REAL-TIME AUDIO FILTERING

Rahul Choudhary

EE, IIT Kanpur

ABSTRACT

This is a report for a course project named "Real-time Audio Filtering". The project is based on simple online audio processing techniques. I made a GUI using Tkinter in Python that is used to design filters. It takes input from microphone and gives output to speaker in real time. The user can specify which type of audio filter to apply on the input voice. The user can plot the characteristics (i.e. impulse response, magnitude response, phase response, and pole-zero plot) of the designed filter. Users are also able to see the plot of input sound and output sound along with their Frequency domain response in real time in a separate window.

***Index Terms*—** Introduction, The GUI, Filters, Plotting the filter characteristics, Real time Streaming

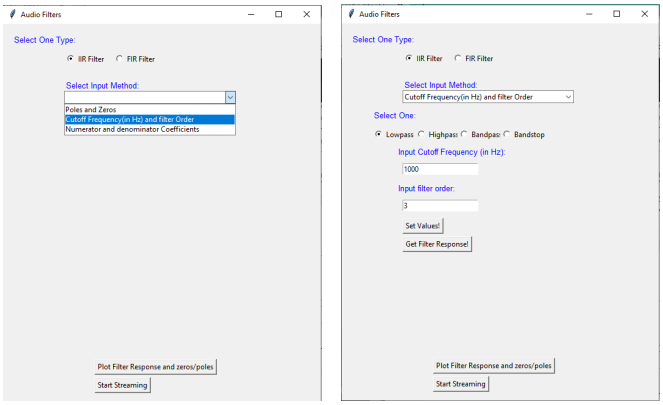
1. INTRODUCTION

I will be describing my project in the following sections along with an example on using the GUI to filter the audio in real time. This GUI uses the input from the user to design the requested filter. The GUI is made using the **tkinter** package in python. The filters are made using the **scipy** library. The audio I/O is done using the **PyAudio** that provides python binding for **PortAudio**, the cross-platform audio I/O library.

2. THE GUI

I made an interactive and dynamic GUI that asks user the preferences to design filter. It first asks the type of filter the user want to design, i.e. IIR of FIR. Then after selecting any one of these, we get a drop-down menu that ask for the input method the user want to give. It provides 3 options for IIR filter and 2 options for FIR filter. After selecting the desired method of input, the user is asked for the input values to design the filter. Once the values are set we can design the filter by clicking on the *"Get filter response"* button.

To show an example, say we want a 3rd order lowpass IIR filter with cutoff frequency 1000 Hz. So first select IIR from the two options. Then select the input method as *"Cutoff frequency (in Hz) and filter order"* as show in Fig.1(a). After this, select the *lowpass* option from four options as *lowpass, highpass, bandpass, and bandstop.* Then the user will input the required cutoff frequency and filter order. Then simply click on *"Set Values"* and then *"Get Filter Response!"* buttons. See Fig.1(b). The filter is designed!



(a) (b)

Fig. 1.

3. FILTERS

The filters are designed based on the user input using $scipy.signal$ module. For filters, I used $scipy.signal.freqz()$ or $scipy.signal.freq_zpk()$ function to get the filter response. The output of these functions is in the form (**H, w**) where **H** is the frequency response of the filter and **w** contains the corresponding frequencies (in rad/s). After that it convert the frequency response to impulse response by simply taking the Inverse FFT of **H**. It also calculate the zeros and poles and phase/magnitude response, for plotting purpose, while calculating the filter response.

4. PLOTTING THE FILTER CHARACTERISTICS

After we have designed our desired filter, we can also plot the filter characteristics to visualize our filter response. The plotting can simply be done by clicking the *"Plot Filter Response and zeros/poles"* button on the GUI. It plot the Filter impulse response, Phase response, Magnitude response, and Poles and Zeros in a separate window.

The plot generated for our designed example filter is shown in Fig.2.

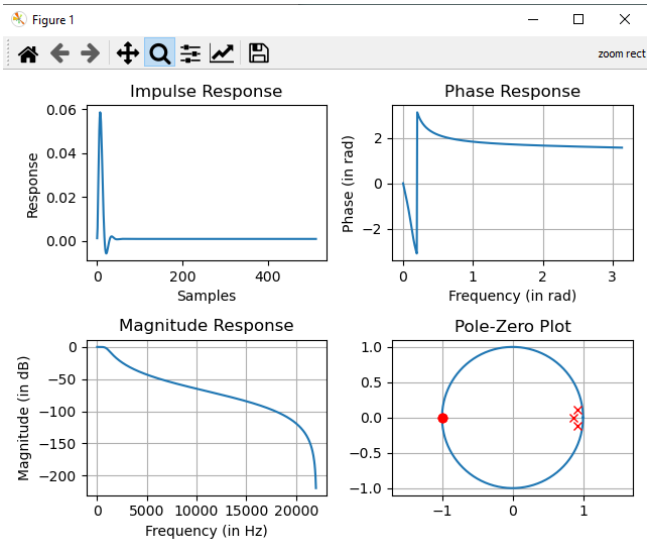


Fig. 2.

5. REAL TIME STREAMING

This is the part where the actual filtering of our voice is done. Once we are satisfied with the filter characteristics, we can move to real time filtering and streaming of our own voice. The input/output part of the process is done using the PyAudio python bindings for PortAudio. It takes input from our microphone, convert the signal to bytes. After this, we convert the data extracted to integer numpy array for further processing. We then have the input signal and the filter impulse response. We get the filtered signal simply by convolving the two signals. Now we have the input signal and the output signal. We calculate their FFTs for visualization purpose.

We plot the input and output signals along with their FFTs in a separate window so that we can visualize the frequencies of our voice and what is the response of the applied filter on our voice. An instance of this is shown in Fig.3. It shows that our lowpass filter is suppressing the high frequency noise.

Fig. 3.

6. REFERENCES

1. EE301 (Digital Signal Processing) lectures, Dr. Vipul Arora
2. Stackoverflow (https://stackoverflow.com/)
3. Scipy open source Python library (https://www.scipy.org/)
4. Tkinter Python binding to Tk GUI toolkit (https://docs.python.org/3/library/tkinter.html)
5. PyAudio python binding to PortAudio (https://people.csail.mit.edu/hubert/pyaudio/)