# AUDIO EFFECTS SYSTEM

## DSP Lab Project based on Audio Effects System


Abstract

The objective of the Lab project is to implement an audio effects system using the Python and Pspice environment. The audio effects system should provide various options including:

* Load in a wav file/Plot/Play File, decimate to reduce data overhead
* Basic Delay, Multiple Delays using IIR Filters



Talha Tallat - D18124645

D18124645@mytudublin.ie

# Table of Contents

# 1. Basic signal echo filter

## 1.1 Description

The 1st order basic echo filter (FIR) is a simple echo filter, that adds a copy of the input to itself dT seconds later, where d is the number of samples in the delay used as shown in the block diagram.

## 1.2 Block diagram

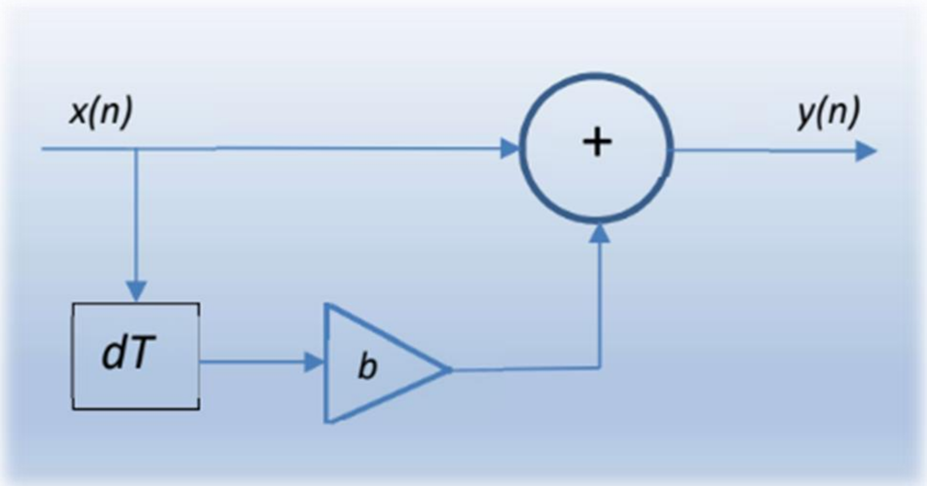



*Figure 1 - Block diagram of a 1st order basic echo filter (FIR)*

## 1.3 Equations

$$y(n) = x(n) + \; bx(n-d)$$
$$=> \; Y(z) = X(z) + \; bX(z)z^{-d}$$
$$=> \; H(z) = \frac{Y(z)}{X(z)} = 1 + bz^{-d}$$

## 1.4 Python code of the single echo filter

**Python code of the echo signal:**

```
import numpy as np
import scipy.fftpack
from scipy.signal import lfilter
import matplotlib.pyplot as plt
from scipy.io import wavfile
# --------------------creating an input signal--------------------
plt.close("all")
fs=8000
b=np.array([1,0,0,0,0,0,0,0,0,0.8])
a,N=1,128
ip=np.zeros(N);
ip[0]=1; #apply impulse to input
op=lfilter(b,a,ip)
plt.plot(op)
plt.grid()
plt.figure(1)
plt.title('Impulse Response')
plt.xlabel('n')
```

```
plt.ylabel('h(n)')
plt.show()
NFFT=1024 # No. of values in FFT
M = 2*np.abs(scipy.fftpack.fft(op,NFFT))/N
M = M[0:int(NFFT/2)] #slicing operation to avoid mirroring
freq = np.arange(0,NFFT/2) #frequency vector
freq = freq*fs/NFFT
plt.figure(2)
plt.plot(freq,M)
plt.title('Spectrum of Single Echo Filter (Delay d=8)')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (V)')
plt.grid()
plt.show()
```

The impulse response and the frequency response of the echo filter are shown below for delays of 8 samples respectively.

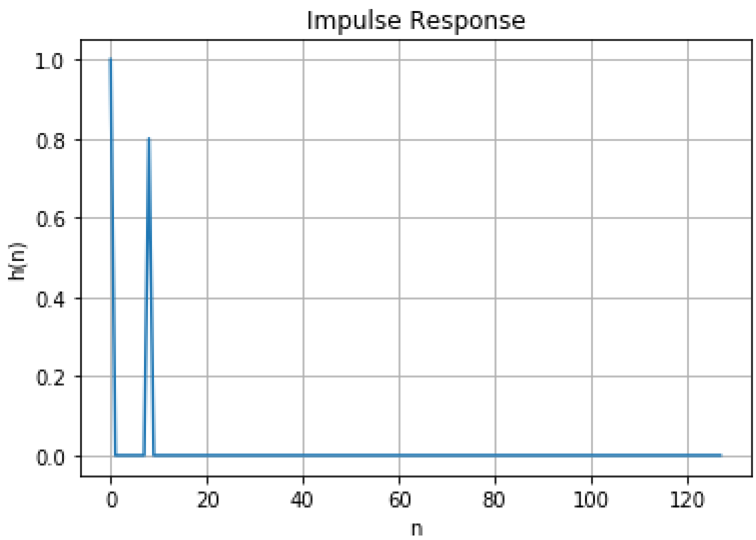



*Figure 2 - Impulse response of the echo filter with 8 samples*

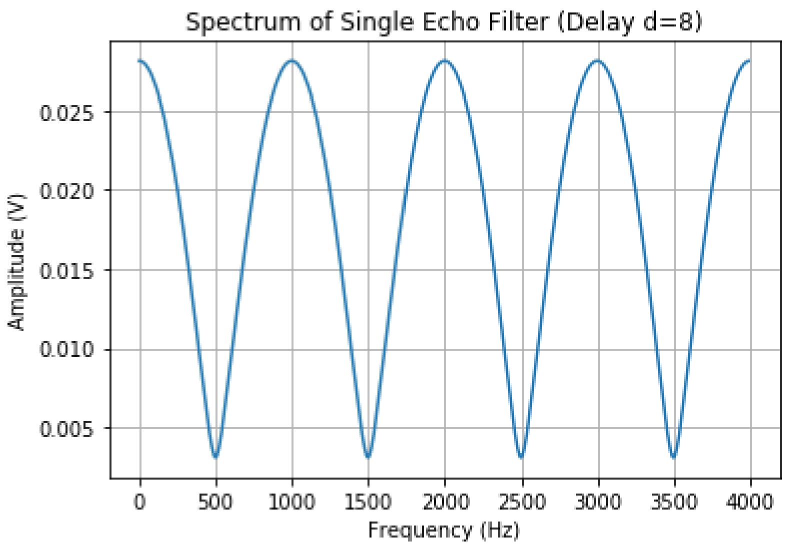



*Figure 3 - Frequency response of the echo filter with the 8 samples*

Verifying that the above impulse response and frequency response are obtained, using the plt.stem function to plot the impulse response for 11 samples of the output.

```
plt.stem(op[0:11])
plt.grid()
plt.figure(1)
plt.title('Impulse Response')
plt.xlabel('n')
plt.ylabel('h(n)')
plt.show()
```

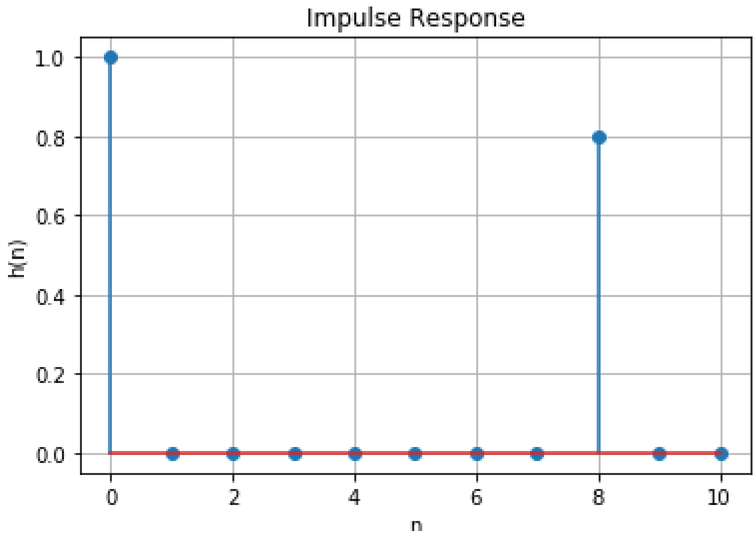



*Figure 4 - Impulse response of the echo filter with 8 samples*

?

?

## 1.5 Implementation of the basic signal echo filter in Pspice

Now investigate the implementation of both above filters in Pspice. Compare the results obtained with the Python system.

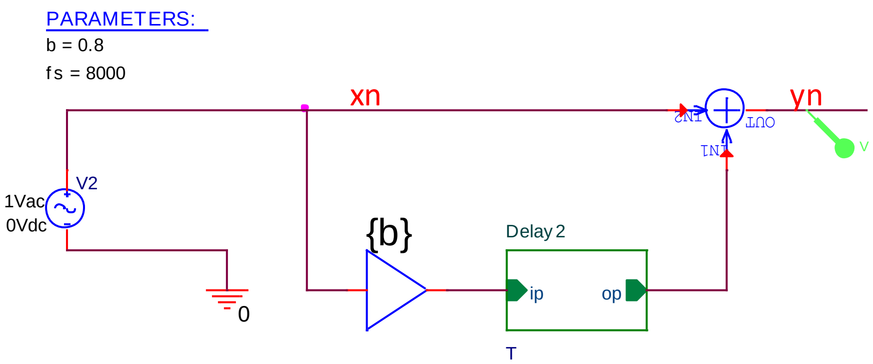



*Figure 5 - schematic diagram of the basic echo filter with 8 samples*

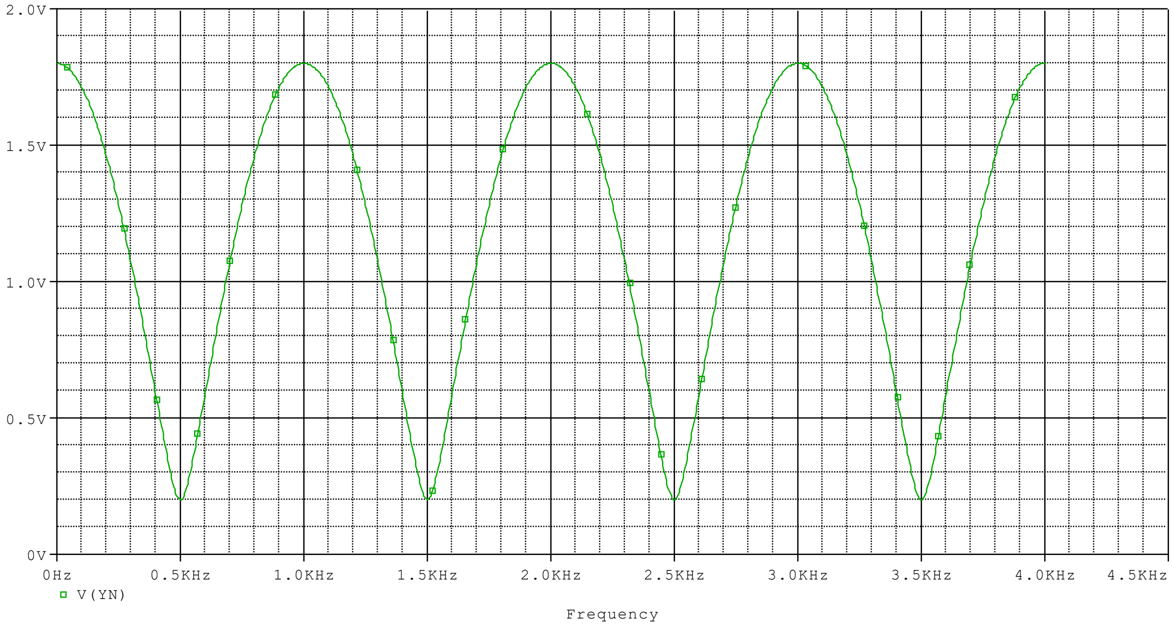



*Figure 6 - Frequency response of the basic echo filter with 8 samples*

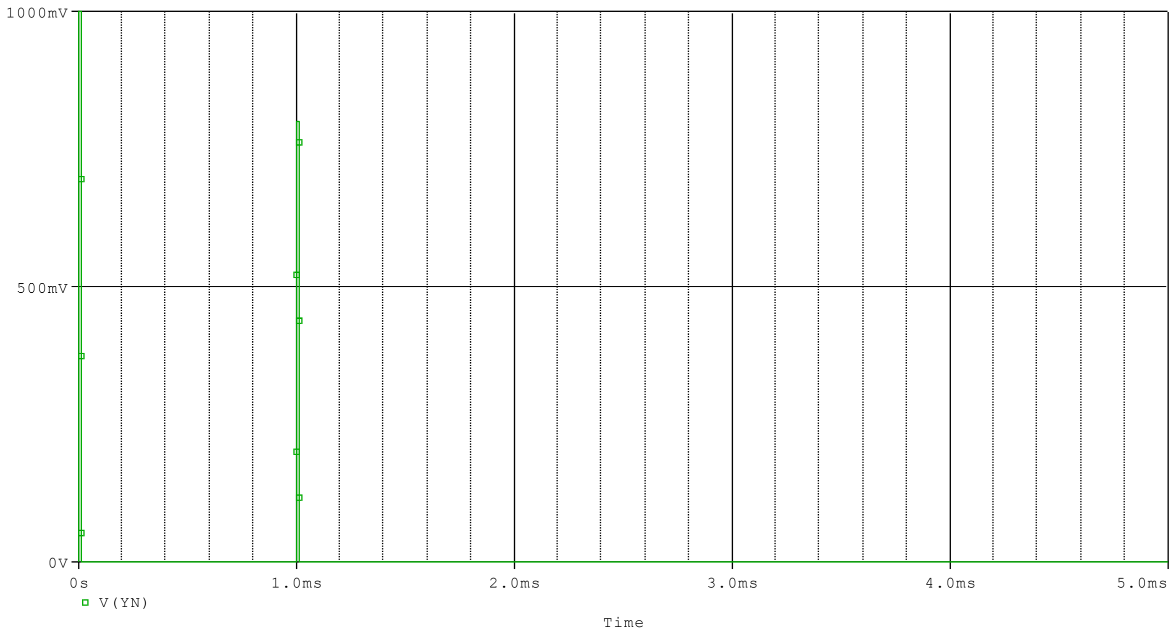



*Figure 7 - Impulse response of the echo filter with 8 samples (TD=8/fs)*

Compare?????

??

??

## 2. Python Impulse and Frequency response of 16 sample delay of signal echo filter

**Modifying the code to give a delay of 16 samples and verifying the following results:**

The array contains 16 delays to create an impulse response of 16 samples.

```
b=np.array([1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0.8])
a,N=1,128
ip=np.zeros(N);
ip[0]=1; #apply impulse to input
op=lfilter(b,a,ip)
plt.plot(op)
plt.grid()
plt.title('Impulse Response')
plt.xlabel('n')
plt.ylabel('h(n)')
plt.show()

NFFT=1024 # No. of values in FFT
M = 2*np.abs(scipy.fftpack.fft(op,NFFT))/N
M = M[0:int(NFFT/2)] #slicing operation to avoid mirroring
freq = np.arange(0,NFFT/2) #frequency vector
freq = freq*fs/NFFT
plt.plot(freq,M)
plt.title('Spectrum of Single Echo Filter (Delay d=8)')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (V)')
plt.grid()
plt.show()

plt.stem(op[0:22])
plt.grid()
plt.title('Impulse Response stem plot')
plt.xlabel('n')
plt.ylabel('h(n)')
plt.show()
```

Modifying the code to give a delay of 16 samples and verifying the following results:

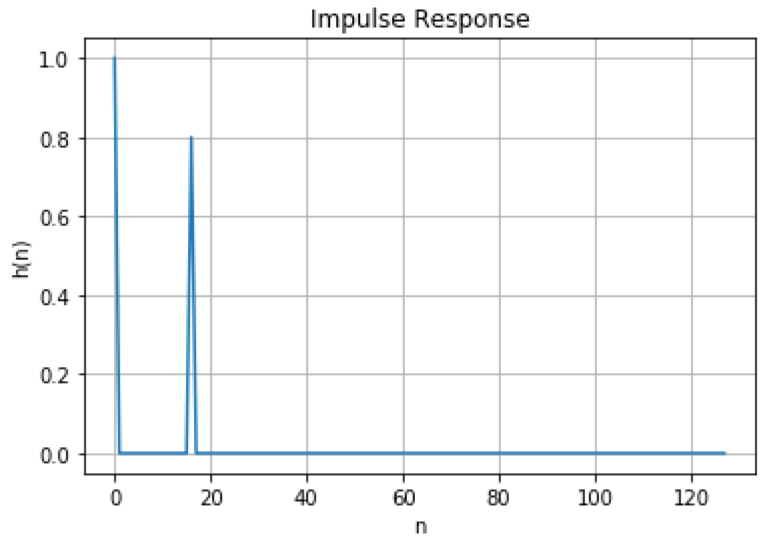



*Figure 8 - Impulse response of the echo filter with the delay of 16 samples*

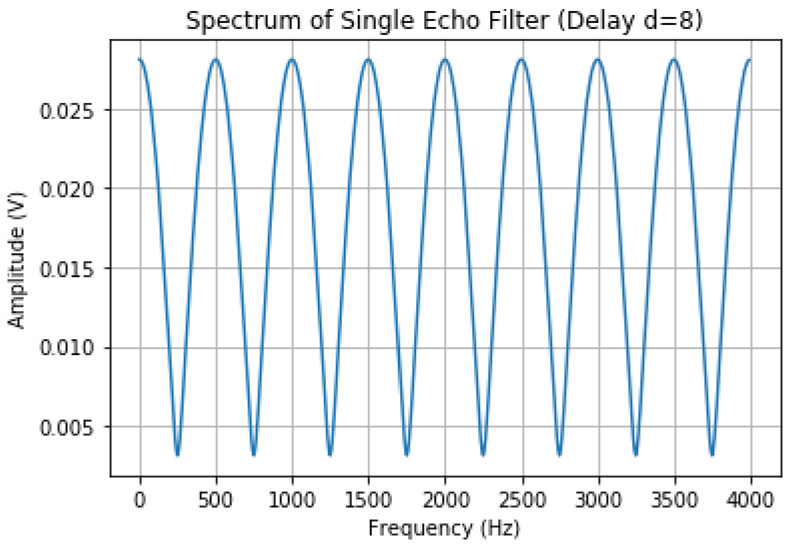



*Figure 9 - Frequency response of the echo filter with the delay of 16 samples*

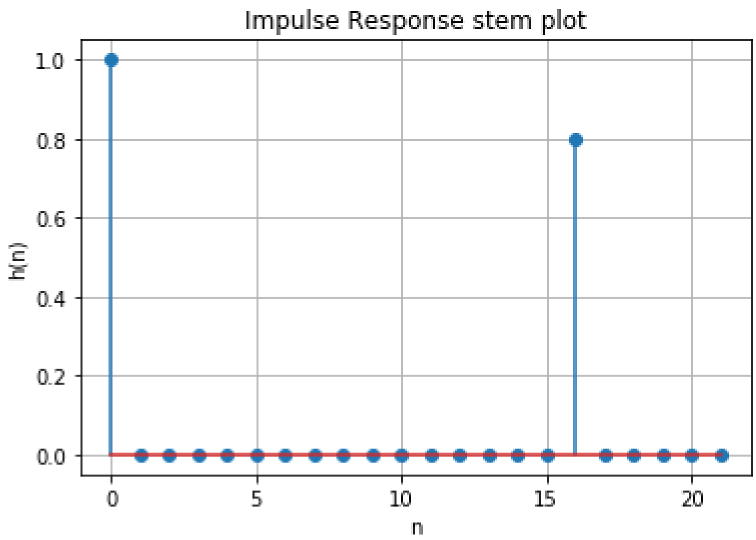



*Figure 10 - Impulse response of the echo filter with the delay of 16 samples*

## 2.1 Description

The Derivation is shown in figure 11, where the peaks in the frequency response occur at multiples of 2π/d where d represents the number of samples that the signal has been delayed. Give a brief discussion of the results so far.

Figure 6 shows the frequency response of the echo filter with 16 samples and this filter is called a comb filter because its frequency response resembles a comb.

$$H(\theta) = 1 + be^{-j\theta d} = 1 + b\{\cos(\theta d) - j\sin(\theta d)\}$$
$$|H(\theta)|^2 = \{1 + b\cos(\theta d)\}^2 + \{b\sin(\theta d)\}^2 = 1 + 2b\cos(\theta d) + b^2\{\cos^2(\theta d) + \sin^2(\theta d)\}$$
$$|H(\theta)|^2 = 1 + 2b\cos(\theta d) + b^2$$
$$|H(\theta)|^2_{\max} = 1 + 2b + b^2 = (1+b)^2 \Rightarrow |H(\theta)|_{\max} = (1+b)$$
$$|H(\theta)|^2_{\min} = 1 - 2b + b^2 = (1-b)^2 \Rightarrow |H(\theta)|_{\min} = (1-b)$$
$$|H(\theta)|_{\max} \; @ \quad \cos(\theta d) = 1 \quad \Rightarrow \quad \theta d = 2n\pi \Rightarrow \theta = \frac{2n\pi}{d}$$
$$If\ d = 8,\ \theta = \frac{2n\pi}{8} = \frac{n\pi}{4} = n \times 1\ kHz\ \ for\ f_s = 8\ kHz$$

*Figure 11 - Derivation for the comb filter*

## 2.2 Implementation of the comb filter in Pspice

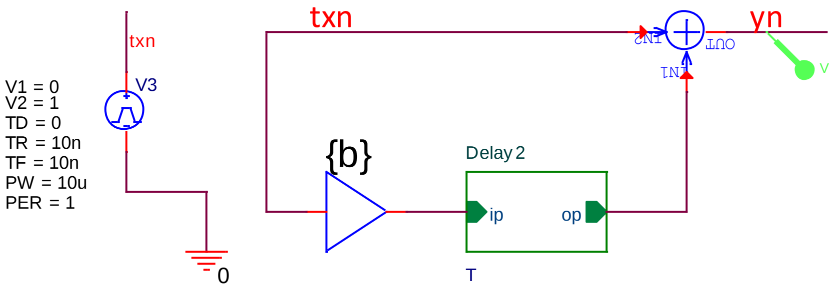



*Figure 12 - schematic diagram of the basic echo filter with 16 samples*

?

?

Frequency response ???

?

?

Details??

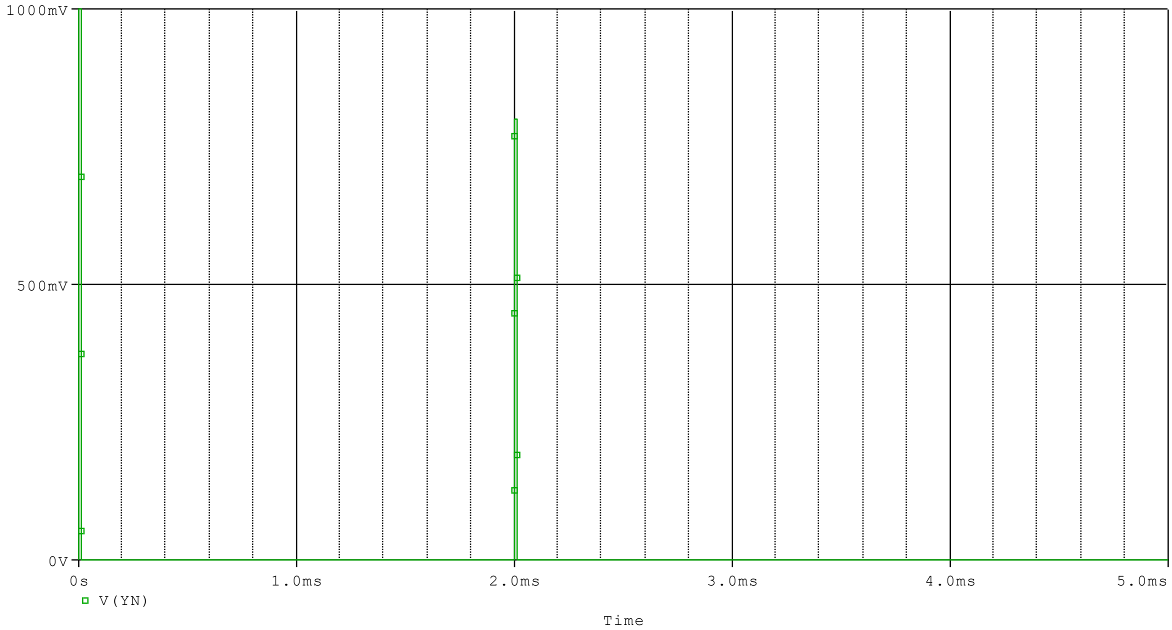



*Figure 13 - Impulse response of the echo filter with 16 samples (TD=16/fs)*

# 3. Multiple Echo filter

## 3.1 Block diagram

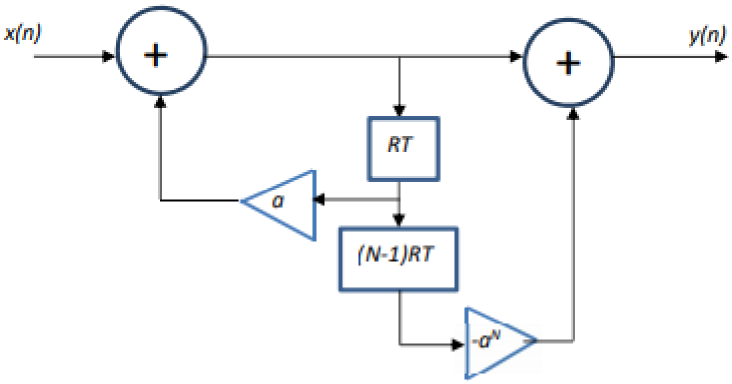



*Figure 14 - Block diagram of the echo filter*

The multiple echo filter adds N-1 delays reducing in amplitude for each delay (a<1). The objective is to realise the following transfer function:

## 3.2 Equations

$$H(z) = \frac{1 - a^N z^{-NR}}{1 - az^{-R}} = 1 + az^{-R} + a^2z^{-2R} + a^3z^{-3R} \pm \cdots + a^{N-1}z^{-(N-1)R}$$

The equations can be written into a more compact form.

$$H(z) = \frac{Y(z)}{X(z)} = 1 + az^{-R} + a^2z^{-2R} + a^3z^{-3R} \pm \cdots + a^{N-1}z^{-(N-1)R}$$

$$az^{-R}\,H(z) = az^{-R} + a^2z^{-2R} + a^3z^{-3R} \pm \cdots + a^N z^{-NR}$$

$$H(z) - az^{-R}\,H(z) = 1 + az^{-R} + a^2z^{-2R} + a^3z^{-3R} \pm \cdots + \,a^{N-1}z^{-(N-1)R} - \{az^{-R} + a^2z^{-2R} + a^3z^{-3R} \pm \cdots + \,a^Nz^{-NR}\}$$

$$H(z)(1 - az^{-R}) = 1 - a^Nz^{-NR}$$

$$H(z) = \frac{1 - a^Nz^{-NR}}{1 - az^{-R}}$$

Verifying that the above block diagram in figure 9 does produce the compact transfer function derived above. Letting the output from the 1st summer be defined as g(n):

$$G(z) = \,X(z) + aG(z)\,z^{-R}$$

$$G(z) - aG(z)z^{-R} = \,X(z)$$

$$X(z) = \,G(z)\,\{1 - az^{-R}\}$$

$$Y(z) = G(z) - a^NG(z)z^{-NR} = \,G(z)\{1 - a^Nz^{-NR}\}$$

$$Y(z) = \frac{X(z)}{\{1 - az^{-R}\}}\{1 - a^Nz^{-NR}\}$$

$$H(z) = \frac{X(z)}{X(z)} = \,az^{-R} + a^2z^{-2R} + a^3z^{-3R} \pm \cdots + a^Nz^{-NR} = \frac{1 - a^Nz^{-NR}}{1 - az^{-R}}$$

## 3.3 Description

Implementing the multiple echo filter in Python & the parameters are set to N = 6, R = 4, alpha = 0.8.

**The code for Multiple echo filter:**

```
N=6; R=4; alpha=0.8;
b=np.zeros(N*R+1);
b[N*R]=-alpha**N; b[0]=1;
a=np.zeros(R+1);
a[R]=-alpha; a[0]=1;
ip=np.zeros(1024);
ip[0]=1;
op=lfilter(b,a,ip);
plt.stem(op[1:40])
plt.title('Impulse Response of Multiple Echo Filter (N=6, R=4, alpha=0.8)')
plt.xlabel('Frequency (Hz)')
plt.ylabel('Amplitude (V)')
plt.grid()
plt.show()
NFFT=1024 # No. of values in FFT
M = 2*np.abs(scipy.fftpack.fft(op,NFFT))/N
M = M[0:int(NFFT/2)] #slicing operation to avoid mirroring
freq = np.arange(0,NFFT/2) #frequency vector
freq = freq*fs/NFFT
plt.plot(freq,M)
plt.title ('Multiple Echo Filter Frequency Response');
plt.xlabel ('Frequency (Hz)');
plt.ylabel ('Magnitude (dB)');
```

```
plt.grid()
plt.show()
```

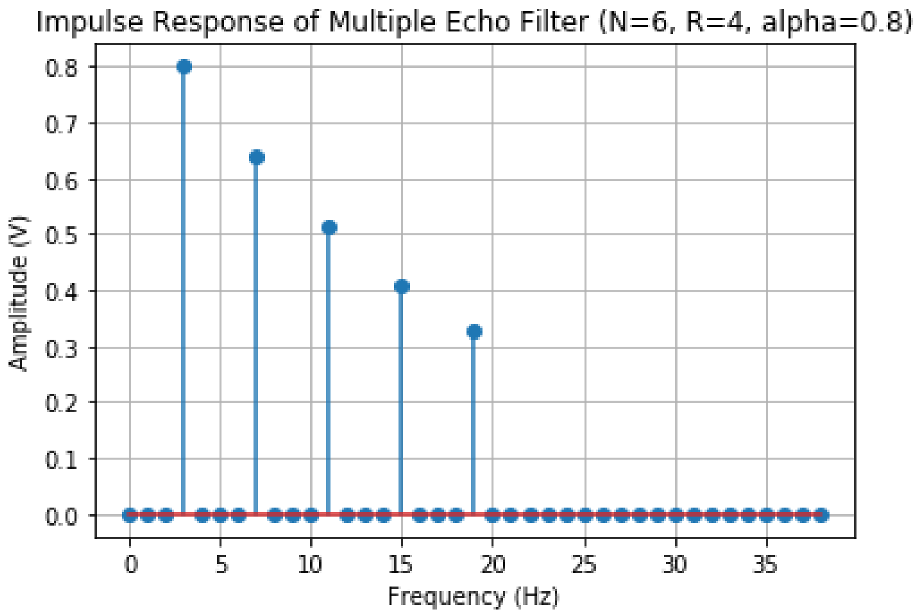



*Figure 15 - Impulse Response of the Multiple Echo filter (N=6, R=4, alpha=0.8)*

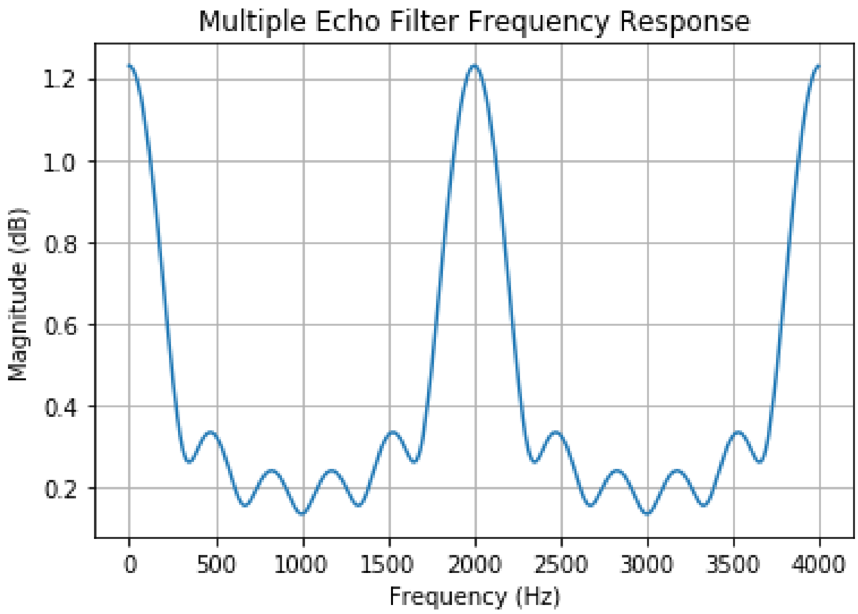



*Figure 16 - Frequency response of the Multiple Echo filter*

Provide a brief discussion of your results obtained in part 1?

?

?

?

## 3.4 Implementation of the multiple echo filter in Pspice

Implementing the multiple echo filter in Pspice.

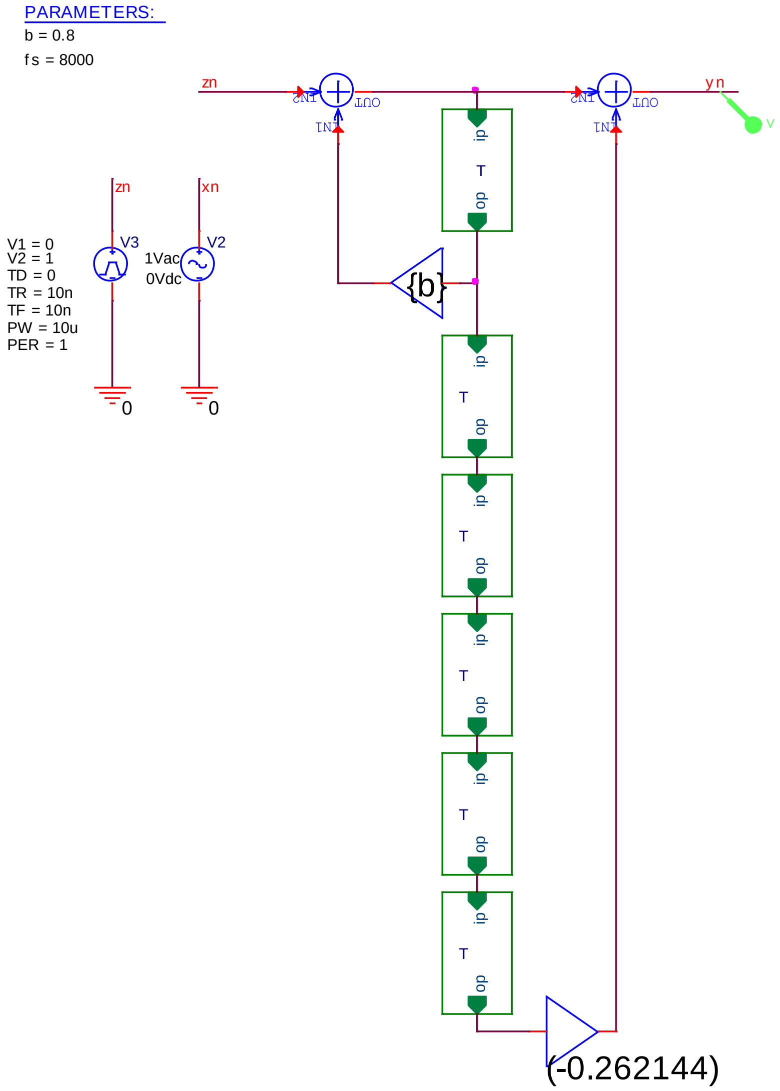



*Figure 17 - Schematic diagram of the Multiple echo filter*

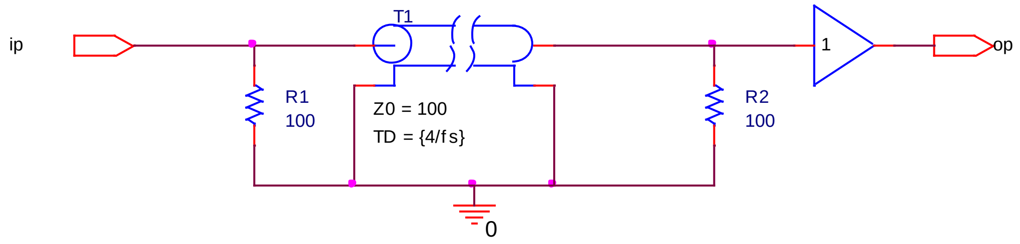



*Figure 18 - Schematic diagram of the time delay (T)*

Time-domain runtime is set to 10ms and maximum step size is set to 1us.

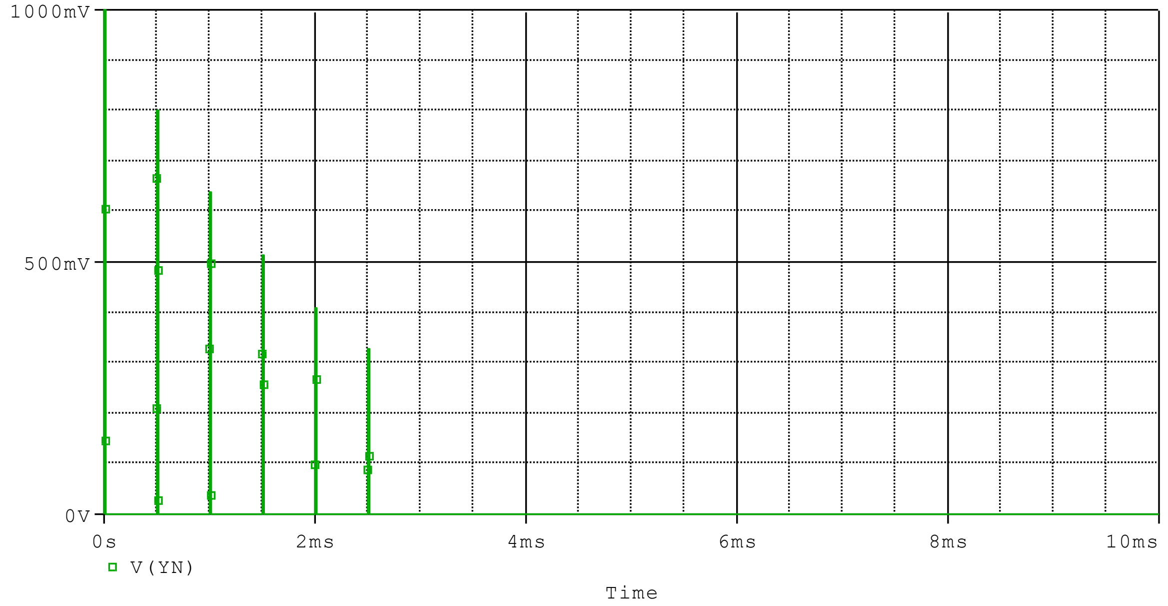



*Figure 19 - Impulse Response of the Multiple Echo filter in Pspice*

Linear AC sweep frequency starts at 1 Hz and ends at 4000 Hz.

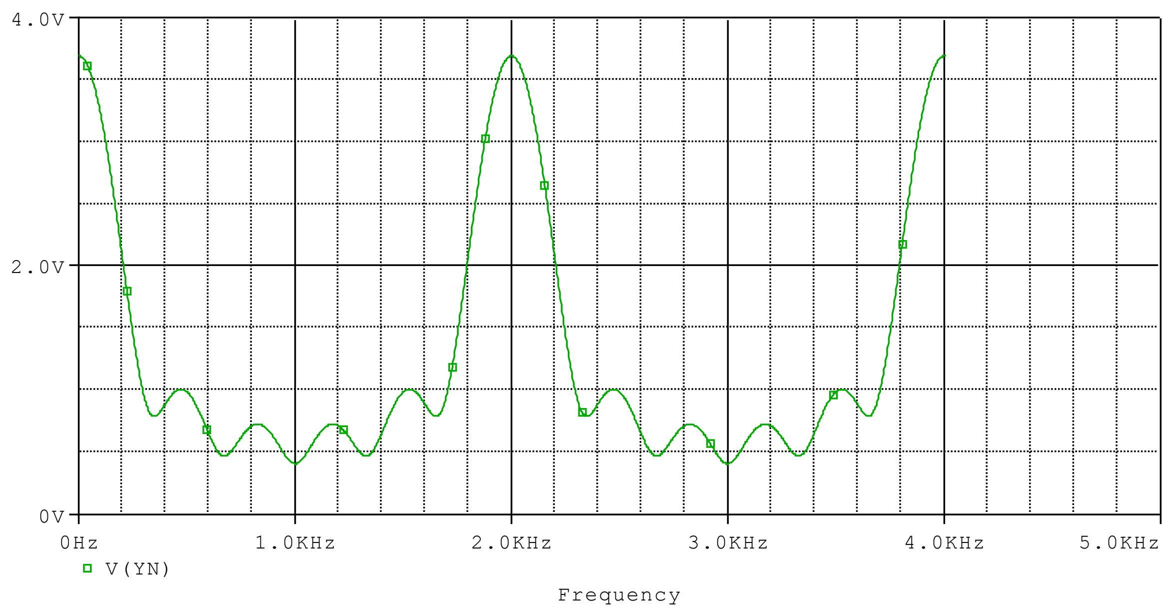


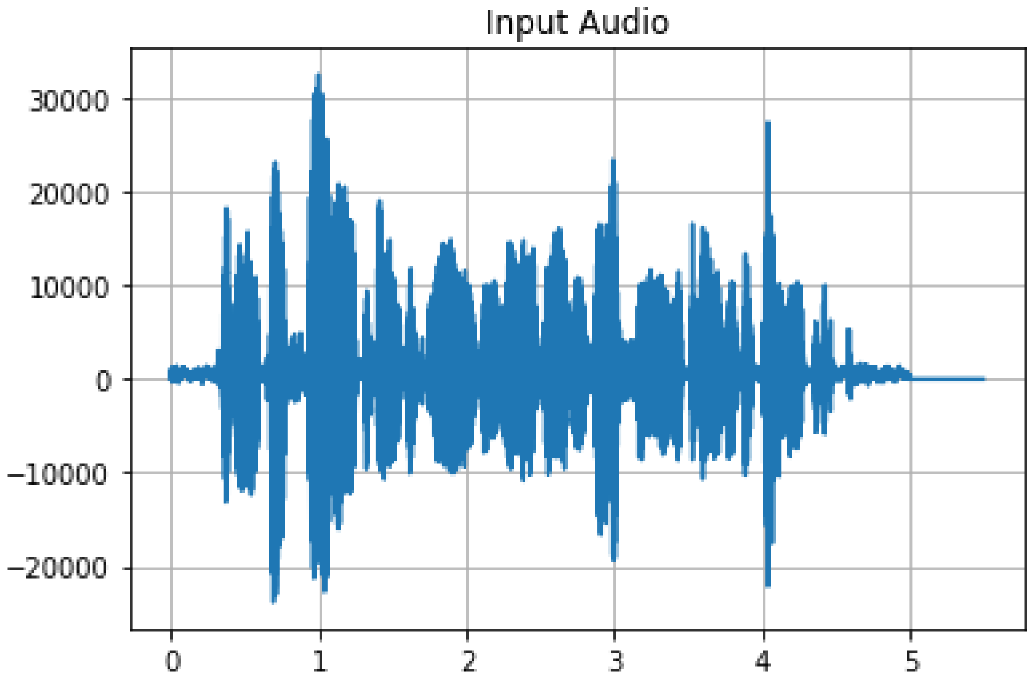

*Figure 20 - Frequency response of the Multiple Echo filter in Pspice*

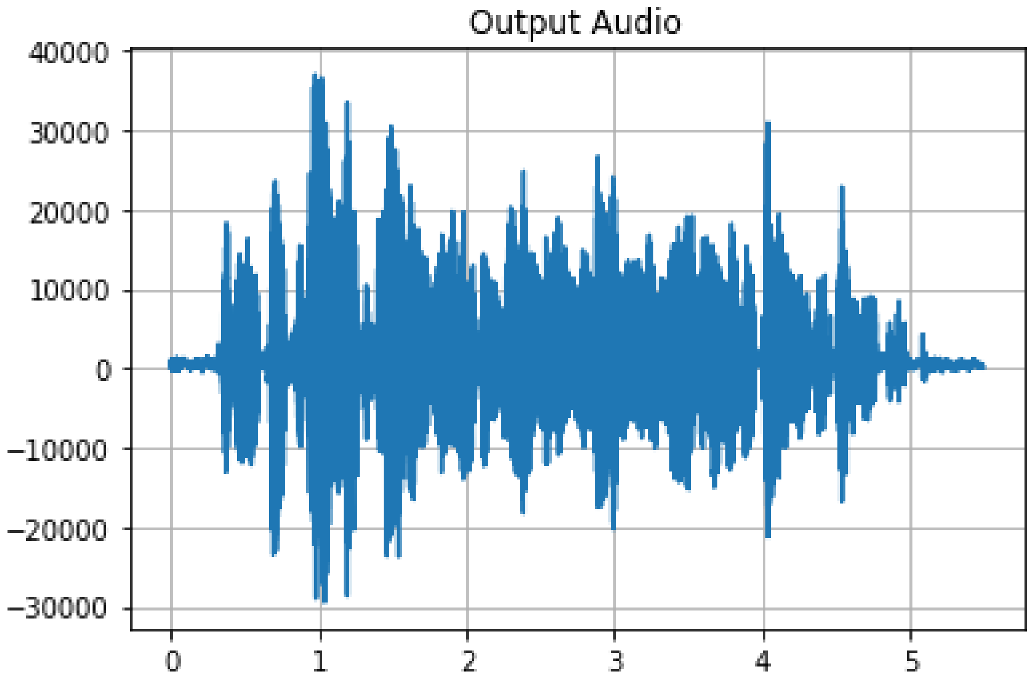
## 4. Python code of speech echo system

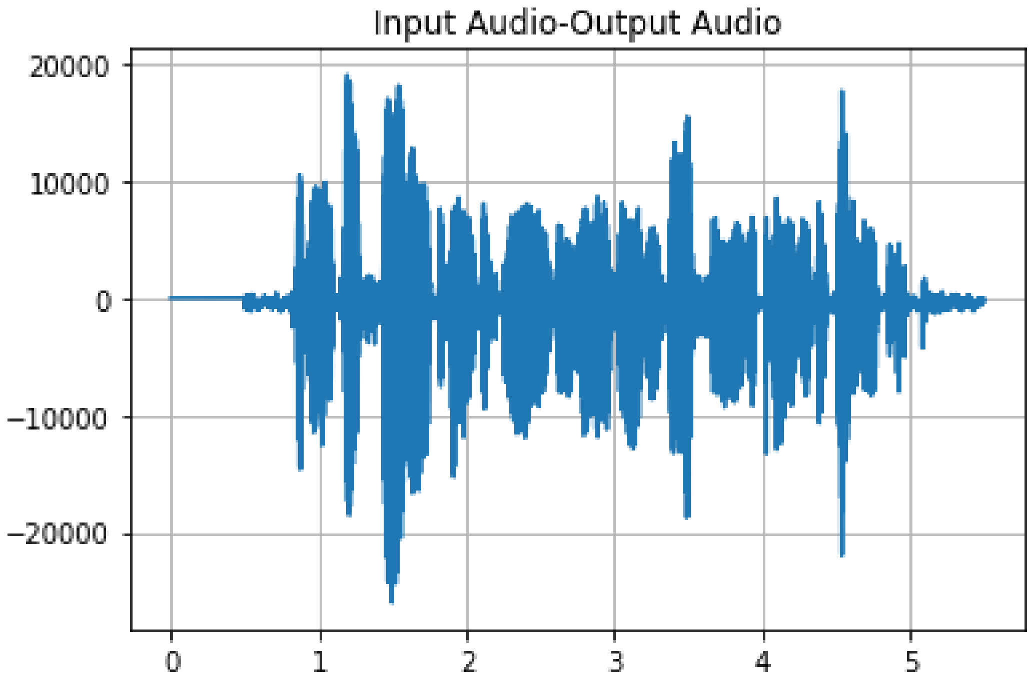
(1) Working with real audio inputs.
It is now required to go back to the 1st section and apply a real audio signal to the input and observe the results. You can read in a real signal using:

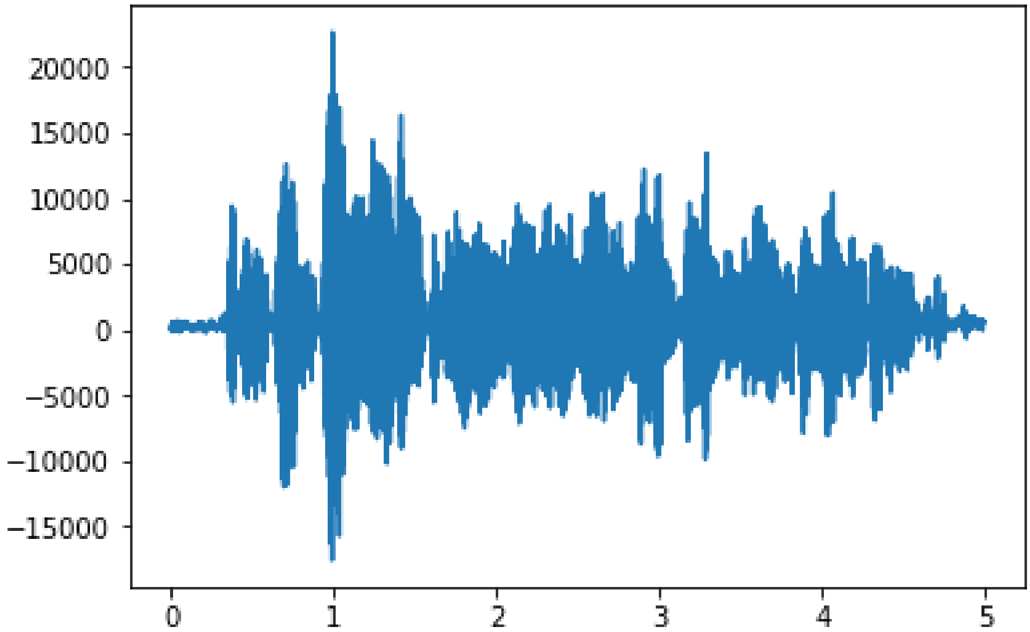
```
import numpy as np
import scipy.fftpack
from scipy.signal import lfilter
import matplotlib.pyplot as plt
from scipy.io import wavfile
plt.close("all")
fs, audio = wavfile.read('c:\Temp\speech_dft.wav')
b=np.zeros(int(fs/2)+1) # Adds a 0.5 sec delay
audio=np.append(audio,b) # Adding zeros prevents the end of the file being truncated.
NN=len(audio)*1.0
t=np.arange(0,NN)/fs
plt.plot(t,audio)
plt.title ('Input Audio');
plt.grid()
plt.show()
b[0]=1
b[int(fs/2)]=0.8
a=1
audioOP=lfilter(b,a,audio)
plt.plot(t,audioOP)
plt.title ('Output Audio');
plt.grid()
plt.show()
diff=audio-audioOP
plt.plot(t,diff)
plt.title ('Input Audio-Output Audio');
plt.grid()
plt.show()
wavfile.write('c:\Temp\save_speech_dft.wav',fs,audio)
audioOP = np.asarray(audioOP, dtype=np.int16)
```

```
wavfile.write('c:\Temp\echo_speech_dft.wav',fs,audioOP)
```



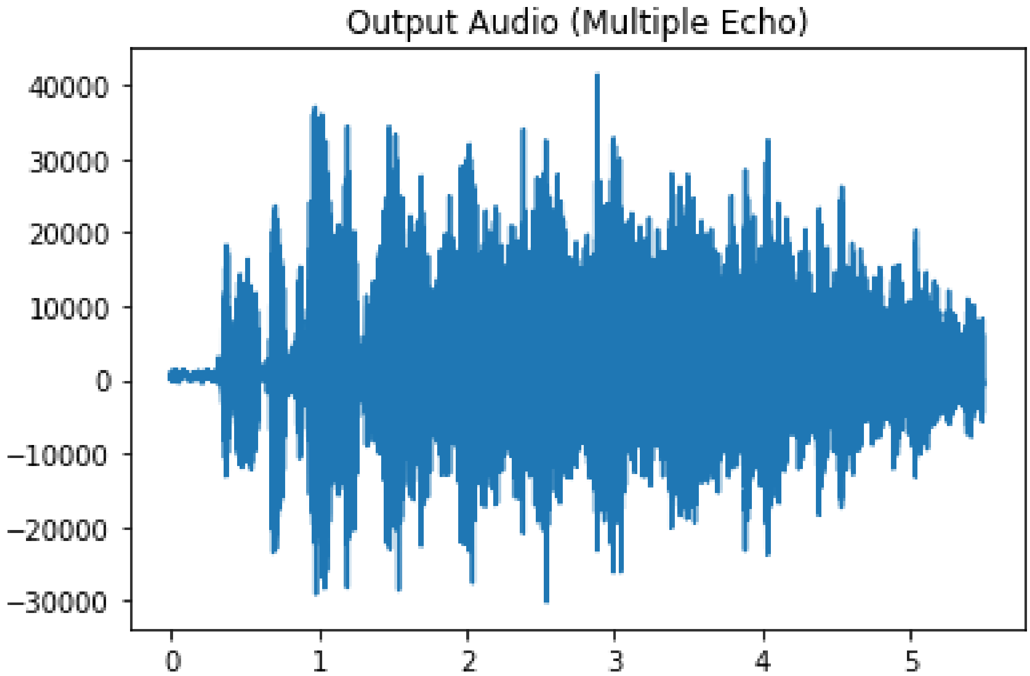



Output Audio (Multiple Echo)

```
N=6; R=int(fs/2)+1; alpha=0.8;
b=np.zeros(N*R+1);
b[N*R]=-alpha**N; b[0]=1;
a=np.zeros(R+1);
a[R]=-alpha; a[0]=1;
audioOP=lfilter(b,a,audio)
plt.plot(t,audioOP)
plt.title ('Output Audio (Multiple Echo)');
audioOP = np.asarray(audioOP, dtype=np.int16)
```

```
wavfile.write('c:\Temp\mult_echo_speech_dft.wav',fs,audioOP)
```




Fs = 22050
R = 11026
Fs/R = 22050/11026 = 0.5

Where Fs is the number of samples per second & R is the number of samples before the next signal repeats.