# DIGITAL SIGNAL PROCESSING
# RETAKE REPORT

THE USAGE OF LOW PASS FILTER ON NOISE REDUCTION

*https://github.com/dennisdinhhung/butter-lowpass-dsp.git*

Dinh Nguyen Viet Hung
Bi9-111
ICT

# I) Introduction

Digital signal processing has increasingly gained more popularity in recent years as people are in need of better and further advancing communicating technology.

With that, the progression of digital signal processing boosts drastically to satisfy the demands for fast, easy and high quality audio for either everyday communication via social media, online calls for meetings and/or online lectures or classes, or even professional audio work in the music industry or scientific research.

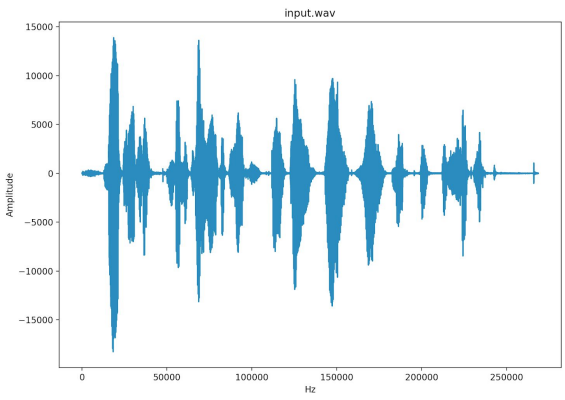
For a normal person, living a mundane life, to be able to acquire high quality audio products is difficult. That is why we needed programming technology to be able to achieve easy and more drastically better quality audio without having to spend money on overpriced audio equipment.

# II) Code Explanation

For the main section of the codes, we will be using and implementing Butterworth's low pass filter (a version of the IIR analog filter).

Firstly, we need an input audio. For this specific project, we will be using a high quality .wav file at 44.1kHz. So the sampling rate will be:

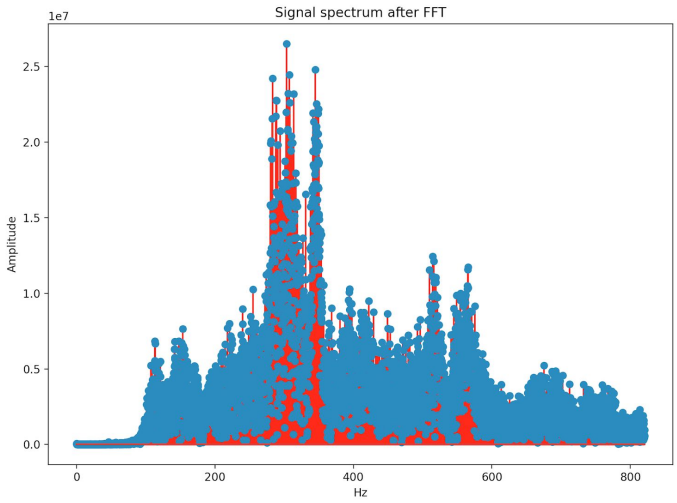
*Fc = 44.1kHz.*



From the input wave, we will use the Fourier transform to achieve a simpler wave in order to add Gaussian noise to the input.wav more effectively using SciPy's fast fourier transform:

*sp.fft.fft(sample)*

After finishing adding the Gaussian Noise to the input.wav audio file, we will be exporting a new .wav file (called input_with_noise.wav) and rescaling its graph for easier viewing and analysis.



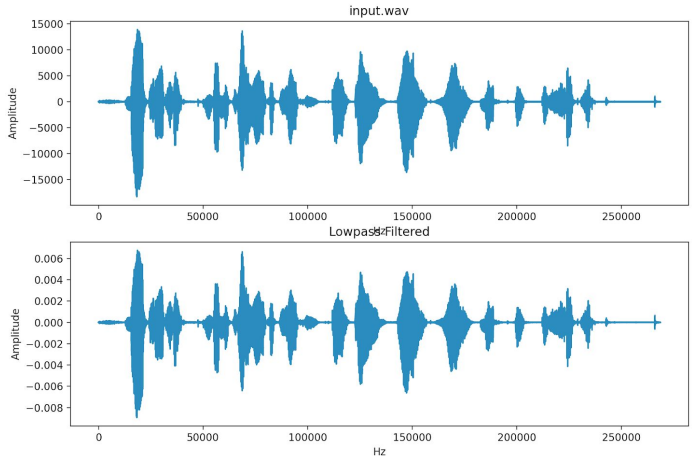

(The rescaled graph illustrates 2 informations: the input.wav and the absolute value of the fourier transformed input.wav)

The third step is putting the newly made input_with_noise.wav through a low pass filter so it filters out the noise with:

*signal.butter(order, W(n), btype, analog)*

For this signal, we will be using:

- 5th order
- W(n) = fs/(fc/2) *[ while ( fc = fs/(t*2) ) (fs = 441000) ]*
- btype = lowpass
- analog = true




*( We can see the differences in Amplitude of the 2 audio files.)*

In the end, we will be exporting a filtered .wav file with significant improvements from its noisy input but with lower volume (dB) and plotting the input.wav and output.wav for comparison using matplotlib's pyplot feature library.

# III) Conclusion

In conclusion, the project is rather simple but very effective in removing noise from the audio, though real life instances may vary in quality, with only one downside of Amplitude reduction.