

**PROJECT REPORT**

**ECHO CANCELLATION SYSTEM IN CALLS**

Using adaptive filters

**SUBMITTED TO**

MA’AM SHAHINZA MANZOOR

**SUBMITTED BY**

NAILA SHEHZADI (008)

**SUBJECT**

DIGITAL SIGNAL PROCESSING LAB

**Adaptive Filter for Echo Cancellation using LMS Algorithm**

**1. Introduction**

Echo cancellation is a crucial aspect of signal processing in various communication systems, such as telephony, video conferencing, and voice assistants. This project aims to design an adaptive filter that can effectively cancel the echo from a signal using the Least Mean Squares (LMS) algorithm.

The basic concept behind echo cancellation involves using an adaptive filter to predict the echo in a signal and subtract it from the signal to obtain the original, clean signal. In this case, we generate a near-end signal by adding an artificial echo to the original speech signal and then use the LMS algorithm to cancel the echo.

**2. Problem Definition**

The objective of this project is to implement an adaptive filter using the LMS algorithm to cancel the echo from a speech signal. The process includes:

* Generating an echo signal by simulating a delayed and attenuated version of the original speech.
* Adding the echo to the original signal to create the "near-end" signal.
* Using the LMS adaptive filter to predict the echo and subtract it from the near-end signal.
* Saving the final echo-canceled signal as an output audio file.

**3. Approach and Methodology**

**Step 1: Loading the Audio Signal**

The project begins by loading the original speech signal (near\_end\_signal.wav) and selecting only the first channel (mono). This signal serves as the clean, original speech without any echo.

**Step 2: Generating the Echo Signal**

An artificial echo is generated by introducing a delay and an attenuation factor to the original signal. The echo signal is created by shifting the original signal and multiplying it by a predefined gain. The delay is specified in seconds, and the gain determines how loud the echo is compared to the original signal.

**Step 3: Creating the Near-End Signal**

The near-end signal is created by adding the generated echo signal to the original speech. This results in a signal that contains both the original speech and the unwanted echo.

**Step 4: Adaptive Filter Initialization**

An adaptive filter of length M is initialized. The filter's weights are initially set to zero, and the LMS algorithm will iteratively adjust these weights to minimize the error between the near-end signal and the echo estimate.

**Step 5: Applying the LMS Algorithm**

The LMS algorithm is applied to the near-end signal. At each time step:

* The input buffer is updated with the new sample from the near-end signal.
* The adaptive filter output (echo estimate) is computed as the dot product of the filter weights and the input buffer.
* The error signal, which is the difference between the near-end signal and the filter output, is computed.
* The filter weights are updated using the LMS update rule, which adjusts the weights in the direction that minimizes the error.

**Step 6: Resulting Echo-Canceled Signal**

The output of the adaptive filter is stored as the error signal, which ideally represents the original speech without the echo.

**Step 7: Saving and Plotting Results**

The echo-canceled signal is saved as a new audio file (echo\_canceled\_output.wav). Additionally, the frequency spectrum of the near-end signal and the echo-canceled signal is plotted to visualize the effectiveness of the echo cancellation.

**Step 8: Playback and Visualization**

The following signals are played for comparison:

1. The near-end signal with the added echo.
2. The echo-canceled signal.

**4. Results and Discussion**

**Audio Playback**

The near-end signal with echo, and echo-canceled signal were played to compare the effectiveness of the echo cancellation.

* **Original Signal**: The original speech signal contains no echo.
* **Near-End Signal**: The near-end signal includes the original speech plus the added echo. It is a mixture of the clean speech and the echo.
* **Echo-Canceled Signal**: The adaptive filter effectively cancels the echo, leaving only the original speech, though some residual noise may still exist.

**Frequency Spectrum Analysis**

The frequency spectrum of both the near-end signal (with echo) and the echo-canceled signal were plotted to observe how the echo cancellation affects the frequency components.

* The near-end signal shows a broader spectrum due to the presence of the echo.
* The echo-canceled signal should exhibit a spectrum more similar to the original signal, with significant reduction in the frequency components caused by the echo.

**Performance of the LMS Algorithm**

The LMS algorithm adjusts the filter weights iteratively, and the convergence rate depends on the chosen step-size mu and the filter length M. In this case, the error signal is the result of subtracting the echo estimate from the near-end signal. As the filter adapts over time, the echo component is increasingly minimized.

**5. Conclusion**

In this project, an adaptive filter using the LMS algorithm was successfully implemented to cancel the echo from a speech signal. The near-end signal, containing the original speech and the echo, was processed, and the echo was effectively reduced through the LMS-based adaptive filtering approach. The final output, an echo-canceled signal, was saved as a new audio file. The project demonstrates the effectiveness of adaptive filtering techniques in echo cancellation for speech signals.

**6. Future Work**

Future improvements to this project could involve:

* **Real-Time Processing**: Implementing the system in real-time for live audio processing.
* **Advanced Echo Cancellation Algorithms**: Exploring other adaptive algorithms, such as Recursive Least Squares (RLS), for better performance in more challenging environments.
* **Noise Robustness**: Extending the system to handle both echo and background noise simultaneously.

This project serves as a foundation for building more advanced echo cancellation systems suitable for real-world applications like telecommunication systems, conferencing tools, and hearing aids.

**MATLAB Code:**

[original\_signal, Fs] = audioread('near\_end\_signal.wav');

original\_signal = original\_signal(:, 1);

echo\_delay = 0.3;

echo\_gain = 0.5;

M = 128;

mu = 0.001;

delay\_samples = round(echo\_delay \* Fs);

echo\_signal = [zeros(delay\_samples, 1);

original\_signal(1:end-delay\_samples)] \* echo\_gain;

near\_end\_signal = original\_signal + echo\_signal;

w = zeros(M, 1);

y = zeros(length(near\_end\_signal), 1);

e = zeros(length(near\_end\_signal), 1);

x\_buffer = zeros(M, 1);

for n = M:length(near\_end\_signal)

x\_buffer = [near\_end\_signal(n); x\_buffer(1:M-1)];

y(n) = w' \* x\_buffer;

e(n) = near\_end\_signal(n) - y(n);

w = w + mu \* e(n) \* x\_buffer;

end

t = (0:length(near\_end\_signal)-1) / Fs;

figure;

subplot(3, 1, 1);

plot(t, near\_end\_signal);

title('Near-End Signal (Speech + Echo)');

xlabel('Time (s)');

ylabel('Amplitude');

subplot(3, 1, 2);

plot(t, y);

title('Adaptive Filter Output (Echo Estimate)');

xlabel('Time (s)');

ylabel('Amplitude');

subplot(3, 1, 3);

plot(t, e);

title('Error Signal (Echo-Canceled Signal)');

xlabel('Time (s)');ylabel('Amplitude');

disp('Playing the near-end signal (with echo)...');

sound(near\_end\_signal, Fs);

pause(length(near\_end\_signal) / Fs + 1);

disp('Waiting before playing the echo-canceled signal...');

pause(1);

disp('Playing the echo-canceled signal...');

sound(original\_signal, Fs);

pause(length(original\_signal) / Fs - 1);

output\_file = 'echo\_canceled\_output.wav';

audiowrite(output\_file, e, Fs);

disp(['The echo-canceled signal has been saved as "' output\_file '".']);

figure;

subplot(2,1,1);

f\_near\_end = fft(near\_end\_signal);

f\_near\_end = abs(f\_near\_end(1:length(near\_end\_signal)/2));

frequencies = linspace(0, Fs/2, length(f\_near\_end));

plot(frequencies, f\_near\_end);

title('Frequency Spectrum of Near-End Signal (Speech + Echo)');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

subplot(2,1,2);

f\_echo\_canceled = fft(e);

f\_echo\_canceled = abs(f\_echo\_canceled(1:length(e)/2));

plot(frequencies, f\_echo\_canceled);

title('Frequency Spectrum of Echo-Canceled Signal');

xlabel('Frequency (Hz)');

ylabel('Magnitude');

**RESULT:**

