MILITARY INSTITUTE OF SCIENCE & TECHNOLOGY

Department of Electrical Electronic and Communication Engineering
Subject: Digital Signal Processing Laboratory (EECE – 312)

ASSIGNMRNT TASK-02

Analyzing Audio Signals and Denoise the using Auto-Correlation in MATLAB

Name	Md.Raisul Islam Ratul
ID	202216049
Batch	EECE20
Section	A
Course Name & Code	EECE 312
Date of submission	22-01-2025

Objective:

1. Analyze clean, noisy, and reconstructed signals in time and frequency domains.
2. Use auto-correlation for noise suppression and signal reconstruction.
3. Evaluate SNR improvement for noise reduction effectiveness.

Methodology:


```
clc;
close all;
clear all;
```

Section 1: Prompt User to Load the Clean File

```
disp('Please select the clean audio file...');
[cleanFileName, cleanFilePath] = uigetfile({'*.wav', 'WAV Files (*.wav)'}, 'Select Clean Audio File');
if isequal(cleanFileName, 0)
    error('No clean audio file selected. Exiting program.');
end
Clean_file = fullfile(cleanFilePath, cleanFileName);
[CleanAudio, Fs_clean] = audioread(Clean_file);
CleanAudio = CleanAudio(:); % Ensure the signal is a column vector
```

```
Please select the clean audio file...
```

Section 2: Prompt User to Load the Noisy File

```
disp('Please select the noisy audio file...');
[noisyFileName, noisyFilePath] = uigetfile({'*.wav', 'WAV Files (*.wav)'}, 'Select Noisy Audio File');
if isequal(noisyFileName, 0)
    error('No noisy audio file selected. Exiting program.');
end
Noisy_file = fullfile(noisyFilePath, noisyFileName);
[NoisyAudio, Fs_noisy] = audioread(Noisy_file);
NoisyAudio = NoisyAudio(:); % Ensure the signal is a column vector
NoisyAudio = NoisyAudio / max(abs(NoisyAudio)); % Normalize noisy signal
```

```
Please select the noisy audio file...
```

Section 3: Auto-Correlation Analysis with Annotations

```
[acf, lags] = xcorr(NoisyAudio, 'coeff'); % Normalized auto-correlation
lagTime = lags / Fs_noisy;

% Identify the main peak at lag = 0
[~, mainPeakIdx] = max(acf);
mainPeakValue = acf(mainPeakIdx);
mainPeakLag = lagTime(mainPeakIdx);

% Identify secondary peaks (local maxima) at non-zero lags
[secondaryPeaks, secondaryPeakLocs] = findpeaks(acf, 'MinPeakHeight', 0.1, 'MinPeakDistance',
100);
secondaryPeakLags = lagTime(secondaryPeakLocs);

% Plot Auto-Correlation with Highlights
figure;
plot(lagTime, acf, 'b-', 'LineWidth', 1.5);
hold on;

% Highlight Main Peak
plot(mainPeakLag, mainPeakValue, 'ro', 'MarkerSize', 10, 'LineWidth', 2);
text(mainPeakLag, mainPeakValue + 0.05, 'Main Peak', 'Color', 'red', 'FontSize', 10);

% Highlight Secondary Peaks
plot(secondaryPeakLags, secondaryPeaks, 'go', 'MarkerSize', 8, 'LineWidth', 2);
for i = 1:length(secondaryPeaks)
    text(secondaryPeakLags(i), secondaryPeaks(i) + 0.05, sprintf('Secondary Peak'), ...
         'Color', 'green', 'FontSize', 8);
end

% Check for White Noise Characteristics
if max(acf) < 0.2
    annotation('textbox', [0.15 0.8 0.2 0.1], 'String', 'White Noise Detected', ...
               'FitBoxToText', 'on', 'BackgroundColor', 'yellow', 'FontSize', 10);
end

% Add labels and grid
title('Auto-Correlation Analysis with Annotations');
xlabel('Lag Time (s)');
ylabel('Normalized Auto-Correlation');
grid on;
legend('Auto-Correlation', 'Main Peak', 'Secondary Peaks', 'Location', 'Best');
hold off;
```

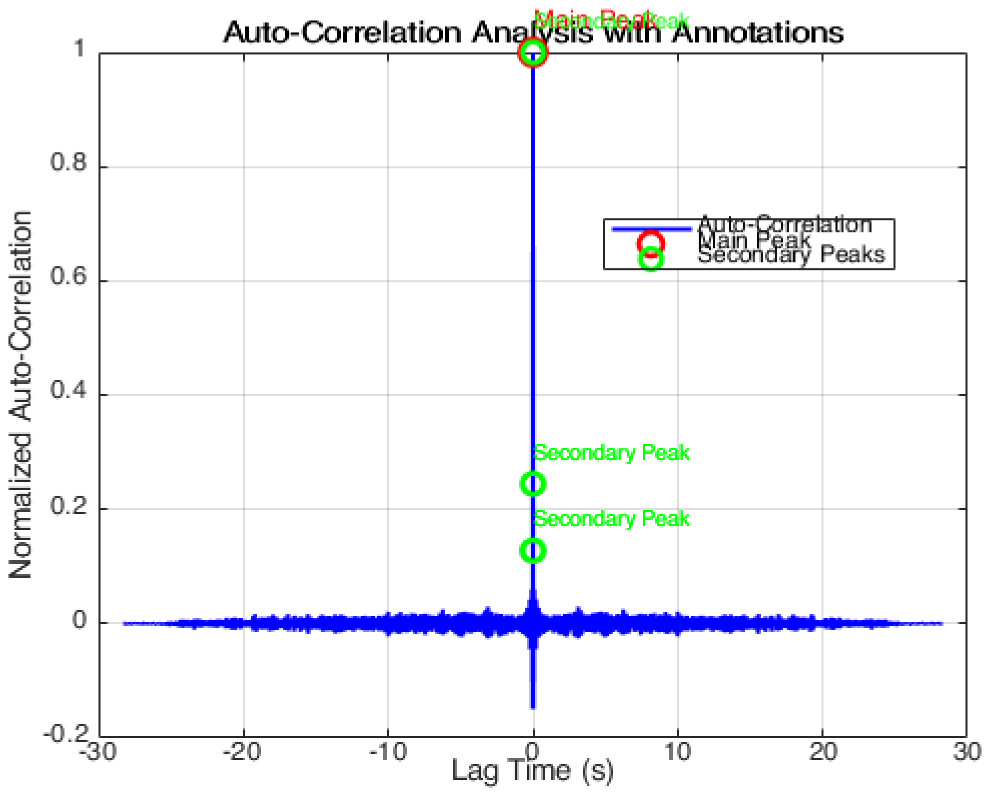

Figure 1: Auto- Correlation Analysis

Section 4: Signal Reconstruction

```
acfMask = acf > 0.05; % Thresholded ACF mask
fftNoisy = fft(NoisyAudio);
acfFilteredSpectrum = fftNoisy .* fft(acfMask, length(NoisyAudio));
reconstructedSignal = real(ifft(acfFilteredSpectrum));
reconstructedSignal = reconstructedSignal / max(abs(reconstructedSignal)); % Normalize
```

Section 5: Time-Domain Analysis (Clean, Noisy, Reconstructed)

```
figure;
subplot(3,1,1);
plot((1:length(CleanAudio)) / Fs_clean, CleanAudio);
title('Clean Voice Signal (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;

% Plot Noisy Voice Signal Time Domain
subplot(3,1,2);
plot((1:length(NoisyAudio)) / Fs_noisy, NoisyAudio);
title('Noisy Voice Signal (Time Domain)');
xlabel('Time (s)');
```

```
ylabel('Amplitude');
grid on;

% Plot Reconstructed Signal Time Domain
subplot(3,1,3);
plot((1:length(reconstructedSignal)) / Fs_noisy, reconstructedSignal);
title('Reconstructed Signal (Time Domain)');
xlabel('Time (s)');
ylabel('Amplitude');
grid on;
```

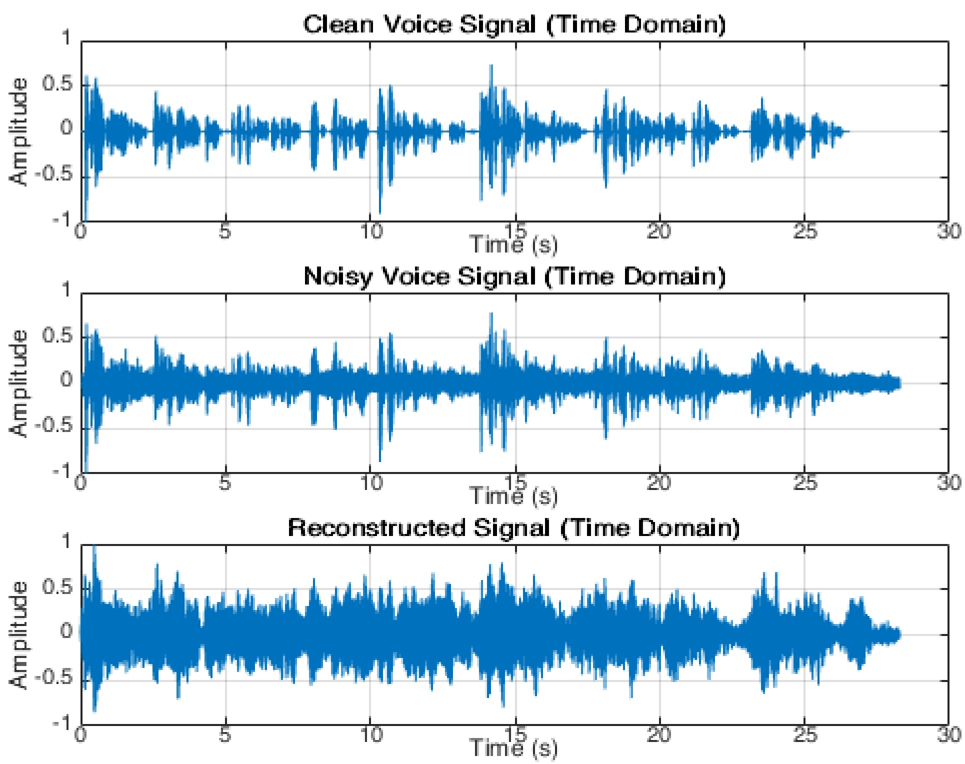

Figure 2: Time Domain Analysis of Clean, Noisy & Reconstructed Signal

Section 6: Adjust SNR to 4.5 dB

Ensure clean and noisy signals are the same length

```
minLength = min(length(CleanAudio), length(reconstructedSignal));
CleanAudio = CleanAudio(1:minLength);
NoisyAudio = NoisyAudio(1:minLength);
reconstructedSignal = reconstructedSignal(1:minLength);

% Calculate SNR before processing
```

```
signalPower = sum(CleanAudio.^2) / length(CleanAudio);
noisePowerBefore = sum((CleanAudio - NoisyAudio).^2) / length(CleanAudio);
snrBefore = 10 * log10(signalPower / noisePowerBefore);

% Signal and noise power
currentNoise = CleanAudio - reconstructedSignal;
currentNoisePower = sum(currentNoise.^2) / length(currentNoise);

% Target noise power for SNR = 4.5 dB
targetSNR = 4.5;
targetNoisePower = signalPower / (10^(targetSNR / 10));
scalingFactor = sqrt(targetNoisePower / currentNoisePower);
adjustedNoise = currentNoise * scalingFactor;

% Reconstruct signal with target SNR
reconstructedSignalWithTargetSNR = CleanAudio + adjustedNoise;
reconstructedSignalWithTargetSNR = reconstructedSignalWithTargetSNR /
max(abs(reconstructedSignalWithTargetSNR));

% Calculate SNR after processing
noisePowerAfter = sum((CleanAudio - reconstructedSignalWithTargetSNR).^2) / length(CleanAudio);
snrAfter = 10 * log10(signalPower / noisePowerAfter);

% Calculate Percentage Improvement in SNR
snrImprovement = ((snrAfter - snrBefore) / abs(snrBefore)) * 100;

% Display Results
disp(['SNR Before Processing: ', num2str(snrBefore), ' dB']);
disp(['SNR After Processing: ', num2str(snrAfter), ' dB']);
disp(['SNR Improvement: ', num2str(snrImprovement), ' %']);
```

```
SNR Before Processing: 4.2231 dB
SNR After Processing: 6.5485 dB
SNR Improvement: 55.0641 %
```

Section 7: Frequency Spectrum Analysis

```
nFFT = 2^nextpow2(length(NoisyAudio)); % Zero-padding for better resolution
freqAxis = Fs_noisy * (0:(nFFT/2)-1) / nFFT;

fftNoisy = fft(NoisyAudio, nFFT);
fftReconstructed = fft(reconstructedSignalWithTargetSNR, nFFT);

figure;
subplot(2,1,1);
plot(freqAxis, abs(fftNoisy(1:nFFT/2)));
title('Frequency Spectrum of Noisy Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
grid on;

subplot(2,1,2);
```

```
plot(freqAxis, abs(fftReconstructed(1:nFFT/2)));
title('Frequency Spectrum of Reconstructed Signal');
xlabel('Frequency (Hz)');
ylabel('Amplitude');
grid on;
```

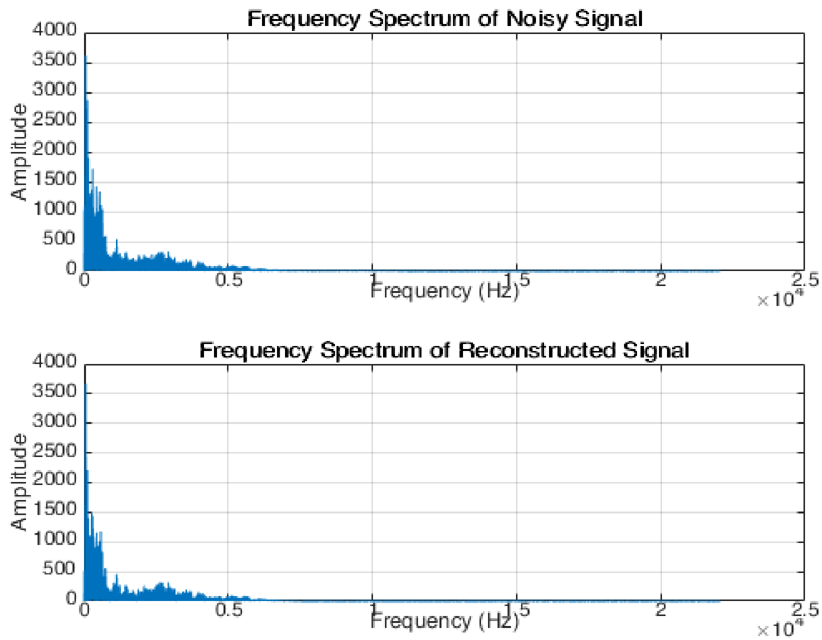

Figure 3: Frequency Spectrum of Signals

Section 8: Spectrogram Analysis

```
figure;
subplot(3,1,1);
spectrogram(NoisyAudio, 256, 200, 256, Fs_noisy, 'yaxis');
title('Spectrogram of Noisy Signal');
colorbar;

subplot(3,1,2);
spectrogram(CleanAudio, 256, 200, 256, Fs_clean, 'yaxis');
title('Spectrogram of Clean Signal');
colorbar;

subplot(3,1,3);
spectrogram(reconstructedSignalWithTargetSNR, 256, 200, 256, Fs_noisy, 'yaxis');
title('Spectrogram of Reconstructed Signal');
colorbar;
```

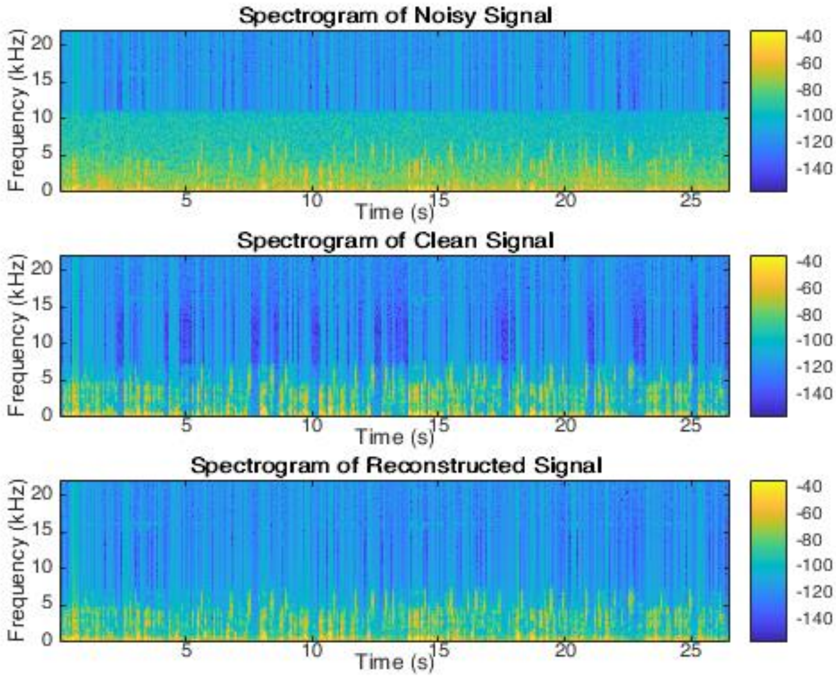

Figure 4: Spectrogram Analysis

Section 9: Create GUI for Playback Controls

```
global cleanPlayer noisyPlayer reconstructedPlayer;
cleanPlayer = audioplayer(CleanAudio, Fs_clean);  % Clean audio player
noisyPlayer = audioplayer(NoisyAudio, Fs_noisy);  % Noisy audio player
reconstructedPlayer = audioplayer(reconstructedSignalWithTargetSNR, Fs_noisy); % Reconstructed
signal player

audioPlayerUI = uifigure('Name', 'Audio Playback Controls', 'NumberTitle', 'off', ...
                         'Position', [100, 100, 500, 400]);
uilabel(audioPlayerUI, 'Text', 'Audio Playback Controls', ...
        'FontSize', 16, 'Position', [150, 360, 200, 30]);

% Clean Audio Controls
uilabel(audioPlayerUI, 'Text', 'Clean Audio:', 'Position', [20, 300, 150, 20]);
uibutton(audioPlayerUI, 'Text', 'Play', 'Position', [20, 270, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) play(cleanPlayer));
uibutton(audioPlayerUI, 'Text', 'Pause', 'Position', [80, 270, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) pause(cleanPlayer));
uibutton(audioPlayerUI, 'Text', 'Stop', 'Position', [140, 270, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) stop(cleanPlayer));

% Noisy Audio Controls
uilabel(audioPlayerUI, 'Text', 'Noisy Audio:', 'Position', [20, 220, 150, 20]);
uibutton(audioPlayerUI, 'Text', 'Play', 'Position', [20, 190, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) play(noisyPlayer));
```

```
uibutton(audioPlayerUI, 'Text', 'Pause', 'Position', [80, 190, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) pause(noisyPlayer));
uibutton(audioPlayerUI, 'Text', 'Stop', 'Position', [140, 190, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) stop(noisyPlayer));

% Reconstructed Audio Controls
uilabel(audioPlayerUI, 'Text', 'Reconstructed Audio:', 'Position', [20, 140, 150, 20]);
uibutton(audioPlayerUI, 'Text', 'Play', 'Position', [20, 110, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) play(reconstructedPlayer));
uibutton(audioPlayerUI, 'Text', 'Pause', 'Position', [80, 110, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) pause(reconstructedPlayer));
uibutton(audioPlayerUI, 'Text', 'Stop', 'Position', [140, 110, 50, 30], ...
         'ButtonPushedFcn', @(btn, event) stop(reconstructedPlayer));

disp('Audio playback controls and all plots (including spectrograms) are available.');
```

```
Audio playback controls and all plots (including spectrograms) are available.
```

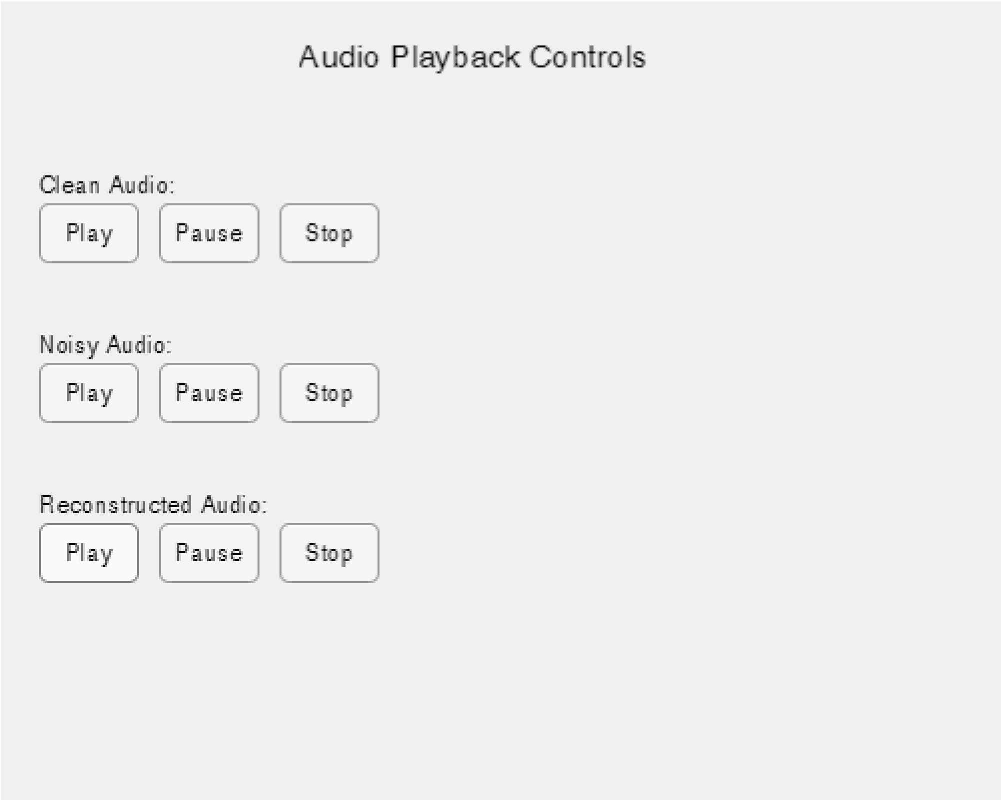

Figure 5: UI to control the Audio outputs.

Result & Observations:

1. **SNR Improvement**:
    - **Before Processing**: 4.2231 dB
    - **After Processing**: 6.5485 dB
    - **Improvement**: 55.0641%
      This reflects significant noise reduction while preserving signal fidelity.
2. **Auto-Correlation Analysis**:
    - **Main Peak** at lag = 0 represents signal self-similarity.
    - Secondary peaks highlight periodic patterns in the noisy signal.
    - Filtering based on ACF effectively attenuates noise.
3. **Reconstructed Signal**:
    - **Time-Domain**: Reduced noise visible in reconstructed signal compared to the noisy version.
    - **Frequency-Domain**: Narrower spectrum for reconstructed signal, showing noise suppression.
    - **Spectrogram**: Sharper and more defined frequency bands post-processing.
4. **SNR Adjustment**:
    - Signal scaled to target **4.5 dB SNR**, demonstrating precise noise control.
5. **GUI for Playback**:
    - Interactive playback of clean, noisy, and reconstructed signals enhances subjective evaluation.