TSIA201 TP1
Practical work on the conversion of sampling rate and STFT

IQBI Hamza

30 September 2022

1 Conversion of sampling rate

1.1 Question 1 :

We have that h is such that :

$$(\forall \nu \in [\frac{-1}{2}, \frac{1}{2}]), H(\exp(2i\pi\nu)) = \begin{cases} L & if|\nu| < min(\frac{1}{2M}, \frac{1}{2L}) \\ 0 & otherwise \end{cases}$$

In the case of a conversion of 48kHz to 32kHz, we have that M=3 and L=2.

Using the python function [scipy.signal.remez], we get the plot of the gain in dB of h :

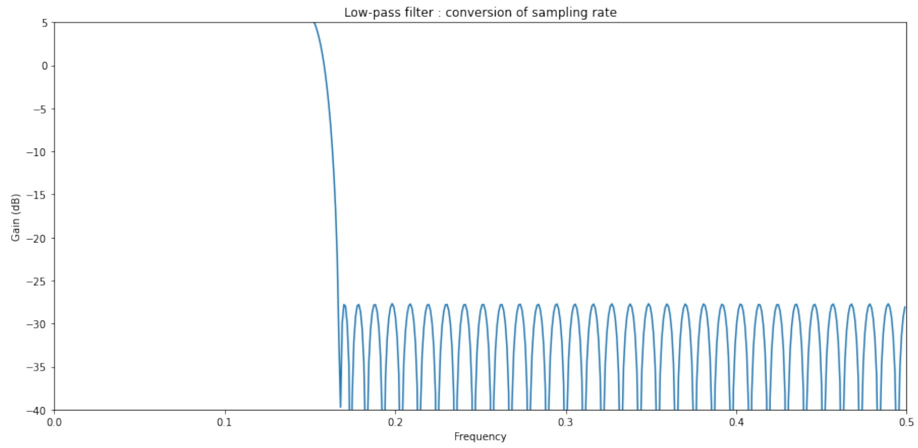

FIGURE 1 – The gain of the filter h in dB

We can see that we get a low-pass filter with a difference of at least 45 dB between the pass-band and the stop-band.

1.2 Question 2 :

The simple implementation of the resampling consists of executing an upsampling of parameter L, then applying the anti aliasing filter h (comptuted in Q1), and after that we proceed to a downsampling of parameter M.

The reconstruction with the simple implementation of the filter took : $0.5671966075897217 seconds$.

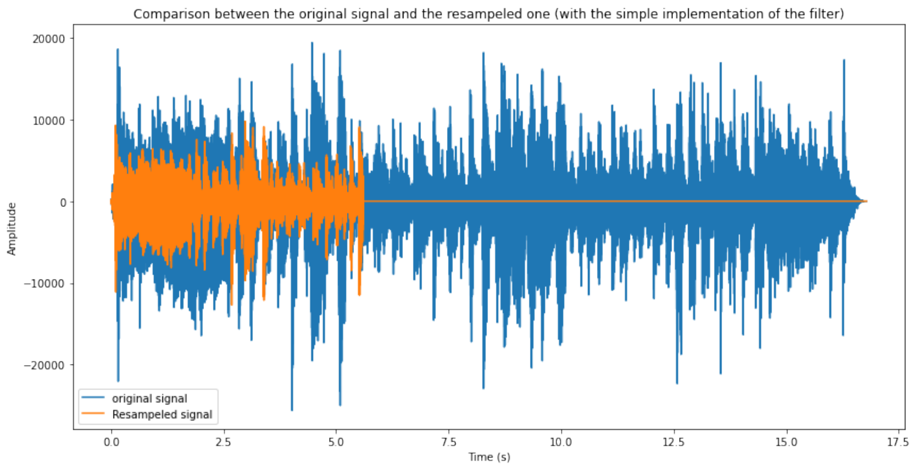

Figure 2 – Comparison between the original signal and the resampeled one

1.3 Questions 3 - 4 :

We choose to implement the filter via the polyphase components of TypeI, because the complexity in this case decreases by M^2, since $M > L$ (as for TypeII, it decreases by L^2)

The resampeled signals obtained with the two implementations "have the same shape", but with a difference in the magnitude. We can say that both implementations give pretty much the same resulting signal.

The construction with the efficient implementation of the filter took $0.18200016021728516 seconds$.

Conclusion : we can say that indeed, thanks to the polyphase components, the time of execution had decreased (approximately by a factor of 3).

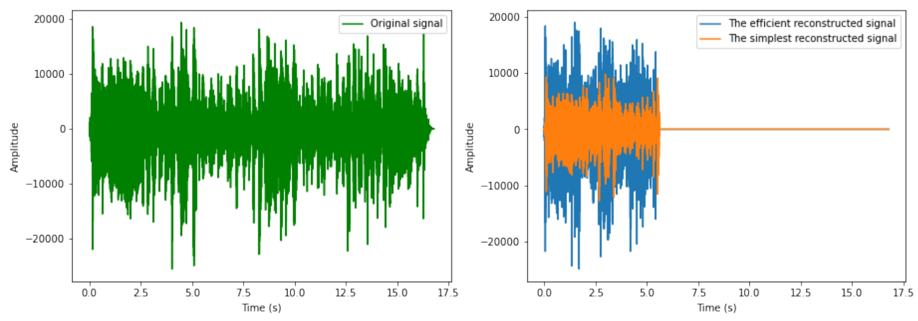

Figure 3 – Comparison between the original signal and the resampeled ones (with both a simple and an efficient implementation)

2 STFT audio equalization :

2.1 STFT analysis :

2.1.1 Question 1 :

Using the function [numpy.hanning(Nw)], we compute a hanning window of size Nw (chosen equal to 512 here).

Using the function [numpy.fft.fft(w)], we compute the Fast Fourier Transform of size M (chosen equal to 2048 here) of the obtained hanning window .

The plot of its DFT gives :

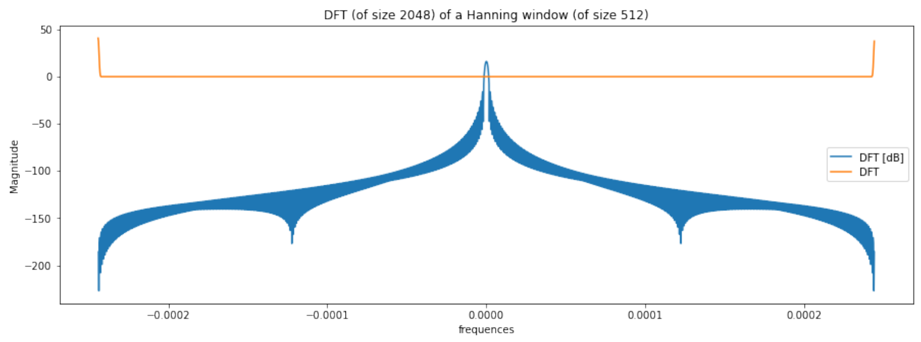

Figure 4 – Discrete Fourier Transform of a Hanning window in magnitude and in dB

We can remark that there is a difference of approximately 150dB between the magnitude of $\hat{w}(0)$ and $\hat{w}(0.0002)$.

2.1.2 Question 2 :

The ”band-pass convention” corresponds to the implementation given in this notebook.

2.1.3 Question 3 :

We get that $x_3 = [7.34399083-0.08287586j, 7.44142878+0.12961596j, 7.68528996+0.39286623j, \ldots, -1.51160285-2.18743411j, -2.06138676-1.94795616j, -2.50967287-1.58929685j]$.

Therefore, x_k is complex.

$\widetilde{X}(k,u)$ performs a band-pass FIR filtering around frequency $\nu_k = \frac{k}{M}F_s$.

In fig.5 we can see the spectrogram of a signal (that we shall call $x_{filtered}$) obtained by applying a band-pass filter (around $\nu_k \approx 4500Hz$) to the original signal x.

In fig.6 we plot the spectrogram of $Re(x_k)$. We can remark that it corresponds to a band-pass filtering, around the frequency $\nu_k \approx 4500Hz$ as expected.

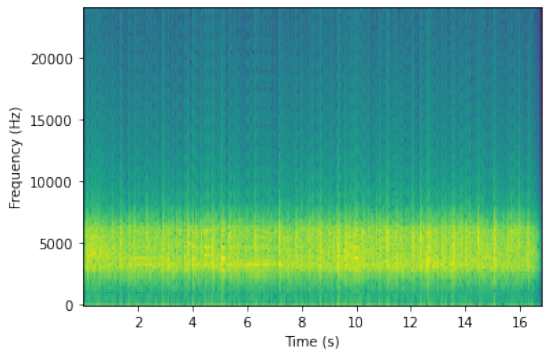

Figure 5 – The spectrogram of $x_{filtered}$

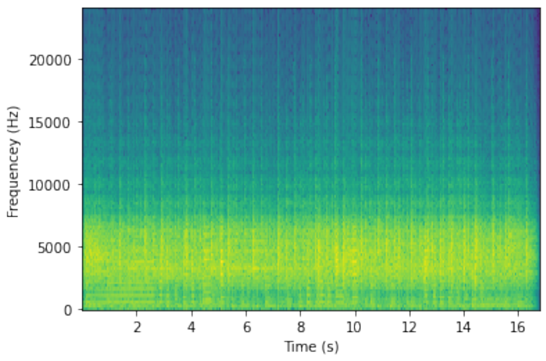

Figure 6 – The spectrogram of x_k

$\therefore$ Thus, we can say that, the STFT is equivalent to a filter bank.

When hearing the sound $Re(x_k)$ (for k = 3), we can still hear and distinguish the original sound, but with many instantaneous noises and with deteriorated quality.

Therefore, we need to reconstruct the signal, via the OverLap-Add method.

2.2 Reconstruction :

2.2.1 Question 4 :

We consider $w = np.hanning(Nw)$ and h the hanning window defined as : $h = \frac{1}{5625}w$.

Then by applying the function *ola* given in the template with the parameters : $\begin{cases} w = np.hanning(N_w) \\ hop = (1 - overlap)N_w \quad withoverlap = 0.75 \\ N_b = 500 \end{cases}$

We get a list f such that its elements sum up to 1.0000000000000369.

Therefore, the condition $[(\forall n \in \mathbb{N}), f(n) = \sum_{u \in \mathbb{Z}} w_a(n-uR)w_s(n-uR) = 1]$ is fulfilled (where $w_a = h$ is the Hanning window used for the analysis and $w_s = h$ is the Hanning window used for the synthesis).

Moreover, and in order to illustrate, we plot the graph of f.

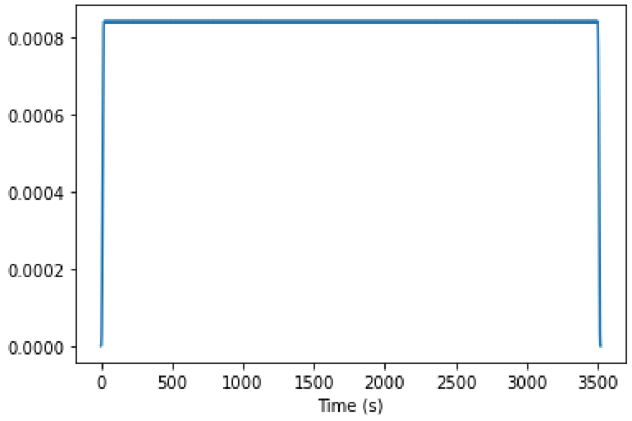

FIGURE 7 – Graph of f

2.2.2 Question 5 :

Since we have $(\forall n \in \mathbb{N}), \sum_{u \in \mathbb{Z}} w_a(n-uR)w_s(n-uR) = 1$, the $OverLap-Add$ method permits an ideal reconstruction of the original signal x as follows :

$$(\forall n \in \mathbb{N}), x(n) = \sum_{u \in \mathbb{Z}} y_s(u, n - uR)$$

where $y_s(u,n) = DFT^{-1}[\widetilde{X}(k,u)](n)h(n)$.

We implement the reconstruction in python in order to plot the graph of the reconstructed signal. We get the comparison as follows :

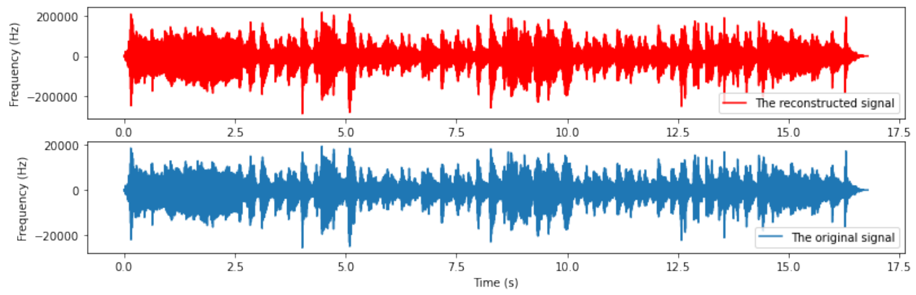

Figure 8 – Comparison between the reconstructed signal (by the OLA method) and the original signal

Conclusion : The OverLap-Add method is efficient and gives an ideal reconstruction of the signal when the condition on the Hanning windows (for analysis and synthesis) is fulfilled.