Laboratoire Bordelais de Recherche en Informatique
CNRS UMR 5800 – Université Bordeaux 1
351, cours de la Libération, 33405 Talence CEDEX, FRANCE

Mémoire de DEA

Accélération de la synthèse sonore

Mathieu LAGRANGE
lagrange@labri.fr

Juin 2001

Directeurs :
Sylvain MARCHAND, Robert STRANDH

Table des matières

Introduction

Le modèle additif est un modèle de représentation du signal sonore sous forme de partiels. Ces partiels sont des oscillateurs quasi sinusoïdaux contrôlés en fréquence et en amplitude. La synthèse additive consiste à générer chaque son grâce à la superposition d'un grand nombre de sinusoïdes. Or le calcul de la fonction sinus est un processus coûteux. En réponse, des algorithmes très efficaces ont été proposés [1, 2], basés sur une description récursive de la fonction sinus, ce qui permet de générer chaque sinus avec seulement une addition et une multiplication. Ceci est optimal en terme de complexité.

Le seuil d'optimalité étant atteint, le seul moyen de réduire le temps consacré à la synthèse est de réduire la taille du problème, c'est-à-dire le nombre de partiels à synthétiser. Ceci est possible car on constate la présence d'informations non pertinentes (imperceptibles pour l'oreille humaine) quand le nombre de partiels augmente. La détection de ces informations non pertinentes est issue de résultats de recherches menés en psycho-acoustique sur la non linéarité du système auditif humain [4]. Ces résultats, qui seront le sujet de la première partie (sections 1 et 2), sont déjà exploités par le projet MPEG II niveau 3 (MP3) pour préserver de l'espace mémoire. A contrario, l'objectif de ce mémoire est de préserver du temps CPU en ne synthétisant que les partiels audibles, le challenge étant alors de détecter ces partiels inaudibles en un intervalle de temps sensiblement plus court que celui de la synthèse effective de ces partiels. La seconde partie (sections 3 et 4) est consacrée aux différents algorithmes mis en œuvre exploitant les résultats développés dans la première partie. Après évaluation des performances (section 5), une dernière partie (section 6) présente une direction de recherche pour améliorer cette première mise en œuvre.

1 Le son

1.1 Généralités

Le mot *son* désigne à la fois une sensation auditive et le phénomène physique susceptible de lui donner naissance. Un son est un être à double face. Face physique, c'est un ébranlement, une perturbation dans un milieu matériel élastique. Et face perceptive, c'est un signal perçu par le sens de l'ouïe. Même si notre langage permet de distinguer certains sons des autres, il est difficile d'établir une classification solide (basée sur des critères physiques). Les sons harmoniques issus d'instruments non percussifs (comme la voix, les instruments à cordes et à vent) nous intéresseront particulièrement, car ils ont la propriété d'être une somme d'oscillations sinusoïdales régulières. En effet, grâce à la transformée de Fourier, on peut décomposer un signal complexe variant en amplitude en fonction du temps en une somme de sinusoïdes ayant chacune une fréquence et une amplitude variant dans le temps (fig. 1). Par la suite, nous appellerons partiels ces sinusoïdes.

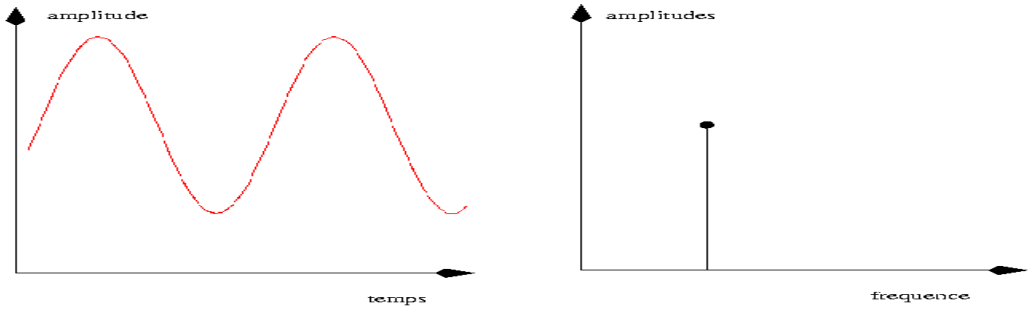


FIG. 1 – Deux représentations : temporelle et fréquentielle.

Cette représentation a l'avantage d'être plus synthétique et plus proche de l'oreille. En effet, les deux signaux de la figure 2 sont somme des deux mêmes sinusoïdes avec des différences de phases. La représentation dans le domaine des fréquences est donc la même, car cette représentation ne prend pas en compte les phases. Or c'est précisément ce que fait l'oreille car ces deux signaux ne sont pas discernables.

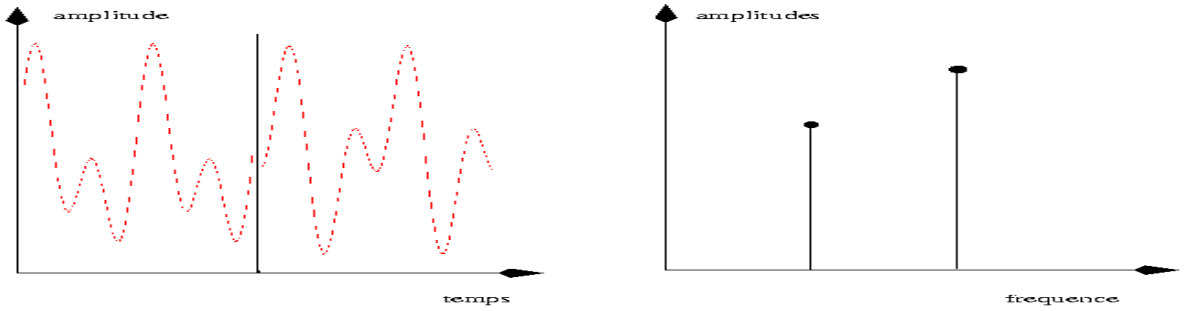


FIG. 2 – Deux signaux et leur représentation sous forme de partiels.

1.2 Modèle sinusoïdal

1.2.1 Propriétés

Un son n'est donc plus pour nous un signal échantillonné, mais une superposition de sinusoïdes dont les fréquences et les amplitudes varient lentement dans le temps. On peut dès lors dégager deux propriétés sur ces partiels.

Propriété 1 *Sur l'échelle des fréquences, deux partiels ne doivent pas être trop proches.*

En effet, si deux sinusoïdes ont la même fréquence, leur rapport de phases entre en jeu. Si les deux sinusoïdes sont en phases, le signal résultant aura une amplitude qui sera la somme des amplitudes des deux partiels. Par contre, si les phases sont opposées, le signal résultant sera la soustraction des deux signaux (fig. 3). Nous sommes donc en présence de phénomènes indésirables et imprédictibles que nous négligerons par la suite en considérant que deux partiels sont toujours suffisamment éloignés.

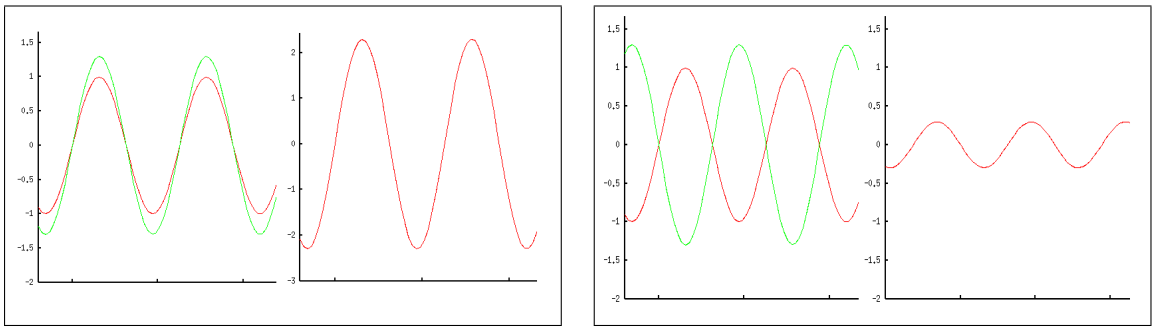


FIG. 3 – Deux signaux en phase et deux signaux en opposition de phase.

Propriété 2 *Le nombre total de partiels, n, est fini.*

Mais ce nombre peut être très grand en pratique. Le but de ce mémoire est justement d'essayer de majorer le nombre de partiels à synthétiser en ne retenant que les partiels significatifs, c'est-à-dire effectivement perçus par l'auditeur.

Note :

L'oscillation associée à un partiel est un signal d'amplitude linéaire comprise entre 0 et 1. Or l'oreille perçoit les variations d'amplitudes de manière logarithmique. On utilisera donc l'échelle bien connue des décibels. Dans la suite, on considérera que le maximum exprimable (1 en amplitude linéaire) correspond à 0 décibels.

1.2.2 Synthèse additive

La synthèse de ces partiels est une synthèse additive. Elle consiste, pour chaque partiel p, à calculer son oscillation sinusoïdale associée et à faire la somme de toutes ces oscillations.

$$s(t) = \sum_{p=0}^{n} a_p(t) \cdot \cos(2\pi \cdot f_p(t)) \tag{1}$$

Le problème est donc de pouvoir calculer simultanément un grand nombre de sinusoïdes. Pour ce faire, des algorithmes très efficaces ont été proposés [1, 2], basés sur une description récursive de la fonction sinus. Avec cette approche, il est possible de générer chaque sinus avec seulement une addition et une multiplication pour chaque partiel.

1.2.3 Limitations

Le modèle sinusoïdal permet des manipulations musicales sur les sons. La chose la plus importante étant la qualité du résultat, on peut admettre que ces manipulations ne se fassent pas en temps réel. Par contre, la synthèse sonore (c'est-à-dire permettre à l'utilisateur d'écouter le résultat de ces manipulations) doit, si possible, pouvoir s'éxécuter en temps réel.

Nous savons qu'un son monophonique comporte quelques dizaines, voire centaines de partiels qu'il faut synthétiser 44100 fois par seconde (en qualité CD). Même si, grâce aux recherches menées, la synthèse d'une sinusoïde peut se réduire à une multiplication et une addition, la synthèse complète d'un chœur de vingt voix en temps réel, par exemple, est très coûteuse pour les machines actuelles. Or la synthèse de certains partiels n'est pas pertinente car ils sont inaudibles pour l'oreille humaine. C'est grâce aux résultats de la psycho-acoustique présentés dans la partie suivante que nous pouvons les détecter.

2 Non linéarité de l'oreille humaine

De façon subjective, le son est une sensation qui, traitée par l'oreille, donne au cerveau des informations sur le monde extérieur. En cela, l'étude de quelques éléments de psycho-acoustique nous permet de mieux comprendre les effets non linéaires que nous exploiterons par la suite. Dans une première partie, nous étudierons le seuil d'audibilité de l'oreille humaine et, dans une seconde, nous parlerons du phénomène de masquage et de ses propriétés.

2.1 Seuil d'audibilité

L'oreille humaine est diversement sensible à des signaux de fréquences différentes. Il existe en effet pour chaque fréquence un seuil absolu d'audibilité en dessous duquel le signal n'est pas perçu. Formellement, ce seuil est caractérisé par le degré d'énergie nécessaire pour générer un son tonal d'une fréquence donnée qui puisse être détecté par un auditeur dans un environnement silencieux [3]. Ce seuil de silence (représentatif de l'audition d'un homme jeune) est bien approximé par la fonction non linéaire de l'équation 2.

$$\mathcal{S}(f) = 3.64(f/1000)^{-0.8} - 6.5e^{-0.6(f/1000-3.3)^2} + 10^{-3}(f/1000)^4 \quad \text{(dB SPL)} \qquad (2)$$

Note :

On a défini le maximum exprimable (1 en amplitude linéaire) à 0 dB. Ce n'est pas la référence prise pour l'équation 2 qui est exprimée en dB SPL (Sound Pressure Level), échelle dans laquelle le maximum exprimable correspond à 120 dB. Il suffit donc de faire une translation en abscisse de -120 dB pour faire correspondre l'équation 2 à la figure 4.

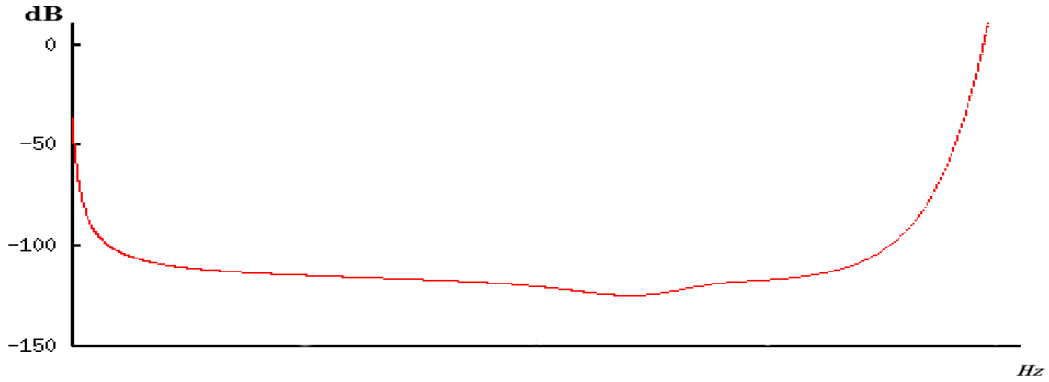


FIG. 4 – Le seuil d'audibilité.

Bien entendu, pour le problème qui nous intéresse, tout partiel ayant une amplitude inférieure à ce seuil n'a pas besoin d'être synthétisé puisqu'il est inaudible.

Notation :

La sélection par le seuil d'audibilité se faisant par rapport à l'amplitude du partiel, on appellera cette sélection la **sélection en amplitude**.

2.2 Le masquage

Nous venons de voir qu'un signal doit être suffisamment puissant pour passer à travers le tympan et exciter la cochlée qui est située dans l'oreille interne. Dans cette seconde partie, nous verrons comment l'oreille effectue son analyse spectrale ce qui nous permettra ensuite d'étudier deux types de masquage : fréquentiel et temporel.

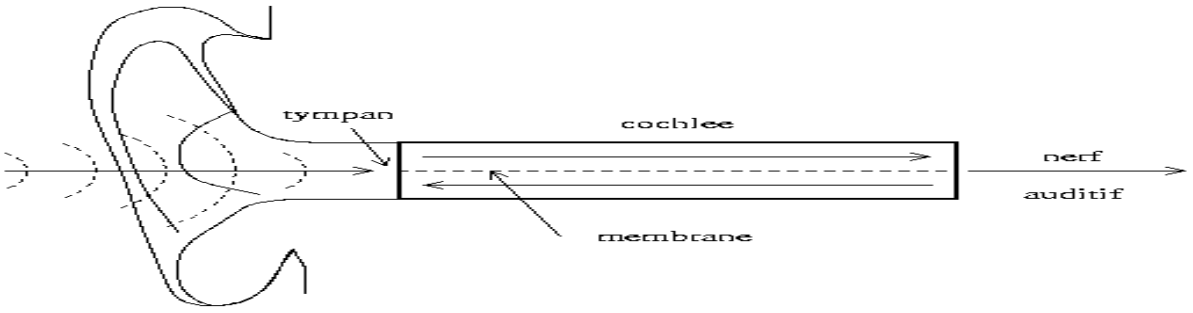


FIG. 5 – Le système auditif humain.

Le long de la cochlée se trouve la membrane basilaire, sur laquelle on peut distinguer des régions constituées d'un ensemble de récepteurs neuronaux réagissant à certaines bandes de fréquences. Le signal sonore tonal (composé d'un seul partiel) voyageant dans

la cochlée déforme la membrane basilaire. Le maximum de cette déformation est situé précisément là où se trouve le groupe de neurones réagissant à la fréquence du partiel.

2.2.1 Le masquage fréquentiel

Dès lors, si un partiel d'amplitude élevée excite ce groupe de neurones, des partiels de fréquence proche et d'amplitude plus faible ne seront même pas détectés par ce groupe. On peut illustrer ce phénomène de masquage en fréquence par l'exemple suivant :
On peut discuter tranquillement (fréquences moyennes), pendant qu'une multitude d'oiseaux s'égosillent dans le jardin (fréquences élevées). On est par contre obligé de hausser la voix si on discute dans un lieu où tout le monde parle en même temps (fréquences semblables). Ce cloisonnement en fréquence a bien sûr ses limites : si un avion décolle, l'amplitude est tellement élevée que tout le spectre s'en trouve masqué.
Ce phénomène de masquage fréquentiel est particulièrement intéressant pour notre problème car il apparaît précisément lorsque la synthèse en temps réel devient impossible à cause du trop grand nombre de partiels. En théorie, on peut penser que plus on augmente le nombre de partiels, plus le rapport entre le nombre de partiels significatifs (non masqués) et le nombre total de partiels diminue. Si on pouvait déterminer rapidement quels sont les partiels significatifs, on pourrait étendre le nombre de partiels sans problème puisque le nombre de partiels à synthétiser resterait sensiblement le même.

2.2.2 Le masquage temporel

Un signal de forte amplitude va en quelque sorte "traumatiser" le groupe de neurones si bien que même si ce signal s'est arrêté, le groupe n'est pas encore prêt à recevoir de nouvelles informations. On appelle ce phénomène le post-masquage. Le plus surprenant est qu'il existe aussi un pré-masquage qui entre en jeu lors de la détection de sons percussifs, mais son influence est trop courte pour être utilisable dans le cadre de cette étude.

2.3 Modélisation du masquage fréquentiel

Comme nous l'avons dit, le masquage fréquentiel est un phénomène très important pour le but que nous nous sommes fixé. Il reste à le modéliser pour pouvoir l'exploiter de façon efficace.

Définition 1 *Un partiel est dit* ***masqué*** *si il est rendu inaudible par la proximité en fréquence d'un autre partiel d'amplitude plus élevé.*

Définition 2 *Un partiel est dit* ***masquant*** *si il n'est pas masqué et si il masque d'autres partiels.*

Définition 3 *On appellera* ***cône de masquage*** *du partiel p la limite supérieure de la zone où se situent les partiels masqués par p.*

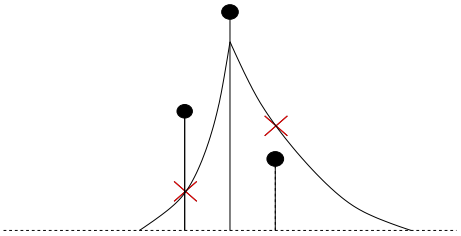


FIG. 6 – Représentation du cône de masquage sur une échelle linéaire de fréquence.

La figure 6 représente le cône de masquage généré par un partiel sur une échelle linéaire de fréquence. Bien entendu, pour déterminer si un partiel est masqué, il faut calculer l'intersection entre une exponentielle (provenant du cône de masquage) et un segment de droite (provenant du partiel).

De même que l'on a introduit les décibels qui sont une mesure logarithmique de l'amplitude, introduisons ici l'échelle Bark (de Barkhausen) qui est une échelle quasi logarithmique de la fréquence. Dans ce nouveau repère, le masque généré par un partiel est proche d'un triangle, bien que ce ne soit pas vraiment le cas en ce qui concerne le sommet du triangle. Par bonheur, cette singularité peut être négligée grâce à une limitation de notre modèle (prop. 1).

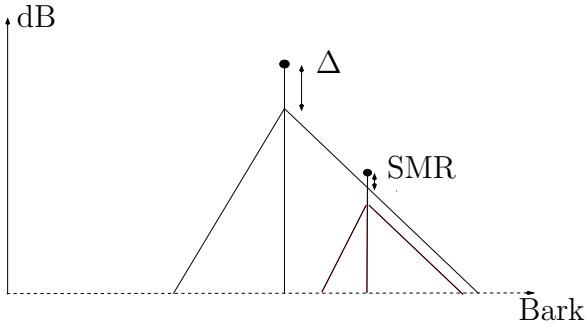


FIG. 7 – Représentation du masque avec l'échelle Bark.

Les caractéristiques de ce triangle utilisées par Garcia et Pampin [5] sont les suivantes :

- La différence Δ entre le volume du partiel et le seuil du masque est de -10 dB.
- La courbe de masquage vers les basses fréquences est de 27 dB/Bark.
- La courbe de masquage vers les hautes fréquences est de -15 dB/Bark.

On voit sur la figure 7 que pour décider si un partiel est masqué dans un repère doté des échelles décibel et Bark, il suffit de faire une intersection de droites. Après avoir défini dans quel repère notre modèle se situe, nous allons en présenter quelques propriétés.

Définition 4 *On appelle* ***Signal to Mask Ratio*** *(SMR) la différence entre l'amplitude a_p d'un partiel p et la valeur du masque à la fréquence f_p (fig. 7).*

Note :

Dans notre implémentation, la valeur du SMR est utilisée comme un booléen. Si un partiel a son SMR inférieur à 0, il est masqué donc on ne le synthétise pas. Dans les algorithmes de compression de type MPEG II niveau 3, plus le SMR est faible, moins on alloue de bits pour coder les différents champs de données. Pour simplifier, moins un partiel est perceptible, moins on a besoin de le coder avec précision. De même, en synthétisant les partiels avec une approximation de la sinusoïde par des polynômes, une façon de "dégrader" un partiel devenu presque inaudible serait de diminuer le degré du polynôme associé. On aurait alors une efficacité accrue car on pourrait progressivement dégrader la sinusoïde attachée à un partiel avant même que le partiel ne soit masqué.

Propriété 3 *Seul un partiel ayant un SMR égal à 10 dB contribue au masque (fig. 8).*

Définition 5 *On appelle* ***contribution*** *d'un partiel masquant l'élévation du masque due à l'inclusion de ce partiel dans le masque (fig. 8).*

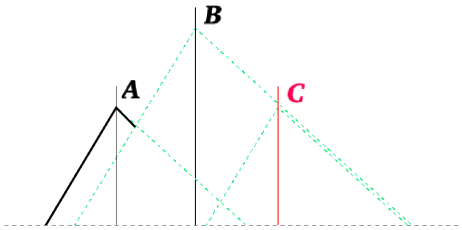


FIG. 8 – En gras, la contribution de A. Le partiel C n'est pas masqué mais ne contribue pas au masque.

Notation :

On notera $>_m$ la relation de masquage.

Ainsi, dans la figure 9, B $>_m$ A, car A est dans le cône de masquage de B.

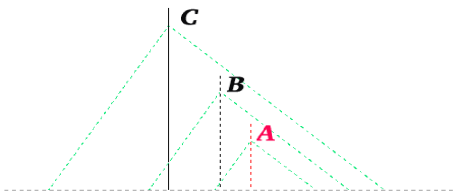


FIG. 9 – Illustration de la propriété de transitivité.

Propriété 4 (de transitivité) *Si un partiel A est dans le cône de masquage du partiel B et que B est masqué par un partiel C, alors A est dans le cône de masquage de C.*

$$B >_m A \wedge C >_m B \Rightarrow C >_m A$$

Notation :

Même si l'amplitude des partiels entre en jeu (les partiels masqués doivent avoir une amplitude au moins inférieure de 10 dB à celle du partiel masquant), la propriété la plus importante est la proximité en fréquence comme le montre l'exemple développé dans la section 2.2.1. On appellera donc **sélection en fréquence** le fait d'isoler les partiels masquants des partiels masqués.

3 Première mise en œuvre

Maintenant que nous avons présenté les différentes propriétés qui nous seront utiles, nous allons nous intéresser à la mise en œuvre logicielle.
Dans une première partie, nous verrons un algorithme naïf de sélection en fréquence qui constituera une référence pour nos résultats ultérieurs. Dans une seconde, nous verrons comment minimiser la taille du problème grâce à la sélection en amplitude par le seuil d'audibilité. Et enfin, dans une troisième partie, nous verrons comment réduire la complexité de l'algorithme de sélection en fréquence.

Notation :

On note :

- n le nombre de partiels à synthétiser ;
- n_A le nombre de partiels ayant leur amplitude supérieure au seuil d'audibilité ;
- n_M le nombre de partiels masqués ;
- n_C le nombre de partiels contribuant au masque ;
- n_S le nombre de partiels effectivement synthétisés.

3.1 Algorithme naïf

L'algorithme le plus simple pour décider du masquage d'un partiel par un autre est en complexité quadratique. En effet, pour chaque partiel, on balaie tout l'ensemble des partiels pour déterminer si au moins un partiel le masque.

Algorithme :

```
1: Pour i de 1 à n faire
2:   Pour j de 1 à n faire
3:     Si (Partiel[i] <_m Partiel[j]) alors
4:       Partiel[i] est masqué ;
5:       terminé ;
6:     fin Si
7:   fin Pour
8:   Partiel[i] n'est pas masqué ;
9: fin Pour
```

3.2 Sélection en amplitude

Une première solution pour accélérer le masquage fréquentiel consiste à réduire la taille du problème. En effet, si un partiel est inaudible (en dessous du seuil d'audibilité), peu importe qu'il soit masqué ou non. On procède donc à une sélection en amplitude d'une complexité linéaire, qui consiste à décider si l'amplitude a_p d'un partiel p est supérieure ou non au seuil d'audibilité à la fréquence f_p.
Comme $\mathcal{S}(x)$ – la fonction non linéaire approximant le seuil d'audibilité (eq. 2) – est une fonction complexe à calculer, on l'échantillonne en fréquence. Ainsi, on calcule les différentes valeurs une seule fois pour un éventail suffisamment large de façon à obtenir une bonne précision, puis on procède ensuite par comparaison et interpolation.

Si la fréquence du partiel f_p est comprise entre f_i et f_{i+1}, alors si a_p est inférieure au minimum entre $\mathcal{S}(f_i)$ et $\mathcal{S}(f_{i+1})$ ou supérieure au maximum de ces deux valeurs, alors l'erreur induite par l'échantillonnage est nulle, c'est-à-dire que le choix effectif est bien le même que celui que l'on aurait fait en comparant effectivement $\mathcal{S}(f_p)$ et a_p.
Par contre, si a_p est située dans la zone d'incertitude (fig. 10), alors on procède par interpolation pour minimiser le risque d'erreur. On considère ainsi que la courbe se réduit à un segment de droite reliant $\mathcal{S}(f_i)$ et $\mathcal{S}(f_{i+1})$.

Complexité :

> Pour chaque partiel, on effectue un nombre constant d'opérations, la complexité est donc en $\mathcal{O}(n)$.

Après avoir réduit autant que possible la taille du problème, l'approche la plus importante consiste à revoir l'algorithme de sélection en fréquence pour en diminuer sa complexité. Dans la partie suivante, nous argumenterons la nécessité d'un tri en amplitude préalable.

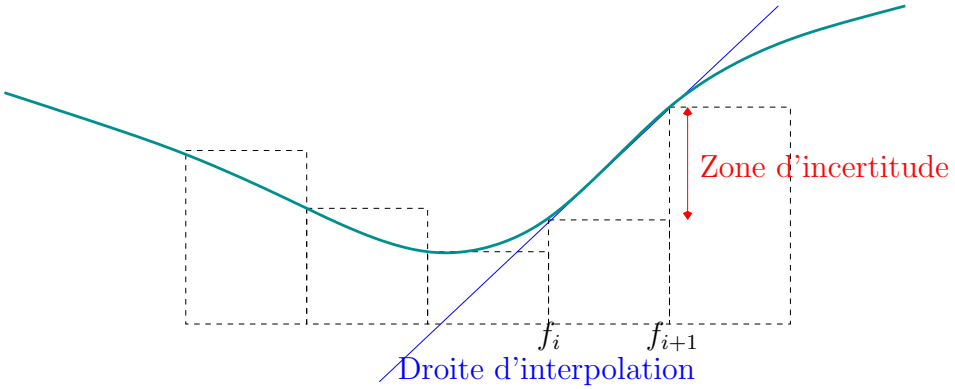


FIG. 10 – L'interpolation.

3.3 Le tri en amplitude

Le tri en amplitude des partiels est un préliminaire qui nous a paru indispensable pour décider du masquage d'un partiel. En effet, dans le cas de partiels non triés, on ne pourra obtenir la forme du masque qu'après un premier parcours pour collecter toutes les contributions des partiels masquants, puis un autre parcours sera nécessaire pour déterminer si un partiel est masqué ou non. De plus, dans le cas le pire où les partiels sont triés en ordre inverse, lors du premier parcours, il faudra remettre à jour le masque pour chaque partiel, car on sera en présence d'un nouveau maximum. Ces mises à jour sont un surcoût en temps qui nous a semblé inacceptable. Par contre, si les partiels sont triés en ordre décroissant d'amplitude, les premiers partiels rencontrés seront ceux qui contribueront au masque. Et si un partiel n'est pas masqué, on est sûr qu'il ne le sera jamais car tous les partiels qui le suivent seront d'amplitude inférieure. Pour résumer, grâce au tri en amplitude, on construit de façon optimale (avec un minimum de modifications) le masque et on décide à la première passe du masquage de tous les partiels.

Complexité :

La complexité du "quick sort" utilisé ici est en $\mathcal{O}(n \cdot \log(n))$.

3.4 Algorithme d'échantillonnage du masque

Cette première solution de sélection en fréquence utilise la même technique que celles utilisées lors de la sélection en amplitude à ceci près que le masque doit être construit dynamiquement lors de chaque synthèse. On garde en quelque sorte une "silhouette" du masque que l'on met à jour si besoin est. Cet algorithme a été proposé par Marchand [1]. Au début, on initialise la silhouette à zéro puis, si un partiel contribue au masque, on inclue sa contribution à la silhouette.

Pour déterminer si un partiel est masqué, on procède aussi par interpolation (fig. 10). Dans ce cas, l'erreur induite par l'échantillonnage est nulle car la silhouette est uniquement composée de segments de droites, sauf dans le cas du sommet du cône de masquage où on est en présence d'une ligne brisée (fig. 11). Ce cas est négligeable, car à cause d'une limitation de notre modèle (prop. 1), les partiels ne peuvent pas être trop proches. Il suffit donc de choisir un nombre d'échantillons suffisant pour que les partiels soient séparés d'au moins un échantillon.

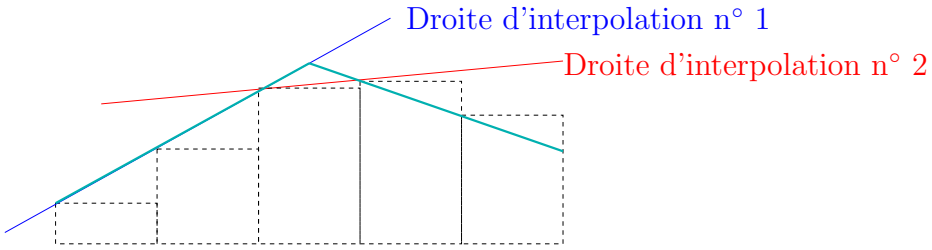


FIG. 11 – Si on interpole une pente (1), l'erreur est nulle. Si on interpole un sommet (2), on fait une erreur.

On remarque sur la figure 12 que la partie effectivement mise à jour peut être très petite en regard du nombre de tests effectués. Or ce cas est particulièrement fréquent lorsque l'on est en présence de sons harmoniques. Les partiels sont espacés régulièrement en fréquence et d'amplitudes assez semblables. La seule façon d'améliorer cet algorithme est donc de supprimer les tests inutiles. Pour cela, on sépare la mise à jour en deux parties, toutes les deux descendantes. On part de la hauteur maximale du cône de masquage à insérer et la deuxième fois que l'on ne met pas à jour la silhouette, on peut arrêter la série de tests, car on sait que l'on restera en dessous de la silhouette du masque. Ceci découle de la propriété de transitivité (prop. 4), les segments formant les cônes de masquages étant parallèles, si une partie de ce segment est masqué par un autre cône, il le sera jusqu'au niveau d'amplitude le plus bas.

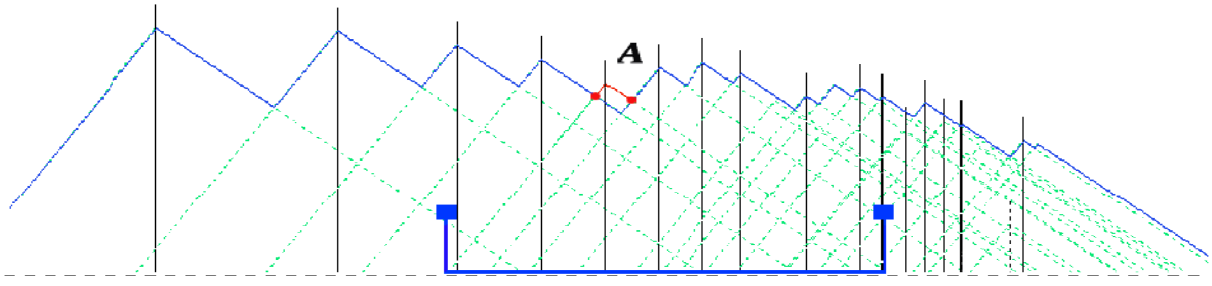


FIG. 12 – De extrême_gauche à extrême_droit entre carrés, la zone effectivement mise à jour entre ronds.

Algorithme :

```
1: limite ⟵ faux ;
2: Si (non masqué) alors
3:   Si (contribue au masque) alors
4:     Pour (i de extrémité gauche du cône à fréquence du partiel) faire
5:       Si (hauteur du cône[i] > silhouette[i]) alors
6:         silhouette[i] ⟵ hauteur du cône[i] ;
7:       sinon
8:         Si (limite) alors
9:           terminé ;
10:        fin Si
11:        limite ⟵ vrai ;
12:      fin Si
13:    fin Pour
14:    limite ⟵ faux ;
15:    Pour (i de fréquence du partiel à extrémité droite du cône) faire
16:      Si (hauteur du cône[i] > silhouette[i]) alors
17:        silhouette[i] ⟵ hauteur du cône[i] ;
18:      sinon
19:        Si (limite) alors
20:          terminé ;
21:        fin Si
22:        limite ⟵ vrai ;
23:      fin Si
24:    fin Pour
25:  fin Si
26: sinon
27:  Le partiel est masqué ;
28: fin Si
```

Notation :

On notera n_{ech} le nombre d'échantillons de la silhouette.

Complexité :

> On doit d'abord effectuer le tri en amplitude. Ensuite, il faut mettre à zéro tous les échantillons et enfin les mettre à jour en fonction de chaque partiel contribuant au masque. La complexité de l'algorithme par échantillonnage est donc en $\mathcal{O}(n_A \cdot \log(n_A) + n_C \cdot n_{\text{ech}})$.

Dans cette troisième partie, nous avons étudié une solution par échantillonnage du masque qui constitue une approche discrète du problème. Un des problèmes liés à ce type d'approche est le choix de la valeur de n_{ech} qui doit être un compromis entre erreur d'approximation et temps de mise à jour prohibitif. Dans la partie suivante, nous parlerons d'une nouvelle approche (continue) qui contourne ce problème, à savoir la construction du masque par stockage des contributions de chaque partiel.

4 Construction rapide du masque

Le but de ce nouvel algorithme est de stocker, pour chaque partiels apportant une contribution au masque, son "champ d'influence".

4.1 Présentation générale

Définition 6 *Le **champ d'influence** d'un partiel masquant est la bande de fréquence dans laquelle il contribue au masque.*

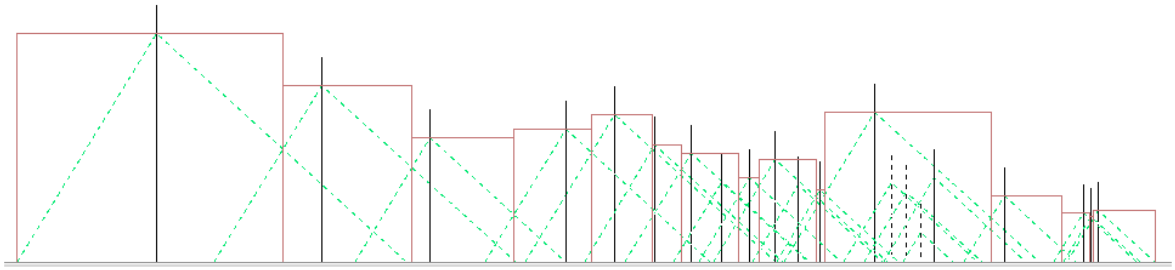


FIG. 13 – Représentation en rectangles.

Comme on le voit sur la figure 13, pour calculer ce champ d'influence, on doit calculer des intersections entre deux cônes voisins. Il est donc nécessaire que la liste soit ordonnée en fréquence. Chaque nœud de la liste doit contenir :

- la fréquence centrale du cône ;
- la hauteur du cône ;
- les valeurs courantes de début et de fin du champ d'influence ;

Et, pour des raisons que nous détaillerons par la suite :

- les valeurs initiales (cas où le cône est unique) de début et de fin du champ d'influence.

On peut alors distinguer trois cas d'insertion :

- Les voisins sont suffisamment éloignés (fig. 14.1). Les valeurs courantes sont égales aux valeurs initiales ;
- Un des voisins est suffisamment proche (fig. 14.2). On doit calculer l'intersection des deux cônes et mettre à jour la fin ou le début du voisin selon qu'il est à gauche ou à droite ;
- Les deux voisins s'intersectaient déjà (fig. 14.3). C'est dans ce cas qu'interviennent les valeurs initiales car ce sont ces valeurs qui doivent être utilisées pour le calcul des deux intersections.

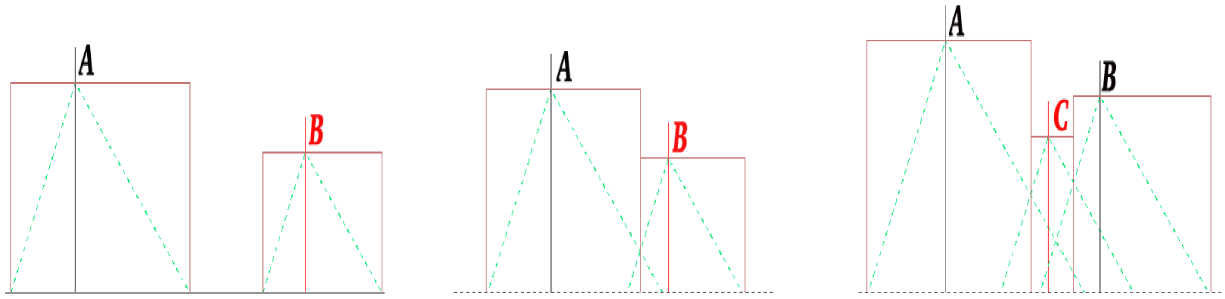


FIG. 14 – Les trois cas d'insertion.

A la lumière de cette énumération, on remarque que l'insertion est simple. Au pire, on a besoin de mettre à jour deux nœuds de la liste, les voisins de gauche et de droite. Ceci est une conséquence du tri en amplitude et de la propriété de transitivité (prop. 4), comme cela est expliqué dans les exemples suivants.

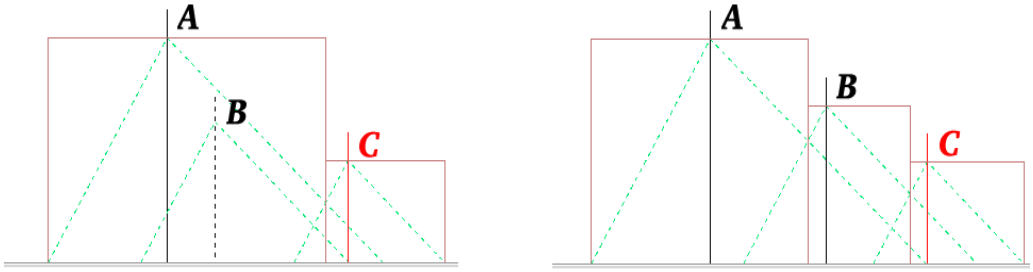


FIG. 15 – Illustration des conséquences du tri préalable.

Dans la figure 15.1, le cône B n'est même pas présent dans la liste car A est arrivé en premier et masque B. Dans la figure 15.2, même si le cône C s'intersecte avec A, comme C est nécessairement plus petit que B, il ne peut modifier la taille du champ d'influence de A.
Avec une telle structure de données, décider si un partiel est masqué revient à tenter de l'insérer. On commence donc par déterminer ses deux voisins, si son SMR est supérieur à 10 dB alors on procède à l'insertion et si son SMR est inférieur à zéro, alors il est masqué.

Algorithme :

1: gauche ⟵ voisin gauche de nouveau ;
2: droit ⟵ voisin droit de nouveau ;
3: limite ⟵ faux ;
4: **Si** (gauche $>_m$ nouveau) **alors**
5: nouveau est masqué ;
6: terminé ;
7: **sinon Si** (nouveau.volume - gauche.amplitude_cone(nouveau.fréquence) $<$ 10) **alors**
8: limite ⟵vrai ;
9: **fin Si**
10: **Si** (droit $>_m$ nouveau) **alors**
11: nouveau est masqué ;
12: terminé ;
13: **sinon Si** (nouveau.volume - droit.amplitude_cone(nouveau.fréquence) $<$ 10) **alors**
14: terminé ;
15: **fin Si**
16: **Si** (limite) **alors**
17: terminé ;
18: **fin Si**
19: Procéder à l'insertion ;
20: Mettre à jour si besoin est les champs des deux voisins ;

Comme nous l'avons vu, maintenir à jour le masque revient à faire des insertions dans une liste triée en fréquence. Or la recherche et l'insertion dans une liste triée classique se font en temps linéaire. Dans la partie suivante nous développerons l'alternative que nous avons adoptée pour rendre cette complexité logarithmique.

4.2 La Skip List

Pour réduire la complexité de recherche dans une liste triée, une solution consiste à avoir recours à une structure arborescente pour obtenir une complexité logarithmique. On a alors une structure en arbre binaire qui permet une recherche très rapide, mais du fait de sa structure très rigide à maintenir, les insertions et suppressions sont laborieuses. Typiquement, ce type d'arbre est utile pour maintenir une base de données très souvent consultée et mise à jour ponctuellement. Or, dans notre cas, les recherches et les insertions sont dans le même ordre d'apparition.

4.2.1 Présentation

Pugh propose une alternative aux arbres binaires : la Skip List [7]. Comme son nom l'indique, c'est une liste simplement chaînée qui peut donc être manipulée comme telle, mais qui possède aussi une arborescence de pointeurs qui permet de "passer par dessus" - skip - des nœuds non pertinents comme on le voit sur la figure 16.

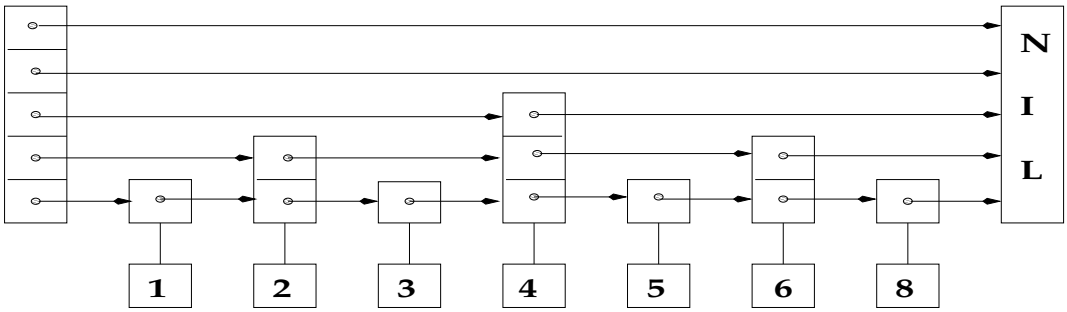


Fig. 16 – Une Skip List idéale.

Voici comment se passe une recherche du nœud 6 (fig. 17). On part du haut de la tête de liste, le nœud pointé contient NIL qui est supérieur à tout les éléments de la liste, on descend alors d'un cran dans le tableau de pointeurs. Le nœud pointé contient 2 qui est inférieur à 6, on se déplace donc. Maintenant, le nœud pointé contient NIL qui est supérieur, on descend alors d'un cran dans le tableau de pointeurs et on atteint ainsi le nœud contenant la clé 6, la recherche est alors terminée.

On peut remarquer sur la figure 16 qu'une Skip List idéale a une structure arborescente semblable à celle d'un arbre binaire équilibré. Le problème se pose lors de l'insertion de 7 par exemple. Pour maintenir cet équilibre parfait (celui d'un arbre binaire équilibré), il va falloir mettre à jour toutes les hauteurs de tableaux de pointeurs, déterminer la hauteur du nouveau nœud en fonction de la valeur de la clé par rapport à celles contenues dans la liste, et effectuer les mises à jour des pointeurs. Pugh propose de conserver les mêmes hauteurs de tableaux des anciens nœuds et de tirer au hasard la hauteur du tableau de pointeurs du nouveau nœud. Une bonne façon de se convaincre de l'efficacité d'un tel procédé est

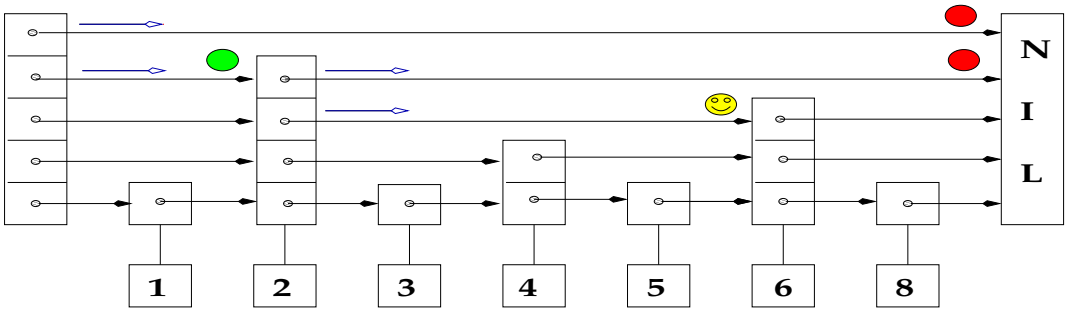


FIG. 17 – Recherche du noeud 6 dans une Skip List.

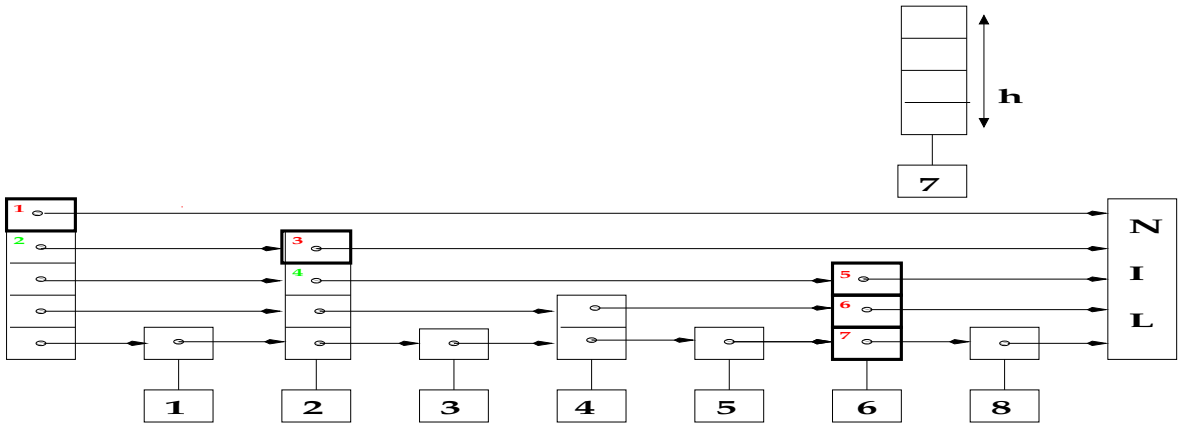
de raisonner en terme de probabilités. En effet, la probabilité de faire un parcours de liste complet lors de la recherche du plus grand élément décroît exponentiellement quand on augmente le nombre de nœuds.

Dès lors, l'insertion est un processus simple, qui consiste d'abord à rechercher l'endroit où insérer le nouvel élément. Pendant cette recherche, on stocke tous les pointeurs accédés qui pointent vers un nœud supérieur à la valeur du nœud que l'on veut insérer (encadrés en gras sur la figure 18). Une fois cette recherche effectuée, on procède comme suit :

a • on tire au hasard une hauteur h de tableau de pointeurs pour le nouveau nœud (fig. 18.1) ;

b • on recopie dans le tableau du nouvel élément tous les pointeurs de hauteur inférieure à h précédemment stockés (fig. 18.2) ;

c • puis on actualise tout ces pointeurs avec l'adresse du nouvel élément (fig. 18.3).



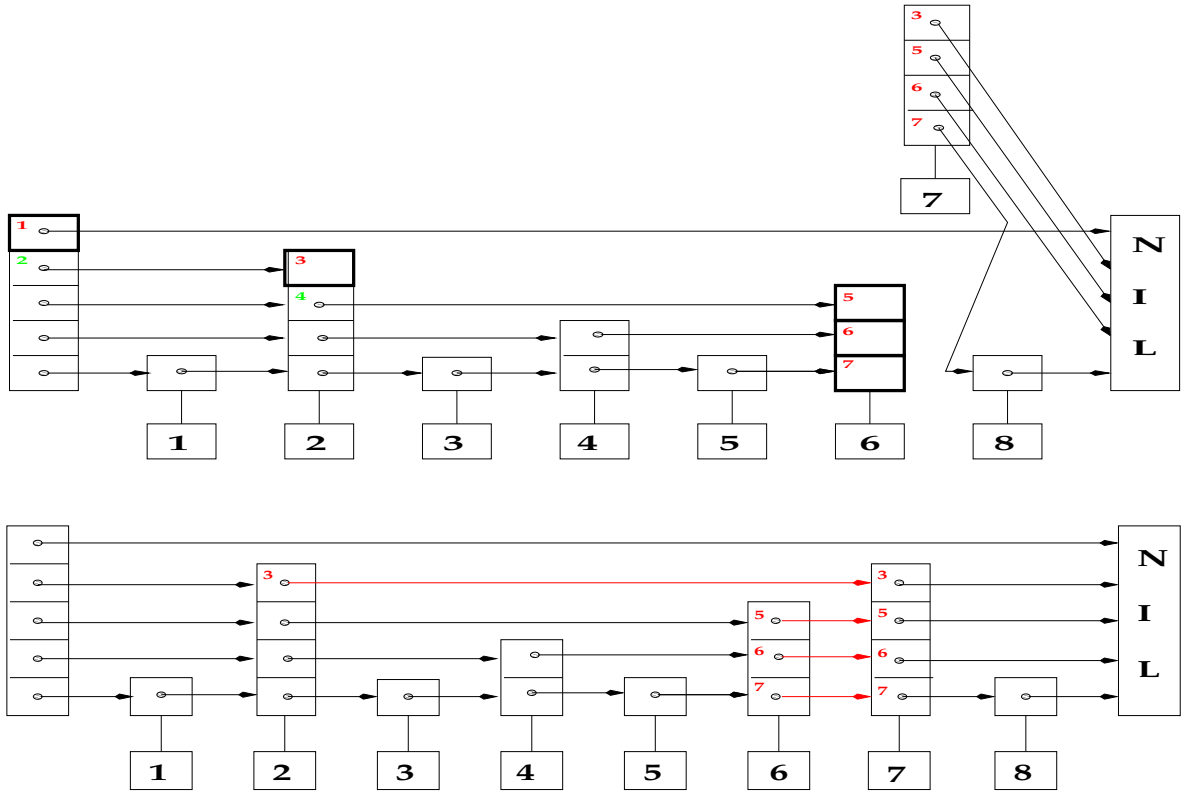


FIG. 18 – Les trois phases de l'insertion.

On peut remarquer que, en faisant varier le taux de probabilité p d'avoir la hauteur maximale permise lors du tirage de la hauteur h, on fait varier la hauteur de l'arborescence. Par exemple, si p est faible, l'arborescence sera très écrasée, donc la taille mémoire sera réduite et le temps d'accès plus long. Par contre, si p est élevé, c'est l'inverse qui se produit (fig. 19). Pugh conseille des valeurs de p comprise entre $\frac{1}{4}$ et $\frac{1}{2}$. Dans notre cas, à savoir 1000 éléments au maximum, les variations de temps d'accès entre ces deux valeurs ne sont pas très sensibles.

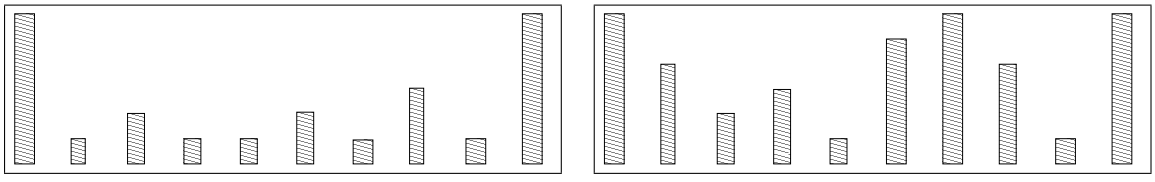


FIG. 19 – Tirage des hauteurs avec p faible, puis élevé.

Dans notre algorithme de stockage des contributions, pour procéder aux mises à jour suivant l'insertion, nous avons besoin du voisin gauche et du voisin droit du nouvel

élément. Après la recherche précédant l'insertion effective, on dispose de l'adresse du voisin droit dont la fréquence est strictement supérieure à celle du nouvel élément (pointeur 7 du nœud 6 dans la figure 18). Il est par contre impossible de trouver le voisin gauche car l'arborescence est unidirectionnelle, il a donc fallu lui rajouter un chaînage arrière.

4.2.2 Allocation mémoire

Dans la version générique utilisée, lors de l'insertion, on alloue un bloc mémoire de la taille du nœud et de son tableau de pointeurs associé à chaque fois que l'on veut insérer une nouvelle contribution. La taille du tableau étant tirée au hasard, la taille du bloc mémoire est très rarement une puissance de 2. On en déduit que, lors de la libération, des opérations très coûteuses de retassement auront lieu. Alors que, lors de la sélection en fréquence, on construit une liste que l'on exploite puis que l'on détruit entièrement pour ensuite en reconstruire une nouvelle de taille semblable. Il est donc beaucoup plus simple d'allouer un bloc mémoire de taille fixe contenant toute la Skip List. L'ajout consiste alors à initialiser les champs du nouveau nœud à partir du pointeur de zone libre et de déplacer ce pointeur de la taille du nœud nouvellement créé. La remise à zéro de la liste est donc particulièrement rapide, car elle consiste à déplacer le pointeur de zone libre à la fin de la zone mémoire occupée par la tête de la liste. Il reste à définir la taille du bloc mémoire. Un nœud (qui est une structure de taille fixe et un tableau de pointeurs ayant une taille maximale connue) va donc avoir une taille connue. Comme, grâce à la propriété 2, le nombre maximal de partiels est connu, la taille du bloc mémoire est donc connue.

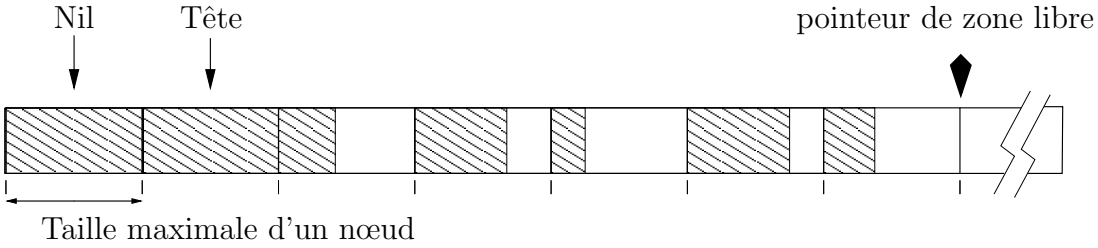


Fig. 20 – Allocation mémoire contiguë d'une Skip List.

La taille du tableau de pointeurs étant rarement maximale, on n'utilise pas de façon optimale la mémoire (au maximum, on occupe 2 fois plus de mémoire que nécessaire), mais notre objectif étant la rapidité, les pertes mémoires dues à cette optique nous ont semblé négligeables.

5 Résultats

Dans cette partie, nous présenterons les performances obtenues.

La légende ci-dessous rappelle la complexité des différents algorithmes.

1 • La sélection en amplitude :
   - algorithme par échantillonnage : $\mathcal{O}(n)$

2 • Le tri en amplitude :
   - "quick sort" : $\mathcal{O}(n_A \cdot \log(n_A))$

3 • La sélection en fréquence :
   - algorithme de référence : $\mathcal{O}(n_A^2)$
   - algorithme par échantillonnage : $tri + \mathcal{O}(n_C \cdot n_{\text{ech}})$
   - algorithme par stockage des contributions : $tri + \mathcal{O}(n_C \cdot \log(n_C))$

5.1 La sélection en fréquence

Dans cette partie, nous discuterons des performances des différentes implémentations du masquage en fréquence. Pour évaluer ces performances, nous avons traité deux cas, celui le pire, où aucun partiels n'est masqué et celui, meilleur, où la fréquence et l'amplitude des partiels sont tirées au hasard[1].

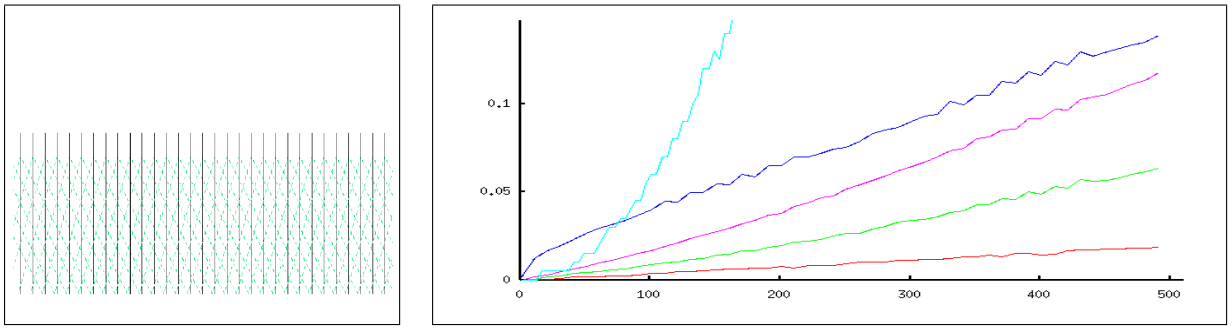


FIG. 21 – Les partiels ont même amplitude et se répartissent régulièrement en fréquence sur l'échelle Bark.

On voit clairement sur la figure 21 que l'algorithme de référence est de complexité quadratique. L'algorithme de masquage en fréquence par échantillonnage semble avoir une évolution quasi logarithmique, ceci est dû à la constante n_{ech} (ici, 2048) qui reste fixe. On pourraient donc penser que pour un nombre de partiels plus élevé, les performances de cet algorithme pourrait dépasser celles de l'algorithme par stockage des contributions. Mais dans ce cas, les erreurs d'arrondis deviennent très importantes. On peut aussi noter le coût assez prohibitif du tri en amplitude.

Dans le cas où la fréquence et l'amplitude des partiels sont tirées au hasard, le temps consacré à la sélection en amplitude (fig. 22) est le même que dans la figure précédente

[1]Cela correspond en moyenne à un taux de masquage de 30 %.

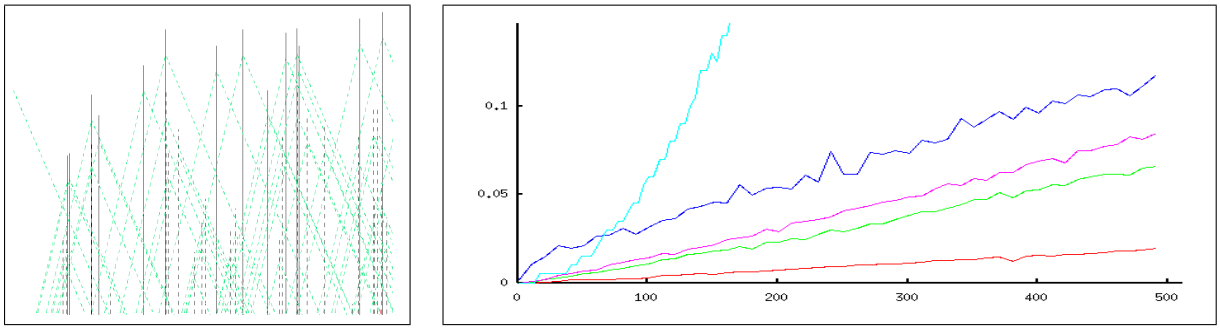


FIG. 22 – La fréquence et l'amplitude des partiels sont tirées au hasard.

(fig. 21). Le temps de tri a quelque peu diminué car certains partiels, étant inférieurs au seuil d'audibilité, ont déjà été filtrés. Quand un partiel ne contribue pas au masque (prop. 8), on n'a aucune mise à jour à effectuer. Ceci explique l'accélération sensible du masquage par stockage des contributions. On peut dégager de ceci une propriété importante : plus le phénomène de masquage est prononcé, plus l'algorithme de masquage est rapide.

5.2 Accélération de la synthèse

Les résultats présentés sont ceux de l'architecture logicielle de la figure 23.

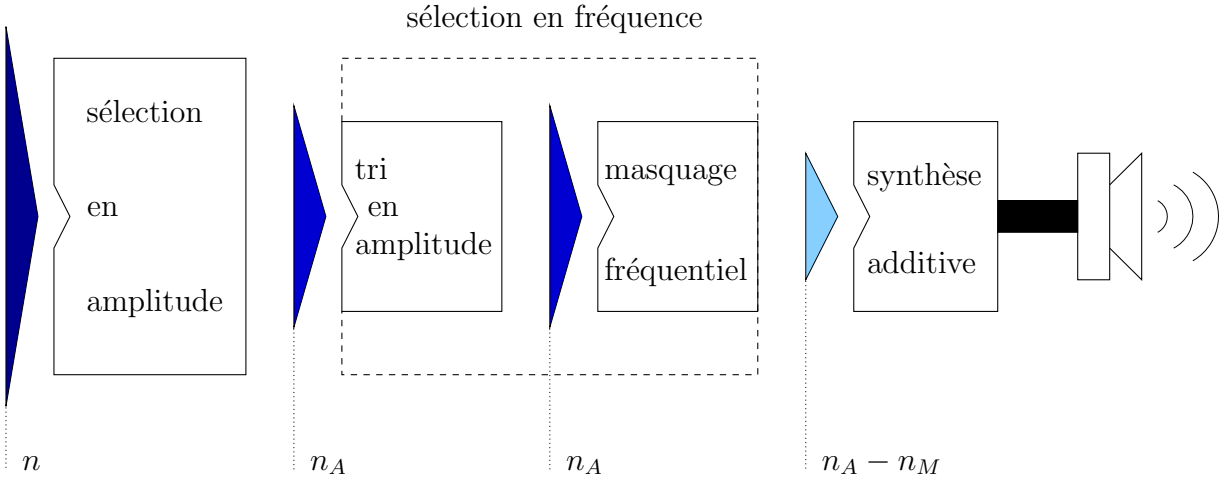
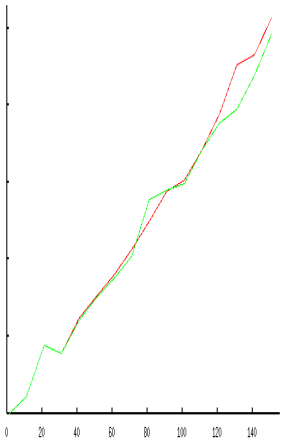


FIG. 23 – Architecture logicielle

L'algorithme de masquage en fréquence utilisé est celui de calcul des contributions. Nous présenterons d'abord les résultats pour un son harmonique simple - monophonique - (un arpège de guitare, une note de saxophone ou une voix seule), puis ceux pour un ensemble polyphonique d'un même instrument puis d'intruments différents.

Note :

En considérant l'évolution du masque dans le temps, on constate que les variations se font très lentement, il ne serait donc pas obligatoire de calculer à chaque fois le masque. Ainsi, Mahieux et Petit utilisent une fenêtre de taille 1024 lors de leur analyse spectrale [8]. La différence est inaudible pour tous les cas de tests dont nous disposons. Sur la figure, on a représenté en abscisse le nombre de partiels détectés comme masqués par l'algorithme. Une courbe représente les résultats obtenus avec une fenêtre de taille 64 et l'autre ceux obtenues avec une taille de 1024. On constate que l'erreur (intervalle entre les deux courbes) est très faible. Cette erreur pourrait être diminuée en effectuant une interpolation du masque entre l'instant t_0 et l'instant t_{1024}.



Légende :

- Synthèse complète.
- Synthèse avec calcul du masque tout les 64 échantillons.
- Synthèse avec calcul du masque tout les 1024 échantillons.

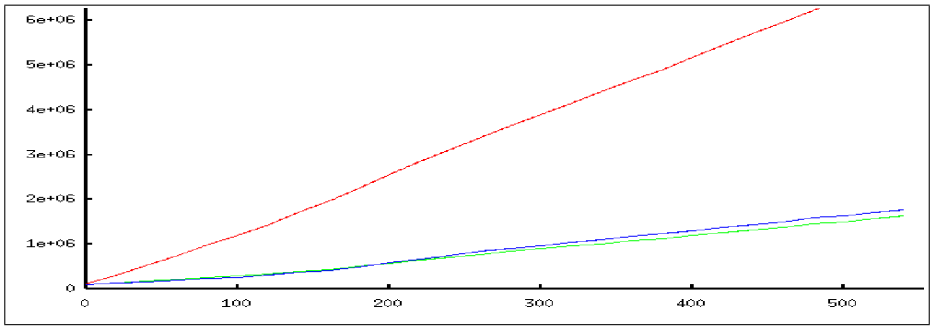


FIG. 24 – Résultats pour un son monophonique.

Le calcul quasi systématique (fenêtre de taille 64) du masque engendre, pour un son monophonique (fig. 24), un surcoût très élevé. Par contre, lorsque l'on calcule le masque avec un intervalle de 1024, le phénomène de masquage, même faible, permet déjà de compenser le surcoût dû au calcul du masque.

Note :

La synthèse de plusieurs sons monophoniques de même timbre (issus d'un même instrument) ne semble pas induire un taux de masquage très élevé. En effet, on voit sur la figure 25 que les rapports de temps sont sensiblement les mêmes que ceux obtenu lors de la synthèse d'un son monophonique (fig. 24).

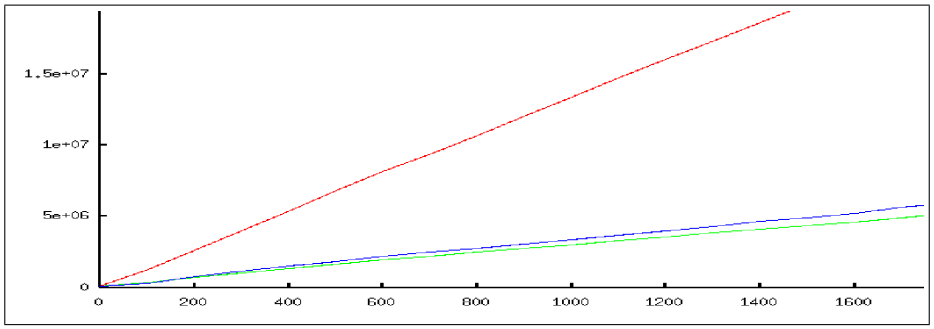


FIG. 25 – Résultats pour un ensemble de 4 sons monophoniques issu d'un même instrument.

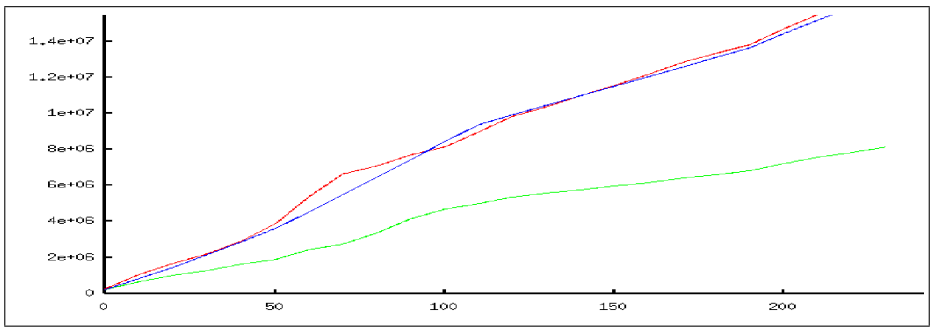


FIG. 26 – Résultats pour un ensemble polyphonique issu d'instruments variés.

Lors de la synthèse de plusieurs sons monophoniques de timbres différents, le phénomène de masquage est plus prononcé. La synthèse avec calcul du masque tous les 1024 échantillons est alors nettement plus efficace (fig. 26). On peut noter que plus le nombre de partiels augmente, moins le temps de ce type de synthèse croît. Alors que le temps de synthèse complète (sans prise en compte du phénomène de masquage) reste strictement linéaire. La suite est dédiée aux extensions possibles pour améliorer ces performances.

6 Extensions envisageables

Au vu des résultats developpés dans la partie précédente, on peut dégager deux problèmes majeurs :

- le coût en temps du tri en amplitude ;
- l'algorithme de sélection en fréquence calcule de façon extensive, apportant ainsi une précision souvent inutile.

Le deuxième point pose un problème d'ordre très général : l'algorithme doit être capable de s'adapter aux types de données qu'il a à traiter et de réguler ainsi le temps dédié

au calcul du masque. Pour cela, il nous faut d'abord un critère d'évaluation qui nous permette de déterminer si le calcul du masque est pertinent ou non.

Définition 7 *On définit l'*efficacité *de l'algorithme de masquage comme :*

$$\mathcal{E}(t) = \frac{n_M * t_{Synthèse}}{t_{Masquage}}$$

$n_M =$ *nombre de partiels déclarés comme masqués par l'algorithme de masquage*
$t_{Synthèse} =$ *temps de synthèse d'un partiel*
$t_{Masquage} =$ *temps dédié au calcul du masque*

Note :

Le principe de l'extension proposé dans cette partie est non plus de calculer le masque de façon extensive mais d'obtenir une approximation. n_M ne représente donc plus le nombre de partiels masqués, mais une valeur potentiellement inférieure à ce nombre. En effet, si on ne trouve pas tous les partiels masqués la synthèse sera plus longue. Par contre, on ne peut pas déclarer masqués plus de partiels qu'il n'y en a effectivement car le rendu sonore pourrait en être modifié.

6.1 Le seuil d'utilité

Grâce à ce critère d'évaluation, on peut savoir quand le masquage est un phénomène intéressant ($\mathcal{E} > 1$). Il faut maintenant pouvoir faire varier le temps consacré au masquage. Avec les algorithmes de calcul extensifs (cf. 3.4 et 4) vu précédemment, le seul moyen est de définir un certain "seuil d'utilité" égal à 1. On ne fait le calcul du masque au temps t uniquement si l'efficacité $\mathcal{E}(t-1)$ est supérieure à ce seuil, sinon on ne calcule plus le masque pendant un temps donné (fixé arbitrairement grâce à des mesures). L'algorithme de fenêtrage que nous présenterons dans la partie qui suit apporte une meilleure souplesse de traitement. En effet, grâce à ce procédé, on sera en mesure de faire varier avec toute la finesse nécessaire le temps consacré au masquage.

6.2 Le fenêtrage

Notation :

On notera n_f le nombre de fenêtres.

Le principe est le suivant : on considère un nombre n_f donné de tranches de la bande des fréquences, chaque tranche formant une fenêtre. Dans chaque fenêtre, on détermine le partiel de plus grande amplitude, en même temps - pour des raisons que nous expliciterons par la suite -, on stocke pour chaque partiel le numéro de fenêtre auquel il appartient.

Le tri en fréquence n'est pas nécessaire, il suffit pour chaque partiel de déterminer la fenêtre à laquelle il appartient et de mettre à jour la valeur courante de cette fenêtre si l'amplitude du partiel est plus élevée.

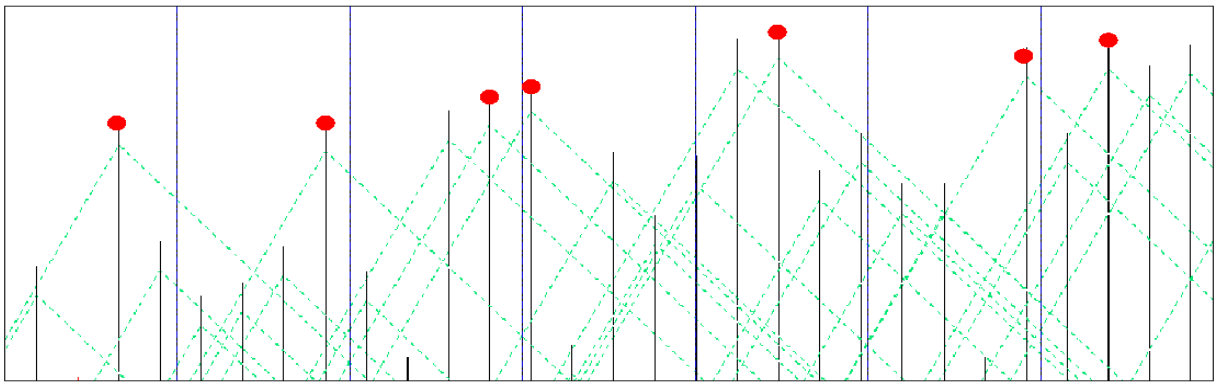


FIG. 27 – Le fenêtrage avec $n_f = 7$.

Algorithme :

```
1: Pour (j de 1 à n_f) faire
2:    partiels[j].fenêtre ⟵ fenêtre contenant j ;
3:    Si (fenêtre[fenêtre contenant j].volume < partiels[j].volume) alors
4:       mettre à jour(fenêtre[j]) ;
5:    fin Si
6: fin Pour
```

On obtient ainsi n_f partiels contribuant à une approximation du masque, la précision de cette approximation dépendant de n_f. Ainsi, si on veut calculer le masque de façon extensive, on fixe n_f à une valeur élevée, par contre, si on veut un temps de calcul très faible, on peut fixer n_f à une valeur faible.

6.3 La régulation

Il est donc aisé de réguler le temps consacré au masquage. En supposant que entre t_1 et t_2 on augmente n_f, si l'efficacité $\mathcal{E}(t)$ augmente ; au temps t_3 il faudra continuer à augmenter n_f. Par contre, si $\mathcal{E}(t)$ diminue, il faudra diminuer n_f. On peut imaginer plusieurs politiques de variations :

1 • A chaque pas, on fait varier n_f comme une puissance de 2 ;

- $n_f(t_{i+1}) = n_f(t_i) * 2$ si $\mathcal{E}(t_{i-1}) < \mathcal{E}(t_i)$ et $n_f(t_{i-1}) < n_f(t_i)$ ou $\mathcal{E}(t_{i-1}) > \mathcal{E}(t_i)$ et $n_f(t_{i-1}) > n_f(t_i)$
- $n_f(t_{i+1}) = n_f(t_i)/2$ si $\mathcal{E}(t_{i-1}) < \mathcal{E}(t_i)$ et $n_f(t_{i-1}) > n_f(t_i)$ ou $\mathcal{E}(t_{i-1}) > \mathcal{E}(t_i)$ et $n_f(t_{i-1}) < n_f(t_i)$

2 • A chaque pas, on fait varier n_f en fonction des valeurs de variations de l'efficacité.

- $n_f(t_{i+1}) = n_f(t_i) * \frac{\mathcal{E}(t_i)}{\mathcal{E}(t_{i-1})}$

Une composition de ces deux politiques est possible. Lorsque l'efficacité varie beaucoup, on peut prospecter grâce à des variations de grandes amplitudes obtenues avec la politique 1. Si n_f se met à osciller autour d'une valeur, on peut alors passer à la politique 2, plus fine. Inversement, lorsque l'efficacité décroit, on réutilise la politique 1, pour trouver la prochaine valeur de croisière.

6.4 La sélection en fréquence

Le fenêtrage permet d'obtenir les partiels qui serviront à construire le masque. Il reste alors à le construire. Il faut trier ces partiels en amplitude, et calculer pour chacun leur champ d'influence comme dans l'algorithme de stockage des contributions vu dans la section (cf. 4) que l'on place à l'intérieur des fenêtres. Le masque étant construit, pour chaque partiels, on détermine s'il est masqué ou non, ceci est particulièrement rapide car on sait déjà à qu'elle fenêtre appartient chaque partiel (6.2).

6.5 Complexité et architecture

Dans cette partie nous présentons la complexité et l'architecture de l'extension présentée précédemment :

1 • La sélection par le seuil d'audibilité

- algorithme par échantillonnage : $\mathcal{O}(n)$

2 • La sélection en fréquence

- Fenêtrage : $\mathcal{O}(n_A)$
- Tri en amplitude : $\mathcal{O}(n_f \cdot \log(n_f))$
- Calcul des contributions : $\mathcal{O}(n_f)$
- Masquage : $\mathcal{O}(n_A)$

La nouvelle architecture est représentée par la figure 28. Le régulateur se charge de définir la nouvelle valeur de n_f en fonction des valeurs de $top_{\text{début}}$, de top_{fin} et de n_M.

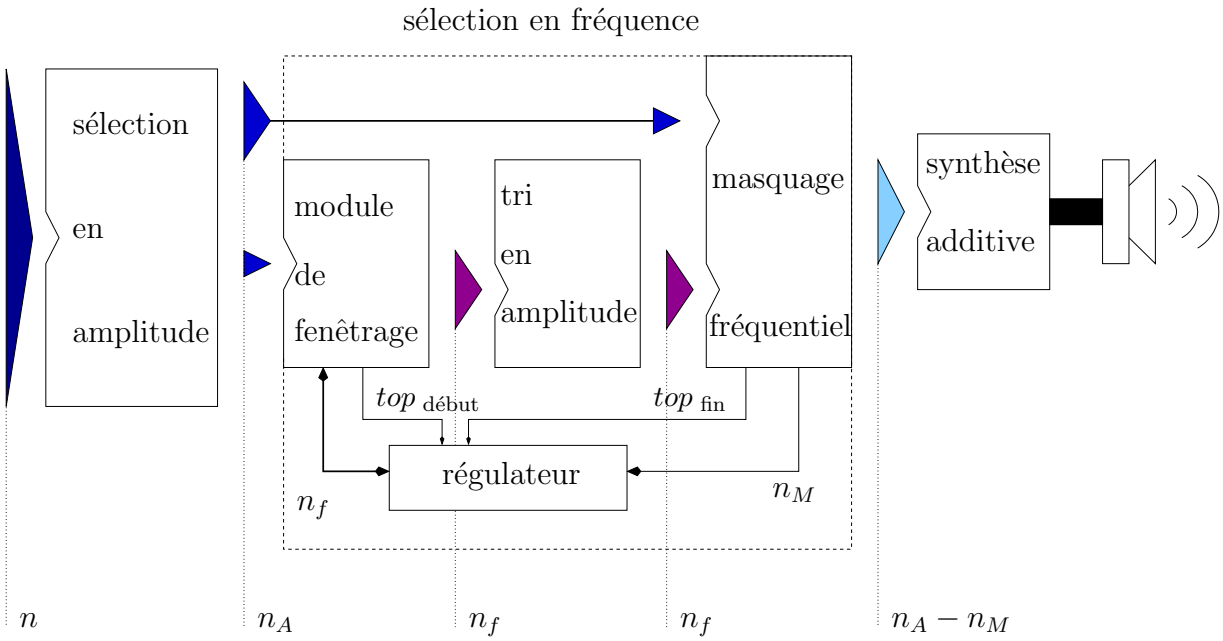


FIG. 28 – Nouvelle architecture logicielle.

Cette utilisation du fenêtrage permet d'abord de réduire la complexité de la construction du masque. En effet, lors du fenêtrage, on effectue un tri en fréquence implicite, ce qui nous permet de ne gérer ensuite qu'une simple liste. Ensuite, toujours grâce au fenêtrage, on constate que tous les modules de complexité élevée (tri en amplitude et calcul des contributions) sont contrôlés par n_f. L'influence de la régulation doit donc avoir une grande amplitude et nous donner toute la souplesse nécessaire.

Conclusion

Le but de ce mémoire était de réduire le nombre de partiels à synthétiser en supprimant les partiels inaudibles. Nous avons étudié et implémenté un algorithme d'échantillonnage du masque fréquentiel proposé par Marchand [1] exprimant une approche discrète du problème. Nous avons alors exploré une nouvelle approche, continue cette fois, en constituant une liste des partiels contribuant au masque. Nous avons alors obtenu un nouvel algorithme plus performant grâce notamment à l'utilisation d'une structure de données bien adaptée : les Skip-Lists.
Le gain obtenu grâce à l'exploitation des phénomènes psycho-acoustiques est encourageant, nous envisageons donc d'intégrer la sélection en amplitude et ce nouvel algorithme de sélection en fréquence dans un module de synthèse additive existant, nommé ReSpect [9].
Mais nous n'exploitons pas tous les aspects du phénomène de masquage en fréquence. En effet, tous les algorithmes de sélection en fréquence présentés se contentent de séparer les partiels masqués des partiels non masqués selon le signe du SMR (def. 4) associé à chaque partiel. Or la valeur du SMR est importante : plus un partiel se rapproche du masque (SMR faible), moins il est audible. La prise en compte de la valeur de ce SMR pourrait permettre d'améliorer encore la performance des algorithmes de synthèse, en autorisant par exemple une dégradation de la qualité de la synthèse pour les partiels presque inaudibles au profit de la rapidité de calcul.
Un module de synthèse utilisant une approximation polynômiale de la sinusoïde offre un large champ d'expérimentation. La synthèse des partiels graves s'y prête bien [1]. A la lumière de ce qui a été exposé dans le paragraphe précédent, un tel module pourrait aussi choisir d'accélérer la synthèse des partiels faiblement audibles en diminuant le degré de leurs polynômes associés.

Références

[1] Sylvain MARCHAND, *"Modélisation informatique du son musical"* Thèse de doctorat, Université Bordeaux 1, 2000.

[2] Robert STRANDH and Sylvain MARCHAND, *"Real-Time Generation of Sound from Parameters of Additive Synthesis."* In Proceedings of the Journées d'Informatique Musicale (JIM), pp. 83-88, Paris, May 1999. CEMAMu

[3] Ted PAINTER and Andreas SPANIAS, *"A Review of Algorithms for Perceptual Coding of Digital Audio Signals"* Page toile : `www.eas.asu.edu/~spanias`.

[4] E. ZWICKER and U. ZWICKER, *"Audio Engineering and Psychoacoustics : Matching Signals to the Final Receiver, the Human Auditory System"* J. Audio Eng. Soc., pp. 115-126, March 1991.

[5] Guillermo GARCIA and Juan PAMPIN, *"Data Compression of Sinusoidal Modeling Parameters Based on Psychoacoustic Masking"* Proceedings ICMC 1999 : pp. 40-43.

[6] P. PAPAMICHALIS, *"MPEG Audio Compression : Algorithms and Implementation,"* Int. Conf. on DSP, pp. 72-77, June 1995.

[7] William PUGH, *"Skip Lists : A Probabilistic Alternative to Balanced Trees"* Communications of the ACM, V.33, June 1990, pp. 668-676.

[8] Y. MAHIEUX and J. PETIT, *"Transform Coding of Audio Signals at 64 kbits/sec"* in Proc. Globecom '90, pp. 405.2.1-405.2.5, November 1990.

[9] Sylvain MARCHAND *ReSpect, Synthesis Software Package* Page toile : `www.scrime.u-bordeaux.fr/ProSpect`.

A Présentation du logiciel Mask

Mask est un logiciel illustrant les différents algorithmes présentés dans le mémoire *"Accélération de la synthèse sonore"* proposé par Sylvain Marchand et Robert Strandh. Ce logiciel (disponible à cette adresse : `http://dept-info.labri.fr/~lagrange`) est dédié à un système d'exploitation de type Linux et son interface utilise le Gimp Tool Kit (GTK) disponibles respectivement à ces adresses :

- `www.debian.org`
- `www.gtk.org`

Nous présenterons dans la suite l'interface et ses raccourcis claviers.

A.1 Interface

La première partie présente les différents boutons permettant de communiquer les paramètres requis. Et la seconde partie est consacrée aux différents outils de visualisation.

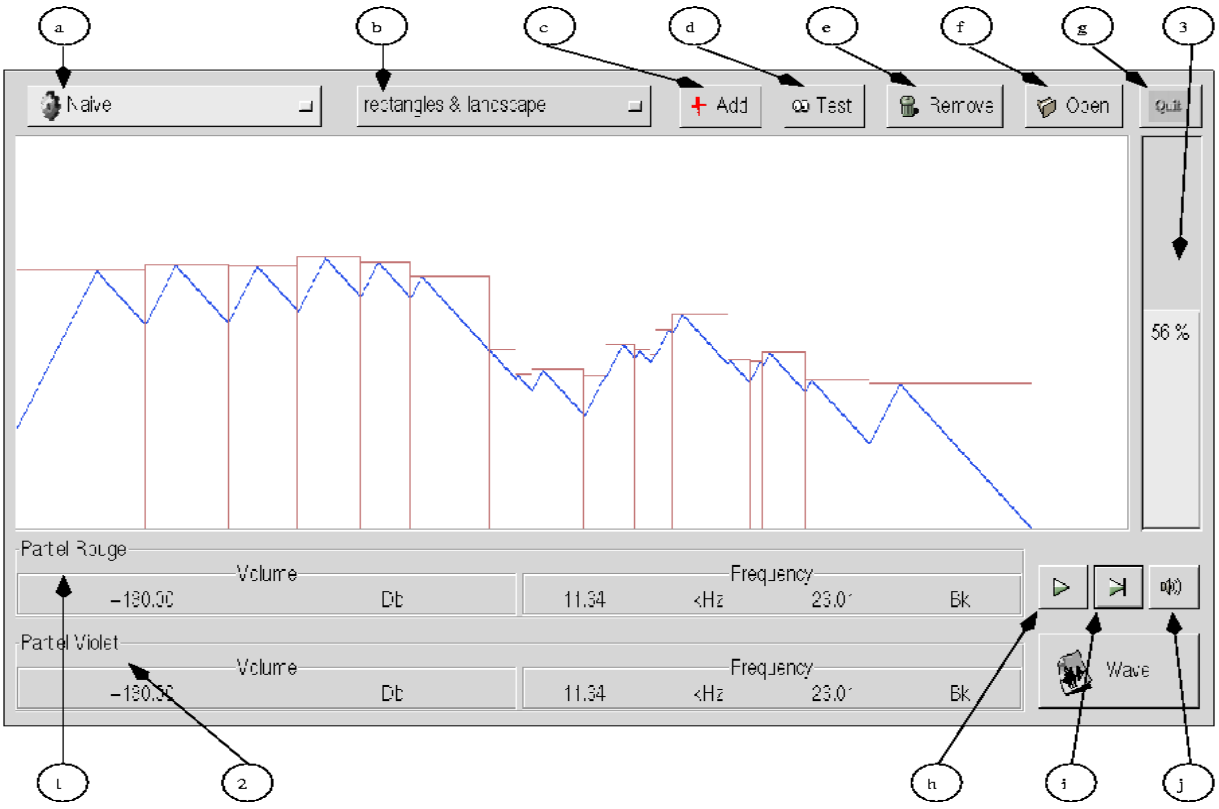


Fig. 29 – Interface graphique de Mask.

A.1.1 Communication

(A) • **Sélecteur de type de sélection :** permet de choisir le type de sélection désiré :
- *none* : aucun filtrage ;
- *silent* : sélection en amplitude ;
- *naive* : sélection en fréquence par l'algorithme naïf ;
- *landscape* : sélection en fréquence par l'algorithme d'échantillonnage ;
- *rectangles* : sélection en fréquence par l'algorithme de stockage des contributions ;
- *silent & rectangles* : sélection en amplitude puis sélection en fréquence par l'algorithme de stockage des contributions.

(B) • **Sélecteur d'affichage :**
- *partials* : affiche les partiels présents et leurs cônes de masquage ;
- *landscape* : affiche la silhouette du masque ;
- *rectangles* : affiche les rectangles représentant la zone d'influence des partiels ;
- *X & Y* : combinaison de X et de Y.

(C) • **Add :** permet d'ajouter un partiel ;

(D) • **Remove :** permet de supprimer le partiel sélectionné ;

(E) • **Test :** permet de créer un ensemble de partiels. Les trois boutons radios de la boite activée permettent respectivement de sélectionner :
- l'échelle des fréquences (Hertz ou Bark) ;
- si les partiels sont régulièrement espacés sur l'axe des fréquences ou si la fréquence des partiels est tirée au hasard pour chaque partiels ;
- si l'amplitude des partiels est tirée au hasard ou non.

(F) • **Open :** permet d'ouvrir un fichier au format MSM.
L'entrée de texte représente Les deux entrées de texte permettent de choisir :
- le décalage en nombre d'échantillons ajouté à chaque partiels du fichier ;
- le nombre de répétition.

Les deux boutons radios permettent de choisir entre :
- inverser ou non les partiels du fichier en cours d'ouverture ;
- supprimer tout les partiels existant avant l'insertion.

(G) • **Quit :** permet de terminer l'application.

(H) • **Forward :** permet d'actualiser tous les partiels avec les valeurs suivantes d'amplitude et de fréquence.

(I) • **forward until End :** permet d'actualiser tous les partiels avec les valeurs suivantes d'amplitude et de fréquence jusqu'à l'absence de nouvelles valeurs.

(J) • **Listen :** permet de jouer les partiels.

A.1.2 Visualisation

Il y a deux boites d'informations, chacune contient l'amplitude du partiel en décibels et la fréquence du partiel exprimée en Bark et en Hertz.

(1) • L'une concerne le partiel sélectionné (en Rouge),

(2) • l'autre le partiel sélectionné précédemment (en Violet) ;

(3) • La barre de progression représente le taux de partiels non synthétisés.

A.2 Raccourcis clavier

A.2.1 Les flèches

Les flèches de déplacements permettent de se déplacer parmi les boutons.

A.2.2 Le pavé numérique

⇑ **(8)** & ⇓ **(2)** permettent de changer la valeur de l'amplitude du partiel sélectionné ;

⇐ **(4)** & ⇒ **(6)** permettent de changer la valeur de la fréquence du partiel sélectionné.

A.2.3 Les touches textes

s & p : changent de partiel sélectionné (s pour suivant, p pour précédent) ;

a : active la boîte d'ajout ;

o : active la boîte d'ouverture de fichier ;

r : supprime le partiel sélectionné ;

t : active la boîte de test ;

f : actualise tous les partiels avec les valeurs suivantes d'amplitude et de fréquence ;

e : actualise tous les partiels avec les valeurs suivantes d'amplitude et de fréquence jusqu'à l'absence de nouvelles valeurs ;

l : joue les partiels ;

b : affiche sur l'entrée standard les temps de synthèse ;

q : termine l'application.

Pour tout commentaires ou détection de bugs, n'hésitez pas à me contacter :

`lagrange@labri.fr`.

B Index des abréviations

On note :

- $>_m$ la relation de masquage ;
- n le nombre de partiels à synthétiser ;
- n_A le nombre de partiels ayant leur amplitude supérieure au seuil d'audibilité ;
- n_M le nombre de partiels masqués ;
- n_C le nombre de partiels contribuant au masque ;
- n_S le nombre de partiels effectivement synthétisés ;
- n_{ech} le nombre d'échantillons de la silhouette ;
- n_f le nombre de fenêtres.