DEBAIN Yann
VINCENT PAUL RAJ Arvin

Rapport :
Décomposition parcimonieuse

Introduction :

On se place dans le contexte suivant : on fait un enregistrement d'une note de guitare qui est malheureusement mauvaise qualité car bruité. Avec nos connaissances, le but est d'améliorer la qualité de l'enregistrement c'est-à-dire le débruité. Pour cela, on va utiliser une méthode appelée décomposition parcimonieuse.

Le projet consiste en la compréhension de la décomposition parcimonieuse par la mise en application de celle-ci dans un cas concret : le débruitage de signaux.

Soit $x(t)$ le signal bruité et $\sum_{i=1}^{J} \alpha_i d_i(t_n)$, la décomposition de $s(t)$ sur un dictionnaire. On cherche donc à résoudre le problème d'optimisation suivant :

$$\underline{\hat{\alpha}} = arg\ min\ \| x(t) - \sum_{i=1}^{J} (\alpha_i d_i(t)) \|_2^2 + \lambda \| \alpha \|_1$$

I. Mise en équation et analyse théorique

$$s(t_n) = \sum_{i=1}^{J} \alpha_i d_i(t_n);$$

$$\underline{s} = \begin{bmatrix} s(t_1) \\ s(t_2) \\ \cdots \\ s(t_n) \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{J} (\alpha_i d_i(t_1)) \\ \sum_{i=1}^{J} (\alpha_i d_i(t_2)) \\ \cdots \\ \sum_{i=1}^{J} (\alpha_i d_i(t_n)) \end{bmatrix} \quad donc\ sous\ forme\ matricielle\ on\ peut\ ecrire\ \underline{s} = D\underline{\alpha}$$

$$\underline{\hat{\alpha}} = arg\ min\ \| x(t) - \sum_{i=1}^{J} (\alpha_i d_i(t)) \|_2^2 + \lambda \| \alpha \|_1 \ \ donc\ \underline{\hat{\alpha}} = arg\ min\ \| \underline{x} - \underline{\alpha} D \|_2^2 + \lambda \| \alpha \|_1$$

$$Donc\ \underline{\hat{\alpha}} = arg\ min\ (\underline{x} - D\underline{\alpha})^T (\underline{x} - D\underline{\alpha}) + \lambda \| \alpha \|_1$$

$$\nabla J(\underline{\alpha}) = \nabla [(\underline{x} - D\underline{\alpha})^T (\underline{x} - D\underline{\alpha})] + \nabla [\lambda \| \underline{\alpha} \|_1]$$

$$\frac{\partial \underline{a}}{\partial \underline{\alpha}} = \begin{bmatrix} \frac{\partial a_1}{\partial \alpha_1} & \frac{\partial a_1}{\partial \alpha_2} & \cdots & \frac{\partial a_1}{\partial \alpha_I} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial a_N}{\partial \alpha_1} & \frac{\partial a_N}{\partial \alpha_2} & \cdots & \frac{\partial a_N}{\partial \alpha_I} \end{bmatrix} = D \quad Avec\ \underline{a} = D\underline{\alpha}$$

$$Donc\ \nabla J(\underline{\alpha}) = -2D^T(\underline{x} + D\underline{\alpha}) + \lambda \nabla \parallel \underline{\alpha} \parallel_1$$

$$\frac{\partial |\underline{\alpha}|}{\partial \alpha_i} = \begin{cases} 1 \quad si\ \alpha_i > 0 \\ -1 \quad si\ \alpha_i < 0 \\ [-1;1] \quad si\ \alpha_i = 0 \end{cases} \longrightarrow \nabla \parallel \underline{\alpha} \parallel_1 = \begin{bmatrix} signe(\alpha_1) \\ signe(\alpha_2) \\ \cdots \\ signe(\alpha_N) \end{bmatrix}$$

II. Décomposition parcimonieuse

Pour comprendre le concept de décomposition parcimonieuse, nous allons commencer par travailler sur une partie du signal bruité et sur des dictionnaires spécifiques. Puis nous adapterons ce principe sur l'ensemble du signal bruité et des dictionnaires quelconques.

II. 1. Décomposition sur dictionnaire unitaire

Nous choisissons de commencer sur une partie du signal : $x(47000:49000,\ 1)$ (2000 points)

On commence par travailler sur un dictionnaire unitaire de cosinus discret car cela simplifie le calcul des différents α_i.
En effet, si D est un dictionnaire unitaire, on a :

$$0 = -2D^T(\underline{x} + D\underline{\alpha}) + \lambda \nabla \underline{\alpha} = \nabla J(\underline{\alpha}) = -2D^T\underline{x} - 2\underline{\alpha} + \lambda \nabla \underline{\alpha}$$

Donc en posant $Y = D^T\underline{x}$ on a $si\ \alpha_i > 0 \longrightarrow -2y_i - 2\alpha_i + \lambda = 0$

$$si\ \alpha_i < 0 \longrightarrow -2y_i - 2\alpha_i - \lambda = 0$$
$$si\ \alpha_i = 0 \longrightarrow y_i * [-1,1] = 0$$

Donc $si\ y_i > \lambda \longrightarrow \alpha_i = y_i - \lambda/2$

$$si\ y_i < \lambda \longrightarrow \alpha_i = y_i + \lambda/2$$
$$si\ y_i \in [-\lambda, \lambda] \longrightarrow \alpha_i = 0$$

Ceci va nous permettre de calculer plus vite les α_i optimaux.

a. Dictionnaire unitaire composé des cosinus discrets

Les éléments de ce dictionnaire sont donnés par la formule suivante :

$$\forall (i,j) \in [1;N]^2\ ,\ D_{i,j} = \sqrt{\frac{2}{N}} \frac{1}{\sqrt{1+\delta_{i,j}}} \cos\left(\frac{\pi}{2N}(2j-1)(i-1)\right),\ avec\ \delta_{i,j},\ le\ symbole\ de\ Kronecker$$

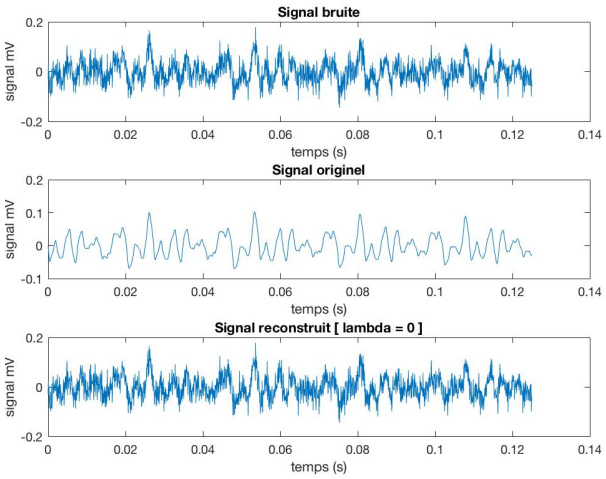
Création du dictionnaire :

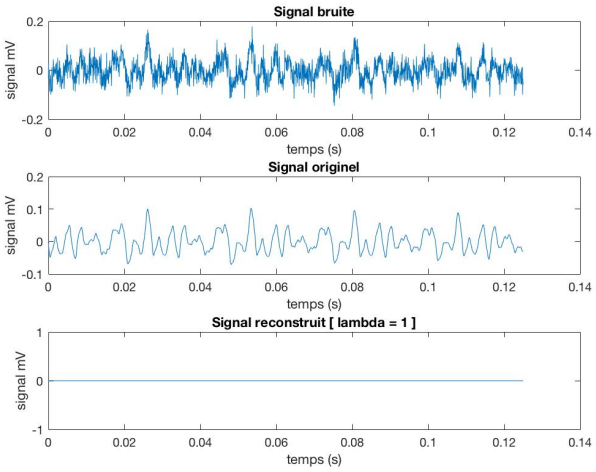
```
for i = 1:N
       for j = 1:N
       D(i, j) = sqrt(2/N)*(1/sqrt(1+eq(i,1)))*cos(pi*(2*j-1)*(i-1)/(2*N));
       end
end
```

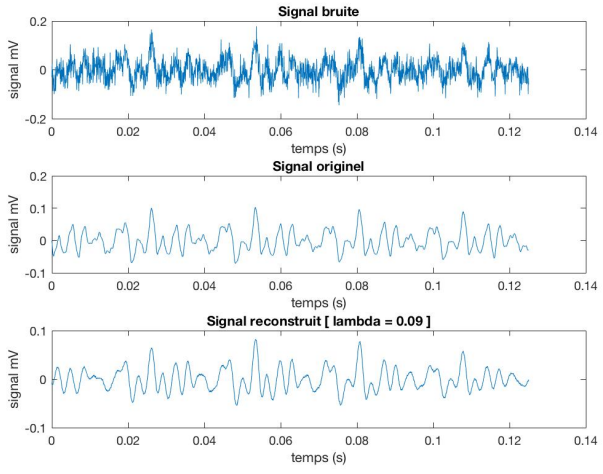
Débruitage d'une portion du signal :

```
x = x(47000:50000, 1)
Y = D'*x;

for i = 1:N
       if Y(i, 1) > lambda
              alpha(i, 1) = Y(i, 1) - lambda;
       elseif Y(i) < -lambda
              alpha(i, 1) = Y(i, 1) + lambda;
       else
              alpha(i, 1) = 0;
       end
end

s_reconstruit = D*alpha;
```

Si on affiche les courbes obtenues pour le premier signal avec différentes valeurs de λ on obtient :







On remarque la valeur du paramètre λ a une grande importance dans la décomposition parcimonieuse.

C'est ce paramètre qui va faire le compromis sur l'atténuation du bruit et le signal originel et donc va nous permettre une meilleure qualité de débruitage en fonction du α calculé.

Pour savoir quel λ choisir, on va s'appuyer sur l'écart moyen quadratique (EQM). On choisira λ tel que l'EQM soit le minimum possible.

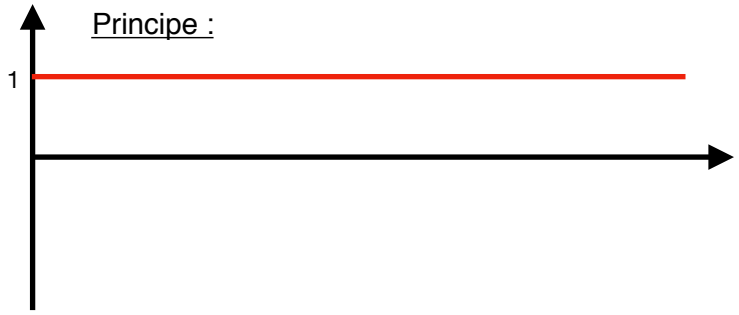
Une fois ce principe compris, appliquons cet algorithme sur le signal entier.

```
Requested 80000x80000 (47.7GB) array exceeds maximum array size preference. Creation of arrays greater than this limit may
take a long time and cause MATLAB to become unresponsive. See array size limit or preference panel for more information.
```

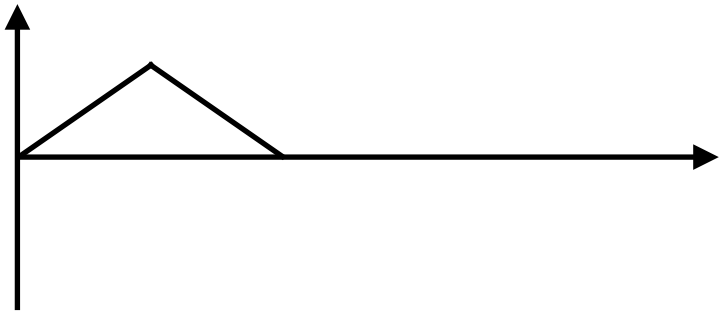
Dû aux limitations techniques, on ne peut pas appliquer cet algorithme sur le signal en entier. On doit alors changer de méthode de débruitage.
L'algorithme a fonctionné sur une partie du signal, on peut donc appliquer cet algorithme sur chaque portion du signal et le reconstruire en sommant toutes les portions reconstitué, mais on a aucune condition sur la continuité du signal reconstitué donc cette méthode permettra de reconstruire le signal, mais avec des discontinuités.

Pour palier à ce problème, on va créer des fenêtres d'analyse qui vont se chevaucher. Le chevauchement, lors de la reconstitution du signal en entier va permettre de gommer les discontinuités.

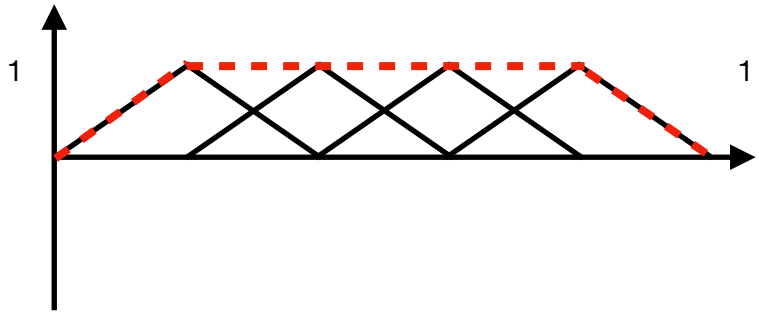
Principe :



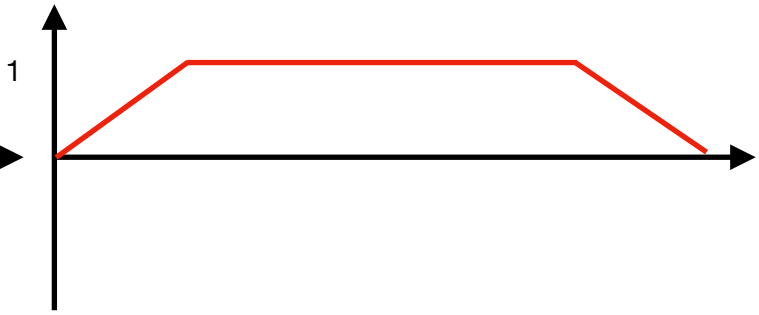
signal d'origine



fenêtre d'analyse



Principe de la décomposition en fenêtre d'analyse se chevauchant



Signal après reconstitution

En prenant une fenêtre d'analyse triangulaire, on concentrera la portion du signal au centre de cette fenêtre.
En faisant chevaucher les chevaucher les différentes portions du signal, on assure que tout le signal est traité.
Ainsi, quand on reformera le signal en sommant les portions débruitées on retrouve bien le signal débruité sauf au début du signal et à la fin à cause des effets de bord de la somme des portions débruitées.

Decomposition du signal :

On applique donc le dictionnaire sur chaque portion du signal et on trouve le α correspondant.

```
for k = 1:2*N/N_decoupage -2
    X = x(debut:fin, 1);
    Y = D'*X;
    for i = 1:N_decoupage
        if Y(i, 1) > lambda
            alpha(i, 1) = Y(i, 1) - lambda;
        elseif Y(i) < -lambda
            alpha(i, 1) = Y(i, 1) + lambda;
        else
            alpha(i, 1) = 0;
        end
    end
    s_decoupe(:, k) = D*alpha;
    debut = debut + N_decoupage/2;
    fin = fin + N_decoupage/2;
end

[N_ligne,N_colonne] = size(s_decoupe);

for l = 1:N_colonne
    s_fenetre(:, l) = s_decoupe(:, l).*fenetre;
end
```

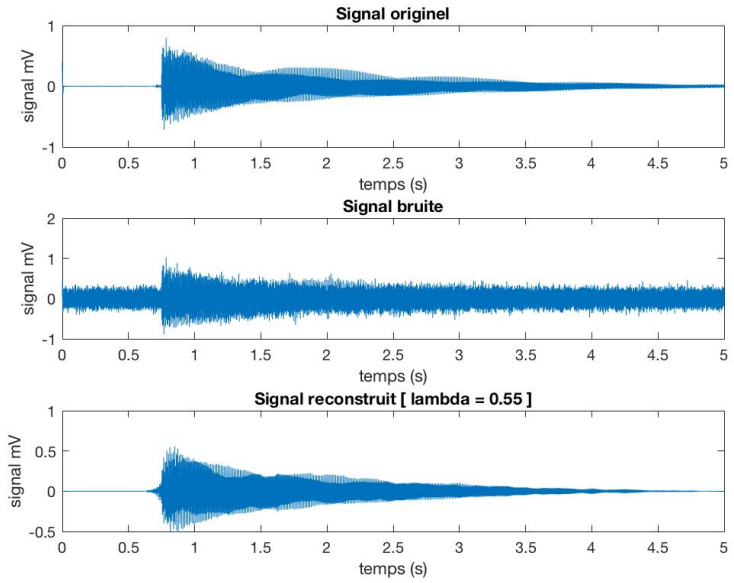
Reconstitution du signal :

Il ne nous reste plus qu'a sommer les différentes portions débruité du signal en respectant bien le principe vu précédemment.

```
[N_ligne,N_colonne] = size(s_fenetre);

s_reconstruit = zeros(N,1);
s_reconstruit(1:N_ligne) = s_fenetre(:, 1);

ligne = N_ligne;

for h = 2:N_colonne
    s_reconstruit(ligne/2 + 1:ligne/2 + N_ligne/2) = s_reconstruit(ligne/2 + 1:ligne/2
+ N_ligne/2) + s_fenetre(1:N_ligne/2, h);
    s_reconstruit(ligne/2 + N_ligne/2 + 1:ligne/2 + N_ligne) = s_fenetre(N_ligne/2 +
1:N_ligne, h);
    ligne = ligne + N_ligne;
end
```

On obtient les résultats finaux suivant :

```
EQM =  0.00089738
le taux de parcimonie est de : 0.535%
>>
```



On vérifie la qualité du débitage à l'aide de l'écart quadratique moyen (EQM) entre le signal débruité et le signal d'origine et le taux de parcimonie permet de savoir si le signal a été « facile » a débruité. Plus l'écart moyen quadratique est faible, mieux est le débruitage et plus le taux de parcimonie est faible plus le signal a été « facile » à débruité.

On peut donc vérifier la qualité du débruitage entre les différentes méthodes et donc comparer les efficacités des méthodes.

On commence par comparer les signaux avec la même méthode et les mêmes paramètres pour savoir si la forme du signal est un paramètre dans le débruitage

	Taux de parcimonie et EQM du signal
Signal 1	EQM = 0.00043982 le taux de parcimonie est de : 0.12%
Signal 2	EQM = 0.0004372 le taux de parcimonie est de : 0.1625%
Signal 5	EQM = 0.00065368 le taux de parcimonie est de : 0.2875%
Signal 7	EQM = 0.00042631 le taux de parcimonie est de : 0.2575%
Signal 8	EQM = 0.00063172 le taux de parcimonie est de : 0.1875%
Signal 11	EQM = 0.00097871 le taux de parcimonie est de : 0.28%
Signal 12	EQM = 0.00018709 le taux de parcimonie est de : 0.03625%
Signal 13	EQM = 0.00089738 le taux de parcimonie est de : 0.535%

Tableau récapitulatif des taux de parcimonies des signaux et des EQM en fonction des signaux

En comparant les différents taux de parcimonie et de les EQM, on remarque que la forme du signal influe peu sur le taux de parcimonie et l'EQM du signal. En effet pour chaque signal, le taux de parcimonie et l'EQM restent sensiblement les mêmes.

	Taux de parcimonie et EQM du signal
Signal 13 découpage : 500 points	EQM = 0.0027044 le taux de parcimonie est de : 1.1513%
Signal 13 découpage : 2000 points	EQM = 0.0013348 le taux de parcimonie est de : 0.715%
Signal 13 découpage : 4000 points	EQM = 0.00089738 le taux de parcimonie est de : 0.535%

Tableau récapitulatif des taux de parcimonies des signaux et des EQM en fonction du découpage

Le découpage du signal influe sur l'efficacité du débruitage. En changeant le nombre de points du découpage, le taux de parcimonie et l'EQM influent fortement.

En effet, plus le nombre de points du découpage est faible plus le taux de parcimonie et l'EQM sont grands.

Les taux de parcimonie et l'EQM sont inversement proportionnels aux nombres de points du découpage.

Conclusion : La méthode de débruitage avec un dictionnaire unitaire de cosinus discret est une méthode efficace pour débruiter le signal car elle a une bonne efficacité (EQM grand) et une rapidité de débruitage très grande (taux de parcimonie faible).

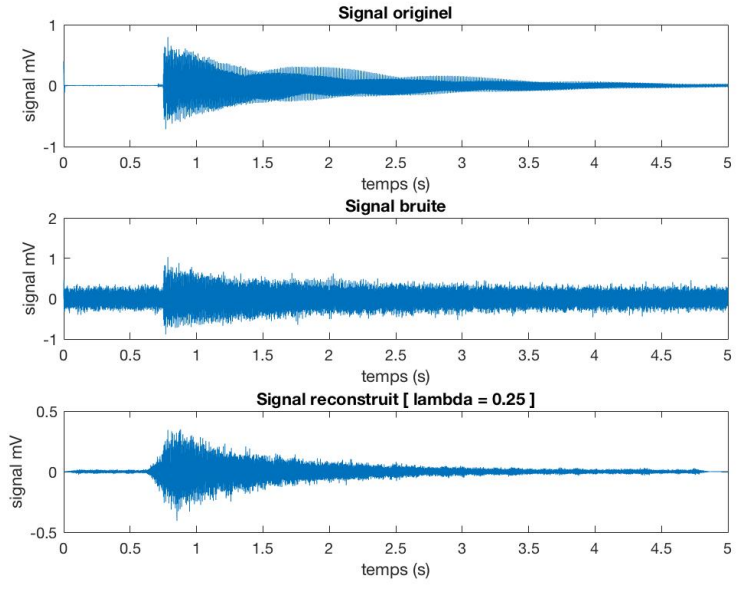
b. Dictionnaire unitaire aléatoire

On crée cette fois-ci un dictionnaire unitaire aléatoire pour ne pas se limiter à une seule sorte de dictionnaire et pour vérifier si le débruitage est aussi efficace quel que soit le dictionnaire (unitaire).

Création du dictionnaire :

```
A = randn(N);
A = A'*A;
[D, Diag] = eig(A); %On veut la matrice D qui est unitaire
```

On utilise le même principe pour pourvoir débruiter le signal et on obtient les résultats suivants pour le signal 13 :




```
EQM =  0.0075929
le taux de parcimonie est de : 18.6513%
```

On remarque cette fois-ci que le débruitage n'est pas optimal. En comparant le taux de parcimonie avec celui de la méthode du dictionnaire unitaire avec des cosinus discrets, on voit qu'il y a un facteur 100, ce qui montre qu'avec ce dictionnaire le débruitage se fait « difficilement ». Ceci influe donc sur la qualité du débruitage où l'EQM est presque 100 fois plus grand que l'EQM de la méthode précédente.

	Taux de parcimonie et EQM du signal
Signal 1	EQM = 0.00097154 le taux de parcimonie est de : 0.025%
Signal 2	EQM = 0.0014996 le taux de parcimonie est de : 0.19625%
Signal 5	EQM = 0.0037277 le taux de parcimonie est de : 4.0237%
Signal 7	EQM = 0.0044632 le taux de parcimonie est de : 8.3712%
Signal 8	EQM = 0.0015134 le taux de parcimonie est de : 0.04375%
Signal 11	EQM = 0.0024447 le taux de parcimonie est de : 0.64%
Signal 12	EQM = 0.00030891 le taux de parcimonie est de : 0%
Signal 13	EQM = 0.0075749 le taux de parcimonie est de : 18.5063%

Tableau récapitulatif des taux de parcimonies des signaux et des EQM en fonction du découpage

Avec ce dictionnaire, les résultats montrent que la forme du signal influe fortement sur le taux de parcimonie et l'EQM.

On peut en déduire que le choix du dictionnaire est important dans le débruitage d'un signal. Le choix du dictionnaire permet d'optimiser la vitesse d'exécution du programme (taux de parcimonie faible) et l'efficacité du débruitage (EQM faible).

	Taux de parcimonie et EQM du signal
Signal 13 découpage : 500 points	EQM = 0.0074452 le taux de parcimonie est de : 18.6212%
Signal 13 découpage : 2000 points	EQM = 0.0075069 le taux de parcimonie est de : 18.455%
Signal 13 découpage : 4000 points	EQM = 0.0075749 le taux de parcimonie est de : 18.5063%

Tableau récapitulatif des taux de parcimonies des signaux et des EQM en fonction du découpage

Le nombre de points du découpage semble ne pas influencer sur l'efficacité et la vitesse de débruitage (l'EQM et le taux de parcimonie restent constants).

Pour accélérer le programme avec ce type de dictionnaire, il vaut donc mieux avoir un nombre de découpages assez faible.

Conclusion : Débruiter à l'aide d'un dictionnaire unitaire aléatoire a mis en valeur le fait que le choix d'un dictionnaire est important dans le débruitage, car ce dernier joue dans l'efficacité et la rapidité du débruitage du signal.

II. 2. Décomposition sur dictionnaire quelconque

Après avoir étudié l'efficacité du débruitage avec un dictionnaire unitaire, on va mettre en place une méthode qui va permettre de débruiter un signal. Le dictionnaire n'étant plus unitaire on ne peut plus calculer les α_i de la même manière.

Pour déterminer les α_i on va donc utiliser une méthode d'optimisation qui va permettre de faire converger $\underline{\alpha}$ vers $\underline{\widehat{\alpha}}$.

La méthode d'optimisation choisie est celle de la descente du gradient à pas fixe.

```
for ind = 2:nbItMax

    % Calcul du gradient
     gradJ = -2*D'*(X - D*alpha) + lambda*sign(alpha);

     % mise a jour des parametres
     alpha = alpha - rho*gradJ;
     J(ind) =  (X - D*alpha)'*(X - D*alpha) + lambda*norm(alpha, 1);

end
```

a. Dictionnaire quelconque de cosinus discrets

On commence avec un dictionnaire particulier composé de cosinus discrets, pour éviter d'avoir un dictionnaire unitaire, on va multiplier le dictionnaire par une fenêtre, ce qui permet d'enlever le caractère unitaire et d'avoir un dictionnaire « quelconque ».

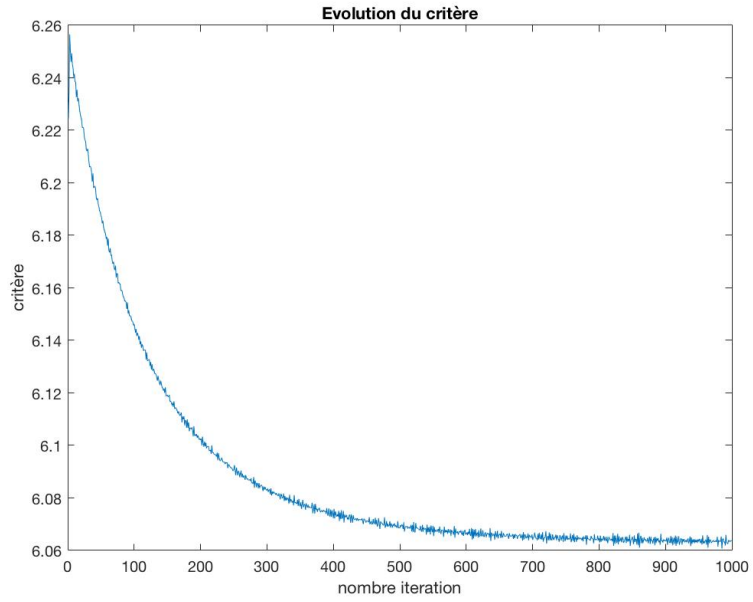
Création du dictionnaire :

```
for i = 1:N
    for j = 1:N
         D(i, j) = sqrt(2/N)*(1/sqrt(1+eq(i,1)))*cos(pi*(2*j-1)*(i-1)/(2*N));
     end
end

D = D.*window(@triang, N);
```

Résolution du problème :

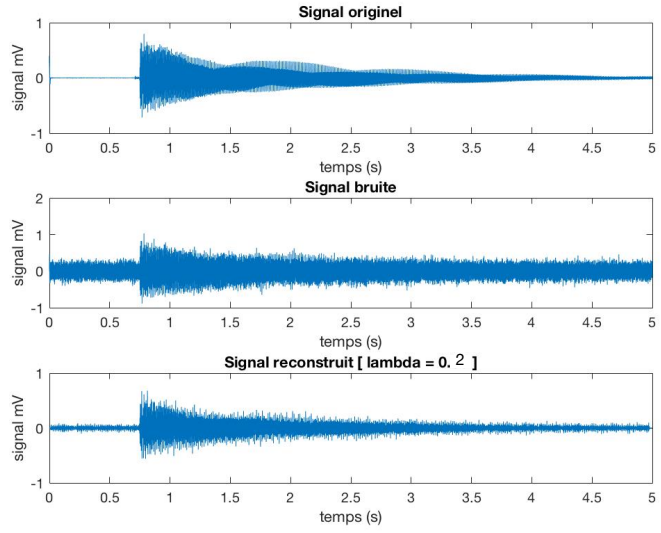
Pour résoudre le problème, on va utiliser le même principe que pour le dictionnaire unitaire sauf qu'à chaque débruitage d'une portion du signal, on recalcule le vecteur $\underline{\alpha}$ avec la méthode de la descente du gradient. On peut déjà en déduire que le débruitage sera plus long, car la méthode du gradient est une méthode itérative.

Pour pouvoir exploiter les résultats, il faut s'assurer que le critère d'optimisation décroît bien et tend vers son minimum.



Le critère décroît bien donc on peut afficher les résultats, les exploiter et faire la comparaison avec les méthodes précédentes.

Resultats :



EQM = 0.0014446
le taux de parcimonie est de : 99.375%

Avec un dictionnaire quelconque de cosinus discrets, on voit le débruitage est lent (le taux de parcimonie est proche de 100%) et que le débitage est approximatif.

On entend à l'oreille que le bruit a été diminué, mais que de nouveaux artefacts se sont ajoutés.

Avec ce type de dictionnaire, il est nécessaire de faire une autre technique ou de combiné la décomposition parcimonieuse avec une technique de débruitage plus poussée de post-production plus poussé comme rajouter un passe-haut pour éliminer les artefacts aigus.

Après plusieurs essais pour observer, on constate que cette méthode itérative est consommatrice en temps. Il faut donc trouver une méthode pour pouvoir débruiter le signal mais, sans attendre 10min. On peut soit changer de méthode itérative (faire la méthode de Newton) ou jouer sur le nombre de flops calculé à chaque itération.

Pour diminuer le nombre de flops, on peut doit insérer un critère d'arrêt, soit mieux découper le signal.

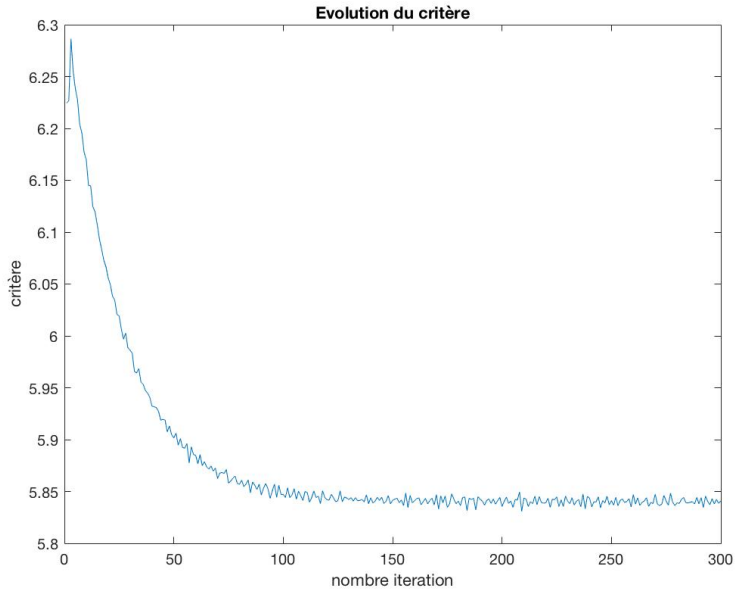
b. Dictionnaire quelconque de cosinus discrets et de sinus discret

On change de dictionnaire pour voir si la décomposition sur une autre base permet d'avoir de meilleurs résultats.

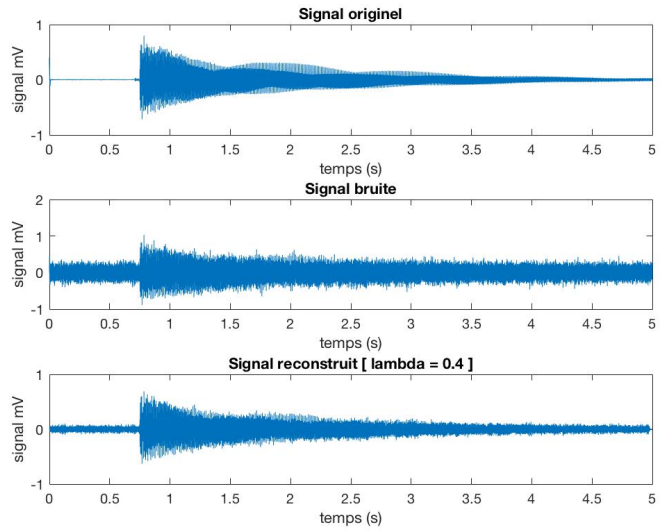
Création du dictionnaire :

```
for i = 1:N
    for j = 1:N
         D(i, j) = sqrt(2/N)*(1/sqrt(1+eq(i,1)))*cos(pi*(2*j-1)*(i-1)/(2*N)) + sqrt(2/
N)*(1/sqrt(1+eq(i,1)))*sin(pi*(2*j-1)*(i-1)/(2*N));
     end
end
```

On réitère les mêmes étapes pour ce dictionnaire (vérification de la décroissance du critère, exploitation des résultats).



Le critère décroit bien, on tend donc vers le vecteur $\underline{\alpha}$ ce qui va nous permettre de débruiter le signal.



EQM = 0.0012689
le taux de parcimonie est de : 99.375%

Vu qu'on utilise la même méthode itérative, on a le même taux de parcimonie donc le débruitage est assez lent. L'EQM, par rapport au dictionnaire précédent, ne change pas.

On retrouve les même artefacts auditifs qu'avec le dictionnaire précédent. Il faudrait ajouter une technique qui permet d'éliminer ces artefacts en sortie du signal débruité reconstitué.

Conclusion : La méthode de débruitage avec un dictionnaire quelconque composé de cosinus discret seul ou avec des sinus discrets est une méthode de débitage qui est « lente », qui débruite assez bien le signal mais en contre partie, crée des artefacts non désirés. Il est donc nécessaire de rajouter un système en sortie de ce signal pour éliminer ces artefacts et avoir le signal débruité aussi proche du signal original.

c. Dictionnaire quelconque aléatoire.

Dans le cas du dictionnaire quelconque aléatoire, nous n'avons pas pu obtenir de résultats probants.

Conclusion :

Ce projet nous a permis de découvrir une méthode pour débruiter un signal. En commençant par une décomposition sur un dictionnaire simple nous avons pu découvrir ce qui caractérise les performances de cette méthode. Nous avons appris à faire le rapport entre subjectivité et objectivité, car nous avons observé les résultats de manière objectif (taux de parcimonie et écart moyen quadratique) mais aussi de manière subjective (à l'écoute).

En accroissant la « difficulté » de la résolution du problème nous avons pu mettre en place une méthode d'optimisation vu en cours, mais aussi remarquer tous les problèmes apportés par les phénomènes d'implémentation (faire attention à l'occupation de la RAM) et ainsi mettre une méthode qui permet d'éviter ce problème. Nous avons aussi appris que la base de décomposition est très importante dans cette méthode, car elle réduit fortement le temps de débruitage et peut rendre le débruitage très efficace.

Nous avons pu ainsi découvrir et apprendre comment aborder un problème scientifique et le résoudre de manière numérique à travers différentes méthodes.