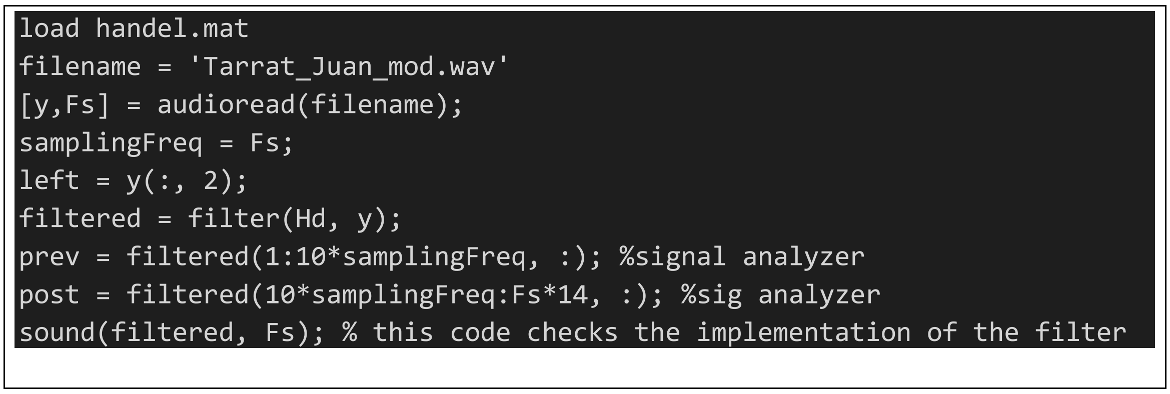
Project Phase II Report

1) Problem Description

This phase consisted of manipulating a recorded signal using both MATLAB and C++. We used a modified audio used in a previous phase of the project that had a sine wave added at the end and pour objective was to remove it using a filter. I used a Band stop filter since I considered it is the best suited for our purpose.

The filtered wave was then exported to another WAV file that is the result of the project for this semester. In our final audio we can hear the original audio with added static noise (from Phase I) and just static noise from second 10 till the end.

Below is the MATLAB code I used to split the audio into two halves in order to perform an analysis of the spectrum of the signal.



```
load handel.mat
filename = 'Tarrat_Juan_mod.wav'
[y,Fs] = audioread(filename);
samplingFreq = Fs;
left = y(:, 2);
filtered = filter(Hd, y);
prev = filtered(1:10*samplingFreq, :); %signal analyzer
post = filtered(10*samplingFreq:Fs*14, :); %sig analyzer
sound(filtered, Fs); % this code checks the implementation of the filter
```

Figure 1: MATLAB script

2) Filter

I decided to generate a band stop filter that would cut all the components of frequency 2200. My Astop is 80, my Fpass1 is 2050, my Fstop1 is 2150, my Fstop2 is 2250 and my Fpass2 is 2350 and it is a FIR.

The reason I chose a FIR filter was to take advantage of the easier implementation it has on code since we do not need to generate a memory buffer for past outputs as well as its better stability compared to an IIRC. The reason for my other parameters is clear. I wanted to get 2200 Hz components out of the file while trying not to change a lot the rest of it. I tried using a more constrained interval, but the order of the filter increased a lot, and it was impossible to get a program that at least was close to operate in real time. This filter generated 650 coefficients.

Figures 2 and 3 show the magnitude and phase spectrums for both filters designed.

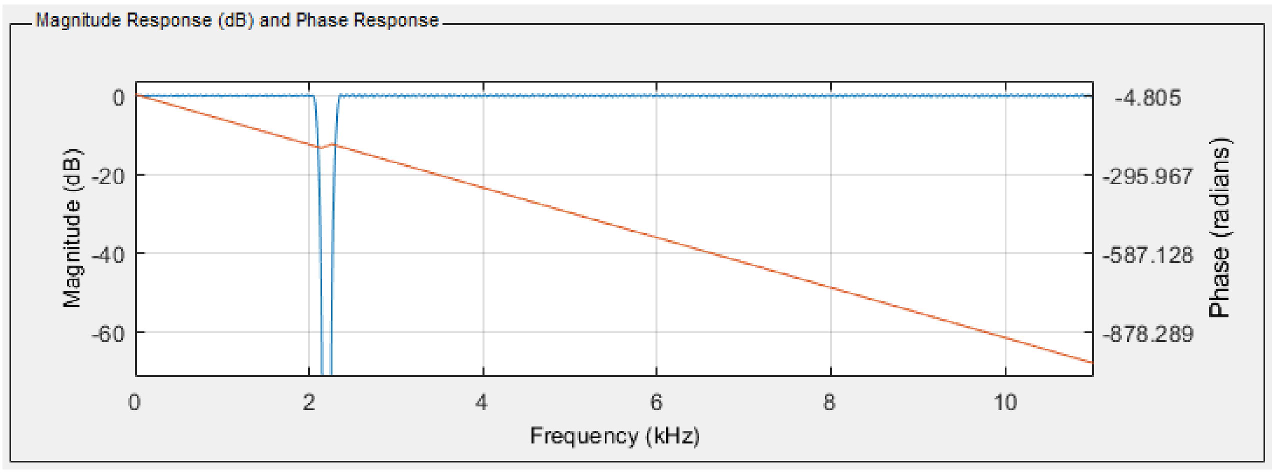


Figure 2: 2050 filter

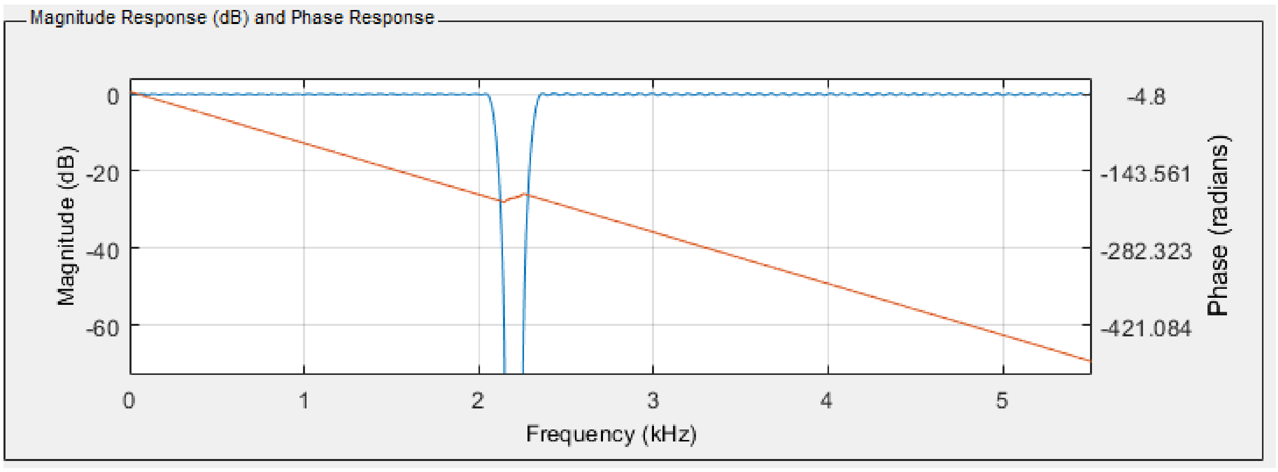


Figure 3: 11025 filter

In the C++ program I first choose the filter I will be using in the program based in the sampling frequency extracted from the header. Knowing this, I create a pointer that points to an array of doubles with the size of the array of the filter coefficients. I also create arrays for x and y in the filter for both left and right channels. Those arrays are then passed into a function that is very similar to one provided in some of the slides that performs the data manipulation of the audio and then is written to the new file. This is done sample by sample, so the flow is: get data from input file, manipulate it and write it in output file.

3) Spectrums

Figures 4 and 5 shows the input signal and its spectrum. Recall the peak at 0.2 that belongs to the sine wave.

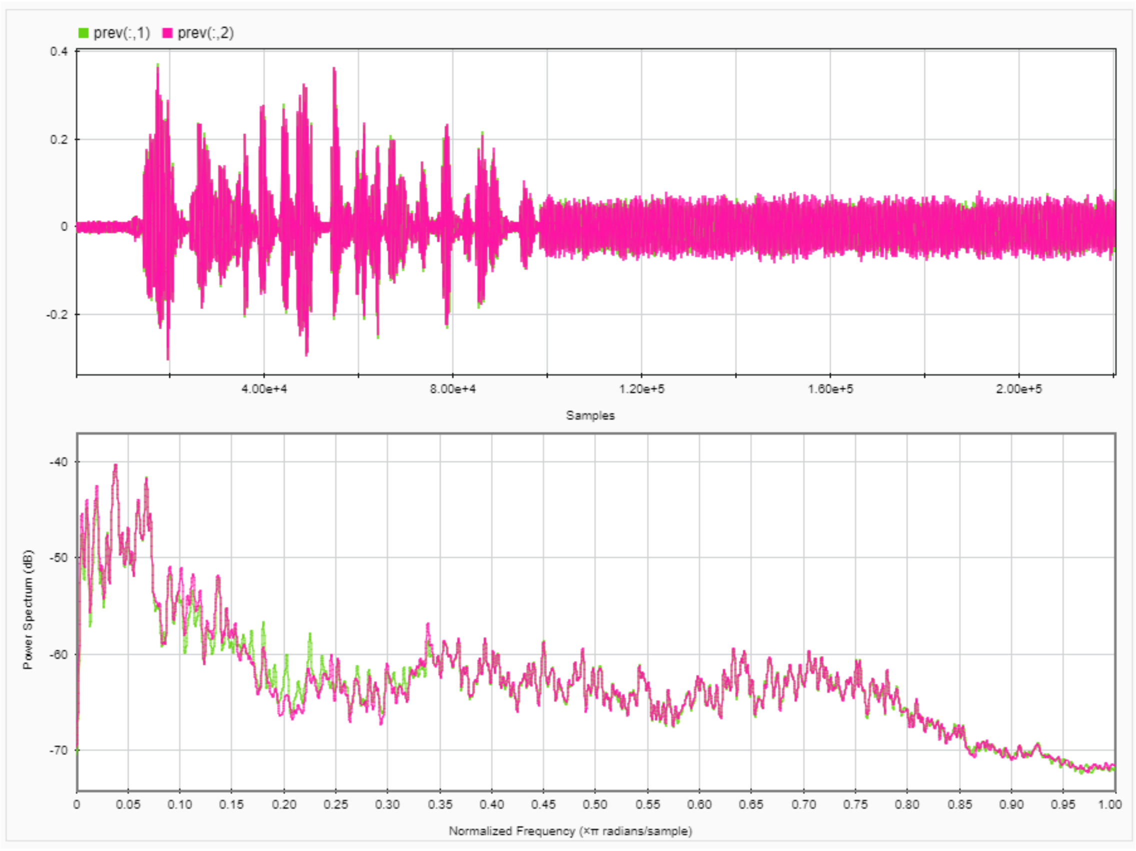


Figure 4: Input signal t <10

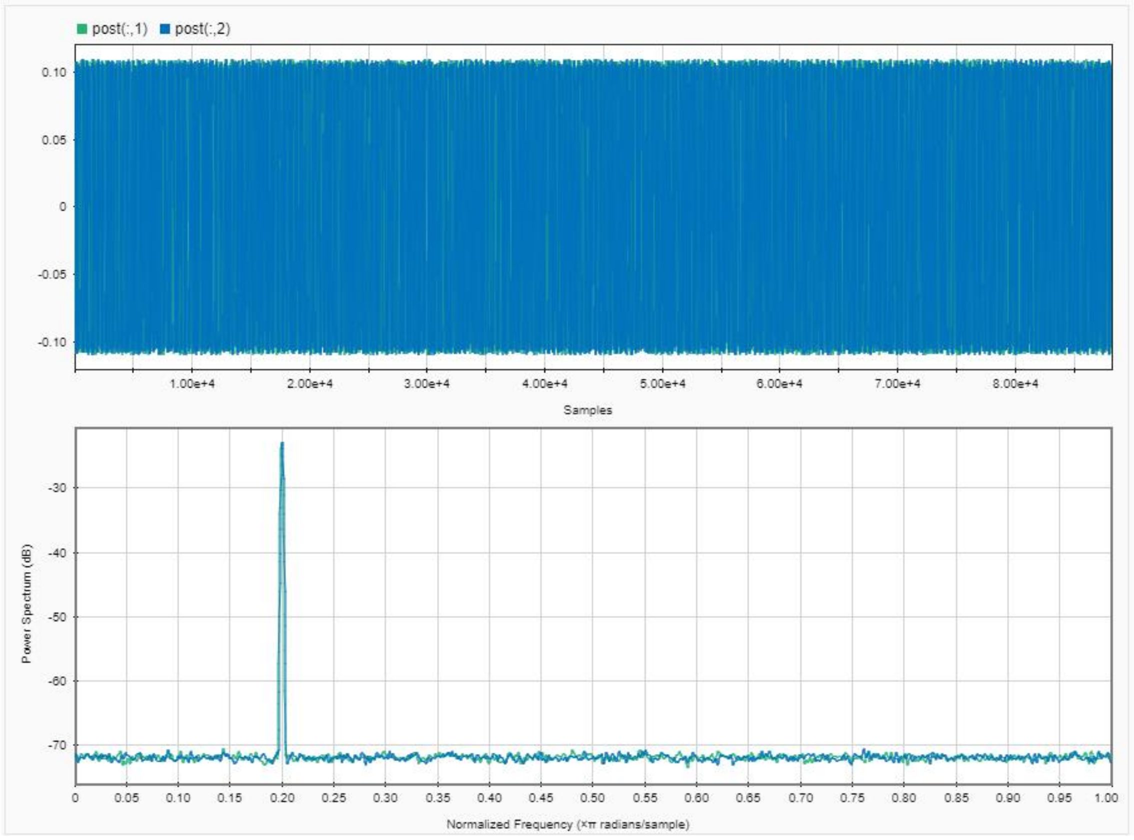


Figure 5: Input signal t >10

Figures 6 and 7 show the spectrums for the filtered signal. Figure 5 belongs to t < 10 and Figure 6 belongs to t > 10s

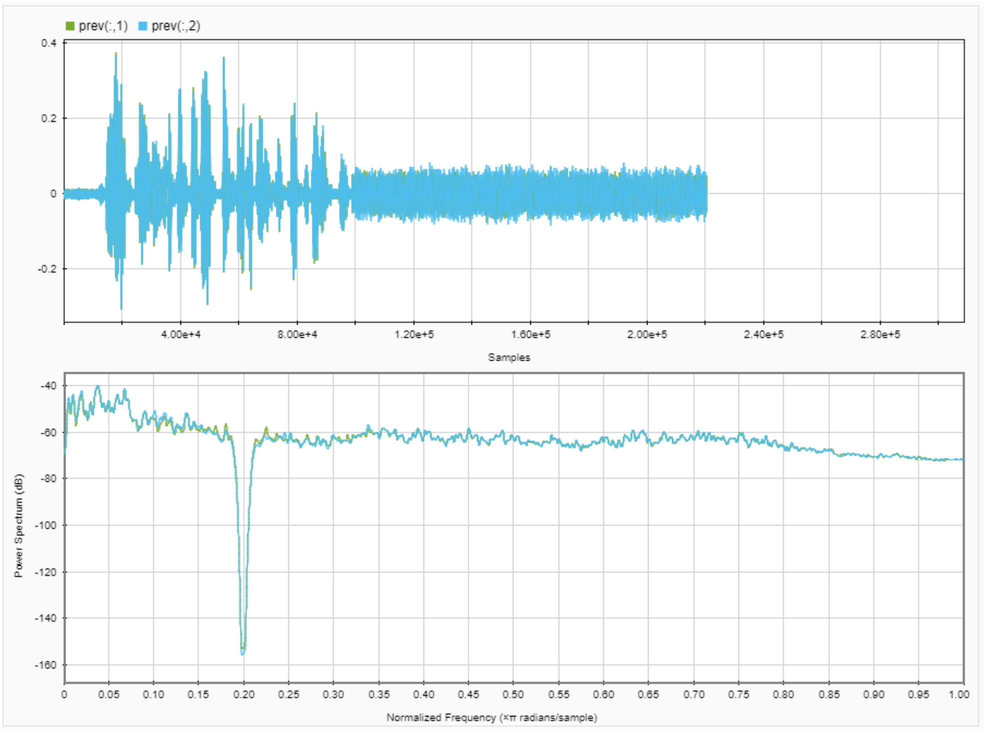


Figure 6: t < 10

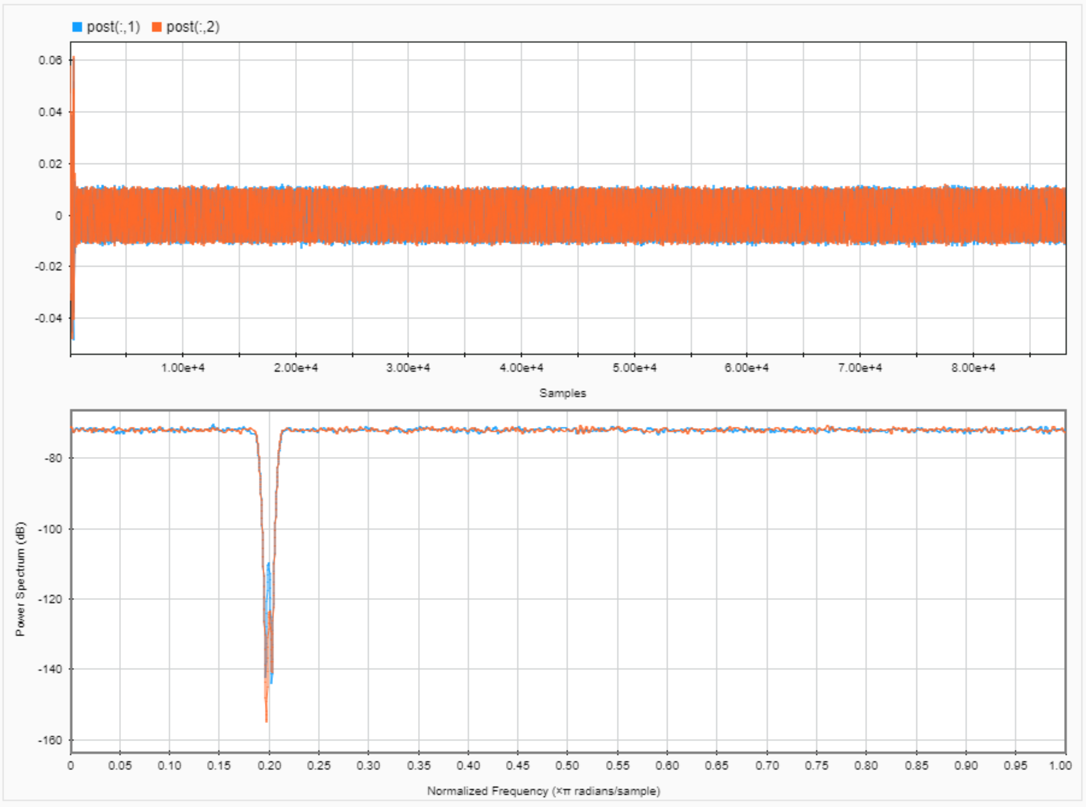


Figure 7: t > 10

4) Performance

When I run my program (the latest version) it takes about 2 seconds to execute (see console if you run it). This means that the program can perfectly work in real time, I have had some problems regarding execution time. Most of it was due to the coefficients and due to the way I was handling the arrays with vectors. However, this final version looks much more efficient if we talk about performance.

5) Conclusions

This project has been very useful. Not only have I learnt how to manipulate signals and got a better understanding of filters, but I have also learnt a lot of programming skills that I did not know before. I liked it because projects like this one are probably close related to issues we as engineers will face some time in our professional career, and that is the most gratifying thing one can get from projects like this one.