Feuille de TP n°1

UF Analyse Hilbertienne et Fourier: Convolution et échantillonnage

L'objectif de cette séance de TP est d'illustrer numériquement les notions de convolution et d'échantillonnage vues en cours et voir leurs applications possibles pour l'analyse de signaux permanents. L'outil de base est la transformée de Fourier discrète et l'algorithme de transformée de Fourier rapide qui lui est associée.

1 Transformée de Fourier discrète et FFT

Principe : on étudie un signal sur un intervalle de temps donné $[0,T]$. Ce signal est périodisé de sorte qu'on étudie une fonction $f:[0,T]\to\mathbb{R}$ (ou à valeurs complexes) qui est égale au signal initial *seulement sur l'intervalle* $[0,T]$. On étudie le spectre du signal à l'aide de la série de Fourier associée

$$S(f)(t)=\sum_{n\in\mathbb{Z}} c_n(f)\exp\left(2\,i\,\pi\frac{n}{T}t\right),\quad c_n(f)=\frac{1}{T}\int_0^T f(s)\exp\left(-2\,i\,\pi\frac{n}{T}s\right)ds.$$

On va approcher le coefficient de Fourier $c_n(f)$ à l'aide de la formule des rectangles :

$$c_n(f)\approx\gamma_n^N(f)=\frac{1}{N}\sum_{k=0}^{N-1} f(k\frac{T}{N})\exp\left(-2\,i\,\pi\frac{nk}{N}\right).$$

On appelle cette opération, une transformée de Fourier discrète. Elle possède des propriétés analogues à la transformée de Fourier (notamment son lien avec la convolution). Ici $N\in\mathbb{N}$ est choisi paire. Concrètement, on a échantillonné le signal f à la fréquence $F_e=\frac{N}{T}$ sur un intervalle de temps $[0,T]$ aux points $t_k=k\,F_e$. La transformée de Fourier de ce signal discret est un signal périodique (cf cours Filtrage) : ce qui se retrouve ici puisque la suite $(\gamma_n^N)_{n\in\mathbb{Z}}$ est N-périodique. Dans ces conditions, les $(\gamma_n^N)_{n=0,\dots,N-1}$ fournissent une approximation de $c_n(f)$ si $c_n(f)\approx\gamma_n^N(f)$ si $n\in[0,N/2-1]$ et $c_n(f)\approx\gamma_{n+N}^N(f)$ si $n\in[-N/2,0]$. Pour un signal réel, $c_n(f)=c_{-n}(f)$ de sorte que l'on représente $|c_n(f)|$ en fonction de $\frac{n}{T}$ (les fréquences associées) pour $n\in[0,N/2-1]$.

Théoriquement, le calcul des γ_n^N se fait par un produit matrice vecteur, ce qui induit un cout de calcul de l'ordre de $2\,N^2$. L'algorithme de transformée de Fourier rapide (FFT en anglais) permet de réduire le cout de calcul à $O(N\log(N))$ pourvu que N soit une puissance de 2 à l'aide d'une implémentation récursive.

Pour reconstruire de manière approchée le signal, on utilise la série de Fourier tronquée qu'on évalue aux points t_k

$$f(t_k)\approx S_N(f)(t_k)=\sum_{k=-N/2}^{N/2} c_n(f)\exp\left(2\,i\,\pi\frac{nk}{N}\right),$$

ce qui correspond à la transformée de Fourier inverse mais du point de vue discret. L'algorithme dédiée est appelée IFFT. Les commandes de base en Python sont `FFT`, `IFFT`, `FFTSHIFT` (pour gérer le décalage d'indice) et font appel à la libraire `FFT` de `NUMPY`. Selon le logiciel utilisé, le coefficient $1/N$ devant la somme peut être absent : il faut alors normliser le coefficient calculé pour retrouver une approximation des coefficients de Fourier.

Exercice 1 – Son sinusoidal. Dans ce qui suit, on travaille avec une fréquence d'échantillonnage des données audio de 8800 Hz.

1) Générer un tableau contenant l'échantillonnage d'une fonction sinusoidale à 440 pulsations (périodes) par seconde pendant 3 secondes.

2) On rappelle que la transformée de Fourier discrète d'un signal f à N échantillons permet d'obtenir ses coordonnées dans la base $(\omega^k)_{-\frac{N}{2}<k\leq\frac{N}{2}}$, avec $\omega_n^k=e^{-2ik\pi\frac{n}{N}}$. Avec ces notations, le *mode* associé au coefficient $\gamma_k^N(f)$ est, par définition, l'indice k.

a) La fonction `fft(signal)` du paquetage `numpy.fft` renvoie la transformée de Fourier discrète d'un signal à une dimension. La fonction `abs` de `numpy` renvoie le module d'un tableau de complexes. Représenter le module de la transformée de Fourier du signal en ordonnées en fonction du mode en abscisse.

b) La fonction `fftfreq(N, d)` de la bibliothèque `numpy.fft` renvoie un tableau contenant les fréquences associées à la transformée de Fourier pour un signal contenant N échantillons espacés d'un intervalle d (d est donc l'inverse de la fréquence d'échantillonnage). Représenter les fréquences du signal sinusoidal en ordonnée en fonction du mode en abscisse. Faire de même pour les parties réelles et imaginaires. Que se passe-t-il si on change la valeur de d ?

c) La fonction `fftshift` de `numpy.fft` effectue une permutation des éléments d'un tableau. Après l'avoir testée sur un tableau renvoyé par `fftfreq`, expliquer son intérêt pour la manipulation des coefficients de la transformée de Fourier.

3) Représenter le module de la FFT du signal en ordonnée en fonction des fréquences en abscisse.

4) Même question en remplaçant le signal sinusoidal par un signal carré (on pourra utiliser l'expression $\text{sign}(\sin x)$, qui renvoie un signal carré de période 2π, compris entre -1 et 1).

5) Même question en remplaçant le signal sinusoidal par un signal triangulaire (on pourra utiliser l'expression $\frac{2}{\pi}\arcsin(\sin x)$, qui renvoie un signal triangulaire de période 2π, compris entre -1 et 1).

6) Décrire qualitativement les résultats obtenus.

Exercice 2 – Visualisation de la transformée de Fourier discrète.

1) En utilisant la commande subplot de Python, écrire un programme permettant de visualiser dans 4 fenêtres un signal S, la partie réelle de $fft(S)$, la partie imaginaire de $fft(S)$ et le module de $fft(S)$.

2) Visualiser à l'aide du programme précédent les fft des signaux d'une gaussienne, d'une fonction indicatrice d'un intervalle, d'un Dirac, d'un peigne de Diracs. Faire varier les paramètres de positions, de fréquence, de variance.

3) Tester d'autres signaux de base : piece-regular ou chirps.

Exercice 3 – Phénomène de Gibbs.

1) Construire un signal représentant la fonction de Heaviside échantillonnée à un pas h sur un intervalle de temps $[-T, T]$. Représenter le diagramme énergie-fréquence et le comparer avec le diagramme théorique.

2) Dans les coefficients de Fourier calculés, ne conserver que ceux associés aux fréquences comprises entre $-\lambda_0$ et λ_0 avec $\lambda_0 < \pi/h$. On pourra choisir une fréquence $F_e = 14000Hz$ et une fréquence de coupure $\lambda_0 = 650Hz$.

3) Faire la transformée de Fourier inverse et comparer avec le signal initial. Changer la fréquence de coupure et observer l'effet sur l'amplitude et la fréquence des oscillations aux voisinages des points de discontinuités.

2 Convolution de deux signaux

La convolution de deux signaux se calcule comme suit :

$$>> \quad S = real(ifft(fft(S_1).*fft(S_2)));$$

Exercice 4 – Visualisation du produit de convolution.

1) Ecrire un programme permettant de visualiser simultanément sur deux colonnes et trois lignes, pour deux signaux S_1 et S_2 : les signaux S_1 et S_2, le produit de convolution $S_1 * S_2$, $fft(S_1)$, $fft(S_2)$ et $fft(S_1 * S_2)$.

2) Visualiser les convolutions de deux sinusoides, de deux Gaussiennes, une sinusoide et trois Diracs, une sinusoide et une gaussienne, une gaussienne et un signal régulier par morceaux.

3 Echantillonnage

1) Ecrire un programme qui sous échantillonne un signal en prenant un échantillon sur p et qui permet de visualiser les signaux et les fft d'un signal original et du signal échantillonné.
On pourra faire deux versions, une qui remplace les autres valeurs par 0, l'autre qui extrait un signal de taille N/p.

2) Visualiser les signaux échantillonnés suivants en faisant varier p : une sinusoïde, une gaussienne, un Dirac placé au hasard, un signal Piece-Regular, un Chirps.

3) Ecrire un programme qui effectue la reconstruction à partir du signal sous-échantillonné et qui visualise le résultat.

4) Tester sur les signaux précédents.

5) Ecrire un programme qui supprime l'aliasing avant l'échantillonnage.

6) Tester à nouveau sur les signaux précédents.

7) Comment effectuer un zoom sur un signal ? Comment interpoler raisonnablement la valeur d'un signal entre les points de mesure en utilisant la fft ?