

**AMRITA VISHWA VIDYAPEETHAM BANGALORE
CAMPUS- 560035**



Department: Electronics and Communication Engineering (ECE)

IV Semester

Electronics and Computer Engineering (EAC)

**19EAC283- Digital Signal Processing Lab Project
Report**

Submitted by

Prashant Ayitapu
(BL.EN.U4EAC21052)

Raja Karthikeya
(BL.EN.U4EAC21054)

Ramu Keerthana P
(BL.EN.U4EAC21055)

Satvik Raghav
(BL.EN.U4EAC21065)

Signature of Students:

Signature of the Faculty:

SIGNAL DENOISING USING WAVELET TRANSFORM

Aim:

Design a program for signal denoising using wavelet transform

Tool used:

MATLAB

Theory:

Signal denoising is the process of removing or reducing unwanted noise from a signal while preserving the important features and information. Noise can distort or obscure the underlying signal, making it difficult to analyze or interpret. Denoising aims to enhance the signal quality by selectively attenuating or removing the noise components.

Wavelet transform is a powerful tool for signal denoising. It decomposes a signal into different frequency components using wavelet functions, which are scaled and shifted versions of a mother wavelet.

The process of signal denoising using wavelet transform typically involves the following steps:

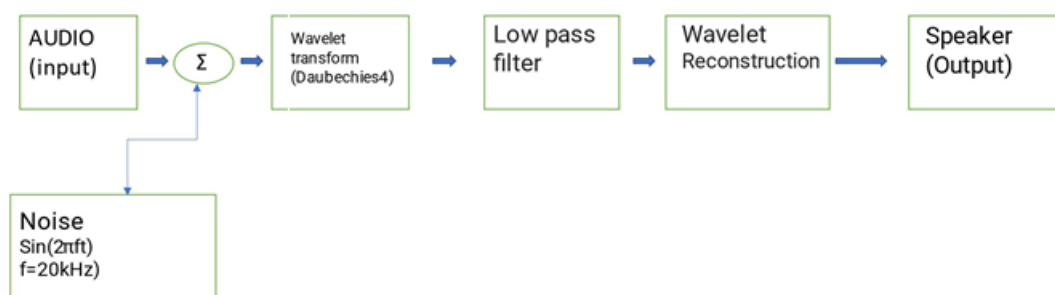
- **Wavelet Decomposition:** The original signal is decomposed into different frequency components at multiple scales using a wavelet transform, such as the Discrete Wavelet Transform (DWT). The wavelet transform decomposes the signal into approximation and detail coefficients, representing the low-frequency and high-frequency components, respectively.
- **Thresholding:** The detail coefficients obtained from the wavelet decomposition contain both signal and noise components. The idea behind signal denoising is to identify and threshold the coefficients that primarily represent noise while preserving the important signal components. Various thresholding techniques, such as soft or hard thresholding, can be employed for this purpose.
- **Coefficient Modification:** The identified noisy coefficients are modified or set to zero based on the chosen thresholding strategy. This effectively reduces the contribution of noise in the signal while preserving the significant signal components.
- **Wavelet Reconstruction:** The modified wavelet coefficients are then used to reconstruct the denoised signal using the inverse wavelet transform. This step combines the approximation coefficients (which were not modified during thresholding) and the modified detail

coefficients to obtain the denoised signal.

The effectiveness of signal denoising using wavelet transform depends on selecting appropriate wavelet basis functions, determining an optimal thresholding strategy, and considering the specific characteristics of the signal and noise. Experimentation and tuning may be necessary to achieve the desired denoising performance.

Signal denoising using wavelet transform finds applications in various fields, including audio and speech processing, biomedical signal analysis, image denoising, and many other domains where noise reduction is essential for accurate analysis or perception of the underlying signal.

Block diagram:



Code:

```
clc;
close all;
clear all;
[x, Fs] = audioread("NInput_audio.wav");

% Display the original audio signal
figure;
subplot(3, 1, 1);
plot(x);
title('Original Audio Signal');
xlabel('Samples');
ylabel('Amplitude');

% Add high-frequency noise to the audio signal
```

```

noiseFrequency = 20000;
t = (0:length(x)-1)/Fs;
noise = sin(2*pi*noiseFrequency*t);
noisyAudio= x + noise;

% Display the noisy audio signal
subplot(3, 1, 2);
plot(noisyAudio);
title('Noisy Audio Signal');
xlabel('Samples');
ylabel('Amplitude');%%done

% Apply the wavelet transform to find the high-frequency coefficients
wname = 'db4'; % Choose the desired wavelet (e.g., Daubechies 4)
level = 5; % Choose the desired decomposition level

% Perform the wavelet decomposition
[c, l] = wavedec(noisyAudio, level, wname);

% Identify the index of the high-frequency coefficients
highFreqCoeffIdx = length(x) / 2 + 1 : length(x);%approx
% Set the high-frequency coefficients to zero
c(highFreqCoeffIdx) = 0;

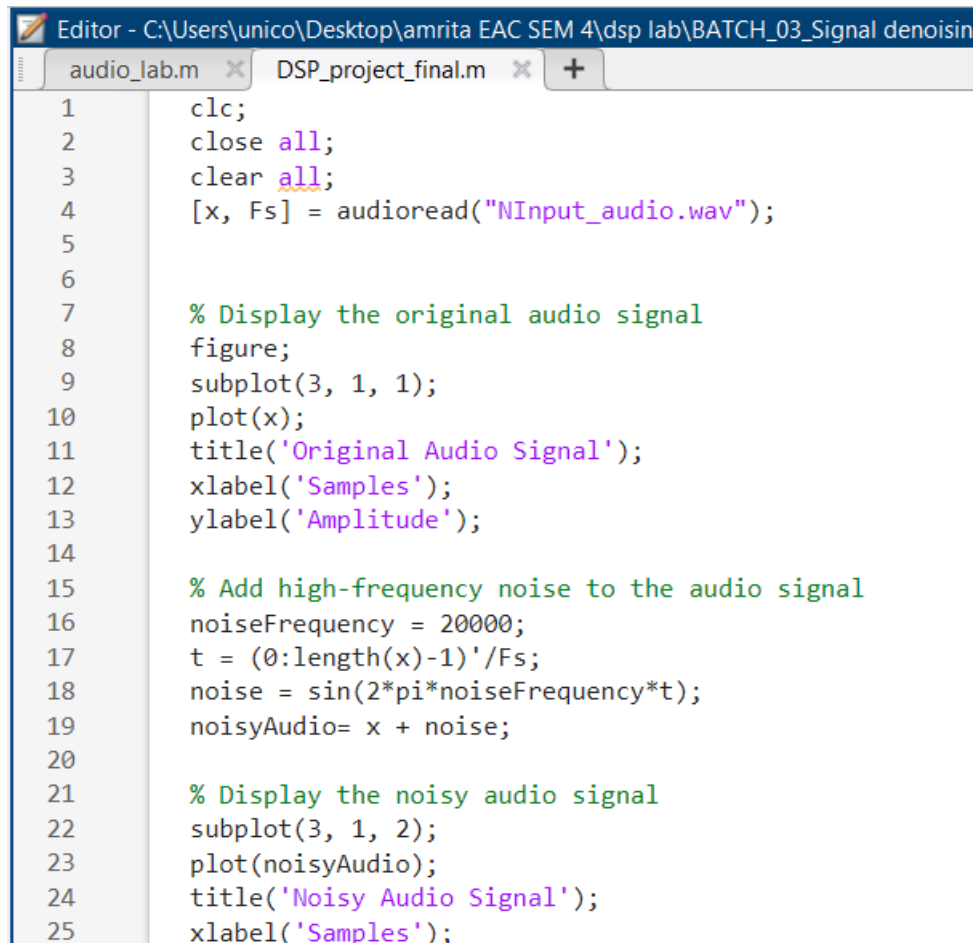
% Reconstruct the denoised audio signal
denoisedAudio = waverec(c, l, wname);

subplot(3, 1, 3);
plot(denoisedAudio);
title('Denoised Audio Signal');
xlabel('Samples');
ylabel('Amplitude');

% Play the original, noisy, and denoised audio signals
disp('Playing the original audio signal...');
sound(x, Fs);
pause(length(x)/Fs + 1);
disp('Playing the noisy audio signal...');
sound(noisyAudio, Fs);
pause(length(noisyAudio)/Fs + 1);
disp('Playing the denoised audio signal...');
sound(denoisedAudio, Fs);

```

Screenshot of the code:

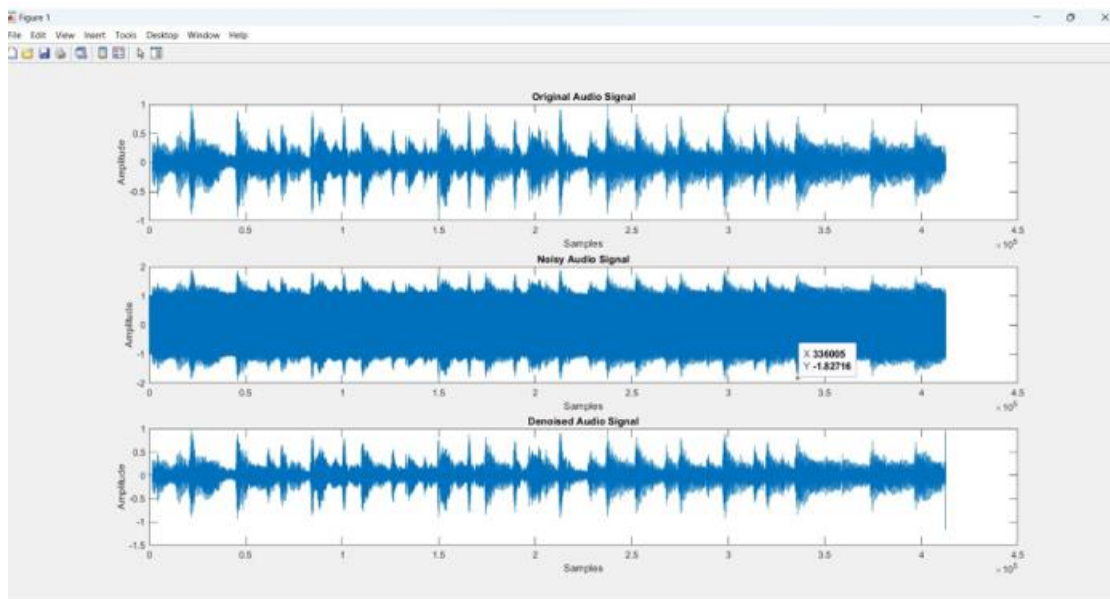


The screenshot shows a MATLAB editor window with the title bar "Editor - C:\Users\unico\Desktop\amrita EAC SEM 4\dsp lab\BATCH_03_Signal denoising". The window contains two tabs: "audio_lab.m" and "DSP_project_final.m". The code in "audio_lab.m" is as follows:

```
1  clc;
2  close all;
3  clear all;
4  [x, Fs] = audioread("NInput_audio.wav");
5
6
7  % Display the original audio signal
8  figure;
9  subplot(3, 1, 1);
10 plot(x);
11 title('Original Audio Signal');
12 xlabel('Samples');
13 ylabel('Amplitude');
14
15 % Add high-frequency noise to the audio signal
16 noiseFrequency = 20000;
17 t = (0:length(x)-1)/Fs;
18 noise = sin(2*pi*noiseFrequency*t);
19 noisyAudio= x + noise;
20
21 % Display the noisy audio signal
22 subplot(3, 1, 2);
23 plot(noisyAudio);
24 title('Noisy Audio Signal');
25 xlabel('Samples');
```

```
Editor - C:\Users\unico\Desktop\amrita EAC SEM 4\dsp lab\BATCH_03_Signal denoising using wavelet transform_DSP
audio_lab.m x DSP_project_final.m x +
25 xlabel('Samples');
26 ylabel('Amplitude');%%done
27
28
29 % Apply the wavelet transform to find the high-frequency coefficients
30 wname = 'db4'; % Choose the desired wavelet (e.g., Daubechies 4)
31 level = 5; % Choose the desired decomposition level
32
33 % Perform the wavelet decomposition
34 [c, l] = wavedec(noisyAudio, level, wname);
35
36 % Identify the index of the high-frequency coefficients
37 highFreqCoeffIdx = length(x) / 2 + 1 : length(x);%approx
38 % Set the high-frequency coefficients to zero
39 c(highFreqCoeffIdx) = 0;
40
41 % Reconstruct the denoised audio signal
42 denoisedAudio = waverec(c, l, wname);
43
44 subplot(3, 1, 3);
45 plot(denoisedAudio);
46 title('Denoised Audio Signal');
47 xlabel('Samples');
48 ylabel('Amplitude');
49
48 ylabel('Amplitude');
49
50 % Play the original, noisy, and denoised audio signals
51 disp('Playing the original audio signal...');
52 sound(x, Fs);
53 pause(length(x)/Fs + 1);
54 disp('Playing the noisy audio signal...');
55 sound(noisyAudio, Fs);
56 pause(length(noisyAudio)/Fs + 1);
57 disp('Playing the denoised audio signal...');
58 sound(denoisedAudio, Fs);
59
```

Output:



Result:

In this project we have added high frequency noise to a input signal and applied wavelet transform to obtain the high frequency and low frequency components. We have forced the high frequency components to become zero and applied inverse wavelet transform to obtain the denoised signal.

Conclusion:

Wavelet transform-based denoising offers an efficient and flexible approach for noise reduction in signals, balancing noise suppression with the preservation of essential signal features. Its ability to capture localized details and adapt to the signal's frequency content makes it a valuable tool for a wide range of applications, including audio processing, image denoising, and biomedical signal analysis.