LIVE AUDIO FILTERING

Ayush Kumar
180174
ayushkmr@iitk.ac.in

Dept of Electrical Engg, Indian Institute of Technology Kanpur

ABSTRACT

Audio filtering is essential part of many digital audio and speech applications. This project is mainly about using different digital filters (like lowpass, highpass etc) to filter live audio.

1. INTRODUCTION

Audio filtering is very essential for audio and speech processing applications. In this project we aim to implement common filters along with giving user the option to design custom filters using lccde equations or pole-zero expressions to design rational filters. These filters can help in removing white noise (low pass filtering). The code has written in python using pyaudio for audio streams. The code is modular to encourage readability.

2. METHODOLOGY

2.1. Recording Audio

We use `pyaudio` to create a read stream with sampling rate of 44100 samples per second. We then read 4410 samples from the stream at a time to get a chunk equivalent to 1 second of audio. Filters are applied on these samples in the next step.

2.2. Designing Filter

Some common filters like low pass, high pass and moving average filters have been implemented. Other than these custom filters using lccde coefficients and pole-zero based rational filters can be designed. This has been implemented using symbolic expressions (`sympy`).

2.3. Applying Audio Filters

Applying a filter is done by convolving the filter with audio samples which is equivalent to multiplication in frequency domain. This is acheived by first taking the DFT of signal (using `numpy.fft.fft`), then multiplying it with the filter and then taking an inverse DFT.
$x[n] \longleftrightarrow X[k]$

$Y[k] = H[k] \times X[k]$

$Y[k] \longleftrightarrow y[n]$

The filter is designed by using lccde equation to create a rational filter H(z). This is then used to compute H($e^{j\omega}$). Then we compute H[k] by (sampling from this) substituting $\omega = 2 * \pi * k/N$, in H($e^{j\omega}$) (where is N is length of DFT). After computing H[k] we multiply it with DFT of signal and take inverse DFT of the result (Y[k]). The final result is then scaled back to [0,255](this is required to play it back with the output audio stream) and written into the output stream.

3. FILTERS

3.1. Low Pass Filter

Given the cutoff frequency fc.
$H(\omega) = \{1 \;\; if(\omega < \omega_c)$ 0 if (othw) $\}$

3.2. High Pass Filter

Given the cutoff frequency fc.
$H(\omega) = \{1 \;\; if(\omega > \omega_c)$ 0 if (othw) $\}$

3.3. LCCDE Coefficients

Given the numerator coefficients b and denominator coefficients a.

$$\text{H(z)} = \frac{b[0] + b[1] \times z^{-1} .... b[n] * z^{-n}}{a[0] + a[1] \times z^{-1} .... a[m] * z^{-m}}$$

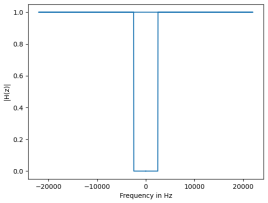
3.4. Pole Zero

Given the poles and zeros

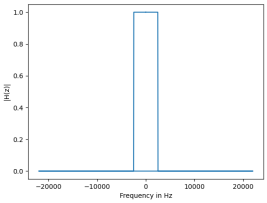
$$\text{H(z)} = \frac{(1 - z^{-1} b[0]) \times (1 - z^{-1} b[1]) \times ...}{(1 - z^{-1} \times a[0]) \times (1 - z^{-1} a[1]) \times ...}$$

4. RESULTS

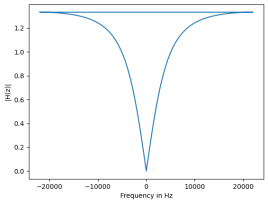
The frequency responses for the filters are as follows:



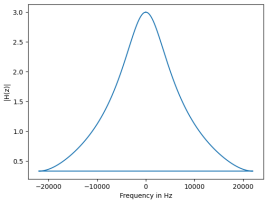
(a) Highpass



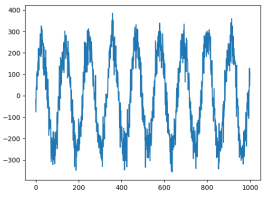
(b) Lowpass



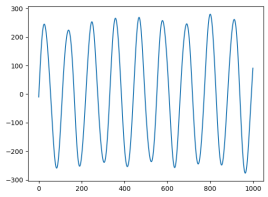
(c) LCCDE High pass filter



(d) LCCDE High pass filter



(a) Sinusoid with noise



(b) Lowpass filtered sinusoid

Fig. 1. Filters and their responses

5. ACKNOWLEDGEMENT

I would like to then Prof Vipul Arora for giving me a chance to work on this project. This project helped me understand the basics of digital filtering. This also gave me experience of practical aspects of filtering.

6. REFERENCES

[1] Stéfan van der Walt, S Chris Colbert, and Gaël Varoquaux, "The numpy array: A structure for efficient numerical computation," *Computing in Science Engineering*, vol. 13, no. 2, pp. 22–30, Mar 2011.

[2] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[3] Vipul Arora, "Ee301 lectures," 2021.

[4] Hubert Pham, "Pyaudio," 2006.

[1] [2] [3] [4]