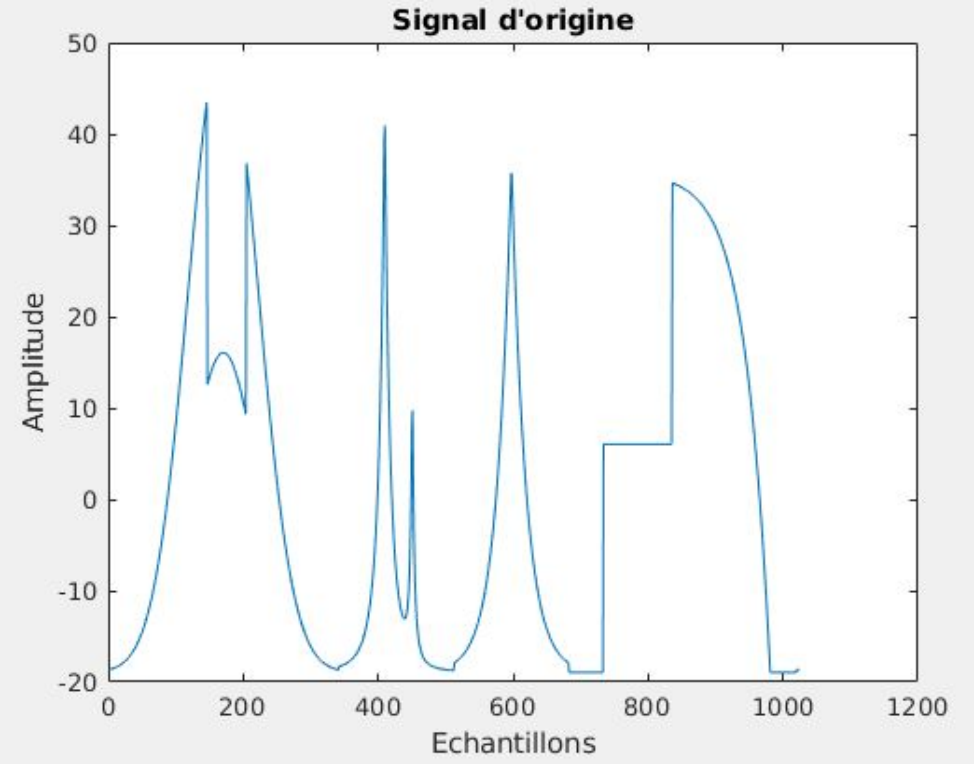
Eftimakis Arthur
Al Safandi Yann

TP : Compression d’images par ondelettes

2- Décomposition en ondelettes et reconstruction parfaite

Chargement du fichier et représentation des signaux test :

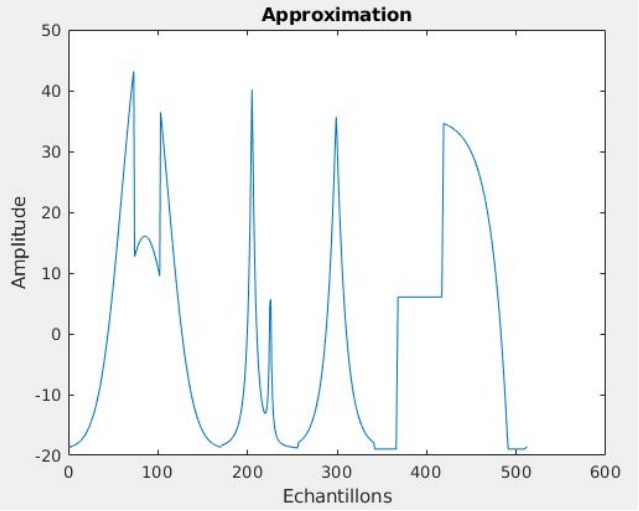
```
load('PieceRegSig.mat'); % crée implicitement une variable sig
figure(1)
plot(sig);
title("Signal d'origine");
xlabel("Echantillons");
ylabel("Amplitude");
```

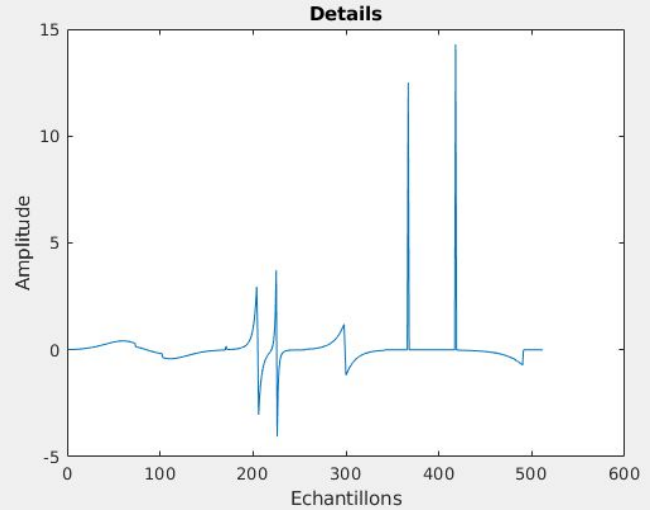


2-1 Première décomposition dans la base de Haar

On sépare les valeurs selon la parité de leur indice. On crée ensuite un vecteur e de longueur moitié, contenant les moyennes de deux valeurs consécutives de sig.

```
% Première décomposition dans la base de Haar
n = length(sig);
ind1 = [1:2:n];
ind2 = [2:2:n];
e = (sig(ind1) + sig(ind2))/2;

figure(1)
plot(e);
title('Approximation');
xlabel("Echantillons");
ylabel("Amplitude");

w = (sig(ind2) - sig(ind1))/2;

figure(2)
plot(w);
title('Details');
xlabel("Echantillons");
ylabel("Amplitude");
```

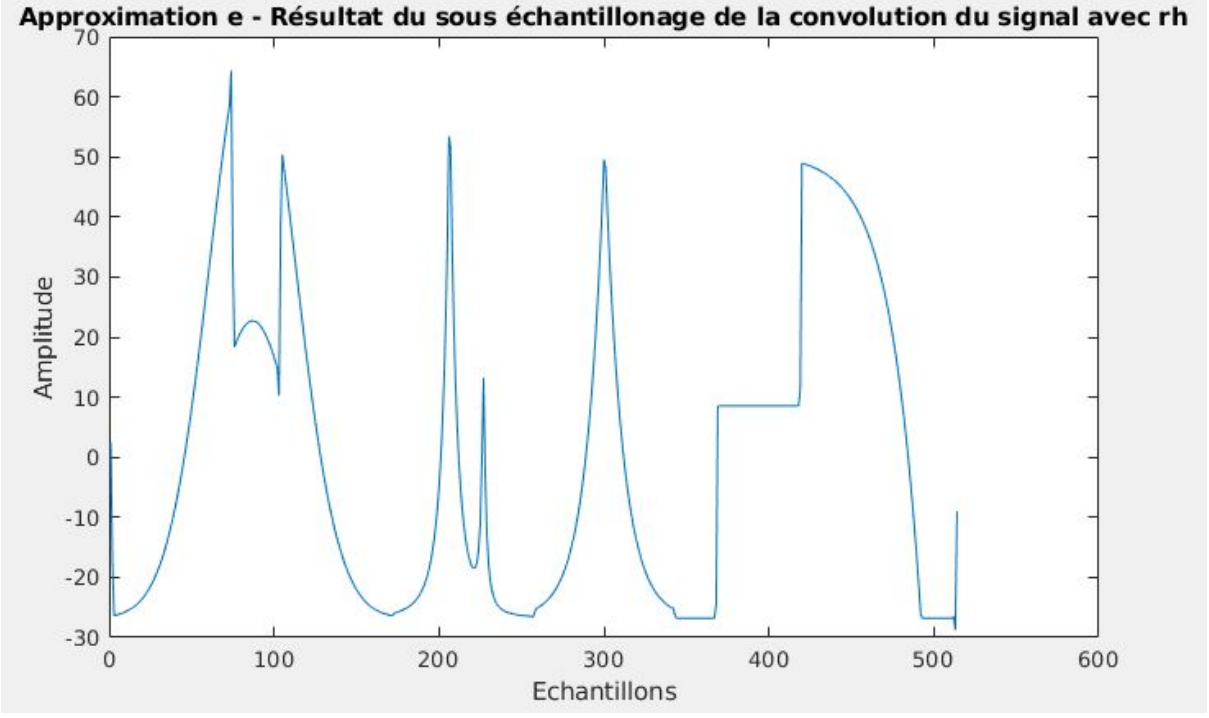


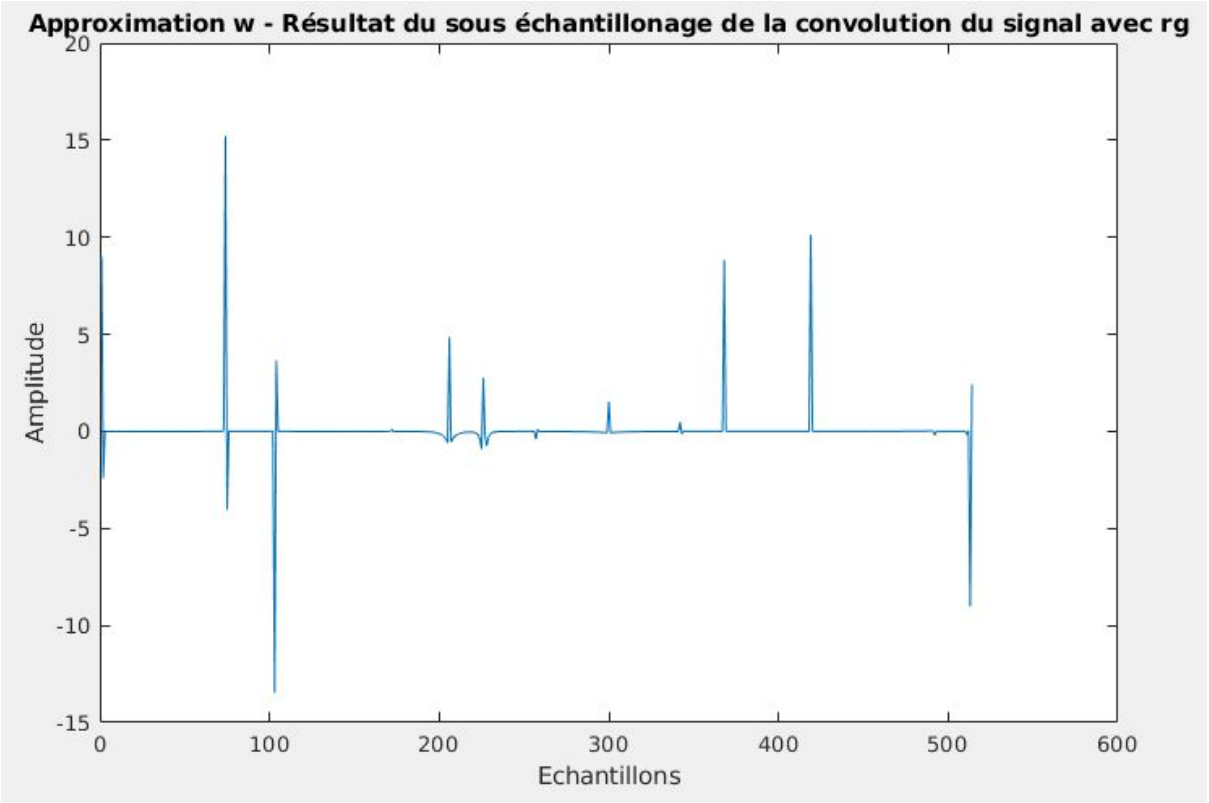


2-2 Convolution

Nous allons implémenter une cascade simple par un banc de filtre de Daubechies.
En chargeant le banc de filtre, on charge les variables h, g, rh et rg.
Ces variables correspondent aux filtres d'analyse et de reconstruction.
On implémente ensuite le schéma fonctionnel de décomposition (cf figure 2) :

```
% Convolution
load Daub4.mat

conv1 = conv(sig,rh);
e1 = conv1(1:2:end);

figure(3)
plot(e1);
title("Approximation e - Résultat du sous échantillonage de la convolution du signal avec rh");
xlabel("Echantillons");
ylabel("Amplitude");

conv2 = conv(sig,rg);
w1 = conv2(1:2:end);

figure(4)
plot(w1);
title("Approximation w - Résultat du sous échantillonage de la convolution du signal avec rg");
xlabel("Echantillons");
ylabel("Amplitude");
```



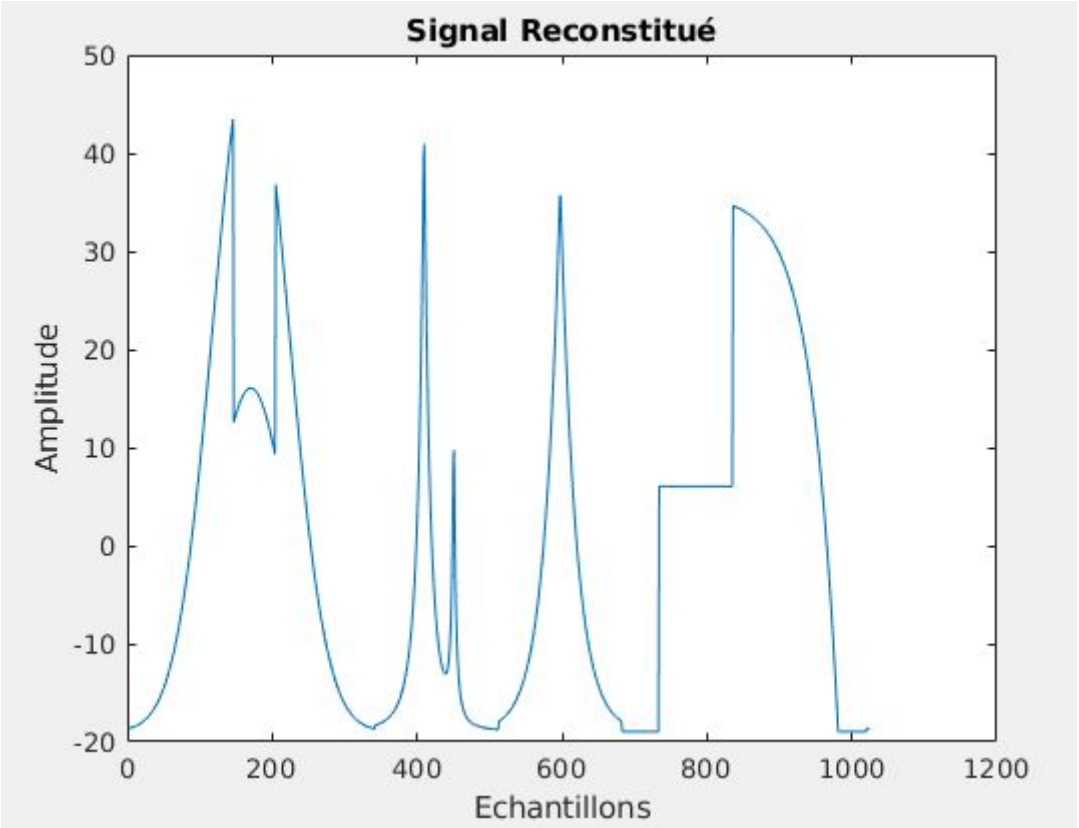


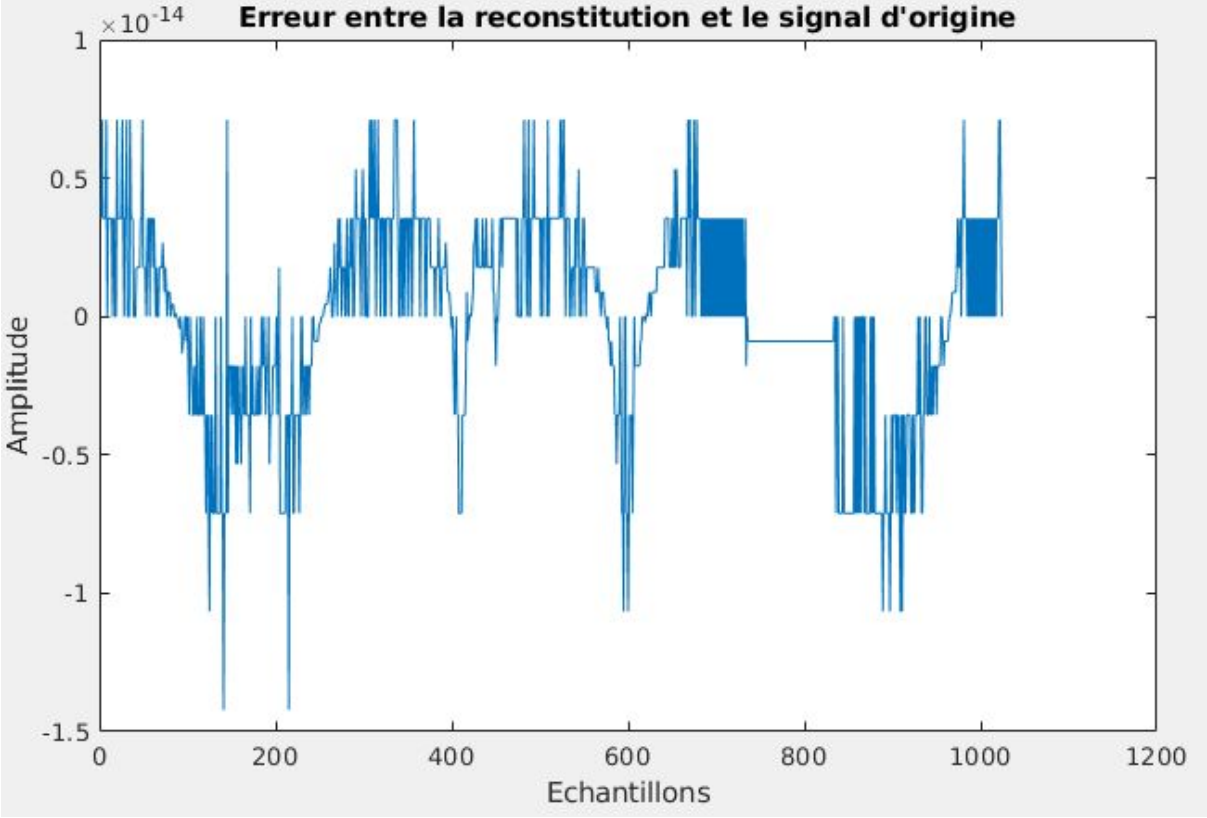
On remarque que les valeurs de e1 et w1 diffèrent des valeurs de e et w, mais l'allure est similaire : on retrouve les pics aux mêmes nombre d'échantillons.
On remarque aussi que l'écart pour les valeurs aux bords est beaucoup plus important. Cela s'explique par le fait que la compression est à pertes.

2-3 Reconstitution

Pour reconstituer le signal compressé, il faut effectuer un sur-échantillonnage par insertion de 0. On réalisera ensuite une convolution.
Les filtres h et rh sont de support [0,3], le support du résultat après application des filtres est donc [-3,1026]. Il faut donc éliminer 3 valeurs à gauche et à droite pour retrouver sig.

```
% Reconstruction
rec1 = zeros(1,1027);

for i=1:1027
    if mod(i,2) == 1
        rec1(1,i) = e1(1,(i+1)/2);
    else
        rec1(1,i) = 0;
    end
end

for i=1:1027
    if mod(i,2)==1
        rec2(1,i) = w1(1,(i+1)/2);
    else
        rec2(1,i) = 0;
    end
end

sigh = conv(rec1,h);
sigg = conv(rec2,g);

sig2 = sigh + sigg;
sig2 = sig2(4:length(sig2)-3);

figure(5)
plot(sig2)
title('Signal Reconstitué');
xlabel("Echantillons");
ylabel("Amplitude");

erreur = sig - sig2;

figure(6)
plot(erreur)
title("Erreur entre la reconstitution et le signal d'origine");
xlabel("Echantillons");
ylabel("Amplitude");
```

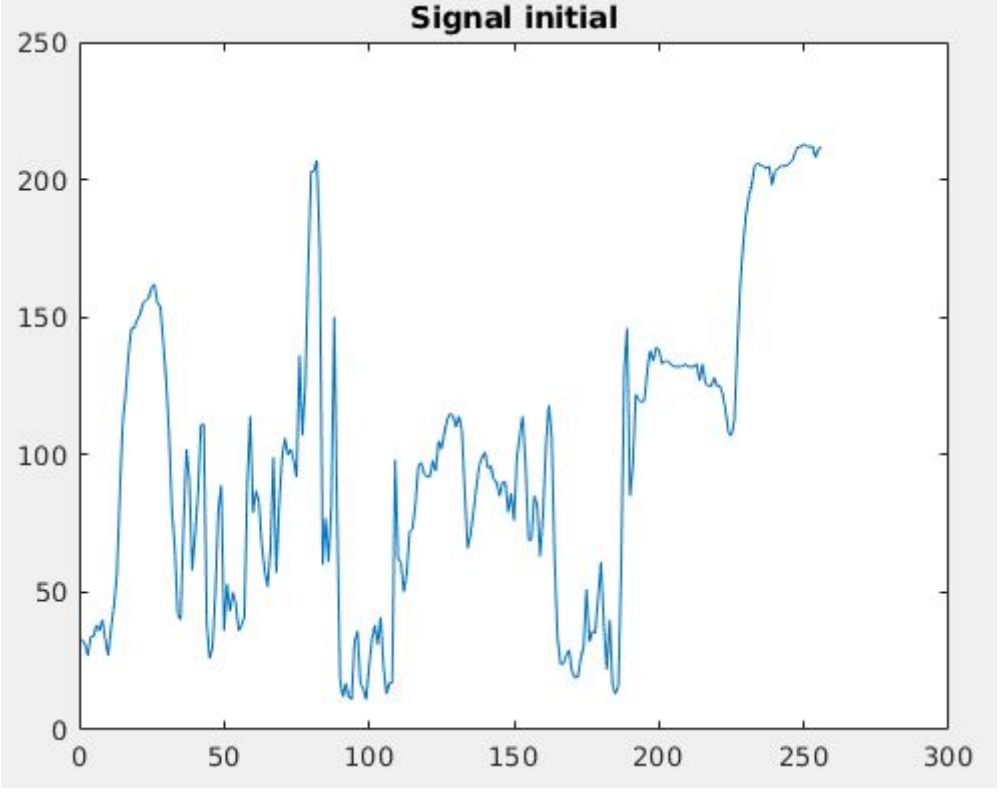


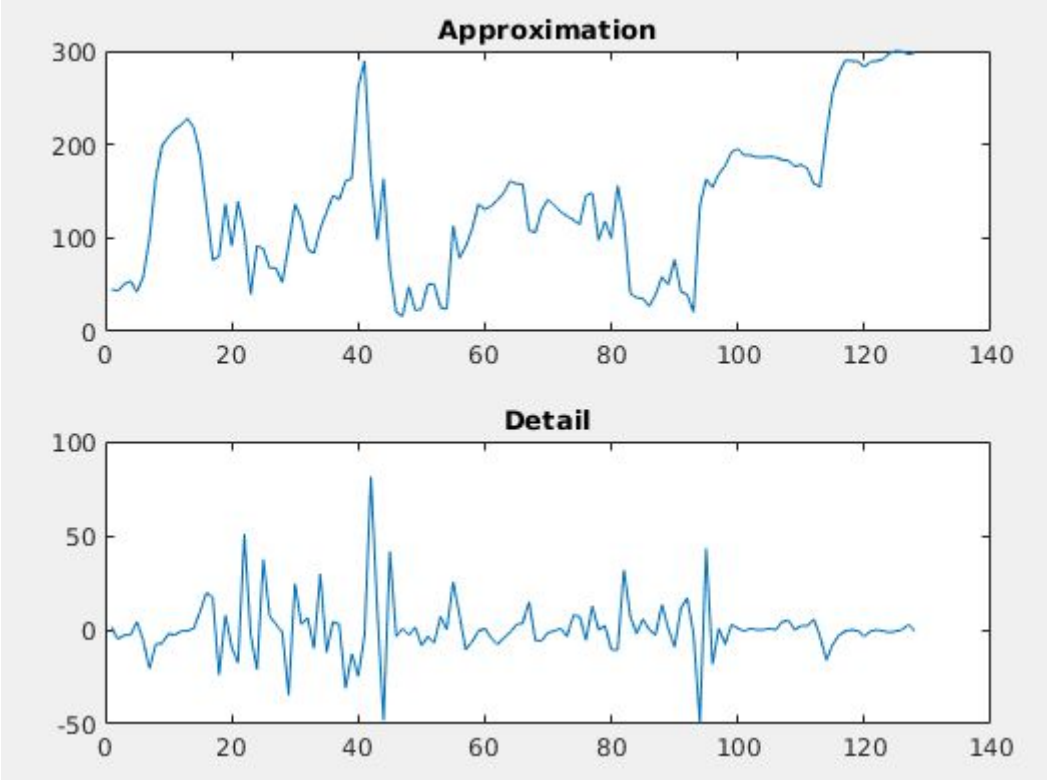


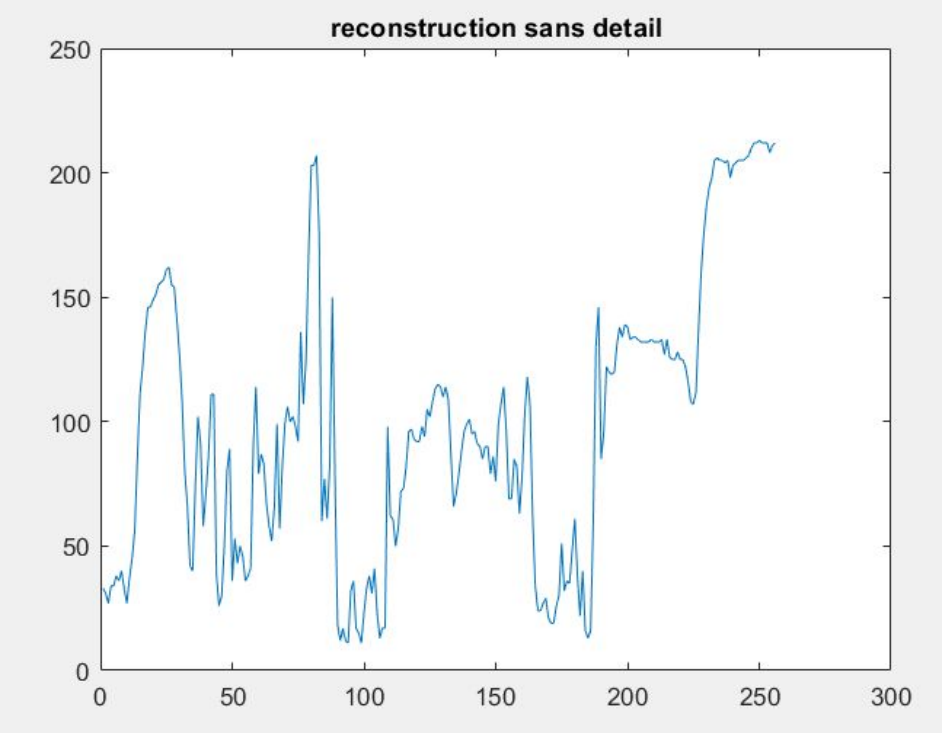
En regardant le signal reconstitué on se doute que nous avons subi très peu de pertes.
En effet, l'erreur est de l'ordre du 10^-14.

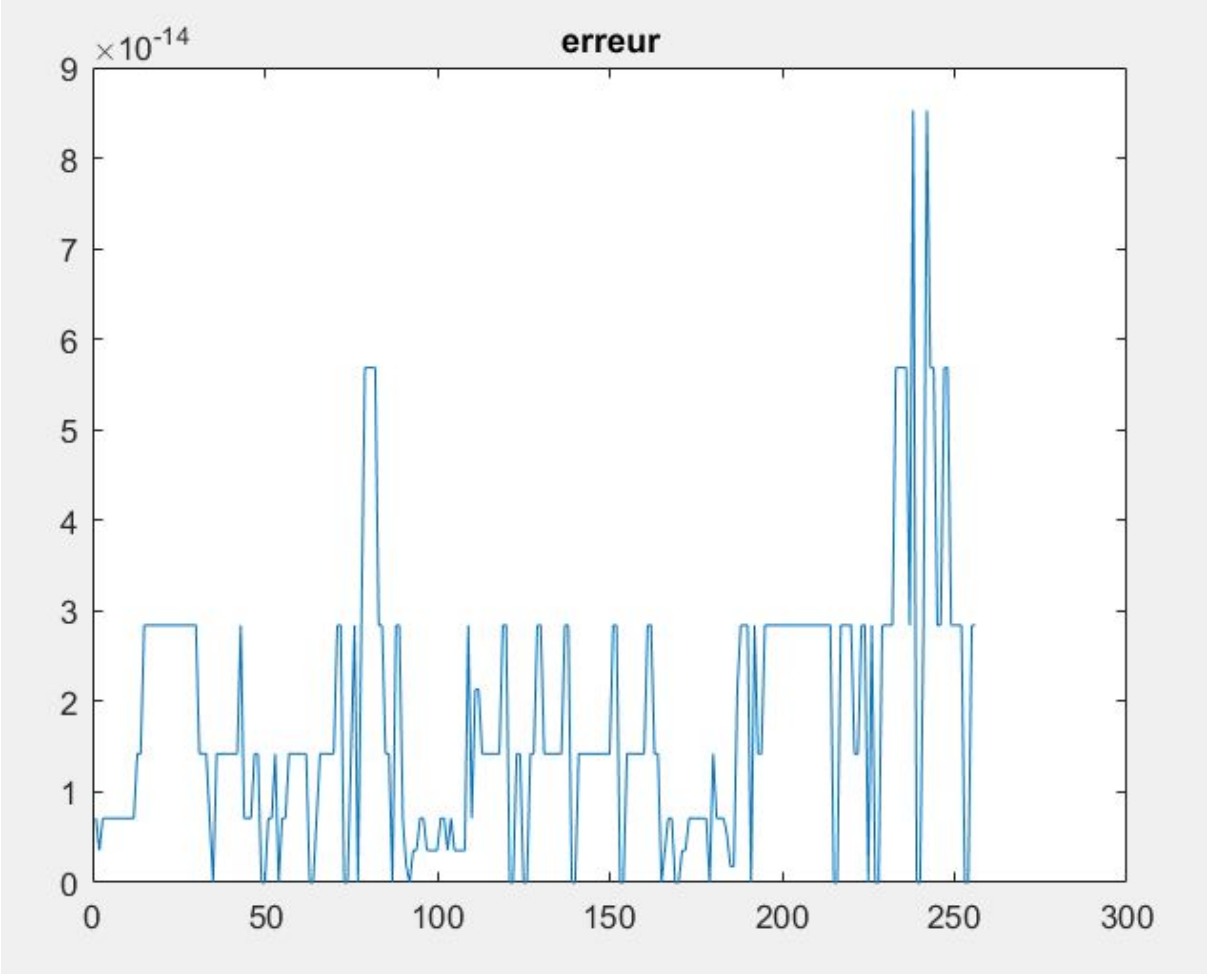
3- Compression des signaux 1D

```
%% Compression des signaux 1D
%% Ondelettes orthogonales %%
[rh, rg, h, g] = GetFiltres('Haar');
x = SignauxTypiques('ligneLena', 256);

figure(1);
plot(x);
title('Signal initial');

% pour les ondelettes orthogonales (longueur paire), utiliser type
% ='a_o' ou 'd_o' (a: approximation, d: detail, o: orthogonales
approximation = DownFilter(x,rh,'a_o');
detail = DownFilter(x,rg,'d_o');

% pour les ondelettes orthogonales (longueur paire), utiliser type
% ='a_o' ou 'd_o'
approximationreconstruit = UpFilter(approximation,h,'a_o');
detailreconstruit = UpFilter(detail,g,'d_o');
xreconstruit = approximationreconstruit + detailreconstruit;

x2=approximationreconstruit+detailreconstruit;
```



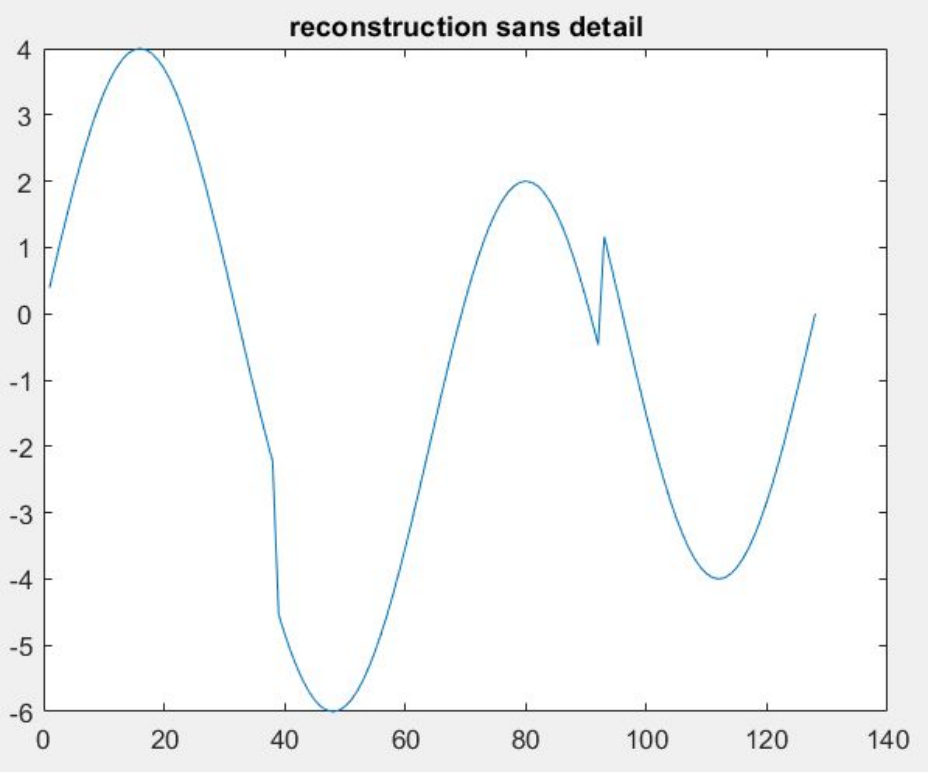

Approximation
300
200
100
0
0
20
40
60
80
100
120
140
Detail
100
50
0
-50
0
20
40
60
80
100
120
140

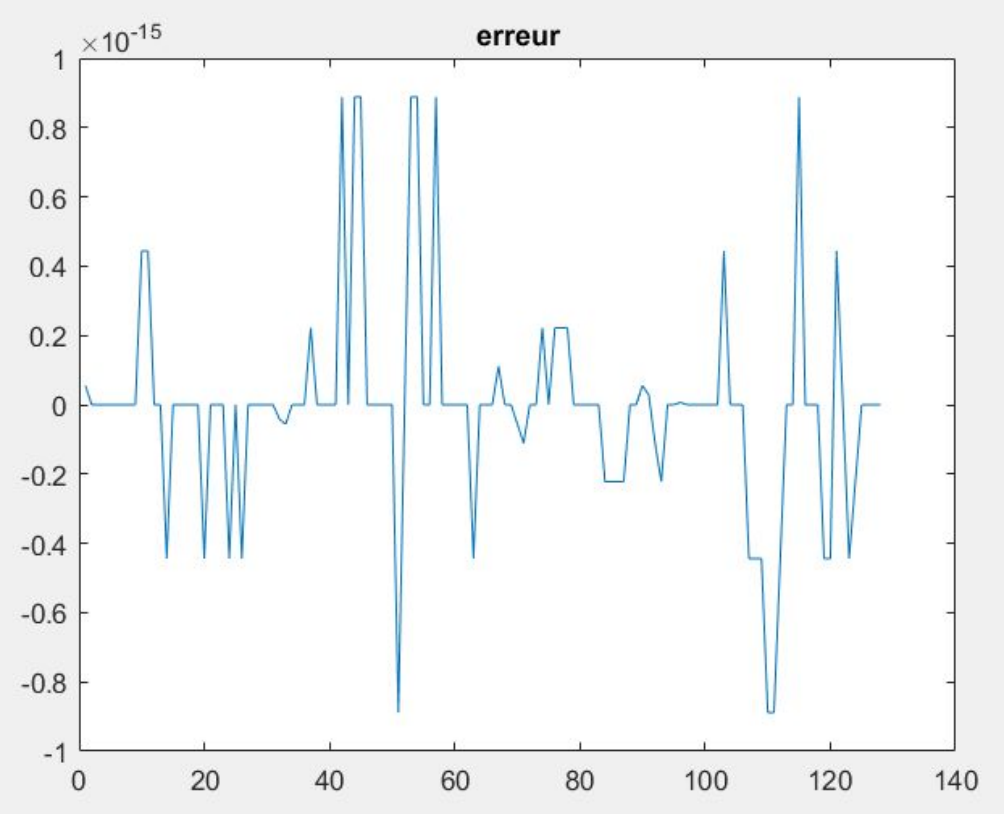

reconstruction sans detail
250
200
150
100
50
0
0
50
100
150
200
250
300



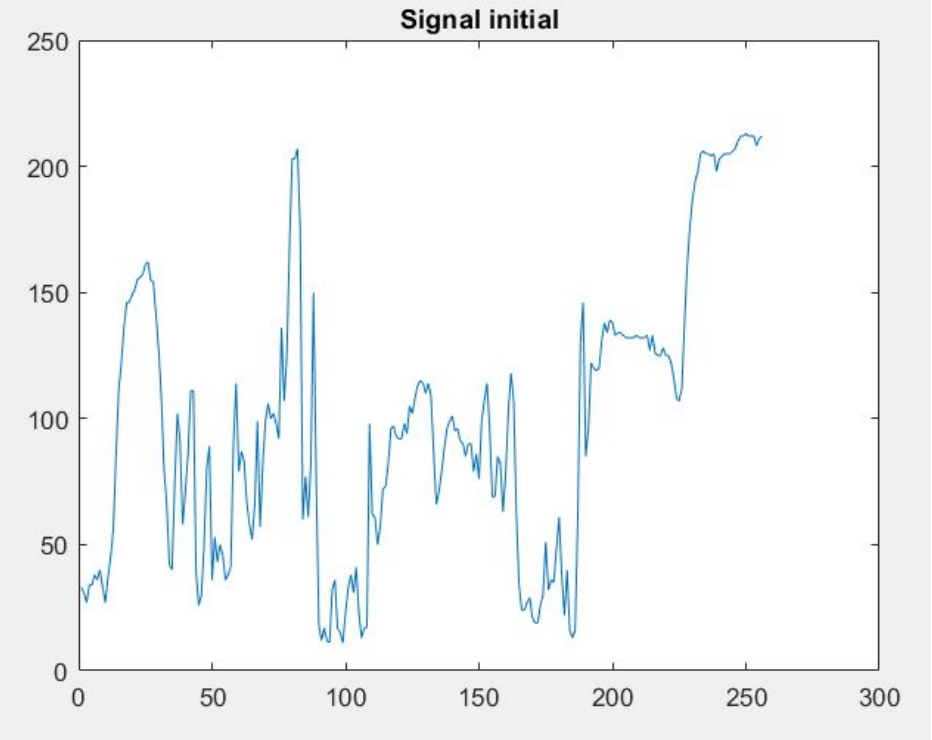
L'erreur observé étant de l'ordre de 10^-14, on peut considérer que la reconstruction est bien parfaite.

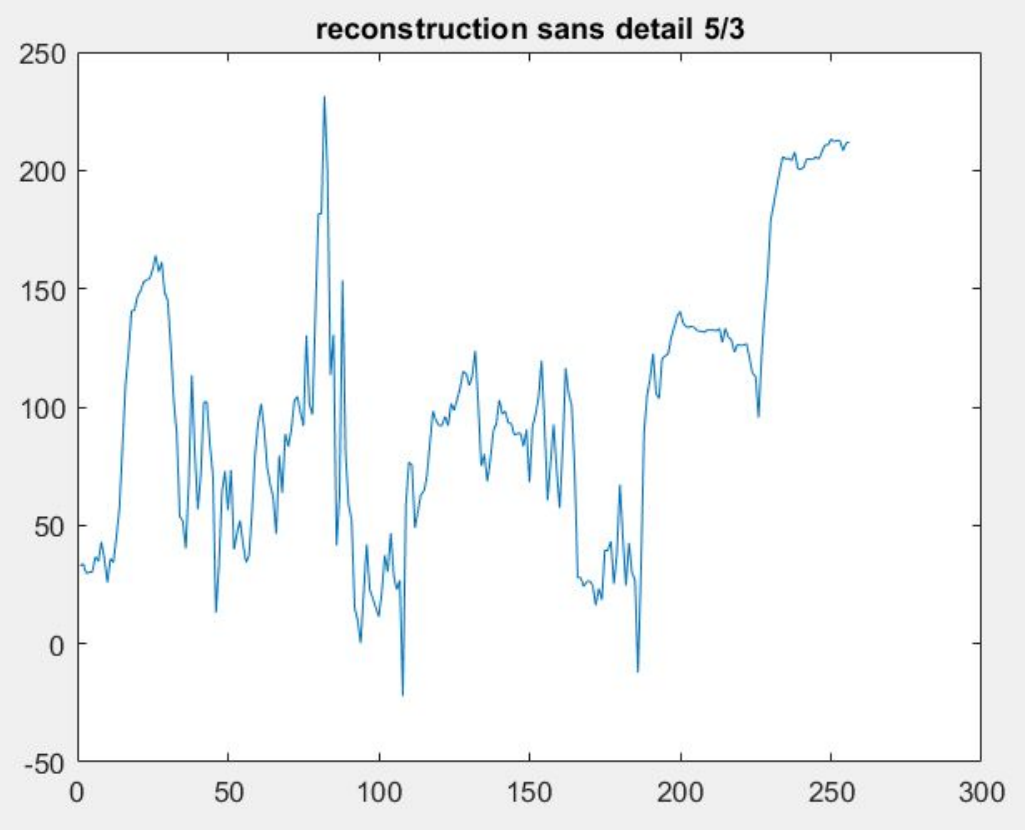
```
%% Ondelettes biorthogonales %% %%

[rh, rg, h, g] = GetFiltres('5/3');

x = SignauxTypiques('sinusDiscontinu',128);

% pour les ondelettes biorthogonales (longueur impaire),
% utiliser type='a_b' ou 'd_b' (a: approximation, d: detail, b: biorthogonales
approximation = DownFilter(x,rh,'a_b');
detail = DownFilter(x,rg,'d_b');

% pour les ondelettes biorthogonales (longueur impaire),
% utiliser type='a_b' ou 'd_b' (a: approximation, d: detail, b: biorthogonales
approximationreconstruit = UpFilter(approximation,h,'a_b');
detailreconstruit = UpFilter(detail,g,'d_b');

%TODO
x3=approximationreconstruit+detailreconstruit;
```

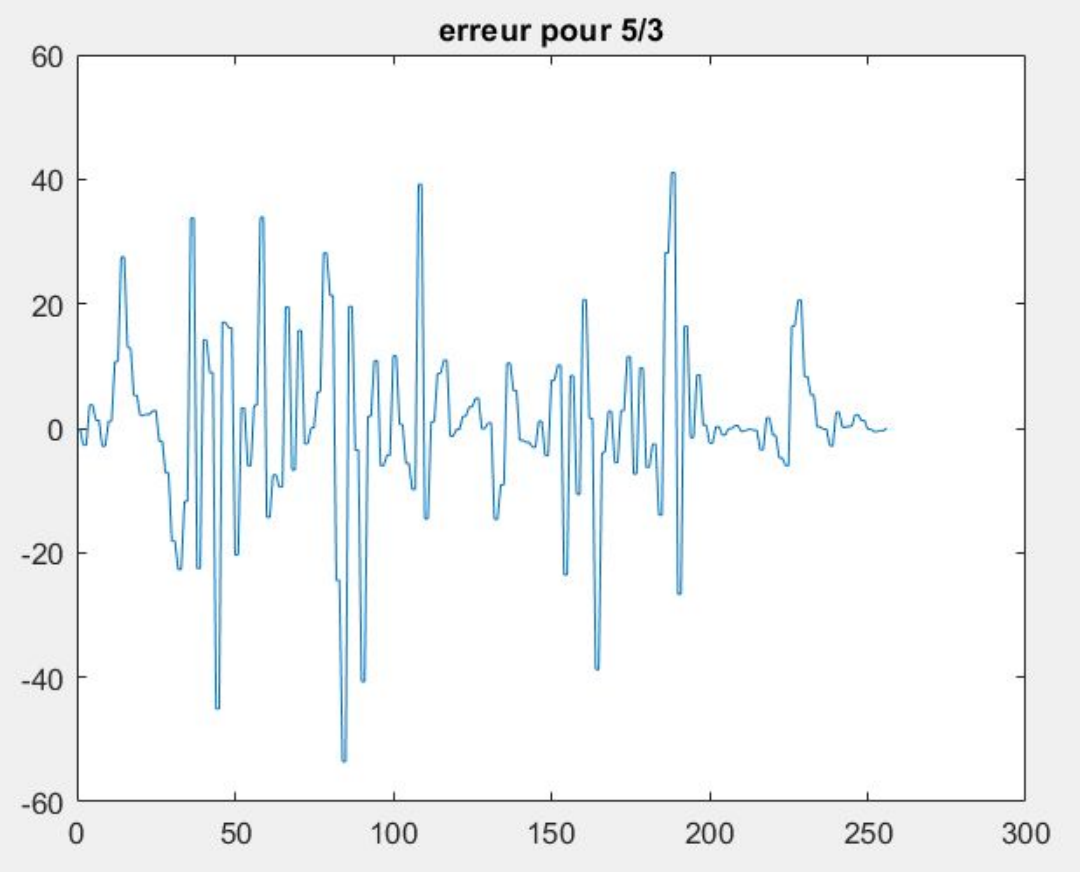

reconstruction sans detail
4
3
2
1
0
-1
-2
-3
-4
-5
-6
0
20
40
60
80
100
120
140


erreur
×10-15
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
0
20
40
60
80
100
120
140

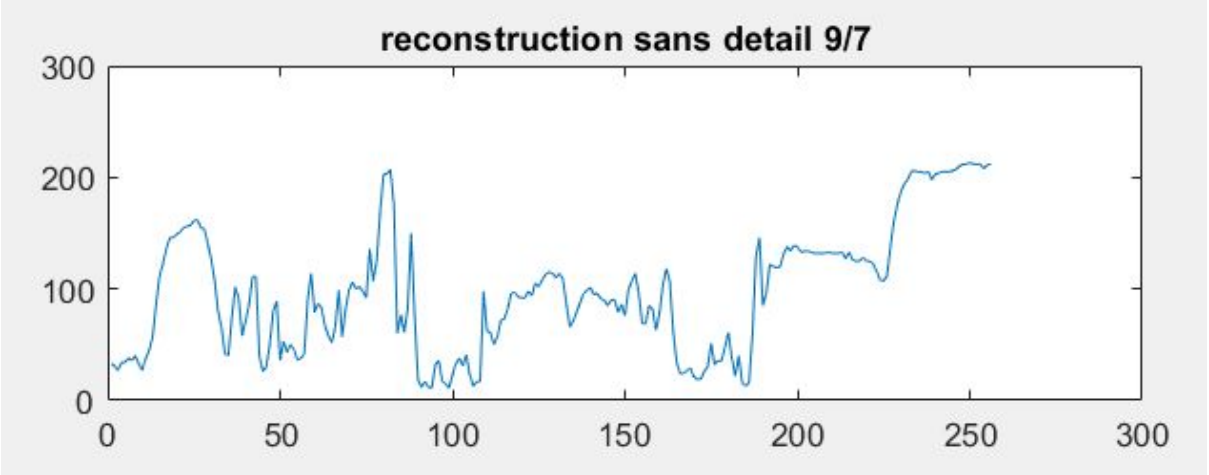
```
[rh, rg, h, g] = GetFiltres('5/3');
x = SignauxTypiques('ligneLena',64);

figure(1);
plot(x);
title('Signal initial');

% pour les ondelettes orthogonales (longueur paire), utiliser type
% ='a_o' ou 'd_o' (a: approximation, d: detail, o: orthogonales
approximation = DownFilter(x,rh,'a_b');
detail = DownFilter(x,rg,'d_b');

% pour les ondelettes orthogonales (longueur paire), utiliser type
% ='a_o' ou 'd_o'
approximationreconstruit = UpFilter(approximation,h,'a_o');
detailreconstruit = UpFilter(detail,g,'d_b');

%TODO
x2=approximationreconstruit+detailreconstruit;

figure(2)
plot(x2)
title('reconstruction sans detail 5/3');

erreur2=x-x2;
figure(3)
plot(erreur2)
title('erreur pour 5/3');
```

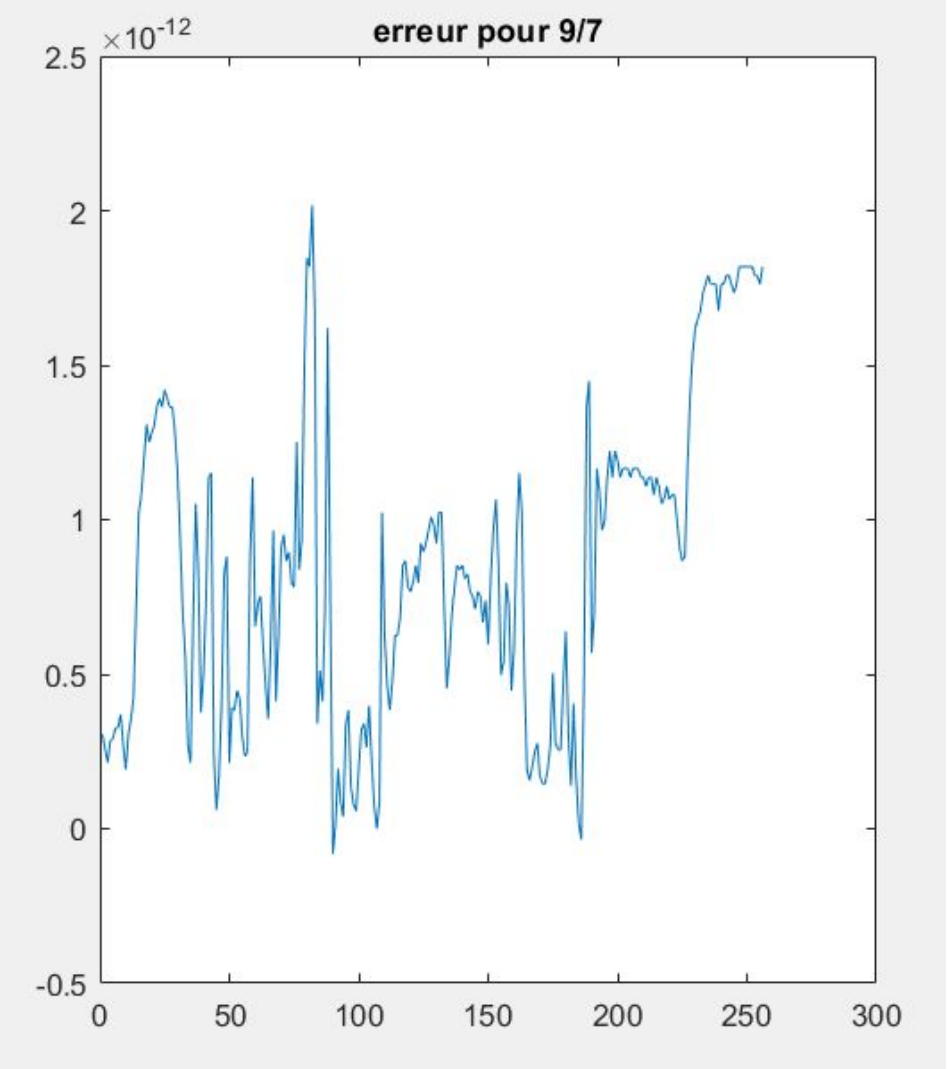



reconstruction sans detail 5/3
250
200
150
100
50
0
-50
0
50
100
150
200
250
300


erreur pour 5/3
60
40
20
0
-20
-40
-60
0
50
100
150
200
250
300

```
[rh, rg, h, g] = GetFiltres('9/7');

x = SignauxTypiques('ligneLena',64);

% pour les ondelettes biorthogonales (longueur impaire),
% utiliser type='a_b' ou 'd_b' (a: approximation, d: detail, b: biorthogonales
approximation = DownFilter(x,rh,'a_b');
detail = DownFilter(x,rg,'d_b');

% pour les ondelettes biorthogonales (longueur impaire),
% utiliser type='a_b' ou 'd_b' (a: approximation, d: detail, b: biorthogonales
approximationreconstruit = UpFilter(approximation,h,'a_b');
detailreconstruit = UpFilter(detail,g,'d_b');

%TODO
x3=approximationreconstruit+detailreconstruit;

figure(4)
plot(x3);
title('reconstruction sans detail 9/7');

figure(5)
erreur3=x-x3;
plot(erreur3)
title('erreur pour 9/7');
```





On remarque que l'erreur est bien plus faible pour 9/7 que pour 5/3, on utilisera donc dans la suite les ondelettes 9/7.

4- Compression d'une image

Pour reconstruire l'image, on va premièrement décomposer l'image : on crée l'image de détail et l'image d'approximation, en utilisant DownFilter.

```
%% Compression d'une image

Image=imread('lena.png');
[rh, rg, h, g] = GetFiltres('9/7');

for i=1:512
    approximation(i,:) = DownFilter(Image(i,:),rh,'a_b');
    detail(i,:) = DownFilter(Image(i,:),rg,'d_b');
end
for i=1:512
    approximationreconstruit(i,:) = UpFilter(approximation(i,:),h,'a_b');
    detailreconstruit(i,:) = UpFilter(detail(i,:),g,'d_b');
    x3(i,:)=approximationreconstruit(i,:)+detailreconstruit(i,:);
end
```

On a vu précédemment comment réaliser la compression d'un signal 1D. Pour compresser l'image, on décompose donc la matrice associée en lignes et en colonnes afin d'appliquer la compression sur chaque partie 1D constituant la matrice de l'image.

```
%% Compression d'une image sans détails diagonaux
%% Décomposition en lignes
Image=imread('lena.png');
[rh, rg, h, g] = GetFiltres('9/7');

   for i=1:512
       if i<257
           x(:,i)=approximation(:,i);
       else
           x(:,i)=detail(:,i-256);
       end
   end

%% Décomposition en Colonnes
y=transpose(x);

for i=1:512
       approximationc(i,:) = DownFilter(y(i,:),rh,'a_b');
       detailc(i,:) = DownFilter(y(i,:),rg,'d_b');
end

   for i=1:512
       if i<257
           z(:,i)=transpose(approximationc(:,i));
       else
           z(:,i)=transpose(detailc(:,i-256));
       end
   end
```

Une fois la compression réalisée sur chaque ligne et chaque colonne, on reconstitue l'image :

```
%% Reconstitution
z2=z(257:1:512,:);

for i=1:512
    if i<257
    z1(i,:)=z(i,:);
    end
end

for i=1:512
zz1(i,:) = UpFilter(transpose(z1(:,i)),h,'a_b');
zz2(i,:) = UpFilter(transpose(z2(:,i)),g,'d_b');
end

a1=reconstitution1(:,1:1:256);
a2=reconstitution1(:,257:1:512);

for i=1:512
aa1(i,:) = UpFilter(a1(i,:),h,'a_b');
aa2(i,:) = UpFilter(a2(i,:),g,'d_b');
end
```