ELEG 305 SIGNALS AND SYSTEMS
COMPUTER ASSIGNMENT #1 (50 points)
Due Tuesday April 23, 2019

ACOUSTIC SIGNAL PROCESSING: SIMULATING AN IMMERSIVE EXPERIENCE

1. MOTIVATION AND SUMMARY OF ASSIGNMENT

One of the most successful applications of signal processing to everyday life is its use in multimedia systems that combine digital audio and video, computer animation, text and images in real-time interactive applications. The main challenge for these systems is to convince the user that he/she is "experiencing" a "virtual world". For example, the people that go to IMAX theaters expect to feel completely surrounded by the movie images and sound and to feel that they are really in the location where the movie was shot. With respect to the sound portion of the movie-going experience, it is obvious that the acoustic environment in an IMAX theater is very different from that in the filming location. Hence, "immersive" audio systems are required to regenerate the sound effects (actors' voices, car engines, etc) from where the movie was filmed. The better the reproduction, the better the spectator will enjoy the movie.

The range of immersive audio applications is very wide: teleconferencing, virtual reality, home cinema, distance learning, and, in general, any application that needs to reproduce an acoustic scenario different from that of the listener's real environment. The human ear and brain are capable of identifying the location of a sound source within a few degrees precision, as well as detecting delays between sounds as small as 4 μs. Therefore, an accurate reproduction of a specific sound scenario requires that the ear hear acoustic signals that make him/her feel that:

a.) The sound source has the same space location as in the original scenario, both in azimuth (horizontal plane angle), elevation (vertical plane angle) as well as distance.
b.) The listening room is the same as the original location (e.g., a church if the user is listening to a mass or an opera house if he is listening to an opera).

These systems are not new. For example, the Dolby Stereo system, created in the 1970s, used left and right loudspeakers, but also a front speaker and additional ones located in the back and at the sides of the spectator to provide a "surround" feeling. Home theaters work similarly, requiring many speakers. However, there are many applications (e.g., when using a laptop) in which only two speakers can be used. **Hence, it would be good to have a system that can reproduce a feeling of 3D sound and location with only two loudspeakers or headphones.**

As you will see below, this can be accomplished with signal processing for LTI systems. The required processing is very complex, however, so implementation in continuous-time is not feasible and digital signal processing tools are required. **In this computer assignment, you will simulate two simple examples of audio signal processing that represent preliminary stages of the implementation of an immersive sound system with headphones:**

a.) Generation of a stereo signal that creates the feeling that the sound source is located in a certain direction in space.
b.) Simulation of the acoustics of an audio hall with specific reverberation features.

As a summary, this assignment consists of (details are in the sections that follow):

1. **Programming the convolution sum and the difference equation to implement the desired sound effects.**
2. **Recording a sound signal (optional), and processing it using your program.**
3. **Reproducing the sound and verifying sound effects.**

Note that a real system would differ from the one in this assignment in two aspects. First, both acoustic effects would be integrated into a single processing unit; second, an efficient implementation would be required capable of operating in real time (computationally efficient algorithms, A/D & D/A conversion, special-purpose processors like DSPs, etc.). Aside from these two factors, the only differences between this assignment and a real system are the higher complexity and the use of more sophisticated models, but the operating principle is the same.

In Section 2, the signal processing applications are described, along with the work that needs to be done by you. First, the acoustic environment and the hearing process are described in detail and modeled as an LTI system. This should give you some appreciation for how we will regenerate this acoustic experience. Specifically, we will focus on describing the **impulse response** of the acoustic environment. In Section 3, some potentially useful Python functions are listed and described. Some suggestions to help you do the assignment are given in Section 4. In Section 5, we describe all of the files and documents that are provided for you on Canvas, and detail what you must deliver to the TAs for grading.

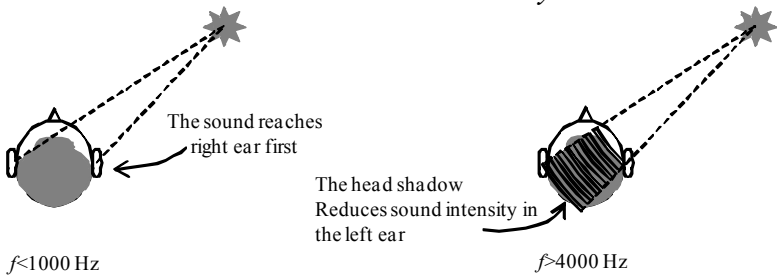
2. ACOUSTIC SIGNAL PROCESSING APPLICATIONS

2.1. Application #1: Spatial Location of a Sound Source

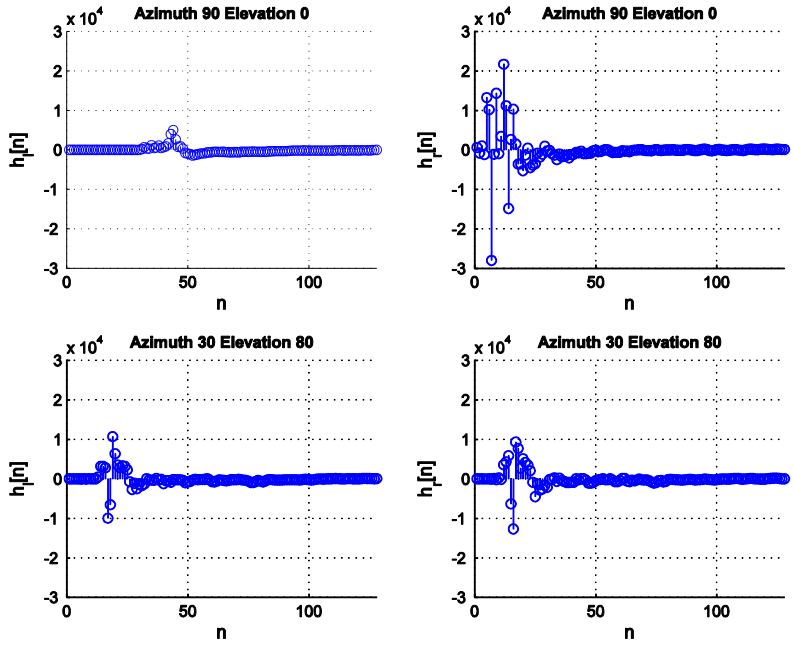
2.1.1. Modeling the Acoustic Channel as an LTI System

The ear and brain form a system capable of locating sound sources in any point in the space. This is possible due to the availability of two ears that act as sound receivers separated in space, and thanks to the complex structure of the auricle (visible part of the ear). Both factors contribute to the perception of sounds with delays, intensities, and frequency contents that depend on the direction of arrival of the source. For example, the information in the horizontal plane (azimuth) is obtained differently depending on the sound frequency (see figure below):

- Low frequencies ($f < 1$ kHz): The sound reaches the left and right ears with different *phases* because they are at different distances from the source. When the difference between the two paths is smaller than a half wavelength, this phase difference can be employed to locate the sound source. (Note that $\lambda = v/f$, where v is the speed of (340 m/s). So $f = 1$ kHz means $\lambda/2 = 17$ cm, roughly the width of your head.)
- High frequencies ($f > 4$ kHz): At these frequencies, the head is an obstacle. It creates an acoustic shadow for sounds whose wavelengths are smaller than its dimensions. Hence, the source can be located based on the different sound *intensity* in both ears.



The reflection and diffraction of sound waves in the head, the torso, the shoulders, and the auricle combine to cause a **frequency response** in the ear that depends on the sound direction of arrival. **The combination of all these factors can be modeled as an LTI system as long as the source and the listener keep still.** The transfer function of this system is known as the *head-related transfer function* (HRTF). The following figure depicts the left/right ear **impulse responses** associated with this HRTF for two different directions of arrival of the sound.



Thanks to spatial directivity characterizations by means of the HRTF, it is possible to create the illusion that a sound source is located in an arbitrary point in space by signal processing for an LTI system, and, therefore, it is possible to create the feeling of 3D sound by means of headphones, as desired in immersive sound systems.

Many experiments have been done to determine these impulse responses. The measurement starts by placing a loudspeaker and a person (or a dummy) with one microphone in each ear in an anechoic (i.e., no echoes) chamber. Afterwards, the impulse response of the HRTF from the loudspeaker to each ear is measured. Unfortunately this response is different for every person because it depends on a person's build, with significant differences between different listeners.

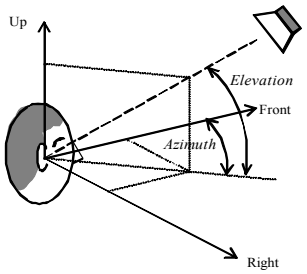
2.1.2. Experiment #1

In this application, we use an LTI system (specifically, a Finite Impulse Response (FIR) filter) to process a "mono" sound signal and create a "stereo" signal that produces the feeling that the sound source is located in a certain direction. The impulse responses have been downloaded from http://sound.media.mit.edu/resources/KEMAR.html, and have the following features:

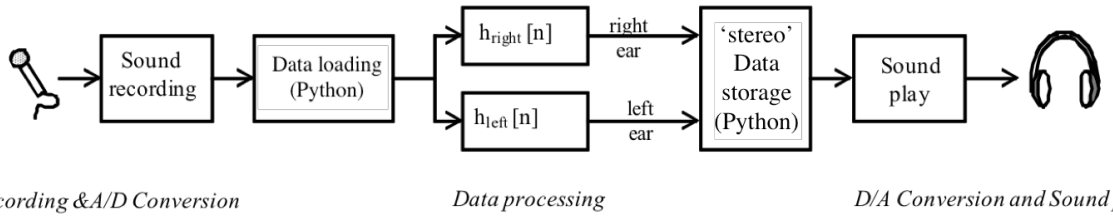
- They were measured using a sampling frequency of 44.1 kHz. This means that the 'mono' sound signal we employ must have been sampled at the same frequency. The Python software provides checks for this requirement.
- The impulse responses are stored in files with the names H*EE*e*AAA*a.dat, where
  EE = Elevation angle with respect to the horizontal plane (in degrees)
  AAA = Azimuth angle with respect to the front direction (in degrees)
- All impulse responses have length 128 samples.
- Each file contains 256 samples corresponding to the left/right ear responses $(h_l[n], h_r[n])$, and each sample is stored as a 16-bit signed integer. The samples are alternated, i.e.,

$$h_l[0], h_r[0], h_l[1], h_r[1], \dots, h_l[127], h_r[127]$$

The following figure depicts the definition of azimuth and elevation angles employed to identify the direction of arrival of the source.



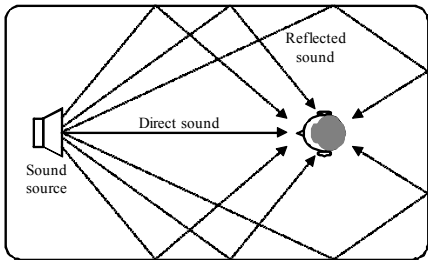
EXPERIMENT #1: The following block diagram summarizes the experiment you will do. First, a speech signal is recorded (you can use the signal we provide in the file 'speech.wav' or you may record one of your own) and it is stored as a discrete-time 'mono' signal in a *.wav* file. Afterwards, this file is opened with Python and the discrete-time signal is convolved with the two impulse responses corresponding to the desired direction of arrival. The two output signals are jointly stored as a 'stereo' signal in another *.wav* file, and, finally, they can be listened to with headphones and a standard sound reproduction application.



2.2. Application #2: Location in a Concert Hall

2.2.1. Modeling the Acoustic Characteristics of a Concert Hall

Some opera aficionados complain because the acoustics of some studio recordings are different from those heard live in a concert hall. This happens because the sound we hear depends very heavily on the room where the listener is located, due to reverberation (i.e., echoes). As depicted in the figure below, in a concert hall the sound reaches the listener directly, but it also propagates in other directions and, when the sound reaches the walls, ceiling, curtains and other objects in the hall, they reflect it. Some of these reflected sounds reach the listener's ear with different amplitudes and delays with respect to the direct sound.

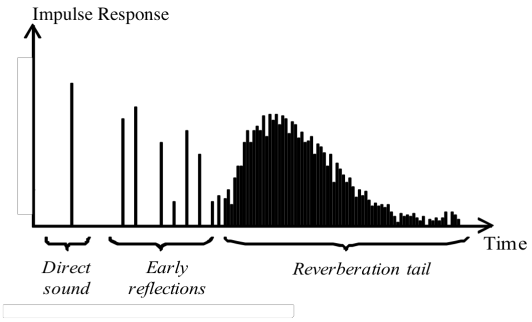


The features of these echoes (number, delay, and amplitude) define the acoustic properties of a concert hall and determine the "liveness", "fullness", "warmth", etc. of this hall from the listener's point of view. The ear receives the direct sound and the multiple echoes and, based on these, gets an idea about the acoustic scenario where the listener is located. For example, the delay between the direct and reverberant sounds can be employed to figure out the hall size (the larger the room, the larger the delay), whereas the relative amplitude of both components

is an indicator of the distance between the source and the listener (the larger the distance the larger the reverberant sound power). The reverberation features depend on the hall geometry as well as the construction materials: the echoes in a room fully covered with marble last longer and have larger amplitudes than those in a room with thick curtains and a carpeted floor.

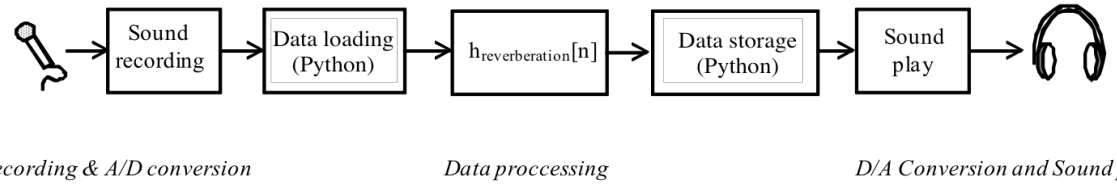
To create the feeling for the listener that he/she is in a certain hall, we must reproduce the echoes that would be generated in that hall, with the same amplitudes and delays. The systems that artificially generate these echoes have very diverse applications, e.g., to replicate the acoustics of a concert hall in our house or to simulate them in a recording studio. For example, imagine that you wish to shoot a commercial in an underground station, but the ambient noise level is too high. So, you could record the video, and afterwards you could record the audio in the studio reproducing the underground station acoustics.

To reproduce the reverberation of a certain concert hall, we need to measure the impulse response of the hall from the sound source to the listener, and then process the sound signal with this impulse response. A typical impulse response is shown below: the direct sound contribution, a first set of echoes that reach the listener shortly after the sound is sent, and a "tail" of diffuse reverberation in which multiple echoes are received. The human ear is capable of recognizing echoes with delays larger than 50 to 100 ms; smaller delays cause the feeling that, after the direct sound, an "aura" is produced that extends the sound in time.

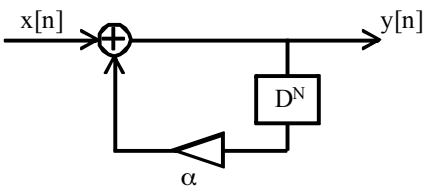


2.2.2. Experiment #2

To illustrate this application, we will generate an artificial reverberation. We do not have any experimental data on impulse responses, so we will employ a simple artificial model that approximates the real behavior. The figure below depicts a diagram for this experiment.



We will use the following model to generate the reverberation (with $|\alpha| < 1$, and where D = time delay). This is the block diagram of a filter with an Infinite Impulse Response (IIR).



EXPERIMENT #2: In this Experiment, you will be required to answer a number of questions, along with the implementation of the model.

- **What is the difference equation of this system? What is its impulse response? Find it and include the derivation in the document you will deliver after finishing the assignment.** Note that we are modeling the reverberation as multiple echoes that reach the listener at regular times and have decreasing amplitude. In real applications, the echoes are not uniformly spaced in time, but their amplitudes do decrease with time. This amplitude decrease is due to the fact that higher delay echoes correspond to those that have suffered more reflections in the walls. Because the reflecting surface absorbs part of the impinging sound energy, each new reflection causes an additional decrease in echo amplitude.
- **Implement this reverberation model using the difference equation derived above. You should do several tests with different values of N and α and observe the effect on the resulting sound signal.** Take into account that, if the reverberation is to be significant, the delay between consecutive echoes should not be much smaller than 50 ms (i.e., 50 ms × 44.1 kHz = 2205 samples if the continuous-time signal was sampled at 44.1 kHz).
- To develop a better model of reverberation, you could model the direct component and the early echoes with an FIR filter and the reverberation tail with an IIR filter. How would you do that? **Propose a block diagram for the overall reverberation model.**

3. USEFUL PYTHON INSTRUCTIONS/FUNCTIONS

Here we provide a brief summary of Python instructions and functions that could be useful for this computer assignment. If you need further assistance with a certain Python command you can ask for information on that routine: `help(`*`command_name`*`)`

Alternatively, you can visit https://docs.scipy.org/doc/numpy/reference/routines.html for more general information, (for Matlab users: https://docs.scipy.org/doc/numpy-dev/user/numpy-for-matlab-users.html). The instructions are listed in different sections depending on their purpose. For every instruction, its name and its purpose are listed.

In many commands below, it is assumed you have executed the following Python commands:

```
from numpy import *
import scipy as Sci
import scipy.linalg
```

a) Special variables and constants

`pi`	`3.1415926…`
`1j`	$\sqrt{-1}$

b) Operators and special characters

`+, -, *, /`	`Plus, minus, multiplication, division`
`**`	`Power`
`%`	`Modulus`
`( )`	`Pass function arguments, prioritize operators`
`array()`	`Construct array, concatenate elements, and specify multiple outputs from function`
`:`	`Create vectors, subscript arrays`
`range()`	`Specify for-loop iterations`
`#`	`Insert comment line into code`
`==, <, >`	`Rational operators: equal to, less than, greater than`
`and, or, not`	`Logical operators: AND, OR, NOT`

c) Basic operations

```
real              Real part of complex number
imag              Imaginary part of complex number
conj              Complex conjugate
abs               Absolute value and complex magnitude
angle             Phase angle
sin               Sine of argument in radians
cos               Cosine of argument in radians
tan               Tangent of argument in radians
exp               Exponential
log10             Common (base 10) logarithm
sqrt              Square root
max               Largest element in array
min               Smallest element in array
```

d) Special functions

```
zeros             Create array of all zeros
ones              Create array of all ones
```

e) Graphical representation (requires matplotlib package)

For the commands here, it is assumed that you have executed the following Python commands:

```
from pylab import *
```

```
figure            Create figure graphics object
plot              Plot lines and/or markers to the axes
show              Display the figure
subplot           Create axes in tiled positions
axis              Axis scaling and appearance
hold              Retain current graph in figure
grid              Turn grid lines for 2-D and 3-D plots on or off
stem              Plot discrete sequence data
```

f) Python as a programming language

```
def               Declare a function
if                Execute statements if condition is true
for               Execute block of code specified number of times
while             Repeatedly execute statements while condition is true
```

g) Workspace handling and variables

```
help(command)     Help for functions in command window
source(command)   Screen display of the contents of a function in the terminal
lookfor           Search for keyword in all help entries
len               Length of vector
```

h) Implementation of LTI discrete-time systems

For the commands here, it is assumed that you have executed the following Python commands:

```
from scypi.signal import *
```

```
lfilter           Implementation of a discrete-time system using its difference
                  equation
convolve          Convolution of two discrete-time sequences
```

i) Sound file handling

For the command here, it is assumed that you have executed the following Python commands:

```
from wave
```

```
wave.open         Write, read WAVE (.wav) sound file
```

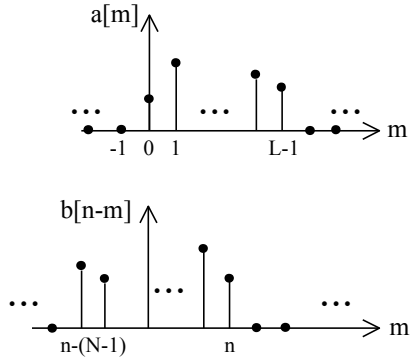
4. POSSIBLE HELPFUL COMMENTS FOR PERFORMING THE ASSIGNMENT

Here, we list problems that you may encounter doing this assignment and some suggestions.

- Note that Python vector indices must always be strictly positive (≥ 0). Also, you must take into account that, even though the convolution output is defined for all output values from $n = -\infty$ to $+\infty$, you only need to compute those that are non-zero. That is, if we convolve a signal $a[n]$ of length L (non-zero samples from 0 to $L-1$) with another signal $b[n]$ of length N (nonzero samples from 0 to $N-1$), to compute the n-th convolution sample,

$$c[n] = \sum_{m=-\infty}^{\infty} a[m]b[n-m],$$

we only need to consider the non-zero terms in the sum, i.e., where the signals overlap (see figure below). The range on the sum will be $[\max\{0, n-N+1\}, \min\{L-1, n\}]$.



- The indices for vectors in Python start at 0. Hence, the signal samples whose nonzero values are for $n = 0$ to $L-1$ are stored in a vector with indices 0 to $L-1$. However, when using a *for* loop you must use *range(0, L)* to set the boundaries of the loop.
- If you record your own speech, please limit its duration to 3 or 4 seconds; otherwise, the sequence will be very long and your program will take quite a long time to run.
- To test the operation of the convolution, it is better to employ a test sequence of very short duration and one for which you know the answer. For example, try:

$$x[n] = \delta[n] + 2\delta[n-1] + 3\delta[n-2]$$

$$h[n] = \delta[n] + 4\delta[n-1] + 5\delta[n-2]$$

$$y[n] = x[n] * h[n] = \delta[n] + 6\delta[n-1] + 16\delta[n-2] + 22\delta[n-3] + 15\delta[n-4]$$

If we write the following instructions in Python to implement the convolution,

```
>>> x = array([1, 2, 3])
>>> h = array([1, 4, 5])
>>> [y] = myconv(x, h)
```

we should get the following output when we write the command `>>> print(y)`

```
[1.  6.  16.  22.  15.]
```

- Unless you have a lot of experience using Python, it is likely that your program will be inefficient and the simulation time might be large. In order to have an efficient Python program, it is important to reduce the number of loops ('while','for') and 'if' instructions.
- The use of *good* headphones is crucial for the first application; otherwise, it is difficult to get the feeling that the signal is coming from the desired direction.

5. RESOURCES FOR ASSIGNMENT AND DELIVERABLES

This assignment should be done in pairs (**two students hand in just one write-up**). If you have never used Python, you can send emails to Juan Florez Ospina (jfflorez@udel.edu), or Guangyi Liu (guangyi@udel.edu) for questions about Python programming.

VERY IMPORTANT: Based on past years, it is best if you download the version of Python from Anaconda, which is an Integrated Development Environment for Python, available for either Mac or Windows. It has all the packages and libraries you need and is easy to use.

5.1. Resource Documents and Programs on Canvas

- *CA #1: Acoustic Signal Processing – Simulating an Immersive Experience.pdf*
- A folder (*Impulse Response Data*) with impulse responses for different source locations for the first experiment. As mentioned in Section 2.1.2, they are of the form H*EE*e*AAA*a.dat.
- A folder with five Python files (*Python Files*):
  - Experiment #1:
    - *main_stereo.py*: The main program you need to run to generate the stereo signal for Experiment #1. You (might) need to modify the name of "input wave file" and the name of "impulse response file". Please read the whole program carefully to understand how it works. In this program, it calls the function *my_conv.py*.
    - *my_conv.py*: **The Python file for the function *my_conv* where you need to write the code for the convolution.** The program *main_stereo.py* will not run until you write this code. A function header is provided which defines the input/output parameters and, therefore, the interface with the main program.
  - Experiment #2:
    - *main_reverberation.py*: The program you need to run to generate the signal with reverberation for Experiment #2. You need to modify the section of the program entitled "LTI system parameters". Please read the whole program carefully to understand how it works. In this program, it calls the function *my_diff_equation.py*.
    - *my_diff_equation.py*: **The Python file for the function *my_diff_equation* where you need to write the code for the difference equation.** *main_reverberation.py* will not run until you write this code. A header is provided which defines the input/output parameters and, therefore, the interface with the main program.
  - *speech.wav*, an audio signal that can be used in the simulations. If you use your own sound signal, it is important that the sound signal is sampled at 44.1 kHz.

5.2. DELIVERABLES (What every pair of students must deliver)

- **Paper copies of the programs that you wrote to implement the convolution and the difference equation → *my_conv.py* and *my_diff_equation.py*.**
- **An email to Guangyi Liu (guangyi@udel.edu) with the files *my_conv.py* and *my_diff_equation.py* as attachments. The "Subject" line of the e-mail must be**

  ELEG305-Comp.Assignment #1 – *Student1 family name, Student2 family name*

- **A document (1 or 2 pages long) with answers/comments on the results, specifically:**
  - **What you learned (one paragraph) from Experiment #1.**
  - **The difference equation and impulse response for the simplified reverberation model in Section 2.2.2 (Experiment #2).**
  - **A discussion of the effect of the values of α and N on the reverberation results.**
  - **A proposal for a better model of reverberation (i.e., a block diagram).**