CSE 3313 – 001

Filter Design Project

Nikolas Murguia

December 7, 2020

Introduction

In this project, I was given an audio file that contained noise in the frequencies above 2.5khz. I designed a Butterworth filter to remove this noise and be able to output an unfiltered version of the audio clip. This was done through signal analysis of the source signal’s DFT and designing the filter around that.

Step 1: Audio Frequency Analysis

a. I began the project by using the *audioread()* command to provide the sampling frequency Fs and read the data from ‘noisyaudio.wav’ into an array y.

b. I then used the *fft()* command to find the DFT of the audio sample.

c. The next step was to form the axis for the DFT and following plots, which I did by using the *linespace()* command and spanning the data from -Fs/2 to Fs/2.

d. I then plot the DFT vs frequency centered around w = 0. I did this by using the *fftshift()* command. The plot was symmetrical as expected and showed the audio file having noise past the 2.5 kHz point.

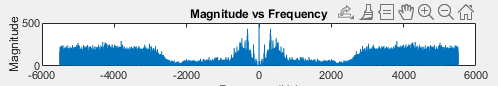


Figure 1

e. The next step was to plot the normalized log plot of the DFT where I could begin to conduct an analysis.

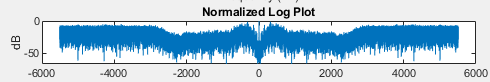


Figure 2

Step 2: Filter Design

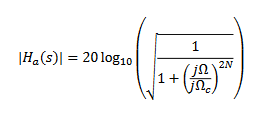
f. Decided on digital frequency wp as 2100 Hz and ws as 2500 Hz.

g. ATp was set as -1dB and the minimum attenuation for the stopband was decided to be -50dB. I decided this based on the plot done in part e shown above.

h. I then used the Butterworth filter design as shown in class. I calculated the ks=1/(10^(ATs/20))^2 – 1 and kp = 1/1/(10^(ATp/20))^2 – 1. I calculated the order N using the provided formula 0.5\*log(ks/kp)/log(ws/wp).

i. I then found the cutoff frequency for the filter using the provided formula COFreq=wp/10^(kp)^(1/2N))

j. I then calculated Ha and plotted the logarithmic gain of the frequency response.



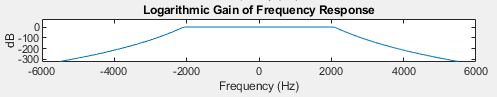


Figure 3

Step 3: Filter Implementation

3a. Utilize the MATLAB *butter()* command to design the digital filter. The parameters of the command is the cutoff frequency Wn = ΩC/ 0.5\*FS and order N found in part 2.

3b. I then used the coefficients a & b generated by the *butter()* command as parameters for the *filter()* command to create the filtered audio sample.

3c. The next step was to use the *fft()* command to calculate the DFT of the filtered audio which I then plotted just as I did with the unfiltered audio DFT

3d. Comparing the unfiltered and filtered plots, I noticed that the noise after the 2 kHz mark was removed from the filtered plot. This meant that the speech that I wanted to maintain was kept and the noise was successfully removed.

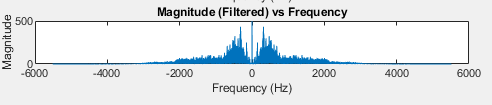


Figure 4

3e. I then used the *sound()* command to play both the unfiltered and filtered audios to hear the difference. The noise on the filtered audio was significantly less.

3f. I then outputted the new filtered audio using the *audiowrite()* command and saved it as ‘filterdaudio.wav’

3g. The audio clip is from the movie *Airplane!*

Conclusion

The project overall allowed me to not only demonstrate my understanding of the Butterworth filters, but also utilize the filter in a meaningful way. During the project I broke down the source signal and displayed the various steps through plots. This allowed me to clearly visualize what and when actions were occurring. Aside from learning much about audio analysis and filter creation, I also learned how to utilize Matlab to achieve this filtering process.