

Digital Noise Reduction on Audio Files using MATLAB

Luis Beltran
Department of Electrical and Computer

Engineering
California State University Chico

Chico, United States
lbeltran@mail.csuchico

Abstract— In this paper, we present the process needed to improve the quality sound of a song that has noise in their audio. The song that will be examined and processed is a famous song from the 70s. This work mainly focuses on experimenting with Fourier Methods and Digital Filters to produce a Digital Noise Reduction algorithm that will provide a transition from a noisy audio clip to a clean audio clip, using the software MATLAB. Data acquisition and Results obtained will be provided to demonstrate the effectiveness of the methods briefly discussed.

I. INTRODUCTION

This study of Digital Noise Reduction aims to filter and remove noise from an audio file; more specifically a hiss sound from a song in .wav format. Hiss noise is a broadband noise whose noise energy is distributed over the entire audible spectrum but with more intensity at higher frequencies. Hiss is very similar to blue noise; in physics and audio engineering the color of noise refers to the power of the spectrum of a noise signal.

Almost all audio circuits create some type of noise which is a problem. This might happen because of blown speakers, amplifier gain in mic lines or more specifically for this research, analog audio equipment. Most of the music during the 1950s through early 2000s has some type of noise carried within their tracks which might be caused by audio equipment. However, in the music industry analog technology is often desirable since it produces nonlinear responses that are very difficult to recreate in the digital domain. Music industries work with very sophisticated music software to achieve high resolution audio for their listeners. Filters play an important role to achieve the high demand for better audio quality and resolution.

Some of the things we need to have in consideration are the methods that are discussed in this paper. As briefly discussed, the study of Fourier Methods for Spectral Analysis will help us determine valuable information about our input signal which will lead the project into designing a digital filter to remove unwanted signals. Filters come in two different types FIR and IIR. FIR stands for finite impulse response, which are purely digital filters, and their most important feature is that they can be implemented with integer math. IIR stands for infinite impulse response and these are filters that can achieve specific filtering characteristics using less memory and calculation than a similar FIR low-pass filter. For this paper, the go to filter that will be applied will be an IIR low pass filter due to its property of removing high frequency content, which hiss carries high frequency components.

Some of the solutions to this problem of hiss noise would be buying expensive equipment when recording any type of video or audio. Film industry uses Mic booms and other expensive equipment to carry out the noisy signals in their raw data. Other methods to eliminate any type of undesirable noise will be to use an Audio Editing Software

such as Audacity or Goldwave. This paper is intended to study Fourier Methods for Spectral Analysis and Digital Filter Implementation to attenuate unwanted signals of a .wav file using the software MATLAB.

II. MATERIALS AND METHOD

The famous song: “The Immigrant Song” by the famous English band “Led Zeppelin” was processed, analyzed, and filtered to produce a better sound quality. The studio version of this 1970s masterpiece comes with three incrementing levels of hiss noise just in the first 1.7seconds. After some research, I came across a website that indicates that the band intentionally uses an echo unit to create that hiss noise. [23] An echo unit creates a copy of the signal and after an interval creates a copy and replays it. This song was meant to be humorous to the English band from the audio effects to the lyrics. After reading these facts about the band, we can still work on improvement on the first seconds of this famous recording. Methods have been briefly discussed in our introduction section and will be further discussed in the following section.

A. Materials

Various types of websites and software were used to achieve a proper DNR on audio files. Some of the websites used to acquire our song or data were YouTube. [a] After performing an online conversion on the audio track to a .wav file it was time to pre-process the audio signal using Audacity. [b] This music editor software is free to download on the web and is very useful to do simple stuff as audio trimming and basic audio effects. Audacity was used to trim the audio clip to make it more manageable when analysis was performed. By trimming the audio using this software we can precisely know when the hiss starts and when it ends. Something else that I picked up during the pre-process step was that we can also hear the drummer of the band counting for the band members to start the song. The precise time was the analysis, and the filter will take the part in the first 1.68s of the audio trim which lasts 4s. Our next step is to upload the .wav file to MATLAB [c] which will help us with our methods.




[a]	[b]	[c]
		

Table 1: Websites and Software used. [14]

B. Methods

Once the .wav file is uploaded into MATLAB using the audioread function. Our sample frequency is at 44.1kHz which makes sense being the standard sample rate for most consumer audio, like cassettes and CDs [5]. When the .wav has been successfully uploaded into MATLAB we can start performing several analyses to implement the required filter that will reduce hiss noise. In Fig 1 we can see the time domain signal of our input signal, which provides the input reading on an Amplitude vs Time graph. From the time graph we can see where the levels of hiss are present. The time domain representation is not giving us sufficient information to start implementing our filter. The next step taken to our signal was to perform an FFT.

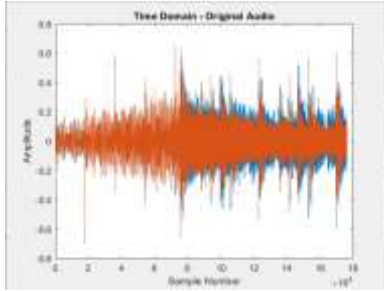


Fig 1: Original Audio in Time Domain

What is FFT? Well, first we need to consider a Fourier transform which decomposes a function into its constituents' frequencies which means that it takes our time domain representation and takes it to the frequency domain. By using the function FFT which computes the discrete Fourier transform (DFT) of our Data using fast Fourier transform (FFT) algorithm. The fast Fourier transform can be mathematically defined as follows

$$Y(k) = \sum_{j=1}^n X(j) W_n^{(j-1)(k-1)}$$

where

$$W_n = e^{(-2\pi i)/n}$$

is one of n -roots of unity [19].

These equations provide different frequency representations of our original signal. Figures will be provided below focusing on each command used. The results from implementing the FFT function are provided in Fig 2 where we can see the frequency response representation of our original signal. From this figure we can see some low and high frequencies but are hard to distinguish.

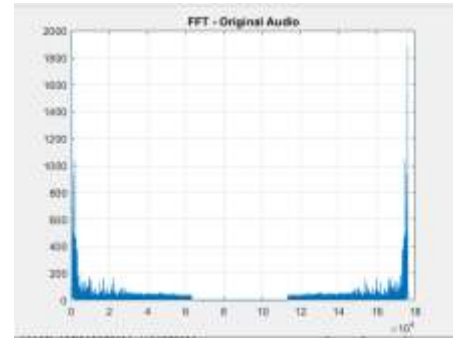


Fig 2: Original Audio in Frequency Domain

To have a better visualization of the Fourier transform we can use the function fftshift which will rearrange the output of our fft by moving the zero-frequency component to the center of the graph. This will be useful to examine the Fourier transform with the zero-frequency component in the middle of the spectrum, let refer to Fig 3 for a better understanding [18].

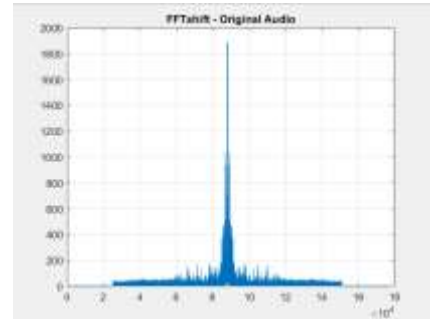


Fig 3: Representation of fftshift

This graph provides a better representation of the frequencies in our signal from our fig 2. It just rearranges the x-axis for better representations. We can aim to keep frequency between 500hz to 2500hz. [6] [8]

Then for another spectrum analysis we can get a periodogram of the signal. A periodogram will show us the power each frequency contains. Periodogram is an estimate of the spectral density of the signal. We can perform an average pwelch spectrum which will give us the average of the periodogram. Fig 4 will provide the average fft of the original signal, which leads us to see the slope of the hiss noise. This will be useful to determine which order of our low pass filter would be suitable to flatten or even get us a negative slope of the filtered signal.[20]

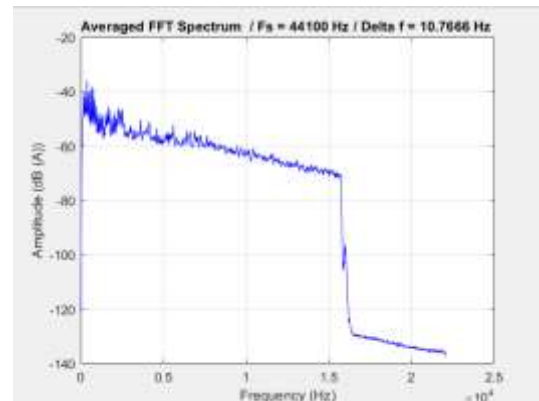


Fig 4: Average FFT

From the results obtained in Fig 4, we can say that there is a 60dB loss which could lead us perhaps to use a third order filter. [6] [7]

We can now proceed to obtain our spectrogram which is just a visual representation of the spectrum of the frequencies of our signal; this is going to help us determine our signal strengths. Spectrograms are usually a two-dimensional graph (Freq vs Time) with a third dimension representing colors. The amplitude of any frequency at any time is represented in the third dimension in color. Dark blue corresponds to low amplitudes and it goes all the way down to yellow. Brighter colors correspond to progressively stronger amplitudes, meaning louder amplitudes. The steps taken to obtain the spectrogram graphs were to use the command `specgram` in MATLAB. Fig 5 will have provided us an insight of the spectrogram of the original signal.

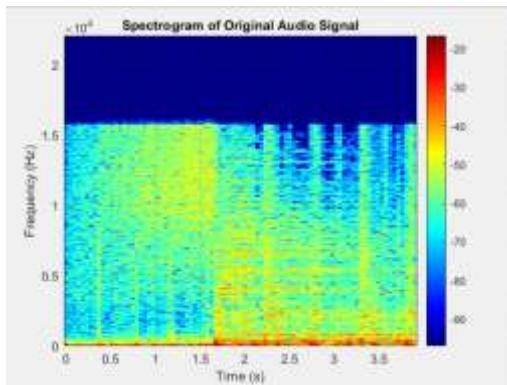


Fig 5: Spectrogram of Original Audio Signal

From our spectrogram we can clearly see the intensity of change of colors from 0s to roughly 1.7s. This is a clear representation of the levels of hiss present in the audio signal. We can examine and determine that the max sensitivity is between 1kHz and 5kHz; therefore, if our noise spectrum has an amplitude above 2.5kHz, we will hear it, and this is what needs to be filtered! [23] [24]

As discussed on the previous progress report, a Low Pass filter removes high frequency content which in our case hiss is typically in the higher frequency or the high pitch. A Bandpass filter will remove Low and High frequency contents while leaving an intermediate band of frequencies intact. [15] This is excellent because we know what the low and high frequencies are for our original signal. From our results we can inspect that it was quieter in the 500hz to 2500hz, therefore no need to reduce sound in this frequency range. Based on this information a suitable filter for this DNR algorithm would be a passband filter. We can use a Bandpass for only the segment where the hiss noise is present this way, we can only filter out the unwanted noise.

What kind of passband filter should we design? The type of filter that will have a flat frequency response in the passband is called a Butterworth filter, named after 1900s physicist and engineer Stephen Butterworth. This filter design will pass all frequency signals in the cut-off range but will suppress signals with higher frequencies and lower than our cut-off frequencies. **[14]. However, the price paid for using a Butterworth filter is that it does achieve this pass

band flatness at the expense of a wide band transition as the filter changes from the pass band to the stop band. [2] [3] This type of filter is usually used for audio quality due to its response. This will be a filter to try and implement for our noisy audio signal due to its characteristics and it is widely used on audio quality. The order of the filter used in this DNR project was a 4th order filter.

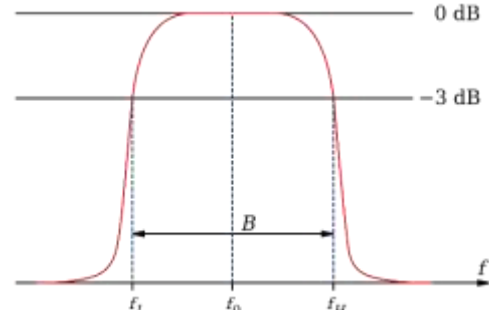


Fig 6: Bandpass filter representation

Figure 6 provides a great view on how the bandpass filter looks and where our low and high frequencies are. We can refer to the code appendix to see the actual implementation of the filter. The cutoff frequency is from a low frequency of 500 Hz and a high frequency of 2500 Hz, while our sample frequency is 44.1kHz, Nyquist frequency is 22050, and for our non-dimensional frequency we can divide our Nyquist frequency by our cutoff frequencies. This information is crucial for the filter implementation, we can refer to Fig 7 which is an exemption from the MATLAB code that successfully filter our signal. [2] [17] [16]

```

75 % Removal of Hiss Noise between t = 0 and t = 1.68s
76 % use of bandpass filter to remove noise below 500 and above 2500 Hz
77 data_filtered = Data; %NEW CLEAN VARIABLE
78 t_min = 0; %Hiss STARTS
79 t_max = 1.68; %LEVELS OF HISS DECREASE
80 ind = 1:fix(t_min*Fs:t_max*Fs-1); %setting index for filter application
81
82 % keep only signal content extracted by bandpass frequencies
83 Fs = 44100; %Sample Frequency
84 nyquist_freq = Fs/2; %Nyquist Frequency
85 low_f = 500; %low frequency
86 high_f = 2500; %high frequency
87 cutoff_freqs = [500 2500]; %cutoff frequencies
88 Wn = nyquist_freq./cutoff_freqs; %non-dimensional frequency
89 N = 3; %order of Butterworth filter
90
91 % Using Built-in-Functions to design Filters
92 %=====
93 %we can work on a better filter design with given parameters on top%%
94 % signal_filtered = resample(signal,1);
95 [filtb,filta] = butter(N,2/Fs*[low_f high_f],'bandpass');

```

Fig 7: MATLAB filter implementation

III. RESULTS AND DISCUSSION

Once all filter parameters are found, we can implement the filter using the function `filtfilt` from MATLAB which implements the filter to the desired range, which in this case is from 0s to 1.68s where the levels of hiss go off and the actual song starts. We can proceed and obtain different types of responses for the filter such as Z-plane, phase and magnitudes responses, and impulse response. These filter responses will tell us how the system is behaving. Some of the following figures 8 through figure 10 will provide just that, filter responses results.

A. Results

In this section we will discuss the different results obtained from the spectral analysis and filtering implementation.

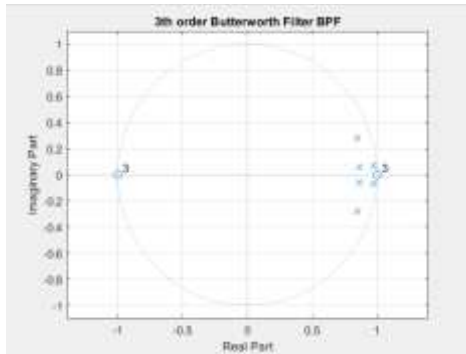


Fig 8: Filter Z-plot Response

Figure 8 provides the Z-plane of the filter which tells us that the system is stable because all its zeros and poles inside the unit circle, therefore this system does meet the Nyquist stability criteria.

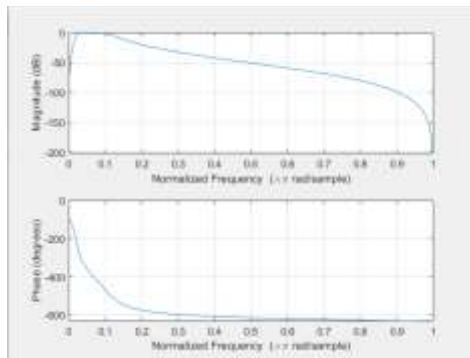


Fig 9: Phase and Magnitude Responses

The figure shown above Fig 9, provides the reader a view of the magnitude response and the phase response of the filter.

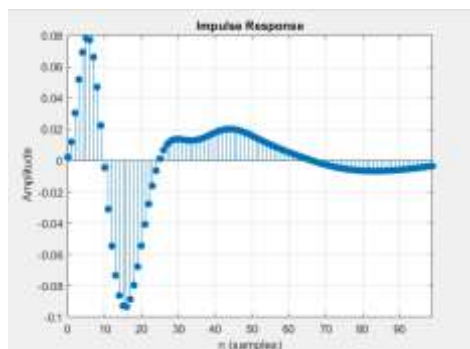


Fig 10: Impulse Response

The impulse response for this filter is in the decaying component; as we can see when samples are increasing, or response decays.

The Final step would be to hear the filter audio output from this filter implementation to the hiss noise present in our song; therefore, an extraction of the filter signal using the variable given to the lines of filtfilt must be performed. I

named the filter audio “Clean_Audio.wav” which will appear as a new file in MATLAB libraries. We can use the sound () command to listen to our audio in MATLAB. Running the DNR algorithm in MATLAB produces this new file that when played does indeed attenuates the hiss noise in the audio clip.

To have a better understanding of what frequencies have been attenuated we can perform again several analyses such the ones discussed in this paper. The following figures are an attempt to have a better understanding of what is happening to our signal now that it has been filtered.

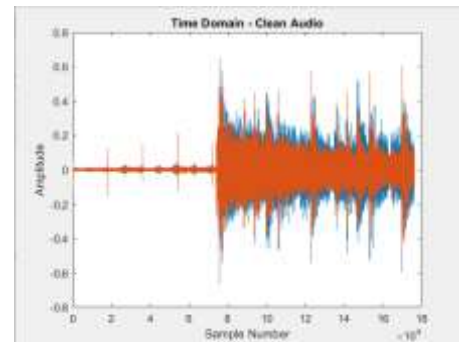


Fig 11: Clean Audio in Time Domain

This graph is very similar to Fig 1 which displays the unfiltered audio signal. Figure 11 provides the clean audio in time domain where we can see how the noise in the first 1.7s have been suppressed. We can also still see the hiss spikes or the levels of incrementing hiss we discuss in the paper. It is very clearly that we have 3 spikes which corresponds to the 3 hisses.

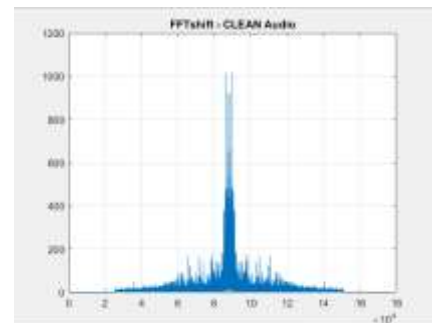


Fig 12: Clean Audio using fftshift

Figure 12 resembles the clean audio's FFT, which is very the counterpart of figure 3. Here we can see that we don't have those spikes of 500 Hz to 2000 Hz

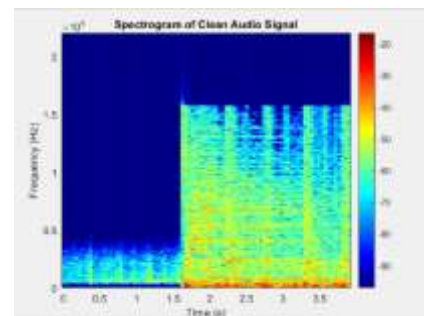


Fig 13: Clean Audio Spectrogram

This figure provides the clean audio spectrogram of our clean signal. This gives a better understanding of what is

happening in the first 1.7s of the clip. As we can see from figure 13 compared to figure 5, most of the frequencies in the first seconds are almost gone. We can also see how clean and sharp the cut off is as soon as 1.7s ends.

B. Discussion

The program written in MATLAB does filter the unwanted noise from the original audio clip and does produce a clean audio signal. Even though in this paper we have discuss the filter implementation using MATLAB built-in function. A better an improve filter can be apply to the same signal following the same filter parameters obtained from the Fourier Methods. The order used in the filter was a 3rd order but different attempts on different orders. The filter orders that work for this signal are of the orders 2nd, 3rd, and 4th. Anything above a 4th order and the system is not stable anymore and therefore does not meet the Nyquist criteria.

IV. CONCLUSION

The DNR algorithm using Fourier Methods for Spectral Analysis and Digital Filters presented in this paper does work. The goal for this paper is to produce a clean an improve audio from a song from the 70s that carries levels of incrementing hiss. Fourier methods for spectral analysis provided useful information to determine the parameters for a filter that will filter unwanted noise. DNR algorithm can have various applications nowadays, for example television news. Television news could use DSP engineers to filter unwanted noise such as ambient noise, environment, traffic noise etc. Which could lead to an audience watcher with a better and clearer sound resolution for each news given. This paper proves that DNR algorithms improve the sound output of any input audio carrying noise.

(1) REFERENCES

- [1] (n.d.). Retrieved December 03, 2020, from <http://www.ece.northwestern.edu/local-apps/matlabhelp/toolbox/signal/specgram.html>
- [2] *, N. (2015, June 10). Butterworth Filter Design, Equations and Calculations. Retrieved December 12, 2020, from <https://www.elprocus.com/butterworth-filter-formula-and-calculations/>
- [3] Band-pass filter. (2020, December 10). Retrieved December 12, 2020, from https://en.wikipedia.org/wiki/Band-pass_filter
- [4] Bandpass. (n.d.). Retrieved December 03, 2020, from <https://www.mathworks.com/help/signal/ug/practical-introduction-to-digital-filtering.html>
- [5] Brown, G. (2020, November 19). Digital Audio Basics: Sample Rate and Bit Depth. Retrieved December 12, 2020, from <https://www.izotope.com/en/learn/digital-audio-basics-sample-rate-and-bit-depth.html>
- [6] Cerna, M., & Harvey, A. F. (n.d.). The Fundamentals of FFT-Based Signal Analysis and Measurement. Retrieved December 04, 2020, from https://www.sjsu.edu/people/burford.furman/docs/me120/FFT_tutorial_NI.pdf
- [7] Chaudhary, K. (2020, July 30). Understanding Audio data, Fourier Transform, FFT, Spectrogram and Speech Recognition. Retrieved December 12, 2020, from <https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520>
- [8] Data Acquisition. (n.d.). Retrieved December 12, 2020, from <https://www.dataq.com/data-acquisition/general-education-tutorials/fft-fast-fourier-transform-waveform-analysis.html>
- [9] Designfilt. (n.d.). Retrieved December 03, 2020, from <https://www.mathworks.com/help/signal/ug/practical-introduction-to-digital-filter-design.html>
- [10] Digital Signal Processing Reference. (n.d.). Retrieved December 03, 2020, from <http://what-when-how.com/Tutorial/topic-716hi0hnp/Important-Concepts-in-Signal-Processing-Image-Processing-and-Data-Compression-19.html>
- [11] Electrical4U, E. (2020, October 11). Band Pass Filter: What is it? (Circuit, Design & Transfer Function). Retrieved December 12, 2020, from <https://www.electrical4u.com/band-pass-filter/>
- [12] Kauppinen, I., & Roth, K. (n.d.). *Improved Noise Reduction in Audio Signals Using Spectral Resolution Enhancement with Time-Domain Signal Extrapolation* (pp. 100-106, Publication). IEEE. doi:10.1109/TSA.2005.851997
- [13] Kumar M, A., & Chari K, M. (n.d.). Efficient Audio Noise Reduction System Using Butterworth Chebyshev and Ellipticalfilter. Retrieved from https://gvpress.com/journals/IJMUE/vol12_no1/19.pdf
- [14] Largest Archive Of Transparent PNG. (n.d.). Retrieved December 09, 2020, from <https://www.pngkey.com/>
- [15] Low Pass Filter - Passive RC Filter Tutorial. (2020, May 01). Retrieved December 12, 2020, from https://www.electronics-tutorials.ws/filter/filter_2.html
- [16] MathWorks, M. (n.d.). Butter. Retrieved December 09, 2020, from <https://www.mathworks.com/help/signal/ref/butter.html?searchHighlight=butter>
- [17] MathWorks, M. (n.d.). Butterworth Filters. Retrieved December 12, 2020, from https://www.mathworks.com/matlabcentral/fileexchange/38584-butterworth-filters?s_tid=answers_rc2-1_p4_Topic
- [18] MathWorks, M. (n.d.). X. Retrieved December 10, 2020, from <https://www.mathworks.com/help/matlab/ref/fftshift.html?searchHighlight=fftshift>
- [19] MathWorks, M. (n.d.). X. Retrieved December 12, 2020, from <https://www.mathworks.com/help/matlab/ref/fft.html>
- [20] MathWorks, M. (n.d.). X. Retrieved December 12, 2020, from <https://www.mathworks.com/help/signal/ref/pwelch.html>
- [21] Notes, E. (n.d.). FFT Spectrum Analyzer. Retrieved December 12, 2020, from <https://www.electronics-notes.com/articles/test-methods/spectrum-analyzer/fft-fast-fourier-transform-spectrum-analyser.php>
- [22] Prajapati, P. G., & Devani, A. N. (2017). *Review Paper on Noise Reduction Using Different Techniques*. Typescript submitted for publication, Shantilal Shah Engineering college, Bhavnagar Gujarat, India. Retrieved November/December, 2020, from <https://www.irjet.net/archives/V4/i3/IRJET-V4I3145.pdf>
- [23] Songfacts. (n.d.). Song Meanings at Songfacts. Retrieved December 12, 2020, from <https://www.songfacts.com/>
- [24] Stewart, R. (n.d.). Preventing and reducing noise in music production. Retrieved December 12, 2020, from <https://www.justmastering.com/article-preventing-noise-in-music-production.php>
- [25] Understanding Spectrograms. (2019, April 11). Retrieved December 12, 2020, from <https://www.izotope.com/en/learn/understanding-spectrograms.html>
- [26] Upton, E., Veloso, M., Gadgil, A., White, T., Monks, B., & Malkin, R. (2015). Today's Engineering Heroes. *IEEE Spectrum*, 52(3), 39-49. doi:10.1109/mspec.2015.7049439
- [27] User968243user968243 71722 gold badges77 silver badges1212 bronze badges, Ssk08ssk08 67866 silver badges99 bronze badges, & HilmarHilmar 17.1k11 gold badge1313 silver badges2525 bronze badges. (1962, April 01). PSD (Power spectral density) explanation. Retrieved December 12, 2020, from <https://dsp.stackexchange.com/questions/8133/psd-power-spectral-density-explanation>
- [28] What is a Power Spectral Density (PSD)? How is it different than an autopower? (n.d.). Retrieved December 12, 2020, from <https://community.sw.siemens.com/s/article/what-is-a-power-spectral-density-psd>

