Digital Signal Processing A.Y. 2019/2020
Homework 1

Donald Shenaj

December 6, 2019

Contents

Exercise 1

1.1 Cuckoo sound

A ornithologist recorded an wav file containing the cuckoo calls and other forest sound are confusing him. We want to help him designing a filter which can filter out the unwanted frequencies. The original audio is visibile in Figure 1.1.

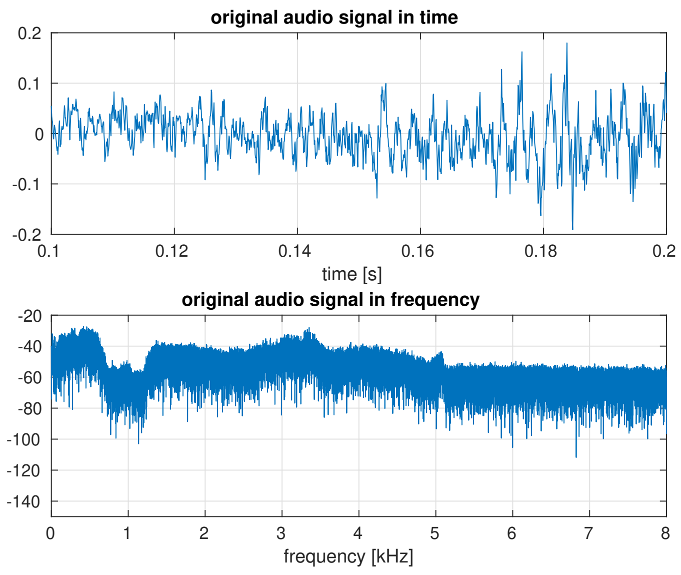

Figure 1.1: Forest sound

1.2 Filter design

We know that the cuckoo sounds are contained in the frequency range from 640 to 1280 Hz, so this would be our pass band frequency range for the Type I pass-band filter. The transition bandwidth is given, and it should be 160 Hz. The order of the filter N should be even for the choosen filter and in particular it has to be 100. I've designed the filter using both linear programming approach and Remez algorithm, in order to see if there is any difference. For both filters i've defined the weighting function using the value 10^{-2} in the pass-band and 10^{-3} in the stop-stop band. The results obtained can be seen respectively in Figure 1.2 and Figure 1.3, and for both we can easily see that our constraints are statisfied.

1.3 Filter response

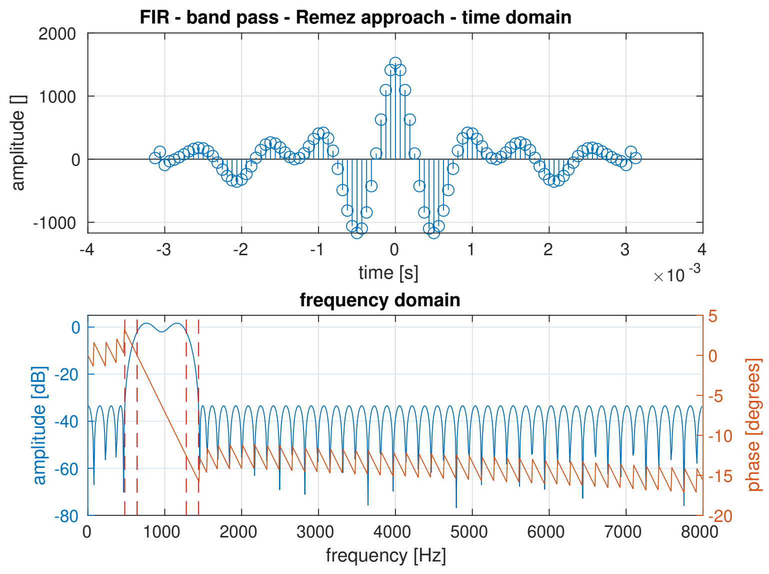

Figure 1.2: Remez algorithm

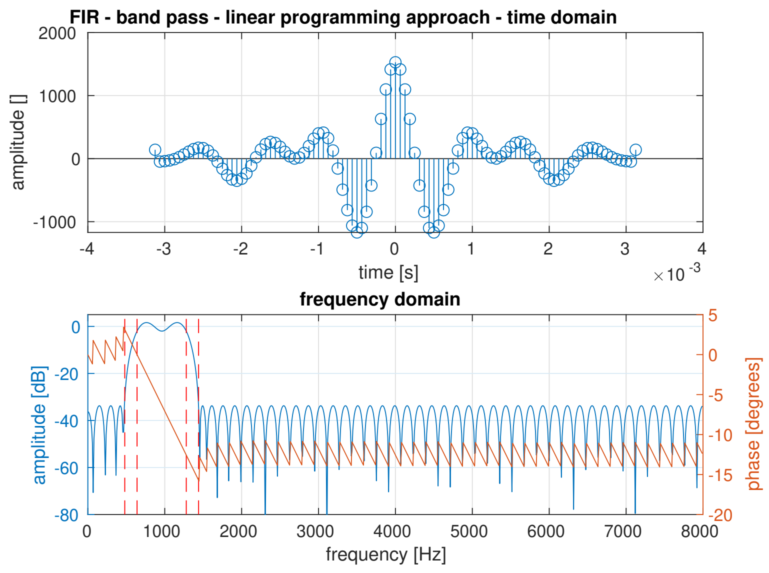

Figure 1.3: Linear programming approch

1.4 Filtering audio signal

After designing the filters, we filtered the signal for example with the linear programming one, and the result can be viewed in the Figure 1.4. As we can see the audio in the frequency range from 640 Hz to 1280 Hz is now magnified, as we wanted. To check if the audio is actually filtered correctly it was produced a wav file called 'donald_shenaj_hm1.1.wav', which confirms the correct behaviour.

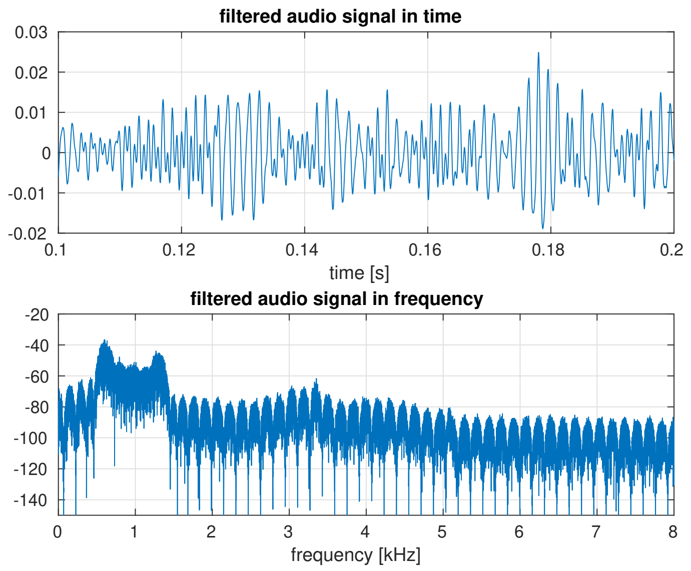

Figure 1.4: Filtered audo signal, cuckoo sound waves

1.5 Type II

For designing a Type II FIR filter i've choosen to use the linear programming approch, which gives us more control on the features, using $N = 101$ the same weighting function as before, as well as the stop-band and pass-band frequencies. The coefficients of the filter can be seen in the Figure 1.5. We can tell that the Type II filter can be used for filtering the audio signal because it's responses (both frequency and time) are adeguate and very similar to what we've found with the Type I FIR filters designed with linear progamming and Remez.

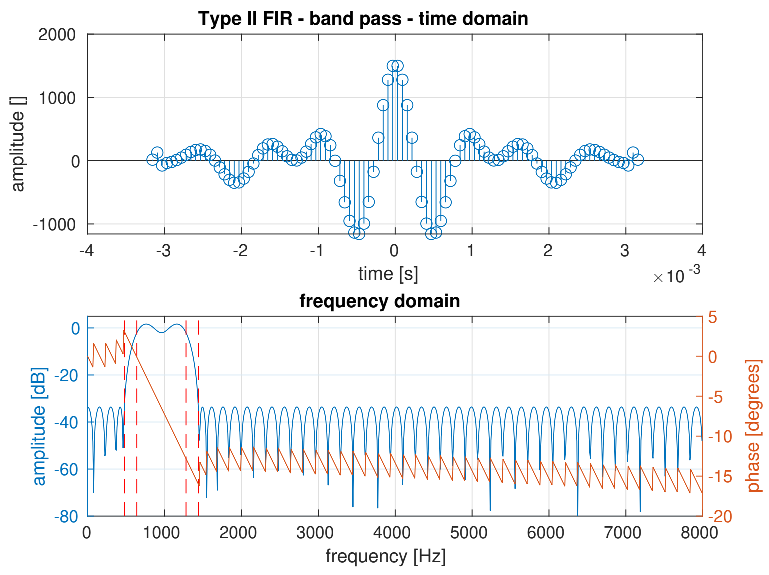

Figure 1.5: Type II pass-band filter using linear programming

Exercise 2

2.1 Introduction

We have to design a *derivative filter* with the following frequency response:

$$H_{\text{ref}}(f) = \begin{cases} i2\pi f, & |f| \leq (1-\alpha)B \\ 0, & |f| \geq (1-\alpha)B \end{cases}$$

We'll consider the case $B = 3\text{kHz}$, $\alpha = 0.05$, and $T = \frac{1}{F_p}$, with $F_p = 8\text{kHz}$.

2.2 Filter design

As we learned during lectures, we can tell that an advisable filter could be for example a Type III FIR because $H_0(f)$ should has odd simmetry, and we can choose $N = 100$ since it has to be an even number. In particular, the reference form of the filter is shown below:

$$H_0(f) = 2T \sum_{n=0}^{\frac{N}{2}-1} h(nT) \sin\left(2\pi f \left(\frac{N}{2} - n\right) T\right)$$

I've designed the filter using both linear programming and Remez algorithm.
For the linear programming approach i've defined a grid of frequencies, from 0 to $(1-\alpha)B$ which is the pass-band, and from $(1+\alpha)B$ to $\frac{f_p}{2}$ which is the stop-band, where i've determined his behaviour. For the first part i've sampled the different values of the function $2\pi f$, and i've set to 0 the values to the remaining frequencies. We can 'forget' about the i complex number for the amplitude of the filter.
For the Remez approach i've used the usual **firpm()** function, adding a new parameter, called 'derivative'. In this case we don't have to give the reference form of H_0, instead we have to give the last value that our filter assumes in the pass-band, which is $2\pi(1-\alpha)B$, and 0 in the other amplitude entries.

2.3 Filter response

The resulting filter in time and frequency is shown in Figure 2.1 for the linear programming solution, and in Figure 2.2 for the Remez solution. To better understand his behaviour in Figure 2.3 is shown the frequency response for both filters in linear scale, compared with the reference form of $H_{\text{ref}}(f)$. We can see clearly that the desired behaviour for the filter is properly obtained for both the implementations of the filter. The main difference between the two filters is that due to the implementation they have inverted values of $h_0(t)$ so there would be a phase of 180 degrees between their results.

2.4 Testing the filter

For testing the filters we can pass in input for example a sine wave with frequency in the pass-band and see the output of the filters. We expect to have the derivative of the input signal, so a cosine wave, which would be magnified according to the frequency of the sine by the factor $2\pi f$, and retarded by the phase of the filter in the corresponding frequency of the input sine. The result (output) for both filters can be seen in the Figure 2.4, obtained by passing in input a sine of 300Hz.
We see that the two filters produce a cosine wave with amplitude $2\pi 300$ as it should be, which is retarded from the input sine by the value of $angle(H_0(300))$. The results between the two filters as we expected are inverted by 180 degrees.

2.5 Smallest order of N

In order to find the smallest order of N that guarantees a stop-band attenuation of at least 40dB, i've tried many decreasing values of N and determined the stop-band attenuation for each N. It can be easily automated with a for loop which decreases

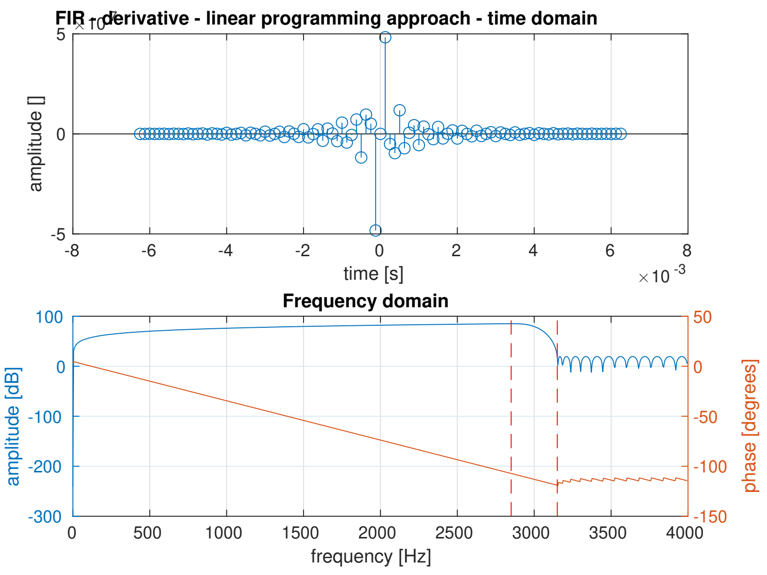

Figure 2.1: Type III derivative filter using linear programming

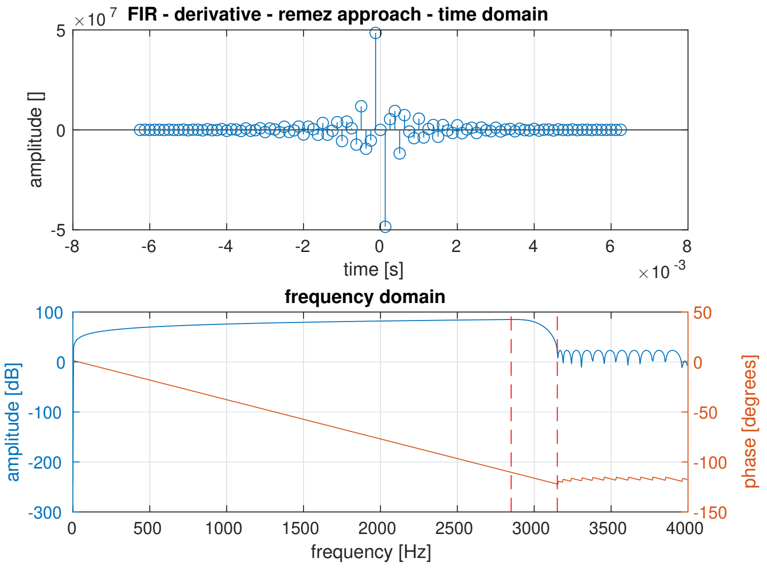

Figure 2.2: Type III derivative filter using Remez

N by 2, but since it has to be solved only one time in the final code i've prefered to leave just the solution for $N = 54$, which is the smallest order of N that we were searching for. The filter with smallest N found that satisfies our costraints can be seen in Figure 2.5.

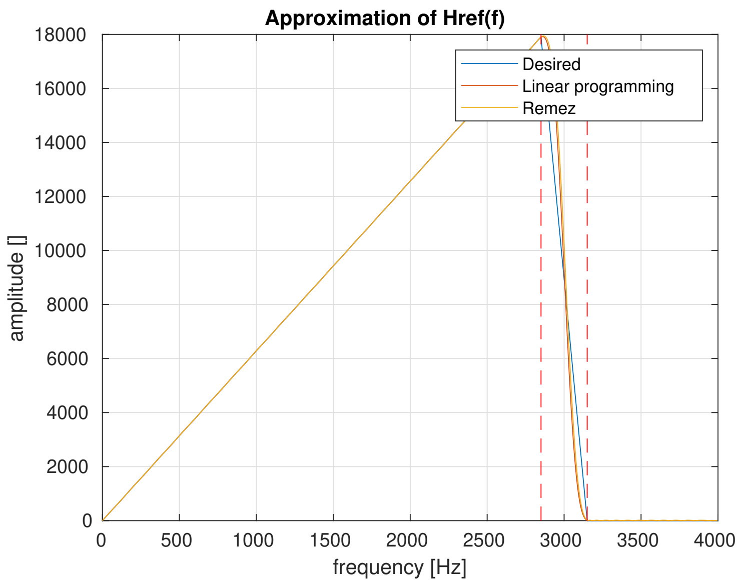

Figure 2.3: Comparing frequency behaviour of the filters with the ideal response

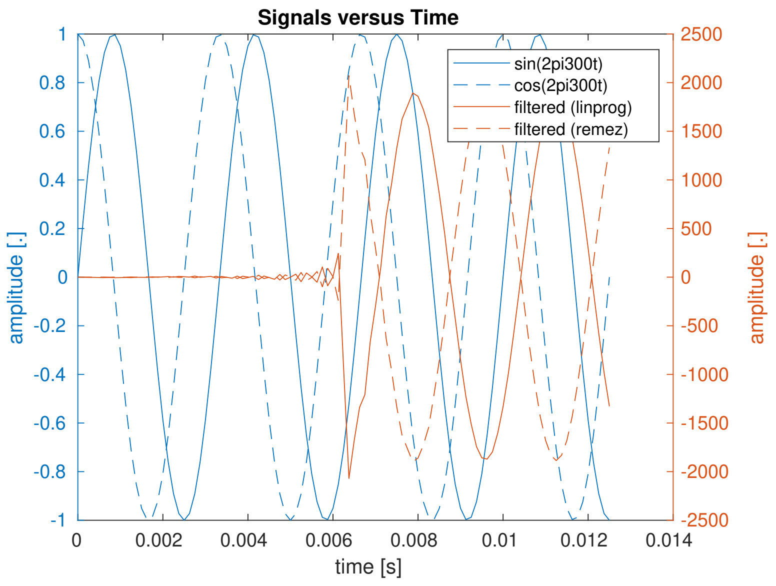

Figure 2.4: Testing the filters with a sine wave

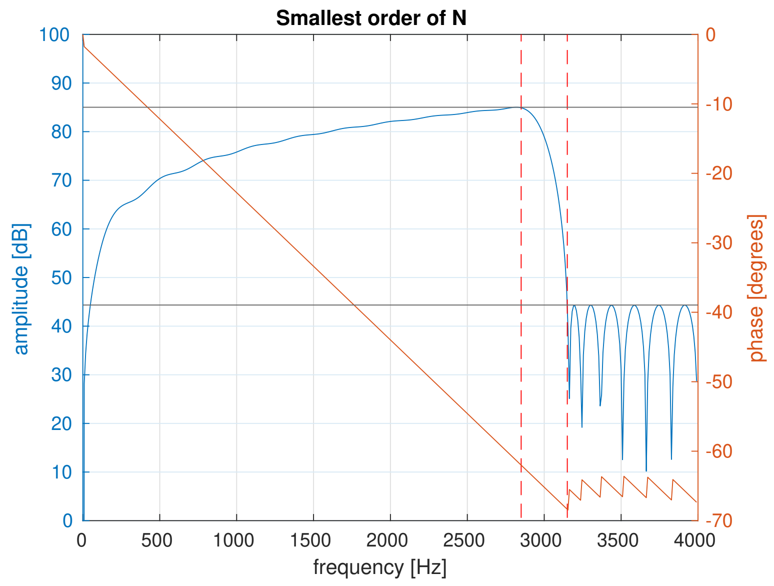

Figure 2.5: N = 54, 40.672 dB stop-band attenuation

Exercise 3

3.1 Introduction

Since my student's code is 1238939 i've to use the audio file called 'hw1_peter_and_the_wolf_9.wav'.

3.2 Filter design by windowing

We want to implement a filter with frequency response:

$$H(f) = \frac{(1 + i2\pi f T_1)(1 + i2\pi f T_3)}{(1 + i2\pi f T_2)}.$$

The Fourier transform of $H(f)$ is know in closed form:

$$h(t) = \frac{A}{T_1}\exp\left\{-\frac{t}{T_1}\right\}1(t) + \frac{1-A}{T_3}\exp\left\{-\frac{t}{T_3}\right\}1(t)$$
$$A = \frac{T_2 - T_1}{T_3 - T_1}$$

Where $1(t)$ is the unit step function and $T_1 = 75us$, $T_2 = 318us$, $T_3 = 3180us$. We can design a very simple filter which uses a windowing technique by sampling the $h(t)$, for $t > 0$ with step $T = \frac{1}{F_p}$. For $t < 0$ instead $h(t) = 0$, due to the unitary step function suppression. In my implementation i've used $N = 100$, which led to 101 values of $h(t)$, using t from $-N/2$ to $N/2$. I've used a standard rectangular window, which is done by 'default' if we don't multiply for a window. In fact it's like we are usign a windows with all ones in the pass-band and zeros outside.

3.3 Filter response

The filter designed by windowing can be seen in Figure 3.1.

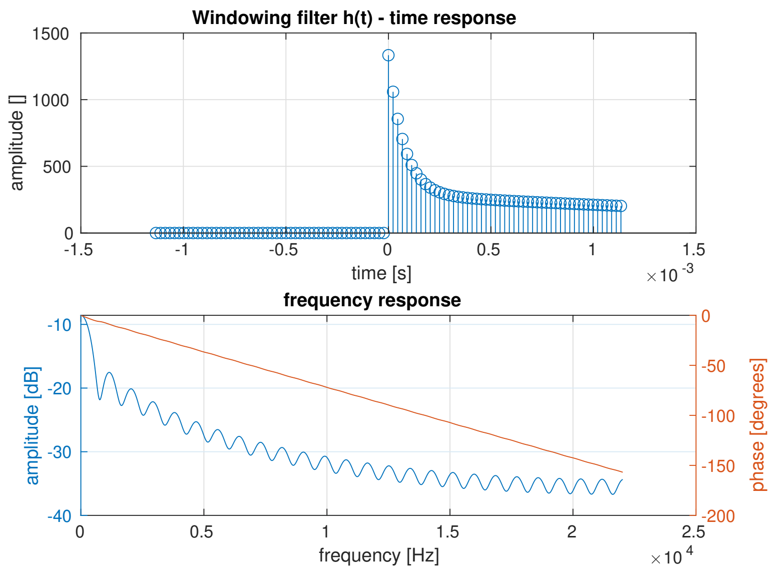

Figure 3.1: Filter design by windowing techninque

3.4 Filtering an audio sample

After applying the filter to the audio sample to see if the filter is working properly it was produced the audio file called 'donald_shenaj_hm1.3.wav'. We can hear easily that the filter is actually working, removing the higher frequencies and boosting the lower ones.

3.5 Advanced

Now we want to design a Type I FIR filter for approximating $R(f) = \text{real}(H(f))$ and a Type III FIR filter for $I(f) = \text{imag}(H(f))$, using a minimax procedure. So we have to express $H(f) = R(f) + iI(f)$, as shown in Equation (3.1).

For designing the Type I and Type III FIR filters i've picked $N = 200$. I defined the frequency grid from 0 to $\frac{F_p}{2}$ with step $F = F_p/N/3/32$ in order to have more values for the fitting. After that i've determined the behaviour of $R(f)$ and $I(f)$ over the grid of frequencies, which are the ideal filter shapes, and i've built the cosines and sines matrices for the linear programming solution as we've done before. The only difference in this case was the choice of the weighting factor. In order to improve the quality of the fitting we used $W(f) = |H(f)|$ to enforce a smaller error at higher frequencies.

$$
\begin{aligned}
H(f) &= \frac{1}{P(f)} \\
&= \frac{(1 + i2\pi f T_2)}{(1 + i2\pi f T_1)(1 + i2\pi f T_3)} \\
&= \frac{1 + i2\pi f T_2}{1 - 4\pi^2 f^2 T_1 T_3 + i2\pi f (T_1 + T_3)} \\
&= \frac{(1 + i2\pi f T_2)(1 - 4\pi^2 f^2 T_1 T_3 - i2\pi f (T_1 + T_3)}{1 + 16\pi^4 f^4 (T_1 T_3)^2 - 8\pi^2 f^2 (T_1 + T_3)^2} \\
&= \frac{1 - 4\pi^2 f^2 T_1 T_3 + 4\pi^2 f^2 T_2 (T_1 + T_3) + i2\pi f T_2 - i2\pi f (T_1 + T_3) - i8\pi^3 f^3 T_1 T_2 T_3}{1 + 4\pi^2 f^2 (4\pi^2 f^2 (T_1 T_3)^2 - 2T_1 T_3 + (T_1 + T_3)^2} \\
&= \left.\frac{1 - 4\pi^2 f^2 (T_1 T_3 - T_2 (T_1 + T_3))}{1 + 4\pi^2 f^2 (4\pi^2 f^2 (T_1 T_3)^2 - 2T_1 T_3 + (T_1 + T_3)^2)}\right\} R(f) = \text{real}(H(f)) \\
&\quad + i \left.\frac{2\pi f (T_2 - (T_1 + T_3) - 4\pi^2 f^2 T_1 T_2 T_3)}{1 + 4\pi^2 f^2 (4\pi^2 f^2 (T_1 T_3)^2 - 2T_1 T_3 + (T_1 + T_3)^2)}\right\} iI(f) = i * \text{imag}(H(f))
\end{aligned}
\tag{3.1}
$$

The time and frequency responses of $R(f)$ and $I(f)$ filter designed can be seen respectively Figure 3.2 and Figure 3.3. We can see thet they satisfy our assumptions about even and odd simmetry, and the shape of the frequency response.

Now we want to use both filters to 'build' $H(f)$. In a very simple way we can just add filter's time responses, and after that convolve with the original audio:

$$h(nT) = h_R(nT) + h_I(nT)$$

The time and frequency responses of the filter designed adding the real and imaginary part, using $N = 200$, can be seen in Figure 3.4. It should look similar to

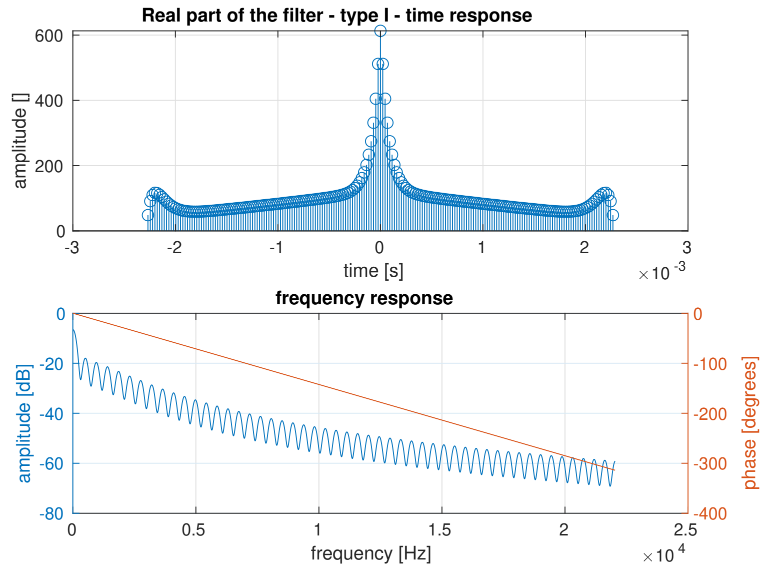

Figure 3.2: Type I FIR - $R(f)$

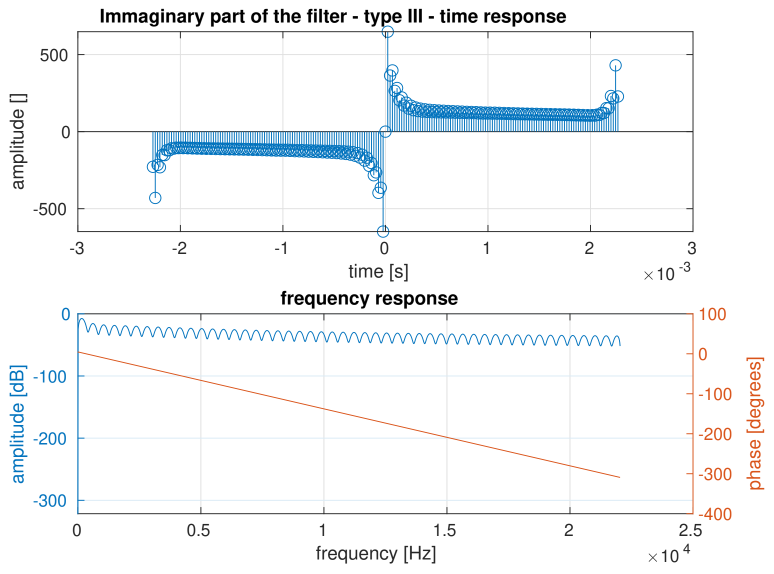

Figure 3.3: Type III FIR - $I(f)$

the filter designed by windowing and in fact the frequency response does correspond. It might seems that the time response is different but actually it is very similar, with a small difference for the zero valued amplitude.

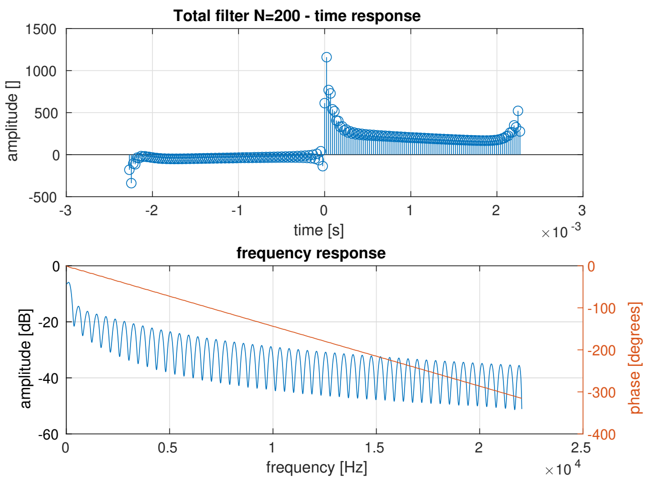

Figure 3.4: N=200

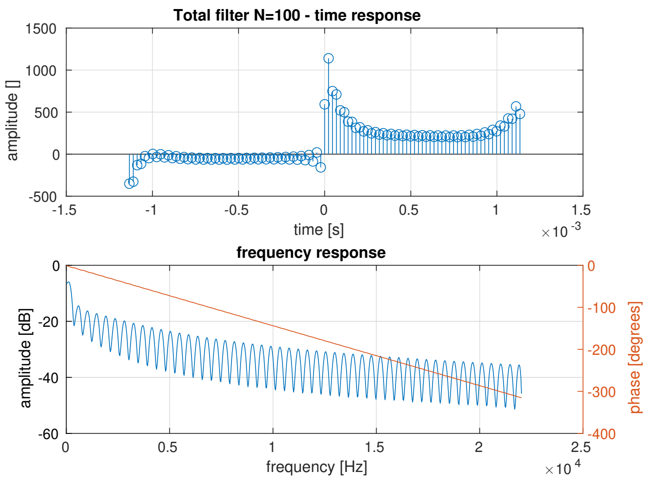

Figure 3.5: N=100

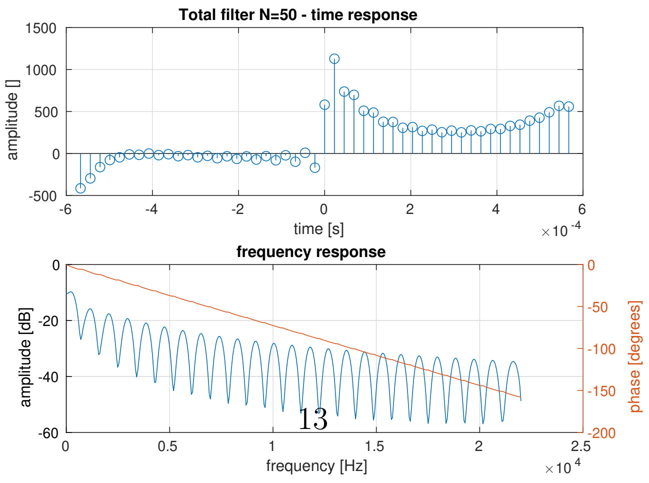

Figure 3.6: N=50

After designing the total filter $H(f)$ using $N = 200$ we considered the cases of $N = 100$ and $N = 50$. Their results can be seen respectively in Figure 3.5 and Figure 3.6.

In order to see which of the three N is the most appropriate we reported the Figure 3.7 were we compared the frequency behaviour, in a linear scale, of the filters with the ideal frequency response. Let's call d_R the ideal frequency response of the Type I filter, which implements the real part of $H(f)$ and d_I the ideal frequency response of the Type III filter, which implements the imaginary part of $H(f)$, then we compared the absolute value of the three filters with the absolute value of $d_r + id_I$. By looking at the picture we can say that with $N = 200$ we have the best fitting.

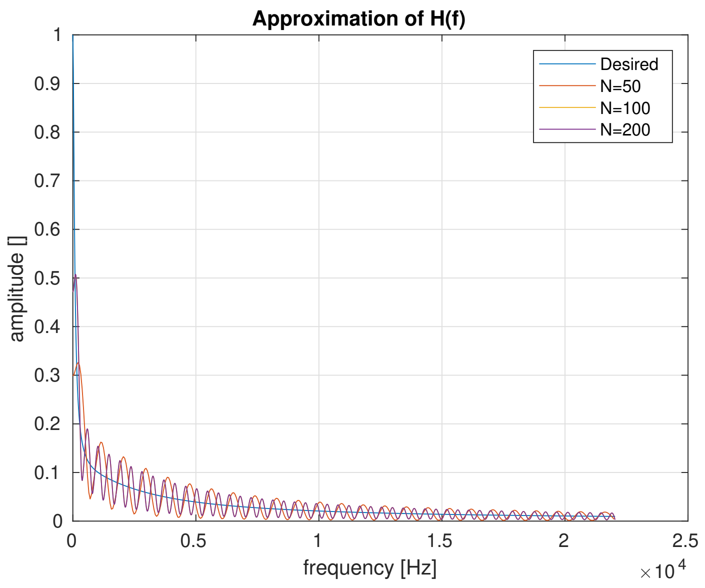

Figure 3.7: Comparing frequency behaviour of the filters N=50,100,200 with the ideal response

For concluding our analysis, if we listen the audio file called 'donald_shenaj_hm1.3_2.wav' produced by the filter obtained using $N = 200$, we can tell that it is working as expected. In particular it does improve what we found with the very simple filter designed with windowing techninque because we can hear better the frequencies involved.