Digital Signal Processing - Line echo cancellation and System Identification

Jana Abu Nasser, *1201110*, Raneem Daqa, *1202093*, Dunia Jaser, *1201345*

Abstract— **This project aims to solve the problem of echoes during phone calls. Sometimes, the sound bounces back and creates unwanted noise, making it difficult to hear clearly. To fix this, we use a device called an adaptive line echo canceller. It listens to the reflected sound and makes a duplicate of the echo. Then, it subtracts this echo from the original sound, making it more understandable. The improved sound is then sent back to the other person, making the conversation better and echo-free.**

I. INTRODUCTION

Telecommunications can be plagued by echoes that severely impact the quality of phone conversations. These echoes arise when the signal, originating from the far-end and destined for the near-end, bounces back at the near-end due to circuit discrepancies like hybrid connections. Consequently, the speaker at the far-end not only hears the intended signal from the near-end but also an attenuated copy of their own voice, known as an echo. This interference is depicted in Figure 1, where the reflected signal returns to the far-end, creating disruptions in the communication.

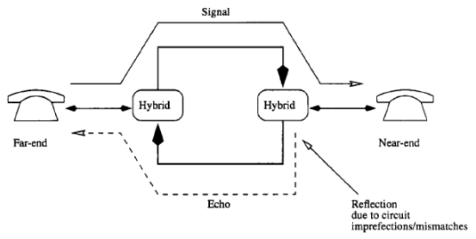


Fig. 1. *The signal at the far-end is reflected at the near-end due to circuit mismatches and travels back to the far-end.*

To mitigate the interference caused by echoes and improve voice quality, adaptive line echo cancellers (LEC) are commonly employed. These devices analyze the reflected signal and generate a replica of the echo. By subtracting this echo replica from the received signal at the near-end, a cleaner and clearer "error signal" is obtained, as shown in Figure 2.

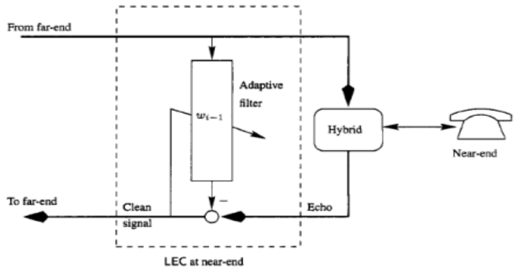


Fig. 2. *An adaptive line echo canceller at the near-end.*

System identification plays a crucial role in echo cancellation. It involves finding the input-output response relationship of the unknown system, which represents the echo-producing circuitry. To achieve this, the project adopts a configuration as shown in Figure 3.

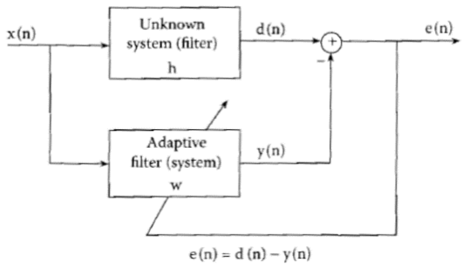


Fig. 3. *System Identification.*

Assuming the unknown system is time-invariant, with finite and constant impulse response coefficients, the desired response is modeled as a linear combination of the system's coefficients and the input signal.

The project utilizes an adaptive Finite Impulse Response (FIR) filter, employing the same number of coefficients as the unknown echo-producing system, to estimate and eliminate echoes. Through adaptive filtering techniques, such as the Normalized Least Mean Squares (NLMS) algorithm, the filter's output is continuously adjusted and compared to the original signal. By updating the filter coefficients based on the step-size factor and input data, the aim is to minimize the error between the canceled echo and the original signal, ultimately improving the overall voice quality in telecommunications.

Inputs:

- M: filter length
- µ:step-size factor
- x(n): input data to the adaptive filter of length N (Vector)
- w(0):initialization filter (vector) =zeros of length M

Outputs at each iteration (n):

- y(n)=w^T (n)x(n)
- e(n)=d(n)-y(n)
- w(n+1)=w(n)+ $\frac{1}{\epsilon+\|x\|^2}$ e(n)x(n), the updated filter

coefficients , || x || is the norm.

II. Problem Specification

In phone line communications, the existence of echoes poses a significant challenge. These echoes are caused by mismatches in circuitry, resulting in a reflected signal from the near-end that interferes with the original signal at the far-end. As a consequence, the received signal suffers from distortion and reduced voice quality, impeding effective communication. The goal of this project is to develop an adaptive line echo cancellation system capable of precisely estimating and eliminating these echoes. By doing so, we aim to enhance voice clarity and optimize the overall communication experience for users.

III. Data Evaluation Criteria

The Line Echo Cancellation project requires a dataset consisting of input signals representing far-end signals received at the near-end, synchronized echo signals representing reflections from the near-end, and corresponding desired signals obtained by subtracting the echo signals from the input signals. All signals should be sampled at 8 kHz. The project's evaluation criteria focus on assessing the performance of the adaptive line echo canceller (LEC) in minimizing echo interference. Evaluation metrics include mean squared error (MSE) between desired and output signals, signal-to-echo ratio (SER), and residual echo level (REL). Additionally, the convergence rate of the adaptive algorithm and the computational complexity of the echo cancellation system are important factors to consider during evaluation.

IV. Approach

The approach for the Line Echo Cancellation project involves using an adaptive algorithm called Normalized Least Mean Squares (NLMS) to identify and model the system responsible for the echo. The system identification is achieved by comparing the desired response (the clean signal) with the output of an adaptive FIR filter. By adjusting the filter coefficients iteratively, the algorithm aims to minimize the error between the desired response and the filter output. The filter coefficients are updated using a step-size factor and the input data, and the process continues until the error approaches zero. This adaptive filtering approach allows for the cancellation of the echo and improves the voice quality in phone line communications.

V. Result and Analysis

In the first part, branch A, we drew the impulse and frequency responses of the echo path

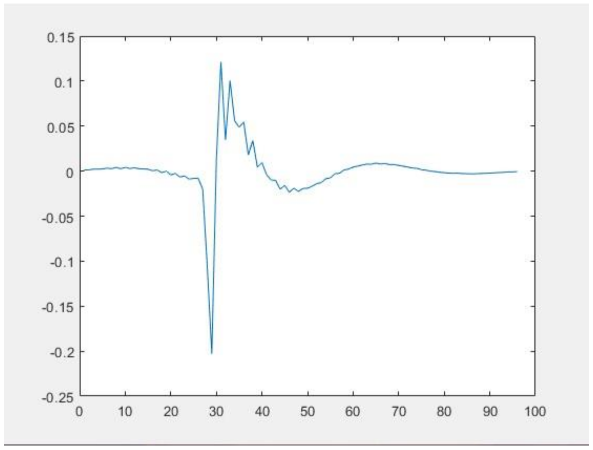


***Fig. 4**. Plot the impulse of the echo path*

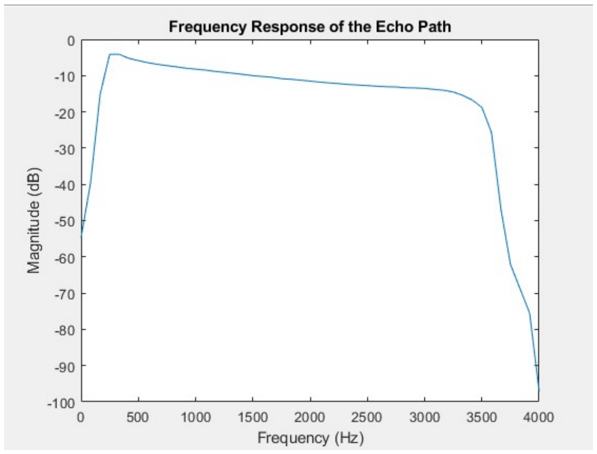


Fig. 5. *Plot the frequency responses of the echo path*

In branch B, we drew the samples of the CSS data, as well as their spectrum (Power Spectrum Density PSD).

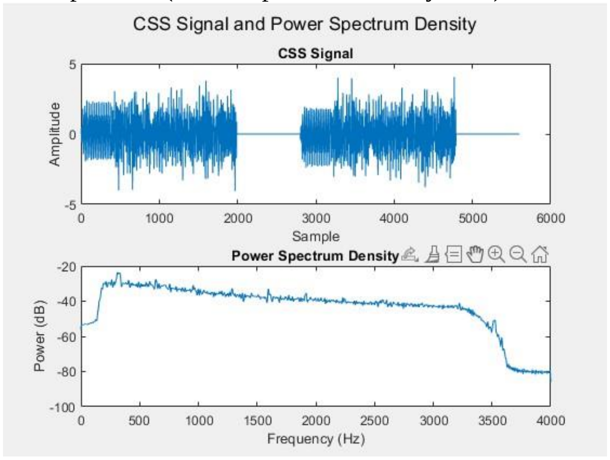


Fig. 6. *Plot the samples of the CSS data, and Power Spectrum Density (PSD)*

In branch c, we concatenated five such blocks and feed them into the echo path and estimated the input and output powers in dB using:

$$\hat{P} = 10\log_{10}\left(\frac{1}{N}\sum_{i=1}^{N}|signal\,(i\,)|^2\right)$$

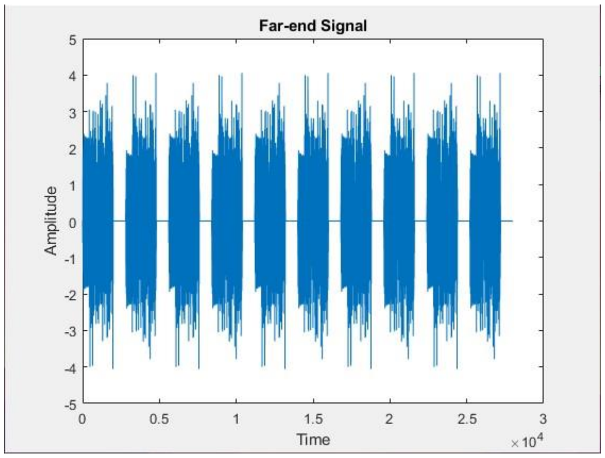


Fig .7. *Plot the resulting Far-end signal*

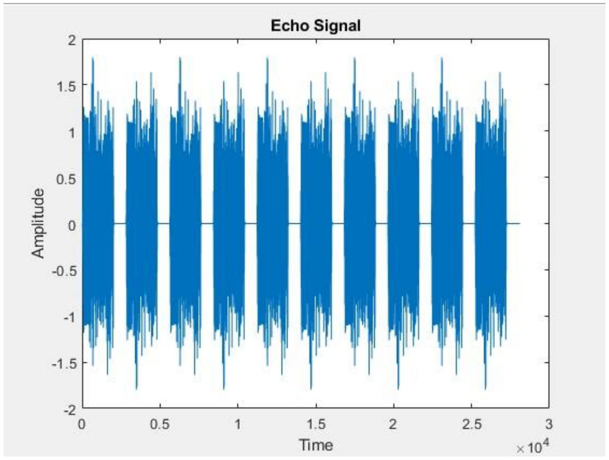


Fig.8. *Plot the resulting echo signal*

In branch d, we used 10 blocks of CSS data as a far-end signal and the corresponding output of the echo path as the echo signal.

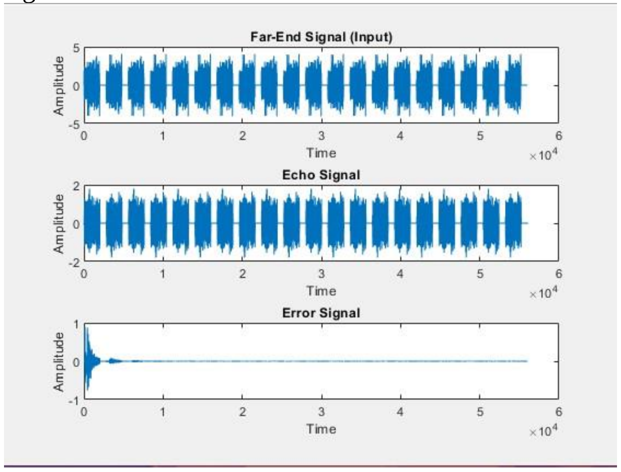
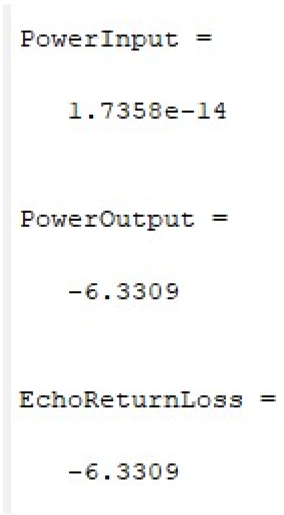


Fig.9. *Plot the far-end, the echo, and the error signal*



```
PowerInput =

   1.7358e-14


PowerOutput =

   -6.3309


EchoReturnLoss =

   -6.3309
```

Fig .10. *Power Input, Power Output and Echo Return Loss values*

In branch e, we drew amplitude and phase response for the estimated FIR channel at the end of the iterations and Compare it with the given FIR system (Path).

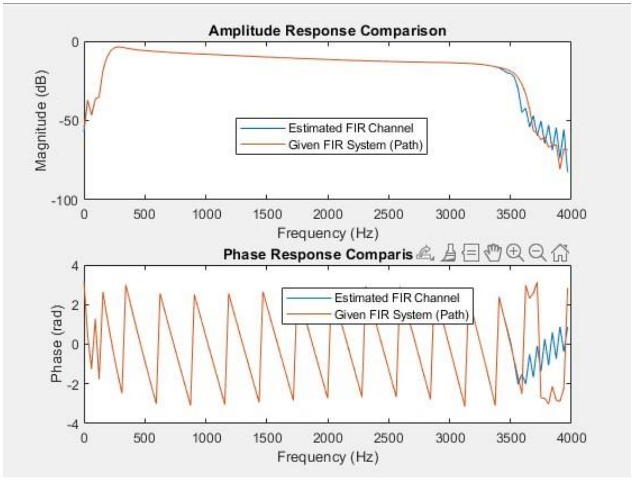


Fig.11. *Plot the amplitude and phase response for the estimated FIR channel*

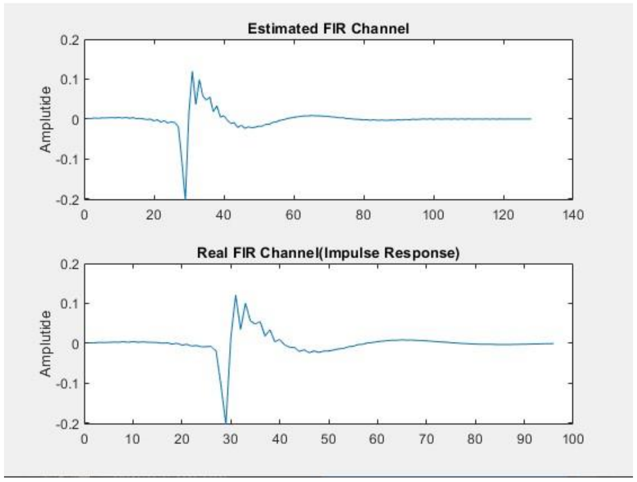


Fig.12.*Compare the amplitude and phase response with the given FIR system (Path).*

In branch F, Propose a different appropriate Adaptive algorithm and compare it to the ∈-NLMS

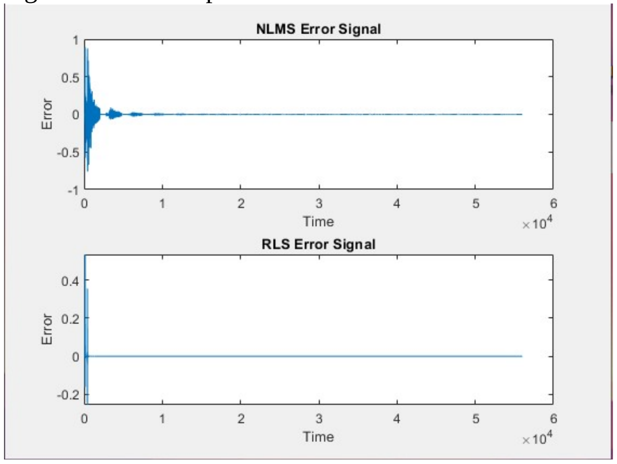


Fig.13.*Plot NLMS & RLS error signal*

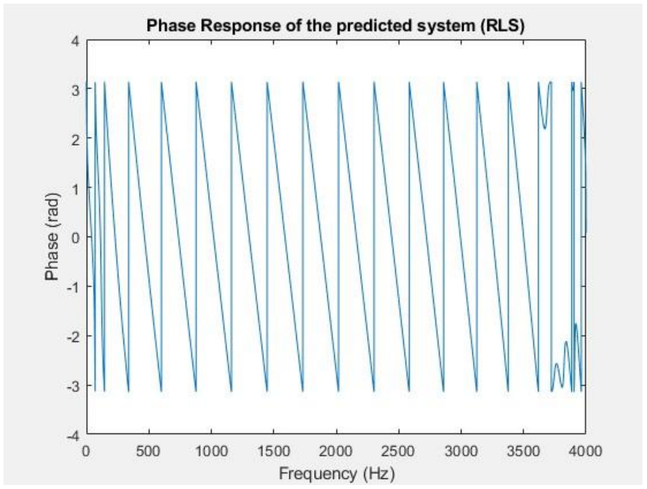


Fig.14.*Plot of the Phase Response of the predicted system RLS*

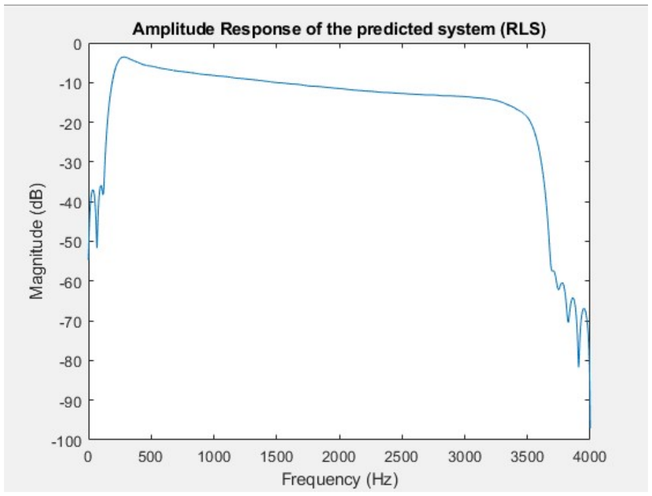


Fig.15.*Plot of the Amplitude Response or the RLS*

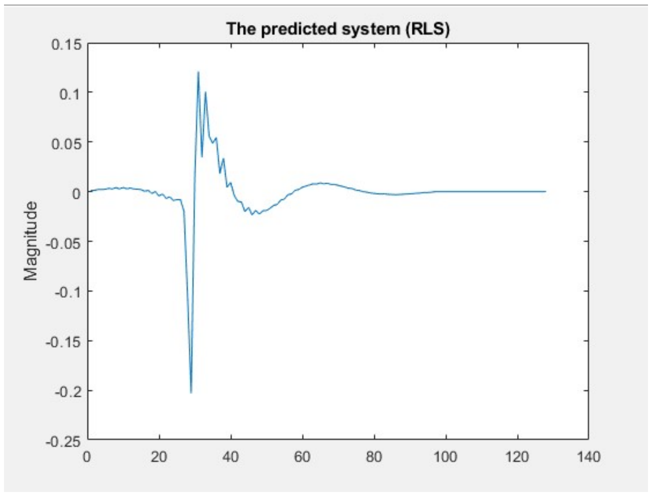


Fig.16.*Plot The predicted System RLS*

VI. Development

By incorporating additional algorithms into the project and ensuring adequate training and testing, we can enhance the operational speed and reduce the error rate of the line echo cancellation system.

VII. Conclusion

In conclusion, the Line Echo Cancellation project effectively tackles the problem of echo interference in phone line communication. By using the adaptive NLMS algorithm and continuously adjusting the filter parameters based on input and desired outcomes, the project successfully eliminates the echo, leading to improved voice quality during calls. The integration of additional algorithms and thorough testing further enhances the system's efficiency and reduces errors. Overall, this project offers a practical and reliable solution for resolving echo issues, resulting in clearer and more enjoyable phone conversations.

VIII. Appendix

Part A:

```
impulseResponse1 = load('path')
impulseResponse = impulseResponse1.path
figure
plot(impulseResponse)
% Compute the Fourier Transform of the impulse response
frequencyResponse = abs(fft(impulseResponse));
fs = 8000; % the sampling frequency is 8000 Hz

% Computing the frequency response
N = length(impulseResponse); % Length of the impulse
response
frequencies = (0:fs/N:fs/2); % Frequency vector

% Computing the frequency response using freqz
[h, f] = freqz(impulseResponse, 1, frequencies, fs);

% Plotting the frequency response
figure;
```

```
plot(f, 20*log10(abs(h)));
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Frequency Response of the Echo Path');
```

Part B:

```
% Load the css.mat file
load('css.mat');

% Extract the CSS signal from the loaded file
cssSignal = css;

% Plot the samples of the CSS data
figure;
subplot(2,1,1);
plot(cssSignal);
title('CSS Signal');
xlabel('Sample');
ylabel('Amplitude');

% Calculate the Power Spectrum Density (PSD)
fs = 8000; % Sample rate (assumed)
psd = pwelch(cssSignal, [], [], [], fs);

% Plot the PSD
subplot(2,1,2);
plot(linspace(0, fs/2, length(psd)), 10*log10(psd));
title('Power Spectrum Density');
xlabel('Frequency (Hz)');
ylabel('Power (dB)');

% Adjust the figure layout
sgtitle('CSS Signal and Power Spectrum Density');
```

Part C:

```
load('css');
% Generate input signal (far-end signal)
numBlocks = 5; % Number of blocks
blockSize = length(css); % Size of each block
inputSignal = zeros(1, numBlocks*blockSize); % Initialize
input signal

for i = 1:numBlocks
    startIndex = (i-1)*blockSize + 1;
    endIndex = i*blockSize;
    inputSignal(startIndex:endIndex) = css; % Generate
random block
end

% Plot the resulting echo signal
figure;
plot(inputSignal);
xlabel('Time');
ylabel('Amplitude');
title('Far-end Signal');

% Compute the resulting echo signal
echoSignal = conv(inputSignal, impulseResponse);

% Plot the resulting echo signal
figure;
plot(echoSignal);
xlabel('Time');
ylabel('Amplitude');
title('Echo Signal');


N = length(inputSignal);
squareSummationInput = 0;
squareSummationOutput = 0;
for i= 1: N
    squareSummationInput = squareSummationInput +
(abs(inputSignal(i)).^2);
    squareSummationOutput = squareSummationOutput +
(abs(echoSignal(i)).^2);

end
PowerInput = 10*log10((1/N)*squareSummationInput)
PowerOutput = 10*log10((1/N)*squareSummationOutput)
EchoReturnLoss = PowerOutput – PowerInput
```

Part D:

```
% Generate input signal (far-end signal)
numBlocks = 10; % Number of blocks
blockSize = length(css); % Size of each block
inputSignalD = zeros(1, numBlocks*blockSize); %
Initialize input signal

for i = 1:numBlocks
    startIndex = (i-1)*blockSize + 1;
    endIndex = i*blockSize;
    inputSignalD(startIndex:endIndex) = css;
end
% Compute the resulting echo signal
echoSignalD = conv(inputSignalD, impulseResponse);
%parameters
M =128;
mu = 0.25;
epsilon = 1e-6;
N = length(inputSignalD);
x = inputSignalD;
d = echoSignalD;

w_n = zeros(1, M); % Initialize filter coefficients
errorSignal = zeros(1,N); % Initialize error signal

% NLMS algorithm
for i = M:N
    inputBlock = x(i:-1:i-M+1);
    y = sum(w_n .* inputBlock);
    errorSignal(i) = d(i) - y;
    % Compute the normalization term
    normalizationTerm = epsilon + norm(inputBlock)^2;
    % Update the filter coefficients
```

```
        w_n = w_n + (mu / normalizationTerm) * errorSignal(i)
* inputBlock;
    end

    % Plot the results
    figure;
    subplot(3, 1, 1);
    plot(x);
    xlabel('Time');
    ylabel('Amplitude');
    title('Far-End Signal (Input)');

    subplot(3, 1, 2);
    plot(d);
    xlabel('Time');
    ylabel('Amplitude');
    title('Echo Signal');

    subplot(3, 1, 3);
    plot(errorSignal);
    xlabel('Time');
    ylabel('Amplitude');
    title('Error Signal');
```

Part E:

```
    % Compute the estimated FIR channel response
    estimatedFIRChannel = w_n;
    figure;
    subplot(2, 1, 1);
    plot(estimatedFIRChannel);
    ylabel('Amplutide');
    title('Estimated FIR Channel');

    subplot(2, 1, 2);
    plot(impulseResponse);
    ylabel('Amplutide');
    title('Real FIR Channel(Impulse Response)');

    % Compute the frequency response of the estimated FIR
channel
    fs = 8000; % Sampling frequency
    N = length(estimatedFIRChannel);
    [estimatedFrequencyResponse, frequencies] =
freqz(estimatedFIRChannel, 1, N, fs);
    estimatedAmplitudeResponse =
20*log10(abs(estimatedFrequencyResponse));
    estimatedPhaseResponse =
angle(estimatedFrequencyResponse);

    % Compute the frequency response of the given FIR system
(Path)
    [givenFrequencyResponse, ~] = freqz(impulseResponse, 1,
N, fs);
    givenAmplitudeResponse =
20*log10(abs(givenFrequencyResponse));
    givenPhaseResponse = angle(givenFrequencyResponse);

    % Plot the amplitude response
    figure;
    subplot(2, 1, 1);
    plot(frequencies, estimatedAmplitudeResponse);
    hold on;
    plot(frequencies, givenAmplitudeResponse);
    hold off;
    xlabel('Frequency (Hz)');
    ylabel('Magnitude (dB)');
    legend('Estimated FIR Channel', 'Given FIR System
(Path)');
    title('Amplitude Response Comparison');

    % Plot the phase response
    subplot(2, 1, 2);
    plot(frequencies, estimatedPhaseResponse);
    hold on;
    plot(frequencies, givenPhaseResponse);
    hold off;
    xlabel('Frequency (Hz)');
    ylabel('Phase (rad)');
    legend('Estimated FIR Channel', 'Given FIR System
(Path)');
    title('Phase Response Comparison');
```

Part F:

```
%RLS algorithm
%parameters
M =128;
mu1 = 0.99;
N = length(inputSignalD);
x = inputSignalD;
d = echoSignalD;
delta = 1;
P_n = delta * eye(M);
w_n_rls = zeros(1, M);
errorSignal_rls = zeros(1,N);

% RLS algorithm
for i = M:N
    inputBlock = x(i:-1:i-M+1);
    y_rls = w_n_rls * inputBlock';
    errorSignal_rls(i) = d(i) - y_rls;
    % Compute the Kalman gain
    K = P_n * inputBlock' / (mu1 + inputBlock * P_n *
inputBlock');
    % Update the filter coefficients
    w_n_rls = w_n_rls + K' * errorSignal_rls(i);
    % Update the inverse correlation matrix
    P_n = (P_n - K * inputBlock * P_n) / mu1;
end
% Plot the error signals
figure;
subplot(2, 1, 1);
plot(errorSignal);
xlabel('Time');
ylabel('Error');
title('NLMS Error Signal');
```

```
subplot(2, 1, 2);
plot(errorSignal_rls);
xlabel('Time');
ylabel('Error');
title('RLS Error Signal');

[w_n_rlsResponse, frequencies] = freqz(w_n_rls, 1, N, fs);
estimatedAmplitudeResponse =
20*log10(abs(w_n_rlsResponse));
estimatedPhaseResponse = angle(w_n_rlsResponse);
% Plot the predicted system (RLS)
figure;
plot(w_n_rls);
ylabel('Magnitude');
title('The predicted system (RLS)');
% Plot the amplitude response
figure;
plot(frequencies, estimatedAmplitudeResponse);
xlabel('Frequency (Hz)');
ylabel('Magnitude (dB)');
title('Amplitude Response of the predicted system (RLS)');

% Plot the phase response
figure;
plot(frequencies, estimatedPhaseResponse);
xlabel('Frequency (Hz)');
ylabel('Phase (rad)');
title('Phase Response of the predicted system (RLS)');
```

IX. References

[1] *https://www.elprocus.com/fir-filter-for-digital-signal-processing/*

[2] *https://www.keil.com/pack/doc/CMSIS/DSP/html/group__LMS__NORM.html*