Lifting en ondelettes

Ronan Le Page *

juin-août 2004

Introduction

Le but de ce document est de synthétiser de façon concise les techniques et connaissances permettant la réalisation de la transformation en ondelettes version lifting, en particulier de réaliser la transformation d'entiers en coefficients d'ondelettes entiers. Les références essentielles sont les articles de Wim Sweldens (disponibles sur son site) et l'excellent tutoriel de Clemens Valens (http://perso.wanadoo.fr/polyvalens/clemens/wavelets/wavelets.html).

Transformée en ondelettes version *lifting*

La transformation en ondelettes version *lifting* est un processus permettant entre autres d'optimiser le nombre d'opérations à exécuter et l'occupation mémoire[1].

Le processus le plus courant pour obtenir une transformation en ondelettes est d'utiliser un banc de filtres (cf. Figure 1), cependant lorsqu'on regarde cette façon de procéder on constate que l'on effectue un sous-échantillonnage par deux après une opération de filtrage : on a donc dépenser en pure perte la moitié du coût de calcul effectué par le filtrage.

L'opération de Lifting en ondelettes peut-être vu comme la transformation réalisée par le banc de filtres, mais en intervertissant les phases de filtrage et de sous-échantillonnage. On limite ainsi le nombre d'opérations a effectuer mais, nous perdons en revanche la propriété d'invariance par translation.

Une autre propriété intéressante est que le schéma de lifting est facilement inversible.

Le schéma de lifting est aussi lié au processus d'interpolation (non explicitement étudié ici).

On désigne par γ les coefficients d'ondelettes et par λ les coefficients d'échelle.

Pour une ondelette particulière (i.e. un couple de filtres (h,g,$\tilde{h}$, $\tilde{g}$ pour l'implémentation par banc de filtres), caractérisé en outre par disons n moments nuls (jouant un rôle dans le processus de décroissance des coefficients d'ondelettes à travers les résolutions) pour le filtre primaire et $\tilde{n}$ moments nuls pour le filtre dual, le schéma d'implantation par lifting permet d'obtenir facilement des ondelettes de moments n et $\tilde{n}$ plus élevé. On a donc *lifté* (i.e.élevé) l'ordre de cette ondelette par ce schéma (d'où la justification du nom *lifting*).

On utilise : les ondelettes paresseuses (*lazy wavelets*) qui servent à séparer un vecteur en composantes paires et impaires, ainsi qu'une matrice polyphase qui permet de travailler sélectivement sur les composantes paires ou impaires du signal. On va factoriser la matrice polyphase et introduire alors deux opérations : une opération de prédiction (*Predict*) qui prédit les échantillons de rang pair à partir des échantillons de rang impair; une opération de mise-à-jour (*Update*) qui

* *aka* so_penible_animation

[1] le coût de calcul est est la moitié du coût de calcul de la FFT !

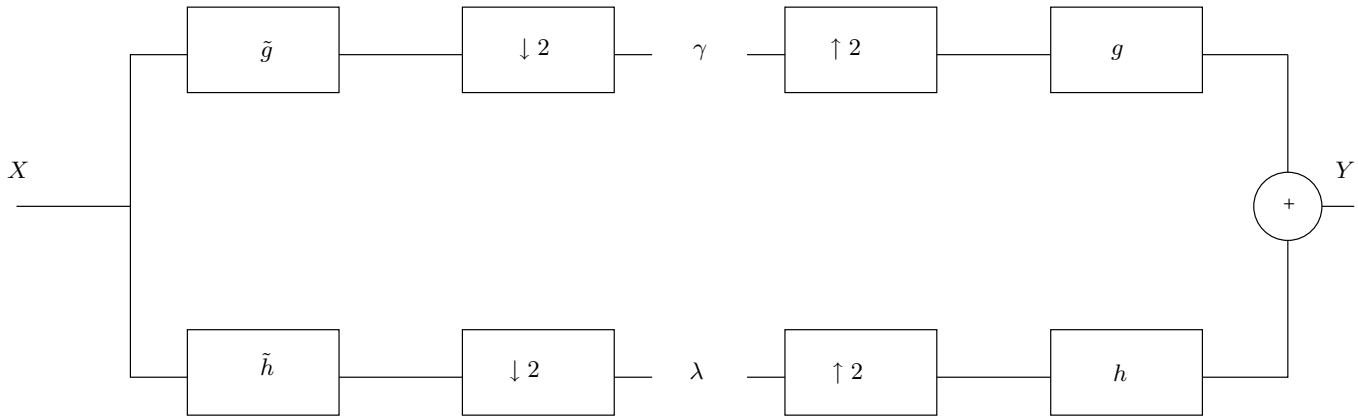

Figure 1: Schéma d'une implémentation en banc de filtres d'une transformation en ondelettes : le signal original X passe par les deux filtres complémentaires $\tilde{h}$ (passe-bas) et $\tilde{g}$ (passe-haut) en sortie on peut sous-échantillonner par 2 ($\downarrow 2$), on obtient alors respectivement des coefficients d'ondelettes γ et des coefficients d'échelle λ. La reconstruction du signal s'effectue par sur-échantillonnage par insertion de zéros ($\uparrow 2$) et passage par les filtres de synthèse h et g puis par addition pour obtenir le signal Y.

permet de conserver sur une partie du signal la valeur moyenne de l'ensemble du signal.

Le formalisme utilisé est celui des articles de références [Sweldens, Valens]. (Attention cependant aux écarts de notations entre les différents articles).

Cette transformation va en outre permettre de réaliser une transformation sur des entiers qui donne des entiers. Cependant il faudra utiliser une étape supplémentaire utilisant le lifting pour la mise à l'échelle.

Vers la réalisation de l'ondelette de Haar version *lifting*

On part des filtres d'analyse et de reconstruction de l'analyse multi-résolution classique par banc de filtres suivant le schéma classique de la Figure 1 :

$$\tilde{h}[n] = \frac{1}{\sqrt{2}}[1 \quad \underline{1}] \qquad \tilde{h}(z) = \frac{1}{\sqrt{2}}(1 + z^{-1})$$

$$\tilde{g}[n] = \frac{1}{\sqrt{2}}[-1 \quad \underline{1}] \qquad \tilde{g}(z) = \frac{1}{\sqrt{2}}(1 - z^{-1})$$

$$h[n] = \frac{1}{\sqrt{2}}[\underline{1} \quad 1] \qquad h(z) = \frac{1}{\sqrt{2}}(1 + z^{-1})$$

$$g[n] = \frac{1}{\sqrt{2}}[\underline{1} \quad -1] \qquad g(z) = \frac{1}{\sqrt{2}}(1 - z^{-1})$$

Attention aux notations, dans certaines références, on trouve souvent $\tilde{g}(z^{-1})$ et $\tilde{h}(z^{-1})$. Dans les expressions des filtres, le coefficient souligné correspond à l'indice $n = 0$.

Les filtres doivent en outre satisfaire les formules suivantes :

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

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

On a besoin de construire la matrice polyphase[2] $\tilde{P}$ (ainsi que sa matrice duale P), on utilise

[2]le terme polyphase vient de la théorie du filtrage numérique où il est utilisé pour décrire le partitionement

la formule suivante de décomposition polyphase sur les filtres précédents :

$$x(z) = x_e(z^2) + z^{-1}\ x_o(z^2) \tag{1}$$

où x_e désignent les échantillons pairs (*even*) et x_o les échantillons impairs (*odd*)[3].

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

$$\tilde{P}(z) = \begin{pmatrix} \tilde{h_e}(z) & \tilde{h_o}(z) \\ \tilde{g_e}(z) & \tilde{g_o}(z) \end{pmatrix}$$

Il vient assez facilement :

$$\tilde{h}(z) = \underbrace{\tilde{h_e}(z^2)}_{1} + z^{-1}.\ \underbrace{\tilde{h_o}(z^2)}_{1}$$

$$\tilde{g}(z) = \underbrace{\tilde{g_e}(z^2)}_{1} + z^{-1}.\ \underbrace{\tilde{g_o}(z^2)}_{-1}$$

$$\tilde{P}(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

sachant que $\det\tilde{P} = -1$ et que pour $\det\tilde{P} = 1$ on a les propriétés suivantes :
$\tilde{h}_e(z) = g_o(z^{-1})$
$\tilde{h}_o(z) = -g_e(z^{-1})$
$\tilde{g}_e(z) = -h_o(z^{-1})$
$\tilde{g}_o(z) = h_e(z^{-1})$

Ces proporiétés montrent que pour passer de l'analyse à la synthèse (de P à $\tilde{P}$) il suffit si $\det\tilde{P} = 1$, de prendre la matrice des cofacteurs en changeant de signe.

de plus sachant que $P(z)\tilde{P}(z^{-1}) = I$[4]

$$P(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Lifting primaire

Le lifting primaire est aussi noté *update*. Le *lifting* à proprement parler consiste à modifier le filtre g en gardant la complémentarité avec h à l'aide de la formule :

$$g_{new}(z) = g(z) + h(z)\ s(z^2)$$

où $s(z)$ est un polynôme de Laurent[5].

La matrice polyphase devient alors :

$$P_{new}(z) = P(z) \begin{pmatrix} 1 & s(z) \\ 0 & 1 \end{pmatrix}$$

---

d'une séquence déchantillons en plusieurs sous-séquences qui peuvent être traitées en parallèle, les sous-séquences peuvent-tre vues comme des versions d'elle-même décalées en phase d'où le nom.

[3] moyen mnémotechnique : *even* a un nombre de lettres *pair*, *odd* a un nombre de lettres *impair*

[4] l'inversion temporelle (z^{-1}) est nécessaire pour compenser le délai introduit par le filtrage

[5] Soit la transformée en Z d'un filtre FIR : $h(z) = \sum_{k=p}^{q} h[k]z^{-k}$ Cette somme est aussi nommée polynôme de Laurent ou encore série de Laurent.

De la même façon pour $\tilde{h}$

$$\tilde{h}_{new}(z) = \tilde{h}(z) + \tilde{g}(z)\ \tilde{s}(z^2)$$

où $\tilde{s}(z)$ est un polynôme de Laurent.

$$\tilde{P}_{new}(z) = \begin{pmatrix} 1 & \tilde{s}(z) \\ 0 & 1 \end{pmatrix} \tilde{P}(z)$$

Lifting dual

Le lifting dual est aussi appelé *prediction*.
Les formules précédentes modifient la sous-bande passe-bas à l'aide de la sous-bande passe-haut. Le lifting dual consiste en l'opération inverse : modifier la sous-bande passe-bas à l'aide de la sous-bande passe-haut.

$$h_{new}(z) = h(z) + g(z)\ t(z^2)$$
$$P_{new}(z) = P(z) \begin{pmatrix} 1 & 0 \\ t(z) & 1 \end{pmatrix}$$

et

$$\tilde{g}_{new}(z) = \tilde{g}(z) + \tilde{h}(z)\tilde{t}(z^2)$$
$$\tilde{P}_{new}(z) = \begin{pmatrix} 1 & 0 \\ t(z) & 1 \end{pmatrix} \tilde{P}(z)$$

où t est un polynôme de Laurent.

On a ainsi élevé (lifté) le niveau de sophistication de la transformation en ondelettes. On a de plus, pour avoir une transformation inversible :

$$t(z) = -\tilde{t}(z)$$

et

$$s(z) = -\tilde{s}(z)$$

Factorisation des filtres

La démarche ici est en quelque sorte la démarche inverse afin de pouvoir passer de forme connue de pairs de filtres d'ondelettes à leur implémentation en terme de lifting d'ondelettes.

on peut réécrire

$$\tilde{g}_{new}(z) = \tilde{g}(z) + \tilde{h}(z)\ \tilde{t}(z^2)$$

en

$$\tilde{g}(z) = \tilde{h}(z)\ \tilde{t}(z^2) + \tilde{g}_{new}(z)$$

qui est une forme de division avec reste[6] où $\tilde{g}_{new}$ est le reste. Alors :

$$\tilde{P}(z) = \begin{pmatrix} 1 & 0 \\ \tilde{t}(z) & 1 \end{pmatrix} \tilde{P}_{new}(z)$$

En itérant cet exemple, on peut arriver à obtenir une matrice polyphase qui est de la forme :

$$P(z) = \begin{pmatrix} K_1 & 0 \\ 0 & K_2 \end{pmatrix} \prod_{i=1}^{m} \begin{pmatrix} 1 & s_i(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t_i(z) & 1 \end{pmatrix}$$

où K_1 et K_2 sont deux constantes (différentes de zéro) que l'on peut éventuellement factoriser en quatre étapes de lifting.

[6] Pour réaliser la division avec reste on utilise l'algorithme d'Euclide.

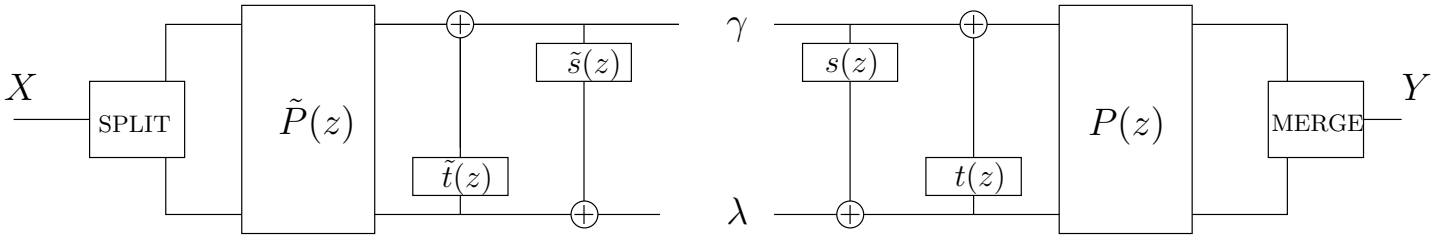

Figure 2: Lifting : chaque voie (sous-bande) est modifié par l'autre

Exemple de l'ondelette de Haar

On est déjà parti des filtres de la transformation en ondelettes de Haar et obtenu la matrice polyphase et sa forme duale. Il reste à factoriser pour avoir l'implémentation en lifting.

$$\tilde{P}(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

0.1 Lifting dual

On veut :

$$\tilde{P}(z) = \tilde{P}_{new}(z) \begin{pmatrix} 1 & 0 \\ \tilde{t}(z) & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

$$\tilde{P}(z) = \begin{pmatrix} \tilde{h_e}^{new}(z) & \tilde{h}_o^{new}(z) \\ \tilde{g_e}^{new}(z) & \tilde{g}_o^{new}(z) \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \tilde{t}(z) & 1 \end{pmatrix} = \begin{pmatrix} \tilde{h}_e^{new} + \tilde{h}_o^{new}(z)\,\tilde{t}(z) & \tilde{h}_o^{new}(z) \\ \tilde{g}_e^{new} + \tilde{g}_o^{new}(z)\,\tilde{t}(z) & \tilde{g}_o^{new}(z) \end{pmatrix}$$

Par identification, il vient :

$$\begin{aligned} \frac{1}{\sqrt{2}} &= \tilde{h_o}^{new}(z) \\ -\frac{1}{\sqrt{2}} &= \tilde{g_o}^{new}(z) \\ \frac{1}{\sqrt{2}} &= \frac{1}{\sqrt{2}}.\,\tilde{t}(z) + \tilde{h_e}^{new}(z) \\ \frac{1}{\sqrt{2}} &= -\frac{1}{\sqrt{2}}.\,\tilde{t}(z) + \tilde{g_e}^{new}(z) \end{aligned}$$

Dans notre cas très simple, il n'est pas nécessaire d'utiliser l'algorithme d'Euclide[7] (pour un exemple détaillé de l'utilisation de l'algorithme d'Euclide voir le cas de l'ondelette de Daubechies D4 1) et on peut prendre

[7]Exemple : soit $-\frac{1}{8}z^{-1} + \frac{3}{4} - \frac{1}{8}z = \tilde{t}(z)\left(\frac{1}{4} + \frac{1}{4}z\right) + \tilde{h}_e^{new}(z)$ On applique l'algorithme d'Euclide : $a(z) = b(z).q(z) + r(z)$. Soient ici $a_0(z) = a(z) = -\frac{1}{8}z^{-1} + \frac{3}{4} - \frac{1}{8}z$ et $b_0(z) = b(z) = \frac{1}{4} + \frac{1}{4}z$

- $a_1(z) = b_0(z)$
- $b_1(z) = a_0(z)\%b_0(z)$
- $q_1(z) = a_0(z)/b_0(z)$

% signifie modulo. La procédure est itérative, plusieurs choix sont en général possible pour le choix de l'ordre du diviseur.

$$-\frac{1}{8}z^{-1} + \frac{3}{4} - \frac{1}{8}z = \begin{cases} (-\frac{1}{2}z^{-1} + \frac{7}{2}) & (\frac{1}{4} + \frac{1}{4}z) & -z \\ (-\frac{1}{2}z^{-1} - \frac{1}{2}) & (\frac{1}{4} + \frac{1}{4}z) & -1 \\ (\frac{7}{2}z^{-1} - \frac{1}{2}) & (\frac{1}{4} + \frac{1}{4}z) & -z^{-1} \end{cases}$$

$$\tilde{g}_e^{new}(z) = 0$$

d'où

$$\tilde{t}(z) = -1$$

et

$$\tilde{h}_e^{new}(z) = \frac{2}{\sqrt{2}}$$

Alors,

$$\tilde{P}(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

0.2 Lifting primaire

On désire réécrire $\tilde{P}(z)$ sous la forme :

$$\tilde{P}(z) = \begin{pmatrix} \tilde{h}_e^{new}(z) & \tilde{h}_o^{new}(z) \\ \tilde{g}_e^{new}(z) & \tilde{g}_o^{new}(z) \end{pmatrix} \begin{pmatrix} 1 & \tilde{s}(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & 1 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

Par identification, il vient :

$$\begin{aligned} \frac{2}{\sqrt{2}} &= \tilde{h_e}^{new}(z) \\ 0 &= \tilde{g_e}^{new}(z) \\ \frac{1}{\sqrt{2}} &= \frac{2}{\sqrt{2}}.\ \tilde{s}(z) + \tilde{h_o}^{new}(z) \\ -\frac{1}{\sqrt{2}} &= 0.\ \tilde{s}(z) + \tilde{g_o}^{new}(z) \end{aligned}$$

on prend :

$$\tilde{s}(z) = \frac{1}{2}$$

et

$$\tilde{h}_o^{new}(z) = 0$$

et

$$\tilde{g}_o^{new}(z) = -\frac{1}{\sqrt{2}}$$

Alors

$$\tilde{P}(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

et

$$P(z) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1}{2} \\ 0 & 1 \end{pmatrix} \sqrt{2} \begin{pmatrix} \frac{1}{2} & 0 \\ 0 & -1 \end{pmatrix}$$

Ces opérations purement matricielles s'implémentent très facilement sous Scilab ou Matlab.

On peut de façon plus condensée (que l'écriture matricielle) écrire le pseudo code pour l'algorithme de calcul *in place*, x_e et x_o représente les coefficients pairs et impairs du signal (respectivement), d représente les détails (i.e. les coefficients issu du filtrage passe haut, donc les coefficients d'ondelettes), s représente le signal grossier (smooth) issu du fitlrage passe-bas et donc les coefficients d'échelle.

$$d = x_o - x_e;$$
$$s = x_o + d/2;$$
$$s = 2/\sqrt{2} * s;$$
$$d = -1/\sqrt{2} * d;$$

Pour le passage à une résolution supérieure on prend le code précédent que l'on réinitialise en commençant par :

$$x_o = s(1:2:length(s));$$
$$x_e = s(2:2:length(s));$$

La reconstruction se fait en inversant l'ordre des opérations (et leurs signes, sauf pour la normalisation ou on prend l'inverse).

$$d = -\sqrt{2} * d;$$
$$s = \sqrt{2} * s;$$
$$x_o = s - d/2;$$
$$x_e = x_o - d;$$

0.3 Implémentation sous Scilab

```
//lifting_Haar.sce
// ce programme implemente une transformation en ondelettes par le
/// lifting scheme de Sweldens
/// (a partir de lifting_int3.sce)
// Globalement par rapport a la transformee "classique" on
// procede d'abord au sous echantillonnage avant d'appliquer le filtre
// on separe coefficient pairs et impair (Split)


clear
xdel(winsid());
stacksize(60000000);




mtlb_load('donnees_rlp_sci.mat'); // un signal ECG sert de signal test
// d1,d2,d3,VR,VL,VF,V3,V4,V5,V6
sig=d2;

N=4096;

sig=round(sig(1:N)); // on arrondit pour travailler avec des entiers
sigold=sig;




//-------------------------------------------------------------------------
//-  Notations et donnees
//-
//-
//-------------------------------------------------------------------------
```

```
// on va prendre l'ondelette de Haar ... on utilise la decomposition polyphase (Valens, Sweldens)
// P est la matrice polyphase
// gam designe les coefficients d'ondelettes
// lam designe les coefficients d'echelle
// Pred est la matrice permettant de faire la prediction
// Upda est la mtrice permettant de faire la mise a jour
// Mnor est la matrice de normalisation
// fnor est le facteur de normalisation
// xe est le vecteur de composantes paires
// xo est le vecteur de composantes impaires
// X est le vecteur xe xo



Wsig=[]; // coefficients d'ondelettes

reso=2; // nombre de resolution


fnor=1/sqrt(2);
fnorinv=sqrt(2);


Mnor=[2 0; 0 -1];
Mnorinv=inv(Mnor);

Pred=[1 0; -1 1];
Upda=[1 1/2; 0 1];

Predi=[1 0; 1 1];
Updai=[1 -1/2; 0 1];



//-------------------------------------------------------------
//--- transformation duale (\tilde{P}) ----------------------
//-------------------------------------------------------------

for iter=1:1:reso
  xe=sig(2:2:$);
  xo=sig(1:2:$);
  X=[xe ; xo];

  Resu=fnor*Mnor*Upda*Pred*X;

  gam=Resu(2,:);
  lam=Resu(1,:);

  //--------------- Affichage -----------------------------------------

  xset('window',max(winsid()+1));
```

```
  subplot(211),plot2d([[1:length(gam)]',  [1:length(lam)]'] ,[gam', lam'],[2, 3],'181',"gama (ondele
  xtitle("coefficients d ondelettes et d echelle "," temps "," amplitude ");
  subplot(212),plot2d([[1:length(gam)]'] ,[gam'],[2],'181',"coefficients d ondelettes");
  xtitle("coefficients d ondelettes "," temps "," amplitude ");

  Wsig=[gam Wsig]; // stockage des coefficients d'ondelettes
  sig=lam;

   /// -- aparte ------------------------------------------------------------
   /// autre facon de l'ecrire "in-place" (plus compacte, moins matriciellement) '

  d=xo-xe;
  s=xo+d/2;
  s=2/sqrt(2)*s;
  d=-1/sqrt(2)*d;

  //--------------- Affichage ----------------------------------------

  //xset("window",max(winsid()+1)),
  //subplot(211),plot2d([[1:length(s)]'],[s'],[3],'081')
  //subplot(212),plot2d([ [1:length(d)]'],[ d'],[2],'081')
  /// -- fin de l aparte ---------------------------------------------------------


end  // fin de l'iteration duale


WSsig=[lam Wsig];      //stockage des coefficients d'echelle et d'ondelettes

//--------------- Affichage ----------------------------------------

xset('window',max(winsid()+1));
plot2d([[1:length(WSsig)]'] ,[WSsig'],[3],'181',"coefficient d ondelettes et d echelle");
//xtitle("coefficient d ondelettes et d echelle","temps","tension");
xtitle("coefficient d echelle et d ondelettes", string(reso)+ " resolutions","amplitude");

xset("window",max(winsid()+1)),
plot2d([[1:length(lam)]' [1:length(gam)]'],[lam', gam'],[3, 2],'081')




//-------------------------------------------------------------------
//------ transformation primaire (P) ----------------------------
//-------------------------------------------------------------------


dgam=gam;  // initialisation pour la reconstruction in place
slam=lam;



 for iter=1:1:reso
   rx=zeros(1,length(Resu));
```

```
   Orig=fnorinv*Predi*Updai*(Mnorinv)*Resu;

   rxe=Orig(1,:);
   rxo=Orig(2,:);

   rx(1:2:length(Resu))=(rxo);
   rx(2:2:length(Resu))=(rxe);

   if iter<reso
     gam=WSsig(length(rx)+1:2*length(rx));
     lam=rx;
   end

    /// -- aparte
    /// autre facon de l'ecrire "in-place" (moins matriciellement) '
        rrx=zeros(1,length(Resu));
        d=dgam;
        s=slam;

       d=-sqrt(2)*d;
       s=sqrt(2)*s;
       rrxo=s-d/2;
       rrxe=rrxo-d;
       rrx(1:2:length(Resu))=(rrxo);
       rrx(2:2:length(Resu))=(rrxe);

        xset('window',max(winsid()+1));
        plot2d([[1:length(rrx)]'] ,[rrx'],[2],'181',"signal reconstruit in place");
        xtitle("signal reconstruit in place ","temps","tension");

       if iter<reso
          dgam=WSsig(length(rrx)+1:2*length(rrx));
           slam=rrx;
        end
   /// -- fin de laparte


   Resu=[lam ; gam];
end

//--------------- Affichage ----------------------------------------

xset('window',max(winsid()+1));
subplot(211),plot2d([[1:length(rx)]',  [1:length(sigold)]'] ,[rx', sigold'],[2, 3],'181',"signal rec
xtitle("signal reconstruit et signal original","temps","tension");
subplot(212),plot2d([[1:length(sigold)]'] ,[(rx-sigold)'],[2],'181',"erreur entre les courbes");
xtitle("difference entre le signal reconstruit et le signal original","temps","ecart");

// fin du programme
```

On présente quelques illustrations du programme : le signal analysé est un signal ECG (fréquence d'échantillonnage 500 Hertz). Le résultat de l'analyse (i.e. coefficients d'ondelettes

et coefficients d'échelle) sont présentés sur la Figure 3. L'erreur de reconstruction de la Figure 4 est due à la présence du rationnel $\sqrt{2}$ dans les calculs.

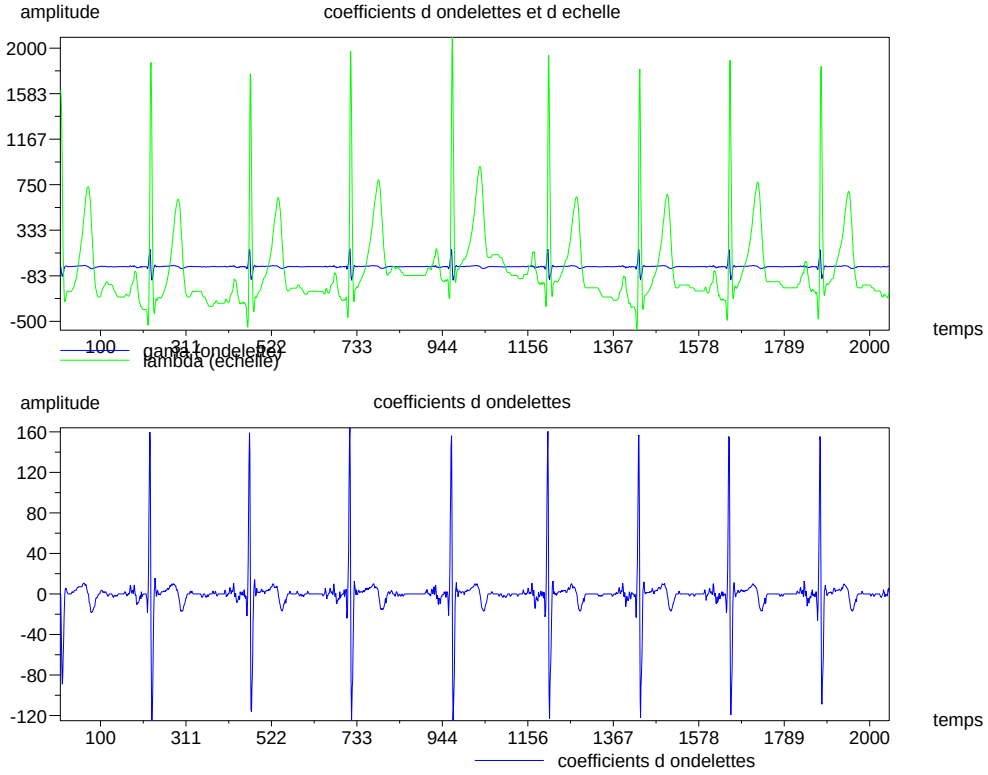

Figure 3: Coefficients d'ondelettes et d'échelle de la résolution 1

Un exemple de coefficients déchelle et d'ondelettes obtenus pour 3 niveaux de résolution est présenté sur la Figure 5.

Transformation en ondelettes qui transforme des entiers en entiers

Obtenir des coefficients d'ondelettes entiers peut-être un atout lorsque :

- des valeurs entières sont recommendées (notemment pour les images)
- pour certains processeurs embarqués ne traitent que les entiers
- l'occupation mémoire est prépondérante (un entier occupe moins de place qu'un flottant)

Lors d'une étape de lifting que l'on a étudié on a utilisé une relation de ce type :

$$x_{new}(z) \longleftarrow x(z) + s(z)\ y(z)$$

Puisque le signal $y(z)$ n'est pas modifié par cette étape de lifting alors on peut arrondir et réécrire l'étape du lifting.

$$x_{new}(z) \longleftarrow x(z) + \lfloor s(z)\ y(z) \rfloor$$

où $\lfloor . \rfloor$ note une opération d'arrondi.

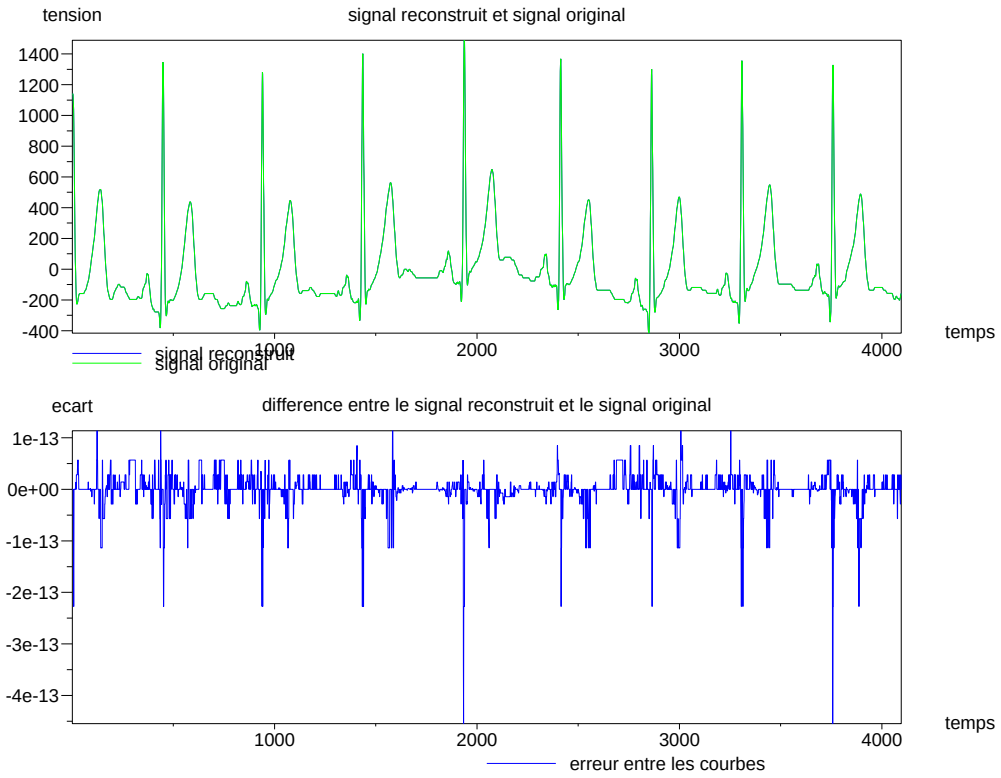

Figure 4: Signal original, signal reconstruit et erreur (différence entre les deux signaux

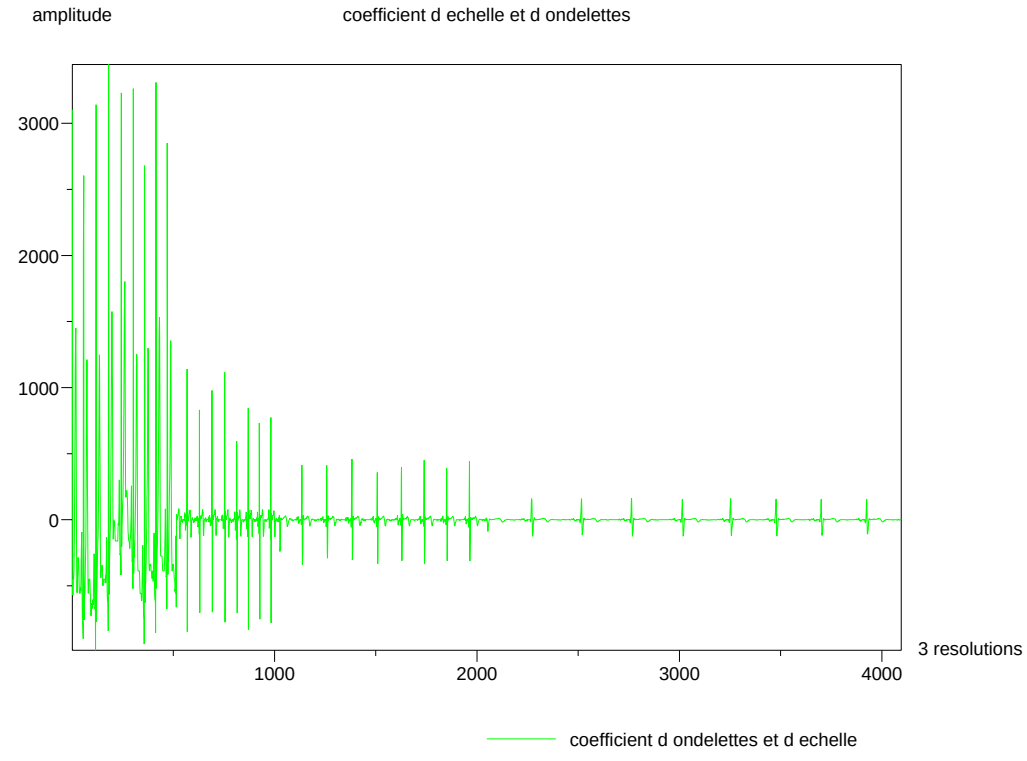

Figure 5: Coefficients d'échelle et d'ondelettes sur 3 niveaux de résolutions

Ce qui est très fort, c'est que, quelque soit l'opération d'arrondi utilisée, la transformation est inversible :

$$x(z) \longleftarrow x_{new}(z) - \lfloor s(z)\ y(z) \rfloor$$

Il y a cependant une précaution à prendre qui concerne la mise à l'échelle (*scaling*), dans le cas où ces coefficents ne sont pas entiers. La solution dans ce cas consiste à transformer cette mise à l'échelle en 3 étapes supplémentaires de lifting suivant le modèle :

$$P(z) = \begin{pmatrix} K & 0 \\ 0 & \frac{1}{K} \end{pmatrix} = \begin{pmatrix} 1 & K - K^2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1}{K} & 1 \end{pmatrix} \begin{pmatrix} 1 & K-1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

ou

$$P(z) = \begin{pmatrix} K & 0 \\ 0 & \frac{1}{K} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 - \frac{1}{K} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ K & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{K^2} - \frac{1}{K} \\ 0 & 1 \end{pmatrix}$$

On rappelle :

$$\tilde{P}(z) = \frac{1}{\sqrt{2}} \begin{pmatrix} 2 & 0 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} = \begin{pmatrix} \sqrt{2} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix}$$

Pour notre ondelette de Haar, on a un signe différent dans la matrice de normalisation, il suffit juste d'adapter les expressions précédentes avec par exemple pour la première (attention cependant lors de l'inversion):

$$P(z) = \begin{pmatrix} K & 0 \\ 0 & -\frac{1}{K} \end{pmatrix} = \begin{pmatrix} 1 & K - K^2 \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1}{K} & 1 \end{pmatrix} \begin{pmatrix} 1 & K-1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

avec $K = \sqrt{2}$

0.4 Implémentation sous Scilab

```
//lifting_Haar_int.sce
// ce programme implemente une transformation en ondelettes par le
/// lifting sceme de Sweldens

// Globalement par rapport a la transformee "classique" on
// procede d'abord au sous echantillonnage avant d'appliquer le filtre
// on separe coefficient pairs et impair (Split)


clear
xdel(winsid());
stacksize(60000000);

/// Le signal a tester est un signal ECG de frequence
//d'echantillonnage de 500Hz


mtlb_load('donnees_rlp_sci.mat');
// d1,d2,d3,VR,VL,VF,V3,V4,V5,V6
sig=d2;        // on choisit la derivation d2

N=4096; // Nb de points a considerer
```

```
sig=round(sig(1:N));////////////////// ATTENTION on arrondit au depart pour travailler sur des enti
sigold=sig;




//-------------------------------------------------------------------------------
//- Maintenant le lifting d'ondelettes
//-
//-
//-------------------------------------------------------------------------------

// on va prendre l'ondelette de Haar ... on utilise la decomposition polyphase (Valens)
// P est la matrice polyphase
// gam designe les coefficients d'ondelettes
// lam designe les coefficients d'echelle
// Pred est la matrice permettant de faire la prediction
// Upda est la mtrice permettant de faire la mise a jour
// Mnor est la matrice de normalisation
//
// xe est le vecteur de composantes paires
// xo est le vecteur de composantes impaires
// X est le vecteur xe xo


Wsig=[];

reso=3; // nombre de resolution




K=1/sqrt(2);


Mnor=[K 0; 0 -1/K];


P1=[1 K-K^2 ; 0 -1];
P2=[1 0 ; -1/K 1 ];
P3=[1 K-1 ; 0 1 ];
P4=[1 0 ; 1 1];




P1i=[1  -(-K+K^2) ; 0 -1];
P2i=[1 0;  1/K 1 ];
P3i=[1  -K+1; 0 1 ];
P4i=[1  0; -1 1];
```

```
Pred=[1 0; -1 1];
Upda=[1 1/2; 0 1];

Predi=[1 0; 1 1];
Updai=[1 -1/2; 0 1];



//-------------------------------------------------------------
//--- transformation duale (\tilde{P}) ----------------------
//-------------------------------------------------------------


for iter=1:1:reso

  xe=sig(2:2:$);
  xo=sig(1:2:$);

  X=[xe ; xo];

  PX=Pred*X;

  Updax=Upda(1,1)*PX(1,:)+round(Upda(1,2)*PX(2,:));
  Upday=Upda(2,1)*PX(1,:)+round(Upda(2,2)*PX(2,:));

  R4x=P4(1,1)*Updax+round(P4(1,2)*Upday);
  R4y=P4(2,1)*Updax+round(P4(2,2)*Upday);

  R3x=P3(1,1)*R4x+round(P3(1,2)*R4y);
  R3y=P3(2,1)*R4x+round(P3(2,2)*R4y);

  R2x=P2(1,1)*R3x+round(P2(1,2)*R3y);
  R2y=round(P2(2,1)*R3x)+round(P2(2,2)*R3y);

  R1x=P1(1,1)*R2x+round(P1(1,2)*R2y);
  R1y=P1(2,1)*R2x+round(P1(2,2)*R2y);


  Resu=[R1x ; R1y];


  gam=Resu(2,:);
  lam=Resu(1,:);

  //--------------- Affichage ----------------------------------------

  xset('window',max(winsid()+1));
  subplot(211),plot2d([[1:length(gam)]',  [1:length(lam)]'] ,[gam', lam'],[2, 3],'181',"gama (ondele
  xtitle("coefficients d ondelettes et d echelle "," temps "," amplitude ");
  subplot(212),plot2d([[1:length(gam)]'] ,[gam'],[2],'181',"coefficients d ondelettes");
  xtitle("coefficients d ondelettes "," temps "," amplitude ");
```

```
  Wsig=[gam Wsig]; // stockage des coefficients d'ondelettes
  sig=lam;

end  // fin de l'iteration duale



WSsig=[lam Wsig];      //stockage des coefficients d'echelle et d'ondelettes

//--------------- Affichage ----------------------------------------

xset('window',max(winsid()+1));
plot2d([[1:length(WSsig)]'] ,[WSsig'],[3],'181',"coefficient d ondelettes et d echelle");
//xtitle("coefficient d ondelettes et d echelle","temps","tension");
xtitle("coefficient d echelle et d ondelettes", string(reso)+ " resolutions","amplitude");

xset("window",max(winsid()+1)),
plot2d([[1:length(lam)]' [1:length(gam)]'],[lam', gam'],[3, 2],'081')




//-------------------------------------------------------------------
//------ transformation primaire (P) ----------------------------
//-------------------------------------------------------------------


 for iter=1:1:reso

  rx=zeros(1,length(Resu));

  R1xi=P1i(1,1)*Resu(1,:)+round(P1i(1,2)*Resu(2,:));
  R1yi=P1i(2,1)*Resu(1,:)+round(P1i(2,2)*Resu(2,:));

  R2xi=P2i(1,1)*R1xi+round(P2i(1,2)*R1yi);
  R2yi=round(P2i(2,1)*R1xi)+round(P2i(2,2)*R1yi);

  R3xi=P3i(1,1)*R2xi+round(P3i(1,2)*R2yi);
  R3yi=P3i(2,1)*R2xi+round(P3i(2,2)*R2yi);

  R4xi=P4i(1,1)*R3xi+round(P4i(1,2)*R3yi);
  R4yi=P4i(2,1)*R3xi+round(P4i(2,2)*R3yi);

  MR=[R4xi ; R4yi];

  Updaxi=(Updai(1,1)*MR(1,:))+round(Updai(1,2)*MR(2,:));
  Updayi=(Updai(2,1)*MR(1,:))+round(Updai(2,2)*MR(2,:));


  Orig=Predi*[Updaxi ; Updayi];

  rxe=Orig(1,:);
  rxo=Orig(2,:);
```

```
  rx(1:2:length(Resu))=(rxo);
  rx(2:2:length(Resu))=(rxe);

  if iter<reso
     gam=WSsig(length(rx)+1:2*length(rx));
     lam=rx;
  end

  Resu=[lam ; gam];

 end

//--------------- Affichage ----------------------------------------

xset('window',max(winsid()+1));
subplot(211),plot2d([[1:length(rx)]',  [1:length(sigold)]'] ,[rx', sigold'],[2, 3],'181',"signal rec
xtitle("signal reconstruit et signal original","temps","tension");
subplot(212),plot2d([[1:length(sigold)]'] ,[(rx-sigold)'],[2],'181',"erreur entre les courbes");
xtitle("difference entre le signal reconstruit et le signal original","temps","ecart");

// fin du programme
```

On a alors obtenu les résultats des Figures 6 et 7. On a pas les infimes d'erreurs de reconstruction précédentes grâce aux arrondis.

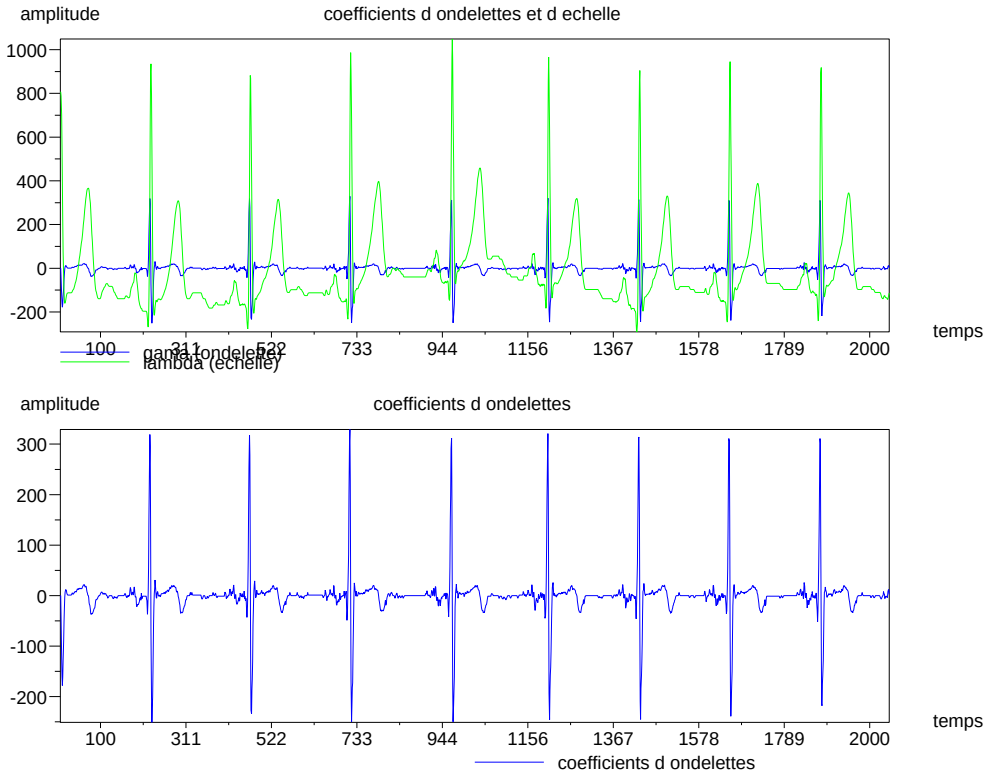

Figure 6: Coefficients d'ondelettes et d'échelle de la résolution 1

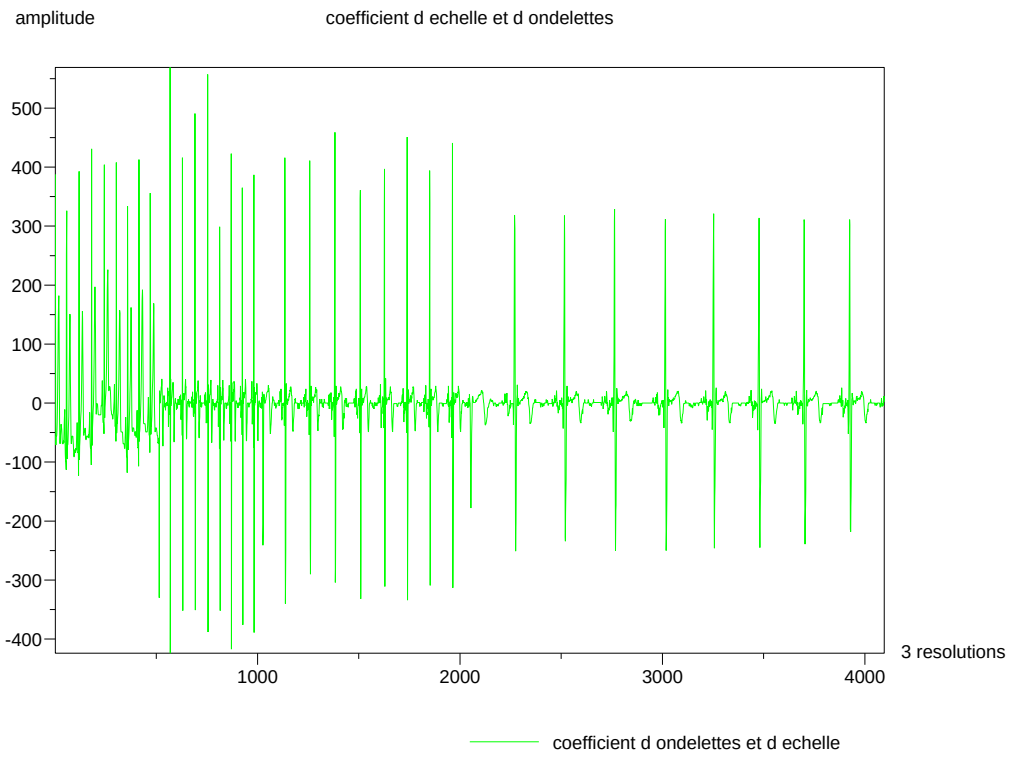

Figure 7: Coefficients d'échelle et d'ondelettes sur 3 niveaux de résolutions

On peut remarquer dans le programme précédent que l'on a perdu en partie le côté pratique du formalisme matriciel (à cause des opérations d'arrondis). À noter aussi que l'inconvénient majeur de conserver des entiers est d'avoir augmenté la dynamique du signal de sortie d'un facteur 2 (cf. Figure 7).

1 Autre exemple l'ondelette Daubechies(4) version lifting

Daubechies(4) signifie que les filtres de cette ondelette possèdent 4 coefficients et $n = 2$ moments nuls.

On a aussi :

$$\tilde{h}(z) = -z^{-1} g(-z^{-1})$$

et

$$\tilde{g}(z) = z^{-1} h(-z^{-1})$$

On trouve son expression dans les tables ou dans [Daubechies] :

$$\begin{aligned} h_0 = h[0] &= \tfrac{1+\sqrt{3}}{4\sqrt{2}} \\ h_1 = h[1] &= \tfrac{3+\sqrt{3}}{4\sqrt{2}} \\ h_2 = h[2] &= \tfrac{3-\sqrt{3}}{4\sqrt{2}} \\ h_3 = h[3] &= \tfrac{1-\sqrt{3}}{4\sqrt{2}} \end{aligned}$$

On a les propriétés intéressantes suivantes :

$$\begin{aligned} & h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \\ & h_3 h_1 + h_2 h_0 = 0 \\ & h_1^2 = -\tfrac{h_2 h_1 h_0}{h_3} \end{aligned}$$

soient

$$\begin{array}{rcl} h[0] & = & .482962913145 \\ h[1] & = & .836516303738 \\ h[2] & = & .224143868042 \\ h[3] & = & -.129409522551 \end{array}$$

$$h(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} + h_3 z^{-3}$$

et

$$g(z) = -h_3 z^2 + h_2 z^1 - h_1 + h_0 z^{-1}$$

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

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

$$h(z) = h_0 + h_2 z^{-2} + z^{-1}(h_1 + h_3 z^{-2})$$

d'où

$$h_e(z) = h_0 + h_2 z^{-1}$$

et

$$h_o(z) = h_1 + h_3 z^{-1}$$

de la même façon pour g

$$g(z) = -h_3 z^2 - h_1 + z^{-1}(h_2 z^2 + h_0)$$

$$g_e(z) = -h_3 z - h_1$$

et

$$g_o(z) = h_2 z + h_0$$

d'où

$$P(z) = \begin{pmatrix} h_0 + h_2 z^{-1} & -h_3 z - h_1 \\ h_1 + h_3 z^{-1} & h_2 z + h_0 \end{pmatrix}$$

On va écrire une étape de lifting primaire :

$$P(z) = \begin{pmatrix} 1 & s(z) \\ 0 & 1 \end{pmatrix} \begin{pmatrix} h_e^{new}(z) & g_e^{new}(z) \\ h_o^{new}(z) & g_o^{new}(z) \end{pmatrix}$$

Alors,

$$\begin{cases} h_o^{new}(z) = h_1 + h_3 z^{-1} \\ g_o^{new}(z) = h_2 z + h_0 \\ h_0 + h_2 z^{-1} = s(z)(h_1 + h_3 z^{-1}) + h_e^{new}(z) \\ -h_3 z - h_1 = s(z)(h_2 z + h_0) + g_e^{new}(z) \end{cases}$$

On utilise alors la division euclidienne : (on note que plusieurs factorisations sont possibles suivant l'ordre dans lequel on choisit de prendre en compte diviseur et dividende [8].)

Par exemple :

$$\begin{array}{rl|l} h_2 z^{-1} & +h_0 & h_3 z^{-1} + h_1 \\ \hline -(h_2 z^{-1} & +\frac{h_2 h_1}{h_3}) & \frac{h_2}{h_3} \\ \hline 0 & +\frac{h_3 h_0 - h_2 h_1}{h_3} & \end{array}$$

ou alors

$$\begin{array}{rl|l} h_0 & +h_2 z^{-1} & h_1 + h_3 z^{-1} \\ \hline -(h_0 & +\frac{h_3 h_0}{h_1} z^{-1}) & \frac{h_0}{h_1} \\ \hline 0 & +\frac{h_2 h_1 - h_3 h_0}{h_1} z^{-1} & \end{array}$$

etc. ...

On se limite à l'exploration de la première solution (les autres aboutiront aussi à des formulations différentes mais équivalentes).

On a alors :

$$s(z) = \frac{h_2}{h_3}$$

et

$$h_e^{new}(z) = \frac{h_3 h_0 - h_2 h_1}{h_3}$$

on explicite alors $g_e^{new}(z)$:

$$\begin{array}{rcl} g_e^{new}(z) & = & -h_3 z - h1 - \frac{h_2}{h_3}(h_2 z + h_0) \\ & = & -(\frac{h_3^2 + h_2^2}{h_3})z - \frac{h_1 h_3 + h_2 h_0}{h_3} \\ & = & -(\frac{h_3^2 + h_2^2}{h_3})z \end{array}$$

d'où

$$P(z) = \begin{pmatrix} 1 & \frac{h2}{h3} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{h_3 h_0 - h_2 h_1}{h_3} & -\frac{h_3^2 + h_2^2}{h_3} z \\ h_1 + h_3 z^{-1} & h_2 z + h_0 \end{pmatrix}$$

Maintenant, le lifiting dual :

$$P(z) = \begin{pmatrix} 1 & \frac{h2}{h3} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ t(z) & 1 \end{pmatrix} \begin{pmatrix} h_e^{new}(z) & g_e^{new}(z) \\ h_o^{new}(z) & g_o^{new}(z) \end{pmatrix} = \begin{pmatrix} 1 & \frac{h2}{h3} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{h_3 h_0 - h_2 h_1}{h_3} & -\frac{h_3^2 + h_2^2}{h_3} z \\ h_1 + h_3 z^{-1} & h_2 z + h_0 \end{pmatrix}$$

alors,

$$\begin{cases} h_e^{new}(z) = \frac{h_3 h_0 - h_2 h_1}{h_3} \\ g_e^{new}(z) = -\frac{h_3^2 + h_2^2}{h_3} z \\ h_1 + h_3 z^{-1} = t(z)(\frac{h_3 h_0 - h_2 h_1}{h_3}) + h_o^{new}(z) \\ h_2 z + h_0 = t(z)(-\frac{h_3^2 + h_2^2}{h_3} z) + g_o^{new}(z) \end{cases}$$

[8] le nombre de possibilités différentes est lié aux degrés (noté $|.|$) des polynômes de Laurent des diviseurs et dividendes. Le degré d'un polynôme de Laurent quelconque $h(z)$ défini par $h(z) = \sum_{k=p}^{q} h[k] z^{-k}$ est $|h| = q - p$. Soit $a = bq + r$, si le degré du diviseur et du dividende sont égaux alors le nombre de possibilités est $4 \cdot 3^{|b|-1}$, si maintenant $|a| = |b| + 1$ alors le nombre de possibilités est $3^{|b|}$

$$\begin{array}{rl|l} h_1 & +h_3z^{-1} & \frac{h_3h_0-h_2h_1}{h_3} \\ \hline -(h_1) & & \frac{h_3h_1}{h_3h_0-h_2h_1}+\frac{h_3^2}{h_3h_0-h_2h_1}z^{-1} \\ \hline 0 & +h_3z^{-1} & \\ & -(h_3z^{-1}) & \\ \hline & 0 & \end{array}$$

donc

$$t(z)=\frac{h_3h_1}{h_3h_0-h_2h_1}+\frac{h_3^2}{h_3h_0-h_2h_1}z^{-1}$$

et

$$h_o^{new}(z)=0$$

alors,

$$\begin{aligned} g_o^{new}(z) &= h_2z+h_0-\left(\tfrac{h_3h_1+h_3^2z^{-1}}{h_3h_0-h_2h_1}\right)\left(-\tfrac{h_3^2+h_2^2}{h_3}z\right) \\ &= \left(h_2+\tfrac{h_1(h_3^2+h_2^2)}{h_3h_0-h_2h_1}\right)z+\tfrac{h_3(h_3^2+h_2^2)}{h_3h_0-h_2h_1}+h_0 \end{aligned}$$

Or

$$h_2+\frac{h_1(h_3^2+h_2^2)}{h_3h_0-h_2h_1}=0$$

d'où

$$g_o^{new}(z)=\frac{h_3(h_3^2+h_2^2)}{h_3h_0-h_2h_1}+h_0$$

Voilà :

$$P(z)=\begin{pmatrix} 1 & \frac{h_2}{h_3} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \frac{h_3h_1}{h_3h_0-h_2h_1}+\frac{h_3^2}{h_3h_0-h_2h_1}z^{-1} & 1 \end{pmatrix}\begin{pmatrix} \frac{h_3h_0-h_2h_1}{h_3} & -\frac{h_3^2+h_2^2}{h_3}z \\ 0 & \frac{h_3(h_3^2+h_2^2)}{h_3h_0-h_2h_1}+h_0 \end{pmatrix}$$

On peut maintenant mettre en évidence la mise à l'échelle d'abord en réécrivant un peu les derniers coefficients obtenus à l'aide des formules sur les coefficients h_i $(0\leq i\leq 3)$:

$$\frac{h_3(h_3^2+h_2^2)}{h_3h_0-h_2h_1}+h_0=h_3\frac{h_3^2+h_2^2+h_0^2-\frac{h_2h_1h_0}{h_3}}{h_3h_0-h_2h_1}$$

d'où

$$\frac{h_3(h_3^2+h_2^2)}{h_3h_0-h_2h_1}+h_0=\frac{h_3}{h_3h_0-h_2h_1}$$

et

$$P(z)=\begin{pmatrix} 1 & \frac{h2}{h3} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \frac{h_3h_1}{h_3h_0-h_2h_1}+\frac{h_3^2}{h_3h_0-h_2h_1}z^{-1} & 1 \end{pmatrix}\begin{pmatrix} \frac{h_3h_0-h_2h_1}{h_3} & -\frac{h_3^2+h_2^2}{h_3}z \\ 0 & \frac{h_3}{h_3h_0-h_2h_1} \end{pmatrix}$$

On a aussi

$$-\frac{h_3^2+h_2^2}{h_3}=\frac{h_3}{h_3h_0-h_2h_1}$$

$$P(z)=\begin{pmatrix} 1 & \frac{h2}{h3} \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \frac{h_3h_1}{h_3h_0-h_2h_1}+\frac{h_3^2}{h_3h_0-h_2h_1}z^{-1} & 1 \end{pmatrix}\begin{pmatrix} \frac{h_3h_0-h_2h_1}{h_3} & \frac{h_3}{h_3h_0-h_2h_1}z \\ 0 & \frac{h_3}{h_3h_0-h_2h_1} \end{pmatrix}$$

On peut donc introduire l'étape de mise à l'échelle (normalisation), en remarquant que

$$\begin{pmatrix} K & \frac{1}{K}z \\ 0 & \frac{1}{K} \end{pmatrix} = \begin{pmatrix} 1 & z \\ 0 & 1 \end{pmatrix} \begin{pmatrix} K & 0 \\ 0 & \frac{1}{K} \end{pmatrix}$$

d'où finalement

$$P(z) = \begin{pmatrix} 1 & \frac{h2}{h3} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{h_3h_1}{h_3h_0-h_2h_1} + \frac{h_3^2}{h_3h_0-h_2h_1}z^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & z \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{h_3h_0-h_2h_1}{h_3} & 0 \\ 0 & \frac{h_3}{h_3h_0-h_2h_1} \end{pmatrix}$$

c'est-à-dire

$$P(z) = \begin{pmatrix} 1 & -\sqrt{3} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{\sqrt{3}}{4} + \frac{\sqrt{3}-2}{4}z^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & z \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\sqrt{3}+1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}-1}{\sqrt{2}} \end{pmatrix}$$

et alors :

$$\tilde{P}(z) = \begin{pmatrix} \frac{\sqrt{3}-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}+1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} 1 & -z \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{\sqrt{3}}{4} - \frac{\sqrt{3}-2}{4}z^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sqrt{3} \\ 0 & 1 \end{pmatrix}$$

On peut de façon plus condensée (que l'écriture matricielle) écrire le pseudo code pour l'algorithme de calcul *in place*, x_e et x_o représente les coefficients pairs et impairs du signal (respectivement), d représente les détails (i.e. les coefficients issu du filtrage passe haut, donc les coefficients d'ondelettes), s représente le signal grossier (smooth) issu du fitlrage passe-bas et donc les coefficients d'échelle.

$s = x_o + \sqrt{3}x_e;$
$d = x_e - \sqrt{3}/4 * s - (\sqrt{3}-2)/4 * [0 \;\; s(1:1:length(s)-1)];$
$s = s - [d(2:1:length(d)) \;\; 0];$
$d = (\sqrt{3}+1)/\sqrt{2} * d;$
$s = (\sqrt{3}-1)/\sqrt{2} * s;$

Pour le passage à une résolution supérieure on prend le code précédent que l'on réinitialise en commençant par :

$x_o = s(1:2:length(s));$
$x_e = s(2:2:length(s));$

L'implémentation peut se faire facilement suivant l'exemple de la Figure 8.

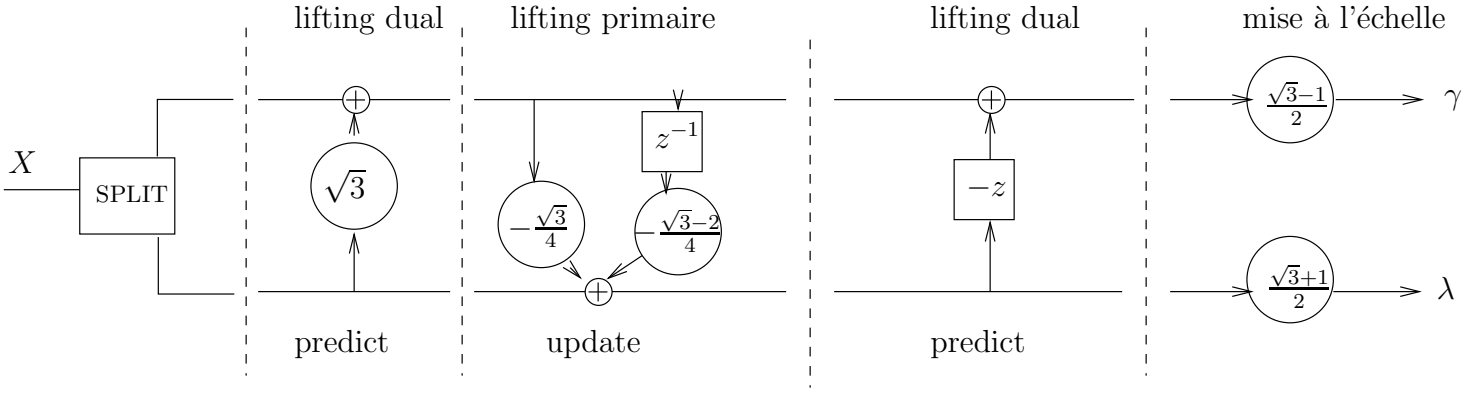

Figure 8: Schéma d'une implémentation du lifting

Implémentation sous Scilab

```
// lifting_D4.sce
// ce programme implemente une transformation en ondelettes par le
/// lifting scheme de Sweldens pour les ondelettes Daubechies(4) (D4)
///
// Globalement par rapport a la transformee "classique" on
// procede d'abord au sous echantillonnage avant d'appliquer le filtre
// on separe coefficient pairs et impair (Split)


clear
xdel(winsid());
stacksize(60000000);


/// Le signal a tester est un signal ECG de frequence
//d'echantillonnage de 500Hz

mtlb_load('donnees_rlp_sci.mat');   // d1,d2,d3,VR,VL,VF,V3,V4,V5,V6
sig=d2; // on choisit la derivation d2

N=4096; // Nb de points a considerer

//////-------------------------------------------------------------------------------------------



  h0=(1+sqrt(3))/(4*sqrt(2));
  h1=(3+sqrt(3))/(4*sqrt(2));
  h2=(3-sqrt(3))/(4*sqrt(2));
  h3=(1-sqrt(3))/(4*sqrt(2));


//////-------------------------------------------------------------------------------------------


sig=round(sig(1:N));////////////////// ATTENTION on arrondit au depart pour travailler sur des enti
sigold=sig;




//-----------------------------------------------------------------------------
//-  Notations et donnees
//-
//-
//-----------------------------------------------------------------------------

// on va prendre l'ondelette de Daubechies D4 ... on utilise la decomposition polyphase (Valens)
// P est la matrice polyphase
// gam designe les coefficients d'ondelettes
// lam designe les coefficients d'echelle
// Pred est la matrice permettant de faire la prediction
```

```
// Upda est la mtrice permettant de faire la mise a jour
// Mnor est la matrice de normalisation
//
// xe est le vecteur de composantes paires
// xo est le vecteur de composantes impaires
// X est le vecteur xe xo


Wsig=[];

reso=3; // nombre de resolutions

K=(sqrt(3)+1)/sqrt(2);

Mnor=[1/K 0; 0 K];

P1=[1 K-K^2 ; 0 1];
P2=[1 0 ; -1/K 1 ];
P3=[1 K-1 ; 0 1 ];
P4=[1 0 ; 1 1];




P1i=[1  (-K+K^2) ; 0 1];
P2i=[1 0;  1/K 1 ];
P3i=[1  -K+1; 0 1 ];
P4i=[1  0; -1 1];



Pred=[1 0; sqrt(3) 1];
Upda=[1 -sqrt(3)/4; 0 1];    /// Attention Updazm(1,2) fait intervenir echantillon precedent
Updazm=[0 -((sqrt(3)-2)/4); 0 0];
Pred2=[1 0; 0 1];                                /// Attention Pred2zp(2,1) fait intervenir echantill
Pred2zp=[0 0; -1 0];


Predi=[1 0; -sqrt(3) 1];
Updai=[1 sqrt(3)/4  ; 0 1];   /// Attention Updaizm(1,2) fait intervenir echantillon precedent
Updaizm=[0 ((sqrt(3)-2)/4)  ; 0 0];
Pred2i=[1 0; 0 1];                                 /// Attention Predi2zp(2,1) fait intervenir echant
Pred2izp=[0 0; 1 0];

//-----------------------------------------------------------
//--- transformation duale (\tilde{P}) ----------------------
//-----------------------------------------------------------


for iter=1:1:reso

  xe=sig(2:2:$);
  xo=sig(1:2:$);
```

```
X=[xe ; xo];


 Px=Pred(1,1)*X(1,:)+(Pred(1,2)*X(2,:));    // d   (details) passe-haut= coeff d'ondelettes
 Py=Pred(2,1)*X(1,:)+(Pred(2,2)*X(2,:));  // s  (smooth)= passe-bas= coeff d'echelle

 Updax=Upda(1,1)*Px+(Upda(1,2)*Py+Updazm(1,2)*[0 Py(1:1:$-1) ]);  // d
 Upday=Upda(2,1)*Px+(Upda(2,2)*Py);   //s

 Pxx=Pred2(1,1)*Updax+(Pred2(1,2)*Upday);   //d
 Pyy=(Pred2(2,1)*Updax+Pred2zp(2,1)*[Updax(2:1:$) 0])+(Pred2(2,2)*Upday);  //s

 R4x=P4(1,1)*Pxx+(P4(1,2)*Pyy); //d
 R4y=P4(2,1)*Pxx+(P4(2,2)*Pyy); //s

 R3x=P3(1,1)*R4x+(P3(1,2)*R4y); //d
 R3y=P3(2,1)*R4x+(P3(2,2)*R4y); //s

 R2x=P2(1,1)*R3x+(P2(1,2)*R3y); //d
 R2y=(P2(2,1)*R3x)+(P2(2,2)*R3y); //s

 R1x=P1(1,1)*R2x+(P1(1,2)*R2y); //d
 R1y=P1(2,1)*R2x+(P1(2,2)*R2y); //s


Resu=[R1x ; R1y];

gam=Resu(1,:); //d
lam=Resu(2,:); //s


//--------------- Affichage ----------------------------------------

xset('window',max(winsid()+1));
subplot(211),plot2d([[1:length(gam)]',  [1:length(lam)]'] ,[gam', lam'],[2, 3],'181',"gama (ondele
xtitle("coefficients d ondelettes et d echelle "," temps "," amplitude ");
subplot(212),plot2d([[2:length(gam)]'] ,[gam(2:1:$)'],[2],'181',"coefficients d ondelettes");
xtitle("coefficients d ondelettes "," temps "," amplitude ");
Wsig=[gam Wsig]; // stockage des coefficients d'ondelettes'
sig=lam;


///-- aparte ------------------------------------------
/// autre facon de l'ecrire "in-place" (plus compacte, moins matriciellement) '

s=xo+sqrt(3)*xe;
d=xe-sqrt(3)/4*s-(sqrt(3)-2)/4*[0 s(1:1:$-1)];
s=s-[d(2:1:$) 0];
d=(sqrt(3)+1)/sqrt(2)*d;
s=(sqrt(3)-1)/sqrt(2)*s;

//--------------- Affichage ----------------------------------------
```

```
  //xset("window",max(winsid()+1)),
  //subplot(211),plot2d([[1:length(s)]'],[s'],[3],'081')
  //subplot(212),plot2d([ [1:length(d)]'],[ d'],[2],'081')

   ///-- fin de l' aparte ------------------------------------------'

  /// comparaison avec les  banc de filtres
  /// wt paquetage --- uvi_wave ---
  ///   wcoef=wt(round(sigold),dH,dG,iter,0,0);
  ///   xset("window",max(winsid()+1)),
  ///  plot2d([1:length(wcoef)]',wcoef,[2],'081')

end  // fin de l'iteration duale



 WSsig=[lam Wsig];      //stockage des coefficients d'echelle et d'ondelettes

//--------------- Affichage ----------------------------------------

 xset('window',max(winsid()+1));
 plot2d([[1:length(WSsig)]'] ,[WSsig'],[3],'181',"coefficient d ondelettes et d echelle");
 xtitle("coefficient d echelle et d ondelettes", string(reso)+ " resolutions","amplitude");

 xset("window",max(winsid()+1)),
 plot2d([[1:length(lam)]' [1:length(gam)]'],[lam', gam'],[3, 2],'081')


//-------------------------------------------------------------------
//------ transformation primaire (P) ----------------------------
//-------------------------------------------------------------------


dgam=gam;  // initialisation pour le in-place
slam=lam;

 for iter=1:1:reso


  rx=zeros(1,length(Resu));

  R1xi=P1i(1,1)*Resu(1,:)+(P1i(1,2)*Resu(2,:));
  R1yi=P1i(2,1)*Resu(1,:)+(P1i(2,2)*Resu(2,:));

  R2xi=P2i(1,1)*R1xi+(P2i(1,2)*R1yi);
  R2yi=P2i(2,1)*R1xi+(P2i(2,2)*R1yi);

  R3xi=P3i(1,1)*R2xi+(P3i(1,2)*R2yi);
  R3yi=P3i(2,1)*R2xi+(P3i(2,2)*R2yi);

  R4xi=P4i(1,1)*R3xi+(P4i(1,2)*R3yi);
  R4yi=P4i(2,1)*R3xi+(P4i(2,2)*R3yi);

  Pxxi=Pred2i(1,1)*R4xi+(Pred2i(1,2)*R4yi);
```

```
 Pyyi=(Pred2i(2,1)*R4xi+Pred2izp(2,1)*[R4xi(2:1:$) 0])+(Pred2i(2,2)*R4yi);

 Updaxi=Updai(1,1)*Pxxi+(Updai(1,2)*Pyyi+Updaizm(1,2)*[0 Pyyi(1:1:$-1)]);
 Updayi=Updai(2,1)*Pxxi+(Updai(2,2)*Pyyi);

 Pxi=Predi(1,1)*Updaxi+(Predi(1,2)*Updayi);
 Pyi=Predi(2,1)*Updaxi+(Predi(2,2)*Updayi);

 Orig=[Pxi ; Pyi];

 rxe=Orig(1,:);
 rxo=Orig(2,:);

 rx(1:2:length(Resu))=(rxo);
 rx(2:2:length(Resu))=(rxe);

 if iter<reso
    gam=WSsig(length(rx)+1:2*length(rx));
    lam=rx;
 end

  ///  -- aparte ----------------------------------------------------------
  /// autre facon de l'ecrire "in-place" (moins matriciellement) '
    rrx=zeros(1,length(Resu));
    d=dgam;
    s=slam;

    s=(sqrt(3)+1)/sqrt(2)*s;
    d=(sqrt(3)-1)/sqrt(2)*d;
    s=s+[d(2:1:$) 0];
    rrxe=d+sqrt(3)/4*s+(sqrt(3)-2)/4*[0 s(1:1:$-1)];
    rrxo=s-sqrt(3)*rrxe;
    rrx(1:2:length(Resu))=(rrxo);
    rrx(2:2:length(Resu))=(rrxe);

    xset('window',max(winsid()+1));
    plot2d([[1:length(rrx)]'] ,[rrx'],[2],'181',"signal reconstruit in place");
    xtitle("signal reconstruit in place ","temps","tension");

    if iter<reso
     dgam=WSsig(length(rrx)+1:2*length(rrx));
     slam=rrx;
    end

  ///  -- fin de l aparte -------------------------------------------------------


 Resu=[gam ; lam];



end
```

```
//--------------- Affichage ----------------------------------------

xset('window',max(winsid()+1));
subplot(211),plot2d([[1:length(rx)]',  [1:length(sigold)]'] ,[rx', sigold'],[2, 3],'181',"signal rec
xtitle("signal reconstruit et signal original","temps","tension");
subplot(212),plot2d([[1:length(sigold)]'] ,[(rx-sigold)'],[2],'181',"erreur entre les courbes");
xtitle("difference entre le signal reconstruit et le signal original","temps","ecart");

// fin du programme
```

On présente encore quelques illustrations du programme : le signal analysé est un signal ECG (fréquence d'échantillonnage 500 Hertz). Le résultat de l'analyse (i.e. coefficients d'ondelettes et coefficients d'échelle) sont présentés sur la Figure 9. Les infimes erreurs de reconstruction de la Figure 10 est due à la présence des rationnels $\sqrt{3}$ et $\sqrt{2}$ dans les calculs.

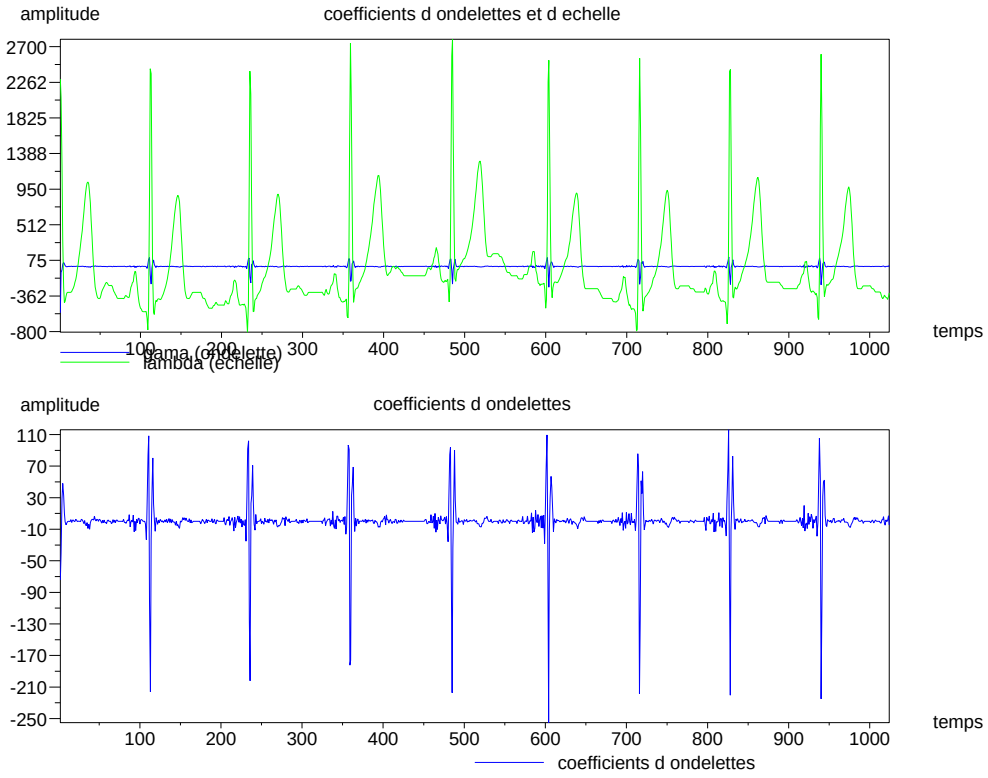

Figure 9: Coefficients d'ondelettes et d'échelle de la résolution 1

Un exemple de coefficients déchelle et d'ondelettes obtenus pour 3 niveaux de résolution est présenté sur la Figure 11.

(On peut également, comme pour l'ondelette de Haar, effectuer avec l'ondelette de Daubechies la transformation d'entiers en coefficients d'ondelettes entier.)

Voilà, *That's all Folks !!!*

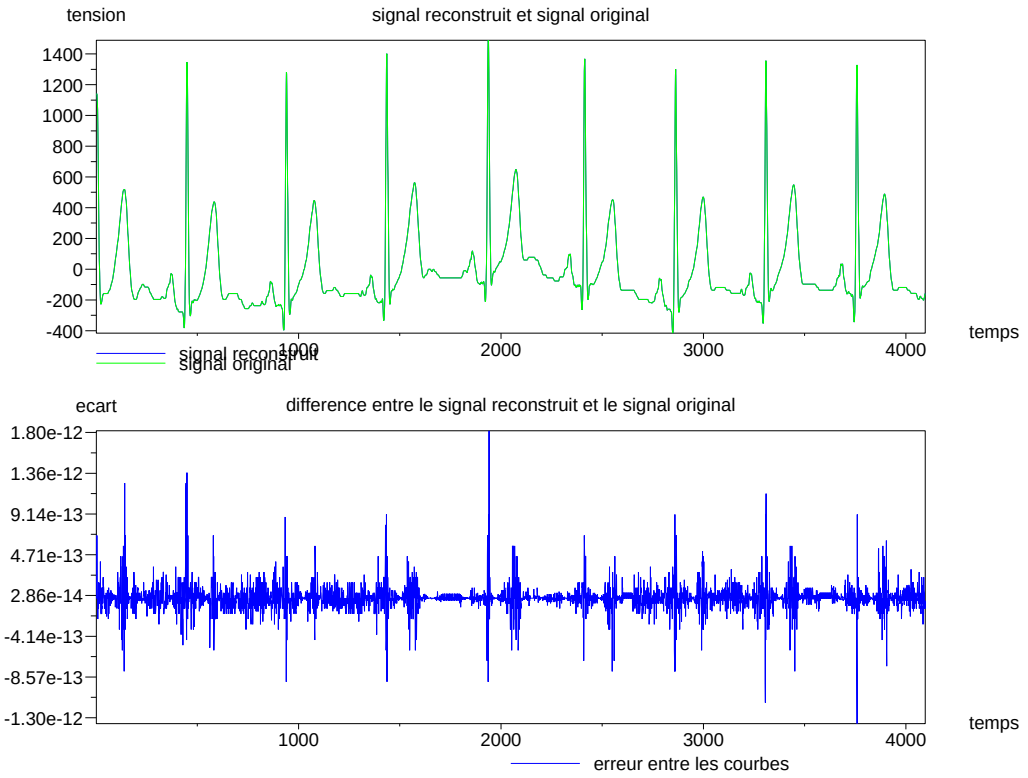

Figure 10: Signal original, signal reconstruit et erreur (différence entre les deux signaux

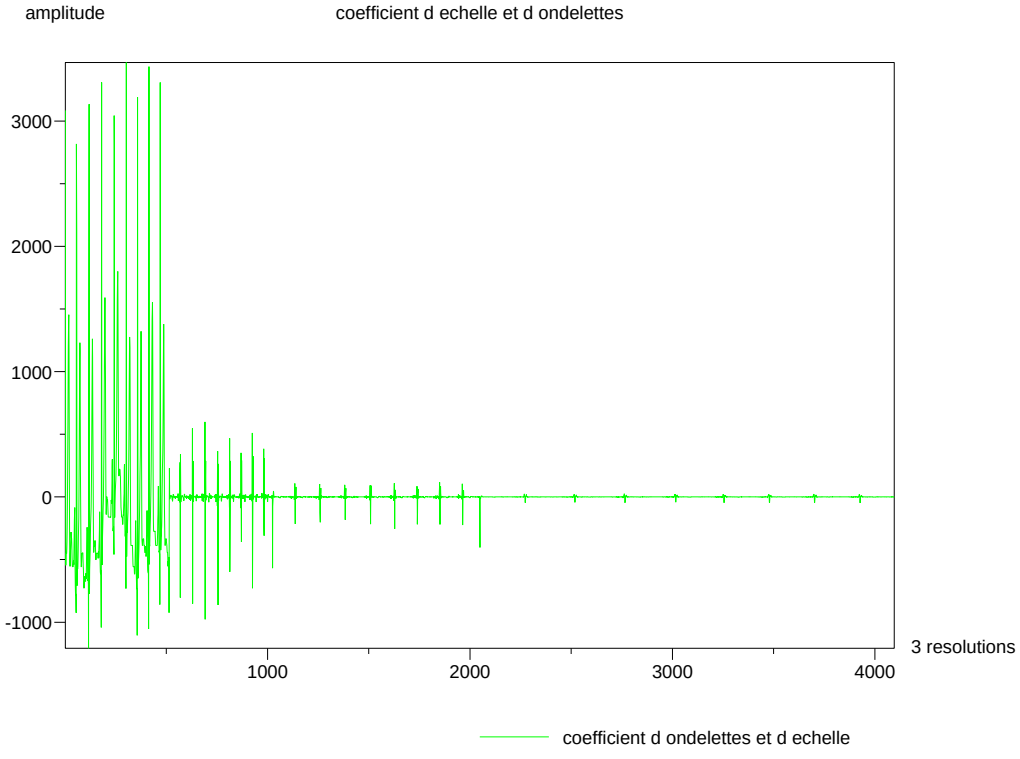

Figure 11: Coefficients d'échelle et d'ondelettes sur 3 niveaux de résolutions