Réalisation d'une bibliothèque C de fonctions ondelettes

Mikaël Bourges-Sevenier

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Réalisation d'une bibliothèque C de fonctions ondelettes.

Mikaël Bourges-Sévenier

N° 2362

Septembre 1994

PROGRAMME 4

ISSN 0249-6399

INRIA
RENNES

Réalisation d'une bibliothèque C de fonctions ondelettes.

Mikaël Bourges-Sévenier*

Programme 4 — Robotique, image et vision
Projet SIAMES

Rapport de recherche n° 2362 — Septembre 1994 — 108 pages

Résumé : Ce rapport s'adresse aux personnes désireuses de connaitre la théorie des ondelettes sans être de pures mathématiciens. Il décrit la théorie des ondelettes tout en mettant l'accent sur les différents types d'ondelettes et leurs propriétés respectives ainsi que les transformées ondelettes. Une librairie, appelée WaveLib, décrite dans ce rapport permet à l'utilisateur la manipulation et l'utilisation des ondelettes et leurs transformées de manière transparente. Des exemples d'utilisation sont donnés afin de montrer la simplicité d'utilisation de cette librairie.

Mots-clé : Ondelettes, Transformées ondelettes, Analyse multirésolution, Analyse temps-fréquence, Langage C, Matlab.

(Abstract: pto)

*. bourges@irisa.fr

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : (33) 99 84 71 00 – Télécopie : (33) 99 84 71 71

Creating a C library of wavelets functions.

Abstract: This report is destinated to people, not necessary mathematicians, willing to know about wavelets' theory. It describes the theory of wavelets while putting an emphasis on different types of wavelets and their respectives properties as well as on wavelet transforms. WaveLib, the library described in this document, allows the user to manipulate and to use wavelets and their transforms in transparent manner. Some examples are given in order to show the facility of using this library.

Key-words: Wavelets, Wavelet transforms, multiresolution analysis, time-frequency analysis, C language, Matlab.

Remerciements

J'aimerais remercier toutes les personnes qui m'ont aidé durant ces six mois de stage. Tout d'abord, j'aimerais remercier Bruno Arnaldi pour m'avoir accueilli dans l'équipe SIAMES de l'IRISA. A Kadi Bouatouch et Sumant Pattanaik, mes deux responsables, je souhaiterais exprimer ma gratitude pour m'avoir conseillé, pour m'avoir guidé tout au long du stage et pour toutes les discussions enrichissantes que nous avons eues.

Mes remerciements vont également à mes amis Jean-Luc Nougaret et Eric Nguyen. Avec Jean-Luc j'ai été initié aux réseaux d'ondelettes et avec Eric, nous avons beaucoup échangé d'idées quant aux ondelettes et leurs applications en compression d'images.

Durant le stage, j'ai été amené à récupérer des données par ftp de divers serveurs. Toutefois, il n'en a pas toujours été ainsi et quelques personnes m'ont procuré des brides de code intéressant qui m'ont fait avancer plus vite. Il s'agit de Jean-Marc Lina, laboratoire de Physique nucléaire de l'Université de Montréal, Eric Veum du Missouri, Yves Misiti Université Paris Sud, Geert Uytterhoeven pour sa thèse sur les multiwavelets et Marten D. van der Laan sans qui je n'aurais pu recevoir la Toolbox de Jeffrey C. Kantor qui fut le point de départ de mon travail sous Matlab.

A tous ceux qui m'ont soutenu et supporté Eric Maisel, Eric Languénou, Hughues Labbé, Michaël Sauvée, Franck Bachelin, Thierry Filleul, Françoise Fouesnel.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

Durant les 15 dernières années, les décompositions en bandes de fréquences ont trouvé de nombreuses applications en traitement du signal. L'expansion d'une fonction sur quelques bandes de fréquences donne une représentation intermédiaire entre une représentation spatiale (ou temporelle) et une représentation de Fourier. En analyse harmonique, ce type de transformation est apparu dans les travaux de Littlewood et Payley dans les années 1930. De nombreuses recherches ont convergé récemment pour la création d'une nouvelle décomposition appelée *Transformations Ondelettes*. Pour mieux comprendre ce modèle, nous allons en décrire les motivations.

La transformée intermédiaire entre une représentation spatiale et une représentation fréquentielle est celle de Fourier. La transformation de Fourier (TF) d'une fonction $f(x)$ donne une mesure des irrégularités (hautes fréquences du signal mais cette information n'est pas localisée car la transformée de Fourier $\hat{f}(\omega)$ est définie par une intégrale couvrant tout le domaine spatial. Pour localiser l'information donnée par la TF, Gabor introduisit la *Transformée de Fourier Glissante* (TFG) en utilisant une fenêtre spatiale $g(x)$ dans l'intégrale de Fourier. La fenêtre est translatée sur tout le domaine spatial du signal. Ainsi, la TFG mesure autour d'un point u, l'amplitude de la composante sinusoidale de fréquence ω. La problème de cette représentation est d'utiliser une fenêtre de taille fixe couvrant le domaine spatio-fréquentiel. Or, on souhaiterait avoir une fenêtre qui s'adapte en fonction des irrégularités du signal. Il s'agit de la *Transformation Ondelette*.

Morlet et Grossmann ont formalisé de nombreux concepts introduits, pour certains, dès le début du siècle. Ils ont ouvert la porte à un champ vaste d'applications et à de nouveaux résultats très importants. Actuellement, il serait difficile d'énumérer tous les domaines d'application de cette théorie tant ils sont nombreux, que ce soit en traitement du signal (bancs de filtres), en compression (parole, images, scènes tri-dimensionnelle), en chime, en physique, en psychophisiologie, en astronomie, etc.

Malgré les recherches abondantes dans ce domaine, peu de code a été publié. Ici, dans l'équipe SIAMES de l'IRISA de Rennes, nous désirions utiliser les ondelettes pour des problèmes d'analyse numérique. En effet, en synthèse d'image réaliste il faut souvent résoudre une équation intégrale du deuxième type (voir la méthode de Galerkin, chapitre 4). En discrétisant cette équation, il ne reste plus qu'à résoudre

une système linéaire d'équations. Afin de réduire le nombre d'opérations, on voulait passer par les ondelettes et transformer la matrice du système en une matrice bande. La réalisation de telles opérations nécessitait la mise en place de code de transformations ondelettes. De plus, nous souhaitions étudier l'effet de divers filtres sur cette compression. Toutes ces idées ont donné lieu à la réalisation d'une bibliothèque de fonctions en langage C avec une interface pour Matlab.

Dans une première partie, nous rappellerons la théorie sous-jacente aux transformées ondelettes. Nous montrerons comment, à partir de l'analyse multi-résolution, on peut générer les fonctions d'échelle ϕ et les fonctions ondelettes ψ. Puis nous parlerons des diverses transformées ondelettes et de leurs implémentations. La théorie des ondelettes ayant beaucoup de racines en théorie des bancs de filtres, nous effectuerons une analogie avec l'analyse multirésolution. Enfin, nous parlerons de l'analyse temps-fréquence et nous donnerons un pseudo-code pour réaliser les transformées ondelettes rapides. Dans la seconde partie, nous expliquerons comment on calcule les différents filtres : des contraintes à résoudre à l'implémentation informatique. La dernière partie sera consacrée à la mise en oeuvre de notre bibliothèque Wavelib 1.0 . A travers des exemples en C et en Matlab, nous montrerons comment utiliser le logiciel.

Notations

Nous présenterons de nouvelles notations au fur et à mesure du texte. Ici, nous rappellons des notations de base. On note L^2 l'ensemble des fonctions de $\mathbb{R}$ de carré intégrable et l^2 l'ensemble des séries de carré intégrable. La notation $f(\bar{x})$ signifie conjugué complexe de la fonction f, par contre $\bar{V}$, où V est un espace, signifie l'adhérence de l'espace V.

- **Norme** l^p d'une séquence x par $\|x\|_p = \sqrt[p]{\sum_i x_i^p}$
- **Convolution** d'une fonction $f(x)$ par une fonction $g(x)$

$$f \star g(u) = \int_{-\infty}^{+\infty} f(x)\overline{g(u-x)}dx$$

- **Produit scalaire** de deux fonctions de carré intégrable $f, g \in \mathrm{L}^2(\mathbb{R})$

$$< f, g >= \int_{-\infty}^{+\infty} f(x)\overline{g(x)}dx,$$

- **Transformée de Fourier** d'une fonction $f \in \mathrm{L}^2(\mathbb{R})$

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(x)e^{-i\omega x}dx.$$

- **Transformée en** z d'une fonction échantillonnée $h \in \mathrm{l}^2(\mathbb{Z})$ ($h = \{h_n, \quad n \in \mathbb{Z}\}$)

$$\hat{f}(z) = \sum_{-\infty}^{+\infty} h_n z^{-n}.$$

  Si on remplace $z = e^{-in\omega}$ on a la transformée de Fourier discrète.

- **Formule sommatoire de Poisson** sous 2 formes:

$$\sum_l f(x-l) = \sum_k \hat{f}(2k\pi)e^{i2k\pi},$$

  et

$$\sum_l < f, g(\cdot - l) > e^{-i\omega l} = \sum_k \hat{f}(\omega + k2\pi)\overline{\hat{g}(\omega + k2\pi)}.$$

- Si aucune borne n'est indiquée sous les sommes, il faut comprendre $\in \mathbb{Z}$. De même, pour les intégrales, pas de bornes signifie de $-\infty$ à $+\infty$.
- **Base de Riesz.** Un ensemble dénombrable $\{f_n\}$ appartenant à un espace de Hilbert est une *base de Riesz* si chaque élément f de l'espace peut être écrit de manière unique $f = \sum_n c_n f_n$, et il existe deux constantes positives A et B telles que

$$A\|f\|_2^2 \leq \sum_n |c_n|^2 \leq B\|f\|_2^2.$$

- **Fonction de Dirac** $\delta_k(x)$ vaut 1 si $x = k$, 0 sinon,

- **Fonction porte** $\chi_{[a,b]}$ vaut 1 sur $[a,b]$, 0 ailleurs,
- **Notation** $\lfloor N/2 \rfloor$**.** signifie qu'on prend l'entier le plus proche de $\frac{N}{2}$ sans jamais le dépasser.
- **Notation** $f^{(l)}$**.** signifie qu'il s'agit de la dérivée $l^{\text{ième}}$ de la fonction f,
- **Fonction sinus cardinal** $\text{sinc}(x) = \frac{sin\pi x}{\pi x}$.

On note $\hat{f}$ la transformée de Fourier d'une fonction $f(x)$ et H la transformée de Fourier du filtre (fonction discrète) h_k.

Abbréviations utilisées

- CWT Continuous Wavelet Transform,
- DWT Discrete Wavelet Transform,
- FWT Fast Wavelet Transform,
- WP Wavelet Packets,
- FFT Fast Fourier Transform,
- xD x dimensions,

Chapitre 2

Théorie sur les ondelettes

La théorie des ondelettes est issue de nombreux travaux en traitement du signal et en compression d'images. En traitement du signal, la théorie des bancs de filtres a donné lieu au fameux schéma de décomposition-reconstruction de Stéphane Mallat [45]. En compression d'images, l'algorithme de décomposition pyramidale d'une image a servi de base pour l'analyse multirésolution. Mais avant tout, qu'est-ce qu'une ondelette? Et, à quoi ça sert ?

L'idée de l'analyse ondelette est de décomposer un signal, une fonction, ..., sur une base de fonctions d'un sous-espace ayant des propriétés bien déterminées. En particulier, on cherche à analyser un signal en cherchant où peuvent être localisées (dans le temps) les irrégularités du signal (i.e. les variations brusques dans le signal ou hautes fréquences). Une ondelette ϕ ou ψ est donc une fonction dont les versions translatées et les dilatées forment une base d'un sous-espace. On appelle ϕ la fonction d'échelle qui permet d'avoir une version filtrée passe-bas du signal original, et ψ la fonction ondelette permettant d'avoir les détails entre deux versions consécutives fitrées par ϕ . Ces fonctions pavent le plan temps-fréquence sous forme de petits rectangles de même surface. On réalise alors un diagramme temps-fréquence, en couleurs ou en niveaux de gris, où on colore un rectangle en fonction de l'énergie des coefficients projetés sur l'ondelette localisée dans celui-ci. On notera $\phi_{jk}(x) = 2^{-j/2}\phi(2^{-j}x - k)$ la version de ϕ à la résolution j, translatées de k, de même pour ψ ; on voit donc que la résolution j n'est autre qu'un paramètre d'échelle dilatant ϕ : plus j est petit, plus l'ondelette à un support étroit, plus j est grand, plus l'ondelette à un support large.

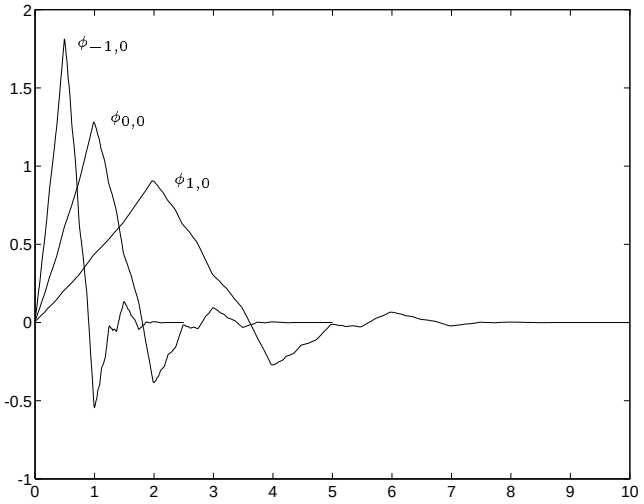


FIG. 2.1 - : *Versions dilatées de ϕ .*

Dans un premier temps, nous rappellerons ce qu'est l'analyse multirésolution et comment on génère les fonctions ondelettes ϕ et ψ. Puis, nous expliquerons ce que sont les diverses transformées ondelettes. Enfin, nous effectuerons une analogie avec la théorie des bancs de filtres et nous donnerons les algorithmes de transformées ondelettes rapides d'un vecteur d'échantillons.

2.1 Analyse multirésolution

Il existe deux façons d'introduire les ondelettes : l'une à travers la transformation ondelette continue, l'autre à travers l'analyse multirésolution. Ici, nous présentons les concepts de l'analyse multirésolution d'après Ingrid Daubechies [24]. Aucune preuve ne sera donnée, pour de plus amples détails voir les articles [47], [48].

Pour mieux comprendre ce que l'on appelle *Analyse Multi-résolution*, prenons cet exemple : quelle est la longueur de la côte bretonne ? Nous pouvons prendre un globe, avec une règle on calcule la distance et avec l'échelle de la carte on en déduit la longueur de la côte. Il s'agit d'une vision très grossière de l'allure de la côte. Prenons maintenant une carte de France, on répète les mêmes opérations. On aura alors un aperçu beaucoup plus précis de la côte et de sa longueur. Maintenant on prend une carte de Bretagne, là encore notre vision de la côte sera bien plus fine. Finalement, on peut se rendre sur le terrain et on aura la vision la plus fine qui soit pour calculer la longueur de la côte. Ainsi, d'échelle en échelle (mais nous préférons le terme de résolution), les détails viennent affiner notre image de la côte, d'une vision grossiére on passe à une vision fine et claire.

Mathématiquement, en analyse multirésolution on cherche à écrire toutes fonctions f de L^2 comme la limite d'approximations successives, chacune étant une version lissée de f, avec des fonctions lissées de plus en plus concentrées. Les ap-

proximations successives utilisent alors différentes résolutions, d'où le nom d'analyse multirésolution. Plus précisément, une analyse multirésolution consiste en :

1. une famille d'espace clos imbriqués les uns dans les autres $V_m \subset L^2(\mathbb{R})$, $m \in \mathbb{Z}$,
$$\cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \tag{2.1}$$
tels que

2. $$\bigcap_{m\in\mathbb{Z}} V_m = 0, \overline{\bigcup_{m\in\mathbb{Z}} V_m} = L^2(\mathbb{R}), \tag{2.2}$$
et

3. $$f \in V_m \iff f(2\cdot) \in V_{m-1}, \tag{2.3}$$
de plus, il existe un $\phi \in V_0$ tel que, pour tout $m \in \mathbb{Z}$, l'ensemble des ϕ_{mn}, avec $\phi_{mn}(x) = 2^{-m/2}\phi(2^{-m}x - n)$ constitue une base inconditionnelle de V_m, i.e.
$$\overline{V_m = \quad \text{combinaison linéaire} \quad \{\phi_{mn}, n \in \mathbb{Z}\}} \tag{2.4}$$
et, il existe $0 < A \leq B < \infty$ tel que, pour tout $(c_n)_{n\in\mathbb{Z}}$,
$$A\sum_n |c_n|^2 \leq \|\sum_n c_n\phi_{mn}\|^2 \leq B\sum_n |c_n|^2. \tag{2.5}$$

Soit P_m l'opérateur de projection orthogonale sur V_m. D'après (2.1), (2.2), on a $\lim_{m\to\infty} P_m f = f$, pour tout $f \in \mathrm{L}^2(\mathbb{R})$. La condition (2.3) assure que les V_m correspondent à différentes résolutions tandis que l'invariance par translation

$$f \in V_m \to f(\cdot - 2^m n) \in V_m \quad \text{pour tout} \quad n \in \mathbb{Z}$$

est une conséquence de (2.4), (2.5).

De cette définition, faisons quelques observations. Du fait que $\phi \in V_0 \subset V_{-1}$, une séquence $\{h_k\} \in l^2(\mathbb{Z})$ existe de telle sorte que la fonction d'échelle satisfait

$$\phi(x) = 2\sum_k h_k \phi(2x - k). \tag{2.6}$$

Cette équation fonctionnelle est appelée *équation de raffinement* ou *équation de dilatation*. La collection de fonctions $\{\phi(x - l), \quad l \in \mathbb{Z}\}$ est maintenant une base de Riesz de V_j.

On note également qu'une analyse multirésolution approche une fonction f par une fonction f^j dans chaque sous-espace V_j. Comme l'union des V_j est dense dans $L^2(\mathbb{R})$, on est assuré que les diverses approximations convergeront vers la fonction originale, $f = \lim_{j\to\infty} f^j$.

En intégrant des deux côtés de l'équation (2.6) et en utilisant le fait que l'intégrale de ϕ ne s'annule pas, on voit que

$$\sum_k h_k = 1 \iff H(0) = 1. \tag{2.7}$$

La fonction d'échelle est, sous des considérations très générales, définie par l'équation de raffinement et la normalisation [24, 26],

$$\int_{-\infty}^{+\infty} \phi(x)dx = 1 \iff \hat{\phi}(0) = 1. \tag{2.8}$$

Dans de nombreux cas, l'expression de ϕ n'est pas disponible. Toutefois, il existe des algorithmes rapides basés sur l'équation de raffinement pour évaluer ϕ aux points dyadiques $\{x = 2^{-j}k, \quad j,k \in \mathbb{Z}\}$ (voir l'algorithme cascade en section 2.2.1). En fait dans la plupart des applications, on ne travaille jamais avec ϕ mais directement avec les h_k.

De façon à pouvoir utiliser la famille $\{\phi(x-l), \quad l \in \mathbb{Z}\}$ pour approcher même les fonctions les plus simples (constantes par exemple), il est naturel d'assumer que la fonction d'échelle et ses translatées entières forment une *partition de l'unité*, i.e.

$$\forall x \in \mathbb{R} \quad : \quad \sum_k \phi(x-k) = 1.$$

Cela est aussi utilisé pour prouver qu'un ϕ donné génère une analyse multirésolution. Par la formule sommatoire de Poisson, la partition de l'unité est équivalente à

$$\hat{\phi}(2\pi k) = \delta_k \quad \text{pour} \quad k \in \mathbb{Z}. \tag{2.9}$$

Par (2.6), la transformée de Fourier de ϕ doit satisfaire

$$\hat{\phi}(\omega) = H(\omega/2)\hat{\phi}(\omega/2), \tag{2.10}$$

où H est une fonction 2π-périodique définie par

$$H(\omega) = \sum_k h_k e^{-ik\omega}$$

Comme $\hat{\phi}(0) = 1$, on peut appliquer récursivement (2.10). Ce qui conduit formellement à

$$\hat{\phi}(\omega) = \prod_{j=1}^{\infty} H(2^{-j}\omega).$$

La convergence de ce produit est examinée en détail dans [24, 16]. Cette formule peut être utilisée pour générer ϕ à partir des h_k. De (2.9), (2.10), on déduit qu'on obtient une partition de l'unité si

$$H(\pi) = 0 \quad \text{ou} \quad \sum_k (-1)^k h_k = 0$$

Une représentation multirésolution donne un aperçu simple de l'interprétation de l'information contenue dans une image [46, 12]. A différentes résolutions, les détails d'une image caractérisent généralement différentes structures physiques d'une scène. A une résolution grossière, les détails correspondent aux structures plus larges qui nous renseignent sur le "contexte" de l'image. Il est par conséquent naturel d'analyser d'abord les détails d'une image à une résolution grossière puis d'augmenter graduellement la résolution. Une telle approche "grossier-à-fin" est très utile pour les algorithmes de reconnaissance de structures. Ce concept a d'ailleurs été très étudié en traitements d'image bas niveau tels que mise en correspondance stéréo et mise en correspondance d'objets.

La fonction ondelette et l'espace de détail W_j

On note W_j l'espace complémentaire de V_j dans V_{j-1}, il vérifie donc

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

où le symbole $\oplus$ signifie somme directe. Cela signifie que l'espace W_j contient la différence d'information entre une approximation à la résolution j et une approximation à la résolution $j-1$. Par conséquent,

$$\bigoplus_j W_j = L^2(\mathbb{R}).$$

Définie ainsi, W_j n'est pas unique.

Une fonction ψ est une *ondelette* si l'ensemble de fonctions $\{\psi(x-l), \quad l \in \mathbb{Z}\}$ forme une base de Riesz de W_0. Tout comme pour ϕ_{jl}, les ψ_{jl} forment une base de Riesz. Comme ψ est un élément de V_{-1}, il existe une séquence $g_k \in l^2(\mathbb{Z})$ telle que

$$\psi(x) = 2\sum_k g_k \phi(2x-k). \tag{2.11}$$

De plus, on demande à l'ondelette d'avoir un certain nombre N de moments nuls (ce qui caractérise l'ordre de la régularité de l'ondelette)

$$\int_{-\infty}^{+\infty} \psi(x)x^l dx = 0, \quad l = 0,1,2,\ldots,N-1 \iff \hat{\psi}^{(l)}(0) = 0. \tag{2.12}$$

La transformée de Fourier d'ue ondelette est donc

$$\hat{\psi}(\omega) = G(\omega/2)\hat{\phi}(\omega/2), \tag{2.13}$$

où G est une fonction 2π-périodique

$$G(\omega) = \sum_k g_k e^{-ik\omega}.$$

De (2.11) et (2.12) on a

$$\sum_k g_k = 0 \quad \text{ou} \quad G(0) = 0.$$

On définit P_j et Q_j les opérateurs de projection sur V_j et W_j. Une fonction f peut alors être écrite

$$f(x) = \sum_j Q_j f(x) = \sum_{j,l} d_l^j \psi_{j,l}(x).$$

Dans la pratique, les indices j et l ne sont jamais infinis. Par conséquent, si on s'arrête au niveau L, on est obligé de tronquer cette somme en

$$\begin{aligned} f(x) &= \sum_{j=0}^{j=L} Q_j f(x) + P_L f(x) \\ &= \sum_{j=0}^{j=L}\sum_k d_k^j \psi_{j,k} + \sum_m c_m^L \phi_{L,m}, \end{aligned}$$

Notons toutefois que le nombre de fonctions ψ et ϕ n'est pas non plus infini : pour bien projeter une fonction f, il suffit que les translatées de $\phi_{j,0}$ et $\phi_{j,0}$ couvrent le domaine de définition de f.

La calcul des coefficients c_k^j et d_k^j est expliqué dans la section (2.6).

Ondelettes et polynômes

On définit le moment d'ordre p des fonctions de ϕ et ψ par

$$\mathcal{M}_p = \int_{-\infty}^{+\infty} \phi(x) x^p dx, \quad \mathcal{N}_p = \int_{-\infty}^{+\infty} \psi(x) x^p dx, \quad p \geq 0$$

Bien sûr, ces intégrales n'ont de sens que si f a une décroissance suffisante. Dans la section 2.1, on a vu que $\mathcal{M}_0 = 1$. Des formules récursives existent pour calculer ces moments [8]. Si on appelle $N \geq 1$ le nombre de moments nuls d'une ondelette :

$$\mathcal{N}_p = 0 \quad \text{pour} \quad 0 \leq p < N \quad \text{et} \quad \mathcal{N}_N \neq 0$$

Cela équivaut à

$$\hat{\psi}^{(p)}(0) = 0 \quad \text{pour} \quad 0 \leq p < N,$$

et comme $\hat{\phi}(0) = \mathcal{M}_0 \neq 0$, on a

$$G^{(p)}(0) = 0 \quad \text{pour} \quad 0 \leq p < N.$$

D'après équation (4.3.5), pour

$$\tilde{G}^{(p)}(0) = 0 \quad \text{pour} \quad 0 \leq p < N,$$

on a

$$H^{(p)}(\pi) = 0 \quad \text{pour} \quad 0 \leq p < N.$$

Cela signifie que H peut être factorisé sous la forme

$$H(\omega) = \left(\frac{1 + e^{-i\omega}}{2} \right)^N K(\omega)$$

avec $K(0) = 1$ et $K(\pi) \neq 0$. Cette factorisation et les conditions de bi-orthogonalité sont utilisées comme point de départ pour la construction d'ondelettes à support compact [17, 24].

On a également

$$i^p \hat{\phi}^{(p)}(2k\pi) = \delta_k \mathcal{M}_p \quad \text{pour} \quad 0 \leq p < N, \tag{2.14}$$

et, par la formule sommatoire de Poisson,

$$\sum_l (x - l)^p \phi(x - l) = \mathcal{M}_p \quad \text{pour} \quad 0 \leq p < N.$$

De cette dernière expression on en déduit que tout polynôme de degré plus petit que N peut être écrit comme une combinaison linéaire des fonctions $\phi(x - l)$, avec $l \in \mathbb{Z}$. Les coefficients de la combinaison linéaire sont eux-mêmes des polynômes en l.

De plus, si $\mathcal{M}_p = 0$ pour $p \in [0, N - 1]$, cela signifie que la projection de polynômes de degré inférieur à N est nulle. Par conséquent, on n'aura pas de coefficients ondelettes. Cette remarque est très importante pour la compression de signaux car

plus on aura de moments nuls pour ψ et plus l'énergie sera concentrée sur quelques coefficients (cas de fonctions régulières). Si $f \in \mathcal{C}^N$ alors

$$\|P_j f(x) - f(x)\| = \mathcal{O}(2^{-nN}).$$

Les conditions (2.14) sont souvent appelées conditions de Strang-Fix et elles ont été établies bien avant le développement de la théorie des ondelettes. Le nombre de moments nuls a également de l'importance pour la détection de singularité dans un signal : une transition significative dans les coefficients ondelettes correspondra à une haute fréquence et par conséquent à une variation brusque dans la fonction originale.

L'exposant N dans la factorisation de H joue un rôle sur la régularité de ϕ qui est inférieure à $N-1$ dû à l'influence de K [53, 26, 25].

2.2 Génération de ϕ et ψ

Les fonctions d'échelle ϕ et ondelette ψ posent un problème délicat pour leur génération en informatique. Dans le cas où les filtres sont orthogonaux (voir chapitre 3) :

$$\phi(x) = 2\sum_n h_n \phi(2x-n) \tag{2.15}$$

$$\psi(x) = 2\sum_n g_n \phi(2x-n) \tag{2.16}$$

Si on prend les transformées de Fourier respectives,

$$\hat{\phi}(\omega) = H(\omega/2)\hat{\phi}(\omega/2) \tag{2.17}$$

$$\hat{\psi}(\omega) = G(\omega/2)\hat{\phi}(\omega/2) \tag{2.18}$$

Par conséquent, par itérations successives,

$$\hat{\phi}(\omega) = \prod_{j=1}^{\infty} H(2^{-j}\omega) \tag{2.19}$$

Daubechies [24] montre que (2.19) converge uniformément pour tout $\omega \in \mathbb{R}$ sur des ensembles compacts si et seulement si

$$\exists \varepsilon > 0 \quad \text{tel que} \quad \sum_n |h_n||n^\varepsilon| < \infty \tag{2.20}$$

Cela conduit au théorème 3.6 de [24]:

Théorème 3.6 [24] *Soit h_n une séquence telle que*

1° $\sum_n |h_n||n^\varepsilon| < \infty$ *pour un* $\varepsilon > 0$,

2° $\sum_n h_{n-2k} h_{n-2l} = \delta_{kl}$,

3° $\sum h_n = \sqrt{2}$,

Supposons que $m_0(\omega) = 2^{-1/2}\sum_n h_n e^{i\omega n}$ *peut être écrit*

$$m_0(\omega) = \left[\frac{1}{2}(1+e^{i\omega})\right]^N \left[\sum_n f_n e^{in\omega}\right], \tag{2.21}$$

où

4^o $\sum_n |f_n||n^\varepsilon| < \infty$ *pour un* $\varepsilon > 0$,

5^o $\sup_{\omega \in \mathbb{R}} |\sum_n f_n e^{in\omega}| < 2^{N-1}$,

On définit

$$g_n = (-1)^n h_{-n+1}, \tag{2.22}$$

$$\hat{\phi}(\omega) = (2\pi)^{-1/2} \prod_{j=1}^{\infty} m_0(2^{-j}\omega), \tag{2.23}$$

$$\psi(x) = \sqrt{2} \sum_n g_n \phi(2x - n). \tag{2.24}$$

Alors les $\phi_{jk}(x) = 2^{-j/2}\phi(2^{-j}x - k)$ *définissent une analyse multirésolution et les* ψ_{jk} *associés forment une base ondelette orthonormale.*

Cela nous montre que connaissant les coefficients du filtre h, on forme m_0 et les itérations succesives de celui-ci conduisent à ϕ . Dit autrement:

Soit h_n *une séquence de longueur finie,* $h_n = 0$ *pour* $n < N_-$ *ou* $n > N_+$, *alors l'ondelette de base correspondante a un support compact:*

$$\phi(x) = \lim_{l \to \infty} \eta_l(x), \tag{2.25}$$

$$\eta_l(x) = \sqrt{2} \sum_{n=N_-}^{N_+} h_n \eta_{l-1}(2x - n), \tag{2.26}$$

$$\eta_0 = \chi_{[-1/2,1/2[}. \tag{2.27}$$

La définition récursive de η_l implique que η_l a un support compact, $\mathrm{supp}\eta_l \subset [N_{l,-}, N_{l,+}]$ avec $N_{l,-} = \frac{1}{2}(N_{l-1,-} + N_-)$ et $N_{l,+} = \frac{1}{2}(N_{l-1,+} + N_+)$ et $N_{0,-} = \frac{-1}{2}, N_{0,+} = \frac{1}{2}$.

Nous allons donc détailler cet algorithme plus communément appelé *Algorithme Cascade* de Daubechies [24].

2.2.1 Algorithme cascade

L'algorithme cascade vient de l'algorithme de la pyramide Laplacienne de P. Burt et E. Adelson [12] pour coder des images. L'ensemble des données 1-dimension peut être représenté par un ensemble de nombres, $\{c_n^0\}_n \in \mathbb{Z}$, représentant les valeurs des pixels. On souhaite décomposer c^0 en différents ensembles correspondant à différents niveaux de fréquences spatiales. Le plus haut niveau, avec seulement les hautes fréquences contenues dans c^0, est obtenu en calculant la différence entre c^0 et une version lisée $\tilde{c}^0$. Ce qui reste, i.e. la version lissée, ne contient que des fréquences spatiales plus basses et par conséquent peut être échantillonnée plus largement que c^0 lui-même, sans perte d'information. Le processus est alors réitéré plusieurs fois de façon à effectuer la décomposition voulue.

Soit un opérateur $F : l^2(\mathbb{Z}) \to l^2(\mathbb{Z})$ (F pour "filtre") défini par

$$(Fa)_k = \sum_n w(n - 2k) a_n \tag{2.28}$$

où les $\{w_n\}$ sont les coefficients du filtre. Alors on a

$$c^1 = Fc^0, \tag{2.29}$$

$$\tilde{c}^0 = F^* c^1 = F^* F c^1, \tag{2.30}$$

$$d^0 = c^0 - \tilde{c}^0 = (\mathrm{I} - F^* F)c^0. \tag{2.31}$$

On utilise la notation F^* pour l'opérateur adjoint de F. De plus, on a supposé que $c^0 \in l^2(\mathbb{Z})$ ou, en terme d'analyse du signal, que la séquence de donnée c^0 est d'énergie finie. En pratique c^0 est toujours finie et cette contrainte ne pose aucun problème.

Si on suppose que la décomposition est totale après L itérations avec

$$c^l = Fc^{l-1}, l = 1, \cdots, L, \tag{2.32}$$

$$d^{l-1} = c^{l-1} - F^* c^l = (\mathrm{I} - F^* F)c^{l-1} \tag{2.33}$$

A partir de la séquence $d^0 = \cdots = d^{L-1} = 0, c^L$, on peut reconstruire c^0 récursivement par

$$c^{l-1} = d^{l-1} + F^* c^l. \tag{2.34}$$

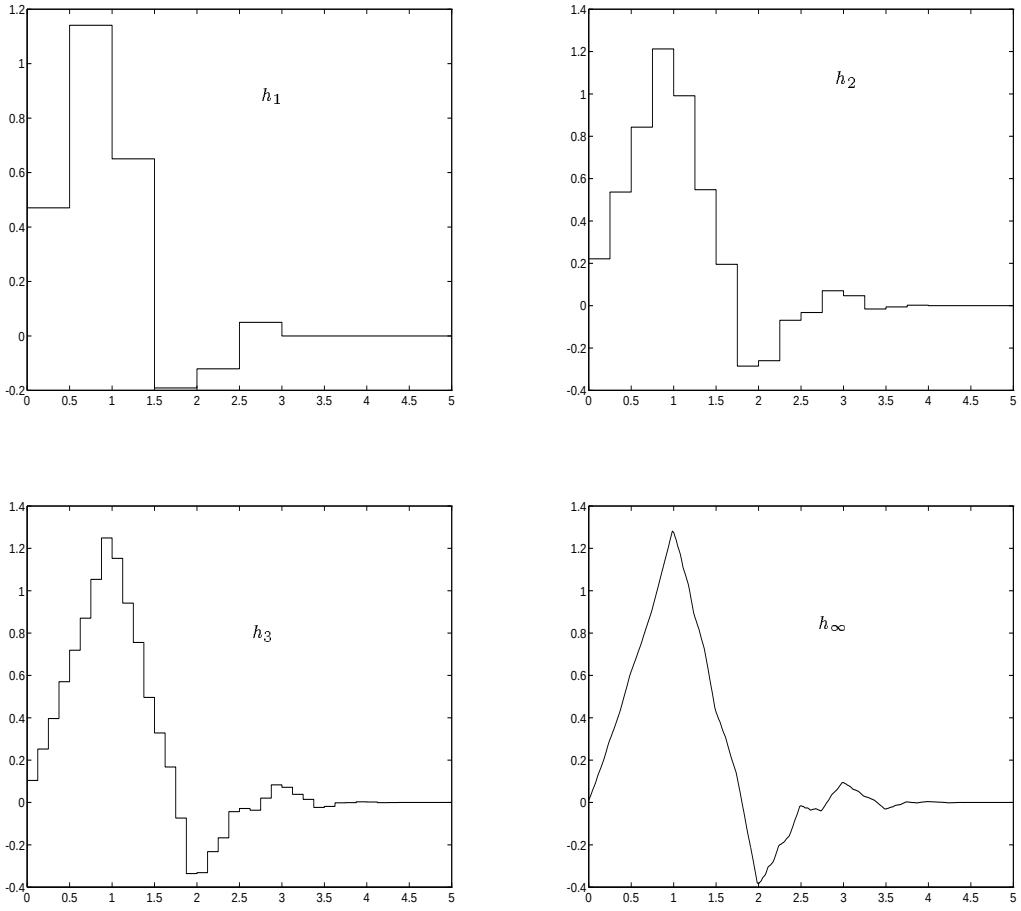


FIG. 2.2 - : *Algorithme cascade: différentes étapes.*

On peut se demander quelle est la séquence c^0 qui donnera la plus simple séquence en décomposition, i.e., telle que $d^0 = \cdots = d^{L-1} = 0$ et $(c^L)_n = \delta_{n0}$. La réponse est naturellement (en utilisant l'algorithme de reconstruction)

$$c^0 = (F^*)^L e, \tag{2.35}$$

où e est une séquence telle que $e_n = \delta_{0n}$. Comme toute séquence peut être considérée comme une somme de version translatée de e, la séquence $c^0 = (F^*)^L e$ donne la brique de base pour le sous-espace $(F^*)^L l^2(\mathbb{Z})$. On peut faire une représentation graphique de c^0 pour des L successifs. Nous reprsentons la séquence e par un simple histogramme:

$$e = \begin{cases} 1 & \text{pour} \quad 0 \leq x < 1, \\ 0 & \text{sinon.} \end{cases} \tag{2.36}$$

La séquence F^*e est définie sur une échelle deux fois plus petite et sera représentée par un histogramme avec des marches de largeur $\frac{1}{2}$ (au lieu de 1 pour e); ses différentes amplitudes sont données par $2(F^*e)_n = 2w(n)$. De même, $(F^*)^l e$ est représenté par un histogramme de largeur de marche 2^{-l}; les amplitudes successives sont données par $2^l((F^*)^l e)_n$. Le facteur 2 introduit ici sert à des fins de normalisation de telle sorte que l'aire sous l'histogramme soit toujours de 1 (voir figure (2.2)).

Cet algorithme peut être résumé mathématiquement en définissant l'opérateur $\tilde{T}_j$:

$$h_j = \tilde{T}_j h_{j-1} \quad \text{ou,} \quad h_j = \tilde{T}_j \tilde{T}_{j-1} \cdots \tilde{T}_1 h_0 \tag{2.42}$$

avec

$$(\tilde{T}_j f)(x) = 2 \sum_k \sum_n w(n)(\chi_{[0,2^{-j+1}(k+1)[} f)(2x - 2^{-j+1}(k+n)). \tag{2.43}$$

L'implémentation numérique est très rapide. Toutefois, les h_j peuvent être écrits différemment. Si on reprend les expressions (2.37), (2.38) pour h_1 et h_2, on a

$$h_1(x) = 2 \sum_n w(n) h_0(2x - n), \tag{2.44}$$

$$\begin{aligned} h_2(x) &= 4 \sum_n w(n) \sum_m \chi_{[0,\frac{1}{2}[}(2x - n - \frac{1}{2}m) \\ &= 2 \sum_n w(n) h_1(2x - n) \end{aligned} \tag{2.45}$$

Ce qui nous amène à

$$h_j = T h_{j-1} = \cdots = T^j h_0, \tag{2.46}$$

avec

$$(Tf)(x) = 2 \sum_n w(n) f(2x - n). \tag{2.47}$$

On a donc deux manières de calculer h_j: (2.42) et (2.46).

Dans Wavelib 1.0 , seule (2.46) a été implémentée. On notera donc qu'une fois que les coefficients du filtre h sont donnés, on peut donc générer la fonction d'échelle ϕ.

Algorithme

Soit $h_0 = e$, les représentations de h_1, h_2 de respectivement F^*e, $(F^*)^2e$ sont donnés par

$$h_1(x) = 2\sum_n w(n)\chi_{[0,\frac{1}{2}[}(x - \frac{1}{2}n) \tag{2.37}$$

$$h_1(x) = 4\sum_n \left[\sum_m w(m)\chi_{[0,\frac{1}{4}[}(x - \frac{1}{2}n - \frac{1}{4}m)\right]. \tag{2.38}$$

Par conséquent pour passer de h_{j-1} à h_j, on doit faire:

1. diviser h_{j-1}, une fonction en escalier de pas $2^{-(j-1)}$ en ses composants
$$h_{j-1} = \sum_k a_{j-1,k}\chi_{[0,2^{-(j-1)}(k+1)[} \tag{2.39}$$
2. remplacer chaque composante par une version proprement mise à l'échelle et recentrée de h_1,
$$\chi_{[0,2^{-(j-1)}(k+1)[} \to h_1(2^{j-1}x-k) = 2\sum_n w(n)\chi_{[0,\frac{1}{2}[}(2^{j-1}x-k-\frac{1}{2}n), \tag{2.40}$$
3. tout sommer,
$$h_j(x) = 2\sum_k a_{j-1,k}\sum_n \chi_{[0,2^{-j}(2x+n+1)[} \tag{2.41}$$

FIG. 2.3 - : *Algorithme cascade*

Implémentation informatique

Comme nous l'avons vu dans la section précédente, si on part de $h_0 = \chi_{[0,1[}$, histogramme de pas 1, h_1 sera de pas $\frac{1}{2}$ et l'amplitude des différentes marches sera les coefficients respectifs du filtre h, i.e. $h_1(n) = h_n$. On note supp$h$ le support du filtre h.

Si supp$h = [a, b]$, on sait que supp$\phi = [a, b]$. En effet,

$$\text{supp}h_1 = \left[\frac{a}{2}, \frac{b}{2}\right],$$

puis on a,

$$\text{supp}h_2 = \left[\frac{a}{4} + \frac{a}{2}, \frac{b}{2} + \frac{b}{4}\right],$$

finalement,

$$\text{supp}h_\infty = \text{supp}\phi = \left[\sum_{n=1}^{\infty} 2^{-n}a, \sum_{m=1}^{\infty} 2^{-m}b\right] \to [a, b].$$

Par conséquent, on remarque qu'au niveau 1, on a a valeurs nulle à gauche et b valeurs nulles à droite. En effet, on a un espace de largeur $\frac{a}{2}$ nulle. Or, au niveau 1, les valeurs de h_1 sont séparées de $\frac{1}{2}$, donc on a a valeurs nulles; idem à droite. Le plus intéressant est d'observer que si on génère ϕ avec cet algorithme et sur L niveaux, on aura toujours a valeurs nulles à gauche et b valeurs nulles à droite mais cette fois-ci séparées de 2^{-L}.

Cette remarque est très importante informatiquement parlant car bien qu'ayant généré ϕ avec cet algorithme, il ne faut pas oublier de réserver a cases vides à gauche et b cases vides à droite dans le tableau contenant les valeurs de ϕ; sinon l'intervalle de définition est faux! De plus, comme nous l'avons dit au paragraphe précédent, nous avons généré une fonction en escalier de pas 2^{-L}. Il s'agit donc d'une approximation (discrétisation) de la fonction continue ϕ. Plus le niveau L sera grand, plus le pas de discrétisation sera faible et plus l'approximation sera bonne.

Nous proposons deux algorithmes (2.1). On remarque qu'à chaque itération tous deux donnent le même ϕ . La seule différence réside dans la manière de le calculer. Dans Wavelib 1.0 , les deux versions existent mais la deuxième est plus simple à implémenter (et donc plus rapide si L devient grand). Elle est par conséquent utilisée par d'autres fonctions tandis que la première version est plus là à titre documentaire.

Notons également qu'à l'itération 0, on a un écart de $\frac{1}{2}$ entre les échantillons de ϕ . Aux itérations suivantes, cet écart devient à chaque fois deux fois plus petit. Un exemple illustrant le fonctionnement de ces algorithmes est donné en figure (2.4).

Idée de Strang

G. Strang [67] propose également une méthode pour générer ϕ. Elle se base sur une remarque quant à l'équation de dilatation (2.6). Dans l'algorithme cascade, on réalise $\phi_j(x) = \sum h_k \phi_{j-1}(2x - k)$ pour passer de l'itération $j - 1$ à l'itération j, ce qui peut s'écrire

$$\phi = L\phi$$

avec

$$L_{ij} = h_{2i-j}.$$

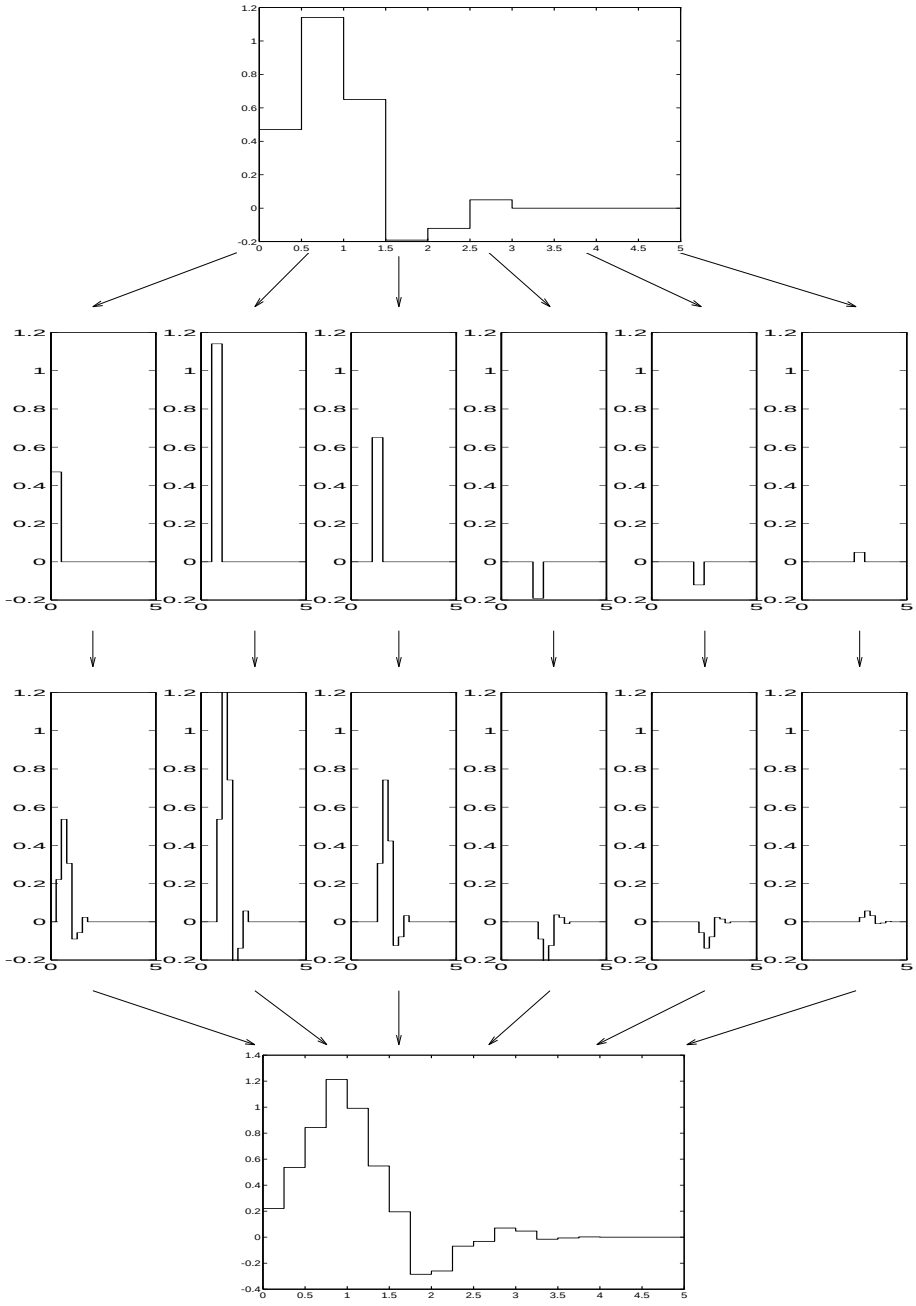


FIG. 2.4 - : *Algorithme cascade: exemple avec D_3.*

Algorithme

Au tout début, on part d'un tableau initialisé avec les valeurs de h. Ensuite viennent deux algorithmes possibles:
On suppose supp$h$ = supp$\phi = [a, b]$,

1. (a) On ajoute autant de zéros à la fin que la longeur de ϕ ,
   (b) On filtre ϕ avec h,
   (c) On insère des zéros entre chaque éléments du filtre h.
   et on recommence $L - 1$ fois.
2. (a) on crée un nouveau tableau pour ϕ de taille [taille précédente + a + b] (car on va rajouter a éléments à gauche et b éléments à droite,
   (b) sur chaque éléments du tableau précédent on centre une version de h multipliée par l'élément correspondant.
   (c) on additionne tout.
   et on recommence $L - 1$ fois.

Au final, on ajoute a cases vides à gauche et b cases vides à droite dans le tableau par respect pour l'intervalle de définition.

TAB. 2.1 - : *Génération de* ϕ

Il montre alors que le vecteur propre correspondant à la valeur propre 1 contient les valeurs de $\phi(x)$ aux entiers. Ce sont donc les valeurs de $\phi_0(x)$.

Cette méthode semble marcher pour toutes les ondelettes. Toutefois, on remarque que cela ne fonctionne pas avec les coiflets. On lui préfère généralement la méthode cascade.

Génération de ψ

De part l'algorithme cascade, la définition récursive de ϕ est à support compact. Or, la séquence g_n est finie. Par conséquent, le support de ψ est lui-même compact. Si

$$\text{supp}\phi = [a, b] \tag{2.48}$$

et

$$\text{supp}g = [c, d] \tag{2.49}$$

alors,

$$\text{supp}\psi = \left[\frac{a + d + 1}{2}, \frac{b + c + 1}{2}\right] \tag{2.50}$$

– *Dans le cas orthogonal (section 3.2),*

$$\psi(x) = 2^{1/2} \sum_n g_n \phi(2x - n),$$

avec

$$\text{supp}\phi = \text{supp}h = [0, 2N-1]$$

et $g_n = (-1)^n h_{-n+1} \Longrightarrow \text{supp}g = [-2N+1, 0]$. Ce qui nous donne $\text{supp}\psi = [-N+1, N]$.

– *Dans le cas biorthogonal (section 3.3),*

$$\psi(x) = 2^{1/2} \sum_n g_{n+1} \phi(2x-n),$$

avec

$$\text{supp}\phi = \text{supp}h = \left[-\frac{N-\chi}{2}, \frac{N+\chi}{2}\right],$$

$$\text{supp}\tilde{\phi} = \text{supp}\tilde{h} = \left[-k+1-\frac{\tilde{N}-\chi}{2}, k-1+\frac{\tilde{N}+\chi}{2}\right]$$

et $g_n = (-1)^{n+1}\tilde{h}_{-n} \Longrightarrow \text{supp}g = \left[-k+1-\frac{\tilde{N}+\chi}{2}, k-1+\frac{\tilde{N}-\chi}{2}\right]$. D'où $\text{supp}\psi = [-k+1, k] = \left[-\frac{N+\tilde{N}-2}{2}, \frac{N+\tilde{N}}{2}\right]$.

L'algorithme (2.2) propose une méthode pour obtenir ϕ et ψ avec le même écart entre échantillons. Si on désire la projection d'une fonction f sur un espace V_j,

$$f^j = \sum_k s_k^j \phi_{jk} + \sum_k d_k^j \psi_{jk} \tag{2.51}$$

Dans ce cas, il est très important de bien connaitre non seulement les supports des fonctions ϕ_{jk} et ψ_{jk} mais également l'approximation faite sur celles-ci doit être la même.

Algorithme

Si on veut créer ϕ avec L itérations et ψ avec le même écart dx entre les échantillons (de façon à ce que l'approximation faite sur chacune de ces fonctions soit la même), il faut:

- appliquer l'algorithme cascade pour ϕ sur $L-1$ itérations (sans ajouter de zéros à gauche ou à droite du tableau),
- calculer $\psi(x) = 2^{1/2} \sum_n g_n \phi(2x-n)$,
- appliquer l'algorithme cascade une fois pour ϕ .

On a alors ϕ et ψ générés avec un écart entre échantillons $dx = 2^{-(L+1)}$.

TAB. 2.2 - : *Génération de ψ.*

2.3 Transformées ondelettes

2.3.1 Transformée ondelette continue - CWT

Une expansion ondelette consiste en des translations et des dilatations d'une seule et même fonction fixée, l'ondelette $\psi \in \mathrm{L}^2(\mathbb{R})$. Dans la transformée ondelette continue, les paramètres de translation et de dilatation varient continuement. Cela signifie qu'on utilise les fonctions

$$\psi_{ab}(x) = \frac{1}{\sqrt{|a|}}\psi(\frac{x-b}{a}) \quad \text{avec} \quad a, b \in \mathbb{R}, a \neq 0. \tag{2.52}$$

Ces fonctions sont mises à l'échelle de telle sorte que leur norme dans $\mathrm{L}^2(\mathbb{R})$ soit indépendante de a. La *transformée ondelette continue* (CWT) d'une fonction $f \in \mathrm{L}^2(\mathbb{R})$ est définie par

$$\mathcal{W}(a,b) = \langle f, \psi_{ab} \rangle. \tag{2.53}$$

En utilisant l'identité de Parseval, on peut aussi l'écrire

$$\mathcal{W}(a,b) = (2\pi)^{-1}\langle \hat{f}, \hat{\psi}_{ab} \rangle. \tag{2.54}$$

où

$$\hat{\psi}_{ab}(\omega) = \frac{a}{\sqrt{|a|}} e^{-i\omega b}\hat{\psi}(a\omega). \tag{2.55}$$

On notera que la CWT convertit une fonction à une variable en une fonction à deux variables. La représentation d'une fonction par sa CWT est redondante et la transformée inverse n'est donc pas toujours unique. De plus, toutes les fonctions $\mathcal{W}(a,b)$ ne sont pas forcément la CWT de la fonction f.

Si l'ondelette ψ satisfait la *condition d'admissibilité*

$$C_\psi = \int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{\omega} d\omega < \infty, \tag{2.56}$$

alors, la Transformée ondelette continue $\mathcal{W}(a,b)$ admet un inverse

$$f(x) = \frac{1}{C_\psi} \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} \mathcal{W}(a,b)\psi_{ab}(x) \frac{dadb}{a^2}. \tag{2.57}$$

D'après la condition d'admissibilité, nous voyons que $\hat{\psi}(0) = 0$ et par conséquent, ψ doit osciller.

La transformée est souvent représentée par une image 2D en couleur ou en niveaux de gris correspondant au module et phase de $\mathcal{W}(a,b)$.

L'avantage de cette caractérisation par rapport à la transformée de Fourier est qu'elle ne donne pas seulement une information sur le type de singularité mais aussi *sa localisation dans le temps*. Il existe aussi une caractérisation analogue avec les fonctions continues de Hölder (Lipschitz) d'ordre $\alpha \leq 1$: l'ondelette doit avoir une nombre de moments nuls supérieur à α, i.e.

$$\int_{-\infty}^{+\infty} \psi(x)x^p dx = \hat{\psi}^{(p)}(0) = 0 \quad \text{pour} \quad p = 0, 1, \cdots, \alpha \quad \text{et} \quad p \in \mathbb{Z}. \tag{2.58}$$

Ainsi, le nombre de moments nuls de l'ondelette limite l'ordre de la régularité qui peut être caractérisée.

2.3.2 Transformée ondelette discrète - DWT

Pour des applications d'analyse du signal, on choisit de restreindre les valeurs des paramètres a et b dans (2.52) à une grille discrète. Dans ce cas on fixe un pas de dilatation $a_0 > 1$ et un pas de translation $b_0 \neq 0$. La famille ondelette qui nous intéresse est alors

$$\psi_{mn}(x) = a_0^{-m/2}\psi(a_0^{-m}x - nb_0) \tag{2.59}$$

Cela correspond aux choix :

$$\begin{aligned} a &= a_0^m, \\ b &= nb_0a_0^m \end{aligned}$$

ce qui nous montre que le paramètre de translation b dépend du taux de dilatation choisi.

Une Transformée Ondelette Discrète (DWT), T, est associée aux ondelettes discrètes (2.59). Elle associe à une fonction f une séquence indexée dans $\mathbb{Z}^2$,

$$\begin{aligned} (Tf)_{mn} &= \langle \psi_{mn}, f \rangle \\ &= a_0^{-m/2} \int \overline{h(a_0^{-m}x - nb_0)} f(x)dx. \end{aligned} \tag{2.60}$$

où m est appelé le paramètre d'échelle et n le paramètre de translation, $\{m, n\} \in \mathbb{Z}^2$.

Si ψ est admissible (i.e. vérifie (2.56)) et si ψ admet une décroisance suffisante, alors $T : \mathrm{L}^2(\mathbb{R}) \rightarrow l^2(\mathbb{Z}^2)$. En général, T n'admet pas d'inverse borné sur son intervalle de définition. S'il en existe un, alors $\exists A, B$ tels que

$$A\|f\|^2 < \sum_{m,n\in\mathbb{Z}} |\langle \psi_{mn}, f \rangle|^2 < B\|f\|^2, \quad \forall f \in \mathrm{L}^2(\mathbb{R}),$$

alors l'ensemble $\{\psi_{mn}; m, n \in \mathbb{Z}\}$ est appelé une "frame". Dans ce cas, on peut construire des algorithmes numériquement stables pour reconstruire f à partir de ses coefficients ondelettes $d_n^m = \langle \psi_{mn}, f \rangle$. En particulier,

$$f = \frac{2}{A+B} \sum_{m,n} \psi_{mn} \langle \psi_{mn}, f \rangle + R, \tag{2.61}$$

avec

$$\|R\| \leq \mathcal{O}(\frac{B}{A} - 1)\|f\|, \quad \text{Terme d'erreur.}$$

Si $B/A \approx 1$ alors R peut être omis.

Dans beaucoup d'applications, il est préférable de réduire au maximum la redondance de cette représentation. Dans ce cas, on choisit des valeurs de a_0, b_0 (typiquement $a_0 = 2$, $b_0 = 1$) pour lesquels les ψ_{mn} constituent une base orthonormale :

$$\psi_{mn}(x) = 2^{-m/2}\psi(2^{-m}x - n), \tag{2.62}$$

Notons que le choix de $b_0 = 1$ est purement arbitraire, et qu'il est choisi par convénience.

2.3.3 Fast Wavelet Transform - FWT

Nous allons maintenant expliciter l'algorithme de décomposition/reconstruction de S. Mallat [45]. On veut décomposer une séquence (discrète) $c^0 = (c^0_n)_n \in l^2(\mathbb{Z})$ en différents niveaux correspondant à différentes bandes de fréquences spatiales. Pour arriver à cela on va utiliser l'analyse multirésolution. On suppose qu'on a choisi des espaces V_m et une fonction ϕ tels que (2.1)-(2.4) soient satisfaits. On suppose que les ϕ_{0n} sont (bi)orthogonaux et soit $\{\psi_{mn}; m, n \in \mathbb{Z}\}$ une base (bi)orthonormale ondelette associée. A partir de la séquence de $c^0 \in l^2(\mathbb{Z})$, on forme la fonction f,

$$f = \sum_n c^0_n \phi_{0n},$$

ou,

$$f(x) = \sum_n c^0_n \phi(x - n).$$

Cette fonction est clairement un élément de V_0. Nous allons donc pouvoir utiliser tout le concept d'analyse multirésolution pour cette fonction. On va calculer les $P_j f$ successifs correspondant à des versions de f de plus en plus lisses, et aussi $Q_j f$ correspondant à la différence d'information entre deux "versions" de f à deux niveaux de résolution consécutifs.
Comme élément de $V_0 = V_1 \oplus W_1$, f peut être décomposée en ses composantes sur V_1 et W_1,

$$f = P_1 f + Q_1 f.$$

et chacune de ces composantes peut être redécomposée respectivement sur les bases (bi)orthogonales ϕ_{1n} et ψ_{1n},

$$P_1 f = \sum_k c^1_k \phi_{1k},$$

$$Q_1 f = \sum_k d^1_k \psi_{1k}.$$

La séquence c^1 représente une version plus lissée que c^0 et d^0 représente la différence d'information entre c^0 et c^1. Comme les ϕ_{1n} forment une base orthogonale de V_1, on a

$$\begin{aligned} c^1_k &= \langle \phi_{1k}, P_1 f \rangle = \langle \phi_{1k}, f \rangle \\ &= \sum c^0_n \langle \phi_{1k}, \phi_{0n} \rangle, \end{aligned}$$

où,

$$\begin{aligned} \langle \phi_{1k}, \phi_{on} \rangle &= 2^{-1/2} \int \phi(\frac{1}{2}x - k)\phi(x - n)dx \\ &= 2^{-1/2} \int \phi(\frac{1}{2}x)\phi(x - (n - 2k))dx. \end{aligned}$$

Ce qui par conséquent peut s'écrire,

$$c^1_k = \sum_n h(n - 2k)c^0_n \tag{2.63}$$

avec,

$$h(n) = 2^{-1/2} \int \phi(\frac{1}{2}x)\phi(x-n)dx.$$

De même, on peut écrire pour les coefficients de détails :

$$d_k^1 = \sum_n g(n-2k)c_n^0 \tag{2.64}$$

avec,

$$h(n) = 2^{-1/2} \int \psi(\frac{1}{2}x)\phi(x-n)dx.$$

Mathématiquement, il est plus commode de définir deux opérateurs,

$$\begin{aligned} (Ha)_k &= \sum_n h(n-2k)a_n, \\ (Ga)_k &= \sum_n g(n-2k)a_n. \end{aligned} \tag{2.65}$$

La procédure de décomposition peut être réitérée autant de fois que l'on veut (en pratique ce n'est vrai que pour un nombre limité de répétition car un signal réel est composé d'un nombre fini d'échantillons).

Algorithme

Soit une séquence $c_n^0 = \langle f, \phi_{0n} \rangle$, projection d'une fonction f sur l'espace V_0. A chaque étape, on a donc :

$$\begin{aligned} P_{j-1}f &= P_j f + Q_j f \\ &= \sum_k c_k^j \phi_{jk} + \sum_k d_k^j \psi_{jk} \end{aligned}$$

avec,

$$c^j = Hc^{j-1} \tag{2.66}$$

$$d^j = Gc^{j-1} \tag{2.67}$$

les opérateurs H et G étant donnés par 2.66.

TAB. 2.3 - : *Algorithme de décomposition ondelette rapide.*

Enfin, ce que l'on va conserver, c'est la décomposition $d^1, d^2, \cdots, d^L, c^L$. L'algorithme est clairement sous forme d'arbre et il est donc très facile de l'implémenter récursivement. Il faut noter également que si au départ on avait N échantillons c^0, après L niveaux, on va en conserver $N/2 + N/4 + \cdots + N/2^{L-1} + N/2^L + N/2^L = N$. L'algorithme de Mallat préserve donc le nombre d'échantillons non-nuls (ce qui est effectivement ce que l'on attend d'un algorithme de décomposition sur des bases orthogonales).

Pour ce qu'il y est reconstruction parfaite [73], il faut et il suffit que les opérateurs H et G et leurs adjoints H^* et G^* vérifient (dans le cas orthogonal),

$$H^*H + G^*G = I. \tag{2.68}$$

Algorithme

supposons qu'on connaisse c^j et d^j, alors

$$\begin{aligned} P_{j-1}f &= P_jf + Q_jf \\ &= \sum_k c_k^j \phi_{jk} + \sum_k d_k^j \psi_{jk}, \end{aligned}$$

on a donc,

$$\begin{aligned} c_n^{j-1} &= \langle \phi_{j-1,n}, P_{j-1}f \rangle \\ &= \sum_k c_k^j \langle \phi_{j-1,n}, \phi_{jk} \rangle + \sum_k d_k^j \langle \phi_{j-1,n}, \psi_{jk} \rangle \\ &= \sum_k h(n-2k) c_k^j + \sum_k g(n-2k) d_k^j, \end{aligned}$$

Ce qui se note $c^{j-1} = H^* c^j + G^* d^j$.

TAB. 2.4 - : *Algorithme de reconstruction ondelette rapide.*

L'algorithme de reconstruction est lui aussi sous forme d'arbre et s'implémente très facilement en récursif. Les filtres utilisés en reconstruction sont les mêmes qu'à la décomposition dans le cas de filtres orthogonaux (section 3.2) sinon ils sont diffèrents (section 3.3).

2.3.4 Tranformations périodiques et symétriques

En pratique, les signaux sont toujours finis. Aussi se pose le délicat problème des effets de bord : que se passe-t-il aux bords du signal lorsque les filtres viennent agir sur des échantillons qui n'existent pas ?

Il y a plusieurs réponses :

- **Naif.** on comble les échantillons manquants par des zéros. Faire cela va introduire aux bords une discontinuité qui va se répercuter de niveaux en niveaux,

- **Périodisation.** On périodise le signal en le répétant de chaque côté du tableau contenant les échantillons. Informatiquement parlant, cela revient à appliquer la fonction modulo au nombre d'échantillons du signal. Cette solution est intéressante car elle provoque une périodisation en fréquence (i.e. dans le plan de Fourier). Or, cette périodisation ne gêne en rien la FWT à condition de respecter le théorème de Nyquist-Shannon sur la fréquence d'échantillonnage du signal lui-même sinon les signaux périodisés vont se chevaucher et on ne pourra plus retrouver le signal original, c'est l'effet d'*aliasing*,

- **Symétrisation.** Aux extrémités du tableau, on symétrise le signal par rapport aux bords respectifs. La périodisation introduit souvent des discontinuités dans le signal aux bords car le signal n'est généralement pas le même à chaque extrémité. Par symétrie, on va avoir un effet de lissage aux bords sur les coeffi-

cients transformées. Par contre, ce lissage peut s'avérer désastreux sur le plan fréquenciel et provoquer une mauvaise recontruction aux bords.

Dans WaveLib 1.0, nous avons choisi de n'utiliser que la périodisation, qui apporte dans la plupart des cas une reconstruction parfaite et des coefficients transformées sans trop d'effets de bords.

2.3.5 DWT standard

DWT 1D

A partir d'un vecteur initial contenant 2^N échantillons, on crée deux sous-vecteurs de 2^{N-1} échantillons. L'un est obtenu en filtrant passe-bas (avec le filtre h) et l'autre en filtrant passe-bande (avec le filtre g). On répète cette procédure sur la version passe-bas. On voit donc que dans le cas discret, on ne peut faire plus de N niveaux de décomposition (2.5)

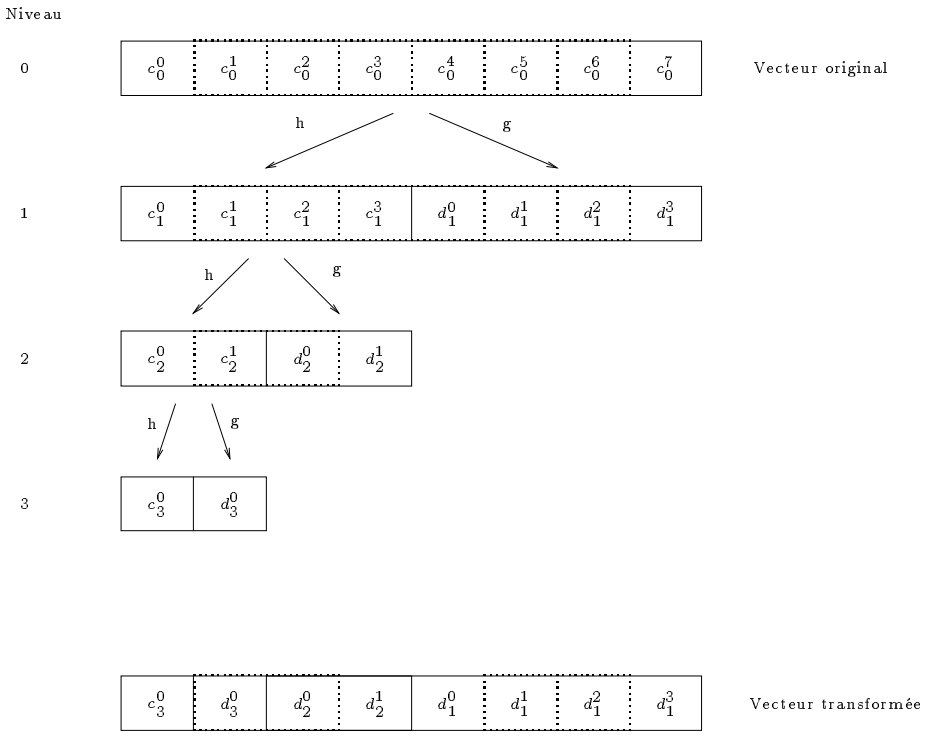


FIG. 2.5 - : *Exemple de DWT sur un vecteur de $N = 2^3$ échantillons.*

L'implémentation est donc très simple : pour passer d'un niveau à l'autre, on filtre et on décime par 2.

DWT 2D

En 2D, on a

$$\begin{aligned} \Phi(x) &= \phi(s)\phi(t) \\ \Psi(x) &= \{\psi(s)\psi(t), \psi(s)\phi(t), \phi(s)\psi(t)\} \end{aligned} \tag{2.69}$$

Leurs versions dilatées et translatées sont

$$\begin{aligned} \Phi_{j,k,l}(x) &= \phi_{j,k}(s)\phi_{j,l}(t) \\ \Psi_{j,k,l}(x) &= \{\psi_{j,k}(s)\psi_{j,l}(t), \psi_{j,k}(s)\phi_{j,l}(t), \phi_{j,k}(s)\psi_{j,l}(t)\} \end{aligned} \tag{2.70}$$

$\{\Phi_{0,k,l}, \Psi_{j,k,l}\}_{j\geq 0,k,l\in\mathbb{Z}}$ forme une analyse multi-résolution en 2D. En général, une base ondelette en d-dimension est composée de $2^d - 1$ fonctions différentes.

Pour faire une DWT en 2-Dimension, il y a 2 possibilités :

- **La méthode du quinconce.** Cette méthode consiste à utiliser des filtres de même dimension que le signal à analyser. Le nombre d'opérations nécessaires au filtrage est assez important. Pour cette raison on lui préfère bien souvent la deuxième méthode,
- **La méthode de la séparabilité du noyau [45, 46, 47].** Elle consiste à réaliser un FWT 1D dans chacune des dimensions du signal. Pour une image, on réalise donc une FWT sur chaque ligne puis une FWT sur les colonnes. Un des avantages de cette méthode est qu'elle est directive en fréquence :

On appelle $A^d_{2^j} f$ la version filtrée passe-bas de l'image originale f au niveau j et $D^k_{2^j} f$ l'image de détail k au niveau j.

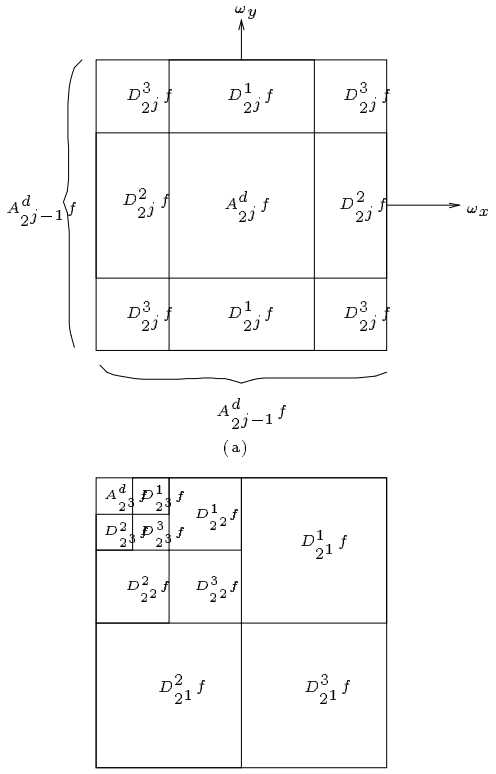
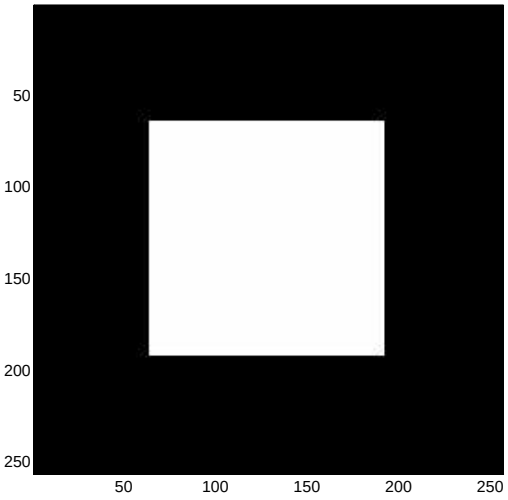
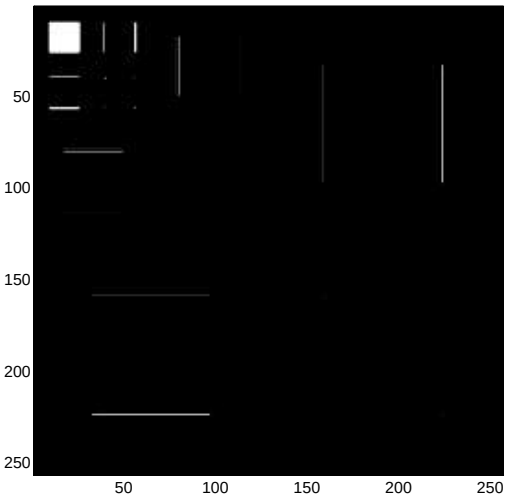


FIG. 2.6 - : *(a) Décomposition fréquentielle du support de l'image $A^d_{2^{j-1}} f$ en $A^d_{2^j} f$ et en images de détails $D^k_{2^j} f$. Les images $A^d_{2^j} f$ correspondent aux basses fréquences*

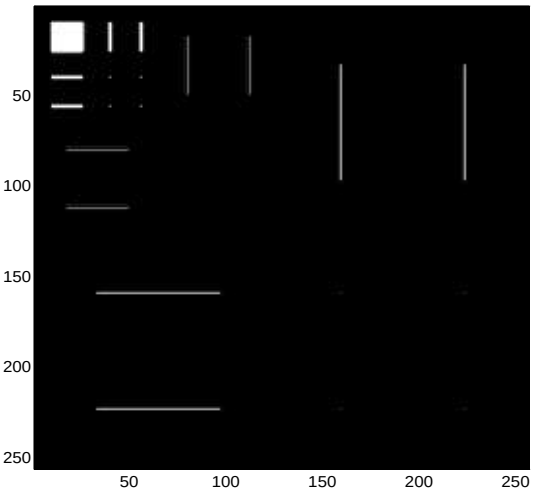
horizontales et verticales de $A^d_{2^{j-1}} f$. $D^1_{2^j} f$ donnent les hautes fréquences horizontales et les basses fréquences verticales, $D^2_{2^j} f$ donnent les hautes fréquences verticales et les basses fréquences horizontales, $D^3_{2^j} f$ donnent les hautes fréquences dans les deux directions. (b) Dispositions spatiales des images $A^d_{2^j} f$ et $D^2_{2^j} f$ de la représentation ondelette que nous utilisons.


50
100
150
200
250
50
100
150
200
250

(a)


50
100
150
200
250
50
100
150
200
250

(b)


50
100
150
200
250
50
100
150
200
250

(c)

FIG. 2.7 - : **Exemple de décomposition sur 3 niveaux avec un filtre D_3 (section 3.2.1).** *On remarquera combien la DWT 2D séparable est directive en fréquence. (a) Image originale. (b) Représentation ondelette sur 3 niveaux. Les pixels noirs, gris et blancs correspondent respectivement aux coefficients ondelettes négatifs, nuls et positifs. La disposition des images de détails est celle de la figure 2.6(b). (c) Image de la valeur absolue des coefficients de (b). Les pixels blancs et noirs correspondent repectivement aux coefficents de hautes valeurs et nuls. L'amplitude est la plus grande le long des bords dans chacune des orientations.*

La première méthode est plus lourde à implémenter du fait du filtrage 2D qui prend beaucoup de temps de calcul.
Quant à la deuxième méthode, elle est très largement utilisée; l'algorithme utilisé est le même que celui pour la FFT 2D, figure (2.8).

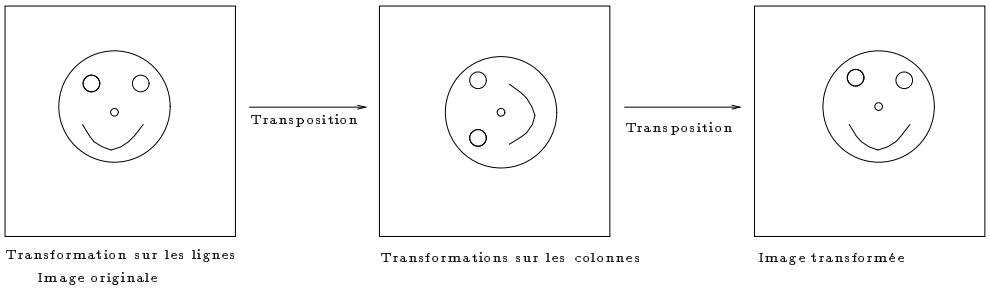


FIG. 2.8 - : *Algorithme DWT 2D séparable.*

Comme pour la DWT 1D, si on a une image $2^N \times 2^N$, on ne peut faire au maximum que N décompositions. Dans le cas bi-dimensionnel, ces algorithmes doivent être modifiés pour ne pas effectuer la décomposition sur N niveaux de chaque ligne ou de chaque colonne de l'image. Il suffit d'effectuer une décomposition et une seule par ligne et par colonne et par niveau. Puis on recommence avec 2 fois moins d'échantillons par colonne et par ligne (donc 4 fois moins d'échantillons au total).

La généralisation à un signal x-dimensionnel est immédiate en remarquant qu'il suffit donc de faire **1** FWT 1D par dimension pour chaque niveau de décomposition.

2.3.6 Paquets d'ondelettes

Les paquets d'ondelette généralisent tous les concepts précédents. On veut construire une librairie d'ondes modulées de laquelle peut être extraite toutes sortes de bases. En particulier, les bases ondelettes, les fonctions de Walsh, et des bases "paquets d'ondelettes" à oscillations rapides, [77, 76, 39]. Nous allons expliciter cette construction dans le cas de filtres orthogonaux.

Soient deux opérateurs W_{2n} et W_{2n+1} de $l^2(\mathbb{Z}) \to l^2(2\mathbb{Z})$:

$$\begin{aligned} W_{2n}(t) &= \sqrt{2}\sum_k h_k W_n(2t-k) \\ W_{2n+1}(t) &= \sqrt{2}\sum_k g_k W_n(2t-k) \end{aligned} \tag{2.71}$$

Algorithme

```
pour niveau 1(1)n
        pour chaque ligne
                faire une FWT sur 1 niveau
        fin

        transposer la matrice

        pour chaque colonne
                faire une FWT sur 1 niveau
        fin

        transposer la matrice
fin
```

TAB. 2.5 - : *FWT 2D.*

On voit clairement que pour $n = 0$, $W_0(x)$ n'est autre que $\phi(x)$ et $W_1(x)$ n'est autre que $\psi(x)$, voir section 3.2. Ces deux opérateurs constituent donc un arbre de bases $\{W_n\}$ sur lesquelles le signal va être projeté. Une fois l'arbre réalisé, on va chercher quelles sont les bases qui minimisent une fonction de coût. Au final, on conservera les coefficients ondelettes de ces bases et on aura autant de coefficients ondelettes que d'échantillons du signal original. Toutefois, suivant le critère et l'ondelette utilisé, on aura plus de coefficients négligables; d'òu un taux de compression plus élevé.

Fonctions de coût utilisées

Chacun peut définir sa propre fonction de coût. Le tout est qu'elle soit additive.

Définition *Une fonction $\mathcal{M}$ de $l^2(\mathbb{Z}) \to \mathbb{R}$ est appelée une fonction de coût additive si $\mathcal{M}(0) = 0$ et $\mathcal{M}(x_i) = \sum_i \mathcal{M}(x_i)$.*

Voici quelques exemples de fonctions employées :

1. **Nombres au-dessus d'un seuil.** Pour un seuil arbitraire ϵ fixé, on compte les éléments de valeur absolue supérieure à ϵ. Cela donne le nombre de coefficients nécessaires à la transmission du signal pour une précision ϵ.

2. **Concentration en norme l^p, $p < 2$.** On choisit un nombre arbitraire $p < 2$ et on définit $\mathcal{M}(x) = \|x\|_p$. On observe que plus la norme l^p d'une fonction d'énergie 1 est petite, plus concentrée est son énergie sur peu de coefficients.

3. **Entropie.** On définit l'entropie au sens de Shannon-Weaver d'une séquence $x = \{x_j\}$ par $\eta(x) = -\sum_j p_j \log p_j$, où $p_j = \frac{|x_j|^2}{\|x\|^2}$ et on impose $p \log p = 0$ si $p = 0$. Il ne s'agit pas d'une fonction de coût additive. Par contre, la fonction $\lambda(x) = -\sum_j |x_j|^2 \log |x_j|^2$ l'est. Par la relation $\eta(x) = \|x\|^{-2}\lambda(x) + \log \|x\|^2$, minimiser la précédente minimisera cette dernière. Un fait classique à propos de l'entropie

est que $\exp \eta(x)$ est proportionnel au nombre de coefficients nécessaires pour représenter un signal à une erreur moyenne quadratique fixée.

4. **Logarithme de l'énergie.** Soit $\mathcal{M}(x) = \sum_j \log |x_j|^2$ avec $\log 0 = 0$ où c'est nécessaire. Cela peut être interprété comme un processus de Gauss-Markov composé de N variables aléatoires gaussiennes de variance $\sigma_1^2 = |x_1|^2, \ldots, \sigma_N^2 = |x_N|^2$. La redéfinition du log à 0 est équivalente à ignorer toute composante inchangée dans le processus. Minimiser cette fonction conduit à la meilleure approximation de la base de Karhunen-Loéve, pour le processus, qui atteint le minimum global de $\mathcal{M}$ sur tout le groupe orthogonal.

Les fonctions de coût d'information additive sont des fonctionnelles sur un ensemble de bases orthonormales i.e. sur le groupe orthogonal. Dans le cas pratique, les ensembles sont compacts donc il existe un minimum global.

Formellement, soit $\mathcal{M}$ donné et soit x un vecteur d'un espace V. Soit B une base de la librairie (i.e. une base correspondant à un noeud de la décomposition par paquets d'ondelettes), et soit Bx une séquence de coefficients de x dans la base V.

Définition. *La meilleure base pour $x \in V$ relative à M est B telle que $\mathcal{M}(Bx)$ est minimale.*

Algorithme

Si la librairie est un arbre, alors on peut trouver la *meilleure base* par induction sur k. Soit $B_{n,k}$ la base de vecteur correspondant à l'intervalle de $\mathbb{N}$ de la forme $I_{n,k} = [2^k n, 2^k(n+1)[$ et soit $A_{n,k}$ la meilleure base pour x projeté sur les vecteurs de $B_{n,k}$.

Pour $k = 0$, il existe une et seule base, celle correspondant à $I_{n,0}$. Donc $A_{n,0} = B_{n,0}$ pour tout $n \geq 0$. On construit $A_{n,k+1}$ pour tout $n \geq 0$ ainsi :

$$A_{n,k+1} = \begin{cases} B_{n,k+1} & \text{si} \mathcal{M}(B_{n,k+1}x) < \mathcal{M}(A_{2n,k}x) + \mathcal{M}(A_{2n+1,k}x), \\ A_{2n,k} \oplus A_{2n+1,k}, & \text{sinon.} \end{cases} \tag{2.72}$$

TAB. 2.6 - : *Algorithme de recherche de la meilleure base.*

L'algorithme (2.6) conduit alors à la meilleure base relativement à $\mathcal{M}$.

Implémentation

Dans WaveLib 1.0, les paquets d'ondelette sont implémentés sous forme d'arbres un peu particulier dans le but d'optimiser la recherche des bases utiles. La décomposition par paquets consiste effectuer une décomposition du signal par les mêmes filtres qu'avant (orthogonaux ou biorthogonaux) mais cette fois-ci on décompose également les coefficients de détails :

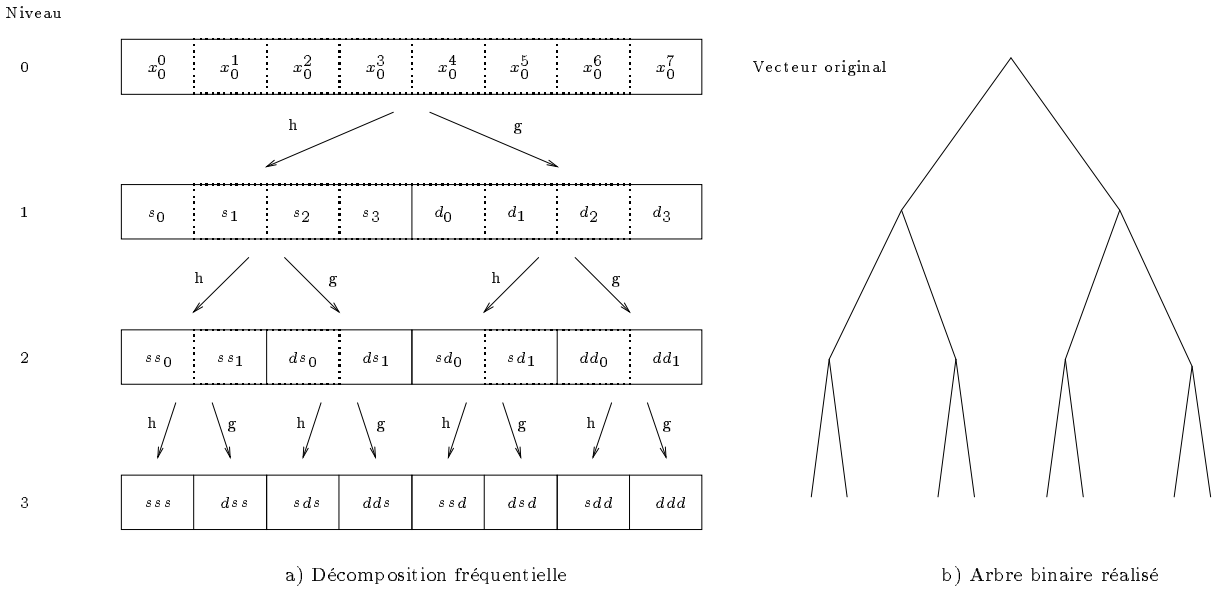


FIG. 2.9 - : *Décomposition par paquets d'ondelettes.*

Sur la figure 2.9b, on a généré un arbre binaire qui est une structure de pointeurs; chaque noeud père pointant sur 2 fils (cas d'un signal 1D). En fait cette structure, quoique très facilement implémentable récursivement, n'est pas la plus adaptée. Nous proposons celle-ci :

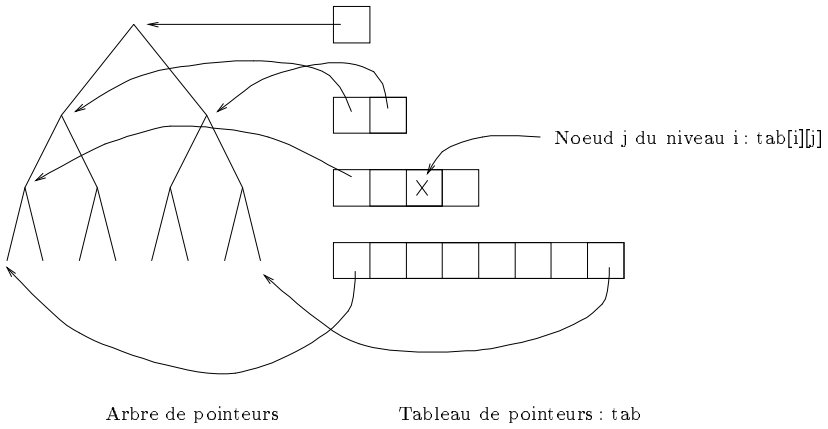


FIG. 2.10 - : *Arbres utilisés en paquets d'ondelettes.*

Le tableau de pointeur *tab* a autant de lignes que de niveaux désirés. Chaque ligne pointe sur un vecteur de pointeurs de structures. Chaque structure pointe sur un noeud de l'arbre. Cette structure est plus souple à gérer. En effet, elle prend moins de place en mémoire qu'un arbre binaire (pour des signaux 1D), quaternaire (images 2D) ou 2^N-aire (signaux N dimensionnel). De plus il est très facile de se déplacer dans l'arbre pour rechercher quelles sont les bases à retenir. Cette recherche serait très lourde si on le faisait avec un arbre de pointeurs : elle demanderait beaucoup plus de temps et bien plus de ressources auxilliaires.

Comme nous l'avons introduit, il ne suffit pas d'avoir un pointeur sur chaque noeud de l'arbre, on a également besoin d'*une fonction de coût* et d'*un booléen associé à chaque noeud* pour nous dire si ce noeud (cette base) a été retenu ou non. Par conséquent, voici la structure d'un élément du tableau :

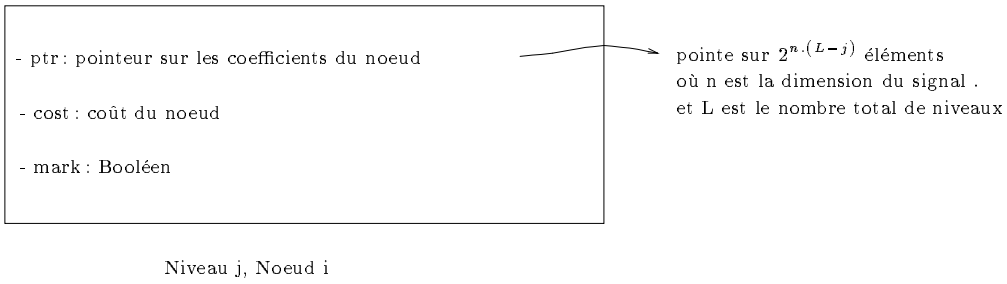


FIG. 2.11 - : *Elément du tableau associé à un noeud.*

Il faut remarquer que

- on a $2^{n.i}$ noeuds au niveau i,
- chaque noeud pointe sur $2^{n.(L-i)}$ coefficients,

avec $i \in [0, L]$, L nombre maximum de niveaux de décomposition et n nombre de dimensions (=1 si signal 1D, =2 si image 2D, ...).
Il existe différents types de méthodes permettant de trouver des bases utiles suivant un critère de coût.

1. **La méthode des bases ondelettes.**
   Il s'agit là de prendre uniquement les bases nécessaires à la FWT. On ne retient que les bases de détail après filtrage de la version passe-bas du signal original. On a alors le schéma suivant :

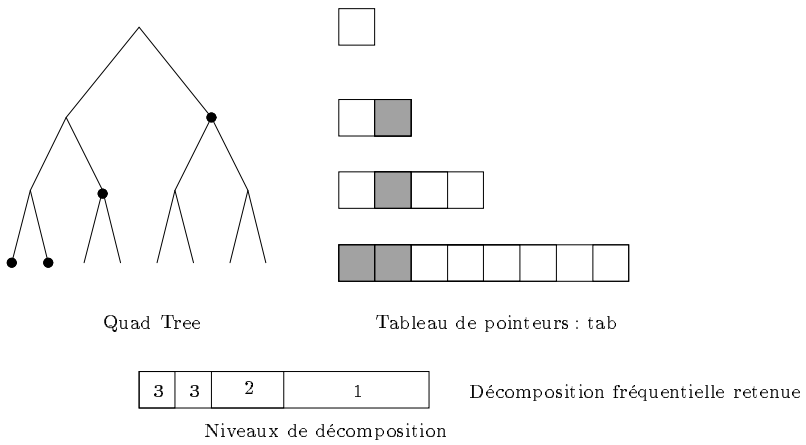


FIG. 2.12 - : *Méthode des bases ondelettes.*

En nD, on prendrait que les bases 1 à $2^n - 1$ de chaque niveau. Ici, la fonction de coût ne sert absolument pas à déterminer les bases utiles, elle ne sert qu'après

coup pour évaluer le coût de la transformation. Le coût de la FWT en langage C est donc :

$$cost = tab[L][0] \rightarrow cost + \sum_{i=1}^{L-1} \sum_{j=1}^{2^n-1} tab[i][j] \rightarrow cost \tag{2.73}$$

2. **La méthode du meilleur niveau.**
On parcourt l'arbre de niveau en niveau et on calcule le coût de chaque niveau.

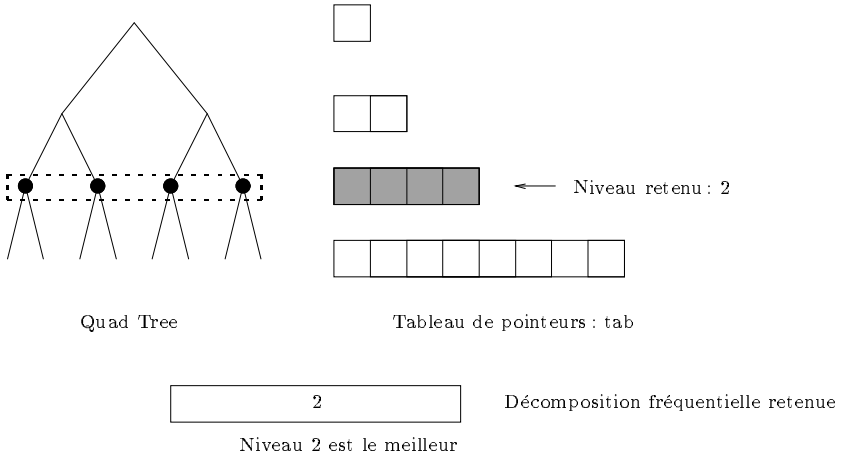


FIG. 2.13 - : *Méthode du meilleur niveau.*

Le coût de la transformation en langage C sera donc

$$cost = \min_{i=1,L} \sum_{j=0}^{2^{n.i}} tab[i][j] \rightarrow cost \tag{2.74}$$

où j indice les noeuds du tableau de pointeurs au niveau i.

3. **La méthode de la meilleure base.**[18, 19, 20, 21, 77]
Il s'agit de la méthode la plus communément employée car elle procure l'entropie la plus faible des trois méthodes.
On part de l'avant-dernier niveau de l'arbre (le niveau $L-1$) et on remonte vers la racine (niveau 0). On teste

$$\begin{cases} \text{Si coût}_{\text{père}} \leq \text{coût}_{\text{des fils}} & \Rightarrow \quad \text{père est conservé pas les fils. Le coût} = \text{coût}_{\text{père}} \\ \text{Sinon} & \Rightarrow \quad \text{seuls les fils sont conservés et le coût} = \text{coût}_{\text{des fils}}. \end{cases} \tag{2.75}$$

A la fin, on est garanti d'avoir, pour cette décomposition (avec les filtres utilisés), les bases qui vont procurer le coût d'information le plus bas (coût fonction de la fonction de coût employée bien sûr).

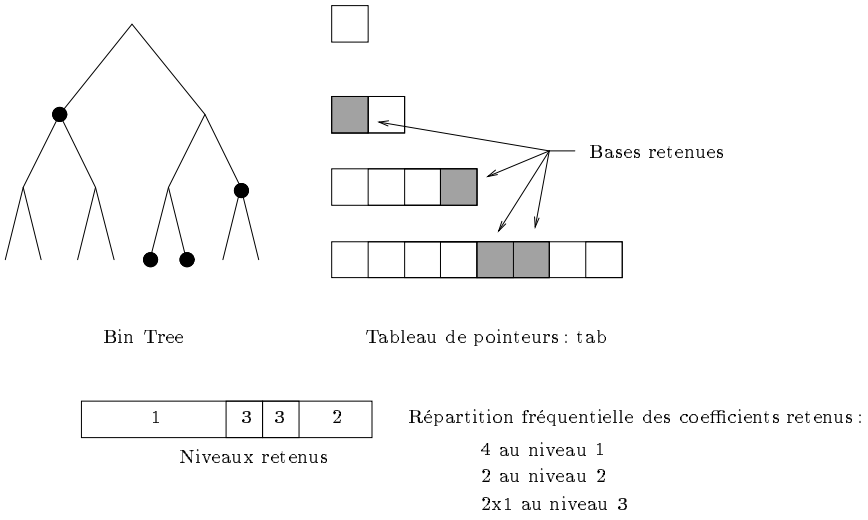


FIG. 2.14 - : *Méthode de la meilleure base.*

Notons que cette méthode est la plus générale et marche quelle que soit la dimension du signal (1D, 2D, ...) : chaque noeud sera décomposé en sous-arbre et on effectuera la recherche des bases contribuant au coût global minimal et ce quel que soit le nombre de noeuds de l'arbre. Grâce à cette méthode, on obtient le coût minimal de l'arbre. Cela crée un partitionnement non régulier en fréquence du plan temps-fréquence.

2.4 Ondelettes et bancs de filtres

2.4.1 Rappels sur les bancs de filtres

La théorie des ondelettes trouve ses fondements dans la théorie des bancs de filtres tellement utilisés en traitement du signal. L'idée est de séparer le signal original en plusieurs bandes de fréquence (typiquement basse fréquence et haute fréquence) pour mieux le compacter et le transmettre. Au récepteur, on reconstruit le signal en rassemblant ces diverses bandes, voir figure (2.15). Le problème est de savoir comment peut-on avoir un signal reconstruit $\hat{X}$ identique au signal original X?

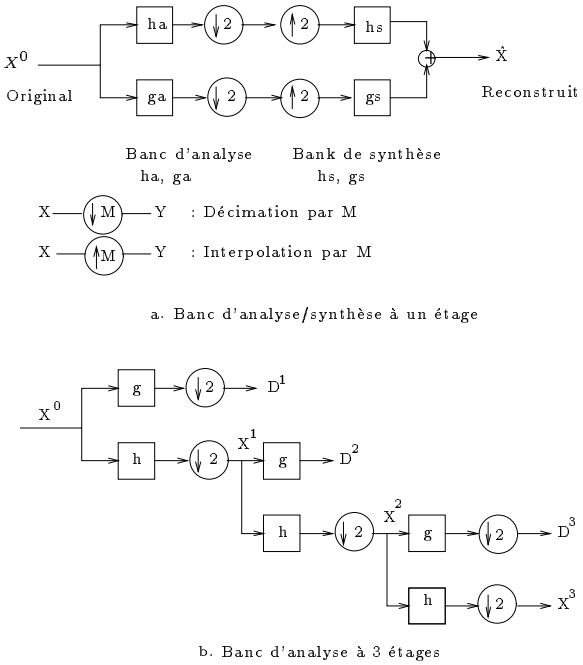


FIG. 2.15 - : *Banc de filtres*

En utilisant les notations de la figure (2.15), on rappelle qu'une décimation par M implique (notation avec la transformée en z)

$$Y(z) = \sum_{k=0}^{M-1} X(z^{1/M}.e^{\frac{-2\pi jk}{M}}),$$

et une interpolation par M

$$Y(z) = X(z^M)$$

On peut donc écrire le banc de filtres 2.15a sous la forme

$$\hat{X}(z) = \frac{1}{2}\left[\begin{array}{cc} Hs(z) & Gs(z) \end{array}\right].\left[\begin{array}{cc} Ha(z) & Ha(-z) \\ Ga(z) & Ga(-z) \end{array}\right]\left[\begin{array}{c} X(z) \\ X(-z) \end{array}\right] \qquad (2.76)$$

D'après l'équation ci-dessus, de façon à éliminer la contribution de $X(-z)$ (qui représente une version aliasée du signal) au signal reconstruit, il est nécessaire et suffisant que les filtres soient tels que

$$\left[\begin{array}{cc} Hs(z) & Gs(z) \end{array}\right] = C(z).\left[\begin{array}{cc} Gs(-z) & -Ga(-z) \end{array}\right]$$

et, si on appelle

$$H_m(z) = \left[\begin{array}{cc} Ha(z) & Ha(-z) \\ Ga(z) & Ga(-z) \end{array}\right]$$

il vient

$$\det H_m(z) = Ha(z)Ga(-z) - Ha(-z)Ga(z) = P(z) - P(-z)$$

Alors, pour avoir reconstruction parfaite, avec des filtres à réponse impulsionnelle finie (RIF), il est nécessaire et suffisant que

$$\det H_m(z) = c.z^{-2l-1}, \quad \text{où} \quad l \in \mathbb{Z} \tag{2.77}$$

En pratique, on prend $c = 2$.

Dans le cas de filtres orthogonaux, l'énergie des coefficients transmis est la même que celle du signal original. Tandis que dans le cas où les filtres sont biorthogonaux, ce n'est pas le cas. Par contre, dans les deux cas, on reconstruira un signal identique à l'original mais avec une phase pouvant être différente. Comme on le vera en section 3.3, l'intérêt des filtres biorthogonaux est la liberté additionnelle d'avoir des filtres à phase linéaire de longueur arbitraire. En effet, il est bien connu [74, 24] que le seul banc de filtres RIF réels ayant une phase linéaire est celui avec des filtres de Haar. Aussi, pour obtenir des RIF réels plus longs, et avoir toujours une phase linéaire, il faut s'affranchir de l'orthogonalité.

2.4.2 Analogie avec l'analyse multirésolution

Nous allons expliquer la version bancs de filtres de l'analyse multirésolution dans le cas de filtres orthogonaux. La généralisation aux filtres biorthogonaux est immédiate.

D'un point de vue traitement du signal, une ondelette est un filtre passe-bande. De plus, dans le cas dyadique, il s'agit d'un filtre octave-bande. Alors, la transformée ondelette peut être interprétée comme un filtrage à Q-constant avec un ensemble de filtres octave-bande suivis d'un échantillonnage aux fréquences de Nyquist respectives (correspondant à la bande passante d'un octave particulier). Il est donc clair qu'en ajoutant des octaves dans des fréquences plus hautes, on ajoute plus de détails ou plus de résolution, au signal. Soit V_0 l'espace de toutes les fonctions bandes-limitées dans les fréquences $[-\pi, +\pi]$, alors l'ensemble des fonctions $\{\phi(x-k), \quad k \in \mathbb{Z}\}$ forme une base orthonormale de V_0. Similairement, soit V_{-1} l'ensemble des fonctions bandes-limitées dans les fréquences $[-2\pi, +2\pi]$, alors l'ensemble des fonctions $\{\phi(2x-k), \quad k \in \mathbb{Z}\}$ appartient à V_{-1}. Or,

$$V_{-1} = V_0 + W_0,$$

alors W_0 est l'espace des fonctions bandes-limitées de fréquences $[-2\pi, \pi] \cup [\pi, 2\pi]$. En répétant le schéma de l'analyse résolution, V_i est l'espace des fonctions bandes-limitées de fréquences dans l'intervalles $[-2^i\pi, 2^i\pi]$ et W_i l'espace des fonctions passe-bandes de fréquences dans l'intervalle $[-2^{-i+1}\pi, -2^{-i}\pi] \cup [2^{-i}\pi, 2^{-i+1}\pi]$. Graphiquement, on a le schéma suivant

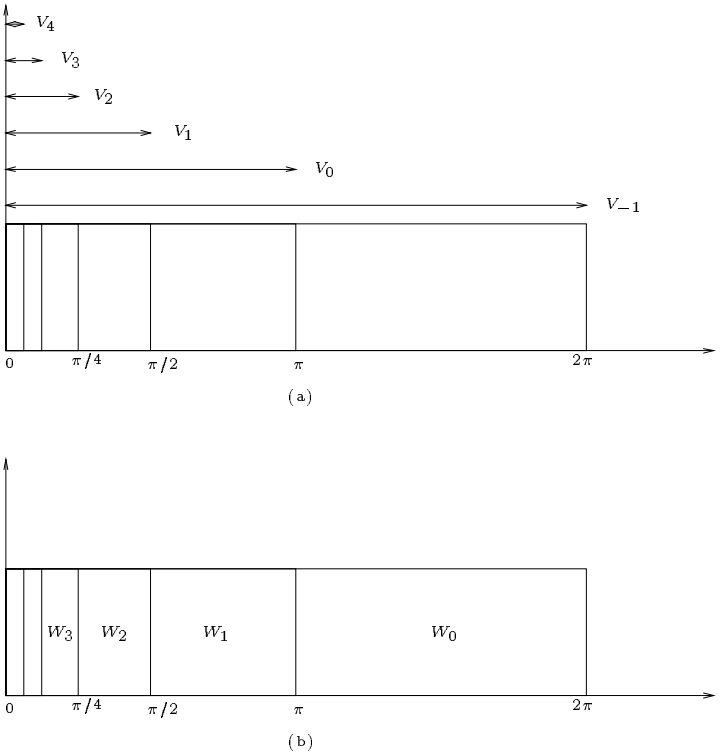


FIG. 2.16 - : **Subdivision idéale de l'espace de Fourier avec $\phi(x) = \mathbf{sinc}(x)$.** *(a) Division en sous-espaces V_i. Notons que $V_{i+1} \subset V_i$. (b) Division en sous-espaces W_i. Notons qu'on a bien $V_{i-1} = V_i + W_i$.*

2.5 Analyse temps-fréquence

On appelle pavage temps-fréquence d'une fonction de base particulière une région du plan qui contient l'énergie des coefficients d'une fonction projetée sur cette base; ces fonctions de base étant localisées dans un intervalle temporel et un intervalle fréquentiel. La pavage montre également les échantillonnages en temps et en fréquence vu qu'il indique où sont localisées en temps et en fréquence les fonctions de base. La représentation sous forme rectangulaire est purement symbolique car aucune fonction n'a de support compact et en temps et en fréquence. Pour une fonction $\phi_{jk}(x) = 2^{-j/2}\phi(2^{-j}x - k)$, le paramètre d'échelle j est lié aux résolutions temporelle et fréquentielle de la fonction de base : plus j est grand, i.e. plus la taille de la fonction de base est grande, meilleur est la localisation en fréquence (et inversement, moins bonne est sa localisation temporelle).

On suppose que l'ondelette ψ et sa transformée de Fourier $\hat{\psi}$ sont des fonctions de centres finis $\bar{x}$ et $\bar{\omega}$ et de rayons finis Δ_x et Δ_ω définis par

$$\begin{aligned} \bar{x} &= \frac{1}{\|\psi\|_{L_2}^2} \int_{-\infty}^{+\infty} x|\psi(x)|^2 dx, \\ \Delta_x^2 &= \frac{1}{\|\psi\|_{L_2}^2} \int_{-\infty}^{+\infty} (x - \bar{x})|\psi(x)|^2 dx, \end{aligned}$$

et de même pour $\bar{\omega}$ et Δ_ω. Bien que la variable x représente ou le temps ou l'espace, nous choisissons de la référencer comme une variable temporelle. D'après (2.53) ou (2.54), on peut montrer que la CWT au point (a, b) contient essentiellement de l'information dans l'intervalle temporel $[b + a\bar{x} - a\Delta_x, b + a\bar{x} + a\Delta_x]$ et dans l'intervalle fréquentiel $[(\bar{\omega} - \Delta_\omega)/a, (\bar{\omega} + \Delta_\omega)/a]$. Ces deux intervalles définissent une *fenêtre temps-fréquence*. Ses largeur, hauteur et position sont gouvernées par a et b. Son aire est constante: $4\Delta_x\Delta_\omega \geq 2$, de part le principe d'incertitude d'Heisenberg. Ces fenêtres temps-fréquence sont appelées *boites d'Heisenberg*.

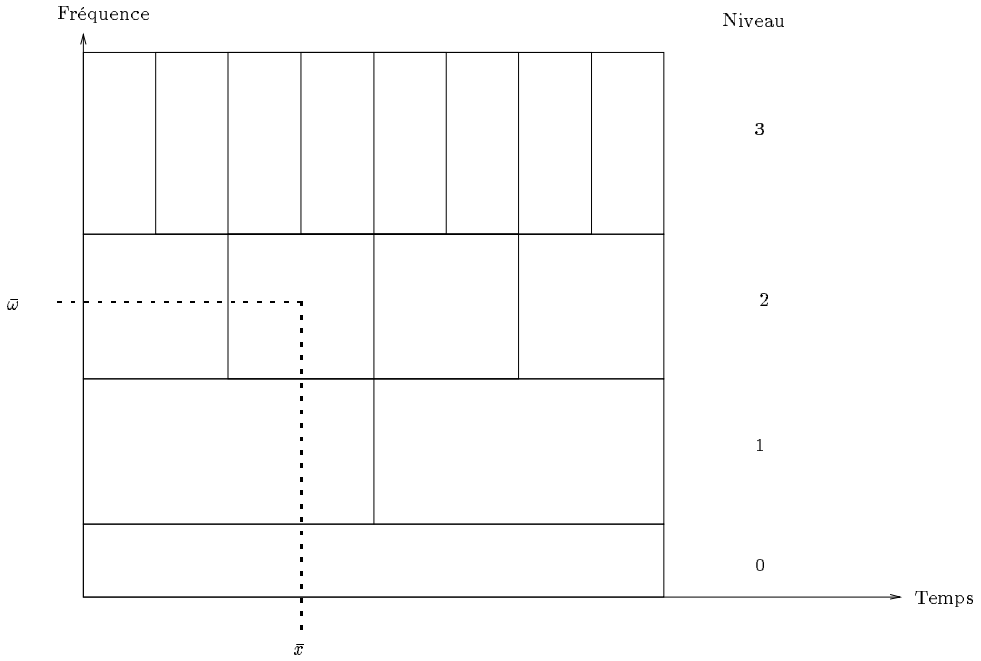


FIG. 2.17 - : *Analyses temps-fréquences: boites d'Heisenberg.*

A chaque boite correspond une ondelette dont la position en temps et en fréquence est le centre de la boite et avec une incertitude relative à la longueur des côtés du rectangle. L'aire de chaque rectangle est constante. Pour cela, en théorie des bancs de filtres, on appelle ces filtres *filtres à Q constant*. Au niveau 0, on voit que l'analyse est fine en fréquence mais large dans le temps. En effet, une basse fréquence correspond à un signal variant peu dans le temps. Par contre, un signal présentant une transistion importante dans ses coefficients, sera analysé dans les hautes fréquences.

Voici quelques exemples de pavage du plan temps-fréquences pour différents types de bases

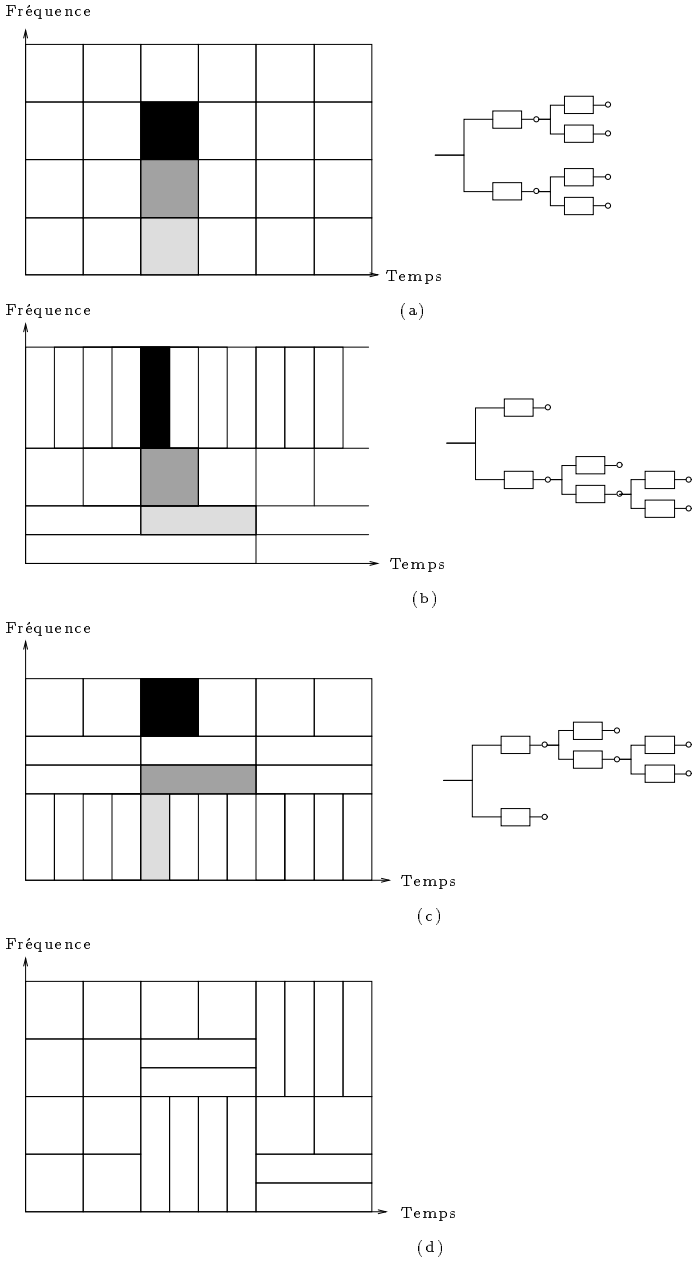


FIG. 2.18 - : **Exemples de pavages temps-fréquence et banc de filtres associés.** *(a) Pavage de la Transformée de Fourier Glissante. (b) Pavage des ondelettes. (c) Pavage des paquets d'ondelette. (d) Pavage généralisé s'adaptant en temps et en fréquence [35].*

Le pavage des ondelettes (2.18)(b) présenté est celui correspondant à un Dirac juste au centre de la bande de fréquence la plus fine.

La caractéristique principale d'un pavage par paquets d'ondelettes est qu'il produit une séparation arbitraire en fréquence, qui peut être adapté au signal. Bien que les paquets d'ondelettes créent un découpage binaire des fréquences (avec une résolution temporelle associée), elles ne changent pas dans le temps. Le dernier type

de pavage est le plus général possible, il consiste à effectuer un pavage adaptatif et dans le temps et dans l'espace. Dans ce cas il est important de noter qu'on utilise des bases variant dans le temps qui sont signal-dépendante. Ainsi, à un signal donné correspondera le meilleur ensemble de fonctions de base pour l'expansion d'un signal particulier (sur une librairie de bases données). Cela est en tout point similaire à la transformation optimale de Karhunen-Loève.

On notera que théoriquement, si le signal est une fonction d'échelle et rien d'autre, l'énergie des coefficients transformés devrait être localisée uniquement dans un seul rectangle. Or, en pratique ce n'est pas du tout le cas. Des expériences ont montré que l'algorithme est trés sensible à la phase du signal: si le signal est une fonction d'échelle bien positionnée, elle sera localisée dans un rectangle. Mais, si on la translate un petit peu alors l'énergie des coefficients ondelette se répartie sur plusieurs rectangles. Cette sensibilité à la phase est une caractéristique normale des fonctions de bases qui ne sont pas invariantes par translation [35].

2.6 Algorithmes de FWT uni-dimensionnelle

Après ces quelques explications, nous donnons un pseudo-code de FWT uni-dimensionnelle périodique des plus classiques et son inverse [39]. Ces algorithmes sont donnés dans le cas général où les filtres du banc d'analyse h et g sont différents des filtres du banc de synthèse $\tilde{h}$ et $\tilde{g}$.

On suppose que len_ha coefficients h_k non-nuls commencent à l'index $k = min\_ha$. On suppose de même pour g_k, $\tilde{h}_k$, $\tilde{g}_k$ de longeur respective len_ga, len_hs et len_gs commençant aux indices min_ga, min_hs et min_gs. Ces coefficients sont stockés dans quatre vecteurs:

$$ha[k] = ah_{k+min\_ha}, \quad ga[k] = ag_{k+min\_ga}, \quad hs[k] = b\tilde{h}_{k+min\_hs}, \quad gs[k] = b\tilde{g}_{k+min\_gs},$$

où $ab = 2$ (généralement $a = b = sqrt2$). On commence avec 2^n coefficients s_l^n d'une fonction de V_n que l'on peut donc décomposer en n étapes de l'algorithme. Ils sont initialement dans le vecteur s_l. Les coefficients ondelettes calculés sont stockés dans le vecteur w de telle sorte que

$$w = \left[s_0^0 d_0^0 d_0^1 d_1^1 d_0^2 \cdots d_3^2 \cdots d_0^{n-1} \cdots d_{2^{n-1}-1}^{n-1} \right].$$

Ces algorithmes ont été écrits dans le but de réduire les opérations dans les boucles imbriquées. Ils ne sont toutefois pas totalement optimisés de façon à ne pas affecter la lisibilité. La notation $a(b)c$ signifie $a, a+b, a+2b, \cdots, c$ et floor(a) arrondit a à l'entier le plus proche vers $-\infty$.

Voici l'algorithme de reconstruction à partir du vecteur de coefficients ondelettes w:

Algorithme

```
for j ← n − 1(−1)0
    w[0(1)2^{j+1} − 1] ← 0
    for l ← 0(1)2^j − 1
        i ← (2 ∗ l + min_hs) mod 2^{j+1}
        for k ← 0(1)len_hs
            w[l] ← w[l] + hs[k] ∗ v[i]
            i ← (i + 1) mod 2^{j+1}
        end for
        i ← (2 ∗ l + min_gs) mod 2^{j+1}
        ls ← l + 2^j
        for k ← 0(1)len_gs
            w[ls] ← w[ls] + gs[k] ∗ v[i]
            i ← (i + 1) mod 2^{j+1}
        end for
    end for
    v ← w[0(1)2^j − 1]
end for
```

TAB. 2.7 - : *FWT standard sur n niveaux.*

Algorithme

```
for j ← 1(1)n
    v[0(1)2^j − 1] ← 0
    for k ← 0(1)2^j − 1
        i ← (floor((k − min_ha)/2)) mod 2^{j−1}
        lb ← (k − min_ha) mod 2
        for l ← lb(2)len_ha
            v[k] ← v[k] + ha[l] ∗ w[i]
            i ← (i − 1) mod 2^{j−1}
        end for
        i ← (floor((k − min_ga)/2)) mod 2^{j−1}
        lb ← (k − min_ga) mod 2
        for l ← lb(2)len_ga
            v[k] ← v[k] + ga[l] ∗ w[i + 2^{j+1}]
            i ← (i − 1) mod 2^{j−1}
        end for
    end for
    w[0(1)2^j − 1] ← v
end for
```

TAB. 2.8 - : *IFWT standard sur n niveaux.*

Chapitre 3

Différents types d'ondelettes

3.1 Propriétés des ondelettes

Dans le chapitre précédent, nous avons déjà discuté des propriétés essentielles de l'analyse multirésolution. Nous allons rappeler les propriétés importantes souhaitées pour une ondelette.

1. **Orthogonalité.** Si les ondelettes sont orthogonales, la FWT est parfaite et la décomposition numérique est stable. Si l'analyse multi-résolution est orthogonale (cela inclut également les ondelettes biorthogonales), les opérateurs de projection dans les différents sous-espaces conduisent à des approximations optimales au sens de $L^2(\mathbb{R})$.

2. **Support compact.** Si la fonction d'échelle ϕ et l'ondelette ψ sont à support compact, les filtres H et G sont des filtres à réponse impulsionnelle finie, ce qui est requis dans les implémentations. Si elles ne sont pas à support compact, une décroissance rapide est souhaitable de sorte que les filtres puissent être raisonnablement approchés par des filtres à réponse impulsionnelle finie.

3. **Coefficients rationnels.** Pour une implémentation informatique, on souhaite que les coefficients des filtres h_k et g_k soient rationnels ou mieux dyadiques. En effet, diviser par une puissance de 2 sur un ordinateur correspond à un simple décalage de bits (donc très rapide).

4. **Symétrie.** Si la fonction d'échelle et l'ondelette sont (anti-) symétriques, alors les filtres sont à phase linéaire. S'il on n'a pas cette propriété, cela entraine une distorsion de phase lors de la reconstruction.

5. **Régularité.** La régularité d'une ondelette est importante pour des applications de compression. La compression est souvent réalisée en mettant les coefficients d_l^j à zéro, cela correspond à rejeter la composante $d_l^j \psi_{j,l}(x)$ de la fonction originale. Si la fonction originale est une image et que l'ondelette n'est pas assez régulière, l'erreur de reconstruction peut facilement être détectée par l'oeil humain. De cet argument, on en conclut que la régularité des fonctions à la décomposition est plus importante qu'à la reconstruction. De plus, plus de régularité implique une meilleure localisation fréquentielle des filtres. Finalement, des bases de fonctions régulières sont souhaitées pour des applications d'analyse numériques impliquant des dérivées.

6. **Nombre de moments nuls.** Comme nous l'avons dit auparavant, cela peut être important pour la détection de singularités et la caractérisation d'espaces réguliers. Il détermine le taux de convergence d'approximations ondelettes de fonctions régulières. Le nombre de moments nuls caractérise également la régularité de l'ondelette.

7. **Expression analytique.** Déjà mentionné auparavant, on ne dispose pas en général de l'expression analytique de ϕ ou ψ . Dans certains cas, l'expression analytique est disponible et c'est très utile. En analyse harmonique, l'expression analytique de la transformée de Fourier est particulièrement utile.

8. **Interpolation.** Si la fonction d'échelle satisfait

$$\phi(k) = \delta_k, \quad \text{pour} \quad k \in \mathbb{Z}$$

alors il est trivial de trouver la fonction de V_j qui interpole les données échantillonées sur une grille de pas 2^{-j} vu que ce sont simplement les valeurs de la fonction échantillonnée.

On s'en doute, il n'est pas possible de construire des ondelettes ayant toutes ces propriétés, il faudra faire un compromis. Voici quelques solutions.

3.2 Ondelettes orthogonales

Nous commençons la construction d'ondelettes orthogonales en introduisant l'*analyse multirésolution orthogonale*. Il s'agit d'une analyse multirésolution où les espaces W_j sont définis comme étant le complément orthogonal de V_j dans V_{j-1}. Par conséquent tous les espaces $W_j, \quad j \in \mathbb{Z}$ sont mutuellement orthogonaux, les projections P_j et Q_j sont orthogonales et l'expansion

$$f(x) = \sum_j Q_j f(x)$$

est une expansion orthogonale. Pour qu'une analyse multirésolution soit orthogonale, une condition suffisante est

$$W_0 \perp V_0,$$

ou

$$\langle \psi, \phi(\cdot - l) \rangle = 0, \quad l \in \mathbb{Z}.$$

En utilisant la formule sommatoire de Poisson, on voit que cette condition est essentiellement équivalente à:

$$\forall \omega \in \mathbb{R} \quad : \quad \sum_k \hat{\psi}(\omega + k2\pi)\overline{\hat{\phi}(\omega + k2\pi)} = 0 \tag{3.1}$$

Une *fonction d'échelle orthogonale* est une fonction ϕ telle que l'ensemble $\{\phi(x-l), l \in \mathbb{Z}\}$ est une *base orthogonale*

$$\langle \phi, \phi(\cdot - l) \rangle = \delta_l, \quad l \in \mathbb{Z}. \tag{3.2}$$

Avec une telle fonction ϕ, la collection de fonction $\{\phi(x-l), \quad l \in \mathbb{Z}\}$ forme une base orthonormale de V_0 et la collection de fonctions $\{\phi_{j,l}, \quad l \in \mathbb{Z}\}$ est une base orthonormale de V_j. Par la formule de Poisson, (3.2) devient,

$$\forall \omega \in \mathbb{R} \quad : \quad \sum_k |\hat{\phi}(\omega + k2\pi)|^2 = F(\omega) = 1. \tag{3.3}$$

Or, d'après (2.5), le fait que ϕ forme un base de Riesz correspond à l'existence de constantes positives A et B telles que,

$$0 < A \leq F(\omega) \leq B < \infty$$

Un exemple des plus classiques est la fonction de Haar :

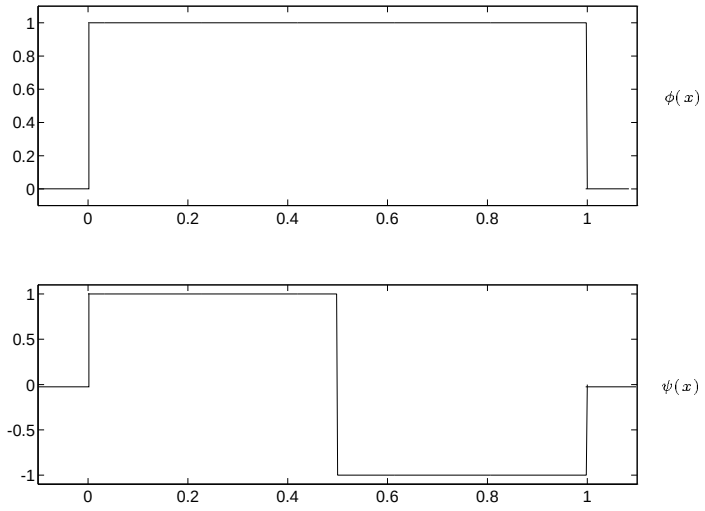


FIG. 3.1 - : *Fonctions de Haar.*

$$\phi_{jk}(x) = \begin{cases} 2^{-j/2} & \text{pour} \quad 2^j(k-1) \leq x < 2^j k \\ 0 & \text{ailleurs,} \end{cases}$$

$$\psi_{jk}(x) = \begin{cases} 2^{-j/2} & \text{pour} \quad 2^j(k-1) \leq x < 2^j(k-1/2) \\ -2^{-j/2} & \text{pour} \quad 2^j(k-1/2) \leq x < 2^j(k+1) \\ 0 & \text{ailleurs.} \end{cases}$$

Cette fonction a les propriétés suivantes :

$$\begin{aligned} \int \psi(x)dx &= 0 \\ \int \phi(x)dx &= 1 \end{aligned} \tag{3.4}$$

Comme on peut le voir, cette fonction n'a qu'un seul moment nul. Si on effectue une projection d'une fonction sur cette base, la projetée aura une allure de fonction en escalier. Si l'on désire plus de régularité, il faut donc utiliser des fonctions avec plus de moments nuls (figure 3.2).

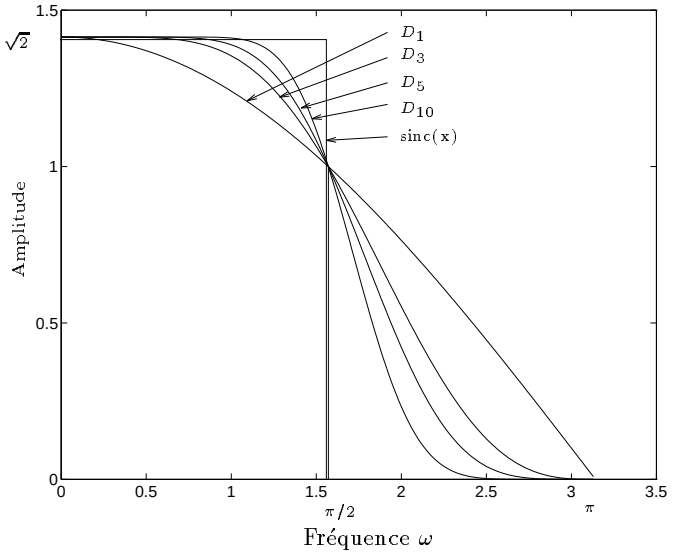


FIG. 3.2 - : **Transformées de Fourier de filtres de Daubechies D_N.** *Noter que plus le nombre de moments est grand, plus on tend à approcher le filtre passe-bas idéal sinc(x).*

3.2.1 Ondelettes de Daubechies

De nombreux chercheurs ont proposé des algorithmes de synthèse de filtres [48, 73, 6, 75], ... Daubechies [24] proposa en 1988 une construction d'ondelettes orthogonales à support compact tout comme les fonctions de Haar (notez que le Daubechies 1 ou D_1 n'est autre que l'ondelette de Haar, figure 3.1).

Une ondelette de Daubechies ayant M moments nuls vérifie l'équation de dilatation (2.6)

$$\begin{aligned} \phi(x) &= \sqrt{2} \sum_{k=0}^{2M-1} h_{k+1} \phi(2x-k) \\ \psi(x) &= \sqrt{2} \sum_{k=0}^{2M-1} g_{k+1} \phi(2x-k) \end{aligned}$$

et les coefficients h_k et $g_k = (-1)^k h_{-k+1}$ pour $k = 1, 2, \ldots, 2M$ satisfont les contraintes suivantes [67, 3]:

- **Consistance.** Une condition nécessaire et suffisante pour l'existence d'une solution à l'équation de dilatation est

$$\sum_{k=1}^{2M} h_k = \sqrt{2} \tag{3.5}$$

ce qui entraine

$$\sum_n h(2n) = \sum_n h(2n+1) = \sqrt{2} \tag{3.6}$$

- **Orthogonalité.** Les translatées et les dilatées entières de ϕ et ψ données par

$$\phi_{jk}(x) = 2^{-j/2} \phi(2^{-j}x - k + 1)$$

et

$$\phi_{jk}(x) = 2^{-j/2}\psi(2^{-j}x - k + 1)$$

forment une famille orthogonale si les coefficients d'échelle satisfont

$$\sum_{k=1}^{2M} h_k h_{k-2m} = \delta_{0m}, \quad \forall m. \tag{3.7}$$

Cela assure les relations d'orthogonalité

$$\begin{aligned} \int \phi(x-k)\phi(x-l)dx &= \delta_{kl} \\ \int \psi(x-k)\psi(x-l)dx &= \delta_{kl} \end{aligned}$$

La relation

$$\int \phi(x-k)\psi(x-l)dx = 0$$

n'est pas une conséquence de cette condition mais plutôt une définition des g_k.

– **Approximation.** La contrainte

$$\sum_{k=1}^{2M}(-1)^k k^m h_{k+1} = 0, \quad m = 0, 1, \ldots, M-1 \tag{3.8}$$

conduit au résultat que $\psi(x)$ a M moments,

$$\int \psi(x)x^m dx = 0, \quad m = 0, 1, \ldots, M-1.$$

Cela signifie qu'il est possible d'approcher une fonction régulière avec une erreur $\mathcal{O}(\alpha^M)$ par combinaisons linéaires des ondelettes à l'échelle α.

– **Régularité.**
Le domaine de $\phi(x)$ est $[0, 2M-1]$
Le domaine de $\psi(x)$ est $[-M+1, M]$
Notons que si on définit $g_k = (-1)^k h_{p-k}$ où p est une entier impair, cela à l'effet de changer le domaine de définition de $\phi(x)$ et les indices non nuls g_k. Choisir $p = 2M+1$ peut être intéressant car dans ce cas le domaine de définition de $\psi(x)$ est le même que celui de $\phi(x)$, ce qui peut s'avérer pratique en informatique vu que les tableaux commencent aux indices supérieurs ou égaux à zéro.

Résolution des contraintes et implémentation

Les équations (3.5), (3.6), (3.7) peuvent être réécrites

$$|m_0(\omega)|^2 + |m_0(\omega+\pi)|^2 = 1 \tag{3.9}$$

avec

$$m_0(\omega) = 2^{-1/2}\sum_n h(n)e^{in\omega} = \left[\frac{1}{2}(1+e^{i\omega})\right]^N Q(e^{i\omega}) \tag{3.10}$$

Daubechies [24] en arrive à sa proposition 4.5:

Proposition 4.5 [24] *Tout polynôme trigonométrique m_0 solution de (3.9) est de la forme*

$$m_0(\omega) = 2^{-1/2} \sum_n h(n) e^{in\omega} = \left[\frac{1}{2}(1+e^{i\omega})\right]^N Q(e^{i\omega}) \tag{3.11}$$

où $N \in \mathbb{N}, N \geq 1$ et où Q est un polynôme à coefficients réels tel que

$$|Q(e^{i\omega})|^2 = \sum_{k=0}^{N-1} \left(\begin{array}{c} N-1+k \\ k \end{array} \right) \sin^{2k} \frac{1}{2}\omega + \left[\sin^{2N} \frac{1}{2}\omega\right] R(\frac{1}{2}\cos\omega) \tag{3.12}$$

où R est un polynôme impair.

Toutefois, ce qui nous intéresse c'est $Q(e^{i\omega})$ et non son module. Aussi, pour en extraire la racine, on utilise une technique dite de factorisation spectrale (figure 3.1)

Algorithme

1. choisir $N \in \mathbb{N}^*$,
2. choisir un polynôme impaire R, sous les restrictions suivantes

$$P_N(y) + y^N R(\frac{1}{2} - y) \geq 0 \quad \text{pour} \quad 0 \leq y \leq 1 \tag{3.13}$$

et,

$$\sup_{0 \leq y \leq 1} \left[P_N(y) + y^N R(\frac{1}{2} - y)\right] < 2^{2(N-1)}, \tag{3.14}$$

3. calculer $|Q(e^{i\omega})|^2$ par (3.12),
4. ne retenir que les zéros à l'intérieur du cercle unité (ou uniquement ceux à l'extérieur),
5. à partir de ces racines, on refactorise le polynôme Q,
6. former

$$\begin{aligned} m_0(\omega) &= \left[\frac{1}{2}(1+e^{i\omega})\right]^N Q(e^{i\omega}) \\ &= e^{\frac{iN\omega}{2}} \left[\frac{e^{i\frac{\omega}{2}} + e^{-i\frac{\omega}{2}}}{2}\right]^N Q(e^{i\omega}) \\ &= e^{\frac{iN\omega}{2}} \cos^N \frac{\omega}{2} Q(e^{i\omega}) \end{aligned}$$

TAB. 3.1 - : *Algorithme de génération de m_0 .*

Il est à noter que dans (3.12),

$$\sin^{2k}(\frac{\omega}{2}) = \left[\frac{e^{\frac{i\omega}{2}} - e^{-\frac{i\omega}{2}}}{2}\right]^{2k}$$

$$= \left[\frac{e^{i\omega} + e^{-i\omega} - 2}{2}\right]^k \tag{3.15}$$

En posant $z = e^{-i\omega}$,

$$\sin^{2k}(\frac{\omega}{2}) = \left[\frac{z^{-1} - 2 + z}{2}\right]^k.$$

On peut donc réécrire (3.12) comme:

$$|Q(z)|^2 = \sum_{k=0}^{N-1} \begin{pmatrix} N-1+k \\ k \end{pmatrix} \left[\frac{z^{-1} - 2 + z}{2}\right]^k + \left[\frac{z^{-1} - 2 + z}{2}\right]^N R(z) \tag{3.16}$$

Pour les filtres de Daubechies $R \equiv 0$ et,

$$m_0(z) = \left[\frac{1 + z^{-1}}{2}\right]^N Q(z). \tag{3.17}$$

On a donc au plus $2N$ valeurs pour les coefficients de h_N

$$h_N(n) = \sqrt{2} m_0(n), \quad \text{avec} \quad 0 \leq n \leq 2N - 1.$$

3.2.2 Coiflets

Les coiflets sont un cas particuler des ondelettes de Daubechies. Elles ont été demandées par Beylkin, Coifman et Rokhlin [8, 9]. En fait Beylkin désirait comprimer des opérateurs intégraux qui sont la plupart du temps des matrices denses (cf. section **??**) en des matrices bandes dont la largeur de bande conditionnerait l'erreur faite sur l'approximation. Le problème était de comprimer les coefficients de l'espace V_L. Comme vu précédemment, plus ψ a de moments, plus on aura de coefficients ondelettes nuls. Cette observation serait donc très utile si ϕ avait elle aussi des moments nuls (sauf le premier bien sûr).

Par conséquent, les coiflets vérifient

$$\begin{aligned} |\text{supp}\phi| &= |\text{supp}\psi| = 2M - 1, \\ \int \phi(x)dx &= 1, \\ \int \phi(x)x^l dx &= 0 \quad \text{pour} \quad l = 1, \ldots, L-1, \\ \int \psi(x)x^l dx &= 0 \quad \text{pour} \quad l = 1, \ldots, L-1. \end{aligned} \tag{3.18}$$

Daubechies et Lagarias [23] construisirent ces ondelettes où l'orthonormalité des bases est très importante en analyse numérique. De manière équivalente, (3.18) peut s'écrire

$$\hat{\phi}(0) = 1, \quad \hat{\phi}^{(l)}(0) = 0 \qquad \text{pour} \quad l = 1, 1, \ldots, L-1$$

$$\hat{\psi}^{(l)}(0) = 0 \qquad \text{pour} \quad l = 0, 1, \ldots, L-1$$

Ce qui, en terme de m_0, devient:

$$m_0^{(l)}(\omega+\pi)=0, \quad \text{pour} \quad l=0,1,\ldots,L-1 \tag{3.19}$$

$$m_0(0)=1, \quad m_0^{(l)}(0)=0, \quad \text{pour} \quad l=1,\ldots,L-1 \tag{3.20}$$

Par (3.19), on a bien un zéro d'ordre L en $\omega=\pi$. Par conséquent, m_0 est de la forme

$$m_0(\omega)=\left(\frac{1+e^{i\omega}}{2}\right)Q(e^{i\omega}), \tag{3.21}$$

où

$$|Q(e^{i\omega})|^2=\sum_{j=0}^{L-1}\binom{L-1+j}{j}\left(\frac{1-\cos\omega}{2}\right)^j+\left(\frac{1-\cos\omega}{2}\right)^L R(\cos\omega), \tag{3.22}$$

et R est un polynôme impair. D'un autre côté, (3.20) implique

$$m_0(\omega)=1+(1-e^{i\omega})^L S(e^{i\omega}), \tag{3.23}$$

En égalisant (3.21) et (3.23), on a L contraintes linéairement indépendantes sur les coefficients de S. De plus, imposer Q de la forme (3.22), avec R polynôme impair, conduit à des contraintes *quadratiques*.

Pour facilité, on choisit de se restreindre au cas $L=2K$ pair. On impose alors m_0 de la forme:

$$m_0(\omega)=\left(\cos^2\frac{\omega}{2}\right)\left[\sum_{k=0}^{K-1}\binom{K-1+k}{k}\left(\sin^2\frac{\omega}{2}\right)^k+\left(\sin^2\frac{\omega}{2}\right)^K f(\omega)\right]. \tag{3.24}$$

ce qui peut aussi s'écrire

$$m_0(\omega)=1+\left(\sin^2\frac{\omega}{2}\right)^K\left[-\sum_{k=0}^{K-1}\binom{K-1+k}{k}\left(\cos^2\frac{\omega}{2}\right)^k+\left(\cos^2\frac{\omega}{2}\right)^K f(\omega)\right]. \tag{3.25}$$

Ces équations satisfont pleinement (3.19) et (3.20). Il ne reste plus qu'à trouver les f_n tel que m_0 satisfasse $\sum h_n=\sqrt{2}$. On pose

$$f(\omega)=\sum_{n=0}^{K'} f_n e^{in\omega}$$

i.e. les f_n forment un filtre causal. de (3.22), on obtient

$$\left[\sum_{k=0}^{K-1}\left(\binom{K-1+k}{k}\right)s_2^k\right]^2+\sum_{k=0}^{K-1}\binom{K-1+k}{k}s_2^{k+K}[f(\omega)+\overline{f(\omega)}] \tag{3.26}$$

$$+s_2^{2K}|f(\omega)|^2=\sum_{j=0}^{2K-1}\binom{2K-1+j}{j}s_2^j+s_2^{2K}R(\cos\omega),$$

où s_2 signifie $\sin^2(\omega/2)$. On détermine les f_n en identifiant les coefficients des s_2^j.

Daubechies construisit ses coiflets en montrant qu'il existe une solution au système si $K'\geq 2K-1$. On choisit donc $K'=2K-1$. Cela conduit à un système de K équations quadratiques à K inconnues; chaque solution correspondant à une coiflet d'ordre $2K$, avec un support de largeur $3K-1$.

Algorithme

Soit $f(\omega) = \sum_{k=0}^{2K-1} f_k e^{ik\omega}$. Le système à résoudre s'écrit

$$f_k = \left(1 - \frac{1}{2}\delta_{k0}\right)(-1)^k \quad \left[2\sum_{n=k}^{K-1} \begin{pmatrix} 2n \\ n-k \end{pmatrix} 4^{-n} \begin{pmatrix} 2K-1+n \\ K+n \end{pmatrix} + \sum_{m=0}^{K-1} \begin{pmatrix} 2m+2K \\ m+K-k \end{pmatrix} 4^{-m-K} x_m \right], \quad 0 \le k \le K-1 \tag{3.27}$$

$$f_k = (-1)^k \sum_{n=k-K}^{K-1} \begin{pmatrix} 2m+2K \\ m+K-k \end{pmatrix} 4^{-m-K} x_m, \quad K \le k \le 2K-1 \tag{3.28}$$

Le système de K équations à K inconnues $x_0, x_1, \ldots, x_{K-1}$ est

$$\sum_{k=0}^{2(K-r)-1} f_k \overline{f_{2r+k}} + \sum_{j=2r}^{2K-2} 4^{-j} \begin{pmatrix} 2j \\ j+2r \end{pmatrix} \sum_{m=\max(0,j-K+1)}^{\min(j,K-1)} \begin{pmatrix} K-1+j-m \\ j-m \end{pmatrix} x_m = 0, \tag{3.29}$$

où $r = 0, 1, \ldots, K-1$. et où (3.28) et (3.28) doivent être substitués en lieu et place des f_k. Ce système quadratique peut avoir une infinité de solutions ou pas du tout. Mais un argument heuristique suggère qu'on a des solutions pour K suffisamment grand. L'équation (3.9) peut alors être réécrite

$$m_0(\omega) = \frac{1}{2} + 2^{-4K+1} K \begin{pmatrix} 2K \\ K \end{pmatrix} \sum_{k=0}^{K-1} \frac{(-1)^k}{2k+1} \begin{pmatrix} 2K-1 \\ K+k \end{pmatrix} \left(e^{i(2k+1)\omega} + e^{-i(2k+1)\omega}\right) + \left(\frac{\sin\frac{\omega}{2}}{2}\right)^{2K} f(\omega). \tag{3.30}$$

Daubechies démontre qu'il est possible de rechercher une solution proche de $x_m = 0, \quad m = 0, 1, \ldots, K-1$.

Nous utilisons donc ce point de départ pour la méthode de Newton détaillée en Annexe B. On note que pour $K \ge 5$ la précision sur les coefficients n'est plus très bonne du à des erreurs d'arrondi.

TAB. 3.2 - : *Algorithme de calcul des coiflets.*

3.3 Ondelettes biorthogonales

La propriété d'orthogonalité, quoique requise pour certaines applications numériques, impose de fortes limitations sur la construction des ondelettes. En effet, on sait qu'il n'existe pas d'ondelettes à support compact, symétrique et orthogonales. Par conséquent on a introduit la généralisation des *ondelettes biorthogonales*. Ici, il existe une fonction d'échelle duale $\tilde{\phi}$ et une fonction ondelette duale $\tilde{\psi}$ qui génèrent une analyse multirésolution avec les sous-espaces $\tilde{V}_j$ et $\tilde{W}_j$ tels que

$$\tilde{V}_j \perp W_j \quad \text{et} \quad V_j \perp \tilde{W}_j, \tag{3.31}$$

et par conséquent

$$\tilde{W}_j \perp W_{j'} \quad \text{pour} \quad j \neq j'.$$

Une analyse multirésolution duale n'est pas nécessairement la même que celle générée par les fonctions primaires (sinon c'est une analyse orthogonale). Equivalent à (3.31), les fonctions duales doivent vérifier

$$\left\langle \tilde{\phi}, \psi(x-l) \right\rangle = \left\langle \tilde{\psi}, \phi(x-l) \right\rangle$$

$$\left\langle \tilde{\phi}, \phi(x-l) \right\rangle = \delta_l \quad \text{et} \quad \left\langle \tilde{\psi}, \phi(x-l) \right\rangle = \delta_l$$

En utilisant un argument d'échelle, avec $\tilde{\phi}_{j,l}$ et $\tilde{\psi}_{j,l}$ définis comme ϕ et ψ ,

$$\left\langle \tilde{\phi}_{j,l}, \phi_{j,l'} \right\rangle = \delta_{l-l'} \quad l,l',j \in \mathbb{Z} \left\langle \tilde{\psi}_{j,l}, \psi_{j',l'} \right\rangle = \delta_{j-j'}\delta_{l-l'} \quad l,l',j,j' \in \mathbb{Z} \tag{3.32}$$

Les **conditions de biorthogonalité** sont alors

$$\forall \omega \in \mathbb{R} \quad \begin{cases} \sum_k \hat{\tilde{\phi}}(\omega + k2\pi)\overline{\hat{\tilde{\phi}}(\omega + k2\pi)} = 1 \\ \sum_k \hat{\tilde{\psi}}(\omega + k2\pi)\overline{\hat{\tilde{\psi}}(\omega + k2\pi)} = 1 \\ \sum_k \hat{\tilde{\psi}}(\omega + k2\pi)\overline{\hat{\tilde{\phi}}(\omega + k2\pi)} = 0 \\ \sum_k \hat{\tilde{\phi}}(\omega + k2\pi)\overline{\hat{\tilde{\psi}}(\omega + k2\pi)} = 0 \end{cases} \tag{3.33}$$

Comme elles définissent une analyse multirésolution les fonctions duales $\tilde{\phi}$ et $\tilde{\psi}$ satisfont

$$\tilde{\phi}(x) = 2\sum_k \tilde{h}_k \tilde{\phi}(2x-k) \quad \text{et} \quad \tilde{\psi}(x) = 2\sum_k \tilde{g}_k \tilde{\phi}(2x-k). \tag{3.34}$$

De même, on définit les fonctions de tranfert $\tilde{H}$ et $\tilde{G}$ similaire à H et G. Les conditions nécessaires et suffisantes sont alors

$$\forall \omega \in \mathbb{R} \quad \begin{cases} \tilde{H}(\omega)\overline{H(\omega)} + \tilde{H}(\omega+\pi)\overline{H(\omega+\pi)} = 1 \\ \tilde{G}(\omega)\overline{G(\omega)} + \tilde{G}(\omega+\pi)\overline{G(\omega+\pi)} = 1 \\ \tilde{G}(\omega)\overline{H(\omega)} + \tilde{G}(\omega+\pi)\overline{H(\omega+\pi)} = 0 \\ \tilde{H}(\omega)\overline{G(\omega)} + \tilde{H}(\omega+\pi)\overline{G(\omega+\pi)} = 0 \end{cases} \tag{3.35}$$

ou

$$\tilde{M}(\omega)\overline{M^t(\omega)} = 1, \tag{3.36}$$

avec M matrice du système

$$M(\omega) = \begin{bmatrix} H(\omega) & H(\omega+\pi) \\ G(\omega) & G(\omega+\pi) \end{bmatrix},$$

La résolution du système donne

$$\tilde{H}(\omega) = \frac{\overline{G(\omega+\pi)}}{\overline{\Delta(\omega)}} \tag{3.37}$$

$$\tilde{G}(\omega) = -\frac{\overline{H(\omega+\pi)}}{\overline{\Delta(\omega)}} \tag{3.38}$$

où

$$\Delta(\omega) = \det M(\omega),$$

Les opérateurs de projection prennent alors la forme

$$P_j f = \sum_l \langle f, \tilde{\phi}_{j,l}\rangle \phi_{j,l}(x) \quad \text{et} \quad Q_j f = \sum_l \langle f, \tilde{\psi}_{j,l}\rangle \psi_{j,l}(x) \tag{3.39}$$

De l'équation (3.39), on note que tous les coefficients ondelettes sont calculés par $\tilde{\phi}$ pour $P_j f$ et $\tilde{\psi}$ pour $Q_j f$. Par conséquent, avec un $\tilde{\phi}$ simple, on peut calculer rapidement ces coefficients. $\tilde{\phi}$ peut ne pas être de grande régularité (par exemple fonction de Haar). Ce manque de régularité sera compenser par ϕ et ϕ pourra être une fonction spline comme on le verra en section suivante. Il s'agit là d'un très grand avantage procuré par ce type de filtres.

D'après (3.32), (3.32), (3.34), on voit que

$$\tilde{h}_{k-2l} = \langle \tilde{\phi}(x-l), \phi(2x-k)\rangle \quad \text{et} \quad \tilde{g}_{k-2l} = \langle \tilde{\psi}(x-l), \phi(2x-k)\rangle. \tag{3.40}$$

De plus, en remarquant que $\phi(2x-k) \in V_{-1}$ on peut la décomposer sur V_0 et W_0

$$\phi(2x-k) = \sum_l \tilde{h}_{k-2l}\phi(x-l) + \sum_l \tilde{g}_{k-2l}\psi(x-l).$$

On a la même chose avec $\tilde{\phi}$ car les fonctions primaires et duales sont interchangeables.

Notons toutefois que le fait que la fonction d'échelle et la fonction ondelette ne soient pas orthogonales ne signifie pas nécesairement que l'analyse multirésolution ne soit pas orthogonale. Si les fonctions d'échelles et ondelettes biorthogonales engendrent une analyse multirésolution orthogonale, on les appelle *semi-orthogonales* [13].

Les contraintes (3.35), en termes de filtres, s'écrivent

$$\sum_n \left[\tilde{h}_{2n-l} h_{2n-k} + \tilde{g}_{2n-l+1} g_{2n-k+1}\right] = \delta_{kl} \tag{3.41}$$

on a alors

$$g_n = (-1)^{n+1}\tilde{h}_{-n}, \quad \tilde{g}_n = (-1)^{n+1} h_{-n}. \tag{3.42}$$

et en utilisant la notation en z

$$h(z)\tilde{h}(z) + h(-z)\tilde{h}(-z) = 2 \tag{3.43}$$

ou

$$\sum_n h_m \tilde{h}_{n+2k} = \delta_{k,0} \tag{3.44}$$

3.3.1 Ondelettes biorthogonales splines

Daubechies, Cohen et Feauveau [17] ont développés des ondelettes biorthogonales à partir de fonctions B-splines cardinales en mimiquant la construction de ϕ , ψ du cas orthogonal [17]. Dans leur cas, on a $\Delta(\omega) = e^{-i\omega}$ et par conséquent

$$G(\omega) = -e^{-i\omega}\overline{\hat{H}(\omega+\pi)} \quad \text{et} \quad \tilde{G}(\omega) = -e^{-i\omega}\overline{H(\omega+\pi)}$$

On définit

$$\begin{aligned} m_0(\omega) &= 2^{-1/2}\sum_n h_n e^{-in\omega} \\ \tilde{m}_0(\omega) &= 2^{-1/2}\sum_n \tilde{h}_n e^{-in\omega} \end{aligned} \tag{3.45}$$

Dans le cas biorthogonal, il est possible de choisir m_0 tel qu'il corresponde à un filtre à phase linéaire, i.e. à une fonction d'échelle ϕ symétrique. Le filtre associé à m_0 à phase linéaire est

$$m_0(\omega) = e^{i\lambda\omega}|m_0(\omega)|, \lambda \in \mathbb{Z} \tag{3.46}$$

Pour h_n réel,

$$m_0(-\omega) = m_0(\omega) \tag{3.47}$$

ou, de manière équivalente,

$$m_0(\omega) = \quad \text{polynôme en} \quad \cos\omega \tag{3.48}$$

On a donc $\phi(x) = \phi(-x)$ vu que ϕ est réel. Or, cette définition n'inclut pas le cas des filtres de Haar dont la fonction d'échelle ϕ est symétrique autour de $x = 1/2$. Cela correspond à

$$m_0(-\omega) = e^{i\omega}m_0(\omega) \tag{3.49}$$

ou, de manière équivalente,

$$m_0(\omega) = e^{-i\omega/2}\cos(\omega/2). \quad \text{polynôme en} \quad \cos\omega \tag{3.50}$$

Nous désirons donc déterminer m_0 et $\tilde{m}_0$ qui vérifient

$$m_0(\omega)\overline{\tilde{m}_0(\omega)} + m_0(\omega+\pi)\overline{\tilde{m}_0(\omega+\pi)} = 1. \tag{3.51}$$

On arrive donc à la proposition 6.4 [17]:

Proposition 6.4 [17] *Soit m_0 un polynôme trigonométrique à coefficients réels satisfaisant ou (3.47) ou (3.49), i.e. m_0 peut être écrit*

$$m_0(\omega) = e^{i\chi\omega/2}(\cos\omega/2)^N p_0(\cos\omega)$$

avec $p_0(-1) \neq 0, N \in \mathbb{N}$, $\chi = 1$ si N impaire, 0 sinon. S'il existe des solutions $\tilde{m}_0$ à (3.51), alors elles sont de la même forme que m_0 , i.e.

$$\tilde{m}_0(\omega) = e^{i\chi\omega/2}(\cos\omega/2)^{\tilde{N}}\tilde{p}_0(\cos\omega)$$

avec $\tilde{p}_0(-1) \neq 0, \tilde{N} \in \mathbb{N}$, $\chi = 1$ si N impaire, 0 sinon. De plus, m_0 et $\tilde{m}_0$ sont contraints par

$$p_0(\cos\omega)\tilde{p}_0(\cos\omega) = \sum_{n=0}^{k-1}\begin{pmatrix} k-1+n \\ n \end{pmatrix}\left(\sin^2\omega/2\right)^n + \left(\sin^2\omega/2\right)^k R(\cos\omega) \tag{3.52}$$

où $N + \tilde{N} = 2k$ et R est un polynôme impaire.

Si on prend $p_0 = 1$, $R \equiv 0$, il vient

$$\begin{aligned} m_0(\omega) &= \left(\frac{1+e^{-i\omega}}{2}\right)^N \\ &= \sum_{n=-\lfloor N/2 \rfloor}^{N-\lfloor N/2 \rfloor} 2^{-N} \begin{pmatrix} N \\ n+\lfloor N/2 \rfloor \end{pmatrix} e^{-in\omega} \end{aligned}$$

En posant $z = e^{-i\omega}$,

$$m_0(\omega) = \sum_{n=-\lfloor N/2 \rfloor}^{N-\lfloor N/2 \rfloor} 2^{-N} \begin{pmatrix} N \\ n+\lfloor N/2 \rfloor \end{pmatrix} z^n \tag{3.53}$$

L'expression de m_0 fait clairement apparaître les coefficients des z^n comme ceux d'une B-spline (voir figures (4.3), (4.4), (4.5)).

Le $\tilde{m}_0$ vérifiant (3.51), qui a la même symétrie que m_0 est alors donné par

$$\tilde{m}_0(\omega) = e^{-i\chi\omega/2}(\cos\omega/2)^{\tilde{N}} \left[\sum_{n=0}^{k-1} \begin{pmatrix} k-1+n \\ n \end{pmatrix} (\sin\omega/2)^{2n}\right]. \tag{3.54}$$

Comme dans le cas orthogonal, avec (3.15),

$$m_0(z) = \left(\frac{1+z}{2}\right)^N z^{-\lfloor N/2 \rfloor} \tag{3.55}$$

$$\tilde{m}_0(z) = z^{-(\tilde{N}+\chi)/2} \left(\frac{1+z^{-1}}{2}\right)^{\tilde{N}} \sum_{n=0}^{k-1} \begin{pmatrix} k+n-1 \\ n \end{pmatrix} \left(\frac{z^{-1}-2+z}{4}\right)^n \tag{3.56}$$

Par conséquent, on en déduit les supports (compacts) de ϕ et $\tilde{\phi}$

$$\begin{aligned} \mathrm{supp}\phi &= \mathrm{supp} m_0(z) = \left[-\frac{N-\chi}{2}, \frac{N+\chi}{2}\right] \\ \mathrm{supp}\tilde{\phi} &= \mathrm{supp}\tilde{m}_0(z) = \left[-(k-1) - \frac{\tilde{N}-\chi}{2}, k-1+\frac{\tilde{N}+\chi}{2}\right] \end{aligned} \tag{3.57}$$

L'algorithme de génération des filtres h et $\tilde{h}$ est donc:

3.3.2 Pseudo-coiflets

On peut se demander s'il ne serait pas plus facile de calculer des filtres biorthogonaux ayant les propriétés des coiflets; à savoir que ϕ à des moments nuls sauf le premier qui vaut 1. Cette idée à permis à Reissell [52] de synthétiser ces filtres.
Avec les notations précédentes, soit $m_0(\omega) = \frac{1}{\sqrt{2}} \sum h_n e^{-in\omega}$, il doit vérifier (voir section précédente):

$$m_0(\omega) = 1 + \left(1-e^{-i\omega}\right)^{2N_2} L_2(\omega) = 1 + (1-\cos\omega)^{N_2} P_2(\omega) \tag{3.58}$$

Algorithme

1. calcul des coeffcients des z dans (3.56) de la somme
$$\sum_{n=0}^{k-1}(-1)^n \begin{pmatrix} k+n-1 \\ n \end{pmatrix} \left(\frac{1-z}{2}\right)^{2n} z^{-n},$$
2. calcul des coefficients des z de $\left(\frac{1+z}{2}\right)^{\tilde{N}}$,
3. on multiplie ces deux polynomes: cela revient à faire une convolution des deux tableaux de coefficients,
4. $$h_n = \sqrt{2}m_0(n), \quad -\frac{N-\chi}{2} \leq n \leq \frac{N+\chi}{2}$$
$$\tilde{h}_n = \sqrt{2}\tilde{m}_0(n), \quad -(k-1)-\frac{N-\chi}{2} \leq n \leq k-1+\frac{N+\chi}{2}$$

TAB. 3.3 - : *Algorithme de calcul des filtres splines biorthogonaux.*

$$m_0(\omega) = \left(1+e^{i\omega}\right)^{2N_1} L_1(\omega) = (1+\cos\omega)^{N_1} P_1(\omega) \tag{3.59}$$

Ici, P_1 et P_2 sont des polynômes en $\cos\omega$. La parité des polynômes garantit la symétrie de la fonction ϕ résultante.
Egaliser ces deux équations revient à résoudre

$$(1+x)^{N_1}P_1(x) - (1-x)^{N_2}P_2(x) = 1$$

avec $x = \cos\omega$. On choisit $N = N_1 = N_2$, il vient

$$(1+x)^{N}P_1(x) - (1-x)^{N}P_2(x) = 1 \tag{3.60}$$

Ceci n'est autre qu'une équation de Bezout [24, 75] (appelée "équation des coiflets") dont la solution est

$$P_1(x) = 2^{-N}\sum_{k=0}^{N-1} \begin{pmatrix} N-1+k \\ k \end{pmatrix} 2^{-k}(1-x)^k + (1-x)^N F(x) \tag{3.61}$$

où F est un polynôme impaire arbitraire.

Dans le cas des pseudo-coiflets, supposons qu'on demande une fonction d'échelle symétrique au banc de synthèse, $\tilde{\phi}$, et que l'on veuille trouver la fonction d'échelle correspondante ϕ au banc d'analyse, on va modifier légèrement la procédure de [17] (section 3.3). Dans cet algorithme, on utilise deux polynômes trigonométriques P et $\tilde{P}$, pour ϕ et $\tilde{\phi}$ respectivement. L'équation est alors résolue pour $P\tilde{P}$ convenablement factorisé en P et $\tilde{P}$. Ici, on connait $\tilde{P}$ donné par (3.61). Reissell montre que connaissant $\tilde{P}$, on peut toujours trouver P vérifiant l'équation des coiflets (3.60).

On définit les fonctions de transfert des filtres d'analyse et de synthèse, resp. m_0 et $\tilde{m}_0$ satisfaisant les $2N$ ou $2\tilde{N}$ premières conditions sur les moments (3.58) et

(3.59) par

$$m_0(\omega) = (1+\cos\omega)^N P(\omega), \quad \tilde{m}_0(\omega) = (1+\cos\omega)^{\tilde{N}} \tilde{P}(\omega) \tag{3.62}$$

En posant $x = \cos\omega$, il vient

$$(1+x)^{N+\tilde{N}} P(x)\tilde{P}(x) - (1-x)^{N+\tilde{N}} P(-x)\tilde{P}(-x) = 1 \tag{3.63}$$

Si $\tilde{P}(x)$ et $\tilde{P}(-x)$ n'ont pas de zéros en commun, l'équation (3.63) admet une solution unique de degré minimum $(P(x), P(-x))$ avec $\deg P = N + \tilde{N} + \deg \tilde{P} - 1$; cela vient du théorème de Bezout ([22], chapitre 6). On montre alors que si $\tilde{P}(x)$ est la solution de degré minimum (3.61) de (3.60) alors il existe P de degré minimum tel que P et $\tilde{P}$ satisfont l'équation de biorthogonalité (3.63). On a alors

$$\deg P = 3N - 2$$

$$\deg \tilde{P} = N - 1$$

L'algorithme de génération des pseudo-coiflets est donc:

Algorithme

Soit P et $\tilde{P}$ deux polynômes trigonométriques,

$$m_0(\omega) = (1+\cos\omega)^N P(\cos\omega), \tag{3.64}$$

$$\tilde{m}_0(\omega) = (1+\cos\omega)^{\tilde{N}} \tilde{P}(\cos\omega). \tag{3.65}$$

1. Calculer (3.61)

$$\tilde{P}(x) = P_1(x) = 2^{-N} \sum_{k=0}^{N-1} \begin{pmatrix} N-1+k \\ k \end{pmatrix} 2^{-k}(1-x)^k,$$

2. Trouver P de degré minimal pour ce $\tilde{P}$ en utilisant le système linéaire (3.63),

$$(1+x)^{N+\tilde{N}} P(x)\tilde{P}(x) - (1-x)^{N+\tilde{N}} P(-x)\tilde{P}(-x) = 1, \tag{3.66}$$

3. Evaluer les filtres d'après (3.64) et (3.65):

$$h_k = \sqrt{2} m_0(k)$$

$$\tilde{h}_k = \sqrt{2} \tilde{m}_0(k)$$

Nous proposons en annexe C une méthode permettant de résoudre ce système.

TAB. 3.4 - : *Algorithme de génération des pseudo-coiflets.*

Les propriétés des fonctions ondelettes obtenues sont les suivantes:

– Les pseudo-coiflets (ϕ, ψ) et $(\tilde{\phi}, \tilde{\psi})$ sont symétriques.

- Les pseudo-coiflets ψ et $\tilde{\psi}$ ont $2N$ moments nuls, tout comme $\tilde{\phi}$.
- La fonction d'échelle de synthèse $\tilde{\phi}$ est interpolante (spline).
- $\deg m_0 = 3N - 2$, $\deg \tilde{m}_0 = N - 1$.
- La longueur du filtre d'analyse h est $8N - 3$ (degrés de $-(4N - 2)$ à $4N - 2$), celle du filtre de synthèse $\tilde{h}$ est $4N - 1$ (degrés de $-(2N - 1)$ à $2N - 1$)

3.4 Récapitulatif des propriétés de quelques filtres

Au début du chapitre, nous avons donné les propriétés recherchées pour une ondelette. Nous avons détaillé la construction de certaines ondelettes implémentées dans notre librairie Wavelib 1.0 . En fait, chacun peut construire sa propre ondelette en respectant les propriétés de base suivant qu'on la souhaite orthogonale ou non. En voici d'autres

- **Ondelettes orthogonales**
  - **Ondelette orthogonales simples.** On peut prendre comme fonction d'échelle $\chi_{[0,1]}(x)$ ou la fonction d'échantillonnage de Shannon $\mathrm{sinc}(\pi x)$. L'ondelette correspondante est

$$\psi_{Haar}(x) = \chi_{[0,1/2]}(x) - \chi_{1/2,1]}(x),$$

    et

$$\psi_{Shannon}(x) = \frac{\sin(2\pi x) - sin(\pi x)}{\pi x}.$$

    En pratique, ces ondelettes ne sont pas très intéressantes, la première a une régularité très faible et l'autre a une décroissance trop faible,
  - **Ondelette de Meyer [48].** Ces fonctions sont C^∞, de support infini et ont décroissance supérieure à une décroissance polynomiale,
  - **Ondelettes de Battle-Lemarié [6, 41].** Elles sont construites en orthogonalisant des fonctions B-splines et elles ont une décroissance exponentielle [6, 41]. L'ondelette a N moments nuls est une fonction polynômiale par morceaux de degré $N - 1$ qui appartient à $\mathcal{C}^{N-2}$.
  - **Ondelettes de Daubechies [22, 24].** Soit N le nombre de moments nuls de l'ondelette, elle est alors de support $2N - 1$. Leur désavantage (sauf pour l'ondelette de Haar ($N = 1$)) est qu'elles ne peuvent être ni symétriques, ni anti-symétriques. Leur régularité augmente avec N et est approximativement de $0,3 \times N$. Trois variations de cette famille ont été construites [23]:
    1. Les ondelettes classiques de Daubechies détaillées auparavant pour N fixé et de support $2N-1$ ne sont pas uniques. Il existe une famille où pour N fixé on retient la solution la plus proche de la phase linéaire (c'est en fait le choix de la phase extrémale),
    2. Une autre famille a plus de régularité mais au prix d'un support plus grand,

3. Les coiflets [23], voir section (3.2.2), ont un support de $3N-1$ et la fonction d'échelle possède $N-1$ moments nuls. Très utiles en analyse numérique.

- **Ondelettes biorthognales.**
  - **Ondelettes de Cohen, Daubechies, et Feauveau [17].** Détaillées auparavant en section (3.3), les fonctions d'échelles sont des B-splines cardinales ainsi que les ondelettes. Toutes ces fonctions ont un support compact et sont à phase linéaire. De plus, tous les coefficients des filtres sont rationnels dyadiques. Un désavantage est que pour des filtres de petites tailles, les fonctions duales ont une régularité très faible,
  - **Ondelettes de Chui [13, 14, 15].** Les fonctions d'échelles sont des splines d'ordre m et les ondelettes sont des splines de support compact $[0, 2m-1]$. Les fonctions sont encore à phase linéaire et les coefficients sont rationnels. Un point très important est qu'il existe une expression analytique pour les fonctions primaires et duales. Un désavantage est que les fonctions duales ne sont pas à support compact mais elles ont une décroissance exponentielle. Les mêmes ondelettes ont été établies différemment par Unser et Al. [72],
  - il existe d'autres types d'ondelettes semi-orthogonales (voir articles de C.A. Micchelli).

Le tableau suivant dresse un rapide aperçu des caractéritiques de quelques ondelettes [39]

Famille ondelette	Support compact		Expression analytique		Symétrie	Orthogonalité		Support compact $\tilde{\psi}$
	Primaire	Duale	Primaire	Duale		Semi	Totale	
α	O	O	N	N	N	O	O	N
β	O	O	O	N	O	N	N	N
γ	O	N	O	O	O	O	N	N
δ	N	N	N	N	O	O	O	O
ϵ	N	N	O	O	O	O	O	N

α : Ondelettes de Daubechies
β : Ondelettes biorthogonales spline
γ : Ondelettes spline semi-orthogonales
δ : Ondelettes de Meyer
ϵ : Ondelettes spline orthogonales

Chapitre 4

Mise en oeuvre et résultats

L'implémentation de fonctions permettant de générer ou d'utiliser les ondelettes a d'abord été faite sous le logiciel de simulation mathématique Matlab, puis en langage C. Matlab est un logiciel comprenant de nombreuses fonctions mathématiques très simples à utiliser évitant ainsi une perte de temps pour le développement de bibliothèques de fonctions annexes pour des opérations de base (matricielles, polynomiales). De plus, il dispose de puissantes fonctions graphiques que nous utilisons abusivement. Malheureusement, il s'agit d'un langage interprété et par conséquent lent pour des opérations se répétant souvent. Nous avons donc été amené à redévelopper les fonctions en langage C pour une rapidité accrue. A titre d'exemple, une FWT 2D complète d'une image 512×512 sous Matlab prend environ 3mn. En C il suffit d'environ une minute (temps système sur une station SUN SPARC 10).

4.1 Sous Matlab

Matlab est un langage où l'objet de base est la matrice. Malgré sa souplesse, Matlab n'est ni dynamique ni récursif et il faut faire très attention au redimensionnement des tableaux. De plus, Matlab ne permet pas l'utilisation de stuctures comme en C. On a alors beaucoup de paramètres à récupérer ou à passer aux fonctions. Un autre désavantage est qu'on ne peut avoir qu'une fonction par fichier. Ainsi, si on a besoin de 10 fonctions, il faudra utiliser 10 fichiers : on arrive donc très vite à un grand nombre de fichiers.

A partir de la Toolbox de J.C. Kantor, qui n'était faite que pour les ondelettes de Daubechies et les transformées orthogonales, nous avons généralisé les fonctions afin de pouvoir générer tous types d'ondelettes et utiliser des transformées orthogonales ou non. Toutefois, nous n'avons pas été jusqu'à implémenter les paquets d'ondelettes sous Matlab. En effet, les transformées par paquets d'ondelettes sont typiquement récursives (section 2.3.6) et, recréer un tel mécanisme sous Matlab implique l'utilisation de piles pour stocker temporairement les bases réellement utilisées parmi toute la librairie de bases de fonctions. Nous avons donc préféré ne pas continuer trop loin le développement sous Matlab pour nous concentrer sur la bibliothèque en C.

Néammoins, Matlab permet de générer tous types de graphiques 1D, 2D, 3D. Nous avons donc développé une interface entre Matlab et la bibliothèque C pour

afficher l'arbre de décomposition totale par paquets d'ondelette, l'arbre de coût et le graphique temps-fréquence correspondant. Nous allons détailler les fonctions utilisées.

- **[deb,fin,fc]=fload('nom_fichier')** pour lire un filtre. Retourne les indices de début *deb* et de fin fin et les coefficients fc du filtre,
- **mat=echload(fp)** charge un vecteur ou une matrice du fichier repéré par son "file pointer" fp. Pour charger une matrice, on fait
  fp=**fopen**('nom_fichier');
  mat=**echload**(fp);
  fclose(fp);
- **wpplot** charge les coefficients *paquets d'ondelettes*, affiche l'arbre des bases, l'arbre de coût et le diagramme temps-fréquence résultant de la transformée ondelette choisie. Note: le fichier contenant toutes les données est par défaut *pcb.dat*,
- **echsave('nom_fichier',mat)** sauve la matrice *mat* dans le fichier $nom\_fichier$. Ce fichier pourra ensuite être relu par Wavelib 1.0 ,
- **wloadtst** permet d'afficher les fonctions ϕ , ψ et $\tilde{\phi}$, $\tilde{\psi}$ (dans le cas orthogonal on aura donc deux fois la même chose), générées par l'algorithme cascade de Wavelib 1.0 ,
- **snum=plotwc(x,tol,color)** affiche les valeurs absolues des coefficients ondelettes x triés par ordre décroissant avec la couleur *color* (défaut rouge = 'r'). Renvoi *snum* le nombre de coefficients qui sont en-dessous de $tol * \max_i\{x_i\}$. Cette fonction permet d'évaluer visuellement la décroissance des coefficients ondelettes et le taux de compression obtenu si on enlève les échantillons sous ce seuil. On prend souvent une tolérance entre 0,1 % et 1%,
- **[order,cost,wp]=wpload('nom_fichier')** charge toutes les données résultant de la transformée par paquets d'ondelettes du fichier $nom\_fichier$: basis tree *order*, cost tree *cost* et les coefficients wp,
- **[deb,fin,dx,j,k,x,phi]=waveload('nom_fichier')** charge tous les paramètres de la structure d'une ondelette *phi* créée par l'algorithme cascade du fichier $nom\_fichier$. Retourne le support de la fonction $[deb, fin]$, l'écart entre échantillons $1/dx$, le niveau j où a été générée cette ondelette et le paramètre de translation k. Retourne également pour l'affichage, un vecteur x et les valeurs *phi* de ϕ aux points dyadiques: pour afficher l'ondelette, il ne reste plus qu'à faire **plot**(x,phi),
- **[coeff,idx]=sortcoeff('nom_fichier',tol,color)** tout comme plotwc mais va chercher dans le fichier $nom\_fichier$ les valeurs des coefficients, les trie par ordre décroissant et retourne les coefficients triés $coeff$ avec l'ordre des permutations idx. Affiche également le nombre de coefficients retirés par rapport au nombre total.

Les autres fonctions relatives aux premiers développements sont également disponibles. Le lecteur est invité à se référer au manuel utilisateur de Wavelib 1.0 [11].

4.2 En langage C

Le développement de Wavelib 1.0 a tout d'abord consisté à translater les fonctions écrites sous Matlab en C. Puis, l'utilisation de la richesse du C (pointeurs, allocations dynamiques, structures, récursivité) ont permis le développement d'une bibliothèque avec des objets génériques simples à utiliser et où la généralisation des fonctions devrait pouvoir se faire assez facilement.

4.2.1 Des premiers développements à Wavelib 1.0

Au tout début, le principal problème était de développer toutes les fonctions matricielles et polynômiales de base nécessaires pour la création des filtres. Puis, nous avons eu besoin d'algorithmes rapides pour résoudre des systèmes d'équations. Il était hors de question de redévelopper des fonctions qui ont été mises au point depuis bien longtemps. Aussi, en se penchant sur le code Fortran de la librairie Lapack, nous avons trouvé la librairie Meschach qui est écrite en C. Cette librairie nous a servi de base pour toutes les routines bas-niveaux (complexes, matrices, vecteur, normalisation, solveur d'équations par méthode LU ou par factorisation QR, méthode de Schur pour l'extraction de valeurs et vecteurs propres (ou racines d'un polynôme)). En fait, il nous a fallu adapter cette librairie pour que le code s'insère dans Wavelib 1.0 . Même si les types de base sont les mêmes, la librairie Meschach possède des mécanismes complexes pour gérer la mémoire qui ne sont pas utiles pour notre librairie.

Une fois le code modifié, nous avons testé et validé les diverses fonctions incluses. On y a également inclus de nouvelles fonctions pour généraliser l'utilisation du solveur par exemple afin de n'avoir qu'une fonction à appeler au lieu de plusieurs.

Wavelib 1.0 comporte actuellement un peu près 250 pages de code. L'objectif qu'on s'est fixé est de procurer à l'utilisateur des fonctions robustes, simples d'appel, sans qu'il ait à se poser la question : qu'elles sont les fonctions à utiliser pour faire une opération simple? Par cette simplification, on cache à l'utilisateur les mécanismes sous-jacents. Mais, quelle intérêt y trouverait-il à utiliser une librairie où il doit passer un certain temps pour bien comprendre toutes les fonctions? Il abandonnerait vite !

Finalement, on a regroupé les types simples en structures pour mieux organiser le code. Certaines structures sont d'ailleurs très générales et servent par simple *casting* aussi bien aux transformées en 1D ou en 2D.

On trouvera dans Wavelib 1.0 des fonctions spécialisées pour la FWT 1D, la FWT 2D et les transformées par paquets d'ondelettes 1D et 2D avec la recherche de la meilleure base (algorithme 2.6). Ces fonctions datent des premiers développements où on cherchait à implémenter en C quelques méthodes développées sous Matlab pour des applications précises.

Actuellement, les fonctions de décomposition par paquets d'ondelettes généralisent les fonctions précédentes. Toutefois, nous les avons conservées car elles sont plus rapides et occupent moins de mémoire.

4.2.2 Organisation de Wavelib 1.0

Wavelib 1.0 est organisé autour de 30 fichiers C et 10 "headers". Le fichier central est *types.h*, il contient tous les types de Wavelib 1.0 . La figure suivante dresse les liens entre les fichiers C et les fichiers headers.

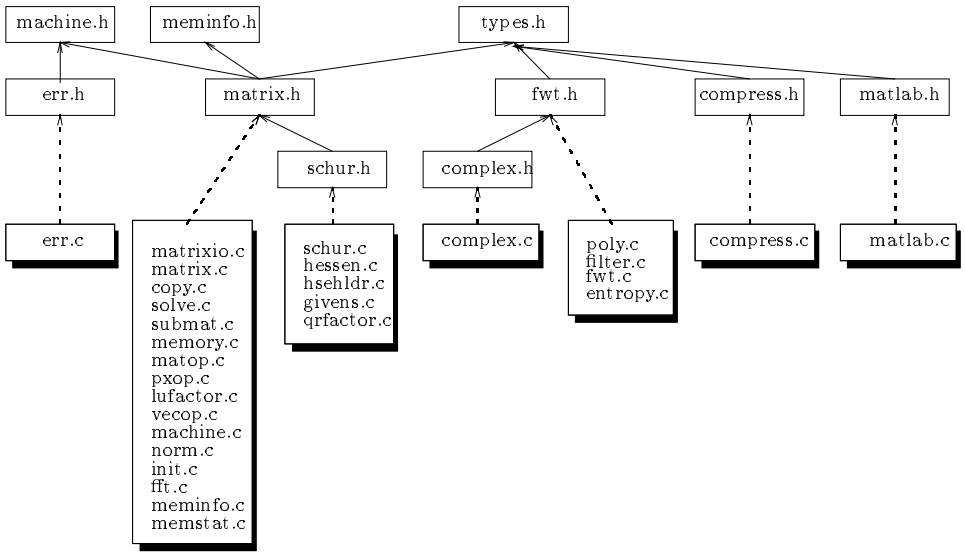


FIG. 4.1 - : *Organisation de Wavelib 1.0 .*

Afin de mieux s'y retrouver, voici à quoi servent les différents headers :

- **err.h, meminfo.h, machine.h :** ne servent qu'à la gestion mémoire (nombre d'allocations de chaque type réalisé, vérification que les zones mémoire allouées ont bien été désallouées). Ce sont des fichiers provenant de la librairie Meschach et qui seront probablement non conservés dans les versions ultérieures de WaveLib. Ils servent comme *"Memory handler"*,
- **compress.h :** définitions des fonctions de compression de *compress.c*,
- **matlab.h :** définitions des fonctions d'interface avec Matlab. Contient également du code pour écrire des matrices sous Matlab 3.x,
- **complex.h :** définitions de la librairie pour nombres complexes,
- **fwt.h :** définitions des fonctions de génération de filtres, de filtrage, de tranformées ondelettes, polynomiales et les fonctions de coût du fichier *entropy.c*,
- **schur.h :** définitions des fonctions pour les factorisations QR, les formes de Schur et d'Hessenberg,
- **matrix.h :** définitions des fonctions les plus générales de (dé)allocation de tous les types de la librairie, les opérations matricielles et vectorielles, les fonctions de calcul de normes, les fonctions du solveur LU,

– **et types.h :** définitions de tous les types de Wavelib 1.0 ainsi que des constantes et tableaux externes.

En utilisant des méthodes de bases de la librairie Meschach, de nombreux autres types ont été définis de façon à ne pas alourdir les passages de paramètres de fonctions. Wavelib 1.0 se veut avant tout simple d'emploi et accessible rapidement à tous ceux qui souhaitent utiliser les ondelettes.

4.3 Exemples d'applications

Afin de balayer les possibilités actuelles de Wavelib 1.0 , nous allons montrer comment utiliser la librairie à travers quelques exemples. Pour l'utilisation des fonctions de Wavelib 1.0 , le lecteur est invité à se rapporter au **manuel de l'utilisateur** du logiciel.

4.3.1 Filtres

Avant tout, on doit choisir les filtres nécessaires aux transformées. Pour toutes les tables données, le lecteur peut se rapporter à l'ouvrage de Daubechies [22]. Les filtres sont donnés avec leurs bornes

$$H(z) = \sum_{n=deb}^{fin} h_n z^{-n}$$

et tous les filtres vérifient

$$\sum_{n=deb}^{fin} h_n = \sqrt{2}$$

– **Filtres de Daubechies.** On note D_N le filtre de Daubechies dont l'ondelette possède N moments nuls.

Source en C

```
FILTER *h, *g;

h=daubechies(N);
g=bimod(h);

KillFilter(h);KillFilter(g);
```

TAB. 4.1 - : *Code pour filtres D_N.*

La fonction **bimod** sert à obtenir le filtre g correspondant au filtre h. Cette fonction marche aussi bien pour les filtres orthogonaux que biorthogonaux.

– **Filtres biorthogonaux splines.** On note ces filtres $B_{N,\tilde{N}}$ avec N nombre de moments nuls de ψ et $\tilde{N}$ nombre de moments nuls de $\tilde{\psi}$.

– **Pseudo-coiflets.** On note ces filtres $P_{N,\tilde{N}}$ avec N nombre de moments nuls de ψ et $\tilde{N}$ nombre de moments nuls de $\tilde{\psi}$.

Type	Coefficients	Type	Coefficients
N=2	0.48296291	N=7	0.077852054
deb=0	0.8365163	deb=0	0.39653932
fin=3	0.22414387	fin=13	0.72913209
	0.22414387		0.46978229
	-0.12940952		-0.143906
			-0.22403618
N=3	0.33267055		0.071309219
	0.80689151		0.080612609
	0.4598775		-0.038029937
	-0.13501102		-0.016574542
	-0.085441274		0.012550999
	0.035226292		0.00042957797
			-0.0018016407
N=4	0.23037781		0.0003537138
deb=0	0.71484657		
fin=7	0.63088077	N=8	0.054415842
	-0.027983769	deb=0	0.31287159
	-0.18703481	fin=15	0.67563074
	0.030841382		0.58535468
	0.032883012		-0.015829105
	-0.010597402		-0.28401554
			0.00047248457
N=5	0.1601024		0.12874743
deb=0	0.60382927		-0.017369301
fin=9	0.72430853		-0.044088254
	0.13842815		0.013981028
	-0.24229489		0.008746094
	-0.03224487		-0.004870353
	0.077571494		-0.00039174037
	-0.0062414902		0.00067544941
	-0.012580752		-0.00011747678
	0.0033357253		
N=6	0.11154074		
deb=0	0.49462389		
fin=11	0.75113391		
	0.31525035		
	-0.22626469		
	-0.12976687		
	0.097501606		
	0.027522866		
	-0.031582039		
	0.0005538422		
	0.0047772575		
	-0.0010773011		

TAB. 4.2 - : **Filtres de Daubechies.**

Source en C

```
FILTER *ha, *hs, *ga, *gs;

bispline(&ha,&hs,N,Ñ);
ga=bimod(hs);
gs=bimod(ha);

KillFilter(ha);KillFilter(ga);
KillFilter(hs);KillFilter(gs);
```

TAB. 4.3 - : *Code pour filtres* $B_{N,\tilde{N}}$.

Type	Coefficients	Type	Coefficients	Type	Coefficients	Type	Coefficients
N=1	0.70710678	$\tilde{N}$=3	-0.088388348	N=3	0.1767767	$\tilde{N}$ = 5	-0.013810679
deb=0	0.70710678	deb=-2	0.088388348	deb=-1	0.53033009	deb=-5	0.041432038
fin=1		fin=3	0.70710678	fin=2	0.53033009	fin=6	0.052480581
			0.70710678		0.1767767		-0.26792718
			0.088388348				-0.071815532
			-0.088388348				0.96674755
		$\tilde{N}$=5	0.016572815				0.96674755
		deb=-4	-0.016572815				-0.071815532
		fin=5	-0.12153398				-0.26792718
			0.12153398				0.052480581
			0.70710678				0.041432038
			0.70710678				-0.013810679
			0.12153398				
			-0.12153398				
			-0.016572815				
			0.016572815				

TAB. 4.4 - : **Ondelettes biorthogonales.** *A N fixé, on fait varier* $\tilde{N}$.

Source en C

```
FILTER *ha, *hs, *ga, *gs;

pseudo_coiflet(&ha,&hs,N,Ñ);
ga=bimod(hs);
gs=bimod(ha);

KillFilter(ha);KillFilter(ga);
KillFilter(hs);KillFilter(gs);
```

TAB. 4.5 - : *Code pour filtres* $P_{N,\tilde{N}}$.

Type	Coefficients	Type	Coefficients	Type	Coefficients	Type	Coefficients
N=1	-0.1767767	$\tilde{N}=1$	0.35355339	N=2	-0.0027621359	$\tilde{N}=2$	-0.044194174
deb=-2	0.35355339	deb=-1	0.70710678	deb=-6	0	deb=-3	0
fin=2	1.0606602	fin=1	0.35355339	fin=6	0.049718446	fin=3	0.39774756
	0.35355339				-0.044194174		0.70710678
	-0.1767767				-0.17401456		0.39774756
					0.39774756		0
					0.96122328		-0.044194174
					0.39774756		
					-0.17401456		
					-0.044194174		
					0.049718446		
					0		
					-0.0027621359		

TAB. 4.6 - : **Pseudo-coiflets.** *A N fixé, on fait varier $\tilde{N}$.*

– **Coiflets.** On note ces filtres C_N avec N nombre de momoents nuls de ψ . Actuellement, WaveLib 1.0 ne peut générer que la coiflet pour $N = 1$.

Source en C

```
FILTER *h, *g;

h=coiflets(N);
g=bimod(h);

KillFilter(h);KillFilter(g);
```

TAB. 4.7 - : *Code pour filtres C_N.*

Type	Coefficients
N=1	-0.07273262
deb=-2	0.33789766
fin=3	0.85257202
	0.38486485
	-0.07273262
	-0.015655728

TAB. 4.8 - : **Coiflets.**

4.3.2 Fonctions ondelettes

Nous donnons des exemples d'ondelettes générées avec les filtres précédents par la méthode cascade sur 5 niveaux (i.e. avec un écart entre valeurs de 2^{-5} ou une erreur de 2^{-5}). Pour créer une ondelette par l'**algorithme cascade**, il faut spécifier soit le nombre de cascade à effectuer L, soit le nombre de points désirés M. Si on veut faire des cascades, on donne $L > 0$ et $M = 0$. Sinon, en fonction de M, l'algorithme

détermine le nombre de niveaux nécessaires pour obtenir autant de points. On peut également spécifier le niveau j auquel est généré l'ondelette ainsi qu'un paramètre de translation k. Les ondelettes créées sont de même support que celui des filtres et auront un écart de 2^{j-L} entre échantillons. *phia* est la fonction d'échelle du banc d'analyse, *phis* celle du banc de synthèse. *psia* est la fonction ondelette du banc d'analyse et *psis* celle du banc de synthèse.

Source en C

```
.../* Les filtres ont été créés. On dispose de ha, ga, hs, gs */
printf("\nNb de niveaux de récursion L = ");scanf("%d",&L);
printf("\nNb de points M = ");scanf("%d",&M);
printf("\nNiveau j = ");scanf("%d",&j);
printf("\nTranslation k = ");scanf("%d",&k);

Phi(ha,hs,&L,&M,&phia,&psia,&phis,&psis,j,k); ...
...
KillFilter(ha);KillFilter(hs); w_free(phia);w_free(phis);
w_free(psia);w_free(psis);
```

TAB. 4.9 - : *Code pour créer des ondelettes.*

Notez également que si on a créé une ondelette et qu'on souhaite la dilater ou la translater, il suffit d'utiliser

phia2=**dilate**(phia, j, k);
pour créer *phia*2 dilatée de j et translatée de k.

On peut afficher les 4 fonctions sous Matlab en tapant **wloadtst**.

– **Ondelettes de Daubechies.**

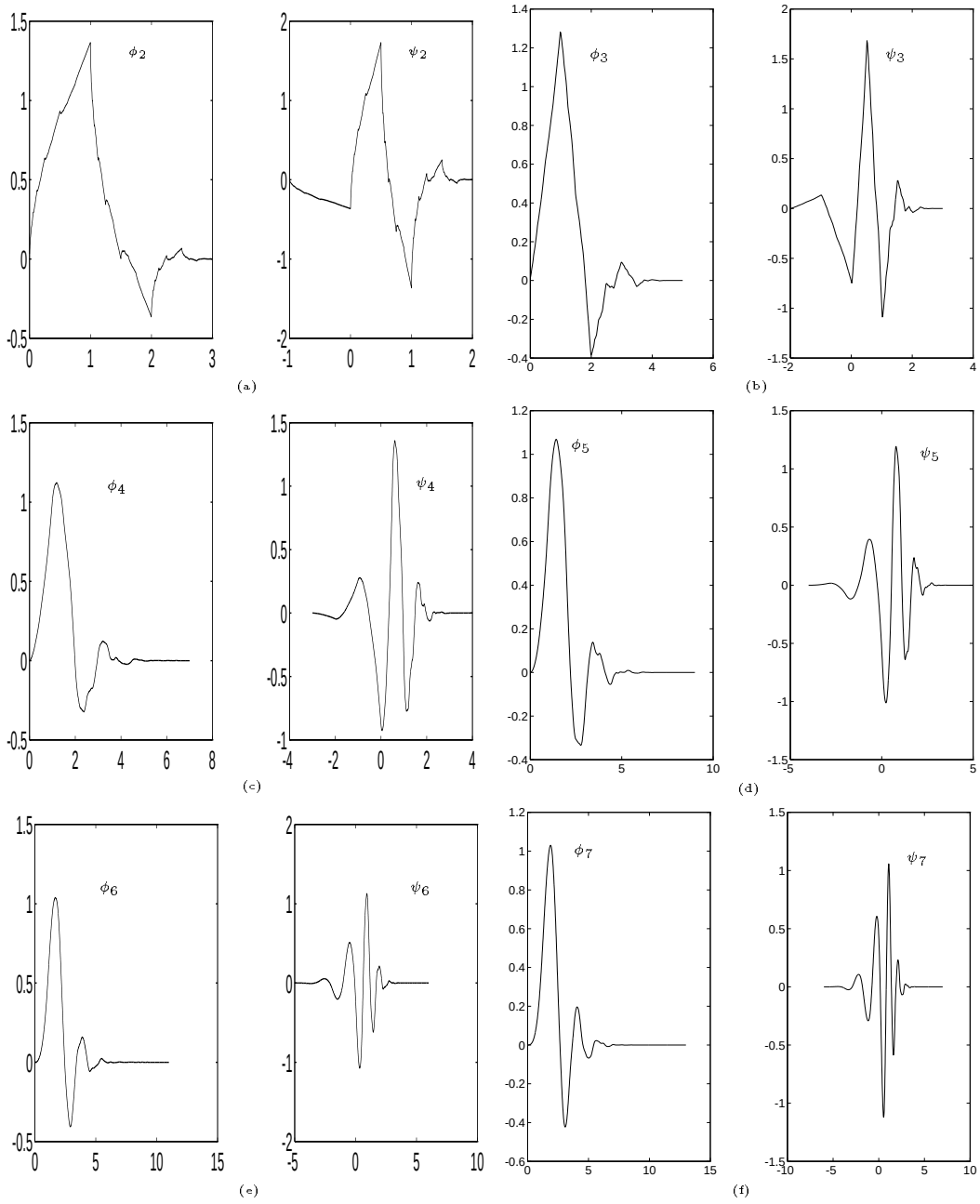


FIG. 4.2 - : **Fontions d'échelle et ondelette de Daubechies.** *(a) pour D_2, (b) pour D_3, (c) pour D_4, (d) pour D_5, (e) pour D_6, (f) pour D_7. Nous rappelons que pour D_1 ce sont les fonctions de Haar.*

– **Ondelettes biorthogonales splines.**
On notera que ces fonctions sont symétriques autour de 0 si N, $\tilde{N}$ sont pairs. Dans le cas où N, $\tilde{N}$ sont impairs, ϕ et ψ sont symétriques par rapport à 0 et, $\tilde{\phi}$ et $\tilde{\psi}$ sont anti-symétriques par rapport à 1/2.

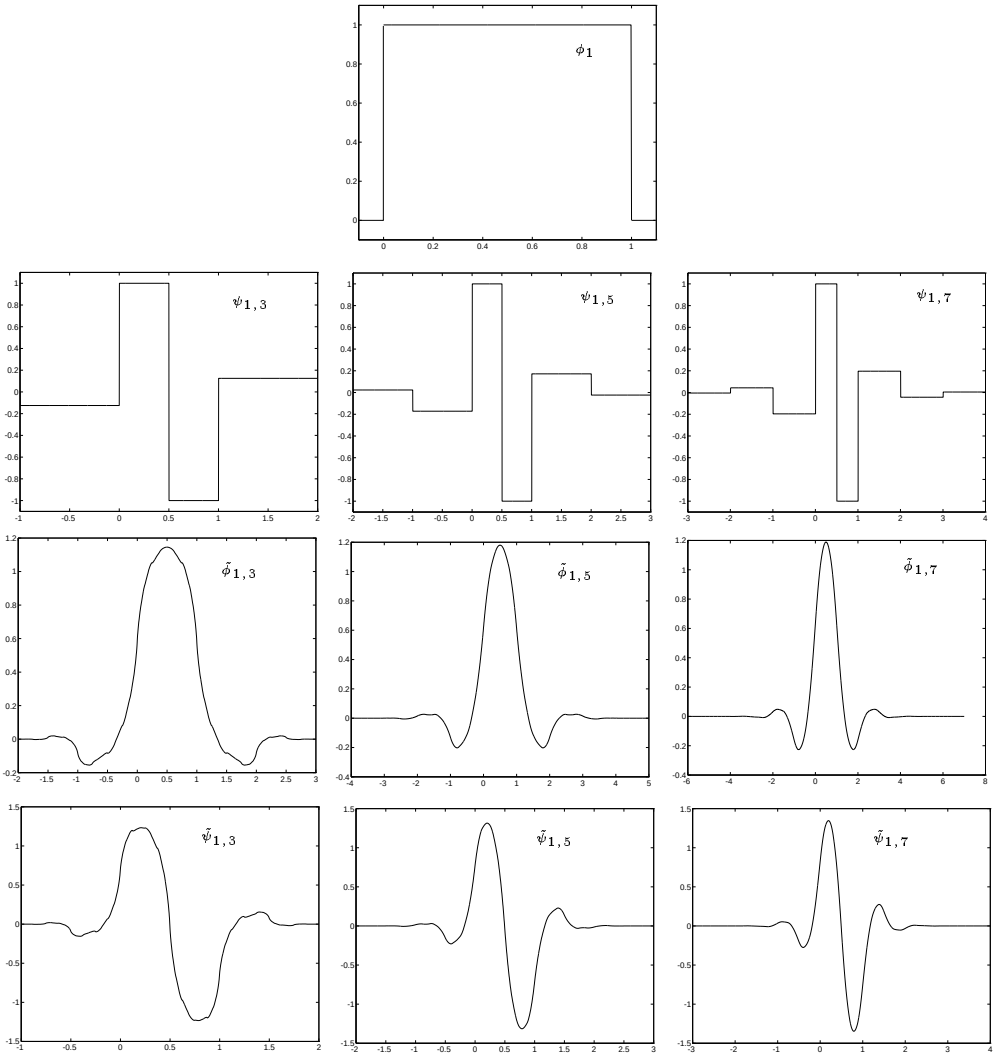


FIG. 4.3 - : **Fontions d'échelle et ondelette biorthogonales splines.** *(a) pour* $B_{1,3}$*, (b) pour* $B_{1,5}$*, (c) pour* $B_{1,7}$*.*

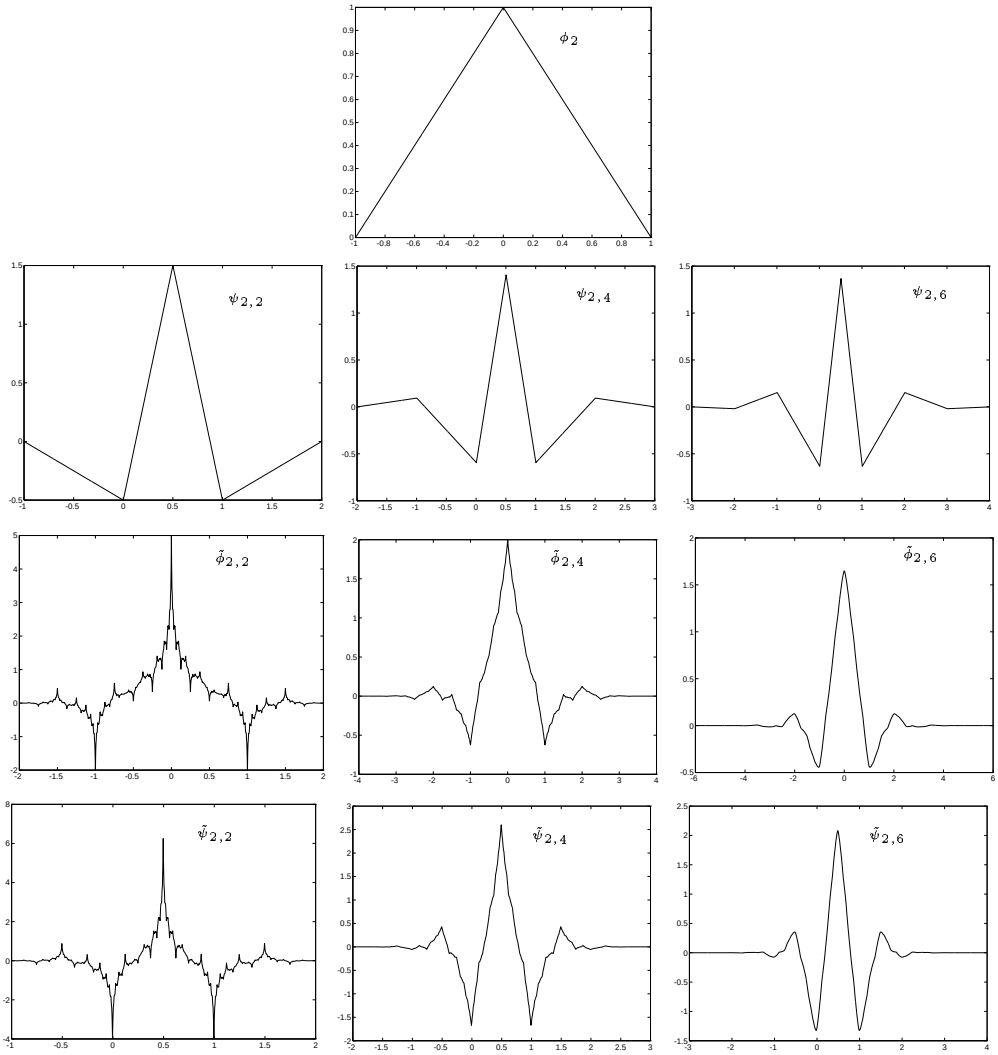


FIG. 4.4 - : **Fontions d'échelle et ondelette biorthogonales splines.** *(a) pour $B_{2,2}$, (b) pour $B_{2,4}$, (c) pour $B_{2,6}$.*

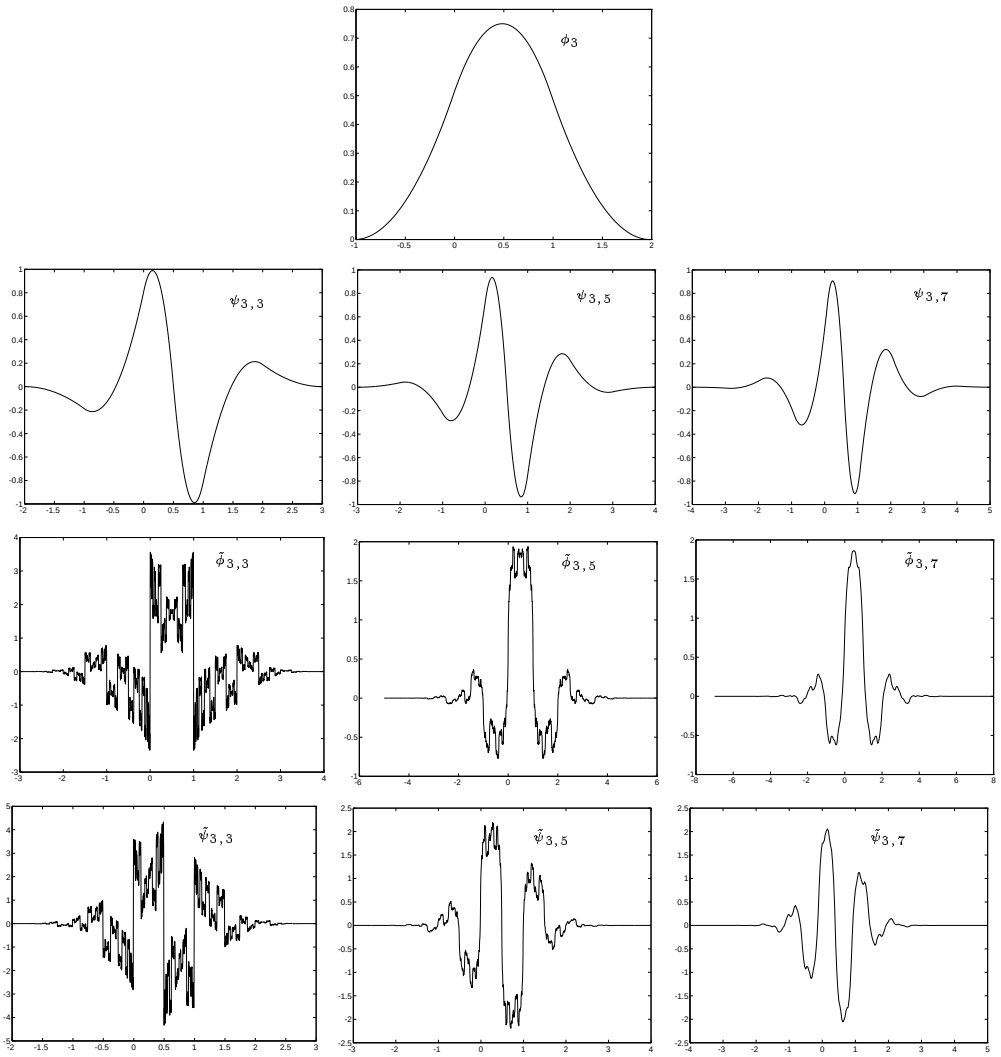


FIG. 4.5 - : **Fontions d'échelle et ondelette biorthogonales splines.** *(a) pour $B_{3,3}$ qui plus photogéniques que régulières mais génèrent toutefois des bases duales de Riesz, (b) pour $B_{3,5}$, (c) pour $B_{3,7}$. Notez que $B_{3,1}$ n'est pas de carré intégrable et n'est pas représenté ici.*

– **Pseudo-coiflets.**

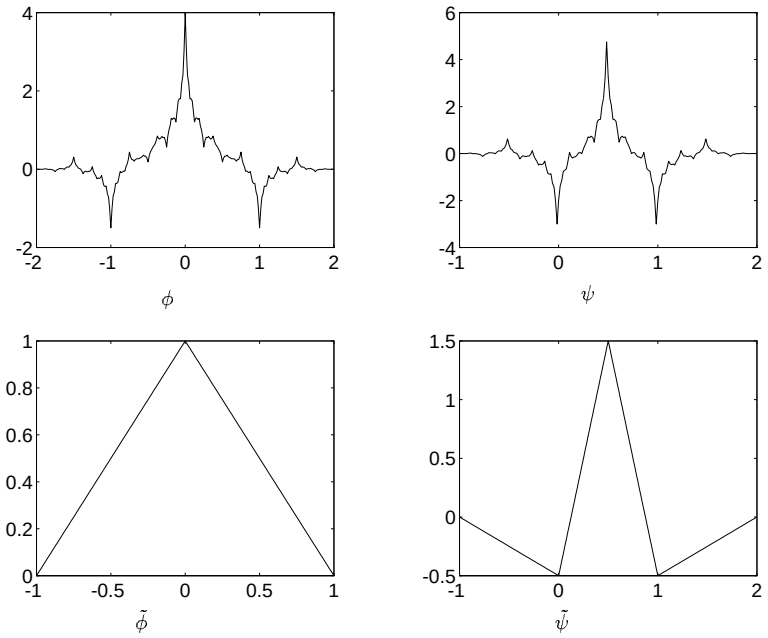


FIG. 4.6 - : *Fontions d'échelle et ondelette pour* $P_{1,1}$.

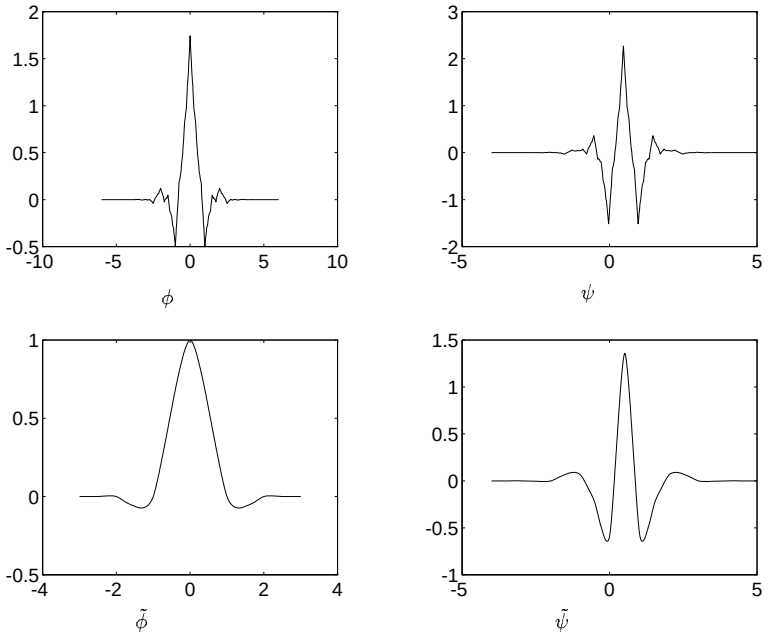


FIG. 4.7 - : *Fontions d'échelle et ondelette pour* $P_{2,2}$.

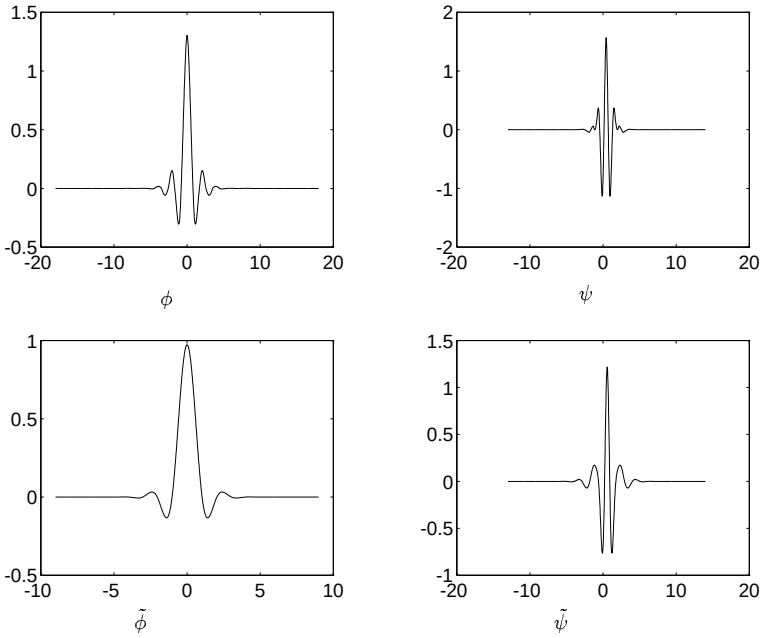


FIG. 4.8 - : *Fontions d'échelle et ondelette pour* $P_{5,5}$.

– **Coiflets.**

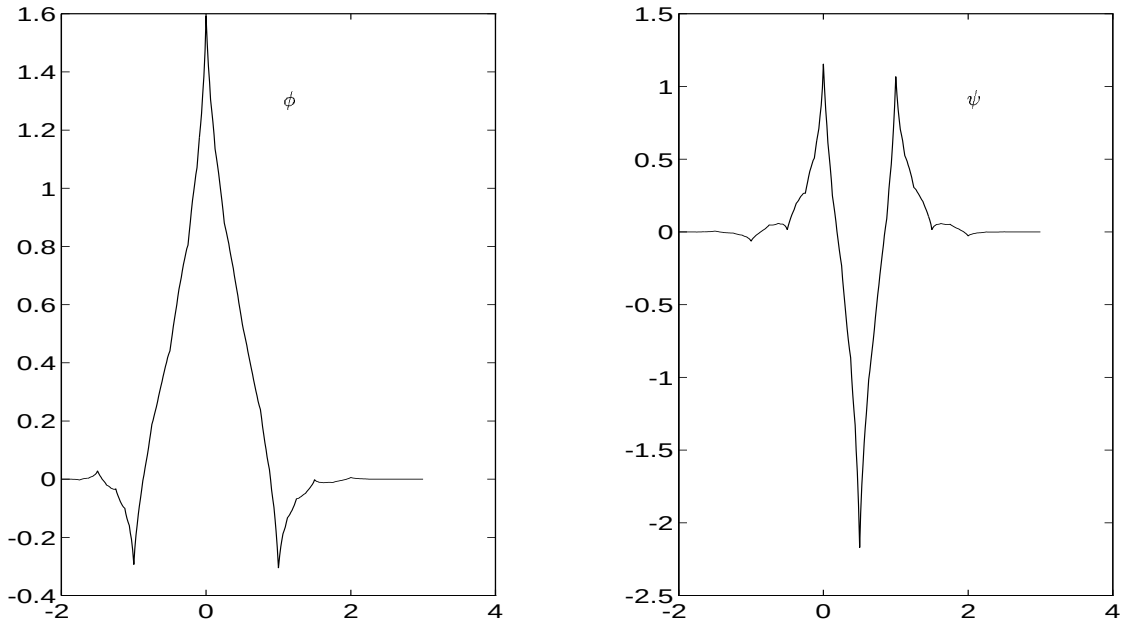


FIG. 4.9 - : *Fontions d'échelle et ondelette pour* C_1.

4.3.3 Transformées ondelettes

En 1D

Soit un signal composé de $2^8 = 256$ échantillons nuls sauf en 2 endroits en $t = 64$ et $t = 128$ où il vaut 1; ce sont deux Dirac. Dû aux grandes transitions en ces

endroits, l'analyse temps-fréquence doit donner, surtout en hautes fréquences, une analyse des plus fines. En effet, à l'endroit des Dirac (temporellement), il apparait deux rectangles très fins en temps mais très longs en fréquence, signe d'une haute fréquence à cet endroit du signal.

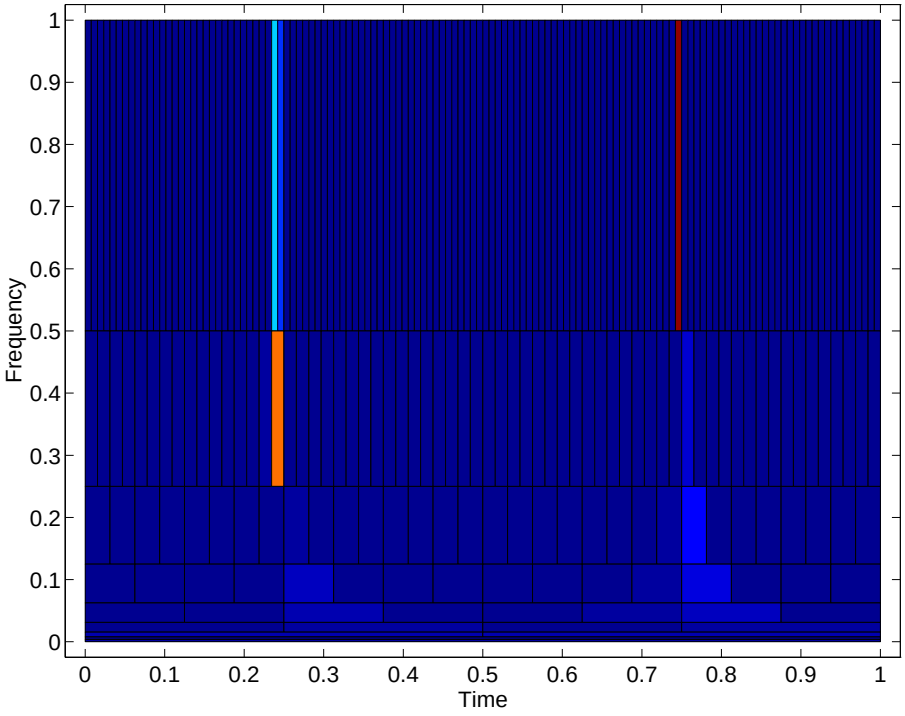


FIG. 4.10 - : *Diagramme temps/fréquence de 2 dirac.*

Nous allons utiliser un signal, *chirp linéaire*, dont la particularité est d'être linéaire en fréquence. Il s'agit d'une fonction du type

$$f(x) = \sin(100\pi t^2)$$

.

Pour réaliser une transformée ondelette, nous présentons le code C le plus général :

Pour l'utilisation des fonctions plus spécialisées de transformation ondelette voir le **manuel de l'utilisateur**.

Sous Matlab, nous récupérons les données générées par WaveLib pour les traiter graphiquement. Tout d'abord, analysons la décomposition par paquets d'ondelettes

Appliquons maintenant les concepts des différentes transformées (bases ondelettes, best level et best basis) pour voir leur influence sur l'analyse temps-fréquence. Les rectangles sont colorés en fonction de l'énergie des coefficients des bases correspondantes.

Source en C

```
VEC *x, *xr; /* x vecteur original, xr vecteur reconstruit */
FILTER *ha, *ga, *hs, *gs; /* filtres des bancs d'analyse/synthèse */
FILE *fp,*fp2;
int L; /* tel que la taille du signal soit 2^L */

...
/* paramètres externes à initialiser avant tout : */
/* NLEVELS = nb max de niveaux de décomposition possible */
/* CLEVEL = "Coaser Level", dernier niveau de décomposition */
NLEVELS=L;
/* ptree est un tableau externe de Wavelib */
/* dans le cas d'un signal 1D on spécifie D1, 2D on met D2 ..., */
/* on spécifie quelle fonction de coût on veut utiliser */
ptree=ptree_get(D1,ENTROPY); /* alloue l'arbre de pointeurs */

/* les filtres ha et ga du banc d'analyse ont été créés auparavant */
wpd1(ha,ga,x,L); /* décomposition complète sur NLEVELS-CLEVELS
niveaux*/

cost=ctree2btree(TRANSFO_TYPE); /* calcul le coût de la trans-
formation et les bases à retenir pour le type de transformation WAVE-
LET_BASIS, BEST_LEVEL ou BEST_BASIS */

/* sauvegarde les données pour Matlab */
fp=fopen("pcb.dat","w");
fp2=fopen("w.dat","w");
pcb_save(fp,fp2);
fclose(fp);fclose(fp2);

...
xr=iwpd1(hs,gs,L); /* reconstruction à partir des bases retenues */

v_free(x);v_free(xr);
KillFilter(ha);KillFilter(ga);
KillFilter(hs);KillFilter(gs);
```

TAB. 4.10 - : *Code pour réaliser tout type de transformation ondelette.*

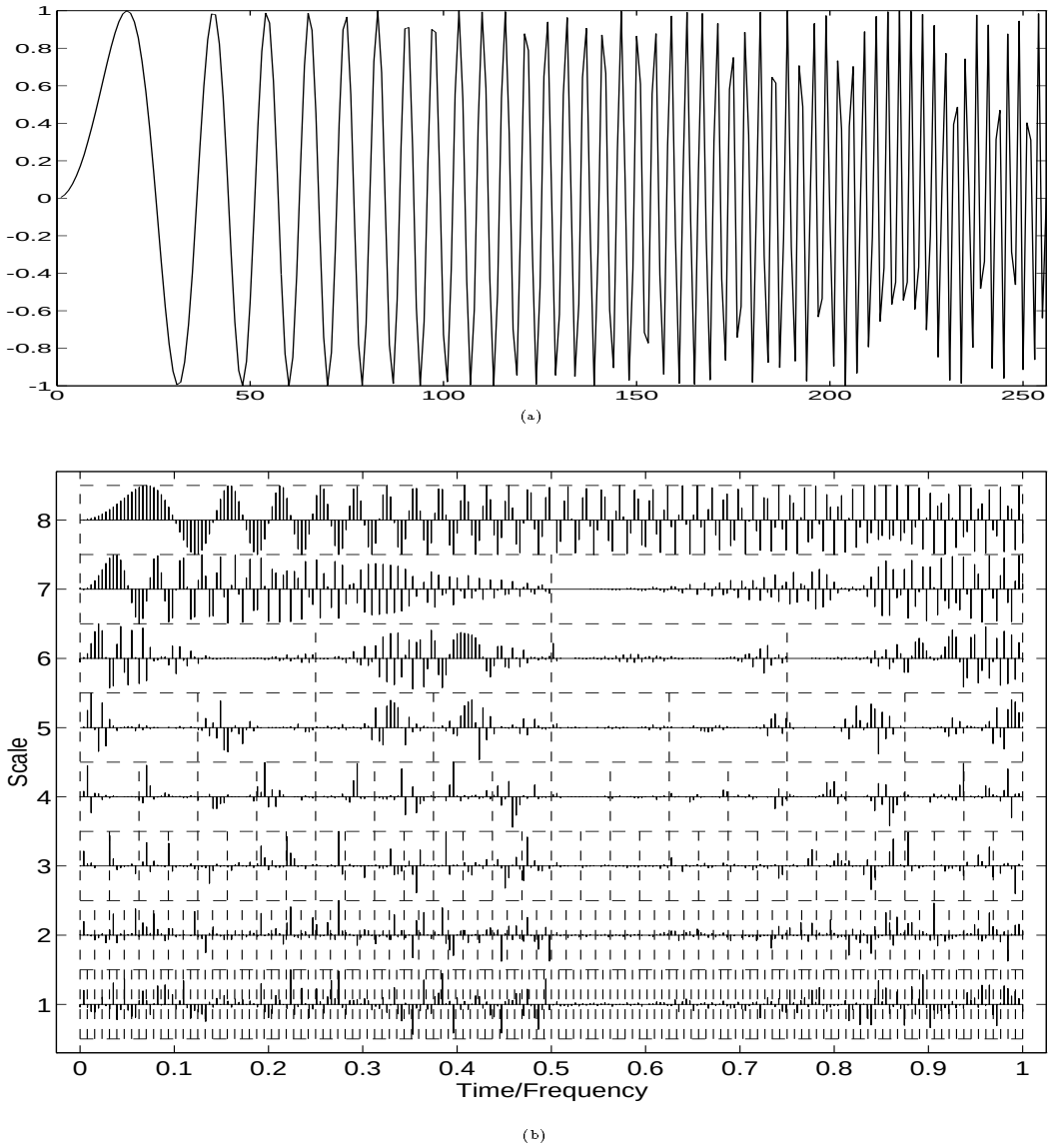


FIG. 4.11 - : **Décomposition d'un chirp linéaire.** *(a) Chirp : $sin(100\pi t^2)$. (b) Décomposition par paquets d'ondelettes avec un D_3. Chaque rectangle a deux fils de même taille, la taille du rectangle père divisée par 2. Le fils de gauche est la version filtrée passe-bas du père et celui de droite celle filtrée passe-bande (voir figure 2.9).*

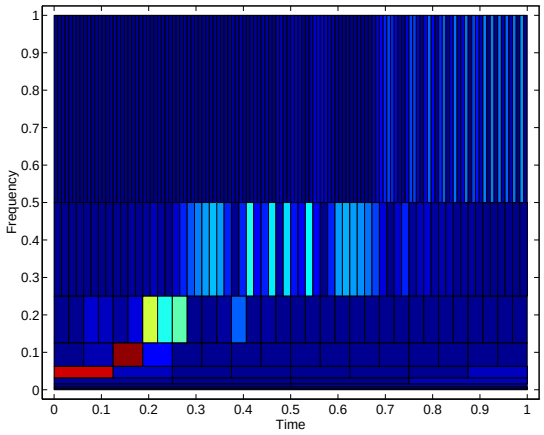


FIG. 4.12 - : **Diagramme temps-fréquence d'un chirp linéaire, base ondelettes.** *Sur des bases ondelettes, on voit très bien une progression linéaire dans les fréquences du signal.*

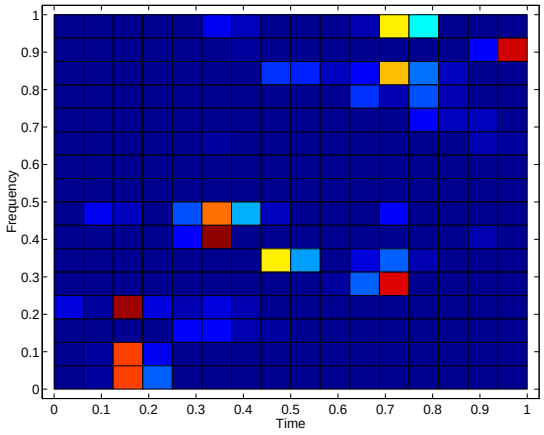


FIG. 4.13 - : **Diagramme temps-fréquence d'un chirp linéaire, best level basis (4).** *Avec une fonction de coût d'entropie minimale, WaveLib a trouvé que le $4^{ième}$ niveau de décomposition est celui dont l'entropie des coefficients est la plus faible.*

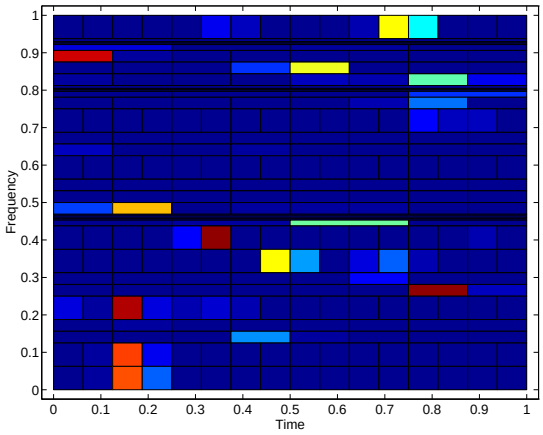


FIG. 4.14 - : **Diagramme temps-fréquence d'un chirp linéaire, best basis.** *Seules les bases contribuant au minimum d'entropie global ont été retenues. Il s'agit d'un diagramme plus difficile à exploiter. Toutefois, on remarquera que de nombreuses bases du best level sont présentes et qu'il s'agit ici du niveau le plus bas de décomposition (4).*

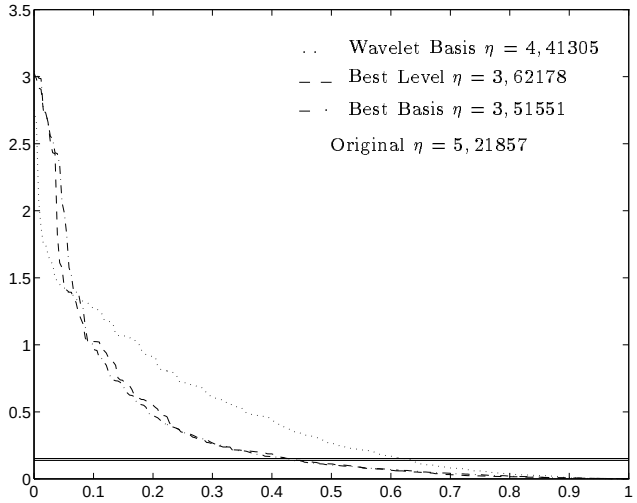


FIG. 4.15 - : **Evolution des coefficients ondelettes.** *Pour chacune des trois méthodes, on a trié les valeurs absolues des coefficients ondelettes par ordre décroissant. Le seuil indiqué correspond à 0,5% du coefficient le plus grand. Dans le cas de la FWT 36% des coefficients seraient retirés. Pour la Best Level, 57% seraient enlevés et pour la Best Basis 58%. Ces coefficients enlevés introduisent une erreur moyenne de reconstruction de* 2.10^{-2}*. Pour une image ce serait mauvais mais pour des applications de calcul numérique cela pourrait suffire.*

En 2D

Pour vérifier nos algorithmes, nous utilisons une image standardisée: Lenna. Elle se compose de 512×512 pixels avec une palette de 256 niveaux de gris.

FIG. 4.16 - : *Lenna* (512×512).

On la décompose sur 3 niveaux.

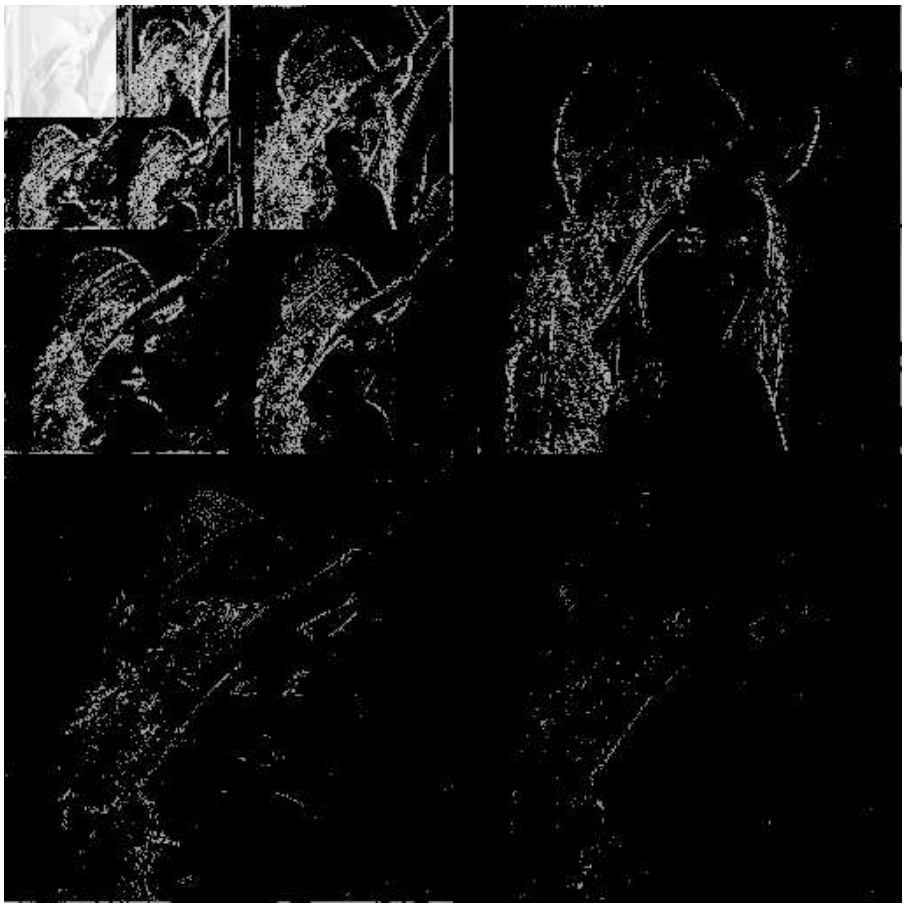


FIG. 4.17 - : *Décomposition ondelette de Lenna* (512×512) *avec un* D_3.

4.3.4 Projection d'une fonction sur différents niveaux

Dans les chapitres précédents, nous avons beaucoup parlé de la décomposition multirésolution d'un signal. Nous allons présenter quelques résultats obtenus.

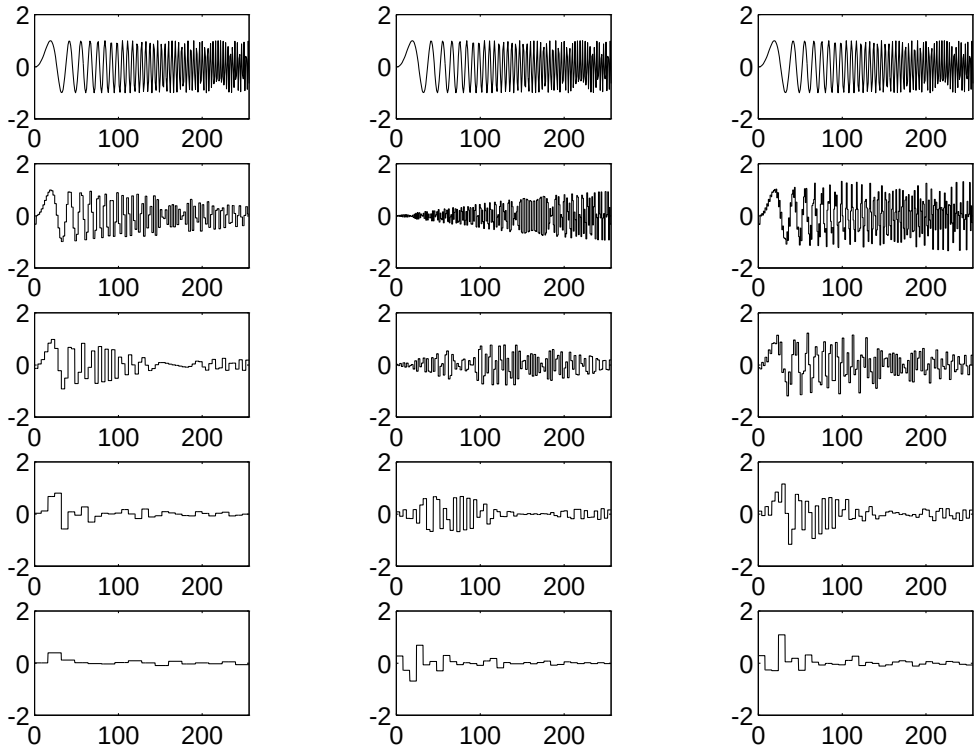


FIG. 4.18 - : **Projection d'un chirp sur 3 niveaux avec un D_1.** *La première ligne n'est que le signal original répété de colonne en colonne. Puis chaque ligne constitute un niveau. Pour le niveau j, la première colonne correspond à la projection sur V_j, la deuxième colonne á la projection sur W_j et la troisième á la reconstruction. Notez l'aspect "marches d'escalier" des fonctions projetées du aux fonctions de Haar.*

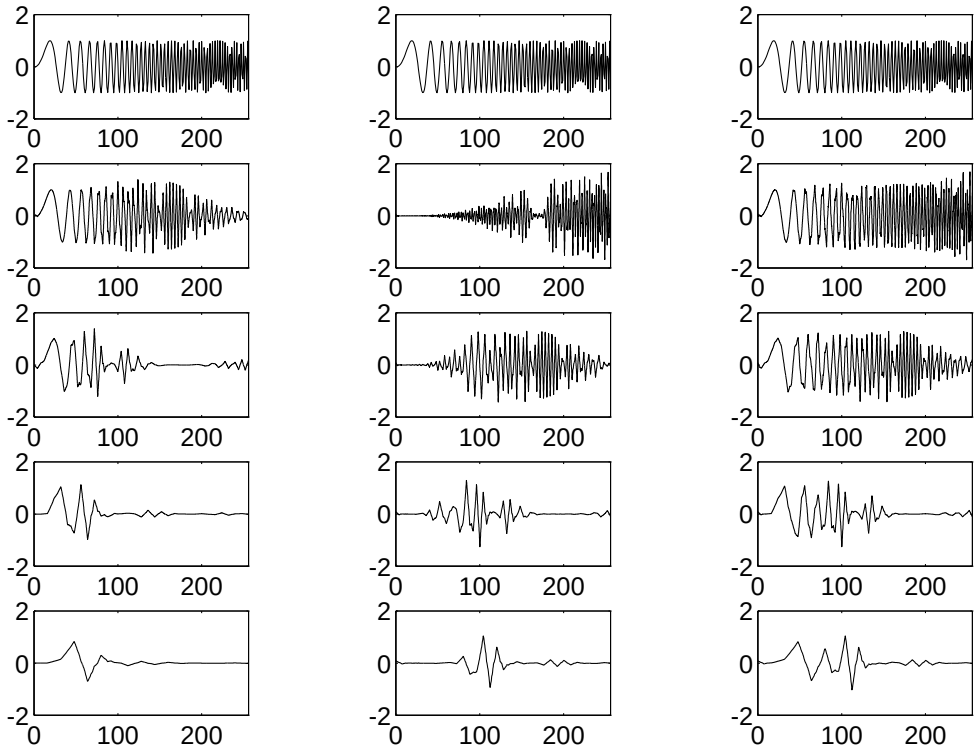


FIG. 4.19 - : **Projection d'un chirp sur 3 niveaux avec un D_3.** *Le nombre de moments nuls ayant augmenté, on a plus de régularité sur les projectées. On remarque également l'effet de filtrage du filtre passe-bas et du filtre passe-haut.*

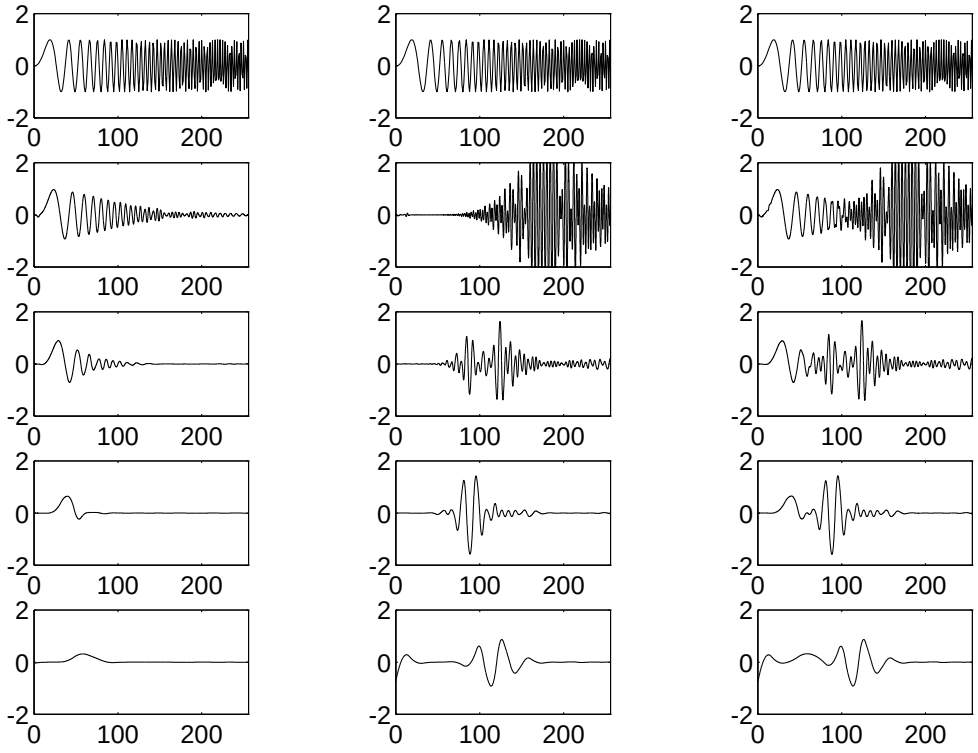


FIG. 4.20 - : **Projection d'un chirp sur 3 niveaux avec un $B_{3,5}$.** *L'effet des fonctions spline pour la fonction d'échelle d'analyse provoque un lissage plus régulier sur les projetées. Notons la concentration du signal sur quelques coefficients seulement de niveaux en niveaux.*

4.3.5 Autres opérations possibles avec WaveLib 1.0

Wavelib 1.0 contient de nombreuses fonctions auxiliaires permettant de réaliser les traitements nécessaires pour résoudre une système d'équations linéaires, un problème de minimisation par les moindres carrés, extraire des valeurs propres et les vecteurs propres associés (pour générer ϕ et ψ , les filtres de Daubechies), réaliser une FFT.

- **Résolution de systèmes linéaires par décomposition LU.** On veut résoudre le système linéaire $A.x = b$ où A est une matrice $(N \times N)$, b est un vecteur $(N \times 1)$. Pour trouver x, on pourrait directement inverser A et la multiplier avec b. Ce processus serait trop coûteux. On décompose $A = L.U$ où L est une matrice unitaire triangulaire inférieure et U une matrice triangulaire supérieure. La décomposition LU est une description algébrique de la méthode d'élimination de Gauss.

Source en C

```
...
/* on suppose connue la matrice A et le vecteur b */
VEC *x;

x=solve(A,b,VNULL);
...
v_free(x);v_free(b);m_free(A);
```

TAB. 4.11 - : *Code pour factorisation LU.*

- **Moindres carrés.** On désire résoudre le problème suivant $\min \|Ax - b\|_2$, de rang plein, par la méthode des moindres carrés. Ici on a plus d'équations que d'inconnus (système surdéterminé) i.e. A est une matrice $(M \times N)$ et b un vecteur $(M \times 1)$, avec $M \geq N$.

Source en C

```
...
/* on suppose connue la matrice A et le vecteur b */
VEC *x;

x=LeastSqSolve(A,b);
...
v_free(x);v_free(b);m_free(A);
```

TAB. 4.12 - : *Code pour solution de systèmes par méthode des moindres carrés.*

– **Extraction de vecteurs et valeurs propres.** On utilise la méthode de la factorisation QR de Hessenberg. Soit une matrice A de taille $(m \times n)$ dont on veut connaitre les valeurs propres, on écrit $A = Q.R$ où Q est une matrice orthogonale de taille $(m \times m)$ et R est triangulaire supérieure de taille $(m \times n)$, avec $m \geq n$. L'itération QR de Hessenberg est utilisée pour calculer les formes de Schur à partir desquelles on peut extraire les valeurs propres et les vecteurs propres. Ce processus est relativement complexe et nous renvoyons le lecteur à des traités de calcul matriciel, notamment l'excellent [31].

Source en C

```
...
/* on suppose connue la matrice A */
CVEC *eig_val; /* vecteur contenant les valeurs propres */
CMAT *eig_vec; /* matrice contenant les vecteurs propres associés */

eigen(A,&eig_val,&eig_vec);

/* à eig_val[i] correspond le vecteur propre *eig_vec[i] */
...
v_free(x);v_free(b);m_free(A);
```

TAB. 4.13 - : *Code pour avoir valeurs et vecteurs propres d'une matrice.*

– **Racines d'un polynôme.** Si on veut les racines d'un polynôme, on peut former d'abord sa matrice companion et chercher les valeurs propres de celle-ci. On appelle matrice companion d'un polynôme $P(x)$, la matrice A telle que

$$\det(A - \lambda I) = P(\lambda) = 0$$

Connaissant les coefficients p_k de $P(x) = \sum_{k=0}^{N} p_k x^k$ on fait

Source en C

```
...
/* on suppose connus les coefficients du polynôme p_k */
MAT *A; /* matrice companion */
CVEC *eig_val; /* vecteur contenant les valeurs propres */
CMAT *eig_vec; /* matrice contenant les vecteurs propres associés */

A=compan(p);
eigen(A,&eig_val,&eig_vec);

/* les racines de P(x) sont dans le vecteur eig_vec */
...
v_free(x);v_free(b);m_free(A);v_free(p);
```

TAB. 4.14 - : *Code pour les racines d'un polynôme.*

– **Fast Fourier Transform - FFT.** Il s'agit là d'un code traduit du Fortran et en cours d'amélioration. Pour faire la FFT d'une signal de 2^N échantillons, on fait

Source en C

```
...

VEC *x_re; /* partie réelle */
VEC *x_im; /* partie imaginaire */

x_re=v_get(2^N);x_im=v_get(2^N);

/* on copie les valeurs du vecteur à transformer dans x_re */
fft(x_re,x_im);

/* et pour l'inverse de la FFT */
ifft(x_re,x_im);
...
v_free(x_re);v_free(x_im);
```

TAB. 4.15 - : *Code pour réaliser FFT et IFFT.*

– **Méthode de Galerkin** On veut résoudre l'équation intégrale linéaire suivante sur l'intervalle $[a, b]$

$$f(x).\gamma + \int_a^b K(x,y)f(y)dy = g(x), \tag{4.1}$$

où

$f : [a,b] \to \mathbb{R}$	est l'inconnu
$K : [a,b] \times [a,b] \to \mathbb{R}$	est le noyeau, connu
$g : [a,b] \to \mathbb{R}$	est le second membre, connu

Cette équation est de la première espèce si $\gamma = 0$, sinon elle est de la deuxième espèce.
Toute équation qui doit être résolue numériquement se doit d'être réduite à un problème de dimensions finies, ou discrétisé [3]. Dans la méthode de Galerkin, on projète f, K et g sur des bases de fonctions. Cette projection est tronquée et le système d'équations linéaire résultant est résolu numériquement.

Soit $b_1, b_2, \ldots$ une base orthonormale de $L^2[a,b]$. La projection de $f, g \in L^2[a,b]$ sur ces bases est donnée par

$$f(x) = \sum_{j=1}^{\infty} f_j b_j(x), \quad g(x) = \sum_{j=1}^{\infty} g_j b_j(x) \tag{4.2}$$

où les coefficients f_j et g_j sont donnés par

$$f_j = \int_a^b b_j(x)f(x)dx, \quad g_j = \int_a^b b_j(x)g(x)dx, \quad j \in \mathbb{N}.$$

Similairement, la projection de $K \in L^2([a,b] \times [a,b])$ est faite sur les deux coordonnées

$$K(x,y) = \sum_{i=1}^{\infty} \sum_{j=1}^{\infty} K_{ij} b_i(x) b_j(y), \tag{4.3}$$

où les coefficients K_{ij} sont donnés par la double intégrale

$$K_{ij} = \int_a^b \int_a^b b_i(x) b_j(y) K(x,y) dx dy, \quad i,j \in \mathbb{N}.$$

En substituant (4.2) et (4.3) dans (4.3.5) conduit à un système infini d'équations

$$f_i.\gamma + \sum_{j=1}^{n} K_{ij} f_j = g_i, \quad i \in \mathbb{N}. \tag{4.4}$$

L'expression de K peut être tronquée en un nombre fini de termes produisant un nombre fini d'équations

$$f_i.\gamma + \sum_{j=1}^{n} K_{ij} f_j = g_i, \quad i = 1, \ldots, n. \tag{4.5}$$

Dans la plupart des applications, la matrice K est dense. Pour l'application d'une matrice $n \times n$ à un vecteur, le coût est $\mathcal{O}(n^2)$. Pour résoudre (4.5), par une méthode du type élimination de Gauss, le coût est $\mathcal{O}(n^3)$ opérations. En fait, la densité de K dépend non seulement du noyau mais aussi de la base. Or, avec des bases ondelettes, K peut devenir creux avec une précision assez grande [8]. On montre alors que projeté sur des bases ondelettes, on peut n'avoir plus que $\mathcal{O}(n)$ à $\mathcal{O}(n \log n)$ éléments non nuls; ce qui simplifiera d'autant le coût calculatoire.

Voici comment réaliser cette méthode:

4.4 Performances, bugs, ...

Wavelib 1.0 en est à sa première version et, comme tout logiciel, connait quelques imperfections. Tout d'abord citons une perfection que tout "waveleter" se doit de connaitre car quoique connue, aucune explication n'a été proposée. Il s'agit d'un problème de calcul des filtres de Daubechies. Les filtres $D_N, \quad N \geq 25$ présentent une anomalie dans leur transformée de Fourier. On remarque une décroissance anormale de la réponse en fréquence alors que théoriquement, plus le nombre de moments nuls N augmente, plus la FFT devrait avoisiner la réponse en fréquence du filtre passe-bas idéal (fonction porte $\chi_{[0,\pi/2]}$).

Source en C

```
int L; /* nombre de niveaux de décomposition pour la FWT non-standard */
float a,b; /* intervalle d'intégration [a, b]
FILTER *ha,*hs; /* filtres des bancs d'analyse ha et de synthèse hs */
MAT *K; /* Kernel */
VEC *g; /* second membre */

...
/* on a calculé les filtres ha et hs */
/* le noyau K a été initialisé ainsi que le second membre g*/
f=galerkin(ha,hs,a,b,L,K,g);

/* all done! f contient le résultat de l'équation intégrale */
...
KillFilter(ha);KillFilter(hs);
v_free(g);v_free(f);
m_free(K);
```

TAB. 4.16 - : *Code pour la méthode de Galerkin.*

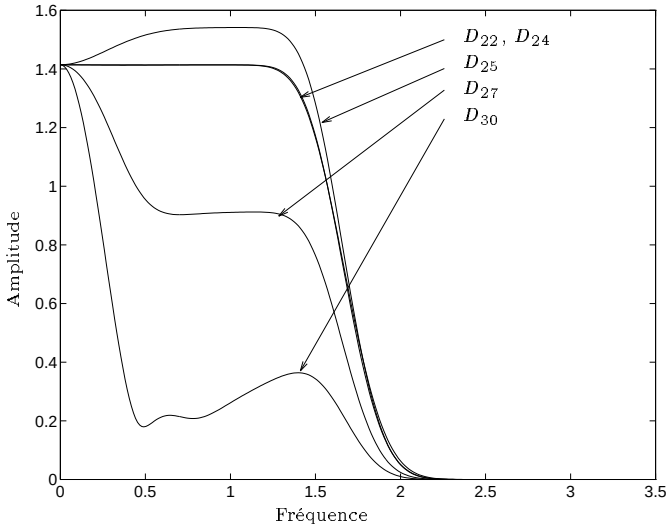


FIG. 4.21 - : **Anomalie dans les filtres D_N, $N \geq 25$.**

Pour évaluer les performances d'un schéma de décomposition-reconstruction, on définit l'*erreur absolue* et l'*erreur relative* comme

$$e_{\text{abs}} \equiv \|X - X^*\|,$$

$$e_{\text{rel}} \equiv \frac{\|X - X^*\|}{\|X\|},$$

on utilise le Peak Signal Noise Ratio (Rapport signal à bruit crête à crête - PSNR) défini par

$$\text{PSNR} \equiv 20 \log_{10} \left[\frac{x_{\text{peak}}}{RMSE} \right] \quad dB$$

avec

$$x_{\text{peak}} \equiv \max_i |x_i|,$$

et l'Erreur Quadratique Moyenne (RMSE - Root Mean-Squared Error) définie par

$$\text{RMSE} \equiv \sqrt{\frac{\sum_{i=0}^{N-1}(x_i - x_i^*)^2}{N}}.$$

Sur une station SUN Sparc 10, voici les temps obtenus avec la procédure générale de décomposition totale par paquets d'ondelettes et recherche de la meilleure base (Best Basis Method) pour une entropie minimale, avec différents filtres, sur un signal *chirp linéaire* $sin(100\pi t^2)$:

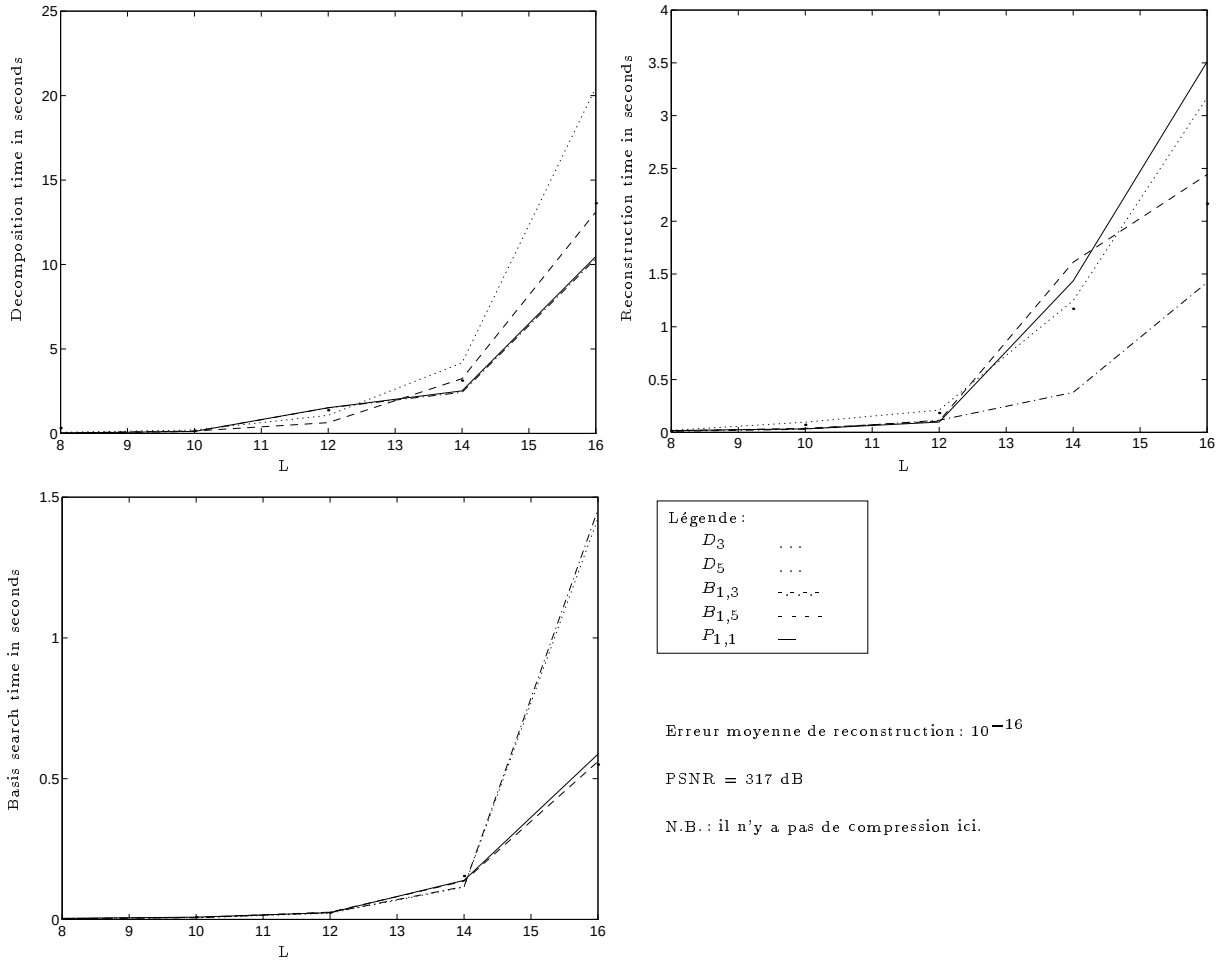


FIG. 4.22 - : **Temps de décomposition, recherche des bases et reconstruction en fonction du nombre de niveaux maximum possibles d'un signal.** *Plus le nombre de coefficients des filtres est grand, plus la décomposition et la reconstruction prennent de temps. Le PSNR est donné à titre indicatif car aucune compression n'est effectuée.*

Bugs

De nombreux développements restent à faire pour agrémenter WaveLib et rendre cette bibliothèque "complète". Actuellement, quelques erreurs subsistent et nous tentons de les corriger :

- le code pour générer des Coiflets d'ordre supérieur à 1 ne marche pas. Il semblerait que la méthode de Newton employée ne soit pas efficace (?)
- la sauvegarde de matrices décomposées par paquets d'ondelettes n'est pas tout à fait générale et est très susceptible!
- le code pour la méthode de Galerkin doit être amélioré pour avoir une erreur de calcul plus faible que 10^{-2},
- autres ?

Développements à rajouter

Parmi les développements à rajouter, voici ceux en cours :

- Généralisation des transformées ondelettes en N-dimensions (N>2),
- Méthode du double quad-tree [35]permettant une adaptativité au des ondelettes et en temps et en fréquence,
- Fonction de coût d'après un critère de débit-taux de distorsion [51],
- Multi-ondelettes ou ondelettes sur l'intervalle,
- Transformées ondelettes avec filtres frontières,
- Ondelettes de Chui, Meyer, Battle-Lemarié...

Chapitre 5

Conclusions

La réalisation d'une bibliothèque de fonctions demande une réflexion approfondie de synthèse tant au niveau théorique qu'au niveau pratique. A travers ce travail, nous avons tenté de rassembler de nombreux concepts tant de traitement du signal que de mathématiques et d'implémenter des outils de base pour tout utilisateur de la théorie des ondelettes.

A travers cette première version de WaveLib, nous avons réalisé, en espérant rendre aisées et simples d'emploi, de nombreuses routines permettant à l'utilisateur, même novice en langage C, de rapidement employer les transfromées ondelettes, de générer divers filtres et leurs ondelettes associées. Sous Matlab, la création d'une interface (graphique) permet de mieux se rendre compte des traitements réalisés par tel ou tel type de filtre ondelette et de manipuler les données générées par la bibliothèque.

Ce logiciel est loin d'être parfait et beaucoup de développements restent à faire pour le rendre complet. Nous souhaitons vivement qu'il servira de base à d'autres chercheurs pour des applications plus complexes et qu'ils contribueront eux-mêmes à son évolution.

Annexe A

Calcul avec des polynômes

A.1 Opérations de base

Soient deux polynômes a et b,

$$a = \sum_{n=\alpha}^{\beta} a_n x^n = a_\alpha x^\alpha + a_{\alpha+1} x^{\alpha+1} + \cdots + a_\beta x^\beta \tag{A.1}$$

$$b = \sum_{n=\gamma}^{\delta} b_n x^n = b_\gamma x^\gamma + b_{\gamma+1} x^{\gamma+1} + \cdots + b_\delta x^\delta \tag{A.2}$$

Ces deux polynômes sont stockés chacun dans son tableau respectif,

$$a = \boxed{a_0}\boxed{a_1}\boxed{a_2}\boxed{\cdots}\boxed{a_N}$$

$$b = \boxed{b_0}\boxed{b_1}\boxed{b_2}\boxed{\cdots}\boxed{b_N}$$

L'addition et la soustraction se font membre à membre et on obtient

$$c = \sum_{n=\min(\alpha,\gamma)}^{\max(\beta,\delta)} c_n x^n. \tag{A.3}$$

La multiplication est réalisée informatiquement par une convolution des deux tableaux a et b: $c = a \star b$. En effet,

$$\begin{aligned} c &= \sum_{n=\alpha}^{\beta} a_n \sum_{m=\gamma}^{\delta} b_m x^{n+m} \\ &= \sum_{n=\alpha}^{\beta} a_n b_{u-n} x^u, \quad u \in [\alpha+\gamma, \beta+\delta] \end{aligned} \tag{A.4}$$

donc c sera de degré $\alpha+\gamma, \cdots, \beta+\delta$.

On notera également que la convolution de deux polynômes de degré N et M donne un polynôme de degré $N+M$.

A.2 Calcul de $(1+x)^N$

Dans Wavelib 1.0 , nous avons souvent besoin de calculer un polynôme du type $\mathcal{X} = (1+x)^n$ dans la synthèse des filtres (3.12), (3.55), (3.56). En passant par la formule du binôme de Newton, il vient

$$(1+x)^N = \sum_{n=0}^{N} \left(\begin{array}{c} N \\ n \end{array} \right) x^n \tag{A.5}$$

Une autre manière très simple de calculer les coefficients $\mathcal{X}_n = \left(\begin{array}{c} N \\ n \end{array} \right)$ est de réaliser des convolutions successives des tableaux $[1 \quad 1]$

$$(1+x)^N \equiv \star_N \quad [1 \quad 1] \tag{A.6}$$

De fait, en informatique, l'utilisateur doit lui-même savoir quelles sont les "bornes" du polynôme.

Annexe B

Méthode de Newton pour systèmes non-linéaires

Il s'agit d'une généralisation de la méthode de Newton pour résoudre des équations non-linéaires. La formule itérative pour résoudre $f(x) = 0$ est la suivante:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

où x est une variable simple, ou encore

$$\Delta x f'(x) = -f(x)$$

avec $\Delta x = x_{i+1} - x_i$.

Dans le cas général , cela devient dans le cas de vecteurs, $F(X) = 0$:

$$F = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} \quad X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$$

Le système à résoudre est alors

$$J(X_1).\Delta X = -F(X_1)$$

où $J(X_1)$ est le Jacobien:

$$J(X_1) = \begin{vmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{vmatrix}$$

et $\Delta X = X_{i+1} - X_i$

La méthode est donc d'évaluer le Jacobien et le second membre de l'équation pour X_i donné. Puis, on résoud le système pour trouver le nouveau Δx. Enfin, on incrémente X_i de Δx et on recommence tant que $\det J \neq 0$ (sinon la méthode diverge).

Annexe C

Solutions de l'équation des pseudo-coiflets

On veut résoudre

$$A(x) + A(-x) = 1$$

avec $A(x) = (1+x)^{N+\tilde{N}} P(x)\tilde{P}(x)$ pour les pseudo-coiflets, c'est l'équation (3.63).

Si $\tilde{P}$ vérifie (3.61) alors $\deg \tilde{P} = d_{\tilde{P}} = \tilde{N} - 1$. Soit $Z(x) = (1+x)^{N+\tilde{N}}\tilde{P}(x)$ alors $\deg Z = N + \tilde{N} + d_{\tilde{P}}$.

Ecrire l'équation (3.63) revient à former la matrice du système

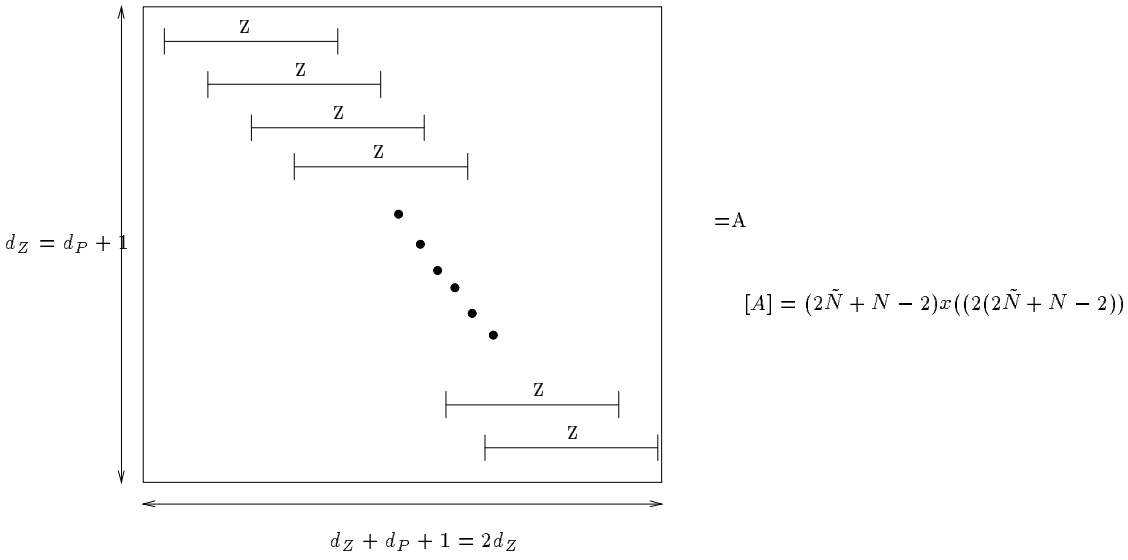


FIG. C.1 - : *Matrice du système des pseudo-coiflets.*

La parité du système implique qu'une colonne sur deux va s'annuler à partir de la deuxième. La solution du système (3.63) est alors donnée par:

Notez que si $\deg Z = 2\tilde{N} + N - 1$ alors la taille de la matrice du système est $[B] = \left(2\tilde{N} + N - 2\right) \times \left(2\tilde{N} + N - 2\right)$ si on veut qu'elle soit carrée donc inversible. Dans ce cas on trouve $\deg P = d_Z - 1 = 2\tilde{N} + N - 2$.

Algorithme

1. Former la matrice $A(x)$ telle que

$$A_{i,k+i} = z_k, \qquad i = 0, \ldots, d_Z$$

$$z_k \ / \ Z(x) = \sum_{k=0}^{d_Z = 2\tilde{N}+N-1} z_k x^k$$

2. Conserver une colonne sur 2: $B_{ik} = A_{i,2k}$

3. Le système est

$$B^T.x = \begin{bmatrix} 2^{N+\tilde{N}-1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Le vecteur de droite contenant d_P zéros.

4. Résoudre le système (décomposition LU par exemple) pour trouver les $2\tilde{N} + N - 2$ coefficients de P.

5. Remplacer $P(x)$ et $\tilde{P}(x)$ dans $m_0(x)$ (3.64) et $\tilde{m}_0(x)$ (3.65).

6. Changement de variable $x = \cos\omega = \frac{z^{-1}+z}{2}$ dans $m_0(x)$ et $\tilde{m}_0(x)$.

7. Calculer $h_k = \sqrt{2}m_0(k)$ et $\tilde{h}_k = \sqrt{2}\tilde{m}_0(k)$

TAB. C.1 - : *Synthèse des pseudo-coiflets.*

Bibliographie

[1] Alpert B., Beylkin G., Coifman R., and Rokhlin V. – Wavelet-like bases for the fast solution of second kind integral equations. – *SIAM J. Sci. Comput.*, 14(1):158–184, January 1993.

[2] Alpert B. K. – A class of bases in l^2 for the sparse representation of integral operators. – *SIAM J. Math. Anal.*, 24(1):246–262, January 1993.

[3] Alpert B.K. – Wavelets and other bases for fast numerical linear algebra. – In C.K. Chui, editor, *Wavelets: A tutorial in theory and applications*, pages 181–216. Academic Press, 1992.

[4] Anderson L., Hall N., and Jawerth B. – Wavelets on closed subsets of the real line.

[5] Antonini M., Barlaud M., Mathieu P., and Daubechies I. – Image coding using wavelet transform. – *IEEE Transactions on Image Processing*, 1(2):205–220, April 1992.

[6] Battle G. – A block spin construction of ondelettes. – *Comm. Math. Phys.*, 110:601–615, 1987.

[7] Belfiore J.C. and Vallet R. – Apport des modulations fractales pour les canaux à évanouissements. – November 1992.

[8] Beylkin G. – On the representation of operators in bases of campactly supported wavelets. – *SIAM J. Numer. Anal.*, 6(6):1716–1740, December 1992.

[9] Beylkin G., Coifman R., and Rokhlin V. – Fast wavelet transforms and numericals algoritms i. – *Communications on Pure and Apllied Mathematics*, XLIV:141–183, 1991.

[10] Bond D.M. and Vavasis S.A. – Fast wavelet transforms for matrices arising from boundary element methods. – March 1994.

[11] Bourges-Sévenier Mikaël. – *WaveLib 1.0 User's guide.* – IRISA/INRIA, Campus de Beaulieu, 35042 RENNES CEDEX, FRANCE, September 1994.

[12] Burt P. and Adelson E. – The laplacian pyramid as a compact image code. – *IEEE Transactions on Communications*, 31:482–540, 1983.

[13] Chui C.K. – *An introduction to wavelets.* – Academic Press, 1992.

[14] Chui C.K. – *Wavelets: A tutorial in theory and applications.* – Academic Press, 1992.

[15] Chui C.K. and Wang J.-Z. – On compactly supported spline wavelets and a duality principle. – *Transactions of the American Mathematical Society*, 330(2):903–915, April 1992.

[16] Cohen A. – Biorthogonal wavelets. – In C.K. Chui, editor, *Wavelets: A tutorial in Theory and Applications*, pages 123–152. Academic Press, 1992.

[17] Cohen A., Daubechies I., and Feauveau J.-C. – Biorthogonal bases of compactly supported wavelets. – *Communications on pure and applied mathematics*, XLV:485–560, 1992.

[18] Coifman R.R. and Meyer Y. – Orthonormal wave packet bases.

[19] Coifman R.R., Meyer Y., Quake S., and Wickerhauser M.V. – Signal processing and compression with wave packet. – Num. Algo. Research Group, Dept of math., Yale University, April 1990.

[20] Coifman R.R. and Wickerhauser M.V. – Entropy-based algorithms for best basis selection. – *IEEE Transactions on Information Theory*, 38(2):713–718, March 1992.

[21] Coifman R.R. and Wickerhauser M.V. – Best-adapted wave packet bases. – Num. Algo. Research Group, Dept of mathematics, Yale University, May 1990.

[22] Daubechies I. – *Ten lectures on wavelets.* – CBMS-NSF, 1993.

[23] Daubechies I. – Orthonormal bases of compactly supported wavelets - ii. variations on a theme. – *SIAM J. Math. Anal.*, 24:499–519, March 1993.

[24] Daubechies I. – Orthonormal bases of compactly supported wavelets. – *Communications on Pure and Applied Mathematics*, 41:909–996, 1988.

[25] Daubechies I. and Lagarias J.C. – Two scale difference equations ii. local regularity, infinite products of matrices and fractals. – *SIAM J. Math. Anal.*, 24(4):1031–1079, July 1992.

[26] Daubechies I. and Lagarias J.C. – Two scale difference equations i. existence and global regularity of solutions. – *SIAM J. Math. Anal.*, 22(5):1388–1410, September 1991.

[27] Delyon B. – Ondelettes orthogonales et biorthogonales. – Publication Interne 732, IRISA, November 1993.

[28] Donoho D.L. – Interpolation wavelet transforms. – October 1992.

[29] Donoho D.L. and Johnstone I.M. – Ideal spatial adaptation by wavelet shrinkage. – April 1993.

[30] Donoho D.L. and Johnstone I.M. – Minimax estimation via wavelet shrinkage. – Dept of Statistics, Stanford University, 1991.

[31] Golub G.H. and Van Loan C.F. – *Matrix computation* 2^{nd} *edition.* – The Johns Hopkins University Press, 1991.

[32] Gootman E.C. and Wickerhauser M.V. – Elementary wavelets. – Dept. of Math., University of Georgia.

[33] Hastings Cecil Jr. – *Approximations for digital computers.* – Princeton University Press, 1955.

[34] Herley C. – *Wavelets and Filter Banks.* – PhD thesis, Columbia University, 1993.

[35] Herley C., Kovačević J., Ramchandran K., and Vetterli M. – Tilings of the time-frequency plane: construction of arbrtrary orthogonal bases and fast tiling algorithms. – *IEEE Transactions on Signal Processing*, 41(12):3341–3359, December 1993.

[36] Hlawatsch F. and Boudreaux-Bartels G.F. – Linear and quadratic time-frequency signal representations. – *IEEE Signal Processing magazine*, pages 21–67, April 1992.

[37] Jain A.K. – *Fundamentals of Digital Image Processing.* – Info and System Siences series. Prentice Hall, 1989.

[38] Jawerth B. and Sweldens W. – Wavelet multiresolution analyses adapted for fast solution of boundary value ordinary differential equations.

[39] Jawerth B. and Sweldens W. – An overview of wavelet based multiresolution analyses. – March 22 1993.

[40] Learned R.E. and Willsky A.S. – A wavelet packet approach to transient signal classification. – *submitted to Applied and Computational Harmonic Analysis*, September 1993.

[41] Lemarié P.G. – Ondelettes à localisation exponentielle. – *J. de Math. Pures et Appl.*, 67(3):227–236, 1988.

[42] Lewis A.S. and Knowless G. – Image compression using the 2d wavelet transform. – *IEEE Transactions on Image Processing*, 1(2):244–250, April 1992.

[43] Lina J.M. – Parametrizations for daubechies wavelets.

[44] Lina J.M. and Mayrand M. – Complex daubechies wavelets.

[45] Mallat S.G. – Multifrequency channel decomposition of images and wavelet models. – *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):2091–2110, December 1989.

[46] Mallat S.G. – A theory for multiresolution signal decomposition: the wavelet representation. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7):674–693, July 1989.

[47] Mallat S.G. – Review of multifrequency channel decompositions of images and wavelet models. – Tech Report 412, New-York University, Courant Institute of Math.ematical Sciences, November 1988.

[48] Meyer Y. – *Ondelettes, in Ondelettes et Opérateurs.* – Hermann, 1990.

[49] Muraki S. – Volume data and wavelet transforms. – *IEEE Computer Graphics and Applications*, pages 50–56, July 1993.

[50] Porter D. and Stirling D.S.G. – *Integral Equations.* – Cambridge Texts in Applied Mathematics, 1990.

[51] Ramchandran K. and Vetterli M. – Best wavelet packet bases in a rate-distorsion sense. – *IEEE Transactions on Image Processing*, 2(2):160–175, April 1993.

[52] Reissell L.-M. – Multiresolution. geometric algorithms using wavelets i: representation for parametric curves and surfaces. – Tech. report 93-17, May 1993.

[53] Rioul O. – Simple regularity criteria for subdivision schemes. – *SIAM J. Math. Anal.*, 23(6):1544–1576, November 1992.

[54] Ruskaï et al. – *Wavelets and their applications.* – Jones et Barlett, 1992.

[55] Ruskaï M. B. et al. – *Wavelets and their applications.* – Jones and Bartlett, 1992.

[56] Saito N. and Beylkin G. – Multiresolution representations using the auto-correlation functions of compactly supported wavelets. – *IEEE Transactions on Signal Processing*, 41(12):3584–3590, December 1993.

[57] Sankar P.V. and Ferrari L.A. – Simple algorithms and architectures for b-spline interpolation. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(2):271–276, March 1988.

[58] Schoenberg I.J. – Cardinal interpolation and spline functions. – *Journal of Approximation Theory*, (2):167–206, 1969.

[59] Schumaker. – *Spline functions - Basic theory.* – Interscience series of texts, monographs abd tracts. Addison-Wesley.

[60] Shann W.-C. – *Finite elements methods for Maxwell equations with stationary magnetic field and Galerkin-wavelets methods for two-point boundary value problem.* – PhD thesis, The Pennsylvania State University, 1991.

[61] Shann W.-C. – Quadrature rules needed in galerkin-wavelets methods. – Dept. of Math., National Central University, Taïwan.

[62] Shann W.-C. and Yan J.-C. – Quadrature involving polynomials and daubechies wavelets. – Preprint, Dept. of Math., National Central University, Taïwan, April 1994.

[63] Shensa M. J. – The discrete wavelet transform: wedding the á trous and mallat algorithms. – *IEEE Transactions on Signal Processing*, 40(10):2464–2482, October 1992.

[64] Smith M.J.T. and Barnwell T.P. – Exact reconstruction techniques for tree-structured subband coders. – *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 34(3):434–441, June 1986.

[65] Soman A.K., Vaidyanathan P.P., and Nguyen T.Q. – Linear phase paraunitary filter banks: theory, factorizations and designs. – *IEEE Transactions on Signal Processing*, 41(12):3480–3495, December 1993.

[66] Steffen P., Heller P.N., Gopinah R.A., and Burrus C.S. – Theory of regular m-band wavelet bases. – *IEEE Transactions on Signal Processing*, 41(12):3497–3511, December 1993.

[67] Strang G. – Wavelets and dilatation equations: a brief introduction. – *SIAM Review*, 31(4):614–627, December 1989.

[68] Sweldens W. and Piessens R. – Asymptotic error expansion of wavelet approximations of smooth functions ii: generalizations.

[69] Taswell C. and McGill K.C. – Wavelet transform algo for finite-duration discrete-time signals.

[70] Tewfik A.H., Sinha D., and Jorgensen P. – On the optimal choice of a wavelet for signal representation. – *IEEE Transactions on Information Theory*, 38(2):747–765, March 1992.

[71] Unser M. and Aldroubi A. – Fast b-spline transforms for continuous image representation and interpolation. – *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):277–285, March 1991.

[72] Unser M., Aldroubi A., and Eden M. – A family of polynomial spline wavelet transforms. – *Signal Processing*, (30):141–162, 1993.

[73] Vaidyanathan P.P. – *Multirate Systems and Filter Banks.* – Prentice Hall Signal Processing Series, 1993.

[74] Vaidyanathan P.P. and Hoang P.-Q. – Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction qmf banks. – *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 36(1):81–94, January 1988.

[75] Vetterli M. and Herley C. – Wavelets and filter banks:theory and design. – *IEEE Transactions on Signal Processing*, 40(9):2207–2232, September 1992.

[76] Wickerhauser M.V. – Inria lectures on wavelet packet algorithms. – In *Problèmes non linéaires appliqués - Ondelettes et paquets d'ondes*, pages 31–99. CEA-EDF-INRIA, June 1991.

[77] Wickerhauser M.V. – Acoustic signal compression with wave packets. – August 1989.

[78] Wickerhauser M.V. – Non-standard matrix multiplication. – Num. Algo. Research Group, Dept. of math., Yale University, May 1990.

[79] Wornell G.W. and Oppenheim A.V. – Wavelet-based representations for a class of self-similar signals with application to fractal modulation. – *IEEE Transactions on information theory*, 38:785–800, March 1992.

[80] Xia X.-G. and Zhang Z. – On sampling theorem, wavelets and wavelet transforms. – *IEEE Transactions on Signal Processing*, 41(12):3524–3535, December 1993.

[81] Xu J.C. and Shann W.C. – Galerkin-wavelet methods for two-point boundary value problems.

[82] Yaou M.-H. and Chang W.-T. – Fast surface interpolation using multiresolution wavelet transform.

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 46 avenue Félix Viallet, 38031 GRENOBLE Cedex 1
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
ISSN 0249-6399