1 Introduction

In this paper, we will test the use of white noise for distance measurement and room response. This is a practical application of discrete autocorrelation and the FFT (Fast Fourier Transform).

2 Mathematical Description

We know that autocorrelation is a description of how correlated a signal is with it self in time. This is useful for detecting repetitive responses in signals whether by nature of the signal or noise, such as echos. White noise, by definition, is a constant power signal in the frequency domain. This implies that it is only correlated at the origin. We will be using a band-limited white noise to approximate this signal.

We know that the spectral density is the Fourier Transform of the the autocorrelation. Since we will be working with a signal in the discrete domain, we will use the FFT. This is an fast running algorithm to compute the discrete time Fourier Transform. We will be able to see what frequencies the room attentuates or accentuates by observing the frequency response since white noise has an even power across the spectral density.

The recorded audio is in 16-bit wav format, sampled at 44 kHz. This means that for every second of audio their is 44,000 16-bit samples. Thus if the autocorrelation peaks at 400 tau then that is $\frac{400}{44000} = 9.1$ milliseconds past the origin. The speed of sound in air is 1125.33 ft/sec. Thus in the previous example, the distance from the source of the echo is $.0091 * 1125.33 = 10.24$ ft. Furthermore, since this is a measurement of an echo, the distance should be divided in half. This is because the echo is a reflection off the wall and thus is first picked up by the microphone from the source, travels Δx to the wall and Δx back to the microphone.

3 Problem Description

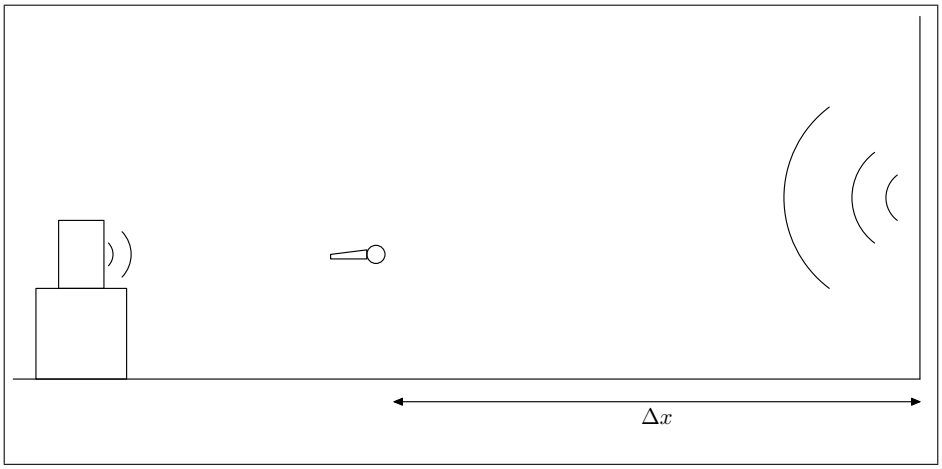

Figure 1: Experiment Setup

The basic setup is as pictured above. We will be using a speaker to broadcast the white noise, a microphone to pick up the transmitted signal, and a wall to measure the distance from. Furthermore, the speaker is a reference monitor which implies that the white noise that it has a flat frequency response. This is the same with the microphone. As in the diagram above, the microphone is facing the back wall. The microphone has a hyper-cardioid pickup pattern. This means it will pickup noise in the front and back but not sides. Thus we will be able to pickup the original white noise and the echo off the back wall only. This is of course only an approximatation since the microphone merely attentuates signals from other directions. However, the attentuatation is large enough to approximate it as described.

For the measurement of distance to the wall, we set the microphone an x-amount of distance from the back wall. For both of the trials this was 8' and 9'9" respectively. Then using a band-limited white noise signal, generated using the audio program Audacity, we recorded audio from the speaker and the relection off the back wall. Then using Matlab we calculated the autocorrelation. Only the first 1000 entries of the autocorrelation were considered in code. This is so you can see the echo on the autocorrelation. The code for this is below.

The spectral density measurement involved the same setup as above and is included in the code below.

For both measuring the autocorrelation and spectral density a utility program was written to compute the parition of a vector into a smaller vector using an average for the partition.

```
SPEED_OF_SOUND = 1125.33;

%for matlab and not octave use audioread
audio_8 = wavread('8ft.wav');
audio_9 = wavread('xft_session.wav');
%reduce to one channel
audio_8 = audio_8(:, 1);
audio_9 = audio_9(:, 1);

[cor_8,lags_8] = xcorr(audio_8, 'biased');
[cor_9,lags_9] = xcorr(audio_9, 'biased');
p_lags_8 = find(lags_8 >= 0);
p_lags_9 = find(lags_9 >= 0);

[l, li] = max(partition(abs(cor_9(p_lags_8(1:1000))), 100));
li*(1000/100)

%figure
%bar(1:1000, cor_8(p_lags_8(1:1000)))

%figure
%bar(1:1000, cor_9(p_lags_9(1:1000)))

%figure
%bar(1:50, partition(abs(cor_9(p_lags_9(1:1000))),50))
```

Listing 1: Main Program

```
function y = partition(vector, num_bins)
  y = zeros(1, num_bins);

  if size(vector,1) > size(vector,2)
    sz = size(vector,1);
  else
    sz = size(vector,2);
  end

  p_sz = ceil(sz/num_bins);
  r_sz = p_sz;
  for i = 1:num_bins
    for j = 1:p_sz
      if i*p_sz + j > sz
        r_sz = sz - i*p_sz +j;
        break;
      end
      y(i) = y(i) + vector(i*p_sz + j);
    end
  end

  for i = 1:num_bins
    if i == num_bins
      y(i) = y(i)/r_sz;
    else
      y(i) = y(i)/p_sz;
    end
  end
end
```

Listing 2: Function to Parition Array using Average Scheme

4 Analysis

Picture above is the result of running the Matlab code. As noticeable their is an echo in the autocorrelation of both signals at x,y for the 8', 9'9" signal respectively. Computing the distance as described earlier yields the following information: