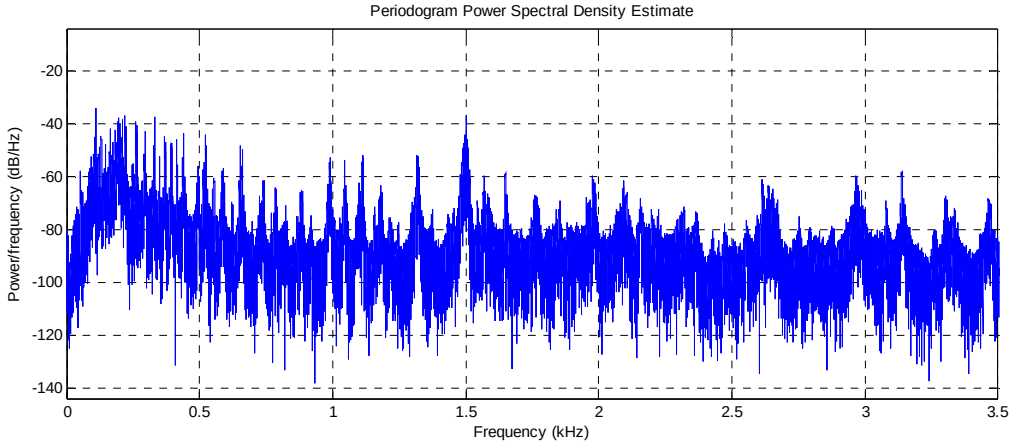

Figure 1 – Periodogram PSD estimate of channel 1 of the noisy sample of Stairway to Heaven

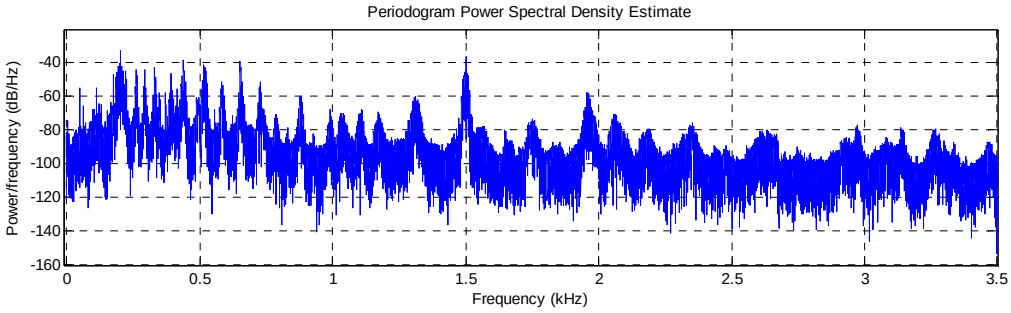

Figure 2 – Periodogram PSD estimate of channel 2 of the noisy sample of Stairway to Heaven

It is possible to notice a very prominent peak at 1.5kHz as well as a smaller one around 200 Hz on channel 2 from the periodograms which would correspond to the intermittent sound heard while playing the track back. However with many other peaks at different frequencies further investigation will have to be done to identify if these frequencies are indeed the noise or not. One method to do so would be to isolate the sections where the noise is heard to sections where no noise is heard and compare the periodograms.

4.2 Identification with clean track

By having the original track the identification of noise if made as a one-to-one comparison can be done.

The first step is to compare, by eye, the differences of the periodograms of the two data sets. From comparing Figure 1 with Figure 3 and Figure 2 with Figure 4 it is possible to compare the two data sets to see where noise is found due to differences in the spectrum.

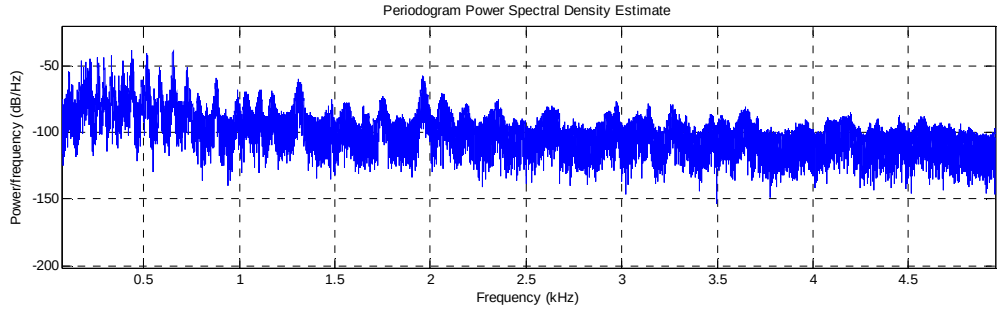

Figure 3 – Periodogram PSD estimate of clean channel 1

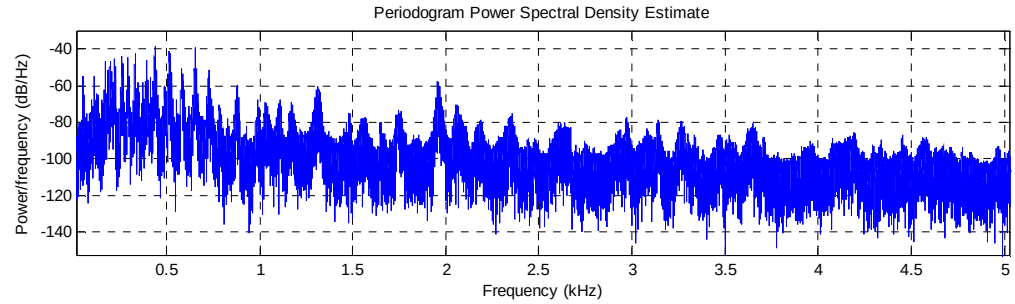

Figure 4 - Periodogram PSD estimate of clean channel 2

Comparing by eye is not the most effective was to extract the noise data, instead two (similar) ways were used to more effectively compare data and find the noise.

The first method consisted of taking the differences between the clean and noisy in frequency domain, giving rise to Figure 5. It allows clear identification of the noise's frequency components as well as the bandwidth the noise occupies.

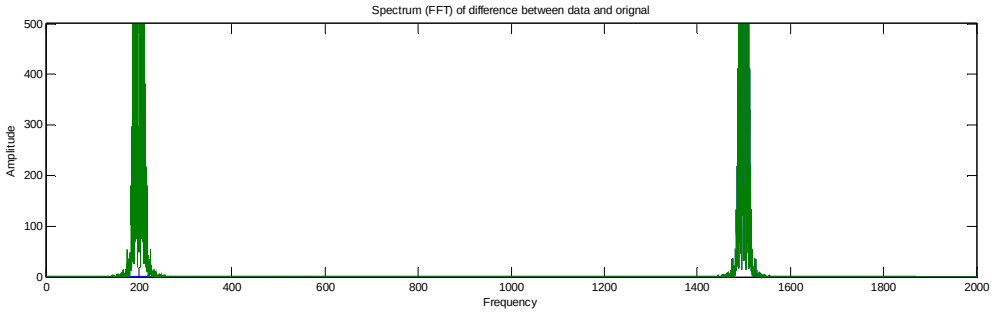

Figure 5 – FFT of the differences between channel 1 (blue) and channel 2 (green) of the clean and noisy signal.

The second method would be to take the differences between clean and noisy in terms of periodogram PSD estimation. In Figure 6 the main noise component in channel 1 can be identified at 1500 Hz and in Figure 7 helps identify the noise spectral components at 200 and 1500 Hz.

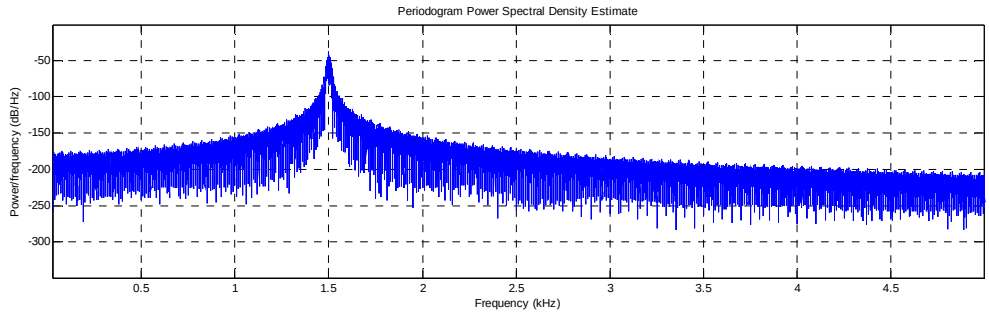

Figure 6 – Periodogram PSD estimate of difference between clean and noisy signal channel 1

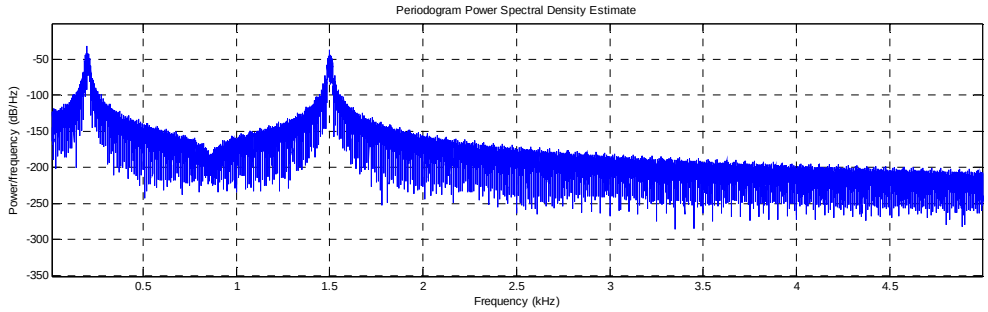

Figure 7 - Periodogram PSD estimate of difference between clean and noisy signal channel 2

4.3 Removing Noise

Once noise has been identified in terms of its spectral components removing it simply consists of constructing the appropriate filters with the intent of removing these frequencies and not harm the rest too much.

Using Butterworth filters designed in matlab the noise from the corrupted track was removed.

This was done by choosing the noise with the appropriate bandwidth to suppress the correct pairs of frequency ranges.

From Figure 5 frequency pairs of 150 to 250Hz for channel 2 and 1420 to 1560 were chosen as they covered the complete spectrum.

```
data=s2h;
[b,a]=butter(3, [1420 1560]/Fs*2, 'stop');
data = filtfilt(b,a,data); %filtfilt used instead of filter to avoid change in phase
[b,a]=butter(3, [150 250]/Fs*2, 'stop');
data(:,2) = filtfilt(b,a,data(:,2));
```

By listening to the output it was determined that such filters were effective and hearing a difference with the clean signal was difficult meaning that the filters had achieved their intent.

4.4 Analysis of Noise removal

Taking the periodograms of the original and the processed data it is possible to appreciate the improvement visually. From Figure 8 it is possible to see that noise is now gone from the

frequencies where it would otherwise be expected but that the rest of the signal is intact, with the same other frequency components - by comparing it to the original clean data.

From Figure 9 it can be seen that the filters are aggressive in terms of spectral removal giving rise to another difference which explains the existence of the peaks at the filtered-out frequencies. However the relative errors of Channel 1 (0.007378) and Channel 2 (0.231638) suggest that noise removal was indeed effective.

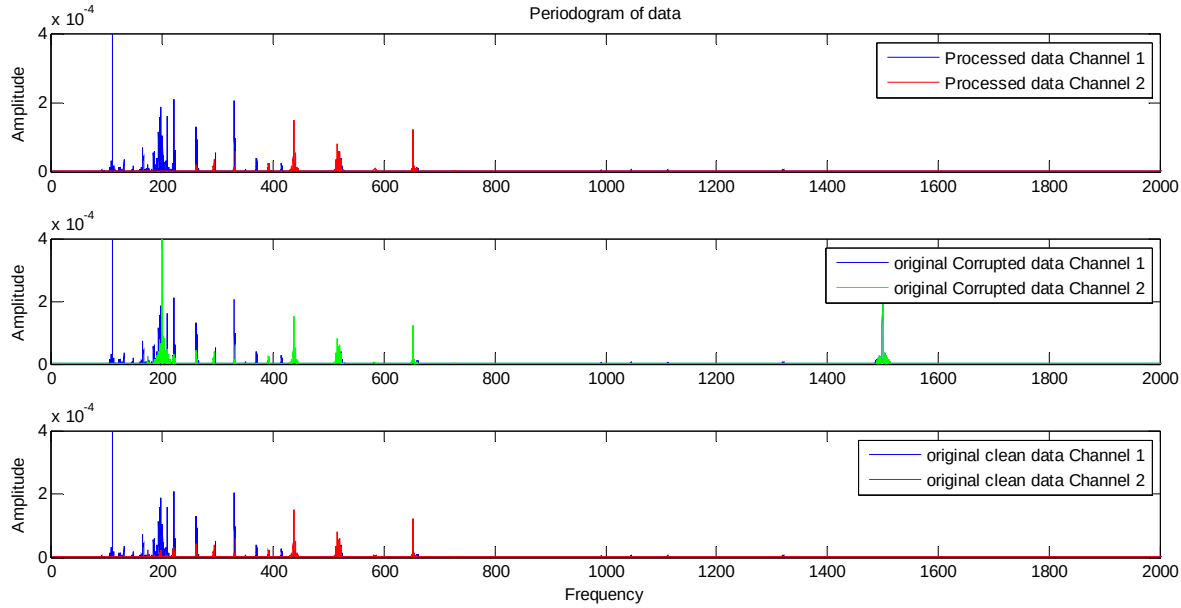

Figure 8 - Periodogram of clean signal, output and input.

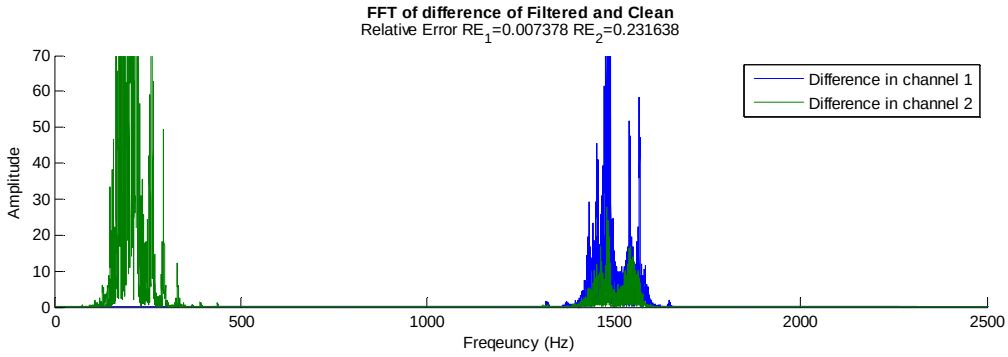

Figure 9 – FFT of difference between filtered signal and clean signal. Relative error is calculated by $\frac{\|P_c - \hat{P_c}\|}{\|P_c\|}$.

4.5 Supervised adaptive learning algorithm

4.5.1 Optimal Coefficients

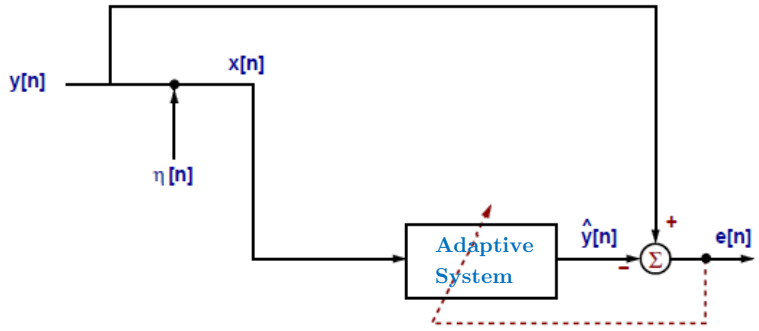

Figure 10 – Diagram of a supervised learning system

Identifying the frequencies of noise and then constructing the relevant filters is not always the best option as this process cannot be done in real. Thus another method – supervised adaptive learning – is used instead.

The idea behind this algorithm is to estimate AR coefficients which would remove noise. For this the Normalized Least Mean Square (NLMS) equations were used:

$$\hat{y}[n] = w[n]^T x[n]$$

$$e[n] = y[n] - \hat{y}[n]$$

$$w[n+1] = w[n] + \mu \times e[n] \times \frac{[x[n], \dots, x[n-p]]}{[x[n], \dots, x[n-p]]^T [x[n], \dots, x[n-p]]}$$

$w[n]$ represents the AR coefficients
$e[n]$ is the error between the estimated signal and the learning signal
$x[n]$ is the input signal
$y[n]$ is the clean signal – also referred to as the learning signal
$\hat{y}[n]$ is the estimated signal

μ is the learning rate – ranging from 0 to 1 for an NLMS filter
p is the order of the filter to filter out the noise.

Figure 10 provides a diagram of the way this adaptive filter works.

The matlab code resembled this:

```
for i = p:N
    ŷ(i) = w(i,:)*x(i:-1:i-p+1); % estimate
    e(i) = y(i) - ŷ(i); % error
    n(i) = x(i:-1:i-p+1)'*x(i:-1:i-p+1); % normalising factor
    if n(i) <= 0 % avoid division by zero!
        w(i+1,:) = w(i,:);
    else
        w(i+1,:) = w(i,:) + μ*e(i)*x(i:-1:i-p+1)'./n(i);
    end
```

```
end
```

Different orders and values of μ were explored over time. First tested was a filter of order 4 with μ=0.5 as seen in Figure 11. The coefficients can be seen to be extremely sensitive and changing very frequently however the result was encouraging due to most noise being removed.

To investigate the effects each of the two parameters was changed, keeping the other one constant, to try and determine optimal parameters. From Figure 11, Figure 12, Figure 13 and Figure 14 and extra collected data, Table 1 was made:

mu\p	p=10		p=8		p=4		p=2		p=1	
mu=1	0.0196	0.0455	0.0192	0.0471	0.0341	0.0481	0.0548	0.0366		
mu=0.5	0.0153	0.0516	0.0163	0.0539	0.0350	0.0501	0.0457	0.0374	0.3877	1.1504
mu=0.1	0.0166	0.086	0.0219	0.0852	0.0534	0.0804	0.0736	0.0806	0.8806	5.554
mu=0.01	0.086	0.1308	0.0488	0.1301	0.0725	0.1302	0.0849	0.1336	0.2398	3.4578

Table 1 – Colour coded table of relative error of channel 1 and channel 2. Green values are lower, red ones are higher. In general it was found that for a $\mu = 0.5$ increasing the p (order) led to less noise and crackling.

From this table it can be seen that having different parameters for difference channels would be preferable. The code was thus changed to reflect this discovery and the following values were used: $p_1 = 10$,$p_2 = 2$, $\mu_1 = 0.5$ and $\mu_2 = 1$. It was found, by listening and looking at the relative error, that increasing the order removed the noise and the crackling noticed at orders around 2 and 4, creating a track with very little crackling. While not computationally desirable a test was done for the values $p_1 = 20$,$p_2 = 40$, $\mu_1 = 0.5$ and $\mu_2 = 1$ and the results can be found in Figure 16. The listening test revealed that most noise and crackling was gone with only a faint intermittent tick.

For computational reasons it would be preferable not to increase the order too much thus it was felt that having $p_1 = 10$,$p_2 = 2$, $\mu_1 = 0.5$ and $\mu_2 = 1$ was the best compromise of quality to computation required.

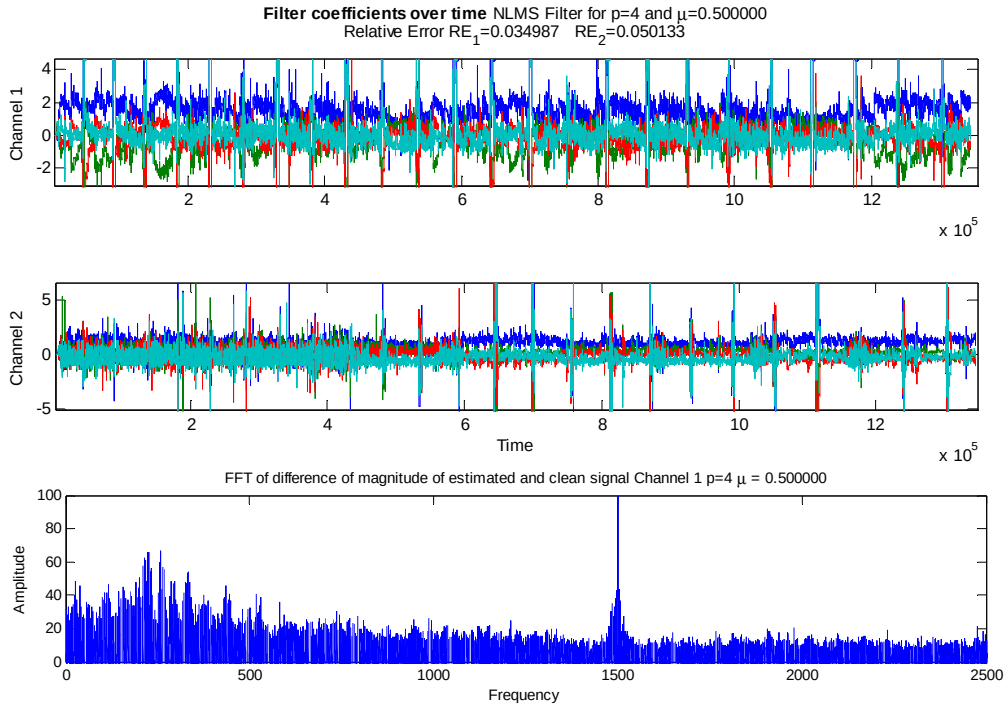

Figure 11 – Evolution of coefficients over time for an order 4 NLMS filter of $\mu = 0.5$ with the associated frequency differences. Listening test: Noise removed by crackling heard instead of noise thus perhaps a bit too aggressive.

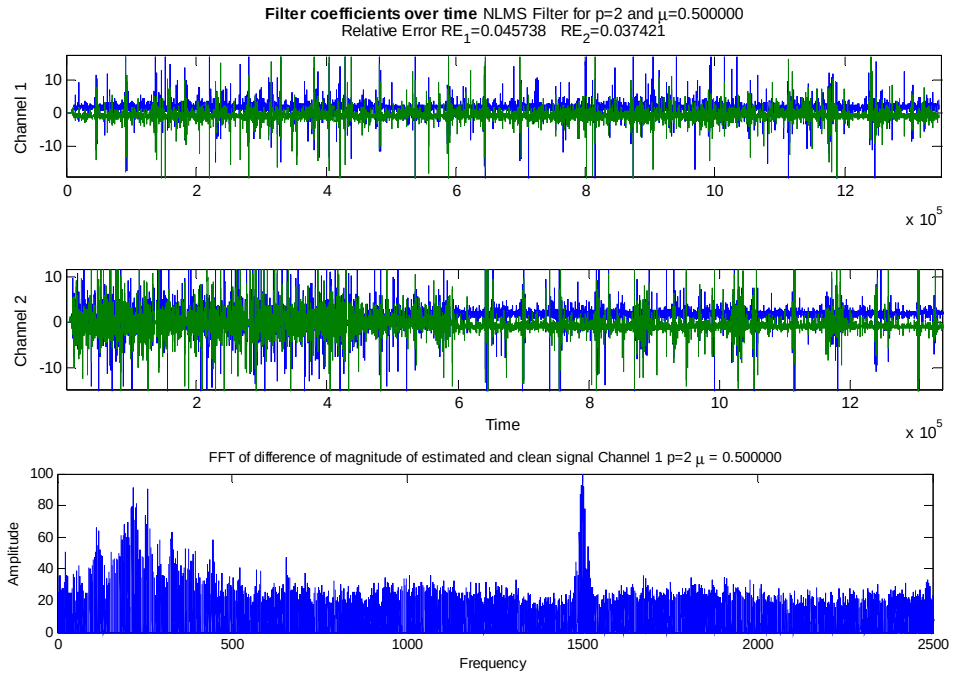

Figure 12 - Evolution of coefficients over time for an order 2 NLMS filter of $\mu = 0.5$ with the associated frequency differences. Listening test: Noise removed by crackling heard instead of noise, not too different from Figure 11 except with possibly more crackling

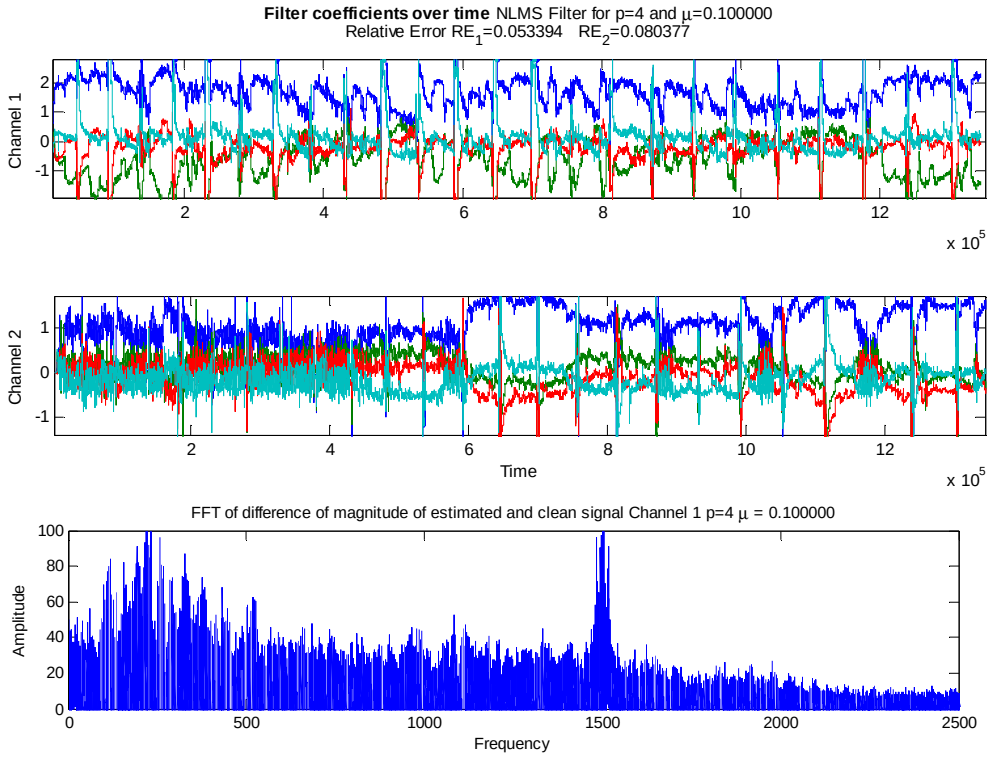

Figure 13 - Evolution of coefficients over time for an order 4 NLMS filter of $\mu = 0.1$ with the associated frequency differences. Listening test: Slight crackling mixed with a soft sound of the original noise. Possibly best listening experience so far due to good compromise between noise removal and crackling,

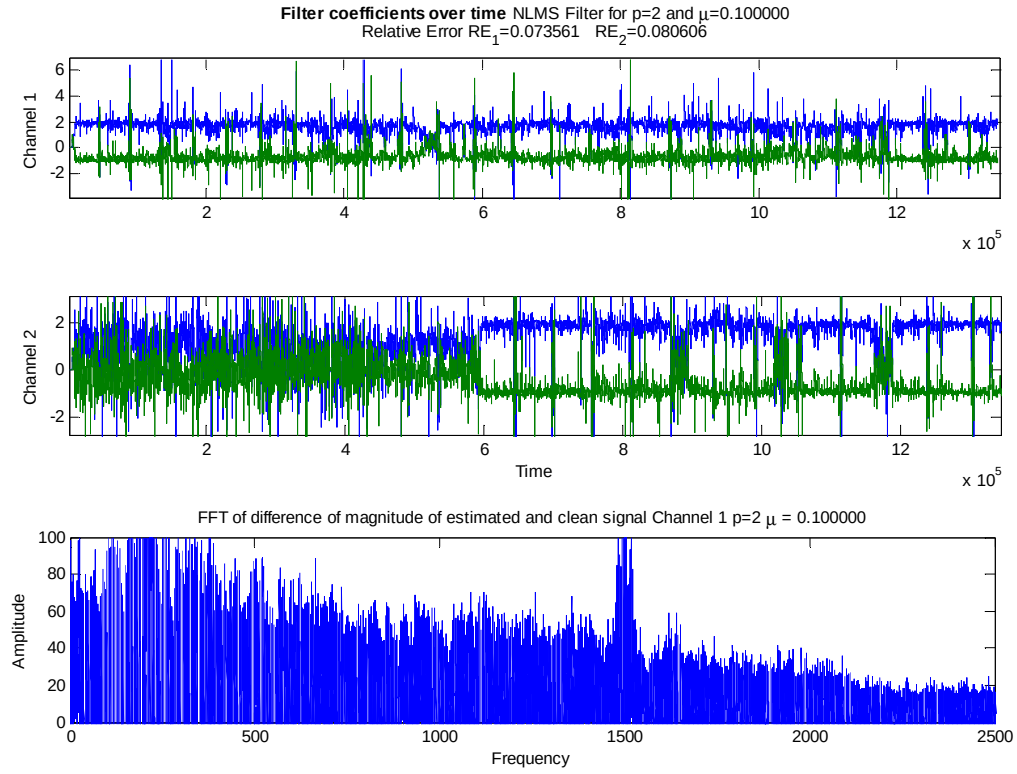

Figure 14 - Evolution of coefficients over time for an order 2 NLMS filter of $\mu = 0.1$ with the associated frequency differences. Listening test: Slight crackling mixed with a soft sound of the original noise. Very similar to the filter with same μ but with order 4.

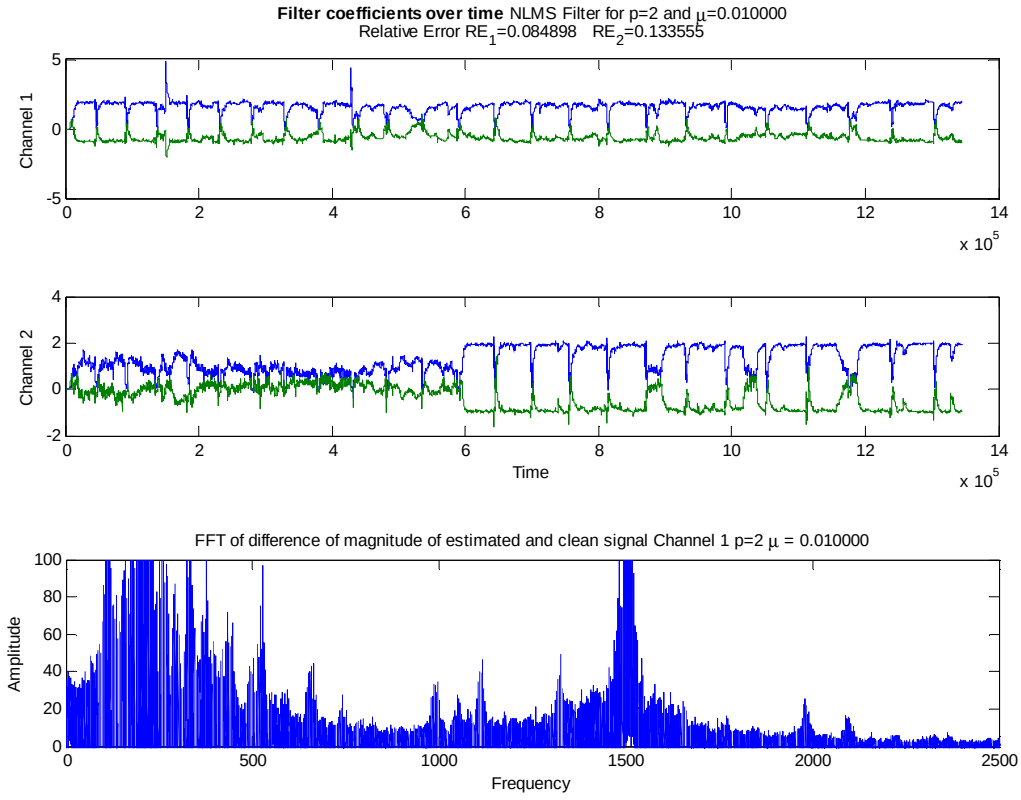

Figure 15 - Evolution of coefficients over time for an order 2 NLMS filter of $\mu = 0.01$ with the associated frequency differences. Listening test: Little to no crackling mixed with a soft sound of the original noise though noise louder than figure 14.

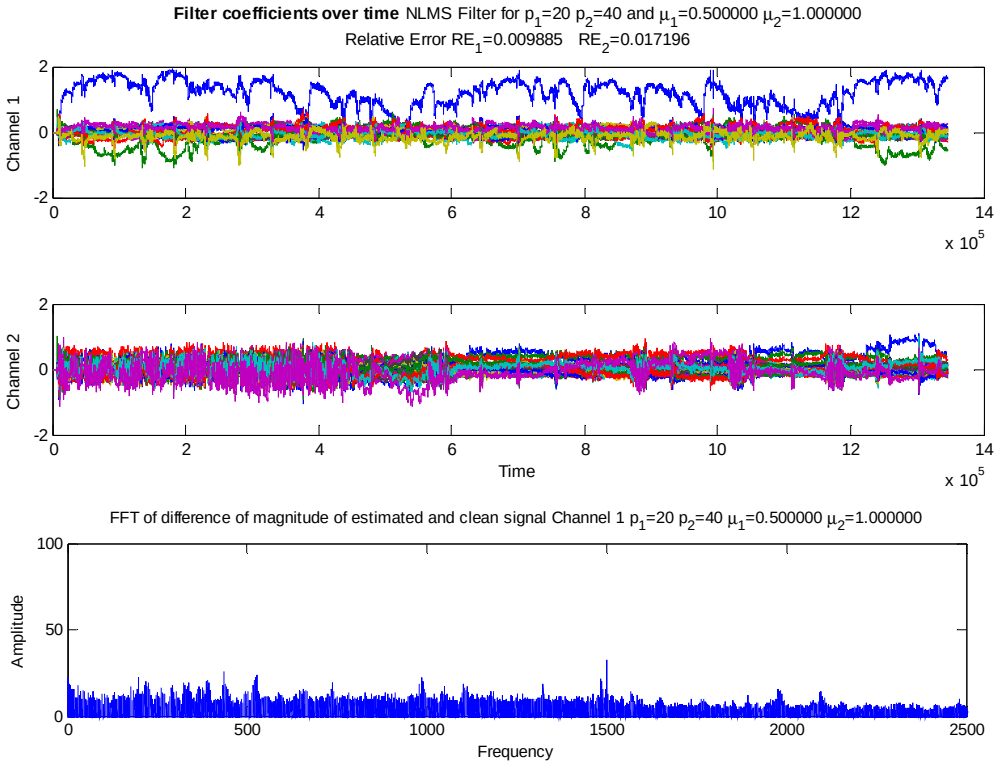

Figure 16 - Evolution of coefficients over time for an order 20 and 40 NLMS filter with the associated frequency differences. It can be seen that increasing the order has led to the FFT of frequency differences to be noticeably lower. Listening test: Very little crackling with no noise heard – only a very faint tick every so often.

4.5.2 Comparing LMS with NLMS

The adaptive filter used what is called a Normalized Least Mean Squared formula – which is designed in order to be stable. This is due to the "basic" LMS filter not being inherently stable and choosing a correct learning rate (μ).

Thus to compare the performance two instances with the same parameters were tested out and the results can be found in Figure 17 and Figure 18. Two measures allow us to see that the NLMS filter is more preferable. Firstly the relative error values of each channel are lower (thus better) for the NLMS filter and secondly the plot of the differences in spectrum of the clean signal and the estimated signal do not show the noise frequency components for NLMS filter whereas a LMS filter does. It is reasonable to conclude that NLMS filters are better for the investigated scenario.

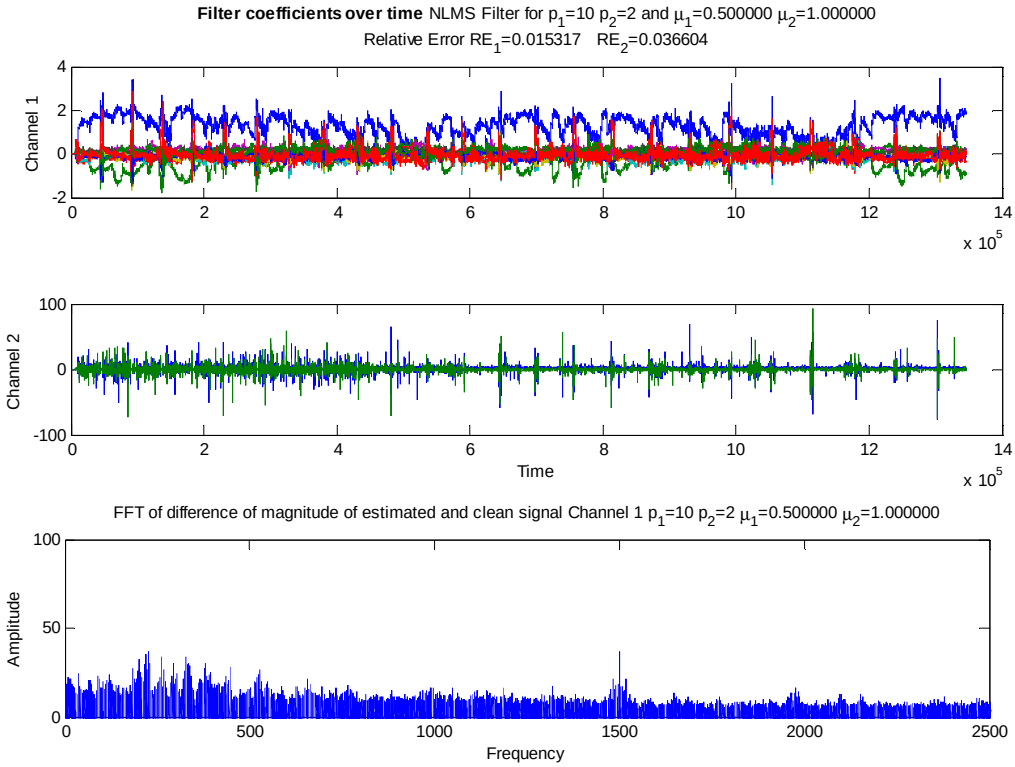

Figure 17 –Evolution of AR coefficients over time for an NLMS filter.

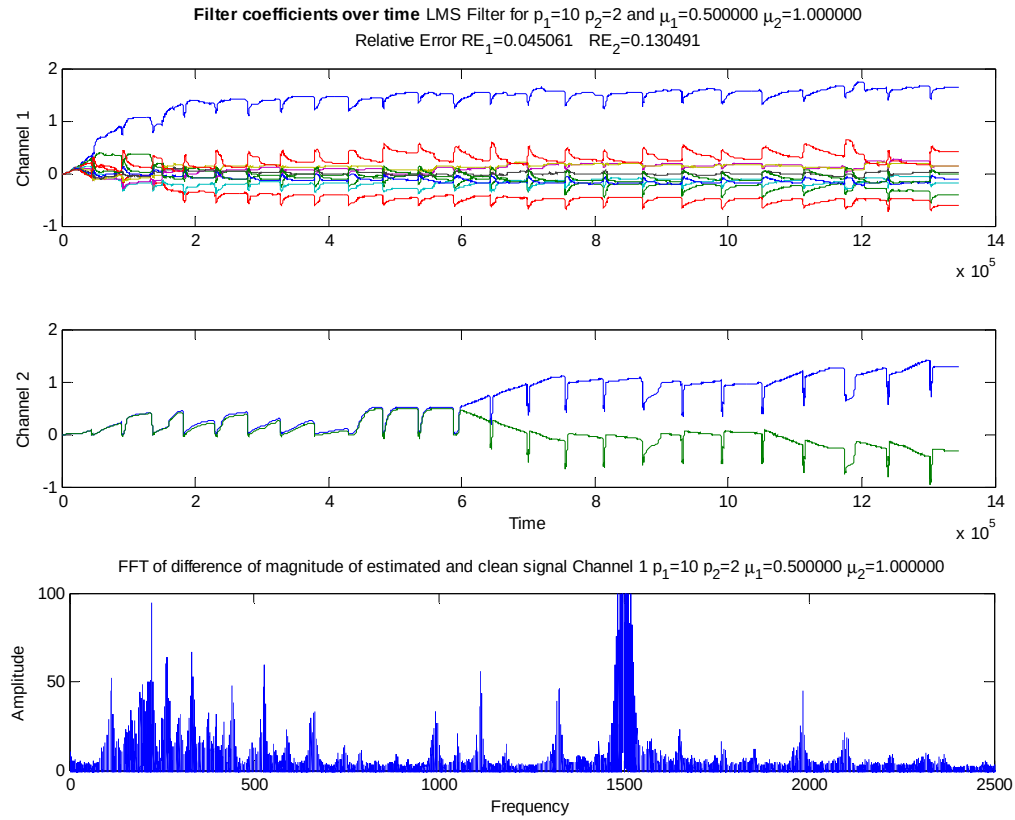

Figure 18 – Evolution of AR coefficients over time for an LMS filter.

4.6 LTE noise removal

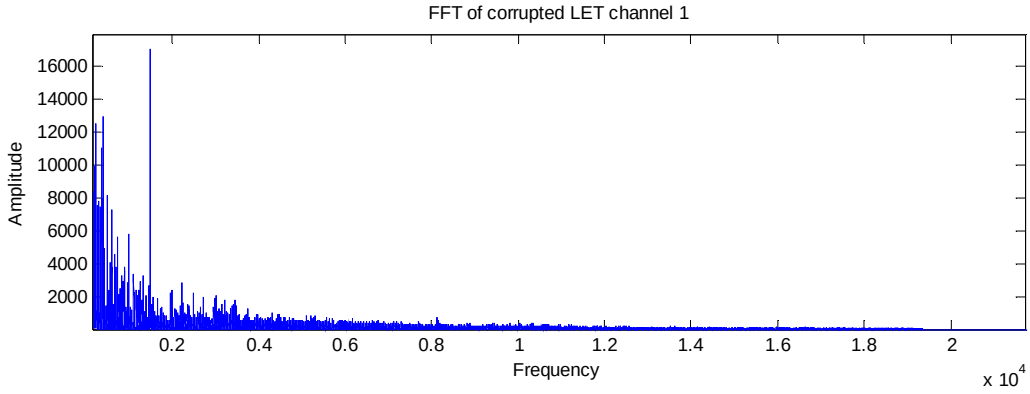

Figure 19 – FFT of the corrupted channel 1 of the LET single

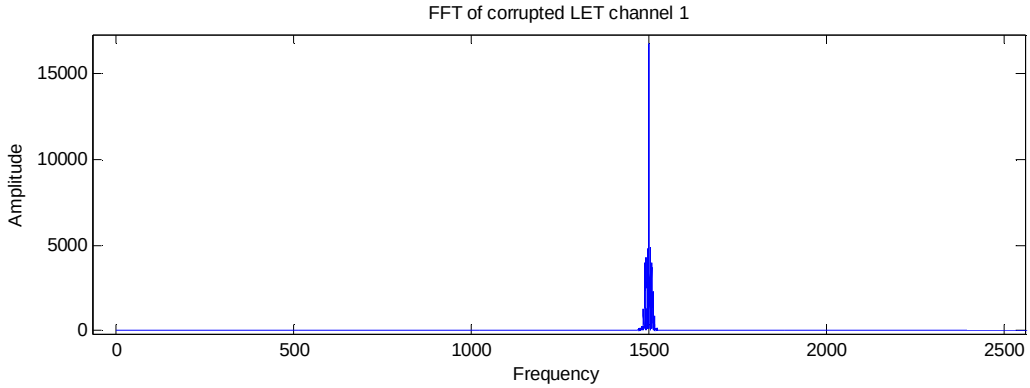

Figure 20 – By taking a difference of the FFT of the clean and corrupted signal noise for channel 1 was identified

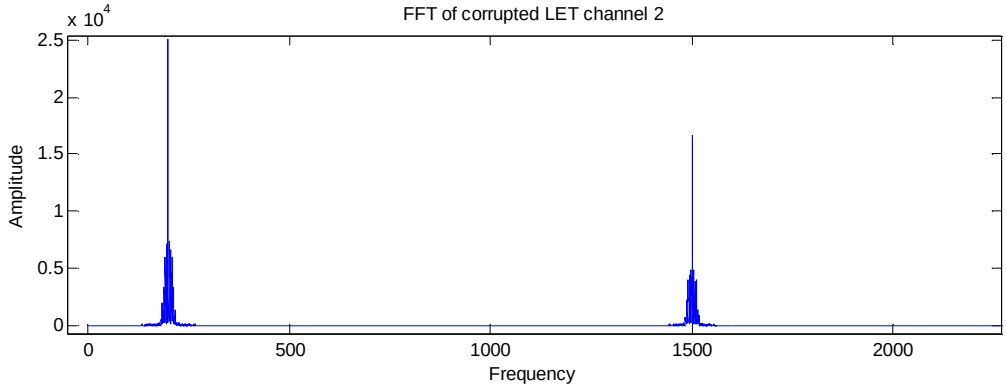

Figure 21 - By taking a difference of the FFT of the clean and corrupted signal noise for channel 2 was identified

The first observation from Figure 19,Figure 20and Figure 21was that the noise was exactly the same as the Stairway to Heaven track. However as the frequencies were more complex it was judged that a simple Butterworth filter would also remove unwanted frequencies as it operates all throughout the track. Instead a NLMS filter was used for its adaptive properties and various mu and p combinations were done.

mu\p	p=10	p=8	p=4	p=2

mu=1	0.0508	0.035	0.0602	0.0349	0.0936	0.0479	0.1287	0.0953
mu=0.5	0.0397	0.0269	0.0464	0.0266	0.0717	0.0383	0.1057	0.0785
mu=0.4	0.0383	0.0256	0.0449	0.0253	0.0696	0.0368	0.105	0.0756
mu=0.1	0.0389	0.022	0.0462	0.0224	0.0767	0.0302	0.1199	0.0624
mu=0.01	0.0581	0.029	0.0693	0.0289	0.0964	0.0318	0.1162	0.0515

Table 2 shows us that this time the optimal mu for both channels is 0.4 and the best tested p is 10. From this these coefficients were chosen and the noise was indeed removed – though some crackling did remain.

Figure 22 shows the evolutions of coefficients over time and justifies the need for a filter of order 10 as the coefficients can be seen to be all distinct. This is due to the LET soundtrack having a wider range of frequencies thus a higher order can lock in to the sound frequencies more accurately and avoid removing essential frequency components.

mu\p	p=10		p=8		p=4		p=2	
mu=1	0.0508	0.035	0.0602	0.0349	0.0936	0.0479	0.1287	0.0953
mu=0.5	0.0397	0.0269	0.0464	0.0266	0.0717	0.0383	0.1057	0.0785
mu=0.4	0.0383	0.0256	0.0449	0.0253	0.0696	0.0368	0.105	0.0756
mu=0.1	0.0389	0.022	0.0462	0.0224	0.0767	0.0302	0.1199	0.0624
mu=0.01	0.0581	0.029	0.0693	0.0289	0.0964	0.0318	0.1162	0.0515

Table 2 - Colour coded table of relative error of channel 1 and channel 2. Green values are lower, red ones are higher. In general it was found that for a $\mu = 0.5$ increasing the p (order) led to less noise and crackling.

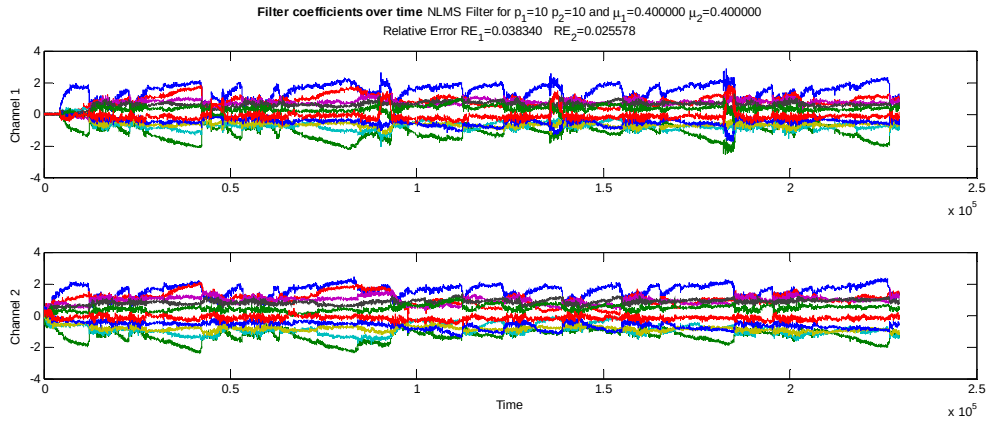

Figure 22 – Evolution of coefficients over time for both channels of the LET album.

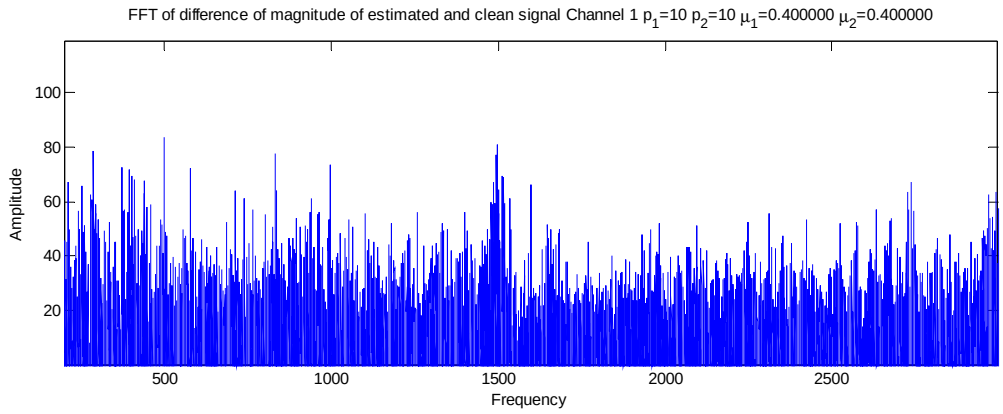

Figure 23 – Difference of the FFT of the estimated signal and the clean signal – as can be seen there still remains some noise at 1500hz but it is greatly attenuated

Appendix

Table of figures

Matlab Code

Part 4.1.1 to 4.1.4

```
clc;
%% Plot periodogram
if ~exist('s2h','var')
    load vinyl.mat
end
Fs=44100;
figure(1);
Hs=spectrum.periodogram;          % Use default values
psd(Hs,s2h_original(:,2),'Fs',Fs)
%% filter
data=s2h;
[b,a]=butter(3, [1420 1560]/Fs*2, 'stop');
data = filtfilt(b,a,data); %filtfilt used instead of filter to avoid change in phase
[b,a]=butter(3, [150 250]/Fs*2, 'stop');
data(:,2) = filtfilt(b,a,data(:,2));
%plot fft of difference
l = length(data);
plot((1:l)*Fs/l,abs(fft(data-s2h_original)))
title('Spectrum (FFT) of difference between data and orignal');
ylabel('Amplitude')
xlabel('Frequency')
axis([0 2000 0 500])
%% play
a = audioplayer(data,Fs);
% play(a);
%% clean periodogram
[periodgm1,w] = periodogram(s2h_original(:,2),[],[],Fs);
figure;
plot(w,periodgm1);
%% 4.1.4 Plot periodograms
figure(4);
% Hs=spectrum.periodogram;        % Use default values
[periodgm1,w] = periodogram(data(:,1),[],[],Fs);
[periodgm2,w] = periodogram(data(:,2),[],[],Fs);
[periodgm3,w] = periodogram(s2h(:,1),[],[],Fs);
[periodgm4,w] = periodogram(s2h(:,2),[],[],Fs);
[periodgm5,w] = periodogram(s2h_original(:,1),[],[],Fs);
[periodgm6,w] = periodogram(s2h_original(:,2),[],[],Fs);

subplot(3,1,1)
plot(w,periodgm1);hold on;
plot(w,periodgm2,'r');hold off;
```

```
title('Periodogram of data')
ylabel('Amplitude')
axis([0 2000 0 4*10^-4]);
legend('Processed data Channel 1','Processed data Channel 2');
axis([0 2000 0 4*10^-4]);

subplot(3,1,2)
plot(w,periodgm3);hold on;
plot(w,periodgm4,'g');hold off;
axis([0 2000 0 4*10^-4]);
legend('original Corrupted data Channel 1','original Corrupted data Channel 2');
axis([0 2000 0 4*10^-4]);
ylabel('Amplitude')

subplot(3,1,3)
plot(w,periodgm5);hold on;
plot(w,periodgm6,'r');hold off;
axis([0 2000 0 4*10^-4]);
legend('original clean data Channel 1','original clean data Channel 2');
axis([0 2000 0 4*10^-4]);
ylabel('Amplitude')
xlabel('Frequency')
%% 4.1.4 Plot data diff
figure(5);
% Hs=spectrum.periodogram;        % Use default values
[periodgm1,~] = periodogram(data(:,1),[],[],Fs);
[periodgm2,~] = periodogram(data(:,2),[],[],Fs);
[periodgm3,~] = periodogram(s2h_original(:,1),[],[],Fs);
[periodgm4,w] = periodogram(s2h_original(:,2),[],[],Fs);
re_1 = abs(norm(periodgm3 - periodgm1)/norm(periodgm3));
re_2 = abs(norm(periodgm4 - periodgm2)/norm(periodgm4));
fftdiff_1 = abs(fft(data(:,1)-s2h_original(:,1)));
fftdiff_2 = abs(fft(data(:,2)-s2h_original(:,2)));
w= (1:length(fftdiff_1))*Fs/length(fftdiff_1);
clf(5);
hold all;
plot(w,fftdiff_1);
plot(w,fftdiff_2);
str = sprintf('\\bf{FFT of difference of Filtered and Clean}\n\\rm{Relative Error RE_1=%f
RE_2=%f}',re_1,re_2);
title(str)
axis([0 2500 0 70]);
ylabel('Amplitude');
xlabel('Freqeuncy (Hz)')
legend('Difference in channel 1','Difference in channel 2');
```

Part 4.1.5

```
%% Adaptation from part 4.1.5 NLMS
clc;
LMS = 1;
p1=10;
p2=2;
mu1=.5;
mu2=1;
x1=s2h(:,1);
x2=s2h(:,2);
N = length(x1);
w1 = zeros(N,p1);
w2 = zeros(N,p2);
e1 = zeros(N,1);
```

```
e2 = zeros(N,1);
y_h1 = zeros(N,1);
y_h2 = zeros(N,1);
y1 = s2h_original(:,1);
y2 = s2h_original(:,2);
n1 = zeros(N,1);
n2 = zeros(N,1);
for i = p1+1:N
    y_h1(i) = w1(i,:)*x1(i-1:-1:i-p1);
    e1(i) = y1(i) - y_h1(i);
    n1(i) = x1(i-1:-1:i-p1)'*x1(i-1:-1:i-p1);
    if LMS == 1
        n1(i) = 1;
    end
    if n1(i) <= 0
        w1(i+1,:) = w1(i,:);
    else
        w1(i+1,:) = w1(i,:) + mu1*e1(i)*x1(i-1:-1:i-p1)'./n1(i);
    end
end
%% bit here
for i = p2+1:N
    y_h2(i) = w2(i,:)*x2(i-1:-1:i-p2);
    e2(i) = y2(i) - y_h2(i);
    n2(i) = x2(i-1:-1:i-p2)'*x2(i-1:-1:i-p2);
    if LMS == 1
        n2(i) = 1;
    end
    if n2(i) <= 0
        w2(i+1,:) = w2(i,:);
    else
        w2(i+1,:) = w2(i,:) + mu2*e2(i)*x2(i-1:-1:i-p2)'./n2(i);
    end
end
disp('Done! :)')
%% fig

%calculate relative error
[periodgm1,~] = periodogram(y_h1,[],[],Fs);
[periodgm2,~] = periodogram(y_h2,[],[],Fs);
[periodgm3,~] = periodogram(s2h_original(:,1),[],[],Fs);
[periodgm4,w] = periodogram(s2h_original(:,2),[],[],Fs);
re_1 = abs(norm(periodgm3 - periodgm1)/norm(periodgm3))
re_2 = abs(norm(periodgm4 - periodgm2)/norm(periodgm4))
%%%%%%%%%%%%%%%%%%%%
figure(1);
subplot(2,1,1);
plot(w1);
str = sprintf('\\bf{Filter coefficients over time} \\rm{NLMS Filter for p_1=%d p_2=%d and
\\mu_1=%f \\mu_2=%f}\nRelative Error RE_1=%f   RE_2=%f',p1,p2,mu1,mu2,re_1,re_2);
title(str);
ylabel('Channel 1')
subplot(2,1,2);
plot(w2);
ylabel('Channel 2')
xlabel('Time')
figure(2)
plot([1:length(y_h1)]*FS/length(y_h1),abs(fft(y_h1))-abs(fft(s2h_original(:,1))))
str = sprintf('FFT of difference of magnitude of estimated and clean signal Channel 1 p_1=%d p_2=%d
\\mu_1=%f \\mu_2=%f',p1,p2,mu1,mu2);
```

```
title(str)
ylabel('Amplitude')
xlabel('Frequency')
axis([0 2500 0 100])
%% 4.1.5 Plot data diff
figure(3);
Fs = 44100;
% Hs=spectrum.periodogram;       % Use default values
[periodgm1,~] = periodogram(y_h1,[],[],Fs);
[periodgm2,~] = periodogram(y_h2,[],[],Fs);
[periodgm3,~] = periodogram(s2h_original(:,1),[],[],Fs);
[periodgm4,w] = periodogram(s2h_original(:,2),[],[],Fs);
periodgm1 = abs((periodgm3-periodgm1))./abs(periodgm3);
periodgm2 = abs((periodgm4-periodgm2))./abs(periodgm4);
abs(periodgm3);
clf(3);
hold all;
plot(w,periodgm1);
str = sprintf('Periodogram of normalized difference of magnitude of estimated and clean signal p_1=%d
p_2=%d \\mu_1=%f \\mu_2=%f',p1,p2,mu1,mu2);
title(str)
ylabel('Amplitude')
plot(w,periodgm2);
legend('(P1-Ph1)/P1','(P2-Ph2)/P2');
axis tight;
ylabel('Amplitude')
xlabel('Frequency')
%% play the sample
a =audioplayer([y_h1 y_h2],FS);
play(a)
```

Part 4.6

```
%% Adaptation from part 4.1.5 NLMS
clc;
LMS = 0;
p1=2;
p2=p1;
mu1=.4;
mu2=mu1;
x1=um(1:end/10,1);
x2=um_original(1:end/10,2);
N = length(x1);
w1 = zeros(N,p1);
w2 = zeros(N,p2);
e1 = zeros(N,1);
e2 = zeros(N,1);
y_h1 = zeros(N,1);
y_h2 = zeros(N,1);
y1 = um_original(:,1);
y2 = um_original(:,2);
n1 = zeros(N,1);
n2 = zeros(N,1);
for i = p1+1:N
    y_h1(i) = w1(i,:)*x1(i-1:-1:i-p1);
    e1(i) = y1(i) - y_h1(i);
    n1(i) = x1(i-1:-1:i-p1)'*x1(i-1:-1:i-p1);
    if LMS == 1
        n1(i) = 1;
    end
    if n1(i) <= 0
        w1(i+1,:) = w1(i,:);
```

```
    else
        w1(i+1,:) = w1(i,:) + mu1*e1(i)*x1(i-1:-1:i-p1)'./n1(i);
    end
end
%% bit here
for i = p2+1:N
    y_h2(i) = w2(i,:)*x2(i-1:-1:i-p2);
    e2(i) = y2(i) - y_h2(i);
    n2(i) = x2(i-1:-1:i-p2)'*x2(i-1:-1:i-p2);
    if LMS == 1
        n2(i) = 1;
    end
    if n2(i) <= 0
        w2(i+1,:) = w2(i,:);
    else
        w2(i+1,:) = w2(i,:) + mu2*e2(i)*x2(i-1:-1:i-p2)'./n2(i);
    end
end
disp('Done! :)')
%% fig
Fs = 44100;
%calculate relative error
[periodgm1,~] = periodogram(y_h1,[],[],Fs);
[periodgm2,~] = periodogram(y_h2,[],[],Fs);
[periodgm3,~] = periodogram(um_original(1:N,1),[],[],Fs);
[periodgm4,w] = periodogram(um_original(1:N,2),[],[],Fs);
re_1 = abs(norm(periodgm3 - periodgm1)/norm(periodgm3))
re_2 = abs(norm(periodgm4 - periodgm2)/norm(periodgm4))
%%%%%%%%%%%%%%%%%%%
figure(1);
subplot(2,1,1);
plot(w1);
str = sprintf('\\bf{Filter coefficients over time} \\rm{NLMS Filter for
p_1=%d p_2=%d and \\mu_1=%f \\mu_2=%f}\nRelative Error RE_1=%f
RE_2=%f',p1,p2,mu1,mu2,re_1,re_2);
title(str);
ylabel('Channel 1')
subplot(2,1,2);
plot(w2);
ylabel('Channel 2')
xlabel('Time')
figure(2)
plot([1:length(y_h1)]*FS/length(y_h1),abs(fft(y_h1))-
abs(fft(um_original(1:N,1))))
str = sprintf('FFT of difference of magnitude of estimated and clean signal
Channel 1 p_1=%d p_2=%d \\mu_1=%f \\mu_2=%f',p1,p2,mu1,mu2);
title(str)
ylabel('Amplitude')
xlabel('Frequency')
axis tight;
%% 4.1.5 Plot data diff
figure(3);
Fs = 44100;
% Hs=spectrum.periodogram;       % Use default values
[periodgm1,~] = periodogram(y_h1,[],[],Fs);
[periodgm2,~] = periodogram(y_h2,[],[],Fs);
[periodgm3,~] = periodogram(um_original(1:N,1),[],[],Fs);
[periodgm4,w] = periodogram(um_original(1:N,2),[],[],Fs);
periodgm1 = abs((periodgm3-periodgm1))./abs(periodgm3);
periodgm2 = abs((periodgm4-periodgm2))./abs(periodgm4);
abs(periodgm3);
clf(3);
```

```
hold all;
plot(w,periodgm1);
str = sprintf('Periodogram of normalized difference of magnitude of
estimated and clean signal p_1=%d p_2=%d \\mu_1=%f
\\mu_2=%f',p1,p2,mu1,mu2);
title(str)
ylabel('Amplitude')
plot(w,periodgm2);
legend('(P1-Ph1)/P1','(P2-Ph2)/P2');
axis tight;
ylabel('Amplitude')
xlabel('Frequency')
%% play the sample
a =audioplayer([y_h1 y_h2],FS);
play(a)
```