

IMT Atlantique

Dépt. Electrical Engineering
Technopôle de Brest-Iroise - CS 83818
29238 Brest Cedex 3
Téléphone : +33 (0)2 29 00 13 04
Télécopie : +33 (0)2 29 00 10 12
URL : www.imt-atlantique.fr

**Compte rendu du projet**

Situation d'Apprentissage Riche – Audio Signal Processing

Alexandre Carneiro-Gillet
Juliette Faurie

Date d'édition : 21 mai 2025

**IMT Atlantique**

Bretagne-Pays de la Loire
École Mines-Télécom

Sommaire

- 1. Synthèse additive 3
 - 1.1. Analyse d’un son harmonique..... 3
 - 1.2. Synthèse 6
- 2. Synthèse soustractive 7
- 3. Effets audio numériques 10
 - 3.1. Effets de réverbération..... 10
 - 3.2. Convolution classique 13
 - 3.3. Convolution rapide 13
 - 3.4. Effet de retard 14

Introduction

Ce projet explore les principes fondamentaux de la synthèse et du traitement numérique des sons à l'aide du logiciel MATLAB. En partant de l'analyse de fichiers audio, nous étudierons différentes techniques de synthèse sonore telles que la synthèse additive, la synthèse soustractive, et des méthodes plus avancées comme la synthèse Karplus-Strong. Par ailleurs, une attention particulière sera portée aux effets audio-numériques, notamment la réverbération et le delay, qui enrichissent le traitement du signal sonore. Ce travail a pour objectif de mieux comprendre les mécanismes de création et de transformation des sons à travers des approches théoriques et pratiques.

Présentation théorique du problème

La synthèse et le traitement du son reposent sur la manipulation de signaux audio, modélisés numériquement. Différentes techniques permettent de générer ou transformer ces sons, comme la synthèse additive (superposition d'harmoniques), la synthèse soustractive (filtrage d'un son riche), ou encore des effets comme la réverbération et le délai. Ces méthodes s'appuient sur des outils fondamentaux du traitement du signal, tels que la transformée de Fourier, les filtres ou les systèmes à réponse impulsionnelle. Le projet vise à explorer ces techniques de manière pratique avec MATLAB.

1. Synthèse additive

1.1. Analyse d'un son harmonique

Question 1.1

Visualisons tout d'abord le spectre d'amplitude obtenu avec une transformée de Fourier discrète de quelques instruments à cordes.

Pour cela, nous créons un code Matlab analysant un des signaux d'instruments à cordes qui nous sont proposées. Le programme ouvre le fichier, récupère le signal et sa fréquence d'échantillonnage, puis calcule sa transformée de Fourier (fft) et la recentre à l'aide de la fonction fftshift. Il renvoie alors le spectre d'amplitude du signal. (Voir fichier q1_1.m)

Voici les différents spectres obtenus, pour plusieurs instruments à cordes comme la guitare, le violoncelle ou encore la harpe celtique :

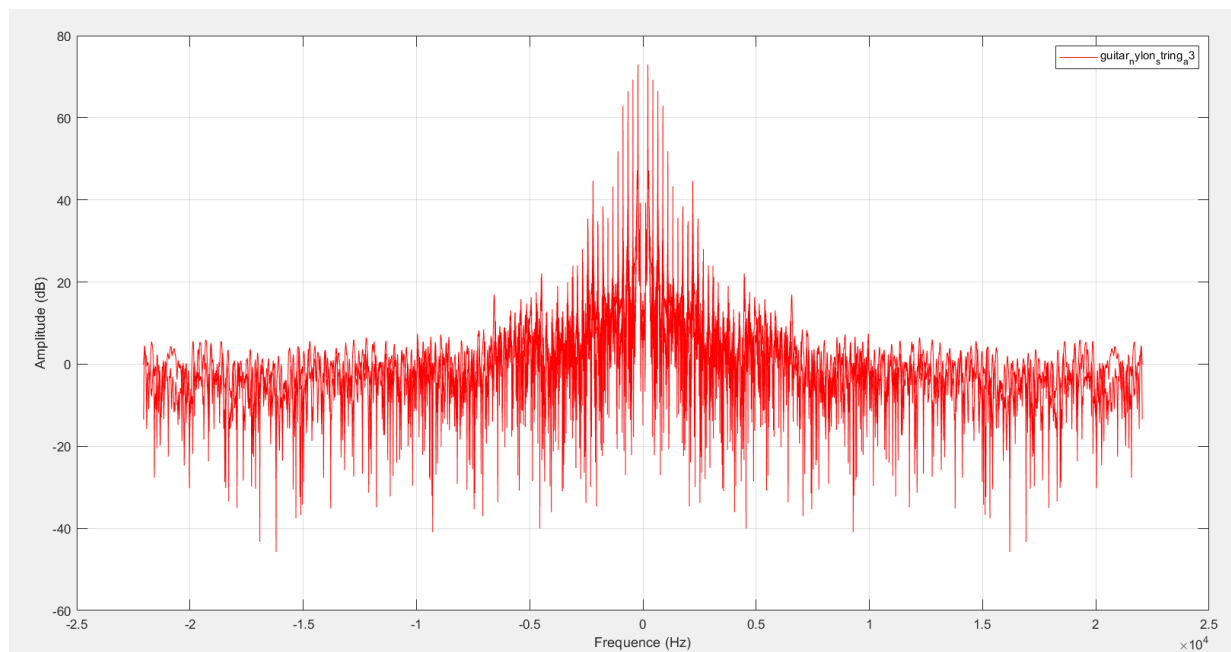


FIGURE 1 – Spectre d'amplitude de la guitare

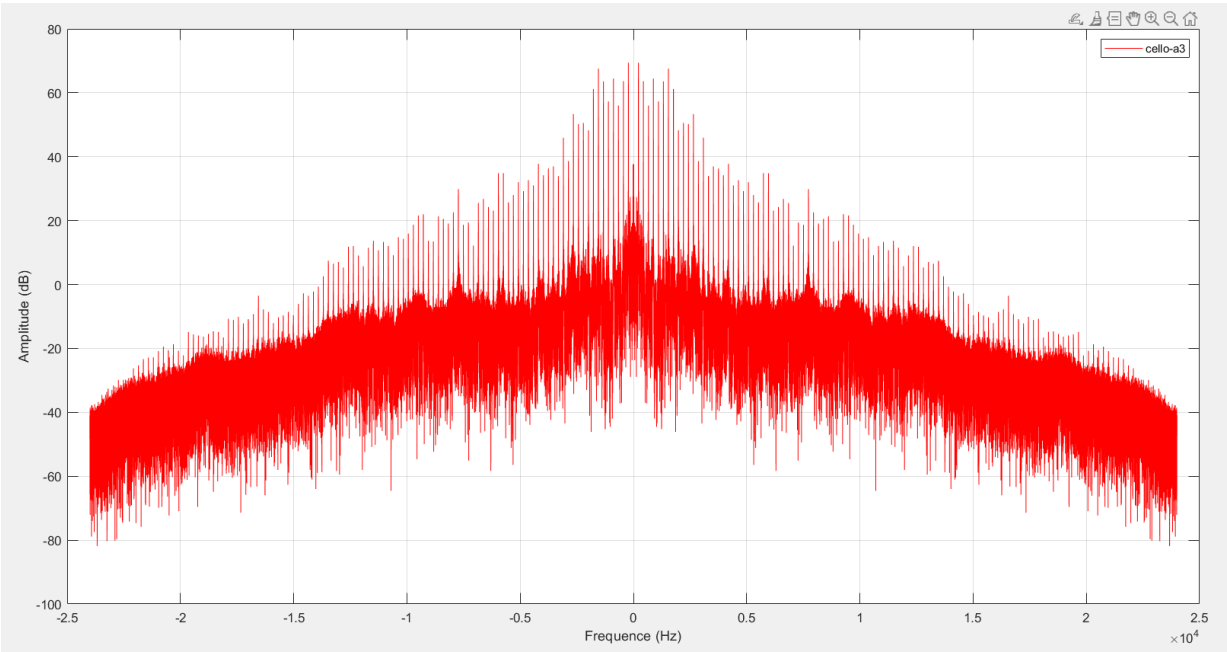


FIGURE 2 – Spectre d’amplitude du violoncelle

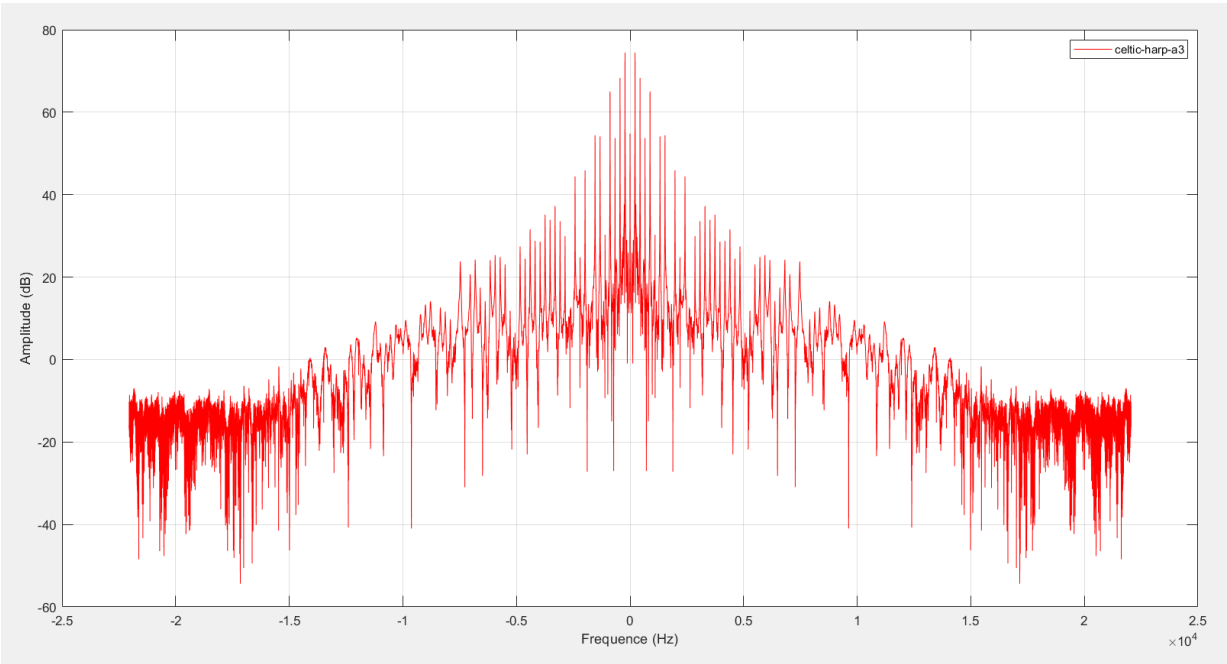


FIGURE 3 – Spectre d’amplitude de la harpe celtique

Pour chaque spectre, on répertorie dans ce tableau les fréquences fondamentales f_1 que l’on a trouvée graphiquement :

Instruments à cordes	Fréquence fondamentale en Hz
Guitare	219.27
Violoncelle	220.73
Harpe celtique	220.07

TABLE 1 – Tableau répertoriant les fréquences fondamentales des signaux de quelques instruments à cordes

Nous constatons ainsi que les fréquences fondamentales des instruments à cordes étudiés sont toutes les mêmes. Ainsi ils jouent la même note, c'est-à-dire un La3 (A3). Nous pouvons également visualiser les différentes harmoniques des signaux, certains sont plus garnis que d'autres, et les amplitudes sont différentes, ce qui produit des sons légèrement différents mais à la même note.

Question 1.2

Nous étudions ensuite deux sons de piano pour évaluer leur degré d'harmonicité. Bien que les pianos produisent un son proche d'une structure harmonique, de légères déviations apparaissent, dues à la raideur des cordes. Ces déviations sont quantifiées par l'inharmonicité, exprimée en cents, à partir de la comparaison entre les fréquences mesurées et celles théoriquement attendues.

Pour étudier ces points spécifiques, nous avons alors procédé à une méthode similaire à celle de la question précédente. Nous avons réutilisé le programme Matlab précédent en affichant les spectres des deux pianos sur le même graphique. (Voir fichier q1_2.m)

Voici les spectres des deux pianos :

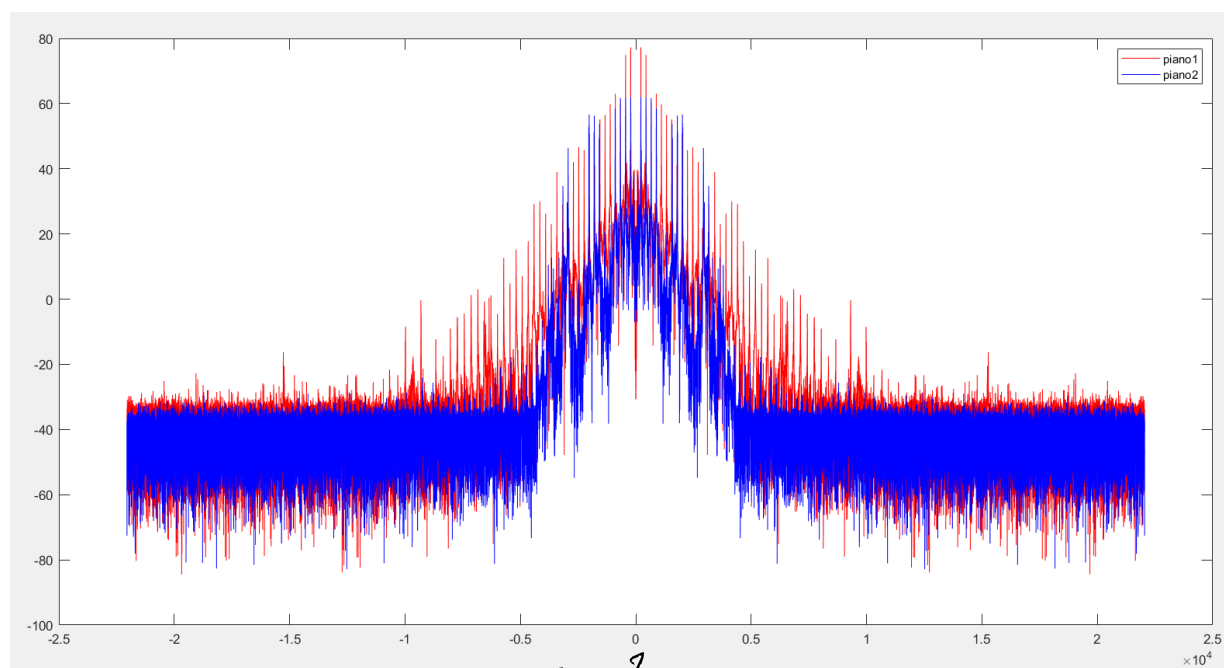


FIGURE 4 – Spectre des pianos 1 et 2

L'analyse spectrale permet de déterminer quelles composantes fréquentielles sont légèrement déplacées par rapport aux harmoniques parfaits, et donc d'identifier le piano dont le son est le plus "pur" ou harmonieux.

Le piano 1 (en rouge) est le plus harmonieux car il présente un nombre plus élevé d'harmoniques avec des amplitudes plus hautes. En effet, dans le spectre du piano 2, nous constatons que des harmoniques sont

manquantes et que le spectre semble tassé sur lui-même et donc de moins bonne qualité.

Graphiquement, on mesure la fréquence fondamentale f_1 , elle est de 220 Hz, comme vu précédemment.

Ensuite, nous avons répertorié les fréquences théoriques, mesurées et l'inharmonicité dans deux tableaux pour les deux pianos :

Fréquence (Hz)	f_2 (Hz)	f_3 (Hz)	f_4 (Hz)	f_5 (Hz)	f_6 (Hz)	f_7 (Hz)
Théorique	440	660	880	1100	1320	1540
Mesurées	442	668	885	1108	1331	1556
Inharmonicité	7.85	20.86	9.81	12.54	14.37	17.89

TABLE 2 – Comparaison des fréquences théoriques, mesurées et de l'inharmonicité pour le piano 1

Fréquence (Hz)	f_2 (Hz)	f_3 (Hz)	f_4 (Hz)	f_5 (Hz)	f_6 (Hz)	f_7 (Hz)
Théorique	440	660	880	1100	1320	1540
Mesurées	442	668	892	1118	1347	1572
Inharmonicité	7.85	20.86	23.44	28.10	35.05	35.61

TABLE 3 – Comparaison des fréquences théoriques, mesurées et de l'inharmonicité pour le piano 2

Grâce à ces tableaux, nous remarquons que, dans le cas du piano 2, l'inharmonicité est clairement plus élevée, ce qui valide notre hypothèse que le piano 1 est plus harmonieux.

1.2. Synthèse

Question 1.3

À partir du son le plus harmonique identifié, c'est-à-dire le piano 1, nous avons extrait les amplitudes des huit premières harmoniques pour générer un son synthétique par synthèse additive : une superposition d'ondes sinusoïdales aux fréquences et amplitudes correspondantes.

Pour cela, nous avons créé les fonctions sinusoïdales correspondantes à l'aide de Matlab, puis nous les avons sommées et écoutées à l'aide de la fonction audiowrite. (Voir fichier q1_3.m)

Nous obtenons bien un son semblable à celui du piano 1, même si celui-ci est quand même moins riche, moins complet que le réel.

Question 1.4

Nous améliorons ensuite ce son en y appliquant une enveloppe ADSR (Attack, Decay, Sustain, Release), qui modélise l'évolution de l'amplitude du son dans le temps, rendant le résultat plus réaliste, et plus rapproché du son du piano.

Nous développons alors un code Matlab utilisant la méthode ADSR avec des temps particuliers, puis nous appliquons cette enveloppe au signal du piano et écoutons le signal produit avec la fonction audiowrite. (Voir fichier q1_4.m)

Le son obtenu est effectivement plus réaliste et plus agréable à écouter. Il semble plus proche de celui du piano et paraît plus naturel grâce à l'enveloppe ADSR ajoutée.

Question 1.5

Enfin, une autre méthode de synthèse est testée : la reconstruction du signal via la transformée de Fourier discrète inverse, en utilisant les mêmes composantes fréquentielles. Cela permet de comparer les deux approches (somme de sinusoides vs transformation inverse) et de vérifier si elles produisent des sons identiques ou non.

Ainsi, en partant des mêmes amplitudes et harmoniques, nous avons fait une synthèse en passant cette fois par la transformée de Fourier discrète inverse, à l'aide d'un programme Matlab. (Voir fichier q1_5.M)

A l'écoute du signal obtenu, nous constatons que nous obtenons un résultat similaire mais pas exactement pareil. La synthèse additive donne un son proche, mais plus neutre et agréable à l'écoute. La transformée de Fourier discrète inverse permet de reconstruire un signal plus fidèle à l'original, notamment grâce aux phases. Ce spectre contient davantage d'informations, notamment les phases initiales des composantes, qui ne sont pas forcément prises en compte dans la synthèse additive simple.

Ainsi, la synthèse additive est plus simple et contrôlée, tandis que la reconstruction par transformation indirecte est plus complète mais paraît moins agréable à l'écoute.

2. Synthèse soustractive

La synthèse soustractive repose sur un principe opposé à celui de la synthèse additive. Plutôt que de construire un son à partir de composantes simples (sinusoïdes), elle part d'un signal riche en harmoniques (comme une onde carrée ou en dents de scie), que l'on va ensuite filtrer pour modéliser le timbre souhaité.

Question 2.1

On commence par calculer et analyser le spectre théorique de deux signaux périodiques classiques :

Le signal carré

Soit un signal carré $x(t)$ de période T et amplitude ± 1 , défini par :

$$x(t) = \begin{cases} 1 & \text{si } 0 \leq t < \frac{T}{2} \\ -1 & \text{si } \frac{T}{2} \leq t < T \end{cases}$$

La série de Fourier de $x(t)$ est donnée par :

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi n f_0 t}$$

où $f_0 = \frac{1}{T}$ est la fréquence fondamentale. Les coefficients c_n sont calculés par :

$$c_n = \frac{1}{T} \int_0^T x(t) e^{-j2\pi n f_0 t} dt$$

En séparant l'intégrale en deux parties (selon les intervalles $0 \leq t < \frac{T}{2}$ et $\frac{T}{2} \leq t < T$), on obtient :

$$c_n = \frac{2}{j\pi n} (1 - (-1)^n)$$

Ainsi, $c_n = \frac{4}{j2\pi n}$ pour n impair, et $c_n = 0$ pour n pair.

Le spectre du signal est donc :

$$X(f) = \frac{4}{\pi} \sum_{n=1,3,5,\dots}^{\infty} \frac{1}{n} \delta(f - nf_0)$$

Ce spectre contient uniquement les harmoniques impaires.

Le signal dent de scie

Soit un signal dent de scie $y(t)$ de période T et amplitude ± 1 , défini par :

$$y(t) = \frac{2}{T}t - 1 \quad \text{pour } 0 \leq t < T$$

La série de Fourier de $y(t)$ est donnée par :

$$y(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi n f_0 t}$$

où $f_0 = \frac{1}{T}$ est la fréquence fondamentale. Les coefficients c_n sont calculés par :

$$c_n = \frac{1}{T} \int_0^T y(t) e^{-j2\pi n f_0 t} dt$$

En effectuant l'intégrale, on obtient :

$$c_n = \frac{1}{j2\pi n} (1 - e^{-j2\pi n})$$

Cela peut être simplifié comme :

$$c_n = \frac{1}{j2\pi n} (1 - (-1)^n)$$

Ainsi, $c_n = \frac{2}{j2\pi n}$ pour $n \neq 0$.

Le spectre du signal est donc :

$$Y(f) = \sum_{n=-\infty}^{\infty} \frac{1}{j2\pi n} \delta(f - nf_0)$$

Ce spectre contient toutes les harmoniques entières.

Ces résultats théoriques peuvent être vérifiés numériquement en générant ces signaux dans Matlab (carré et dent de scie), puis en utilisant la transformée de Fourier discrète (fft) pour visualiser leur spectre fréquentiel et observer si celui-ci est en accord avec nos calculs.

Néanmoins, cette méthode présente quelques limites. En effet, on doit avoir une fréquence d'échantillonnage assez élevée pour éviter les problèmes de bord et le spectre peut souffrir de fuites dues à la discontinuité du signal dent de scie par exemple. Une autre limite peut être les effets de troncature (fenêtrage).

Question 2.2

Nous allons ensuite appliqué un filtre passe-bas d'ordre 1 à ces signaux, et supprimé progressivement les hautes fréquences, ce qui permet de "sculpter" le spectre pour obtenir un son plus doux.

Nous considérons un filtre passe-bas d'ordre 1 défini par la relation suivante :

$$y(k) = \frac{1}{2} (x(k) + x(k-1))$$

Sa fonction de transfert est :

$$H(z) = \frac{1}{2}(1 + z^{-1})$$

En remplaçant z par $e^{j2\pi f}$ (avec f la fréquence normalisée), on obtient la réponse fréquentielle :

$$H(f) = \frac{1}{2} (1 + e^{-j2\pi f})$$

Le module de cette fonction est donné par :

$$|H(f)| = \left| \frac{1}{2} (1 + e^{-j2\pi f}) \right| = |\cos(\pi f)|$$

Ce filtre laisse donc passer les composantes basses fréquences ($f = 0 : |H(f)| = 1$) et atténue les hautes fréquences ($f = 0,5 : |H(f)| = 0$). Il agit comme un filtre passe-bas simple, réduisant l'énergie des harmoniques les plus élevées.

Pour obtenir le spectre de sortie d'un signal filtré, il suffit de multiplier le spectre du signal d'entrée $X(f)$ par la réponse en fréquence $H(f)$:

$$Y(f) = H(f) \cdot X(f)$$

→ aurait été encore mieux de développer les calculs.

Ce résultat est ensuite comparé à une simulation numérique réalisée avec Matlab, en utilisant la commande :

```
y = filter([0.5 0.5], 1, x);
```

où x est le signal d'entrée périodique (signal carrée ou en dent de scie). (Voir fichier q2_2.m)

Le spectre de sortie montre que bien que les harmoniques proches de la fréquence fondamentale sont préservées et que celles plus élevées sont progressivement atténuées. L'effet global est un lissage du timbre, avec un son plus doux.

Question 2.3

Nous comparons ensuite les sons obtenus par synthèse soustractive à ceux produits par la synthèse additive dans la section précédente. Bien que les deux méthodes puissent générer des sons similaires, la synthèse soustractive offre un contrôle plus intuitif du timbre à partir d'un son brut riche (dents de scie ou carrés), alors que la synthèse additive demande un réglage précis des fréquences et amplitudes de chaque harmonique.

Les différences sont particulièrement audibles en jouant sur la forme du signal source : une onde carrée produit un son plus "creux", tandis que la dent de scie donne un timbre plus "brillant".

Question 2.4

Pour améliorer la qualité sonore, on utilise des filtres plus élaborés via la fonction `designfilt()` de Matlab. Cette approche permet de spécifier la fréquence de coupure, l'ordre du filtre, et sa structure, ce qui influe fortement sur la précision du filtrage et donc sur le rendu final.

En expérimentant avec différents paramètres (par exemple, en augmentant l'ordre du filtre ou en abaissant sa fréquence de coupure), on affine le spectre de sortie pour obtenir un son plus proche de l'instrument souhaité. L'outil `filterAnalyzer()` permet d'analyser visuellement les caractéristiques du filtre pour guider ces ajustements. (Voir fichier `q2_4.m`)

3. Effets audio numériques

Dans cette dernière partie, nous étudions un effet de delay enrichi par un filtrage dans la boucle de retour. L'objectif est d'améliorer la qualité sonore en simulant la perte naturelle des hautes fréquences lors des réflexions acoustiques. Cette modification permet d'obtenir un son plus chaud et moins artificiel que celui produit par un delay simple.

3.1. Effets de réverbération

Question 3.1

Nous exprimons la fonction d'intercorrélation R_{yx} entre le signal d'excitation x et le signal capté y en fonction de la réponse impulsionnelle h et de la fonction d'autocorrélation R_{xx} . Cette relation fondamentale permettra d'estimer la réponse impulsionnelle de la pièce.

On a :

$$y(u) = (h * x)(u)$$

En auto-corrélant, on obtient :

$$R_{yx}(u) = (h * R_{xx})(u)$$

↓ détails?

Question 3.2

En supposant que la fonction d'autocorrélation R_{xx} se comporte comme une impulsion unité, la réponse impulsionnelle x peut être estimée directement à partir de la fonction d'intercorrélation R_{yx} .

On estime que $R_{xx}(u) = d(u)$. Ainsi :

$$R_{yx}(u) = (h * d)(u)$$

$$R_{yx}(u) = h(u)$$

Une méthode pour déterminer la réponse impulsionnelle est donc de faire l'intercorrélation du signal envoyé avec le signal reçu.

Question 3.3

En comparant les fonctions d'autocorrélation des deux signaux d'excitation `xe1` et `xe2` à l'aide de la fonction `xcorr` de Matlab, nous déterminons quel signal présente une autocorrélation la plus proche d'une impulsion unité, ce qui le rend le plus approprié pour mesurer la réponse impulsionnelle. (Voir fichier `q3_3.m`)

Nous affichons tout d'abord l'autocorrélation des deux variables.

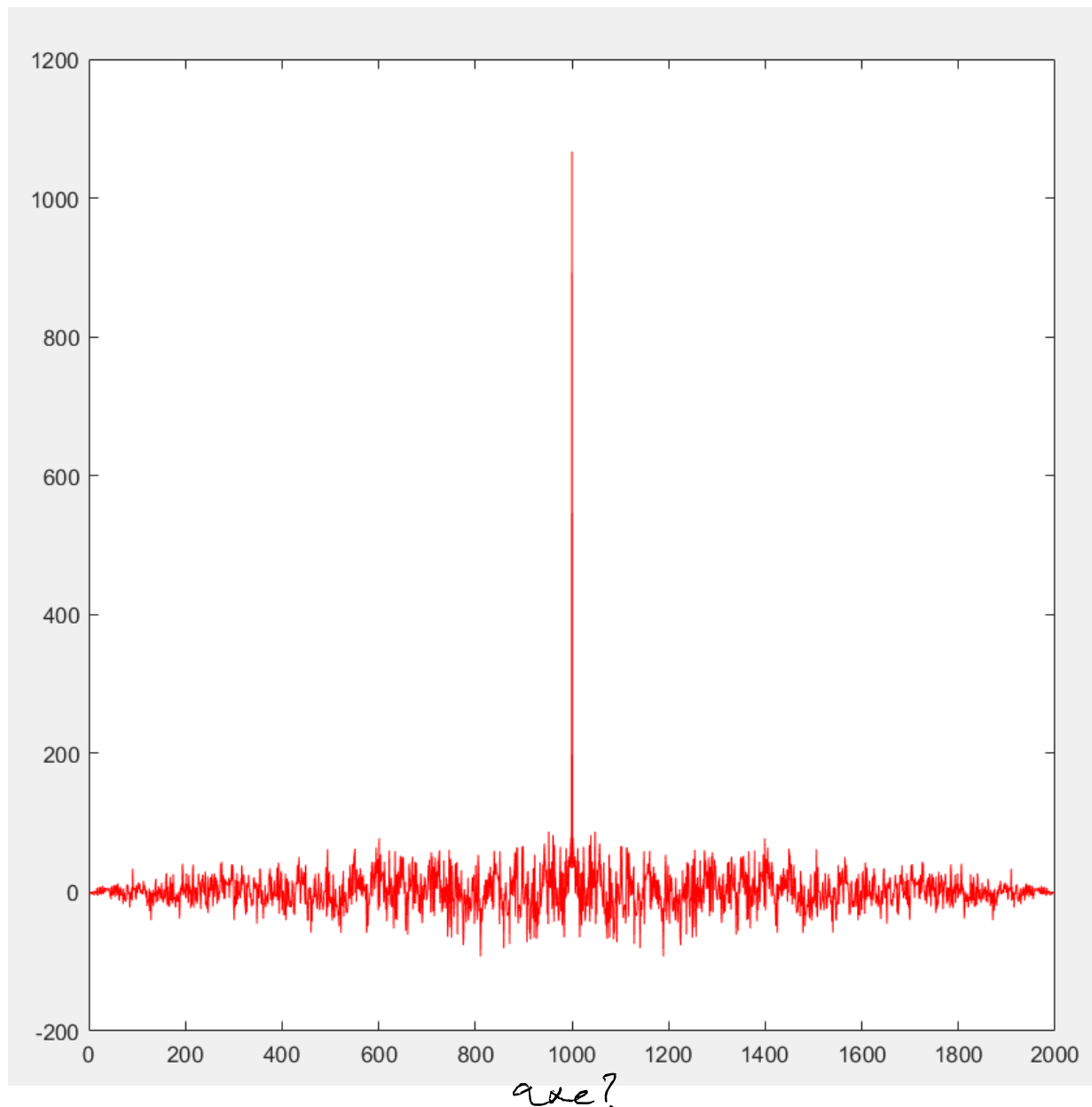


FIGURE 5 – Autocorrélation du signal xe1

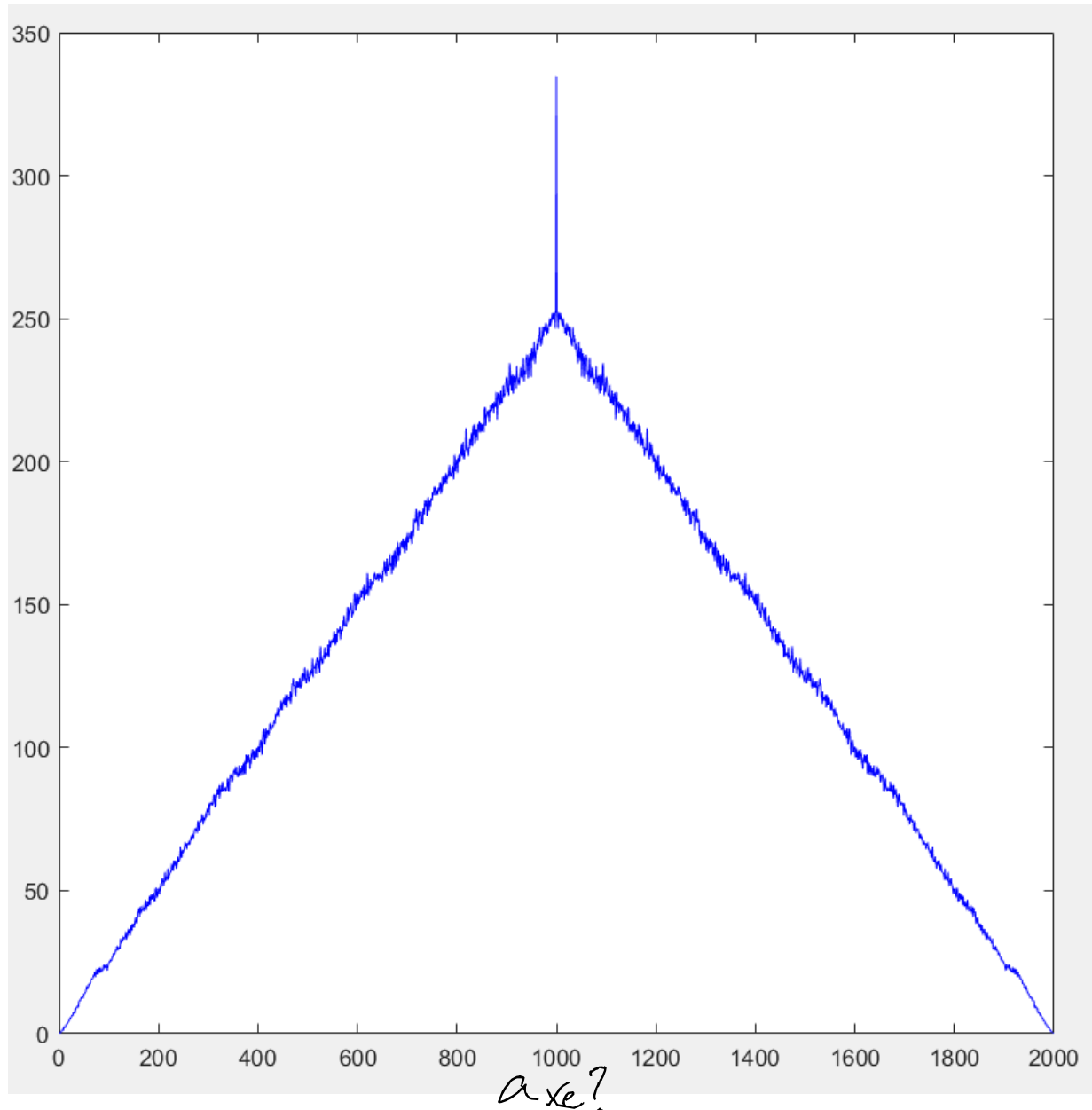


FIGURE 6 – Autocorrélation du signal xe2

Nous avons ainsi constaté que le signal le plus proche d'un dirac est **xe1**. Il semblerait ainsi être le plus adapté comme signal d'excitation pour mesurer la réponse impulsionnelle de la pièce. ✓

Question 3.4

Nous mettons en œuvre la méthode d'estimation de la réponse impulsionnelle en convoluant le signal d'excitation choisi avec la réponse simulée de la pièce, puis en calculant la fonction d'intercorrélation via `xcorr`. Le résultat est représenté graphiquement pour visualiser la réponse impulsionnelle estimée en fonction du temps. (Voir fichier `q3_4.m`)

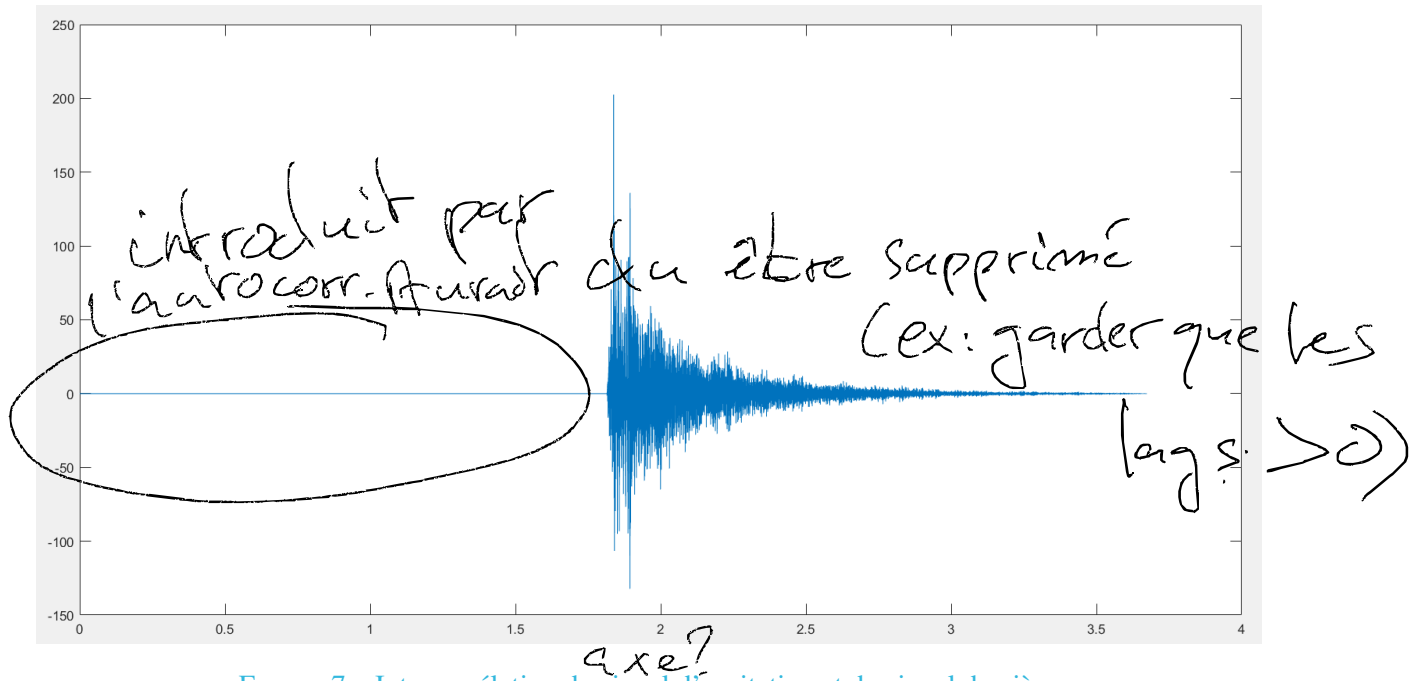


FIGURE 7 – Intercorrélation du signal d'excitation et du signal de pièce

3.2. Convolution classique

Question 3.5

Nous programmons la fonction `effet_reverb` qui réalise la convolution classique entre un signal d'entrée et une réponse impulsionnelle estimée. Cette opération est réalisée via la fonction Matlab `filter` pour simuler l'effet de réverbération. (Voir fichier `effet_reverb.m`)

Question 3.6

En testant la fonction `effet_reverb` sur un son de guitare, nous mesurons le temps d'exécution à l'aide des commandes `tic` et `toc` afin d'évaluer la performance de l'implémentation classique de la convolution. (Voir fichier `test_effet_reverb.m`)

Nous obtenons alors un temps d'exécution de 7.9111 s, ce qui est plutôt lent.

3.3. Convolution rapide

Question 3.7

Pour réduire le temps de calcul, nous implémentons la fonction `effet_reverb_FFT` qui réalise la convolution dans le domaine fréquentiel en utilisant la FFT et l'IFFT. (Voir fichier `effet_reverb_FFT`)

Nous la testons ensuite avec un programme Matlab semblable à celui des questions précédentes. (Voir fichier `test_effet_reverb_FFT`)

Le temps de calcul nécessaire à l'exécution de cette fonction est de 0.16 s. C'est clairement plus rapide que pour la convolution directe, surtout pour des signaux de grande taille.

Question 3.8

La convolution par FFT est mathématiquement équivalente à la convolution classique, à condition d'utiliser un nombre de points suffisant pour éviter le repliement. Elle donne donc le même résultat, tout en

étant bien plus efficace en termes de calcul.

3.4. Effet de retard

Question 3.9

Nous avons la relation de récurrence :

$$y(k) = x(k) - gy(k - \tau)$$

Montrons par récurrence que la réponse impulsionnelle théorique est égale à :

$$h(k) = (-g)^n \text{ si } k = n\tau$$

$$h(k) = 0 \text{ sinon}$$

On recherche la réponse impulsionnelle théorique $h(k)$, c'est-à-dire la sortie du système lorsque $x(k) = \delta(k)$ et que $h(k) = y(k)$

Ainsi la formule de récurrence devient :

$$h(k) = \delta(k) - gh(k - \tau)$$

Initialisation : Cas où $k = n\tau = 0$

$$h(0) = \delta(0) - gh(0 - \tau) = 1 - gh(-\tau)$$

Or le filtre est causal, donc pour tout $k < 0$, $h(k) = 0$, donc $h(-\tau) = 0$.

Ainsi, on a :

$$h(0) = 1 = g^0$$

Hérédité : Supposons la propriété vraie au résultat $k = n\tau$. Montrons qu'elle est vraie pour $k = (n+1)\tau$.

$$h((n+1)\tau) = \delta((n+1)\tau) - gh((n+1)\tau - \tau)$$

$$h((n+1)\tau) = \delta((n+1)\tau) - gh(n\tau)$$

Or par récurrence, on a : $h(n\tau) = (-g)^n$

$$h((n+1)\tau) = \delta((n+1)\tau) - g(-g)^n = 0 + (-g)^{(n+1)}$$

D'où le résultat, pour tout $k = n\tau$. Sinon, on a bien $h(k) = 0$ puisque le dirac sera nul et la réponse impulsionnelle également.

Question 3.10

D'après l'expression de h trouvée précédemment, on déduit simplement que pour que h converge, il faut $|g| < 1$ ie $0 < g < 1$.

Nous déterminons que la condition nécessaire à la stabilité du filtre de delay est que le coefficient d'amortissement g soit strictement inférieur à 1 en valeur absolue, assurant ainsi que les réponses successives décroissent.

Question 3.11

À partir de l'équation du filtre delay, nous identifions les vecteurs de coefficients a et b utilisés dans la fonction filter de Matlab, qui caractérisent respectivement la partie récursive et la partie directe du filtre.

Ainsi nous obtenons : $a = [1 \ 0 \ \dots \ 0 \ -g]$ avec $\tau - 1$ zéros et $b = [1 \ 0 \ \dots]$ avec τ zéros.

Question 3.12

Nous réalisons un script permettant d'obtenir la réponse impulsionnelle du filtre delay en utilisant la fonction filter, et présentons son tracé temporel pour visualiser l'effet du retard et de l'atténuation. (Voir fichier analyse_delay)

Voici la figure obtenue :

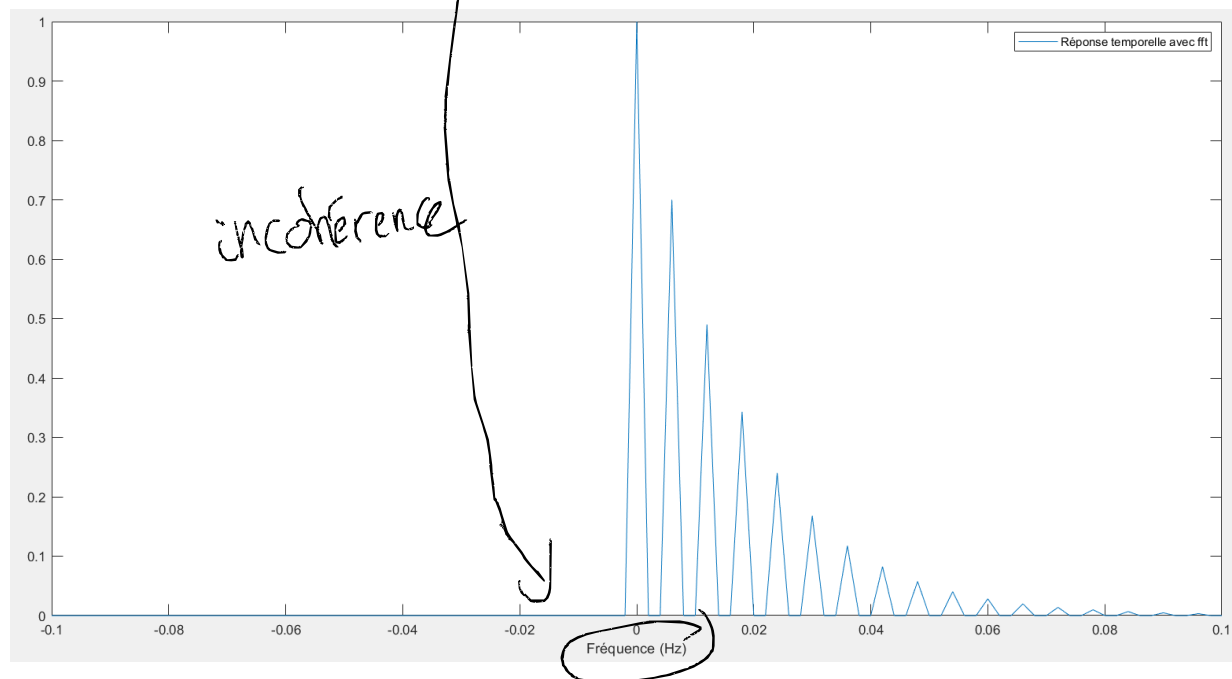


FIGURE 8 – Réponse impulsionnelle de h

La figure 8 représente la réponse temporelle de h à un dirac de hauteur 1 en $t = 0$. La fréquence d'échantillonnage est de 500 Hz, $g = 0.7$ et $\tau = 3$.

Question 3.13

Calculons maintenant la réponse en fréquence du filtre. On a :

$$h(k) = (-g)^n \text{ si } k = n\tau$$

$$h(k) = 0 \text{ sinon}$$

En passant dans le domaine fréquentiel, on obtient :

$$\hat{h}(\nu) = \sum_{k \in \mathbf{Z}} h(k) \exp(-2kj\pi\nu\tau)$$

En utilisant la formule calculée à la question 3.9 :

$$\hat{h}(\nu) = \sum_{n\tau \in \mathbf{Z}} (-g)^n \exp(-2nj\pi\nu\tau)$$

On remarque une somme géométrique, où $|-g \exp(-2j\pi\nu\tau)| = |g| < 1$, donc elle converge et on connaît le résultat :

$$\hat{h}(\nu) = \sum_{n\tau \in \mathbf{Z}} (-g \exp(-2j\pi\nu\tau))^n$$

$$\hat{h}(\nu) = \frac{1}{1 + g \exp(-2j\pi\nu\tau)}$$

Le module de cette réponse est :

$$|\hat{h}(\nu)| = \left| \frac{1}{1 + g \exp(-2j\pi\nu\tau)} \right|$$

$$|\hat{h}(\nu)| = \frac{1}{\sqrt{(1 + g \cos(2\pi\nu\tau))^2 + g^2 \sin^2(2\pi\nu\tau)}}$$

$$|\hat{h}(\nu)| = \frac{1}{\sqrt{1 + 2g \cos(2\pi\nu\tau)}} \rightarrow g^2$$

La phase de cette réponse est :

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

Question 3.14

↑ +, pas -

Nous traçons sur un même graphique la réponse en fréquence théorique et celle obtenue numériquement par la FFT de la réponse impulsionnelle. (Voir fichier analyse_delay

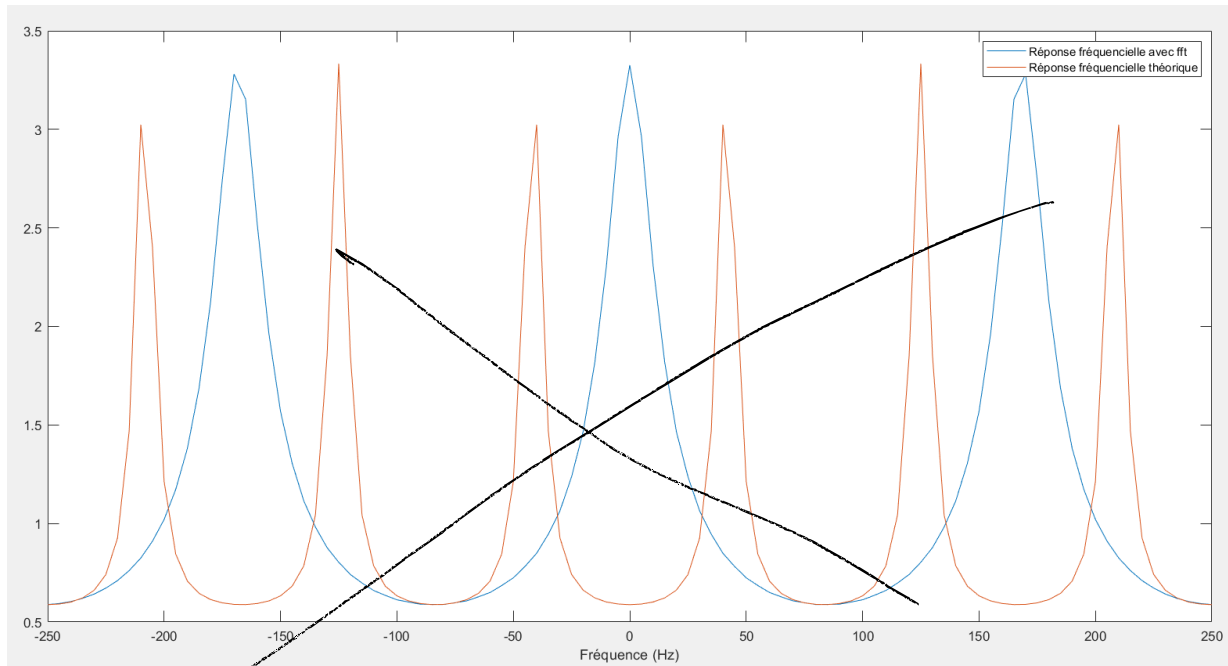


FIGURE 9 – Réponse fréquentielle du filtre

La figure 9 représente les deux réponses fréquentielles, la théorique en rouge et celle obtenue grâce à la fonction fft en bleu. Les paramètres sont les mêmes que précédemment.

Les deux courbes présentent une structure en peigne, avec des pics réguliers correspondant aux fréquences où l'énergie se renforce à cause du retard.

De légères différences peuvent apparaître dues à la troncature de la réponse impulsionnelle (limitée dans le temps pour le calcul numérique) ou à des effets de discrétisation. Toutefois, le comportement global reste similaire, validant la cohérence entre théorie et simulation.

Question 3.15

Nous programmons la fonction `effet_delay` qui applique l'effet delay sur un signal donné en fonction des paramètres de retard, coefficient d'amortissement et fréquence d'échantillonnage, utilisant la relation de récurrence du filtre. (Voir fichier `effet_delay`)

Question 3.16

Nous testons la fonction sur un accord de piano avec des paramètres spécifiques, et observons à l'écoute que nous avons l'accord original joué plusieurs fois, chaque fois un peu moins fort, avec une période de 0,25 s.

Question 3.17

Nous développons la fonction `effet_delay_filtre` qui améliore le delay en insérant un filtre moyenneur dans la boucle de retour, ce qui permet de moduler la couleur sonore en atténuant les hautes fréquences du signal réfléchi. (Voir fichier `effet_delay_filtre`)

Question 3.18

Le script de test est modifié pour utiliser la fonction `effet_delay_filtre` avec des paramètres précis. Ici, nous entendons encore les mêmes accords qu'à la question 3.16, mais cette fois les aigus sont atténués

beaucoup plus vite que les graves.

Question 3.19

Nous justifions l'effet de coupure progressive des hautes fréquences observé à l'écoute en traçant la réponse en fréquence du filtre moyenneur utilisé dans la boucle de feedback.

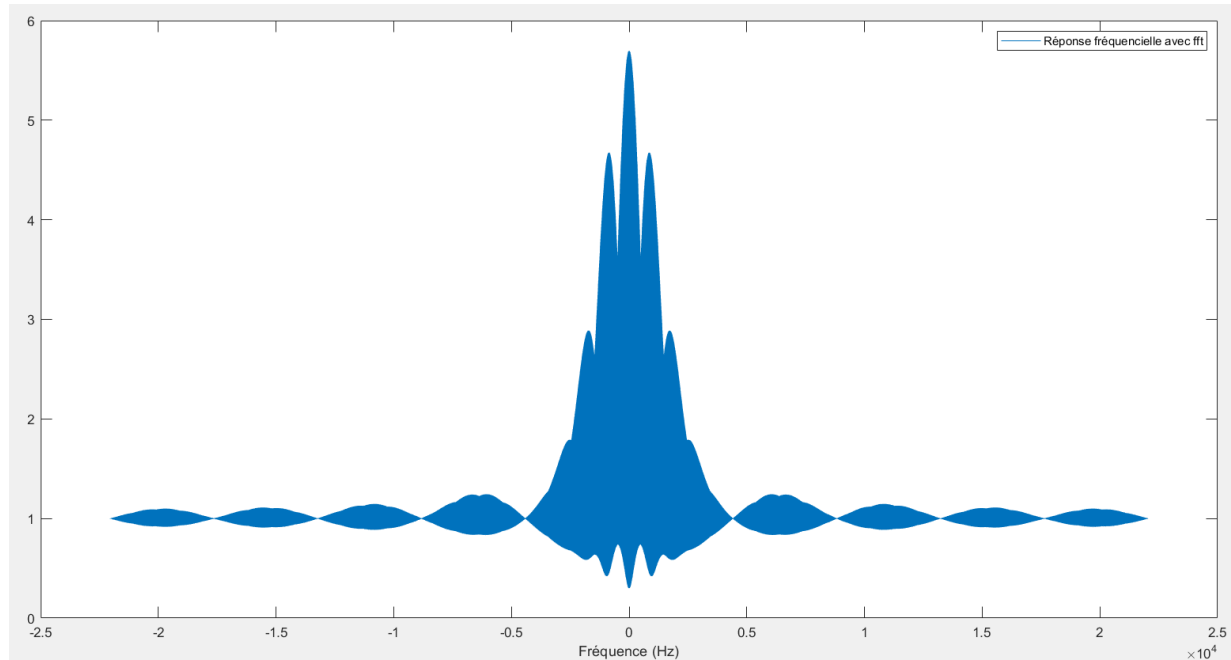


FIGURE 10 – Réponse fréquentielle de h_r

Cette réponse montre clairement une atténuation progressive qui explique la coloration sonore plus douce et naturelle.

Pour aller plus loin : synthèse Karplus-Strong

Il est demandé dans cette partie d'implémenter un filtre défini par un schéma. On obtient après analyse du schéma la relation suivante :

$$y(k) = x(k - N) + \alpha y(k - N) + (1 - \alpha)gy(k - 1 - N)$$

On en déduit les valeurs des vecteurs a et b , à savoir : $a = [1 \text{ zeros}(1, N-1) \ g*\alpha \ g*(1-\alpha)]$ et $b = [\text{zeros}(1, N) \ 1 \ 0]$.

Ainsi, on peut aisément coder cette synthèse de la même manière que `delay_filtre`. Après tests, nous avons implémenté l'enveloppe ~~ADRS~~ **ADSR** travaillée précédemment pour gagner en fidélité.

Pour réaliser le test de cette fonction, nous avons pris un fichier séparé et nous avons pris en entrée un bruit blanc gaussien concaténé avec du silence (de manière à entendre résonner la corde). Nous avons alors écouté le résultat et fait varier nos paramètres en conséquence, en jouant en particulier sur `T_note`, la durée d'excitation de la corde. Si nous la laissions élevée, le son se rapprochait de celui d'un violon, tandis qu'en la mettant proche d'une milliseconde, nous avions un son proche d'une guitare, ce qui était le but de cette synthèse. (Voir fichier `karplus_strong.m` et `test_karplus_strong.m`)