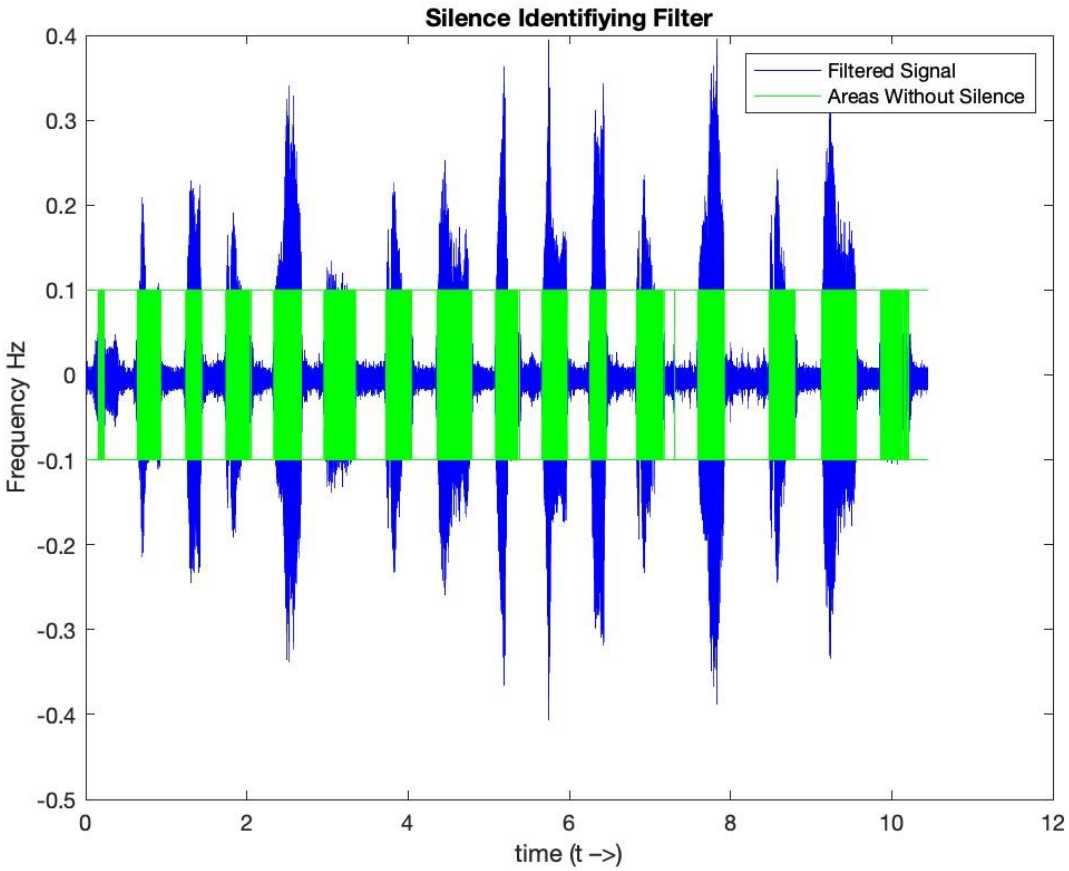
252 Linear Signals and Systems Group Project



Names: Madhav Anugopal, Simha Kalimipalli

Group: 17

Date: 24th Nov. 2022

Course Number & Name: SYDE 252 Signals

Instructor's Name: Charbel Azzi

Abstract

This section should begin on a new page. The abstract is a one or two paragraph summary of the work. It stands alone with no reference to figures, charts, or tables in the text. The line spacing default is double-spacing for academic reports. Other manuscripts may require different line spacing options. The abstract should not exceed one page of double-spaced text.

This project was conducted for the linear systems and signals class. In this project the MATLAB programming language was used to conduct various experiments on three signals. This includes uploading the signals, graphing them as well as reducing noise in the signals. There were multiple methods that were used to reduce the noise in the signals including the weighted average filter, the median filter as well as the moving average filter.

Our research was able to use the MATLAB language to accomplish the previous objectives. The researched was also able to answer three additional research questions that were in the project scope. This was done using the knowledge learned in the previous sections. This project shows that it is indeed possible to use MATLAB to conduct signals and signals operations with high success. This report advocates the use of MATLAB to help students gain knowledge of real word applications of signal processing.

Table of Contents

List of Figures ***4***

List of Tables ***5***

1) Introduction ***6***

2) Background ***8***

3) Methodology ***9***

Part 1: 9

Part 2: 9

Part 3: 12

3.1) Moving Average Filter Design *14*

4) Results and discussion ***16***

Part 1: 16

Part 2: 16

Part 3: 23

5) Conclusions and recommendations ***25***

7) Bibliography ***27***

Appendix: ***28***

Appendix 1: Part 1 Code *28*

Appendix 2: Part 2 Code, Birds *29*

Appendix 3: Part 2 Code, Drums *31*

Appendix 4: Part 2 Code, Speech *33*

Appendix 5: Part 3 Codes *35*

List of Figures

Figure 1 Diagram of Part 1 Decision Making 9
Figure 2 Moving Average Filter Example 14
Figure 3 Part 1 Bird Sound 16
Figure 4 Part 1 Drums Sound 16
Figure 5 Part 1 Speech Sound 16
Figure 6 Slightly Distorted Bird Mean Filtering 17
Figure 7 Extremely Distorted Bird Mean Filtering 17
Figure 8 Ideal Bird Mean Filtering 17
Figure 9 Slightly Distorted Bird Weighted Average Filtering 17
Figure 10 Extremely Distorted Bird Weighted Average Filtering 18
Figure 11 Ideal Bird Weighted Average Filtering 18
Figure 12 Slightly Distorted Bird Median Average Filtering 18
Figure 13 Extremely Distorted Bird Median Average Filtering 18
Figure 14 Ideal Bird Median Average Filter 18
Figure 15 Extremely Distorted Drum Mean Filtering 19
Figure 16 Slightly Distorted Drum Mean Filtering 19
Figure 17 Ideal Drum Mean Filtering 19
Figure 18 Extremely Distorted Drum Weighted Average Filtering 19
Figure 19 Slightly Distorted Drum Weighted Average Filtering 20
Figure 20 Ideal Drum Weighted Average Filtering 20
Figure 21 Extremely Distorted Drum Median Average Filtering 20
Figure 22 Slightly Distorted Drum Median Average Filtering 20
Figure 23 Ideal Drum Median Average Filtering 20
Figure 24 Extremely Distorted Speech Mean Average Filtering 21
Figure 25 Slightly Distorted Speech Mean Average Filtering 21
Figure 26 Ideal Speech Mean Average Filtering 21
Figure 27 Extremely Distorted Speech Weighted Average Filtering 21
Figure 28 Slightly Distorted Speech Weighted Average Filtering 22
Figure 29 Ideal Speech Weighted Average Filtering 22
Figure 30 Extremely Distorted Speech Median Average Filtering 22
Figure 31 Slightly Distorted Speech Median Average Filtering 22
Figure 32 Ideal Speech Median Average Filtering 22
Figure 33 Part 3 Syllables 23
Figure 34 Part 3 BPM 23
Figure 35 Silence Identifying Graph for Bird Sound File 24

List of Tables

Table 1 Table of Window Size for Bird Sound File 17
Table 2 Table of Window Size for Drum Sound File 19
Table 3 Table of Window Size for Speech Sound File 21

1) Introduction

This is the 252 Linear Signals and Systems group project. In this project, this group performed investigations involving certain signals i.e. that of a Bird, a Drum and a human voice. We applied the knowledge that we acquired in this course such as that of filters and the time and frequency domain.

There were three main objectives of this project. The first main objective was to read and correctly sample the three main audio files. The second main objective was to create low-pass filters to filter through the three main sounds. The examples of low-pass filters that were used include a moving average filter, a weighted average filter as well as a median filter. Detailed explanations of this terminology will be found in the background section. Additionally ideal window size and filter were observed from the above filters. In the second objective, there are additional questions to be answered using experience from the filters including describing a moving average filter and high pass filter.

The filter in window size was adjusted and closely monitored to give the best results for the three low-pass filters. The third main objective was to use our cleaned audio files to perform some analysis of the sound patterns. The three tasks were as follows: 1) Find the number of syllables in the human voice audio. 2) Find the number of beats per minute in the Drum sample. 3)The third objective was to find the silent regions in the Bird audio file.

For this project, the Matlab language was utilized to conduct the experiments. More specifically a combination of online Matlab as well as Matlab for Mac computers was used. Both group members contributed to the group via the two mediums and the outputs of the research were combined into singular outputs. There were several research resources used for this project including Matlab's extensive documentation found on the MathWorks website.

The main criterion for the output of this project includes graphs of the input signals that have been uploaded to Matlab and the resample outputs for the first objective. For the second objective, plots of the three filtered signals using the three different filters will be included. A final explanation of the ideal window size and filter shall be given along with an answer to two non–analytical questions in the project guidelines. For the third objective, the plot of the signals that were used to help conclude as well as the conclusion itself will be presented. The full codebase that was used to achieve the above three objectives will also be presented.

Additionally, important terminology used in the project will be explained in the background section.

2) Background

This is a brief background of some of the terminology and concepts used in this project. Matlab will be the main programming language as well as the development environment for this project. It is a programming language developed by the American software company MathWorks. Matlab began as a series of programs developed from the late 1960s onwards to help conduct linear algebra analysis. The first commercial PC version of Matlab only arose in 1985 following the introduction of the IBM PC. MATLAB has expanded its scope and has added functions and applications of the software. These included the signals and systems applications which will be used in this project.

The main process that will be used in this project is low-pass filters. Low-pass filter input signals are designed to let signals pass when they are below is below a certain threshold of frequency and prevent signals from passing when they are above this threshold of frequency. In practice, these are used to reduce noise in the audio samples as the noise generally has an abnormally high frequency. Applying the Low pass filter helps improve the audio clarity of a noise. In this project, three main types of lowpass filters will be used and a brief description of the three filters will be given below.:

A mean or moving average filter takes the value of the signal at a particular point in time. It then takes a buffer a certain amount of time before that point of time. It then takes the average value of a signal over the duration of that time buffer and replaces the signal value at that point in time with that average value. This helps to reduce the outliers and extreme values in the signal helping to reduce the noise and increase clarity as well

A weighted average filter gives importance to some samples of a signal at the expense of the importance of other samples. Thus, the weighted average of the signal is produced. Generally, the values of a signal that are closest to the value of a signal at a time observed are given the highest importance and those that are furthest away in time from the time spots are given the least amount of importance.

A median filter is similar to a mean filter in that a time buffer is taken before a certain observed point of time and the central tendency of that time buffer is taken to be the same to replace the value of the signal in that time. However, the main difference is that the method chosen to evaluate the central tendency is the median function.

3) Methodology

Part 1:

The signals are assumed to be continuous time signals. The signals are then converted from stereo to mono. Then the sampling rate of the signal is changed to meet 16 kHz. This can be seen in the signal processing diagram below.

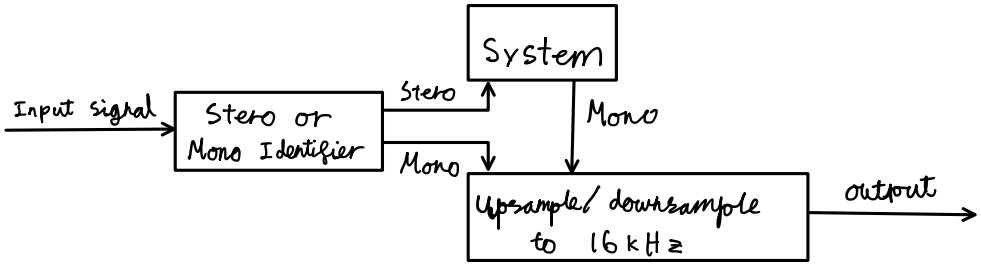


Figure 1 Diagram of Part 1 Decision Making

Part 2:

The signals were assumed to be continuous time signals in the input section. For the mean filtering, the summation of the current input and the previous inputs in the window range is taken and then divided by the window size for the new output at this point. This changes the noise in the signal as the window size changes. The equation for a mean filter is $y[n] = \frac{1}{L}\sum_{k=0}^{L-1} x[n-k],$ while the equation used to calculate it in the program is $y(n) = \frac{1}{windowSize}(x(n) + x(n-1) + \ldots + x(n-(windowSize-1)))$. The equations are fundamentally identical, and this allows the filter to be accurately implemented into the testing section of the report. The ideal window size varies between the 3 different audio inputs and so do the slightly distorted and extremely distorted values because of the varying frequency in all three signals.

For the Bird sound file, the ideal window size was determined to be unit 3 as it had the best filtering of the sound in both auditory and graphical outputs. This was done by testing the two extremes of distortion between slightly and extremely distorted. The value used for slightly distorted was a window size of 2 while the extremely distorted size is a window size of 500. The reason for going all the way up

to 500 was because the sizes below it didn't filter out the sound graphically due to the nature of the sound's frequencies.

For the Drum sound file, the ideal window size after a lot of trial and error was determined to be 3/2. The smaller window size was determined due to the lower frequency of high peaks in the sound file and the softer frequency tones in between the peaks. The extremely distorted window size is not as large as the Bird sound file with a value of 25. This once again supports the idea of a lack of high amplitude peaks results in an easier window size to clean up the audio file.

For the Speech sound file, the ideal window size was determined to be 6. This was determined after figuring out that the minimum window size needed for slight distortion is 3 and for extreme distortion is 100. The large window size is required for extreme distortion as the Speech sound file has a lot of varying amplitudes next to each other like the Birds sound file. This results in extreme noise filtering becoming difficult in low window sizes.

For the Weighted Average filter, some values are given a larger weight than others. This is calculated using the equation $y[n] = \sum_{k=0}^{L-1} b_k x[n-k],$ while it is ideal to mathematically interpret, it is a bit harder to recreate this algorithmically for the software simulate. Therefore, a gaussian distribution filter was used as a weighted average filter. The gaussian filter, $y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ $\mu =$ Mean $\sigma =$ Standard Deviation, allows one to filter out the noise while ensuring that the more prominent values are given the weight required to result in a more comprehensive filtering.

The application of a weighted filter in the Bird sound file was implemented with an ideal window size of 5. This is bigger than the window size used for the mean filter as the weighted filter requires a larger window to ensure there is enough data to create a more comprehensive system. As not all values

that are inputted are weighed equally like in the mean filter algorithm. The slightly distorted window size is still 2 while the extremely distorted filter has a window size of 50. This is smaller as with the more frequent changes in the amplitude the weighted average takes a smaller window size in comparison to the mean filter to completely filter out the noise to the extreme.

In the Drum sound file, an ideal gaussian filter has a window size of 10. This was selected with trial and error as the slightly distorted filter has a window size of 2 while the extremely distorted filter has a window size of 25. The window size of 10 is close to the median of the two extremes and it is the perfect size to get the best filtering without having the sound file extremely distorted.

The Speech file has an ideal filter window size of 15. This took a lot of experimenting to get right as many silent areas in the sound file affected the results of the filter. This caused the slightly distorted window size to be set to 7 while the extremely distorted window size was set to the number 1,000. Even with such a high number the sound is not completely filtered out as the nature of the filter and the silent areas in the sound file prevent it from occurring.

The median average filter is slightly different as this filter considers past and future events alongside the current event to find the median of the signals present. It does this by using the following formula, $y[n] = median(x[n], x[n-1], x[n-2], ..., x[n-k]), \text{for k} = 0, 1, 2, ..., \text{L}$. This was implemented with an adjustable window size represented by the letter L.

All three of the sound files have a similar filtering window for the median filter with an ideal filter window of 5. A slightly distorted filter window of 3 and an extremely distorted filter window of 7. This is a result of the more adaptive filtering a median filter has as it takes the middle value of the max and min in the window.

Part 3:

The Speech file is assumed to have 10 syllables. To calculate after careful listening, it was determined that the high-frequency peaks represent the syllables. To accurately count the number of peaks, the noise must be first filtered out so that only the highest 10 peaks showing the syllables are still visible. After trial and error, the Moving Average filter was chosen to filter out the noise and the window size of 2.23 was selected after a systematic process of testing using a visual representation of the filtered output. Then the peaks are counted by using an algorithm that does two specific things. The first role is to identify if the data point is above the threshold frequency. This acts essentially as a custom high pass filter, which ensures that the unfiltered and background noise in between is ignored by the second part of the algorithm. The second part identifies if the peak is local maxima by checking to see if values on both sides of the current peak are decreasing. If so, the algorithm adds 1 to the counter and then keeps parsing through the signal until the entirety is parsed through.

To identify the beats per minute of the clip a similar approach was also implemented. After repeated listening, a small low-frequency peak was identified as the metronome sound in the background of the clip. However, the clip is too noisy in general to be able to accurately find this low-frequency peak with an algorithm. Therefore, after reviewing the three filters used for Drums in part 2, the gaussian filter was selected to remove the noise while keeping the shape and relative amplitude of the peaks intact. The window size chosen was small with the number 5. As the metronome is not the highest peak in the filtered signal, it is filtered once more through a low pass filter, which ensures that the new highest peaks are the metronome sounds in the signal.

Then the peaks are counted by using an algorithm that does two specific things. The first role is to identify if the data point is above the threshold frequency. This acts essentially as a custom high pass filter, which ensures that the unfiltered and background noise in between is ignored by the second part of the algorithm. Which is identical to the algorithm used for the Speech filter. The second part identifies if the peak is local maxima by checking to see if values on both sides of the current peak are decreasing. If

so, the algorithm adds 1 to the counter and then keeps parsing through the signal until the entirety is parsed through. Then the total number calculated is divided by 2 as half beats are also recorded by the algorithm. Then the total length of the audio file is found on MATLAB (45,019 units) and externally (3 sec). The MATLAB unit is then divided by 3 and then multiplied by 60 to get the MATLAB units for 1 min which is equal to 900,380. Then the total number calculated is divided by 45,019 and then multiplied by 900,380 to identify the beats per minute instead of the beats per 3 seconds.

The final task was to identify the silent areas of the Birds sound audio file. To do this first the signal has to filter to ensure the threshold of the lowpass filter can be reduced. This filtering was done using a moving average filter with a window size of 10. The big window size ensures a lot of the variations in the signal are reduced while the entity of the signal is not filtered out. Then a custom lowpass filter is created with a threshold of 0.05 which ensures that all the signals below the threshold will be identified as silent areas. Then a graphing function with an if the condition is created where it will box and color in the areas with Bird noises and the silent areas will be just boxed in.

3.1) Moving Average Filter Design

A Movie Average Filter or the mean filter is the summation of the current input and the previous and future 'n' inputs, where 'n' is the window range. The Summation is then divided by the window size for the new output at the current time for the inputs given. The equation for a moving average filter is mathematically represented as the fallowing: $y[n] = \frac{1}{3}(x[n-1] + x[n] + x[n+1])$. The moving average filter will essentially take average of the values around the current value thus ensuring that sudden and instantaneous fluctuations would be filtered out and smoothened which can be seen in the example below.

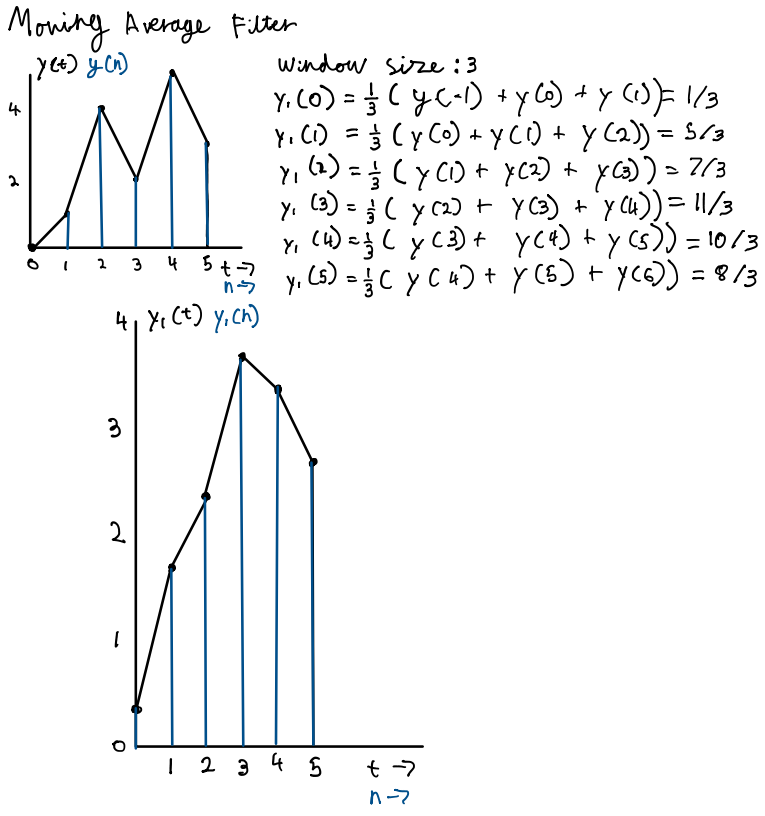


Figure 2 Moving Average Filter Example

In the example for both continuous and discrete domain the moving average filter will smoothen out and sudden dips in the noise and make the signal smoother and clearer to see and analyze. However, this filter also does reduce the absolute max because of the algorithm it uses this can be seen as $\frac{11}{3}$ is the

new max peak of the filtered sample signal. This signal can be represented as the fallowing impulse response:

$$h[n] = \frac{1}{M_1 + M_2 + 1} \sum_{k=-M_1}^{M_2} f(n-k) = \begin{cases} \frac{1}{M_1 + M_2 + 1} & -M_1 \le n \le M_2 \\ 0 \quad otherwise \end{cases}$$

Equation 1, in part 2, ($y[n] = \frac{1}{L}\sum_{k=0}^{L-1} x[n-k],$) can be modified to have an impulse response for the section $x[n-k]$ to get a high pass filter. The high pass filter would be:

$$hpf[n] = \frac{1}{L}\sum_{k=0}^{L-1} \begin{cases} x[n-k] & x[n] \ge th \\ 0 & x[n] < th \end{cases}$$

$$where\ th = minvalue\ frequency\ for\ filter\ to\ allow\ pass\ through$$

This impulse response summation function essentially goes through the signal according to a mean value filter system and then outputs the input signal only if it's frequency passes the threshold set by the 'th' value.

4) Results and discussion

Part 1:

For part one, the only visual output is the following figures for all three sound files which show the frequency of the sound files graphed over time are shown below.

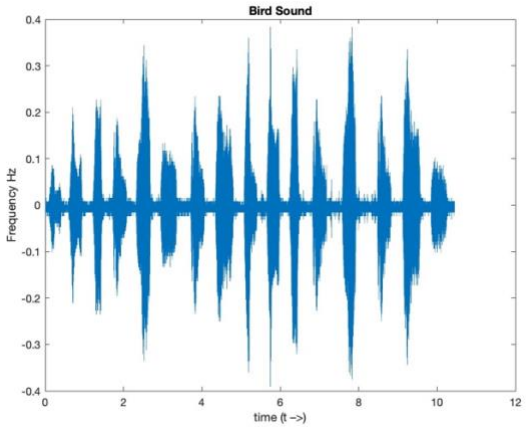


Figure 3 Part 1 Bird Sound

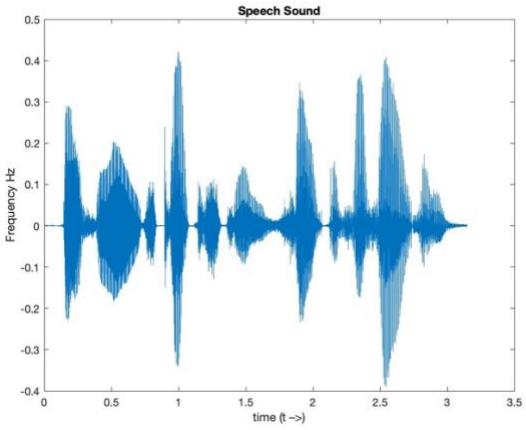


Figure 5 Part 1 Speech Sound

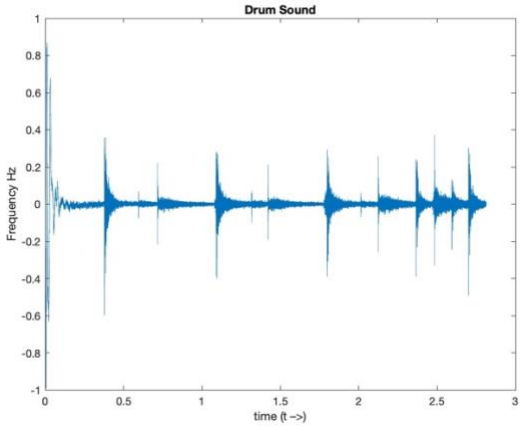


Figure 4 Part 1 Drums Sound

Part 2:

For the second part of the report the 3 filters were implemented into all three sound files. For each filter the ideal filter window was determined after determining the two extremes in the filters and identifying at which window does the best sound filtering occur. After trial-and-error table 1 shows the windows determined for Bird sound file. The outputted graphs with the window sizes are also below.

Table 1 Table of Window Size for Bird Sound File

Filter Type	Slightly Distorted Window	Ideal Window	Extremely Distorted Window
Mean Filter	2	3	500
Weighted Average Filter	2	5	50
Median Filter	3	5	7

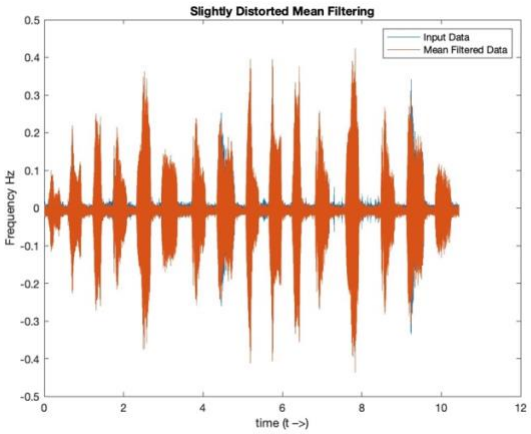


Figure 6 Slightly Distorted Bird Mean Filtering

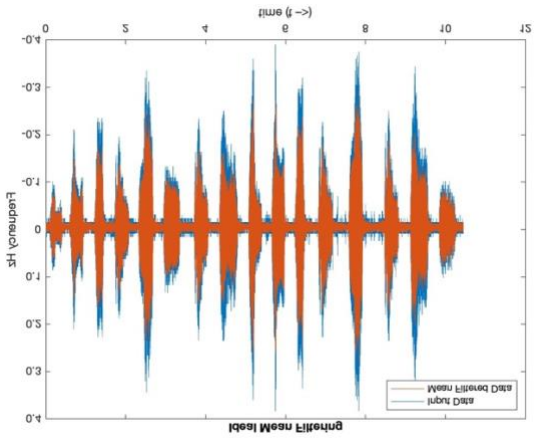


Figure 8 Ideal Bird Mean Filtering

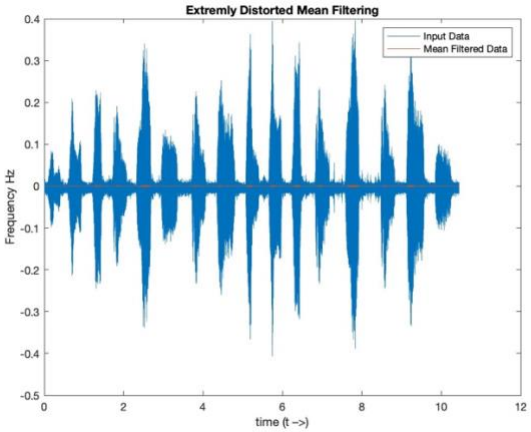


Figure 7 Extremely Distorted Bird Mean Filtering

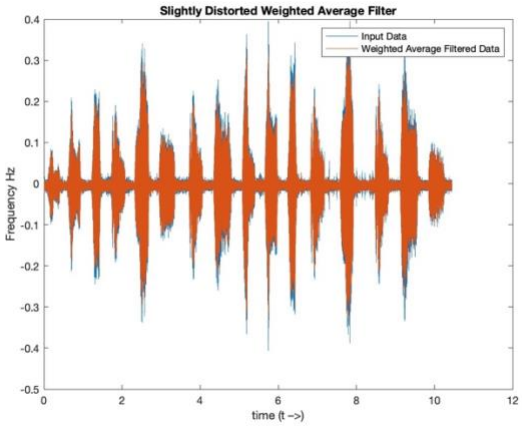


Figure 9 Slightly Distorted Bird Weighted Average Filtering

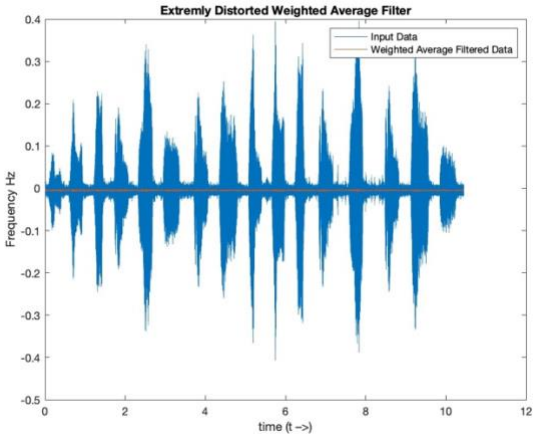


Figure 10 Extremely Distorted Bird Weighted Average Filtering

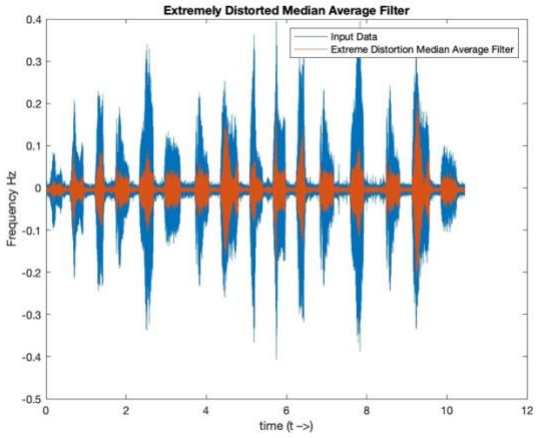


Figure 13 Extremely Distorted Bird Median Average Filtering

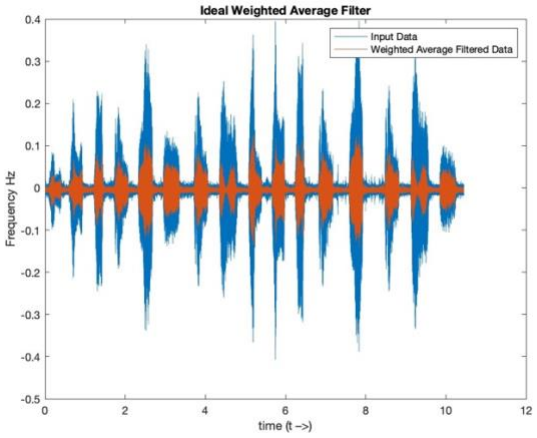


Figure 11 Ideal Bird Weighted Average Filtering

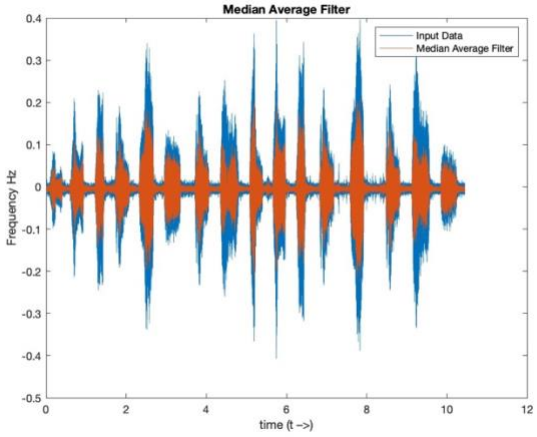


Figure 14 Ideal Bird Median Average Filter

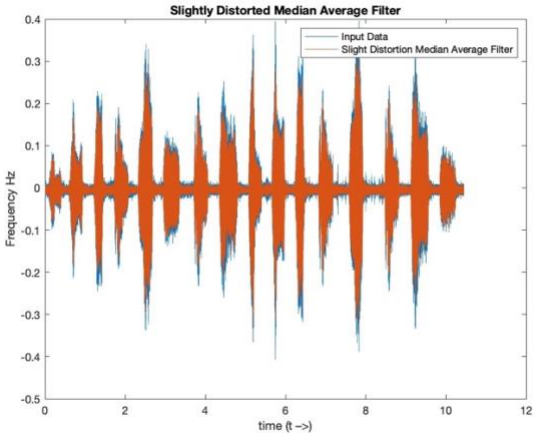


Figure 12 Slightly Distorted Bird Median Average Filtering

After another trial-and-error with using extreme window sizes table 2 shows the windows determined for Drum sound file. The outputted graphs with the window sizes are also below.

Table 2 Table of Window Size for Drum Sound File

Filter Type	Slightly Distorted Window	Ideal Window	Extremely Distorted Window
Mean Filter	2	3/2	25
Weighted Average Filter	2	10	25
Median Filter	3	5	7

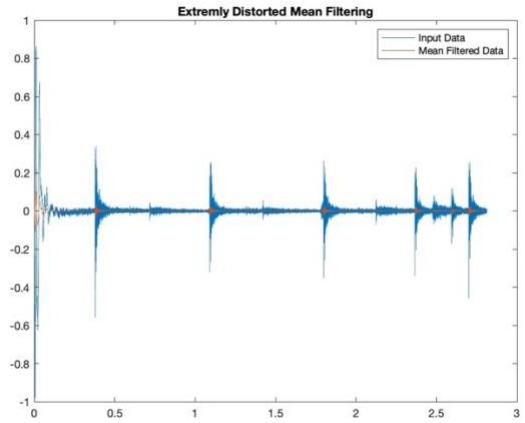


Figure 15 Extremely Distorted Drum Mean Filtering

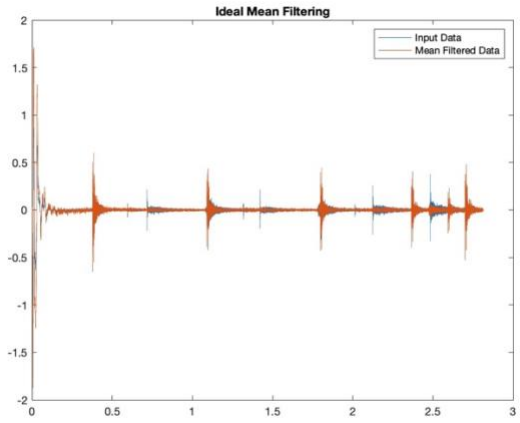


Figure 17 Ideal Drum Mean Filtering

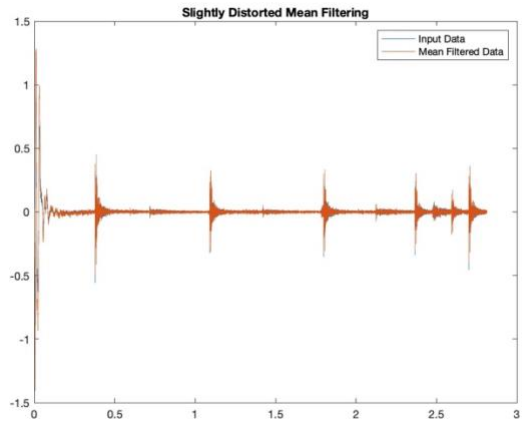


Figure 16 Slightly Distorted Drum Mean Filtering

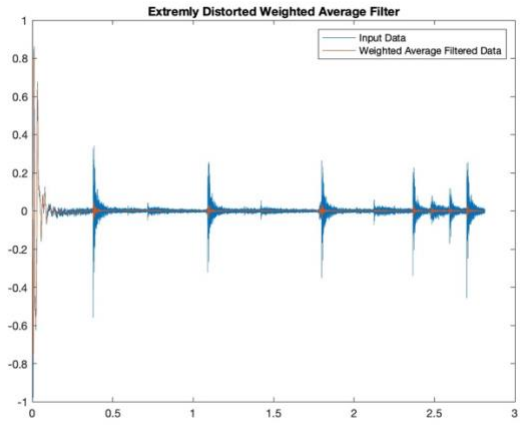


Figure 18 Extremely Distorted Drum Weighted Average Filtering

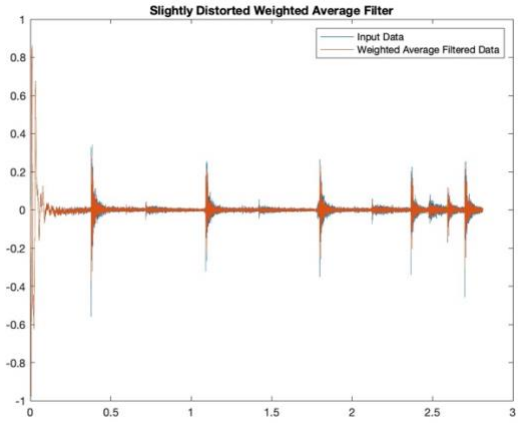


Figure 19 Slightly Distorted Drum Weighted Average Filtering

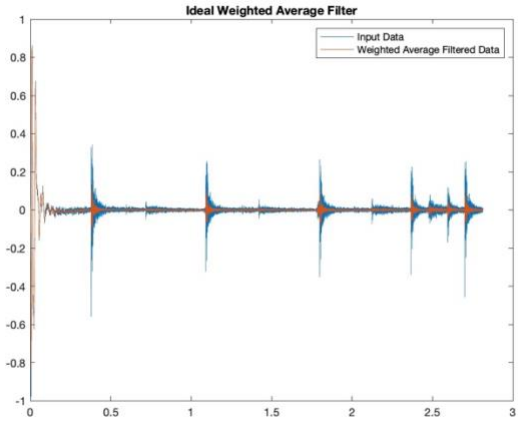


Figure 20 Ideal Drum Weighted Average Filtering

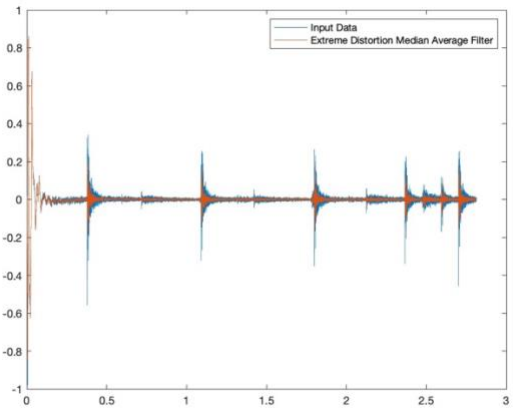


Figure 21 Extremely Distorted Drum Median Average Filtering

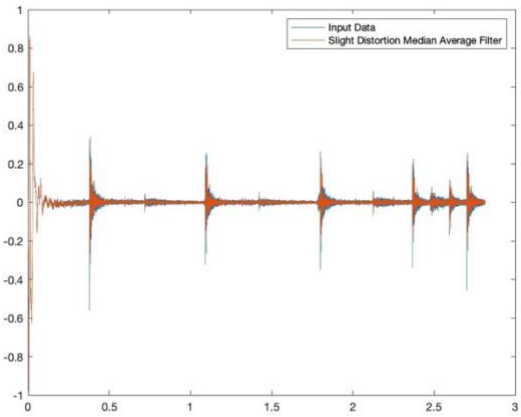


Figure 22 Slightly Distorted Drum Median Average Filtering

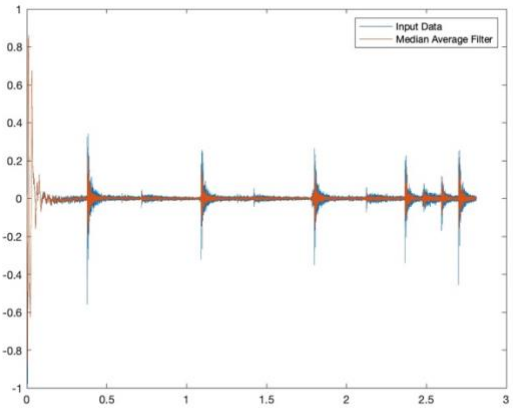


Figure 23 Ideal Drum Median Average Filtering

The final set of testing using extreme window sizes results in table 3, which shows the windows determined for Speech sound file. The outputted graphs with the window sizes are also below.

Table 3 Table of Window Size for Speech Sound File

Filter Type	Slightly Distorted Window	Ideal Window	Extremely Distorted Window
Mean Filter	3	6	100
Weighted Average Filter	7	15	1000
Median Filter	3	5	7

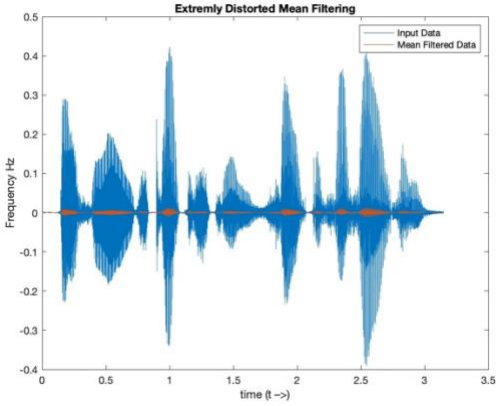


Figure 24 Extremely Distorted Speech Mean Average Filtering

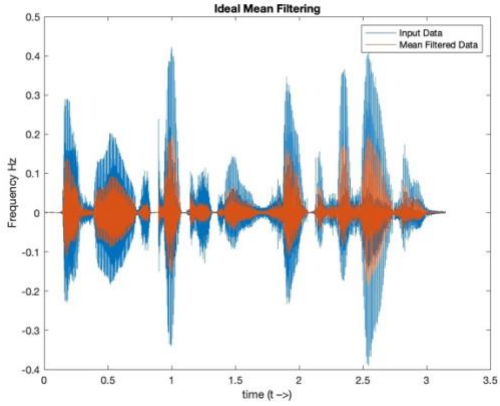


Figure 26 Ideal Speech Mean Average Filtering

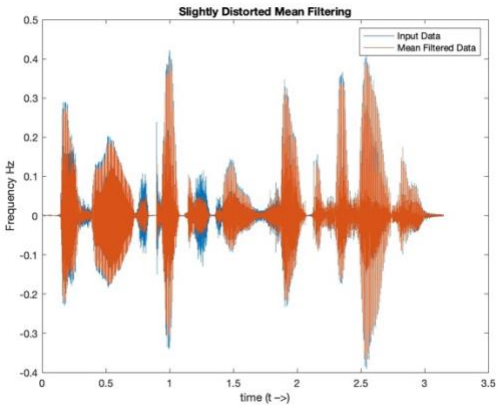


Figure 25 Slightly Distorted Speech Mean Average Filtering

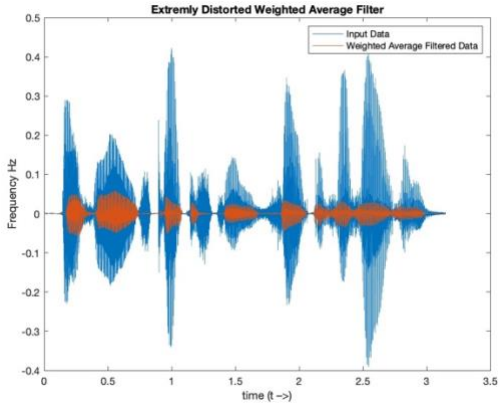


Figure 27 Extremely Distorted Speech Weighted Average Filtering

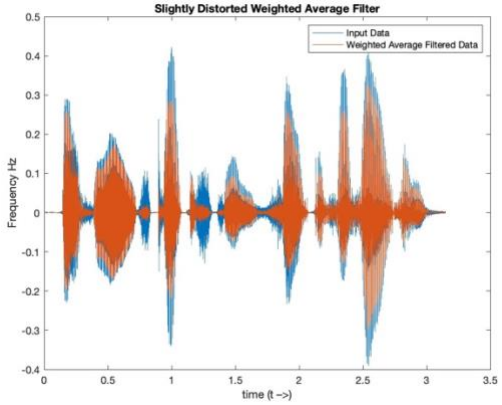


Figure 28 Slightly Distorted Speech Weighted Average Filtering

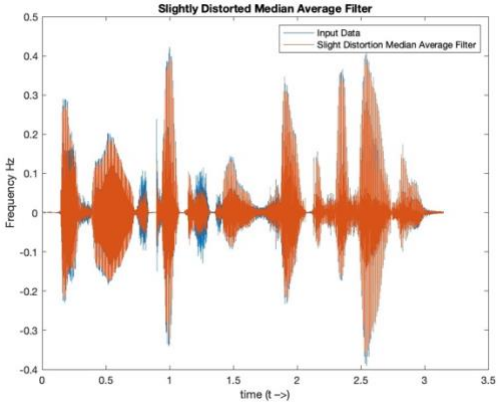


Figure 31 Slightly Distorted Speech Median Average Filtering

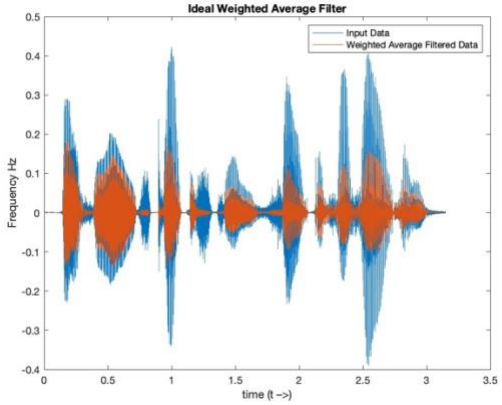


Figure 29 Ideal Speech Weighted Average Filtering

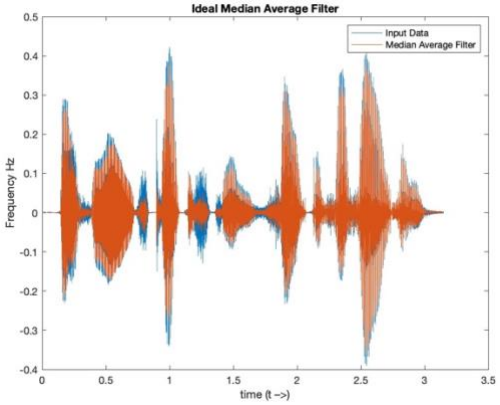


Figure 32 Ideal Speech Median Average Filtering

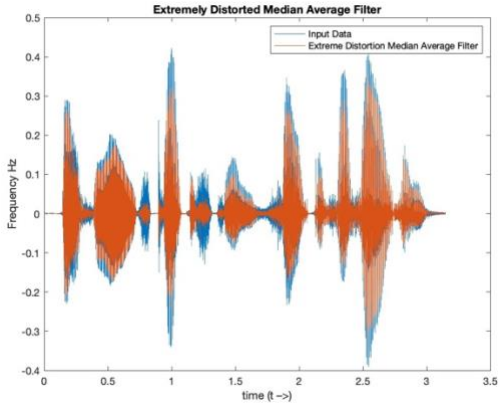


Figure 30 Extremely Distorted Speech Median Average Filtering

Part 3:

The result for how many syllables in the Speech audio file is 10. The representation of the filtered graph used to calculate it is below.

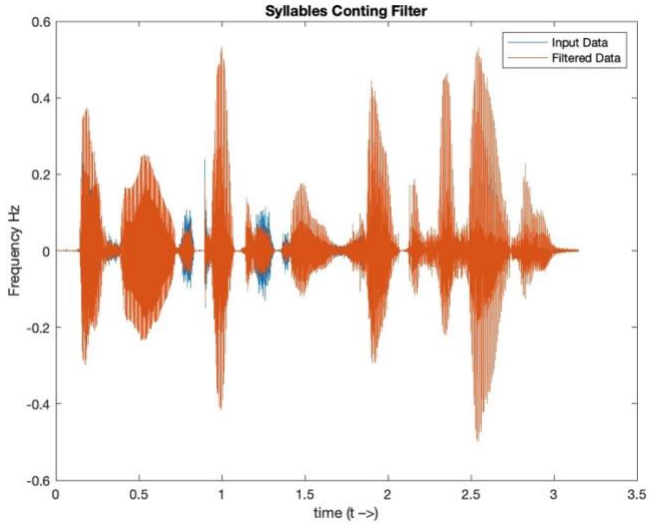


Figure 33 Part 3 Syllables

The result for how many bpm (beats per minute) in the Drum audio file is 80. The representation of the filtered graph used to calculate it is below.

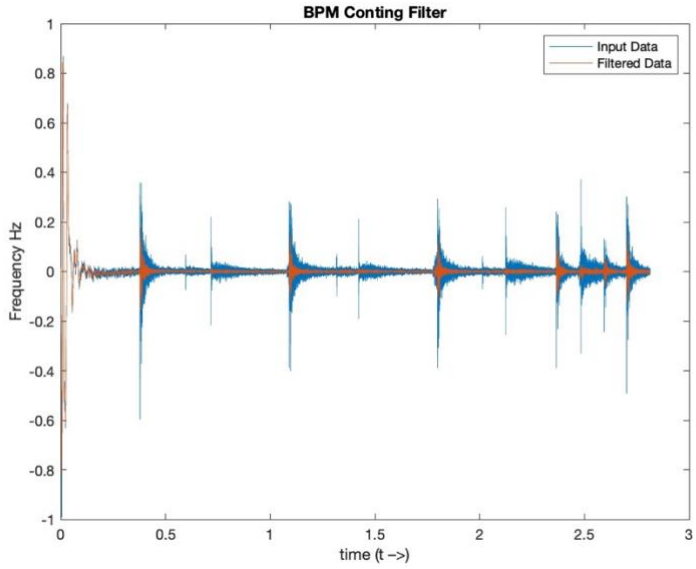


Figure 34 Part 3 BPM

The result for the silence identifier is below in graphical form, to see the array that created it is located in the soft code included with the MATLAB files.

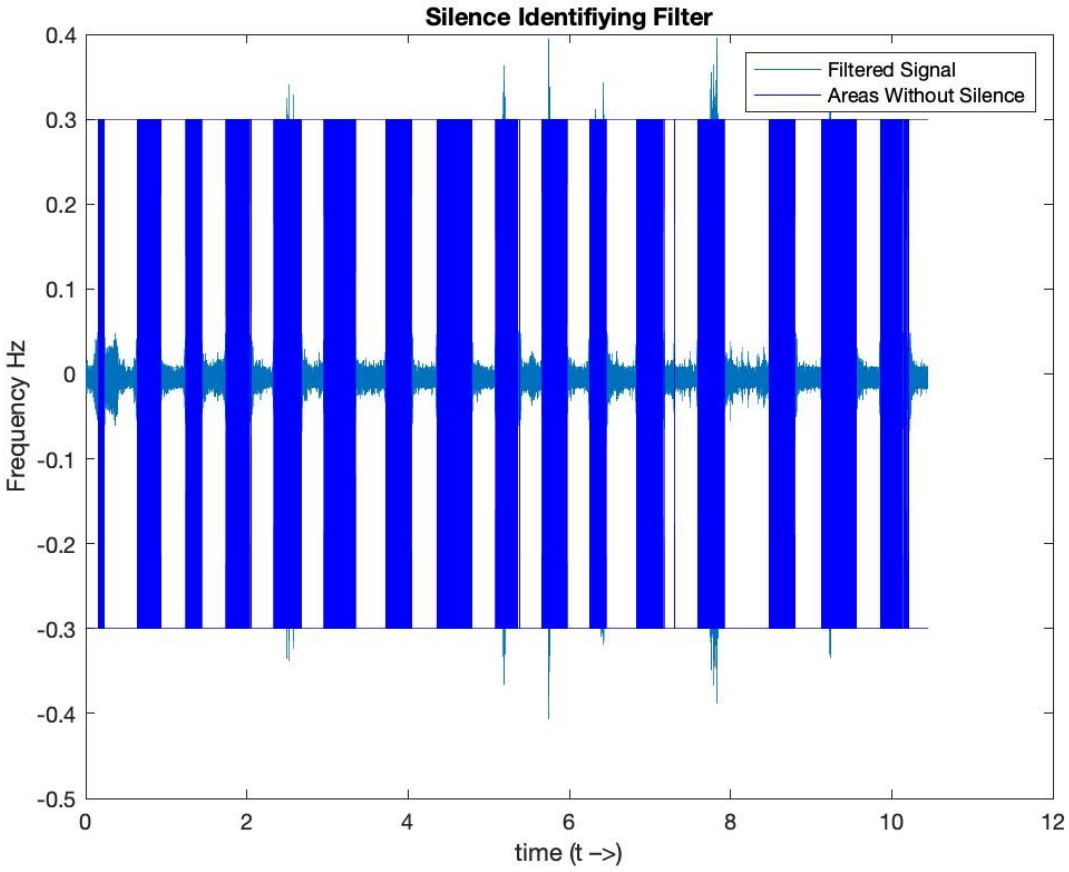


Figure 35 Silence Identifying Graph for Bird Sound File

5) Conclusions and recommendations

Overall, the report focuses on three different concepts in each part. Part 1 was all about discovering and exploring how to input sound files into Matlab and then how to listen to them and how to visually represent them so that one can see the impact a filter has on a sound-based signal. Part one also shows how different frequencies of sound exist for the 3 different types of the sound input. This is seen in the high frequency and sharpness of Figure 3 with the Bird sound input compared to the low frequency and sharpness of Figure 4 with the Drum sound input. There is also the contrast between the high frequency and soft blurry sound input in Figure 5 of the Speech input.

Part 2 is all about understanding and determining how window size and the type of filters affect the output of the input signal. Different signals have different reactions to the three filters. The mean filter caused the amplitude of the filtered Bird signal to be lower while removing sharp changes in signals with a more smoothed-out one. The weighted average acted similarly however the lower frequency signals had a higher rate of decrease in comparison to the higher frequencies. The median filter at least for the Bird signal acted similarly to the mean average filter.

For the Drum sound file, the mean filter filtered out the metronome section of the sound file while keeping the remainder of the signal nearly identical. Whereas the weighted average filter smoothed out all the signals and reduced the frequency of every single part of the signal while making the metronome section seem like a silent pause. The final median filter filtered out nearly identically to the weighted average filter, except it didn't filter out the last metronome beat but kept it in just filtered to a lower frequency.

The Speech sound file reacted differently to the other two sounds. It reduced the amplitude with the mean filter, but it kept the blurry noisy signal noisy and didn't filter it out well. The only time that noise was filtered out was when it went to the extremely distorted window size. The weighted average filter did a bit better as it brought more sharpness and clarity to the blurred area however it still had more blurred area present in comparison to the mean filter and that too with a bigger window in comparison

when looking at the extremely distorted cases. Finally, the Median average filter worked well for some low-frequency noises in-between but for the main high noise and blurriness signals, it didn't have a significant impact.

Part 3 was about applying parts 1 & 2 to solve problems. The syllables problem was solved exactly like it was mentioned in the methodology getting the accurate answer of 10 syllables which can be determined by hearing the sentence. The BPM problem was a bit more difficult with needing to identify the metronome and then convert it from 3 seconds to beats per minute. However, the filters in Part 2 easily got rid of the metronome, therefore an inverse of that filter with a peak counter easily got the value required to solve for the total BMP of 80. The silence identifier was done similarly to the BPM where a filter used in Part 2 was modified so that a parsing identifier can be created to go through and identify areas under a certain threshold which would be identified as white noise or background noise, thus allowing one to accurately box in the areas of silence in the entire sound clip.

Overall, the findings are mostly accurate, an improvement in the median code to make it more dynamic would help improve the quality of the results concerning the median window size in part 2. An improvement in the code and formatting will also make the silence identifier improve as well as currently, it is hard to see the areas with sound as the current code fills in the area that has sound while boxing the areas of silence.

7) Bibliography

Cupertino AF, Pereira HA. 2021. Moving Average Filter - an overview | ScienceDirect Topics. wwwsciencedirectcom. https://www.sciencedirect.com/topics/engineering/moving-average-filter.

MathWorks. 2019. MATLAB Documentation. Mathworkscom. [accessed 2022 Nov 25]. https://www.mathworks.com/help/index.html.

MathWorks. 2022a. A Brief History of MATLAB. wwwmathworkscom. https://www.mathworks.com/company/newsletters/articles/a-brief-history-of-Matlab.html.

MathWorks. 2022b. Lowpass-filter signals - MATLAB lowpass. wwwmathworkscom. https://www.mathworks.com/help/signal/ref/lowpass.html.

Median Filter - an overview | ScienceDirect Topics. 2009. Sciencedirectcom. https://www.sciencedirect.com/topics/computer-science/median-filter.

N/A. 2014 Jun 19. How to apply Average filter, Weighted filter and Median Filter to Noisy Image? @ARN. https://www.Matlabclass.com/2014/06/how-to-apply-average-filter-weighted.html.

N/A. Home - MATLAB Central. wwwmathworkscom. [accessed 2022 Nov 25]. https://www.mathworks.com/Matlabcentral/?s_tid=hc_resources.

Appendix:

Appendix 1: Part 1 Code

```
%STEP1 A1
[y,Fs] = audioread("Birds.wav");
[m, n] = size(y);
%STEP3 A1
soundsc(y,Fs);
pause(10);
%STEP4 A1
audiowrite("Birds1.wav",y,Fs);
%STEP5 A1
N = length(y);
l = linspace(0, N/Fs, N);
plot(l, y);
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
title('Bird Sound');

% STEP1 A2
[y,Fs] = audioread("Drum.wav");
[m, n] = size(y);
% STEP2 A2
Mono = (y(:,1)+y(:,2))/2;
size(Mono)
% STEP3 A2
soundsc(Mono,Fs)
pause(5);
% STEP4 A2
audiowrite("Drum1.wav",Mono,Fs)
% STEP5 A2
N = length(Mono);
t = linspace(0, N/Fs, N);
plot(t, Mono);
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
title('Drum Sound');
```

```
%STEP1 A3
[y,Fs] = audioread("Speech.wav");
size(y);
%STEP3 A3
soundsc(y,Fs);
pause(5);
%STEP4 A3
audiowrite("Speech1.wav",y,Fs)
%STEP5 A3
N = length(y);
t = linspace(0, N/Fs, N);
plot(t, y);
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
title('Speech Sound');
```

Appendix 2: Part 2 Code, Birds

```
%STEP1 A1

[y,Fs] = audioread("Birds.wav");

[m, n] = size(y);

%STEP3 A1

soundsc(y,Fs);

pause(10);

%STEP4 A1

audiowrite("Birds1.wav",y,Fs);

%STEP5 A1

N = length(y);

l = linspace(0, N/Fs, N);

plot(l, y);

hold on;

%STEP6 A1

y = resample(y,16000, Fs);

%Mean Filter 3-point average filter

d=(1/3)*[1 1 1];

c=1;

a = filter(d,c,y);

La = length(a);

b = linspace(0, N/Fs, La);

plot(b,a);

legend('Input Data','Mean Filtered Data');

title('Ideal Mean Filtering');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsMean.wav",a,Fs);

[reada,Fsreada] = audioread("BirdsMean.wav");

soundsc (reada,Fsreada);

pause(16.5);

figure;

%Slightly Distored filtering

d=(1/2)*[1 1 1];

c=1;

a2 = filter(d,c,y);

L2 = length(a);

b2 = linspace(0, N/Fs, L2);

plot(b, y, b2,a2);

legend('Input Data','Mean Filtered Data');

title('Slightly Distorted Mean Filtering');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsMeanSD.wav",a2, Fs);

[reada,Fsreada] = audioread("BirdsMeanSD.wav");

soundsc (reada,Fsreada);

pause(16.5);

figure;

%Extremly Distored filtering

d=(1/500)*[1 1 1];

c=1;

a2 = filter(d,c,y);

L2 = length(a);

b2 = linspace(0, N/Fs, L2);

plot(b, y, b2,a2);

legend('Input Data','Mean Filtered Data');

title('Extremly Distorted Mean Filtering');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsMeanED.wav",a2, Fs);

[reada,Fsreada] = audioread("BirdsMeanED.wav");

soundsc (reada,Fsreada);

pause(16.5);

figure;

%Weighted Average Filter

g = fspecial('gaussian',[1 5],10);

e = conv(y,g);

L3 = lengthI;

d = linspace(0, N/Fs, L3);

plot(b, y, d,e);

legend('Input Data','Weighted Average Filtered Data');

title('Ideal Weighted Average Filter');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsAverage.wav",e,Fs);

[readb,Fsreadb] = audioread("BirdsAverage.wav");

soundsc (readb,Fsreadb);

pause(16.5);

figure;

%SD Weighted Average Filter

g = fspecial('gaussian',[1 2],10);

e2 = conv(y,g);

L3 = length(e2);

d = linspace(0, N/Fs, L3);

plot(b, y, d,e2);

legend('Input Data','Weighted Average Filtered Data');

title('Slightly Distorted Weighted Average Filter');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File
```

```
audiowrite("BirdsAverageSD.wav", e2,Fs);

[readb,Fsreadb] = audioread("BirdsAverageSD.wav");

soundsc (readb,Fsreadb);

pause(16.5);

figure;

%ED Weighted Average Filter

g = fspecial('gaussian',[1 50],10);

e2 = conv(y,g);

L3 = length(e2);

d = linspace(0, N/Fs, L3);

plot(b, y, d,e2);

legend('Input Data','Weighted Average Filtered Data');

title('Extremly Distorted Weighted Average Filter');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsAverageED.wav", e2,Fs);

[readb,Fsreadb] = audioread("BirdsAverageED.wav");

soundsc (readb,Fsreadb);

pause(16.5);

figure;

%Median Average Filter

z = [];

z(1) = median ([0 0 y(1) y(2) y(3)]);

z(2) = median ([0 y(1) y(2) y(3) y(4)]);

for I = 3:length(y)-2

   z(i) = median([y(i-2) y(i-1) y(i) y(i+1) y(i+2)]);

end

Lz = length(z);

dz = linspace(0, N/Fs, Lz);

L10 = length(y);

dl = linspace(0, N/Fs, L10);

plot(dl,y,dz,z);

legend('Input Data','Median Average Filter');

title('Median Average Filter');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsMedian.wav",z,Fs);

[readc,Fsreadc] = audioread("BirdsMedian.wav");

soundsc (readc,Fsreadc);

figure;

%Median Average Filter SD

z1 = [];

z1(1) = median ([0 y(1) y(2)]);

for I = 2:length(y)-1

   z1(i) = median([y(i-1) y(i) y(i+1)]);

end

Lz = length(z1);

dz = linspace(0, N/Fs, Lz);

L10 = length(y);

dl = linspace(0, N/Fs, L10);

plot(dl,y,dz,z1);

legend('Input Data','Slight Distortion Median Average Filter');

title('Slightly Distorted Median Average Filter');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsMedian.wav",z1,Fs);

[readc,Fsreadc] = audioread("BirdsMedian.wav");

soundsc (readc,Fsreadc);

figure;

%Median Average Filter ED

z2 = [];

z2(1) = median ([0 0 0 y(1) y(2) y(3) y(4)]);

z2(2) = median ([0 0 y(1) y(2) y(3) y(4) y(5)]);

z2(3) = median ([0 y(1) y(2) y(3) y(4) y(5) y(6)]);

for I = 4:length(y)-3

   z2(i) = median([y(i-3) y(i-2) y(i-1) y(i) y(i+1) y(i+2) y(i+3)]);

end

Lz = length(z2);

dz = linspace(0, N/Fs, Lz);

L10 = length(y);

dl = linspace(0, N/Fs, L10);

plot(dl,y,dz,z2);

legend('Input Data','Extreme Distortion Median Average Filter');

title('Extremely Distorted Median Average Filter');

xlabel('time (t –>)') ;

ylabel('Frequency Hz') ;

%Write File

audiowrite("BirdsMedian.wav",z2,Fs);

[readc,Fsreadc] = audioread("BirdsMedian.wav");

soundsc (readc,Fsreadc);
```

Appendix 3: Part 2 Code, Drums

```
% STEP1 A2

[y,Fs] = audioread("Drum.wav");

[m, n] = size(y);

% STEP2 A2

Mono = (y(:,1)+y(:,2))/2;

size(Mono)

% STEP3 A2

soundsc(Mono,Fs)

pause(5);

% STEP4 A2

audiowrite("Drum1.wav",Mono,Fs)

% STEP5 A2

N = length(Mono);

t = linspace(0, N/Fs, N);

plot(t, Mono);

hold on;

% STEP6 A2

y = resample(Mono,16000, Fs);

%Mean Filter 3-point average filter

d=(2/3)*[1 1 1];

c=1;

a = filter(d,c,y);

La = length(a);

b = linspace(0, N/Fs, La);

plot(b,a);

legend('Input Data','Mean Filtered
Data');

title('Ideal Mean Filtering');

%Write File

audiowrite("DrumMean.wav",a,Fs);

[reada,Fsreada] =
audioread("DrumMean.wav");

soundsc (reada,Fsreada);

pause(5);

figure;

%Slightly Distored filtering

d=(1/2)*[1 1 1];

c=1;

a2 = filter(d,c,y);

L2 = length(a);

b2 = linspace(0, N/Fs, L2);

plot(b, y, b2,a2);

legend('Input Data','Mean Filtered
Data');

title('Slightly Distorted Mean
Filtering');

%Write File

audiowrite("DrumMeanSD.wav",a2
,Fs);

[reada,Fsreada] =
audioread("DrumMeanSD.wav");

soundsc (reada,Fsreada);

pause(5);

figure;

%Extremly Distored filtering

d=(1/25)*[1 1 1];

c=1;

a2 = filter(d,c,y);

L2 = length(a);

b2 = linspace(0, N/Fs, L2);

plot(b, y, b2,a2);

legend('Input Data','Mean Filtered
Data');

title('Extremly Distorted Mean
Filtering');

%Write File

audiowrite("DrumMeanED.wav",a2
,Fs);

[reada,Fsreada] =
audioread("DrumMeanED.wav");

soundsc (reada,Fsreada);

pause(5);

figure;

%Weighted Average Filter

g = fspecial('gaussian',[1 10],10);

e = conv(y,g);

L3 = length(e);

d = linspace(0, N/Fs, L3);

plot(b, y, d,e);

legend('Input Data','Weighted
Average Filtered Data');

title('Ideal Weighted Average
Filter');

%Write File

audiowrite("DrumAverage.wav",e,F
s);

[readb,Fsreadb] =
audioread("DrumAverage.wav");

soundsc (readb,Fsreadb);

pause(5);

figure;

%SD Weighted Average Filter

g = fspecial('gaussian',[1 2],10);

e2 = conv(y,g);

L3 = length(e2);

d = linspace(0, N/Fs, L3);

plot(b, y, d,e2);

legend('Input Data','Weighted
Average Filtered Data');

title('Slightly Distorted Weighted
Average Filter');

%Write File

audiowrite("DrumAverageSD.wav",
e2,Fs);

[readb,Fsreadb] =
audioread("DrumAverageSD.wav");

soundsc (readb,Fsreadb);

pause(5);

figure;

%ED Weighted Average Filter

g = fspecial('gaussian',[1 25],10);
```

```
e2 = conv(y,g);

L3 = length(e2);

d = linspace(0, N/Fs, L3);

plot(b, y, d,e2);

legend('Input Data','Weighted Average Filtered Data');

title('Extremly Distorted Weighted Average Filter');

%Write File

audiowrite("DrumAverageED.wav", e2,Fs);

[readb,Fsreadb] = audioread("DrumAverageED.wav") ;

soundsc (readb,Fsreadb);

pause(5);

figure;

%Median Average Filter

z = [];

z(1) = median ([0 0 y(1) y(2) y(3)]);

z(2) = median ([0 y(1) y(2) y(3) y(4)]);

for i = 3:length(y)-2

   z(i) = median([y(i-2) y(i-1) y(i) y(i+1) y(i+2)]);

end

Lz = length(z);

dz = linspace(0, N/Fs, Lz);

L10 = length(y);

dl = linspace(0, N/Fs, L10);

plot(dl,y,dz,z);

legend('Input Data','Median Average Filter');

%Write File

audiowrite("DrumMedian.wav",z,Fs );

[readc,Fsreadc] = audioread("DrumMedian.wav");

soundsc (readc,Fsreadc);

figure;

%Median Average Filter SD

z1 = [];

z1(1) = median ([0 y(1) y(2)]);

for i = 2:length(y)-1

   z1(i) = median([y(i-1) y(i) y(i+1)]);

end

Lz = length(z1);

dz = linspace(0, N/Fs, Lz);

L10 = length(y);

dl = linspace(0, N/Fs, L10);

plot(dl,y,dz,z1);

legend('Input Data','Slight Distortion Median Average Filter');

%Write File

audiowrite("DrumMedian.wav",z1, Fs);

[readc,Fsreadc] = audioread("DrumMedian.wav");

soundsc (readc,Fsreadc);

figure;

%Median Average Filter ED

z2 = [];

z2(1) = median ([0 0 0 y(1) y(2) y(3) y(4)]);

z2(2) = median ([0 0 y(1) y(2) y(3) y(4) y(5)]);

z2(3) = median ([0 y(1) y(2) y(3) y(4) y(5) y(6)]);

for i = 4:length(y)-3

   z2(i) = median([y(i-3) y(i-2) y(i-1) y(i) y(i+1) y(i+2) y(i+3)]);

end

Lz = length(z2);

dz = linspace(0, N/Fs, Lz);

L10 = length(y);

dl = linspace(0, N/Fs, L10);

plot(dl,y,dz,z2);

legend('Input Data','Extreme Distortion Median Average Filter');

%Write File

audiowrite("DrumMedian.wav",z2, Fs);

[readc,Fsreadc] = audioread("DrumMedian.wav");

soundsc (readc,Fsreadc);
```

Appendix 4: Part 2 Code, Speech

```
%STEP1 A3
[y,Fs] = audioread("Speech.wav");
size(y);
%STEP3 A3
soundsc(y,Fs);
pause(5);
%STEP4 A3
audiowrite("Speech1.wav",y,Fs)
%STEP5 A3
N = length(y);
t = linspace(0, N/Fs, N);
plot(t, y);
hold on;
%STEP6 A3
y = resample(y,16000, Fs);
%Mean Filter 3-point average filter
d=(1/6)*[1 1 1];
c=1;
a = filter(d,c,y);
La = length(a);
b = linspace(0, N/Fs, La);
plot(b,a);
legend('Input Data','Mean Filtered
Data');
title('Ideal Mean Filtering');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechMean.wav",a,Fs
);
[reada,Fsreada] =
audioread("SpeechMean.wav");
soundsc (reada,Fsreada);
pause(5);
figure;
%Slightly Distored filtering
d=(1/3)*[1 1 1];
c=1;
a2 = filter(d,c,y);
L2 = length(a);
b2 = linspace(0, N/Fs, L2);
plot(b, y, b2,a2);
legend('Input Data','Mean Filtered
Data');
title('Slightly Distorted Mean
Filtering');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechMeanSD.wav",a
2,Fs);
[reada,Fsreada] =
audioread("SpeechMeanSD.wav");
soundsc (reada,Fsreada);
pause(5);
figure;
%Extremly Distored filtering
d=(1/100)*[1 1 1];
c=1;
a2 = filter(d,c,y);
L2 = length(a);
b2 = linspace(0, N/Fs, L2);
plot(b, y, b2,a2);
legend('Input Data','Mean Filtered
Data');
title('Extremly Distorted Mean
Filtering');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechMeanED.wav",a
2,Fs);
[reada,Fsreada] =
audioread("SpeechMeanED.wav");
soundsc (reada,Fsreada);
pause(5);
figure;
%Weighted Average Filter
g = fspecial('gaussian',[1 15],10);
e = conv(y,g);
L3 = length(e);
d = linspace(0, N/Fs, L3);
plot(b, y, d,e);
legend('Input Data','Weighted
Average Filtered Data');
title('Ideal Weighted Average
Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechAverage.wav",e,
Fs);
[readb,Fsreadb] =
audioread("SpeechAverage.wav");
soundsc (readb,Fsreadb);
pause(5);
figure;
%SD Weighted Average Filter
g = fspecial('gaussian',[1 7],10);
e2 = conv(y,g);
L3 = length(e2);
d = linspace(0, N/Fs, L3);
plot(b, y, d,e2);
legend('Input Data','Weighted
Average Filtered Data');
title('Slightly Distorted Weighted
Average Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechAverageSD.wav
",e2,Fs);
[readb,Fsreadb] =
audioread("SpeechAverageSD.wav"
);
soundsc (readb,Fsreadb);
pause(5);
figure;
%ED Weighted Average Filter
g = fspecial('gaussian',[1 1000],10);
e2 = conv(y,g);
L3 = length(e2);
d = linspace(0, N/Fs, L3);
plot(b, y, d,e2);
legend('Input Data','Weighted
Average Filtered Data');
title('Extremly Distorted Weighted
Average Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechAverageED.wav
",e2,Fs);
[readb,Fsreadb] =
audioread("SpeechAverageED.wav"
);
soundsc (readb,Fsreadb);
pause(5);
figure;
%Median Average Filter
z = [];
z(1) = median ([0 0 y(1) y(2) y(3)]);
z(2) = median ([0 y(1) y(2) y(3)
y(4)]);
for i = 3:length(y)-2
    z(i) = median([y(i-2) y(i-1) y(i)
y(i+1) y(i+2)]);
end
Lz = length(z);
dz = linspace(0, N/Fs, Lz);
L10 = length(y);
dl = linspace(0, N/Fs, L10);
plot(dl,y,dz,z);
legend('Input Data','Median
Average Filter');
title('Ideal Median Average Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechMedian.wav",z,
Fs);
[readc,Fsreadc] =
audioread("SpeechMedian.wav");
soundsc (readc,Fsreadc);
figure;
%Median Average Filter SD
z1 = [];
z1(1) = median ([0 y(1) y(2)]);
for i = 2:length(y)-1
    z1(i) = median([y(i-1) y(i)
y(i+1)]);
end
Lz = length(z1);
dz = linspace(0, N/Fs, Lz);
L10 = length(y);
dl = linspace(0, N/Fs, L10);
plot(dl,y,dz,z1);
legend('Input Data','Slight
Distortion Median Average Filter');
```

```
title('Slightly Distorted Median
Average Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechMedian.wav",z1
,Fs);
[readc,Fsreadc] =
audioread("SpeechMedian.wav");
soundsc (readc,Fsreadc);
figure;
%Median Average Filter ED
z2 = [];
z2(1) = median ([0 0 0 y(1) y(2)
y(3) y(4)]);
z2(2) = median ([0 0 y(1) y(2) y(3)
y(4) y(5)]);
z2(3) = median ([0 y(1) y(2) y(3)
y(4) y(5) y(6)]);
for i = 4:length(y)-3
    z2(i) = median([y(i-3) y(i-2) y(i-
1) y(i) y(i+1) y(i+2) y(i+3)]);
end
Lz = length(z2);
dz = linspace(0, N/Fs, Lz);
L10 = length(y);
dl = linspace(0, N/Fs, L10);
plot(dl,y,dz,z2);
legend('Input Data','Extreme
Distortion Median Average Filter');
title('Extremely Distorted Median
Average Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
%Write File
audiowrite("SpeechMedian.wav",z2
,Fs);
[readc,Fsreadc] =
audioread("SpeechMedian.wav");
soundsc (readc,Fsreadc);
```

Appendix 5: Part 3 Codes

```
%Speech:
%STEP1 A3
[y,Fs] =
audioread("Speech.wav");
size(y);
%STEP5 A3
N = length(y);
t = linspace(0, N/Fs, N);
plot(t, y);
hold on;
%STEP6 A3
y = resample(y,16000, Fs);
d=(1/2.23)*[1 1 1];
c=1;
y = filter(d,c,y);
La = length(y);
b = linspace(0, N/Fs, La);
plot(b,y);
legend('Input Data','Filtered
Data');
title('Syllables Conting
Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
thL=0.5;
rise = 0;
for n=3:length(y)
   if y(n-1) - y(n-2) > 0 &&
y(n)-y(n-1) < 0 && y(n-1) >=
thL
      rise = rise + 1;
   end
end
```

```
%BPM:
% STEP1 A2
[y,Fs] =
audioread("Drum.wav");
[m, n] = size(y);
Mono = (y(:,1)+y(:,2))/2;
size(Mono)
N = length(Mono);
t = linspace(0, N/Fs, N);
plot(t, Mono);
hold on;
% STEP6 A2
y = resample(Mono,16000,
Fs);
g = fspecial('gaussian',[1
5],10);
e = conv(y,g);
lp1 = lowpass(e,0.09);
p2 = length(lp1);
b = linspace(0, N/Fs, p2);
plot(b,lp1);
legend('Input Data','Filtered
Data');
title('BPM Conting Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
thL=0.7;
rise1 = 0;
for n=3:length(y)
   if y(n-1) - y(n-2) > 0 &&
y(n)-y(n-1) < 0 && y(n-1) >=
thL
      rise1 = rise1 + 1;
   end
end
timebeat = length(y);
newtime = ((timebeat/3)*60);
bpm =
(((rise1/2)/timebeat)*newtim
e);
```

```
%Silence:
%STEP1 A1
[y,Fs] =
audioread("Birds.wav");
[m, n] = size(y);
%STEP5 A1
N = length(y);
l = linspace(0, N/Fs, N);
plot(l, y);
hold on;
%STEP6 A1
y = resample(y,16000, Fs);
d=(1/10)*[1 1 1];
e=1;
a = filter(d,e,y);
La = length(a);
b = linspace(0, N/Fs, La);
plot(b,a);
legend('Input Signal','Filtered
Signal');
title('Low-Pass Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
figure;
z = [ ];
z2 = [ ];
for c = 1:La
      if y(c) < 0.05
         z(c) = 0.3;
         z2(c) = -0.3;
      else
         z(c) = 0;
         z2(c) = 0;
      end
end
plot(b,y,b,z,'b',b,z2,'b');
legend('Filtered
Signal','Areas Without
Silence');
title('Silence Identifiying
Filter');
xlabel('time (t –>)') ;
ylabel('Frequency Hz') ;
```