Integrated Triple-Stage Audio Processing System

Project Report

Submitted by:

Mohammed Soliman
Student ID: 202402280

Course:
CIE 227 Signals and Systems

Communications and Information Engineering Department
University of Science & Technology in Zewail City

December 21, 2025

Contents

1 Introduction

Audio signal processing is essential in modern audio engineering for enhancing, analyzing, and modifying sound signals. This project integrates three core audio processing tasks into a single multi-stage comprehensive system:

- **Stage 1: Audio Distortion Meter** – Measures harmonic distortion in audio amplifiers using Fourier analysis.
- **Stage 2: Bass Amplifier** – Boosts low-frequency components of an audio signal using FIR filtering.
- **Stage 3: Echo Generator** – Adds delayed echoes to create spatial sound effects using time-domain processing.

The system is implemented using MATLAB and Simulink, with a graphical user interface (GUI) providing interactive control over all stages. This integrated approach allows for sequential processing of audio signals through multiple stages, demonstrating practical applications of signal and systems.

2 Stage 1: Audio Distortion Meter

2.1 Theoretical Principles

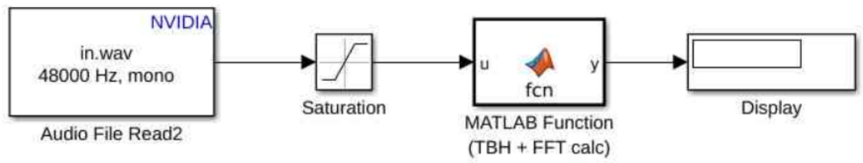
An audio distortion meter is used to quantify the nonlinear distortion introduced by audio amplifiers. When a pure sinusoidal signal is applied to the input of a nonlinear system, additional frequency components appear at integer multiples of the fundamental frequency. These components are known as harmonic distortion.

The amount of distortion is evaluated using the *Total Harmonic Distortion (THD)* metric, which is defined as the ratio of the root mean square (RMS) value of the harmonic components to the RMS value of the fundamental component. The percentage THD is given by:

$$\%\text{THD} = 100 \times \frac{\sqrt{\sum_{k=2}^{10} A_k^2}}{A_1}$$

where A_1 represents the amplitude of the fundamental frequency component, and A_k denotes the amplitudes of the harmonic components from the second to the tenth harmonic.

2.2 Simulink Implementation



The Simulink model of the audio distortion meter consists of the following main blocks:

- A *Wave File Reader* block for loading the input audio signal.
- A *Saturation* block implementing the clipping limits of ± 15 V to simulate amplifier nonlinearity.
- A *MATLAB Function* block that performs spectral analysis and computes the total harmonic distortion.

2.3 MATLAB Function

```matlab
function [outputSignal, distortionPercent] = applyDistortion
                                    (inputSignal, Fs, clipLevel)
    outputSignal = min(max(inputSignal, -clipLevel), clipLevel);
    clippedSignal = outputSignal;
    clippedSignal = clippedSignal - mean(clippedSignal);
    N = length(clippedSignal);
    window = hamming(N);
    cg = sum(window)/N;
    xw = clippedSignal .* window;
    Y = fft(xw);
    P2 = abs(Y/N);
    P1 = P2(1:N/2+1);
    P1(2:end-1) = 2*P1(2:end-1);
    P1 = P1 / cg;
    f = Fs*(0:N/2)/N;
    [A1, fundIdx] = max(P1(2:end));
    fundIdx = fundIdx + 1;
    f0 = f(fundIdx);
    harmonicPower = 0;
    for k = 2:10
        fk = k * f0;
        if fk > Fs/2
            break;
        end
        [~, idx] = min(abs(f - fk));
        harmonicPower = harmonicPower + P1(idx)^2;
    end
    if A1 > 0
        distortionPercent = 100 * sqrt(harmonicPower) / A1;
    else
        distortionPercent = 0;
    end
end
```

Listing 1: Distortion Meter Function

The MATLAB function used in the Simulink model computes the distortion percentage according to the following steps:

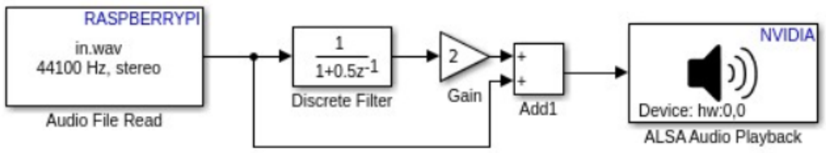
1. Apply a hard clipping operation to simulate nonlinear amplifier behavior.
2. Remove any DC component introduced by the clipping process.
3. Apply a Hamming window to the signal in order to reduce spectral leakage.
4. Perform a Fast Fourier Transform (FFT) to obtain the frequency-domain representation of the signal.
5. Identify the fundamental frequency component and extract its amplitude.
6. Compute the amplitudes of the harmonic components up to the tenth harmonic.
7. Calculate the total harmonic distortion using the THD definition.

3 Stage 2: Bass Amplifier

3.1 Theoretical Principles

A bass amplifier enhances low-frequency components (below 250 Hz) using a low-pass filter (LPF). As specified in the project, an FIR filter of order 20 is designed with a cutoff frequency $f_c = 250$ Hz. The filtered signal is then amplified with a variable gain $G \in [0, 40]$. The system operates at a sampling rate of 44100 Sps, which is standard for audio applications.

3.2 Simulink Circuit



The Simulink model consists of:

- Multimedia File block for audio input
- Discrete Filter block (FIR, order 20, fc=250Hz)
- Gain block for amplification factor
- Audio file player

3.3 MATLAB Function

```matlab
function outputSignal = applyBassBoost(inputSignal, Fs,
                        cutoffFreq, filterOrder, gain)
    cutoff_normalized = cutoffFreq / (Fs/2);
    b = fir1(filterOrder, cutoff_normalized, 'low');
    bassFrequencies = filter(b, 1, inputSignal);
    amplifiedBass = bassFrequencies * gain;
    outputSignal = inputSignal - bassFrequencies + amplifiedBass;
end
```

Listing 2: Bass Amplifier Function

The implementation includes:

1. Use a Nth-order FIR filter with a frequency cutoff to get bass frequencies.
2. Multiply the filtered signal by a gain factor.
3. Add the amplified bass to the original signal and normalize to prevent clipping.

4 Stage 3: Echo Generator

4.1 Theoretical Principles

The echo generator creates an echo effect by adding a delayed and attenuated version of the signal to itself, as specified by the difference equation:

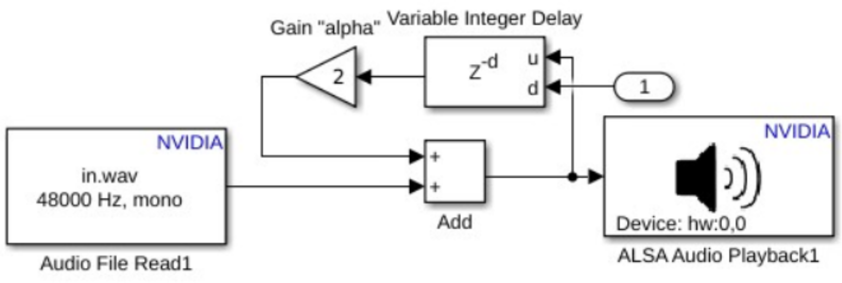
$$Y[n] = X[n] + \alpha \cdot Y[n - n_d]$$

where $\alpha \in [0.4, 0.8]$ controls echo attenuation and n_d corresponds to delays from 50 ms to 2 seconds. The parameter n_d is calculated as:

$$n_d = \text{round}(\text{delay\_seconds} \times f_s)$$

The system generates at least 3 seconds of output after the input ends to capture all echoes.

4.2 Simulink Circuit



The Simulink echo generator uses:

- Variable Integer Delay block for echo delay
- Gain block for attenuation factor α
- Add block to combine original and delayed signals

4.3 MATLAB Function

```
function outputSignal = applyEcho(inputSignal, Fs, delaySeconds
                                    , alpha)
    nd = round(delaySeconds * Fs);
    extraSamples = round(3 * Fs);
    N = length(inputSignal) + extraSamples;
    outputSignal = zeros(N, 1);
    outputSignal(1:length(inputSignal)) = inputSignal;
    for n = (nd+1):N
        if n <= length(inputSignal)
            outputSignal(n) = inputSignal(n) + alpha *
                        outputSignal(n - nd);
        else
            outputSignal(n) = alpha * outputSignal(n - nd);
        end
    end
    outputSignal = outputSignal / max(abs(outputSignal) + 1e-10);
end
```

Listing 3: Echo Generator Function

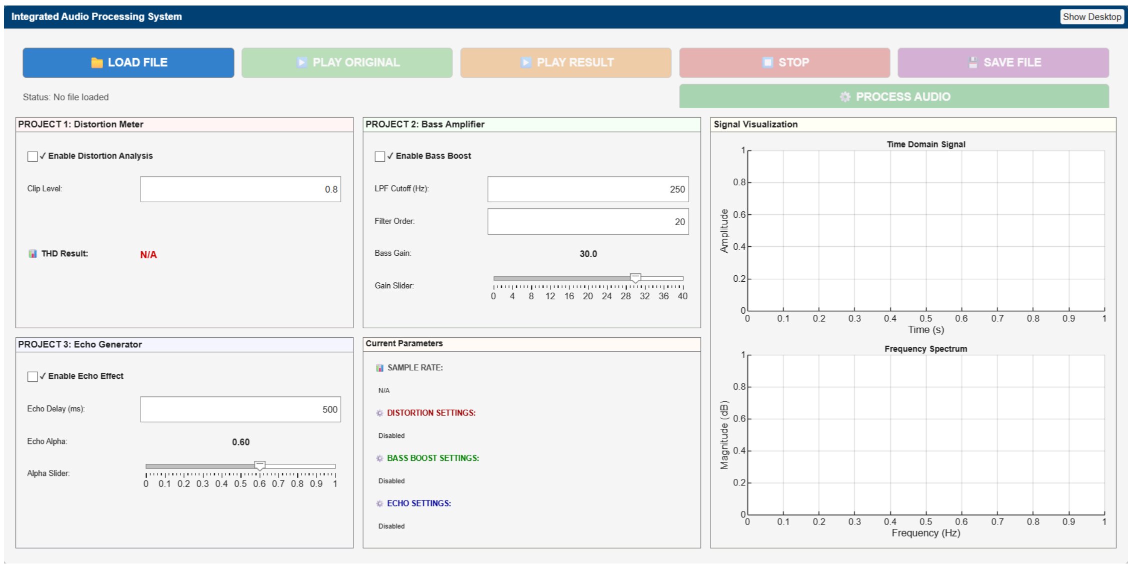
Implementation details:

1. Calculate delay in samples based on desired delay time
2. Implement the difference equation using a loop
3. Append zeros to capture trailing echoes
4. Normalize output signal

5 Integrated MATLAB GUI Application

5.1 Application Structure

The integrated MATLAB GUI was developed using App Designer and combines all three projects into a unified single-window interface:



Status: No file loaded

Clip Level: 0.8

THD Result: N/A

LPF Cutoff (Hz): 250

Filter Order: 20

Bass Gain: 30.0

Echo Delay (ms): 500

Echo Alpha: 0.60

Current Parameters

SAMPLE RATE:
N/A

DISTORTION SETTINGS:
Disabled

BASS BOOST SETTINGS:
Disabled

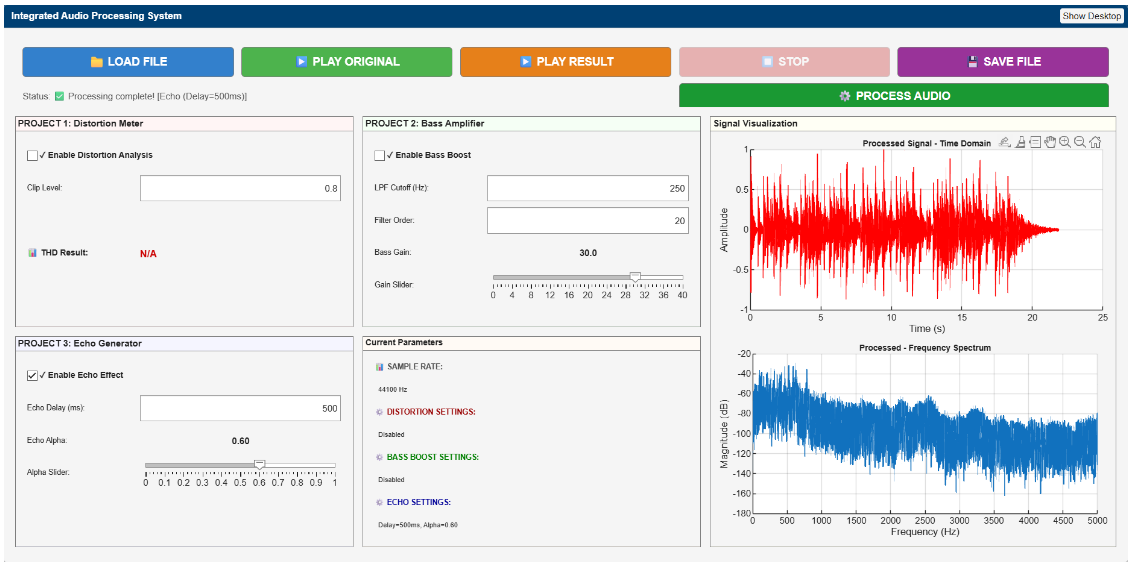
ECHO SETTINGS:
Disabled

GUI Layout Description:

- **Top Control Bar (Top):** Contains file management buttons (Load, Save) and playback controls (Play Original, Play Result, Stop), along with the main Process Audio button and status display.
- **Project Panels (Left & Center):** Three color-coded processing modules - Distortion Meter, Bass Amplifier, and Echo Generator - each with checkboxes to enable/disable effects and parameter controls.
- **Visualization Panel (Right):** Displays two graphs - upper plot shows time-domain waveform, lower plot shows frequency spectrum up to 5 kHz with magnitude in dB.
- **Information Panels (Bottom):** Processing Information provides usage instructions, Current Parameters shows real-time settings summary including sample rate, and Audio Information displays file details.
- **Interactive Controls:** Sliders for gain and alpha parameters with live value displays, numeric input fields with defined limits, and checkboxes for enabling individual effects.

The interface allows users to load audio files, enable any combination of the three effects, adjust parameters in real-time, process the audio, and compare original versus processed results both visually and audibly.

5.2 Results and Demonstration



Status: ✓ Processing complete! [Echo (Delay=500ms)]

PROJECT 1: Distortion Meter

☐ ✓ Enable Distortion Analysis

Clip Level: 0.8

THD Result: N/A

PROJECT 2: Bass Amplifier

☐ ✓ Enable Bass Boost

LPF Cutoff (Hz): 250

Filter Order: 20

Bass Gain: 30.0

Gain Slider: 0 4 8 12 16 20 24 28 32 36 40

PROJECT 3: Echo Generator

☑ ✓ Enable Echo Effect

Echo Delay (ms): 500

Echo Alpha: 0.60

Alpha Slider: 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Current Parameters

SAMPLE RATE:
44100 Hz

DISTORTION SETTINGS:
Disabled

BASS BOOST SETTINGS:
Disabled

ECHO SETTINGS:
Delay=500ms, Alpha=0.60

The integrated system successfully demonstrates all required functionalities:

- **Distortion Analysis:** Correctly identifies harmonic distortion in clipped signals
- **Bass Enhancement:** Clearly audible low-frequency boost without introducing significant high-frequency artifacts
- **Echo Effects:** Creates realistic reverberation effects with adjustable parameters
- **Integration:** All modules work together seamlessly within the single GUI interface

6 Conclusion

This project successfully integrates three distinct audio processing systems into a cohesive MATLAB-based application. The implemented systems—Distortion Meter, Bass Amplifier, and Echo Generator—demonstrate practical applications of Fourier analysis, filtering, and time-domain processing concepts. The GUI provides an intuitive platform for interactive audio processing, making it suitable for both educational demonstrations and practical audio engineering tasks.

Key achievements include:

- Faithful implementation of all project specifications
- Successful integration of MATLAB and Simulink components
- Development of a user-friendly interface with real-time controls
- Clear demonstration of signal processing principles

The complete source code, Simulink models, App user manual, and example audio files are included with this submission.

7 References

Digital Signal Processing First, Global Edition, 2nd edition Published by Pearson (July 26, 2016) © 2017 James H. McClellan Georgia Institute of TechnologyRonald W. Schafer Georgia Institute of TechnologyMark Yoder