

Capacité numérique

Réaliser, à l'aide d'un langage de programmation, un filtrage numérique passe-bas d'un signal issu d'une acquisition et mettre en évidence la limitation introduite par l'échantillonnage.

On s'intéresse dans ce document au filtrage numérique de l'enregistrement de deux notes jouées successivement au piano (Ré de l'octave six (2349 Hz)¹ puis Do de l'octave deux (131 Hz)). Nous étudierons plus particulièrement l'influence de la fréquence d'échantillonnage.

Le signal sonore, d'une durée de 3 s, est, dans un premier temps, échantillonné à la fréquence $f_e = 44,1$ kHz. Le filtre numérique utilisé est synthétisé à partir d'un filtre analogique passe bas du premier ordre via la méthode d'Euler.

L'objectif est d'atténuer la note la plus aigüe (Ré) sans modifier significativement la note la plus grave (Do) ainsi que de déterminer si le filtre se comporte comme attendu pour des fréquences d'échantillonnages plus faibles.

Les scripts Python utilisés sont retranscrits dans les annexes A et B du présent document. Les versions exécutables des scripts, le fichier audio utilisé, ainsi que ceux générés suite aux rééchantillonnage et filtrages sont également joints à cette ressource.

A. Introduction au filtrage numérique sur un exemple simple

Avant de s'intéresser au filtrage numérique de l'enregistrement audio, nous nous proposons d'étudier le filtrage d'un signal élémentaire, de durée $\Delta t = 0,2$ s, se décomposant en un signal sinusoïdal de fréquence $f = 30$ Hz et d'amplitude $A = 1$ V, superposé à un « bruit » de plus haute fréquence $f_b (> f)$ et d'amplitude $A_b = 0,6$ V :

$$e(t) = A \sin(2\pi ft) + A_b \sin(2\pi f_b t)$$

Le signal $e(t)$ est alors échantillonné à la fréquence $f_e = \frac{1}{T_e} = 2000$ Hz et les valeurs des $N_e = 1 + \left\lfloor \frac{\Delta t}{T_e} \right\rfloor$ échantillons sont données par la relation :

$$\forall k \in [0, N_e - 1] \quad e[k] = A \sin(2\pi fkT_e) + A_b \sin(2\pi f_b kT_e)$$

¹ Comme on le verra plus loin, le piano étant légèrement désaccordé, la fréquence est en fait plus proche de 2360 Hz.

Considérons un premier cas où $f_b = 290$ Hz (le critère de Shannon est alors vérifié : $f_e > 2f_b$) et traçons la représentation temporelle de $e(t)$ ainsi et son spectre² sur l'intervalle $[0, f_e]$.

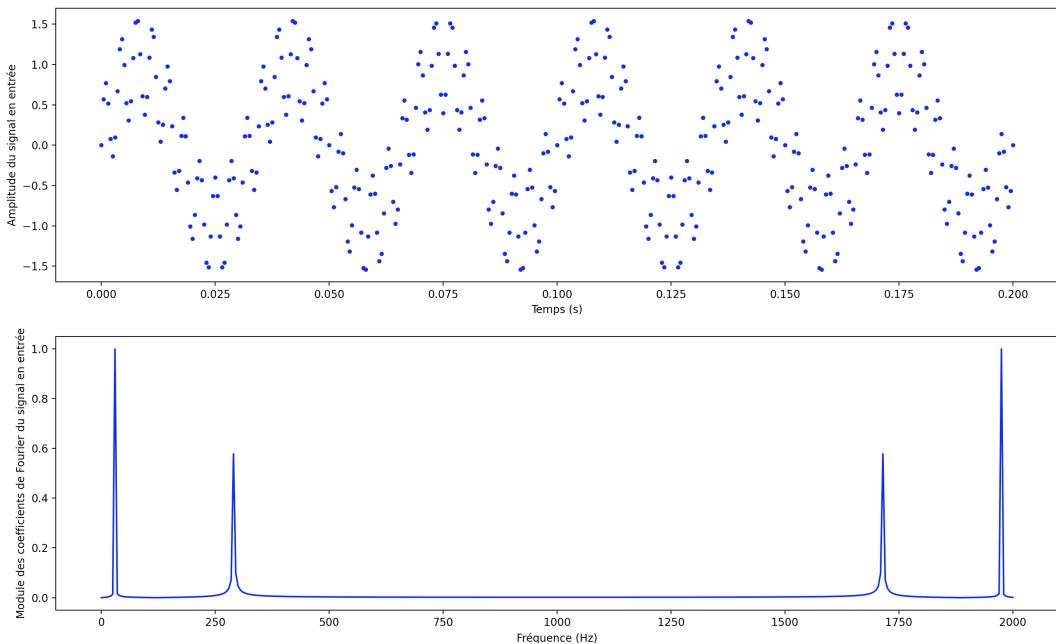


FIGURE 1 – Représentations du signal en entrée du filtre numérique
($f = 30$ Hz, $f_b = 290$ Hz, $f_e = 2000$ Hz)

On observe bien les répliques spectrales aux fréquences $f_e - f = 1970$ Hz et $f_e - f_b = 1710$ Hz dues à l'échantillonnage. L'ensemble du spectre utile est contenu dans le domaine $[0, f_e/2]$.

Afin de synthétiser le filtre numérique, considérons l'équation différentielle correspondant à un filtre analogique passe bas du premier ordre :

$$e(t) = s(t) + \frac{1}{2\pi f_c} \frac{ds}{dt}(t)$$

$s(t)$ et $e(t)$ étant respectivement les signaux en entrée et sortie du filtre et f_c sa fréquence de coupure. Dans la suite, on choisit $f_c = 40$ Hz.

La méthode d'Euler explicite permet d'écrire :

$$e[k] = s[k] + \frac{1}{2\pi f_c} \frac{s[k+1] - s[k]}{T_e}$$

Ce qui conduit au schéma numérique :

$$s[k+1] = (1 - 2\pi f_c T_e)s[k] + 2\pi f_c T_e e[k]$$

Il est alors possible de générer le signal numérique $s[k]$ et de tracer ses représentations.

² Pour ce faire, on pourra utiliser le module `numpy.fft`.

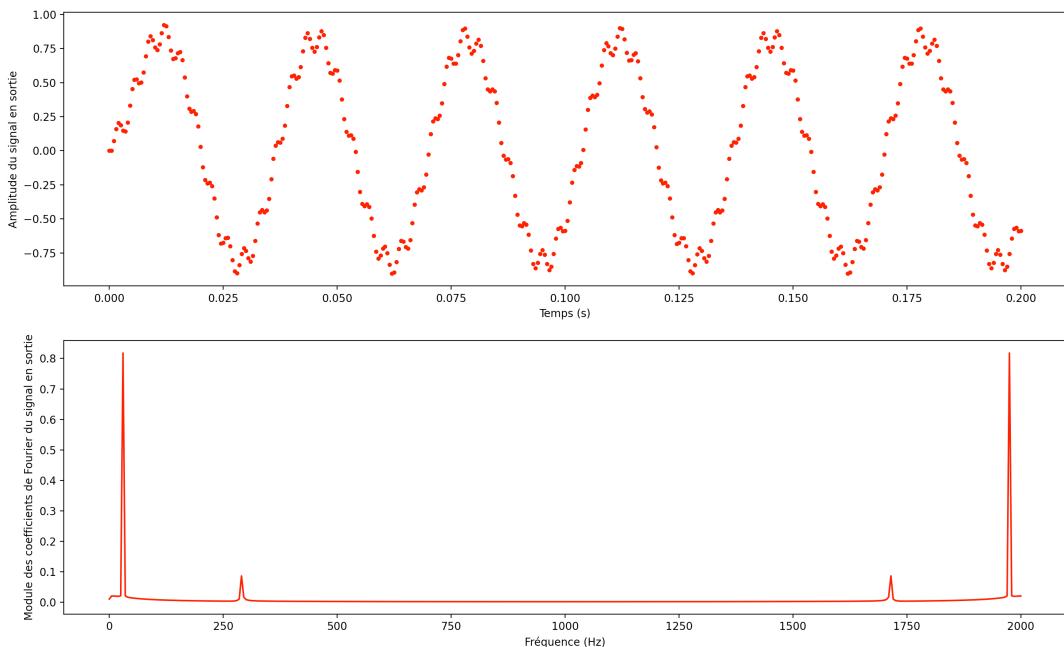


FIGURE 2 – Représentations du signal en sortie du filtre numérique
($f = 30 \text{ Hz}$, $f_b = 290 \text{ Hz}$, $f_e = 2000 \text{ Hz}$, $f_c = 40 \text{ Hz}$)

On observe une réduction significative du bruit et on en conclut que dans le cas où le critère de Shannon est respecté, le filtre numérique joue effectivement son rôle de passe-bas.

Reprendons la simulation précédente dans le cas où $f_b = 1990 \text{ Hz}$. Désormais, **le critère de Shannon n'est plus vérifié** ($f_e < 2f_b$) et l'échantillonnage fait apparaître, entre autres, un pic à une fréquence de $f_e - f_b = 10 \text{ Hz}$:

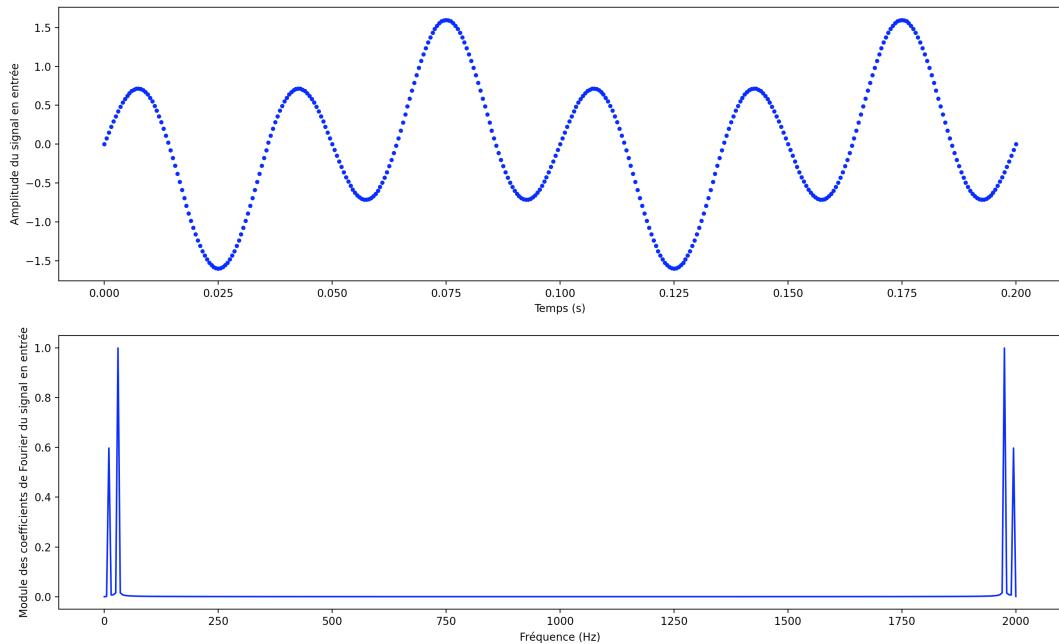


FIGURE 3 – Représentations du signal en entrée du filtre numérique
($f = 30 \text{ Hz}$, $f_b = 1990 \text{ Hz}$, $f_e = 2000 \text{ Hz}$)

L'action du filtre numérique permet de générer le signal ci-dessous :

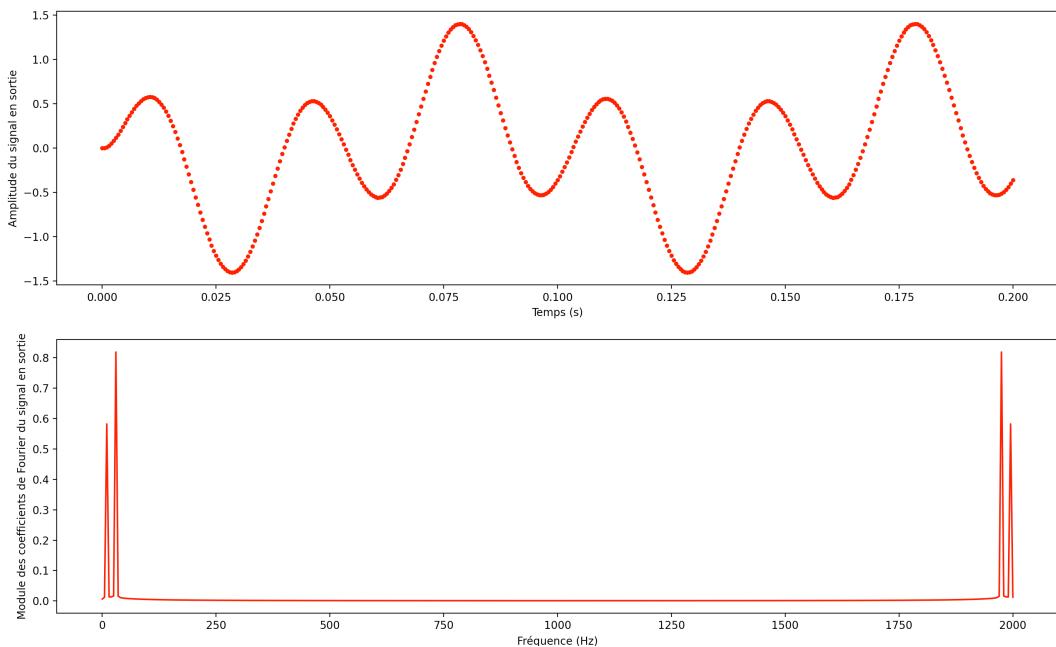


FIGURE 4 – Représentations du signal en sortie du filtre numérique
($f = 30$ Hz, $f_b = 1990$ Hz, $f_e = 2000$ Hz, $f_c = 40$ Hz)

Comme on pouvait s'y attendre, l'action d'un filtre de fréquence de coupure $f_c = 40$ Hz n'a quasiment aucun effet : Le filtre ne joue plus son rôle de passe bas au sens où il ne permet plus de réduire le bruit de fréquence $f_b = 1990$ Hz.

B. Filtrage numérique d'un enregistrement sonore

On s'intéresse désormais au filtrage de l'enregistrement³ de deux notes jouées successivement au piano (Ré de l'octave six (2349 Hz) puis Do de l'octave deux (131 Hz)) et échantillonné à la fréquence $f_e = 44,1$ kHz (Le critère de Shannon est ici largement respecté).

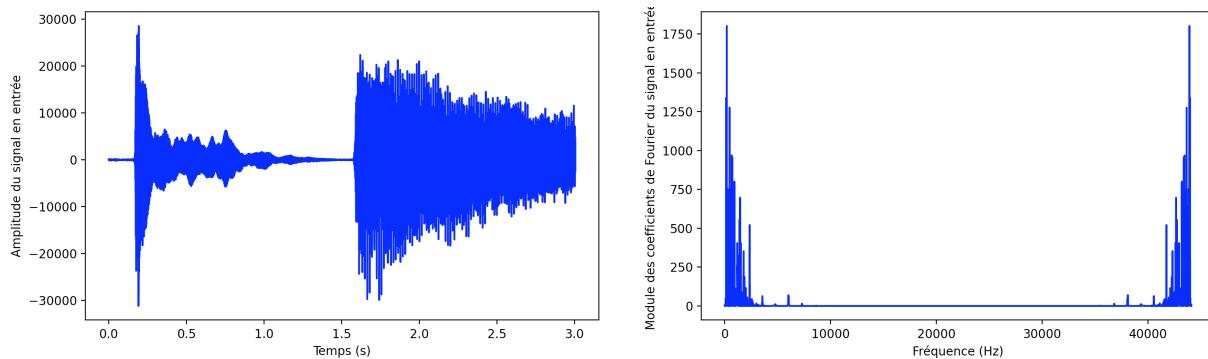


FIGURE 5 – Représentation temporelle et spectre de l'enregistrement audio ($f_e = 44,1$ kHz)

³ On pourra utiliser le module `scipy.io.wavfile` pour lire l'enregistrement audio stocké sur le disque au format WAV.

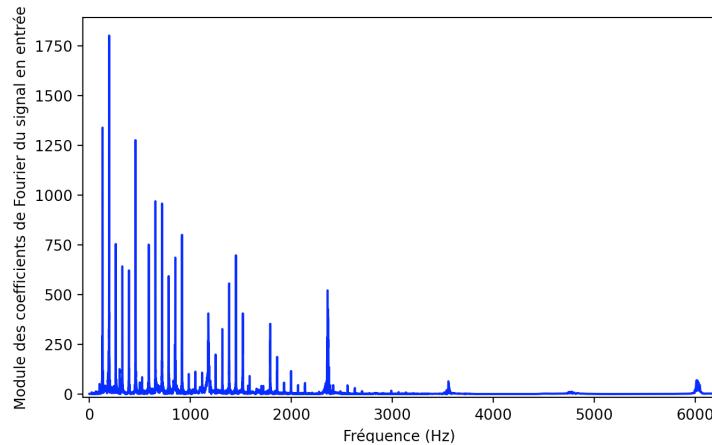


FIGURE 6 – Agrandissement du spectre de l'enregistrement audio ($f_e = 44,1$ kHz)

De façon prévisible, le spectre fait apparaître un pic significatif à 131 Hz (correspondant à la note Do) ainsi qu'un autre proche de 2349 Hz (en fait 2360 Hz, correspondant à la note Ré). De multiples harmoniques sont également présentes. Notons la présence d'un pic à 1180 Hz correspondant au fondamental de la note Ré.

Le filtre introduit dans la partie précédente est utilisé en fixant la fréquence de coupure à 720 Hz. On trace alors le signal en sortie du filtre numérique que l'on peut comparer au signal audio de départ.

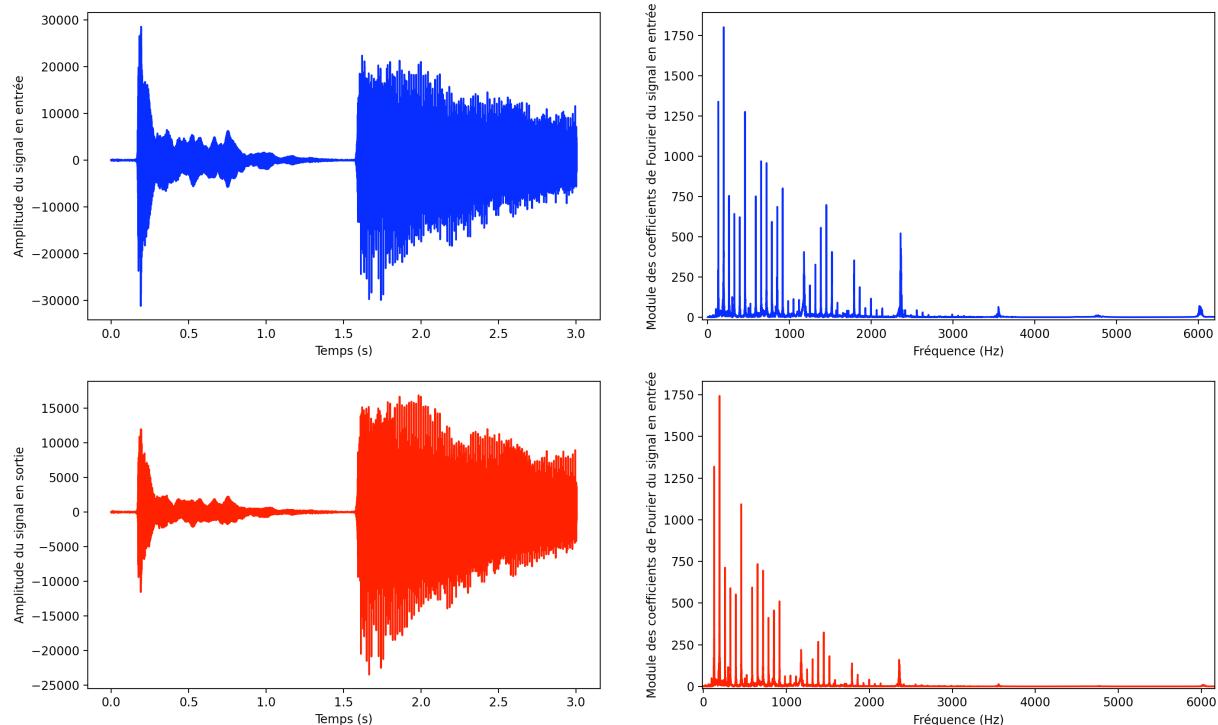


FIGURE 7 – Comparaison des signaux en entrée et sortie du filtre numérique ($f_e = 44,1$ kHz)

Au niveau de la représentation temporelle, on observe immédiatement que l'amplitude du Ré est divisée d'un facteur 3 environ, tandis que celle du Do n'est diminuée que d'un facteur d'environ

1,3, confirmant ainsi le caractère passe bas du filtre numérique. Au niveau des spectres, on constate également que les pics correspondant au Ré sont significativement réduits.

On peut enfin générer⁴ un fichier audio à partir du signal en sortie du filtre et se rendre compte, à l'écoute, que l'atténuation est plus importante pour la note la plus aiguë (l'effet est cependant atténué par la réponse non linéaire de l'oreille humaine).

Là encore, le filtre joue effectivement son rôle de passe bas, le critère de Shannon ayant bien été respecté lors de l'échantillonnage de l'enregistrement audio.

On peut illustrer une nouvelle fois qu'en cas de non-respect du critère, le signal en sortie n'est plus celui attendu (où seule la note Ré est significativement atténuée).

Fixons désormais la fréquence d'échantillonnage à $f_e = 2450$ Hz. On obtient alors les formes suivantes pour le signal et son spectre :

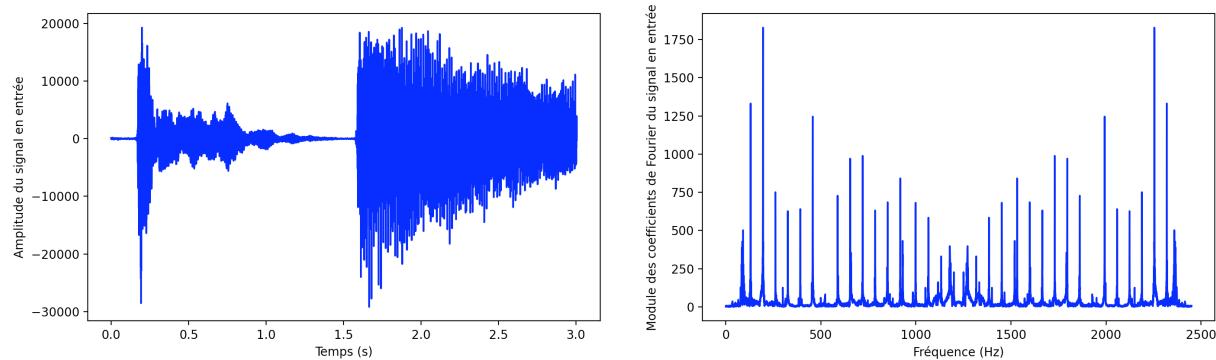


FIGURE 8 – Représentation temporelle et spectre de l'enregistrement audio ($f_e = 2450$ Hz)

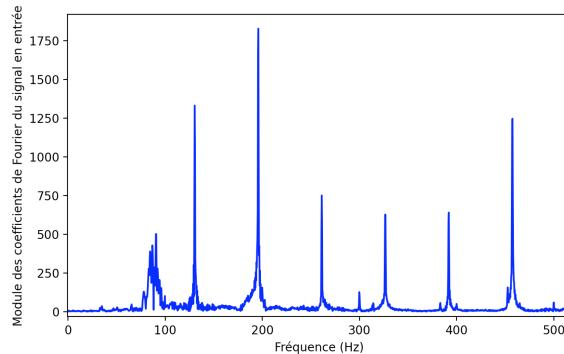


FIGURE 9 – Agrandissement du spectre de l'enregistrement audio ($f_e = 2450$ Hz)

On observe, entre autres, l'apparition d'un pic autour de 90 Hz correspondant à une réplique spectrale du pic du Ré à 2360 Hz échantillonné à 2450 Hz.

En générant un fichier audio à partir de l'enregistrement ré échantillonné à $f_e = 2450$ Hz, on constate que la première note est désormais plus grave⁵.

⁴ On pourra utiliser, ici aussi, le module `scipy.io.wavfile`.

⁵ L'opération d'échantillonnage ayant significativement réduit l'intensité sonore, on veillera à augmenter le volume.

Notons qu'à une fréquence d'échantillonnage aussi basse, la méthode d'Euler explicite n'est pas assez précise pour donner des résultats satisfaisants et on lui préférera la méthode d'Euler implicite :

$$e[k] = s[k] + \frac{1}{2\pi f_c} \frac{s[k] - s[k-1]}{T_e}$$

Ce qui conduit au schéma numérique :

$$s[k] = \frac{1}{(1 + 2\pi f_c T_e)} s[k-1] + \frac{2\pi f_c T_e}{(1 + 2\pi f_c T_e)} e[k]$$

Il est alors possible de générer le signal numérique $s[k]$ et de comparer les signaux en entrée et sortie du filtre.

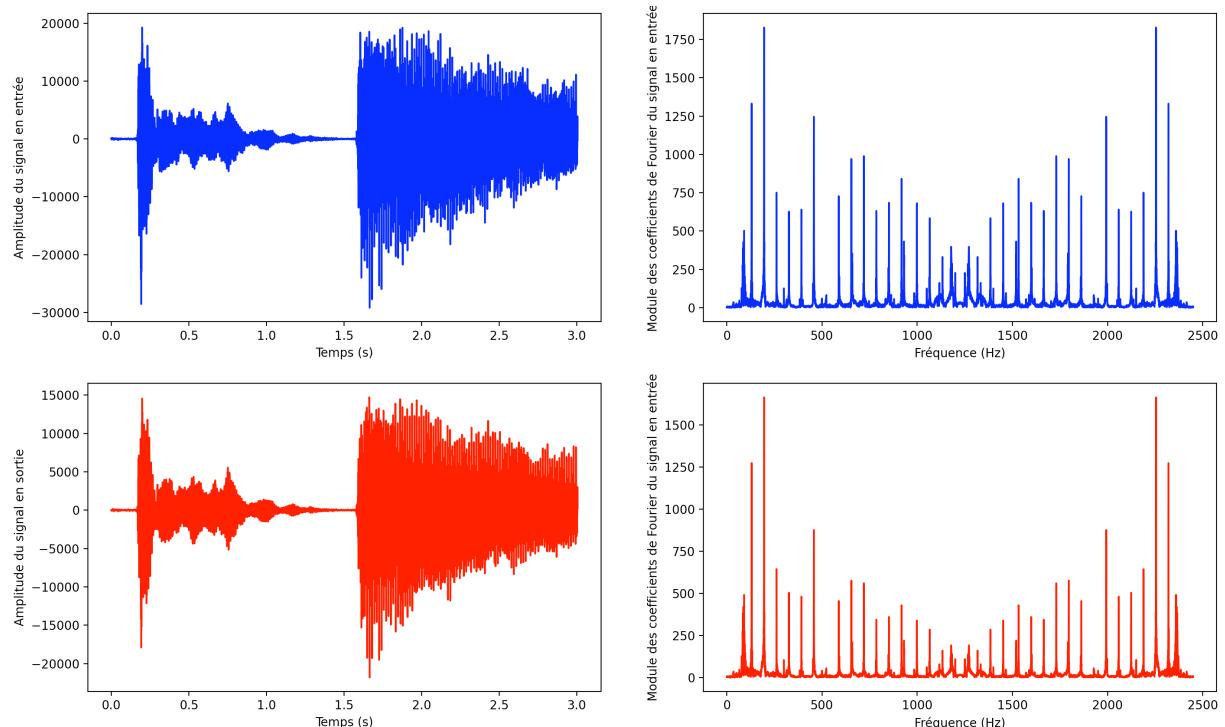


FIGURE 10 – Comparaison des signaux en entrée et sortie du filtre numérique ($f_e = 2450$ Hz)

On constate que contrairement à la situation précédente, l'amplitude des deux notes a été réduite dans des proportions équivalentes.

Annexe A : Script Python (Exemple simple)

```

1 ## Importation des bibliothèques
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6
7 ## Génération d'un signal sinusoïdal bruité échantillonné
8
9 A = 1 # Amplitude du signal
10 f = 30 # Fréquence (en Hertz) du signal sinusoïdal
11 Ab = 0.6 # Amplitude du bruit
12 fb = 1990 # Fréquence (en Hertz) du bruit # On peut choisir 1990 pour
13 # illustrer la limitation due à l'échantillonnage
14
15 duree_signal = 0.2 # Durée du signal en seconde
16 fe = 2000 # Fréquence d'échantillonnage
17 Te = 1 / fe # Période d'échantillonnage
18 Ne = 1 + int(duree_signal / Te) # Nombres d'échantillons
19
20 temps = np.array([i * Te for i in range(Ne)])
21 signal_entree = np.array([A*np.sin(2*np.pi*f*Te*i) +
22 Ab*np.sin(2*np.pi*fb*Te*i) for i in range(Ne)]) # Génération d'un signal
23 # bruité
24
25 ## Vérification du critère de Shannon
26
27 print("La fréquence du signal sinusoïdal est ", f, " Hz")
28 print("La fréquence du bruit sinusoïdal est ", fb, " Hz")
29 print("La fréquence d'échantillonnage est ", fe, " Hz")
30
31 if fe > 2*f and fe > 2*fb :
32     print("Le critère de Shannon est bien vérifié")
33 else :
34     print("Le critère de Shannon n'est pas vérifié")
35
36 ## Affichage du signal d'entrée échantillonnée et de son spectre
37 plt.figure()
38 plt.subplot((211))
39 plt.plot(temps, signal_entree, 'b.')
40 plt.xlabel('Temps (s)')
41 plt.ylabel('Amplitude du signal en entrée')
42
43 FFT_signal_entree = np.fft.fft(signal_entree)
44 coef_fourier_entree = np.concatenate((FFT_signal_entree[0] / Ne,
45 FFT_signal_entree[1:] * 2 / Ne), axis = None) # Permet d'obtenir les
46 # bonnes amplitudes sur le spectre
47 module_coef_fourier_entree = np.absolute(coef_fourier_entree)
48
49 freq = np.array([i/duree_signal for i in range(Ne)]) # L'écart
50 # fréquentiel entre deux valeurs du spectre est 1/duree_signal
51
52 plt.subplot((212))
53 plt.plot(freq, module_coef_fourier_entree, 'b')
54 plt.xlabel('Fréquence (Hz)')
55 plt.ylabel('Module des coefficients de Fourier du signal en entrée')

```

```

54 ## Filtre numérique passe bas d'ordre 1
55
56 fc = 40 # Fréquence de coupure
57
58 signal_sortie = np.zeros(len(signal_entree))
59
60 for i in range(Ne - 1):
61     signal_sortie[i+1] = signal_sortie[i] * (1 - 2*np.pi*fc/fe) +
62     signal_entree[i] * (2*np.pi*fc/fe) # Schéma Euler explicite
63     #signal_sortie[i+1] = signal_sortie[i] / (1 + 2*np.pi*fc/fe) +
64     signal_entree[i-1] * ((2*np.pi*fc/fe) / (1 + 2*np.pi*fc/fe)) # Schéma
65     Euler implicite
66
67 ## Affichage du signal de sortie échantillonnée et de son spectre
68
69 plt.figure()
70 plt.subplot((211))
71 plt.plot(temp, signal_sortie, 'r.')
72 plt.xlabel('Temps (s)')
73 plt.ylabel('Amplitude du signal en sortie')
74
75 FFT_signal_sortie = np.fft.fft(signal_sortie)
76 coef_fourier_sortie = np.concatenate((FFT_signal_sortie[0] / Ne,
77 FFT_signal_sortie[1:] * 2 / Ne), axis = None) # Permet d'obtenir les
78 bonnes amplitudes sur le spectre
79 module_coef_fourier_sortie = np.absolute(coef_fourier_sortie)
80
81 freq = np.array([i/duree_signal for i in range(Ne)])
82
83 plt.subplot((212))
84 plt.plot(freq, module_coef_fourier_sortie, 'r')
85 plt.xlabel('Fréquence (Hz)')
86 plt.ylabel('Module des coefficients de Fourier du signal en sortie')
87
88 ## Affichage des courbes
89
90 plt.show()

```

Annexe B : Script Python (Acquisition audio)

```

1 ## Importation des bibliothèques
2
3 import numpy as np
4 import scipy.io.wavfile as wav
5 import matplotlib.pyplot as plt
6
7
8 ## Affichage du signal audio échantillonné et de son spectre
9
10 fe_origine, signal_entree_origine = wav.read("notes_piano.wav") # fe est
la fréquence d'échantillonnage # Attention à spécifier le chemin d'accès
du fichier "notes_piano.wav"
11
12 fe = 2450 # Nouvelle fréquence d'échantillonnage qui doit être
inférieure à fe_origine (44100 dans le cas de cette ressource)
13
14 signal_entree = signal_entree_origine[::int(fe_origine/fe)]
15
16 Te = 1 / fe
17
18 Ne = len(signal_entree) # Ne est le nombre d'échantillon
19 duree_signal = (Ne - 1) * Te
20
21 temps = np.array([i * Te for i in range(Ne)])
22
23 plt.subplot(221)
24 plt.plot(temps, signal_entree, 'b')
25 plt.xlabel('Temps (s)')
26 plt.ylabel('Amplitude du signal en entrée')
27
28 FFT_signal_entree = np.fft.fft(signal_entree)
29 coef_fourier_entree = np.concatenate((FFT_signal_entree[0] / Ne,
FFT_signal_entree[1:] * 2 / Ne), axis = None) # Permet d'obtenir les
bonnes amplitudes sur le spectre
30 module_coef_fourier_entree = np.absolute(coef_fourier_entree)
31
32 freq = np.array([i/duree_signal for i in range(Ne)]) # L'écart
fréquentiel entre deux valeurs du spectre est 1/Durée_signal
33
34 plt.subplot(222)
35 plt.plot(freq, module_coef_fourier_entree, 'b')
36 plt.xlabel('Fréquence (Hz)')
37 plt.ylabel('Module des coefficients de Fourier du signal en entrée')
38
39
40 ## Filtrage numérique passe bas d'ordre 1
41
42 fc = 720 # Fréquence de coupure
43
44 signal_sortie = np.zeros_like(signal_entree)
45
46 for i in range(Ne - 1):
    signal_sortie[i + 1] = signal_sortie[i] * (1 - 2*np.pi*fc*Te) +
signal_entree[i] * (2*np.pi*fc*Te) # Schéma Euler explicite
    signal_sortie[i + 1] = signal_sortie[i] / (1 + 2*np.pi*fc*Te) +
signal_entree[i + 1] * ((2*np.pi*fc*Te) / (1 + 2*np.pi*fc/fe)) # Schéma
Euler implicite

```

```

51 ## Affichage du signal sortie du filtre et de son spectre
52
53 plt.subplot(223)
54 plt.plot(temp, signal_sortie, 'r')
55 plt.xlabel('Temps (s)')
56 plt.ylabel('Amplitude du signal en sortie')
57
58 FFT_signal_sortie = np.fft.fft(signal_sortie)
59 coef_fourier_sortie = np.concatenate((FFT_signal_sortie[0] / Ne,
60 FFT_signal_sortie[1:] * 2 / Ne), axis = None) # Permet d'obtenir les
61 bonnes amplitudes sur le spectre
62 module_coef_fourier_sortie = np.absolute(coef_fourier_sortie)
63
64 freq = np.array([i/duree_signal for i in range(Ne)])
65
66 plt.subplot(224)
67 plt.plot(freq, module_coef_fourier_sortie, 'r')
68 plt.xlabel('Fréquence (Hz)')
69 plt.ylabel('Module des coefficients de Fourier du signal en entrée')
70
71 ## Enregistrement du signal de sortie
72 wav.write("notes_piano_re_ech.wav",fe,signal_entree)
73 wav.write("notes_piano_re_ech_filtre.wav",fe,signal_sortie)
74
75
76 ## Affichage des courbes
77
78 plt.show()

```