Line Echo Cancellation

Department of Electrical and Computer Engineering

Birzeit University

Group members:
Abdullah Sami Naser, 1201952
Ahmad Wasel Ghanem , 1201954
Mohammad M. Jarrar, 1201726

Submitted to: Dr. Qadri Mayalla

***Abstract*— This project explores the use of the Normalized Least Mean Square (e-NLMS) algorithm in a Line Echo Canceller (LEC) implemented in MATLAB. The study focuses on the algorithm's performance in echo cancellation, demonstrating its potential for improving telecommunication systems' efficiency.**

I. Introduction

Digital Signal Processing (DSP) is a critical component of many modern technologies and industries, including telecommunications, radar, audio and video processing, biomedical engineering, and more. It involves the manipulation and analysis of signal data, which can be in the form of sound, images, or any other type of signal that can be digitally encoded. The primary goal of DSP is to improve the quality, reliability, and efficiency of signal data transmission or storage. It achieves this through various techniques such as filtering, compression, error detection and correction, and modulation.

DSP operates by converting analog signals into digital form through a process known as sampling. Once in digital form, these signals can be processed using mathematical operations. This digital representation allows for more precise control and manipulation of the signal, leading to enhanced performance and functionality. Key concepts in DSP include the Fourier Transform, Z-Transform, and Digital Filter Design, which provide the mathematical foundations for signal processing. As technology continues to advance, DSP is becoming increasingly important, driving innovations in areas such as machine learning, data science, and the Internet of Things (IoT). [1]

MATLAB, a high-level programming language and interactive environment, is widely used in Digital Signal Processing (DSP) for its powerful computational and visualization capabilities. It provides a range of built-in functions and toolboxes specifically designed for signal processing tasks, such as filtering, spectral analysis, and waveform generation. These tools allow engineers and researchers to quickly prototype algorithms, simulate systems, and analyze the performance of DSP techniques. With its ability to handle complex numerical computations and its extensive visualization features, MATLAB has become an indispensable tool in the field of DSP. [2]

This project aims to using MATLAB programming in order to design and implement a line echo canceller system that. improves the efficiency of telecommunication systems by removing the reflected far-end signal as an echo.

II. Problem Specification

In telecommunication systems, the presence of echo signals significantly degrades the quality of voice communication. This echo is primarily caused by the impedance mismatch at the hybrid junction (circuit mismatch), which leads to the reflection of the voice signal back to the sender. The echo signals can cause considerable delay and distortion, because these signals interfere with the received signal leading to a poor user experience.

The challenge is to design and implement an effective Line Echo Canceller (LEC) system that can mitigate these echo signals. The LEC system should be capable of identifying and eliminating the echo components from the received signal, thereby enhancing the overall quality of the communication.

For this project, the Normalized Least Mean Square (e-NLMS) algorithm will be used for the LEC system. The e-NLMS algorithm is known for its superior convergence speed and echo cancellation performance. However, the implementation and optimization of the e-NLMS algorithm in a real-time system present a significant challenge.

The project will involve the design, simulation, and performance analysis of the e-NLMS based LEC system in MATLAB. The performance of the system will be evaluated based on metrics such as the Power Spectral Density (PSD) of the input and output signals analysis, Echo Return Loss (ERL) and many other methods introduced in the approach part.

The ultimate goal of this project is to develop an efficient and reliable LEC system that can significantly improve the quality of voice communication in telecommunication systems.

III. DATA

. Input data are couple (.mat) files. The first one is (css.mat) which contains synthetic signal representing a composite source signal **x[n]** as 5600 sample that emulates the properties of speech. Specifically, it contains segments of pause, segments of periodic excitation and segments with white-noise properties. Note that all signals in this project are sampled at 8kHz. The second one is (path.mat) which contains the impulse response of the typical echo path **h[n].** see Fig.1. The signals generated during implementation will be the echo signal $\mathbf{x_e[n]}$, the estimated echo signal **y[n]**, the error signal **e[n]** and the estimated echo path **w[n]** that provided by the adaptive filter.

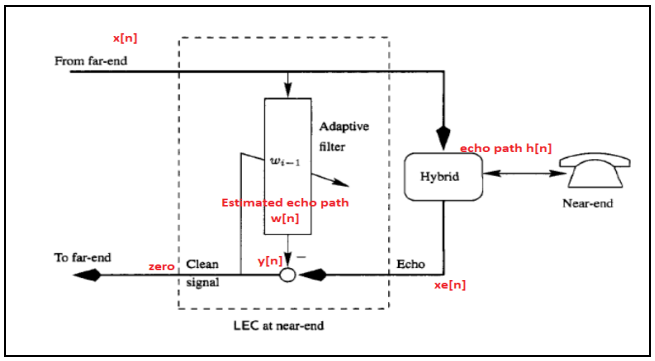

Figure 1: Adaptive Filter Components

IV. EVALUATION CRITERIA

The evaluation of the project is mainly based on the clarity of the output signal. We got better results successively through doing more research of the available functions in MATLAB that helps us achieve our goal more accurately. The first results we got were incorrect due to high randomness of the output signal that completely was so far from the original signal which caused high data loss, after careful consideration of the functions used and better understanding of LEC systems and adaptive filters we got the desired results we were satisfied with, we knew that these results were correct because the estimated signal almost matched the original signal, as that accurate result referred to that the adaptive filter w[n] was almost equal to hybrid signal h[n] which gives us clearest signal you can have.

Staging in methods even make results more accurate throw working. At the end of the work on the project, the results were so satisfying comparing to the results and accuracy at earlier stages of working on the project.

V. APPROACH

In this study, an adaptive filtering approach was employed to estimate and compensate for the echo introduced by the FIR channel. The algorithm of choice was the Normalized Least Mean Squares (LMS) filter, implemented using the **dsp.LMSFilter** object in MATLAB. The LMS filter was selected due to its simplicity and effectiveness in adaptive filtering applications.

A. Echo Path Analysis

The impulse response sequence of the echo path was loaded from the file **path.mat**. The impulse response represents the time-domain characteristics of the echo path. It was plotted using the **stem** function, providing insights into the duration and magnitude of the echoes. The frequency response of the echo path was computed using the **freqz** function, revealing the gain and phase characteristics across different frequencies. See Result and Analysis A and Appendix A.

B. Composite Source Signal (CSS) Analysis

The composite source signal (CSS) data was loaded from the file **css.mat**. This synthetic signal emulates the properties of speech and consists of segments with pause, periodic excitation, and white-noise properties. The individual samples of the CSS data were plotted, providing a visual representation of the signal waveform. The power spectrum density (PSD) of the CSS data was computed and plotted, revealing the frequency content and distribution of power across different frequency components using **pwelch** function. See Result and Analysis B and Appendix B.

C. Echo Signal Generation and Power Estimation

To simulate the echo signal, five blocks of the CSS data were concatenated using **repmat** function to form an extended input signal. This extended signal was then fed into the echo path by convolving it with the impulse response sequence obtained in Part A using **conv** function. The resulting echo signal was plotted, illustrating the presence of echoes and their characteristics in the time domain.

The input and output powers of the echo signal were estimated in decibels (dB) to quantify the signal levels. The power of a signal was computed as the squared magnitude of the signal samples using **sum** function. The input power was estimated by computing the power of the concatenated CSS signal. Similarly, the output power was estimated by computing the power of the resulting echo signal. The power values were then converted to dB using the appropriate logarithmic scaling. See Result and Analysis C and Appendix C.

D. Adaptive Filtering Approach

The first step involved preparing the input signals. The **far_end_signal** was generated by replicating a Common Source Signal (CSS) multiple times, representing the desired signal. The **echo_signal** was obtained by convolving the **far_end_signal** with the known FIR system (Path), simulating the echo signal.

The LMS filter was then initialized with appropriate parameters. The filter l**ength** was set to **128 taps**, providing sufficient degrees of freedom to estimate the FIR channel. A **step size** of **0.25** was chosen to control the adaptation rate, balancing convergence speed and stability. The '**Normalized LMS**' method was selected to normalize the step size, ensuring robustness to variations in input signal power.

The adaptive filtering process was performed using the **step** method of the LMS filter object. By passing the

far_end_signal and **echo_signal** as input arguments, the LMS filter iteratively updated its coefficients to minimize the error between the estimated and actual echo signals. The estimated echo signal (**y**) and the error signal (**e**) were obtained as outputs.

To evaluate the performance of the adaptive filter, the estimated FIR channel coefficients were analyzed. These coefficients were extracted from the LMS filter using the Coefficients property **w**. The amplitude response and phase response of the estimated FIR channel were computed using the **freqz** function, providing insights into the frequency-dependent gain and phase shift introduced by the FIR channel.

Additionally, the estimated impulse response of the FIR channel was plotted using the **stem** function. This visualization enabled a closer examination of the temporal characteristics of the FIR channel, including any delays, magnitude changes, or filtering effects it introduced to the signals passing through it.

By comparing the estimated FIR channel with the known FIR system **(Path)**, it was possible to assess the effectiveness of the adaptive filtering approach. The aim was to minimize the error between the estimated and actual echo signals, thereby reducing the echo and improving the overall signal quality. See Result and Analysis D and Appendix D.

E. Using Another Adaptive Algorithm

RLS algorithm was chosen to be the second adaptive algorithm. The same procedure made on NLMS was made on it. Finally, the results were compared. See Result and Analysis E and Appendix E for additional information about this algorithm.

VI. Results and Analysis

A. Echo Path Analysis:

The first subplot displays the impulse response of the echo path. The impulse response sample index, after using stem function we got the results.as shown in Fig.2

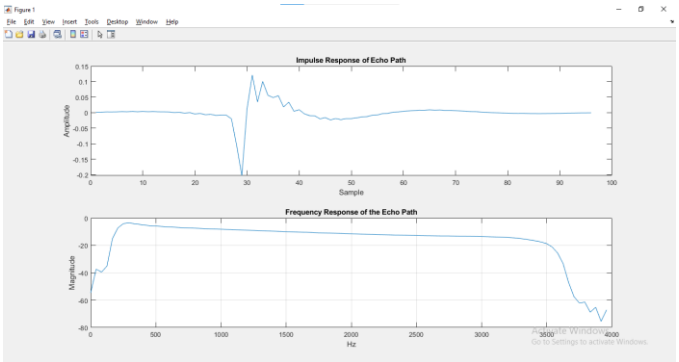

Figure 2. Echo Path Analysis

The second subplot computes the frequency response of the echo path using the **freqz** function. The frequency response shows how the system responds to different frequencies.

These plots help analyze and understand the behavior of the echo path system.

B. Composite Source Signal (CSS) Analysis

The first figure displays the samples of the CSS data. It shows the amplitude of the signal over the sample index. This plot provides a visual representation of the CSS waveform as sown in Fig.3

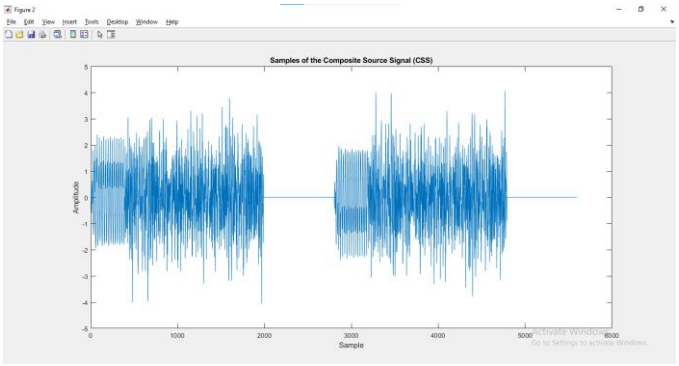

Figure 3 Composite Source Signal (CSS) Plot

The second figure calculates and plots the Power Spectral Density (PSD) of the CSS signal. The PSD represents the distribution of power across different frequencies in the signal By using the **pwelch** function, the code estimates the PSD, shown in Fig.4:

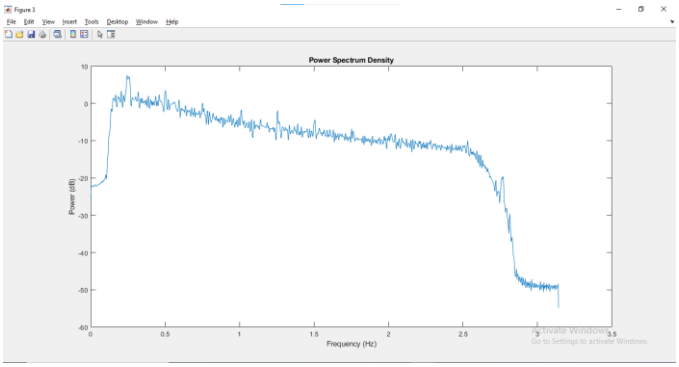

Figure 4: Power Spectral Density (PSD) Plot

C. Echo Signal Generation and Power Estimation

To simulate CSS concatenation of 5 blocks we used "**concatenated_signal = repmat(css, 1, 5);"** that creates a new signal by repeating the CSS signal 5 times consecutively,

To simulate the echo effect, we convolved the concatenated signal with the impulse response of the echo path as follows: **"echo_signal = conv(concatenated_signal, path);",** the result is shown in Fig 5.

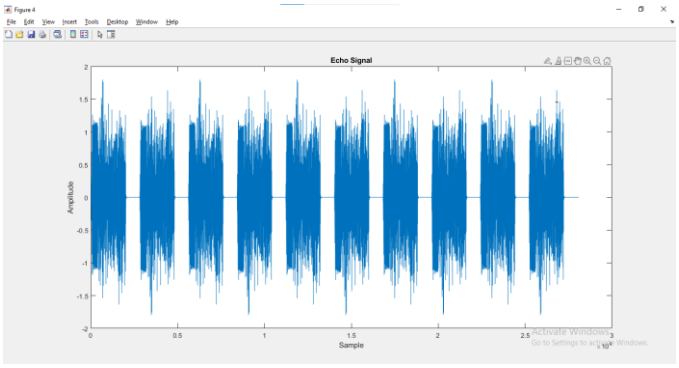

Figure 5 Concatenated Signal

To calculate the input power of the concatenated signal. As well as the "**output_power = sum(echo_signal.^2)/Nc;"** calculates the output power of the echo signal. The line "**ERL = output_power - input_power;"** computes the Echo Return Loss, which represents the difference in power between the input and output signals as shown in Fig.6.

```
>> code
    1.9287e-15

    -6.3309

    -0.7672
```

Figure 6 Displaying Results

D. Adaptive Filtering

The first plot shows the far-end signal, which represents the original signal sent from the source. It appears as a constant signal since it is replicated multiple times (**repmat**) to simulate a continuous input.

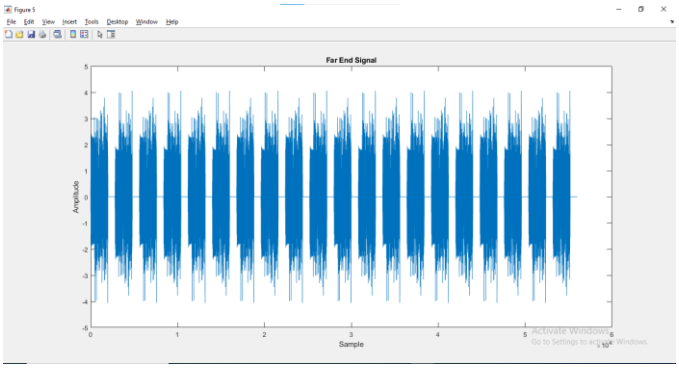

Figure 7 far end signal

The second plot represents the error signal, which is the difference between the estimated echo signal and the actual echo signal. The goal of the echo cancellation system is to minimize this error signal as minimum as possible, or close to zero.

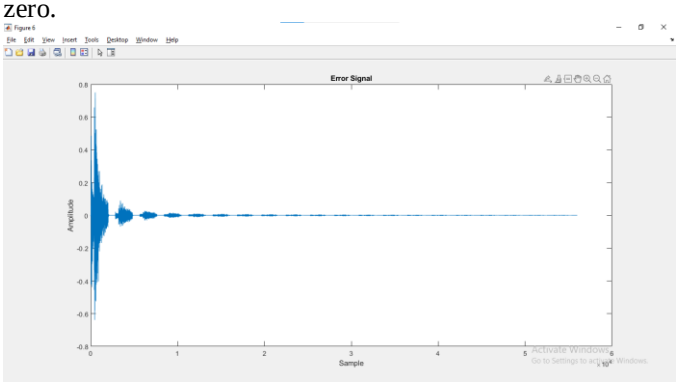

Figure 8 Error signal

The third plot consists of two subplots. The top subplot displays the actual echo signal, which is the result of convolving the far-end signal with the echo path. The bottom subplot shows the estimated echo signal obtained from the e-NLMS filter. The goal is to make the estimated echo signal as close as possible to the actual echo signal.

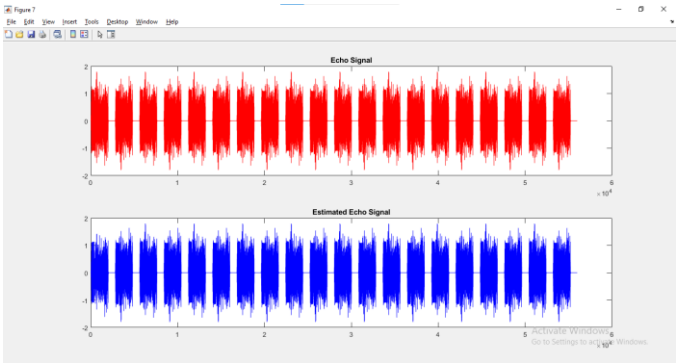

Figure 9 Echo Signal and Estimation

The fourth plot compares the estimated echo path (filter coefficients) with the actual echo path (the path variable). The stem plot visualizes the impulse response of both signals. Ideally, the estimated echo path should match the actual echo path, indicating successful echo cancellation.

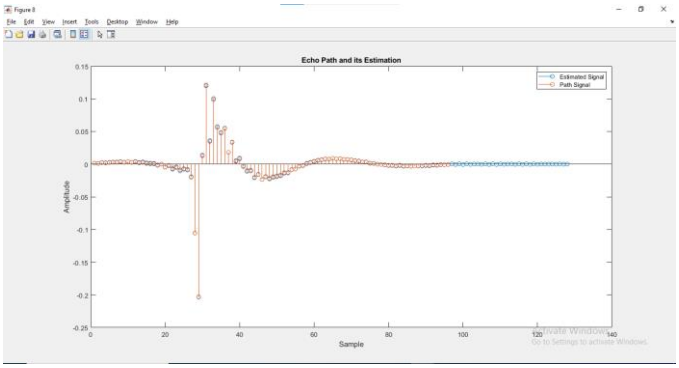

Figure 10 Echo Path and its Estimation

- The e-NLMS filter successfully estimates the echo signal, as shown by the similarity between the estimated echo signal (blue) and the actual echo signal (red) in the third plot.
- The error signal represents the residual echo that remains after cancellation. Ideally, the error signal should be minimized, indicating effective cancellation. If the error signal is large, further optimization or tuning of the filter parameters may be required.
- The comparison between the estimated echo path and the actual echo path in the fourth plot indicates how well the filter adapts to the characteristics of the echo path. A close match between the two signals suggests that the filter has effectively learned the echo path.

The first subplot displays the magnitude response of the original FIR filter. The x-axis represents frequency in Hz, and the y-axis represents the magnitude of the filter response. The "**dB_mag"** variable which converts the magnitude to decibels. By plotting this response, you can observe the frequency characteristics of the echo path.

The second subplot shows the phase response of the original FIR filter. Like the previous subplot, the x-axis represents frequency, and the y-axis represents the phase in degrees or radians. The "**dB_phase**" variable which converts the magnitude to decibels. Examining this plot can provide insights into the phase distortion introduced by the echo path.

The third subplot displays the magnitude response of the estimated FIR filter. The **path** variable is likely the estimated filter coefficients. The x-axis represents frequency, and the y-axis represents the magnitude in decibels. By comparing this plot with the first subplot, you can assess how accurately the estimated FIR filter approximates the original echo path.

The fourth subplot shows the phase response of the estimated FIR filter. It is like the previous subplot, with the x-axis representing frequency and the y-axis representing phase in decibels. Comparing this plot with the second subplot can help you evaluate the accuracy of the estimated FIR filter's phase response.

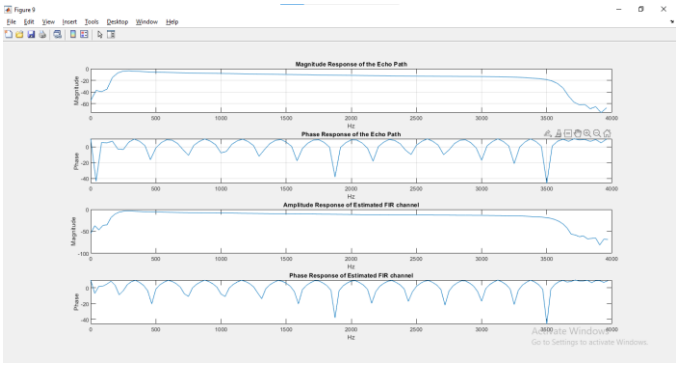

Figure 11 FIR filter and the estimated FIR filter

E. Using Another Adaptive Algorithm (RLS)

The code initializes an RLS adaptive filter (**hd_r)** with a length of 128 taps and a forgetting factor **mu_r** of 0.25. The RLS filter is used to adaptively estimate the echo path and cancel the echo from the far-end signal.

The code creates a figure to plot the error signal (**e_r**). This plot will show how well the RLS filter is removing the echo from the far-end signal. A well-functioning RLS filter will result in a reduced error signal, indicating successful echo cancellation.

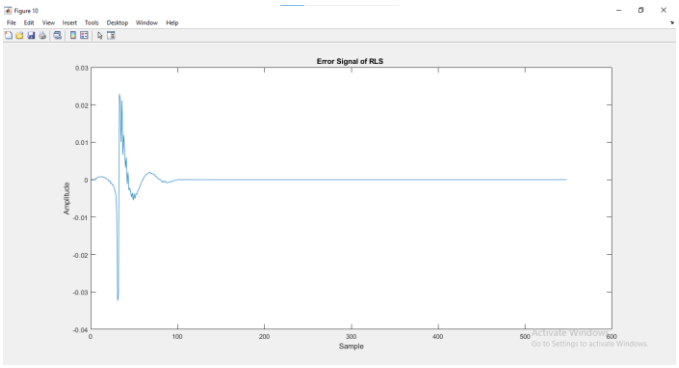

Figure 12 Error Signal of RLS

The code creates a figure with two subplots. The first subplot displays the original echo signal (**echo_signal_r**) in red, while the second subplot shows the part of the estimated echo signal (**y_r**) in blue. By comparing these two signals, you can assess how effectively the RLS filter has removed the echo from the original echo signal and therefore how clear the output signal is.

By comparing the two algorithms RLS and NLMS we noticed that RLS generally exhibits faster convergence compared to NLMS. Moreover, RLS tends to achieve lower steady-state error compared to NLMS.

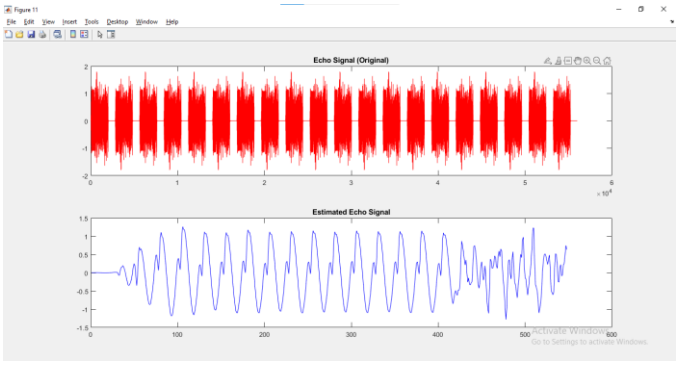

Figure 13 Echo Signal and its Estimated Signal from the RLS

VII. Development

One potential improvement for our project is to enhance the modeling of the echo path. The accuracy of echo cancellation heavily relies on the effectiveness of the echo path model. By incorporating more advanced techniques, such as system identification algorithms or adaptive filtering algorithms with better convergence properties, we can achieve a more accurate and reliable model of the echo path. This can include considering non-linearities, time-varying characteristics, or multi-channel scenarios that may exist in real-world echo environments. Improved modeling will lead to more precise estimation of the echo path and subsequently enhance the performance of the echo cancellation system.

VIII. Conclusion

In this project, we focused on line echo cancellation, which is a crucial aspect of communications over phone lines. We addressed the problem of signal reflections due to mismatches in circuitry, resulting in echoes that interfere with the quality of the received signal. The objective was to implement an adaptive line echo canceller (LEC) to allay the effects of echoes and improve voice quality at both ends of the communication. We analyzed impulse response sequence of a typical echo path, we worked with a synthetic composite source signal (CSS) that emulated speech properties Using the far-end signal and the corresponding output of the echo path as the echo signal, we implemented an adaptive line echo canceller with 128 taps. We trained the canceller using the normalized least mean squares (NLMS) algorithm, Using the far-end signal and the corresponding output of the echo path as the echo signal, we implemented an adaptive line echo canceller with 128 taps. We trained the canceller using the normalized least mean squares (NLMS) algorithm.

We measured the accuracy of the adaptive filter by analyzing the amplitude and phase response of the estimated FIR channel at the end of the iterations and compared it with the given FIR system (Path).

In conclusion, this project provided a comprehensive understanding of line echo cancellation and demonstrated the effectiveness of an adaptive line echo canceller The project opens avenues for further research and development in the field of adaptive filtering and communication systems.

References

[1] https://www.techtarget.com/whatis/definition/digital-signal-processing-DSP

[2] https://www.mathworks.com/discovery/what-is-matlab.html

APPENDIX

```
% Students :
%     Mohammad Mahdi Jarrar - 1201726
%     Ahmad Wasel Ghanem - 1201954
%     Abdullah Sami Naser - 1201952
% Instructor :
%     Dr.Qadri Mayala

%-----------------------------------------------------------------
%                    DSP Project
%               % LINE ECHO CANCELLATION %
%-----------------------------------------------------------------


% PART A: Path load and plot
%-----------------------------------------------------------------
load('path.mat');

    % Sampling frequency
Fs = 8000;

    %plot impulse response
figure;
subplot(2,1,1);
plot(path);
title('Impulse Response of Echo Path');
xlabel('Sample');
ylabel('Amplitude');

    % Use freqz to compute the frequency response

    % Number of points for frequency response computation
Na = numel(path);
[H, F] = freqz(path, 1, Na, Fs);

    % Plot the magnitude response in dB scale
subplot(2,1,2);
dB_mag = 20 * log10(abs(H));
plot(F, dB_mag);
xlabel('Hz');
ylabel('Magnitude');
grid on;
title('Frequency Response of the Echo Path');
%-----------------------------------------------------------------




% PART B: CSS plot and PSD of CSS
%-----------------------------------------------------------------
load('css.mat');

    % Plot the samples of the CSS data
figure;
plot(css);
xlabel('Sample');
ylabel('Amplitude');
title('Samples of the Composite Source Signal (CSS)');

    % Calculate and plot the Power Spectral Density (PSD)
figure;
[Pxx, freq_psd] = pwelch(css);
plot(freq_psd, 10*log10(Pxx));
title('Power Spectrum Density');
xlabel('Frequency (Hz)');
ylabel('Power (dB)');
```

```
% %PART C: Concatenate 5 blocks
% %-----------------------------------------------------------------

    % make the concatenated signal
concatenated_signal = repmat(css, 1, 5);
    % Compute the echo signal
echo_signal = conv(concatenated_signal, path);
    %plot the echo signal
figure;
plot(echo_signal);
title('Echo Signal');
xlabel('Sample');
ylabel('Amplitude');

Nc = length(concatenated_signal);
input_power = sum(concatenated_signal.^2)/Nc;
output_power = sum(echo_signal.^2)/Nc;

disp(10*log10(input_power));
disp(10 * log10(output_power));
ERL = output_power - input_power;
disp(ERL)




% %PART D :
% %-----------------------------------------------------------------

    %generate far_end signal and echo_signal
far_end_signal = repmat(css, 1, 10);
echo_signal= conv(far_end_signal, path);
    %run the e-NLMS filter
    % we use min_length to run the filter properly
min_length = min(length(far_end_signal), length(echo_signal));
far_end_signal = far_end_signal(1:min_length);
echo_signal = echo_signal(1:min_length);
    %filter parameters
mu = 0.25;
hd = dsp.LMSFilter('Length', 128  , 'StepSize', 0.25, 'Method', 'Normalized LMS' );

%filter outputs
    %y ==> Estimated echo signal (filter output)
    %e ==> Error signal
    %w ==> Coeficients of the filter (FIR Estimated channel)
[y, e ,w ] = step(hd, far_end_signal.', echo_signal.');

   %plot signals

  %far end signal
figure
plot(far_end_signal);
title('Far End Signal');
xlabel('Sample');
ylabel('Amplitude');

    %error signal
figure
plot(e);
title('Error Signal');
xlabel('Sample');
ylabel('Amplitude');


    %echo signal and its estimated signal from filter
figure
subplot(2,1,1);
plot(echo_signal ,  'r');
title('Echo Signal');
subplot(2,1,2);
plot(y , 'b');
title('Estimated Echo Signal');
```

```
figure

stem(w);

hold on
stem(path);

legend( 'Estimated Signal', 'Path Signal');
title('Echo Path and its Estimation');
xlabel('Sample');
ylabel('Amplitude');
%-------------------------------------------

%Part E :
%-------------------------------------------
%magnitude response of Original FIR
figure
subplot(4,1,1)
plot(F, dB_mag);
xlabel('Hz');
ylabel('Magnitude');
grid on;
title('Magnitude Response of the Echo Path');

%phase response of origian FIR Path
subplot(4,1,2)
dB_phase = 20 * log10(angle(H));
plot(F, dB_phase);
xlabel('Hz');
ylabel('Phase');
grid on;
title('Phase Response of the Echo Path');

%magnitude response of estimated FIR
subplot(4,1,3)
Ne = numel(w);
[He, Fe] = freqz(path, 1, Ne, Fs);
dB_mage = 20 * log10(abs(He));
plot(Fe, dB_mage);
xlabel('Hz');
ylabel('Magnitude');
title('Amplitude Response of Estimated FIR channel');

%phase response of estimated FIR
subplot(4,1,4)
dB_phaseE = 20 * log10(angle(He));
plot(Fe, dB_phaseE);
xlabel('Hz');
ylabel('Phase');
title('Phase Response of Estimated FIR channel');

%-------------------------------------------------------------------
%PART F: RLS Adaptive Algorithm
%------------------------


% Generate far-end signal and echo_signal
far_end_signal_r = repmat(css, 1, 10);
echo_signal_r = conv(far_end_signal_r, path);

% Run the RLS filter
min_length_r = min(length(far_end_signal_r), length(echo_signal_r));
far_end_signal_r = far_end_signal_r(1:min_length_r);
echo_signal_r = echo_signal_r(1:min_length_r);
mu_r = 0.25;
hd_r = dsp.RLSFilter('Length', 128, 'ForgettingFactor',mu_r);

% Filter outputs
[y_r, e_r] = step(hd_r, far_end_signal_r.', echo_signal_r.');

% Plot signals
```

```
% Error signal
figure
plot(e_r);
title('Error Signal of RLS ');
xlabel('Sample');
ylabel('Amplitude');

% Echo signal and its estimated signal from the filter RLS
figure
subplot(2, 1, 1);
plot(echo_signal_r, 'r');
title('Echo Signal (Original)');
subplot(2, 1, 2);
plot(y_r, 'b');
title('Estimated Echo Signal');
```