Modèle auditif en temps réel

Dan Gnansia

Mémoire pour le Master Sciences et Technologie de l'UPMC
Mention Informatique
Spécialité MIS
Parcours ATIAM

Stage au Département d'étude cognitive, Equipe audition
Encadré par Daniel Pressnitzer

Juillet 2005

Remerciements

Je souhaite d'abord remercier Daniel Pressnitzer, mon responsable de stage pour sa grande disponibilité, ses grandes qualités scientifiques et la confiance qu'il m'a accordée.

Je suis très heureux d'avoir participé à la création du laboratoire à l'ENS, en compagnie de mon maître de stage et de Alain de Cheveigné, que je remercie pour ces précieux conseils en traitement du signal.

Je remercie également Christian Lorenzi, Directeur du laboratoire, de me permettre de poursuivre en thèse avec l'équipe, sa confiance en moi et son enthousiasme m'ont beaucoup touchés.

Enfin, je remercie mes parents de m'avoir soutenu et aidé, sans eux le présent rapport aurait bien trop de fautes d'orthographes !

Table des matières

Chapitre I.
Introduction

I.1. Pourquoi faire un modèle auditif en temps réel ?

Le système auditif est un ensemble complexe de structures mécaniques et biologiques, influant toutes sur notre façon de percevoir les sons. Au fils des recherches menées dans ce domaine, de nombreux modèles informatiques ont été élaborés pour simuler le fonctionnement de l'oreille, et plusieurs servent maintenant de référence au calcul de plusieurs paramètres subjectifs de l'audition. Il est question dans cette étude d'élaborer un modèle auditif qui va permettre plusieurs applications en terme de psychophysique et de traitement du signal. Quel va être l'intérêt de notre modèle par rapport à ceux qui existent déjà ? Quels sont les atouts du temps réel ?

Il y a plusieurs raisons d'élaborer un modèle perceptif. D'abord, c'est un moyen de vérifier que les données physiologiques du fonctionnement de la perception (en particulier celles liées au masquage) sont cohérentes entre elles, et ainsi de pouvoir les relier. Un modèle est un moyen de synthétiser les connaissances actuelles dans un domaine, il est donc courant de voir que les différents modèles d'un même sujet vont en s'affinant à mesure que la connaissance du domaine se précise.

Un modèle est ensuite un outil : il permet de faire des prédictions. Dans notre cas, c'est par exemple un moyen de représenter la façon dont un son est perçu, et de confronter ces résultats aux expériences subjectives (de masquage par exemple) et aux mesures des paramètres de l'audition (attributs perceptifs notamment).

Un intérêt majeur du modèle est son architecture. En effet, tel que nous l'envisageons, il s'agit d'un banc de filtre auditif recréant l'analyse temps - fréquence dans la cochlée, un étage non linéaire, et quelques traitements supplémentaires. La plupart des attributs perceptifs peuvent être estimés grâce à cette construction commune. Il est cependant peu probable que ces prédictions se confondent parfaitement avec les résultats expérimentaux, mais le premier étage d'analyse restant plausible, les tendances importantes doivent être respectées. L'architecture commune permet par ailleurs de comparer les calculs d'attributs entre eux, et de montrer ce qui est spécifique à chacun.

Dans le domaine de l'informatique musicale, il existe un certain nombre d'algorithmes de traitement audio qui se basent sur des représentations temps-fréquence. Un modèle auditif produit un type particulier de représentation temps-fréquence, qui peut donc servir de base à de nouveaux traitements audio. Nous allons en particulier choisir une implémentation du modèle permettant une resynthèse très peu dégradée. Il est ainsi possible d'imaginer des manipulations temporelles et fréquentielles après un étage de décomposition par le modèle auditif, puis de resynthétiser le son résultant. L'avantage de notre modèle est que les paramètres accessibles pour les manipulations sont sensés avoir un véritable effet perceptif.

C'est enfin un traitement calculé en temps réel. Une telle spécificité ouvre de nouveaux horizons, surtout en terme d'applications. Il est ainsi possible d'envisager des traitements audio en vue d'applications musicales, où les paramètres sont changés en temps réel. Plusieurs simulations temps réel sont réalisables : simulation d'implant cochléaire par vocodeur, synthèse croisée…

Le but de ce stage est donc d'implémenter le modèle auditif dans l'environnement temps réel Pure Data, et d'utiliser par la suite cet outil dans un maximum de contextes (simulations, calculs d'attributs, traitements audio). Bien évidemment, de nombreux modèles auditifs existent déjà, pourtant une seule version temps réel a été relevée [19], mais utilise une autre approche de décomposition, et surtout ne permet pas la resynthèse.

Le présent rapport va d'abord présenter des modèles de certains aspects de la perception auditive. La méthode de décomposition en banc de filtre retenue pour le modèle, la décomposition par banc de filtre Gammatone, sera alors précisée. Nous verrons les caractéristiques de ces filtres, ainsi que la construction des bancs d'analyse et de synthèse, pour laquelle il a fallu apporter un soin particulier, notamment quelques modifications et adaptations nécessaires dans le contexte temps réel.

Ensuite, nous présenterons l'environnement temps réel Pure Data et les choix retenus pour l'implémentation et l'utilisation du banc de filtre. Une bibliothèque complète a en effet été développée pour accompagner les bancs d'analyse et de synthèse, plusieurs problèmes ont dû être résolus pour y parvenir.

Enfin, il sera proposé plusieurs utilisations du modèle :
- Une visualisation, qui reproduit certaines caractéristiques de la perception, qui permet de se rapprocher d'une visualisation de type 'ce que l'on voit est ce que l'on entend'.
- Deux traitements audio (simulation d'implant cochléaire, synthèse croisée) mettant en évidence les possibilités offertes par le temps réel.
- Le calcul d'attributs perceptifs en temps-réel (sonie, rugosité).

I.2. Masquage auditif

Le masquage auditif est à la base de l'élaboration du modèle perceptif sujet de l'étude. Il est donc important de définir précisément ce terme et de décrire qualitativement ce phénomène.

Le masquage auditif est un phénomène perceptif qui fait qu'un son pourtant audible n'est pas perçu, à cause de la présence d'un autre son : le signal non perçu est alors dit masqué. En termes psychoacoustiques, le son masqué est appelé son cible, et l'autre masqueur. De façon plus précise, le masqueur élève localement le seuil d'audition, le son cible nécessite alors plus d'énergie pour être perçu.

Il existe deux types de masquage : le masquage simultané ou fréquentiel, et le masquage temporel. Le masquage fréquentiel apparaît lorsque que les deux sons sont présentés simultanément : on observe une élévation du seuil d'audition pour les fréquences

proches du masqueur. Le masquage temporel est par contre produit par la présence du masqueur juste avant ou juste après la cible : le seuil de détection est augmenté aux abords du masqueur. Le masquage fréquentiel est celui qui produit les effets de plus grandes amplitudes. Dans la suite le l'étude, nous nous intéresserons seulement à ce masquage simultané.

Le phénomène de masquage est à rapprocher du concept de 'bande critique auditive'. Cette approche découle du traitement de l'information sonore dans la cochlée (organe de l'oreille interne), et plus particulièrement de la vibration de la membrane basilaire. En effet, cet organe vibre en présence d'un son, et chaque point est sensible à une certaine fréquence pour laquelle l'amplitude de la vibration sera maximale. Ainsi, pour une fréquence donnée, la membrane basilaire est excitée a un point précis, entraînant une vibration de plus faible amplitude aux fréquences voisines : chaque point de la membrane basilaire peut alors être considéré comme un filtre passe bande d'une certaine largeur, la bande critique. Cette organisation, où chaque fréquence correspond à une localisation d'excitation, est appelée tonotopique.

Si la membrane basilaire est excitée à une certaine fréquence (par un masqueur), une nouvelle excitation à une fréquence proche (son cible) ne sera alors perçue que si elle surmonte la vibration induite par la première. Il est généralement admis que cette compétition entre excitations fourni la base des phénomènes de masquage.

Cette approche de bande critique auditive va servir de base à l'élaboration du modèle auditif. La modélisation fonctionnelle du concept de bande critique se fera à l'aide d'un banc de filtre visant à reproduire l'excitation mécanique de la membrane basilaire. Le signal de chaque canal du banc de filtre correspond alors à la déformation de la membrane basilaire à un endroit donné. A partir de l'observation des phénomènes de masquage, on va pouvoir déterminer la forme de ces filtres, leur largeur, et la façon de les disposer dans le banc. Il est important de remarquer que le modèle basé sur ces filtres ne sert pas seulement pour le masquage, puisque nous allons pouvoir calculer plusieurs attributs perceptifs qui ne sont pas directement liés au masquage. Le modèle simule en fait la variation de sélectivité fréquentielle du système auditif en fonction de la fréquence, ce qui est la principale différence des étapes précoces de l'encodage cochléaire avec une analyse de type transformée de Fourier à court terme.

I.3. Le stage

J'ai effectué mon stage au sein de l'équipe 'Audition: psychophysique, modélisation, neurosciences'. Cette jeune équipe possède une double affiliation: au Département d'Etudes Cognitives (DEC) de l'ENS et au Laboratoire de Psychologie Expérimentale. L'équipe compte actuellement quatre membres statutaires et un membre attaché. Elle possède déjà de nombreuses collaborations internes, internationales et universitaires.

Le stage a été encadré par Daniel Pressnitzer, chargé de recherche de première classe au CNRS, et s'est déroulé du 1 mars au 1 juillet 2005.

Les travaux effectués pendant le stage font l'objet d'une présentation au 'Summer School S2S²', en juillet 2005 à Gênes, et d'un article coécrit par Daniel Pressnitzer, acceptée pour publication à la conférence ICMC en septembre 2005 à Barcelone [26].

Chapitre II.
Le Modèle auditif

II.1. Les modèles de perception

II.1.a. Filtres auditifs

La perception du son commence par la conversion d'un champ de pression acoustique en impulsions neuronales grâce aux cellules ciliées disposées le long de la membrane basilaire. Chacune de ces cellules correspond à une fréquence, et les caractéristiques du système sont très sélectives. A l'aide d'une fréquence pure, une expérience de neurophysiologie chez l'animal (où l'on enregistre l'activité d'une fibre du nerf auditif) permet de construire le seuil de détection pour un seul neurone : c'est la plus faible amplitude nécessaire pour exciter le nerf auditif en fonction de la fréquence. Une approximation est montrée figure 2.1. On remarque un passe-bande pointu centré sur la fréquence d'excitation. L'idée de la modélisation est donc de considérer la membrane basilaire et les cellules ciliées comme un banc de filtres passe-bande, appelés filtres auditifs. Dans le cadre de filtres linéaires, il est possible de retrouver le transfert en gain de ces filtres en inversant la courbe de seuil de détection.

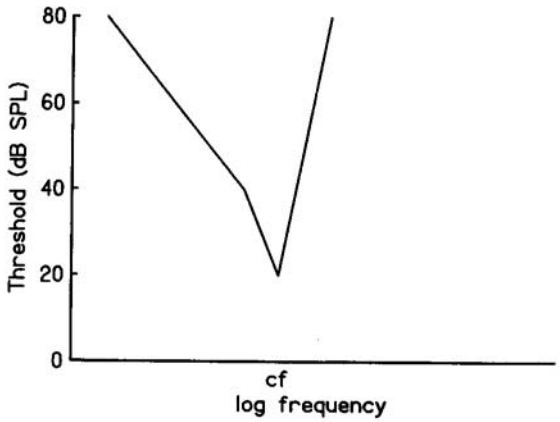


Fig. 2.1 : Modèle de seuil de détection pour un neurone du nerf auditif. Le seuil à la fréquence cible est de 20 dB SPL. D'après Hartmann [1].

Une telle approche est cohérente avec les phénomènes de masquage simultané. En effet, pour une fréquence d'excitation donnée, les fréquences proches sont sollicitées. Il est donc prévisible que le seuil d'audition sera ainsi augmenté pour un autre son pur aux abords de cette première fréquence, si celui-ci n'a pas assez d'amplitude, il est alors masqué.

Historiquement, les filtres auditifs ont été caractérisés en premier lieu par leur largeur de bande, sous l'hypothèse qu'ils pouvaient être modélisés par des filtres passe-bande idéaux, c'est-à-dire rectangulaires. Etant donné la complexité des phénomènes mis en jeu au niveau de la cochlée, cette hypothèse est évidemment très simplificatrice. En effet, des mesures plus précises des filtres auditifs ont montré leur caractère non linéaire et en particulier leur dépendance vis-à-vis du niveau des stimuli. Les filtres auditifs ont en réalité une bande passante plus étroite pour les niveaux modérés que pour les niveaux forts [22].

Cependant, dans ce cadre très complexe, une modélisation par des filtres passe-bande idéaux constitue une première approche pratique et permet de quantifier approximativement la résolution fréquentielle du système auditif. Moore et Glasberg [2] ont proposé une mesure de la largeur de ces filtres idéaux appelée *largeur de bande rectangulaire équivalente* ou *Equivalent Rectangular Bandwidth* (ERB). Pour chaque filtre auditif, la valeur d'ERB est définie comme la largeur d'un filtre passe bande idéal de même fréquence centrale qui, alimenté par un bruit blanc, produirait la même énergie en sortie. La mesure des ERB illustre la résolution fréquentielle du système auditif. La formule suivante donne la relation moyenne pour plusieurs normo-entendants entre la fréquence centrale f et l'ERB en Hz :

$$ERB(f) = l + \frac{f}{q}, \text{ avec } l = 24{,}7 \text{ et } q = 9{,}265 \qquad (2.1)$$

On peut définir une échelle des ERB. Chaque numéro d'ERB correspond alors à une distance fréquentielle d'une bande rectangulaire équivalente.

$$N°ERB(f) = q \log_{10}\left(1 + \frac{f}{l.q}\right) \qquad (2.2)$$

Les filtres auditifs sont définis pour toute fréquence audible et la largeur des filtres augmente avec la fréquence, de telle sorte que cette largeur vaut approximativement 11% à 17% de la valeur de la fréquence centrale. Ces caractéristiques influent sur la résolution fréquentielle du système auditif, en effet, la discrimination fréquentielle de l'oreille est meilleure pour les basses fréquences que pour les hautes.

Schématiquement, on peut résumer ainsi les propriétés générales des filtres auditifs :

- Le sommet de chaque filtre est étroit et à peu près symétrique en fréquence.
- Les bords sont larges et asymétriques, en particulier leur pente du côté des basses fréquences est moins forte.
- Plus le niveau du stimulus est élevé, plus les filtres sont larges.

La modélisation par les filtres auditifs permet également une représentation schématique de l'excitation de la membrane basilaire en réponse à un stimulus. Cette représentation est appelée *pattern d'excitation*, anglicisme tiré de l'expression *excitation pattern*. Moore et Glassberg ont proposé une méthode simple pour estimer le pattern d'excitation à partir de la sortie des filtres auditifs. Celui-ci est obtenu en relevant le niveau de sortie de chaque filtre en fonction de sa fréquence centrale.

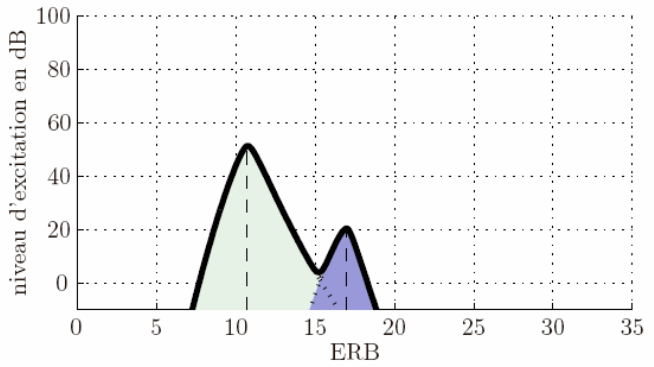


Fig. 2.2 : Pattern d'excitation de deux sons purs de niveau modéré. F1 = 500Hz (50 dB SPL) et F2 = 1200Hz (20 dB SPL).

La figure 2.2 montre un exemple de différents patterns d'excitation :

- celui d'un son pur à 500Hz (10,7 en échelle ERB) et d'intensité 50 dB SPL indiqué par la courbe englobant la zone claire centrée à 10,7 ERB,
- celui d'un son pur à 1200Hz (17 en échelle ERB) et d'intensité 20 dB SPL indiqué par la courbe englobant la zone d'aire foncée centrée à 17 ERB,
- enfin le pattern d'excitation complexe composé des deux fréquences précédentes, représenté par la courbe supérieure en trait gras.

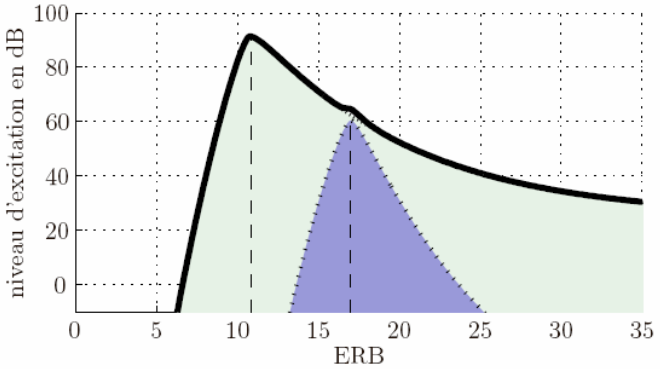


Fig. 2.3 : Pattern d'excitation de deux sons purs de fort niveau. F1 = 500Hz (90 dB SPL) et F2 = 1200Hz (60 dB SPL).

Le pattern d'excitation est un outil intéressant car il permet d'une part de retrouver la tonotopie de la membrane basilaire, et d'autre part d'illustrer avec clarté le phénomène de masquage auditif. En effet, le stimulus de la figure 2.2 présente deux pics fréquentiels distincts correspondant aux deux fréquences d'excitation. Si on considère maintenant le même exemple mais pour des sons purs rehaussés de 40dB chacun, on obtient le pattern d'excitation de la figure 2.3. A ce niveau de stimulation, les filtres auditifs sont plus larges. On remarque alors que le pattern d'excitation du premier son pur indique une excitation beaucoup plus étendue sur la membrane basilaire que dans le premier exemple et surtout qu'en tout point de la membrane basilaire, ce premier apporte plus d'énergie que le second. Le second pattern d'excitation est ainsi complètement recouvert par le premier. Le pattern d'excitation résultant

est pratiquement confondu avec le premier et n'indique plus qu'un seul pic fréquentiel : on voit donc ici clairement apparaître le phénomène de masquage.

Notons que l'effet d'élargissement des filtres avec le niveau d'excitation ne sera pas explicitement simulé dans le banc de filtre mais plutôt dans un étage de compression non linéaire qui sera placé après, et qui produit des effets qualitativement similaires.

II.1.b. Les modèles auditifs existants

Plusieurs modèles auditifs existent et sont implémentés sur diverses plateformes. Les plus utilisés sont DSAM, HUTear et Auditory Toolbox.

La bibliothèque DSAM est implémentée en C. Elle peut être utilisée pour programmer des applications de traitement du son, ou directement grâce au logiciel AMS (Auditory Model Simulator). Cette bibliothèque simule de nombreux modèles issus de la recherche sur la physiologie de l'oreille, en particulier le banc de filtre auditif Gammatone, que nous allons détailler dans la suite de l'étude, le modèle de cellules ciliés externes de R. Meddis [6, 7], et l'outil d'imagerie auditive Auditory Image Model (AIM) de Patterson [8, 9, 10].

HUTear est une boite à outils Matlab. Elle simule chaque étage de l'audition : l'oreille externe, interne, la cochlée, les cellules ciliées externes, et les neurones du nerf auditif. Pour chaque étage, plusieurs modèles sont disponibles, ils sont détaillés figure 2.4. Pour utiliser HUTear, il faut choisir un modèle par niveau d'analyse, un signal d'entrée, et le programme calcule l'état à chaque étage. Des pré et post-traitements sont également disponibles.

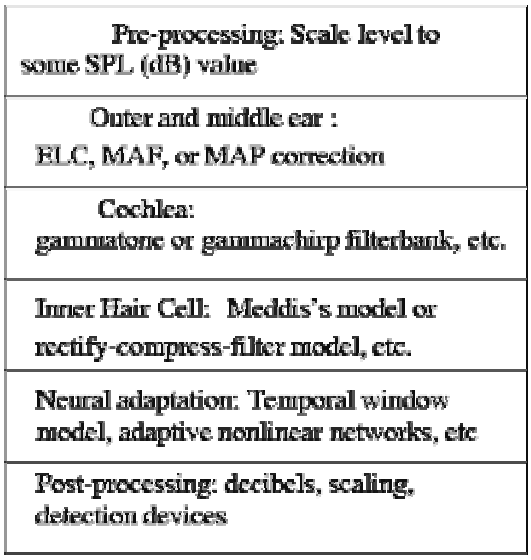


Fig. 2.4 : Diagramme d'un modèle auditif générique avec HUTear.

La bibliothèque Auditory Toolbox est une autre boite à outil Matlab développée par Malcolm Slaney. C'est également une compilation de plusieurs modèles auditifs issus de la recherche, mais il faut noter l'importante part réservée à la représentation. En plus du modèle en banc de filtre Gammatone, et du modèle de cellules cillées externes de R. Meddis, la bibliothèque contient le modèle auditif de R. F. Lyon [11] basé sur une approche de la membrane basilaire en ligne de transmission suivi de plusieurs étages d'adaptation, et celui de

S. Seneff [12] qui décrit la cochlée en combinant un banc de filtre à bandes critiques et des modèles de détection. D'autres outils de représentation et de reconnaissance vocale basés sur l'échelle de MEL et les coefficients cepstraux sont aussi disponibles dans la bibliothèque.

Tous ces modèles présentent en fait une architecture et une approche très similaires. Le diagramme de la figure 2.4 peut en vérité être celui de n'importe quel d'entre eux. Le modèle auditif que nous allons mettre en œuvre respecte également cette architecture, même si les contraintes de puissance de calcul liées au temps réel nous imposent plusieurs simplifications. Il est en effet important de préciser que ces modèles auditifs sont implémentés sur des plateformes qui ne fonctionnent pas en temps réel.

II.2. Le banc de filtres Gammatone

II.2.a. Introduction sur les filtres Gammatone

Les filtres Gammatone sont souvent utilisés pour la modélisation du traitement du son dans la cochlée [4]. Plusieurs implémentations de ces filtres sont connues, toutes proviennent de diverses approximations du filtre Gammatone original.

Pour notre modèle auditif, il a été retenu un banc de filtres Gammatone tout pôle d'ordre 4, à sortie complexe, proposée par Hohmann [3]. Afin de pouvoir générer rapidement le filtre, ce dernier a donc été simplifié par omission du numérateur, en effet, il est dérivé de la version échantillonnée de la réponse impulsionnelle Gammatone analogique :

$$g_\gamma(n) = n^{\gamma-1}.\tilde{a}^n \ , \ n \geq 0 \qquad (2.3)$$
$$\text{avec} \ \ \tilde{a} = \lambda.\exp(i\beta) \ ,$$

où λ traduit l'atténuation, β la fréquence d'oscillation, γ l'ordre du filtre de n le temps échantillonné, $\tilde{a}$ est un coefficient complexe. Pour γ=4, on obtient la transformée en z suivante :

$$G(z) = \frac{\tilde{a}z^{-1} + 4.(\tilde{a}z^{-1})^2 + (\tilde{a}z^{-1})^3}{(1-\tilde{a}z^{-1})^4} \qquad (2.4)$$

On simplifie alors cette fonction de transfert pour ne garder finalement que le dénominateur :

$$K(z) = \frac{1}{(1-\tilde{a}z^{-1})^4} \qquad (2.5)$$

Les pôles d'une fonction de transfert ayant généralement une influence bien plus importante sur la réponse en fréquence que les zéros pour les filtres a réponse impulsionnelle infinie (*Infinite Impulse Response*, IIR), tout laisse à penser que la réponse en fréquence ne sera que légèrement modifiée. On peut effectivement le vérifier sur la figure 2.5 : ne pas prendre en compte le numérateur de la fonction de transfert ne modifie que très peu la réponse en fréquence, et la plupart de ces différences sont situées dans la bande coupée. Par contre, comme les effets sur la réponse temporelle peuvent être plus prononcés, nous allons calculer l'erreur temporelle commise par cette approximation.

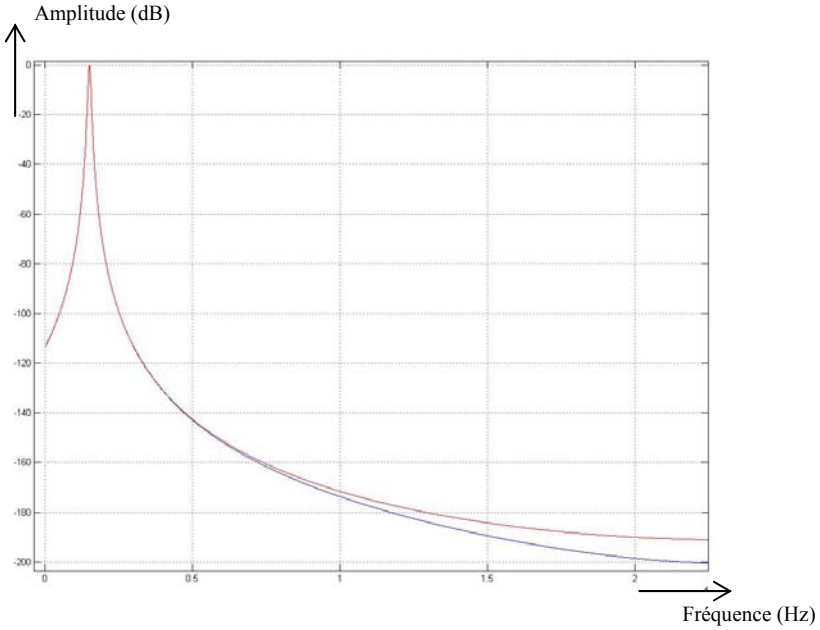


Fig. 2.5 : Réponse en fréquence des fonction G(z) (en bleu) et K(z) (en rouge) pour $\tilde{a}$ *=0.9693 + 0.2104i, correspondant à une fréquence centrale de 1,5kHz et un largeur de bande de 50Hz.*

L'expression temporelle $k(n)$ de la fonction de transfert $K(z)$ est la suivante :

$$k(n) = \frac{\tilde{a}^n}{6}.(n^3 + 6n^2 + 11n + 6)$$

En négligeant le facteur $\frac{1}{6}$, l'erreur absolue $e(n)$ et relative $r(n)$ de $k(n)$ par rapport a $g(n)$ est :

$$e(n) = 6.k(n) - g(n) = g(n)\left(\frac{6}{n} + \frac{11}{n^2} + \frac{6}{n^3}\right),\ n>0 \qquad (2.6)$$

$$r(n) = \frac{e(n)}{g(n)} = \frac{6}{n} + \frac{11}{n^2} + \frac{6}{n^3} \qquad (2.7)$$

L'erreur relative étant en $1/n$, la forme et la structure temporelle fine du filtre Gammatone, qui ont une importance majeure pour la modélisation, notamment la caractéristique de phase de la cochlée, sont préservées par cette approximation tout-pôle.

II.2.b. Synthèse d'un filtre

Nous allons maintenant voir comment synthétiser un filtre à partir d'une fréquence centrale et d'une largeur de bande, puis décrire la construction du banc de filtre complet.

L'approximation tout pôle que nous avons faite limite la synthèse d'un filtre au calcul de $\tilde{a}$. Tout d'abord, la fréquence centrale est donnée par la phase β de $\tilde{a}$. Pour une fréquence centrale et une fréquence d'échantillonnage donnée, le paramètre β prend la valeur :

$$\beta = 2\pi \frac{f_c}{f_s}, \qquad (2.8)$$

il représente alors directement l'angle correspondant à f_c dans le plan complexe en z.

Pour le calcul du paramètre λ, on trouve une relation avec la largeur de bande f_b en calculant le gain de la fonction de transfert à la moitié de la largeur de bande désirée. Ainsi en partant de l'expression de la fonction de transfert Gammatone 2.5 :

$$10\log_{10}\left(\frac{1}{\left(1-\tilde{a}z^{-1}\right)^4}\right) = -3$$

$$\text{avec } \tilde{a} = \lambda e^{2i\pi\frac{f_c}{fs}} \text{ et } z = e^{2i\pi\frac{f_b/2}{f_s}}$$

on obtient :

$$10\log_{10}\left(\frac{(1-\lambda)^2}{1-2\lambda\cos\phi+\lambda^2}\right) = -\frac{3}{4},\ \phi = 2\pi\frac{f_b/2}{f_s}$$

On résout cette expression ainsi :

$$\lambda = -\frac{p}{2} - \sqrt{\frac{p^2}{4}-1}, \text{ avec } p = \frac{-2+2.10^{u/10}\cos\phi}{1-10^{u/10}} \text{ et } u = -\frac{3}{4} \qquad (2.9)$$

On vérifie figure 2.6 les caractéristiques temporelles et figure 2.7 les caractéristiques fréquentielles d'un filtre Gammatone simplifié tout pôle généré à l'aide des équations ci-dessus. Notons que le filtre est normalisé par un facteur $2(1-|\tilde{a}|)^4$ pour que le module de la fonction de transfert soit de 0dB à la fréquence centrale.

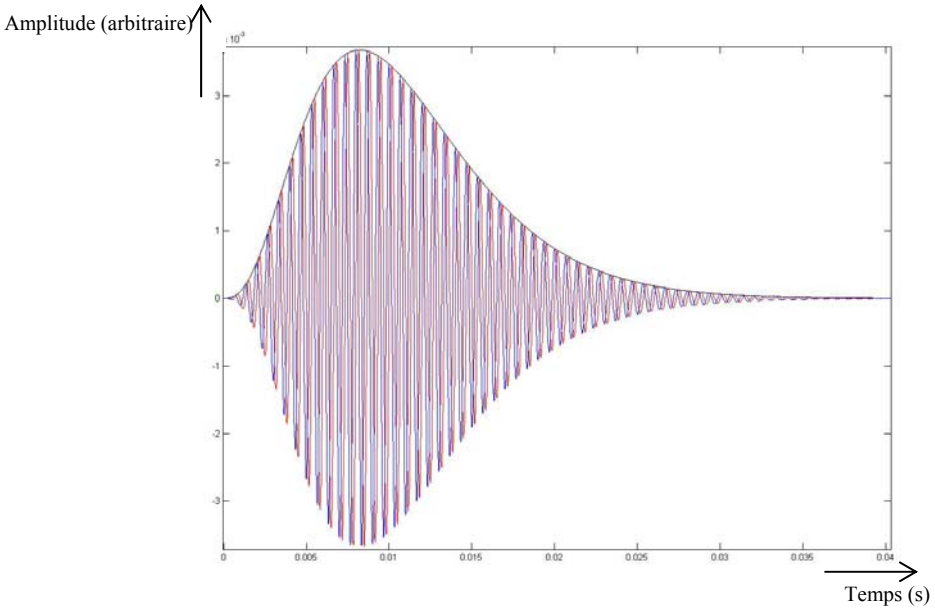
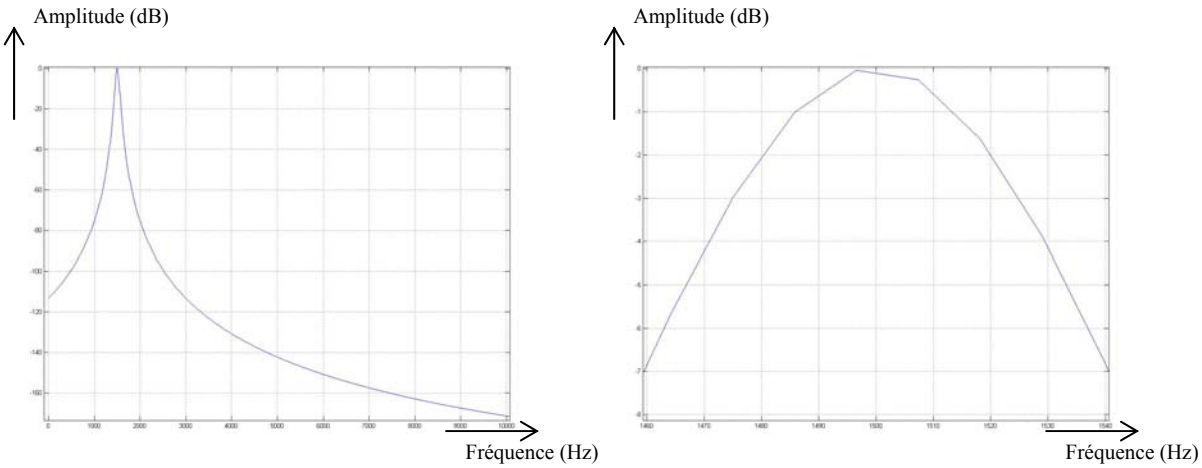


Fig. 2.6 : Réponse impulsionnelle d'un filtre Gammatone, fréquence centrale f_c =1,5kHz, largeur de bande f_b =50Hz. Partie réelle en bleu, imaginaire en rouge, valeur absolue (enveloppe) en noir.



(a) *(b)*

Fig 2.7 : Réponse en fréquence d'un filtre Gammatone, fréquence centrale f_c =1,5kHz, largeur de bande f_b =50Hz (a), détails (b).

Pour un tel filtre, on trouve $\tilde{a}$ =0.9693 + 0.2104i.

II.2.c. Banc d'analyse complet

Il faut maintenant construire la totalité du banc de filtre. Pour cela, on reprend le concept de largeur de bande rectangulaire équivalente (ERB) de Moore et Glasberg [2] vu dans la section II.1.a pour choisir les fréquences centrales et les largeurs de bande de nos filtres Gammatone. Il faut alors repartir des équations 2.1 et 2.2 afin de définir les fréquences centrales et les largeurs de bande de chaque filtre du banc. La génération de ces paramètres est faite de la façon suivante :

- on choisit une fréquence de départ, fréquence la plus basse du banc de filtre, $f_{\min}$
- on choisit un pas en ERB (inverse du nombre de filtre par ERB désiré), ERB_{scale}
- on choisit le nombre de bandes du banc, $N_{channels}$

Il faut d'abord déterminer la valeur en ERB du premier filtre grâce à l'expression 2.2 :

$$ERB_0 = q.\log_{10}\left(\frac{f_{\min}}{l.q}\right)$$

On peut alors calculer toutes les fréquences centrales des filtres Gammatone du banc en inversant l'expression 2.2 de la façon suivante :

$$\text{Pour } i=0 \text{ à } N_{channels}-1,\ f_c[i] = \left(\exp\left(\frac{ERB_0 + i.ERB_{scale}}{q}\right) - 1\right).l.q$$

Ensuite, pour chaque filtre on détermine la largeur de bande f_b à partir de 2.1. Patterson *et al.* [1] nous indique que la bande passante à -3dB d'un filtre Gammatone est bien approximée par la largeur de bande rectangulaire équivalente, à laquelle on ajoute un coefficient correctif *k*. En repartant de 2.1, on a finalement pour chaque filtre l'expression f_b suivante :

$$f_b = k\left(l + \frac{f_c}{q}\right), \text{ avec } k = 0.8861 \text{ pour des filtres Gammatone d'ordre 4.}$$

II.2.d. Résultats

La figure 2.8 nous donne la représentation fréquentielle d'un banc de filtres Gammatone 30 bandes, de fréquence minimale 70Hz. On remarque alors que la caractéristique du système auditif liée à la précision spectrale est préservée dans ce modèle, puisqu'elle est meilleure pour les basses fréquences que pour les hautes, avec un pattern qui s'élargit au fur et à mesure que la fréquence augmente. Cette propriété était prévisible puisque ce concept des bandes critiques a servi de base à l'élaboration du banc de filtre.

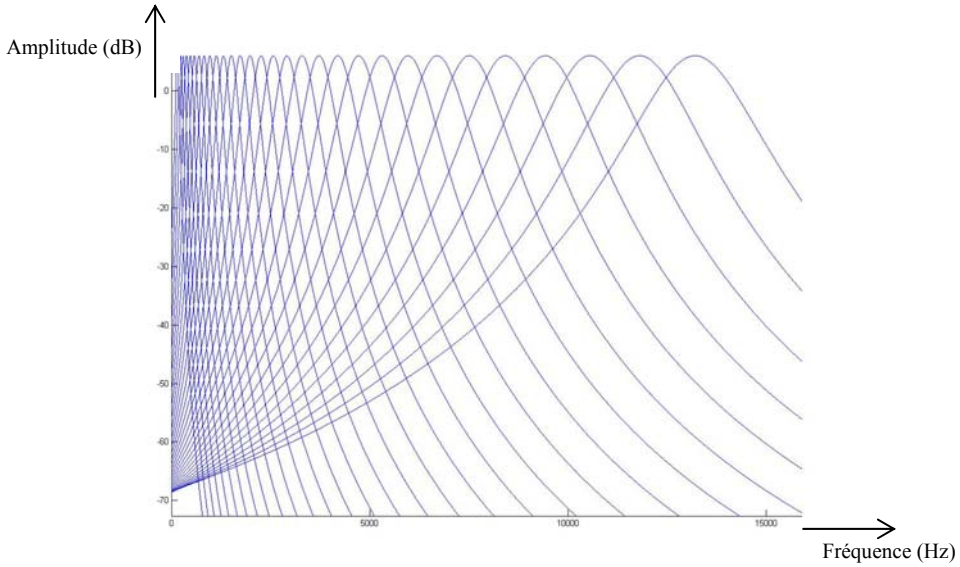


Fig. 2.8 : Réponse en fréquence d'un banc de filtre Gammatone 30 bandes, $f_{\min}$ *=70Hz,* ERB_{scale} *=1.*

II.3. La synthèse

II.3.a. Principe

Comme les réponses impulsionnelles des filtres des différents canaux du banc ont toutes une structure fine et une phase différente, la reconstruction du signal par simple

sommation des sorties du banc de filtre dégraderait le signal synthétisé. En d'autres termes, la réponse impulsionnelle du système global est trop dispersive : on va tenter de faire en sorte que la réponse temporelle globale ressemble le plus possible à un retard pur, et que la réponse fréquentielle soit la plus plate possible.

Pour synthétiser le son analysé par le banc de filtre Gammatone, Hohmann propose une méthode [3]. L'enveloppe temporelle de la réponse impulsionnelle de chaque canal est retardée d'une valeur choisie pour que la structure fine et l'enveloppe temporelle de chaque canal atteignent leurs maxima au même instant. Une sommation peut alors être effectuée, en effet la réponse impulsionnelle du système global tend vers une impulsion retardée dans la mesure où chaque canal du banc apportera une contribution positive au même instant (après un délai de groupe choisi), et que les effets des structures fines vont avoir tendance à s'annuler ailleurs.

Le retard de groupe est fixé, il ne doit pas être trop élevé pour d'évidentes raisons de performances, dans le contexte du temps réel. Pour des canaux ou la largeur de bande du filtre est étroite (les fréquences basses du banc), le retard de groupe fixé au départ est souvent plus petit que le temps nécessaire à l'enveloppe de la réponse impulsionnelle pour atteindre son maximum (temps de montée du filtre). A cause de ces canaux, la synthèse donc est quelque peu dégradée mais une solution de compromis existe : pour ces filtres où la réponse est très lente, on fait en sorte qu'à l'instant du retard de groupe soit placé un maximum local de la structure fine de la réponse impulsionnelle.

Les étapes de la synthèse sont donc les suivantes :

- Choisir un délai de groupe *t*. Pour notre modèle, *t*=4ms. Ce retard semble raisonnable d'un point de vue perceptif d'après [3], sachant que dans le cadre d'une application temps réel, nous cherchons un retard minimum.
- Pour chaque canal, regarder si le maximum de l'enveloppe de la réponse impulsionnelle temporelle tombe avant (cas n°1) ou après (cas n°2) le délai de groupe fixé et appliquer l'alignement du maximum adapté
- Calculer une somme pondérée sur tous les canaux. La pondération sert à compenser les imperfections dues aux interférences constructives au délai de groupe fixé, en particulier dans le cas n°2, pour obtenir au final une réponse en fréquence aussi plate que possible.

II.3.b. Alignement des maxima

Examinons comment sont alignées les réponses impulsionnelles des filtres dans les deux cas de figure possibles.

- Cas n°1 : La partie réelle de la réponse impulsionnelle (structure fine) est d'abord multipliée par une valeur complexe constante pour que son maximum coïncide avec le maximum de l'enveloppe, le tout est ensuite retardé par le retard de groupe choisi.

Un signal de sortie complexe $\tilde{y}_k(n)$ d'un canal k du banc de filtre est multiplié par un coefficient $\tilde{b}_k$, facteur de phase d'amplitude 1, pour obtenir le signal $\tilde{y}'_k(n)$ tel que :

$$\tilde{y}'_k(n) = \tilde{b}_k . \tilde{y}_k(n),\ 0 \le k < N_{channels}$$

$$\text{avec } \widetilde{b}_k = e^{i\phi_k},\ \phi_k = -2\pi.f_k.t_k$$

où f_k est la fréquence centrale f_c du canal k, t_k est l'instant où l'enveloppe de la réponse impulsionnelle du canal *k* est maximum.

Seule la partie réelle de ce signal est suffisante pour la synthèse, on ne garde alors que $y'_k(n) = \mathrm{Re}\left(\widetilde{y}'_k(n)\right)$.

Chaque signal réel est ensuite retardé de Δn_k points, on obtient donc $y''_k(n)$ ainsi :

$$y''_k(n) = y'_k(n - \Delta n_k),\ \text{avec}\ \Delta n_k = \mathrm{int}(\Delta t_k.f_s)\ \text{ et }\ \Delta t_k = t - t_k$$

Δt_k représente la différence de temps entre le délai de groupe désiré *t* et l'instant où l'enveloppe de la réponse impulsionnelle du filtre *k* est maximum.

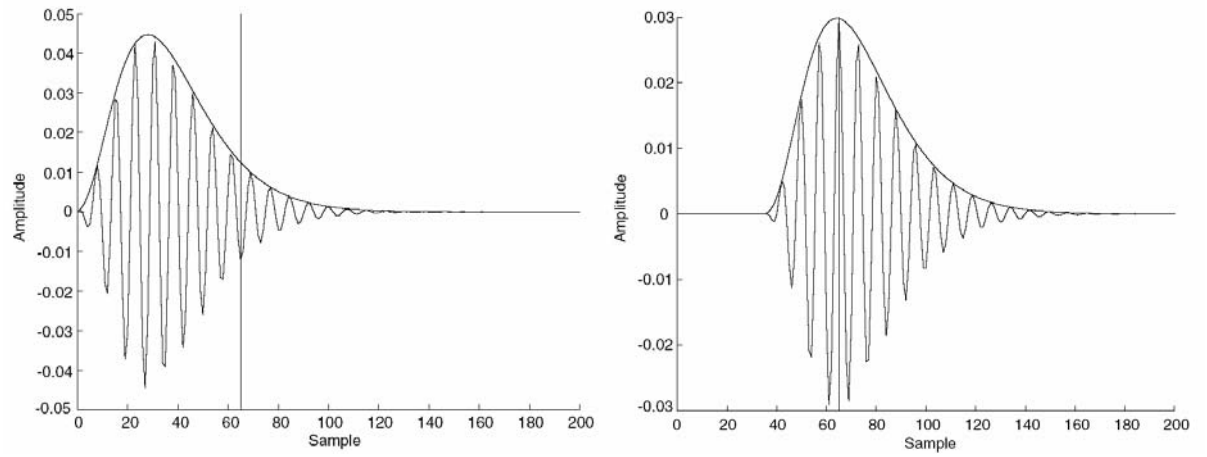


Fig. 2.9 : Partie réelle (structure fine) et valeur absolue (enveloppe) de la réponse impulsionnelle d'un filtre Gammatone avant et après alignement, cas n°1, le retard de groupe fixé à 4 ms est indiqué par un trait vertical, d'après [3].

- Cas n°2: La partie réelle de la réponse impulsionnelle est également multipliée par une valeur complexe constante pour qu'elle ait un maximum local au retard de groupe choisi.

Un signal de sortie complexe $\widetilde{y}_k(n)$ d'un canal *k* du banc de filtre est multiplié par un coefficient $\widetilde{b}_k$, facteur de phase d'amplitude 1, pour obtenir le signal $\widetilde{y}'_k(n)$ tel que :

$$\widetilde{y}'_k(n) = \widetilde{b}_k.\widetilde{y}_k(n),\ 0 \le k < K$$
$$\text{avec } \widetilde{b}_k = e^{i\phi_k},\ \phi_k = -2\pi.f_k.t$$

où f_k est la fréquence centrale f_c du canal *k*, *t* est le délai de groupe désiré.

Seule la partie réelle de ce signal est suffisante pour la synthèse, on ne garde alors que $y'_k(n) = \mathrm{Re}\left(\widetilde{y}'_k(n)\right)$.

Il n'y a pas de délai à appliquer dans ce cas, d'où : $y''_k(n) = y'_k(n)$

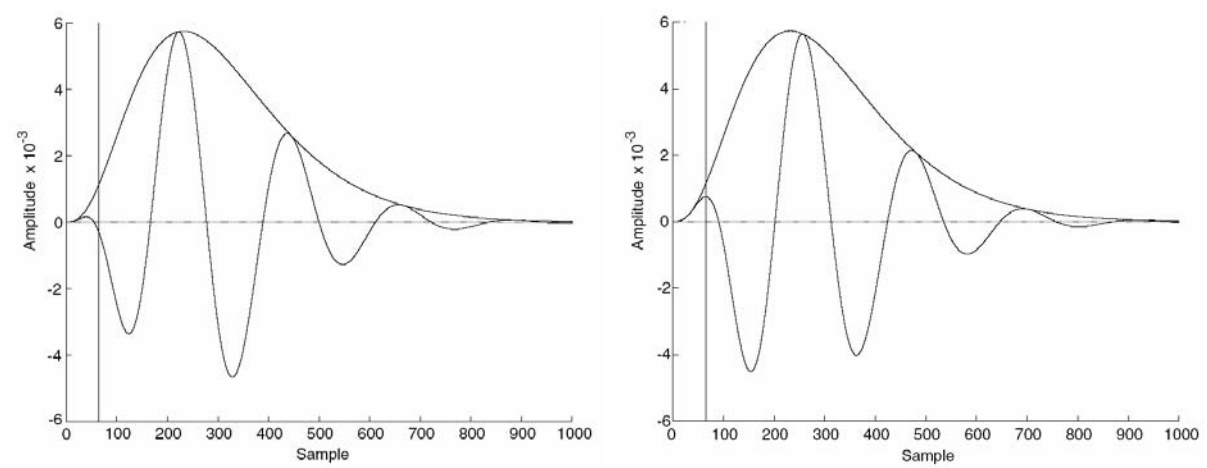


Fig. 2.10 : Partie réelle (structure fine) et valeur absolue (enveloppe) de la réponse impulsionnelle d'un filtre Gammatone avant et après alignement, cas n°2, le retard de groupe fixé à 4 ms est indiqué par un trait vertical, d'après [3].

II.3.c. Pondération

Une fois le signal de chaque canal aligné, il faut ensuite procéder à une somme pondérée des $N_{channels}$ signaux. Tous les maxima (de l'enveloppe ou locaux, selon le temps de réponse du filtre) arrivant a l'instant *t*, délai de groupe fixé, ces contributions positives déforment quelque peu la réponse globale du banc de filtre. La pondération va nous permettre de corriger ces imperfections.

Après avoir effectué les opérations d'alignement et de retard, on dispose d'un signal réel $y''_k(n)$, on va alors chercher la suite g_k telle que :

$$y'''(n) = \sum_{k=0}^{N_{channels}-1} g_k . y''_k(n)$$

où $y'''(n)$ est le signal synthétisé final.

Une implémentation Matlab programmée par Tobias Peters de l'université d'Oldenbourg, est proposée dans l'article de Hohmann [3]. Le principe de l'algorithme est de calculer le module aux fréquences centrales des fonctions de transfert une fois alignées, et de trouver, par itération, des coefficients de normalisation à la somme de ces valeurs.

Cette méthode donne des résultats satisfaisants, mais bien trop coûteuse en ressource de calcul dans le cadre d'une application temps réel. En effet, les fonctions de transfert alignées n'étant pas disponibles, il est nécessaire de calculer la transformée de Fourier des réponses impulsionnelles égalisées, opération qui demande bien trop de ressources de calcul.

La solution proposée dans le cadre de notre modèle est d'approximer la suite g_k par des fonctions simples. L'algorithme de Tobias Peters a donc été implémenté sous Matlab, et l'étude de la suite g_k a révélé plusieurs formes récurrentes pouvant être approximées à l'aide de simples segments de droites. Un travail mettant en relation les paramètres du banc d'analyse (fréquences centrales, nombre de filtres par ERB…) et la forme de la suite g_k a donc

été mené pour calculer les coefficients des droites. L'algorithme ainsi trouvé ne présentant pas d'intérêt particulier, se référer au fichier 'gammasynth.cpp' fourni en annexe pour les détails.

On voit alors figure 2.11 l'approximation de g_k par des segments de droites. Il y a certainement des erreurs par rapport au résultat de l'algorithme original, mais le plus important est de respecter les ordres de grandeur. En effet, la méthode de Tobias Peters, à cause de l'étape d'itération, ne donne pas de résultats exacts. De plus, Hohmann propose cette méthode dans [3], mais arrondi systématiquement le résultat au demi décibel près. Cette méthode de synthèse ne permet pas une reconstruction parfaite, toutes les étapes de cette synthèse ne font que limiter les distorsions sur la réponse globale du système.

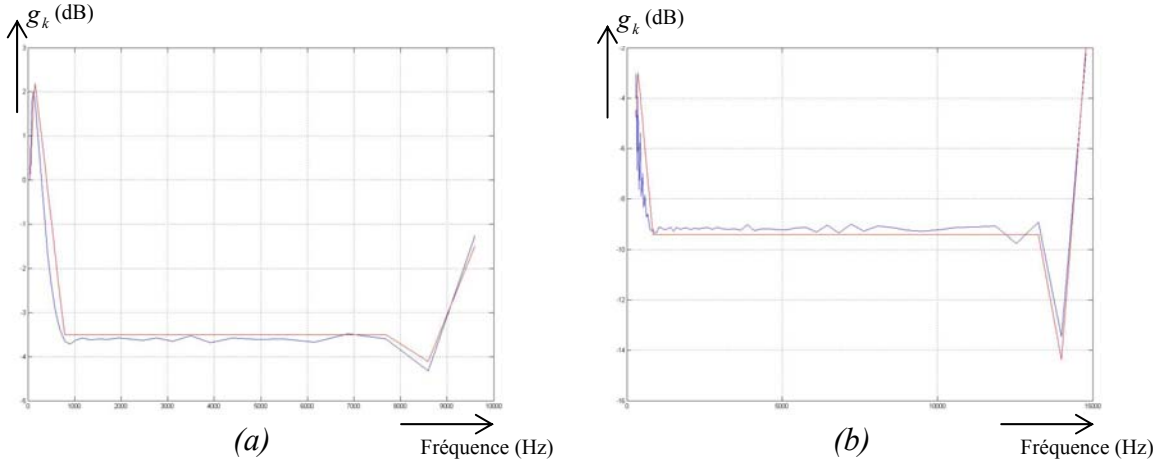
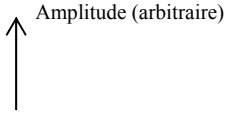


Fig. 2.11 : Suite g_k (en bleu) et approximation par segment de droites (en rouge) pour un banc de filtre de 34 bandes, fréquence basse 50Hz, ERB_{scale} =1 (a), et pour un autre de 63 bandes, fréquence basse 300Hz, ERB_{scale} =0.5, en fonction de la fréquence f_c de la bande k.

II.3.d. Résultats

Pour vérifier l'algorithme de synthèse du son après le banc de filtre, on observe figure 2.12 la réponse impulsionnelle globale du système 'banc de filtre + synthèse', et figure 2.13 sa transformée de Fourier. On remarque d'une part de nombreux parasites dans la réponse temporelle, conséquences de la reconstruction non parfaite. D'autre part, on vérifie que le retard global du système est de 241 échantillons, mais comme cette mesure a été relevée sur l'interface temps réel, nous discuterons de ces problèmes de retard et de latence à la section III.2.b.



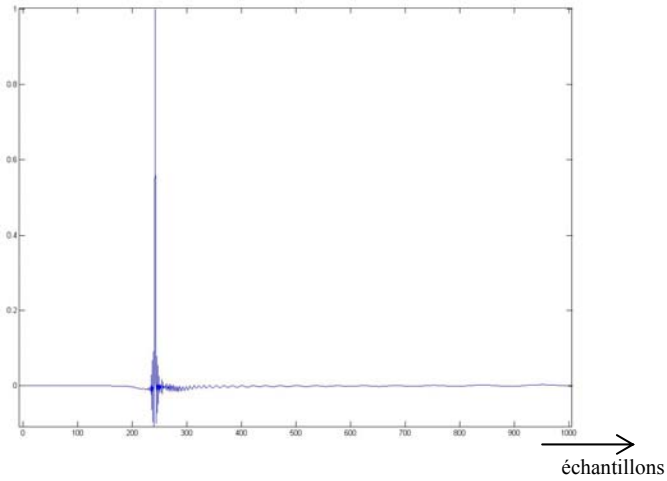


Fig. 2.12 : Réponse impulsionnelle du système global banc de filtre et banc d´analyse. Les paramètres sont : 40 bandes, fréquence basse 50Hz, ERB_{scale} =1.

La réponse en fréquence montre de nombreuses variations dans la bande passante, mais celles-ci n'ont pas un impact majeur dans la perception du son synthétisé, en effet, des tests d'écoute ont été menés [3] et révèlent que la différence subjective entre le son original et le son synthétisé, pour une synthèse avec un retard de 4ms, est très faible. Les distorsions de la reconstruction sont à peine audibles. Pour des retards supérieurs, les distorsions sont indétectables, mais ce paramètre a été fixé à 4ms dans notre modèle et n'est pas modifiable.

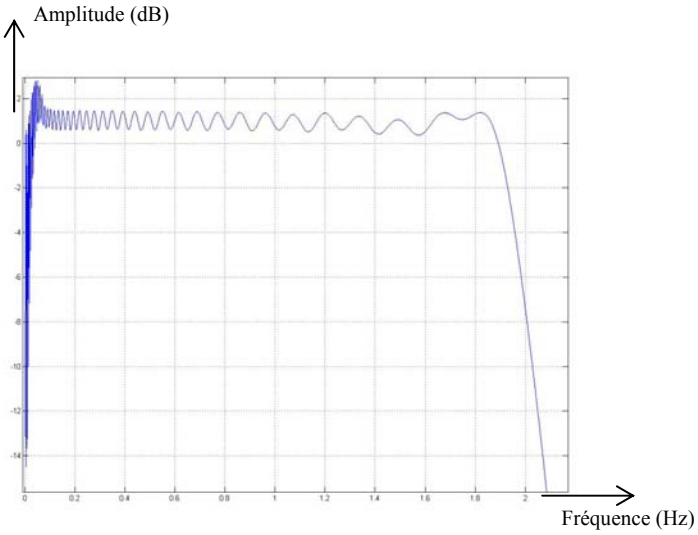


Fig. 2.13 : Réponse fréquentielle du système global banc de filtre et banc d'analyse. Les paramètres sont : 40 bandes, fréquence basse 50Hz, ERB_{scale} =1.

Cette méthode de synthèse est très pratique pour effectuer des traitements sonores. En effet, le son est décomposé avec un souci perceptif par le banc de filtre, il peut ensuite être manipulé, puis il est reconstruit, permettant une écoute. La synthèse quant à elle est simplement un outil de traitement du signal permettant une resynthèse la plus exacte possible d'un signal modifié après le banc de filtre Gammatone en l'absence de modification, en d'autre terme, elle rend l'analyse par banc de filtres Gammatones inversible.

Chapitre III.
Implémentation temps réel

III.1. Le logiciel Pure Data

Dans cette première partie nous allons présenter brièvement le logiciel Pure Data en définissant les termes les plus courants relatifs à son utilisation et dont nous aurons besoin par la suite.

III.1.a. Introduction

Pure Data (PD) est un environnement de programmation graphique temps réel pour l'audio, la vidéo et le traitement graphique. C'est un logiciel libre écrit et mis a jour par Miller Puckette et par les nombreux développeurs qui composent la très active communauté créée autour de ce logiciel.

Pour réaliser un traitement (nous nous limiterons au cas de l'audio), il faut relier des boites, chacune réalisant une fonction. Chacune de ces boites est appelé objet, un assemblage d'objets en vue d'un traitement est appelé patch. Les entrées sont sur le haut de l'objet, les sorties en bas. Il y a à notre disposition d'autres éléments (nombres, messages, tableaux…) permettant de contrôler et de paramétrer ces objets.

Il est également possible de créer ses propres objets en C [5], ou en C++ grâce à la couche 'Flext', et de les intégrer à ses patches. Plusieurs objets peuvent être regroupés dans une librairie.

Un exemple est montré figure 3.1, il s'agit d'un patch très simple avec un premier objet oscillateur, qui génère un sinus à 1000Hz, que l'on divise par un nombre, contrôlable en temps réel par l'utilisateur, par l'intermédiaire d'un objet division, et qui est envoyé vers la sortie de la carte son de l'ordinateur grâce a un dernier objet.

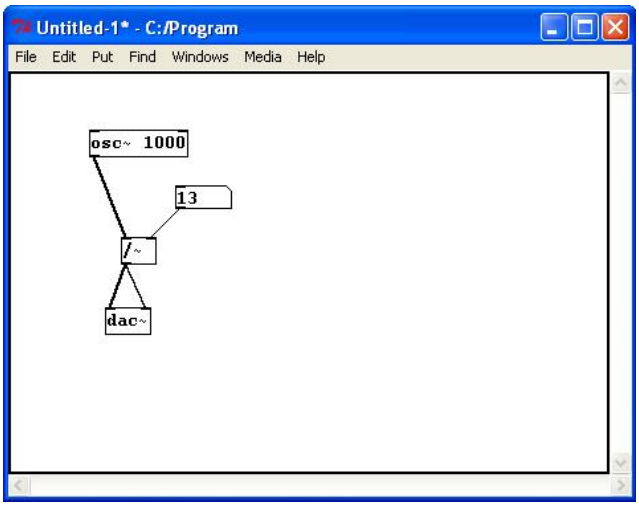


Fig. 3.1 : Exemple de patch simple avec 3 objets et un nombre.

III.1.b. Fonctionnement

A l'allumage du logiciel, une première fenêtre principale s'ouvre : c'est la console principale qui diffuse les messages du logiciel et à partir de laquelle sont ouvert les patches. Chaque patch est en fait une nouvelle fenêtre qui dialogue avec la console.

Dans le logiciel Pure Data, les éléments les plus importants sont les objets. Ils réalisent les traitements des données. Il existe deux sortes d'objet : les objets 'contrôle' et les objets 'signal' (qui ont un nom d'appel suivi de '~'). Ces objets sont reliés entre eux par des traits, les liaisons épaisses correspondent à un flux de signal, alors que les liaisons fines sont attribués aux les données de contrôle. Ainsi, sur une liaison de contrôle, on peut véhiculer un nombre, un symbole, ou une liste de nombre et de symboles. Certaines listes sont appelées messages, elles permettent de contrôler des objets. Par exemple, 'bang' est le message standard de déclenchement d'un évènement. Les liaisons signal en traits larges représente la circulation du flux sans interruption, quand l'audio est allumé sur la console principale.

La plupart des objets utilisés font partie intégrante du logiciel de base, ils sont dits internes. Il est possible de créer ses propres objets en C ou en C++ et de les inclure dans le logiciel, ceux-la sont alors dit externes.

III.2. Choix de programmation

III.2.a. Banc d'analyse/synthèse et multiplexage

Le modèle auditif est implémenté dans l'interface sous forme d'objets externes : un objet réalise la décomposition du signal entrant par le banc de filtre, un autre fait la synthèse. Les paramètres de contrôle disponibles pour la génération du banc de filtre sont le nombre de bande, la fréquence centrale la plus basse et l'inverse du nombre de filtre par ERB. Une fois ces paramètres choisis, un message 'bang' génère ensuite le banc de filtre. Le signal entrant dans le banc de filtre peut alors être traité.

L'enjeu majeur pour notre application est celui du format de sortie. En effet, le banc de filtre génère plusieurs signaux (autant que de bandes dans le banc de filtre) et il faut un moyen de les représenter de façon raisonnable. Il semble difficile de gérer un patch avec un banc de filtre de plusieurs dizaines de sortie. Par ailleurs, il faut un accès temps réel au

nombre de canaux, et il est impossible de changer le nombre d'entrées/sorties d'un objet de façon dynamique. L'idée est donc la suivante : utiliser un format où les flux audio sont multiplexés dans une seule liaison.

Un tel format n'étant pas disponible directement dans PD, un format de transfert multiplexé a donc été créé. Il correspond en fait à un flux de listes de nombres flottants, chacune d'entre elles étant une concaténation de tous les flux audio en sortie du banc de filtre. Afin de permettre bonne modularité (nombre de canaux, nombre de points par canal) et d'éviter les erreurs de transfert, une entête est ajoutée à chaque liste. Elle donne les informations nécessaires à l'objet dans lequel elle rentre pour pouvoir être interprétée.

L'objet réalisant la synthèse est lié aux paramètres d'un banc de filtre. Il reçoit donc dans une entrée les parties réelles multiplexés du signal analysé par le banc de filtre, et dans une autre les parties imaginaires. Le paramétrage est effectué par une troisième entrée, directement depuis la troisième sortie du banc de filtre. La sortie est au format signal standard de PD. Notons que cet objet est équipé d'une version de synthèse simplifiée dans le cas ou seule la partie réelle du signal est disponible.

La figure 3.2 révèle les choix qui ont été faits. Le banc de filtre est paramétré à l'aide de messages, et comporte trois sorties. La première est un multiplexage des parties réelles du signal filtré par chacun des canaux du banc de filtre. Même chose pour la seconde sortie avec la partie imaginaire. La dernière sortie sert à paramétrer l'objet réalisant la synthèse, ce dernier ayant besoin de plusieurs éléments propres au banc de filtre.

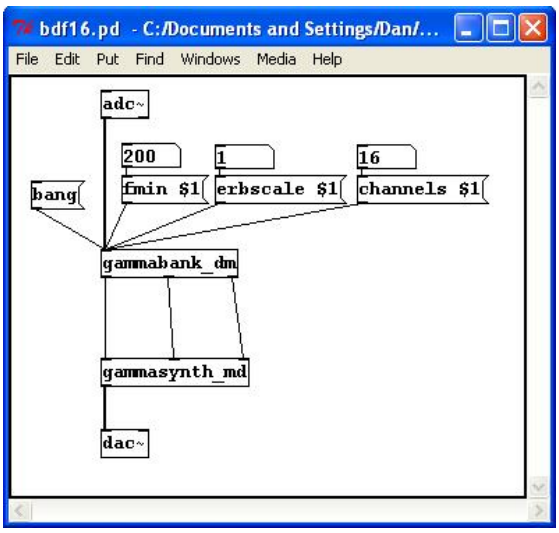


Fig. 3.2 : Patch d'analyse et synthèse d'un son provenant de l'entrée de la carte son par un banc de filtre Gammatone de 16 bandes, fréquence basse 200Hz, 1 filtre par ERB.

III.2.b.Performances

Nous avons vu dans la section II.3 que le banc de synthèse est construit avec un retard de 4ms. L'utilisation de Pure Data ajoute une latence supplémentaire. D'abord, pour des raisons techniques liées à l'ordinateur, PD introduit une latence entre l'entrée et la sortie de la carte son, déterminée par l'utilisateur mais fixée à 70ms (trop mauvaises performances pour des valeurs inférieures). Enfin, les flux sonores sont en réalité traités par bloc (64

échantillons, soit 1,5 ms à 44,1kHz), il y a donc un bloc dit ‘de bufferisation’ de retard au sein même des traitement effectués par PD.

Si on reprend la réponse globale du système banc de filtre et synthèse présentée figure 2.12 dans la section II.3.d, on a un retard de construction de la synthèse de 4ms, soit 177 échantillons à 44100Hz, auquel il faut ajouter la bufferisation de 64 échantillons, soit 241 au total.

En ce qui concerne les performances du modèle, avec notre matériel (PC biprocesseur Intel Pentium 4 à 3Ghz), le temps réel avec la latence de 70 ms (PC et interfaces) + 1,5 ms (bufferisation logicielle) + 4ms (propres à la synthèse) est conservé jusqu'à 90 bandes pour le banc de filtre pour le patch présenté figure 3.2.

III.2.c. La bibliothèque ‘Audition’

Ce format de données multiplexées va nous être très pratique pour simplifier les patches, en particulier pour limiter le nombre de liaisons et en changer le nombre dynamiquement. Afin de se donner une idée, pour un banc de filtre de 16 bandes, il aurait fallu au moins 32 connections pour une analyse et synthèse simple. La figure 3.3 nous montre cet exemple, en démultiplexant et en remultiplexant les signaux calculés. Dans ce cas, il est encore possible de représenter les connexions, mais pour un banc de filtre de 80 bandes, le patch devient totalement illisible.

Pour conserver l’intérêt du multiplexage, il est nécessaire de développer plusieurs outils standard de traitement adaptés à ce format de données. En effet, pour pouvoir effectuer un traitement entre le banc d’analyse et le banc de synthèse avec les seuls objets inclus avec PD, il faut démultiplexer, faire le traitement dans chaque bande, puis multiplexer à nouveau, comme sur la figure 3.3, le nouveau format n’a alors plus d’intérêt particulier.

Pour permettre un traitement facile de données multiplexées, une bibliothèque (nommée ‘Audition Library’) est développée. Cette bibliothèque est écrite en C++ à l’aide de la couche logicielle Flext. Les bancs d’analyse et de synthèse sont bien sûr inclus, ainsi que plusieurs objets de traitement de base (opérateurs algébriques et logiques simples, outils de multiplexage/démultiplexage, filtres simples). Une liste exhaustive des objets de la bibliothèque est visible dans le fichier ‘libaudition.cpp’ fourni en annexe. Une topologie est adoptée, le nom de chaque objet de la bibliothèque a un suffixe de type _mm, _md, _dm, ou _m. Ils correspondent aux formats d’entrée et de sortie de l’objet, ‘m’ pour multiplexé, ‘d’ pour démultiplexé. Par exemple, le banc de filtre Gammatone a pour suffixe _dm puisqu’il traite un seul flux par une entrée signal classique, et délivre des signaux multiplexés en sortie. Les objets _m prennent des données multiplexées en entrée et ne restituent pas de flux en sortie.

L’innovation de cette bibliothèque est le format multiplexé des flux audio. Même s’il a été pensé pour l’utilisation du banc de filtre Gammatone, rien n’empêche de s’en servir dans d’autres applications. En effet, la bibliothèque comporte plusieurs outils de multiplexage et de traitement des flux multiplexés. L’extension de cette bibliothèque par de nouveaux objets est bien entendu possible, et si la communauté des développeurs et des utilisateurs de Pure Data y trouve un intérêt, elle sera à prévoir.

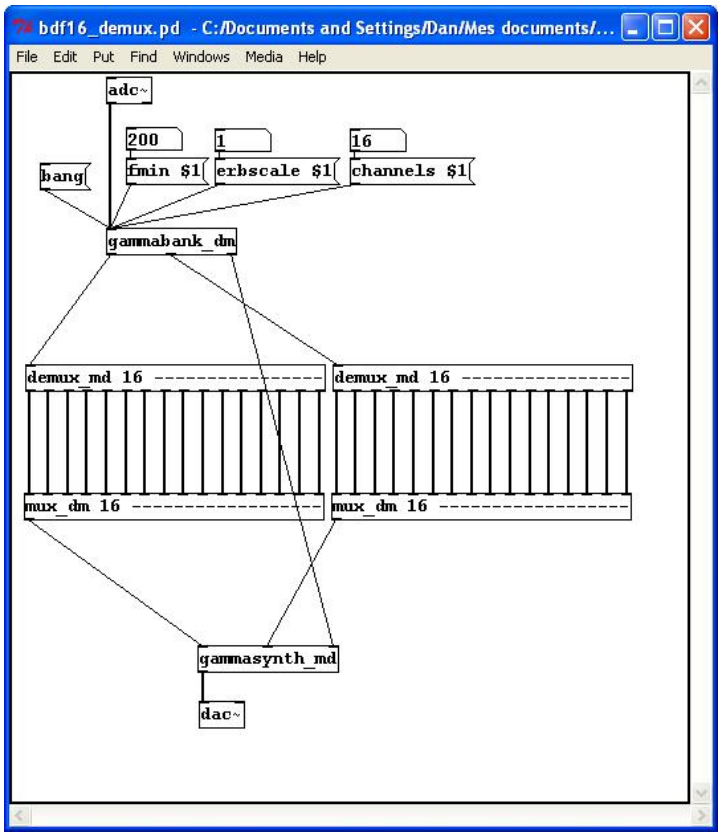


Fig. 3.3 : Patch d'analyse et synthèse d'un son provenant de l'entrée de la carte son par un banc de filtre Gammatone de 16 bandes, fréquence basse 200Hz, 1 filtre par ERB. Les sorties du banc de filtre son démultiplexées puis multiplexées à nouveau.

Chapitre IV.
Utilisation du modèle

Nous avons notre modèle implémenté sur une plateforme temps réel. Voyons maintenant quelques utilisations de cet outil.

IV.1. Visualisation

IV.1.a. Intérêts

La visualisation des signaux acoustiques est très répandue, de nombreux appareils grand public de diffusion du son sont dotés de systèmes de représentation. La plupart du temps, il s'agit de visualisation temps/fréquence de type transformée de Fourier à court terme (ou spectrogramme). Ces représentations ne sont pas mauvaises, au contraire puisqu'elles correspondent à une réalité physique, mais il faut savoir les interpréter par rapport au son entendu.

Nous avons pu voir que cette réalité est perçue différemment par nos oreilles. En particulier, la représentation par spectrogramme est à pas de représentation (fréquentiels et temporels) fixe, choisi arbitrairement, ce qui n'est pas le cas du système auditif : une approche à l'aide de notre modèle permettrait une représentation ou la précision diminue à mesure que la fréquence augmente (voir section II.1), sachant que le pas respecte des données perceptives.

Cette sélectivité des filtres est donc importante. En effet, plusieurs phénomènes (comme par exemple les battements) peuvent être perçus par l'oreille mais en réalité non visualisés sur un spectrogramme, ce dernier étant trop précis. En effet dans le cas de la somme de deux fréquences proches, des battements peuvent être clairement discernés à l'oreille sans qu'ils n'apparaîtrent sur le spectrogramme. Une représentation basée sur le banc de filtre auditif reproduit les effets du dispositif perceptif en visant ne montrer que ce qui est perçu, et tend à se rapprocher de la notion de 'ce que l'on voit est ce que l'on entend'.

IV.1.b. Principe

Plusieurs représentations sont possibles en utilisant le banc de filtre Gammatone. Nous allons présenter une méthode possible, mise en évidence par un patch.

L'idée est de visualiser dans un graphique ayant en abscisse un nombre de point égal au nombre de canaux du banc le son analysé par le banc de filtre, chaque point du graphique est alors contrôlé par le flux dans un canal. Cette représentation va varier en fonction de l'amplitude du signal dans chaque canal, elle offre des information spectrales évoluant au cours du temps, tout en respectant les patterns d'excitation du système auditif.

La figure 4.1 est une capture d'écran du patch utilisé pour cette visualisation. Le signal représenté est la somme d'un sinus pur à 1,5kHz et d'un bruit à bande étroite autour de 1kHz.

Les paramètres du banc de filtre sont : 85 bandes, 300Hz pour la fréquence basse, et 2.5 filtres par ERB. Plusieurs étages supplémentaires sont nécessaires pour permettre des bonnes performances :

- une valeur absolue correspondant au mouvement unilatéral des cellules ciliées (une rectification demi période semble mieux adaptée, mais d'un point de vue signal, une valeur absolue donne le même résultat)
- un filtre passe bas (réglé à 3Hz), permettant de lisser la courbe obtenue, et agissant comme un intégrateur
- une compression par loi d'exposant, accentuant les variations,

L'objet 'snapshot_m' renvoie la valeur de chaque canal à l'instant où il reçoit un message 'bang'.

On remarque sur l'exemple présenté que cette représentation diffère des autres visualisations spectrales classiques, on distingue en effet sur celle-ci le pattern d'excitation caractéristique du système auditif. On se rapproche de ce que l'on a pu observé figure 2.2 et 2.3.

Plusieurs difficultés dues à l'outil graphique de PD nous empêchent de placer une échelle.

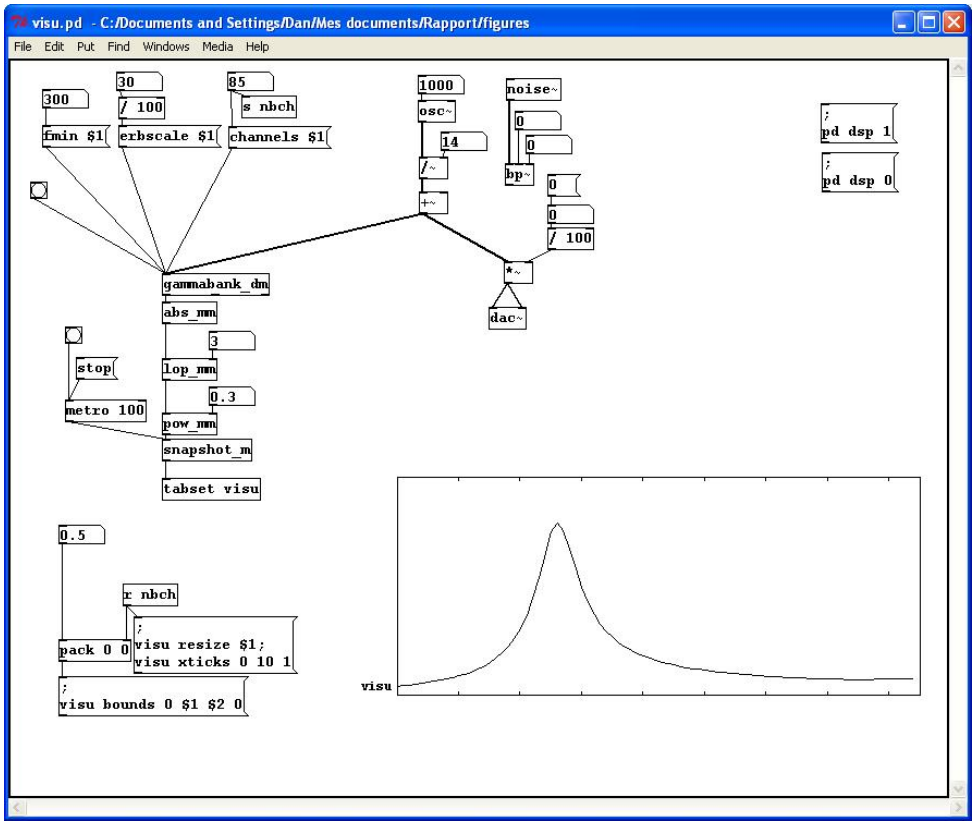


Fig. 4.1 : Patch réalisant la visualisation du signal entrant (fréquence pure à 1,5kHz) à l'aide du filtre Gammatone.

La forme du pattern visualisé semble raisonnable et en rapport avec les observations faites à la section II.1.a. Le pattern à l'air un peu dissymétrique (pente un peu plus raide côté des basses fréquences) ce qui est normal. En effet, même si les filtres sont symétriques, ils sont plus étroits et plus resserrés dans les basses fréquences, une fréquence pure excite donc plus particulièrement un filtre que ces voisins, alors qu'en hautes fréquences, comme les

filtres sont larges, une excitation par une fréquence pure va plus facilement se répercuter dans les filtres voisins du filtre central.

IV.1.c. Expériences – Résultats

Afin de vérifier que cette visualisation correspond bien à nos attentes, nous allons tenter de reconstituer le phénomène de masquage présenté figure 2.3. A partir du patch de visualisation de la figure 4.1, on diminue lentement la fréquence du sinus pur, tout en écoutant le signal entrant. Le résultat de cette expérience est montré figure 4.2. On reconnaît d'une part distinctement la forme du pattern d'excitation du système auditif. On visualise d'autre part, le phénomène de masquage simultané : la fréquence pure est masquée par le bruit à bande étroite. Entre 1.1kHz et 900Hz, le sinus pur disparaît dans le pattern du bruit (centré sur 1kHz). Une autre expérience va mettre en évidence la dissymétrie du pattern.

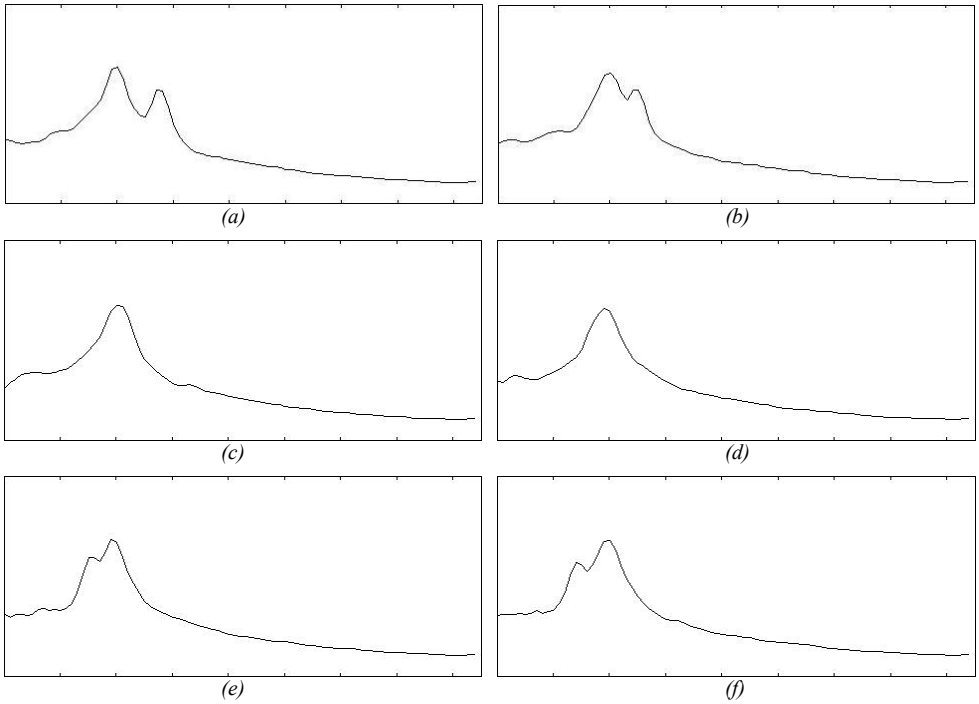


Fig. 4.2 : Représentations à l'aide du banc de filtre Gammatone d'un signal composé de la somme d'un bruit à bande étroite centré à 1kHz et d'une fréquence pure à 1,5kHz(a), 1,3kHz(b), 1,1kHz(c), 900Hz(d), 800Hz(e) et 750Hz(f).

Cette expérience a bien entendu été faite en écoutant le son résultant afin de vérifier que le masquage a effectivement bien lieu. On vérifie de façon très informelle que la représentation répond à priori à nos attentes en terme de pattern de masquage. Cependant, si on veut vérifier ces patterns avec les résultats théoriques présentés figure 4.3, il nous manque les amplitudes des excitations, grandeurs difficiles à déterminer avec l'interface temps réel. En revanche, nous pouvons vérifier la forme des patterns de masquage visualisés, en particulier la dissymétrie : la pente du pattern est plus raide du coté des basses fréquences que des hautes. Une seconde expérience va nous permettre d'examiner cette tendance. Il s'agit de présenter deux sons cibles qui sont placés à égale distance en fréquence de la fréquence

centrale du masqueur, de part et d'autre, et de vérifier que le son cible aigu est masqué et que le grave ne l'est pas.

La figure 4.3 illustre cette expérience, on vérifie (même si les tendances sont minimes) que le pattern de masquage visualisé à l'aide de cette représentation utilisant le banc de filtre Gammatone est bien asymétrique, puisqu'on présente le son cible à 80Hz de différence avec la fréquence centrale du masqueur, et on distingue mieux la cible quand celle-ci est plus grave que le masqueur.

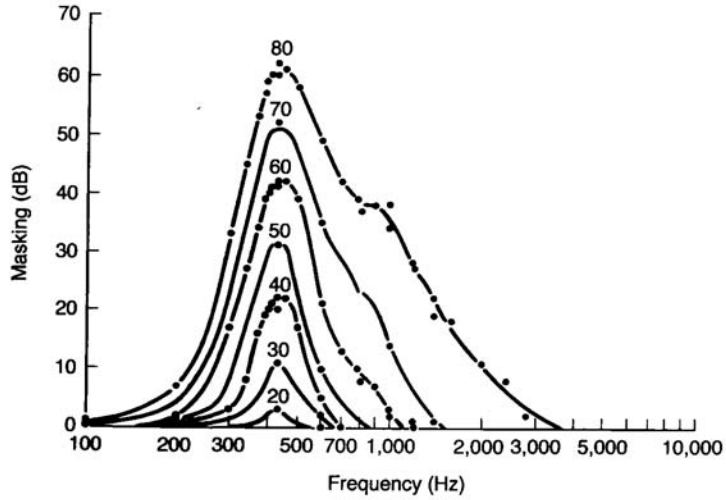


Fig. 4.3 : Patterns de masquage pour un bruit à bande étroite centré à 410Hz, d'après [15]. Image de [13].

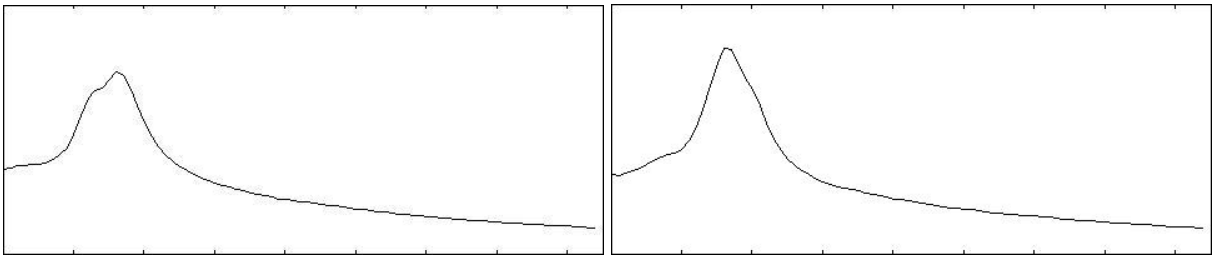


Fig. 4.4 : Représentations à l'aide du banc de filtre Gammatone d'un signal composé de la somme d'un bruit à bande étroite centré à 500Hz et d'une fréquence pure à 420Hz(a), et 580Hz(b).

IV.2. Simulation d'implant cochléaire

Le banc de filtre Gammatone, associé à l'algorithme de synthèse permet d'effectuer des traitements audio en temps réel. Le son est analysé et décomposé par bande de fréquence, modifié, puis resynthétisé. Nous proposons ici de simuler l'effet d'un implant cochléaire à l'aide de notre modèle auditif.

IV.2.a. Qu'est ce qu'un implant cochléaire ?

L'implant cochléaire est un appareil permettant aux personnes atteintes de surdité profonde de retrouver une audition partielle. Le principe de ce dispositif est d'exciter directement le nerf auditif à l'aide d'électrodes placées dans la cochlée par une opération chirurgicale. Une partie de l'implant est situé en dehors du conduit auditif (le plus souvent derrière l'oreille), c'est là que se trouve le microphone et le processeur réalisant les calculs nécessaires à la production du signal excitateur envoyé aux électrodes.

Les électrodes placées dans la cochlée réalisent une excitation ponctuelle le long de la membrane basilaire, il y a donc un nombre fini de points d'excitation pour un implant donné. Nous avons vu précédemment que la cochlée agit comme un banc de filtre passe bande : chaque électrode excite donc un de ces filtres. Avec les technologies actuelles, les implants cochléaires vont de 4 à 24 bandes.

IV.2.b. Détails de la simulation

Intéressons nous plus précisément au signal excitateur. Plusieurs approches sont adoptées par les différents fabricants d'implants, nous allons simuler une d'entre elle nommée Continuous Interleaved Sampling (CIS).

Le nombre de points excités étant limité, et afin de fournir suffisamment d'énergie dans chaque bande, le signal extérieur capté par le microphone passe dans un vocodeur. En effet, pour chaque bande à exciter, le signal filtré module en amplitude un bruit de même largeur de bande. La figure 4.5 est le patch utilisé pour ce traitement temps réel : le signal entrant et le bruit sont d'abord découpés en 16 bandes de fréquences grâce au banc de filtre Gammatone, puis on récupère l'enveloppe du signal par redressement double alternance et filtrage passe bas, ensuite on multiplie cette enveloppe par le bruit pour réaliser la modulation dans chaque bande, enfin on synthétise le signal obtenu. Notons que seule la partie réelle du signal est utilisée, en effet le son résultant ayant la structure fine d'un bruit, il est hors de propos de soigner la reconstruction.

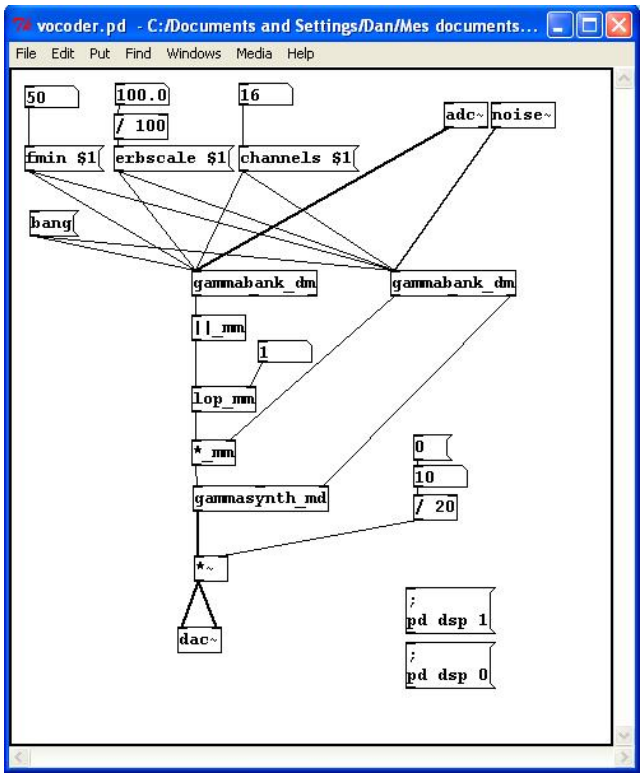


Fig. 4.5 : Patch réalisant une simulation d'implant cochléaire.

Le son obtenu correspond à ce que peut entendre une personne implantée. En effet, les informations relatives à la structure fine et au spectre du son original sont perdues, mais l'enveloppe, et donc les informations basses fréquences nécessaires à la compréhension de la parole, est conservée.

IV.2.c. Discussion

Il semble clair que cette simulation ne correspond pas à un implant cochléaire existant, une particularité des vrais implants cochléaires n'étant pas prise en compte. En effet, pour des raisons technologiques, toutes les bandes ne sont pas excitées simultanément. En fonction des modèles d'implant cochléaires, un nombre limité d'électrode, celles où l'énergie est la plus importante, émettent dans la cochlée. Dans notre simulation, toutes les bandes sont sollicitées à chaque instant. Limiter le nombre d'électrodes utilisées simultanément est une amélioration possible de cette simulation.

D'autres algorithmes que CIS sont utilisés dans plusieurs implants cochléaire. Il est ainsi envisageable d'améliorer la simulation en proposant ces autres méthodes. En particulier, certains implants modulent un train d'impulsion par l'enveloppe du signal extérieur, d'autres associent une détection de formants de la voix.

Il demeure une question importante : pourquoi avoir utilisé le modèle auditif pour cette simulation ? Les électrodes disposées le long du nerf auditif d'un sujet sourd implanté ne réalisent pas exactement une excitation ponctuelle, le contact a bien entendu une certaine largeur. Dans un véritable implant, c'est une fois la largeur de ces contacts connus que les paramètres du banc de filtre (qui n'est pas un banc de filtre Gammatone) sont déterminés. Nous réalisons ici une simulation destinée à des personnes normo entendantes, l'utilisation du

banc de filtre Gammatone se justifie alors puisqu'il n'y a pas d'électrodes implantées, les largeurs de bande du banc de filtre vont être celles du système auditif. En d'autres termes, le banc de filtre Gammatone minimise les modifications introduites par l'oreille d'un normo-entendant sur les simulations que nous réalisons.

IV.3. Synthèse croisée

Nous proposons un autre traitement audio réalisé à l'aide du banc de filtre Gammatone, il s'agit d'une synthèse de son dite croisée.

IV.3.a. Introduction

La synthèse croisée est le mélange de deux sons pour n'en faire plus qu'un seul. Ce type de traitement audio s'il est fait en temps réel, peut avoir des applications musicales puisqu'il offre la possibilité de faire des sons originaux.

Il s'agit, à partir de deux sons différents, de synthétiser un nouveau son ayant la structure fine de l'un et l'enveloppe de l'autre. Comme ce type de traitement nécessite un découpage en bandes de fréquence, nous avons réalisé ce procédé à l'aide du banc de filtre Gammatone afin de respecter les données de la perception.

Ce type de traitement audio est très répandu, et de nombreuses méthodes autres que celle que nous avons retenue sont utilisées. Notons en particulier que de nombreux vocodeurs destinés à des applications musicales récupèrent l'enveloppe de la voix par diverses techniques, et modulent directement le son de l'instrument de musique désiré. Ces vocodeurs existent en version analogique, la décomposition en banc de filtre réalisée est parfois sommaire pour ce type d'appareil (de l'ordre de 3 à 5 bandes de fréquence seulement) mais suffisante du point de vue du résultat.

D'autres méthodes ont été retenues pour certains vocodeurs numériques. Ainsi, plusieurs vocodeurs, dont fait partie le célèbre GSM présent dans les téléphones cellulaires, fonctionnent à l'aide d'un algorithme reprenant des modèles source – filtre. Le conduit vocal est modélisé par un filtre linéaire adaptatif, calculé à l'aide de la voix, et le second signal est simplement placé en entrée de ce filtre. Dans le cas du GSM, ce signal placé en entrée du filtre est choisi pour recréer au mieux le signal de parole, mais de nombreux vocodeurs numériques utilisent cette même approche source – filtre pour des applications musicales.

Les approches présentées s'appliquent surtout pour les vocodeurs, c'est-à-dire que la voix sert de signal modulant. La synthèse croisée que nous allons présenter n'utilise aucun à priori sur les signaux à croiser.

IV.3.b. Principe

Précisons la méthode pour extraire la structure fine d'un son. Une fois le son décomposé par le banc de filtre, dans chaque canal k de fréquence centrale $f_{c,k}$, on récupère d'abord la phase $\Phi(n)$ du son $\tilde{y}_k(n)$ ainsi :

$$\Phi(n) = \tan^{-1}\left(\frac{\mathrm{Re}(\tilde{y}_k(n))}{\mathrm{Im}(\tilde{y}_k(n))}\right),$$

pour avoir dans chaque canal k le signal $z_k(n)$ ne gardant que la structure fine :

$$z_k(n) = \cos\left(2.\pi.f_{c,k} + \Phi(n)\right)$$

Le signal $z_k(n)$ ne porte aucune information d'amplitude, c'est le contrôle de la phase qui permet de récupérer la structure fine.

On récupère l'enveloppe de l'autre son en calculant le module du signal complexe dans chaque bande, à laquelle on ajoute un filtrage passe bas pour lisser le résultat.

Une fois la structure fine et l'amplitude déterminées, il ne reste plus qu'à multiplier dans chaque bande les cosinus modulés en phases $z_k(n)$ par les enveloppes, et de synthétiser le résultat. La figure 4.6 présente le patch conçu pour cette synthèse croisée. On peut noter la présence de l'objet 'ccarrier_mm' qui réalise la détection de la structure fine, fait avec trois entrées puisqu'en plus des parties réelles et imaginaires du signal, il faut lui communiquer les fréquences centrales des canaux du banc de filtre.

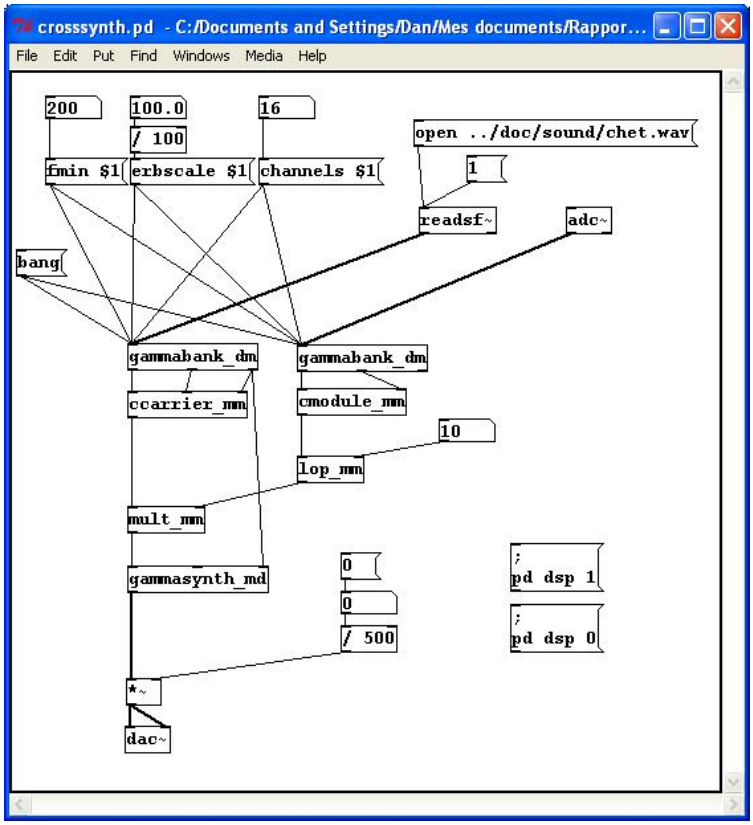


Fig. 4.6 : Patch réalisant une synthèse croisée entre la voix (venant d'un microphone branché sur l'entrée de la carte son de l'ordinateur) et un extrait de musique.

IV.3.c. Conclusions

Ce procédé est le plus fréquemment utilisé comme un vocodeur, en récupérant la structure fine d'un son complexe (extrait de musique par exemple) et l'enveloppe de la voix. Smith *et al.* [21] soulignent d'ailleurs l'intérêt pour la voix de séparer structure fine et enveloppe : l'enveloppe permet de reconnaître la parole, et la structure fine la hauteur. Ils utilisent notamment la transformée de Hilbert pour séparer structure fine et enveloppe, ce procédé de calcule découle directement d'une décomposition de type Gammatone.

Le patch tel qu'il est présenté réalise la synthèse croisée de cette façon là, mais rien n'empêche de changer les sources.

IV.4. Sonie

Voyons maintenant comment notre modèle peut nous aider à calculer des attributs perceptifs. Dans un premier temps, intéressons nous à la détermination de la sonie, grandeur perceptive liée à la puissance du son.

IV.4.a. Définition

L'oreille humaine peut détecter une large gamme d'intensité, en effet, les sons les plus intenses que nous pouvons détecter sans douleur ont un niveau de 120dB de plus que le plus faible son détectable. La sonie est l'attribut du système auditif qui permet de classer sur une échelle cette gamme d'intensité, du son le plus faible jusqu'au plus puissant. Il est important d'ajouter que la sonie est une quantité psychophysique donc subjective, par contraste avec l'intensité du son qui est une grandeur physique se mesurant avec des capteurs de pression par exemple.

La sonie est différente de l'intensité sur un premier point : l'oreille externe et moyenne ne transmet pas toutes les fréquences avec la même puissance, en d'autres termes, l'oreille n'a pas une réponse en fréquence plate. En effet, avec sa géométrie compliquée, l'ensemble composé de l'oreille externe et moyenne a une fonction de transfert faisant apparaître un large pic aux alentours de 3kHz. Cette fonction de transfert peut être mesurée à l'aide d'un micro placé dans le canal auditif, on relève le transfert montré figure 4.7.

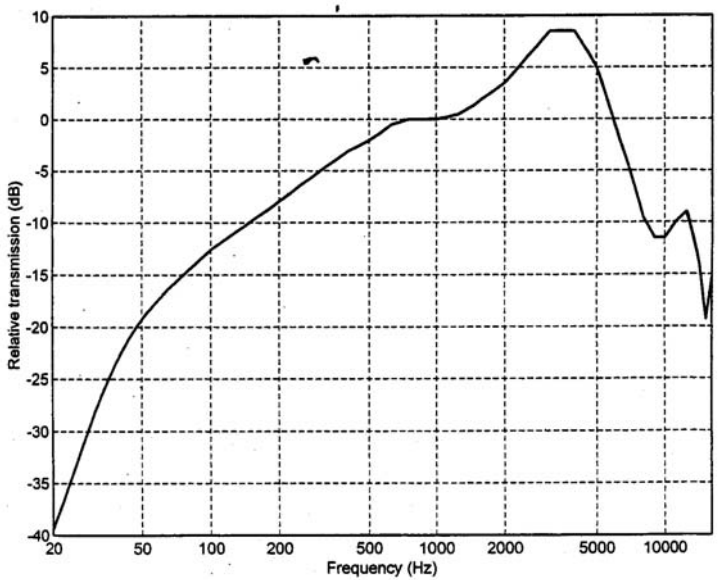


Fig. 4.7 : Fonction de transfert de l'ensemble composé de oreille externe et de l'oreille moyenne, d'après [14].

Pour mettre en évidence cette dépendance de la fréquence dans la perception de l'intensité d'un son, on demande à des sujets normo entendants d'ajuster le niveau d'un son pour qu'il semble être à la même puissance qu'un premier son étalon (à une fréquence différente). On vérifie alors que, pour la même sensation de puissance, l'intensité du son est différente selon la fréquence. Cette expérience, dont le résultat est présenté figure 4.8, permet de définir l'échelle des 'phones'. La sonie en phones d'une fréquence pure à 1kHz est égal à son intensité en dB (étalonnage de l'échelle) ; la sonie en phones d'une fréquence pure quelconque est celui d'une fréquence pure à 1kHz qui produit le même effet perceptif d'intensité.

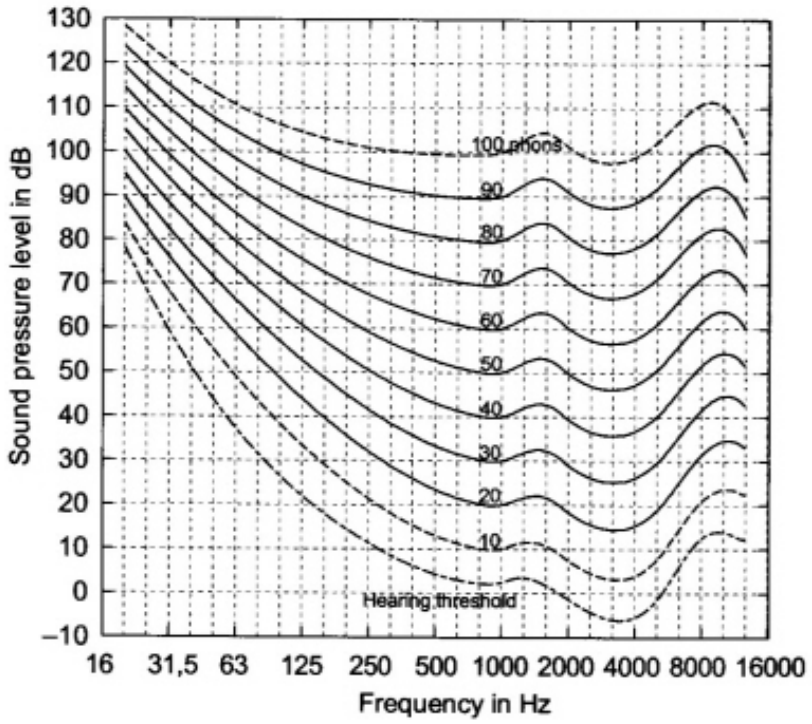


Fig. 4.8 : Courbes isophones en fonction de la fréquence, d'après [16]. Image de [13].

On remarque aisément que l'allure des courbes isophones correspond à celle de la fonction de transfert de l'oreille externe et moyenne retournée (pour les faibles intensités), ce qui indique que cette variation de la perception du sonie en fonction de la fréquence est en grande partie due au filtrage effectué par l'oreille externe et moyenne.

Une autre échelle de sonie a été développée par Stevens [17, 18]. A partir de tests subjectifs, il a observé que la sonie perçue L est une puissance de l'intensité du son I :

$$L = k.I^{0,3}$$

avec k constante dépendant des unités. Stevens propose alors une nouvelle échelle de sonie, celle des 'sones'. Pour étalonner l'échelle, 1 sone est défini arbitrairement comme étant la sonie d'une fréquence pure à 1kHz de 40 dB SPL, présentée dans les deux oreilles depuis une position frontale. Ainsi, une fréquence pure à 1kHz de 50 dB SPL est jugé deux fois plus fort que celle à 40 dB SPL, elle est donc de 2 sones. Cette relation semble ne pas fonctionner pour des sons inférieurs à 40 dB SPL, où la sonie change plus rapidement. L'échelle des sones est présentée figure 4.9. Même si la valeur 0,3 de l'exposant est contestée et différente dans plusieurs modèles, cette échelle a été validée par un grand nombre d'expériences utilisant plusieurs techniques de mesure, et c'est celle que nous allons tenter d'évaluer en temps réel à l'aide de nos outils.

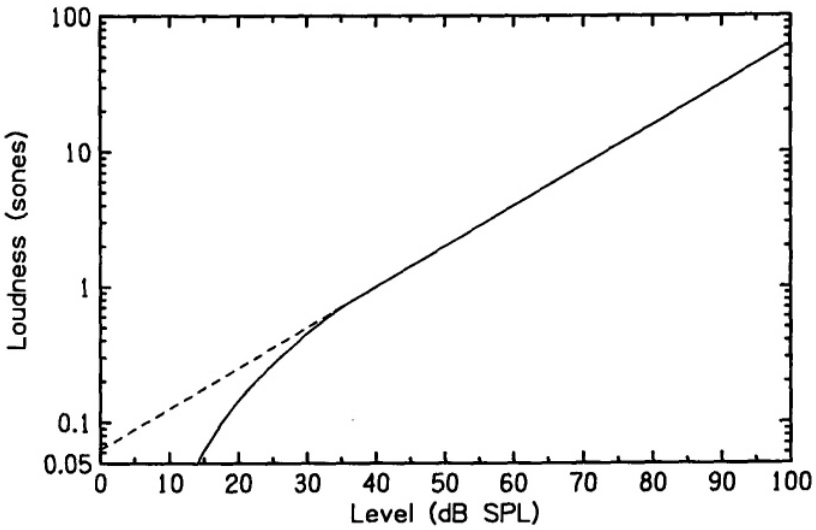


Fig. 4.9 : En trait plein, échelle des sones en fonction de l'intensité du son, mesures pour une fréquence pure à 1kHz. En trait pointillé correspond la loi d'exposant de Steven [17, 18]. Image de [1].

IV.4.b. Evaluation par le modèle temps réel

Calculons à présent la sonie en temps réel à l'aide de notre modèle, il faut alors reprendre les différentes étapes décrites précédemment. Notons par ailleurs que l'approche qui a été adopté est celle de Glassberg et Moore [14].

D'abord, il faut recréer le filtrage effectué par l'oreille externe et moyenne, dont la fonction de transfert est précisée figure 4.7. Pour des raisons de performance, en respectant le cadre tu temps réel, une solution à l'aide de filtres à réponse impulsionnelle infinie (filtrage récursif) s'impose. En effet, on réduit considérablement le temps de calcul par rapport à un filtre à réponse finie, qui a besoin de trop de points pour avoir une précision suffisante (Glassberg et Moore choisissent une solution par filtre à réponse impulsionnelle finie de 4097 coefficients [14]).

La fonction de transfert du filtre retenu est présentée figure 4.10. Le filtre a été réalisé à l'aide de quatre filtres passe-bande d'ordre 4 à réponse impulsionnelle finie. Ils sont montés en parallèle, et une somme pondérée des quatre sorties est effectuée ensuite. On remarque une très grande similitude avec le filtre mesuré présenté à la figure 4.7, notamment le pic fréquentiel situé aux alentour de 3kHz, et le plateau à 0 dB aux alentours de 1kHz.

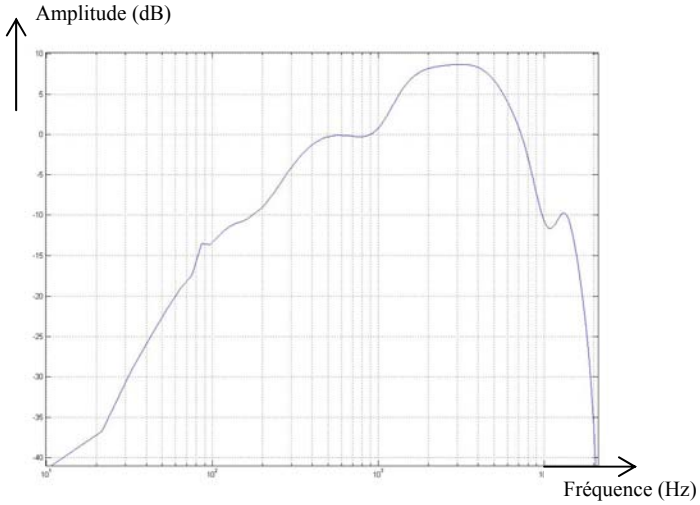


Fig. 4.10 : Réponse en fréquence de la modélisation de l'oreille externe et moyenne.

Après être passé par l'oreille moyenne et externe, le son entre dans l'oreille interne, c'est-à-dire la cochlée. Notre banc de filtre Gammatone va bien sûr réaliser cette étape.

Le long de la membrane basilaire, les cellules ciliées vont ensuite être mises en vibration. Il est important de reproduire deux phénomènes : le fait que les cellules ciliées ne vibrent que dans un seul sens (redressement mono alternance en quelque sorte), et surtout l'effet du seuil d'audition, en effet, pour des sons de faible amplitude, on a vu figure 4.9 que la sonie ne suit plus la loi de compression. A une excitation de type fréquence pure, le système excite principalement une bande de fréquence et les bandes voisines (suivant la forme du pattern fréquentiel). Cependant, si l'excitation est trop faible, la contribution des bandes de fréquence voisines tombe sous le seuil d'audition, elle n'est plus perçue et provoque une baisse rapide de la sonie.

Une façon simple de réaliser cette étape est de disposer un seuil à la sortie du banc de filtre. Seuls les canaux apportant une contribution suffisante (au dessus du seuil d'audition) seront ainsi pris en compte.

Stevens [17, 18] nous indique ensuite que la perception de la sonie suit une loi de compression, nous la matérialiserons à l'aide d'un exposant. Afin de réaliser une intégration du flux et ainsi lisser le résultat, on place ensuite un filtre passe-bas coupant 1Hz.

Une fois cette chaîne terminée, pour calculer la sonie instantanée à partir du signal transformé dans chaque bande, il ne reste plus qu'à faire la somme de tous les canaux.

Le patch qui calcule la sonie en temps réel présenté figure 4.11. Chaque étape est modélisée par un objet, on note la présence d'un objet '+_m' qui donne la somme des échantillons présents dans chaque bande à l'instant où il reçoit un message 'bang'. Les valeurs de seuil, d'exposant et de normalisation (multiplication finale) ont été déterminées à l'utilisation du patch, elles permettent en particulier de respecter l'échelle des sones pour une valeur d'excitation en dB (à préciser en entrée du banc de filtre).

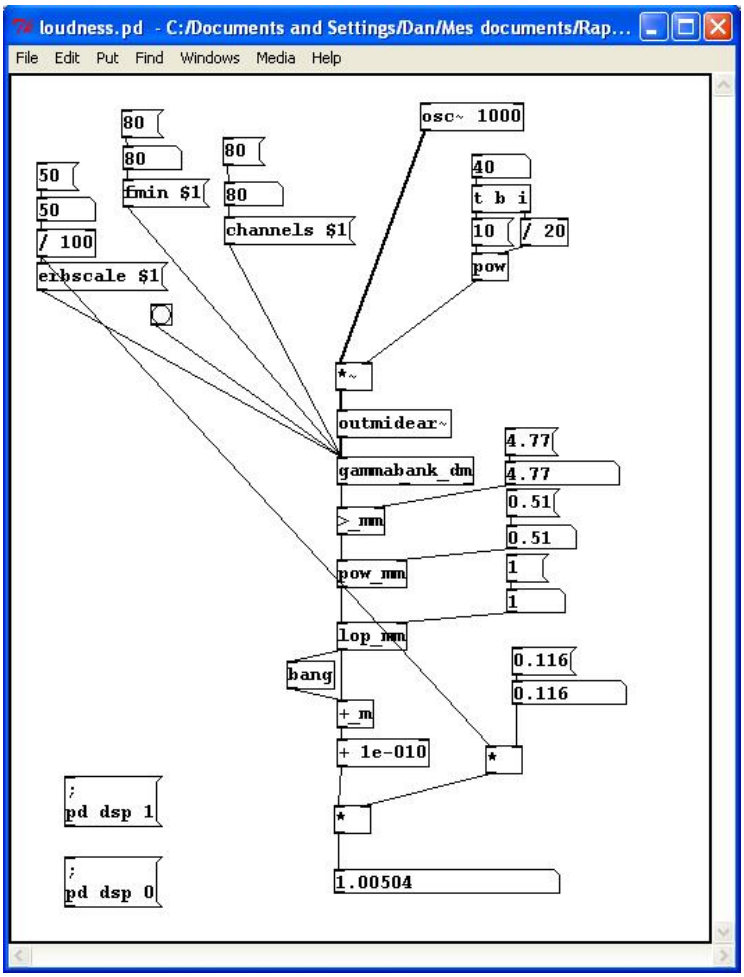


Fig.4.11 : Patch réalisant le calcul de la sonie, on remarque le résultat de 1 sone pour une fréquence pure à 1kHz à 40 dB.

IV.4.c. Résultats - Discussion

A l'aide du patch précédent, on réalise la figure 4.12. On retrouve les mêmes valeurs de sonie que la courbe théorique de la figure 4.9. A l'aide de nos outils de modélisation, il semble que l'on ait reproduit le modèle de calcul de sonie de Glassberg et Moore [14] en temps réel.

Il faut compléter la validation du modèle avec la mesure de la sonie pour des bruits à bandes étroite : quand la largeur de bande d'un bruit augmente (tout en conservant la même puissance), la sonie n'évolue pas jusqu'à un certain point appelée largeur de bande critique, puis augmente [14, 20, 13]. Cette mesure a été faite avec le patch temps réel, la constance de la sonie dans la bande critique et l'augmentation au delà est reproduite mais cette élévation est beaucoup trop importante. Cela pourrait indiquer que la sonie des sons à large bande va être surestimée, et que donc une amélioration future du modèle de sonie sur ce point serait intéressante.

Le calcul a été déterminé à partir d'observations sur des fréquences pures et des bruits, mais il est maintenant possible de calculer la sonie de sons complexes quelconques en utilisant le même procédé, pour un vumètre par exemple.

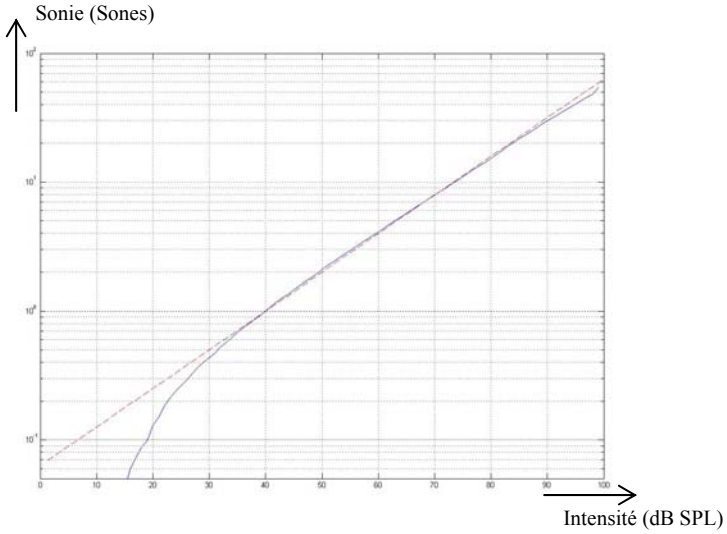


Fig. 4.12 : En trait plein, mesure de la sonie par notre modèle d'une fréquence pure à 1kHz en fonction de l'intensité. En trait pointillé correspond la loi d'exposant de Steven [17, 18].

IV.5. Rugosité

Nous pouvons calculer en temps réel un attribut perceptif avec notre modèle, la sonie. Intéressons nous maintenant au calcul de la rugosité, une autre grandeur subjective de la perception.

IV.5.a. Définition

Les études concernant la perception de la rugosité sont nombreuses et leurs approches parfois contradictoires, ce qui rend toute tentative de définition simple soit partielle, soit incorrecte. La première description du phénomène, dite spectrale, est donnée par Helmholtz [23], c'est celle que nous allons détailler dans un premier temps.

Si deux sons sont de fréquence différentes, leur superposition produit la perception d'un son pur à la fréquence moyenne des sons originaux dont l'amplitude fluctue légèrement. Ce phénomène connu sous le nom de battements peut être clairement entendu jusqu'à 4 à 6Hz de différence entre les deux sons purs.

Quand on augmente l'écart entre les deux fréquences, les battements s'accélèrent. Au delà d'une dizaine de Hz, leur perception s'estompe, il devient par exemple impossible de les compter. Néanmoins, Helmholtz affirme que nous continuons à avoir conscience de la présence de ces battements rapides : c'est cette impression qui donne au son cet aspect rugueux.

La figure 4.13 illustre ces conclusions, comment la perception des battements devient celle de la rugosité, jusqu'à ce que les deux sons puissent être perçus séparément. Par ailleurs, selon Helmholtz, des battements de l'ordre de 33Hz produisent une rugosité maximale.

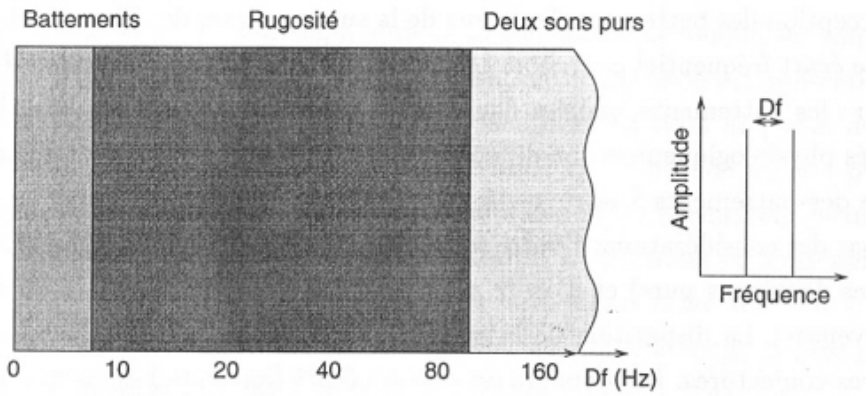


Fig. 4.13 : Qualités perceptives d'une paire de sons purs en fonction de leur écart fréquentiel. Les frontières entre régions ne sont pas abruptes, notamment la frontière supérieure entre rugosité et deux sons purs distincts dépend du registre. D'après [24].

Une autre approche de la rugosité, dite temporelle, est envisagée. Au lieu de choisir le stimulus comme étant la somme de deux sons purs, choisissons un signal sinusoïdal modulé en amplitude. Dans le premier cas, l'enveloppe du signal s'appelle battements, dans un second il s'agit de la fréquence de modulation. Zwicker et Fastl [25] établissent de nombreux résultats à l'aide de ce type de stimulus, notamment liés à la fréquence de la modulation : pour un son pur de 1kHz, modulé en amplitude avec une profondeur unitaire, la rugosité maximale est atteinte pour une fréquence de modulation de 70Hz. On dit alors que ce type de signal a une rugosité de 1 sur l'échelle de la rugosité. La figure 4.14 représente cette échelle de rugosité en fonction de la fréquence de modulation. On remarque que la rugosité est maximum pour des fréquences de modulation de l'ordre de 70Hz.

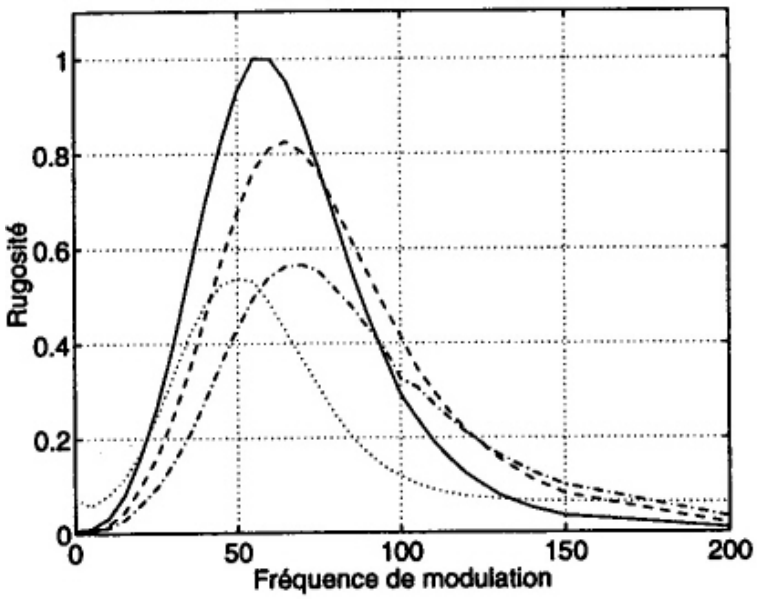


Fig. 4.14 : Rugosités d'une fréquence pure à 1kHz (trait plein), 500Hz (trait pointillé), 2kHz (trait hachuré) et 4kHz (trait alterné) modulé en amplitude en fonction de la fréquence de modulation, profondeur de modulation unitaire, indiqué m=1. D'après [24].

Zwicker et Fastl ont également examiné l'influence de la profondeur de modulation sur la rugosité. Les résultats de leurs observations sont montrés figure 4.15 : la rugosité est une fonction croissante de la profondeur de modulation. Zwicker et Fastl proposent alors une loi pour déterminer la rugosité d'une fréquence pure de 1kHz modulée en amplitude à 70Hz avec une profondeur de modulation m : on obtient la rugosité en calculant $m^{1,6}$.

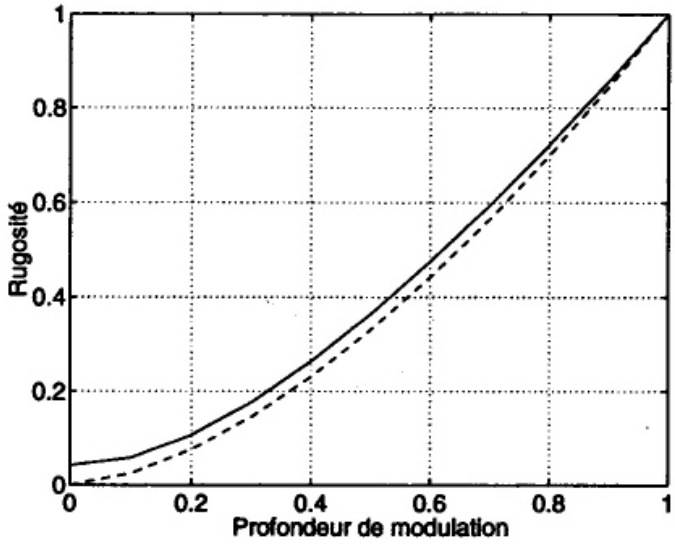


Fig. 4.15 : Rugosité d'une fréquence pure à 1kHz modulé en amplitude à une fréquence de modulation de 70Hz, en fonction de la profondeur de modulation. En pointillé est représenté la loi $m^{1,6}$ proposé par Zwicker et Fastl [25]. D'après [24].

IV.5.b. Evaluation par le modèle temps réel

Les deux approches, temporelles et spectrales, sont équivalentes sur de nombreux points. Seule la façon d'exprimer les paramètre du stimulus et les résultats attendus diffèrent vraiment.

L'approche temporelle de Zwicker et Fastl [25] va servir de base à l'élaboration du patch de calcul de la rugosité. On reprend le modèle de Pressnitzer présenté dans [24] : le signal original passe d'abord dans un préfiltrage de type passe-bande entre 450 et 1000Hz, une décomposition en sous bandes fréquentielles est ensuite réalisée par le banc de filtre Gammatone. L'utilisation du banc de filtre Gammatone s'explique par le fait que la rugosité est reliée à la quantité de modulation d'amplitude perçue, autour de 70Hz de fréquence de modulation. Perçue, cela veut dire que la modulation doit être présente après filtrage auditif. Par exemple, dans les basses fréquences, les filtres sont plus étroits que 70Hz, donc la modulation est atténue pour des fréquences de modulation plus faibles que 70Hz, et la rugosité est aussi faible.

Dans chaque sous bande on reproduit l'effet des cellules ciliés par un redressement demi période suivit d'une compression par loi d'exposant 0.5, puis on applique un filtrage passe-bas suivit d'un passe-haut tous deux de l'ordre 70Hz pour extraire l'enveloppe efficace où seules les fréquences de modulation contribuant à la rugosité sont préservées. La rugosité par canal est finalement obtenue en calculant la valeur RMS de l'enveloppe efficace, puis en élevant cette valeur au carré. La rugosité totale est la somme des rugosités dans chaque canal.

Le patch réalisé pour calculer la rugosité est présenté figure 4.16. On reconnaît les divers éléments du modèle de calcul choisit, en particulier le préfiltrage avec les deux passe-bas et les deux passe-haut placés avant le banc de filtre. Il n'existe pas encore d'objet permettant le calcul de la valeur RMS sur un nombre suffisant de point, la solution proposée dans ce patch est de filtrer par un passe bas avec une fréquence de coupure très faible agissant comme un intégrateur, et de faire une somme instantanée des valeurs dans chaque bande pour avoir la rugosité globale.

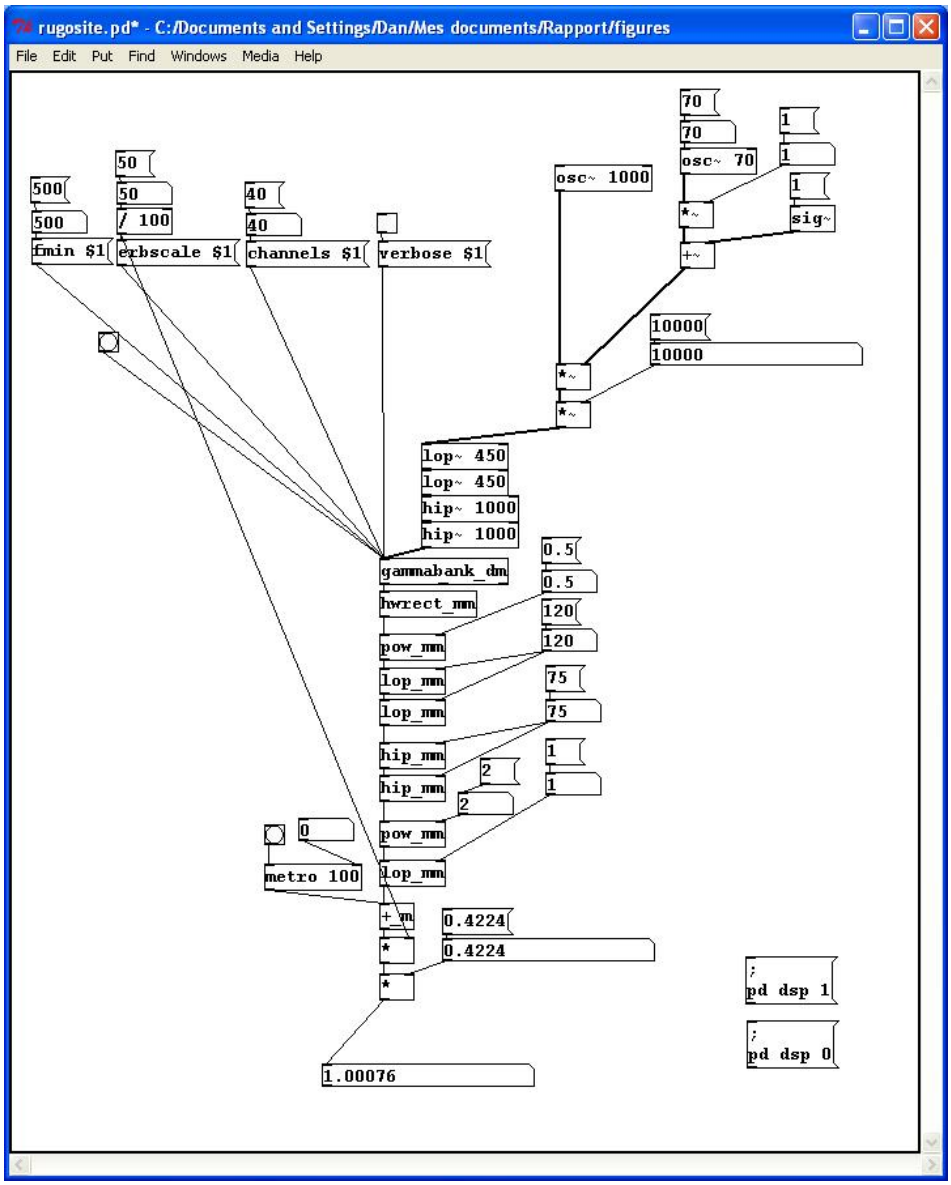


Fig. 4.16 : Patch réalisant le calcul de la rugosité, on remarque une rugosité de 1 pour une fréquence pure à 1kHz modulée en amplitude à 70Hz avec une profondeur de modulation unitaire.

IV.5.c. Résultats

On relève à l'aide de notre modèle la rugosité d'une fréquence pure modulée en amplitude en fonction de la fréquence de modulation (figure 4.17, pour plusieurs valeurs de fréquences porteuses), et en fonction de la profondeur de modulation (figure 4.18). Les allures et les ordres de grandeurs sont assez bien respectés, on vérifie peut de différences avec les figures théoriques 4.14 et 4.15. On note cependant une faible rugosité quand la fréquence porteuse est élevée (4kHz, en vert sur figure 4.17), la rugosité risque d'être quelque peu sous évaluée pour des sons à hautes fréquences.

Comme pour le calcul de la sonie, le patch est étalonné à partir de signaux test de type sinus modulés, mais le calcul de la rugosité de sons complexes quelconques est maintenant envisageable.

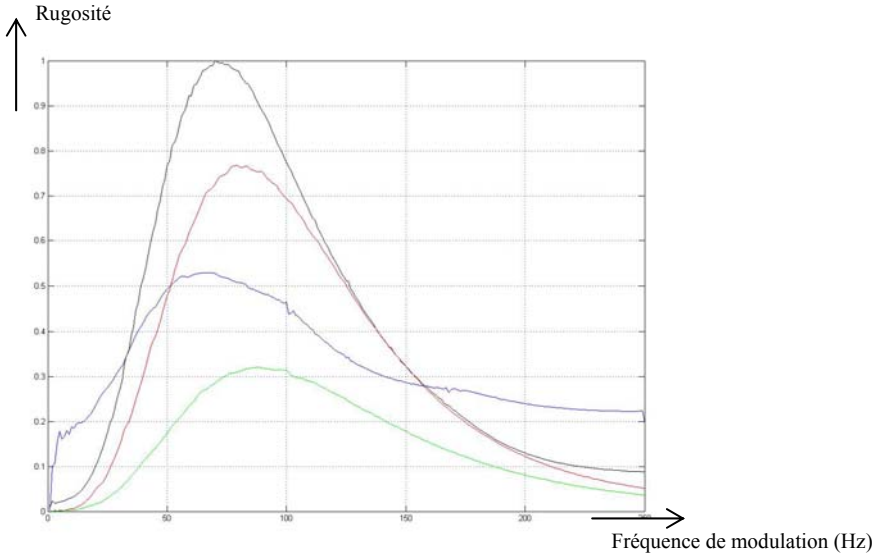


Fig. 4.17 : Rugosités calculées par le modèle d'une fréquence pure à 1kHz (noir), 500Hz (bleu), 2kHz (rouge) et 4kHz (vert) modulé en amplitude en fonction de la fréquence de modulation, profondeur de modulation unitaire, indiqué m=1.

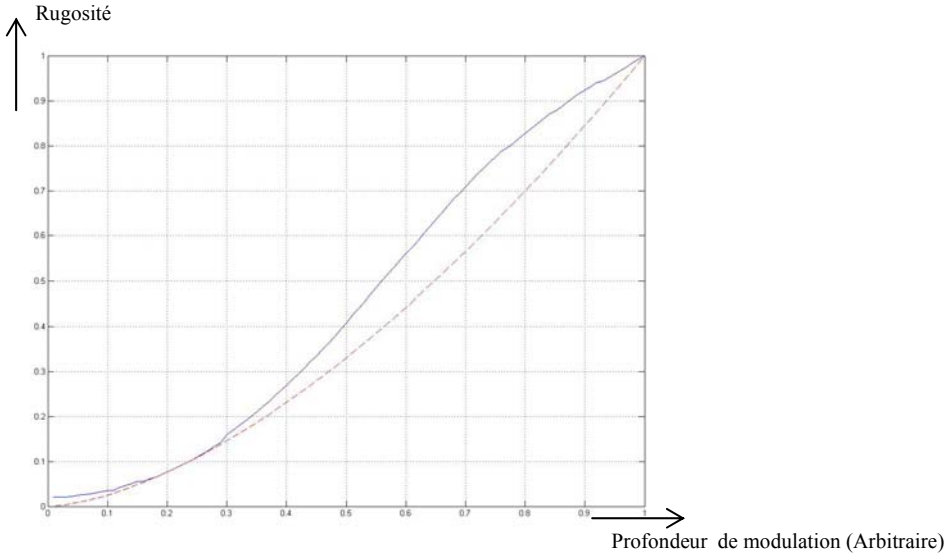


Fig. 4.18 : Rugosité calculée par le modèle d'une fréquence pure à 1kHz modulé en amplitude à une fréquence de modulation de 70Hz, en fonction de la profondeur de modulation. En pointillé rouge est représenté la loi $m^{1,6}$ proposé par Zwicker et Fastl [25].

Chapitre V.
Conclusion

Un modèle auditif en temps réel a été développé pendant ce stage. Cette modélisation vise le traitement auditif périphérique, qui comprend plusieurs étapes : l'oreille externe, moyenne, et interne c'est-à-dire cochlée puis cellules ciliées le long de la membrane basilaire. L'enjeu majeur de la modélisation est l'étape de la cochlée puisqu'elle décompose le signal entrant en plusieurs bandes de fréquences, ce découpage devant répondre à des critères mécaniques précis et ayant de fortes conséquences perceptives.

Après un état des lieux des différentes approches possibles et existantes, la solution retenue pour modéliser de façon fonctionnelle l'oreille interne est celle du banc de filtre Gammatone. Ce procédé est largement utilisé dans de nombreux modèles auditifs existants car il reproduit avec une fidélité acceptable les principales caractéristiques fréquentielles de l'analyse effectuée par l'oreille humaine. Par ailleurs, d'un point de vue traitement du signal, les filtres Gammatone sont de simples filtres à réponse impulsionnelle infinie (filtres récursifs) d'ordre 4, approximés par des filtres tout-pôles d'ordre 4, ils sont donc peu consommant en terme de puissance de calcul, ce qui joue en leur faveur dans le cadre d'une implémentation temps réel. De plus, ce choix du banc de filtre Gammatone a été motivé par la possibilité d'une resynthèse du son décomposé, ainsi, on peut voir le modèle comme un simple banc de filtre : décomposer, modifier puis resynthétiser le son pour faire des traitements audio bénéficiant de la base perceptive de la décomposition.

En ce qui concerne les autres étapes de la modélisation, des approches simples ont été choisies. Un filtrage par quatre filtres récursifs passe-bande disposés en parallèle a permis d'une part de recréer les effets de l'oreille externe et moyenne sur le spectre du son transmis à l'oreille interne, et une combinaison de redressement demi période, filtrage passe bas et loi d'exposant a été choisie pour simuler de façon le résultat net observée dans le nerf auditif après les cellules ciliés internes. Ces procédés simples ont permis une implémentation temps réel performante en limitant les calculs nécessaires.

Une fois les choix faits pour les traitements audio, le modèle a été programmé sur une interface temps réel, Pure Data. Les traitements audio sont réalisés à l'aide d'objets assemblés les uns aux autres, il est notamment possible de développer ses propres objets en C++ grâce à la couche logicielle Flext.

Il a fallu répondre à une question d'ordre architecturale importante : quel format audio adopter sachant que la sortie du banc de filtre Gammatone est multiple (une par bande de fréquence) et variable (nombre de canaux du banc de filtre modifiable dynamiquement) ? Un format audio multiplexé s'est imposé comme solution à notre problème. Toutes les sorties du banc de filtre sont ainsi matérialisées par une seule et même liaison, facilitant considérablement l'utilisation du modèle.

Afin de conserver l'intérêt du multiplexage, de nombreux objets de traitement compatibles avec ce format de données ont été développés et regroupés au sein d'une bibliothèque nommée 'Audition Library'. Cette bibliothèque permet alors de réaliser simplement de nombreux traitements (utilisant ou non le banc de filtre Gammatone) en profitant du format multiplexé.

Nous avons par ailleurs profité de la bibliothèque pour commencer à illustrer l'utilisation notre modèle dans plusieurs applications. D'abord, un outil de visualisation a été pensé, a partir du banc de filtre Gammatone, et des approches modélisant les autres parties de l'oreille. Cette visualisation se comporte donc comme le système auditif, il ne s'agit pas d'une visualisation spectrale de type spectrogramme à pas fréquentiel et temporel fixes, mais d'une représentation spectrale se rapprochant du concept 'ce que l'on voit est ce que l'on entend'.

L'objet de resynthèse a ensuite permis l'élaboration de deux traitements audio : une simulation d'implant cochléaire et une synthèse croisée. L'utilisation de la décomposition par le banc de filtre Gammatone a été justifiée dans le cas de la parole par [21], même s'il est possible de mettre n'importe quelle source en entrée de la synthèse croisée notamment.

Avec le modèle complet de l'oreille, il a enfin été possible de calculer des attributs perceptifs en temps réel. La sonie et la rugosité sont les grandeurs subjectives qui ont été évaluées à partir de modèles n'existant pas en temps réel et utilisant une décomposition par banc de filtres auditif.

Des améliorations sont envisageables à plusieurs niveaux. D'abord, certaines étapes de la modélisation, plus particulièrement celles correspondant aux cellules ciliées externes, peuvent être perfectionnées. En effet, des étapes non linéaires plus réalistes, et un mécanisme d'adaptation et contrôle de gain, qui dans le vrai système accentue les attaques, sont des voies à suivre. L'accent a été mis sur le banc de filtre Gammatone pendant ce stage, mais ces nouvelles approches peuvent faire l'objet d'un travail futur.

Par ailleurs, la bibliothèque 'Audition Library' peut largement évoluer puisque seuls des traitements élémentaires sont disponibles pour le format de données multiplexées. Cette bibliothèque va être soumise à la communauté des développeurs et utilisateurs de Pure Data, notamment par le bais d'une page Internet, et l'expansion se fera si des personnes trouvent un intérêt dans le modèle auditif et le format de données multiplexées.

Les utilisations proposées dans ce rapport sont encourageantes car les effets perceptifs attendus sont au rendez-vous. Les traitements audio prennent mieux en compte la décomposition fréquentielle du son par le système auditif que la plupart des traitements audio temps réel connus, la visualisation semble pertinente vis-à-vis des phénomènes de masquage notamment, et les attributs perceptifs sont correctement évalués à partir d'une architecture commune simple. Il est bien sûr envisageable d'affiner les modèles de calcul envisagés pour ces procédés, mais surtout d'en faire des nouveaux tirant partie de cette structure commune désormais mise en place. Il serait par exemple intéressant de faire en temps réel des compressions audio, des simulations de surdités légères ou plus importantes et comportant un recrutement de sonie, et des calculs de données subjectives comme la hauteur ou diverses dimensions du timbre comme le centre de gravité spectral ou tout autre descripteur audio adapté à une application temps réel, musicale ou autre.

Nous avons donc désormais implémenté un ensemble de briques élémentaires qui sont largement utilisées dans la communauté de la modélisation auditive. Le choix de Pure Data comme environnement temps réel semble important pour favoriser l'utilisation de ces outils issus des connaissances acquises en psychoacoustique, ce logiciel étant utilisé entre autre par des musiciens plus ou moins confirmés et des passionnés d'informatique musicale. Ces

utilisateurs développent toujours, avec la pratique, des intuitions quant à l'emploi de ces outils. Il serait intéressant par la suite de voir comment vont être utilisées ces nouvelles fonctions, en espérant que leur l'apport perceptif offre de nouvelles perspectives musicales.

Annexes

Les fichiers ‘gammabank_dm.cpp’, ‘gammasynth_md.cpp’, et ‘libaudition.cpp’ sont fournis dans cette annexe.

```
/*****************************************************************/
/* gammabank_dm.cpp                                              */
/* realise le banc de filtre gammatone et traite le flux         */
/* argument: nombre de canaux, defaut=16                    */
/* in1: signal, bang, fmin float(freq min), erbscale float(pas en erb),*/
/*      channels int(nombre de canaux),                          */
/*          verbose bool(indications dans console)               */
/* out1: multiplex signal reel; out2: multiplex signal imag;     */
/* out3: parametres du banc (out sur bang)                       */
/*                                                               */
/*   Dan Gnansia       mai 2005                                  */
/*****************************************************************/


#include <flext.h>
//#include "libaudition.h"
#include "gammabank_dm.h"

#if !defined(FLEXT_VERSION) || (FLEXT_VERSION < 401)
#error You need at least flext version 0.4.1
#endif



FLEXT_LIB_DSP_V("help, gammabank_dm gammabank_dm~", gammabank_dm)


gammabank_dm::gammabank_dm(int argc,t_atom *argv):
      nbch(0)
{
      int fl=0;
      fbcontrol=1;

      if(argc!=0) fl=(int)atom_getfloat(&argv[0]);

      if(argc==2) fbcontrol=atom_getfloat(&argv[1]);

      if(fl>0)nbch = fl;
      else
      {
            nbch = 16;
            post("gammabank_dm : default channel number 16");
      }


      AddInSignal("audio in");
      AddOutList("audio real out");
```

```
	AddOutList("audio imag out");
	AddOutList("init out");

	FLEXT_ADDBANG(0,m_bang);
	FLEXT_ADDMETHOD_F(0,"fmin",m_infmin);
	FLEXT_ADDMETHOD_F(0,"erbscale",m_inerbscale);
	FLEXT_ADDMETHOD_B(0,"verbose",m_verbose);
	FLEXT_ADDMETHOD_I(0,"channels",m_channels);


	int i;

	fmin = 50;
	erbpas = 1;
	first=1;
	verbose=false;
	post1=0;
	newnbch=nbch;

	yreel1=new float [nbch];
	yimag1=new float [nbch];
	yreel2=new float [nbch];
	yimag2=new float [nbch];
	yreel3=new float [nbch];
	yimag3=new float [nbch];
	yreel4=new float [nbch];
	yimag4=new float [nbch];
	fc=new float [nbch];
	areel=new float [nbch];
	aimag=new float [nbch];
	norm=new float [nbch];

	for(i=0;i<(int)(nbch);i++)
	{
		yreel1[i]=0;
		yimag1[i]=0;
		yreel2[i]=0;
		yimag2[i]=0;
		yreel3[i]=0;
		yimag3[i]=0;
		yreel4[i]=0;
		yimag4[i]=0;
		areel[i]=0;
		aimag[i]=0;
		norm[i]=0;
	}
	m_bang();

}

gammabank_dm::~gammabank_dm()
{
	delete [] yreel1;
	delete [] yimag1;
	delete [] yreel2;
	delete [] yimag2;
	delete [] yreel3;
	delete [] yimag3;
	delete [] yreel4;
	delete [] yimag4;
	delete [] fc;
```

```
	delete [] areel;
	delete [] aimag;
	delete [] norm;
}

void gammabank_dm::m_infmin(float in)
{
	if(in>0)fmin=in;
	else fmin=0;
}

void gammabank_dm::m_inerbscale(float in)
{
	if(in>0 && in<5)erbpas=in;
	else if(in<0)erbpas=0.001;
	else if(in>5)erbpas=5;
}

void gammabank_dm::m_verbose(bool in)
{
	verbose=in;
}

void gammabank_dm::m_channels(int in)
{
	if(in>0)
	{
		newnbch=in;
		post1=0;
	}
	else
	{
		if(post1==0)
		{
			post("gammabank_dm: channel number must be >0");
			post1=1;
		}
	}

}

void gammabank_dm::channels(int in)
{
	delete [] yreel1;
	delete [] yimag1;
	delete [] yreel2;
	delete [] yimag2;
	delete [] yreel3;
	delete [] yimag3;
	delete [] yreel4;
	delete [] yimag4;
	delete [] fc;
	delete [] areel;
	delete [] aimag;
	delete [] norm;

	nbch=in;

	int i;

	yreel1=new float [nbch];
```

```
	yimag1=new float [nbch];
	yreel2=new float [nbch];
	yimag2=new float [nbch];
	yreel3=new float [nbch];
	yimag3=new float [nbch];
	yreel4=new float [nbch];
	yimag4=new float [nbch];
	fc=new float [nbch];
	areel=new float [nbch];
	aimag=new float [nbch];
	norm=new float [nbch];

	for(i=0;i<(int)(nbch);i++)
	{
		yreel1[i]=0;
		yimag1[i]=0;
		yreel2[i]=0;
		yimag2[i]=0;
		yreel3[i]=0;
		yimag3[i]=0;
		yreel4[i]=0;
		yimag4[i]=0;
		areel[i]=0;
		aimag[i]=0;
		norm[i]=0;
	}
}

void gammabank_dm::gammabank_dm_acalcul(float fc, float fb, float fs, int
i)
{
	float beta,lambda,p,phi;

	beta = 6.2831853072*(fc/fs);

	phi = 6.2831853072*(fb/2)/fs;
	p = (1.6827902833*cos(phi)-2)/(1-0.8413951417);
	lambda = -p/2-sqrt((pow(p,2)/4)-1);

	areel[i] = lambda*cos(beta);
	aimag[i] = lambda*sin(beta);

}

void gammabank_dm::m_bang()
{
	float q=9.265;
	float l=24.7;
	float erbscale=q*log(1+fmin/(l*q));
	float fs=sys_getsr();
	float fb;
	int i;
	t_atom *tab;

	if(newnbch!=nbch && post1==0)channels(newnbch);

	if(verbose==true) post("\n");
	for(i=0;i<nbch;i++)
	{
		fc[i]=(exp((erbscale+i*erbpas)/q)-1)*l*q;
		fb=(l+fc[i]/q)*0.8861*fbcontrol;
```

```
            gammabank_dm_acalcul(fc[i], fb, fs, i);

            if(fc[i]<=fs/2)   norm[i] = 2*pow(1-
(sqrt(pow(areel[i],2)+pow(aimag[i],2))),4);
            else
            {
                  norm[i]=0.0;
                  post("gammabank_dm: a center frequency is out of
range!");
            }

            if(verbose==true) post("fc: %f  fb: %f  ar: %f  ai: %f",
fc[i],fb,areel[i],aimag[i]);
      }

      tab = new t_atom[3*nbch+2];

      tab[0].a_type=A_FLOAT;
      tab[0].a_w.w_float=nbch;
      tab[1].a_type=A_FLOAT;
      tab[1].a_w.w_float=1/(erbpas);

      for(i=0;i<nbch;i++)
      {
            tab[i+2].a_type=A_FLOAT;
            tab[i+2].a_w.w_float=(float)areel[i];

            tab[i+2+nbch].a_type=A_FLOAT;
            tab[i+2+nbch].a_w.w_float=(float)aimag[i];

            tab[i+2+2*nbch].a_type=A_FLOAT;
            tab[i+2+2*nbch].a_w.w_float=fc[i];
      }

      if(first==0)
      {
            post("gammabank_dm: sending init data");
            ToOutList(2,3*nbch+2,tab);
      }
      else first=0;

      delete [] tab;

}

void gammabank_dm::gammabank_dm_gammafilter(float in, int j, float *outr,
float *outi)
{
      float yreel,yimag;

      //1e-20 anti-denormal number, évite la sur-consommation quand signal
entrant faible
      yreel = (in + 1e-20) + areel[j] * yreel1[j] - aimag[j] * yimag1[j];
      yimag =    1e-20     + areel[j] * yimag1[j] + aimag[j] * yreel1[j];
      yreel1[j] = yreel;
      yimag1[j] = yimag;

      yreel = yreel + areel[j] * yreel2[j] - aimag[j] * yimag2[j];
      yimag = yimag + areel[j] * yimag2[j] + aimag[j] * yreel2[j];
      yreel2[j] = yreel;
      yimag2[j] = yimag;
```

```
	yreel = yreel + areel[j] * yreel3[j] - aimag[j] * yimag3[j];
	yimag = yimag + areel[j] * yimag3[j] + aimag[j] * yreel3[j];
	yreel3[j] = yreel;
	yimag3[j] = yimag;

	yreel = yreel + areel[j] * yreel4[j] - aimag[j] * yimag4[j];
	yimag = yimag + areel[j] * yimag4[j] + aimag[j] * yreel4[j];
	yreel4[j] = yreel;
	yimag4[j] = yimag;

	*(outr) = yreel * norm[j];
	*(outi) = yimag * norm[j];
}

void gammabank_dm::m_signal(int n, float *const *ins, float *const *outs)
{

	const float *in = ins[0];
	int N=n;
	int count=2;
	int i;
	t_atom *out1 = new t_atom[n*nbch+2];
	t_atom *out2 = new t_atom[n*nbch+2];

	float tmp1;

	out1[0].a_type=A_FLOAT;
	out1[0].a_w.w_float=(float)nbch;
	out1[1].a_type=A_FLOAT;
	out1[1].a_w.w_float=(float)n;

	out2[0].a_type=A_FLOAT;
	out2[0].a_w.w_float=(float)nbch;
	out2[1].a_type=A_FLOAT;
	out2[1].a_w.w_float=(float)n;

	while(n--)
	{
		tmp1=*(in++);
		for(i=0;i<nbch;i++)
		{
			out1[count].a_type=A_FLOAT;
			out2[count].a_type=A_FLOAT;

	gammabank_dm_gammafilter(tmp1,i,&out1[count].a_w.w_float,&out2[count]
.a_w.w_float);
			count++;
		}
	}

	ToOutList(0,(N*nbch)+2,out1);
	ToOutList(1,(N*nbch)+2,out2);

	delete [] out1;
	delete [] out2;

}
```

```
/*********************************************************************/
/* gammasynth_md.cpp                                                 */
/* synthetise un signal analysé par gammabank_dm                     */
/* argument: nombre de canaux, defaut=16                             */
/* in1: signal reel multiplex; in2: signal imag multiplex            */
/* in3: init(format de gammabank_dm::out3)                           */
/* out1: signal                                                      */
/*                                                                   */
/*   Dan Gnansia       mai 2005                                      */
/*********************************************************************/


#include <flext.h>
//#include "libaudition.h"
#include "gammasynth_md.h"

#if !defined(FLEXT_VERSION) || (FLEXT_VERSION < 401)
#error You need at least flext version 0.4.1
#endif

FLEXT_LIB_DSP_V("help, gammasynth_md gammasynth_md~", gammasynth_md)

gammasynth_md::gammasynth_md(int argc,t_atom *argv):
      nbch(0)
{
      int fl;
      if(argc!=0) fl=(int)atom_getfloat(&argv[0]);
      else fl=0;

      if(fl>0)nbch = (int)fl;
      else
      {
            nbch = 16;
            post("gammasynth_md : default channel number 16");
      }

      AddInAnything("real in");
      AddInList("imag in");
      AddInList("init in");
      AddOutSignal("audio out");

      FLEXT_ADDMETHOD_(0,"list",inlistreal);
      FLEXT_ADDMETHOD_(1,"list",inlistimag);
      FLEXT_ADDMETHOD_(2,"list",inlistinit);

      first1=1;
      first2=1;
      initok=0;
      error1=0;
      error2=0;
      error3=0;
      post1=0;
      post2=0;
      post3=0;
      post4=0;
      post5=0;
      realonly=1;

      phi = new float [nbch];
      deltan=new float [nbch];
```

```
	poids=new float [nbch];
	wrptr=new float [nbch];
	delayline=new float [nbch*200];

}

gammasynth_md::~gammasynth_md()
{
	delete [] phi;
	delete [] deltan;
	delete [] poids;
	delete [] wrptr;
	delete [] delayline;
	if(first1==0) delete [] inlist1;
	if(first2==0) delete [] inlist2;
}

void gammasynth_md::channels(int in)
{
	delete [] phi;
	delete [] deltan;
	delete [] poids;
	delete [] wrptr;
	delete [] delayline;
	if(first1==0) delete [] inlist1;
	if(first2==0) delete [] inlist2;

	nbch=in;

	first1=1;
	first2=1;
	initok=0;
	error1=0;
	error2=0;
	error3=0;
	post1=0;
	post2=0;
	post3=0;
	realonly=1;

	phi = new float [nbch];
	deltan=new float [nbch];
	poids=new float [nbch];
	wrptr=new float [nbch];
	delayline=new float [nbch*200];
}


void gammasynth_md::inlistreal(int argc,t_atom *argv)
{
	int i;

	if(argv[0].a_w.w_float!=nbch){
		error1=1;
		if(post1==0){
			post("gammasynth_md: inlet1: error invalid number of
multiplex channel");
			post1=1;}}
	else error1=0;

	if(argc!=nbch*argv[1].a_w.w_float+2){
```

```
            error1=2;
            if(post1==0){
                  post("gammasynth_md: inlet1: error invalid multiplex
signal data");
                  post1=1;}}
      else error1=0;


      if(error1==0){
      post1=0;
      if(argc!=oldsize1)
      {
            if(first1==0) delete [] inlist1;
            inlist1=new float [argc];
      }

      oldsize1=argc;
      for(i=0;i<argc;i++)
      {
            inlist1[i]=atom_getfloat(&argv[i]);
      }
      first1=0;}
}

void gammasynth_md::inlistimag(int argc,t_atom *argv)
{
      int i;

      realonly=0;


      if(argv[0].a_w.w_float!=nbch){
            error2=1;
            if(post2==0){
                  post("gammasynth_md: inlet2: error invalid number of
multiplex channel");
                  post2=1;}}
      else error2=0;

      if(argc!=nbch*argv[1].a_w.w_float+2){
            error2=2;
            if(post2==0){
                  post("gammasynth_md: inlet2: error invalid multiplex
data");
                  post2=1;}}
      else error2=0;


      if(error2==0){
      post2=0;
      if(argc!=oldsize2)
      {
            if(first2==0) delete [] inlist2;
            inlist2=new float [argc];
      }

      oldsize2=argc;
      for(i=0;i<argc;i++)
      {
            inlist2[i]=atom_getfloat(&argv[i]);
      }
```

```
	first2=0;}
}

void gammasynth_md::inlistinit(int argc,t_atom *argv)
{
	int nb,i,m,count,indmax,indice_seuil,indice_inflexion;
	float tmp1,tmp2,b1,b2,x0,moda,filters_per_ERB;
	float *fc,*areel,*aimag;
	float fs=sys_getsr();
	float delay=0.004;

	nb=(int)argv[0].a_w.w_float;

	initok=0;

	//verifie le nombre de canaux, change automatiquement si necessaire
	if(nb!=nbch && nb>0) channels(nb);
	else if(nb<=0)
	{
		if(post5==0)
		{
			post("gammasynth_md: invalid number of multiplex
channels");
			post5=1;
		}
	}

	//if(nb==nbch && nb==(argc-2)/3){
	if(nb==(argc-2)/3){

	filters_per_ERB=1/argv[1].a_w.w_float;

	fc=new float [nb];
	areel=new float [nb];
	aimag=new float [nb];

	for(i=0;i<nb;i++)
	{
		areel[i]=argv[i+2].a_w.w_float;
		aimag[i]=argv[i+nb+2].a_w.w_float;
		fc[i]=argv[i+2*nb+2].a_w.w_float;
	}



	//recherche des max d'amplitude des RI
	//determination du delay en sample a appliquer par canal (deltan)
	//calcul de phi (alignement des phases) par canal
	indmax=(int)(delay*fs)+1;		//delay de groupe de 4ms fixé!!
	for(i=0;i<nb;i++)
	{
		count=0;
		m=0;
		//moda=(float)(sqrt(pow(areel[i],2)+pow(aimag[i],2)));
		moda=(areel[i]*areel[i]+aimag[i]*aimag[i]);
		tmp2=log(moda)/2.0;
		while(m==0)
		{

	tmp1=tmp2*pow((float)count,3)+(6*tmp2+3)*pow((float)count,2)+(11*tmp2
+12)*count+6*tmp2+11;
```

```
            count++;

            if(tmp1<0) m=1;
      }
      if(count>indmax)
      {
            deltan[i]=0;
            tmp1=0.004;
      }
      else
      {
            deltan[i]=(indmax-count);
            tmp1=((float)count)/fs;
      }
      phi[i]=-6.2831853072*fc[i]*tmp1;
}


//Modele de prediction de la ponderation
//indice de debut de seuil
m=0;
indice_seuil=0;
while(m<nb && deltan[m]==0)
{
      indice_seuil++;
      m++;
}
if(indice_seuil>1) indice_seuil--;
if(filters_per_ERB<0.6 && indice_seuil>1) indice_seuil--;


//valeur seuil
if(filters_per_ERB<=1) tmp1=-5.85*filters_per_ERB+2.35;
else tmp1=-8.54*log(filters_per_ERB)-3.5;
for(i=indice_seuil;i<nb-2;i++)
{
      if(tmp1>0) poids[i]=0;
      else poids[i]=tmp1;
}


//avant dernier point
tmp2=exp(2.1*filters_per_ERB-2.6);
if(tmp2<20) poids[nb-2]=tmp1-tmp2;
else poids[nb-2]=tmp1-20;


//dernier point
if(filters_per_ERB>1) poids[nb-1]=-2;
else poids[nb-1]=-2.5*filters_per_ERB+1;


//premier point
if(filters_per_ERB<=1) tmp1=0;
else tmp1=-4.75*filters_per_ERB+4.75;

if(fc[0]<40 && filters_per_ERB>1) tmp2=-fc[0]*0.086 + 3.44;
else tmp2=0;

poids[0]=tmp1-tmp2;
```

```
	//pentes ascendante(1) et descendante(2)
	if(filters_per_ERB>=1 && filters_per_ERB<=2) tmp2=-
0.0045*filters_per_ERB-0.0045;
	else if(filters_per_ERB>2) tmp2=-0.02;
	else if(filters_per_ERB>=0.5 && filters_per_ERB<1) tmp2=(-
0.0164*filters_per_ERB+0.0074);
	else
	{
		if(filters_per_ERB>=0.35) tmp2=(-0.0164 * filters_per_ERB +
0.0074)/2;
		else tmp2=0;
	}
	b2=poids[indice_seuil]-tmp2*fc[indice_seuil];
	tmp1=-3*tmp2;
	b1=poids[0]-tmp1*fc[0];

	x0=(b2-b1)/(tmp1-tmp2);     //point d'intersection
	indice_inflexion=1;
	if(x0>=0 && x0<fc[indice_seuil])
	{
		m=1;
		while(fc[m]<x0) m=m+1;
		indice_inflexion=m;
	}

	if(indice_inflexion>1)
	{
		indice_inflexion++;
		for(i=1;i<indice_inflexion;i++) poids[i]=tmp1*fc[i]+b1;
	}
	for(i=indice_inflexion;i<indice_seuil;i++) poids[i]=tmp2*fc[i]+b2;

	if(poids[indice_inflexion-1]-poids[indice_inflexion]>1)
poids[indice_inflexion-1]=tmp2*fc[indice_inflexion-1]+b2;



	delete [] fc;
	delete [] areel;
	delete [] aimag;

	for(i=0;i<nb;i++) poids[i]=pow(10.0,poids[i]/20.0);

	for(i=0;i<nb;i++)
	{
		wrptr[i]=0.0;
		for(m=0;m<200;m++) delayline[i*200 + m]=0.0;
	}

	error3=0;
	initok=1;
	post3=0;
	post("gammasynth_md: init OK");

	}
	else
	{
		error3=1;
		if(post3==0){
```

```
                  post("gammasynth_md: inlet3: error invalid multiplex init
data");
                  post3=1;}
      }
}

float gammasynth_md::gammasynth_md_delay(float in, int i)
{
      float out;
      int ind=wrptr[i];

      out=delayline[i*200 + ind];
      delayline[i*200 + ind]=in;
      wrptr[i]++;
      if(wrptr[i] >= deltan[i]) wrptr[i]=0.0;
      return(out);
}


void gammasynth_md::m_signal(int n, float *const *ins, float *const *outs)
{
      const float *in1;
      const float *in2;
      float *out=outs[0];
      int count=2;
      int i;

      float tmp1;
      float tmp2=0;

      if(error1==0 && error2==0 && error3==0 && first1==0 && first2==0 &&
initok==1 && realonly==0)
      {
            in1=inlist1;
            in2=inlist2;
            while(n--)
            {
                  for(i=0;i<nbch;i++)
                  {
                        tmp1 = in1[count]*cos(phi[i]) -
in2[count]*sin(phi[i]);
                        tmp1 = gammasynth_md_delay(tmp1,i);
                        count++;
                        tmp2 += tmp1 * poids[i];
                  }
                  *out++=tmp2;
                  tmp2=0;
            }
            post4=0;
      }
      else if(error1==0 && error3==0 && first1==0 && initok==1 &&
realonly==1)
      {
            in1=inlist1;
            while(n--)
            {
                  for(i=0;i<nbch;i++)
                  {
                        tmp1 = gammasynth_md_delay(in1[count],i);
                        count++;
                        tmp2 += tmp1 * poids[i];
```

```
                }
                *out++=tmp2;
                tmp2=0;
          }
          post4=0;
    }
    else if(initok==0)
    {
          if(post4==0){
          post("init first!");
                post4=1;}
          while(n--) *out++=0.0;
    }
    else
    {
          post4=0;
          while(n--) *out++=0.0;
    }

    realonly=1;

}
```

```
/*********************************************************************/
/* libaudition.cpp                                                   */
/*     declaration de la bibilothèque Audition                       */
/*                                                                   */
/*   Dan Gnansia        mai 2005                                     */
/*********************************************************************/


// include flext header
#include <flext.h>

// check for appropriate flext version
#if !defined(FLEXT_VERSION) || (FLEXT_VERSION < 400)
#error You need at least flext version 0.4.0
#endif


/*libaudition::libaudition()
{
}
*/

static void libaudition_setup()
{
      post("");
      post("<<<Audition Library - v1.0 2005>>>");
      post("after _ : I/O format");
      post("_dm: from demultiplex to multiplex format");
      post("_md: from multiplex to demultiplex format");
      post("_mm: from multiplex to multiplex format");
      post("");
      post("objects in the library:");
      post("--> gammabank_dm - mux_dm - one2n_dm");
      post("--> gammasynth_md - demux_md - plus_md");
      post("--> +_mm - *_mm - /_mm - ^_mm - ||_mm");
      post("--> lop_mm - hip_mm - sup_mm - hwrect_mm");
      post("--> cmodule_mm - cangle_mm - ccarrier_mm");
      post("--> snapshot_m - plus_m - getfc_m");
      post("--> outmidear~");
      post("");

      // call the objects' setup routines
      FLEXT_DSP_SETUP(gammabank_dm);
      FLEXT_DSP_SETUP(mux_dm);
      FLEXT_DSP_SETUP(one2n_dm);

      FLEXT_DSP_SETUP(demux_md);
      FLEXT_DSP_SETUP(gammasynth_md);
      FLEXT_DSP_SETUP(plus_md);

      FLEXT_DSP_SETUP(outmidear_tilde);

      FLEXT_SETUP(plus_mm);
      FLEXT_SETUP(mult_mm);
      FLEXT_SETUP(div_mm);
      FLEXT_SETUP(pow_mm);
      FLEXT_SETUP(cmodule_mm);
      FLEXT_SETUP(cangle_mm);
      FLEXT_SETUP(abs_mm);
      FLEXT_SETUP(lop_mm);
      FLEXT_SETUP(ccarrier_mm);
```

```
        FLEXT_SETUP(hwrect_mm);
        FLEXT_SETUP(sup_mm);
        FLEXT_SETUP(hip_mm);

        FLEXT_SETUP(snapshot_m);
        FLEXT_SETUP(plus_m);
        FLEXT_SETUP(getfc_m);

}
// setup the library
FLEXT_LIB_SETUP(Audition,libaudition_setup)
```

Références bibliographiques

[1] W. M. Hartmann : Signals, sound and sensation, *AIP Press.*

[2] B. R. Glasberg & B. C. J. Moore : Derivation of auditory filter shapes from notched-noise data. *Hearing Research*, 47:p 103–138, August 1990.

[3] V. Hohmann : Frequency analysis and synthesis using a Gammatone filterbank, *Acta Acustica united with Acustica* Vol 88 (2002) 433 – 442.

[4] R. D. Patterson, J. Nimmo-Smith, J. Holdsworth, P. Rice: An efficient auditory filterbank based on the Gammatone function, *Paper presented at a meeting of the IOC Speech Group on Auditory Modeling at RSRE*, December 1987.

[5] J. M. Zmölnig : How to write external for Pure Data, *Institute for electronic music and acoustics.*

[6] R. Meddis : Simulation of mechanical to neural transduction in the auditory recepter, *Journal of the Acoustical Society of America*, vol.79, no.3, p. 702-711, March 1986.

[7] M. J. Hewitt & R. Meddis : Implementation details of a computation model of the inner hair-cell/auditory-nerve synapse, *Journal of the Acoustical Society of America*, vol.87, no.4, p. 1813-1816, April 1990.

[8] R. D. Patterson, K. Robinson, J. Holdsworth, D. McKeown, C. Zhang & M. Allerhand : Complex sounds and auditory images, *Auditory physiology and perception, Proceedings of the 9h International Symposium on Hearing,* Y Cazals, L. Demany, K. Horner (eds), Pergamon, Oxford, 429-446. (1992).

[9] R. D. Patterson, Mike H. Allerhand & Christian Giguere (1995). Time-domain modeling of peripheral auditory processing : A modular architecture and a software platform, *J. Acoust. Soc. Am.* vol 98, 1890-1894.

[10] R. D. Patterson : Auditory images: How complex sounds are represented in the auditory system, *J Acoust Soc Japan(E) 21 (4),* 183-190 (2000).

[11] M. Slaney : Lyon's Cochlear Model, *Apple Computer Technical Report #13,* 1988.

[12] S. Seneff : A joint synchrony/mean-rate model of auditory speech processing, *Journal of Phonetics*, Vol. 16, pp. 55-76, 1988.

[13] B. C. J. Moore : An introduction to the psychology of hearing, Fifth edition, *Academic Press,* 2003.

[14] B. R. Glasberg & B. C. J. Moore : A model of Loudness Applicable to time-varying sounds, *J. Audio Eng. Society*, Vol 50, No 5 May 2002.

[15] J. P. Egan & H. W. Hake : On the masking pattern of a simple auditory stimulus. *Journal of the Acoustical Society of America,* 22:622-630 (1950).

[16] H. Fletcher & W. A. Munson : Loudness, its definition, measurement and calculation. *Journal of the Acoustical Society of America*, 5:82-108 (1933).

[17] S.S. Stevens : Concerning the form of the loudness function. *J. of the Acoustical Society of America,* 29:603-606 (1957a).

[18] S.S. Stevens : Perceived level of noise by Mark VII and decibels (E). *J. of the Acoustical Society of America,* 51:575-601 (1972).

[19] T. De Mulder, J.P. Martens, M. Lesaffre, M. Leman, B. De Baets & H. De Meyer4 : Recent improvements of an auditory model based front end for the transcription of vocal queries, *ICASSP*, Montral, Canada, 2004.

[20] E. Zwicker, G. Flottorp & S. S. Stevens : Critical band width in loudness summation. *J. of the Acoustical Society of America,* 29:548-557 (1957).

[21] Z. M. Smith, B. Delgutte & A. J. Oxenham : Chimaeric sounds reveal dichotomies in auditory perception, *Nature*, Vol. 416, 7 March 2002

[22] B. R. Glasberg & B. C. J. Moore : Frequency selectivity as a function of level and frequency measured with uniformly exciting notched noise, *J. Acoust. Soc. Am.*, 108: p 2318–2328, 2000.

[23] H. L. F. von Helmholtz : On the sensation of tone as the physiological basis for the theory of music (1877), 2nd Ed. Trans A. J. Ellis (1885), *German 4th Ed., Dover, New York,* 1954.

[24] D. Pressnitzer : Perception de rugosité psychoacoustique : d'un attribut élémentaire de l'audition à l'écoute musicale, *Thesis at Paris VI University*, 1998.

[25] E. Zwicker & H. Fastl : Psychoacoustics, facts and models, Berlin : Springer Verlag, 1990.

[26] D. Pressnitzer & D. Gnansia : Real time auditory model, *ICMC Conference*, 1134, September 2005