FILTERS FOR AUDIO PROCESSING

Pulkit Gopalani
180564
gpulkit@iitk.ac.in

Deptt. of Electrical Engineering, IIT Kanpur

ABSTRACT

We attempt to filter audio signals using simple Frequency domain based filters, e.g. Low-pass, High-pass, band-pass and so on. We employ ideal filters for attempting to filter noise using these filters on static noisy mixtures of sinusoids, pre-recorded audio and real-time audio. We also use filters in the frequency domain defined by Linear Constant Coefficient Differential Equations (LCCDE) coefficients, and by poles and zeros (for a rational filter). We also demonstrate white noise filtering using low-pass and Gaussian filters.

1. BACKGROUND

Filtering signals is usually achieved by convolving with a predefined filter. According to the Convolution theorem, convolving in the time domain is equivalent to point-wise multiplication in the Frequency domain and vice-versa.

Formally, Convolution theorem states:

$$x[n] \star h[n] \longleftrightarrow \mathrm{X(z)} \cdot \mathrm{H(z)}$$

where $\mathrm{X(z)}, \mathrm{H(z)}$ are the Fourier transforms of the respective time domain signals $x[n], h[n]$.

Hence, we take the Fourier transform of input signals, and process them using filters defined over the range of frequencies of the input $x[n]$, and eventually convert them back to the time domain.

A lot of filters can be therefore easily obtained in the frequency domain to remove (or retain) unwanted frequencies in the input signal.

The filters used in this work are as follows:

1. Low-Pass filter:

$$|H(f)| = \begin{cases} 1 & |f| < f_0 \\ 0 & |f| \geq f_0 \end{cases}$$

2. High-Pass filter:

$$|H(f)| = \begin{cases} 1 & |f| > f_0 \\ 0 & |f| \leq f_0 \end{cases}$$

3. Band-Pass filter:

$$|H(f)| = \begin{cases} 1 & f_l < |f| < f_h \\ 0 & \text{otherwise} \end{cases}$$

4. Band-Stop filter:

$$|H(f)| = \begin{cases} 0 & f_l < |f| < f_h \\ 1 & \text{otherwise} \end{cases}$$

5. Gaussian filter: (for given variance σ^2)

$$|H(f)| = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{f^2}{2\sigma^2}\right) \quad f \in R$$

6. Rational filter (for given coefficients or poles-zeros pairs):

$$|H(z)| = \frac{\prod_{n=1}^{N}(z - z_n)}{\prod_{m=1}^{M}(z - p_m)}$$

We have used ideal filters, noting however that practical realization may not be possible to such an extent. The plots for the magnitude response of all such filters are appended on the next page for reference. Cutoff frequency is 5000 Hz for a) and b), lower cutoff is 4000 Hz and 6000 Hz for c) and d), the standard deviation is 1500 Hz for e) and the poles, zeros for f) are : poles : $(1.0, 2.0)$, zeros : $(3.0, 4.0)$.

Additive White Gaussian Noise (AWGN) has a Fourier transform which is constant (C) for all values of frequencies i.e.

$$\mathrm{AWGN}(t) \longleftrightarrow C \quad \forall f \in [-\infty, \infty]$$

Therefore this noise can be suitably removed, if the actual signal frequency band is localised, by applying a suitable band filter.

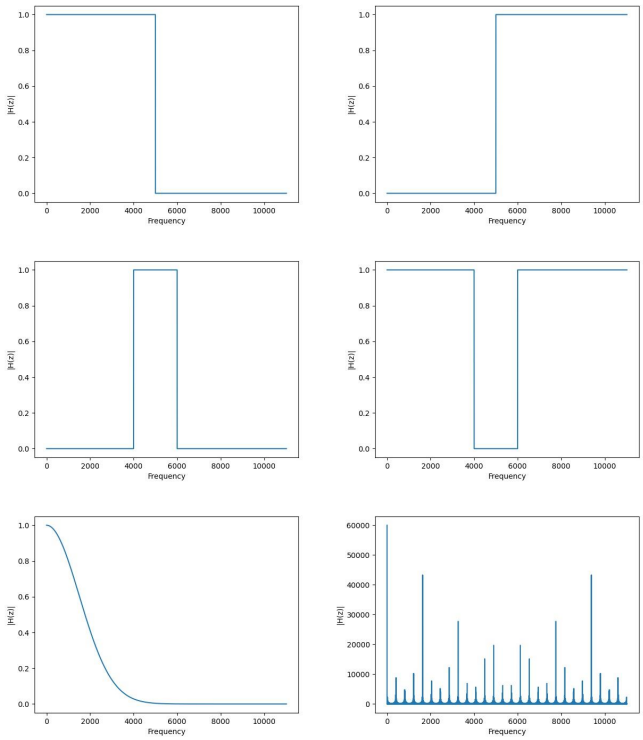


Fig. 1. Filters used in order: a), b), c), d), e), f).

2. IMPLEMENTATION

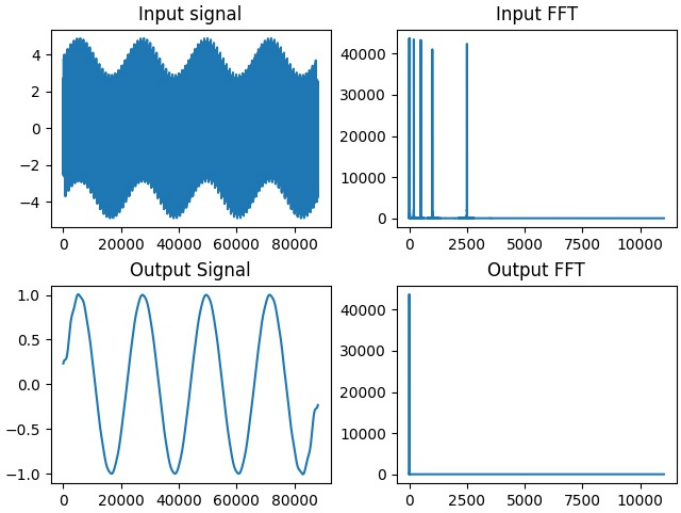
We implement all filters using Numpy. The Fourier transforms are computed using `numpy.fft.fft` utility in Numpy [1], and the inverse Fourier transform is obtained using `numpy.fft.ifft`. The polynomials for the rational transform function were realized using `numpy.poly1d`. All plots are done using Matplotlib [2]. Sound processing was done using PyAudio [3] and wave library in Python. The audio used for experiments is single-channel, with a sampling frequency 22050 fps. We use a `chunk` size of 1024 to process input frames. (except for live audio, where `chunk` size = 2^{16}.) We also acknowledge the instructor's lectures for this work. [4]

3. RESULTS

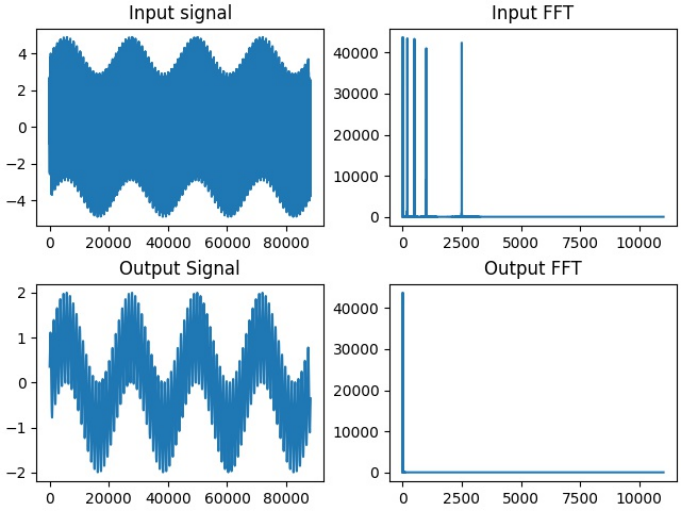
We start with the analysis of static sinusoid signals. (input frequencies are $(1.0, 20.0, 200.0, 500.0, 1000.0, 2500.0)$ Hz.

We also experimented with white noise, results are appended in the adjacent column (bottom-most frame).

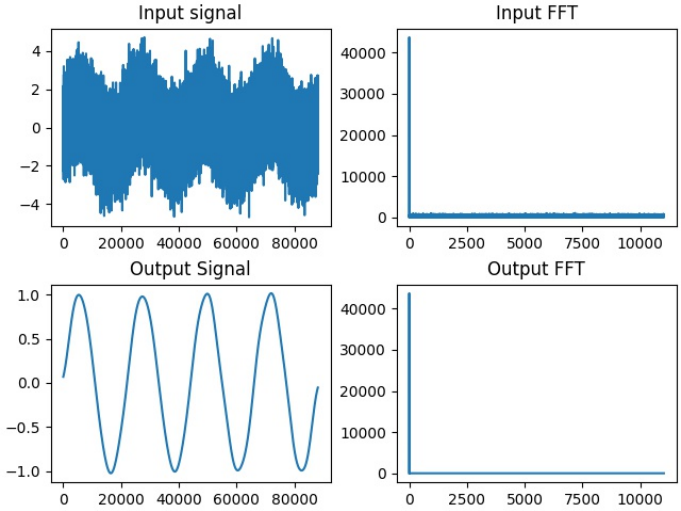
Results for pre-recorded audio (available here) and live audio are given on the following pages, in the left and right column respectively.



(a) Low-pass filter, cutoff: 10 Hz

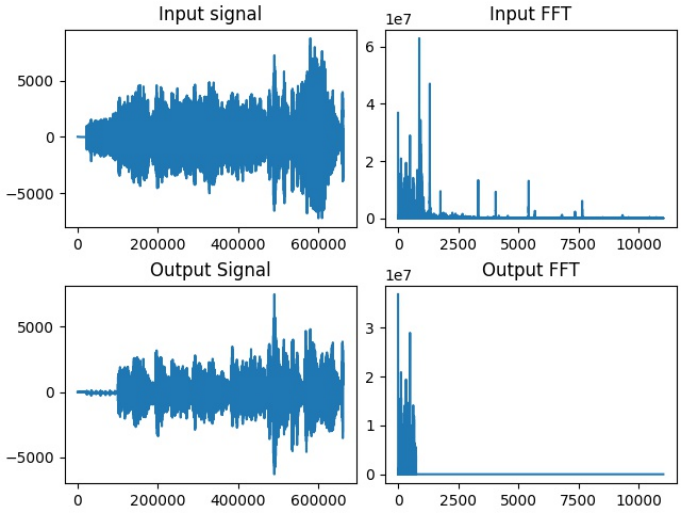


(b) Low-pass filter, cutoff: 100 Hz

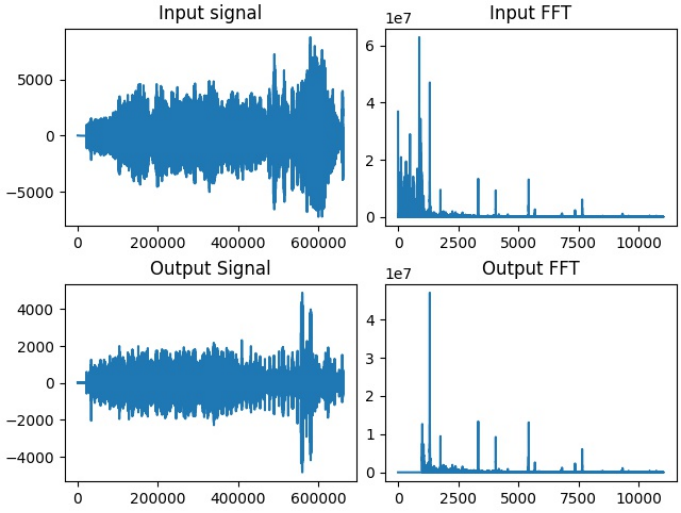


(c) AWGN + sinusoid input, Low-pass filter, cutoff: 5 Hz

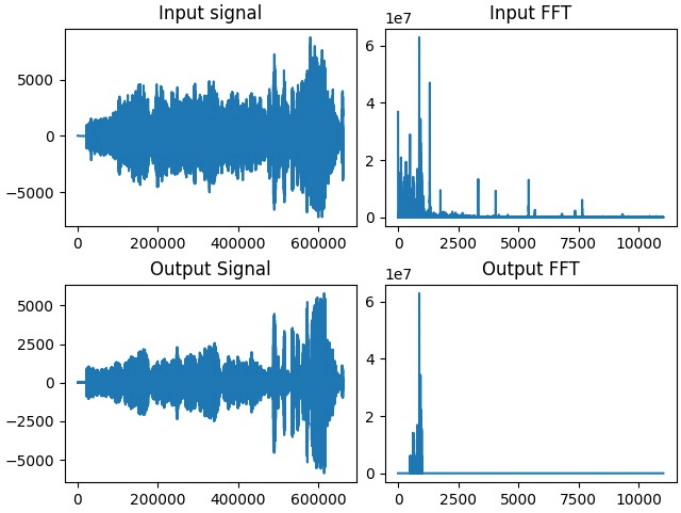
Fig. 2. Static sinusoid analysis



(a) Low-pass filter, cutoff: 750 Hz

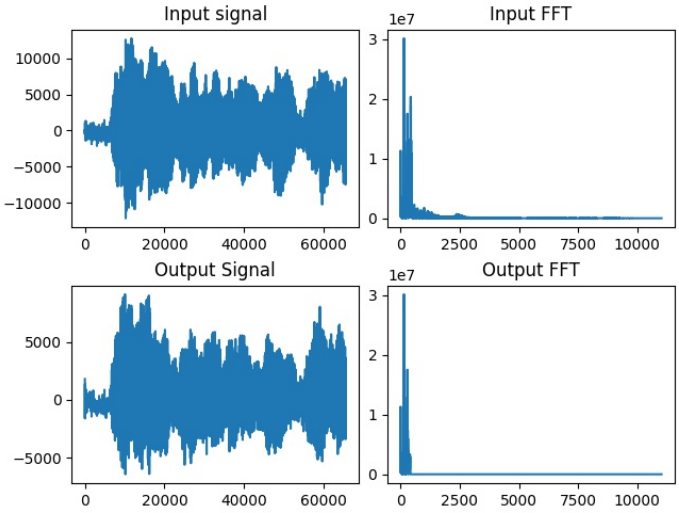


(b) High-pass filter, cutoff: 1000 Hz

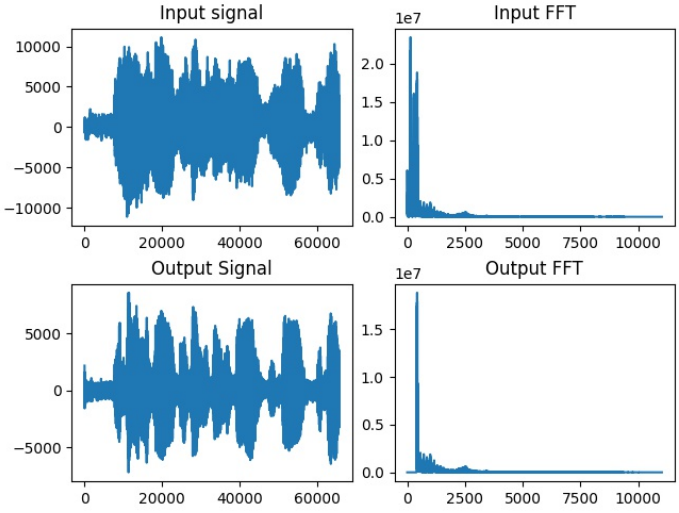


(c) Band-pass filter, lower cutoff: 500 Hz, upper cutoff: 1000 Hz

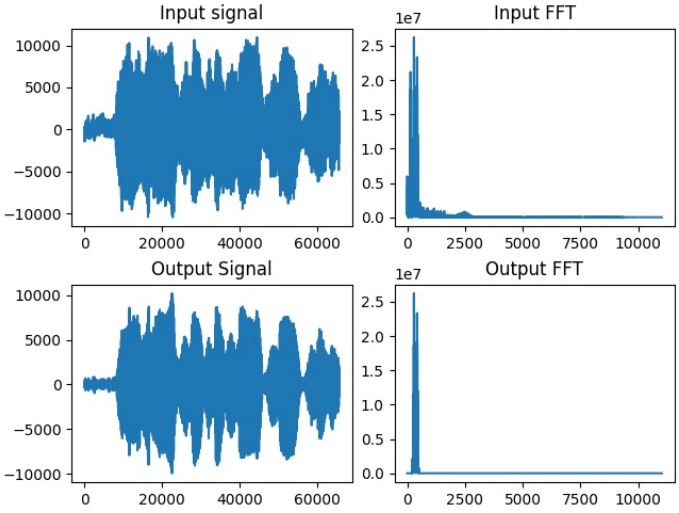
Fig. 3. Pre-recorded audio analysis (Link)



(a) Low-pass filter, cutoff: 400 Hz



(b) High-pass filter, cutoff: 400 Hz



(c) Band-pass filter, lower cutoff: 200 Hz, upper cutoff: 500 Hz

Fig. 4. Live audio analysis (run time = 5 sec)

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

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