AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY
School of Engineering

TERM WORK PROJECT PROPOSAL

# AUDIO DENOISING

19CCE450 – WAVELETS AND APPLICATION

Group-11

TEAM GROUP MEMBERS


1. DHASHYANTH N – CB.EN.U4CCE20018
2. HEMCHAND R – CB.EN.U4CCE20024

## PROBLEM STATEMENT:

The challenge lies in developing robust and efficient audio denoising algorithms that effectively remove background noise while preserving the integrity and natural characteristics of the original audio signal. Minimizing artifacts, distortion, and loss of relevant information, and optimizing for real-time applications are crucial considerations for enhancing the listening experience.

## OBJECTIVES OF THE WORK:

1. Remove background noise from audio recordings.
2. Improve the signal-to-noise ratio in audio files.
3. Preserve and enhance the clarity and intelligibility of speech in noisy audio.
4. Minimize artifacts and distortions introduced during the denoising process.
5. Maintain the overall quality and fidelity of the original audio content.
6. Enhance the listening experience by reducing unwanted noise distractions.
7. Enable better analysis and understanding of audio data by reducing noise interference.
8. Provide real-time or near-real-time denoising capabilities for live audio applications.
9. Support denoising across various audio formats and platforms.
10. Develop efficient and scalable algorithms for audio denoising.

## SOFTWARES REQUIRED FOR SIMULATION:

Jupiter Notebook

GNU Radio

## METHODOLOGY:

### Methodology for Audio Denoising in Python:

1. Preprocessing: Load the audio file and convert it into a suitable format for processing. Common formats include WAV or MP3.

2. Spectral Analysis: Apply a Fourier transform to convert the audio from the time domain to the frequency domain. This allows you to analyze the different frequency components of the audio signal.

3. Noise Estimation: Estimate the noise profile by analyzing a section of the audio that contains only noise. This can be done by taking a portion of the audio with no speech or desired signal present.

4. Spectral Subtraction: Subtract the estimated noise profile from the frequency spectrum of the original audio. This removes the noise components from the audio signal.

5. Inverse Fourier Transform: Apply the inverse Fourier transform to convert the denoised audio back to the time domain.

6. Post-processing: Apply any additional filters or techniques to further enhance the denoised audio quality. This may include adaptive filtering, wavelet denoising, or nonlinear filtering.

7. Output: Save the denoised audio to a file or stream it for real-time applications.

**Methodology for Audio Denoising in GNU Radio:**

1. Design the Flowgraph: Create a flowgraph using GNU Radio Companion (GRC) or write the flowgraph in GNU Radio's graphical language (GRC-L).

2. Audio Source: Add an audio source block to read the input audio file or audio stream.

3. Signal Processing Blocks: Use appropriate signal processing blocks such as FFT, filter banks, or spectral subtraction blocks to perform denoising operations on the audio signal.

4. Noise Estimation: Incorporate blocks to estimate the noise profile using a noise-only portion of the audio.

5. Denoising Algorithm: Implement denoising algorithms such as spectral subtraction, adaptive filtering, or wavelet denoising within the flowgraph.

6. Audio Sink: Include an audio sink block to output the denoised audio to a file or play it in real-time.

7. Parameters and Tuning: Adjust the parameters of the denoising blocks to optimize the denoising process based on the specific audio characteristics and noise conditions.

8. Execution: Execute the flowgraph to perform the denoising operation on the input audio.

9. Evaluation: Assess the denoised audio quality by listening to the output and comparing it with the original audio. Make adjustments as necessary.

10. Optimization: Fine-tune the flowgraph and denoising parameters to achieve the desired audio denoising performance.

**DELIVERABLES:**

Audio denoising using Python and GNU Radio can yield impressive results in improving the quality of audio recordings. By applying techniques such as spectral subtraction, adaptive filtering, and noise estimation, both Python and GNU Radio offer effective means to reduce background noise and enhance speech intelligibility. These methodologies allow for real-time denoising and support various audio formats. Python provides flexibility for algorithm development and post-processing, while GNU Radio offers a visual programming interface for building signal processing flowgraphs. With careful parameter tuning and optimization, the denoised audio output can exhibit reduced noise interference, improved clarity, and an overall enhanced listening experience.

**ALGORITHM:**

The first step is importing the libraries that we are going to use. These libraries that we import are installed using the pip command in the terminal as shown below:

pip install scipy

pip install numpy

pip install skimage

pip install matplotlib

We import the wavfile from the scipy.io to read audio signals in the .wav format.

We then read the audio file using the wavfile.read() function and normalize the amplitude as shown below:

```
Fs, x = wavfile. read('guitar.wav')
```

The signal samples will be stored in the x variable and the sampling frequency to the Fs variable.

The amplitude is normalized because wavfile reads the audio in int16 mode. This mode gives large values. Therefore, we need to reduce these values through normalization.

Since the audio signal has no noise, we add noise

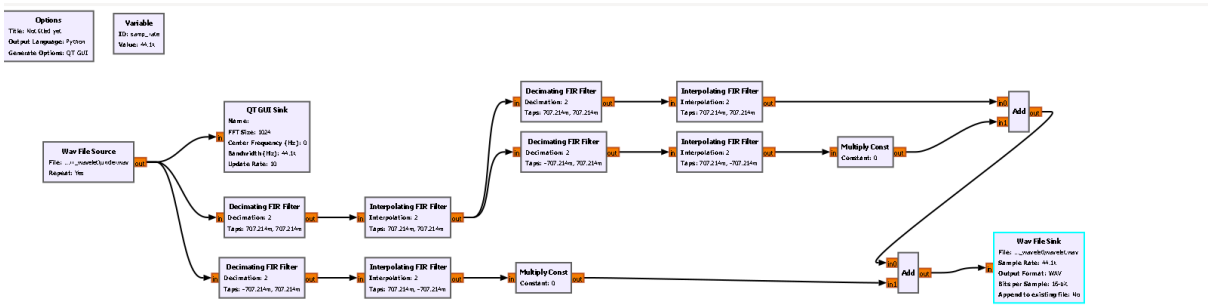
```
x_noisy = x + sigma * np.random.randn(x.size)
```

After adding the random noise, we denoise the signal using the denoise_wavelet() function.

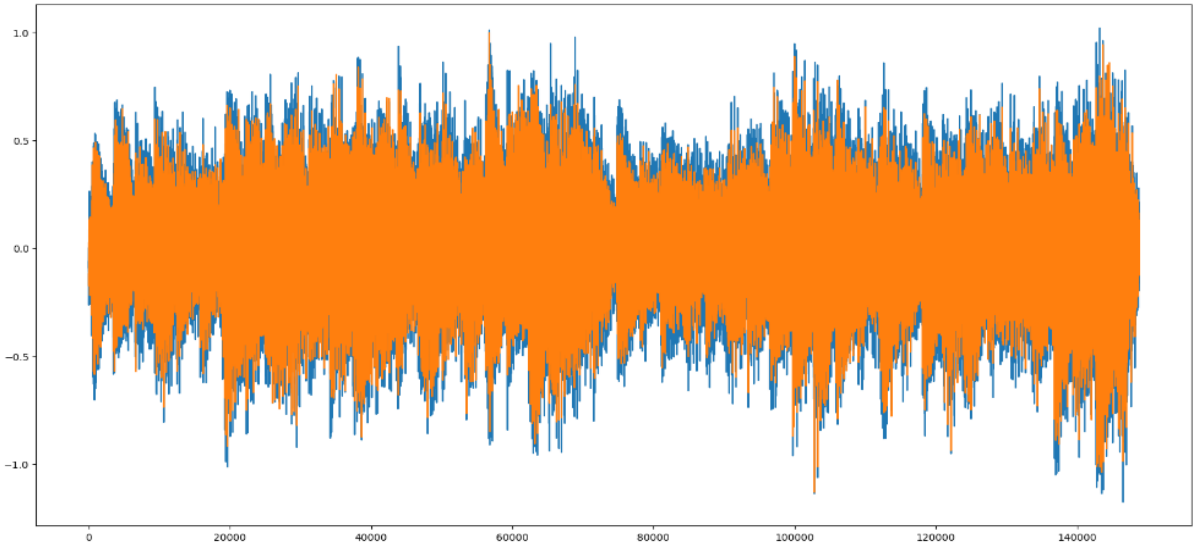
```
x_denoise = denoise_wavelet(x_noisy, method='VisuShrink',
mode='soft', wavelet_levels=3, wavelet='sym8',
rescale_sigma='True')
```

we use the VishuShrink method. For visualization, we plot the output.

## BLOCK DIAGRAM (GNU RADIO):




## OUTPUT:




## CONCLUSION:

In conclusion, both Python and GNU Radio provide powerful tools and methodologies for audio denoising. Using Python, denoising algorithms can be developed and customized for specific applications, while GNU Radio offers a graphical programming interface for designing signal processing flowgraphs. By leveraging techniques such as spectral subtraction, adaptive filtering, and noise estimation, these platforms can effectively

reduce background noise, enhance speech intelligibility, and improve the overall quality of audio recordings. With careful parameter tuning and optimization, audio denoising using Python and GNU Radio can produce impressive results, leading to a more enjoyable listening experience with reduced noise interference.

**REFERENCES:**


1. Xu, Yong, and Jun Du. "A Review of Sparse Representation-Based Techniques for Speech Enhancement." IEEE/CAA Journal of Automatica Sinica 4.2 (2017): 198-207.

2. Tan, Zheng-Hua, and DeLiang Wang. "An overview of noise-robust automatic speech recognition." IEEE/ACM Transactions on Audio, Speech, and Language Processing 22.4 (2014): 745-777.

3. Williamson, David S., and DeLiang Wang. "Complex ratio masking for monaural speech separation." IEEE Transactions on Audio, Speech, and Language Processing 16.4 (2008): 766-778.

4. Cohen, Israel, et al. "Speech enhancement for non-stationary noise environments." Speech Communication 51.6 (2009): 450-466.