BELMONT UNIVERSITY, CURB COLLEGE OF ENTERTAINMENT AND MUSIC BUSINESS

# AET 5420 Project 1

## Audio Signal Processing

Due: April 3rd, 2024

## 1 LINEAR SYSTEM MODELING

Many audio signal processing systems can be modeled as linear systems. A linear system processes the amplitude of the input signal by a linear combination of a signal's samples. This combination involves **multiplying** the signal's amplitude by scalar values and **adding** together the result with previous samples. An example of a linear system is when sound travels through an acoustic space. When a signal is sent through a chamber reverb and recorded, the resulting signal is a combination of delayed repetitons of the input signal that have been reduced in amplitude.

### 1.1 IMPULSE RESPONSE

All digital signals are comprised of a sequence of samples or impulses. It does not matter whether a signal is a periodic sine wave or aperiodic white noise, all sampled signals can be reduced to impulses at different amplitudes. If it is assumed that a linear system processes each sample independently (time-invariance) and that loud samples are treated identically to quiet samples (linearity), then a system can be modeled by how it *responds* to an *impulse*. An **impulse response** represents the output of a system when a single impulse is sent through the system as the input signal. This impulse response can be used to process each sample in a signal.

### 1.2 CONVOLUTION

Convolution is the mathematical operation by which a sequence of samples (i.e. a signal) are processed by the impulse reponse of a linear system. Essentially, each input sample, $x[n]$, is independently sent thought the system, $h[n]$, to contribute to the output signal, $y[n]$, at

different times. Therefore, the output signal is created by summing together a sequence of impulse responses. Fig. 1.1 represents a block diagram for the input and output of a linear system.

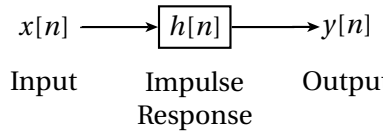


Figure 1.1: Block Diagram of a Linear System

The following equation for convolution can be used to determine the amplitude of the output signal, $y[n]$, at any time sample, $n = 1,2,3,...$

$$y[n] = x[n] * h[n] = \sum_{i=1}^{M} x[n-m+1] \cdot h[m], \text{ where M = length(h)} \tag{1.1}$$

A built-in MATLAB function is available to perform convolution called: ***y=conv(x,h)***. It may be helpful to use during the project. The input variables to the function are a signal, $x$, and a system, $h$, which are both a sequence of samples. The output variable, $y$, is a sequence of samples that represent the processed signal.

### 1.2.1 Frequency-Domain Convolution

A different way to perform convolution is to use a calculation in the frequency domain. As shown previously in class:

$$fft\Big(x[n] * h[n]\Big) = X[\omega_k] \cdot H[\omega_k]$$
$$x[n] * h[n] = ifft\Big(X[\omega_k] \cdot H[\omega_k]\Big)$$

In situations where the input signal is known (ex: synthesized sine-sweep signal, $x[n]$) and the output signal is measured (recording of the sine-sweep through a system, $y[n]$), the impulse response of the system, $h[n]$, can be determined using the following steps:

$$\begin{aligned} X[\omega_k] &= fft\Big(x[n]\Big) \\ Y[\omega_k] &= fft\Big(y[n]\Big) \\ H[\omega_k] &= \frac{Y[\omega_k]}{X[\omega_k]} \\ h[n] &= ifft\Big(H[\omega_k]\Big) \end{aligned} \tag{1.2}$$

# 2 PROJECT

For this project, your group will submit a **written report** summarizing the results of the following experiment. Your group will synthesize different types of signals to measure the response of various systems. You will measure digital systems by capturing the impulse response of several plug-ins within Pro Tools. You will measure analog systems by capturing the response of an acoustic chamber and an electronic circuit (hardware EQ). After capturing the response of each system, you will import each .wav file into MATLAB and analyze the result. Finally, you will convolve the provided test signal, *Vocal.wav* in MATLAB with each impulse response.

You will submit a report including a written description of the steps you used, your analysis of each impulse response (written description with pictures), and your code as an attached appendix. Please capture screen shots from Pro Tools and take pictures during your measurements to include in the report that demonstrate the process of capturing each system. Your report should stand on its own, such that any audio engineer could read it and understand what you did and why you did it. The report should have the following sections: introduction, methods, results, and conclusions.

## 2.1 PART 1 - DIRECT IR METHOD

For digital systems, the best way to measure the response of a linear system is to use an impulse signal as the input, $x[n]$. In this case, the output signal is equal to the response of the system, $y[n] = h[n]$, and can be used directly.

### 2.1.1 SYNTHESIZING AN IMPULSE SIGNAL

An impulse signal is a sequence of samples that are all zero except for one sample which has an amplitude of 1. To synthesize an impulse signal in MATLAB, first create an array of all zeros. For practical purposes, you can make the signal to be one second in duration, although this is not required. Next, assign the first sample in the array to have a value of 1. Finally, create a **.wav file** from your array to be imported into Pro Tools.

*Note:* It is recommended to use a consistent sampling rate of 48 kHz throughout the project.

### 2.1.2 CAPTURING AN IMPULSE RESPONSE

After importing your impulse .wav file into Pro Tools on an audio track, open the following plug-ins as inserts of the audio track: Mod Delay III and 7-band EQ. *Note:* you may substitute other plug-ins in place of the recommended plug-ins if they perform similar processing.

- Medium Delay
  - Use a medium delay time (250 - 500 ms) with negative feedback (-40 - 60 %)
- Create the following EQ effect

  - Use a HPF at 300 Hz
  - Scoop out the mids by using a peaking filter at 2 kHz
  - Boost the amplitude of the highs by using a shelf at 12 kHz

Print the result of playing the impulse signal through each plug-in separately onto new audio tracks. Label each new audio file according to the plug-in used to create it. Each audio file represents the impulse response of the plug-in with the current settings and parameters. *Note:* it is important that the printed signal does not clip during recording. If necessary, lower the amplitude of the impulse signal such that the output does not clip.

Also, print the *Vocal.wav* file through each effect. Save the result for later. You will use the processed file in MATLAB for a Null Test.

#### 2.1.3 Analyzing the Impulse Responses

It is important to analyze each impulse response based on the purpose of the processing. For spectral effects (filtering and EQ), you should produce plots of the frequency spectrum for each impulse response. This can be accomplished using a built-in MATLAB function called: *freqz(h)*, where *h* is the array for the impulse response of the filter and eq effects. You should be able to demonstrate with the figures that your impulse responses are accomplishing their spectral processing purpose.

- Plot the magnitude (amplitude only, not phase) response
  - Use a decibel scale for the vertical axis
  - Use a logarithmic scale for the horizontal axis - *semilogx*

For the delay, you should start by simply plotting the time-domain waveform of the impulse responses. You should be able to demonstrate with MATLAB figures that your delay effect has repeating impulses at multiples of the time set by the plug-in. Make sure to set the scale of the horizontal axis of the waveform to seconds.

#### 2.1.4 Convolving the Test Signal

Import the provided test signal, *Vocal.wav*, into MATLAB along with each of the impulse responses captured in Pro Tools. Use the built-in MATLAB function, ***y=conv(x,h)***, to create a processed version of the input signal with each impulse response. Listen to the result to make sure it has worked successfully. Create new .wav files from the output signals to be included along with your report.

#### 2.1.5 Null Test

Compare the printed, processed versions of the sound files in Pro Tools to the output, convolved arrays in MATLAB. Perform a **Null Test** by subtracting one signal from the other. If the signals are identical (after compensating for a delay offset), then the result should be an array

of zeros. Plot the output array over time from **Null Test** for all effects. Be sure to describe the result in your paper. Is the result exactly zero? Why or why not?

## 2.2 Part 2 - The Sine Sweep Method

A different approach to measuring the impulse response of a system is to use a sine sweep as the input signal. A sine sweep is a signal which constantly increases in frequency from 0 Hz to the Nyquist frequency throughout its duration.

The sine sweep method is preferrable for analog systems compared to the Direct IR Method. When measuring continuous systems (electronic circuits or acoustic environments), it is necessary to use a continuous signal as the input. An impulse spike does not work ideally when sent through D/A converters and a loudspeaker. Similarly, approximations of an impulse such as a ballon pop, cap/nail gun, or hand clap can be used, but with less than ideal results. Rather, the smooth sine sweep is preferred to measure the response of the system at all frequencies equally.

An additional step necessary for the Sine Sweep Method is the process of *deconvolution*. Here, the input signal is deconvolved from the output signal to recover the system's impulse response, $h[n]$. This process is complicated to perform in the time-domain based on Eq. 1.1, but is straight-forward to perform in the frequency domain (Eq. 1.2).

In practice, there are a few extra steps necessary to achieve optimal results. The details of this approach (Tikhonov Regularization) are described in the included document, *Deconvolution.pdf*.

## 2.3 Measuring the System Response

The following steps can be used to measure the response of different systems and recover the impulse response.

- Synthesize a sine sweep signal in Matlab
  - ***t = [ 0 : Ts : 10 ];*** *% 10 second sweep*
  - ***x = chirp(t,0,10,22000);*** *% built-in sweep synthesizer*
  - Add a fade-in and fade-out to the sweep
  - You may experiment with adding additional parameters to the chirp function
    - Try linear and logarithmic sweeps
    - Start the cosine with a phase of -90 degrees instead of 0 degrees
- Measure the response of one (1) acoustic environment
  - This could be any space on campus: studio, stairwell, Neely Dining Hall, etc.
  - You will play the sweep back through a loudspeaker and capture it with a microphone

- Measure the response of one (1) hardware equalizer
  - Use different settings than the digital EQ from Part 1
  - Include both boosting and cutting
- After measuring these systems, you will have recorded the response, $y[n]$
- Use the following steps in Matlab to recover the impulse response, $h[n]$
  - Import the original sine sweep, $x[n]$ and the measurement, $y[n]$
  - If necessary, zero-pad $x[n]$ so it is the same length as $y[n]$
  - Convert both signals to the frequency domain, $X[\omega_k]$ and $Y[\omega_k]$
  - Create a regularization factor, $\lambda$
    - Experiment with values between 0 and 1
- Use a loop to iterate over each frequency bin, $k$
- Calculate $H_k$ for each frequency bin
  - $H_k = \frac{X_k^* \cdot Y_k}{(X_k^* \cdot X_k + \lambda) T_s}$
  - Where: $X_k^* = conj(X_k)$
  - *Hint:* the regularization factor is supposed to prevent the denominator from approaching ~0
  - Also, experiment with calculating it directly for comparison purposes: $H_k = \frac{Y_k}{X_k}$
- Convert $H[\omega_k]$ to the time domain
  - $h[n] = real\Big(ifft\Big(H[\omega_k]\Big)\Big)$
- Normalize the amplitude of $h[n]$
- If there is silence at the beginning of $h[n]$ before the main impulse, it can be removed

### 2.3.1 Analyzing the Impulse Responses

For the hardware equalizer, you should create a similar type of plot as the digital equalizer from Part 1.

- Plot the magnitude (amplitude only, not phase) response
  - Use a decibel scale for the vertical axis
  - Use a logarithmic scale for the horizontal axis - *semilogx*

For the reverb effect, you should look at the time is takes for the impulse response amplitude to decrease by 60 dB. To perform this analysis, you should plot the impulse response relative to the decibel scale: $plot(t, 20 * log10(abs(h)))$.

### 2.3.2 Convolving the Test Signal

Finally, convolve the provided test signal, *Vocal.wav*, with your impulse responses in Part 2. Listen to the result to make sure it has worked successfully. Create new .wav files from the output signals to be included along with your report.

### 2.3.3 Null Test

Also perform the Null Test with the physical systems. Be sure to describe the result in your paper. Is the result exactly zero for all samples of the subtracted signal? Why or why not?

## 3 Submission

To submit your project, create a single zip file that contains your written report, any MATLAB files used during the experiment, and any sound files synthesized during the experiment. Name the zip file: xxxxx_AET5420_PJ1.zip, where *xxxxx* is the last name of one group member. Be sure to include the names of all group members in the report. Email the zip file to: eric.tarr@belmont.edu before the start of class on April 3rd.