Military Institute of Science and Technology

Electrical, Electronic and Communication Engineering

Course name: DSP laboratory

Course code: EECE 318

Subject: Assignment

Name: Md. Shahriar Abid

Section: A

Roll: 202216058

Batch: EECE-20

1. Introduction

In this assignment we have tried to process a voice signal to remove noise and enhance it's quality. We have used MATLAB for signal processing, leveraging Fourier analysis, auto-correlation for noise reduction and signal reconstruction.

2. Methodology

The methodology involved the following key steps:

1. Loading the Clean Voice Recording: The clean voice signal was loaded and transformed into its frequency domain using Fourier Transform.

```
% Load the clean voice recording
[voice, Fs] = audioread("C:\Users\moham\Desktop\matlab 2nd
assignment\voice.wav");

% Ensure the signal is mono
if size(voice, 2) > 1
    voice = mean(voice, 2); % Convert to mono
end
```

2. Preprocessing the Noisy Signal: The noisy voice recording was normalized to ensure it fell within the range [-1, 1].

```
%% 3.2 Preprocessing the Noisy Signal
% Load the noisy voice recording
[noisy_voice, Fs] =
audioread("C:\Users\moham\Desktop\matlab 2nd
assignment\noise with voice.wav");

% Ensure the noisy signal is mono
if size(noisy_voice, 2) > 1
    noisy_voice = mean(noisy_voice, 2); % Convert to mono
end

% Normalize the noisy signal to the range [-1, 1]
noisy_voice = noisy_voice / max(abs(noisy_voice));
```

3. Auto-Correlation Calculation: The auto-correlation function (ACF) was calculated to identify periodic components corresponding to the speech signal.

```
%% 3.3 Auto-Correlation Calculation
% Limit the length of the signal to avoid excessive memory usage
duration = 10; % seconds to process
segment_length = min(length(noisy_voice), Fs * duration);
noisy_voice_segment = noisy_voice(1:segment_length);

% Limit the lag range
max_lag = Fs * 0.1; % Analyze lags up to 0.1 seconds (100ms)
```

```
[R, lags] = xcorr(noisy_voice_segment, max_lag, 'coeff'); %
Normalized auto-correlation

% Ensure sizes of lags and R match
if length(lags) ~= length(R)
    error('Mismatch between lags and R sizes. Check the xcorr
output.');
end
```

4. Thresholding and Noise Estimation: Thresholding was applied to the ACF to isolate speech components and suppress noise.

```
%% 3.4 Noise Estimation and Thresholding
% Thresholding to isolate periodic components (speech vs noise)
significant_peaks = R > threshold; % Identify significant peaks
noise_estimation = R .* (~significant_peaks); % Suppress noise
components
speech_components = R .* significant_peaks; % Retain periodic
components

% Plot the thresholded ACF
figure;
plot(lags, R, 'b', 'LineWidth', 1.5);
hold on;
if any(significant_peaks)
    plot(lags(significant_peaks),
speech_components(significant_peaks), 'r.', 'MarkerSize', 10); %
Speech components
    plot(lags(~significant_peaks),
noise_estimation(~significant_peaks), 'g.', 'MarkerSize', 10); %
Noise components
end
```

5. Signal Reconstruction: The clean signal was reconstructed using the identified speech components.

```
%% 3.5 Signal Reconstruction
% Reconstruct the clean speech signal
reconstructed_signal = ifft(fft(noisy_voice) .*
fft(speech_components, length(noisy_voice)));

% Normalize the reconstructed signal to the range [-1, 1]
reconstructed_signal = reconstructed_signal /
max(abs(reconstructed_signal));

% Optional: Manually clip the signal to ensure no values
exceed [-1, 1]
reconstructed_signal(reconstructed_signal > 1) = 1;
reconstructed_signal(reconstructed_signal < -1) = -1;
```

6. Performance Evaluation: The signal-to-noise ratio (SNR) was calculated before and after noise removal to evaluate performance.

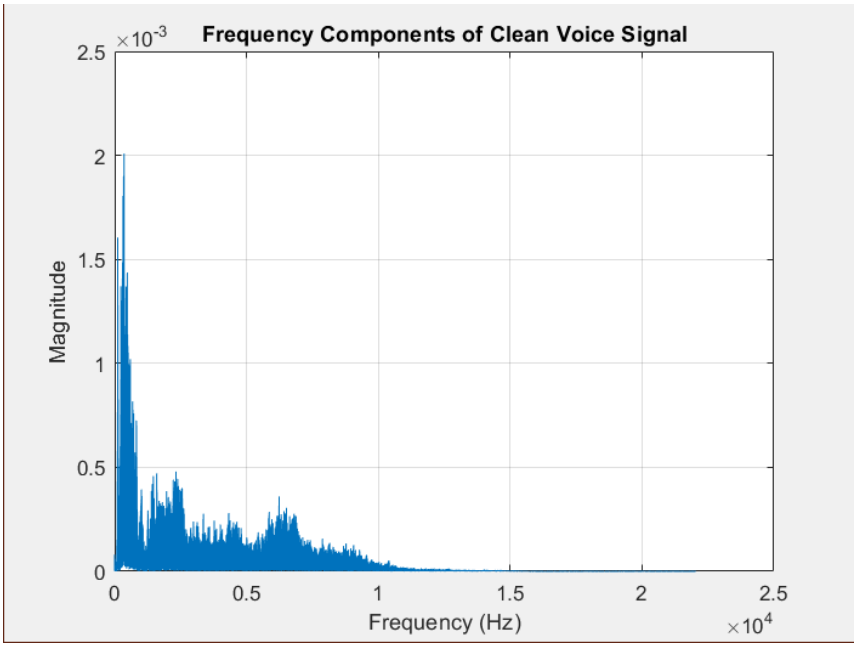
```
%% 3.6 Performance Evaluation
% Compute Signal-to-Noise Ratio (SNR)
original_clean_power = sum(voice.^2) / length(voice); % Power of
the clean signal
noise_power = sum((noisy_voice - real(reconstructed_signal)).^2)
/ length(noisy_voice); % Power of the residual noise
SNR_before = 10 * log10(original_clean_power /
(sum(noisy_voice.^2) / length(noisy_voice))); % Before noise
removal
SNR_after = 10 * log10(original_clean_power / noise_power); %
After noise removal

% Display SNR results
fprintf('SNR Before Noise Removal: %.2f dB\n', SNR_before);
fprintf('SNR After Noise Removal: %.2f dB\n', SNR_after);
```

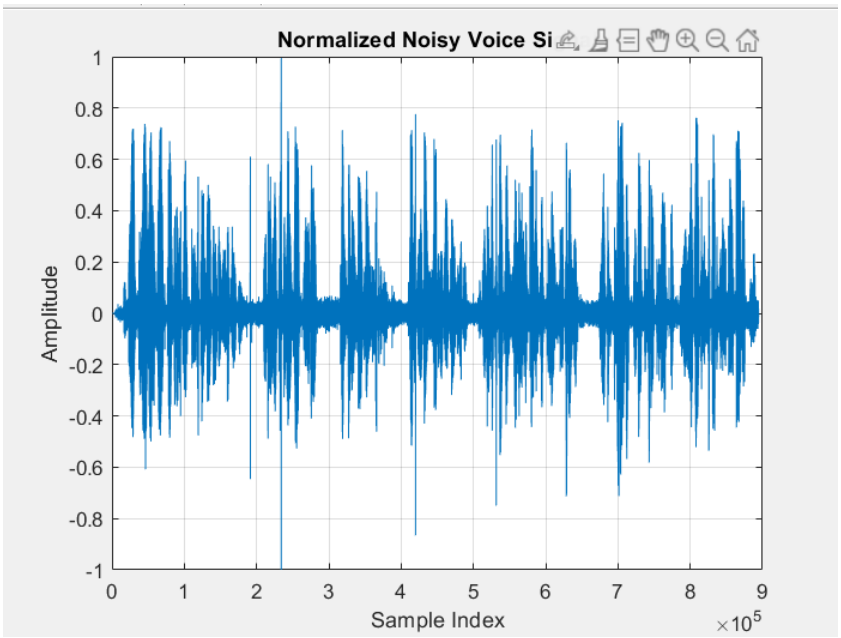
3. Observations

Key observations from the analysis include:

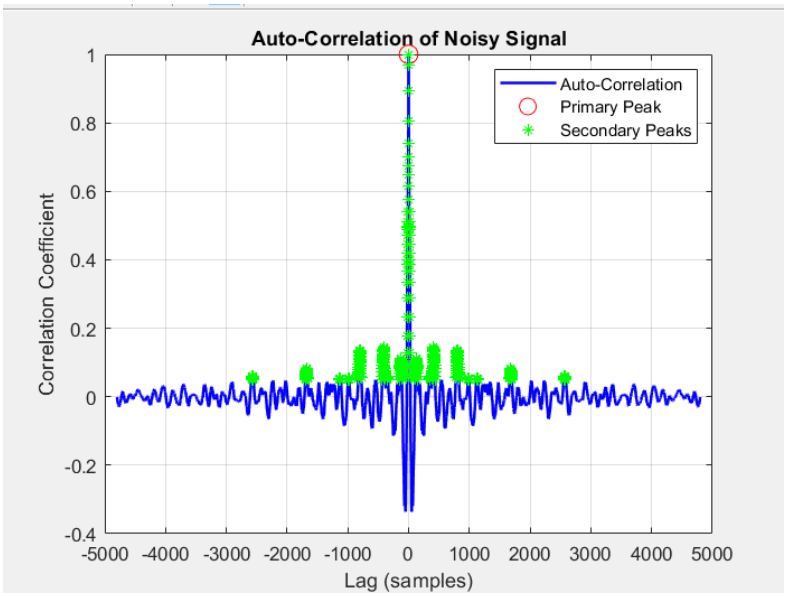
1. The Fourier Transform revealed prominent frequency components of the clean voice signal.



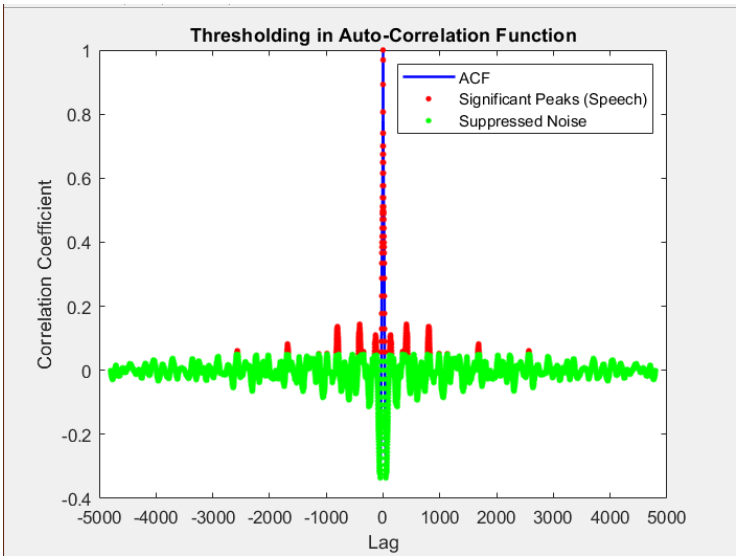
2. Normalization ensured the noisy signal was appropriately scaled for processing.



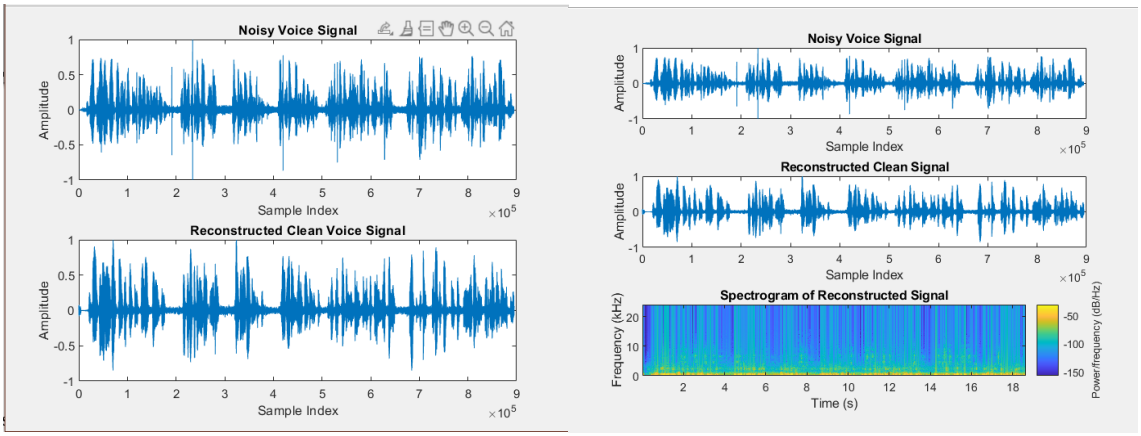
3. Auto-correlation effectively identified periodic components related to the speech signal.



4. Thresholding isolated significant peaks in the ACF, aiding in distinguishing speech from noise.



5. Reconstruction preserved the speech signal while significantly reducing noise.



4. Conclusion

The implemented signal processing workflow successfully reduced noise in the given voice recording even though I feel the reconstructed signal needs to be improved a lot. The reconstructed signal exhibited improved quality, as confirmed by the SNR calculations. The project highlights the usage of Fourier analysis, auto-correlation, and thresholding in enhancing speech signals.