Transformée en ondelettes 1D

Ch. Dossal

Juillet 2008

1 Wavelab

Pour effectuer le TD vous aurez besoin de la toolbox wavelab. Téléchargez là, (tapez wavelab802 dans google et suivez le premier lien). Cette toolbox développée par l'équipe de David Donoho à Stanford est la toolbox de référence en ondelettes, elle est totalement gratuite. Installez là en suivant les instructions. La commande qui réalise la transformée en ondelette (périodisée et orthonormée) est

```
>> W=FWT_PO(S,l,qmf);
```

Elle prend pour argument
- Un vecteur S de taille $N = 2^d$,
- Un nombre $l < d$ tel que 2^l est la taille de l'approximation de S.
  Quand $l = d - 1$, on a effectué une étape de transformée en ondelettes.
- Un filtre qmf qui spècifie l'ondelette.

Dans ce TD nous allons étudier différentes transformées en ondelettes. Chaque transformée est associé au couple de filtres (h, g) du cours, ces filtres sont appelés 'Quadratic Mirror Filters'. Comme g dépend directement de h, on ne précise que le filtre h. La commande MakeONFilter founit les filtres associés à quelques ondelettes de Référence. Utilisez la commande help pour l'utiliser. Vous pourrez tester les ondelettes de Haar, de Daubechies 4,6 et 8 et les ondelettes de Coifman. Par exemple

```
>> qmf=MakeONFilter('Haar');
```

```
>> qmf2=MakeONFilter('Daubechies',4);
```

Attention aux majuscules !
Vous pourrez comparer les ondelettes de Daubechies 4 avec les Coiflets 1 et les Dauchebies 8 avec les Coiflets 2.
La commande de transformée en ondelettes inverse est

```
>> S2=IWT_PO(W,l,qmf)
```

où W est une transformée en ondelettes réalisée avec le filtre qmf.

2 Visualisation

Vous pourrez utiliser les signaux de références de la fonction MakeSignal pour tester vos programmes. En particulier Blocks, Piece-Regular, Piece-Polynomial, HeaviSine et Ramp de taille 1024 ou 2048 par exemple. Je vous conseille de les renormaliser en norme 2 en les divisants par la commande

```
>> S=S/norm(S,2);
```

afin de pouvoir comparer les erreurs d'approximation sur les différents signaux.

1. En utilisant uniquement la commande

   ```
   >>Ond=IWT_PO(S,d,qmf);
   ```

   visualiser les différentes ondelettes et fonctions d'échelles (pour différents filtres qmf) à différentes échelles j. Vous pourrez calculer la transformée inverse d'un vecteur ayant très peu de composantes non nulles. Visualiser deux ondelettes ou fonctions d'échelle consécutives. Pour Haar, les ondelettes d'une même échelle sont disjointes mais ce n'est pas le cas pour les autres. Combien y a til de fonctions d'échelle à l'échelle la plus grossière pour les ondelettes de Daubechies 4 ? Coiflets 2 ?
2. Visualiser les différentes étapes de la transformée en ondelettes d'un signal S en faisant varier le deuxième paramètre de la fonction FWT PO pour différentes ondelettes.
3. Pour les ondelettes de Daubechies 4, 6 et 8, visualiser le module de la Transformée de Fourier des ondelettes de différentes échelles. Observer la localisation fréquentielle de ces ondelettes et des fonctions d'échelles.

3 Approximation linéaire

4. Ecrire un programme qui calcule la projection d'un vecteur S sur un espace V_j, qui affiche l'approximation obtenue et qui calcule l'erreur ℓ_2 obtenue.
5. Ecrire un programme qui calcule et affiche en fonction de j, l'erreur d'approximation linéaire en ondelettes.
6. Comparer la performance des ondelettes à une approximation correspondante en Fourier (avec le même nombre de coefficients), sur différents signaux.
   Vous devez observer qu'à l'exception peut être de Blocks, on n'a pas un gros gain à faire de l'aproximation linéaire en ondelettes. Faire des ondelettes pour faire de l'approximation linéaire n'a pas un grand intérêt. Les ondelettes prennent tout leur intérêt en non linéaire.

4 Approximation non linéaire

7. Ecrire un programme d'approximation non linéaire en ondelettes et qui calcule l'erreur ℓ_2. Le nombre de coefficients conservé est un entier quelconque, pas nécessairement une puissance de 2.
8. Ecrire un programme qui calcule l'erreur ℓ_2 en fonction du nombre de coefficents conservés. Attention, il n'est pas utile de faire la reconstruction pour calculer l'erreur. On peut s'en sortir en deux lignes avec la commande "sort" et la commande cumsum en ne considérant que les coefficients sans jamais faire une reconstruction.
9. Comparer les erreurs d'approximation linéaire et non linéaire en utilisant différentes ondelettes et différents signaux.
10. Ecrire un programme qui affiche les courbes d'erreurs associées à plusieurs ondelettes et de Fourier non linéaire pour un même signal.
11. Comparez ces erreurs sur les différents signaux. Qu'observez vous ?

12. Le nombre de moments nuls de l'ondelettes définit le degré des polynomes qui sont orthogonaux aux ondelettes. Ecrire un programme qui permet de tester le nombre de moments nulles d'une ondelettes. Pouvez vous en déduire le degré des polynomes utilisés dans le signal "Piece-Polynomial" ?

5 Quantification et Entropie

Dans une optique de compression, il est important de concentrer l'énergie du signal étudié sur un petit nombre de coefficients. Il ne serait cependant pas pertinent de ne conserver un petit coefficient pour les coder ensuite de manière très précises. En pratique, pour comprimer un signal, on effectue une quantification qui réalise un travail plus complet que l'approximation non linéaire en s'assurant que chaque coefficient conservé sera codable sur un minimum de bits.

13. En utilisant les fonctions de quantification et d'entropie programmées lors d'un TD précédent écrire un programme qui qui qauntifie un signal donnée dans une base d'ondelette.

    `[Rec,E,er]=QuantiOnd(S,qmf,L,pas)`

    Cette fonction prend en entrée un vecteur S, un filtre qmf, un paramètre L définissant le nombre d'étapes dans la transformée en ondelettes et un pas de quantification. Elle renvoie un vecteur quantifié Rec, une entropie E et une erreur ℓ_2 er.

14. Ecrire un programme

    `[Rec,E]=QuantiOnd2(S,qmf,L,er)`

    similaire au précédent mais qui prend en entrée une erreur ℓ_2 er et qui calcule le pas de quantification assurant une erreur de quantification er.

15. Tester cette fonction sur les signaux de bases en faisant tous les paramètres. Il peut être intéressant pour déterminer l'influence d'un paramètre de le faire varier en conservant les autres constants.

16. Pour chacun des signaux de référence et pour une erreur donnée, déterminer la base d'ondelettes qui assure l'entropie et donc une taille de codage minimale. Dans quelle mesure cette base dépend t elle de l'erreur ?

17. Est il utile de toujours aller au bout de la transformée en ondelettes ?