"Heaven's Light is Our Guide"

Rajshahi University of Engineering & Technology

Department of Computer Science & Engineering

Assignment

Course Code : CSE 3210

Course Title : Digital Signal Processing Sessional

Submitted By-	Submitted To-
Name: Sajidur Rahman Tarafder	Md. Farukuzzaman Faruk
Department : CSE	Assistant Professor
Roll No : 2003154	Department of CSE, RUET
Section : C	
Session:2020-21	

Introduction:

Digital Signal Processing (DSP) is a vital tool for transforming and analyzing real-world signals like audio. In this assignment, we use a convolution-based low-pass filter to reduce high-frequency noise in an audio file. We demonstrate the effectiveness of a moving average filter using Python, Librosa, and SciPy by filtering a noisy **.**wav file and visualizing the outcome.

Motivation:

Low-pass filtering is foundational in DSP and has broad applications:

- Noise reduction in audio and communication systems
- Smoothing and signal shaping in audio editing tools
- Practical understanding of DSP through real-time implementation

This task reinforces the theoretical knowledge of filtering using hands-on coding and waveform analysis.

Background Study:

Convolution in Signal Processing:

Convolution is used in signal processing to apply transformations like filtering. In audio processing, it modifies the input signal by convolving it with a filter kernel to alter frequency characteristics.

Low-Pass Filters (LPF):

An LPF allows low-frequency components to pass while suppressing high-frequency noise. A moving average LPF averages neighboring sample values, which smoothens rapid changes and attenuates noise.

Social and Economic Impact:

Societal Benefits:

- **Hearing aids:** Enhance speech clarity
- **Music production:** Noise suppression and quality improvement
- **Telecommunications:** Improved voice clarity

Economic Importance:

- **Entertainment:** Clean audio improves user experience in music, games, and films
- **VoIP:** Better signal quality enhances communication
- **Medical tech:** Filters in hearing devices rely on DSP

Related Mathematical Studies:

Discrete-Time Convolution:

The convolution of discrete-time signals is represented as:

$$y[n] = \sum_{\{k=-\infty\}}^{\{\infty\}} x[k] \cdot h[n - k]$$

Where **x[n]** is the input, h[n] is the impulse response, and y[n] is the output.

Moving Average Filter:

A basic LPF is defined as:

$$h[n] = \frac{1}{N} \quad , if\ 0 \le n \le N$$

$$= 0\ , otherwise$$

This filter suppresses rapid variations and highlights the overall trend of the signal.

Theory:

Working procedure of the filter:

- Each output sample is the average of n = 201 neighboring samples.
- This smoothens the waveform, reducing sharp transitions.
- The longer the filter, the stronger the noise suppression.

Expected Outcomes:

- **Time domain:** Smooth waveform, lower sharpness
- **Auditory:** Less hiss and harshness, more muffled tone

Implementation:

Tools Used:

- Python 3
- Librosa for loading audio
- SciPy for convolution
- Matplotlib for visualization
- Jupyter/Colab for execution

Steps Performed:

1. Loaded "Signal.wav" and trimmed it to 10 seconds.

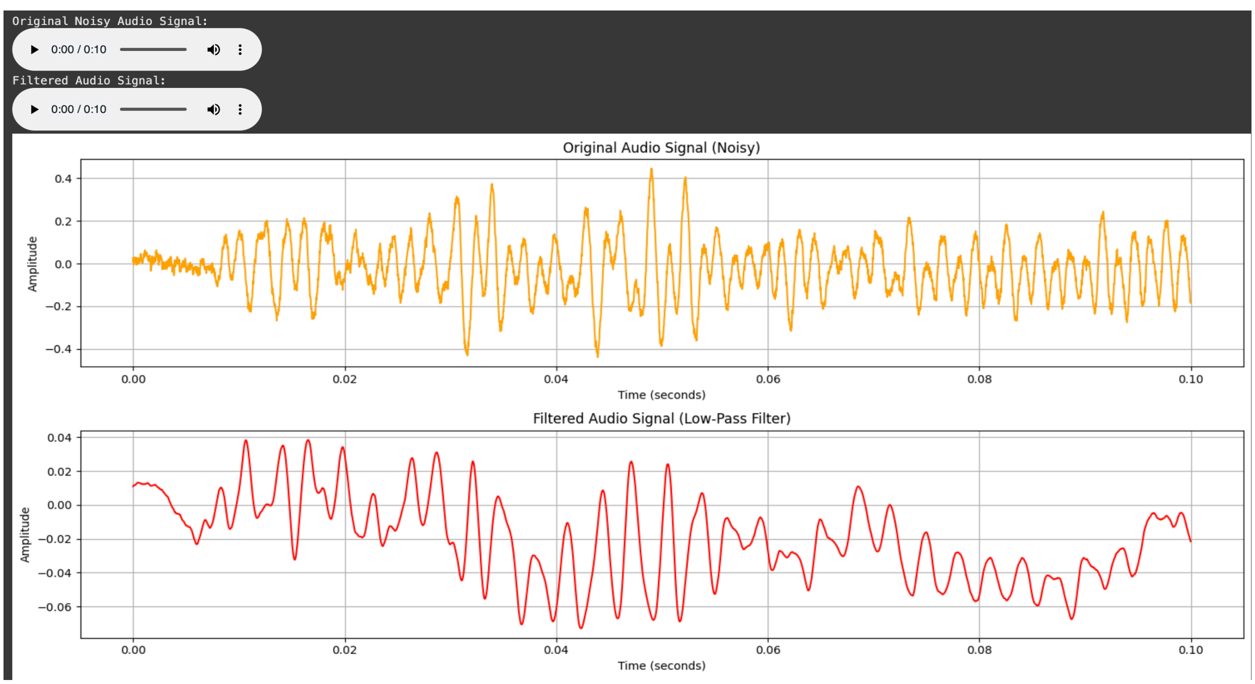
2. Defined a low-pass filter with kernel size 201.
3. Applied convolution to the signal using mode='same'.
4. Played both original and filtered signals.
5. Plotted waveform comparison.

Code:

```
import librosa
import numpy as np
import matplotlib.pyplot as plt
from scipy.signal import convolve
from IPython.display import Audio

# Signal
audio_data, sampling_rate = librosa.load("Signal.wav",
sr=None)

duration_sec = 10
audio_data = audio_data[:sampling_rate * duration_sec]

# Low-Pass Filter
n = 201
Filter = np.ones(n) / n

Output_Signal = convolve(audio_data, Filter, mode='same')

Time = np.linspace(0, len(audio_data) / sampling_rate,
len(audio_data))

print("Original Noisy Audio Signal:")
display(Audio(data=audio_data, rate=sampling_rate))

print("Filtered Audio Signal:")
display(Audio(data=Output_Signal, rate=sampling_rate))

# Plotting Signals
plt.figure(figsize=(15, 7))
```

```
# Original Signal
plt.subplot(2, 1, 1)
plt.plot(Time[:sampling_rate // 10],
audio_data[:sampling_rate // 10], color='orange')
plt.title("Original Audio Signal (Noisy)")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.grid(True)

# Filtered Signal
plt.subplot(2, 1, 2)
plt.plot(Time[:sampling_rate // 10],
Output_Signal[:sampling_rate // 10], color='red')
plt.title("Filtered Audio Signal (Low-Pass Filter)")
plt.xlabel("Time (seconds)")
plt.ylabel("Amplitude")
plt.grid(True)

plt.tight_layout()
plt.show()
```

Output:



Conclusion:

This experiment demonstrated:

- Implementation of a simple LPF using convolution
- Effective reduction of high-frequency noise in real audio
- Clear auditory and visual improvements
- Strong connection between DSP theory and practice