Presented to the Department of Electronics and Computer Engineering

De La Salle University - Manila

Term 2, A.Y. 2022-2023

In partial fulfillment

of the course Signals, Spectra, and Signal Processing Laboratory

In Course LBYEC4A-EK2

Audio Effects Application within the MATLAB Software

Submitted by:

DEE, Ethan Joseph S.

ONG, Mariel Annika S.

PERUEL, Matthew S.

Submitted to:

Mr. Ramon Stephen Ruiz

Date Submitted:

April 18, 2022

I. ABSTRACT

This research aims to compare the efficacy and efficiency of audio effects implemented through MATLAB functions, specifically for the delay, distortion, phaser, and flanger, against existing code with similar functionality. The study involves developing MATLAB functions for each of the audio effects, applying them to a test instrumental sample, and completing a comparative analysis of the performance of the designed functions with those of existing audio effect algorithms.

From the results of the study, the designed MATLAB functions for audio effects were found to provide a higher quality customizable output, optimal for audio mixing and sound design with only a marginally higher memory usage compared to existing code.

II. INTRODUCTION

While there is a wide range of audio effects that can be used to enhance and modify audio recordings within the music production industry, the most popular among guitar effects are the following: delay, distortion, flanger, and phaser. Each of these differs in the unique element they add to the overall sound created. Delay creates an echo, or a repeat of the original sound, by delaying the signal a certain amount of time and playing the delayed sound back alongside the original. Distortion effects create an overdriven sound, clipping the original audio to create harmonic distortion. Flanger effects are unique in that they create a sweeping, “jet plane”-like effect by mixing a slightly delayed version of the original audio but modifying the delay with respect to time. This in turn causes a phase shift that sounds as though the audio is sweeping back and forth. Phasers, while like flanger effects, create a more distinct effect by splitting the given signal into two, delaying one of the splitted signals and phase-shifting it. The signals are then mixed back together to create a swirling effect highlighted well in guitar solos.

These effects are able to add interest, depth, and complexity to audio samples and allow producers and creators to customize their sound and have artistic freedom with their music.

III. THEORETICAL CONSIDERATIONS

A. Loading, saving and running MATLAB files from the code window

A MATLAB live script may run other live scripts using the command *run(scriptname)* where scriptname is the file name of the .m or .mlx file. Instead of passing variables to the script file, the user may save the workspace variables in a .mat file using the *save(filename, variables)* where filename is the name of the .mat file to be written and variables are the desired variables to be saved in string format. To access saved variables from a .mat file, the user may use the *load(filename, variables)*.

B. Data normalization

Given an array 'audio' containing audio data, the data may be normalized with the following MATLAB syntax: *audio = audio/(max(abs(audio))*. The denominator returns the maximum value of the array 'audio' so the entire syntax normalizes the audio data to peak at absolute value 1.

C. Data interpolation

Data interpolation is similar to time scaling in the sense that the user may add more samples in between the original audio samples to increase the resolution while still resembling the original audio data. For this purpose, the user may use the MATLAB function *vq = interp1(x,v,xq)* where 'vq' are the interpolated values, 'x' are the sample points, 'v' are the sample values, and 'xq' are the query points. Use the MATLAB function

D. Array size preallocation

MATLAB may sometimes return a warning stating that an array changes size at each iteration of a loop function. This programming practice may cause the program to run slowly. For this, the user may use the MATLAB syntax: *myarray = zeros(sz)*. The *zeros()* function was used to initialize the array to have all cell values equal to zero. The variable 'sz' refers to the desired size of the initialized array.

E. Low pass and high pass filters

To design a Butterworth filter in MATLAB, it is necessary to use the *buttord* and *butter* functions. The MATLAB syntax *[n,Wn] = buttord(Wp,Ws,Rp,Rs)* returns the unitless filter order 'n' and the normalized cutoff frequency 'Wn'.

IV. METHODOLOGY

The simulation consists of five live scripts: home page, distortion effect, digital delay effect, phaser effect, and flanger effect. The procedure for completing each live script are listed below.

A. Homepage

1. Add an interactive file selector to allow the user to select the input audio file.
2. Create a drop down menu for the user to select the guitar effect to use.

3. Create variables for the audio effects parameters listed below. Use interactive sliders.
    a. Distortion - gain, tone
    b. Digital delay - time, feedback
    c. Phaser – mix, delay, range, sweep frequency
    d. Flanger – depth, rate, mix
4. Create checkboxes for the following user preferences.
5. Gather audio data and the sampling frequency from the selected audio file.
6. Save the gathered data to a mat file.
7. Run the specific live script depending on the selection from the drop down menu in procedure A4.
8. Load the effect output from a .mat file.
9. Write the effect output as a new audio file.
10. Play the effect output sound.

B. Distortion effect

1. Load data from the home page .mat file.
2. Normalize the audio input.
3. Apply the gain parameters to the audio.
4. Apply the filter using the tone parameter.
    a. Design the first-order low pass Butterworth with a cutoff frequency of 8 kHz.
    b. Design the first-order high pass Butterworth with a cutoff frequency of 2 kHz.
    c. Convert the filter ZPK models to transfer function representation.
    d. Filter the amplified audio signal.
5. Generate the output as the wet signal.
6. Save the effect output to a .mat file.

C. Digital delay effect

1. Load data from the home page .mat file.
2. Normalize the audio input.
3. Duplicate the input signal
4. Apply feedback and iterated attenuation using the duplicate signal.
5. Generate output as sum of the dry and wet signal.
6. Normalize the output to avoid clipping.
7. Scale output to avoid clipping.
8. Save output to a .mat file.

D. Phaser effect

1. Load data from the home page .mat file.
2. Convert the delay parameter in ms to number of samples.,
3. Interpolate the audio data.

4. Preallocate array sizes for the output, dry, and wet signals.
5. Generate the output signal
    a. Generate the dry signal as the input signal.
    b. Calculate the index with the applied delay.
    c. Calculate the phase shift with the time varying delay.
    d. Array indices must be integers. Round the 'sweep' variable in case if not integer.
    e. Calculate the total delay index.
    f. Generate the wet signal as the delayed and time-varying phase shift input.
    g. Generate the output as the sum of the dry and wet signal.
    h. Downsample the input and audio data.

E. Flanger effect

1. Load data from the home page .mat file.
2. Normalize the audio input.
3. Interpolate the audio data.
4. Generate the time vector with the same length as the interpolated audio data.
5. Generate the modulating signal.
6. Design the FIR filter with the following syntax:
    a. numsamples = round(depth * Fs);
    b. h = [1 zeros(1, numsamples - 1) -0.5 zeros(1, numsamples - 1)];
7. Filter the input to produce a delayed signal.
8. Mix the delayed signal and modulating signal to produce the wet signal.
9. Generate the output as the sum of the dry and wet signal.
10. Downsample the input and output audio data.
11. Scale the output to avoid clipping.
12. Save the output to a .mat file.

V. RESULT

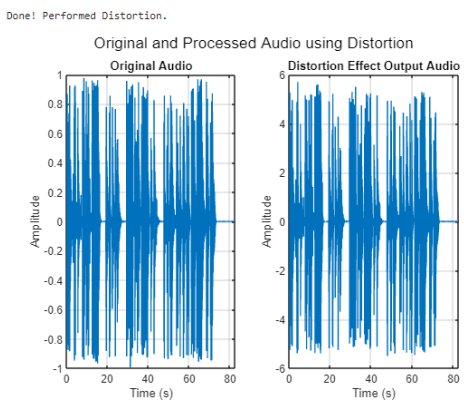

Figure 5.1. Input and Output Signals for Distortion Effect (Self-design)

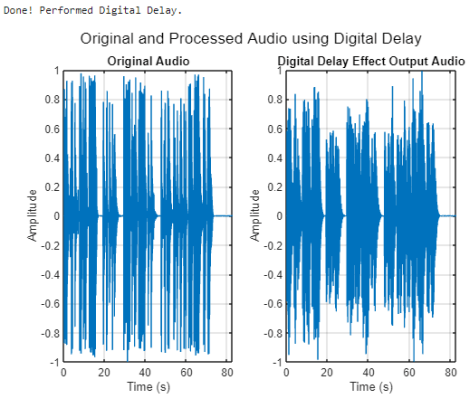

Figure 5.3. Input and Output Signals for Digital Delay Effect (Self-design)

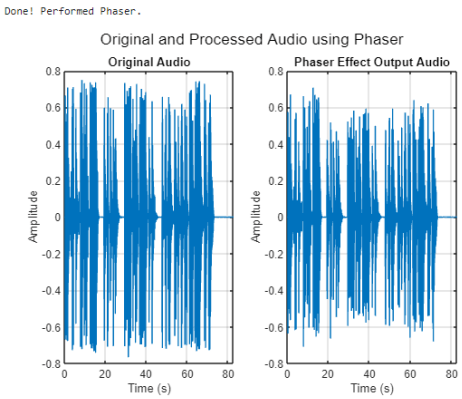

Figure 5.5. Input and Output Signals for Phaser Effect (Self-design)

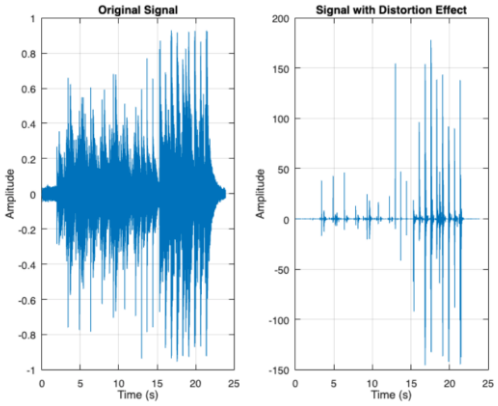

Figure 5.2. Input and Output Signals for Distortion Effect (Existing)

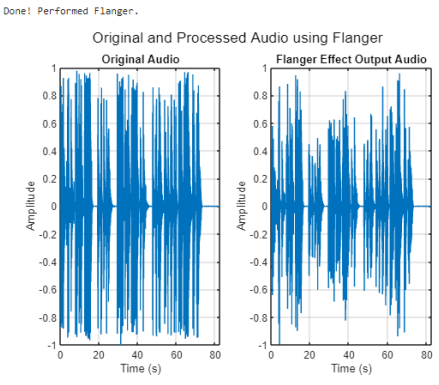

Figure 5.7. Input and Output Signals for Flanger Effect (Self-design)

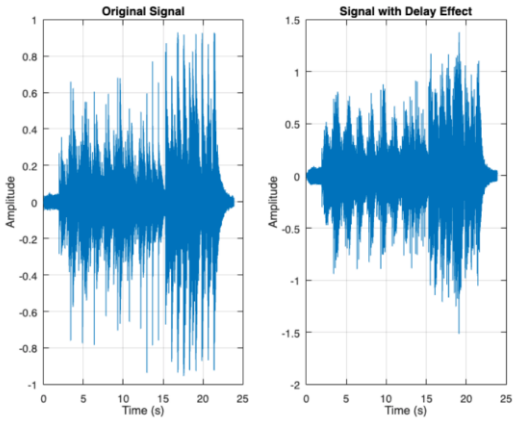

Figure 5.4. Input and Output Signals for Digital Delay Effect (Existing)

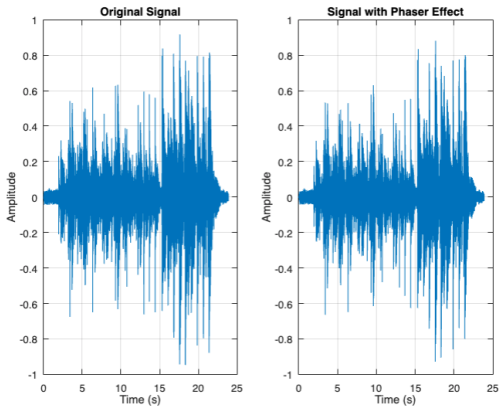

Figure 5.6. Input and Output Signals for Phaser Effect (Existing)

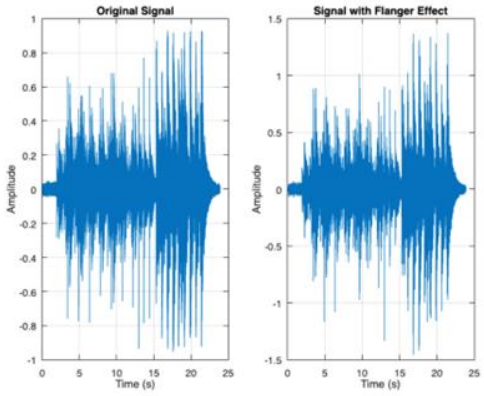

Figure 5.8. Input and Output Signals for Flanger Effect (Existing)

VI. DISCUSSION

The term (x(t)/abs(x(t))) in the distortion formula generates a sign function that preserves the polarity of the input signal. This ensures that the output signal will have the same direction as the input signal, which is important for maintaining the original character of the sound being distorted. The second term in the formula, e^((A * x(t)^2)/abs(x(t))), generates a curve that smoothly rises as the absolute value of the input signal increases. This curve saturates at a maximum output level determined by the gain parameter A, which controls the amount of amplification applied to the distorted signal. The effect of this exponential function is to create a gradual increase in distortion as the input signal becomes larger, which can result in a warm, natural-sounding distortion [1].

$$f(x) = \frac{x}{|x|}(1 - e^{\frac{ax^2}{|x|}})$$

Based on Figure 5.1, applying distortion to the input signal increases the amplitude of the output signal. When a signal is distorted, there are additional frequency components like harmonics or overtones added to the input signal. The amplitude of the additional harmonics can be greater than the input signal which results in an overall increase in the amplitude of the output signal. This increase in amplitude may be interpreted as an increase in the "loudness" of the audio signal.

The formula y(t) = x(t) + G*y(t-d) produces the repeating echo effect by combining the current input signal with a delayed version of the input signal. The quantity of signal transmitted back into the delay line is controlled by the feedback gain parameter G, which also affects the echo effect's overall character and decay rate. The length of the delay, which impacts the echo's pitch and rhythmic feel, is controlled by the delay time parameter d [2].

For digital delay, it was observed that the output signal has more space since the effect can be done by adding a delayed copy of the input signal to the output signal. This creates more depth or produces a wider sound. Comparing Figures 5.3 and 5.4, the outputs are similar where there's a time shift of the entire waveform. The parameters for the delay effect and the characteristics of the input signal will determine the appearance of the time domain plot.

The phaser effect resembles the flanger effect, but instead of employing a delay that varies with time, it uses a sequence of all-pass filters that depend on frequency. By arranging multiple all-pass filters in a row, each with its own feedback gain and delay time, the phaser effect can be generated. Consequently, the sound of a phaser is generated by a sequence of frequency spectrum notches that move up and down in frequency [3].

$$y(t) = 0.5x(t) + 0.5x(t) * [h_{\mathrm{AP1}}(t) * h_{\mathrm{AP2}}(t) * \cdots * h_{\mathrm{APN}}(t)].$$

A phaser effect can add motion and texture to an audio signal by altering the phase relationship between different parts of the signal. As a result, this creates peaks and dips in the frequency response of the output signal which is what produces the whooshing sound associated with the phaser effect. It was observed that the amplitude of the output is smaller than the input. This could be due to the characteristic of the effect where the amplitude is frequency dependent. The phaser effect creates a combing filter which is an all-pass filter, and this affects the phase response of the output signal.

The formula for producing the flanger effect is depicted in the figure below, where y(n) represents the resulting signal, a and b are linear gain factors that determine the extent to which the filter's frequency response notches are emphasized, and M denotes the delay time [4].

$$y(n) = ax(n) + bx(n - M);\ \text{where the delay is of M samples.}$$

Similar to the phaser effect, the flanger effect also creates a comb filter effect where a delayed copy of the input signal is modulated and peaks in the frequency response are created. A time-varying LFO (low-frequency oscillator) is used to modulate the delayed copy of the audio signal in a flanger effect. This causes the delay time to change continuously over time, creating a sweeping, jet-like effect that is characterized by a series of overlapping and changing delay times. It was observed in Figure 5.7 that there are many peaks that vary in amplitude over time. This is the same observation in Figure 5.8.

Upon conducting an audio test for each effect's output signal, it has been noted that the output for both self-design and existing codes of the distortion effect are similar. The difference was the loudness however, this can be adjusted by refining the existing code to match the self-design code. As for the delay and phaser effects, both output signals are almost similar however, the self-design code produced a better application of the said audio effects. Only the existing code for the flanger effect produced an output that doesn't have a difference from the input.

VII. CONCLUSION

The group was successful in creating an interactive program that offers audio processing through 4 effects: distortion, digital delay, phaser, and flanger. DSP concepts such as signal operations, FIR filters, and low pass and high pass filters. These concepts were applied in the simulation written as MATLAB live scripts, one for each guitar effect and the home page. Compared to existing codes, the distortion, digital delay, and phaser effects of the authors' program define the characteristics of each effect more clearly than that of the existing MATLAB codes. However, for the flanger effect, there is not much difference in the plotted waveform and sound of the tested audio.

VIII. AUTHORS CONTRIBUTION

Member	Contribution
DEE, Ethan Joseph S.	Abstract, Introduction, Flowchart
ONG, Mariel Annika S.	MATLAB Script on Existing Codes, Discussion, Project Poster
PERUEL, Matthew S.	MATLAB Script on Author's codes, theoretical considerations, methodology

.

IX. REFERENCES

[1] O. DAS, "DIGITAL AUDIO EFFECTS." [Online]. Available: https://ccrma.stanford.edu/~orchi/Documents/DAFx.pdf

[2] "Lab 2: Audio Effects and Real-Time Processing." [Online]. Available: http://pfister.ee.duke.edu/courses/ece485/lab2.pdf

[3] U. Zölzer, "Audio Effects Generation," in Handbook of Signal Processing in Acoustics, D. Havelock, S. Kuwano, and M. Vorländer, Eds. New York, NY: Springer, 2008, pp. 785–796 [Online]. Available: https://doi.org/10.1007/978-0-387-30441-0_41. [Accessed: Apr. 17, 2023]

[4] A. Wright and V. Välimäki, "Neural Modeling of Phaser and Flanging Effects," JAES, vol. 69, no. 7/8, pp. 517–529, Jul. 2021 [Online]. Available: https://www.aes.org/e-lib/browse.cfm?elib=21119. [Accessed: Apr. 17, 2023]