HIGHER MATHEMATICS

WS2017/2018 Project ID HiMa_2

JANUARY 7, 2018

RUO-SHAN, TAN | 20813
Friedrich-Heinrich-Allee 25, 47475 Kamp-Lintfort, Deutschland

Abstract

In this project, a simple acoustic signal function is generated from MATLAB through addition of a few sin functions of different amplitude and frequencies. This project in particular, there are 3 sin waves whereby the 2 of higher frequency waves are assumed as “Noise Signals”. In order to remove the 2 unwanted signals, the acoustic signal is first transformed from the time domain to the frequency domain by Fast Fourier Transform (FFT) to identify the frequency components that make up the total original signal. Then, the signal is masked in the Fourier space using the low-pass FIR filter to attenuate unwanted, higher frequency signals. The final output is resulted by revert back the masked signal in Fourier space into the time domain using Inverse Fourier Transform (IFFT).

Theoretical Description

Sound

Sound is produced by a transverse longitudinal wave that has compressions and rarefactions, which they "travel" through the air. A cosine or sinusoidal wave is used as a representation of sound to illustrate the sinusoidal nature of the pressure-time fluctuations through crests and troughs.

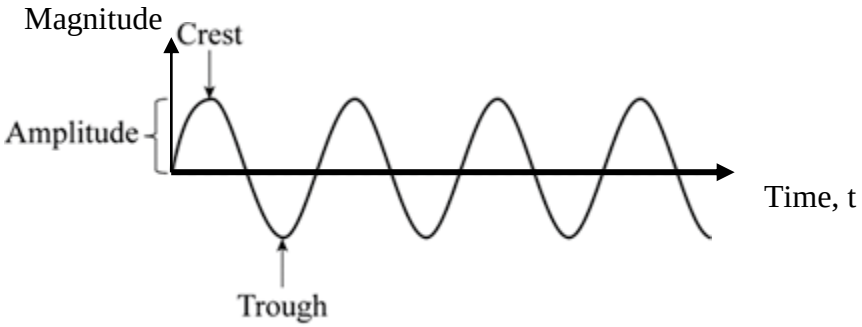


Figure 1 *Representation of Sound Waves using Crests and Troughs. (Annenberg Foundation 2017)*

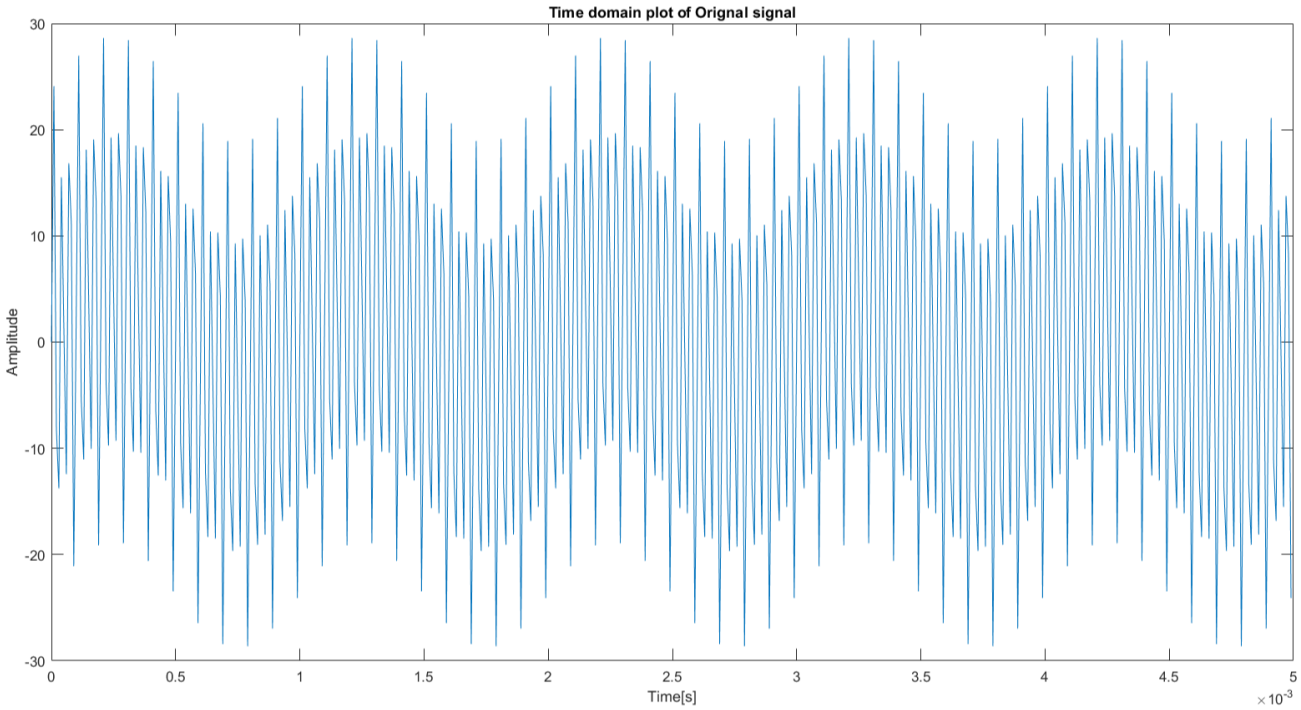
A sinusoid has a specific functional form that is described using the trigonometric function, it has the general function as follows:

$$\boldsymbol{A}\sin(2\pi(ft+\phi))$$

Where A describes the amplitude which corresponds to volume, f is the frequency which corresponds to the pitch, and φ represents the phase. A single sinusoid function acts as the basic building block and they can be combined to form complex sounds simply through superposition of the constituted "building blocks":

$$\boldsymbol{A1}\sin(2\pi(f_1t+\phi 1))+\boldsymbol{A2}\sin(2\pi(f_2t+\phi 2))$$

In this particular project, the original signal consists of the superposition of multiple frequency components, thus resulting in the waveform as follows:



Assuming that we do not know what are the sinusoidal components that constitutes the above signal wave, we apply the concept of Fast Fourier Transform (FFT).

Modifying Signals in Frequency Domain

Fourier Transform

The fundamental knowledge about the Fourier Transform, is that, all waveforms can be described as a sum of simple sinusoids of different frequencies. The relation is also known as the Fourier Series:

$$f(x) = a_0 + \sum_{n=1}^{\infty} \left(a_n \cos \frac{n\pi x}{L} + b_n \sin \frac{n\pi x}{L} \right)$$

Fourier Transform resolves a time waveform to its sinusoidal components by transforming a signal in the time domain into the frequency domain. In this project in particular, Fast Fourier Transform is used. The FFT is a digital implementation of the Fourier transform. Thus, the FFT does not yield a continuous spectrum. Instead, the FFT returns a discrete spectrum, in which the frequency content of the waveform is resolved into a finite number of frequency lines, or bins. The Fourier Transform of a function is defined as follows:

$$\mathcal{F}\{f(t)\} = F(f) = \int_{-\infty}^{\infty} f(t) e^{-2\pi i f t} dt$$

Number of Samples

The sampled time waveform input to an FFT determines the computed spectrum. The relation of N and the sampling frequency is shown as follow:

$$T = \frac{N}{f_s}$$

If an input signal is sampled at a rate equal to f_s over an acquisition time T, a total of N samples is acquired. In this project, the input signal has a time duration of 5ms, and the sampling period is set at 100kHz. Therefore, the total number of samples acquired is 500.

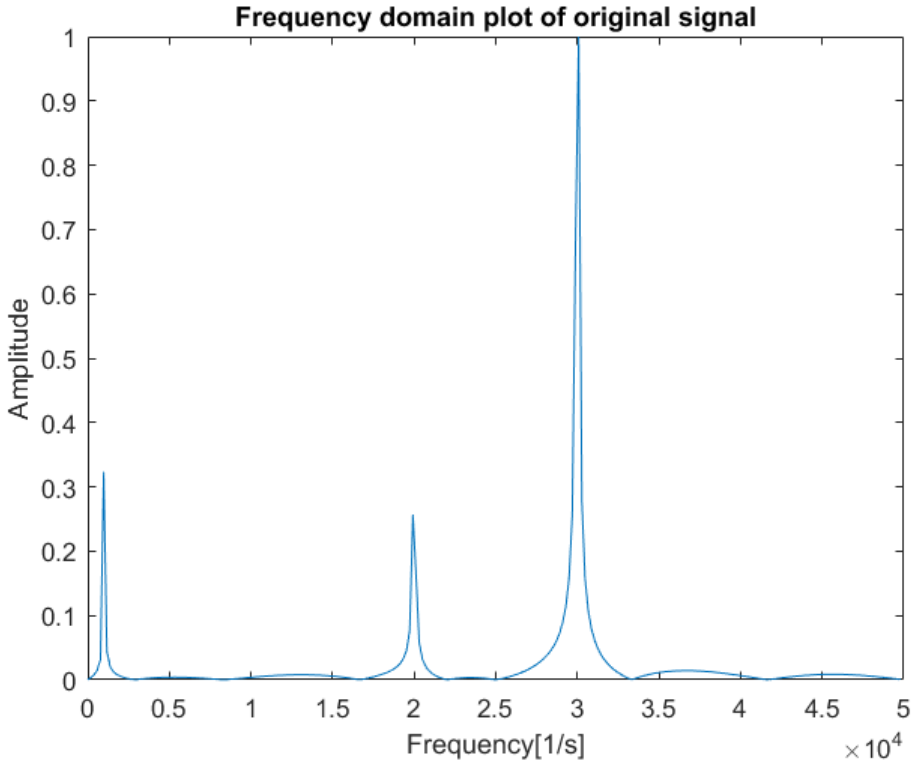
Frequency Resolution and Nyquist Frequency

The spectrum computed from the sampled signal has a frequency resolution df. It is the distance in Hz between two adjacent data points in the Fourier space. The frequency resolution is calculated using the following equation:

$$\mathrm{df} = \frac{1}{T} = \frac{f_s}{N}$$

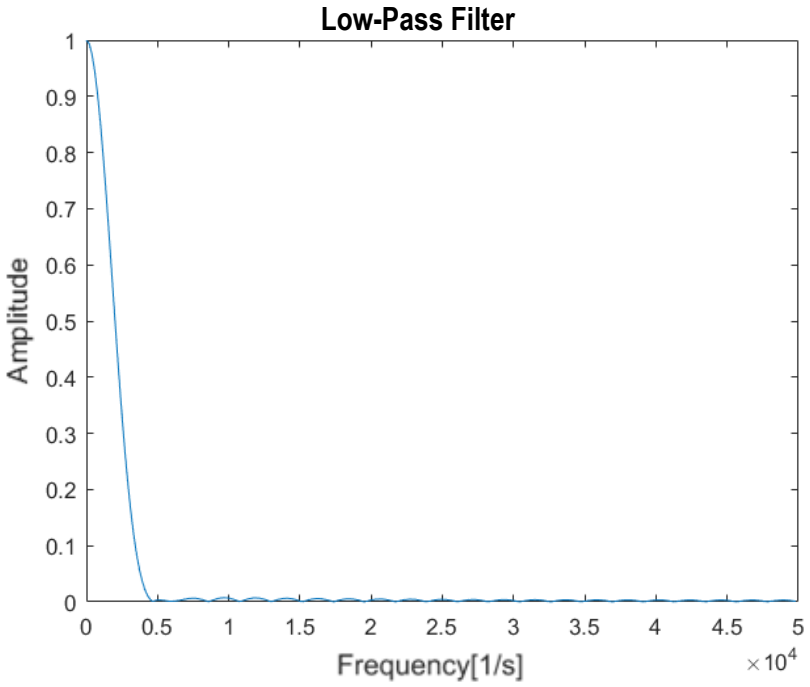
From the relation above, we can see that the frequency resolution can be improved by either increasing the acquisition time or by decreasing the sampling frequency. However, there is a minimum sampling frequency that one should take note of to avoid loss of data, that is defined as the Nyquist Frequency.

According to the Nyquist Theorem, a bandlimited continuous-time signal can be sampled and reconstructed without loss of information from its sample if and only if the waveform is sampled with a sampling frequency of at least twice of the highest frequency component in the input signal. Taking the project as an example, the input signal is made up of 3 main frequency components, having the highest frequency component of 30kHz. Therefore, in order to be able to sample the input signal without loss of information, the sampling frequency f_s should be at least 60kHz. In the MATLAB application, sampling frequency of 100kHz is used thus no information is loss when the output is reconstructed. The transformed signal input in the frequency domain is plotted as the following graph:

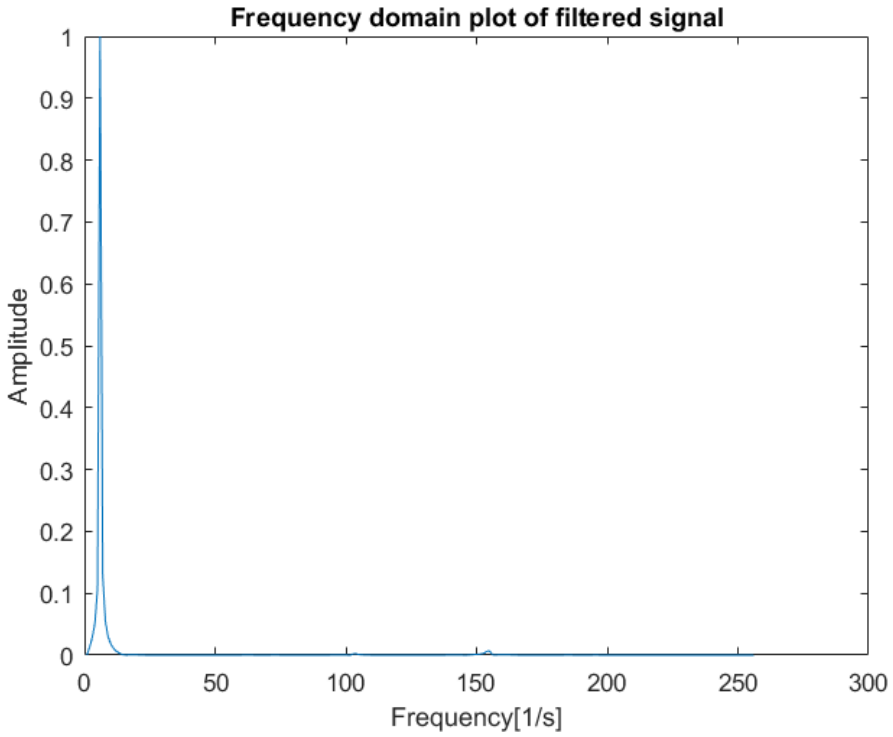


Modifying Signal with Low-Pass Filter

A Low-pass Filter is a filter that passes signal with a frequency lower than the cut-off frequency, and attenuates signals with frequencies higher than the cut-off frequency. In this project, the designated low-pass frequency has a cut-off frequency of 1.5kHz and it is illustrated in the Fourier space as the following diagram:



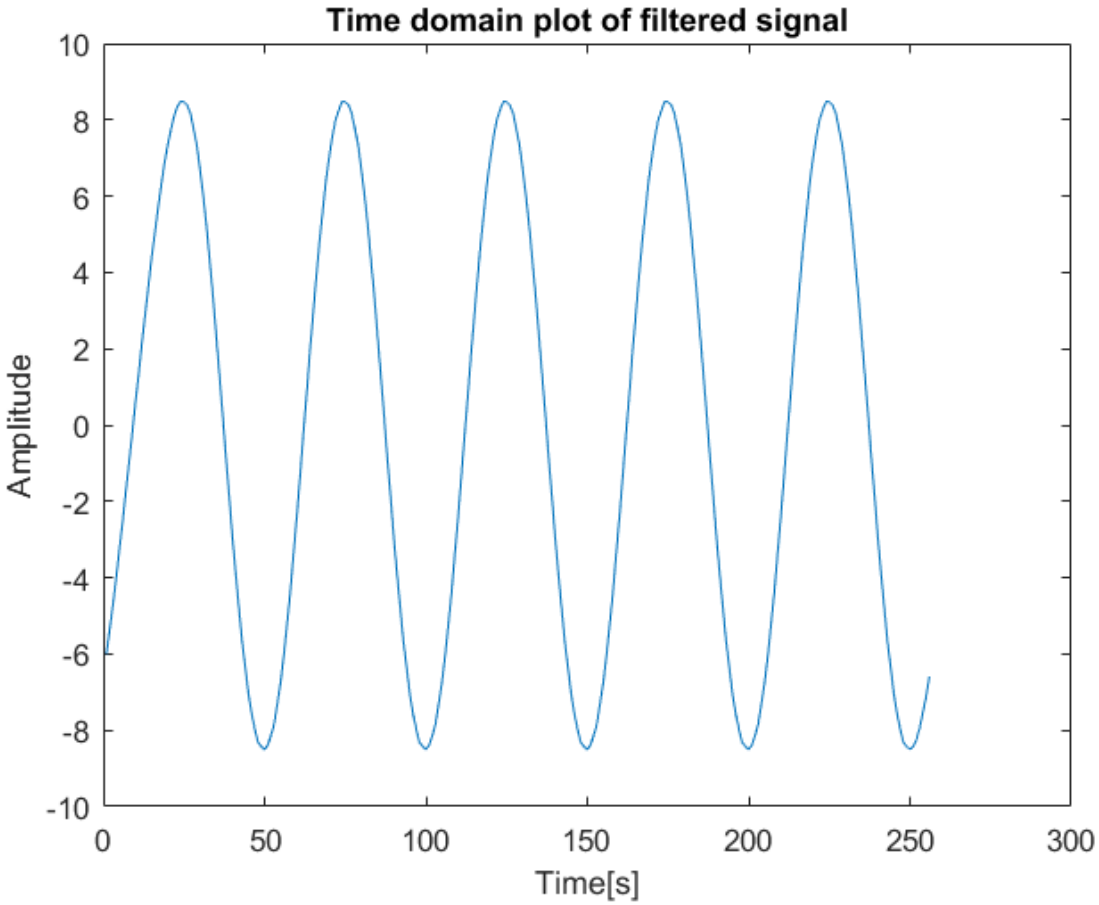
As shown in the filter in its Fourier space, the filter has an absolute magnitude of 1 at lower frequency values. Therefore, when the filter and input signal is directly multiplied in the Fourier Space, only the lower frequency component is multiplied with an absolute value of 1, while the other 2 higher frequency components are multiplied with value of zero. The resulting output in the frequency domain is therefore plotted as follows:



Finally, the resulting output signal should be reverted back to its time domain representation to visualize the respective change after filtering it with a low-pass filter. This can be done using the function named as Inverse Fourier Transform. The inverse Fourier Transform is defined by the following equation:

$$\mathcal{F}^{-1}\{F(f)\} = \int_{-\infty}^{\infty} F(f)e^{2\pi ift}df = f(t)$$

The resulted inverse Fourier Transform is plotted as the following figure:



The above result is expected as when the low pass filter has attenuated the 2 higher frequency components, the resulting signal is only left with one single frequency component. Therefore, when inverse Fourier Transform is applied to revert back the signal from its frequency domain to time domain, it is therefore a single sine wave as a result.

Critical Reflection

I have discovered the advantages of using Fourier Transform as a tool in signal processing, be it 1 dimensional signals such as sound, music, or 2 dimensional signals such as image processing. In the time domain, where signal representation is the amplitude of the signal over time, it is difficult to identify what exactly constitutes the signal. However, using Fourier Transform, it allows one to view the signals in the frequency domain to simplify difficult problems. It is because

Fourier Transform answers us the question: “What frequency signals are present in my signal? In what proportions?” Taking this project activity as an example, in the time domain, the signal representation is a very noisy signal where I have no idea what signal constitutes the waveform. Things are much more simplified when Fourier Transform is applied. It clearly shows the 3 distinct frequency values that makes up the waveform and also its proportions, that allows me to choose which frequency sine wave to attenuate and which to allow passing through the filter.

References

1. Annenberg Learner 2017, Representation of Sound Waves using Crests and Throughs, digital image, Annenberg Foundation, accessed 26th January 2018, <https://www.learner.org/courses/mathilluminated/units/10/textbook/03.php>
2. Weisstein, Eric W, "Fourier Series." From MathWorld--A Wolfram Web Resource, accessed 26th January 2018, <http://mathworld.wolfram.com/FourierSeries.html>
3. Weisstein, Eric W, "Fourier Transform." From MathWorld--A Wolfram Web Resource, accessed 26th January 2018, <http://mathworld.wolfram.com/FourierTransform.html>
4. Wikipedia, "Filter (signal processing)" From Wikipedia--The Free Encyclopedia, accessed 26th January 2018, <https://en.wikipedia.org/wiki/Filter_(signal_processing>