# Adding and removing noise from an audio file

## Digital Signal Processing Retake Report

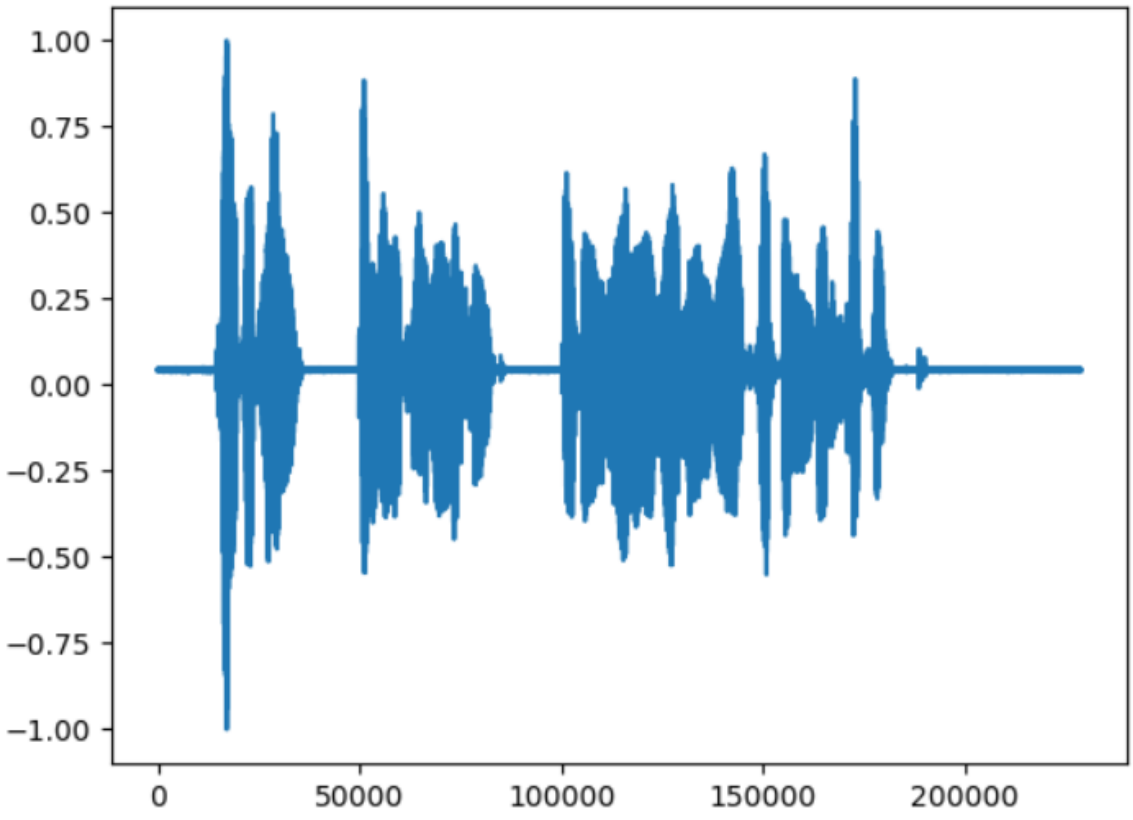
Đào Dương Hoàng Long


July, 2020

# Contents

## 1. Adding white noise

Plot of the signal (saying : “Dear teacher, I'm gonna tell you the truth, I do really really not hating this subject”)




## Signal to Noise Ratio

Signal to noise ratio (SNR) can be defined as follows :

$$SNR = 10log(\frac{RMS_{signal}^2}{RMS_{noise}^2})$$

Definition of SNR

where RMS_signal is the Root Mean Square value of signal and RMS_noise is that of noise.
As in the code, RMS value of the signal can be calculate as follow:

```
RMS_s=math.sqrt(np.mean(signal**2))
```

RMS value of noise:

```
RMS_n=math.sqrt(RMS_s**2/(pow(10,SNR/20)))
```

## Additive White Gaussian Noise (AWGN)

Randomly sampled from a Gaussian distribution with mean value of zero. Standard deviation can vary. It contains all the frequency components in an equal manner (hence “white” noise).
From SNR definition equation, we can obtain:

$$RMS_{noise} = \sqrt{\frac{RMS_{signal}^2}{10^{SNR/10}}}$$

Required RMS value of noise

Also,

$$RMS_{noise} = \sqrt{\frac{\sum n_i^2}{n}}$$

$$STD_{noise} = \sqrt{\frac{\sum (n_i - \mu_{noise})^2}{n}}$$

STD_noise is the standard deviation of noise

Since mean value of AWGN is zero we can see that:

STD_noise = RMS_noise

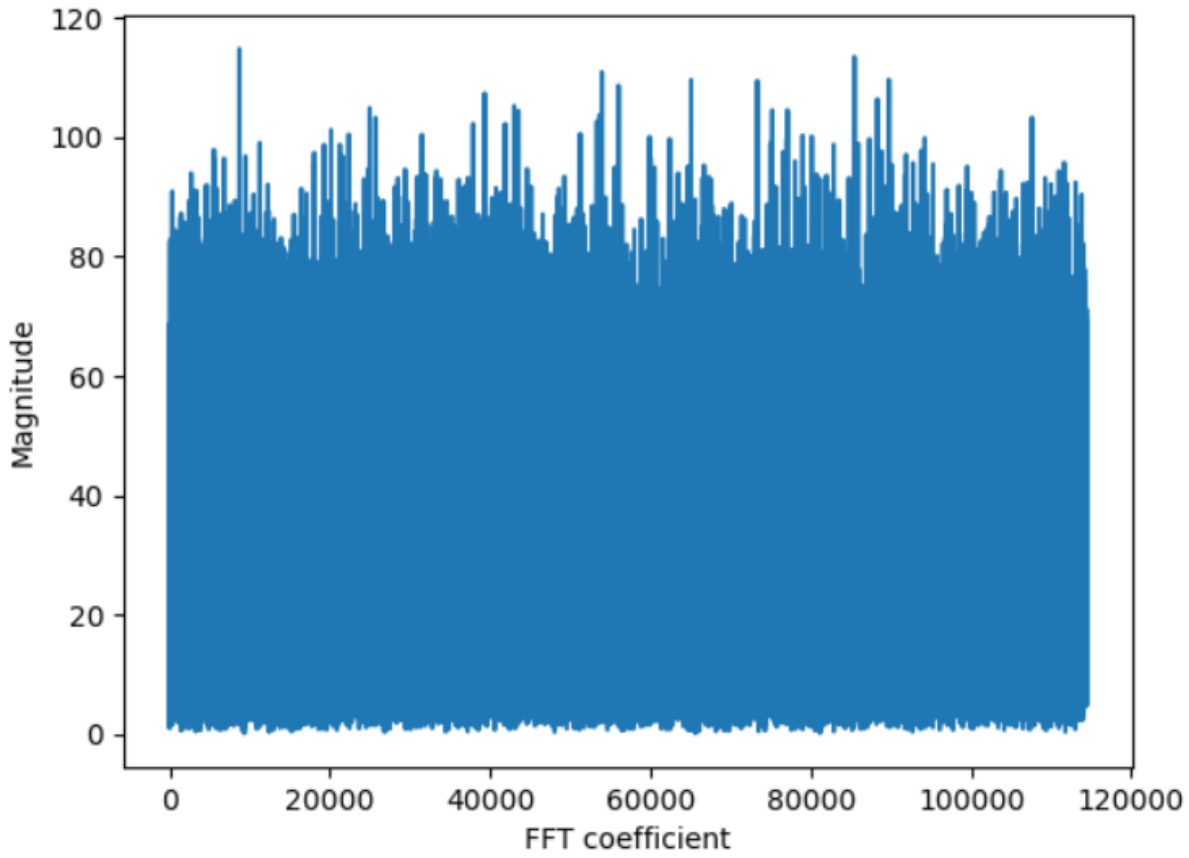
To generate noise, we can sample from a Gaussian distribution with mean =0 and standard deviation = RMS_required:

```
noise=np.random.normal(0, STD_n, signal.shape[0])
```

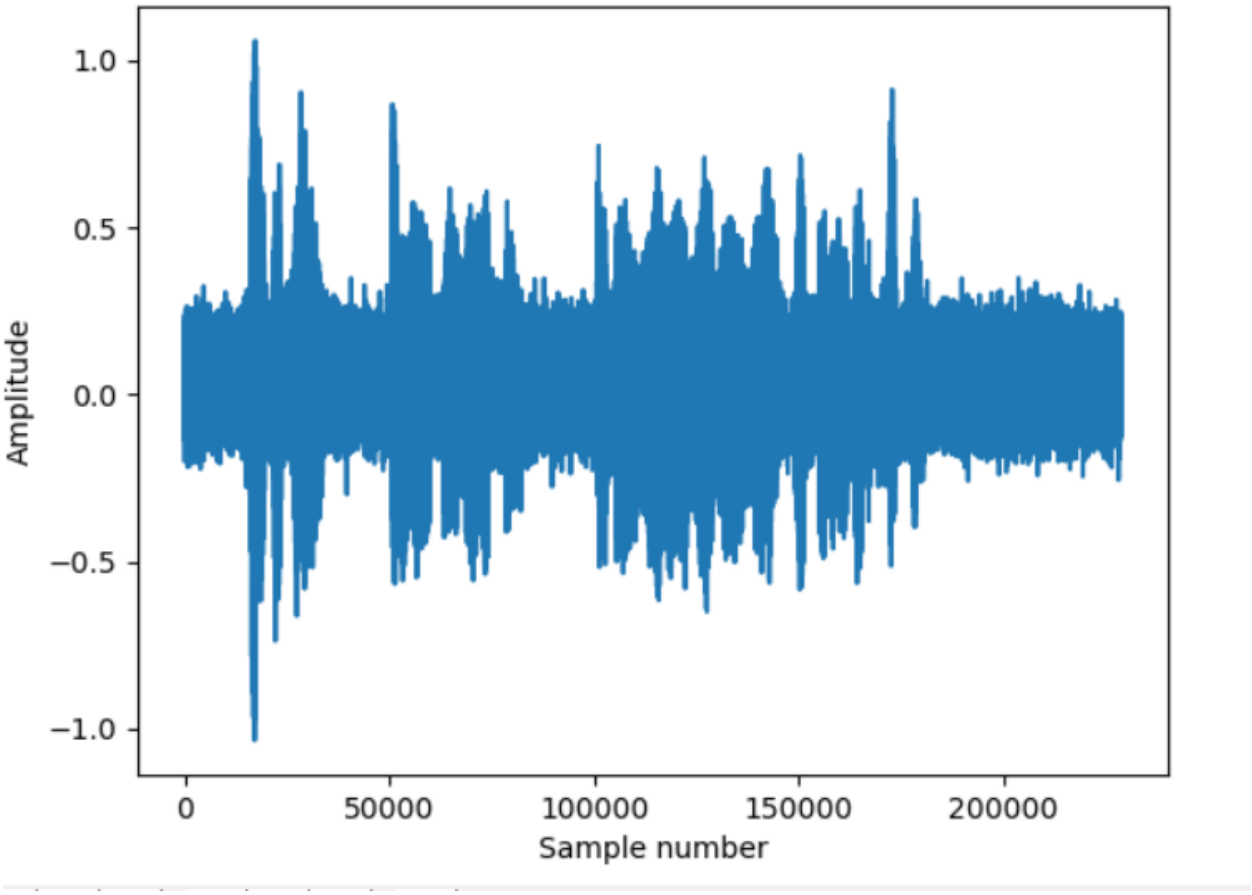
To analyze the frequency content we use fast Fourier transformation:

```
#***convert complex np array to polar arrays (2 apprays; abs and angle)

def to_polar(complex_ar):

   return np.abs(complex_ar),np.angle(complex_ar)

X=np.fft.rfft(noise)

radius,angle=to_polar(X)
```

Frequency distribution of noise:




Finally, we add noise to signal




Input_with_noise signal(SNR= 10)

## 2.Remove noise using low-pass filter

In this small project, I used a Moving average filter which is a simple low-pass filter. Mathematically, a moving average is a type of convolution and so it can be viewed as an example of a low-pass filter used in signal processing.
The parameter of the moving average is the window length**.**
To calculate the window length we have to choose the cutoff frequency (which is 100Hz). The relationship between window length and cutoff frequency is:

If N is the length of the moving average, then an approximate cut-off frequency Fco (valid for N>=2) in normalized frequency F=f/fs is:

$$F_{co} = \frac{0.442947}{\sqrt{N^2 - 1}}$$

The inverse of this is:

$$N = \frac{\sqrt{0.196202 + F_{co}^2}}{F_{co}}$$

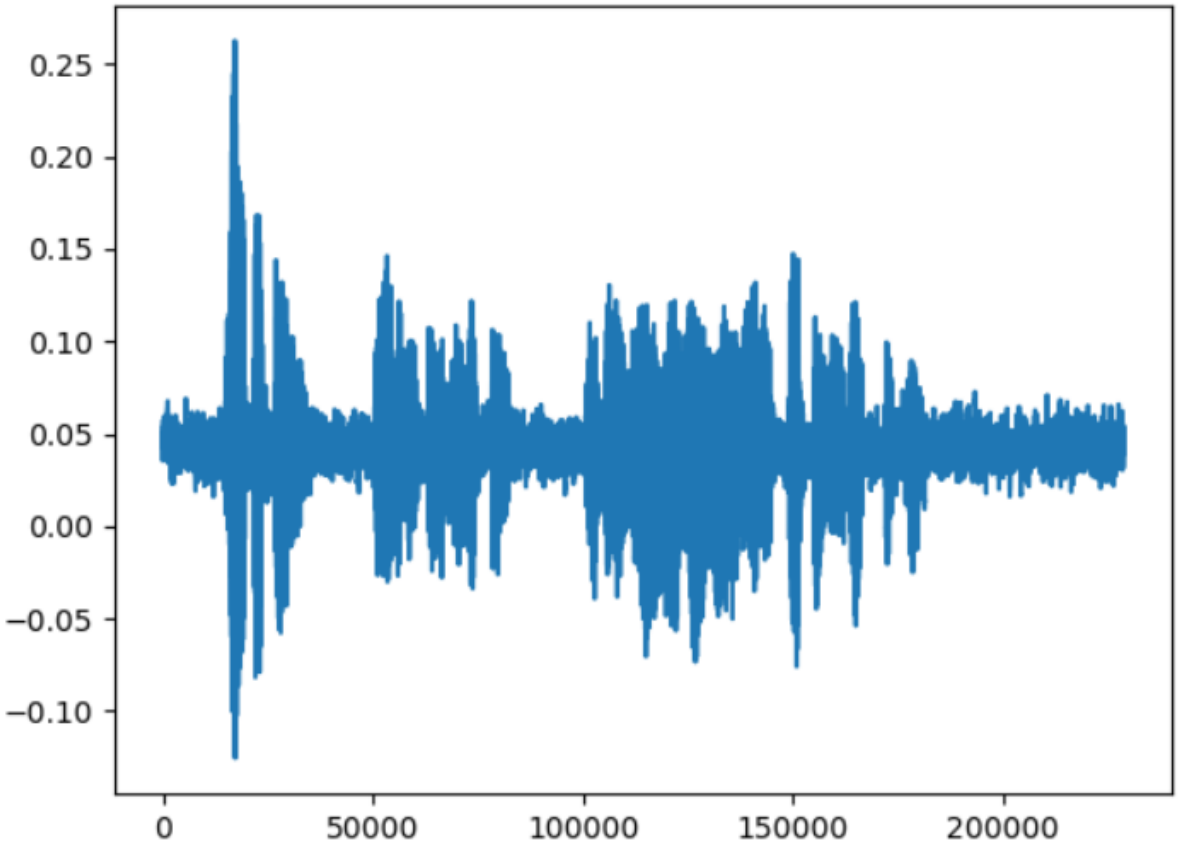
The code will be:

```
freqRatio = (cutOffFrequency/sampleRate)

N = int(math.sqrt(0.196196 + freqRatio**2)/freqRatio)
```

So after calculate the window length, i put the signal through a running mean.The running mean is a case of the mathematical operation of convolution. For the running mean, you slide a window along the input and compute the mean of the window's contents.

```
def running_mean(x, windowSize):
  cumsum = np.cumsum(np.insert(x, 0, 0))
  return (cumsum[windowSize:] - cumsum[:-windowSize]) / windowSize
```

Plot of the output signal:



## 3.Problem

1. AWGN is important because it is easier to model for analytical analyzes and it’s easier to generate. But it may not represent realistic noise conditions for some applications. Maybe add real world noise next time

2.I am still confused about the concept of running mean

## 4. References


### Adding noise to audio clips

https://medium.com/analytics-vidhya/adding-noise-to-audio-clips-5d8cee24ccb8

### Filtering a wav file using python

https://stackoverflow.com/questions/24920346/filtering-a-wav-file-using-python