Computing Assignment 1

EE229

SIGNAL PROCESSING

Hemang Dave – 23B3990

Jay Patel – 23B3983

Objective: To generate a convolved audio signal output by processing a given audio file with multiple impulse responses, which are given as audio files.

The input audio and impulse response files are discrete-time signal files with 16,000 samples recorded per second.

We have used Scilab to implement the convolution

Outline of the code:

1. We first read the input audio file and the impulse response into the *inp* and *rir* variables, respectively, using the wavread() function present in Scilab.
2. Next, we define our convolution function, fun_con(inp1,rir1), where inp1 and rir1 are the input audio and room impulse response parameters, respectively.
   In the convolution function, we first calculate the length of the audio file and the impulse response and store it in the inp_len and rir_len variables, respectively. The length of the output file will be as follows:

   $$Out\_len = inp\_len + rir\_len - 1$$

   We initialize the output variable as an array of zeroes of specified output length.
   Now, we have to update the values in the output array. For this, we will run a for loop.
   We know that for the convolution of a discrete-time signal,

   $$y[n] = \sum h[k]*x[n-k]$$

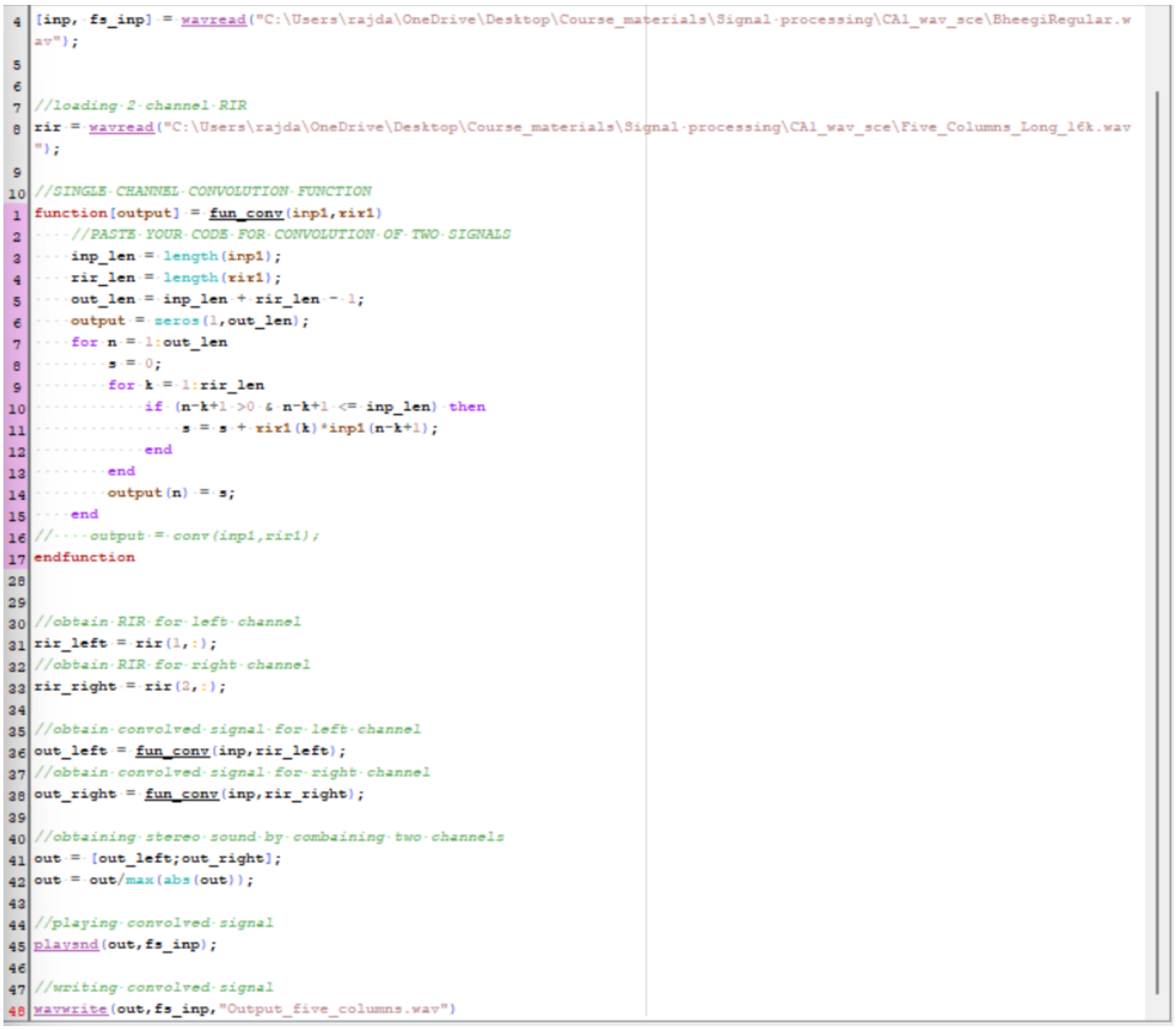
   where k ranges from -∞ to + ∞

   In the code below, we initialize a variable s, which is y[n].
   Next, we run a for loop over the impulse response function h[k]. The value k ranges from 1 to rir_len. Before implementing the summation for the convolution, we need to check if the input audio signal is non-zero for inp1(n-k) and that inp1(n-k) does not exceed the length of the input file.
   Now we implemented the summation function for y[n] given by,

   s = s + rir1(k)*inp(n-k+1)

   An additional factor of 1 exists in the formula because array numbering in Scilab starts from 1 and not from 0.
   Once we have calculated the value of s, we assign the value of y[n] to be equal to s.
3. Our convolution function is now ready. The code that follows obtains the impulse response for the left and right audio channels from the input rir file. We then perform the convolution on the left and right audio channels and then finally combine them together and play the sound using the playsnd() function. The audio file generated is saved into the folder using the wavwrite(function).

Code for convolution



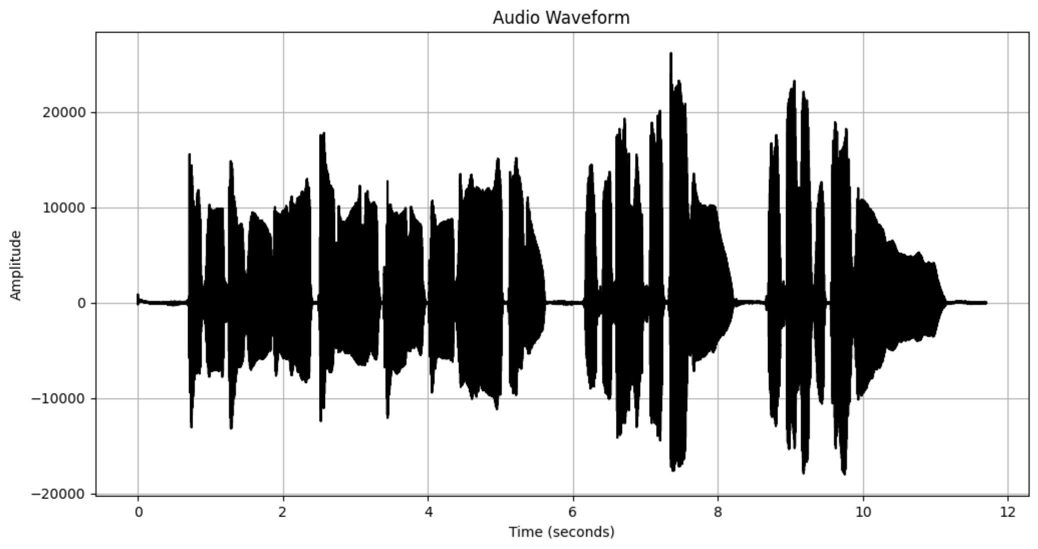
```
[inp, fs_inp] = wavread("C:\Users\rajda\OneDrive\Desktop\Course_materials\Signal processing\CA1_wav_sce\BheegiRegular.wav");


//loading 2 channel RIR
rir = wavread("C:\Users\rajda\OneDrive\Desktop\Course_materials\Signal processing\CA1_wav_sce\Five_Columns_Long_16k.wav");

//SINGLE CHANNEL CONVOLUTION FUNCTION
function[output] = fun_conv(inp1,rir1)
    //PASTE YOUR CODE FOR CONVOLUTION OF TWO SIGNALS
    inp_len = length(inp1);
    rir_len = length(rir1);
    out_len = inp_len + rir_len - 1;
    output = zeros(1,out_len);
    for n = 1:out_len
        s = 0;
        for k = 1:rir_len
            if (n-k+1 >0 & n-k+1 <= inp_len) then
                s = s + rir1(k)*inp1(n-k+1);
            end
        end
        output(n) = s;
    end
//    output = conv(inp1,rir1);
endfunction


//obtain RIR for left channel
rir_left = rir(1,:);
//obtain RIR for right channel
rir_right = rir(2,:);

//obtain convolved signal for left channel
out_left = fun_conv(inp,rir_left);
//obtain convolved signal for right channel
out_right = fun_conv(inp,rir_right);

//obtaining stereo sound by combaining two channels
out = [out_left;out_right];
out = out/max(abs(out));

//playing convolved signal
playsnd(out,fs_inp);

//writing convolved signal
wavwrite(out,fs_inp,"Output_five_columns.wav")
```

However, the given code is not optimized and is computationally intensive. The code took approximately half an hour even to generate an output file of 1 second!

Therefore, to properly document the results of the convolution, we used the built-in conv() function present in Scilab to generate the output files, which uses the FFT (Fast Fourier Transform) technique to speed up the convolution process.

Input audio file:
https://drive.google.com/file/d/1Y6lcVuUWmkwgaLb632YLudpJn00_PEJS/view?usp=sharing

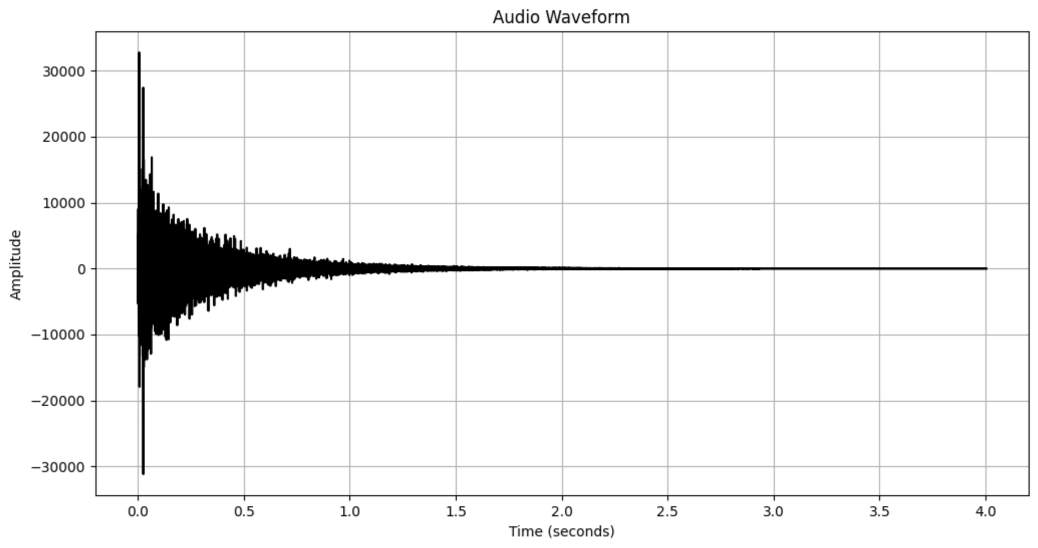
Input audio waveform:



Impulse response 1: Five long columns

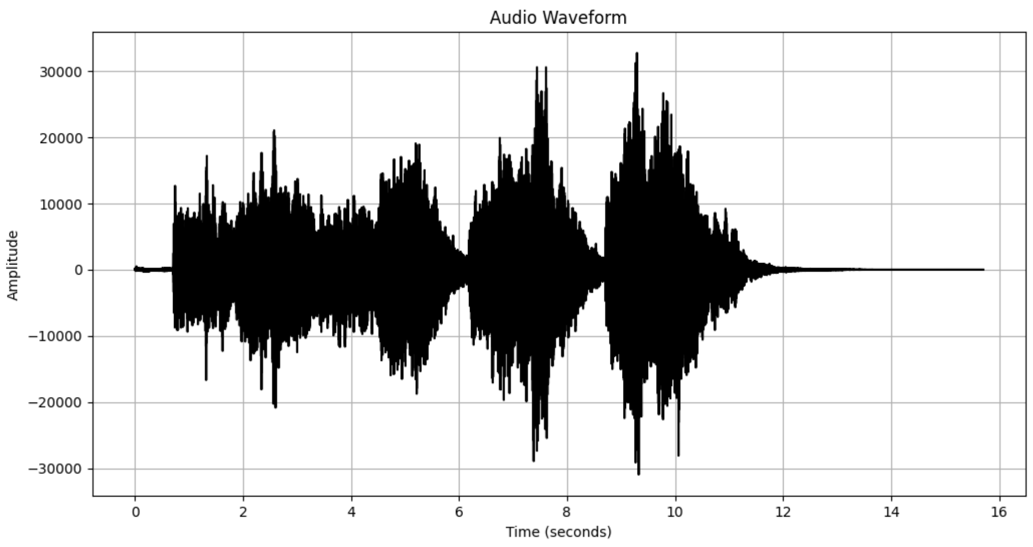
Impulse response audio file:
https://drive.google.com/file/d/12PO0GP4_AjiUJonKu1YWAoZSV1PIaxlc/view?usp=sharing

Impulse response audio waveform:



Convolved output: https://drive.google.com/file/d/1nQvEqwh_umI_uLJ657E-pgkrUJuMeAII/view?usp=sharing

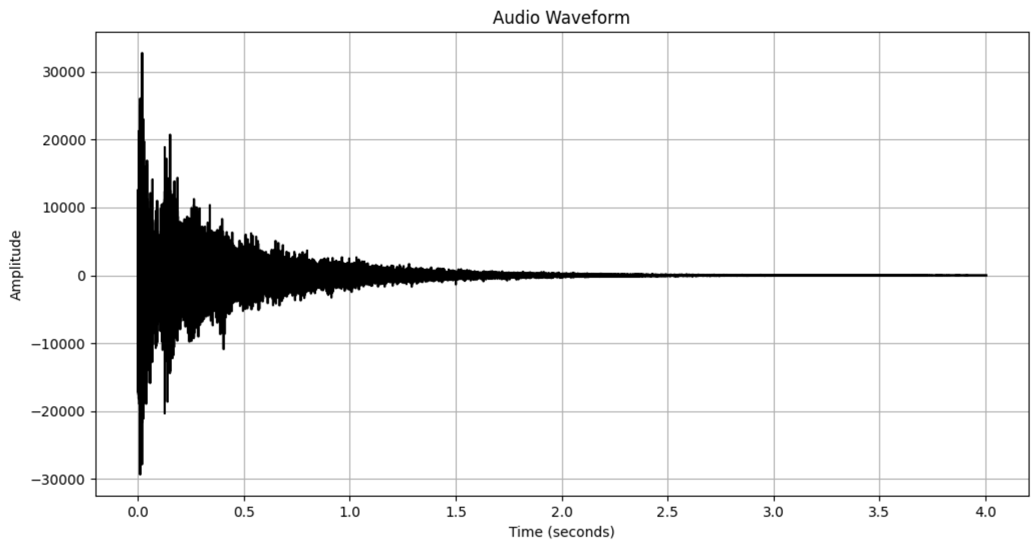
Convolved output audio waveform:



Impulse response 2: Long Hall echo

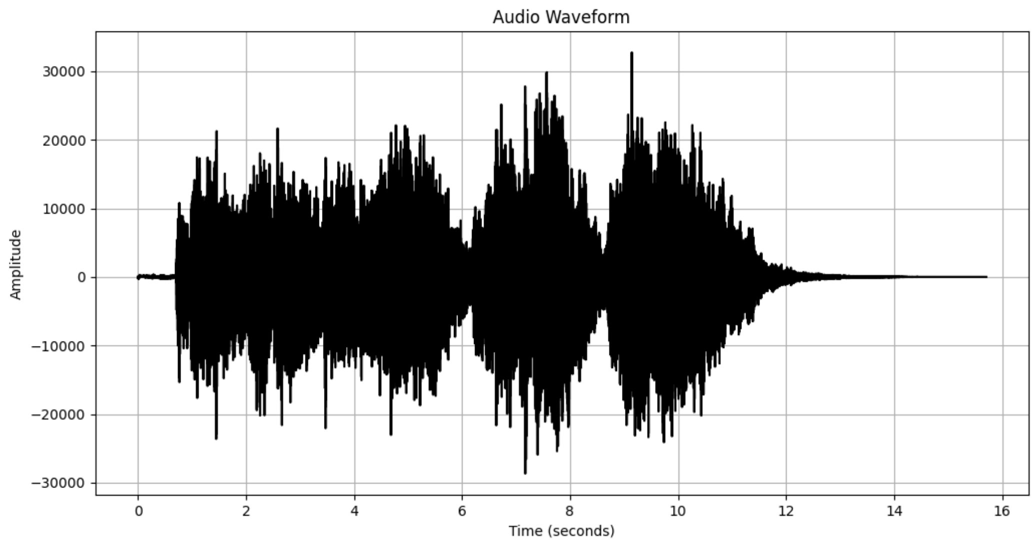
Impulse response file: https://drive.google.com/file/d/1sD_tsjf2M7Qvgo-wmaPnHzRFGfmWH3Eo/view?usp=sharing

Impulse response waveform:



Convolved output file: https://drive.google.com/file/d/1gtPFiinmDk08G3bFKF2AbYPBHZiEZ3rH/view?usp=sharing

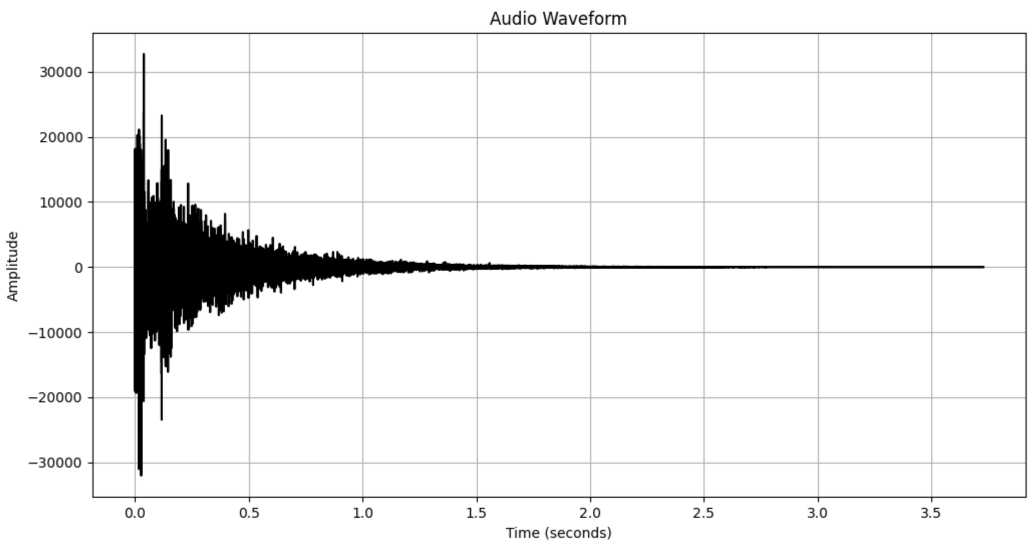
Convolved output waveform:



Impulse Response 3: Parking Garage

Impulse response file:
https://drive.google.com/file/d/1gtPFiinmDk08G3bFKF2AbYPBHZiEZ3rH/view?usp=shari
ng

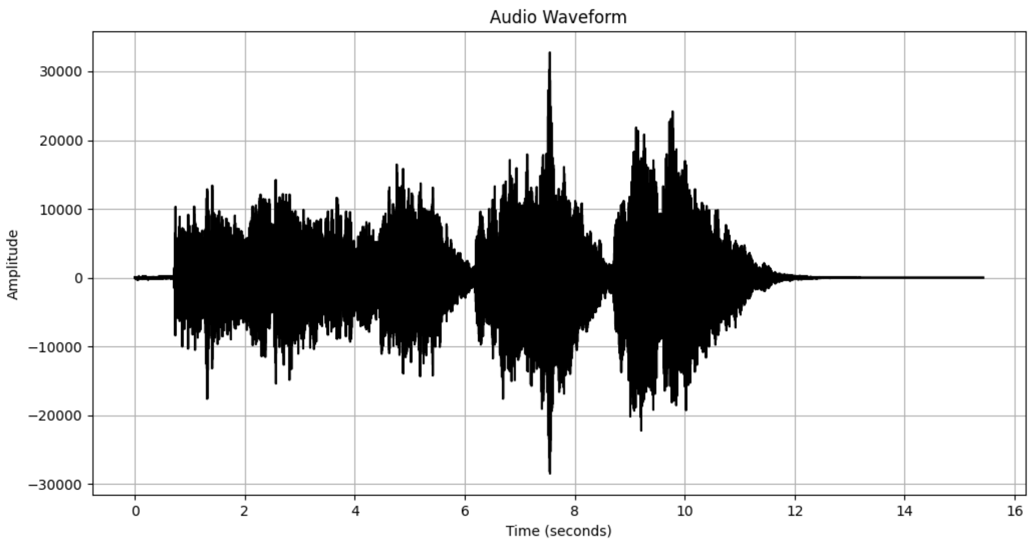
Impulse response waveform:



Convolved output file:

https://drive.google.com/file/d/1Z206m6LkKRLx-
02O1TJkANzR7xGn6Uh1/view?usp=sharing

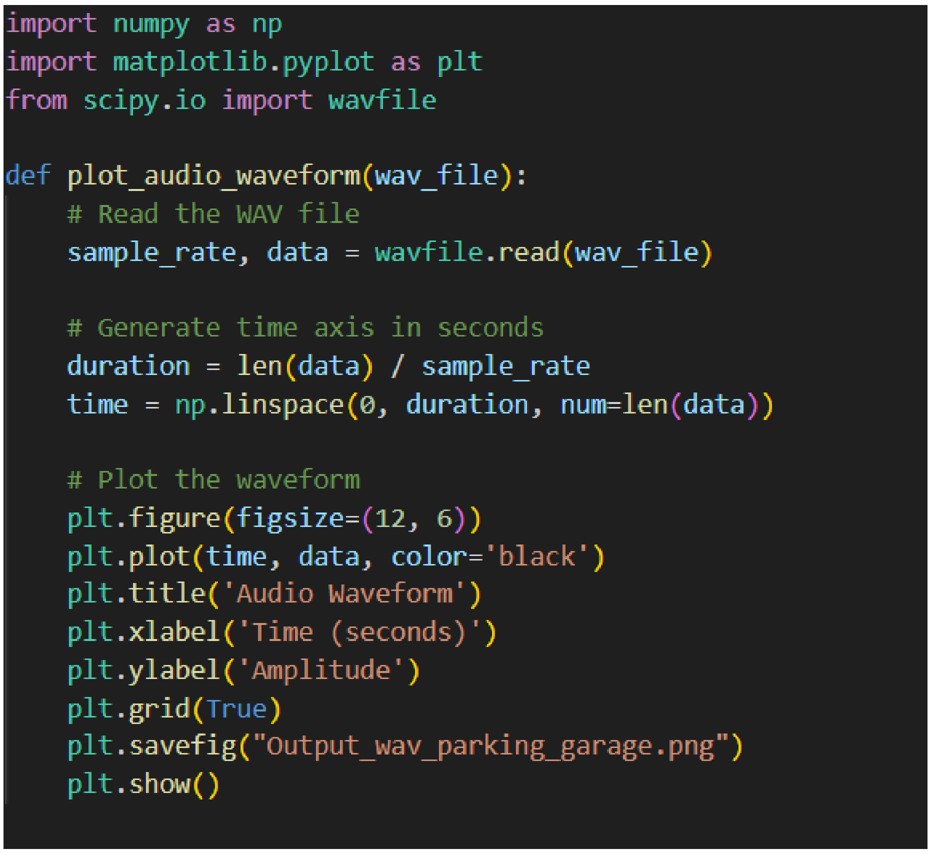
Convolved output waveform:



General observations

1. By observing the audio waveforms, we can see that the input waveform is about 11 to 12 seconds long, and after convolving it with various impulse responses, we get convolved outputs of varying lengths.
2. The impulse response for the long hall echo is the longest due to multiple reverberations for a given impulse. The convolved output is also the longest in this case.
3. We can clearly hear the differences in the convolved audio files, visually compare their waveforms, and compare the various features of the output audio samples.

To plot the audio waveforms the following python code was used:



```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import wavfile

def plot_audio_waveform(wav_file):
    # Read the WAV file
    sample_rate, data = wavfile.read(wav_file)

    # Generate time axis in seconds
    duration = len(data) / sample_rate
    time = np.linspace(0, duration, num=len(data))

    # Plot the waveform
    plt.figure(figsize=(12, 6))
    plt.plot(time, data, color='black')
    plt.title('Audio Waveform')
    plt.xlabel('Time (seconds)')
    plt.ylabel('Amplitude')
    plt.grid(True)
    plt.savefig("Output_wav_parking_garage.png")
    plt.show()
```

Link for Python code file:
https://drive.google.com/file/d/1rzVE1NI8fOXzoX6qPi4FnkPnvGSmJ7Vm/view?usp=sharing

Contributions:

After receiving the assignment, we did some background learning from the internet and learned some basic syntaxes of Scilab. Once we got a basic grasp of Scilab, we independently wrote the convolution functions and compared them.

After writing the final convolution function, we ran the code to generate a one-second output clip using the convolution function we had written by forcefully setting the output length to 16,000 samples.

Link for one-second clip:
https://drive.google.com/file/d/1Q0yOrttqt4n3Vx6R0aN8skLvh4trS6xV/view?usp=sharing

This shows that our convolution function is correct (however, not optimized).

Next, we generated the outputs for the various impulse responses using the conv() function. These convolutions were computed on Jay Patel’s system, and we both contributed to it.

The Python code to generate the output waveforms was executed on Hemang Dave’s system

We both made equal contributions to the making of this report.

From this assignment, we learned and practically wrote a functioning convolution program to generate output audio files for a given input file and an impulse response. We got to hear and visually compare various output waveforms and observed the effect of each impulse response when convolved with a given input audio file. This assignment helped us to get a better grasp on the concepts and fundamentals of convolution.