UNIVERSITE SAAD DAHLAB DE BLIDA

Faculté de Technologie

Département d'Electronique

THÈSE DE DOCTORAT

Spécialité : Electronique

CONTRIBUTION AUX TECHNIQUES DE QUANTIFICATION VECTORIELLE PAR RESEAUX REGULIERS ET PAR TREILLIS APPLICATION AU CODAGE DE LA PAROLE

Par

Abdellah KADDAI

devant le jury composé de :

A. Guessoum	Professeur, U. de Blida	Président
M. Bensebti	Professeur, U. de Blida	Examinateur
D. Berkani	Professeur, E.N.P.	Examinateur
H. Sayoud	Professeur, U.S.T.H.B.	Examinateur
M. Halimi	Maître de Recherche « A », C.R.S.T.S.C., Alger	Rapporteur

Blida, Septembre 2012

À mes proches.

REMERCIEMENTS

Je tiens tout d’abord à remercier mon directeur de thèse, Monsieur HALIMI Mohammed, Maître de Recherche « A » au CRSTSC de Chéraga, d’avoir accepté de diriger ce travail avec beaucoup de disponibilité et d’efficacité et pour les conseils et encouragements qu’il a su me prodiguer pendant toute la durée de cette thèse.

Je souhaite aussi exprimer ma gratitude aux membres du jury, le Professeur GUESSOUM Abderrazak, le Professeur BENSEBTI Messaoud, le Professeur BERKANI Daoud et le Professeur SAYOUD Halim pour avoir accepté d’évaluer ma thèse.

Enfin, je remercie les membres de ma famille pour leur compréhension et leur soutien indéfectible.

J’espère que cette thèse est à la hauteur de ce que je dois moralement à tous ces gens.

الملـــخـــص

يعتبر التكميم الشعاعي المرمز بالشبكة " ت.س.ف.ك" جد فعال، خاصة وأنه ناتج عن تركيب تقنيتين فعالتين في انضغاط الإشارات، ألا وهما التكميم المرمز بالشبكة "ت.س.ك" والتكميم الشعاعي. لكن هاته التقنية محدودة عندما يكون قاموس الشبكة ذو حجم كبير، حيث أن تكوينه مكلف من حيث المدة المستغرقة في الحسابات من جهة، ومن جهة أخرى يشكل تخزينه في حد ذاته عائق آخر، لكن العائق الرئيس يظهر على مستوى تكميم الإشارة. فإذا كان القاموس ذو حجم جد كبير، فالبحث عن أفضل الطرق في المرمز "ت.س.ف.ك" باستعمال خوارزمية "فيتاربي" يعتبر جد معقد في تطبيقات الزمن الحقيقي. كل هاته السلبيات أو المعوقات نستطيع تجنبها بإدخال القاموس الجبري (شبكة منظمة للنقاط) في المرمز "ت.س.ف.ك" والذي ينتج عنه تقنية جديدة في التكميم، تسمى التكميم الشعاعي الجبري بالشبكة "أ.ت.ف.ك". الإطار التطبيقي المختار لهذا التصور هو تكميم ترددات خطوط الطيف "ل.س.ف" في الشريط الموسع.

النتائج المتحصل عليها في هاته الأطروحة تبين أن "أ.ت.ف.ك" تحصل على النوعية الشفافة عند 47 عنصر ثنائي لكل مجموعة من العينات، كما أن لها نفس فعالية التكميم الشعاعي المجزأ "أس. ف. ك" والمكمم "ت.س.ف.ك"، ولكن مع الأفضلية المعتبرة في خفض درجة التعقيد (سواءا في الحسابات أو في التخزين).

كلمات مفتاحية: التكميم المرمز بالشبكة، قاموس جبري، شبكة منظمة للنقاط، ترميز الكلام في الشريط الموسع.

RESUME

La quantification vectorielle codée par treillis (TCVQ) est assez performante étant donné qu'elle fait combiner deux puissantes techniques de compression des signaux, à savoir, la quantification codée par treillis (TCQ) et la quantification vectorielle. Mais cette technique a des limites lorsque le dictionnaire (du treillis) requis nécessite une grande taille. La constitution d'un dictionnaire de taille élevée est très lourde en temps de calcul. Le stockage d'un tel dictionnaire pose aussi problème. Mais l'inconvénient apparaît au niveau de la quantification du signal. Si le dictionnaire est trop grand, la recherche du meilleur chemin dans le codeur TCVQ en utilisant l'algorithme de Viterbi sera trop complexe pour une application en temps réel. Ces inconvénients peuvent être évités grâce à l'introduction d'un dictionnaire algébrique (réseau régulier de points) au sein du codeur TCVQ et qui engendre une nouvelle technique de quantification, appelée Quantification vectorielle algébrique en treillis (ATVQ : Algebraic Trellis Vector Quantization). Le cadre d'application choisi de cette approche est celui de la quantification des fréquences de raies spectrales (LSF : Line Spectral Frequencies) en bande élargie.

Les résultats de cette thèse montrent que l'ATVQ atteint la qualité transparente à 47 bits/trame et donne pratiquement les mêmes performances que la quantification vectorielle fendue (SVQ) et la TCVQ, mais avec une réduction considérable de la complexité (en stockage et en calcul).

Mots-clés : Quantification codée par treillis, dictionnaire algébrique, réseaux réguliers de points, LSF, codage de parole en bande élargie.

ABSTRACT

The Trellis Coded Vector Quantization (TCVQ) is rather powerful in view of the fact that it combines two powerful techniques of signals compression, namely, the Trellis Coded Quantization (TCQ) and the Vector Quantization (VQ). However the exponentially growing of the codebook size poses a formidable barrier. The construction of high size codebook is very consuming in computing time. The storage of such codebook poses also problem. But the most drawbacks appear on the signal quantization. If the codebook is too large, the search of the best path in TCVQ quantizer by using the Viterbi algorithm will be too complex for an application in real time. These disadvantages can be avoided by the introduction of an algebraic codebook (lattice) within the TCVQ quantizer which generates a new technique of quantization, called Algebraic Trellis Vector Quantization (ATVQ). The selected application framework of this approach is that of the wideband quantization of the Spectral Line Frequencies (LSF).

The results of this thesis show that the ATVQ reached transparent quality at 47 bits/frame and gives practically the same performances as the split vector quantization (SVQ) and the TCVQ, but with a considerable reduction of complexity (in storage and calculation).

Key words: Trellis coded quantization, algebraic codebook, lattice, LSF, wideband speech coding.

TABLE DES MATIERES

LISTE DES ILLUSTRATIONS, GRAPHIQUES ET TABLEAUX

INTRODUCTION

Actuellement, les télécommunications numériques se sont largement imposées, grâce notamment à leur grande robustesse vis-à-vis des perturbations apportées par le canal. A leur début, les communications numériques possédaient portant un handicap majeur, elles consommaient une bande de fréquence beaucoup plus importante que les communications analogiques. Aujourd'hui, ce n'est plus le cas grâce notamment au développement des techniques performantes de compression des signaux.

La numérisation et la compression des signaux, telles que la parole, la musique et l'image, ont fait l'objet de très nombreuses études depuis que Claude Shannon a posé les bases théoriques de la théorie de l'information dans les années 1940 et 1948. Ces études couvrent aujourd'hui un vaste champ et se concentrent principalement sur la diminution de la distorsion et l'économie des ressources physiques de transmission ou de stockage.

La quantification vectorielle a été considérablement développée depuis les années 1979 et a émergé récemment comme une puissante technique de compression des signaux. La quantification vectorielle n'est pas simplement une généralisation de la quantification scalaire (unidimensionnelle). Elle exploite directement la corrélation existant dans le signal. La théorie de l'information démontre que les performances de quantification peuvent toujours être améliorées en augmentant la dimension du vecteur à quantifier. Mais la quantification vectorielle classique, c'est-à-dire la quantification vectorielle statistique, a des limites lorsque le dictionnaire requis nécessite d'avoir une grande taille. La

constitution d’un dictionnaire de taille élevée est très lourde en temps de calcul. Un autre problème engendré concerne le stockage d’un tel dictionnaire. Mais l’inconvénient le plus gênant apparaît au niveau de la quantification du signal. Si le dictionnaire est trop grand, la recherche de la meilleure valeur type dans le quantificateur sera trop complexe pour une application en temps réel. Par conséquent, il est nécessaire de développer de nouvelles techniques de quantification vectorielle.

La quantification vectorielle par réseaux réguliers, appelée aussi quantification algébrique, est l’extension multidimensionnelle de la quantification scalaire uniforme. Ces réseaux, grâce à leurs structures régulières, permettent d’effectuer une quantification très rapide, c'est-à-dire parcourir l’ensemble des valeurs disponibles pour le quantificateur (le temps de codage est fonction de la dimension du vecteur à quantifier, et non du nombre de points présents dans le réseau). Il n’y a pas besoin de stocker de dictionnaire. Par contre, une telle quantification ne sera vraiment performante que si le signal à coder est uniformément distribué, ce qui n’est pas forcément le cas des signaux de parole, de musique ou d’image. Pour se placer dans de bonnes conditions, il est néanmoins possible de faire subir aux signaux une transformation visant à les équirépartir, dans l’espace de représentation, avant de procéder à la quantification.

Les travaux de Ungerboeck sur la TCM (Trellis Coded Modulation) ont inspiré Marcellin et Fischer qui ont proposé d’utiliser le treillis avec des alphabets scalaires dans la quantification de source. Ils ont démontré que pour un débit donné, la TCQ (Trellis Coded Quantization) permet d’atteindre un rapport signal sur bruit comparable à celui des meilleures techniques de quantification, en particulier la quantification vectorielle, sans impliquer une complexité aussi grande que dans le cas de ces quantificateurs haut de gamme.

L’article de Marcellin et Fischer (1990) a lancé le développement de la quantification basée sur des treillis. Plusieurs variantes et applications basées sur les idées de la TCQ ont vu le jour. A titre d’information, depuis une décennie - dans le contexte du codage d'images - la TCQ a été étendue pour le codage entropique (entropy-constrained TCQ) et au codage multi-débits de type hiérarchique, à descriptions multiples, etc. A noter que la TCQ est utilisée dans l'annexe J du G.728 (Codeur de parole Low-Delay Code

Excited Linear Prediction), en plus de la norme de compression d'images JPEG2000. La TCQ a été appliquée également à la quantification des coefficients de prédiction linéaire.

L'objet de cette thèse est de développer une nouvelle technique de quantification permettant de réduire la complexité de la quantification vectorielle en se basant sur les réseaux réguliers et la TCQ et de l'appliquer dans le contexte de codage de la parole en s'appuyant sur les travaux réalisés dans la littérature.

Cette thèse est organisée comme suit :

- Le chapitre 1 décrit les notions de base concernant le codage de la parole en s'orientant plus particulièrement vers les paires de raies spectrale (LSF) à coder.
- La première partie du chapitre 2 est un rappel sur la quantification scalaire et vectorielle. La seconde partie présente la quantification codée par treillis. Nous commençons tout d'abord par donner quelques notions de la théorie débit-distorsion avec contrainte sur la taille de l'alphabet de sortie. On décrit ensuite comment fonctionne la TCQ et comment elle a été conçue. Finalement, les performances de la TCQ seront données.
- La description de la nouvelle technique de quantification basée sur les treillis et sur les réseaux réguliers et son application à la quantification des paramètres LSF font l'objet du chapitre 3. Après la présentation de la structure de cette nouvelle technique, nous décrivons le dictionnaire algébrique du treillis, les algorithmes de détermination du plus proche voisin ainsi que l'algorithme de l'indexage. Finalement, les résultats des simulations sont exposés.

CHAPITRE 1
CODAGE DE LA PAROLE

Depuis quatre décennies, beaucoup de travaux se sont penchés sur les techniques de codage de la parole et ont abouti à des progrès considérables. Le domaine des applications pratiques utilisant les résultats des chercheurs est de plus en plus large. Le codage de la parole est essentiel dans les transmissions numériques des messages du réseau téléphonique, dans la radiotéléphonie mobile, dans le stockage numérique du signal et dans les applications multimédia.

L'exigence de la qualité des codeurs varie selon l'application. Dans certains cas, on demande une excellente qualité, dans d'autres cas, on est satisfait si la parole reconstruite est compréhensible. Le but des recherches dans tous les domaines est de réduire le débit nécessaire tout en maintenant la qualité ou inversement, avoir une meilleure performance pour un débit fixé. Outre le débit, la complexité du codeur est un aspect important surtout dans les applications en temps réel.

Pour obtenir de bonnes performances de compression d'un signal, il est nécessaire de bien connaître les caractéristiques de ce dernier et la façon dont il est perçu. Dans le cas de la parole, il s'agira de bien modéliser l'appareil vocal et de prendre en compte les caractéristiques de l'audition humaine.

1.1 Description de l'appareil vocal

Le signal de parole est complexe et possède de nombreuses caractéristiques qui permettent de le modéliser correctement. Le signal de parole est produit par un ensemble d'organes constituant l'appareil vocal. Ce système comporte les poumons, la trachée artère, le larynx, les cordes vocales et se termine par les cavités buccale et nasale (Figure 1.1).

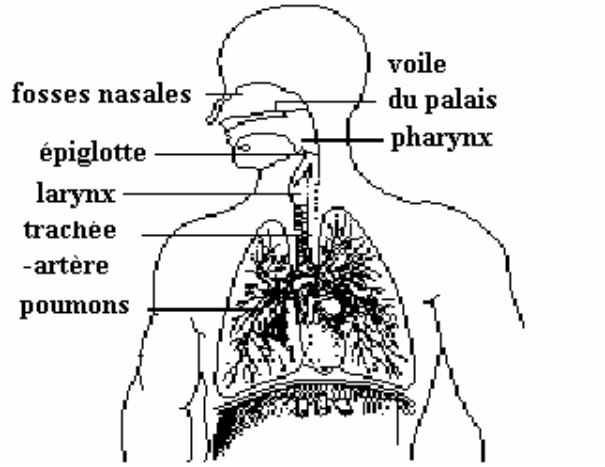


Figure 1.1 : Appareil phonatoire humain.

Les poumons fournissent l'énergie nécessaire à la phonation en insufflant l'air dans la trachée artère jusqu'au larynx. En fonction de l'ouverture des cordes vocales, on peut obtenir généralement deux principaux types de sons : voisé ou non-voisé. Pour la production des sons non-voisés, l'air passe librement à travers les cordes vocales. Au contraire, les sons voisés sont crées par une vibration périodique des cordes vocales. La fréquence de vibration est appelée *pitch* ou *fréquence fondamentale* et elle varie en fonction du locuteur. Typiquement, elle varie de 50 à 200 Hz pour les hommes, de 150 à 450 Hz pour les femmes et de 200 à 600 Hz pour les enfants. Les sons ainsi créés passent ensuite dans le conduit vocal qui joue un rôle de résonateur en privilégiant certaines zones fréquentielles, *les formants*, et en affaiblissant d'autres zones, *les anti-formants* ou *vallées*. Les formants et les vallées correspondent respectivement aux pôles et aux zéros de la fonction de transfert associée au conduit vocal.

La Figure 1.2 présente un exemple d'un signal de parole voisée et non voisée, tiré de la phrase: «*She had your dark suit in greasy wash water all year*».

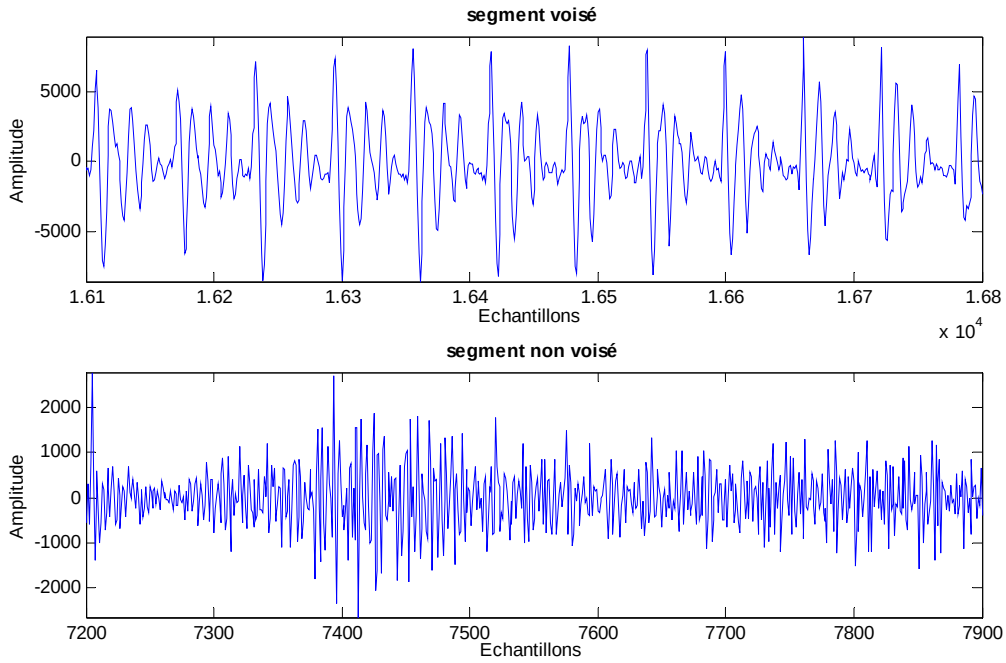


Figure 1.2 : Exemple d'un signal de parole voisée et non voisée.

La connaissance du système de production des sons voisés et non-voisés permet d'obtenir une bonne modélisation du signal de parole. Un système de modélisation classique (appelé modèle source - filtre) est représenté sur la Figure 1.3. Le filtre de synthèse peut être excité par un bruit blanc pour obtenir un son non-voisé ou par un train d'impulsion (Peigne de Dirac) pour obtenir un son voisé. La multiplication par un gain représente l'énergie avec laquelle l'air est expulsé du système respiratoire et le filtre de synthèse représente le conduit vocal.

Cette modélisation est assez simple et aura des performances restreintes. En effet l'excitation est seulement à deux états, voisée et non-voisée. Des modèles source – filtre plus évolués existent et permettent de meilleures performances. Nous allant voir par la suite les grandes catégories des codeurs de parole existants et les différentes méthodes de représentation associées. Dans un premier temps, nous allons présenter les caractéristiques fondamentales du signal de parole pouvant être modélisées par un codage prédictif qui est inclus dans de nombreux codeurs de parole actuels.

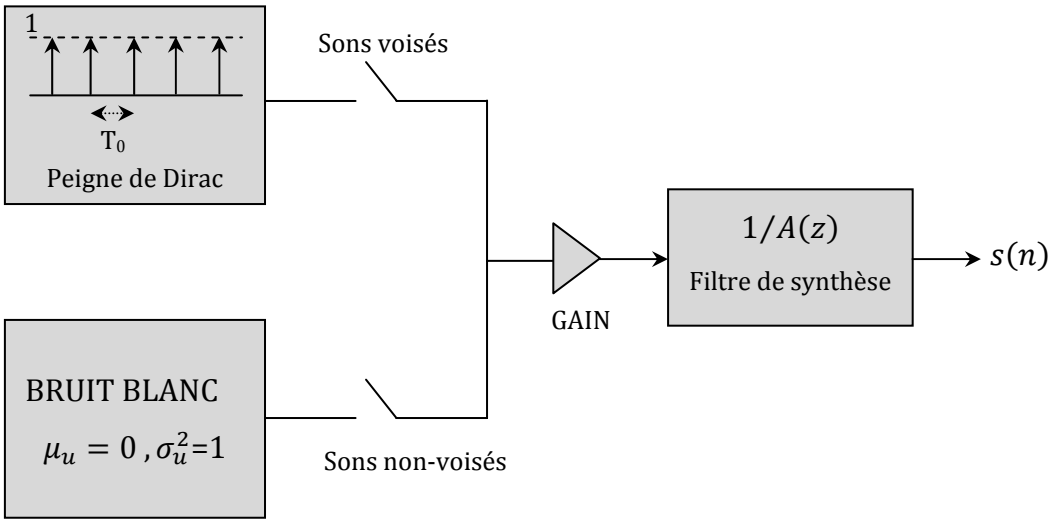


Figure 1.3 : Modèle simplifié de production de la parole.

1.2 Modélisation de l'appareil vocal

Le rôle de l'appareil vocal est de produire des sons audibles et intelligibles. Pour ce faire, l'état du canal évolue suivant le son à produire. La fonction de transfert de l'appareil vocal peut être modélisée par un filtre ARMA[1] :

$$H(z) = \frac{N(z)}{D(z)} \tag{1.1}$$

où $N(z)$ et $D(z)$ sont des polynômes en $1/z$ d'ordre respectif q et p. Les racines de $N(z)$ et $D(z)$ donnent respectivement les zéros et les pôles de la fonction de transfert $H(z)$. En pratique, on peut simplifier ce modèle sans dégradation significative de la qualité. On se contente d'un modèle ne contenant que des pôles, à savoir un modèle AR[2].

Le signal de parole est corrélé. Le filtre $H^{-1}(z)$ permet donc d'éliminer les redondances présentes sur ce type de signaux.

On peut distinguer deux types de corrélation sur le signal de parole : la corrélation à court terme qui prend en compte les informations du signal passé récent, et la corrélation à long terme qui va « chercher » dans un passé plus lointain. Le filtre de synthèse est donc

---

[1] ARMA : **A**uto**R**égressif à **M**oyenne **A**justée.

[2] AR : **A**uto**R**égressif.

séparé en deux analyses : court terme et long terme. On peut considérer que le signal de parole est quasi-stationnaire sur des durées allant jusqu'à 30 ms [1]. Les analyses sont généralement effectuées sur des trames de 20 ms.

Si l'on considère, qu'après filtrage inverse $H^{-1}(z)$ d'un signal de parole, la corrélation du signal a complètement été éliminée (en théorie, l'ordre du filtre devrait être infini), le signal en sortie (signal résiduel) est un bruit blanc. Dans ce cas, le signal de parole reconstruit est le résultat du filtrage du bruit blanc par le filtre $H(z)$.

1.2.1 Prédiction à court terme

La prédiction à court terme permet de modéliser l'enveloppe spectrale de la parole qui est la transformée de la réponse impulsionnelle du conduit vocal. Ce type de codage repose sur la constatation que les échantillons adjacents de la parole sont fortement corrélés. Dans l'hypothèse où les échantillons sont liés par une relation linéaire, l'échantillon présent peut être prédit par une combinaison linéaire de ses échantillons voisins. On parle alors de codage prédictif linéaire LPC[3].

L'analyse à court-terme cherche à prédire l'échantillon $s(n)$ à l'instant n par une combinaison linéaire des p échantillons précédents du signal de parole $s(n)$:

$$\tilde{s}(n) = -\sum_{i=1}^{p} a_i s(n-i) \qquad (1.2)$$

où p est l'ordre de prédiction, les a_k représentent les coefficients de la prédiction linéaire et $\tilde{s}(n)$ est la prédiction de $s(n)$. L'ordre p de prédiction est relativement petit car la prédiction à court terme n'exploite seulement que la redondance des échantillons voisins. Pour un codage en bande étroite, c.-à-d. pour une fréquence d'échantillonnage de 8 kHz, on utilise généralement une prédiction d'ordre 10 comme c'est le cas pour la norme G. 729 [2]. Pour un codage en bande élargie, c. –à-d. pour une fréquence d'échantillonnage de 16 kHz, on utilise plutôt un ordre de 16.

La prédiction (1.2) est une façon efficace de représenter la corrélation des échantillons. L'observation passée permet de réduire l'incertitude sur l'échantillon présent à coder, ce

---

[3] LPC : **L**inear **P**redictive **C**oding.

qui engendre une réduction du nombre de bits nécessaire pour transmettre l'information. Le résidu de la prédiction est obtenu en soustrayant à l'échantillon présent la prédiction $\tilde{s}(n)$. Il correspond alors à l'excitation du modèle source-filtre.

$$e(n) = s(n) - \tilde{s}(n) = s(n) + \sum_{i=1}^{p} a_i s(n-i) \tag{1.3}$$

Après la transformation en z, la relation précédente donne :

$$E(z) = S(z)A(z) \tag{1.4}$$

avec

$$A(z) = 1 + \sum_{i=1}^{p} a_i z^{-i} \tag{1.5}$$

La fonction de transfert du canal vocal est modélisée par un filtre tous pôles $H(z) = 1/A(z)$.

$$S(z) = H(z)E(z) = \frac{1}{A(z)} E(z) \tag{1.6}$$

Le filtre d'analyse $A(z)$ excité par le signal original $s(n)$ produit l'erreur de prédiction $e(n)$. Les coefficients a_k optimaux sont calculés en minimisant l'erreur quadratique moyenne (1.7) par rapport aux coefficients a_j avec $j = 1, \cdots, p$.

$$\text{Min } E[e^2(n)] \tag{1.7}$$

On obtient alors le système d'équations suivant :

$$\sum_{k=1}^{p} a_k R_s(i-k) = -R_s(i)\,, \qquad i = 1, \cdots, p \tag{1.8}$$

où R_s la fonction d'autocorrélation du signal de parole supposé stationnaire.

Dans les codeurs paramétriques et hybrides, comme nous le verrons par la suite, on quantifie et transmet entre autres les paramètres de l'analyse LPC. Les coefficients (quantifiés) de prédiction linéaire ne permettent pas d'assurer facilement la stabilité du filtre $1/A(z)$ et la qualité est conservée au prix d'une augmentation importante du débit binaire. C'est pourquoi d'autres paramètres, mathématiquement équivalents aux

coefficients de prédiction linéaire, ont été introduits car ils possèdent des propriétés permettant une bonne quantification et assurant la stabilité du filtre.

1.2.1.1 Coefficients de corrélation partielle

La matrice du système d'équations (1.8) est une matrice de Toeplitz. Nous pouvons donc appliquer l'algorithme de Levinson - Durbin [3]. Cet algorithme itératif sur l'ordre de prédiction calcule le prédicteur optimal d'ordre $j+1$ à partir du prédicteur optimal d'ordre j. En utilisant la notation $a_i^{(j)}$ pour le $i^{ème}$ coefficient du modèle d'ordre j, l'algorithme de Levinson est défini par les équations suivantes :

Initialisation : $$a_0^{(j)} = 1, \qquad j = 1, \cdots, p \tag{1.9}$$

$$a_{j+1}^{(j+1)} = -\frac{\sum_{i=0}^{j} a_i^{(j)} R_s(j+1-i)}{\sum_{i=0}^{j} a_i^{(j)} R_s(i)} \tag{1.10}$$

$$a_i^{(j+1)} = a_i^{(j)} + a_{j+1}^{(j+1)} a_{j+1-i}^{(j)}, \quad 1 \le i \le j \tag{1.11}$$

En notant $k_j = a_j^{(j)}$, le polynôme $A_j(z)$ associé à l'analyse d'ordre j, satisfait la relation de récurrence suivante :

$$A_j(z) = A_{j-1}(z) + k_j z^{-j} A_{j-1}(z^{-1}), \ \ j = 1, \cdots, p \tag{1.12}$$

avec

$$A_0(z) = 1 \tag{1.13}$$

Les coefficients $k_j, j = 1, \cdots, p$ sont appelés les coefficients de corrélation partielle PARCOR[4]. Physiquement, ils représentent les coefficients de réflexion aux variations de section du modèle acoustique du conduit vocal. L'algorithme de Levinson détermine directement les coefficients PARCOR.

---

[4] PARCOR : **Par**tial **Corr**elation.

Ces coefficients ont les propriétés suivantes :

- Leurs valeurs absolues sont toujours inférieures à 1 (condition nécessaire et suffisante pour la stabilité du filtre). Cette propriété rend les coefficients PARCOR plus adaptés à la quantification.
- Les coefficients d'ordre le plus bas sont les plus importants : le spectre du signal synthétisé est beaucoup plus sensible à une faible variation de k_1 qu'à celle de même valeur de k_p.
- Ils sont indépendants de l'ordre d'analyse.

1.2.1.2 Rapports d'aires logarithmiques

Les coefficients de rapport d'aires logarithmiques LAR[5] sont définis en fonction des coefficients PARCOR par :

$$LAR_i = log\left(\frac{1-k_j}{1+k_j}\right), \qquad j = 1, \cdots, p \tag{1.14}$$

Des études sur la distribution statistique des coefficients PARCOR ont montré qu'il est plus intéressant de quantifier les LAR que les PARCOR eux-mêmes [4].

1.2.1.3 Paires de raies spectrales

Pour $j = p + 1$ dans (1.12), nous avons :

$$A_{p+1}(z) = A_p(z) + k_{p+1} z^{-(p+1)} A_p(z^{-1}) \tag{1.15}$$

Nous allons considérer les deux cas $k_{p+1} = 1$ et $k_{p+1} = -1$ pour construire deux fonctions de transfert directe et inverse à l'ordre $p + 1$. Elles correspondent aux deux conditions extrêmes, c'est-à-dire à une fermeture complète ou une ouverture complète de la glotte dans le modèle acoustique. Sous ces conditions, nous obtenons les deux polynômes suivants :

$$P(z) = A_p(z) + z^{-(p+1)} A_p(z^{-1}) \tag{1.16}$$

---

[5] LAR : **L**ogarithm **A**rea **R**atio.

pour $k_{p+1} = +1$, et

$$Q(z) = A_p(z) - z^{-(p+1)}A_p(z^{-1}) \tag{1.17}$$

pour $k_{p+1} = -1$. Ces deux polynômes possèdent des propriétés très intéressantes qui peuvent être résumées comme suit [5]:

- Toutes les racines de $P(z)$ et $Q(z)$ sont sur le cercle unité,
- Les racines de $P(z)$ et $Q(z)$ s'alternent deux à deux sur le cercle unité,
- La propriété de la phase minimale du filtre $A(z)$ (ou la stabilité des filtres $H(z)$ et son inverse) est facilement préservée après la quantification des racines de $P(z)$ et $Q(z)$.

Comme les racines de $P(z)$ et $Q(z)$ sont sur le cercle unité, elles sont données par $e^{j\omega_i}$, et il est facile de montrer que dans le cas d'un ordre de prédiction p pair, $P(z)$ et $Q(z)$ sont donnés par :

$$P(z) = (1 + z^{-1}) \prod_{i=1,3,\cdots,p-1} (1 - 2cos(\omega_i)z^{-1} + z^{-2}) \tag{1.18}$$

et

$$Q(z) = (1 - z^{-1}) \prod_{i=2,4,\cdots,p} (1 - 2cos(\omega_i)z^{-1} + z^{-2}) \tag{1.19}$$

Les fréquences angulaires ω_i, qui correspondent aux racines des polynômes $P(z)$ et $Q(z)$, dans les équations (1.18) et (1.19) sont normalisées par la fréquence d'échantillonnage. Les paramètres ω_i, $i = 1, \cdots, p$, sont définis comme les fréquences de raies spectrales LSF (**L**ine **S**pectral **F**requencies) ou les paires de raies spectrales LSP (**L**ine **S**pectrum **P**airs). Il est important de noter que $\omega_0 = 0$ et $\omega_{p+1} = \pi$ sont des racines fixes correspondant à $z = 1$ et $z = -1$ respectivement. Ils sont donc exclus de l'ensemble des paramètres LSF nécessaires pour caractériser le filtre de synthèse. Les LSF peuvent être interprétés comme les fréquences de résonances du conduit vocal sous les deux conditions limites au niveau de la glotte (ouverture complète ou fermeture complète). La deuxième propriété des paramètres LSF peut être donnée sous la forme suivante :

$$\omega_0 < \omega_1 < \omega_2 < \cdots < \omega_{p-1} < \omega_p < \omega_{p+1} \tag{1.20}$$

avec $\omega_0 = 0$ et $\omega_{p+1} = \pi$. Cette relation est connue sous le nom propriété d'ordonnancement des LSF. Tant que cette propriété est conservée, la stabilité du filtre de synthèse $H(z)$ est assurée.

Les paramètres LSF se prêtent mieux à la quantification que les autres représentations du filtre LPC à cause des propriétés suivantes :

1. Les domaines de variation des différents LSF sont relativement limités (dans la Figure 1.4 on donne les histogrammes des 16 LSF), ce qui facilite leur quantification.
2. La stabilité du filtre de synthèse est assurée par préservation de la propriété d'ordonnancement. De plus, cette propriété permet la détection des erreurs de transmission des LSF sans introduire de redondance.

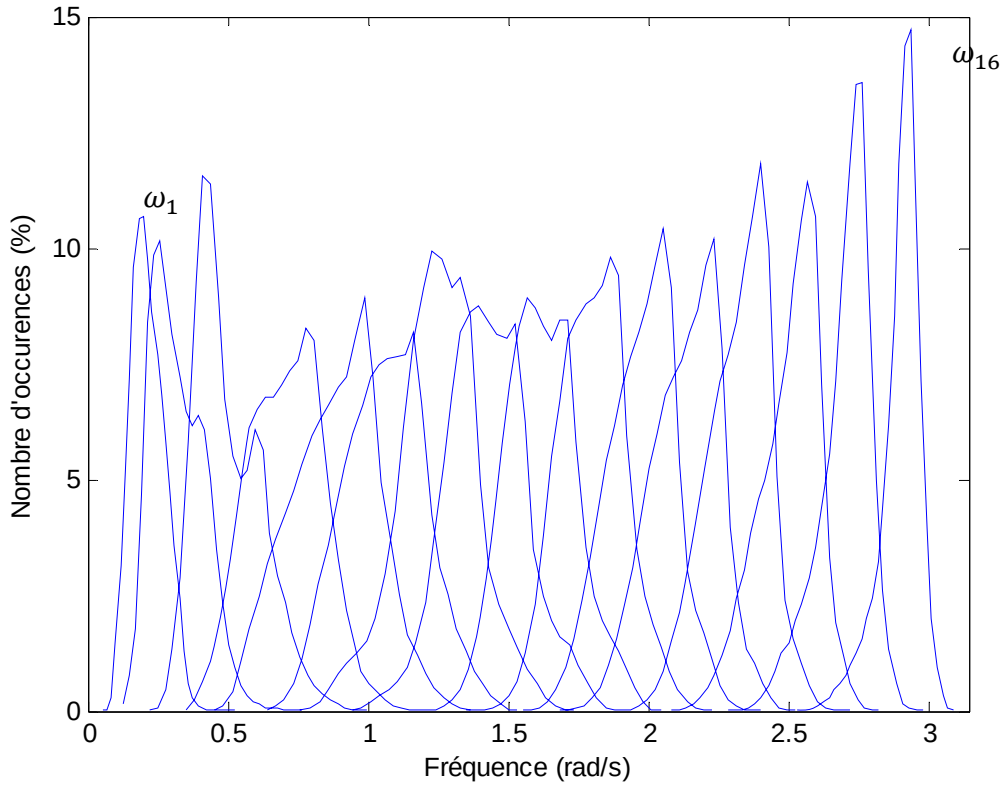


Figure 1.4 : Histogrammes des 16 LSF (la première composante et la dernière sont étiquetées).

3. Les coefficients LSF sont des paramètres fréquentiels. La proximité de deux raies crée un formant (Figure 1.5). A partir des coefficients LSF, il est donc possible d'identifier grossièrement les zones auditivement importantes dans le spectre du signal de façon très aisée.

4. Une erreur dans un seul coefficient LSF ne se propage pas plus loin, son effet spectral est limité dans la région étroite autour de la fréquence correspondant à ce coefficient (figure 1.6).

5. Les LSF entre deux fenêtres d'analyse adjacentes sont fortement corrélés.

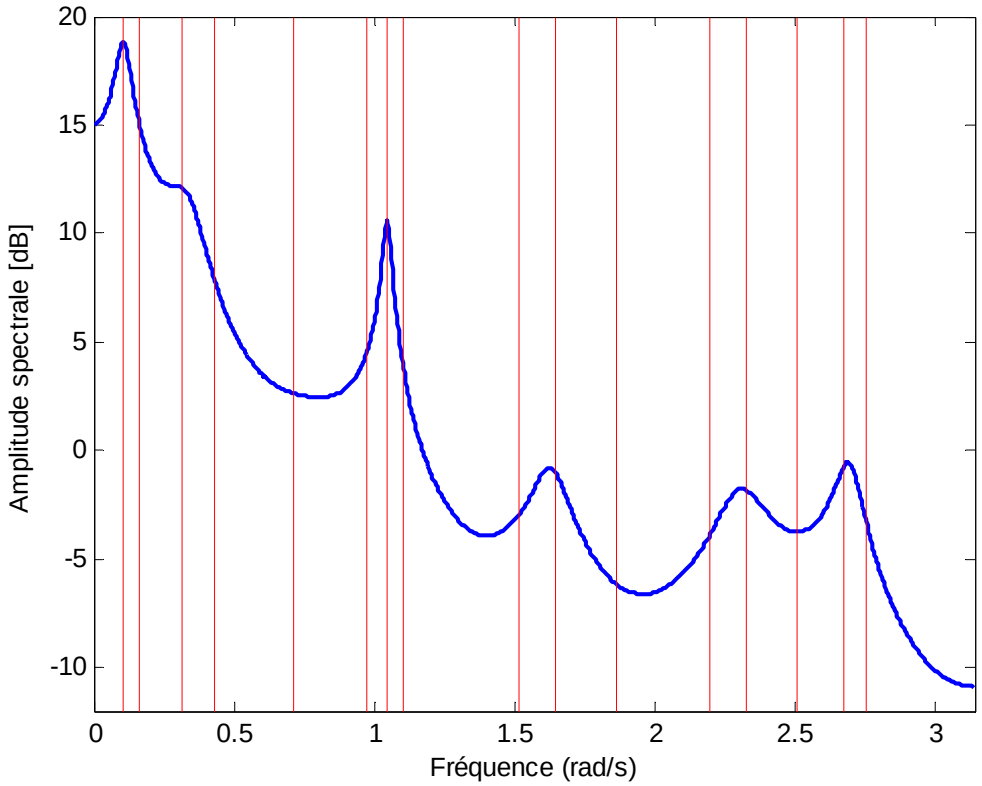


Figure 1.5 : Le spectre LPC et les LSF associés.

Les deux polynômes $P(z)$ et $Q(z)$, qui sont symétrique et antisymétrique respectivement, ont pour racines fixes $z = 1$ et/ou $z = -1$ qui peuvent être enlevées par division polynomiale,

$$G_1(z) = \frac{P(z)}{1+z^{-1}} \quad \text{et} \quad G_2(z) = \frac{Q(z)}{1-z^{-1}}, \quad p \text{ pair},$$

$$G_1(z) = P(z) \quad \text{et} \quad G_2(z) = \frac{Q(z)}{1-z^{-2}}, \quad p \text{ impair}, \qquad (1.21)$$

Les résultants $G_1(z)$ et $G_2(z)$ sont des polynômes symétriques d'ordre pair. Comme les racines n'apparaissent que par paires complexes conjuguées, il est seulement nécessaire de déterminer celles situées sur la moitié supérieure du cercle unité. Les racines en question sont $exp(j\omega_i)$ pour $i = 1, \cdots, p$. Les LSF sont donc les positions angulaires des racines $(0 < \omega_i < \pi)$.

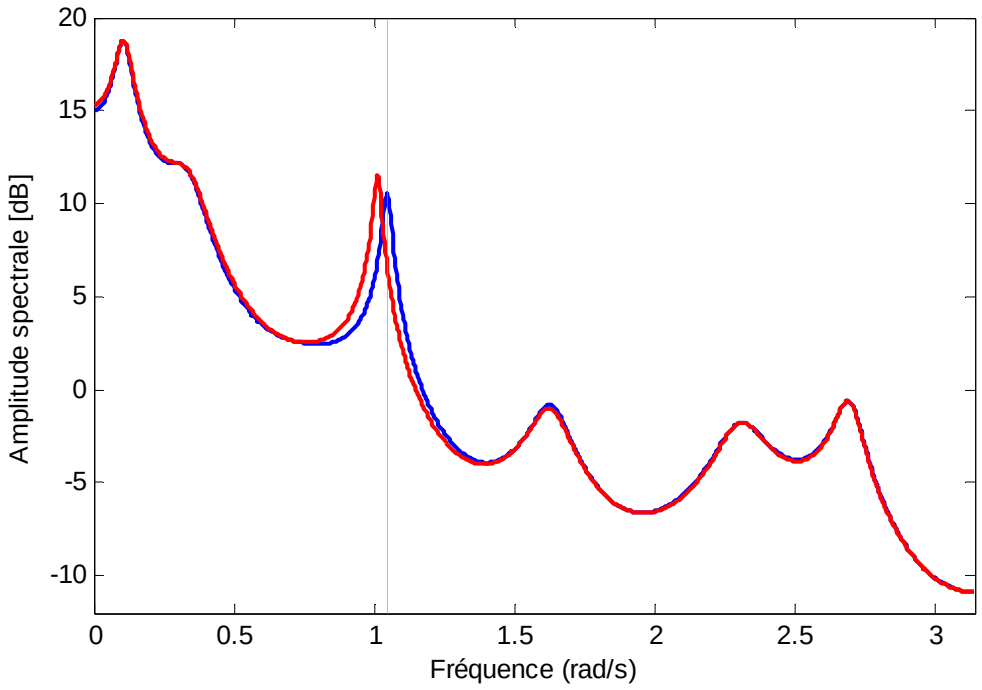


Figure 1.6 : Modification de la fonction de transfert du filtre de synthèse H(z) due à une augmentation de 0.01π du septième LSF (la position de ce LSF est indiquée par « le trait vert »). La courbe en bleu représente la fonction de transfert initiale tandis que la courbe en rouge représente la fonction de transfert correspondant au jeu de LSF modifié.

La figure 1.7 montre les zéros de $P(z)$ et $Q(z)$ pour un ordre p pair et impair. Ces tracés montrent les positions actuelles des racines pour un segment de la parole (fréquence d'échantillonnage de 16 kHz). Les polynômes $G_1(z)$ et $G_2(z)$ ont les mêmes zéros que

$P(z)$ et $Q(z)$, respectivement, excepté les zéros à $z = \pm 1$. On n'omettra pas de noter que pour n'importe quel ordre, la plus basse LSF correspond à une racine de $G_1(z)$.

Les cas d'un nombre impair et d'un nombre pair de LSF diffèrent de quelques détails. Posons l'ordre des polynômes $G_1(z)$ et $G_2(z)$ est égal à $2M_1$ et $2M_2$, respectivement,

$$M_1 = \frac{p}{2} \quad \text{et} \quad M_2 = \frac{p}{2} \quad , \qquad p \text{ pair},$$

$$M_1 = \frac{p+1}{2} \quad \text{et} \quad M_2 = \frac{p-1}{2} \quad , \qquad p \text{ impair}, \tag{1.22}$$

En faisant le changement de variable donné en (1.22), on peut exprimer sous forme explicite la symétrie des polynômes $G_1(z)$ et $G_2(z)$ comme suit :

$$\begin{aligned} G_1(z) &= 1 + g_1(1)z^{-1} + \cdots + g_1(M_1)z^{-M_1} + \cdots + g_1(1)z^{-(2M_1-1)} + z^{-2M_1} \\ G_2(z) &= 1 + g_2(1)z^{-1} + \cdots + g_2(M_2)z^{-M_2} + \cdots + g_2(1)z^{-(2M_2-1)} + z^{-2M_2} \end{aligned} \tag{1.23}$$

$G_1(z)$ possède M_1 paires de zéros conjugués et $G_2(z)$ possède M_2 de zéros conjugués ($M_1 + M_2 = p$). Le terme de phase linéaire peut être enlevé pour donner deux développements en séries de cosinus à phase nulle

$$\begin{aligned} G_1(\omega) &= e^{-j\omega M_1} G_1'(\omega) \\ G_2(\omega) &= e^{-j\omega M_2} G_2'(\omega) \end{aligned} \tag{1.24}$$

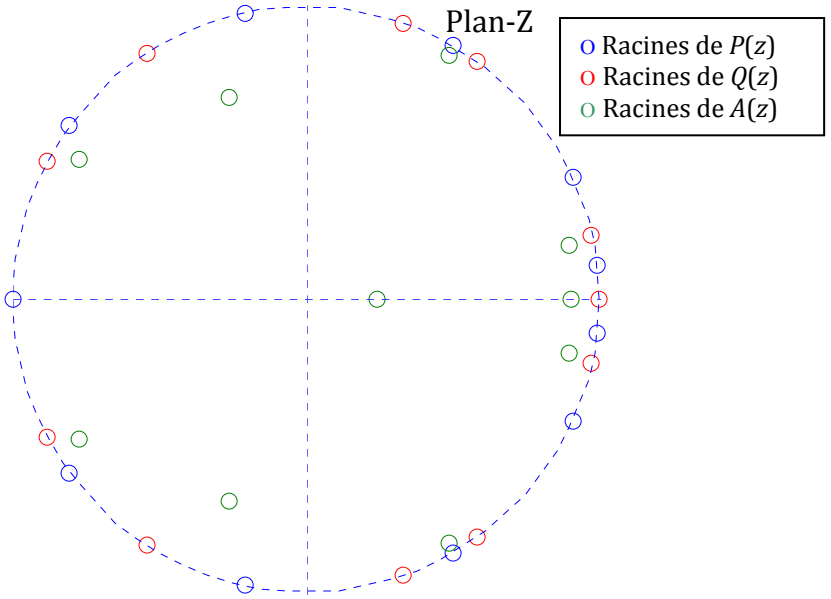
où

$$\begin{aligned} G_1'(\omega) &= 2cos(M_1\omega) + 2g_1(1)cos[(M_1 - 1)\omega] + \cdots + 2g_1(M_1 - 1)cos(\omega) + g_1(M_1) \\ G_2'(\omega) &= 2cos(M_2\omega) + 2g_2(1)cos[(M_2 - 1)\omega] + \cdots + 2g_2(M_2 - 1)cos(\omega) + g_2(M_2) \end{aligned} \tag{1.25}$$

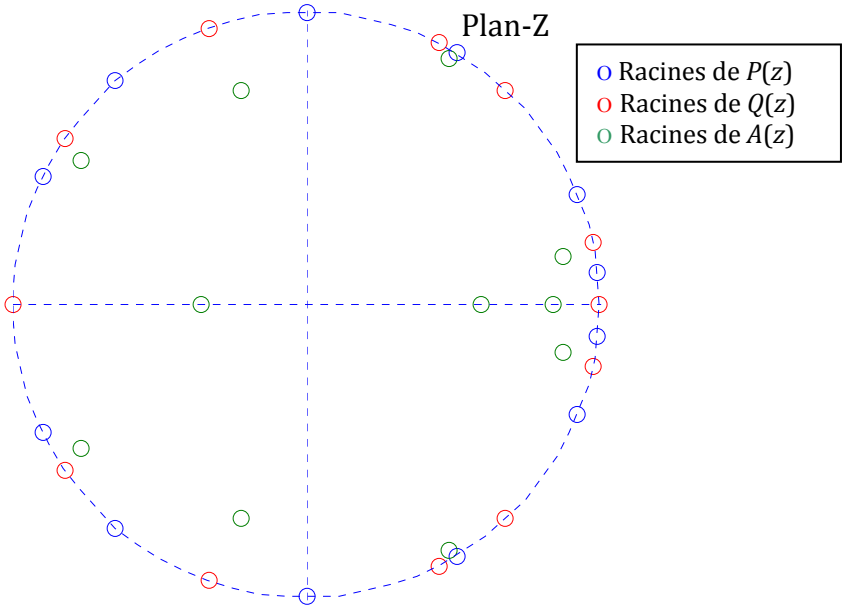
Plusieurs méthodes ont été proposées pour calculer les racines de $G_1'(\omega)$ et $G_2'(\omega)$. Parmi ces méthodes, on trouve la méthode développée par P. Kabal et R. P. Ramachandran [6]. Ils ont utilisé un polynôme de Chebyshev d'ordre m pour la détermination des LSF.

Si nous considérons la relation $cos(\omega) = x$, alors

$$cos(m\omega) = T_m(x) \tag{1.26}$$



(a)



(b)

Figure 1.7 : Emplacements des racines. (a) p pair ($p = 10$). (b) p impair ($p = 11$).

où $T_m(x)$ est un polynôme de Chebyshev d'ordre m. Les polynômes de Chebyshev satisfont la relation de récurrence :

$$T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x) \tag{1.27}$$

avec les conditions initiales, $T_0(x) = 1$ et $T_1(x) = x$.

L'équation (1.25), en utilisant le développement en polynômes de Chebyshev, devient :

$$\begin{aligned} G_1'(x) &= 2T_{M_1}(x) + 2g_1(1)T_{M_1-1}(x) + \cdots + 2g_1(M_1-1)T_1(x) + g_1(M_1) \\ G_2'(x) &= 2T_{M_2}(x) + 2g_2(1)T_{M_2-1}(x) + \cdots + 2g_2(M_2-1)T_2(x) + g_2(M_2) \end{aligned} \tag{1.28}$$

Lorsque les racines $\{x_i\}$ de $G_1'(\omega)$ et $G_2'(\omega)$ sont déterminées, les LSF correspondants sont données par $\omega_i = arccos(x_i)$. Le changement du variable $x = cos(\omega)$ transforme la moitié supérieure du cercle unité, dans le plan Z, dans l'intervalle $[-1, +1]$, sur la droite des x. Ainsi tout les $\{x_i\}$ sont situées entre -1 et $+1$, avec la racine correspondante à la plus faible fréquence située plus près de $+1$ (voir figure 1.8).

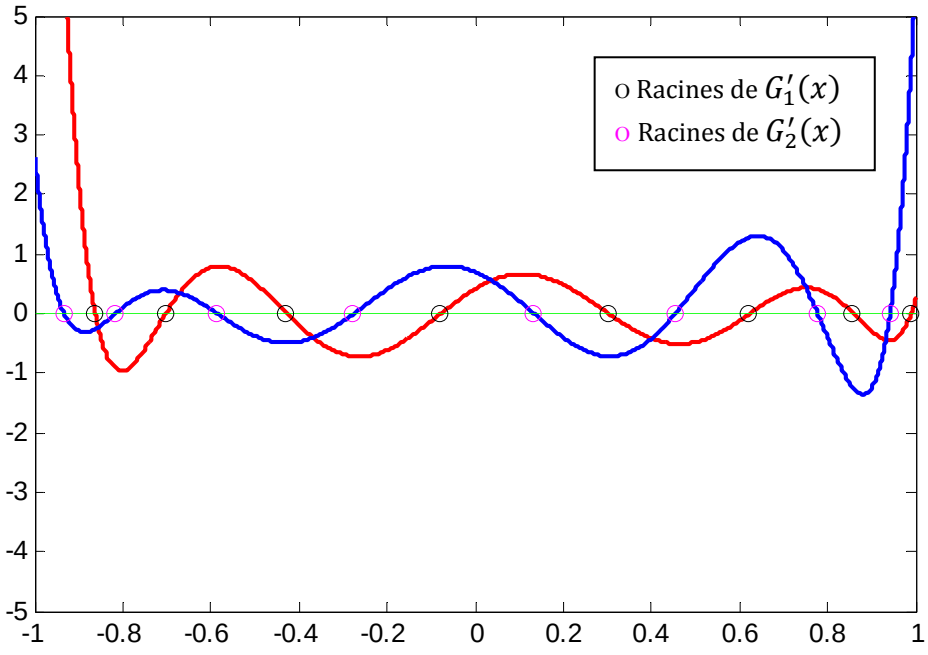


Figure 1.8 : Allure de $G_1'(x)$ (la ligne en rouge) et $G_2'(x)$ (la ligne en bleu) pour $p = 16$.

La série en polynôme de Chebyshev se prête à une évaluation efficace et précise qui évite une expansion en puissances de x. La série, à évaluer, peut être représentée comme suit :

$$Y(x) = \sum_{k=0}^{N-1} c_k T_k(x) \tag{1.29}$$

Considérons la récurrence suivante :

$$b_k(x) = 2x b_{k+1}(x) - b_{k+2}(x) + c_k \tag{1.30}$$

avec les conditions initiales $b_N(x) = b_{N+1}(x) = 0$. Cette récurrence est utilisée pour calculer $b_0(x)$ et $b_2(x)$. Alors, $Y(x)$ peut être exprimée en termes de $b_0(x)$ et $b_2(x)$ par la relation :

$$Y(x) = \sum_{k=0}^{N-1} [b_k(x) - 2x b_{k+1}(x) + b_{k+2}(x)]\, T_k(x)$$

$$= \frac{b_0(x) - b_2(x) + c_0}{2} \tag{1.31}$$

L'avantage de cette formulation est que les erreurs d'évaluation de $b_0(x)$ et $b_2(x)$ tendent à s'annuler. En se rappelant la propriété d'entrelacement et sachant que la plus proche racine de $x = 1$ correspond au polynôme $G_1'(\omega)$, la procédure d'extraction des x_i peut être abordée :

$x_0 = x_1 = 1$

$i = 1$

Pour $j = 1$ **à** p

1. Evaluer $G_i'(x_0)$ en utilisant (1.30) et (1.31).
2. $x_1 = x_1 - \delta$
3. Evaluer $G_i'(x_1)$ en utilisant (1.30) et (1.31).
4. ***Si*** $G_i'(x_0) \cdot G_i'(x_1) > 0$ ***aller à*** 2.
5. Rechercher la solution précise x_s par divisions successives (dichotomie) de $[x_0, x_1]$.

6. ***Si*** $i = 1$ ***alors*** $i = 2$ ***sinon*** $i = 1$
7. $x_0 = x_s$
8. ***aller*** **à** 2

Fin

Enfin, il faut noter que le δ doit être assez faible pour pouvoir détecter toutes les racines, mais pas trop faible afin d'accélérer l'algorithme de résolution.

1.2.2 Prédiction à long terme

Bien que la prédiction à court terme supprime la corrélation entre les échantillons adjacents, il reste une corrélation à plus long terme se traduisant par des pics d'amplitudes quasi périodiques au niveau du résidu de la prédiction à court terme. Ces pics sont d'autant plus marqués que le signal est voisé. Cette périodicité, correspondant à la fréquence fondamentale du signal, peut être estimée par une prédiction à long terme. La fréquence fondamentale comme nous l'avons vu précédemment va de 50 Hz pour les voix les plus graves à 600 Hz pour les voix aiguës, ce qui correspond à une période de 1.6 à 20 ms, période bien au-delà de la durée de la prédiction à court terme de l'ordre de la milliseconde.

La mise en œuvre de la prédiction à long terme est un moyen efficace pour représenter cette périodicité du signal de parole. Alors que la modélisation AR représente l'enveloppe du spectre réel, la prédiction à long terme a pour objet de modéliser les fines structures spectrales dues aux vibrations des cordes vocales. On l'effectue en général sur le signal résiduel $e(n)$ obtenu après le filtrage à court terme. On considère ce signal $e(n)$ comme le signal glottique qui possède la propriété de périodicité pour les sons voisés. Le signal de parole étant échantillonné à la fréquence F_e, la fréquence fondamentale F_0 est représentée par un certain nombre P appelé pitch.

$$P = \frac{F_e}{F_0} \tag{1.32}$$

Il est à noter que la valeur de P n'est pas forcément un nombre entier.

Le prédicteur à long terme extrait les redondances (corrélation à long terme) d'un signal de parole périodique en prédisant l'échantillon présent à partir d'une combinaison linéaire des échantillons précédents retardés de P échantillons, où P est un nombre entier. La forme générale d'un prédicteur long terme d'ordre l, impair, est la suivante :

$$P_l(z) = 1 - \sum_{k=-(l-1)/2}^{(l-1)/2} b_k z^{-(P+k)}, \qquad l = 1,3,\cdots \tag{1.33}$$

où les b_k sont les coefficients de prédiction à long terme.

Cette expression permet une modélisation plus fine de la structure périodique, notamment dans le cas où la fréquence d'échantillonnage F_e n'est pas un multiple de la fréquence fondamentale F_0. Pour ne pas augmenter le débit binaire, on se contente la plupart du temps d'un prédicteur du premier ordre.

1.3 Différents types de codeurs

Les méthodes de codage de la parole sont nombreuses et sont classées généralement en trois grandes catégories : les codeurs de forme d'onde (ou temporels), les codeurs paramétriques, appelés vocodeurs, et les codeurs hybrides. Le choix d'une méthode va dépendre surtout de l'application visée et des contraintes sur le débit. Les codeurs de forme d'onde sont surtout performants pour des débits élevés (au–delà des 16 kbit/s). Les vocodeurs quant à eux sont plutôt destinés aux très bas débits (au-dessous de 2.4 kbit/s) et aux bas débits (2.4 à 8 kbit/s). Les codeurs hybrides ont des débits nominaux intermédiaires (8 à 16 kbit/s), bien qu'ils puissent aussi être utilisés pour de bas débits. Une autre façon de distinguer les codeurs de parole est la largeur de bande codée. Historiquement, les codeurs de parole sont à bande étroite, c.-à-d. codent le signal sur la bande de 300 à 3400 Hz. Le signal d'entrée est alors échantillonné à 8 kHz. Ces dernières années, la tendance est d'augmenter la largeur de bande vers la bande élargie de 50 à 7000 Hz en utilisant une fréquence d'échantillonnage de 16 kHz. La qualité de la synthèse et de la communication s'en trouve nettement améliorée [7] [8]. L'agrandissement de la bande passante pour la parole n'apporte pas grand-chose du point de vu de l'information. Contrairement à la musique, où la bande élargie peut comporter des phénomènes supplémentaires (notes aigués), l'information de parole (l'intelligibilité) est intégralement

contenue dans la bande téléphonique (sauf peut être pour quelques langues très exotiques). On peut néanmoins espérer deux améliorations :

- Pour les phonèmes voisés, l'addition de la bande de fréquences 50 Hz - 300 Hz donne une meilleure représentation des premières harmoniques. On remarque surtout cela pour un locuteur masculin pour lequel la fréquence de pitch est assez faible. D'une manière plus générale les basses fréquences procurent une sensation de confort et un sentiment de parler « face à face ».
- L'apport des hautes fréquences, supérieures à 3400 Hz, n'a de l'importance que pour les phonèmes plus complexes tels que les fricatives non voisées (ex: « S », « CH », « F »), les fricatives voisées (ex: « Z ») ou encore les plosives (ex: « T », « D »).

1.3.1 Codage de forme d'onde

Les codeurs de forme d'onde s'attellent à représenter le plus exactement possible la forme de l'onde du signal sans nécessairement exploiter les propriétés de la parole et de l'audition. De ce fait, ils sont en général plus indépendants du signal d'entrée que les autres types de codeurs. En contrepartie, ils génèrent des débits plus élevés.

Le codage de forme d'onde le plus simple est sans conteste la modulation d'impulsion codée (PCM[6]). Il s'agit d'une simple quantification scalaire uniforme ou non uniforme (loi A ou μ) des amplitudes temporelles. A la fréquence d'échantillonnage de 8 kHz, le codage sur 8 bits de chaque échantillon par une quantification non uniforme permet de reproduire un signal synthétique indiscernable de l'original pour un débit de 64 kbit/s. On considère que c'est une technique d'échantillonnage sans compression. Par contre, la modulation d'impulsion codée différentielle adaptative (ADPCM[7]) utilise un codage prédictif exploitant la corrélation interéchantillon à court terme. La norme ITU-T G.721 [9] utilise un codage ADPCM à 32 kbit/s. Elle a été étendue pour les débits 16, 24 et 40 kbit/s avec les normes G.723 et G.726 [10]. La Figure 1.9 est le schéma de principe d'un codeur ADPCM. On ne quantifie plus l'amplitude de chaque échantillon mais la différence entre

---

[6] PCM : Pulse-Code Modulation (recommandé en 1972 par l'ITU-T sous la norme G.711).

[7] ADPCM : Adaptive Differential Pulse Code Modulation.

cette valeur et une valeur prédite déterminée par filtrage adaptatif. La valeur du pas de quantification varie en fonction de l'amplitude de la différence afin de limiter la distorsion en cas d'erreur importante de la prédiction. En effet, la méthode de prédiction la plus simpliste considère que l'échantillon traité est identique au précédent donc si le signal subit une grande variation, la prédiction sera fortement erronée.

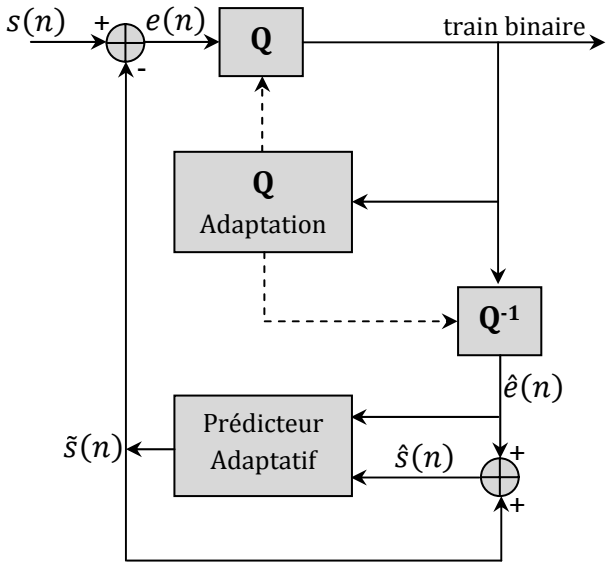


Figure 1.9 : Système ADPCM.

Le codage ADPCM offre une bonne qualité de codage pour des débits assez élevés mais ses performances chutent rapidement en dessous de 16 kbit/s. Un autre inconvénient majeur de ce type de codage est la faible robustesse face aux erreurs de canal.

Il existe aussi une version large bande (Le signal est échantillonné à 16 kHz) de l'ADPCM avec la norme G.722 [11]. Elle combine une décomposition en deux sous-bandes de fréquences de largeurs égales (0-4000 Hz et 4000-8000Hz) et une version modifiée du G.721 pour chaque sous-bande. Les débits de sortie sont soit 48, 56 ou 64 kbit/s.

1.3.2 Codage paramétrique

L'utilisation des techniques temporelles conduit à une excellente qualité de parole mais elle entraine également un débit élevé. La fréquence d'échantillonnage étant fixée, la réduction du débit entraine la diminution du nombre de niveaux de quantification responsable d'un bruit inacceptable pour de faibles débits. Pour obtenir une représentation

numérique du signal de parole plus économique en débit, tout en respectant le compromis existant entre la qualité de la parole décodée et le débit, il devient nécessaire de prendre en compte les propriétés spécifiques de ce signal.

Les codeurs paramétriques ou vocodeurs[8] n'essaient pas de synthétiser un signal ressemblant temporellement à l'original, mais tentent plutôt de produire un signal à partir d'un modèle approché de la phonation. Ils visent à reproduire correctement l'enveloppe spectrale. Le signal de parole est considéré comme la réponse d'un filtre linéaire, modélisant le conduit vocal, à un signal d'excitation, modélisant la source vocale. Dans le cas des sons voisés, le signal d'excitation est composé d'impulsions périodiques, de période égale à la période de voisement. Dans le cas d'un son non voisé, le signal d'excitation du filtre est un bruit blanc. Il faut donc détecter si le signal est voisé ou non et, ensuite, mesurer la valeur du fondamental dans le cas des sons voisés.

On extrait du signal de parole deux types de paramètres : ceux du filtre de synthèse et ceux du signal d'excitation. Le débit obtenu avec une telle méthode va de 2 à 4 kbits, la qualité d'écoute restant très modeste (qualité synthétique, « robot »). Parmi ces codeurs, nous pouvons citer le LPC-10 [12], remplacé par le MELP[9] [13], destinés à des applications militaires.

1.3.3 Codage hybride

Le codage hybride est un compromis intéressant entre les deux catégories précédentes en termes de qualité et de débit. Il associe une modélisation poussée du signal de parole avec une optimisation par analyse par synthèse. L'analyse par synthèse consiste en une boucle fermée permettant d'optimiser le codage de l'excitation en minimisant une mesure de la différence entre le signal original et sa synthèse. La Figure 1.10 donne le schéma bloc d'un codage par analyse par synthèse.

---

[8] Acronyme de « Voice Coder ».

[9] MELP : **M**ixed **E**xcitation **L**inear **P**rediction.

Le codage CELP[10] a été introduit par Atal et Schroeder [14] et fait parti des codeurs hybrides. Il est très efficace pour les débits intermédiaires de 5 kbit/s à 16 kbit/s, comme en témoignent les nombreuses normes qui l'utilisent [15, 16, 2, 17]. La Figure 1.11 représente le principe du codage CELP.

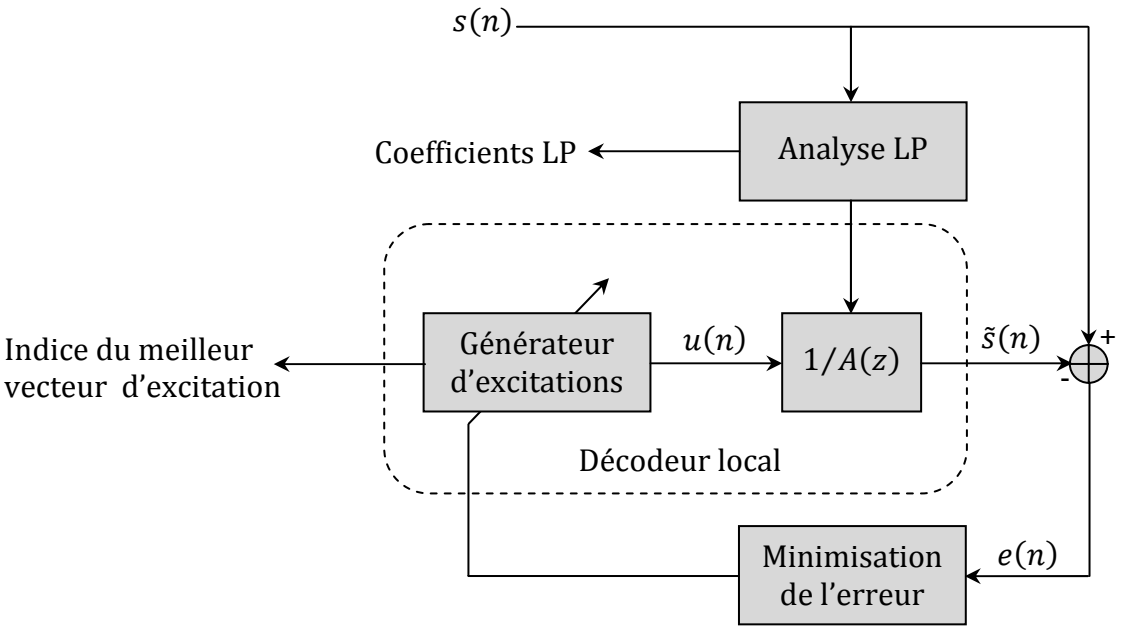


Figure 1.10 : Principe de codage par analyse par synthèse.

Le système consiste en un filtre de synthèse $1/A(z)$ à court terme représentant l'enveloppe spectrale du signal. Une prédiction linéaire à long terme représentée par le filtre $P(z)$ permet de modéliser la périodicité du signal, c.-à-d. le pitch. Pour diminuer la complexité du codeur, la contribution du pitch est souvent traduite par l'excitation passée et périodisée par la période fondamentale obtenue par la prédiction. On parle alors de dictionnaire adaptatif. Sa sortie est ajoutée à un vecteur du dictionnaire innovateur choisi à l'issue de l'analyse par synthèse. La minimisation de l'erreur se fait selon un critère perceptuel, qui est obtenu en filtrant la synthèse par un filtre dit perceptuel ou pondéré $W(z)$. Ce filtre pondère l'erreur dans le domaine fréquentiel en tenant compte des caractéristiques de l'audition humaine. Il atténue les zones du spectre à forte amplitude

[10] CELP : **C**ode **E**xcited **L**inear **P**rediction.

(formants) et amplifie les zones de faible amplitude. Une forme classique du filtre est la suivante :

$$W(z) = \frac{A(z/\gamma_1)}{A(z/\gamma_2)} \qquad \text{avec} \qquad 0 < \gamma_2 < \gamma_1 \leq 1 \tag{1.34}$$

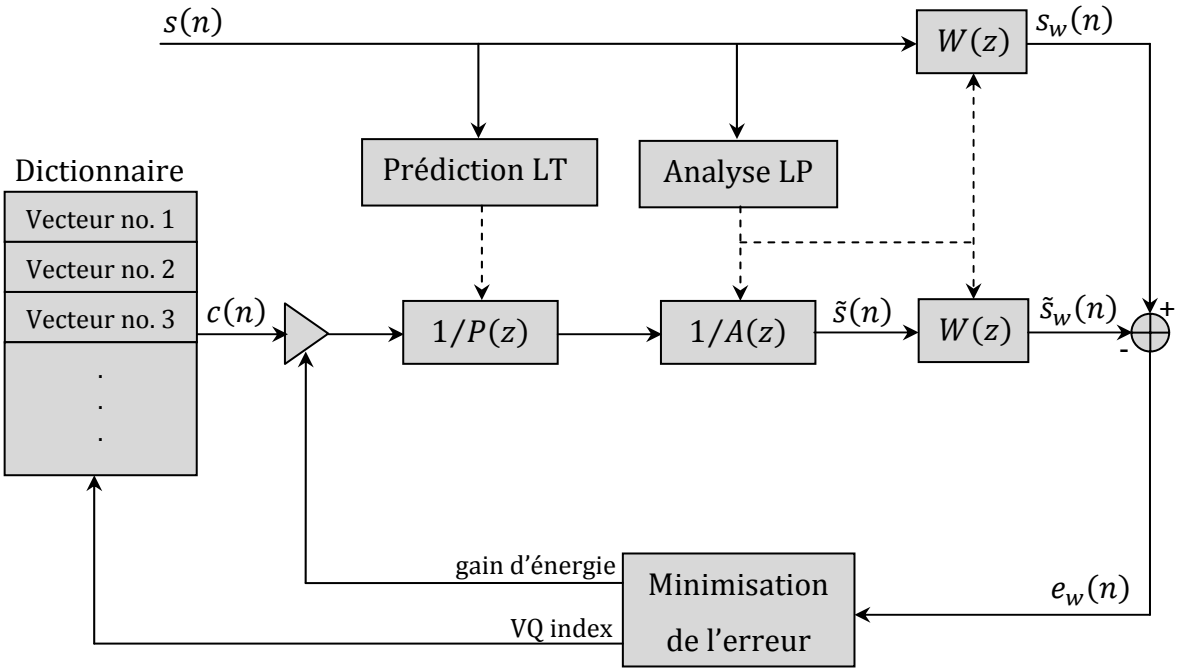


Figure 1.11 : Principe du codage CELP.

Le facteur γ_2 a pour effet de déplacer les pôles vers le centre du cercle par rapport aux pôles du filtre de synthèse $1/A(z)$. Le filtre est donc plus stable, il y a moins de résonance au niveau des formants. La Figure 1.12 représente le spectre de $W(z)$ pour $\gamma_1 = 0.98$ et $\gamma_2 = 0.6$. On voit bien que $W(z)$ est formé d'antirésonances au niveau des formants du signal. Le filtre $W(z)$ favorise alors l'erreur de quantification à avoir une énergie plus importante au niveau des formants (énergies élevées).

Dans la version originale du CELP, le dictionnaire innovateur était un dictionnaire stochastique nécessitant d'une part de la mémoire de stockage, mais surtout un temps de calcul non négligeable pour la recherche du meilleur représentant par le processus d'analyse par synthèse. Les efforts de recherche se sont alors concentrés sur ce problème. On peut citer par exemple la technologie Vector-Sum Excited Linear Prediction (VSELP)

[16] utilisant des dictionnaires très structurés afin de réduire la complexité algorithmique. Elle a été adoptée en Amérique du Nord pour la téléphonie mobile de deuxième génération. Mais la technologie la plus répandue est la technologie ACELP (Algebraic-CELP) [18]. Elle utilise un dictionnaire algébrique pour modéliser l'innovation permettant ainsi de réduire la complexité algorithmique et les ressources de stockage. Elle est utilisée par de nombreux codeurs modernes de parole pour des communications sans fil et filaires, tels le G.729 [2] et l'AMR-WB [17].

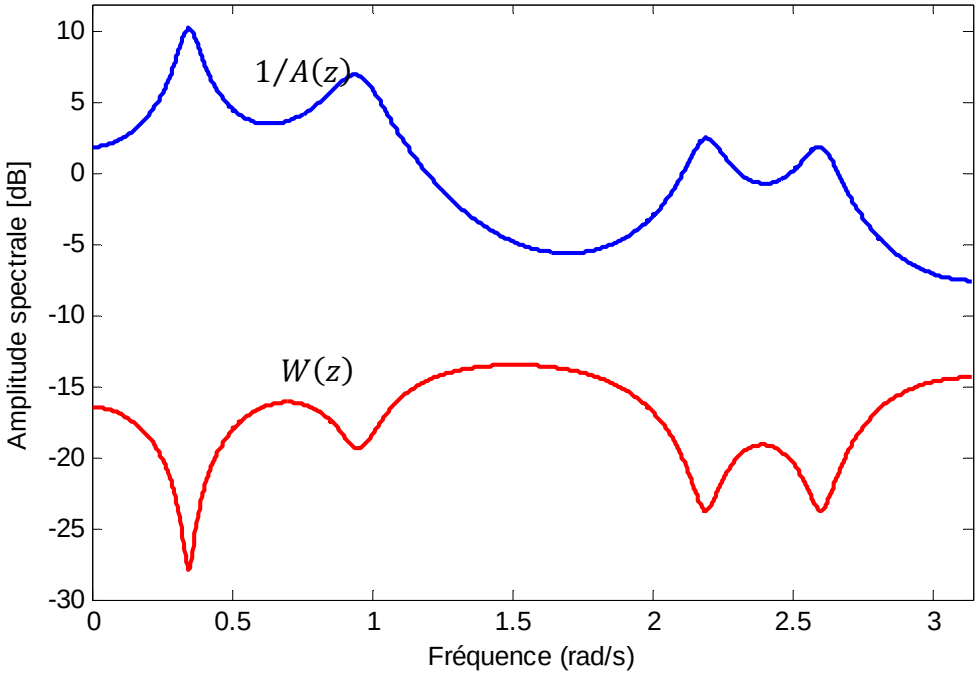


Figure 1.12 : Spectres du filtre de synthèse et du filtre perceptuel.

Le codeur AMR-WB, standardisé sous la recommandation de l'ITU-T G.722.2 [19], utilise le codage ACELP décrit précédemment. Il fonctionne pour 9 débits différents allant de 6.6 kbit/s à 23.85 kbit/s[11]. Les modes allant de 12.65 kbit/s jusqu'au débit maximal offre une excellente qualité de codage de la parole large bande tandis que les deux modes inférieurs (6.6 et 8.85 kbit/s) sont prévus pour être utilisés temporairement, en général dans

[11] Les modes du codeur large bande AMR-WB (ITU-T G.722.2) sont : 23.85, 23.05, 19.85, 18.25, 15.85, 14.25, 12.65, 8.85, et 6.6 kbit/s.

des cas de saturation du réseau. Le codec opère à la fréquence d'échantillonnage de 16 kHz et sur des blocs de 20 ms. Deux bandes de fréquence, 50-6400 Hz et 6400-7000 Hz, sont codées séparément afin de diminuer la complexité et de bien répartir l'allocation en bits sur les gammes de fréquence subjectivement importantes [17]. Le bloc d'entrée est d'abord sous-échantillonné à 12.8 kHz. Après cette décimation et avant le processus de codage, deux fonctions de prétraitements sont appliquées au signal : un filtrage passe-haut et préaccentuation. Le filtre passe-haut sert de protection contre des composantes parasites à basse fréquence. On utilise un filtre de fréquence de coupure à fréquence de coupure de 50 Hz, dont la formule est la suivante :

$$H_h(z) = \frac{0.989502 - 1.979004z^{-1} + 0.989502z^{-2}}{1 - 1.978882z^{-1} + 0.9799126z^{-2}} \tag{1.35}$$

Lors de la préaccentuation, un filtre passe-haut du premier ordre est utilisé afin d'accentuer les très hautes fréquences. Ce filtre est donné par :

$$H_p(z) = 1 - 0.68z^{-1} \tag{1.36}$$

Chaque trame de 20 ms est découpée en 4 sous-trames de 5 ms chacune. Au niveau du codeur, le calcul des coefficients du filtre LPC d'ordre 16 est effectué pour chaque trame de 20 ms au moyen de la méthode d'autocorrélation dans une fenêtre asymétrique de 30 ms. Une expansion fréquentielle de 60 Hz est utilisée pour éviter des pics spectraux pointus dans l'analyse LPC. Par ailleurs, le premier coefficient d'autocorrélation $r[0]$ est multiplié par le facteur de correction du bruit blanc égal à 1.0001. Les paramètres nécessaires aux dictionnaires adaptatif et fixe sont calculés pour chaque sous-trame. En plus de ces différents paramètres, une valeur de VAD (Voice Activity Detection) est transmise. Comme la bande de fréquence basse contient presque toute l'information, la grande majorité des bits est allouée à son traitement.

1.4 Mesures de la distorsion objective

L'appareil auditif humain est l'ultime évaluateur de la qualité d'un codeur de la parole. Néanmoins, les mesures objectives peuvent donner un estimateur immédiat et fiable de la qualité perceptuelle d'un algorithme de codage.

1.4.1 Mesures dans le domaine temporel

La mesure objective de qualité la plus couramment utilisée, pour les codeurs qui essaient de préserver la forme du signal, est le rapport signal sur bruit (SNR)[12] et le rapport signal sur bruit segmental (SNR_{Seg}).

1.4.1.1 Rapport signal sur bruit

Le rapport signal sur bruit mesure la longueur relative de la puissance du signal sur la puissance de bruit. La mesure de SNR, en décibel (dB), est définie comme suit :

$$SNR = 10log_{10}\left(\frac{\sum_{n=-\infty}^{\infty} s^2(n)}{\sum_{n=-\infty}^{\infty}[s(n)-\tilde{s}(n)]^2}\right) \text{ (dB)} \qquad (1.35)$$

où $\tilde{s}(n)$ est la version codée de l'échantillon du signal de parole original. Cependant la mesure de SNR n'est pas un bon estimateur de la qualité de la parole. Cette mesure pondère de la même manière toutes les erreurs dans le signal, négligeant le fait que l'énergie du signal de parole est variable dans le temps. Le signal de parole étant par nature non-stationnaire, certains segments du signal peuvent avoir une énergie plus ou moins grande. En supposant que l'énergie de l'erreur soit à peu près constante, le SNR pourra être soit très important soit très faible. Pour avoir une idée de la qualité de la parole synthétique, on utilise plutôt le SNR segmental (SNR_{Seg}).

1.4.1.2 Rapport signal sur bruit segmental

Le SNR_{Seg} est la moyenne géométrique des mesures SNR calculées sur différentes trames. La mesure SNR_{Seg} en dB, sur M segment de parole, est définie comme suit :

$$SNR_{Seg} = \frac{1}{M}\sum_{m=0}^{M-1} 10log_{10}\left[\frac{\sum_{n=1}^{N} s^2(n+Nm)}{\sum_{n=1}^{N}[s(n+Nm)-\tilde{s}(n+Nm)]^2}\right] \text{ (dB)} \qquad (1.36)$$

où chaque segment « m » est de longueur N. Pour un signal de parole avec une fréquence d'échantillonnage de 16 kHz les valeurs de N varient typiquement entre 320 et 480 échantillons (20 à 30ms).

---

[12] SNR : **S**ignal to **N**oise **R**atio.

Cette mesure présente l'avantage de tenir compte de l'évolution de SNR au cours du temps, et en particulier, de bien prendre en compte les segments de faible énergie. Il faut noter que dans le calcul du SNR_{Seg}, on élimine les zones de silence[13] du calcul.

1.4.2 Mesures dans le domaine spectral

En codage hybride ou paramétrique de la parole, on cherche à restituer un signal de synthèse qui soit perceptuellement aussi ressemblant que possible à l'original. L'oreille est très sensible aux distorsions du spectre, ce qui explique l'introduction de mesure de distances spectrales. Ces mesures sont fonction des densités spectrales des signaux original et codé.

La mesure de distorsion $d(x, \tilde{x})$ entre deux vecteurs x et $\tilde{x}$, caractérisant le signal de parole, doit satisfaire les deux conditions suivantes [20]:

$$\begin{aligned} d(x,x) &= 0 \\ d(x,\tilde{x}) &\geq 0 \end{aligned} \tag{1.37}$$

La métrique, doit satisfaire deux autres conditions [21] :

$$\begin{aligned} d(x,\tilde{x}) &= d(\tilde{x},x) \\ d(x,\tilde{x}) &\leq d(x,y) + d(y,\tilde{x}) \end{aligned} \tag{1.38}$$

En général, toute mesure de performance est la moyenne d'une mesure de distorsion ou distance. La mesure est généralement faite en utilisant des trames de parole de 20 à 30 ms de longueur.

$$D = \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} d(x_i, \tilde{x}_i) \tag{1.39}$$

Une mesure de distorsion dans le cas de la parole devrait avoir une signification dans le domaine fréquentiel. Les différences entre l'enveloppe spectrale originale et codée, qui peuvent conduire perceptuellement à des sons différents, sont dues essentiellement à deux effets :

---

[13] Les trames de parole avec une puissance inférieure à 40 dB sont considérées comme zones de silence.

- Les résonances ou formants de l'enveloppe spectrale originale et codée se produisent à des fréquences considérablement différentes.
- Les largeurs de bande de formant de l'enveloppe spectrale originale et codée diffèrent considérablement.

Plusieurs mesures de distorsion spectrale ont été proposées dans la littérature. On citera par exemple la mesure de distorsion spectrale et la mesure de distance Euclidienne pondérée.

1.4.2.1 Mesure de distorsion Log spectrale

La mesure de distance spectrale (SD[14]) dans la norme L_p est définie par :

$$SD_n^p = \frac{1}{\pi}\int_0^{\pi}\left|10log_{10}[S_n(\omega)] - 10log_{10}\left[\tilde{S}_n(\omega)\right]\right|^p d\omega \tag{1.40}$$

où le spectre d'amplitude fréquentiel $S_n(\omega)$ d'une trame de parole « n » est donné par :

$$\begin{aligned} S_n(\omega) &= \frac{1}{|A_n(\omega)|^2} \\ &= \frac{1}{\left|1+\sum_{k=1}^{p} a_k e^{-jk\omega}\right|^2} \end{aligned} \tag{1.41}$$

où $\{a_k\}$ sont les coefficients de prédiction linéaire.

Quand $p = 2$ (norme L_2) la mesure de distorsion spectrale pour une trame de parole « n » se réduit à une distance quadratique moyenne. Elle est définie par:

$$SD_n^2 = \frac{100}{\pi}\int_0^{\pi}\left|log_{10}\left[\frac{S_n(\omega)}{\tilde{S}_n(\omega)}\right]\right|^2 d\omega \tag{1.42}$$

Quand l'évaluation est faite sur une base de données du signal de parole, les performances sont généralement présentées comme la moyenne de la distorsion spectrale et le pourcentage des trames « outliers » ayant une SD supérieure à un seuil prédéfini [22]. Il y a deux façons différentes pour calculer la distorsion spectrale moyenne. La première est donnée par :

---

[14] SD : **S**pectral **D**istortion.

$$\overline{SD}_{RMS} = \sqrt{\frac{1}{N_f}\sum_{n=1}^{N_f} SD_n^2} \tag{1.43}$$

où SD_n est la distorsion spectrale, donnée en décibels, pour la trame « n » et N_f est le nombre total de trames dans la base de données. L'autre façon est exprimée comme suit :

$$\overline{SD}_{MRS} = \frac{1}{N_f}\sum_{n=1}^{N_f} \sqrt{SD_n^2} \tag{1.44}$$

Bien que la $\overline{SD}_{RMS}$[15] ait été utilisée dans certains travaux, par exemple dans [23, 24], la $\overline{SD}_{MRS}$[16] est la plus connue et la plus utilisée.

La quantification avec qualité transparente consiste à quantifier l'information LPC sans dégradation perceptuelle. Pour ce faire, il est habituellement admis en codage à bande étroite qu'il faut satisfaire aux trois conditions suivantes [22]:

- La distorsion spectrale moyenne $\overline{SD}_{MRS}$ est d'environ 1dB.
- La proportion de trames ayant une distorsion spectrale entre 2 et 4 dB est moins que 2%.
- Aucune trame n'a une distorsion supérieure à 4 dB.

Une trame possédant une distorsion spectrale supérieure à 2 dB est souvent appelée trame « outlier ».

Les tests d'écoute de Guibé *et al.* [25] ont montré que les conditions pour avoir une qualité transparente en codage à bande étroite peuvent être également appliquées dans le cas de la bande large.

1.4.2.2 Mesure de distance euclidienne pondérée

La mesure de distorsion spectrale correspond bien à l'évaluation subjective de la qualité de la quantification des coefficients LPC. Une distorsion spectrale de 1 dB suggère une excellente performance. En raison de la complexité de calcul, la distorsion spectrale est

---

[15] $\overline{SD}_{RMS}$: Root-Mean-Square SD.

[16] $\overline{SD}_{MRS}$: Mean-Root-Square SD.

inutilisable pour la recherche du plus proche voisin et pour la construction du dictionnaire. Pour remplacer la distance spectrale dans ces opérations, certains auteurs utilisent soit la simple distance euclidienne donnée par :

$$d(\omega, \widetilde{\omega}) = \sum_{i=1}^{P} (\omega_i - \widetilde{\omega}_i)^2 \tag{1.45}$$

soit une distance euclidienne pondérée exprimée comme suit :

$$d(\omega, \widetilde{\omega}) = \sum_{i=1}^{P} w_i (\omega_i - \widetilde{\omega}_i)^2 \tag{1.46}$$

Nous donnons quelques exemples de facteurs de pondération w_i :

- Le poids d'un paramètre LSF est égal à la valeur prise par la densité spectrale de puissance pour ce LSF, élevée à la puissance r :

  $$w_i = [S(\omega_i)]^r, \;\; i = 1, \cdots, p \tag{1.47}$$

  où r est une constante, par exemple dans [22] sa valeur est égale à 0.30. Cette fonction de pondération a pour rôle d'accentuer les pics spectraux dans les régions de formants.

- La pondération proposée dans [26] est basée sur la propriété des paramètres LSF suivante : lorsque deux paramètres LSF sont proches l'un de l'autre, la fonction de transfert du canal vocal présente un maximum (formant), donc c'est une zone auditivement importante. La sensibilité spectrale de ces paramètres LSF est plus considérable, et donc leur poids dans l'expression de distance sera plus important. Cette fonction de pondération est appelée la moyenne harmonique inverse (IHM[17]), elle a été utilisée dans [27]:

  $$w_i = \frac{1}{\omega_i - \omega_{i-1}} + \frac{1}{\omega_{i+1} - \omega_i}, \quad i = 1, \cdots, p \tag{1.48}$$

  avec $\omega_0 = 0$ et $\omega_{p+1} = \pi$.

[17] IHM : **I**nverse **H**armonic **M**ean.

1.5 Conclusion

Dans ce chapitre, nous avons rappelé brièvement quelques domaines liées au sujet de cette thèse. Nous avons présenté les paramètres LSF que nous allons quantifier dans la suite. Dans les codeurs actuels de parole, le codage des paramètres LSF occupe une part importante du débit, de la charge de calcul ainsi que de la complexité de stockage. Il faut donc bien choisir la technique de quantification des paramètres LSF afin d'avoir un codeur de parole conforme aux besoins désirés. Dans les chapitres suivants, nous allons proposer une technique de quantification pour les LSF qui permet de réduire la charge de calcul et la capacité de stockage tout en gardant la bonne qualité requise.

CHAPITRE 2
QUANTIFICATION VECTORIELLE ET TREILLIS

Au cours du traitement et de la transmission numérique du signal, toutes les données sont représentées sur un certain nombre d'éléments binaires, donc avec une précision finie. Les données de nature analogique doivent ainsi être quantifiées avant leur traitement numérique. La méthode la plus naturelle et la plus simple est de quantifier les données une par une, c'est la quantification scalaire (QS). La quantification multidimensionnelle ou vectorielle (QV) regroupe les données à quantifier en vecteurs et choisit leur représentant dans un ensemble fini de vecteurs prédéterminés de même dimension.

Outre le fait que la quantification soit une nécessité pour le traitement numérique des données, elle est aussi un moyen de compression de données. Sur cet aspect, beaucoup de travaux ont été consacrés à la quantification vectorielle car elle permet d'atteindre les mêmes performances que la quantification scalaire en exigeant moins de bits pour le codage des paramètres à quantifier.

La quantification vectorielle, bien que permettant de concevoir des quantificateurs dont les performances approchent la limite de Shannon, présente donc des difficultés d'implantation qui rendent difficile la construction de quantificateurs ayant cette performance. Le quantificateur codé par treillis, qui sera présenté dans ce chapitre, apporte une solution élégante à ces problèmes de construction et permet d'atteindre, de façon

simple, des performances équivalentes ou supérieures à celle des meilleurs quantificateurs vectoriels.

Ce chapitre présente tout d'abord une définition généralisée de la quantification pour, par la suite, décrire en détail la quantification scalaire, vectorielle et codée par treillis ainsi que la généralisation vectorielle de cette dernière.

2.1 Définitions

La quantification est une opération non-bijective qui fait correspondre à un point $x \in \mathbb{R}^L$ un point $y \in \mathbb{R}^L$, élément d'un sous ensemble fini appelé le dictionnaire (ou l'alphabet de reproduction). Les éléments de ce dictionnaire portent le nom de représentants (ou de symboles de reproduction ou de mots de code). La sélection des représentants varie en fonction du type de quantification qui est appliqué. Pour tous les types de quantificateurs, le point y est choisi de telle sorte que, parmi tous les points du dictionnaire, il soit celui qui minimise un certain critère d'erreur de quantification.

L'erreur (ou bruit) de quantification est définie comme la différence entre le point x et sa valeur quantifiée $\hat{x} = Q(x)$.

$$e = x - \hat{x} \tag{2.1}$$

Cette erreur est nommée erreur granulaire dans le cas où le point x est situé à l'intérieur des frontières du quantificateur Q. Dans le cas où le point x est situé à l'extérieur des limites de Q, l'erreur e est nommée erreur de saturation ou erreur de dépassement.

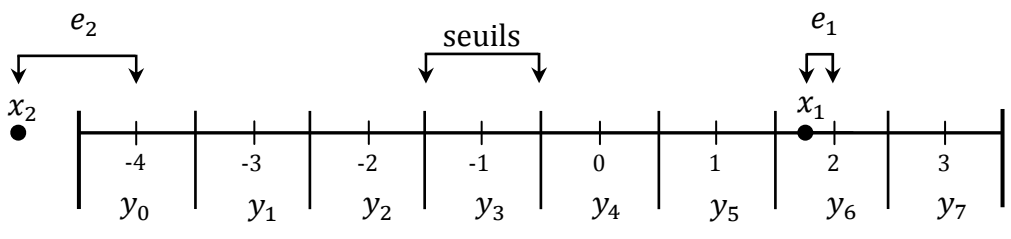


Figure 2.1 : Erreurs de quantification.

La figure 2.1 représente l'erreur granulaire e_1 engendrée par la quantification du point x_1 et l'erreur de saturation e_2 engendrée par la quantification du point x_2.

Le débit du quantificateur R exprime le nombre de bits alloués par échantillon. Un plus grand débit permet d'allouer davantage de points dans une région pour un pas de quantification fixe, ou de déplacer les bornes du quantificateur afin de réduire les erreurs de saturation pour une même résolution granulaire. Dans le cas d'un quantificateur à L dimensions possédant M représentants, le débit R est exprimé de la façon suivante :

$$R = \frac{1}{L}\log_2(M) \text{ bits/échantillon} \tag{2.2}$$

2.2 Quantification scalaire

La quantification scalaire est la technique de compression de données la plus ancienne, la plus étudiée, la plus utilisée et la plus simple à réaliser.

Dans l'opération de la quantification scalaire uniforme, le dictionnaire est formé de $M-1$ seuils (ou M régions). A chacune de ces régions est associé un indice i, lequel sera éventuellement transmis pour le décodage. Dans la figure 2.2, parmi les 8 différents codes possibles, le point x aura comme indice le mot 110.

Bien adapté à la quantification de sources uniformes, ce type de quantificateur est toutefois déficient pour la quantification de signaux tels que la parole, l'image,…, etc. Cela nous conduit à chercher un quantificateur qui prend en compte la distribution des représentants suivant les caractéristiques statistiques de la source et cela n'est rien d'autre que le quantificateur scalaire optimal.

Les quantificateurs scalaires optimaux sont des quantificateurs non uniformes qui ont été étudiés, pour les sources sans mémoire, par S. P. Lloyd [28] et J. Max [29]. Clairement, pour minimiser la distorsion due aux bruits de quantification, un quantificateur doit remplir les conditions suivantes :

1. Il doit représenter chaque symbole de la source par le symbole de reproduction le plus proche. Il en résulte implicitement une partition de l'ensemble des valeurs de la source dans des sous-ensembles, appelés également "régions de Voronoï". Chaque

sous-ensemble est associé à un symbole de reproduction et il contient tous les symboles de la source représentés par ce niveau de quantification.

2. Chaque symbole de reproduction doit minimiser la distorsion moyenne pour la représentation des symboles de source de sa région de Voronoï. Si la mesure de distorsion est la distance euclidienne, cela veut dire que les symboles de reproduction doivent être les centroïdes des régions de Voronoï.

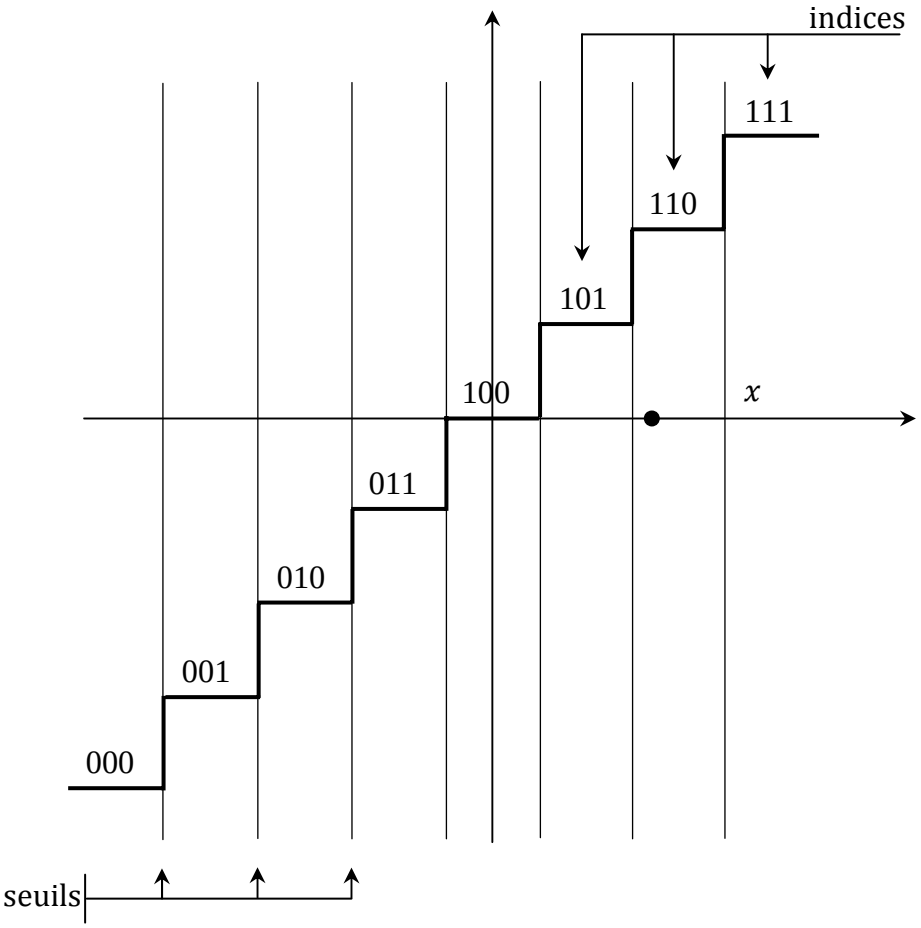


Figure 2.2 : Quantificateur scalaire uniforme.

Pour la quantification scalaire, les régions de Voronoï sont des segments de la droite réelle, délimités par leurs extrémités, appelées "seuils de quantification" ou bien "niveaux de décision". L'application itérative des règles **1** et **2** permet le calcul des

symboles de reproduction (niveaux de quantification) et des seuils de quantification optimaux.

Les performances trouvées pour un quantificateur scalaire optimal sont loin de rejoindre le maximum annoncé par la théorie d'information. De plus, le QS optimal ne permet pas d'obtenir un débit inférieur à 1 bit/échantillon. Il devient nécessaire de regrouper plusieurs échantillons dans un vecteur et chercher à quantifier l'ensemble. On parle alors de quantification vectorielle.

2.3 Quantification vectorielle

La quantification scalaire présente l'avantage d'être simple mais n'est toutefois pas adaptée à la quantification de toutes les sources. En transposant en deux dimensions un quantificateur scalaire (unidimensionnel), les points du quantificateur sont distribués uniformément dans le plan et forment des carrés (produit cartésien). La figure 2.3 illustre ceci.

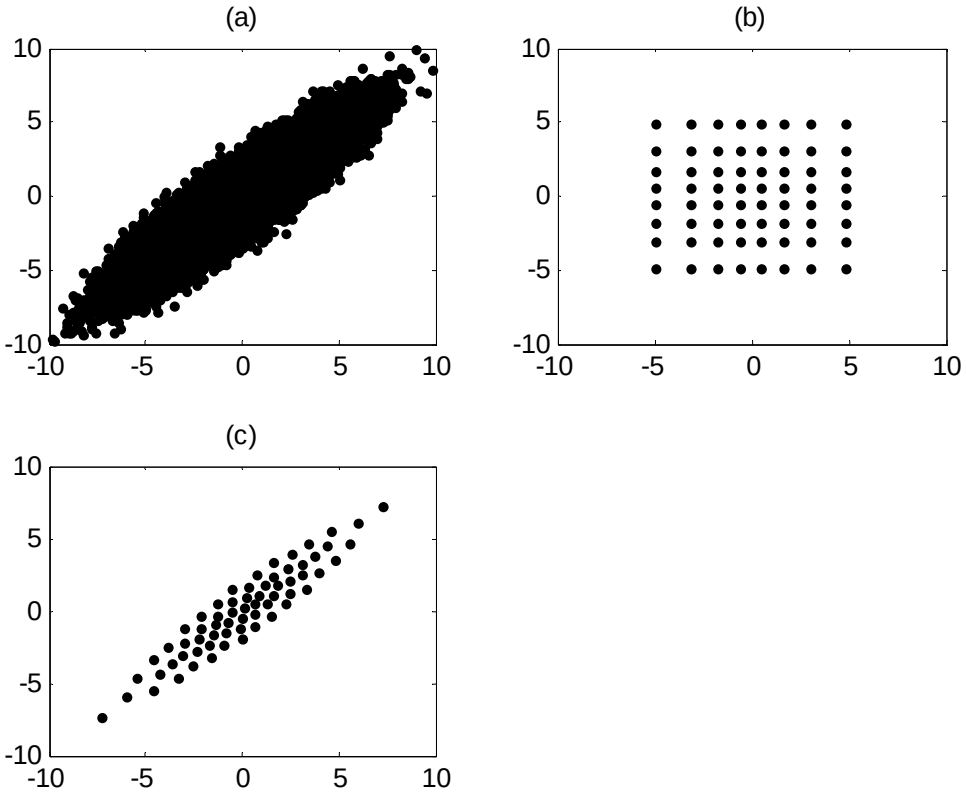


Figure 2.3 : Illustration d'une source avec mémoire et deux différents types de quantificateurs. (a) Source avec mémoire. (b) Quantificateur scalaire 2D. (c) Quantificateur vectoriel adapté à la source représentée en (a).

Souvent, les paramètres à quantifier présentent des distributions bien différentes de celle d'une source uniforme (figure 2.3-a). Ils sont souvent localisés dans une région bien précise. Par exemple, à la figure 2.3-b, il apparaît clairement que les points du quantificateur dans la quasi-totalité des cadrans 2 et 4 ont une probabilité nulle d'être utilisés si les échantillons de la source représentés par la figure 2.3-a devaient être quantifiés.

Dans un tel cas, il est évident que la répartition des points du dictionnaire d'un quantificateur scalaire bidimensionnel est inadéquate pour une multitude de sources à quantifier. Les bits utilisés pour représentés les points dans les cadrans 2 et 4 sont mal utilisés, ce qui rend le quantificateur sous-optimal.

L'introduction d'un nouveau quantificateur dont le dictionnaire est adapté à la source permet d'obtenir de meilleurs résultats. En quantification vectorielle statistique, les points du dictionnaire sont organisés de telle sorte qu'ils sont bien adaptés à la distribution de la source, comme il est montré à la figure 2.3-c. Ainsi, il y a davantage de représentants autour des régions ayant une plus grande probabilité d'occurrence et l'erreur quadratique moyenne entre x et $\hat{x}$ en est diminuée. A chaque représentant $\hat{x}$ correspond une région de l'espace appelée région de Voronoï. Celle-ci est définie comme étant la région à l'intérieur de laquelle la distorsion $d(x, \hat{x})$ est minimisée.

$$d(x, \hat{x}) = \|x - \hat{x}\|^2 = \sum_{i=0}^{N} (x_i - \hat{x}_i)^2 \tag{2.3}$$

Afin de déterminer à quelle région de Voronoï appartient un point x à quantifier, la mesure de distorsion doit être appliquée à l'ensemble des représentants du dictionnaire Q.

En pratique, un dictionnaire contenant M points (ou vecteurs types) en L dimensions est stocké en mémoire. Puisque les points sont généralement sauvegardés avec une précision de 32 bits, il vient que le dictionnaire requiert une espace mémoire de $L \times M \times 4$ octets. Etant donné la limitation physique des systèmes, la dimension et le débit des quantificateurs vectoriels doivent être eux aussi limités. De plus, le temps de calcul requis pour effectuer M calcul de distorsion sur un vecteur à L dimensions s'avère très coûteux et est également une limitation à la taille et au débit du quantificateur.

Jusqu'à maintenant il a été question du dictionnaire adapté à la source à quantifier sans qu'il soit question d'une méthode pour obtenir ce dictionnaire. Le paragraphe suivant introduira une méthode, nommé algorithme des k-moyennes, utilisée pour obtenir un dictionnaire adapté à une source [30].

2.3.1 Algorithme des k-moyennes

Cette méthode de construction des dictionnaires, est basée sur une séquence d'apprentissage regroupant un nombre fini de réalisations de la source ayant une probabilité de distribution qui représente le plus fidèlement possible les statistiques de la source.

Lorsque le dictionnaire initial est déterminé, l'algorithme s'exécute en une série d'itérations jusqu'à ce que l'amélioration de l'erreur moyenne totale ait atteint un certain seuil.

Nous appellerons m le numéro de l'itération et $C_i(m)$, le groupe de vecteurs à l'itération m, avec y_i le centroïde de C_i. Etant donné un corpus d'apprentissage contenant P vecteurs, que l'on notera $x(k)$ avec $1 \leq k \leq P$, l'algorithme se déroule de la façon suivante :

Etape 1 : Initialisation : $m = 0$

Un premier dictionnaire de représentants $y_i(0)$, avec $1 \leq i \leq M$, est choisi de façon adéquate dans la base d'apprentissage.

Etape 2 : les vecteurs $x(k)$ de la base d'apprentissage sont classifiés dans les cellules C_i à l'aide de la règle du plus proche voisin :

$$x \in C_i(m), \quad \text{si et seulement si} \quad d\big(x, y_i(m)\big) \leq d\left(x, y_j(m)\right), \quad \forall j \neq i \tag{2.4}$$

Etape 3 : $m \leftarrow m + 1$

Les représentants sont réadaptés dans chaque cellule en calculant les centroïdes des vecteurs de la base d'apprentissage compris dans les cellules :

$$y_i(m) = cent\big(C_i(m)\big), \qquad 1 \leq i \leq M \tag{2.5}$$

Etape 4 : Test d'arrêt

Si la diminution de la distorsion globale $D(m)$ par rapport à $D(m-1)$ est en dessous d'un certain seuil, l'algorithme s'arrête. Sinon, il est réitéré à partir de l'étape 2. La distorsion produite par le dictionnaire actuel est calculée par sommation des distances entre les vecteurs de la base d'apprentissage $x(k)$ et leur plus proche voisin $y(k) = Q(x(k))$ dans le dictionnaire :

$$D(m) = \frac{1}{K}\sum_{k=1}^{P} d\big(x(k), Q(x(k))\big) \tag{2.6}$$

Il faut noter que la fin de l'algorithme peut également être limitée par un nombre maximal d'itérations. De plus, la convergence vers un minimum global n'est pas assurée dans cet algorithme. Une optimalité globale peut être atteinte approximativement en initialisant différemment le premier dictionnaire et en répétant l'algorithme un certain nombre de fois. Le meilleur dictionnaire final sera celui qui aura la distorsion globale la plus faible.

2.3.2 Dictionnaire initial

Le choix du dictionnaire est initial est très important pour la convergence de l'algorithme des k-moyennes car il conditionne ses résultats finaux. Plusieurs méthodes ont été proposées pour le déterminer :

1) **Initialisation aléatoire :** consiste à sélectionner aléatoirement des vecteurs de la base d'apprentissage. Les M vecteurs sont pris bien éloignés temporellement dans la séquence d'apprentissage afin d'éviter toute corrélation éventuelle entre les représentants. Ces vecteurs peuvent bien sûr ne pas être du tout représentatifs de la suite d'apprentissage et on aboutit à des résultats très médiocres.

2) **Méthode par dichotomie vectorielle (« Splitting ») :** Cette méthode itérative a été proposée par Linde, Buzo et Gray [31]. Les étapes de construction du dictionnaire sont les suivantes :

   Etape 1 : Détermination du centre de gravité de la séquence d'apprentissage (centroïde de départ).

   Etape 2 : division des classes (méthode de « splitting »)

Le centroïde de chaque classe est perturbé par un coefficient ε pour donner deux vecteurs $y_i + \varepsilon$ et $y_i - \varepsilon$. Ce qui produit un nouveau dictionnaire de taille double.

Etape 3 : Convergence

L'algorithme des k-moyennes est effectué sur toute la base d'apprentissage avec le nouveau dictionnaire issu de l'étape 2, comme dictionnaire initial.

Etape 4 : Test d'arrêt

Tant que le nombre de représentants souhaité n'est pas atteint, l'algorithme est repris à l'étape 2.

Un inconvénient de cette méthode est que la perturbation des représentants conduit parfois à des classes vides. Celles-ci peuvent se multiplier au cours des itérations. La convergence s'effectue alors vers un dictionnaire contenant en réalité moins de M représentants. Afin d'éviter cela, après chaque perturbation, il faut détecter les classes vides et faire une nouvelle perturbation à ces endroits [32].

3) **Méthode de maximum de distance :** est une autre technique efficace. Cette méthode a été proposée par I. Katsavounidis *et al.* [33]. Ils ont montré que cette technique donne un meilleur minimum local que la méthode «*Splitting*» et réduit la complexité des calculs. Le principe de cette technique consiste à accorder une attention particulière aux vecteurs d'apprentissage qui sont les plus éloignés l'un de l'autre, car ils sont susceptible d'appartenir à des classes différentes (Voir figure 2.4).

Soit $x(k),\ k = 1,2,\cdots,P$ les vecteurs de la séquence d'apprentissage. La procédure peut être exprimée comme suit:

a. Calculer les normes de tous les vecteurs de l'ensemble d'apprentissage. Choisir le vecteur ayant la norme maximum comme le premier représentant.

b. Calculer la distance de tous les vecteurs d'apprentissage par rapport au premier représentant, et choisir le vecteur ayant la plus grande distance comme second représentant. On a alors un dictionnaire de taille 2.

c. Généralement, avec un dictionnaire de taille $m,\ m = 2,3,\cdots,M-1$ nous calculons la distance entre les vecteurs d'apprentissage restants $x(k)$ et tous les

représentants existants. La plus petite valeur trouvée est appelée distance entre $x(k)$ et le dictionnaire. Par conséquent, le vecteur d'apprentissage ayant la plus grande distance par rapport au dictionnaire est choisi pour être le $(m+1)^{ième}$ représentant. La procédure s'arrête quand on obtient un dictionnaire de taille M.

L'idée de base de cette procédure est d'utiliser le vecteur le plus «différent» des codes vecteurs existants comme un nouveau représentant. La procédure est applicable pour n'importe quelle taille de dictionnaire.

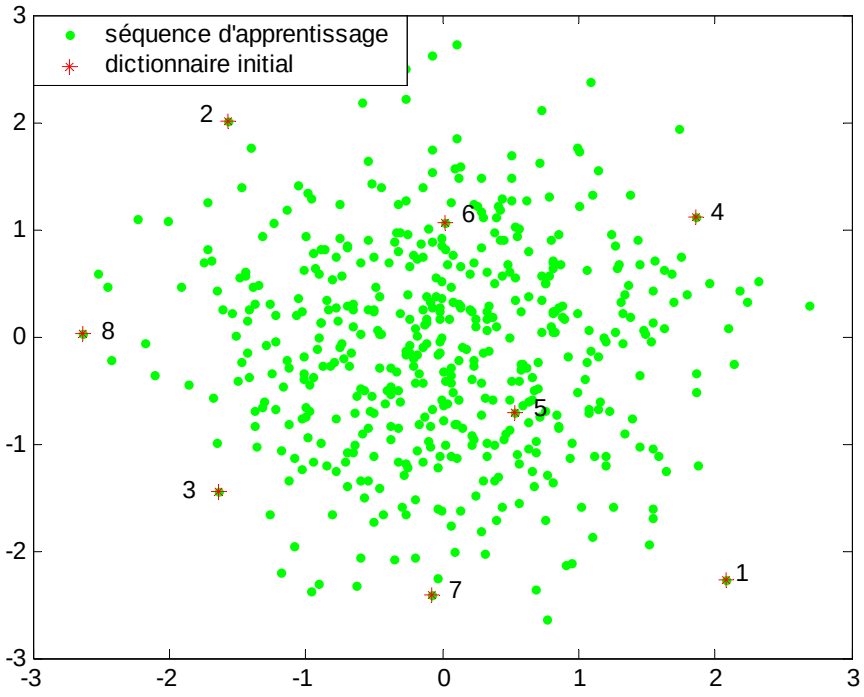


Figure 2.4 : Illustration du principe de fonctionnement de l'algorithme de maximum de distance pour $M = 8$ et $P = 500$.

2.4 Quantification codée par treillis (TCQ/TCVQ)

La théorie relative à la fonction débit-distorsion avec contrainte sur la taille de l'alphabet de sortie ("Alphabet constrained rate distortion theory"), développée par Pearlman *et al.* [34]-[36], indique que la performance débit-distorsion optimale peut être asymptotiquement approchée par augmentation de la taille du dictionnaire scalaire ou vectoriel.

En s'inspirant de cette théorie ainsi que des résultats obtenus par Ungerboeck en modulation codée par treillis [37], Marcellin et Fischer ont proposé la TCQ (Trellis Coded Quantization) qui utilise le treillis pour construire des quantificateurs scalaires d'une complexité modeste mais dont les performances s'approchent de celles des meilleurs quantificateurs vectoriels [38]. Le mélange entre quantification vectorielle classique et celle codée par treillis a donné naissance à une méthode efficace nommée "Quantification vectorielle codée par treillis" [39] ayant le même principe que la TCQ mais au lieu d'étiqueter les branches du treillis par des échantillons, on les étiquette par des vecteurs.

Dans ce qui suit, on reprend les idées proposées par Marcellin et Fischer sur le fonctionnement des quantificateurs scalaires et vectoriels en treillis, tout en détaillant la méthode de conception de ce type de quantificateur.

2.4.1 Courbes débit-distorsion

L'alphabet de reproduction utilisé dans la TCQ est justifié par la théorie de l'information concernant la fonction débit-distorsion. Cette théorie montre que, pour un débit donné, un quantificateur utilisant un nombre de symboles de reproduction double par rapport au quantificateur scalaire peut pratiquement réaliser la même performance qu'en absence de toute contrainte sur le nombre de symboles de reproduction.

R. E. Blahut [40] a établi un algorithme qui permet de calculer la fonction débit distorsion pour la quantification d'un signal sans mémoire (caractérisé par une masse de probabilité) utilisant un alphabet de reproduction donné. Pour appliquer cet algorithme, l'ensemble des valeurs que peuvent prendre les échantillons du signal (l'alphabet de source) ainsi que l'alphabet de reproduction doivent être de dimension finie. Cette hypothèse n'affecte en rien l'utilité de la théorie car on peut supposer que le signal à quantifier est obtenu par une "*préquantification*" très fine d'un signal initial prenant des

valeurs dans un ensemble infini. La distorsion introduite par cette préquantification peut être rendue aussi petite que l'on désire. Par ailleurs, le plus souvent en pratique, les sources à quantifier ont déjà subi une préquantification lors de conversion analogique/numérique.

Si on utilise un préquantificateur de Lloyd-Max, Marcellin *et al.* [38] ont démontré que :

$$D_C(R) = E[(X-U)^2] + D_U(R) \tag{2.7}$$

où

X: représente la source originale continue (infinie),

U: représente la source discrète (finie) résultante de la préquantification de X,

$D_C(R)$: représente la fonction débit distorsion à alphabet de reproduction donné de la source continue X,

$D_U(R)$: représente la fonction débit distorsion de la source discrète U.

L'utilisation de l'algorithme de Blahut, nous permet de calculer la fonction débit distorsion $D_U(R)$ pour la source discrète finie U et l'alphabet de reproduction donné.

Les figures 2.5, 2.6 et 2.7 donnent la fonction débit distorsion $D_C(R)$ pour les sources uniforme, gaussienne et laplacienne respectivement avec un préquantificateur de Lloyd-Max de 8 bits/échantillon. L'alphabet de reproduction utilisé est constitué des niveaux de quantification de Lloyd-Max à J niveaux.

On remarque qu'il y a une diminution de performance, due à l'utilisation de la dimension limitée de J, par rapport à la courbe débit distorsion sans limitation sur le nombre de symboles de reproduction. Cette dégradation s'accentue pour $R > \log_2(J) - 1$. Pour ces débits, on peut y remédier en utilisant un alphabet de reproduction de dimension $2J$ dans le cas d'une source uniforme ou gaussienne (ou $4J$ dans le cas d'une source laplacienne) pour obtenir une performance très proche de celle de la quantification sans contrainte sur J. Cette observation justifie l'utilisation d'un alphabet de dimension $2 \cdot 2^R$ (ou $4 \cdot 2^R$) pour la TCQ à R bits/échantillon.

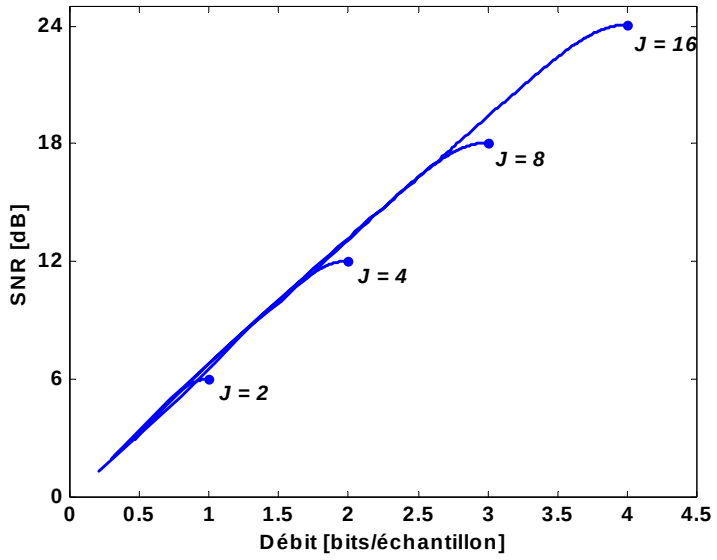


Figure 2.5: Courbe débit distorsion pour une source uniforme sans mémoire utilisant des alphabets de reproduction à $J = 2, 4, 8, 16$ symboles. Les points représentent les niveaux de Lloyd-Max.

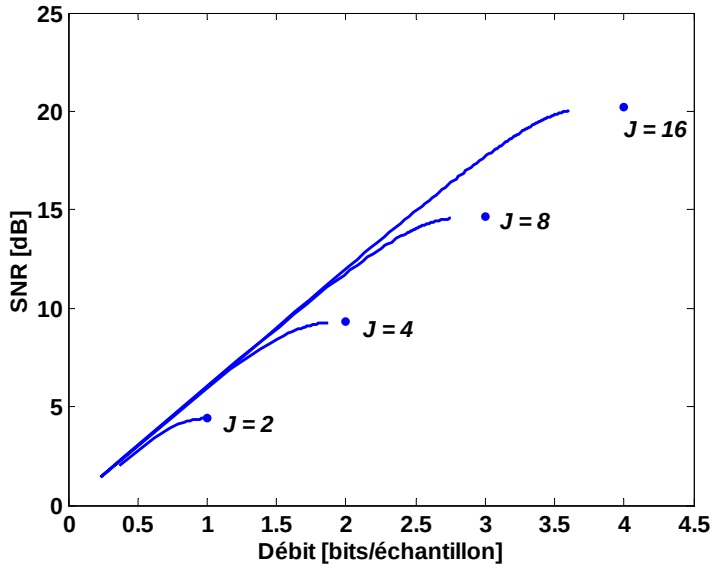


Figure 2.6: Courbe débit distorsion pour une source gaussienne sans mémoire utilisant des alphabets de reproduction à $J = 2, 4, 8, 16$ symboles. Les points représentent les niveaux de Lloyd-Max.

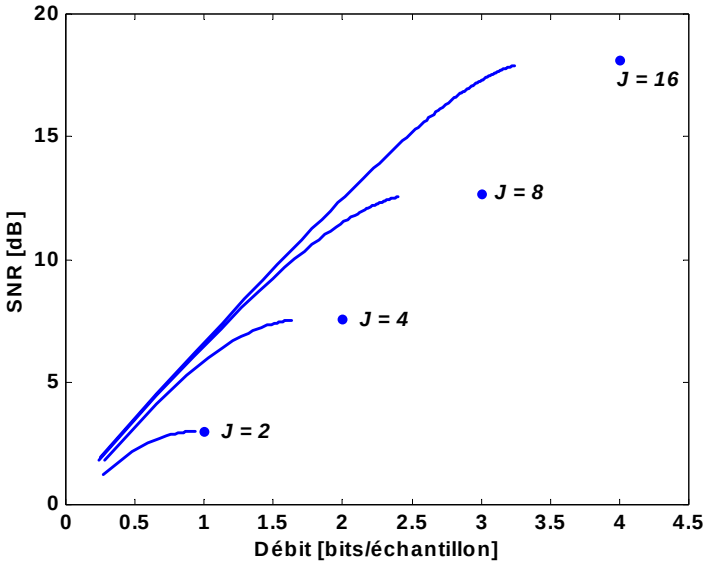


Figure 2.7: Courbe débit distorsion pour une source laplacienne sans mémoire utilisant des alphabets de reproduction à $J = 2, 4, 8, 16$ symboles. Les points représentent les niveaux de Lloyd-Max.

2.4.2 Principe d'un quantificateur codé par treillis

Comme on vu dans le paragraphe précédent, la quantification vectorielle à R bits/échantillon utilise un alphabet de 2^{RL} vecteurs en L dimensions. Avec la quantification vectorielle codée par treillis, cet alphabet est augmenté à $2^{(R+\tilde{R})L}$ vecteurs. $\tilde{R}$ s'appelle le facteur d'expansion de dictionnaire du treillis (en bits/échantillon). Il nous faudra donc imposer quelques règles sur l'utilisation de ce type de quantificateur afin de conserver le débit intact. Pour cela, l'alphabet (ou dictionnaire) du treillis est partitionné en $K = 2^{\tilde{R}L+m}$ sous-alphabets S_i, $i = 0, \cdots, K-1$. Ce qui veut dire que chaque sous-alphabet contient 2^{RL-m} vecteurs. Les sous-alphabets seront assignés aux branches de treillis (N états, 2^m branches). On n'omettra pas de noter que chacun des sous alphabets obtenus doit être un alphabet raisonnable, mais pas nécessairement optimal, s'il devait être utilisé dans un quantificateur classique.

A première vue, dans le cas d'un quantificateur vectoriel classique, il faudrait $\tilde{R}L + m$ bits et $RL - m$ bits pour identifier respectivement le sous-alphabet et le vecteur type, ce qui donnerait un débit total de $(R + \tilde{R})L$ bits/échantillon.

Cependant, en faisant appel à un treillis, on aura la possibilité de ramener le débit à RL bits/échantillon. La technique consiste à n'utiliser que 2^{RL} des $2^{(R+\tilde{R})L}$ vecteurs types disponibles dans l'alphabet du treillis mais en changeant les membres de ce groupe de 2^{RL} vecteurs types selon des règles connues tant du codeur que du décodeur.

2.4.3 Mécanisme d'un quantificateur codé par treillis

Bien que l'algorithme du quantificateur en treillis soit relativement simple, sa compréhension exige une familiarité avec plusieurs concepts. Cette sous-section va illustrer ces concepts à l'aide d'un exemple simple : celui d'un quantificateur scalaire codé par treillis défini par les caractéristiques suivantes : $R = 2$ bits/échantillon, $\tilde{R} = 1$ bit/échantillon, $m = 1$ et $N = 4$ états.

D'après ces caractéristiques, l'alphabet du treillis, noté A_0, contient 8 représentants au lieu de 4 représentants utilisés dans un quantificateur classique de même débit. Les valeurs types (ou représentants) de l'alphabet du quantificateur en treillis seront donc deux fois plus rapprochées de celles du quantificateur classique.

Cet alphabet est séparé en deux groupes : B_0 et B_1, chacun contenant le même nombre de valeurs types (*i.e.* 4 représentants dans chacun). La taille de ces sous-alphabets est en accord avec le débit, alors que celle de l'alphabet A_0 exigerait un bit supplémentaire pour décrire tous les indices des valeurs types.

Le choix des membres de B_0 et B_1 parmi les valeurs types de l'alphabet A_0 se fait à l'aide des règles du « set-partitioning », énoncées par Ungerboeck, et qui seront décrites plus loin dans ce chapitre. Pour l'instant, il est possible de résumer ces règles en disant que chacun des sous-alphabets obtenus doit être un alphabet raisonnable.

La figure 2.8 illustre le partage d'un alphabet A_0 de 8 valeurs types, donc destiné à un quantificateur en treillis d'un débit de 2 bits par échantillon.

Après le premier partage, chacun des sous-alphabets, B_0 et B_1, possède la moitié des valeurs types, soit le nombre permis par le débit envisagé. Le processus de partage est alors repris pour chacun des sous-alphabets, en suivant les mêmes règles ad hoc que le partage de A_0. Quatre sous-alphabets sont alors obtenus, C_0 et C_2, formés par le partage de l'alphabet B_0; et C_1 et C_3, formés par le partage de B_1 . Chacun des sous-alphabets C_0, C_1, C_2 et C_3 exige un bit de moins que le débit permis pour indexer ses valeurs types.

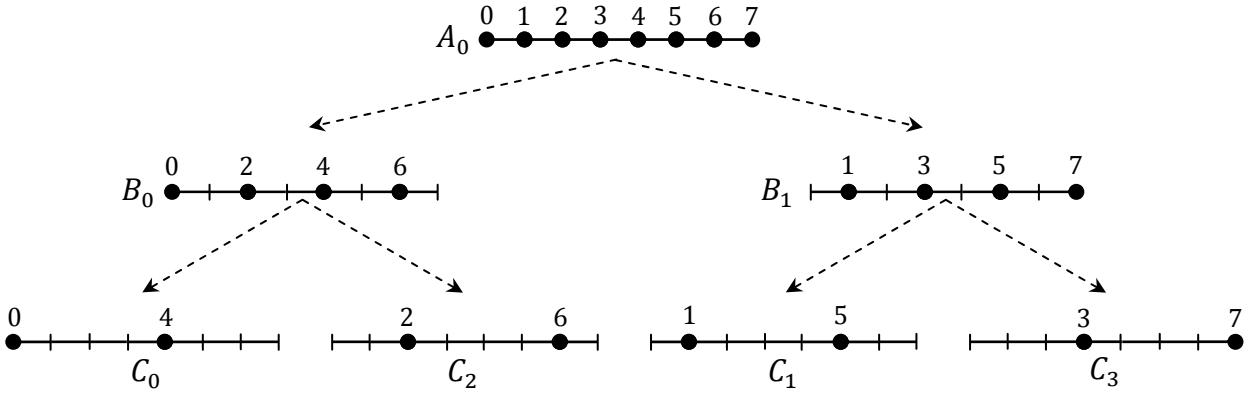


Figure 2.8 : Partage d'un alphabet scalaire en sous-alphabets d'après les règles d'Ungerboeck.

Jusque-là, le problème du bit en trop dans A_0 demeure entier. Un quantificateur pourrait utiliser le sous-alphabet B_0, ou le sous-alphabet B_1, mais pas les deux à la fois. Le quantificateur en treillis va utiliser une astuce pour faire comme si les deux sous-alphabets B_0 et B_1 étaient disponibles en même temps tout en respectant le débit imposé.

Cette astuce consiste à imposer une signification « cachée » à l'utilisation des sous-alphabets C_i, $i = 0, 1, 2, 3$. L'indice du sous-alphabet C_i utilisé pour quantifier un échantillon à un moment donné sert de « prophète » annonçant lequel des sous alphabets, B_0 ou B_1, sera utilisé pour quantifier le prochain échantillon. Cette annonce apparaîtra comme une contrainte à laquelle le quantificateur devra obéir dans la succession des choix des valeurs types de l'alphabet. C'est de cette contrainte que viendra la possibilité pour le quantificateur codé par treillis d'utiliser un alphabet de taille supérieure tout en respectant le débit puisqu'il n'utilisera qu'une partie de cet alphabet à chaque échantillon [41].

L’ensemble des contraintes auxquelles le treillis doit obéir dans le choix des séquences d’utilisation des sous-alphabets s’exprime bien sous la forme d’un diagramme de transitions tel que celui montré sur la figure 2.9. Les nombres encerclés sont les états possibles, les flèches représentent les transitions permises à partir d'un état donné. Il y a exactement deux transitions qui partent de chaque état, et exactement deux transitions qui arrivent à chaque état. Sur chaque transition, on retrouve le sous-alphabet utilisé parmi les quatre, et le numéro de transition r.

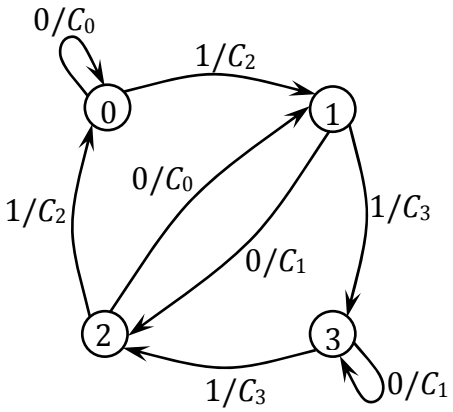


Figure 2.9 : Diagramme de transition du treillis à quatre états.

On peut aussi représenter le diagramme d'état par un treillis, ce que montre la figure 2.10. Sur le treillis, les états sont numérotés de haut en bas, les transitions bouclées sur un même état du diagramme d'état ont leur équivalent dans les transitions horizontales du treillis. Sur chaque transition du treillis, on retrouve aussi le sous-alphabet utilisé et le numéro de transition r. L’avantage de la représentation en treillis est qu'elle permet de distinguer les états initiaux et finaux. Dans le treillis une suite de transitions s'appelle chemin.

On remarque que les états du quantificateur autorisent l'utilisation des sous-alphabets C_0 et C_2 ou des sous-alphabets C_1 et C_3. Toutefois, le choix entre les deux quantificateurs possibles a aussi une implication fortement importante sur la façon dont seront quantifiés les échantillons suivants. De plus, le choix d'un sous-alphabet permis occasionne une transition vers un nouvel état [42]-[43].

Supposons par exemple que l'algorithme de quantification commence son activité dans l'état zéro (0) et qu'il quantifie un échantillon, x_0, tel que montré sur la figure 2.11. L'algorithme est alors devant une alternative : utiliser une valeur type appartenant à C_0 ou utiliser une valeur type appartenant à C_2. Dans chaque cas, une erreur de quantification sera commise : $e_0^{(0)}$ si une valeur type appartenant à C_0 est utilisée, $e_2^{(0)}$ si c'est une valeur type de C_2 qui est utilisée.

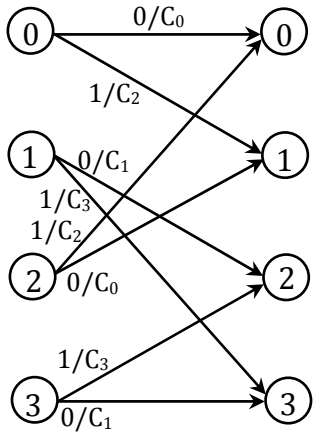


Figure 2.10: Treillis à quatre états étiqueté par quatre sous-alphabets.

Sauf dans le cas particulier où l'échantillon est exactement à mi-distance entre un des membres de C_0 et un des membres de C_2, l'une de ces erreurs est plus grande que l'autre. Il peut alors paraître surprenant que l'algorithme hésite entre les deux valeurs types plutôt que de se contenter de prendre celle étant la plus proche de l'échantillon. C'est que le choix entre les sous-alphabets C_0 et C_2 détermine directement les valeurs types qui seront disponibles pour la quantification du prochain échantillon, et par conséquent, l'ampleur de l'erreur qui sera commise sur celui-ci.

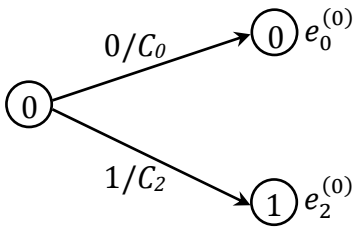


Figure 2.11: Quantification du premier échantillon par le quantificateur codé par treillis.

En effet, si un membre du sous-alphabet C_0 est utilisé pour quantifier l'échantillon x_0, le prochain échantillon, x_1, devra être quantifié avec un membre de B_0 puisque l'algorithme de quantification sera encore à l'état zéro (0). En revanche, l'utilisation d'un membre de C_2 pour quantifier x_0 force l'utilisation d'un membre de B_1 pour quantifier x_1. Or, à cause de la façon de la séparation de l'alphabet original, les sous-alphabets B_0 et B_1 sont complémentaires. Lorsqu'une valeur type de B_1 quantifie un échantillon donné avec le maximum d'erreur, il y a dans B_0 une autre valeur type qui le fait avec un minimum et vice-versa.

Comme le quantificateur ignore tout sur la nature du prochain échantillon, sa meilleure stratégie consiste à attendre de le connaître pour décider lequel des sous-alphabets, B_0 et B_1, contient la valeur type la plus appropriée pour quantifier cet échantillon avec le minimum d'erreur en moyenne [41].

Lorsque le deuxième échantillon se présente dans le quantificateur, ce dernier détermine, pour chacun des sous-alphabets C_0, C_1, C_2 et C_3, la valeur type la plus proche de l'échantillon en terme de distance euclidienne. Il calcule ensuite ce qu'il en coûterait d'utiliser chacune de ces valeurs types, soit les erreurs de quantification $e_0^{(1)}$, $e_1^{(1)}$, $e_2^{(1)}$ et $e_3^{(1)}$, respectivement associées aux sous-alphabets C_0, C_1, C_2 et C_3. Ces erreurs sont additionnées à celle qui a été commise pour atteindre l'état permettant d'utiliser ce sous-alphabet. On obtient ainsi une erreur de quantification qui représente l'erreur faite sur les deux premiers échantillons [41].

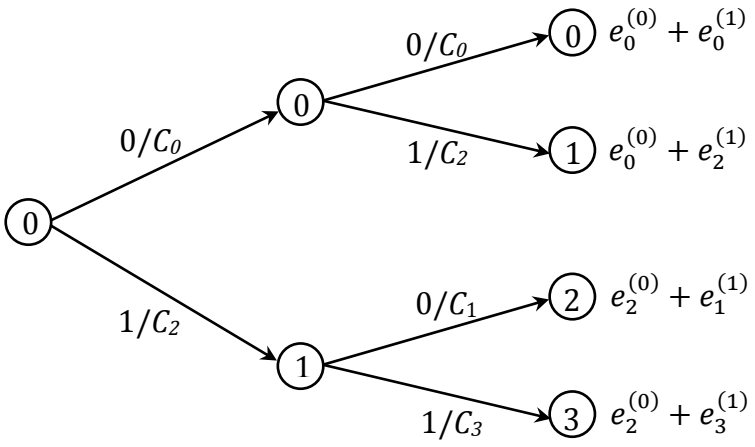


Figure 2.12 : Quantification du deuxième échantillon.

La figure 2.12 illustre cette procédure de quantification du deuxième échantillon. L'arbre binaire qui se développe dans la figure montre les quatre stratégies que le quantificateur codé par treillis élabore lors de la quantification des deux premiers échantillons.

Si l'algorithme de quantification devait choisir dès maintenant sa meilleure stratégie de quantification, il sélectionnerait bien sûr celle ayant cumulée la plus faible erreur de quantification, puisque cette stratégie correspond au meilleur compromis entre la plus faible erreur de quantification qu'il est possible de commettre sur chaque échantillon et le choix du sous-alphabet le plus approprié à la quantification de la séquence d'échantillons prise dans son ensemble.

Mais l'algorithme de quantification n'a aucune raison de faire ce choix maintenant. Puisque les alphabets dont il disposera pour quantifier les prochains échantillons dépendent du choix du sous-alphabet pour l'échantillon présent, il vaut sans doute mieux qu'il attende de connaître les échantillons suivants avant d'arrêter sa décision. Ainsi, il pourra raffiner ses hypothèses et trouver la meilleure stratégie appliquée à une plus longue séquence d'échantillons.

Mais l'accumulation des possibilités rendrait pour le moins hypothétique l'implantation d'un tel quantificateur pour des délais importants ; le nombre de stratégies de quantification à considérer doublant à chaque échantillon dans l'arbre binaire amorcé à la figure 2.11. Heureusement une astuce va permettre de diminuer de manière spectaculaire le nombre de branches qu'il est nécessaire de conserver pour trouver le meilleur compromis de quantification [41].

La figure 2.13 montre en effet que dès le troisième échantillon, chaque état peut être atteint par deux chemins différents. Or, un seul de ces chemins présente de l'intérêt : celui ayant cumulé la plus faible erreur de quantification.

Puisque les choix disponibles à un état donné sont indépendants de la façon dont cet état est atteint, les deux stratégies de quantification passant par un même état à un moment donné cumuleront par la suite deux différentes erreurs de quantification. La plus coûteuse de ces stratégies peut donc être éliminée d'office puisque les chemins qu'elle va engendrer n'ont aucune chance de constituer la meilleure stratégie de quantification, étant d'avance

battus par ceux de la stratégie la moins coûteuse. Il n'y aura donc que quatre chemins survivants, et ce peu importe le nombre d'échantillons que nous considérons.

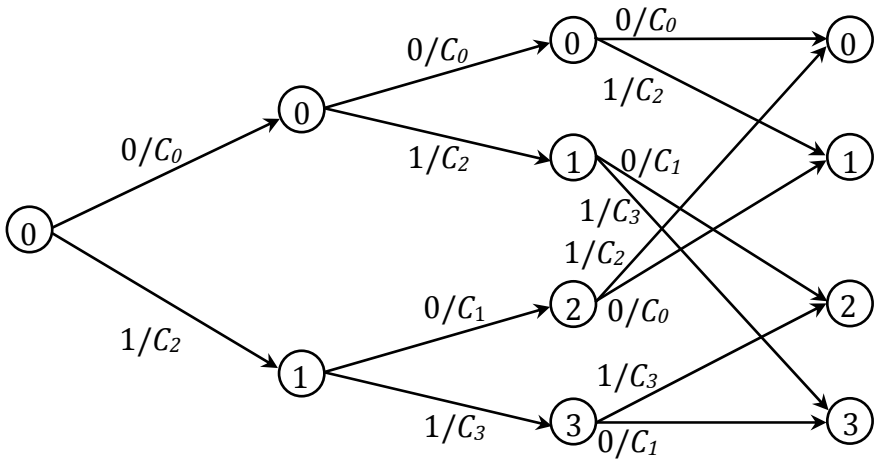


Figure 2.13 : Quantification du troisième échantillon.

La figure 2.14 montre le développement du treillis complet après la quantification de plusieurs échantillons, les chemins éliminés, et les chemins survivants.

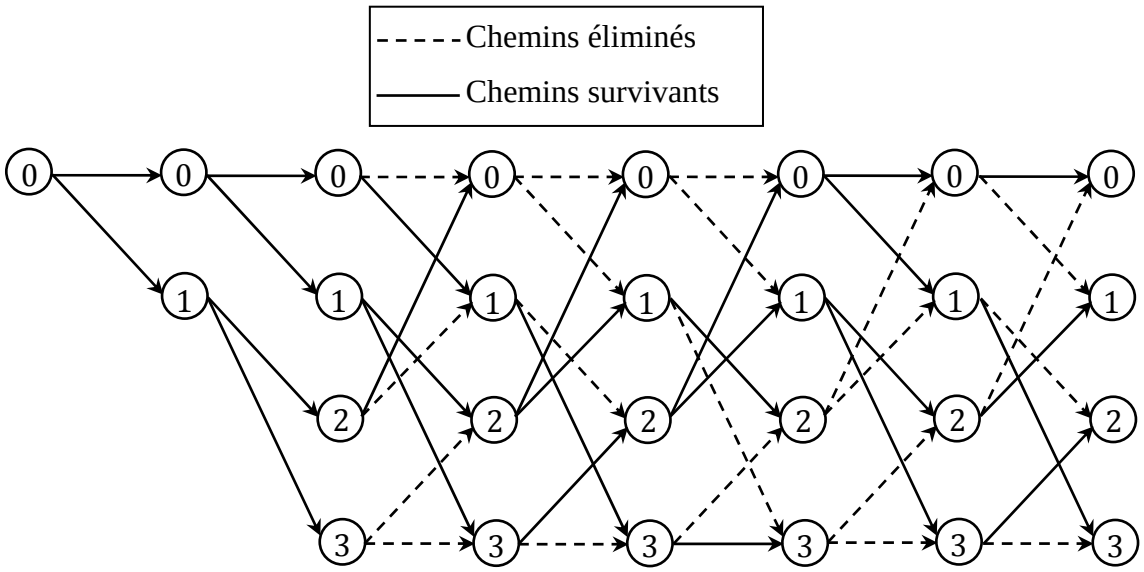


Figure 2.14 : Les chemins possibles dans le treillis à quatre états.

Après avoir quantifié un nombre prédéterminé d'échantillons, l'algorithme de quantification codé par treillis n'aura qu'à choisir la meilleure stratégie de quantification en sélectionnant le chemin qui possède la plus petite erreur cumulée.

Le code est formé en allouant, pour chaque échantillon, 1 bit pour la description du chemin à suivre dans le treillis et le reste des bits permis par le débit pour l'identification de la valeur type utilisée pour quantifier l'échantillon.

Le bit décrivant le chemin indique au décodeur laquelle des deux transitions quittant un état a été choisie par le quantificateur. Ceci détermine de manière unique l'indice du sous-alphabet C_i utilisé puisque selon l'état où se trouve le décodeur, un seul des sous-alphabets B_i est permis et que chacune de ses moitiés, les sous-alphabets C_i, sont associées à l'une de ces transitions. A l'aide du diagramme de la figure 2.9, le décodeur arrive facilement à suivre le chemin dans le treillis à l'aide d'un seul bit.

Les autres bits – le débit permis moins un bit – donnent, pour une transition, la valeur type utilisée dans le sous-alphabet. Ceci décrit complètement la valeur type utilisée puisque que chacun des alphabets C_i contient seulement la moitié des valeurs types permises par le débit.

L'ensemble des contraintes qui s'exercent sur le choix des membres de l'alphabet permet donc au quantificateur codé par treillis d'avoir une taille apparente deux fois plus grande que celle d'un quantificateur scalaire classique ; tout en conservant un débit égal à ce dernier [41].

2.4.4 Algorithme du quantificateur codé par treillis

On se propose maintenant de quantifier une séquence de P vecteurs de L dimensions par un treillis de N états et 2^m branches. Les indices i des vecteurs sont tels que : $i = 0, \cdots, P-1$. l'erreur $e_j^{(i)}$ sera la petite erreur ou distance euclidienne entre un vecteur d'indice i à quantifier et un vecteur type le plus proche voisin dans le sous-alphabet S_j considéré avec $j = 0, \cdots, K$.

L'algorithme se subdivise en sept étapes :

Etape 1 : Soient $E_0, E_1, \cdots, E_{T-1}$, les erreurs cumulées de chacun des N états du treillis. L'une de ces erreurs se voit assigner initialement la valeur zéro (0), les autres une valeur infinie, ou du moins très grande. Le codeur et le décodeur doivent s'entendre au préalable sur le choix de l'état ayant une erreur nulle.

Etape 2 : Prendre L échantillons de la séquence et former un vecteur i à L dimensions.

Etape 3 : Calculer les erreurs $e_j^{(i)}$ correspondant à l'erreur de quantification causée par le meilleur représentant de chacun des sous-alphabets S_j, $j = 0, \cdots, K-1$.

Etape 4 : Pour chaque état du treillis, calculer les erreurs cumulées des transitions qui mènent à cet état. On fait ce calcul en sommant l'erreur cumulée de l'état parent avec l'erreur $e_j^{(i)}$ du sous-alphabet associé à la transition arrivant à l'état considéré. Conserver celle des deux transitions qui donne l'erreur cumulée la plus faible pour cet état.

Etape 5 : Pour chacun des états, mémoriser le chemin survivant (1 bit) ainsi que l'indice du vecteur type du sous-alphabet associé à la transition ayant gagnée la compétition de l'étape 4.

Etape 6 : Si moins de P vecteurs ont été considérés, reprendre à l'étape 2. P est ici la profondeur du treillis.

Etape 7 : Si P vecteurs ont été considérés, choisir parmi les N états celui ayant la plus faible erreur cumulée et coder les informations le concernant telles que les transitions et indices des vecteurs types des sous-alphabets pour chaque étage. Reprendre à l'étape 1.

L'algorithme de codage source qui vient d'être décrit est l'algorithme de Viterbi [44].

Dans l'étape 1 de l'algorithme du quantificateur codé par treillis, on assigne une erreur cumulée nulle à l'un des états du treillis, qu'on appelle *état initial*, et une erreur infinie aux autres ce qui signifie qu'initialement on a une *recherche en arbre* (les $\log_2 N$

premières étapes) et qui devient par la suite une *recherche en treillis*. Pour les longues séquences, le choix de l'état initial n'influe pas sur les performances du quantificateur. Par contre, dans le cas des courtes séquences, le choix de l'état initial est capital. La meilleure solution pratique dans ce cas est d'attribuer une erreur nulle à tous les états ce qui équivalent à dire qu'on a une recherche en treillis à partir de la première étape. Dans ce cas, on rajoute au débit total les bits désignant l'état initial ($\log_2 N$ bits) déterminé à la fin de la quantification de la séquence.

2.5 Complexité de la quantification codée par treillis

On peut résumer l'opération de codage par cette technique en deux étapes: dans une première étape d'exécution, pour chaque vecteur i, de la séquence à coder, on cherche le plus proche vecteur type dans chaque sous-alphabet qui correspond à une distorsion d_j. Dans une deuxième étape, l'algorithme de Viterbi est utilisé pour trouver le chemin qui minimise la métrique cumulée, telle que la métrique d'une branche associée à un sous alphabet S_j est la distorsion trouvée dans la première étape.

La complexité de codage par l'algorithme de Viterbi, pour un quantificateur vectoriel codée par treillis (TCVQ), peut être facilement évaluée. Dans un codeur TCVQ, il y a $2^{\tilde{R}L+m}$ sous-alphabets, chacune contenant 2^{RL-m} vecteurs types.

Dans la première étape de codage, pour trouver le plus proche voisin vecteur type dans chaque sous-alphabet pour un vecteur source i de L dimensions, il faut:

$L \cdot 2^{RL-m}$ additions,
$L \cdot 2^{RL-m}$ multiplications,
$2^{RL-m} - 1$ comparaisons.

Le treillis possède N états et 2^m branches rejoignent chaque nœud, ce qui signifie que la détermination du survivant pour chaque état, dans la deuxième étape de codage, exige:

2^m additions,
$2^m - 1$ comparaisons.

La complexité totale par vecteur de L dimensions est donc:

$L \cdot 2^{(R+\tilde{R})L}$ multiplications,

$L \cdot 2^{(R+\tilde{R})L} + N \cdot 2^m$ additions,

$2^{\tilde{R}L+m} \cdot (2^{RL-m} - 1) + N \cdot (2^m - 1)$ comparaisons.

A titre de comparaison, la complexité de codage par un quantificateur vectoriel classique (VQ) opérant sur des vecteurs de L' dimensions est donné par:

$L' \cdot 2^{RL'}$ multiplications,

$L' \cdot 2^{RL'}$ additions,

$2^{RL'} - 1$ comparaisons.

On remarque que la TCVQ a une complexité élevée par rapport à la VQ classique dans le cas où $L = L'$. C'est donc le coût à payer pour augmenter les performances. D'un autre coté, dans le cas où les deux systèmes de codage, VQ classique et TCVQ, ont la même distorsion (souvent, $L < L'$), il est possible que le système TCVQ soit moins complexe que la VQ à recherche exhaustive.

En se basant sur une évaluation expérimentale des performances de la TCVQ, Fischer *et al.* [39] ont conclu qu'on peut modéliser la variation de l'erreur quadratique moyenne en fonction de N et L. A partir de là, si on fixe le niveau de complexité à une valeur donnée, on peut sélectionner des valeurs optimales de N et L pour une application donnée.

2.6 Conception d'un quantificateur codé par treillis

Après avoir donné une idée sur le principe d'un système TCVQ, on présente, dans ce paragraphe, la procédure à suivre pour concevoir un quantificateur codé par treillis dans le cas général.

2.6.1 Treillis utilisés

Comme on a vu précédemment, le treillis peut être vu comme un arbre dont le nombre d'états (nœuds) croit de façon exponentielle au début, puis reste constant (égal à N).

Comme dans le cas de l'arbre, le treillis est une représentation graphique des chemins qui peuvent être suivis par le décodeur, à partir d'un état initial en fonction de l'information codée qu'il reçoit.

Après l'étalement initial et précisément après $\log_2 N$, le treillis à N états est une juxtaposition répétitive du diagramme de transition d'état de son automate associé.

Les treillis utilisés par Fischer *et al.* [38],[39] dans la TCQ sont ceux d'Ungerboeck qui sont engendrés par des codeurs convolutionnels. Dans [45], Ungerboeck propose une procédure de recherche de bons treillis dans le but de maximiser la distance entre les séquences distinctes de symboles-canal. Il donne une série de treillis utilisant les constellations suivantes:

- Modulation d'amplitude: 4AM, 8AM, etc.
- Modulation de phase: 8PSK, 16PSK.
- Modulation d'amplitude/phase: 16QAM,…

Dans ce travail, on utilise uniquement les treillis d'Ungerboeck pour modulation d'amplitude. De cette façon on exploite la dualité entre la constellation pour modulation d'amplitude et l'alphabet de reproduction pour la quantification scalaire. Par la suite, lorsqu'on fait référence à un treillis à N états, on sous-entend implicitement qu'il s'agit du treillis d'Ungerboeck pour modulation d'amplitude à N états.

Tous ces treillis peuvent être engendrés à partir du diagramme de transition d'état d'un codeur convolutionnel de taux 1/2. Le schéma général d'un tel codeur est présenté à la figure 2.15. Dans la figure 2.16 on présente le codeur convolutionnel à 8 états (le treillis associé à ce codeur convolutionnel se trouve dans la figure 2.21).

Les codeurs convolutionnels sont définis à l'aide de deux polynômes à coefficients binaires, h^0 et h^1. On associe à ces polynômes les nombres formés par concaténation de leurs coefficients, qui sont donnés en octal [45]. Selon le coefficient d'ordre $i+1$ (bit $i+1$) de h^0, noté h^0_{i+1}, est égal à 1 ou 0, le dernier bit $T_0(n-1)$ de l'état précédent entre ou non dans la somme modulo 2 pour le calcul du bit i de l'état $T_i(n)$ (équation 2.11). Selon que le bit h^1_{i+1} est égal à 1 ou 0, le bit d'entrée $x_0(n-1)$ entre ou non dans la somme

modulo 2 pour le calcul de $T_i(n)$. Les équations qui définissent la sortie du codeur convolutionnel en fonction de l'entrée et de l'état du codeur sont:

$$y_1(n) = x_0(n) \quad \text{et} \tag{2.8}$$

$$y_0(n) = T_0(n). \tag{2.9}$$

Les équations qui définissent l'état suivant en fonction de l'entrée et de l'état du codeur sont:

$$T_{\nu-1}(n) = T_0(n-1) \quad \text{et} \tag{2.10}$$

$$T_i(n) = T_{i+1}(n-1) \oplus h_{i+1}^0 T_0(n-1) \oplus h_{i+1}^1 x_0(n-1), \qquad i = 0, \cdots, \nu-2 \tag{2.11}$$

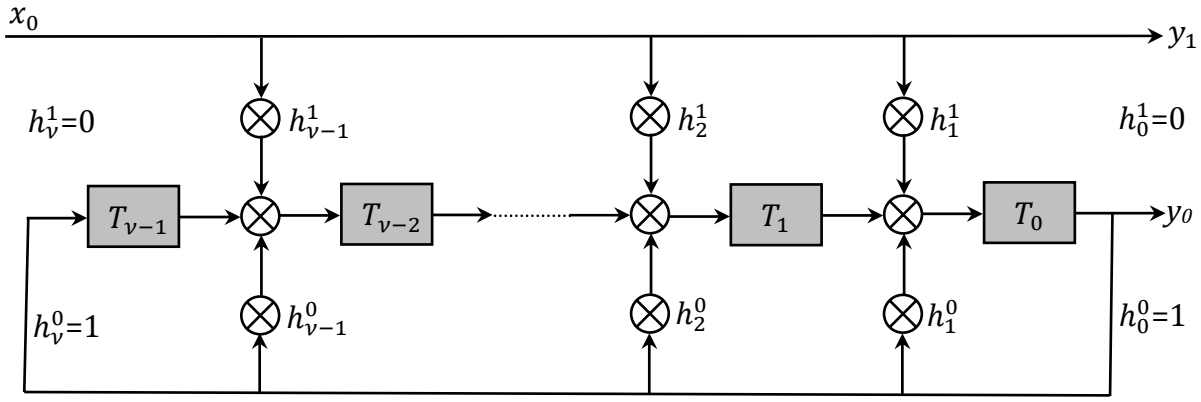


Figure 2.15: Schéma général d'un codeur convolutionnel de taux 1/2.

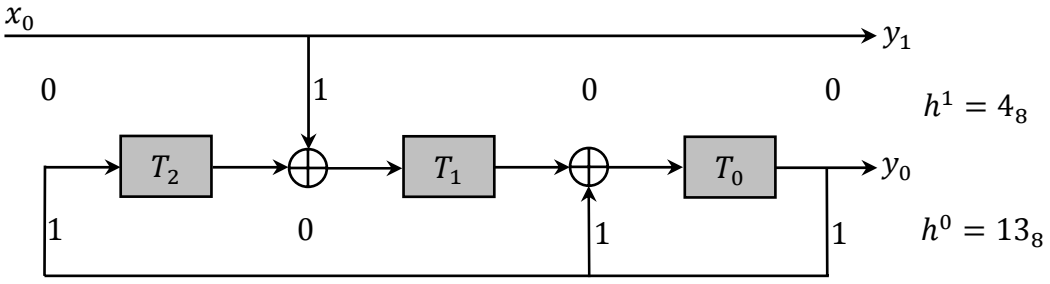


Figure 2.16 : Codeur convolutionnel à 8 états pour modulation d'amplitude.

Dans la représentation du codeur convolutionnel (figure 2.15), les bits à 1 de h^0 signifient une liaison entre le dernier bit du registre d'état et un additionneur modulo 2 et les bits à 1 de h^1 signifient une liaison entre le bit d'entrée et un additionneur modulo 2.

Les polynômes définissant les codeurs convolutionnels pour des treillis allant jusqu'à 256 états sont donnés dans la table 2.1.

Tableau 2.1: Les polynômes associés aux codeurs convolutionnels à $4 \cdots 256$ états.

Nombre d'états	4	8	16	32	64	128	256
h^0	5_8	13_8	23_8	45_8	103_8	235_8	515_8
h^1	2_8	4_8	4_8	10_8	24_8	126_8	362_8

2.6.2 Dictionnaire du treillis

Le dictionnaire A_0 d'un système TCVQ à R bits/échantillon contient $2^{(R+\tilde{R})L}$ vecteurs de L dimensions. Sa conception est faite suivant le principe de l'algorithme de Lloyd-Max [28] dans le cas scalaire et l'algorithme LBG [31] dans le cas vectoriel.

Puisque la complexité de la TCVQ augmente exponentiellement avec le produit $\tilde{R}L$, il faut donc accorder une grande importance au choix de $\tilde{R}$ dans la conception d'un codeur TCVQ de façon à avoir des performances supérieures à celles d'un VQ classique avec un minimum de $\tilde{R}$. La théorie débit-distorsion avec contrainte sur la taille de l'alphabet de sortie donne une réponse à ce problème [39].

Les courbes débit-distorsion donnés à la section (§2.4.1) pour le cas scalaire, montre qu'il suffit de doubler la taille de l'alphabet, c'est-à-dire $\tilde{R} = 1$, pour les sources uniforme et gaussienne pour s'approcher de la fonction débit distorsion optimale. Mais pour la source de Laplace, il est nécessaire de quadrupler la taille de l'alphabet, ce qui signifie $\tilde{R} = 2$.

Fischer *et al.* [39] ont étudié le cas vectoriel bidimensionnel ($L = 2$) et ils ont trouvé qu'il faut prendre $\tilde{R} \geq 1$pour converger vers la caractéristique débit distorsion optimale pour les faibles débits. Pour coder à des débits supérieurs, il faudra que $\tilde{R}$ soit large. Il est exigé que $\tilde{R} \geq 1.5$ dans le cas de la source de Laplace pour s'approcher de la limite théorique de Shannon.

2.6.3 Partage du dictionnaire

Le dictionnaire du treillis, déjà conçu, est partitionné suivant la règle de « set partitioning » d'Ungerboeck en $K = 2^{\tilde{R}L+m}$ sous-dictionnaires notés $S_i, i = 0, \cdots, K-1$, ce qui veut dire que chaque sous dictionnaire contient 2^{RL-m} vecteurs.

Rappelons que la méthode de partition d'ensemble d'Ungerboeck commence par une constellation de modulation (pour la TCVQ cela correspond au dictionnaire du treillis) qui est partitionnée en deux sous-ensembles ayant le même nombre d'éléments, puis chacun de ces deux sous-ensembles est partitionné à son tour en deux, etc. Chaque partition est faite dans le but de maximiser la distance Δ_i entre les deux symboles les plus proches contenus dans chaque sous-ensemble résultant.

Suivant les notations d'Ungerboeck, le dictionnaire optimisé A_0 (de taille $2^{(R+\tilde{R})L}$) est partitionné en deux sous-ensembles B_0 et B_1 de taille $2^{(R+\tilde{R})L-1}$; chacun de ces sous-ensembles est ensuite partitionné en deux. On obtient quatre sous-ensembles C_0 , C_2, C_1, C_3. de taille $2^{(R+\tilde{R})L-2}$. Chaque partition est faite de manière à maximiser la distance minimale à l'intérieur des sous-ensembles qui en résultent. Par exemple, la partition de A_0 en B_0 et B_1 doit maximiser:

$$\Delta_{min} = min\left\{\min_{x,y\in B_0} d(x,y)\,, \min_{x,y\in B_1} d(x,y)\right\} \qquad (2.12)$$

Ici $d(x,y)$ est la distance euclidienne entre x et y.

Les noms donnés aux sous-dictionnaires lors de ce processus de partage dépendent du chemin à suivre pour les atteindre dans l'arbre de partage. En effet, la lettre dépend du niveau occupé par le sous-dictionnaire dans cet arbre ; alors que l'indice est un nombre binaire formé, de la droite vers la gauche, au fur et à mesure de la descente. Chaque fois que l'embranchement de gauche est pris, un 0 est ajouté à la gauche de l'indice « parent » ; alors qu'un 1 est ajouté lorsque l'embranchement de droite est pris (figure 2.17).

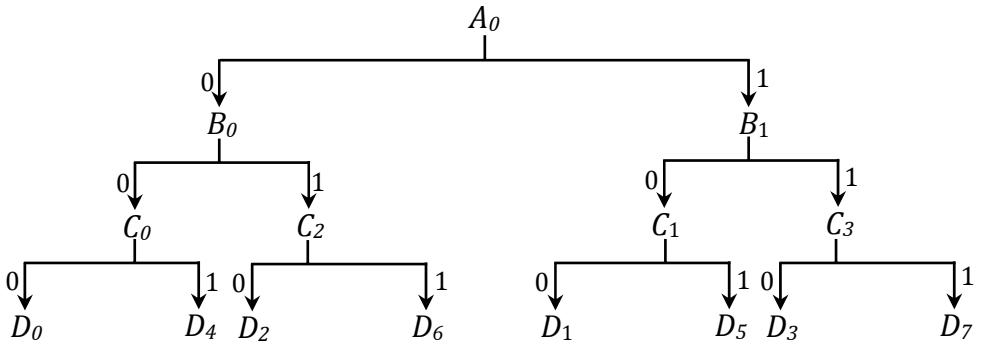


Figure 2.17: la partition d'un dictionnaire d'après les règles d'Ungerboeck en 8 classes. Remarquant que le dictionnaire subit une suite de partitions symétriques qui forme un arbre binaire. Le numéro des sous alphabets résultants correspond à leur position dans l'arbre des partitions.

Dans le cas scalaire, afin de partitionner un dictionnaire quelconque en quatre sous dictionnaires, Marcellin *et al.* [38] ont proposé une méthode simple et efficace. Il suffit d'attribuer les niveaux de quantification de façon croissante (D_0, D_1, D_2, D_3, D_0, D_1, D_2, D_3,…) jusqu'au niveau de quantification le plus grand comme le montre la figure 2.18.

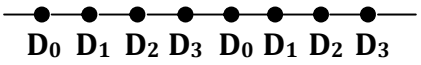


Figure 2.18: partition d'un alphabet de reproduction en 4 sous alphabets pour une TCQ à 2 bits/échantillon.

Dans le cas vectoriel, le dictionnaire du treillis n'a pas toujours une structure régulière comme dans le cas de la modulation codée. La partition analytique du dictionnaire est alors impossible et on est amené à apporter une solution algorithmique à ce problème. La solution directe est de maximiser Δ_{min} pour toutes les partitions possibles de A_0 dans deux sous-ensembles de même taille. Mais, quand $(R+\tilde{R})L$ augmente, cette méthode devient très coûteuse en calculs, rendant son implantation irréaliste. Par exemple, pour $(R+\tilde{R})L = 6$ bits/échantillon, il s'agit de choisir une partition parmi $\binom{64}{32}$, qui est de l'ordre de 10^{18} [46].

Une solution serait d'utiliser un dictionnaire algébrique dont la symétrie nous permettrait de le partitionner analytiquement suivant les règles d'Ungerboeck [46].

Wang et Moayeri [47] ont proposé une autre solution pour partitionner un alphabet vectoriel en deux sous-alphabets en respectant toujours la règle d'Ungerboeck. En effet, pour partitionner le dictionnaire en $2^{\tilde{R}L+m}$ sous-dictionnaires, on fait appel à cet algorithme $\tilde{R}L+m$ fois et de façon similaire à la TCM pour le numérotage des sous dictionnaires résultants (figure 2.17). Cet algorithme est constitué des étapes suivantes :

Etape 1 : Etant donné le dictionnaire du treillis A_0, conçu par l'algorithme LBG, de taille $M = 2^{(R+\tilde{R})L}$ vecteurs types que l'on veut partitionner en deux sous alphabets B_0 et B_1. Les distances entre toutes les paires possibles de vecteurs-types sont calculées et classées par ordre croissant dans un tableau avec les paires correspondantes. Cette opération nous permet d'avoir un tableau de taille $M \cdot (M-1)/2$, où une entrée d'indice i de ce tableau correspond aux vecteurs types V_i et W_i qui sont à une distance $d_i = \|V_i - W_i\|$.

Etape 2 : Mettre V_0 et W_0 dans les sous alphabets B_0 et B_1 respectivement, et enlever la première entrée du tableau.

Etape 3 : Chercher dans le tableau un index j tel que $\forall i < j$, ni V_i ni W_i n'appartiennent à l'union $B_0 \cup B_1$ mais au moins un des deux vecteurs V_j ou W_j appartient à l'union $B_0 \cup B_1$.

Etape 4 : Si les deux vecteurs types V_j et W_j appartiennent à l'union $B_0 \cup B_1$, ce qui signifie que les deux vecteurs types sont déjà assignés, mettre:

$$V_i \leftarrow V_{i+1}; \qquad W_i \leftarrow W_{i+1}; \qquad d_i \leftarrow d_{i+1}; \quad \forall i \geq j.$$

Puis aller à l'étape 3.

Etape 5 :

- Si V_j appartient à B_0 (ou à B_1), alors mettre W_j dans B_1 (ou dans B_0).
- Si W_j appartient à B_0 (ou à B_1), alors mettre V_j dans B_1 (ou dans B_0).

Etape 6 : Si la taille de B_0 (ou de B_1) a atteint $M/2$, alors mettre les vecteurs types non assignés (restants) dans B_1 (ou dans B_0) et Stop**.** Sinon, aller à l'étape 3.

2.6.4 Etiquetage des branches du treillis

Dans le schéma général d'un codeur convolutionnel de taux 1/2 (figure 2.15), il y a un code de sortie $y_1 y_0$ (écrit en binaire) qui représente en réalité l'indice du sous-dictionnaire étiquetant la branche liée entre l'état passé et l'état présent pour un numéro de transition donné. Par exemple, pour un treillis à 4 états, la figure 2.19 montre son circuit convolutionnel où le code sortie est défini par:

$$y_1 = r \text{ et} \tag{2.13}$$

$$y_0 = T_0 \, . \tag{2.14}$$

Comme illustration, à partir de l'état 1 ($T_1 T_0 = 01$), on peut aller à l'état 2 ($T_1 T_0 = 10$) en empruntant la transition $r = 0$ (utilisation du sous dictionnaire S_1), ce qui produit un code sortie $y_1 y_0 = 01$; ou encore, on peut aller à l'état 3 ($T_1 T_0 = 11$) en empruntant la transition $r = 1$ (utilisation du sous dictionnaire S_3), ce qui produit le code sortie $y_1 y_0 = 11$.

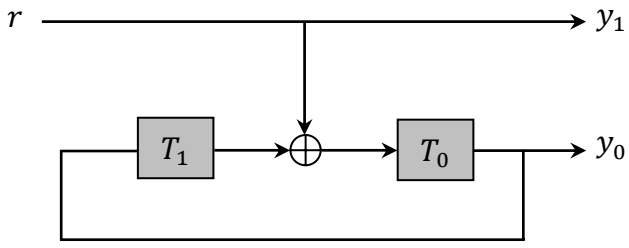


Figure 2.19: circuit à quatre états. $y_1 y_0$ nombre écrit en binaire représente l'indice du sous-dictionnaire utilisé.

Mais le problème est posé pour un nombre des sous-dictionnaires supérieur à quatre. Fischer *et al.* [39] ont donné des règles générales pour l'étiquetage des treillis d'Ungerboeck :

Règle 1 : l'union des sous-alphabets associés aux branches quittant ou rejoignant un état du treillis doit être un quantificateur raisonnable pour la source à quantifier.

Règle 2 : la fréquence relative d'apparition d'un sous dictionnaire comme étiquette des branches du treillis est liée au poids représentatif dans la source à quantifier.

Malone *et al.* [48] ont donné une méthode simple pour l'étiquetage des treillis d'Ungerboeck par K sous-dictionnaires. Si l'on considère que S_i et $S_{i+K/2}$, sont les sous alphabets du même ensemble intermédiaire dans la dernière étape du set partitioning d'Ungerboeck (voir figure 2.17), pour l'étiquetage, S_i et $S_{i+K/2}$ sont les sous-dictionnaires associés aux branches rejoignant un état du treillis donné (figure 2.20).

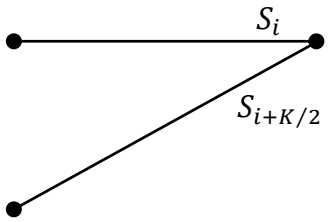


Figure 2.20: Etiquetage du treillis en utilisant la méthode de Malone et al. [48].

La figure 2.21 donne un exemple d'étiquetage d'un treillis à 8 états par 4 sous-dictionnaires et 8 sous-dictionnaires. On attribue respectivement les branches du haut aux premiers sous-dictionnaires et les branches du bas aux deuxièmes sous-dictionnaires.

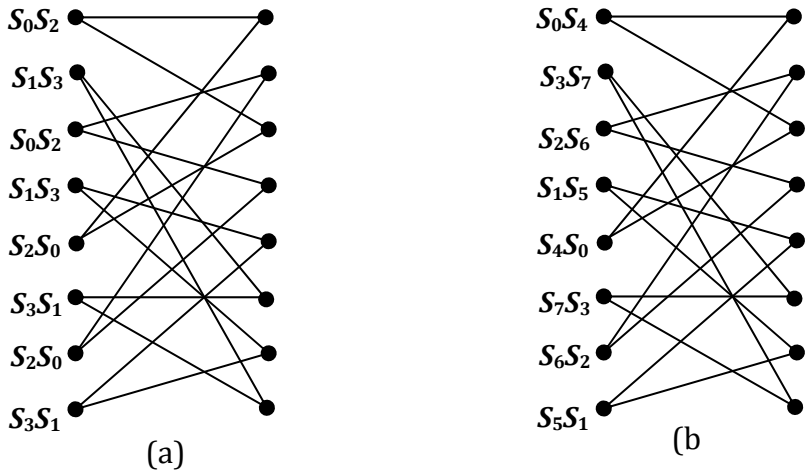


Figure 2.21: Etiquetage d'un treillis à 8 états par (a) 4 sous-dictionnaires. (b) 8 sous-dictionnaires.

2.6.5 Optimisation du dictionnaire du treillis

Il est désirable d'optimiser le dictionnaire du treillis afin d'augmenter les performances du système du codage TCVQ. Cette optimisation doit prendre en compte la structure de la machine à états finis. Dans [38], une méthode "Quasi-Newton"[49] est utilisée pour optimiser un alphabet scalaire. D'après Fischer [39], la complexité des calculs de cette méthode augmente avec la taille de l'alphabet et surtout avec la dimension des vecteurs. Alternativement, l'algorithme "Trellis GLA" de Stewart *et al.* [50] peut être utilisé pour optimiser l'alphabet du treillis. Dans notre travail, on a choisi cet algorithme pour les deux cas, scalaire et vectoriel. Une convergence plus rapide est obtenue avec cet algorithme si l'on impose, de plus, une symétrie au dictionnaire.

Le déroulement de cet algorithme est le suivant :

Etape 1 : Etiqueter le treillis avec les sous alphabets $S_i, i = 0,1,\cdots,K-1$ résultants de la partition du dictionnaire du treillis A^0. Mettre $j = 0$.

Etape 2 : Coder une séquence d'apprentissage représentative χ en utilisant l'algorithme de Viterbi et la structure TCVQ avec l'alphabet A^j. Le processus de codage induit une partition en classes de la base d'apprentissage, chaque classe B_i^j étant composée des vecteurs codés utilisant le vecteur $y_i^j \in A^j$. La distorsion résultante est donnée par :

$$\rho(j) = \frac{1}{\|\chi\|}\sum_{x\in\chi} d(x,y) \tag{2.15}$$

où $\|\chi\|$ est la longueur de la séquence d'apprentissage χ.

Etape 3 : Arrêter l'algorithme si une condition convenable est accomplie, par exemple si entre deux itérations successives la distorsion n'a augmenté que d'une quantité très faible, ce qui équivaut à l'écriture mathématique suivante:

$$|\rho(j-1) - \rho(j)|/\rho(j) < \epsilon \tag{2.16}$$

Sinon continuer.

Etape 4 : Remplacer chaque y_i^j par un vecteur y_i^{j+1} qui représente la classe B_i^j avec une moindre distorsion. Si on utilise comme critère de distorsion la distance

euclidienne, les vecteurs y_i^{j+1} sont les centres de gravité des classes B_i^j, ce qui signifie:

$$y_i^{j+1} = \frac{1}{\left\|B_i^j\right\|} \sum_{x \in B_i^j} x \tag{2.17}$$

Mettre $j \leftarrow j+1$ et aller à l'étape 2

La séquence d'apprentissage utilisée dans l'algorithme de Stewart contient des symboles représentatifs de la source pour laquelle le quantificateur est optimisé. Typiquement, on considère que la séquence d'apprentissage doit contenir au minimum dix symboles de source par symbole de reproduction optimisé [31].

2.7 Performances de la quantification codée par treillis

Nous donnons les résultats pour la TCQ des sources uniforme, gaussienne et laplacienne sans mémoire (leurs fonctions de densité et de probabilité sont définies dans le tableau 2.2) en fonction du débit de codage et de la taille du treillis.

Tableau 2.2 : Trois modèles de sources sans mémoire avec une moyenne égale à zéro et une variance unitaire.

Source	Notation	$f_X(x)$
Uniforme ou Rectangulaire	U	$\frac{1}{\sqrt{12}}$, si $\lvert x\rvert < \sqrt{3}$ 0, ailleurs
Gaussienne ou Normale	G $N(0,1)$	$\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$
Laplacienne	L	$\frac{1}{\sqrt{2}} \exp\left(-\lvert x\rvert\sqrt{2}\right)$

Pour chaque type de source, les rapports signal sur bruit (RSB) sont estimés pour le codage d'une séquence de 100000 échantillons (un tirage différent de celui utilisé dans l'optimisation des sous-dictionnaires). Le rapport signal sur bruit, est défini comme suit :

$$RSB = 10 \log_{10}(S/D) \tag{2.18}$$

où :

S : représente la variance de la source.

D : est la distorsion (la variance de l'erreur de quantification).

Pour permettre un calcul fiable du rapport signal sur bruit évoqué ci-dessus, la séquence à coder est divisée en ($T = 100$) blocs de ($P = 1000$) échantillons. Pour calculer les intervalles de confiance, La variance S_i et la distorsion D_i de chaque bloc sont considérées comme des variables aléatoires. L'intervalle de confiance est estimé de la manière suivante : Les variances totales de la source et du bruit de quantification sont calculées par : $S = \frac{1}{T}\sum_{i=1}^{T} S_i$ et $D = \frac{1}{T}\sum_{i=1}^{T} D_i$. Puisque chaque expérience implique $P = 1000$ échantillons, alors ça sera éventuellement valide pour assumer S_i et D_i comme des variables aléatoires normalement distribuées. Donc pour les variances totales *S* et *D*, les intervalles de confiance à $\alpha \times 100\%$ sont :

$$\left(S - z_\alpha\, \sigma_S/\sqrt{T}\,, S + z_\alpha\, \sigma_S/\sqrt{T}\right) \qquad \text{et} \qquad \left(D - z_\alpha\, \sigma_D/\sqrt{T}\,, D + z_\alpha\, \sigma_D/\sqrt{T}\right),$$

où : $\sigma_S^2 = \frac{1}{T-1}\sum_{i=1}^{T}(S_i - S)^2$, $\sigma_D^2 = \frac{1}{T-1}\sum_{i=1}^{T}(D_i - D)^2$, et z_α est choisie telle que :

$$\int_{-z_\alpha}^{z_\alpha} f_{T-1}(y)\, dy = \alpha \tag{2.19}$$

Où : $f_{T-1}(y)$ est la fonction de densité de probabilité de Student à $(T-1)$ degrés de liberté. La probabilité que S et D soient à l'intérieur de leurs intervalles de confiance respectifs est $\alpha \cdot \alpha$ et l'intervalle de confiance résultant à $\alpha^2 \times 100\%$ pour S/D est :

$$(S/D) \in \left[\frac{S - z_\alpha\, \sigma_S/\sqrt{T}}{D + z_\alpha\, \sigma_D/\sqrt{T}}, \frac{S + z_\alpha\, \sigma_S/\sqrt{T}}{D - z_\alpha\, \sigma_D/\sqrt{T}}\right] \tag{2.20}$$

pour $\alpha^2 = 0.95$, $z_\alpha = 2.27$ (peut être obtenue en résolvant l'équation 2.19, soit d'une façon numérique ou par utilisation des tables de loi Student $\alpha \approx 0.975$) [51].

Dans un premier temps, on a évalué les performances de la TCQ dans le cas d'un alphabet doublé ($\widetilde{R} = 1$) en terme rapport signal sur bruit (RSB). Pour apprécier l'amélioration apportée par l'optimisation du dictionnaire du treillis (§2.6.5), on a donné les résultats trouvés pour une TCQ utilisant les niveaux de quantification de Lloyd-Max et une TCQ utilisant les niveaux issus de l'optimisation par "Trellis GLA".

Les tableaux 2.3, 2.4 et 2.5 donnent les RSB moyens obtenus lors de la quantification par treillis en utilisant les niveaux de Lloyd-Max à des débits entiers de 100 séquences de 1000 échantillons pour les sources (sans mémoire) uniforme, gaussienne et laplacienne respectivement.

Pour chaque type de source, les RSB sont estimés pour le codage d'une séquence de 100000 échantillons (un tirage différent de celui utilisé dans l'optimisation des sous-dictionnaires). Les tableaux 2.2, 2.3 et 2.4 donnent les RSB moyens obtenus lors de la quantification par treillis en utilisant les niveaux de Lloyd-Max à des débits entiers de 100 séquences de 1000 échantillons pour les sources (sans mémoire) uniforme, gaussienne et laplacienne respectivement.

On a donc calculé pour ces RSB moyens, les intervalles de confiance à 95% qui sont de : ±0.04 dB, ±0.07 dB et ±0.12 dB pour les sources uniforme, gaussienne et laplacienne respectivement.

Tableau 2.3: Rapports signal sur bruit moyens obtenus par TCQ d'une source uniforme utilisant les niveaux de Lloyd-Max. A titre de comparaison, nous reproduisons les RSB obtenus avec un quantificateur de Lloyd-Max et la limite théorique de Shannon.

Débit (bpe)	Dimension du treillis							Quantificateur Lloyd-Max	D(R)
	4 états	8 états	16 états	32 états	64 états	128 états	256 états		
1	5.81	5.97	6.08	6.15	6.22	6.31	6.34	6.02	6.79
2	12.48	12.62	12.71	12.78	12.84	12.91	12.96	12.04	13.21
3	18.80	18.92	18.99	19.07	19.13	19.19	19.23	18.06	19.42

En exécutant une optimisation par l'algorithme "Trellis GLA" avec une séquence d'apprentissage de 100000 échantillons, on trouve les résultats inscrits dans les tableaux 2.6, 2.7 et 2.8 et on a estimé aussi les intervalles de confiance à 95% qui sont de : ±0.04 dB, ±0.08 dB et ±0.14 dB pour les sources uniforme, gaussienne et laplacienne respectivement.

Tableau 2.4: Rapports signal sur bruit moyens obtenus par TCQ d'une source gaussienne utilisant les niveaux de Lloyd-Max. A titre de comparaison, nous reproduisons les RSB obtenus avec un quantificateur de Lloyd-Max et la limite théorique de Shannon.

Débit (bpe)	Dimension du treillis							Quantificateur Lloyd-Max	D(R)
	4 états	8 états	16 états	32 états	64 états	128 états	256 états		
1	4.66	4.81	4.89	4.97	5.03	5.09	5.14	4.40	6.02
2	10.18	10.30	10.36	10.45	10.50	10.56	10.60	9.30	12.04
3	15.83	15.94	16.03	16.10	16.16	16.23	16.27	14.62	18.06

Tableau 2.5: Rapports signal sur bruit moyens obtenus par TCQ d'une source laplacienne utilisant les niveaux de Lloyd-Max. A titre de comparaison, nous reproduisons les RSB obtenus avec un quantificateur de Lloyd-Max et la limite théorique de Shannon.

Débit (bpe)	Dimension du treillis							Quantificateur Lloyd-Max	D(R)
	4 états	8 états	16 états	32 états	64 états	128 états	256 états		
1	3.71	3.81	3.90	3.97	4.03	4.08	4.12	3.01	6.62
2	8.89	9.02	9.10	9.18	9.24	9.30	9.34	7.54	12.66
3	14.38	14.52	14.59	14.67	14.74	14.80	14.84	12.64	18.68

Tableau 2.6: Performance de la TCQ pour une source uniforme sans mémoire utilisant un alphabet de reproduction doublé et optimisé. Sont également donnés les RSB obtenus avec le quantificateur scalaire optimal (Lloyd Max) ainsi que la limite théorique de Shannon.

Débit (bpe)	Dimension du treillis							Quantificateur Lloyd-Max	D(R)
	4 états	8 états	16 états	32 états	64 états	128 états	256 états		
1	6.27	6.37	6.43	6.48	6.51	6.58	6.60	6.02	6.79
2	12.64	12.76	12.83	12.88	12.94	13.00	13.03	12.04	13.21
3	18.87	18.98	19.04	19.11	19.16	19.22	19.26	18.06	19.42

On remarque que, de meilleurs résultats s'obtiennent avec un dictionnaire optimisé, et cela est dû à l'efficacité de l'algorithme qui prend en compte la structure de la machine à états finis engendrant le treillis.

Tableau 2.7: Performance de la TCQ pour une source gaussienne sans mémoire utilisant un alphabet de reproduction doublé et optimisé. Sont également donnés les RSB obtenus avec le quantificateur scalaire optimal (Lloyd Max) ainsi que la limite théorique de Shannon.

Débit (bpe)	Dimension du treillis							Quantificateur Lloyd-Max	D(R)
	4 états	8 états	16 états	32 états	64 états	128 états	256 états		
1	5.02	5.21	5.28	5.36	5.44	5.52	5.57	4.40	6.02
2	10.53	10.68	10.76	10.84	10.90	10.96	11.01	9.30	12.04
3	16.19	16.31	16.39	16.47	16.54	16.57	16.62	14.62	18.06

Tableau 2.8: Performance de la TCQ pour une source laplacienne sans mémoire utilisant un alphabet de reproduction doublé et optimisé. Sont également donnés les RSB obtenus avec le quantificateur scalaire optimal (Lloyd Max) ainsi que la limite théorique de Shannon.

Débit (bpe)	Dimension du treillis							Quantificateur Lloyd-Max	D(R)
	4 états	8 états	16 états	32 états	64 états	128 états	256 états		
1	4.36	4.49	4.59	4.67	4.76	4.82	4.87	3.01	6.62
2	9.47	9.64	9.71	9.77	9.86	9.93	9.98	7.54	12.66
3	14.94	15.11	15.17	15.25	15.34	15.41	15.45	12.64	18.68

Les performances de la TCVQ bidimensionnel ($L = 2$) pour la source laplacienne sont données dans le tableau 2.9. La TCVQ apporte une augmentation de 0.6 dB par rapport à la TCQ.

Tableau 2.9: Rapports Signal sur Bruit obtenus par la TCVQ pour une source laplacienne sans mémoire pour un débit égal à 1 bit/échantillon. Sont également donnés les RSB obtenus avec le quantificateur vectoriel optimal ainsi que la limite théorique de Shannon.

Dimension des vecteurs	Dimension du treillis		VQ	Limite théorique
	8 états	16 états		
1	4.49	4.59	----	6.62
2	5.14	5.20	3.70	

2.8 Conclusion

La quantification codée par treillis est une technique de quantification qui permet d'augmenter la taille de dictionnaire tout en maintenant le même débit. On a exposé la méthode qui permet de faire la conception d'un codeur TCVQ. Les résultats des simulations trouvées illustre l'efficacité de cette technique pour la compression des signaux et principalement lorsqu'on augmente le nombre d'états du treillis et la dimension des vecteurs.

CHAPITRE 3
QUANTIFICATION VECTORIELLE ALGEBRIQUE EN TREILLIS

La quantification vectorielle codée par treillis (TCVQ) est assez performante étant donné qu'elle fait combiner deux puissantes techniques de compression des signaux, à savoir, la TCQ et la quantification vectorielle. Mais cette technique a des limites lorsque le dictionnaire (du treillis) requis nécessite d'avoir une grande taille. La constitution d'un dictionnaire de taille élevée est très lourde en temps de calcul. Le stockage d'un tel dictionnaire pose aussi problème. Mais l'inconvénient le plus gênant apparaît au niveau de la quantification du signal. Si le dictionnaire est trop grand, la recherche du meilleur chemin dans le codeur TCVQ en utilisant l'algorithme de Viterbi sera trop complexe pour une application en temps réel. Ces inconvénients peuvent être évités grâce à l'introduction d'un dictionnaire algébrique (*i.e.* réseau régulier de points) au sein du codeur TCVQ et qui engendre une nouvelle technique de quantification, appelée ATVQ[1]. Le cadre d'application choisi de cette approche est celui de la quantification des paramètres LSF en bande élargie vu son très bon rapport performance/complexité par rapport au cas en bande étroite. La supériorité trouvée en bande élargie est due principalement à l'ordre de prédiction élevé utilisé ainsi qu'au nombre considérable de bits alloué par le quantificateur.

[1] Quantification vectorielle algébrique en treillis ("Algebraic Trellis Vector Quantization").

Après une description de la structure de l'ATVQ, nous allons exposer dans la suite de ce chapitre la méthode de conception du dictionnaire algébrique du treillis et nous montrerons comment réaliser la recherche du plus proche voisin et l'indexage au sein du dictionnaire. Enfin, nous donnerons les résultats de simulation qui permettent de comparer l'ATVQ avec d'autres méthodes de quantification.

3.1 Structure de l'ATVQ

L'idée de base de l'ATVQ est de substituer le dictionnaire vectoriel stochastique utilisé dans la TCVQ par le dictionnaire algébrique de Xie et Adoul [52] afin de pallier la complexité de stockage et de calcul dans un dictionnaire trop important.

En s'inspirant de l'idée de Xie et Adoul, le vecteur des LSF de dimension 16 est partitionné en cinq sous-vecteurs comme suit :

$$\begin{Bmatrix} (\omega_0, \omega_4, \omega_8, \omega_{12}), (\omega_1, \omega_2, \omega_3), (\omega_5, \omega_6, \omega_7), \\ (\omega_9, \omega_{10}, \omega_{11}), (\omega_{13}, \omega_{14}, \omega_{15}) \end{Bmatrix} \quad (3.1)$$

En se référant à la propriété d'ordonnancement des paramètres LSF, il est à mentionner que les composantes du premier sous-vecteur (*i.e.* $(\omega_0, \omega_4, \omega_8, \omega_{12})$) correspondent aux frontières des autres sous-vecteurs. Par exemple, $\omega_0$et ω_4 sont les frontières du triplet ordonné $(\omega_1, \omega_2, \omega_3)$.

Chaque sous-vecteur est associé à une étape de treillis. En conséquence, il aura une recherche en treillis sur cinq étapes. Un dictionnaire stochastique partitionné en 8 ou 16 sous-alphabets est assigné à la première étape du treillis, tandis qu'on utilise pour chacune des autres étapes un dictionnaire algébrique partitionné en 16 sous-alphabets.

Les treillis utilisés dans l'ATVQ possèdent 16 ou 32 états. Deux (2) branches quittent et deux (2) rejoignent chaque nœud ou état. L'étiquetage des treillis par les sous-alphabets est fait suivant la procédure décrite dans le deuxième chapitre (§2.6.4). Les tableaux 3.1 et 3.2 illustrent l'étiquetage des treillis à 16 et à 32 états par 8 et 16 sous-alphabets.

Le dictionnaire stochastique de la première étape du treillis est d'abord conçu par l'algorithme des k-moyennes [31] en utilisant la méthode de maximum de distance [33]

pour le choix du dictionnaire initial. Une fois que cette tâche est accomplie, ce dictionnaire est ensuite partitionné en sous-alphabets par l'utilisation de l'algorithme de Wang et Moayari [47].

Tableau 3.1 : Structure d'un treillis de 16 états étiqueté par 8 et 16 sous-alphabets.

	Etat courant															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Etat précédent	0	2	4	6	8	10	12	14	3	1	7	5	11	9	15	13
	4	6	0	2	12	14	8	10	7	5	3	1	15	13	11	9
8 sous-alphabets associés	0	2	0	2	0	2	0	2	1	3	1	3	1	3	1	3
	4	6	4	6	4	6	4	6	5	7	5	7	5	7	5	7
16 sous-alphabets associés	0	4	2	6	0	4	2	6	1	5	3	7	1	5	3	7
	8	12	10	14	8	12	10	14	9	13	11	15	9	13	11	15

Tableau 3.2 : Structure d'un treillis de 32 états étiqueté par 8 et 16 sous-alphabets.

	Etat courant															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Etat précédent	0	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30
	8	10	12	14	0	2	4	6	24	26	28	30	16	18	20	22
8 sous-alphabets associés	0	2	0	2	0	2	0	2	16	18	20	22	24	26	28	30
	4	6	4	6	4	6	4	6	24	26	28	30	16	18	20	22
16 sous-alphabets associés	0	4	2	6	0	4	2	6	0	2	0	2	0	2	0	2
	8	12	10	14	8	12	10	14	4	6	4	6	4	6	4	6
	Etat courant															
	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Etat précédent	5	7	1	3	13	15	9	11	21	23	17	19	29	31	25	27
	13	15	9	11	5	7	1	3	29	31	25	27	21	23	17	19
8 sous-alphabets associés	1	3	1	3	1	3	1	3	1	3	1	3	1	3	1	3
	5	7	5	7	5	7	5	7	5	7	5	7	5	7	5	7
16 sous-alphabets associés	1	5	3	7	1	5	3	7	1	5	3	7	1	5	3	7
	9	13	11	15	9	13	11	15	9	13	11	15	9	13	11	15

Les quatre derniers sous-vecteurs ont subi certaines transformations pour les raisons suivantes :

1. Pour exploiter la corrélation existante entre deux paramètres LSF adjacentes.
2. Pour adapter la distribution conjointe des composantes des sous-vecteurs à une forme simple et régulière afin d'obtenir le gain de forme du quantificateur algébrique.
3. Pour réduire la distorsion totale de quantification.
4. Pour éviter la violation de la propriété d'ordonnancement des paramètres LSF après quantification.

En conséquence, après l'application de ces transformations sur les quatre derniers sous-vecteurs, ces derniers deviennent (x_1, x_2, x_3, x_4) et sont définis comme suit :

$$x_{11} = \frac{\omega_1 - \hat{\omega}_0}{\hat{\omega}_4 - \hat{\omega}_0} \qquad x_{12} = \frac{\omega_2 - \hat{\omega}_1}{\hat{\omega}_4 - \hat{\omega}_0} \qquad x_{13} = \frac{\omega_3 - \hat{\omega}_2}{\hat{\omega}_4 - \hat{\omega}_0} \tag{3.2}$$

$$x_{21} = \frac{\omega_5 - \hat{\omega}_4}{\hat{\omega}_8 - \hat{\omega}_4} \qquad x_{22} = \frac{\omega_6 - \hat{\omega}_5}{\hat{\omega}_8 - \hat{\omega}_4} \qquad x_{23} = \frac{\omega_7 - \hat{\omega}_6}{\hat{\omega}_8 - \hat{\omega}_4} \tag{3.3}$$

$$x_{31} = \frac{\omega_9 - \hat{\omega}_8}{\hat{\omega}_{12} - \hat{\omega}_8} \qquad x_{32} = \frac{\omega_{10} - \hat{\omega}_9}{\hat{\omega}_{12} - \hat{\omega}_8} \qquad x_{33} = \frac{\omega_{11} - \hat{\omega}_{10}}{\hat{\omega}_{12} - \hat{\omega}_8} \tag{3.4}$$

$$x_{41} = \frac{\omega_{13} - \hat{\omega}_{12}}{\mu - \hat{\omega}_{12}} \qquad x_{42} = \frac{\omega_{14} - \hat{\omega}_{13}}{\mu - \hat{\omega}_{12}} \qquad x_{43} = \frac{\omega_{15} - \hat{\omega}_{14}}{\mu - \hat{\omega}_{12}} \tag{3.5}$$

où ω_i et $\hat{\omega}_i$ sont respectivement les $i^{\text{ème}}$ paramètres des vecteurs LSF original et quantifié. La constante μ utilisé dans l'équation 3.5, au lieu de π comme dans [52], est égale à 3.0245. Elle est définie comme la valeur maximale du $16^{\text{ième}}$ LSF. Nos expériences indiquent l'efficacité de μ par rapport à π en termes de performances de quantification. Noter que pour déterminer la valeur de μ à partir de base d'apprentissage, l'histogramme du $16^{\text{ième}}$ LSF est pris en compte où les valeurs très grandes sont écartées (1% du nombre total).

Il faut noter que les composantes $x_{ij}\,(i, j = 1,2,3,4)$ des sous-vecteurs x_1, x_2, x_3 et x_4 doivent être non négatives au sens de la stabilité. Cependant il est possible que des valeurs négatives de x_{ij} résultent des expressions ci-dessus à cause de l'utilisation des paramètres quantifiés. Dans ce cas, on impose $x_{ij} = 0$.

Quand nous substituons les valeurs quantifiées dans les équations (3.2)-(3.5) par leurs valeurs originales (non quantifiées), nous pouvons tracer les distributions conjointes des composantes des sous-vecteurs x_1, x_2, x_3 et x_4 (Figure 3.1).

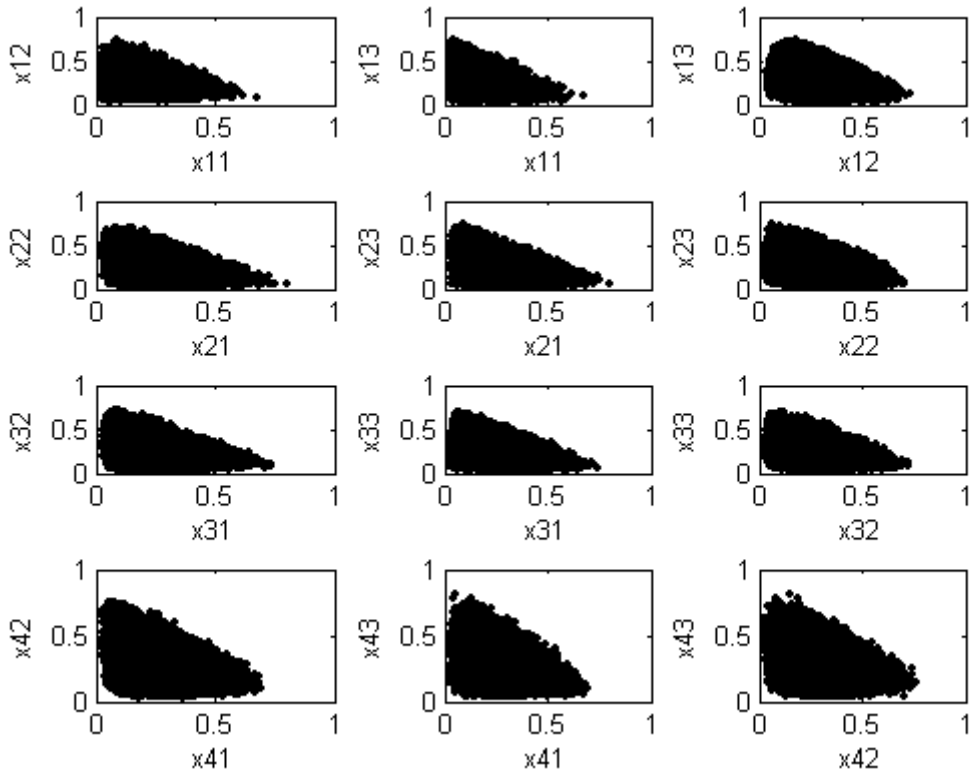


Figure 3.1 : Distributions conjointes des composantes des sous-vecteurs x_1, x_2, x_3 et x_4.

On peut voir de la figure 3.1 que la distribution conjointe des composantes de chaque sous-vecteur possède une forme de pyramide triangulaire (*i.e.* une forme régulière). Il est très intéressant de bénéficier de cette propriété géométrique en utilisant, dans le processus de quantification, un dictionnaire algébrique (*i.e.* un réseau régulier de points dans l'espace), qui a les deux caractéristiques attirantes suivantes :

1. Le dictionnaire algébrique, grâce à son structure régulière, permet d'effectuer une quantification très rapide, c'est-à-dire sans parcourir l'ensemble des valeurs disponibles pour le quantificateur (le temps de codage est fonction de la dimension du vecteur à quantifier, et non du nombre de points présents dans le réseau).

2. Les vecteurs sont générés d'une façon très simple au lieu d'être stocker dans un dictionnaire (*i.e.* il n'y a pas besoin de stocker de dictionnaire).

Nous concentrons maintenant notre attention sur la structure du dictionnaire algébrique aussi bien que sa procédure de conception.

3.2 Dictionnaire algébrique du treillis

Un réseau régulier dans $\mathbb{R}^N$ est l'ensemble des points x qui s'obtiennent par combinaison linéaire de N vecteurs de base indépendants, $z_1, z_2, \cdots, z_N$, avec des coefficients de proportionnalité entiers, $k_1, k_2, \cdots, k_N$ [53]:

$$\Lambda_{\mathrm{N}} = \{x | x = \sum_{i=1}^{N} k_i z_i\} \tag{3.6}$$

Les vecteurs de base sont choisis de telle manière que le réseau soit le plus dense possible, c'est-à-dire qu'il y ait un maximum de points dans un minimum d'espace.

Géométriquement, un réseau régulier est un arrangement régulier de points dans l'espace euclidien de dimension N.

Le réseau le plus simple, noté Z_N, est composé de tous les points à coordonnées entières de $\mathbb{R}^N$. Il est en forme de grille cartésienne.

Un autre réseau intéressant, surtout dans notre travail, est le réseau D_N. Il contient les points de Z_N dont la somme des coordonnées est paire.

Les sous-alphabets qui constituent le dictionnaire algébrique du treillis sont engendrés à base du réseau régulier D_3 modifié, noté D_3^M, qui est défini comme suit :

$$D_3^M = \{c \in \mathbb{Z}^3 |\ c_i \text{ est un entier pair }\} \tag{3.7}$$

Dans les applications pratiques dans la quantification algébrique, les réseaux réguliers sont généralement tronqués afin d'obtenir un dictionnaire de taille finie. Pour une distorsion minimale, il est nécessaire de déterminer la forme de troncature en tenant compte de la distribution la plus probable des vecteurs source [54]. Le nombre de points conservés du réseau est aussi fonction du débit alloué au quantificateur. Dans notre cas, le

réseau régulier D_3^M est tronqué selon la forme de pyramide triangulaire en utilisant les contraintes exprimées comme suit :

$$c_i \geq 0,\ i = 1,2,3 \qquad \text{et} \qquad \sum_{i=1}^{3} c_i \leq S \tag{3.8}$$

où c_i est la $i^{\text{ème}}$ composante d'un vecteur type dans le dictionnaire algébrique et S est un entier pair positif fixe qui détermine la taille du dictionnaire. Le tableau 3.3 présente pour chaque nombre de bits alloué, la valeur correspondante de S ainsi que la taille du dictionnaire algébrique.

Tableau 3.3 : Taille du dictionnaire algébrique pour différentes valeurs de S.

Nombre de bits	$\boldsymbol{S}$	Taille du dictionnaire
4	4	10
5	6	20
6	10	56
7	14	120
8	18	220
9	24	455
10	32	969

Comme nous avons mentionné auparavant, le dictionnaire algébrique du treillis est constitué de 16 sous-alphabets. En se référant aux tableaux 3.1 et 3.2, et suivant les règles *ad hoc* décrites dans [39] sur la construction des sous-alphabets et sur l'étiquetage des branches de treillis, les sous-alphabets $S_i, i = 0,1,\cdots,15$ sont exprimés comme suit :

$$S_i = \{D_3^M + v_i\},\ \ i = 0,1,\cdots,15 \tag{3.9}$$

où $\{D_3^M + v_i\}$ est une notation pour l'ensemble contenant les vecteurs du réseau régulier D_3^M translatés par le vecteur v_i. Le tableau 3.4 montre le vecteur correspondant v_i pour chaque S_i .

Noter qu'à partir du tableau 3.4, nous pouvons voir qu'il y a quelques sous-alphabets identiques, par exemple, S_0 et S_{12}.

Dans le but d'avoir de meilleures performances, différents facteurs d'échelle $\alpha_i, i = 0,1,\cdots,7$ sont assignés aux 8 sous-alphabets non-identiques. Ces facteurs d'échelle doivent être adaptés à la distribution conjointe des composantes des sous-vecteurs. Afin de tirer profit de la première caractéristique du dictionnaire algébrique citée précédemment, les composantes du vecteur à quantifier sont multipliées par ces facteurs α_i plutôt que les composantes des vecteurs type du réseau régulier. Mais avant cette étape de multiplication par un facteur d'échelle, on décale le vecteur d'entrée dans le sens négatif par un vecteur d'offset (de décalage) $\beta = (\beta_x, \beta_y, \beta_z)$. Cette approche améliore davantage les performances.

Tableau 3.4 : Liste des vecteurs de translation correspondants aux sous-alphabets.

Sous-alphabet	Vecteur de translation
S_0	(0,0,0)
S_1	(0,0,1)
S_2	(0,1,1)
S_3	(0,1,0)
S_4	(1,1,1)
S_5	(1,1,0)
S_6	(1,0,0)
S_7	(1,0,1)
S_8	(1,1,1)
S_9	(1,1,0)
S_{10}	(1,0,0)
S_{11}	(1,0,1)
S_{12}	(0,0,0)
S_{13}	(0,0,1)
S_{14}	(0,1,1)
S_{15}	(0,1,0)

Les valeurs optimales des facteurs d'échelle α_i sont obtenues expérimentalement sur la base d'apprentissage. Pour chaque sous-alphabet et à un débit donné, nous testons plusieurs facteurs d'échelle, et nous évaluons pour chacun l'erreur quadratique moyenne

après le codage de la séquence d'apprentissage en utilisant le sous-alphabet algébrique, et ensuite nous tabulons la liste des meilleurs facteurs d'échelle.

Pour les sous-alphabets identiques, notés S_i et S_j avec $i < j$, L'opération de multiplication par facteurs d'échelle est donnée par la méthode suivante : nous assignons α_i à S_i et $(\gamma\alpha_i)$ à S_j , où γ est un facteur scalaire obtenu par apprentissage avec la même manière que les α_i.

Les composantes du vecteur d'offset (de décalage) $\beta = (\beta_x, \beta_y, \beta_z)$ correspondent aux composantes de l'origine de la pyramide triangulaire, et qui peuvent être déterminées d'une façon empirique en éliminant une portion de la forme pyramidale de la distribution conjointe du sous-vecteur x_i et exactement de la partie inférieure de la pyramide dans les trois directions de l'espace. Une portion égale à 5% a été trouvée satisfaisante.

Il est à noter que nous suivons la procédure expliquée ci-dessus pour concevoir les quatre dictionnaires algébriques correspondants aux quatre dernières étapes du treillis.

3.3 Détermination du plus proche voisin et indexage

L'étape suivante après la conception du dictionnaire est l'encodage, et qui se compose de deux parties importantes. La première est la recherche du plus proche voisin, qui a comme objectif de trouver le meilleur vecteur type pour un vecteur à quantifier donné. La seconde étape est l'indexage (ou indexation) qui sert à déterminer un index pour un vecteur type donné.

Pour un vecteur d'entrée (à quantifier) donné, la recherche optimale de son plus proche voisin (*i.e.* recherche exhaustive) est de comparer le vecteur d'entrée à tous les vecteurs types du dictionnaire, et ensuite sélectionner le vecteur type qui minimise un certain critère de distorsion. Ce type de recherche limite certainement la taille du dictionnaire dans les applications en temps réel. Pour pouvoir profiter de dictionnaires plus grands, on est obligé d'utiliser un algorithme de codage accéléré.

L'élément déterminant pour l'introduction de la quantification algébrique, vient de la mise en œuvre possible d'algorithmes de quantification rapide. Cette fois l'encodage n'est plus effectué par recherche exhaustive au sein d'un dictionnaire du vecteur type le plus

proche, mais en appliquant directement les calculs sur les composantes du vecteur à encoder. Conway et Sloane ont développé pour certaines classes de réseaux réguliers des algorithmes de quantification rapide [55]. Le réseau régulier D_3, utilisé dans notre travail, appartient à ces classes. La recherche du plus proche voisin dans le réseau régulier D_3 est effectuée en deux étapes comme suit :

Etape 1 : Trouver le plus proche voisin vecteur type y du vecteur d'entrée x dans le réseau $\mathbb{Z}_3$. Cela est réalisé en arrondissant individuellement chaque composante de x au nombre entier le plus proche.

Etape 2 : Si la somme de composantes de y est impaire, modifier y en arrondissant de la « mauvaise façon » la composante de x qui a une plus grande distorsion de quantification que les autres.

Par exemple, un vecteur d'entrée $x = [4.1, 0.4, -2.8]$ est représenté par $y = [4,\ 1,\ -3]$ au lieu de $y' = [4,\ 0,\ -3]$.

Dans notre travail, la recherche du plus proche voisin dans chaque sous-alphabet est effectuée selon son vecteur de translation correspondant par une manière très simple. Soit $a = (a_1, a_2, a_3)$ le vecteur de la source à quantifier, et soit $v_i = (v_{i1}, v_{i2}, v_{i3})$ le vecteur de translation associé au sous-alphabet S_i. Arrondir la composante $a_j, j = 1,2,3,$ du vecteur d'entrée au nombre entier pair le plus proche si $v_{ij} = 0$, et au nombre entier impair le plus proche si $v_{ij} = 1$.

Dans cette étape d'encodage, nous devons concevoir un algorithme efficace pour résoudre le problème des vecteurs d'entrée qui tombent à l'extérieur du réseau régulier tronqué. Dans la littérature, ces vecteurs d'entrée sont appelés vecteurs de surcharge[1]. Afin de résoudre ce problème, nous avons développé un algorithme qui est donné en détail comme suit :

[1] En anglais : overload vectors.

Soit $a = (a_1, a_2, a_3)$ un vecteur de surcharge et S un nombre entier pair (défini dans §2.2). Dénotant la somme des composantes de chaque vecteur de translation v_i (correspondant au sous-alphabet S_i) par m_i.

Etape 1 : Trouver le plus proche voisin vecteur type, noté $c = (c_1, c_2, c_3)$, du vecteur de surcharge a dans le sous-alphabet S_i.

Etape 2 : Si ce vecteur type tombe à l'extérieur du réseau régulier tronqué $\{i.e.\ \sum_{j=1}^{3} c_j > (S + m_i)\}$, alors calculer la différence entre c et a. Ce vecteur de différence d est donné par : $d_i = c_i - a_i, i = 1,2,3$.

Etape 3 : Créer un vecteur d'index, $id = (id_1, id_2, id_3)$, pour le vecteur d, *i.e.* un vecteur de pointeurs qui indique quel $d_i, i = 1,2,3$ est classé en premier lieu suivant l'ordre décroissant, qui est en second lieu et ainsi de suite.

Etape 4 : Fixer $c(id_1) = c(id_1) - 2$. Si $\sum_{j=1}^{3} c_j \leq (S + m_i)$, alors stop. Sinon continue.

Etape 5 : Fixer $c(id_2) = c(id_2) - 2$. Si $\sum_{j=1}^{3} c_j \leq (S + m_i)$, alors stop. Sinon continue.

Etape 6 : Fixer $c(id_3) = c(id_3) - 2$. Si $\sum_{j=1}^{3} c_j \leq (S + m_i)$, alors stop. Sinon aller à l'étape 4.

Pour vérification et évaluation en même temps, des simulations ont été effectuées sur cet algorithme ainsi que sur l'algorithme de recherche exhaustive. Les résultats trouvés permettent de constater que les deux algorithmes donnent, pour un vecteur d'entrée de surcharge, le même plus proche voisin vecteur type dans le réseau régulier tronqué.

L'ultime étape dans l'utilisation des dictionnaires en réseaux réguliers de points est l'indexage, qui consiste à associer à chacun des points dans le réseau régulier tronqué un index (ou indice) qui sera codé et transmis au décodeur. Si la quantification vectorielle algébrique apporte une solution à la coûteuse recherche exhaustive au sein du dictionnaire, la difficulté s'est déplacée vers l'indexage qui est devenu une étape complexe. En effet pour la quantification vectorielle statistique, les indices sont obtenus immédiatement car ils

sont stockés avec les représentants (ou vecteurs types) dans un tableau, et ce dernier a été parcouru lors de la recherche du plus proche voisin. Avec la quantification algébrique le représentant est calculé directement à partir des composantes à coder, et il faut ensuite calculer l'index correspondant (il n'est pas envisageable de parcourir une liste de transcodage) [56].

En se basant sur la méthode d'indexage en [52], nous avons proposé une version modifiée de cette méthode. Ses détails sont résumés par la description algorithmique suivante de l'algorithme de codage (produit un index pour chaque vecteur type dans le réseau régulier tronqué) ainsi que l'algorithme de décodage (assigne à chaque index le vecteur type correspondant dans le réseau régulier tronqué).

ALGORITHME DE CODAGE

Soit $c = (c_1, c_2, c_3)$ un vecteur type du réseau régulier tronqué, et soit $v_i = (v_{i1}, v_{i2}, v_{i3})$ le vecteur de translation associé au sous-alphabet S_i.

Etape 1 : Calculer

$t_j = \left(c_j - v_{ij}\right)/2,$ pour $j = 1,2,3.$

$s_2 = t_1 + t_2.$

$s_3 = s_2 + t_3.$

$k_2 = \frac{1}{2} s_2(s_2 + 1).$

$k_3 = \frac{1}{6} s_3(s_3 + 1)(s_3 + 2).$

Etape 2 : L'index à transmettre est : $k = k_3 + k_2 + t_1.$

Noter que la valeur du k_3 représente le nombre des vecteurs types dans la forme pyramidale (trois dimensions) dont la somme de leurs composantes est inférieure à $(2s_3)$. Les valeurs possibles de k_3 sont : $\{0,1,4,10,20,35,56,84,120,165,\cdots\}$ qui correspondent aux valeurs suivantes de s_3 : $\{0,1,2,3,4,5,6,7,8,9,\cdots\}$. La valeur de k_2 a la même signification que k_3 mais dans la forme triangulaire (deux dimensions). Les valeurs possibles de k_2 sont : $\{0,1,3,6,10,15,21,28,36,45,\cdots\}$ qui correspondent aux valeurs suivantes de s_2 : $\{0,1,2,3,4,5,6,7,8,9,\cdots\}$.

ALGORITHME DE DECODAGE

Soit $v_i = (v_{i1}, v_{i2}, v_{i3})$ le vecteur de translation associé au sous-alphabet S_i. En donnant un index k.

Etape 1 : Extraire la valeur de k_3 à partir de l'index k où k_3 est la plus grande valeur appartenant à la liste de ses valeurs possibles, et en même temps inférieure ou égale à k. Dénotant la position de la valeur trouvée de k_3 dans la liste de ses valeurs possibles par : l, où la première valeur est considérée à la position zéro (0).

Etape 2 : Extraire la valeur de k_2 à partir de la différence $(k - k_3)$ où k_2 est la plus grande valeur appartenant à la liste de ses valeurs possibles, et en même temps inférieure ou égale à $(k - k_3)$. Dénotant la position de la valeur trouvée de k_2 dans la liste de ses valeurs possibles par : m.

Etape 3 : Calculer $t_1 = (k - k_3 - k_2)$.

Etape 4 : Le vecteur type c correspondant à l'index k est donné par ses composantes comme suit :

$c_1 = v_{i1} + 2t_1$;

$c_2 = v_{i2} + 2(m - t_1)$;

$c_3 = v_{i3} + 2(l - m)$;

3.4 Résultats de simulation

Dans cette étude, la base de données de parole utilisée se compose de 233007 trames de 20 ms prise de la base données TIMIT (ensemble de phrases prononcées dans la langue anglaise par plusieurs locuteurs, hommes et femmes), dont la fréquence d'échantillonnage est de 16000 Hz [57]. Une partie de la base de données (121478 trames prononcées par 80 locuteurs) est choisie comme base d'apprentissage pour la conception des dictionnaires, et la partie restante, de 111529 trames prononcées par 73 locuteurs est réservée pour l'évaluation des performances. Tous les signaux de parole sont d'abord filtrés en utilisant

le masque P.341 [58,59]. Le niveau du signal de parole est ajusté à -26 dBov[1] [59]. Nous avons utilisé le prétraitement et l'analyse LPC du codeur de parole AMR-WB (version en point flottante)[60] pour produire les coefficients de prédiction d'ordre 16, et qui sont ensuite convertis en paramètres LSF en utilisant la méthode des polynômes de Chebyshev de Kabal et Ramachandran [6].

Le plan d'allocation de bits utilisée dans nos expériences au sein de la structure ATVQ, qui donne les meilleures performances est donné dans le tableau 3.5. Le nombre des sous-alphabets employé dans chaque étage du treillis pour différents débits dans la région de 43 à 47 bits/trame est résumé dans le tableau 3.6. Le tableau 3.7 montre les performances pour différents débits en terme de distorsion spectrale (SD) du quantificateur ATVQ ainsi que son espace mémoire occupé par ses données (*i.e.* les éléments du dictionnaire stochastique du premier étage du treillis).

Tableau 3.5 : Plan d'allocation de bits pour chaque étape du treillis de la quantification vectorielle algébrique en treillis en fonction du débit.

	Débit (bits/trame)				
	43	44	45	46	47
Etat initial	4	5	5	5	5
Chemin	5	5	5	5	5
1er étage	7	7	7	8	8
2ème étage	7	7	7	7	7
3ème étage	7	7	7	7	8
4ème étage	7	7	7	7	7
5ème étage	6	6	7	7	7

[1] Le niveau de puissance en décibels est défini par rapport au niveau de référence de puissance qui est égal à $P_0 = 1.0$: $L = 10log_{10}(P/P_0)$ (dBov)

Le niveau de puissance $P = 1.0$ est donc 0 dBov (où les caractères « ov » indique « *the digital overload signal level* »), qui est choisi pour être le niveau de référence.

On peut voir dans ce tableau que l'ATVQ à 47 bits/trame permet d'obtenir une qualité transparente de la quantification des paramètres LSF, c'est-à-dire une distorsion spectrale moyenne d'environ 1 dB, moins de 2% de trames « outliers » dans la région de 2 à 4 dB et aucune trame « outlier » ayant une distorsion spectrale supérieure à 4 dB.

Tableau 3.6 : Nombre de sous-alphabets dans chaque étape du treillis de la quantification vectorielle algébrique en treillis.

	Débit (bits/trame)				
	43	44	45	46	47
Nombre de sous-alphabets dans le 1er étage	16	16	16	8	8
Nombre de sous-alphabets dans les étages (2,3,4,5)	16	16	16	16	16

Tableau 3.7 : Performances (SD) et exigence en mémoire de stockage de la quantification vectorielle algébrique en treillis pour 43-47 bits/trame.

Débit (bits/trame)	SD (dB)	Outliers (%)		Espace mémoire occupé (flottants)
		2-4 dB	$>$ 4 dB	
43	1.16	0.35	0.00	8192
44	1.12	0.28	0.00	8192
45	1.08	0.24	0.00	8192
46	1.04	0.21	0.00	8192
47	1.01	0.19	0.00	8192

On compare les performances de la quantification vectorielle algébrique par treillis (ATVQ) avec celles de deux autres méthodes de quantification des paramètres LSF en bande large. Les deux autres méthodes de quantification qui entrent dans la comparaison sont :

1. La SVQ[1] traditionnelle qui est utilisée comme référence de comparaison de performances, comme dans les travaux présentés dans [61,62].

[1] Quantification vectorielle fendue ("Split Vector Quantization").

2. La TCVQ proposée par Fischer *et al.* et qui est présentée dans [39].

Pour ces deux quantificateurs SVQ et TCVQ, le vecteur LSF de dimension 16 est partitionné en cinq sous-vecteurs de dimension (3,3,3,3,4).

Afin de faire une comparaison correcte des trois quantificateurs, ils devraient être comparés en utilisant la même base de données de parole pour la principale raison que différentes bases de données peuvent mener à des performances objectives radicalement différentes pour le même type de quantificateur [63].

Les performances de la SVQ et son espace mémoire requis sont résumés dans le tableau 3.8 ainsi que les allocations optimales de bits qui donnent les meilleures performances. Les tableaux 3.9 et 3.10 montrent respectivement le plan d'allocation de bits et les performances de la technique de quantification TCVQ, avec 32 états et 5 étages.

Tableau 3.8 : Performances (SD) et exigence en mémoire de stockage de la quantification vectorielle fendue (SVQ) en (3,3,3,3,4) pour 43-47 bits/trame.

Débit (bits/trame)	SD (dB)	Outliers (%)		Espace mémoire occupé (flottants)
		2-4 dB	> 4 dB	
43 $_{(8,9,9,8,9)}$	1.16	1.06	0.00	6656
44 $_{(8,9,9,9,9)}$	1.12	0.75	0.00	7424
45 $_{(9,9,9,9,9)}$	1.10	0.66	0.00	8192
46 $_{(9,9,9,9,10)}$	1.05	0.40	0.00	10240
47 $_{(9,9,10,9,10)}$	1.02	0.28	0.00	11776

Tableau 3.9 : Plan d'allocation de bits pour chaque étape du treillis de la quantification vectorielle codée par treillis (TCVQ) en fonction du débit.

	Débit (bits/trame)				
	43	44	45	46	47
Etat initial	5	5	5	5	5
Chemin	5	5	5	5	5
1^{er} étage	6	6	7	7	7
$2^{ème}$ étage	7	7	7	7	7
$3^{ème}$ étage	7	7	7	7	8
$4^{ème}$ étage	6	7	7	7	7
$5^{ème}$ étage	7	7	7	8	8

Tableau 3.10 : Performances (SD) et exigence en mémoire de stockage de la quantification vectorielle codée par treillis (TCVQ) pour 43-47 bits/trame.

Débit (bits/trame)	SD (dB)	Outliers (%)		Espace mémoire occupé (flottants)
		2-4 dB	$>$ 4 dB	
43	1.16	0.58	0.00	13312
44	1.12	0.37	0.00	14848
45	1.09	0.30	0.00	16384
46	1.04	0.16	0.00	20480
47	1.00	0.09	0.00	23552

Les résultats obtenus montrent que la technique ATVQ permet d'obtenir pratiquement les mêmes performances que la SVQ et la TCVQ. Au débit de 47 bits/trame, l'ATVQ fournit une qualité transparente similaire à la SVQ et à la TCV Q, mais avec une réduction significative et simultanée de la complexité de calcul et de l'exigence de mémoire de stockage. La réduction de la complexité de calcul est due à la recherche du plus proche voisin car des arrondis simples sont utilisés dans l'ATVQ tandis que dans la SVQ et la TCVQ, la recherche exhaustive avec la distance euclidienne pondérée IHM (voir § 1.4.2.2) comme critère de mesure de distorsion est utilisée. Le tableau 3.11 donne la complexité du calcul de la recherche du plus proche voisin, en terme du nombre des opérations arithmétiques, des trois techniques de quantification ATVQ, SVQ et TCVQ au

débit de 47 bits/trame. Les histogrammes de distorsion spectrale (SD) à ce débit sont également fournis pour les trois techniques dans la figure 3.2. On peut voir à partir de cette figure que l'ATVQ réduit légèrement la quantité des trames « outliers », en comparaison avec la SVQ.

Tableau 3.11 : Comparaison de la complexité de calcul de la recherche du plus proche voisin pour les trois techniques de quantification SVQ, TCVQ et ATVQ à 47 bits/trame.

Opération	SVQ	TCVQ	ATVQ
Multiplication	11776	23552	8960
Addition	19968	40064	15744
Comparaison	3579	7314	2226
Arrondi			768

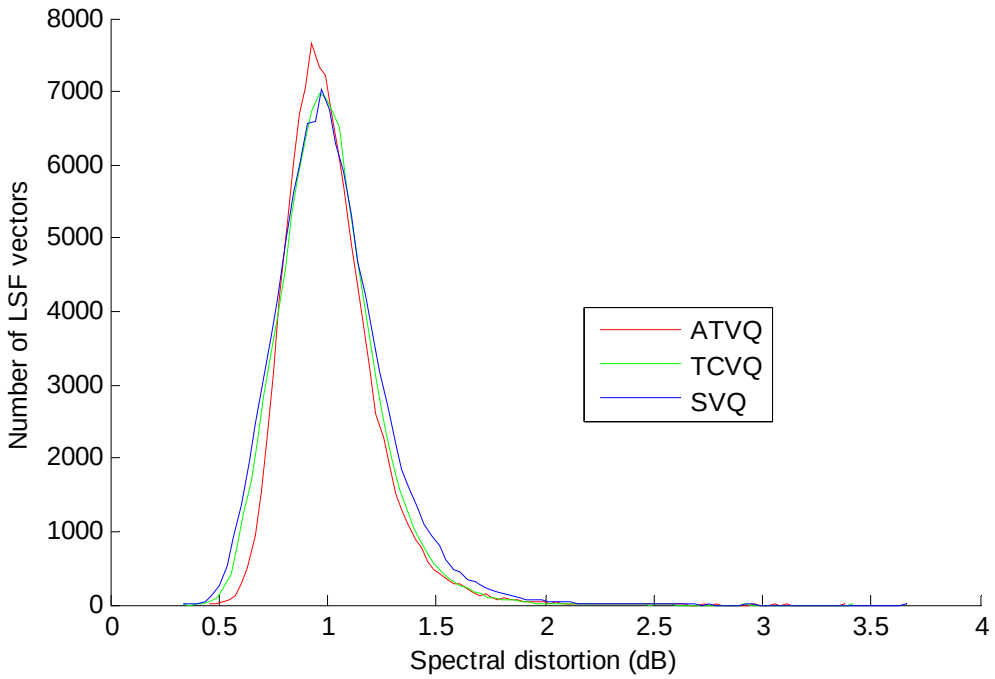


Figure 3.2 : Histogrammes des distorsions spectrales (SD) correspondants aux techniques de quantification SVQ, TCVQ, et ATVQ à 47 bits/trame.

Il est à noter que la distance euclidienne avec la fonction de pondération IHM a été utilisée dans l'ATVQ pour la sélection du chemin, dans la SVQ pour la sélection du vecteur type, et dans la TCVQ pour les deux : sélection du chemin et du vecteur type.

Finalement, le tableau 3.12 donne les meilleurs facteurs d'échelle ainsi que les composantes du vecteur de décalage (d'offset) utilisés par l'ATVQ dans le cas de 47 bits/trame. On n'omettra pas de signaler que ces facteurs fixes et les paramètres du treillis à 32 états doivent être stockés mais dans un espace mémoire très réduit (n'exige pas beaucoup de mémoire).

Tableau 3.12 : Facteurs d'échelle et composantes du vecteur de décalage (d'offset) pour le quantificateur vectoriel algébrique en treillis à 47 bits/trame.

	Etage du treillis			
	$2^{ème}$ étage	$3^{ème}$ étage	$4^{ème}$ étage	$5^{ème}$ étage
α_0	21.3	26.6	22.4	21.5
α_1	22.7	27.8	23.9	23.0
α_2	24.2	29.3	25.3	24.3
α_3	22.8	27.9	23.9	23.0
α_4	25.7	30.9	27.0	26.0
α_5	24.3	29.4	25.5	24.6
α_6	22.8	27.9	24.0	23.0
α_7	24.2	29.4	25.4	24.3
γ	0.93	0.94	0.92	0.92
β_x	0.04	0.08	0.08	0.09
β_y	0.10	0.07	0.08	0.10
β_z	0.09	0.06	0.08	0.10

3.5 Conclusion

Dans ce chapitre, nous avons présenté la nouvelle technique de quantification des paramètres LSF en large bande appelée quantification vectorielle algébrique en treillis (ATVQ) développée lors de la thèse. ATVQ est le résultat de l'introduction des

dictionnaires algébriques au sein de la quantification vectorielle codée par treillis (TCVQ). Tout d'abord, nous avons décrit les étapes de construction de l'ATVQ, à savoir : conception des dictionnaires du treillis, recherche du plus proche voisin, quantification des vecteurs de surcharge et indexage. La seconde étape a été d'évaluer ses performances et de les comparer à celles de la SVQ et la TCVQ. Les résultats des tests démontrent que l'ATVQ atteint la qualité transparente à 47 bits/trame et donne pratiquement les mêmes performances que la SVQ et la TCVQ, mais sur le plan de la complexité, l'ATVQ a deux avantages par rapport à la SVQ et TCVQ :

- L'espace mémoire pour le stockage des dictionnaires est considérablement économisé,
- La recherche du plus proche voisin s'effectue par des opérations simples, telles que l'arrondi, l'addition et la comparaison, et le calcul de la norme quadratique et la recherche exhaustive ne sont pas requis.

On n'omettra pas de mentionner que la méthode proposée fournit une réduction de la complexité (calcul et stockage) par rapport à la SVQ pour les débits binaires supérieurs à 45 bits/trame et par rapport à la TCVQ pour tous les débits binaires examinés.

CONCLUSION

Une nouvelle technique de quantification vectorielle, basée sur les treillis et les réseaux réguliers de points, a été présentée dans cette thèse. Cette technique efficace qui a permis de réduire la charge de calcul ainsi que la complexité de stockage a été appliquée à la quantification des paramètres LSF du signal de parole en bande large.

Après un rappel sur la quantification scalaire et vectorielle, nous avons étudié la quantification codée par treillis et sa généralisation vectorielle. Les performances trouvées montrent son efficacité en compression des signaux surtout lorsqu'on augmente le nombre d'états du treillis et la dimension des vecteurs mais au détriment de la complexité (en stockage et en calcul).

Cependant, en faisant appel à un dictionnaire du treillis algébrique à la place d'un dictionnaire statistique, on aura la possibilité de réduire cette complexité de façon importante tout en gardant la même qualité. C'est le principe de base de cette nouvelle technique, appelée ATVQ. Nous avons développé les algorithmes de construction de l'ATVQ, à savoir : conception des dictionnaires du treillis en tenant compte des caractéristiques des vecteurs à quantifier, recherche du plus proche voisin, quantification des vecteurs de surcharge et indexage.

Puisque l'objectif de notre travail concerne l'application de cette nouvelle technique au codage de la parole, et particulièrement à la quantification des paramètres LSF, nous

avons été amenés à déterminer, dans un premier temps, les coefficients de prédiction linéaire LPC d'un signal de parole en bande large puis d'effectuer l'extraction des paramètres LSF à partir de ces coefficients.

En assurant une qualité transparente de la quantification des paramètres LSF d'un signal de parole en bande élargie à 47 bits/trame, cette nouvelle technique a réduit considérablement la charge de calcul et l'espace mémoire requis par le stockage du quantificateur par rapport à la quantification vectorielle fendue (SVQ) ainsi que la quantification vectorielle codée par treillis (TCVQ).

Le travail présenté dans cette thèse confirme que la combinaison entre la quantification par treillis et la quantification algébrique ouvre des voies vers des techniques de quantification puissantes et prometteuses dans le sens ou elles améliorent la complexité et/ou la qualité.

Comme perspectives de ce travail, nous pouvons envisager l'application de cette technique pour d'autres signaux que le signal de parole, et/ou de développer d'autres techniques en se basant toujours sur les treillis et les réseaux réguliers de points.

REFERENCES

1. Jayant, N. S. and Noll, P., "Digital Coding of Waveforms" Englewood Cliffs, NJ: Prentice-Hall, (1984).

2. Salami, R., Laflamme, C., Adoul, J-P., Kataoka, A., Hayashi, S., Moriya, T., Lamblin, C., Massaloux, D., Proust, S., Kroon, P. and Shoham Y., "Design and description of CS-ACELP: A toll quality 8 kb/s speech coder", IEEE Trans. Speech and Audio Processing, vol. 6, (Mar. 1998), 116–130.

3. Levinson, N., "The Wiener RMS (root mean square) error criterion in filter design and prediction", J. math. Phys., (1947).

4. Itakura, F. and Saïto, S., "On the optimum quantization of feature parameters in PARCOR speech synthesis", Proc. Conf. Speech Communication on Processing, volume L-4, USA, (1972), 434–437.

5. Soong, F. K., and Juang, B. H., "Line spectrum pair (LSP) and speech data compression", Proc. IEEE Int. Conf. on Acoust. Speech signal Processing, (Mar. 1984), 1.10.1–1.10.4.

6. Kabal, P. and Ramachandran, R., "The computation of line spectral frequencies using Chebyshev polynomials", IEEE Trans. on Acoustics, Speech, and Signal Processing, vol. 34, (Dec. 1986), 1419–1426.

7. Adoul, J.-P. and Lefebvre, R., “Wideband speech coding”, in Speech Coding and Synthesis, W.B. Kleijn and K.K. Paliwal, Ed. Amsterdam: Elsevier, (1995), 289–309.

8. Paulus, J.W. and Schnitzler, J., “16 kbit/s wideband speech coding based on unequal subbands”, Proc. IEEE Int. Conf. on Acoust. Speech signal Processing, (1996), 255–258.

9. ITU-T, Recommendation G.721, “32 kbit/s adaptive differential pulse code modulation (ADPCM)”, ITU, Geneva, (Oct. 1988).

10. ITU-T, Recommendation G.726, “40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM)”, ITU, Geneva, (Dec. 1990).

11. Maitre, X., “7 kHz Audio Coding Within 64 kbit/s” , IEEE Journal on Selected Areas on Communications, vol. 6, no. 2, (Feb. 1988), 283–298.

12. Tremain, T. E., “The government standard linear predictive coding algorithm: LPC-10”, Speech Technologies, (Apr. 1982), 40–49.

13. Supplee, L. M., Cohn, R. P. and Collura, J. S., “MELP: the new standard at 2400bps”, Proc. IEEE Int. Conf. on Acoust. Speech signal Processing, (Apr. 1997), 1591–1594.

14. Schroeder, M. R. and Atal, B., “Code excited linear prediction (CELP): High quality speech at very low bit rates”, Proc. IEEE Int. Conf. on Acoust. Speech Signal Processing, (Apr. 1985), 937–940.

15. Federal Standard 1016, “Telecommunications: Analog to digital conversion of radio voice by 4800 bit/second code excited linear prediction (CELP)”, Tech. Rep., National communication System-Office of Technology and Standards, (Feb. 1991).

16. Gerson, I. and Jasiuk, M., “Vector sum excited linear prediction (VSELP) speech coding at 8 kbits/s”, Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, (Apr. 1990), 461–464.

17. Bessette, B. *et al.*, “The adaptive multirate wideband speech codec (AMR-WB)”, IEEE Trans. on Speech Audio Processing, vol. 10, no. 8, (Nov. 2002), 620–636,.

18. Laflamme, C., Adoul, J-P., Su, H.Y. and Morissette, S., “On reducing computational complexity of codebook search in CELP Coder through the use of algebraic codes”, Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing, (1990), 177–180.

19. ITU-T, Recommendation G.722.2, “Wideband coding of speech at around 16 kbit/s using adaptive multi-rate wideband (AMR-WB)”, ITU, Geneva, (Jul. 2003).

20. Gray, R. M., Buzo, A., Gray, A. H., Matsuyama, Y., “Distorsion measures for speech processing”, IEEE Trans. Acoust., Speech, Signal Proc., vol. ASSP-28, (Aug. 1980), 367–376.

21. Gray, A. H., Markel, J. D., “Distance measures for speech processing”, IEEE Trans. Acoust., Speech, Signal Proc., vol. ASSP-24, (Oct. 1976), 380–391.

22. Paliwal, K. K. and Atal, B. S., “Efficient vector quantization of LPC parameters at 24 bits/frame”, IEEE Trans. Speech and Audio Processing, vol. 1, (Jan. 1993), 3–14.

23. Hedelin, P., “Single stage spectral quantization at 20 bits”, Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing, vol. 1, Adelaide, Australia, (1994), 525–528.

24. Gardner, W. R. and Rao, B. D., “Theoretical analysis of the high-rate vector quantization of LPC parameters”, IEEE Trans. Speech Audio Processing, vol. 3, (Sept. 1995), 367–381.

25. Guibé, G., How, H. T., and Hanzo, L., “Speech spectral quantizers for wideband speech coding”, European Transactions on Telecommunications, 12(6), (2001), 535–545.

26. Laroia, R., Phamdo, N., and Farvardin, N., “Robust and efficient quantization of speech LSP parameters using structured vector quantizers”, Proc. IEEE Int. Conf. on Acoust. Speech signal Processing, (May 1991), 641–644.

27. Pan, J. and Fischer, T. R., “Vector quantization-lattice quantization of speech LPC coefficients”, Proc. IEEE Int. Conf. on Acoust. Speech signal Processing, Adelaide, Australia, (Apr. 1994), I.513–I.516.

28. Lloyd, S. P., “Least squares quantization in PCM”, IEEE Trans. Inform. Theory, vol. IT-28, (Mar. 1982), 129–136.

29. Max, J., “Quantizing for minimum distortion”, IRE Trans. Inform. Theory, vol. 6, (Mar. 1960), 7–12.

30. Gersho, A. and Gray, R. M., “Vector Quantization and Signal Compression”, Kluwer Academic Publishers, (1992).

31. Linde, Y., Buzo, A. and Gray, R. M., “An algorithm for vector quantizer design”, IEEE. Trans. Commun., vol. COM-28, (Jan. 1980), 84–95.

32. Hay, K., “Etude des techniques de codage audio à débit réduit et à faible délai”, Thèse de Doctorat, Université de Rennes 1, (1997).

33. Katsavounidis, I., Kuo, C. J. and Zhang, Z., “A new initialization technique for generalized Lloyd iteration”, IEEE Signal Processing Letters, vol. 1, (Oct. 1994), 144–146.

34. Finamore, W. A. and Pearlman, W. A., “Optimal encoding of discrete-time continuous amplitude memoryless sources with finite output alphabets”, IEEE Trans. Inform. Theory, vol. IT-26, (Mar. 1980), 144–155.

35. Pearlman, W. A., “Sliding-block and random source coding with constrained size reproduction alphabets”, IEEE Trans. Commun., vol. COM-30, (Aug. 1982), 1859–1867.

36. Pearlman, W. A. and Chekima, A., “Source coding bounds using quantizer reproduction levels”, IEEE Trans. Inform. Theory, vol. IT-30, (May 1984), 559–567.

37. G. Ungerboeck, “Channel coding with multilevel/phase signals”, IEEE Trans. Inform. Theory, vol. IT-28, (Jan. 1982), 55–67.

38. Marcellin, M. W. and. Fischer, T. R, “Trellis coded quantization of memoryless and Gauss-Markov sources”, IEEE Trans. Commun., vol. 38, (Jan. 1990), 82–93.

39. Fischer, T. R., Marcellin, M. W., and Min Wang, “Trellis-coded vector quantization”, IEEE Trans. Inform. Theory, vol. IT-37, (Nov. 1991), 1551–1566.

40. Blahut, R. E., “Computation of channel capacity and rate-distortion functions”, IEEE Trans. Inform. Theory, vol. IT-18, (Jul. 1972), 460–473.

41. Turgeon, G., “La quantification vectorielle en treillis: principe, conception et performance”, Mémoire de maîtrise, Université de Sherbrooke, (1992).

42. Berkani, D., “Design d'un quantificateur vectoriel à deux dimensions: le Spiral Quantizer”, Thèse de Doctorat, Ecole Nationale Polytechnique, (1991).

43. Berkani, D., Turgeon, G., Chekima, A. and Derras B., “Utilisation de l'algorithme LBG et du treillis en quantification vectorielle”, Journal of Technology, n° 8, (1992), 35–52.

44. Forney, G. D., “The Viterbi algorithm”, Proc. IEEE (Invited Paper), vol. 61, (Mar. 1973), 268–278.

45. Ungerboeck, G., “Trellis-coded modulation with redundant signal sets–Part II: State of the art”, IEEE Commun. Mag., vol. 25, (Feb. 1987), 12–21.

46. Popescu, A., “Codage de parole CELP à excitation vectorielle codée par treillis”, Thèse de Doctorat, ENST, (1995).

47. Wang, H. S., Moayeri, N., “Trellis coded vector quantization”, IEEE Trans. Commun., vol. COM-40, (Aug. 1992), 1273–1276.

48. Malone, K. T. and Fischer, T. R., “Trellis-searched adaptive predictive coding of speech”, IEEE Trans. Speech and Audio Processing, vol. 1, (Apr. 1993), 196–206.

49. Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P., “Numerical Recipes in C: The Art of Scientific Computing”, 2nd ed. Cambridge University Press, (1992).

50. Stewart, L. C., Gray, R. M. and Linde, Y., “The design of trellis waveform coders”, IEEE. Trans. Commun., vol. COM-30, (Feb. 1982), 702–710.

51. Van Der Vleuten, R. J., “Trellis-based source and channel coding”, Ph.D. dissertation, Delft Univ. Technol., Delft, Netherlands, (Mar. 1994).

52. Xie, M. and Adoul, J. P., “Algebraic vector quantization of LSF parameters with low storage and computational complexity”, IEEE Trans. Speech and Audio Processing, vol. 4, no. 3, (May 1996), 234–239.

53. Adoul, J. P., “La quantification vectorielle des signaux: approche algébrique”, Annales des Télécommunications, vol. 41, no. 3-4, (1986), 158–177.

54. Jeong, D. G. and Gibson, J. D., “Uniform and piecewise uniform lattice vector quantization for memoryless Gaussian and Laplacian sources”, IEEE Trans. Inform. Theory, vol. IT-39, no. 3, (May 1993), 786–803.

55. Conway, J. H. and Sloane, N. J. A., “Fast quantizing and decoding algorithms for lattice quantizers and codes”, IEEE Trans. Inform. Theory, vol. IT-28, no. 2, (Mar. 1982), 227–232.

56. Ricordel, V., “Etude de schémas de quantification vectorielle algébrique et arborescente. Application à la compression de séquence d’images numériques”, Thèse de Doctorat, Université de Rennes 1, (Déc. 1996).

57. NIST, “The DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus”, (Oct. 1990).

58. ITU-T, Recommendation P.341, “Characteristics of wideband terminals”, ITU, Geneva, (1994).

59. ITU-T, Recommendation G.191, “Software tools for speech and audio coding standardization”, ITU, Geneva, (Nov. 2000).

60. “3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Speech codec speech processing functions; Adaptive Multi-Rate - Wideband (AMR-WB) speech codec; Transcoding functions (Release 7)”, 3GPP TS 26.190, (Jun. 2007).

61. So, S. and Paliwal, K. K., “A comparative study of LPC parameter representations and quantisation schemes for wideband speech coding”, Digital Signal Processing, vol. 17, no. 1, (Jan. 2007), 138–171.

62. Chatterjee, S. and Sreenivas, T. V., “Conditional PDF-based split vector quantization of wideband LSF parameters”, IEEE Signal Processing Letters, vol. 14, no. 9, (Sep. 2007).

63. Eriksson, T., Linden, J. and Skoglund, J., “Interframe LSF quantization for noisy channels”, IEEE Trans. Speech and Audio Processing, vol. 7, no. 5, (Sep. 1999), 495–509.

64. Kövesi, B., “Quantification vectorielle des paires de raies spectrales pour la compression de la parole à débit réduit : application à la téléphonie numérique sur canal acoustique sous-marin”, Thèse de Doctorat, Université de Rennes 1, (1997).

65. Xie, M., “Quantification vectorielle algébrique et codage de parole en bande élargie”, Thèse de Doctorat, Université de Sherbrooke, (Fév. 1996).

66. Fuchs G., “Codage audio hiérarchique à faibles débits”, Thèse de Doctorat, Université de Sherbrooke, Février 2007.

67. Halimi M., Kaddai A. and Bengherabi M., “A New Multistage Search of Algebraic CELP Codebooks Based on Trellis Coding”, Special Issue on Speech

Information Processing, IEICE Transactions on Information and Systems, vol. E86-D, no. 3, (Mar. 2003), 406−411.

68. Kaddai A. and Halimi M., “Low-Complexity Wideband LSF Quantization Using Algebraic Trellis VQ”, IEICE Transactions on Information and Systems, vol. E92-D, no. 12, (Dec. 2009), 2478−2486.