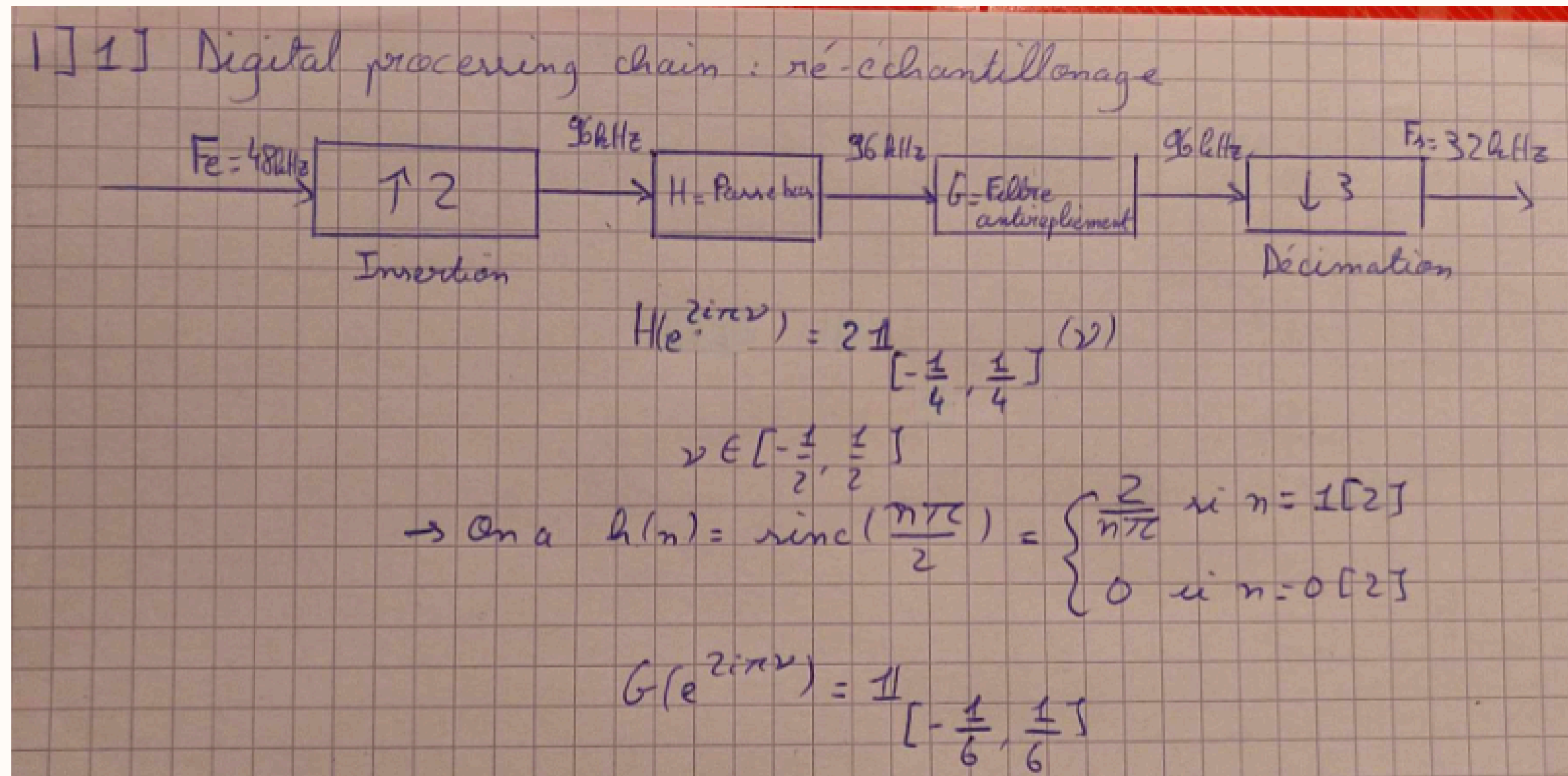


1. Conversion of sampling rate

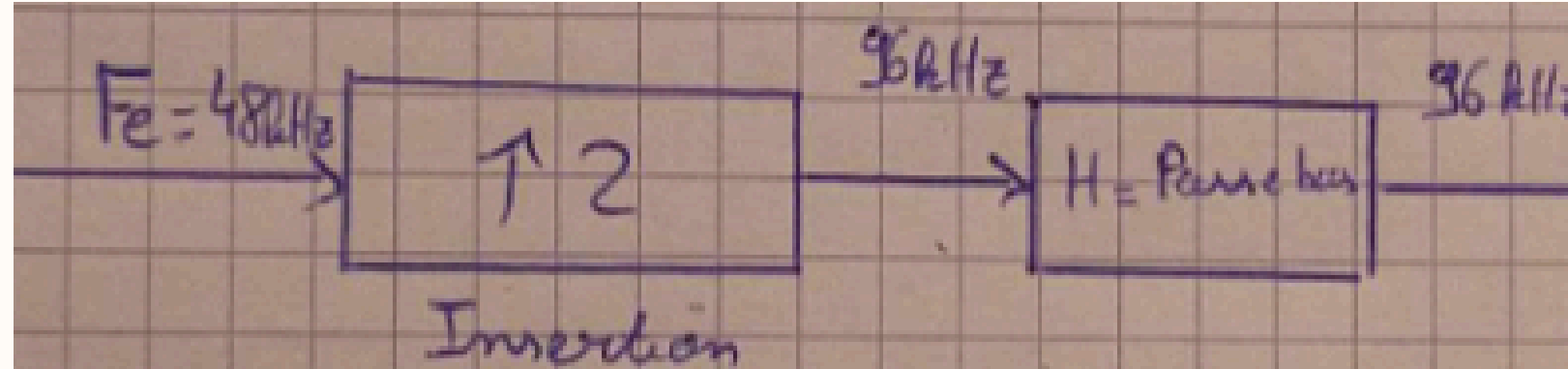
We want to achieve the conversion of sampling rate from $F_s = 48\text{kHz}$ to $F_s = 32\text{kHz}$.

1. Describe and draw the digital processing chain that will permit you to achieve such a conversion (in particular you need to specify the characteristics of the filter $H(z)$ that you will have to use).

Schéma du processus de rééchantillonnage



1. Conversion of sampling rate



On commence par un sur-échantillonnage à l'aide d'une insertion pour doubler la fréquence d'échantillonnage.

- Pour ce faire, on insère des 0 entre deux éléments successifs de notre signal (discrétisé par notre fréquence d'échantillonnage initial).

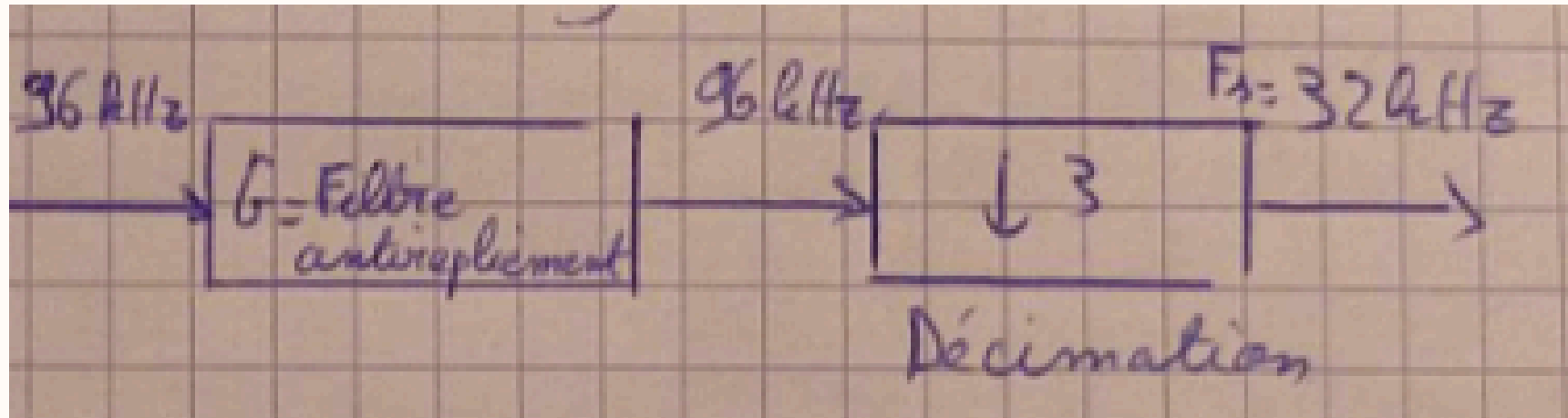
Ensuite, on applique un filtre passe-bas pour retirer les hautes fréquences (pour conserver le théorème de Shannon et conserver les fréquences de notre signal).

Notre nouvelle fréquence d'échantillonnage est 96 kHz .

Filtre passe-bas après sur-échantillonnage :

Handwritten equations for the low-pass filter. The first equation is $H(e^{j2\pi\nu}) = 2 \mathbb{1}_{[-\frac{1}{4}, \frac{1}{4}]}$ with a (ν) superscript. Below it, the frequency range is given as $\nu \in [-\frac{1}{2}, \frac{1}{2}]$. There is a small ~ 2 at the bottom right.

1. Conversion of sampling rate



On souhaite ensuite sous échantillonner le signal pour réduire la fréquence d'échantillonnage de 96 kHz à 32 kHz (rapport 3).

Ce processus s'appelle la décimation, mais pour éviter le repliement spectral, il est nécessaire d'appliquer d'abord un filtre anti repliement.

Filtre anti repliement :

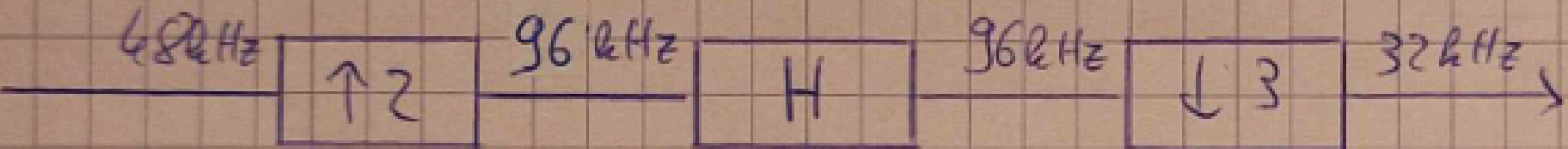
$$G(e^{2i\pi\nu}) = 1 \quad \left[-\frac{1}{6}, \frac{1}{6}\right]$$

Pour la décimation, il suffit alors de prendre un échantillon sur 3, et l'on obtient notre signal rééchantillonné.

1. Conversion of sampling rate

On a alors la chaîne suivante, ainsi que notre filtre qui est le produit du filtre passe-bas après sur-échantillonnage avec le filtre anti repliement. On retrouve aussi la réponse impulsionnelle du filtre par le calcul.

On a ainsi :


$$H(e^{j2\pi\nu}) = 2 \frac{1}{\left[-\frac{1}{6}, \frac{1}{6}\right]} \quad (\nu) \quad \nu \in \left[-\frac{1}{2}, \frac{1}{2}\right]$$
$$h(n) = \int_{-\frac{1}{2}}^{\frac{1}{2}} H(e^{j2\pi\nu}) e^{j2\pi\nu n} d\nu = 2 \int_{-\frac{1}{6}}^{\frac{1}{6}} e^{j2\pi\nu n} d\nu = 2 \left[\frac{e^{j2\pi\nu n}}{j2\pi n} \right]_{-\frac{1}{6}}^{\frac{1}{6}}$$
$$= \frac{2}{3} \operatorname{sinc}\left(\frac{\pi n}{3}\right)$$

1. Conversion of sampling rate

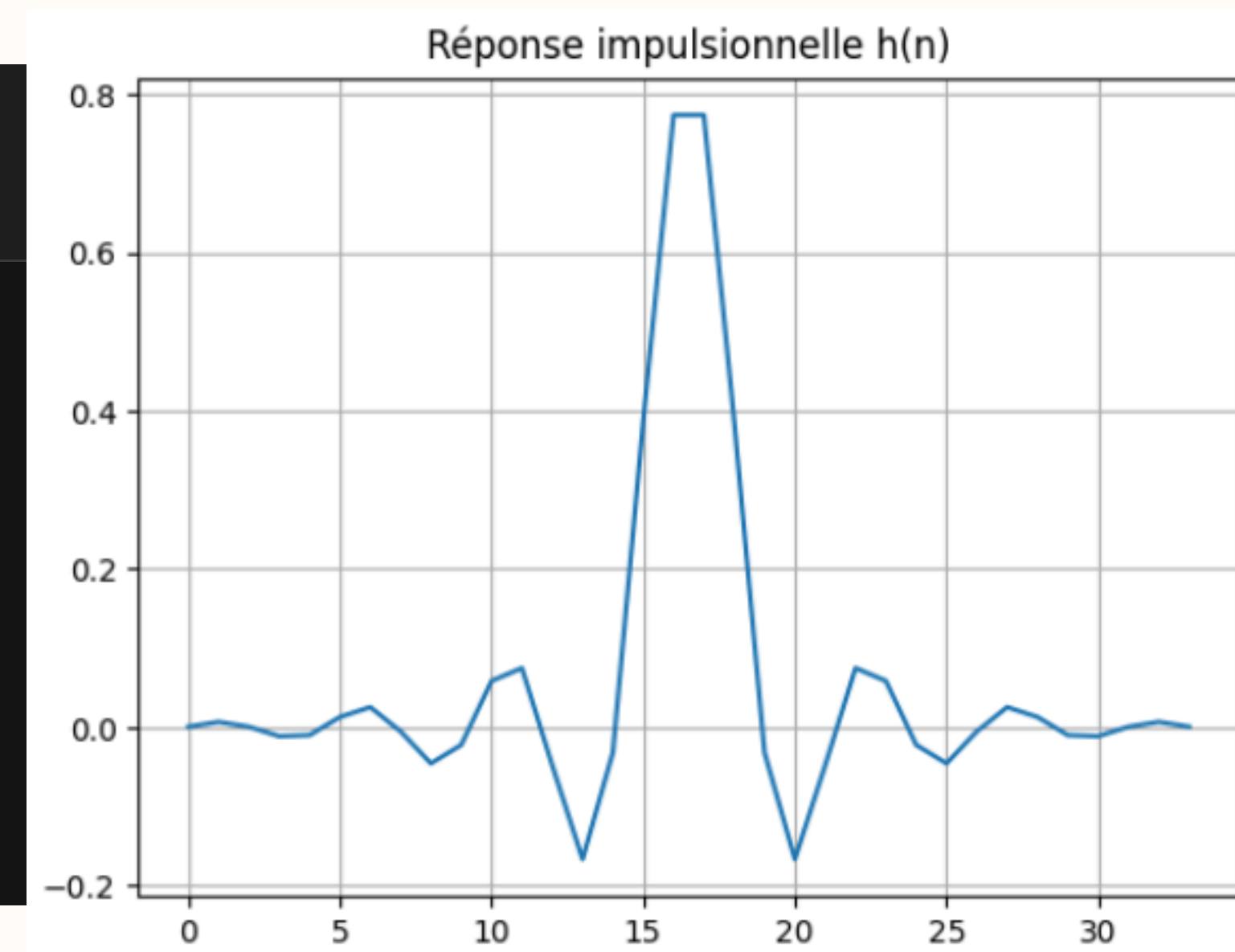
2. Synthesize an impulse response $h(n)$ appropriate for this conversion of sampling rate.

On retrouve une forme similaire à celle du sinus cardinal obtenu précédemment.

Question 2 :

```
# h = scipy.signal.remez(even length, [0, nu_c, nu_a, .5], [L 0])
filter_order = 34
nu_c = 1/6
nu_a = 1/4
L=2
h_remez = scipy.signal.remez(filter_order, [0, nu_c, nu_a, .5], [L, 0])

plt.plot([i for i in range(len(h_remez))], h_remez)
plt.title("Réponse impulsionnelle h(n)")
plt.grid()
plt.axis()
plt.show()
```



1. Conversion of sampling rate

2. Synthesize an impulse response $h(n)$ appropriate for this conversion of sampling rate.

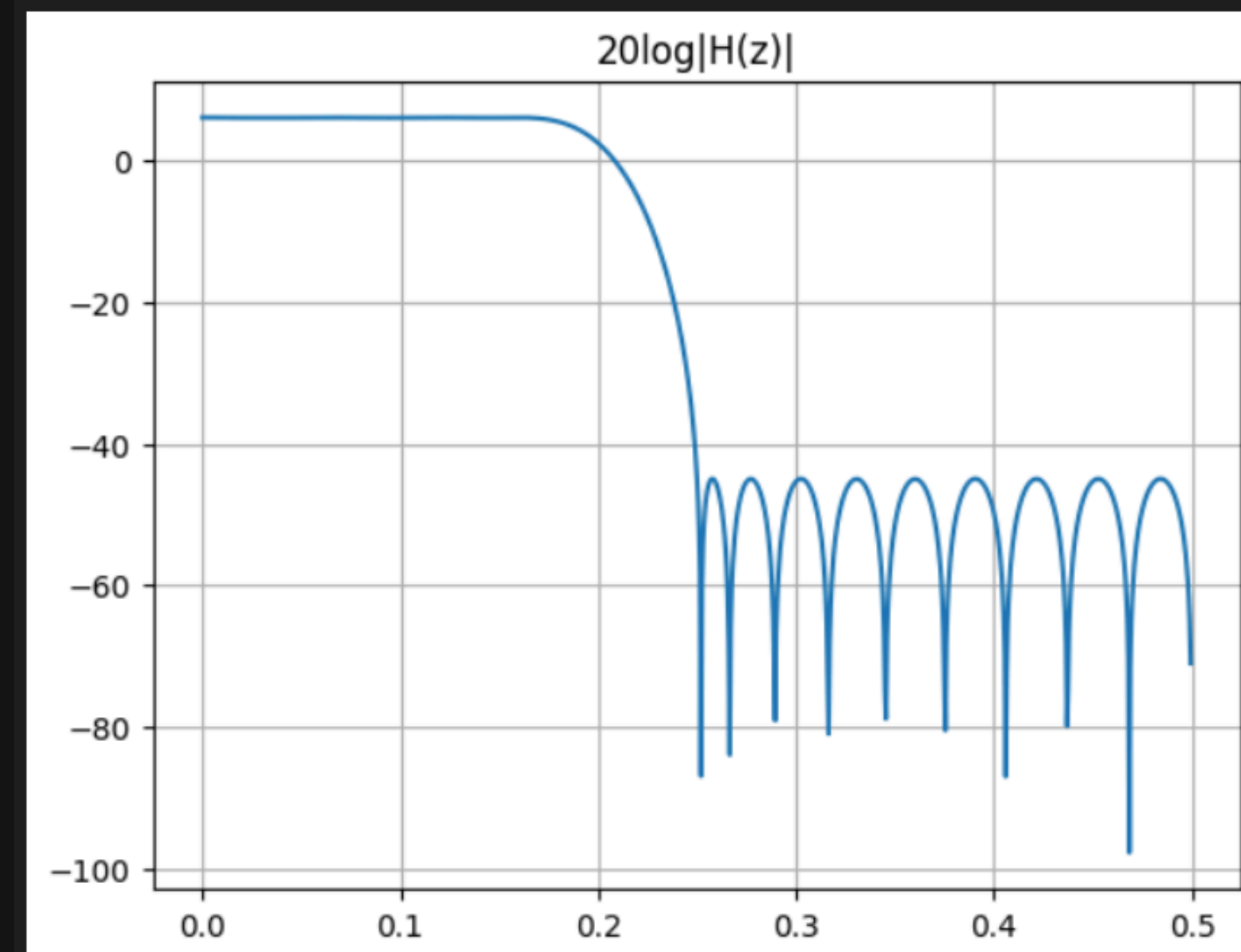
```
# On a  $H(\exp(2i\pi\nu)) = \text{Somme } h(n)\exp(-2i\pi\nu n)$ 
```

```
def abs_H(nu, n = filter_order):  
    r = 0  
    for i in range(n):  
        r += h_remez[i]*np.exp(-2j*np.pi*i*nu)  
    return(np.abs(r))
```

```
N = 1000 # Pas fréquentiel
```

```
X = [i/(2*N) for i in range(N)]  
Y = [20*np.log10(abs_H(i/(2*N))) for i in range(N)]
```

```
plt.plot(X,Y)  
plt.title("20log|H(z)|")  
plt.grid()  
plt.axis()  
plt.show()
```



L'ordre du filtre choisi (ici, 34) a été obtenu en essayant d'obtenir l'écart de 50 dB entre les fréquences passantes et les fréquences coupées.

1. Conversion of sampling rate

3. Program the simplest digital processing chain that will permit you to achieve the conversion. You can use the Matlab function `filter` or the Python function `scipy.signal.lfilter`.

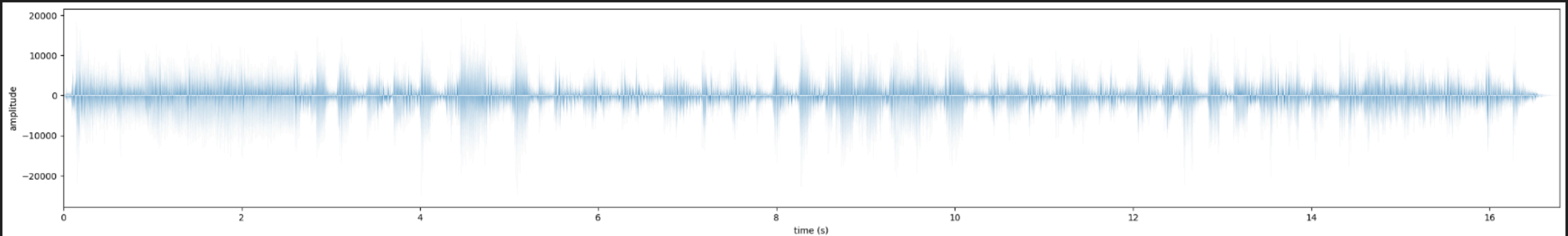
```
# Signal initial

timestep = 1/float(Fs)
times = np.arange(len(x))*timestep
plot_sound(x, times)

play = True
if play :
    play_sound(sound_48kHz)
```

✓ 18.8s

Pyth



1. Conversion of sampling rate

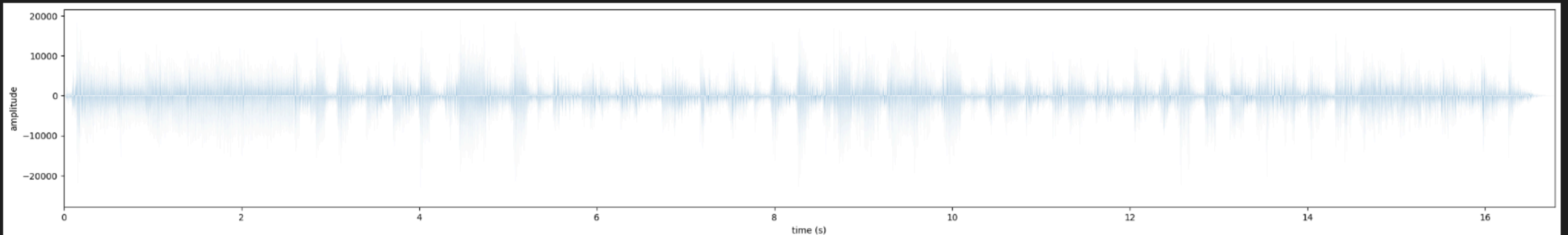
3. Program the simplest digital processing chain that will permit you to achieve the conversion. You can use the Matlab function `filter` or the Python function `scipy.signal.lfilter`.

```
# Etape 1 : Sur-échantillonnage, on effectue une insertion
```

```
y1 = []  
for i in range (len(x)):  
    y1.append(x[i])  
    y1.append(0)  
  
timestep = 1/float(2*Fs)  
times = np.arange(len(y1))*timestep  
plot_sound(y1, times)
```

✓ 6.3s

Pytho

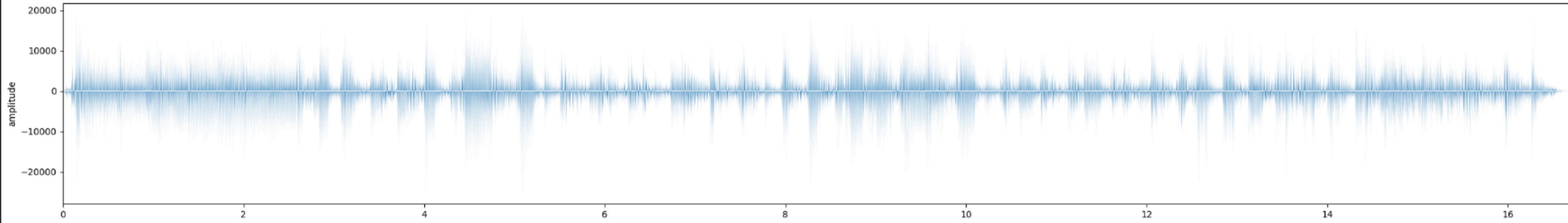


1. Conversion of sampling rate

3. Program the simplest digital processing chain that will permit you to achieve the conversion. You can use the Matlab function `filter` or the Python function `scipy.signal.lfilter`.

```
# Etape 2 : Application du filtre  
  
y2 = scipy.signal.lfilter(h_remez, 1.0, y1)  
  
timestep = 1/float(2*Fs)  
times = np.arange(len(y2))*timestep  
plot_sound(y2, times)
```

✓ 1.7s



1. Conversion of sampling rate

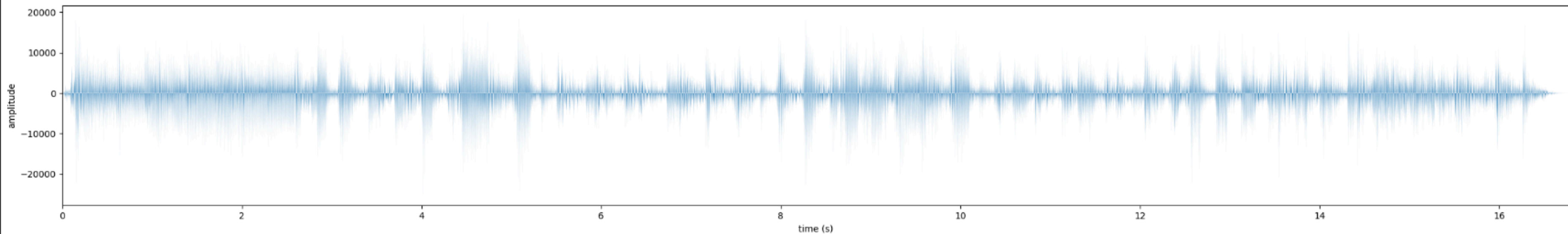
3. Program the simplest digital processing chain that will permit you to achieve the conversion. You can use the Matlab function `filter` or the Python function `scipy.signal.lfilter`.

```
# Etape 3 : Sous-échantillonnage, on effectue une décimation
```

```
y3 = []
for i in range(len(y2)):
    if i%3 == 0:
        y3.append(y2[i])

timestep = 1/float(2*Fs/3)
times = np.arange(len(y3))*timestep
plot_sound(y3, times)
```

✓ 1.1s



1. Conversion of sampling rate

3. Program the simplest digital processing chain that will permit you to achieve the conversion. You can use the Matlab function `filter` or the Python function `scipy.signal.lfilter`.



```
# On écrit le signal dans un nouveau fichier wav

write('caravan_32kHz.wav', 32000, np.array(y3, dtype=np.int16)) # Fs = 32kHz fréquence d'échantillonnage finale

# On l'écoute

data_path = os.getcwd()
filename = 'caravan_32kHz.wav'
sound_32kHz = os.path.join(data_path, filename)
play = True
if play:
    play_sound(sound_32kHz)
/ 17.8s
```

On observe que le signal rééchantillonné semble inchangé à l'écoute. Cependant, à l'enregistrement, le fichier a été compressé : les hautes fréquences (peu ou pas audibles) ont été supprimées du signal. La taille du fichier passe de 1574 Ko à 1050 Ko (on peut notamment remarquer que le rapport des tailles des fichiers correspond plus ou moins au rapport de la fréquence d'entrée avec celle de sortie : 3/2)

 caravan_32kHz.wav	03/10/2024 23:33	Fichier WAV	1 050 Ko
 caravan_48khz.wav	08/01/2013 09:13	Fichier WAV	1 574 Ko

1. Conversion of sampling rate

4. Check and exploit the equivalence between the two diagrams of Figure 1.

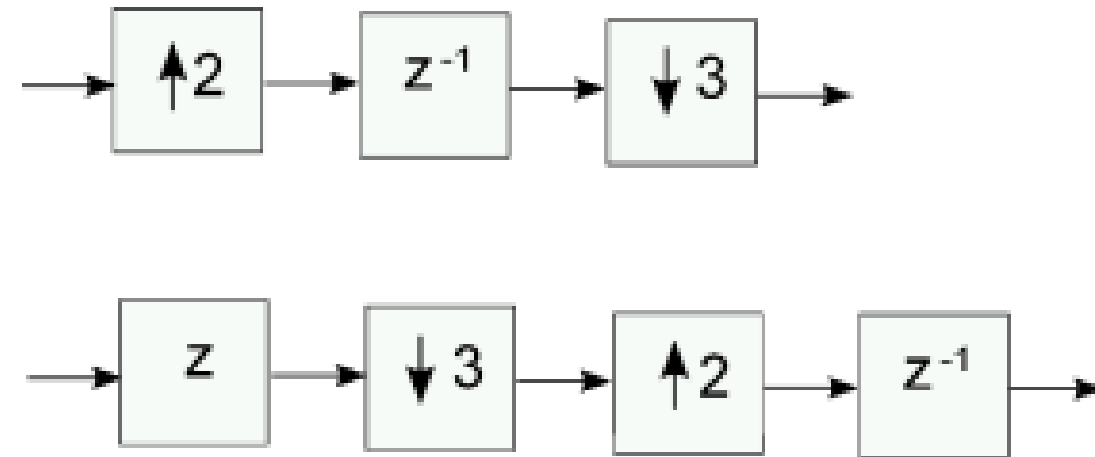
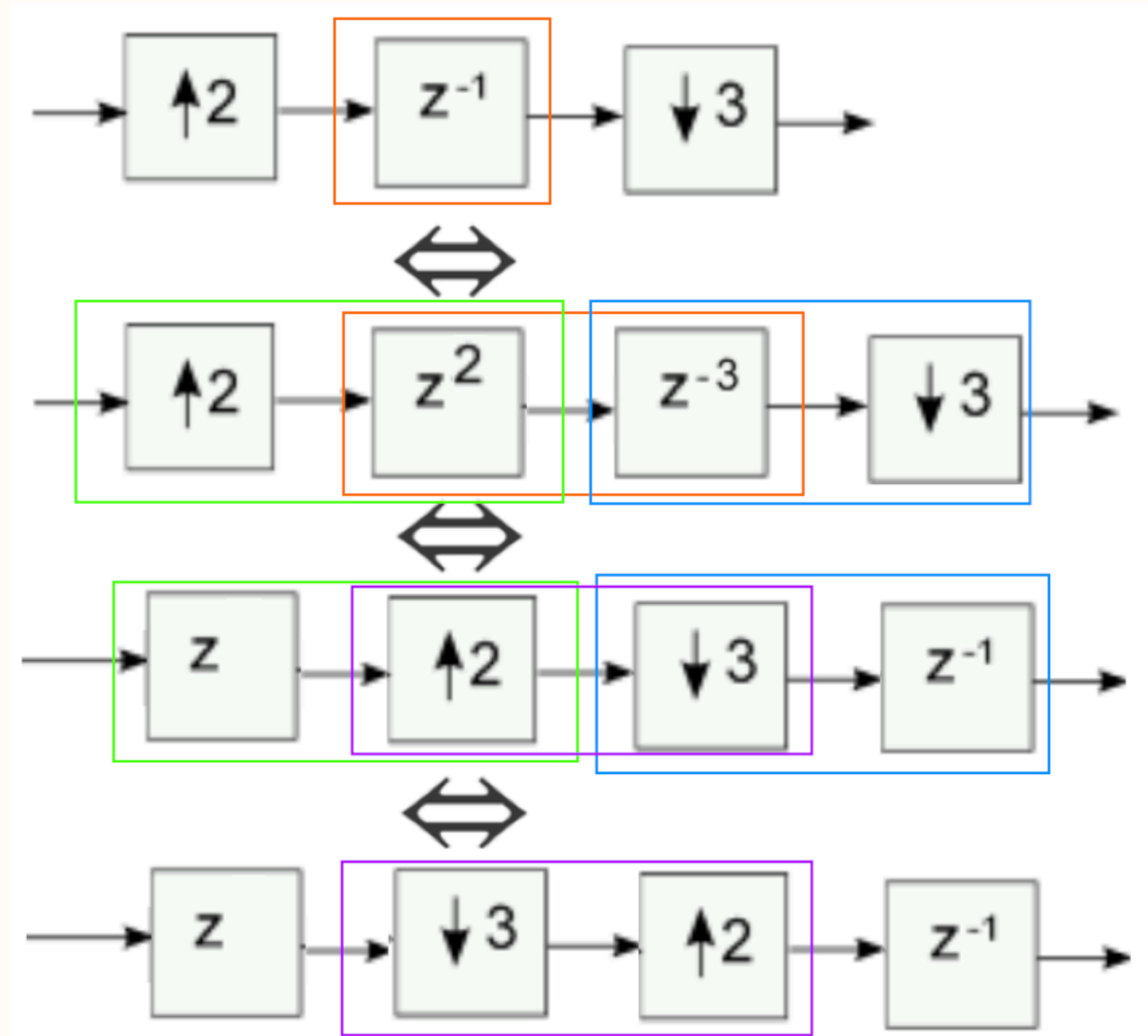


Figure 1: Equivalence

Sur le prochain schéma qui démontre l'équivalence :

- Les rectangles verts représentent des identités nobles pour changer les positions des processus d'insertion et de décimation
- Les rectangles violets représentent l'utilisation de l'identité noble pour intervertir l'inversion et la décimation, étant donné que **$3^2 = 1$** (3 et 2 sont premiers entre eux).

1. Conversion of sampling rate



1. Conversion of sampling rate

5. Program the efficient implementation of this conversion of sampling rate by using two successive polyphase decompositions.
6. Compare the results obtained with the two implementations and compare the running times of the two approaches (you can use the Matlab functions `tic` and `toc`, or the Python function `time.time()`).

```
def shift(u,k=1): # Décalage de k indice la suite u
    L = [0]*k
    return(np.concatenate((L,u[:-k])))

def decimation(u,k): # Décimation du signal u d'un rapport k
    v = []
    for i in range (len(u)):
        if i%k == 0:
            v.append(u[i])
    return(v)

def insertion(u,k): # Décimation du signal u d'un rapport k
    v = []
    for i in range (len(u)):
        v.append(u[i])
        for j in range(k):
            v.append(0)
    return(v)
```

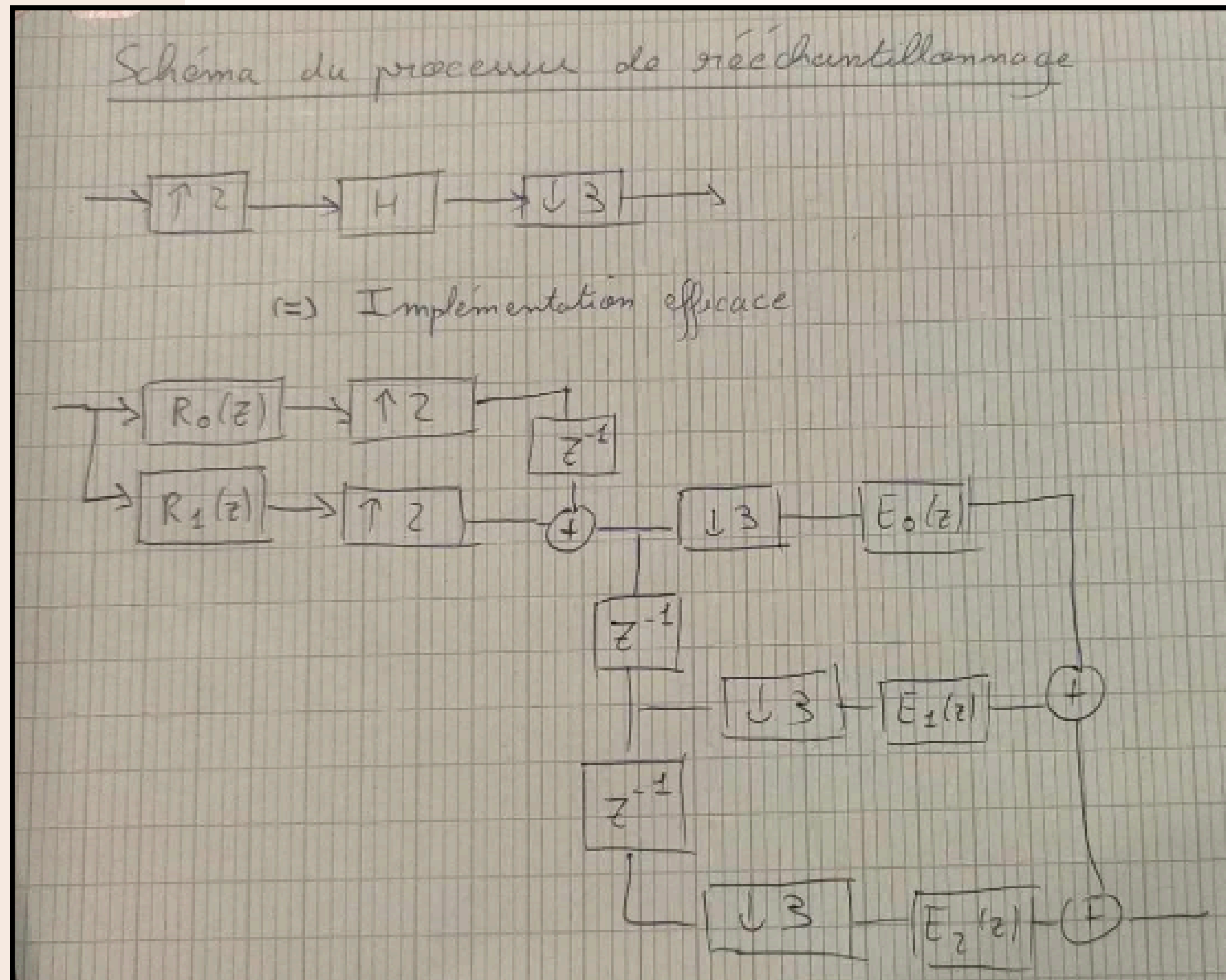
On définit les opérateurs de décalage (shift) pour traduire le produit par z^{-1} , de décimation et d'insertion.

Il est important d'appliquer les opérateurs de shift et de décimation à la réponse impulsionnelle du filtre obtenu par la fonction :

`h = scipy.signal.remez`

Pour obtenir les fonctions E0, E1 et E2, ainsi que R0 et R1.

1. Conversion of sampling rate



Les schémas équivalents pour nous donner l'implémentation efficace du rééchantillonnage est ci-contre. Je n'ai malheureusement pas réussi à l'exploiter.

(Tentative de code sur le Jupyter Notebook)

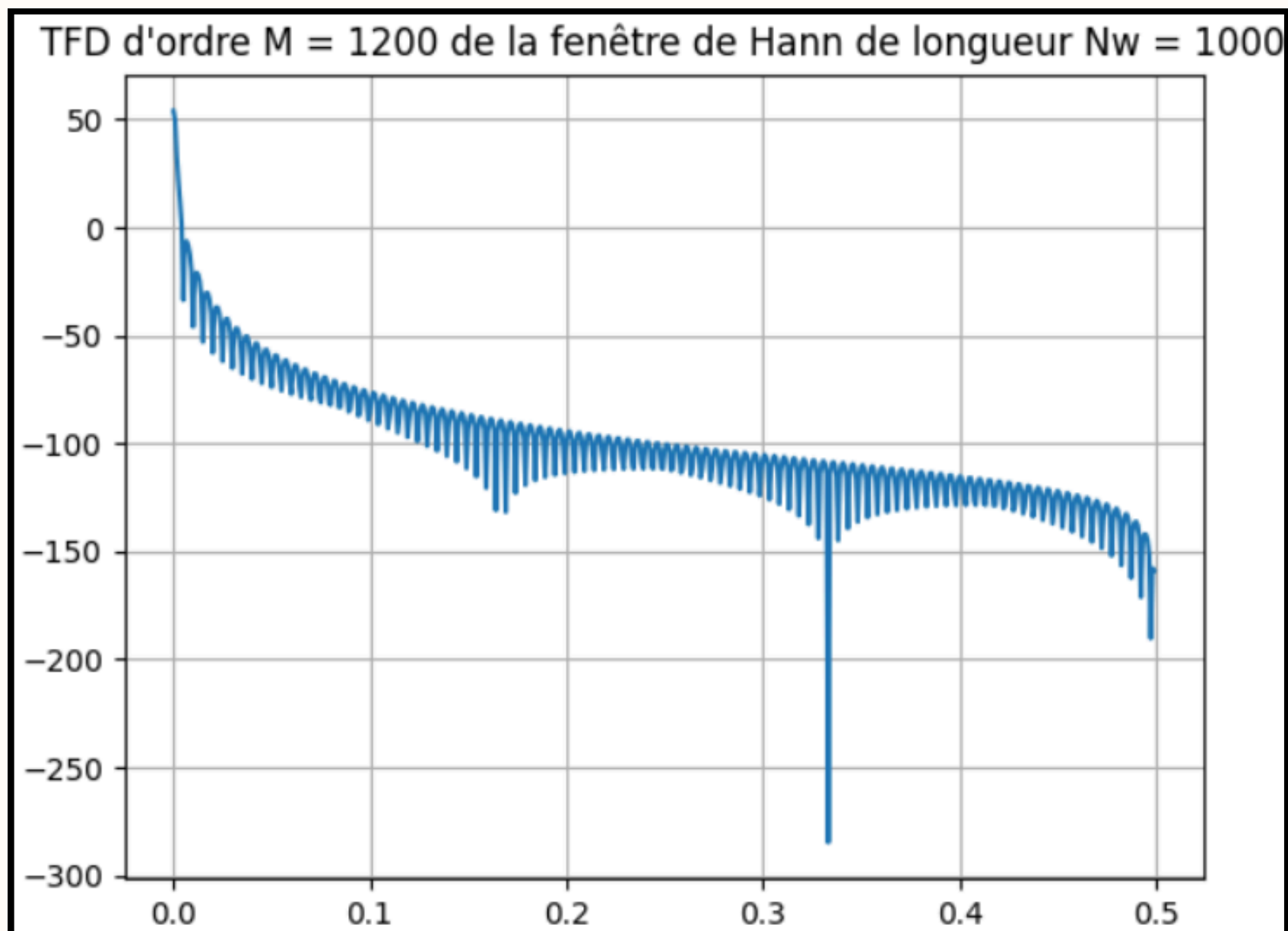
2. STFT audio equalization

The definition of the STFT that is referred to as "low-pass convention" is given in discrete time by :

$$W_x(\lambda, b) = \sum_{n \in \mathbb{Z}} x(n)w(n-b)e^{-j2\pi\lambda n}, \quad (1)$$

where $w(n)$ is the analysis window in discrete time, which is supposed summable, real and symmetric.

1. We choose $w(n)$ as a Hann (Hanning) window of length N_w . Plot the DFT of w , i.e. $\hat{w}(k/M)$ where $M, M \geq N_w$ is the order of the DFT. What is the width of the main lobe as a function of N_w ?



2. STFT audio equalization

The definition of the STFT that is referred to as "low-pass convention" is given in discrete time by :

$$W_x(\lambda, b) = \sum_{n \in \mathbb{Z}} x(n)w(n-b)e^{-j2\pi\lambda n}, \quad (1)$$

where $w(n)$ is the analysis window in discrete time, which is supposed summable, real and symmetric.

1. We choose $w(n)$ as a Hann (Hanning) window of length N_w . Plot the DFT of w , i.e. $\hat{w}(k/M)$ where $M, M \geq N_w$ is the order of the DFT. What is the width of the main lobe as a function of N_w ?

```
# Pour trouver la largeur du lobe principal, on regarde la valeur maximale (en 0)
seuil = Y[0] - 3 # Niveau à -3 dB

for i in range(len(X)) :
    if Y[i] < Y[0] - 3:
        bande = 2*X[i]
        break

print("Largeur du lobe à -3dB en fonction de Nw :")
print(f"Valeur théorique : {2/Nw}")
print(f"Valeur obtenue graphiquement : {bande}")
```

Largeur du lobe à -3dB en fonction de N_w :

Valeur théorique : 0.002

Valeur obtenue graphiquement : 0.00166666666666666668

On obtient l'égalité
lorsque $M=N$
Valeur théorique : $2/Nw$

2. STFT audio equalization

2. Note that the expression (1), taken at fixed λ , can be written as a convolution and deduce an interpretation of the STFT in terms of filtering. Explain the role of the corresponding filter (low-pass? band-pass? high-pass?). As a linear phase FIR filter, specify its type (type 1,2,3,4?) according to its length (even or odd).

Pour déterminer si le type de filtre correspondant, on sait que notre fenêtre est de longueur paire N_w et est symétrique. Le filtre à réponse impulsionnelle finie obtenue après la STFT est donc elle aussi symétrique et de longueur paire, c'est donc un filtre de type 2.

Je n'ai pas réussi dans le temps imparti pour le TP à programmer la fonction qui correspond au filtre.

Serait-il possible d'avoir un corrigé du TP ?