Analyse de Fourier et hilbertienne
ENS de Cachan
Département de Mathématiques Verndredi 07 décembre 2012

Feuille de TP numéro 3

1 Consignes et mise en route

1.1 Rapport de TP

- Consignes pour le rapport de TP : cette feuille de TP comporte des questions auxquelles il est demandé de répondre dans un rapport de TP individuel.

- Format du rapport : les rapports de TP sont à rendre sous forme de fichier .pdf rédigés avec LaTeX.

- Remise du rapport : le rapport de ce TP doit être rendu avant le **vendredi 11 janvier 2013**. Les rapports, au format .pdf, doivent être envoyés par email à l'adresse **gabriele.facciolo@cmla.ens-cachan.fr**. Le sujet de l'email doit être **Rapport du TP3 de *Nom Prénom*** et le fichier .pdf doit être intitulé ***nom_prenom*_tp3.pdf**.

Ce TP est basé sur un des nombreux *numerical tours* de traitement d'image/signal créé par Gabriel Peyre. Pour plus d'informations vous pouvez visiter :
`http://www.ceremade.dauphine.fr/~peyre/numerical-tour/tours/`.

1.2 Téléchargement des fichiers nécessaires

Ouvrez un terminal et tapez-y ligne par ligne les instructions suivantes pour télécharger les fichiers nécessaires et lancer Matlab (on suppose ici que le dossier `tp_fourier` a été créé dans votre dossier `home` lors du premier TP) :

```
cd tp_fourier
wget dev.ipol.im/~facciolo/hilbert/tp3.zip
unzip tp3.zip
cd tp3
matlab
```

1.3 Visualiser les signaux

Les signaux sont des vecteurs 1D, le plus souvent stocké sous forme de tableaux de dimensions $(n, 1)$ où n est le nombre d'échantillons. Utiliser la fonction `load_signal.m` pour charger un signal de dimension n. Cette fonction doit se trouver dans votre répertoire courant.

```
n = 1024;
s = load_signal('Piece-regular', n);
```

Pour le visualiser, on utilise la fonction `plot.m`.

```
plot(s); axis tight;
```

1.4 Lire et visualiser les images

Pour lire une image `u` on utilise la fonction `imread`, pour la visualiser (ou n'importe quelle tableau) nous utiliserons `imagesc}` :

```
u =  double(imread('file.png'));
imagesc(u, [0,255]);
colormap gray;
axis image;
```

2 Approximation des signaux et images - bases hilbertiennes

Soit $\mathcal{B} = \{e_m\}_m$ une base orthonormée d'un espace de Hilbert $\mathbb{H}$. Tout $u \in \mathbb{H}$ peut se décomposer sur cette base

$$u = \sum_m \langle u, e_m \rangle \, e_m.$$

Si au lieu de représenter u par tous les produits scalaires $\{\langle u, e_m \rangle\}$, on se limite à un certain sous-ensemble d'indices I_M, avec $M = |I_M|$ éléments, on obtient une approximation :

$$u_{I_M} = \sum_{m \in I_M} \langle u, e_m \rangle \, e_m.$$

Cette approximation est la projection orthogonale de u sur l'espace engendré par $\{e_m\}_{m \in I_M}$.

► **Question 1 :** Montrer que l'erreur d'approximation $e[I_M] = ||u - u_{I_M}||^2$ est alors

$$e[I_M] = \sum_{m \notin I_M} |< u, e_m >|^2 .$$

et tend nécessairement vers 0 quand $M \to \infty$. Cependant, la vitesse de décroissance varie selon la base et le sous-ensemble I_M choisis.

On va étudier dans la suite les bases orthogonales de Fourier et cosinus (qui est au coeur de JPEG). Les lecteurs interessés peuvent poursuivre l'étude avec les bases orthogonales en ondelettes (qui sont au coeur de JPEG-2000).

2.1 Approximaton linéaire de Fourier

On considère une base de Fourier et on approxime u en prenant les M premiers coefficients, $I_M = \{m : -M/2 \leq m < M/2\}$. Dans ces conditions M définit l'ensemble I_M. On peut donc noter sans ambiguïté $u_M = u_{I_M}$ et $e[M] = e[I_M]$. C'est ce qu'on appelle l'approximation linéaire.

► **Question 2 :** Calculez la transformée de Fourier (`fft`) pour le signal $s =$ `Piece-regular`. Annulez les coefficients de Fourier des hautes fréquences, puis reconstruisez le signal s_M en ne gardant que M coefficients de Fourier, avec M en paramètre.
Afficher dans une figure avec la commande `subplot` le signal original s, ses coefficients de Fourier (module), ainsi que l'approximation s_M et les coefficients de Fourier réduits. Utiliser la fonction `fftshift` pour mettre au milieu les basses fréquences.

Justifiez les phénomènes observés. Faites varier M. Que pouvez-vous dire sur la décroissance du spectre ? Combien de coefficients faut-il garder pour avoir une approximation d'ordre 10^{-2} ?

► **Question 3 :** Calculer la transformée de Fourier (`fft2`) d'une image u (par example `lena`) et afficher son module sur une échelle logarithmique (appliquer la fonction `fftshift` pour que la fréquence zéro soit située au centre). Annulez les coefficients de Fourier des hautes fréquences, puis reconstruisez l'image u_{M^2} en ne gardant que M^2 coefficients de Fourier, avec M^2 en paramètre. Attention, cette image peut être à valeurs complexes, donc on visualisera seulement sa partie réelle.
Quels artefacts observez-vous ? Comment explique-t-on la présence des oscillations ? Faites varier M et trouvez une valeur optimale. Justifiez votre choix.

2.2 Approximation non-linéaire de Fourier

On se demande si l'erreur diminuera plus vite en considérant un autre ensemble de coefficients I_M de taille M, au lieu des coefficients qui correspondent aux basses fréquences. On propose donc de trouver une approximation

$$u_T = \sum_{|<u,e_m>|\geqslant T} \langle u, e_m \rangle \, e_m$$

où T est un paramétre.

► **Question 4 :** Pour les signaux et les images d'avant, calculer la meilleure approximation avec M termes, dans la base de Fourier. (Il s'agit de trouver le seuil T tel qu'on ait exactement M coefficients qui vérifient $|< u, e_m >| \geqslant T$). Qu'observez-vous ? Commentez les résultats. Calculez et affichez à l'échelle logarithmique l'approximation linéaire d'erreurs $e[M]^2$.

NB : Ce morceau de code pourrait être pratique.

```
% trouver le bon seuillage T en fonction de M
   M = 50^2;
   a = sort( abs(u_fft(:)) );
   T = a(end-M);
   uT_fft = u_fft .* (abs(u_fft)>T);
```

2.3 Transformée en cosinus

Souvent il est intéressant de faire une symétrisation d'un signal, avant de faire sa transformée de Fourier (Pourquoi ?). La transformée en cosinus discrète (DCT) est similaire à l'approximation de Fourier, à l'exception qu'elle utilise une condition de bord symétrique, au lieu des conditions périodiques. Les produits scalaires dans cette base sont calculés à l'aide de *Fast Cosine Transform algorithm*, qui nécessite $O(Nlog(N))$ opérations. Les fonctions matlab sont :

```
dct.m % pour les signaux  1D
dct2.m % pour les images
idct.m % la fonction inverse.
idct2.m % la fonction inverse.
```

► **Question 5 :** Quels avantages voyez-vous à la DCT par rapport à la TFD ?

Appliquez la DCT aux signaux 1D et images d'avant. Afficher le signal d'origine et l'amplitude des coefficients DCT. Notez que les basses fréquences sont dans le coin supérieur gauche.

► **Question 6 :** Pour les signaux et les images d'avant, calculez la meilleure approximation avec M termes, dans la base de cosinus. Calculez et affichez à l'échelle logarithmique l'approximation linéaire d'erreurs $e[M]^2$. Comparez les approximations de Fourier et DCT.

2.4 Transformée en cosinus locale. Algorithme JPEG

JPEG est un format de représentation des images. L'algorithme sépare l'image en blocs 8×8 et calcule la DCT sur chacun de ces blocs. Après, les hautes fréquences, celles auxquelles l'oeil humain est très peu sensible, sont ramenénées à 0. L'intérêt est qu'une longue suite de zéros nécessite très peu de place pour être stockée dans un fichier. La fonction qui permet de coder en JPEG l'image est la suivante :

```
[comp_Img,file_size] = jpeg_comp(Image,quality);
```

► **Question 7 :** Etudiez l'effet de ce codage sur plusieurs images en choisissant différents paramétres de qualité q (ex. 3,10,50,80,100). Ce type de codage produit-il des pertes sur l'image ? Quels artefacts apparaissent dans l'image codée ?

3 Du bruit dans les images

Dans cette partie, on s'intéressera à la réduction du bruit dans les images.

Extrait de Wikipedia : Le bruit d'image est la présence d'informations parasites qui s'ajoutent de façon aléatoire aux détails de la scène photographiée numériquement. Il est plus particulièrement visible dans les zones peu éclairées, où le rapport signal/bruit est faible, mais aussi dans les parties uniformes telles qu'un ciel bleu. Il a pour conséquence la perte de netteté dans les détails.

3.1 Bruit blanc Gaussien

Le modèle le plus simple d'image bruitée est le modèle de bruit additif. Cela consiste à considérer l'image bruitée comme l'addition d'une image nette (idéale) et d'une réalisation aléatoire de moyenne nulle. Plus précisément, nous considérons :

$$v = u + n \quad \text{avec} \quad n \sim N(0, \sigma^2 Id)$$

où u est l'image originale, v est l'image bruitée et n est le bruit i.i.d. (indépendant et identiquement distribué, c'est-à-dire que le bruit en un pixel est indépendant et a la même loi de probabilité que le bruit en n'importe quel autre pixel), gaussien (la loi du bruit de chaque pixel est gaussienne) et indépendant de l'image originale. Nous supposerons que la moyenne du bruit est nulle et de variance σ^2 (ce qui représente sa puissance). Ce modèle est illustré dans les Figures 1 et 2.

► **Question 8 :** A l'aide de la commande `randn` créez un signal 1D d'un bruit gaussien de moyenne nulle et de variance $\sigma^2 = 3$. Quel allure a le spectre de ce bruit ?

Affichez dans la même figure le signal 1D original, le bruit blanc gaussien et le signal 1D bruitée.

► **Question 9 :** Générez une image bruité avec un bruit Gaussien de variance $\sigma^2 = 10$ et afficher dans une figure l'image originale, le bruit et l'image bruité. Commentez les résultats.

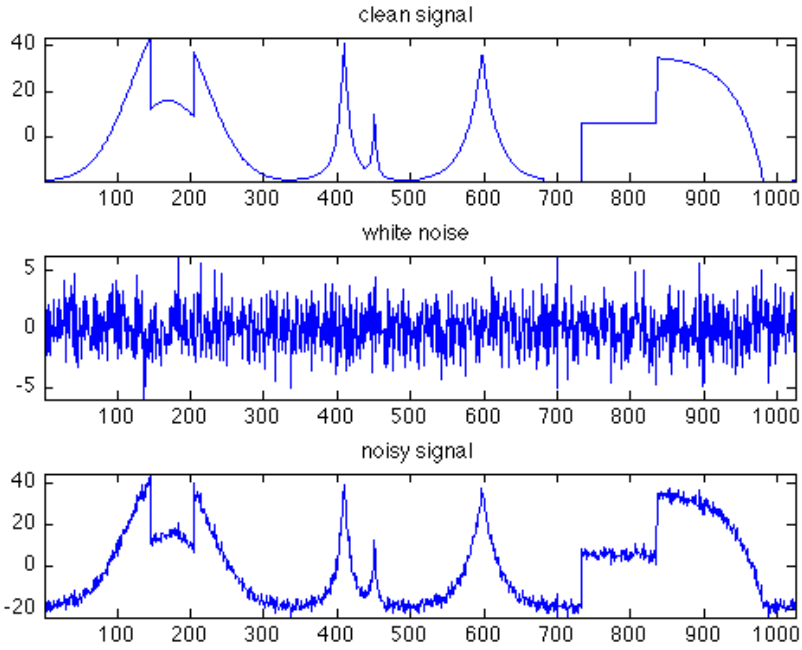

FIGURE 1 – Obtention d'un signal bruité.

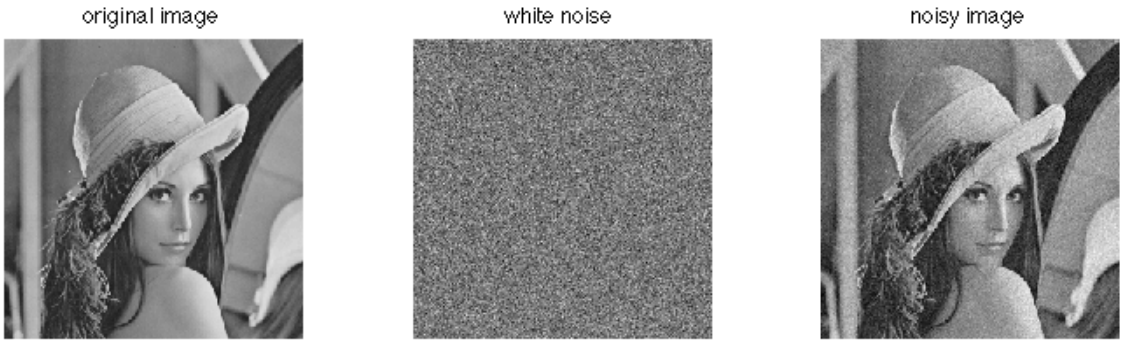

FIGURE 2 – Obtention d'une image bruitée.

3.2 Débruitage des images avec des méthodes linéaires

Un opérateur linéaire et invariant par translation est nécessairement une convolution avec un noyau. Il correspond à une opération diagonale dans le domaine de Fourier, qui multiplie chaque coefficient de Fourier du signal/image bruité avec le coefficient de Fourier de la transformée du filtre. Nous utilisons un lissage simple pour supprimer les fréquences élevées causées par le bruit.

$$u = u_b \star K \tag{1}$$

et on cherche à trouver une estimation du signal net u à partir de u_b (signal bruité).

► **Question 10 :** Ecrivez l'équation 1 dans le domaine fréquentiel.
Dans la pratique, on utilise un filtre gaussien h, de moyenne 0 et d'écart-type s. Notez que la DFT $\hat{h}$ est aussi gaussienne. Quelle est sa variance ? Essayez et décrivez le code ci-dessous.

```
u = load_signal('Piece-regular', 200);
N=length(u);

s=3;
g = fspecial('gaussian', [N,1], s);
g=fftshift(g);

smoothu = real(ifft(fft(g).*fft(u)));

plot(1:N,smoothu, 1:N, u)
```

► **Question 11 :** Adaptez le code précédent pour appliquer la convolution pour débruiter les images (bruitées) obtenues dans la question 9. Affichez le signal original, le signal bruité et le signal débruité par convolution. Comparez les coefficients de Fourier des signaux bruité et débruité, à l'echelle logarithmique. Commentez les résultats. Quelle valeur de s donne le résultat le plus visuellement agréable ?

► **Question 12 :** Comment utiliseriez-vous l'approximation linéaire / non-linéaire (2.1 et 2.2) de Fourier pour faire du débruitage ? Quels problèmes peuvent se produire ?