



UE Electrical Engineering

SAR Traitements audio : partie signal

Date d'édition : 23 avril 2025
Version : 1.0



IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Sommaire

Introduction	3
Avant-propos	3
1. Synthèse additive	3
1.1. Analyse d'un son harmonique	3
1.2. Synthèse	4
2. Synthèse soustractive	5
3. Effets audio-numériques	5
3.1. Effets de réverbération	6
3.2. Effet de retard	7
4. Pour aller plus loin : synthèse Karplus-Strong	9

Liste des figures

1.	Une interface de controle d’enveloppe ADSR sur un synthétiseur Roland (1980) [?]	4
2.	Principe de la génération soustractive	5
3.	Effet de delay (le symbole $z^{-\tau}$ désigne un retard de τ échantillons).	7
4.	Effet de delay ($H_r(z)$ désigne un filtre de réponse impulsionnelle $h_r(k)$).	9
5.	Synthèse Karplus-Strong (modèle avec perte d’énergie) [?]	9

Liste des tableaux

1.	Composantes fréquentielles d’un son de piano	4
----	--	---

Introduction

L'objectif de ce projet est d'illustrer et d'appliquer les notions vues dans le cours de traitement du signal. Ce projet est organisé en quatre sections :

1. Avant-propos : présentation de quelques fonctions matlab utiles à la réalisation du projet.
2. Echantillonnage : application et illustration du théorème de Shannon-Nyquist.
3. Analyse fréquentielle : application de la transformée de Fourier discrète.
4. Effets audio-numériques : implémentation de systèmes linéaires invariants dans le temps.

Ce projet sera réalisé à l'aide de l'environnement de développement Matlab. Il est donc INDISPENSABLE de travailler en amont sur le tutoriel de prise en main de cet environnement disponible sur Moodle. Merci également de venir en séance avec un casque audio.

Pour rappel, un compte-rendu de travaux pratiques comprend :

1. un titre,
2. une introduction,
3. une présentation théorique du problème
4. une présentation des protocoles expérimentaux mis en oeuvre
5. des illustrations des résultats obtenus
6. des analyses et interprétations de ces résultats
7. une conclusion

Avant-propos

Dans cette partie, nous allons utiliser le logiciel Matlab. Une introduction et prise en main de ce logiciel vous est proposée dans l'annexe ??.

Lorsque les sons que nous allons traiter sont stockés dans des fichiers, ou si nous souhaitons sauvegarder nos propres sons dans des fichiers, nous utiliserons le format .wav (non compressé).

Le chargement et la lecture d'un fichier sonore s'obtiennent en exécutant les fonctions suivantes :

```
>> [x, fe] = audioread(filename);  
>> soundsc(x, fe);
```

où fe désigne la fréquence d'échantillonnage (en Hz). La variable x contient les échantillons du signal. Cette variable est un vecteur de taille $\mathbb{R}^{N \times 1}$ dans le cas d'un fichier monophonique ou une matrice de taille $\mathbb{R}^{N \times 2}$ dans le cas d'un fichier stéréophonique. Des informations complémentaires sur le fichier audio, comme par exemple le nombre de bits de quantification utilisés, peuvent être obtenues en appelant la fonction `audioinfo`.

La sauvegarde d'un fichier son sur le disque dur se réalise comme suit :

```
>> audiowrite(filename, x, fe);
```

où la fréquence d'échantillonnage doit être un entier.

1. Synthèse additive

1.1. Analyse d'un son harmonique

Question 1.1 Visualiser le spectre d'amplitude obtenu avec une transformée de Fourier discrète de quelques instruments à cordes. Comparer-les avec d'autres instruments. L'amplitude du spectre sera représentée en décibels et pour les fréquences contenues entre $-\frac{f_e}{2}$ et $\frac{f_e}{2}$ (utiliser les commandes `fft()` et `fftshift()`). Déterminez la fréquence fondamentale f_1 du signal.

Pour la plupart des instruments à cordes, le contenu fréquentiel d'une note est (quasi) harmonique, c'est-à-dire que les composantes fréquentielles sont placées sur des multiples entiers de la fréquence fondamentale, et données par

$$f_n = n f_1, \quad (1)$$

où f_1 correspond à la fréquence fondamentale et $n \in \mathbb{N}$. Il est remarquable de constater que les signaux subjectivement (ou culturellement) agréables à l'oreille comportent une structure mathématique aussi singulière.

En réalité, les instruments à cordes présentent rarement une structure harmonique aussi parfaite. En effet, l'équation qui régit le mouvement d'une corde réelle est en fait une combinaison de l'équation d'onde de la corde idéale et de celle d'une tige fixée aux deux extrémités pour laquelle les fréquences propres suivent une progression en n^2 . Le degré d'inharmonicité ξ se mesure en centième de demi-ton ou *cent* (i.e., rapport de fréquence de $2^{1/1200}$). Ce degré d'inharmonicité s'obtient pour la n -ième harmonique comme suit :

$$\xi = 1200 \times \left(\log_2(\hat{f}_n) - \log_2(n f_1) \right),$$

où \hat{f}_n est la n -ième fréquence propre mesurée sur le spectre.

Nous proposons ici de constater la présence d'inharmonicité sur les 2 sons de piano.

Question 1.2 Visualisez sur une même figure, la représentation spectrale des sons de piano enregistrés dans les fichiers `piano1.wav` et `piano2.wav`. Quel est le piano le plus harmonieux ? Justifiez. A partir de la représentation fréquentielle précédente et de la valeur de f_1 précédemment relevée, déterminez l'emplacement fréquentiel théorique (voir (1)) et mesuré des différentes composantes. Déduisez-en l'inharmonicité de ces 2 pianos en cents. La commande Matlab pour calculer le \log_2 est : `log2`.

Fréquence (Hz)	f_2 (Hz)	f_3 (Hz)	f_4 (Hz)	f_5 (Hz)	f_6 (Hz)	f_7 (Hz)
Théoriques						
Mesurées						
Inharmonicité						

TABLE 1 – Composantes fréquentielles d'un son de piano

1.2. Synthèse

Question 1.3 Pour le son le plus harmonique, récupérer les amplitudes des harmoniques et générer un son en superposant les ondes sinusoidales aux fréquences et amplitudes correspondantes (se limiter aux 8 premières harmoniques).

Les sons générés par les instruments à cordes n'ont pas une enveloppe constante en fonction du temps. L'enveloppe couramment utilisée en synthèse sonore est l'enveloppe dite ADSR [?].

Question 1.4 Ajouter une modification d'enveloppe de type ADSR pour mieux approcher le son du piano.

Question 1.5 En partant des mêmes amplitudes et harmoniques, faites une synthèse en passant cette fois par la transformée de Fourier discrète inverse. Obtient-on le même résultat ?



FIGURE 1 – Une interface de contrôle d'enveloppe ADSR sur un synthétiseur Roland (1980) [?]

2. Synthèse soustractive

Dans la section précédente, la génération d'une note a été faite en partant de la fréquence fondamentale enrichie par des harmoniques. Nous allons maintenant aborder une autre technique partant d'un signal possédant naturellement des harmoniques (signal périodique) que nous allons ensuite filtrer. C'est le principe utilisé dans le célèbre synthétiseur TB-303 qui joua un rôle important dans le développement de la musique électronique dans les années 1980 [?]. Ce principe est illustré sur la figure 2.

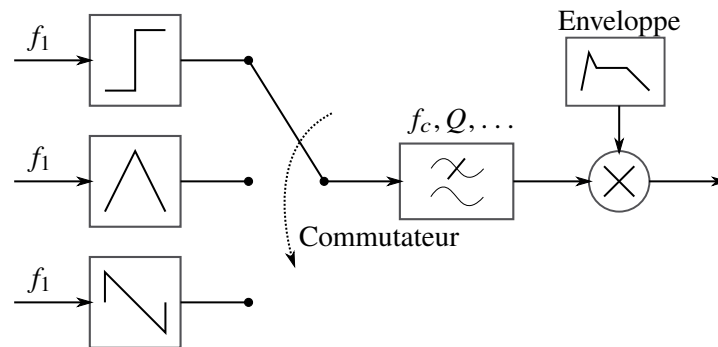


FIGURE 2 – Principe de la génération soustractive

Question 2.1 Calculer le spectre d'un signal carré (centré, amplitude ± 1 , période T) ? d'un signal dent de scie (centré, amplitude ± 1 , période T) ? Proposez une méthode pour vérifier numériquement vos calculs en indiquant les limites de cette méthode.

Matlab possède une commande permettant de calculer directement la réponse d'un filtre :

```
>> y = filter(b, a, x);
```

où :

- x est un vecteur contenant le signal d'entrée,
- $a = [a_0, \dots, a_N]$ est un vecteur comportant les coefficients de la partie réursive,
- $b = [b_0, \dots, b_M]$ est un vecteur comportant les coefficients de la partie non-réursive.

Question 2.2 En utilisant le filtrage passe-bas d'ordre 1 suivant : $y(k) = \frac{1}{2}(x(k) + x(k-1))$, calculer le spectre théorique de sortie (en négligeant le repliement spectral) et le vérifier par simulation.

Question 2.3 En utilisant les mêmes enveloppes, écouter et comparer les sons produits par synthèse soustractive en changeant le signal périodique de source, et comparer également à la synthèse additive.

Pour améliorer le rendu, un filtre d'ordre plus élevé peut être utilisé. La fonction matlab `designfilt` permet d'obtenir un filtre selon certaines spécifications. Par exemple la commande :

```
d = designfilt('lowpassfir', 'FilterOrder', 2, 'HalfPowerFrequency', 0.25)
```

permet d'obtenir un filtre RIF passe-bas d'ordre 2 ayant une fréquence réduite de coupure de 0.25. La fonction `filterAnalyzer(d)` permet de faire une analyse du filtre passé en paramètre. Dans ces fonctions, d est une structure dont les champs sont accessibles de manière classique, comme par exemple `d.Numerator` pour avoir le vecteur des coefficients du numérateur de la fonction de transfert.

Question 2.4 En changeant la fréquence de coupure, l'ordre du filtre ou sa nature de type RIF ou RII, essayer d'améliorer le son produit par synthèse soustractive.

3. Effets audio-numériques

Nous allons nous intéresser à l'implémentation de systèmes linéaires invariants en temps en prenant comme cadre d'application les effets audio-numériques [?, ?][Zolzer, Choqueuse et al.]. Ces effets modifient un signal audio numérisé souvent dans un but artistique mais aussi parfois pour en corriger ses défauts.

3.1. Effets de réverbération

La réverbération permet d'ajouter de l'espace au son en simulant sa propagation dans un environnement le plus souvent fermé (salle, amphithéâtre, arène, église, etc). Physiquement, la réverbération est un mécanisme complexe faisant intervenir des phénomènes de réflexion, réfractions, etc. Ces phénomènes sont traités par une discipline nommée l'"acoustique des salles". Lorsque l'environnement n'évolue pas en fonction du temps, la réverbération peut être modélisée par un système linéaire invariant en temps. La réverbération est alors complètement décrite par sa réponse impulsionnelle.

Différentes approches peuvent être utilisées pour simuler les effets de réverbération.

- L'approche algorithmique (algorithme de Moorer, Feedback Delay Network,...),
- L'approche physique (lancer de rayon, technique image-source, méthode des éléments-finis),
- L'approche par convolution.

Dans cette section, nous allons implémenter la troisième approche. L'approche par convolution s'implémente facilement et produit des résultats très convaincants. Elle est composée des étapes suivantes :

1. Mesure de la réponse impulsionnelle $h(k)$ (l'empreinte acoustique) d'un lieu.
2. Convolution d'un son $x(k)$ par la réponse impulsionnelle.

3.1.1. Mesure de réponse impulsionnelle

La mesure de réponse impulsionnelle d'une salle est très utile pour contrôler l'acoustique de celle-ci ou pour recréer cette acoustique virtuellement en appliquant un effet de réverbération à un signal d'intérêt. Cette mesure n'est pas aussi aisée qu'on pourrait le penser car générer une impulsion acoustique est assez complexe et pose quelques problèmes en terme de répétabilité de la mesure et de saturation des éléments de la chaîne de captation (microphones ou haut-parleurs, convertisseur analogique-numérique, etc). Parmi les différentes alternatives de mesure possibles, nous allons nous intéresser à la méthode dite *pseudo-impulsionnelle* qui repose sur la génération de séquences ayant une fonction d'autocorrélation proche de l'impulsion unité.

Soit $x(k)$, $k \in \mathbb{Z}$, le signal d'excitation et $y(k)$ le signal capté par un microphone positionné dans une pièce. On note $h(k)$ la réponse impulsionnelle de la pièce (incluant également la réponse de la chaîne de génération-captation). On note $R_{yx}(u)$ la fonction d'intercorrrelation entre y et x et $R_{xx}(u)$ la fonction d'autocorrélation de x .

Question 3.1 Exprimez R_{yx} en fonction de R_{xx} et de h .

Question 3.2 En supposant que $R_{xx}(u) \approx d(u)$, en déduire une méthode d'estimation de réponse impulsionnelle.

Question 3.3 Chargez dans votre espace de travail le fichier `signal_excitation.mat`. Il contient les signaux `xe1` et `xe2` ainsi que leur fréquence d'échantillonnage. En vous aidant de la fonction `xcorr` de Matlab, lequel des deux signaux vous semble le plus adapté comme signal d'excitation pour mesurer la réponse impulsionnelle de la pièce ?

Question 3.4 A l'aide la fonction `xcorr`, mettez en place la méthodologie de mesure proposée précédemment. L'effet de la pièce sur le signal d'excitation choisi sera simulé en appelant la fonction `simule_piece`. Affichez la réponse impulsionnelle estimée en fonction du temps.

3.1.2. Convolution classique

L'application de l'effet de réverbération s'obtient en convoluant la réponse impulsionnelle estimée précédemment avec un signal source c-a-d $y(k) = h(k) * x(k)$. L'effet de réverbération pourra donc être implémentée sous Matlab via la fonction `filter`.

Question 3.5 Programmez la fonction `effet_reverb`. Cette fonction prendra en paramètre d'entrée :

- un signal x ,

- la réponse impulsionnelle estimée précédemment.

La fonction renverra en sortie le signal x convolué par la réponse impulsionnelle estimée.

Question 3.6 Lancez le script `test_effet_reverb` pour tester votre fonction sur un son de guitare. A l'aide des commandes `tic` et `toc`, mesurez le temps de calcul nécessaire à l'exécution de la fonction `effet_reverb`.

3.1.3. Convolution rapide

Pour diminuer le temps de calcul, un étudiant propose d'implémenter l'effet de réverbération dans le domaine fréquentiel en utilisant la FFT. Il suggère d'écrire la sortie du filtre comme

$$y_2(k) = \text{IFFT}(\text{FFT}(h(k)) \cdot \text{FFT}(x(k))). \quad (2)$$

Question 3.7 Créez une fonction nommée `effet_reverb_FFT` permettant de réaliser le filtrage dans le domaine fréquentiel (utilisation des fonctions `fft` et `ifft`). Les transformées de Fourier discrète de la réponse impulsionnelle et du signal source seront calculées sur `NFFT` points, où `NFFT` correspond à la longueur maximale des signaux $h(k)$ et $x(k)$. Pour déterminer `NFFT`, il est conseillé d'utiliser les fonctions `length(.)` et `max(.)`. Notez le temps de calcul nécessaire à l'exécution de la fonction `effet_reverb_FFT` et comparez-le à celui obtenu par application de convolution directe.

Question 3.8 A votre avis, est-ce que la méthode utilisant la FFT est équivalente à la convolution classique de la section 3.1.2 ? Justifiez.

3.2. Effet de retard

Les effets de retard (delay) sont très utilisés en musique. Ces effets permettent d'ajouter de l'espace en simulant les réflexions des ondes acoustiques dans un espace clos. Dans cette prédétermination, vous allez analyser la réponse impulsionnelle et la réponse fréquentielle de l'effet de delay.

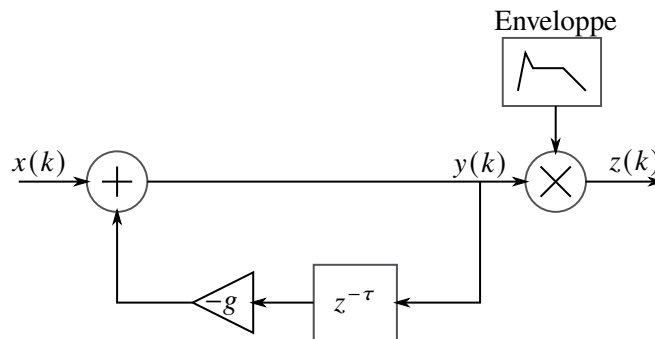


FIGURE 3 – Effet de delay (le symbole $z^{-\tau}$ désigne un retard de τ échantillons).

Pour simuler un effet de retard, nous allons considérer le système décrit par le schéma de la figure 3. Mathématiquement, cet effet peut être modélisé par la relation de récurrence suivante ($\tau \in \mathbb{N}$) :

$$y(k) = x(k) - g y(k - \tau). \quad (3)$$

Le système décrit par (3) désigne un filtre IIR puisque le système possède une partie récursive non nulle. La partie récursive permet de simuler les réflexions du son. Ainsi le signal de sortie est renvoyé avec τ échantillons de retard et un coefficient d'atténuation g . Dans toutes les questions qui suivent, le signal $x(k)$ est supposé échantillonné à $F_e = 44100\text{Hz}$.

Question 3.9 En utilisant l'équation (3) et en considérant un filtre causal, montrez que la réponse impulsionnelle théorique est égale à :

$$h(k) = \begin{cases} (-g)^n & \text{si } k = n\tau \\ 0 & \text{sinon} \end{cases} \quad (4)$$

Question 3.10 Déterminez la condition sur g pour que le filtre de la figure 3 soit stable.

Question 3.11 Pour le filtre de l'équation (3), déterminez le vecteur \mathbf{a} et le vecteur \mathbf{b} .

Question 3.12 Créez un script nommé `analyse_delay` permettant d'obtenir la réponse impulsionnelle via la fonction `filter`. Joignez la représentation temporelle de $h(k)$ à votre compte rendu.

La réponse impulsionnelle décrit le comportement du filtre dans le domaine temporel. Pour mieux comprendre le comportement du filtre, il est parfois plus simple de raisonner à partir de sa réponse fréquentielle. Cette réponse décrit la modification (module, phase) de chaque composante fréquentielle entre l'entrée et la sortie du filtre.

Question 3.13 Montrez que la réponse en fréquence du filtre de retard est

$$\hat{h}(v) = \frac{1}{1 + g e^{-2j\pi v\tau}}, \quad (5)$$

où v représente les fréquences réduites. Déterminez ensuite le module et la phase de cette réponse.

Nous pouvons alors constater que le module du filtre est maximal pour les fréquences réduites $v_k = \frac{(2k+1)}{2\tau}$ ($k \in \mathbb{Z}$) et que $|\hat{h}(v_k)| = \frac{1}{1-g}$. Notre effet est donc un filtre en peigne.

Question 3.14 Tracez sur un même graphique le module de la réponse en fréquence obtenu théoriquement et celui obtenu numériquement à l'aide de la transformée de Fourier discrète de la réponse impulsionnelle. Joignez ce graphique à votre compte rendu et expliquez les différences observées.

Nous allons à présent programmer l'effet de delay sous forme d'une fonction Matlab. Nous testerons ensuite cette fonction sur un son de piano.

Question 3.15 Programmez la fonction `effet_delay`. Cette fonction prendra comme paramètres d'entrée :

- un signal \mathbf{x} ,
- le temps du delay (en s),
- le coefficient d'amortissement g ,
- la fréquence d'échantillonnage F_e (en Hz).

La fonction renverra en sortie le signal \mathbf{x} modifié par le filtre de l'équation (3).

Nous allons tester cette fonction sur un accord de piano.

Question 3.16 Lancez le script `test_effet_delay` avec comme paramètres : $\tau = 0.25 \times F_e$ et $g = 0.9$.

À l'écoute, nous observons que cet effet produit des sonorités "synthétiques". Pour rendre l'effet plus naturel, nous allons modifier le son après réflexion en filtrant le signal dans la boucle de retour. Cette modification est illustrée par la figure 4.

Dans la boucle de retour, nous choisissons un filtre dont la réponse impulsionnelle est donnée par :

$$h_r(k) = \begin{cases} \frac{1}{K} & \text{si } k = 0, 1, \dots, K-1 \\ 0 & \text{sinon} \end{cases} \quad (6)$$

Ce filtre réalise une moyenne glissante sur K échantillons. L'effet de delay s'exprime alors par la relation de récurrence suivante :

$$y(k) = x(k) - \frac{g}{K} \sum_{n=0}^{K-1} y(k - \tau - n) \quad (7)$$

Question 3.17 Programmez la fonction `effet_delay_filtre`. Cette fonction prendra en paramètre d'entrée :

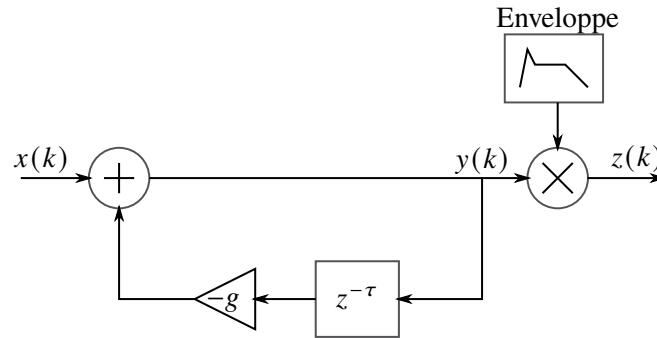


FIGURE 4 – Effet de delay ($H_r(z)$ désigne un filtre de réponse impulsionnelle $h_r(k)$).

- un signal x ,
- le temps du delay (en s),
- le coefficient d'amortissement g ,
- la longueur K ,
- la fréquence d'échantillonnage F_e (en Hz).

La fonction renverra en sortie le signal x modifié par le filtre de l'équation (7).

Question 3.18 Modifiez le script `test_effet_delay` pour tester votre fonction avec $\tau = 0.25 \times F_e$, $g = 0.9$ et $K = 10$. Sauvegardez le résultat sonore dans le fichier `piano_delay_filtre.wav`.

Question 3.19 L'écoute du résultat montre que le filtre $h_r(k)$ semble "couper" progressivement les hautes-fréquences du signal. Justifiez cette observation en traçant numériquement la réponse en fréquence de l'effet de retard filtré.

4. Pour aller plus loin : synthèse Karplus-Strong

Nous vous proposons dans cette partie d'implémenter l'algorithme de Karplus-Strong [?] dont la publication proposait une méthode de synthèse de son de type *corde pincée*. La figure 5 illustre le principe de l'algorithme KS avec une perte d'énergie. Le retard de N échantillons est directement lié à la fréquence fondamentale $f_1 = F_e/N$ que l'on souhaite obtenir ; g est un coefficient d'amortissement, α est celui qui définit le filtre de boucle. On pourra essayer la synthèse avec $g = 0.99$ et $\alpha = 0.5$, puis en les faisant varier.

La synthèse nécessite un signal d'entrée $x(k)$ que l'on peut choisir de 2 manières différentes :

- un signal aléatoire blanc gaussien (fonction matlab `randn`)
- un son d'une note jouée par un instrument (guitare,...)

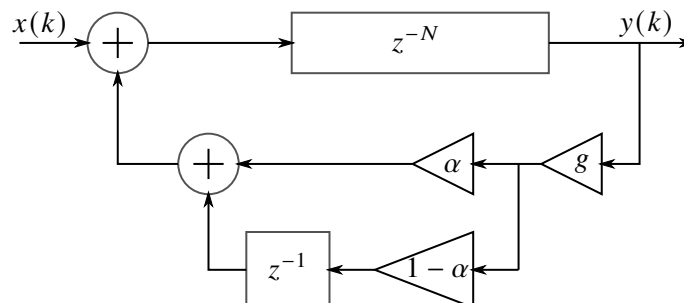


FIGURE 5 – Synthèse Karplus-Strong (modèle avec perte d'énergie) [?]