RTDSP Week 14 Homework

CYEE 10828241 Chen Da-Chuan

June 17, 2023

Contents

List of Figures

List of Tables

1 Custom LMS Adaptive Filter

1.1 Requirements

The requirement of this homework is to simulate an acoustic echo path generated when using hands-free telephones, and to cancel the echo using an adaptive filter.

1. Acoustic echo path $P(z)$ is simulated by $p = fir1(511, 0.9)$
2. Perform acoustic echo cancellation using adaptive filter with LMS and NLMS algorithms with filter lengths of 256, 512, and 1024.

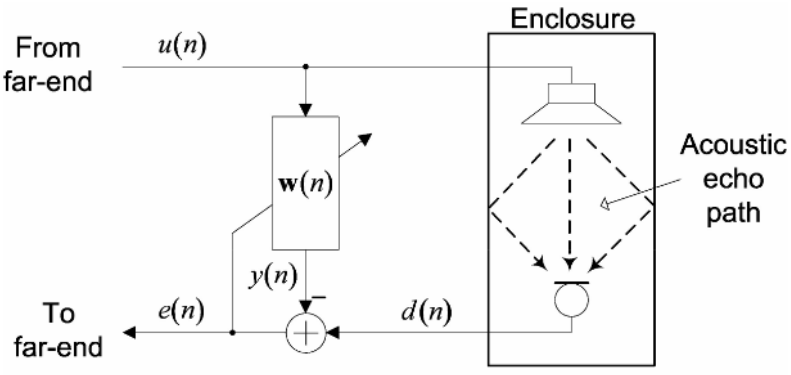

Figure 1: System Diagram

1.2 LMS Adaptive Filter Design

The custom coded LMS adaptive filter works as follows:

1. Acquire input signal $x(n)$, desired signal $d(n)$, step size μ, and filter length L.
2. Compute output signal by
$$y(n) = w^T(n)x(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l) \tag{1}$$
3. Compute error signal by
$$e(n) = d(n) - y(n) \tag{2}$$
4. Update filter coefficients by
$$w(n+1) = w_l(n) + \mu x(n-l)e(n), \quad l = 0, 1, \ldots, L-1 \tag{3}$$

Full implementation is available in `https://github.com/belongtothenight/RTDSP_Code/blob/main/src/w14/LMS.m`.

1.3 NLMS Adaptive Filter Design

The custom coded NLMS adaptive filter works as follows:

1. Acquire input signal $x(n)$, desired signal $d(n)$, step size μ, and filter length L.
2. Compute output signal by
$$y(n) = w^T(n)x(n) = \sum_{l=0}^{L-1} w_l(n)x(n-l) \tag{4}$$
3. Compute error signal by
$$e(n) = d(n) - y(n) \tag{5}$$

4. Estimate input signal power (MSE) by

$$P(n) = (1-b)P(n-1) + bx^2(n), \quad b = \frac{1}{L} \tag{6}$$

5. Compute step size by

$$\mu = \frac{\alpha}{L\hat{P_x}(n) + c}, \quad 0 < \alpha < 2, \quad c \approx \epsilon \tag{7}$$

6. Update filter coefficients by

$$w(n+1) = w_l(n) + \mu x(n-l)e(n), \quad l = 0, 1, \ldots, L-1 \tag{8}$$

Full implementation is available in `https://github.com/belongtothenight/RTDSP_Code/blob/main/src/w14/NLMS.m`.

1.4 Verify the Adaptive Filters

I'm going to use a $600Hz$ signal as the input signal, and generate the desired signal by simulating the acoustic echo path $P(z)$.

In order to validate my custom coded LMS and NLMS adaptive filters, I'm going to compare the results with the adaptive filters from the official MATLAB library.

From results of raw data (fig 2/3), amplitude spectrum (fig 4/5), and power spectrum (fig 6/7), we can see that the results from my custom coded LMS adaptive filter are almost identical to the results from the official MATLAB library. However, the results from my custom coded NLMS adaptive filter are not as good as the results from the official MATLAB library. The reason is likely due to the parameter settings for α and c.

Since the LMS/NLMS algorithms are trying to minimize the mean square error (MSE), we can see from figure 8 that the MSE is decreasing as the number of iterations increases. The number of MSE spikes at the beginning of the learning curve is likely due to only a small number of samples are used to compute the MSE. The MSE spikes will disappear as the number of samples increases.

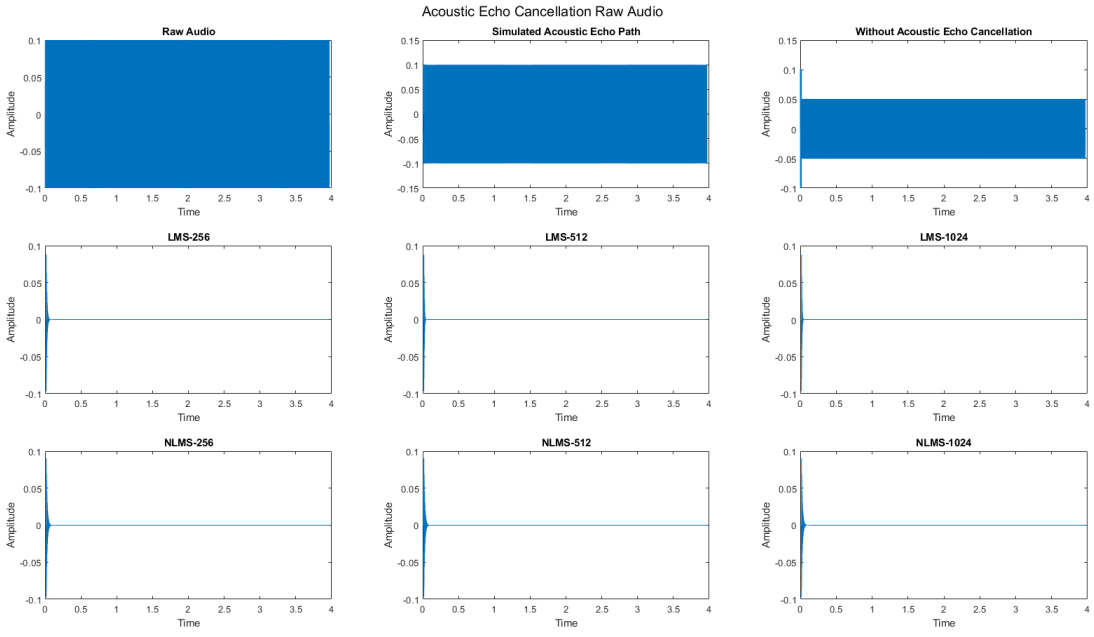

Figure 2: Sinewave Raw data from Official MATLAB library

2 LMS Adaptive Filter Processing Results

Audio file and its corresponding data in CSV format for sinewave verification are zipped as "output_sine.zip", audio 1 as "output_timit_a.zip", audio 2 as "output_timit_b.zip".

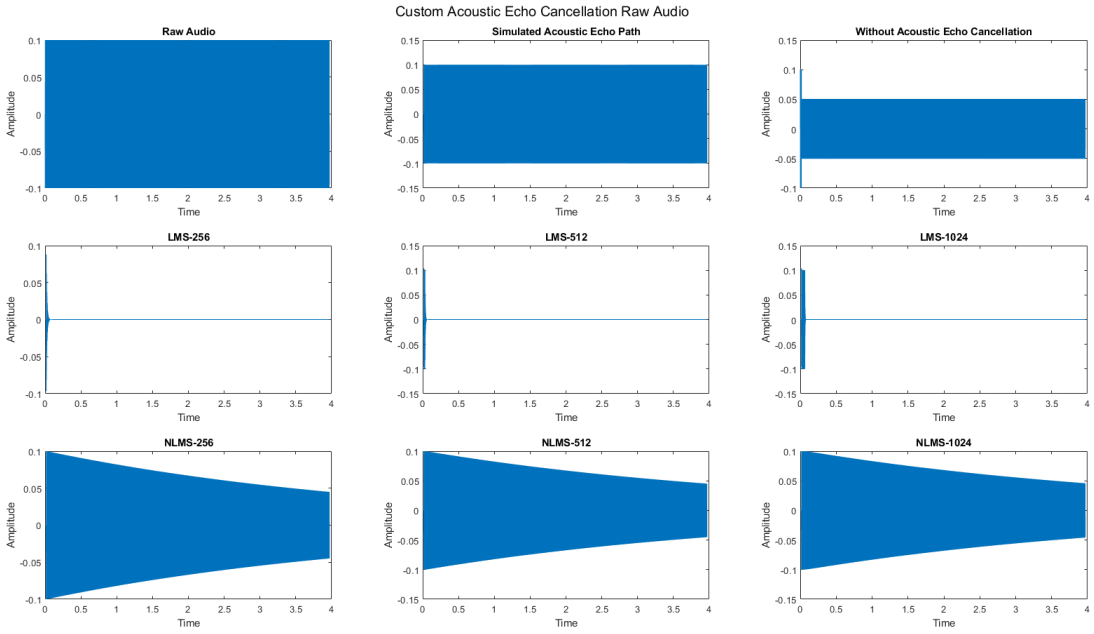

Figure 3: Sinewave Raw data from Custom Coded LMS and NLMS

2.1 Audio 1

The result of audio 1 includes raw data (fig 10/11), amplitude spectrum (fig 12/13), power spectrum (fig 14/15), learning curve (fig 16), and ERLE (fig 17). From figure 16, the MSE is kept from continuous increasing which means the adaptive filter is working. From figure 17, the ERLE is kept from continuous increasing which means the adaptive filter is working.

2.2 Audio 2

The result of audio 2 includes raw data (fig 18/19), amplitude spectrum (fig 20/21), power spectrum (fig 22/23), learning curve (fig 24), and ERLE (fig 25). From figure 24, the MSE is kept from continuous increasing which means the adaptive filter is working. From figure 25, the ERLE is kept from continuous increasing which means the adaptive filter is working.

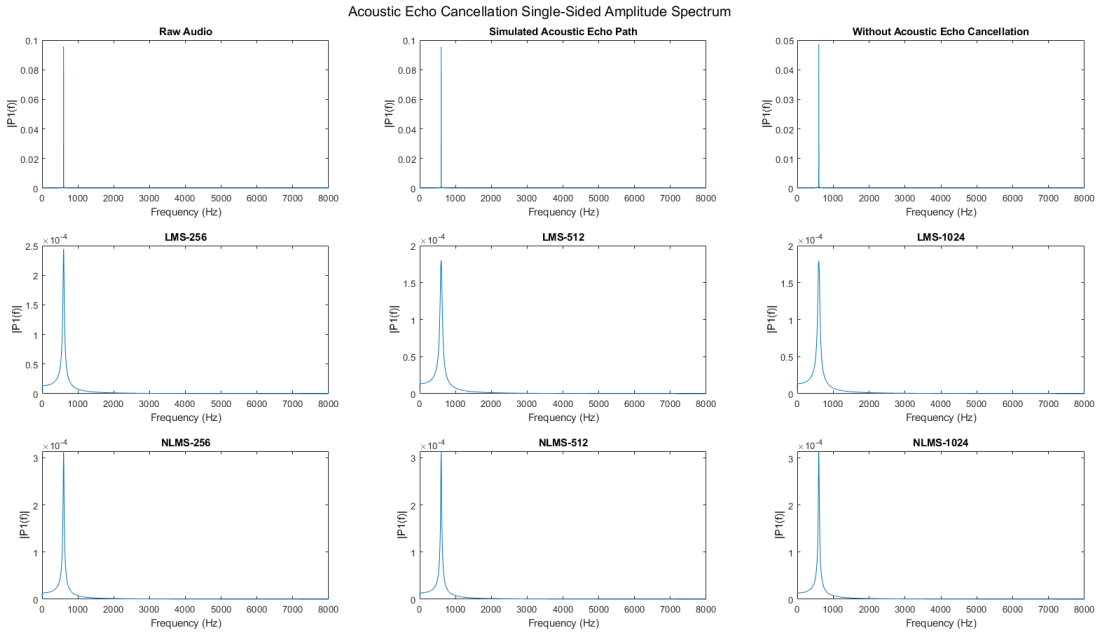

Figure 4: Sinewave Amplitude Spectrum from Official MATLAB library

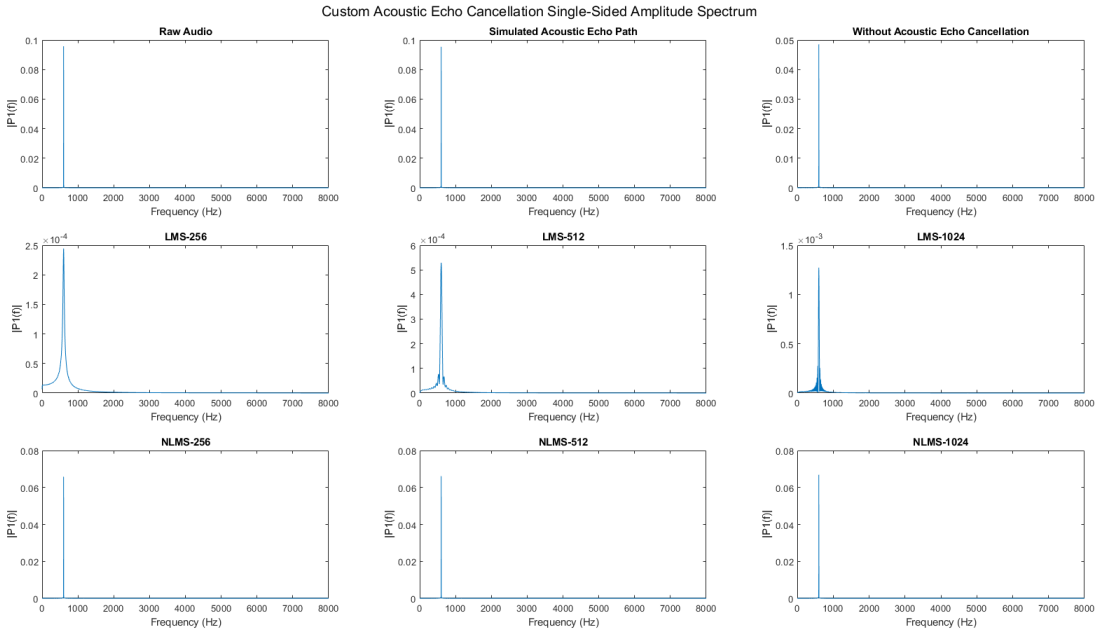

Figure 5: Sinewave Amplitude Spectrum from Custom Coded LMS and NLMS

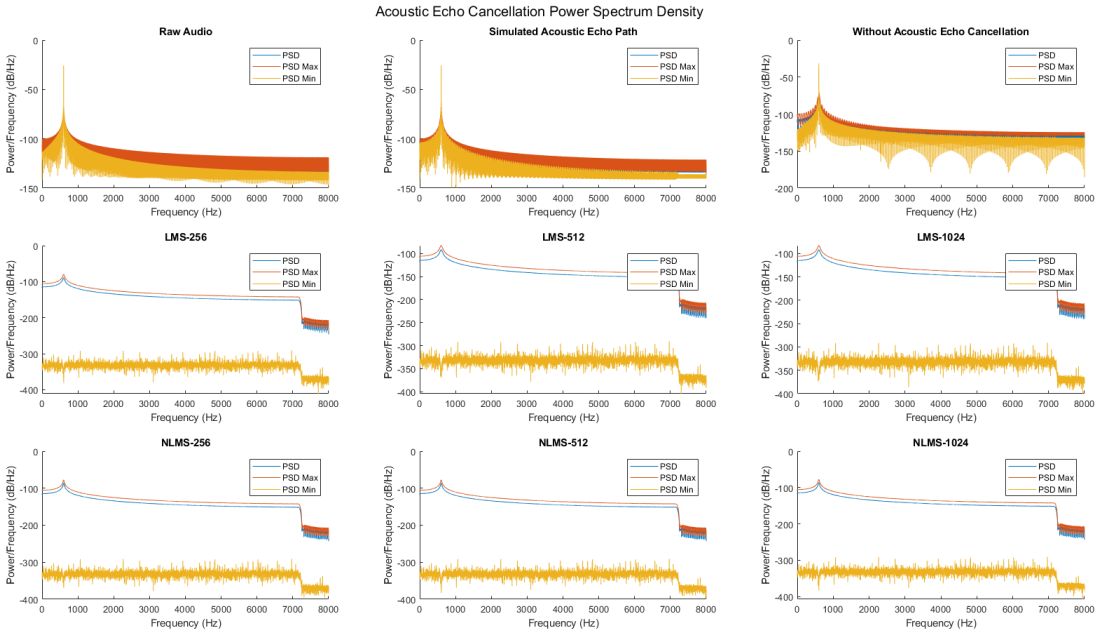

Figure 6: Sinewave Power Spectrum from Official MATLAB library

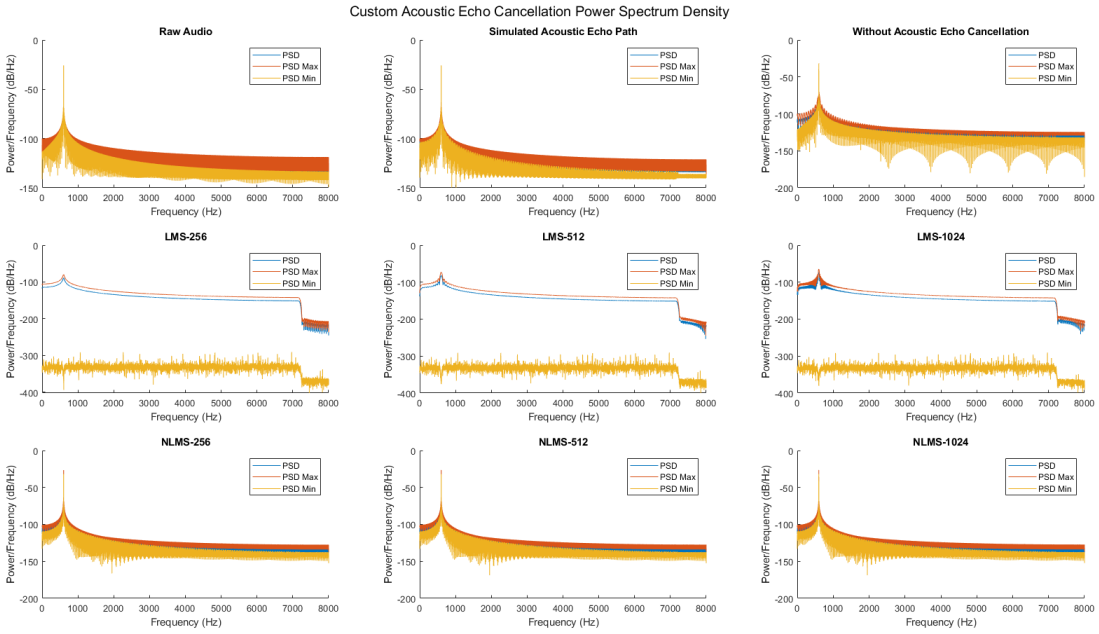

Figure 7: Sinewave Power Spectrum from Custom Coded LMS and NLMS

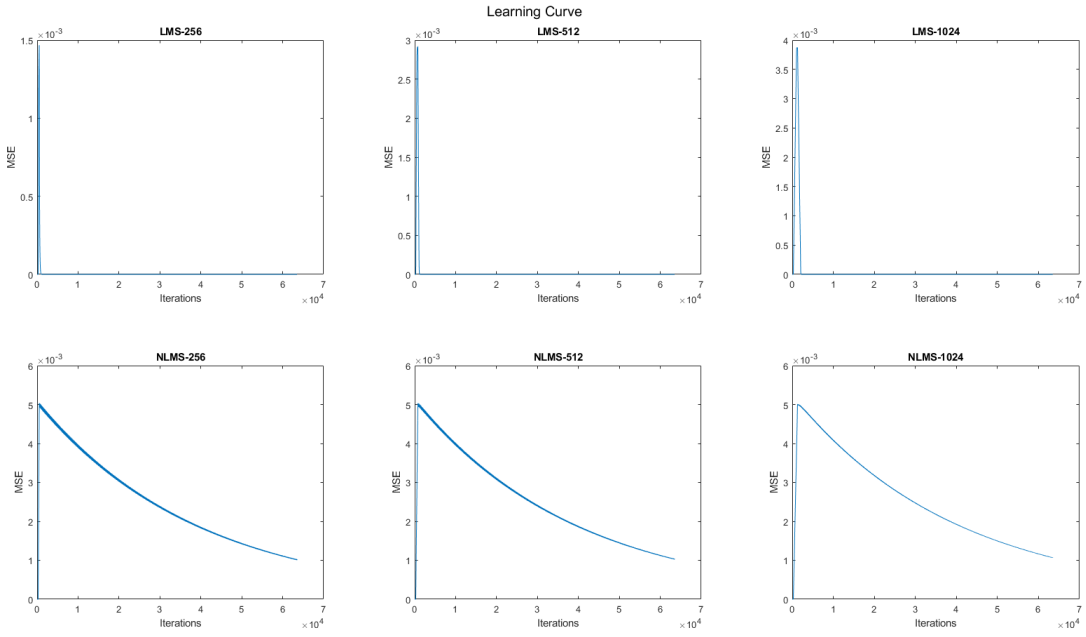

Figure 8: Sinewave Learning Curve from Custom Coded LMS and NLMS

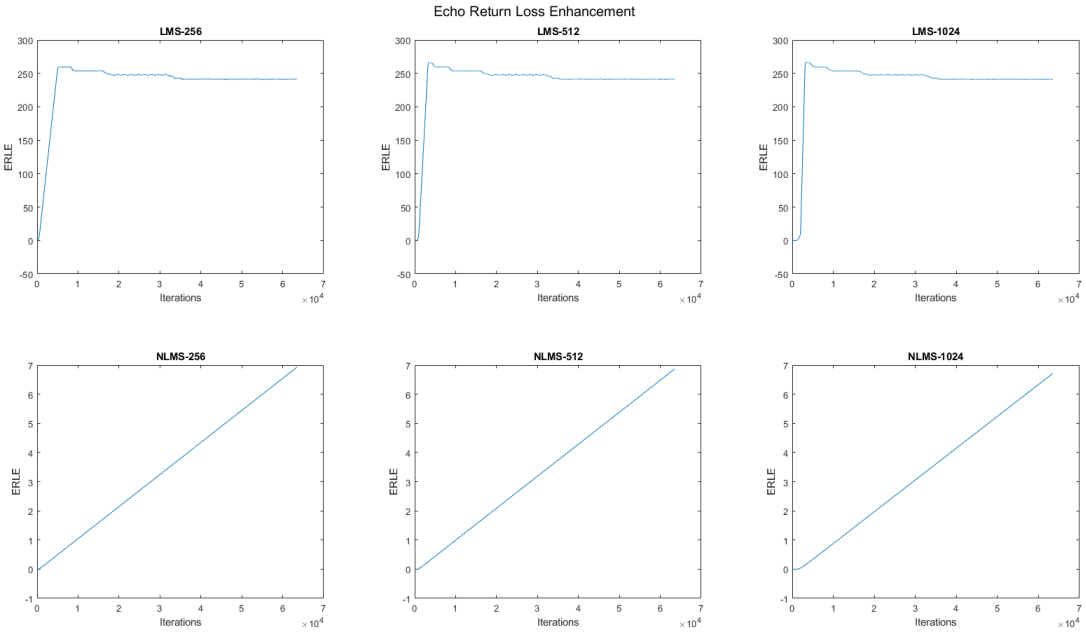

Figure 9: Sinewave ERLE from Custom Coded LMS and NLMS

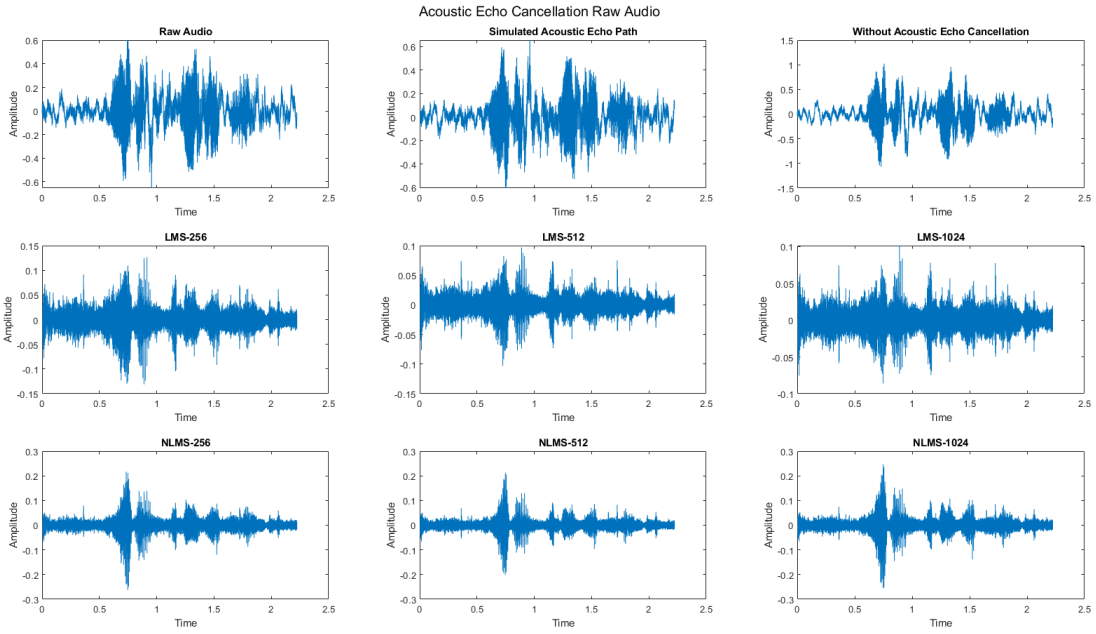

Figure 10: Audio 1 Raw data from Official MATLAB library

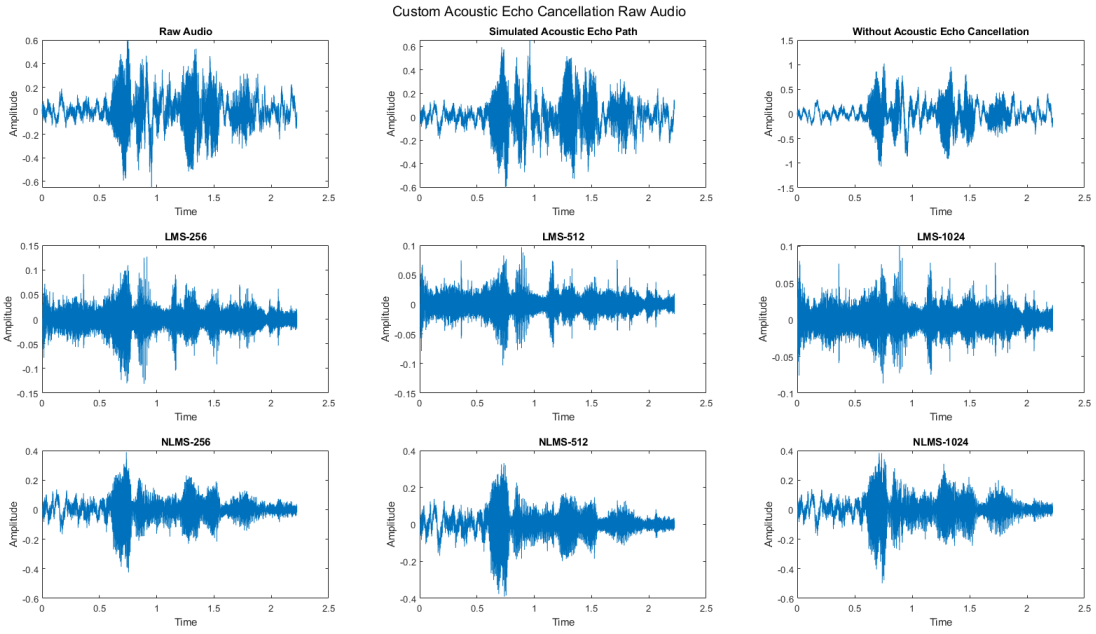

Figure 11: Audio 1 Raw data from Custom Coded LMS and NLMS

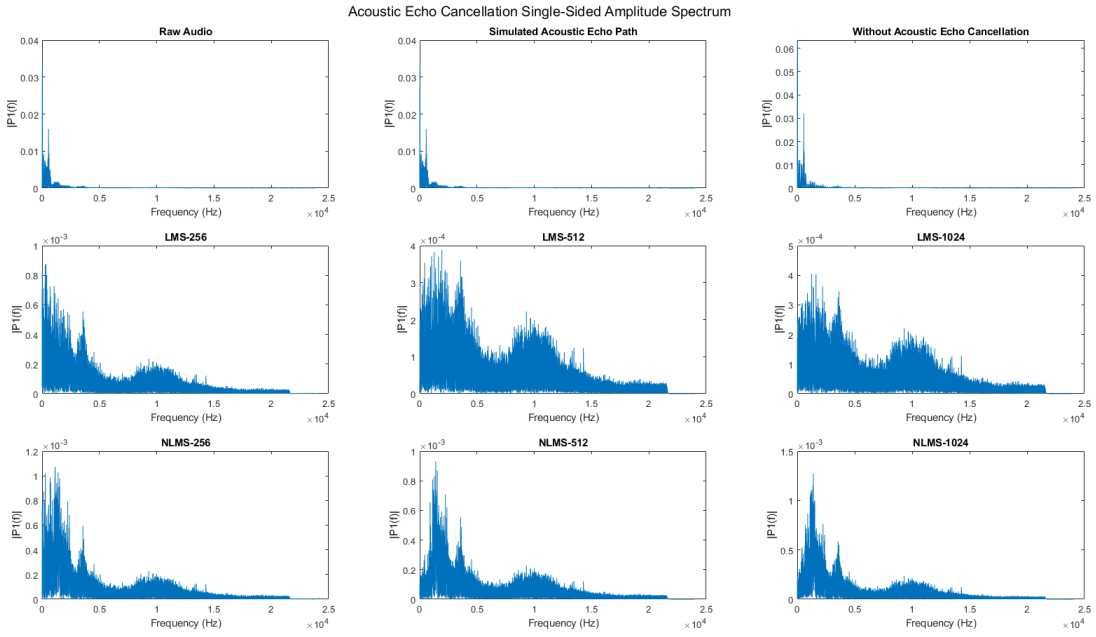

Figure 12: Audio 1 Amplitude Spectrum from Official MATLAB library

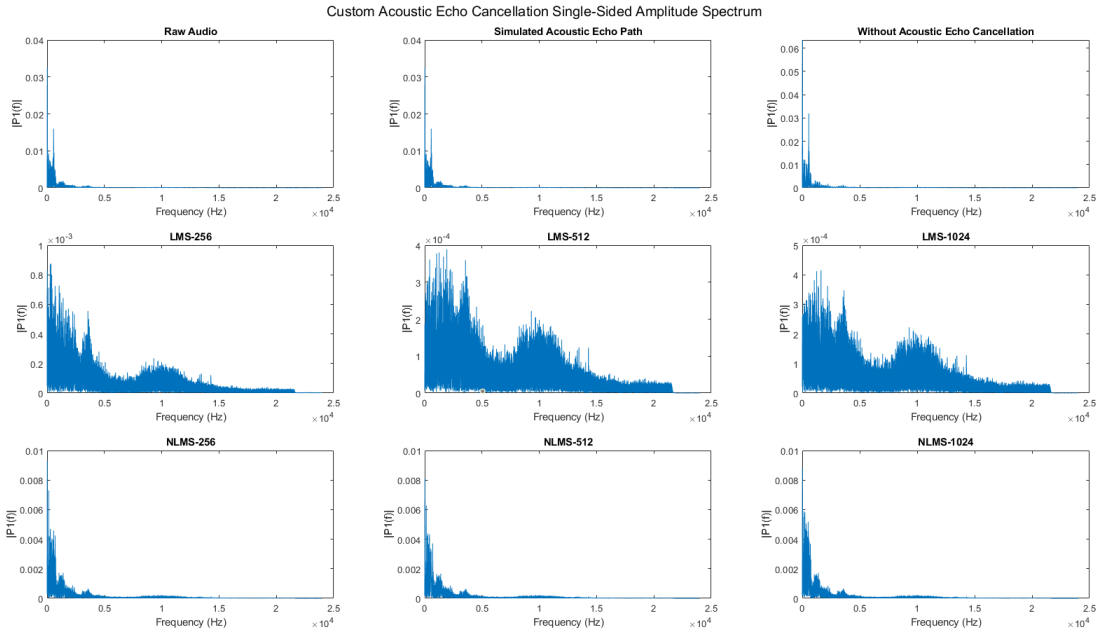

Figure 13: Audio 1 Amplitude Spectrum from Custom Coded LMS and NLMS

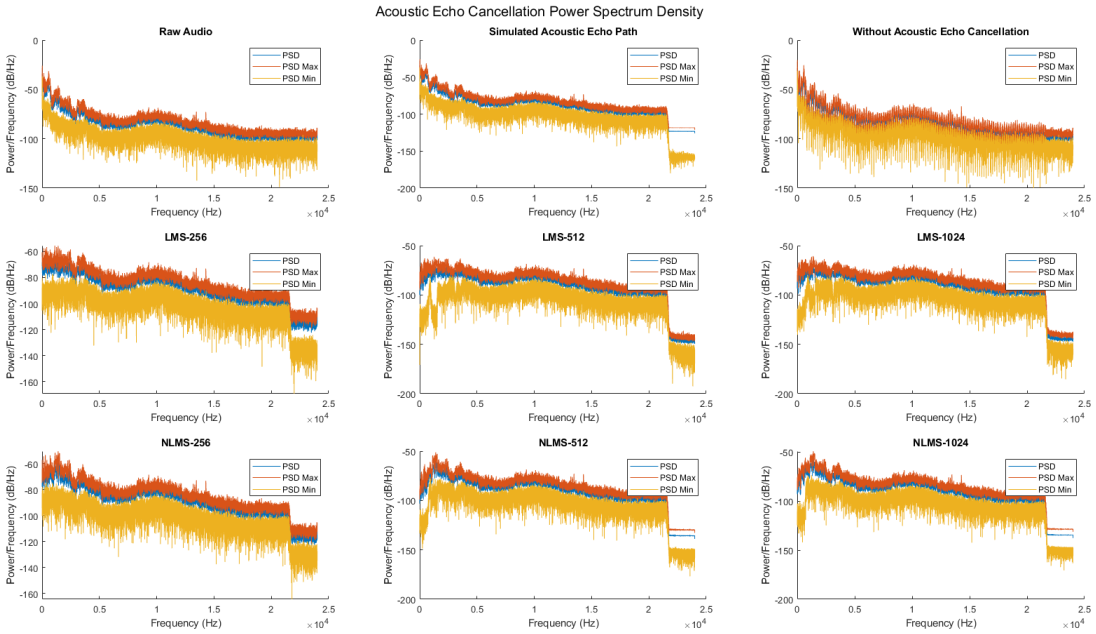

Figure 14: Audio 1 Power Spectrum from Official MATLAB library

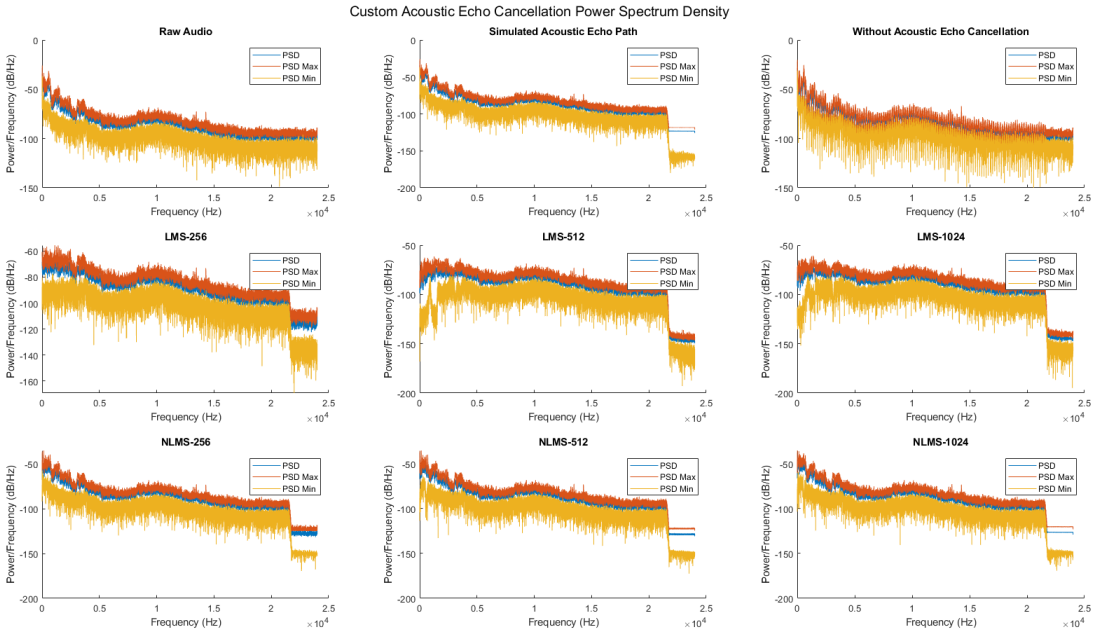

Figure 15: Audio 1 Power Spectrum from Custom Coded LMS and NLMS

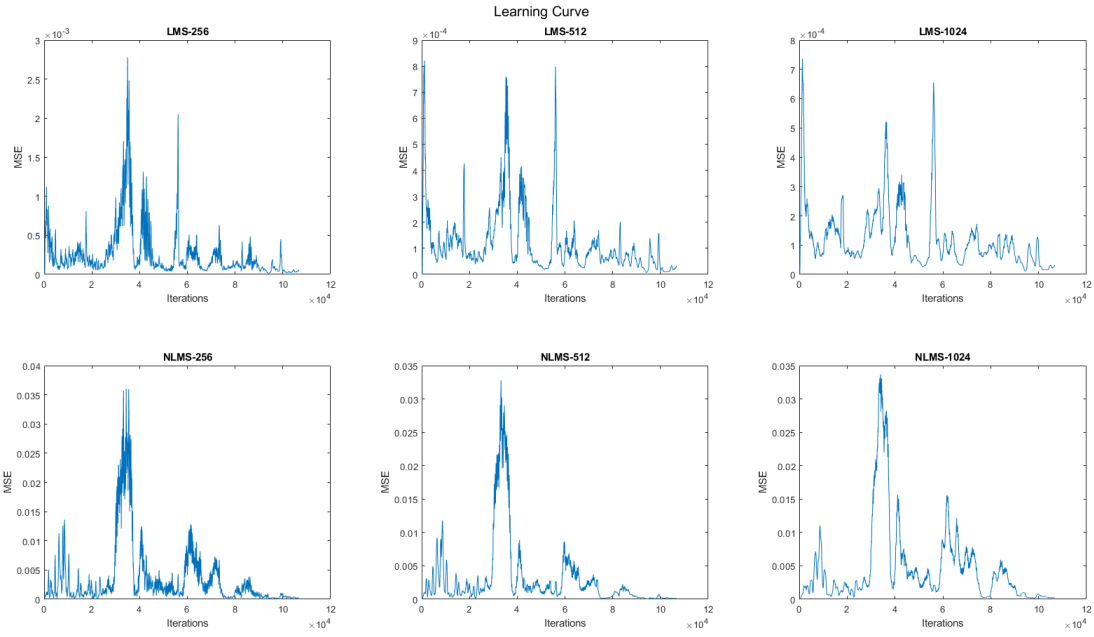

Figure 16: Audio 1 Learning Curve from Custom Coded LMS and NLMS

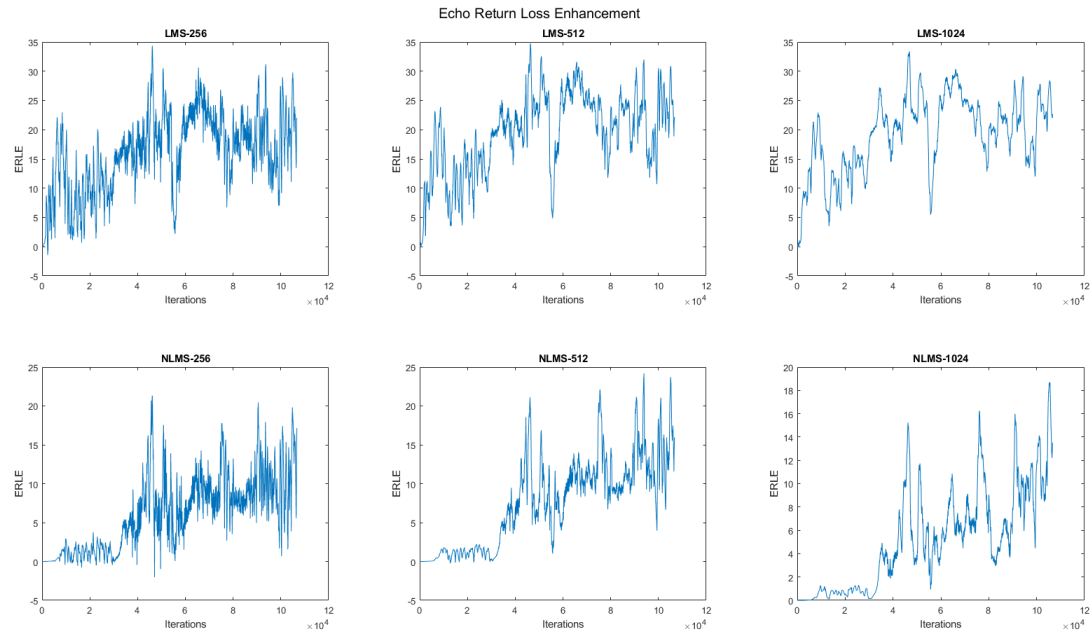

Figure 17: Audio 1 ERLE from Custom Coded LMS and NLMS

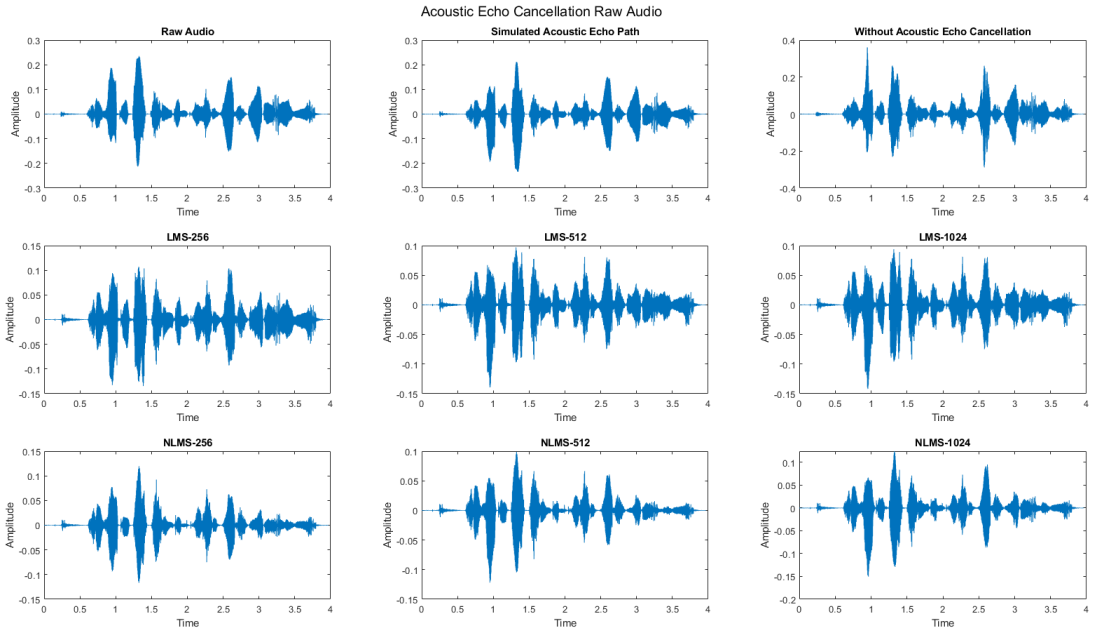

Figure 18: Audio 2 Raw data from Official MATLAB library

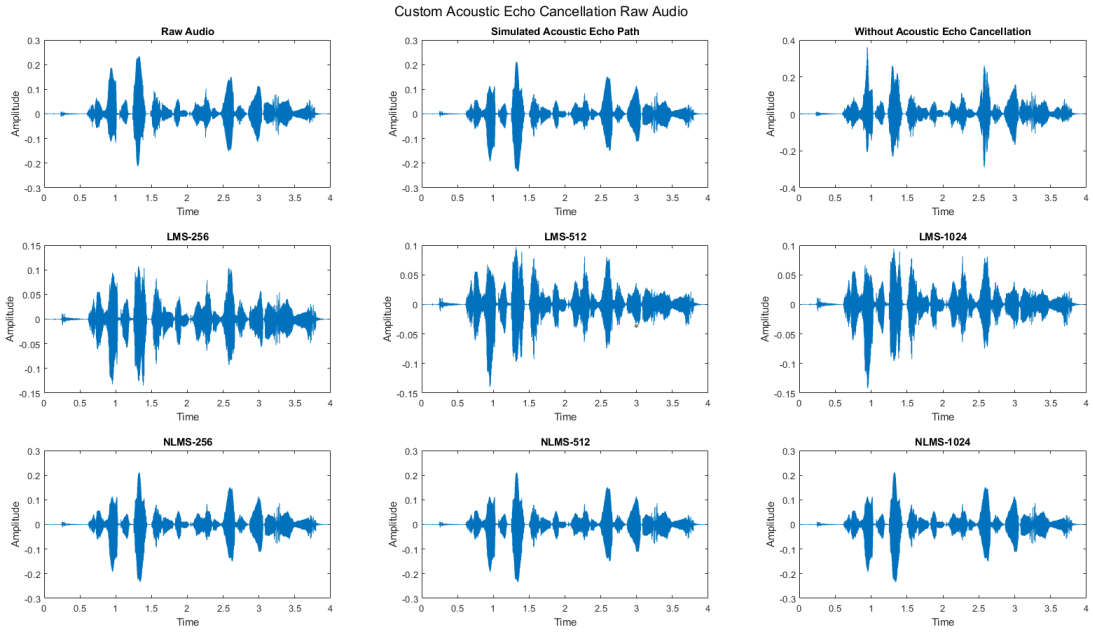

Figure 19: Audio 2 Raw data from Custom Coded LMS and NLMS

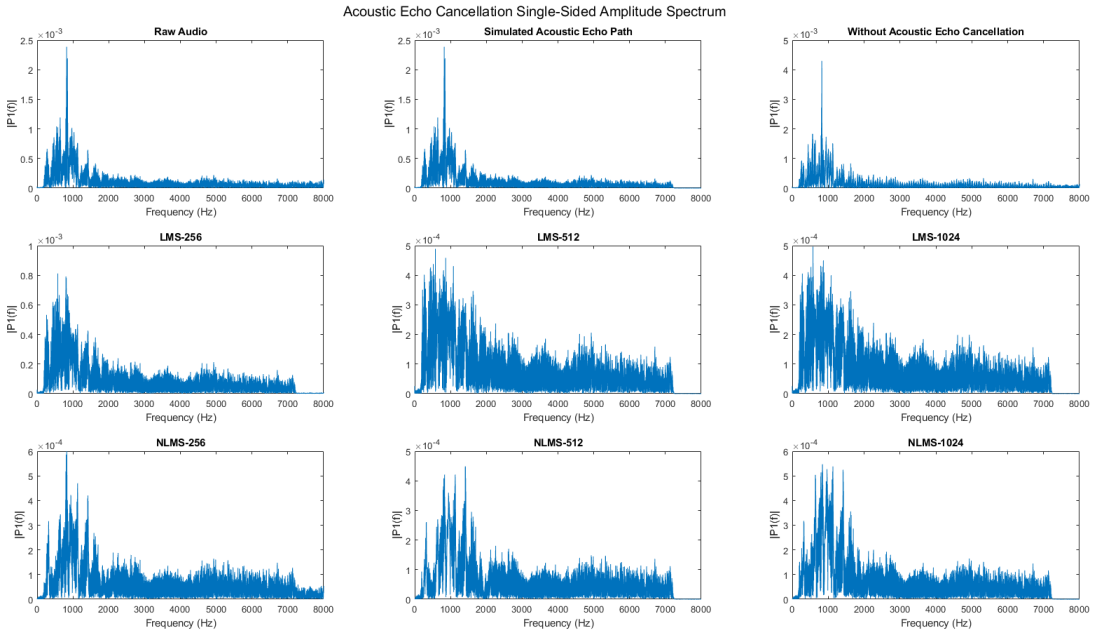

Figure 20: Audio 2 Amplitude Spectrum from Official MATLAB library

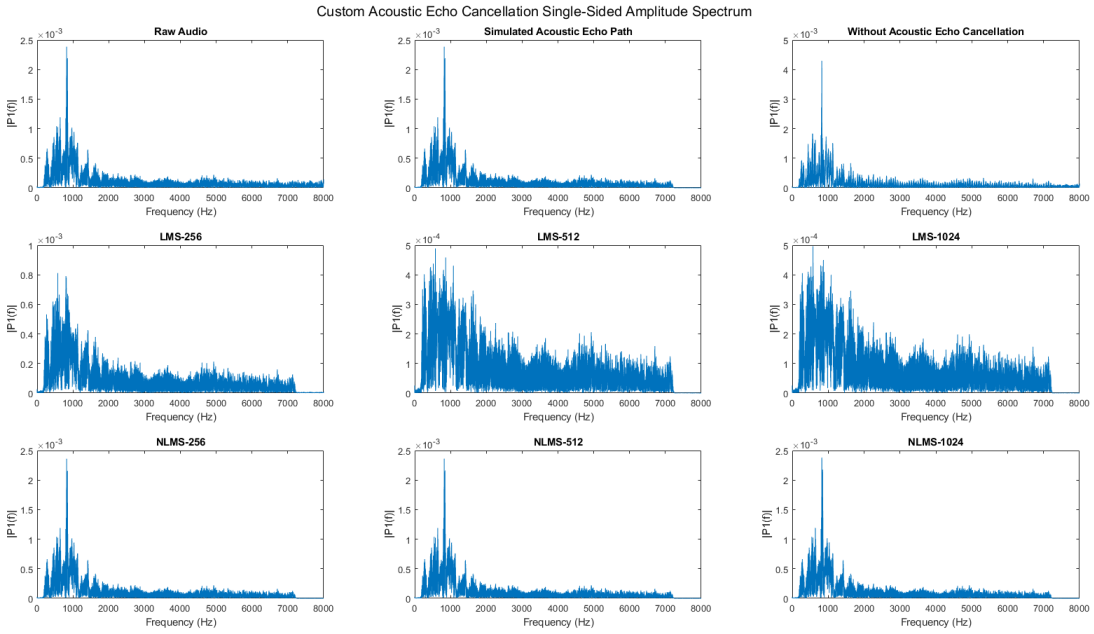

Figure 21: Audio 2 Amplitude Spectrum from Custom Coded LMS and NLMS

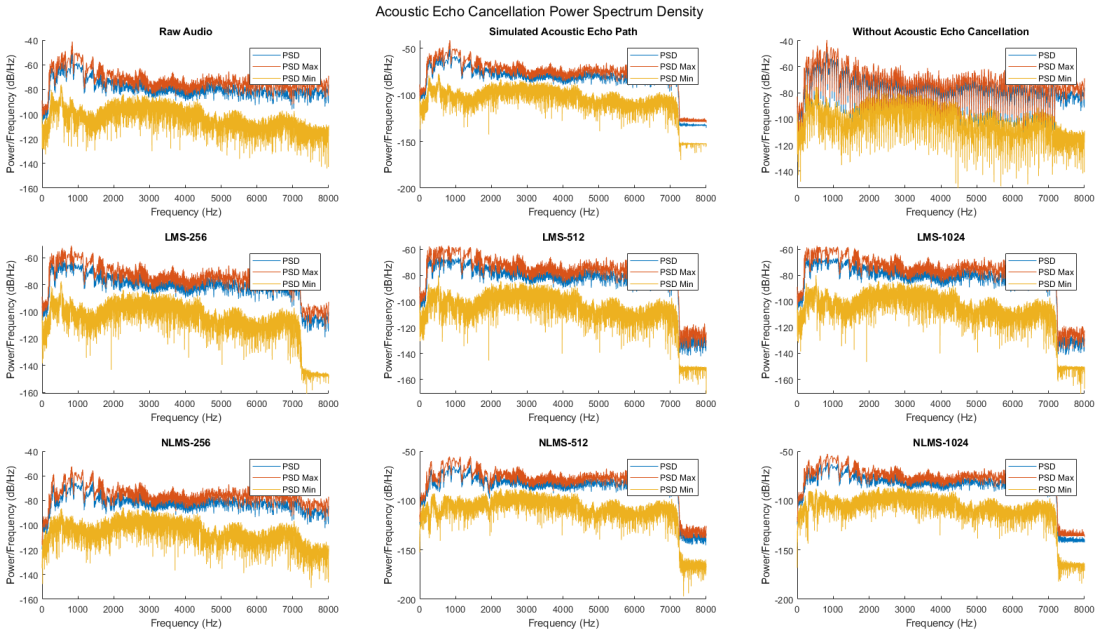

Figure 22: Audio 2 Power Spectrum from Official MATLAB library

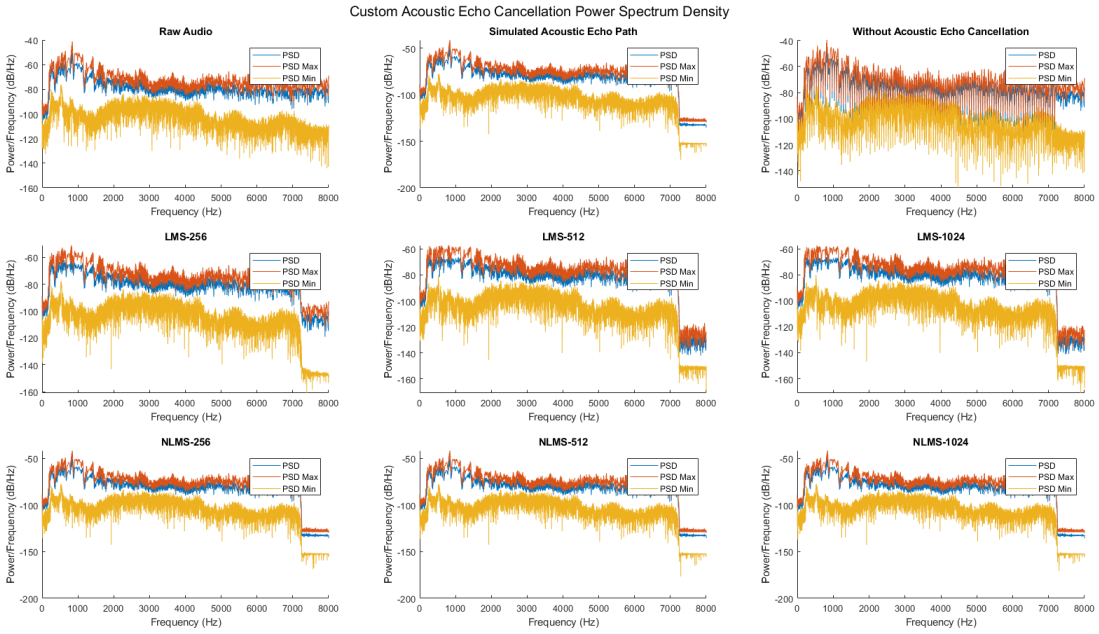

Figure 23: Audio 2 Power Spectrum from Custom Coded LMS and NLMS

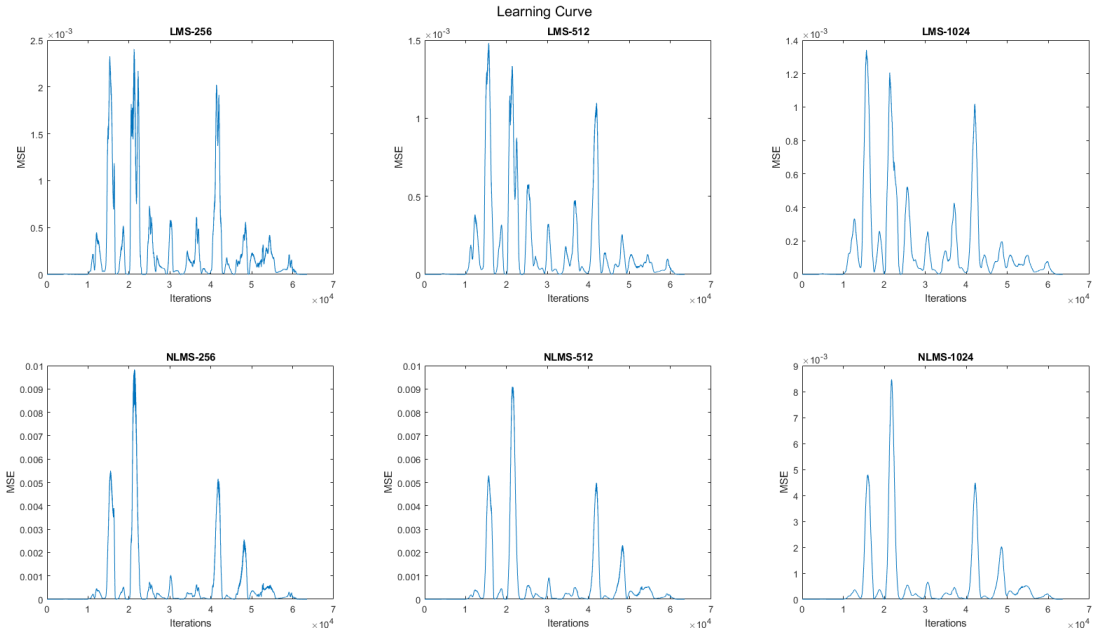

Figure 24: Audio 2 Learning Curve from Custom Coded LMS and NLMS

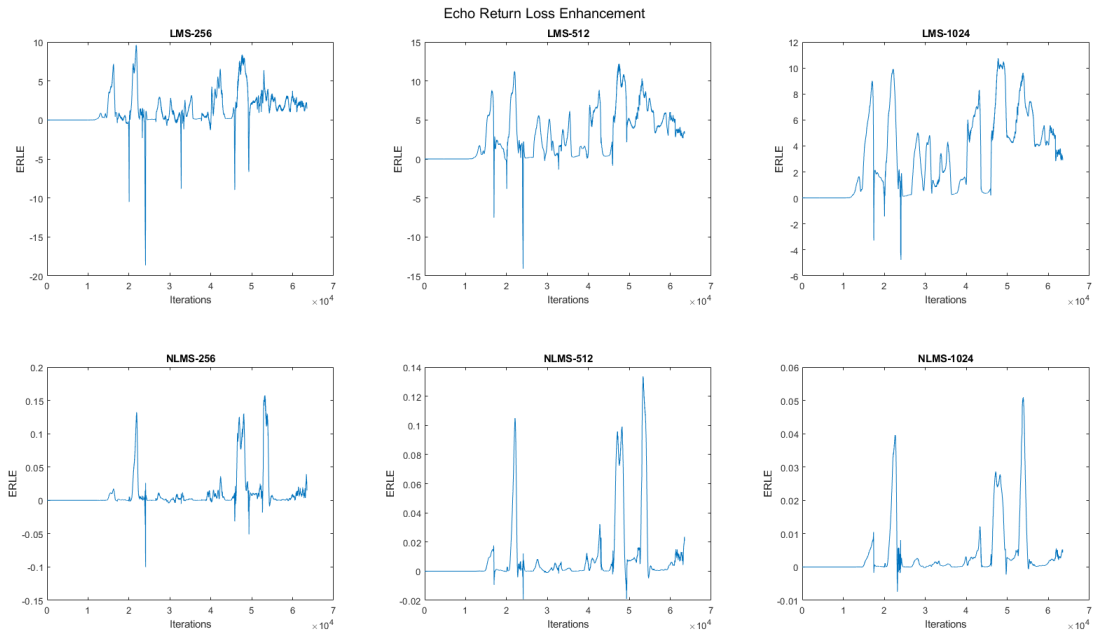

Figure 25: Audio 2 ERLE from Custom Coded LMS and NLMS