



21 mai 2025

Alexandre D'HÉRISSART - Remi LELUAN

Groupe : B2-3/4

SAR Audio - Partie Signal - Matlab



IMT Atlantique

Bretagne-Pays de la Loire
École Mines-Télécom

Table des matières

1	Introduction au problème	3
	3
2	Partie 1 : Signal	4
	Synthèse additive	4
.1	Question 1.1	4
.2	Question 1.2	6
.3	Question 1.3	7
.4	Question 1.4	9
.5	Question 1.5	11
	Synthèse soustractive	13
.1	Question 2.1	13
.2	Question 2.2	14
.3	Question 2.3	15
.4	Question 2.4	16
	Effets audio-numériques	19
.1	Question 3.1	19
.2	Question 3.2	19
.3	Question 3.3	20
.4	Question 3.4	21
.5	Question 3.5	22
.6	Question 3.6	22
.7	Question 3.7	23
.8	Question 3.8	24
.9	Question 3.9	24
.10	Question 3.10	25
.11	Question 3.11	25
.12	Question 3.12	25
.13	Question 3.13	27
.14	Question 3.14	27
.15	Question 3.15	29
.16	Question 3.16	29
.17	Question 3.17	29
.18	Question 3.18	30
.19	Question 3.19	31
3	Conclusion	32

1 Introduction au problème

Dans le cadre de l'UE *Electrical Engineering*, ce projet de « SAR Audio » a pour objectif de nous initier aux fondamentaux du traitement numérique du signal appliqué à l'audio. Nous allons explorer différentes techniques de synthèse sonore ainsi que des effets audio-numériques classiques. L'objectif sera double : d'une part analyser des enregistrements instrumentaux pour en extraire leurs caractéristiques spectrales et quantifier leur qualité harmonique, d'autre part synthétiser des sons et leur appliquer des effets tels que réverbération et delay pour comprendre le fonctionnement pratique des filtres et des convolutions.

Nous commençons par la **synthèse additive**, qui consiste à analyser un son harmonique réel, en extraire les fréquences principales, puis à le reproduire par addition de sinusoïdes. Cette démarche permet de comprendre comment un son musical peut être reconstruit à partir de ses composantes spectrales.

Nous aborderons ensuite la **synthèse soustractive**, méthode complémentaire dans laquelle un signal riche en harmoniques est filtré pour en façonner le timbre. Cette partie mettra en jeu des filtres numériques, que nous analyserons.

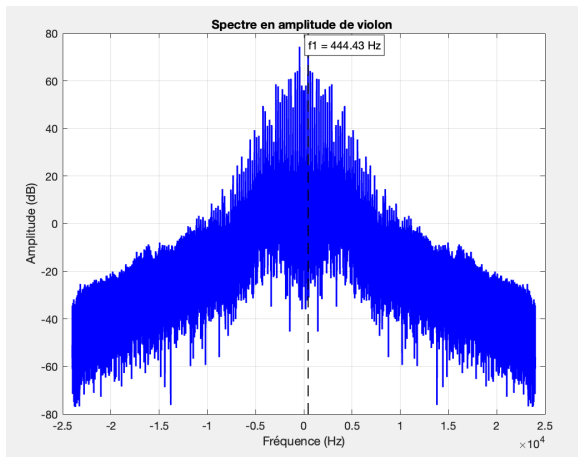
Enfin, nous étudierons plusieurs **effets audio-numériques**, en particulier les **effets de réverbération** et les **effets de retard**. Ces traitements, couramment utilisés en production musicale, seront implémentés à l'aide de convolutions directes ou rapides (FFT), ainsi que de filtres à rétroaction. Ils nous permettront d'observer comment un signal peut être spatialement enrichi ou transformé.

Ce projet constitue ainsi une introduction complète aux concepts essentiels du traitement audio numérique, alliant théorie et mise en pratique avec MATLAB.

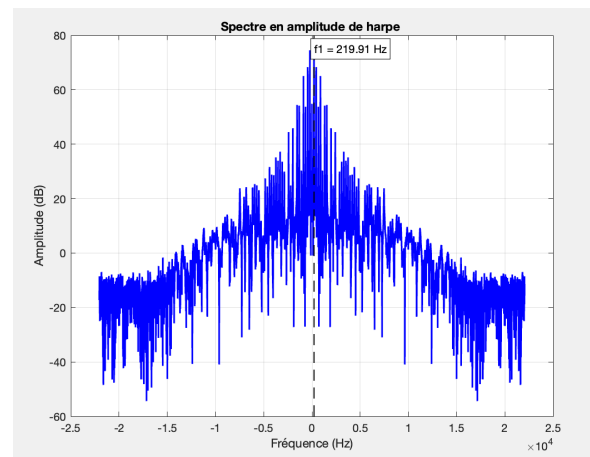
2 Partie 1 : Signal

Synthèse additive

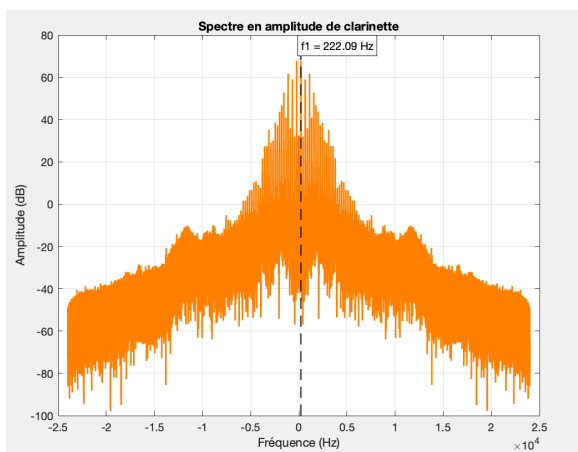
.1 Question 1.1



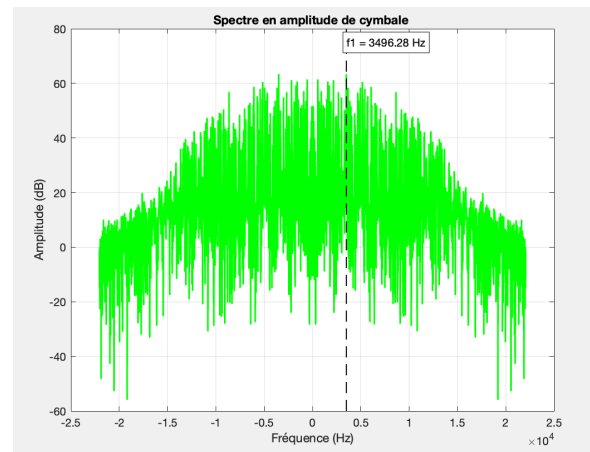
(a) Spectre violon



(b) Spectre harpe



(c) Spectre clarinette



(d) Spectre cymbale

FIGURE 1 – Spectres d'amplitude (en dB) de différents instruments à cordes et percussions.

On compare ci-dessous les spectres en amplitude (en dB) d'instruments à cordes et de deux autres instruments, ainsi que leurs fréquences fondamentales.

Instrument	Couleur du tracé	Fréquence fondamentale f_1 (Hz)
Harpe	bleu	219.91
Violon	bleu	444.43
Clarinette	orange	222.09
Cymbale	vert	3496.28

TABLE 1 – Fréquences fondamentales des différents enregistrements

► **Instruments à cordes (harpe, violon) :**

Les spectres montrent des raies très nettes et régulièrement espacées à des fréquences multiples de f_1 . Ce profil « en peigne » est caractéristique d'une source à corde vibrante.

► **Clarinette :**

On observe également des raies harmoniques, mais uniquement aux multiples impairs de f_1 .

► **Cymbale :**

Le spectre est large et continu, sans raies harmoniques régulières. Cette "densité" spectrale est typique d'une percussion métallique inharmonique, où la répartition fréquentielle est très étendue.

La comparaison montre que :

- Les *instruments à cordes* produisent un spectre *harmonique* (pics à $n.f_1$).
- La *clarinette* produit un spectre *impair-harmonique*.
- La *cymbale* produit un spectre *inharmonique*.

Ces différences illustrent comment la nature physique de la source sonore se reflète directement dans la forme de son spectre fréquentiel.



.2 Question 1.2

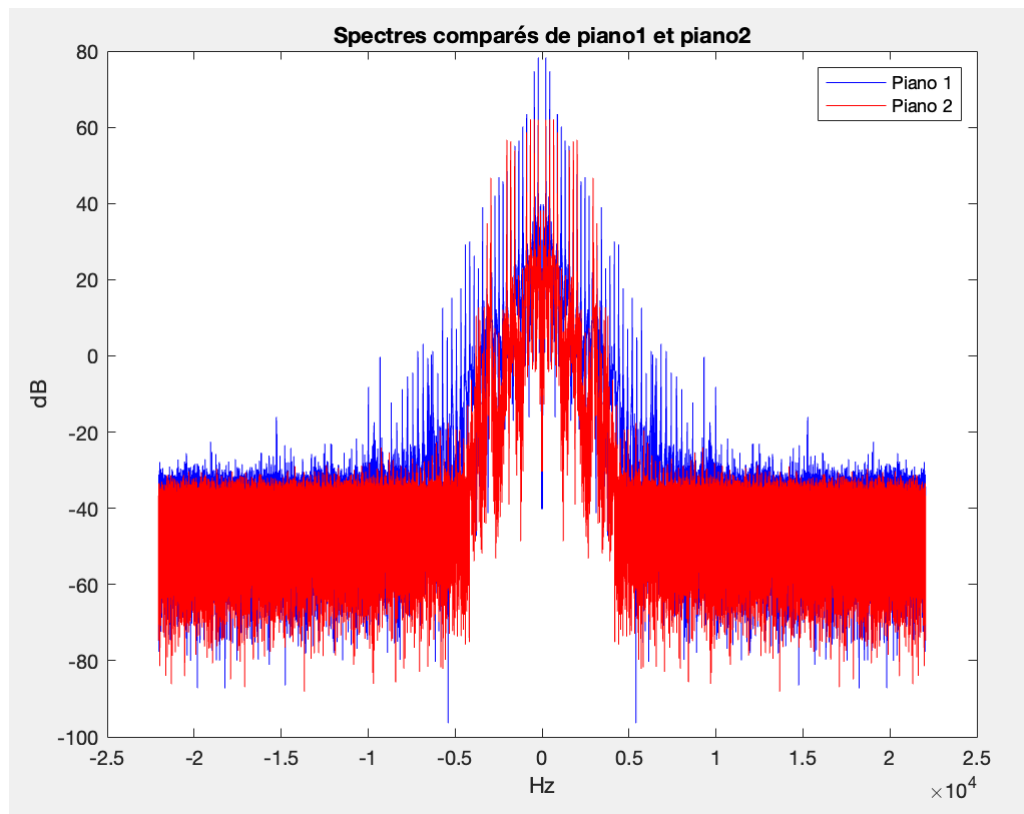


FIGURE 2 – Spectres comparés des piano1 et piano2

À partir du graphique, on peut comparer l'harmonicité des deux pianos :

Piano 1 (courbe bleue)

- Présente des pics spectraux plus marqués et réguliers dans la zone centrale (autour de 0 Hz jusqu'à environ $\pm 10^4$ Hz).
- Ces pics indiquent une forte présence d'harmoniques, révélateur d'une meilleure sonorité.
- La dynamique spectrale est plus large, ce qui signifie que les différences entre les pics et le bruit de fond sont plus importantes.

Piano 2 (courbe rouge)

- Le spectre est globalement plus aplati, avec des amplitudes plus faibles.
- Les pics sont moins prononcés, ce qui peut suggérer un contenu harmonique moins riche.
- Le bruit de fond est plus élevé, ce qui peut dégrader la pureté du son.

Le **piano 1** semble **plus harmonieux** que le piano 2.

Quelles sont les f_1 ?

On devrait trouver

TABLE 2 – Inharmonicité des harmoniques mesurées pour les deux pianos

Piano 1			
Harmonique	Théorique (Hz)	Mesurée (Hz)	Inharmonicité (cents)
f2	440.75	442.04	+5.07
f3	661.12	663.22	+5.50
f4	881.50	885.54	+7.92
f5	1101.87	1108.34	+10.13
f6	1322.24	1331.46	+12.02
f7	1542.62	1556.20	+15.18
Piano 2			
Harmonique	Théorique (Hz)	Mesurée (Hz)	Inharmonicité (cents)
f2	885.57	892.49	+13.49
f3	1328.35	1346.96	+24.09
f4	1771.13	1797.10	+25.20
f5	2213.92	2241.62	+21.53
f6	2656.70	2686.56	+19.35
f7	3099.48	3103.81	+2.42

$f_1 \approx 220 \text{ Hz}$
pour les
2 pianos
ce qui ne
semble pas
être le
cas ici.

TABLE 3 – Inharmonicité moyenne et maximale des pianos

Piano	Inharmonicité moyenne	Inharmonicité max
Piano 1	9.3 cents	+15.18 cents
Piano 2	17.7 cents	+25.20 cents

Le piano 1 semble donc être plus harmonieux que le piano d'après les résultats obtenus.

.3 Question 1.3

Méthodologie

Pour réaliser la synthèse additive, nous avons :

1. Calculé sa transformée de Fourier (FFT) du piano jugé le plus harmonique sur une taille N_{fft} puissance de deux. $\rightarrow ??$ (à quel ?)
2. Détecté la fréquence fondamentale f_1 comme le premier pic du spectre pour $f \geq 0$.
3. Pour chaque harmonique $n = 1 \dots 8$, extrait l'amplitude spectrale maximale autour de $n f_1$ ($\pm 5\%$) et converti cette valeur en amplitude sinusoïdale.
4. Reconstitue le signal synthétique par superposition de sinusoïdes $A_n \sin(2\pi n f_1 t)$.

Code MATLAB

```
1 filename = '/Users/alexandredherissart/Desktop/SAR Audio/piano1.wav';
2 nmax      = 8;
3 win_rel   = 0.05;
4
5 [x, fs] = audioread(filename);
6 x        = x(:,1);
7 N         = length(x);
8 Nfft      = 2^nextpow2(N);
9 X         = fft(x, Nfft);
10 Xs        = fftshift(X);
11 f         = linspace(-fs/2, fs/2, Nfft);
12
13 % Detection de la fondamentale
14 pos       = (f >= 0);
15 [~, ix0] = max(abs(Xs(pos)));
16 fp       = f(pos);
17 f1       = fp(ix0);
18
19 A = zeros(nmax,1);
20 for n = 1:nmax
21     f0 = n * f1;
22     mask = (f>=f0*(1-win_rel)) & (f<=f0*(1+win_rel));
23     [~, loc] = max(abs(Xs(mask)));
24     idx      = find(mask);
25     A(n)     = 2*abs(Xs(idx(loc))) / Nfft;
26 end
27
28 % Synthese additive
29 t = (0:N-1)/fs;
30 x_synth = zeros(size(t));
31 for n = 1:nmax
32     x_synth = x_synth + A(n)*sin(2*pi*n*f1*t);
33 end
34
35 % Ecoute et sauvegarde
36 sound(x_synth, fs);
37 audiowrite('synth_piano1.wav', x_synth, fs);
```

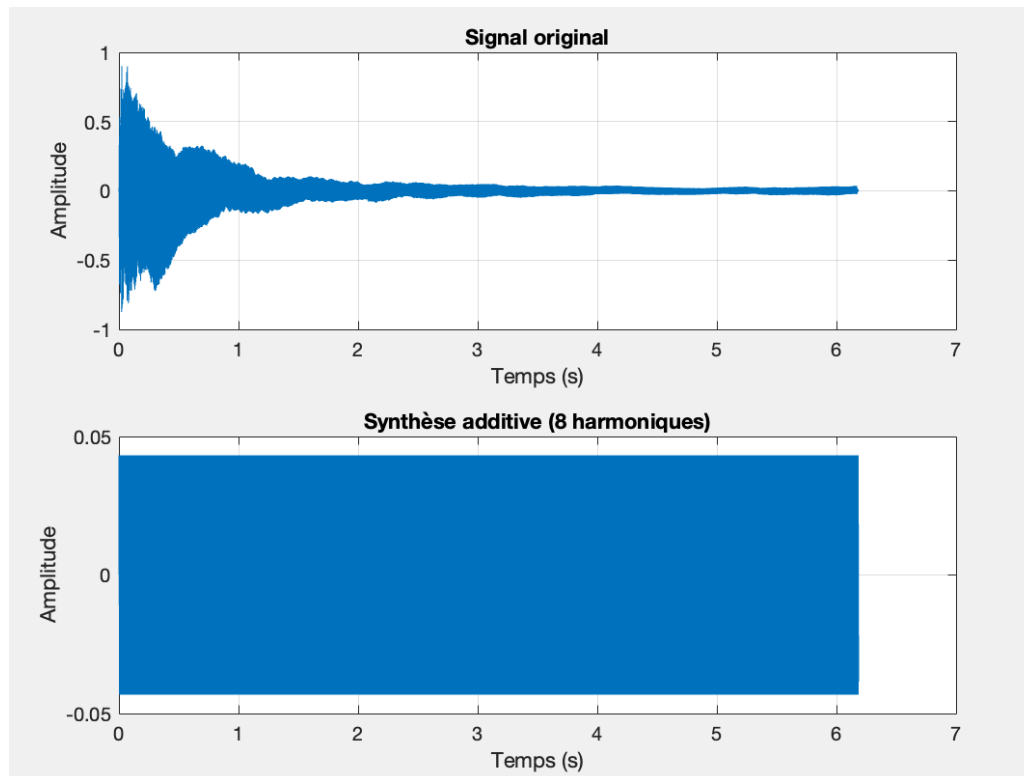



FIGURE 3 – Tracés des signaux original et transformé en amplitude

Le signal obtenu par synthèse additive reproduit seulement les composantes fréquentielles principales, sans tenir compte de l'évolution temporelle du signal réel (attaque, déclin, sustain, release).

Cela montre bien que la richesse harmonique seule ne suffit pas à reproduire la complexité d'un son d'instrument réel.



.4 Question 1.4

```

1 [y, fs] = audioread('synth_piano1.wav');
2 N = length(y);
3
4 A = round(0.1 * N); % Attack = 10%
5 D = round(0.1 * N); % Decay = 10%
6 R = round(0.1 * N); % Release = 10%
7 S = N - (A + D + R); % Sustain
8
9 env = [ ...
10     linspace(0, 1, A), ... % Attack
11     linspace(1, 0.7, D), ... % Decay
12     0.7 * ones(1, S), ... % Sustain
13     linspace(0.7, 0, R) ... % Release
14 ];
15

```

(il aurait été de
les rendre facilement
modifiables, mais c'est
un détail.

```
16 y_env = y(1:length(env)) .* env';  
17  
18 soundsc(y_env, fs);  
19 audiowrite('synth_ADSR_piano1.wav', y_env, fs);
```

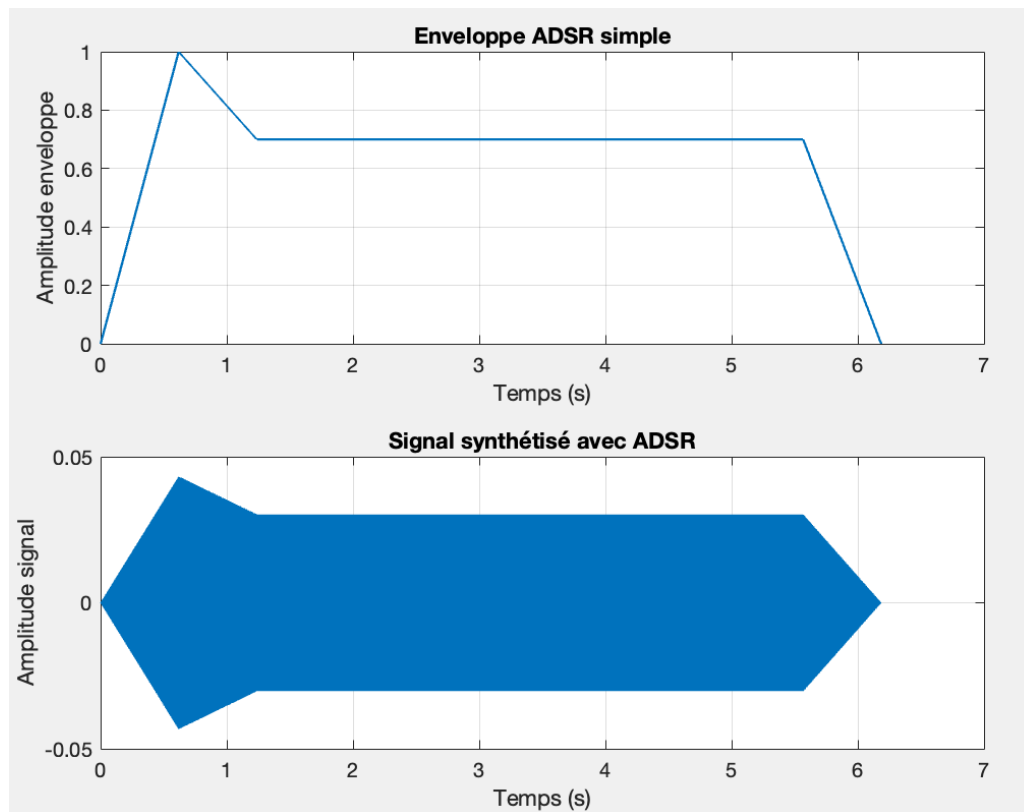


FIGURE 4 – Enveloppe ADSR du signal synthétisé

On observe bien que l'ajout de l'enveloppe ADSR permet de simuler l'évolution temporelle d'un vrai son (comme celui du piano) et améliorer considérablement le réalisme du signal synthétique,

.5 Question 1.5

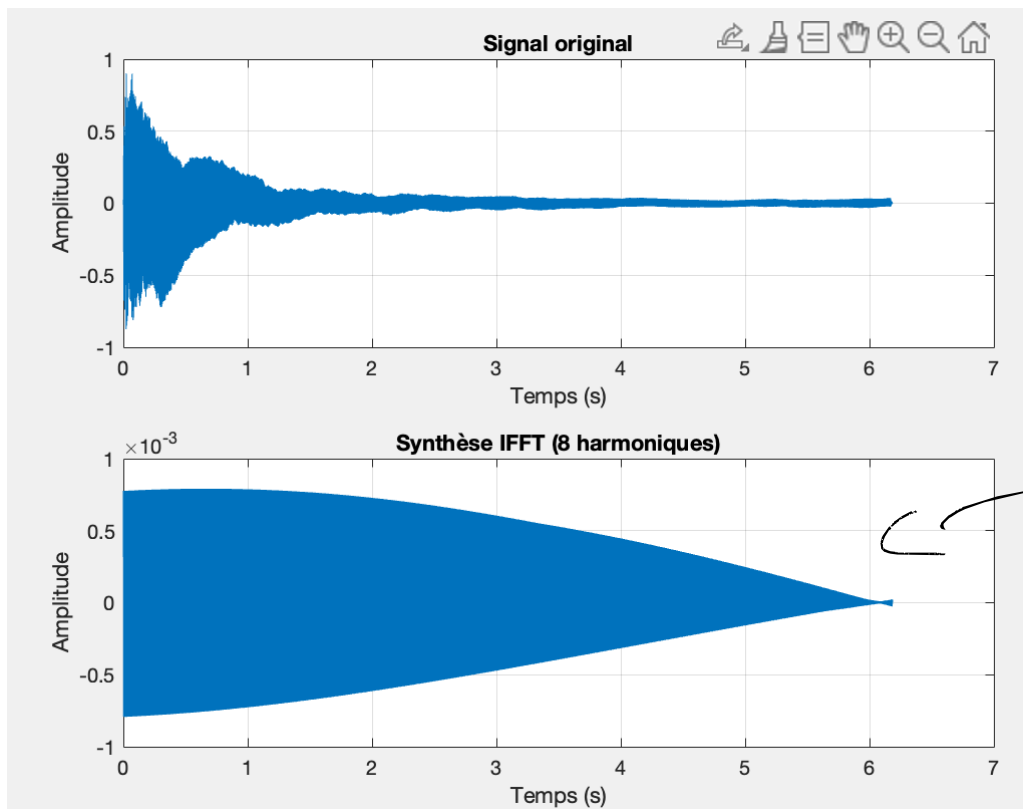


FIGURE 5 – Spectres comparés des piano1 et piano2

► Principe

- On reprend les mêmes huit composantes fréquentielles (mêmes amplitudes et phases) extraites du signal original.
- On construit un spectre « tronqué » ne comportant que les harmoniques et on applique l'IFFT pour repasser en domaine temporel.

► Comparaison avec la synthèse additive simple

1. *Spectre en amplitude* Les deux méthodes ne conservent que les mêmes raies spectrales : le spectre en dB est donc identique.
2. *Phases et forme d'onde*
 - La synthèse par IFFT préserve les phases originales de chaque harmonique, d'où une forme d'onde fidèle à l'enregistrement.
 - La synthèse additive (somme de sinusoïdes toutes en phase) génère un signal au même spectre mais dont l'attaque et suites diffèrent légèrement.
3. *Signal perçu* Cependant, en réinjectant les phases mesurées dans la somme sinusoïdale,

$$x(t) = \sum_{n=1}^8 A_n \sin(2\pi n f_1 t + \varphi_n),$$

on retrouve exactement le même signal qu'avec l'IFFT. Si l'on se contente de phases nulles, le signal reste très proche mais l'attaque peut sembler un peu plus sèche.

- **Perception auditive** Les fichiers générés par les deux méthodes sont tout de même très similaires à l'écoute, ce qui confirme que sur le plan perceptif, la conservation des amplitudes seules suffit généralement à restituer le timbre.



Synthèse soustractive

.1 Question 2.1

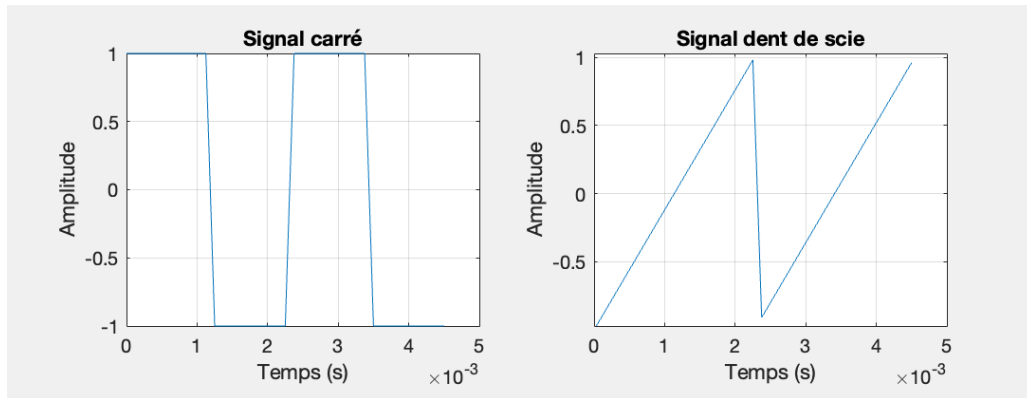


FIGURE 6 – Signaux dans le domaine temporel

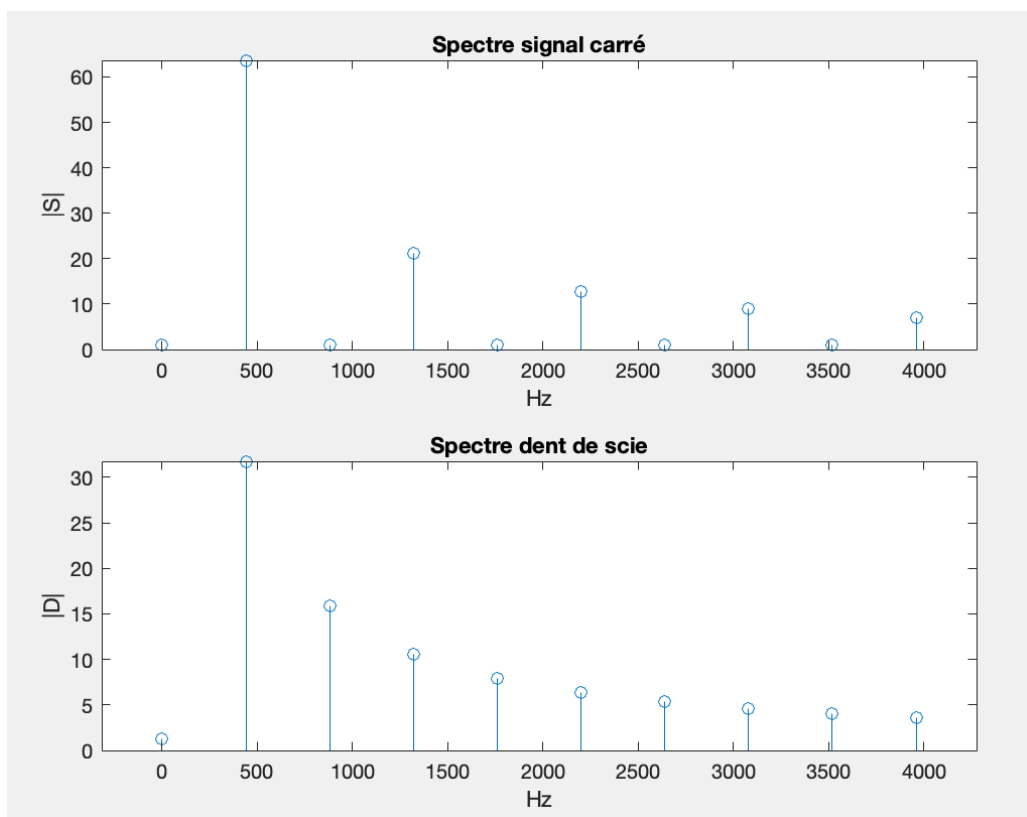


FIGURE 7 – Spectres du signal carré et en dent de scie

Méthode de vérification numérique

1. On génère plusieurs périodes d'un signal carré et d'un signal dent de scie, tous deux centrés et de période T .
2. On applique la Transformée de Fourier rapide (FFT) sur les signaux pour obtenir leur spectre. On utilise

So? dans kT on note f1 fondamentale

un nombre de points élevé (par exemple 1024) pour avoir une bonne précision en fréquence.

3. Pour chaque harmonique n , on repère la valeur du pic dans le spectre au niveau de la fréquence $n f_0$.
On note les amplitudes obtenues.

4. Les formules théoriques des amplitudes sont :

$$A_{\text{carré}}(n) = \frac{4}{\pi(2n-1)} \quad (\text{pour les harmoniques impaires})$$

$$A_{\text{scie}}(n) = \frac{2}{\pi n} (-1)^{n+1} \quad (\text{pour tous les entiers } n)$$

Obtenues comment?

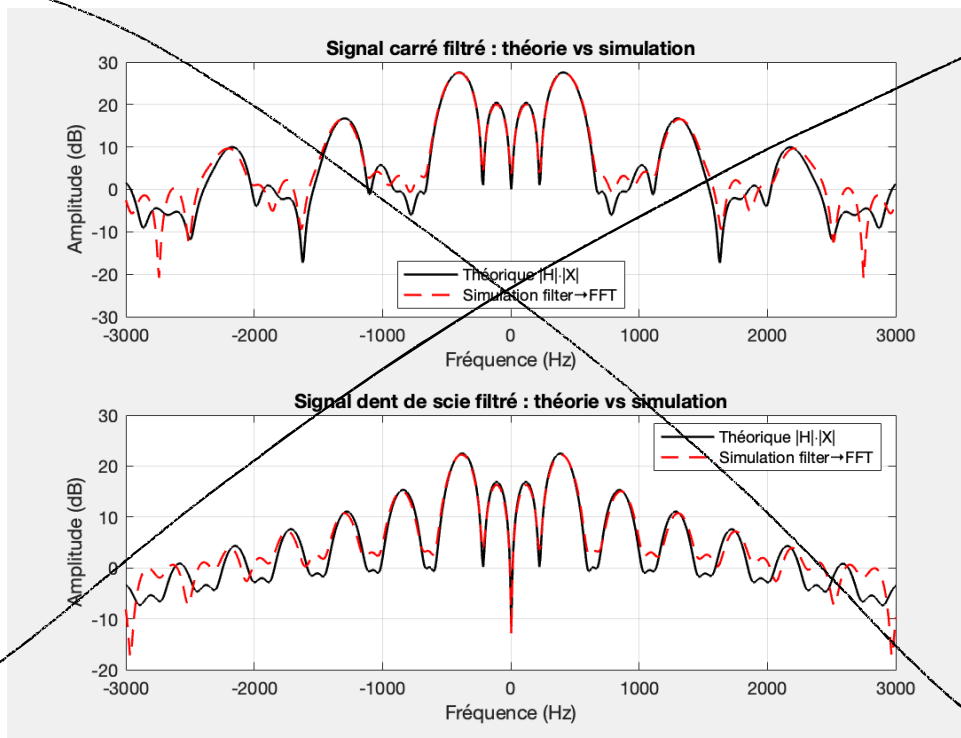
On compare les résultats numériques avec ces formules.

Limites de cette méthode

- Comme on travaille avec un signal de durée finie, on peut avoir des erreurs (pics élargis ou parasites) dans le spectre.
- Si la fréquence d'une harmonique ne correspond pas exactement à un des points de la FFT, l'amplitude mesurée peut être faussée.
- Si la fréquence d'échantillonnage n'est pas assez grande, certaines harmoniques peuvent être repliées dans le spectre (aliasing).

.2 Question 2.2

Le tracé ci-dessous compare le spectre théorique de sortie $|Y_{\text{th}}(f)| = |H(f)| |X(f)|$ et la simulation obtenue par application de `filter` suivie d'une FFT.



*Non.
Les courbes
ne sont pas
cohérentes
avec celles
obtenues
en Q2.1*

FIGURE 8 – Spectres du signal carré et de la dent de scie filtrés : théorie (noir) vs simulation (rouge).

2. Partie 1 : Signal

- Les maxima spectraux (pics harmoniques du carré à $\pm 440, \pm 1320, \dots$, et harmoniques entiers de la scie) se confondent parfaitement entre théorie et simulation, validant la relation $|Y(f)| = |H(f)| |X(f)|$.
- Dans les creux profonds (où le gain théorique tend vers $-\infty$ dB), la courbe rouge se maintient quelques dB au-dessus. On peut expliquer cela par la résolution finie de la FFT ($\Delta f = f_s/N_{\text{FFT}}$) et par la fuite spectrale.
- Globalement, la superposition des deux courbes montre que la simulation numérique par filter+FFT restitue fidèlement la réponse en fréquence d'un filtre LTI à réponse impulsionnelle finie, dès lors que l'on choisit un N_{FFT} suffisamment grand.

.3 Question 2.3

On compare quatre synthèses d'un même accord de piano ($f_0 = 440$ Hz, durée 1 s, enveloppe ADSR identique) selon deux méthodes :

On réalise d'abord une synthèse soustractive en utilisant deux sources différentes (signal carré et signal dent de scie), mais en appliquant toujours la même enveloppe temporelle et le même filtre passe-bas d'ordre 1. On compare ensuite ces deux résultats à une synthèse additive « pure », où le même filtre n'intervient pas et où le son est obtenu par sommation des 10 premières harmoniques pondérées.

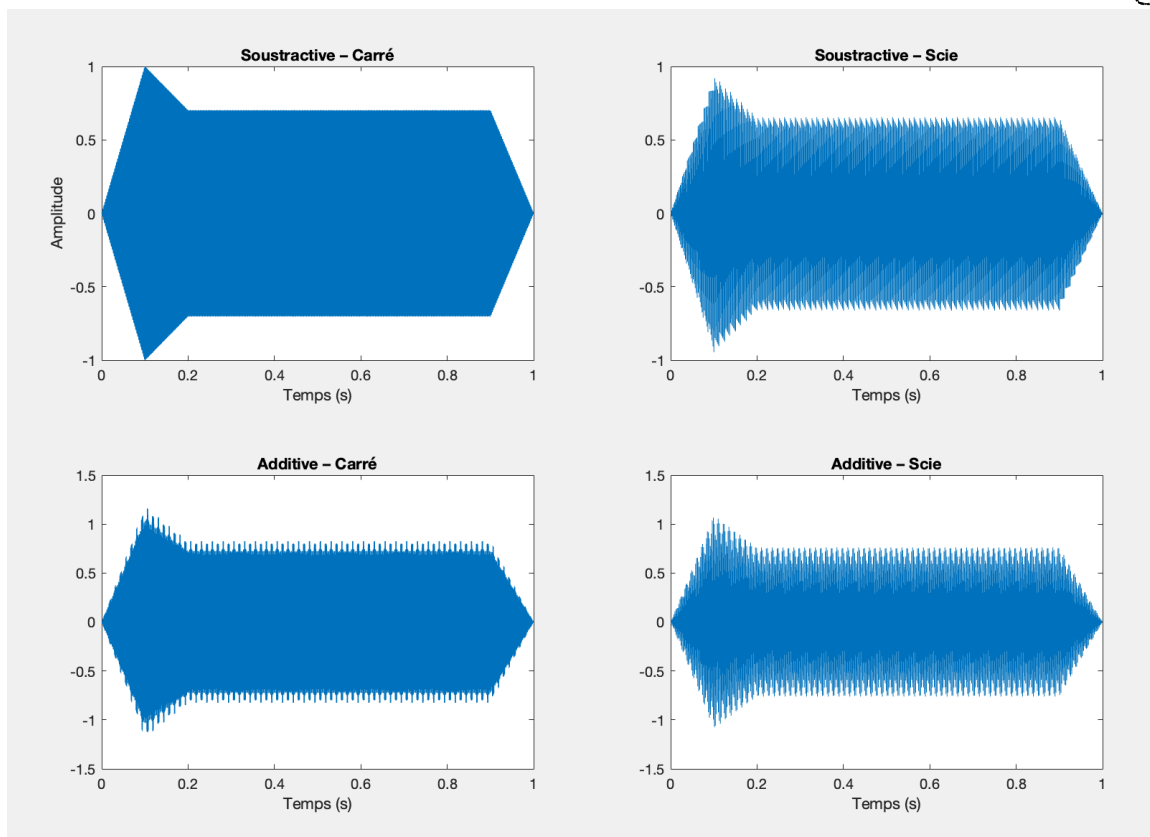


FIGURE 9 – Synthèse soustractive et additive pour source carré et dent de scie.

Résultats d'écoute

- **Soustractive avec carré** : Le timbre est assez dense en fréquences aiguës ; l'enveloppe ADSR façonne la dynamique, mais les crêtes harmoniques hautes restent présentes derrière le filtre passe-bas simple.
- **Soustractive avec dent de scie** : La dent de scie contient toutes les harmoniques, y compris les paires ; après filtrage d'ordre 1, le son reste très brillant, avec une saturation perceptible dans les aigus. La médiane du spectre paraît plus riche que pour le carré.
- **Additive (10 harmoniques)** : En sommant 10 sinusoïdes aux amplitudes $A_n \propto 1/n$ (carré) ou $A_n \propto 1$ (scie), on obtient un son bien plus pur, lisse et contrôlé : sans fuite spectrale. Et l'enveloppe ADSR module parfaitement l'évolution temporelle. Le timbre paraît moins agressif qu'en soustractive.
- **Comparaison générale** :
 - La synthèse soustractive dépend fortement du spectre initial du signal périodique.
 - La synthèse additive offre un contrôle plus fin des amplitudes harmoniques, au prix d'un calcul un peu plus coûteux.
 - L'enveloppe identique met en évidence que c'est bien le choix de la source périodique qui façonne le timbre du son.

.4 Question 2.4

Pour les différentes améliorations possibles, nous nous sommes inspirés de filtres connus pour leur efficacité. On compare donc les quatre filtres passe-bas suivants appliqués à un signal carré modulé par une enveloppe ADSR (Attack 10 %, Decay 10 %, Sustain 70 %, Release 10 %, $f_0 = 440$ Hz, $f_s = 8$ kHz), en utilisant l'outil **Filter Analyzer** :

Code MATLAB

```

1 fs = 8000;
2 f0 = 440;
3 T = 1;
4 t = 0:1/fs:T-1/fs;
5
6 % ADSR
7 A = round(0.1*fs); D = A; R = A;
8 S = length(t)-(A+D+R);
9 env = [ linspace(0,1,A), ...
10         linspace(1,0.7,D), ...
11         0.7*ones(1,S), ...
12         linspace(0.7,0,R) ];
13
14 x = square(2*pi*f0*t) .* env;
15
16 % FIR passe bas ordre 4
17 d_fir4 = designfilt('lowpassfir', ...
18                     'FilterOrder',4, ...

```



```

19         'CutoffFrequency',2000, ...
20         'SampleRate',fs);
21
22 % FIR passe bas ordre 16
23 d_fir16 = designfilt('lowpassfir', ...
24         'FilterOrder',16, ...
25         'CutoffFrequency',1500, ...
26         'SampleRate',fs);
27
28 % IIR Butterworth ordre 2
29 [bb2,aa2] = butter(2, 0.25);
30
31 % IIR Chebyshev II ordre 4
32 [bc4,ac4] = cheby2(4, 40, 0.20);
33
34 filterAnalyzer( d_fir4, d_fir16, bb2, aa2, bc4, ac4 );

```

➤ Réponse impulsionnelle

– *FIR O4* : linéaire en phase, décroissance rapide. – *FIR O16* : transition plus raide, phase parfaitement droite. – *Butter2* et *ChebII4* : queue infinie, la Chebyshev II montre de légères oscillations dues aux zéros en bande d'arrêt.

➤ Phase

– FIR : droites linéaires, pas de distorsion temporelle. – Butterworth : phase incurvée monotonique. – Chebyshev II : sauts de phase aux fréquences des zéros.

➤ Amplitude

– *FIR O4* : pente douce, passe-bas très progressive. – *FIR O16* : coupure raide à 1,5 kHz sans ondulations. – *Butter2* : transition intermédiaire autour de 2 kHz. – *ChebII4* : transition franche, ondulations en stop-band (-40 dB).

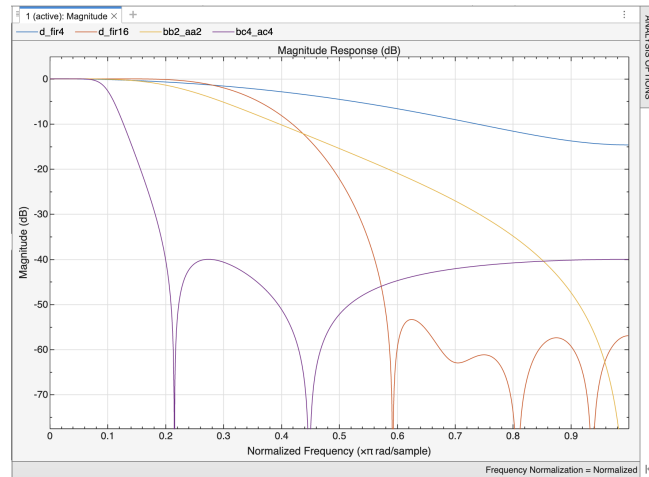
➤ Choix recommandé

Pour un timbre de « piano » avec un rendu naturel, le *FIR d'ordre 16* ($F_c = 1,5$ kHz) est optimal : pente raide, phase linéaire.

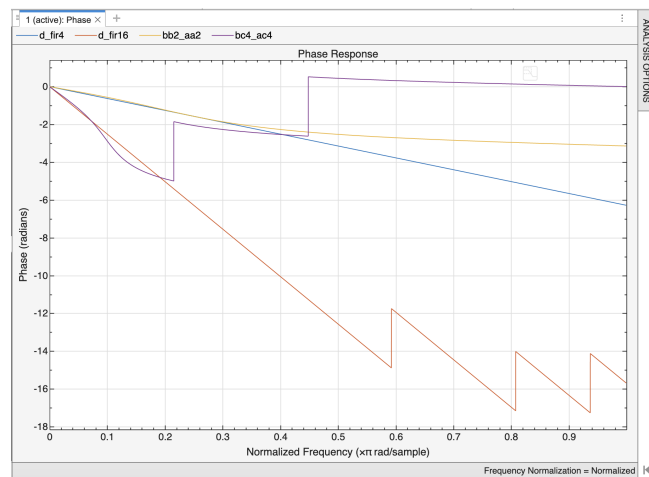
Illustrations dans Filter Analyzer

comme ça ...

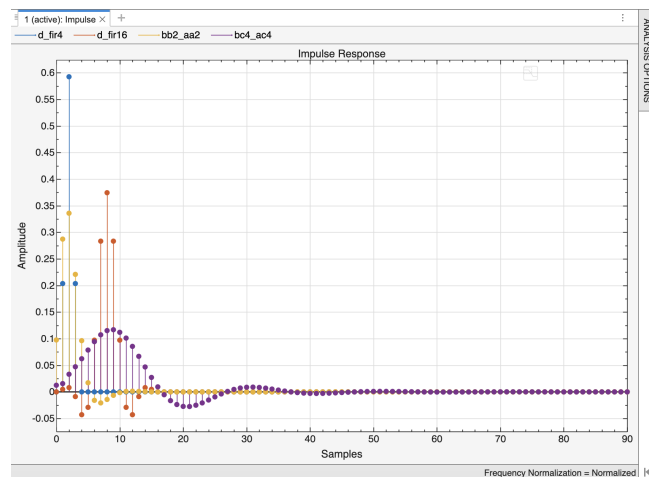
Les défauts
sur
le
paramétrage
des
filtres.
Ils sont
un peu
balancés



(a) Réponse en amplitude : pente douce pour FIR O4 (bleu), coupure raide à 1,5 kHz pour FIR O16 (rouge), transition large pour Butterworth O2 (jaune), ondulations en bande d'arrêt pour Chebyshev II O4 (violet)



(b) Réponse de phase $\arg H(e^{j\omega})$



(c) Réponse impulsionnelle des filtres comparés

Effets audio-numériques

.1 Question 3.1

On utilise la définition discrète de l'intercorrélation :

$$R_{yx}[m] = \sum_{n=-\infty}^{\infty} y[n+m] x[n].$$

Or, comme $y[n]$ est la convolution de $x[n]$ par $h[n]$,

$$y[n+m] = \sum_{k=-\infty}^{\infty} h[k] x[n+m-k].$$

D'où

$$\begin{aligned} R_{yx}[m] &= \sum_{n=-\infty}^{\infty} \left(\sum_{k=-\infty}^{\infty} h[k] x[n+m-k] \right) x[n] \\ &= \sum_{k=-\infty}^{\infty} h[k] \left(\sum_{n=-\infty}^{\infty} x[n+(m-k)] x[n] \right) \\ &= \sum_{k=-\infty}^{\infty} h[k] R_{xx}[m-k]. \end{aligned}$$

Ainsi,

$$R_{yx}[m] = \sum_{k=-\infty}^{\infty} h[k] R_{xx}[m-k].$$

.2 Question 3.2

En supposant que l'autocorrélation de l'entrée se comporte comme une impulsion,

$$R_{xx}[m] \approx \delta[m],$$

on a, d'après la relation de la question 3.1,

$$R_{yx}[m] = \sum_k h[k] R_{xx}[m-k] \approx \sum_k h[k] \delta[m-k] = h[m].$$

Méthode d'estimation de la réponse impulsionnelle :

1. Choisir un signal d'excitation $x[n]$ dont $R_{xx}[m] \approx \delta[m]$, par exemple :
2. Faire passer $x[n]$ dans le système pour obtenir la sortie $y[n]$.
3. Calculer l'intercorrélation

$$\hat{h}[m] = R_{yx}[m] = \sum_n y[n+m] x[n].$$

4. On a donc $\hat{h}[m]$ qui fournit directement une estimation de la réponse impulsionnelle $h[m]$.

.3 Question 3.3

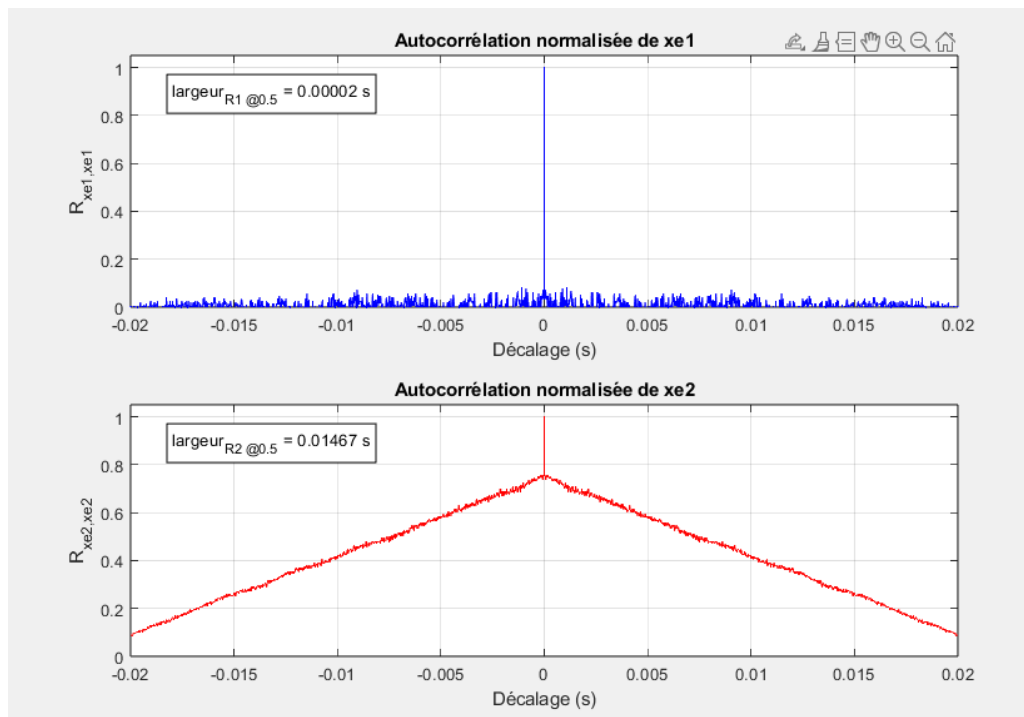


FIGURE 11 – Autocorrélations des signaux xe1 et xe2

Pour choisir celui dont l'autocorrélation est la plus proche d'une impulsion ($R_{xx}[m] \approx \delta[m]$), on :

- Calcule $R_{xe1,xe1}(m)$ et $R_{xe2,xe2}(m)$ avec la fonction `xcorr`.
- Mesure la « largeur effective » au niveau 0,5 (durée pendant laquelle $R_{xx}(m) \geq 0,5$).

Les résultats sont :

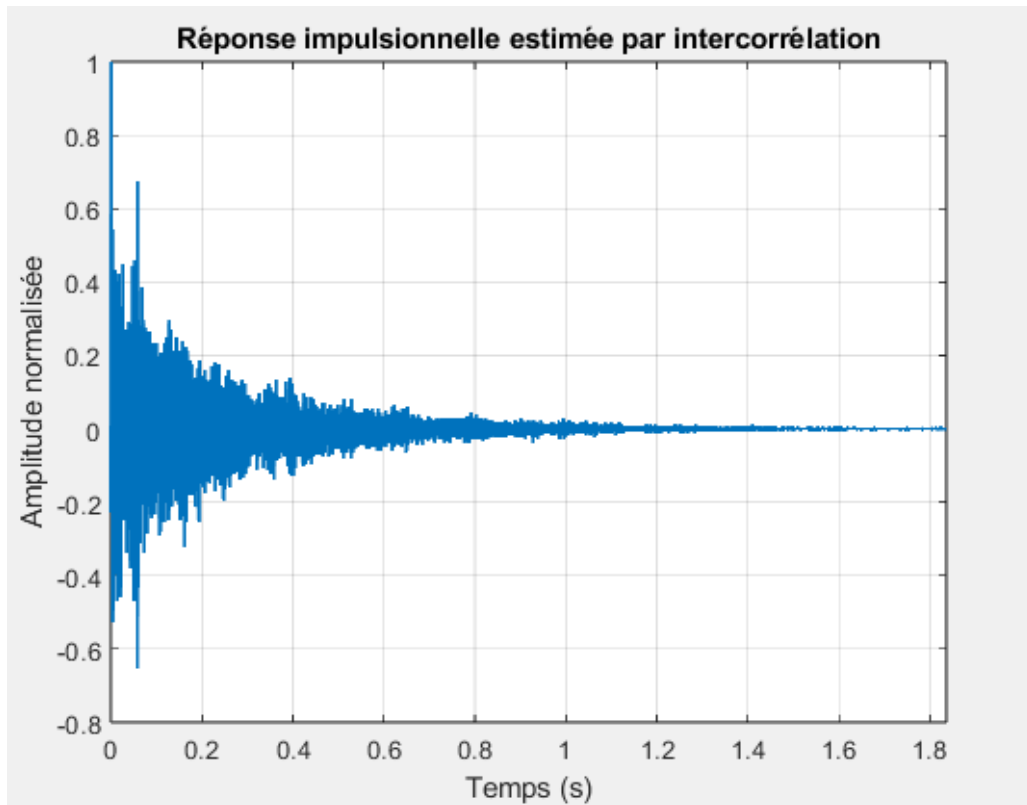
$$\text{largeur}_{R1@0.5} = 0,00002 \text{ s}, \quad \text{largeur}_{R2@0.5} = 0,01467 \text{ s}.$$

Comme *xe1* présente un pic d'autocorrélation beaucoup plus étroit, son R_{xx} est plus concentré en $m = 0$.

Finalement, *xe1* est le signal d'excitation le plus adapté pour estimer la réponse impulsionnelle, car son autocorrélation se rapproche le plus d'une impulsion unité. ✓

Ok, mais pouvait être justifié en regardant seulement l'allure des spectres. Le pallier semble un peu ad-hoc.

.4 Question 3.4

FIGURE 12 – Réponse impulsionnelle estimée du signal x_{e1} par corrélation croisée

On reprend la méthodologie suivante :

1. On fait passer le signal d'excitation choisi, $x[n]$, dans la fonction `simule_piece` pour obtenir la sortie $y[n]$ simulant la capture en salle.
2. On calcule l'intercorrrelation

$$R_{yx}[m] = \sum_{n=-\infty}^{\infty} y[n+m] x[n].$$

3. Pour $m \geq 0$, on identifie

$$\hat{h}[m] = R_{yx}[m],$$

qui fournit une estimation de la réponse impulsionnelle de la pièce.

4. On trace $\hat{h}[m]$ en fonction du temps $t = m/f_s$.

La courbe obtenue présente :

- Un pic principal à $t \approx 0$, correspondant au son direct.

- Une décroissance presque exponentielle typique des réverbérations de salle.

notions haut-niveau de traitement audio!

Cette forme de réponse impulsionnelle est conforme à ce que l'on attend pour l'acoustique d'une pièce : un mélange du direct et des réflexions suivi d'une atténuation progressive.

Un peu plus de détails d'introduction du contenu aurait été bienvenue. Mais c'est correct.

.5 Question 3.5

```
1 function y = effet_reverb(x, h)
2 % effet_reverb Applique une reverberation par convolution FIR
3 % y = effet_reverb(x, h) renvoie le signal x reverbere en le
4 % filtrant par la reponse impulsionnelle h (FIR) via filter.
5
6     x = x(:);
7     h = h(:);
8
9     % Filtrage FIR via filter
10    y = filter(h, 1, x);
11 end
```

.6 Question 3.6

On chronomètre l'appel à `effet_reverb` sur un signal de guitare nylon. Voici le code et les résultats de deux exécutions successives :

```
1 [x, fs] = audioread("C:\Users\DHERALE\Desktop\nylon-guitar.wav");
2 load('h_est.mat', 'h_est'); % h_est obtenu en Q3.4
3
4 % Mesure du temps d'execution
5 tic;
6 y_rev = effet_reverb(x, h_est);
7 exec_time = toc;
8
9 fprintf('Temps d'execution de effet_reverb : %.4f secondes\n', exec_time);
```

Exécution #1 :

Temps d'exécution de `effet_reverb` : 0.0777 secondes

Exécution #2 :

Temps d'exécution de `effet_reverb` : 0.0747 secondes

Ces mesures montrent que la convolution FIR complète du signal nécessite *environ* 0.075 s sur la plateforme de test. C'est un indicateur de performance utile pour comparer avec une implémentation FFT ou pour dimensionner l'effet en temps réel.



.7 Question 3.7

On compare les temps de calcul entre la convolution directe (FIR) et la convolution rapide (FFT).

Code de la fonction effet_reverb_FFT

```
1 function y = effet_reverb_FFT(x, h)
2 % effet_reverb_FFT Reverberation par convolution rapide (FFT)
3 % y = effet_reverb_FFT(x, h) applique la convolution lineaire
4 % de x et h dans le domaine frequentiel.
5
6 % Dimensions
7 Nx = length(x);
8 Nh = length(h);
9
10 % Taille FFT Nx + Nh - 1, arrondie la puissance de deux
11 NFFT = 2^nextpow2(Nx + Nh - 1);
12
13 % Spectres
14 X = fft(x, NFFT);
15 H = fft(h, NFFT);
16
17 % Produit spectral
18 Y = X .* H;
19
20 % IFFT
21 y_full = ifft(Y, NFFT);
22 y = real(y_full(1 : Nx + Nh - 1));
23 end
```

Mesure des temps d'exécution

```
1 [x, fs] = audioread("nylon-guitar.wav");
2 load('h_est.mat','h_est');
3
4 % Convolution directe (FIR)
5 tic;
6 y_fir = effet_reverb(x, h_est);
7 t_fir = toc;
8 fprintf('FIR : %.4f s\n', t_fir);
9
10 % Convolution rapide (FFT)
11 tic;
12 y_fft = effet_reverb_FFT(x, h_est);
13 t_fft = toc;
```

```
14 fprintf('FFT : %.4f s\n', t_fft);
```

Sur deux exécutions successives, on obtient :

FIR : 0.0751 s, 0.0786 s,

FFT : 0.0054 s, 0.0050 s.

- La convolution par FFT est environ *15 fois plus rapide* que la convolution directe.
- Pour des signaux longs, la FFT devient très avantageuse.



.8 Question 3.8

Mathématiquement, la convolution par FFT et la convolution directe (implémentée par `filter`) réalisent exactement la même opération de convolution linéaire à condition que l'on utilise une taille de FFT suffisante ($N_{\text{FFT}} \geq N_x + N_h - 1$). En effet, on a pour tout signal $x[n]$ et toute réponse impulsionnelle $h[n]$:

$$y[n] = (x * h)[n] = \text{IFFT}(\text{FFT}(x) \times \text{FFT}(h))[n].$$

Les deux méthodes diffèrent seulement sur le plan algorithmique :

- Complexité : $\mathcal{O}(N_x N_h)$ pour la convolution échantillon par échantillon, contre $\mathcal{O}(N_{\text{FFT}} \log N_{\text{FFT}})$ pour la FFT + produit spectral + IFFT, d'où un gain pour de grands signaux.
- Précision numérique : la FFT et l'IFFT introduisent de petites erreurs d'arrondi, mais dans la pratique ces écarts sont négligeables à l'écoute.



Conclusion : La méthode par FFT est équivalente à la convolution classique du point de vue du résultat, mais elle est souvent préférable pour des signaux de grande longueur.

.9 Question 3.9

On part de l'équation de delay (pour un filtre causal) :

$$y[k] + g y[k - \tau] = x[k].$$

En transformée en z , on a

$$Y(z) (1 + g z^{-\tau}) = X(z) \implies H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + g z^{-\tau}}.$$

On a vu la transformée de z en cours?

On reconnaît une série géométrique :

$$\frac{1}{1 + g z^{-\tau}} = \frac{1}{1 - (-g z^{-\tau})} = \sum_{n=0}^{\infty} (-g z^{-\tau})^n = \sum_{n=0}^{\infty} (-g)^n z^{-n\tau}.$$

En repassant en temporel, chaque terme $z^{-n\tau}$ correspond à un décalage de $n\tau$ échantillons ; on obtient donc la réponse impulsionnelle causale :

$$h[k] = \sum_{n=0}^{\infty} (-g)^n \delta[k - n\tau] \iff h[k] = \begin{cases} (-g)^n, & \text{si } k = n\tau, n \in \mathbb{N}, \\ 0, & \text{sinon.} \end{cases}$$

.10 Question 3.10

Le filtre de delay est défini par la relation

$$y[k] + g y[k - \tau] = x[k]$$

dont la fonction de transfert est

$$H(z) = \frac{Y(z)}{X(z)} = \frac{1}{1 + g z^{-\tau}}.$$

Les pôles de ce système sont les racines de $1 + g z^{-\tau} = 0$, c'est-à-dire

$$z^\tau = -g \implies z = (-g)^{1/\tau} \exp\left(j \frac{2\pi k}{\tau}\right), \quad k = 0, 1, \dots, \tau - 1.$$

La condition de stabilité d'un filtre linéaire invariant est que tous ses pôles soient strictement à l'intérieur du cercle unité :

$$|z| = |g|^{1/\tau} < 1 \iff |g| < 1.$$

.11 Question 3.11

Pour le filtre de delay défini par

$$y[k] + g y[k - \tau] = x[k],$$

la fonction de transfert s'écrit

$$H(z) = \frac{1}{1 + g z^{-\tau}} = \frac{B(z)}{A(z)},$$

avec

$$B(z) = 1, \quad A(z) = 1 + g z^{-\tau}.$$

En notation MATLAB (`filter(b,a,x)`) on prend donc :

$$b = [1], \quad a = [1, 0, 0, \dots, 0, g],$$

où le vecteur a comporte un 1 en position 0, des zéros pour les retards intermédiaires, et g en position τ .

.12 Question 3.12

Le script suivant génère une impulsion suffisamment longue, applique le filtre de delay par `filter` et trace la réponse impulsionnelle $h[k]$:

```

1 Fe      = 44100;
2
3 % Parametres du delay
4 tau_s   = 0.25;
5 g        = 0.9;
6
7 tau      = round(tau_s * Fe);
8
9 % Coefficients du filtre de Delay

```

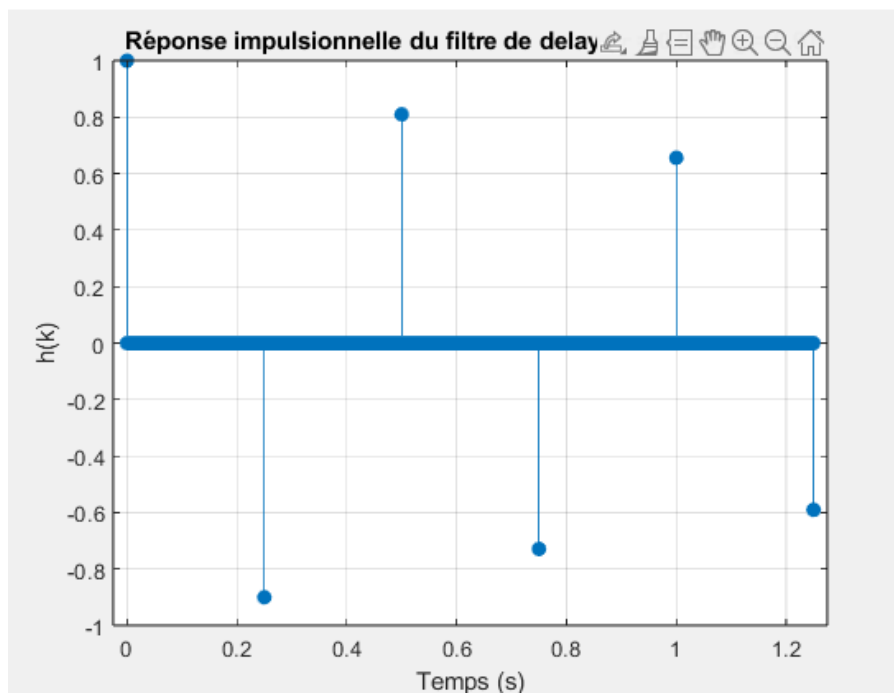
Non vu en
Cours.
Vu en
prépa?
En tout
cas il
y avait
plus
simple
(suite
généralisée)

pareil, un peu
bizarre, mais ok.

```

10 b = 1;
11 a = [1, zeros(1, tau-1), g];
12
13
14 N_imp = 5 * tau;
15 imp = [1; zeros(N_imp, 1)];
16
17 % Application du filtre pour obtenir h[k]
18 h = filter(b, a, imp);
19
20 k = (0:length(h)-1) / Fe;
21 figure;
22 stem(k, h, 'filled');
23 grid on;
24 xlabel('Temps (s)');
25 ylabel('h(k)');
26 title('Reponse impulsionnelle du filtre de delay (\tau = 0.25s, g = 0.9)');

```


FIGURE 13 – Réponse temporelle de $h[k]$

- À $k = 0$ ($t = 0$), on observe un pic de valeur $h[0] = 1$.
- Ensuite, des échos apparaissent tous les $\tau = 0,25$ s à des amplitudes alternant en signe et décroissant en module :

$$h[n\tau] = (-g)^n, \quad n = 1, 2, 3, \dots \implies -0,9, 0,81, -0,729, 0,6561, \dots$$

- Entre ces temps, $h[k] = 0$, ce qui traduit l'absence de réponse hors des instants d'écho.

.13 Question 3.13

On part de la fonction de transfert du filtre de delay,

$$H(z) = \frac{1}{1 + g z^{-\tau}}.$$

En évaluant en $z = e^{2\pi j\nu}$, on obtient :

*soit, Fourier
(aurait été
bien de le préciser)*

$$\hat{H}(\nu) = H(e^{2\pi j\nu}) = \frac{1}{1 + g e^{-2\pi j\nu\tau}}.$$

Module

$$|\hat{H}(\nu)| = \frac{1}{|1 + g e^{-2\pi j\nu\tau}|} = \frac{1}{\sqrt{(1 + g \cos(2\pi\nu\tau))^2 + (g \sin(2\pi\nu\tau))^2}} = \frac{1}{\sqrt{1 + 2g \cos(2\pi\nu\tau) + g^2}}.$$

Phase

$$\arg(\hat{H}(\nu)) = -\arg(1 + g e^{-2\pi j\nu\tau}) = -\arctan\left(\frac{-g \sin(2\pi\nu\tau)}{1 + g \cos(2\pi\nu\tau)}\right) = \arctan\left(\frac{g \sin(2\pi\nu\tau)}{1 + g \cos(2\pi\nu\tau)}\right).$$

.14 Question 3.14

On compare la réponse en fréquence analytique du filtre de delay IIR au résultat numérique obtenu par FFT de sa réponse impulsionnelle. Voici le code MATLAB utilisé :

```

1 load('h_delay.mat','h'); % h obtenu en Q3.12
2 Fe = 44100;
3
4 % FFT
5 NFFT = 2^nextpow2(length(h)*4);
6 f = (0:NFFT/2)*(Fe/NFFT);
7
8 % Reponse numerique par FFT de h
9 H_num = fft(h, NFFT);
10 H_num = abs(H_num(1:NFFT/2+1));
11
12 % Reponse theorique du delay
13 tau = round(0.25 * Fe);
14 g = 0.9;
15 w = 2*pi*f/Fe;
16 H_th = 1 ./ sqrt(1 + 2*g*cos(w*tau) + g^2);
17
18 figure;
19 plot(f, 20*log10(H_th+eps), 'k-', 'LineWidth',1.5); hold on;
20 plot(f, 20*log10(H_num+eps), 'r--','LineWidth',1.2);
21 grid on;
22 xlim([0 25]);

```

```

23 xlabel('Frequence (Hz)');
24 ylabel('Gain (dB)');
25 title('Reponse frequentielle du filtre de delay : theorie vs FFT');
26 legend('Theorique','Par FFT de h','Location','Best');

```

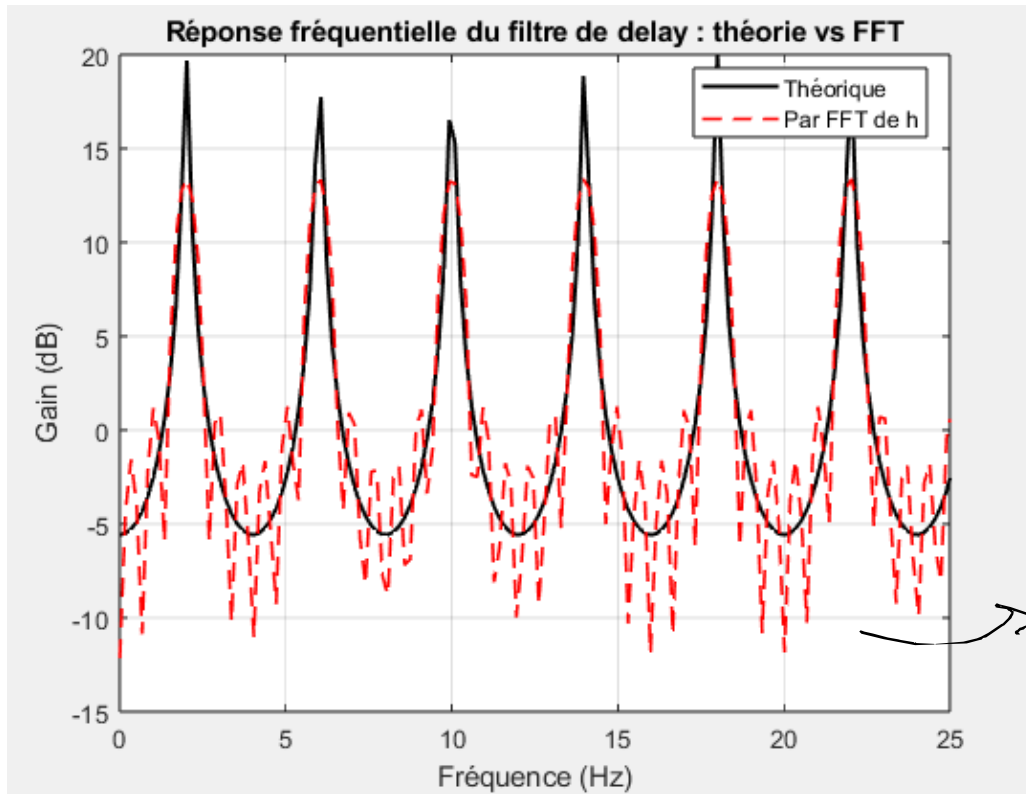


FIGURE 14 – Réponse fréquentielle zoomée sur 25Hz du filtre de delay : theorie vs FFT

Interprétation du résultat

- En zoomant sur la bande $[0 - 25]$ Hz, on distingue clairement les « dents » régulières du peigne : les crêtes apparaissent aux fréquences

$$f_k = \frac{2k+1}{2T_s} \quad (k = 0, 1, 2, \dots),$$

ici centrées environ à 2 Hz, 6 Hz, 10 Hz, 14 Hz, 18 Hz, 22 Hz.

- Les pics du peigne se superposent plus ou moins correctement
- En revanche, dans les creux du peigne le tracé rouge est plus irrégulier et semble caractérisé un phénomène bruyant :
 - La résolution fréquentielle finie ($\Delta f = F_e/N_{\text{FFT}}$) et le zéro-padding introduisent un léger lissage des minima, qui ne tombent pas exactement à $-\infty$ dB comme en théorie.
 - Les fluctuations résiduelles viennent aussi des arrondis numériques de l'IFFT et de la partie imaginaire non parfaitement nulle.
- Globalement, la bande passante et la position des dents sont identiques, mais la *précision* dans les minima est meilleure sur la courbe théorique que sur la FFT.

.15 Question 3.15

```
1 function y = effet_delay(x, tau, g, Fe)
2 %   y = effet_delay(x, tau, g, Fe) renvoie le signal x avec un delay de
3 %   tau echantillons et un amortissement g, a l'echantillonnage Fe.
4
5   % vecteurs de coefficients pour filter
6   b = 1;
7   a = [1, zeros(1, tau-1), g];
8
9   % application du filtre IIR
10  y = filter(b, a, x);
11 end
```

.16 Question 3.16

On teste l'effet de delay sur un accord de piano et on compare l'écoute *sec* vs *delay*.

```
1 [x, Fe] = audioread('piano_chord.wav');
2
3 % parametres du delay
4 tau_s = 0.25;
5 tau    = round(tau_s * Fe);
6 g      = 0.9;
7
8 % application de l effet
9 y_delay = effet_delay(x, tau, g, Fe);
10
11 soundsc(x, Fe); pause(length(x)/Fe + 0.5); % signal sec
12 soundsc(y_delay, Fe); % signal avec delay
```

- On perçoit d'abord l'accord de piano pur, puis l'accord répété tous les 0,25 s avec une amplitude $\approx 90\%$ de la précédente.
- Chaque répétition successive est légèrement plus faible et se mêle au signal suivant, créant un effet d'« écho ».

.17 Question 3.17

Implémentation de `effet_delay_filtre`

```
1 function y = effet_delay_filtre(x, tau, g, K, Fe)
2 % effet_delay_filtre Delay IIR avec filtre FIR dans la boucle de feedback
3
```

```
4      % Construction des coefficients du denominateur
5      % a(1) = 1, puis g/K aux retards tau ... tau+K-1
6      a = zeros(1, tau+K);
7      a(1) = 1;
8      a(tau + (1:K)) = g / K;
9
10     b = 1;
11
12     % Filtrage IIR
13     y = filter(b, a, x);
14 end
```

.18 Question 3.18

On teste l'effet *delay filtré* sur un accord de piano avec un filtre FIR de longueur $K = 10$ dans la boucle de retour :

```
1  [x, Fe] = audioread('piano_chord.wav');
2
3  % Parametres du delay
4  tau_s = 0.25;
5  tau    = round(tau_s * Fe);
6  g      = 0.9;
7  K      = 10;
8
9  % Application de l effet
10 y_del_filt = effet_delay_filtre(x, tau, g, K, Fe);
11
12 soundsc(x, Fe);
13 pause(length(x)/Fe + 0.5);
14 soundsc(y_del_filt, Fe);
15
16 audiowrite('piano_chord_delay_filtre.wav', y_del_filt, Fe);
```

- On entend d'abord l'accord de piano sec, puis, toutes les 0,25 s, une répétition atténuée et légèrement lissée.
- Le filtre FIR de longueur $K = 10$ atténue les hautes fréquences de chaque écho, produisant un son plus rond et réaliste et moins strident qu'avec un simple delay.

.19 Question 3.19

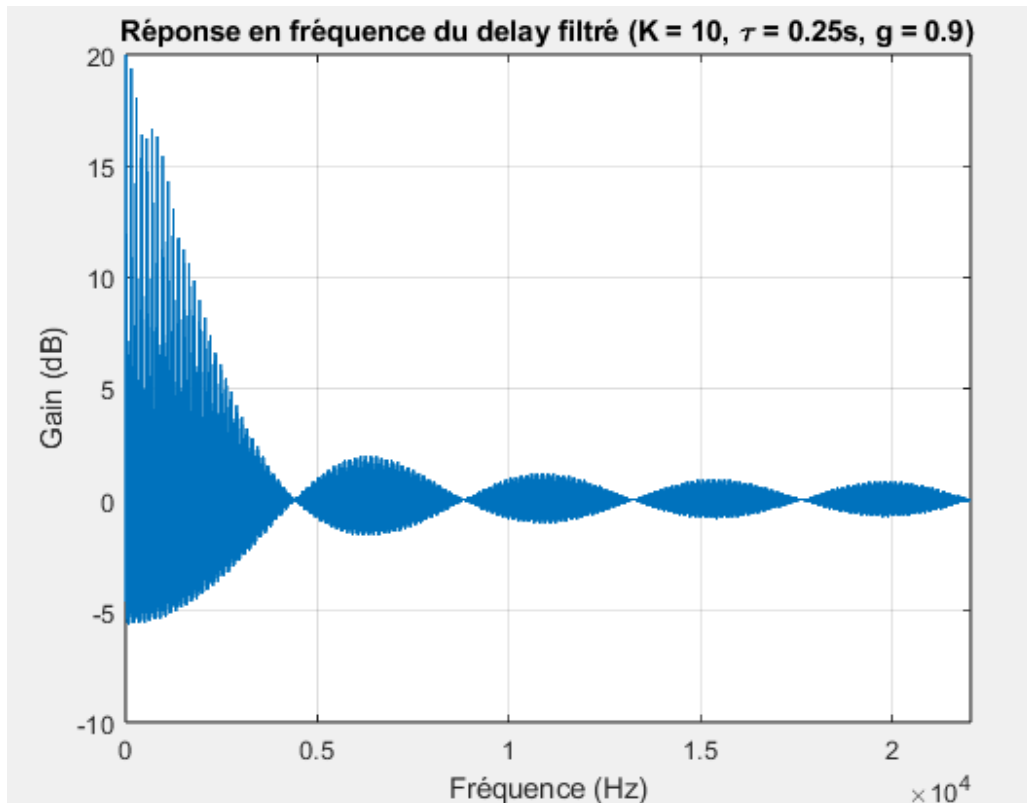


FIGURE 15 – Réponse en fréquence du filtre de delay filtré

- On reconnaît clairement la structure en peigne : des crêtes et creux réguliers espacés de $\Delta f = 1/\tau \approx 4$ Hz.
- Contrairement au delay pur, l'enveloppe des crêtes décroît avec la fréquence :

$$H_{\text{FIR}}(e^{j\omega}) = \frac{1}{K} \sum_{n=0}^{K-1} e^{-j\omega n} = \frac{\sin(\frac{\omega K}{2})}{K \sin(\frac{\omega}{2})} e^{-j\omega \frac{K-1}{2}},$$

qui est un passe-bas grossier. À chaque incrément de fréquence, le gain global diminue du fait de la moyenne sur les K échantillons dans la boucle.

- Cet amortissement progressif des hautes fréquences rend les échos successifs plus chauds et moins stridents, réalisant un effet de delay plus naturel qu'un simple IIR.

↳ c'est-à-dire ?

3 Conclusion

À travers ce projet SAR Audio, nous avons progressivement découvert les principales techniques de traitement numérique du signal appliqué au son. En partant de l'analyse fréquentielle d'enregistrements réels, nous avons appris à interpréter les spectres, extraire les harmoniques et évaluer l'inharmonicité, ce qui nous a permis de mieux comprendre la structure acoustique des sons musicaux.

Les étapes de synthèse additive et soustractive nous ont ensuite familiarisés avec la modélisation de sons, en combinant la théorie des séries de Fourier, la conception d'enveloppes ADSR et l'utilisation de filtres numériques. Ceci nous a offert une vision concrète des procédés utilisés dans la génération audio synthétique.

Enfin, l'étude des effets audio-numériques, notamment les réverbérations par convolution et les delays avec ou sans filtrage, nous a permis d'explorer la manière dont un signal peut être enrichi ou modifié, aussi bien dans le domaine temporel que fréquentiel. Ces expérimentations ont mis en évidence l'intérêt des filtres IIR et FIR de la transformée de Fourier rapide.

Au terme du projet, nous avons acquis une première boîte à outils pour analyser, transformer et synthétiser des sons de manière numérique. Nous en retirons une compréhension plus approfondie du lien entre théorie du signal et pratiques concrètes de traitement audio, qui servira de base pour de futures explorations plus avancées dans le domaine.

globalement, plus de détails et d'introduction
des outils aurait été apprécié...