# ESIR 1 / Algorithmique et Complexité : TP 2
# Transformée de Fourier rapide (deuxième partie)

Pierre Maurel, pierre.maurel@irisa.fr

Dans la première partie de ce TP, vous avez implémenté une version récursive du calcul des coefficients de Fourier discrets d'un vecteur. Lors de ce TP, nous allons voir deux applications pratiques de la TFD : l'analyse fréquentielle d'un signal et la compression d'images.

Vous préparerez un compte-rendu répondant aux différentes questions, exposant vos conclusions et illustrant vos résultats avec les images qui vous semblent significatives.

## 1 Représentation fréquentielle d'un signal

### 1.1 Signal à 1 dimension

#### 1.1.1 Cas général

On a défini la transformée de Fourier discrète (TFD) d'un vecteur $\mathbf{a} = (a_0, \ldots, a_{n-1})$ ainsi :

$$TFD(\mathbf{a}) = \mathbf{y} \quad \text{avec} \quad y_k = \sum_{j=0}^{n-1} a_j \, \omega_n^{kj}.$$

Jusqu'à maintenant, dans le TD et la première partie du TP, le vecteur $\mathbf{a}$ était considéré comme représentant les coefficients d'un polynôme (qu'on voulait multiplier par un autre polynôme en $\Theta(n \log n)$).

Ici ce vecteur représentera l'**échantillonnage temporel** d'un certain signal. La figure 1 illustre ce qu'on entend par "échantillonnage" : un signal temporel $\mathbf{a}(t)$ est représenté approximativement par un nombre fini de valeurs pour différents $t_i$ :

$$\mathbf{a}(t) \text{ est représenté par } (a_0, a_1, \ldots, a_n) = (\mathbf{a}(t_0), \mathbf{a}(t_1), \ldots, \mathbf{a}(t_n))$$

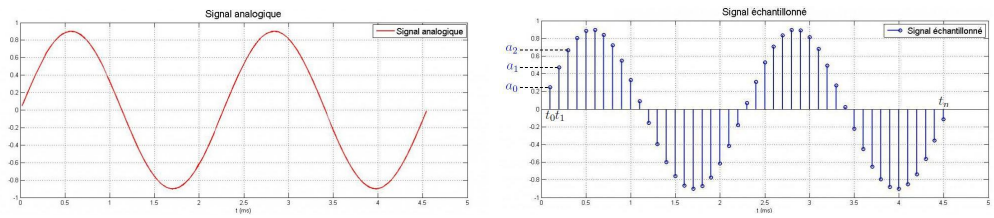



FIGURE 1 – Exemple d'échantillonnage (le signal d'origine est un "la" à 440 Hz)

On a déjà vu que $TFD^{-1}(\mathbf{y}) = \frac{1}{n}(TFD(\mathbf{y}^*))^*$. On peut montrer que la transformée de Fourier discrète inverse peut aussi s'écrire :

$$TFD^{-1}(\mathbf{y}) = \mathbf{a} \quad \text{avec} \quad a_j = \frac{1}{n}\sum_{k=0}^{n-1} y_k\, \omega_n^{-kj} \quad \text{et} \quad \omega_n = e^{\frac{2\pi i}{n}}$$

On a donc

$$\boxed{\mathbf{a}(t) = \frac{1}{n}\sum_{k=0}^{n-1} y_k\, \mathbf{e_k}(t)}$$

en posant $\mathbf{e_k}(t) = e^{\frac{2\pi i k}{n}\, t}$, pour $k = 0, \ldots, n-1$.

Calculer la transformée de Fourier d'un signal $\mathbf{a}$ correspond donc à décomposer $\mathbf{a}$ sur les fonctions $\{\mathbf{e_k}(t)\}_{k=0,\ldots,n-1}$, les coefficients $y_k$ de la TFD correspondant alors à l'importance de chaque $\mathbf{e_k}(t)$ dans $\mathbf{a}$.

### 1.1.2 Cas d'un signal réel

Dans le cas où l'on sait que $a_j \in \mathbb{R}$ (c'est le cas qui nous intéresse ici), on peut montrer que la TFD de $\mathbf{a}$ est symétrique. En effet on a $\cos(\theta) = \frac{e^{i\,\theta} + e^{-i\,\theta}}{2}$. Donc :

$$\cos(\frac{2\pi k}{n}\, t) = \frac{1}{2}\,\mathbf{e_k(t)} + \frac{1}{2}\,\mathbf{e_{-k(t)}} = \frac{1}{2}\,\mathbf{e_k(t)} + \frac{1}{2}\,\mathbf{e_{n-k}(t)}$$

Dans la suite, on considérera donc que la première moitié de la TFD donne directement les facteurs de la décomposition du signal sur les cosinus de différentes fréquences. La figure 2 illustre cette interprétation.

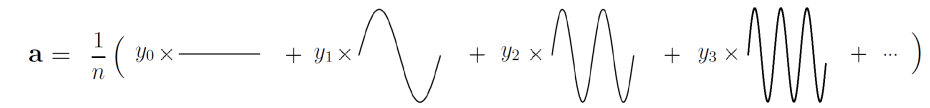



FIGURE 2 – Décomposition de $\mathbf{a}$ comme combinaison linéaire de différents cosinus. On peut montrer que la première moitié du vecteur $\mathbf{y} = TFD(\mathbf{a})$ donne les facteurs de la combinaison linéaire.

**Exercice 1** Pour $n = 16$, calculez et affichez la TFD des vecteurs suivants, interprétez tous les résultats.

- vecteur constant $\mathbf{a} = (a, \ldots, a)$. Testez différentes valeurs de $a \in \mathbb{R}$.
- sinusoïde pure : $\mathbf{a} = \big(\cos(\frac{2\pi k}{n}0), \cos(\frac{2\pi k}{n}1), \ldots, \cos(\frac{2\pi k}{n}(n-1))\big) = \frac{1}{2}\mathbf{e_k} + \frac{1}{2}\mathbf{e_{n-k}}$. Vous testerez différentes valeurs de $k \in \{0, \ldots, n-1\}$.
- somme de 2 sinusoïdes : $\mathbf{a} = (a_0, \ldots, a_{n-1})$ avec $a_t = \cos(\frac{2\pi}{n}\, t) + \frac{1}{2}\,\cos(\frac{2\pi\, 3}{n}\, t)$.

— la fonction : $\mathbf{a} = (a_0, \ldots, a_{n-1})$ avec $a_t = 4 + 2\,\sin(\frac{2\pi\,2}{n}\,t) + \cos(\frac{2\pi\,7}{n}\,t)$.

La figure ci-dessous [1] résume graphiquement le comportement de la transformée de Fourier en 1D. Les applications de la transformée de Fourier sont multiples : analyse spectrale d'enregistrement sonore, imagerie médicale (une image IRM est par exemple directement acquise dans l'espace de Fourier et reconstruite par transformée de Fourier inverse), défloutage d'une image, compression d'images (voir partie suivante).

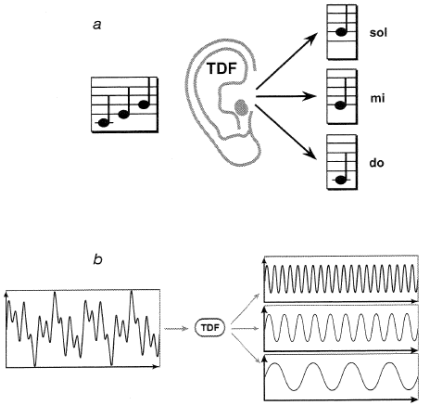



Fig. 6-18.
*(a)* Un accord do-mi-sol est entendu par l'oreille, qui reconnaît les différentes notes (fréquences élémentaires) qui le composent (le do, le mi et le sol).
*(b)* Représentation graphique : à partir du signal composite, la transformée de Fourier permet de reconnaître les trois fréquences élémentaires.

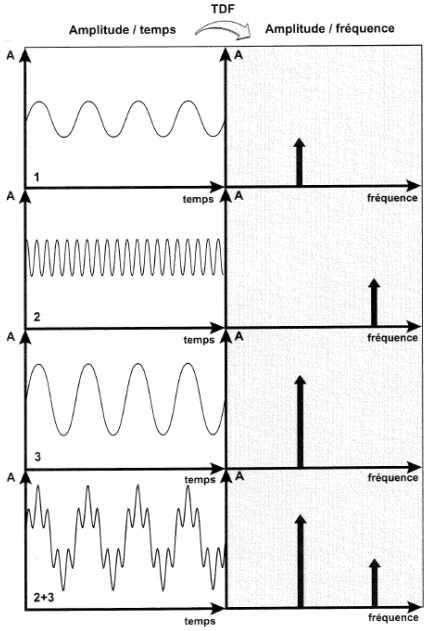



Fig. 6-19. — *Représentation graphique* : la transformée de Fourier (TDF) permet de passer de l'amplitude par rapport au temps à l'amplitude par rapport à la fréquence.
1. Graphe de la fonction **sin(x)** et sa représentation par un vecteur dans le domaine fréquentiel après TDF.
2. Fonction **sin (5x)** : la fréquence étant plus élevée, le vecteur est décalé vers les plus hautes fréquences.
3. Fonction **2sin(x)** : la fréquence est la même qu'en 1, mais l'amplitude est deux fois plus élevée.
4. On a additionné les fonctions 2 et 3 soit **sin (5x) + 2sin(x)** : le graphe représentant cette somme dans le domaine temporel ne permet pas de différencier les 2 fréquences qui la composent; après TDF, les 2 fréquences sont « visibles ».

## 1.2 Signal à 2 dimensions

### 1.2.1 Définition de la TFD en 2 dimensions

On va maintenant étendre le calcul de la TFD à des signaux à deux dimensions (représentant des images par exemple). Une image $I$ en niveaux de gris peut être vue comme un tableau à deux dimensions dont les éléments $I(k,l)$ représentent la valeur des pixels $(k,l)$. Pour une image de $I$ taille $n \times n$, la TFD de $I$ est également un tableau $F$ à deux dimensions de même taille que $I$ et dont les éléments $F(u,v)$ sont donnés par :

$$F(u,v) = \frac{1}{n} \sum_{k=0}^{n-1} \sum_{l=0}^{n-1} I(k,l) e^{-2i\pi(uk)/n}\, e^{-2i\pi(vl)/n} \tag{1}$$

1. tirée de "Comprendre l'IRM" de B.Kastler et D.Vetter

On peut montrer que la TFD 2D d'une image $I$ peut être obtenue en 3 étapes :

1. TFD en 1D sur chaque ligne de $I$
2. TFD en 1D sur chaque colonne du résultat de la première étape
3. Diviser tous les éléments par $n$

Vous trouverez sur l'ENT un squelette de programme `FFT_2D.java`, contenant déjà l'implémentation de la FFT (à partir de la méthode ci-dessus, utilisant donc votre implémentation de la FFT en 1D) et de la FFT inverse en 2D. Vous trouverez également un fichier `CpxImg.java`. La classe `CpxImg` permet de representer une image carré ($n \times n$) dont chaque élément est un complexe. Vous regarderez le code pour identifier les différentes méthodes disponibles.

Vous aurez également besoin des fichiers `BytePixmap.java` et `Pixmap.java` qui permettent d'ouvrir les images de type `pgm` et de les sauver dans un objet `BytePixmap` par la commande :

```
BytePixmap p = new BytePixmap("image.pgm");
```

Inversement, pour enregistrer un `BytePixmap p` dans une image `pgm` :

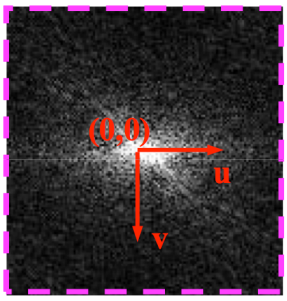
```
p.write("image.pgm");
```

Vous disposez de plus dans la classe `CpxImg` d'un constructeur permettant de construire un `CpxImg` à partir d'un `BytePixmap`, ainsi que d'une fonction `BytePixmap convert_to_BytePixmap()` qui convertit un `CpxImg` en `BytePixmap`.

**Exercice 2** Vérifiez le bon fonctionnement des fonctions fournies (et donc indirectement de vos implémentations 1D) en calculant le TFD de l'image `tigre_512.pgm`, puis en effectuant la transformée de Fourier inverse du résultat et en sauvant (et en affichant) le résultat obtenu dans une nouvelle image.

### 1.2.2 Interprétation fréquentielle de la TFD 2D

Par souci de visualisation, il est plus facile d'interpréter la TFD en 2D après avoir décalé les éléments du tableau de $n/2$ indices en horizontal et de $n/2$ indices en vertical (modulo $n$ pour ne pas sortir du tableau).



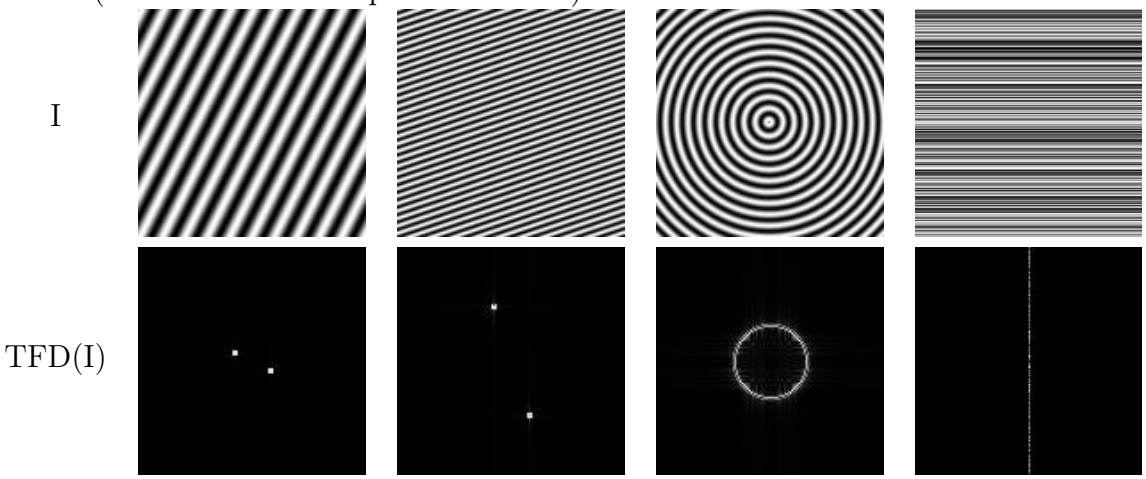

Ainsi, de la même manière qu'en 1D le premier coefficient de Fourier valait la somme des coefficients d'origine, ici c'est le coefficient central $F(\frac{n}{2}, \frac{n}{2})$ qui vaut également la somme des pixels de l'image.

De la même façon que pour la TFD en 1D, la TFD 2D permet de faire une analyse fréquentielle des images. Le coefficient $F(u, v)$ représente les oscillations spatiales dans la direction donnée par le vecteur $((\frac{n}{2}, \frac{n}{2}) \rightarrow (u, v))$ contenues dans l'image. Par exemple, les coefficients $F(u, \frac{n}{2})$ vont représenter les oscillations verticales contenues dans l'image, la fréquence de ces motifs étant proportionnelle à $1/u$.

Ainsi, les coefficients $|F(u, \frac{n}{2})|$ de la TFD d'une image seront d'autant plus forts que l'image comporte des oscillations verticales.

Pour résumer :
- $|F(\frac{n}{2}, \frac{n}{2})|$ est proportionnel à la moyenne de l'image
- $|F(u, v)|$ représente les motifs périodiques de l'image dans la direction donnée par le vecteur $((\frac{n}{2}, \frac{n}{2}) \rightarrow (u, v))$ de fréquence proportionnelle à : $\dfrac{1}{\sqrt{(u - \frac{n}{2})^2 + (v - \frac{n}{2})^2}}$

Dans la matrice représentant la TFD d'une image, les basses fréquences (motifs homogènes, peu de variations) sont donc au centre de l'image, et les hautes fréquences (zones avec beaucoup de variations) sont sur les bords.




**Exercice 3** Regardez l'image `mire1.pgm`. Selon vous, quelle sera la forme du module de sa TFD ? Vérifiez. Faites de même avec les fichiers `mire2`, `mire3` et `fingerprint.pgm`. Analysez également la TDF de l'image `barbara`.

# 2 Application à la compression d'images

La représentation fréquentielle des images est utilisée dans les standards de compression d'image (comme JPEG par exemple). L'objectif des standards de compression est de réduire la taille des images en les détériorant le moins possible. Pour cela, ils s'appuient souvent sur une propriété du système visuel humain : l'oeil est beaucoup moins sensible aux hautes fréquences qu'aux basses fréquences (une modification dans une image sera beaucoup plus visible sur une zone uniforme que sur une zone avec de fortes variations).

Une idée de base de la compression d'images est donc la suivante : une fois que l'on a obtenu une représentation fréquentielle de notre image, on décide de ne garder

qu'un certain nombre de coefficients de celle-ci. Ce seront ces coefficients qui vont maintenant représenter notre image compressée : en effet puisqu'on a gardé moins de coefficient, la taille des données est donc plus faible.

Le choix des coefficients à garder peut se faire selon plusieurs critères.

## 2.1 Sélection des coefficients "basse fréquence"

**Exercice 4** Que pensez-vous que l'on obtienne en faisant les opérations suivantes ?

— on calcule la transformée de Fourier $F$, de l'image `tigre`

— on annule le coefficient central $F(\frac{n}{2}, \frac{n}{2})$

— on calcule la transformée de Fourier inverse

Vous testerez et vérifierez vos hypothèses. Faites de même avec le coefficient $F(\frac{n}{2}+1, \frac{n}{2})$, puis avec un coefficient "haute fréquence", i.e plus éloigné du centre ? Conclusions ?

**Exercice 5** Complétez la fonction `void compression(CpxImg FI, int k)`. Cette fonction doit annuler tous les coefficients de `FI` (`FI` sera la transformée de Fourier de l'image $I$ à compresser) qui sont en dehors du carré de coté $2k$ centré sur le centre du tableau (i.e, $n/2, n/2$). On ne conserve donc que les coefficients "basse fréquence" et on "oublie" les coefficients "haute fréquence". L'image est donc représentée par un plus petit nombre de coefficients et pourrait donc être sauvée dans un fichier de taille plus faible (c'est le principe à la base de la plupart des algorithmes modernes de compression d'images, le format JPEG [2] par exemple).

Regardez l'impact sur l'image originale (en appliquant la transformée de Fourier inverse sur le résultat), comparez différentes valeurs de $k$ et concluez.

## 2.2 Sélection des coefficients par seuillage

**Exercice 6** La sélection précédente supposait que les coefficients les plus "significatifs" étaient forcément ceux le plus près du centre (basse fréquence). Cela peut dépendre des images. Une autre méthode consiste à sélectionner les coefficients ayant une valeur suffisamment grande et à annuler les autres. Écrivez une fonction `int compression_seuil(CpxImg FI, double seuil)` qui parcours l'image `FI`, annule les valeurs inférieures à `seuil` (en module), et renvoie le nombre de coefficients "significatifs" (donc non annulés).

**Exercice 7** Vous comparerez les résultats de ces 2 méthodes en comparant, pour un même nombre de coefficients gardés (donc un taux de compression équivalent), les images compressées obtenues.

Testez sur les images `barbara_512.pgm` et `tigre_512.pgm` pour un taux de compression d'environ 10% (c'est-à-dire en ne gardant que 10% des coefficients de Fourier).

2. http ://fr.wikipedia.org/wiki/JPEG