Homework 1 DSP

Elena Camuffo 1234370

7 November - 8 December 2019

1 Exercise - The cuckoo sound

Firstly loading the signal, it results very noisy. Lots of birds' sound can be heard but not the one of the cuckoo. It needs to be filtered. As the cuckoo frequencies lay in the interval $[640Hz, 1280Hz]$, we are expecting to find out a band-pass filter which operates in that range of frequencies.

1.1 Linear Program - Type I

The pass-band of the filter is determined by the two frequencies set to $f_a = 640Hz$ and $f_b = 1280Hz$. Starting with a linear program approach we thus design a **Type I band-pass filter** setting the cutoff frequencies to:

$$f_{s1} = f_a - B_t/2 = 560Hz, \qquad f_{p1} = f_a + B_t/2 = 720Hz$$

$$f_{p2} = f_b - B_t/2 = 1200Hz, \qquad f_{s2} = f_b - B_t/2 = 1360Hz$$

where $B_t = 160Hz$ is the *transition bandwidth.* Consequently the reference behaviour is:

$$R(f) = \begin{cases} 0, & 0 \leq f \leq f_{s1} \\ 1, & f_{p1} \leq f \leq f_{p2} \\ 0, & f_{s1} \leq f \leq \frac{F_p}{2} \end{cases}$$

F_p is the sampling frequency extracted from the input audio source. The number of samples is $N = 100$ (*even* because type I) and the weighting vector is set as $\boldsymbol{w} = [10^{-3}, 10^{-2}, 10^{-3}]$ so that to the main region of the band-pass filter correspond the heaviest weights.
We proceed in performing the linear programming filter design. As the filter is of type I, $H_0(f)$ has to be built as $H_0(f) = V \cdot \boldsymbol{x}$ where matrix V and vector $\boldsymbol{x}$

are build in the following way:

$$V = \underbrace{\begin{bmatrix} 2T\cos(2\pi f \frac{N}{2}T) \\ 2T\cos(2\pi f (\frac{N}{2}-1)T) \\ 2T\cos(2\pi f (\frac{N}{2}-2)T) \\ \vdots \\ T \end{bmatrix}^T}_{k\times\frac{N}{2}+1} \qquad \boldsymbol{x} = \begin{bmatrix} h_0(0) \\ h_0(T) \\ h_0(2T) \\ \vdots \\ h_0(\frac{N}{2}T) \end{bmatrix}$$

As the weighting vector is NOT $\boldsymbol{w} = \mathbb{1}$, we use an empirical method to find the *stop-band attenuation* $A_s = -20log_{10}(\delta_s) = 34.1017dB$ and the *pass-band ripple* $R_p = 20log_{10}(\frac{H_{max}}{H_{min}}) = 0.77262dB$, setting:

$$\delta_s = max\{|H_0(f)|\}, \quad f \in [f_{s1}, f_{p1}] \cup [f_{p2}, f_{s2}],$$

where

$$H_{max} = max\{|H_0(f)|\}, \quad f \in [f_{p1}, f_{p2}],$$

$$H_{min} = min\{|H_0(f)|\}, \quad f \in [m_1, m_2]$$

and m_1, m_2 correspond to the first and the last maxima of $H_0(f)$, with $f \in [f_{p1}, f_{p2}]$. The result can be seen in figure 1.

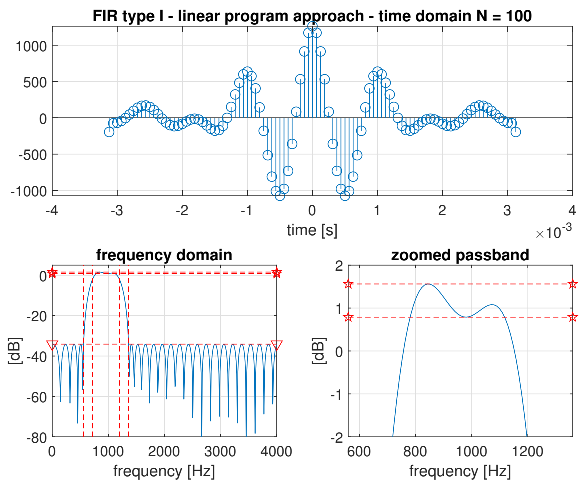

Figure 1: Type I filter designed with linear programming. $N = 100$, $\boldsymbol{w} = [10^{-3}, 10^{-2}, 10^{-3}]$.

1.2 Remez Algorithm

Afterwards we design the filter with **Remez algorithm**, leaving the parameters and the weighting vector set as before. Using the same procedure we determine the *stop-band attenuation* and the *pass-band ripple*, which result respectively $A_s = 33.9329dB$ and $R_p = 0.57538dB$.

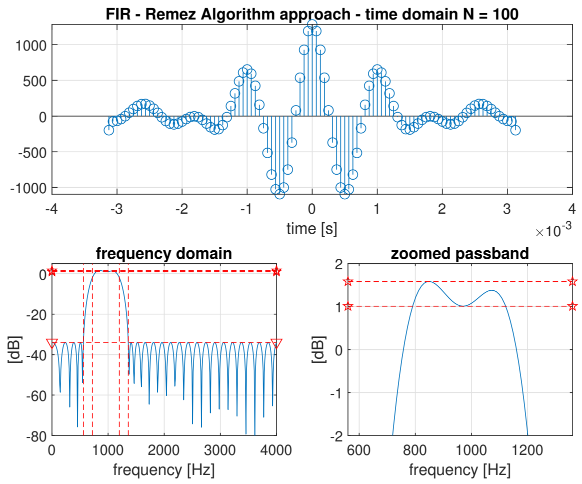

Figure 2: Filter designed with Remez Algorithm finds the optimal value for $N = 256$ but we use $N = 100$ (both even because type I), $\boldsymbol{w} = [10^{-3}, 10^{-2}, 10^{-3}]$.

1.3 Linear Program - Type II

Finally we apply again linear programming but designing a **type II band-pass filter** (We change N in order to be an odd number i.e. $N = 101$). We need to change matrix V and vector $\boldsymbol{x}$ as:

$$V = \underbrace{\begin{bmatrix} 2T\cos(2\pi f(\frac{N}{2})T) \\ 2T\cos(2\pi f(\frac{N}{2}-1)T) \\ 2T\cos(2\pi f(\frac{N}{2}-2)T) \\ \vdots \\ 2T\cos(\pi fT) \end{bmatrix}^T}_{k\times\frac{N+1}{2}} \qquad \boldsymbol{x} = \begin{bmatrix} h_0(0) \\ h_0(T) \\ h_0(2T) \\ \vdots \\ h_0((\frac{N-1}{2})T) \end{bmatrix}$$

The result is very similar to the other ones, we can see a little (due to the fact that N has increased a little) in the values of *stop-band attenuation* $A_s = 34.7584dB$ and *pass-band ripple* $R_p = 0.95529dB$.

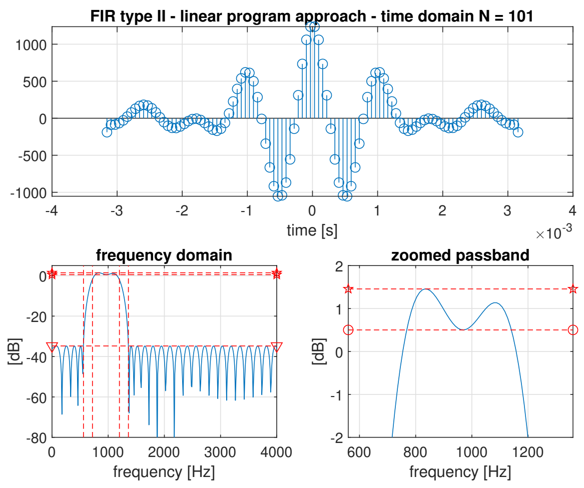

Figure 3: Type II filter designed with linear programming. $N = 101$, $\boldsymbol{w} = [10^{-3}, 10^{-2}, 10^{-3}]$.

However, these three solutions are not able to completely remove the forest sound keeping only the cuckoo's in filtering the audio source.

1.4 Better solution using Remez Algorithm

Then we design with Remez a most suitable solution, setting $N = N_{Remez} + 8 = 322$ and the weighting vector to $\boldsymbol{w} = [10^{-4}, 10^{-2}, 10^{-4}]$. The resulting sound has only the cuckoo sound. We obtain so as *stop-band attenuation* $A_s = 77.1899dB$ and *pass-band ripple* $R_p = 0.23657dB$.
This filter has very little ripple in the pass-band and a considerably low stop-band.

1.5 Conclusions

The final exported audio signal is filtered by means of our *final filter* with $N = 322$ and it is shown in figure 5. The sound is completely different from the original. The noise is attenuated, the other birds' sounds are almost nonexistent and the cuckoo sound is clear (using a filter with ideal behaviour $A = [0, 5, 0]$ the cuckoo sound is also amplified).

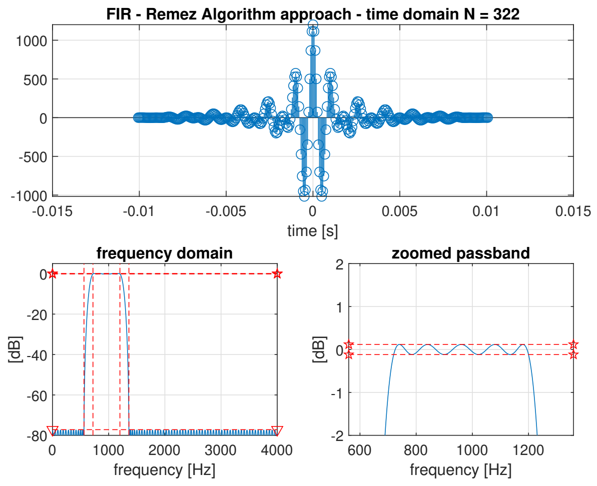

Figure 4: Better solution using Remez Algorithm, $N = 322$, $\boldsymbol{w} = [10^{-4}, 10^{-2}, 10^{-4}]$.

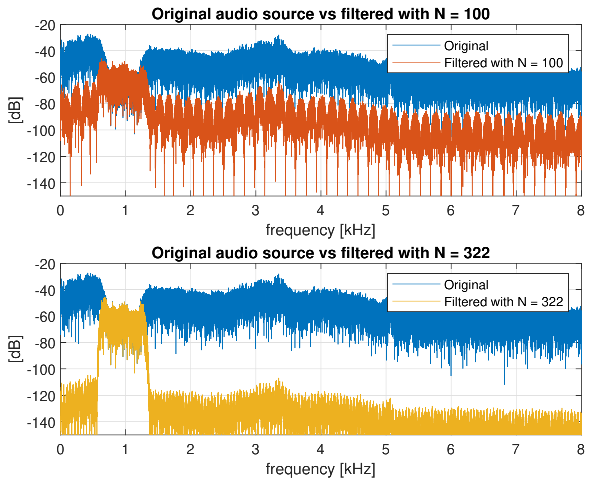

Figure 5: The audio source before and after the filtering

2 Exercise - Derivative Filter

For first we set the constants $B = 3kHz$, $\alpha = 0.05$, and $T = 1/F_p$ with $F_p = 8kHz$, where α is the transition bandwidth in percentage.
The we proceed in designing the reference filter. We define $f_a = (1 - \alpha)B$ and $f_b = (1 + \alpha)B$ the cutoff frequencies.

2.1 Filter design

$H_{Ref}(f)$ is a derivative filter with approximate frequency response $i2\pi f$ over the active signal bandwidth B. So that we decide to build the reference behaviour for a filter design with a **linear program approach**, considering the *positive-valued* samples, as follows:

$$R(f) = \begin{cases} |i2\pi f|, & 0 \leq f \leq f_a \\ 0, & f_b \leq f \leq \frac{F_p}{2} \end{cases}$$

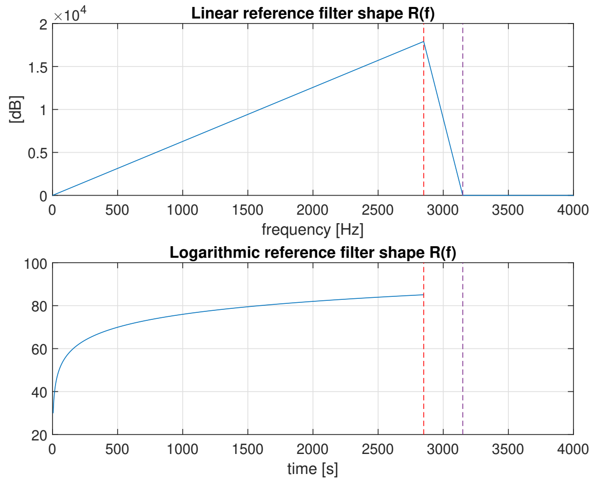

Figure 6: The reference form of the derivative filter, linear and logarithmic.

As the impulse response of a derivative filter $h_0(f)$ *is odd*, we must use a type III or a type IV filter. We choose **type III**. We start with $N = 100$ to ensure a good approximation of the ideal reference form. We thus set the weighting vector $\boldsymbol{w} = \mathbb{1}$.
As the filter we are designing is of type III, $H_0(f)$ has to be build as $H_0(f) = V \cdot \boldsymbol{x}$ where matrix V and vector $\boldsymbol{x}$ are built in the following way:

$$V = \underbrace{\begin{bmatrix} 2T\sin(2\pi f \frac{N}{2} T) \\ 2T\sin(2\pi f(\frac{N}{2}-1)T) \\ 2T\sin(2\pi f(\frac{N}{2}-2)T) \\ \vdots \\ 2T\sin(2\pi f T) \end{bmatrix}^T}_{k\times\frac{N}{2}} \qquad \boldsymbol{x} = \begin{bmatrix} h_0(0) \\ h_0(T) \\ h_0(2T) \\ \vdots \\ h_0((\frac{N}{2}-1)T) \end{bmatrix}$$

The result can be seen in figures 7 and 8. It is possible to notice that the filter obtained by linear programming retraces the reference one overlaying on it as well.

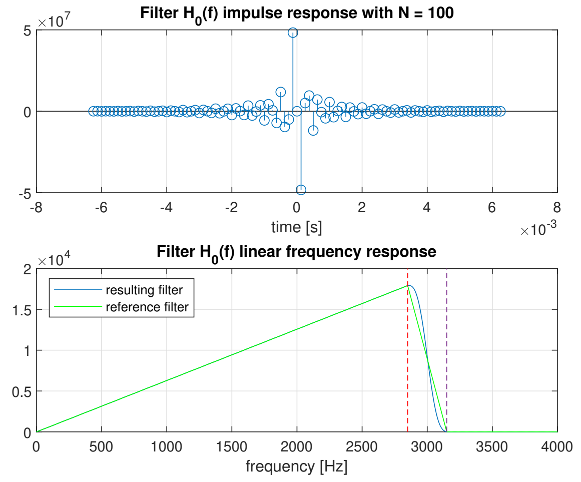

Figure 7: The result of type III linear programming filter design with $N = 100$ in time and frequency (linear).

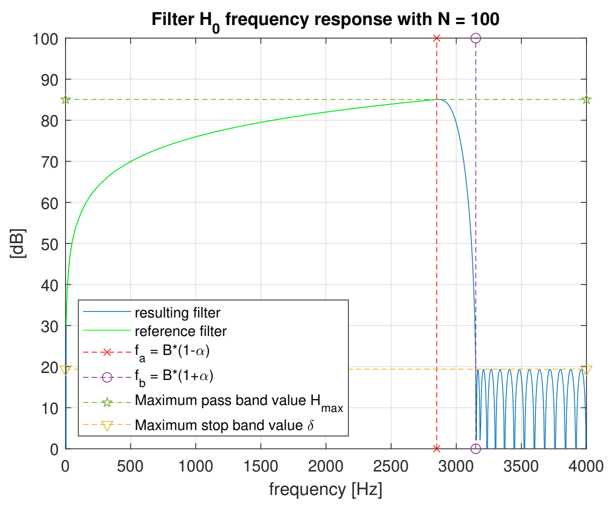

Figure 8: The result of type III linear programming filter design with $N = 100$, $\boldsymbol{w} = \mathbb{1}$.

2.2 Test on a simple function

We can now test our filter on a simple function, like a *well-known waveform*. For this aim we choose the function $f(t) = sen(2\pi f_0 t)$ with $f_0 = 0.2Hz$. Its derivative is known in the closed-form as $\frac{d}{dt}f(t) = 2\pi f_0 cos(2\pi f_0 t)$.
In figure 9 we can compare the original waveform, its well-known derivative and the filtered waveform. The filtered signal differs from the ideal one only a little in amplitude because of the oscillatory behaviour of the designed frequency response.

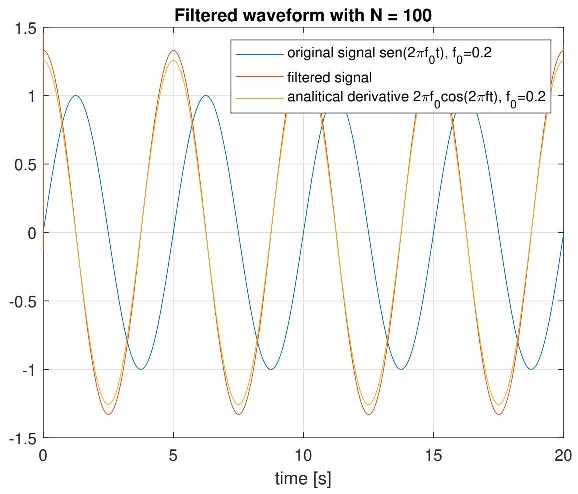

Figure 9: The original waveform, its well-known derivative and the filtered waveform with $N = 100$.

2.3 Smallest value for N

In order to find the smallest value of N which guarantees a *stop-band attenuation* $As \geq 40dB$ we perform the linear programming for **progressively increasing** values of N (starting from $N = 2$ and considering only *even* values) until A_s reaches the value of $40dB$. Here we define $A_s = 20log_{10}(\frac{H_{max}}{\delta_s})$ with:

$$\delta_s = max\{|H_0(f)|\}, \quad f \in [f_b, F_p/2],$$

$$H_{max} = max\{|H_0(f)|\}, \quad f \in [0, f_a]$$

The value we find is $N = 54$. The filter can be seen in figure 10 and its test on the waveform $f(t)$ appears in figure 11.

2.4 Additional considerations

The obtained value for N can change according to the **weighting vector** chosen for the procedure. We decided to set it to the simplest one (i.e. $\boldsymbol{w} = \mathbb{1}$),

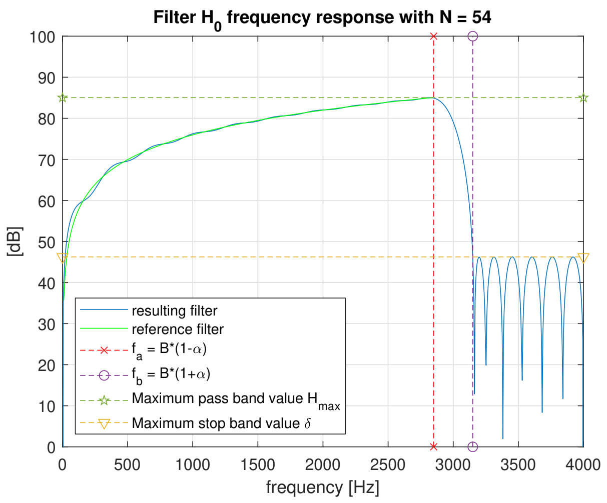

Figure 10: The result of type III linear programming filter design with $N = 54$.

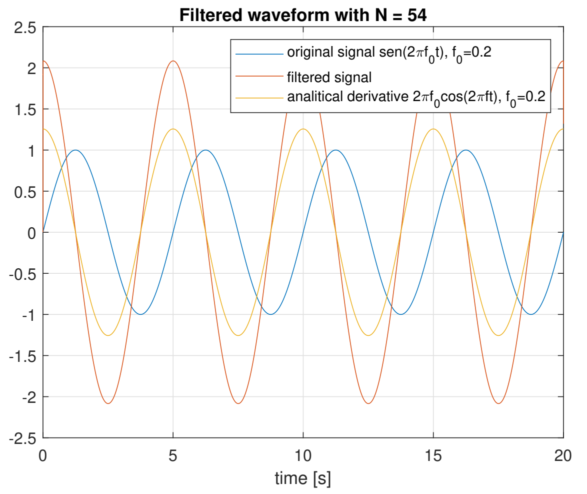

Figure 11: The original waveform, its well-known derivative and the filtered waveform with $N = 54$.

however we could have chosen it in many different ways.
For example with $W(f) \propto |f|$ the filter fits very well *low frequencies* also for small values of N.

3 Exercise - Peter and the wolf

Firstly we load the audio signal, which results very flat and rumbling, and set the constants as: $T_1 = 75\mu s$, $T_2 = 318\mu s$, $T_3 = 3180\mu s$, $A = \frac{(T_2-T_1)}{(T_3-T_1)}$. We decide to set $N = 200$ for the windowing part.

3.1 Windowing Technique

For first we proceed in filtering the audio signal by means of a **windowing technique**. The chosen window is a simple *rectangular window*, because even though we have tried also different window shapes (*Hann, Hamming and Blackman*) we have noticed that the results are very similar one another.
Thus we build our filter only sampling the signal $h(t) = \frac{A}{T_1}e^{-t/T_1} \cdot \mathbb{1}(t) + \frac{1-A}{T_3}e^{-t/T_3} \cdot \mathbb{1}(t)$ over the interval $[0, N]$ with step $T = 1/F_p$ where F_p is the sampling frequency extracted from the audio source.

We also derive the analytical shape of the filter basing on the expression of

$$H(f) = \frac{1}{P(f)} = Re[H(f)] + i \cdot Im[(H(f)]$$

where

$$P(f) = \frac{(1+2i\pi fT_1)(1+2i\pi fT_3)}{(1+2i\pi fT_2)}$$

So that we compare the frequency response of the ideal filter with the one obtained from windowing. The result is shown in figure 12. It is possible to notice that our filter overlays the ideal filter, therefore its coefficients guarantee an appropriate approximation to the desired response. We can notice that it fits better for low frequencies.
It can be shown very clearly in the plot of **real and imaginary parts**. The real component of the windowed filter assumes higher amplitudes w.r.t. the ideal one for high frequencies, but it is sufficiently compensated by the imaginary component which assumes amplitudes lower than the ideal shape for high frequencies (fig. 13).
Finally we filter the audio source with the obtained filter. The results seems very natural, polished by artifacts of the original source.

3.2 Minmax procedure

We design now the filter extracting the real and the imaginary parts of $H(f)$ and using a *minmax procedure* which consists in computing them separately with a **linear program approach**. This procedure is repeated for three different values of samples: with $N = 50$, $N = 100$ and $N = 200$.

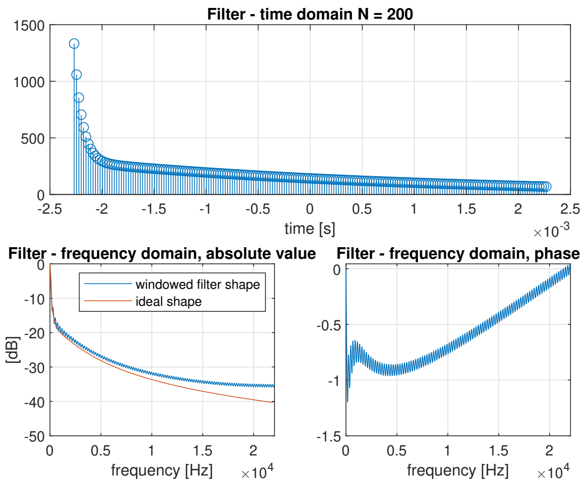

Figure 12: The resulting filter obtained with a rectangular window of $N = 200$ (201 samples) in time and frequency, compared to the ideal one.

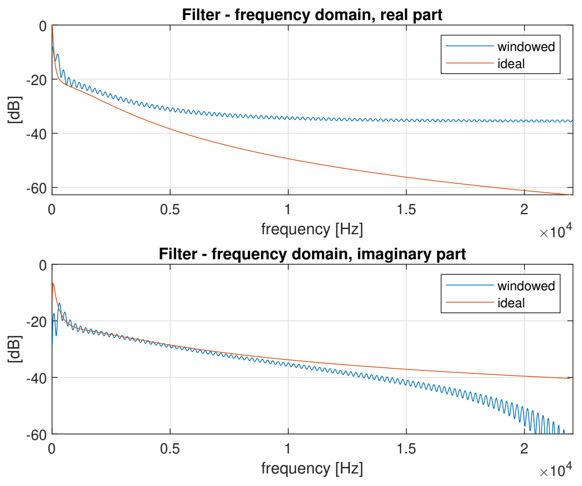

Figure 13: Real and imaginary parts of Windowed and ideal filters.

3.2.1 Real Part

For first we compute the real part which corresponds to a low-pass filter, taking as a reference shape $R_R(f) = Re[H(f)]$ (i.e. the real part of the ideal filter). Then build matrix V according to the requirements of **Type I** filters as in Exercise 1.

3.2.2 Imaginary Part

Then we compute the imaginary part which corresponds to an high-pass filter, taking as a reference shape $R_I(f) = Im[H(f)]$ (i.e. the imaginary part of the ideal filter). Then build matrix V according to the specifies of **Type III** filters as in Exercise 2.

The weighting factor at frequency f is set to $W(f) = |H(f)|$, for both real and imaginary parts, in such a way that a smaller error is enforced at higher frequencies. In addition is important to consider the frequency range for the evaluation of $H(f)$ as $[0, F_p - \epsilon]$ where $\epsilon > 0$ (Set here $\epsilon = 100 \ll F_p$) is necessary to obtain the desired shape for the filter and have good fitting, especially in the imaginary part.

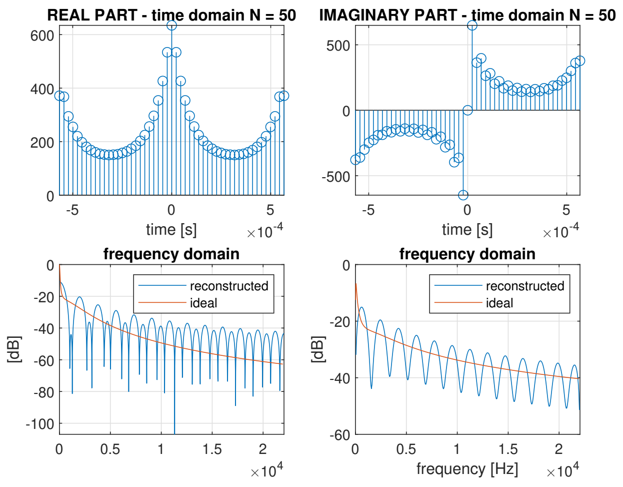

Figure 14: Real and imaginary parts of the filter obtained with minmax procedure with $N = 50$ and the ideal filter.

Finally we rebuild the vector merging the real and imaginary parts as:

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

The final result can be shown in figures 17, 18 and 19. The better one is clearly the one with $N = 200$, because it holds a behaviour less oscillating and retraces better the ideal shape. We can also notice that the *phase* of the filter is continuous with little bounces in correspondence with the local minima of the absolute value of the filter.
The filtered audio source results improving the audio quality, but the output is not as clear as the one obtained with windowing filter design.

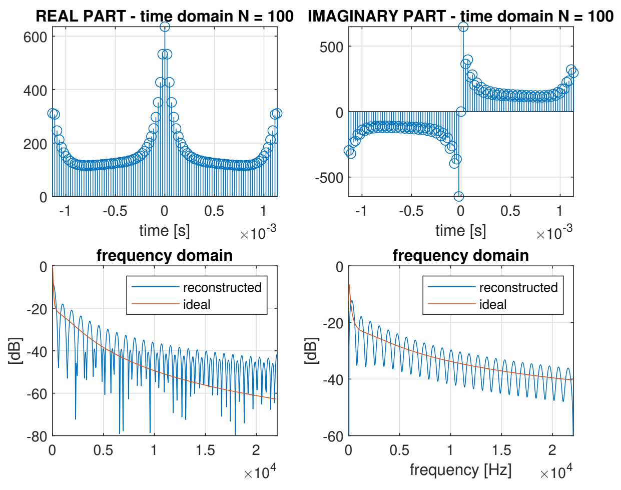

Figure 15: Real and imaginary parts of the filter obtained with minmax procedure with $N = 100$ and the ideal filter.

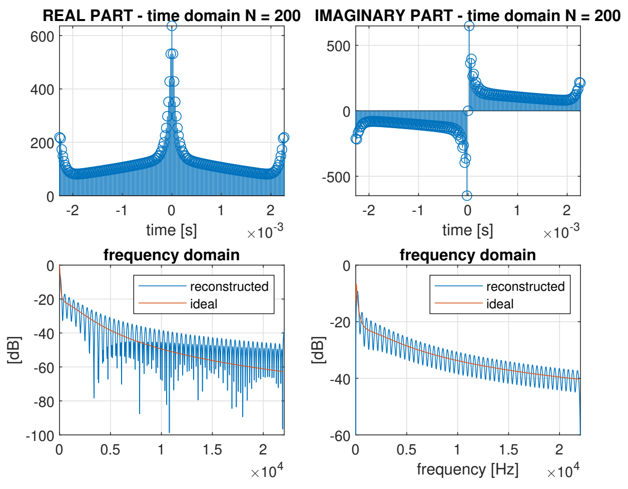

Figure 16: Real and imaginary parts of the filter obtained with minmax procedure with $N = 200$ and the ideal filter.

3.3 Final comparison and Conclusions

Figures 20, 21 and 22 compare the ideal filter, the windowed filter and the one obtained through minmax procedure for the three values of N (Rebuilding the windowed filter according to the value of N to guarantee a comparison between filters holding the same number of samples).

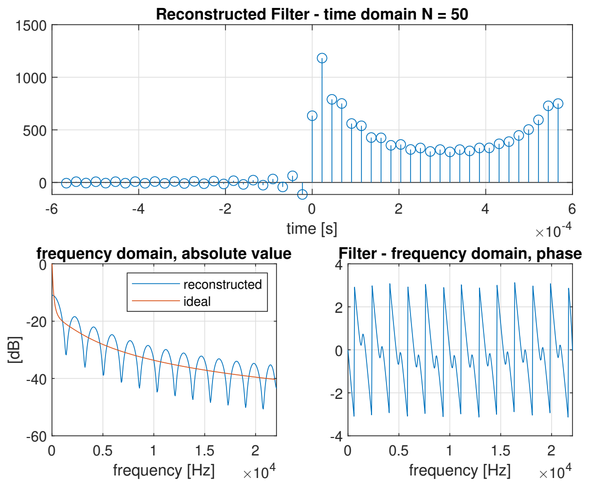

Figure 17: Filter obtained from minmax procedure with $N = 50$

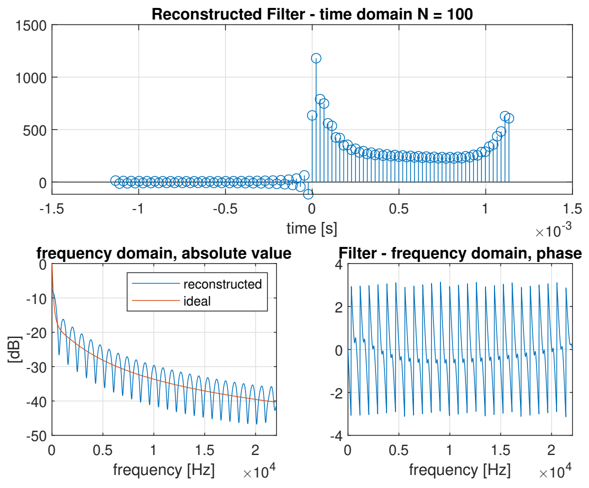

Figure 18: Filter obtained from minmax procedure with $N = 100$

We can observe that a *large number of samples*, despite it increases the computational time, assures a better fitting.
Moreover, the windowed filter is more efficient than the other ones because it holds a *less oscillating behaviour* and we can hear a cleaner resulting audio, even though, as remarked before, the windowed filter frequency response assumes increasing bigger amplitudes for high frequencies w.r.t. the ideal filter, while the other one does not.

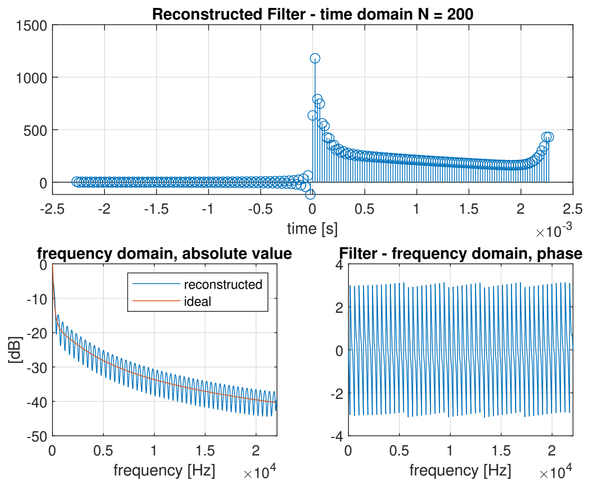

Figure 19: Filter obtained from minmax procedure with $N = 200$

For these reasons the audio file exported is the one with $N = 200$ obtained through the *filter designed with windowing technique.*

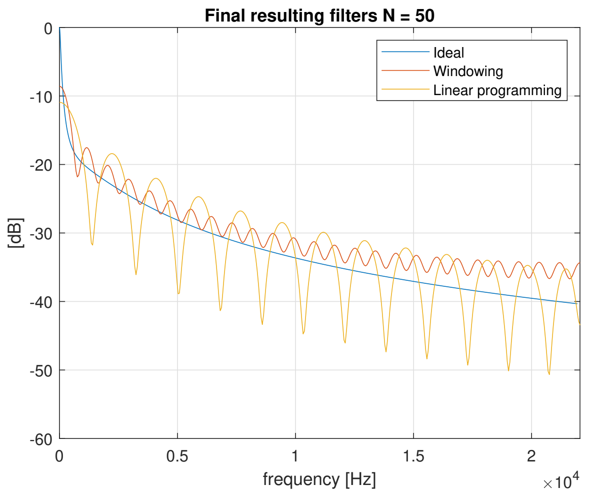

Figure 20: Filters comparison with $N = 50$.

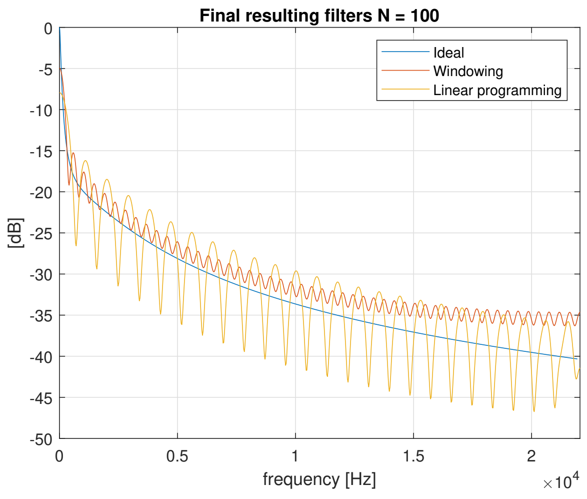

Figure 21: Filters comparison with $N = 100$.

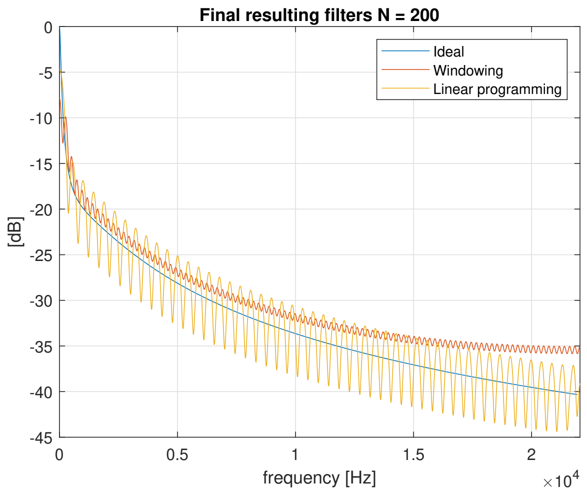

Figure 22: Filters comparison with $N = 200$.