Audio Effect Processor

Digital Signal Processing

1st Naimur Rahman
Id: 021 221 043
Department Of EEE
United International University
Dhaka, Bangladesh
nrahman221043@bseee.uiu.ac.bd

2nd Md. Nayon Khan
Id: 021 221 045
Department Of EEE
United International University
Dhaka, Bangladesh
mkhan221045@bseee.uiu.ac.bd

3rd Suvom Karmakar
Id: 021 221 027
Department Of EEE
United International University
Dhaka, Bangladesh
skarmakar221027@bseeee.uiu.ac.bd

***Abstract*—This project report documents the design and implementation of an Audio Effects Processor using MATLAB, focusing on providing real-time manipulation of audio signals through various digital signal processing (DSP) techniques. The system enables the application of multiple audio effects like gain, echo, reverb, distortion, speed adjustment, and more. This report covers the architecture, design methodology, user interface development, and performance of our processor, providing a comprehensive guide for anyone interested in understanding or replicating this project.**

***Index Terms*—DSP, Signal, Effects**

I. Introduction

In the era of digital sound manipulation, there is an ever-growing need for flexible and effective tools that allow users to modify audio in creative ways. This project is aimed at designing and implementing a MATLAB-based Audio Effects Processor, which offers a variety of effects ranging from simple gain adjustments to more complex effects such as reverb and chorus. The project is designed to demonstrate the practical applications of DSP techniques, enabling users to manipulate audio easily through an User Interface by MATLAB App Designer.

II. Components

- **Computer:** A personal computer for developing and running the processor.
- **MATLAB Software:** The main development environment used for implementing the effects and building the GUI.

III. Methodology

The Audio Effects Processor implements the following effects. Each effect is implemented using Digital Signal Preocessing(DSP) techniques that manipulate the audio waveform in distinct ways.

IV. User Interface

We developed a user-friendly interface using MATLAB's App Designer, enabling users to interact with the processor without needing to understand the underlying code. This GUI significantly improves the accessibility of our system, catering to both experienced users and novices in DSP. We made two interfaces:

- User Interface for all Effects.
- User Interface for Reverse Effects.

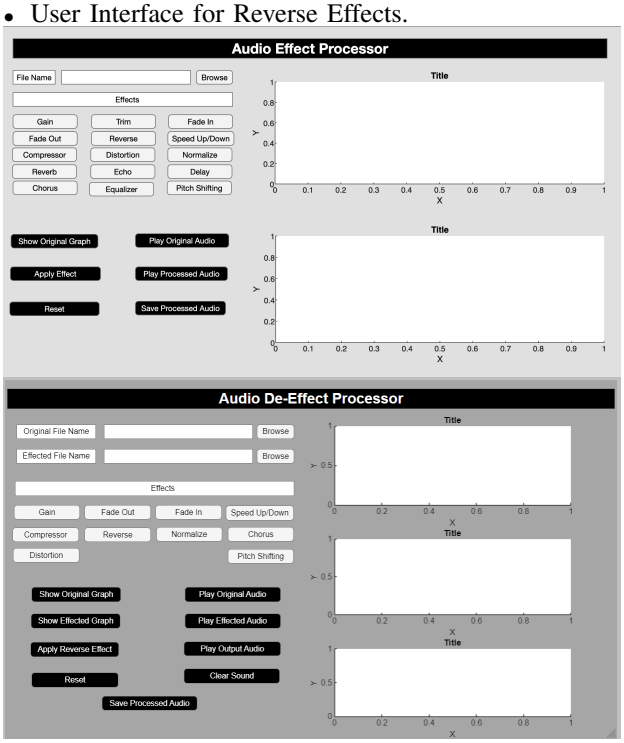


Fig.34: Two Interfaces.

A. Buttons in the Interface

- Browse button for file choose
- All available effect buttons
- Button to apply effect
- Play different audios
- Sound clear button
- Reset button
- Save button

B. Features

- User can choose any file by browsing
- Can see both original and effected graph
- Can check if wanted to that reverse graph matched with original
- Can save the audio
- Can apply multiple effect over one audio

V. Basic Effects

A. Gain

The Gain effect changes the volume of the audio. Anyone can make it louder or quieter by adjusting the gain. It works by multiplying the audio signal by a certain factor—higher values make the sound louder, and values¡1 make it quieter.

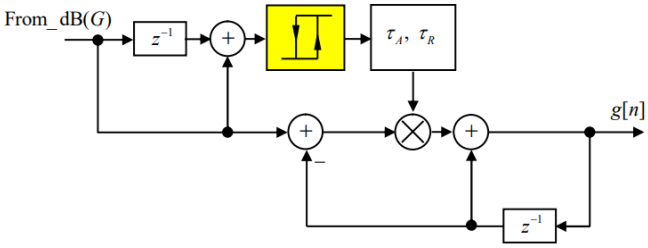


Fig.01:Gain Control Block Diagram

1) Implementation: The audio signal is multiplied by a user-specified gain factor to increase or decrease volume.

2) Input and output Analysis: We gave a limit of 0 to 4 times for gain. Below one for lowering the sound and more than one to boost the sound. We had keep the limit because more than 4 times gain can cause distortion in audio. Below a sample of input and output is given where first we gave input 2x boost and then we gave 0.5x. We find out our gaining effect is perfectly working from amplitude analysis of the signal.

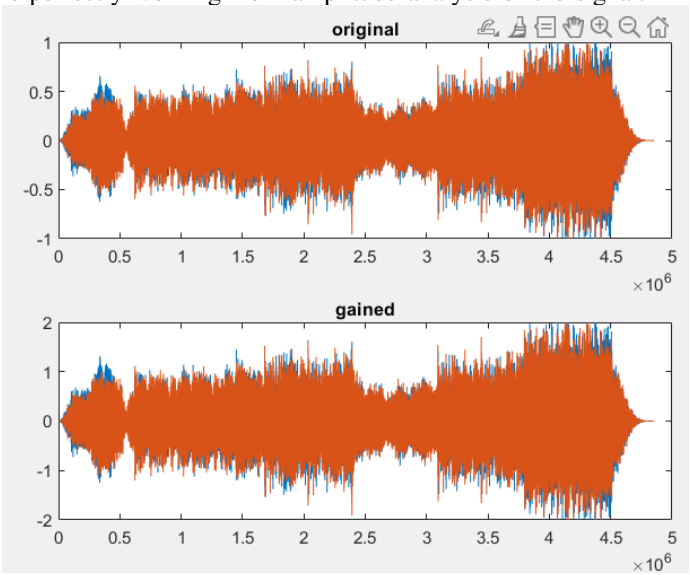


Fig.02:Gain of 2x

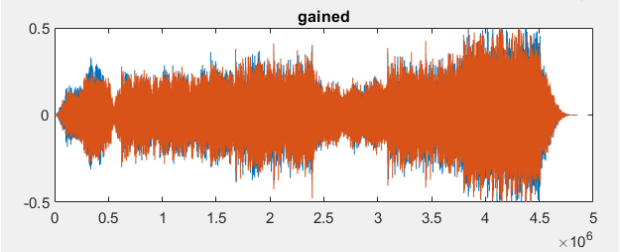


Fig.03:Gain of 0.5x

B. Trimming

Trimming means cutting a certain section of an audio, anywhere , any section of the audio according to users preference.

1) Implementation: From the user input, from starting time to end time, we find out the total sample and audio write that portion in a new signal variable.

2) Input and output Analysis: We make a limit in input like if user put end point time second larger than audio signal than we had convert it to the end time. Here in this sample we choose to test it by trimming 3 to 6 sec portion where from graph we got to know that it is working. X-axis difference is because original and trimmed one respectively in $10^5 and 10^4$.

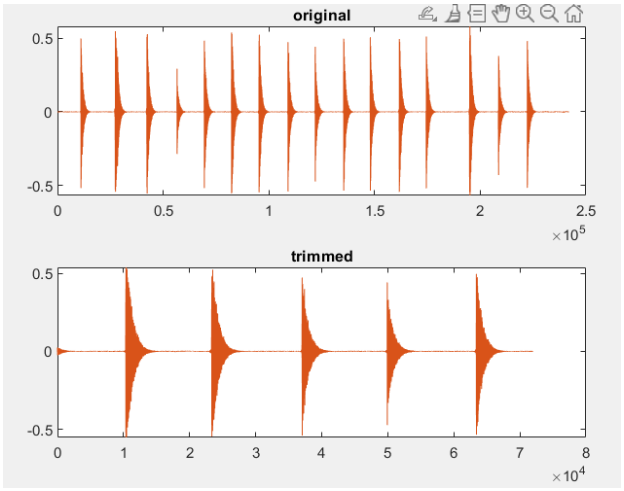


Fig.04:Trimmed 3 to 6 sec.

C. Fade In

The Fade In effect gradually increases the volume of the audio upto the point user wanted and finally reach the maximum level and stay in max. The effect doesn't have any extra gaining. So it starts soft and then becomes louder. This is useful for making smooth transitions into a track.

1) Implementation: A linear ramp is applied to the initial part of the audio signal, gradually increasing the amplitude to maximum.

2) Input and output Analysis: Here user can choose how much percentage of the audio will be affected by fade-in effect. For example, here we choose 70 percent of our audio would be faded in.

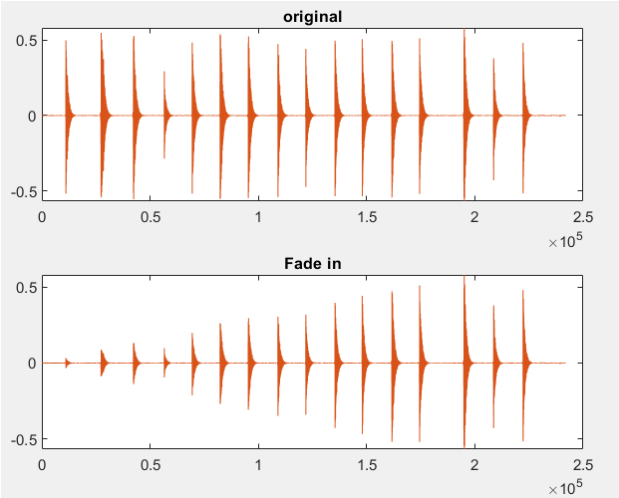


Fig.05: Fade-in.

D. Fade Out

The Fade Out effect does the opposite of fade-in. It gradually decreases the volume at the end of the audio, so it fades into silence. This creates a smooth ending.

1) Implementation: A linear ramp is applied to the end of the audio signal, reducing the amplitude to zero.

2) Input and output Analysis: Here user can choose how much percentage of the audio will be affected by fade-out effect. For example, here we choose 90 percent of our audio would be faded out.

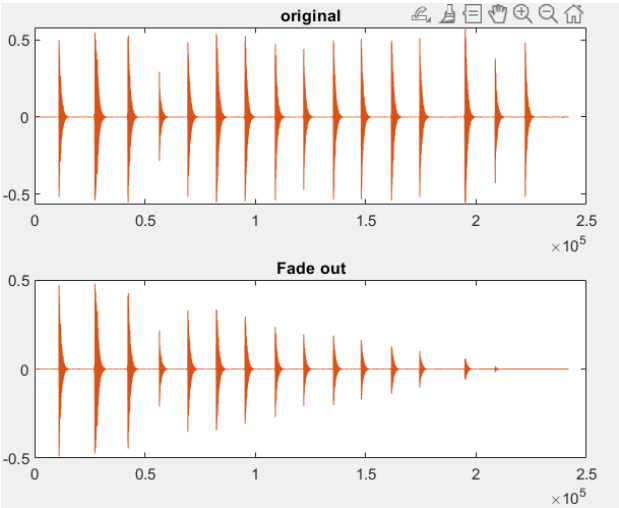


Fig.06: Fade-out.

E. Reverse

The Reverse effect plays the audio backward, starting from the end and moving toward the beginning.

1) Implementation: A matlab function ”flipud” used to flip the audio and play from the backward.

2) Input and output Analysis: Audio will be reversed like one to ten counting audio will be reversed, ten to one but ten will be spelled as reversed ”net” and one will be ”eno”. Here is the sample output.

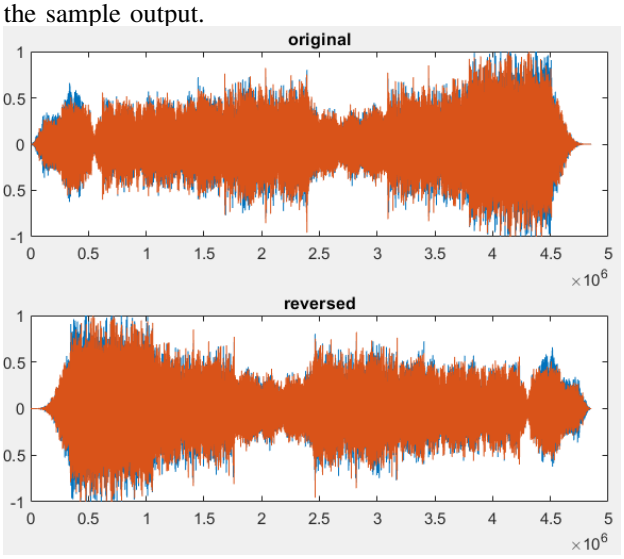


Fig.07: Reverse.

F. Speed Up

The Speed Up effect changes the playback speed of the audio. Speeding it up makes it play faster.

1) Implementation: Multiplying frequency is applied to the audio to change its duration, speeding it up without altering pitch.

2) Input and output Analysis: User can chose how much faster they want the audio to be. They can choose between 1 to 4 times. From the graph we can’t analyse this effect. If signal plotted with time it can be inspected.

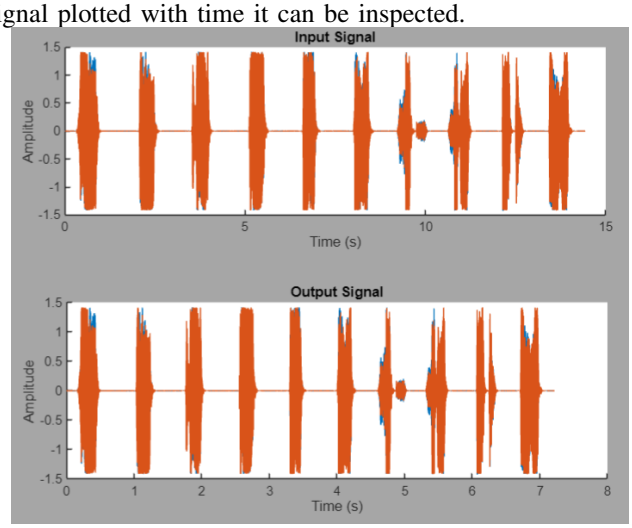


Fig.08:Speed up

VI. Dynamic Effects

A. Compressor

The Compressor reduces the difference between the loud and soft parts of the audio. If a sound gets too loud, the compressor lowers the volume to make it more balanced, so the listener doesn’t have to adjust the volume constantly.

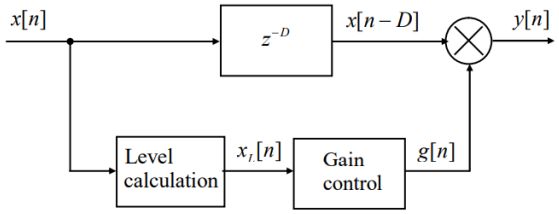


Fig.08:Compressor Block Diagram

1) Implementation: Compression is applied based on a threshold, ratio, makeupgain, attacktime, releasetime reducing the amplitude of loud sections. Threshold is limit, where if sound passes threshold then it will be decrease according to the ratio, like if ration is 5 means after passing threshold every 5dB will be converted to 1dB. Make up gain is additional gain if user wanted to have gain then attacktime is how quickly the compressor reacts and starts reducing the volume (gain reduction) after the audio signal exceeds the threshold and The release time refers to how quickly the compressor stops compressing (returns the gain back to normal) once the signal falls below the threshold.

2) Input and output Analysis: Here as a sample input threshold set to -85dB, attacktime 0.1s,releasetime 0.01s,makeupgain 1 and ratio 5.

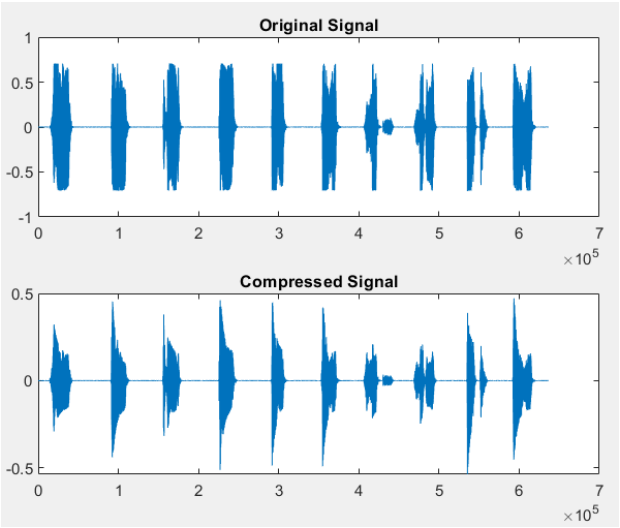


Fig.09: Compressor.

B. Distortion

The Distortion effect changes the audio to make it sound rough or gritty, usually by increasing the signal strength beyond its limits. This can give the sound a fuzzy or overdriven quality, often used in electric guitars.

1) Implementation: The signal is clipped or non-linearly transformed to create harmonic distortion.

2) Input and output Analysis: User can choose amplifying factor to have more distortion. Here in this sample, input of amplifying given 1.

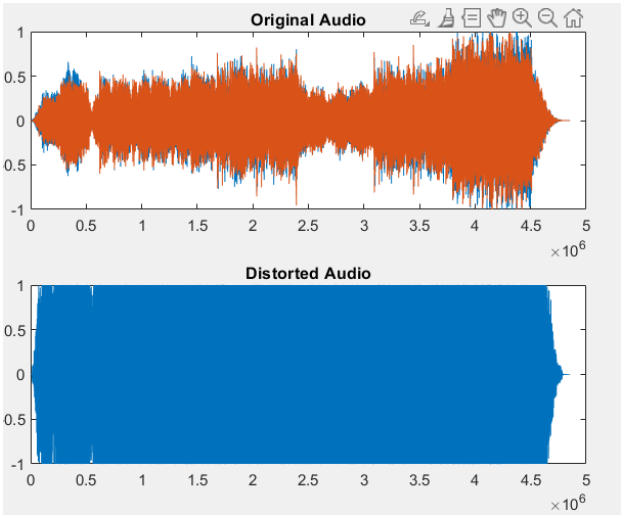


Fig.10: Distortion.

C. Normalize

The Normalize effect adjusts the overall volume so that the loudest part of the audio reaches a target level. It ensures the audio isn't too soft or too loud by scaling the volume to a certain point.

1) Implementation: The signal is scaled proportionally to reach the desired peak value.

2) Input and output Analysis: User can choose the peak of the signal. The normalized signal basically scaled at amplitude of1 then multiply with user gain. Here is an example output with peak 1 chosen.

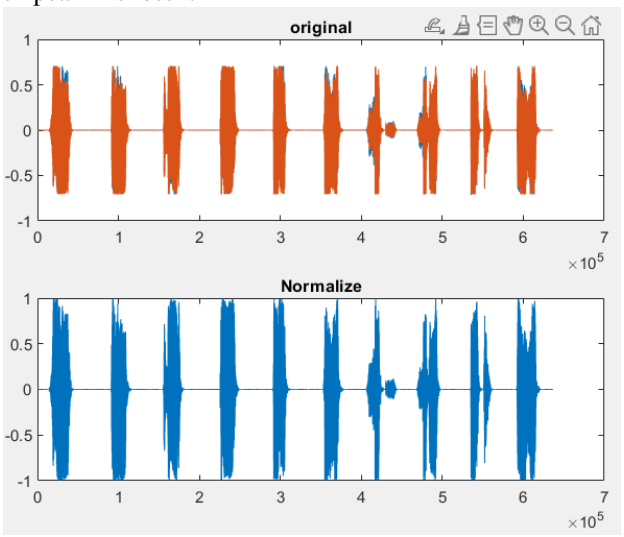


Fig.11: Normalize.

VII. Time Based Effects

A. Reverb

The Reverb effect simulates how sound behaves in a space like a room or hall. It adds a sense of space and depth to the audio by creating small echoes, making the sound feel more "live" or distant.

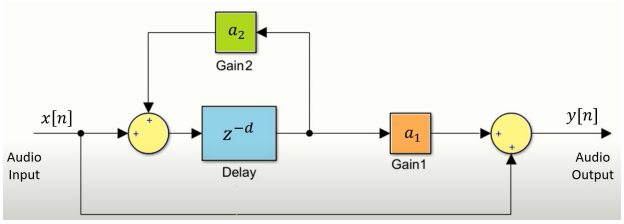


Fig.12: Reverb Block.

1) Implementation: We used a MATLAB built-in function which convolve the audio signal with an impulse response, simulating the reverberation of a specific environment. The function take sample rate of the signal, Pre- delay, wet, high frequency damping factor. We fixed all of the factor for reverb and let the user to handle wet mixing. Here, Pre-delay is the amount of time between the original sound (the dry signal) and when the reverb effect begins. It's essentially a delay before the reverb kicks in and wet means the percentage of signal affected by reverb effect and high-frequency damping refers to how the higher frequencies in the reverb decay compared to lower frequencies.

2) Input and output Analysis: User can chose wet mixing percentage fo reverb effect. For example, in the sample her we choose wet to be 100 percent.

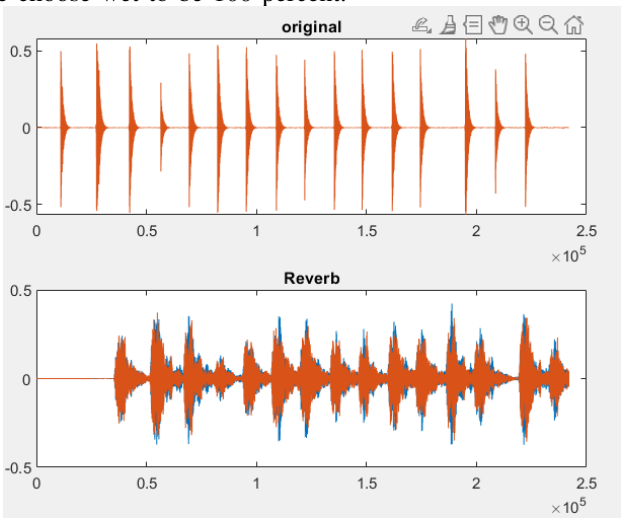


Fig.13: Reverb.

B. Echo

The Echo effect is like delay but with multiple repetitions. Each time the sound repeats, it becomes softer and softer, giving the impression of sound bouncing off walls, like in a canyon or large room.

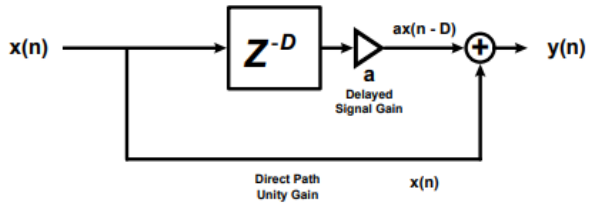


Fig.14: Echo Block.

1) Implementation: We created echo by delaying the original signal and adding it to itself with reduced amplitude. This is done by appending zeros to the signal, shifting it, scaling it, and adding it back to the original signal by taking delay time(0-0.5)s input from user.

2) Input and output Analysis: Here as a sample we choose 0.1 as delay time.

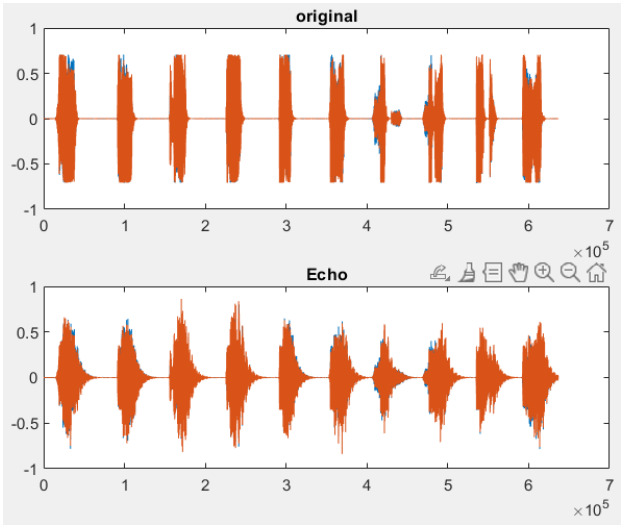


Fig.15: Echo.

C. Delay

The Delay effect adds a copy of the audio that plays after a short pause. This delayed copy is mixed with the original sound, making it sound like there's an echo or repetition.

1) Implementation: Echo and delay effect are quite similar but in delay their is more delay time than echo and no attenuation in echoes.

2) Input and output Analysis: Here we take delay time 1 to 4 s as user input. In the sample we tested with delay time of 2s.

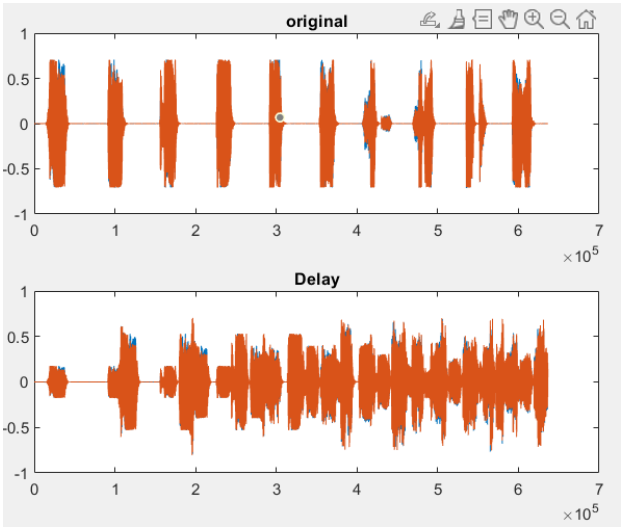


Fig.16: Delay.

VIII. Spectrum Processing Effects

A. Chorus

The Chorus effect creates a richer, fuller sound by layering the original audio with slightly delayed and pitch-shifted versions of itself. This makes it sound like multiple people or instruments are playing the same part.

1) Implementation: The chorus effect created by duplicating the original signal, applying small, varying delays to each copy, and mixing them back with the original. This simulates the effect of multiple voices or instruments playing the same note with slight timing variations.

2) Input and output Analysis: Here user can give input delay time in seconds 0 to 0.3 and a feedback gain 0 to 1. That means delayed multiples of original sound gain would be same or less according to user needs. Here, in example output we choose delay time as 0.1 and feedback gain of 1.

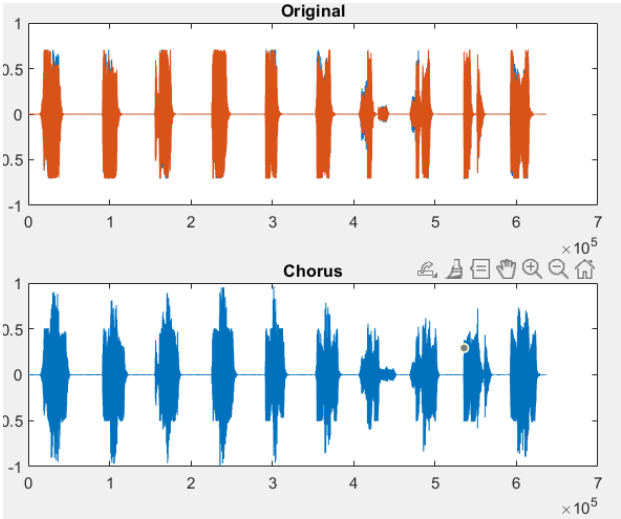


Fig.17: Chorus.

B. Equalization

The Equalization (EQ) effect adjusts the balance of different frequency ranges in the audio. We can boost or reduce the bass, mids, or treble to change the tone of the sound, similar to adjusting the ”bass” and ”treble” on a stereo.

1) Implementation: A 9-band equalizer is implemented using filters to control low, mid, and high-frequency content. We keep both IIR and FIR filters and we choose 1st band 85 Hz, 2nd band 280 Hz, 3rd band 450 Hz, 4th band 1kHz, 5th band 3 KHz, 6th band 6 KHz, 7th band 10 KHz, 8th band 13 KHz and 9th band 16 KHz. Two basic effects were given built in, they are ’Old Radio’ and Lo-Fi effect.

2) Input and output Analysis: User can choose gain for every bands in dB. Here 3 output sample shown for random input of Gain for 85 Hz: -20, Gain for 280 Hz: -16, Gain for 450 Hz: -14, Gain for 1000 Hz: -8, Gain for 3 kHz: -4, Gain for 6 kHz: 0, Gain for 10 kHz: 4, Gain for 13 kHz: 10, Gain for 16 kHz: 12, filter type IIR and 2nd output for old radio effect and 3rd one for Lo-Fi.

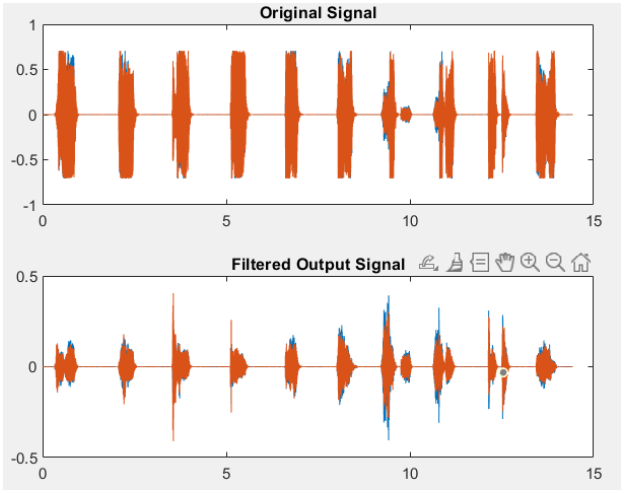


Fig.18: Random input equalizer.

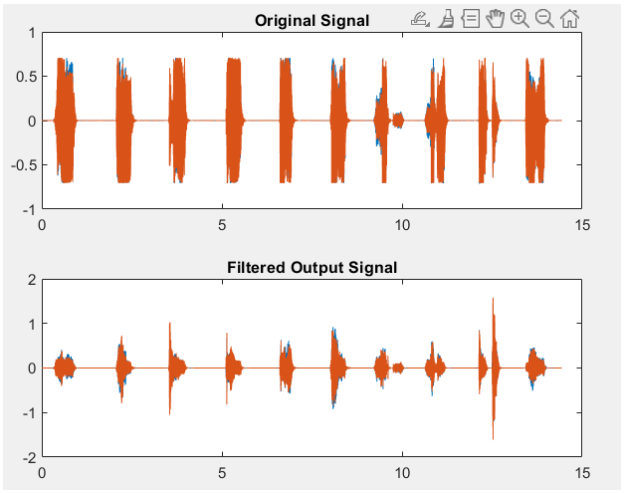


Fig.19: Old radio equalizer.

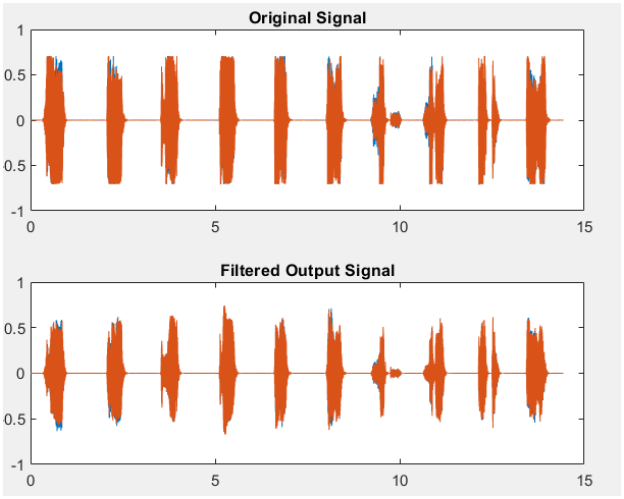


Fig.20: Lofi equalizer.

C. Pitch Shifting

Pitch shifting changes the pitch of the audio without affecting the speed can create deep to shallow voices by changes in voice frequency caused by shift of pitch.

1) Implementation: We pitch shifting achieve by resampling the signal at a different rate, effectively changing its pitch. We have used matlab built-in function shiftpitch to resample and shift the pitch.

2) Input and output Analysis: User can choose semitones pitch shifting parameter from -8 to 8. More positive means higher frequency voice and negative means lower frequency deep voice. Here we chose two outputs taking semitones of -5 and +5 and we can analyse pitch shifting from graph by look over x axis.

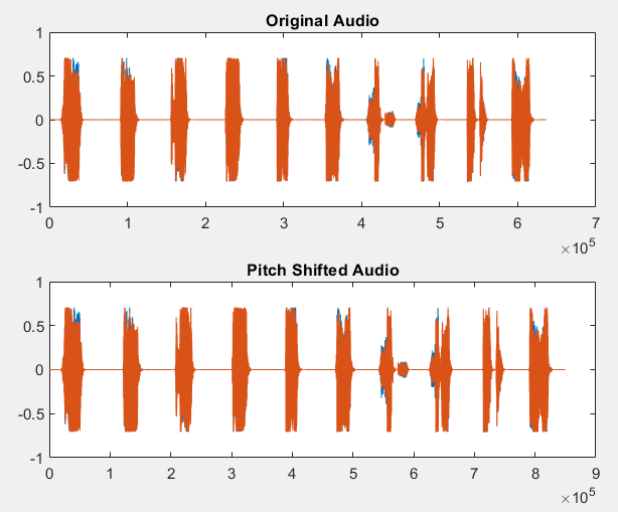


Fig.21: Pitch Shift(-5).

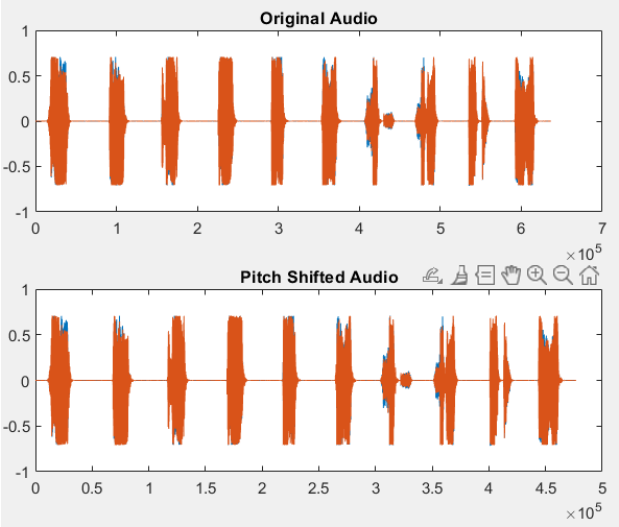


Fig.22: Pitch Shift(+5).

IX. Reverse Basic Effect

A. Reverse Gain

The Reverse Gain effect changes the volume of the audio. Anyone quieter by adjusting the gain. It works by multiflying the audio signal by a certain factor—higher values make the sound quiter.

1) Implementation: The audio signal is multiplied by a user-specified gain factor to increase or decrease volume.

2) Input and output Analysis: We gave a limit of 0 to 1 times for gain. Any value lower 1 will decrease the gain. Here example of 0.5x gain.

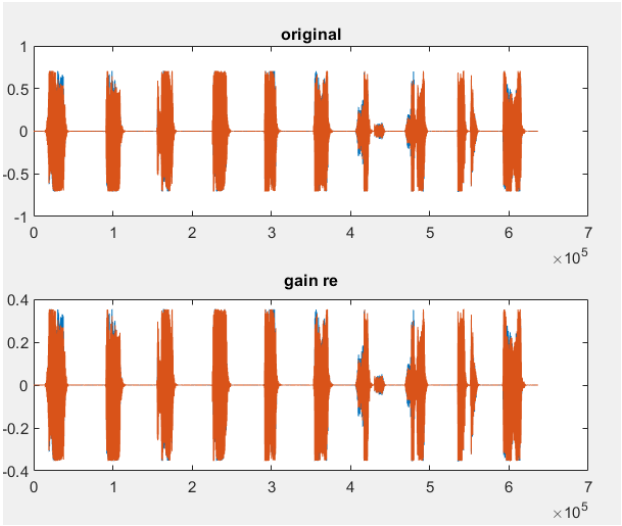


Fig.23:Gain of 0.5x

B. Reverse Trimming

Reverse trimming isn't possible due to loss of data or signal. Trimmed signal can't be reversed until the signal trimmed here want to undo , otherwise not possible.

C. Reverse Fade-in

Reverse fade-in also approximately possible and can't back the exact original signal.

1) Implementation: Same as Fade in implementation, just here user give the percentage of faded in, approximately or when the used fade in previously, then we just amplify that percent of signal.

2) Input and output Analysis: Here for sample output, we gave 70 percent faded in signal. s

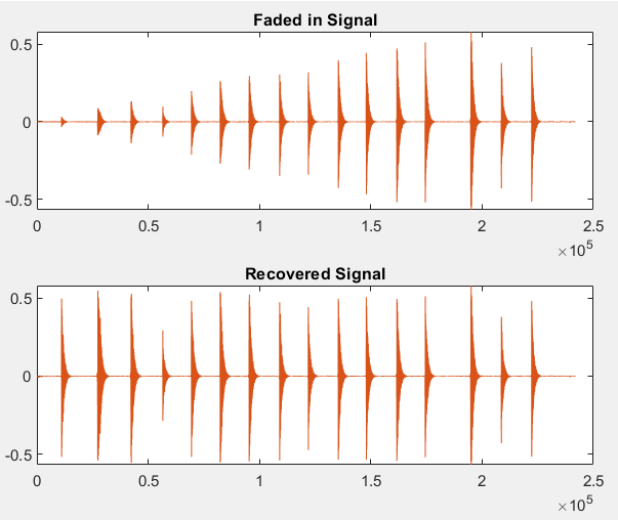


Fig.24:Recovered original

D. Reverse Fade-out

Reverse fade-out also approximately possible and can't back the exact original signal.

1) Implementation: Same as Fade in implementation, just here user give the percentage of faded out, approximately or when the used fade out previously, then we just amplify that percent of signal.

2) Input and output Analysis: Here for sample output, we gave 90 percent faded out signal.

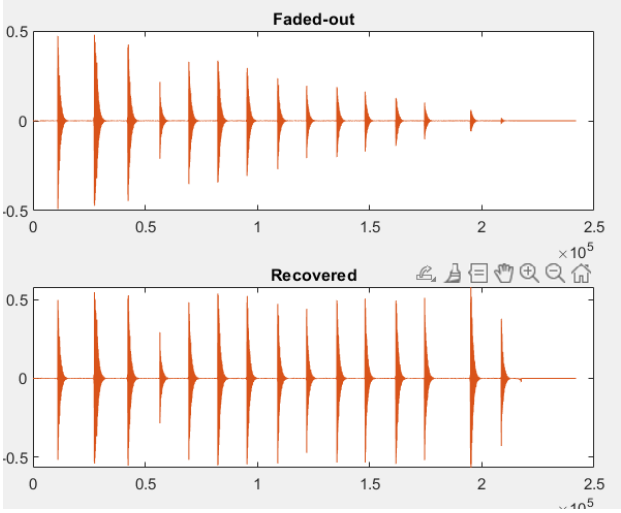


Fig.25:Recovered original

E. Reverse of Reversed Signal

The Reverse effect plays the audio backward, starting from the end and moving toward the beginning. Reverse it now back to original.

1) Implementation: A MATLAB function "flipud" used to flip the audio and play from the backward means original.

2) Input and output Analysis: Audio will be reversed, so we will find the original.

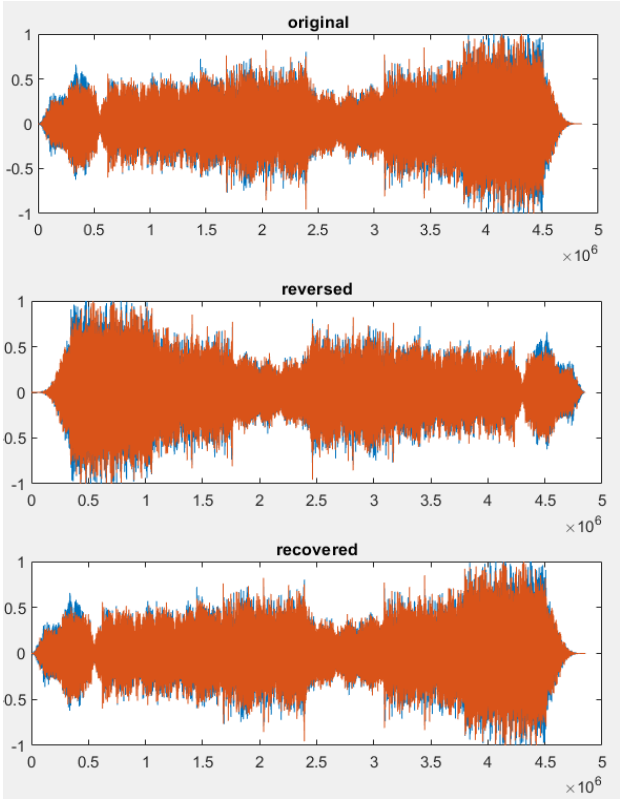


Fig.26:Recovered signal

F. Slows Down

The slows down(reverse speed) effect changes the playback speed of the audio. Slowing down makes it play slower.

1) Implementation: divide in frequency is applied to the audio to change its duration, slows it down without altering pitch.

2) Input and output Analysis: User can chose how much slower they want the audio to be. They can choose between 1 to 4 times. From the graph we can't analyse this effect. If signal plotted with time it can be inspected.

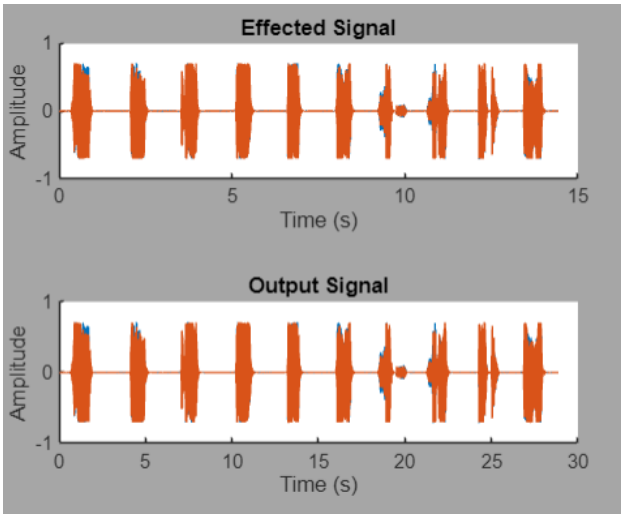


Fig.27: slows down

X. Reverse Dynamic Effect

A. Reverse Compressor

The Reverse compressor is just opposite of compressor.

1) Implementation: By doing reverse code engineering we can find the original signal.

2) Input and output Analysis: Here, user have to give the inputs they used for compressing. Here, we take input as previously compressed output signal.

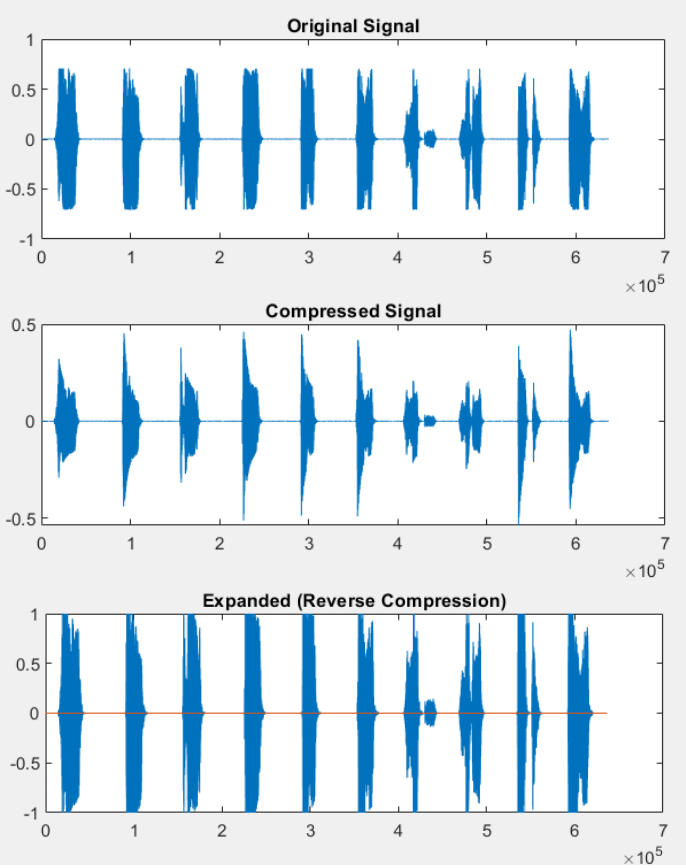


Fig.29: Recovered Signal.

B. Reverse Normalize

For reverse normalize just amplifying the signal is enough. User can use the gain effect.

1) Input and output Analysis: Taking Previous normalized output as an input. In that signal original peak was 0.75. So, here input will be 0.75 peak. However, perfect recovery may not be possible.

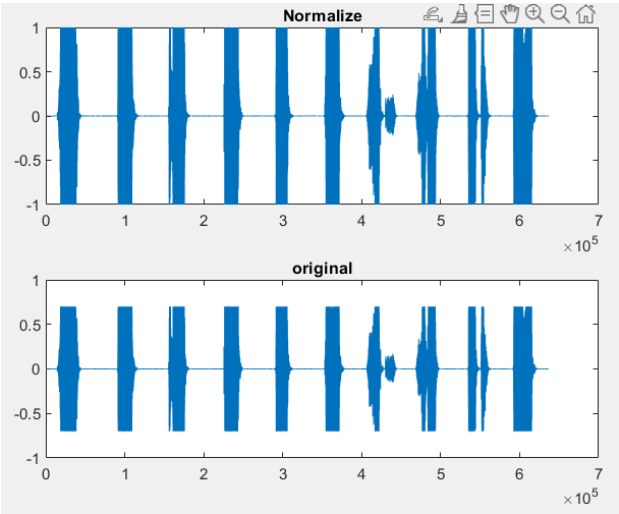


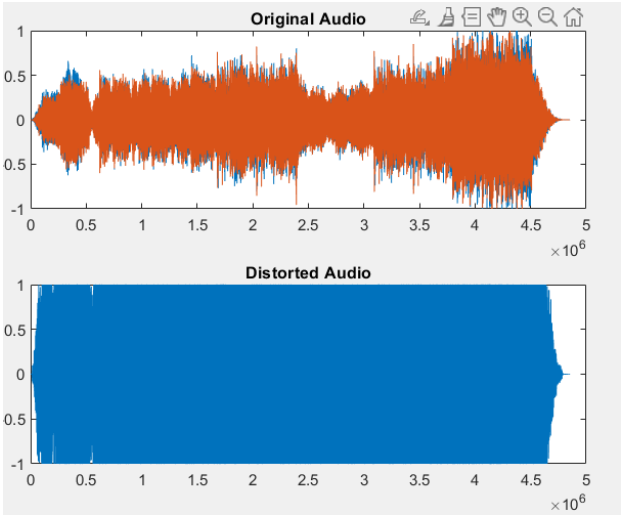
Fig.30: Recovering original from Normalize.

C. Reverse Distortion

The Reverse Distortion effect alters the audio by making it sound smoother or softer, typically by reducing the signal strength below its normal limits. This can give the sound a muted or compressed quality.

1) Implementation: We built a hanning filter to remove distortion.

2) Input and output analysis: Users can choose a reduction factor to apply reverse distortion. In this sample, an input attenuation factor of 1 is given. We will take previously distorted output as an input signal. However, we can't recover fully from noisy signal, we tried best to have the best recovery but from graph we can see still noisy output.



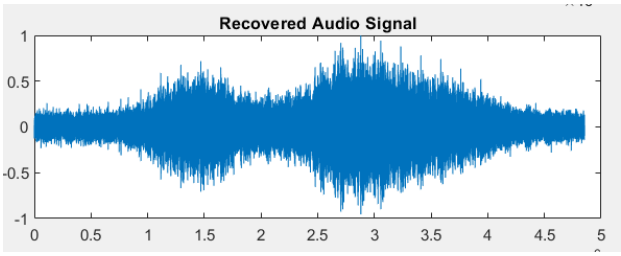


Fig.31: Recovered Signal.

XI. Reverse Spectrum Effect

A. Reverse Chorus

Reverse chorus means removing the clone voices form signal.

1) Implementation: Same way of chorus implement just reverse this times.

2) Input and Output Analysis: Here, we take feedback gain and delay time used to make chorus, from user.For sample , we used the previously chorused signal. However, there is error that we didn't fully succeed to remove chorus.

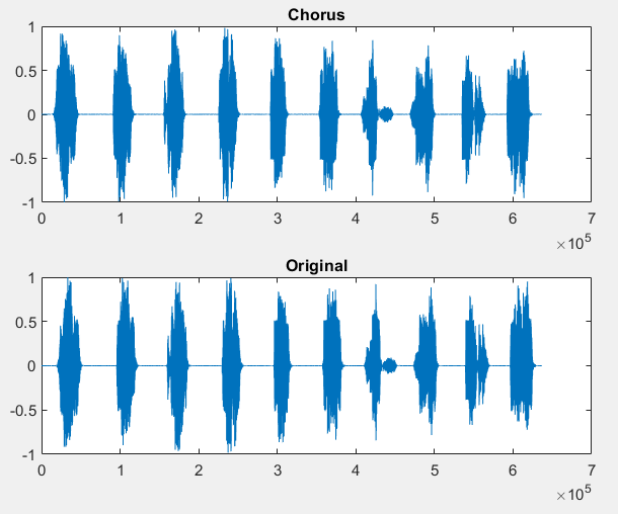


Fig.32: Recovered Signal.

B. Reverse Equalizer

For reversing the equalizer just need to give gain opposite of previously given input, in every band to retrieve the original signal.

C. Reverse Pitch Effect

Reverse pitch shifting restores the pitch to its original value after it has been shifted. This allows returning deep or shallow voices back to their natural pitch.

1) Implementation: We reverse pitch shifting by resampling the signal using the inverse of the original shift factor. We use MATLAB's built-in resample function to undo the pitch change.

2) Input and Output Analysis: User inputs the same semitone shift used originally (between -8 and 8) to reverse the effect. For instance, using semitones of -5 will return the pitch to its original value, as seen from the graphs when comparing x-axis shifts.

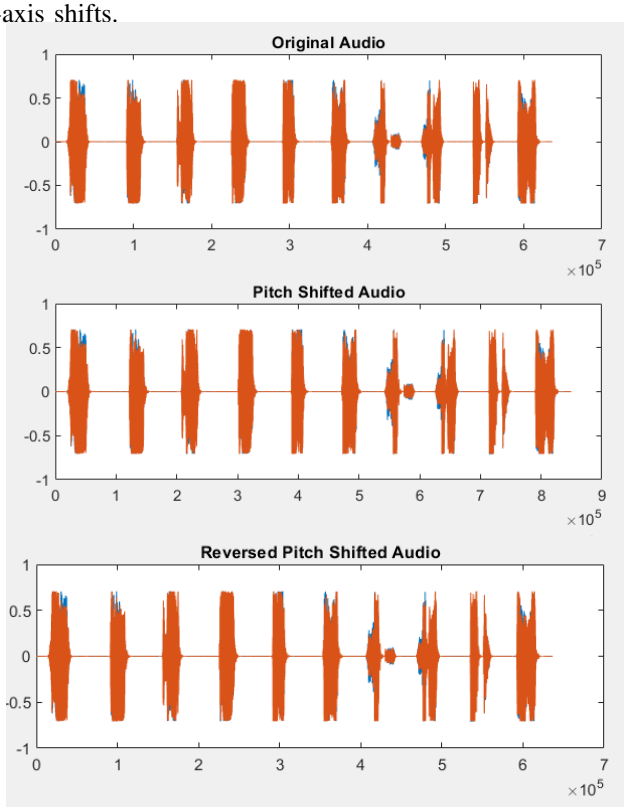


Fig.33: Recovered Signal.

XII. Reverse Time Based Effect

Reversing time-based audio effects to perfectly restore the original signal is inherently challenging. Despite extensive exploration, including attempts using techniques such as FFT, reverse convolution, and thorough research of MATLAB documentation and online resources, we were unable to find a method capable of accurately undoing these effects. The nature of time-based processing often leads to irreversible alterations, making precise restoration impractical.

XIII. Result and Analysis

Our system successfully implements all audio effects expect some reverse and provides intuitive controls for each. We tested it using various audio types, including speech and music, and the effects were applied effectively. Users can quickly modify audio and hear the impact of their chosen effects in real-time, demonstrating the processor's usability and flexibility.

XIV. Applications

- Real-Time Audio Processing.
- Audio Editing and Production.
- Research in Audio Analysis.
- Interactive Audio Applications.

XV. Limitations

- Irreversibility of Time-Based Effects.
- Limited User Customization.

XVI. Conclusion

The Audio Effects Processor project highlights the power of DSP in audio manipulation, successfully integrating various effects into a cohesive and easy-to-use system. The project's practical application of DSP techniques, combined with a user-friendly interface, offers a valuable tool for audio enthusiasts and professionals. Future work could focus on optimizing the system for real-time processing and extending its capabilities with more advanced effects.

XVII. Attached Project Code, File, and Details

- Project Details - Google Drive
- GitHub Repository: Audio Effect Processor
- GitHub Repository: Audio De-Effect Processor
- GitHub Repository: Audio Equalizer