Homework 2 DSP

Elena Camuffo 1234370

10 December 2019 - 10 January 2020

1 Exercise - Vuvuzela

Listening to the input signal, we can notice that the audio is very noisy. The voice is overhung by the vuvuzela sound, which is very disturbing. The objective is to reduce as much as possible this annoying sound, which has its first harmonic at about $f_0 = 235Hz$.

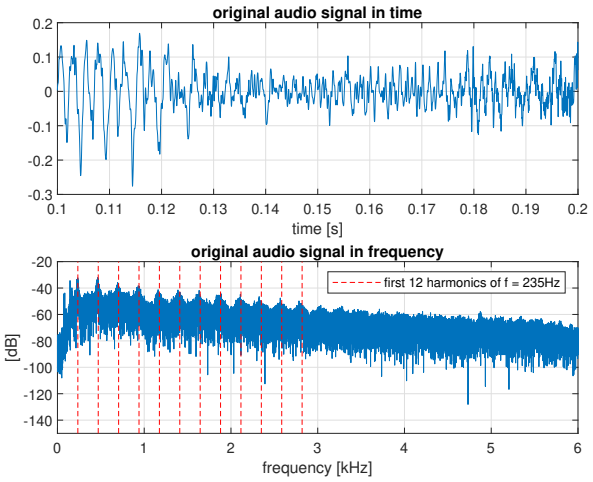

Figure 1: Original audio source in time and frequency. We can see the peaks in correspondence with the harmonics.

1.1 Filter design

As we have to kill one single frequency (and eventually its *harmonics*), we design a **Notch filter** (fig. 2), a second order IIR filter (with $M = N = 2$).

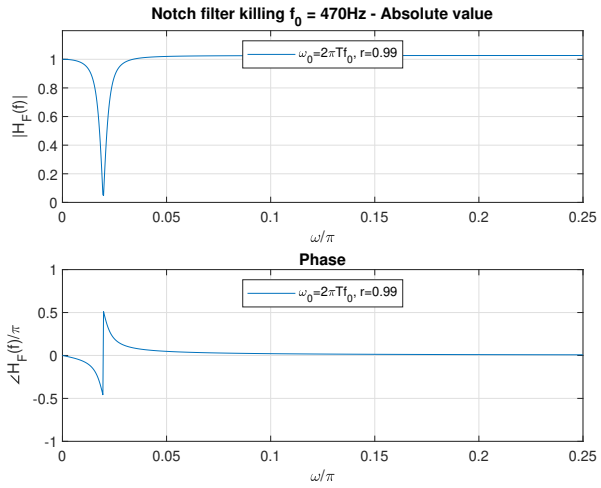

Figure 2: Notch filter which kills the first harmonic of the vuvuzela sound $f_0 = 235Hz$.

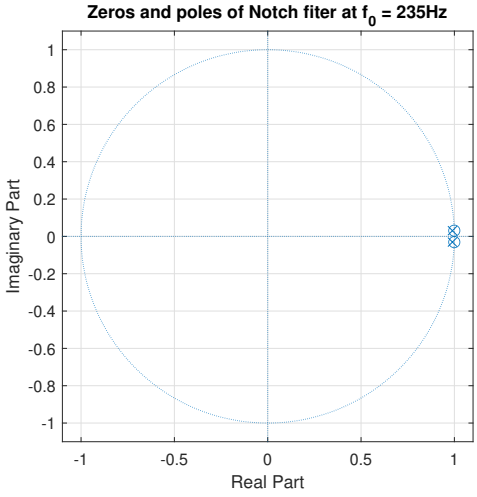

Figure 3: Zeros and poles of Notch filter which kills the first harmonic of the vuvuzela sound $f_0 = 235Hz$.

The filter is built in the following way:

$$H_{Notch}(z) = \frac{1 - \cos(\theta_0)z^{-1} + z^{-2}}{1 - 2r \cdot \cos(\theta_0)z^{-1} + z^{-2}} \cdot \underbrace{\frac{1 - 2r \cdot \cos(\theta_0) + r^2}{2 - 2\cos(\theta_0)}}_{Normalization factor}$$

Where $\theta_0 = 2\pi f_0 T$ and $r \approx 1$ ($r = 0.99$).

In figure 3 we can see that the locations of the poles $p_k = re^{\pm i\theta_0}$ and zeros $z_k = e^{\pm i\theta_0}$ of the filter are *symmetric* w.r.t. the real axis.

Then we delete $m = 12$ harmonics, set at about integer multiples of f_0, i.e. $f_k = (k+1)f_0$, with $k = 1, ..., m$, in correspondence with the principal peaks of the input sound of figure 1.

We simply **repeat the procedure above m times**, taking at iteration k as input signal the output of iteration $k-1$ as represented in figure 4.
Figure 5 shows the bunch of m different filters obtained, $h_1(\cdot), ..., h_m(\cdot)$.

Figure 4: Scheme of Filtering.

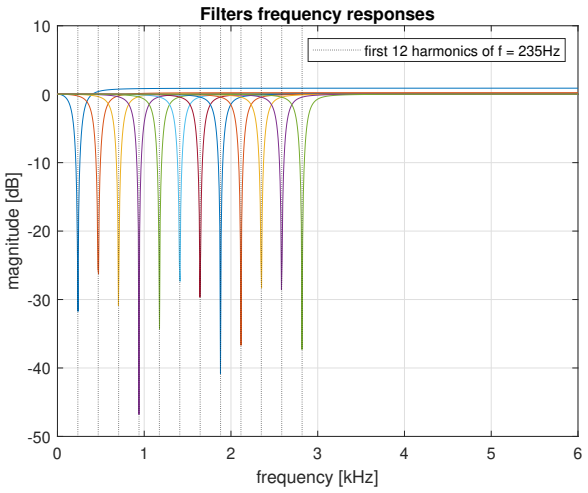

Figure 5: All the filters used.

The final audio $y_m(\cdot)$ shows a consistent improvement w.r.t. the original source (fig. 6). The vuvuzela sound is not completely suppressed, but perceptively reduced.

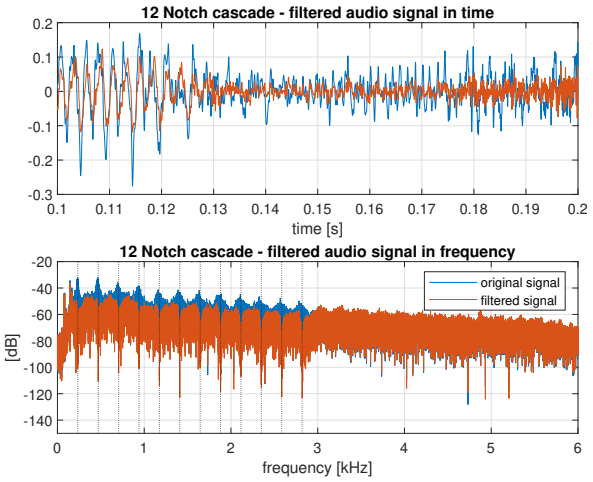

Figure 6: The resulting audio filtered as in scheme of figure 4.

1.2 Additional Tentatives

In order to reach some improvements we try to kill **all the harmonics**. Firstly following the iterative method used before (with Matlab's Notch filter, keeping the same value for the parameters) and then with a *Notch comb*.
The resulting audios from these methods (fig. 7) are much more polished and the vuvuzelas are more attenuated.
The first seems to bring a real improvement, while in the second the voice is *a little distorted*. Anyway we exported the audio obtained from the first procedure.

1.3 Conclusions

The deletion of a single frequency is *not sufficient* to clean the audio source from the vuvuzela sounds. It is necessary to delete also a couple of **other harmonics** to notice a significant improvement in the output.
In the filter design is important to select a value of r *close enough to one* to avoid information loss.

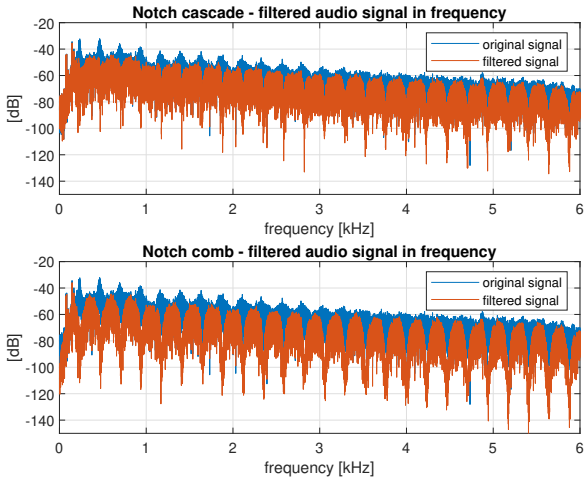

Figure 7: The resulting audio filtered with the notch cascade and the comb notch.

2 Exercise - Peter and the wolf

The input audio signal results very flat and rumbling, like in *exercise 3 of homework 1*. The considerations are the same, and the same is also the reference behaviour of the equalizer, which corresponds to the analog filter with **Laplace transform**:

$$H_a(s) = \frac{(1 + sT_2)}{(1 + sT_1)(1 + sT_3)}$$

Where $T_1 = 75\mu s$, $T_2 = 318\mu s$, $T_3 = 3180\mu s$.

2.1 Transform method

To filter the signal we use the *Transform Method* to design an IIR filter ($M = N = 2$) starting from $H_a(s)$.
The purpose is to map the continuous-time analog filter to the discrete-time domain by means of a **bilinear transform** i.e. operating substitution $s = k \cdot \frac{z-1}{z+1}$. Therefore the expression obtained is the following:

$$H_d(z) = H_a(s)|_{s=k\cdot\frac{z-1}{z+1}} =$$

$$= \frac{(1 + kT_2)z^2 + 2z + (1 - kT_2)}{(1 + kT_3 + kT_1 + T_1T_3k^2)z^2 + (2 - 2k^2T_1T_3)z + (1 - kT_3 - kT_1 + k^2T_1T_3)}$$

As the axis in the s-plane are mapped to the unit circle in the z-plane, the following relation holds:

$$z = e^{i2\pi fT} \longrightarrow s = k \cdot \frac{e^{i2\pi fT} - 1}{e^{i2\pi fT} + 1} = k \cdot i \tan(\pi fT)$$

Since for small values of f, $\tan(\pi fT) \approx \pi fT$ and in the analog domain $s = i2\pi f$, it follows that:

$$i2\pi f \approx k \cdot i\pi fT$$

Consequently the scaling factor results $k = \frac{2}{T}$ and we get:

$$H_d(z) = H_a(s)|_{s=k \cdot \frac{z-1}{z+1}} = H_a\big(\frac{2}{T} \cdot \frac{z-1}{z+1}\big)$$

This bilinear transformation is also known as *Tustin's method*.

As shown in figure 8, the obtained filter matches the reference behaviour for *low frequencies*, while for high frequencies it decreases more rapidly.
The BIBO stability of $H_d(z)$ is confirmed by zeros and poles inside the unit circle (fig. 9). In the filtered audio we can notice an improvement w.r.t the original.

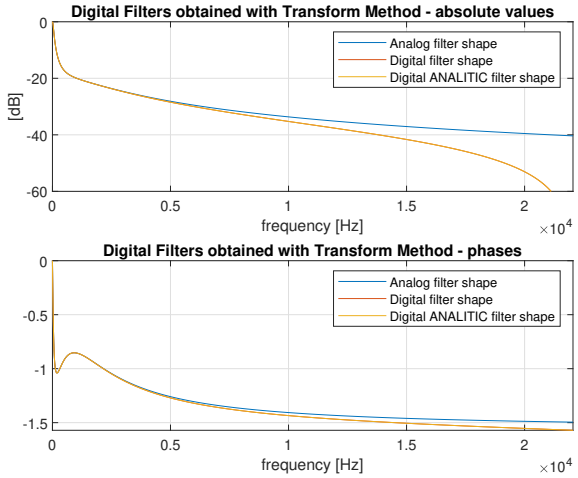

Figure 8: Filter obtained with the Transform method. The red one via the *Tustin's trasformation* provided by Matlab, the yellow one manually operating the substitution.

2.2 Optimization method

Then we design the filter with a direct optimization method based on a **linear programming approach** trying to obtain a match with the absolute value of the analog filter, i.e. $D(f) = |H_a(i2\pi f)|$.

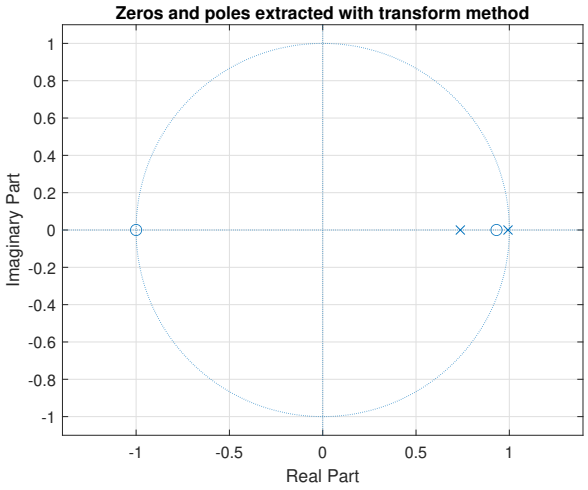

Figure 9: Zeros and poles of the filter obtained with the Transform method.

As a *weighting vector* we set $W(f) = \frac{1}{D(f)}$ in order to enforce a smaller error at low frequencies.
We consider the frequency range $[0, F_P - \epsilon]$ for the evaluation, where $\epsilon > 0$ (Set here $\epsilon = 100 \ll F_P$), with sampling step $F = \frac{F_P}{10^3} = 44.1Hz$.

The obtained zeros and poles from the procedure (fig. 10) are the ones of the *self reciprocal polynomial* $D(f)^2 = \left|\frac{P(z)^2}{Q(z)^2}\right| = \left|\frac{P(z)\hat{P}(z)}{Q(z)\hat{Q}(z)}\right|$ where $\hat{P}(z)$ and $\hat{Q}(z)$ are the *mirror image polynomials* of $P(z)$ and $Q(z)$.

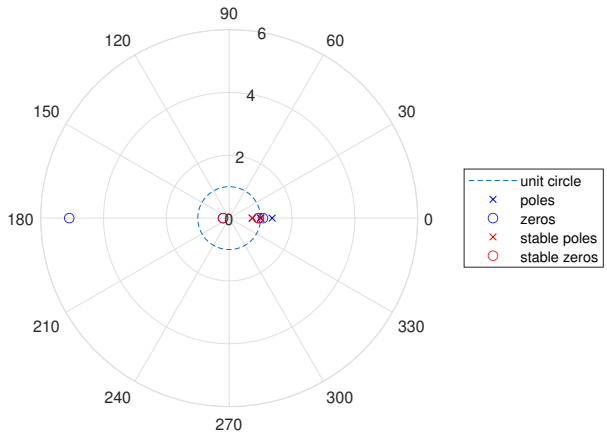

Figure 10: All zeros and poles of the self reciprocal polynomial.

Then reducing *zeros and poles* to the *stable* ones (fig. 11) we get the minimum phase filter of figure 12.

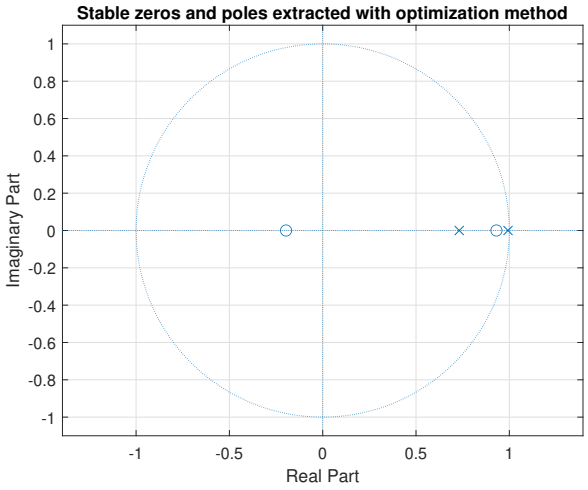

Figure 11: Stable zeros and poles of the filter obtained with optimization method.

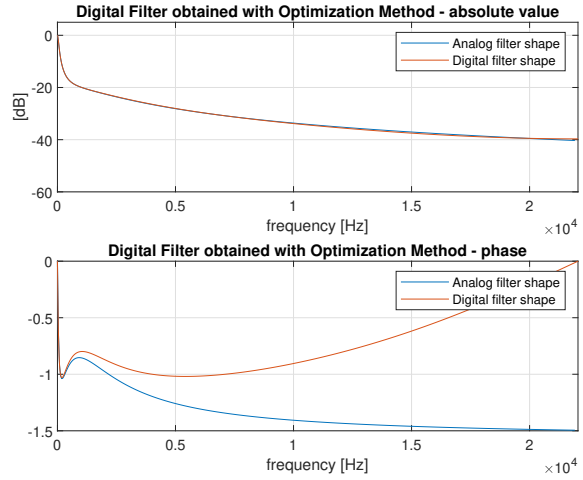

Figure 12: Filter obtained with optimization method.

We can notice that this filter shape matches even better the original shape than the previous, in terms of *absolute value*, while the match is not so good as regards the phase.

2.3 Conclusions

The filter designed with the **transform method** reshapes well the analog filter and it requires only 4 coefficients w.r.t. the 200 of the filter designed in *exercise 3 of homework 1*.
However, for high frequencies it decreases more rapidly than the analog. This is because the approximation $\tan(\pi f T) \approx \pi f T$ holds only for low frequencies.

As regards the direct **optimization method**, it also approximates well the analog filter shape. Compared to the one obtained with the transform method, it gives a better approximation of the absolute value and a worse one of the phase, as its zeros locations are different.
This method requires a *good choice of the initial conditions* to have a good approximation of the desired filter shape, but achieves better results for high frequencies.

The *output audios* are very similar each other, and we can't hear any improvement brought by the second one. For this reason we exported the one obtained with the first procedure.

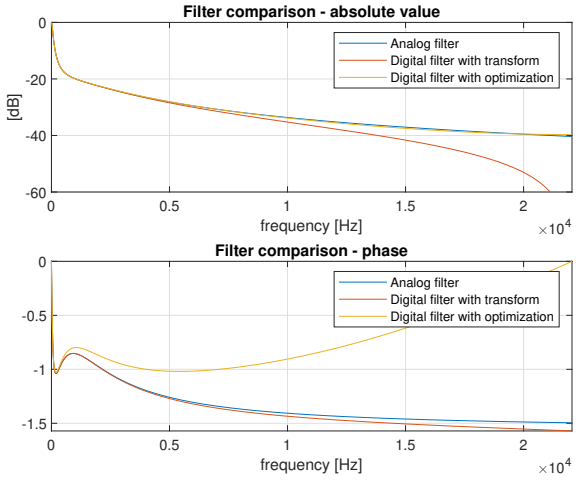

Figure 13: Comparison between the two filters.

3 Exercise - Buongiorno

The loaded audio source is a *stereo* sound (it is subdivided into 2 channels). We need to convert it from rate $F_P = 44.1kHz$ to $F'_P = 48kHz$. To reduce the computational time required, we take for our analysis only a portion (1/4) of the entire soundtrack.

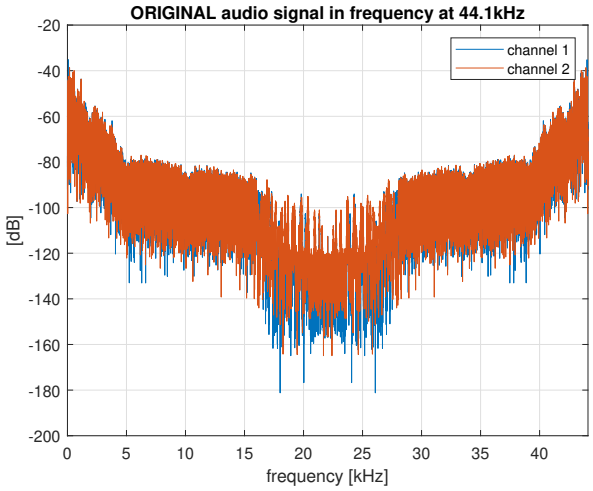

Figure 14: Original audio signal.

3.1 Rate Conversion Algorithm

To operate the conversion we relied on the *rate conversion algorithm*, which is represented in figure 15 and is composed of the following steps:

- For first **up-sample** the input signal $x(nT)$, with sampling frequency F_P and sampling period $T = 1/F_P$, adding *L-1 zeros* between original samples and multiplying by L. The output signal has sampling period $F''_P = L \cdot F_P$.
- Then **filter** it with a *low pass* filter with cutoff frequency $f_0 = min\{\frac{F_P}{2}, \frac{F'_P}{2}\}$. This filter is a composition of two filters: a filter to preserve only the information of the input signal and a filter to avoid aliasing.
- Finally **down-sample** the result, taking one over M samples and obtaining this way a final output signal of sampling period $F'_P = \frac{F''_P}{M} = F_P \cdot \frac{L}{M}$.

3.1.1 Interpolation

The interpolation step is here simply realized by inserting $L - 1 = 159$ zeros between samples of the original signal and multiplying it by L. The result is

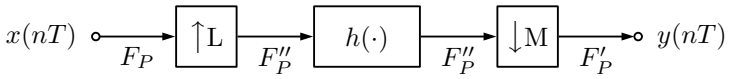

Figure 15: Scheme of multirate system used. $L = 2^5 \cdot 5 = 160$, $M = 7^2 \cdot 3 = 147$. $F_P = 44.1kHz$, $F_P'' = \boldsymbol{L} \cdot F_P$, $F_P' = \frac{\boldsymbol{L}}{\boldsymbol{M}} \cdot F_P = 48kHz$.

the signal in figure 16 with sampling frequency $F_P'' = 160 \cdot F_P = 7056kHz$.

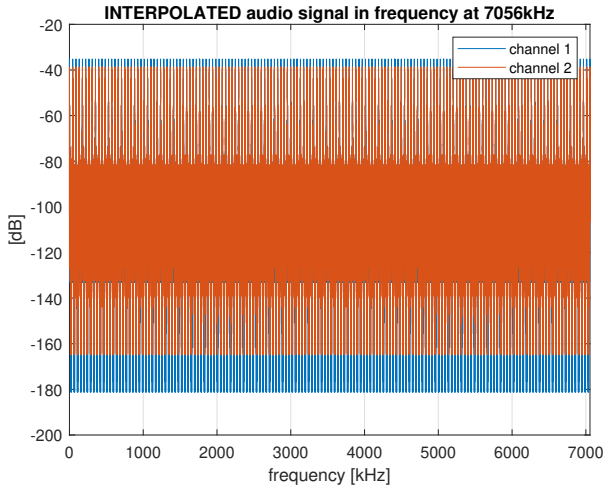

Figure 16: Interpolated audio signal by a factor $L = 2^5 \cdot 5 = 160$.

3.1.2 Filtering

The filtering operation was made designing a suitable filter with cutoff frequency $f_0 = F_P/2$ and order $N = 11954$, estimated relying on *firpmord function*.

The filter is designed via a **windowing technique** (fig. 17) technique because, even if a filter designed with Remez would provide better performances in terms of pass-band ripple and stop-band attenuation, it cannot guarantee to satisfy the *correct interpolation property* and requires more computational time.

The result of the filtering step is shown in figure 18.

3.1.3 Decimation

The final step is decimation, simply realized taking one sample over $M = 147$. The final result at frequency $F_P' = 48kHz$ is really similar to the original audio,

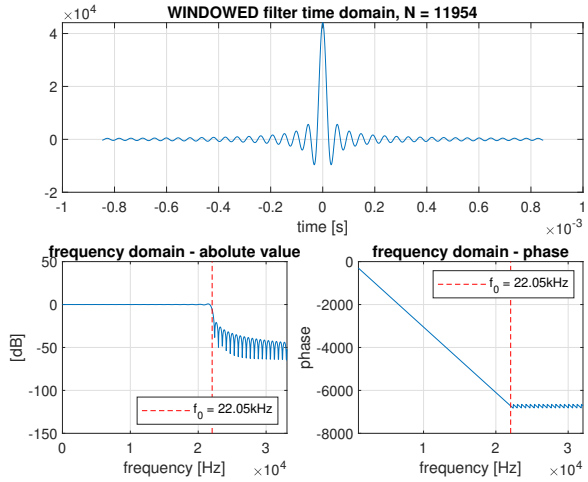

Figure 17: Filter designed with a windowing technique. Cutoff frequency is $f_0 = F_P/2 = 22.05kHz$.

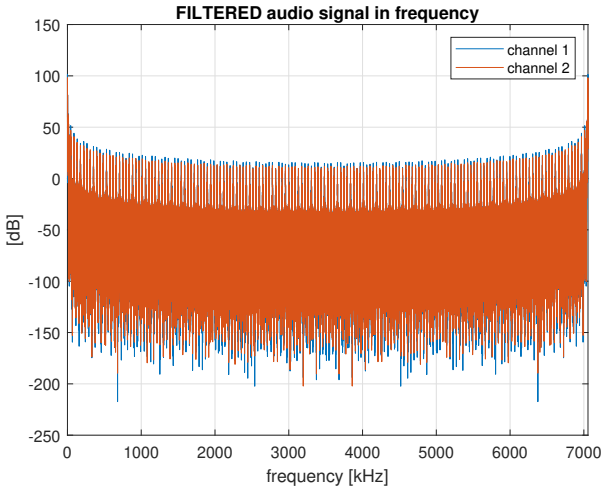

Figure 18: Filtered audio signal with a windowing technique.

and it is shown in figure 19.

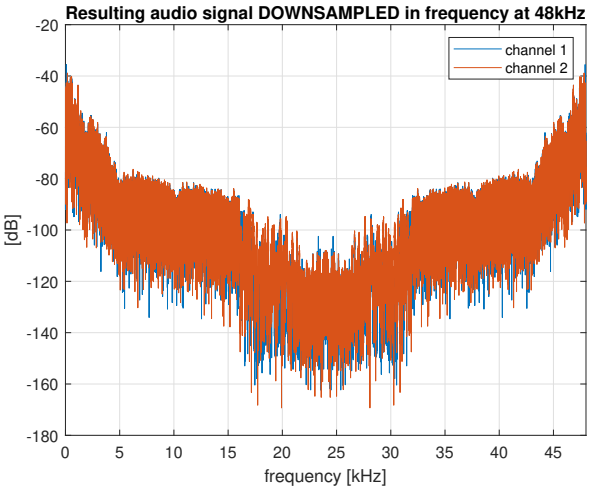

Figure 19: Final audio, decimated by a factor $M = 7^2 \cdot 3 = 147$.

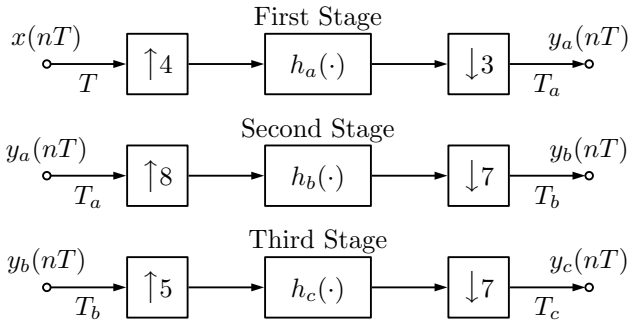

Figure 20: Scheme of multistage rate conversion. $T = 44.1kHz = 1/F_P$, $T_a = \frac{3}{4}T$, $T_b = \frac{7}{8}T_a$, $T_c = \frac{7}{5}T_b = T' = 48kHz$. The cutoff frequecies of the three filters are respectively $f_a = F_P/2$, $f_b = F_a/2$ where $F_a = 1/T_a$ and $f_c = F'_P/2$.

3.2 Multistage Approach

Then we realize the same rate conversion, but using a *multistage* approach. Also here we follow the *rate conversion algorithm* explained before and we iteratively apply it to the audio source, in such a way that the output of the first iteration is the input of the second and the output of the second is the input of the third, as shown in scheme of figure 20.

The three filters have respectively orders $N_a = 299$, $N_b = 598$ and $N_c = 523$, estimated relying on *firpmord function*.
The first 2 filters have cutoff frequencies respectively $f_a = F_P/2 = 22.05kHz$,

$f_b = F_a/2 = 29.4kHz$, like in the overall conversion.
On the other hand, the third one has $L < M$. So its cutoff frequency must be $f_c = F'_P/2 = F_c/2 = 24kHz$ and not $F_b/2$, in order to *avoid aliasing* in the decimation stage.

3.3 Conclusions

The multistage approach results the better solution both in terms of filters complexity and computational time required:
The three filters built for the multistage hold a really little number of samples (order of 10^2) compared to the filter of the single stage approach (order of 10^4) to reach more or less the same quality in the output sound.
The single stage approach is also more *computationally demanding*, especially for long input audios.

For these reasons the exported audio file is the one obtained with the multistage approach.

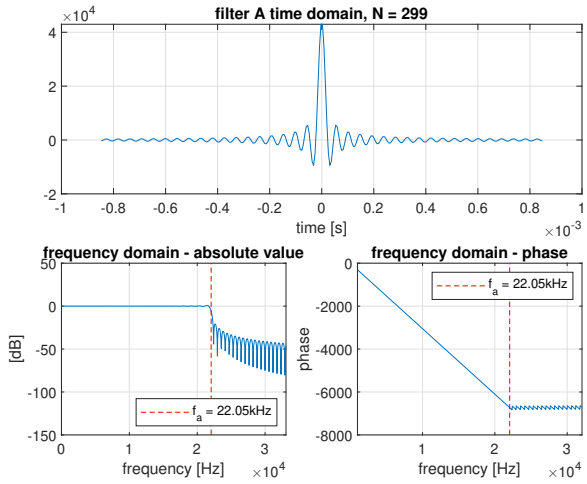

Figure 21: Filter used for rate conversion with $L = 4$, $M = 3$.

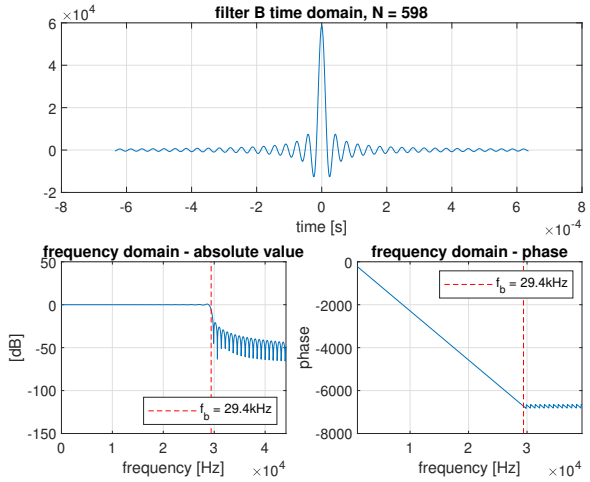

Figure 22: Filter used for rate conversion with $L = 8$, $M = 7$.

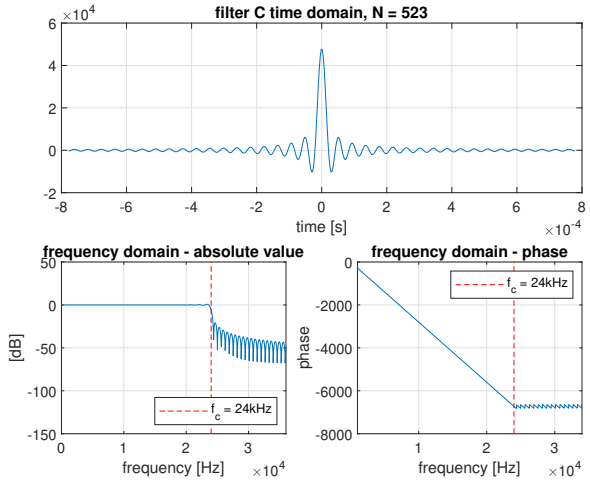

Figure 23: Filter used for rate conversion with $L = 5$, $M = 7$.

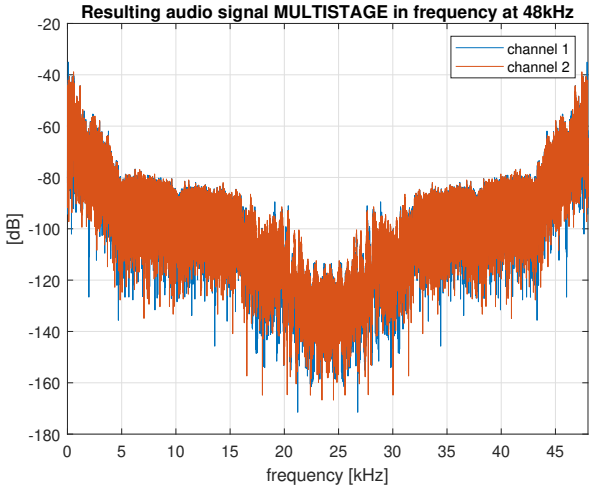

Figure 24: Filtered audio signal with a multistage approach.