Table des matières

1 Ondelette 1D

L'objectif de cette section est d'expliquer comment on peut approcher une fonction de $L^2(\mathbb{R})$ à une échelle de précision désirée et de manière efficace.

1.1 Un peu de théorie

a) Analyse multirésoltion de $L^2(\mathbb{R})$

L'approche multiéchelle consiste à approcher $L^2(\mathbb{R})$ par une suite croissante de sous espaces notés $(V_j)_{j\in\mathbb{Z}}$. Ces espaces vont du plus grossier ($j \to -\infty$) au plus fin ($j \to +\infty$). C'est ce que traduit la définition d'analyse multirésolution ci-dessous.

Définition 1.

Une analyse multirésoltion de $L^2(\mathbb{R})$ est une suite d'espaces vectoriels $(V_j)_{j\in\mathbb{Z}}$, $V_j \subset L^2(\mathbb{R})$ avec

- $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}, \overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R}), V_j \subset V_{j+1}$
- $f \in V_j \iff f(2\cdot) \in V_{j+1}$
- $f \in V_0 \Longrightarrow f(\cdot - k) \in V_0 \ \forall k \in \mathbb{Z}$
- *Il existe $\phi \in V_0$ telle que $\{\phi(\cdot - k), k \in \mathbb{Z}\}$ est une base hilbertienne de V_0.*

En effet,

- $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$ signifie que pour toute fonction $f \in L^2(\mathbb{R})$, il existe un entier j_0 (suffisamment petit) tel que la fonction f ne soit pas dans V_j pour $j \leq j_0$. Intuitivement, on pourrait dire que lorsque ($j \to -\infty$), les espaces V_j contiennent de moins en moins de fonctions.
- $\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R})$ signifie que quelle que soit la précision souhaitée, toute fonction de $L^2(\mathbb{R})$ peut être approchée par une fonction de V_j (pour j suffisamment grand).

Ces espaces $(V_j)_{j\in\mathbb{Z}}$ permettent donc une description de $L^2(\mathbb{R})$ à différentes échelles. De plus le deuxième point de la définition ci-dessus suppose l'existence d'un facteur d'échelle (ici 2). Cela signifie que les fonctions de V_{j+1} représentent les mêmes phénomènes que celles de V_j mais à une échelle de temps deux fois plus petite.

Enfin, les deux derniers points de la définition ci dessus imposent à V_0 (et donc à V_j) d'avoir une structure simple. L'espace V_0 peut être décrit par la donnée d'une fonction ϕ, la fonction d'échelle.

Le point clé est alors que, sous ces hypothèses, l'orthogonal de V_j dans V_{j+1} a une structure semblable à celle de V_j. Cette structure est donnée par l'ondelette que l'on notera ψ.

Précisément, on définit W_j le supplémentaire orthogonal de V_j dans V_{j+1}.
On a alors le résultat suivant :

Théorème 1.

Il existe une fonction ψ telle que $\{\psi(\cdot - k), k \in \mathbb{Z}\}$ soit une base hilbertienne de W_0. De plus, il existe un algorithme de calcul pour obtenir ψ lorsque ϕ est donnée.

La situation est donc la suivante :

$$\begin{aligned} V_{j+1} &= V_j \oplus W_j \\ &= (V_{j-1} \oplus W_{j-1}) \oplus W_j \\ &= V_0 \oplus \left(\oplus_{k=0}^{j} W_k\right) \end{aligned}$$

Et l'on dispose de bases hilbertiennes des différents espaces. Une base hilbertienne de V_j est $\{\phi_{j,k}, k \in \mathbb{Z}\}$, celle de W_j est $\{\psi_{j,k}, k \in \mathbb{Z}\}$ et celle de $L^2(\mathbb{R})$ est $\{\psi_{j,k}, j \in \mathbb{Z}, k \in \mathbb{Z}\}$ où l'on note $f_{j,k}(x) = 2^{j/2} f(2^j x - k)$ si $f \in L^2(\mathbb{R})$.

Le théorème suivant précise le comportement des coordonnées d'une fonction f dans les bases de V_j et W_j. Ce comportement justifie en partie les algorithmes présentés dans les parties suivantes.

Théorème 2.

On note $c_{j,k} =< f, \phi_{j,k} >$ et $d_{j,k} =< f, \psi_{j,k} >$ alors on a le résultat suivant pour f de classe C^N et ψ avec N moments nuls :

$$|2^{j/2}c_{j,k} - f(k2^{-j})| \xrightarrow[j\to+\infty]{} 0$$

$$|d_{j,k}| \leq 2^{-j(N+1/2)}C \text{ ie } d_{j,k} \xrightarrow[j\to+\infty]{} 0$$

b) Décomposition en ondelettes-Algorithme en arbre

Il est possible d'obtenir des relations entre les coefficients $c_{j,k}$ et $d_{j,k}$ entre deux niveaux d'échelle successif. On obtient ainsi des algorithmes en arbre d'une grande rapidité.

Pour cela remarquons que les fonctions ϕ et ψ sont dans V_0 et donc dans V_1, on peut ainsi les décomposer dans la bases $\phi_{1,k}$.

La fonction ϕ donne une relation d'échelle :

$$\phi(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} a_k \phi(2x-k)$$

où $a_k =< \phi_{1,k}, \phi >$.

On note de même $b_k =< \phi_{1,k}, \psi >$ et ainsi

$$\psi(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} b_k \phi(2x-k)$$

Si $f \in V_j$ alors $f = \sum_{k\in\mathbb{Z}} c_{j,k}\phi_{j,k}$.

Or $V_j = V_{j-1} \bigoplus W_{j-1}$ donc f s'écrit aussi $f = \sum_{k\in\mathbb{Z}} c_{j-1,k}\phi_{j-1,k} + \sum_{k\in\mathbb{Z}} d_{j-1,k}\psi_{j-1,k}$.

Les coefficients $c_{j,k}, c_{j-1,k}$ sont les coefficients d'échelle et les $d_{j-1,k}$ sont les coefficients d'ondelette.

Grâce à la relation d'échelle on obtient une formule reliant le niveau $j-1$ au niveau j. En effet,

$$\begin{aligned} c_{j-1,k} &= < f, \phi_{j-1,k} > \\ &= < \sum_{l\in\mathbb{Z}} c_{j,l}\phi_{j,l}, \phi_{j-1,k} > \\ &= \sum_{l\in\mathbb{Z}} c_{j,l} < \phi_{j,l}, \phi_{j-1,k} > \\ &= \sum_{l\in\mathbb{Z}} c_{j,l} < \phi_{1,l-2k}, \phi > \end{aligned}$$

$$\boxed{c_{j-1,k} = \sum_{l\in\mathbb{Z}} c_{j,l} a_{l-2k}}$$

Et

$$\begin{aligned} d_{j-1,k} &= < f, \psi_{j-1,k} > \\ &= < \sum_{l\in\mathbb{Z}} c_{j,l}\phi_{j,l}, \psi_{j-1,k} > \\ &= \sum_{l\in\mathbb{Z}} c_{j,l} < \phi_{j,l}, \psi_{j-1,k} > \\ &= \sum_{l\in\mathbb{Z}} c_{j,l} < \phi_{1,l-2k}, \psi > \end{aligned}$$

$$\boxed{d_{j-1,k} = \sum_{l\in\mathbb{Z}} c_{j,l} b_{l-2k}}$$

De même on trouve une relation entre le niveau j et niveau $j-1$:

$$\boxed{c_{j,k} = \sum_{l\in\mathbb{Z}} c_{j-1,l}\overline{a_{k-2l}} + \sum_{l\in\mathbb{Z}} d_{j-1,l}\overline{b_{k-2l}}}$$

Une décomposition en ondelettes correspondant à $V_j = V_0 \bigoplus \sum_{l=1}^{j-1} W_l$ est :

$$\begin{array}{ccccccc} c_{j,k} & \longrightarrow & c_{j-1,k} & \longrightarrow & c_{j-2,k}\cdots & \longrightarrow & c_{0,k} \\ & \searrow & d_{j-1,k} & \searrow & d_{j-2,k}\cdots & \searrow & d_{0,k} \end{array}$$

Stocker $c_{j,k}$ est alors équivalent à stocker $[c_0, d_0, \cdots, d_{j-1}]$. Ces deux stockages comportent le même nombre de coefficients mais dans le deuxième les valeurs absolues des coefficients sont très petites, on pourra négliger certains de ces coefficients.

1.2 Exemples considérés

On va s'intéresser aux exemples de fonctions f suivantes sur $[0,1]$:

1. $f(x) = e^x$
2. $f(x) = \begin{cases} x & \text{si} \quad x < \dfrac{\pi}{6} \\ x - \dfrac{\pi}{6} & \text{sinon} \end{cases}$
3. $f(x) = \cos(15x)$
4. $f(x) = \cos(60(x - \frac{1}{2}))\cos(3(x - \frac{1}{2}))$

1.3 Ondelette de Haar

Dans toute la suite, on ne considèrera que l'ondelette de Haar, c'est-à-dire $\phi(x) = \chi_{[0;1]}(x)$.
On pose alors, pour $j \in \mathbb{Z}$, $V_j = \{f \in L^2(\mathbb{R})$ telle que pour tout $k \in \mathbb{Z}$, f est constante sur $]k2^{-j}, (k+1)2^{-j}[\}$.

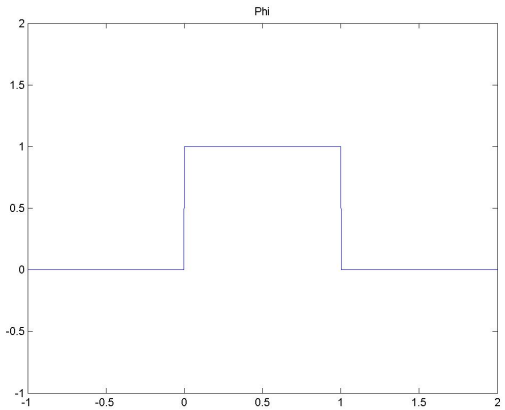


FIG. 1 – Fonction d'échelle ϕ

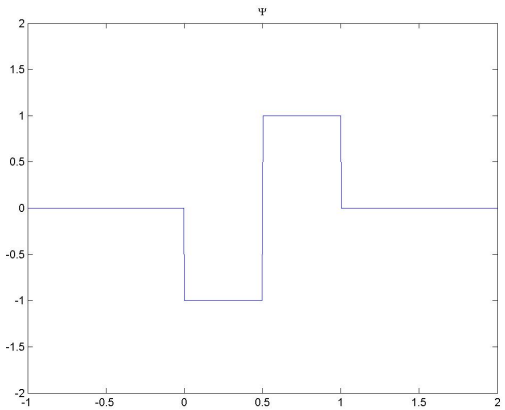


FIG. 2 – Ondelette ψ

Ainsi la relation d'échelle s'écrit :

$$\phi(x) = \sqrt{2}\left(\frac{1}{\sqrt{2}}\phi(2x) + \frac{1}{\sqrt{2}}\phi(2x-1)\right)$$

ie $a_0 = a_1 = \frac{1}{\sqrt{2}}$ et sinon $a_k = 0$. On a également :

$$\psi(x) = \sqrt{2}\left(-\frac{1}{\sqrt{2}}\phi(2x) + \frac{1}{\sqrt{2}}\phi(2x-1)\right)$$

ie $b_0 = -\frac{1}{\sqrt{2}}, b_1 = \frac{1}{\sqrt{2}}$ et sinon $b_k = 0$.

L'algorithme de décomposition du niveau j au niveau $j-1$ s'écrit alors :

$$\boxed{\begin{array}{rcl} c_{j-1,k} & = & \frac{1}{\sqrt{2}}(c_{j,2k} + c_{j,2k+1}) \\ d_{j-1,k} & = & -\frac{1}{\sqrt{2}}(c_{j,2k} - c_{j,2k+1}) \end{array}}$$

Et celui de remonté du niveau $j-1$ au niveau j s'écrit alors :

$$\boxed{\begin{array}{rcl} c_{j,2k} & = & \frac{1}{\sqrt{2}}(c_{j-1,k} - d_{j-1,k}) \\ c_{j,2k+1} & = & \frac{1}{\sqrt{2}}(c_{j-1,k} + d_{j-1,k}) \end{array}}$$

Ces formules conduisent à un moyen efficace de représenter une fonction $f \in L^2(\mathbb{R})$. Choisissons $j \in \mathbb{Z}$ une échelle et notons f_j la projection de f sur l'espace V_j. La fonction f_j est entièrement déterminée par $c_j = (c_{jk})_{k\in\mathbb{Z}}$ le vecteur de ses coordonnées dans la base $(\phi_{jk})_{k\in\mathbb{Z}}$. On effectue ensuite l'algorithme de décomposition pour obtenir sa représentation $[c_0, d_0, \cdots, d_{j-1}]$ dans la base "classique" $\{\phi_{0,k}, \psi_{0,k}, \cdots, \psi_{j-1,k}, k \in \mathbb{Z}\}$. A ce stade, on n'a pas de perte d'informations (la donnée de c_j équivaut à celle de $[c_0, d_0, \cdots, d_{j-1}]$) mais les coefficients de $[c_0, d_0, \cdots, d_{j-1}]$ sont plus petits en valeur absolue. C'est pour cela qu'on a introduit le procédé de compression.

En effet, la compression consiste à négliger les coefficients dont la valeur absolue est en deçà d'une tolérance $\varepsilon > 0$, càd si $|d_{j,k}| < \varepsilon$ alors on impose $d_{j,k} = 0$.

Les résultats que nous avons obtenus sur les exemples cités au paragraphe précédent illustre les points suivants :

- La fonction f_j, projection de f dans V_j, est une "bonne" approximation de f. Le tableau suivant montre la différence entre f et f_j en norme infinie.

	$\|\|f - f_j\|\|_\infty$
Exemple 1	0.0012722
Exemple 2	0.43672
Exemple 3	0.0071728
Exemple 4	0.027056

- Les données compressées constituent encore de "bonne" approximation de f. Le tableau ci-dessous montre la différence, en norme infinie, entre la valeur exacte de f et les valeurs calculées à partir de c_j ou de $[c_0, d_0, \cdots, d_{j-1}]$ en ayant négligé les termes dont la valeur absolue est inférieure a 10^{-4}.

Exemple 1	A partir de c_j	0.0012722
	A partir de $[c_0, d_0, \cdots, d_{j-1}]$	0.003143
Exemple 2	A partir de c_j	0.43672
	A partir de $[c_0, d_0, \cdots, d_{j-1}]$	0.43672
Exemple 3	A partir de c_j	0.0071728
	A partir de $[c_0, d_0, \cdots, d_{j-1}]$	0.0071728
Exemple 4	A partir de c_j	0.027056
	A partir de $[c_0, d_0, \cdots, d_{j-1}]$	0.027056

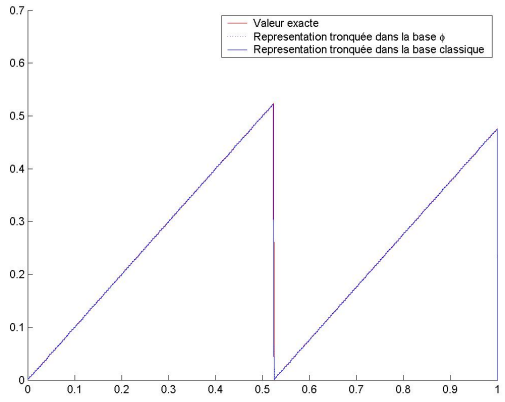


FIG. 3 – Solution pour l'exemple 2

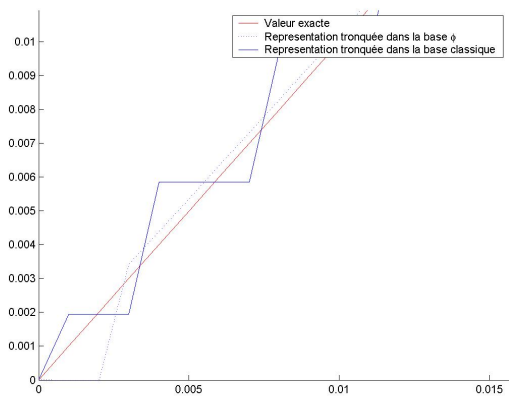


FIG. 4 – Solution zoomée pour l'exemple 2

De plus, les figures 3 et 4 illustrent aussi ce point. Elles représentent la valeur exacte de f, la représentation obenue à partir de c_j puis celle obenue à partir de $[c_0, d_0, \cdots, d_{j-1}]$ dans le cas de l'exemple 2. La figure 3 montre que l'allure est préservée, cependant un zoom (figure 4) montre que la fonction f_j est constituée de pallier. Cette obsevation est bien en accord avec la définition de V_j.

– Enfin, le stockage de $[c_0, d_0, \cdots, d_{j-1}]$ est bien plus efficace. Le tableau ci-dessous montre le nombre de termes à stocker (ie supérieur en valeur absolue à 10^{-4}) dans c_j et dans $[c_0, d_0, \cdots, d_{j-1}]$.

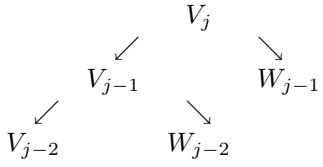
Exemple 1	Dans c_j	1024
	Dans $[c_0, d_0, \cdots, d_{j-1}]$	386
Exemple 2	Dans c_j	1019
	Dans $[c_0, d_0, \cdots, d_{j-1}]$	258
Exemple 3	Dans c_j	1024
	Dans $[c_0, d_0, \cdots, d_{j-1}]$	908
Exemple 4	Dans c_j	1020
	Dans $[c_0, d_0, \cdots, d_{j-1}]$	963

Le stockage de $[c_0, d_0, \cdots, d_{j-1}]$ est plus efficace mais demande un travail avant d'obtenir des données concrètes sur f (remontée à c_j).

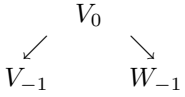
1.4 Paquet d'ondelettes

On a vu que le stockage de la base classique était plus intéressant que celui de c_j. N'y en a t il pas un encore plus intéressant ?

On a décomposé V_j de la manière suivante :



Peut-on aussi décomposer W_j ?
Pour cela, reprenons les notations du paragraphe 1.1 et examinons la décomposition de V_0 :



Cette décomposition est imposée puisque V_{-1} est fixé.
De plus, les fonctions $(\phi_{-1,0}(\cdot - 2k))_{k\in\mathbb{Z}}$ et $(\psi_{-1,0}(\cdot - 2k))_{k\in\mathbb{Z}}$ qui génèrent respectivement V_{-1} et W_{-1} s'obtiennent, à partir de ϕ qui elle "génère" V_0, par les formules suivantes :

$$\phi_{-1,0}(x) = \sum_{k\in\mathbb{Z}} a_k \phi(x-k)$$

$$\psi_{-1,0}(x) = \sum_{k\in\mathbb{Z}} b_k \phi(x-k)$$

Nous raisonnons donc ici à l'inverse, W_0 est donné, il est "généré" par la fonction ψ. Posons alors

$$\psi^1(x) = \sum_{k\in\mathbb{Z}} a_k \psi(x-k)$$

$$\psi^2(x) = \sum_{k\in\mathbb{Z}} b_k \psi(x-k)$$

Les fonctions $(\psi^1(\cdot - 2k))_{k\in\mathbb{Z}}$ et $(\psi^2(\cdot - 2k))_{k\in\mathbb{Z}}$ génèrent alors deux espaces W^1_{-1} et W^2_{-1} qui permettent de décomposer W_0.

On peut en fait montrer un résultat un peu plus général :

Théorème 3.

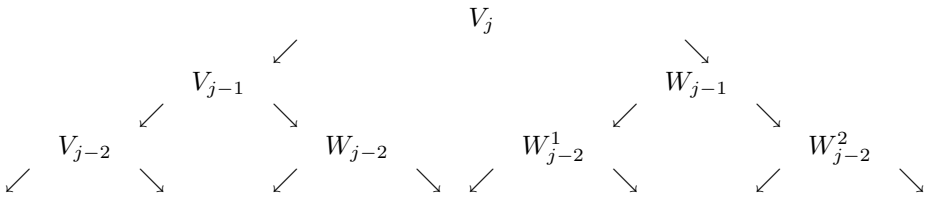
Soit $f \in L^2(\mathbb{R})$ telle que $(f(\cdot - k))_{k\in\mathbb{Z}}$ soit une famille orthonormale. Posons

$$f^1(x) = \sum_{k\in\mathbb{Z}} a_k f(x-k)$$

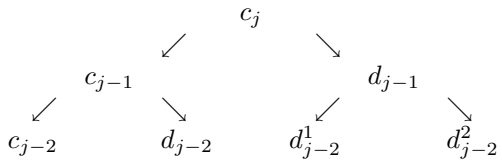
$$f^2(x) = \sum_{k\in\mathbb{Z}} b_k f(x-k)$$

alors $Vect((f(\cdot - k))_{k\in\mathbb{Z}}) = Vect((f^1(\cdot - 2k))_{k\in\mathbb{Z}}) \oplus Vect((f^2(\cdot - 2k))_{k\in\mathbb{Z}})$

Par dilatation, on peut obtenir une décomposition des W_j puis adapter le procédé ci-dessus pour obtenir une décomposition des espaces W^1 et W^2. On obtient une décomposition de la forme suivante :



Ainsi si on représente une fonction f par ces coefficients dans les différentes bases ci dessus, on obtient l'arbre de décomposition suivant :



avec l'algorithme de décomposition du niveau j au niveau $j-1$ suivant :

$$\begin{aligned}
c_{j-1,k} &= \frac{1}{\sqrt{2}}(c_{j,2k} + c_{j,2k+1}) \\
d_{j-1,k} &= -\frac{1}{\sqrt{2}}(c_{j,2k} - c_{j,2k+1}) \\
d^1_{j-1,k} &= \frac{1}{\sqrt{2}}(d_{j,2k} + d_{j,2k+1}) \\
d^2_{j-1,k} &= -\frac{1}{\sqrt{2}}(d_{j,2k} - d_{j,2k+1})
\end{aligned}$$

L'algorithme de remontée est similaire.

Cette décomposition met donc en jeux de nouvelles fonctions de base. On ne connaît pas leur expression explicite mais les graphiques (5, 6) en représentent certaines.

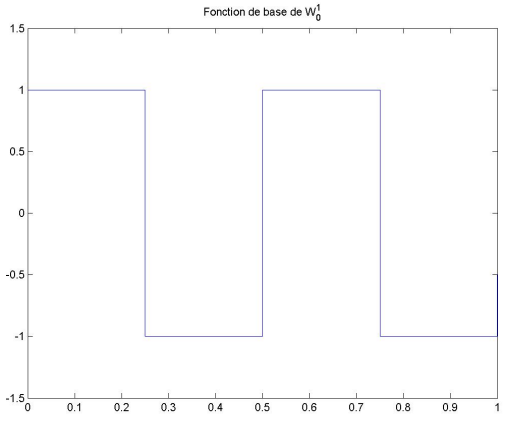


FIG. 5 – Fonction de base de W_0^1

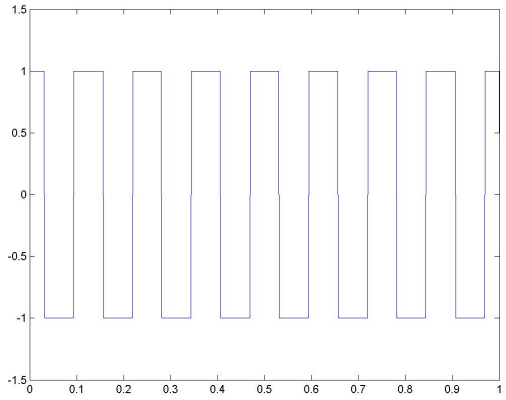
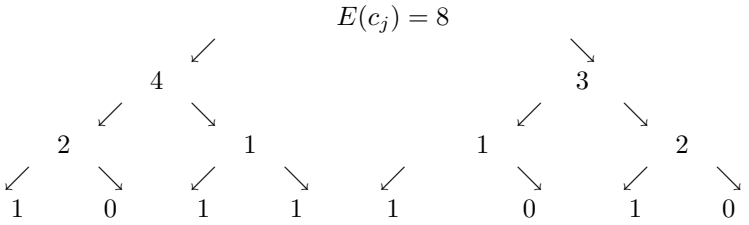


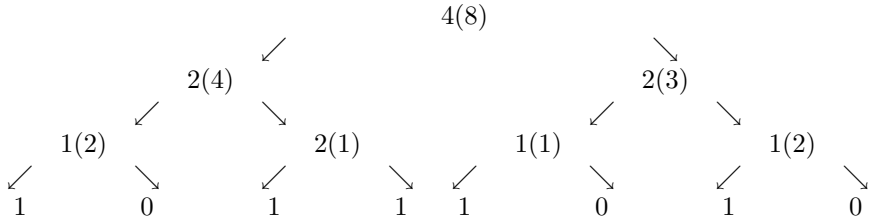
FIG. 6 – Autre fonction de base

Cherchons à obtenir la meilleure base de stockage càd celle qui minimise le coût du stockage. Pour cela on se donne une fonction entropie (ou fonction coût) qui définit ce que l'on entend par coût. Nous avons choisi la définition suivante : L'entropie E d'un vecteur u est le cardinal de l'ensemble $\{u_k, \text{ tels que } |u_k| > \varepsilon\}$ où $\varepsilon > 0$ est une tolérance fixée.
Pour calculer la meilleure base, on utilise un algorithme de remonter dans l'arbre (cf exemple ci-dessous).
Voici un exemple explicite de calcul de la meilleure base :

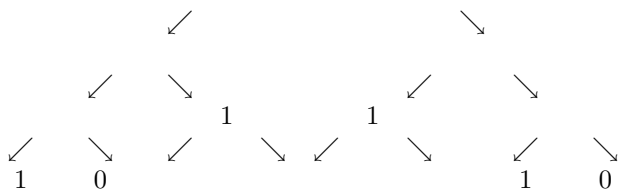
1. La première étape du processus consiste à calculer l'entropie de chacun des éléments de l'arbre de décomposition, cela conduit à un tableau du type :



2. La deuxième étape consiste à parcourir l'arbre en partant du bas, et à comparer la somme de l'entropie obtenue par les enfants à celle obtenue par le parent notée entre parenthèses :



3. A la fin du processus, la base possédant l'entropie la plus faible aura été sélectionnée parmi toutes les bases possibles :
La meilleure base est donc la suivante $[c_{j-3}; d_{j-3}; d_{j-2}; d_{j-2}^1; d_{j-3}^{2,1}; d_{j-3}^{2,2}]$.



Le principe de programmation est le suivant :
On se donne une fonction f et on la projette sur V_j, à j fixé. On effectue l'algorithme de décomposition jusqu'au niveau 0. On obtient un tableau C de taille : $j+1$ lignes et 2^j colonnes. Chaque ligne l correspond à une échelle et les colonnes aux coefficients $(c_{l,k})_{k\in\mathbb{Z}}, (d_{l,k})_{k\in\mathbb{Z}} \cdots$. On calcule l'entropie de tous les coefficients de C. Ensuite l'algorithme sélectionne la meilleure base et retourne deux vecteurs : l'un mb contenant les coefficients sélectionnés dans C et l'autre ech contenant leur position dans le tableau C. Ainsi à partir de ces deux tableaux, on peut reconstruire f projetée sur V_j.
Dans l'exemple précédent on a obtenu : $\begin{array}{rcl} mb & = & [1,0,1,1,1,0] \\ ech & = & [0,0,1,1,0,0] \end{array}$

Appliquons cet algorithme aux 4 exemples choisis.
Observons les résultats :

– Le stockage dans la meilleur base ou plutôt la compression détériore-t-elle la qualité de l'approximation ? Le tableau ci dessous présente la différence en norme infinie entre la valeur exacte et celle calculée à l'aide de la meilleure base.

Exemple 1	0.0036952
Exemple 2	0.44734
Exemple 3	0.0079586
Exemple 4	0.02835

Ces valeurs sont à comparer avec celles obtenues dans le paragraphe précédent. L'approximation est un peu moins bonne mais la différence reste du même ordre de grandeur.
De plus les graphe suivants (7, 8, 9, 10) permettent de comparer la valeur exacte, la représentation dans la base ϕ, celle compressée dans la base ϕ, celle compressée dans la base classique et enfin celle compressée dans la meilleure base pour les 4 exemples de fonctions f :

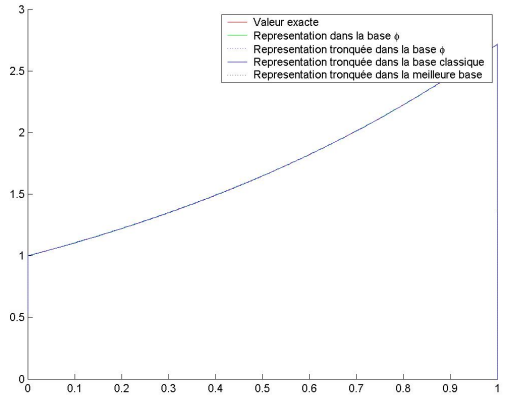


FIG. 7 – Solution pour l'exemple 1

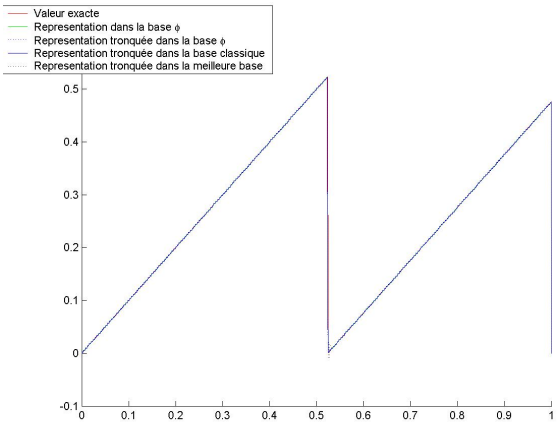


FIG. 8 – Solution pour l'exemple 2

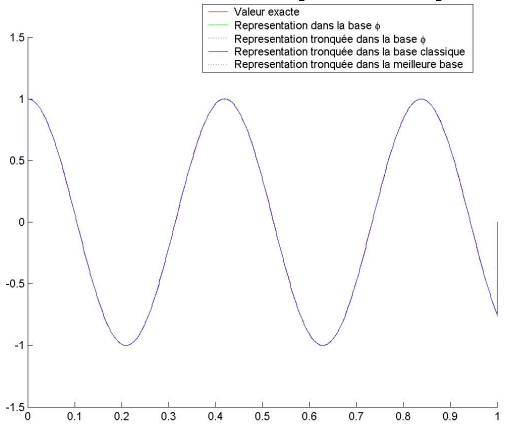


FIG. 9 – Solution pour l'exemple 3

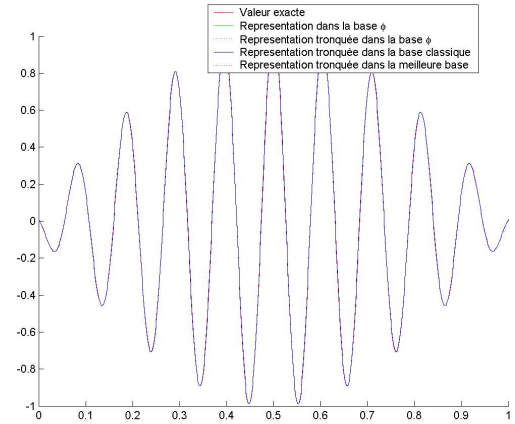


FIG. 10 – Solution pour l'exemple 4

Pour mieux voir leurs différences on a zoomé sur une partie pour obtenir les graphes suivants (11, 12, 13, 14)

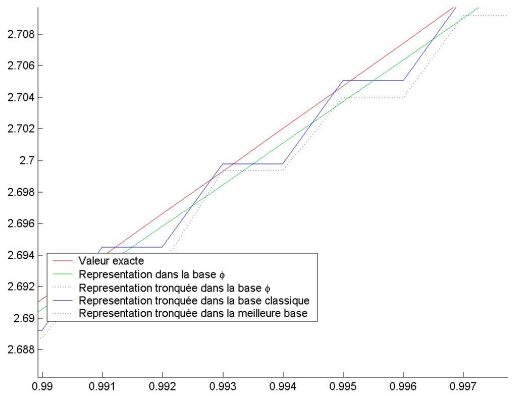


FIG. 11 – Solution zoomée pour l'exemple 1

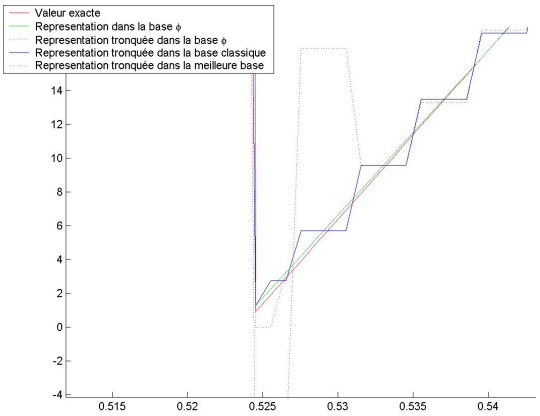


FIG. 12 – Solution zoomée pour l'exemple 2

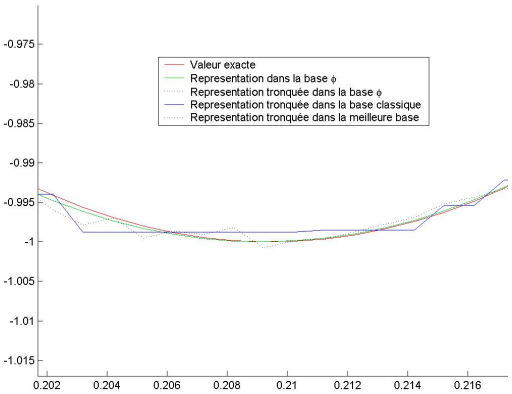


FIG. 13 – Solution zoomée pour l'exemple 3

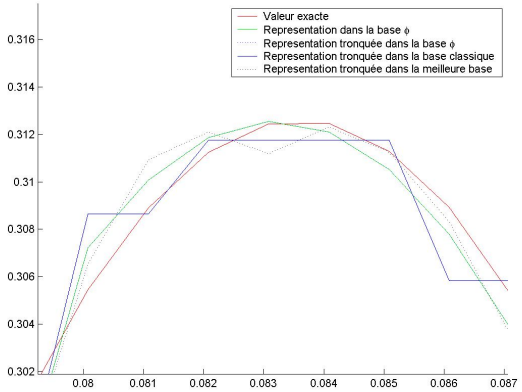


FIG. 14 – Solution zoomée pour l'exemple 4

– Le choix de la meilleure base diminue-t-il significativement le nombre de coefficients à stocker ?
Le tableau suivant reprend le nombre de coefficient à stocker pour une tolérance de 10^{-4}.

Exemple 1	f	1024
	base classique	386
	meilleure base	48
Exemple 2	f	1019
	base classique	258
	meilleure base	31
Exemple 3	f	1022
	base classique	908
	meilleure base	244
Exemple 4	f	1020
	base classique	963
	meilleure base	252

Le stockage à l'aide de la meilleure base permet de stocker beaucoup moins de coefficients tout en conservant une approximation convenable.

2 Ondelette 2D

2.1 Cas général

La base hilbertienne de $L^2(\mathbb{R})$ était $\{\psi_{j,k}, j \in \mathbb{Z}, k \in \mathbb{Z}\}$ pour construire la base hilbertienne de $L^2(\mathbb{R}^2)$ on va juste faire le produit tensoriel des fonctions généré par les deux bases 1D :

$$\psi_{j1,k1;j2,k2}(x,y) = \psi_{j1,k1}(x)\psi_{j2,k2}(y)$$

La base hilbertienne de $L^2(\mathbb{R}^2)$ est donc $\{\psi_{j1,k1;j2,k2}, j1, k1, j2, k2 \in \mathbb{Z}\}$.

Les espaces $V_j, j \in \mathbb{Z}$ deviennent $\mathbb{V}_j = V_j \otimes V_j$. On a toujours $\bigcap_{j\in\mathbb{Z}} \mathbb{V}_j = \{0\}$ et $\overline{\bigcup_{j\in\mathbb{Z}} \mathbb{V}_j} = L^2(\mathbb{R}^2)$. On va de même construire le complémentaire orthogonal $\mathbb{W}_{j-1}$ de $\mathbb{V}_{j-1}$ dans $\mathbb{V}_j$

$$\begin{aligned}
\mathbb{V}_j &= V_j \otimes V_j \\
&= (V_{j-1} \oplus W_{j-1}) \otimes (V_{j-1} \oplus W_{j-1}) \\
&= V_{j-1} \otimes V_{j-1} \oplus [(W_{j-1} \otimes V_{j-1}) \oplus (V_{j-1} \otimes W_{j-1}) \oplus (W_{j-1} \otimes W_{j-1})] \\
&= \mathbb{V}_{j-1} \oplus \mathbb{W}_{j-1}
\end{aligned}$$

L'espace $\mathbb{W}_j$ est consistué de 3 groupes dont les bases hilbertiennes sont $\psi_{j-1,k1}(x)\phi_{j-1,k2}(y)$ (pour $W_{j-1} \otimes V_{j-1}$), $\phi_{j-1,k1}(x)\psi_{j-1,k2}(y)$ (pour $V_{j-1} \otimes W_{j-1}$) et $\psi_{j-1,k1}(x)\psi_{j-1,k2}(y)$ (pour $W_{j-1} \otimes W_{j-1}$). On définit ainsi trois ondelettes :

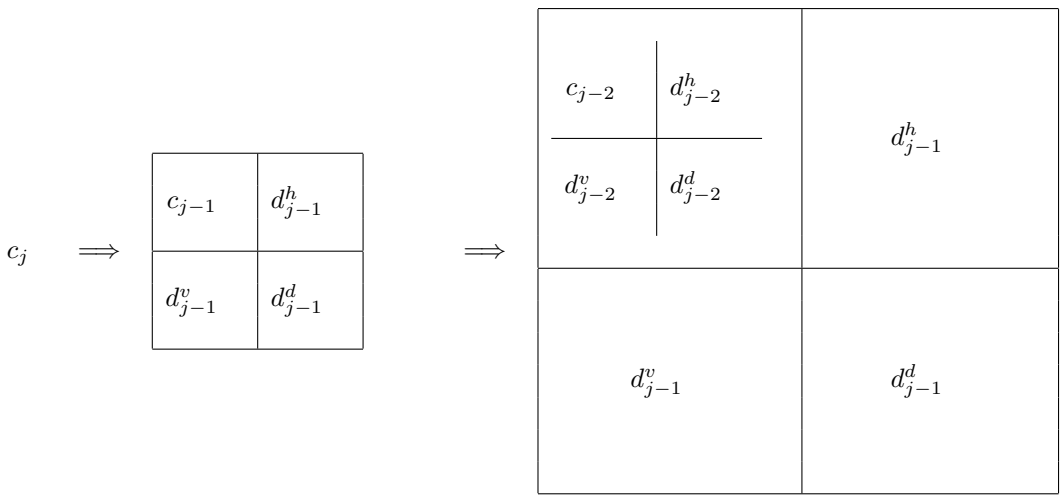
$$\begin{aligned}
\Psi^h(x,y) &= \phi(x)\psi(y) \\
\Psi^v(x,y) &= \psi(x)\phi(y) \\
\Psi^d(x,y) &= \psi(x)\psi(y)
\end{aligned}$$

où les indices h, v, d signifient horizontale, verticale et diagonale.

La base hilbertienne de $\mathbb{W}_j$ est $\{\Psi^\lambda_{j;k1,k2}, k1, k2 \in \mathbb{Z}, \lambda = h, v, \text{ ou } d\}$.

On a de plus que $L^2(\mathbb{R}^2) = \overline{\bigoplus_{j\in\mathbb{Z}} \mathbb{W}_j}$. Donc la base hilbertienne de $L^2(\mathbb{R}^2)$ est $\{\Psi^\lambda_{j;k}, j \in \mathbb{Z}, k \in \mathbb{Z}^2, \lambda = h, v, \text{ ou } d\}$.

Les décompositions en ondelettes correspondant à $\mathbb{V}_j = \mathbb{V}_{j-1} \oplus \mathbb{W}_{j-1}$ puis à $\mathbb{V}_j = (\mathbb{V}_{j-2} \oplus \mathbb{W}_{j-2}) \oplus \mathbb{W}_{j-1}$ sont l'arbre de décomposition suivant :



où les d^h, d^v, d^d sont les détails horizontaux, verticaux, diagonaux.

2.2 Paquet d'ondelettes

Généralisons le cas des paquets d'ondelette en dimension 2 dans le cas de Haar.

Algorithme de décomposition du niveau j au niveau $j-1$:

$$
\begin{aligned}
c_{j-1,k,l} &= \frac{1}{2}(c_{j,2k,2l} + c_{j,2k,2l+1} + c_{j,2k+1,2l} + c_{j,2k+1,2l+1}) \\
d^h_{j-1,k,l} &= \frac{1}{2}(-c_{j,2k,2l} + c_{j,2k,2l+1} - c_{j,2k+1,2l} + c_{j,2k+1,2l+1}) \\
d^v_{j-1,k,l} &= \frac{1}{2}(-c_{j,2k,2l} - c_{j,2k,2l+1} + c_{j,2k+1,2l} + c_{j,2k+1,2l+1}) \\
d^d_{j-1,k,l} &= \frac{1}{2}(c_{j,2k,2l} - c_{j,2k,2l+1}c_{j,2k+1,2l} + c_{j,2k+1,2l+1})
\end{aligned}
$$

Algorithme de remonter du niveau $j-1$ au niveau j :

$$
\begin{aligned}
c_{j,2k+1,2l+1} &= \frac{1}{2}(c_{j-1,k,l} + d^h_{j-1,k,l} + d^v_{j-1,k,l} + d^d_{j-1,k,l}) \\
c_{j,2k,2l} &= \frac{1}{2}(c_{j-1,k,l} - d^h_{j-1,k,l} - d^v_{j-1,k,l} + d^d_{j-1,k,l}) \\
c_{j,2k,2l+1} &= \frac{1}{2}(c_{j-1,k,l} + d^h_{j-1,k,l} - d^v_{j-1,k,l} - d^d_{j-1,k,l}) \\
c_{j,2k+1,2l} &= \frac{1}{2}(c_{j-1,k,l} - d^h_{j-1,k,l} + d^v_{j-1,k,l} - d^d_{j-1,k,l})
\end{aligned}
$$

Le principe de programmation :

On choisit une image du type $2^j \times 2^j$ (j est alors fixé). On effectue l'algorithme de décomposition jusqu'au niveau 0. On obtient un tableau C de taille : $2^j \times 2^j \times (j+1)$. En effet $C(:,:,1)$ contient les valeurs de l'image d'origine, ie $2^j \times 2^j$ coefficients . Le tableau $C(:,:,2)$ contient les valeurs de l'image au niveau $j-1$, ie $2^{j-1} \times 2^{j-1}$ coefficients et les 3 détails $(d^\lambda_{j-1})_{\lambda=h,\ v,\ d}$ chacun de taille $2^{j-1} \times 2^{j-1}$ ce qui fait au total $2^j \times 2^j$ coefficients...

On calcule l'entropie de tous les coefficients de C. Ensuite l'algorithme sélectionne la meilleure base et retourne deux matrices : l'une mb contenant les coefficients sélectionnés dans C de taille $2^j \times 2^j$ et l'autre ech contenant leur position dans le tableau C de taille $(j+1) \times 2^j$. La ligne k de ech contient les numéros des coefficients sélectionnés au niveau k. Ces deux matrices nous permettent de reconstruire l'image.

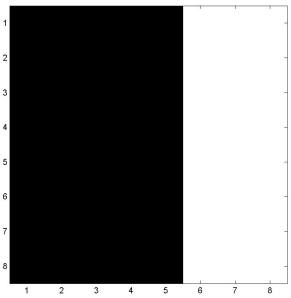
Exemples : $j = 3$

Exemple 1 :

$c_3 =$

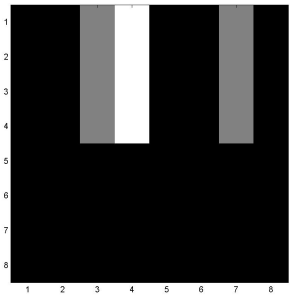
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	1	1	1

$\Longrightarrow$



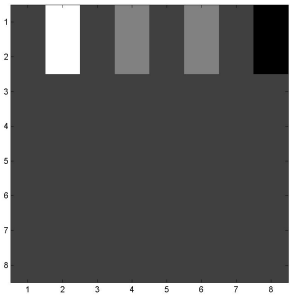
$c_2 =$

0	0	1	2	0	0	1	0
0	0	1	2	0	0	1	0
0	0	1	2	0	0	1	0
0	0	1	2	0	0	1	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\Longrightarrow$ 

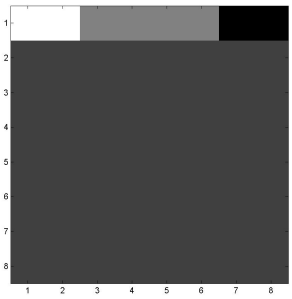
$c_1 =$

0	3	0	1	0	1	0	−1
0	3	0	1	0	1	0	−1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\Longrightarrow$ 

$c_0 =$

3	3	1	1	1	1	−1	−1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\Longrightarrow$ 

On définit l'entropie E d'un tableau par le nombre de coefficients non nuls. On calcule l'entropie des coefficients de c_0 à c_3 :

$E(c_0) =$

1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$E(c_1) =$

2	2	2	2
0	0	0	0
0	0	0	0
0	0	0	0

$E(c_2) =$

8	4
0	0

$E(c_3) =$ $\boxed{24}$

On commence par comparer les tableaux $E(c_0)$ et $E(c_1)$. Pour cela on compare chaque case de $E(c_1)$ à ses quatre cases filles dans $E(c_0)$. Par exemple la première case en haut à gauche $\boxed{2}$ aux quatre cases $\begin{array}{|c|c|}\hline 1 & 1 \\ \hline 0 & 0 \\ \hline \end{array}$. On

choisit de conserver le tableau dont la somme des coefficients est la plus petite. Ici, il y a égalité, on conserve alors le tableau parent. Conserver signifie modifier les tableaux *ech* et $E(c_1)$ en conséquence. Dans le tableau *ech*, on doit ajouter les numéros des cases que l'on conserve et retirer ceux des cases qu'on abandonne. Le tableau $E(c_1)$ est mis à jour en remplaçant la valeur d'une case par la somme des valeurs de ses cases filles (lorsque celle-ci est plus petite). Dans notre exemple ces tableaux valent maintenant :

$E(c_1) =$

2	2	2	2
0	0	0	0
0	0	0	0
0	0	0	0

et ech $=$

1	2	3	...	15	16

Ensuite on reprend ce procédé pour comparer $E(c_1)$ et $E(c_2)$. Après modification on obtient les tableaux suivants :

$E(c_2) =$

4	4
0	0

et ech $=$

2	3	4	
1	2	5	6

Enfin, pour la dernière étape il reste à comparer $E(c_2)$ et $E(c_3)$. On obtient alors le résultat final :

$E(c_3) =$ | 8 |

et ech $=$

2	3	4	
1	2	5	6

On peut alors représenter schématiquement le résultat obtenu :

mb $=$

c_1 | d_1^h / d_1^v | d_1^d	d_2^h
d_2^v	d_2^d

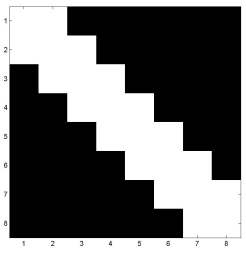
Ici, le stockage à l'aide de la meilleure base permet de ne stocker que 8 coefficients contre 24 dans l'image originale sans aucune perte d'information. Il y a trois fois moins de coefficient à stocker mais il est nécessaire de traiter les données (algorithme de remonté) avant de retrouver l'image initiale.

Exemple 2 :

$c_3 =$

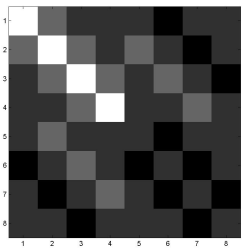
1	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	1	0
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1

$\Longrightarrow$



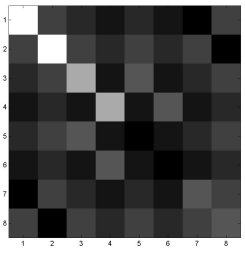
$c_2 =$

2	0.5	0	0	0	−0.5	0	0
0.5	2	0.5	0	0.5	0	−0.5	0
0	0.5	2	0.5	0	0.5	0	−0.5
0	0	0.5	2	0	0	0.5	0
0	0.5	0	0	0	−0.5	0	0
−0.5	0	0.5	0	−0.5	0	−0.5	0
0	−0.5	0	0.5	0	−0.5	0	−0.5
0	0	−0.5	0	0	0	−0.5	0

$\Longrightarrow$ 

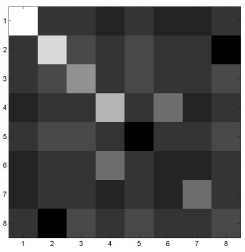
$c_1 =$

2.5	.25	0	−0.25	0	−0.25	−0.5	0.25
0.25	2.5	0.25	0	0.25	0	0.25	−0.5
0	0.25	1.5	−0.25	0.5	−0.25	0	0.25
−0.25	0	−0.25	1.5	−0.25	0.5	−0.25	0
0	0.25	0.5	−0.25	−0.5	−0.25	0	0.25
−0.25	0	−0.25	0.5	−0.25	−0.5	−0.25	0
−0.5	0.25	0	−0.25	0	−0.25	0.5	0.25
0.25	−0.5	0.25	0	0.25	0	0.25	0.5

$\Longrightarrow$ 

$c_0 =$

2.75	0	0	−0.25	0	−0.25	−0.25	0
0	2.25	0.25	0	0.25	0	0	−0.75
0	0.25	1.25	0	0.25	0	0	0.25
−0.25	0	0	1.75	0	0.75	−0.25	0
0	0.25	0.25	0	−0.75	0	0	0.25
−0.25	0	0	0.75	0	−0.25	−0.25	0
−0.25	0	0	−0.25	0	−0.25	0.75	0
0	−0.75	0.25	0	0.25	0	0	0.25

$\Longrightarrow$ 

Vu les valeurs contenues dans les tableaux ci dessus on choisit de négliger les coefficients inférieurs en valeurs absolues à $\varepsilon = 0.3$. On définit donc l'entropie E d'un tableau par le nombre de ses coefficients supérieur à 0.3. On calcule l'entropie des coefficients de c_0 à c_3 :

$E(c_0) =$

1	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
0	0	1	0	0	0	0	0
0	0	0	1	0	1	0	0
0	0	0	0	1	0	0	0
0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0
0	1	0	0	0	0	0	0

$E(c_1) =$

2	0	0	2
0	2	2	0
0	2	2	0
2	0	0	2

$E(c_2) =$

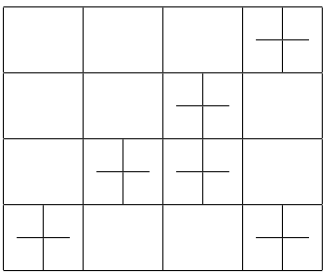
10	6
6	6

$E(c_3) =$

22

Détaillons les étapes de l'algorithme de choix de la meilleure base.

– Première étape : Comparaison entre $E(c_0)$ et $E(c_1)$.
La sélection des échelles qui en découle est représentée par le schéma suivant :



et $E(c_1)$ est modifié de la manière suivante :

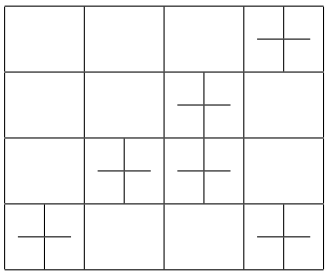
$$E(c_1) = \begin{array}{|c|c|c|c|}\hline 2&0&0&1\\\hline 0&2&1&0\\\hline 0&1&1&0\\\hline 1&0&0&1\\\hline\end{array}$$

– Deuxième étape : Comparaison entre $E(c_1)$ et $E(c_2)$.
Les échelles sélectionnées sont toujours les mêmes mais $E(c_2)$ est modifié de la manière suivante :

$$E(c_2) = \begin{array}{|c|c|}\hline 4&2\\\hline 2&2\\\hline\end{array}$$

– Dernière étape : Comparaison de $E(c_2)$ et $E(c_3)$
Le dernier tableau a une entropie de (4+2+2+2) 10 alors que l'entropie c_3 est de 22 donc on obtient :

mb =



On vient de sélectionner une base qui permet de minimiser le nombre de coefficients supérieur en valeur absolue à 0.3. Ce stockage contient 36 coefficients non nuls. Il y en a plus que dans l'image originale (figure 15) qui en contenait 22. Cependant si on décide de négliger les coefficients inférieur à 0.3 il ne reste plus que 10 coefficients à stocker. Il y a bien sur une perte d'information mais l'image n'est pas "trop dénaturée" (voir figure 16).

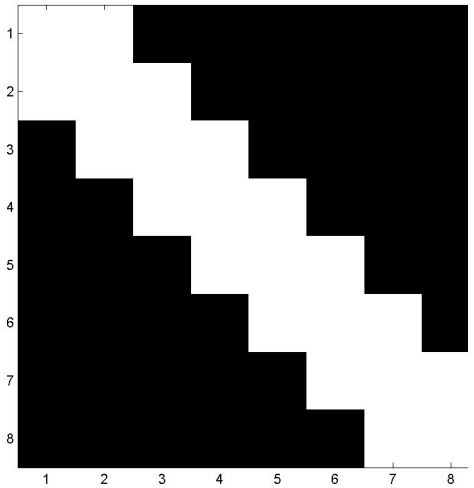


FIG. 15 – Image originale

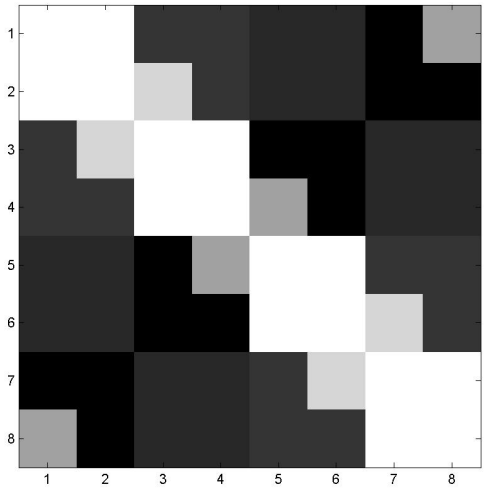


FIG. 16 – Image compressée

Finissons par illustrer notre propos sur une image bien connue des internautes. Cette image est stockée dans un tableau 512×512. On pose donc $j = 9$.

FIG. 17 – Lena

Les figures suivantes correspondent à la décomposition en ondelette au niveau $j-1$ figure 18 et $j-2$ figure 19.

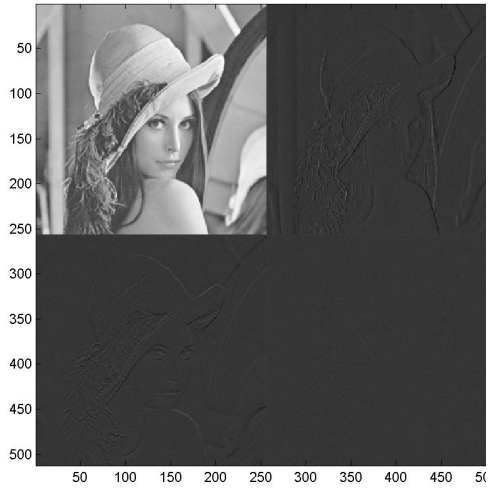


FIG. 18 – Décomposition au niveau 8

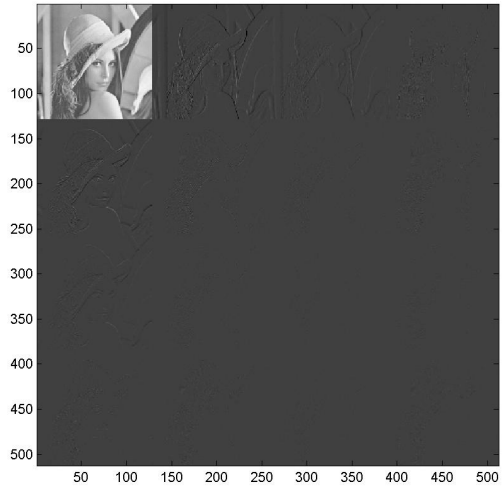


FIG. 19 – Décomposition au niveau 7

La décomposition en ondelette dans la meilleure base pour Lena correspond à la figure 20.

Cela correspond à la décomposition suivante :

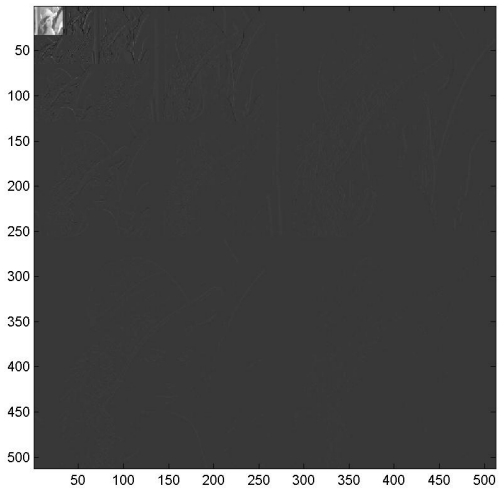


FIG. 20 – Décomposition dans la meilleure base