Reducing Audio Bandwidth with an FFT-based Low-Pass Filter

Trent Geisler
Analytics and Data Science
Kennesaw State University
Kennesaw, Georgia 30144

Andrew Henshaw
Analytics and Data Science
Kennesaw State University
Kennesaw, Georgia 30144

Lauren Staples
Analytics and Data Science
Kennesaw State University
Kennesaw, Georgia 30144

***Abstract*—The abstract goes here.**

I. Introduction

Filtering an audio signal is extremely important for many reasons. When an appropriate filter is designed and used to remove any unwanted frequencies, then the bandwidth and power necessary for audio signal transmission is reduced. Why is this important? The frequency bands used for the transmission of many types of signals are scarce resources. Every transmitter that can interfere with others has to operate in a licensed band and is subject to bandwidth limitations. This is why multiple radio stations can operate in the same geographical area. They are assigned different center frequencies and they have a certain bandwidth they can occupy. If one radio station transmits beyond their assigned bandwidth, they will impact the signal transmission of other local radio stations. One way to ensure that a radio station stays within their assigned bandwidth is through the use of filtering. [2]

Additionally, if you travel to a different city, you will find radio stations that operate at the same frequency as the radio stations located locally here in Atlanta. This is possible because radio transmitters are limited in power. With unlimited power, for example, one could hear FM 106.7 throughout the United States and no other radio station would be able to use the frequency 106.7 MHz. But power is limited, and high-power radio transmissions cost a lot of money [2]. This is why some radio stations like to brag about how powerful their transmissions are. It is also why start-up radio stations have a very limited range. Therefore, if power costs money, one does not want to waste it by boadcasting frequencies that most humans cannot hear.

Both bandwidth and power savings are the motivation behind Fast Fourier Transform (FFT)-based low-pass filters. This paper will demonstrate how a Finite Impulse Response (FIR) low pass filter can remove unecessary frequencies in order to reduce the bandwidth and power required for transmission.

A. Objective

Our objective is to design and demonstrate an FFT-based FIR low-pass filter to reduce bandwidth of a recorded audio signal for the purposes of AM Radio transmission.

II. Methods

We collected data, designed a low pass filter, pushed the signal through the filter, and then evaluated the resulting signal.

A. Data

We collected an audio signal by recording the song 'Flight of the Bumblebee' at the full frequency spectrum that a compact disc (CD) is recorded at, which is 44.1 kHz. The human ear does not even hear this full spectrum. This means that there are potential bandwidth 'savings' for transmission! This collected audio recording is referred to as the 'raw signal.'

B. Filter Design

The typical Human can only hear frequencies in the range of 20Hz through 20kHz, and this range only decreases as humans age [3]. Using the website http://onlinetonegenerator.com/hearingtest.html, we will test the hearing range of each member listening to our final presentation. This demonstration will provide a tangible example why appropriate filtering of an audio signal will have little to no impact on the quality of sound that a person hears after the filtering.

When our project group performed the hearing test, the highest audible frequency was 15kHz. Since our project group was not able to hear the frequencies above the 15kHz threshold, any low-pass filter that removed the high frequencies above 15kHz would have no impact on the quality of the audio signal that we would hear. Recall, removing the unecessary frequencies from an audio signal transmission provides a few nice benefits such as the following:

1) It reduces the necessary power for transmission of the audio signal
2) It reduces the necessary bandwidth for transmission

Filtering is the processing of a time-domain signal resulting in some change in that signals original spectral content. The change is usually the reduction or filtering of unwanted input spectral components. [4]. Given the human threshold for hearing, an obvious opportunity is to apply low-pass filtering in order to transmit audio signals that humans can actually hear with smaller bandwidths. The ideal low pass filter would completely eliminate all frequencies above a certain cutoff point (in our hearing test example, the cutoff would be at

15kHz) while passing all frequencies below the cutoff point [5].

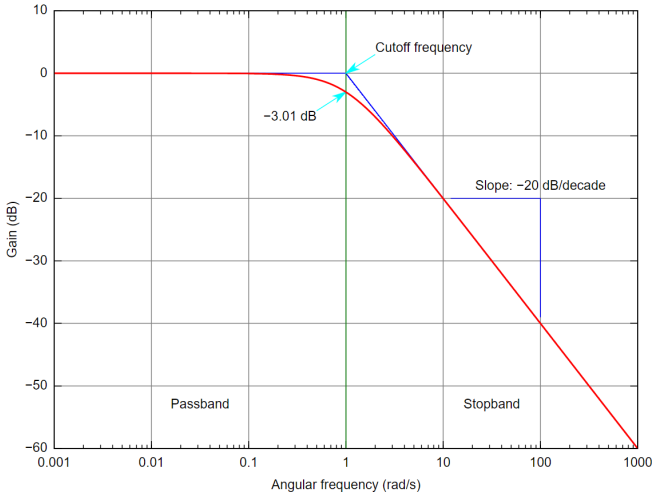

Fig. 1. Example of Low Pass Filter: https://upload.wikimedia.org/wikipedia/commons/6/60/Butterworth_response.svg

In order to filter out higher frequencies in an audio signal transmission, we will build a low pass filter. Specifically, we will describe how a low pass filter is created using a finite number of non-zero filter coefficients which is called a *Finite Impulse Response* filter or FIR. Given an impulse response, we can find the coefficients of the filter, and vice versa [2]. For example, Figure 1 shows a graphical depiction of a low pass filter that begins to cut out the frequencies above 1 rad/s. Frequencies below the cutoff frequency are considered to be in the "Passband," or allowable frequency band. Frequencies above the cutoff frequency filtered out and are considered in the "Stopband." Ideally, the slope between the Passband and the Stopband would be as steep as possible to limit the passing of unwanted high audio frequencies. The question still remains, how do you build a filter that eliminates these frequencies? What is an impulse response? Why do we need to find coefficients of a filter and what do they do? Let's address those questions next.

1) Impulse Input, Impulse Response, and FFT: In our filtering example, we will be dealing with a Linear Time-Invariant (LTI) System. A linear system is a class of systems where the system's outputs are the sum of the outputs of it's parts. Additionally, Time-Invariant refers to a system where a time delay in the input sequence causes an equivalent delay in the output sequence. Now, if we are given a LTI system, we can calculate everything about the system if we know the *unit impulse response*. The *unit impulse response* refers to the system's time-domain output sequence when the input is a single unity-valued sample (unit impulse) surounded by zero-valued samples. Furthermore, knowing the impulse response of an LTI system, the *output sequence* is calculated by taking the *convolution* of the input sequence and the system's impulse response [4]. We will talk more about convolutions later, but we typically do not perform convolutions in the time domain since they are computationally expensive. Instead, multiplication is performed in the frequency domain. In order to transform the impuse response into a *frequency response*, we take the Fast Fourier Transform (FFT) of the impulse response [4].

Let's explain with a simple example, if we let $x(n)$ be a discrete-time sequence of individual signal amplitudes, then we can define a simple LTI *averager* system as follows:

$$y(n) = \frac{1}{4}\left[x(n) + x(n-1) + x(n-2) + x(n-3)\right] = \frac{1}{4}\sum_{k=n-3}^{n} x(k)$$

Given, this simple averager, we can show the block diagram in Figure 2 (a), impulse input and impulse response in Figure 2 (b), and the frequency magnitude response created from the FFT of the impulse response in Figure 2 (c). The block diagram in Figure 2 (a) - also referred to as the *Filter Structure*, simply shows how an impulse input is transformed into a impulse response. The impulse response, $y(n)$, is created by passing the impulse input, $x(n)$, into the system and storing the four most recent values. Once there are four values, they are added together and multiplied by $\frac{1}{4}$ to calculate the average. The sequence proceeds one step averaging the most recent four values of the impulse input, $x(n)$, along the way. Since there are four separate input sample values to calculate an output value, the structure of this filter can be referred to as a *4-tap tapped-delay line FIR filter* using digital filter vernacular. The coefficients of this filter are all $\frac{1}{4}$ and we can use an Fast Fourier Transform on the filter to provide all the frequency domain information.

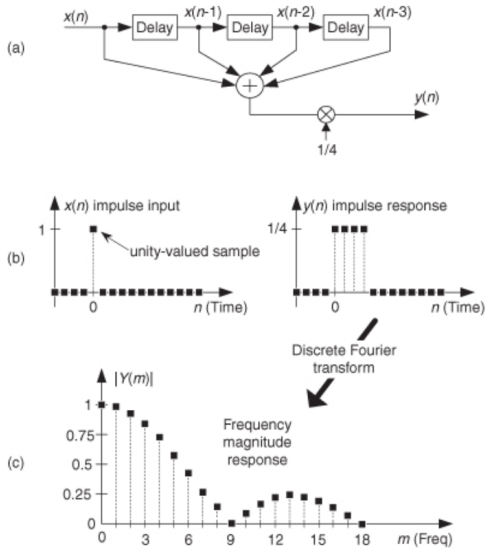

Fig. 2. Block Diagram (a), Impulse Input and Impulse Response (b), and Frequency Magnitude Response after FFT of Impulse Response (c). Taken from Reference [4] below and is Figure 1-12 from Chapter 1

From Figure 2 (c), the FFT transforms the impulse response $y(n)$ into $Y(m)$, which is the frequency domain information. $Y(m)$ provides the frequency magnitude response of the

simple 4-point averager. This is actually an example of a low pass filter where the *averager* reduces the amplitude (attenuates) the high-frequency signal. In our construction of a filter, we will have to use a different impulse response since we want to remove, not attenuate, the high frequency information content. The following section will explain how we design a Finite Impulse Response (FIR) filter using the window method in order to create a low pass filter that removes high frequency information content.

2) Finite Impulse Response (FIR): An FIR filter has a finite duation of nonzero output values given a finite duration of input values (this is how they were named!). Recall, in our example above, we used four "taps" that were all equal to $\frac{1}{4}$. The two factors that affect an FIR filter's frequency response are the number of taps and the coefficients [4]. If we were to calculate the impulse response, $y(n)$, from the impulse input, $x(n)$, in the time-domain, then we would have to perform the mathematical operation of a convolution. More specifically for an *M*-tap filter, we would calculate $y(n)$ as follows:

$$y(n) = \sum_{k=0}^{M-1} h(k)x(n-k)$$

Where $h(k)$ are the filter coefficients for each of the taps. **The terms FIR filter coefficients and impulse response mean the same thing** [4]. We can re-write the above equation using *convolution* notation as follows:

$$y(n) = h(k) \star x(n)$$

The major concept is that convolution in the time domain is equal to multiplication in the frequency. The process in which we can move from one domain to the other is through the Fast Fourier Transform (FFT). The FFT of $y(n)$ is equal to $Y(m) = H(m)X(m)$, which is the spectrum of the filter output. In a similar way, we can determine $h(k) \star x(n)$ by taking the inverse of the FFT (we will denote this as IDFT - Inverse Discrete Fourier Transform) [4]. The important relationships are as follows:

1) $x(n) \overset{FFT}{\longrightarrow} X(m)$
2) $X(m) \overset{IDFT}{\longrightarrow} x(n)$
3) $h(k) \overset{FFT}{\longrightarrow} H(k)$
4) $H(k) \overset{IDFT}{\longrightarrow} h(k)$
5) $y(n) \overset{FFT}{\longrightarrow} Y(m) \Rightarrow h(k) \star x(n) \overset{FFT}{\longrightarrow} H(k)X(m)$
6) $Y(m) \overset{IDFT}{\longrightarrow} y(n) \Rightarrow H(k)X(m) \overset{IDFT}{\longrightarrow} h(k) \star x(n)$

Now that we can easily move between the frequency and time-domains using the FFT or IDFT, let's design our own low-pass filter by determining the desired frequency response and moving back to the time domain through an IDFT to calculate the filter coefficients that will provide the desired frequency response. Recall, in our hearing test example, our project group was unable to hear any frequency information content above 15kHz. As such, we will attempt to design a low-pass filter that removes all of the frequency content above this threshold. Therefore, our *cutoff* frequency is 15kHz (see Figure 1 for a depiction of the cutoff frequency). The method that we will use is to define the individual frequency-domain samples representing $H(m)$, and then use Python and the inverse FFT function on these samples to give the FIR filter coefficients. In order to do this, we must define $H(m)$ over the frequency span from 0 to f_s, where f_s is the sampling rate. As we will discuss in the next section, the sampling rate that we will use is $f_s = 44.1$ kHz.

C. Nyquist-Shannon Sampling Theorem

When recording audio signals, one has to set a sampling frequency (or a frequency for recording the signal). There are several reasons why we set a sampling frequency:

1) Data storage conservation,
2) Bandwith conservation,
3) Power conservation.

The formula for sampling is:

$$x[n] = x(t)|_{t=nT_S} = x(nT_S) \tag{1}$$

However, we have to be careful not to set this sampling frequency too low, or else we run into a problem called aliasing.

III. Evaluation of Filtered Signal

We reduced the raw signal from 44.1 kHz to () kHz, with no perceived quality degradation to the three judges.

IV. Conclusion

The conclusion goes here.

References

[1] *GNU Radio*, https://www.gnuradio.org/.
[2] Baxley, R., Henshaw, A., Nowlan, S., & Trewhitt, E. *Software-Defined Radio with GNU Radio: Theory and Application, Georgia Tech Professional Education course notes.* (2017)
[3] Rossing, Thomas (2007). Springer Handbook of Acoustics. Springer. pp. 747, 748. ISBN 978-0387304465.
[4] Lyons, Richard G. Author. "Understanding Digital Signal Processing." Ch. 1, 3, and 5. Web.
[5] https://en.wikipedia.org/wiki/Low-pass_filter. Accessed Nov. 7, 2019.
[6] Harris, Fred J. "Multirate Signal Processing for Communication Systems." Page 216, Equation 8.16.