Analyse de Fourier et hilbertienne
ENS de Cachan
Département de Mathématiques Vendredi 25 Octobre 2013

Feuille de TP numéro 1

1 Consignes pour le rapport de TP

Cette feuille de TP comporte 16 questions. Les réponses et les résultat d'experiences (images, figure) devront être rassemblés dans un rapport **individuel** à rendre au plus tard le **Lundi 18 Novembre 2013**. Une pénalité de **deux points par jour** sera appliquée en cas de retard.

1.1 Format du rapport

Les rapports de TP seront rédigés avec LaTeXet rendus au format `.pdf`. LaTeX est le langage standard **incontournable** pour la rédaction de documents mathématiques (et dans bien d'autres disciplines scientifiques).
Pour vous aidez dans la rédaction de ce rapport un modèle de fichier LaTeX(extension `.tex`) téléchargeable à l'adresse `http://dev.ipol.im/~facciolo/hilbert/modele_latex.zip`. Des conseils concernant les logiciels à utiliser ainsi que des liens utiles accompagnent ce fichier.

1.2 Remise du rapport

Le rapport (PDF) du TP doit être rendu au plus tard le **Lundi 18 Novembre 2013**. Il sera envoyé à l'adresse email **gabriele.facciolo@cmla.ens-cachan.fr**. Le sujet de l'email doit être **Rapport du TP1 de *Nom Prénom*** et le fichier `.pdf` doit être intitulé ***nom_prenom*_tp1.pdf**.

2 Mise en route

Ouvrez un terminal (fenêtre permettant d'entrer des lignes de commandes, accessible depuis le menu : `Menu KDE -> Applications -> Systèmes -> Terminal`). Tapez les commandes suivantes pour créer un dossier `tp_fourier` et se placer dans ce dossier :

```
mkdir tp_fourier
cd tp_fourier
```

Dans le même terminal tapez les commandes suivantes pour télécharger les fichiers nécessaires dans votre dossier `tp_fourier`.

```
wget dev.ipol.im/~facciolo/hilbert/tp1.zip
unzip tp1.zip
cd tp1
```

Toujours dans le même terminal lancez Matlab grâce à la commande `matlab`. Vous écrirez les différentes instructions dans des fichiers du type `mon_script.m` et vous lancerez les différentes instructions en surlignant avec la souris les passages à évaluer et en appuyant sur la touche `F9` (ou alors clique droit et `Evaluate selection`).

3 Notions de base sur les images numériques

Nous travaillerons exclusivement avec des images en niveaux de gris.

3.1 Niveaux de gris

Une image numérique standard est un tableau de pixels (vu comme une matrice), dont les coefficients sont des entiers compris entre 0 et 255. Il n'y a donc que 256 valeurs possibles pour les niveaux de gris. Il est très facile de créer des images simples avec Matlab. Par exemple le script suivant crée une image de taille 500×500 représentant un carré noir sur fond blanc :

```
u = 255*ones(500,500);
u(201:400,101:300) = zeros(200,200);
```

Pour visualiser une image `u` (ou n'importe quelle tableau) nous utiliserons les instructions suivantes (le problème de la visualisation d'une image est abordée en détail dans la partie suivante) :

```
imagesc(u, [0,255]);
colormap gray;
axis image;
```

► **Question 1 :** Créez une image de taille 256×256 qui soit constante sur chaque colonne et dont le niveau de gris des colonnes varie progressivement de 0 à 255.
Aide Matlab : Si `a` et `b` sont deux entiers alors `(a:b)` est le vecteur ligne `[a,a+1,...,b-1,b]`.

► **Question 2 :** Quoi représente l'image suivante ?
Par quel suite de commandes peut-on afficher l'image 9×9 suivante ?

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	120	120	120	120	120	0	0
0	0	120	255	255	255	120	0	0
0	0	120	255	255	255	120	0	0
0	0	120	255	255	255	120	0	0
0	0	120	120	120	120	120	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

3.2 Visualisation des images

3.2.1 Plage de valeurs

La manipulation mathématique des images amène à considérer des images dont les niveaux de gris sont des nombres réels (ou du moins la représentation informatique des nombres réels,

c'est-à-dire des nombres flottants). En effet, toutes les images ne peuvent pas être limitées à des valeurs entières entre 0 et 255. Néanmoins il faut garder à l'esprit que toutes les images affichées à l'écran sont des images standards qui respectent ces contraintes.

Il y a alors deux méthodes pour visualiser une image `u` à valeur réelle. Soit l'image `u` est proche d'une image standard et alors on arrondit chaque valeur `u(i,j)` à l'entier le plus proche entre 0 et 255 (ce que réalise la fonction Matlab `uint8`). Soit on effectue un changement de variable affine qui ramène l'intervalle $[\min(u), \max(u)]$ sur l'intervalle $[0, 255]$, et on effectue ensuite une discrétisation des valeurs avec `uint8`. Cette deuxième méthode est particulièrement adaptée pour visualiser des images dont la plage de valeurs n'est pas standard (comme la somme de deux images par exemple).

3.2.2 Lecture et écriture des fichiers d'images

La fonction Matlab `imread` permet de lire des fichiers images :

```
u = imread('tile.png');
```

Ainsi obtenue, `u` a des valeurs de type `uint8` (*i.e.* des entiers compris entre 0 et 255). On peut le vérifier par la commande suivante :

```
whos u
```

Pour pouvoir faire des calculs sur cette image, il faut la convertir en nombre flottant grâce à la fonction `double` :

```
u = double(u);
% ou directement
u = double( imread('tile.png') );
```

Pour enregistrer une image `u` dans un fichier d'image il faut faire la conversion inverse et utiliser la fonction `imwrite` :

```
imwrite(uint8(u),'nouvelle_image.png');
```

ATTENTION : Matlab écrase les fichiers existants sans prévenir !

Alternativement, pour enregistrer le contenu d'une fenêtre, vous pouvez aussi utiliser l'option Enregistrer sous.

3.2.3 Visualisation des images avec Matlab

imagesc : La fonction qu'on utilisera pour visualiser des images avec Matlab est `imagesc(im, [rang])`. Le paramètre `im` est le nom d'une variable contenant un tableau à valeurs réelles ou entières. Le paramètre optionnel `rang` indique la plage de valeurs à montrer ; Par exemple si `rang = [ -10, 10 ]`, la valeur -10 correspondra au noir, la valeur 10 au blanc et les valeurs intermédiaires à differents niveaux de gris (si colormap est "gray"), toutes les valeurs plus petites que -10 ou plus grandes que 10 seront saturées et donc montrées comme noire ou blanche respectivement. Si le paramètre `range` n'est pas spécifié, la gamme sera automatiquement fixé à `[min (im(:)), max (im(:))]`. Consulter l'aide (`help imagesc`) pour plus d'information.

colormap : La gamme de couleurs utilisées pour l'affichage de l'image s'appelle `colormap`, le colormap par défaut pour une nouvelle figure est "jet" (arc en ciel), la commande `colormap gray;` affiche les images avec une gamme de niveaux de gris. Consulter l'aide (`help colormap`) pour plus d'information.

► **Question 3 :** Chargez l'image `water.png` dans la variable `im`, et testez les commandes suivantes :

```
imagesc(im);           colormap gray; axis image;
imagesc(im,[0,255]);   colormap gray; axis image;
imagesc(im,[0, 100]); colormap gray; axis image;
imagesc(im,[0,255]);   colormap jet;  axis image;
```

Comment expliquez-vous les différences observées entre les visualisations ?

3.2.4 Visualisation conservant les proportions de l'image

axis : La commande `axis` contrôle l'aspect des axes. En particulier `axis image` rend les pixels carrés. Consulter l'aide (`help axis`) pour plus d'information.

Essayez avec l'exemple ci-dessous :

```
u = double(imread('water.png'));
imagesc(u);
colormap gray;
axis image;
axis off;
```

4 Transformée de Fourier discrète (TFD)

4.1 En dimension 1

► **Question 4 :** Lire l'aide de la commande `fft` (`doc fft` où `help fft`). La définition de la TFD dans matlab diffère de celle du cours. Précisez.

En réalité, ce ne sont que des différrences de conventions. Les propriétés de la TFD restent inchangées. Par soucis de simplicité nous utiliserons pour ces TPs la définition de Matlab.

Créez un signal discret u de longueur paire $N = 512$ représentant un créneau :

```
% Longueur du signal :
N=256;
ind = 1:N;
% Creation du signal :
u = zeros(1,N);
u( (N/4+1) : (3*N/4) ) = ones(1,N/2);
% Affichage du signal :
figure(1);
plot(ind, u, 'b*-'); axis([0, N, -0.1, 1.1]);
```

► **Question 5 :** Calculez le module de Fourier `modu` du signal `u` (utilisez `abs`) et visualisez-le avec la commande `plot(ind, modu); axis tight;`. Le graphique obtenu est-il conforme à vos attentes ? Où se trouve la fréquence nulle ? Recommencez en utilisant la commande `fftshift`.

4.2 En dimension 2

4.2.1 TFD et TFD inverse

La TFD en dimension 2 est donnée par la fonction `fft2`, la transformation inverse est donnée par `ifft2`. Lisez la section "Algorithm" de l'aide en ligne de cette fonction (`doc fft2`). Remarquez que la propriété de séparabilitée de la transformation est mise en œuvre.

Remarque : En pratique, il est possible que la TFD inverse de la TFD d'une image réelle ne soit pas rélle. A cause des erreurs d'arrondi, la partie imaginaire peut ne pas être strictement nulle. Dans ce cas, on pourra extraire la partie réelle avec la commande `real`.

4.2.2 Visualisation du module

Pour visualiser le module d'une image nous utiliserons la fonction `vis_mod_img` fournie dans l'archive `tp1.zip`. Consultez l'aide de cette fonction (`help vis_mod_img`).

► **Question 6 :** Lisez le fichier `vis_mod_img.m`. En utilisant la fonction `subplot`, visualisez sur une même figure le module de l'image `room.png` avec les trois jeux de paramètres `(d,opt) = (0,0), (1,0), (1,1)`. Expliquez l'intérêt du seuillage. Quelle propriété de symétrie du module peut-on observer ?

On ne visualisera pas la phase d'une image car son interprétation est assez difficile. Néanmoins, comme le montre les expériences qui suivent, la phase d'une image est au moins aussi importante que son module du point de vue de l'information qu'elle contient.

4.2.3 Echange du module et de la phase de deux images

Calculez et visualisez les deux images obtenues en échangeant le module et la phase des images `building.png` et `tile.png`. Recommencez cette expérience avec les couples d'images (`sand.png`, `water.png`) et (`tile.png`, `sand.png`).

► **Question 7 :** Quelles conclusions générales peut-on en tirer sur les contenus visuels du module et de la phase d'une image ? (On distingue deux types de "contenu visuel" : les contours géométriques et les textures.)

5 Effet de Gibbs et zooms avant

5.1 Polynôme trigonométrique associé à un signal discret

Soit u_k, $k = 0, \ldots, N-1$ un signal discret et $\tilde{u}_n$, $n = -\frac{N}{2}, \ldots, \frac{N}{2} - 1$ sa TFD. On suppose que N est un entier pair. On rappelle que le polynôme trigonométrique associé à u est le polynôme P_u défini par

$$P_u(x) = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \tilde{u}_n e^{\frac{2i\pi nx}{N}}.$$

Le polynôme P_u vérifie $P_u(k) = u_k$ pour tout $k = 0, \ldots, N-1$. Alors P_u est un polynôme trigonométrique qui interpole les valeurs du signal u. En revanche, P_u n'est pas nécessairement

à valeur réelle ce qui n'est pas satisfaisant en pratique.

► **Question 8 :** Montrer que P_u est à valeur réelle si et seulement si $\tilde{u}_{-\frac{N}{2}} = 0$. Noter également que pour tout signal u, la partie réelle de P_u est un polynôme trigonométrique qui interpole les valeurs de u.

En pratique, on utilise la partie réelle de P_u pour interpoler le signal u. Il en est de même en dimension 2.

5.2 Effet de Gibbs sur les images

L'effet de Gibbs est le phénomène suivant : les projections basses fréquences d'un signal présentant des discontinuités ont des comportements oscillants autour des discontinuités. Nous allons mettre en évidence ce phénomène à partir d'une image discrète (voir le polycopié pour le cas des signaux unidimensionnels).

Créez une image `rectangle` de taille 512×512 repésentant un rectangle gris claire (niveau de gris 200) sur un fond gris foncé (niveau de gris 50), voire la section 3.1. Calculez la TFD de l'image `rectangle`. Annulez toutes les fréquences qui ne sont pas dans le carré central de taille `n`×`n` avec `n = 64` et calculez l'image `rectbf` correspondant à ce spectre ne comportant que des basses fréquences. On pourra visualiser cette image en niveau de gris mais aussi en tant que surface grâce aux commandes suivantes :

```
surf(rectbf, 'EdgeColor', 'none');
colormap(jet); colorbar; axis tight;
```

► **Question 9 :** Commentez les résultats observés.

5.3 Zooms avant

Le fichier d'instructions `zooms_avant.m` vous permet de comparer trois méthodes d'interpolation pour zoomer une image.

- La première est la méthode "pixel le plus proche".
- La deuxième repose sur l'interpolation bilinéaire.
- La troisième est le zoom par zéro-padding (extension du spectre avec des hautes fréquences nulles).

► **Question 10 :** Testez les zooms sur plusieurs images. Décrivez brièvement les défauts de chacune des méthodes. Expliquez le lien entre les défauts du zoom par zéro-padding et l'effet de Gibbs.

6 Translation non entière et rotation

6.1 Translation non entière

Soit $(u_{k,l})$ une image carrée de taille $N \times N$ et $(\tilde{u}_{m,n})$ sa TFD. On pose

$$P_u(x,y) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} \tilde{u}_{m,n} e^{\frac{2i\pi mx}{N}} e^{\frac{2i\pi ny}{N}}$$

le polynôme trigonométrique représenté par u.

► **Question 11 :** Soit a et b deux réels. Exprimez la TFD de $(k,l) \mapsto P_u(k-a, l-b)$ en fonction des $\tilde{u}_{m,n}$. Codez une fonction `v = translation_fourier(u,a,b)` qui calculera la translation de l'image u par le vecteur (a,b) (**Aide Matlab :** l'opération multiplication élément par élément de deux matrices `A` et `B` s'écrit `A.*B`). Cette fonction retournera la partie réelle de la translation de P_u, en accord avec les remarques de la section 5.1.
Testez votre fonction en faisant une translation sous-pixelienne de (3.3 ; 50.7) pixels d'une image.

6.2 Rotation d'une image *(devoir à la maison)*

La rotation d'une image par la méthode de Yaroslavsky se base sur la décomposition suivante :

$$R_{-\theta} := \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix}$$

$$\begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & \tan\frac{\theta}{2} \\ 0 & 1 \end{pmatrix}}_{:=T(\theta)} \underbrace{\begin{pmatrix} 1 & 0 \\ -\sin\theta & 1 \end{pmatrix}}_{:=S(\theta)} \begin{pmatrix} 1 & \tan\frac{\theta}{2} \\ 0 & 1 \end{pmatrix}$$

$$R_{-\theta} := T_\theta S_\theta T_\theta$$

Remarquez que le changement de variable donné par $T(\theta)$ revient à translater chaque colonne : $u(k,l)$ devient $u\left(k + \tan\left(\frac{\theta}{2}\right) l, l\right)$. De même, le changement de variable donné par $S(\theta)$ revient à translater chaque ligne : $u(k,l)$ devient $u\left(k, l - \sin(\theta)k\right)$.

Les consignes qui suivent ont pour but de vous aider à coder l'algorithme de rotation de Yaroslavsky.

► **Question 12 :** Codez une fonction `v = translation_colonnes(u,t)` qui translate chaque colonne `k` de `u` par le vecteur `t(k)` en utilisant la TFD unidimensionnelle (**Aide Matlab :** si `A` est une matrice, alors `fft(A)` renvoie la matrice des TFD des colonnes de `A`). Testez cette fonction avec des vecteurs de translation non entiers.

► **Question 13 :** Codez une fonction `v = translation_lignes(u,t)` qui translate chaque ligne `k` de `u` par le vecteur `t(k)`. On va utiliser la fonction `translation_colonnes` et les transpositions de matrice (**Aide Matlab :** la transposée de `A` est `A'`).

► **Question 14 :** Coder la fonction `v = rotation_fourier(u,theta)` en effectuant trois appels successifs aux fonctions `translation_colonnes` et `translation_lignes`. Commencez par copier l'image `u` au centre d'une image nulle deux fois plus grande pour éviter les problèmes de bords. Imposez à chacune des translations de ne pas déplacer la colonne (ou la ligne) du centre de l'image. Illustrez la méthode en visualisant chacune des images intermédiaires.
Testez votre fonction en faisant une rotation de 13° d'une image.

7 Zoom arrière et aliasing *(devoir à la maison)*

Le fichier d'instructions `aliasing.m` illustre le phénomène d'aliasing (ou repliement de spectre). Suivre les instructions dans le fichier et répondre aux questions suivantes.

► **Question 15 :** Dans la partie 1). Qu'observe-t-on ? Comment cela se traduit-il au niveau du spectre ?

► **Question 16 :** Dans la partie 2). Expliquez précisément le comportement des deux ondes sous-échantillonnées. Quelle est la fréquence de l'onde de l'image sous-échantillonnée par 4 ? Qu'observe-t-on ? Comment cela se traduit-il au niveau du spectre ?