République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Ecole Nationale Polytechnique

Département d'électronique

Mémoire de fin d'études en vue de l'obtention
du diplôme d'ingénieur d'état en électronique

Thème :

Compression multicanaux quasi sans perte de signaux électroencéphalographiques

Etudié par :
TALEB-HACINE Mikhaïl

Président du jury :
Dr. R. SADOUN

Examinateur :
Dr. M. ADNANE

Rapporteurs :
Mme A. BOUSBIA-SALAH (USTHB)
Dr. H. BOUSBIA-SALAH

Promotion Juillet 2011
E.N.P- 10, Avenue Hassen Badi, 16200 El Harrach, Alger

ملخص:
تعد إشارات الخطيط الدماغي ذات أهمية كبيرة في معالجة الإضطرابات الدماغية وفي الأبحاث المتمركزة حول علوم الإستيعاب. نظرا لطول التسجيلات والحجم الكبير التي تشغله على وسائل الخزن الرقمية فلا بد من ضغط هاته الإشارات.
نقترح من خلال هذا العمل خوارزمية ضغط إشارات الخطيط الدماغي تعتمد على تحويل كارهونان لوف لحذف الإطناب الحيزي وعلى تحويل المويجات على طريقة المصعد (ليفتينغ) لحذف الإطناب الزمني. وأخيرا، نعتمد مرمز هوفمان لحذف ما تبقى من الإطناب دون الإساءة إلى المعلومات.
بإجراء تجارب عديدة على برنامج ماتلاب، نبين أن الخوارزمية المقترحة تمتاز بنسبة ضغط أعلى من جميع برامج الضغط الشاملة التي تم مقارنتها بها، وذلك بالمحافظة على المعلومات.
الكلمات المفاتيح: الخطيط الدماغي، ضغط المعلومات، المويجات، تحويل كارهونان لوف، مرمز هوفمان.

Résumé :
L'examen électroencéphalographique revêt une très grande importance dans le diagnostic de pathologies cérébrales et dans la recherche en sciences cognitives. Etant donné le volume important des enregistrements sur supports numériques, il est nécessaire de faire appel à des programmes de compression performants.
Nous proposons dans ce document un algorithme de compression basé sur la transformation de Karhunen-Loève, le lifting d'ondelettes et le codage entropique de Huffman. Nous montrons grâce à des tests effectués sous MATLAB® que l'algorithme proposé est plus performant que tous ceux auxquels il est comparé.
Mots-clés : EEG, électroencéphalogramme, compression quasi sans perte, ondelettes, lifting, KLT, codage de Huffman.

Abstract :
Electroencephalogram has a major role in diagnostic of cerebral disorders as well as in modern researches in cognitive sciences. One major issue we face with electroencephalograms is the amount of data which need to be stored on drives. Thus, efficient compression programs are needed.
We propose in this work a compression algorithm based on Karhunen-Loève transform, wavelet lifting and Huffman entropy coding.
By means of tests on MATLAB®, we show that this scheme overtakes all the algorithms it had been compared to.
Keywords : EEG, data compression, wavelets, lifting, KLT, Huffman coding.

Remerciements

Je tiens à exprimer toute ma gratitude à mes promoteurs, Mme et M. BOUSBIA-SALAH pour leurs orientations pertinentes, la disponibilité et l'écoute permanente dont ils auront fait preuve tout au long de ce travail.

Je tiens également à remercier les membres du jury pour l'intérêt accordé à notre travail.

Enfin, je remercie tous ceux qui auront contribué à ce travail et à mon cursus universitaire.

Dédicaces

Je dédicace ce travail à :
Ma très chère famille,
Mes chers amis,
Aux membres de la famille de l'Ecole Polytechnique.

Table des matières

Introduction générale

L'examen électroencéphalographique revêt aujourd'hui une très grande importance aux côtés des techniques d'imagerie cérébrale dont il est un complément indispensable. En effet, que ce soit dans la diagnostic de pathologies (épilepsies par exemple) ou dans la recherche en sciences neurocognitives, l'électroencéphalogramme constitue une indispensable source d'informations.

Cet examen produit néanmoins de longs enregistrements sur de nombreux canaux avec une redondance élevée. La transmission et le stockage de ces données exigent donc le recours à une compression sans pertes, ou quasi sans pertes c.-à-d. induisant une distorsion très minime.

Le présent document propose un schéma de compression sans pertes des données EEG multicanaux. Nous avons obtenu cet algorithme (figure 1) en tenant compte des spécificités du signal EEG et en nous basant sur différents travaux et publications relatifs à la compression de données au sens large.

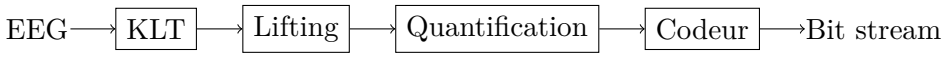


FIGURE 1 – Algorithme proposé

Le premier chapitre fournit une introduction à la constitution du système nerveux en général et du cortex cérébral en particulier avant d'étudier l'électroencéphalographie à proprement parler et ses applications.

Dans le second chapitre, nous étudions l'analyse en composantes principales qui nous permet ici d'éliminer la redondance spatiale existant entre les différents canaux de notre signal.

Le chapitre qui suit est consacré aux ondelettes auxquelles nous nous

intéressons tout particulièrement. Elles interviennent dans la réduction de la redondance temporelle présente au niveau de chaque canal. Toujours dans l'étude des ondelettes, nous nous concentrons sur une méthode très efficace d'application de ces dernières : le lifting.

Dans le quatrième chapitre, nous présentons et comparons deux techniques de codage de source sans pertes afin d'éliminer la redondance persistante après les deux précédentes étapes de l'algorithme.

Le cinquième chapitre clôt la partie théorique de notre travail en présentant les résultats obtenus avec notre algorithme et en fournissant quelques pistes à priori intéressantes pour son amélioration.

Nous passons ensuite à l'étude des caractéristiques du DSP TMS320C6713 de Texas Instruments. Enfin, nous proposons une implémentation sur ce DSP du lifting d'ondelettes, élément central de notre algorithme de compression. A terme, c'est l'ensemble de l'algorithme de compression qui doit être implémenté sur DSP.

Chapitre 1

L'électroencéphalographie

1.1 Introduction

Dans ce chapitre, nous allons d'abord décrire très brièvement la constitution et le fonctionnement du système nerveux chez l'être humain [2, 25]. Nous aborderons ensuite plus en détail la constitution physiologique et le rôle du cerveau, et en particulier du cortex cérébral [1, 4, 8, 14, 22]. En rapport avec ces précédentes descriptions, nous présenterons le principe de l'électroencéphalographie [5, 7, 17] et ses diverses applications en neurosciences [3, 9].

1.2 Système nerveux chez l'être humain

1.2.1 Constitution générale

Le système nerveux consiste en un ensemble d'organes constitués de cellules spécialisées appelées neurones, assurant la coordination des actions et différentes fonctions.

Le système nerveux se décompose en plusieurs sous-ensembles [2] :

Système nerveux central Il comprend :

1. L'encéphale entouré par le crâne ;
2. La moelle épinière entourée par les vertèbres.

Différents fluides et tissus isolent ses parties constitutives.

Système nerveux périphérique Le système nerveux périphérique a pour rôle de connecter le système central à d'autres organes du corps ; il ne se compose que de nerfs.
Les dendrites et autres axones sont entourés de myéline et assurent une

transmission très rapide. Les corps cellulaires appartiennent au système central ou à des ganglions nerveux (jouant le rôle de relais).

On distingue les chemins afférents (depuis le corps vers le système nerveux central) et les chemins efférents (depuis le système nerveux central vers les éléments moteurs et les glandes).

Système nerveux somatique Il comprend les nerfs contrôlant les muscles et les récepteurs sensoriels.

Système nerveux autonome ou neuro-végétatif Il est constitué de neurones moteurs contrôlant les muscles du cœur et les muscles lisses d'autres organes internes tels que les intestins.
Il comprend 2 sous-systèmes ayant des rôles opposés :

1. Système nerveux sympathique : il intervient dans les situations dites « Fear, fight and flight »(littéralement : craindre, se battre et s'envoler) correspondant à un stress intense ; il augmente la pression artérielle, accélère le rythme cardiaque et la respiration ;
2. Système nerveux parasympathique : il joue un rôle inverse à celui du système nerveux sympathique. Il a pour mission de conserver l'énergie en abaissant la fréquence cardiaque, la pression artérielle et la fréquence respiratoire. Il a également pour rôle de restaurer l'énergie en facilitant la digestion.

1.2.2 L'encéphale

Constitution

L'encéphale est illustré par la figure 1.1 [14].
L'encéphale comprend [2] :

1. Cerveau (Cerebrum) : c'est la partie la plus volumineuse de l'encéphale. Il est constitué de deux hémisphères, droit et gauche, connectés via un ensemble de nerfs formant le corps calleux (corpus callosum).
   Le cerveau assure la coordination des données en provenance des récepteurs sensoriels et les fonctions motrices ; il est le siège de l'intelligence, de l'apprentissage et de la mémoire.
   Il est recouvert d'une fine matière grise (de 1 à 4 mm d'épaisseur) : le cortex cérébral ;
2. Cervelet (Cerebellum) : il assure l'équilibre et la coordination des muscles ;
3. Bulbe rachidien (Medulla Oblongata) : il joue un rôle dans la régulation des battements du cœur, respiration et vasoconstriction (pression

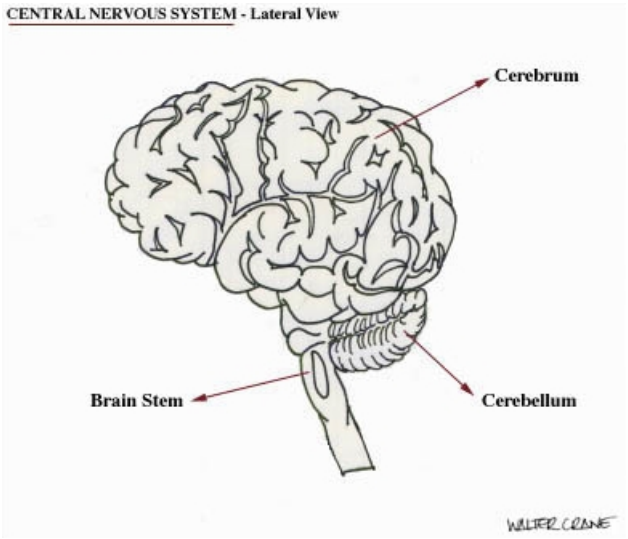


FIGURE 1.1 – Cerveau et cervelet - Reproduit avec autorisation

artérielle) ainsi que certains reflexes : toux, éternuement, avalement, hoquet, etc ;

4. Diencéphale : il comprend :
   – Hypothalamus : il présente plusieurs aires, chacune étant responsable de la régulation d'une fonction ou d'une sensation : faim, soif, température corporelle, etc. Il relie le système nerveux au système endocrinien ;
   – Thalamus : il constitue un relai central pour les influx nerveux entrants.

1.3 Le neurone

C'est l'unité fonctionnelle de base du système nerveux ; le seul cerveau humain en recèle environ 100 milliards.

1.3.1 Constitution du neurone

Elle est commune à tous les neurones (figure 1.2, [25]) , même si différents types de ces derniers existent [25, 2] :

1. Corps cellulaire ou soma : il partage la constitution des cellules eucaryotes, c.-à-d. que son cytoplasme contient entre autres un noyau et différents organites tels que les mitochondries (pour la transformation de la matière organique en énergie) ;

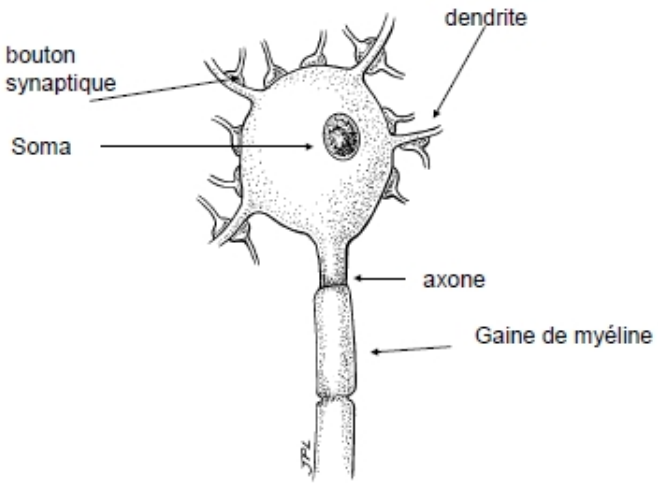


FIGURE 1.2 – Cellule neuronale - Reproduit avec autorisation

2. Axone : constitué de fibres longues, il conduit l'influx nerveux depuis le corps de la cellule ;
3. Arbre dendritique : composé de dendrites (fibres courtes et très connectées) qui reçoivent des messages depuis d'autres cellules et les transmettent vers le soma.

1.3.2 Classification fonctionnelle

On peut classer les neurones suivant la fonction qu'ils assurent en trois grandes catégories [2] :

1. Neurones sensoriels : transmettent l'information depuis les différents récepteurs sensoriels vers le système nerveux central. Ils ont de longues dendrites et un axone court ;
2. Neurones moteurs : transmettent l'information depuis le système nerveux central vers les muscles et glandes. Ils ont un long axone et de courtes dendrites ;
3. Interneurones : propres au système nerveux central, ils traitent les informations en provenance des neurones sensoriels et relaient l'information vers les neurones moteurs.

1.4 Transmission de l'influx nerveux

On distingue deux types de transmission de l'influx nerveux : transmission chimique [2, 25] et transmission électrique [25].

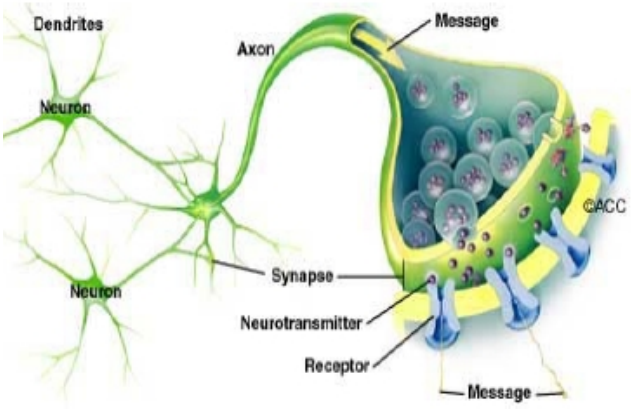


FIGURE 1.3 – Transmission chimique - Reproduit avec autorisation

1.4.1 Transmission chimique : potentiel d'action

La transmission de l'influx nerveux est effectuée par des agents chimiques (neurotransmetteurs) via des synapses dits chimiques. Son principe général est illustré à la figure 1.3 [25].

Synapse C'est une région de contact entre deux neurones ou entre un neurone et une cellule effectrice (muscle ou glande).
La cellule émettrice est dite pré-synaptique, la cellule réceptrice est quant à elle dite post-synaptique, les deux étant séparées sur une distance allant de 20 à 40 nm par l'espace intersynaptique (ou fente synaptique).

Potentiel de repos Au repos, il y a une différence de concentration d'ions des deux côtés de la membrane plasmique de la cellule neuronale : la charge est positive à l'extérieur (cations de sodium) tandis qu'elle est négative à l'intérieur (anions de potassium).
Cette distribution résulte en un potentiel dit de repos d'une valeur approximative de - 65 mV.
La pompe de sodium-potassium maintient cet état et réalise un transport actif (grâce à des protéines et à la consommation d'énergie sous forme d'ATP [1]), allant à l'encontre du gradient de concentration.

Potentiel d'action Le potentiel d'action est une inversion de la polarité le long de la membrane plasmique durant quelques millisecondes suite à une

1. Adénosine triphosphate

stimulation de la cellule nerveuse.
La genèse a lieu au niveau du cône d'émergence (à l'origine de l'axone, contient un grand nombre de canaux). Ce cône fait uns synthèse de tous les potentiels aux niveaux des synapses dendrites/soma et génère un potentiel d'action qui se déroule comme suit :

1. Ouverture des canaux de sodium, entrée d'ions Na+ et génération du potentiel d'action ;
2. La dépolarisation suffit pour ouvrir tous les canaux ;
3. Ouverture des canaux de potassium, plus de gradient électrochimique ;
4. Ouverture des canaux de rectification du sodium, repolarisation progressive du neurone.

Le retour à l'état de repos et la repolarisation ont lieu grâce au travail de la pompe active évoquée plus haut.
La transmission du potentiel d'action autorise la libération de vésicules contenant des neurotransmetteurs dans la fente synaptique. Ces derniers se lient alors à l'élément post-synaptique au niveau de récepteurs prévus à cet effet et influent sur son potentiel, avec soit un effet inhibiteur PPSI (Potentiel PostSynaptique Inhibiteur) soit un effet excitateur PPSE (Potentiel PostSynaptique Excitateur), le premier inhibe la transmission de l'influx nerveux depuis la cellule post-synaptique vers d'autres cellules comme son nom l'indique, le second ayant un effet opposé.

1.4.2 Transmission électrique

Ce type de transmission est assuré par la circulation d'ions et de molécules. Il se distingue par sa rapidité en comparaison avec la transmission chimique.
Les cellules présynaptique et postsynaptique sont reliées via des jonctions communicantes. Ces jonctions consistent en des molécules protidiques appelées connexines regroupées par 6 et formant un canal cylindrique appelé connexon, ce dernier relie les cytoplasmes des 2 cellules (figure 1.4, [25]).

Cette jonction permet le passage bi-directionnel de molécules d'un diamètre inférieur à 1.5 à 2 nm. Il s'agit en particulier de seconds messagers jouant le rôle d'intermédiaires et qui sont produits en réponse à des messagers primaires (hormones, neurotransmetteurs).

1.5 Le cortex cérébral

Les deux hémisphères du cerveau sont composés de matière blanche et de matière grise.

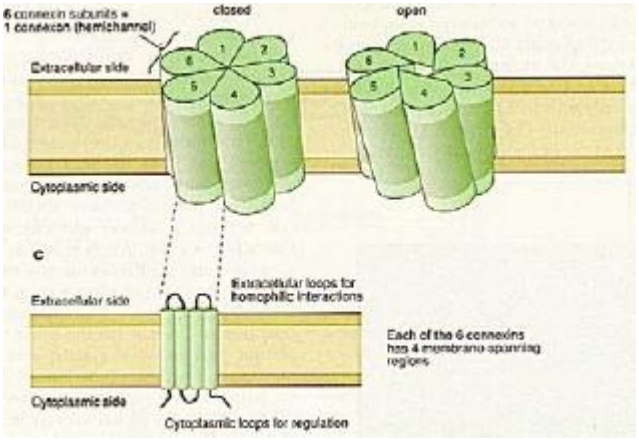


FIGURE 1.4 – Connexon - Transmission électrique - Reproduit avec autorisation

1.5.1 Définition

Le cortex est la substance grise qui recouvre l'ensemble du cerveau. D'une épaisseur allant de 1 à 4 mm, il contient environ 10 milliards de neurones et se divise en 4 lobes [14] :

1. Lobe occipital : réception et traitement de l'information visuelle ;
2. Lobe temporal : reçoit les signaux auditifs ; il est également en charge du traitement du langage et du sens des mots ;
3. Lobe pariétal : il traite les informations issues du toucher, goût, la douleur, le chaud et le froid ;
4. Lobe frontal : il assure les fonctions motrices, la parole et la pensée.

Les lobes sont illustrés à la figure 1.5 [14].

Bien qu'il y ait une interdépendance fonctionnelle entre tous ces lobes, les fonctions majeures remplies par chacun d'eux ont été déterminées. La classification dite de Brodmann associe à 52 aires spécifiques du cortex une tâche donnée. Cette classification est illustrée à la figure 1.6 [1].

Il existe trois catégories de cortex classées, entre autres critères, du point de vue phylogénétique [4], c.-à-d. de l'évolution des espèces :

1. Archicortex : le plus ancien ;
2. Paléocortex : ancien ;
3. Néocortex : le plus récent, responsable de la perception, de l'apprentissage et de l'intelligence.

A noter que seul le néocortex nous intéresse pour la suite de notre étude.

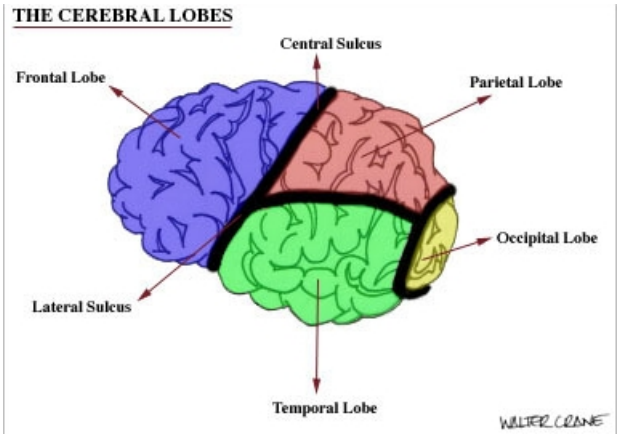


FIGURE 1.5 – Lobes du cerveau - Reproduit avec autorisation

Matière blanche

La matière blanche consiste en un ensemble de fibres (axones) à travers lesquelles le cortex est relié à d'autres zones du cerveau. On distingue 3 types de fibres [4] :

1. Fibres de projection : elles connectent le cortex avec des centres sous-corticaux (le thalamus par exemple) ;
2. Fibres commissurales : elles connectent les cortexes des deux hémisphères et forment le corps calleux ;
3. Fibres d'association : elles connectent différentes zones du cortex d'un même hémisphère.

1.5.2 Cellules du cortex

On retrouve deux types de cellules dans le cortex : les cellules pyramidales et les cellules granulaires.

Cellule pyramidale Elles ont un diamètre (inf à 30 μm) et sont nommées ainsi du fait de la forme triangulaire (cônique) de leur corps cellulaire (soma). Elles ont aussi pour particularité d'avoir deux arbres dendritiques distincts [22] :

- Arbre de dendrites apicales : les dendrites s'étendent vers les couches supérieures et sont généralement parallèles les unes aux autres ;
- Arbre de dendrites basales : il s'étend latéralement au même niveau que le soma.

Ces cellules sont prédominantes dan le cortex cérébral.

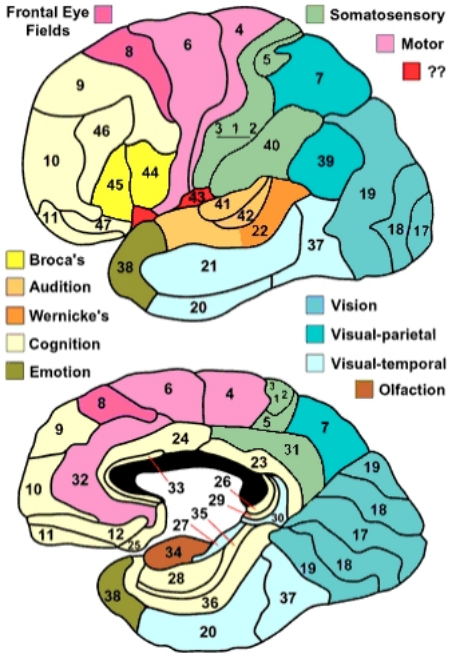


FIGURE 1.6 – Aires de Brodmann - Reproduit avec autorisation

Cellule granulaire Elles sont petites et de forme ronde, (inf à 10 μm de diamètre) elles servent d'interneurones, reçoivent des informations de fibres corticales afférentes et se connectent aux neurones de sortie pyramidaux.

1.5.3 Structure histologique du néocortex

On entend par là structure du tissu, mais également les relations fonctionnelles immanentes à cette structure [8].
Le néocortex cérébral est composé de 6 couches superposées horizontalement et interconnectées [8] dont les épaisseurs varient suivant la région du cerveau considérée :

1. Couche moléculaire (molecular layer) : réceptrice, contient des cellules de Cajal ;
2. Couche granulaire externe : réceptrice ;
3. Couche pyramidale externe : effectrice, cellules pyramidales moyennes et petites ;
4. Couche granulaire interne : réceptrice, interneurones ;
5. Couche pyramidale interne ou ganglionnaire : effectrice, cellules pyramidales grandes ;
6. Couche de cellules polymorphes (forme variable) : effectrice.

Cette structure est illustrée par la figure 1.7 [4]. « layer »est traduit par couche, les mots « inner, outer »sont traduits par interne et externe.

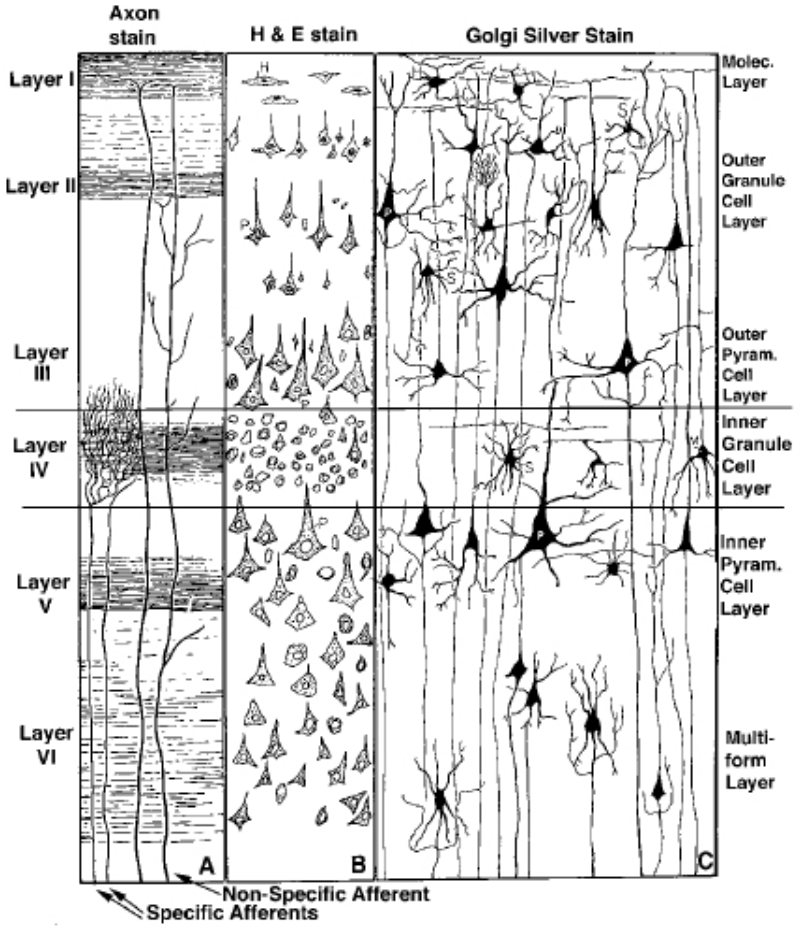


FIGURE 1.7 – Structure histologique du néocortex par différentes méthodes de coloration - Reproduit avec autorisation

La couche 2 reçoit des informations d'autres zones corticales du même hémisphère, tandis que la couche 4 reçoit des informations de zones sous-corticales. La première émet des axones directement vers la couche 1 tandis que la seconde émet des axones vers les couches 3 et 5.
Ces deux dernières, composées de cellules pyramidales, nous intéressent tout particulièrement. En effet, les dendrites de ces cellules sont associées aux axones issus des grains de la couche 2 au niveau des cellules de Cajal de la couche moléculaire 1. [8, 4]
C'est l'activité électrique de ces dendrites qui est mesurée par l'EEG et qui nous intéresse.

A noter :
- La couche 3 émet également des fibres associatives vers les couches 2 et 5 d'autres zone du même hémisphère ;
- La couche 5 émet des axones vers les zones les centres sous-corticaux ;
- La couche 6 émet des fibres commissurales vers l'hémisphère opposé.

1.5.4 Rythmes cérébraux

L'analyse de l'activité électrique du cerveau permet d'observer des signaux périodiques contenus dans des bandes de fréquences données et correspondant à des états et des zones du cerveau déterminés. Il s'agit des rythmes cérébraux [5].

Citons les rythmes cérébraux et leurs caractéristiques :

1. Rythme alpha : ondes de 8 à 12 Hz, caractéristique d'un adulte en veille diffuse (éveillé avec les yeux fermés), les plus fortes amplitudes sont enregistrées au niveau des zones occipitales et pariétales. Ces amplitudes diminuent en général lorsque le sujet ouvre les yeux ;
2. Rythme bêta : ondes de 13 à 30 Hz, localisées au niveau des zones frontales. Elles correspondent tout autant à une activité normale ou un effort mental intense qu'à la phase du sommeil paradoxal ;
3. Rythme gamma : à 40 Hz, correspondent à une forte sollicitation du cerveau et à des processus cognitifs tels que la perception, le langage ou les émotions ;
4. Rythme delta : au-dessous de 4 Hz, ondes caractéristiques du sommeil profond ;
5. Rythme thêta : de 4 à 7 Hz. Localisé dans les zones temporales, il est lui aussi caractéristique du sommeil profond. Il tend d'ailleurs à apparaître de concert avec la diminution de l'amplitude des ondes alpha.

1.6 Electroencéphalogramme

1.6.1 Définition

L'EEG ou électroencéphalogramme est une mesure des potentiels électriques au niveau du scalp générés par la circulation de courants électriques au niveau du cerveau. Ces potentiels reflètent son activité électrique et donc son activité physiologique.

Historique

Le biologiste britannique Richard Caton (1842-1926) est le premier à avoir observé en 1875 la présence de courants électriques chez le singe et le

lapin, et ce, en relevant des oscillations à l'aide d'un galvanomètre et d'électrodes au niveau du cortex ou à la surface du crâne.

Hans Berger (1873-1941) appliquera cette méthode en 1924 à l'homme et obtiendra le premier relevé EEG de l'histoire. En 1934 allait être introduite la visualisation sur papier du tracé grâce à l'inscription à jet d'encre.

Si les principes de base de l'examen sont restés les mêmes depuis, le traitement des données, la qualité de la mesure et l'exploitation des informations ont grandement bénéficié des progrés constants de l'informatique (traitement par ordinateurs), de l'électronique (progrès des circuits électroniques tels que les amplificateurs à faible bruit) et des neurosciences.

1.6.2 Source du signal EEG

Les neurones pyramidaux du néocortex apportent la plus forte contribution au signal EEG via le flux de courant le long de leurs dendrites apicales (orientées normalement à la surface du scalp) et appelé courant primaire [7].

Le courant primaire mis en jeu au niveau d'une dendrite étant extrêmement faible, le signal n'est mesurable que si des milliers de cellules pyramidales sont activés de manière synchrone.

Ces courants primaires induisent une distribution du potentiel au niveau macroscopique, le champ électrique résultant engendre des courants secondaires qui circulent dans le milieu conducteur entourant les dendrites.

Toute la difficulté consiste à localiser la source des courants primaires et donc les zones responsables d'une activité cérébrale donnée à partir des données mesurées : ceci est connu comme étant le problème inverse. Ce dernier n'ayant pas de solution unique.

Il est également à noter que d'autres sources d'activité électrique existent :

- Potentiel d'action le long des axones : peuvent être modélisées par des quadripôles, les deux courants électriques s'annulent mutuellement, leur contribution est donc faible ;
- Flux de courant à travers la fente synaptique entre neurones et dendrites : lui aussi contribue très peu au signal EEG. La fente ayant de très faibles dimensions, le flux de courant est donc très faible lui aussi. De plus, leur disposition aléatoire autour du neurone a pour conséquence l'annulation d'une grande partie de ces courants deux à deux.

1.6.3 Montage EEG

Disposition des électrodes

Les électrodes sont disposées sur le scalp en accord avec la norme internationale 10/20, celle-ci déterminant comme suit les positions : la distance interélectrode doit représenter 10 ou 20 % de la distance nasion-inion. Cette norme vise assurer un enregistrement optimal l'activité électrique de l'ensemble des lobes
Le nombre d'électrodes peut varier de 8 à 21. A noter qu'un nombre plus important d'électrodes peut-être requis pour des examens particuliers [17].
La figure 1.8 [13] illustre un montage aux normes, les électrodes sont représentées par des lettres :

- **C** : centrale ;
- **A** : pour les électrodes positionnées sur le lobe de l'oreille ;
- **P** : pour pariétale ;
- **F** : pour frontale ;
- **O** : pour occipitale ;
- **Fp** : pour frontale-polaire ;
- **T** : pour temporal.

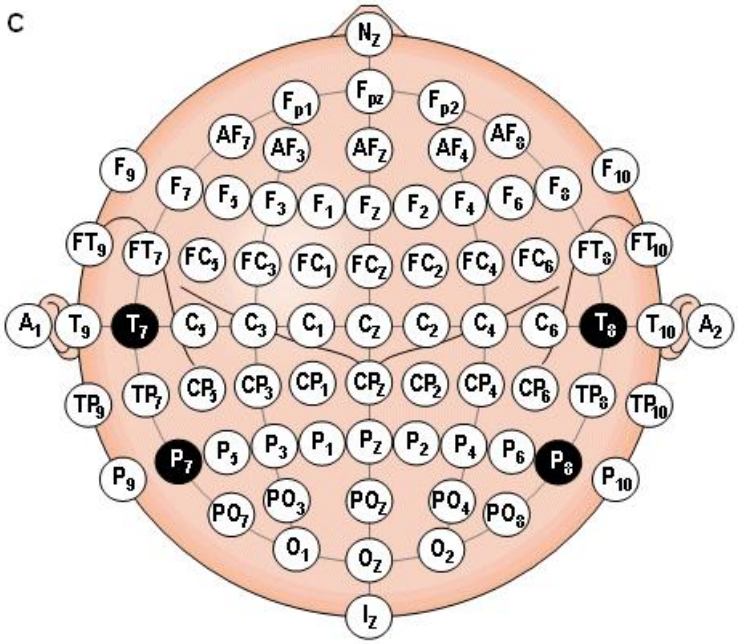


FIGURE 1.8 – Système 10/20

L'électrode placée sur le scalp enregistre l'activité électrique juste sous son emplacement ; une superficie de l'ordre du cm^2 correspondant approximativement à l'activité de 100000 neurones et peut-être modélisée par un

dipôle.

Méthodes d'enregistrement

Il existe deux méthodes d'enregistrement suivant le montage choisi :

1. Montage monopolaire : le potentiel de chaque électrode est comparé à celui d'une électrode de référence. Théoriquement, l'électrode de référence est censée avoir un potentiel nul. On distingue trois options :
   - Electrode céphalique : en général sur le menton ou le nez, ou même deux électrodes au niveau des mastoïdes ou des oreilles ;
   - Montage extracéphalique : la référence est sterno-épineuse ;
   - Référence moyenne de Wilson : ses électrodes sont reliées entre elles à travers de grandes impédances, la moyenne des potentiels servant de référence.
2. Montage bipolaire : les potentiels des électrodes sont comparés deux à deux, de proche en proche, longitudinalement (d'avant en arrière), transversalement (de droite à gauche), ou en décrivant un zigzag. Une électrode de référence est également nécessaire suivant les options citées dans le cas du montage monopolaire.

Types d'électrodes

Il existe 3 types d'électrodes couramment employées en électroencéphalographie, le choix d'un type plutôt que les autres est dicté par des considérations cliniques :

1. Electrodes conventionnelles : ce sont des tiges d'argent qui présentent à leur extrémité un tampon imbibé d'une solution saline ;
2. Electrodes aiguilles : comme leur nom l'indique, c'est une aiguille qui est piquée dans le scalp. Elles ont l'avantage d'assurer un contact électrique de qualité mais rendent l'examen invasif, ce qui n'est pas le cas en règle générale ;
3. Electrodes cupules : elles sont remplies d'une pâte conductrice, qui assure également l'adhésion de l'aiguille sur le scalp.

1.6.4 Avantages et inconvénients

L'électroencéphalogramme est devenu un procédé standard en neurosciences. Par rapport à d'autres examens, il présente des avantages mais également des limitations.

Avantages par rapport à d'autres techniques

- Résolution temporelle de l'ordre de la milliseconde, elle est en générale de l'ordre de la seconde pour les techniques d'imagerie ;

– Examen non-invasif c.-à-d. aucun passage à travers les tissus cutanés n'est nécessaire (sauf dans le cas des électrodes aiguille) ;
– Coût contenu.

Limitations par rapport à d'autres techniques

1. La résolution spatiale est assez limitée et l'examen EEG doit donc souvent être accompagné d'examens d'imagerie médicale tels que l'IRM (Imagerie à Résonance Magnétique) ou la TEMP (Tomographie d'Emission Mono-Photonique) ;
2. L'amplitude des signaux à mesurer est faible, le rapport signal/bruit en est donc d'autant plus bas. C'est un inconvénient majeur de l'EEG.

1.7 Applications de l'EEG

On peut distinguer trois grandes catégories d'application de l'électroencéphalographie : les diagnostics médicaux, le neurofeedback et la recherche en sciences cognitives.

1. Diagnostics médicaux : le recours à l'électroencéphalographie est nécessaire dans bon nombre de maux et de situations d'urgence [3]. L'interprétation des signaux EEG est délicate et requiert une spécialisation, ainsi qu'une prise en compte de facteurs cliniques.

   L'EEG intervient directement dans les cas suivants :

   (a) Potentiels évoqués (PE) : ce sont des relevés EEG obtenus suite à une stimulation particulière, le but étant d'examiner la réaction de larges ensembles de neurones chez le sujet suivant la nature de stimulation [19].
   Il existe 3 types de PE suivant la nature de la stimulation :
   – PE visuel : provoqué par des flashs lumineux ou des tracés tels qu'un damier où les couleurs seraient inversées alternativement. Ce type d'examens permet de déceler certaines lésions des voies optiques ;
   – PE auditif : provoqué par des notes musicales ou des déclics. Ce type de PE peut indiquer la sortie du coma d'un sujet ou des déficiences auditives chez l'enfant ;
   – PE somesthésiques : obtenus par application de courants électriques de courte durée aux nerfs moteurs et sensoriels périphériques. Ce type d'EP intervient dans le diagnostic d'anomalies au niveau de la matière blanche et d'éventuels traumatismes de la moelle épinière pendant ou en dehors des opérations chirurgicales.

(b) Epilepsies : via l'enregistrement de tracés entre les crises (tracé intercritique) ou pendant une crise (tracé critique), il est possbiles de détecter des anomalies lentes ou des anomalies épileptiformes, signalées par de fortes pointes et de fortes décharges électriques au niveau du cerveau. A titre d'exemple, une activité rapide et l'apparition de rythmes rapides et aigus peuvent indiquer une épilepsie ;

(c) Diagnostic d'une mort cérébrale ;

(d) Troubles du sommeil (apnée, privation de sommeil chronique, narcolepsie, cataplexie) : l'analyse des ondes caractéristiques des différentes phases du sommeil (sommeil lent léger ou profond, sommeil paradoxal) peut apporter de précieuses informations ;

(e) Surveillance des enfants prématurés : les rythmes cérébraux se forment au fur et à mesure de la maturation du cerveau, d'où l'intérêt de détecter d'éventuelles anomalies chez les enfants prématurés ;

(f) Traitement en urgence des victimes de traumatismes crâniens : un ralentissement et des ondes lentes postérieures dans le relevé EEG peuvent indiquer la présence de commotions cérébrales.
L'EEG aide également à la localisation d'hématomes intercrâniens dans les cas de l'altération de la conscience et la détection d'états de mal non convulsifs pour les personnes présentant des syndromes confusionnels (baisse de la vigilance, perturbations des activités intellectuelles, désorientation).
L'EEG peut également constituer une mesure de la profondeur d'un coma post-traumatique ;

2. Neurofeedback (Biofeedback EEG ou neurothérapie) :

C'est une discipline particulière du biofeedback qui est un processus permettant à un humain de modifier son activité physiologique (rythme cardiaque, respiration, température corporelle).
Les ondes cérébrales sont traitées par des algorithmes spécifiques produisant un feedback qui est renvoyé au sujet sous une représentation graphique (la plupart du temps un jeu vidéo temps-réel auquel le sujet joue), le tout sous le contrôle d'un praticien qui fixe l'objectif à atteindre. Le sujet apprend ainsi graduellement à réguler par lui-même son activité cérébrale ; la capacité d'apprentissage en est améliorée. Concrètement, le neurofeedback effectue une modulation de l'activité cérébrale en agissant sur les mécanismes d'excitation et d'inhibition d'ensembles de neurones.
Des applications existent pour le traitement de l'épilpesie, l'autisme, le déficit d'attention, ou les stress post-traumatique, et le domaine est actuellement en plein essor [9].

3. Sciences de la cognition : les stimulations sont organisées en paradigme, le sujet devant exécuter différentes tâches. Le plus fréquemment utilisé étant le paradigme Oddball : deux stimuli sont distribués de manière pseudo-aléatoire dans une excitation. Par exemple, deux notes musicales différentes, l'une plus fréquente que l'autre. De tels tests apportent de précieuses informations sur la prise de décision, l'attention et la mémoire.

1.8 Conclusion

Ce chapitre nous a permis de comprendre en profondeur les mécanismes biologiques à l'origine du signal électroencéphalographique. Par la suite, l'étude de ses applications nous a permis de mesurer toute l'importance de cet examen.
Dans le chapitre prochain, nous abordons la première étape de l'algorithme de compression : l'analyse en composantes principales via la transformation de Karhunen Loève.

Chapitre 2

Analyse en composantes principales : la KLT

2.1 Introduction

L'analyse en composantes principales est une méthode d'analyse des données multidimensionnelles : à partir d'un ensemble de variables aléatoires corrélées et à l'aide de transformations, nous obtenons un nouvel ensemble de variables (en nombre inférieur ou égal) orthogonales et décorrélées, dites composantes principales [10]. Nous faisons ici appel à la KLT (Karhunen-Loève Transform).

2.2 Transformation de Karhunen-Loève

2.2.1 Définition

Cette transformation consiste à diagonaliser la matrice de covariance d'un ensemble de variables aléatoires (ici les signaux électriques relevés par les électrodes) constituant un processus, ceci revenant à décorréler totalement les unes des autres nos variables aléatoires.
La KLT est une transformation optimale d'un point de vue théorique mais ne va pas sans poser quelques difficultés d'implémentation.

2.2.2 Formalisme

Soit un processus stochastique à N variables aléatoires défini par un vecteur x comme suit :

$$x = [x[0], \ldots, x[N-1]]^T \tag{2.1}$$

Soit à présent Σ_x la matrice de covariance du signal x, ϕ_k ses vecteurs propres et λ_k les valeurs propres associées à chacun d'entre eux tel que :

$$\Sigma_x \phi_k = \lambda_k \phi_k \tag{2.2}$$

Sachant que la matrice de covariance est symétrique, ces vecteurs propres sont orthogonaux, et nous pouvons donc construire une matrice carrée $N \times N$:

$$\Phi = [\phi_0, \ldots, \phi_{N-1}] \tag{2.3}$$

satisfaisant

$$\Phi^T \Phi = I \text{ c.-à-d. } \Phi^{-1} = \Phi^T \tag{2.4}$$

nous obtenons

$$\begin{bmatrix} \ldots & \ldots & \ldots \\ \ldots & \sigma_{ij} & \ldots \\ \ldots & \ldots & \ldots \end{bmatrix} [\phi_0, \ldots, \phi_{N-1}] = [\phi_0, \ldots, \phi_{N-1}] \begin{bmatrix} \lambda_0 & 0 & \ldots & 0 \\ 0 & \lambda_1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \ldots & 0 & \lambda_{N-1} \end{bmatrix} \tag{2.5}$$

et :

$$\Phi^T \Sigma_x \Phi = \Phi^{-1} \Sigma_x \Phi = \Lambda \tag{2.6}$$

où :

$$\Lambda = diag(\lambda_0, \ldots, \lambda_{N-1}) \tag{2.7}$$

Nous obtenons finalement l'expression de la projection de notre signal x dans l'espace dont la base est formée par les vecteurs propres de la matrice de covariance :

$$y = \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix} = \Phi^T x = \begin{bmatrix} \phi_0^T \\ \vdots \\ \phi_{N-1}^T \end{bmatrix} x \tag{2.8}$$

L'opération inverse est donnée par la relation :

$$x = \Phi y = [\phi_0, \ldots, \phi_{N-1}] \begin{bmatrix} y_0 \\ \vdots \\ y_{N-1} \end{bmatrix} = \sum_{i=0}^{N-1} y_i \phi_i \tag{2.9}$$

2.2.3 Propriétés de la KLT

La transformation de Karhunen-Loève a deux propriétés fondamentales qui nous seront utiles en compression :

Décorrélation des signaux

Cette propriété est très utile en compression sans pertes, en effet la matrice de covariance du signal après projection est donnée par :

$$\Sigma_y = \Phi^T \Sigma_x \Phi = \Lambda = \begin{bmatrix} \lambda_0 & 0 & \dots & 0 \\ 0 & \lambda_1 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & \lambda_{N-1} \end{bmatrix} = \begin{bmatrix} {\sigma_0}^2 & 0 & \dots & 0 \\ 0 & {\sigma_1}^2 & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \dots & 0 & {\sigma_{N-1}}^2 \end{bmatrix} \tag{2.10}$$

Tous les éléments de la matrice de covariance à l'exception de ceux de la diagonale étant nuls après transformation, les intercorrélations entre tous les canaux le sont donc également : la redondance entre eux a été éliminée grâce à la KLT, d'où son intérêt en compression des données.
On notera également que les variances de chaque canal, égales aux valeurs propres restent inchangées avant et après projection ; l'énergie étant préservée, cette propriété trouve tout son intérêt dans le cas d'une compression sans pertes.

Redistribution de l'énergie

Si nous nous plaçons dans le cas général d'une transformation orthogonale quelconque :

$$A = [a_0, \dots, a_{N-1}] \tag{2.11}$$

$$y = A^T x \tag{2.12}$$

et :

$$x = Ay \tag{2.13}$$

Si nous prenons m composantes sur les N composantes du signal et mesurons l'énergie associée :

$$S_m(A) = \sum_{i=0}^{m-1} E\left[(y_i - \mu_{y_i})^2\right] = \sum_{i=0}^{m-1} {\sigma_{y_i}}^2 \tag{2.14}$$

On voit que l'énergie $S_m(A)$ contenue dans les m premières composantes dépend de la matrice de transformation A. On peut démontrer grâce à la méthode du multiplicateur de Lagrange [26] que la transformation qui maximise $S_m(A)$ est la KLT.

Ainsi, nous choisissons les m composantes correspondant aux valeurs les plus élevées de l'énergie (c.-à-d. les variances ou valeurs propres les plus élevées) :

$$\Phi_m = [\phi_0, \ldots, \phi_{m-1}] \tag{2.15}$$

la transformation est effectuée comme suit :

$$y = \Phi_m^T x \tag{2.16}$$

quant à la reconstruction :

$$x = \Phi_m y \tag{2.17}$$

On voit l'importance de cette propriété en compression avec pertes : lorsque la dimensionalité est réduite, une partie de l'énergie est perdue, mais peut-être négligeable si les vecteurs propres retenus correspondent à des valeurs propres élevées.

2.2.4 KLT adaptative

La KLT dépend du signal analysé, la matrice de covariance doit donc être calculée pour un nombre fini d'échantillons des signaux, et donc périodiquement. Cette technique est couramment employée dans la compression multi-canaux de signaux audio par exemple [28].

2.2.5 Application au signal EEG

Nous disposons d'un signal EEG issu de la base de données Physiobank [6], sur 22 électrodes et échantillonné à 256 Hz.

Un résultat typique de ceux obtenus pour le calcul de la matrice de covariance est illustré à la figure 2.1.

Les résultats montrent que les canaux 1,2,5,9,13 correspondant respectivement aux électrodes FP1-F7, F7-T7, FP1-F3, FP2-F4 et FP2-F8 (voir figure 1.8) sont particulièrement corrélés entre eux. Suivant l'architecture retenue, il pourrait être difficile de calculer les valeurs propres d'une matrice 22x22 tout en assurant la stabilité. Une solution potentielle et sous-optimale

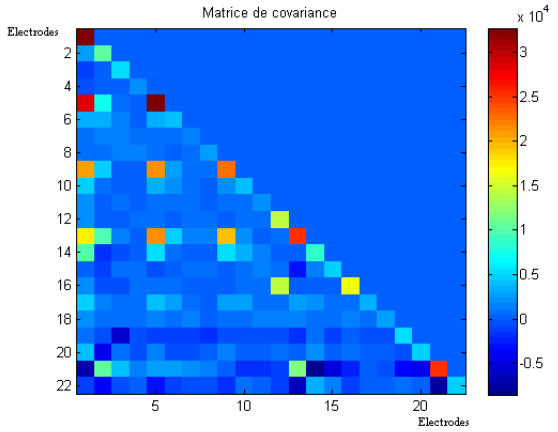


FIGURE 2.1 – Covariance d'un enregistrement

pour la compression serait de regrouper ces électrodes dans une seule et même matrice pour éliminer la redondance élevée entre elles.

La figure 2.2 illustre encore davantage la forte redondance spatiale entre canaux :

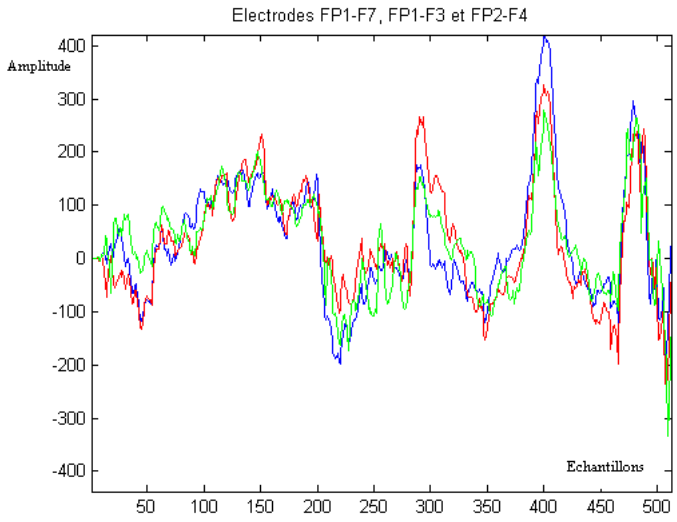


FIGURE 2.2 – Signaux fortement corrélés

2.3 Conclusion

A travers cette étude de la KLT, nous avons perçu tout son intérêt dans l'élimination de la redondance existant entre les électrodes d'un même enregistrement et donc son importance en tant que première étape de l'algorithme de compression multicanaux.
Dans le chapitre suivant, nous étudions très en détail les ondelettes qui interviennent sur les canaux précédemment décorrélés entre eux.

Chapitre 3

Ondelettes

3.1 Introduction

Les ondelettes peuvent être définies comme suit [15] :

> ***« Un outil récent de traitement du signal permettant l'analyse, à plusieurs échelles de temps, des propriétés locales de signaux complexes pouvant présenter des zones d'instationnarités. »***

Les ondelettes présentent de nombreuses applications dans divers domaines, celle qui nous intéresse ici étant bien entendu la compression de données appliquée aux signaux EEG.

3.1.1 Généralités

Principe d'Heisenberg-Gabor

Ce principe décrit par la formule suivante :

$$\sigma_t \sigma_\omega \geq \frac{1}{2} \tag{3.1}$$

fournit une limitation aux résolutions qu'il est possible d'obtenir simultanément en temps et fréquence (notées respectivement σ_t et σ_ω) pour la représentation d'un signal.

STFT

La STFT (Short Time Fourier Transform ou transformée de Fourier à court terme) correspond à une projection du signal sur un espace de fenêtres translatées et modulées :

$$STFT(x) = X(u,\omega) = \int_{-\infty}^{+\infty} x(t)h(t-u)\exp^{-jwt}\,dt \tag{3.2}$$

La STFT permet de représenter le signal dans le plan temps-fréquence. Néanmoins, cette représentation est très redondante d'une part, et d'autre part, elle ne permet pas d'obtenir la résolution maximale donnée par le principe d'Heisenberg. De plus, elle ne présente pas d'intérêt particulier en compression de données. Nous donnons néanmoins sa définition pour mieux aborder la représentation temps-échelle offerte par les ondelettes.

Définition

Une ondelette est une fonction à supports temporel et fréquentiel limités, respectivement σ_t et σ_ω. Ces supports correspondent aux variances d'une densité de probabilité et représentent les dimensions d'un atome (rectangle) du plan temps-fréquence ; σ_t et σ_ω obéissent donc au principe d'incertitude d'Heisenberg :

L'ensemble des atomes recouvrant l'espace est obtenu par dilatation et translation de l'ondelette dite mère :

$$\psi_{u,s}(t) = \frac{1}{\sqrt{s}}\psi\left(\frac{t-u}{s}\right) \tag{3.3}$$

Cette opération revient à multiplier le support temporel par s et à diviser le support fréquentiel par s. Conformément au principe d'Heisenberg, augmenter la résolution temporelle revient à diminuer la résolution fréquentielle et vice versa.

Condition d'admissibilité

Une fonction est appelée ondelette si elle satisfait la condition suivante dite d'admissibilité :

$$\int_{\mathbb{R}^+} \frac{|\hat{\psi}(\omega)|^2}{|\omega|}d\omega = \int_{\mathbb{R}^-} \frac{|\hat{\psi}(\omega)|^2}{|\omega|}d\omega < +\infty \tag{3.4}$$

Une condition suffisante peut également être énoncée :

$$\int_{\mathbb{R}} \psi(t)dt = 0 \tag{3.5}$$

3.1.2 Transformée continue en ondelettes

Elle est donnée par la formule suivante :

$$Wf(u,s) = \int_{-\infty}^{+\infty} f(t)\frac{1}{\sqrt{s}}\psi^*(\frac{t-u}{s})dt \tag{3.6}$$

C'est une représentation très redondante du signal. Nous étudions un peu plus loin dans ce document comment construire une base d'ondelettes orthogonales ou biorthogonales.
A noter qu'il est nécessaire d'adjoindre une fonction dite d'échelle, et notée φ pour compléter l'analyse du signal : la fonction φ fournit une approximation locale tandis que l'ondelette ψ traite les approximations autour de cette approximation locale [15].

Les spectrogrammes calculés à l'aide d'une transformée de Fourier à court terme (STFT) de ces deux fonctions dans le cas d'une ondelette 'db3' sont illustrés aux figures 3.1 et 3.2.

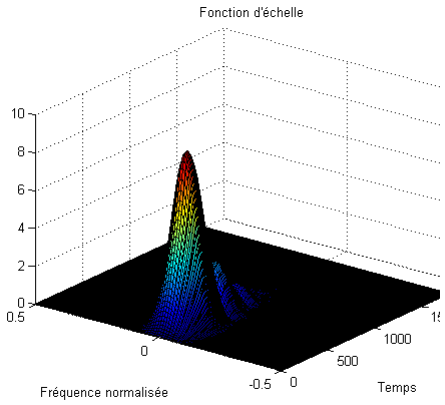


FIGURE 3.1 – Spectrogramme φ

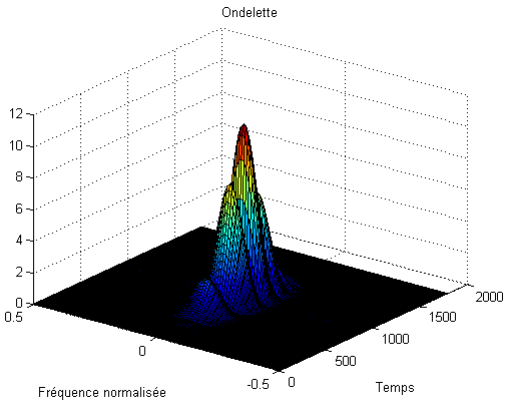


FIGURE 3.2 – Spectrogramme ψ

3.2 Ondelettes et régularité d'un signal

Nous allons à présent étudier la transformée en ondelettes du point de vue de la régularité du signal. Nous en déduirons une propriété extrêmement importante en compression des signaux.

3.2.1 Régularité Lipschitzienne

Soit une fonction $f(t)$ qui est m fois différentiable, et soit

$$p_v(t) = \sum_{k=0}^{n-1} \frac{f^{(k)(v)}}{k!}(t-v)^k \tag{3.7}$$

son développement en série de Taylor.

Cette fonction est dite Lipschitz α en un point v si :

$$\forall t \in \mathbb{R} \quad |f(t) - p_v(t) = \epsilon_v(t)| \leq K\,|t-v|^\alpha \tag{3.8}$$

On dit que cette fonction est uniformément Lipschitz sur un intervalle si la précédente condition est vérifiée en tout point v de ce domaine. Dans le domaine fréquentiel, cette propriété est assurée si la condition suivante est réalisée :

$$\int_{-\infty}^{+\infty} \left|\hat{f}(\omega)\right| (1+|\omega|^\alpha) d\omega < +\infty \tag{3.9}$$

Cela équivaut à dire que la fonction $f(t)$ décroit rapidement aux hautes fréquences pour α donné.

3.2.2 Ondelette et moments nuls

Une ondelette est dite à décroissance rapide si elle satisfait la condition :

$$\forall t \in \mathbb{R},\ \exists C_m \text{tq}\ \ |\psi(t)| \leq \frac{C_m}{1+|t|^m} \tag{3.10}$$

Cette ondelette a n moments nuls si et seulement si elle dérive d'une fonction θ à décroissance rapide :

$$\psi(t) = (-1)^n \frac{d^n\theta(t)}{dt^n} \tag{3.11}$$

Cette ondelette satisfait alors à l'équation suivante :

$$\int_{-\infty}^{+\infty} t^k \psi(t) dt = 0,\ 0 \leq k < n \tag{3.12}$$

c.-à-d. qu'elle est orthogonale aux polynômes de degré $n-1$ avec les résultats importants suivants :

$$Wf(u,s) = W\epsilon_v(u,s) \tag{3.13}$$

$$Wf(u,s) = s^n \frac{d^n}{du^n}(f \star \bar{\theta}_s)(u) \tag{3.14}$$

$$\bar{\theta}_s(t) = \frac{1}{\sqrt{s}}\theta(\frac{-t}{s}) \tag{3.15}$$

On dit dans ce cas que cette transformée en ondelettes correspond à un opérateur différentiel multiéchelle : la régularité d'une fonction peut-être décrite grâce à une transformée en ondelettes, des théorèmes liant la régularité aux coeffcients de la transformation.
Dans notre cas, l'intérêt réside dans la propriété qu'a la transformée en ondelettes de donner des coefficients de faible amplitude ou nuls dans les portions du signal régulières pour peu que l'ondelette ait un nombre de moments nuls suffisant. En compression, cela permet de réduire le nombre de coefficients à coder en effectuant un seuillage des coefficients quasi-nuls ou à l'aide d'un quantificateur scalaire à zone centrale.

3.2.3 Etude d'un exemple

Ci-dessous (figure 3.3), l'exemple de l'analyse d'un enregistrement EEG avec une transformée continue en ondelettes effectuée sur MATLAB. Dans le premier cas, nous faisons appel à une ondelette dite de Daubechies à un seul moment nul, dans le second à une ondelette de Daubechies à 3 moments nuls.

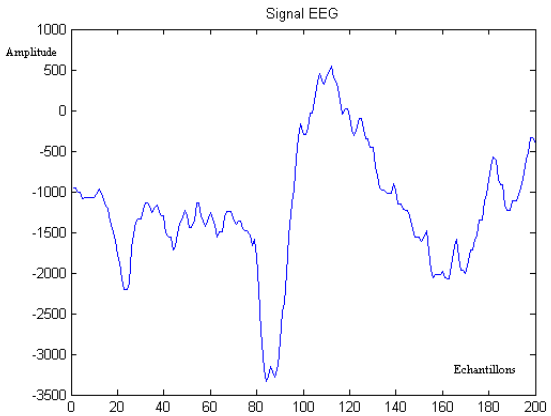


FIGURE 3.3 – Signal EEG

Le signal étudié présente deux transitions rapides localisées entre les échantillons 80 et 100. Les figures 3.4 et 3.5 illustrent le point étudié plus

haut : l'ondelette à 3 moments nuls donne des coefficients de moindre amplitude aux échelles fines pour la seconde singularité.

En revanche, les coefficients ont une amplitude très proche dans les deux cas pour la première singularité : la régularité du signal EEG n'étant pas uniforme (valeurs de α différentes pour les deux singularités et donc degrés du développement de Taylor différents), l'augmentation du nombre de moments nuls n'apporte pas systématiquement un gain sensible.

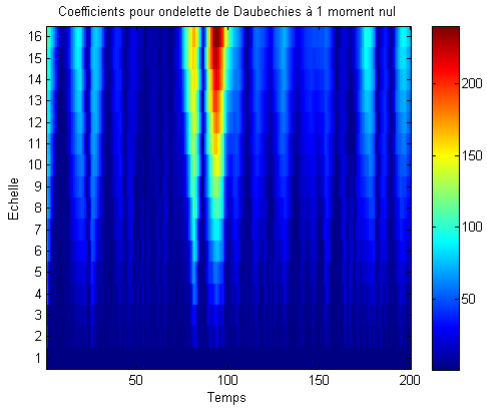


FIGURE 3.4 – CWT avec db1

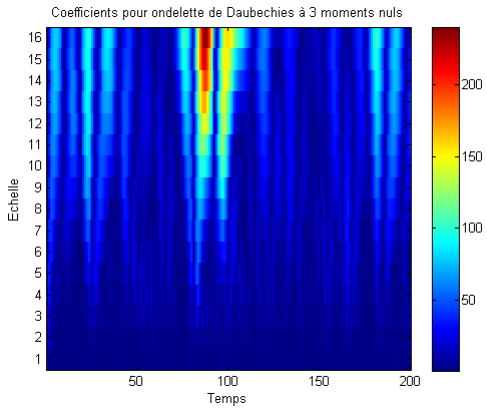


FIGURE 3.5 – CWT avec db3

Conclusion L'augmentation du nombre de moments nuls ne réduit pas systématiquement l'amplitude des coefficients pour l'ensemble du relevé EEG. Comme on le verra plus tard, augmenter le nombre de moments nuls a un

coût en termes de calculs à effectuer et il convient donc de trouver un compromis entre le gain en compression offert sur l'ensemble des données traitées et la complexité des calculs.

3.3 Bancs de filtres

Avant d'étudier les transformées en ondelettes discrètes et orthogonales, étudions les bancs de filtres. La figure 3.6 illustre justement un banc de filtres ou décomposition en sous-bandes d'un signal x : le signal est décomposé grâce aux filtres passe-bas $\tilde{h}(z^{-1})$ et passe-haut $\tilde{g}(z^{-1})$ avant sous-échantillonnage.

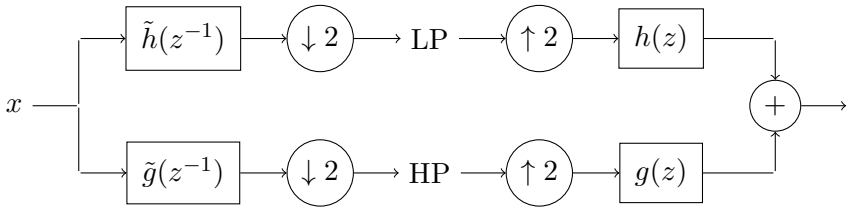


FIGURE 3.6 – Décomposition en sous-bandes par banc de filtres

A la réception, le signal est interpolé par ajout d'un zéro entre chaque paire d'échantillons, avant application des filtres de synthèse.

La reconstruction du signal est parfaite si les conditions suivantes, dites conditions de biorthogonalité du banc de filtres sont respectées :

$$h(z)\tilde{h}(z^{-1}) + g(z)\tilde{g}(z^{-1}) = 2 \tag{3.16}$$

$$h(z)\tilde{h}(-z^{-1}) + g(z)\tilde{g}(-z^{-1}) = 0 \tag{3.17}$$

où ces expressions sont obtenues grâce aux propriétés de la transformée en Z. [26] et correspondent dans le domaine temporel aux équations suivantes :

$$\sum_k h[k]\tilde{h}[2n-k] = \delta[n] \tag{3.18}$$

$$\sum_k g[k]\tilde{g}[2n-k] = \delta[n] \tag{3.19}$$

$$\sum_k g[k]\tilde{h}[2n-k] = 0 \tag{3.20}$$

$$\sum_k h[k]\tilde{g}[2n-k] = 0 \tag{3.21}$$

A noter que dans le cas particulier d'une ondelette orthogonale (que nous étudions ci-après), nous avons :

$$h = \tilde{h} \tag{3.22}$$
$$g = \tilde{g} \tag{3.23}$$

3.4 Ondelettes orthogonales

3.4.1 Approximations multirésolutions

Une suite d'espaces V_j constitue une approximation multirésolution si elle vérifie certaines propriétés [11, 15] dont nous ne citerons que trois :

$$\forall (j,k) \in \mathbb{Z}^2, f(t) \in V_j \Leftrightarrow f(t - 2^j k) \in V_j \tag{3.24}$$
$$\forall j \in \mathbb{Z}, V_{j+1} \subset V_j \tag{3.25}$$
$$\forall j \in \mathbb{Z}, f(t) \in V_j \Leftrightarrow f\left(\frac{t}{2}\right) \in V_{j+1} \tag{3.26}$$

La première propriété est synonyme d'invariance par translation proportionnelle à 2^j, la seconde implique que toute approximation à une échelle donnée peut être obtenue à partir d'une approximation précédente à une échelle plus fine. La troisième enfin, signifie simplement qu'une fonction appartenant à l'espace V_j appartient après dilatation à l'espace d'approximations plus grossières V_{j+1}.

La fonction $f(t)$ est projetée dans une base de Riesz [11] :

$$f(t) = \sum_{n=-\infty}^{+\infty} a[n]\theta(t-n) \tag{3.27}$$
$$A\|f\|^2 \leq \sum_{n=-\infty}^{+\infty} |a[n]|^2 \leq B\|f\|^2 \tag{3.28}$$

Une base de Riesz est une famille de vecteurs linéairement indépendants mais non orthogonaux. Ces deux équations correspondent donc en réalité à une projection sur un frame qui est une représentation moins redondante qu'une transformée continue en ondelettes. La fonction θ doit obéir à la condition suivante :

$$\frac{1}{B} \leq \sum_{k=-\infty}^{+\infty} |\hat{\theta}(\omega - 2k\pi)|^2 \leq \frac{1}{A}, \forall \omega \in [-\pi, \pi] \tag{3.29}$$

Il est possible de construire une base orthonormée à partir de la fonction θ pour une approximation multirésolution :

$$\hat{\varphi}(\omega) = \frac{\hat{\theta}(\omega)}{\sqrt{\sum_{k=-\infty}^{+\infty} |\hat{\theta}(\omega - 2k\pi)|^2}} \tag{3.30}$$

La base obtenue est alors exprimée, à la résolution 2^j comme suit [11] :

$$\varphi_{j,n} = \frac{1}{\sqrt{2^j}} \varphi \left(\frac{t}{2^j} - n \right) \tag{3.31}$$

La projection du signal dans cette base n'est pas redondante contrairement au cas des frames et de la transformée continue en ondelettes : nous obtenons une transformation en ondelette discrète et orthogonale.

3.4.2 Filtres miroirs conjugués

Nous avons vu auparavant qu'il était possible d'extraire une approximation à une échelle donnée à partir d'une approximation plus fine ($V_{j+1} \subset V_j$) alors :

$$\frac{1}{\sqrt{2}} \varphi \left(\frac{t}{2} \right) = \sum_{n=-\infty}^{+\infty} h[n] \varphi(t - n) \tag{3.32}$$

$$\hat{\varphi}(2\omega) = \frac{1}{\sqrt{2}} \hat{h}(\omega) \hat{\varphi}(\omega) \tag{3.33}$$

$$h[n] = \left\langle \frac{1}{\sqrt{2}} \varphi \left(\frac{t}{2} \right), \varphi(t - n) \right\rangle \tag{3.34}$$

La fonction d'échelle dilatée $\varphi\left(\frac{t}{2}\right)$ peut donc s'exprimer dans la base formée par la suite des fonctions $\varphi(t - n)$ grâce à un filtre h. Ce résultat peut-être généralisé pour une dilatation par un facteur 2^P :

$$\hat{\varphi}(\omega) = \left(\prod_{p=1}^{P} \frac{\hat{h}(2^{-p}\omega)}{\sqrt{2}} \right) \hat{\varphi}(2^{-P}\omega) \tag{3.35}$$

Le filtre discret h doit respecter l'égalité suivante, (théorème de Mallat et Meyer [11]) :

$$\left| \hat{h}(\omega) \right|^2 + \left| \hat{h}(\omega + \pi) \right|^2 = 2 \tag{3.36}$$

On parle dans ce cas d'un filtre miroir conjugué.

3.4.3 Ondelettes orthogonales

Il est nécessaire de compléter le sous-espace V_j par un sous-espace W_j contenant les détails manquants afin de restituer le signal à l'échelle 2^{j-1} :

$$V_{j-1} = V_j \oplus W_j \tag{3.37}$$

$$P_{V_{j-1}} f = P_{V_j} f + P_{W_j} f \tag{3.38}$$

La suite de fonctions $\psi_{j,n} = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t}{2^j} - n\right)$ est construite en suivant la même démarche qu'auparavant :

$$\frac{1}{\sqrt{2}} \psi\left(\frac{t}{2}\right) = \sum_{n=-\infty}^{+\infty} g[n] \varphi(t-n) \tag{3.39}$$

$$g[n] = \left\langle \frac{1}{\sqrt{2}} \psi\left(\frac{t}{2}\right), \varphi(t-n) \right\rangle \tag{3.40}$$

Le filtre g est dit filtre miroir conjugué du filtre h. La décomposition de $f(t)$ à une échelle 2^j est donnée par les relations suivantes :

$$f(t) = \sum_{n=-\infty}^{+\infty} a_j[n] \varphi_{j,n} + \sum_{n=-\infty}^{+\infty} d_j[n] \psi_{j,n} \tag{3.41}$$

$$f(t) = \sum_{n=-\infty}^{+\infty} \langle f, \varphi_{j,n} \rangle \varphi_{j,n} + \sum_{n=-\infty}^{+\infty} \left\langle f, \psi_{j,n} \right\rangle \psi_{j,n} \tag{3.42}$$

3.4.4 Transformée rapide

Nous allons à présent voir l'implémentation pratique de ces transformées en ondelettes orthogonales grâce au théorème de Mallat. [15, 11].

Nous avons :

$$a_j[n] = \langle f, \varphi_{j,n} \rangle \tag{3.43}$$

$$d_j[n] = \left\langle f, \psi_{j,n} \right\rangle \tag{3.44}$$

Les projections sur les espaces V_{j+1} et W_{j+1} sont données respectivement par les équations suivantes :

$$a_{j+1}[p] = \sum_{n=-\infty}^{+\infty} h[n-2p] a_j[n] = a_j \star \bar{h}[2p] \tag{3.45}$$

$$d_{j+1}[p] = \sum_{n=-\infty}^{+\infty} g[n-2p] a_j[n] = a_j \star \bar{g}[2p] \tag{3.46}$$

La reconstruction de P_{V_j} à partir de $P_{V_{j+1}}$ et de $P_{W_{j+1}}$ est donnée par les relations qui suivent :

$$a_j[p] = \sum_{n=-\infty}^{+\infty} h[p-2n]a_{j+1}[n] + \sum_{n=-\infty}^{+\infty} g[p-2n]d_{j+1}[n] \tag{3.47}$$

$$a_j[p] = \check{a}_{j+1} \star h[n] + \check{d}_{j+1} \star g[n] \tag{3.48}$$

avec $\bar{x}[n] = x[-n]$ et $\check{x}[n]$ un signal nul pour les indices n impairs.

Cet algorithme est particulièrement simple et se caractérise par une complexité linéaire par rapport à la taille des données [15] : la décomposition en sous-bandes telle que donnée par le théorème de Mallat est illustrée à la figure 3.6 déjà vue auparavant dans la cas particulier des ondelettes orthogonales que nous venons d'étudier $h = \tilde{h}$ et $g = \tilde{g}$.

3.5 Ondelettes biorthogonales

3.5.1 Définition

Les ondelettes biorthogonales permettent d'améliorer la symétrie (pour éviter des problèmes de déphasage) et la régularité des ondelettes afin d'obtenir des signaux reconstruits réguliers. En effet, obtenir ces caractéristiques est plus aisé qu'avec des ondelettes orthogonales, cette caractéristique étant une forte contrainte [15].
A titre d'exemple, il est impossible d'obtenir symétrie et reconstruction parfaite pour une ondelette orthogonale [15, 11].

Le développement d'une fonction dans des bases biorthogonales ψ et $\tilde{\psi}$ (la seconde étant appelée base duale) est donné par la relation suivante :

$$f = \sum_{n,j=-\infty}^{+\infty} \left\langle f, \tilde{\psi}_{j,n} \right\rangle \psi_{j,n} \tag{3.49}$$

$$f = \sum_{n,j=-\infty}^{+\infty} \langle f, \psi_{j,n} \rangle \, \tilde{\psi}_{j,n} \tag{3.50}$$

sachant que la biorthogonalité des bases impose

$$\left\langle \psi_{j,n}, \tilde{\psi}_{j',n'} \right\rangle = \delta[n-n']\delta[j-j'] \tag{3.51}$$

Les fonctions $\varphi_{j,n}$ et $\tilde{\varphi}_{j,n}$ constituent les bases des sous-espaces V_j et $\tilde{V}_j$ respectivement. De même, les fonctions $\psi_{j,n}$ et $\tilde{\psi}_{j,n}$ forment les bases des espaces de détails W_j et $\tilde{W}_j$:

$$V_{j-1} = V_j \oplus W_j \tag{3.52}$$
$$\tilde{V}_{j-1} = \tilde{V}_j \oplus \tilde{W}_j \tag{3.53}$$

3.5.2 Transformée rapide

La transformée rapide en ondelettes biorthogonales correspond au banc de filtre illustré à la figure 3.6 vue auparavant avec le cas général où les filtres h et $\tilde{h}$ sont différents.

La construction d'ondelettes biorthogonales s'effectue de la même manière que pour les ondelettes orthogonales à partir de filtres et de la condition de reconstruction parfaite.

3.5.3 Ordonnancement des ondelettes

Il est préférable d'utiliser l'ondelette ayant le plus de moments nuls pour l'analyse, et de réserver l'ondelette symétrique et plus régulière à l'étape de reconstruction [11].

3.5.4 Ondelettes *bior*

Nous retenons les ondelettes LeGall 5/3 et Cohen-Daubechies-Feauvau 9/7, standards dans la norme de compression des images JPEG2000. La première est réservée à la compression avec ou sans pertes tandis que la seconde est réservée à la compression avec pertes. Elles correspondent en fait aux ondelettes « bior2.2 »et « bior4.4 »sous MATLAB (à un facteur multiplicatif près).

Ces ondelettes biorthogonales présentent les avantages suivants :
- Leur support est compact et il est minimal vu leur nombre de moments nuls : cette considération a prévalu dans leur adoption par la norme JPEG2000 [24] ;
- Elles sont symétriques ;

L'ondelette LeGall 5/3 présente l'inconvénient d'avoir des indices de frame assez élevées (elle est donc peu orthogonale). A contrario, les frames constituées par l'ondelette 9/7 sont beaucoup plus resserrées (elle est quasi orthonormale).
Différents critères de régularité (Critère de Sobolev, critère de Hölder) semblent également indiquer que l'ondelette 9/7 est un meilleur choix, au prix d'un support plus grand et donc de calculs plus élevés [24]. Il faut néanmoins tenir compte de la nature du signal EEG.

Les figures 3.7 à 3.9 illustrent les fonctions d'échelle, ondelettes et filtres associés (ainsi que leur spectre) à l'ondelette 'bior2.2' :

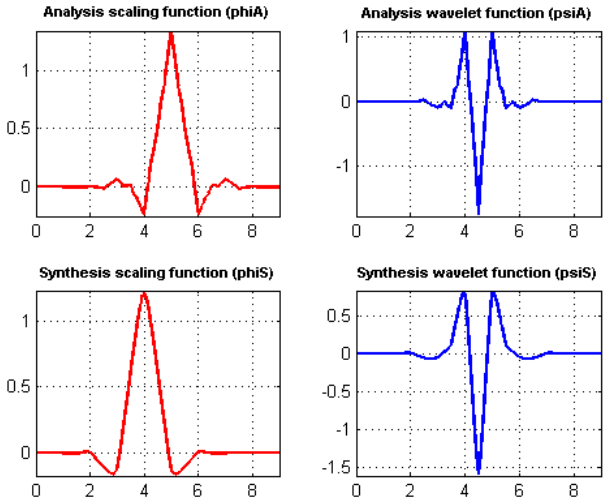


FIGURE 3.7 – Ondelette bior2.2 - Fonction d'échelle et ondelettes

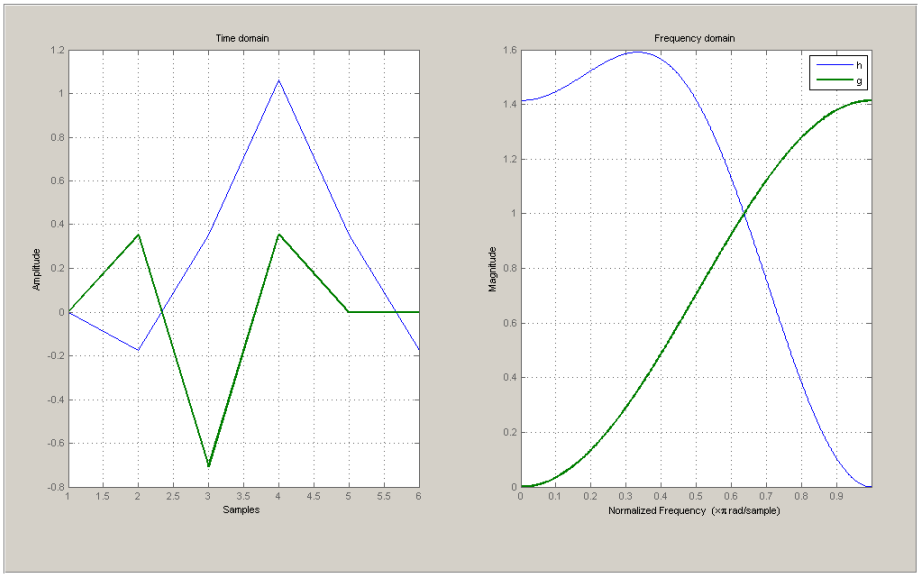


FIGURE 3.8 – Filtres h et g

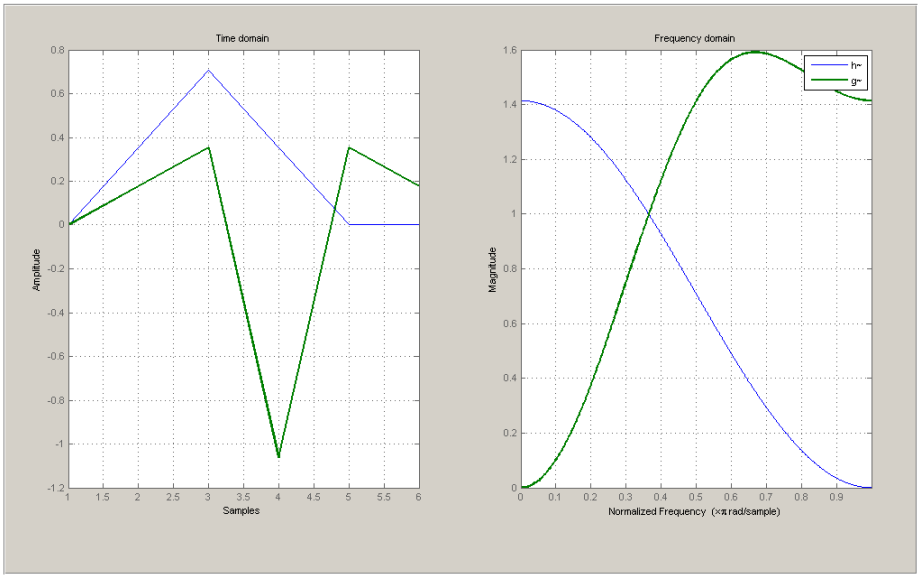


FIGURE 3.9 – Filtres $\tilde{h}$ et $\tilde{g}$

Les figures 3.10 à 3.12 illustrent les fonctions d'échelle, ondelettes et filtres associés (ainsi que leur spectre) à l'ondelette 'bior4.4' :

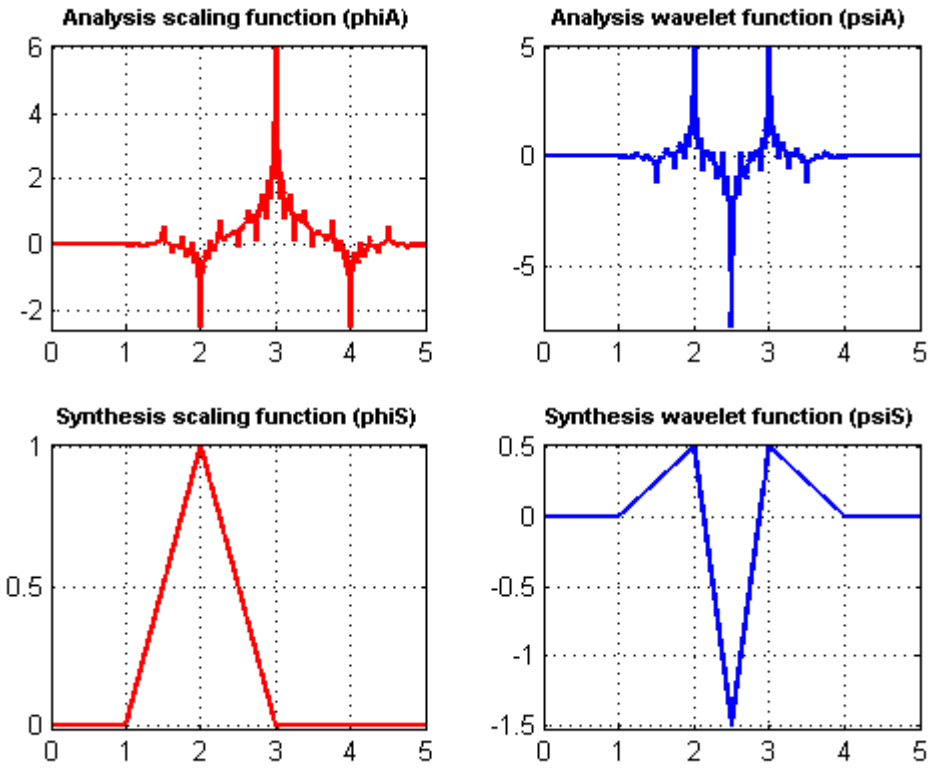


FIGURE 3.10 – Ondelette 9/7 - Fonction d'échelle et ondelettes

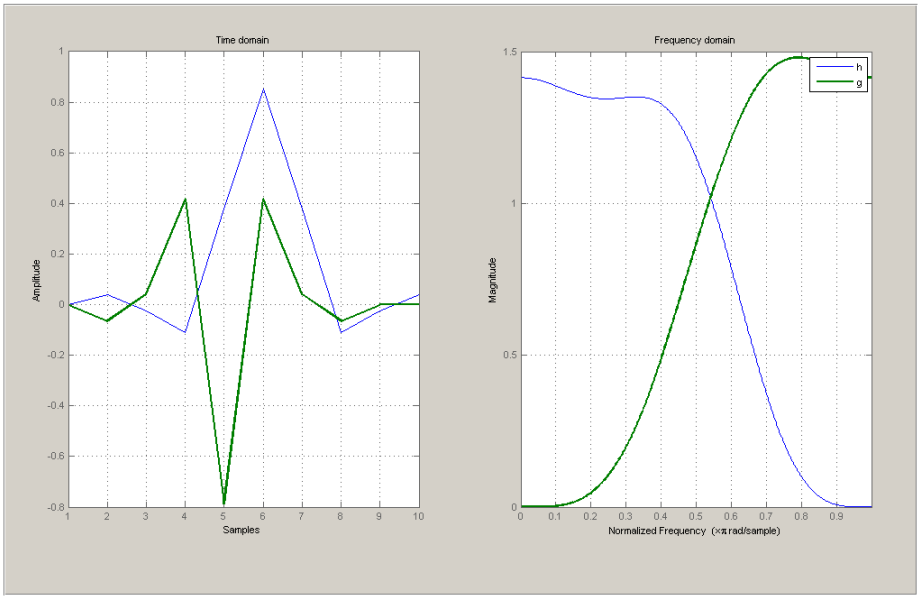


FIGURE 3.11 – Filtres h et g

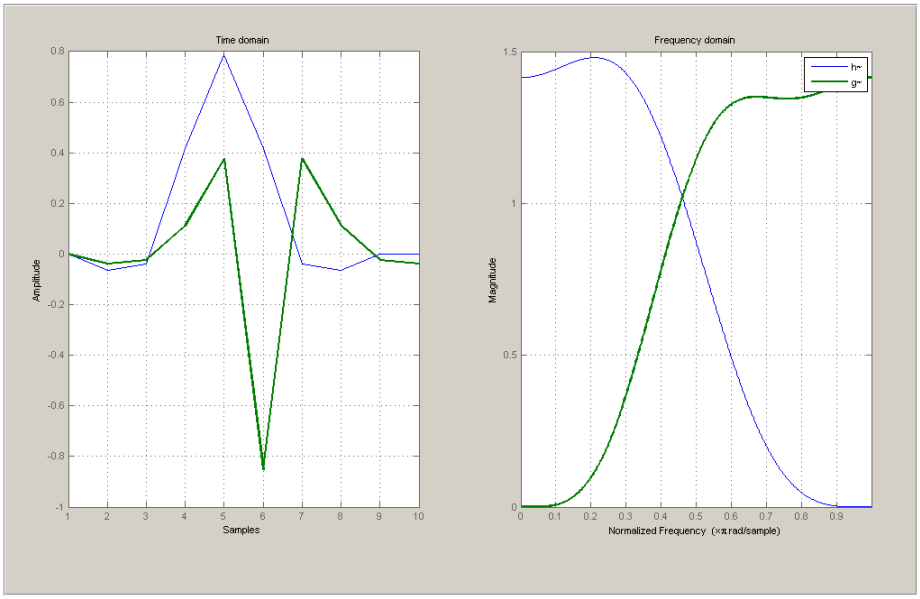


FIGURE 3.12 – Filtres $\tilde{h}$ et $\tilde{g}$

3.5.5 Choix de l'ondelette

L'ondelette « bior4.4 »présente l'avantage d'avoir plus de moments nuls que l'ondelette »bior2.2 « . La seconde nécessite cependant moins de calculs étant donné son support plus réduit.

Le choix de l'ondelette la plus adaptée a été effectué après application de la KLT à un enregistrement EEG d'une durée de 30 minutes sur 22 canaux. Les coefficients en ondelettes ont été seuillés à 2 afin de vérifier pratiquement si l'ondelette 9/7 donnait davantage de coefficients de faible amplitude.

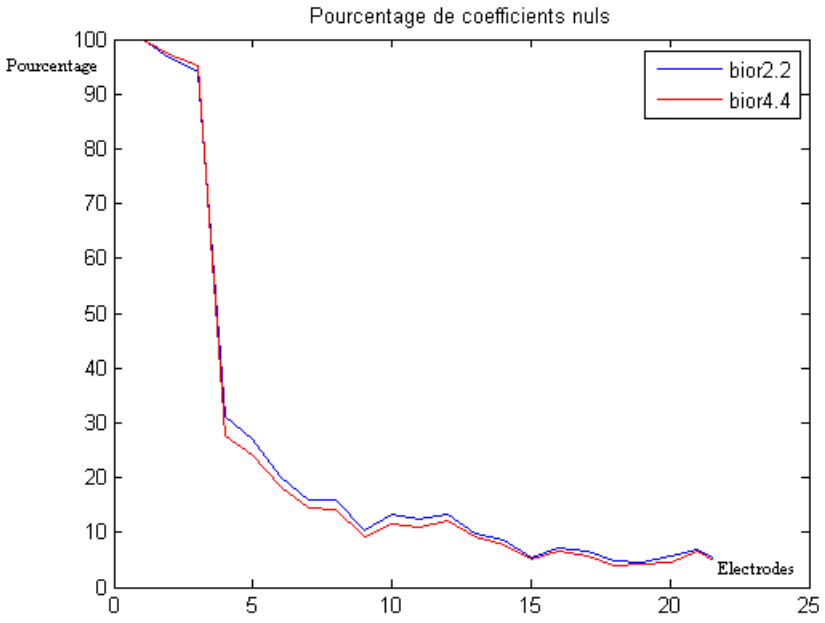


FIGURE 3.13 – Taux de coefficients en ondelettes nuls

La figure 3.13 montre que pour les 22 canaux, l'ondelette bior2.2 donne autant de coefficients nuls si ce n'est plus que que l'ondelette bior4.4 : ceci indique, d'une part que la régularité lipschitzienne du signal EEG n'est pas uniforme, d'autre part, sa régularité ponctuelle est telle que :

$$\alpha \leq 2 \tag{3.54}$$

L'ondelette bior4.4 n'améliore donc pas globalement le rendement de la compression du signal EEG, en plus d'allonger la durée des calculs.

Ci-dessous l'exemple de l'analyse d'un signal EEG d'une durée de 2 secondes :

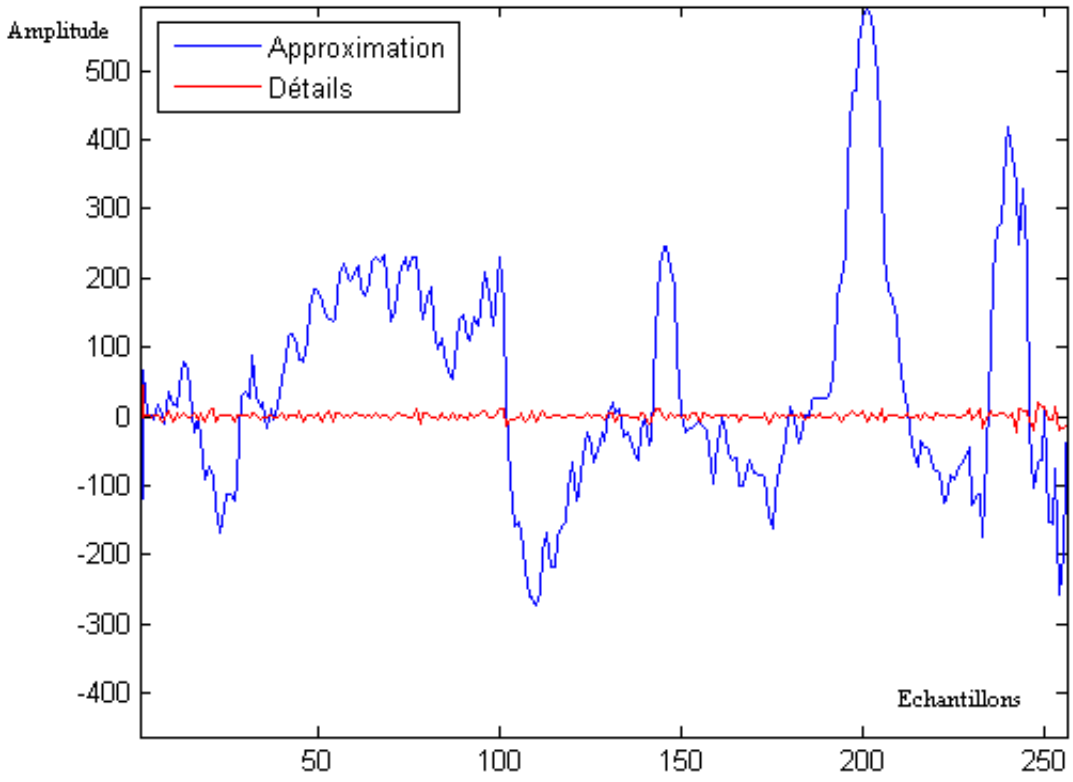


FIGURE 3.14 – Approximation et détails

Les figures 3.15 et 3.16 représentent les histogrammes des coefficients d'approximation et de détails. On voit clairement que les détails sont très concentrés autour de 0.

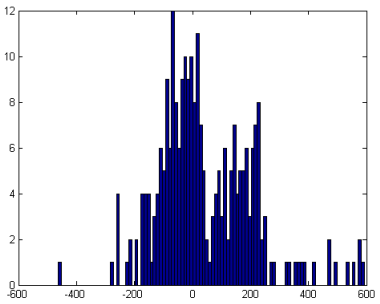


FIGURE 3.15 – Histogramme - approximation

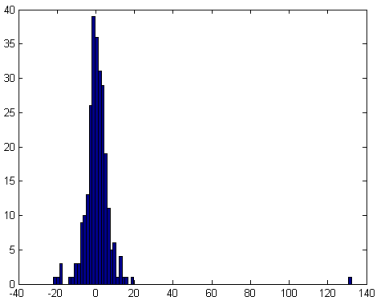


FIGURE 3.16 – Histogramme - détails

3.6 Représentation polyphase

Cette représentation permet d'introduire la procédure de lifting. Le filtre h est représenté comme suit :

$$h(z) = h_e(z^2) + z^{-1}h_o(z^2) \tag{3.55}$$

où h_e contient les coefficients pairs et h_o contient les coefficients impairs.

$$h_e(z) = \sum_k h_{2k} z^{-k} = \frac{h(z) + h(-z)}{2} \tag{3.56}$$

$$h_o(z) = \sum_k h_{2k+1} z^{-k} = \frac{h(z) - h(-z)}{2z^{-1}} \tag{3.57}$$

La matrice polyphase est donnée comme suit :

$$P(z) = \begin{bmatrix} h_e(z) & g_e(z) \\ h_o(z) & g_o(z) \end{bmatrix} \tag{3.58}$$

La condition de reconstruction parfaite est alors exprimée par l'égalité suivante :

$$P(z)\tilde{P}(z^{-1})^t = I \tag{3.59}$$

Le déterminant de la matrice P est inversible si celui-ci est un monôme de Laurent en z c.-à.d. $detP(z) = Cz^l$.
Le cas important où $P = I$ correspond à celui de l'ondelette dite paresseuse (lazy wavelet) : le signal est simplement décomposé en échantillons pairs et impairs.

3.7 Lifting

3.7.1 Lifting primal

Si la matrice polyphase a un déterminant égal à 1, alors les filtres sont dits complémentaires et dans ce cas il est possible de construire un nouveau filtre tel que :

$$g^{nouv}(z) = g(z) + h(z)s(z^2) \tag{3.60}$$

La nouvelle matrice polyphase est exprimée comme suit

$$P^{nouv}(z) = P(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \tag{3.61}$$

La matrice polyphase duale devient quant à elle :

$$\tilde{P}^{nouv}(z) = \tilde{P}(z) \begin{bmatrix} 1 & 0 \\ -s(z^{-1}) & 1 \end{bmatrix} \tag{3.62}$$

Nous pouvons déduire de là l'expression du nouveau filtre $\tilde{h}^{nouv}$:

$$\tilde{h}^{new}(z) = \tilde{h}(z) - \tilde{g}(z)s(z^{-2}) \tag{3.63}$$

La procédure de lifting primal est illustrée à la figure 3.17

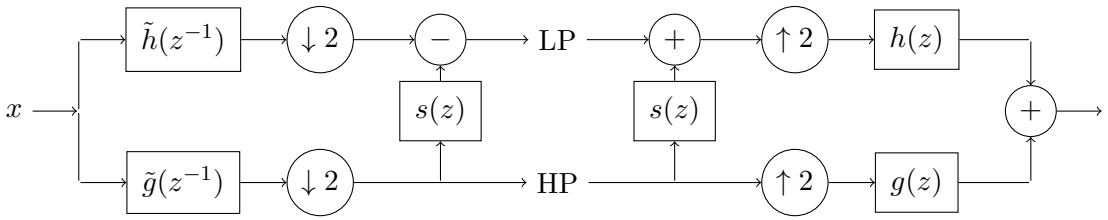


FIGURE 3.17 – Lifting primal

3.7.2 Lifting dual

Si on construit un nouveau filtre h^{new}

$$h^{nouv}(z) = h(z) + g(z)t(z^2) \tag{3.64}$$

La nouvelle matrice polyphase est donnée par l'expression :

$$P^{new}(z) = P(z) \begin{bmatrix} 1 & 0 \\ t(z) & 1 \end{bmatrix} \tag{3.65}$$

Nous obtenons un nouveau filtre

$$\tilde{g}^{nouv}(z) = \tilde{g}(z) - \tilde{h}(z)t(z^{-2}) \tag{3.66}$$

La procédure de lifting dual est illustrée à la figure 3.18

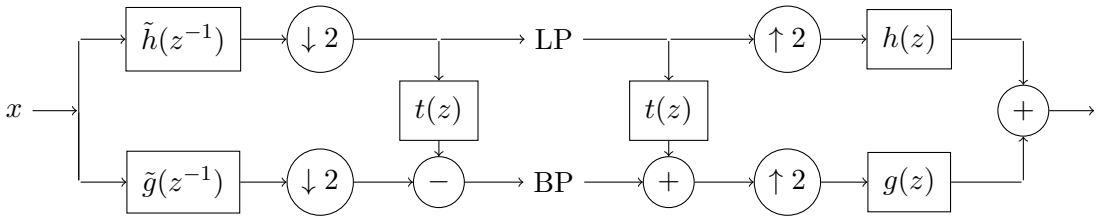


FIGURE 3.18 – Lifting dual

3.7.3 Algorithme d'Euclide

Cet algorithme est appliqué à la division de polynômes de Laurent, à la différence que les solutions ne sont pas uniques dans ce cas-ci.

Soient deux polynômes de Laurent $a(z)$ et $b(z)$. On pose initialement $a_0(z) = a(z)$ et $b_0(z) = b(z)$, l'itération est effectuée comme suit :

$$a_{i+1}(z) = b_i(z) \tag{3.67}$$
$$b_{i+1}(z) = a_i(z)\%b_z \tag{3.68}$$

L'itération est stoppée pour $b_{i=n}(z) = 0$ et $a_n(z)$ est alors le plus grand commun diviseur des polynômes a et b.
Si nous posons également $q_{i+1}(z) = a_i(z)/b_i(z)$ alors nous obtenons l'égalité suivante :

$$\begin{bmatrix} a_n(z) \\ 0 \end{bmatrix} = \prod_{i=n}^{1} \begin{bmatrix} 0 & 1 \\ 1 & -q_i(z) \end{bmatrix} \begin{bmatrix} a(z) \\ b(z) \end{bmatrix} \tag{3.69}$$

équivalant à :

$$\begin{bmatrix} a(z) \\ b(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_n(z) \\ 0 \end{bmatrix} \tag{3.70}$$

Comme nous allons le voir, l'algorithme d'Euclide est très important pour la construction d'une procédure de lifting.

3.7.4 Algorithme de factorisation

Nous avons :

$$\begin{bmatrix} h_e(z) \\ h_o(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K \\ 0 \end{bmatrix} \tag{3.71}$$

A partir d'un filtre h, il est toujours possible de déterminer un filtre g^0 complémentaire pour obtenir une matrice polyphase P dont le déterminant vaut 1 :

$$P^o(z) = \begin{bmatrix} h_e(z) & g_e^0(z) \\ h_o(z) & g_o^0(z) \end{bmatrix} = \prod_{i=1}^{n} \begin{bmatrix} q_i(z) & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \tag{3.72}$$

Par des transformations matricielles, nous obtenons :

$$P^0(z) = \prod_{i=1}^{n} \begin{bmatrix} 1 & q_{2i-1}(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ q_{2i}(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \tag{3.73}$$

On peut ensuite obtenir le filtre g à partir de g^0 par un lifting primal :

$$P(z) = P^0(z) \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \tag{3.74}$$

Finalement :

$$P(z) = \prod_{i=1}^{m} \begin{bmatrix} 1 & s(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ t_i(z) & 1 \end{bmatrix} \begin{bmatrix} K & 0 \\ 0 & \frac{1}{K} \end{bmatrix} \tag{3.75}$$

Cette formule signifie qu'il est possible d'effectuer une transformation en ondelettes discrètes à l'aide d'une succession de m liftings primaux et duaux.

La matrice polyphase duale est donnée par l'expression :

$$\tilde{P}(z) = \prod_{i=1}^{m} \begin{bmatrix} 1 & 0 \\ -s_i(z^{-1}) & 1 \end{bmatrix} \begin{bmatrix} 1 & -t_i(z^{-1}) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{K} & 0 \\ 0 & K \end{bmatrix} \tag{3.76}$$

Les figures 3.19 et 3.20 illustrent la procédure de lifting et l'opération inverse (aux multiplications par K et $\frac{1}{K}$ près).

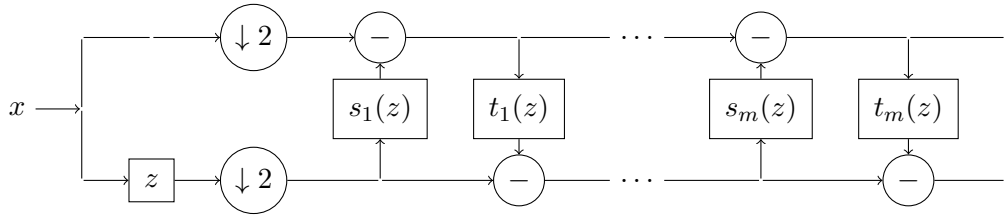


FIGURE 3.19 – Procédure de lifting

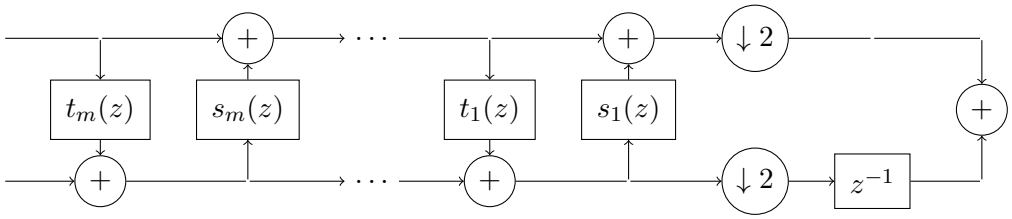


FIGURE 3.20 – Lifting inverse

3.7.5 Lifting d'ondelette bior2.2

Comme nous l'avons vu auparavant, il est possible de déduire une factorisation (parmi toutes celles possibles) de la matrice polyphase en liftings primaux et duaux. Pour ce faire, nous faisons appel aux fonctions intégrées dans la Wavelet Toolbox de MATLAB.

Le schéma correspondant en lifting est obtenu grâce aux fonctions *liftwave* et *displs* :

$$P(z) = \begin{bmatrix} 1 & 0.25z^{-1} + 0.25 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -0.5z - 0.5 & 1 \end{bmatrix} \begin{bmatrix} 1.4142 & 0 \\ 0 & 0.7071 \end{bmatrix} \quad (3.77)$$

3.7.6 Complexité algorithmique

L'intérêt principal du lifting consiste en la simplification de l'algorithme et la diminution du nombre d'opérations à exécuter [23]. Les opérations de filtre étant notamment pratiquées après sous-échantillonnage et non après.

Si l'on évalue la complexité des calculs en termes de nombre d'opérations d'addition et de multiplication, la comparaison entre l'algorithme standard et l'algorithme de lifting donne les résultats suivants [23] :

Ondelette	**Algorithme standard**	**Lifting**	**Gain**
Filtres à 4 coefficients	14	9	56%
Filtres à 6 coefficients	22	14	57 %
CDF 9/7	23	14	64 %

TABLE 3.1 – Tableau comparatif

3.8 Conclusion

En partant des notions fondamentales sur les ondelettes, nous sommes arrivés à une exploitation de celles-ci dans l'algorithme de compression. Nous avons également choisi l'ondelette la plus adaptée et étudié l'implémentation d'une transformation par lifting.
Dans le chapitre suivant, nous nous intéressons à l'étape importante de codage entropique, appliquée dans notre cas aux coefficients en ondelettes.

Chapitre 4

Codage entropique

4.1 Introduction

Toutes les données numériques peuvent être vues comme des valeurs émises par une source aléatoire X et formant un ensemble $\mathcal{A}_X = \{a_1, a_2, \dots a_M\}$. La théorie de l'information a pour objectif de déterminer une représentation binaire et non-redondante de ces données pour diminuer leur taille (en l'occurence les coefficients produits par le lifting d'ondelettes) [12, 20]. Dans notre cas, cette opération doit se faire sans distorsion, on parle alors de ***codage entropique***. Nous étudions dans ce chapitre deux codeurs entropiques différents.

4.2 Définitions

4.2.1 Entropie

L'entropie d'une source est donnée par la relation suivante :

$$H(X) = \sum_{x \in \mathcal{A}_X} P(x) \log \frac{1}{P(x)} \tag{4.1}$$

où log exprime le logarithme en base 2 étant donné la nature binaire de la représentation. L'entropie quantifie l'information fournie par une source aléatoire.

4.2.2 Théorème du codage de source

C'est un théorème que l'on doit à Claude Shannon :

> ***« N variables indépendantes et identiquement distribuées d'entropie H peuvent être compressées sur plus de NH bits avec une perte négligeable d'information si***

$N \to \infty$. ***Inversement, une compression sur moins de*** NH ***bits entraîne une distorsion du signal.*** »

C'est un théorème fondamental en théorie de l'information car il fixe une condition sine qua non et la limite (l'entropie) pour une réduction de la taille des données sans détériorer ces dernières. Dans le cas du signal EEG, pour des raisons médicales et légales, toute distorsion du signal pouvant fausser le diagnostic est exclue.

4.2.3 Code

Un code C consiste en un mapping de $\mathcal{A}_X$ vers $\{0,1\}$ où chaque symbole x émis par la source est représenté par un mot $c(x)$ de longueur $l(x)$:

$$\mathcal{A}_X \xrightarrow{C} \{0,1\}^+ \tag{4.2}$$

Par la suite, nous étudierons et comparerons deux méthodes de codage : le codage de Huffman et le codage arithmétique.

4.2.4 Code préfixe

Egalement appelé code instantané, c'est un code où aucun mot ne peut servir de préfixe à un autre c.-à-d. que le décodeur est en mesure de distinguer chaque mot de l'autre et de la décoder sans ambiguïté.

4.2.5 Longueur moyenne

La longueur moyenne d'un symbole du code est exprimée par la relation suivante :

$$L(C) = \sum_{x \in \mathcal{A}_X} P(x) l(x) \tag{4.3}$$

4.2.6 Inégalité de Kraft McMillan

Soit un code binaire instantané, la suite formée par la longueur de chaque symbole est notée l_i et satisfait à l'inégalité suivante dite de Kraft-McMillan :

$$\sum_i 2^{-l_i} \leq 1 \tag{4.4}$$

Inversement, à partir d'une suite de l_i satisfaisant à cette inégalité, il est possible de déduire un code préfixe.

Cette inégalité permet de déduire qu'un code préfixe existe toujours et remplit la condition suivante :

$$H(X) \leq L(C) \leq H(X) + 1 \tag{4.5}$$

Le cas idéal $L(C) = H(X)$ n'est obtenu que si

$$\sum_i 2^{-l_i} = 1 \tag{4.6}$$

$$l_i = \log(\frac{1}{p_i}) \tag{4.7}$$

4.3 Codage de Huffman

Le codage de Huffman est une méthode dûe à David Huffman (1952). La méthode de Huffman est une technique de codage quasi-optimale sans perte et générant un code instantané avec un surdébit maximal de 1 bit/symbole. Plusieurs normes de compression font appel à ce codage (JPEG,MP3 et MPEG-2).

Cette méthode de codage consiste en la création d'un arbre binaire pour construire le mapping de $\mathcal{A}_X$ vers $\{0, 1\}$

4.3.1 Principe

On prend d'abord tous les symboles de $\mathcal{A}_X$. Les deux noeuds ayant les probabilités associées les plus faibles (noeuds fils) fusionnent pour créer un noeud père qui se voit associer la somme des deux probabilités tandis que les fils se voient assigner les bits 0 et 1. La procédure est répétée de manière récursive jusqu'à arriver au sommet de l'arbre avec une probabilité 1. A noter qu'en règle générale, le zéro est attribué au fils gauche et le 1 au fils droit.

4.3.2 Décodage

Au décodage, en recevant le premier bit du mot binaire, on parcourt le même arbre binaire généré aupravant jusqu'à détecter un mot.

Un exemple de codage de Huffman extrait de [12] est fourni en annexes.

4.3.3 Surdébit

Le codage de Huffman a pour inconvénient de créer un surdébit allant de 0 à 1 bit par rapport à la longueur optimale. Ceci peut se révéler gênant pour une source de faible entropie et grever les performances de la compression.

Néanmoins, des travaux récents [16] permettent de mieux estimer la redondance persistante après codage. Les bornes supérieure et inférieure de cette redondance sont données respectivement par les formules suivantes, connaissant la probabilité associée à un symbole arbitraire :

$$R_{max} = \begin{cases} 2 - p - \mathcal{H}(p) \text{ si } 0.5 \leq p < 1 \\ 1 + p - \mathcal{H}(p) \text{ si } 0 \leq p < 0.5 \end{cases} \tag{4.8}$$

$$R_{min}(p) = mp - (p) - (1-p)\log(1 - 2^{-m}) \tag{4.9}$$

avec :

$$m = \lfloor -\log p \rfloor \text{ ou } \lceil -\log p \rfloor \tag{4.10}$$

$$\mathcal{H}(p) = -p\log p - (1-p)log(1-p) \tag{4.11}$$

Ces formules sont illustrées à la figure 4.1 ci-dessous. Les conclusions adéquates dans notre cas seront tirées au chapitre suivant.

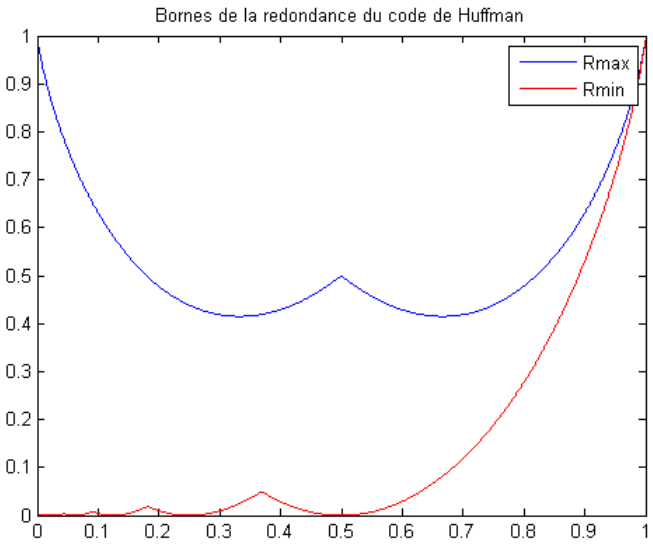


FIGURE 4.1 – Redondance du code de Huffman

4.4 Codage arithmétique

Le codage arithmétique est une méthode de codage de source sans bruit avec codes à longueur variable. Le codage arithmétique est dû entre autres

à Rissanen, Pasco, Witten, Neal et Cleary.

S'il est réellement optimal au sens de Shannon, le codage arithmétique a néanmoins pour inconvénient d'être assez complexe à mettre en oeuvre et de tomber, dans certains cas, dans le domaine de la propriété intellectuelle (soumis à royalties ou octroi de licence). Autre inconvénient majeur, il n'est pas instantané.

4.4.1 Principe

On souhaite coder une séquence de N symboles $S = \{x_1, x_2, \ldots, x_N\}$. Le codage arithmétique associe à cette suite de symboles une valeur réelle v entre $[0, 1)$. En théorie, il est donc possible de coder une séquence infinie de symboles. Dans la pratique, la précision de la représentation des nombres pose des limites.

Notons deux vecteurs correspondant à la densité de probabilité et à la fonction de répartition :

$$\mathbf{p} = [p_1, p_2 \ldots p_M] \tag{4.12}$$

$$\mathbf{c} = [c_1, c_2 \ldots c_{M-1}, c_M] \tag{4.13}$$

Le codage consiste en une subdivision récursive d'intervalles $\Phi_k(S) = [\alpha_k, \beta_k)$ pour $k = 1, 2, \ldots, N$ par la méthode d'Elias. A la fin de cette série de subdivisions, la séquence est représentée par un nombre réel v.

Les intervalles en question peuvent également être notés $|b, l\rangle$ où b est la base de l'intervalle et l sa longueur.

Les suites α_k et β_k obéissent aux conditions suivantes :

$$0 \leq \alpha_k \leq \alpha_{k+1} \tag{4.14}$$

$$\beta_{k+1} \leq \beta_k \leq 1 \tag{4.15}$$

et sont obtenues comme suit :

$$\Phi_0(S) = |b_0, l_0\rangle = |0, 1\rangle \tag{4.16}$$

$$\Phi_k(S) = |b_k, l_k\rangle = |b_{k-1} + c(s_k) l_{k-1}, p(s_k) l_{k-1}\rangle \tag{4.17}$$

A la fin de l'étape de codage, nous obtenons une valeur $\hat{v}(S)$ appartenant à l'intervalle $\Phi_N(S)$. Il est possible de choisir n'importe quelle valeur réelle contenue dans cet intervalle mais il est préférable de prendre la représentation binaire la plus courte.

Le nombre minimal de bits pour représenter $\hat{v}$ est donnée par la formule suivante :

$$B_{min} = \lceil -log_2(l_N) \rceil \tag{4.18}$$

(on arrondit vers la valeur supérieure).

4.4.2 Décodage

La séquence de symboles doit être récupérée à partir de la valeur binaire transmise :

$$\hat{S}(\hat{v}) = \{\hat{x}_1(\hat{v}), \hat{x}_2(\hat{v}), \ldots, \hat{x}_N(\hat{v})\} \tag{4.19}$$

Le décodage est effectué de manière similaire à l'opération de codage et fait appel aux équations récursives suivantes :

$$\tilde{v}_1 = \hat{v} \tag{4.20}$$

$$\hat{x}_k(\hat{v}) = \{x : c(x) \leq \tilde{v}_k < c(x+1)\}, k = 1, 2, \ldots, N, \tag{4.21}$$

$$\tilde{v}_{k+1} = \frac{\tilde{v}_k - c(\hat{x}_k(\hat{v}))}{p(\hat{x}_k(\hat{v}))}, k = 1, 2, \ldots, N-1 \tag{4.22}$$

Ces équations permettent de retrouver les symboles de s_1 à s_N dans l'ordre.

Une autre méthode de décodage existe. Elle reconstruit les intervalles à l'identique du codeur pour identifier les symboles émis et est décrite par les équations suivantes :

$$\Phi_0(\hat{S}) = |b_0, l_0\rangle = |0, 1\rangle \tag{4.23}$$

$$\hat{x}_k(\hat{v}) = \left\{x : c(x) \leq \frac{\hat{v} - b_{k-1}}{l_{k-1}} < c(x+1)\right\} \tag{4.24}$$

$$\Phi_k(\hat{S}) = |b_k, l_k\rangle = |b_{k-1} + c(\hat{x}_k(\hat{v}))l_{k-1}, p(\hat{x}_k(\hat{v}))l_{k-1}\rangle \tag{4.25}$$

Un exemple de codage/décodage extrait de [20] est fourni en annexes.

4.4.3 Optimalité du codage arithmétique

Le codage arithmétique est optimal pour une source sans mémoire au sens de Shannon : la fonction de répartition des codes en sortie du codeur

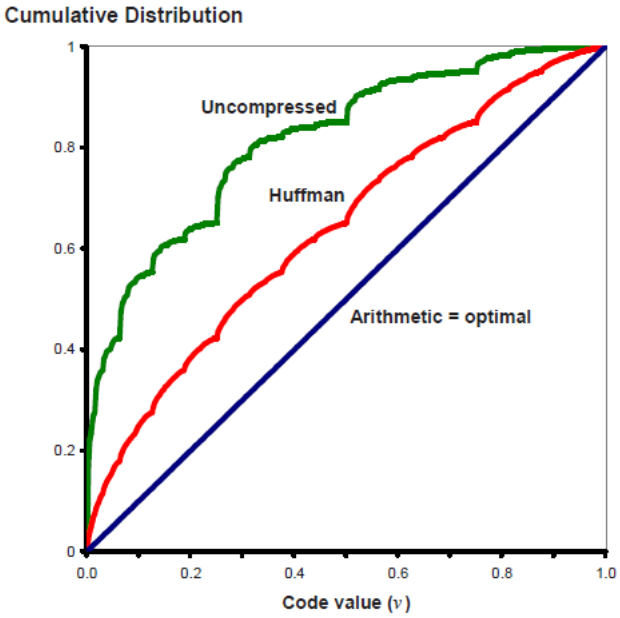


FIGURE 4.2 – Comparaison de techniques de codage

est linéaire de $(0,0)$ à $(1,1)$ (figure 4.2), en effet contrairement au codage de Huffman, le codeur arithmétique peut n'allouer qu'une fraction de bit à un symbole. L'intérêt se fait particulièrement ressentir pour une source à faible entropie (sources binaires notamment) ;

Cette optimalité au sens de Shannon peut également être déduite à partir de la formule exprimant l_N. Le nombre de bits nécessaires pour coder une séquence de N symboles obéit à la relation suivante avec un overhead noté σ (informations à joindre aux données codées à proprement parler) [20] :

$$B_S \leq \frac{\sigma - log_2(l_N)}{N} \text{ bits/symbole} \tag{4.26}$$

or :

$$l_N = \prod_{k=1}^{N} p(s_k)$$

alors :

$$E\{B_s\} \leq \frac{\sigma - \sum_{k=1}^{N} E\{log_2 p(x_k)\}}{N}$$
$$E\{B_s\} \leq H(\Omega) + \frac{\sigma}{N}$$

Etant donné que le nombre moyen de bits alloués par symbole ne peut être inférieur à l'entropie alors :

$$H(\Omega) \leq \bar{B} \leq H(\Omega) + \frac{\sigma}{N}$$

et :

$$lim_{N\to\infty} \left\{ \bar{B} \right\} = H(\Omega) \tag{4.27}$$

le codage arithmétique est donc optimal.

4.5 Codage adaptatif

Une compression efficace nécessite une modélisation statistique de la source aussi fidèle que possible.

Le codage adaptatif correspond à une estimation des probabilités d'apparition de chaque symbole durant l'étape de codage, le codeur et le décodeur devant être synchrones pour un bon fonctionnement.

Plusieurs stratégies de modélisation de la source sont envisageables :
- Utiliser une distribution unique, obtenue à partir d'observations prolongées des données à compresser. Cette méthode est peu flexible et ne donne pas des résultats optimaux pour des sources complexes ;
- Utiliser une distribution prédéfinie avec des paramètres estimées de manière adaptative (moyenne, variance). Le codeur et le décodeur peuvent adapter ces paramètres de la même manière ;
- Codage en deux passes : durant la première passe, des statistiques sont récoltées, la seconde passe code les données. Le dictionnaire (codebook) contenant les statistiques récoltées, nécessaire au décodeur peut également être compressé pour diminuer l'overhead ;
- Avoir recours à une distribution établie à partir des symboles codées auparavant et donc totalement adaptative. La mise en oeuvre de cette technique est complexe.

Nous allons à présent voir quelle stratégie convient le mieux au signal EEG.

4.6 Application au signal EEG

Un signal EEG d'une durée de 30 secondes a été analysé avec une ondelette bior2.2 après application de la KLT. Les coefficients ont ensuite été seuillés et leur histogrammes calculés afin d'observer leurs distributions. Les résultats obtenus pour deux électrodes sont illustrés par les figures 4.3 et 4.4 :

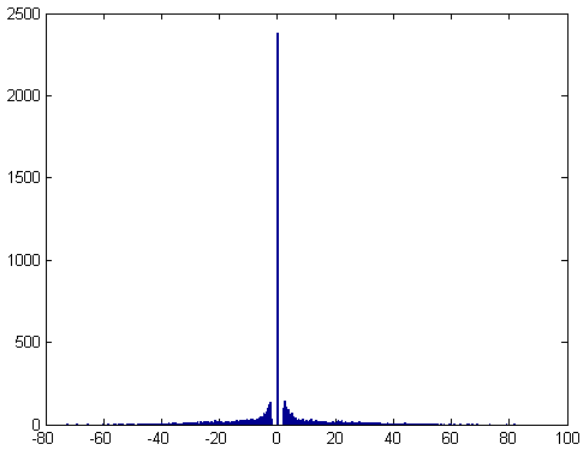


FIGURE 4.3 – Histogramme de coefficients 1

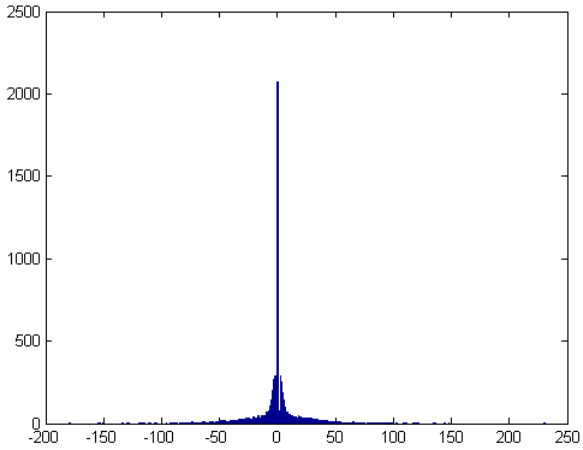


FIGURE 4.4 – Histogramme de coefficients 2

Les distributions diffèrent d'une électrode à l'autre et sont difficilement modélisables par des distributions connues. Les deux premières méthodes de codage évoquées auparavant sont donc à exclure. Un codage totalement adaptatif est la seule solution à même d'assurer une compression correcte et efficace. Pour des raisons d'optimalité et de complexité, nous retenons la 3ème méthode, celle du codage en 2 passes (parfois appelé codage semi-adaptatif).

4.7 Conclusion

Après avoir étudié deux codeurs entropiques, nous en avons sélectionné un sur la base de la propriété d'instantanéité et avons choisi une méthode semi-adaptative.
Dans le prochain chapitre, nous comparons les performances globales de cet algorithme de compression avec d'autres.

Chapitre 5

Résultats

5.1 Tests

L'algorithme de compression proposé a été testé à l'aide d'une interface graphique sous MATLAB. Dans une logique d'optimisation, une implémentation en langage C est très nettement préférable, en particulier pour l'étape de codage/décodage entropique.

FIGURE 5.1 – Algorithme proposé

Les données EEG utilisées durant nos tests proviennent de la base de données CHB (Children's Hospital Boston) du MIT (Massachusetts Institute of Technology) [6, 18]. Chaque enregistrement comprend 22 canaux échantillonnés à 256 Hz et sur 16 bits.

Les enregistrements utilisés comme échantillons dans nos tests sont mentionnés dans le tableau 5.1.

5.1.1 Taille des échantillons

Les blocs sur lesquels sont effectués les tests sont d'une durée de 30 secondes. Cette durée, longue à priori, a été choisie pour deux raisons :

1. Eliminer davantage de redondance ;
2. Répartir l'overhead dû au dictionnaire sur davantage d'échantillons afin de ne pas grever les performances de la compression ;

Enregistrement	Sexe	Age	Crises
chb01-01/06	F	11	non
chb02-04	M	11	non
chb03-01/10/31	F	14	oui
chb04-01/08	M	22	oui
chb05-04/ch5-06	F	7	oui
chb10-20	M	3	oui

TABLE 5.1 – Enregistrements

5.1.2 Codeur

Nous n'avons retenu que le codeur de Huffman dans nos tests. En effet, le codeur arithmétique a présenté deux inconvénients majeurs :

1. Il n'est pas instantané ;
2. Le codage de 2 symboles a nécessité de 8 à 36 bits du fait des disparités importantes entre les probabilités et la valeur élevée de l'entropie (de 5 à 7 bits). De tels écarts conjugués au premier inconvénient nécessitent l'usage de tags pour décoder correctement les données, mais au prix d'une performance bien moindre de la compression. Le codeur arithmétique est donc plus adapté à des sources avec dictionnaire moins volumineux (sources binaires en particulier).

5.2 Algorithmes de référence

Les résultats obtenus avec l'algorithme proposé ont été comparés à des algorithmes de référence utilisés par le logiciel libre de compression 7-Zip [30] :

1. SPIHTbio 1-D (Set Partitioning In Hierarchical Trees - parition d'ensembles en arbres hiérarchisés) : basé sur une analyse multirésolution entière dite S+P, l'algorithme SPIHT a été initialement conçu pour la compression d'images mais une version existe pour la compression de signaux biomédicaux 1-D [29] ;
2. PPMd (Prediction by Partial Matching by dmitri - prédiction par reconnaissance partielle) : c'est un algorithme de compression faisant appel à la modélisation de contextes et à la prédiction ;
3. DEFLATE : il est basé sur le codage de Huffman et l'algorithme LZ77 (codeur à dictionnaire de Lempel-Ziv) ;
4. DEFLATE64 : amélioration du précédent algorithme avec quelques modifications ;

5. LZMA Lempel-Ziv-Markov-Chain-Algorithm : il fait appel à l'algorithme LZ77 et à un codeur similaire au codeur arithmétique. C'est la méthode de compression par défaut du logiciel 7-zip ;
6. LZMA 2 : évolution du précédent ;
7. bzip2 : il fait appel à la transformation de Burrows-Wheeler et au codage de Huffman.

5.3 Comparaison des résultats

Les données brutes ont été inscrites dans un fichier binaire sans header avec l'outil *fwrite* de MATLAB ; les échantillons ont été écrits bout à bout. Ce fichiers binaires d'une taille de 330 ko ont ensuite été compressés à l'aide du logiciel 7-zip en mode ULTRA (compression maximale) avec mots de 16 bits. A noter que le premier algorithme de référence a été testé à l'aide du logiciel spbio.
On observera également que ces algorithmes de compression n'occasionnent aucune perte au signal.

Algorithme	Taille (ko)	Taux de compression	Bits par symbole
Proposé	116.8	64.60 %	5.66
SPIHTbio	159	51.82 %	7.71
PPMd	161	51,21 %	7.81
DEFLATE	196	40.61 %	9.50
DEFLATE64	194	41.21 %	9.41
LZMA	160	51.52 %	7.76
LZMA 2	160	51.52 %	7.76
bzip 2	169	48.79 %	8.20

TABLE 5.2 – Enregistrement chb03_01

Algorithme	Taille (ko)	Taux de compression	Bits par symbole
Proposé	154.8	53.08 %	7.51
SPIHTbio	179	45.76 %	8.68
PPMd	180	45.45 %	8.73
DEFLATE	218	33.94 %	10.57
DEFLATE64	217	34.24 %	10.52
LZMA	188	43.03 %	9.12
LZMA 2	188	43.03 %	9.12
bzip 2	186	43.64 %	9.02

TABLE 5.3 – Enregistrement chb01_01

Les résultats obtenus avec l'algorithme proposé sont meilleurs que tous

ceux obtenus avec d'autres méthodes et que nous avons pu tester, le taux de compression est 8 à 13 % au-dessus du meilleur taux obtenu avec les autres algorithmes.

5.4 Distorsion

5.4.1 Sources de distorsion

La distorsion du signal a trois origines :

1. Seuillage éventuel des coefficients ;
2. Quantification scalaire ;
3. Précision limitée des calculs.

5.4.2 Résultats

Pour étudier la distorsion occasionnée par notre algorithme et en particulier par le seuillage, nous avons calculé deux métriques de distorsion :

1. SNR : Rapport signal sur bruit (Signal-to-Noise Ratio) donné par la formule suivante :

$$10 \log \frac{(\sum_{i=1}^{N} (S(k) - \bar{S})^2}{(\sum_{i=1}^{N} S(k) - \hat{S}(k))^2}$$

2. NMAE : Erreur maximale normalisée (Normalized Maximum Amplitude Error), elle est donnée par la formule suivante :

$$NMAE = \frac{max \left| S - \hat{S} \right|}{max(S) - min(S)}$$

où S,$\hat{S}$ et $\bar{S}$ représentent respectivement le signal original, le signal reconstitué et la moyenne du signal initial.

Les valeurs obtenues pour ces métriques avec et sans seuillage sont résumées au tableau suivant :

Enregistrement	Seuil	SNR_{min}	SNR_{max}	$NMAE_{min}$	$NMAE_{max}$	Taux
chb_03_01	0	37.6 dB	51.2 dB	$9.57 * 10^{-4}$	$5.71 * 10^{-3}$	61 %
chb_03_01	2	29.3 dB	44.6 dB	$2.32 * 10^{-3}$	$1.57 * 10^{-2}$	64.6 %

TABLE 5.4 – Résultats de la compression

La figure 5.2 illustre l'erreur quandratique moyenne.

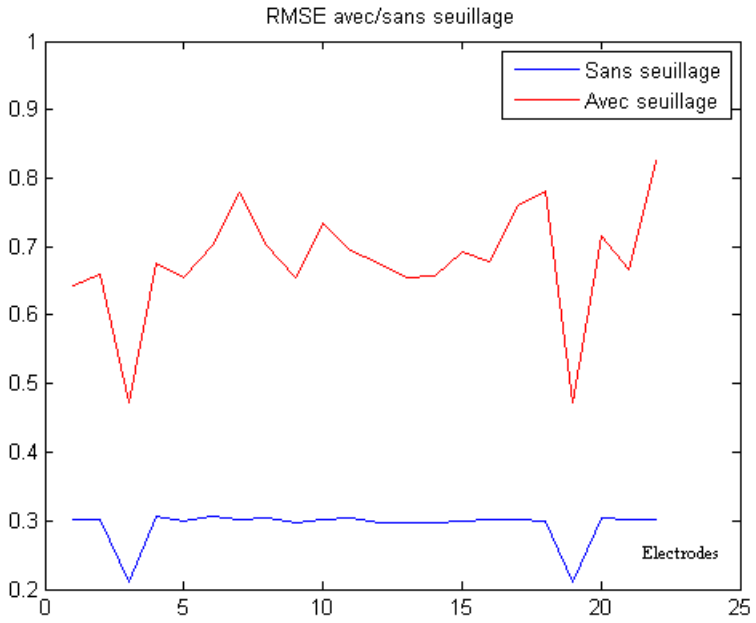


FIGURE 5.2 – RMSE

A la vue des résultats, on constate que le seuillage permet une augmentation appréciable du taux de compression, les résultats demeurant supérieurs à ceux obtenus avec 7zip. Nous constatons également qu'il augmente sensiblement l'erreur quandratique moyenne. Toutefois, celle-ci prend dans un cas comme dans l'autre de très faibles valeurs.

La figure 5.3 contient un tracé critique (crise) pour le canal correspondant au SNR_{min} (l'enregistrement 03_01) avec/sans seuillage à 2 des coeffecients.

La figure 5.4 correspond quant à elle à un enregistrement sain. Les deux figures et les métriques calculées auparavant montrent que la différence entre signal initial et signal reconstruit est quasi-inexistante, avec ou sans seuillage des coefficients et ce même dans le cas le plus défavorable. On peut réellement parler de compression quasi sans perte, tant la distorsion est faible.

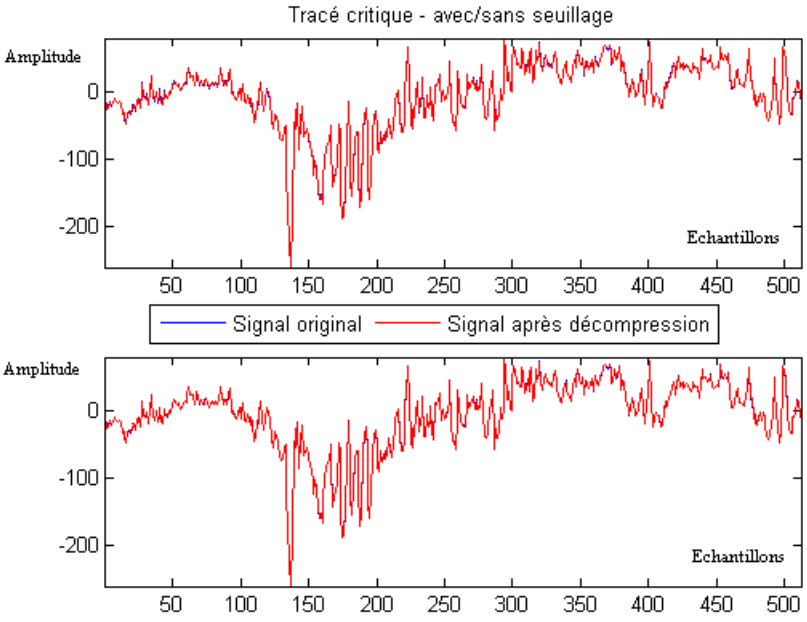


FIGURE 5.3 – Tracé critique initial et décompressé

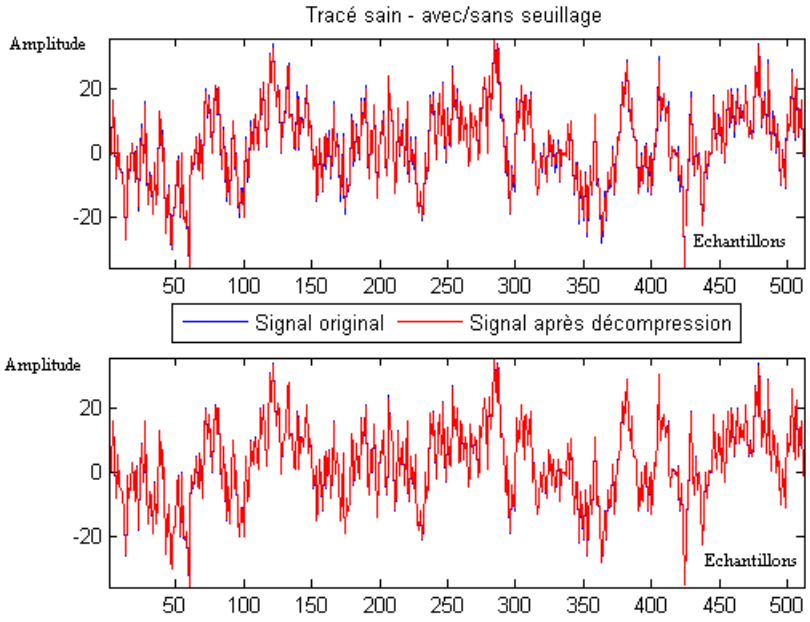


FIGURE 5.4 – Tracé sain initial et décompressé

5.5 Conclusion

Ce chapitre vient en conclusion de la partie théorique de notre travail. Nous avons mesuré et comparé les performances de l'algorithme de compression. Ces dernières sont supérieures à celles des algorithmes auxquels nous l'avons comparé en terme de taux de compression pour une distorsion quasi nulle.
Dans le chapitre suivant, nous nous intéressons à l'implémentation à une des étapes de l'algorithme, en l'occurence le lifting d'ondelettes.

Chapitre 6

DSP TMS320C6713

6.1 Introduction

Notre choix pour l'implémentation s'est porté sur un processeur de traitement de signal numérique DSP car ce type de circuits est optimisé comme son nom l'indique pour le traitement de signal numérique et les opérations élémentaires auxquelles il fait appel (convolution, convolutions circulaires, etc).
Nous nous intéressons ici au DSP TMS320C6713 de la série C6000 de chez Texas Instruments. Nous étudions ses principales caractéristiques et proposons une implémentation du lifting d'ondelette sur ce DSP à l'aide de l'environnement de développement Code Composer Studio.

L'architecture de ce DSP est illustrée à la figure 6.1.

6.2 CPU

Le TMS320C6713 de Texas Instruments est basé sur le cœur C67x à architecture VLIW (Very Long Instruction Word - Mot Instruction Très Long). Ce cœur fonctionne avec une horloge allant de 167 MHz à 225 MHz selon les versions et effectue des calculs aussi bien en notation fixe qu'en virgule flottante (simple et double précision à la norme IEEE 754). Il peut atteindre 450 millions d'opérations de multiplication/accumulation par seconde.

Les performances globales de ce DSP en fonction de la fréquence d'horloge sont résumées au tableau suivant :

où MIPS signifie millions d'opérations par seconde et MFLOPS millions d'opérations en virgule flottante par seconde.

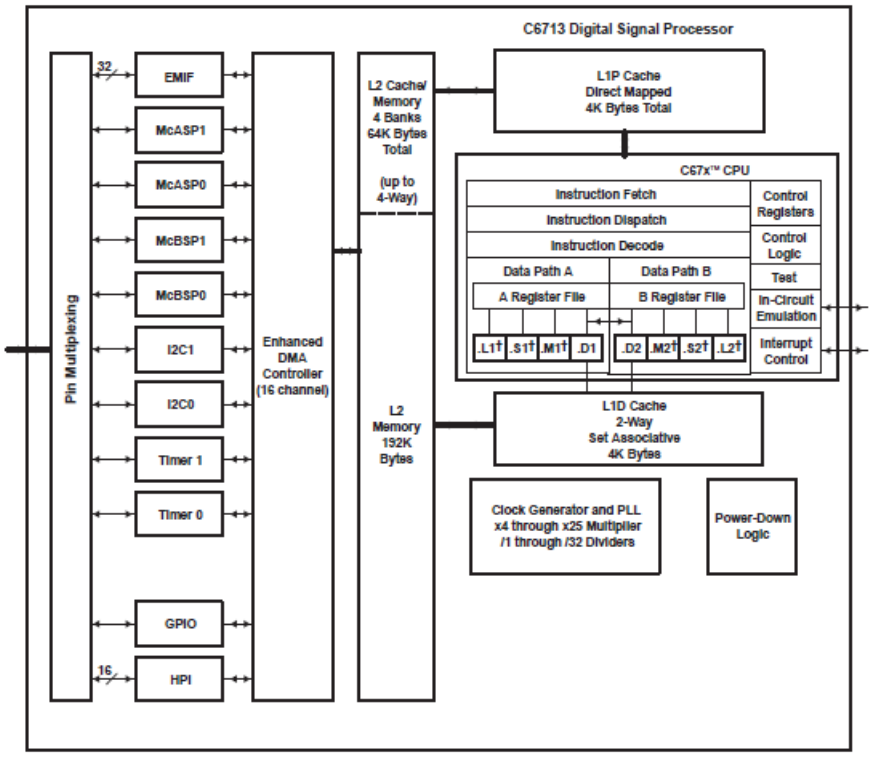


FIGURE 6.1 – Architecture du TMS320C6713

Horloge	MIPS	MFLOPS
167 MHz	1336	1000
200 MHz	1600	1200
225 MHz	1800	1350

TABLE 6.1 – Tableau comparatif

6.2.1 Parallélisme

Les instructions sont extraites en paquets de 8 (fetch packet). Le premier bit de chaque instruction a une signification particulière : il indique si l'instruction doit être exécutée en parallèle avec d'autres instructions (execute packet) au nombre maximal de 8.

Nous décrivons les différentes unités fonctionnelles de ce DSP ci-après avant de citer les contraintes imposées par ce mode de fonctionnement au point suivant.

6.2.2 Unités arithmétiques et logiques

Ce DSP comprend 8 unités fonctionnelles de calcul réparties sur deux côtés (voir annexe) :

1. 2 unités L1 et L2 : elles effectuent des opérations arithmétiques sur 32/40 bits en virgule fixe. Des opérations logiques sur 32 bits et des opérations arithmétiques et de conversion entier/virgule flottante sont également effectuées au niveau de cette unité ;
2. 2 unités S1 et S2 : elles effectuent des opérations arithmétiques et logiques sur 32 bits et des opérations de décalage sur 32/40 bits. Elles intervient également dans les branchements . Les opérations en virgule flottante se limitent aux comparaisons, conversions simple précision/double précision et opérations réciproques ;
3. 2 unités M1 et M2 : elles effectuent uniquement des multiplications en virgule fixe (16x16 et 32x32 bits) et en virgule flottante ;
4. 2 unités D1 et D2 : elle a pour le charge le calcul d'adresses en modes linéaire et circulaire (pour le calcul d'une convolution circulaire et de la FFT) y compris pour le chargement de nombre sur 64 bits pour des calculs en double précision.

6.2.3 Limitations et contraintes

Le parallélisme impose certaines contraintes pour la programmation qu'il est possible de résumer comme suit :

1. Il est impossible de faire appel à la même unité fonctionnelles dans un même paquet à exécuter ;
2. Deux opérations de chargement ou d'écriture vers/depuis le même bloc (A ou B) ne peuvent appartenir au même paquet à exécuter ;
3. Pas plus de quatre lectures d'un même registre dans un paquet à exécuter ;
4. Il est impossible d'écrire sur le même registre dans deux instructions exécutées de manière parallèle.

6.2.4 Chemin de données croisé

Cette fonctionnalité permet à certaines unités arithmétiques d'accéder aux registres de l'autre fichier grâce à deux chemins de données 1X et 2X. Les unités M et S peuvent exploiter un seul opérande grâce aux chemins 1X et 2X tandis que les deux opérandes des unités L peuvent être extraits de cette manière.

Une seule unité peut effectuer une seule lecture par paquet d'exécution par chemin (1X ou 2X). Un exemple de code valide/invalide est fourni en annexes.

6.3 Mémoire cache

La mémoire cache (deux niveaux) est séparée entre données et programmes. Elle est adressée par octets avec 2 bus de 32 bits pour l'accès aux données et un seul port pour les programmes.

6.3.1 Cache niveau L1

1. Données : 4Ko, associativité '2-way' c.-à-d. la mémoire cache est divisée en v ensembles contenant chacun 2 lignes (d'où le 2-way) le numéro d'ensemble correspondant à un bloc mémoire j est obtenu en calculant j *modulo* v.
2. Programmes : 4 Ko en mapping direct c.-à-d. que tout bloc en mémoire principale correspond à une ligne déterminée du cache donnée par la valeur $i = j$ *modulo* c où c est le nombre de lignes du cache et j correspond au bloc de la mémoire principale.

6.3.2 Cache niveau L2

Ce cache comprend :

1. 64 Ko unifiés entre cache et mémoire avec associativité 4-way ;
2. 192 Ko de RAM supplémentaire ;

6.4 Registres

6.4.1 Registres généraux

Le C6713 comprend 32 registres de 32 bits à usage général (données, adressage indirect ou registres de conditions) répartis sur deux blocs A et B. Ces registres peuvent également contenir des données sur 40 bits ou des flottants en 64 bits lorsqu'ils sont appairés. A noter enfin que seuls les registres A4 à A7 et B4 à B7 peuvent être configurés pour un adressage circulaire à partir du registre AMR.

6.4.2 Registres de contrôle

Ces registres assurent le contrôle et indiquent l'état du DSP. Seule l'instruction MVC permet d'écrire dans ces registres ou d'en lire le contenu

1. Registre de contrôle d'adressage (AMR - Address Mode Register) : contrôle le mode d'adressage en offrant le choix entre adressage linéaire et circulaire ;

2. Registre statut CSR (Control Status Register) : comme son nom l'indique, il contrôle différents éléments ou fonctionnalités du DSP (boutisme, activation des interruptions,cache,ect). Il indique également une éventuelle saturation au niveau d'une des unités fonctionnelles ;
3. Registre compteur E1 : il indique sur 32 bits l'adresse du paquet à la phase E1 du pipeline ; (voir chapitre)
4. Registres de contrôle des interruptions.

6.4.3 Registres spéciaux

Nous nous intéressons ici aux registres ayant trait aux calculs en virgule flottante :

1. Registre FADCR (Floating-Point Adder Configuration Register) : il est lié aux unités fonctionnelles L1 et L2 ; il indique d'éventuelles situations de dépassement (overflow), de soupassement (underflow) du mode d'arrondi ou de calculs inexacts ;
2. Registre FAUCR (Floating-Point Auxilary Configuration Register) : ce registre est quasisimilaire au précédent à la différence qu'il concerne les unités S1 et S2 et qu'il indique certaines situations propres aux opérations effectuées par cette unité (opérations réciproques, comparaison)
3. Registre FMCR (Floating-Point Configuration Register) : relatif aux multiplieurs M1 et M2, il fournit les mêmes indications que FADCR ;

6.5 Autres caractéristiques

Nous ne nous intéressons ici qu'aux caractéristiques secondaires susceptibles d'être utiles dans un dispositif EEG :

- Boutisme court et long ;
- Contrôleur EDMA (Enhanced Direct Memory Access) à 16 canaux indépendants ;
- 2 bus I^2C (bus série et synchrone) ;
- Interface E/S à 16 broches avec interruptions externes programmables ;
- Générateur d'horloge à PLL ;
- Deux timers multi-usage à 32 bits.

La liste complète des caractéristiques de ce DSP est fournie en annexes

6.6 Pipeline

Nous décrivons à présent les différentes étapes de l'exécution d'une instruction : le pipeline. Le pipeline comprend trois phases pour l'exécution d'une instruction :

6.6.1 Fetch

Comprend à son tour 4 étapes :

1. PG (Program address Generate) : génération de l'adresse du programme ;
2. PS (Program address Send) : émission de l'adresse vers la mémoire ;
3. PW (Program access ready Wait) : lecture en mémoire ;
4. PR (Program fetch packet Receive) : réception du paquet extrait (fetch packet) au niveau du CPU.

Ce paquet comprend 8 instructions qui passent les 3 premières étapes simultanément.

6.6.2 Décodage

Elle se décompose en deux autres étapes :

1. Dispatching des instructions : le paquet extrait est réparti en paquets d'exécution contenant comme on l'a vu, une à 8 instructions parallèles qui sont ensuite assignées à des unités arithmétiques ;
2. Décodage des instructions : durant cette étape, les registres source et destination ainsi que les chemins de données sont décodés.

6.6.3 Exécution

Cette phase est subdivisée en 10 étapes symbolisées de E1 à E10 pour les opérations en virgule flottante tandis qu'elle se limite à 5 pour les opérations en virgule fixe. Suivant l'instruction à exécuter, différents nombres de phases sont nécessaires.

On peut indiquer à titre d'exemple que la phase E1 correspond à l'évaluation de conditions et à la lecture d'opérandes. La phase E10 correspond quant à elle à une écriture des 32 bits de poids fort dans les registres dans le cas de l'instruction MPYDP (produit en double précision).

6.7 Lifiting

Nous rappelons le schéma du lifting :

Les filtres correspondants aux liftings primal et dual sont donnés par les expressions suivantes :

$$s_1(z) = 0.25z^{-1} + 0.25$$
$$t_1(z) = 0.5 + 0.5z$$

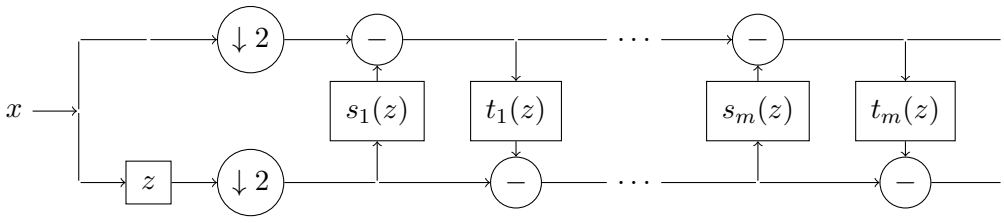


FIGURE 6.2 – Lifting

Ces deux filtres ont l'avantage d'être symétriques. Le calcul d'un élément de la convolution nécessite donc une addition et une multiplication au lieu de deux multiplications et d'une addition. En outre, les valeurs des coefficients 0.25 et 0.5 permettent de remplacer l'opération de multiplication par des opérations de décalage après conversion en entier des valeurs en sortie de la KLT.

L'opération de décomposition ne nécessite pas de recourir à de la mémoire supplémentaire, cet avantage de la méthode du lifting s'ajoute à celui de la durée de calcul. La figure 6.3 illustre l'implémentation du lifting pour un signal en entrée de 8 échantillons :

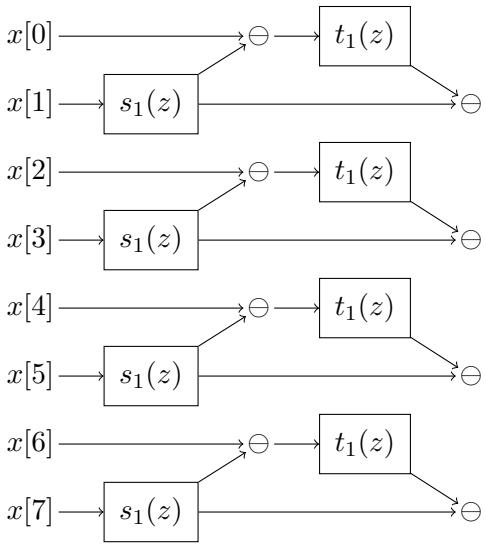


FIGURE 6.3 – Schéma

Le programme correspondant est conçu en langage C. Il est ensuite compilé grâce à l'environnement de développement Code Composer Studio de Texas Instruments. A noter que le compilateur optimise également le code (jusqu'à un certain stade) en tirant parti du parallélisme.
Le tableau suivant fournit des indications sur l'implémentation :

Donnée	**Nombre**
Cycles CPU	292
Cycles NOP (No OPeration)	114
Paquets exécutés	221
Accès en lecture	43
Accès en écriture	15
Instructions décodées	279
Instructions exécutées	275
Accès au cache L1D - lecture	43
Accès au cache L1D - écriture	15
Accès au cache L1P	41

TABLE 6.2 – Résultats de l'implémentation

Le programme obtenu est fourni en annexes. La durée d'exécution du programme est d'approximativement $1.5\mu s$, on peut déduire qu'une durée approximative de $0.03s$ est nécessaire pour effectuer l'ensemble de la décomposition. Une programmation en assembleur, quoique plus complexe et plus coûteuse en terme de durée de réalisation est préférable.

Nous constatons une différence entre le nombre d'instructions décodées et d'instructions exécutées. Ceci est dû à des réinitialisations elles-mêmes dûes aux dépendances de contrôle et de données.

6.8 Conclusion

Nous avons mesuré grâce à ce chapitre l'efficacité du lifting, l'adéquation d'un DSP avec le traitement numérique du signal, et le gain en temps offert par la programmation du DSP en langage C. Nous avons également constaté qu'une marge de manœuvre importante restait pour l'implémentation totale de l'algorithme.

Conclusion générale

L'algorithme proposé répond parfaitement au problème posé : la compression des signaux EEG. La combinaison de la KLT, du lifting d'ondelettes et du codage de Huffman étudiés tout au long de ce travail offre des performances supérieures à celles de tous les algorithmes généralistes auxquels il a été comparé.

En tirant profit de ces outils issus du traitement de signal et de la théorie de l'information, nous avons obtenu des taux de compression atteignant jusqu'à 64 %.

Nous proposons néanmoins quelques pistes à priori intéressantes en vue d'améliorer et les performances, et l'efficacité de l'algorithme :

1. Expérimenter une décomposition en ondelettes sur 2 niveaux ou en paquets d'ondelettes afin de traiter les fréquences au-delà des rythmes gamma ;
2. Expérimenter une étape de run-length avant le codage de Huffman ;
3. Expérimenter une architecture avec traitement en parallèle des différents canaux sur FPGA.

Par conséquent, les résultats obtenus, quoique déjà très satisfaisants sont un prélude à des travaux futurs plus poussés, toujours dans la même optique.

Bibliographie

[1] DUBIN, M. Brodmann Areas in the Human Brain with an Emphasis on Vision and Language [WWW] University of Colorado, Dept. of Molecular, Cellular & Developmental Biology. Disponible à : `http://spot.colorado.edu/~dubin/talks/brodmann/brodmann.html`.

[2] FARABEE, M.J. On-line biology book [WWW] Estrella Mountain Community College. Disponible à : `http://www.emc.maricopa.edu/faculty/farabee/biobk/BioBookTOC.html` .

[3] FARNARIER, G. (1998) Indications urgentes de l'EEG dans la situation d'un traumatisme crânien, chez l'enfant et chez l'adulte, Neurophysiologie clinique. 28(2), pp. 103-153.

[4] FLETCHER, T.F. Cerebral hemisphere and cortex [WWW] University of Minnesota, Department of Veterinary & Biomedical Sciences. Disponible à : `http://vanat.cvm.umn.edu/` .

[5] GITTIS, A. The Measurement of Brain Waves [WWW] Westminster College, Psychology departement. Disponible à `http://www.psych.westminster.edu/psybio/BN/Labs/Brainwaves.htm`.

[6] GOLDBERGER, A.L, AMARAL, LAN, GLASS, L, HAUSSDORF J.M, IVANOV, P.C.H, MARK R.G, MIETUS J.E, MOODY G.B, PENG C-K et STANLEY, H.E. PhysioBank, PhysioToolkit, and PhysioNet : Components of a New Research Resource for Complex Physiologic Signals. Circulation 101(23) :e215-e220 [Circulation Electronic Pages ; `http://circ.ahajournals.org/cgi/content/full/101/23/e215`] ; 2000 (June 13).

[7] HAUK, O. Introduction to EEG and MEG [WWW] University of Cambridge, MRC Cognition and Brain Sciences Unit. Disponible à : `http://www.mrc-cbu.cam.ac.uk/research/eeg/eeg_intro.html` .

[8] HAUMONT, S. et DENEF, J.F. Histologie générale, architecture du système nerveux central [WWW] Université catholique de Louvain, Faculté de médecine. Disponible à : `http://www.isto.ucl.ac.be/introhg.htm`.

[9] INTERNATIONAL SOCIETY FOR NEUROFEEDBACK AND RESEARCH. Definition of neurofeedback [WWW] Internationale Society for Neurofeedback and Research. Disponible à : `http://www.isnr.org/information/index.cfm#Def`.

[10] LECOUTRE, E. Méthodes de statistiques multifactorielles, Analyse en Composantes Principales [WWW] Université Catholique de Louvain. Institut de statistiques. Disponible à : http://www.stat.ucl.ac.be/ISpersonnel/lecoutre/stats/ACP/index.html.

[11] MALLAT, S (2000) Une exploration des signaux en ondelettes. Palaiseau : Les Editions de l'Ecole Polytechnique.

[12] MacKAY, D.J.C. (2003) Information Theory, Inference, and Learning Algorithms. Cambridge University Press.

[13] MALMIVUO, J. et PLONSEY, R (1995) Bioelectromagnetism : principles and applications of bioelectric and biomagnetic fields. New York : Oxford University Press [WWW] Disponible à : http://www.bem.fi/.

[14] McCAFFREY, P. Neuroanatomy of Speech, Swallowing, and Language [WWW] California State University. Disponible à : http://www.csuchico.edu/~pmccaffrey//syllabi/ .

[15] MISITI, M, MISITI, Y, OPPENHEIM, G. et POGGI, J.M (2003) Les ondelettes et leurs applications. Paris : Lavoisier, Hermes Sciences (Traitement du signal et de l'image : traité IC2).

[16] MOHAJER, M, KAKHBOD, A. et PAZKAD, P. (2006) Tight bounds on the redundancy of Huffman codes. Dans : IEEE Information Theory Workshop. Punta del Este, Uruguay, 13-17 Mars, 2006. pp. 131-135.

[17] PEREZ-MARTIN, A. et SCHUSTER, I. et DAUZAT, M. 1er cycle : PCEM2 - Module de base 6 - Physiologie. Electroencéphalographie [WWW] Faculté de Médecine Montpellier-Nîmes. Disponible à : http://www.med.univ-montp1.fr/enseignement/cycle_1/PCEM2/mod-base/MB6_physio/Ressources_locales/tp/TP_04_EEG_2006A.pdf.

[18] PhysioNet, CHB-MIT Scalp EEG Database. [WWW] National Institue of biomedical imaging and bioengineering. Disponible sur : http://www.physionet.org/physiobank/database/chbmit/

[19] QUIAN QUIROGA, R. (2006) Evoked potentials. Dans : WEBSTER, J.G. Encyclopedia of Medical Devices and Instrumentation. 2nde éd. Hoboken : John Wiley & Sons [WWW] Disponible à : http://www.vis.caltech.edu/~rodri/papers/Evoked_Potentials.pdf .

[20] SAID, A. (2004) Introduction to arithmetic coding - theory and practice. Dans : SAYOOD, K. Lossless compression handbook. San Diego : Elsevier. pp 101-152.

[21] SHI, Y.Q. et SUN, (2010) Run-length and dictionary coding [WWW] H.Helsinki University of Technology, S-88 Signal Processing Laboratory. Disponible à : http://signal.hut.fi/introduction.php.

[22] SPRATLING, M.W. (2002) Cortical region interactions and the functional role of apical dendrites. Behavioral and Cognitive Neuroscience Reviews, 1(3), pp.219-228.

[23] SWELDENS, W. et DAUBECHIES, I. (1998) Factoring wavelet transforms into lifting steps. Journal of Fourier analysis and applications, 4(3), pp.247-269.

[24] UNSER, M. et BLU, T. (2003) Mathematical Properties of the JPEG2000 Wavelet Filters. IEEE transactions on signal processing, 12(9),pp. 1080 - 1090.

[25] VAN COPPENOLLE, F. Transmission synaptique et cellules excitables [WWW] Université de Lille. Disponible à : `http://physioanimale.univ-lille1.fr/Poly_Synapses_etudiants_FVC.pdf` .

[26] WANG, R. Lectures on Computer Image Processing and Analysis (E161) : Principal components analysis [WWW] Harvey Mudd College. Disponible à : `http://fourier.eng.hmc.edu/e161/lectures/klt/index.html`.

[27] WONGSAWAT, Y. et al (2006) Lossless multi-channel EEG compression. Dans : International Symposium on Circuits and Systems (ISCAS). Ile de Kos, Grèce, 21-24 Mai 2006. IEEE. pp 1611-1614.

[28] YANG, D.T, KYRIAKAKIS, C. et JAY KUO C.C. High-Fidelity Multichannel Audio Coding. New York : Hindawi Publishing Corporation (EURASIP Book Series on Signal Processing and Communications).

[29] SPIHT image compression, Biological/Medical Signal Compression [WWW] Disponible à : `http://www.cipr.rpi.edu/research/SPIHT/spiht3.html`.

[30] 7zip by ipavlov [WWW] Sourceforge 7zip. Disponible à : `http://sourceforge.net/projects/sevenzip/`.

Annexe A

Codage entropique

Deux exemples de codage entropique sont fournis : le premier concerne le codage de Huffman, le second le codage arithmétique.

The Huffman coding algorithm

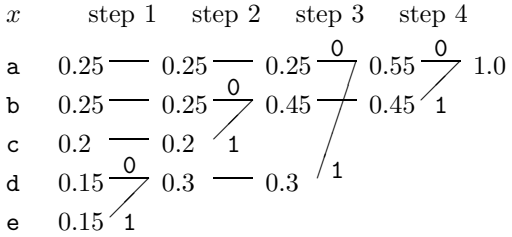
We now present a beautifully simple algorithm for finding an optimal prefix code. The trick is to construct the code *backwards* starting from the tails of the codewords; *we build the binary tree from its leaves.*

Algorithm 5.4. Huffman coding algorithm.

1. Take the two least probable symbols in the alphabet. These two symbols will be given the longest codewords, which will have equal length, and differ only in the last digit.
2. Combine these two symbols into a single symbol, and repeat.

Since each step reduces the size of the alphabet by one, this algorithm will have assigned strings to all the symbols after $|\mathcal{A}_X| - 1$ steps.

Example 5.15. Let $\mathcal{A}_X = \{\,$ a, b, c, d, e $\}$
and $\mathcal{P}_X = \{\,0.25, 0.25, 0.2, 0.15, 0.15\,\}$.



a_i	p_i	$h(p_i)$	l_i	$c(a_i)$
a	0.25	2.0	2	00
b	0.25	2.0	2	10
c	0.2	2.3	2	11
d	0.15	2.7	3	010
e	0.15	2.7	3	011

Table 5.5. Code created by the Huffman algorithm.

The codewords are then obtained by concatenating the binary digits in reverse order: $C = \{$00, 10, 11, 010, 011$\}$. The codelengths selected by the Huffman algorithm (column 4 of table 5.5) are in some cases longer and in some cases shorter than the ideal codelengths, the Shannon information contents $\log_2 1/p_i$ (column 3). The expected length of the code is $L = 2.30$ bits, whereas the entropy is $H = 2.2855$ bits. □

If at any point there is more than one way of selecting the two least probable symbols then the choice may be made in any manner – the expected length of the code will not depend on the choice.

Exercise 5.16.[3, p.105] Prove that there is no better symbol code for a source than the Huffman code.

Example 5.17. We can make a Huffman code for the probability distribution over the alphabet introduced in figure 2.1. The result is shown in figure 5.6. This code has an expected length of 4.15 bits; the entropy of the ensemble is 4.11 bits. Observe the disparities between the assigned codelengths and the ideal codelengths $\log_2 1/p_i$.

Constructing a binary tree top-down is suboptimal

In previous chapters we studied weighing problems in which we built ternary or binary trees. We noticed that balanced trees – ones in which, at every step, the two possible outcomes were as close as possible to equiprobable – appeared to describe the most efficient experiments. This gave an intuitive motivation for entropy as a measure of information content.

a_i	p_i	$\log_2 \frac{1}{p_i}$	l_i	$c(a_i)$
a	0.0575	4.1	4	0000
b	0.0128	6.3	6	001000
c	0.0263	5.2	5	00101
d	0.0285	5.1	5	10000
e	0.0913	3.5	4	1100
f	0.0173	5.9	6	111000
g	0.0133	6.2	6	001001
h	0.0313	5.0	5	10001
i	0.0599	4.1	4	1001
j	0.0006	10.7	10	1101000000
k	0.0084	6.9	7	1010000
l	0.0335	4.9	5	11101
m	0.0235	5.4	6	110101
n	0.0596	4.1	4	0001
o	0.0689	3.9	4	1011
p	0.0192	5.7	6	111001
q	0.0008	10.3	9	110100001
r	0.0508	4.3	5	11011
s	0.0567	4.1	4	0011
t	0.0706	3.8	4	1111
u	0.0334	4.9	5	10101
v	0.0069	7.2	8	11010001
w	0.0119	6.4	7	1101001
x	0.0073	7.1	7	1010001
y	0.0164	5.9	6	101001
z	0.0007	10.4	10	1101000001
–	0.1928	2.4	2	01

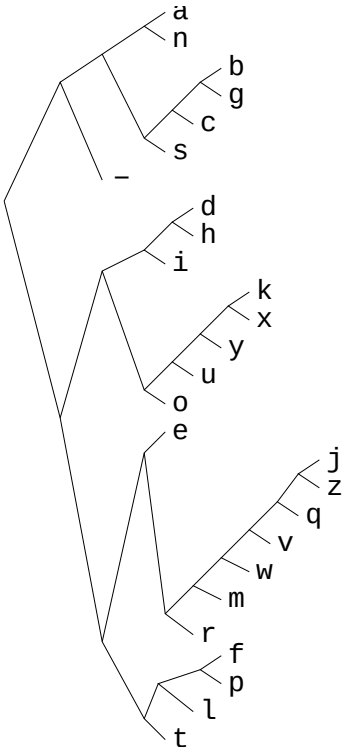


Figure 5.6. Huffman code for the English language ensemble (monogram statistics).

It is not the case, however, that optimal codes can *always* be constructed by a greedy top-down method in which the alphabet is successively divided into subsets that are as near as possible to equiprobable.

Example 5.18. Find the optimal binary symbol code for the ensemble:

$$\begin{aligned} \mathcal{A}_X &= \{ \texttt{a}, \quad \texttt{b}, \quad \texttt{c}, \quad \texttt{d}, \quad \texttt{e}, \quad \texttt{f}, \quad \texttt{g} \} \\ \mathcal{P}_X &= \{ 0.01, 0.24, 0.05, 0.20, 0.47, 0.01, 0.02 \} \end{aligned} . \tag{5.24}$$

Notice that a greedy top-down method can split this set into two subsets $\{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$ and $\{\texttt{e}, \texttt{f}, \texttt{g}\}$ which both have probability 1/2, and that $\{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$ can be divided into subsets $\{\texttt{a}, \texttt{b}\}$ and $\{\texttt{c}, \texttt{d}\}$, which have probability 1/4; so a greedy top-down method gives the code shown in the third column of table 5.7, which has expected length 2.53. The Huffman coding algorithm yields the code shown in the fourth column, which has expected length 1.97. □

a_i	p_i	Greedy	Huffman
a	.01	000	000000
b	.24	001	01
c	.05	010	0001
d	.20	011	001
e	.47	10	1
f	.01	110	000001
g	.02	111	00001

Table 5.7. A greedily-constructed code compared with the Huffman code.

▶ 5.6 Disadvantages of the Huffman code

The Huffman algorithm produces an optimal symbol code for an ensemble, but this is not the end of the story. Both the word 'ensemble' and the phrase 'symbol code' need careful attention.

Changing ensemble

If we wish to communicate a sequence of outcomes from one unchanging ensemble, then a Huffman code may be convenient. But often the appropriate

ensemble changes. If for example we are compressing text, then the symbol frequencies will vary with context: in English the letter `u` is much more probable after a `q` than after an `e` (figure 2.3). And furthermore, our knowledge of these context-dependent symbol frequencies will also change as we learn the statistical properties of the text source.

Huffman codes do not handle changing ensemble probabilities with any elegance. One brute-force approach would be to recompute the Huffman code every time the probability over symbols changes. Another attitude is to deny the option of adaptation, and instead run through the entire file in advance and compute a good probability distribution, which will then remain fixed throughout transmission. The code itself must also be communicated in this scenario. Such a technique is not only cumbersome and restrictive, it is also suboptimal, since the initial message specifying the code and the document itself are partially redundant. This technique therefore wastes bits.

The extra bit

An equally serious problem with Huffman codes is the innocuous-looking 'extra bit' relative to the ideal average length of $H(X)$ – a Huffman code achieves a length that satisfies $H(X) \leq L(C,X) < H(X)+1$, as proved in theorem 5.1. A Huffman code thus incurs an overhead of between 0 and 1 bits per symbol. If $H(X)$ were large, then this overhead would be an unimportant fractional increase. But for many applications, the entropy may be as low as one bit per symbol, or even smaller, so the overhead $L(C,X) - H(X)$ may dominate the encoded file length. Consider English text: in some contexts, long strings of characters may be highly predictable. For example, in the context '`strings_of_ch`', one might predict the next nine symbols to be '`aracters_`' with a probability of 0.99 each. A traditional Huffman code would be obliged to use at least one bit per character, making a total cost of nine bits where virtually no information is being conveyed (0.13 bits in total, to be precise). The entropy of English, given a good model, is about one bit per character (Shannon, 1948), so a Huffman code is likely to be highly inefficient.

A traditional patch-up of Huffman codes uses them to compress *blocks* of symbols, for example the 'extended sources' X^N we discussed in Chapter 4. The overhead per block is at most 1 bit so the overhead per symbol is at most $1/N$ bits. For sufficiently large blocks, the problem of the extra bit may be removed – but only at the expenses of (a) losing the elegant instantaneous decodeability of simple Huffman coding; and (b) having to compute the probabilities of all relevant strings and build the associated Huffman tree. One will end up explicitly computing the probabilities and codes for a huge number of strings, most of which will never actually occur. (See exercise 5.29 (p.103).)

Beyond symbol codes

Huffman codes, therefore, although widely trumpeted as 'optimal', have many defects for practical purposes. They *are* optimal *symbol* codes, but for practical purposes *we don't want a symbol code*.

The defects of Huffman codes are rectified by *arithmetic coding*, which dispenses with the restriction that each symbol must translate into an integer number of bits. Arithmetic coding is the main topic of the next chapter.

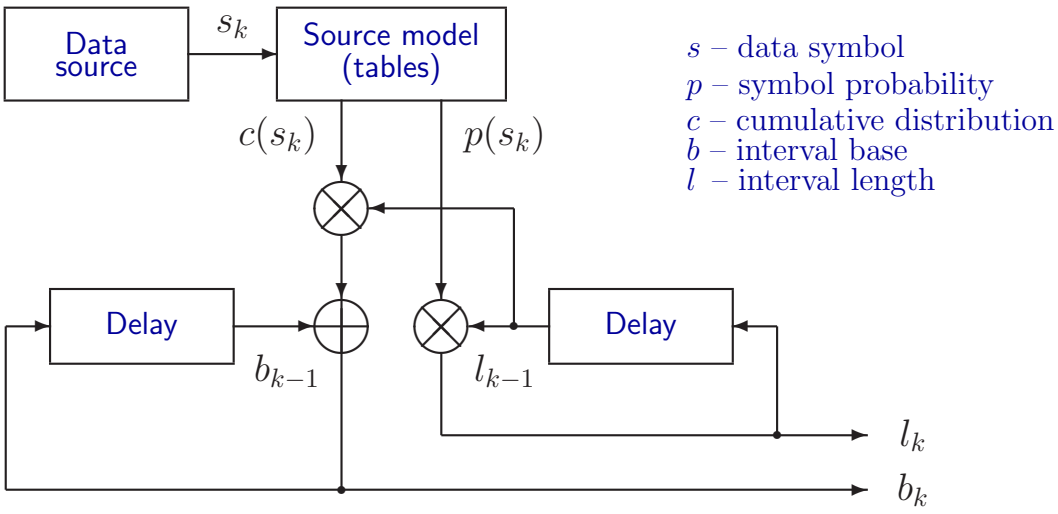


Figure 1.3: Dynamic system for updating arithmetic coding intervals.

since the first arithmetic coding papers [12, 13]. Other authors have intervals represented by their extreme points, like [base, base+length), but there is no mathematical difference between the two notations.

Example 3

◁ Let us assume that source Ω has four symbols ($M = 4$), the probabilities and distribution of the symbols are $\mathbf{p} = [\,0.2 \;\; 0.5 \;\; 0.2 \;\; 0.1\,]$ and $\mathbf{c} = [\,0 \;\; 0.2 \;\; 0.7 \;\; 0.9 \;\; 1\,]$, and the sequence of ($N = 6$) symbols to be encoded is $S = \{2, 1, 0, 0, 1, 3\}$.

Figure 1.4 shows graphically how the encoding process corresponds to the selection of intervals in the line of real numbers. We start at the top of the figure, with the interval [0, 1), which is divided into four subintervals, each with length equal to the probability of the data symbols. Specifically, interval [0, 0.2) corresponds to $s_1 = 0$, interval [0.2, 0.7) corresponds to $s_1 = 1$, interval [0.7, 0.9) corresponds to $s_1 = 2$, and finally interval [0.9, 1) corresponds to $s_1 = 3$. The next set of allowed nested subintervals also have length proportional to the probability of the symbols, but their lengths are also proportional to the length of the interval they belong to. Furthermore, they represent more than one symbol value. For example, interval [0, 0.04) corresponds to $s_1 = 0,\ s_2 = 0$, interval [0.04, 0.14) corresponds to $s_1 = 0,\ s_2 = 1$, and so on.

The interval lengths are reduced by factors equal to symbol probabilities in order to obtain code values that are uniformly distributed in the interval [0, 1) (a necessary condition for optimality, as explained in Section 1.3). For example, if 20% of the sequences start with symbol "0", then 20% of the code values must be in the interval assigned to those sequences, which can only be achieved if we assign to the first symbol "0" an interval with length equal to its probability, 0.2. The same reasoning applies to the assignment of the subinterval lengths: every occurrence of symbol "0" must result in a reduction of the interval length to 20% its current length. This way, after encoding

Iteration k	Input Symbol s_k	Interval base b_k	Interval length l_k	Decoder updated value $\tilde{v}_k = \frac{\hat{v}-b_{k-1}}{l_{k-1}}$	Output symbol $\hat{s}_k$
0	—	0	1	—	—
1	2	0.7	0.2	0.74267578125	2
2	1	0.74	0.1	0.21337890625	1
3	0	0.74	0.02	0.0267578125	0
4	0	0.74	0.004	0.1337890625	0
5	1	0.7408	0.002	0.6689453125	1
6	3	0.7426	0.0002	0.937890625	3
7	—	—	—	0.37890625	1
8	—	—	—	0.3578125	1

Table 1.2: Arithmetic encoding and decoding results for Examples 3 and 4. The last two rows show what happens when decoding continues past the last symbol.

several symbols the distribution of code values should be a very good approximation of a uniform distribution.

Equations (1.8) and (1.9) provide the formulas for the sequential computation of the intervals. Applying them to our example we obtain:

$$\begin{aligned}
\Phi_0(S) &= |\,0,\,1\,\rangle = [\,0,\,1\,)\,,\\
\Phi_1(S) &= |\,b_0 + c(2)l_0,\; p(2)l_0\,\rangle = |\,0 + 0.7 \times 1,\; 0.2 \times 1\,\rangle = [\,0.7,\,0.9\,)\,,\\
\Phi_2(S) &= |\,b_1 + c(1)l_1,\; p(1)l_1\,\rangle = |\,0.7 + 0.2 \times 0.2,\; 0.5 \times 0.2\,\rangle = [\,0.74,\,0.84\,)\,,\\
&\vdots\\
\Phi_6(S) &= |\,b_5 + c(3)l_5,\; p(3)l_5\,\rangle = |\,0.7426,\; 0.0002\,\rangle = [\,0.7426,\,0.7428\,)\,,
\end{aligned}$$

The list with all the encoder intervals is shown in the first four columns of Table 1.2. Since the intervals quickly become quite small, in Figure 1.4 we have to graphically magnify them (twice) so that we can see how the coding process continues. Note that even though the intervals are shown in different magnifications, the intervals values do not change, and the process to subdivide intervals continues in exactly the same manner. ▷

The final task in arithmetic encoding is to define a code value $\hat{v}(S)$ that will represent data sequence S. In the next section we show how the decoding process works correctly for *any* code value $\hat{v} \in \Phi_N(S)$. However, the code value cannot be provided to the decoder as a pure real number. It has to be stored or transmitted, using a conventional number representation. Since we have the freedom to choose any value in the final interval, we want to choose the values with the shortest representation. For instance, in Example 3, the shortest decimal representation comes from choosing $\hat{v} = 0.7427$, and the shortest binary representation is obtained with $\hat{v} = 0.10111110001_2 = 0.74267578125$.

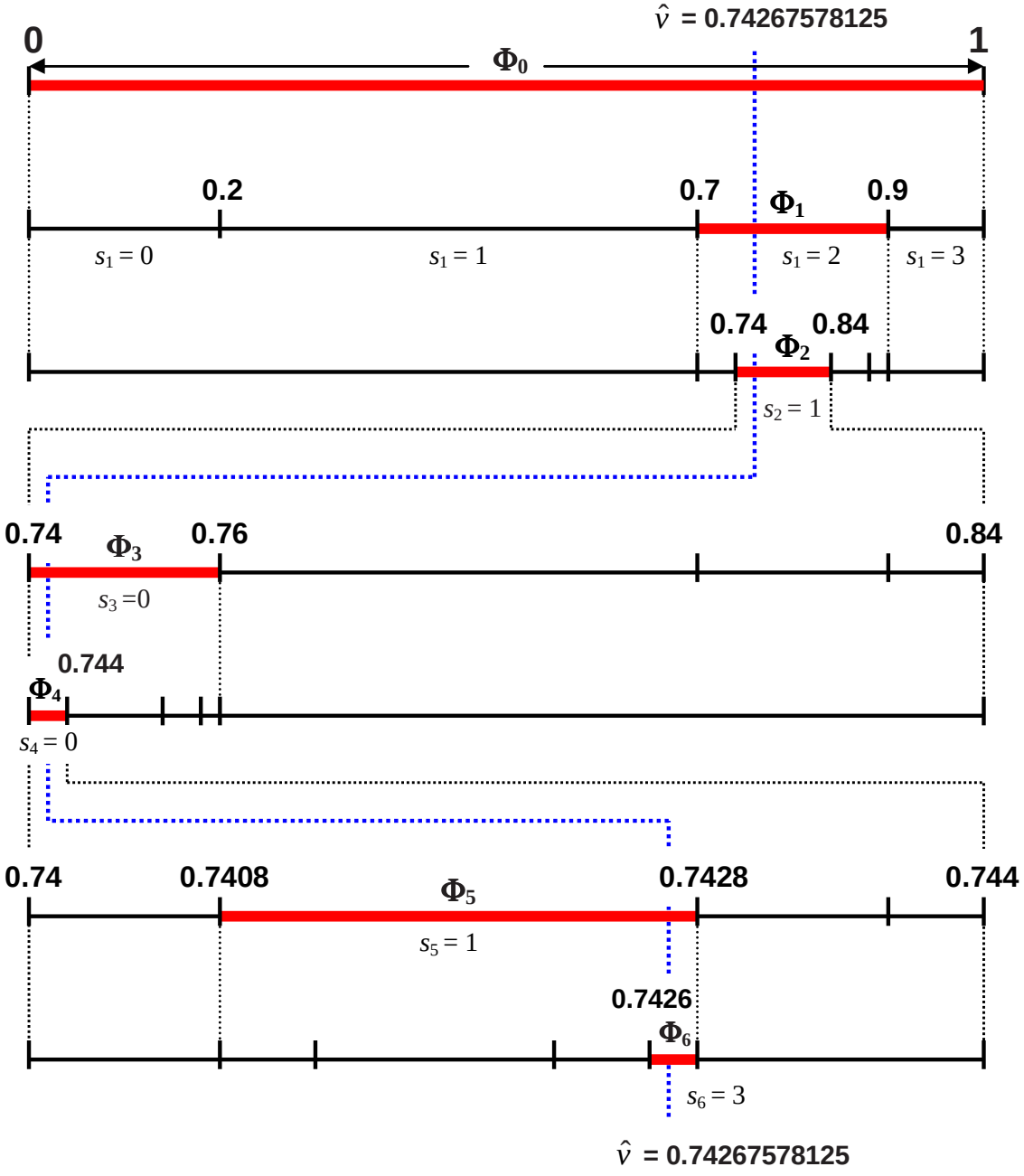


Figure 1.4: Graphical representation of the arithmetic coding process of Example 3: the interval $\Phi_0 = [0, 1)$ is divided in nested intervals according to the probability of the data symbols. The selected intervals, corresponding to data sequence $S = \{2, 1, 0, 0, 1, 3\}$ are indicated by thicker lines.

Annexe B

DSP

Cette annexe contient :
- Les caractéristiques détaillées du DSP ;
- L'architecture d'un cœur ;
- Le mapping de la mémoire ;
- Les différents modes du cache L2 ;
- Le programme en assembleur obtenu ;

TMS320C6713
FLOATING-POINT DIGITAL SIGNAL PROCESSOR

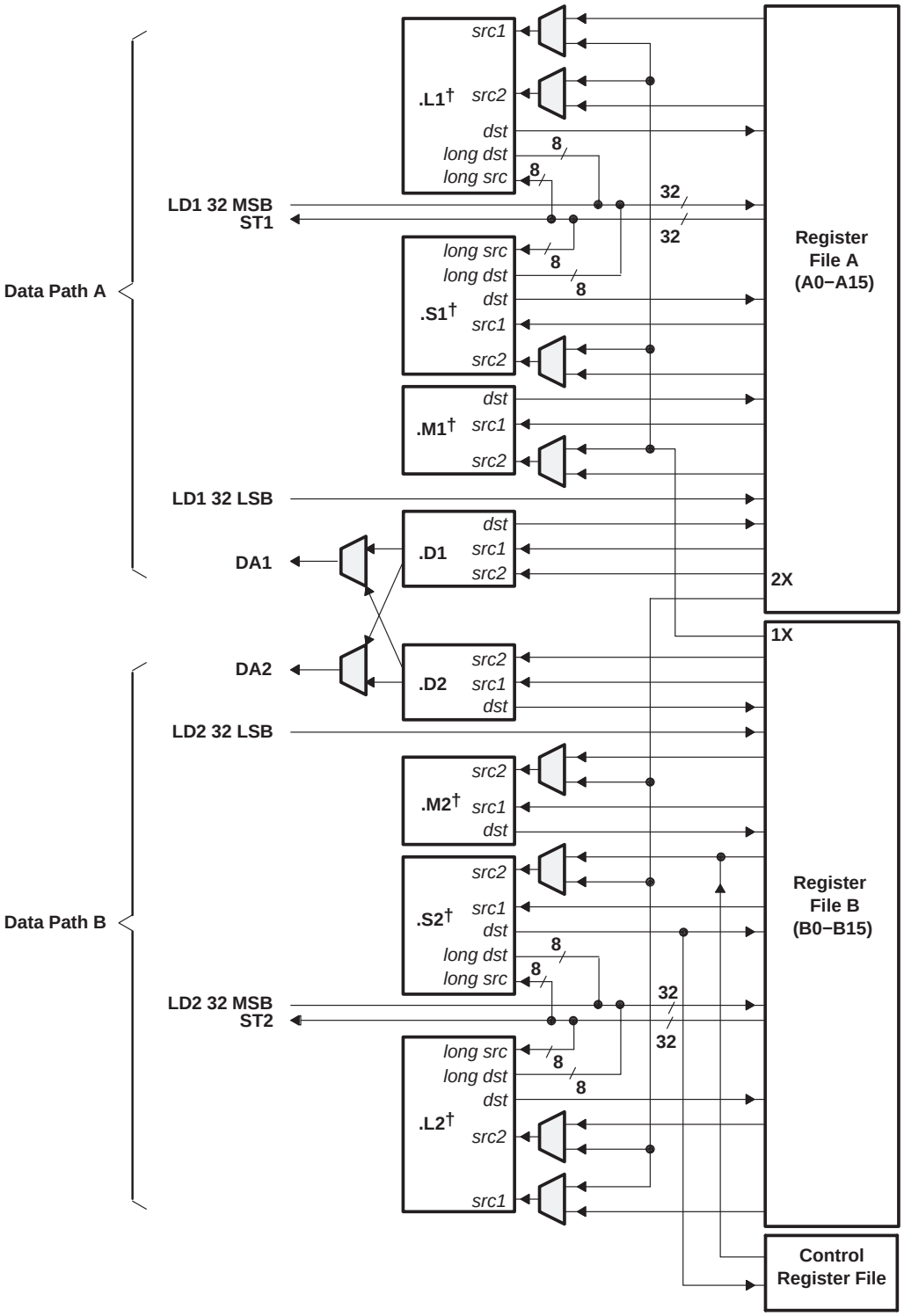
SPRS186L − DECEMBER 2001 − REVISED NOVEMBER 2005

- **Highest-Performance Floating-Point Digital Signal Processor (DSP)**
  - **Eight 32-Bit Instructions/Cycle**
  - **32/64-Bit Data Word**
  - **225-, 200-MHz (GDP), and 200-, 167-MHz (PYP) Clock Rates**
  - **4.4-, 5-, 6-Instruction Cycle Times**
  - **1800/1350, 1600/1200, and 1336/1000 MIPS/MFLOPS**
  - **Rich Peripheral Set, Optimized for Audio**
  - **Highly Optimized C/C++ Compiler**
  - **Extended Temperature Devices Available**
- **Advanced Very Long Instruction Word (VLIW) TMS320C67x™ DSP Core**
  - **Eight Independent Functional Units:**
    - **Two ALUs (Fixed-Point)**
    - **Four ALUs (Floating- and Fixed-Point)**
    - **Two Multipliers (Floating- and Fixed-Point)**
  - **Load-Store Architecture With 32 32-Bit General-Purpose Registers**
  - **Instruction Packing Reduces Code Size**
  - **All Instructions Conditional**
- **Instruction Set Features**
  - **Native Instructions for IEEE 754**
    - **Single- and Double-Precision**
  - **Byte-Addressable (8-, 16-, 32-Bit Data)**
  - **8-Bit Overflow Protection**
  - **Saturation; Bit-Field Extract, Set, Clear; Bit-Counting; Normalization**
- **L1/L2 Memory Architecture**
  - **4K-Byte L1P Program Cache (Direct-Mapped)**
  - **4K-Byte L1D Data Cache (2-Way)**
  - **256K-Byte L2 Memory Total: 64K-Byte L2 Unified Cache/Mapped RAM, and 192K-Byte Additional L2 Mapped RAM**
- **Device Configuration**
  - **Boot Mode: HPI, 8-, 16-, 32-Bit ROM Boot**
  - **Endianness: Little Endian, Big Endian**
- **32-Bit External Memory Interface (EMIF)**
  - **Glueless Interface to SRAM, EPROM, Flash, SBSRAM, and SDRAM**
  - **512M-Byte Total Addressable External Memory Space**
- **Enhanced Direct-Memory-Access (EDMA) Controller (16 Independent Channels)**
- **16-Bit Host-Port Interface (HPI)**
- **Two McASPs**
  - **Two Independent Clock Zones Each (1 TX and 1 RX)**
  - **Eight Serial Data Pins Per Port: Individually Assignable to any of the Clock Zones**
  - **Each Clock Zone Includes:**
    - **Programmable Clock Generator**
    - **Programmable Frame Sync Generator**
    - **TDM Streams From 2-32 Time Slots**
    - **Support for Slot Size: 8, 12, 16, 20, 24, 28, 32 Bits**
    - **Data Formatter for Bit Manipulation**
  - **Wide Variety of I2S and Similar Bit Stream Formats**
  - **Integrated Digital Audio Interface Transmitter (DIT) Supports:**
    - **S/PDIF, IEC60958-1, AES-3, CP-430 Formats**
    - **Up to 16 transmit pins**
    - **Enhanced Channel Status/User Data**
  - **Extensive Error Checking and Recovery**
- **Two Inter-Integrated Circuit Bus (I2C Bus™) Multi-Master and Slave Interfaces**
- **Two Multichannel Buffered Serial Ports:**
  - **Serial-Peripheral-Interface (SPI)**
  - **High-Speed TDM Interface**
  - **AC97 Interface**
- **Two 32-Bit General-Purpose Timers**
- **Dedicated GPIO Module With 16 pins (External Interrupt Capable)**
- **Flexible Phase-Locked-Loop (PLL) Based Clock Generator Module**
- **IEEE-1149.1 (JTAG†) Boundary-Scan-Compatible**
- **208-Pin PowerPAD™ Plastic (Low-Profile) Quad Flatpack (PYP)**
- **272-BGA Packages (GDP)**
- **0.13-μm/6-Level Copper Metal Process**
  - **CMOS Technology**
- **3.3-V I/Os, 1.2-V‡ Internal (GDP & PYP)**

Please be aware that an important notice concerning availability, standard warranty, and use in critical applications of Texas Instruments semiconductor products and disclaimers thereto appears at the end of this data sheet.

TMS320C67x and PowerPAD are trademarks of Texas Instruments.
I2C Bus is a trademark of Philips Electronics N.V. Corporation
All trademarks are the property of their respective owners.
† IEEE Standard 1149.1-1990 Standard-Test-Access Port and Boundary Scan Architecture.
‡ These values are compatible with existing 1.26−V designs.

CPU (DSP core) description (continued)



† In addition to fixed-point instructions, these functional units execute floating-point instructions.

Figure 1. TMS320C67x™ CPU (DSP Core) Data Paths

memory map summary

Table 3 shows the memory map address ranges of the C6713 device.

Table 3. TMS320C6713 Memory Map Summary

MEMORY BLOCK DESCRIPTION	BLOCK SIZE (BYTES)	HEX ADDRESS RANGE
Internal RAM (L2)	192K	0000 0000 – 0002 FFFF
Internal RAM/Cache	64K	0003 0000 – 0003 FFFF
Reserved	24M – 256K	0004 0000 – 017F FFFF
External Memory Interface (EMIF) Registers	256K	0180 0000 – 0183 FFFF
L2 Registers	128K	0184 0000 – 0185 FFFF
Reserved	128K	0186 0000 – 0187 FFFF
HPI Registers	256K	0188 0000 – 018B FFFF
McBSP 0 Registers	256K	018C 0000 – 018F FFFF
McBSP 1 Registers	256K	0190 0000 – 0193 FFFF
Timer 0 Registers	256K	0194 0000 – 0197 FFFF
Timer 1 Registers	256K	0198 0000 – 019B FFFF
Interrupt Selector Registers	512	019C 0000 – 019C 01FF
Device Configuration Registers	4	019C 0200 – 019C 0203
Reserved	256K − 516	019C 0204 – 019F FFFF
EDMA RAM and EDMA Registers	256K	01A0 0000 – 01A3 FFFF
Reserved	768K	01A4 0000 – 01AF FFFF
GPIO Registers	16K	01B0 0000 – 01B0 3FFF
Reserved	240K	01B0 4000 – 01B3 FFFF
I2C0 Registers	16K	01B4 0000 – 01B4 3FFF
I2C1 Registers	16K	01B4 4000 – 01B4 7FFF
Reserved	16K	01B4 8000 – 01B4 BFFF
McASP0 Registers	16K	01B4 C000 – 01B4 FFFF
McASP1 Registers	16K	01B5 0000 – 01B5 3FFF
Reserved	160K	01B5 4000 – 01B7 BFFF
PLL Registers	8K	01B7 C000 – 01B7 DFFF
Reserved	264K	01B7 E000 – 01BB FFFF
Emulation Registers	256K	01BC 0000 – 01BF FFFF
Reserved	4M	01C0 0000 – 01FF FFFF
QDMA Registers	52	0200 0000 – 0200 0033
Reserved	16M − 52	0200 0034 – 02FF FFFF
Reserved	720M	0300 0000 – 2FFF FFFF
McBSP0 Data Port	64M	3000 0000 – 33FF FFFF
McBSP1 Data Port	64M	3400 0000 – 37FF FFFF
Reserved	64M	3800 0000 – 3BFF FFFF
McASP0 Data Port	1M	3C00 0000 – 3C0F FFFF
McASP1 Data Port	1M	3C10 0000 – 3C1F FFFF
Reserved	1G + 62M	3C20 0000 – 7FFF FFFF
EMIF CE0†	256M	8000 0000 – 8FFF FFFF
EMIF CE1†	256M	9000 0000 – 9FFF FFFF
EMIF CE2†	256M	A000 0000 – AFFF FFFF
EMIF CE3†	256M	B000 0000 – BFFF FFFF
Reserved	1G	C000 0000 – FFFF FFFF

† The number of EMIF address pins (EA[21:2]) limits the maximum addressable memory (SDRAM) to 128MB per CE space.

L2 memory structure expanded

Figure 2 shows the detail of the L2 memory structure.

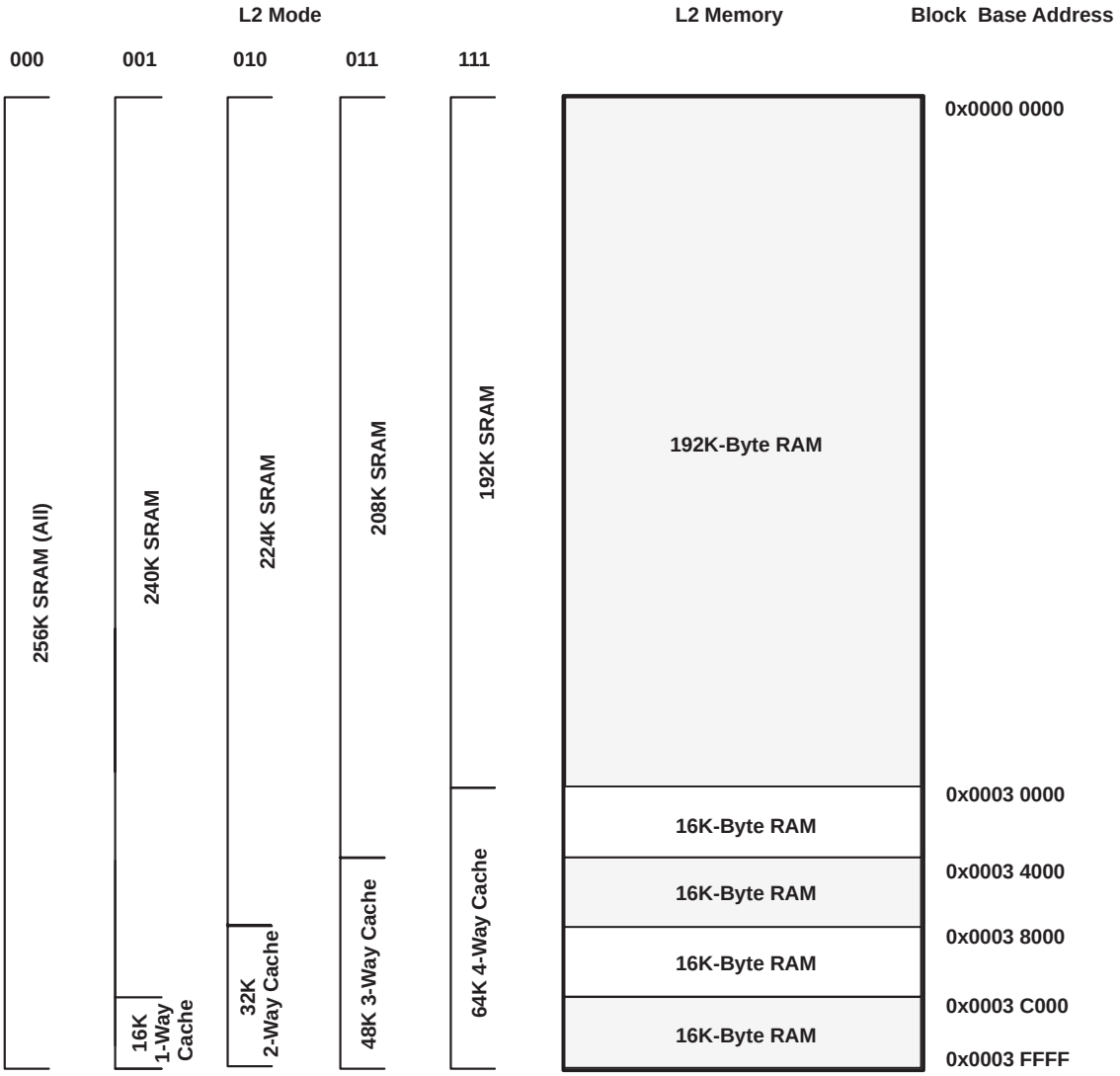


Figure 2. L2 Memory Configuration

B.1 Code assembleur

```
int i;
float pt[8];
float coeff25=0.25;
float coeff05=-0.5;
void main()
{

  //Extension du signal par symétrie
  pt[0]=pt[0]-(pt[1]+pt[3])*coeff25;

00000000 02069C2B            MVK.S2        0x0d38,B4
00000004 01869828  ||        MVK.S1        0x0d30,A3
00000008 0200006B            MVKH.S2       0x0000,B4
0000000C 01800068  ||        MVKH.S1       0x0000,A3
00000010 021002E7            LDW.D2T2      *+B4[0x0],B4
00000014 018C0264  ||        LDW.D1T1      *+A3[0x0],A3
00000018 0280016E            LDW.D2T2      *+DP[0x1],B5
0000001C 00004000            NOP           3
00000020 01907218            ADDSP.L1X     A3,B4,A3
00000024 0206962A            MVK.S2        0x0d2c,B4
00000028 0200006A            MVKH.S2       0x0000,B4
0000002C 021002E6            LDW.D2T2      *+B4[0x0],B4
00000030 01947E00            MPYSP.M1X     A3,B5,A3
00000034 00004000            NOP           3
00000038 020C92B8            SUBSP.L1X     B4,A3,A4
0000003C 01869628            MVK.S1        0x0d2c,A3
00000040 01800068            MVKH.S1       0x0000,A3
00000044 00000000            NOP
00000048 020C0274            STW.D1T1      A4,*+A3[0x0]
0000004C 00002000            NOP           2

for (i=1;i <=3; i++)
00000050 020000AA            MVK.S2        0x0001,B4
00000054 0200007E            STW.D2T2      B4,*+DP[0x0]
00000058 00002000            NOP           2
0000005C 00106ADA            CMPLT.L2      3,B4,B0
00000060 20001490     [ B0]  B.S1          DW$L$_main$2$E
00000064 00008000            NOP           5

000000E4 0200006E            LDW.D2T2      *+DP[0x0],B4
000000E8 00006000            NOP           4
000000EC 0210205A            ADD.L2        1,B4,B4
```

```
000000F0 0200007E            STW.D2T2      B4,*+DP[0x0]
000000F4 00002000            NOP           2
000000F8 00106ADA            CMPLT.L2      3,B4,B0
000000FC 3FFFF110     [!B0]  B.S1          L1
00000100 00008000            NOP           5

pt[i*2]=pt[2*i]-(pt[i*2+1]+pt[i*2-1])*coeff25;

00000068          DW$L$_main$2$B, L1:
00000068 0290807A            ADD.L2        B4,B4,B5
0000006C 02944CA2            SHL.S2        B5,0x2,B5
00000070 0206962B            MVK.S2        0x0d2c,B4
00000074 0310807A  ||        ADD.L2        B4,B4,B6
00000078 03184CA3            SHL.S2        B6,0x2,B6
0000007C 01869628  ||        MVK.S1        0x0d2c,A3
00000080 0318805B            ADD.L2        4,B6,B6
00000084 0200006B  ||        MVKH.S2       0x0000,B4
00000088 029489C3  ||        SUB.D2        B5,0x4,B5
0000008C 01800068  ||        MVKH.S1       0x0000,A3
00000090 01987079            ADD.L1X       A3,B6,A3
00000094 0214807A  ||        ADD.L2        B4,B5,B4
00000098 018C0265            LDW.D1T1      *+A3[0x0],A3
0000009C 021002E6  ||        LDW.D2T2      *+B4[0x0],B4
000000A0 0480006E            LDW.D2T2      *+DP[0x0],B9
000000A4 0400016E            LDW.D2T2      *+DP[0x1],B8
000000A8 0386962A            MVK.S2        0x0d2c,B7
000000AC 0380006A            MVKH.S2       0x0000,B7
000000B0 01907218            ADDSP.L1X     A3,B4,A3
000000B4 021D3E42            ADDAD.D2      B7,B9,B4
000000B8 021002E6            LDW.D2T2      *+B4[0x0],B4
000000BC 0086962A            MVK.S2        0x0d2c,B1
000000C0 01A07E00            MPYSP.M1X     A3,B8,A3
000000C4 0124005A            MV.L2         B9,B2
000000C8 0080006A            MVKH.S2       0x0000,B1
000000CC 00000000            NOP
000000D0 018C92B8            SUBSP.L1X     B4,A3,A3
000000D4 02045E42            ADDAD.D2      B1,B2,B4
000000D8 00002000            NOP           2
000000DC 019002F4            STW.D2T1      A3,*+B4[0x0]
000000E0 00002000            NOP           2

 //Extension du signal par symétrie
  pt[1]=pt[1]-coeff05*pt[2];
  pt[3]=pt[3]-coeff05*(pt[0]+pt[2]);
```

```
 00000104          L2, DW$L$_main$2$E:
00000104 02069A2A            MVK.S2        0x0d34,B4
00000108 0280026F            LDW.D2T2      *+DP[0x2],B5
0000010C 0200006A  ||        MVKH.S2       0x0000,B4
00000110 021002E6            LDW.D2T2      *+B4[0x0],B4
00000114 01869828            MVK.S1        0x0d30,A3
00000118 01800068            MVKH.S1       0x0000,A3
0000011C 018C0264            LDW.D1T1      *+A3[0x0],A3
00000120 0486982A            MVK.S2        0x0d30,B9
00000124 02148E02            MPYSP.M2      B4,B5,B4
00000128 0480006A            MVKH.S2       0x0000,B9
0000012C 00002000            NOP           2
00000130 029072BA            SUBSP.L2X     A3,B4,B5
00000134 00004000            NOP           3
00000138 02A402F6            STW.D2T2      B5,*+B9[0x0]
0000013C 00002000            NOP           2
00000140 01869629            MVK.S1        0x0d2c,A3
00000144 02069A2A  ||        MVK.S2        0x0d34,B4
00000148 01800069            MVKH.S1       0x0000,A3
0000014C 0200006A  ||        MVKH.S2       0x0000,B4
00000150 018C0265            LDW.D1T1      *+A3[0x0],A3
00000154 021002E6  ||        LDW.D2T2      *+B4[0x0],B4
00000158 0280026E            LDW.D2T2      *+DP[0x2],B5
0000015C 04869C28            MVK.S1        0x0d38,A9

00000160 04800068            MVKH.S1       0x0000,A9
00000164 00000000            NOP
00000168 01907218            ADDSP.L1X     A3,B4,A3
0000016C 04069C2A            MVK.S2        0x0d38,B8
00000170 0400006A            MVKH.S2       0x0000,B8
00000174 022002E6            LDW.D2T2      *+B8[0x0],B4
00000178 01947E00            MPYSP.M1X     A3,B5,A3
0000017C 00004000            NOP           3
00000180 020C92B8            SUBSP.L1X     B4,A3,A4
00000184 00004000            NOP           3
00000188 02240274            STW.D1T1      A4,*+A9[0x0]
0000018C 00002000            NOP           2

  for (i=2;i<=3;i++)
  00000190 01800128            MVK.S1        0x0002,A3
00000194 0180007C            STW.D2T1      A3,*+DP[0x0]
00000198 00002000            NOP           2
0000019C 000C7ADA            CMPLT.L2X     3,A3,B0
000001A0 20001910     [ B0]  B.S1          DW$L$_main$4$E
```

```
000001A4 020C105A            MV.L2X        A3,B4
000001A8 00006000            NOP           4

00000248 0200006E            LDW.D2T2      *+DP[0x0],B4
0000024C 00006000            NOP           4
00000250 0210205A            ADD.L2        1,B4,B4
00000254 0200007E            STW.D2T2      B4,*+DP[0x0]
00000258 00002000            NOP           2
0000025C 00106ADA            CMPLT.L2      3,B4,B0
00000260 3FFFE990     [!B0]  B.S1          L3
00000264 00008000            NOP           5

pt[2*i+1]=pt[2*i+1]-coeff05*(pt[(i-1)*2]+pt[(i-2)*2]);
000001AC          DW$L$_main$4$B, L3:
000001AC 0206962B            MVK.S2        0x0d2c,B4
000001B0 0290005B  ||        MV.L2         B4,B5
000001B4 03100942  ||        MV.D2         B4,B6
000001B8 03186CA3            SHL.S2        B6,0x3,B6
000001BC 00000000  ||        NOP
000001C0 02946CA3            SHL.S2        B5,0x3,B5
000001C4 01869628  ||        MVK.S1        0x0d2c,A3
000001C8 029609C3            SUB.D2        B5,0x10,B5
000001CC 031B005B  ||        SUB.L2        B6,8,B6
000001D0 0200006B  ||        MVKH.S2       0x0000,B4
000001D4 01800069  ||        MVKH.S1       0x0000,A3
000001D8 00000001  ||        NOP
000001DC 00000000  ||        NOP
000001E0 0294807B            ADD.L2        B4,B5,B5
000001E4 01987079  ||        ADD.L1X       A3,B6,A3
000001E8 0480006E  ||        LDW.D2T2      *+DP[0x0],B9
000001EC 031402E7            LDW.D2T2      *+B5[0x0],B6
000001F0 018C0264  ||        LDW.D1T1      *+A3[0x0],A3
000001F4 0400026E            LDW.D2T2      *+DP[0x2],B8
000001F8 04869628            MVK.S1        0x0d2c,A9
000001FC 04800068            MVKH.S1       0x0000,A9

00000200 0225207A            ADD.L2        B9,B9,B4
00000204 02104CA3            SHL.S2        B4,0x2,B4
00000208 02987218  ||        ADDSP.L1X     A3,B6,A5
0000020C 0210805A            ADD.L2        4,B4,B4
00000210 01913078            ADD.L1X       A9,B4,A3
00000214 020C0265            LDW.D1T1      *+A3[0x0],A4
00000218 0380006E  ||        LDW.D2T2      *+DP[0x0],B7
0000021C 01A0BE00            MPYSP.M1X     A5,B8,A3
```

```
00000220 0106962A            MVK.S2        0x0d2c,B2
00000224 0100006A            MVKH.S2       0x0000,B2
00000228 00000000            NOP
0000022C 029CE07B            ADD.L2        B7,B7,B5
00000230 018C8238  ||        SUBSP.L1      A4,A3,A3
00000234 02944CA2            SHL.S2        B5,0x2,B5
00000238 0294805A            ADD.L2        4,B5,B5
0000023C 0214407A            ADD.L2        B2,B5,B4
00000240 019002F4            STW.D2T1      A3,*+B4[0x0]
00000244 00002000            NOP           2
```