Feuille de TP n°3

Licence 3 MApI3, module **Signal, Fourier, image**

Implémentation de la FFT

Exercice 1 – Algorithme naïf.

1) Écrire une fonction `myDFT(v)` qui renvoie la transformée de Fourier d'un tableau v à une dimension selon l'algorithme itératif donné en cours. On pourra, si on le souhaite, utiliser la fonction `M.dot(v)` qui effectue le produit de la matrice M (un tableau numpy à deux dimensions) par le vecteur v (un tableau à une dimension).

Indications : lors de la création du tableau servant à contenir le résultat, il faudra indiquer à `python` que celui-ci contiendra des nombres complexes. On pourra utiliser la fonction `zeros(taille, complex)` qui renvoie un tableau de complexes de la taille désirée.

La bibliothèque `numpy.fft` introduit la fonction `fft` qui renvoie la transformée de Fourier d'un tableau à une dimension. On pourra l'utiliser pour vérifier le fonctionnement de l'algorithme.

2) À l'aide de l'identité $\check{v} = \frac{1}{N}\left(\widehat{v^*}\right)^*$ (où $*$ désigne le conjugué et $\vee$ la transformée de Fourier inverse), écrire une fonction `myIDFT(v)` qui implémente la transformée de Fourier inverse.

Indication : la fonction `numpy.conj` permet de calculer le conjugué d'un complexe ou d'un tableau en Python.

Exercice 2 – Algorithme rapide.

1) Écrire une fonction récursive `myFFT(v)` qui renvoie la transformée de Fourier de v en utilisant l'algorithme rapide vu en cours. Pour simplifier, on supposera (ainsi que dans les questions suivantes) que la taille de v est une puissance de 2.

2) Sur le modèle de l'exercice précédent, écrire une fonction `myIFFT` qui renvoie la transformée inverse en utilisant l'algorithme rapide.

3) Écrire une fonction `myFFT2(v)` qui renvoie la transformée de Fourier rapide à deux dimensions d'une image v. On pourra mettre à profit le lien entre les transformées 1D et 2D de manière à réutiliser la fonction `myFFT`.

4) Même question pour `myIFFT2(v)`, la transformée inverse.

Utilisation de la FFT

Exercice 3 – Réverbération acoustique par convolution. On fournit trois fichiers audio de travail : `bird.wav`, `drum.wav` et `tictac.wav`. On fournit également trois filtres de réverbération : `garage.wav`, `hall1.wav` et `hall2.wav`.

1) Écrire une fonction `reverb(u, f)` qui renvoie le résultat de la convolution du signal u par le filtre $f + \delta_0$.

Indications : on supposera que u et f sont deux grands tableaux numpy à une dimension, donc qu'il faut passer par une transformée de Fourier pour effectuer le calcul rapidement (on pourra utiliser la fonction `numpy.fft.fft(v)`, qui permet de travailler sur des tableaux de taille quelconque). On pensera à extraire la partie réelle de `ifft` avant d'enregistrer les sons pour éviter des problèmes de typage.

On supposera également que la longueur de f peut être inférieure à celle de u, donc qu'il faut ajouter des zéros à la fin de f pour travailler sur des tableaux de même longueur (on pourra utiliser la fonction `f.resize(taille)` de numpy qui permet d'agrandir un tableau en ajoutant des zéros à la fin).

Enfin, avant d'ajouter δ_0 à f il est conseillé de normaliser f en le divisant par la somme de ses coefficients (obtenue grâce à `f.sum()`) pour éviter que le signal, après convolution, ne sorte de l'intervalle $[-1, 1]$.

2) Tester la fonction `reverb` avec les fichiers audio de travail, écouter et décrire les résultats obtenus. Expliquer pourquoi on ajoute δ_0 à h. On pourra également écouter les fichiers audio de réverbération fournis.

Exercice 4 – Rotation, interpolation par plus proche voisin. Pour commencer, on va assimiler une image v de taille $N \times N$ à une fonction $\tilde{v}$ constante par morceaux sur $\mathbb{R}^2$ telle que l'origine soit au centre de l'image. Ainsi, on a

$$\forall x, y \in \mathbb{R}^2 \colon \tilde{v}(x, y) = v_{\mathrm{round}\left(x+\frac{N}{2}\right),\mathrm{round}\left(y+\frac{N}{2}\right)} \qquad (1)$$

où round(r) désigne l'entier le plus proche de r (avec la convention round$(k+\frac{1}{2}) = k+1$ si $k \in \mathbb{Z}$). La fonction mathématique "round" est implémentée par la fonction numpy du même nom en Python.

Soit $\theta \in \mathbb{R}$. Dans $\mathbb{R}^2$, on note R_θ la rotation (vectorielle) d'angle θ et on pose $\tilde{w} = \tilde{v} \circ R_\theta$.

1) Rappeler quelle est la matrice de R_θ dans la base canonique.

2) Soit w l'échantillonnage de $\tilde{w}$ de taille $N \times N$ avec les mêmes conventions (centre de l'image à l'origine). Pour tout couple (m, n), il existe un couple (m', n') tel que $w_{m,n} = v_{m',n'}$. Exprimer m' et n' en fonction de m et n.

3) Définir une fonction `rotate1(v, theta)` qui renvoie l'image w obtenue après rotation d'angle θ avec les conventions précédentes. La tester à l'aide des images fournies.

4) Pour éviter qu'une partie de l'image se retrouve en-dehors du fenêtrage après rotation, définir une fonction `enlarge(v)` qui, à partir d'une image de taille $N \times N$, renvoie l'image de taille $2N \times 2N$ obtenue en entourant l'image v d'une bande noire de taille $\frac{N}{2}$.

5) Appliquer la fonction `enlarge` à l'image `barbara.png`, puis deux fois la fonction `rotate1` successivement avec des angles de $+30°$ et $-30°$. Comparer l'image obtenue à l'image initiale.

6) Générer pour $N = 40$ l'image $v_{k,l} = cos(8\pi\frac{k+l}{N})$, lui appliquer `enlarge` puis 36 fois `rotate1` avec un angle de $10°$. Comparer le résultat obtenu avec l'image initiale.

7) Décrire qualitativement les résultats obtenus.

Exercice 5 – Rotation sans perte avec Fourier. Soient $a, b \in \mathbb{R}$. Dans ce qui suit, on note R_θ la matrice de la rotation d'angle θ, et

$$M_x = \begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix} \qquad\qquad M_y = \begin{pmatrix} 1 & 0 \\ b & 1 \end{pmatrix}. \tag{2}$$

1) Interpréter géométriquement les applications linéaires D_x et D_y dont M_x et M_y sont les matrices.

2) Calculer le produit $M_x M_y M_x$.

3) En utilisant les formules

$$\cos\theta = 1 - 2\sin^2\frac{\theta}{2} \qquad\qquad \sin\theta = 2\sin\frac{\theta}{2}\cos\frac{\theta}{2}, \tag{3}$$

vérifier qu'en prenant $a = -\tan\frac{\theta}{2}$ et $b = \sin\theta$ on a l'identité $R_\theta = M_x M_y M_x$.

4) Soit $\tilde{v}$ une image analogique . On note $\mathcal{F}_x$ la transformée de Fourier (unidimensionnelle) selon la première variable et $\mathcal{F}_y$ la transformée de Fourier selon la seconde variable. Exprimer $\mathcal{F}_x(\tilde{v} \circ D_x)$ en fonction de $\mathcal{F}_x(\tilde{v})$ et $\mathcal{F}_y(\tilde{v} \circ D_y)$ en fonction de $\mathcal{F}_y(\tilde{v})$.

5) Donner une généralisation approchée des formules précédentes dans le cas d'une image discrète v (on utilisera la même convention que l'exercice précédent, qui place l'origine du repère au centre de l'image).

6) Implémenter les fonctions `Xshear(v, a)` et `Yshear(v, b)` qui effectuent une transformée de Fourier selon une direction, qui multiplient les coefficients obtenus par le déphasage déterminé à la question précédente, puis qui renvoient le résultat de la transformée de Fourier inverse selon la direction considérée. Vérifier le résultat obtenu avec des images de travail pour des valeurs de a ou b comprises entre -1 et 1.

Indications : les fonctions `fft` et `ifft` de Python peuvent appliquer une transformée de Fourier ou son inverse sur les lignes ou les colonnes d'un tableau à deux dimensions. Il faut leur donner respectivement le paramètre supplémentaire `axis=0` ou `axis=1`. On pourra également mettre à profit la fonction `fftfreq(N, 1.0 / N)` qui renvoie un tableau contenant les fréquences correspondantes comprises entre $-\frac{N}{2}$ et $\frac{N}{2}$.

7) Définir une fonction `rotate2(v, theta)` qui applique successivement `Xshear`, `Yshear` puis `Xshear` à l'image v, avec les valeurs de a et de b déterminées précédemment. La tester avec les images fournies pour des angles compris entre $-45°$ et $+45°$.

8) Refaire les questions **5)** à **7)** de l'exercice précédent en utilisant cette fois la fonction `rotate2`.