Sound Filtering Project Report

In this sound filtering project I am going to use sound recorded in real life and add Gaussian noise to it in which the values at any pair of times are identically distributed and statistically independent and hence uncorrelated. In communication channel testing and modelling, Gaussian noise is used as additive white noise to generate additive white Gaussian noise. After a good noisy sound we will now move on to the main part where we make the noisy sound clear like shining silver with the help of some filtering algorithms such as Butterworth filter ,LPF,HPF. Butterworth it's an ideal electrical filter not only completely rejects the unwanted frequencies but also have uniform sensitivity for the wanted frequencies it's a type of signal processing filter designed to have a frequency response as flat as possible in the passband. It is also referred to as a maximally flat magnitude filter. Butterworth also showed that the basic low-pass filter could be modified to give low-pass, high-pass, band-pass and band-stop functionality. By all the important libraries and the filtering algorithms now we have come up with example of Sound filtering project.

Content

- Srep1…….Read the sound recorded
- Step2…….Add Gaussian noise and calculating transformations
- Step3…….Applying the signal and Filtering it

Step 1

Because of that Python language is designed to be user friendly meaning easy to learn it has such libraries that can make our project way easy.

Numpy will help us to manage multi-dimensional arrays very efficiently. Maybe you won't do that directly, but since the concept is a crucial part of data science, many other libraries (well, almost all of them) are built on **Numpy** And **Scipy** provides the core mathematical methods to do the complex machine learning processes

Matplotlib is a plotting library for the **Python** programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter

So we have to read the date from our wav file and change it to usable format

```
(Frequency, array) = read('eagle.wav')          # Reading the sound file.

len(array) # length of our array
```

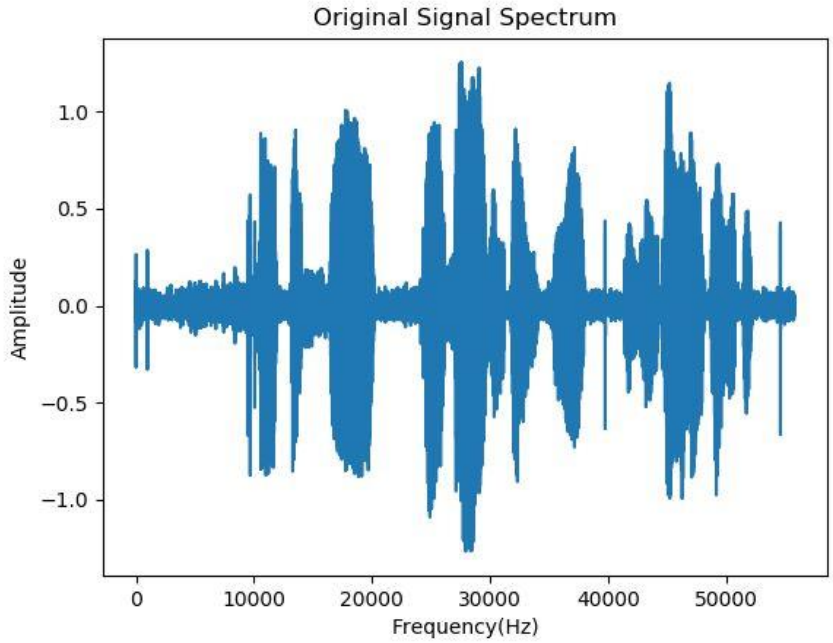


Figure 1

Step 2

Now we calculate the Fourier(FFT) transformation of the signal

```
FourierTransformation = sp.fft(array)
```

```
scale = sp.linspace(0, Frequency, len(array))
```

Fourier analysis is a method for expressing a function as a sum of periodic components, and for recovering the signal from those components. When both the function and its Fourier transform are replaced with discretized counterparts, it is called the discrete Fourier transform (DFT). The DFT has become a mainstay of numerical computing in part because of a very fast algorithm for computing it,

called the Fast Fourier Transform (FFT)

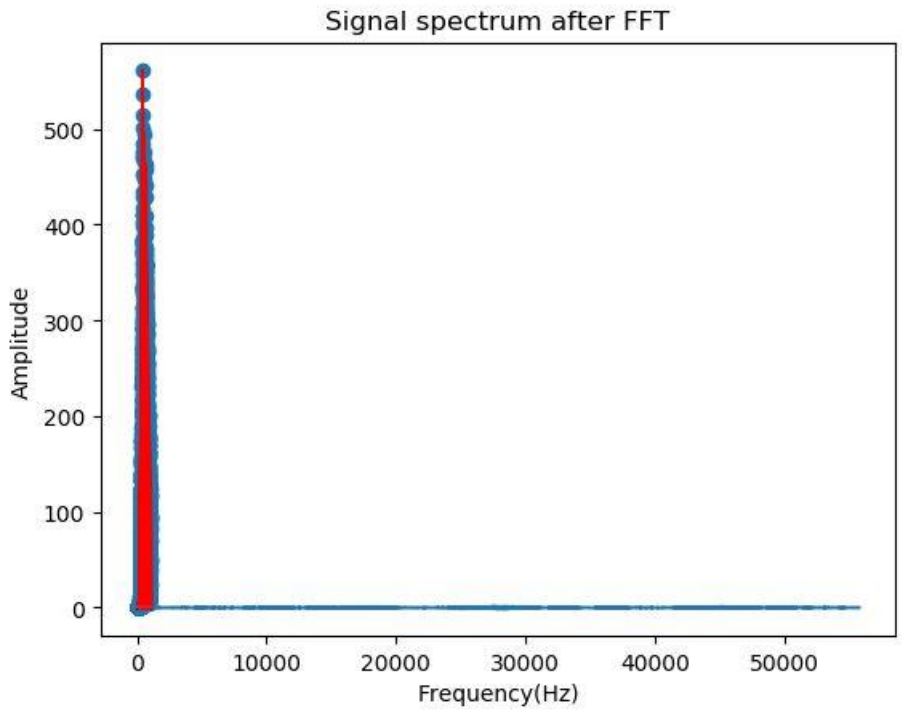


Figure 2

Now we prepare our new sound which is Gaussian Noise with Signal spectrum after FFT. And this will help us make our original sound noisy as broken radio. After we have noise ready we add it to the original sound and save it to the new xxxxx.wav file

```
GaussianNoise = np.random.rand(len(FourierTransformation))
```

Gaussian noise, named after Carl Friedrich Gauss, is statistical noise having a probability density function (PDF) equal to that of the normal distribution, which is also known as the Gaussian distribution. In other words, the values that the noise can take on are Gaussian-distributed.

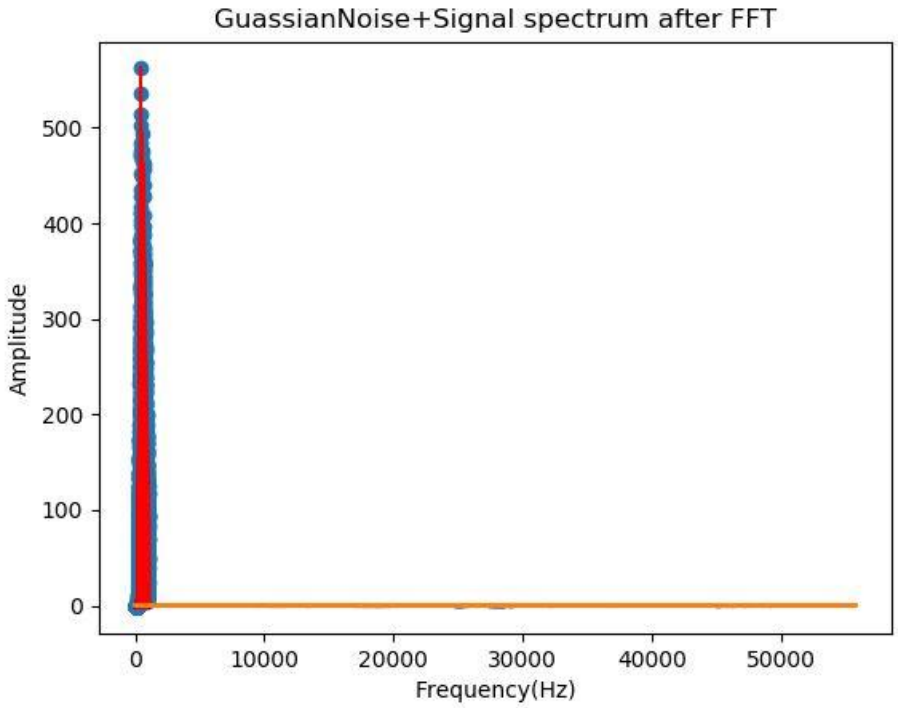


Figure 3

Step 3

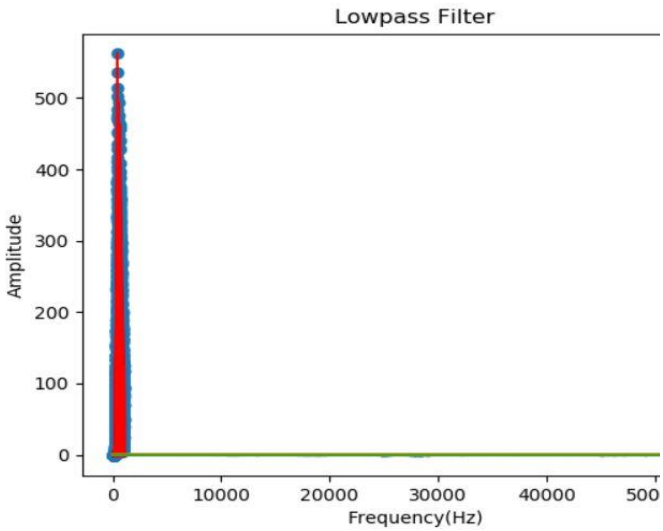
Now we have the signal ready and noisy we can do the main part which is creating **ButterWorth** low and high pass filters and apply it to the signal also saving it to the new .wav file so that we can play it and see the differences

b,a = signal.butter(5, 1000/(Frequency/2), btype='highpass') # ButterWorth filter 4350

c,d = signal.butter(5, 380/(Frequency/2), btype='lowpass') # ButterWorth low-filter

newFilteredSignal = signal.lfilter(c,d,filteredSignal) # Applying the filter to the signal

Butterworth filters are called *maximally flat* filters because, for a given order, they have the sharpest roll-off possible without inducing peaking in the Bode plot. The two-pole filter with a damping ratio of 0.707 is the second-order Butterworth filter. Butterworth filters are used in control systems because they do not have peaking. The requirement to eliminate all peaking from a filter is conservative. Allowing some peaking may be beneficial because it allows equivalent attenuation with less phase lag in the lower frequencies



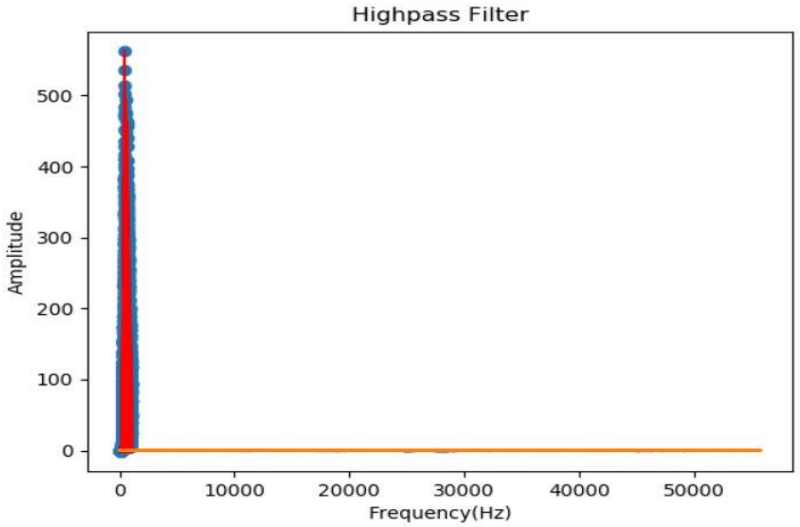


Figure 4.0 ,4.1