Sarker Nadir Afridi Azmi

1001644326

CSE3313 – 001 – Introduction to Signal Processing

# Butterworth Filter Design

Introduction & Problem statement

The purpose of this project was to design and implement a Butterworth filter which took a noisy audio file and filtered out the high frequency noise in the background. To give the reader some context, the hissing sound present in the background in a lot of audio files is the thing that the filter is trying to reduce. This is the reason for choosing the Butterworth filter as it is a low pass filter which filters out high frequency signals.

Analyzing the signal

*Getting the DFT plot*

Even before we can begin designing our filter, we need to know what signals we are trying to reduce. To analyze the audio, the file was read in and a graph of the magnitude of the Discrete Fourier Transform vs Frequency was plotted. The MATLAB audioread() function was utilized to get the audio data in an array and also the sampling rate of the audio. Since the x-axis frequency values were required for the plot, using the sampling rate, an array of frequency values was generated spaced at regular intervals. Using the MATLAB fft() function, the Discrete Fourier Transform was obtained and a transpose of the returned array was taken due to the nature of how my frequency values were generated (MATLAB requires that the dimensions of the x and y axis be the same). Since the DFT is complex, the magnitude of the DFT was taken. To maintain the convention of the frequency plots where the center of the frequency axis is zero, the fftshift() function was utilized to shift the signal so that ω is at the center of the graph instead of being the first element. Now, after plotting, the graph show in Figure 1 was obtained.

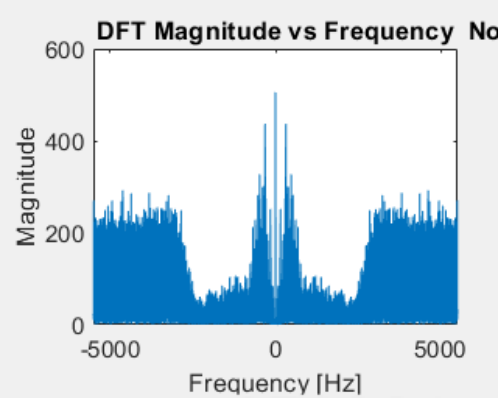


*Figure 1 – DFT magnitude vs Frequency*

*Getting the log plot of the DFT*

Alongside the DFT, I also needed to know below which decibel level, the noise was attenuated enough that the filtered version did not contain the hissing. For this, the DFT array was normalized using the maximum value inside of the DFT array and using the formula 20*log10(abs(DFT)/max(abs(DFT)), a new array was obtained and plotted using the same frequency values as before to get the plot shown below in Figure 2.

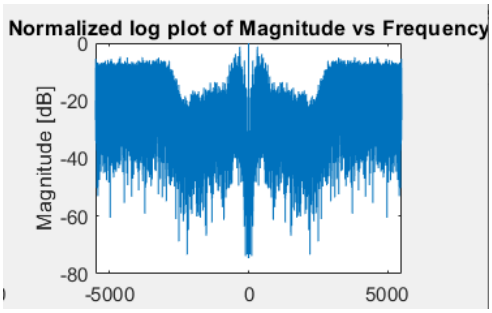


*Figure 2 – Normalized log plot of Magnitude vs Frequency*

Filter Design

Based on the graphs obtained shown in Figure 1 and Figure 2, the below specifications for the Butterworth filter were chosen.

$\omega_p$ = 1073 Hz (End of the passband)

$\omega_s$ = 2267 Hz (Start of the stop band)

$1 - \delta_p$ = -1 dB (Attenuation in the passband)

$\delta_s$ = -80 dB (Attenuation in the stopband)

These specifications were used to obtain the filter shown below in Figure 3 (Pardon my MS paint skills):

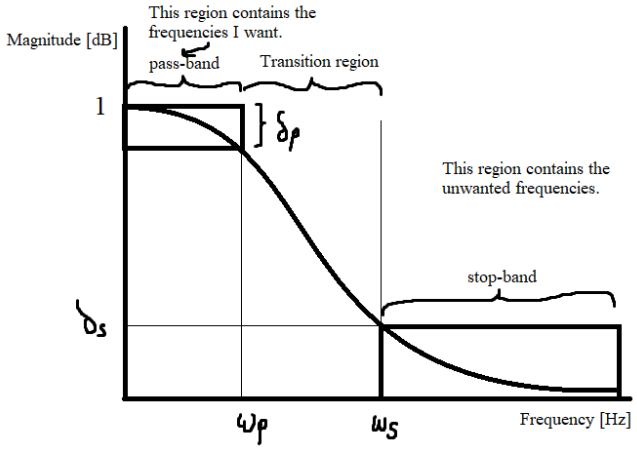


*Figure 3 – Filter Design*

*Calculations*

For the purposes of my calculation, I have auxiliary functions to map the frequency to angular frequency (line 86), transform Ω to ω for bilinear transformation, and convert from decibels which takes a decibel value and converts it into a value we can work with.

Following the lecture, I defined two equations to make calculations easier:

1) $k_p = \frac{1}{(1 - \delta p)^2} - 1$ [1]
2) $k_s = \frac{1}{\delta s^2} - 1$ [2]

An analog design of the Butterworth filter was taken and then mapped to a digital one. The transfer function of the analog filter is given below.

$$|H_a(s)|^2 = \frac{1}{1+(\frac{j\Omega}{j\Omega_c})^{2N}} \quad [2]$$

To map the analog filter to a digital one, bilinear transformation was used. ω was mapped to Ω and was found using equation 4 below.

$$\Omega = 2*\tan(\omega) \quad [4]$$

Now, all the required variables are there to calculate N. The N in the equation determines the order of the filter. The order determines how fast the transition region part of the curve in Figure 3 falls. Think of Figure 3 in the analog domain. So, $1 - \delta_p$ is at $\Omega_p$ and $\delta_s$ is at $\Omega_s$. This gives us the equations below.

$$\frac{1}{1+(\frac{j\Omega_p}{j\Omega_c})^{2N}} = (1 - \delta p)^2 \text{ [5]}$$

$$\frac{1}{1+(\frac{j\Omega_s}{j\Omega_c})^{2N}} = (\delta s)^2 \text{ [6]}$$

Re-arranging the equations, a form is obtained which resembles $k_p$ and $k_s$ and substituting both these variables and combining both equations the below equation is obtained.

$$N = \frac{0.5 * \log(\frac{k_s}{k_p})}{\log(\frac{2*\tan(\omega_s)}{2*\tan(\omega p)})} \text{ [7]}$$

The value obtained has to be rounded up using the ceil() function because we can not have a fractional filter order.

Changing the order of the filter means that the points where the cut-off or stopband frequencies are present change. This needs to be accounted for and for bilinear transformation, we meet at the stop band frequency using equation 8 since there is no aliasing and allow lower attenuation for the passband.

$$\Omega_c = \frac{2 * \tan(\omega_s)}{k_s^{\frac{1}{2N}}}$$

Now, for the final step, the $\Omega_c$ is converted back into $\omega_c$ using equaltion 4 for the equivalent cut-off frequency in the digital domain.

The plot in Figure 4 shows the low-pass nature of the filter. It is a plot of the logarithmic gain of the frequency response versus frequency.

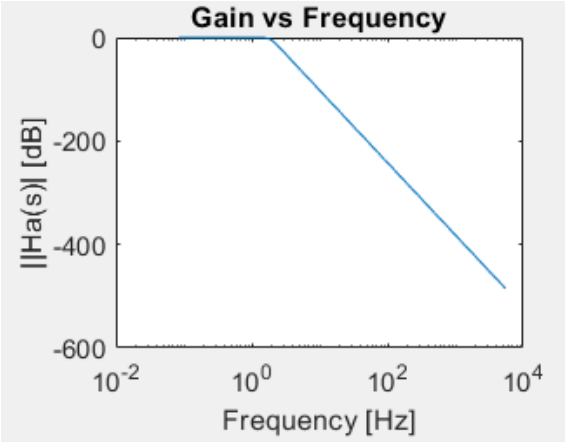


*Figure 4 – Logarithmic gain of the frequency response versus frequency*

It can be seen from Figure 4 that the higher frequencies got attenuated in accordance with how a low-pass filter works.

Filter Implementation

For implementing the filter, the MATLAB butter() function was used. It required that the cut-off frequency was inputted as a percentage of half the sampling frequency. Since in the digital domain, the angular frequency is used, the cut-off frequency was taken as a percentage of π. The resulting numerator and denominator polynomial coefficients were passed into the filter() and the resulting data was used to calculate the DFT and the plot is shown in Figure 5.

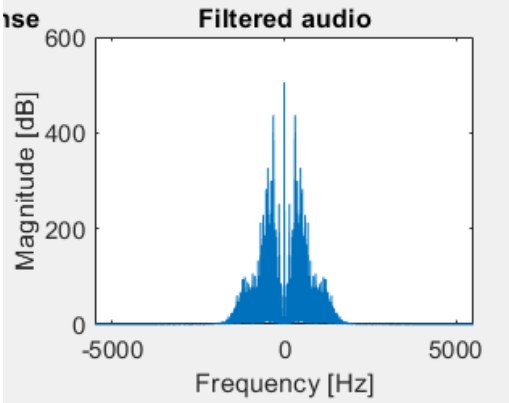


*Figure 5 – Filtered audio Magnitude vs Frequency*

Comparing Figure 1 and Figure 5, it can be seen that the higher frequencies have all been filtered out and listening to the audio shows that the hissing sound (noise) has been reduced enough that the words can be clearly made out!

*Calibration*

An initial frequency of 800 Hz was chosen for the pass band and a frequency of 2000 Hz was chosen for the stop band. The attenuation for the pass band was fixed at -1 dB and an initial attenuation of -40 dB was chosen for the stop band attenuation. The resulting audio was still noisy. Using the output from Figure 1, the stop band and pass band frequencies were adjusted arriving at $\omega_p$ = 1073 Hz (End of the passband) $\omega_s$ = 2267 Hz (Start of the stop band) and using the output from Figure 2, the attenuation of the stop band was adjusted arriving at -80dB producing a clear audio.

Conclusion

This project was one of the most fun projects I have done because it is closely tied with something I really enjoy, music! It gave me a chance to filter out noise from actual audio and the end-product was amazing for me. Not only this, I have always loved applying theory to real world scenarios (through Embedded Systems), but now, I might have found a new field of interest through Signal Processing! Thank you for an amazing semester of learning.

Extra Credit

The audio clip is from the Movie Airplane, 1980.

References

Dr. Mitchell, 2021. *Lecture 23 Video – Butterworth Filter Design Example*. YouTube. https://www.youtube.com/watch?v=UidVGAriyJI&feature=youtu.be