Université du Québec
INRS (Centre Énergie Matériaux Télécommunications)

Codage échelonnable à granularité fine de la parole utilisant la quantification vectorielle arborescente

Par
Mouloud DJAMAH

Thèse présentée pour l'obtention du grade de

Doctorat en Télécommunications

Jury d'évaluation

Président du jury et examinateur interne	Leszek Szczecinski, INRS-EMT
Examinateur externe	Roch Lefebvre, Dép. génie électrique et génie informatique Université de Sherbrooke
Examinateur externe	Jean Rouat, Dép. génie électrique et génie informatique Université de Sherbrooke
Directeur de recherche	Douglas O'Shaughnessy, INRS-EMT

À ma mère
À la mémoire de mon père

Remerciements

Ce travail a été réalisé au sein du groupe communication parlée à l'Institut National de la Recherche Scientifique-Centre Énergie Matériaux Télécommunications (INRS-EMT) de l'Université du Québec. Je voudrais remercier le directeur de cette thèse le Professeur Douglas O'Shaughnessy pour sa disponibilité, son encadrement et ses suggestions.

J'adresse mes vifs remerciement à Monsieur Leszek Szczecinski, Professeur à l'INRS-EMT de l'Université du Québec, d'avoir accepté de présider le jury de thèse.

J'adresse aussi mes sincères remerciements à Monsieur Roch Lefebvre, Professeur à l'université de Sherbrooke, et à Monsieur Jean Rouat, Professeur à l'université de Sherbrooke, pour avoir accepté d'examiner ce travail et de participer au jury.

J'exprime aussi ma reconnaissance à toutes les personnes qui m'ont apportés leur soutien, leur amitié tout au long de cette thèse. Je cite en particulier mon ami Chikhi Samir et Chowdhury Md Foezur Rahma (de l'INRS-EMT).

Merci enfin à ma mère mes frères et sœurs pour leur soutien à distance mais sans faille.

Résumé

Avec le déploiement de la transmission de la parole sur les réseaux à paquets, où la congestion du trafic est chose commune, il est devenu une exigence de développer des algorithmes de compression de parole qui doivent être capables d'adapter dynamiquement le débit binaire de la parole compressée à la bande passante du réseau de transmission produisant une variation lisse et graduelle de la qualité de la parole synthétisée.

Dans certains codeurs normalisés de type CELP ou paramétrique, l'encodeur génère un seul type de flux-binaire à un débit fixe. Cependant, si le trafic dans le canal de transmission est congestionné, les données transmises pourraient être perdues. Pour gérer la congestion, le flux-binaire est organisé en différentes parties avec divers degrés d'importance. La partie la plus importante est désignée par la couche noyau et les parties les moins importantes sont désignées par les couches d'amélioration. Lorsque le canal de transmission est congestionné, les couches d'amélioration peuvent être rejetées une couche à la fois sans que la qualité de la parole décodée ne subisse une dégradation importante. Le débit binaire peut être ajusté à la volée par la troncation du flux-binaire incorporé à un point quelconque de la chaîne de communication. Différentes versions de la parole peuvent, donc, être reconstruites au niveau du décodeur et la suppression des paquets, qui altère sévèrement la qualité de la parole décodée, est ainsi évitée. L'échelonnabilité à granularité fine FGC (Fine Granularity Scalability) est une approche où le flux-binaire peut être rejeté avec une granularité plus fine, bit par bit dans le cas extrême, au lieu d'une couche entière.

L'objectif de ce travail consiste à modifier des codeurs de parole normalisés de telle manière à ce qu'ils produisent des flux-binaires incorporés rendant ainsi ces codeurs échelonnables et évitant ainsi la suppression des paquets. Cependant les modifications qui doivent être introduites dans ces codeurs (pour les rendre échelonnables) ne doivent pas altérer les performances de ces derniers aux débits pour lesquelles ils ont été destinées à l'origine. Un algorithme efficace de conception d'un quantificateur vectoriel arborescent, utilisant la technique de fusion de cellule, est proposé. La quantification vectorielle arborescente est utilisée pour produire une échelonnabilité à granularité fine (bit par bit).

La qualité de la parole produite par ces codeurs ainsi modifiés est évaluée en fonction de la longueur (nombre de bits) du flux-binaire reçu par le décodeur.

Des procédures de recherche rapides pour localiser des vecteurs-code au sein d'un quantificateur ayant une structure multi-étages arborescente ou non sont proposées. Ces algorithmes sont évalués pour la quantification des coefficients LSF (Line Spectral Frequencies) en comparaison à d'autres algorithmes de recherche rapides connus. Un logiciel interactif d'outils pour le codage de la parole est aussi proposé.

Abstract

With the deployment of speech transmission over packet networks, where traffic congestion is common, it has become a requirement to develop speech compression algorithms that must be able to adapt dynamically the bit-rate, of the compressed speech, to the channel bandwidth producing a smooth and gradual variation of the synthesized speech quality.

In several standardized CELP or parametric speech coders, the encoder generates only one type of bit-stream at a fixed bit-rate. However, if the traffic in the transmission channel is congested, the transmitted data could be lost. To handle the congestion, the bit-stream is organized into different portions with various degrees of importance. The most important portion is referred to as the core layer and the less important portions are referred to as enhancement layers. When the transmission channel is congested, the enhancement layers can be discarded one layer at time without a strong degradation of the speech quality. The bit-rate can be adjusted on the fly by truncating the embedded bit-stream at any point of the communication chain. Therefore, different versions of the speech can be provided at the decoder and packet dropping that severely impairs the decoded speech quality is avoided. Fine Granularity Scalability (FGS) is an approach wherein the bit-stream can be discarded with finer granularity, on a bit-by-bit basis in the extreme case, instead of the whole layer.

The objective of this work consists of modifying standardized speech coders in such a way that these coders produce embedded bit-streams avoiding packet dropping. However the introduced modifications should not deteriorate the performances of these coders when operating at their original rates. An efficient codebook design for tree-structured vector quantization, using the cell merging technique, is proposed. The tree-structured vector quantization is used to provide fine granularity scalability. The speech quality produced by the modified coders is evaluated according to the length (number of bits) of the bit-stream received by the decoder.

Fast search algorithms to reduce the search complexity required to locate the codevectors during encoding in multistage tree-structured (or not) vector quantization are proposed. These algorithms are compared to other known fast algorithms for vector

quantization of LSF (Line Spectral Frequencies) parameters. Interactive software of tools for speech coding is proposed too.

Table des Matières

Page

Liste des Figures

Liste des Tableaux

Liste des Abréviations

ACELP	Prédiction linéaire avec excitation par code algébrique *(algebraic CELP)*
ACR	Évaluation par catégories absolues (*Absolute Category Rating*)
ADPCM	PCM différentiel adaptatif *(Adaptive Differential PCM)*
AR	Autorégressif *(Auto Regressive)*
ARMA	Autorégressif à moyenne ajustée *(Auto Regressive Moving Average)*
BWE	Extension de bande passante *(bandwidth extension)*
CELP	Prédiction linéaire avec excitation par code (*code-excited linear prediction*)
CS-ACELP	Conjugate Structure - Algebraic CELP
DEMUX	Démultiplexeur
DFT	Transformée de Fourier Discret *(Discrete Fourier transform)*
FEC	Masquage d'effacement de trame *(frame erasure concealment)*
FFT	Transformée de Fourier rapide *(Fast Fourier Transform)*
FGC	Echelonnabilitée à granularité fine *(Fine Granularity Scalability)*
FIR	Réponse impulsionnelle finie *(finite impulse response)*
FS1016	Codeur de parole fédérale standard 1016 *(US Federal standard 1016 speech coder)*
HB	Bande supérieure *(higher band)*
HPF	Filtre passe-haut *(high pass filter)*
HVXC	Harmonic Vector Excitation Coding
IP	Protocole Internet *(internet protocol)*
IRS	Système de référence intermédiaire (*Intermediate Reference System*)
LAR	Logarithme des rapports d'aires *(log area rations)*
LB	Bande inférieure *(lower-band)*
LD-CELP	CELP à faible délai (*low delay CELP*)
LP	Prédiction linéaire *(linear prediction)*
LPC	Codage par prédiction linéaire *(linear predictive coding)*
LPF	Filtre passe-bas *(low pass filter)*
LPAS	Analyse par synthèse *(Linear Prediction Analysis-by-Synthesis)*
LSB	Bit de plus faible poids *(least significant bit)*

LSF	Fréquence de raie spectrale *(line spectrum frequency)*
LSP	Paire de raies spectrales *(line spectrum pair)*
LTP	Prédiction à long terme *(long-term prediction)*
MA	Moyenne ajustée (Moving Average)
MELP	Le codeur à prédiction linéaire à excitation mixte *(Mixed Excitation Linear Prediction)*
MDCT	Transformée en cosinus discret modifié *(modified discrete cosine transform)*
MJLS	Recherche conjointe multi-trajets assistée par des niveaux *(Multipath Joint Level-assisted Search)*
MJTS	Recherche conjointe multi-trajets assistée par une structure arborescente *(Multipath Joint Tree-assisted Search)*
MMSE	Minimisation de l'erreur quadratique moyenne *(Minimum Mean Square Error)*
MOS	Note moyenne d'opinion *(Mean Opinion Score)*
MPE-LPC	Le codeur à prédiction linéaire excité par des impulsions *(Multi Pulse Excitation - LPC)*
MPEG	Motion Pictures Expert Group
MCPM	Minimum Cost Perfect Matching
MSB	Bit de plus fort poids *(most significant bit)*
MSS	Recherche séquentielle multi-trajets *(Multipath sequential search)*
MSTS	Recherche séquentielle multi-trajets assistée par une structure arborescente *(Multipath Sequential Tree-assisted Search)*
MSVQ	Quantification vectorielle multi-étages *(Multistage Vector Quantization)*
MTVQ	Quantificateur vectoriel arborescent multi-étages *(Multistage Tree-structured Vector Quantization)*
PCM	Modulation par impulsion codée *(Pulse Code Modulation)*
pdf	Fonction densité de probabilité *(probability density function)*
PESQ	Évaluation de la qualité vocale perçue (*Perceptual Evaluation of Speech Quality*)
PTSVQ	Quantificateur TSVQ élagué *(pruned TSVQ)*
PVQ	Quantification vectorielle prédictive *(predictive VQ)*
QMF	Banc de filtrage miroir en quadrature *(quadrature mirror filterbank)*
RMS	Valeur quadratique moyenne (*root mean square*)

RPE	Le codeur excité par des impulsions régulièrement espacées *(Regular Pulse Excitation)*
SD	Distorsion spectrale *(Spectral distorsion)*
SNR	Rapport signal sur bruit (*signal-to-noise ratio*)
SQ	Quantificateur scalaire (scalar quantizer)
STL2005	version 2005 de la bibliothèque d'outils logiciels (*Software Tools Library release 2005*)
SVQ	Quantification vectorielle divisée *(Split Vector Quantization)*
TDAC	Annulation de crénelage de domaine temporel *(time domain aliasing cancellation)*
TDBWE	Extension de bande passante de domaine temporel *(time-domain bandwidth extension)*
TSVQ	Quantification vectorielle arborescente *(Tree structured VQ)*
ITU	Union internationale des télécommunications *(International Telecommunication Union)*
VoIP	Téléphonie utilisant le protocole Internet *(voice over IP)*
VQ	Quantification vectoriel *(vector quantization)*
WB	Large bande *(wideband)*

Introduction

Les communications numériques sont en plein développement actuellement et les applications de téléphonies numérique et mobile se multiplient. Pour économiser les ressources des canaux de communication, le signal numérisé doit alors être compressé pour réduire le flux de données nécessaires à une reconstruction de bonne qualité du signal de parole d'origine. Le terme numérisation correspond à l'opération d'échantillonnage du signal de parole continu suivi d'une quantification, ce qui représente respectivement une discrétisation des axes des temps et des amplitudes. Le codage de parole a bénéficié d'un grand nombre d'études, ce qui a permis d'avoir une bonne connaissance sur le système de production de la parole. D'importantes réductions de débit peuvent ainsi être obtenues par des algorithmes de compression efficaces, qui tiennent compte des redondances naturelles de la parole.

Dès le début des années 70 apparaît la technique de Prédiction Linéaire LPC (Linear Predictive Coding), supposant que le signal de parole peut être considéré comme la sortie d'un filtre tout-pôle. Un codeur LPC particulièrement performant utilise des techniques d'analyse par synthèse LPAS (Linear Prediction Analysis-by-Synthesis). Dans un système de codage LPAS, le signal de parole à transmettre est codé en recherchant la meilleure excitation possible d'un filtre de synthèse, dont les coefficients sont déterminés par une analyse de Prédiction Linéaire. Le plus connu de ces codeurs LPAS est le codeur de type CELP (Code-Excited Linear Prediction) conçu pour le codage de la parole à des débits allant de 4 à 16 kbits/s. Pour des débits inférieurs à 4 kbps, les codeurs paramétriques sont plus efficaces. Pour atteindre ces taux de compression, les systèmes de codage paramétriques se basent sur la connaissance du processus de production de la parole. La technique consiste à extraire du signal de parole les paramètres les plus pertinents permettant au décodeur de le synthétiser. Les performances des codeurs paramétriques, appelés aussi vocodeurs, dépendent de la précision des modèles de production de parole. La plupart des codeurs paramétriques sont basés sur le codage prédictif linéaire (LPC), connus sous le nom de vocodeurs prédictifs. Dans ce type de vocodeurs, le conduit vocal est modélisé par un filtre tout-pole. Grâce au développement de processeurs DSP (Digital Signal Processing) performants, des modèles mathématiques

très proches du système phonatoire humain ont pu être mis au point et utilisés dans des algorithmes de codage de plus en plus complexes. Bien que les progrès technologiques soient considérables, la complexité algorithmique a encore un impact sur le coût d'un encodeur/décodeur de parole. En effet, pour un matériel ayant une puissance de calcul et une capacité de mémoire limitées, les paramètres de complexité et de temps de traitement deviennent des facteurs cruciaux lorsqu'il s'agit d'applications en temps réel. La plupart des codeurs de parole récemment standardisés utilisent la quantification vectorielle pour la transmission de certains paramètres, notamment les coefficients de prédiction linéaire. Afin de réduire la complexité de calcul exigé durant le processus d'encodage dans une quantification vectorielle, des méthodes de recherche rapides et efficaces sont généralement utilisées.

Avec le déploiement de la transmission de la parole sur les réseaux à paquets (voix sur IP), où la congestion du trafic est chose commune, il est devenu une exigence de développer des algorithmes de compression de parole qui doivent être capables d'adapter dynamiquement le débit binaire, de la parole compressée, à la bande passante du réseau de transmission produisant une variation lisse et graduelle de la qualité subjective de la parole synthétisée. Ainsi l'échelonnabilitée du débit binaire dans les codeurs de parole devient une exigence importante. Dans plusieurs codeurs normalisés de type CELP ou de type paramétrique, l'encodeur génère un seul type de flux-binaire à un débit fixe. Cependant, si le trafic dans le canal de transmission est congestionné, les données transmises pourraient être perdues. Pour gérer la congestion, une solution consiste à organiser le flux-binaire en différentes parties avec divers degrés d'importance. La partie la plus importante est désignée par la couche noyau (ou couche de base), et les parties les moins importantes sont désignées par les couches d'amélioration. La couche noyau fournit la qualité de base. Cette qualité peut être améliorée par les couches d'amélioration qui peuvent être décodées comme des extensions à la couche noyau. Lorsque le canal de transmission est congestionné, les couches d'amélioration peuvent être rejetées une couche à la fois (de la couche la moins importante à la couche la plus importante) sans que la qualité de la parole décodée ne subisse une dégradation importante. En fonction des contraintes du réseau de transmission, le débit binaire peut être ajusté à la volée par

la troncation du flux-binaire incorporé à un point quelconque de la chaîne de communication telle que les routeurs. Grâce à cette adaptation très flexible du débit binaire, différentes versions de la parole peuvent être reconstruites (à de divers niveaux de qualité) au niveau du décodeur et la suppression des paquets, qui altère sévèrement la qualité de la parole décodée, est ainsi évitée. L'échelonnabilitée à granularité fine FGC (Fine Granularity Scalability) est une approche où le flux-binaire peut être rejeté avec une granularité plus fine, bit par bit dans le cas extrême, au lieu d'une couche entière.

L'objectif de ce travail consiste à modifier des codeurs de paroles normalisés de tel manière à ce qu'ils produisent des flux-binaires incorporés rendant ainsi ces codeurs échelonnables et évitant ainsi la suppression des paquets. Cependant les modifications qui doivent être introduites dans ces codeurs (pour les rendre échelonnables) ne doivent pas altérer les performances de ces derniers aux débits pour lesquelles ils ont été destinées à l'origine. La quantification vectorielle arborescente conçue par fusion de cellule est utilisée pour produire une échelonnabilitée à granularité fine (bit par bit). La qualité de la parole produite par ces codeurs ainsi modifiés est évaluée en fonction de la longueur (nombre de bits) du flux-binaire reçu par le décodeur. Des procédures de recherche rapides pour localiser des vecteurs-code au sein d'un quantificateur ayant une structure multi-étages arborescente ou non sont proposées. Ces algorithmes sont évalués pour la quantification des coefficients LSF (Line Spectral Frequencies) en comparaison à d'autres algorithmes rapides. Un logiciel interactif d'outils pour le codage de la parole est aussi proposé.

La méthodologie adoptée, dans la rédaction de cette thèse, est présentée comme suit :

Dans le Chapitre 1, nous décrivons brièvement le principe de la quantification vectorielle comme méthode de compression de données. Cette méthode s'applique essentiellement dans les domaines de traitement d'image et de la parole. L'algorithme de Lloyd-Max généralisé ou encore l'algorithme LBG pour l'élaboration de dictionnaires non-structurés est présenté. On parlera aussi de quantification vectorielle divisée SVQ (Split Vector Quantization), de quantification vectorielle multi-étages MSVQ (Multistage Vector Quantization) ainsi que d'autres types de quantificateur vectorielle qui

correspondent à des schémas structurés permettant une réduction substantielle en complexité de mémoire et en complexité de calcul par comparaison à une quantification vectorielle non-structurée. La plupart des codeurs de paroles récemment standardisés utilisent la quantification vectorielle hybride, formée par la combinaison de plusieurs types de quantificateurs structurés, pour la transmission de certains paramètres comme les coefficients de prédiction linéaire.

Le Chapitre 2 donne quelques généralités sur le codage de la parole. Les caractéristiques et la modélisation du système phonatoire humain y sont présentées. Il rappelle aussi les principes des différentes techniques connues pour compresser un signal de parole. Le principe du codage CELP, basé sur le codage par prédiction linéaire et l'analyse par synthèse, est présenté avec plus de détail. Le codeur CS-ACELP (Conjugate Structure - Algebraic CELP) normalisé par la recommandation G.729 de l'Union Internationale des Télécommunications (UIT-T) et du codeur paramétrique fédéral standard MELP sont décrits. Une description de la quantification des coefficients LSF dans la norme G.729 est décrite aussi. Les différentes techniques d'échelonnabilitée dans le cadre du codage de parole sont introduites et une brève description de quelques codeurs échelonnables est donnée. Une description un peu plus détaillée du codeur G.729.1 récemment standardisé par l'UIT-T est donnée comme un exemple de codeur englobant plusieurs techniques d'échelonnabilitée.

Le Chapitre 3 introduit le principe d'un quantificateur vectoriel à structure arborescente binaire équilibrée comme un quantificateur incorporé lié au concept d'échelonnabilité à granularité fine. Les deux principales méthodes de conception d'un quantificateur arborescent qui sont les méthodes descendante et ascendante (ou encore par fusion de cellules) sont décrites en détails. Un algorithme efficace de conception d'un quantificateur arborescent binaire équilibré basé sur la technique de fusion de cellules est proposé. Cet algorithme est basé sur le principe de construction d'un arbre binaire équilibré d'une certaine taille par connexion de sous-arbres de petites tailles. La méthode proposée est basée sur deux algorithmes qui sont l'algorithme de groupage et l'algorithme

de permutation d'indices qui sont décrits en détails. La performance de ces deux procédures est testée.

Dans le Chapitre 4, des procédures de recherche rapides utilisées lors du processus d'encodage pour la quantification vectorielle à plusieurs étages sont décrites dans un premier temps. Pour un quantificateur multi-étages MSVQ, l'algorithme MSS (Multipath Sequential Search), connu aussi sous le nom de recherche *M-L*, est introduit. La performance de cet algorithme est très proche de l'algorithme de recherche exhaustif avec une complexité de calcul beaucoup plus faible. Pour un quantificateur vectoriel arborescent multi-étages MTVQ (Multistage Tree-structured Vector Quantization), l'algorithme MSTS (Multipath Sequential Tree-assisted Search) récemment proposé est décrit. En comparaison à ces deux méthodes, deux procédures de recherche rapides sont proposées. La première méthode appelée algorithme MJTS (Multipath Joint Tree-assisted Search) utilise une structure arborescente multi-étages, et la deuxième méthode appelée algorithme MJLS (Multipath Joint Level-assisted Search) est moins exigeante en mémoire et utilise une structure multi-étages avec une contrainte sur le nombre de dictionnaires par étage.

Le Chapitre 5 décrit, d'une manière générale, le logiciel d'outils pour le codage de la parole réalisé dans le cadre de ce travail. Les différents algorithmes présentés dans ce projet ont été implémentés sous forme d'un logiciel interactif utilisant le langage C# de Visual Studio 2005. Certains outils de la bibliothèque STL2005 de l'UIT-T ont été aussi intégrés dans ce logiciel, comme procédures de prétraitements, permettant ainsi de les appliquer sur des bases de données de parole importantes. D'autres normes ont été intégrées dans le logiciel, notamment les outils utilisés pour mesurer la qualité de la parole synthétisée. Le logiciel réalisé peut ainsi être utilisé par d'autres chercheurs comme un outil normalisé (utilisant des outils normalisés de l'UIT-T) efficace pour le prétraitement et l'évaluation de bases de données de parole importantes.

Dans le Chapitre 6, nous modifions deux normes de codage de parole qui sont : le codeur standard fédéral MELP et le codeur ITU-T G.729 (CS-ACELP) de telle manière à ce qu'ils produisent des flux-binaires incorporés rendant ainsi ces codeurs échelonnables

avec une granularité fine. La qualité de la parole produite par ces codeurs, ainsi modifiés, est évaluée en fonction de la longueur (nombre de bits) du flux-binaire reçu par le décodeur. La procédure de recherche rapide MJLS proposée au Chapitre 4 est testée dans le cadre de la quantification vectorielle des paramètres LSF. L'algorithme est examiné pour deux attributs : la qualité des paramètres encodés et la complexité d'encodage. Diverses comparaisons de performance (à différents débits binaire) avec les méthodes MSS et MSTS sont également fournies.

Enfin, une conclusion générale termine ce travail de thèse.

a) Contributions

• Un algorithme efficace de conception d'un quantificateur TSVQ binaire équilibré basé sur la technique de fusion de cellules est proposé. La structure arborescente est conçue depuis le niveau le plus élevé de l'arbre vers les niveaux les plus bas. L'idée principale de la méthode proposée est basée sur la construction d'un arbre binaire d'une certaine taille (certaine hauteur) comme une connexion de sous-arbres optimaux de petites tailles. Cet algorithme de conception utilise conjointement trois algorithmes: la méthode de recherche exhaustive conjointe (proposé par Chu en 2006), l'algorithme de groupage et la procédure de permutation d'indices. Ces deux derniers algorithmes (algorithme de groupage et la procédure de permutation d'indices) ont été proposés dans ce travail de recherche. Des simulations ont prouvées l'efficacité de ces algorithmes. Par extension l'algorithme proposé est utilisé pour la conception d'un quantificateur MTVQ à K étages qui peut être vu comme une structure de K étages où chaque étage correspond à une structure arborescente. Notons que l'algorithme de groupage (utilisé conjointement avec la procédure de permutation d'indices) peut également être utilisé pour une construction descendante d'un arbre binaire équilibré à partir des vecteurs-code d'un dictionnaire non-structuré.

• Deux normes de codeur de parole ont été modifiées : un codeur paramétrique (la norme FS-MELP) et un codeur hybride de type CELP (la norme G.729 ou CS-ACELP). Pour ces deux codeurs, les quantificateurs originaux (pour la quantification des coefficients LSF et/ou des amplitudes de Fourier) sont remplacés par des quantificateurs basés sur une

structure TSVQ ou MTVQ. La conception de ces quantificateurs est basée sur l'algorithme proposé dans ce travail (technique de fusion de cellules). Les codeurs modifiés sont échelonnables en débit binaire avec une granularité fine avec un changement graduel de la qualité de la parole synthétisée (corrompue ou non). Aux débits les plus élevés (et dans le cas d'une transmission non-bruitée) les codeurs modifiés ont les mêmes qualités que les codeurs normalisés correspondants. Pour le cas de la quantification SVQ dans le contexte d'une quantification vectorielle prédictive (le deuxième étage de la structure MSVQ du quantificateur prédictif des coefficients LSF de la norme G.729), il est montré que la performance (en terme de note PESQ) est meilleure quand la priorité est accordée aux cinq coefficients LSF inférieurs au lieu d'accorder la même priorité à tous (les dix) les coefficients.

• Deux algorithmes de recherche rapides pour réduire la complexité de calcul exigée pour localiser des vecteurs-code pendant le processus d'encodage, utilisant une quantification vectorielle multi-étages, sont proposés. L'algorithme MJTS utilise une structure MTVQ dont le coût de mémorisation est approximativement le double de celle d'une structure MSVQ. L'algorithme MJLS utilise un certain nombre de niveaux (de dictionnaires) à chaque étage et le coût de mémorisation peut être proche de celle d'une structure MSVQ lorsque seuls deux dictionnaires sont utilisés à chaque étage. La performance de la procédure de recherche MJLS est comparée aux procédures MSTS et MSS. La quantification des paramètres LSF à différents débits binaires est réalisée pour fournir des résultats expérimentaux. Comparé à la technique MSTS, l'algorithme MJLS proposé donne de meilleurs résultats en termes de distorsion spectrale moyenne et/ou les «outliers» avec une réduction de la complexité de mémorisation et une réduction significative de la complexité de calcul. En effet, la charge mémoire est réduite de 28% à 40% et la complexité de calcul de 30% à 40% alors qu'une meilleure qualité (en terme de SD et de «outliers») est obtenue pour des débits allant de 22 à 28 kbits/s. Comparé à l'algorithme MSS bien connu, l'algorithme de MJLS peut réaliser une importante réduction de la complexité de calcul avec une faible dégradation de la performance (ou une performance comparable) et une légère augmentation de la complexité de mémorisation. En effet, la charge mémoire est augmentée de 13% à 25% et la complexité

de calcul est réduite de 30% à 55% alors que la distorsion spectrale est augmentée de 0.01 à 0.04 dB pour des débits allant de 22 à 28 kbits/s. Les mêmes remarques peuvent être faites pour l'algorithme MJTS comparé aux algorithmes MSS et MSTS. Cependant les procédures MJTS et MSTS ont la même complexité de mémorisation puisque utilisent la même structure de quantificateur.

• Un logiciel interactif utilisant Visual Studio 2005 (le langage C#) est proposé. En plus, des différents algorithmes présentés dans ce travaille de recherche, certains outils de la bibliothèque STL2005 ainsi que la norme ITU-T P.862 PESQ de l'IUT-T ont également été intégrés dans ce logiciel permettant ainsi d'effectuer des prétraitements sur une base de données de parole. Aussi, une base de données de parole initiale peut être comparée à une base de données de parole dégradée. Le logiciel implémenté peut ainsi être utilisé par d'autres chercheurs comme un outil normalisé (utilisant des outils normalisés de l'UIT-T) efficace pour le prétraitement et l'évaluation de bases de données assez large faisant ainsi économiser un temps important.

Certaines parties de notre travail ont été publiées dans des conférences internationales et un article journal est sous presse dans la revue Speech Communication:

M. Djamah and D.O'Shaughnessy, "Low-complexity encoding of speech LSF parameters using multistage tree-structured vector quantization: application to the MELP coder", *IEEE Canadian Conf. on Electrical and Computer Eng.*, pp. 376-380, 2009.

M. Djamah and D.O'Shaughnessy "A fast tree-structured search procedure for multistage vector quantization of LSF parameters", *International Conference Signal and Image Processing*, Honolulu Hawaii, USA, pp. 42-46, August 2009.

M. Djamah and D.O'Shaughnessy "Fine-Granular Scalable MELP Coder Based on Embedded Vector Quantization", *10th Annual Conference of the International Speech Communication Association (Interspeech 2009)*, pp 2603-2606, September 2009.

M. Djamah and D.O'Shaughnessy, "An efficient tree-structured codebook design for embedded vector quantization", *IEEE ICASSP*, pp. 4686-4689, March 2010.

M. Djamah and D.O'Shaughnessy, "Fine Granularity Scalable Speech Coding Using Embedded Tree-structured Vector Quantization", *Elsevier Speech Comm. (2011)*, doi: 10.1016/j.specom.2011.06.002.

Chapitre 1

Quantification vectorielle

1.1 Introduction

La quantification vectorielle, comme méthode de compression de données, joue un rôle très important dans le domaine de la communication, que ce soit dans un but de transmission ou d'archivage d'informations. Cette méthode s'applique essentiellement dans les domaines de traitement d'image et de la parole. La quantification vectorielle quantifie un ensemble de valeurs conjointement, comme un seul bloc ou vecteur. Cela consiste alors à représenter tout vecteur $\boldsymbol{x}$ par un autre vecteur $\boldsymbol{y}_i$ de même dimension mais ce dernier appartenant à un ensemble fini de L vecteurs (niveaux). Le but de la compression étant d'extraire une information maximale tout en ne créant qu'un minimum de distorsion par rapport au signal original. La théorie de l'information [1, 2] annonce d'une façon générale que de meilleurs résultats en codage sont obtenus si l'on quantifie des vecteurs plutôt que des scalaires. En contre partie une complexification des systèmes ainsi que des délais de calcul plus importants sont nécessaires. Ce gain de la quantification vectorielle sur la quantification scalaire est notamment dû à l'exploitation des corrélations existantes entre les composantes des vecteurs.

1.2 La quantification scalaire

La quantification scalaire est un cas particulière de la quantification vectorielle où la dimension des vecteurs est égale à un.

1.2.1 La quantification scalaire uniforme

Le quantificateur scalaire uniforme à débit fixe est entièrement déterminé par :

• Les $L+1$ niveaux de décisions : x_0, x_1, …, x_L, qui partitionnent en L intervalles égaux l'axe des réels $\mathbb{R}$ et déterminent le pas de quantification.

• Les L valeurs de reproduction : y_1, y_2, …, y_L, qui sont les centres de masse de chacun des intervalles de décision.

Le débit (en bits/échantillon) de ce quantificateur est donc donné par $R = \log_2 L$ [bits/échantillon]. Il apparait deux sortes d'erreurs ou bruits de quantification :

• Le bruit granulaire qui se produit lorsque la valeur d'entrée x se situe dans l'un des intervalles $[x_i, x_{i+1}]$, l'erreur résultante qui est la différence entre x et la valeur quantifiée $Q(x)$ ne peut être supérieure à un demi pas de quantification. Le pas de quantification étant défini par: $\Delta = y_{i+1} - y_i$, $i = 1,2,...,L-1$.

• Le bruit de surcharge ou de dépassement qui se produit lorsque la valeur d'entrée se situe hors de l'intervalle $[x_0, x_L]$. La valeur de reproduction est alors : y_1 ou y_L, et l'erreur de quantification peut être supérieure à un demi pas de quantification.

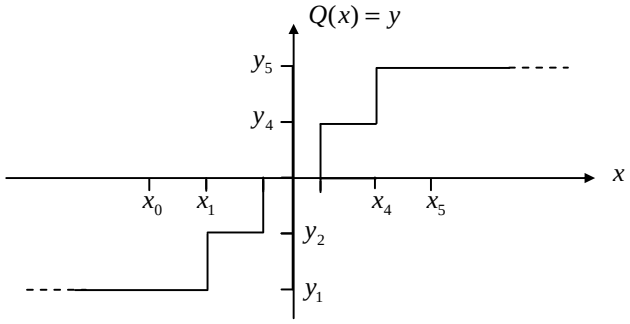


Figure 1.1 : Exemple d'un quantificateur scalaire uniforme avec cinq niveaux $L = 5$

1.2.2 Le quantificateur scalaire optimal

Le quantificateur scalaire optimal est celui qui minimise, pour une source donnée et un débit fixé, l'erreur moyenne de reconstruction due aux bruits de quantification et de surcharge. Les niveaux de reconstruction sont donc répartis en tenant compte de la densité de probabilité de la variable à quantifier. La concentration des niveaux de reconstruction est plus importante dans la zone de l'espace où la densité de probabilité des valeurs à quantifier est plus élevée. Dans la pratique on ne connaît pas la densité de probabilité des valeurs à quantifier. On utilise donc une base de données d'apprentissage composée d'un grand nombre d'échantillons représentatifs de la source. Pour construire un quantificateur optimal, on utilise l'algorithme (très connu) de Lloyd-Max [3, 4] sur la base de données d'apprentissage.

1.3 Principe de la quantification vectorielle

Par extension à la quantification scalaire, qui quantifie chaque valeur d'un signal d'une manière indépendante, la quantification vectorielle quantifie un ensemble de valeurs conjointement, comme un seul bloc ou vecteur [5-9]. La quantification vectorielle VQ (vector quantization) consiste alors à représenter tout vecteur $\boldsymbol{x}$ de dimension k par un vecteur $\boldsymbol{y}_i$ de même dimension appartenant à un ensemble fini appelé dictionnaire $\mathbf{Y}$. Les vecteurs $\boldsymbol{y}_i \in \mathbf{Y}$ sont appelés les vecteurs représentants, les vecteurs de reproduction ou les vecteurs-code.

Un quantificateur vectoriel de dimension k et de taille N peut être défini mathématiquement comme une application Q de R^k vers le dictionnaire $\mathbf{Y}$ de R^k :

$$\begin{array}{ll} Q: & R^k \rightarrow \mathbf{Y} \\ & \boldsymbol{x} \rightarrow Q(\boldsymbol{x}) = \boldsymbol{y}_i \end{array} , \qquad (1.1)$$

où $\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{N-1}\}$ et $\boldsymbol{y}_i \in R^k$ pour $0 \le i < N$. Cette application Q détermine implicitement une partition de l'espace source R^k en N régions : $\mathbf{R}_i$, $i = 0,...,N-1$ où la région $\mathbf{R}_i$ est associée au vecteur $\boldsymbol{y}_i$. Ces régions sont appelées classes, cellules ou

régions de Voronoï. Généralement, la taille du dictionnaire $\mathbf{Y}$ (le nombre de vecteurs-code du dictionnaire) est de la forme $N = 2^n$, où n est le nombre de bits utilisés pour représenter chaque indice binaire d'un vecteur-code $\boldsymbol{y}_i$. Pour une abréviation de notation, l'indice binaire i est écrit comme un entier positif. Un quantificateur vectoriel se décompose généralement en deux fonctions séparées : l'encodeur et le décodeur.

1.3.1 L'encodeur

Le rôle de l'encodeur consiste, pour tout vecteur $\boldsymbol{x}$ du signal d'entrée, à rechercher dans le dictionnaire $\mathbf{Y}$ le vecteur-code $\boldsymbol{y}_i$ le plus proche de $\boldsymbol{x}$. La notion de proximité est modélisée par une fonction de distance entre les deux vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}_i$: $d(\boldsymbol{x}, \boldsymbol{y}_i)$. Dans le cas de la distance Euclidienne on a :

$$d(\boldsymbol{x}, \boldsymbol{y}_i) = \sum_{j=0}^{k-1} (x_j - y_{i,j})^2 , \tag{1.2}$$

où $\boldsymbol{x} = [x_0, x_1, ..., x_{k-1}]^T$ et $\boldsymbol{y}_i = [y_{i,0}, y_{i,1}, ..., y_{i,k-1}]^T$. Les régions de Voronoï sont données par :

$$\mathbf{R}_i = \left\{ \boldsymbol{x} \in R^k / Q(\boldsymbol{x}) = \boldsymbol{y}_i, \text{si } d(\boldsymbol{x}, \boldsymbol{y}_i) \le d(\boldsymbol{x}, \boldsymbol{y}_j), \forall\ j \ne \text{i} \right\}, \tag{1.3}$$

avec : $\mathbf{R}_i \cap \mathbf{R}_j = \Phi$, $i \ne j$, $\forall\, i = 0, 1, ..., N-1$, $\forall\ j = 0, 1, ..., N-1$. Tous les vecteurs $\boldsymbol{x}$ de R^k qui appartiennent à la région $\mathbf{R}_i$ sont représentés par le même vecteur $\boldsymbol{y}_i$ du dictionnaire $\mathbf{Y}$.

Soit $I = \{0, 1, ..., N-1\}$ l'ensemble des indices correspondant au dictionnaire $\mathbf{Y}$ et $\boldsymbol{y}_i$ $(i \in I)$ l'ensemble des vecteurs de $\mathbf{Y}$. La compression d'information est réalisée à ce niveau car c'est uniquement l'indice du vecteur-code $\boldsymbol{y}_i$ minimisant le critère de distorsion qui sera transmis ou stocké au lieu du vecteur lui même. Cette opération, qui perd de l'information, fait que la QV est une opération irréversible, le signal original ne pourra plus être restitué. La quantité d'information requise pour représenter le vecteur

source est donnée par le débit binaire r en bits par composante du vecteur, ou bits par dimension, ou encore bits par échantillon [5] :

$$r = \frac{1}{k}\log_2(N). \quad \text{[bits/échantillon]} \tag{1.4}$$

Le nombre de bits par vecteur-code (appelé aussi la résolution du quantificateur) est alors donné par : $n = kr$ (bits/vecteur).

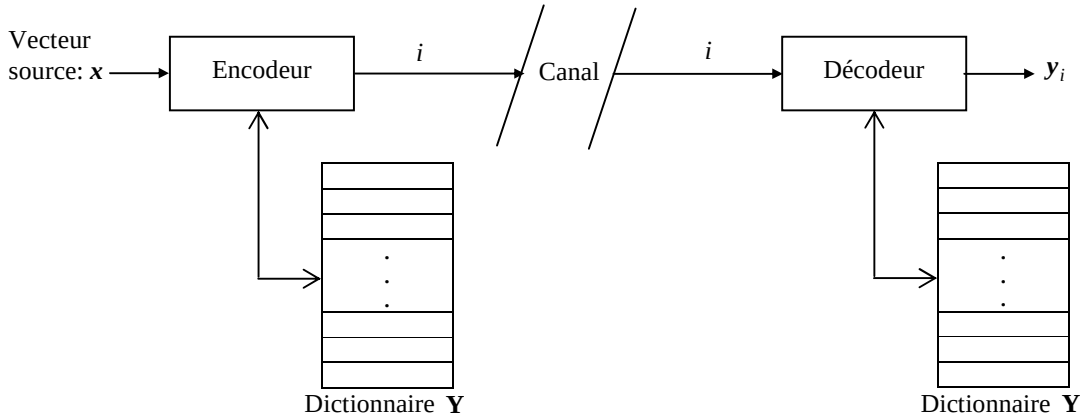


Figure 1.2 : Schéma général d'un quantificateur vectoriel

1.3.2 Le décodeur

Le décodeur est considéré comme un récepteur dont la tâche est la reconstruction du vecteur source. Pour cela, il dispose d'une réplique du dictionnaire qu'il consulte afin de restituer le vecteur-code correspondant à l'indice reçu. Le décodeur réalise donc l'opération de décompression. On donne à la figure 1.2 le schéma général d'un quantificateur vectoriel.

1.3.3 Quantificateur vectoriel optimal

Pour une distribution statistique donnée de la source et un débit donné, le quantificateur optimal est celui qui minimise la distorsion moyenne [5] :

$$D = E\left[d(\boldsymbol{X}, Q(\boldsymbol{X})\right] = \int_{\boldsymbol{x}\in R^k} d(\boldsymbol{x}, Q(\boldsymbol{x}))\, f_{\boldsymbol{X}}(\boldsymbol{x})\, d\boldsymbol{x}, \tag{1.5}$$

où $\boldsymbol{X}$ est un vecteur source aléatoire dans $\mathbb{R}^k$, avec la fonction densité de probabilité (pdf) $f_{\boldsymbol{X}}(\boldsymbol{x})$ spécifiée et qui correspond à la pdf conjointe des composantes X_i, $i = 0, ..., k-1$, du vecteur $\boldsymbol{X}$. Pour un quantificateur vectoriel donné ayant pour dictionnaire $\mathbf{Y} = \{\boldsymbol{y}_i, i = 0 \text{ à } N-1\}$ et pour partition les classes $\mathbf{R}_i$, $i = 0 \text{ à } N-1$, la distorsion moyenne est donnée par :

$$D = \sum_{i=0}^{N-1} \int_{\boldsymbol{x} \in \mathbf{R}_i} d(\boldsymbol{x}, \boldsymbol{y}_i) f_{\boldsymbol{X}}(\boldsymbol{x}) d\boldsymbol{x}. \tag{1.6}$$

Dans la pratique, on utilise un ensemble suffisamment grand de vecteurs d'apprentissage $\boldsymbol{x}_j$ ($j = 0 \text{ à } N_t$) assez représentatif de la statistique de la source à coder. Dans ce cas la distorsion moyenne peut être estimée par l'opérateur :

$$D = \sum_{i=0}^{N-1} \left[\sum_{j | \boldsymbol{x}_j \in \mathbf{R}_i} d(\boldsymbol{x}_j, \boldsymbol{y}_i) \right]. \tag{1.7}$$

Pour un débit donné, le quantificateur optimal est celui qui minimise la distorsion moyenne *D*. La recherche de l'optimum global est très complexe (cela consiste à chercher simultanément la meilleure partition et les meilleurs vecteurs-code qui minimisent la distorsion moyenne *D*), dans la pratique la conception d'un quantificateur (localement) optimal est réalisée d'une manière itérative en vérifiant les deux conditions d'optimalités suivantes [5] :

1. Pour un dictionnaire $\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{N-1}\}$ donné, la partition optimale est celle qui vérifie :

$$\mathbf{R}_i = \left\{ \boldsymbol{x} / d(\boldsymbol{x}, \boldsymbol{y}_i) \le d(\boldsymbol{x}, \boldsymbol{y}_j), \ j \ne i, \forall \ j \in \{0, 1, ..., N-1\} \right\}. \tag{1.8}$$

C'est la règle du plus proche voisin.

2. Pour une partition donnée, le vecteur représentant $\boldsymbol{y}_i$ doit minimiser la distorsion associée à la cellule $\mathbf{R}_i$. Le vecteur $\boldsymbol{y}_i$ est donc le centroïde de cette cellule :

$\boldsymbol{y}_i = Cent(\mathbf{R}_i)$. Lorsque la distance Euclidienne est utilisée, le centroïde de la cellule $\mathbf{R}_i$ correspond au vecteur moyen de tous les vecteurs appartenant à cette cellule :

$$\boldsymbol{y}_i = \frac{1}{\|\mathbf{R}_i\|} \sum_{j|\boldsymbol{x}_j \in \mathbf{R}_i} \boldsymbol{x}_j \quad , \tag{1.9}$$

où $\|\mathbf{R}_i\|$ est le cardinal (le nombre de vecteurs) de la cellule $\mathbf{R}_i$. L'élaboration du dictionnaire d'un quantificateur vectoriel se fait à partir d'une séquence d'apprentissage. La méthode la plus populaire est l'algorithme de Lloyd-Max généralisé, appelé aussi algorithme LBG [9].

1.3.4 Algorithme LBG

Cet algorithme proposé par Linde, Buzo et Gray [9] correspond à une extension de l'algorithme de Lloyd-Max utilisé pour l'élaboration de dictionnaires dans le cas de la quantification scalaire [3, 4, 5]; d'où son appellation Lloyd-Max généralisé. Pour un dictionnaire initial donné et utilisant une séquence d'apprentissage représentative de la statistique de la source à coder, le rôle de l'algorithme est d'essayer d'optimiser l'encodeur et le décodeur. L'algorithme LBG est une application des deux conditions d'optimalité où la partition et le dictionnaire sont mis à jour itérativement. L'algorithme est donné au tableau 1.1 [9]. Cet algorithme d'optimisation itératif fonctionne à partir d'un dictionnaire initial. Le problème est donc reporté sur le choix de ce dictionnaire initial dont l'expérience a prouvé qu'il contribuait fortement aux performances de l'algorithme LBG. Chaque itération ne provoquant qu'un changement local du dictionnaire, l'algorithme converge vers le minimum local le plus proche du dictionnaire initial.

Plusieurs méthodes ont été proposées pour concevoir un bon dictionnaire initial. Une description de ces méthodes d'initialisation peut être trouvée dans les références [10, 11]. Citons la méthode d'initialisation par dichotomie vectorielle, cette méthode a été proposée dans la version initiale de l'algorithme de LBG sous le nom de "splitting" [9].

Tableau 1.1 : Algorithme LBG

• Entrées :
- Une séquence d'apprentissage: $\boldsymbol{x}_i$, $i = 0,1,..., N_t - 1$.
- Un dictionnaire initial $\mathbf{Y}^{(0)}$ de taille N
- Un seuil $\alpha > 0$
- Un nombre maximum d'itération p

1- Initialisation :
- $D^{(-1)} = \infty$
- $m = 0$

2- Classification :

2.1- Pour le dictionnaire courant $\mathbf{Y}^{(m)} = \left\{\boldsymbol{y}_i^{(m)}, i = 0,1,..., N-1\right\}$, trouver la partition optimale $\mathbf{R}^{(m)} = \left\{\mathbf{R}_i^{(m)}, i = 0,1,..., N-1\right\}$ qui minimise la distorsion moyenne, soit :

$$\boldsymbol{x} \in \mathbf{R}_i^{(m)} \text{ si } d(\boldsymbol{x}, \boldsymbol{y}_i^{(m)}) < d(\boldsymbol{x}, \boldsymbol{y}_j^{(m)}), \quad \forall\, j \neq i$$

2.2- Calcul de la distorsion moyenne : $D^{(m)} = \sum_{i=0}^{N-1} \left[\sum_{j|\boldsymbol{x}_j \in \mathbf{R}_i^{(m)}} d(\boldsymbol{x}_j, \boldsymbol{y}_i^{(m)}) \right]$

3- Test de fin

Si $\left(\frac{D^{(m-1)} - D^{(m)}}{D^{(m)}} \leq \alpha \text{ ou } m = p \right)$ alors arrêter le traitement (le VQ final est décrit par la partition $\mathbf{R}^{(m)}$ et le dictionnaire $\mathbf{Y}^{(m)}$), autrement continuer à l'étape suivante.

4-Réactualiser le dictionnaire :
- Pour la partition $\mathbf{R}^{(m)}$, calculer le dictionnaire optimal $\mathbf{Y}^{(m+1)} = \left\{\boldsymbol{y}_i^{(m+1)}, i = 0,1,..., N-1\right\}$ avec $\boldsymbol{y}_i^{(m+1)} = cent(\mathbf{R}_i^{(m)})$
- $m \leftarrow m+1$.
- Aller à l'étape 2.

Cette méthode produit une succession de dictionnaires de tailles croissantes où à chaque boucle le nombre de vecteurs-code, de dimension fixe, est multiplié par deux. L'algorithme est résumé comme suit [9] :

1-Le premier dictionnaire est constitué d'un seul vecteur-code $\boldsymbol{y}_0$, qui est le centroïde de toute la séquence d'apprentissage $\{\boldsymbol{x}_i\}_{i=0,...,N_t}$.

2-Ce vecteur-code $\boldsymbol{y}_0$ est ensuite fractionné en deux vecteurs-code $\boldsymbol{y}_0 - \boldsymbol{\varepsilon}$ et $\boldsymbol{y}_0 + \boldsymbol{\varepsilon}$, où $\boldsymbol{\varepsilon}$ est un vecteur de perturbation fixe. L'algorithme LBG est ensuite appliqué sur ce dictionnaire (de deux vecteurs) pour produire un dictionnaire (localement optimal) de taille 2.

3-Le processus de fractionnement est appliqué sur chaque vecteur-code du dictionnaire (de taille 2) de l'étape précédente. Ceci génère un dictionnaire de taille 4 et l'algorithme LBG est appliqué pour optimiser ce dictionnaire. Le processus est répété jusqu'à obtenir un dictionnaire optimal de taille désirée N .

1.3.5 Complexité de la recherche au sein du dictionnaire

Le dictionnaire, de taille *N*, obtenu par l'algorithme LBG ne possède aucune structure particulière facilitant la recherche du plus proche voisin. Ainsi pour trouver le vecteur-code le plus proche du vecteur source à coder, N calculs de distances sont nécessaires. Cette procédure de recherche exhaustive a une complexité d'ordre $N = 2^{rk}$ où *r* est le débit, en bits/échantillon, et *k* est la dimension des vecteurs. Il apparaît donc que la complexité (de calcul et de mémorisation) croit exponentiellement avec la dimension (à débit fixe) et le débit (à dimension fixée). La complexité de mémorisation (ou de stockage) est définie comme la taille mémoire nécessaire pour le stockage du dictionnaire : elle est de Nk unités mémoires (réels ou floats). Lorsque la distance Euclidienne est utilisée (équation (1.2.)), la complexité de recherche au sein du dictionnaire exige : Nk multiplications, $(2k-1)N$ additions et $N-1$ comparaisons. Or des performances élevées ne sont atteintes qu'aux débits élevés; mais l'opération de codage devient alors très coûteuse en calculs. Ceci a conduit à envisager de nouvelles méthodes de quantification vectorielle permettant un codage avec une complexité de calcul et de mémorisation plus faibles. On parle alors de quantification vectorielle structurée à l'opposée de la quantification vectorielle non-structurée.

1.4 Quantification Vectorielle structurée

D'une façon générale ces méthodes consistent à imposer au dictionnaire des contraintes structurelles afin de simplifier (réduire la complexité) le codage, cependant une certaine dégradation de qualité est observée comparée à celle d'une quantification vectorielle non-structurée (aucune structure n'est imposé au dictionnaire). Il s'agit alors de faire un compromis. Dans la plupart des applications pratiques, cette dégradation des performances est souvent compensée par une réduction importante en complexité.

1.4.1 Quantification vectorielle divisée

La quantification vectorielle divisée SVQ (Split Vector Quantization) [5, 12], consiste à diviser le vecteur source, candidat à la quantification, en un certain nombre de sous-vecteurs de dimensions fixes et prédéterminés. Chaque sous-vecteur est codé avec un quantificateur indépendant. Soit $\boldsymbol{x} = [x_0, x_1, ..., x_{k-1}]^T$ le vecteur source de dimension k qui doit être quantifié par une structure SVQ. Le vecteur $\boldsymbol{x}$ peut être écrit comme suite : $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_K^T]^T$ où T désigne la transposée et $\boldsymbol{x}_i$ est un sous-vecteur (colonne) de dimension l_i tel que :

$$\sum_{i=1}^{K} l_i = k \,. \tag{1.10}$$

La figure 1.3 montre le diagramme de l'encodeur et du décodeur SVQ. Le sous-vecteur $\boldsymbol{x}_m$, de dimension l_m, est quantifié en $\hat{\boldsymbol{x}}_m = \boldsymbol{y}_i^{(m)}$ qui est un vecteur-code d'indice i issu du dictionnaire $\mathbf{Y}^{(m)}$ de taille N_m et de dimension l_m. Donc, le vecteur-code $\boldsymbol{y}_{i_1}^{(1)}$ est extrait du premier dictionnaire $\mathbf{Y}^{(1)}$ de taille N_1 et de dimension l_1; le vecteur-code $\boldsymbol{y}_{i_2}^{(2)}$ est extrait du deuxième dictionnaire $\mathbf{Y}^{(2)}$ de taille N_2 et de dimension l_2; et ainsi de suite. Par abus de langage on désigne par Q_m et Q_m^{-1} respectivement les opérations de codage et de décodage du quantificateur. L'opérateur Q_m fournit, alors, l'indice du vecteur-code (issu du dictionnaire $\mathbf{Y}^{(m)}$) le plus proche du sous-vecteur $\boldsymbol{x}_m$.

En utilisant la distance Euclidienne, l'erreur de quantification entre le vecteur source $\boldsymbol{x}$ et le vecteur quantifié $\hat{\boldsymbol{x}}$ est le résultat de la somme des erreurs de quantification des K sous-vecteurs :

$$\left\|\boldsymbol{x} - \hat{\boldsymbol{x}}\right\|^2 = \sum_{i=1}^{K} \left\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\right\|^2 . \tag{1.11}$$

Par conséquent, la recherche au sein de chaque dictionnaire peut être effectuée indépendamment des autres dictionnaires (l'ordre dans lequel la recherche est effectuée

n'est pas important) toute en garantissant que le vecteur quantifié obtenu est le meilleur résultat. Donc, l'ensemble d'indices $\boldsymbol{i} = [i_1, i_2, ..., i_K]$ trouvé au niveau du l'encodeur est l'ensemble optimal.

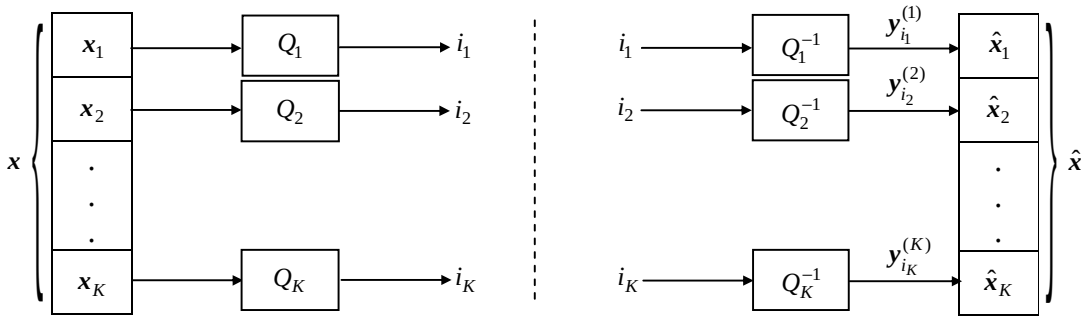


Figure 1.3: Encodeur et décodeur du quantificateur SVQ

Donc, pour la quantification SVQ, la recherche séquentielle au sein de tous les dictionnaires est optimale puisque elle est équivalente à la recherche exhaustive qui consiste à essayer tous les ensembles d'indices possibles. Notons enfin que la conception d'un quantificateur SVQ à *K* dictionnaires consiste à élaborer chaque dictionnaire (algorithme LBG) indépendamment des autres. La quantification SVQ a été utilisée par Paliwal et Atal [12] pour une quantification efficace des coefficients de prédiction linéaire (Appendice A) avec un débit de 24 bits/trame.

1.4.2 Quantification vectorielle multi-étages

L'idée fondamentale de la quantification multi-étages MSVQ (multistage vector quantization) consiste à diviser l'opération de quantification en plusieurs étages successifs. Le premier étage quantifie le vecteur source ***x***. Puis, le deuxième étage opère une quantification sur le vecteur d'erreur entre le vecteur source original et le vecteur quantifié par le premier étage. Le vecteur d'erreur fournit une deuxième approximation au vecteur source original produisant ainsi une représentation plus précise du vecteur source. Un troisième étage peut alors être utilisé pour quantifier l'erreur du deuxième étage pour fournir une autre amélioration et ainsi de suite [5, 13, 14]. Un quantificateur MSVQ à *K* étages est associé à *K* dictionnaires (un dictionnaire pour chaque étage) appelés dictionnaires-étage. La figure 1.4 donne les diagrammes de l'encodeur et du

décodeur d'un quantificateur MSVQ. À l'encodeur, le vecteur source $\boldsymbol{x}$ est comparé avec:

$$\hat{\boldsymbol{x}} = \boldsymbol{y}_{i_1}^{(1)} + \boldsymbol{y}_{i_2}^{(2)} + ... + \boldsymbol{y}_{i_K}^{(K)}, \tag{1.12}$$

où $\boldsymbol{y}_i^{(m)}$ est le vecteur-code d'indice i issu du m ième dictionnaire-étage (le dictionnaire associé a l'étage m de la structure MSVQ). Donc, le vecteur-code $\boldsymbol{y}_{i_1}^{(1)}$ est issu du premier dictionnaire-étage $\mathbf{Y}^{(1)}$ de taille N_1 ; le vecteur-code $\boldsymbol{y}_{i_2}^{(2)}$ est issu du deuxième dictionnaire-étage $\mathbf{Y}^{(2)}$ de taille N_2 ; et ainsi de suite. Notons que tous les vecteurs-code sont de même dimension.

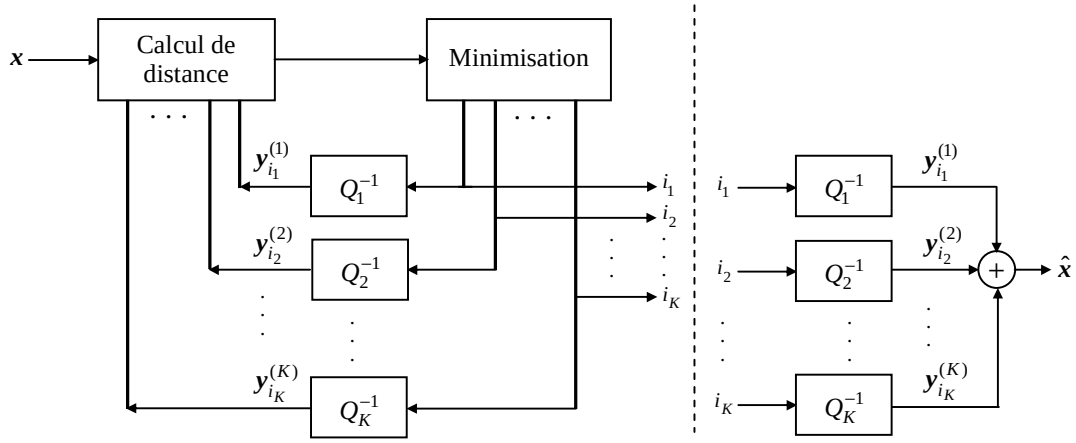


Figure 1.4: Encodeur et décodeur du quantificateur MSVQ à K étages

En choisissant différents ensembles d'indices, l'encodeur essaie de minimiser la distance entre les vecteurs $\boldsymbol{x}$ et $\hat{\boldsymbol{x}}$. L'ensemble d'indices $\boldsymbol{i} = [i_1, i_2, ..., i_K]$, minimisant la distance, est transmis au décodeur. Les vecteurs-code issus des différents étages sont alors additionnés pour former le vecteur quantifié $\hat{\boldsymbol{x}}$ du vecteur source $\boldsymbol{x}$ (figure 1.4).

Pour un quantificateur MSVQ à K étages avec K dictionnaires de tailles N_1, N_2, …, N_K, la résolution (en bits/vecteur) de chaque étage est donnée par : $r_1 = \log_2 N_1$, $r_2 = \log_2 N_2$, …, $r_K = \log_2 N_K$. La résolution du quantificateur est donnée par :

$$r = \sum_{i=1}^{K} r_i = \sum_{i=1}^{K} \log_2 N_i = \log_2 \left(\prod_{i=1}^{K} N_i \right). \tag{1.13}$$

Pour un quantificateur MSVQ à *K* étages correspondants à *K* dictionnaires-étage de tailles N_1, N_2, …, N_K, la tâche de l'encodeur consiste à chercher l'ensemble d'indices $\boldsymbol{i} = [i_1, i_2, ..., i_K]$ qui minimise l'erreur de quantification entre le vecteur quantifié $\hat{\boldsymbol{x}}$ (équation (1.12)) et le vecteur source $\boldsymbol{x}$. La détermination de l'ensemble d'indices optimal $\boldsymbol{i}$ est un problème d'optimisation combinatoire complexe qui peut être résolu, en général, seulement en essayant tous les ensembles d'indices possibles. On parle alors de recherche exhaustive dont la complexité de calcul est très élevée. Des méthodes de recherche séquentielles sous optimales sont typiquement utilisées dans la pratique pour réduire la complexité de calcul. Dans [13, 14] une structure MSVQ est utilisée pour la quantification des coefficients de prédiction.

La conception d'un quantificateur MSVQ à *K* étages consiste à élaborer les *K* dictionnaires associés. Deux méthodes de conception sont largement utilisées en pratique. L'algorithme de conception séquentielle (où les *K* dictionnaires sont élaborés séquentiellement et de manière ordonnée du premier au dernier dictionnaire) et l'algorithme de conception conjoint, où tous les dictionnaires sont conjointement optimisés. L'algorithme de conception conjoint est plus performant que l'algorithme de conception séquentiel [13, 14].

La méthode de conception séquentielle procède comme suit :

1- L'algorithme LBG est appliqué sur la séquence d'apprentissage $\boldsymbol{x}_n$, $n = 0, 1, ..., N_t - 1$ pour produire le dictionnaire du premier étage $\mathbf{Y}^{(1)}$.

2-Pour le deuxième étage, une nouvelle séquence d'apprentissage est générée comme suit:

$$\boldsymbol{x}_n^{(2)} = \boldsymbol{x}_n - Q_1(\boldsymbol{x}_n), \tag{1.14}$$

où $Q_1(.)$ est la fonction de quantification du premier étage. Cette nouvelle séquence d'apprentissage est utilisée pour créer (en appliquant l'algorithme LBG) le dictionnaire du deuxième étage. En général, le dictionnaire du *l* ième étage est généré en utilisant la séquence d'apprentissage suivante :

$$\boldsymbol{x}_n^{(l)} = \boldsymbol{x}_n^{(l-1)} - Q_{l-1}(\boldsymbol{x}_n^{(l-1)}) = \boldsymbol{x}_n - Q_1(\boldsymbol{x}_n) - \ldots - Q_{l-1}(\boldsymbol{x}_n^{(l-1)}) = \boldsymbol{x}_n - \sum_{i=1}^{l-1} Q_i(\boldsymbol{x}_n^{(i)}), \qquad (1.15)$$

avec $\boldsymbol{x}_n^{(1)} = \boldsymbol{x}_n$.

Il est facile de montrer que la quantification SVQ est un cas particulier de la quantification MSVQ [13]. En effet, la quantification vectorielle divisée est une version particulière de la quantification vectorielle multi-étages où une contrainte est imposée : les vecteurs-code de chaque étage ont des composantes nulles et il n'y a pas d'échantillons non nuls qui se recouvrent entre les vecteurs-code de deux étages quelconques. La quantification MSVQ est moins sous-optimale que la quantification SVQ puisqu'aucune contrainte n'est imposée sur les composantes des vecteurs-code. Cependant la quantification MSVQ est plus complexe.

1.4.3 Quantification vectorielle arborescente

La quantification vectorielle arborescente TSVQ (Tree structured VQ) est une technique très efficace pour la réduction de la complexité de codage d'un quantificateur vectoriel [5, 15]. Cependant, le prix à payer consiste en une augmentation de la complexité de mémorisation et une dégradation de performance comparée à la quantification vectorielle non-structurée. Un arbre équilibré *m*-aire de hauteur (profondeur) *L* est un arbre constitué de plusieurs niveaux (du niveau $l = 1$ à L). Chaque niveau est constitué de plusieurs nœuds et chaque nœud est associé à un dictionnaire de *m* vecteurs-code. *m* branches sortent de chaque nœud (une branche pour chaque vecteur-code du dictionnaire associé au nœud) et chaque branche pointe vers un nœud du niveau suivant. Chaque niveau l est donc constitué de m^{l-1} nœuds (dictionnaires) ou encore de m^l vecteurs-code. Le nombre de vecteurs-code du niveau le plus élevé de l'arbre (niveau *L*) est donc de $N = m^L$. La figure 1.5 donne un exemple d'un arbre *m*-aire de hauteur $L = 3$ (du niveau $l = 1$ à 3).

L'encodeur effectue en premier lieu une recherche au sein du dictionnaire racine $\boldsymbol{C}$ pour localiser le vecteur-code produisant la distorsion minimale (le vecteur-code le plus proche du vecteur source $\boldsymbol{x}$). L'indice *i* du vecteur-code sélectionné pointe vers le

prochain dictionnaire (du niveau $l = 2$ de l'arbre) impliqué dans le processus de recherche. La recherche est effectuée donc au sein du dictionnaire $\boldsymbol{C}_i$ et l'indice *j* du vecteur-code sélectionné pointe vers le prochain dictionnaire $\boldsymbol{C}_{i,j}$. Supposons que le prochain niveau $l = 3$ est le dernier (comme montré à la figure 1.5) niveau, la recherche au sein du dictionnaire $\boldsymbol{C}_{i,j}$ produit alors le vecteur-code correspondant au vecteur quantifié $\hat{\boldsymbol{x}}$ du vecteur source $\boldsymbol{x}$. Pour un arbre *m*-aire de hauteur *L*, le nombre de calculs de distances est réduit donc à mL au lieu de m^L (pour une recherche exhaustive de tous les vecteurs-code du niveau le plus élevé de l'arbre).

Le décodeur n'a besoin que des vecteurs-code des dictionnaires du niveau le plus élevé de l'arbre et est identique à une QV conventionnelle. Cependant, pour une transmission progressive, le vecteur quantifié est mis à jour à chaque fois qu'un niveau de l'arbre est atteint. Une séquence complète d'indices est transmise (un indice pour chaque niveau de l'arbre) par l'encodeur et dans ce cas le décodeur a besoin de tous les dictionnaires de la structure arborescente pour construire des approximations de plus en plus fine du vecteur quantifié [15].

Pour un arbre *m*-aire de hauteur *L*, le nombre total de nœuds est donc de : $1 + m + m^2 + ... + m^{L-1}$ et puisque chaque nœud correspond à un dictionnaire de *m* vecteurs-code, le nombre de vecteurs-code qui doivent être stockés est donc de :

$$N' = m(1 + m + m^2 + ... + m^{L-1}) = \frac{m(m^L - 1)}{m - 1}. \tag{1.16}$$

Donc la complexité de mémorisation augmente pour une quantification TSVQ comparée à une quantification non structurée où seulement m^L vecteurs-code doivent être stockés. Cependant la réduction en complexité de calcul peut être très importante pour une quantification TSVQ : mL au lieu de m^L (pour un dictionnaire non structuré). Pour un arbre binaire équilibré, on a $m = 2$ et la complexité de recherche est réduite par un facteur de $2^L / 2L$, alors que la complexité de mémorisation est de $2(2^L - 1)$ vecteurs, un peu moins que le double de la mémoire requise pour le stockage d'un dictionnaire non-structuré.

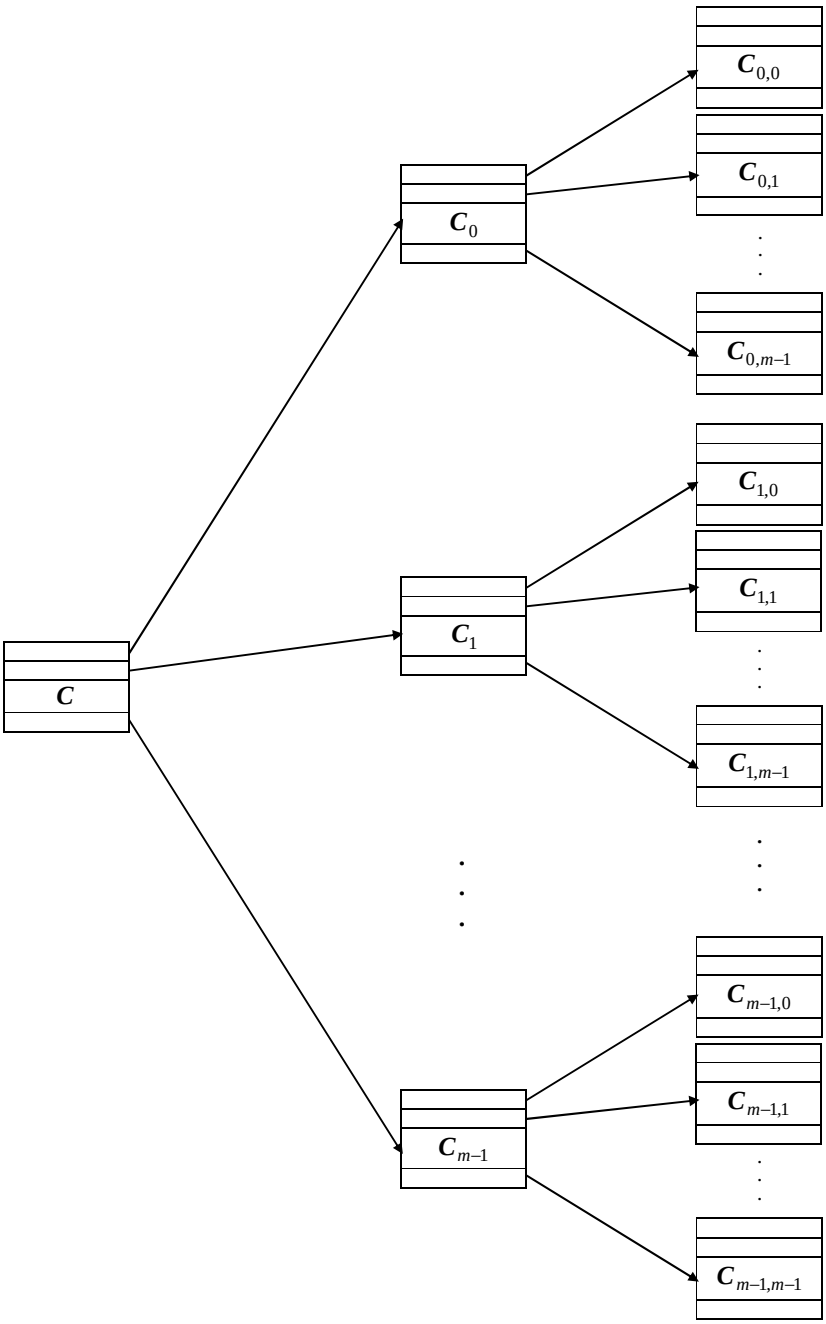


Figure 1.5: quantification vectorielle arborescente

Notons qu'une structure MSVQ peut être vue comme un cas particulier d'une structure TSVQ. En effet, prenons l'exemple de la figure 1.5. Le nombre de dictionnaires du deuxième et du troisième niveau de l'arbre pourrait être réduit à un seul dictionnaire contenant des vecteurs-code erreurs. Dans ce cas la structure correspondrait à une structure MSVQ à trois étages. D'une façon plus générale une structure TSVQ peut subir une contrainte : Au lieu d'une croissance exponentielle du nombre de dictionnaires d'un niveau à l'autre, le nombre de dictionnaires pourrait être limité à partir du deuxième niveau de l'arbre. On parle alors de quantification vectorielle avec contrainte de stockage CSVQ (constrained-storage VQ) [5, 16, 17].

1.4.4 Quantification vectorielle prédictive

Une quantification vectorielle prédictive PVQ (predictive VQ) est utilisée pour exploiter la corrélation entre les vecteurs consécutifs se présentant à l'entrée du quantificateur [5, 18]. Le schéma de base étant une extension de la quantification prédictive scalaire utilisée dans la norme DPCM [18]. La figure 1.6 montre les schémas de principe de l'encodeur et du décodeur pour un quantificateur PVQ qui est typiquement appliqué aux séquences de vecteurs ayant un certain degré de redondance. Par exemple les composantes du vecteur courant et celles des vecteurs passés peuvent avoir un certain degré de corrélation.
Pendant l'opération de l'encodage, le vecteur de différence ou vecteur d'erreur de prédiction est calculé par :

$$\boldsymbol{e}(n) = \boldsymbol{x}(n) - \boldsymbol{x}_p(n), \tag{1.17}$$

où la notation $\boldsymbol{x}(n)$ désigne un vecteur à l'instant *n*. Le vecteur quantifié du vecteur source $\boldsymbol{x}(n)$ est donné par :

$$\hat{\boldsymbol{x}}(n) = \hat{\boldsymbol{e}}(n) + \boldsymbol{x}_p(n), \tag{1.18}$$

où $\hat{\boldsymbol{e}}(n)$ est le vecteur quantifié du vecteur erreur et $\boldsymbol{x}_p(n)$ est le vecteur de prédiction. La prédiction est généralement basée sur un petit nombre de vecteurs quantifiés passés. Du

côté du décodeur, l'indice $i(n)$ est utilisé pour récupérer le vecteur erreur quantifié, qui est combiné avec le vecteur de prédiction pour former le vecteur quantifié du vecteur source (équation (1.18)). Notons que la prédiction est basée sur les vecteurs quantifiés passés. L'approche est utilisée principalement parce que le décodeur n'a pas accès aux vecteurs sources originaux, et une synchronisation doit être maintenue entre l'encodeur et le décodeur. Le prédicteur de la figure 1.6 utilise les vecteurs (sources) quantifiés passés, le prédicteur est donc basé sur le modèle AR (autorégressif).

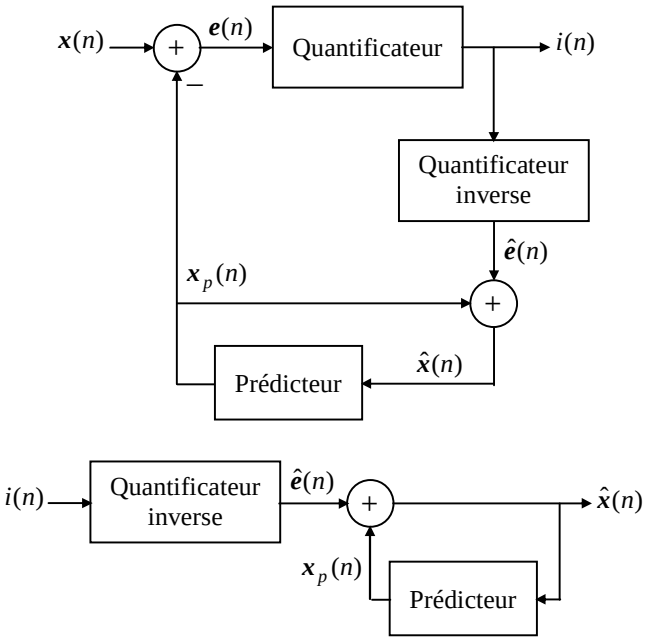


Figure 1.6: Encodeur et décodeur du quantificateur vectoriel prédictif : PVQ.

Une alternative consiste à utiliser un prédicteur basé sur le modèle MA (moyenne ajustée), où l'entrée du prédicteur sont les vecteurs d'erreur de prédiction quantifiés passés (figure 1.7). La performance d'un prédicteur MA est en générale inférieure à celle d'un prédicteur AR ; cependant, la prédiction MA est plus robuste contre les erreurs de canal [18]. La figure 1.7 montre les diagrammes de l'encodeur et du décodeur d'un quantificateur utilisant un prédicteur basé sur le modèle MA : PVQ-MA. La conception d'un quantificateur PVQ-MA consiste à trouver le dictionnaire de VQ optimal et le prédicteur de type linéaire. Dans ce cas on a :

$$\boldsymbol{x}_p(n) = \sum_{i=1}^{M} b_i \hat{\boldsymbol{e}}(n-i), \tag{1.19}$$

où $\boldsymbol{b} = [b_1, b_2, ..., b_M]^T$ est le vecteur des coefficients de prédiction et M est l'ordre de prédiction. La procédure de conception d'un quantificateur PVQ-MA peut être séparée en deux parties : Dictionnaire de quantification vectorielle et prédicteur. Lorsque le dictionnaire est mis à jour, le prédicteur est fixe et lorsque le prédicteur est mis à jour le dictionnaire reste fixe [5, 18].

Notons que lorsque le prédicteur n'est pas fixe, on parle alors de quantification vectorielle prédictive adaptative APVQ (adaptive PVQ). Dans [19] un encodeur APVQ est conçu comme une extension vectorielle du l'encodeur normalisé ADPCM (Adaptive DPCM) [18]. Au niveau de l'encodeur, en plus d'un dictionnaire de vecteurs-code, on peut également avoir un ensemble de prédicteurs prédéterminés. Ainsi en plus de l'indice du vecteur-code, l'indice du prédicteur (sélectionné au niveau de l'encodeur) est également transmit vers le décodeur. La norme G.729 utilise deux prédicteurs pour la quantification des coefficients LSF [39].

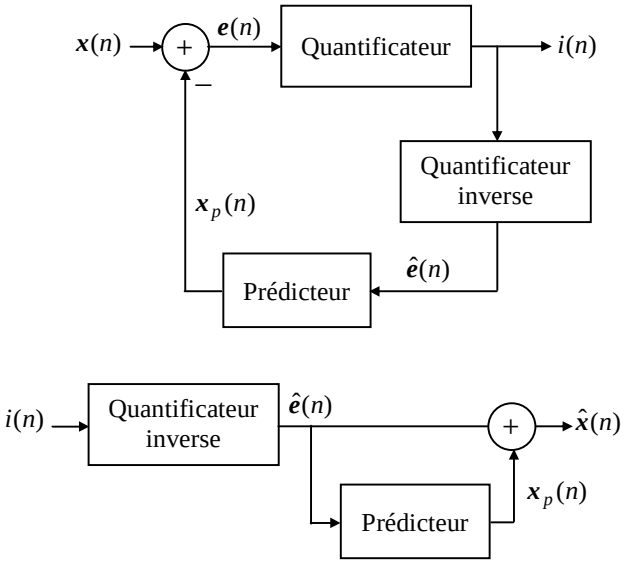


Figure 1.7: Encodeur et décodeur du quantificateur prédictif avec une prédiction MA : PVQ-MA

1.5 Conclusion

L'opération de codage par quantification, qui perd de l'information, fait que la QV est une opération irréversible, le signal original ne pourra plus être restitué. Le codage par quantification permet de réduire le nombre de bits nécessaire à la représentation d'un signal. Le schéma de codage le plus simple est la quantification scalaire qui opère sur des échantillons individuels. L'algorithme de Lloyd-Max permet d'obtenir des niveaux de reproduction produisant une distance minimale. Les performances d'un quantificateur scalaire peuvent être améliorées en quantifiant des vecteurs plutôt que des scalaires. La quantification vectorielle permet un codage à des débits (en bits par échantillon) fractionnaire, ce que la quantification scalaire n'autorise pas. Pour un quantificateur vectoriel utilisant un dictionnaire-non structuré de N vecteurs-code de k composantes chacun, l'algorithme de Lloyd-Max généralisé ou encore algorithme LBG est appliqué pour l'élaboration du dictionnaire. La complexité du codage (pour localiser un vecteur-code au sein du dictionnaire) et la complexité de stockage du dictionnaire est de l'ordre de 2^{rk}. Il apparait donc que la complexité croit exponentiellement avec la dimension k des vecteurs-code et le débit $r = \log_2(N)/k$ (en bits/échantillon). Or des performances élevées ne peuvent être atteintes qu'aux débits élevés et la complexité (de calcul et de mémorisation) devient alors élevée. Afin de réduire cette complexité, des schémas structurés sont souvent imposés. On parle alors de quantification vectorielle SVQ, MSVQ, etc… On parle aussi de quantification vectorielle algébrique, appelée aussi la quantification vectorielle par réseaux réguliers. En utilisant la nature fortement structurée des réseaux réguliers, cette technique permet de réduire la complexité (en stockage et en calcul) de la quantification vectorielle [20-23].

Notons que beaucoup de quantificateurs vectorielles structurés peuvent être considérés comme faisant partie d'une classe plus générale de quantificateurs (structurés) appelés les codes produit (product codes) [5, 8, 16, 17].

La plupart des codeurs de paroles récemment standardisés utilisent la quantification vectorielle structurée pour la transmission de certains paramètres, notamment les coefficients de prédiction qui sont transmis sous forme de coefficients LSF [18]. Cependant on utilise, généralement, des quantificateurs hybrides formés par la

combinaison de plusieurs types de quantificateurs structurés. Ainsi, dans la norme ITU-T G.729 (Chapitre 2) une quantification prédictive (commutée), multi-étages et divisée est utilisée pour la quantification des coefficients LSF. La quantification vectorielle arborescente multi-étages MTVQ (Multistage tree-structured vector quantization) peut aussi être considérée comme la combinaison de deux types de quantification : MSVQ et TSVQ (Chapitre 3).

Chapitre 2

Codage de la parole

2.1 Introduction

Le codage de parole permet la réduction de débit de transmission du signal dans des canaux à largeur de bande limitée. La largeur de bande du canal de transmission doit être minimisée tout en préservant la qualité du signal vocal reconstruit [24]. Dans le cas de la transmission de la voix sur IP, la réduction du débit limite le nombre ou la taille des paquets à envoyer sur le réseau [25].

Tout codage de parole réalise dans un premier temps un prélèvement d'échantillons du signal de parole analogique à un taux d'échantillonnage et une résolution suffisamment grande pour préserver la qualité désirée. Généralement, une fréquence d'échantillonnage de 8/16 kHz et une quantification linéaire de 16 bits sont utilisées. Le signal est alors analysé en utilisant des techniques de traitement numérique du signal pour en extraire une représentation compacte sous forme d'un nombre restreint de bits qui sont envoyés à travers le canal de transmission. Au niveau du décodeur, les informations reçues sont utilisées pour reconstruire le signal de parole, qui est converti de nouveau en un signal analogique. Les codeurs de parole exploitent les redondances du signal de parole pour améliorer leur efficacité. Différentes techniques peuvent être utilisées.

Certains codeurs modélisent le système de production de parole par extraction de paramètres décrivant le signal vocal. D'autres essaient de réduire au minimum l'erreur entre le signal original et les formes d'onde de la parole reconstruite. Plusieurs codeurs à bas débit utilisent une combinaison des deux techniques [18, 26].

2.2 Caractéristiques d'un signal de parole

La connaissance du système vocal et des propriétés du signal de parole est essentielle pour concevoir des codeurs de parole efficaces. Les propriétés du système auditif humain peuvent aussi être exploitées pour améliorer la qualité perceptive du signal codé.

2.2.1 Production de la parole

Le signal vocal est engendré par l'appareil phonatoire avant d'être émis puis détecté par un auditeur. À la perception, l'oreille analyse ce signal et transmet au cerveau les informations nécessaires à son interprétation. La figure 2.1 présente le système phonatoire qui se compose des poumons, du larynx et du conduit vocal formé par le pharynx ainsi que les cavités buccales et nasales.

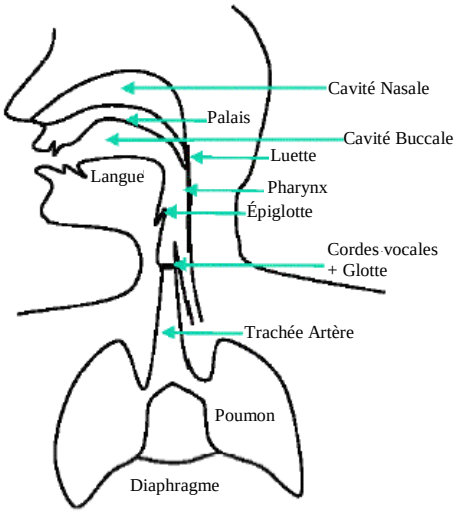


Figure 2.1: Système phonatoire [25]

La parole est généralement produite par expiration de l'air à travers la glotte et le conduit vocal. L'air venant des poumons est modulé par vibrations des cordes vocales et par déformation (élargissement ou resserrement) du conduit [24].

L'appareil respiratoire fournit l'énergie nécessaire à la production du son, en poussant de l'air à travers la trachée artère. Au sommet de celle-ci se trouve le larynx où l'air est modulé avant d'être appliqué au conduit vocal. Les cordes vocales sont en fait deux lèvres symétriques placées en travers du larynx. Ces lèvres peuvent fermer complètement le larynx et, en s'écartant progressivement, déterminent une ouverture triangulaire appelée glotte. L'air y passe librement pendant la respiration et la voix chuchotée, ainsi que pendant la phonation des sons non voisés (des consonnes). Les sons voisés (par exemple, les voyelles) résultent au contraire d'une vibration périodique des cordes vocales.

2.2.2 Sons voisés et non-voisés

Le signal de parole étant un signal réel, continu, d'énergie finie et non stationnaire, les caractéristiques du signal de parole et du conduit vocal évoluent dans le temps. Les positions du système phonatoire agissent comme une opération de filtrage, en augmentant certaines fréquences tout en atténuant d'autres [24]. Malgré sa structure complexe, un signal de parole peut être décomposé, par simplification, en deux types de sons :

• Les sons voisés, tels que des voyelles, sont produits par le passage de l'air des poumons à travers la trachée qui met en vibration les cordes vocales. Ce mode est caractérisé en général par une quasi-périodicité et une énergie élevée.

• Les sons non-voisés, tels que certaines consonnes, sont obtenus par resserrement du conduit vocal, et sont habituellement d'énergie inférieure aux sons voisés. Les cordes vocales sont écartées et n'entrent pas en vibration. Ces sons sont considérés comme ayant les mêmes caractéristiques que le bruit.

La fréquence de vibration des cordes vocales (pour les sons voisés) est appelée fréquence fondamentale F_0 ou encore fréquence du pitch. La période fondamentale ou période du pitch ou simplement le pitch (en nombre d'échantillons) est alors donnée par :

$$T = \frac{F_e}{F_0}, \tag{2.1}$$

où F_e est la fréquence d'échantillonnage.

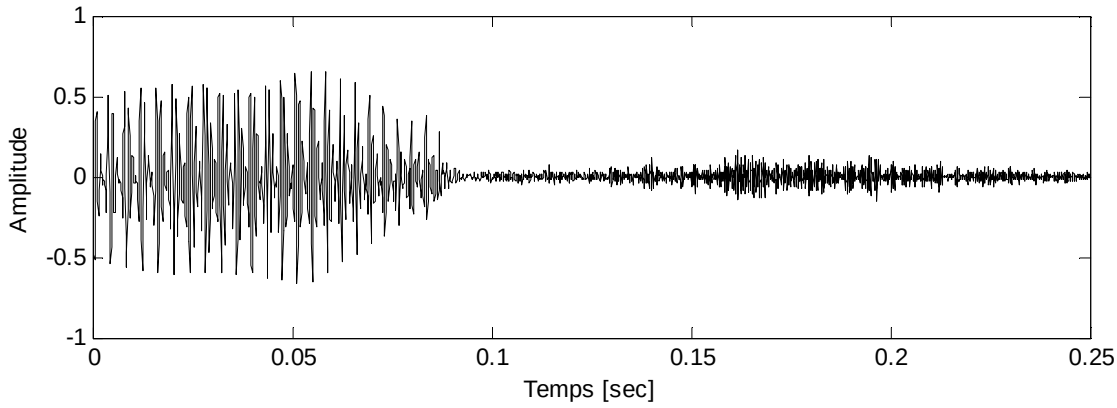
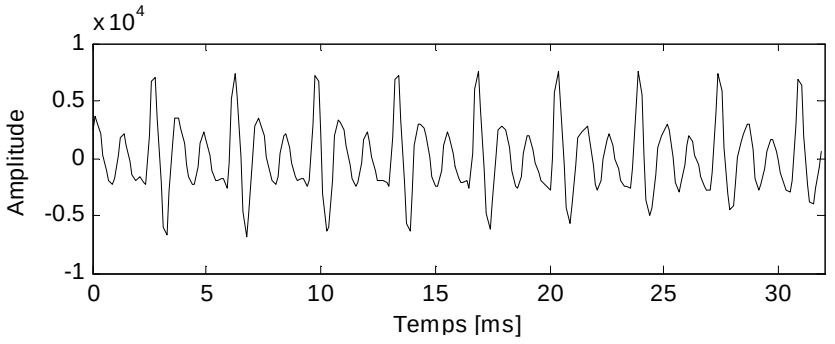
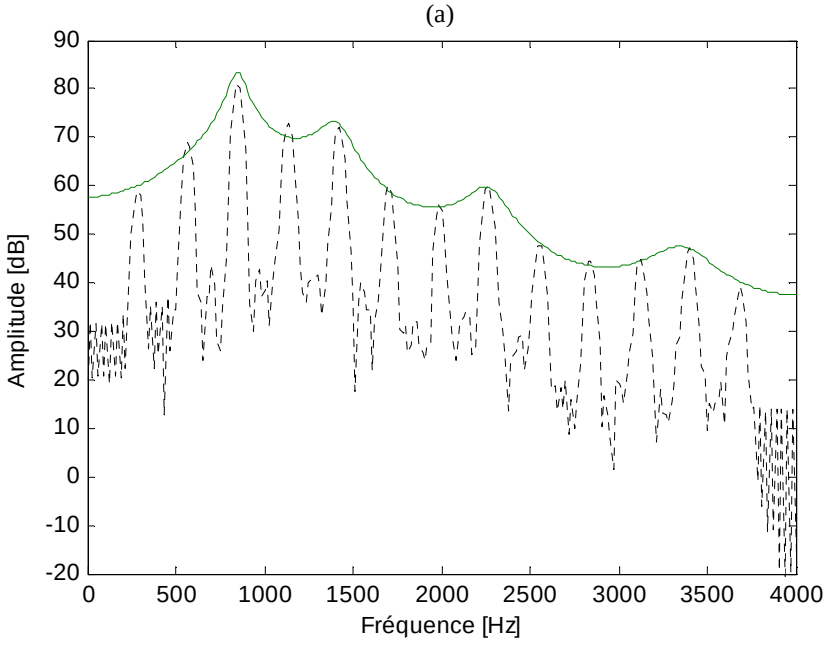


Figure 2.2: Transition d'une séquence de parole voisée vers une séquence non voisée.



(a)



(b)

Figure 2.3: (a) Séquence de parole voisée. (b) densité spectrale de puissance estimée par periodogramme (trait en pointillé) et par les coefficients de prédiction linéaire (trait plein).

La fréquence fondamentale F_0 varie en fonction de chaque individu. En général, la variation s'étend de 80 à 200 Hz pour une voix masculine (voix grave), de 150 à 450 Hz pour une voix féminine et de 200 à 600 Hz pour une voix d'enfant (voix aiguës).

La figure 2.3 montre la densité spectrale de puissance d'un son voisé. La représentation d'un segment de parole périodique dans le domaine fréquentiel (figure 2.3) forme des harmoniques dont l'amplitude est modulée par l'effet de filtrage du conduit vocal. Il en résulte plusieurs zones de fréquences renforcées appelées formants. Notons les trois pics d'amplitude dans le spectre à environ 900, 1500 et 2400 Hz, qui correspondent aux résonances du conduit. Le nombre, les amplitudes et les fréquences des formants varient dans le temps suivant l'articulation des sons voisés. Les deux premiers formants suffisent généralement à la reconnaissance d'une voyelle [24]. Notons que le signal de parole est généralement considéré comme quasi-stationnaire sur des périodes de l'ordre de quelques dizaines de millisecondes (20 ms à 30 ms) appelé trame.

2.2.3 Perception de la parole

La perception de la parole par l'oreille humaine est très complexe. Bien que la quantification d'un signal vocal par une forme d'onde présente des déformations significatives, un auditeur pourra tout de même comprendre la parole synthétisée. Il a été montré que la gamme de fréquences de perception appartenant à l'intervalle [200, 3700] Hz est la plus importante pour l'intelligibilité de la parole. Cette bande de fréquences, où le système auditif est le plus sensible, justifie le taux d'échantillonnage de 8 kHz pour les codeurs de parole [24].

Les analyses temporelles ou spectrales du signal de parole doivent tenir compte des caractéristiques du système auditif pour améliorer l'efficacité des algorithmes de codage de la parole. Les composantes de phase d'un signal de parole jouent un rôle négligeable dans la perception du langage. L'oreille humaine perçoit principalement la parole grâce aux informations du spectre d'amplitude. Les pics du spectre sont plus importants pour la perception que les vallées du spectre et les basses fréquences plus importantes que les hautes fréquences. Le bruit est moins perceptible dans les zones fréquentielles où le signal de parole a beaucoup d'énergie. On parle alors de masquage de fréquences. Tout bruit

inférieur à un certain seuil pourra ainsi être masqué par le signal et devenir inaudible. On dit que le signal masque le bruit. Le bruit est alors plus toléré dans les zones formantiques que dans celles antifomantiques du signal. Pour cela, un masque fréquentiel dont la forme est semblable à l'enveloppe spectrale du signal est, généralement, utilisé [24, 27].

2.2.4 Modèle de production de la parole

Dans une grande majorité de codeurs de parole, la production de parole est modélisée par une opération de filtrage, où une source excite un filtre représentant le conduit vocal [24, 28]. La source et le filtre sont considérés comme séparés lors de l'analyse des signaux de parole. Une modélisation précise du conduit vocal et de la source d'excitation est nécessaire pour produire un signal intelligible et naturel. Des études menées dans le but de définir un modèle produisant un signal proche du signal de parole ont permis à Fant [29] de décrire un modèle de production linéaire. Le signal de parole est modélisé par la sortie d'un filtre excité par un train d'impulsions périodiques (à la cadence du pitch) pour les sons voisés et un bruit blanc pour les sons non-voisés (figure 2.4).

2.3 Codage numérique de la parole

L'accroissement des demandes au niveau des réseaux de communications numériques a favorisé d'importantes recherches. Pour permettre à un plus grand nombre d'usagers de communiquer, il faut que les équipements qui véhiculent ces énormes flux d'informations ne soient pas trop encombrés et saturés ; d'où l'utilité de compresser l'information avant sa transmission. Ainsi des méthodes de traitement du signal sont apparues dans le but de réduire le nombre de bits nécessaires à la représentation du signal de parole et à la transmission des informations tout en maintenant un niveau de qualité suffisant et une complexité de calcul raisonnable. Le développement de microprocesseurs de plus en plus performants a rendu possible la mise en œuvre d'algorithmes de plus en plus sophistiqués pour la compression du signal de parole. Avant tout traitement, il est nécessaire de numériser le signal de parole. Le débit binaire du signal numérisé est alors égal au produit de la fréquence d'échantillonnage par le nombre d'éléments binaires nécessaire à la représentation de toutes les valeurs discrètes du signal. Dans les systèmes

de téléphonie filaire classiques, la parole est numérisée à 64 kbit/s. De nombreux algorithmes ont été proposés pour diminuer ce débit tout en essayant de conserver une qualité subjective suffisante [18, 26-31].

2.3.1 Les codeurs d'onde

Les codeurs temporels utilisent des techniques de codage qui cherchent avant tout à préserver la forme temporelle du signal de parole, ce qui les rend robustes aux différents types d'entrée et ne sont donc pas spécifiques au signal de parole. Ce type de codeur offre des débits supérieurs à 16 kbit/s. La qualité du signal synthétisé obtenue est excellente pour un débit relativement élevé. Le plus ancien procédé de transmission numérique de la parole est la modulation par impulsion codée MIC ou encore PCM (Pulse Code Modulation) [30]. L'Union Internationale des Télécommunications (UIT) a normalisé le codeur G.711 en 1972 [32], un codeur logarithmique de parole de type PCM pour la transmission téléphonique avec un débit de 64 kbit/s. Ce type de codage échantillonne le signal de parole à une fréquence de 8 kHz et opère une quantification sur 8 bits du signal de parole dans la bande de fréquences [300, 3400] Hz. Afin d'exploiter la corrélation présente dans le signal de parole, une technique dite MICDA (MIC différentiel adaptatif) ou encore ADPCM (Adaptive Differential PCM) émerge au début des années 80 et permet de réduire le débit de moitié par rapport à la loi PCM sans détériorer la qualité de la parole. Le principe de la technique MICDA consiste à quantifier non plus la valeur d'un échantillon à un instant donné mais la différence avec une valeur prédite à partir d'échantillons précédents utilisant un prédicteur adaptatif [18]. Le codeur normalisé G.721 est un exemple de système ADPCM qui fonctionne à 32 kbits/s.

2.3.2 Les codeurs paramétriques

Ces codeurs ont été conçus pour des applications à très bas débit (inférieur à 4 kbits/s) et sont principalement prévus pour maintenir l'intelligibilité du signal vocal. Pour atteindre ces taux de compression, les systèmes de codage paramétriques se basent sur la connaissance du processus de production de la parole. La technique consiste à extraire du signal de parole les paramètres les plus pertinents permettant au décodeur de le

synthétiser. Les performances des codeurs paramétriques, appelés aussi vocodeurs, dépendent de la précision des modèles de production de parole. La plupart des codeurs paramétriques sont basés sur le codage prédictif linéaire (LPC), connus sous le nom de vocodeurs prédictifs. Ces codeurs fournissent, à faibles débits, des performances supérieures à celles des codeurs temporels [18].

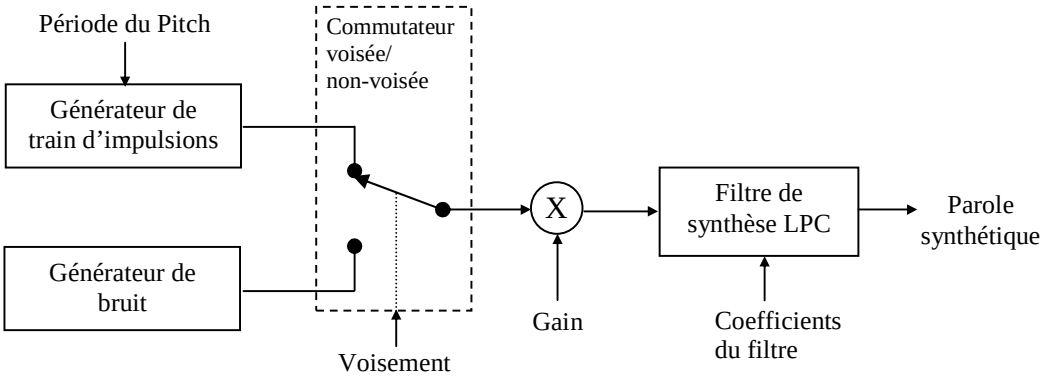


Figure 2.4: Le modèle LPC de production de la parole.

Dans les vocodeurs prédictifs, le conduit vocal est modélisé par un filtre tout-pôle de fonction de transfert $H(z)$:

$$H(z) = \frac{1}{A(z)}, \tag{2.2}$$

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k} \tag{2.3}$$

Le signal de parole est découpé en trames de courte durée. Pour une trame de 20 à 30 ms, le signale de parole est considéré comme stationnaire. Chaque trame de parole est analysée afin de déterminer les paramètres suivants :

- Les coefficients de prédiction $\{a_k\}_{k=1,..,p}$ pour le filtre de synthèse.
- Si la trame de parole est une trame voisée ou non voisée.
- La période du pitch dans le cas d'une trame voisée.
- Le gain principalement lié à l'énergie de la trame de parole.

Ces paramètres sont transmis au décodeur. À la réception, le signal de parole est reconstitué par le passage d'un signal d'excitation à travers le filtre de synthèse LPC $1/A(z)$ (figure 2.4). Le signal d'excitation est un train d'impulsions périodiques (à la cadence du pitch) ou un bruit blanc selon que le signal à reproduire est voisé ou non voisé. Ce signal d'excitation modélise le signal résiduel de prédiction obtenu par le passage du signal de parole à travers le filtre inverse $A(z)$. Le modèle LPC de production de la parole est donné à la figure 2.4. Le standard fédéral FS1015 (LPC10E) fonctionnant à un débit de 2.4 kbit/s est un exemple de codeur de parole paramétrique basé sur le modèle de production LPC [18, 33].

2.3.3 Les codeurs hybrides

La fréquence d'échantillonnage étant fixe, la réduction de débit des codeurs d'onde fait chuter rapidement la qualité d'écoute pour des débits inférieurs à 16 kbit/s. Les codeurs hybrides utilisent les deux méthodes temporelle et paramétrique de façon complémentaire, ce qui permet un codage de parole de bonne qualité à des débits relativement faibles. Ces codeurs sont basés sur des techniques de codage temporel auxquelles des modèles de production de parole sont associés pour améliorer leur efficacité. Cependant, ce type de codage nécessite des coûts de calculs plus importants. Tous les codeurs hybrides s'appuient sur une analyse LPC pour obtenir les modèles de synthèse de parole. Les deux techniques paramétrique et temporelle modélisent respectivement le conduit vocal et le signal résiduel ou erreur de prédiction (l'excitation).

Différentes techniques efficaces ont été proposées pour représenter le signal d'excitation comme, par exemple, le codeur à prédiction linéaire excité par des impulsions MPE-LPC (Multi Pulse Excitation - LPC) [34] qui utilise comme signal d'excitation une suite d'impulsions qui peuvent prendre des amplitudes et être situées à des positions arbitraires. La forme d'onde d'excitation est obtenue en optimisant les positions et les amplitudes d'un nombre fixe d'impulsions par trame. Le codeur excité par des impulsions régulièrement espacées RPE (Regular Pulse Excitation) [35] représente le signal d'excitation par un ensemble d'impulsions régulièrement espacées. Ces codeurs

fournissent une bonne qualité de la parole avec une complexité raisonnable pour un débit autour de 10 kbit/s.

En 1985, Atal et Schroeder [36] définissent le codeur à prédiction linéaire excité par code CELP (Code Excited Linear Prediction), qui détermine une forme d'onde optimale du signal résiduel de prédiction en utilisant la technique d'analyse par synthèse. Le codage CELP réalise une modélisation paramétrique du signal de parole sous la forme d'un signal d'excitation, généralement issu d'un dictionnaire (codebook) de formes d'ondes prédéterminées, passant au travers d'un ou plusieurs filtres. La technique CELP est parmi une des idées les plus influentes dans le codage de la parole et ses principes constituent la base de beaucoup de codeurs normalisés [18]. Par exemple, un codeur LD-CELP (Low Delay-CELP) ciblant un faible délai de codage/décodage a été normalisé par la recommandation G.728 de l'UIT [37]. Ensuite, le codeur G.729 (CS-ACELP), basé sur un codage de parole ACELP (Adaptive Code Excited Linear Predictive) à 8 kbits/s, a été normalisé en 1996 [38, 39].

2.4 Principe du codage CELP

La technique CELP constitue la base de beaucoup de codeurs de parole normalisés [18]. Cette technique, permettant des taux de compression significatifs, cherche à exploiter la redondance du signal de parole. Il suffira alors de ne transmettre au décodeur que l'information non prédictible. Le codage CELP est basé sur le codage prédictif et l'analyse par synthèse (LPAS : Linear Prediction Analysis by Synthesis) [18, 36, 40, 41]. Dans le codage LPAS (figure 2.5), le signal d'entrée est analysé et un signal d'excitation est déterminé. La fonction du codage prédictif consiste à définir les coefficients du filtre de prédiction tandis que le signal d'excitation est modélisé par l'analyse par synthèse. L'erreur entre le signal d'entrée et celui mis en forme par le filtre de synthèse, reproduisant les résonances (formants) du conduit vocal, est alors minimisée par le critère des moindres carrés MMSE (Minimum Mean Square Error) pour choisir le meilleur signal d'excitation. Les données à transmettre ne sont plus des valeurs du signal échantillonné mais des paramètres résultant de la prédiction linéaire et du codage du signal d'excitation. Le calcul des coefficients de prédiction est effectué par trames de parole en boucle ouverte alors que la recherche de l'excitation optimale est généralement

effectuée par sous-trame (la trame étant subdivisée en deux à quatre sous-trames) en boucle fermée. Les paramètres correspondants sont alors transmis au décodeur qui reconstruira le signal de parole en utilisant la même structure de synthèse que celle utilisée en phase d'analyse.

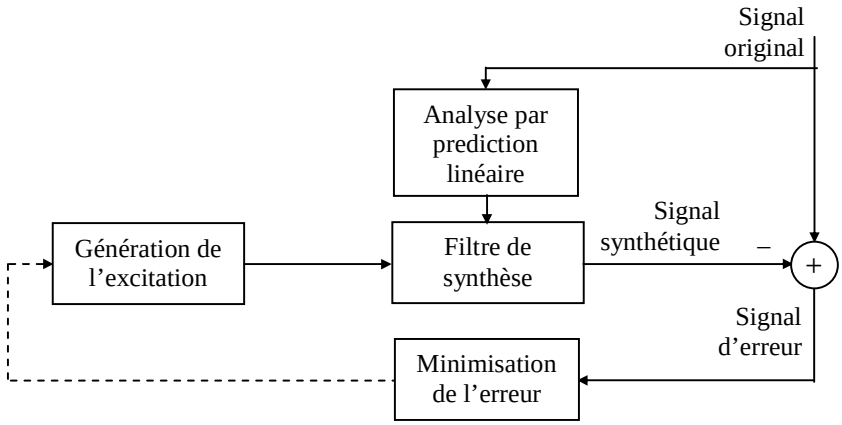


Figure 2.5: Principe du codeur LPAS.

Une des raisons du succès du codage LPAS est la possibilité d'incorporer dans sa structure une fonction qui prend en compte la perception de l'appareil auditif humain. Ce principe a pour but de minimiser un critère d'erreur plus subjectif entre le signal de parole original et le signal de parole synthétique. Ceci est réalisé par pondération des fréquences du signal d'erreur pendant la sélection du signal d'excitation optimal.

2.4.1 L'encodeur

Le schéma de principe du l'encodeur et du décodeur CELP est donné à la figure 2.6. Une analyse par prédiction linéaire est effectuée pour calculer les coefficients de prédiction du filtre de synthèse $1/A(z)$. Le filtre de synthèse de pitch $1/P(z)$ est remplacé par une recherche dans un dictionnaire adaptatif, dont le contenu est mis à jour à l'aide de l'excitation codée passée (Appendice A). Le signal d'excitation est extrait d'un dictionnaire fixe composé de formes d'ondes prédéterminées. La synthèse du pitch revient à ajouter, à l'excitation extraite du dictionnaire fixe, une excitation extraite du dictionnaire adaptatif composé des séquences d'excitations codées passées. Cette technique de codage sélectionne, en parcourant les dictionnaires adaptatif et fixe, les formes d'ondes pour former le signal d'excitation du filtre de synthèse $1/A(z)$ qui

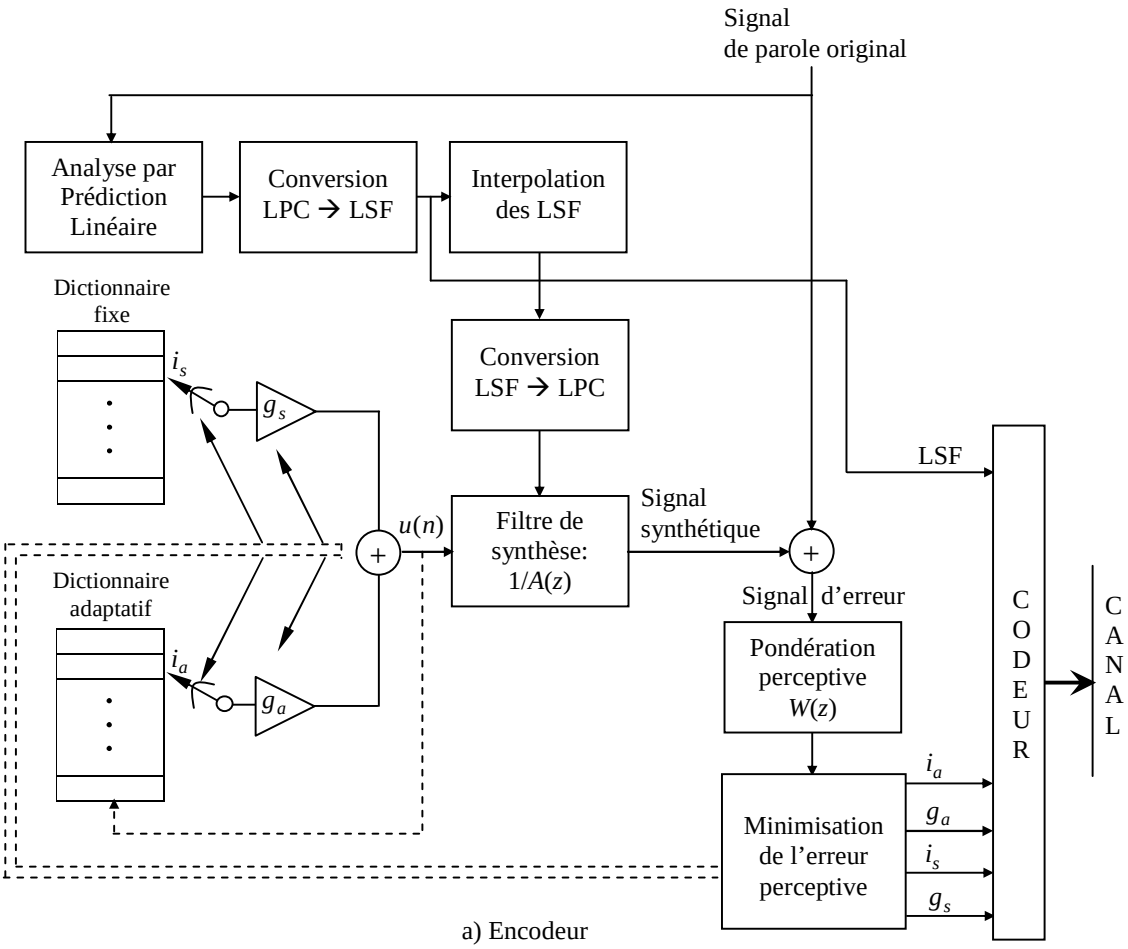
minimise (en tenant compte d'une pondération perceptive) l'erreur quadratique moyenne entre le signal de parole originale et le signal de parole synthétique [18, 36, 40, 41]. Notons que la sélection des formes d'ondes issues des dictionnaires adaptatif et fixe s'effectue de façon séquentielle : la forme d'onde issue du dictionnaire adaptatif est d'abord sélectionnée puis celle issue du dictionnaire fixe. Ce processus, qui consiste à choisir les paramètres d'un dictionnaire (l'indice de la forme d'onde et le gain associé) de manière à optimiser la qualité du signal synthétique, est désigné sous le nom de recherche en boucle fermée et fait suite à l'analyse par prédiction linéaire, dite en boucle ouverte, qui détermine les coefficients de prédiction en analysant uniquement le signal d'entrée.

2.4.2 Analyse par prédiction linéaire

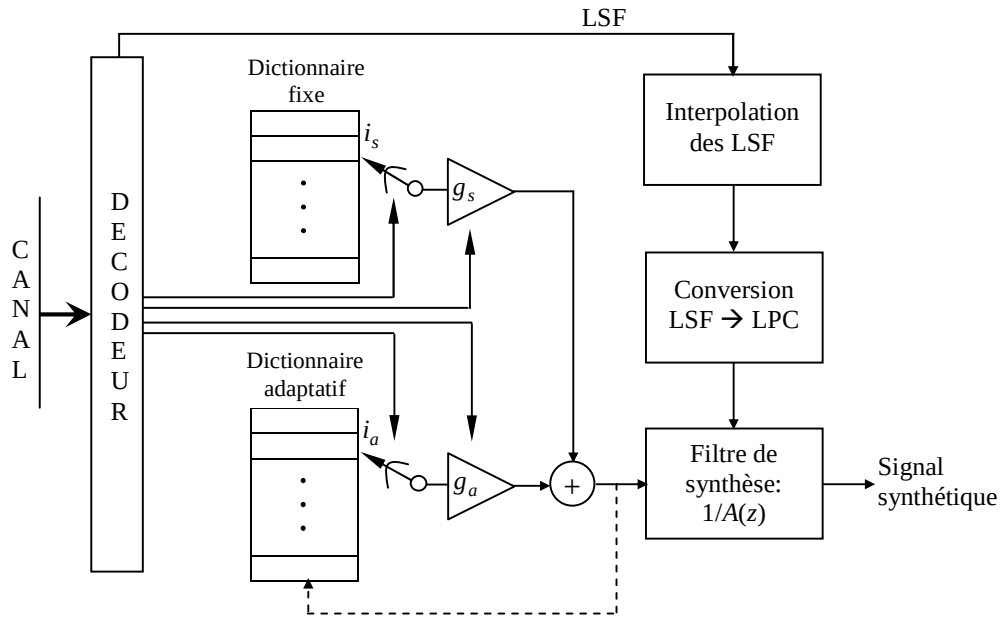
Une analyse par prédiction linéaire est effectuée sur des trames de 10 à 30 ms pour le calcul des coefficients de prédiction [42]. Les coefficients de prédictions $\{a_k\}_{k=1,..,p}$ sont convertis aux coefficients fréquences de raies spectrales LSF : $\{\omega_k\}_{k=1,..,p}$ [43, 44]. Les coefficients LSF sont donc calculés à chaque trame d'analyse de 10 ms à 30 ms. La trame à coder est généralement divisée en 2 à 4 sous-trames. En effet, la modélisation de l'excitation $u(n)$ du filtre de synthèse est effectuée pour chaque sous-trame. Dans le traitement par sous-trame, une interpolation linéaire est généralement réalisée entre deux ensembles de p coefficients LSF chacun (correspondant à deux trames consécutifs) pour former un ensemble intermédiaire de p coefficients LSF pour chaque sous-trame. L'interpolation linéaire est réalisée pour lisser l'évolution des paramètres LP et réduire ainsi la présence de transitions brutales dues aux changements rapides des paramètres LP entre les trames.

2.4.3 Modélisation du signal d'excitation

Le dictionnaire fixe est caractérisé par sa structure. Schroeder et Atal [36] ont proposé un dictionnaire de vecteurs constitués d'échantillons aléatoires gaussiens. Le dictionnaire stochastique obtenu par apprentissage a été proposé en [45].



a) Encodeur



b) Décodeur

Figure 2.6: Schéma de principe du codeur CELP. Au niveau de l'encodeur les paramètres $\{i_a, g_a\}$ et $\{i_s, g_s\}$ de l'excitation $u(n)$ ainsi que les coefficielnts LSF sont transmis au décodeur.

D'autres types de structures ont été envisagés dans le but de réduire la complexité des algorithmes de sélection de la forme d'onde optimale ou le nombre de bits nécessaires pour son codage. On parle alors de dictionnaires algébriques [39, 46, 47, 48].
Le filtre de synthèse de pitch est généralement considéré comme une recherche dans un dictionnaire adaptatif dont le contenu est mis à jour à l'aide de l'excitation passée (Appendice A). La périodicité à long terme, présente particulièrement pour les sons voisés, est ainsi modélisée en représentant l'excitation courante par une excitation passée pondérée par un gain. Le dictionnaire fixe aura alors pour objectif de modéliser les composants aléatoires restants et les segments non-voisés du signal d'excitation. Pour déterminer les paramètres d'excitation, une recherche exhaustive au sein des dictionnaires adaptatif et fixe est effectuée pour déterminer l'excitation $u(n)$, qui est une combinaison linéaire de la contribution des deux dictionnaires (figure 2.6-a) :

$$u(n) = g_s c_{i_s}(n) + g_a v_{i_a}(n) \ , \quad 0 \le n < L \ , \tag{2.4}$$

où les formes d'onde $\{v_{i_a}(n)\}_{n=0,\ldots,L-1}$ et $\{c_{i_s}(n)\}_{n=0,\ldots,L-1}$, pondérées par les gains g_a et g_s respectivement, sont respectivement extraites du dictionnaire adaptatif (à l'indice i_a) et du dictionnaire fixe (à l'indice i_s). On cherche alors les paramètres $\{i_a, g_a\}$ et $\{i_s, g_s\}$ de l'excitation $u(n)$ qui minimisent l'erreur quadratique moyenne (en tenant compte d'une pondération perceptive) entre les sous-trames de parole originales et celles de parole synthétiques reconstruites par le passage de l'excitation $u(n)$ à travers le filtre de synthèse $1/A(z)$. Pour chaque trame de *N* échantillons, l'encodeur transmet *p* coefficients LSF et N/L fois les paramètres du signal d'excitation : $\{i_a, g_a\}$ et $\{i_s, g_s\}$. Le décodeur peut reconstituer le même signal d'excitation qu'à l'encodeur puisque il dispose des mêmes dictionnaires fixe et adaptatif (figure 2.6-b).

2.4.4 Pondération perceptive

Le principe de l'analyse par synthèse, utilisée pour déterminer le signal d'excitation, consiste à minimiser un critère d'erreur entre une séquence du signal de parole original $s(n)$ et le signal de parole synthétisé $\hat{s}(n)$. L'excitation optimale pourrait être obtenue

par minimisation de l'erreur quadratique moyenne entre le signal original et le signal synthétique : $\sum_n (s(n) - \hat{s}(n))^2$. Cependant, ce critère n'étant pas bien adapté à notre système auditif, une correction est intégrée pour pallier à cet inconvénient. Le critère d'erreur est modifié par une pondération perceptive dont le but est de contrôler la répartition fréquentielle de l'erreur en se basant sur le phénomène de masquage spectral du bruit [49]. Le signal d'erreur est moins perceptible dans les régions où le signal a une grande énergie (zones formantiques). On dit que le bruit est masqué par le signal de parole. Ceci permet donc, de tolérer une erreur plus grande dans les zones formantiques que dans celles antiformantique du signal. C'est pourquoi Atal a proposé d'insérer un filtre de pondération de fonction de transfert $W(z)$ avant le critère à minimiser. Ce filtre est donné par [40, 41]:

$$W(z) = \frac{A(z)}{A(z/\gamma)} = \frac{1 + \sum_{k=1}^{p} a_k z^{-k}}{1 + \sum_{k=1}^{p} a_k \gamma^k z^{-k}}, \quad 0 < \gamma < 1. \tag{2.5}$$

En général, la valeur de γ est au voisinage de 0.8. La figure 2.7 donne l'exemple d'un spectre de signal de parole et de la réponse en fréquence du filtre de pondération perceptive associée.

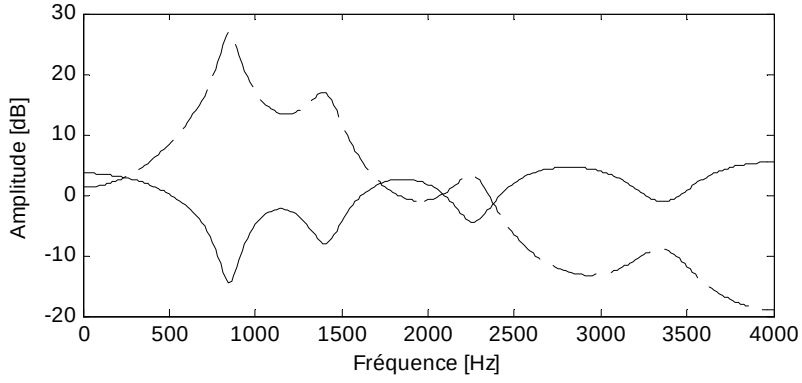


Figure 2.7: Spectre LPC du signal de parole (en pointillé) et réponse fréquentielle du filtre de pondération perceptive $A(z)/A(z/\gamma)$ (trait plein) avec $\gamma = 0.8$.

2.4.5 Le décodeur

Le décodeur dispose des mêmes dictionnaires fixe et adaptatif que l'encodeur. Le dictionnaire adaptatif est actualisé de la même manière qu'à l'encodeur. Le décodeur (figure 2.6) est relativement plus simple à mettre en œuvre :

• les fréquences de raies spectrales (LSF) sont interpolées et reconverties en coefficients de filtre de Prédiction Linéaire pour chaque sous-trame de L échantillons.

•l'excitation est construite par combinaison des contributions adaptatif et fixe.

•le signal de parole est reconstitué par filtrage de l'excitation à travers le filtre de synthèse $1/A(z)$.

Un bloc de post-traitement est ajouté afin d'améliorer la qualité subjective du signal de parole synthétisé. Ce bloc comprend un post-filtre adaptatif composé de trois filtres en cascade: un post-filtre long-terme (utilisé pour améliorer la périodicité du signal) qui nécessite l'estimation de la période du pitch, un post-filtre court terme (ou post-filtre de formant) utilisé pour améliorer la structure formantique du signal synthétisé et un filtre de compensation de l'inclinaison spectrale dû au filtrage passe bas causé par le post-filtrage court terme; suivi par une procédure de contrôle adaptatif du gain (utilisé pour compenser la différence du gain entre le signal de parole synthétisé et le signal post-filtré) [18].

2.5 Le codeur G.729 (CS-ACELP)

La recommandation G.729 de l'Union Internationale des Télécommunications normalise un codeur de parole, basé sur le modèle de codage Prédictif Linéaire à Excitation par Code (CELP) de type CS-ACELP (Conjugate Structure-Algebraic CELP) [18, 38, 39] fonctionnant à 8 kbits/s.

2.5.1 L'encodeur

Le principe du l'encodeur G.729 est donné à la figure 2.8. Avant tout traitement le signal de parole d'entrée subit dans une procédure dite de pré-traitement une normalisation et un filtrage passe-haut dont la fréquence de coupure du filtre est égale à 140 Hz. En sortie de cette opération de prétraitement, le signal noté $s(n)$, est utilisé comme entrée de tous les blocs successifs du l'encodeur. L'encodeur G.729 opère sur des

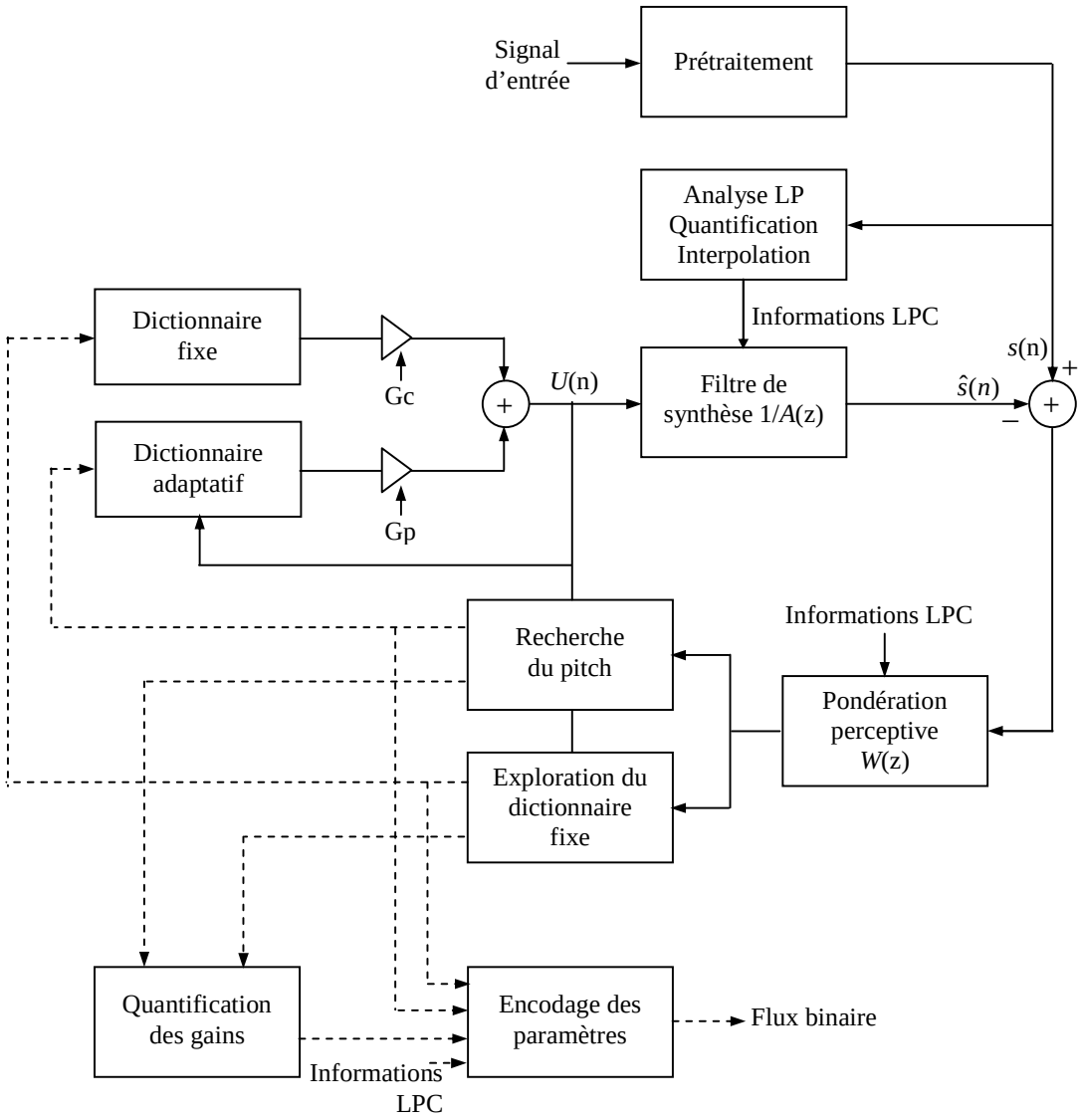


Figure 2.8: Principe du l'encodeur G.729 (CS-ACELP) [39].

Tableau 2.1: Allocation des bits pour le codeur G.729

Paramètres	Nombre de bits Sous-trame n°1	Sous-trame n°2	Transmission par Trame
Indice LPC			18
Indice du dictionnaire adaptatif (période du pitch)	8	5	13
Bit de parité pour la période du pitch	1	0	1
Indice du dictionnaire fixe	13	13	26
Signe de la contribution du dictionnaire fixe	4	4	8
Gains des contributions fixe et adaptative	7	7	14
Total			80

trames de parole de 10 ms correspondant à 80 échantillons pour une fréquence d'échantillonnage de 8 kHz. Le signal de parole est analysé à chaque trame pour extraire les coefficients du filtre de Prédiction Linéaire : $A(z) = 1 + \sum_{k=1}^{10} a_k z^{-k}$ du dixième ordre utilisant l'algorithme de Levinson-Durbin (Appendice A). Le filtre perceptif est basé sur les coefficients de prédiction $\{a_i\}_{i=1,..,p}$ (non quantifiés) et est donné par : $W(z) = A(z/\gamma_1)/A(z/\gamma_2)$ où les valeurs de γ_1 et γ_2 sont fonctions de la forme spectrale du signal d'entrée (spectre plat ou non) [39]. Les coefficients de prédiction sont convertis en fréquences de raies spectrales (LSF) et codés sur 18 bits. Par la suite, les paramètres d'excitation, tels que les indices ainsi que les gains des dictionnaires fixe et adaptatif, sont estimés sur la base de sous-trames de 40 échantillons, soit de 5 ms [38, 39]. L'encodeur G.729 code le signal d'excitation en recherchant dans deux dictionnaires (adaptative et fixe) les formes d'onde qui minimisent l'erreur entre les signaux de parole original et reconstruit en tenant compte d'une pondération perceptive qui améliore la qualité de restitution de la parole. Les formes d'onde étant normalisées, un gain leur est associé de manière à modéliser au mieux les séquences du signal d'excitation. Cette méthode de recherche est dite en boucle fermée ou encore analyse par synthèse.

Le tableau 2.1 donne le schéma d'allocation des bits pour les paramètres issus de l'encodeur G.729. Ces paramètres correspondant aux informations nécessaires à la reconstitution d'une trame vocale de 10 ms. Donc un total de 80 bits sont transmis par trame de 10 ms. Il en résulte un débit binaire de 8000 bits/s.

2.5.2 Quantification des coefficients LSF

Pour la quantification des coefficients LSF, une prédiction MA (moyenne ajustée) commutée de 4ème ordre est utilisée pour prédire les coefficients LSF de la trame courante. La différence entre les coefficients de la trame *m* courante $\{\omega_i^{(m)}\}_{i=1,...,10}$ et les coefficients prédis est donnée par :

$$l_i = \left(\omega_i^{(m)} - \sum_{k=1}^{4} \hat{P}_{i,k} \hat{l}_i^{(m-k)} \right) \Big/ \left(1 - \sum_{k=1}^{4} \hat{P}_{i,k} \right), \quad i = 1,...,10, \tag{2.6}$$

où $\hat{P}_{i,k}$, $1 \le k \le 4$, sont les coefficients du prédicteur MA commuté. Les valeurs initiales des coefficients $\{\hat{l}_i^{(k)}\}_{i=1,...,10}$ sont données par $\hat{l}_i^{(k)} = i\pi/11,\ i = 1,...,10$, pour $k < 0$.

Le vecteur $\{l_i\}_{i=1,...,10}$ est quantifié en utilisant une structure MSVQ à deux étages. Le premier étage est un quantificateur vectoriel de dimension 10 utilisant un dictionnaire $\boldsymbol{C}^{(1)} = \{\boldsymbol{c}_i^{(1)}\}_{i=0,...,127}$ avec 128 entrées (7 bits). Le deuxième étage est un quantificateur vectoriel structuré de 10-bit, qui est implémenté sous forme d'une structure SVQ utilisant deux dictionnaires $\boldsymbol{C}^{(2)} = \{\boldsymbol{c}_i^{(2)}\}_{i=0,...,31}$ et $\boldsymbol{C}^{(3)} = \{\boldsymbol{c}_i^{(3)}\}_{i=0,...,31}$ correspondant au cinq premier coefficients et aux cinq derniers coefficients respectivement. Donc les dictionnaires $\boldsymbol{C}^{(2)}$ et $\boldsymbol{C}^{(3)}$ sont de dimension 5 chacun et contenant 32 entrées (5 bits) chacun. Ainsi, un total de 18 bits par trame sont utilisés pour la quantification des coefficients LSF : 17 bits sont utilisés pour quantifier les vecteurs erreur-de-prédiction de dimension 10 et un bit est utilisé pour sélectionner un des deux prédicteurs MA possibles. La quantification du vecteur $\{l_i\}_{i=1,...,10}$ donne le vecteur quantifié $\{\hat{l}_i\}_{i=1,...,10}$ défini par :

$$\hat{l}_i = \begin{cases} c_{i_{o1},i}^{(1)} + c_{i_{o2},i}^{(2)} & i = 1,...,5 \\ c_{i_{o1},i}^{(1)} + c_{i_{o3},i-5}^{(3)} & i = 6,...,10 \end{cases}, \tag{2.7}$$

où i_{o1}, i_{o2} et i_{o3} sont les indices des vecteurs optimaux issus des dictionnaires $\boldsymbol{C}^{(1)}$, $\boldsymbol{C}^{(2)}$ et $\boldsymbol{C}^{(3)}$ respectivement. Les coefficients $\{\hat{l}_i\}_{i=1,..10}$ sont arrangés comme suit :

$$\begin{cases} \hat{l}_{i-1} = (\hat{l}_i + \hat{l}_{i-1} - J)/2 \\ \hat{l}_i = (\hat{l}_i + \hat{l}_{i-1} + J)/2 \end{cases}, \text{ si } \hat{l}_{i-1} > \hat{l}_i - J\ . \tag{2.8}$$

La procédure d'arrangement est appliquée deux fois avec les valeurs $J = 0.0012$ et $J = 0.0006$ respectivement. Les coefficients LSF quantifiés $\hat{\omega}_i^{(m)}$ de la trame courante *m* sont calculés en fonction des valeurs quantifiées précédentes $\hat{l}_i^{(m-k)}$ et la valeur quantifiée courante $\hat{l}_i^{(m)}$:

$$\hat{\omega}_i^{(m)} = \left(1 - \sum_{k=1}^{4} \hat{P}_{i,k}\right) \hat{l}_i^{(m)} + \sum_{k=1}^{4} \hat{P}_{i,k} \hat{l}_i^{(m-k)}, \quad i = 1,...,10 \tag{2.9}$$

Afin d'assurer la stabilité du filtre de synthèse, les fréquences de raies spectrales quantifiées sont ordonnées : $0.005 \le \hat{\omega}_1 < \hat{\omega}_2 < ... < \hat{\omega}_{10} \le 3.135$ et une distance minimale de 0.0391 entre deux coefficients adjacents est imposée.

La procédure de codage des paramètres LSF est décrite comme suit : Pour chacun des deux prédicteurs MA, la meilleure approximation $\{\hat{\omega}_i\}_{i=1,..,10}$ des coefficients $\{\omega_i\}_{i=1,..,10}$ de la trame courante est celle qui minimise la distance Euclidienne pondérée :

$$d(\boldsymbol{\omega}, \hat{\boldsymbol{\omega}}) = \sum_{i=1}^{10} w_i (\omega_i - \hat{\omega}_i)^2 , \tag{2.10}$$

où les poids $\{w_i\}_{i=1,...,10}$ sont calculés en fonction des coefficients LSF non quantifiées :

$$w_i = \begin{cases} 1.0 & \text{si } \omega_{i+1} - \omega_{i-1} - 1 > 0 \\ 10(\omega_{i+1} - \omega_{i-1} - 1)^2 + 1 & \text{autrement} \end{cases}, \quad i = 1,...,10 , \tag{2.11}$$

avec $\omega_0 = 0.04\pi$ et $\omega_{11} = 0.92\pi$, qui représentent les valeurs minimale et maximale respectivement. Le premier dictionnaire $\boldsymbol{C}^{(1)}$ est exploré et l'indice i_{o1} du vecteur qui minimise la distance Euclidienne (non pondérée) est retenu. Le deuxième dictionnaire $\boldsymbol{C}^{(2)}$, qui définit la partie inférieure du deuxième étage du quantificateur, est ensuite exploré. Pour chaque candidat possible, le vecteur partiel $\{\hat{\omega}_i\}_{i=1,..,5}$ est reconstruit en utilisant l'équation (2.9), et réarrangé pour garantir une distance minimale de 0.0012. La distance Euclidienne pondérée, équation (2.10), est calculée, et le vecteur d'indice i_{o2} produisant la distance minimale est retenu. Utilisant les vecteurs d'indice i_{o1} et i_{o2} sectionnés, respectivement, au premier étage et au deuxième étage (partie inferieure) du quantificateur, la partie supérieure du deuxième étage est recherchée au sein du dictionnaire $\boldsymbol{C}^{(3)}$ et comme précédemment une distance minimale de 0.0012 est imposée. Le vecteur d'indice i_{o3} minimisant la distance Euclidienne pondérée est retenu. Le vecteur résultant $\{\hat{l}_i\}_{i=1,...,10}$ est arrangé selon l'équation (2.8) avec $J = 0.0006$. Ce processus est effectué pour chacun des deux prédicteurs MA, et le prédicteur MA qui

produit la distance Euclidienne pondérée la plus faible est choisi. Comme expliqué précédemment le vecteur quantifié résultant $\{\hat{l}_i\}_{i=1,...,10}$ est réarrangé deux fois selon (2.8) et un contrôle de la stabilité du filtre de synthèse est appliqué (ordonnancement et imposition d'une distance minimale) pour produire les coefficients LSF quantifiés $\{\hat{\omega}_i\}_{i=1,...,10}$ [39].

Notons que les coefficients LSP de la trame courante sont utilisés pour la deuxième sous-trame (de la trame courante) alors qu'une interpolation entre les coefficients LSP (de la trame précédente et de la trame courante) est effectuée pour produire les coefficients LSP (quantifiés et non quantifiées) de la première sous-trame [39].

2.5.3 Dictionnaire adaptatif

Le nombre de périodes de pitch à examiner, en boucle fermée, est réduit en effectuant une procédure à deux étapes [39] :

• Une analyse en boucle ouverte (effectuée sur chaque trame de 10 ms) estime, dans un premier temps, la période de pitch optimale T_{op} en effectuant une recherche sur des valeurs entières favorisant les petites valeurs du pitch (pour éviter le choix d'un multiple du pitch).

• La recherche au sein du dictionnaire adaptatif (la recherche du pitch en boucle fermée) est effectuée pour chaque sous-trame de 5 ms. La complexité de la recherche en boucle fermée est réduite grâce à la réduction de l'intervalle des valeurs de pitch possible autour de la valeur optimale T_{op} obtenue en boucle ouverte. Pour la première sous-trame, le retard T_1 est trouvé en recherchant autour du pitch en boucle ouverte T_{op}. Pour la seconde sous-trame, la recherche de pitch en boucle fermée est faite sur 9 délais entiers autour de la valeur $\text{int}(T_1)$. où $\text{int}(T_1)$ est la partie entière du pitch T_1 de la première sous-trame. Les délais entiers de T_1 et T_2, sont affinés en faisant intervenir des délais fractionnaires (autour de la valeur entière déjà trouvée).

2.5.4 Dictionnaire fixe

Le dictionnaire fixe est basé sur un dictionnaire à structure algébrique où chaque vecteur de dimension $L = 40$ échantillons contient quatre impulsions signées, chacune d'elles pouvant occuper 8 à 16 positions possibles. La structure du dictionnaire fixe est donnée au tableau 2.2. Le vecteur issu du dictionnaire est construit en prenant un vecteur nul de dimension $L = 40$, et en plaçant les quatre impulsions unité aux positions $\{m_k\}_{k=0,...,3}$, multipliés avec leur signe correspondant [39] :

$$c(n) = s_0\delta(n-m_0) + s_1\delta(n-m_1) + s_2\delta(n-m_2) + s_3\delta(n-m_3), \quad n = 0,...,L-1, \tag{2.14}$$

où $\delta(n)$ est l'impulsion unité. Pour une valeur de pitch entière *T* inférieure à la longueur de la sous-trame $L = 40$, la périodicité du signal est mise en valeur en modifiant le vecteur $[c(0),...,c(L-1)]$ comme suite [39] :

$$c(n) = \begin{cases} c(n) & n = 0,...,T-1 \\ c(n) + \beta c(n-T) & n = T,...,L-1 \end{cases}, \tag{2.15}$$

où $0.2 \le \beta \le 0.8$ est le gain optimal de la contribution du dictionnaire adaptatif de la sous-trame précédente.

Tableau 2..2: Structure du dictionnaire fixe

Impulsion	Signe	Positions
i_0	$s_0 : \pm 1$	$m_0 : 0, 5, 10, 15, 20, 25, 30, 35$
i_1	$s_1 : \pm 1$	$m_1 : 1, 6, 11, 16, 21, 26, 31, 36$
i_2	$s_2 : \pm 1$	$m_2 : 2, 7, 12, 17, 22, 27, 32, 37$
i_3	$s_3 : \pm 1$	$m_3 : 3, 8, 13, 18, 23, 28, 33, 38$ $4, 9, 14, 19, 24, 29, 34, 39$

2.5.5 Le décodeur

Le décodeur dispose des mêmes dictionnaires fixe et adaptatif que l'encodeur. Les fonctions à effectuer ont déjà été rencontrées à l'encodeur : Les coefficients LSP sont interpolés et reconverties en coefficients de prédiction linéaire pour chaque sous-trame de 5 millisecondes. L'excitation est construite par combinaison des vecteurs-code adaptatif

et fixe, multipliés par leur gain respectif. Le signal vocal est reconstitué par filtrage de l'excitation à travers le filtre de synthèse LP. La qualité du signal de parole est ensuite améliorée à l'aide d'un bloc de post-traitement [39].

2.6 Le codeur FS-MELP

L'utilisation d'un codeur CELP pour le codage de la parole tend à donner de bons résultats pour des débits assez bas, compris entre 4 et 16 kbits/s. Au-dessous de 4 kbits/s, la qualité se dégrade brusquement. Pour des débits inférieurs à 4 kbits/s, les codeurs paramétriques sont plus efficaces. Le codeur à prédiction linéaire à excitation mixte MELP (Mixed Excitation Linear Prediction) a été développé à l'origine par McCree [50, 51]. Il est devenu officiellement la nouvelle norme fédérale FS-MELP, fonctionnant à 2400 bits/s, en 1997 [18, 50, 51]. Le codeur MELP a été développé pour surmonter certaines limitations du codeur LPC (figure 2.4). Une des limitations fondamentales du codeur LPC est la classification stricte d'une trame de parole en deux classes : voisé et non voisé. Le codeur MELP utilise un modèle de production de la parole plus sophistiqué. Le schéma fonctionnel du modèle MELP est donné à la figure 2.9 [18].

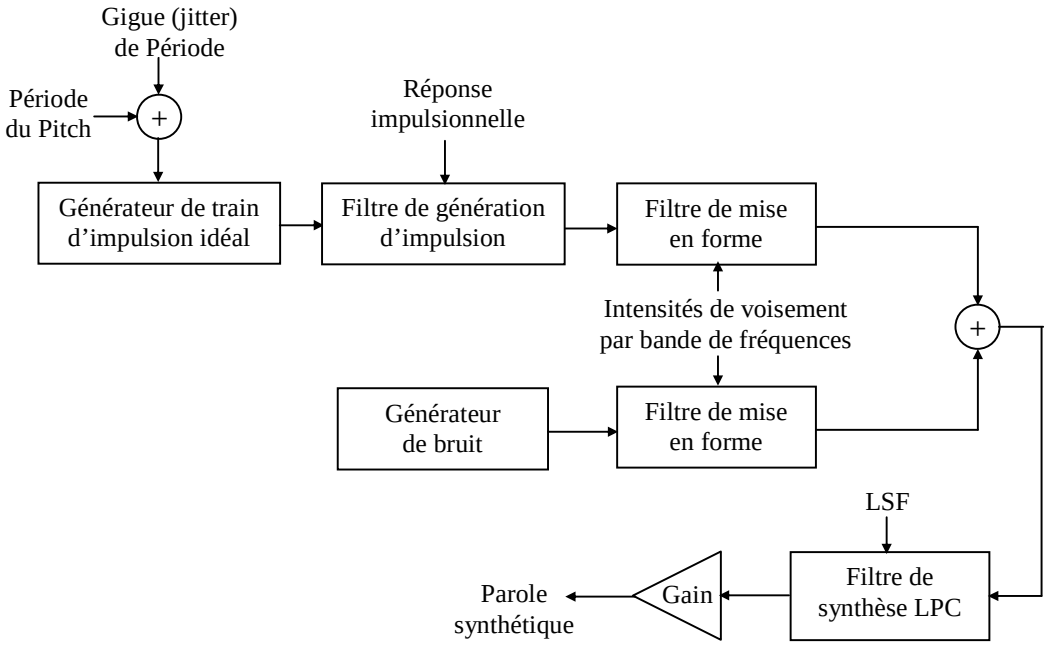


Figure 2.9: Le modèle MELP de production de la parole. Le modèle MELP utilise une excitation mixte formée de la somme d'une composante impulsionnelle et d'une composante bruit. Cette excitation est une excitation multi-bande avec une intensité de voisement définie pour chaque bande de fréquence.

Le modèle MELP utilise une excitation mixte formée de la somme d'une composante impulsionnelle et d'une composante bruit. La composante impulsionnelle est formée d'un train d'impulsions périodique ou apériodique commandé par la gigue de période (pour introduire une fluctuation de la période du pitch), qui est un nombre aléatoire uniformément distribué entre $\pm 25\%$ de la période du pitch. Cette excitation est une excitation multi-bande avec une intensité de voisement (qui mesure la quantité de voisement) définie pour chaque bande de fréquence. Ceci est l'idée principale du codeur MELP qui est basée sur des observations pratiques où le signal résiduel (erreur de prédiction) est une combinaison d'un train d'impulsion avec du bruit. Ainsi, le modèle MELP est beaucoup plus réaliste que le modèle LPC10E (FS1015), où l'excitation est soit un train d'impulsions ou du bruit (figure 2.4). Les opérations effectuées au niveau du l'encodeur et du décodeur sont décrites brièvement ci-dessous.

2.6.1 L'encodeur

L'encodeur MELP fait une première estimation de la fréquence fondamentale. Puis cinq filtres d'analyse (avec les bandes passantes définies par 0-500, 500-1000, 1000-2000, 2000-3000, et 3000-4000 Hertz) sont utilisés pour séparer le signal d'entrée en cinq bandes de fréquence adjacentes et l'intensité de voisement est calculée dans chacune des cinq bandes. L'intensité de voisement est déterminée dans chaque bande par la valeur de l'autocorrélation normalisée [18]. Dans la norme, cette intensité est codée sur 1 bit, chaque bande est donc classée voisée ou non voisée. Après analyse, l'encodeur peut positionner un drapeau appelé drapeau d'apériodicité (*aperiodic flag*) pour indiquer au décodeur que la composante impulsionnelle doit être apériodique. L'encodeur effectue par ailleurs une analyse spectrale par prédiction linéaire et les amplitudes de la transformée de Fourier Discret DFT (Discrete Fourier transform) du signal erreur de prédiction sont calculées ; des valeurs crêtes du spectre correspondant aux 10 premières harmoniques liées à la fréquence du pitch sont mesurées. Ces valeurs maximales sont appelées les amplitudes de Fourier et sont transmises au décodeur et seront utilisées pour construire la réponse impulsionnelle du filtre générateur d'impulsions (figure 2.9). Le tableau 2.3 donne le schéma d'allocation des bits pour les paramètres transmis par l'encodeur MELP [18]. Les 10 coefficients LPC (tableau 2.3) sont transformés en 10

paramètres LSF et sont quantifiés en utilisant un quantificateur vectoriel MSVQ à quatre étages de résolutions 7, 6, 6 et 6 bits respectivement. La période du pitch finale et l'intensité de voisement de la première bande (la bande la plus basse) sont quantifiés conjointement en utilisant 7 bits. Chaque intensité de voisement est codée sur 1 bit. Deux gains (par trame) correspondant aux énergies de 2 demi-trames sont calculés et codés sur 3 et 5 bits. Les 10 amplitudes de Fourier sont normalisées et quantifiées utilisant un quantificateur vectoriel de résolution 8 bits (un dictionnaire de 256 vecteurs). La protection contre les erreurs de canal est assurée pour les trames non voisées seulement, en utilisant 13 bits. Un total de 54 bits est transmis par trame de 22.5 ms (180 échantillons pour une fréquence d'échantillonnage de 8 kHz). Il en résulte un débit binaire de 2400 bits/s. La plupart des paramètres transmis sont interpolés linéairement pendant la synthèse (au niveau du décodeur) de la parole.

Tableau 2.3: Allocation de bits pour le codeur MELP

Paramètres	Voisée	Non voisée
Coefficients LPC	25	25
Période du pitch/intensité de voisement (bande-basse)	7	7
Intensités de voisement	4	---
Premier gain	3	3
Deuxième gain	5	5
Drapeau d'apériodicité	1	---
Amplitudes de Fourier	8	---
Synchronisation	1	1
Protection contre les erreurs	---	13
Total	54	54

2.6.2 Le décodeur

Après décodage des paramètres reçus, le décodeur interpole les différents paramètres de manière synchrone au pitch. Pour les trames non voisées (détectables à partir du code de la période du pitch/intensité de voisement de la bande la plus basse), des valeurs par défaut pour certains paramètres sont utilisés : 50 pour la période du pitch, 0.25 pour la gigue, les amplitudes de Fourier sont initialisées à 1, et les intensités de voisement sont mises à 0. Les valeurs par défaut sont nécessaires pour les trames non voisées puisque l'interpolation linéaire est effectuée de manière synchrone au pitch [18].

Le signal d'excitation est synthétisé une fois par période de pitch. Chaque impulsion (de l'excitation impulsionnelle) est obtenue sur une période de pitch et possède une

certaine forme, différente d'une impulsion idéale. La forme de l'impulsion est capturée par les amplitudes de Fourier. Les amplitudes de Fourier sont utilisées pour générer la réponse impulsionnelle du filtre de génération d'impulsion (figure 2.9), responsable de la synthèse de l'excitation impulsionnelle. Lorsque l'indicateur d'apériodicité est positionné, une gigue est appliquée à la valeur de la période fondamentale. La valeur de la gigue est assignée comme suite : $\text{gigue} = 0.25$ si le drapeau d'apériodicité est égal à 1 ; autrement, $\text{gigue} = 0$. Cette possibilité d'excitation impulsionnelle non périodique est particulièrement intéressante pour les zones de transitions entre sons.

Le décodeur MELP se sert de deux filtres de mise en forme pour combiner l'excitation impulsionnelle avec l'excitation bruit pour former le signal d'excitation mixte. Chaque filtre de mise en forme se compose de cinq filtres, appelé les filtres de synthèse, puisqu'ils sont utilisés pour synthétiser le signal d'excitation mixte. Chaque filtre de synthèse commande une bande de fréquence particulière, avec les bandes passantes définies par 0-500, 500-1000, 1000-2000, 2000-3000, et 3000-4000 Hz. Les réponses de ces cinq filtres sont commandées par les cinq intensités de voisement. En variant les intensités de voisement avec le temps, une paire de filtres variant dans le temps en résulte. Le filtrage appliqué à la composante impulsionnelle a pour réponse impulsionnelle la somme de toutes les réponses impulsionnelles des cinq filtres passe-bande pour les bandes voisées. Le filtrage de la composante bruit est déterminé de la même façon à partir des bandes non voisées. En notant les réponses impulsionnelles des cinq filtres de synthèse par $h_i(n)$, $i = 1 \text{ à } 5$, respectivement, la réponse globale du filtre de mise en forme de l'excitation impulsionnelle est donnée par : $\sum_{i=1}^{5} vs_i h_i(n)$, où $0 \le vs_i \le 1$ ($i = 1 \text{ à } 5$) sont les intensités de voisement. La réponse globale du filtre de mise en forme de l'excitation bruit est donnée par : $\sum_{i=1}^{5} (1 - vs_i) h_i(n)$. L'excitation mixte est ensuite filtrée par le filtre de synthèse LPC. Le signal synthétique résultant est mis à l'échelle en fonction de l'énergie de la trame originale. Notons que dans la norme MELP, deux filtres additionnels sont ajoutés le long de la chaine de traitement : le filtre d'amélioration spectral prenant l'excitation mixte comme entrée, et le filtre de dispersion d'impulsions à

l'extrémité de la chaîne de traitement. Ces filtres améliorent la qualité perceptive de la parole synthétique [18].

2.7 Codage échelonnable de la parole

Le besoin de codeurs échelonnables est dû, en grande partie, au déploiement de la transmission de la parole à travers les réseaux à paquets (l'internet). Pour les codeurs de parole produisant un seul type de flux-binaire à débit fixe, les données transmises pourraient être perdues si le trafic dans le canal de transmission est congestionné. Pour gérer la congestion, une solution consiste à organiser le flux-binaire en différentes parties avec divers degrés d'importance. La partie la plus importante est désignée par la couche noyau (ou couche de base), et les parties les moins importantes sont désignées par les couches d'amélioration (figure 2.10).

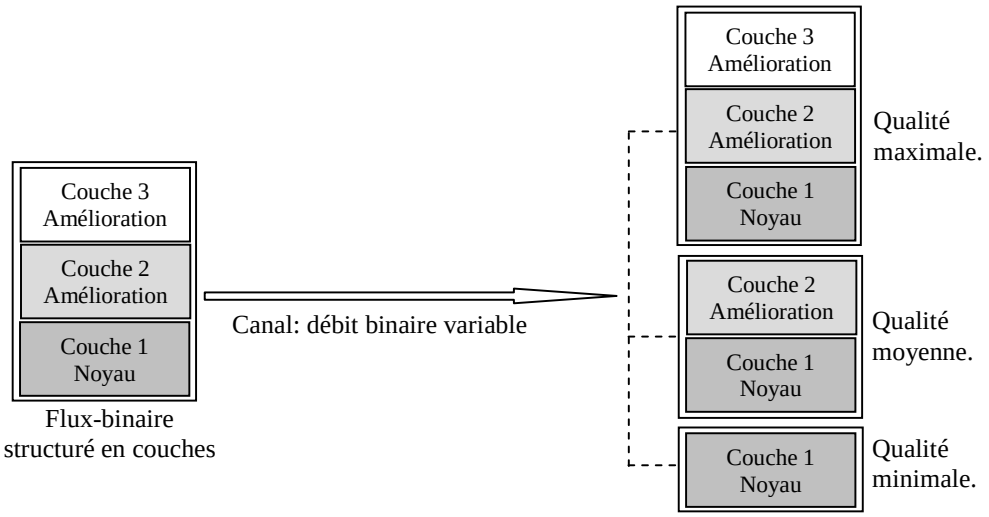


Figure 2.10: Principe du codage échelonnable où le flux-binaire est structuré en couches : une couche noyau et plusieurs couches d'amélioration. Selon le nombre de couches reçus par le décodeur, différentes qualités de la parole synthétisée peuvent être produites au niveau du décodeur.

La couche noyau fournit la qualité de base. Cette qualité peut être améliorée par les couches d'amélioration qui peuvent être décodées comme des extensions à la couche noyau. Lorsque le canal de transmission est congestionné, les couches d'amélioration peuvent être rejetées une couche à la fois sans que la qualité de la parole décodée ne subisse une dégradation importante. Le débit binaire peut être ajusté à la volée par la troncation du flux-binaire incorporé à un point quelconque de la chaîne de

communication telle que les routeurs. Grâce à cette adaptation très flexible du débit binaire, différentes versions de la parole peuvent être reconstruites, à de divers niveaux de qualité, au niveau du décodeur en fonction des contraintes du réseau de transmission et la suppression des paquets, qui altère sévèrement la qualité de la parole décodée, est ainsi évitée [52, 53].

Pour le codage de parole, l'échelonnabilité est généralement réalisée de deux manières, on parle alors : d'échelonnabilité SNR (SNR scalability) et d'échelonnabilité de bande élargie (bandwidth scalability) [53]. Les codeurs à échelonnabilité SNR peuvent être divisés en deux types : Les codeurs à échelonnabilité SNR dépendant (Dependent SNR scalable coders) et les codeurs à échelonnabilité SNR indépendant (Independent SNR scalable coders). Les codeurs à échelonnabilité SNR dépendant utilisent un seul type de codage et sont désignés de telle manière que la couche d'amélioration est intimement liée à la couche noyau. Les codeurs à échelonnabilité SNR indépendant sont désignés tel que la couche d'amélioration soit liée le moins possible à la couche noyau.

2.7.1 Codeurs à échelonnabilité SNR

Dans les codeurs à échelonnabilité SNR dépendant, la couche d'amélioration peut utiliser l'information de la couche noyau de sorte que le débit binaire (pour la couche d'amélioration) puisse être réduit au minimum et réaliser en même temps une bonne performance. Un des premiers codeurs échelonnable de ce type est le codeur MICDA imbriqué opérant à 5, 4, 3 et 2 bits par échantillon (40, 32, 24 et 16 kbit/s). Ce codeur utilise une quantification scalaire incorporée (imbriquée) où les niveaux de décision des quantificateurs des plus faibles débits sont des sous-ensembles de ceux du quantificateur du débit le plus élevé. Le codeur MICDA imbriqué a été standardisé en 1990 comme la norme ITU-T G.727 [54]. La figure 2.11 donne un exemple d'un quantificateur scalaire uniforme imbriqué de 16 niveaux. L'idée clé est que les quantificateurs de 8 et 4 niveaux ont des niveaux de décision qui sont des sous-ensembles de ceux du quantificateur de 16 niveaux. Ainsi avec un code de 4 bits associé au quantificateur de 16 niveaux, si le bit de plus faible poids est écarté on obtient un code de 3 bits associé au quantificateur de 8 niveaux. Si deux bits de plus faibles poids sont écartés on obtient un code de 2 bits (du quantificateur de 4 niveaux).

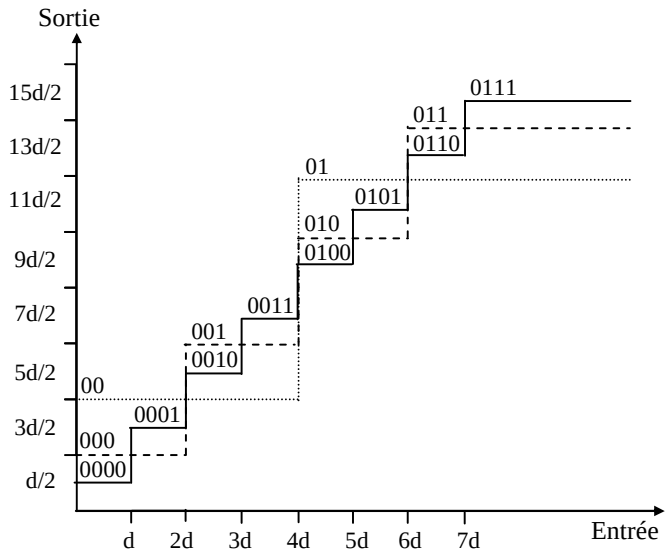
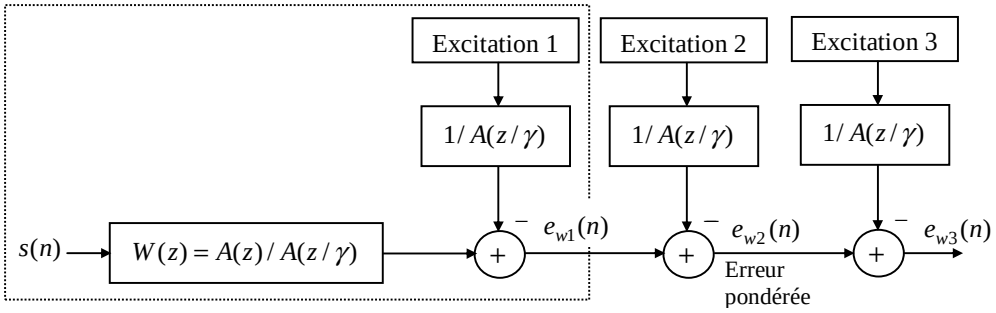


Figure 2.11: Quantification scalaire uniforme incorporée. Les codes des sorties sont représentés avec un bit de signe (le bit de plus fort poids).

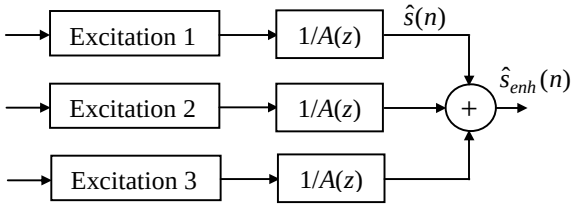
La quantification vectorielle multi-étages (Chapitre 1) produit une structure naturelle pour une quantification vectorielle incorporée. Ainsi pour une structure MSVQ à deux étages (avec un débit de M_1 et M_2 bits/vecteur respectivement), si le code du vecteur-code du deuxième étage est écarté, seul le code du vecteur-code du premier étage (de M_1 bits/vecteur) est utilisé au décodeur. En [55], une quantification MSVQ de 4 étages est utilisée pour le codage de la parole de 32 à 8 kbits/s avec un pas de 8 kbits/s.

La technique CELP constitue la base de beaucoup de codeurs de parole à bas débit. Ainsi, différents codeurs CELP échelonnables ont été proposés [56-60]. Dans [59] le signal d'excitation du codeur CELP est divisé en trois contributions produisant un codeur CELP échelonnable à des débits de 6.4, 8 et 9.6 kbits/s. De même, dans [60] un codeur incorporé excité par des impulsions régulièrement espacées ERPE (embedded RPE) est proposé. Ce codeur est basé sur un codage RPE, sans prédiction du pitch, où le signal d'excitation est divisé en trois contributions. La figure 2.12 peut donc être interprétée comme un codage RPE à trois étages, où chaque étage utilise les paramètres de prédiction linéaire obtenus à partir du signal de parole original $s(n)$. Le deuxième étage et les étages suivants codent le signal d'erreur de l'étage précédent. Les excitations codées et transmises du deuxième étage et des étages suivants peuvent être écartées par ordre

d'importance (de la moins importante à la plus importante) si le canal de transmission est congestionné, permettant ainsi une réduction progressive du débit de transmission. La première contribution de l'excitation produit le débit de 6.4 kbits/s (correspondant à la couche noyau) ; l'addition de la deuxième et de la troisième contribution produisent les débits de 10.6 et 14.8 Kbits/s (correspondant aux couches d'améliorations).



a) Encodeur



b) Décodeur

Figure 2.12: Schéma fonctionnel du l'encodeur et du décodeur RPE incorporé, sans prédiction du pitch, basé sur une structure à trois étages (le signal d'excitation est divisé en trois contributions).

Au décodeur, le signal de parole reconstruit est obtenu en additionnant les signaux résultants du passage des excitations décodées (reçues) à travers le filtre de synthèse. La qualité du signal de parole reconstruite est maximale lorsque les trois excitations sont reçues et dans ce cas le signal reconstruit correspond à $\hat{s}_{enh}(n)$ (figure 2.12). Notons qu'au débit le plus élevé, un codeur échelonnable de parole est moins performant qu'un codeur non-échelonnable de même débit.

Les codeurs à échelonnabilité SNR indépendant prennent avantage des codeurs standards existants qui sont, généralement, utilisés pour produire la couche noyau du flux-binaire. Le schéma de codage des couches d'amélioration n'est pas nécessairement lié au codeur de la couche noyau. Le codage indépendant offre donc plus de flexibilité au détriment du débit/qualité et une augmentation de la complexité. La figure 2.13 montre le schéma de principe d'un codeur à échelonnabilité SNR indépendant à deux étages.

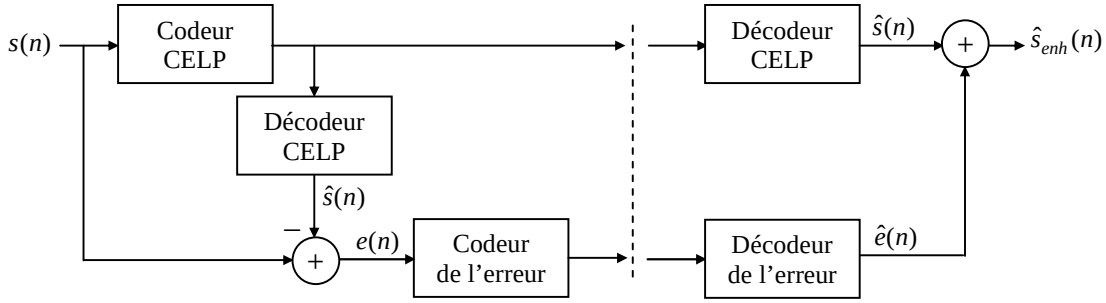


Figure 2.13: Schéma fonctionnel d'un codeur échelonnable basé sur une structure à deux étages.

Le premier étage, qui est un codeur CELP (dans l'exemple de la figure 2.13), code le signal de parole original $s(n)$ et produit le signal synthétique $\hat{s}(n)$. Le premier étage utilise, généralement, un codeur standard et fournit la couche noyau du flux-binaire. Le signal erreur : $e(n) = s(n) - \hat{s}(n)$ est codé par le deuxième étage et fourni la couche d'amélioration du flux-binaire. Le codeur du deuxième étage n'est pas nécessairement de même type que celui du premier étage. Dans [53] un codeur échelonnable appelé CELPTree est conçu par combinaison de deux techniques de codage : le codage CELP (comme premier étage) et le codage en arbre (Tree coding) comme deuxième étage.

2.7.2 Codeurs à échelonnabilité de bande élargie

L'échelonnabilité de bande élargie [53, 61] utilise une structure en sous bandes. Le signal de parole à large-bande (échantillonné à 16 kHz) est décomposé, en général, en deux signaux de sous-bande. Le signal de parole de bande-inférieure est codé en utilisant un codeur noyau (généralement de type CELP) et fournit le signal de sortie de base. Le signal de parole de bande-supérieure est codé en utilisant un autre codeur (de type CELP ou de type paramétrique) comme une couche d'extension de la largeur de bande. Le signal de parole de sortie peut être à bande-étroite si seule la couche noyau est utilisée ou à large-bande si les deux couches (noyau et amélioration) sont utilisées. La figure 2.14 donne le schéma de principe d'un codeur échelonnable où le codeur de la couche noyau est de type CELP. L'échelonnabilité de bande élargie peut être implémenté d'une autre manière : Le signal à bande-étroite, obtenu par sous-échantillonnage du signal large-bande, est codé par le codeur noyau. Alors que le signal large-bande est codé par le codeur d'amélioration qui utilise les informations issues du codeur noyau.

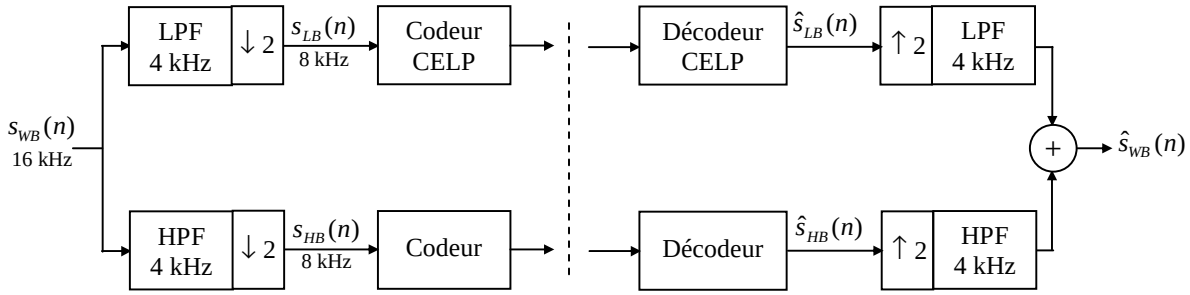


Figure 2.14: Schéma fonctionnel d'un codeur échelonnable de bande élargie.
Le codeur noyau est un codeur de type CELP.

La norme MPEG-4 propose un ensemble d'outils pour le codage des sons naturels (tels que la parole et la musique) [53, 62-64]. Le codage CELP est recommandé pour le codage de la parole (en bande étroite ou en bande élargie) entre 4 et 24 kbps. L'échelonnabilité SNR indépendant est utilisée pour le signal à bande étroite : un codeur CELP est utilisé pour coder le signal d'entrée à un débit prédéterminé (entre 4 et 12 kbits/s) et fournir ainsi la couche noyau. La couche d'amélioration consiste à coder le signal d'erreur en utilisant un système d'analyse par synthèse avec une excitation multi-impulsionnelle. Jusqu'à trois couches, de 2 kbits/s chacune, peuvent être ajoutées. L'échelonnabilité par extension de largeur de bande est réalisée en utilisant, pour la couche d'amélioration, un codeur CELP pour coder le signal d'entrée à 16 kHz. Ce codeur est similaire à celui de la couche noyau qui est utilisé pour le codage du signal à 8 kHz (obtenu par sous-échantillonnage du signal large-bande). La différence est que les coefficients LSP, le délai du pitch et le signal d'excitation sont codées en utilisant ceux du codeur CELP de la couche noyau. Pour des débits plus bas, le codage de la parole est pris en charge par le codeur HVXC (Harmonic Vector Excitation Coding). Ce codeur est échelonnable à des débits de 2 kbits/s et 4 kbits/s [31, 64].

La norme ITU-T G.729.1 est un codeur échelonnable récemment normalisé pour les applications de téléphonie à large bande et la voix sur IP (VoIP) [65]. On décrit le principe de fonctionnement de ce codeur qui utilise les deux types d'échelonnabilité cités plus haut.

2.8 Le codeur échelonnable ITU-T G.729.1

Le codeur G.729.1 est un codeur large bande échelonnable de 8 à 32 kbit/s interopérable, à 8 kbit/s, avec la norme UIT-T G.729. Le codeur G.729.1 est conçu pour les applications de téléphonie à large bande (50 à 7000 Hz) et la voix sur IP (VoIP) [65]. Le codeur produit un flux binaire incorporé structuré en 12 couches correspondant aux 12 débits binaires disponibles de 8 à 32 kbits/s. La couche 1 est la couche noyau et correspond au débit de 8 kbit/s. La couche 2 est une couche d'amélioration de bande étroite qui ajoute 4 kbit/s, alors que les couches 3 à 12 sont des couches d'amélioration de large bande qui ajoutent 20 kbit/s par pas de 2 kbit/s. Ce codeur est conçu pour fonctionner avec un signal numérique échantillonné à 16 kHz et 8 kHz. Le codeur G.729.1 est bâtit selon une structure à trois étages : la prédiction linéaire avec excitation par code CELP incorporé de la bande inférieure (50-4000 Hz), le codage paramétrique de la bande supérieure (4000-7000 Hz) par extension de largeur de bande en domaine temporel TDBWE (time-domain bandwidth extension), et l'amélioration de la bande complète (50-7000Hz) par une technique de codage connue sous le nom d'annulation de crénelage de domaine temporel TDAC (time-domain aliasing cancellation). L'étage CELP incorporé génère les couches 1 et 2 qui donnent une synthèse de bande étroite (50 à 4000 Hz) à 8 et 12 kbit/s. L'étage TDBWE génère la couche 3 et permet de produire une sortie en large bande (50 à 7000 Hz) à 14 kbit/s. L'étage TDAC fonctionne dans le domaine de la transformée en cosinus discret modifié MDCT (modified discrete cosine transform) et génère les couches 4 à 12 pour améliorer la qualité de 16 à 32 kbit/s. Le codage TDAC représente à la fois le signal d'erreur de codage CELP pondéré dans la bande 50-4000 Hz et le signal d'entrée dans la bande 4000-7000 Hz.

2.8.1 L’encodeur

La figure 2.15 donne le diagramme fonctionnel de l’encodeur G.729.1. Par défaut, le signal d'entrée $s_{WB}(n)$ est échantillonné à 16 kHz. L’encodeur fonctionne sur des trames de 20 ms (320 échantillons). Le signal d'entrée $s_{WB}(n)$ est partagé en deux sous-bandes en utilisant un banc de filtrage QMF (*quadrature mirror filterbank*) $H_1(z)$ et $H_2(z)$.

Le signal de bande inférieure $s_{LB}(n) = s(n)$ est obtenu après décimation et prétraitement par un filtre passe-haut $H_{h1}(z)$ avec une fréquence de coupure de 50 Hz. Ce signal est codé par l'encodeur CELP incorporé de bande étroite à 8-12 kbit/s. La couche d'amélioration à 12 kbit/s s'appuie sur un dictionnaire fixe supplémentaire qui, lorsqu'il est combiné à l'excitation de la couche noyau à 8 kbit/s, offre un signal d'excitation plus riche. Les paramètres du dictionnaire fixe supplémentaire sont transmis au décodeur comme la couche 2 du flux-binaire.

La différence $d_{LB}(n)$ entre le signal $s(n)$ et le signal synthétique $s_{enh}(n)$ de l'encodeur CELP à 12 kbit/s est traitée par le filtre de pondération perceptif $W_{LB}(n)$. La différence pondérée $d_{LB}^{w}(n)$ est alors transformée dans le domaine fréquentiel par transformée en cosinus discrète modifiée (MDCT).

Le signal d'entrée de bande supérieure $s_{HB}(n)$ est obtenu après décimation, repliement spectral par $(-1)^n$ et prétraitement par un filtre passe-bas $H_{h2}(z)$ avec une fréquence de coupure de 3000 Hz. Le signal $s_{HB}(n)$ est codé par l'encodeur d'extension de bande passante de domaine temporel TDBWE. L'encodeur TDBWE extrait une description paramétrique du signal $s_{HB}(n)$. Cette description paramétrique comporte les paramètres : enveloppe temporelle et enveloppe fréquentielle [65]. Les enveloppes temporelle et fréquentielle sont transmis comme la couche 3 du flux-binaire. Le signal de bande supérieure $s_{HB}(n)$ est aussi transformé dans le domaine fréquentiel par MDCT. Les deux ensembles de coefficients MDCT, $D_{LB}^{w}(k)$ et $S_{HB}(k)$, sont finalement codés par l'encodeur d'annulation de crénelage de domaine temporel TDAC. Les spectres MDCT $\{D_{LB}^{w}(k)\}_{k=0,\ldots,159}$ et $\{S_{HB}(k)\}_{k=0,\ldots,159}$ des bandes inférieure et supérieure sont fusionnés en un spectre de pleine bande (0 - 7 kHz) : $\{Y(k)\}_{k=0,\ldots,319}$. Les coefficients de MDCT $Y(k)$ sont divisés en 18 sous-bandes où les 17 premières sous-bandes comprennent 16 coefficients MDCT (400 Hz), la dernière sous-bande comporte 8 coefficients (200 Hz). L'enveloppe spectrale est calculée comme la valeur quadratique moyenne (rms) dans le domaine logarithmique des 18 sous-bandes. Les sous-bandes sont triées par importance perceptive décroissante. Cet ordre est utilisé pour l'allocation des bits et le multiplexage

des indices des vecteurs résultant de la quantification vectorielle des coefficients MDCT de chaque sous-bande. Les bits associés au codage d'enveloppe spectrale de bande supérieure sont multiplexés avant les bits associés au codage d'enveloppe spectrale de bande inferieure. De plus, les bits associés au codage des sous-bandes sont multiplexés dans l'ordre d'importance perceptive décroissante : les sous-bandes qui sont perceptuellement plus importantes sont écrites en premier dans le flux binaire. Les enveloppes spectrales (de bandes inferieure et supérieure) ainsi que les coefficients de MDCT de chaque sous bande (la structure fine) sont transmis au décodeur comme les couches 4 à 12 du flux-binaire.

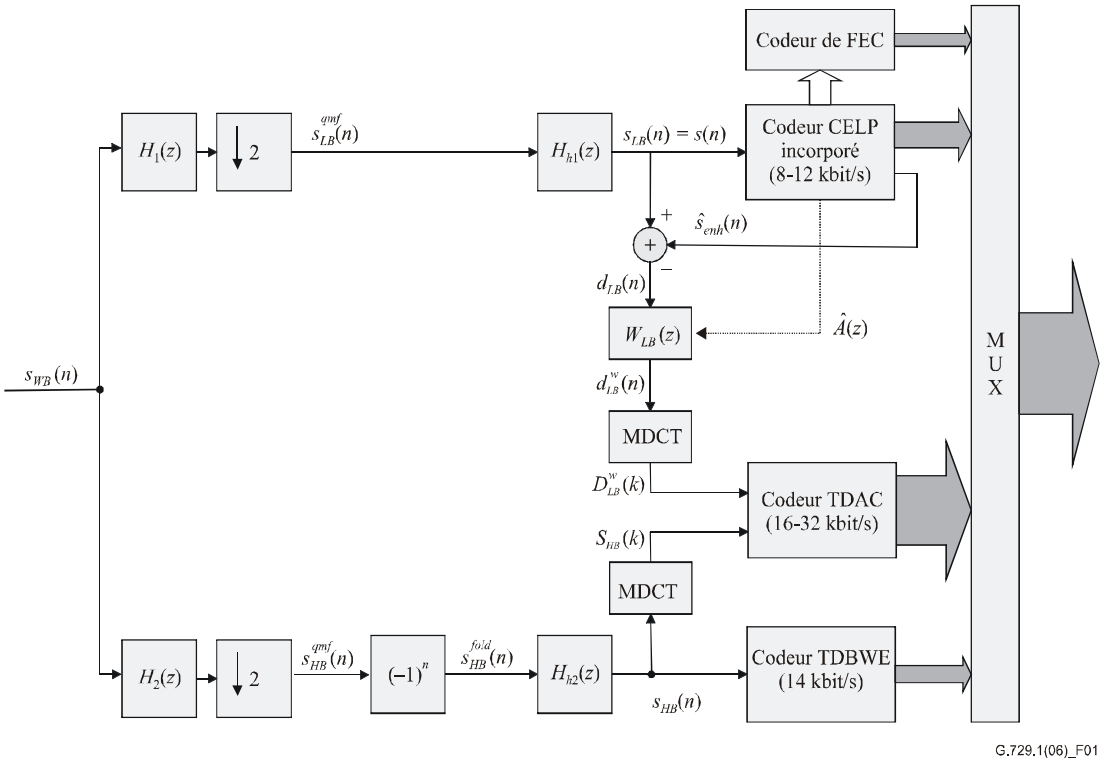


Figure 2.15: Diagramme fonctionnel du codeur G.729.1. Le cas spécifique du masquage d'effacement de trame n'est pas pris en compte dans ce diagramme [65]

De plus, certains paramètres sont transmis par l'encodeur de masquage d'effacement de trame FEC (frame erasure concealment) afin d'introduire une redondance dans le flux-binaire. Cette redondance permet d'améliorer la qualité en présence de trames perdues.

2.8.2 Le décodeur

La figure 2.16 donne le diagramme fonctionnel du décodeur. Le décodage dépend du nombre de couches reçues ou du débit binaire reçu équivalent. Si le débit binaire reçu est:

• 8 kbit/s (couche 1) et 12 kbit/s (couches 1 et 2) : le décodeur CELP incorporé produit le signal synthétique $\hat{s}_{LB}(n) = \hat{s}(n)$ ou $\hat{s}_{LB}(n) = \hat{s}_{enh}(n)$ selon que seule la première couche ou les deux premières couches sont reçues. À 12 kbit/s, l'excitation à 8 kbit/s et l'excitation de dictionnaire fixe à 12 kbit/s supplémentaire sont combinées et le signal d'excitation résultant est utilisé comme entrée pour le filtre de synthèse à prédiction linéaire pour la reconstruction du signal $\hat{s}_{LB}(n) = \hat{s}_{enh}(n)$. Le signal $\hat{s}_{LB}(n)$ est post-filtré et post-traité par un filtre passe-haut (HPF). Le banc de filtrage de synthèse QMF défini par les filtres $G_1(z)$ et $G_2(z)$ génère la sortie avec une synthèse de signale haute fréquence réglée à zéro.

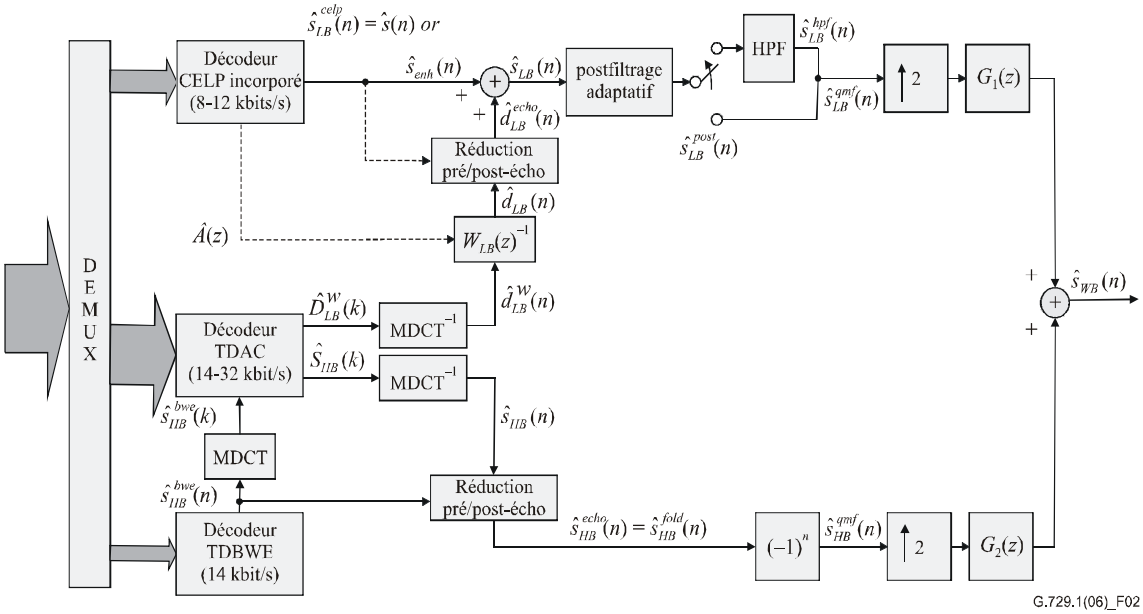


Figure 2.16: Diagramme fonctionnel du décodeur G.729.1 [65]

• 14 kbit/s (couches 1 à 3) : Les paramètres d'extension TDBWE reçus de la couche 3 du flux-binaire (les enveloppes temporelle et fréquentielle du signal de bande supérieure $s_{HB}(n)$ du côté du l'encodeur) sont utilisés pour mettre en forme un signal d'excitation

(formée par l'addition d'une contribution voisée et d'une contribution non voisée) généré sur la base des paramètres transmis dans les couches 1 et 2 du flux binaire (tel la période du pitch). Le formatage de l'enveloppe temporelle du signal d'excitation utilise les paramètres de l'enveloppe temporelle décodés. Le signal résultant est ensuite formaté fréquentiellement en utilisant un banc de filtre composé de 12 filtres (utilisant les 12 paramètres d'enveloppe fréquentielle décodés). Pour améliorer le signal résultant un post-traitement est effectué [65]. Le décodeur TDBWE produit donc un signal synthétique de haute fréquence $\hat{s}_{HB}^{bwe}(n)$ qui est transformé dans le domaine fréquentiel par MDCT de façon à annuler la bande de fréquence au-dessus de 3000 Hz dans le spectre de bande supérieure $\hat{S}_{HB}^{bwe}(k)$. Le spectre résultant $\hat{S}_{HB}(k)$ est transformé en domaine temporel par MDCT inverse avant le repliement spectral par $(-1)^n$. Le signal de bande supérieure reconstruit est combiné avec le signal de bande inférieure reconstruit à 12 kbit/s (décodage CELP sans filtrage passe-haut) utilisant un banc de filtrage de synthèse QMF.

• Au-dessus de 14 kbit/s (couches 1 à 4 et plus) : en plus du décodage CELP à bande étroite et du décodage TDBWE, le décodeur TDAC reconstruit les coefficients MDCT $\hat{D}_{LB}^{w}(k)$ et $\hat{S}_{HB}(k)$, qui correspondent à la différence pondérée reconstruite dans la bande inférieure (0-4 kHz) et au signal reconstruit dans la bande supérieure (4-7 kHz). Les sous-bandes (de bande supérieure) non reçues sont remplacées par les sous-bandes à niveau ajusté de $\hat{S}_{HB}^{bwe}(k)$. Les coefficients $\hat{D}_{LB}^{w}(k)$ et $\hat{S}_{HB}(k)$ sont tous deux transformés en domaine temporel par la transformée MDCT inverse. Le signal $\hat{d}_{LB}^{w}(n)$ de bande inférieure est alors traité par le filtre de pondération perceptif inverse $W_{LB}(z)^{-1}$. La synthèse de bande inférieure $\hat{s}_{LB}(n)$ est post-filtrée et la synthèse de bande supérieure $\hat{s}_{HB}^{fold}(n)$ est spectralement repliée. Les signaux $\hat{s}_{LB}^{qmf}(n)$ et $\hat{s}_{HB}^{qmf}(n)$ sont suréchantillonnés d'un facteur 2 et filtrés respectivement à travers les filtres $G_1(z)$ et $G_2(z)$. Les signaux résultants sont ajoutés pour obtenir le signal de sortie $\hat{s}_{WB}(n)$.

2.9 Conclusion

Les codeurs de type CELP sont des systèmes de codage mixtes qui utilisent à la fois des représentations paramétriques et temporelles du signal de parole. La technique CELP constitue la base de beaucoup de codeurs de parole normalisés fonctionnant à bas et moyens débits de 4 à 16 kbits/s. Pour des débits inférieurs à 4 kbits/s les codeurs de type paramétriques sont utilisés. L'échelonnabilité est généralement réalisée de deux manières; on parle alors d'échelonnabilité SNR et par bande élargie. Cependant plusieurs codeurs normalisés produisent des flux-binaires à débit fixe et ne sont donc pas échelonnables. Pour ces codeurs, si le trafic dans le canal de transmission est congestionné, les données transmises pourraient être perdues. Ainsi, pour éviter la suppression des paquets, qui altère sévèrement la qualité de la parole décodée, il est très souhaitable de modifier ces codeurs de telle manière à ce qu'ils produisent des flux-binaires incorporés rendant ainsi ces codeurs échelonnables. Cependant les modifications qui doivent être introduites dans ces codeurs (pour les rendre échelonnables) ne doivent pas altérer les performances de ces derniers aux débits pour lesquels ils ont été destinés à l'origine.

Chapitre 3

Conception d’un quantificateur vectoriel arborescent

3.1 Introduction

Pour plusieurs normes de codeurs de parole, l'encodeur produit un seul type de flux-binaire (bitstream) à un débit fixe [18]. Cependant, si le trafic dans le canal de transmission est congestionné, les données transmises (encodées) pourraient être perdues. Pour gérer la congestion, le flux-binaire est organisé en différentes parties avec divers degrés d'importance : une couche noyau et une ou plusieurs couches d’amélioration. Lorsque le canal de transmission est congestionné, les couches d’amélioration peuvent être rejetées une couche à la fois sans que la qualité finale de la parole décodée ne subisse une dégradation importante.

L’échelonnabilitée à granularité fine FGC (Fine Granularity Scalability) [57, 58, 66] est une approche où le flux binaire peut être rejeté avec une granularité plus fine, bit par bit dans le cas extrême au lieu d’une couche entière. La quantification vectorielle à structure arborescente TSVQ (tree-structured vector quantization) est un quantificateur incorporé qui est lié au concept de FGC [66]. Récemment Chu dans [66] a proposé un

algorithme d'élaboration de dictionnaires pour la conception d'un quantificateur arborescent à plusieurs étages (multi-étage) pour la quantification incorporée des paramètres LSF. Dans ce travail de recherche, nous proposons un algorithme efficace d'élaboration de dictionnaires basé sur la technique de fusion de cellules. L'arbre binaire est conçu à partir du niveau le plus élevé vers les niveaux inferieurs sous forme de connexions de sous-arbres optimaux selon un critère d'optimalité donné. L'algorithme proposé est généralisé pour la conception d'un quantificateur arborescent multi-étages.

3.2 Quantification vectorielle arborescente

On distingue entre la quantification vectorielle constituée d'une seule structure arborescente (constituée d'un seul étage) et la quantification vectorielle arborescente à plusieurs étages qui peut être vue comme une extension de la première forme [5, 15, 67].

3.2.1 Quantification vectorielle arborescente à un seul étage

Un quantificateur vectoriel à structure arborescente TSVQ (tree-structured vector quantization) ayant une structure binaire équilibrée de hauteur (de profondeur) L possède $L+1$ niveaux et correspond à un arbre ayant 2^l nœuds au niveau l avec $l = 0$ à L où $l = 0$ correspond au niveau racine et contient un seul nœud (nœud racine). Chaque nœud interne pointe vers deux nœuds enfants (nœuds descendants). Un arbre binaire équilibré de hauteur L se termine toujours avec 2^L nœuds-feuille (correspondants au niveau le plus élevé de l'arbre) [5, 15]. La figure 3.1-a donne un exemple d'un arbre binaire équilibré de hauteur $L = 4$. Cette structure binaire peut être représentée sous forme d'un ensemble de dictionnaires comme à la figure 3.1-b. Chaque nœud correspond à un vecteur-code et l'ensemble des vecteurs-code du niveau l de l'arbre correspond à un dictionnaire de taille $N = 2^m$ avec $m = l$ où m est la résolution (en bits/vecteur) du dictionnaire. Le dictionnaire de résolution m correspond, donc, à l'ensemble des vecteurs-code du niveau *m* de l'arbre et est noté par :

$$\boldsymbol{C}_m = \{\boldsymbol{c}_{m,i}, i = 0 \text{ à } 2^m - 1\}. \tag{3.1}$$

L'ensemble des vecteurs-code du niveau le plus élevé ($m = L$) de l'arbre correspond au dictionnaire-étage $\boldsymbol{C}_L$. L'ensemble des vecteurs-code du niveau m (avec $m < L$) est appelé dictionnaire-arbre $\boldsymbol{C}_m$. Donc pour un dictionnaire-étage de taille $N = 2^L$ correspond L (avec $L = \log_2 N$) dictionnaires-arbre de tailles $N/2$, $N/4$, …, 8, 4, 2 et 1.

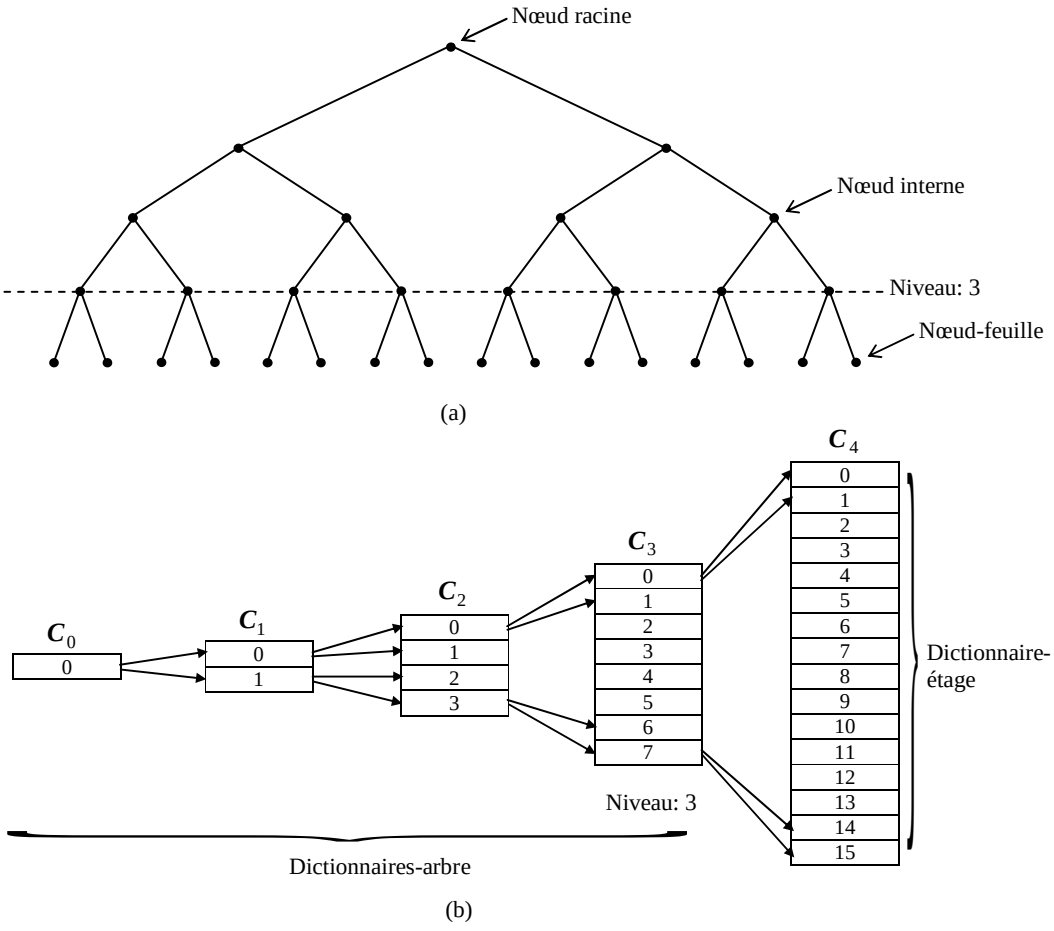


Figure 3.1 : (a) Un exemple d'un arbre binaire équilibré de hauteur (profondeur) $L = 4$. L'arbre est constitué de $L+1$ niveaux: du niveau $l = 0$ (niveau racine) contenant un seule nœud au niveau $l = 4$ contenant 16 nœuds-feuille (nœuds terminaux). Chaque nœud de l'arbre correspond à un vecteur-code. (b) représentation de l'arbre binaire sous forme d'une suite de dictionnaires: un dictionnaire-étage contenant les 16 vecteurs-code correspondant aux 16 nœuds-feuille de l'arbre et quatre dictionnaires-arbre de tailles 8, 4, 2 et 1 correspondant à l'ensemble des nœuds de l'arbre aux niveaux 3, 2, 1 et 0 respectivement. Le niveau l de l'arbre binaire correspond à la résolution m du dictionnaire composé des nœuds (vecteurs-code) de ce niveau. Le vecteur-code d'indice i, issu du dictionnaire de résolution m, pointe vers deux vecteurs-code d'indices $2i$ et $2i+1$, issus du dictionnaire de résolution $m+1$.

Le vecteur-code d'indice i, issu du dictionnaire de résolution *m*, pointe vers deux vecteurs-code d'indices $2i$ et $2i+1$ issus du dictionnaire de résolution $m+1$. Notons que, dans notre cas, seuls les indices des vecteurs-code, issu du dictionnaire-étage, sont transmis par l'encodeur. Les dictionnaires-arbre sont utilisés pour accélérer le processus de l'encodage. La quantification vectorielle arborescente (TSVQ) est une alternative à la quantification vectorielle exhaustive (non-structurée). Pour une structure arborescente de hauteur $L = \log_2 N$ (où N est le nombre de nœuds-feuille), la recherche est effectuée par niveaux où à chaque niveau un nombre restreint de candidats sont sélectionnés. En conséquence, la complexité de codage augmente en fonction de $\log_2 N$ au lieu d'être en fonction de N (dans le cas d'une recherche exhaustive). La complexité de codage augmente donc linéairement avec le débit (en bits/vecteur) plutôt qu'exponentiellement.

Notons que dans le cas particulier de la reconstruction progressive [15, 68], le décodeur utilise les nœuds internes de l'arbre. Au lieu d'attendre que le vecteur source ait été complètement spécifié par l'encodeur, la transmission se fait à chaque niveau de l'arbre. Au fur et à mesure que le codeur progresse dans sa recherche, le vecteur représentatif du vecteur source est transmis utilisant un certain nombre de bits. Le décodeur reconstruit des reproductions de plus en plus fines (du vecteur source) au fur et à mesure que les bits arrivent. En [15] un quantificateur TSVQ a été conçu pour résoudre le problème d'assignement d'indices pour la transmission progressive des images. Plutôt que d'envoyer tous les bits pour chaque point de l'image en une seule fois, l'encodeur lit l'image, envoi une faible quantité d'information (un certain nombre de bits) pour chaque point de l'image, et recommence le processus de lecture. La qualité de l'image reçue s'améliore de plus en plus au fur et à mesure que les bits arrivent.

Notons aussi qu'un quantificateur TSVQ peut être combiné avec d'autres types de quantificateurs. Phamdo et al. [69] a proposé un quantificateur hybride TSVQ-SQ pour l'encodage des paramètres LSP (line spectrum pairs), où les vecteurs LSP sont quantifiés en utilisant une structure TSVQ et des quantificateurs scalaires SQs (Scalar quantizers) sont utilisés pour l'encodage du vecteur erreur.

3.2.2 Quantification vectorielle arborescente multi-étages

La quantification vectorielle arborescente multi-étages MTVQ (Multistage tree-structured vector quantization) peut être vue comme une combinaison de la quantification vectorielle arborescente TSVQ et de la quantification vectorielle multi-étage MSVQ (multistage vector quantization) [66]. Un quantificateur MTVQ a une structure multi-étages (comme pour un quantificateur MSVQ) où chaque étage supporte une structure arborescente (comme pour un quantificateur TSVQ). La figure 3.2 donne un exemple d'un quantificateur MTVQ à deux étages de taille 16 (de résolution 4) chacun. Notons que la taille/résolution d'un étage d'une structure MTVQ est définie comme la taille/résolution du dictionnaire-étage de cet étage. Le dictionnaire de résolution m, à l'étage *s* de la structure MTVQ, correspondant à l'ensemble des vecteurs-code du niveau *m* de l'arbre est noté par :

$$\boldsymbol{C}_m^{(s)} = \{\boldsymbol{c}_{m,i}^{(s)}, i = 0 \text{ to } 2^m - 1\}. \tag{3.2}$$

Pour un quantificateur MTVQ, les vecteurs-code du premier étage : $\boldsymbol{c}_{m,i}^{(1)}$ sont de même type que le vecteur à quantifier (vecteur-entrée) $\boldsymbol{x}$, alors que les vecteurs-code du deuxième étage et des étages suivants : $\boldsymbol{c}_{m,i}^{(s)}$ (avec $s > 1$) correspondent à des vecteurs-code erreurs (comme pour un quantificateur MSVQ). Notons que, pour une structure MTVQ à *K* étages, seuls les indices des vecteurs-code, issus des *K* dictionnaires-étage, sont transmis par l'encodeur.

Le vecteur quantifié $\hat{\boldsymbol{x}}$ du vecteur-entrée $\boldsymbol{x}$ est alors donné par l'équation suivante :

$$\hat{\boldsymbol{x}} = \boldsymbol{c}_{r_1,i_1}^{(1)} + \boldsymbol{c}_{r_2,i_2}^{(2)} + \ldots + \boldsymbol{c}_{r_K,i_K}^{(K)}, \tag{3.3}$$

où *K* est le nombre d'étages, r_s ($s = 1$ à K) sont les résolutions des dictionnaires-étage et i_s ($s = 1$ à K) sont les indices des vecteurs-code sélectionnés au niveau des *K* dictionnaires-étage.

Soit le cas où le vecteur à quantifier $\boldsymbol{x}$ (de dimension P) est divisé en K sous-vecteurs $\boldsymbol{x}_s$ ($s = 1 \text{ à } K$) de dimensions p_s ($s = 1 \text{ à } K$) avec $P = \sum_{s=1}^{K} p_s$ et $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_K^T]^T$ où T désigne la transposée et les vecteurs sont des vecteurs colonnes. Chaque sous-vecteur $\boldsymbol{x}_s$ est quantifié par l'étage s (de résolution r_s et de dimension p_s) d'une structure arborescente à K étages. Dans ce cas, le vecteur quantifié $\hat{\boldsymbol{x}}$ du vecteur-entrée $\boldsymbol{x}$ est donné par :

$$\hat{\boldsymbol{x}} = \left[\boldsymbol{c}_{r_1,i_1}^{(1)T}, \boldsymbol{c}_{r_2,i_2}^{(2)T}, ..., \boldsymbol{c}_{r_K,i_K}^{(K)T}\right]^T . \tag{3.4}$$

Un exemple d'une telle structure à deux étages pourrait être celle donnée par la figure 3.2 où dans ce cas de figure les deux étages sont de dimensions différentes (au lieu d'être de même dimension dans le cas d'une structure MTVQ) et les vecteurs-code du premier étage se concatènent avec ceux du deuxième étage (au lieu de s'additionner dans le cas d'une structure MTVQ). Une telle structure pourrait être vue comme une combinaison de la quantification vectorielle arborescente TSVQ et de la quantification vectorielle divisée SVQ (split vector quantization) qu'on pourrait appeler quantification vectorielle arborescente divisée. Notons enfin qu'une telle structure peut être vue aussi comme un cas particulier d'une structure MTVQ puisque une concaténation de K vecteurs $\boldsymbol{x}_s$ ($s = 1 \text{ à } K$) de dimension p_s ($s = 1 \text{ à } K$) : $\boldsymbol{x} = [\boldsymbol{x}_1^T, \boldsymbol{x}_2^T, ..., \boldsymbol{x}_K^T]^T$ est égale à l'addition de K vecteurs de même dimension $P = \sum_{s=1}^{K} p_s$: $\boldsymbol{x} = \boldsymbol{x}_1' + \boldsymbol{x}_2' + ... + \boldsymbol{x}_K'$ avec $\boldsymbol{x}_1' = [\boldsymbol{x}_1^T, \boldsymbol{0}_2^T, ..., \boldsymbol{0}_K^T]^T$, $\boldsymbol{x}_2' = [\boldsymbol{0}_1^T, \boldsymbol{x}_2^T, \boldsymbol{0}_3^T, ..., \boldsymbol{0}_K^T]^T$, …, $\boldsymbol{x}_K' = [\boldsymbol{0}_1^T, ..., \boldsymbol{0}_{K-1}^T, \boldsymbol{x}_K^T]^T$ où $\boldsymbol{0}_s$ est le vecteur nul (toutes les composantes sont nulles) de même dimension que le sous-vecteur $\boldsymbol{x}_s$. Donc, pour une telle structure MTVQ, les vecteurs-code de chaque étage ont des composantes nulles et il n'y a pas d'échantillons non nuls qui se recouvrent entre les vecteurs-code de deux étages quelconques. De la même manière on peut considérer la quantification vectorielle divisée SVQ comme un cas particulier de la quantification vectorielle multi-étages MSVQ [13].

La quantification MTVQ a été précédemment appliquée au codage audio [70, 71] où une structure de deux à trois étages est utilisée pour le codage des coefficients transformés en cosinus discrètes : DCT (Discrete Cosine Transform). Chemla et al. dans [72, 73] ont proposé l'utilisation d'un quantificateur MTVQ pour le codage des paramètres LSF, où la structure MTVQ est conçue et parcourue comme une seule structure arborescente binaire globale. Récemment Chu dans [66] a proposé un algorithme efficace d'élaboration de dictionnaires pour concevoir un quantificateur MTVQ pour la quantification incorporée des paramètres LSF.

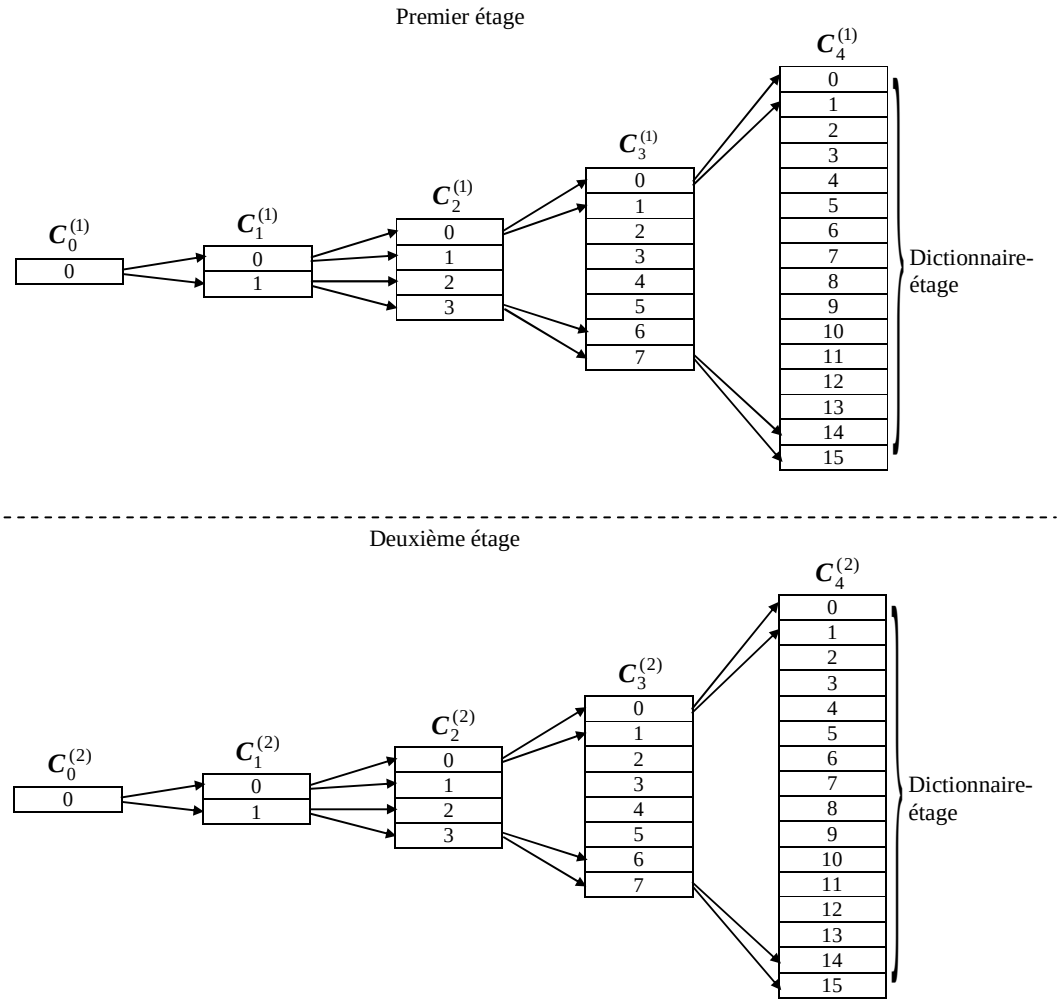


Figure 3.2 : Exemple d'une structure arborescente multi-étages MTVQ à deux étages. Chaque étage correspond à un arbre binaire équilibré composé d'un dictionnaire-étage et de quatre dictionnaires-arbre. Dans cet exemple les deux dictionnaires-étage (du premier et du deuxième étage) ont le même nombre de vecteurs-code.

Notons qu'une structure MTVQ (telle que décrite plus haut) peut être vue globalement comme une structure TSVQ avec contrainte de stockage (constrained-storage TSVQ) [73]. Les nœuds des premiers niveaux de la structure TSVQ correspondent à des vecteurs de même caractéristique que le vecteur-entrée (comme dans un TSVQ classique) alors que les nœuds des niveaux suivants correspondent à des vecteurs erreurs (comme pour le deuxième étage et les étages suivant d'une structure MTVQ). Le concept de dictionnaire partagé par plusieurs vecteurs sources [16] est utilisé pour contraindre la structure TSVQ à avoir une complexité de mémorisation (stockage) raisonnable. Ceci peut être accompli notamment en limitant le nombre de dictionnaires de certains niveaux particuliers de l'arbre au lieu que le nombre de dictionnaires soit le double (dans le cas d'un TSVQ binaire) de celui du niveau précédent [72, 73]. Notons aussi que même une structure MSVQ peut être vue comme un cas particulier d'une structure TSVQ [16, 17].

3.3 Quantification vectorielle incorporée

Soit un quantificateur MTVQ à K étages de tailles N_s et de résolutions r_s ($s = 1$ à K). On donne à la figure 3.3 un exemple d'une structure MTVQ à deux étages ($K = 2$) de résolutions $r_1 = 4$ et $r_2 = 3$ respectivement. Les indices des vecteurs-code sont exprimés dans le système de numérotation binaire. Les indices des vecteurs-code sont formés en traversant l'arbre (du niveau racine vers le niveau le plus élevé) le long des branches et en groupant les bits se trouvant le long du chemin. Une branche est représentée par une flèche liant deux vecteurs-code et un chemin est une succession de plusieurs branches. Au niveau de l'encodeur, seul les K indices des vecteurs-code, issus des dictionnaires-étage, sont transmis vers le décodeur.

Au niveau de l'encodeur, les dictionnaires-arbre (de chaque étage) sont utilisés pour accélérer le processus de recherche des indices. Ainsi, les indices i_s ($s = 1$ à K) des vecteurs-code, issus des dictionnaires-étage, sont codés sur r_s bits ($s = 1$ à K) respectivement et sont transmis vers le décodeur sous forme d'un flux-binaire de longueur r bits avec $r = \sum_{s=1}^{K} r_s$ (voir figure 3.4) :

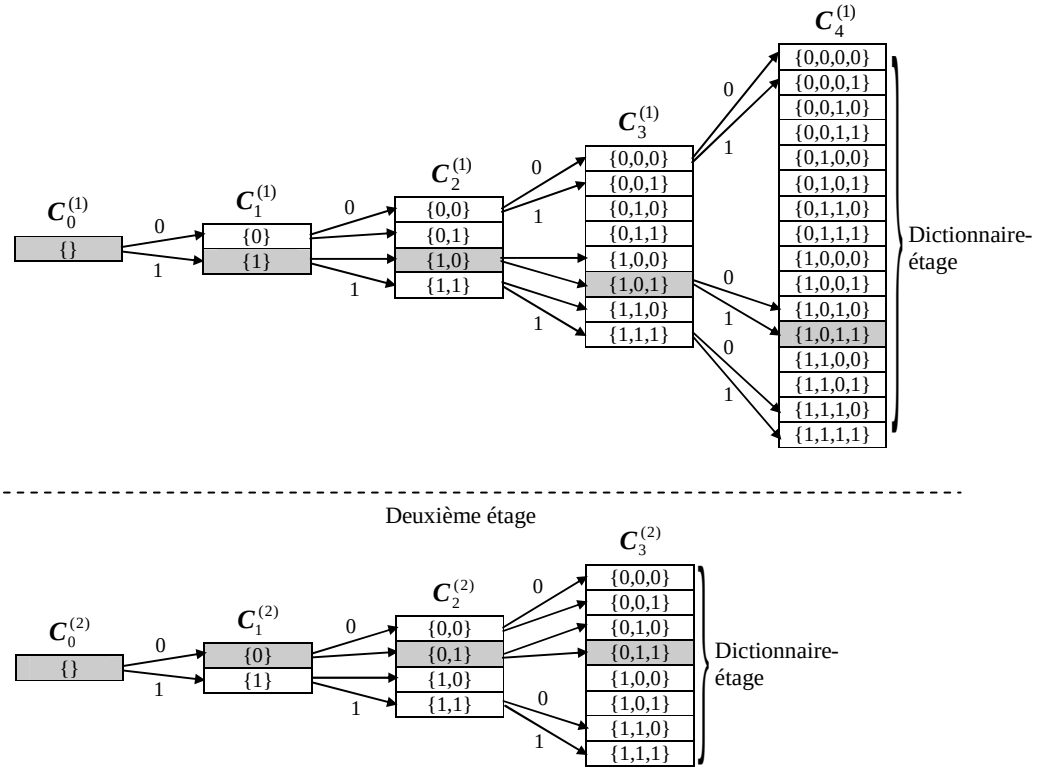


Figure 3.3 : Exemple d'un quantificateur MTVQ à deux étages de résolution 4 et 3 respectivement. Les indices des vecteurs sont formés en traversant l'arbre (du niveau racine vers le niveau le plus élevé) le long des branches et en groupant les bits se trouvant le long du chemin. Les deux vecteurs sélectionnés ont pour indice: $i_1 = 11 = \{1,0,1,1\}$ et $i_2 = 3 = \{0,1,1\}$. Selon le nombre de bits *m* reçus par le décodeur, 7 versions du vecteur quantifié peuvent être construits (tableau 3.1)

i_1 (r_1 bits)	i_2 (r_2 bits)	…	i_K (r_K bits)

Figure 3.4 : Indices des vecteurs-code transmis sou forme d'un flux-binaire de $r = r_1 + r_2 + ... + r_K$ bits

Si le canal de transmission est encombré, les bits du flux-binaire sont rejetés bit-par-bit en commençant par la fin du flux-binaire (par l'indice i_K). Donc, le décodeur reçoit un flux-binaire de *m* bits avec $m \le r$. En notant par m_s ($s = 1$ à K) le nombre de bits disponibles pour chaque étage (de la structure MTVQ), nous avons :

$$\begin{cases} m_1 = \min(m, r_1) \\ m_s = \min\left(\left(m - \sum_{i=1}^{s-1} m_i\right), r_s\right) \qquad s = 2 \text{ à } K. \end{cases} \tag{3.5}$$

Donc la priorité est accordée aux étages inférieurs en assignant les bits disponibles à ces étages en premier. Si $m_s = r_s$, le décodeur récupère le vecteur-code d'indice i_s issu du dictionnaire-étage; autrement (si $m_s < r_s$) le dictionnaire-arbre de résolution m_s bits est utilisé pour localiser un vecteur-code d'indice i'_s résultant des m_s bits les plus significatifs MSB (most significant bits) de l'indice i_s. Notons que si $m_s = 0$ (aucun bit n'est reçu pour l'étage *s*) alors le vecteur-code issu du dictionnaire-arbre de résolution 0 (correspondant au nœud racine de l'arbre à l'étage *s*) est utilisé comme vecteur par défaut. Donc, au décodeur, les dictionnaires-arbre sont utilisés pour récupérer des versions approximées du vecteur-code, issu du dictionnaire-étage, lorsque l'indice complet de ce dernier n'est pas disponible. Le vecteur quantifié $\hat{\boldsymbol{x}}$ du vecteur-entrée $\boldsymbol{x}$ construit au niveau du décodeur est donné par :

$$\hat{\boldsymbol{x}} = \boldsymbol{c}^{(1)}_{m_1,i'_1} + \boldsymbol{c}^{(2)}_{m_2,i'_2} + ... + \boldsymbol{c}^{(K)}_{m_K,i'_K} . \tag{3.6}$$

De cette façon, la qualité du vecteur quantifié $\hat{\boldsymbol{x}}$, construit au niveau du décodeur, dépend du nombre de bits $m = m_1 + m_2 + ... + m_K$ reçus. La meilleure qualité (du vecteur quantifié) est atteinte lorsque le flux-binaire est reçu en entier, c'est-à-dire lorsque les indices des vecteurs-code issus des dictionnaires-étage sont reçus en entier. Pour ce cas, on a $m_s = r_s$ et $i'_s = i_s$ ($s = 1$ à K) et par conséquent le vecteur quantifié est le même que celui trouvé au niveau de l'encodeur : $\hat{\boldsymbol{x}} = \boldsymbol{c}^{(1)}_{r_1,i_1} + \boldsymbol{c}^{(2)}_{r_2,i_2} + ... + \boldsymbol{c}^{(K)}_{r_K,i_K}$. Différentes qualités du vecteur quantifié sont obtenues lorsque le flux-binaire n'est pas reçu en entier (les indices des vecteurs-code issus des dictionnaires-étage ne sont pas reçus en entier). Dans ce cas, le vecteur-code (issu du dictionnaire-étage) dont l'indice n'est pas reçu en entier est remplacé par un vecteur-code issu de l'un des dictionnaires-arbre associés. Le vecteur quantifié ainsi obtenu (équation 3.6) est de moindre qualité puisque différent de celui trouvé au niveau de l'encodeur. Le vecteur quantifié de plus mauvaise qualité est celui obtenu lorsqu'aucun indice n'est reçu : $\hat{\boldsymbol{x}} = \boldsymbol{c}^{(1)}_{0,0} + \boldsymbol{c}^{(2)}_{0,0} + ... + \boldsymbol{c}^{(K)}_{0,0}$ et le vecteur quantifié de la plus meilleure qualité est celui obtenu lorsque tous les indices (transmis par l'encodeur)

sont reçus en entier : $\hat{\boldsymbol{x}} = \boldsymbol{c}^{(1)}_{r_1,i_1} + \boldsymbol{c}^{(2)}_{r_2,i_2} + ... + \boldsymbol{c}^{(K)}_{r_K,i_K}$. Une qualité intermédiaire peut être obtenue lorsque les indices ne sont pas reçus en entier (équation (3.6)).

Donc un quantificateur arborescent (à un seul ou plusieurs étages) est un quantificateur incorporé. En effet les indices des vecteurs-code issus des dictionnaires-étage (sélectionnés au niveau de l'encodeur) correspondent au débit le plus élevé du quantificateur. En tout point du réseau de transmission, un ou plusieurs bits de plus faible poids de chaque indice peuvent être écartés afin de réduire le débit de transmission et ainsi atténuer l'encombrement du réseau à paquets. Au niveau du décodeur, le quantificateur global peut être vue comme étant constitué de quantificateurs de plus faibles débits imbriqués dans le quantificateur correspondant au débit le plus élevé. Lorsque les indices transmis par l'encodeur ne sont pas reçus en entier (le débit a été réduit), le quantificateur, au niveau du décodeur, est capable de calculer une version approximée du vecteur quantifié trouvé au niveau de l'encodeur. Étant donné que la réduction du débit de transmission peut se produire durant la transmission (après l'opération de l'encodage), le codage incorporé (utilisant un quantificateur incorporé) diffère du codage à débit variable, dans lequel l'encodeur et le décodeur doivent utiliser le même nombre de bits pour chaque indice transmit. Dans les deux cas, le décodeur doit être informé du nombre de bits à utiliser pour chaque indice reçu.

La figure 3.3 donne un exemple d'une structure MTVQ à deux étages de résolution 4 et 3 respectivement. Dans cet exemple on considère qu'au niveau de l'encodeur les vecteurs-code (issus des dictionnaires-étages) d'indices: $i_1 = 11 = \{1,0,1,1\}$ et $i_2 = 3 = \{0,1,1\}$ ont été sélectionnés. Au niveau du décodeur huit versions du vecteur quantifié à différents qualités peuvent être construits en fonction du nombre de bits *m* reçus (tableau 3.1). Un quantificateur peut avoir une structure hybride : Par exemple un quantificateur pourrait être composé d'une structure MTVQ à deux étages de résolutions 6 et 8 bits respectivement où le deuxième étage correspond à deux structures TSVQ (dont les vecteurs se concatènent) de même résolution de 4 bits chacun. La structure globale peut être vue comme une structure à trois étages de résolutions $r_1 = 6, r_2 = 4$ et $r_3 = 4$ bits où chaque étage correspond à une structure arborescente. Les vecteurs-code (de dimension k_1) du premier étage s'additionnent avec le résultat de la concaténation des

Tableau 3.1 : Exemple d'un quantificateur MTVQ à deux étages de résolutions 4 et 3 respectivement. Les deux vecteurs-code sélectionnés aux niveaux de l'encodeur ont pour indice: $i_1 = 11 = \{1,0,1,1\}$ et $i_2 = 3 = \{0,1,1\}$. Selon le nombre de bits *m* reçus par le décodeur 7 versions du vecteur quantifié à différents qualités peuvent être construits.

m	(m_1, m_2)	Vecteur quantifié
7	(4, 3)	$\boldsymbol{c}_{4,11}^{(1)} + \boldsymbol{c}_{3,3}^{(2)}$
6	(4, 2)	$\boldsymbol{c}_{4,11}^{(1)} + \boldsymbol{c}_{2,1}^{(2)}$
5	(4, 1)	$\boldsymbol{c}_{4,11}^{(1)} + \boldsymbol{c}_{1,0}^{(2)}$
4	(4, 0)	$\boldsymbol{c}_{4,11}^{(1)} + \boldsymbol{c}_{0,0}^{(2)}$
3	(3, 0)	$\boldsymbol{c}_{3,5}^{(1)} + \boldsymbol{c}_{0,0}^{(2)}$
2	(2, 0)	$\boldsymbol{c}_{2,2}^{(1)} + \boldsymbol{c}_{0,0}^{(2)}$
1	(1, 0)	$\boldsymbol{c}_{1,1}^{(1)} + \boldsymbol{c}_{0,0}^{(2)}$
0	(0, 0)	$\boldsymbol{c}_{0,0}^{(1)} + \boldsymbol{c}_{0,0}^{(2)}$

Tableau 3.2 : Exemple d'un quantificateur hybride correspondant à une structure globale de trois étages de résolution $r_1 = 6$, $r_2 = 4$, $r_3 = 4$ bits. Les vecteurs-code du premier étage s'additionnent avec le résultat de la concaténation des vecteurs-code du deuxième étage et des vecteurs-code du troisième étage. Les trois vecteurs-code sélectionnés aux niveau du l'encodeur ont pour indices: $i_1 = 46 = \{1,0,1,1,1,0\}$, $i_2 = 13 = \{1,1,0,1\}$ et $i_3 = 5 = \{0,1,0,1\}$. Selon le nombre de bits *m* reçus par le décodeur 14 versions du vecteur quantifié à différents qualités peuvent être construits.

m	(m_1, m_2, m_3)	Vecteur quantifié
14	(6, 4, 4)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{4,13}^{(2)}, \boldsymbol{c}_{4,5}^{(3)}]$
13	(6, 4, 3)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{4,13}^{(2)}, \boldsymbol{c}_{3,2}^{(3)}]$
12	(6, 4, 2)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{4,13}^{(2)}, \boldsymbol{c}_{2,1}^{(3)}]$
11	(6, 4, 1)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{4,13}^{(2)}, \boldsymbol{c}_{1,0}^{(3)}]$
10	(6, 4, 0)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{4,13}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
9	(6, 3, 0)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{3,6}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
8	(6, 2, 0)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{2,3}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
7	(6, 1, 0)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{1,1}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
6	(6, 0, 0)	$\boldsymbol{c}_{6,46}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
5	(5, 0, 0)	$\boldsymbol{c}_{5,23}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
4	(4, 0, 0)	$\boldsymbol{c}_{4,11}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
3	(3, 0, 0)	$\boldsymbol{c}_{3,5}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
2	(2, 0, 0)	$\boldsymbol{c}_{2,2}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
1	(1, 0, 0)	$\boldsymbol{c}_{1,1}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$
0	(0, 0, 0)	$\boldsymbol{c}_{0,0}^{(1)} + [\boldsymbol{c}_{0,0}^{(2)}, \boldsymbol{c}_{0,0}^{(3)}]$

vecteurs-code (de dimension k_2) du deuxième étage et des vecteurs-code (de dimension k_3) du troisième étage. La dimension du premier étage doit être égale à la somme des dimensions du deuxième et du troisième étages : $k_1 = k_2 + k_3$. Supposons que les trois vecteurs-code sélectionnés aux niveau de l'encodeur ont pour indices : $i_1 = 46 = \{1,0,1,1,1,0\}$, $i_2 = 13 = \{1,1,0,1\}$ et $i_3 = 5 = \{0,1,0,1\}$. Selon le nombre de bits *m* reçus par le décodeur les 14 valeurs possibles du vecteur quantifié qui peuvent être construits sont données au tableau 3.2. Notons que le flux-binaire à transmettre peut être arrangé de tel façon à changer l'ordre des priorités des bits des indices transmis. Par exemple, dans l'exemple ci-dessus, la priorité la plus élevée pourrait être donnée au premier étage alors que le deuxième et le troisième étage auraient la même priorité.

Dans le contexte du codage d'image, un TSVQ est exploré pour la quantification incorporée des coefficients de la transformée en ondelette dans [74, 75]. Chu dans [66] a utilisé la quantification MTVQ pour la quantification incorporée des paramètres LSF. En générale les codeurs de parole qui incluent la quantification incorporée sont échelonnables. Comme la norme ITU-T G.727 ADPCM : la modulation par impulsions et codage différentiel adaptatif (MICDA) imbriqué, opérant à 5, 4, 3 et 2 bits par échantillon (40, 32, 24 et 16 kbit/s). En effet, la norme MICDA imbriqué est constituée d'une suite d'algorithmes telle que les niveaux de décision des quantificateurs des plus faibles débits sont des sous-ensembles de ceux du quantificateur du débit le plus élevé [54].

3.4 Méthodes de conception d'un quantificateur arborescent

On distingue deux méthodes principales de construction d'une structure arborescente: La première méthode consiste à intégrer la construction de l'arbre (binaire) à celle du dictionnaire-étage correspondant aux nœuds-feuille de l'arbre. Cette approche est une approche descendante : l'arbre est construit depuis le nœud racine en descendant vers les nœuds-feuille (figure 3.1-(a)). La deuxième méthode consiste à construire l'arborescence une fois le dictionnaire-étage construit de manière non-structurée. C'est-à-dire que l'arbre est construit à partir des nœuds-feuille en montant vers le nœud racine. Cette approche est une approche ascendante [5, 67].

3.4.1 Arborescences descendantes

On distingue deux types de construction de structures arborescentes descendantes : On parle d'arborescences descendantes équilibrées et non-équilibrées.

3.4.1.1 Arborescences descendantes équilibrées

Pour un arbre binaire équilibré, les N nœuds-feuille de l'arbre sont situés sur le même niveau L donné par : $L = \log_2 N$ (voir figure 3.5). La structure arborescente est construite un niveau à la fois. Une fois le premier niveau (niveau racine) construit, chaque niveau de l'arbre est construit en divisant en deux nœuds enfants chaque nœud (parent) du niveau précédent. Chaque nœud interne de l'arbre est associé à une partition (région ou cellule), et les nœuds enfants sont produits en partitionnant encore plus ces régions individuellement. L'algorithme est résumé comme suit [5, 67] :

1-Soit T la séquence d'apprentissage, le vecteur-code correspondant au nœud racine $\boldsymbol{c}_{0,0}$ est calculé comme le centroïde de la séquence d'apprentissage prise dans sa totalité. Deux nouveaux vecteurs sont générés en perturbant le vecteur-code racine : $\boldsymbol{c}_{0,0} + \boldsymbol{\varepsilon}$ et $\boldsymbol{c}_{0,0} - \boldsymbol{\varepsilon}$ où $\boldsymbol{\varepsilon}$ est un vecteur de perturbation fixe. L'algorithme LBG est ensuite appliqué sur la séquence d'apprentissage T en prenant comme dictionnaire initiale les deux vecteurs générés précédemment ($\boldsymbol{c}_{0,0} + \boldsymbol{\varepsilon}$ et $\boldsymbol{c}_{0,0} - \boldsymbol{\varepsilon}$). Nous obtenons alors les deux nœuds (vecteurs-code) $\boldsymbol{c}_{1,0}$ et $\boldsymbol{c}_{1,1}$ du niveau $l = 1$ de l'arbre et qui correspondent aux deux nœuds enfants du nœud racine $\boldsymbol{c}_{0,0}$. L'ensemble d'apprentissage T est partitionné en deux sous ensembles : $T_{1,0}$ et $T_{1,1}$ qui contiennent respectivement les vecteurs d'apprentissage les plus proches des vecteurs-code $\boldsymbol{c}_{1,0}$ et $\boldsymbol{c}_{1,1}$.

2-L'ensemble $T_{1,0}$ est utilisé comme séquence d'apprentissage pour générer les deux nœuds enfants $\boldsymbol{c}_{2,0}$ et $\boldsymbol{c}_{2,1}$ du nœud $\boldsymbol{c}_{1,0}$. De même $T_{1,1}$ est utilisé comme séquence d'apprentissage pour générer les deux nœuds enfants $\boldsymbol{c}_{2,2}$ et $\boldsymbol{c}_{2,3}$ du nœud $\boldsymbol{c}_{1,1}$. Nous obtenons alors les quatre vecteurs-code correspondants aux nœuds du niveau $l = 2$ de l'arbre (figure 3.5).

3-Partitionner chaque ensemble $T_{1,i}$ $(i = 0,1)$ en deux sous ensembles $T_{2,2i}$ et $T_{2,2i+1}$ qui contiennent respectivement les vecteurs d'apprentissage les plus proches des vecteurs-code $\boldsymbol{c}_{2,2i}$ et $\boldsymbol{c}_{2,2i+1}$. L'algorithme LBG est ensuite appliqué sur chaque nouvel ensemble d'apprentissage $T_{2,i}$ $(i = 0,1,2,3)$ pour construire les nœuds (vecteurs-code) enfants $\boldsymbol{c}_{3,2i}$ et $\boldsymbol{c}_{3,2i+1}$ du nœud $\boldsymbol{c}_{2,i}$. Ainsi le niveau $l = 3$ (composé des vecteurs-code $\boldsymbol{c}_{3,i}$, $i = 0$ à 7) de l'arbre binaire est construit.

4-Continuer ce processus jusqu'à ce que le niveau (prédéterminé) L soit atteint. Les nœuds-feuille correspondent aux $N = 2^L$ vecteurs-code utilisés pour la transmission. Un quantificateur TSVQ binaire de hauteur (de profondeur) L correspond à un débit fixe de L bits/vecteur.

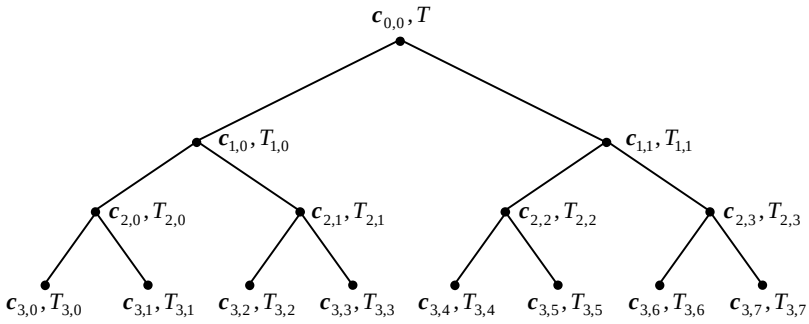


Figure 3.5 : Exemple de construction d'un arbre binaire équilibré de hauteur L = 3. $T_{m,i}$ est l'ensemble de vecteurs d'apprentissage encodés par le vecteur-code $\boldsymbol{c}_{m,i}$. L'ensemble de vecteurs d'apprentissage $T_{m,i}$ est utilisé pour générer les deux nœuds (vecteurs-code) enfants $\boldsymbol{c}_{m+1,2i}$ et $\boldsymbol{c}_{m+1,2i+1}$ du nœud parent $\boldsymbol{c}_{m,i}$.

La quantification vectorielle arborescente équilibrée est sous-optimale par rapport à la quantification vectorielle non-structurée (exhaustive) et ceci pour deux principales raisons [67] :

• Sous-optimalité du dictionnaire, composé de l'ensemble des vecteur-code correspondants aux nœuds-feuille de l'arbre, puisque celui-ci contient moins de représentants pertinents que dans le cas exhaustif du fait de la contrainte à laquelle il est soumis lors de sa construction.

• Sous-optimalité de l'encodage : Du fait du procédé de recherche dans l'arbre, le vecteur-code trouvé (parmi les nœuds-feuille de l'arbre) n'est pas obligatoirement le vecteur-code le plus proche du vecteur source (vecteur-entrée).

Un quantificateur vectoriel MTVQ à *K* étages peut être vu comme une succession de *K* structures arborescentes (TSVQ). Dans ce cas, la structure MTVQ est construite d'une manière séquentielle en commençant par construire l'arbre du premier étage en appliquant la méthode décrite plus haut, puis celui du deuxième étage et ainsi de suite. Notons que le premier étage est construit à partir d'une séquence d'apprentissage contenant des vecteurs de même caractéristique que les vecteurs-entrée (vecteurs sources). Pour les étages suivants, chaque étage est construit à partir d'une nouvelle séquence d'apprentissage, contenant des vecteurs erreurs (de quantification), calculée à partir de la séquence d'apprentissage de l'étage précédent. Une approche similaire a été proposée en [72, 73] pour la construction d'un quantificateur MTVQ et d'un quantificateur TSVQ avec contrainte de stockage (constrained-storage TSVQ) qui utilise le concept de dictionnaire partagé [16] pour réduire la complexité de stockage d'une structure TSVQ. Notons qu'en [15] une méthode qui consiste à construire un arbre binaire équilibré à partir d'un dictionnaire non-structuré a été proposée.

3.4.1.2 Arborescences descendantes non-équilibrées

Une autre méthode de construction d'un quantificateur arborescent (binaire) consiste à accroitre l'arbre en trouvant le nœud dont l'éclatement apportera la meilleure amélioration de la qualité de la structure arborescente. Ce nœud est ensuite divisé en deux nœuds enfants. Utilisant la distorsion moyenne comme mesure de qualité, l'idée consiste à éclater seulement le nœud terminal, dans l'arbre courant, qui contribue le plus à la distorsion moyenne totale [5, 67]. L'algorithme LBG est ensuite appliqué comme expliqué précédemment (cas d'un arbre équilibré) sauf que cette fois l'arbre se développe par nœuds et non pas par niveaux et la forme de la structure arborescente résultante est en générale non-équilibrée (figure 3.6) : les nœuds terminaux (nœuds-feuille) apparaissent à différents niveaux de l'arbre. La longueur du mot-code associée à chaque nœud-feuille est égale à la hauteur (niveau) du nœud-feuille dans l'arbre. Ceci conduit à un débit variable présentant l'avantage de distribuer efficacement les bits selon l'activité des vecteurs (plus de bits sont affectés aux vecteurs les plus importants et moins de bits sont affectés aux vecteurs moins importants). Plusieurs critères ont été définis pour déterminer quel nœud doit être éclaté [5, 67, 76, 77] :

1- Le critère le plus naturel est de prendre le nœud qui présente la plus grande baisse de distorsion. Dans ce cas, il est nécessaire de calculer la distorsion que provoquerait l'éclatement de chacun des nœuds terminaux (ce qui revient à construire un arbre de hauteur moyenne $h' = h + 1$ où h est la hauteur moyenne de l'arbre courant).

2- Un deuxième critère consiste à diviser le nœud contribuant le plus à la distorsion moyenne totale (le nœud qui a actuellement la plus grande distorsion moyenne). Ce critère est moins lourd en complexité de calculs que le critère précédent puisqu'il n'est plus nécessaire d'accroitre l'arbre pour déterminer le nœud qui doit être éclaté.

3- Un troisième critère consiste à choisir le nœud qui assure le meilleur compromis "baisse de la distorsion/hausse de débit". Pour chaque nœud terminal (nœud-feuille) courant n qui est découpé en deux nœuds enfants n_L et n_R, on calcule la variation de distorsion ΔD et la variation de débit ΔH par :

$$\Delta D = p(n_L)d(n_L) + p(n_R)d(n_R) - p(n)d(n) \tag{3.7}$$

$$\Delta H = p(n_L)h(n_L) + p(n_R)h(n_R) - p(n)h(n) \tag{3.8}$$

où $h(n)$, $h(n_L)$ et $h(n_R)$ sont les hauteurs des nœuds n, n_L et n_R respectivement avec : $h(n_L) = h(n_R) = h(n) + 1$ puisque la hauteur d'un nœud est toujours égale à la hauteur du nœud parent incrémenté de 1. $p(n)$, $p(n_L)$ et $p(n_R)$ sont les probabilités des nœuds n, n_L et n_R respectivement avec $p(n) = p(n_L) + p(n_R)$. La probabilité du nœud n peut être estimé comme le taux de vecteurs de la séquence d'apprentissage encodés par le nœud n (dont le chemin d'encodage passe par le nœud n). $d(n)$, $d(n_L)$ et $d(n_R)$ sont les distorsions associées aux nœuds n, n_L et n_R respectivement. La distorsion $d(n)$ associée au nœud n étant la somme des distances des vecteurs d'apprentissage encodés par le nœud n par rapport au vecteur-code associé au nœud n. Ainsi, le rapport entre la diminution de la distorsion et la hausse de débit résultant du découpage du nœud-feuille n est défini par [76] :

$$\lambda(n) = -\frac{\Delta D}{\Delta H} \tag{3.9}$$

La structure arborescente du quantificateur TSVQ peut être conçue un nœud à la fois en découpant le nœud-feuille courant fournissant la plus grande valeur $\lambda(n)$. Ainsi chaque découpage d'un nœud optimise le compromis «débit-distorsion» plutôt que de découper un nœud sans considérer l'augmentation en débit ou la diminution en distorsion résultante.

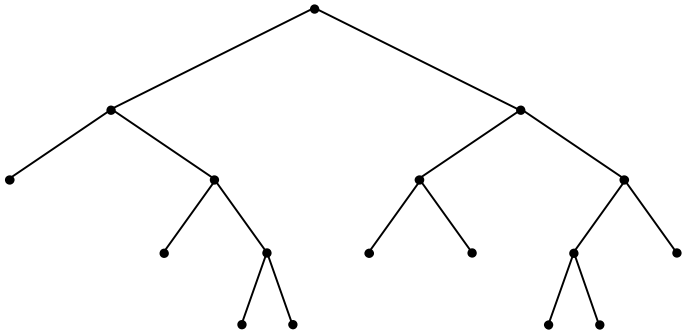


Figure 3.6 : Exemple d'un arbre binaire non-équilibré. Tous les nœuds-feuille ne se situent pas sur le même niveau (ne sont pas sur la même hauteur) et leur nombre n'est donc pas nécessairement une puissance de 2. La longueur du mot de code associée à chaque nœud-feuille est égale à la hauteur du nœud-feuille dans l'arbre.

Une autre méthode de construction d'un arbre non-équilibré consiste à appliquer l'algorithme BFOS du nom de ses auteurs Breiman, Friedman, Olshen et Stone [5]. La première étape, avant même l'application de cet algorithme consiste à construire une structure arborescente de taille importante (utilisant l'approche descendante), cet arbre étant élagué par la suite (pruned tree) en appliquant l'algorithme BFOS. Le but de l'élagage est de supprimer les parties (sous arbres) de l'arbre ayant le plus petit rapport de l'augmentation de la distorsion sur la diminution du débit [5] (un critère semblable à celui de l'équation (3.9)). On parle de quantificateur TSVQ élagué (pruned TSVQ : PTSVQ). Un quantificateur PTSVQ est plus performant (dans le sens débit-distorsion) qu'un quantificateur TSVQ (équilibré) en étant capable d'affecter plus de bits aux nœuds (vecteurs) les plus importants. Plusieurs autres approches ont été proposées (surtout dans le domaine du traitement d'image). Par exemple en [78, 79] les méthodes proposées opèrent sur un dictionnaire non-structuré pour construire une structure arborescente (généralement non-équilibré) pour accélérer le processus de l'encodage par rapport à la recherche exhaustive. La méthode proposée en [80] utilise le concept de dictionnaire

partagé par plusieurs vecteurs sources [16] pour construire une structure arborescente descendante non-équilibrée. Cette méthode permet de réduire de manière importante la complexité de stockage du quantificateur résultant. Dans [81] l'auteur propose un quantificateur vectoriel arborescent à nombre de branches variables. Contrairement à une structure arborescente à *m* branches où chaque nœud possède un nombre fixe (*m*) de nœuds enfants, la méthode présentée en [81] consiste à chercher (pour un codage optimal) le nombre de nœuds enfants pour chaque nœud devant être découpé.

3.4.2 Arborescences ascendantes

La construction de l'arborescence se fait une fois le dictionnaire (correspondant à l'ensemble des *N* vecteurs-code associés aux nœuds-feuille de l'arbre) ayant été construit. Ce dictionnaire est construit de manière tout à fait classique (algorithme de LBG par exemple) et ne présente à priori aucune structure particulière. L'édification d'une classification hiérarchique ascendante consiste à former à partir de petites classes très homogènes des classes de moins en moins homogènes jusqu'à l'obtention d'une classe unique. Cette méthode est appelée technique de fusion de cellules (cell-merging technique) dans [66] et peut être considérée comme une alternatif à la méthode de construction d'une arborescence descendante équilibrée (Paragraphe 3.4.1.1). En effet pour la méthode descendante équilibrée, le dictionnaire (l'ensemble des nœuds-feuille) résultant est sous optimale puisque celui-ci contient moins de représentants (vecteurs-code) pertinents que dans le cas d'un dictionnaire non-structuré (exhaustif) du fait de la contrainte à laquelle il est soumis lors de sa construction. En effet, les vecteurs-code correspondants aux nœuds-feuille de l'arbre sont fortement influencés par les vecteurs-code correspondant aux nœuds internes (Paragraphe 3.4.1.1). De meilleurs résultats (c'est-à-dire des vecteurs-code plus pertinents) peuvent être obtenus lorsque les vecteurs-code, correspondants aux *N* nœuds-feuille de l'arbre, sont optimisés indépendamment.

Riskin et al. ont proposé dans [15] la technique de fusion de cellules pour résoudre le problème d'assignement d'indices pour la transmission progressive des images utilisant un dictionnaire non-structuré. Les vecteurs-code du dictionnaire non-structuré sont d'abord trouvés. Un arbre binaire équilibré (de hauteur égale à la résolution, en bits/vecteur, du dictionnaire non-structuré) est construit à partir du niveau le plus haut de

l'arbre. En effet, les vecteurs-code du dictionnaire sont affectés aux nœuds-feuille de l'arbre puis les cellules associées à ces vecteurs-code sont fusionnées pour former les nœuds des niveaux inferieurs de l'arbre. Les nœuds (vecteurs-code) internes sont calculés à partir des vecteurs-code associés aux nœuds-feuille. Notons que les vecteurs-code du dictionnaire (non-structuré) sont ré-indexés pour qu'ils correspondent aux nœuds-feuille de l'arbre ainsi construit. Le but est donc de construire un arbre binaire équilibré à partir des vecteurs-code d'un dictionnaire non-structuré. La procédure est décrite comme suite [15] :

1-L'algorithme LBG est appliqué sur la séquence d'apprentissage T pour construire un dictionnaire non-structuré de N vecteurs-code $\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{N-1}\}$. Ces vecteurs-code sont placés au niveau le plus élevé (niveau $L = \log_2 N$) de l'arbre binaire (correspondant aux nœuds-feuille de l'arbre). L'ensemble T_i est défini comme le sous-ensemble, de T, des vecteurs les plus proches au vecteur-code $\boldsymbol{y}_i$ ($\boldsymbol{y}_i$ est le centroïde des vecteurs appartenant à T_i).

2-Le problème peut être exprimé comme suit : Il s'agit de trouver une séquence de paires d'indices $\{(i_0, j_0), (i_1, j_1), ..., (i_{(N/2)-1}, j_{(N/2)-1})\}$ tel que $i_k \neq j_k$ pour $k = 0 \text{ à } (N/2)-1$ et $\bigcup_{k=0 \text{ à } (N/2)-1} (i_k, j_k) = \{0, 1, ..., N-1\}$ où la paire (i_k, j_k) correspond aux indices des deux vecteurs-code $\boldsymbol{y}_{i_k}$ et $\boldsymbol{y}_{j_k}$ issus du dictionnaire $\mathbf{Y}$. Fusionner les vecteurs $\boldsymbol{y}_{i_k}$ et $\boldsymbol{y}_{j_k}$ en un seul vecteur $\boldsymbol{y}'_k$ revient à fusionner les cellules $\mathbf{R}_{i_k}$ et $\mathbf{R}_{j_k}$ (associées à $\boldsymbol{y}_{i_k}$ et $\boldsymbol{y}_{j_k}$ respectivement) en une seule cellule $\mathbf{R}'_k$, associée à $\boldsymbol{y}'_k$, tel que : $\mathbf{R}'_k = \mathbf{R}_{i_k} \cup \mathbf{R}_{j_k}$. On a donc :

$$T'_k = T_{i_k} \cup T_{j_k} \quad , \tag{3.10}$$

$$\boldsymbol{y}'_k = \frac{\left\|T_{i_k}\right\| \boldsymbol{y}_{i_k} + \left\|T_{j_k}\right\| \boldsymbol{y}_{j_k}}{\left\|T'_k\right\|}, \tag{3.11}$$

$$d(\boldsymbol{y}'_k) = \sum_{\boldsymbol{x} \in T'_k} \left\|\boldsymbol{x} - \boldsymbol{y}'_k\right\|^2 , \tag{3.12}$$

où T_{i_k} et T_{j_k} sont les sous ensembles des vecteurs d'apprentissage les plus proches des vecteurs-code $\boldsymbol{y}_{i_k}$ et $\boldsymbol{y}_{j_k}$ respectivement. T'_k est le sous ensemble des vecteurs d'apprentissage associé au vecteur $\boldsymbol{y}'_k$ (résultant de la fusion de $\boldsymbol{y}_{i_k}$ et $\boldsymbol{y}_{j_k}$). $\|T'_k\|$ correspond au nombre de vecteurs dans T'_k et $d(\boldsymbol{y}'_k)$ est la distorsion associée au vecteur $\boldsymbol{y}'_k$. D'après l'équation (3.11) $\boldsymbol{y}'_k$ est donc le centroïd des vecteurs appartenant à T'_k. Soit Δ_k l'augmentation de la distorsion due au fusionnement de la pair $(\boldsymbol{y}_{i_k}, \boldsymbol{y}_{j_k})$ en un seul vecteur $\boldsymbol{y}'_k$. On a :

$$\Delta_k = d(\boldsymbol{y}'_k) - d(\boldsymbol{y}_{i_k}) - d(\boldsymbol{y}_{j_k}) = \sum_{\boldsymbol{x}\in T'_k} \|\boldsymbol{x} - \boldsymbol{y}'_k\|^2 - \sum_{\boldsymbol{x}\in T_{i_k}} \|\boldsymbol{x} - \boldsymbol{y}_{i_k}\|^2 - \sum_{\boldsymbol{x}\in T_{j_k}} \|\boldsymbol{x} - \boldsymbol{y}_{j_k}\|^2 . \qquad (3.13)$$

Le fusionnement du dictionnaire de taille N : $\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{N\text{-}1}\}$ résulte donc en un nouveau dictionnaire de taille $N/2$: $\mathbf{Y}' = \{\boldsymbol{y}'_0, \boldsymbol{y}'_1, ..., \boldsymbol{y}'_{(N/2)\text{-}1}\}$ où $\boldsymbol{y}'_k$ résulte de la fusion de $\boldsymbol{y}_{i_k}$ et $\boldsymbol{y}_{j_k}$. Soit $D(\mathbf{Y}')$ la distorsion associée au dictionnaire $\mathbf{Y}'$ obtenu par fusion :

$$D(\mathbf{Y}') = \sum_{k=0}^{(N/2)-1} d(\boldsymbol{y}'_k) = \sum_{k=0}^{(N/2)-1} \left[d(\boldsymbol{y}_{i_k}) + d(\boldsymbol{y}_{j_k}) + \Delta_k \right] = D(\mathbf{Y}) + \sum_{k=0}^{(N/2)-1} \Delta_k \ , \qquad (3.14)$$

où $D(\mathbf{Y})$ est la distorsion associée au dictionnaire initial (à fusionner) et est connue. Il s'agit donc de minimiser la distorsion (3.14) utilisant la séquence d'apprentissage et tous les dictionnaires possibles $\mathbf{Y}'$, de taille $N/2$, obtenus par fusion du dictionnaire $\mathbf{Y}$ de taille N. Par exemple, soit un dictionnaire initial (non-structuré) de taille 8 : $\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_7\}$ et le dictionnaire fusionné de taille 4 : $\mathbf{Y}' = \{\boldsymbol{y}'_0, \boldsymbol{y}'_1, \boldsymbol{y}'_2, \boldsymbol{y}'_3\}$ où les vecteurs sont obtenus comme suite (l'opérateur *fusion* désigne l'opération de fusion et le vecteur d'indice k est obtenu par fusion des vecteurs d'indices i_k et j_k) : $\boldsymbol{y}'_0 = \mathit{fusion}(\boldsymbol{y}_0, \boldsymbol{y}_4)$, $\boldsymbol{y}'_1 = \mathit{fusion}(\boldsymbol{y}_2, \boldsymbol{y}_6)$, $\boldsymbol{y}'_2 = \mathit{fusion}(\boldsymbol{y}_1, \boldsymbol{y}_5)$, $\boldsymbol{y}'_3 = \mathit{fusion}(\boldsymbol{y}_3, \boldsymbol{y}_7)$. La séquence de paires d'indices correspondante est donnée alors par: $\{(i_0, j_0), (i_1, j_1), (i_2, j_2), (i_3, j_3)\} = \{(0,4), (2,6), (1,5), (3,7)\}$. L'équation (3.14) devient alors $D(\mathbf{Y}') = D(\mathbf{Y}) + \Delta_0 + \Delta_1 + \Delta_2 + \Delta_3$.

3-Une fois le dictionnaire optimal $\mathbf{Y}'_{opt}$ trouvé, le processus est répété : c'est-à-dire qu'un dictionnaire de taille $N/4$ (minimisant le critère (3.14)) est obtenu par fusion des vecteurs-code du dictionnaire $\mathbf{Y}'_{opt}$ de taille $N/2$. Le processus est itéré jusqu'à obtenir le dictionnaire racine composé d'un seul vecteur-code. Les dictionnaires ainsi obtenus (le dictionnaire initial inclus) de tailles N, $N/2$, $N/4$,…, 4, 2 et 1 contiennent les vecteurs-code qui seront associés aux nœuds des niveaux $\log_2 N$, $\log_2(N/2)$, $\log_2(N/4)$,…,2,1 et 0 respectivement de l'arbre. Notons qu'une ré-indexation des vecteurs-code du dictionnaire initiale ainsi que ceux des dictionnaires intermédiaires est nécessaire pour que ces vecteurs-code correspondent aux nœuds de l'arbre binaire (équilibré) ainsi construit : le vecteur-code (nœud) d'indice i du niveau m ($\boldsymbol{c}_{m,i}$) est obtenu par fusion des deux vecteurs d'indices $2i$ et $2i+1$ du niveau $m+1$ de l'arbre ($\boldsymbol{c}_{m+1,2i}$ et $\boldsymbol{c}_{m+1,2i+1}$). Après ré-indexation les dictionnaires ainsi réarrangés correspondent aux dictionnaires $\boldsymbol{C}_L$, $\boldsymbol{C}_{L-1}$, …, $\boldsymbol{C}_0$ avec $L = \log_2 N$ de la figure (3.1-(b)).

Pour une séquence d'apprentissage et un dictionnaire donné de taille *N,* le problème se pose donc de la manière suivante : Quelles cellules (régions) faut' il fusionner pour construire un arbre binaire équilibré de telle manière que chaque dictionnaire intermédiaire (appelé dictionnaire-arbre dans la figure 3.1-(b)), correspondant à chaque niveau de l'arbre, soit de distorsion minimale. Rappelons que pour trouver un dictionnaire intermédiaire $\mathbf{Y}' = \{\boldsymbol{y}'_0, \boldsymbol{y}'_1, ..., \boldsymbol{y}'_{(N/2)-1}\}$ à partir du dictionnaire initial $\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{N-1}\}$ cela revient à chercher la séquence de paires d'indices $\{(i_k, j_k), k = 0, 1, ..., N/2-1\}$ avec $i_k \neq j_k$ et $\underset{k=0 \text{ à } (N/2)-1}{\cup}(i_k, j_k) = \{0, 1, ..., N-1\}$ où (i_k, j_k), $0 \le i_k, j_k < N$, sont les indices des deux vecteurs (issus du dictionnaire $\mathbf{Y}$) à fusionner pour l'obtention du vecteur d'indice k, $0 \le k < N/2$, du dictionnaire $\mathbf{Y}'$. Différentes techniques ont été proposées par Riskin et al [15] :

• La quantification vectorielle à dictionnaires ordonnés décrite en [82, 83]. Dans [15] l'algorithme LBG, utilisant la règle de dichotomie (splitting) [9] pour l'initialisation du dictionnaire, est utilisé. Le dictionnaire obtenu est ordonné; c'est-à-dire que le vecteur d'indice i est proche (dans l'espace d'entré) du vecteur d'indice $i+1$. Donc, partant

d'un dictionnaire ordonné initial de taille N (indexés de 0 à $N-1$) la séquence des pairs d'indices $\{(0,1),(2,3),...,(N-2,N-1)\}$ est utilisée pour obtenir le dictionnaire de taille $N/2$. De même la séquence $\{(0,1),(2,3),...,(N/2-2,N/2-1)\}$ est utilisée pour obtenir le dictionnaire de taille $N/4$ à partir du dictionnaire de taille $N/2$. Et ainsi de suite.

• Correspondance parfaite à coût minimal (MCPM : Minimum Cost Perfect Matching) : La méthode MCPM est originaire de la théorie d'optimisation [84] et a été appliquée en traitement d'image dans [15] pour la technique de fusion. Dans cette méthode un graphe complet de N nœuds indexés de 0 à $N-1$ est utilisé. Il s'agit de trouver une séquence de paires d'indices $\{(i_k, j_k), k=0,1,...(N/2)-1\}$ avec $i_k \neq j_k$ et $\underset{k=0 \text{ à } (N/2)-1}{\cup}(i_k, j_k)=\{0,1,...,N-1\}$ où (i_k, j_k) correspond aux indices de deux nœuds (connectés) du graphe. Une fonction de coût symétrique $\text{cost}(i_k, j_k)$ est associé à chaque paire de nœuds (i_k, j_k). Pour une certaine connexion du graphe, un coût total est associé : $\sum_{k=0}^{(N/2)-1}\text{cost}(i_k, j_k)$. Pour que les algorithmes proposés en MCPM puissent être utilisés dans la technique de fusion de cellules, il suffit donc que les indices des nœuds du graphe correspondent à ceux des vecteurs du dictionnaire (à fusionner) et que le coût soit défini par : $\text{cost}(i_k, j_k)=\Delta_k$ (dans ce cas le coût total est semblable au critère (3.14).

Récemment, Chu dans [66] a proposé un algorithme, à faible complexité, d'élaboration de dictionnaires pour la conception d'un quantificateur MTVQ pour la quantification vectorielle incorporé des coefficients LSF. Cet algorithme de conception se compose de deux étapes : D'abord, les dictionnaires-étage sont conçus; en second lieu un procédé de conception sous-optimal est appliqué pour concevoir les dictionnaires-arbre associés à chaque dictionnaire-étage (figure 3.2); ou d'une manière équivalente, concevoir la structure arborescente binaire (équilibrée) associée à chaque étage (de la structure MTVQ). Le procédé de conception est basé sur la technique de fusion de cellule. Comme pour la méthode proposée par Riskin et al. [15], le procédé est itératif : l'arbre binaire, correspondant à un étage particulier, est construit par niveaux (des niveaux les plus élevés de l'arbre vers les niveaux inférieures) :

1- à partir de la résolution la plus élevée $m=L$ correspondant au dictionnaire-étage $\boldsymbol{C}_L$

de taille $N = 2^L$, un certain nombre de séquences de paires d'indices (de vecteurs-code) candidates au processus du fusionnement : $\{(i_k, j_k), k = 0,1,...,N/2-1\}$, $0 \le i_k, j_k < N$, sont évaluées pour ne retenir que la séquence qui minimise un certain critère (la distorsion total). Donc les meilleurs *N*/2 paires de cellules sont fusionnées et les centroïdes de ces cellules fusionnées sont calculés pour former les *N*/2 vecteurs-code du dictionnaire-arbre de résolution $m = L-1$: $\boldsymbol{C}_{L-1}$.

2- le même procédé est appliqué pour trouver le dictionnaire-arbre de résolution $m = L-2$ à partir du dictionnaire-arbre de résolution $m = L-1$. Le processus est répété jusqu'à ce qu'il atteigne la résolution *m* = 1, où tous les vecteurs-code sont trouvés (le dernier vecteur à la résolution $m = 0$ est calculé comme étant le centroïde des deux cellules trouvées à la résolution $m = 1$).

Nous proposons, dans le cadre de ce projet, un nouvel algorithme d'élaboration de dictionnaires (codebook design) où un arbre binaire est conçu à partir des niveaux supérieurs vers les niveaux inferieurs comme une connexion de sous-arbres optimaux selon un critère d'optimalité donné.

3.5 Un algorithme de construction d'une structure arborescente

Etant donné un arbre binaire équilibré de hauteur (profondeur) *L*, la tâche de concevoir un quantificateur TSVQ consiste à trouver les vecteurs-code qui peuplent le dictionnaire-étage de taille $N = 2^L$ (correspondant aux nœuds-feuilles du niveau *L* de l'arbre) et les dictionnaires-arbre associés de tailles $N/2$, $N/4$, $N/8$, .., 2, 1 correspondants respectivement à l'ensemble des nœuds des niveaux $L-1$, $L-2$, $L-3$, …, 1, 0 de l'arbre binaire (figure 3.1). La pratique courante d'élaboration des dictionnaires consiste à commencer le processus d'élaboration à partir du dictionnaire de plus petite taille vers les dictionnaires de plus grandes tailles. On parle de construction d'une arborescence descendante équilibrée (Paragraphe 3.4.1.1).

Pour une structure MTVQ à *K* étages (figure 3.2), les *K* étages sont conçus séquentiellement (du premier au dernier étage) où le procédé en arborescence descendante est appliqué à chaque étage (qui correspond à un TSVQ). Le problème avec

le procédé de conception en arborescence descendante équilibrée est que les vecteurs-code générés à une résolution (niveau) élevée tendent à être fortement influencés par ceux générés aux résolutions inférieures; par conséquent le dictionnaire-étage est sous optimal puisque contient moins de représentants pertinents que dans le cas d'un dictionnaire non-structuré (de meilleurs résultats peuvent être obtenus quand les vecteurs-code correspondants au dictionnaire-étage sont optimisés indépendamment). De même pour une structure MTVQ à K étages, de meilleurs résultats sont obtenus lorsque les K dictionnaires-étage sont optimisés conjointement sans tenir compte des dictionnaires-arbres associés. Une fois les dictionnaires-étage optimisés, les dictionnaires-arbre associés à chaque dictionnaire-étage sont alors construits. On parle de construction d'arborescence ascendante. Notons que dans notre cas seuls les dictionnaires-étage sont utilisés pour la transmission. Les dictionnaires-arbre peuvent être utilisés pour accélérer le processus de l'encodage. Au niveau du décodeur, les dictionnaires-arbre son utilisés pour récupérer une version approximée du vecteur quantifié (quantification incorporée, Paragraphe 3.3). Nous proposons un algorithme de construction d'une arborescence ascendante basé sur la technique de fusion de cellules [85].

3.5.1 Principe

Considérons un quantificateur vectoriel à un seul étage avec un dictionnaire-étage de taille N (de résolution $r = \log_2 N$) :

$$\mathbf{Y} = \{\boldsymbol{y}_0, \boldsymbol{y}_1, ..., \boldsymbol{y}_{N-1}\}. \tag{3.15}$$

Pour construire l'arbre binaire équilibré, les vecteurs-code $\boldsymbol{y}_i$ $(i = 0 \text{ to } N-1)$ sont placés aux positions des nœuds-feuille (le niveau le plus élevé) de l'arbre. Ceci est un procédé d'assignement d'indices décrit par [66] :

$$\boldsymbol{c}_{r,i} = \boldsymbol{y}_{a[i,k]}, \quad i = 0 \text{ to } N-1, \tag{3.16}$$

où la notation $\boldsymbol{c}_{r,i}$ indique que les vecteurs-code $\boldsymbol{y}_{a[i,k]}$ sont placés au niveau r ($r = \log_2 N$) de l'arbre binaire. $a[i,k] \in [0, N-1]$ est désigné comme la séquence

d'assignement d'indices avec $k = 0$ to $N!-1$, puisque avec N indices il y a $N!$ permutations (séquences). La figure 3.7 montre un exemple d'un arbre binaire de hauteur $L = 3$ correspondant à un dictionnaire-étage de taille $N = 8$.

Pour une séquence d'assignement d'indices donnée, le processus de fusion de cellules consiste à fusionner les cellules de résolutions plus élevées pour former les cellules de résolutions inférieures, comme suit [66] :

$$\mathbf{R}_i^m = \mathbf{R}_{2i}^{m+1} \cup \mathbf{R}_{2i+1}^{m+1}, \tag{3.17}$$

avec $m = r-1$ à 0 et $i = 0$ à $2^m - 1$. Les vecteurs-code $\boldsymbol{c}_{m,i}$ peuvent alors être calculés comme les centroïdes des cellules $\mathbf{R}_i^m$. Pour la mesure de performance, on peut se baser sur le critère proposé par Chu en [66] et qui peut être exprimé comme suit :

$$D = \sum_{m=0}^{r-1} \sum_{i=0}^{2^m-1} \left[P_{2i}^{m+1}\, d(\boldsymbol{c}_{m,i}, \boldsymbol{c}_{m+1,2i}) + P_{2i+1}^{m+1}\, d(\boldsymbol{c}_{m,i}, \boldsymbol{c}_{m+1,2i+1}) \right], \tag{3.18}$$

où P_i^m est la probabilité du vecteur-code $\boldsymbol{c}_{m,i}$ (définie comme la probabilité qu'un vecteur aléatoire appartienne à la cellule $\mathbf{R}_i^m$ associée au vecteur-code $\boldsymbol{c}_{m,i}$) avec $P_i^m = P_{2i}^{m+1} + P_{2i+1}^{m+1}$ et $d(\boldsymbol{x}, \boldsymbol{y})$ est la distance entre les vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$.

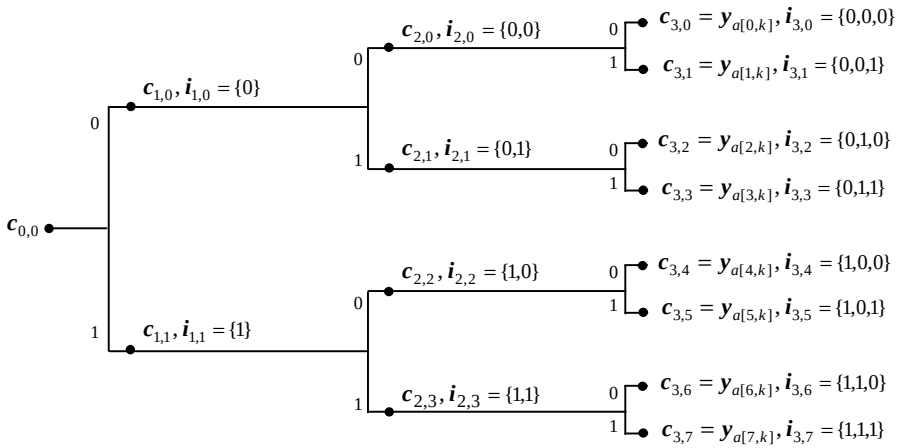


Figure 3.7 : Exemple d'un arbre binaire de résolution *r* = 3 bits. Le *j* ième vecteur-code au niveau (résolution) *m* est noté par $\boldsymbol{c}_{m,j}$ avec l'indice noté par $\boldsymbol{i}_{m,j}$.

Une manière simple pour l'élaboration des dictionnaires consiste à évaluer exhaustivement toutes les séquences possibles d'assignement d'indices et de retenir la meilleure séquence qui correspond à celle qui minimise le critère (3.18). Cette stratégie est appelée *recherche exhaustive conjointe d'assignement d'indices* et le nombre de séquences qui doivent être évalués est donné par [66] :

$$N1 = \prod_{i=0}^{\log_2(N/2)} \left[\frac{(N/2^i)!}{2((N/2^{i+1})!)^2} \right]^{2^i} \quad ; \quad N \geq 2 . \tag{3.19}$$

Pour $N \leq 8$, la valeur de $N1$ est relativement faible ($N1 \leq 315$). Cependant, dans la pratique, la taille N du dictionnaire-étage peut être relativement élevée et la procédure de recherche exhaustive conjointe devient impraticable (par exemple: $N1 = 6385.10^8$ pour $N = 16$). Le problème peut être résolu en divisant le dictionnaire-étage de taille N en N/n sous-dictionnaires de taille n chacun avec $n \leq 8$. Pour chaque sous-dictionnaire et utilisant la recherche exhaustive conjointe, le sous-arbre optimal (selon le critère (3.18)) est trouvé. La figure 3.8 donne un exemple de construction d'un arbre où l'ensemble des nœuds-feuille correspond à un dictionnaire-étage de taille N = 32.

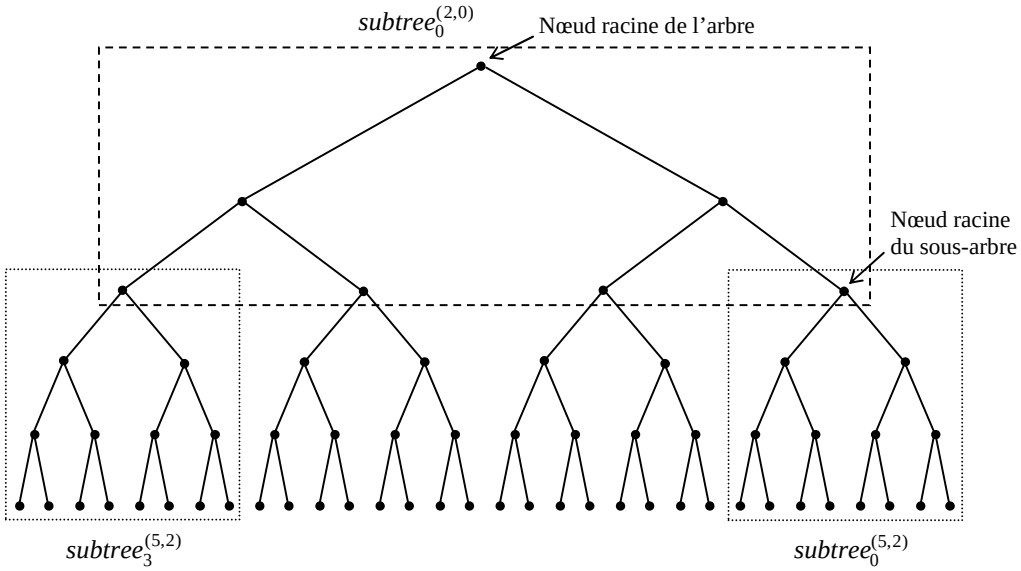


Figure 3.8 : Exemple du principe de construction d'un arbre binaire de hauteur 5 (du niveau 0 à 5) en divisant l'arbre en sous-arbres avec une hauteur maximale de 3 (du niveau 0 à 3) chacun.

L'objectif (pour l'exemple de la figure 3.8) est de construire un arbre de six niveaux : du niveau $l = 0$ (composé d'un seul vecteur-code) au niveau $l = 5$ (composé de 32 vecteurs-code). À partir du dictionnaire-étage de taille $N = 32$, quatre sous-dictionnaires de taille $n = 8$ chacun sont trouvés; puis les quatre sous-arbres optimaux (utilisant la recherche exhaustive conjointe) correspondants aux quatre sous-dictionnaires sont construits : $subtree_i^{(5,2)}$ ($i = 0$ to 3) où $subtree_i^{(p,q)}$ est le sous-arbre construit du niveau *p* au niveau *q* de l'arbre. Ainsi le sous-arbre $subtree_i^{(p,q)}$ a une hauteur de $L' = p - q$ (du niveau 0 à L'). Une fois que les sous-arbres $subtree_i^{(5,2)}$ ($i = 0$ à 3) sont construits, les quatre vecteurs-code $\boldsymbol{c}_{2,i}$ ($i = 0$ à 3) correspondants aux nœuds du niveau $l = 2$ de l'arbre sont calculés et la procédure de recherche exhaustive conjointe peut être appliquée (puisque le nombre de vecteurs-code est faible) pour construire le sous-arbre optimal $subtree_0^{(2,0)}$. La dernière étape de construction est l'opération d'assignement des indices, qui consiste à affecter les vecteurs-code aux nœuds de l'arbre binaire équilibré ainsi construit. Le procédé décrit ci-dessus peut être généralisé pour un arbre binaire de hauteur quelconque. Le processus de conception est répété (des niveaux supérieurs vers les niveaux inferieurs de l'arbre) jusqu'à ce que l'arbre binaire soit conçu en totalité.

Pour un quantificateur MTVQ à *K* étages, la structure est conçue en deux étapes : en premier lieu les *K* dictionnaires-étage (correspondants aux niveaux les plus élevés des *K* arbres binaires) sont construits et optimisés conjointement [13, 14]; en seconde étape, la construction de la structure MTVQ est complétée de façon séquentielle en utilisant la méthode décrite plus haut pour compléter la construction de l'arbre binaire équilibré associé à chaque étage (du premier au dernier étage). La performance de la procédure décrite ci-dessus dépend de la performance de l'algorithme qui consiste à diviser un ensemble de *N* vecteurs-code en N / n sous-ensembles de n vecteurs-code chacun. Nous proposons, ci-dessous, un algorithme efficace désigné sous le nom d'algorithme de groupage.

3.5.2 Algorithme de groupage

Soit $I_i^{(n)}$ un ensemble (d'indice *i*) de *n* indices de vecteurs-code sélectionnés parmi N vecteurs (indexés de 0 à $N-1$). Le nombre d'ensembles d'indices distincts (les n indices contenus dans l'ensemble $I_i^{(n)}$ doivent être différents et l'ordre des indices n'est pas pertinent) est donné par:

$$N2 = \frac{N!}{n!(N\text{-}n)!}. \tag{3.20}$$

Nous définissons la distance associée à l'ensemble d'indices $I_i^{(n)}$ par :

$$D[i,n] = \sum_{j \in I_i^{(n)}} P_j (\boldsymbol{c}_i - \boldsymbol{y}_j)^T \mathbf{W}(\boldsymbol{y}_j)(\boldsymbol{c}_i - \boldsymbol{y}_j), \tag{3.21}$$

où P_j est la probabilité du vecteur-code $\boldsymbol{y}_j$, qui est définie comme étant la probabilité que le vecteur aléatoire $\boldsymbol{X}$ appartienne à la cellule $\boldsymbol{R}_j$ (ayant le vecteur $\boldsymbol{y}_j$ comme centroïde). $\mathbf{W}(\boldsymbol{x})$ est une matrice de pondération diagonale dépendant du vecteur $\boldsymbol{x}$. Donc $D[i,n]$ est la somme des distances Euclidiennes pondérées et graduées entre les vecteurs-code $\boldsymbol{y}_j$ ($j \in I_i^{(n)}$) et le vecteur $\boldsymbol{c}_i$. Lorsque la distance Euclidienne est utilisée, la matrice $\mathbf{W}$ devient la matrice identité. En utilisant un ensemble suffisamment grand de N_t vecteurs d'apprentissage, le facteur de graduation P_j est estimé par :

$$P_j = P\{\boldsymbol{Y} \in \boldsymbol{R}_j\} = \frac{\left\|\boldsymbol{R}_j\right\|}{N_t}, \tag{3.22}$$

où $\left\|\boldsymbol{R}_j\right\|$ désigne le cardinal de la cellule $\boldsymbol{R}_j$, c'est à dire, le nombre de vecteurs appartenant à cette cellule. Les probabilités sont introduites dans l'équation (3.21) pour donner plus de poids aux cellules les plus larges, puisqu'elles ont une plus grande contribution à la somme des distances. En minimisant (3.21), le centroïde $\boldsymbol{c}_i$ est donné par :

$$\boldsymbol{c}_i = \left[\sum_{j \in I_i^{(n)}} P_j \mathbf{W}(\boldsymbol{y}_j) \right]^{-1} \left[\sum_{j \in I_i^{(n)}} P_j \mathbf{W}(\boldsymbol{y}_j) \boldsymbol{y}_j \right] . \tag{3.23}$$

Le vecteur $\boldsymbol{c}_i$ peut être interprété comme le centroïde de la cellule résultante de la fusion des n cellules ayant comme centroïdes les vecteurs $\boldsymbol{y}_j$ ($j \in I_i^{(n)}$). Une séquence de *N/n* ensembles (d'indices) disjoints est définie par :

$$S_k^{(n)} = \left\{ I_{b[i,k]}^{(n)},\ i = 0 \text{ à } (N/n) - 1 \right\} \tag{3.24}$$

où $b[i,k] \in [0, N2-1]$ avec k est l'indice de la séquence et $N2$ est le nombre d'ensembles d'indices distincts donné par l'équation (3.20). Évidemment nous avons :

$$\bigcup_{i=0,..,(N/n)-1} I_{b[i,k]}^{(n)} = \{0,1,..., N-1\} . \tag{3.25}$$

Par la suite, on désignera le paramètre N/n par la taille de la séquence $S_k^{(n)}$ et le paramètre n par la dimension de la séquence $S_k^{(n)}$ ainsi que celle des ensembles contenus dans cette séquence. Pour trouver la meilleure séquence, nous minimisons la distorsion totale suivante :

$$D_T[k,n] = \sum_{i=0}^{(N/n)-1} D[b[i,k],n] . \tag{3.26}$$

Selon notre critère d'optimalité (3.26), la meilleure séquence $S_k^{(n)}$ peut être trouvée en faisant une recherche exhaustive sur toutes les séquences possibles (l'ordre des ensembles d'indices dans une séquence n'est pas pertinent) qui produisent différentes valeurs de la distorsion (3.26). Cependant, cette recherche exhaustive est impraticable parce que le nombre de possibilités est astronomique même pour des valeurs modérées de N et n. La complexité élevée, pour la recherche de la séquence optimale, est due à deux paramètres : Le nombre d'ensembles d'indices distincts $N2$ (équation (3.20)) qui dépend de N (nombre de vecteurs-code) et n (dimension de l'ensemble) et le nombre de séquences

distinctes qui correspond à la combinaison de N/n ensembles (symboles) pris parmi $N2$ ensembles avec la contrainte que les N/n ensembles contenus dans une séquence doivent être disjoints pour que l'équation (3.25) soit vérifiée. Par exemple : $N2 \approx 10^{12}$ pour $N = 128$ et $n = 8$. Cependant pour $n = 2$, la valeur de $N2$ est réduite à 8128.

Nous considérons une procédure sous-optimale qui consiste, dans une première étape, à évaluer un nombre limité de séquences (de dimension $n = 2$) pour en sélectionner les M_L séquences ayant les plus faibles distorsions. A partir de ces M_L séquences de dimension $n = 2$, un nombre limité de séquences de dimension $n = 4$ (la dimension suivante) sont construites et évaluées pour ne retenir que les M_L séquences de plus faibles distorsions. Le procédé est répété jusqu'à ce que la dimension désirée $n1$ soit atteinte et que la meilleure séquence qui minimise la distorsion globale (3.26) soit trouvée. L'algorithme est donné au tableau (3.3).

La solution optimale qui consiste à évaluer toutes les séquences possibles $S_k^{(n1)}$ à la dimension désirée $n1$ est impraticable (le nombre de possibilités est astronomique). Pour augmenter la chance de trouver la séquence optimale, M_L séquences ayant les plus faibles distorsions ($M_L > 1$) sont sélectionnées à chaque dimension *n*; puis, pour chaque séquence ainsi sélectionnée, N_L séquences ($N_L > 1$) de dimension 2*n* sont évaluées. Notons que les paramètres N_L et M_L affectent le compromis entre la performance et la complexité de l'algorithme de groupage.

3.5.3 Extraction des séquences

L'extraction de N_L séquences de dimension $2n$: $S_k^{(2n)}$, $k = 0$ to $N_L - 1$ à partir d'une séquence de dimension n : $S_k^{(n)}$ peut être faite en étendant et en généralisant la méthode présentée dans [66] (qui consiste à extraire des séquences d'ensembles de 2 indices chacun à partir d'une séquence d'indices). Supposons une séquence $S_k^{(n)} = \{I_{b[i,k]}^{(n)}, i = 0 \text{ to } (N/n) - 1\}$; un ensemble d'indices de dimension $2n$ est formé par l'opération de réunion entre deux ensembles d'indices, de dimension n chacun, pris parmi les ensembles de la séquence $S_k^{(n)}$ (voir équation (3.27)).

Tableau 3.3: Algorithme de groupage : Divise un ensemble de *N* vecteurs-code en *N*/*n* sous-ensembles de *n* vecteurs-code chacun

• Entrées :

Les vecteurs-code $\boldsymbol{y_j}$ ($j = 0$ to $N-1$),

Les probabilités P_j ($j = 0$ to $N-1$),

Les paramètres: $n1$ (dimension désirée), N_L et M_L avec $M_L < N_L$.

1- Initialisation :

$S_0^{(1)} = \{\{0\},\{1\},...,\{N-1\}\}$, $n \leftarrow 2$.

2- Sélectionner les M_L meilleures séquences de dimension 2

2.1-Extraire N_L séquences :

En utilisant la séquence $S_0^{(1)}$ de dimension 1, extraire un maximum de N_L séquences $S_k^{(2)}$ ($k = 0$ to $N_L - 1$) de dimension 2.

2.2- Sélectionner les M_L meilleures séquences :

En utilisant le critère (3.26), calculer les distorsions $D_T[k,2]$ ($k = 0$ to $N_L - 1$) correspondantes aux N_L séquences extraites à l'étape 2.1; puis retenir les M_L séquences distinctes $S_{k(j)}^{(2)}$ ($j = 0$ to $M_L - 1$) correspondantes aux M_L séquences produisant les plus faibles distorsions.

3- Traitement des dimensions suivantes

3.1- Extraire $N_L M_L$ séquences :

Pour chaque séquence $S_{k(j)}^{(n)}$, extraire un maximum de N_L séquences de dimension $2n$.

Donc nous avons $N_L M_L$ séquences $S_k^{(2n)}$ ($k = 0$ to $N_L M_L - 1$).

3.2- Sélectionner les M_L meilleures séquences:

En utilisant le critère (3.26), calculer les distorsions $D_T[k,2n]$ ($k = 0$ to $N_L M_L - 1$) correspondantes aux $N_L M_L$ séquences de l'étape 3.1 ; puis retenir les M_L séquences distinctes $S_{k(j)}^{(2n)}$ ($j = 0$ to $M_L - 1$) correspondantes aux M_L plus faibles distorsions.

3.3- Test de terminaison :

$n \leftarrow 2n$,

Si $n = n1$, aller à l'étape suivante, autrement répéter les étapes (3.1) à (3.2).

4- Sélection de la meilleure séquence de dimension $n1$ (dimension désirée).

Parmi les M_L séquences sélectionnées à l'étape 3.2, retenir la séquence $S_K^{(n1)}$ qui minimise la distorsion globale $D_T[k,n1]$.

• Sortie :

La séquence de $N/n1$ ensembles de $n1$ indices chacun: $S_K^{(n1)} = \{I_{b[i,K]}^{(n1)},\ i = 0 \text{ to } (N/n1)-1\}$

avec $\underset{i=0,..,(N/n1)-1}{\cup} I_{b[i,K]}^{(n1)} = \{0,1,...,N-1\}$.

$$\{I_{b[i,k]}^{(n)} \cup I_{b[j,k]}^{(n)}\}, \quad i = 0,..., (N/n) - 2, \quad j = i + 1,..., (N/n) - 1. \tag{3.27}$$

Les indices i et j sont choisis de telle manière que les indices contenus dans l'ensemble de dimension $2n$, résultant de l'opération (3.27), doivent être différents. Ainsi nous avons un total de $N3 = (N/n)!/\big(2!((N/n)\text{-}2)!\big)$, une combinaison de 2 symboles pris parmi N/n symboles, ensembles d'indices distincts de dimension $2n$: $I_i^{(2n)}$ $(i = 0,..., N3-1)$. En utilisant l'équation (3.21), les distances $D[i,2n]$ $(i = 0,..., N3-1)$ associées aux ensembles $I_i^{(2n)}$ $(i = 0,..., N3-1)$ sont calculées. Les distances sont ordonnées dans l'ordre croissant, avec les ensembles associés placés dans le même ordre. Pour les ensembles d'indices ainsi ordonnés et utilisant l'ensemble ayant la plus faible distance (le premier ensemble dans la liste ordonnée) comme référence, on élimine tous les autres ensembles de la liste qui ne sont pas disjoints avec la référence. On continue le processus avec le prochain ensemble dans la liste ordonnée jusqu'au point où tous les ensembles restants dans la liste soient disjoints. À la fin du processus une seule séquence de $N/2n$ ensembles reste et forme la première séquence de dimension $2n$: $S_0^{(2n)} = \{I_{b[i,0]}^{(2n)}, i = 0,..., (N/2n) - 1\}$. Nous pouvons appliquer la même méthode à plusieurs reprises pour extraire d'autres séquences; nous recommençons avec la liste ordonnée intacte, nous ignorons le premier ensemble et utilisons le deuxième ensemble dans la liste ordonnée comme référence puis nous appliquons le même procédé pour extraire une autre séquence. Le processus est répété N_L fois produisant un maximum de N_L séquences différentes $S_k^{(2n)}$ $(k = 0,..., N_L - 1)$.

3.5.4 Amélioration de l'algorithme de groupage par permutation d'indices

La performance de l'algorithme de groupage peut être améliorée. Considérons une séquence $S_k^{(n)}$ de N/n ensembles d'indices; nous pouvons permuter les indices (un indice à la fois) entre deux ensembles de la séquence $S_k^{(n)}$. Cette opération produit une nouvelle séquence dont la distorsion totale (3.26) est évaluée. L'opération de permutation est retenue si la distorsion totale est minimisée, autrement l'opération est annulée.

Tableau 3.4: Amélioration de l'algorithme de groupage par l'opération de permutation d'indices

• Entrée :

Une séquence $S_k^{(n)} = \{I_{b[i,k]}^{(n)}, i = 0,..., (N/n)-1\}$.

1-Initialisation :

1.1- Calcul de distorsion de la séquence $S_k^{(n)}$:

$$D_T(S_k^{(n)}) \leftarrow \sum_{i=0}^{(N/n)-1} D[b[i,k],n],$$

$i \leftarrow 0$.

1.2- $j \leftarrow i+1$,

2-Permutation d'indices

2.1-Permutation d'indices entre deux ensembles

Permuter un indice à la fois entre les ensembles $I_{b[i,k]}^{(n)}$ et $I_{b[j,k]}^{(n)}$. Ceci produit deux nouveaux ensembles, donc une nouvelle séquence $S_k'^{(n)} = \{I_{b[i,k]}'^{(n)}, i = 0,..., (N/n)-1\}$.

2.2-Calcul de la distorsion associée à la nouvelle séquence

$$D_T'(S_k'^{(n)}) \leftarrow \sum_{i=0}^{(N/n)-1} D[b[i,k],n].$$

2.3-Tester si l'opération de permutation doit être annulée

Si $D_T' < D_T$ alors $D_T \leftarrow D_T'$ et $S_k^{(n)} \leftarrow S_k'^{(n)}$,

autrement l'opération de permutation faite à l'étape 2.1 est annulée.

2.4-Test :

Si le nombre d'opérations de permutation d'indices faites (entre les deux ensembles) à l'étape 2.1 est égale à n^2, aller à l'étape suivante, autrement répéter les étapes 2.1 à 2.3.

3-Passer aux deux ensembles suivants

3.1-Passer au deuxième ensemble suivant:

$j \leftarrow j+1$,

Si $j < N/n$, alors répéter les étapes 2.1 à 2.4, autrement aller à l'étape suivante.

3.2- Passer au premier ensemble suivant:

$i \leftarrow i+1$,

Si $i < (N/n)-1$, alors répéter les étapes 1.2 à 3.1, autrement arrêter le traitement.

•Sortie :

La séquence $S_k^{(n)} = \{I_{b[i,k]}^{(n)}, i = 0 \text{ to } (N/n)-1\}$ correspondante à la nouvelle séquence de plus faible distorsion.

L'opération de permutation peut être appliquée pour tous les indices d'un ensemble et pour toutes les combinaisons possibles de deux ensembles de la même séquence $S_k^{(n)}$. Le tableau 3.4 décrit cette opération. L'opération de permutation d'indices peut être appliquée à chaque dimension *n* (de l'algorithme de groupage du tableau 3.3) et pour chacune des $N_L M_L$ séquences. Cependant, pour réduire la complexité de calcul, la procédure est

appliquée seulement à la dimension désirée $n1$ (à l'étape 4 de l'algorithme de groupage) pour les M_L séquences ayant les plus faibles distorsions.

3.5.5 Résultats d'évaluation

Pour illustrer la performance de l'algorithme de groupage utilisé conjointement avec la procédure de permutation d'indices, des expériences sont réalisées comme suite :

• Pour la première expérience, on construit un dictionnaire de $N = 32$ vecteurs-code bidimensionnels en appliquant l'algorithme LBG [9] (utilisant la distance Euclidienne) sur des données d'apprentissage de 160,000 échantillons (80.000 vecteurs) générés à partir d'une variable aléatoire gaussienne de moyenne nulle et de variance unitaire. Puis, on applique l'algorithme de groupage avec $N_L = 40$ et $M_L = 8$ (la procédure de permutation est appliquée pour les M_L séquences sectionnées à la dimension désirée) pour trouver les meilleures séquences de dimensions 2, 4, 8 et 16. La figure 3.9 donne le diagramme de Voronoi des 32 vecteurs-code, ainsi que celui : (a) des 16 vecteurs-code correspondant à la séquence $S_K^{(2)}$ de taille 16 et de dimension 2; (b) des 8 vecteurs-code correspondant à la séquence $S_K^{(4)}$ de taille 8 et de dimension 4; (c) des 4 vecteurs-code correspondant à la séquence $S_K^{(8)}$ de taille 4 et de dimension 8; (d) des 2 vecteurs-code correspondant à la séquence $S_K^{(16)}$ de taille 2 et de dimension 16 .

Notons que le vecteur-code associé à l'ensemble d'indices $I_{b[i,K]}^{(n)}$ (de la séquence $S_K^{(n)}$) est le vecteur moyen des n vecteurs-code contenus dans cet ensemble. En effet, dans cette expérience, les 32 vecteurs-code (du dictionnaire) sont considérés ayant la même probabilité et la distance Euclidienne est utilisée. Par conséquent, nous avons $P_j = 1/32$, $\mathbf{W}$ est la matrice identité et $\boldsymbol{c}_i$ correspond au vecteur moyen dans l'équation (3.23). Trouver la séquence $S_K^{(n)}$, de dimension n et de la taille N/n, qui minimise la distorsion (3.26) revient donc à partitionner un ensemble de N vecteurs-code en N/n sous-ensembles de cardinalité n chacun : $I_{b[i,K]}^{(n)}\ (i = 0,..., (N/n) - 1)$.

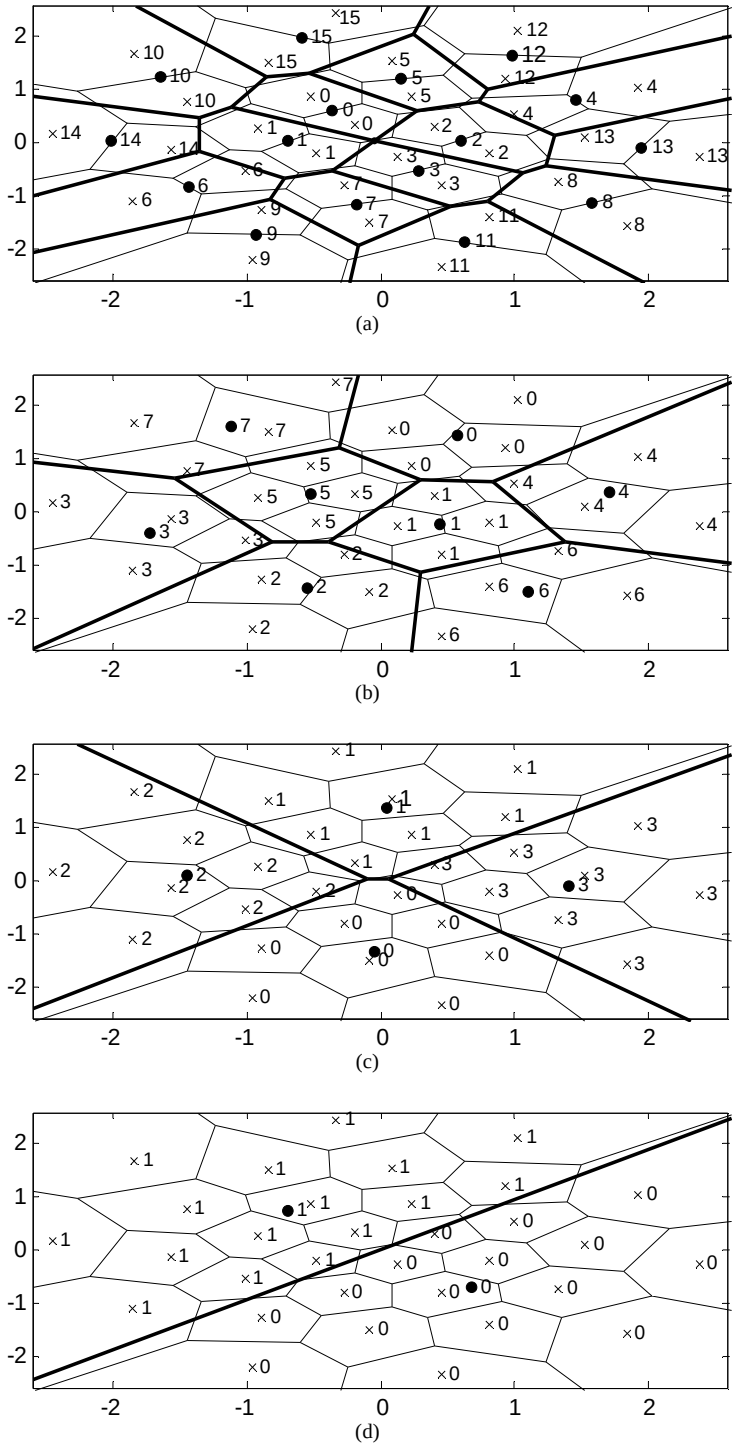


Figure 3.9: Diagrammes de Voronoi des 32 vecteurs gaussiens bidimensionnels marqués en 'x' groupés en: (a) 16 ensembles de 2 vecteurs chacun; (b) 8 ensembles de 4 vecteurs chacun; (c) 4 ensembles de 8 vecteurs chacun et (d) 2 ensembles de 16 vecteurs chacun. Les vecteurs du même ensemble sont marqués du même numéro que leur vecteur moyen (marqué en '•'). L'algorithme de groupage est appliqué avec $N_L = 40$ et $M_L = 8$ (la procédure de permutation d'indices est appliquée aux M_L séquences de plus faibles distorsions sélectionnées à la dimension désirée).

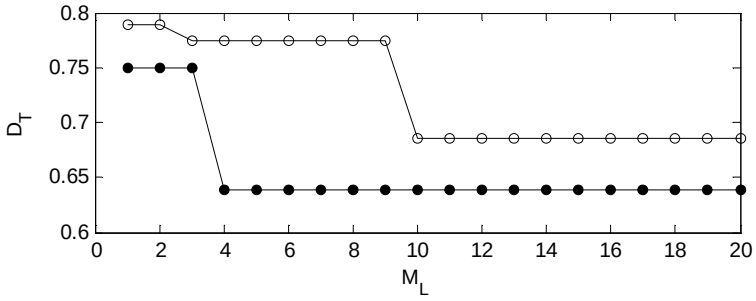


Figure 3.10: Distorsion totale D_T (avec $N = 32$ et $n = 8$) en fonction du nombre de séquences M_L (avec $N_L = 40$). Des vecteurs gaussiens bidimensionnels sont utilises. Pour les points '●' la procédure de permutation d'indices est appliquée aux M_L séquences de plus faibles distorsions de dimension 8 et la meilleure (de plus faible distorsion) séquence est sélectionnée.

• Pour la deuxième expérience, un dictionnaire de 32 vecteurs-code bidimensionnels est conçu en utilisant l'algorithme LBG (avec la distance Euclidienne) où chaque vecteur-code a une certaine probabilité. Les 32 vecteurs-code sont groupés en 4 ensembles de 8 vecteurs-code chacun (correspondant à une séquence de dimension 8 et de taille 4) en utilisant l'algorithme de groupage avec $N_L = 40$ et $M_L = 1,..., 20$. La figure 3.10 donne la distorsion totale, calculée par l'équation (3.26), en fonction des valeurs du paramètre M_L. Nous pouvons apprécier l'avantage d'inclure plus d'une séquence ($M_L > 1$) à chaque dimension et la procédure de permutation d'indices. La procédure de permutation d'indices (tableau 3.4) peut améliorer de manière significative la solution finale même si les valeurs des paramètres N_L et/ou M_L ne sont pas suffisamment élevées.

3.6 Conclusion

Un algorithme efficace de conception d'un quantificateur TSVQ binaire équilibré basé sur la technique de fusion de cellules est proposé. Dans cette méthode le dictionnaire-étage correspondant à l'ensemble des vecteurs-code (nœuds) du niveau le plus élevé de l'arbre binaire est d'abord conçu. La conception de la structure arborescente binaire se compose de deux étapes : D'abord, les *N* vecteurs-code issus du dictionnaire-étage sont groupés en *N*/*n* ensembles de *n* vecteurs-code chacun en utilisant l'algorithme de groupage dont l'efficacité est améliorée lorsque utilisé conjointement avec la procédure de permutation d'indices. En second lieu, pour chaque ensemble de *n* vecteurs-

code, le sous-arbre optimal correspondant (de hauteur $\log_2 n$) est conçu en utilisant la méthode de recherche exhaustive conjointe. Le processus de conception est répété (des niveaux supérieurs vers les niveaux inférieurs de l'arbre) jusqu'à ce que l'arbre binaire soit conçu en totalité. Pour un quantificateur MTVQ à *K* étages, la structure est conçue en deux étapes : en premier lieu les *K* dictionnaires-étage (correspondants aux niveaux les plus élevés des *K* arbres binaires) sont construits et optimisés conjointement; en seconde étape, la construction de la structure MTVQ est complétée de façon séquentielle du premier au dernier étage où la méthode décrite plus haut est appliquée pour compléter la construction de l'arbre binaire équilibré associé à chaque étage.

Les méthodes proposées par Riskin [15] et Chu [66], pour la construction d'un arbre binaire équilibré utilisant la technique de fusion de cellules procèdent par un seul niveau à la fois. Une fois les vecteurs-code du niveau le plus élevé de l'arbre trouvés, les niveaux inférieurs sont construits d'une manière séquentielle où le niveau m de l'arbre est construit à partir du niveau $m+1$. La méthode proposée dans ce travail procède d'une façon différente : Une fois le niveau L le plus élevé de l'arbre construit, les $N = 2^L$ vecteurs-code de ce niveau sont groupés en en N/n ensembles de n vecteurs-code ($n \leq 8$) chacun. Puis $\log_2 n + 1$ niveaux inférieurs (un maximum de quatre niveaux) sont conjointement construits comme la connexion de N/n sous-arbres optimaux de hauteur $\log_2 n$ chacun. La même méthode est répétée à partir du niveau $L - \log_2 n$ de l'arbre. Le processus est répété jusqu'à ce que les vecteurs-code de tous les niveaux soient trouvés.
Donc, pour la méthode proposée dans ce travail, un ensemble de niveaux est construit conjointement au lieu d'un niveau à la fois pour les méthodes présentées en [15, 66]. La solution optimale consiste à construire toute la structure arborescente en entier (tous les niveaux de l'arbre sont construits conjointement).

Notons que l'algorithme de groupage (utilisé conjointement avec la procédure de permutation d'indices) peut être utilisé pour une construction descendante d'un arbre binaire équilibré à partir des vecteurs-code d'un dictionnaire non-structuré. La construction commence par le dictionnaire de résolution 0 (niveau 0 de l'arbre) qui correspond au centroïde pondéré (3.23) de tous les vecteurs-code (du dictionnaire non-structuré). À chaque étape, le prochain niveau de l'arbre est construit en partitionnant

chaque ensemble (de vecteurs-code) en deux sous-ensembles de même cardinalité (utilisant l'algorithme de groupage). Par exemple, pour construire le niveau 1 de l'arbre à partir du niveau 0, cela consiste à partitionner l'ensemble des N vecteurs-code du dictionnaire non-structuré en deux sous-dictionnaires de $N/2$ vecteurs-code chacun en utilisant l'algorithme de groupage. Les deux centroïdes des deux sous-dictionnaires forment les deux nœuds du niveau 1 de l'arbre. Puis, chaque dictionnaire de taille $N/2$ est divisé en deux sous-dictionnaires de taille $N/4$ chacun (en appliquant l'algorithme de groupage); les centroïdes des 4 sous-dictionnaires forment les 4 vecteurs-code du niveau 2 de l'arbre. Le processus est repris jusqu'à atteindre le niveau le plus élevé L de l'arbre où les N sous-dictionnaires (de taille 1 chacun) correspondent aux $N = 2^L$ vecteurs-code (du dictionnaire non-structuré) associés aux nœuds-feuille de l'arbre.

Chapitre 4

Procédures de recherche rapide

4.1 Introduction

Pour un quantificateur vectoriel utilisant un dictionnaire non structuré de N vecteurs-code de k composantes chacun, la complexité d'encodage (pour localiser un vecteur-code du dictionnaire) et la complexité de stockage (du dictionnaire) est de l'ordre de 2^{rk}. Il apparait donc que la complexité croit exponentiellement avec la dimension des vecteurs-code k et le débit $r = \log_2(N)/k$ (en bits/dimension). Or de bonnes performances ne peuvent être atteintes qu'aux débits moyens et élevés mais l'opération d'encodage devient alors coûteuse en calculs. Il a donc été nécessaire d'envisager de nouvelles techniques permettant d'atteindre des résultats d'une précision acceptable tout en gardant une complexité de calculs (pour le processus d'encodage) et une complexité de mémorisation (pour le stockage du dictionnaire) raisonnables.

Afin de réduire la complexité de recherche exigée pour localiser des vecteurs-code (issus d'un dictionnaire) durant le processus d'encodage dans une quantification vectorielle, des schémas structurés sont souvent imposés (Chapitre 1). La quantification vectorielle divisée SVQ (split VQ) et la quantification vectorielle multi-étages MSVQ (multistage VQ) correspondent à des schémas structurés permettant une réduction

substantielle en mémoire et en complexité de calcul par comparaison à une quantification vectorielle non structurée [5]. Pour un quantificateur TSVQ, la structure arborescente permet de réaliser des méthodes de recherche rapides et efficaces où la recherche est effectuée par niveaux. Pour une structure MSVQ, la meilleure méthode connue pour la recherche au sein des dictionnaires est la méthode de recherche séquentielle multi-trajets MSS (Multipath Sequential Search) connue aussi sous le nom de la recherche *M-L* [13, 14, 86]. Il a été montré que la performance de la procédure MSS est très proche de celle de la recherche exhaustive avec une complexité de calcul beaucoup plus faible [86]. Pour un quantificateur MTVQ, Chu dans [66, 87] a proposé, récemment, un algorithme de recherche de dictionnaires efficace (pour la quantification des coefficients LSF) désigné sous le nom de recherche séquentielle multi-trajets assistée par une structure arborescente MSTS (Multipath Sequential Tree-assisted Search). Dans ce travail de recherche, nous proposons deux algorithmes de recherche rapide qui réduisent la complexité de calcul des algorithmes MSS et MSTS en utilisant une structure MTVQ ou une structure moins exigeante en charge mémoire.

4.2 Structure à plusieurs étages

Soit une structure MSVQ à K étages avec N_1, N_2, …, N_K les tailles des K dictionnaires. Le processus d'encodage consiste à chercher un ensemble de K indices $\boldsymbol{i} = [i_1, i_2, ..., i_K]$ qui minimise l'erreur de quantification. Différentes stratégies de recherche existent [5, 13, 14, 18].

4.2.1 Recherche exhaustive

La détermination de l'ensemble optimal d'indices $\boldsymbol{i}_o = [i_{o1}, i_{o2}, ..., i_{oK}]$ est un problème d'optimisation combinatoire complexe qui peut être résolu en évaluant toutes les combinaisons possibles de K indices pour ne retenir que celle qui minimise l'erreur de quantification. C'est-à-dire toutes les combinaisons d'indices $\boldsymbol{i} = [i_1, i_2, ..., i_K]$ sont évaluées et la combinaison d'indices optimale $\boldsymbol{i}_o = [i_{o1}, i_{o2}, ..., i_{oK}]$ est celle qui vérifie le critère suivant :

$$d\left(\boldsymbol{x}, \boldsymbol{y}_{i_{o1}}^{(1)} + \boldsymbol{y}_{i_{o2}}^{(2)} + ... + \boldsymbol{y}_{i_{oK}}^{(K)}\right) \leq d\left(\boldsymbol{x}, \boldsymbol{y}_{i_1}^{(1)} + \boldsymbol{y}_{i_2}^{(2)} + ... + \boldsymbol{y}_{i_K}^{(K)}\right) \quad \forall \boldsymbol{i} \neq \boldsymbol{i}_o, \tag{4.1}$$

où $d(\boldsymbol{x}, \boldsymbol{y})$ est la distance (mesure de distorsion) entre les vecteurs $\boldsymbol{x}$ et $\boldsymbol{y}$. $\boldsymbol{y}_{i_k}^{(k)}$ est le vecteur-code (extrait du *k* ième dictionnaire) d'indice i_k. Puisque toutes les combinaisons possibles des *K* indices (parmi les *K* étages) sont évaluées, le nombre total de distances à calculer est donné par :

$$ND = N_1 N_2 ... N_K = \prod_{s=1}^{K} N_s, \tag{4.2}$$

où N_1, $N_2 \ldots$, N_K sont les tailles des *K* dictionnaires correspondant au *K* étages de la structure MSVQ. La complexité de calcul de la recherche exhaustive est très élevée. Pour réduire la complexité de calcul, des solutions séquentielles sous-optimales sont typiquement utilisées dans la pratique.

4.2.2 Recherche séquentielle

La procédure de recherche séquentielle détermine l'ensemble d'indices $\boldsymbol{i}_o = [i_{o1}, i_{o2}, ..., i_{oK}]$ d'une manière séquentielle. Partant du premier étage, l'indice i_{o1} est déterminé tel que :

$$d\left(\boldsymbol{x}, \boldsymbol{y}_{i_{o1}}^{(1)}\right) \leq d\left(\boldsymbol{x}, \boldsymbol{y}_{i_1}^{(1)}\right) \qquad \forall i_1 \neq i_{o1}. \tag{4.3}$$

Pour l'étage suivant (2 ième étage), l'indice i_{o1} est fixe (calculé au premier étage) et l'indice i_{o2} est déterminé tel que :

$$d\left(\boldsymbol{x} - \boldsymbol{y}_{i_{o1}}^{(1)}, \boldsymbol{y}_{i_{o2}}^{(2)}\right) \leq d\left(\boldsymbol{x} - \boldsymbol{y}_{i_{o1}}^{(1)}, \boldsymbol{y}_{i_2}^{(2)}\right), \quad \forall i_2 \neq i_{o2}. \tag{4.4}$$

En général, pour le *j* ième étage, l'indice i_j est tel que :

$$d\left(\boldsymbol{x} - \boldsymbol{y}_{i_{o1}}^{(1)} - \boldsymbol{y}_{i_{o2}}^{(2)} - ... - \boldsymbol{y}_{i_{o(j-1)}}^{(j-1)}, \boldsymbol{y}_{i_{oj}}^{(j)}\right) \leq d\left(\boldsymbol{x} - \boldsymbol{y}_{i_{o1}}^{(1)} - \boldsymbol{y}_{i_{o2}}^{(2)} - ... - \boldsymbol{y}_{i_{o(j-1)}}^{(j-1)}, \boldsymbol{y}_{i_j}^{(j)}\right), \quad \forall i_j \neq i_{oj} \tag{4.5}$$

Les *K* dictionnaires sont parcourus séquentiellement, le nombre total de distances à calculer est de:

$$ND = N_1 + N_2 + ... + N_K = \sum_{s=1}^{K} N_s \ . \tag{4.6}$$

La complexité de calcul de la procédure de recherche séquentielle est faible; cependant la performance peut être améliorée en sélectionnant un ensemble d'indices à chaque étape.

4.2.3 Recherche en arbre

La recherche en arbre (tree search) est connue aussi sous le nom de recherche *M-L* (*M-L* search) ou encore recherche séquentielle multi-trajets MSS (multipath sequential search) [13, 14, 18, 86]. La procédure de recherche en arbre est une généralisation de la procédure de recherche séquentielle dans laquelle un ensemble d'indices (non pas un seul indice) est sélectionné (à chaque étage) et est passé d'un étage à l'autre. Au premier étage, l'ensemble des M_a indices correspondants aux M_a distorsions minimales est déterminé. En notant cet ensemble par $\boldsymbol{I}_1$, on a :

$$i_{o1} \in \boldsymbol{I}_1 \quad \text{si et seulement si} \quad d\left(\boldsymbol{x}, \boldsymbol{y}_{i_{o1}}^{(1)}\right) \le d\left(\boldsymbol{x}, \boldsymbol{y}_{i_1}^{(1)}\right), \quad \forall\, i_1 \notin \boldsymbol{I}_1. \tag{4.7}$$

Notons qu'au premier étage N_1 calculs de distances sont requis. Au deuxième étage, une recherche conjointe est effectuée avec la contrainte que le premier indice appartient à $\boldsymbol{I}_1$. Les M_a paires d'indices correspondants aux M_a distances minimales sont maintenus. En notant par $\boldsymbol{I}_2$ cet ensemble de M_a paires d'indices, on a :

$$\begin{aligned}(i_{o1}, i_{o2}) \in \boldsymbol{I}_2 \quad \text{si et seulement si} \quad & d\left(\boldsymbol{x} - \boldsymbol{y}_{i_{o1}}^{(1)}, \boldsymbol{y}_{i_{o2}}^{(2)}\right) \le d\left(\boldsymbol{x} - \boldsymbol{y}_{i_{o1}}^{(1)}, \boldsymbol{y}_{i_2}^{(2)}\right), \\ & \forall i_{o1} \in \boldsymbol{I}_1, \ \ (i_{o1}, i_2) \notin \boldsymbol{I}_2\end{aligned} \tag{4.8}$$

En supposant que $M_a \le N_1$, au deuxième étage $M_a N_2$ calculs de distances sont requis.

De même, au *j* ième étage, l'ensemble des M_a *j*-plets d'indices $\boldsymbol{I}_j$ est déterminé comme:

$$(i_{o1},...,i_{0j}) \in \boldsymbol{I}_j \quad \text{si et seulement si} \quad d\left(\boldsymbol{x} - \sum_{l=1}^{j-1} \boldsymbol{y}_{i_{ol}}^{(l)}, \boldsymbol{y}_{i_{oj}}^{(j)}\right) \le d\left(\boldsymbol{x} - \sum_{l=1}^{j-1} \boldsymbol{y}_{i_{ol}}^{(l)}, \boldsymbol{y}_{i_j}^{(j)}\right). \qquad (4.9)$$
$$\forall (i_{o1},...,i_{o(j-1)}) \in \boldsymbol{I}_{j-1},\ (i_{o1},...,i_{o(j-1)}, i_j) \notin \boldsymbol{I}_j$$

En supposant que $M_a \le N_{j-1}$ alors $M_a N_j$ calculs de distances sont requis. Au *K* ième étage (étage finale), le *K*-plet $[i_{o1},...,i_{oK}]$, extrait de l'ensemble $\boldsymbol{I}_K$, qui minimise la distorsion totale est choisi comme l'optimum global.

La figure (4.1) donne un exemple d'une structure MSVQ de trois étages avec $M_a = 3$. Les trois dictionnaires sont de tailles $N_1 = 8$ $(i_1 = 0,...,7)$, $N_2 = 16$ $(i_2 = 0,...,15)$ et $N_3 = 4$ $(i_3 = 0,...,3)$.

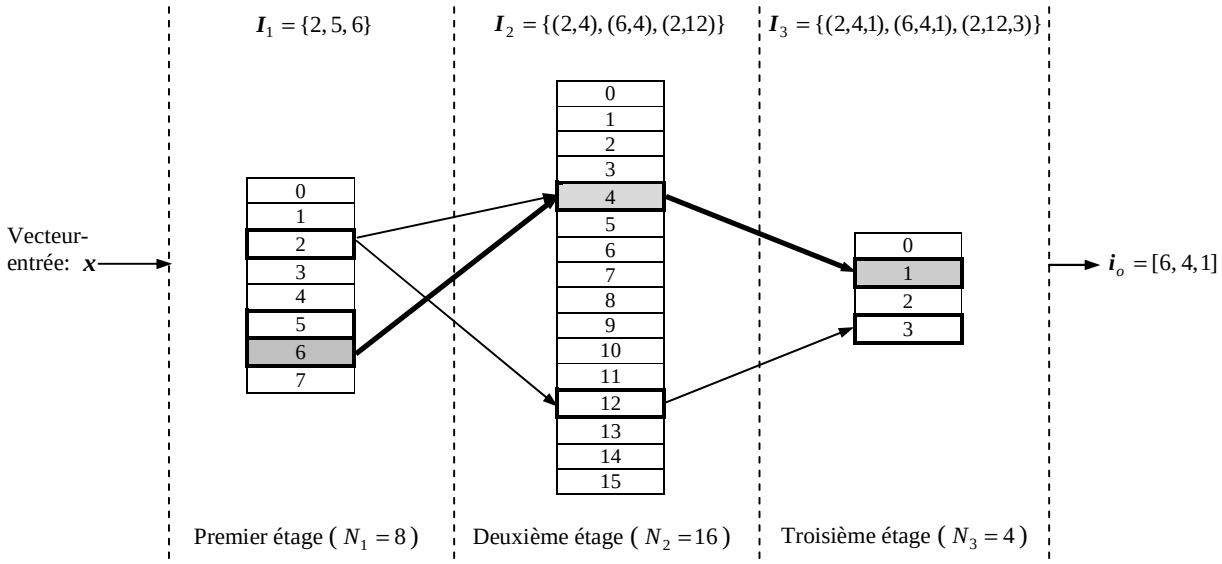


Figure 4.1: Illustration de la procédure de recherche en arbre utilisant une structure MSVQ de trois étages correspondants à trois dictionnaires de tailles $N_1 = 8$, $N_2 = 16$ et $N_3 = 4$. $\boldsymbol{I}_3$ contient les trois meilleurs chemins ($M_a = 3$) du premier au troisième étage. Les trois chemins sont indiqués par des flèches liants les (indices des) vecteurs-code. Les vecteurs-code sélectionnés sont marqués par des cadres en traits épais. Le meilleur chemin est indiqué par des flèches épaisses. Le résultat final est un ensemble de trois indices correspondant aux trois vecteurs-code (marqués par des cadres pleins en gris) extraits du premier, du deuxième et du troisième dictionnaires.

Au premier étage, $\boldsymbol{I}_1 = \{2, 5, 6\}$ si les trois indices $i_{o1} = 2, 5, 6$ produisent les trois plus faibles valeurs de la distorsion (l'inégalité (4.7) est vérifiée pour les trois indices). Au deuxième étage, 48 paires d'indices (i_1, i_2): $(2, 0)$ à $(2, 15)$, $(5, 0)$ à $(5, 15)$ et $(6, 0)$ à

$(6,15)$ sont à considérer et les trois paires qui vérifient l'inégalité (4.8) sont retenues; soit $\boldsymbol{I}_2 = \{(2,4),(6,4),(2,12)\}$. Au troisième étage, 12 triplets d'indices (i_1, i_2, i_3): $(2,4,0)$ à $(2,4,3)$, $(6,4,0)$ à $(6,4,3)$ et $(2,12,0)$ à $(2,12,3)$ sont à considérer et $\boldsymbol{I}_3 = \{(2,4,1),(6,4,1),(2,12,3)\}$ si ces trois triplets produisent les plus faibles distorsions (inégalité (4.9)). Le triplet $\boldsymbol{i}_o = [6,4,1]$ est l'optimum global s'il produit la plus petite distorsion et le vecteur entrée $\boldsymbol{x}$ aura pour vecteur quantifié : $\boldsymbol{y}_6^{(1)} + \boldsymbol{y}_4^{(2)} + \boldsymbol{y}_1^{(3)}$.

Le nombre de calculs de distances requis pour la procédure MSS est donné par :

$$ND = N_1 + M_a \sum_{s=2}^{K} N_s \ , \quad \text{si} \quad M_a < \min_s N_s \, . \tag{4.10}$$

La complexité de mémorisation pour une structure MSVQ de *K* étages est donnée par :

$$MC = p \sum_{s=1}^{K} N_s \ , \tag{4.11}$$

où p est la dimension des vecteurs-code. Chaque composant d'un vecteur-code exige une unité de mémoire (un réel).

4.3 Structure arborescente

Pour une structure arborescente (correspondant à la structure d'un quantificateur TSVQ) la recherche est effectuée par niveaux. Pour chaque niveau, un sous ensemble important de vecteurs candidats est écarté du processus de recherche [5, 15, 67].

4.3.1 Recherche à un seul chemin

Pour effectuer sa recherche, l'encodeur dispose de l'arborescence et démarre sa recherche à partir de la racine de l'arbre (le niveau $l = 0$). À chaque étape, on calcule la distance du vecteur-entrée $\boldsymbol{x}$ par rapport à chacun des enfants (descendants) du nœud courant et on sélectionne le nœud minimisant la distorsion (ce nœud détermine le chemin à choisir pour atteindre le prochain niveau de l'arbre); la recherche se poursuit ensuite

dans le sous-arbre ayant ce nœud comme racine et ce processus est itéré jusqu'à ce que l'on atteigne un nœud terminal de l'arbre. Le vecteur-code associé à ce nœud terminal est alors considéré comme le représentant du vecteur-entrée $\boldsymbol{x}$ [5]. L'inconvénient de l'algorithme de recherche à un seul chemin est que les vecteurs-code (correspondant aux nœuds-feuille) sélectionnés ne sont pas, en général, les plus proches des vecteurs-entrée. Pour réduire la distorsion entre le vecteur-entrée et le vecteur-code associé il a été proposé un algorithme de recherche à plusieurs chemins [15, 67].

4.3.2 Recherche multi-trajet (à plusieurs chemins)

Disposant de la structure arborescente, la recherche démarre à partir du niveau l ($l>1$) de l'arbre. On calcule la distance du vecteur-entrée $\boldsymbol{x}$ par rapport aux η ($\eta = 2^l$) nœuds (vecteurs-code) du niveau l et on sélectionne les M_t ($1 < M_t < \eta$) nœuds qui produisent les plus faibles distorsions. Ces M_t nœuds pointent vers $2M_t$ nœuds du niveau $l+1$ (prochain niveau) de l'arbre binaire. On calcule la distance du vecteur-entrée par rapport à ces $2M_t$ nœuds du niveau $l+1$ et on sélectionne les M_t nœuds qui produisent les plus faibles distorsions. La recherche se poursuit ensuite à partir du niveau $l+1$ avec les M_t nœuds ainsi sélectionnés qui pointent vers $2M_t$ nœuds du prochain niveau (niveau $l+2$) et le processus est itéré jusqu'à ce que l'on atteigne le niveau terminal de l'arbre où parmi les $2M_t$ nœuds considérés seul le nœud produisant la plus faible distorsion est retenu. Le vecteur-code associé à ce nœud terminal est alors considéré comme le représentant du vecteur-entrée $\boldsymbol{x}$. L'indice de ce vecteur-code est alors transmis par l'encodeur. Notons que les indices transmis sont ceux des vecteurs-code des nœuds-feuille (les nœuds du niveau le plus élevé de l'arbre) et que les vecteurs-code des nœuds internes sont utilisés pour accélérer le processus de recherche. Dans ce cas, une recherche exhaustive consisterait à parcourir le dictionnaire composé des $N = 2^L$ vecteurs-code du niveau L le plus élevé.

La figure (4.2) donne un exemple de l'algorithme de recherche multi-trajets utilisant une structure en arbre binaire (équilibré) de cinq niveaux (du niveau 0 au niveau 4). Dans cet exemple $M_t = 2$ et le processus de recherche est entamé à partir du dictionnaire de

taille $\eta = 4$ (correspondant à l'ensemble des vecteurs-code du niveau $l = 2$ de l'arbre). Donc au niveau $l = 2$ de l'arbre binaire le vecteur-entrée $\boldsymbol{x}$ est comparé à tous les vecteurs-code (nœuds) de ce niveau et les 2 vecteurs-code ($M_t = 2$) les plus proches sont retenus. Supposons que les vecteurs-code d'indices $\boldsymbol{i}_2 = \{0, 2\}$ soient retenus, ainsi pour le prochain dictionnaire de taille 8 (correspondant aux vecteurs-code du niveau $l = 3$ de l'arbre), les vecteurs-code à considérer sont ceux d'indices 0, 1, 4 et 5. Si les vecteurs-code d'indices $\boldsymbol{i}_3 = \{1, 5\}$ produisent les distorsions les plus faibles, alors les vecteurs-code à considérer pour le dictionnaire de taille 16 (le niveau le plus haut de l'arbre) sont ceux d'indices 2, 3, 10 et 11. Ces vecteurs-code sont comparés au vecteur-entrée $\boldsymbol{x}$ et l'indice du vecteur-code le plus proche du vecteur entrée $\boldsymbol{x}$ (le vecteur-code d'indice $i_o = 11$ dans notre cas) est transmis. Notons que pour des structures arborescentes relativement importantes en nombre de niveaux, la réduction en nombre de calculs de distances peut être importante. Par exemple pour un arbre binaire de hauteur 7 et utilisant les paramètres suivant : $\eta = 16$ et $M_t = 4$ le nombre de calculs de distances est de 40 au lieu de 128 (si la recherche a été effectué sur le dictionnaire de 128 vecteurs-code).

Le nombre de calculs de distances pour la procédure de recherche utilisant un arbre binaire équilibré est donnée par :

$$ND = \eta + 2M_t \log_2(N/\eta), \tag{4.12}$$

où M_t est le nombre de candidats à retenir à chaque niveau de l'arbre. η est la taille du dictionnaire (correspondants à l'ensemble des vecteurs-code au niveau $\log_2 \eta$ de l'arbre) à partir duquel le processus de recherche est entamé. N est la taille du dictionnaire correspondant à l'ensemble des vecteurs-code au niveau le plus élevé de l'arbre. $\log_2(N/\eta)$ est le nombre de dictionnaires de tailles $2\eta, 4\eta, ..., N/2$ et N. La complexité de mémorisation est donnée par :

$$MC = p\left[\eta + \sum_{i=1}^{\log_2(N/\eta)-1} 2^i \eta + N\right], \tag{4.13}$$

où p est la dimension des vecteurs-code. Notons que les dictionnaires-arbre de tailles inférieures à η ne sont pas utilisés dans le processus de recherche et il n'est donc pas nécessaire de les sauvegarder en mémoire.

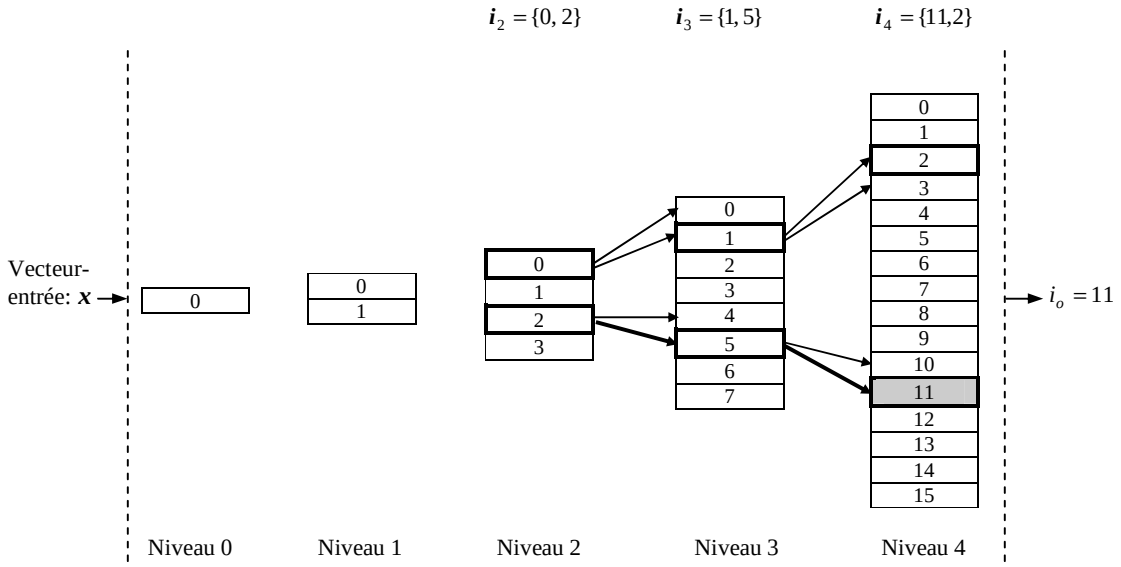


Figure 4.2: Illustration de la procédure de recherche multi-trajets (deux trajets: $M_t = 2$) utilisant un arbre binaire (équilibré) de profondeur 4. i_j est l'ensemble des indices (des vecteurs-code) sélectionnés au niveau *j* de l'arbre. La recherche est entamée à partir du dictionnaire de taille 4 (le niveau 2 de l'arbre). Les flèches pointent vers les vecteurs-code concernés par le processus de recherche. Les vecteurs-code sélectionnés sont marqués par des cadres en traits épais. Le meilleur chemin à travers l'arbre est indiqué par des flèches épaisses. Le résultat final est l'indice du vecteurs-code (marqué par un cadre plein en gris) extrait du niveau le plus élevé de l'arbre (nœud-feuille).

4.4 Structure arborescente à plusieurs étages

Dans ce cas la structure a plusieurs étages où chaque étage supporte une structure arborescente. Cela correspond à la structure d'un quantificateur MTVQ.

4.4.1 Recherche séquentielle multi-trajets assistée par une structure arborescente

La procédure de recherche désignée sous le nom de recherche séquentielle multi-trajets assistée par une structure arborescente MSTS (Multipath Sequential Tree-assisted Search) a été proposée récemment par Chu [66, 87]. La procédure utilise une structure MTVQ à *K* étages où une recherche séquentielle est utilisée lorsque un ensemble de

vecteurs-cible est passé d'un étage à l'autre. Utilisant une structure arborescente, une recherche à plusieurs chemins (multi-trajets) est effectuée pour chaque vecteur-cible.

Les paramètres qui contrôlent l'algorithme MSTS sont :

1- η_s : est la taille du dictionnaire-arbre à partir duquel la recherche est entamée à l'étage *s*

2- M_t : est le nombre maximum de candidats à retenir après avoir parcouru un dictionnaire-arbre.

3- M_a : est le nombre maximum de candidats à retenir après avoir parcourue un dictionnaire-étage.

Supposons une structure MTVQ à K étages où N_s $(s = 1 \text{ à } K)$ sont les tailles des dictionnaires-étage et η_s $(s = 1 \text{ à } K)$ sont les tailles des dictionnaires-arbre à partir duquel la recherche est entamée à chaque étage *s*. Une brève description de la procédure MSTS est donnée ci-dessous:

• Premier étage : Pour un vecteur-entrée $\boldsymbol{x}$, le dictionnaire-arbre de taille η_1 est parcouru une seule fois et les M_t meilleurs (produisant les plus faibles distances) vecteurs-code sont retenus. Utilisant la structure en arbre binaire, $2M_t$ candidats sont localisés au niveau du dictionnaire-arbre suivant (de taille $2\eta_1$) et les M_t meilleurs vecteurs-code sont retenus. Cette procédure est répétée jusqu'à ce que le dictionnaire-étage soit atteint; pour ce dictionnaire, $2M_t$ vecteurs-code sont localisés et un maximum de M_a, c'est-à-dire $\min(2M_t, M_a)$, vecteurs-code sont sélectionnés. Chacun de ces vecteurs-code est soustrait du vecteur-entrée $\boldsymbol{x}$ pour créer un ensemble de $\min(2M_t, M_a)$ vecteurs-cible qui est transmis à l'étage suivant.

• Deuxième étage: la procédure de recherche à plusieurs chemins utilisant la structure arborescente (où la recherche est entamée à partir du dictionnaire-arbre de taille η_2) est effectuée pour chaque vecteur-cible. Ainsi pour chaque vecteur cible le dictionnaire-arbre de taille η_2 est parcouru en totalité pour ne retenir que M_t vecteurs-code puis $2M_t$ vecteurs-code sont localisés au niveau du dictionnaire arbre suivant (de taille $2\eta_2$) et seulement M_t vecteurs-code sont retenus et ainsi de suite jusqu'à que le dictionnaire-

étage soit atteint. Donc au total $\min(2M_t, M_a)2M_t$ paires d'indices (indice vecteur-cible, indice vecteur-code du dictionnaire-étage) sont sélectionnées au niveau du dictionnaire-étage. Au final $\min(\min(2M_t, M_a)2M_t, M_a)$ paires d'indices sont retenues (un maximum de M_a paires sont retenus) et transmis à l'étage suivant. Cette procédure est répétée jusqu'à ce que le dernier étage soit atteint et que les meilleurs *K* indices (des vecteurs-code) issus des *K* dictionnaires-étage soient trouvés [66, 87].

Le nombre de calculs de distances pour la procédure MSTS est donné par :

$$ND = \sum_{s=1}^{K} n_{s-1}\left(\eta_s + 2M_t \log_2(N_s/\eta_s)\right), \tag{4.14}$$

avec

$$n_0 = 1,\ n_s = \min(2n_{s-1}M_t, M_a), \tag{4.15}$$

où $\log_2(N_s/\eta_s)$ est le nombre de dictionnaires de tailles $2\eta_s, 4\eta_s, ..., N_s/2$ et N_s.
Notons que dans l'équation (4.14) toutes les distances sont calculées (même si $M_a \geq 2n_{s-1}M_t$) et si la taille du dictionnaire-arbre à partir duquel la recherche est entamée (à chaque étage) est la même pour tous les étages (c'est-à-dire, $\eta_s = \eta$ pour $s = 1$ à K), alors dans ce cas les équations données dans la référence [66] correspondent aux équations (4.14) et (4.15). La complexité de mémorisation est donnée par :

$$MC = p\sum_{s=1}^{K}\left[\eta_s + \sum_{i=1}^{\log_2(N_s/\eta_s)-1} 2^i\eta_s + N_s\right]. \tag{4.16}$$

où *p* est la dimension des vecteurs. Les dictionnaires-arbre de tailles inférieures à η_s (à l'étage *s*) n'ont pas besoin d'être sauvegardés puisqu'ils ne sont pas utilisés dans le processus de recherche.

4.4.2 Recherche conjointe multi-trajets assistée par une structure arborescente

La procédure de recherche proposée est désignée sous le nom de : recherche conjointe multi-trajets assistée par une structure arborescente MJTS (Multipath Joint Tree-assisted Search). La procédure utilise une structure MTVQ à K étages. Une recherche conjointe est utilisée lorsqu'un ensemble de vecteurs-cible est passé d'un étage à l'autre et une recherche à plusieurs chemins (multi-trajets) est utilisée à l'intérieur d'un étage utilisant la structure arborescente (arbre binaire) associée à cet étage.
Les paramètres qui contrôlent l'algorithme MJTS sont :

1- η_s : est la taille du dictionnaire-arbre à partir duquel la recherche est entamée à l'étage s

2- M_q : est le nombre maximal de candidats à retenir après avoir parcouru un dictionnaire-arbre du premier étage.

3- M_r : est le nombre maximal de candidats à retenir après avoir parcouru un dictionnaire-arbre du deuxième étage et des étages suivants.

4- M_a : est le nombre maximal de candidats à retenir après avoir parcouru un dictionnaire-étage.

Soit une structure MTVQ à K étages où N_s $(s = 1 \text{ à } K)$ sont les tailles des dictionnaires-étage et η_s $(s = 1 \text{ à } K)$ sont les tailles des dictionnaires-arbre à partir desquels la recherche est entamée à chaque étage. Une brève description de la procédure MJTS est donnée ci-dessous :

• Premier étage : Pour un vecteur-entrée $\boldsymbol{x}$, le dictionnaire-arbre de taille η_1 est parcouru une seule fois et les M_q meilleurs (produisant les plus faibles distances) vecteurs-code sont retenus. Utilisant la structure en arbre binaire, $2M_q$ vecteurs-code candidats sont localisés au dictionnaire-arbre suivant (de taille $2\eta_1$) et les M_q meilleurs vecteurs-code sont retenus. Cette procédure est répétée jusqu'à ce que le dictionnaire-étage soit atteint; pour ce dictionnaire $2M_q$ vecteurs-code sont localisés et un maximum de M_a, c'est-à-dire $\min(2M_q, M_a)$, vecteurs-code sont sélectionnés. Chacun de ces vecteurs-code est

soustrait du vecteur-entrée $\boldsymbol{x}$ pour créer un ensemble de $\min(2M_q, M_a)$ vecteurs-cible, qui sont transmis à l'étage suivant.

• Deuxième étage : Une recherche conjointe est effectuée entre les $\min(2M_q, M_a)$ vecteurs-cible (calculés au premier étage) et le dictionnaire-arbre de taille η_2. Ainsi, le dictionnaire-arbre est recherché $\min(2M_q, M_a)$ fois. Par conséquent, $\min(2M_q, M_a)\ \eta_2$ distances sont calculées et les M_r paires d'indice (indice du vecteur-cible, indice du vecteur-code issu du dictionnaire-arbre) sont trouvées. En utilisant la structure arborescente (arbre binaire), $2M_r$ paires d'indices (indice du vecteur-cible, indice du vecteur-code du dictionnaire-arbre de taille $2\eta_2$) sont générées pour la recherche du dictionnaire-arbre de taille $2\eta_2$ et les M_r paires d'indices produisant les plus faibles distances sont retenues. Cette procédure est répétée jusqu'à ce que le dictionnaire-étage soit atteint; pour ce dictionnaire $2M_r$ paires d'indices sont utilisées et un maximum de M_a (c'est-à-dire $\min(2M_r, M_a)$) paires d'indices sont retenus et utilisées pour calculer le même nombre de vecteurs-cible qui sont transmis à l'étage suivant. Cette procédure est répétée jusqu'à ce que le dernier étage soit atteint et que les meilleurs *K* indices (des vecteurs-code) issus des *K* dictionnaires-étage soient trouvés. Les détails de l'algorithme sont donnés au tableau (4.1).

Le nombre de calculs de distances pour l'algorithme MJTS utilisant une structure MTVQ à *K* étages est donné par :

$$ND = \sum_{s=1}^{K} \left[n_{s-1}\eta_s + 2M_s \log_2(N_s/\eta_s) \right], \tag{4.17}$$

avec

$$M_1 = M_q, \text{ et } M_s = M_r \text{ pour } s > 1, \tag{4.18}$$

$$n_0 = 1, \text{ et } n_s = \min(2M_s, M_a) \text{ pour } s \geq 1, \tag{4.19}$$

où n_{s-1} est le nombre de vecteurs-cible utilisés (dans le processus de recherche) à l'étage *s*. N_s et η_s sont, respectivement, les tailles du dictionnaire-étage et du dictionnaire-arbre à partir duquel la recherche est entamée à l'étage s.

Tableau 4.1: Algorithme de recherche conjointe multi-trajets assistée par une structure arborescente MJTS (Multipath Joint Tree-assisted Search)

• Entrées :

Vecteur-entrée $\boldsymbol{x}$, une structure MTVQ à K étages et les parametres M_q, M_r et M_a

1-Initialisation :

$s = 1$ (premier étage),

$\boldsymbol{t}_0^{(0)} = \boldsymbol{x}$ et $n_0 = 1$ (le vecteur-entrée est le vecteur-cible initial),, $M \leftarrow M_q$.

2- Recherche du dictionnaire-arbre de taille η_s (à l'étage s)

2-1 Ajuster le nombre de candidats:

$M \leftarrow \min(n_{s-1}\eta_s, M)$, $m \leftarrow \log_2 \eta_s$

2-2 Calcul des distances (une recherche conjointe entre les n_{s-1} vecteurs-cible et le dictionnaire-arbre de taille η_s) :

$V_1 \leftarrow \{[i, j] \ , 0 \le i < n_{s-1}, 0 \le j < 2^m\}$

$D[i, j] \leftarrow d(\boldsymbol{t}_i^{(s-1)}, \boldsymbol{c}_{m,j}^{(s)})$ pout tout $[i, j] \in V_1$

2.3- Sélectionner les M meilleures paires d'indices (indice vecteur-cible, indice vecteur-code) :

$U_1 \leftarrow \{[k,l] | [k,l] \in V_1\}$, de cardinalité $|U_1| = M$ et

tel que $D[k,l] \le D[i, j]$, pour tout $[k,l] \in U_1$ et pour tout $[i, j] \in \{V_1 - U_1\}$.

3-Recherche à travers les dictionnaires-arbre suivants incluant le dictionnaire-étage :

3.1- Calcul des distances :

$V_2 \leftarrow \{[i,2j], \ [i,2j+1] \ \big| \ [i, j] \in U_1\}$ de cardinalité $|V_2| = 2M$

$D[i, j] \leftarrow d(\boldsymbol{t}_i^{(s-1)}, \boldsymbol{c}_{m+1,j}^{(s)})$, pour tout $[i, j] \in V_2$

3.2- Tester si le dictionnaire actuel est le dictionnaire-étage de taille N_s (à l'étage s) :

Si $(m+1) = \log_2(N_s)$ et $s < K$ alors $M \leftarrow \min(2M, M_a)$

Si $(m+1) = \log_2(N_s)$ et $s = K$ alors $M \leftarrow 1$ (une seule paire d'indices doit être trouvée)

3.3- Sélectionner les M meilleures paires d'indices

$U_2 = \{[k,l] \ \big| \ [k,l] \in V_2\}$ de cardinalité $|U_2| = M$

tel que $D[k,l] \le D[i, j]$, pour tout $[k,l] \in U_2$ et pour tout $[i, j] \in \{V_2 - U_2\}$

3.4 Test de décision :

$m \leftarrow m + 1$,

Si $m < \log_2(N_s)$ alors $U_1 \leftarrow U_2$ et répéter les étapes 3.1 à 3.3, autrement aller à l'étape suivante.

4- Calcul des vecteurs-cible (pour le prochain étage): $r \leftarrow \log_2 N_s$

$$\begin{cases} \boldsymbol{t}_k^{(s)} = \boldsymbol{t}_i^{(s-1)} - \boldsymbol{c}_{r,j}^{(s)} & s < K \\ \boldsymbol{i}_k^{(s)} = j & s = 1 \\ \boldsymbol{i}_k^{(s)} = [\boldsymbol{i}_i^{(s-1)}, j] & 1 < s \le K \end{cases} \qquad \begin{array}{l} [i, j] \in U_2 \ \text{ pour chaque } k, \\ \text{avec } k = 0 \text{ à } |U_2| - 1 \end{array}$$

$n_s = |U_2|$ est le nombre de vecteurs-cible et $[\boldsymbol{i}_i^{(s-1)}, j]$ est l'opération de concaténation.

5-Test de décision :

$s \leftarrow s + 1$,

si $s \le K$ alors $M \leftarrow M_r$ et répéter les étapes 2 à 4, autrement arrêter le traitement.

• Sortie :

Un ensemble de K indices de vecteurs-code issus des K dictionnaires-étage :

$\boldsymbol{i}_0^{(K)} = [i_1, i_2, ..., i_K]$.

La complexité de mémorisation pour l'algorithme MJTS est la même que celle du MSTS si la même valeur du paramètre η_s est utilisée pour les deux méthodes. On a donc :

$$MC = p\sum_{s=1}^{K}\left[\eta_s + \sum_{i=1}^{\log_2(N_s/\eta_s)-1} 2^i\eta_s + N_s\right], \tag{4.20}$$

où p est la dimension des vecteurs-code.

4.5 Recherche conjointe multi-trajets assistée par des niveaux

La deuxième procédure de recherche que nous proposons est désignée sous le nom de recherche conjointe multi-trajets assistée par des niveaux MJLS (Multipath Joint Level-assisted Search). La procédure utilise une structure à *K* étages avec un certain nombre de niveaux (dictionnaires) à chaque étage. Une recherche conjointe est utilisée lorsque un ensemble de vecteurs-cible est passé d'un étage à l'autre et une recherche à plusieurs chemins (multi-trajets) est utilisée à l'intérieure d'un étage.

4.5.1 Élaboration des dictionnaires

Dans le cas ou seuls deux dictionnaires sont utilisés à chaque étage *s*, ces deux dictionnaires correspondent au dictionnaire-étage de taille N_s et un deuxième dictionnaire (appelé dictionnaire-fusionné) de taille η_s avec $\eta_s < N_s$. Le dictionnaire-fusionné peut être conçu de deux manières :

• La première méthode consiste à utiliser une structure MTVQ (conçue par la méthode décrite dans le Chapitre 3 ou celle proposée par Chu [66] ou une autre méthode) et à ne retenir, à chaque étage, que deux dictionnaires : le dictionnaire-étage de taille N_s et le dictionnaire-arbre de taille η_s [89].

• La deuxième méthode est la plus optimale et consiste à utiliser l'algorithme de groupage, utilisée conjointement avec la procédure de permutation d'indices (Chapitre 3), pour élaborer un dictionnaire-fusionné de taille η_s à partir du dictionnaire-étage de taille N_s à l'étage *s*. Ceci correspond à la recherche de la meilleure séquence de dimension

$n = N_s / \eta_s$ et de taille η_s: $S^{(s,n)} = \{I_i^{(s,n)}, i = 0 \text{ to } \eta_s - 1\}$ à l'étage *s*. Dans ce cas, le dictionnaire-fusionné de taille η_s (de résolution $m = \log_2 \eta_s$) est le dictionnaire $\boldsymbol{C}_m^{(s)} = \{\boldsymbol{c}_{m,i}^{(s)}, i = 0,..., \eta_s - 1\}$ où le vecteur-code $\boldsymbol{c}_{m,i}^{(s)}$ est associé à l'ensemble $I_i^{(s,n)}$ qui pointe vers *n* vecteurs-code du dictionnaire-étage; le vecteur-code $\boldsymbol{c}_{m,i}^{(s)}$ est calculé en utilisant l'équation (3.23). Notons que le symbole *s* est introduit dans la notation pour indiquer que le dictionnaire et la séquence sont associés à l'étage *s*.

Dans le cas où plusieurs niveaux sont utilisés à chaque étage, le *l* ième dictionnaire-fusionné sera élaboré à partir du $l-1$ ième dictionnaire-fusionné de la même manière que le premier dictionnaire-fusionné est élaboré à partir du dictionnaire-étage.

4.5.2 Procédure de Recherche

Les paramètres qui contrôlent l'algorithme MJLS sont :

1- η_s : est la taille du dictionnaire-fusionné à l'étage *s.*

2- M_q : est le nombre maximum de candidats à retenir après avoir parcouru le dictionnaire-fusionné du premier étage.

3- M_r : est le nombre maximum de candidats à retenir après avoir parcouru le dictionnaire-fusionné du deuxième étage et des étages suivants.

4- M_a : est le nombre maximum de candidats à retenir après avoir parcouru un dictionnaire-étage.

Soit une structure à *K* étages utilisant un seul dictionnaire fusionné à chaque étage (en plus du dictionnaire-étage), une brève description de la procédure MJLS (sur la base d'un traitement étage par étage) est donnée ci-dessous :

• Premier étage : Le dictionnaire-étage et le dictionnaire-fusionné sont de tailles N_1 et η_1 respectivement. La séquence (de taille η_1 et de dimension $n = N_1 / \eta_1$) liée au dictionnaire-fusionné est la séquence $S^{(1,n)} = \{I_i^{(1,n)}, i = 0,..., \eta_1 - 1\}$ où l'ensemble $I_i^{(1,n)}$ (associé au vecteur-code d'indice *i* du dictionnaire-fusionné) pointe vers *n* vecteurs-code du dictionnaire-étage. Pour un vecteur-entrée donné $\boldsymbol{x}$, le dictionnaire-fusionné est

recherché une seule fois et les M_q meilleurs (ayant les plus faibles distances par comparaison au vecteur-entrée) vecteurs-code sont retenus. En utilisant la séquence $S^{(1,n)}$, nM_q vecteurs-code du dictionnaire-étage sont localisés et comparés au vecteur-entrée. Alors $\min(nM_q, M_a)$, c'est-à-dire un maximum de M_a vecteurs-code produisant les plus faibles distances sont retenues. Chacun de ces vecteurs-code est soustrait du vecteur-entrée $\boldsymbol{x}$ pour créer un ensemble de $\min(nM_q, M_a)$ vecteurs-cible, qui sont transmis à l'étage suivant.

• Deuxième étage : Une recherche conjointe est effectuée entre les $\min((N_1/\eta_1)M_q, M_a)$ vecteurs-cible (calculés au premier étage) et le dictionnaire-fusionné de taille η_2. Ainsi, le dictionnaire-fusionné est recherché $\min((N_1/\eta_1)M_q, M_a)$ fois. Par conséquent, $\min((N_1/\eta_1)M_q, M_a)\,\eta_2$ distances sont calculées et les M_r paires d'indice (indice du vecteur-cible, indice du vecteur-code issu du dictionnaire-fusionné) sont trouvées. En utilisant la séquence $S^{(2,n)}$ (de dimension $n = N_2/\eta_2$ où N_2 est la taille du dictionnaire-étage), nM_r paires d'indices (indice du vecteur-cible, indice du vecteur-code issu du dictionnaire-étage) sont générés et utilisés pour la recherche au sein du dictionnaire-étage. Les $\min(nM_r, M_a)$ paires d'indices produisant les plus faibles distances sont retenues. Alors $\min(nM_r, M_a)$ vecteurs-cible sont calculés et transmis au prochain étage. Cette procédure est répétée jusqu'à ce que le dernier étage soit atteint et que les meilleurs *K* indices (extraits des *K* dictionnaires-étage) soient trouvés. Les détails de l'algorithme sont donnés au tableau (4.2).

On donne à la figure (4.3) une illustration de la procédure de recherche MJLS utilisant une structure à deux étages correspondants à deux dictionnaires-étage de même taille $N_1 = N_2 = 16$ et les deux dictionnaires-fusionné de même taille $\eta_1 = \eta_2 = 4$. Les paramètres suivant sont utilisés :

- deux indices du dictionnaire-fusionné sont sélectionnés au premier étage ($M_q = 2$).
- trois pairs d'indices (indice vecteur-cible, indice vecteur-code issu du dictionnaire-fusionné) sont sélectionnées au deuxième étage ($M_r = 3$).

- trois pairs d'indices (indice vecteur-cible, indice vecteur-code du dictionnaire-étage) sont sélectionnées ($M_a = 3$).

On montre surtout comment les indices des vecteurs-code se développent (voir le tableau 4. 2 pour les détails).

4.5.3 Complexité de calcul

Le nombre de calculs de distances pour l'algorithme MJLS (utilisant une structure à *K* étages où chaque étage utilise deux dictionnaires) est donné par :

$$ND = \sum_{s=1}^{K} \left(n_{s-1}\eta_s + (N_s / \eta_s) M_s \right), \tag{4.21}$$

avec

$$M_1 = M_q \text{ et } M_s = M_r \text{ pour } s > 1 \tag{4.22}$$

$$n_0 = 1 \text{ et } n_s = \min\left((N_s / \eta_s) M_s, M_a\right) \text{ pour } s \geq 1, \tag{4.23}$$

où n_{s-1} est le nombre de vecteurs-cible utilisés (dans le processus de recherche) à l'étage *s*. N_s et η_s sont respectivement les tailles du dictionnaire-étage et du dictionnaire-fusionné à l'étage s. La complexité de mémorisation pour l'algorithme MJLS (utilisant deux dictionnaires à chaque étage) est donnée par :

$$MC = p \sum_{s=1}^{K} \left(\eta_s + N_s \right), \tag{4.24}$$

où p est la dimension des vecteurs-code. Chaque composant d'un vecteur-code exige une unité de mémoire (un réel). L'espace mémoire requis pour la sauvegarde des séquences d'ensembles d'indices (pour lier les dictionnaires-fusionné aux dictionnaires-étage) a été négligée dans l'équation (4.24). Notons que la complexité de mémorisation de l'algorithme MJLS est inférieure à celle du MJTS (en comparant les équations (4.20) et (4.24)) et peut être proche de celle de l'algorithme MSS si la valeur de η_s est relativement faible (équations (4.11) et (4.24)).

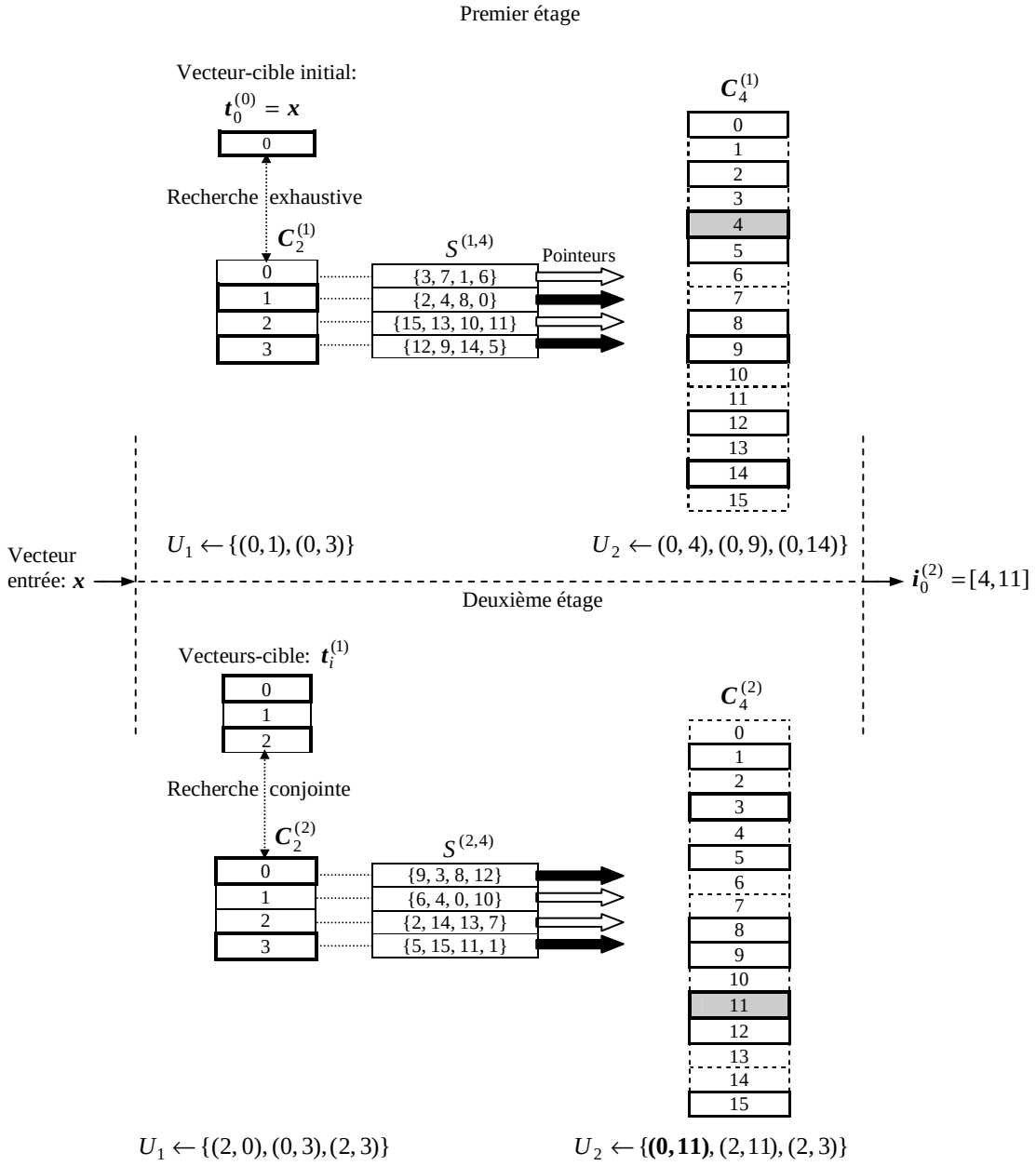


Figure 4. 3: Illustration de la procédure de recherche MJLS utilisant une structure à deux étages correspondants à deux dictionnaires-étage de même taille $N_1 = N_2 = 16$ et deux dictionnaires-fusionné de même taille $\eta_1 = \eta_2 = 4$. Les paramètres de recherche utilisés sont: $M_q = 2$, $M_r = 3$ et $M_a = 3$. Les vecteurs-code impliqués dans le processus de recherche sont marqués par des cadres en traits pleins. Les vecteurs-code et les vecteurs-cible sélectionnés sont marqués par des cadres en traits épais. $S^{(1,4)}$ et $S^{(2,4)}$ sont les séquences d'ensembles d'indices reliant le dictionnaire-fusionné au dictionnaire-étage au premier et au deuxième étage respectivement. U_1 et U_2 contiennent les paires d'indices sélectionnées au niveau du dictionnaire-fusionné et du dictionnaire-étage respectivement. Le résultat final est l'ensemble $i_0^{(2)}$ de deux indices correspondants aux deux vecteurs-code (marqués par des cadres pleins en gris) issus du premier et du deuxième dictionnaires-étage.

Tableau 4.2: Algorithme de recherche conjointe multi-trajets assistée par des niveaux MJLS (Multipath Joint Level-assisted Search)

• Entrées :

Vecteur-entrée $\boldsymbol{x}$, une structure à K étages où chaque étage correspond à un dictionnaire-étage, un dictionnaire-fusionné et une séquence d'ensembles d'indices et les parametres: M_q, M_r, M_a.

1- Initialisation :

$s = 1$ (premier étage),

$\boldsymbol{t}_0^{(0)} = \boldsymbol{x}$ et $n_0 = 1$ (le vecteur-entrée est le vecteur-cible initial), $M \leftarrow M_q$.

2- Recherche à travers le dictionnaire-fusionné de taille η_s (à l'étage s)

2.1- Ajuster le nombre de candidats:

$M \leftarrow \min(n_{s-1}\eta_s, M)$.

2.2- Calcul des distances (une recherche conjointe entre les n_{s-1} vecteurs-cible et le dictionnaire-fusionné de taille η_s) :

$V_1 \leftarrow \{[i, j] \ , 0 \le i < n_{s-1}, 0 \le j < \eta_s\}$

$D[i, j] \leftarrow d(\boldsymbol{t}_i^{(s-1)}, \boldsymbol{c}_{m,j}^{(s)})$ pout tout $[i, j] \in V_1 \quad (m = \log_2 \eta_s)$

2.3- Sélectionner les M meilleures paires d'indices (indice vecteur-cible, indice vecteur-code):

$U_1 \leftarrow \{[k,l] \,|\, [k,l] \in V_1\}$, de cardinalité $|U_1| = M$ et

tel que $D[k,l] \le D[i, j]$, pour tout $[k,l] \in U_1$ et pour tout $[i, j] \in \{V_1 - U_1\}$.

3- Recherche à travers le dictionnaire-étage de taille N_s ($r \leftarrow \log_2 N_s$).

3.1- Calcul des distances :

$V_2 \leftarrow \{[i, j_1], \ [i, j_2], ..., [i, j_n] \,|\, [i, j] \in U_1 \text{ et } I_j^{(s,n)} = \{j_1, j_{2,...,} j_n\}\}$,

de cardinalité $|V_2| = nM$ avec $n = N_s / \eta_s$.

$D[i, j] \leftarrow d(\boldsymbol{t}_i^{(s-1)}, \boldsymbol{c}_{r,j}^{(s)})$, pour tout $[i, j] \in V_2$.

3.2- Tester si le dernier dictionnaire-étage est atteint :

si $s = K$ alors $M \leftarrow 1$ (une seule paire d'indices doit être trouvée)

autrement $M \leftarrow \min((N_s / \eta_s)M, M_a)$ (ce n'est pas le dernier dictionnaire-étage)

3.3- Sélectionner les M meilleures paires d'indices

$U_2 \leftarrow \{[k,l] \,|\, [k,l] \in V_2\}$ de cardinalité $|U_2| = M$ et

tel que $D[k,l] \le D[i, j]$, pour tout $[k,l] \in U_2$ et pour tout $[i, j] \in \{V_2 - U_2\}$.

4- Calcul des vecteurs-cible (pour le prochain étage) :

$$\begin{cases} \boldsymbol{t}_k^{(s)} = \boldsymbol{t}_i^{(s-1)} - \boldsymbol{c}_{r,j}^{(s)} & s < K \\ \boldsymbol{i}_k^{(s)} = j & s = 1 \\ \boldsymbol{i}_k^{(s)} = [\boldsymbol{i}_i^{(s-1)}, j] & 1 < s \le K \end{cases} \qquad \begin{array}{l} [i, j] \in U_2 \ \text{ pour chaque } k, \\ \text{avec } k = 0 \text{ à } |U_2| - 1 \end{array}$$

$n_s = |U_2|$ est le nombre de vecteurs-cible et $[\boldsymbol{i}_i^{(s-1)}, j]$ est l'opération de concaténation.

5- Test de terminaison :

$s \leftarrow s + 1$,

si $s \le K$ alors $M \leftarrow M_r$ et répéter les étapes 2 à 4, autrement arrêter le traitement.

• Sortie :

Un ensemble de K indices de vecteurs-code issus des K dictionnaires-étage :

$\boldsymbol{i}_0^{(K)} = [i_1, i_2, ..., i_K]$.

4.6 Calcul des distances

Dans le processus de recherche utilisant une structure arborescente à plusieurs étages (correspondant a celle d'un quantificateur MTVQ), la distance (à l'étage s) entre le *k* ième vecteur-cible (parmi les *M* vecteurs-cible calculés à l'étage $s-1$) et le vecteur-code d'indice *i* issu d'un dictionnaire de résolution *m* (à l'étage *s*) est donnée par :

$$d(\boldsymbol{t}_k^{(s-1)}, \boldsymbol{c}_{m,i}^{(s)}) = (\boldsymbol{t}_k^{(s-1)} - \boldsymbol{c}_{m,i}^{(s)})^T \mathbf{W}(\boldsymbol{t}_k^{(s-1)} - \boldsymbol{c}_{m,i}^{(s)}) \ , \tag{4.25}$$

qui peut s'écrire sous la forme suivante :

$$d(\boldsymbol{t}_k^{(s-1)}, \boldsymbol{c}_{m,i}^{(s)}) = \boldsymbol{t}_k^{(s-1)^T} \mathbf{W} \boldsymbol{t}_k^{(s-1)} - 2\boldsymbol{t}_k^{(s-1)^T} \mathbf{W} \boldsymbol{c}_{m,i}^{(s)} + \boldsymbol{c}_{m,i}^{(s)^T} \mathbf{W} \boldsymbol{c}_{m,i}^{(s)}. \tag{4.26}$$

Tous les vecteurs de l'équation (4.26) sont des vecteurs colonnes de dimension p. $\mathbf{W}$ est une matrice de pondération diagonale dérivée du vecteur-entrée [12]. Le première terme correspond à une distorsion minimale calculée à l'étage précédente (étage $s-1$). En effet on a :

$$\boldsymbol{t}_k^{(s-1)^T} \mathbf{W} \boldsymbol{t}_k^{(s-1)} = (\boldsymbol{t}_i^{(s-2)} - \boldsymbol{c}_{r,j}^{(s-1)})^T \mathbf{W}(\boldsymbol{t}_i^{(s-2)} - \boldsymbol{c}_{r,j}^{(s-1)}) = d(\boldsymbol{t}_i^{(s-2)}, \boldsymbol{c}_{r,j}^{(s-1)}) \, , \tag{4.27}$$

où $[i, j]$ est une des M paires d'indices (indice du vecteur-cible, indice du vecteur-code) sélectionnée à l'étage $s-1$ (voir l'étape 4 correspondant au calcul des vecteurs-cible pour les algorithmes des tableaux 4.1 et 4.2) et *r* est la résolution du dictionnaire-étage. Donc le premier terme de l'équation (4.26) est connu, le deuxième terme exige $2p$ opérations arithmétiques et le dernier terme exige $3p-1$ opérations arithmétiques. Notons que le terme $\mathbf{W}\boldsymbol{c}_{m,i}^{(s)}$ se trouve dans le second et le dernier terme de l'équation (4.26) et est calculé une seule fois (dans le dernier terme). Enfin 2 additions sont exigées pour calculer la distance globale. Ainsi $5p+1$ opérations arithmétiques sont nécessaires pour évaluer l'équation (4.26). Pour un vecteur-code $\boldsymbol{c}_{m,i}^{(s)}$ donné, nous devons calculer M distances correspondantes aux M $(k = 0,...,M-1)$ vecteurs-cible; une réduction importante en nombre d'opérations arithmétiques est obtenue en calculant le dernier terme de l'équation

(4.26) une seul fois puisque ce dernier ne dépend pas des vecteurs-cible. Dans ce cas le nombre d'opérations arithmétiques pour le calcul des *M* distances correspondant aux *M* vecteurs-cible (pour un vecteur code donné) est de $(3p-1)+M(2p+2)$. Donc une réduction assez importante en nombre d'opérations arithmétiques est obtenue du fait de l'utilisation de l'équation (4.26) au lieu de l'équation (4.25) qui doit être calculée en totalité pour chaque vecteur-code et chaque vecteur-cible; le nombre d'opérations arithmétiques pour les *M* vecteurs-cible pour un vecteur-code donné est de $M(4p-1)$. Notons enfin qu'en utilisant l'équation (4.26) toutes les distances doivent être calculées même si le nombre de candidats au niveau d'un dictionnaire-étage est inférieur à la valeur du paramètre M_a (le nombre de candidats à retenir) dans les équations (4.14), (4.17) et (4.21).

L'algorithme de recherche par distance partielle PDS (Partial distance search algorithm) rapporté par Bei et Gray [90] est une méthode très simple et efficace pour améliorer l'efficacité de l'encodage pour la quantification vectorielle en réduisant la complexité de calcul tout en produisant la même sortie que la méthode de recherche exhaustive conventionnelle. Il est important de noter que cette méthode n'exige aucune augmentation de la complexité de mémorisation. Pour le calcul de la distance partielle, l'équation (4.25) peut s'écrire sous la forme :

$$d(\boldsymbol{t}_k^{(s-1)}, \boldsymbol{c}_{m,i}^{(s)}) = \sum_{j=0}^{p-1} w_j (\boldsymbol{t}_k^{(s-1)}(j) - \boldsymbol{c}_{m,i}^{(s)}(j))^2 , \tag{4.28}$$

où $\boldsymbol{c}_{m,i}^{(s)} = [\boldsymbol{c}_{m,i}^{(s)}(0), \boldsymbol{c}_{m,i}^{(s)}(1), ..., \boldsymbol{c}_{m,i}^{(s)}(p-1)]^T$, $\boldsymbol{t}_k^{(s-1)} = [\boldsymbol{t}_k^{(s-1)}(0), \boldsymbol{t}_k^{(s-1)}(1), ..., \boldsymbol{t}_k^{(s-1)}(p-1)]^T$ et les coefficients $w_j (j = 0 \text{ à } p-1)$ sont les valeurs de la matrice diagonale $\mathbf{W}$ qui sont des valeurs positives. Lorsque la distorsion accumulée pour les *j* premiers composantes est plus grande que la plus petite distorsion *dmin* trouvée jusqu'ici, le vecteur-code est rejeté avant de compléter le calcul de la distorsion (4.28) (calcul de la distance totale). Cette sortie précoce réduira le nombre d'opérations arithmétiques. Notons que pour chaque valeur de *j* de l'équation ci-dessus, 4 + 1 (1 pour la comparaison à *dmin*) opérations arithmétiques sont exigées. Lorsque les *M* plus faibles distances sont maintenues, *dmin*

correspond à la plus grande distance (parmi les *M* distances). Notons que, pour soutenir les procédures MSS, MSTS, MJTS et MJLS, des instructions de comparaison sont nécessaires pour maintenir les *M* vecteurs-code ayant les plus faibles distances et pour éliminer les autres candidats ayant de plus grandes distances.

Si une prédiction de la distorsion minimale initiale est incluse, l'algorithme PDE peut être plus efficace et dans ce cas l'algorithme est désigné par l'algorithme PDE prédictif dans [91] et a été appliqué pour le cas d'une quantification vectorielle (pour le traitement d'images) utilisant un dictionnaire non-structuré et la distance Euclidienne. D'autres travaux se sont concentrés sur l'utilisation de la propriété d'inégalité triangulaire des espaces métriques [92-95]. L'algorithme ENNS (equal-average nearest neighbour search algorithm) [93-95] utilise une inégalité entre la distance à calculer (entre un vecteur entrée et un vecteur-code) et la différence des valeurs moyennes (du vecteur-entrée et du vecteur-code). Cette inégalité est utilisée pour rejeter les vecteurs-code qui ne peuvent être les plus proches voisins et ainsi éviter le calcul de la distance (entre le vecteur-entrée et le vecteur-code testé) qui est plus couteuse en complexité de calcul. Dans [94] l'algorithme proposé utilise la moyenne et la variance du vecteur-entrée et du vecteur-code pour développer une nouvelle inégalité plus efficace que celle proposée en [93]. Cependant la plupart de ces algorithmes exigent un coût de mémorisation additionnel (pour le stockage des valeurs moyennes et/ou des covariances des vecteurs-code) et utilisent la distance Euclidienne pour la mesure de distorsion et donc ne sont pas appropriés pour la distance Euclidienne pondérée. Bai dans [95] a proposé un algorithme de recherche rapide basé sur l'algorithme ENNS mais utilisant la distance Euclidienne pondérée comme mesure de distorsion. Les vecteurs-code sont réordonnés dans l'ordre croissant de leurs moyennes pondérées qui doivent être stockées en mémoire.

4.7 Conclusion

Des procédures de recherche rapides pour réduire la complexité de calcul exigée pour localiser des vecteurs-code pendant le processus d'encodage utilisant une quantification vectorielle multi-étages sont proposées. L'algorithme MJTS utilise une structure MTVQ dont le coût de mémorisation est approximativement le double de celle d'une structure MSVQ. L'algorithme MJLS utilise un certain nombre de niveaux (de

dictionnaires) à chaque étage et le coût de mémorisation peut être proche de celle d'une structure MSVQ lorsque seul deux dictionnaires sont utilisés à chaque étage. Dans ce cas, à chaque étage, le dictionnaire-fusionné est élaboré à partir du dictionnaire étage utilisant l'algorithme de groupage et le procédé de permutation d'indices (Chapitre 3). Dans ce cas chaque vecteur-code, issu du dictionnaire-fusionné, pointe vers un ensemble de vecteurs-code issus du dictionnaire-étage où tous les ensembles de vecteurs-code sont de même cardinalité. Notons que l'algorithme du tableau 4.2 peut être adapté lorsque L dictionnaires sont utilisés à chaque étage. Dans ce cas le l ième dictionnaire-fusionné peut être élaboré à partir du $l-1$ ième dictionnaire-fusionné de la même manière que le premier dictionnaire-fusionné est élaboré à partir du dictionnaire-étage.

Pour une structure MSVQ, l'algorithme MSS (connu aussi sous le nom de recherche *M-L*) est le plus connu. Il a été montré qu'en utilisant un nombre restreint de candidats par étage, la performance de l'algorithme MSS est très proche de celle de l'algorithme de recherche exhaustive avec une complexité de calcul beaucoup plus faible. Ainsi la performance des autres algorithmes sera comparée à celle de l'algorithme MSS. L'algorithme MSTS utilise la structure arborescente, associée à chaque étage, pour accélérer le processus de recherche et a été comparé dans [66] à l'algorithme MSS en termes de nombre de calculs de distances Euclidiennes. On montrera dans le Chapitre résultat que la performance de l'algorithme MJLS est très proche de celle de l'algorithme MSS et est meilleure que celle de l'algorithme MSTS en termes de complexité (de calcul et de mémorisation) et de distorsion spectrale pour la quantification vectorielle des paramètres LSF. Notons que la complexité de calcul sera mesurée en nombre de calcul de distances et en nombre d'opérations en virgule flottante flops (floating point operations). Chaque addition/soustraction, multiplication ou comparaison est considérée comme un flop.

Chapitre 5

Implémentation d’un logiciel d’outils pour le codage de parole

5.1 Introduction

Le langage MATLAB n’est pas très adapté aux traitements de bases de données assez importantes et aux structures de données complexes. En effet, les algorithmes de conception de quantificateurs, présentés au chapitre 3, nécessitent des structures de données relativement complexes et un coût de calcul relativement élevé puisque ces algorithmes sont appliqués sur des bases de données de parole assez importantes. De même, les algorithmes de recherche rapide, présentés au chapitre 4, doivent être appliqués sur une base de données relativement importante et différentes valeurs pour les paramètres d’entrée (de ces algorithmes) doivent être testées afin d’établir une étude comparative entre ces différents algorithmes de recherche. L’implémentation de ces algorithmes en MATLAB nécessiterait un coût de calcul encore plus élevé comparé aux langages de programmation comme C/C++ et C#. En plus, dans le cas de ce projet de recherche on est amené à modifier certains codeurs normalisés. Ces codeurs sont mis à la disposition des chercheurs sous forme de fichiers source écrits en langage C/C++, il est

donc plus efficace d'utiliser le langage C/C++ ou un langage proche/interopérable avec ce dernier. Cependant, notons que le langage MATLAB [96] fournit beaucoup d'outils de traitement de signal et est très efficace pour tester des algorithmes ne nécessitant pas un coût de calcul important.

La programmation est une tâche importante dans notre projet. En effet, l'implémentation des différents algorithmes (tels les algorithmes d'apprentissage, les algorithmes de recherche rapides, la structure des différents quantificateurs etc..) doit être très flexible. En effet on est amené à faire beaucoup de tests avec différentes valeurs pour les paramètres d'entrée de ces algorithmes ; notamment pour l'étude comparative entre les différents algorithmes de recherche rapides (Chapitre 4). De plus ces algorithmes s'appliquent, généralement, sur une base de données assez importante. La sélection des fichiers de la base de données doit se faire d'une façon très flexible. La modification de codeurs de parole normalisés (tel les codeurs FS-MELP et G.729) nécessite aussi une bonne maitrise du fonctionnement de ces codeurs ainsi que le langage dans lequel ils sont implémentés (C/C++). Ceci nous a conduit à la conception et l'implantation d'un logiciel qui nous fournirait les outils et les mécanismes qui pourraient nous faciliter la tâche de test et d'évaluation des différents algorithmes. Le choix du langage orienté objet C# (C Sharp) [97] du Visual Studio 2005 est justifié par sa fiabilité au niveau de la gestion de la mémoire et la détection des erreurs de programmation comparé au langage C/C++. En effet, le dépassement de la taille des tableaux est détecté et les conversions implicites de type de données (source d'erreurs difficile à détecter) ne sont pas permises. Notons aussi que la syntaxe du C# est très proche de celle de C/C++ avec l'avantage de ne plus exiger la déclaration des entêtes des fonctions dans des fichiers isolés (les fichiers .h en C/C++). Il est très important de noter que les langages de programmation définis dans Visual Studios 2005 (C++, C#, Visual Basic,....) sont interopérables. Ceci permet de combiner du code écrit en C# et C/C++.

5.2 Structure du logiciel

Le logiciel réalisé est composé de plusieurs bibliothèques DLL (Dynamic Link Library) et de programmes exécutables, notamment pour les codeurs normalisés modifiés ainsi que les outils de prétraitement (sous-échantillonnage, etc…) et d'évaluation de la

qualité de la parole synthétisée. Un programme principal composé d'une fenêtre principale et d'un menu principal se charge de la gestion des appels : création des objets (à partir des classes) et appel des fonctions membres et appel des programmes exécutable. On résume les principales bibliothèques et programmes exécutables :

GUILib.dll : Implémente tous ce qui concerne les outils graphiques de saisie et d'affichage : telles les fenêtres de saisie pour les valeurs des paramètres d'entrée des différents algorithmes, la fenêtre de chargement de la base de données, etc... Cette bibliothèque fournit aussi les outils pour l'affichage des signaux de paroles (stockés dans des fichiers) dans des fenêtres graphiques. Elle permet aussi le défilement, l'agrandissement et l'écoute de ces signaux.

MathLib.dll : Facilite l'écriture des opérations entre vecteurs et matrices sous forme scientifique. Deux classes principales sont définies : *Mat* (Matrice) et *Vec* (Vecteur). Certains opérateurs arithmétiques et logiques (+ , - , *, /, ==, != etc…) sont redéfinis dans ces classes pour qu'ils puissent être appliqués sur des vecteurs et des matrices. Des fonctions mathématiques (puissance, racine carrée, sinus, cosinus,….) sont définies de sorte qu'elles puissent être appliquées sur des matrices et des vecteurs. Par exemple, on pourra écrire du code semblable à celui en MATLAB :
A = (v1 * v2) * (M1 * M2) – 4 * Sin(M3) où v1 et v2 sont des vecteurs. M1, M2 et M3 sont des matrices. Sin : est le sinus de tous les éléments de la matrice. Le résultat A est une matrice. Dans ce cas, l'operateur * correspond au :

- produit scalaire quand il est utilisé entre deux vecteurs.
- produit matriciel quand il est utilisé entre deux matrices.
- produit entre un scalaire et une matrice (ou un vecteur) quand il est utilisé entre un scalaire et une matrice (ou un vecteur).

LPCLib.dll : Implémente tous ce qui concerne la prédiction linéaire : calcul des coefficients de prédiction linéaire, calcul des coefficients LSF, calcul de la densité spectrale de puissance ou de l'enveloppe spectrale (à partir des coefficients LPC) etc…

VQuantLib.dll : Implémente les classes correspondantes aux différents quantificateurs vectoriels (pour la conception des quantificateurs ainsi que les opérations de l'encodage et du décodage) présentés au chapitre 3. On retrouve aussi les classes correspondantes aux algorithmes rapides du chapitre 4.

ITUTG191.exe : Est un ensemble d'outils logiciels pour la normalisation du codage des signaux vocaux et audiofréquences de la recommandation UIT-T G.191 qui est désigné par "version 2005 de la bibliothèque d'outils logiciels" ou en abrégé STL2005 (Software Tools Library release 2005). Dans de nombreux cas, les résultats expérimentaux obtenus avec différents outils logiciels ne peuvent être directement comparés. La nécessité d'un ensemble commun d'outils a été, donc, reconnue dans les activités de l'UIT-T. En conséquence, une bibliothèque de programmes logiciels fiables a été établie. Le but des outils de la STL2005 est d'aider la communauté scientifique à élaborer plus efficacement des normes nouvelles dans le domaine des télécommunications [98].

ITUTP862.exe : Contient les fonctions correspondant à une méthode objective d'évaluation de la qualité vocale, de la recommandation ITU-T P.862, appelée "Evaluation de la qualité vocale perçue" PESQ (perceptual evaluation of speech quality). Cette méthode s'applique non seulement aux codeurs vocaux mais aussi aux mesures de bout en bout [99].

ITUTG729.exe : Implémente le codeur normalisé G.729 modifié pour l'introduction de la caractéristique d'échelonnabilité.

FSMELP.exe : Implémente le codeur normalisé FS-MELP modifié pour l'introduction de la caractéristique d'échelonnabilité.

5.3 Description du logiciel

On donne à la figure 5.1 la fenêtre principale du logiciel. Elle est composée d'un menu principal, une bande d'outils contenant des raccourcis vers les rubriques du menu et une bande d'état pour l'affichage d'informations. On décrit quelques rubriques du menu principal.

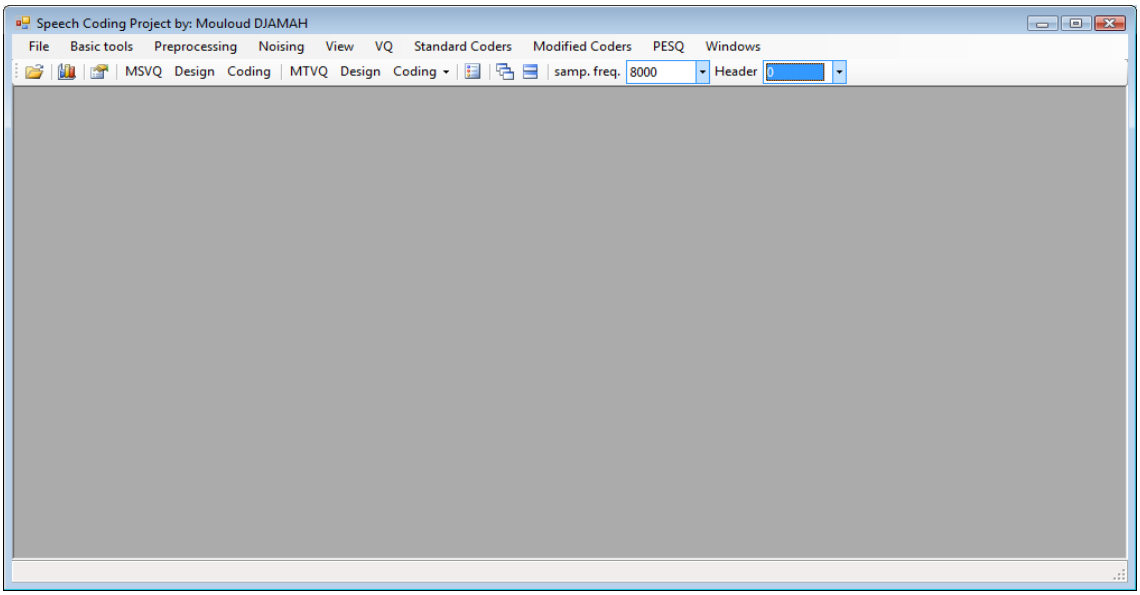


Figure 5.1: Fenêtre principale du logiciel

5.3.1 Chargement des bases de données

[Fichier/Base de données] : Cette rubrique permet de charger, à partir du disque dur, les fichiers des bases de données test et d'apprentissage. La figure 5.2 montre la fenêtre graphique utilisée pour le chargement des différentes bases de données. Les noms complets (chemin d'accès, le nom et l'extension) des fichiers sont affichés. En plus de la possibilité de charger des fichiers, une commande permet de charger tous les fichiers contenus dans un dossier (ainsi que les sous-dossiers de ce dossier) spécifique du disque dur. Plusieurs dossiers peuvent être chargés séquentiellement. Cependant seuls les fichiers sélectionnés (les noms de fichiers en couleur foncée) sont utilisés lors de lancement d'un traitement utilisant une base de données test ou d'apprentissage. Ceci étant très pratique lors de la phase test des algorithmes ou seuls quelques fichiers sont utilisés. Ainsi toute la base de données est chargée une seule fois et selon le traitement souhaité, quelques fichiers ou tous les fichiers de la base de données sont traités.

Un fichier de parole est, généralement, stocké sous forme d'un entête suivie des données. L'entête occupe un certain nombre d'octets du début du fichier. Par exemple, pour les fichiers de la base de données TIMIT [100] l'entête occupe les 1024 premiers octets. L'entête d'un fichier est un ensemble de champs d'information permettant d'identifier le format du fichier (format 'wav' et autres) et le format des données. Par

exemple pour les fichiers 'wav' de la base de données TMIT, le format des données concerne : le nombre de bits utilisé pour représenter chaque échantillon du signal de parole (16 bits), la fréquence d'échantillonnage, etc… L'entête d'un fichier ne fait pas partie des données et doit donc être écarté avant tout traitement. La taille de l'entête est spécifiée dans un champ de saisie de la bande d'outils (figure 5.1). Ainsi pour tous les traitements effectués par la suite, l'entête de chaque fichier sera écarté et seules les données seront traitées. Les fichiers de sortie résultants de ces traitements seront des fichiers contenant que des données (des fichiers sans entêtes).

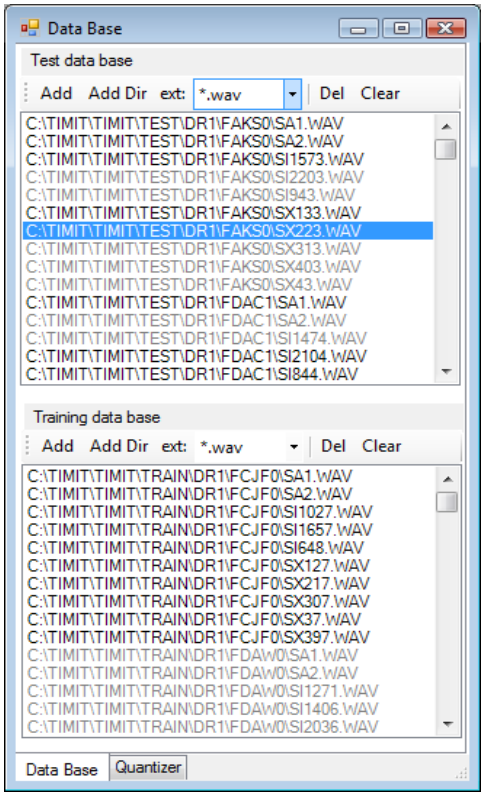


Figure 5.2: Fenêtre pour le chargement des bases de données test et apprentissage.

5.3.2 Prétraitements

[Prétraitement/Sous-échantillonnage (2 :1)] : Le signal de parole des bases de données test et d'apprentissage (figure 5.2) est sous-échantillonné par un facteur de deux. Chaque signal, de fréquence d'échantillonnage F_e, est filtré par un filtre passe bas (à phase linéaire et à réponse impulsionnelle finie FIR) pour limiter sa largeur de bande à $F_e/4$,

une décimation par un facteur de 2 est alors effectuée : seul un échantillon sur deux est retenu. Cette rubrique utilise un outil de la bibliothèque STL2005 [98].

[Prétraitement/Pondération IRS-modifié à 16 kHz] : Le filtre IRS-modifié est appliqué sur la parole, échantillonnée à 16 kHz, des bases de données test et apprentissage. Les caractéristiques de fréquence à l'émission du système IRS modifié (modified Intermediate Reference System), tel qu'il est défini dans la recommandation P.830 [101], simulent les caractéristiques de fréquence d'émission d'un combiné téléphonique. Ce filtre provient de la bibliothèque STL2005.

[Prétraitement/ Voltmètre de parole P56] : Le niveau de parole actif (des bases de données test et apprentissage) est ajusté à un niveau donné (en dB relative au point de surcharge) utilisant l'algorithme voltmètre de parole SVP56, de la recommandation P.56 [102] d'ITU-T, implémenté dans la bibliothèque STL2005.

[Prétraitement/Filtre Delta-SM à 16 kHz] : Ce filtre est appliqué sur la parole, échantillonnée à 16 kHz, des bases de données test et apprentissage (figure 5.2). Le filtre de pondération Δsm est utilisé pour convertir un signal acoustique enregistré depuis un champ lointain à l'aide d'un microphone omnidirectionnel à l'équivalent de ce signal comme s'il était dans le fond du téléphone d'un utilisateur. Le filtre Δsm est utilisé pour pondérer les bruits d'environnement (voiture, etc…) avant de les additionner avec le signal de parole propre afin de simuler de la parole corrompue par un bruit de fond. Ce filtre provient de la bibliothèque STL2005.

[Bruitage/Bruit de voiture] : Ajoute un bruit de voiture à la parole des bases de données test et apprentissage. D'autre type de bruit (bruit de rue, brouhaha, etc..) peuvent être ajoutés. Les signaux bruit sont stockés dans des fichiers.

5.3.3 Traitements

[Codeurs modifiés/G.729 (CS-ACELP)] : La parole de la base de données test (figure 5.2) est traitée (encodée et décodée) par le codeur G.729 modifié. Le codeur G.729 est modifié de telle manière à ce qu'il intègre la caractéristique d'échelonnabilité (Chap. 6).

[Codeurs modifiés/FS-MELP] : La parole de la base de données test (figure 5.2) est traitée par le codeur FS-MELP modifié. Le codeur FS-MELP est modifié de telle manière à ce qu'il intègre la caractéristique d'échelonnabilité (Chapitre 6).

[Quantification vectorielle/Paramètres]: Cette rubrique permet d'afficher une fenêtre de saisie (figure 5.3) pour les paramètres d'entrée des différents algorithmes : quantification vectorielle, algorithmes de recherche rapides, valeur SNR pour l'ajout de bruit, etc...

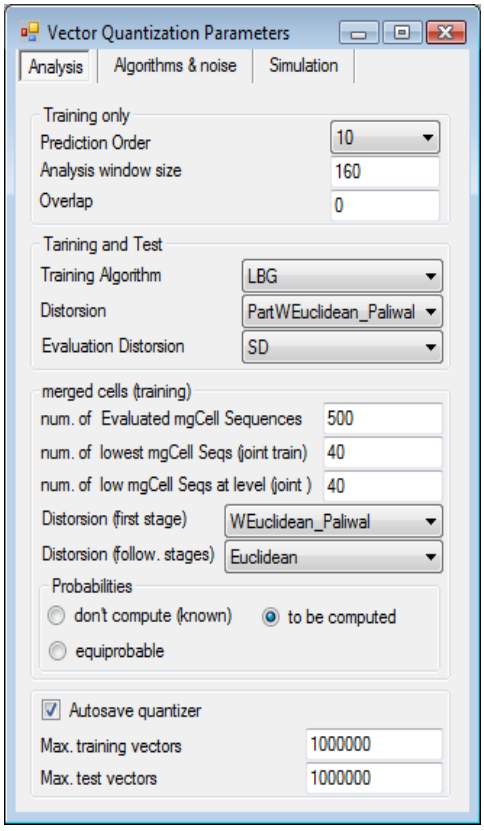


Figure 5.3: Fenêtre de saisie pour différents paramètres.

[Quantification vectorielle/MTVQ/Apprentissage] : Utilise la base de données apprentissage (figure 5.2) pour l'élaboration des dictionnaires (codebook design), utilisant la technique de fusion de cellules (Chapitre 3), du quantificateur MTVQ. Une fois conçu le quantificateur (les dictionnaires associés au quantificateur) est stocké dans un fichier.

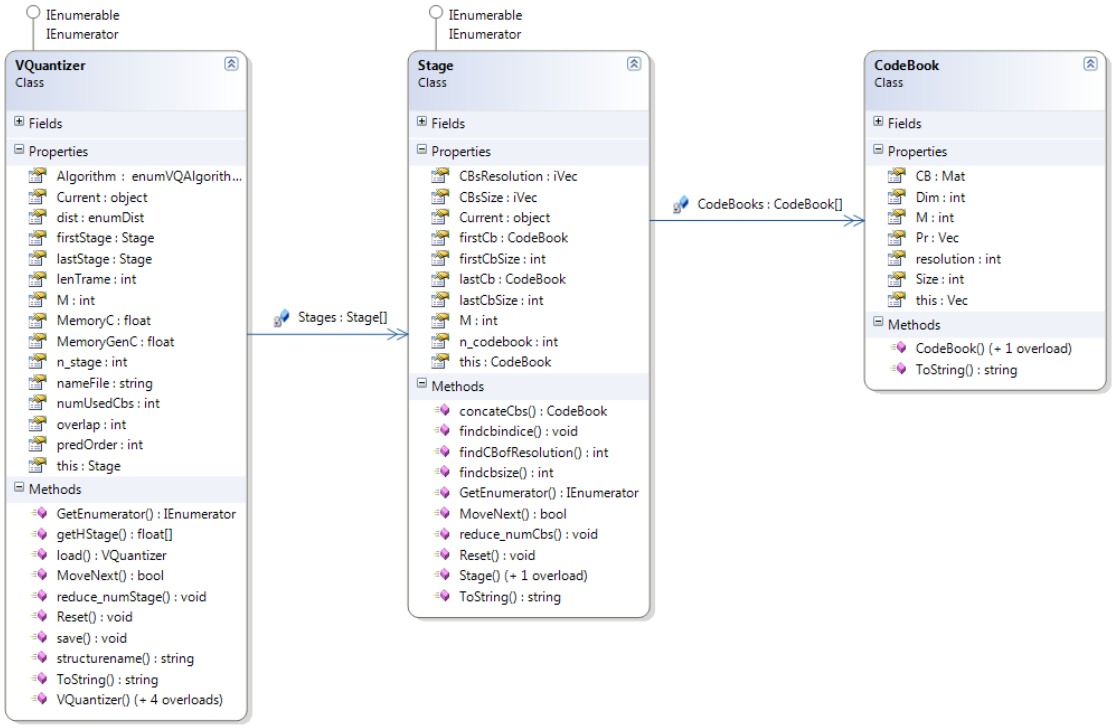


Figure 5.4: Diagramme des classes pour la structure du quantificateur : La classe *VQuantizer* contient un tableau de classes de type *Stage* et la classe *Stage* contient un tableau de classes de type *Codebook*.

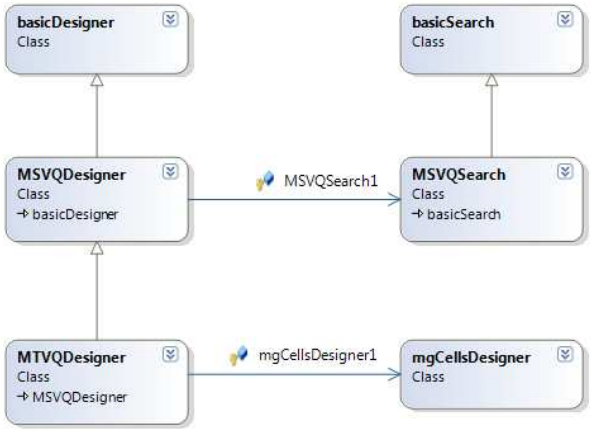


Figure 5.5: Diagramme des classes pour la conception d'un quantificateur MTVQ (Codebook Design) utilisant la technique de fusion de cellules.

Le diagramme des classes correspondant à la structure du quantificateur est donné à la figure 5.4. Ce diagramme se base sur la structure d'un quantificateur MTVQ composée de plusieurs étages où chaque étage est composé de plusieurs dictionnaires (voir l'exemple de la figure 3.2). Cette structure est très flexible ; en effet, selon le nombre d'étages et le nombre de dictionnaires (et la dimension de ces dictionnaires) par étage, une grande variété de quantificateurs peuvent être présentés tels les quantificateurs MSVQ, SVQ, MTVQ, TSVQ ainsi que des quantificateurs hybrides tel celui du quantificateur LSF de la norme G.729 (Chapitre 2). Le diagramme des classes pour la conception d'un quantificateur MTVQ est donné à la figure 5.5.

[Quantification vectorielle/MTVQ/Test] : Encode et décode une base de données de vecteurs LSF utilisant une quantification MTVQ. Les coefficients LSF sont calculés à partir de la parole de la base de données test (figure 5.2). Le diagramme des classes du codeur (encodage et décodage) MTVQ est donné à la figure 5.6. Ce diagramme donne aussi les classes qui définissent les différentes procédures de recherches rapides : MSS, MSTS, MJTS, MJLS (Chapitre 4). Après chaque traitement, les résultats obtenus, tels la distorsion spectrale moyenne, complexité de calcul (Chapitre 6), sont sauvegardés dans un fichier sous forme d'une base de données de type access [103]. Les valeurs des paramètres des différents procédures de recherche sont, aussi, sauvegardés dans la base de données. La base de données pourra, par la suite, être affichée dans une fenêtre, facilitant ainsi l'étude comparative entre les différents algorithmes de recherche. On donne à la figure 5.7 un exemple d'affichage de résultats.

[PESQ/PESQ bande étroite] : La norme ITU-T P.862 PESQ [99] est utilisée pour mesurer la qualité de la parole synthétisée comparée au signal de parole original. Le modèle perceptif d'évaluation PESQ est utilisé pour calculer une distance entre le signal de parole original et le signal de parole dégradé. Les fichiers originaux (contenant la parole originale) et les fichiers dégradés (parole synthétique) sont chargés dans une fenêtre semblable à celle de la figure 5.2. Cependant lorsqu'un traitement par un codeur est exécuté sur une base de données test, l'évaluation PESQ peut être exécutée automatiquement à la fin du traitement, une fois la base de données des fichiers dégradés est rendue disponible.

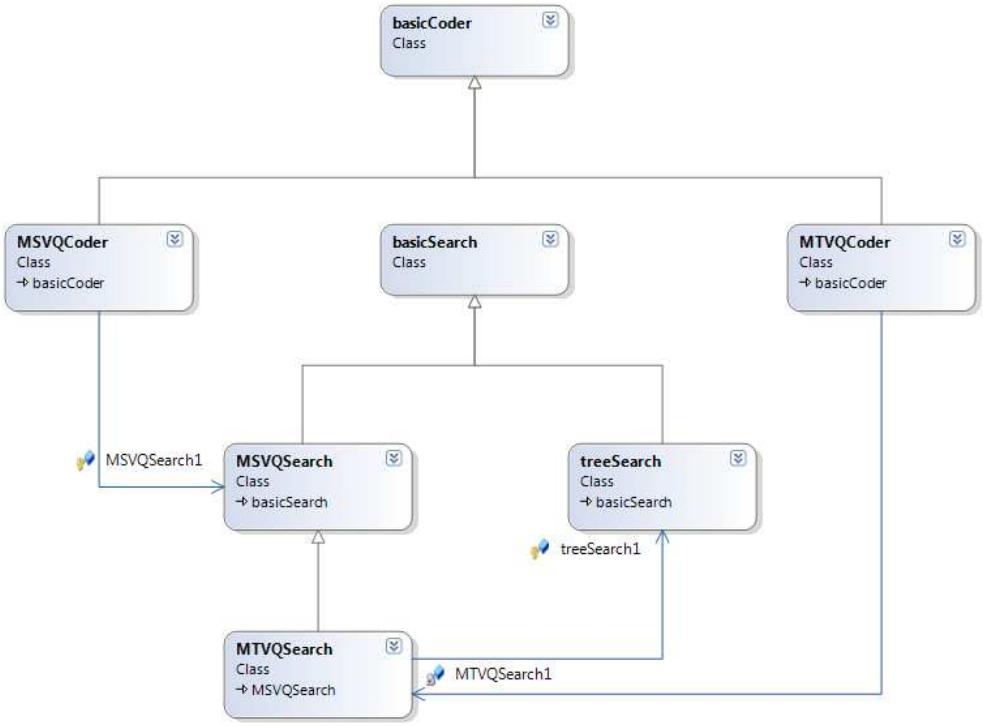
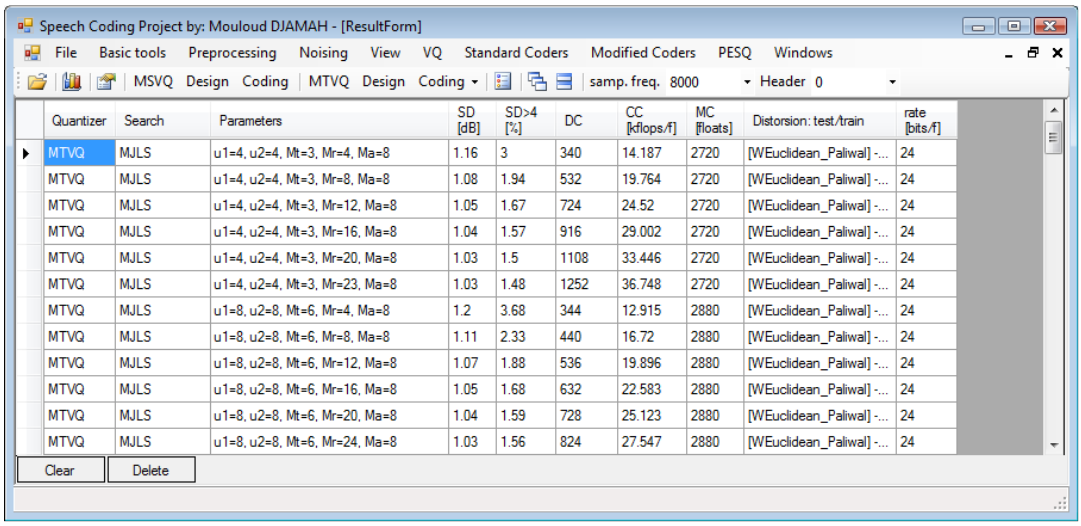


Figure 5.6: Diagramme des classes pour le codeur (encodeur et décodeur) MSVQ et MTVQ ainsi que pour les différentes procédures de recherches rapides.



Quantizer	Search	Parameters	SD [dB]	SD>4 [%]	DC	CC [kflops/f]	MC [floats]	Distorsion: test./train	rate [bits/f]
MTVQ	MJLS	u1=4, u2=4, Mt=3, Mr=4, Ma=8	1.16	3	340	14.187	2720	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=4, u2=4, Mt=3, Mr=8, Ma=8	1.08	1.94	532	19.764	2720	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=4, u2=4, Mt=3, Mr=12, Ma=8	1.05	1.67	724	24.52	2720	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=4, u2=4, Mt=3, Mr=16, Ma=8	1.04	1.57	916	29.002	2720	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=4, u2=4, Mt=3, Mr=20, Ma=8	1.03	1.5	1108	33.446	2720	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=4, u2=4, Mt=3, Mr=23, Ma=8	1.03	1.48	1252	36.748	2720	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=8, u2=8, Mt=6, Mr=4, Ma=8	1.2	3.68	344	12.915	2880	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=8, u2=8, Mt=6, Mr=8, Ma=8	1.11	2.33	440	16.72	2880	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=8, u2=8, Mt=6, Mr=12, Ma=8	1.07	1.88	536	19.896	2880	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=8, u2=8, Mt=6, Mr=16, Ma=8	1.05	1.68	632	22.583	2880	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=8, u2=8, Mt=6, Mr=20, Ma=8	1.04	1.59	728	25.123	2880	[WEuclidean_Paliwal] -...	24
MTVQ	MJLS	u1=8, u2=8, Mt=6, Mr=24, Ma=8	1.03	1.56	824	27.547	2880	[WEuclidean_Paliwal] -...	24

Figure 5.7: Exemple d'affichage de résultats obtenus après lancement d'une opération de quantification MTVQ sur une base de données de vecteurs LSF. Les résultats sont stockés sous forme d'une base de données de type access.

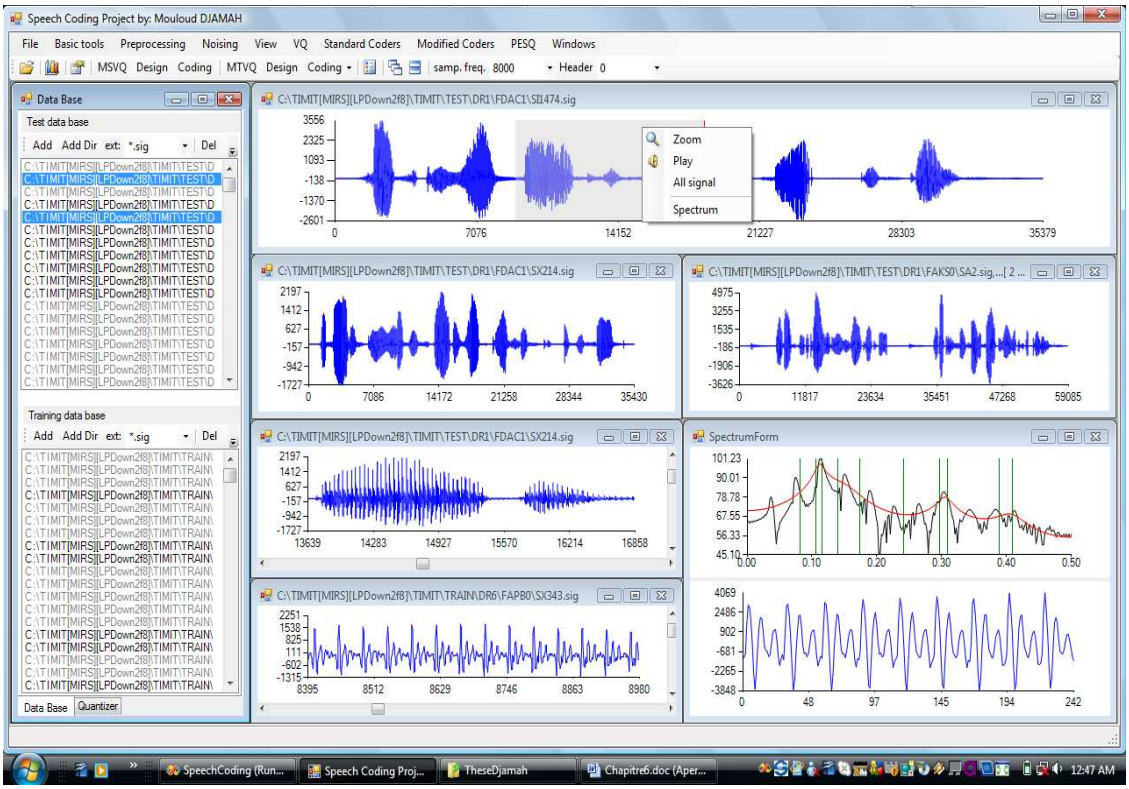


Figure 5.8: Exemple d'affichage de l'interface du logiciel.

[Afficher/Signaux] : Permet l'affichage des signaux de paroles dans des fenêtres graphiques. Des commandes permettent l'agrandissement, le défilement et l'écoute de ces signaux. Cette rubrique permet aussi de sélectionner une partie d'un signal de parole et d'effectuer des traitements (calcul du spectre LPC, coefficients LSF, ...) sur cette partie du signal. La figure 5.8 donne un exemple d'affichage de plusieurs signaux.

5.3.4 Traitements séquentiels

Les différents traitements (ou prétraitements) sont exécutés séquentiellement. Généralement, chaque traitement utilise une base de données d'entrée. Chaque fichier de la base de données est traité et un fichier résultat est fourni. À la fin du traitement, une autre base de données est donc créée comme base de données de sortie (figure 5.9).

Figure 5.9: Traitement d'une base de données.

La base de données initiale, composée de plusieurs fichiers de parole, peut avoir une structure de dossiers assez profonde (cas de la base de données TIMIT). Ainsi, les fichiers de parole sont stockés dans une structure de dossiers selon le locuteur, la région géographique et le type de la base de données (test ou apprentissage). Les fichiers de paroles correspondants aux mêmes phrases peuvent avoir les mêmes noms (nom et extension du fichier). Il n'est donc pas possible de mettre tous les fichiers de la base de données dans un seul dossier, à condition de modifier les noms des fichiers, ce qui n'est pas simple à faire (le nombre de fichiers étant assez élevé) et en plus on perdrait la structure initiale de la base de données qui porte une information (sur la région, le locuteur, etc…).

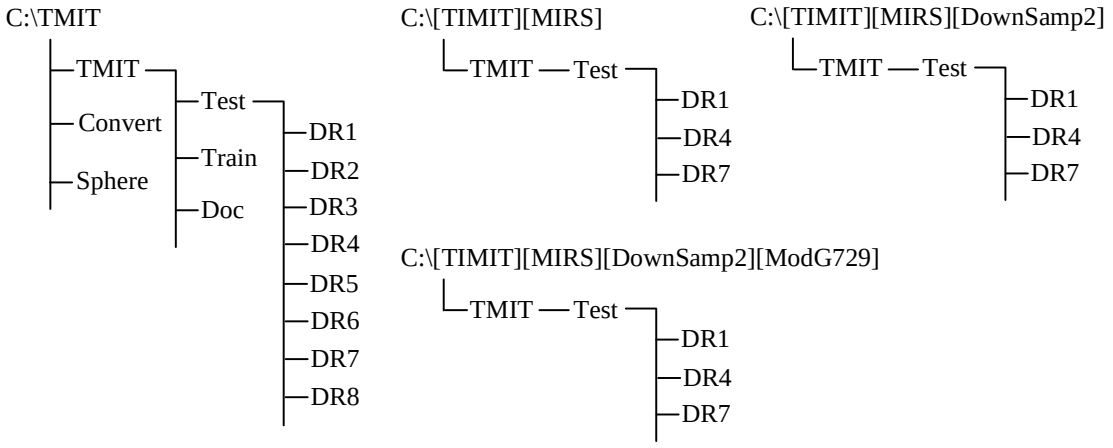


Figure 5.10: Structure de dossiers des différentes bases de données : Base de données initiale (à gauche) suivie des bases données correspondantes à trois traitements effectuées séquentiellement. Le nom du premier dossier reflète les traitements effectués (dans cet exemple seuls les dossiers DR1, DR4 et DR7 sont traités).

Pour notre cas la structure de la base de données d'entrée n'est pas modifiée. La base de données de sortie a la même structure de dossiers interne que celle de la base de données d'entrée ; par contre le nom du premier dossier est modifié de façon à refléter le traitement effectué. Les mêmes noms de fichiers sont maintenus alors que l'extension des fichiers (dans la base de données de sortie) est modifiée en '.sig' pour refléter le fait qu'il s'agit de fichiers de données (fichiers sans entêtes). La figure 5.10 donne la structure de dossiers de la base de données initiale ainsi que les structures de dossiers des trois bases de données correspondantes aux trois traitements effectués séquentiellement : filtrage IRS-modifié, sous-échantillonnage (par un facteur de 2) et codage par le codeur modifié

G.729 (voir Chapitre suivant). Ainsi, en gardant la même structure de dossier et les mêmes noms de fichier (des bases de données d'entrée et de sortie) en ne modifiant que le nom du premier dossier en fonction du traitement effectuée, il devient simple d'automatiser une suite de traitements démarrant d'une base de donnes initiale. Il est simple, aussi, d'associer un fichier d'entrée à un fichier de sortie obtenu après avoir effectué un seul ou une suite de traitements.

5.4 Conclusion

Les différents algorithmes présentés dans ce projet (algorithmes d'apprentissage, de recherche rapides, etc…) ont été implémentés sous forme d'un logiciel interactif utilisant le langage C# (version 2) de Visual Studio 2005. Certains outils de la bibliothèque STL2005 ont été, aussi, intégrés dans ce logiciel (comme procédures de prétraitements) permettant ainsi de les appliquer sur une base de données composée de plusieurs fichiers de parole. La norme ITU-T P.862 PESQ utilisée pour mesurer la qualité de la parole synthétisée comparée au signal de parole original a été intégré dans le logiciel. Ainsi une base de données composée de fichiers de parole initiale peut être comparée à une base de données composée des fichiers de parole dégradée. Le logiciel implémenté peut ainsi être utilisé par d'autres chercheurs comme un outil normalisé (utilisant des outils normalisés de l'UIT-T) efficace pour le prétraitement et l'évaluation de bases de données importantes faisant ainsi économiser un temps important. Tous les résultats présentés dans le Chapitre suivant ont été obtenus en utilisant ce logiciel.

Notons qu'a partir de la version 3 du langage C# [104], du Visual Studio 2010, on peut écrire du code plus clair et plus compréhensible (en utilisant le LINQ: Language INtegrated Query) donc plus simple à maintenir. Notons aussi l'introduction récente du langage F# [105] du Microsoft. Ce langage est très adapté à la programmation scientifique et permet de faire des constructions très complexes qui ne sont pas encore possibles avec les autres langages (C#, C++, etc…). Notons que de la documentation gratuite et récente peut être trouvée au lien suivant : www.free-ebooks-download.org

Chapitre 6

Évaluation des Performances

6.1 Introduction

Nous modifions deux normes de codage de la parole. Le codeur standard fédéral MELP est modifié en remplaçant le quantificateur MSVQ original (pour la quantification des coefficients LSF) et le quantificateur non structuré original (pour la quantification des amplitudes de Fourier) par un quantificateur MTVQ et un quantificateur TSVQ respectivement. Le codeur ITU-T G.729 (CS-ACELP) est modifié en remplaçant le quantificateur original prédictif basé sur une structure MSVQ (pour la quantification des coefficients LSF) par un quantificateur prédictif basé sur une structure MTVQ. Lorsque un quantificateur TSVQ est conçu, la méthode proposée dans le Chapitre 3 est appliquée; les valeurs $N_L = 500$ et $M_L = 40$ sont utilisées pour l'algorithme de groupage (tableau 3.3) avec les probabilités des vecteurs-code (du niveau le plus élevé de l'arbre) estimées en utilisant un ensemble suffisamment grand de vecteurs d'apprentissage. L'algorithme de permutation d'indices (tableau 3.4) est appliqué seulement sur les M_L séquences à la dimension désirée. Pour une structure MTVQ, la conception se fait d'une façon séquentielle : étage par étage (du premier au denier étage).

La procédure de recherche rapide proposée (Chapitre 4) pour la quantification des paramètres LSF est examinée pour deux attributs importants : la qualité des paramètres encodés et la complexité du codage. Les considérations de complexité comprennent la complexité de calcul et de mémorisation (la complexité de mémorisation étant définie par l'espace mémoire requis pour le stockage des dictionnaires du quantificateur). En outre, diverses comparaisons de performance (à différents débits binaire) avec les méthodes MSS et MSTS sont fournies.

6.2 Évaluation de la qualité de la parole synthétisée

Nous utilisons la norme ITU-T P.862 PESQ (Perceptual evaluation of speech quality : évaluation de la qualité vocale perçue) pour mesurer la qualité de la parole synthétisée (produite par un codeur de parole) comparée au signal de parole initial. Le modèle perceptif d'évaluation PESQ est utilisé pour calculer une distance entre le signal de parole initial et le signal de parole dégradé (note d'évaluation PESQ). Cette opération doit passer par une fonction monotone pour obtenir une prévision de la note MOS subjective pour un essai subjectif donné. La note d'évaluation PESQ est appliquée à une échelle de type MOS (mean opinion score : note moyenne d'opinion), sous la forme d'un numéro unique compris entre −0.5 et 4.5, bien que dans la plupart des cas le signal de sortie soit compris entre 1.0 et 4.5, ce qui correspond à l'amplitude normale de valeurs MOS observées dans une expérience de qualité d'écoute ACR (absolute category rating : évaluation par catégories absolues) [99].

Le matériel de parole utilisé est extrait de la base de données TIMIT-test [100] et se compose de 160 fichiers de parole prononcée par 16 orateurs (8 hommes et 8 femmes), chacun prononçant 10 phrases. Comme dans la plupart des expériences retenues pour l'étalonnage et la validation de l'évaluation PESQ [99], des groupes de deux phrases (séparées par un silence), d'une durée totale de huit secondes, sont combinées de telle manière qu'il n'y ait pas de relation évidente de sens entre les 2 phrases et où la parole est active pendant 50 à 80% du temps. Les 80 exemples de parole sont prétraités avec le filtre système de référence intermédiaire IRS (Intermediate Reference System) modifié. Ce filtre provient de la bibliothèque d'outils logiciels ITU-T G.191 STL2005 (Software Tool Library) et opère à 16 kHz [98].

Les caractéristiques de fréquence à l'émission du système de référence intermédiaire IRS modifié (tel qu'il est défini dans la recommandation P.830 [101]) simulent les caractéristiques de fréquence d'émission d'un combiné téléphonique. Le niveau de parole actif est ajusté à -26 dBov (dB relative au point de surcharge) utilisant l'algorithme voltmètre de parole SVP56 (recommandation P.56 d'ITU-T [102]) implémenté dans l'ITU-T STL2005. Le signal de parole est filtré avec un filtre passe-bas pour une décimation à une fréquence d'échantillonnage de 8 kHz. Le filtre utilisé est le filtre de changement de fréquence d'échantillonnage haute qualité 2:1 de l'ITU-T STL2005. Les 80 exemples de parole, totalisant une durée de 10.67 minutes, constituent le signal initial pour l'évaluation PESQ.

La procédure pour le mélange de la parole avec les bruits environnementaux est montrée à la figure 6.1 [106]. Le signal de parole original est ajusté à -26 dBov puis mélangé aux bruits environnementaux : bruits de voiture et de brouhaha avec un SNR (rapport signal-sur-bruit) de 10, 15, 20 et 25 dB. Le bruit est filtré par le système IRS modifié (caractéristique d'émission). Le niveau du bruit est ajusté utilisant l'algorithme voltmètre de parole SVP56 conforme à la recommandation P.56. Le bruit est ajusté aux niveaux -36, -41, -46 et -51 dBov correspondant à un SNR de 10, 15, 20 et 25 dB respectivement. Le signal de parole original est également filtré et le niveau de la parole actif ajusté à -26 dBov. Finalement, le bruit est numériquement mélangé à la parole. Si l'amplitude résultante de signal excède le point de surcharge (sur 16 bits), elle est limitée à la valeur maximale.

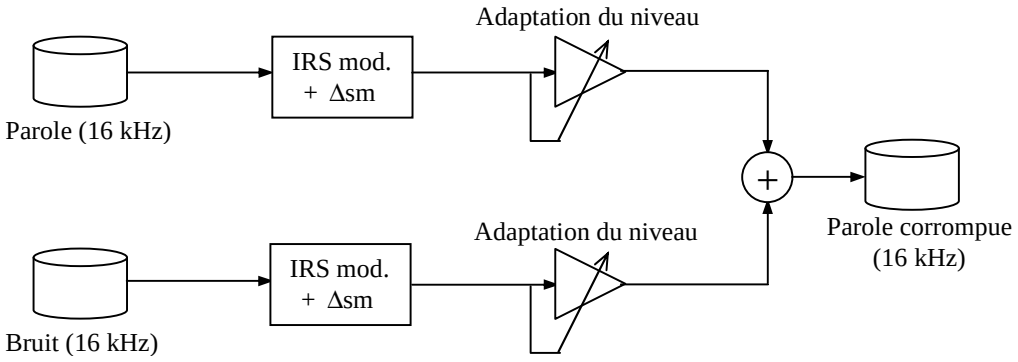


Figure 6.1: Prétraitement pour la corruption d'un signal de parole par un bruit de fond.

Le filtre de pondération Δsm est utilisé pour convertir un signal acoustique enregistré depuis un champ lointain à l'aide d'un microphone omnidirectionnel à l'équivalent de ce signal comme s'il était dans le fond du téléphone d'un utilisateur. Le filtre Δsm est utilisé pour pondérer les bruits d'environnement (brouhaha, voiture, etc…) avant de les additionner avec le signal de parole propre afin de simuler de la parole corrompue par un bruit de fond [98]. Les filtres MIRS (IRS modifié) et Δsm sont implémentés comme des filtres FIR (à réponse impulsionnelle finie) et ont été conçu pour de la parole échantillonnée à 16 kHz.

6.3 Modification du codeur fédéral standard MELP

Le codeur standard fédéral MELP [18, 50, 51] (Chapitre 2) utilise une longueur de trame de 180 échantillons (correspondant à 22.5 ms pour une fréquence d'échantillonnage de 8 kHz), où 54 bits sont utilisés pour encoder chaque trame produisant un débit binaire de 2400 bits/s. L'encodeur utilise un quantificateur MSVQ de quatre étages pour la quantification des paramètres LSF correspondants à quatre dictionnaires-étage de tailles 128, 64, 64 et 64 (de résolutions 7, 6, 6 et 6 bits), conduisant à un débit de 25 bits par trame (Chapitre 2). Les probabilités des vecteurs-code (des dictionnaires-étage) sont estimées et un TSVQ lié à chaque dictionnaire-étage (du quantificateur original) est conçu en utilisant le procédé décrit dans le Chapitre 3. Au premier étage la distance Euclidienne pondérée [12] est utilisée, alors que pour les étages suivants, la distance Euclidienne est utilisée (les vecteurs-code correspondent aux vecteurs erreurs et non pas aux vecteurs LSF). Finalement, nous obtenons un quantificateur MTVQ correspondant à quatre structures TSVQ de résolutions (profondeurs) 7, 6, 6 et 6. Les quatre dictionnaires-étage du quantificateur MTVQ ont les mêmes vecteurs-code que ceux du quantificateur MSVQ original (du standard MELP) mais disposés dans différents ordres.

Pour les trames voisées, le codeur MELP se base sur le calcul de 10 amplitudes de transformée de Fourier (correspondantes aux dix premiers harmoniques du pitch), utilisant le signal résiduel de prédiction, pour capturer la forme de l'impulsion d'excitation. Ces 10 amplitudes de Fourier sont quantifiées avec un quantificateur vectoriel de 8 bits (256 niveaux). Nous utilisons les mêmes vecteurs-code que le quantificateur original pour concevoir un quantificateur TSVQ. Comme dans le codeur

MELP les 10 amplitudes de Fourier sont pondérées (basé sur un critère perceptuel) avec des poids fixes qui donnent plus d'importance à la région de basse fréquence; la distance Euclidienne est alors utilisée pour l'élaboration de la structure arborescente (l'arbre binaire). À la fin du processus de conception, nous obtenons un quantificateur TSVQ ayant les mêmes vecteurs-code (au niveau le plus élevé de l'arbre) que le quantificateur original (des amplitudes de Fourier) mais disposés dans différents ordres.

L'encodeur MELP modifié utilise un quantificateur MSVQ utilisant les nouveaux dictionnaires-étage pour la quantification des coefficients LSF et un quantificateur non structuré utilisant le nouveau dictionnaire pour la quantification des amplitudes de Fourier. Les nouveaux dictionnaires ont les mêmes vecteurs-code que ceux des dictionnaires originaux mais disposés dans différents ordres. Le flux-binaire généré est décodé par le décodeur MELP modifié en utilisant une structure MTVQ pour le décodage incorporé des coefficients LSF (où le nombre de bits varie de 0 à 25 bits) et une structure TSVQ pour le décodage incorporé des amplitudes de Fourier (où le nombre de bits varie de 0 à 8 bits) [107]. Il est important de noter que le codeur modifié a la même performance que le codeur original lorsque les coefficients LSF et les amplitudes de Fourier sont décodés avec 25 et 8 bits respectivement. La base de données test se compose de 32.51 minutes de parole extraite de la base de données TIMIT-test. La parole test est prétraitée avec le filtre IRS modifié puis filtré avec un filtre passe-bas pour une décimation à une fréquence d'échantillonnage de 8 kHz. La base de données test produit un total de 87,012 vecteurs tests LSF. En utilisant le codeur MELP modifié, les vecteurs tests sont encodés en utilisant 25 bits et les indices résultants sont décodés d'une manière incorporée (utilisant une structure MTVQ) avec le nombre de bits variant de 0 à 25 bits. Pour le décodage incorporé, la priorité est accordée aux étages inférieurs en assignant les bits disponibles à ces étages d'abord (tableau 6.1).

Pour évaluer la qualité de la quantification, la distorsion spectrale moyenne SD (spectral distorsion) et le pourcentage des «outliers» (les valeurs de SD supérieures à 4 dB et les valeurs comprises entre 2 et 4 dB) [12] pour l'ensemble des vecteurs tests sont tracés à la figure 6.2. Notons qu'il y a une dégradation progressive pour la distorsion spectrale moyenne quand le nombre de bits est décrémenté, ce qui confirme que le quantificateur LSF est échelonnable bit-par-bit.

Tableau 6.1: Schéma d'allocation de bits pour le décodage utilisant une structure MTVQ de quatre étages de résolutions $r_1 = 7$ bits et $r_2 = r_3 = r_4 = 6$ bits. Le nombre total de bits disponibles pour le décodage est noté par m avec le nombre de bits alloués aux premier, deuxième, troisième, et quatrième étages noté respectivement par m_1, m_2, m_3 et m_4.

m	(m_1, m_2, m_3, m_4)
25	(7, 6, 6, 6)
24	(7, 6, 6, 5)
23	(7, 6, 6, 4)
22	(7, 6, 6, 3)
21	(7, 6, 6, 2)
20	(7, 6, 6, 1)
19	(7, 6, 6, 0)
18	(7, 6, 5, 0)
17	(7, 6, 4, 0)
16	(7, 6, 3, 0)
15	(7, 6, 2, 0)
14	(7, 6, 1, 0)
13	(7, 6, 0, 0)
12	(7, 5, 0, 0)
11	(7, 4, 0, 0)
10	(7, 3, 0, 0)
9	(7, 2, 0, 0)
8	(7, 1, 0, 0)
7	(7, 0, 0, 0)
6	(6, 0, 0, 0)
5	(5, 0, 0, 0)
4	(4, 0, 0, 0)
3	(3, 0, 0, 0)
2	(2, 0, 0, 0)
1	(1, 0, 0, 0)
0	(0, 0, 0, 0)

Pour les amplitudes de Fourier, la base de données test (de parole voisée) produit un total de 65,469 vecteurs de 10 amplitudes de Fourier chacun qui sont encodés en utilisant 8 bits avec les indices résultants décodés d'une façon incorporée (utilisant une structure TSVQ) où le nombre de bits varie de 0 à 8 bits. Notons que pour 0 bit, le vecteur-code du niveau racine 0 (de l'arbre) est utilisé comme vecteur par défaut. Pour évaluer la qualité de la quantification, l'erreur quadratique moyenne (le vecteur d'amplitude de Fourier est perceptuellement pondéré comme dans le codeur standard) est tracée à la figure 6.3. Ici aussi on observe une augmentation graduelle de l'erreur quadratique moyenne lorsque le nombre de bits utilisés au décodage diminue bit-par-bit.

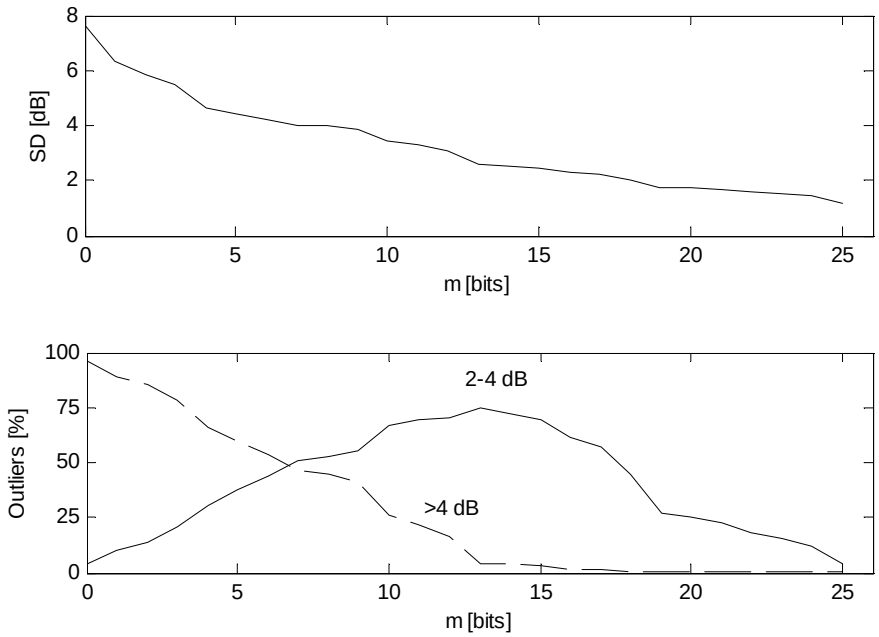


Figure 6.2: Résultats expérimentaux pour la quantification des coefficients LSF utilisant un quantificateur MTVQ à quatre étages (25 bits). En haut: Distorsion spectrale moyenne (SD) en fonction du nombre de bits utilisés au décodage. En bas: Pourcentage des « outliers » ayant un SD [2-4] et SD > 4 dB en fonction du nombre de bits utilisés au décodage.

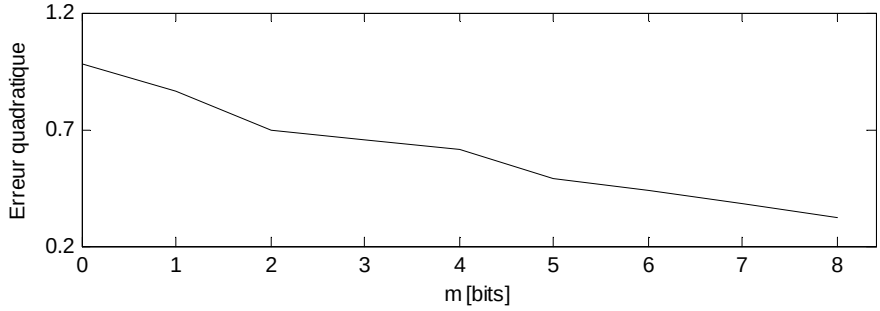


Figure 6.3: Résultats expérimentaux pour la quantification des amplitudes de Fourier utilisant un quantificateur TSVQ (8 bits). Erreur quadratique moyenne en fonction du nombre de bits utilisés au décodage.

Nous utilisons la norme ITU-T P.862 PESQ [99] pour mesurer la qualité de la parole synthétisée produite par le codeur standard MELP, lorsque les quantificateurs originaux des coefficients LSF et des amplitudes de Fourier sont remplacés par un MTVQ et un TSVQ respectivement. Le tableau 6.2 donne les valeurs PESQ (données sous forme graphique par les figures 6.4 et 6.5) obtenus pour le codeur MELP modifié utilisant de la parole propre et de la parole corrompue (par le bruit de voiture et de brouhaha avec un SNR de 10, 15, 20 et 25 dB) comme signal d'entrée de durée 10.67 minutes (Paragraphe 6.2). Notons que les 33 bits (les 8 bits pour les amplitudes de Fourier et les 25 bits pour les paramètres LSF) sont décodés de manière incorporée de 0 à 33 bits. Les 25 premiers bits (0 à 25) sont utilisés pour le décodage des coefficients LSF et les 8 bits suivants (26 à 33) sont utilisés pour le décodage des amplitudes de Fourier. Ainsi, pour $m = 0$ à 25 aucun bit (0 bit) n'est utilisé pour le décodage des amplitudes de Fourier (dans ce cas, le vecteur-code au niveau racine 0 du TSVQ est utilisé comme valeur par défaut) et pour $m = 25$ à 33, 25 bits sont utilisés pour le décodage des coefficients LSF. Notons que, lorsque aucun bit n'est utilisé pour le décodage des coefficients LSF (correspondant au cas : $m = 0$), les K vecteurs-code des niveaux racines de la structure MTVQ (à K étages) sont additionnés pour former le vecteur-code (par défaut) au décodage. Comme montré sur les figures 6.4 et 6.5, une dégradation graduelle de la qualité est obtenue pour la parole synthétique (pour les cas de la parole propre et de la parole corrompue) quand le nombre de bits disponible pour le décodage des coefficients LSF et des amplitudes de Fourier est décrémenté un par un.

Pour la parole non corrompue, la note PESQ est comprise entre 1.81 et 3.38 avec un incrément moyen de 0.05 par bit. Le changement de qualité est léger quand aucun bit n'est utilisé pour les amplitudes de Fourier et 19 bits sont utilisés pour des coefficients LSF. Lorsque $m < 7$ (le cas ou seul le premier étage du MTVQ est utilisé) le changement de la qualité est plus prononcé. Pour la parole corrompue par le bruit de voiture avec un SNR de 20 dB, la note PESQ est comprise entre 1.67 et 2.76 avec un incrément moyen de 0.03 par bit.

Tableau.6.2: Les notes PESQ en fonction du nombre de bits utilisés pour le décodage des coefficients LSF et des amplitudes de Fourier pour le codeur MELP modifié. Le signal d'entrée (de durée 10.67 minutes) est constitué de parole propre et de parole corrompue avec du bruit.

		Parole corrompue par le bruit de voiture avec un SNR de				Parole corrompue par le bruit de brouhaha avec un SNR de			
Nombre de bits	Parole Propre	25 dB	20 dB	15 dB	10 dB	25 dB	20 dB	15 dB	10 dB
33	3.376	2.913	2.764	2.563	2.215	2.857	2.702	2.497	2.212
32	3.372	2.912	2.763	2.563	2.214	2.858	2.704	2.494	2.211
31	3.368	2.910	2.760	2.561	2.215	2.855	2.699	2.493	2.209
30	3.365	2.905	2.759	2.561	2.212	2.852	2.698	2.489	2.208
29	3.362	2.900	2.753	2.556	2.209	2.848	2.695	2.489	2.207
28	3.359	2.897	2.750	2.555	2.209	2.845	2.694	2.487	2.204
27	3.355	2.899	2.751	2.556	2.210	2.846	2.695	2.490	2.206
26	3.353	2.896	2.750	2.553	2.208	2.847	2.694	2.488	2.204
25	3.353	2.894	2.745	2.551	2.207	2.845	2.694	2.489	2.204
24	3.318	2.878	2.732	2.538	2.198	2.828	2.676	2.475	2.193
23	3.297	2.866	2.720	2.527	2.191	2.815	2.667	2.467	2.188
22	3.294	2.858	2.717	2.529	2.190	2.812	2.665	2.464	2.185
21	3.288	2.854	2.713	2.526	2.185	2.807	2.662	2.461	2.184
20	3.277	2.855	2.713	2.527	2.185	2.804	2.655	2.458	2.181
19	3.276	2.852	2.711	2.522	2.186	2.802	2.654	2.457	2.181
18	3.222	2.817	2.683	2.502	2.169	2.767	2.621	2.430	2.161
17	3.176	2.792	2.663	2.482	2.151	2.736	2.601	2.411	2.147
16	3.164	2.786	2.655	2.478	2.146	2.733	2.596	2.405	2.140
15	3.122	2.764	2.638	2.464	2.134	2.706	2.573	2.381	2.123
14	3.107	2.758	2.632	2.458	2.133	2.699	2.568	2.378	2.122
13	3.097	2.752	2.627	2.452	2.130	2.691	2.562	2.374	2.117
12	2.939	2.647	2.540	2.386	2.081	2.586	2.472	2.300	2.064
11	2.869	2.588	2.486	2.340	2.048	2.523	2.413	2.256	2.029
10	2.827	2.563	2.468	2.323	2.028	2.493	2.387	2.232	2.007
9	2.708	2.468	2.378	2.245	1.967	2.407	2.308	2.161	1.955
8	2.674	2.444	2.353	2.227	1.955	2.383	2.291	2.151	1.947
7	2.665	2.438	2.348	2.222	1.946	2.377	2.282	2.144	1.940
6	2.568	2.358	2.274	2.151	1.883	2.301	2.217	2.083	1.884
5	2.505	2.303	2.225	2.111	1.857	2.250	2.169	2.044	1.856
4	2.402	2.231	2.156	2.047	1.816	2.186	2.107	1.985	1.801
3	2.178	2.020	1.962	1.860	1.636	1.984	1.919	1.814	1.659
2	2.091	1.896	1.843	1.746	1.531	1.876	1.821	1.734	1.596
1	1.872	1.766	1.727	1.651	1.469	1.740	1.691	1.613	1.487
0	1.815	1.689	1.669	1.629	1.448	1.694	1.682	1.644	1.544

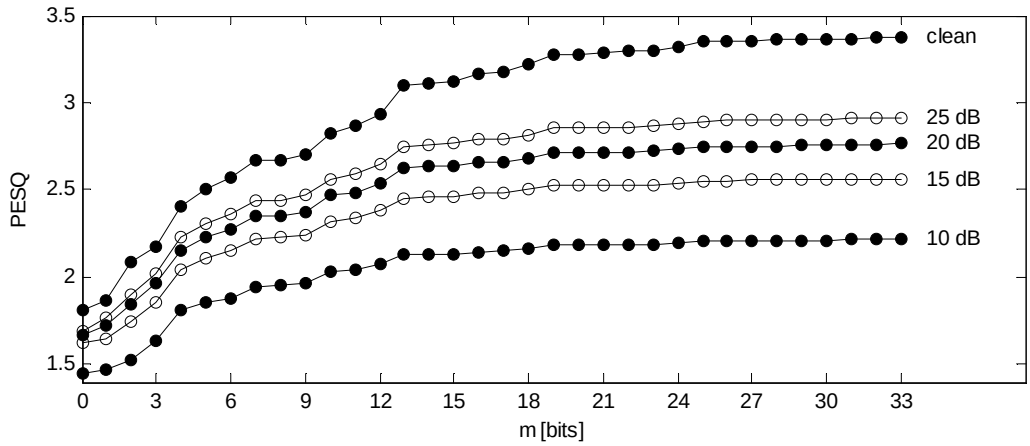


Fig. 6.4: La note PESQ en fonction du nombre de bits utilisés pour le décodage des coefficients LSF et des amplitudes de Fourier pour le codeur MELP modifié. La parole test utilisée a une durée de 10.67 minutes. Du haut vers le bas: parole propre et parole corrompue par le bruit de voiture avec un SNR de: 25, 20, 15 et 10 dB.

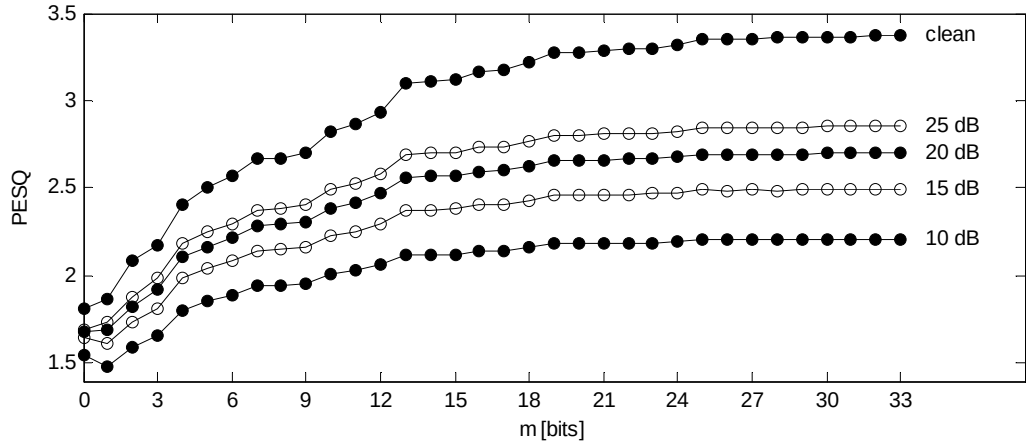


Fig. 6.5: La note PESQ en fonction du nombre de bits utilisés pour le décodage des coefficients LSF et des amplitudes de Fourier pour le codeur MELP modifié. La parole test utilisée a une durée de 10.67 minutes. Du haut vers le bas: parole propre et parole corrompue par le bruit de Brouhaha avec un SNR de: 25, 20, 15 et 10 dB.

La figure 6.6 montre quelques spectrogrammes de la parole synthétique, où on observe une perte progressive dans les détails spectraux quand le nombre de bits pour le décodage des amplitudes de Fourier et des coefficients LSF est réduit. Donc, le codeur modifié est échelonnable en débit binaire avec une granularité fine ou le changement de qualité est graduel lorsque le débit varie de 1288.9 à 2400 bits/s pour les trames de parole non voisées et de 933.3 à 2400 bps pour les trames voisés, avec un incrément de 44.4 bits/s. À 2400 bits/s le codeur modifié a la même performance que le codeur standard.

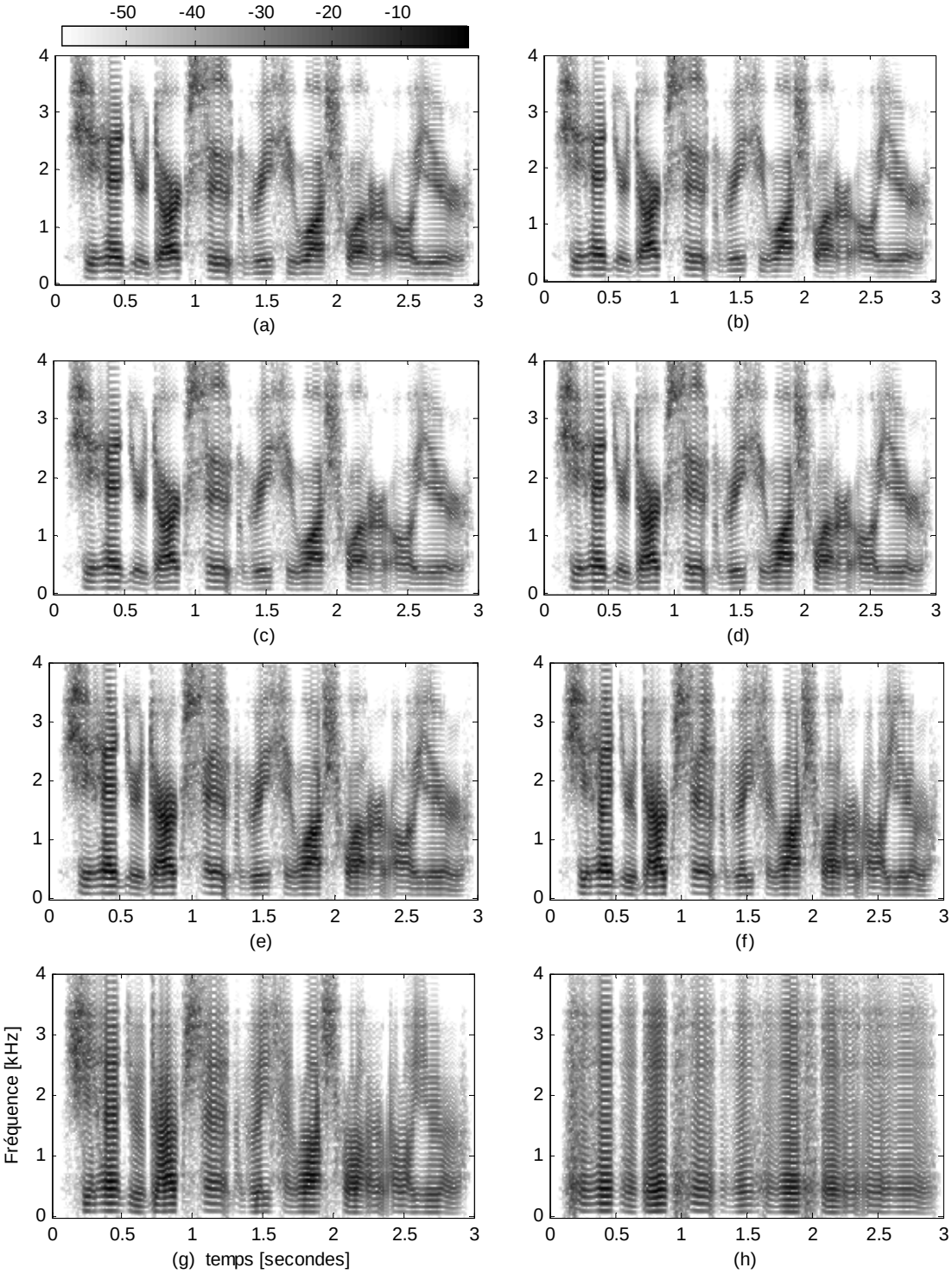


Figure 6.6: Les spectrogrammes pour de la parole synthétique correspondant à la phrase « She had your dark suit in greasy wash water all year » prononcée par un locuteur masculin : MELP original (a) et MELP modifié où le nombre de bits utilisés pour le décodage des amplitudes de Fourier et des coefficients LSF est égal respectivement à 5 et 25 bits (b), 0 et 25 bits (c), 0 et 20 (d), 0 et 15 bits (e), 0 et 10 bits (f), 0 et 5 bits (g), 0 et 0 bits (h). Le codeur MELP original utilise 8 et 25 bits pour le décodage des amplitudes de Fourier et les coefficients LSF respectivement.

6.4 Modification du codeur ITU-T G.729 (CS-ACELP)

Le codeur MELP appartient à la classe des codeurs paramétriques. Pour ce type de codeurs, le signal de parole est caractérisé en termes d'un ensemble de paramètres d'un modèle, où les paramètres sont extraits à partir du signal d'entrée (système en boucle ouverte) et séparément quantifiés, par conséquent une quantification incorporée peut être appliquée efficacement à certains paramètres (les paramètres LSF et les amplitudes de Fourier dans notre cas) puisque les interactions entre les différents paramètres quantifiés ne sont pas très fortes. Le codeur hybride CELP (Chapitre 2) est basé sur le principe d'analyse-par-synthèse, où certains paramètres (par exemple : les séquences d'excitation extraites des dictionnaires adaptatif et fixe) sont sélectionnés selon une méthode en boucle-fermée basée sur les coefficients de prédiction quantifiés (les coefficients de prédiction étant déterminés en boucle ouverte). Ainsi, les dépendances entre les divers paramètres quantifiés sont plus fortes (que pour le cas du codeur MELP). Pour évaluer l'effet de la distorsion des coefficients LPC quantifiés (LSF quantifiés) lors de la phase de décodage dans le cadre d'un codeur CELP, nous utilisons la quantification incorporée des coefficients LSF pour le codeur G.729 et nous proposons un schéma d'allocation de bits plus efficace que celui proposé en [66].

Le codeur standard G.729 (CS-ACELP) [38, 39] (Chapitre 2) est basé sur le modèle de prédiction linéaire avec excitation par code (CELP). Le codeur opère sur des trames de parole de 10 ms correspondant à 80 échantillons pour une fréquence d'échantillonnage de 8 kHz, où 80 bits sont utilisés pour chaque trame produisant ainsi un débit binaire de 8 kbit/s. Pour la quantification des coefficients LSF, une prédiction MA (moyenne ajustée) commutée de 4ème ordre est utilisée pour prédire les coefficients LSF de la trame courante. La différence entre les coefficients calculés et les coefficients prédits est quantifiée en utilisant une structure MSVQ à deux étages. Le premier étage est un quantificateur vectoriel de dimension 10 utilisant un dictionnaire avec 128 entrées (7 bits). Le deuxième étage est un quantificateur vectoriel structuré de 10-bits, qui est implémenté sous forme d'une structure SVQ utilisant deux dictionnaires de dimension 5 chacun (les cinq coefficients LSF inférieurs et les cinq coefficients LSF supérieurs) et contenant 32 entrées (5 bits) chacun. Ainsi, un total de 18 bits par trame sont utilisés pour

la quantification des coefficients LSF: 17 bits sont utilisés pour quantifier les vecteurs erreur-de-prédiction de dimension 10 et un bit est utilisé pour sélectionner un des deux prédicteurs possibles (Chapitre 2). Pour concevoir le quantificateur des coefficients LSF basé sur une structure MTVQ, les dictionnaires-arbre (les structures arborescentes) associés au premier dictionnaire-étage, au deuxième dictionnaire-étage (partie inférieure), et au deuxième dictionnaire-étage (partie supérieur) sont séparément générés en utilisant le procédé de conception décrit dans le Chapitre 3 (en utilisant la distance Euclidienne). Finalement, nous obtenons un quantificateur MTVQ correspondant à trois structures TSVQ de résolutions (profondeurs) 7, 5 et 5 bits avec trois nouveaux dictionnaires-étages (de dimensions 10, 5 et 5) ayant les mêmes vecteurs-code que le quantificateur original mais arrangés dans différents ordres.

L'encodeur G.729 modifié utilise les nouveaux dictionnaires (ayant les mêmes vecteurs-code que ceux des dictionnaires originaux mais disposés dans différents ordres) pour produire un flux binaire. Le décodeur modifié utilise un quantificateur prédictif basé sur une structure MTVQ pour le décodage incorporé (de 0 à 17 bits) des indices des vecteurs erreurs-de-prédiction des coefficients LSF. Dans le cas où aucun bit n'est perdu (les indices transmis des vecteurs-code sont reçus au complet), le codeur G.729 modifié a la même performance que le codeur standard.

Une base de données de parole de durée 32.51 minutes (extraite de la base de données TIMIT-test) est filtrée par le filtre IRS modifié et sous-échantillonnée à une fréquence de 8 kHz. La base de données test produit un total de 194,779 vecteurs tests LSF. En utilisant le codeur G.729 modifié, les vecteurs tests sont encodés utilisant 17 bits avec les indices résultants décodés d'une façon incorporée (en utilisant un quantificateur prédictif basé sur une structure MTVQ) où le nombre de bits s'étend de 0 à 17 bits. Pour le décodage incorporé, le schéma 2 d'allocation de bits du tableau 6.3 est utilisé. La distorsion spectrale moyenne (SD) et le pourcentage des « outliers » pour l'ensemble des données test sont tracés à la figure 6.7. On observe une dégradation progressive de la distorsion spectrale moyenne quand le nombre de bits est décrémenté, confirmant que le quantificateur est échelonnable bit-par-bit.

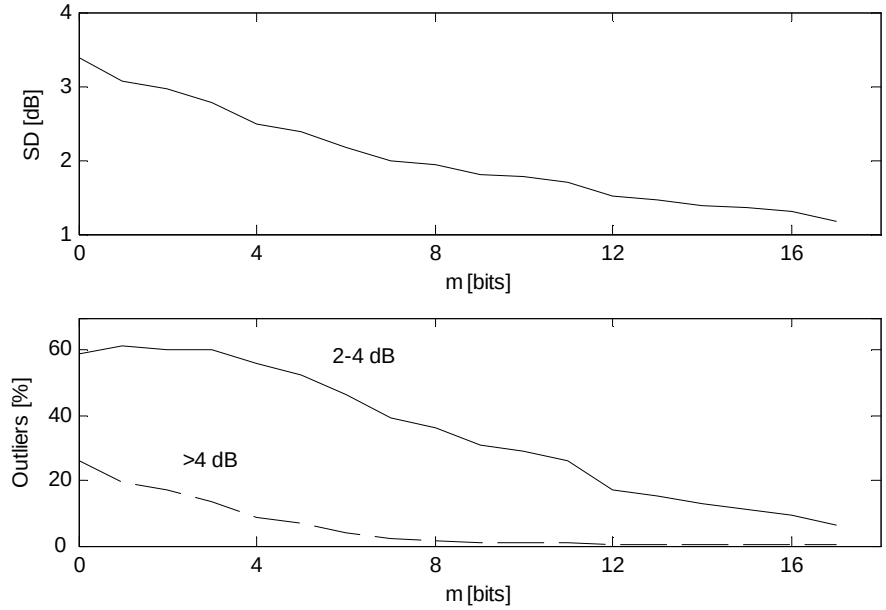


Figure 6.7: Résultats expérimentaux pour un quantificateur LSF prédictif basé sur une structure MTVQ à trois étages (17 bits pour les indices). En Haut: Distorsion spectrale moyenne (SD) en fonction du nombre de bits utilisés pour le décodage (utilisant le schéma 2 d'allocation de bit du tableau 6.3). En bas: Pourcentage de «outliers» ayant une SD dans l'intervalle [2-4] dB et SD > 4 dB en fonction du nombre de bits utilisés pour le décodage.

Le tableau 6.3 donne deux schémas d'allocation de bit pour le décodage incorporé des indices des vecteurs-code où le schéma 1 correspond à celui utilisé dans [66]. Pour les deux schémas (d'allocation de bit) la priorité est accordée au premier étage. Pour le schéma 1 la même priorité est accordée au deuxième étage partie-inférieure, et au deuxième étage partie-supérieure. Lorsque le nombre de bits disponible n'est pas divisible par 2**,** le deuxième étage partie-inférieure est assignée 1 bit de plus que le deuxième étage partie-supérieure. Pour le schéma 2 la priorité est accordée au deuxième étage partie-inférieure, en assignant les bits disponibles à ce dernier d'abord, comparé au deuxième étage partie-supérieure.

La figure 6.8 donne la note PESQ pour les deux schémas d'allocation de bit du tableau 6.3 où un signal de parole propre (non corrompu) de duré 10.67 minutes est utilisé (Paragraphe 6.2). Comparé au schéma 1, le schéma 2 améliore la performance (en termes de note PESQ). Il est bien connu que l'oreille humaine ne peut pas résoudre les différences aux hautes fréquences avec aussi de précision qu'aux basses fréquences (les composantes basses fréquences sont plus importantes); c'est-à-dire, les coefficients LSF inférieurs sont plus importants que les coefficients LSF supérieurs.

Tableau 6.3: Deux schémas d'allocation de bit utilisés pour le décodage des coefficients LSF pour le codeur G.729 modifié.

Nombre Total de bits	Scheme 1	Scheme 2
17	(7, 5, 5)	(7, 5, 5)
16	(7, 5, 4)	(7, 5, 4)
15	(7, 4, 4)	(7, 5, 3)
14	(7, 4, 3)	(7, 5, 2)
13	(7, 3, 3)	(7, 5, 1)
12	(7, 3, 2)	(7, 5, 0)
11	(7, 2, 2)	(7, 4, 0)
10	(7, 2, 1)	(7, 3, 0)
9	(7, 1, 1)	(7, 2, 0)
8	(7, 1, 0)	(7, 1, 0)
7	(7, 0, 0)	(7, 0, 0)
6	(6, 0, 0)	(6, 0, 0)
5	(5, 0, 0)	(5, 0, 0)
4	(4, 0, 0)	(4, 0, 0)
3	(3, 0, 0)	(3, 0, 0)
2	(2, 0, 0)	(2, 0, 0)
1	(1, 0, 0)	(1, 0, 0)
0	(0, 0, 0)	(0, 0, 0)

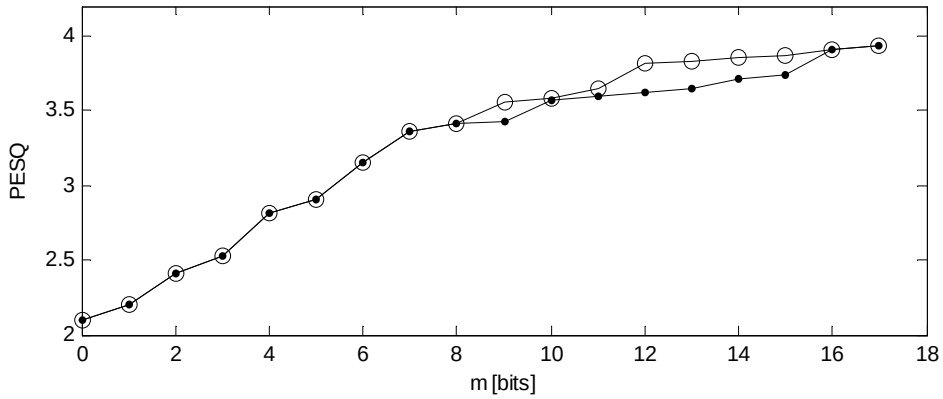


Figure 6.8: La note PESQ (pour les deux schémas d'allocation de bits du tableau 6.3) en fonction du nombre de bits utilisés pour le décodage des coefficients LSF pour le codeur G.729 modifié. De la parole non corrompue de durée 10.67 minutes est utilisée.

En outre, les sensibilités spectrales des coefficients LSF sont localisées (un changement d'un coefficient LSF donné produit un changement du spectre de puissance LPC seulement dans son voisinage) [12]. Donc, en distordant les cinq coefficients LSF quantifiés supérieurs (tout en maintenant les cinq coefficients inférieurs intacts) pendant le décodage serait plus avantageux que de distordre tous les coefficients LSF (les 10 coefficients). Le tableau 6.4 donne les notes PESQ (données sous forme graphique par

les figures 6.9 et 6.10) obtenues pour le codeur G.729 modifié en utilisant de la parole propre et de la parole corrompue (par le bruit de voiture et le bruit de Brouhaha avec un SNR de 10, 15, 20 et 25 dB) comme signal d'entrée de durée 10.67 min (Paragraphe 6.2). Comme montré aux figures 6.9 et 6.10, une dégradation graduelle de la qualité est obtenue pour la parole synthétique (propre et corrompue) quand le nombre de bits disponible pour le décodage des coefficients LSF est décrémenté un-par-un. Pour la parole propre et corrompue le changement de la qualité est légère quand les cinq coefficients LSF supérieurs sont distordus (correspondant à $m = 12 \text{ à } 16$). Pour la parole non corrompue, la note PESQ est comprise entre 2.10 et 3.93 avec un incrément moyen de 0.11 par bit.

Tableau 6.4: Les notes PESQ (utilisant le schéma 2 d'allocation de bit du tableau 6.3) en fonction du nombre de bits utilisés pour le décodage des coefficients LSF pour le codeur G.729 modifié. Le signal d'entrée (de durée 10.67 minutes) est constitué de parole propre et de parole corrompue avec du bruit.

		Parole corrompue par le bruit de voiture avec un SNR de				Parole corrompue par le bruit de brouhaha avec un SNR de			
Nombre de bits	Parole Propre	25 dB	20 dB	15 dB	10 dB	25 dB	20 dB	15 dB	10 dB
17	3.933	3.268	3.070	2.839	2.520	3.205	3.003	2.748	2.450
16	3.902	3.261	3.062	2.833	2.515	3.193	2.993	2.740	2.447
15	3.874	3.254	3.059	2.829	2.511	3.184	2.985	2.735	2.443
14	3.854	3.252	3.057	2.829	2.511	3.179	2.982	2.733	2.444
13	3.829	3.246	3.053	2.825	2.509	3.172	2.978	2.730	2.443
12	3.812	3.246	3.053	2.825	2.510	3.161	2.968	2.722	2.437
11	3.646	3.145	2.970	2.751	2.422	3.077	2.893	2.658	2.377
10	3.585	3.105	2.934	2.719	2.388	3.043	2.864	2.633	2.353
9	3.556	3.085	2.916	2.704	2.365	3.020	2.846	2.618	2.337
8	3.409	2.977	2.828	2.615	2.291	2.929	2.769	2.552	2.272
7	3.366	2.955	2.811	2.602	2.279	2.907	2.751	2.535	2.260
6	3.148	2.823	2.693	2.504	2.191	2.759	2.619	2.425	2.174
5	2.907	2.698	2.590	2.418	2.105	2.629	2.509	2.331	2.090
4	2.811	2.624	2.525	2.363	2.064	2.562	2.455	2.284	2.050
3	2.530	2.310	2.238	2.113	1.890	2.284	2.212	2.091	1.907
2	2.413	2.272	2.210	2.098	1.878	2.240	2.180	2.072	1.894
1	2.207	2.134	2.086	1.993	1.793	2.099	2.052	1.964	1.807
0	2.096	2.001	1.976	1.863	1.730	1.993	1.976	1.911	1.776

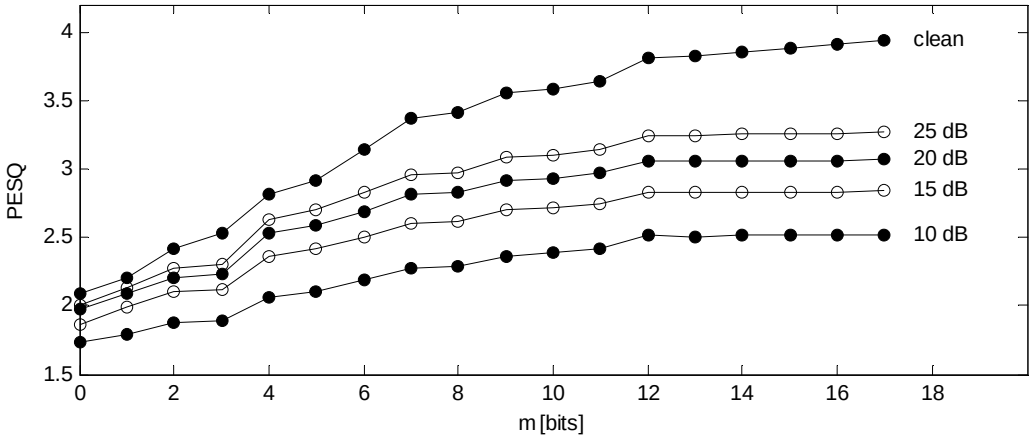


Figure 6.9: La note PESQ (utilisant le schéma 2 d'allocation de bit du tableau 6.3) en fonction du nombre de bits utilisés pour le décodage des coefficients LSF pour le codeur G.729 modifié. La parole test utilisée a une durée de 10.67 minutes. Du haut vers le bas: parole propre et parole corrompue par le bruit de voiture avec un SNR de 25, 20, 15 et 10 dB.

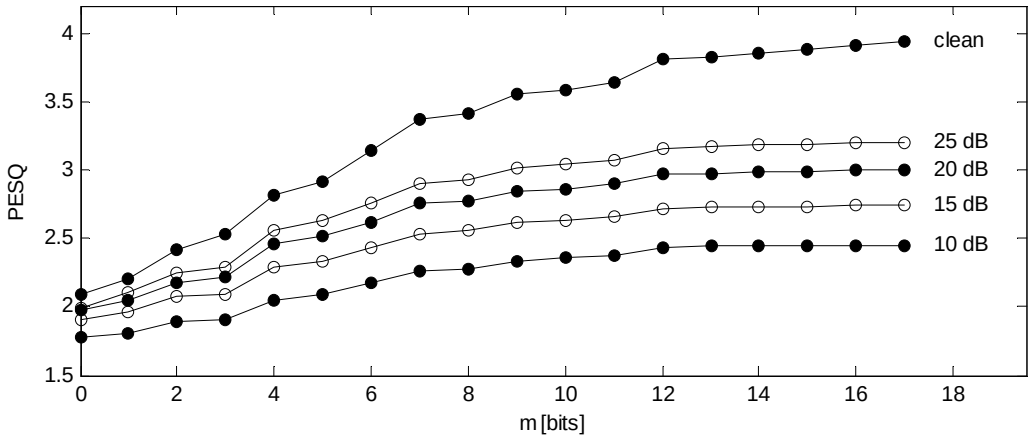


Figure 6.10: La note PESQ (utilisant le schéma 2 d'allocation de bit du tableau 6.3) en fonction du nombre de bits utilisés pour le décodage des coefficients LSF pour le codeur G.729 modifié. La parole test utilisée a une durée de 10.67 minutes. Du haut vers le bas: parole propre et parole corrompu par le bruit de Brouhaha avec un SNR de 25, 20, 15 et 10 dB.

Pour la parole corrompue par le bruit de voiture (avec un SNR de 20 dB), la note PESQ est comprise entre 1.98 et 3.07 avec un incrément moyen de 0.06 par bit. La figure 6.11 montre quelques spectrogrammes de la parole synthétique, où on observe une perte progressive dans les détails spectraux quand le nombre de bits pour le décodage des coefficients LSF est réduit. Le codeur modifié est échelonnable en débit binaire avec une granularité fine avec un changement progressif de la qualité quand le débit varie de 6300 bits/s à 8000 bits/s, avec un incrément de 100 bits/s. À 8000 bits/s le codeur modifié a la même performance que le codeur standard.

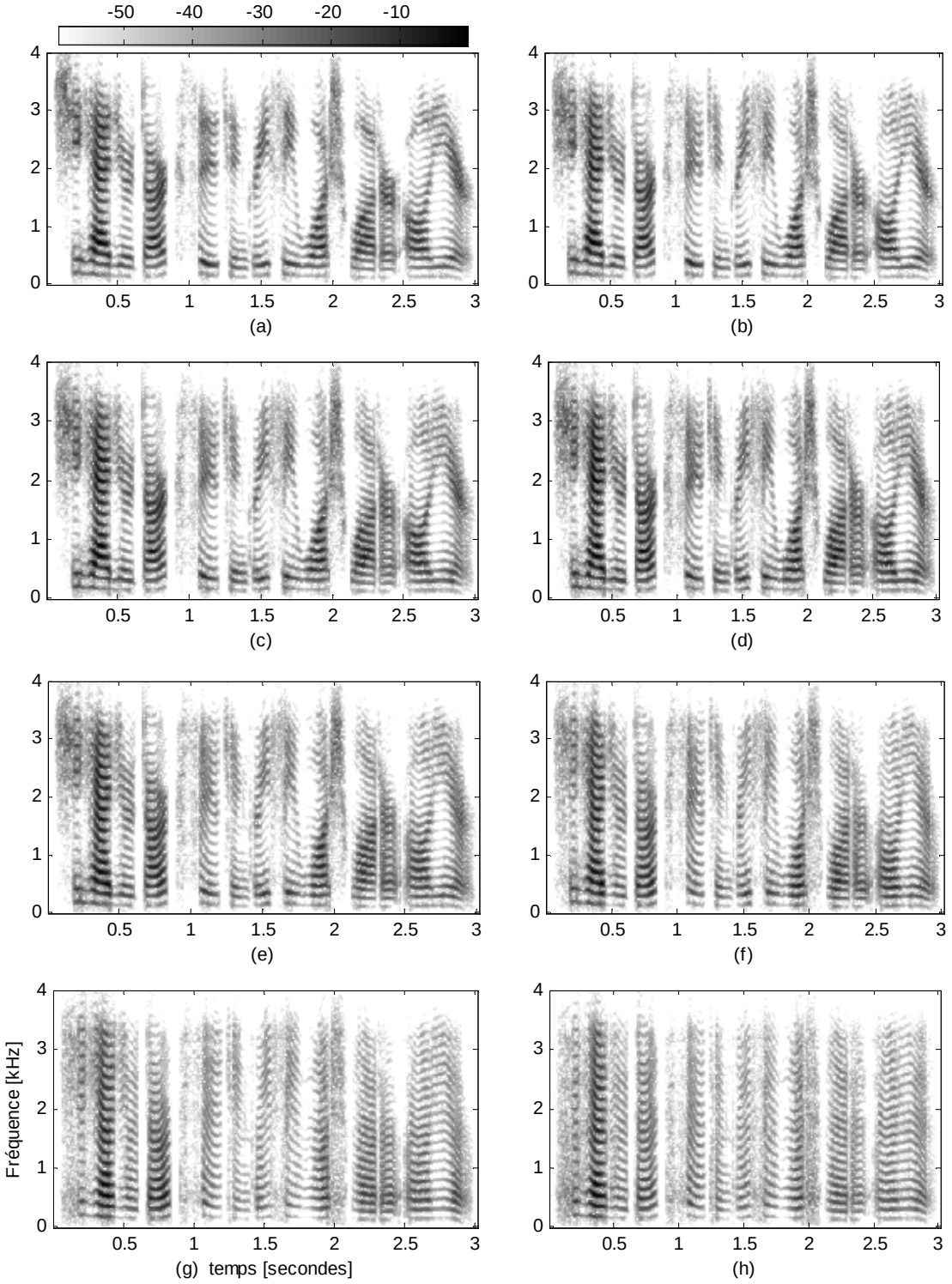


Figure 6.11: Les spectrogrammes pour de la parole synthétique correspondant à la phrase « She had your dark suit in greasy wash water all year » prononcée par un locuteur féminin : G.729 original (a) et G.729 modifié où le nombre de bits utilisés pour le décodage des coefficients LSF est égal à 12 bits (b), 10 bits (c), 8 bits (d), 6 bits (e), 4 bits (f), 2 bits (g), 0 bits (h). Le codeur G.729 original utilise 17 bits pour le décodage des coefficients LSF (erreur de prédiction).

Le schéma de quantification incorporé des coefficients LSF proposé dans ce travail peut être introduit dans la norme ITU-T G.729.1 (partie bande étroite) en conjonction avec les autres techniques utilisées dans ce codeur ou dans d'autres codeurs CELP échelonnable (Chapitre 2). En effet, certains codeurs CELP sont basés sur une certaine forme de structure échelonnable appliqué au dictionnaire fixe (excitation) comme dans l'ITU-T G.729.1 (partie bande étroite) [65], le standard MPEG-4 CELP [63]. Une autre méthode bien connue consiste à utiliser un codeur CELP pour produire le flux binaire de base (noyau) et pour améliorer la qualité de la parole décodée, un flux binaire de perfectionnement est produit en encodant le signal différence entre la parole originale et la parole décodé (Chapitre 2).

Du coté du décodeur, un quantificateur TSVQ exige environ deux fois la quantité de mémoire nécessaire pour le stockage des dictionnaires comparé à la quantification vectorielle conventionnelle, mais cette limitation n'est pas critique pour les plateformes modernes de traitement du signal [66]. En outre, chaque nœud interne d'un arbre binaire est le vecteur moyen de ses deux nœuds enfants (lorsque la distance Euclidienne est utilisée). Par conséquent, la connaissance de l'enfant gauche, sa probabilité et son parent (et sa probabilité) sont suffisants pour déterminer l'enfant droit. Ceci suggère qu'on a besoin de sauvegarder seulement les enfants gauches et leurs probabilités, réduisant de ce fait la complexité de mémorisation de moitié approximativement [108]. Ceci implique un coût de calcul additionnel pour générer les vecteurs-code non sauvegardés du coté du décodeur (dont la complexité de calcul est beaucoup moins importante que celle de la partie encodeur). Si seul les enfants droits au niveau le plus élevé d'une structure TSVQ sont supprimés (le dictionnaire-étage est réduit de moitié), la mémoire est réduite approximativement de 25% et le coût de calcul (pour générer les vecteurs-code non-sauvegardés) est réduit. Les remarques ci-dessus suggèrent que la quantification incorporée utilisant une structure TSVQ peut être appliquée aux codeurs de parole à large bande où la résolution des quantificateurs augmente.

6.5 Encodage à faible complexité des paramètres LSF

Deux bases de données séparées sont utilisées pour l'apprentissage et la validation de la procédure de recherche rapide décrite dans le Chapitre 4. La base de données d'apprentissage se compose de 67.59 minutes de parole (extraite de la base de données TIMIT-train) filtrée par le filtre IRS modifié et sous échantillonnée à 8 kHz. Une analyse LPC d'ordre 10 basée sur la méthode d'autocorrélation est effectuée sur des trames de 160 échantillons (20 ms) utilisant une fenêtre de Hamming. Les 10 coefficients de prédiction linéaire sont convertis dans le domaine LSF [43, 44] (Appendice B). La base de données d'apprentissage produit un total de 202,135 vecteurs de dimension 10. Une autre base de données test de 97,238 vecteurs LSF (calculée à partir de 32.51 minutes de parole extraite de la base de données TIMIT-test) est utilisée pour tester la performance de la procédure de recherche MJLS comparée aux procédures MSS et MSTS. La distorsion spectrale mesurée dans l'intervalle de fréquence 0-3 kHz [12] est utilisée pour évaluer la qualité de quantification des coefficients LSF.

Afin d'évaluer et de comparer les performances des procédures MSS, MSTS et MJLS, la même structure MTVQ à K étages devrait être utilisée par les trois algorithmes. Cependant, la procédure MSS utilise seulement les K dictionnaires-étage de tailles $N_s, s = 1 \text{ à } K$ (de la structure MTVQ) et la procédure MJLS utilise deux dictionnaires à l'étage s : le dictionnaire-étage de taille N_s et le dictionnaire-fusionné de taille η_s. Le quantificateur MTVQ à K étages est conçu comme suite :

1- Les K dictionnaires-étage (de tailles N_s, $s = 1 \text{ à } K$) sont élaborés en utilisant l'algorithme de conception conjoint, où tous les dictionnaires sont conjointement optimisés. L'algorithme de conception conjoint est plus performant que l'algorithme de conception séquentiel [13, 14].

2- Pour chaque dictionnaire-étage de taille N_s, le dictionnaire-fusionné de taille η_s correspondant est élaboré en utilisant l'algorithme de groupage (les probabilités des vecteurs-code du dictionnaire-étage sont estimées en utilisant les vecteurs d'apprentissage) et l'algorithme de permutation d'indices. Ceci est équivalent à trouver la meilleure séquence d'ensembles d'indices de dimension $n = N_s / \eta_s$ (Chapitre 3).

3- Pour chaque étage *s*, construire l'arbre binaire à partir du niveau le plus élevé $\log_2 N_s$ au niveau $\log_2 \eta_s$ en utilisant la recherche exhaustive conjointe sur chacun des η_s ensembles de *n* indices (chacun) de la séquence trouvée à l'étape 2 (Chapitre 3). Ainsi, le dictionnaire-fusionné de taille η_s (trouvé à l'étape 2) correspond à l'ensemble des vecteurs-code du niveau $\log_2 \eta_s$ de l'arbre binaire. Cet ensemble de vecteurs-code correspond au dictionnaire-arbre de taille η_s à partir duquel la procédure MSTS commence la recherche. Notons que les dictionnaires-arbre de tailles inférieures à η_s ne sont pas stockés en mémoire puisqu'ils ne sont pas utilisés.

Nous concevons quatre quantificateurs MTVQ correspondants aux débits de 22, 24, 26 et 28 bits/frame. Le tableau 6.5 donne la taille du dictionnaire-étage et la taille du dictionnaire-arbre (dictionnaire-fusionné) à partir duquel la procédure MSTS (la procédure MJLS) commence la recherche. Le tableau 6.6 donne la complexité de mémorisation (la quantité de mémoire requise pour le stockage des dictionnaires) pour les quatre quantificateurs du tableau 6.5 (voir les équations 4.11, 4.16 et 4.24). La figure 6.12 donne la distorsion spectrale moyenne pour l'ensemble des données test en fonction de la complexité de calcul aux débits de 22, 24, 26 et 28 bits/trame pour les trois procédures de recherche : MSS, MSTS et MJLS. Rappelons que le paramètre M_a est utilisé par les trois procédures de recherche, le paramètre η_s est utilisé par MSTS et MJLS, les paramètres M_q et M_r sont utilisés par MJLS et le paramètre M_t est utilisé par MSTS (Chapitre 4). Les tableaux 6.7 et 6.8 montrent les résultats obtenus pour 24 bits/trame. Notons que dans le tableau 6.7, le dictionnaire-fusionné de taille $\eta_1 = ... = \eta_4 = \eta = 16$ sont conçu à partir du dictionnaire-étage (comme indiqué dans le tableau 6.5); par contre les dictionnaires de taille $\eta = 4 \text{ et } 8$ correspondent aux dictionnaires-arbre de la structure MTVQ conçue selon la méthode proposée au Chapitre 3. Dans ce cas, la procédure de recherche MJLS est optimisé pour le dictionnaire-fusionné de taille $\eta = 16$. Cependant la procédure MJLS est aussi efficace lorsque le dictionnaire-fusionné est pris parmi un des dictionnaires-arbre d'une structure MTVQ. Ainsi si une structure TSVQ ou MTVQ est déjà conçu (utilisant un autre algorithme) la procédure MJLS peut être appliquée avec efficacité sur cette structure. La complexité de calcul est mesurée en nombre de calculs de

distances et en nombre d'opérations en virgule flottante (flops : floating point operations). Chaque addition/soustraction, multiplication ou comparaison est considérée comme un flop.

Tableau 6.5: La taille N_s du dictionnaire-étage et la taille η_s du dictionnaire-arbre et du dictionnaire-fusionnée (à partir duquel les procédures MSTS et MJLS commencent la recherche) correspondantes à l'étage s des quatre structures MTVQ de débits: 22, 24, 26 et 28 bits/trame.

Débit [bits/trame]	Étages			
	$N_1 : \eta_1$	$N_2 : \eta_2$	$N_3 : \eta_3$	$N_4 : \eta_4$
22 (8,7,7)	256 : 32	128 : 16	128 : 16	-
24 (6,6,6,6)	64 : 16	64 : 16	64 : 16	64 : 16
26 (8,6,6,6)	256 : 32	64 : 16	64 : 16	64 : 16
28 (7,7,7,7)	128 : 16	128 : 16	128 : 16	128 : 16

Tableau 6.6: Complexité de mémorisation correspondant à trois procédures de recherche (MSS, MSTS et MJLS) utilisant quatre quantificateurs (aux débits 22, 24, 26 et 28 bits/trame) correspondants à ceux du tableau 6.5.

Débit [bits/trame]	Complexité de mémorisation [réels]		
	MSS	MSTS	MJLS
22 (8,7,7)	5 120	9 600	5 760
24 (6,6,6,6)	2 560	4 480	3 200
26 (8,6,6,6)	4 480	8 160	5 280
28 (7,7,7,7)	5 120	9 600	5 760

À partir des tableaux 6.7, 6.8 et la figure 6.12, les remarques suivantes peuvent être faites :

• Pour la procédure MSTS, l'utilisation de la distance partielle CCp (équation 4.28) comparée à la distance totale CCf (utilisant l'équation 4.14) n'est pas efficace.

• Pour la procédure MJLS, l'utilisation de la distance partielle CCp comparée à la distance totale CCf est efficace à condition que la taille du dictionnaire-étage soit suffisamment plus grande que la taille du dictionnaire-fusionné. Par exemple, l'utilisation de la distance partielle n'est pas efficace à 24 kbits/s pour $\eta = 16$ (tableau 6.7), parce que le dictionnaire-étage est seulement quatre fois plus large que le dictionnaire-fusionné pour les quatre étages et dans ce cas l'utilisation de l'équation (4.26) pour la distance totale est plus efficace.

• Pour la procédure MSS une réduction significative de la complexité de calcul est obtenue en utilisant la distance partielle comparée à la distance totale. Par exemple, une réduction de 20% est réalisée à 24 kbits/s (tableau 6.7).

Tableau 6.7: Distorsion spectrale moyenne, « outliers » et complexité de calcul pour différentes configurations de la procédure MJLS utilisant une structure de quarte étages à un débit de 24 bits/trame (6 bits pour chaque étage). ND est le nombre de distances à calculer. CCf et CCp correspondent à la complexité de calcul utilisant la distance totale et la distance partielle respectivement.

		$M_a = 8$					
	M_r	SD [dB]	Outliers > 2 [%]	ND	CCf [kflops/f]	CCp [kflops/f]	MC [réels]
$\eta = 4$ $M_q = 3$	4	1.16	3.00	340	14.187	11.440	2720
	8	1.08	1.94	532	19.764	16.526	
	12	1.05	1.67	724	24.520	20.945	
	16	1.04	1.57	916	29.002	24.941	
	20	1.03	1.50	1108	33.446	28.737	
	23	1.03	1.48	1252	36.748	31.475	
$\eta = 8$ $M_q = 6$	4	1.20	3.68	344	12.915	11.600	2880
	8	1.11	2.33	440	16.720	15.244	
	12	1.07	1.88	536	19.896	18.307	
	16	1.05	1.68	632	22.583	20.897	
	20	1.04	1.59	728	25.123	23.321	
	24	1.03	1.56	824	27.547	25.605	
	28	1.03	1.51	920	29.902	27.781	
	32	1.03	1.48	1016	32.201	29.847	
$\eta = 16$ $M_q = 8$	4	1.18	3.44	480	15.343	14.456	3200
	8	1.10	2.22	528	17.591	17.774	
	12	1.06	1.84	576	19.699	20.386	
	16	1.05	1.69	624	21.466	22.465	
	20	1.04	1.60	672	23.196	24.415	
	24	1.03	1.53	720	24.832	26.205	
	28	1.03	1.52	768	26.392	27.873	
	32	1.03	1.50	816	27.888	29.430	
MSS		1.02	1.45	1600	44.598	35.671	2560

Tableau 6.8: Distorsion spectrale moyenne, « outliers » et complexité de calcul pour différentes configurations de la procédure MSTS utilisant une structure MTVQ à quatre étages à un débit de 24 bits/trame où chaque étage a la même résolution de 6 bits.

		$M_a = 8$					
η	M_t	SD [dB]	Outliers > 2 [%]	ND	CCf [kflops/f]	CCp [kflops/f]	MC [réels]
4	2	1.27	6.85	420	15.303	16.644	4960
	3	1.13	3.19	644	22.373	25.476	
8	2	1.26	6.46	420	15.005	16.37	4800
	3	1.13	3.15	598	20.931	23.561	
	4	1.07	2.17	800	27.059	31.474	
	5	1.05	1.79	950	31.637	37.647	
	6	1.04	1.64	1100	35.607	42.916	
	7	1.03	1.55	1250	39.822	48.463	
16	2	1.21	4.88	504	16.756	18.409	4480
	3	1.11	2.75	644	21.507	24.421	
	4	1.07	2.04	800	26.419	30.707	
	5	1.05	1.73	900	29.764	35.153	
	6	1.04	1.63	1000	32.517	38.639	
	7	1.03	1.55	1100	35.533	42.37	
MSS		1.02	1.45	1600	44.598	35.671	2560

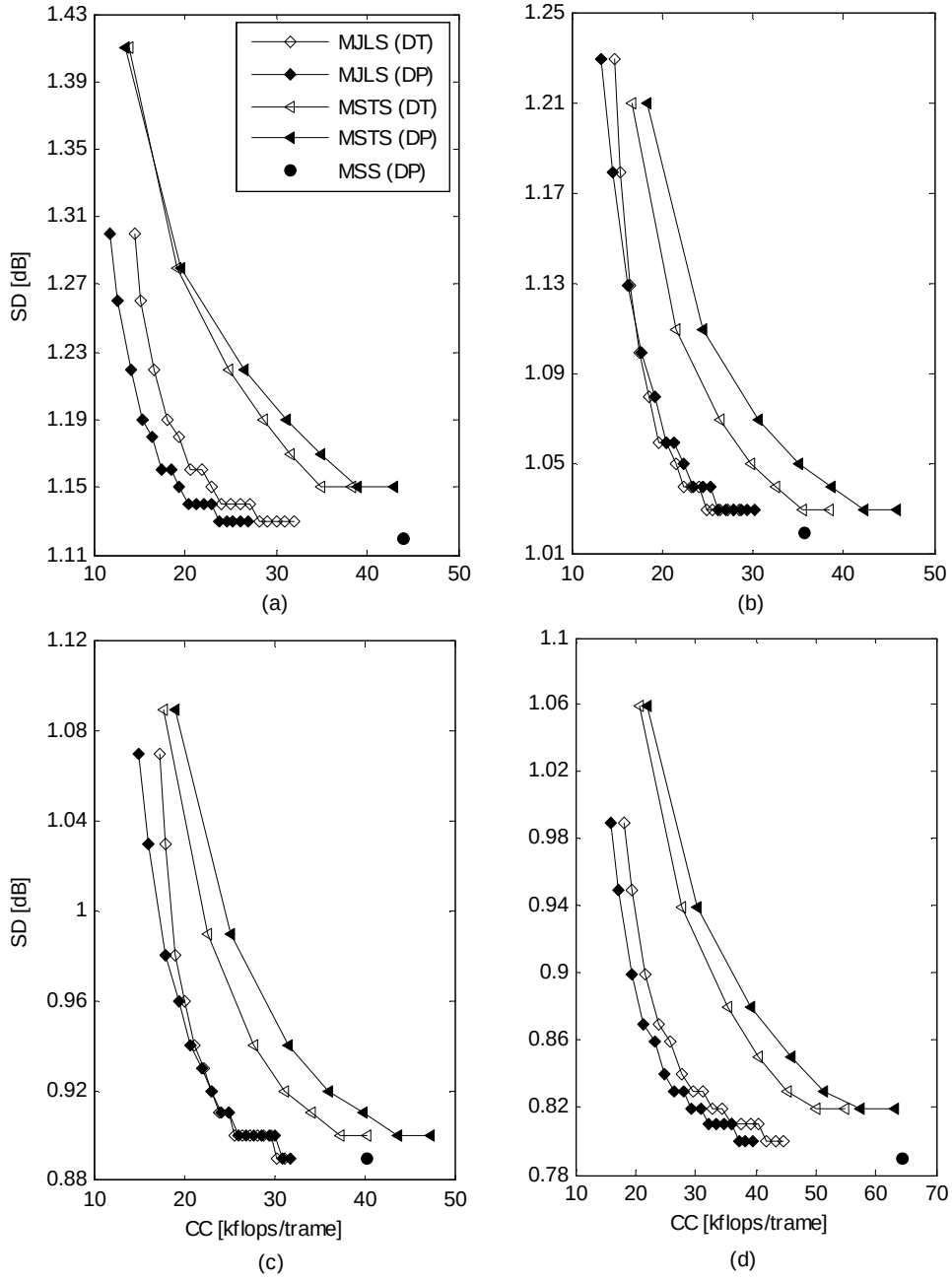


Figure 6.12: Distorsion spectrale moyenne (SD) en fonction de la complexité de calcul (CC), utilisant la distance partielle (DP) et la distance total (DT) aux débits de: 22 (a), 24 (b), 26 (c) et 28 bits/trame (d). Le dictionnaire-arbre/dictionnaire-fusionné de taille η_s (à l'étage *s*) à partir duquel la procédure MSTS/MJLS commence la recherche est donnée au tableau 6.5. Les paramètres de recherche sont comme suit: $M_a = 8$ (pour MSS, MSTS et MJLS), $M_q = 8$ et $M_r = 3,4,6,...32$ et 34 (pour MJLS), et $M_t = 2$ à 8 (pour MSTS).

• Une réduction significative de la complexité (de calcul et de mémorisation) est obtenue tout en produisant la même qualité ou une qualité meilleure (en termes de distorsion spectrale et/ou des outliers) pour la procédure MJLS comparée à la procédure MSTS. Par exemple, à 24 kbits/s et pour $\eta = 16$, la complexité de mémorisation est réduite environ de 28% (tableau 6.6) et la complexité de calcul CCf est réduite de 30% pour la même SD de 1.03 DB (tableaux 6.7 et 6.8).

La réduction en complexité de calcul (CC) peut être de 40% aux débits de 22 et 28 bits/s tandis qu'une meilleure qualité est réalisée pour MJLS comparé à MSTS (figure 6.12). Notons que l'algorithme MSTS devient inefficace pour des valeurs relativement petites du paramètre M_t.

• Pour la procédure MSTS, le pas de variation de CC est relativement grand lorsque la valeur du paramètre de recherche M_t (les paramètres η_s, pour l'étage *s*, et M_a étant fixes) est incrémenté. Pour la procédure MJLS, le pas de variation de CC est faible lorsque la valeur du paramètre M_r (les paramètres : η_s, M_q et M_a étant fixes) est incrémenté. Ceci signifie que l'algorithme MJLS est plus approprié pour le choix de la qualité désirée (SD) en fonction de la complexité de calcul en comparaison à l'algorithme MSTS.

• L'efficacité de l'algorithme MJLS est augmentée quand les tailles des dictionnaires-étage sont supérieures ou égal à huit fois la taille des dictionnaires-fusionnés comme aux débits de 22 et 28 bits/trame.

• La performance du MJLS peut être très proche de celle du MSS avec une réduction substantielle en complexité de calcul. À 24 kbits/s et pour $\eta = 16$ (tableau 6.7), on observe une performance comparable (SD est incrémenté seulement de 0.01 DB) tandis que la complexité de calcul pour le MJLS (utilisant la distance totale CCf) est réduite environ de 30% et 44% comparés au MSS utilisant la distance partielle (CCp) et la distance totale (CCf) respectivement. Il y a une légère augmentation en SD (0.02 à 0.04 dB) et dans la complexité de mémorisation (+25%) tandis que la complexité de calcul CCf du MJLS est réduite environ de 44% et 55% comparé aux CCp et CCf du MSS respectivement. aux débits de 22 et 28 bits/trame (figure 6.12), on observe une performance comparable alors que la complexité de calcul utilisant la distance partielle

est réduite plus de 40% pour le MJLS comparé au MSS. On observe une légère dégradation de performance (pour le MJLS) alors que la CC (utilisant la distance partielle) est réduite plus de 58% aux débits de 22 et 28 bits/frames comparée au MSS utilisant la distance partielle (figure 6.12). L'augmentation en complexité de mémorisation est légère (moins de 13%).

Notons que, dans le paragraphe 6.3, la procédure MSS utilisant la distance totale (exactement le même procédé que celui utilisé dans le codeur standard MELP) est utilisée pour localiser les vecteurs-code LSF du coté de l'encodeur du codeur MELP modifié. La procédure MSS peut être remplacée par la procédure MJLS pour réaliser une réduction significative en complexité de calcul tout en utilisant le décodage incorporé des indices des vecteurs-code du coté du décodeur pour réaliser un décodage échelonnable à granularité fine.

6.6 Conclusion

Deux normes de codeur de parole sont modifiées : un codeur paramétrique et un codeur hybride (de type CELP). Le codeur standard fédéral MELP est modifié en remplaçant le quantificateur original (pour la quantification des amplitudes de Fourier) et le quantificateur MSVQ original (pour la quantification des coefficients LSF) par un quantificateur TSVQ et un quantificateur MTVQ respectivement, qui sont conçus à partir des quantificateurs originaux correspondants (utilisant la technique de fusion de cellules). Le codeur standard G.729 est modifié en remplaçant le quantificateur prédictif original basé sur une structure MSVQ (pour la quantification des coefficients LSF) par un quantificateur prédictif basé sur une structure MTVQ (conçu à partir du quantificateur original). Pour le cas de la quantification SVQ dans le contexte d'une quantification vectorielle prédictive (le deuxième étage de la structure MSVQ du quantificateur prédictif des coefficients LSF), il est montré que la performance (en termes de note PESQ) est meilleure quand la priorité est accordée aux cinq coefficients LSF inférieurs au lieu d'accorder la même priorité à tous (les dix) les coefficients. Les codeurs modifiés sont échelonnables en débit binaire avec une granularité fine où une dégradation graduelle de la qualité de parole synthétique est obtenue quand le nombre de bits disponible (pour le

décodage incorporé des indices transmis) diminue bit-par-bit. Lorsque les indices sont reçus en totalité, le codeur modifié a la même performance que le codeur original.

La performance de la procédure rapide de recherche MJLS est comparée aux procédures MSTS et MSS. La quantification des paramètres LSF à différents débits binaires est réalisée pour fournir des résultats expérimentaux. Comparé à la technique MSTS, l'algorithme MJLS proposé donne de meilleurs résultats en termes de distorsion spectrale moyenne et/ou « outliers » avec une réduction de la complexité de mémorisation et une réduction significative de la complexité de calcul. En effet, la charge mémoire est réduite de 28% à 40% et la complexité de calcul de 30% à 40% alors qu'une meilleure qualité (en terme de SD et de «outliers») est obtenue pour des débits allant de 22 à 28 kbits/s. Comparé à l'algorithme bien connu de MSS, l'algorithme de MJLS peut réaliser une importante réduction de la complexité de calcul avec une faible dégradation de la performance (ou une performance comparable) et une légère augmentation de la complexité de mémorisation. En effet, la charge mémoire est augmentée de 13% à 25% et la complexité de calcul est réduite de 30% à 55% alors que la distorsion spectrale est augmentée de 0.01 à 0.04 dB pour des débits allant de 22 à 28 kbits/s.

Notons que la procédure MJTS a été comparée aux procédures MSTS et MSS et son efficacité a été prouvé en [88]. Cependant l'algorithme MJLS utilisant seulement deux dictionnaires à chaque étage (d'une structure à K étages) a une complexité de mémorisation plus faible que la procédure MJTS.

Conclusion

Un algorithme efficace de conception d'un quantificateur TSVQ binaire équilibré basé sur la technique de fusion de cellules est proposé. La structure arborescente est conçue depuis le niveau le plus élevé de l'arbre vers les niveaux les plus bas. L'idée principale de la méthode proposée est basée sur la construction d'un arbre binaire d'une certaine taille (certaine hauteur) comme une connexion de sous-arbres de petites tailles. Pour construire le niveau l de l'arbre, les 2^l nœuds (vecteurs-code) sont groupés en $2^l / n$ ensembles de n nœuds chacun en utilisant l'algorithme de groupage et la procédure de permutation d'indices. Pour chaque ensemble de n nœuds, le sous-arbre optimal correspondant (de hauteur $\log_2 n$) est conçu en utilisant la méthode de recherche exhaustive conjointe. Le processus de conception est répété (des niveaux supérieurs vers les niveaux inferieurs de l'arbre) jusqu'à ce que l'arbre binaire soit conçu en totalité. Des simulations (Chapitre 3) ont prouvée l'efficacité de l'algorithme de groupage ainsi que de la procédure de permutation d'indices. Donc, pour la méthode proposée dans ce travail, un ensemble de niveaux est construit conjointement au lieu d'un niveau à la fois pour les méthodes présentées en [15, 66]. La solution optimale consiste à construire toute la structure arborescente en entier (tous les niveaux de l'arbre sont construits conjointement). Notons que l'algorithme de groupage (utilisé conjointement avec la procédure de permutation d'indices) peut être adapté pour une construction descendante d'un arbre binaire équilibré à partir des vecteurs-code d'un dictionnaire non-structuré. Un quantificateur MTVQ à K étages peut être vu comme une structure de K étages où chaque étage correspond à une structure arborescente (un arbre binaire équilibré). Le quantificateur MTVQ est alors conçu en deux étapes : en premier lieu les K niveaux les plus élevés des K arbres sont construits et optimisés conjointement; en second lieu, la construction de la structure MTVQ est complétée de façon séquentielle du premier au dernier étage où la technique de fusion de cellules est appliquée pour compléter la construction de l'arbre binaire associé à chaque étage.

Des procédures de recherche rapides pour réduire la complexité de calcul exigée pour localiser des vecteurs-code pendant le processus d'encodage utilisant une quantification vectorielle multi-étages sont proposées. L'algorithme MJTS utilise une

structure MTVQ dont le coût de mémorisation est approximativement le double de celle d'une structure MSVQ. L'algorithme MJLS utilise un certain nombre de niveaux (de dictionnaires) à chaque étage et le coût de mémorisation peut être proche de celle d'une structure MSVQ lorsque seuls deux dictionnaires sont utilisés à chaque étage. Dans ce cas, à chaque étage, le dictionnaire-fusionné est élaboré à partir du dictionnaire-étage utilisant l'algorithme de groupage et le procédé de permutation d'indices. La performance de la procédure de recherche MJLS est comparée aux procédures MSTS et MSS. La quantification des paramètres LSF à différents débits binaires est réalisée pour fournir des résultats expérimentaux. Comparé à la technique MSTS, l'algorithme MJLS proposé donne de meilleurs résultats en termes de distorsion spectrale moyenne et/ou les «outliers» avec une réduction de la complexité de mémorisation et une réduction significative de la complexité de calcul. En effet, la charge mémoire est réduite de 28% à 40% et la complexité de calcul de 30% à 40% alors qu'une meilleure qualité (en terme de SD et de «outliers») est obtenue pour des débits allant de 22 à 28 kbits/s. Comparé à l'algorithme MSS bien connu, l'algorithme de MJLS peut réaliser une importante réduction de la complexité de calcul avec une faible dégradation de la performance (ou une performance comparable) et une légère augmentation de la complexité de mémorisation. En effet, la charge mémoire est augmentée de 13% à 25% et la complexité de calcul est réduite de 30% à 55% alors que la distorsion spectrale est augmentée de 0.01 à 0.04 dB pour des débits allant de 22 à 28 kbits/s. Les mêmes remarques peuvent être faites pour l'algorithme MJTS comparé aux algorithmes MSS et MSTS. Cependant les procédures MJTS et MSTS ont la même complexité de mémorisation puisque utilisent la même structure de quantificateur.

Les différents algorithmes présentés dans ce projet ont été implémentés sous forme d'un logiciel interactif utilisant Visual Studio 2005. Certains outils de la bibliothèque STL2005 ainsi que la norme ITU-T P.862 PESQ de l'IUT-T ont également été intégrés dans ce logiciel permettant ainsi d'effectuer des prétraitements sur une base de données de parole. Aussi, une base de données de parole initiale peut être comparée à une base de données de parole dégradée. Le logiciel implémenté peut ainsi être utilisé par d'autres chercheurs comme un outil normalisé (utilisant des outils normalisés de l'UIT-T) efficace

pour le prétraitement et l'évaluation de bases de données assez large faisant ainsi économiser un temps important.

Deux normes de codeur de parole ont été modifiées : un codeur paramétrique et un codeur hybride de type CELP. Le codeur standard fédéral MELP est modifié en remplaçant le quantificateur non-structuré original (pour la quantification des amplitudes de Fourier) et le quantificateur MSVQ original (pour la quantification des coefficients LSF) par un quantificateur TSVQ et un quantificateur MTVQ respectivement conçus par la technique de fusion de cellules à partir des quantificateurs originaux. Le codeur modifié est échelonnable en débit binaire avec une granularité fine où le changement de qualité est graduel lorsque le débit varie de 1288.9 à 2400 bits/s pour les trames de parole non voisées et de 933.3 à 2400 bps pour les trames voisés, avec un incrément de 44.4 bits/s. À 2400 bits/s le codeur modifié a la même performance (pour un canal non bruité) que le codeur standard puisque (dans ce cas) le décodeur modifié utilise les même vecteurs-code que ceux du décodeur standard. Le codeur standard G.729 est modifié en remplaçant le quantificateur prédictif original basé sur une structure MSVQ (pour la quantification des coefficients LSF) par un quantificateur prédictif basé sur une structure MTVQ conçu à partir du quantificateur original. Le codeur modifié est échelonnable en débit binaire avec une granularité fine avec un changement graduel de la qualité quand le débit varie de 6300 bits/s à 8000 bits/s, avec un incrément de 100 bits/s. À 8000 bits/s le codeur modifié a la même performance (pour un canal non bruité) que le codeur standard puisque (dans ce cas) le décodeur modifié utilise les même vecteurs-code que ceux du décodeur standard. Pour le cas de la quantification SVQ dans le contexte d'une quantification vectorielle prédictive (le deuxième étage de la structure MSVQ du quantificateur prédictif des coefficients LSF), il est montré que la performance (en terme de note PESQ) est meilleure quand la priorité est accordée aux cinq coefficients LSF inférieurs au lieu d'accorder la même priorité à tous (les dix) les coefficients.

Au niveau du décodeur, un quantificateur TSVQ exige deux fois la quantité mémoire nécessaire pour le stockage des dictionnaires comparé à une quantification conventionnelle. Cependant cette limitation n'est pas critique pour les plateformes modernes de traitement du signal. En plus, lorsque la distance Euclidienne est utilisée,

chaque nœud interne est le vecteur moyen (pondéré) de ses deux nœuds enfants. Par conséquent, la connaissance du nœud enfant gauche (et sa probabilité) et du nœud parent (et sa probabilité) sont suffisants pour déterminer le nœud enfant droit. Ceci suggère que l'on a besoin de sauvegarder seuls les nœuds enfants gauches et leurs probabilités, réduisant ainsi la complexité de mémorisation approximativement de moitié [108]. Ceci implique un coût de calcul additionnel pour générer les vecteurs-code non-sauvegardés au niveau du décodeur qui est beaucoup moins complexe que l'encodeur. Si seuls les nœuds enfants droits du niveau le plus élevé de la structure TSVQ sont supprimés (le nombre de vecteurs-code du dictionnaire-étage est réduit de moitié), la quantité de mémoire est réduite approximativement de 25% et le coût de calcul additionnel (pour générer les vecteurs-code supprimés) est réduit. Les remarques ci-dessus suggèrent que la quantification incorporée utilisant un quantificateur TSVQ ou MTVQ puisse être appliquée aux codeurs de parole à large bande pour lesquels la résolution et/ou la complexité de mémorisation des quantificateurs peuvent augmenter [47, 48].

La quantification incorporée des coefficients LSF présentée dans ce travail peut être intégrée dans la norme G.729.1 (partie bande étroite) conjointement avec d'autres techniques utilisées dans ce codeur échelonnable ainsi que pour d'autres codeurs de parole prédictifs. Les algorithmes de conception et de recherche rapide, proposés dans ce travail, peuvent facilement être adaptés pour des paramètres autres que les coefficients LSF et les amplitudes de Fourier. L'algorithme de groupage (avec la procédure de permutation d'indices) peut être utilisé comme une étape d'un autre algorithme qui nécessiterait une procédure efficace pour diviser un ensemble de paramètres en sous-ensembles de même cardinalité. Enfin, comme perspective, il reste à étudier le cas où le canal de transmission est bruité (présence des erreurs de transmission).

Appendice A

Codage par Prédiction Linéaire

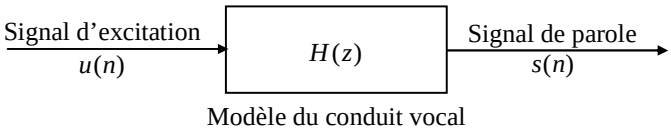
La Prédiction Linéaire LP (Linear Prediction) est un des outils les plus importants de l'analyse de parole [42]. Son efficacité et sa facilité de mise en œuvre rendent cette technique prédominante dans les systèmes de codage de parole à bas débit.

La prédiction linéaire à court terme considère que tout échantillon de parole peut être exprimé comme une combinaison linéaire d'échantillons précédents. Un ensemble de coefficients de prédiction est alors déterminé et utilisé pour supprimer la redondance proche entre les échantillons du signal de parole (redondance à court terme). La prédiction à long terme prend en compte la corrélation entre des échantillons éloignés du signal de parole (présente particulièrement pour les sons voisés). L'extraction de cette périodicité est obtenue par estimation de la période du pitch.

A.1 Prédiction à court terme

Dans le processus de production de la parole, la génération de chaque phonème est caractérisée par deux facteurs : la source d'excitation et la forme du conduit vocal. Pour modéliser le processus de production, il est nécessaire de modéliser le conduit vocal et la source d'excitation. Des études menées dans le but de définir un modèle produisant un

signal proche du signal de parole ont permis à Fant [29] de décrire un modèle de production linéaire. Le signal de parole $s(n)$ est modélisé par la sortie d'un filtre $H(z)$ excité par l'entrée $u(n)$. Le filtre $H(z)$ modélise le conduit vocal et l'excitation glottale $u(n)$ est modélisée par un train d'impulsions périodiques (à la cadence de la période du pitch) pour les sons voisés et un bruit blanc pour les sons non-voisés [29]. Le modèle de production est donné à la Figure A.1.



Modèle du conduit vocal

Figure A.1: Modèle de production de la parole

Le modèle *H*(z) le plus utilisé pour représenter le conduit vocal est un modèle pôle-zéro ou AutoRégressif à Moyenne Ajustée ARMA (Auto Regressive Moving Average). Dans ce cas, le signal $s(n)$ est une combinaison linéaire de ses échantillons antérieurs et du signal d'excitation du système $u(n)$:

$$s(n) = -\sum_{k=1}^{p} a_k s(n-k) + G\sum_{l=0}^{q} b_l u(n-l), \qquad b_0 = 1, \tag{A.1}$$

où le gain G et les coefficients $\{a_k\}_{k=1,..,p}$ et $\{b_l\}_{l=0,..,q}$ du filtre sont les paramètres du système. Les *p* échantillons passés étant considérés, *p* est l'ordre de prédiction linéaire. En appliquant la transformée en *z* à l'équation (A.1), la fonction de transfert du système $H(z)$ (dont les pôles du filtre représentent les formants du spectre) est donnée par :

$$H(z) = \frac{S(z)}{U(z)} = G\frac{1+\sum_{l=1}^{q} b_l z^{-l}}{1+\sum_{k=1}^{p} a_k z^{-k}}. \tag{A.2}$$

Deux cas particuliers sont alors possibles pour ce filtre :

• si $b_l = 0$ pour $1 \le l \le q$, $H(z)$ est réduit à un modèle tout-pôle appelé modèle AR (autorégressif).

• si $a_k = 0$ pour $1 \le k \le p$, $H(z)$ devient un modèle tout-zéro ou à Moyenne Ajustée MA (Moving Average).

Le modèle tout-pôle ou modèle AR est souvent utilisé pour son efficacité et sa simplicité de mise en œuvre. Le modèle AR représente parfaitement les résonances spectrales (les formants) des sons voisés. Les zéros apparaissent pour les sons nasals et les sons non voisés. Ces zéros peuvent être approximés par des pôles. En plus pour déterminer les zéros et les pôles d'un modèle ARMA, un ensemble d'équations non-linéaires doit être résolu. Par contre pour le cas d'un modèle tout-pôle, seul un ensemble d'équations linéaires devrait être résolu.

Pour un modèle AR (tout-pole) la fonction de transfert est donné par :

$$H(z) = \frac{G}{1 + \sum_{k=1}^{p} a_k z^{-k}}. \qquad \text{(A.3)}$$

Le facteur de gain *G* est généralement égal à 1, pour obtenir le filtre :

$$H(z) = \frac{1}{A(z)}\ , \qquad \text{(A.4)}$$

avec:

$$A(z) = 1 + \sum_{k=1}^{p} a_k z^{-k}\ , \qquad \text{(A.5)}$$

où les paramètres $\{a_k\}_{k=1,..,p}$ sont appelés coefficients de prédiction ou encore coefficients LP. La modélisation complète peut être décomposée en deux parties : La partie analyse et la partie synthèse (Figure A.2).

La partie analyse filtre le signal de parole original avec la fonction de transfert $A(z)$. Ce filtre tout zéro est défini comme le filtre d'analyse LP et permet d'extraire

l'information prédictible du signal et de définir un signal d'erreur de prédiction (appelé aussi signal résiduel ou encore signal d'excitation) entre le signal de parole original $s(n)$ et sa prédiction comme une combinaison linéaire des p échantillons précédents : $-\sum_{k=1}^{p} a_k s(n-k)$. Le signal résiduel est définie alors par :

$$e(n) = s(n) + \sum_{k=1}^{p} a_k s(n-k). \qquad \text{(A.6)}$$

Lors d'une analyse à court terme, la redondance proche entre les échantillons du signal de parole est supprimée par le filtre d'analyse LP $A(z)$. Ce filtre permet d'obtenir un signal de sortie de faible énergie, correspondant à l'erreur de prédiction $e(n)$.

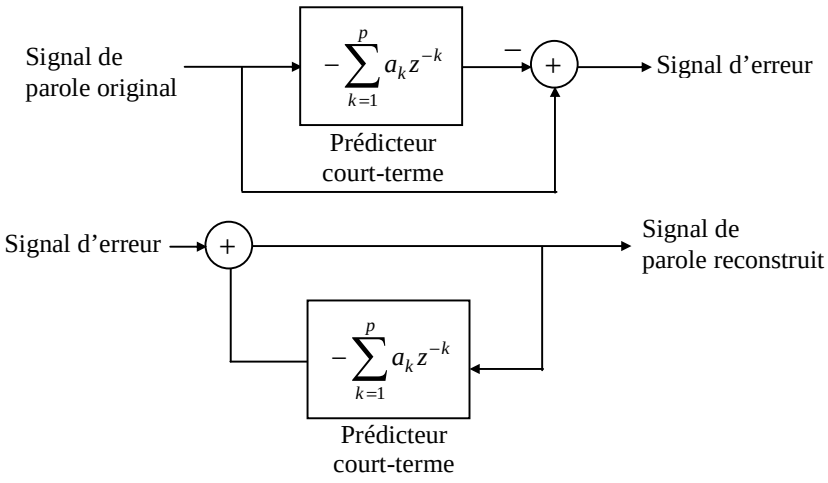


Figure A.2: Analyse et synthèse LP utilisant un prédicteur court-terme.

La partie synthèse effectue un filtrage de fonction de transfert $1/A(z)$. Ce filtre tout-pôle, appelé filtre de synthèse LP, permet de reconstruire, à l'aide d'un signal d'excitation approprié, un signal de parole reconstruit (appelé aussi signal synthétique). Le signal résiduel $e(n)$ est l'excitation idéale du modèle du conduit vocal $1/A(z)$. Notons que la fonction de transfert du filtre de synthèse décrit l'enveloppe spectrale du signal de parole.

Lorsque le signal résiduel calculé dans la partie analyse n'est pas utilisé comme signal d'excitation du filtre de synthèse, ou lorsque le filtre de synthèse n'est pas exactement l'inverse du filtre d'analyse, le signal de parole synthétique sera différent du signal de parole original.

A.1.1 Estimation des coefficients de prédiction

Différentes techniques existent pour estimer les coefficients de prédiction $\{a_k\}_{k=1,..,p}$ telles que la méthode d'autocorrélation et de covariance [24]. La méthode d'autocorrélation reste la plus populaire et la plus efficace grâce à l'algorithme de résolution de Levinson-Durbin [42], qui détermine les coefficients $\{a_k\}_{k=1,..,p}$ de manière à minimiser le signal résiduel par la méthode classique des moindres carrés. La méthode de covariance ne garantit pas la stabilité du filtre de synthèse. Pour la méthode d'autocorrélation, le signal de parole $s(n)$ est multiplié par une fenêtre d'analyse appropriée $w(n)$, correspondant généralement à une fenêtre de Hamming, de longueur N_w. Le signal pondéré $s_w(n)$ est donné par: $s_w(n) = w(n)s(n)$. L'énergie du signal résiduel (erreur de prédiction) est définie comme suit:

$$E = \sum_{n=-\infty}^{+\infty} e^2(n) = \sum_{n=-\infty}^{+\infty} \left(s_w(n) + \sum_{k=1}^{p} a_k s_w(n-k) \right)^2 . \qquad \text{(A.7)}$$

Les valeurs des coefficients $\{a_k\}_{k=1,..,p}$ qui minimisent l'énergie E sont calculés en annulant les dérivés partielles de E par rapport à chaque coefficient a_k pour $k = 1,.., p$. Ce qui donne p équations avec p inconnues, aussi connues comme les equations de Yule-Walker :

$$\sum_{k=1}^{p} a_k \sum_{n=-\infty}^{+\infty} s_w(n-i)s_w(n-k) = -\sum_{n=-\infty}^{+\infty} s_w(n-i)s_w(n), \quad 1 \le i \le p . \qquad \text{(A.8)}$$

Le signal fenêtré $s_w(n)$ est nul a l'extérieur de la fenetre $w(n)$ de longueur N_w et est supposé stationnaire sur cette durée. La fonction d'autocorrélation du signal fenêtré est donnée par:

$$R(i) = \sum_{n=i}^{N_w} s_w(n)s_w(n-i), \quad 1 \le i \le p . \qquad \text{(A.9)}$$

Sachant que la fonction d'autocorrélation est paire, l'équation (A.8) devient:

$$\sum_{k=1}^{p} R(|\, i-k\,|)a_k = -R(i), \quad 1 \le i \le p\,. \tag{A.10}$$

On obtient un système d'équations linéaires qui peut s'écrire sous la forme matricielle:

$$\begin{bmatrix} R(0) & R(1) & ... & R(p-1) \\ R(1) & R(0) & ... & R(p-2) \\ ... & ... & & ... \\ R(p-1) & R(p-2) & ... & R(0) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ ... \\ a_p \end{bmatrix} = -\begin{bmatrix} R(1) \\ R(2) \\ ... \\ R(p) \end{bmatrix}. \tag{A.11}$$

La matrice carrée d'autocorrélation est une matrice de Toeplitz symétrique. En effet, cette matrice est symétrique par rapport à la diagonale principale et les éléments d'une même diagonale sont égaux. Ceci facilite la recherche de la solution du système (A.11) utilisant un algorithme rapide tel celui de Levinson-Durbin [42]. La structure de Toeplitz garantie que le filtre d'analyse $A(z)$ est à phase minimal (les zéros sont à l'intérieur du cercle unité). Ceci assure donc la stabilité du filtre de synthèse $1/A(z)$. L'algorithme de Levinson-Durbin est un algorithme récursif sur l'ordre de prédiction. Le prédicteur optimal à l'ordre $m+1$ est déduit à partir de celui de l'ordre m. Les équations récursifs (A.12) sont calcules pour $m = 1, 2, ..., p$:

$$\begin{cases} k_m = -\left(R(m) + \sum_{k=1}^{m-1} a_k^{(m-1)} R(m-k) \right) \Big/ E^{(m-1)} \\ a_m^{(m)} = k_m \\ a_k^{(m)} = a_k^{(m-1)} + k_m a_{m-k}^{(m-1)} \qquad 1 \le k < m \\ E^{(m)} = (1-k_m^2) E^{(m-1)} \end{cases}, \tag{A.12}$$

avec la condition initiale $E^{(0)} = R(0)$. Les coefficients de prédiction du prédicteur d'ordre p sont: $a_k = a_k^{(p)}$, $1 \le k \le p$.

Les paramètres k_m $(1 \le m \le p)$ avec $|k_m| \le 1$ sont appelés coefficients de réflexion et sont utilisés comme une autre représentation des coefficients de prédiction.

L'analyse par prédiction linéaire ne peut estimer avec précision l'enveloppe spectrale des sons voisés avec une large fréquence du pitch. Elle peut produire des filtres de synthèse avec des pics spectraux accentués. Pour résoudre ce problème, une expansion de la largeur de bande des formants du spectre et généralement utilisée [18]. Le filtre tous-pole résultant, avec le facteur d'expansion de largeur de bande γ, a la forme suivante :

$$H'(z) = \frac{1}{A'(z)} = \frac{1}{A(z/\gamma)}. \qquad \text{(A.13)}$$

Les coefficients de prédiction après expansion sont alors: $a'_k = a_k \gamma^k$ $(1 \le k \le p)$. L'expansion de bande peut être calculée comme suite (où F_e est la fréquence d'échantillonnage):

$$\Delta B = -\frac{F_e}{\pi} \log(\gamma), \qquad \text{(A.14)}$$

A.2 Prédiction à long terme

Le principe de la prédiction linéaire consiste à considérer le signal de parole comme la sortie du filtre de synthèse $1/A(z)$. Cette prédiction est dite court terme puisque pour prédire la valeur du signal à l'instant *n* on utilise les *p* échantillons précédents. Le filtrage du signal de parole original par le filtre d'analyse $A(z)$ enlève les redondances des échantillons proches du signal de parole. Cependant, les trames de parole voisées contiennent une corrélation qui reste présente dans le signal résiduel à court terme. Les segments voisés étant les plus importants pour la perception globale d'un signal de parole, la plupart des codeurs se concentrent sur l'efficacité de codage des signaux d'excitation quasi-périodiques en utilisant une analyse à Long Terme LTP (Long Term Prediction) la plus efficace possible [18].

La mise en œuvre d'une Prédiction à Long Terme LTP, lors d'un codage par prédiction linéaire, est un moyen efficace de représenter la périodicité du signal de parole.

Cette analyse n'a pas d'effet sur des trames de parole non voisée qui n'ont pas de structure harmonique. La prédiction à long terme a pour but de modéliser les sons voisés qui possèdent des corrélations à long terme et se traduisent au niveau du signal de parole par une fréquence fondamentale F_0 (fréquence du pitch), représentée par un nombre d'échantillons T appelé pitch :

$$T = \frac{F_e}{F_0}. \tag{A.15}$$

Notons que dans le cas où la fréquence fondamentale F_0 n'est pas un diviseur de la fréquence d'échantillonnage F_e, la valeur de T n'est pas un nombre entier.

L'estimation du pitch [109-111] est très importante lors de l'analyse de la parole car l'oreille est sensible aux changements de fréquence F_0. Mais l'identification de la fréquence fondamentale est rendue difficile par plusieurs facteurs: Dans les régions de transition, les caractéristiques de la parole peuvent changer rapidement. La possibilité d'une présence simultanée d'une excitation voisée et non-voisée. La présence de bruit ambiant etc...Le pitch correspond à la période fondamentale du signal et représente le nombre d'échantillons de retard qu'il est nécessaire de prendre pour retrouver une portion de signal similaire. La redondance à long terme peut être modélisée en utilisant un filtre linéaire $P(z)$ du premier ordre. Le filtre d'analyse de pitch est donné par :

$$P(z) = 1 - bz^{-T}, \tag{A.16}$$

où b est le coefficient prédicteur (ou encore gain du pitch), correspondant au degré de périodicité avec $0 < b < 1$, et T l'estimation en nombre d'échantillons de la période fondamentale. Pour un signal voisé, la valeur de b est proche de 1. Pour un signal non voisé ou transitoire (qui n'a pas de structure périodique marquée), la valeur du gain optimal b est proche de zéro. Dans le domaine temporel, le filtre d'analyse de pitch $P(z)$ soustrait (de l'échantillon $x(n)$ du signal d'entrée) une valeur prédite à partir d'un échantillon retardé d'un délai T : $bx(n-T)$ avec T étant l'estimation de la période du

signal d'entrée. Dans le domaine fréquentiel, le filtre d'analyse LTP enlève la structure harmonique du signal d'entrée.

Puisqu'il n'y a pas de relation entre les fréquences d'échantillonnage F_e et fondamentale F_0, la période de pitch n'est pas nécessairement un nombre entier d'échantillons. Pour une meilleure représentation du pitch, une solution consiste à augmenter la résolution du signal. Kroon et Atal [110] proposent alors un filtre d'analyse $P(z)$ d'ordre supérieur. Ce filtre fournit une interpolation des échantillons et permet d'approximer le signal de parole, à une fréquence d'échantillonnage plus importante [111], par une combinaison linéaire de ses échantillons antérieurs. Toutefois ce traitement augmente la complexité des calculs et le nombre de coefficients du filtre. Généralement l'ordre du prédicteur est limité à 3:

$$P(z) = 1 - \sum_{k=-1}^{1} b_k z^{-T+k} \,. \tag{A.17}$$

Une autre méthode, pour améliorer l'efficacité de prédiction, est d'employer un pitch de meilleure résolution temporelle. Cette période, appelée pitch fractionnaire, pourra prendre des valeurs non entières. Dans ce cas le pitch T de l'équation (A.16) peut prendre des valeurs fractionnaires. La figure A.3 montre les parties analyse et synthèse du modèle de production de la parole utilisant un prédicteur court-terme et un prédicteur long-terme.

Lors de l'analyse, les coefficients de prédiction $\{a_k\}_{k=1,..,p}$ et les paramètres du pitch b et T (où le décalage T pouvant être fractionnaire) sont calculés. Les coefficients de prédiction sont calculés directement sur le signal de parole original (algorithme de Levinson-Durbin). Cette méthode est dite en boucle ouverte. Les paramètres du pitch peuvent être calculés en boule ouverte ou en boucle fermée. Pour la méthode dite en boucle ouverte, le pitch T est calculé en utilisant le signal de parole original ou encore le signal d'erreur de prédiction court-terme. Cela consiste, en générale, à la recherche d'un maximum local de la fonction de corrélation. La méthode en boucle ouverte est utilisée par les codeurs paramétriques. Pour ces codeurs les parties : analyse et synthèse sont complètement séparées.

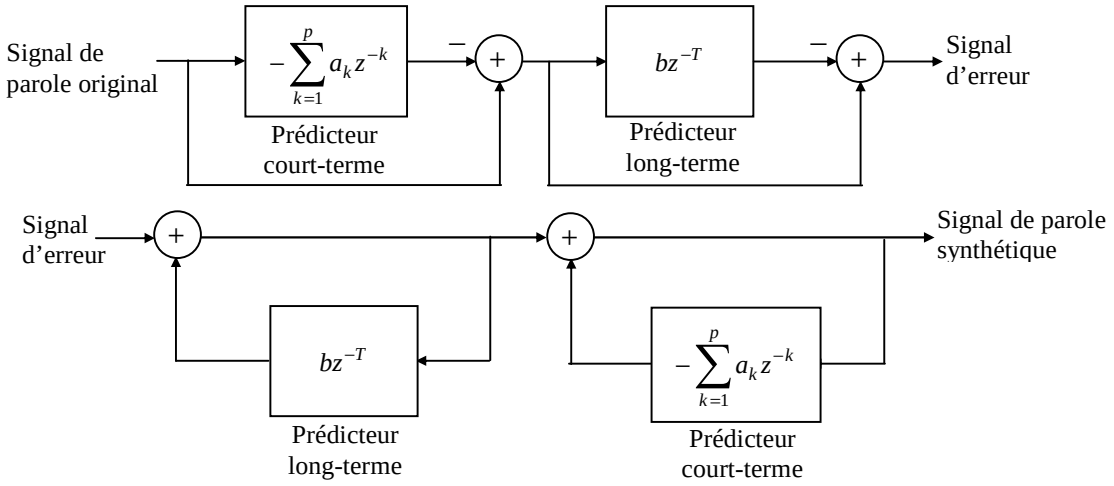


Figure A.3: Analyse et synthèse LP utilisant un prédicteur court-terme et un prédicteur long-terme.

Pour la méthode dite en boucle fermée, les paramètres du pitch sont déterminés de manière à optimiser le signal de parole synthétisé. On parle alors d'analyse par synthèse. Cette méthode est utilisée dans le cas de codeur CELP.

La synthèse consiste en le passage d'un signal d'excitation (le signal d'erreur dans le cas idéal) à travers le filtre de synthèse du pitch $1/P(z)$ permettant de reconstruire la structure harmonique (la périodicité) caractérisant les sons voisés et le filtre de synthèse $1/A(z)$ pour la reconstruction de la structure formantique ou encore l'enveloppe spectrale du signal de parole.

A.3 Dictionnaire adaptatif

Pour les codeurs CELP (Chapitre 3), le signal d'erreur est remplacé par une forme d'onde extraite d'un dictionnaire (fixe) composé de formes d'onde prédéterminées et normalisés. La forme d'onde étant pondérée par un gain. Le système de synthèse de la figure A.3 devient alors celui de la figure A.4.

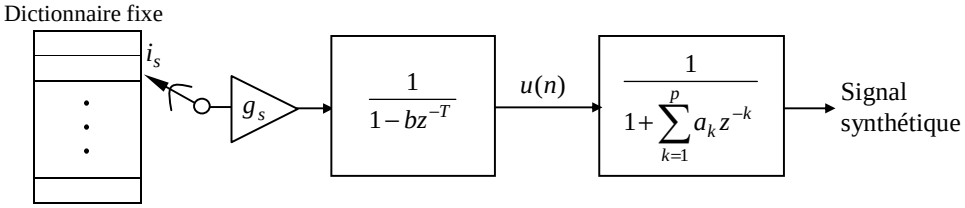


Figure A.4: Synthèse du codeur CELP utilisant un dictionnaire fixe et un prédicteur long-terme.

L'excitation est déterminée par sous-trame de L échantillons. L'excitation du filtre de synthèse LP à court-terme est donnée par (figure A.4) :

$$u(n) = g_s c_{i_s}(n) + bu(n-T)\ , \quad 0 \le n < L\ . \tag{A.18}$$

où la forme d'onde $\{c_{i_s}(n)\}_{n=0,...,L-1}$ est extraite du dictionnaire fixe (à l'indice i_s). Les paramètres du filtre de synthèse du pitch : le gain b et le décalage T (qui correspond à la valeur du pitch et peut être fractionnaire) peuvent être calculés en utilisant le signal de parole original ou encore le signal résiduel court-terme ; cette méthode est dite en boucle ouverte. La méthode dite en boucle fermée, consiste à rechercher les valeurs de b et T de tel manière que l'erreur entre le signal de parole original et synthétique soit minimisée. Dans cette méthode, le filtre de synthèse du pitch est remplacé par un dictionnaire adaptatif. Le dictionnaire adaptatif doit mémoriser les excitations passées. La figure A.5 montre un exemple d'un dictionnaire adaptatif sous forme d'un vecteur de longueur $T_{\max} + L$ échantillons.

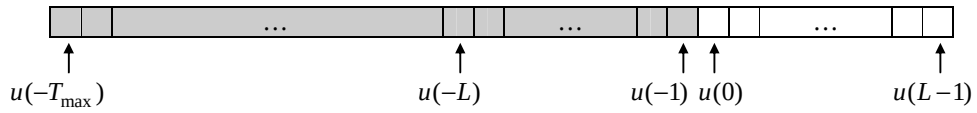


Figure A.5: Dictionnaire adaptatif avant la sélection de l'excitation courante $u(n)$, $0 \le n < L$

Les premiers $T_{\max}$ échantillons correspondent à ceux de l'excitation passée qui sont connus, et les L échantillons suivants correspondent à ceux de l'excitation courante $u(n)$, $n = 0$ à $L-1$, qui restent à déterminer. Ce vecteur est actualisé par décalage à gauche à chaque passage d'une sous-trame à l'autre. Le dictionnaire adaptatif peut aussi être représenté sous forme d'une matrice où chaque ligne $\boldsymbol{v}_j = [u(-L-j), ..., u(-1-j)]$ représente l'excitation passée (de L échantillons) correspondant au délai $T = L + j$. Pour j allant de 0 à $T_{\max} - L$, les valeurs du pitch autorisées sont alors comprises entre L et $T_{\max}$: $L \le T \le T_{\max}$.

$$\mathbf{V}=\begin{bmatrix} u(-L) & u(-L+1) & ... & u(-1) \\ u(-L-1) & u(-L) & ... & u(-2) \\ ... & ... & ... & ... \\ u(-T_{\max}+1) & u(-T_{\max}+2) & ... & u(-T_{\max}+L) \\ u(-T_{\max}) & u(-T_{\max}+1) & ... & u(-T_{\max}+L-1) \end{bmatrix} \begin{matrix} \boldsymbol{v}_0 \\ \boldsymbol{v}_1 \\ \\ \\ \boldsymbol{v}_{T_{\max}-L} \end{matrix} . \qquad \text{(A.19)}$$

La structure Toeplitz de la matrice **V** est exploitée pour accélérer la procédure de recherche exhaustive au sein du dictionnaire adaptatif pour localiser la valeur optimale du décalage (pitch) *T*.

L'excitation est calculée par sous-trame de *L* échantillons. Pour des retards *T* inférieurs à *L*, un problème se pose puisque des échantillons de l'excitation courante (qui n'est pas encore connue), font parties des échantillons de l'excitation passée. Dans la norme FS1016, ce problème est résolu en appliquant le principe de la périodicité: en complétant les échantillons inconnus par duplication de ceux qui sont connues [40] et les valeurs possibles du pitch appartiennent à l'intervalle [20, 147]. Dans la norme G.729 [39], les échantillons inconnus sont remplacés par ceux du signal résiduel court-terme et les valeurs possibles du pitch appartiennent à l'intervalle [20, 143]. En introduisant le dictionnaire adaptatif en remplacement du synthétiseur du pitch, la figure A.5 devient :

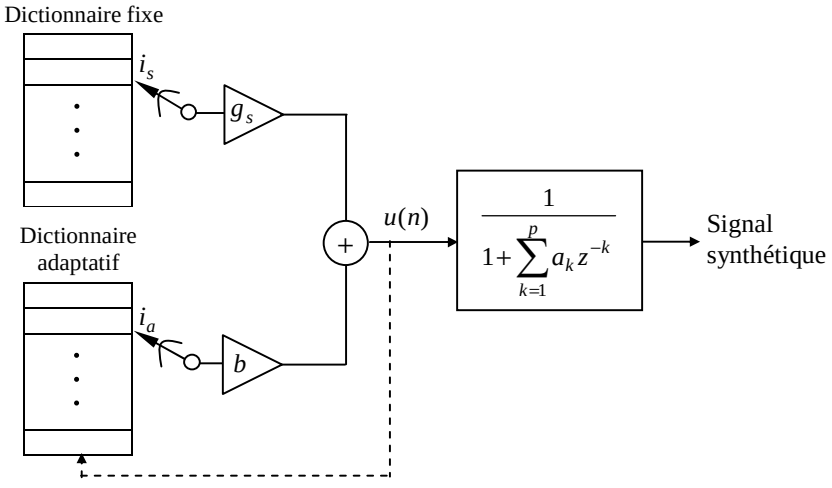


Figure A.6: Synthèse de codeur CELP utilisant un dictionnaire fixe et un dictionnaire adaptatif

Notons que le dictionnaire adaptatif doit être actualisé, puisque il doit contenir les échantillons de l'excitation passée par rapport à l'excitation courante qui est à déterminer.

Appendice B

Détermination des coefficients LSF

Dans les applications de codage de parole à bas débit, il est nécessaire de quantifier les coefficients de prédiction $\{a_k\}_{k=1,..p}$, pour les transmettre, en utilisant un nombre restreint de bits. Cependant, les coefficients LP ne sont pas appropriés à une quantification directe. En effet, leur codage nécessiterait 8 à 10 bits par coefficient [42]. En plus, la stabilité du filtre de synthèse LP (importante pour la qualité de parole synthétisée) n'est pas assurée. Afin de pallier ces difficultés, des paramètres mathématiquement équivalents aux coefficients de prédiction $\{a_k\}_{k=1,..p}$ sont calculés. Ces nouveaux coefficients préservent toute l'information des coefficients LP mais possèdent des propriétés particulières.

Le passage des coefficients de prédiction linéaire $\{a_k\}_{k=1,..p}$ à ceux de corrélation partielle (coefficients PARCOR ou encore coefficients de réflexion) $\{k_i\}_{i=1,..p}$ et inversement s'obtient de manière récursive à partir de l'algorithme de Levinson-Durbin (équation A.12). Ces coefficients correspondent aux facteurs de gain dans une structure en treillis du filtre d'analyse $A(z)$. Les coefficients de réflexion vérifient l'inégalité $|k_i| < 1$, ce qui garantit la stabilité du filtre de synthèse $1/A(z)$. Cette propriété rend les

coefficients de réflexions plus adaptés à la quantification que les coefficients LP. Toutefois, la distribution statistique des coefficients de réflexions est loin d'une loi uniforme sur leur intervalle de définition $[-1,1]$. Étant donné qu'il est plus simple de quantifier un scalaire présentant une distribution relativement uniforme, les paramètres *LAR* (Log Area Ratio), ou rapport d'aires logarithmiques, sont déduits des coefficients de réflexion ayant subi une transformation non-linéaire et sont utilisés pour leurs bonnes propriétés de quantification linéaire.

$$LAR_i = \log\left(\frac{1+k_i}{1-k_i}\right), \quad 1 \le i \le p\,. \tag{B.1}$$

Les lignes de raies spectrales LSP (Line Spectrum Pairs) ou les fréquences de rais spectrales LSF (Line Spectral Frequencies) sont l'alternative de représentation des coefficients de prédiction linéaire la plus utilisée [43, 44]. Les paramètres LSF, qui ont la propriété d'être ordonnés, se prêtent à une quantification robuste et efficace des coefficients LP. Pour déterminer les fréquences de raies spectrales, la technique utilisant les polynômes de Chebyshev est la plus courante [44].

B.1 Les fréquences de raies spectrales

Les paramètres LSF sont actuellement classés parmi les choix les plus appropriés et les plus utilisés pour représenter les coefficients LP. Pour déterminer les coefficients LSF, deux polynômes $P(z)$ et $Q(z)$ (symétrique et antisymétrique respectivement) d'ordre $p+1$ sont utilisés. Ces deux polynômes sont dérivés du filtre d'analyse $A(z)$:

$$P(z) = A(z) + z^{-(p+1)}A(z^{-1}) = (1+z^{-(p+1)}) + \sum_{k=1}^{p}(a_k + a_{p+1-k})z^{-k}\,, \tag{B.2}$$

$$Q(z) = A(z) - z^{-(p+1)}A(z^{-1}) = (1-z^{-(p+1)}) + \sum_{k=1}^{p}(a_k - a_{p+1-k})z^{-k}\,, \tag{B.3}$$

ce qui donne :

$$A(z) = [P(z) + Q(z)]/2 \tag{B.4}$$

Soong et Juang [112] ont montré que si le filtre de synthèse $1/A(z)$ est stable (le filtre d'analyse $A(z)$ est à phase minimale), alors toutes les racines des polynômes $P(z)$ et $Q(z)$ sont sur le cercle unité et sont entrelacés.

Si l'ordre p est paire, les polynômes $P(z)$ et $Q(z)$ possèdent respectivement une racine pour $z = e^{j\Pi} = -1$ et $z = e^{j0} = 1$ et peuvent donc être réécrits de la manière suivante :

$$P(z) = (1+z^{-1})P'(z) = (1+z^{-1})\sum_{k=0}^{p} t_k z^{-k} = (t_0 + t_p z^{-(p+1)}) + \sum_{k=1}^{p} (t_k + t_{k-1}) z^{-k}\,, \tag{B.5}$$

$$Q(z) = (1-z^{-1})Q'(z) = (1-z^{-1})\sum_{k=0}^{p} t'_k z^{-k} = (t'_0 - t'_p z^{-(p+1)}) + \sum_{k=1}^{p} (t'_k - t'_{k-1}) z^{-k}\,. \tag{B.6}$$

Par identification des équations (B.2) et (B.3) et des équations (B.5) et (B.6), les composantes t_k et t'_k s'expriment en fonction des coefficients LP en utilisant les équations récursives suivantes :

$$\begin{aligned}
&t_0 = t'_0 = 1 && t_p = t'_p = 1\\
&t_k = a_k + a_{p+1-k} - t_{k-1}, && t_{m-1} = a_m + a_{p+1-m} - t_m\,,\\
&t'_k = a_k - a_{p+1-k} + t'_{k-1}, && t'_{m-1} = a_{p+1-m} - a_m + t'_m\,,\\
&\text{avec}\quad k = 1, 2, ..., p/2 && \text{avec}\quad m = p,\ p-1, ..., (p/2)+2\,.
\end{aligned} \tag{B.7}$$

À partir des équations (B.7), on déduit les égalités suivantes : $t_k = t_{p-k}$ et $t'_k = t'_{p-k}$ pour $k = 0, 1, ..., (p/2)-1$. Donc seuls les $p/2$ premiers coefficients sont nécessaires pour spécifier chaque polynôme. Du fait que les racines des polynômes $P'(z)$ et $Q'(z)$ sont sur le cercle unité, ces polynômes peuvent être complètement définis par la position angulaire de leur racines. En outre, les racines se présentent sous forme de paires de nombres complexes conjugués. Par conséquent uniquement les positions angulaires situées dans le demi-cercle supérieur du plan z sont nécessaires pour déterminer complètement $P'(z)$ et $Q'(z)$. Les fréquences de raies spectrales LSF sont définis

comme les positions angulaires des racines de $P'(z)$ et $Q'(z)$ situées dans le demi-cercle supérieur du plan z. Ces paramètres LSF ont la caractéristique d'être entrelacés et ordonnés :

$$0 < \omega_1 < \omega_2 < ... < \omega_p < \pi, \tag{B.8}$$

où $\omega_1, \omega_3, ..., \omega_{p-1}$ sont les $p/2$ racines du polynôme $P'(z)$ et $\omega_2, \omega_4, ..., \omega_p$ sont les $p/2$ racines du polynôme $Q'(z)$. L'équation ci-dessus exprime la propriété d'ordonnancement des coefficients LSF. C'est une condition nécessaire et suffisante pour la stabilité du filtre de synthèse basé sur les coefficients LSF [44].

Les polynômes $P'(z)$ et $Q'(z)$ peuvent donc s'exprimer comme suite :

$$P'(z) = t_l z^{-l} + \sum_{k=0}^{l-1} t_k (z^{-k} + z^{-(2l-k)}) = 2z^{-l}\left[\frac{t_l}{2} + \sum_{k=0}^{l-1} \frac{t_k}{2}(z^{+(l-k)} + z^{-(l-k)})\right], \tag{B.9}$$

$$Q'(z) = t'_l z^{-l} + \sum_{k=0}^{l-1} t'_k (z^{-k} + z^{-(2l-k)}) = 2z^{-l}\left[\frac{t'_l}{2} + \sum_{k=0}^{l-1} \frac{t'_k}{2}(z^{+(l-k)} + z^{-(l-k)})\right], \tag{B.10}$$

avec $l = p/2$.

En divisant les polynômes $P'(z)$ et $Q'(z)$ par $2z^{-l}$ et en les évaluant sur le cercle unité ($z = e^{j\omega}$), les équations ci-dessus deviennent :

$$P'(\omega) = \frac{t_l}{2} + \sum_{k=0}^{l-1} t_k \cos((l-k)\omega), \tag{B.11}$$

$$Q'(\omega) = \frac{t'_l}{2} + \sum_{k=0}^{l-1} t'_k \cos((l-k)\omega), \tag{B.12}$$

avec $l = p/2$.

Pour rechercher les zéros des deux polynômes $P'(\omega)$ et $Q'(\omega)$, Soong et Juang [112] ont présenté une méthode utilisant la transformation discrète en cosinus (DCT). Les racines

correspondant aux coefficients LSF, sont trouvées en étudiant les changements du signe des polynômes $P'(\omega)$ et $Q'(\omega)$ sur l'intervalle $[0, \pi]$. Une autre méthode a été proposée par Kabal et Ramachandran [44] et qui consiste à utiliser les polynômes de Chebyshev définis par :

$$T_m(x) = \cos(m\omega) = 2.x.T_{m-1}(x) - T_{m-2}(x), \tag{B.13}$$

avec $T_0(x) = \cos(0) = 1$ et $T_1(x) = \cos(\omega) = x$. La transformation $x = \cos(\omega)$ effectue une correspondance entre le demi-cercle supérieur du plan z et l'intervalle réel $[-1, 1]$. En utilisant les polynômes de Chebyshev les equations (B.11) et (B.12) deviennent :

$$P'(x) = \frac{t_l}{2} + \sum_{m=1}^{l} t_{l-m} T_m(x), \tag{B.14}$$

$$Q'(x) = \frac{t'_l}{2} + \sum_{m=1}^{l} t'_{l-m} T_m(x), \tag{B.15}$$

avec $l = p/2$.

La méthode de dichotomie consiste alors à étudier les signes des polynômes $P'(x)$ et $Q'(x)$ pour des valeurs *x* appartenant à $[-1, 1]$. Chaque changement de signe correspond à l'existence d'une racine. Les passages par l'axe des zéros sont recherchés commençant de $x = +1$, avec un décrément de Δ. Une fois un intervalle contenant un passage par l'axe des zéros est retrouvé, la position de la racine est raffinée. Les racines des polynômes $P'(x)$ et $Q'(x)$ étant entrelacées, un zéro de $P'(x)$ servira de point de départ pour la recherche de la racine de $Q'(x)$ suivante, et vice versa. Les racines ainsi retrouvés correspondent aux lignes de raies spectrales LSP et les coefficients LSF sont obtenus par transformation inverse : $\omega = \arccos(x)$.

Notons qu'une évaluation efficace et récursive de $P'(x)$ et $Q'(x)$ pour une valeur donnée de *x* a été proposée par Kabal et Ramachandran :

$$
\begin{aligned}
&b_l(x)=1,\ \ b_{l+1}(x)=0 && b'_l(x)=1,\ \ b'_{l+1}(x)=0\\
&b_m(x)=2xb_{m+1}(x)-b_{m+2}(x)+t_{l-m} && b'_m(x)=2xb'_{m+1}(x)-b'_{m+2}(x)+t'_{l-m}\\
&P'(x)=xb_1(x)-b_2(x)+\frac{t_l}{2} && Q'(x)=xb'_1(x)-b'_2(x)+\frac{t'_l}{2}
\end{aligned}
\qquad \text{(B.16)}
$$

avec $m=l-1,l-2,...,1$ et $l=p/2$.

La Figure (B.1) montre la position des paramètres LSF par rapport au spectre (d'une trame de parole) estimé par les coefficients de prédiction linéaire. Les paramètres LSF présentent des propriétés importantes : La position d'un formant dans le spectre du signal de parole est liée généralement à un rapprochement de deux paramètres LSF consécutifs autour de la fréquence du formant considéré ; tandis qu'un espacement large entre deux paramètres LSF correspond à une vallée. Ainsi, il est donc possible d'identifier approximativement les zones auditivement importantes dans le spectre du signal.

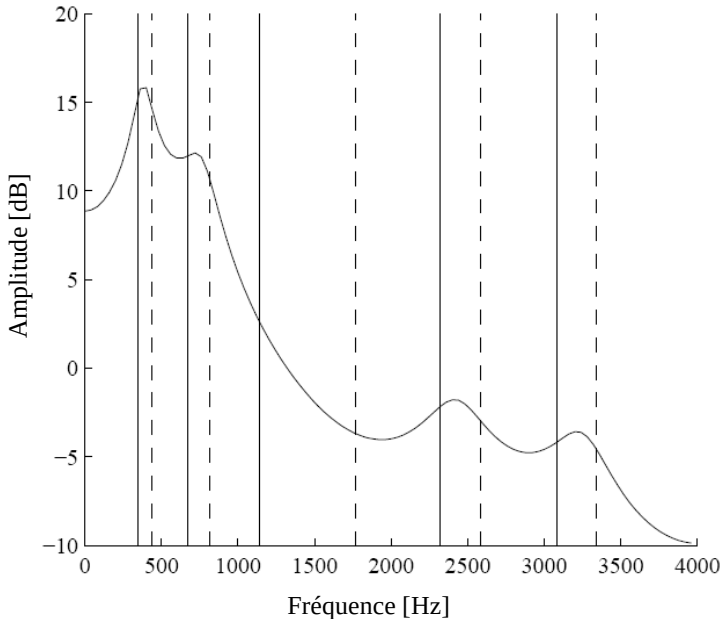


Figure B.1: Densité spectrale de puissance estimée par les coefficients de prédiction linéaire et les coefficients LSF correspondant exprimés en Hz (les lignes verticales).

En général, la sensibilité spectrale de chaque paramètre LSF est localisée, ce qui veut dire qu'un changement dans un paramètre cause un changement dans le spectre de puissance près de son voisinage. Grace à ces propriétés, les paramètres LSF peuvent être codés (quantifiés) efficacement.

Bibliographie

[1] C.E. Shannon, “A mathematical theory of communication”, *Bell Systems Technical Journal*, Vol. 27, No. 3-4, pp. 379-423, pp. 623-656, July, October, 1948.

[2] C.E. Shannon, “Coding theorems for a discrete source with a Fidelity criterion”, *IRE National Convention Record*, pp. 142-163, March 1959.

[3] S.P. Lloyd, “Least squares quantization in PCM”, *IEEE Trans. Inform. Theory*, Vol. IT-28, No.2, pp. 129-137, March 1982.

[4] J. Max, “Quantizing for minimum distortion”, *IEEE Transactions on Information Theory*, Vol. IT-6, pp.7-12, March 1960.

[5] A. Gersho and R.M. Gray, *Vector Quantization and Signal Compression*, Kluwer Academic Publishers, Boston, 1992.

[6] A. Gersho, “Principles of Quantization”, *IEEE Transactions on Circuits and Systems*, Vol. cas-25, No. 7, July 1978.

[7] J. Makhoul, S. Roucos, and H. Gish, “Vector quantization in speech coding”, *Proceedings of the IEEE*, Vol. 73, No. 11, pp. 1551-1588, November 1985

[8] W.-Y. Chan, “The Design of Generalized Product Code Vector Quantizers”, *IEEE ICASSP*, pp. 389-392, San Francisco, March 1992.

[9] Y. Linde, A. Buzo, and R.M. Gray, “An algorithm for vector quantization design”, *IEEE Trans. Comm.* Vol. 28, No. 1, pp. 84-95, Jan. 1980.

[10] I. Katsavounidis, C. Kuo, and Z. Zhang, “A new initialisation technique for generalized Lloyd iteration”, *IEEE Signal Processing Letters*, Vol. 1, pp. 144- 146, October 1994.

[11] M. Bouzid, “Codage Conjoint de Source et de Canal pour des transmissions par Canaux Bruités”, Thèse de doctorat, Université USTHB, Algérie, Avril 2006.

[12] K. K. Paliwal and B. S. Atal, “Efficient vector quantization of LPC parameters at 24 bits/frame”, *IEEE Trans. Speech Audio Process.*, Vol. 1, No. 1, pp. 3-14, Jan. 1993.

[13] W. F. LeBlanc, B. Bhattacharya, S. A. Mahmoud, and V. Cuperman, “Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding,” *IEEE Trans. Speech Audio Process.*, Vol. 1, No. 4, pp. 373-385, October 1993.

[14] W. F. LeBlanc, “Speech coding at low to medium Bit Rate”, Ph.D. Thesis, Department of Systems and Computer Engineering Carlton University, Ottawa, Canada, Sep. 1992.

[15] E. Riskin, R. Ladner, R. Wang, and L. Atlas, “Index Assignment for Progressive Transmission of Full-Search Vector Quantization”, *IEEE Transactions on Image Processing*, Vol. 3, No. 3, pp. 307-312, May 1994.

[16] W. Y. Chan and A. Gersho, “Constrained storage quantization of multiple vector sources by codebook sharing”, *IEEE Trans. Commun.*, Vol. 38, pp. 11–13, Jan. 1991.

[17] S. Ramakrishnan, K. Rose, and A. Gersho, “Constrained-Storage Vector Quantization with a Universal Codebook”, *IEEE Trans. on Image Process.*, Vol. 7, No. 6, June 1998.

[18] W. C. Chu, *Speech Coding Algorithms: Foundation and Evolution of Standardized Coders*, John Wiley & Sons, 2003.

[19] J.M. Salavedra; E. Masgrau, "APVQ encoder applied to wideband speech coding", *International Conference on Spoken Language*, Vol.2, pp. 941 - 944, 1996.

[20] S. Ragot, R. Lefebvre, R. Salami, and J.-P. Adoul, "Stochastic-algebraic wideband LSF quantization, *IEEE ICASSP*, Vol. 2, pp. 1169-1172, 2000.

[21] S. Ragot, J.-P. Adoul, R. Lefebvre, and R. Salami, "Low complexity LSF quantization for wideband speech coding", *IEEE Workshop on Speech Coding*, pp. 22-24, 1999.

[22] S. Ragot, H. Lahdili, and R. Lefebvre, "Wideband LSF quantization by generalized Voronoi codes", *Eurospeech, Aalborg, Denmark*, pp. 2319-2322, 2001.

[23] M. Xie and J.-P. Adoul, "Embedded algebraic vector quantizers (EAVQ) with application to wideband speech coding", *IEEE ICASSP*, vol. 1, pp. 240-243, 1996.

[24] D. O'Shaughnessey, *Speech Communication - Human and Machine,* Addison-Wesley Publishing Company, 1987.

[25] G. Madre, "Application de la transformée en nombres entiers à l'étude et au développement d'un codeur de parole pour transmission sur réseaux IP", Thèse de doctorat, Université de Bretagne Occidentale-École Doctorale SMIS, Octobre 2004.

[26] A.S. Spanias, "Speech Coding : A tutorial Review", *Proceedings of the IEEE*, Vol. 82, No. 10, pp. 1541-1582, October 1994.

[27] X. Huang, A. Acero and H.W. Hon, *Spoken language processing: a guide to theory, algorithm, and system development*, Prentice Hall PTR, New Jersey, 2001.

[28] L.R. Rabiner, and R.W. Schafer, *Digital Processing of Speech Signals*, Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[29] G. Fant, *Acoustic Theory of Speech Production*, Mouton, The Hague, 1960.

[30] R. Goldberg, and L. Riek, *A Practical Handbook of Speech Coders*, CRC Press, Boca Raton London New York Washington, D.C., 2000.

[31] G. Baudoin, J. Cernocky, P. Gournay, et G. Chollet, "Codage de la parole à bas et très bas débit", *Annales des Télécommunications*, Vol. 55, No. 9-10, pp. 1-21, 2000.

[32] ITU-T Recommendation G.711, "Pulse code modulation (PCM) of voice frequencies". *ITU Publication*, Nov 1988.

[33] J. Campbell and T. Tremain, "Voiced/unvoiced classification of speech with applications to the U.S. Government LPC-10e algorithm", *IEEE ICASSP*, pp. 473-476, 1986.

[34] B. S. Atal and J. R. Remde, "A new model of LPC excitation for producing natural sounding speech at low bit rates," *Proc. IEEE ICASSP*, Paris, pp. 614-617, May 1982.

[35] P. Kroon, E. F. Deprettere, and R. J. Sluyter, "Regular-Pulse Excitation: A novel approach to effective and efficient multi-pulse coding of speech," *IEEE Trans. Acoustics, Speech, Signal Processing*, Vol. ASSP-34, pp. 1054-1063, Oct. 1986.

[36] M. R. Schroeder and B. S. Atal, ”Code-excited linear predictive (CELP) : High quality speech at very low bit rates,” *IEEE ICASSP*, pp. 937-940, March 1985.

[37] ITU-T Recommendation G.728, “Coding of speech at 16 kbit/s using low-delay code excited linear prediction (LD-CELP)”, *ITU-T Publication*, Sep. 1992.

[38] Salami, R., C. Laflamme, J-P. Adoul, A. Kataoka, S. Hayashi, T. Moriya, C. Lamblin, D. Massaloux, S. Proust, P. Kroon, and Y. Shoham. ‘‘Design and Description of CS-ACELP: A Toll Quality 8 kb/s Speech Coder,’’ *IEEE Transactions on Speech and Audio Processing*, Vol. 6, No. 2, pp. 116–130, March 1998

[39] ITU-T Recommendation G.729, “Coding of speech at 8 kbits/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP)”, *ITU-T Publication*, Jan. 2007.

[40] J. P. Campbell, T. E. Tremain, and V. C. Welch, “The proposed Federal Standard 1016 4800 bps Voice Coder : CELP”, *Speech Technology Magazine*, pp. 58-64, April 1990.

[41] N. Moreau, “Codage prédictif du signal de parole a débit réduit : une présentation unifiée”, Annales des Télécommunications, Vol. 46, No. 3-4, pp. 223-239, 1991.

[42] John Makhoul, ”Linear Prediction: A Tutorial Review,” *Proc. of the IEEE*, Vol. 63, No. 4, April 1975.

[43] N. Sugamura and F. Itakura, “Speech Analysis and Synthesis Methods Developed at ECL in NTT–From LPC to LSP,” *Elsevier Speech Comm.*, pp. 199-215, Jan. 1986.

[44] P. Kabal, R.P. Ramachandran, “The computation of line spectral frequencies using Chebyshev polynomials”, *IEEE Trans. Acoust. Speech, Signal Process.* Vol. 34, pp. 1419-1426, Dec. 1986.

[45] N. Naja, “Construction de dictionnaires et Quantification vectorielle pour les codeurs de parole LSP-CELP”, Doctorat de l’Université de Rennes I, Mention Traitement du Signal et Télécommunications, Juin 1994.

[46] C. Laflamme, J-P Adoul,H. Y. Su, and S. Morissette, “On Reducing Computational Complexity of Codebook Search in CELP Coder Throught the Use of Algebraic Codes”, *IEEE ICASSP*, pp. 177-180, 1990.

[47] B. Bessette, R. Salami, R. Lefebvre, M. Jelínek,J. R.-Pukkila, J. Vainio, H. Mikkola, and K. Järvinen “The Adaptive Multirate Wideband Speech Codec (AMR-WB)”, *IEEE Trans. on Speech and Audio Process.*, Vol. 10, No. 8, pp. 620-636, Nov. 2002,

[48] R. Salami, C. Laflamme, J-P. Adoul, K. Jarvinen, J. Vainio, P. Kapanen, T. Honkanen, and P. Haavisto. “GSM Enhanced Full Rate Speech Codec’’, *IEEE ICASSP*, Vol.2, pp. 771–774, 1997.

[49] B. S. Atal and M. R. Schroeder, ”Predictive coding of speech signals and subjective error criteria,” *IEEE trans. on Acoustics, Speech and Signal Process.*, Vol. 27, No. 3, pp. 247-254, June 1979.

[50] A.V. McCree, and T.P. Barnwell III, “A mixed excitation LPC vocoder model for low bit rate speech coding” *IEEE Trans. on Speech and Audio Processing*, Vol. 3, No. 4, pp. 242 -250, July 1995.

[51] A. McCree, L. Supplee, R. Cohn, and J. Collura, "MELP: The New Federal Standard at 2400 bps*"*, *IEEE ICASSP*, pp. 1591-1594, 1997.

[52] Y. Hiwasaki, T. Mori, H. Ohmuro, J. Ikedo, D. Tokumoto, and A. Kataoka, "Scalable Speech Coding Technology for High-Quality Ubiquitous Communications", *NTT Technical Review*, Vol. 2 No. 3, pp. 53-58, Mar. 2004.

[53] H. Dong, "SNR and bandwidth scalable speech coding", Ph.D. thesis, Engineering Southern Methodist University, December 2002.

[54] ITU-T Recommendation G.727 "5-, 4-, 3- and 2-bits sample embedded adaptive differential pulse code modulation (ADPCM)", *ITU-T Publication*, December 1990.

[55] A. Haoui and D. G. Messerschmitt, "Embedded coding of speech: a vector quantization approach", *IEEE ICASSP*, Vol.10, pp. 1703-1706, 1985.

[56] T. Nomura**,** M. Iwadare, M. Serizawa, and K. Ozawa "A bitrate and bandwidth scalable CELP Coder", *IEEE ICASSP*, Vol.1, pp. 341- 344, 1998.

[57] F. Chen and I. Lee, "CELP based speech coding with fine granularity scalability," *IEEE ICASSP*, pp. II-145–II-148, 2003.

[58] S. Chompun, "Fine Granularity Scalability for MP-CELP Based Speech Coding with HPDR Technique", *IEEE Asia-Pacific Conference on Circuits and Systems*, Dec. 2004.

[59] R. D. De lacovo and D. Sereno, "Embedded CELP coding for variable bit-rate between 6.4 and 9.6 kbit/s", *IEEE ICASSP*, Vol. 1, pp. 681-684, 1991

[60] S. Zhang and G. Lockhart, "Embedded RPE Based on Multistage Coding", *IEEE Trans. on speech and audio process.*, Vol. 5, No. 4, pp. 367-371, July 1997.

[61] V. Berisha, "Bandwidth extension of speech using perceptual criteria", Ph. D. thesis, Arizona state university, December 2007.

[62] K. Brandenburg, O. Kunz, A. Sugiyama, "MPEG-4 natural audio coding", *Signal Processing: Image Communication*, *Elsevier Science*, pp. 423-444, 2000.

[63] B. Edler, "Speech Coding in MPEG-4", *International Journal of Speech Technology 2*, Kluwer Academic Publishers, pp. 289-303, 1999.

[64] M. Nishiguchi, A. Inoue, Y. Maeda and J. Matsumoto "Parametric Speech Coding - HVXC AT 2.0-4.0 KBPS," *IEEE Workshop on Speech Coding* , pp. 84-*86*, 1999.

[65] ITU-T Recommendation G.729.1, "An 8-32 kbit/s scalable wideband coder bitstream interoperable with G.729", *ITU-T Publication*, May 2006.

[66] W.C. Chu, "Embedded quantization of line spectral frequencies using a multistage tree-structured vector quantizer", *IEEE Trans. on Audio, Speech and Language Processing*, Vol. 14, No. 4, pp. 1205-1217, July 2006.

[67] P. Fiche, V. Ricordel, C. Labit, "Etude d'algorithmes de quantification vectorielle arborescente pour la compression d'images fixes", *Rapport de recherche*, INRIA, 1994.

[68] R.-Y. Wang, E. A. Riskin, and R. Ladner, “Codebook organization to enhance maximum a posteriori detection of progressive transmission of vector quantized images over noisy channels”, *IEEE ICASSP*, Vol. 5, pp. 233-236, 1993.

[69] N. Phamdo and N. Farvardin, "Coding of Speech LSP Parameters Using TSVQ with Interblock Noiseless Coding," *IEEE ICASSP*, pp. 193-196, Albuquerque, April 1990.

[70] W.-Y. Chan and A. Gersho, "High Fidelity Audio Transform Coding with Vector Quantization," *IEEE ICASSP*, pp. 1109-1112, Albuquerque, April 1990.

[71] W.-Y.Chan and A. Gersho, "Constrained-Storage vector Quantization in High Fidelity Audio Transform Coding," *IEEE ICASSP*, pp. 3597- 3600, Toronto, May 1991.

[72] D. Chemla, S.-W. Soong and W.-Y. Chan, “Tree-Structured Vector Quantization of Speech LSF Parameters", *IEEE Workshop on Speech Coding for Telecommunications*, pp. 71-72, Oct. 1993.

[73] W. Chan and D. Chemla, “Low-Complexity Encoding of Speech LSF Parameters Using Constrained-Storage TSVQ”, *IEEE ICASSP*, pp. 521-524, April 1994.

[74] K. Bao, and X. Xia, “Image compression using a new discrete multi-wavelet transform and a new embedded vector quantization”, *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 10, No. 6, pp. 833-842, Sep. 2000.

[75] H. Jafarkhani and N. Farvardin, “A scalable wavelet image coding scheme using multi-stage pruned tree-structured vector quantization,” in *Proc. IEEE Int. Conf. Image Processing*, Vol. 02, pp. 81–84, Oct. 1995.

[76] E. A. Riskin and R. M. Gray, “A greedy tree growing algorithm for the design of variable rate vector quantizers,” *IEEE Trans. Signal Proc.*, Vol. 39, pp. 2500-2507, Nov. 1991.

[77] P. A. Chou, T. Lookabaugh, and R. M. Gray, “Optimal Pruning with Applications to Tree-Structured Source Coding and Modeling”, *IEEE Trans. on Information Theory*, Vol. 35, No. 2, March 1989.

[78] I. Katsavounidis, C. Kuo, and Z. Zhang, “Fast Tree-Structured Nearest Neighbor Encoding for Vector Quantization”, *IEEE Transactions on Image Processing*, Vol. 5, No.2, pp.398-404, February 1996.

[79] N. Moayeri, D. L. Neuhoff, and W. E. Stark, “Fine-coarse vector quantization,” *IEEE Trans. Signal Processing*, vol. 39, no. 7, pp. 1503-1515, July 1991.

[80] U. Bayazit and W. A. Pearlman, “Variable-length constrained-storage tree-structured vector quantization,” *IEEE Trans. Image Processing*, Vol. 8, pp. 321–331, Mar. 1999.

[81] S.-B. Yang, “Variable-Branch Tree-Structured Vector Quantization” *IEEE Trans. Image Processing*, Vol. 13, No. 9, pp. 1275- 1285, Sep. 2004

[82] E. Cammarota and G. Poggi, “Address vector quantization with topology-preserving codebook ordering” *13rh GRETSI Symp.*, pp. 853-856, Sept. 1991

[83] E. A. Riskin, L. E. Atlas, and S.-R. Lay, “Ordered neural maps and their application to data compression,” *IEEE Workshop Neural Networks Signal Process.*, pp. 543-551, 1991.

[84] E. L. Lawler**,** *Combinatorial Optimization: Networks and Matroids.* New York: Holt, Rinehart and Winston, 1976.

[85] M. Djamah and D. O'Shaughnessy, "An efficient tree-structured codebook design for embedded vector quantization*", IEEE ICASSP*, pp. 4686-4689, March 2010.

[86] B. Bhattacharya, W. P. LeBlanc, S. Mahmoud, and V. Cuperman, "Tree searched multi-stage vector quantization of LPC parameters for 4 kb/s speech coding*," IEEE ICASSP*, 1992.

[87] W. C. Chu, "A fast search technique for multistage vector quantization based on the introduction of tree-structure to each stage", *IEEE Signals, Systems and Computers*, Vol. 1, pp. 425-429, Nov. 2004.

[88] M. Djamah and D. O'Shaughnessy, "Low-complexity encoding of speech LSF parameters using multistage tree-structured vector quantization: application to the MELP coder", *IEEE Canadian Conf. on Electrical and Computer Eng.*, pp. 376-380, 2009.

[89] M. Djamah and D. O'Shaughnessy "A fast tree-structured search procedure for multistage vector quantization of LSF parameters", *International Conference Signal and Image Processing*, Honolulu Hawaii, USA, pp. 42-46, August 2009.

[90] C.-Da Bei; Gray, R.; "An Improvement of the minimum distortion encoding Algorithm for Vector Quantization", *IEEE Trans. on Comm.*.Vol. 33. p. 1132-1133, Oct. 1985.

[91] J. Ngwa-Ndifor, and T. Ellis, "Predictive partial search algorithm for vector quantization," *IEE Electronic Letters*, Vol. 27, No. 19, pp.1722-1723, Sep. 1991.

[92] L. Torres and J. Huguet, **"**An improvement on codebook search for vector quantization," *IEEE Trans. Commun.*, Vol. 42, No. 2-4, pp. 208-210, 1994.

[93] S. W. Ra and J. K. Kim, "A fast mean-distance-ordered partial codebook search algorithm for image vector quantization", *IEEE Trans. Circuits Syst.II*, Vol. 40, No. 9, pp. 576–579, Sept. 1993.

[94] S. Baek, B. Jeon, and K.-M. Sung "A Fast Encoding Algorithm for Vector Quantization" *IEEE signal Processing Letters*, Vol. 4, No. 12, pp. 325-327, Dec. 1997

[95] Y. Bai, and C. Bao, "A Fast Search Approach for LSF Parameters Codebook", *IEEE ICASSP*, pp. I-841-I-844, 2006.

[96] S. E. Lyshevski, *Engineering and Scientific Computations Using MATLAB*, A John Wiley & Sons, Inc., 2003.

[97] J. Ferguson, B. Patterson, J. Beres, P. Boutquin, and M. Gupta, *C# Bible*, Wiley Publishing, Inc, Indianapolis, Indiana, 2002.

[98] ITU-T Recommendation G.191, "ITU-T Software Tool Library" , *ITU-T Publication*, August 2005

[99] ITU-T Recommendation P.862, "Perceptual Evaluation of Speech Quality (PESQ), an Objective Method for End-to-End Speech Quality Assessment of Narrow-Band Telephone Networks and Speech Codecs", Feb. 2001 (Amendment 2 November 2005).

[100] DARPA TIMIT, "Acoustic-Phonetic Continuous Speech Corpus*", Nat. Inst. Standards and Technology, Gaitherburg, MD,* 1993.

[101] ITU-T Recommendation P.830, "Subjective performance assessment of telephone-Band and Wideband digital codecs", *ITU-T Publication,* 1696.

[102] ITU-T Recommendation P.56, "Objective Measurement of Active Speech Level", *ITU-T Publication,* March 1993.

[103] http://www.uakron.edu/its/learning/training/Office2003.php

[104] I. Griffiths, M. Adams, and J. Liberty, *Programming C# 4.0*, O'Reilly Media, 2010.

[105] D. Syme, A. Granicz, and A. Cisternino, *Expert F#*, Apress, 2007.

[106] ITU-T Study Group 12, "Subjective Test Plan for Characterization of an 8 kbit/s Speech Codec", Document SQ-46.95R3 (Issue 2), Sept. 1995.

[107] M. Djamah and D. O'Shaughnessy "Fine-Granular Scalable MELP Coder Based on Embedded Vector Quantization", *10th Annual Conference of the International Speech Communication Association (Interspeech),* Brighton, UK, pp 2603-2606, Sept. 2009.

[108] D.F. Lyons, D.L. Neuhoff, and D. Hui, "Reduced storage tree-structured vector quantization", *IEEE ICASSP*, Vol. 5, pp. 602 -605, April 1993.

[109] J. Rouat, Y. C. Liu, and D. Morissette, "A pitch determination and voiced/unvoiced decision algorithm for noisy speech", *Elsevier Speech Communication,* vol. 21, pp. 191-207, 1997.

[110] P. Kroon and B. S. Atal, "Pitch predictors with high temporal resolution," *IEEE ICASSP*, pp. 661-664, April 1990.

[111] P. Kabal and R.P. Ramachandran, "Pitch prediction filters in speech coding", *IEEE Trans. Acoustics, Speech, Signal Processing*, vol. 37, pp. 467-478, Apr. 1989.

[112] F.K. Soong, B.-H. Juang, "Line Spectrum Pair (LSP) and Speech Data Compression", *IEEE ICASSP*, pp. 1.10.1-1.10.4, March 1984.