BIRZEIT UNIVERSITY
Electrical and Computer Engineering Department
ENCS4310, Digital Signal Processing
Project 1

Leena Affouri - 1200335
Mariam Hamad - 1200837
Raghad Musleh - 1200711

July 2023

Anonymous ACL-IJCNLP submission

1 Introduction

Line echo cancellation is a technique used in telecommunication systems where two users are trying to communicate with each other. When the first user speaks, the signal is transmitted through a channel. In some cases, reflections may occur due to circuit mismatches or imperfections, resulting in an echo. The echo is a copy of the sender's signal that returns to the sender, potentially causing interference and making it difficult for the user to hear the intended audio clearly.

To address this issue, adaptive line enhancement techniques, such as adaptive filters, are used to minimize or eliminate the echo. The adaptive filter aims to make the echo signal zero, effectively canceling the unwanted echo and improving the overall audio quality for the user.

In this project, we will explore various methods and algorithms for line echo cancellation in the context of digital signal processing. We will analyze different approaches and evaluate their performance in terms of echo suppression and audio quality improvement.

2 Problem Specification

Echo cancellation is a crucial problem in communication systems, When the echo returns to the first user, it interferes with the original signal, leading to an unpleasant listening experience, This adaptive filter should be able to adapt to changing channel conditions and provide real-time echo cancellation.

3 Data

To develop and evaluate our line echo cancellation algorithm, I would need a dataset that includes pairs of near-end speech signals, corresponding far-end received signals with echo, echo reference signals, and possibly background noise signals. Having a diverse dataset with variations in speech content, levels, and acoustic conditions will enable us to train and test our algorithm under different scenarios and assess its effectiveness in echo cancellation.

4 Evaluation Criteria

Echo cancellation systems use various metrics to measure performance, such as Echo Return Loss Enhancement, Echo Return Loss, Mean Square Error , Signal-to-Noise Ratio, Perceptual Evaluation of Speech Quality, and Objective Difference Grade. Higher Echo Return Loss Enhancement values indicate better performance, while lower Echo Return Loss values indicate less residual echo leakage or artifacts. Higher Signal-to-Noise Ratio values indicate better echo cancellation performance, as it indicates higher signal quality relative to unwanted components. Overall, these metrics help evaluate the effectiveness of echo cancellation systems in improving speech quality.

5 Approach

Two opposing approaches to the line echo cancellation issue can be taken into consideration: the conventional Filtered-XLMS (FXLMS) algorithm and the more contemporary Deep Learning-based strategy. In order to reduce the error between the estimated and actual echo in the received signal, the FXLMS algorithm estimates the echo and adapts filter coefficients. The Deep Learning-based technique, in contrast, directly learns the mapping from the received signal to the desired signal using training data using deep neural networks. The Deep Learning-based approach gives the capacity to learn complicated mappings but may need more data and processing resources. The FXLMS algorithm is well-established and computationally efficient.

6 Development

The baseline methodology would be used as a comparison when evaluating these advances. The performance gains can be evaluated using metrics like ERLE, ERL, MSE, SNR, PESQ, and ODG. It's also crucial to keep an eye out for any unanticipated side effects, including a rise in computing complexity or any artifacts brought on by the improvements.

7 Results and Analysis

```
%PART A
%1- IMPULSE RESPONSE
load('path.mat');
figure;
plot(path);
title('Impulse Response of Echo Path');
xlabel('Sample Index');
ylabel('Amplitude');

%2- FREQUENCY RESPONSE
fs = 8000;  % 8 kHz
figure;
[b, a] = freqz(path, 1,[], fs);
magnitude_response = abs(b);
f = linspace(0, fs/2, length(b));
plot(f, 20*log10(magnitude_response));
title('Frequency Response of Echo Path');
xlabel('HZ');
ylabel('db');
```

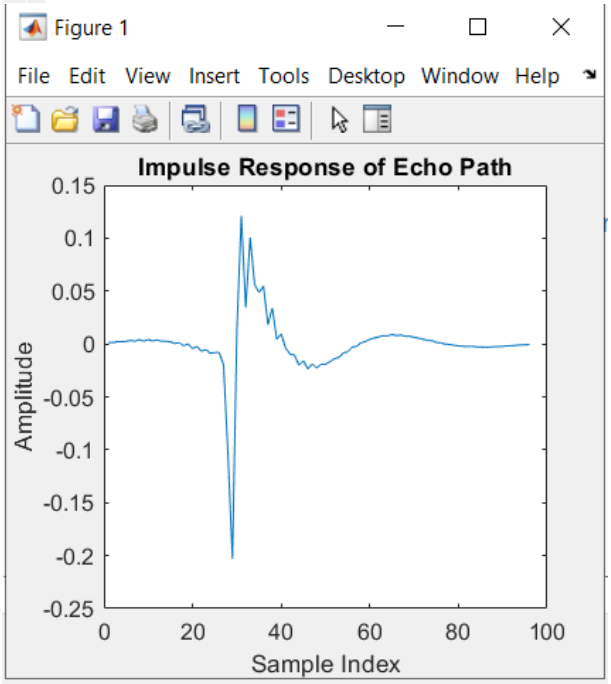


Figure 1: Impulse Response of Echo Path

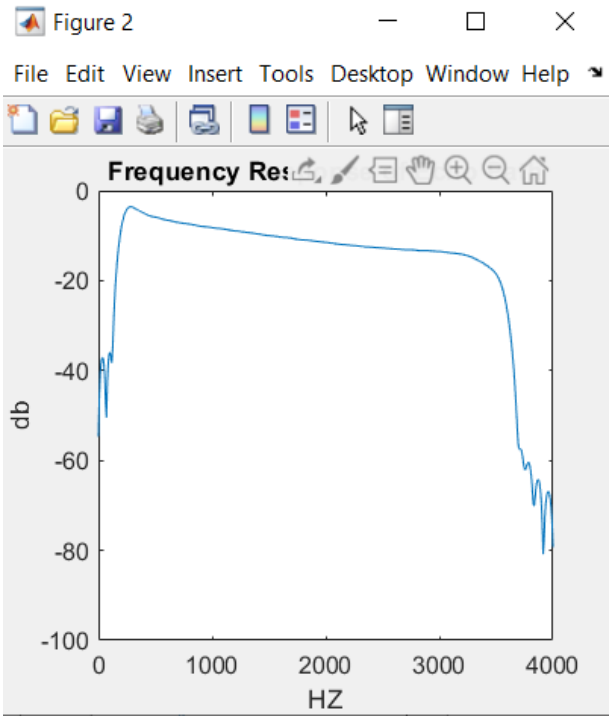


Figure 2: Frequency Response of Echo Path

Part A: Xe = h[n]*x[n] To find the echo we have to do convolution between the path which is h[n] with fare end signal, the result for them is the echo, as we can see from the figures there is the impulse response of the echo path and the frequency response.

```
% PART B
load('css.mat');
% Plot CSS Data
figure;
plot(css);
title('CSS Data (Composite Source Signal)');
xlabel('Time');
ylabel('Amplitude');
% plot the Power Spectrum Density (PSD)
figure;
x = css;
nfft = 512;
window = hamming(nfft);
Pxx = pwelch(x, window);
f = linspace(0, fs/2, length(Pxx));

plot(f, 10*log10(Pxx));
xlabel('Frequency (Hz)');
ylabel('PSD (dB)');
title('PSD for CSS');
```

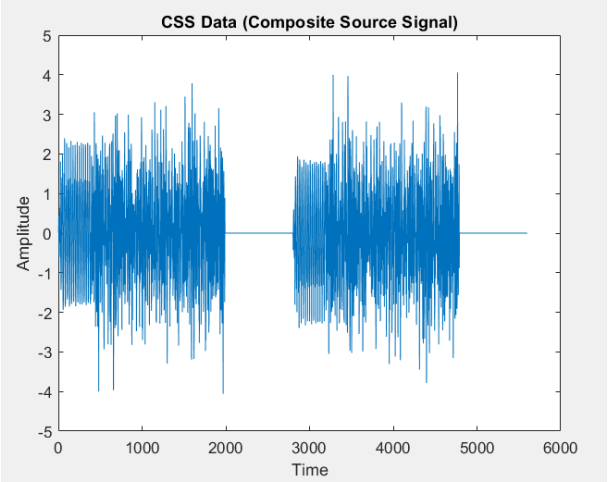


Figure 3: CSS Data (Composite Source Signal)

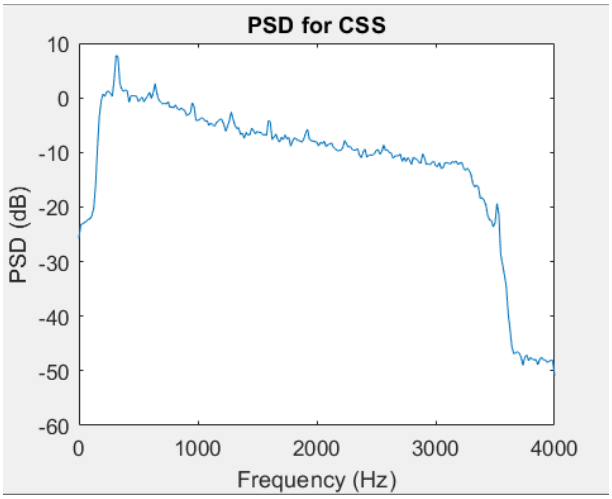


Figure 4: CSS Data (Power Spectrum Density)

Part B: the css signal which simulates properties of speech, we first concatenate the signal 5 times then we do convolution for the sound that contains 5600 sample with the channel, and to find the spectrum we take the square of the sample

```
%part c
% Length of echo path impulse response
N = length(path);
% Concatenate five blocks of css
css_concat = repmat(css, 1, 5);
% Apply echo path to the concatenated signal
echo_signal = conv(css_concat, path);
% Plot the far-end signal
figure;
plot(css_concat);
title('Far-End Signal');
xlabel('Sample Index');
ylabel('Amplitude');
% Plot the echo signal
figure;
plot(echo_signal);
title('Echo Signal');
xlabel('Sample Index');
ylabel('Amplitude');
ylim([-5, 5]);  % Set the y-axis limits to -5 and 5
P_input = 10 * log10(sum(css_concat.^2) / length(css_concat));
P_output = 10 * log10(sum(echo_signal.^2) / length(echo_signal));
% Calculate echo-return-loss (ERL) in dB
ERL = P_input - P_output;
% Display input power, output power, and ERL on the screen
disp(['Input Power: ', num2str(P_input, '%g'), ' dB']);
disp(['Output Power: ', num2str(P_output, '%g'), ' dB']);
disp(['ERL: ', num2str(ERL, '%g'), ' dB']);
```

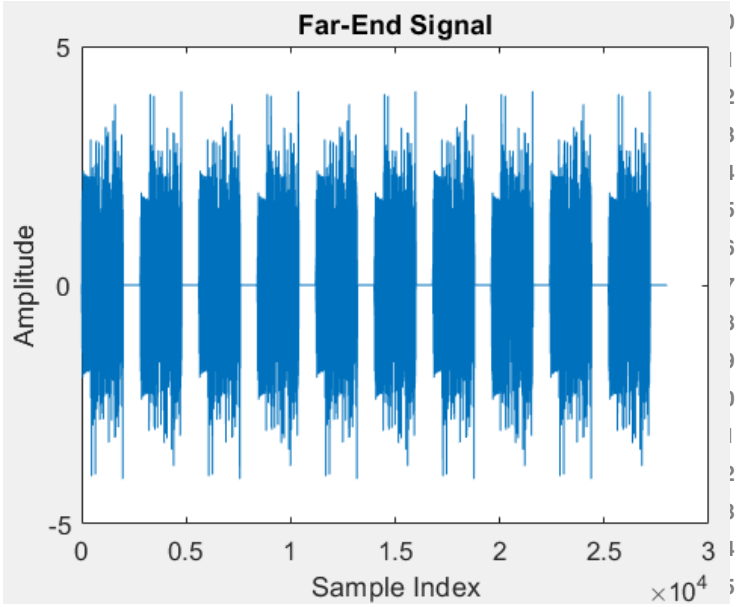


Figure 5: Far-End Signal

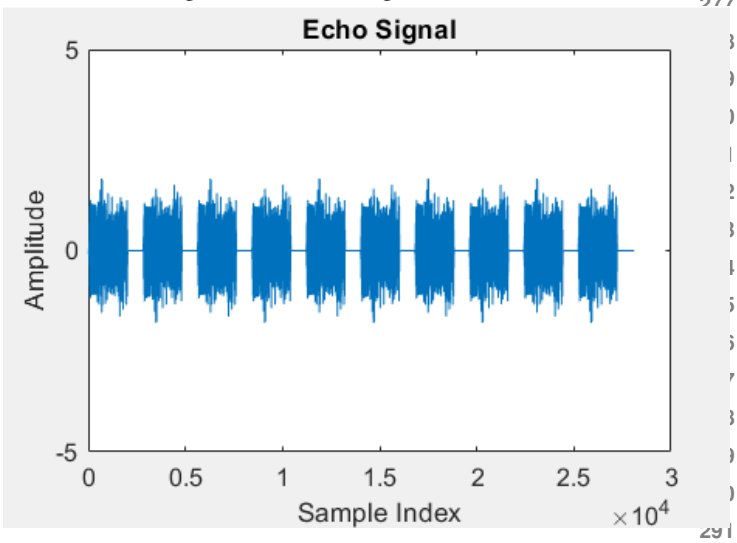
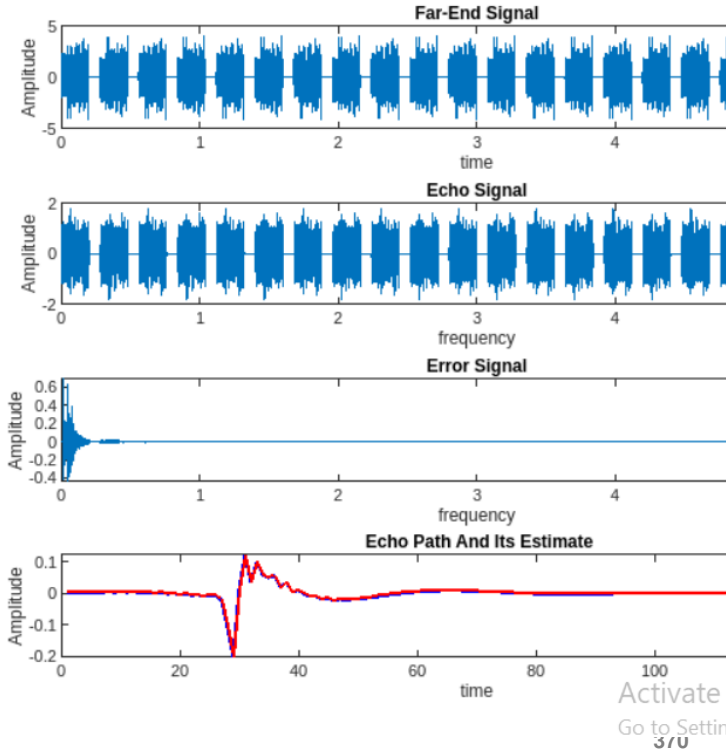


Figure 6: Echo Signal

Part C: we get X Concatenate which is x repeated 5 times to the echo path that we load it in part a and the result is echo signal which is x Concatenate convolution with h[n] then depend on the given equation we find the input and output power after putting N =5600*5 and finally we find echo-return-loss which is the difference between power input and output

```
>> dsppart22
Input Power: 1.73579e-14 dB
Output Power: -6.34556 dB
ERL: 6.34556 dB
```

```
%part D
tap_count = 128;
mu = 0.25;
epsilon = 6 * 10^-10;
far_end_signal = repmat(css, 1, 10);
echo_signal = conv(path,far_end_signal);
w = zeros(tap_count, 1);
y = zeros(1, length(far_end_signal));
e = zeros(1, length(far_end_signal));
for n = tap_count+1:length(far_end_signal)
    x = far_end_signal(n:-1:n-tap_count+1)';
    y(n) = w' * x;
    e(n) = echo_signal(n) - y(n);
    w = w + (1 / (epsilon + x' * x)) * x * e(n);
end
figure;
subplot(4, 1, 1);
plot(far_end_signal);
title('Far-End Signal');
xlabel('time');
ylabel('Amplitude');
subplot(4, 1, 2);
plot(echo_signal);
title('Echo Signal');
xlabel('frequency');
ylabel('Amplitude');
subplot(4, 1, 3);
plot(e);
title('Error Signal');
xlabel('frequency');
ylabel('Amplitude');
subplot(4, 1, 4);
plot(1:length(path),path,'b','LineWidth',1.5);
hold on;
plot(1:tap_count, w , 'r','LineWidth',1.5);
title(' Echo Path And Its Estimate');
xlabel('time');
ylabel('Amplitude');
legend('Given', 'Estimsated');
```



Part D: here we need 10 blocks of CSS data we also enter it into the echo path, then we do an estimation for the filter(we use e-NLMS filter) which contains 128 taps, and the output of the filter y=w*x We don't have the value for w so we estimate it to make the difference between the output of the filter and echo get clean signal Adaptive Algorithms this algorithm help to find w using input and error

```
%PART E
[b, a] = freqz(w, 1,[], fs);
magnitude_response1 = abs(b);
phase_response1 = unwrap(angle(b));
[n, m] = freqz(path, 1,[], fs);
magnitude_response2 = abs(n);
phase_response2 = unwrap(angle(n));
f1 = linspace(0, 1, length(unwrap(magnitude_response
f2 = linspace(0, 1, length(unwrap(phase_response1)))
figure;
subplot(2, 1, 1);
plot(f1, 20*log10(magnitude_response1), 'b', 'LineWidth', 3);
hold on;
plot(f1,20*log10(magnitude_response2) , 'r', 'LineWidth', 1.5);
xlabel('frequency');
ylabel('Amplitude');
title('Amplitude Response');
legend('Estimated', 'Given');
subplot(2, 1, 2);
plot(f2,phase_response1,'b','LineWidth',3);
hold on;
plot(f2,phase_response2,'r','LineWidth',1.5);
xlabel('frequency');
ylabel('Phase');
title('Phase Response');
legend('Estimated', 'Given');
```

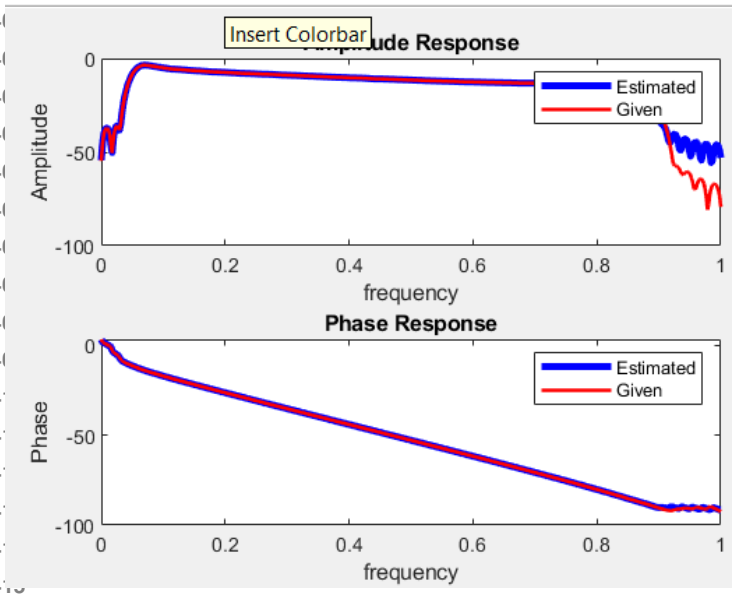


Figure 7: Impulse and phase response

Part E: we plot the phase and amplitude response for the filter then we compare it with the path they have to be the same as the input signal or close to it.

8 Conclusions

This project offered insightful solutions for the line echo cancellation issue. It proved the value of comparative analysis, the influence of improvements, the significance of evaluation metrics, and the necessity of taking trade-offs and probable negative impacts into account. Future researchers and practitioners might create efficient echo cancellation systems that are suited to certain requirements and limits by understanding these factors.

9 References

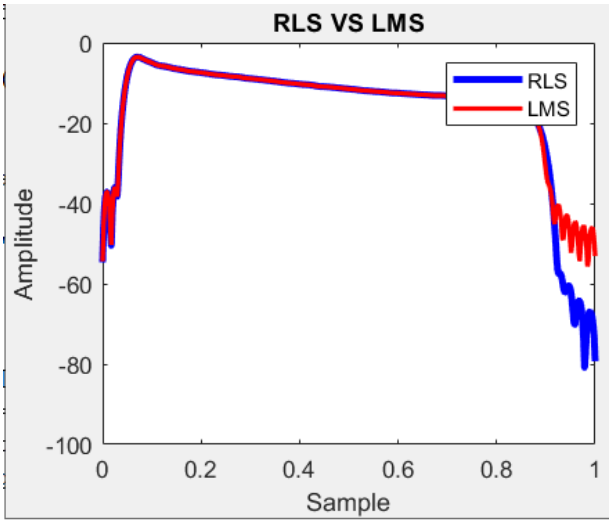
- MATLAB File Exchange: https://www.mathworks.com/search.html?c%5B%5D=entire_site&q=path.mat&page=1
- MATLAB File Exchange : https://www.mathworks.com/search.html?c%5B%5D=entire_site&q=css.mat&page=1
- MATLAB Documentation - pspectrum Function. Retrieved from: https://www.mathworks.com/help/signal/ref/pspectrum.html

  https://www.youtube.com/watch?v=SVe9twn-w7A


```
% Part F
order = 128;
lambda = 0.99;
delta = 1e-3;
j = zeros(order, 1);
P = eye(order) / delta;
x =  far_end_signal ;
d = conv(path,x);
x_n = zeros(order, 1);
y = zeros(1, length(far_end_signal));
e = zeros(1, length(far_end_signal));
for n = 1:length(x)
    x_n = [x(n); x_n(1:end-1)];
    y(n) = j' * x_n;
    e = d(n) - y(n);
    K = P * x_n / (lambda + x_n' * P * x_n);
    j = j + K * e;
    P = (P - K * x_n' * P) / lambda;
end

figure;
[b, a] = freqz(j, 1,[], fs);
magnitude_response1 = abs(b);
f1=linspace(0,1,length(magnitude_response1));
plot(f1, 20*log10(magnitude_response1), 'b', 'LineWidth', 3);
hold on;
[b, a] = freqz(w, 1);
magnitude_response1 = abs(b);
f1=linspace(0,1,length(magnitude_response1));
plot(f1, 20*log10(magnitude_response1), 'r', 'LineWidth', 2);
xlabel('Sample');
ylabel('Amplitude');
title('RLS VS LMS');
legend('RLS', 'LMS');s
```

Figure 8: RLS AND LMS



Part F: we do the implementation for LRS and compare it with the e-NLMS filter.