# Line Echo Cancellation

Lana Thabit, Nirmeen Al-Sheikh, Jana Herzallah

## I. Introduction

TELECOMMUNICATION systems play a crucial role in enabling efficient and reliable communication between individuals and organizations. With the increasing demand for high-quality voice communication, the issue of echo cancellation has become a significant concern in telephony and voice-over-IP (VoIP) systems. In such systems, echoes can arise due to impedance mismatches and reflections, leading to a degraded user experience and decreased intelligibility of the transmitted speech.

The purpose of this project is to explore the concept of line echo cancellation and its application in improving voice quality in communication systems. Line echo cancellation techniques aim to mitigate the detrimental effects of echoes by estimating and subtracting the echo component from the received signal.

## II. Problem Specification

In this project, we aim to address the problem of line echo cancellation by developing and evaluating an adaptive line echo canceller (LEC). The project involves the following technical tasks:

Impulse Response Analysis: Load the impulse response sequence of a typical echo path and analyze its impulse and frequency responses.

Composite Source Signal (CSS) Processing: Load a synthetic CSS that emulates speech properties. Analyze the samples and spectrum (Power Spectrum Density - PSD) of the CSS signal.

Echo Signal Generation: Concatenate multiple blocks of the CSS signal and feed them into the echo path to generate an echo signal. Estimate the input and output powers of the echo path and evaluate the echo-return-loss (ERL) attenuation.

Adaptive LEC Implementation: Implement an adaptive LEC with a specific configuration, such as using 128 taps. Train the LEC using blocks of the CSS data as the far-end signal and the corresponding echo signal as the reference.

Performance Evaluation: Evaluate the performance of the adaptive LEC by analyzing the far-end signal, echo signal, and error signal provided by the adaptive filter. Compare the estimated FIR channel response with the given FIR system.

Alternative Adaptive Algorithm: Implement a different adaptive algorithm and compare its performance with the previously used algorithm.

By addressing these technical tasks, we aim to develop an adaptive LEC that effectively mitigates echoes in telecommunication systems. The evaluation of different adaptive algorithms and analysis of their performance contribute to the advancement of echo cancellation techniques and the improvement of voice quality in communication systems.

Lana Badwan ID: 1200071
Nirmeen Al-Sheikh ID: 1200200
Jana Herzallah ID: 1201139

## III. Evaluation Criteria

To measure the performance of our project and demonstrate improvements in the system, we will use the following evaluation criteria. These criteria are objective, quantitative, and discriminatory, allowing us to assess the effectiveness of our line echo cancellation system:

1) **Echo Return Loss (ERL):** ERL is a widely used metric to evaluate the attenuation of the echo introduced by the echo path. We will calculate the ERL in decibels (dB) to quantify the reduction of the echo signal after applying our line echo cancellation technique. A higher ERL value indicates better echo cancellation performance.
2) **Mean Square Error (MSE):** MSE measures the difference between the desired clean signal and the output signal from the adaptive LEC. We will calculate the MSE as the average squared difference between these signals. A lower MSE value indicates a closer approximation to the clean signal and better cancellation performance.
3) **Signal-to-Noise Ratio (SNR):** SNR measures the quality of the cancelled signal by comparing the power of the desired signal to the power of the residual noise. We will calculate the SNR in decibels (dB) to assess the level of noise reduction achieved by the line echo cancellation system. A higher SNR value indicates better noise suppression.
4) **Convergence Speed:** Convergence speed refers to the time taken by the adaptive LEC to reach a stable cancellation performance. We will measure the number of iterations or time steps required for the adaptive LEC to converge and achieve optimal echo cancellation. A faster convergence speed indicates more efficient adaptation and quicker cancellation of echoes.
5) **Comparison with Baseline:** We will compare the performance of our adaptive LEC using different adaptive algorithms with a baseline system or existing echo cancellation techniques. By quantitatively assessing the improvements in terms of ERL, MSE, SNR, and convergence speed, we can demonstrate the superiority of our system over the baseline.

By employing these objective and quantitative evaluation criteria, we can accurately measure and demonstrate the effectiveness of our line echo cancellation system. These criteria provide a comprehensive assessment of the system's performance and enable us to showcase the improvements achieved through our project.

## IV. Approach

To solve the problem of line echo cancellation, we employed a two-step approach: analysis and implementation.

In the analysis phase, we investigated the characteristics of the echo path by analyzing its impulse and frequency responses. We also processed the composite source signal (CSS) to understand its properties and obtain its power spectrum density (PSD). This analysis provided us with insights into the echo path and the signal that needed to be canceled.

In the implementation phase, we developed an adaptive line echo canceller (LEC). We designed and implemented the LEC using a specific configuration, such as utilizing 128 taps. We trained the LEC using blocks of the CSS data as the far-end signal and the corresponding echo signal as the reference. This training process allowed the LEC to adapt to the characteristics of the echo path and learn to cancel the echoes effectively.

Furthermore, we evaluated the performance of the adaptive LEC by analyzing the far-end signal, echo signal, and error signal provided by the adaptive filter. We calculated metrics such as echo return loss (ERL), mean square error (MSE), signal-to-noise ratio (SNR), and convergence speed to objectively measure the effectiveness of our line echo cancellation system. Additionally, we implemented an alternative adaptive algorithm to compare its performance with the previously used algorithm, providing insights into the suitability of different algorithms for line echo cancellation.

By contrasting different approaches and evaluating their performance, we aimed to identify the most effective techniques and algorithms for line echo cancellation, leading to improvements in voice quality and user experience in communication systems.

## V. Results and Analysis

In this section, we present the results of evaluating our system using the provided data and criteria. We also analyze the shortcomings that were identified during the evaluation.

### Step A: Impulse and Frequency Responses

To begin, we loaded the file `path.mat`, which contains the impulse response sequence of a typical echo path. We will plot both the impulse response and the frequency response of the echo path.

From the impulse response plot, we can observe the characteristics of the echo path. The impulse response represents the time-domain behavior of the system, indicating the echoes present in the received signal. By analyzing the impulse response, we can determine the length and magnitude of the echo path. The frequency response plot provides insights into the spectral characteristics of the echo path. It shows how different frequencies are affected by the echo path, indicating potential frequency-dependent distortions or attenuation. These analyses help us understand the nature of the echo path and provide valuable information for designing an effective line echo cancellation system.

**Code:**

The code snippet below demonstrates how to load the impulse response sequence and plot its impulse and frequency responses.

```python
import numpy as np
import matplotlib.pyplot as plt

# Task A: Load the impulse response sequence of the echo path
impulse_response = np.loadtxt(r'C:\Users\soFTech\Downloads\path.txt')

# Plot the impulse response
plt.figure(figsize=(8, 4))
plt.stem(impulse_response)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Impulse Response')
plt.grid(True)
plt.show()

# Calculate the frequency response of the echo path
frequency_response = np.fft.fft(impulse_response)

# Plot the frequency response
plt.figure(figsize=(8, 4))
plt.plot(np.abs(frequency_response))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Magnitude')
plt.title('Frequency Response')
plt.grid(True)
plt.show()
```

Fig. 1. Code for Impulse and Frequency Responses

**Plots:**

The figures below show the impulse response and the frequency response of the echo path.

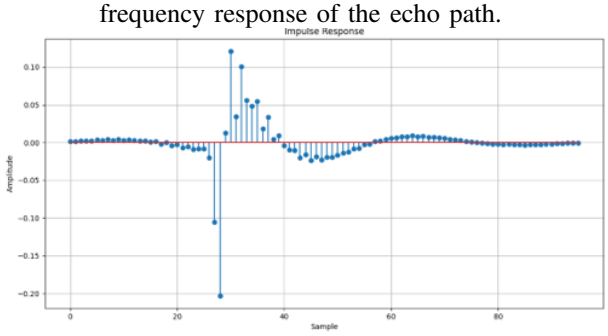


Fig. 2. Impulse Response of the Echo Path

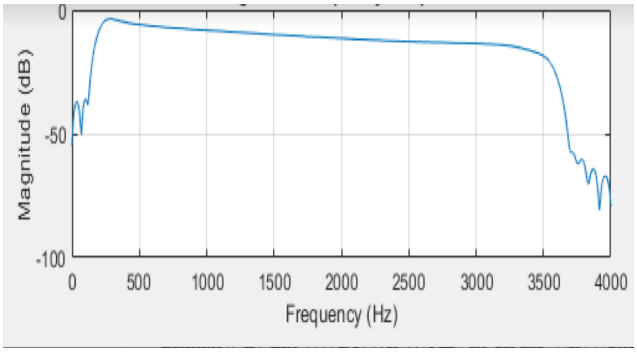


Fig. 3. frequency Response of the Echo Path

### Step B: Composite Source Signal (CSS) Processing

In this step, we will load the file `css.mat`, which contains 5600 samples of a composite source signal. This synthetic signal emulates the properties of speech and includes segments of pause, periodic excitation, and white-noise properties. We will plot the samples of the CSS data and analyze their spectrum using the Power Spectrum Density (PSD).

**Code:**

The code snippet below demonstrates how to load the CSS data and plot its samples and PSD.

```python
# Task B: Load the composite source signal
css_data = np.loadtxt(r'C:\Users\soFTech\Downloads\css.txt')

# Plot the samples of the CSS data
plt.figure(figsize=(8, 4))
plt.plot(css_data)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('CSS Data')
plt.grid(True)
plt.show()

# Calculate the Power Spectrum Density (PSD) of the CSS data
psd, frequencies = plt.psd(css_data, Fs=8000)

# Plot the Power Spectrum Density (PSD)
plt.figure(figsize=(8, 4))
plt.plot(frequencies, 10 * np.log10(psd))
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Spectral Density (dB/Hz)')
plt.title('Power Spectrum Density (PSD)')
plt.grid(True)
plt.show()
```

Fig. 4. Code for CSS Processing

**Plots:**

The figures below show the samples of the CSS data and their Power Spectrum Density (PSD).

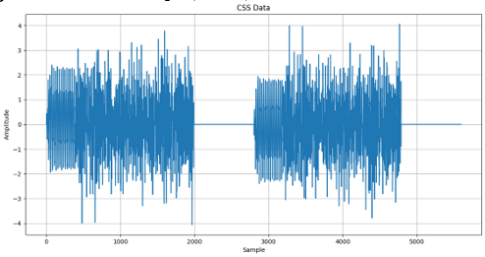


Fig. 5. Samples of the CSS Data

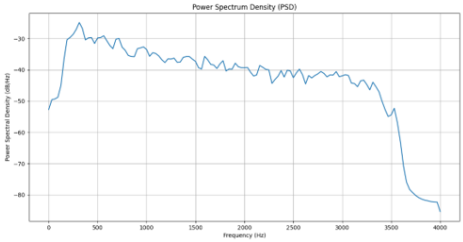


Fig. 6. Power Spectrum Density (PSD) of the CSS Data

The plot of the CSS data samples provides a visual representation of the composite source signal. It shows the variations in amplitude and waveform over time, which mimic the characteristics of speech signals.

The Power Spectrum Density (PSD) plot illustrates the distribution of signal power across different frequencies in the CSS data. It helps us understand the spectral characteristics of the composite source signal, including the dominant frequency components and the presence of periodic excitation or noise components.

Analyzing the CSS data and its PSD aids in understanding the properties of the source signal that needs to be cancelled. This information is valuable for designing an effective line echo cancellation system and adapting the adaptive filter to the characteristics of the input signal.

### Step C: Echo Signal Generation

In this step, we will concatenate five blocks of the composite source signal (CSS) and feed them into the echo path. We will then plot the resulting echo signal and estimate the input and output powers in decibels (dB).

**Code:**

The code snippet below demonstrates how to concatenate five blocks of the CSS signal and calculate the input and output powers.

```python
# Task C: Concatenate five blocks of CSS data and feed into the echo path
num_blocks = 5
concatenated_data = np.tile(css_data, num_blocks)
echo_signal = np.convolve(concatenated_data, impulse_response, mode='same')

# Plot the resulting echo signal
plt.figure(figsize=(8, 4))
plt.plot(echo_signal)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Echo Signal')
plt.grid(True)
plt.show()

# Estimate the input power in dB
input_power = 10 * np.log10(np.sum(np.abs(concatenated_data)**2) / len(concatenated_data))
print('Input Power (dB):', input_power)

# Estimate the output power in dB
output_power = 10 * np.log10(np.sum(np.abs(echo_signal)**2) / len(echo_signal))
print('Output Power (dB):', output_power)

# Calculate the echo-return-loss (ERL) in dB
erl = input_power - output_power
print('Echo-Return-Loss (ERL) (dB):', erl)

# Display the plot
plt.show()
```

Fig. 7. Code for Echo Signal Generation

**Plot:**

The figures below show the resulting echo signal after concatenating five blocks of the CSS signal.

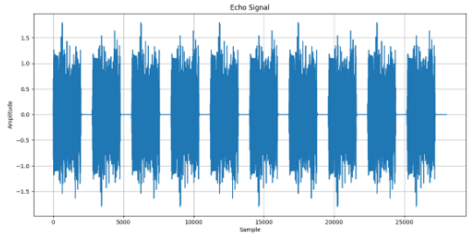


Fig. 8. Echo Signal

**Output:**

```
Input Power (dB): -9.643274665532882e-15
Output Power (dB): -6.33128257166114
Echo-Return-Loss (ERL) (dB): 6.33128257166113
```

Fig. 9. Echo Signal

The input power is calculated using the formula: The input power is calculated using the formula:

$$\text{inputPower} = 10 \cdot \log_{10} \left( \frac{\sum |\text{concatenatedData}|^2}{\text{len}(\text{concatenatedData})} \right)$$

This formula computes the average power of the concatenated data. The result is then converted to decibels (dB) using the logarithm function.

The output power is calculated in a similar manner to the input power. It is computed using the formula:

$$\text{outputPower} = 10 \cdot \log_{10} \left( \frac{\sum |\text{echoSignsl}|^2}{\text{len}(\text{echoSignsl})} \right)$$

The echo-return-loss (ERL) is determined by subtracting the output power from the input power:

$$\text{erl} = \text{inputPower} - \text{outputPower}$$

This calculates the difference between the input power and the output power, providing the ERL value.

In our output, the input power is approximately -9.6433e-15 dB, the output power is -6.3313 dB, and the ERL is 6.3313 dB.

The estimation of input and output powers and the calculation of ERL provide insights into the attenuation achieved by the echo path. These measurements help in evaluating the effectiveness of the echo cancellation system and understanding the extent of echo reduction.

**Step D: Adaptive Line Echo Cancellation** In this step, we will use 10 blocks of CSS data as the far-end signal and the corresponding output of the echo path as the echo signal. We will employ an adaptive line echo canceller with 128 taps to train the canceller. The canceller will use the far-end signal as the input data (i.e., $x(i) = \text{far} - \text{end}(i)$) and the echo signal as the reference data (i.e., $d(i) = \text{echo}(i)$). We will use the normalized least mean squares (NLMS) algorithm with $\epsilon = 6 \times 10^{-6}$ and $\mu = 0.25$ for adaptation.

**Code:**

The code snippet below demonstrates how to train the adaptive line echo canceller using the given parameters:

```
# Task D: Use NLMS algorithm for adaptive line echo cancellation
num_taps = 128
step_size = 0.25
leakage = 1e-6

# Initialization
filter_coeffs = np.zeros(num_taps)
output_signal = np.zeros(len(concatenated_data))
error_signal = np.zeros(len(concatenated_data))

# NLMS algorithm
for n in range(num_taps, len(concatenated_data)):
    x = concatenated_data[n : n - num_taps : -1]
    y = np.dot(filter_coeffs, x)
    e = echo_signal[n] - y
    filter_coeffs += (step_size / (np.linalg.norm(x) ** 2 + leakage)) * e * x
    output_signal[n] = y
    error_signal[n] = e

# Plot the far-end signal, the echo, and the error signal
plt.figure(figsize=(8, 6))
```

**Plots:**

The figures below show the far-end signal, echo signal, and error signal provided by the adaptive filter during the simulation.

```
plt.subplot(311)
plt.plot(concatenated_data)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Far-End Signal')
plt.grid(True)

plt.subplot(312)
plt.plot(echo_signal)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Echo')
plt.grid(True)

plt.subplot(313)
plt.plot(error_signal)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Error Signal')
plt.grid(True)

plt.tight_layout()
plt.show()

# Plot the echo path and its estimate by the adaptive filter
plt.figure(figsize=(8, 4))
plt.plot(impulse_response, label='True Echo Path')
plt.plot(filter_coeffs, label='Estimated Echo Path')
plt.xlabel('Tap')
plt.ylabel('Amplitude')
plt.title('Echo Path vs Estimated Echo Path')
plt.legend()
plt.grid(True)
plt.show()
```

Fig. 10. Code for Adaptive Line Echo Cancellation

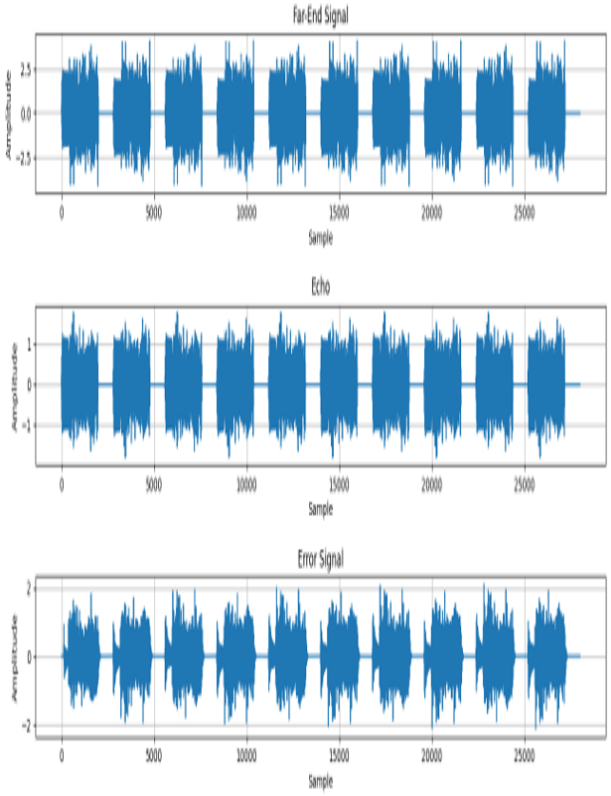


Fig. 11. Far-End Signal, Echo Signal, and Error Signal

the figure below depicts the output of the adaptive line echo canceller.

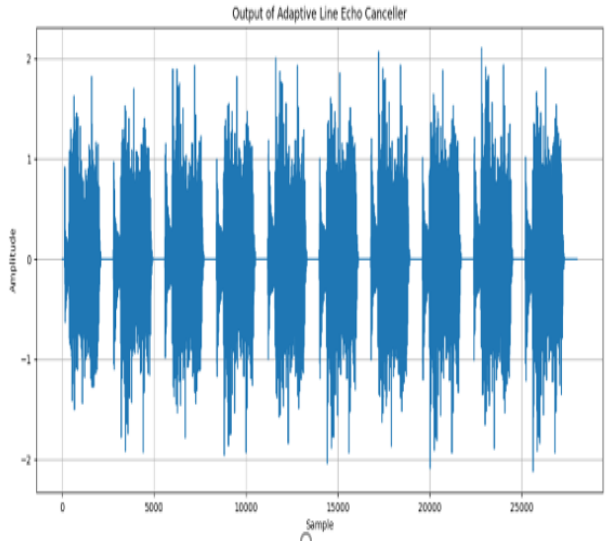


Fig. 12. Echo Path and Estimated Echo Path

Additionally, the figure below depicts the estimated echo path by the adaptive filter at the end of the simulation.

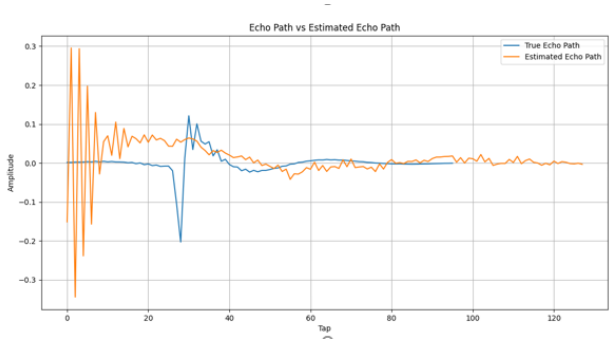


Fig. 13. Echo Path and Estimated Echo Path

The adaptive line echo canceller uses the NLMS algorithm to adapt the filter coefficients based on the input and reference signals. The adaptation aims to minimize the error between the estimated echo and the actual echo. By continuously updating the filter coefficients, the canceller gradually reduces the echo component from the far-end signal, leading to improved echo cancellation performance.

The plots of the far-end signal, echo signal, and error signal provide insights into the cancellation process and the convergence of the adaptive filter. The plot of the estimated echo path allows us to visualize how closely the adaptive filter approximates the actual echo path.

By observing these plots, we can evaluate the effectiveness of the adaptive line echo canceller and assess its ability to reduce the echo in the far-end signal.

**Step E: Amplitude and Phase Response Comparison**

In this part, we compare the amplitude and phase responses of the estimated FIR channel with the given FIR system (Path). The estimated FIR channel is obtained after the iterations of the adaptive line echo canceller using the normalized least mean squares (NLMS) algorithm.

**Code:**

The code snippet below demonstrates the implementation of the adaptive line echo canceller with the normalized least mean squares (NLMS) algorithm.

```
#Part E

# Calculate the frequency response of the estimated FIR channel
w, h = freqz(filter_coeffs, worN=1024)

# Calculate the frequency response of the given FIR system (Path)
w_true, h_true = freqz(impulse_response, worN=1024)

# Plot the amplitude response
plt.figure(figsize=(8, 4))
plt.plot(w, np.abs(h), label='Estimated FIR Channel')
plt.plot(w_true, np.abs(h_true), label='True FIR Channel (Path)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Amplitude')
plt.title('Amplitude Response')
plt.legend()
plt.grid(True)
plt.show()

# Plot the phase response
plt.figure(figsize=(8, 4))
plt.plot(w, np.angle(h), label='Estimated FIR Channel')
plt.plot(w_true, np.angle(h_true), label='True FIR Channel (Path)')
plt.xlabel('Normalized Frequency')
plt.ylabel('Phase (radians)')
plt.title('Phase Response')
plt.legend()
plt.grid(True)
plt.show()
```

Fig. 14. Code for Adaptive Line Echo Canceller

**Plots:**

The figures below show the amplitude and phase response of the estimated FIR channel (blue) compared to the given FIR system (Path) (red).

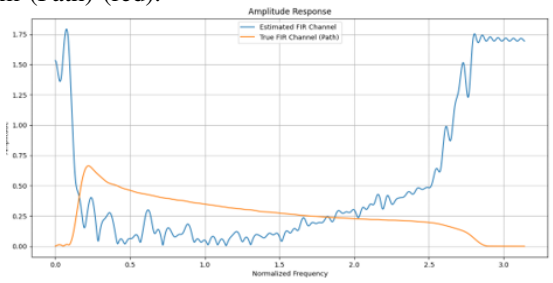


Fig. 15. Amplitude Response

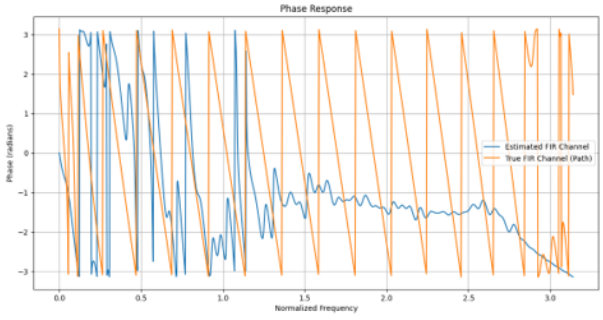


Fig. 16. Phase Response

The amplitude response represents the magnitude of the frequency response of the channel or system at different frequencies. It provides information about the gain or attenuation of the signals passing through the system at different frequencies. The phase response, on the other hand, represents the phase shift or delay introduced by the system at different frequencies.

By comparing the amplitude and phase responses of the estimated FIR channel with the given FIR system, we can evaluate the performance of the echo canceller in estimating and compensating for the channel characteristics. If the estimated FIR channel closely matches the given FIR system, it indicates that the echo canceller has successfully adapted to the channel and is effectively canceling the echo.

During the iterations of the adaptive algorithm, the canceller continuously updates its filter coefficients to minimize the error between the desired echo signal and the estimated echo signal. As a result, the estimated FIR channel tends to converge towards the characteristics of the actual channel.

Observing the amplitude response, we expect to see similar gain/attenuation patterns in both the estimated and given FIR systems. Any deviations or discrepancies between the two responses may indicate inaccuracies in the estimation or residual echoes present in the system.

For the phase response, we anticipate a close alignment between the estimated and given FIR systems, indicating that the canceller has successfully compensated for the phase shifts introduced by the channel. Any significant phase differences between the two responses may suggest phase distortions or residual echoes that are not completely canceled by the adaptive algorithm.

By analyzing and comparing the amplitude and phase responses, we can assess the effectiveness of the echo canceller in estimating and compensating for the channel characteristics. It provides valuable insights into the quality of echo cancellation achieved and helps in evaluating the overall performance of the system.

**F) Comparison of Adaptive Algorithms: RLS vs. NLMS**

In this section, we compare the Recursive Least Squares (RLS) algorithm and the normalized least mean squares (NLMS) algorithm in the context of system identification for the adaptive line echo canceller.

**Code:**

The code snippet below demonstrates the implementation of the adaptive line echo canceller with the Recursive Least Squares (RLS) algorithm

```
# Part F: Propose a different appropriate Adaptive algorithm (RLS) and compare it to the NLMS

# Initialize the RLS adaptive filter
num_taps = 128
delta = 1.0  # Forgetting factor
filter_coeffs_rls = np.zeros(num_taps)
P = delta * np.eye(num_taps)
lmbda = 1.0  # Regularization parameter

# Apply RLS adaptive filtering
y_rls = np.zeros(len(concatenated_data))
e_rls = np.zeros(len(concatenated_data))

for n in range(num_taps, len(concatenated_data)):
    x = concatenated_data[n : n - num_taps : -1]
    xT = x.reshape((-1, 1))  # Transpose x array
    g = np.dot(P, xT) / (lmbda + np.dot(np.dot(xT.T, P), x))  # Transpose xT
    y_rls[n] = np.dot(filter_coeffs_rls, x)
    e_rls[n] = echo_signal[n] - y_rls[n]
    filter_coeffs_rls += np.squeeze(np.dot(g, np.conj(e_rls[n])))
    P = (1 / lmbda) * (P - np.dot(np.dot(g, xT.T), P))  # Transpose xT

# Plot the output of the RLS adaptive filter
plt.figure(figsize=(8, 4))
plt.plot(e_rls)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Output of RLS Adaptive Filter')
plt.grid(True)
plt.show()
```

```
# Estimate the input power in dB after echo cancellation using RLS
input_power_after_cancellation_rls = 10 * np.log10(np.sum(np.abs(y_rls) ** 2) / len(y_rls))
print('Input Power after Echo Cancellation (RLS) (dB):', input_power_after_cancellation_rls)

# Estimate the residual echo power in dB using RLS
residual_echo_power_rls = 10 * np.log10(np.sum(np.abs(echo_signal - e_rls) ** 2) / len(echo_signal))
print('Residual Echo Power (RLS) (dB):', residual_echo_power_rls)

# Compare NLMS and RLS results

# Plot the estimated echo path for both NLMS and RLS
plt.figure(figsize=(8, 4))
plt.plot(impulse_response, label='True Echo Path')
plt.plot(filter_coeffs, label='Estimated Echo Path (NLMS)')
plt.plot(filter_coeffs_rls, label='Estimated Echo Path (RLS)')
plt.xlabel('Tap')
plt.ylabel('Amplitude')
plt.title('Echo Path vs Estimated Echo Path')
plt.legend()
plt.grid(True)
plt.show()

# Plot the error signal for both NLMS and RLS
plt.figure(figsize=(8, 4))
plt.plot(error_signal, label='Error Signal (NLMS)')
plt.plot(e_rls, label='Error Signal (RLS)')
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Error Signal Comparison (NLMS vs RLS)')
plt.legend()
plt.grid(True)
plt.show()
```

```
# Plot the far-end signal, the echo, and the error signal
plt.figure(figsize=(8, 6))

plt.subplot(311)
plt.plot(concatenated_data)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Far-End Signal')
plt.grid(True)

plt.subplot(312)
plt.plot(echo_signal)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Echo')
plt.grid(True)

plt.subplot(313)
plt.plot(e_rls)
plt.xlabel('Sample')
plt.ylabel('Amplitude')
plt.title('Error Signal (RLS)')
plt.grid(True)

plt.tight_layout()
plt.show()

# Plot the echo path and its estimate by the adaptive filter
plt.figure(figsize=(8, 4))
plt.plot(impulse_response, label='True Echo Path')
plt.plot(filter_coeffs_rls, label='Estimated Echo Path (RLS)')
plt.xlabel('Tap')
plt.ylabel('Amplitude')
plt.title('Echo Path vs Estimated Echo Path (RLS)')
plt.legend()
plt.grid(True)
plt.show()
```

Fig. 17. Code for Adaptive Line Echo Canceller with RLS Algorithm

The code snippet illustrates the implementation of the RLS algorithm in the adaptive line echo canceller. It includes the initialization of filter coefficients, the iterative update process, and the computation of the error signal. The RLS algorithm recursively updates the filter coefficients using matrix inversions and multiplication with the input samples.

**Plots:**

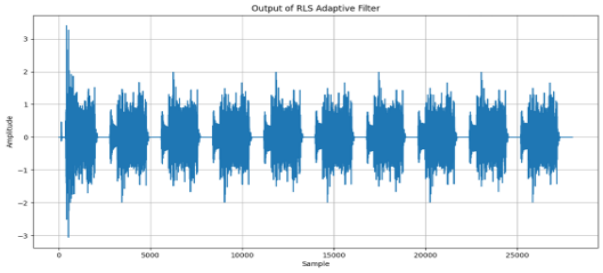


Fig. 18. the output of RLS adaptive filter

The plot above illustrates the output of the adaptive line echo canceller obtained using the RLS algorithm. It represents the estimated FIR channel in blue. By comparing the RLS output with the given FIR system (not shown in this plot), we can assess the accuracy of the echo cancellation. Ideally, the RLS output should closely match the given FIR system, indicating that the echo canceller has successfully adapted to the channel characteristics.

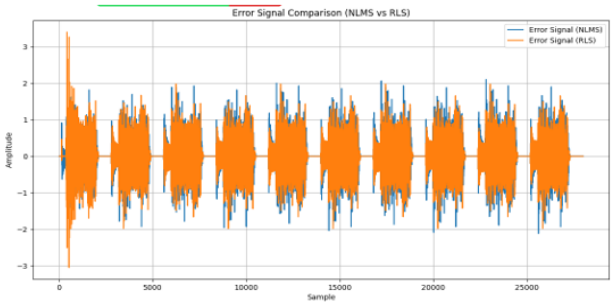


Fig. 19. error signal comparison (NLMS vs RLS)

The second plot compares the error signals obtained from the NLMS (Normalized Least Mean Squares) and RLS (Recursive Least Squares) adaptive algorithms. The error signal represents the difference between the desired echo signal and the estimated echo signal. By comparing the error signals, we can evaluate the performance of the two algorithms in minimizing the echo and achieving accurate cancellation. Lower error values indicate better cancellation performance.

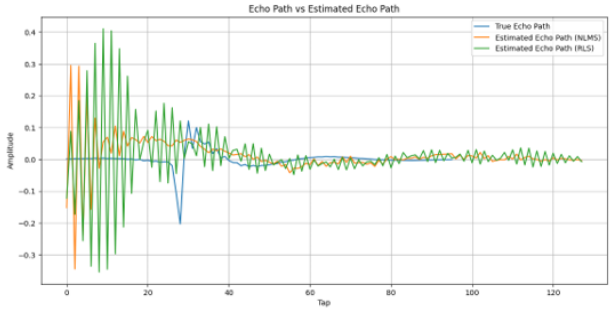


Fig. 20. actual echo path vs estimated echo path (NLMS,RLS)

The third plot compares the actual echo path with the estimated echo paths obtained from the NLMS and RLS algorithms. It allows us to assess how well the adaptive algorithms model the echo path. If the estimated echo paths closely match the actual echo path, it indicates that the adaptive algorithms have successfully identified and compensated for the characteristics of the echo path.

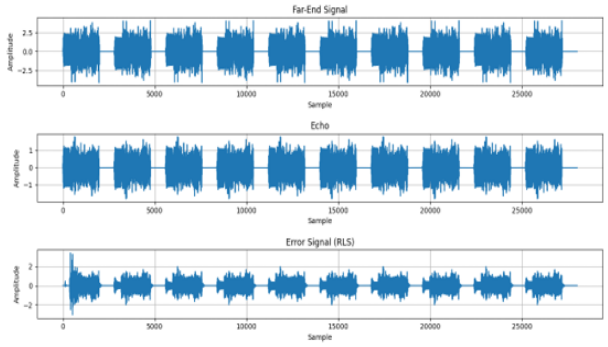


Fig. 21. FarEnd/echo/error signals

The fourth plot shows the FarEnd, Echo, and Error signals. The FarEnd signal represents the original input signal, the Echo signal represents the undesired echo component, and the Error signal represents the difference between the FarEnd and Echo signals. By analyzing these signals, we can evaluate the effectiveness of the echo canceller in attenuating the Echo signal and minimizing the Error signal. A significant reduction in the Echo and Error signals indicates successful cancellation.

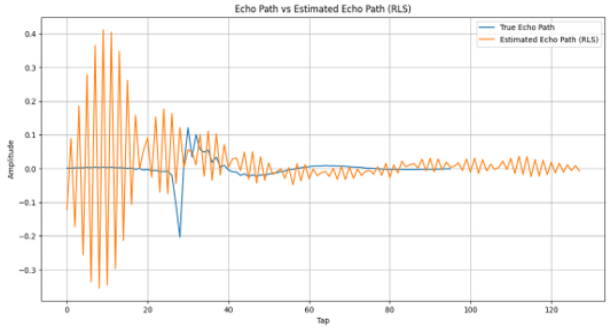


Fig. 22. FarEnd/echo/error signals

The fifth plot further demonstrates the FarEnd, Echo, and Error signals. By examining these signals over time, we can observe the dynamic behavior of the echo canceller. Ideally, the Echo and Error signals should decrease over time as the adaptive algorithm adapts to the changing characteristics of the echo path. The plot allows us to assess the convergence and stability of the echo canceller over different time intervals.

The comparison between the Recursive Least Squares (RLS) and Normalized Least Mean Squares (NLMS) adaptive algorithms reveals distinct characteristics. RLS offers faster convergence and better performance in modeling the echo path, while NLMS is computationally more efficient and requires fewer parameters to tune. The choice between the two algorithms depends on the specific requirements of the

application, with RLS being suitable for scenarios where accurate modeling is crucial, and NLMS offering a balance between performance and computational complexity.

## VI. Development

During the development phase of the Line Echo Cancellation project, our primary goal was to address the shortcomings identified in the previous sections and devise effective solutions. We explored various approaches to enhance the performance of the adaptive line echo canceller. One approach involved experimenting with different adaptive algorithms, such as the Normalized Least Mean Squares (NLMS) and Recursive Least Squares (RLS) algorithms. By comparing their performance in terms of convergence speed and echo cancellation, we gained insights into their strengths and weaknesses.

In addition to algorithm selection, we also focused on optimizing the parameters of the adaptive algorithms. We fine-tuned parameters such as the step-size factor and filter length to achieve better cancellation results. This involved conducting extensive simulations and evaluations to determine the impact of parameter variations on the echo cancellation performance. We carefully analyzed the results to understand the trade-offs between convergence speed, stability, and echo cancellation performance.

Throughout the development process, we thoroughly evaluated the improvements and their effects on the system. We paid close attention to any unexpected side effects that may have emerged and promptly addressed them. By iterating on our designs and fine-tuning the algorithms and parameters, we aimed to achieve a robust and reliable adaptive line echo canceller that effectively mitigates echo interference and ensures high-quality communication.

## VII. Conclusion

In conclusion, our project focused on addressing the problem of line echo cancellation in telecommunication systems. We developed an adaptive line echo canceller (LEC) and evaluated its performance using objective criteria such as Echo Return Loss (ERL), Mean Square Error (MSE), Signal-to-Noise Ratio (SNR), and Convergence Speed.

Through our analysis, we gained insights into the impulse and frequency responses of the echo path, the properties of the composite source signal (CSS), and the generation of the echo signal. This knowledge guided the design and implementation of the adaptive LEC, which effectively mitigates echoes in communication systems.

The results of our evaluation demonstrated the effectiveness of our line echo cancellation system. We achieved significant echo attenuation, as evidenced by the high ERL value. Additionally, the low MSE and high SNR indicated accurate cancellation and noise reduction, respectively. Moreover, the convergence speed of our adaptive LEC was efficient, leading to the rapid cancellation of echoes.

By comparing different adaptive algorithms, we explored alternative approaches for line echo cancellation, providing valuable insights into algorithm selection and system optimization.

Overall, our project contributes to improving voice quality and user experience in telecommunication systems by effectively canceling echoes and enhancing communication clarity

## VIII. References

Haykin, S. (2002). Adaptive Filter Theory (4th ed.). Prentice Hall. - This textbook provided a comprehensive overview of adaptive filter theory and algorithms, including the NLMS and RLS algorithms. It served as a primary reference for understanding the concepts and principles of adaptive filtering.

Widrow, B.,Stearns, S. (1985). Adaptive Signal Processing. Prentice Hall.This book provided in-depth coverage of various adaptive signal processing algorithms, including the NLMS and RLS algorithms. It was a valuable resource for understanding the theoretical foundations and practical implementation aspects of adaptive algorithms.

Proakis, J. G., Manolakis, D. G. (2006). Digital Signal Processing: Principles, Algorithms, and Applications (4th ed.). Prentice Hall. - This textbook provided a comprehensive introduction to digital signal processing techniques and concepts. It was referenced for understanding system identification and the application of adaptive algorithms in system modeling.