

Transformée discrète en ondelettes : travaux pratiques

Yassine Jamoud, Samy Haffoudhi

29 décembre 2021

Introduction

Le but de ce TP est de prendre en main la transformée en ondelettes à l'aide de Matlab et de la boîte à outils *Wavelab*. Nous allons alors commencer par tracer des fonctions ondelettes, échelles et une transformée en ondelettes. Nous implémenterons ensuite une procédure de "débruitage" basée sur la transformée en ondelettes et enfin nous réaliserons de la compression d'images.

1 Tracé d'ondelettes et de fonctions échelles par DWT inverse

1. Sous forme informatique la DWT est représentée sous la forme :

$$\text{DWT}(x) = [a_J, d_J, d_{J-1}, \dots, d_1]$$

où J est l'échelle maximale.

Ainsi, $\forall j, k$,

- Le coefficient $a_J[k]$ est à l'indice k dans la représentation.
- Le coefficient $d_j[k]$ est à l'indice $\frac{N}{2^j} + k$

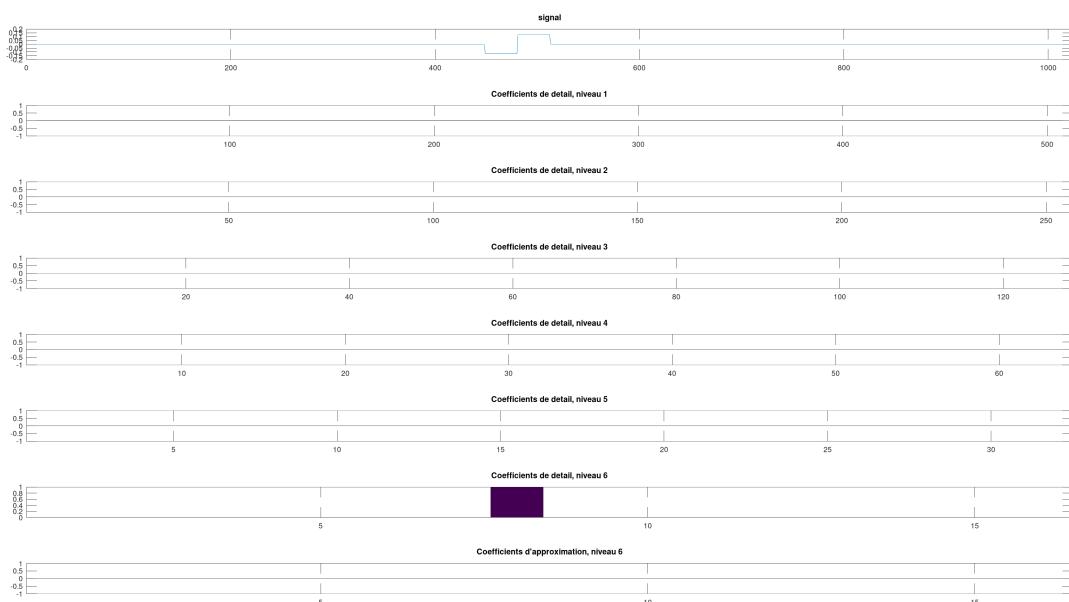
2. On souhaite construire un vecteur DWT x contenant seulement un coefficient non-nul de détail à la plus grande échelle et situé approximativement au milieu de l'axe temporel.
On place alors d'après 1. ce coefficient non-nul à l'indice : $\frac{N}{2^J} + \frac{N}{2^{J+1}} = \frac{3N}{2^{J+1}}$
3. Traçons alors l'allure de la fonction ondelette correspondant à la transformée de Haar :

FIGURE 1 – Fonction ondelette (Haar)



4. Représentons les fonctions échelles correspondantes :

FIGURE 2 – Fonctions échelles (Haar)



6. Répétons la procédure pour les ondelettes de Daubechies d'ordre 4 et 8 :

FIGURE 3 – Fonction ondelette (Daubechies d'ordre 4)



FIGURE 4 – Fonctions échelles (Daubechies d'ordre 4)

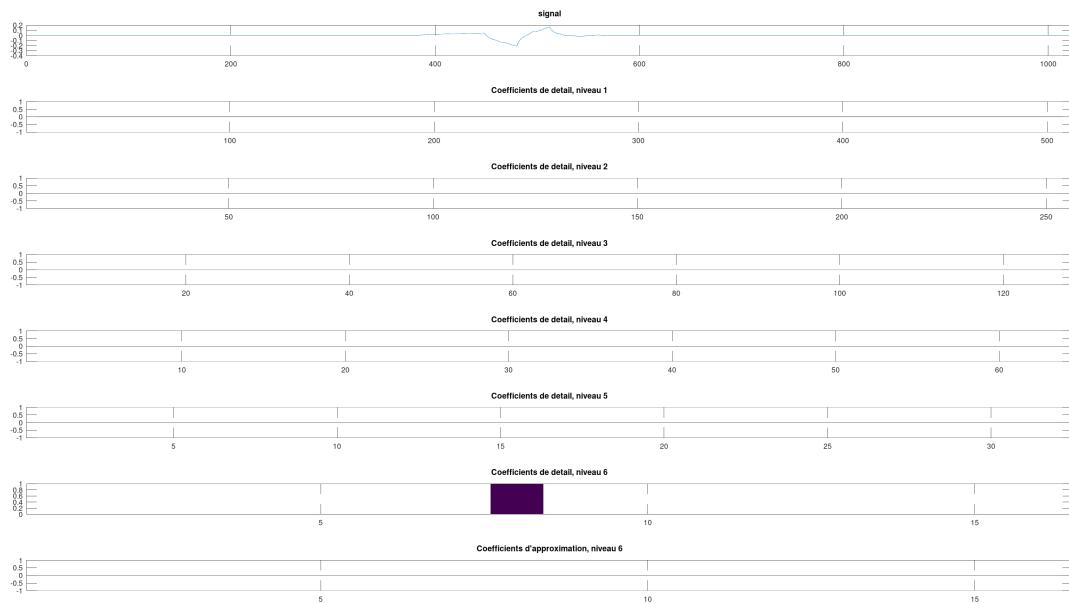


FIGURE 5 – Fonction ondelette (Daubechies d'ordre 8)

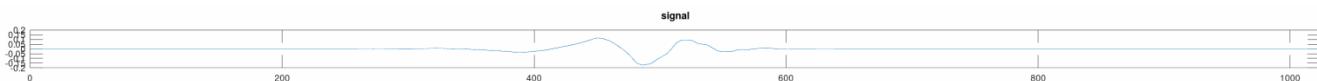
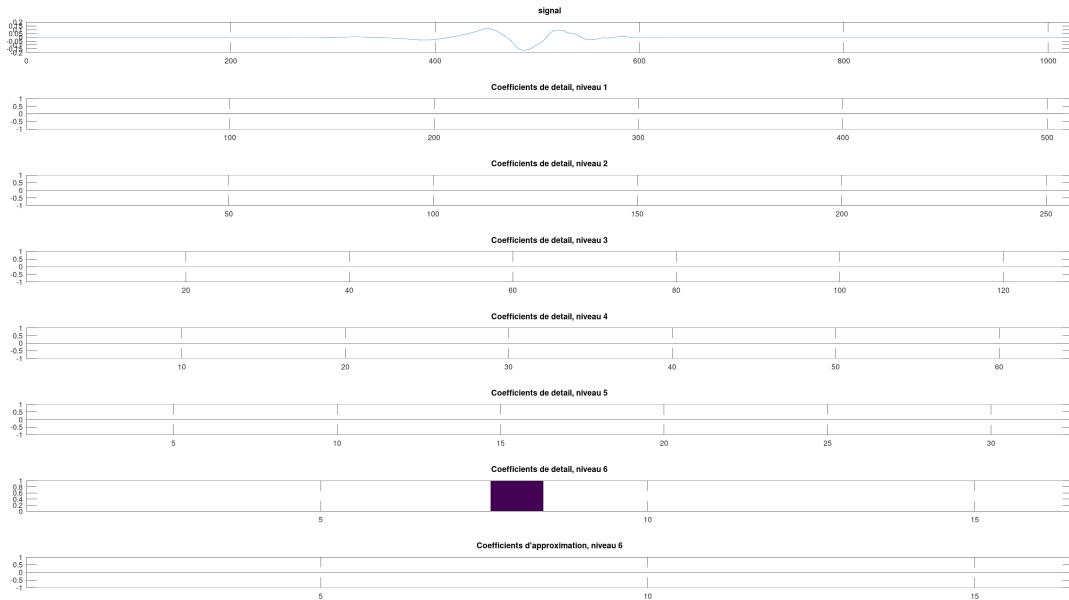


FIGURE 6 – Fonctions échelles (Daubechies d'ordre 8)



7. Ainsi, d'après les tracés précédents, la fonction ondelette correspondant à la transformée de Haar est constante par morceaux, celles de Daubechies sont plus complexes mais on retrouve les formes vues en cours, on note notamment la présence des pics.

2 Débruitage dans l'espace des ondelettes

On considèrera dans toute cette partie une ondelette de Haar.

1. Créons un signal dont la DWT jusqu'à l'échelle $J = 7$ contient uniquement quelques coefficients de détail non nuls.
Représentons alors le signal obtenu et sa DWT :

FIGURE 7 – Signal

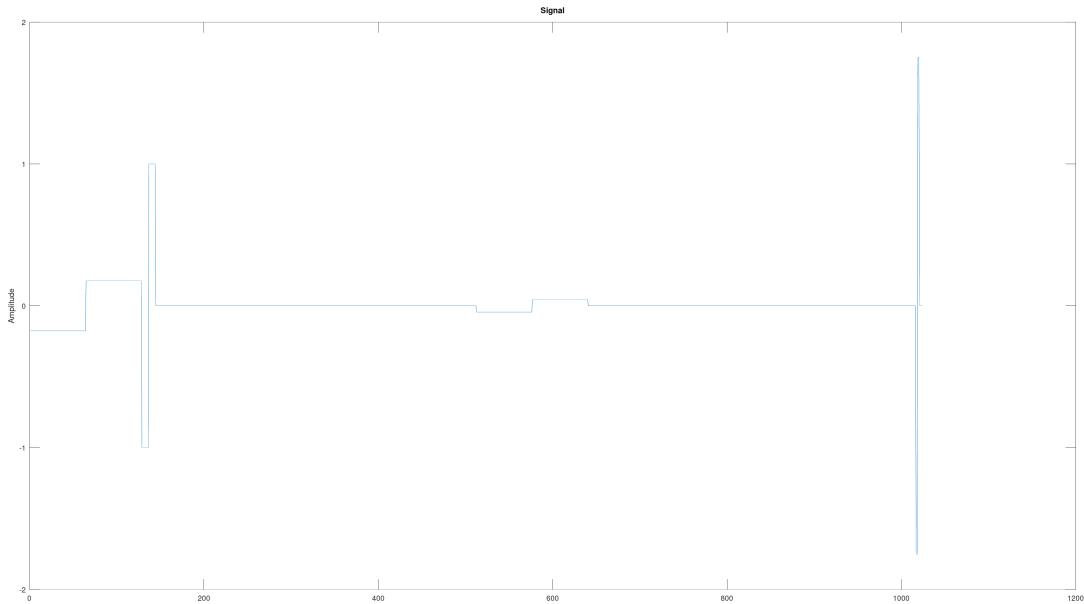
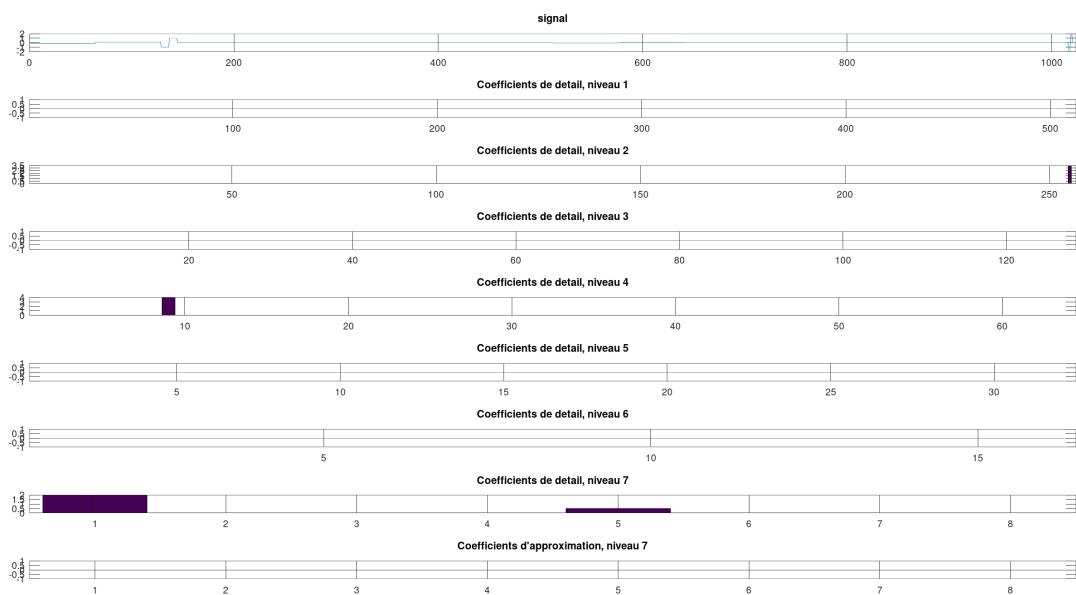


FIGURE 8 – DWT du signal



2. Ajoutons du bruit au signal obtenu et visualisons ce nouveau signal ainsi que sa DWT :

FIGURE 9 – Signal bruité

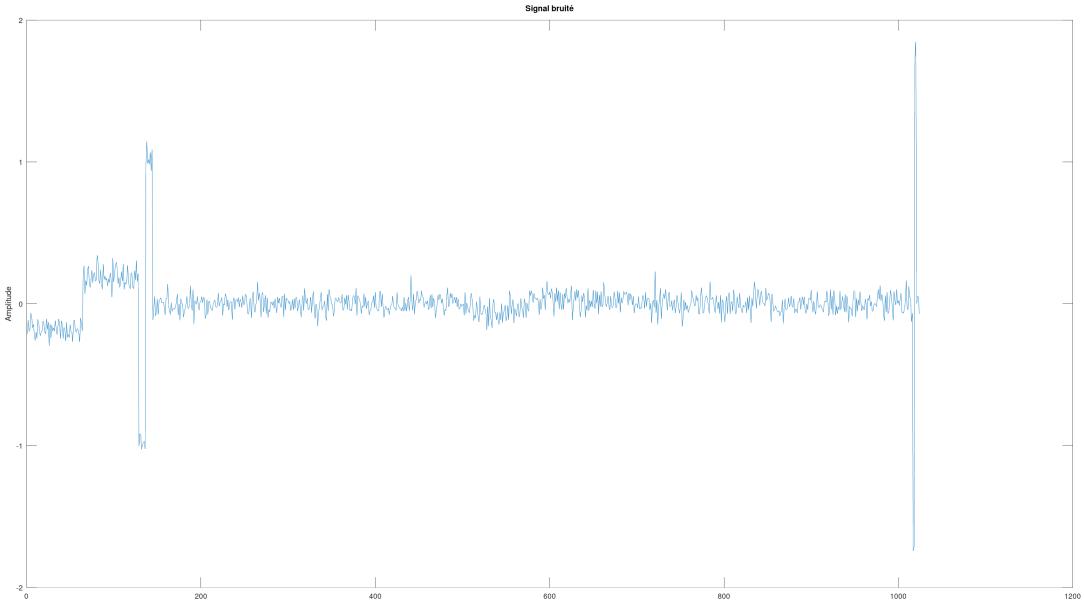
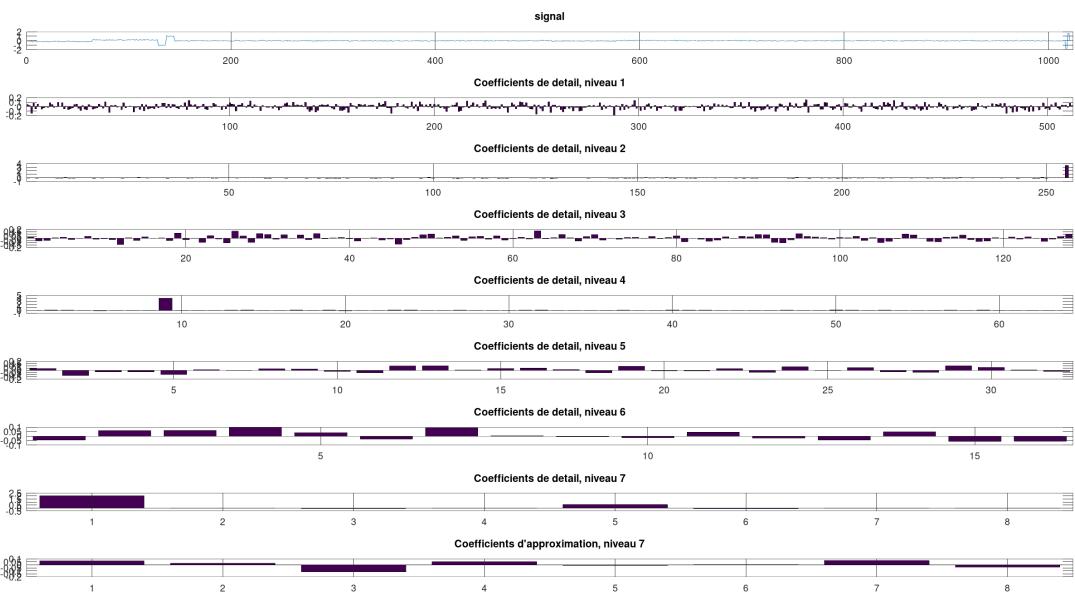


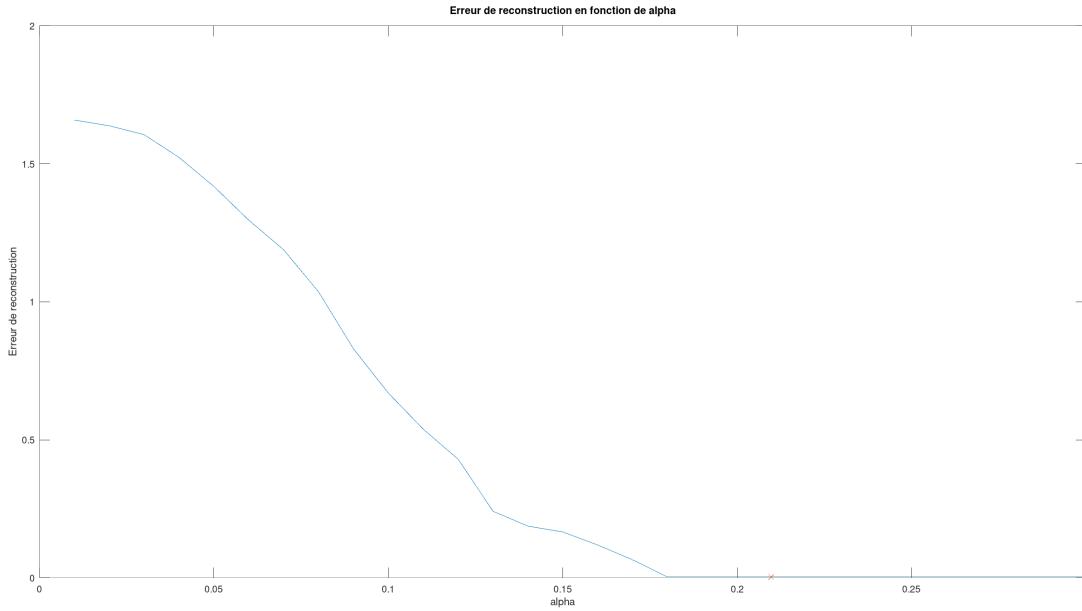
FIGURE 10 – DWT du signal bruité



On observe ainsi que l'ajout de bruit au signal résulte également en un ajout de bruit à sa DWT. Toutefois on remarque que les nouveaux coefficients non nuls liés au bruit sont d'amplitude faible devant celle des coefficients de la DWT du signal sans bruit. Un filtrage des coefficients à l'aide d'un seuil devrait permettre un débruitage du signal.

3. Mettons maintenant en œuvre le processus de débruitage décrit. Représentons alors l'erreur de construction commise en fonction de la valeur du seuil α et celle correspondant au seuil $\alpha^* = \sigma_b \sqrt{2 \ln(N)}$:

FIGURE 11 – Erreur de reconstruction en fonction de alpha



On obtient alors un seuil optimal correspondant à la valeur proposée.

4. On génère les signaux "Blocks" et "Doppler" à l'aide de la fonction `makesig`.
5. Représentons ces deux signaux ainsi que leur DWT :

FIGURE 12 – Singal "Blocks"

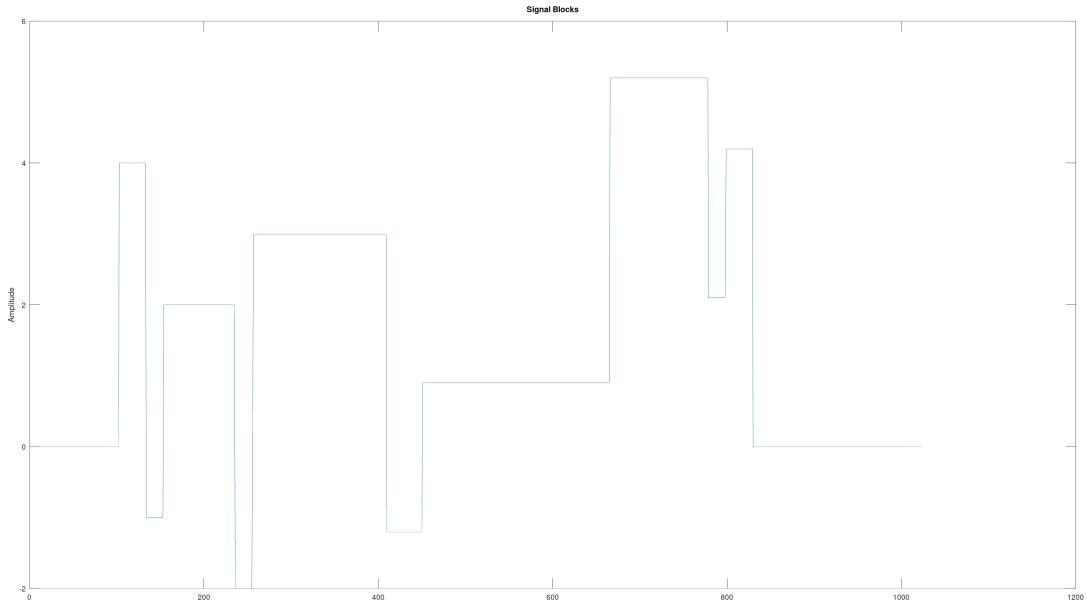


FIGURE 13 – Singal "Blocks"

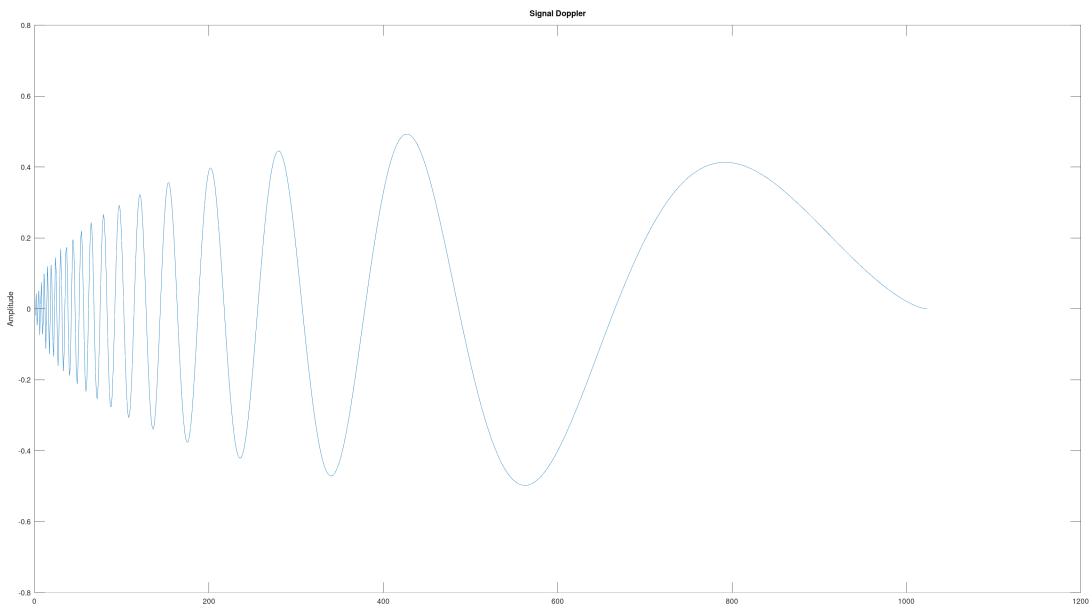


FIGURE 14 – DWT signal "Blocks"

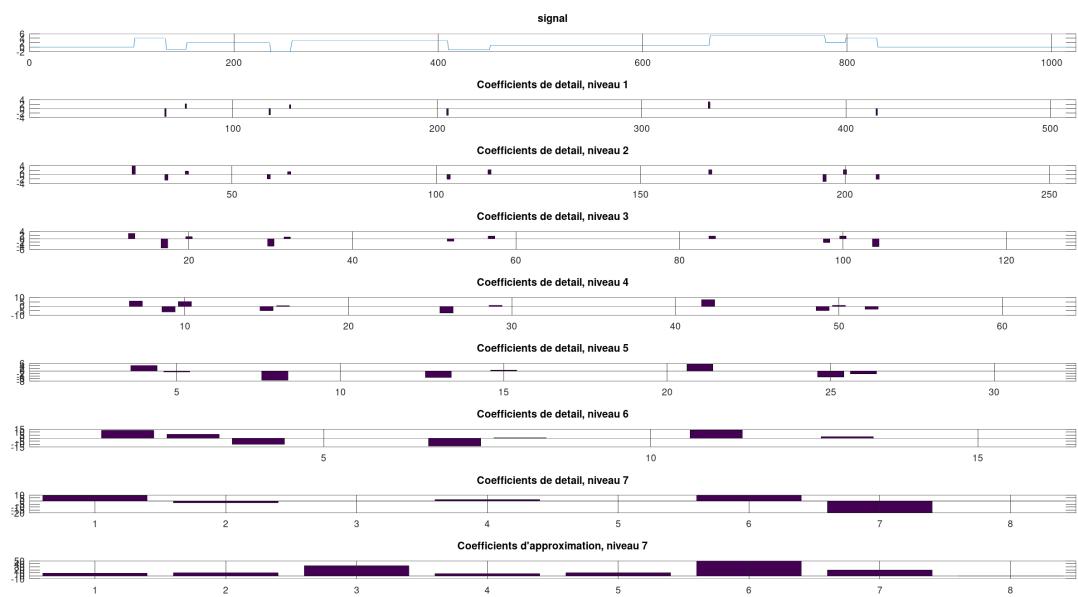
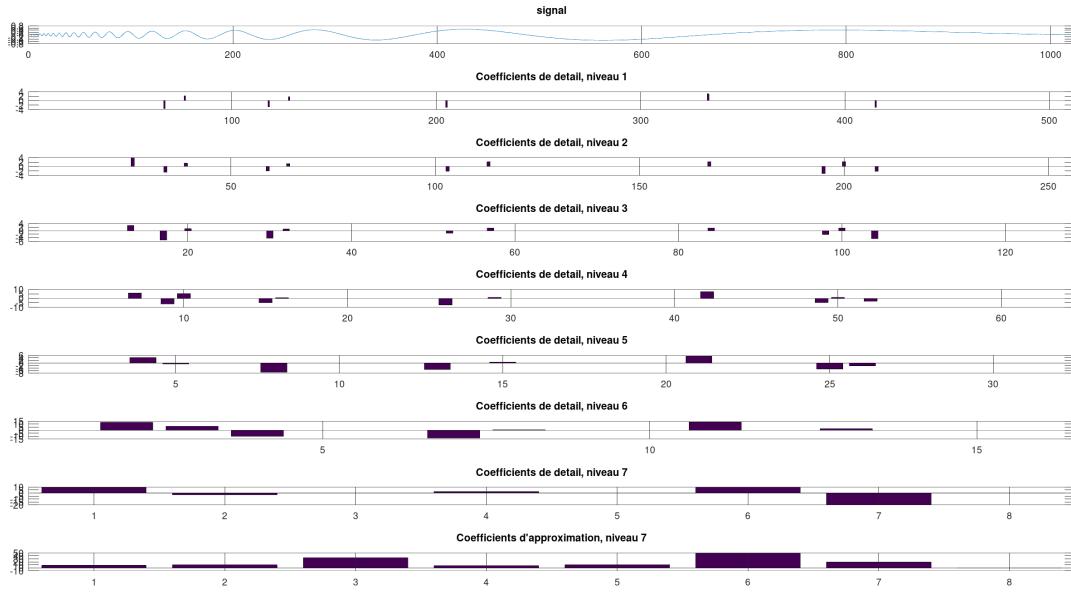


FIGURE 15 – DWT signal "Doppler"



Ces signaux ont bien des allures cohérentes avec leurs noms. Les DWT disposent de plus de coefficients non nuls que pour le signal généré précédemment

Ajoutons du bruit avec un RSD de 20 dB et appliquons la procédure précédente. On obtient les tracés d'erreur de reconstruction en fonction du seuil suivants :

FIGURE 16 – Erreur de reconstruction en fonction du seuil (Blocks)

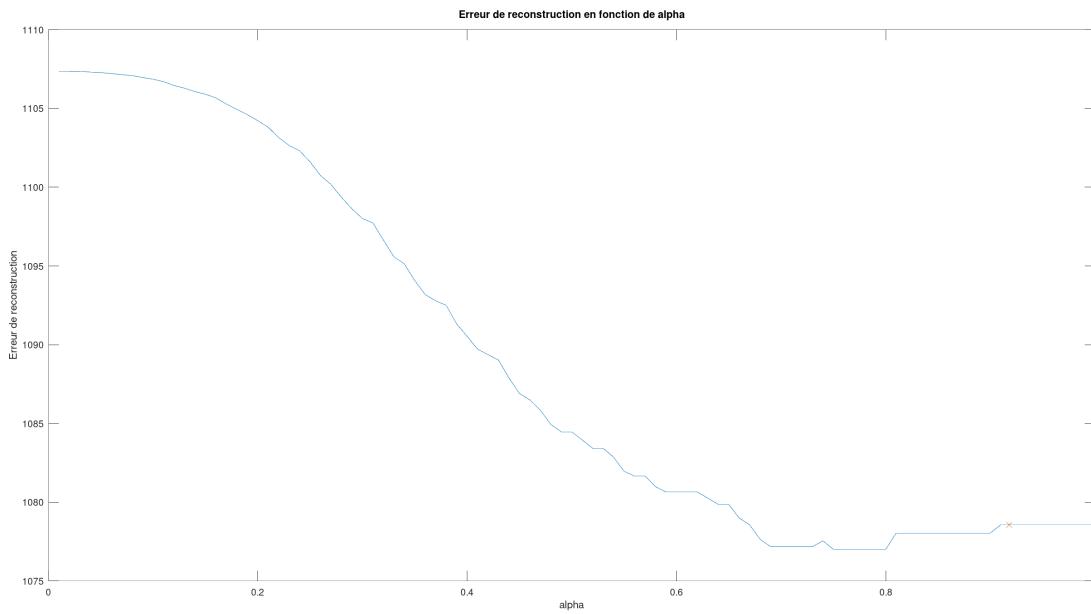
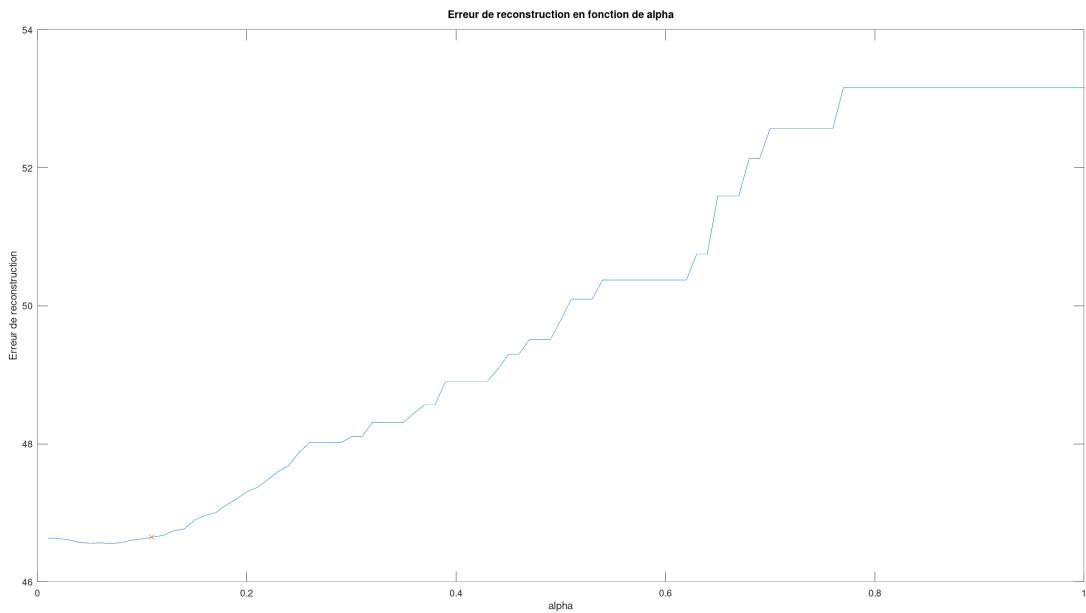


FIGURE 17 – Erreur de reconstruction en fonction du seuil (Doppler)



On obtient cette fois pour le signal "Blocks" une valeur de seuil optimale différente de celle proposée mais une valeur comparable dans le cas du signal "Doppler".

6. En utilisant maintenant une ondelette de Daubechies d'ordre 4, on obtient les tracés suivants :

FIGURE 18 – Erreur de reconstruction en fonction du seuil (Blocks)

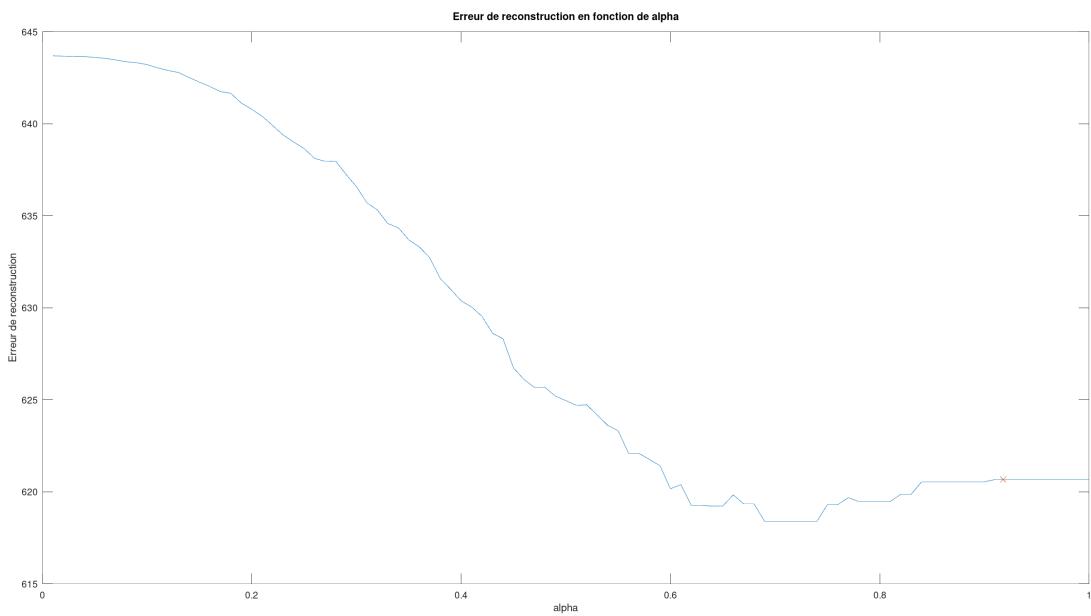
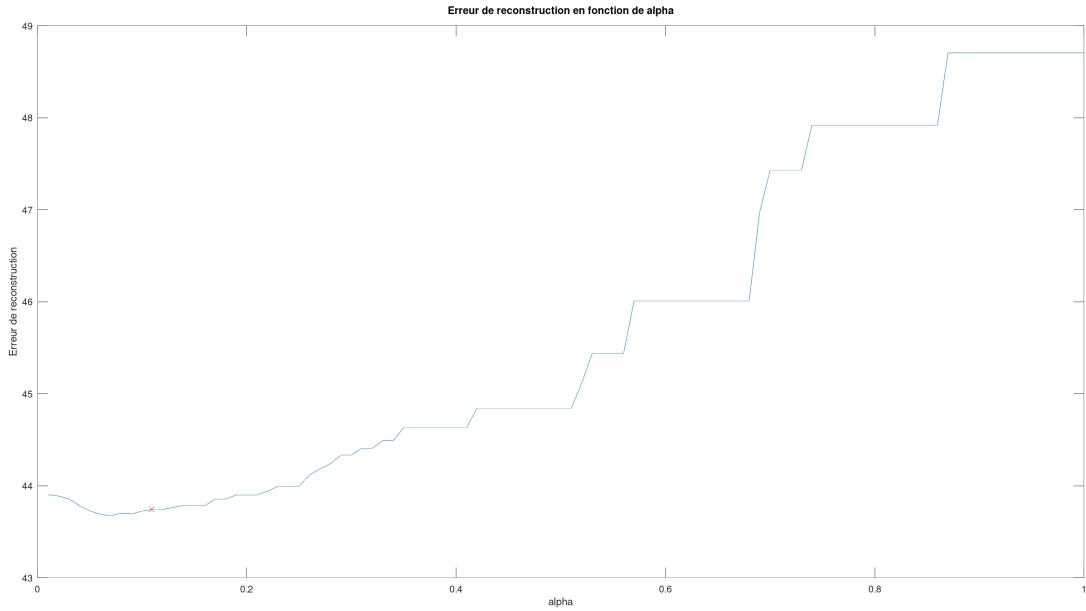


FIGURE 19 – Erreur de reconstruction en fonction du seuil (Doppler)

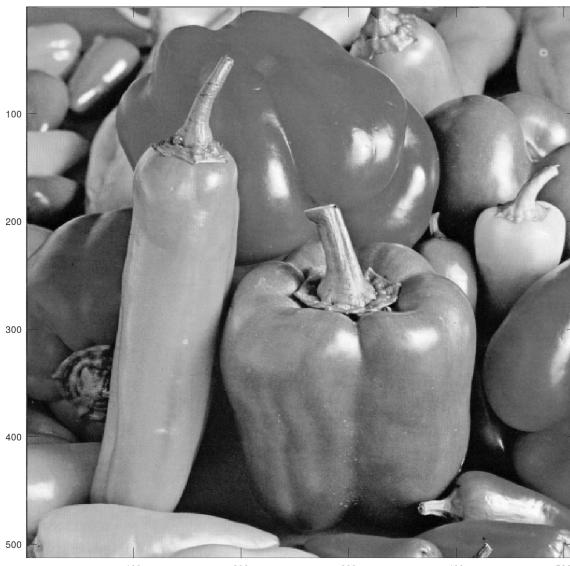


On remarque ainsi que pour ces deux signaux on obtient de meilleurs résultats en terme d'erreur de reconstruction. Cependant, il convient également de comparer la forme des signaux obtenus afin de vérifier que le signal reconstruit permet aussi de rendre compte des transitions brusques dans le cas du signal "Blocks" par exemple.

3 Compression d'images

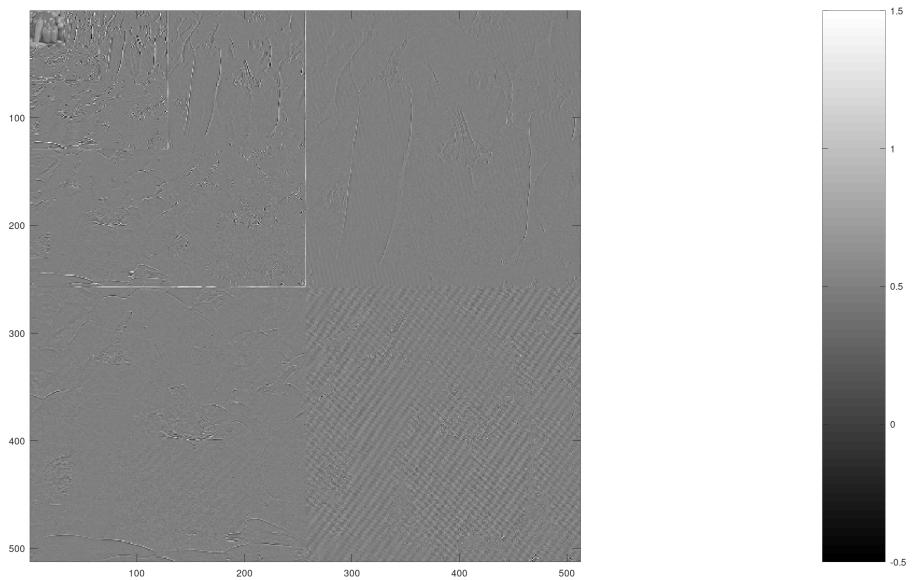
1. On ouvre l'image `peppers.tiff` à l'aide de la fonction `imread` de Matlab.
2. Visualisons cette image :

FIGURE 20 – Image `peppers.tiff`



3. On calcule sa transformée en ondelettes 2D à l'échelle $J = 4$
4. Visualisons alors cette DWT-2D :

FIGURE 21 – DWT-2D



5. Représentons l'histogramme des coefficients et traçons les par ordre décroissant en valeur absolue :

FIGURE 22 – Histogramme

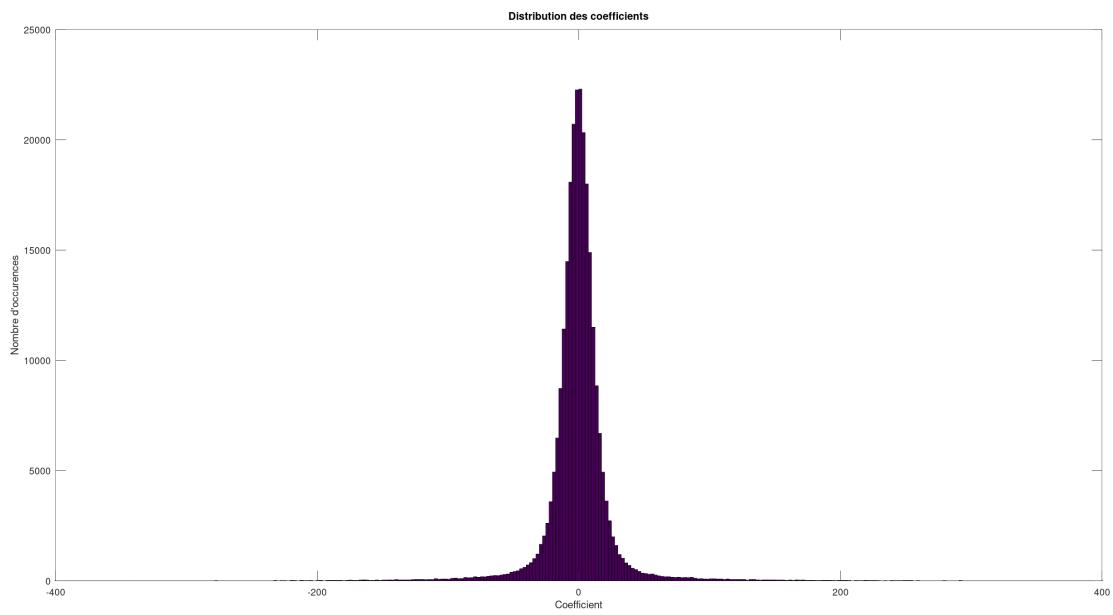
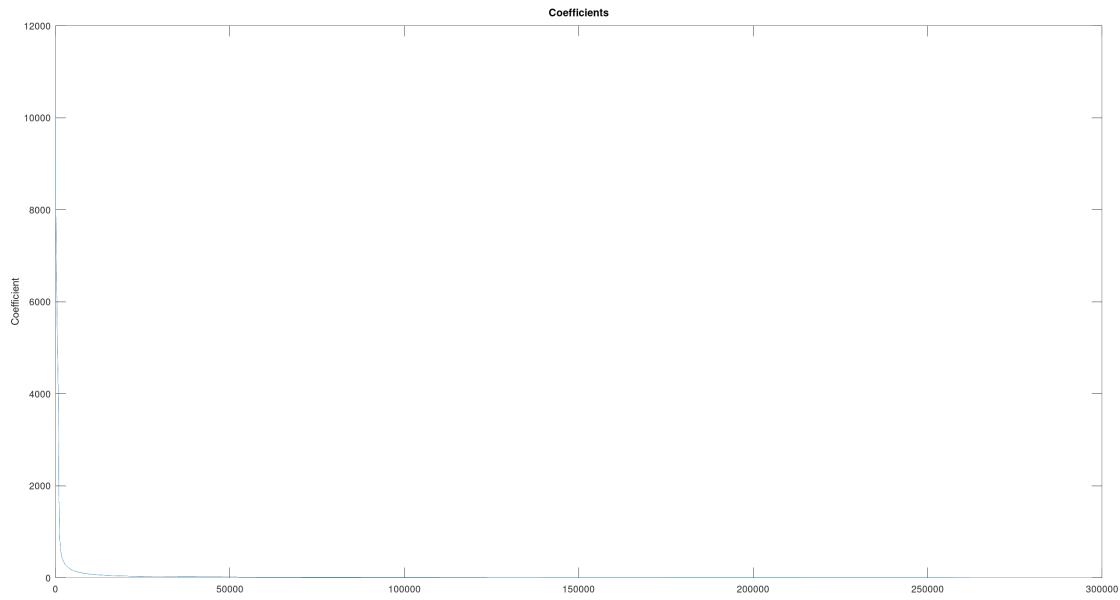


FIGURE 23 – coefficients (ordre décroissant)



On remarque que cet histogramme contient beaucoup plus de valeurs faibles que de valeurs élevées et s'apparente à une gaussienne de variance faible.

6. La procédure de compression consiste en la reconstruction de l'image à partir de sa DWT seuillée. On se donne un τ et on ne conserve que les $\tau\%$ coefficients les plus grands.
7. Visualisons l'image reconstruite pour $\tau \in \{1, 5, 20\}$.

FIGURE 24 – Résultats pour $\tau = 1\%$

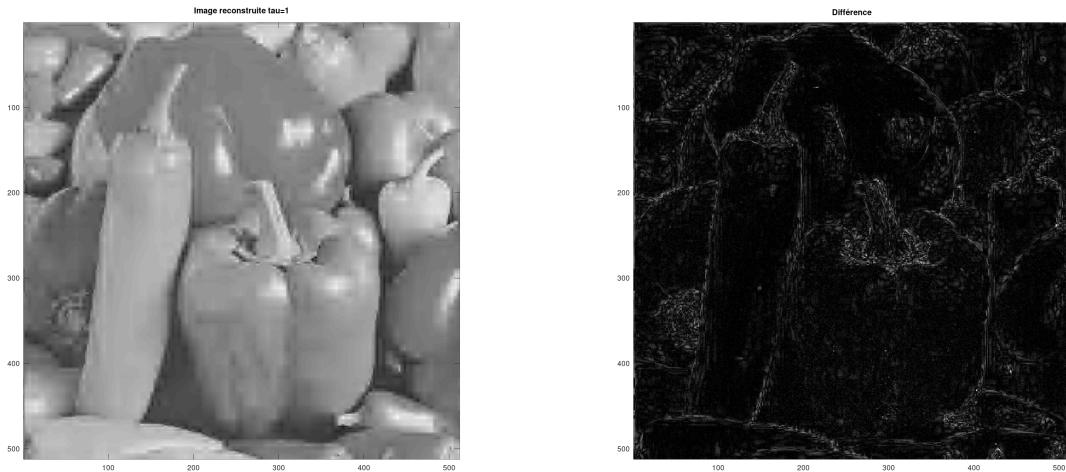


FIGURE 25 – Résultats pour $\tau = 5\%$

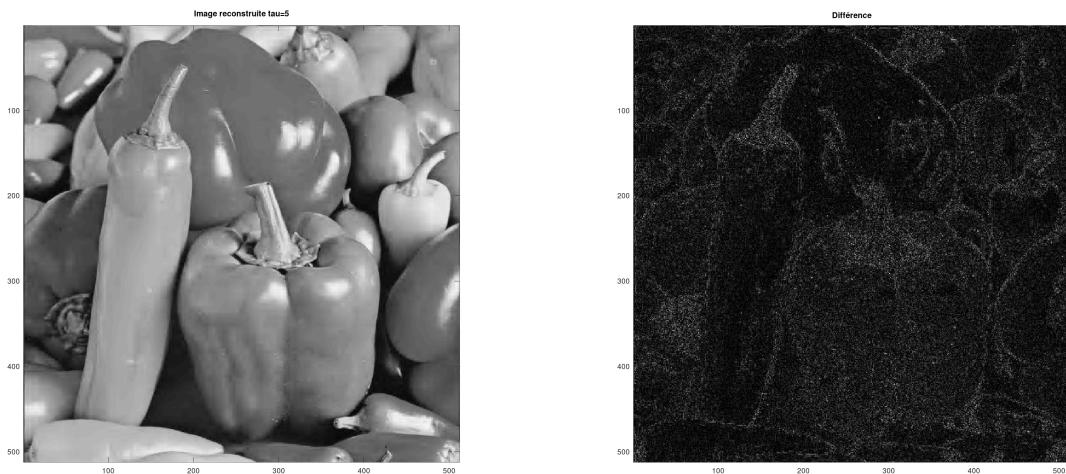
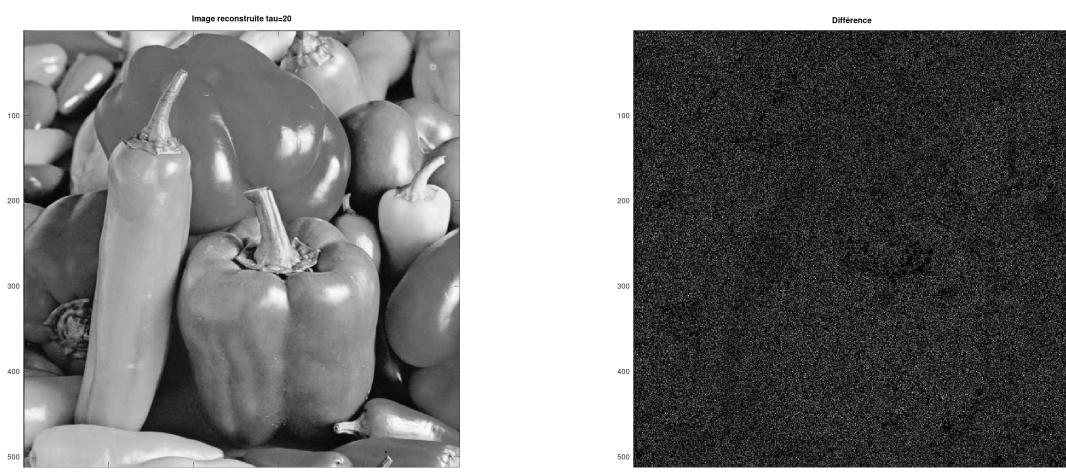


FIGURE 26 – Résultats pour $\tau = 20\%$



On remarque alors que cette procédure de compression permet déjà de reconnaître l'image avec $\tau = 1\%$. Pour $\tau = 20\%$, l'image "différence" n'est pas exactement nulle mais conserver par exemple 20% des coefficients suffit à obtenir une image quasiment identique à l'originale.

8. Avec une ondelette de Haar, on obtient les images suivantes :

FIGURE 27 – Résultats pour $\tau = 1\%$

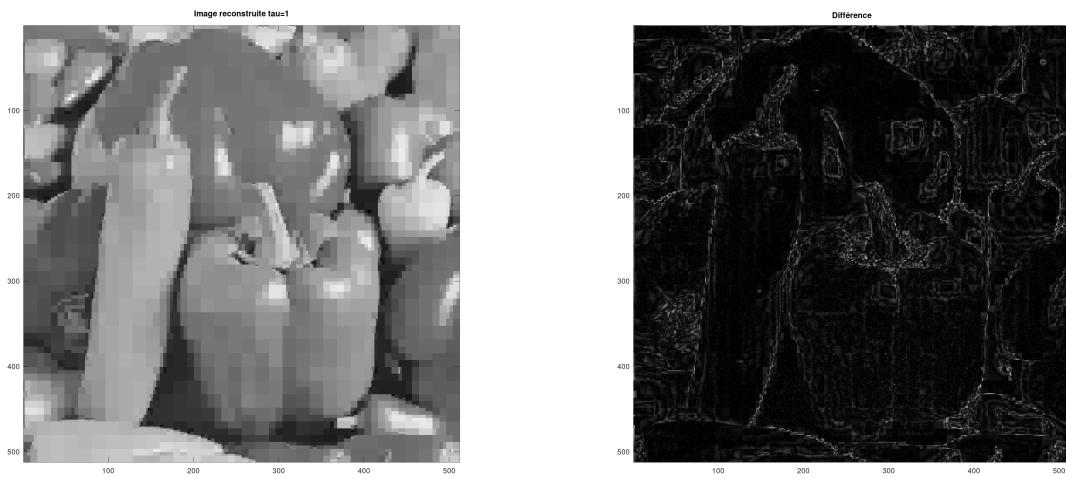


FIGURE 28 – Résultats pour $\tau = 5\%$

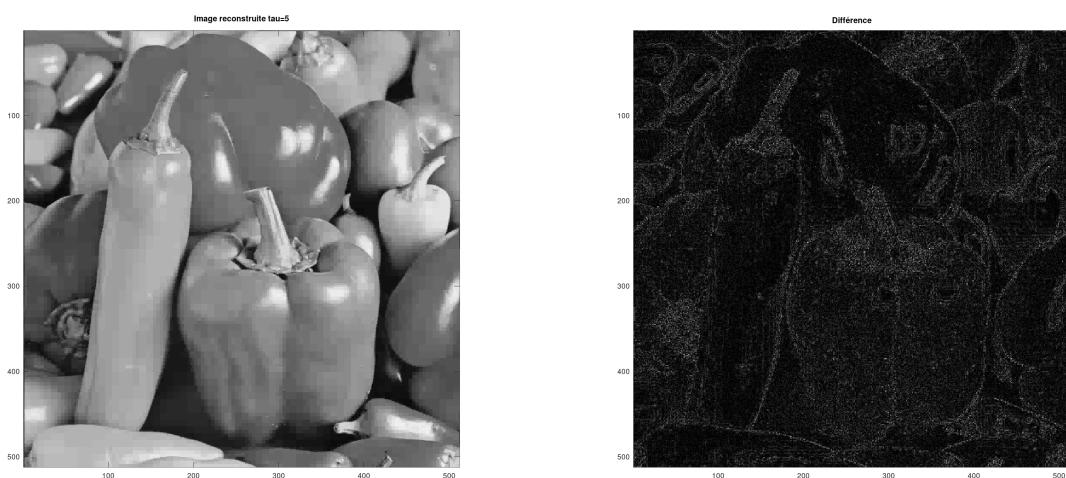
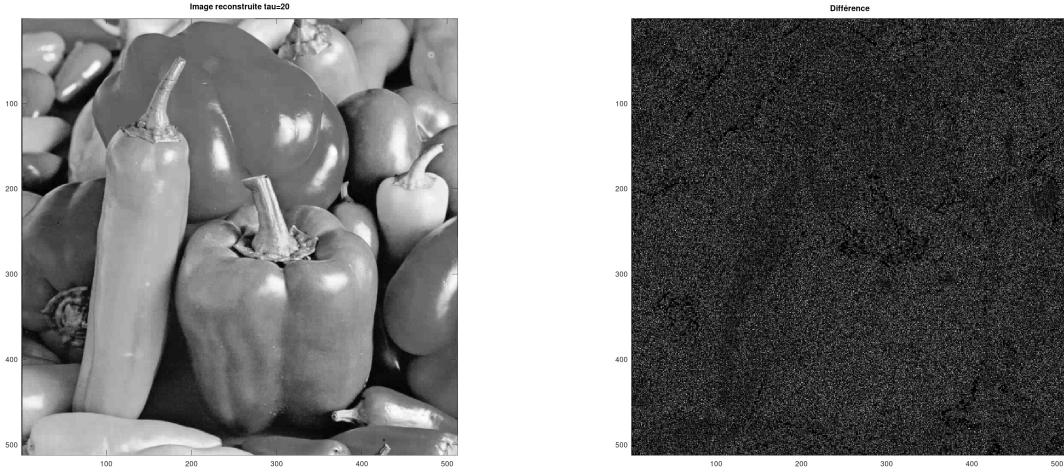


FIGURE 29 – Résultats pour $\tau = 20\%$



On remarque alors que les résultats obtenus avec un filtre générateur de Daubechies d'ordre 4 les résultats sont meilleurs qu'avec une ondelette de Haar. En effet, l'image de différence dans le cas d'une ondelette de Haar est plus riche. On a ainsi annulé des coefficients correspondant à des informations utiles, il faut une valeur de τ plus élevée que pour le premier cas.

Conclusion

Ainsi, la transformée en ondelettes discrète a plusieurs applications. Elle permet par exemple le débruitage de signaux ou la compression d'images. Ces applications reposent sur le fait que cette transformée laisse généralement place à plusieurs coefficients nuls. Nous avons aussi pu manipuler différents types d'ondelettes (Haar, Daubechies) et comprendre que pour chaque utilisation il convient de choisir attentivement l'ondelette afin d'obtenir de meilleurs résultats.

A Tracé d'ondelettes et de fonctions échelles par DWT inverse

```
1 addpath(genpath('Wavelab850'));  
2  
3 N = 1024;  
4 J = 6;  
5 P = 10;  
6 L = P-J;  
7  
8 DWT = zeros(1,N);  
9 DWT(N*(1/(2^J)+1/(2^(J+1)))) = 1;  
10  
11 h1 = MakeONFilter('Haar');  
12 h2 = MakeONFilter('Daubechies',4);  
13 h3 = MakeONFilter('Daubechies',8);  
14  
15 x1 = IWT_PO(DWT,L,h1);  
16 figure(1)  
17 plot_dwt(x1,DWT,J);  
18  
19 x2 = IWT_PO(DWT,L,h2);  
20 figure(2)  
21 plot_dwt(x2,DWT,J);  
22  
23 x3 = IWT_PO(DWT,L,h3);  
24 figure(3)  
25 plot_dwt(x3,DWT,J);
```

B Débruitage dans l'espace des ondelettes

```
1 addpath(genpath('Wavelab850'));  
2  
3 N = 1024;  
4 P = log2(N);  
5 J = 7;  
6 L = P-J;  
7  
8 h = MakeONFilter('Haar');  
9 % h = MakeONFilter('Daubechies',4);  
10  
11 DWT = zeros(1,N);  
12 DWT(N/(2^J)+1) = 2;  
13 DWT(N/(2^J)+5) = 0.5;  
14 DWT(N/(2^J)+65) = 4;  
15 DWT(N/(2^J)+503) = 3.5;  
16 x = IWT_PO(DWT,L,h);  
17  
18 figure(10)  
19 plot(x)  
20 title("Signal")  
21 ylabel("Amplitude")  
22  
23 [y,sigma] = ajoute_bruit(x,10);  
24 DWTbruit = FWT_PO(y,L,h);  
25 figure(1)  
26 plot_dwt(x,DWT,J);  
27 figure(2)  
28 plot_dwt(y,DWTbruit,J);  
29
```

```

30 figure(11)
31 plot(y)
32 title("Signal bruit ")
33 ylabel("Amplitude")
34
35 K = 100;
36 erreurs = zeros(1,K);
37 seuils = zeros(1,K);
38
39 for k = 1:K
40     DWTseuil = DWTbruit;
41
42     alpha = 0.01 * k;
43     seuils(k) = alpha;
44
45 DWTseuil(DWTbruit < alpha) = 0;
46
47 xdebruite = IWT_PO(DWTseuil,L,h);
48
49 erreur = sum((xdebruite-x).^2);
50 erreurs(k) = erreur;
51 end
52
53 alphaOpti = sqrt(sigma*2*log(N));
54 DWTseuil = DWTbruit;
55 DWTseuil(DWTbruit < alphaOpti) = 0;
56 xdebruite = IWT_PO(DWTseuil,L,h);
57 erreurOpti = sum((xdebruite-x).^2);
58
59 figure(3)
60 plot(seuils ,erreurs)
61 title("Erreur de reconstruction en fonction de alpha")
62 xlabel("alpha")
63 ylabel("Erreur de reconstruction")
64 xlim([0 K*0.01])
65 hold on
66 plot(alphaOpti ,erreurOpti , 'x')
67
68 xB = MakeSignal('Blocks',N);
69 xD = MakeSignal('Doppler',N);
70
71 figure(8)
72 plot(xB)
73 title("Signal Blocks")
74 ylabel("Amplitude")
75
76 figure(9)
77 plot(xD)
78 title("Signal Doppler")
79 ylabel("Amplitude")
80
81 DWIB = FWT_PO(xB,L,h);
82 DWID = FWT_PO(xD,L,h);
83
84 % figure(4)
85 % plot_dwt(xB,DWIB,J);
86 % figure(5)
87 % plot_dwt(xD,DWIB,J);
88

```

```

89 [yB,sigmaB] = ajoute_bruit(xB,20);
90 [yD,sigmaD] = ajoute_bruit(xD,20);
91
92 DWTBbruit=FWT_PO(yB,L,h);
93 DWTDbruit=FWT_PO(yD,L,h);
94
95 erreursB = zeros(1,K);
96 erreursD = zeros(1,K);
97
98 for k = 1:K
99     DWTBseuil = DWTBbruit;
100    DWTDseuil = DWTDbruit;
101
102    alpha = 0.01 * k;
103
104    DWTBseuil(DWTBbruit < alpha) = 0;
105    DWTDseuil(DWTDbruit < alpha) = 0;
106
107    xBdebruite = IWT_PO(DWTBseuil,L,h);
108    xDdebruite = IWT_PO(DWTDseuil,L,h);
109
110    erreurB = sum((xBdebruite-xB).^2);
111    erreurD = sum((xDdebruite-xD).^2);
112    erreursB(k) = erreurB;
113    erreursD(k) = erreurD;
114 end
115
116 alphaBOpti = sqrt(sigmaB*2*log(N));
117 alphaDOpti = sqrt(sigmaD*2*log(N));
118 DWTBseuil = DWTBbruit;
119 DWTDseuil = DWTDbruit;
120 DWTBseuil(DWTBbruit < alphaBOpti) = 0;
121 DWTDseuil(DWTDbruit < alphaDOpti) = 0;
122 xBdebruite = IWT_PO(DWTBseuil,L,h);
123 xDdebruite = IWT_PO(DWTDseuil,L,h);
124 erreurBOpti = sum((xBdebruite-xB).^2);
125 erreurDOpti = sum((xDdebruite-xD).^2);
126
127 figure(6)
128 plot(seuils,erreursB)
129 title("Erreur de reconstruction en fonction de alpha")
130 xlabel("alpha")
131 ylabel("Erreur de reconstruction")
132 xlim([0 K*0.01])
133 hold on
134 plot(alphaBOpti,erreurBOpti,'x')
135
136 figure(7)
137 plot(seuils,erreursD)
138 title("Erreur de reconstruction en fonction de alpha")
139 xlabel("alpha")
140 ylabel("Erreur de reconstruction")
141 xlim([0 K*0.01])
142 hold on
143 plot(alphaDOpti,erreurDOpti,'x')

```

C Compression d'images

```

1 addpath(genpath('Wavelab850'));

```

```

2
3 N = 1024;
4 P = log2(N);
5 J = 7;
6 L = P-J;
7
8 h = MakeONFilter( 'Haar' );
9 % h = MakeONFilter( 'Daubechies' ,4);
10
11 DWT = zeros(1,N);
12 DWT(N/(2^J)+1) = 2;
13 DWT(N/(2^J)+5) = 0.5;
14 DWT(N/(2^J)+65) = 4;
15 DWT(N/(2^J)+503) = 3.5;
16 x = IWT_PO(DWT,L,h);
17
18 figure(10)
19 plot(x)
20 title("Signal")
21 ylabel("Amplitude")
22
23 [y,sigma] = ajoute_bruit(x,10);
24 DWTbruit = FWT_PO(y,L,h);
25 figure(1)
26 plot_dwt(x,DWT,J);
27 figure(2)
28 plot_dwt(y,DWTbruit,J);
29
30 figure(11)
31 plot(y)
32 title("Signal bruit ")
33 ylabel("Amplitude")
34
35 K = 100;
36 erreurs = zeros(1,K);
37 seuils = zeros(1,K);
38
39 for k = 1:K
40     DWTseuil = DWTbruit;
41
42     alpha = 0.01 * k;
43     seuils(k) = alpha;
44
45     DWTseuil(DWTbruit < alpha) = 0;
46
47     xdebruite = IWT_PO(DWTseuil,L,h);
48
49     erreur = sum((xdebruite-x).^2);
50     erreurs(k) = erreur;
51 end
52
53 alphaOpti = sqrt(sigma*2*log(N));
54 DWTseuil = DWTbruit;
55 DWTseuil(DWTbruit < alphaOpti) = 0;
56 xdebruite = IWT_PO(DWTseuil,L,h);
57 erreurOpti = sum((xdebruite-x).^2);
58
59 figure(3)
60 plot(seuils,erreurs)

```

```

61 title("Erreur de reconstruction en fonction de alpha")
62 xlabel("alpha")
63 ylabel("Erreur de reconstruction")
64 xlim([0 K*0.01])
65 hold on
66 plot(alphaOpti , erreurOpti , 'x')
67
68 xB = MakeSignal('Blocks' ,N);
69 xD = MakeSignal('Doppler' ,N);
70
71 figure(8)
72 plot(xB)
73 title("Signal Blocks")
74 ylabel("Amplitude")
75
76 figure(9)
77 plot(xD)
78 title("Signal Doppler")
79 ylabel("Amplitude")
80
81 DWIB = FWT_PO(xB,L,h);
82 DWID = FWT_PO(xD,L,h);
83
84 % figure(4)
85 % plot_dwt(xB,DWIB,J);
86 % figure(5)
87 % plot_dwt(xD,DWIB,J);
88
89 [yB,sigmaB] = ajoute_bruit(xB,20);
90 [yD,sigmaD] = ajoute_bruit(xD,20);
91
92 DWTBbruit=FWT_PO(yB,L,h);
93 DWTDbrait=FWT_PO(yD,L,h);
94
95 erreursB = zeros(1,K);
96 erreursD = zeros(1,K);
97
98 for k = 1:K
99     DWTBseuil = DWTBbruit;
100    DWTDbrait = DWTDbrait;
101
102    alpha = 0.01 * k;
103
104    DWTBseuil(DWTBbruit < alpha) = 0;
105    DWTDbrait(DWTDbrait < alpha) = 0;
106
107    xBdebruite = IWT_PO(DWTBseuil,L,h);
108    xDdebruite = IWT_PO(DWTDbrait,L,h);
109
110    erreurB = sum((xBdebruite-xB).^2);
111    erreurD = sum((xDdebruite-xD).^2);
112    erreursB(k) = erreurB;
113    erreursD(k) = erreurD;
114 end
115
116 alphaBOpti = sqrt(sigmaB*2*log(N));
117 alphaDOpti = sqrt(sigmaD*2*log(N));
118 DWTBseuil = DWTBbruit;
119 DWTDbrait = DWTDbrait;

```

```

120 DWTBseuil(DWTBbruit < alphaBOpti) = 0;
121 DWTDseuil(DWTDbruit < alphaDOpti) = 0;
122 xBdebruite = IWT_PO(DWTBseuil,L,h);
123 xDdebruite = IWT_PO(DWTDseuil,L,h);
124 erreurBOpti = sum((xBdebruite-xB).^2);
125 erreurDOpti = sum((xDdebruite-xD).^2);
126
127 figure(6)
128 plot(seuils,erreursB)
129 title("Erreur de reconstruction en fonction de alpha")
130 xlabel("alpha")
131 ylabel("Erreur de reconstruction")
132 xlim([0 K*0.01])
133 hold on
134 plot(alphaBOpti,erreurBOpti,'x')
135
136 figure(7)
137 plot(seuils,erreursD)
138 title("Erreur de reconstruction en fonction de alpha")
139 xlabel("alpha")
140 ylabel("Erreur de reconstruction")
141 xlim([0 K*0.01])
142 hold on
143 plot(alphaDOpti,erreurDOpti,'x')

```