CPE 381: Fundamentals of Signals and Systems for Computer Engineers.

Project

Programming Assignment Phase 2.

Submitted by: Dan Otieno.

Due Date: 04/17/23

Introduction:

The purpose of this project is to work with and understand sample processing of signals. We use MATLAB and C/C++ to analyze a sample, which is provided as a voice recording tentatively set at 14 seconds in length. The final 7 seconds of the audio signal is modified in MATLAB and we use C++ to analyze the modified file and output an analysis summary in a text file. For the second phase, which is the topic discussed in this report, the requirements are to employ filtering to eliminate the unwanted frequency at 1900Hz.

Results & Observation:

Assignment:

Description:

The assignment instructions are to write a program in C/C++ to read our file that is modified from phase 1 of the project. Once the file is read, we check the sampling frequency from the file header and select the appropriate filter for that specific frequency. Finally, we process the signal according to the sampling frequency of the record, and we measure the execution time of the program. We also use Matlab to perform spectral analysis of both the modified and filtered signals.

Solution Methods:

I decided to use the bandpass filter for this portion of the project because my understanding was it would be best to use over a range of frequencies. Because I wanted to isolate a specific frequency range (in this case, up to 1900Hz) to isolate my frequency spectrum, I determined that using bandpass filter would be best where the selectivity matters. It would also help eliminate as much noise as possible across the range.

Per assignment instructions, the Filter Designer tool in MATLAB is used to create and generate a filtering set, which we export as a C header file and declare in our main source code. This is done for both 11,025Hz and 22,050Hz.

The Filter Designer configuration I used in MATLAB is as follows:

Figure 1: Design Configuration for 11,025Hz

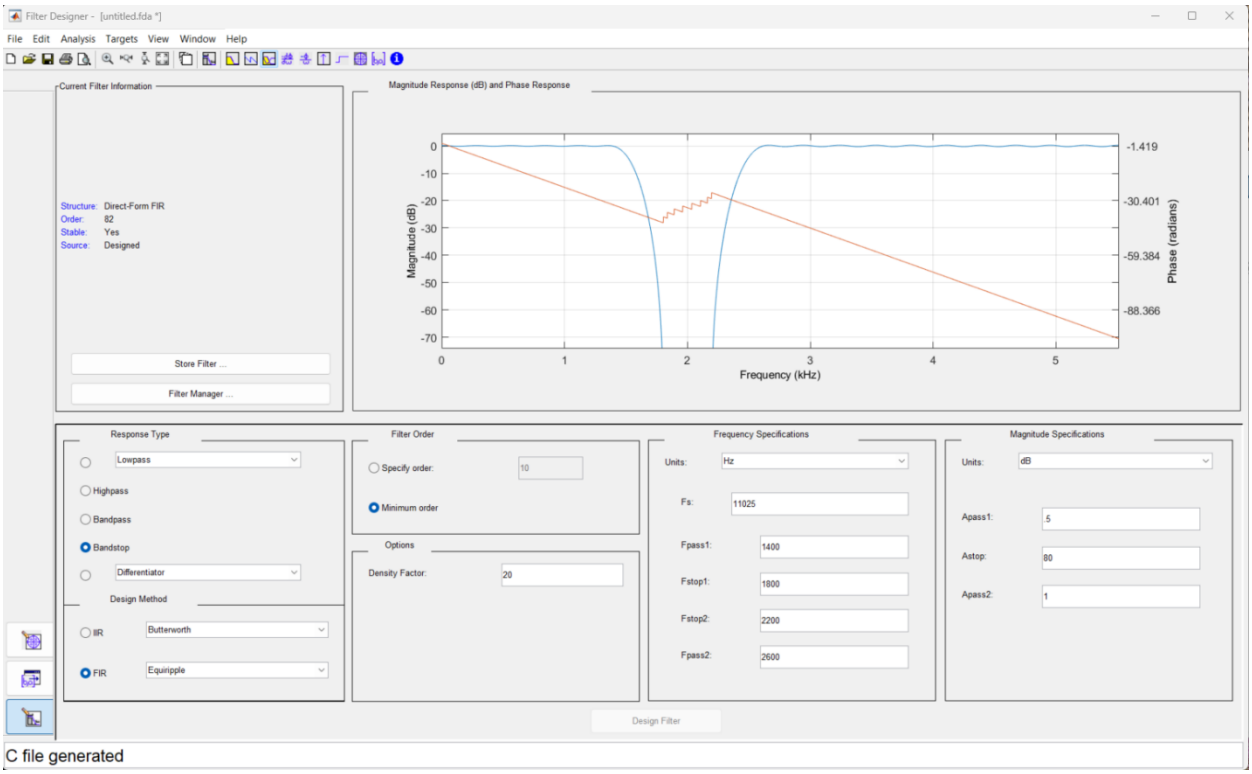
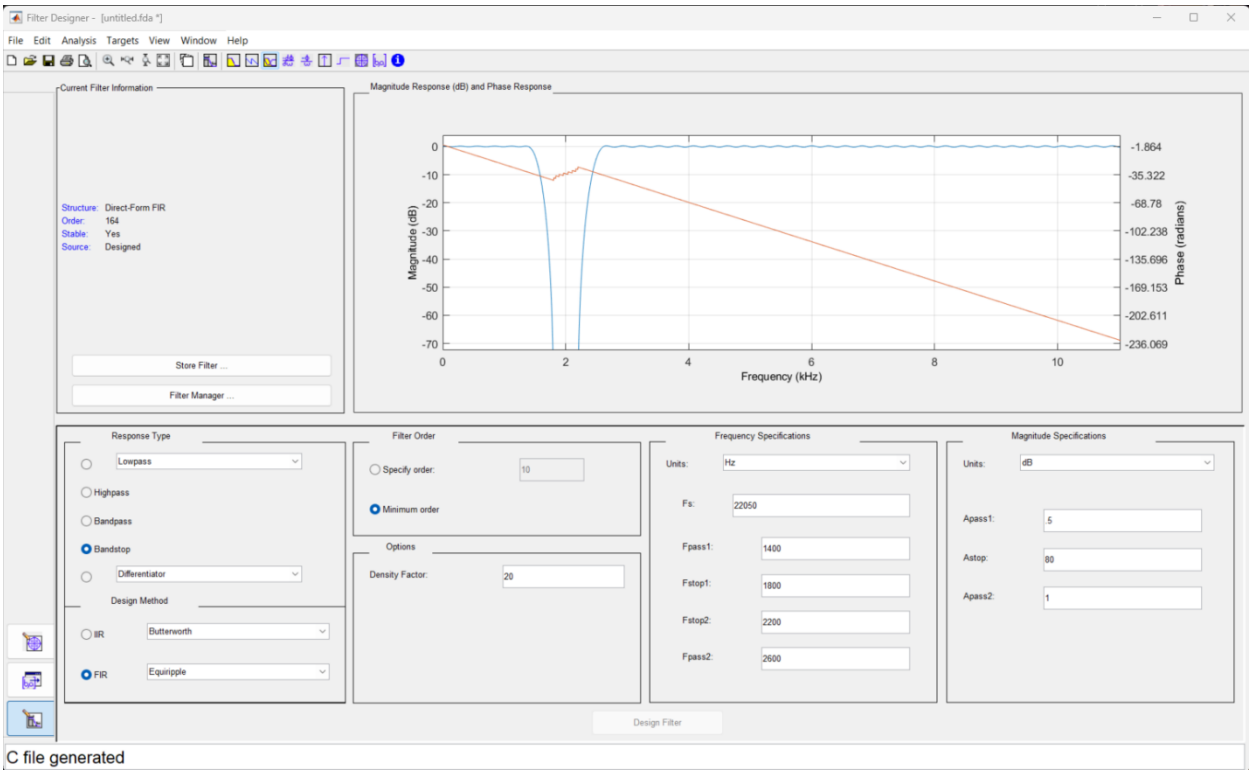


Figure 2: Design Configuration for 22,050Hz



For both designs, the Fpass range is from 1400Hz to 2600Hz, with Fstops at 1800Hz and 2200Hz, so there is a constant increment of 400Hz across the entire range. We are also advised to use an attenuation of 80dB (AStop configuration). I set my configuration that way to ensure I get at least 80 coefficients for my 11,025Hz filter set, and at least double that, 160 coefficients, for my 22,050Hz filter set, which I was able to achieve (see above screenshots).

Based on my understanding from the class lectures, FIR was the best option, IIR could be used, but there were additional elements to consider if using it, and I decided to go with FIR, which was a more straightforward approach from my perspective.

The C file generated from the tool is declared in my C++ source file to read the coefficients stored in an array during program execution.

Program Outputs:

Figure 3: Console Program Output.

```
dpo0002prac@DESKTOP-140DFI9:/mnt/c/Users/d_oti/Desktop/CPE_CLASSES/CPE_381/Project_phase II$ g++ Otieno_D_ph2.cpp -o oPhase2
dpo0002prac@DESKTOP-140DFI9:/mnt/c/Users/d_oti/Desktop/CPE_CLASSES/CPE_381/Project_phase II$ ./oPhase2
Enter input filename: Otieno_D_mod.wav
Reading from file path, please wait....
Size of File: 617444
-------------------------------------
Size of Header(bytes): 1
-------------------------------------
Bits per Sample: 16
-------------------------------------
Bytes per sample: 2
-------------------------------------
Samples: 308718
-------------------------------------
Sample Rate: 11025
-------------------------------------
Duration of Program execution: 0.199442secs.
Processing complete ... exiting Program....
dpo0002prac@DESKTOP-140DFI9:/mnt/c/Users/d_oti/Desktop/CPE_CLASSES/CPE_381/Project_phase II$
```

From the above console output, we can see the program execution time of 0.199secs, or roughly 0.2secs, this is attainable in real-time.

Spectrum Analysis Configuration and Outputs:

The instructions for spectrum analysis are to analyze the input wav file (modified) and the filtered wav file for t<7s and t>7s. Below are the outputs:

INPUT WAV FILE (s1):

Figure 4: Input WAV file Spectrum for t<7

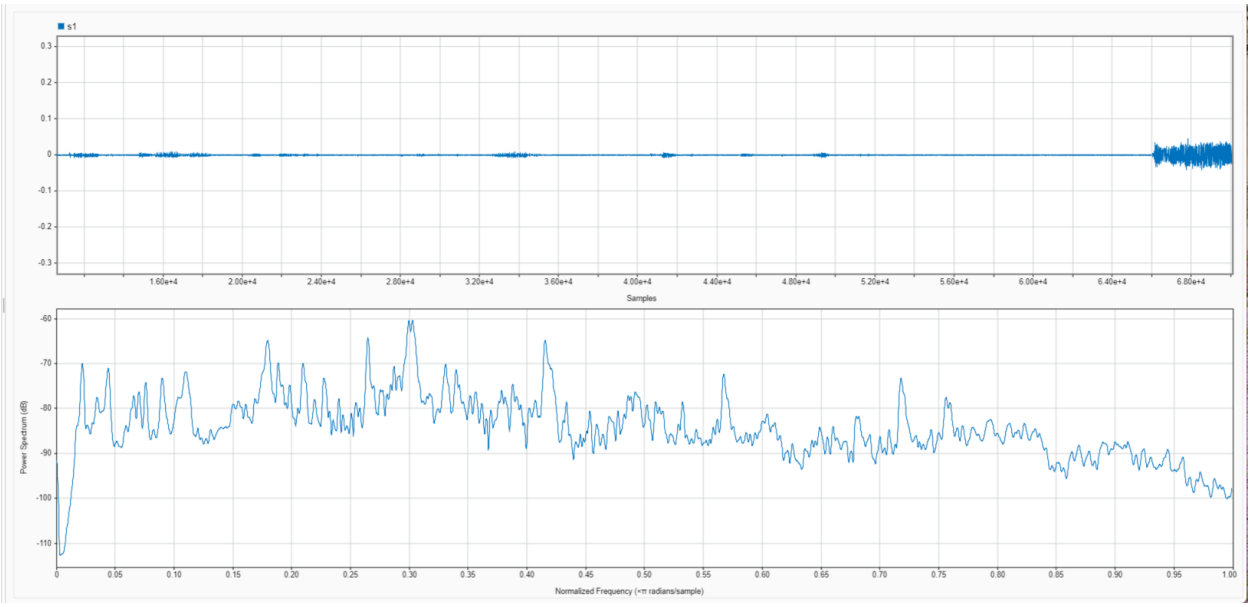
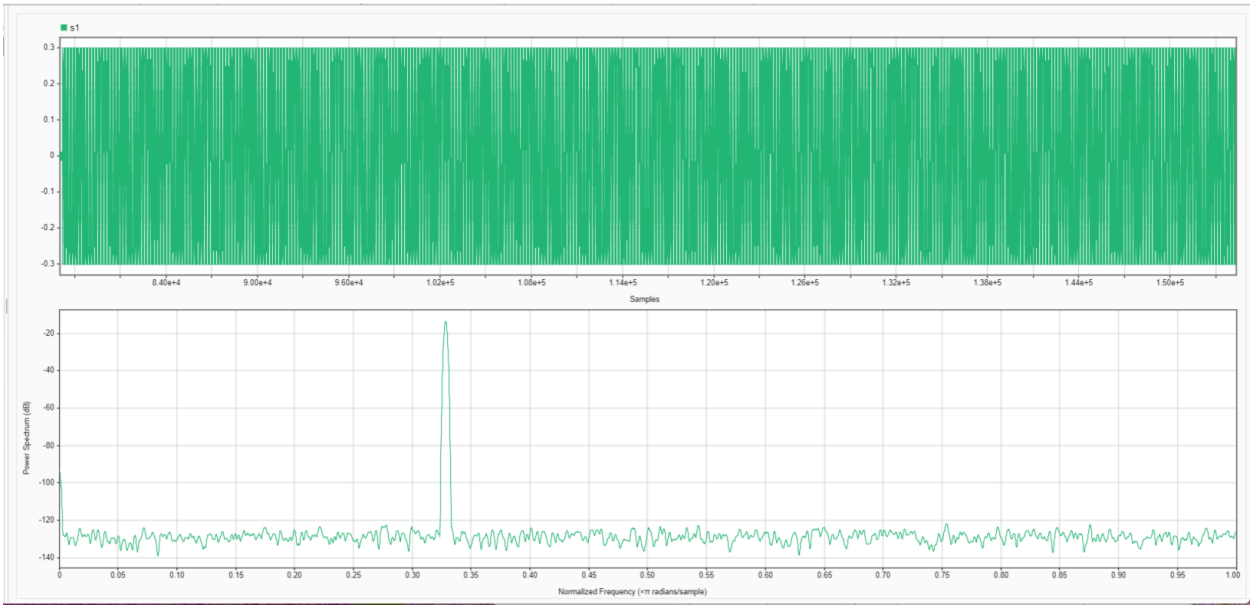


Figure 5: Input WAV file Spectrum for t>7



FILTERED WAV FILE (s2):

Figure 6: Spectrum for filtered WAV file - t<7

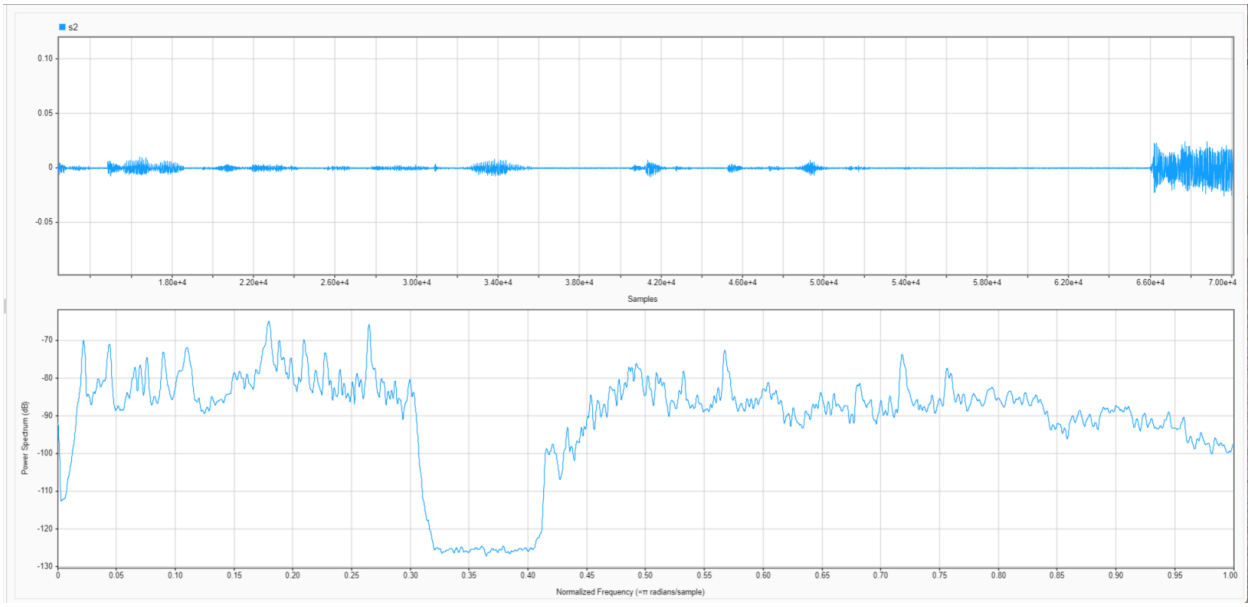
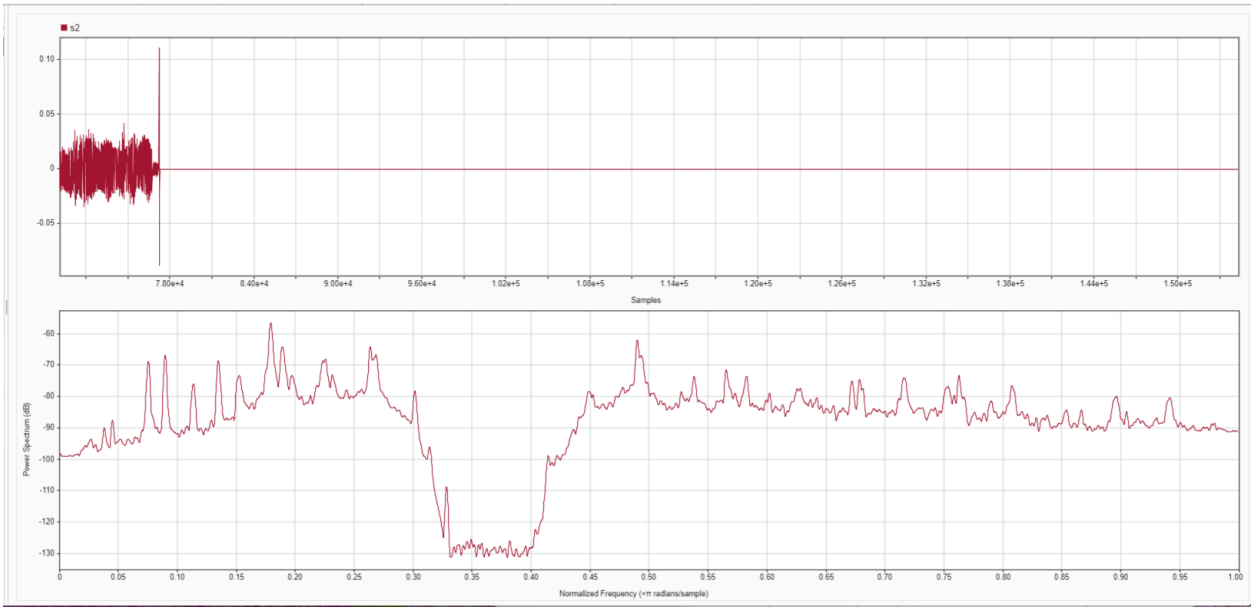


Figure 7: Spectrum for filtered WAV file - t>7



Conclusion:

This project provided a great insight into how filtering works, and the reasoning behind employing them in signal analysis. We can analyze and modify signals to reduce or eliminate noise using the powerful tools provided within the MATLAB environment (I didn't know about the Filter Designer prior to working on this project). The entirety of this activity, from phase 1 to this point, provided a lesson in signal processing, and helped me understand the concepts we learned in class, sampling being another, by applying them in a real-life scenario, I believe this is something I will be interested in exploring further.