Travaux Pratiques 2 - OASIS SI101 :

Transformation(s) de Fourier et échantillonnage

---

1 Préliminaires :

Dans ce TP nous illustrons les propriétés de la transformation de Fourier. Dans une première partie, nous manipulons la fonction **fft** pour visualiser des transformées de Fourier et bien en comprendre le fonctionnement. Nous verrons en particulier comment l'algorithme Fast Fourier Transform peut être utilisé pour calculer rapidement des choses qui, *a priori*, n'ont pas de rapport avec la transformation de Fourier.

Ensuite, nous construisons une fonction spectrogramme qui permet de visualiser l'évolution dans le temps du contenu fréquentiel d'un signal. Nous construisons aussi un signal dont la fréquence locale croit de manière linéaire en fonction du temps. Nous l'analyserons au travers de la fonction spectrogramme.

Nous abordons ensuite les problèmes que pose l'échantillonnage des signaux. D'abord en écoutant l'effet qu'a un sous-échantillonnage sur un son, puis par la visualisation par spectrogramme de cet effet.

Vous ferez un compte-rendu manuscrit rendu en fin de séance de TP aux encadrants. (pour la partie informatique du lancement, voir les instructions dans la partie TP du site pédagogique)

2 Transformation de Fourier

Sur ordinateur, on ne manipule que des suites finies de nombres. En cours, nous avons vu diverses transformations de Fourier. *A priori*, seule la TFD (contrairement à la TFtD, TFtC et coefficients de Fourier) peut être calculée sur ordinateur.

Comme lors du TP 1, on fera la convention que, si h dénote une suite à support fini, alors $\mathbf{h}$ dénote le vecteur matlab qui la représente et on a $\mathbf{h}(\mathbf{1}) = h_0$ car matlab indexe les éléments d'un tableau à partir de 1.

Si $\mathbf{h}$ est vecteur matlab et M un entier. Alors [1] :

```
fft(h) # renvoie un tableau de même taille que h et qui en est la TFD
fft(h,M) # Si M est plus grand que la taille de h, renvoie la TFD de h complété par
          #des zéros pour atteindre la taille M.
          # Si M est plus petit que la taille du vecteur h alors: renvoie la TFD
          # du vecteur h[0:M] (i.e. les M premiers échantillons de h)
```

On rappelle que la TFD d'une séquence finie $h_0, ..., h_{N-1}, 0, 0...$ où l'on a complété la séquence h_n par des zéros pour obtenir un signal de taille M est exactement l'échantillonnée de la TFtD de la séquence $h_n,\ n \in \mathbb{Z}$ aux fréquences k/M pour k allant de 0 à $M-1$.

Il ne faut pas confondre FFT et TFD. La FFT est un algorithme rapide de calcul de la TFD. Numpy fait cette confusion, c'est dommage. L'algorithme de la FFT est particulièrement rapide pour des vecteurs de taille une puissance de 2. Il s'exécute en un temps proportionnel à $N \log(N)$ où N est la taille du signal.

— Créer deux vecteurs de longueur 100 qui sont deux ondes de Fourier de fréquence 0.1 et 0.123. Et observer leur TFD par la commande **plot(abs(fft(x)))** [2] (**x** est le vecteur). Quelle différence remarquez-vous entre les deux cas ? Pourquoi ? (souvenez-vous de la différence entre une onde de Fourier sur $\mathbb{Z}$ et une onde de Fourier sur $\{0, \ldots, N-1\}$)

1. En fait **fft** est un raccourci pour **numpy.fft.fft**
2. on peut aussi utiliser **stem** à la place de **plot** pour de petits signaux

— Pouvez-vous expliquer la position exacte du pic dans la TFD du signal de fréquence 0.1 (=10/100) ? (en bougeant le curseur on a en bas de la figure l'affichage de x,y [3]). Expliquer aussi la position du second pic qui apparaît dans le cas où l'on trace la TFD de la partie réelle de l'onde de fréquence 0.1 (**plot(abs(fft(real(x))))**). Pour un signal réel, il est inutile de tracer toute la TFD, seule la moitié de celle-ci suffit le reste se déduisant par symétrie hermitienne.

— La commande **plot(x,y)** trace la courbe représentant les **y** en fonction de **x**. Pour avoir un axe des abscisses bien indexé lors de la visualisation d'une TFD il convient de faire

```
t=np.arange(0,100); #vecteur 0,N-1 pour N=100
nu=0.1; #fréquence
onde=exp(2*i*pi*nu*t); #création de l'onde de fréquence nu
plot(t/100,abs(fft(onde))); #tracé correctement indexé de la TFD
```

— Vérifier sur des exemples simples que le produit de convolution est bien transformé en multiplication en Fourier (exemple, la séquence [1 ,2 ,3] convoluée avec [1, 1, 0] dont on calcule la convolution circulaire à la main). Vérifier également que la multiplication par une onde de Fourier conduit à une translation de la TFD.

— Enfin, la fonction **ifft**(**y**) réalise la TFD inverse de y. Que donnent les commandes suivantes ?

```
ifft(fft([1 ,1 ,1]))
ifft(fft([1 ,1 ,1],4))
ifft(fft([1, 2, 3]).*fft([1 ,1 ,0]))-[4, 3 ,5]
```

Comme vu en cours la formule de TFD inverse est très proche de celle de la TFD. L'algorithme qui la calcule est aussi rapide que la FFT.

— **Simuler une convolution sur $\mathbb{Z}$ grâce à une convolution circulaire :** Si $h_0, \ldots, h_{N-1}$ et $g_0, \ldots, g_{N-1}$ sont deux séquences de taille N et si **h** et **g** sont les vecteur (ligne) matlab qui les représentent. Que donne la séquence matlab suivante ? (dans la variable out)

```
N=len(h);# supposé égale à length(g)
hz=np.concatenate((h, ...)) # completer par de zéros
gz=np.concatenate((h, ...))
out=ifft(fft(hz)*fft(gz)) #TFD inverse de produit de TFD
out=out(1:2*N-1)
```

Cette méthode permet en particulier de calculer le produit de deux polynôme de degré $N-1$ en un temps proportionnel à $N\log(N)$ alors qu'il faudrait de l'ordre de N^2 opérations en appliquant simplement la formule qui donne les coefficients du polynôme produit en fonction des coefficients des polynômes d'origine.

3 Fréquence en Hz, fréquence réduite

Si $h_n = f(nT_e)$ est le résultat de l'échantillonnage d'une fonction f à la fréquence $F_e = 1/T_e$ alors une onde de Fourier sur $\mathbb{R}$ de fréquence ξ va donner

$$h_n = e^{2i\pi\nu n T_e} = e^{2i\pi\frac{\nu}{F_e}n}$$

soit une onde de Fourier à la fréquence $\frac{\nu}{F_e}$. Ainsi, la bonne normalisation pour l'axe des abscisses, si on connaît la fréquence d'échantillonnage F_e, est

```
Fe=1000; Te=1/Fe; t=np.arange(0,300)*Te; #fréquence d'échant. 1000Hz. t  qui représente le t
                    # pour 300 échantillons
xi=100; #fréquence en Hertz
h=cos(2*pi*xi*t); # équivalant à cos(2*pi*(nu/Fe)*(0:299))
fh=fft(h)
fh=fh[1:151] # comme h est réel inutile de garder plus de la moitié de la TF
plot(np.arange(0,151)/300*Fe, abs(fh)); # on indexe en vraies fréquences.
```

3. Pour cela il faut configurer spyder pour effectuer les affichages de figures en extérieur Outils->Préférences->Console ipython->Graphique et relancer une nouvelle console ipython

```
# la position 150 dans la TFD donne Fe/2=500Hz, ici.
```

4 Spectrogramme et fréquence instantanée

On veut construire une fonction spectrogramme. Pour une suite u définie sur $\mathbb{Z}$ le spectrogramme est une fonction à deux paramètres : $\nu = k/M$ une fréquence entre $-\frac{1}{2}$ et $\frac{1}{2}$ et une position n

$$\forall (n,k) \in \mathbb{Z} \times \{0,\dots,M-1\},\ U\left(n, \frac{k}{M}\right) = \sum_{m\in\mathbb{Z}} u_m w_{m-n} e^{-2i\pi \frac{k}{M} m}$$

où w est une fenêtre de taille N (seuls $w_0 ... w_{N-1}$ sont non nuls). On ne s'intéressera qu'au module de U. Dans tout ce TP on prendra $w_m = 0.54 - 0.46 cos(2\pi \frac{m}{N-1})$ (pour $m = 0...N-1$)

- Montrer que pour n fixé, on peut calculer tous les $U\left(n, \frac{k}{M}\right)$ grâce à une TFD (d'ordre M) bien choisie (suivie par une multiplication par $e^{-2i\pi \frac{k}{M} n}$ qui ne change rien au module).
- Complétez la fonction **une_colonne_spectrogramme** qui prend comme paramètres : M, N, n et le signal u et renvoie le vecteur $|U(n, \frac{k}{M})|$ pour $k = 0...M/2$ (on suppose désormais que l'on traite un signal réel et que M est toujours pair).
- Complétez la fonction **affiche_spectrogramme** de la manière suivante : Elle prend un paramètre supplémentaire nb et ne prend plus une position particulière n et renvoie un tableau **spectro** à double entrée de taille $(M/2+1) \times (Ent((L-N)/nb)+1)$ qui définit ainsi

  $$\mathbf{spectro}[k,j] = U(nb * j, k/M)$$

  (L =longueur du signal u, et Ent est la fonction partie entière)

  Finaliser l'affichage en choisissant bien les valeurs de début et de fin du temps et des fréquences.

Observer le spectrogramme du signal sonore "aeiou.mat" [4]. Pouvez-vous compter le nombre de voyelles prononcées en regardant le spectrogramme ?

Si on modélise la production d'une voyelle par le processus suivant : Un signal est envoyé des poumons, passe par les cordes vocales, puis va dans la cavité buccale qui agit comme un SLI. À votre avis d'où proviennent les raies que l'on observe dans le spectrogramme d'une voyelle ? Du SLI cavité buccale ? Des cordes vocales ? Ou des poumons ? [5]

Fréquence instantanée On veut construire un signal dont la fréquence instantanée varie lentement et observer un tel signal au travers de son spectrogramme. Supposons qu'un signal soit défini par $f(t) = cos(2\pi r(t))$. Si $r(t)$ est une fonction linéaire $r(t) = \nu t$ alors on voit que ce signal est une onde (réelle) de fréquence ν. Plus, généralement, supposons que $r(t)$ soit une fonction dérivable, et soit t_0 un instant autour duquel on veut analyser $f(t)$. On écrit

$$f(t) \approx \cos(2\pi(r(t_0) + r'(t_0)(t-t_0))) = cos(2\pi r'(t_0)t + \phi)$$

Le signal $f(t)$ se comporte comme une onde de fréquence $r'(t_0)$ autour du point t_0.

On se donne une fréquence d'échantillonnage F_e construire un signal (**s**) d'une durée de 2 secondes dont la fréquence instantanée varie de 0 à $F_e/2$ entre le début et la fin. On prendra, pour tester, $F_e = 8000$.

écouter le signal construit par **play(s,Fe)**. Observer son spectrogramme.

Observer le spectrogramme de **s[: :2]**. Que constatez vous ? Comment faut-il normaliser les axes de visualisation pour ce spectrogramme ?

Rappeler la valeur de la TFtD de **s[: :2]** en fonction de celle de **s** (voir TD2), et expliquer ainsi l'observation faite sur le spectrogramme.

Écouter **play(s[: :2],Fe/2) ;** (pourquoi $F_e/2$?). Est-ce en accord avec votre observation ?

4. La lecture du fichier depuis le disque dur est déjà écrite. La fréquence d'echantillonnage nécessaire à l'écoute et à l'analyse est $Fe = 22050$Hz.

5. Si vous ne voyez pas les raies, il faut songer à augmenter les paramètres N et M pour augmenter la résolution fréquentielle. L'ordre de grandeur de l'écart entre les raies est de 100Hz. Pensez à utiliser la fonction de zoom du visualiseur de python (voir note 3).