REAL TIME AUDIO FILTERING

Sakib Malik

180652
Electrical Engineering
Indian Institute of technology, Kanpur

ABSTRACT

Audio filtering have many applications in real life. In this project we aim to implement a real time audio filtering platform using python. In which the user can decide the type of filter according to their application and filter their voice in real time. It can be used to filter out high frequency noise (using a low pass filter) or may be to extract out a particular frequency component from an audio piece.

1. INTRODUCTION

Audio filters are an important part of the basic building block of an audio system, they can amplify or attenuate a range of frequencies from the audio input.

1.1. Types of filters

Low-pass
Low-pass filters pass through frequencies below their cutoff frequencies, and progressively attenuates frequencies above the cutoff frequency.
High-pass
A high-pass filter does the opposite, passing high frequencies above the cutoff frequency, and progressively attenuating frequencies below the cutoff frequency
Band-pass/reject
A bandpass filter passes frequencies between its two cutoff frequencies, while attenuating those outside the range. A band-reject filter attenuates frequencies between its two cutoff frequencies, while passing those outside the 'reject' range.
All-pass
An all-pass filter passes all frequencies, but affects the phase of any given sinusoidal component according to its frequency.

1.2. Digital filter theory

An LTI filter is represented by $h[n]$, where n is the sample number and $h[n]$ is the value of that sample. It is more useful to consider it's z-transform i.e the transer function H(z).

If the input is represented as $x[n]$ and output of the filter by $y[n]$ and their z-transforms by X(z) and Y(z) then we have, $H(z) = Y(z)/X(z)$ which is a polynomial in z. Which can be represented as

$$H(z) = \frac{b[0] + b[1]z^{-1} + ... + b[M]z^{-M}}{a[0] + a[1]z^{-1} + ... + a[N]z^{-N}}$$

Hence a filter can be uniquely represented using the numerator coefficients (given by the list b) and denominator coefficients (given by the list a).

To implement the filter we first take the input from the user which is the coefficients of the filter $H(z)$. We then Calculated it's DFT, by calculating $H(e^{jw})$ for $w = 2 * \pi * k/N$, where $k \in (0, .., N-1)$, where N is the number of samples in input $x[n]$.

2. WORKING

In this section, We describe the working of real-time audio filtering, as we have learned in the theory various discrete filters such as IIR or FIR can be used for filtering, and also they can be customized to work as Low-pass, high-pass, band-pass, or band-stop. The voice signals are digital (sampled) and hence we can apply digital filters to them to filter unwanted frequency components out of them, And achieve a noise-free filtered output.

We have provided the calculation of DFT of $H[n]$ denoted by $H[k]$ in section 1.2. We now provide how to calculate the filtered output $y[n]$.

we know that,

$$Y[k] = H[k] \times X[k]$$
$$y[n] = \text{IDFT}(X[k] * Y[k])$$

where $F[K]$ is the DFT of $F[n]$ for any F. and IDFT is inverse discrete fourier transform

we can calculate the DFT of $x[n]$ i.e $X[k]$ using `np.fft.fft` function provided by the numpy library in python. We then multiply $X[k]$ and $H[k]$. take its IDFT using `np.fft.ifft` to get $y[n]$.

During the execution these are the steps that are performed -

- User provides the numerator and denominator coefficients for the transfer function of the filter $H(z)$.
- User can hear the filtered output simultaneously with the recording (in real-time) or later on using the saved `output.wav` file.
- You can choose the amount of time for record and playback.
- Make sure to place the speaker away from the mic to avoid reverbing of the voice or use earphones.
- Can use the low-pass, high-pass, band-pass, band-stop filters according to the appropriate transfer function provided by the user for custom filtering.
- Use `play.py` (to play `.wav` file using its filename)

3. IMPLEMENTATION DETAILS

The input voice is recorded using `pyaudio`, it is sampled at 44100 samples/sec. The input voice is saved in the file `input.wav` and the filtered output is saved in `output.wav`.

The input to the program consists of :

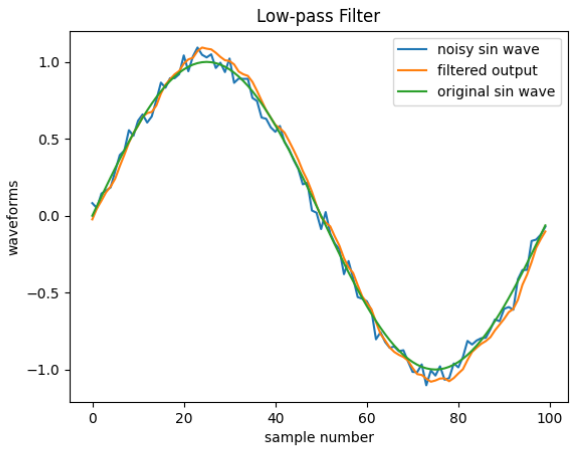
1. Coefficients for H(Z) i.e b, a lists.
2. Whether you wanna play the filtered output along with the input voice or just save the filtered output to `output.wav`.
3. The time for recording voice.

The output of the program consists of:

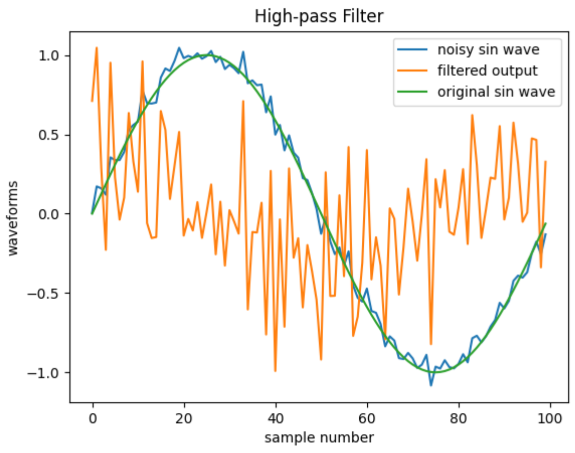
1. The filtered voice played along with the input voice if the answer to (2) above was a `y`.
2. `Input.wav` which is the recording of your input voice.
3. `output.wav` which is the filtered output.

4. RESULTS

In this section we demonstrate the use of this system on a sin wave. we first produce a $x[n] = sin(wn)$ where $w = 2*\pi*f$ and $f = 1$ with number of samples = 100. We also produce a noisy version of x by adding Gaussian noise with 0 mean and 0.05 std.



(a) Using LP (Low-pass) filter



(b) Using HP (high-pass) filter

Fig. 1. Example of using LP and HP filters on noisy sin wave.

We then use our filters on the noisy sin wave as input.

The filtered that we have used comes from a family of filters given by

$$H(z) = \frac{(1-\alpha)}{2}\frac{(1+z^{-1})}{(1-\alpha z^{-1})} \qquad \textbf{LP}$$

$$H(z) = \frac{(1-\alpha)}{2}\frac{(1-z^{-1})}{(1-\alpha z^{-1})} \qquad \textbf{HP}$$

$\alpha = 0.5$ in our expriments.

We verify that the low pass filter produces a smoother output removing high frequency noise from the noisy input. Where as the high-pass filter does the opposite, it removes the low frequency signal and retains the high frequency noise.

5. ACKNOWLEDGMENT

I would like to thanks prof. Vipul Arora for giving us the opportunity to work on this project which helped us to have a better understanding of digital signal processing, and also the TA's to help us regarding the queries related to this project.

6. REFERENCES

[1] Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 0.1038/s41586-020-2649-2.

[2] PyAudio : Python Bindings for PortAudio. Copyright (c) 2006 Hubert Pham

[3] Oppenheim, Alan V., Alan S. Willsky, and Ian T. Young. Signals and Systems. Englewood Cliffs, N.J.: Prentice-Hall, 1983.

[4] Hunter, J.D., 2007. Matplotlib: A 2D graphics environment. Computing in science and engineering, 9(3), pp.90–95.

[5] Lecture slides, EE301A Digital signal processing by prof. vipul arora, IIT Kanpur