Hochschule Karlsruhe
University of
Applied Sciences

MATLAB GUI
for
Active Noise Cancellation using Adaptive Filtering

By

Nithil A Rao

#93851

Overview

- Introduction to Active Noise Cancellation
- Overview of the MATLAB GUI Application
- Code and Algorithms
- Signal Processing and Mathematical Concepts
- Demonstration of the Application

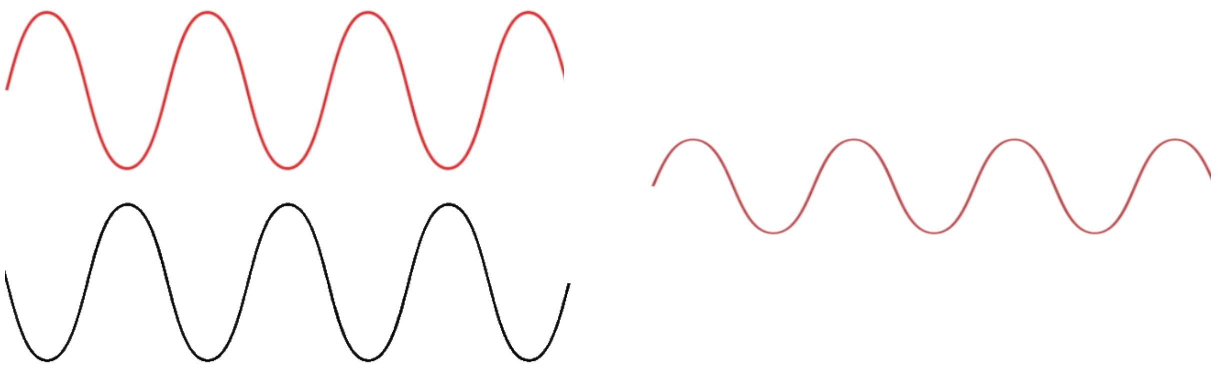
What is Active Noise Cancellation?

Definition:

- Active Noise Cancellation is a method of reducing unwanted sound by adding a second sound specifically designed to cancel the first.

Concept Illustration:

- Uses the principle of destructive interference where two sound waves of equal amplitude and opposite phase cancel each other out.



Applications of ANC

Automotive Industry:

- *Enhancing passenger comfort by reducing engine and road noise.*

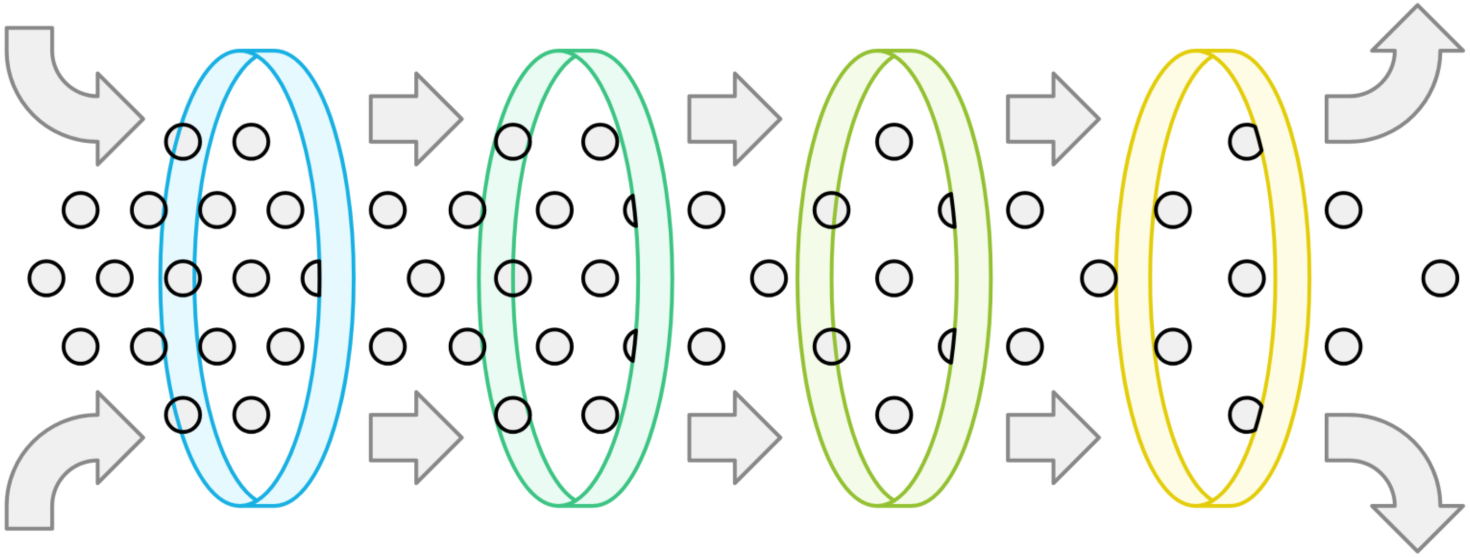
Consumer Electronics:

- *Noise-cancelling headphones and earphones.*

Industrial Settings:

- *Reducing noise pollution in environments like factories.*

Overview of the MATLAB GUI Application



Load Signals

Import audio files into the system

Visualize Signals

Display audio waveforms for analysis

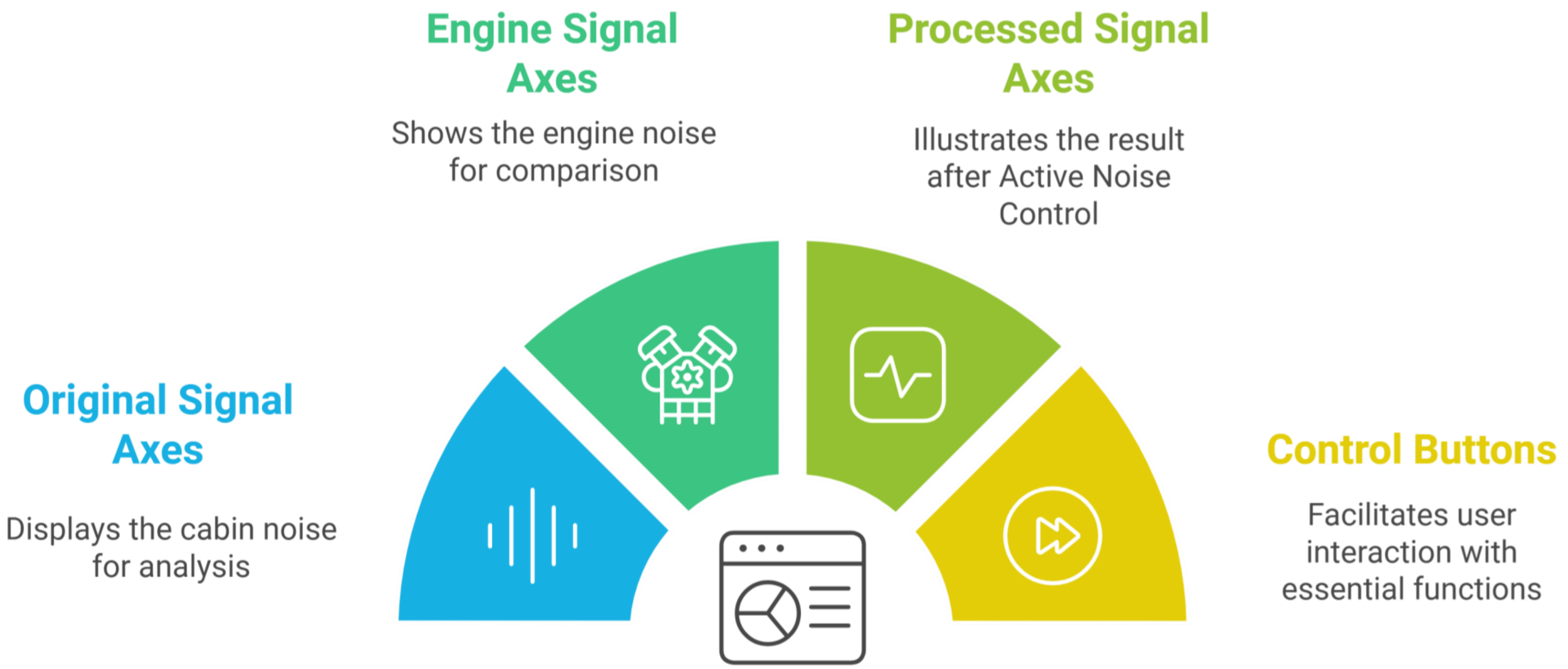
Apply ANC

Implement noise cancellation technique

Play Audio

Listen to processed audio output

User Interface Layout



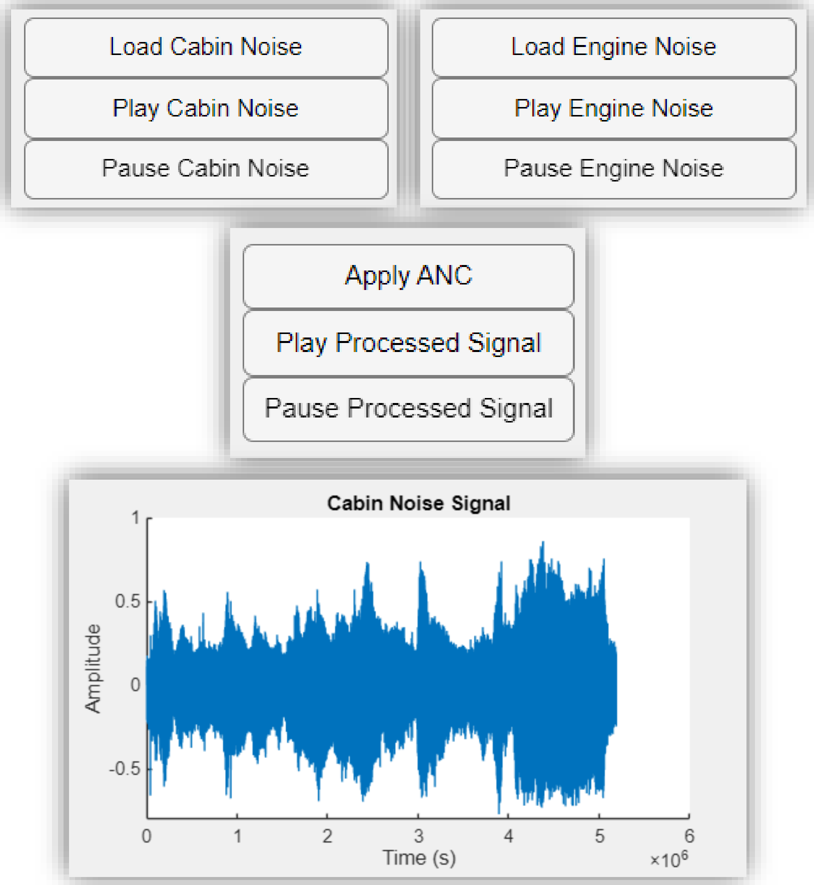
Loading and Visualizing Signals

Loading Signals:

- Users can load audio files representing cabin and engine noises.
- Supports common audio formats like WAV and MP3.

Visualization:

- Signals are plotted for visual analysis.
- Helps in understanding the characteristics of the noises.



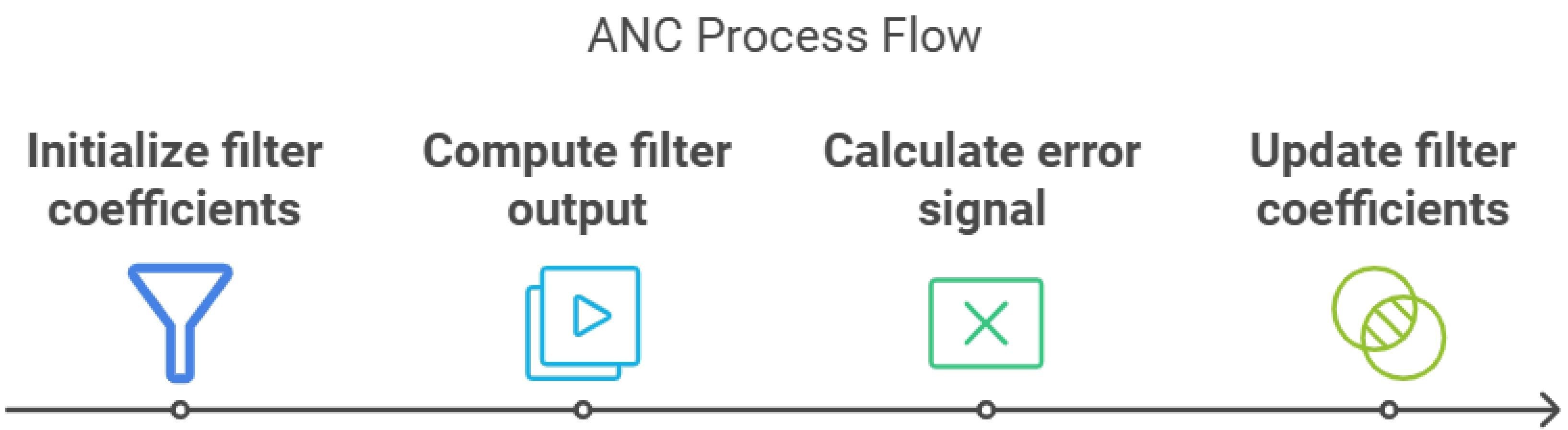
The ANC Algorithm

Adaptive Filtering:

- *Uses the Least Mean Squares (LMS) algorithm.*

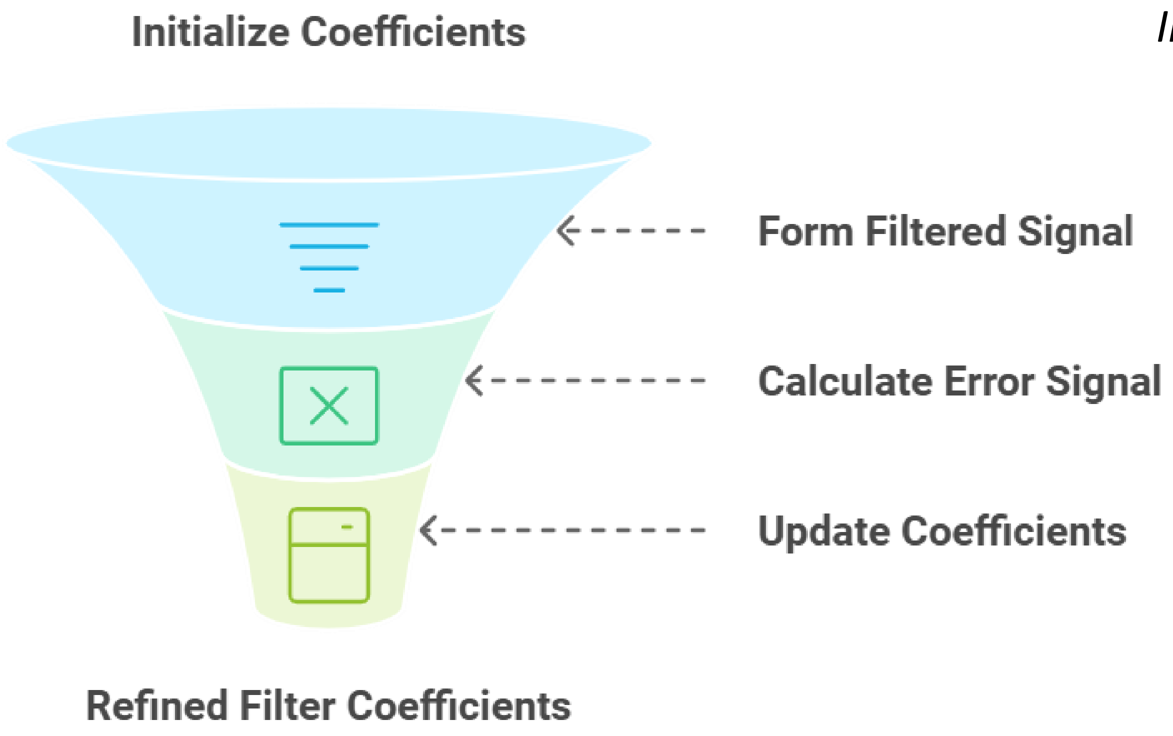
Purpose of LMS:

- *Adjusts filter coefficients to minimize the error between the desired and actual signal.*



Mathematical Foundations

LMS Algorithm



Initialize the filter coefficient vector w(0) to zero.

$y(n) = w(n-1)^T * x(n)$

$e(n) = d(n) - y(n)$

$w(n) = w(n-1) + \mu * e(n) * x(n)$

Example

- **d(n):** [1, 2, 3, 4, 5] (cabin noise)
- **x(n):** [0.5, 1, 1.5, 2, 2.5] (engine noise)
- **μ:** 0.1 (Step Rate)
- **M: 2** (Filter Order)

Initialization:

- $w(0) = [0, 0]$

Iteration 1:

- $y(1) = w(0)^T * x(1) = 0$
- $e(1) = d(1) - y(1) = 1 - 0 = 1$
- $w(1) = w(0) + \mu * e(1) * x(1) = [0, 0] + 0.1 * 1 * [0.5] =$ **[0.05, 0]**

Iteration 2:

- $y(2) = w(1)^T * x(2) = 0.05 * 1 = 0.05$
- $e(2) = d(2) - y(2) = 2 - 0.05 = 1.95$
- $w(2) = w(1) + \mu * e(2) * x(2) = [0.05, 0] + 0.1 * 1.95 * [1] =$ **[0.245, 0]**

Example

n=1
y(1) = 0
e(1) = 1
w(1) = [0.05, 0]

n=2
y(2) = 0.05
e(2) = 1.95
w(2) = [0.245, 0]

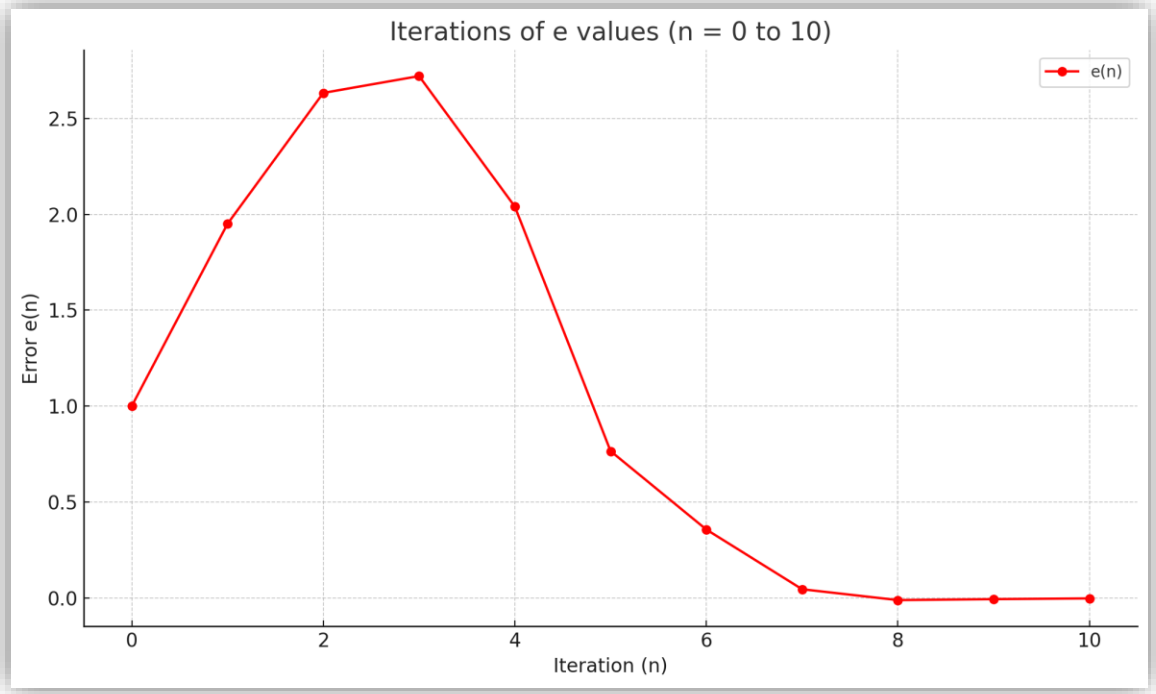
n=3:
y(3)=1.27
e(3)=2.72
w(4)=[1.18,0]

n=4:
y(4)=2.95
e(4)=2.04
w(5)=[1.69,0]

n=5:
y(5)=4.23
e(5)=0.76
w(6)=[1.88,0]

n=6:
y(6)=5.64
e(6)=0.35
w(7)=[1.98,0]

n=7:
y(7)=6.95
e(7)=0.04
w(8)=[2.00,0]



MATLAB Code

```
classdef automotiveANCApp < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                  matlab.ui.Figure
        LoadNoiseButton           matlab.ui.control.Button
        LoadEngineNoiseButton     matlab.ui.control.Button
        ApplyANCButton            matlab.ui.control.Button
        PlayOriginalButton        matlab.ui.control.Button
        PlayEngineButton          matlab.ui.control.Button
        PlayProcessedButton       matlab.ui.control.Button
        PauseOriginalButton       matlab.ui.control.Button
        PauseEngineButton         matlab.ui.control.Button
        PauseProcessedButton      matlab.ui.control.Button
        OriginalSignalAxes        matlab.ui.control.UIAxes
        EngineSignalAxes          matlab.ui.control.UIAxes
        ProcessedSignalAxes       matlab.ui.control.UIAxes
    end

    % Properties for internal data storage
    properties (Access = private)
        cabinNoise                % Cabin noise signal
        engineNoise               % Engine noise signal (reference)
        processedSignal           % Signal after ANC
        Fs                        % Sampling frequency
        originalPlayer            % Audio player for original noise
        enginePlayer              % Audio player for engine noise
        processedPlayer           % Audio player for processed signal
    end
```

1.Initialising properties

```
    % Adaptive LMS filter function
    function [y, e] = lmsFilter(app, d, x, mu, filterOrder)
        nIterations = length(d);
        y = zeros(nIterations, 1);
        e = zeros(nIterations, 1);
        w = zeros(filterOrder, 1);

        % Initialize Progress Dialog
        progressDlg = uiprogressdlg(app.UIFigure, 'Title', 'Processing',
            'Message', 'Applying ANC...', 'Cancelable', 'off', ...
            'Indeterminate', 'on');

        for n = filterOrder:nIterations
            x_vec = x(n:-1:n-filterOrder+1);

            % Ensure x_vec is a column vector
            if isrow(x_vec)
                x_vec = x_vec';
            end

            y(n) = w' * x_vec;
            e(n) = d(n) - y(n);
            w = w + 2 * mu * e(n) * x_vec;

            % Update progress every 1000 iterations or at the end
            if mod(n, 1000) == 0 || n == nIterations
                progressDlg.Value = n / nIterations;
                progressDlg.Message = sprintf('Applying ANC... %.2f%%', (n / nIterations)*100);
                drawnow;
            end
        end
```

3.LMS Filter Function

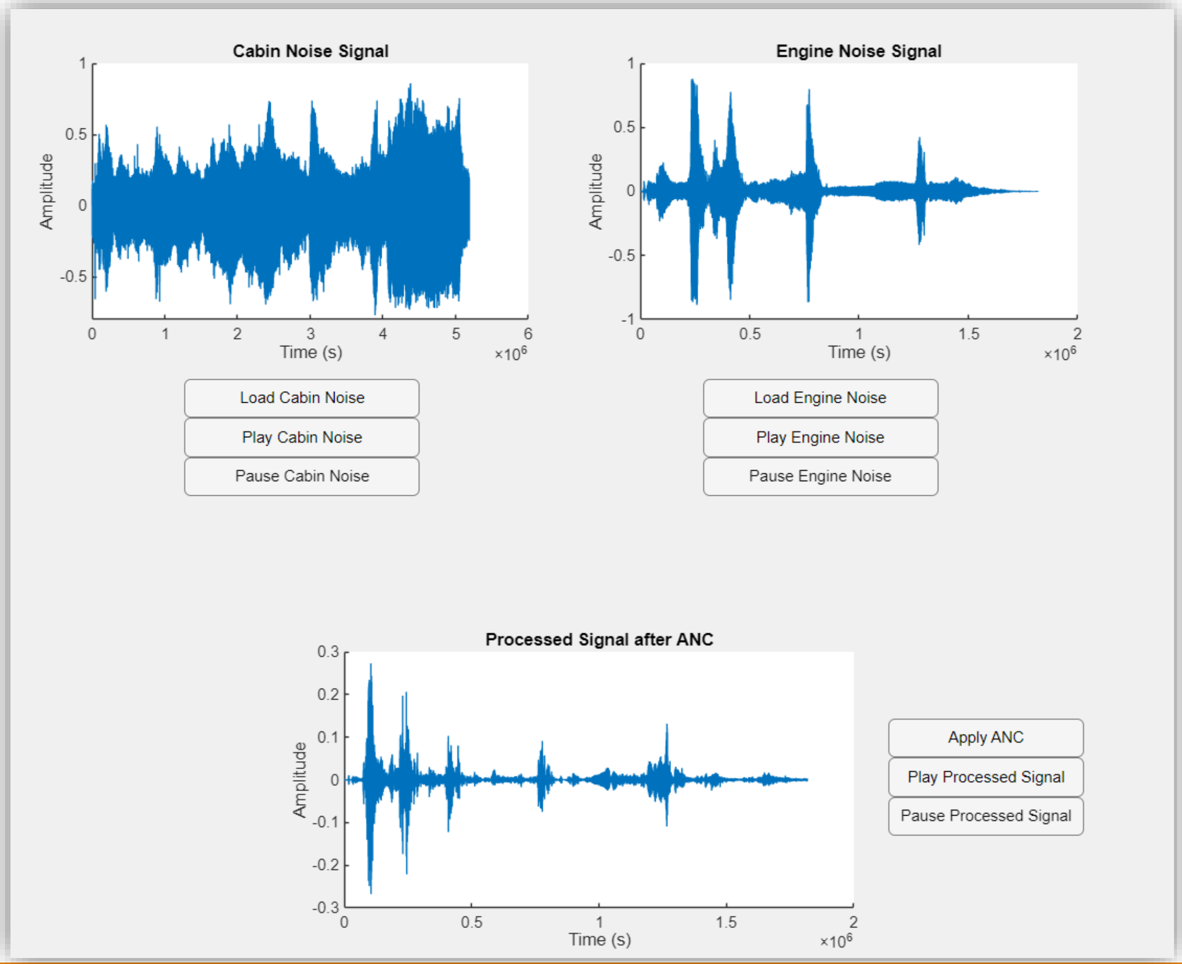
```
        % Adjust signals to the same length
        len = min(length(app.cabinNoise), length(app.engineNoise));
        d = app.cabinNoise(1:len);
        x = app.engineNoise(1:len);

        % Ensure d and x are column vectors
        if isrow(d)
            d = d';
        end
        if isrow(x)
            x = x';
        end

        % Apply Adaptive Noise Cancellation
        mu = 0.001; % Step size
        filterOrder = 64;
        [y, ~] = app.lmsFilter(d, x, mu, filterOrder); % Corrected call
        app.processedSignal = y;
        plot(app.ProcessedSignalAxes, app.processedSignal);
        title(app.ProcessedSignalAxes, 'Processed Signal after ANC');
        xlabel(app.ProcessedSignalAxes, 'Time (s)');
        ylabel(app.ProcessedSignalAxes, 'Amplitude');
        app.processedPlayer = audioplayer(app.processedSignal, app.Fs);
    end
```

2.Adjusting Signal Length

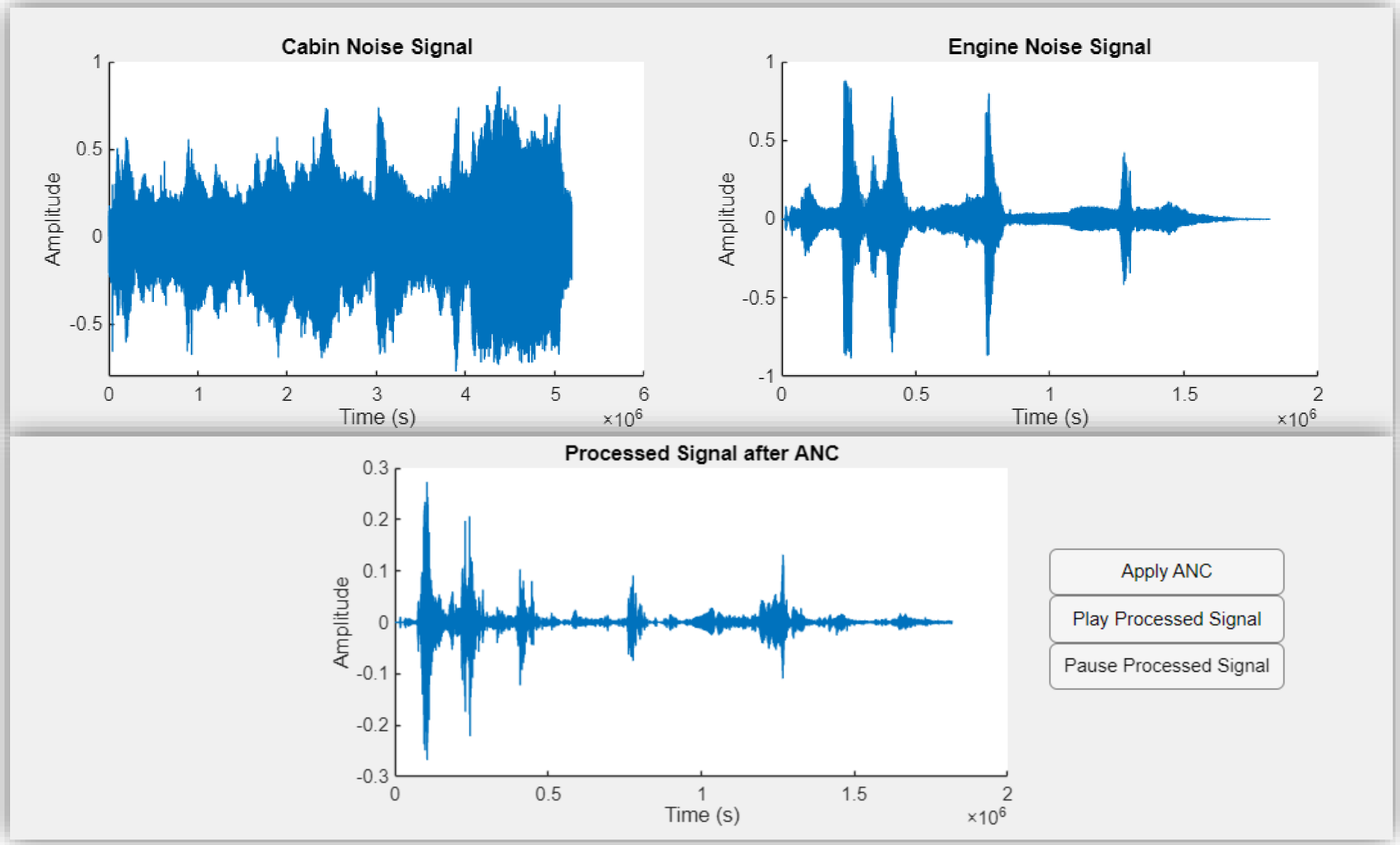
Demonstration



Comparison of LMS Adaptive Filtering vs. Destructive Interference

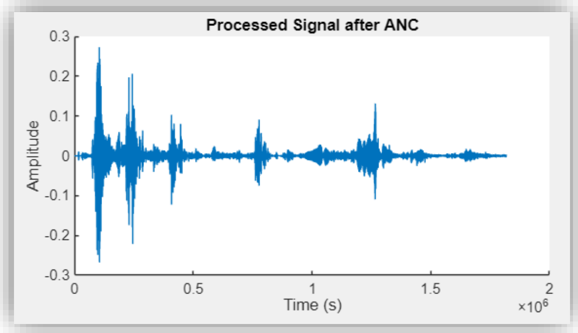
Aspect	LMS Adaptive Filtering	Destructive Interference
Concept	Adapts to reduce noise error	Creates an opposite wave to cancel noise
Noise Type	Works with changing noise patterns	Works best with steady, predictable noise
Flexibility	Adjusts continuously	Limited to consistent noise
Approach	Minimizes error, no exact phase inversion	Uses precise phase inversion
Strength	Handles complex, dynamic noise	Effective for constant noise
Weakness	Needs time to adapt	Struggles with varying noise

Signal Analysis



Role of Parameters

Step Rate (µ) : A larger mu leads to faster convergence but can make the filter more sensitive to noise. A smaller mu leads to slower convergence but can improve the filter's stability.



Filter Order (M): Determines the length of the filter's impulse response. A higher order filter can capture more complex noise patterns but requires more computational resources.

n=1

y(1) = 0

e(1) = 1

w(1) = [0.05, 0]

n=2

y(2) = 0.05

e(2) = 1.95

w(2) = [0.245, 0]

Practical Implications

Conclusion

Recap:

- *Developed a MATLAB GUI demonstrating ANC using adaptive filtering.*
- *Showcased the potential to reduce unwanted noise in vehicles.*

Final Thoughts:

- *ANC presents a significant opportunity to improve acoustic environments in various applications.*

Reference

How Ford is implemented active noise cancellation in cars ?
https://www.youtube.com/watch?v=Te5UUCXMSIg

Listen As Active Noise Cancellation Makes Car Interiors 90% More Silent
https://www.youtube.com/watch?v=pUDu_pyaMtQ

Innovation: Active Noise Cancellation | New Range Rover Sport
https://www.youtube.com/watch?v=uRNLIDpB4Xs

Thank You