Music Analysis and Audio Processing

Saksham Arora
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
sakshamarora.bt19ece@pec.edu.in

Shubham Soni
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
shubhamsoni.bt19ece@pec.edu.in

Diamond Malik
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
diamond.bt19ece@pec.edu.in

Bhupati Bhardwaj
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
bhupatibhardwaj.bt19ece@pec.edu.in

Ashok Singla
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
ashok.bt19ece@pec.edu.in

Shubham Garg
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
shubhamgarg.bt19ele@pec.edu.in

Utkarsh Thatai
Electronics and Communication Engineering
Punjab Engineering College
Chandigarh, India
utkarshthatai.bt19civil@pec.edu.in

***Abstract*—MATLAB is a multi-paradigm programming language primarily used for numerical computations. This paper is primarily focused on applications of MATLAB to audio processing. First, we implement filters, graphic equalizers and give some beautiful sound effects like Echo, Reverb, and Flange. Then we discuss the implementation of a functionality that could extract the notes from a music piece as it is played using concepts of downsampling, averaging filter, moving threshold, and FFTs. Finally, we make a song using its fundamental frequency using a decreasing function for smoothening purposes.**

Keywords—Fourier Transform, White Gaussian noise, Mapping process, SNR, Beta, Moving average technique, Integrators, Filters, Harmonizing, Attenuation, Moving Threshold Technique

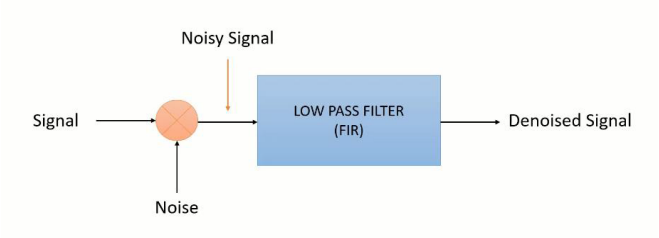
I. Introduction

Audio signal processing is concerned with the electronic manipulation of audio signals and is a subfield of signal processing. Audio signals are an electronic representation of sound waves. Enhancement is one of the critical signal processing applications and includes filtering, equalization, echo, reverb, and flange. In the second part, we downsample a piece of music using an average filter to take out some high-frequency content and decrease the number of samples. The FFT of a lesser number of samples can be calculated faster than the case with a more significant number of samples. Then by moving threshold and taking FFT, we may map the frequencies to given notes, and hence the song can be broken into corresponding notes. Finally, we use fundamental frequency to create a different pitch and then use a decreasing function to make it more smooth.

II. Enhancements Of Audio Signals

A. Filtering

This operation is done on audio signals in order to remove white gaussian noise and hence improve the SNR of the signal. A low pass filter is used, which acts as an integrator and smoothens out the signal, thereby decreasing noise. The main aim is to remove Gaussian noise from a music signal and increase SNR at the output and increase the quality of music. A low pass (FIR) filter is used, which has a finite impulse response (from zero to cut-off frequency). A low pass filter acts as an integrator and hence smoothens the signal. The effect of noise decreased as the waveform of the signal smoothens out. This filter is implemented using designfilt function in MATLAB, which creates a filter object.



B. Equalizer

A graphic equalizer is a sound engineering tool that adjusts the output of different frequencies[1]. It can be an electronic device or computer program that allows the separate control of the strength and quality of frequency bands in an audio signal. Users can cut and boost the levels of specific frequency ranges, providing more granular control of the sound volume.

Graphic equalizers[2] are the simplest type of equalizer, consisting of multiple sliders and graphical controls that can be used to change and control the frequency response of an audio system. They can be used to boost or cut frequency bands of an audio signal and help users shape the audio output and get the most out of the digital music library and speaker setup of audio systems.

The principle of working of a Graphic equalizer is using a certain number of filters in series as the input signal passes through each filter of a specific frequency. Changing the slider positions can increase or decrease the frequency components of the signal. The vertical position of each slider indicates the applied gain in the frequency band, so the

sliders look like a graph showing the response of the equalizer in relation to its frequency.

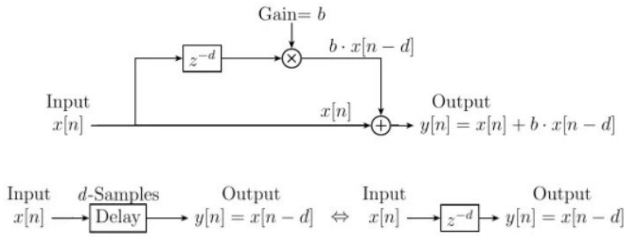
The number of control elements of a graphic equalizer depends on the number of fixed frequencies with which it is to work and the number of frequency channels in the equalizer. It depends on your intended use. A typical five-band graphic equalizer has sliders for five fixed frequency bands, namely:

- Deep bass (30 Hz)
- medium bass (100 Hz)
- Medium range (1 kHz)
- Medium-high range (10 kHz)
- Treble (20 kHz)

Each control or slider acts as a filter and allows you to vary the frequency range passed through the speakers. It also helps improve audio details by boosting high frequencies and reducing distortion and noise in the audio output signal. It is also helpful in adapting sound systems to the different acoustic properties of different rooms.

C. Echo

Echo effects are based on delaying a signal over time. In this case, listeners hear a repetition of the audio signal after some duration of time. Listeners are able to perceive distinct echoes only when the time delay is relatively long.
We will Create it by using a parallel delay path. Then, the unprocessed path is summed together with the delayed path. This type of echo creation is based on feed-forward methods. We may also create echo using feed-backward methods, but we choose feed-forward for our project purposes.



NEED FOR HARMONIZING THE DELAY

In order to make an excellent musical piece with an echo effect, we need to sync the rhythm of the song with the delay introduced while creating echo in the MATLAB Program. Also, we need to take care of the time signature of our chosen song.
Time Signature [3]: - It is basically a notational convention in the world of western music which contains the information as to how many beats are present in each measure. E.g., 2/4 time signature, which is most commonly used, contains 2 beats in one measure, where each beat is a one-quarter note (1/4).
Also, we would need the information of Beats per minute(bps) of the given song in order to synchronize it with the samples of delay we would be using in our improved echo production program.
So, we came up with the following formula: -

Number of samples to be delayed=(secondperbeat*Fs) *beta

Fs= sampling frequency
Here we define ‘beta’=notefactor
Secondperbeat=1/(beatpersecond)=60/(beatpermin)

The above formula considers the number of seconds per beat and also the note duration. We then multiply with the sampling frequency (Fs) to get it in terms of the number of samples.
Assuming a time signature of 4/4, beta values for different notes would be: -
- for a quarter note: - beta=1
- for a half note: - beta=2
- for whole note: - beta=4

As the time signature of our chosen song, "shape of you” is 4/4 type, we would only need quarter notes,
So, we take beta=1 for our project purposes.

So, we have successfully optimized the number of samples to be delayed. We should carefully round off the number of samples so that we can use it further for array indexing. This delay would produce an echo according to the rhythm of the song rather than just a distorted and delayed audio.

D. Reverb

Reverb occurs when a sound hits any hard surface [4] and reflects back to the listener at varying times and amplitudes to create a complex echo & the time between successive reflections is less than 0.1 seconds. Therefore, we cannot differentiate between reflection sound and produced sound, and we observe one persistent sound that is the reverb effect. It results from a large number of reflected waves, which can be implemented in MATLAB [5] by mixing input with the delayed value of the input with decreasing amplitudes.

E. Flange

Flange is an audio consequence produced with the aid of blending equal signals together. One signal is delayed using a small and cautiously changing period, generally smaller than 20 milliseconds. This produces a swept combfilter effect: peaks and notches are produced in the emerging frequency spectrum, associated with each other in a linear harmonic series. Varying the time delay results those to sweep up and down the frequency spectrum. This variable delay is created using a low-frequency sine wave. Delay varies as the shape of a sine wave. The delay is passed through attenuation before mixing with the original signal.

The input-output relation for a fundamental flanger may be written as
Y[n]=x[n] +g.x[n – d[n]]
in which x[n] is the input signal amplitude at n=0,1, 2..., y[n], is the output, g is the "depth"' of the flanging effect, and d[n] is the duration of the delay line at time n. The delay length d[n] is typically varied according to a triangular or sinusoidal waveform. To reap the greatest effect, the depth control, g, must be set to 1. A depth of g=0 gives no effect.

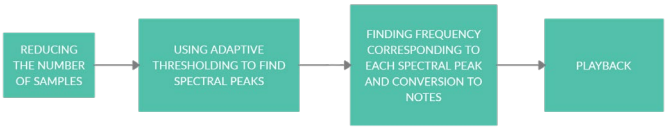
The flanging result has been delineated as a sort of “whoosh” passing gently through the sound. The effect is additionally compared to the sound of a jet passing overhead, during which the direct signal and ground reflection make a varied relative delay.

III. Detecting Frequencies In A Musical Piece And Converting It To Notes

The purpose is to map the frequencies of a given song to musical notes (like C, D, E, or Sa Re Ga Ma, etc.). For this, we needed to devise a method to break the song in a way as to get specific chunks of the complete song. If the song is broken in any random way, we cannot decode it optimally, and we cannot map the song to its notes correctly. This problem was solved by a lot of trial and error and some musical knowledge. After devising optimum methods, it was possible to convert them into corresponding frequencies using Fast Fourier Methods (FFT). Finally, we could get musical notes from the found frequency.

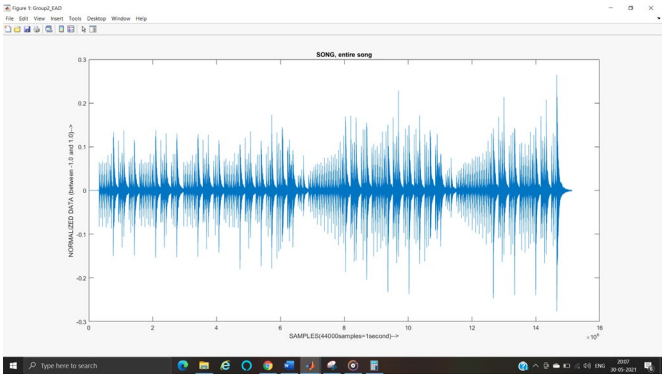
In working, we first proceed with the general steps of sampling our sound file and extracting the various amplitudes in a vector, simultaneously storing its sampling frequency(Fs). After that, analyzing window of the chosen song was taken, and the general business of defining MATLAB variables, storing specific values, etc., was dealt with.

The entire process would be implemented in the following order:

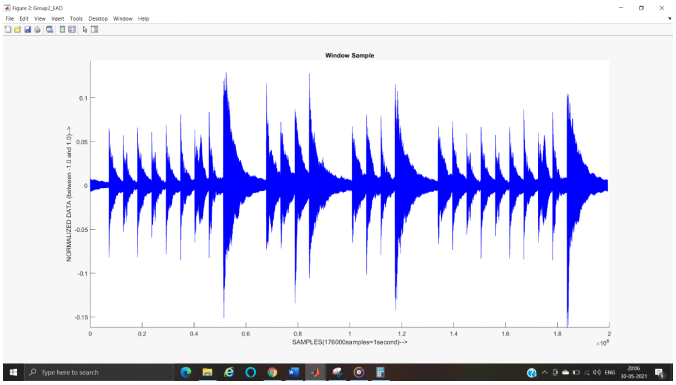


A. *Reducing the number of Samples*

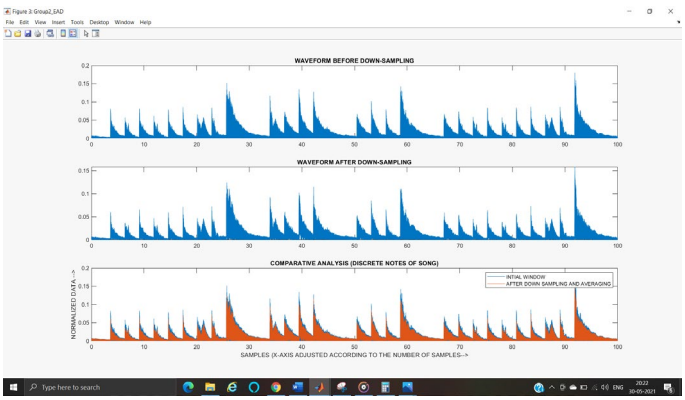
Now, for finding the musical notes, we primarily need to find the frequencies of the given audio signal. For finding the frequencies, we resorted to Fast Fourier Transform(FFT), which uses a high-speed computational algorithm to find the Discrete Fourier Transform of a signal. In the small window of the song, we considered, we had 20,00,000 samples, which would have been too much for FFT calculations and would have slowed our entire program. So, for reducing the number of samples, we down-sampled by a factor of 20 using moving average techniques. So, finally, we successfully reduced the number of samples to 1,00,000 while still retaining the shape and characteristics of our original signal to a reasonable extent.



Original song (Sampled at 44000 samples/sec)



A window of the song(Sampled at 1,76,000 samples/sec)

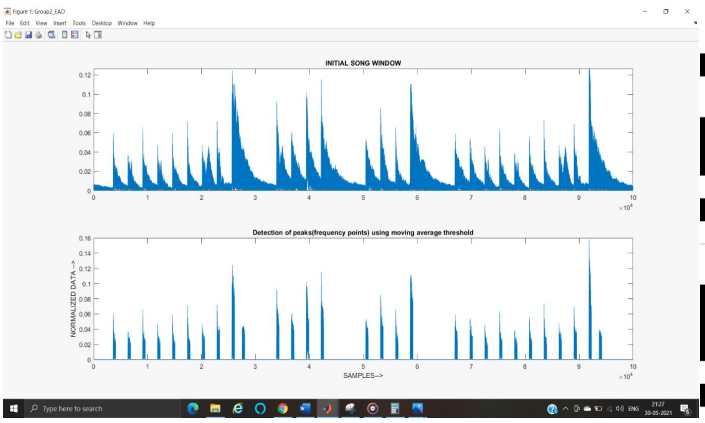


Waveforms after downsampling

B. *Need for Threshold*

Now, for finding distinct notes in a given song, we need to store different areas of our waveform which are very prominent as compared to others because those are the places that contain the frequency at which the note was played [6], other areas are just continuation or transition from one note to other. However, now we also have the possibility that some other areas may also contain some significantly less prominent peaks that we may miss if we use a constant threshold for all samples, so we thought of using a moving threshold technique so that the threshold defined is according to the values of samples in close vicinity. Hence, the peaks will always be detected.

We observe that our song has about 30-40 prominent peaks, which means that on an average, we may have a peak in every 2000 samples (Total Samples/50), which also means that each note also approximately 10ms, as mentioned by the authors of [7], so we define a moving threshold which accounts for a statistical value of 5000 samples in the vicinity. Once the sample data value exceeds the threshold, we say the prominent peak is found. We store some values corresponding to the peak and then make constant jumps of samples (to save calculations), considering that we have a peak in every 2000 samples approximately.



Detection of peaks using moving threshold

C. *Find the frequency of each note*

Having detected the notes from the threshold, we now find the frequencies corresponding to that notes. We traverse through the significant notes and compute their Discrete Fourier Transform. A total of 29 notes were detected from the song, which had their own fundamental frequencies. The notes which had one peak in their frequency domain were more cleaner than the ones which had more than one peak.

	A	B	C	D	E	F	G	H
1	1	Eb4	9	G3	17	A3	25	A3
2	2	D4	10	G2	18	Bb3	26	C4
3	3	Eb4	11	Bb2	19	D3	27	Bb3
4	4	D4	12	D3	20	D4	28	G3
5	5	Eb4	13	G3	21	D4	29	G2
6	6	A3	14	A3	22	Eb4		
7	7	C4	15	D3	23	D4		
8	8	Bb3	16	Gb3	24	Eb4		

D. *Playback*

This was the last step. Herein we play the notes we got after the complete mapping process. The nodes are played in such an order that they roughly re-create the original song.

After playing the section, it can be inferred that though there was a difference between the original song and the re-created song, a close resemblance still exists between them.

IV. Making A Rhythm From Given Frequency Notations

With the help of MATLAB, we had tried to make a song or a melody. We generated different pitches with the help of fundamental frequencies [8], and by combining them, we constituted a tune. By smoothing that tune with the help of decreasing function, we got a melody.

So basically it can be done in two steps :

• Generating different types of sound waves with the help of the fundamental frequency concept of piano keys.

• Alter the amplitude of the generated waves according to the melody we want to create.

56	e″	E_5	659.2551
55	d♯″/e♭″	$D♯_5/E♭_5$	622.2540
54	d″	D_5	587.3295
53	c♯″/d♭″	$C♯_5/D♭_5$	554.3653
52	c″ 2-line octave	C_5 Tenor C	523.2511
51	b′	B_4	493.8833
50	a♯′/b♭′	$A♯_4/B♭_4$	466.1638
49	a′	A_4 A440	**440.0000**
48	g♯′/a♭′	$G♯_4/A♭_4$	415.3047
47	g′	G_4	391.9954
46	f♯′/g♭′	$F♯_4/G♭_4$	369.9944
45	f′	F_4	349.2282
44	e′	E_4	329.6276
43	d♯′/e♭′	$D♯_4/E♭_4$	311.1270
42	d′	D_4	293.6648
41	c♯′/d♭′	$C♯_4/D♭_4$	277.1826
40	c′ 1-line octave	C_4 Middle C	261.6256

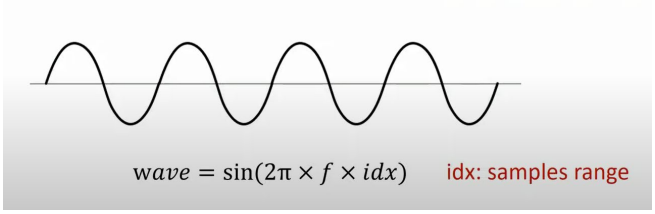
If we want to generate a sound of any particular key of the piano, it can be done with the help of an index known as 'Key Number.' There is a particular key number that corresponds to every key. Furthermore, with the help of that key number, we can get the fundamental frequency corresponding to that key.

A. *Generation of Sound Wave*

There is a relation between the above-mentioned index(or key number) and the corresponding frequency of any particular key[9]:-

$$n = 12log_2\left(\frac{f}{440\ Hz}\right) + 49$$

With the help of this fundamental frequency, we can generate a sound wave of a particular pitch.



Wave = sin (2π × f × idx)
where idx: samples range

E.g., To generate a sound corresponds to E4, so we just give the key index, and it will generate the wave.

44	e′	E_4	329.6276

However, here we also have to provide the duration of the sound, i.e., for how long that particular pitch should sound. So here we take two types of durations :

• Full cycle, which is equal to one second.
• Half cycle, which is equal to half a second.

When we create a wave with the help of the above-mentioned method, the generated wave has constant amplitude throughout. Sound with constant amplitude is not so pleasant to hear.

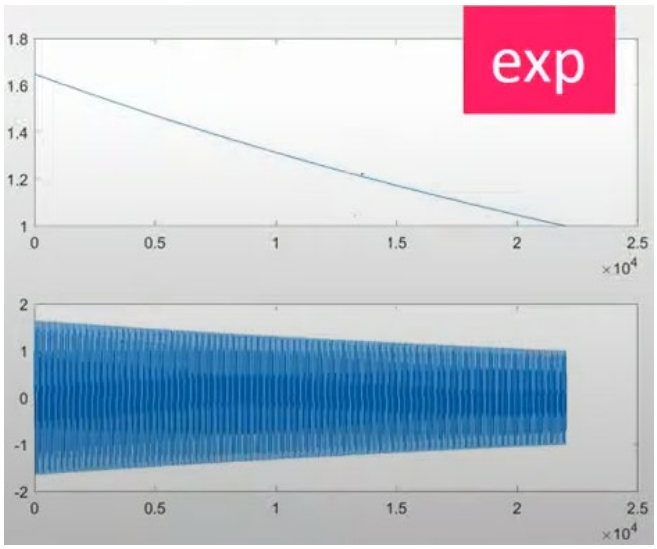
To be called as melody, an audio should be altered in two manners :

• From strong to weak.
• From weak to strong, then from strong to weak.

The method which you should use mainly depends on what type of audio you are altering and what type of melody do you want.

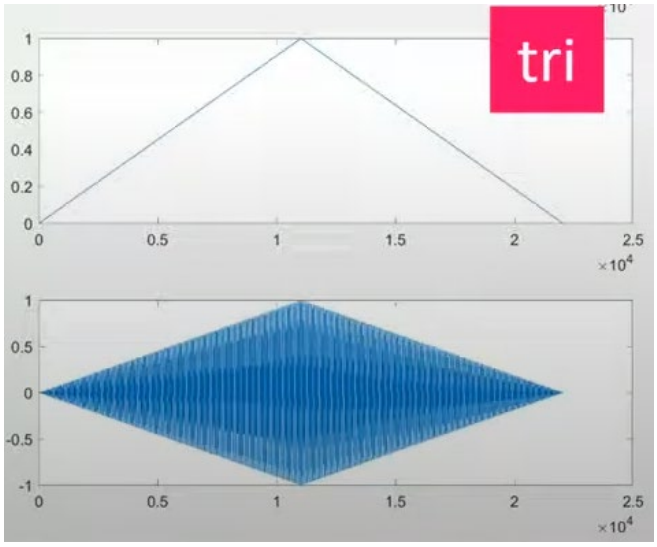
B. Exponential Decreasing

In this method, we multiply an exponential function with our generated wave to make its amplitude continuously decreasing. It sounds good if we apply it in a group of lines of song or melody.



C. Triangular Decreasing

If we multiply a triangular function with our generated sound wave, we get an output like mentioned above. In this, firstly, the amplitude will increase with a constant rate and then decrease after half cycle with the same rate.



ACKNOWLEDGMENT

We thank respected Monoj Kumar Singha Sir for his constant support and encouragement throughout the project period. In fact, we were also helped by him to choose the correct topic according to our interests.

REFERENCES

[1] TechTerms “EQUALIZER”: - https://techterms.com/definition/equalizer

[2] Techopedia “Graphic Equalizer”: - https://www.techopedia.com/definition/15886/graphic-equalizer

[3] Michele Aichele “Musical Time-Signatures” Understanding Time Signatures and Meters: A Musical Guide (libertyparkmusic.com)

[4] Daniel Brooks “what does a reverb-effect do? -The Basics”: - https://reverb.com/news/what-does-a-reverb-effect-do-the-basics

[5] “Reverberator”: - Add reverberation to audio signal - MATLAB - MathWorks India

[6] Rui Pedro Paiva , Teresa Mendes , Amílcar Cardoso “A Methodology For Detection of Polyphonic Musical Signals” University of Coimbra, Coimbra, Portugal

[7] Prashanth T R, Radhika Venugopalan “Note Identification In Carnatic Music From Frequency Spectrum” National Institute of Technology Trichy, Tamil Nadu, India.

[8] Caleb J.Murphy “what-fundamental-frequencies-have-to-do-with-recording-a-great-song” :- https://blog.sonicbids.com/what-fundamental-frequencies-have-to-do-with-recording-a-great-song

[9] “Piano Key Frequencies”: - https://en.wikipedia.org/wiki/Piano_key_frequencies