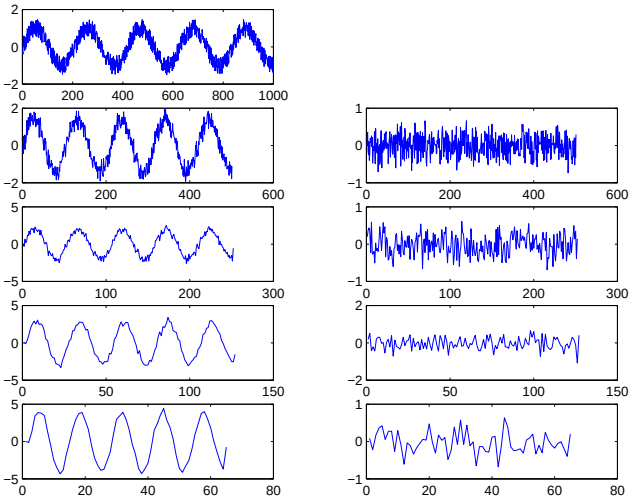
Boîte à outils ondelettes sous Matlab

J. Landré et F. Truchetet
Le2i - Université de Bourgogne
Formation C.R.I.

Mars 2002



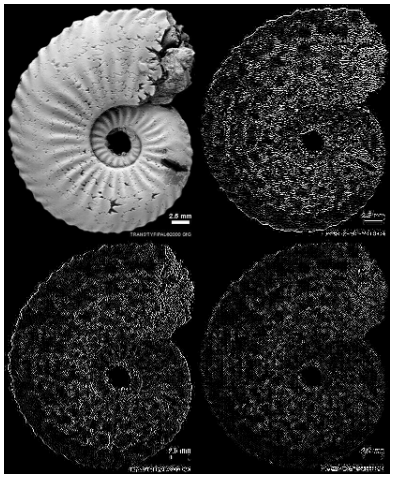


Table des matières

1 Introduction

1.1 Le traitement du signal

En traitement du signal, on étudie l'information portée par un signal. La transformée de Fourier (TF) permet de transformer un signal en une somme (qui peut être infinie) de série de fonctions sinus et cosinus.

$$\hat{f}(\omega) = \int_{-\infty}^{+\infty} f(t)e^{-i\omega t}dt$$

Cette transformation réversible permet de passer du domaine temporel au domaine fréquentiel (et inversement) afin de mettre en évidence les propriétés du signal. De nombreuses applications utilisent les résultats et les propriétés de la TF.
La caractéristique primordiale de cette transformée est sa réversibilité, ainsi on a :

$$f(t) = \int_{-\infty}^{+\infty} \hat{f}(\omega)e^{i\omega t}d\omega$$

Le problème de la TF est son manque de résolution temporelle. Cela signifie simplement que si on est effectivement capable de connaître toutes les fréquences présentes dans un signal, on est en revanche incapable de dire à quel moment elles se produisent dans le signal.
Un exemple de transformée de Fourier d'une image est donné en figure 1.

Exemple 0 : Transformation de Fourier discrète d'un signal 2d.

```
>> load wbarb;
>> sig=X;
>> figure;plot(sig);print -deps img
>> ff=fft(sig);
>> figure;plot(real(ff));
>> figure;plot(imag(ff));
>> figure;plot(abs(ff));
>> figure;plot(angle(ff));
>> help cwt
>> figure;c=cwt(sig,1:80,'db4','lvl');print -depsc2 c
>> figure;mesh(c);colormap(hot);print -depsc2 c3d
```

Pour pallier au problème de manque de résolution, Gabor a introduit la TF à fenêtre glissante. Dans cette méthode, on considère le signal à analyser comme appartenant à une fenêtre de longueur fixe qui glisse sur le signal pendant la transformation.

$$\hat{\hat{f}}(s,\omega) = \int_{-\infty}^{+\infty} f(t)g(t-s)e^{-i\omega t}dt$$

Cette approche permet d'avoir une meilleure résolution que la TF, mais la taille fixe de la fenêtre est un gros inconvénient. L'outil idéal serait une fenêtre qui s'adapte aux variations de fréquence dans le signal à analyser.

1.2 La transformée en ondelettes

La théorie des ondelettes est apparue il y a quelques années suite aux travaux de Morlet et Grossman [1] sur la caractérisation de signaux sismiques. Depuis, les travaux de recherche sur les ondelettes se sont multipliés sous l'influence de chercheurs

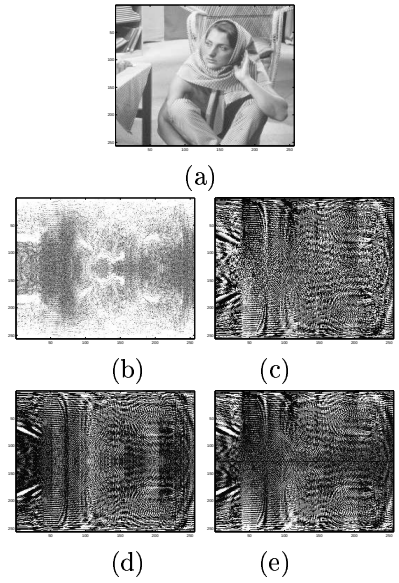


FIG. 1 – (a) Image originale (b) module (c) argument (d) partie réelle et (e) partie imaginaire de la transformée de Fourier de l'image

français notamment [3]. De nombreuses voies nouvelles sont en cours d'exploration. Les applications des ondelettes touchent tous les domaines du traitement du signal.

La boîte à outils Matlab est française, elle a été réalisée par Michel Misiti, Yves Mitisi, Georges Oppenheim et Jean-Michel Poggi. Elle propose des transformations 1d et 2d, de nombreuses familles d'ondelettes, des algorithmes de calculs...

Le présent document a pour vocation de présenter les principales fonctions ondelettes de Matlab. La référence reste bien entendu la documentation officielle de la toolbox Wavelet par The Mathworks. Dans la suite, les principales fonctions 1d puis 2d seront décrites et utilisées dans le cadre de travaux pratiques.

NOTES

2 Les fonctions 1d

2.1 La transformée en ondelettes continue

La transformation en ondelettes continue est définie comme suit :

$$\tilde{f}(a,b) = \int_{-\infty}^{+\infty} f(t)\psi^*_{a,b}(t)dt$$

Dans cette équation, $\psi^*(t)$ est l'ensemble des fonctions de base appelées ondelettes. L'étoile * représente le complexe conjugué. La famille d'ondelettes est générée à partir de la dilatation et de la translation d'une ondelette mère $\psi(t)$. Les ondelettes sont donc définies par :

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}}\psi(\frac{t-b}{a})$$

L'équation ci-dessus devient :

$$\tilde{f}(a,b) = \frac{1}{\sqrt{a}}\int_{-\infty}^{+\infty} f(t)\psi^*(\frac{t-b}{a})dt$$

Où ψ désigne la fonction ondelette mère dont les dilatées et translatées sont les bases d'un espace d'analyse en ondelettes sur lequel est projetée la fonction $f(t)$.
La transformée en ondelettes est réversible, on peut donc passer de l'analyse d'une fonction à sa reconstruction par :

$$f(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \tilde{f}(a,b)\psi_{a,b}(t)\frac{da}{a^2}db$$

Remarquons que contrairement à la transformée de Fourier, la fonction ψ n'est pas imposée. Par contre, ψ doit posséder certaines caractéristiques pour que la transformée en ondelettes soit possible. Toute fonction qui vérifie la condition d'admissibilité ci-dessous est une ondelette potentielle :

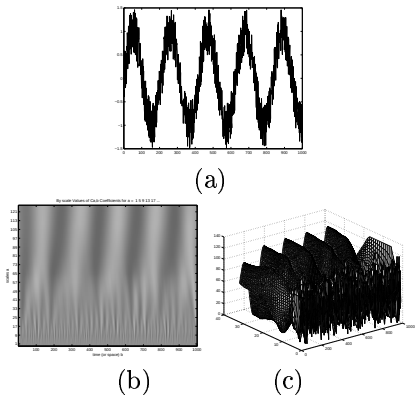
$$\int_{-\infty}^{+\infty} \frac{|\hat{\psi}(\omega)|^2}{|\omega|}d\omega < +\infty$$

C'est-à-dire que l'ondelette est à énergie finie. De plus, on a $|\hat{\psi}(0)|^2 = 0$, autrement dit la fonction ψ est un filtre passe-bande. Cela implique que $\int_{-\infty}^{+\infty}\psi(t)dt = 0$ donc ψ est de moyenne nulle, elle doit osciller, ψ est une onde.

Exemple 1 : Transformation en ondelettes continue 1d.

```
>> load noissin;
>> whos
>> sig=noissin;
>> figure;plot(sig);print -depsc2 sig
>> ff=fft(sig);
>> figure;plot(real(ff));
>> figure;plot(imag(ff));
>> figure;plot(abs(ff));
>> figure;plot(angle(ff));
>> help cwt
>> figure;c=cwt(sig,1:80,'db4','lvl');print -depsc2 c
>> figure;mesh(c);colormap(hot);print -depsc2 c3d
```

On charge le fichier signal à analyser : *load noissin.mat*. On regarde ce qu'il contient grâce à whos. On appelle notre signal *sig*. On affiche le signal. On calcule la transformée de Fourier du signal *ff*, on calcule *ffr* et *ffi* qui représentent la partie réelle et imaginaire de *ff*, *mod* et *arg* sont respectivement le module et l'argument de *ff*. On obtient le résultat présenté en figure 3.



(a) (b) (c)

FIG. 2 – (a) Signal original $f(t)$ (b) transformée en ondelettes continue de $f(t)$ (c) Vue 3d de la transformée en ondelettes continue.

Exercice 1 : Transformation en ondelettes continue 1d.

Dans cet exercice, il est demandé d'effectuer la série de transformation de l'exemple 1 sur le signal nommé "leleccum.mat".

2.2 La transformée en ondelettes discrète

La transformée en ondelettes continue est très redondante. Afin d'appliquer la transformée en ondelettes aux signaux discrets, il convient de discrétiser la fonction ψ en utilisant des valeurs prises sur une grille discrète.
On impose donc une grille de valeurs discrètes prises par la fonction ψ. On pose $a = a_0^m$ et $b = nb_0a_0^m$ avec $a_0 \in \mathbb{Z}$ et $b_0 \in \mathbb{Z}$.
La transformée en ondelettes discrète est donnée par :

$$\tilde{f}(m,n) = a_0^{-\frac{m}{2}} \int_{-\infty}^{+\infty} f(t)\psi(a_0^{-m}t - nb_0)dt$$

Si on choisit $a_0 = 2$ et $b_0 = 0$, on se place dans le cas dyadique. On a alors :

$$\tilde{f}(m,n) = 2^{-\frac{m}{2}} \int_{-\infty}^{+\infty} f(t)\psi(2^{-m}t)dt$$

L'exemple 2 montre l'analyse en ondelettes et la reconstruction d'un signal sinusoïdal bruité.

Exemple 2 : Transformation en ondelettes discrète 1d à un seul niveau et transformation inverse.

```
>> load noissin;
>> whos
>> sig=noissin;
>> longsig=length(sig);
>> figure;plot(sig);print -depsc2 ex2-1
>> [a,d]=dwt(sig,'db2');
>> figure;plot(a);print -depsc2 ex2-2
>> figure;plot(d);print -depsc2 ex2-3
>> a1=upcoef('a',a,'db2',1,longsig);
>> d1=upcoef('d',d,'db2',1,longsig);
>> figure;plot(a1);print -depsc2 ex2-4
>> figure;plot(d1);print -depsc2 ex2-5
>> a0=idwt(a,d,'db2',longsig);
>> figure;plot(a0);print -depsc2 ex2-6
>> erreur=sig-a0;
>> figure;plot(erreur);print -depsc2 ex2-7
```

Dans cet exemple, le signal *noissin* est chargé dans Matlab, sa taille est stockée dans *longsig*. La transformée en ondelettes discrète est réalisée grâce à la fonction *dwt*. L'approximation et les détails sont affichés à l'échelle de l'analyse. Puis, on reconstruit l'approximation et les détails à l'échelle de l'image originale avec *upcoef*. Les figures sont à nouveau affichées. Enfin, le signal de départ est reconstruit à l'aide de la fonction *idwt*. L'erreur entre la fonction de départ et la fonction reconstruite est évaluée et est négligeable.

Exercice 2 : Transformation en ondelettes discrète à un seul niveau.

Dans cet exercice, il est demandé d'effectuer la série de transformation de l'exemple 2 sur le signal nommé "leleccum.mat".

Afin de réaliser une transformée en ondelettes à plusieurs niveaux de résolution, soit une écrit une fonction qui itère le calcul plusieurs fois, soit on utilise la fonction de calcul à plusieurs résolutions *wavedec* fournie par la boîte à outils.

Exemple 3 : Transformation en ondelettes discrète 1d à plusieurs niveaux.

```
>> load noissin;
>> whos
>> sig=noissin;
>> longsig=length(sig);
>> [c,l]=wavedec(sig,4,'db2');
>> ap4=appcoef(c,l,'db2',4);
>> ap3=appcoef(c,l,'db2',3);
>> ap2=appcoef(c,l,'db2',2);
>> ap1=appcoef(c,l,'db2',1);
>> d4=detcoef(c,l,4);
>> d3=detcoef(c,l,3);
>> d2=detcoef(c,l,2);
>> d1=detcoef(c,l,1);
>> figure;
>> subplot(5,2,1);plot(sig);
```

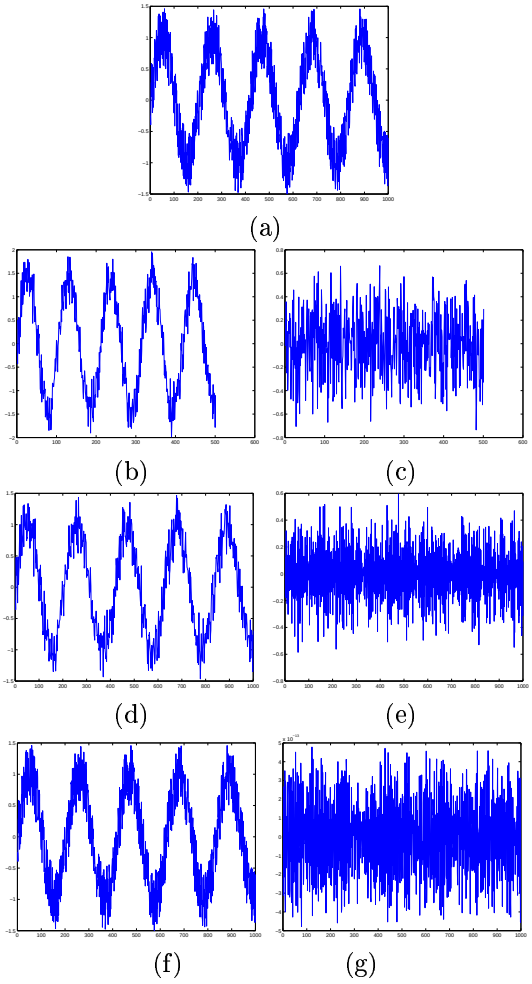


FIG. 3 – (a) Signal original $f(t)$ (b) approximation et (c) détails de la transformée en ondelettes de niveau 1 (d) approximation et (e) détails reconstruits au niveau 0 (f) signal reconstruit à partir de la transformée en ondelettes inverse (g) erreur commise lors de la reconstruction.

```
>> subplot(5,2,3);plot(ap1);
>> subplot(5,2,4);plot(d1);
>> subplot(5,2,5);plot(ap2);
>> subplot(5,2,6);plot(d2);
>> subplot(5,2,7);plot(ap3);
>> subplot(5,2,8);plot(d3);
>> subplot(5,2,9);plot(ap4);
>> subplot(5,2,10);plot(d4);
>> print -depsc2 ex3-1
>> recap3=wrcoef('a',c,l,'db2',3);
>> recd1=wrcoef('d',c,l,'db2',1);
>> recd2=wrcoef('d',c,l,'db2',2);
>> recd3=wrcoef('d',c,l,'db2',3);
>> figure;
>> subplot(4,1,1);plot(recap3);title('Approximation a3');
>> subplot(4,1,2);plot(recd1);title('details d1');
>> subplot(4,1,3);plot(recd2);title('details d2');
>> subplot(4,1,4);plot(recd3);title('details d3');
>> print -depsc2 ex3-2
>> orig=waverec(c,l,'db2');
```

```
>> figure;plot(orig);print -depsc2 orig
>> figure;plot(orig-sig);print -depsc2 ex3-erreur
```

Le résultat de l'exemple 3 est donné en figure 4.

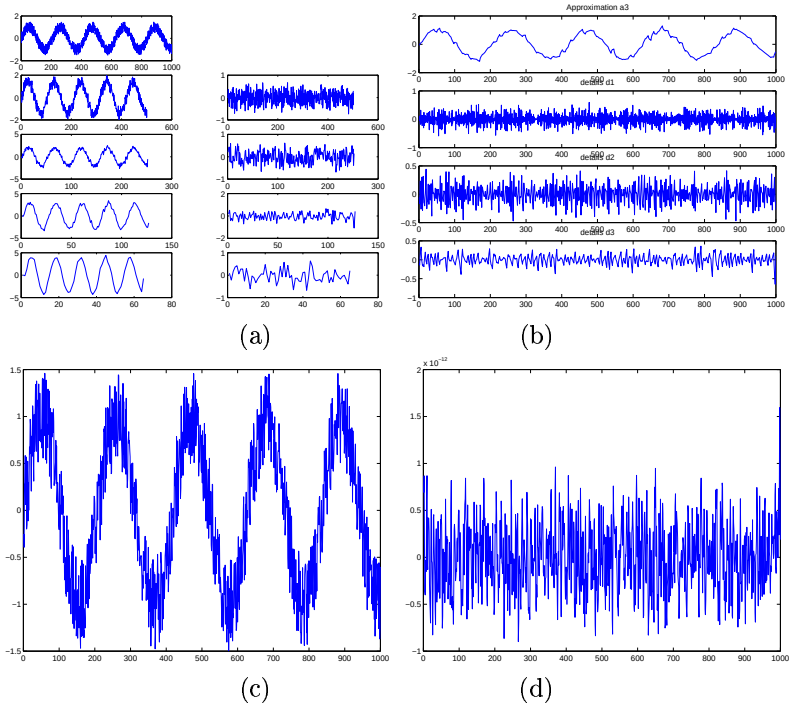


FIG. 4 – (a) transformée en ondelettes à plusieurs niveaux de résolution (b) approximation et détails (c) signal reconstruit (d) erreur par rapport au signal d'origine.

Exercice 3 : Transformée en ondelettes à plusieurs niveaux de résolution.

Dans cet exercice, il est demandé d'effectuer la transformation de l'exemple 3 sur le signal nommé "leleccum.mat".

NOTES

3 Les fonctions 2d

3.1 Transformer les images

Avant de transformer des images en ondelettes, il est utile de rappeler quelques fonctions Matlab bien utiles à travers un exemple.

Exemple 4 : Lecture d'une image au format jpeg.

```
>> im=imread('lena.jpg');
>> im=im2double(rgb2gray(im));
>> whos
  Name      Size          Bytes  Class

  im      512x512x3    6291456  double array

Grand total is 786432 elements using 6291456 bytes
>> figure;imshow(im);axis('image');axis on
>> print -depsc2 lena.eps
```

On lit tout d'abord l'image JPEG "lena.jpg" et on la stocke dans la variable "im". On passe de l'image "im" codée en entiers en une image codée en réels. On passe d'un espace couleur à un espace niveaux de gris. On affiche l'image avec une échelle en pixels. Enfin on sauve l'image affichée dans le répertoire courant au format postscript encapsulé (EPS) sous le nom "lena.eps". Le résultat est donné en figure 5.

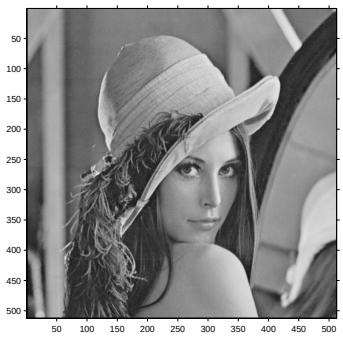


FIG. 5 – Exemple de lecture de l'image lena.jpg.

3.2 Analyse multirésolution

Soit un ensemble de sous-espaces de $L^2(\mathbb{R})$ (l'ensemble des signaux à énergie finie) tels que :

$\ldots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset \ldots \subset V_{j+1} \subset V_j \subset \ldots$
$\overline{\cup_{j\in\mathbb{Z}} V_j} = L^2(\mathbb{R})$
$\cap_{j\in\mathbb{Z}} V_j = 0$
$\forall j \in \mathbb{Z}, f(x) \in V_j \iff f(2^{-1}x) \in V_{j+1}$
$\forall k \in \mathbb{Z}, f(x) \in V_0 \iff f(x-k) \in V_0$

Ces propriétés définissent une analyse multirésolution sur $L^2(\mathbb{R})$.

L'analyse multirésolution a été définie par Mallat [3]. L'idée est de projeter un signal $f(t) \in L^2(\mathbb{R})$ appartenant à un espace V_j sur un sous-espace V_{j+1} et un sous-espace W_{j+1} dans le but de réduire la résolution de moitié. Le schéma est donné en figure 6. Il existe donc un opérateur de projection A_j et un opérateur de projection D_j qui projettent respectivement le signal $f(t)$ sur V_{j+1} et W_{j+1}. V_{j+1} est le sous-espace d'approximation et W_{j+1} le sous-espace de détails. On peut démontrer qu'il existe une fonction d'échelle $\phi(t) \in L^2(\mathbb{R})$ qui engendre par dilatation et translation une base orthonormée de V_{j+1} et une fonction d'ondelettes $\psi(t) \in L^2(\mathbb{R})$ qui engendre par dilatation et translation une base orthonormée de W_{j+1}. Les espaces obtenus ne sont pas quelconques, ils possèdent des propriétés intéressantes. Par construction, les espaces d'approximation V_{j+1} et de détails W_{j+1} sont supplémentaires : $V_j = V_{j+1} \bigoplus W_{j+1}$.

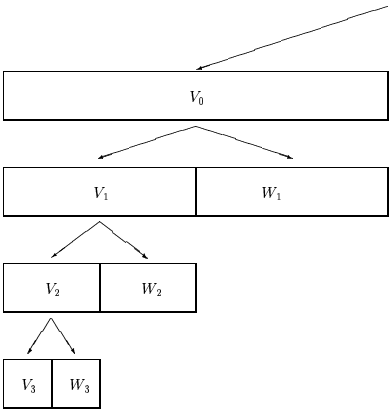


FIG. 6 – Principe de l'analyse multirésolution.

Les fonctions de bases dilatées sont données par les relations :
$\phi_{j,n}(t) = 2^{-j/2}\phi(2^{-j}t - n)$ avec $n \in \mathbb{Z}$ et $\psi_{j,n}(t) = 2^{-j/2}\psi(2^{-j}t - n)$ avec $n \in \mathbb{Z}$
On a donc $A_j f = \sum_n < f, \phi_{j,n} > \phi_{j,n}$ et $D_j f = \sum_n < f, \psi_{j,n} > \psi_{j,n}$ où $< f(t),\ g(t) >$ désigne le produit scalaire de $f(t)$ par $g(t)$: $< f(t),\ g(t) >= \int_{-\infty}^{+\infty} f(t)\ \overline{g(t)} dt$
Puisque les fonctions utilisées appartiennent toutes à $L^2(\mathbb{R})$, on a $\overline{g(t)} = g(t)$. On pose $a_{j,n} =< f, \phi_{j,n} >$ et $d_{j,n} =< f, \psi_{j,n} >$. $a_{j,n}$ et $d_{j,n}$ sont respectivement les coefficients d'approximation et de détails de la transformée en ondelettes de la fonction f.
Le résultat de l'analyse multirésolution d'une image est donné en figure 8. On voit la diminution de la résolution, l'image d'approximation et les images de détails horizontaux, verticaux et diagonaux.

3.3 Algorithme d'analyse de Mallat

Stéphane Mallat a donné un algorithme de décompositon en ondelettes qui permet d'obtenir une analyse multirésolution du signal. Cet algorithme travaille par filtrage de l'image suivant les lignes puis les colonnes par deux filtres, $\tilde{g}$ passe-haut et $\tilde{h}$ passe-bas. $\tilde{h}$ va donc s'attarder sur les basses fréquences dans l'image (l'approximation) et $\tilde{g}$ sur les hautes fréquences (les détails).

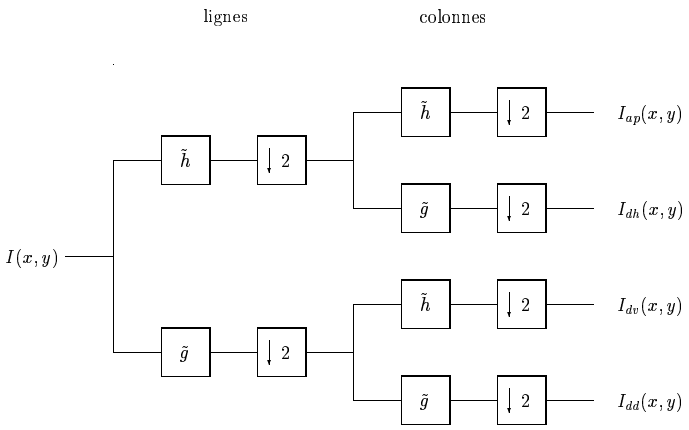


FIG. 7 – Algorithme d'analyse de Mallat.

L'exemple suivant illustre l'analyse multirésolution d'une image en niveaux de gris selon l'algorithme de Mallat.

Exemple 5 : Analyse multirésolution d'une image en niveaux de gris à un niveau de résolution et reconstruction.

```
>> clear;
>> img=imread('lena.jpg');
>> img=im2double(rgb2gray(img));
>> colormap(gray);
>> figure;imshow(img);axis('image');axis on;print -depsc2 ex5-1
>> [ap,dh,dv,dd]=dwt2(img,'haar');
>> figure;imshow([ap,dh;dv,dd]);axis('image');axis on;print -depsc2 ex5-2
>> figure;mesh([ap,dh;dv,dd]);print -depsc2 ex5-3
>> figure;imshow([ap/2,dh*16;dv*16,dd*16]);axis('image');axis on;print -depsc2 ex5-4
>> whos
  Name      Size          Bytes  Class

  ap      256x256       524288  double array
  dd      256x256       524288  double array
  dh      256x256       524288  double array
  dv      256x256       524288  double array
  img     512x512      2097152  double array

Grand total is 524288 elements using 4194304 bytes
>> ap1=upcoef2('a',ap,'haar',1);
>> dh1=upcoef2('d',dh,'haar',1);
>> dv1=upcoef2('d',dv,'haar',1);
>> dd1=upcoef2('d',dd,'haar',1);
>> figure;colormap(gray);
>> subplot(2,2,1);image(wcodemat(ap1,192));
>> subplot(2,2,2);image(wcodemat(dh1,192));
>> subplot(2,2,3);image(wcodemat(dv1,192));
>> subplot(2,2,4);image(wcodemat(dd1,192));
>> print -depsc2 ex5-5
>> orig=idwt2(ap,dh,dv,dd,'haar');
>> figure;imagesc(orig);colormap(gray);axis('image');axis on;
>> figure;imagesc(orig-img);print -deps ex5-erreur
>> max(max(orig-img))

ans =

  6.6613e-016
```

FIG. 8 – Un exemple de décomposition en ondelettes d'une image au premier niveau de résolution.

Exercice 5 : Analyse multirésolution et reconstruction d'une image.

Effectuer l'analyse multirésolution de l'image contenue dans "wbarb.mat". A l'aide des symlets (5), décomposer et afficher l'image de départ puis reconstruire á partir de la décomposition, afficher et comparer les résultats et conclure.

3.4 Algorithme de reconstruction de Mallat

La reconstruction des signaux analysés est effectuée à l'aide d'un banc de filtres h et g qui sont les filtres conjugués de $\tilde{h}$ et $\tilde{g}$. Selon l'ondelette choisie pour l'analyse, les filtres d'analyse et de reconstruction peuvent être de même taille, symétriques ou bien de taille différente, non symétriques.

Dans la reconstruction, on travaille alternativement sur les colonnes puis sur les lignes lorsque les ondelettes sont séparables.

Exemple 6 : Analyse multirésolution d'une image couleur à plusieurs niveaux de résolution et reconstruction.

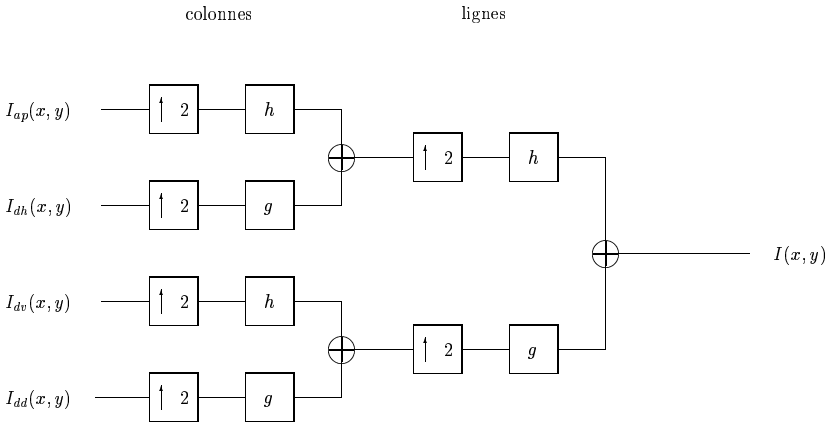


FIG. 9 – Algorithme de reconstruction de Mallat.

```
>> clear
>> load wbarb
>> whos
  Name      Size          Bytes  Class

  X       256x256       524288  double array
  map     192x3           4608  double array

Grand total is 66112 elements using 528896 bytes

>> figure;image(X);colormap(map);
>> print -depsc2 ex6-1.eps
>> [c,s]=wavedec2(X,3,'coif2');
>> a3=appcoef2(c,s,'coif2',3);
>> dh3=detcoef2('h',c,s,3);
>> dv3=detcoef2('v',c,s,3);
>> dd3=detcoef2('d',c,s,3);
>> a2=appcoef2(c,s,'coif2',2);
>> dh2=detcoef2('h',c,s,2);
>> dv2=detcoef2('v',c,s,2);
>> dd2=detcoef2('d',c,s,2);
>> a1=appcoef2(c,s,'coif2',1);
>> dh1=detcoef2('h',c,s,1);
>> dv1=detcoef2('v',c,s,1);
>> dd1=detcoef2('d',c,s,1);
>> figure;colormap(map);
>> subplot(2,2,1);image(X);
>> subplot(2,2,2);image(a1/2);
>> subplot(2,2,3);image(a2/4);
>> subplot(2,2,4);image(a3/8);
>> print -depsc2 ex6-2.eps
>> img3=[a3/8,dh3*2;dv3*2,dd3*2];
>> img2=[a2/4,dh2*2;dv2*2,dd2*2];
>> img1=[a1/2,dh1*2;dv1*2,dd1*2];
>> figure;colormap(map);image(img3);print -depsc2 ex6-3.eps
>> figure;colormap(map);image(img2);print -depsc2 ex6-4.eps
>> figure;colormap(map);image(img1);print -depsc2 ex6-5.eps
>> ar2=wrcoef2('a',c,s,'coif2',2);
>> dhr2=wrcoef2('h',c,s,'coif2',2);
>> figure;image(ar2);colormap(map);print -depsc2 ex6-6.eps
>> figure;image(dhr2*16);colormap(map);print -depsc2 ex6-7.eps
>> \% reconstruction
>> Xrec=waverec2(c,s,'coif2');
>> figure;image(Xrec);colormap(map);print -depsc2 ex6-8.eps
```

FIG. 10 – Décompositon et reconstruction à plusieurs niveaux de résolution.

Exercice 6 : Analyse multirésolution et reconstruction d'une image à plusieurs niveaux de résolution.

Effectuer l'analyse et la reconstruction de l'image "gatlin.mat" à l'aide des daublets (3) à trois niveaux de résolution, afficher la décomposition et reconstruire, comparer le signal original et la reconstruction, conclure.

La boîte à outils permet de choisir la manière selon laquelle l'image sera complétée pendant la transformation afin de limiter les effets de bords en utilisant la fonction *dwtmode*. Il existe quatre méthodes d'extensions que nous allons détailler maintenant.
Extension à zéro :

Exemple 7 : Extension d'une image avec des zéros.

```
>> dwtmode('zpd');
>> [c,s]=wavedec2(X,3,'sym4');
>> a=wrcoef2('a',c,s,'sym4',3);
>> image(wcodemat(a,size(map,1)));
```

Extension symétrique :

Exemple 8 : Extension d'une image par symétrie.

```
>> dwtmode('sym');
>> [c,s]=wavedec2(X,3,'sym4');
>> a=wrcoef2('a',c,s,'sym4',3);
>> image(wcodemat(a,size(map,1)));
```

Extension progressive :

Exemple 9 : Extension progressive d'une image.

```
>> dwtmode('spd');
>> [c,s]=wavedec2(X,3,'sym4');
>> a=wrcoef2('a',c,s,'sym4',3);
>> image(wcodemat(a,size(map,1)));
```

Extension périodique de l'image :

Exemple 10 : Péridisation d'une image.

Il existe dans la boîte à outils une extension périodisée de la transformation en utilisant les fonctions dwtper, dwtper2, idwtper et idwtper2. Le lecteur courageux sera en mesure de les tester par lui-même.

NOTES

4 Paquets d'ondelettes

Le principe de la transformation en paquets d'ondelettes est donné en figure **??**. La décomposition en paquets d'ondelettes est la généralisation de la décomposition multirésolution. A chaque niveau de résolution, l'approximation et les détails sont décomposés. La transformation en paquets d'ondelettes fournit 2^n façon de décomposer un signal alors que l'analyse en ondelettes n'en donnait que $n + 1$. Cette transformation est redondante mais offre le choix de la représentation du signal selon un critère d'entropie donné.

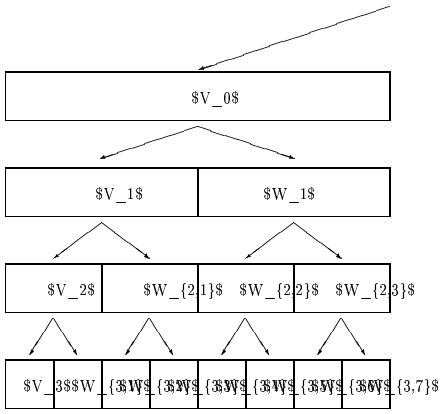


FIG. 11 – Principe de l'analyse en paquets d'ondelettes.

Il existe des fonctions de transformation en paquets d'ondelettes qui permettent d'effectuer les calculs manuellement. Cependant, il est préférable pour des raisons de facilité d'utiliser les fonctions d'analyse graphiques.

Exemple 11 : Utilisation de l'interface graphique d'analyse en paquets d'ondelettes.

```
>> wavemenu
```

NOTES

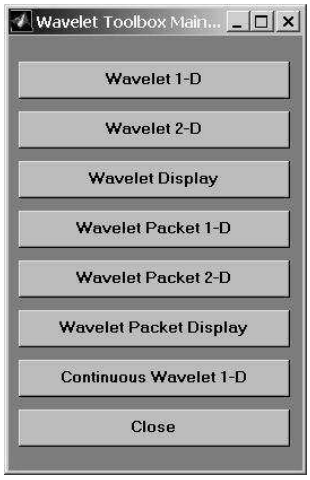


FIG. 12 – Le menu de l'outil graphique ondelettes.

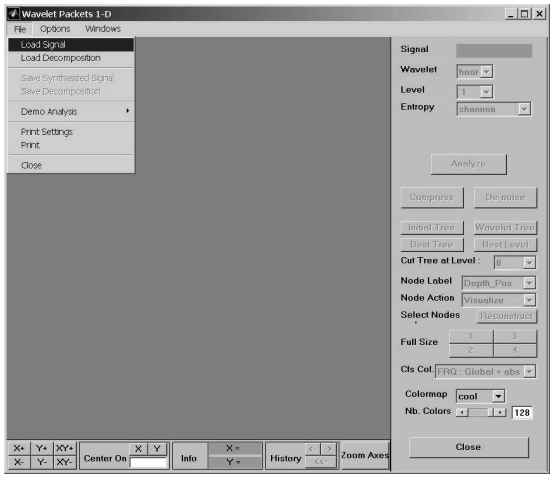


FIG. 13 – Les paquets d'ondelettes 1d - chargement d'un signal.

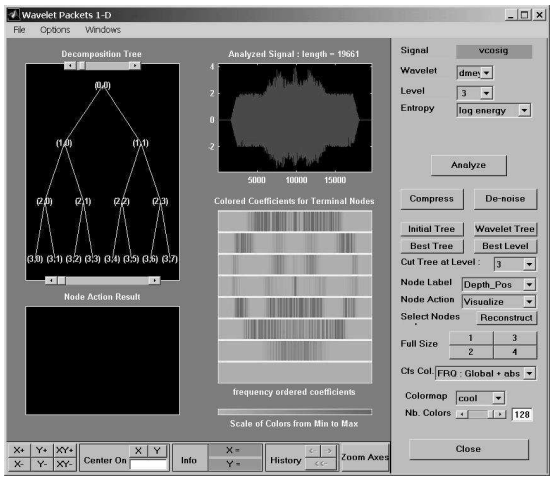


FIG. 14 – Les paquets d'ondelettes 1d - en cours d'analyse.

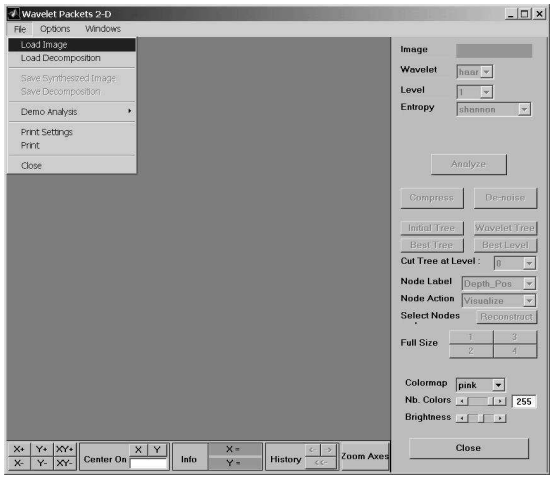


FIG. 15 – Les paquets d'ondelettes 2d - chargement d'une image.

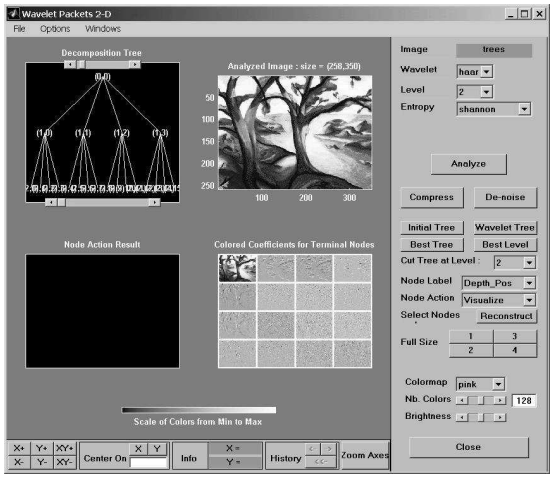


FIG. 16 – Les paquets d'ondelettes 2d - en cours d'analyse .

5 Quelles ondelettes ?

Toute fonction qui satisfait la condition d'admissibilité peut être choisie pour la transformée en ondelettes. Ainsi chacun est capable de créer ses propres ondelettes en fonction du problème à traiter.
La boîte à outils wavelets fournit un ensemble de fonctions d'ondelettes classiques qui doivent permettre de résoudre un grand nombre de problèmes.

Exemple : Familles d'ondelettes de Matlab.

```
>> wavemngr('read',1);

ans =

===================================
Haar            haar
===================================
Daubechies      db
-------------------------------
db1 db2 db3 db4 db5 db6 db7 db8 db9 db10    db**
===================================
Symlets         sym
-------------------------------
sym2    sym3    sym4    sym5 sym6    sym7    sym8    sym**
===================================
Coiflets        coif
-------------------------------
coif1   coif2   coif3   coif4 coif5
===================================
BiorSplines     bior
-------------------------------
bior1.1 bior1.3 bior1.5 bior2.2 bior2.4 bior2.6 bior2.8 bior3.1
bior3.3 bior3.5 bior3.7 bior3.9 bior4.4 bior5.5 bior6.8
===================================
ReverseBior     rbio
-------------------------------
rbio1.1 rbio1.3 rbio1.5 rbio2.2 rbio2.4 rbio2.6 rbio2.8 rbio3.1
rbio3.3 rbio3.5 rbio3.7 rbio3.9 rbio4.4 rbio5.5 rbio6.8
===================================
Meyer           meyr
===================================
DMeyer          dmey
===================================
Gaussian        gaus
-------------------------------
gaus1   gaus2   gaus3   gaus4 gaus5   gaus6   gaus7   gaus8 gaus**
===================================
Mexican_hat     mexh
===================================
Morlet          morl
===================================
```

Voici la liste des ondelettes présentes dans la boîte à outils :
- Les ondelettes de Haar 'haar',
- Les ondelettes de Daubechies 'dbX' $X \in [1, 10]$,
- Les symlets 'symX' $X \in [2, 8]$,
- Les Coiflets 'coifX' $X \in [1, 5]$,
- Les splines biorthogonales 'bior'
  $X \in \{1.1, 1.3, 1.5, 2.2, 2.4, 2.6, 2.8, 3.1, 3.3, 3.5, 3.7, 3.9, 4.4, 5.5, 6.8\}$,

– Les splines biorthogonales inverses 'rbio'
  $X \in \{1.1, 1.3, 1.5, 2.2, 2.4, 2.6, 2.8, 3.1, 3.3, 3.5, 3.7, 3.9, 4.4, 5.5, 6.8\}$,
– Les ondelettes de Meyer 'meyr' et 'dmey',
– Les gaussiennes 'gaussX' $X \in [1, 8]$,
– Le chapeau mexicain 'mexh',
– les ondelettes de Morlet 'morl'.

Pour obtenir des informations plus précises sur une famille d'ondelettes, il faut utiliser la fonction *waveinfo* comme indiqué dans l'exemple 8.

Exemple 8 : Informations sur une famille d'ondelettes de Matlab.

```
>> waveinfo('sym');

 SYMINFO Information on near symmetric wavelets.

    Symlets Wavelets

    General characteristics: Compactly supported wavelets with
    least assymetry and highest number of vanishing moments
    for a given support width.
    Associated scaling filters are near linear-phase filters.

    Family                  Symlets
    Short name              sym
    Order N                 N = 2, 3, ...
    Examples                sym2, sym8

    Orthogonal              yes
    Biorthogonal            yes
    Compact support         yes
    DWT                     possible
    CWT                     possible

    Support width           2N-1
    Filters length          2N
    Regularity
    Symmetry                near from
    Number of vanishing
    moments for psi         N

    Reference: I. Daubechies,
    Ten lectures on wavelets,
    CBMS, SIAM, 61, 1994, 194-202.
```

Les différentes fonctions de la toolbox sont interrogeables par l'aide.

Exemple 9 : L'aide de Matlab est une excellente source d'informations.

```
>> help wavelet

  Wavelet Toolbox.
  Version 1.2   (R11) 10-Jul-1998

  What's new.
    Readme      - New features, bug fixes, and changes in this version.
                  To display the Readme file for Version 1.2, type
                  "whatsnew wavelet" in the MATLAB command window.
```

```
Wavelet Toolbox GUI (Graphical User Interface).
  wavemenu    - Launching point for the interactive Wavelet GUIs.

Wavelets General.
  biorfilt    - Biorthogonal wavelet filter set.
  dyaddown    - Dyadic downsampling.
  dyadup      - Dyadic upsampling.
  intwave     - Integrate wavelet function psi.
  orthfilt    - Orthogonal wavelet filter set.
  qmf         - Quadrature mirror filter.
  wavefun     - Wavelet and scaling functions.
  wavemngr    - Wavelet manager.
  wfilters    - Wavelet filters.
  wmaxlev     - Maximum wavelet decomposition level.

Wavelet Families.
  biorwavf    - Biorthogonal spline wavelet filters.
  coifwavf    - Coiflets wavelet filters.
  dbaux       - Daubechies wavelet filters computation.
  dbwavf      - Daubechies wavelet filters.
  gauswavf    - Gaussian wavelets.
  mexihat     - Mexican hat wavelet.
  meyer       - Meyer wavelet.
  meyeraux    - Meyer wavelet auxiliary function.
  morlet      - Morlet wavelet.
  rbiowavf    - Reverse Biorthogonal spline wavelet filters.
  symaux      - Symlets wavelet filters computation.
  symwavf     - Symlets wavelet filters.

Continuous Wavelet: 1-D.
  cwt         - Continuous wavelet coefficients 1-D.

Discrete Wavelets: 1-D.
  appcoef     - Extract 1-D approximation coefficients.
  detcoef     - Extract 1-D detail coefficients.
  dwt         - Single-level discrete 1-D wavelet transform.
  dwtper      - Single-level discrete 1-D wavelet transform (periodized).
  dwtmode     - Discrete wavelet transform extension mode.
  idwt        - Single-level inverse discrete 1-D wavelet transform.
  idwtper     - Single-level inv. discrete 1-D wavelet transform (periodized).
  upcoef      - Direct reconstruction from 1-D wavelet coefficients.
  upwlev      - Single-level reconstruction of 1-D wavelet decomposition.
  wavedec     - Multi-level 1-D wavelet decomposition.
  waverec     - Multi-level 1-D wavelet reconstruction.
  wrcoef      - Reconstruct single branch from 1-D wavelet coefficients.

Discrete Wavelets: 2-D.
  appcoef2    - Extract 2-D approximation coefficients.
  detcoef2    - Extract 2-D detail coefficients.
  dwt2        - Single-level discrete 2-D wavelet transform.
  dwtper2     - Single-level discrete 2-D wavelet transform (periodized).
  dwtmode     - Discrete wavelet transform extension mode.
  idwt2       - Single-level inverse discrete 2-D wavelet transform.
  idwtper2    - Single-level inv. discrete 2-D wavelet transform (periodized).
  upcoef2     - Direct reconstruction from 2-D wavelet coefficients.
  upwlev2     - Single-level reconstruction of 2-D wavelet decomposition.
  wavedec2    - Multi-level 2-D wavelet decomposition.
  waverec2    - Multi-level 2-D wavelet reconstruction.
  wrcoef2     - Reconstruct single branch from 2-D wavelet coefficients.

Wavelets Packets Algorithms.
  besttree    - Best tree (wavelet packet).
  bestlevt    - Best level tree (wavelet packet).
  entrupd     - Entropy update (wavelet packet).
  wentropy    - Entropy(wavelet packet).
  wp2wtree    - Extract wavelet tree from wavelet packet tree.
```

```
  wpcoef      - Wavelet packet coefficients.
  wpcutree    - Cut wavelet packet tree.
  wpdec       - Wavelet packet decomposition 1-D.
  wpdec2      - Wavelet packet decomposition 2-D.
  wpfun       - Wavelet packet functions.
  wpjoin      - Recompose wavelet packet.
  wprcoef     - Reconstruct wavelet packet coefficients.
  wprec       - Wavelet packet reconstruction 1-D.
  wprec2      - Wavelet packet reconstruction 2-D.
  wpsplt      - Split (decompose) wavelet packet.

De-noising and Compression for Signals and Images.
  ddencmp     - Default values for de-noising or compression.
  thselect    - Threshold selection for de-noising.
  wdcbm       - Wavelet 1-D deno. and comp. using Birge-Massart strategy.
  wdcbm2      - Wavelet 2-D deno. and comp. using Birge-Massart strategy.
  wden        - Automatic 1-D de-noising using wavelets.
  wdencmp     - De-noising or compression using wavelets.
  wnoise      - Generate noisy wavelet test data.
  wnoisest    - Estimate noise of 1-D wavelet coefficients.
  wpdencmp    - De-noising or compression using wavelet packet.
  wpthcoef    - Wavelet packet coefficients thresholding.
  wthcoef     - Wavelet coefficients thresholding 1-D.
  wthcoef2    - Wavelet coefficients thresholding 2-D.
  wthresh     - Perform soft or hard thresholding.

Tree Management Utilities.
  allnodes    - Tree nodes.
  depo2ind    - Node depth-position to node index.
  ind2depo    - Node index to node depth-position.
  isnode      - True for existing node.
  istnode     - True for terminal nodes.
  leaves      - Terminal nodes.
  maketree    - Make tree.
  nodeasc     - Node ascendants.
  nodedesc    - Node descendants.
  nodejoin    - Recompose node.
  nodepar     - Node parent.
  nodesplt    - Split (decompose) node.
  noleaves    - Not Terminal nodes.
  ntnode      - Number of terminal nodes.
  plottree    - Plot tree.
  tnodes      - Terminal nodes.
  treedpth    - Tree depth.
  treeord     - Tree order.
  wdatamgr    - Manager for data structure.
  wtreemgr    - Manager for tree structure.

Tree Utilities.
  chgwname    - Change the name of wavelet in a WP data structure.
  drawtree    - Draw Wavelet Packet Decomposition Tree (GUI).
  readtree    - Read Wavelet Packet Decomposition from a figure.

General Utilities.
  deblankl    - Convert string to lowercase without blanks.
  errargn     - Check function arguments number.
  errargt     - Check function arguments type.
  num2mstr    - Convert number to string in maximum precision.
  wcodemat    - Extended pseudocolor matrix scaling.
  wcommon     - Find common elements.
  wextend     - Extend a Vector or a Matrix.
  wkeep       - Keep part of a vector or a matrix.
  wrev        - Flip vector.

Other.
  instdfft    - Inverse non-standard 1-D fast Fourier transform.
  nstdfft     - Non-standard 1-D fast Fourier transform.
```

```
  Wavelets Information.
    waveinfo    - Information on wavelets.

  Demonstrations.
    wavedemo    - Wavelet Toolbox demos.

>> help dwt

 DWT Single-level discrete 1-D wavelet transform.
    DWT performs a single-level 1-D wavelet decomposition
    with respect to either a particular wavelet ('wname',
    see WFILTERS) or particular wavelet filters
    (Lo_D and Hi_D) you specify.

    [CA,CD] = DWT(X,'wname') computes the approximation
    coefficients vector CA and detail coefficients vector CD,
    obtained by a wavelet decomposition of the vector X.
    'wname' is a string containing the wavelet name.

    [CA,CD] = DWT(X,Lo_D,Hi_D) computes the wavelet decomposition
    as above given these filters as input:
    Lo_D is the decomposition low-pass filter and
    Hi_D is the decomposition high-pass filter.
    Lo_D and Hi_D must be the same length.

    If LX = length(X) and LF is the length of filters then
    length(CA) = length(CD) = LA, where LA = CEIL(LX/2)
    if the DWT extension mode is the periodized mode.
    LA = FLOOR((LX+LF-1)/2) for the other extension modes.
    For the different signal extension modes, see DWTMODE.

    [CA,CD] = DWT(X,'wname','mode',mode) or
    [CA,CD] = DWT(X,Lo_D,Hi_D,'mode',mode) computes the wavelet
    decomposition and the extension mode can be specified.

    See also DWTMODE, IDWT, WAVEDEC, WAVEINFO.
```

NOTES

6 Conclusion

La boîte à outils wavelets de matlab fournit un ensemble de fonctions pour le calcul des ondelettes 1d et 2d assez complet et efficace. Ce document a présenté les principales fonctions dans leur ensemble. La meilleure source de renseignement reste toutefois le manuel de l'utilisateur de la toolbox Wavelets [5].
En ce qui concerne la théorie des ondelettes, de nombreux ouvrages traitent de ce sujet [4] et de nombreux sites sur Internet [2] proposent des tutoriaux, des exemples d'application...
De nouvelles théories issues des ondelettes commencent à apparaître (multi-ondelettes, ondelettes rationnelles, bandelettes...) et les recherches sur le sujet avancent à vive allure.
Enfin, certains aspects des ondelettes n'ont pas été abordés dans ce manuel pour des raisons de place ou de complexité (ondelettes non séparables, quinconces, construction de familles d'ondelettes...), de très nombreux livres existent à ce sujet pour quelqu'un qui souhaite se plonger en profondeur dans la théorie.

Références

[1] A. Grossmann and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM J. Math. Anal.*, 15(4) :723–736, 1984.

[2] Mathsoft Inc. *http ://www.mathsoft.com/wavelets.html - Mathsoft wavelets home page.* Many applications of wavelets in many different domains.

[3] Stéphane G. Mallat. A theory for multiresolution signal decomposition : The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(7) :674–693, July 1989.

[4] Stéphane G. Mallat. *A Wavelet Tour of Signal Processing.* Academic Press, 1999.

[5] M. Misiti, Y. Misiti, G. Oppenheim, and J.-M. Poggi. *Wavelet Toolbox.* Users's guide version 1 - The Mathworks.