Bengisu Münteha Koç

21801974

EEE 321 – 01

31/10/2024

EEE 321 Lab Work 3

Introduction

The objective of this experiment is to simulate how music would sound in a concert auditorium by capturing the acoustics of the space and applying them to an anechoic recording. This is achieved by first recording an approximate impulse response of the auditorium, which captures how sound reflects and decays within the space. Using the impulse response as a filter, the music recorded in an anechoic environment (where there are no reflections or echoes) is passed through a linear time-invariant (LTI) system to simulate how the music would be perceived by a listener seated in a specific position in the auditorium. This process allows for a more realistic recreation of the auditory experience, including the effects of the environment's acoustics on the clarity, reverberation, and overall sound quality.

Background

Convolution is a mathematical operation that combines two signals: the input signal and the impulse response. In this assignment, input signal is the anechoic music while the impulse response is the balloon pop recording from the echoic environment. The convolution result is the output that simulates how the input signal would sound if played in the environment characterized by impulse response.

The continuous time convolution is defined as

$$y(t) = (x * h)(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\ d$$

Where x(t) is input signal and h(t) is impulse response signal.

The discrete time convolution is defined as

$$y[n] = (x * h)[n] = \sum_{k=0}^{N-1} x[k]h[n-k]$$

Where x[n] is input signal and h[n] is impulse response signal.

Impulse is a signal that has all its energy concentrated at a signal instant in time, represented by δ(t) in continuous systems and δ[n] in discrete systems. It is characterized by being zero everywhere except at the point of application. In other words, it is an idealized, instantaneous burst of energy. It is a tool for understanding how systems react to a broad range of inputs. In this assignment, balloon pop is the impulse.

Impulse response is the output of a system when an impulse signal is applied as input. In describes the system's behavior by showing how it responses over time to impulse. The input response characterizes the system's inherent properties. It is a fundamental tool for analyzing linear time invariant (LTI) systems. In other words, the input response reveals how a system processes any arbitrary input, as any complex signal can be decomposed into a series of impulses. In this assignment, the recording of balloon pop is the impulse response.

The output signal is the result of the convolution. On the basis of the assignment, when the anechoic music is convolved with balloon pop recording, the room's acoustic characteristics are applied to the anechoic music.

Mathematically, convolution operation slides the input response over the input signal, calculating weighted sum at each step. This process multiplies each point of the input signal by the impulse response at different time shifts and sums them up. The values of the impulse response determine how much of the input signal gets added

to the output at each point in time. In this case, the impulse response represents the echoic behavior of the room, it decides how the echoes and reverberations get distributed over the input signal.

In the assignment, the impulse response is a series of peaks and decays which represent the reflections of sound from various surfaces in the environment. When it is convolved with anechoic music, the convolution operation spreads out and overlaps the input music with these peaks and decays, creating the effect of echoes and reverberations.

The peaks in impulse response signal lead delays at the output and simulate the reflections. The height and width of these peaks in impulse response control the strength and length of the echoes in the output.

Each point in time in anechoic music is replaced by a scaled version of the impulse response. For example, in the loud part in the music, the impulse response gets scaled according to the loudness of the music at that point, and then added to the output.

The first part of the impulse response corresponds to the direct sound heard immediately. The middle part of the impulse response corresponds to early reflections (short echoes). The later parts of the impulse response represent the long, decaying tail of reverberation caused by multiple reflections.

This is how the convolution adds the environment's treatment to the anechoic music.

Methodology

The balloon pop is recorded at Odeon from the point as shown in figure 1. The balloon is popped in the stage as shown in figure 2.

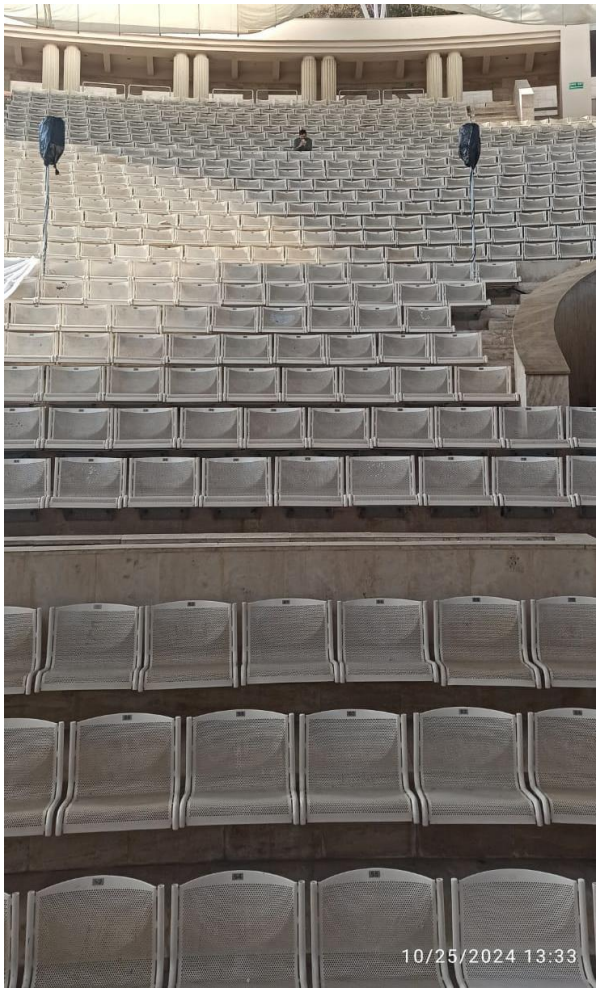

Figure 1: Recording Point

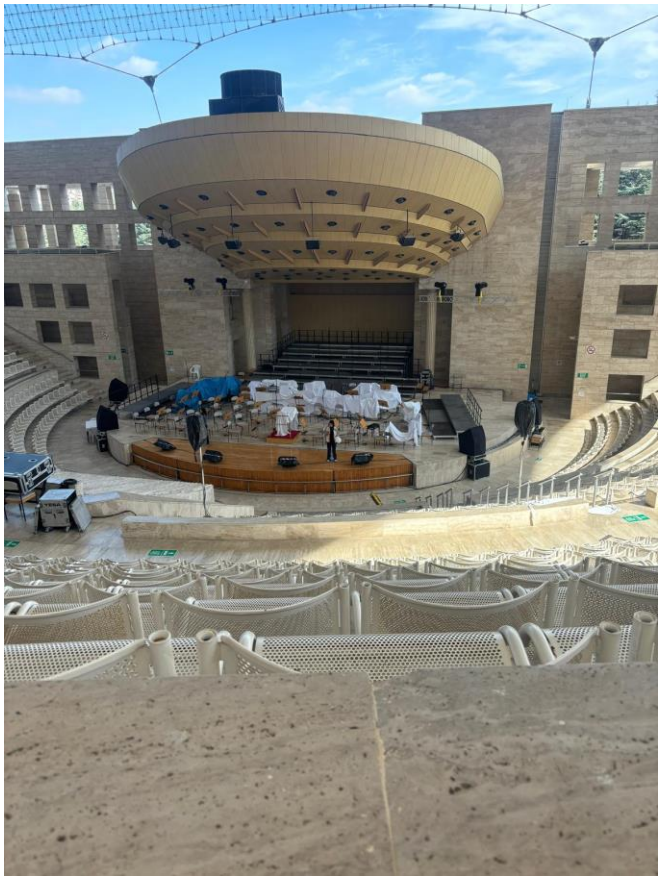

Figure 2: Balloon Pop from the Stage

By the explosion of the balloon, the impulse for the assignment is created and the impulse response is recorded with a cell phone.

For the input signal, one of the anechoic symphony music is chosen from the website [1] which is Symphony no.7, I movement, bars 1-53 from L. van Beethoven. In the downloaded music file, the piece was recorded separately in an anechoic environment. In this report, two approaches are adopted, one is compiling all the instruments before the convolution and then conduction convolution, the other one is picking one of the instruments and conducting convolution. Since convolution is an LTI system, both can be implemented.

The remaining process is maintained in MATLAB.

Implementation and Results

The sound of the balloon pop is saved as “balloon.wav” and the impulse response graph and sound are observed with the following code snippet:

```
% Impulse Response - Balloon Recording
impulseResponseFile = 'balloon.wav';
[impulseResponse, Fs] = audioread(impulseResponseFile); % Sampling Rate
extracted from the file


% Time vector based on sampling rate
% Linking samples to time for plotting
t = (0:length(impulseResponse)-1) / Fs;


% Impulse Response Plot
```

```
figure;
plot(t, impulseResponse);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Impulse Response of Balloon Pop');
grid on;


% Impulse Response Sound
sound(impulseResponse, Fs);
```

In this code, the recording is defined and audioread() function is used to read audio data. This function also extracts the sampling rate of the audio in the file; hence sampling rate is also defined as “Fs”. Then a time vector is created since graph of the impulse response will be plotted. The time vector is created by basing on sampling rate to link the samples to time for plotting. Then the plot is created as shown in figure 3 and the sound of the impulse response is played by the code.

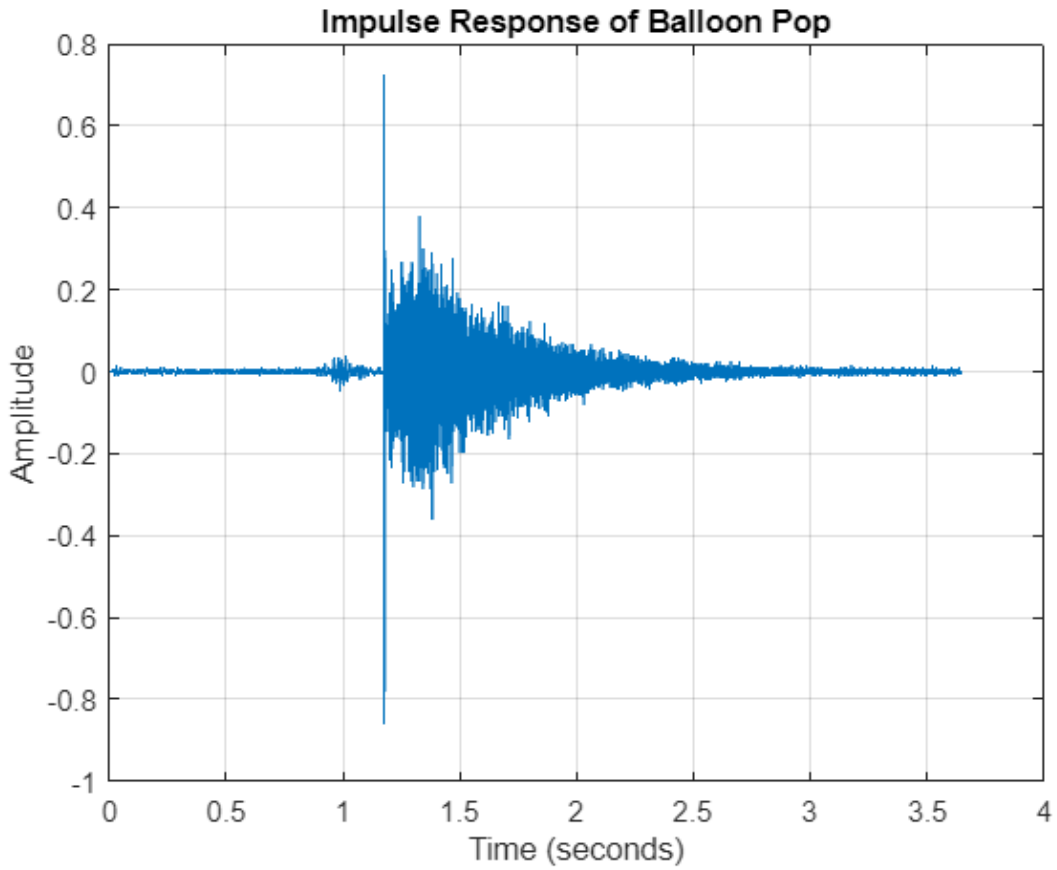

Figure 3: Plot of the Impulse Response (The Sound of the Balloon Pop)

In the system, impulse is not an ideal impulse since the ideal impulse starts and ends at the same point. Nevertheless, the exploding balloon is a good approximation of the acoustic impulse function. Hence, the impulse response in the above figure is not also an ideal impulse response. Also, the impulse response includes some distortions due to environment and recording device inadequacy.

In the first approach, all instruments in the file are compiled in MATLAB, then the sound is saved and played as shown in the following code

```
% Audio Files of the Instruments
instrumentFiles = {
    'beethoven_bsn1_5.mp3', 'beethoven_bsn1_6.mp3', 'beethoven_bsn2_5.mp3',
...
    'beethoven_bsn2_6.mp3', 'beethoven_cl1_6.mp3', 'beethoven_cl2_6.mp3', ...
    'beethoven_corno1_6.mp3', 'beethoven_corno1_8.mp3',
'beethoven_corno2_6.mp3', ...
    'beethoven_corno2_8.mp3', 'beethoven_fl1_6.mp3', 'beethoven_fl2_6.mp3',
...
    'beethoven_kb_6.mp3', 'beethoven_ob1_6.mp3', 'beethoven_ob2_6.mp3', ...
    'beethoven_timp_6.mp3', 'beethoven_tr1_6.mp3', 'beethoven_tr2_6.mp3', ...
    'beethoven_vc_6.mp3', 'beethoven_vl1a_6.mp3', 'beethoven_vl1b_6.mp3', ...
    'beethoven_vl2a_6.mp3', 'beethoven_vl2b_6.mp3', 'beethoven_vlaa_6.mp3',
...
    'beethoven_vlab_6.mp3'
};


% Variable to Store Complied Audio & Sampling Rate Initialization
compiledMusic = [];
Fs = [];


% Combining all instrument files to play the completed music
for i = 1:length(instrumentFiles)
    % Each auido file
    [instrumentAudio, currentFs] = audioread(instrumentFiles{i});

    % Stroing sampling rate which are extracted from the files
    if isempty(Fs)
        Fs = currentFs;
    elseif Fs ~= currentFs
        error('Sampling rates for all files must be the same.');
    end

    % Matching the longest file length
    if isempty(compiledMusic)
        compiledMusic = zeros(size(instrumentAudio));
    elseif length(instrumentAudio) > length(compiledMusic)
        compiledMusic = [compiledMusic; zeros(length(instrumentAudio)-
length(compiledMusic), size(compiledMusic, 2))];
    elseif length(instrumentAudio) < length(compiledMusic)
        instrumentAudio = [instrumentAudio; zeros(length(compiledMusic)-
length(instrumentAudio), size(instrumentAudio, 2))];
    end

    % Adding the audio files
    compiledMusic = compiledMusic + instrumentAudio;
end


% Reduce clipping
```

```
compiledMusic = compiledMusic / max(abs(compiledMusic(:)));


% Set duration to 30sec.
durationInSeconds = 30;
durationInSamples = Fs * durationInSeconds;


% Extracted 30sec from the middle of the record
totalSamples = length(compiledMusic);
startSample = round((totalSamples - durationInSamples) / 2);
endSample = startSample + durationInSamples - 1;


% Sample length setting
if endSample > totalSamples
    endSample = totalSamples;
end


% Extracting the segment
compiledMusicSegment = compiledMusic(startSample:endSample);


% Step 4: Plot the compiled music waveform (30 seconds)
t_segment = (0:length(compiledMusicSegment)-1) / Fs;
figure;
plot(t_segment, compiledMusicSegment);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Compiled Music (30 seconds)');
grid on;


% Step 5: Play the 30-second segment
sound(compiledMusicSegment, Fs);


% Step 6: Save the 30-second segment to a file (optional)
audiowrite('beethoven30sec.wav', compiledMusicSegment, Fs);
```

All the instrument audio files are added to defined variable and sampling rates are matched. Then the audio is arranged to reduce clipping. The duration of the music is set to 30sec to easy demonstration. The 30sec is extracted from the middle of the music. The sample lengths are matched, and the input signal is plotted as shown in figure 4.

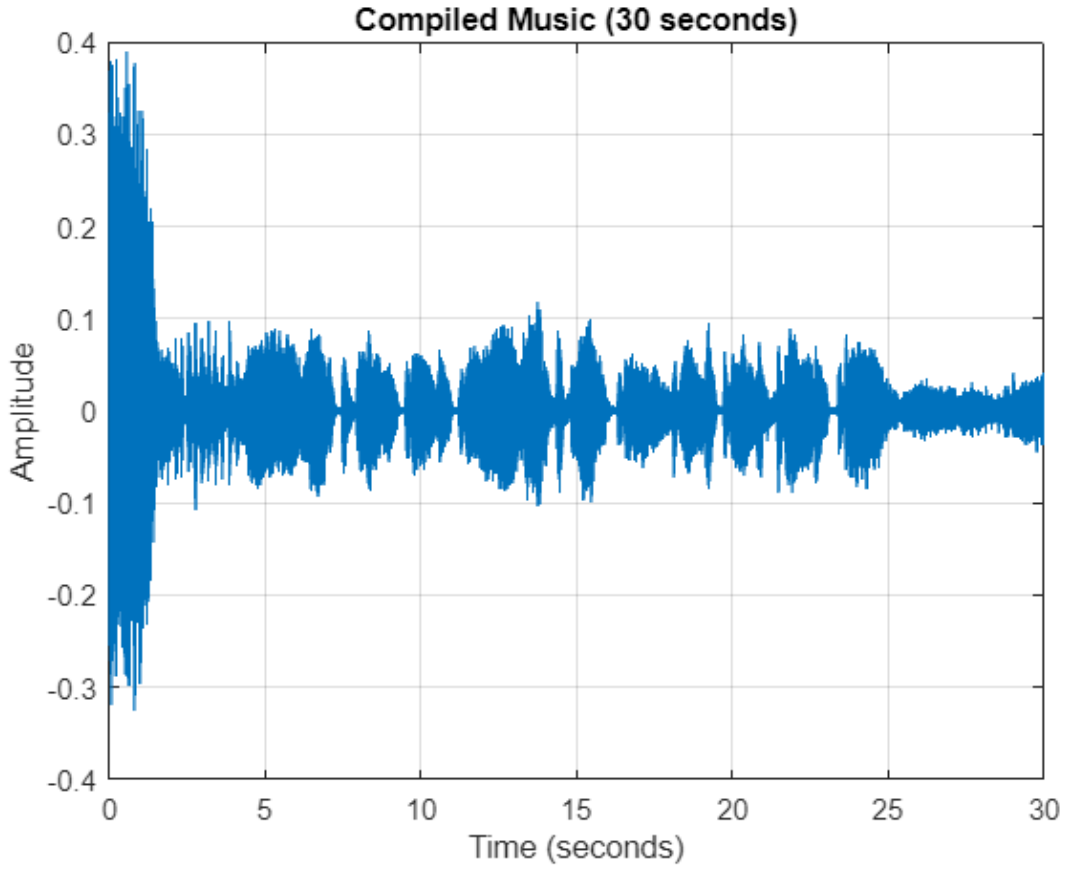

Figure 4: Input Signal (Compiled Music)

The compiled is saved to a file named “beethoven30sec.wav”.

The convolution is conducted with MATLAB as following

```
% Loading impulse response record to use
impulseResponseFile = 'balloon.wav';
[impulseResponse, Fs_imp] = audioread(impulseResponseFile);


% Samplign rate check
if Fs ~= Fs_imp
    error('Sampling rates of compiled music and impulse response must
match!');
end


% Convolving music with impulse response
outputMusic = conv(compiledMusicSegment, impulseResponse, 'same');


% Clipping Reduce
outputMusic = outputMusic / max(abs(outputMusic(:)));
```

```
% Convolved Output Music Plot
t_output = (0:length(outputMusic)-1) / Fs;
figure;
plot(t_output, outputMusic);
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Output Music (30 seconds)');
grid on;


% Play the Output
sound(outputMusic, Fs);


% Saving Output
audiowrite('output.wav', outputMusic, Fs);
```

In the code block, the impulse response is defined from the balloon pop recording audio file and the impulse response is convolved with compiled music segment with MATLAB built-in function conv(). Again, to reduce distortion, clippings are reduced. Then the output signal is plotted as shown in figure 5 and the output signal sound is recorded to an audio file named as “output.wav”.

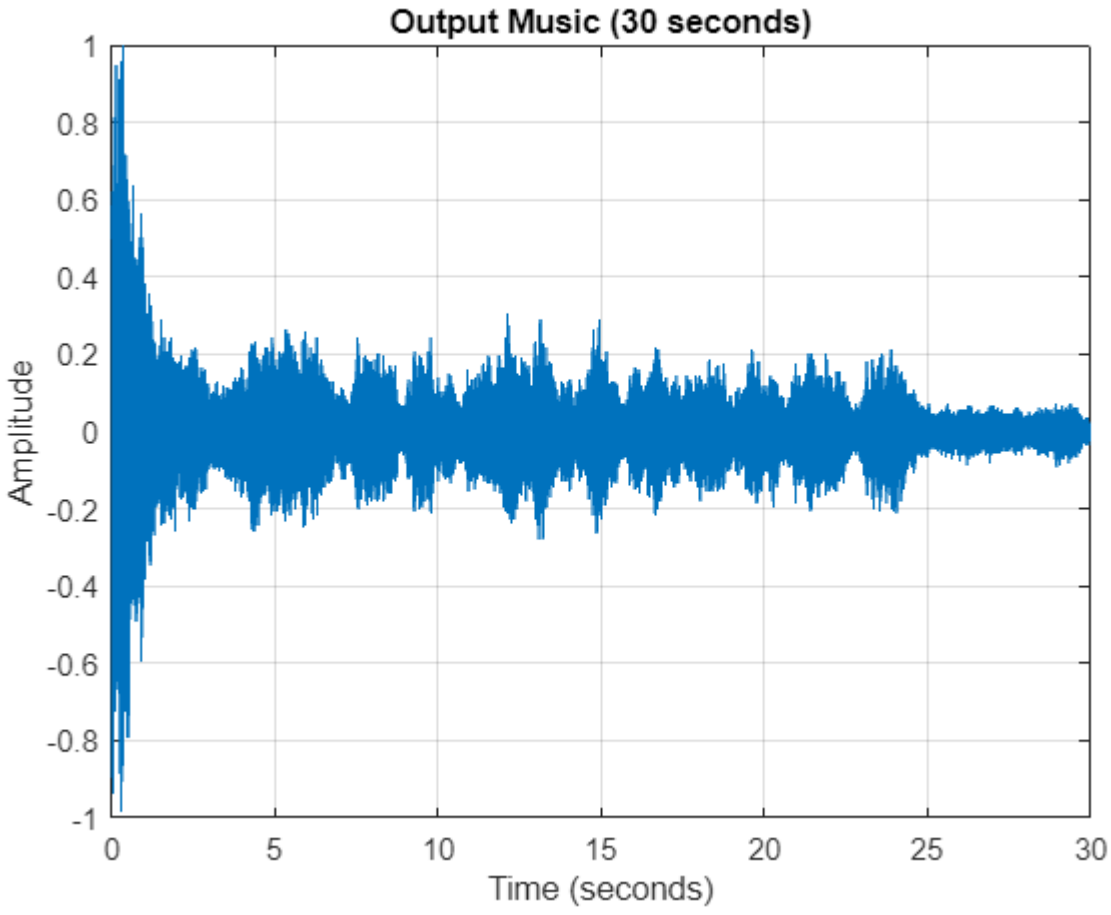

Figure 5: The Output Signal

The input sound does not include any echo since it is recorded in the anechoic room. However, the output sound has echoes and heard as if it is recorded in a music auditorium. This sound is the simulation of the music that will be actually heard at that seat of Odeon when the originally recorded music is played on the stage where the balloon was exploded. This is the case because, as mentioned before, the convolution operation applies the treatment of the environment to the impulse to the input signal.

In the second approach, file of an instrument is picked as input signal and its 30sec is convolved with same impulse response signal with the following code block

```
% File of a instrument
inputFile = 'beethoven_vlaa_6.mp3';  % Try beethoven_tr1_6.mp3 also
[inputMusic, Fs_music] = audioread(inputFile);


% Amplification
amplificationFactor = 5;
inputMusic = inputMusic * amplificationFactor;


% Impulse Response
impulseResponseFile = 'balloon.wav';  % Replace with the correct impulse
response file
[impulseResponse, Fs_imp] = audioread(impulseResponseFile);


% Matching Sampling Rates
if Fs_music ~= Fs_imp
    error('The sampling rates of the input music and the impulse response must
match.');
end


% Extracting 2:00 - 2:30 minutes
startTime = 2 * 60;  % Start at 2:00 minutes
endTime = 2.5 * 60;  % End at 2:30 minutes


% Time to samples
startSample = round(startTime * Fs_music);
endSample = round(endTime * Fs_music);


% Segment of input
inputMusicSegment = inputMusic(startSample:endSample);


% Convolution of segmented part with impulse response
outputMusicSegment = conv(inputMusicSegment, impulseResponse, 'same');  %
'same' ensures output length matches input length


% Clipping Reduce
outputMusicSegment = outputMusicSegment / max(abs(outputMusicSegment(:)));
```

```
% Plot input
t_input = (0:length(inputMusicSegment)-1) / Fs_music;  % Time vector for input
music
figure;
subplot(3,1,1);
plot(t_input, inputMusicSegment, 'r');
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Input Music (Beethovenvlaa6.mp3, 2:00 - 2:30, Amplified)');
grid on;


% Impulse response (Balloon pop)
t_impulse = (0:length(impulseResponse)-1) / Fs_imp;  % Time vector for impulse
response
subplot(3,1,2);
plot(t_impulse, impulseResponse, 'r');
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Impulse Response (Balloon Pop)');
grid on;


% Plot convolved output music segment
t_output = (0:length(outputMusicSegment)-1) / Fs_music;  % Time vector for
output
subplot(3,1,3);  % Third plot
plot(t_output, outputMusicSegment, 'r');
xlabel('Time (seconds)');
ylabel('Amplitude');
title('Output Music (2:00 - 2:30)');
grid on;


% Step 9: Play the original music segment
disp('Playing original input music segment (2:00 - 2:30)...');
sound(inputMusicSegment, Fs_music);
pause(length(inputMusicSegment) / Fs_music + 1);  % Waiting for the original
music to finish playing


% Step 10: Play the convolved output music segment
disp('Playing convolved output music segment (2:00 - 2:30)...');
sound(outputMusicSegment, Fs_music);


% Step 11: (Optional) Save the output segment
audiowrite('outputsingle.wav', outputMusicSegment, Fs_music);
```

During the process, the same sampling rate matching and distortion reduction are applied to the input and output signals. The input signal plot is shown in figure 6.

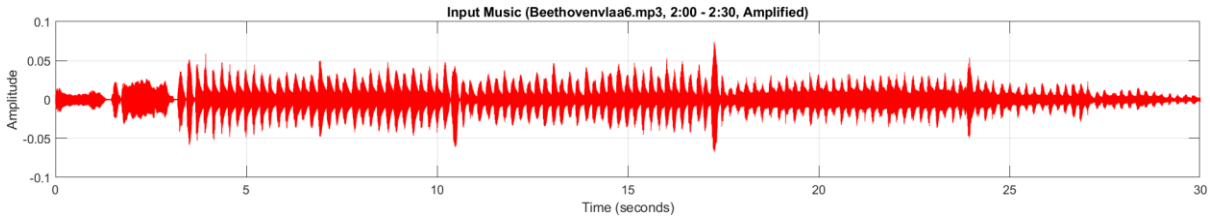

Figure 6: The Input Signal (One Instrument)

The output signal is shown in figure 7.

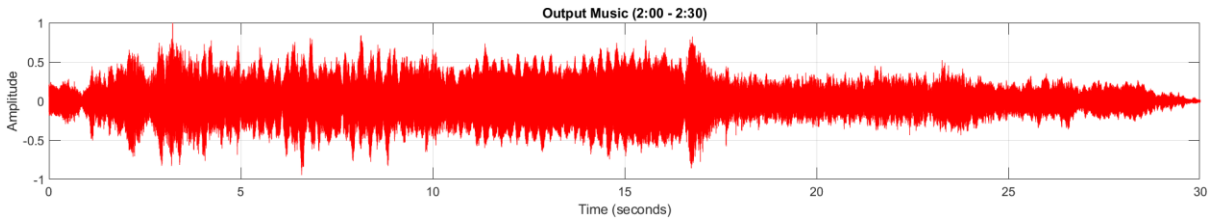

Figure 7: The Output Signal

As can be observed from the above figures, the echo is added to the recorded music in the anechoic room.

Discussion

Ideally, the impulse should be a quick, high energy burst with minimal background noise and clear initial reflections. The balloon pop should be sharp and loud enough to be captured the acoustics of the environment. In this case, the impulse was good enough to conduct the assignment, but it was not ideal since there is background noise and initial reflection might not be pure. However, it is a quite good approximation of an impulse. A well-defined impulse produces a clear impulse response. It can be seen from the plot of the impulse response; the impulse was not ideal but adequate.

As mentioned before, the impulse response reveals the characteristics of the environment, including the presence of early reflections and reverberation. A music auditorium should ideally produce an impulse response with distinct early reflections, followed by a gradually decaying reverberation. This decay gives insight into the room's absorption and reflective qualities, showing how sound energy dissipates over time. A complex impulse response suggests a rich, layered acoustic environment.

Assuming the acoustic environment as a Linear Time-Invariant (LTI) system means that its response to an impulse does not change over time and is linear (i.e., doubling the input doubles the output). In reality, this assumption is valid for most standard, unchanging rooms or halls where the acoustics remain stable over time. However, factors like moving objects, audience presence, and temperature changes could introduce non-linearities. In a controlled environment, the LTI assumption generally holds, but for truly dynamic spaces, it may only be an approximation.

Distortions can occur for various reasons. Low-quality microphones or recording devices may add unwanted artifacts. If the impulse is not sharp or clean, it can introduce distortion into the impulse response. Background noises or unexpected sound reflections can interfere, especially if the room has irregular surfaces or objects that affect the sound path unpredictably. Convolution and normalization may also introduce minor distortions. These distortions can affect the clarity of the final convolved audio, making it sound muddier or adding unintended coloration to the sound.

Noise during the impulse recording can introduce unwanted background hums, hisses, or reverberations that were not part of the intended acoustic environment. This noise becomes embedded in the impulse response,

meaning it will also be applied to the input signal during convolution. As a result, the final audio may sound less clean and more “noisy” than intended, detracting from the clarity and realism of the simulated acoustic experience.

The acoustical structure of the listening environment, including room size, shape, material surfaces, and seating arrangement, profoundly impacts listening quality. A well-designed concert hall with good acoustics enhances the clarity, warmth, and richness of the sound. Early reflections contribute to a sense of fullness, while controlled reverberation adds depth without blurring details. However, poorly designed acoustics can lead to excessive reverberation, muddiness, or “dead spots” where certain frequencies are diminished, making the listening experience less enjoyable.

In my opinion, anechoic music sounds unnatural and dry. The simulated environment adds warmth, depth, and sense of space which is more engaging. I would prefer acoustic environment.

Conclusion

In this assignment, we explored the process of capturing an impulse response in an acoustic environment and applying it to an anechoic music recording through convolution. By recording a balloon pop as the impulse, we simulated the effect of a concert hall on clean, echo-free audio, transforming it to sound as though it was performed in that space. The MATLAB code provided a practical approach to manipulating and analyzing audio data, enhancing our understanding of how environments shape sound perception. This experiment emphasized the importance of impulse response quality, highlighted the assumptions of linear time-invariant systems, and demonstrated the impact of acoustics on listening experience, ultimately enriching our appreciation of spatial sound design.

REFERENCES

[1] J. Pätynen, V. Pulkki, and T. Lokki, "Anechoic recording system for symphony orchestra," *Acta Acustica united with Acustica*, vol. 94, no. 6, pp. 856-865, Nov./Dec. 2008. [Online]. Available: IngentaConnect