IMPLEMENTATION D'UN CODEUR LPC10 COMPLET SOUS MATLAB

Alexis Moinet & Maxime Tryhoen

Faculté Polytechnique de Mons, Belgique, 9 Rue de Houdain, 7000 Mons

email : alexis.moinet@student.fpms.ac.be
maxime.tryhoen@student.fpms.ac.be

RESUME

Le codage du signal de parole est nécessaire pour permettre une réduction du débit, c'est à dire du nombre de bits que l'on envoie sur un canal par seconde, et cela pour faire une meilleure utilisation de ces canaux de télécommunication. Il existe plusieurs techniques de codage, cet article s'intéressera particulièrement au codage LPC10 dont la norme de l'OTAN permet d'obtenir un débit de 2,4 kbits/s.

L'implémentation d'un codeur LPC10 se fait en plusieurs étapes que nous décrirons. Il faut tout d'abord analyser le signal d'entrée pour obtenir les coefficients que nous coderons puis décoderons pour finalement reconstituer un signal de sortie. La dernière étape consistera en une série de tests de validation et en une comparaison avec un codeur amélioré.

1 INTRODUCTION

Le codage LPC10 traite le signal de parole en faisant une distinction entre les parties voisées et non voisées du son. Les parties voisées présentent une certaine périodicité, ce qui permet de trouver une fréquence fondamentale. La partie non voisée n'est aucunement périodique et il n'y a donc pas de fréquence fondamentale. Les paramètres renvoyés par cette analyse sont la fréquence fondamentale (uniquement pour les parties voisées), un gain et les coefficients d'un filtre tout pôle (filtre autorégressif AR). Le filtre représente l'enveloppe spectrale du signal.

Ces paramètres seront quantifiés avant d'être envoyé sur un canal de transmission. C'est cette étape de quantification de quelques paramètres qui permet de diminuer le débit.

A l'autre bout du canal, on décode et on reconstitue le signal. Les parties voisées seront reconstituées en passant un train d'impulsions (deux impulsions sont espacées d'une période fondamentale) dans le filtre tout pôle et les parties non voisées en colorant un bruit blanc gaussien par ce même filtre.

Pour implémenter un codeur LPC10, nous utiliserons Matlab et une toolbox dédiée au traitement de parole. Cet environnement nous permettra d'utiliser des fonctions avancées qui fournissent des fenêtres de pondération, qui détectent la fréquence fondamentale des parties voisées, qui donnent les coefficients du filtre tout pôle, qui modifient ces coefficients pour faciliter le codage. Dans cet article nous ferons des références à ces fonctions Matlab tout en les expliquant.

2 ANALYSE LPC

L'analyse consiste en un découpage du signal en tranches de quelques dizaines de millisecondes. Comme expliqué dans l'introduction, des coefficients vont être associés par un algorithme à chacune de ces tranches.

Par après, au point 8, nous choisirons différentes valeurs pour la durée de ces tranches[1] et le décalage[2] entre les débuts de deux tranches successives. Pour fixer les idées, la norme OTAN - LPC10 utilise des tranches de 45 ms décalées de 22,5 ms.

Dans un premier temps, chaque tranche de son va être pondérée par une fenêtre. Plusieurs choix de fenêtre sont possibles, nous avons choisi la fenêtre de Hamming.

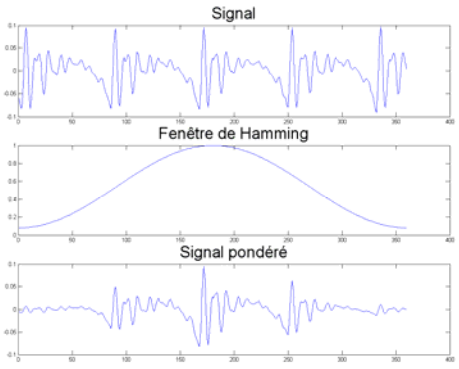

Figure 1 : signal, fenêtre de Hamming et signal pondéré par cette fenêtre.

[1] Que nous appellerons L dans notre programme
[2] Que nous appellerons E dans notre programme

Le but de cette opération est de limiter les erreurs lors du calcul des coefficients. En effet, l'analyse LPC est une analyse prédictive qui utilise donc les valeurs entourant un échantillon pour le prédire. Comme les échantillons du début et de la fin n'ont soit pas de prédécesseurs, soit pas de successeurs, leur prédiction sera entachée d'erreur. Ce phénomène s'appelle l'effet de bord. On utilise donc une fenêtre de Hamming, nulle en ses deux extrémités, pour diminuer l'influence de ces échantillons extrêmes.

Après cette pondération, chaque tranche va être passée dans un algorithme afin de calculer les coefficients a_i. Rappelons que ces coefficients sont ceux du dénominateur du filtre tous pôles dont la transmittance est l'enveloppe spectrale du signal.

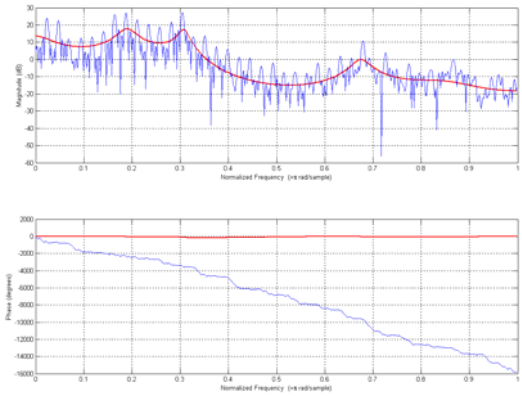

Figure 2 : spectre d'un signal (en bleu) et filtre AR (en rouge).

Ce calcul se base sur la résolution des équations de Yule-Walker, qui peuvent notamment être résolues par l'algorithme de Levinson et l'algorithme de Schur. Néanmoins, la fonction *"lpcauto"* de Matlab qui réalise ce calcul le fait par une méthode d'élimination gaussienne.

Cette fonction, utilisée pour un ordre p=10, retourne douze coefficients. Le premier coefficient est toujours égal à un, les dix suivants sont les a_i. Ces onze coefficients représentent le dénominateur du filtre autorégressif. Le dernier coefficient correspond, lui, au carré du gain G du filtre autorégressif (AR).

$$T_{AR}(z) = \frac{G}{1 + a_1 z^{-1} + \ldots + a_p z^{-p}}$$

Notons que ce douzième paramètre est l'énergie du résidu, c'est-à-dire l'énergie de la sortie qu'on obtient quand on fait passer le signal de la tranche analysée dans un filtre de transmittance $T_{AR}(z)^{-1}$. On appelle ce signal de sortie le résidu car le but de l'analyse LPC est de minimiser son énergie.

Pour notre programme, nous avons utilisé une autre manière de représenter les tranches de son. Plutôt que d'utiliser les a_i, dont de très faibles variations entraînent de forts changement de la transmittance, voire l'instabilité du filtre AR, on va utiliser les coefficients PARCOR (ou coefficients de réflexion) qu'on représente par k_i et qui sont calculés à partir des a_i grâce à la fonction "*lpcar2rf*".

Ces k_i ont plusieurs avantages. Tout d'abord ils doivent être bornés, entre +1 et -1. C'est une condition nécessaire et suffisante de stabilité du filtre AR[3].

Ensuite, une variation de leur valeur (tout en restant dans les limites de stabilité) modifie peu la transmittance du filtre AR.

Enfin, contrairement aux a_i ils ont une signification physique. Ils représentent les rapports de surface entre les p tubes successifs qui simulent le conduit vocal dans le modèle LPC.

Remarquons que la fonction "*lpcar2rf*" donne 11 coefficients à partir des 11 coefficients a_i, alors qu'il n'y a que 10 k_i. C'est parce que la fonction considère qu'il y a un "k_0" qui vaut toujours 1. Il ne sera donc pas nécessaire de le coder.

Lorsqu'on a les k_i, ils sont transmis au codeur.

3 PREACCENTUATION

Il serait plus logique de parler de la préaccentuation avant de parler de l'analyse LPC puisqu'elle la précède, mais l'utilité de cette préaccentuation se justifie par certaines observations qui peuvent être faites sur l'analyse LPC. C'est pourquoi nous l'expliquons si tard.

La préaccentuation consiste à faire passer les tranches de signal dans un filtre passe-haut du premier ordre, de transmittance

$$T(z) = 1 - \alpha z^{-1}$$

avec α positif inférieur à un (nous avons pris une valeur de 0,95)

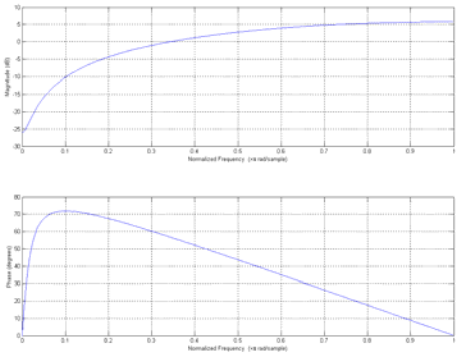

Figure 3 : transmittance du filtre de préaccentuation.

[3] N'oublions pas que ce filtre a pour coefficients au dénominateur les a_i, pas les k_i. Il faut donc utiliser la fonction inverse de "*lpcar2rf*" pour pouvoir étudier sa transmittance. Il s'agit de "*lpcrf2ar*".

Le but de cette préaccentuation est de diminuer l'influence des basses fréquences du signal et, par là, d'augmenter la précision de l'analyse LPC. En effet, ce filtrage va diminuer la hauteur relative entre les formants et le reste du spectre, il va "aplatir" le spectre. Cela va permettre de calculer un filtre AR qui collera mieux à l'enveloppe du spectre. En effet, les variations du spectre étant moindre, il sera plus facile de les suivre.

4 DETECTION DE PITCH

Pour la détection de la fréquence fondamentale de chaque tranche, nous avons utilisé une fonction trouvée sur le site Internet de Matlab, "*shrp*".

Cette fonction utilise un algorithme basé sur les rapports harmoniques-subharmoniques[4].

5 CODEUR

Le codage consiste en une quantification des différents paramètres que renvoie l'analyse LPC. Douze paramètres sont renvoyés pour chaque fenêtre d'analyse, il s'agit de la fréquence fondamentale, du gain et des coefficients du filtre autorégressif. Nous avons créé la fonction Matlab "*analyselpc*" qui utilise la fonction "*lpcauto*" et renvoie donc les coefficients a_i qui sont ensuite transformés en k_i grâce à la fonction Matlab "*lpcar2rf*". Ici, nous coderons plutôt les valeurs $10\log(\frac{1+k_i}{1-k_i})$ qui portent le nom de coefficients LARi[5]. Les erreurs de quantification commises seront du même ordre de grandeur pour tous les LARi, ce qui n'était pas le cas pour les k_i.

Le nombre de bits que nous utilisons pour coder chaque paramètre est semblable à celui utilisé par l'OTAN[6].

Pour les parties voisées, la fréquence fondamentale est codée sur sept bits de la façon suivante,

$$F_{0\,codée} = round(\frac{126}{350} \times (F_0 - 50)) + 1$$

On arrondit la valeur donnée par $F_{0codée}= f(F_0)$ où f est une droite qui fait passer les valeurs des fréquences allant de 50 à 400 Hz sur une échelle allant de 1 à 127 (fig. 4) puisque nous utilisons sept bits pour coder et que la valeur zéro est utilisée pour les sons non voisés.

[4] SHR : subharmonic-to-harmonic ratios
[5] Logarithm area ratios
[6] Lors des tests, nous avons essayé divers codages

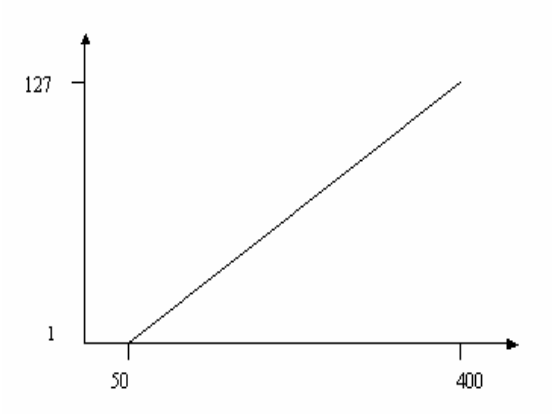

Figure 4 : fonction de compression de F_0. axe horizontal en Hz, axe vertical en valeurs quantifiées.

Le gain est codé sur cinq bits. Expérimentalement, il ne dépasse jamais la valeur de 1.2, on fait donc une transformation qui passe d'une échelle allant de 0 à 1.2 à une échelle allant de 0 à 32 :

$$G = round(\frac{G \times 32}{1{,}2}).$$

Les quatre premiers coefficients LARi sont codés sur cinq bits. Nous avons observé que les LARi étaient tous compris entre 1 et –1. Il suffit alors de faire une correspondance comme montré à la figure2.

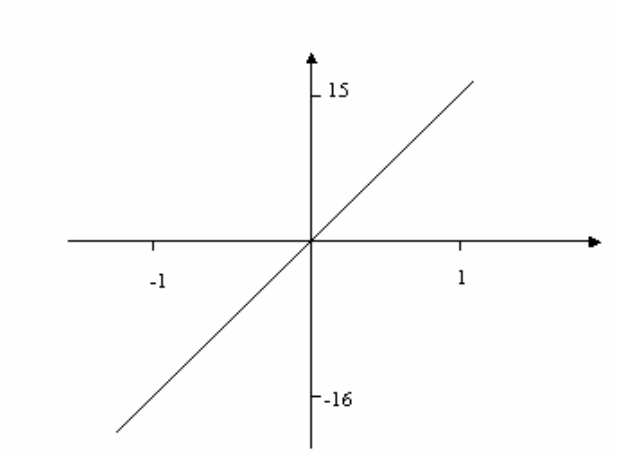

Figure 5 : droite de compression des LARi

Pour réaliser cette opération, on utilise la ligne de code suivante:

LARi=round(LARi*16)

Nous devons bien sûr arrondir la valeur de sortie de $LARi_{codée}$=f(LARi), où f est la droite de la figure2, pour qu'il y ait effectivement quantification.

Les quatre coefficients LARi suivants sont codés sur quatre bits et les deux derniers sur trois et deux bits respectivement de la même manière que les premiers coefficients. La droite de correspondance a néanmoins une pente plus faible quand il y a moins de bits pour la quantification:

LARi=round(LARi*8) pour 4 bits

LARi=round(LARi*4) pour 3 bits

LARi=round(LARi*2) pour 2 bits

Il est à noter que les LARi peuvent théoriquement avoir une valeur dans l'intervalle [-∞,∞] mais nous limiterons les transformations $LARi_{codée}$=f(LARi) dans des intervalles [-16,15], [-8,7], [-4,3], [-2,1] suivant le nombre de bits de quantification.
Pour les parties non voisées, la fréquence fondamentale est mise à zéro, le gain est codé sur cinq bits, les quatre premiers coefficients LARi sur cinq bits, les six derniers seront fixés à une valeur nulle, toujours avec la même méthode.

6 DECODEUR

Le décodage est l'opération inverse du codage, c'est-à-dire que l'on transforme les valeurs quantifiées obtenues par le codage en valeurs utilisables pour la synthèse. Pour le décodage des coefficients LARi, on applique la transformation suivante : $LARi_{décodée}$=f($LARi_{codée}$) où f est une droite de coefficient angulaire inverse par rapport au coefficient angulaire de la droite de codage:

LARi=LARi/16
LARi=LARi/8
LARi=LARi/4
LARi=LARi/2

Pour le décodage de la fréquence fondamentale, l'opération inverse doit tenir compte du fait que la droite ne passe pas par l'origine:

F0=(F0c-1)*350/126 +50

Enfin, le gain peut être décodé par le formule suivante

$$G = round(\frac{G \times 1,2}{32})$$

Il n'y a évidemment plus ici besoin d'arrondir les valeurs décodées.

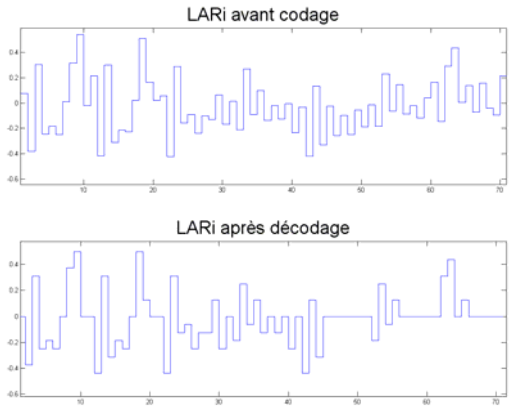

Figure 6 : figure du haut, LARi avant codage, figure du bas, LARi après codage-décodage

7 RECONSTRUCTION

La synthèse est l'étape qui va reconstruire un signal de parole à partir des paramètres calculés par l'analyse LPC. Le schéma de principe est représenté à la figure 7

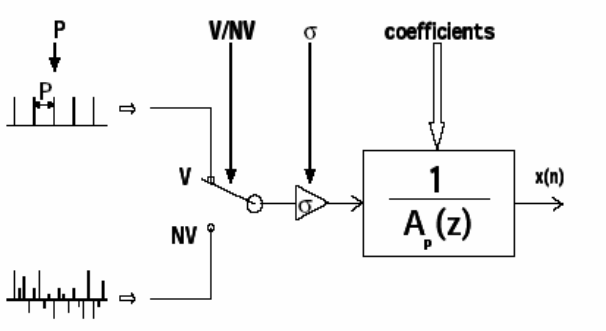

Figure 7 : synthétiseur LPC (cf.[1])

Un train d'impulsions ou un bruit blanc gaussien est passé dans le filtre AR dont les coefficients du dénominateur sont les coefficients a_i , le numérateur est le gain G qui est la racine carré de l'énergie du résidu (ce qui explique la notation σ du schéma).

La détection de la fréquence fondamentale lors de l'analyse LPC permet de choisir l'entrée du filtre. En effet, l'algorithme utilisé renvoie, soit une valeur de fréquence qui sera la fréquence des impulsions du train, soit zéro. Dans le deuxième cas, on entre un bruit blanc gaussien de moyenne nulle et de variance unitaire dans le filtre.

Les actions décrites précédemment sont effectuées pour chaque tranche d'analyse indépendamment. A la sortie du filtre AR on a donc une reconstitution d'une tranche à la fois. Il faut ensuite additionner ces tranches pour retrouver un signal de même longueur que le signal avant analyse et codage. Pour se faire, on passe les sorties du filtre AR dans des fenêtres de pondération que l'on additionne ensuite. Etant donné que lors de l'analyse on découpait le signal en tranches qui se recouvraient, la synthèse doit bien sûr tenir compte de ce recouvrement et additionner les fenêtres avec le même recouvrement comme le montre la figure 8

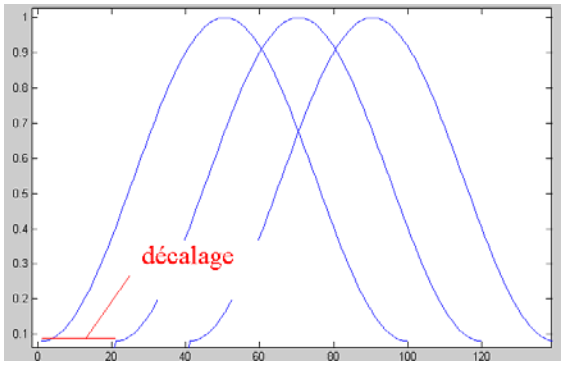

Figure 8 : fenêtres de Hamming et superposition.

Avant l'analyse, on passait le signal dans un filtre d'accentuation pour faciliter le calcul des coefficients du filtre AR. Avant d'additionner les fenêtres, il est nécessaire de toutes les passer dans un filtre de désaccentuation pour annuler l'effet du filtre d'accentuation.

Il est très important de remarquer que chaque tranche que l'on reconstitue n'a en commun avec la tranche originelle que l'enveloppe spectrale. On peut d'ailleurs voir à la figure 9 que l'aspect temporel diffère fortement entre le signal initial (en haut de la figure) et le signal reconstruit (en bas)

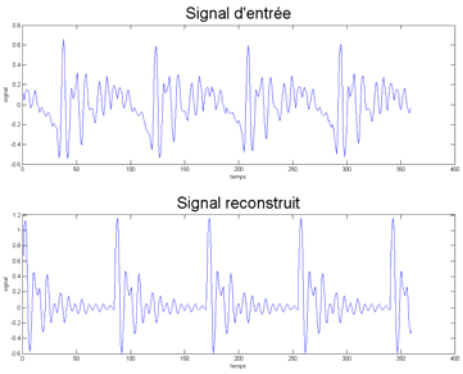

Figure 9 : signal d'entrée et signal de sortie.

8 TESTS

Dans cette partie, lorsque nous dirons qu'une phrase est moins bonne ou moins compréhensible qu'une autre, nous faisons référence à nos propres perceptions des sons. D'autres personnes pourraient avoir un avis différent. Nous avons néanmoins essayé d'être les plus objectifs possible.

Nous avons maintenant tous les éléments pour pouvoir tester un codeur LPC : un analyseur qui donne des paramètres, un codeur qui code ces derniers, un décodeur et finalement un synthétiseur qui reforme un signal de parole.

Nous avons enregistré des phrases sur Matlab avec la fonction *"wavrecord"*, toutes les phrases étaient échantillonnées à 8000 Hz. Cette fréquence d'échantillonnage était bien celle à utiliser puisque l'ordre d'un codeur LPC est donné par F_e (en kHz) +2 et nous voulions bien un LPC10.

Les différentes phrases étaient ensuite traitées successivement par les fonctions Matlab d'analyse, de codage, de décodage et de synthèse. Le son obtenu en sortie du module de synthèse devait être compréhensible et restituer la même phrase que celle préenregistrée.

Il s'est avéré que le son de sortie dépendait fortement des longueurs des fenêtres d'analyse.

Des fenêtres d'analyse de 30 ms espacées de 10 ms permettaient de synthétiser une phrase tout à fait compréhensible et fidèle à celle d'entrée, c'est-à-dire que la sortie permettait de reconnaître l'entrée. Ceci s'explique par le fait que l'analyse et la synthèse doivent se faire sur des parties de parole stationnaires. La stationnarité n'est jamais obtenue en parole, mais il faut un morceau de parole qui s'en approche, on parle alors de pseudo stationnarité. Si les fenêtres d'analyse sont trop petites ou trop grandes, on s'éloigne de la stationnarité. Des fenêtres de 10 ms espacées de 5 ms ne donnent pas de bons résultats et des fenêtres de 300 ms espacées de 150 ms ne donnent rien d'acceptable non plus. Par différents essais, nous avons déterminé que les limites pour les tailles de fenêtres étaient situées entre 20 et 60 ms. L'espacement entre fenêtre doit, quant à lui, être inférieur à la taille d'une fenêtre.

En ce qui concerne les résultats compréhensibles, les phrases d'entrée apparaissaient, en sortie, distordues. En effet, le son était métallique et accompagné de chuintements. Ces caractéristiques sont typiques d'un codage LPC.

Il est à souligner que le codeur LPC fonctionne moins bien pour les son nasalisés. Ceci est dû au fait que les sons nasalisés présentent des antiformants c'est-à-dire un creux dans le spectre. Lors du calcul du filtre AR, celui-ci ne suit pas ces creux et on perd donc de l'information (figure 10). Nous avons testé notre codeur avec une phrase contenant beaucoup de son nasalisés ("les petits lapons aux pompons pimpants en peau de lapin") et la sortie du synthétiseur était effectivement moins bonne qu'avec des phrases normales.

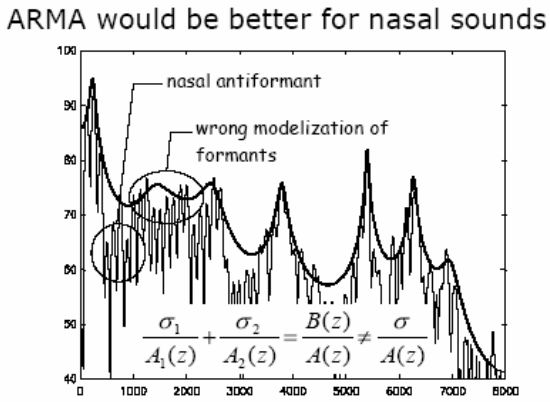

Figure 10 : spectre d'un son nasalisé (cf. [1])

Nous avons également essayé de diminuer le nombre de bits de codage pour avoir un débit encore plus faible, mais les phrases synthétisées en sortie étaient de qualité médiocre, voire incompréhensibles.

9 RAPPORT SIGNAL A BRUIT

Dans le but de déterminer une méthode objective de mesure de qualité des sons que nous reconstituons, nous avons essayé de définir un rapport signal à bruit.

Pour ce faire, il nous faut tout d'abord définir ce qu'est le bruit. Malheureusement, comme nous l'avons expliqué plus haut, les signaux d'entrée et de sortie n'ont rien en commun dans leur aspect temporel, on ne peut donc pas simplement définir le bruit comme étant la différence entre le signal d'entrée et le signal de sortie. La seule définition exacte de ce qu'est le bruit lors d'une analyse LPC serait basée sur des facteurs propres à la perception des sons par l'Homme, mais nous ne disposons pas d'informations suffisantes pour le définir.

Par contre, comme les spectres sont fort proches, nous avons décidé de poser une définition du bruit au niveau spectral. Ce "bruit spectral" serait donc la différence entre le spectre d'entrée et le spectre de sortie.

Faire la différence des spectres donnés par la FFT n'aurait pas plus de sens que de faire la différence entre les signaux temporels. En effet, ces deux opérations sont équivalentes.

Nous allons donc faire la différence entre les modules des spectres d'entrée et de sortie en dB.

$$B_{dB}(\omega) = 20 \times \log_{10}(|I(\omega)|) - 20 \times \log_{10}(|O(\omega)|)$$

Ensuite, de la même manière qu'en temporel, où on utilise la variance pour représenter une suite d'échantillons, il faut associer une valeur représentative aux vecteurs de valeurs des spectres. Par exemple, si on fait la FFT d'un signal sur 256 points, on obtient 256 valeurs du spectre, donc un échantillonnage du spectre[7]. Il faut donc définir une valeur représentative de ces 256 points. Une équivalence spectrale à la variance du signal.

Nous allons calculer l'intégrale du spectre

$$\sum_{k=1}^{N} F(\omega_k) \cdot \Delta\omega$$

avec $\Delta\omega = \frac{2\pi F_e}{N}$

Les termes constants (F_e, 2π) seront les mêmes pour toutes les courbes, ils introduisent juste un facteur multiplicatif. Ils n'ont donc aucune importance dans le calcul qui va suivre. Par ailleurs, il est très important de remarquer que cette intégrale ne dépend pas de la valeur de N, le nombre de points sur lesquels la FFT a été réalisée, car si N varie, le nombre de termes de la somme varie et $\Delta\omega$ aussi, les deux effets s'annulent.

Ensuite, pour calculer le rapport signal à bruit, nous allons diviser l'intégrale correspondant au spectre du signal d'entrée par celle du spectre de bruit.

On obtient donc la formule suivante :

$$RSB = \frac{\sum 20 \times \log_{10}(|I(\omega)|)}{\sum [20 \times \log_{10}(|I(\omega)|) - 20 \times \log_{10}(|O(\omega)|)]}$$

Les $\Delta\omega$ se sont simplifiés, le numérateur et le dénominateur dépendent tous les deux de N à présent. Mais est-ce que le résultat global dépend de N ? La question reste posée. En tout cas, les valeurs de RSB calculées pour différentes valeurs de L ne reflètent pas du tout l'impression auditive ressentie sur la qualité de la reconstruction du signal.

Sur le graphique suivant, on peut voir l'évolution de la qualité du signal telle que perçue par les auditeurs. La courbe en elle-même est empirique et n'a aucune valeur scientifique, elle est juste là pour donner une idée de cette évolution.

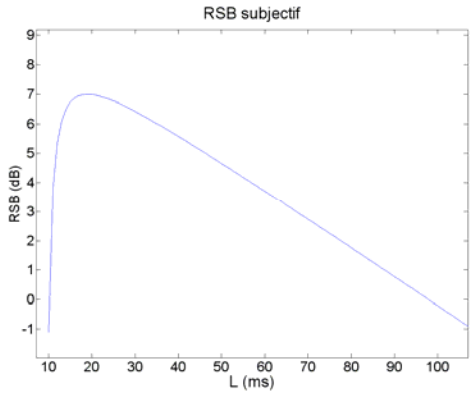

Figure 11 : évolution de la qualité du signal du point de vue des auditeurs. L variable, E=L/2.

Sur le graphique de la figure 12, on peut voir le rapport signal à bruit tel qu'il est calculé par la méthode expliquée. Si on se réfère à ce graphique, la meilleure qualité est obtenue pour de grandes valeurs de L (100 ms), ce qui est absolument faux.

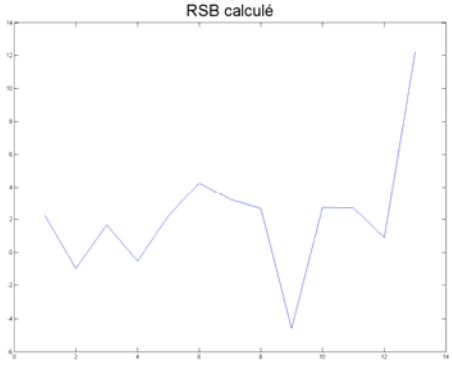

Figure 12 : évolution de la qualité du signal en fonction de L, par calcul.

10 BRUIT DE QUANTIFICATION

Il nous a également paru intéressant de chercher à visualiser l'effet du codage, de déterminer le bruit de quantification. Pour cela, nous avons voulu

[7] à l'erreur induite par la convolution de ce spectre avec celui de la fenêtre d'observation près.

comparer le signal de sortie avec codage des paramètres et le signal de sortie qui serait produit si on ne codait pas les paramètres.

Comme il s'agit de deux signaux de sortie, le problème de la différence des signaux dans le domaine temporel ne se pose pas, nous allons comparer des signaux qui ont grosso modo le même aspect temporel, aux erreurs induites par la quantification près.

Nous allons donc décomposer les deux signaux en tranches de 45 ms, décalées de 22,5 ms et calculer un RSB pour chaque tranche. Puis faire une moyenne des RSB sur tout le signal.

Le bruit sur une tranche est ici défini comme étant la tranche du signal de sortie sans codage moins la tranche du signal de sortie avec codage.

On calcule la variance du signal non codé et celle du bruit et ensuite, on calcule le rapport signal à bruit segmental (en dB) en faisant

$$RSB_{SEG\,(dB)} = 10 \times \log_{10}\left(\frac{\sigma^2_{sortie\ sans\ codage}}{\sigma^2_{bruit}}\right)$$

On effectue ce calcul pour toutes les tranches et on fait la moyenne de tous les RSB segmentaux obtenus.

Nous avons effectué le calcul pour différents codages et les résultats obtenus correspondent assez bien à ce que nous entendons. Il faut toutefois préciser que pour des codages proches du codage selon la norme de l'OTAN et pour tous les codages avec un plus grand nombre de bits, la différence est quasi imperceptible pour l'oreille humaine.

La figure 13 représente un tel RSB en fonction du nombre de bits consacrés au quatre premiers LARi.

Le nombre de bits consacrés aux autres paramètres est dans une même proportion[8] que pour la norme OTAN - LPC10.

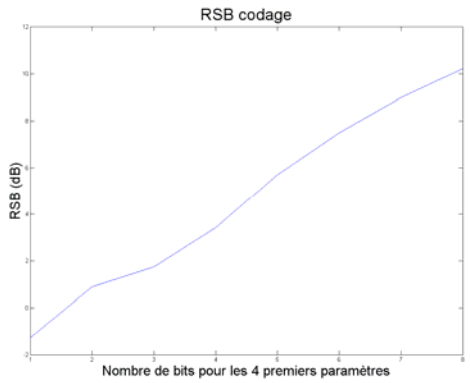

Figure 13 : RSB dû au codage.

On voit donc bien que le RSB augmente avec le nombre de bits utilisés, ce qui est logique.

[8] Sauf si le nombre de bits obtenu par cette proportion est nul. Dans ce cas, on code le LARi sur un seul bit.

11 CONCLUSIONS

Un codeur LPC permet de diminuer de manière significative le débit binaire associé. Au prix d'une chute de la qualité de la parole. Le tout étant de ne pas diminuer trop ce débit afin de conserver l'intelligibilité du signal.

Une simulation de ce système sur Matlab nous a permis de bien comprendre les principes de fonctionnement sans devoir trop s'attarder sur l'écriture de fonctions.

Lors de la réalisation de ce programme nous n'avons pas rencontré trop de difficultés, nous avons surtout consacré beaucoup de temps aux tests concernant le codage et la largeur de fenêtre, ainsi qu'au rapport signal à bruit. Ce dernier nous a posé énormément de problèmes, en effet la définition du bruit pour un codeur LPC n'est pas clairement établie, il faudrait utiliser un modèle de perception humaine pour mener une telle opération à bien.

12 BIBLIOGRAPHIE

[1] Thierry Dutoit, Notes de cours de traitement de la parole, 5ème année.

[2] Thierry Dutoit, Traitement du signal, Notes de cours.

[3] Thierry Dutoit, Traitement du signal, Projet de 4ème année, 2003-2004.

[4] www.mathworks.com