CCIR-IdF ——— ESIEE	Traitement du signal	ISBS

Remis par M. J.-F. BERCHER

ÉNONCÉ

Ce TD-TP a pour but d'illustrer certaines notions sur la représentation de Fourier, les problèmes de résolution, masquage, puis la problématique de l'échantillonnage et la TFD. Vous récupérerez les fichiers adéquats sur la page Moodle de l'unité, ou sur la page web de l'enseignant.

On utilisera donc à nouveau Python, avec les bibliothèques scientifiques scipy et numpy, ainsi que les modules scipy.signal. Quelques indications sur les commandes et les fonctions utiles sont fournis dans l'énoncé. Pour le reste, il vous faudra utiliser l'enseignant et la documentation en ligne...

Scripts et images fournis :

Votre serviteur, dans le but noble de vous faciliter tâche sans sacrifier la compréhension, a préparé quelques fonctions utiles, notamment :

— `rien`

On rappelle que si on veut correctement représenter les signaux en temps et en fréquence, il faut définir les axes temporel et fréquentiel, et utiliser un décalage en fréquence pour centrer le zéro de la TF : *par exemple...*

```
Fe, Te = 32, 1/32
N=len(x)
M=8*N #Utilisé comme longueur des fft
# ou alors M=1000, comme on veut qui soit pertinent
# On définit les vecteurs t et f
t=arange(N)*Te
f=(arange(M)/M-1/2)*Fe
plot(t,x)
title('Signal temporel')
figure()
plot(f,abs(fftshift(fft(x,M))))
title('Signal en fréquence')
xlabel('Fréquences')
```

I. RÉSOLUTION DE FOURIER

Pour le moment on n'examinera pas les spécificités de la Transformée de Fourier discrète. On ne s'intéressera qu'au problème de la limitation en temps. Les transformées de Fourier seront calculées sur un grand nombre de points (par exemple 1000), selon `fft(x,1000)`, afin de ne pas être embêtés par la discrétisation de l'axe fréquentiel.

A. Analyse sur une durée limitée - résolution

1. Créez une sinusoïde sur 1000 points, avec une fréquence normalisée $f_0 = 0.05$. Remarquons que sur N=1000 points, on a exactement $Nf_0 = 50$ périodes, et qu'ici ce nombre est entier. Tracez le signal temporel puis sa transformée de Fourier
2. Supposons maintenant que la sinusoïde soit de durée limitée, par exemple sur $L = 100$ ou $L = 50$ points. Créez une sinusoïde tronquée `x_tronq` sur L points (créez un vecteur de zéros puis remplissez les L premiers points par la sinusoïde). Examinez ensuite ce que vaut alors sa TF, et comparez ce résultat à la TF initiale (pour la visualisation, prendre en compte un facteur d'amplitude L/N reflétant le fait qu'il y a moins d'énergie dans le signal tronqué)
3. Il devrait être apparent que les impulsions de départ ont été "élargies" par la limitation temporelle. En fait, le signal de départ peut-être vu comme ayant été multiplié par une fenêtre rectangulaire pour donner le signal tronqué. Il s'en suit une convolution dans le domaine fréquentiel. Vérifiez ceci en calculant la

TF d'une fenêtre rectangulaire et en convoluant celle-ci (fonction `convolve` de numpy) avec la TF de la sinuoïde initiale.

— Convoluez les TF *centrées*, ie après un `fftshift`, sinon il peut y avoir des effets de bords désagréables

— Il y a un facteur N après la convolution - en tenir compte dans la comparaison

4. Prendre maintenant la somme de deux sinusoïde, de fréquences respectives $f_0 = 0.05$ et $f_1 = 0.1$; vous prendrez $\phi = 0$.

$$x(t) = \sin(2\pi f_O t) + \sin(2\pi f_1 t + \phi) \tag{1}$$

Ces deux sinusoïdes sont prises sur L points, sur un horizon total de $N = 1000$. Examiner les TF dans le cas $L = 100$ puis $L = 50$. Reprendre ensuite ceci avec deux sinusoïdes de fréquences plus proches, par exemple avec $f_0 = 0.05$ et 0.06.

5. Du fait du recouvrement entre les sinus cardinaux, on note un *déplacement des maxima* et pour L=50, la perte des deux pics. On peut également noter, avec cet écart de fréquences, que le résultat est hautement sensible à la phase des signaux.Expérimentez en modifiant la phase de la seconde sinusoïde, par exemple en premant $\phi = \pi/4$, puis $\phi = 2\pi/3$.

II. Masquage et fenêtrage

Un autre problème est celui du masquage des signaux faibles. Pour le constater, il suffit par exemple de considérer la somme de deux sinusoïdes de fréquences respectives $f_0 = 0.05$ et $f_1 = 0.2$ et d'amplitudes $A_0 = 1$ et $A_1 = 0.01$.

1. Simulez ce signal et considérez sa transformée de Fourier. Vous aurez intérêt à tracer la représentation en décibels : prendre $20\log(|X(f)|)$
2. Plutôt que la pondération rectangulaire, prendre une autre fenêtre de pondération, comme la fenêtre de Hamming, et définissez le signal tronqué comme

```
x_tronq[arange(L)}=x[arange(L)]*sp.hanning(L)
```

Pour celà, importez le module signal de scipy, par `import scipy.signal as sp` Consultez l'aide sur `sp.hanning`, puis comparez là à la fenêtre rectangulaire `sp.boxcar`.

3. Tracez les représentations pour les différentes fenêtres, d'abord en temps, puis en fréquence en échelle log. Conclure.

III. Interpolation de Shannon

Il s'agit de vérifier la formule d'interpolation de Shannon pour un signal correctement échantillonné :

$$x(t) = \sum_{n=-\infty}^{+\infty} x(nT_e)\frac{\sin(\pi F_e(t - nT_e))}{\pi F_e(t - nT_e)} \tag{2}$$

Pour ce faire, vous créerez une sinusoïde de fréquence f_0, que vous échantillonnerez à 4 échantillons par période ($F_e = 4f_0$), vous implanterez la formule d'interpolation et comparerez l'approximation (nombre fini de termes dans la somme) au signal initial. Le module `numpy` fournit une fonction `sinc`, mais vous prendrez garde au fait que la définition de celle-ci inclut le π : $sinc(x) = \sin(\pi x)/(pix)$

IV. Dualité échantillonnage-périodisation – vers la TFD

Vous disposez d'un signal $x(n)$, échantillonné à $Fe = 32$.

1. Chargez ce signal par

```
f=numpy.load('signal.npz')
#f.keys()
x=f['x'].flatten()
```

Le signal est alors chargé dans l'environnement sous le nom x. Visualisez x dans le domaine temporel et fréquentiel. Quelle est sa durée temporelle ? Quelle est approximativement la bande occupée ?

2. On étudie d'abord l'effet d'une répétition du signal. Créez un nouveau signal, *xr(n)* et répétant 8 fois le motif *x(n)* (fonction `repeat`). Visualisez le signal temporel, puis comparez les réponses en fréquence de *x(n)* et *xr (n)*. Conclusions.
3. On s'intéressera ensuite aux effets de l'échantillonnage : rééchantillonnez le signal aux fréquences Fse = 16, Fse = 8, Fse = 4 (créez les signaux xe1(n), xe2(n) et xe3(n)), en utilisant la fonction `echant`. Visualisez les signaux temporels, et comparez les réponses fréquentielles (toujours sur [-Fe/2, Fe/2], avec Fe = 32, la fréquence d'échantillonnage initiale).
4. Créez enfin un signal périodique échantillonné xre (n), en périodisant le signal initial (en créant par exemple 8 périodes) puis en échantillonnant le signal résultant. Analysez le signal obtenu en temps et en fréquence. Conclusions.

V. La TFD : échantillonnage de la TF

La TFD peut être comprise simplement comme résultant de l'échantillonnage de la TF à fréquence continue d'une séquence discrète, sur une grille régulière. Il est possible d'illustrer très simplement ceci par une petite expérimentation :

1. Créez une sinusoïde de fréquence $f_0 = 0.07$, sur $N = 50$ points, et
2. Calculez et tracez (fonction `plot`) sa transformée de Fourier `Sz` calculée sur 1000 points `Sz=fft(s,1000)`, et ce en fonction d'un vecteur de fréquences `f` correctement défini sur $[0, 1$
3. Sur le même graphique, représentez (fonction `stem`) la TF calculée uniquement sur N points `S=fft(s)`, et ce en fonction d'un vecteur de fréquences `f2` correctement défini sur $[0, 1$
4. Enfin, ajoutez sur le graphique cette même TF où les points sont reliés par des segments de doite (tout simplement la fonction `plot`)

VI. Spectrogramme

Pour des signaux dont le contenu fréquentiel varie au cours du temps, plutôt que faire une analyse globale, qui finalement revient à moyenner les variations de fréquences, on eut utiliser une *transformée de Fourier à court terme*. Celle-ci consiste à calculer le TF d'une une portion localisée du signal, après pondération par une fenêtre (afin d'éviter les lobes secondaires de la fenere rectangulaire –cf plus haut). On obtient alors une transformée de Fourier, qui dépend non seulement de la fréquence, mais aussi de la localisation temporelle

$$X(f,\tau) = \mathrm{TF}\{x(n).w(n-\tau)\} = \sum_n x(n).w(n-\tau)e^{-j2\pi fn} \tag{3}$$

où $w(n-\tau)$ désigne la fenêtre $w(n)$ décalée de τ. Par suite, il est possible de construire une représentation conjointe, *en temps et en fréquence*, en collectionnant l'ensemble des transformées à court terme. Cette représentation conjointe, sous la forme d'une image du contenu fréquentiel en fonction du temps, s'appelle un *Spectrogramme*.Vous allez chercher à contruire un spectrogramme élémentaire, puis annalyser quelques signaux. Pour ce faire, vous créerez une fonction `spectro`, qui parcourera tous les échantillons `n` du signal d'entrée et, pour tout n, devra :

1. Extraire une portion du signal de départ `x` et le pondérer par une fenêtre : `xw=x[n:n+L]*window`
2. Calculer la TF correspondante et ranger le résultat dans un tableau

   ```
   Xftmp=fft(xw,N)
   Xwf{}0:N/2+1,k{}=Xftmp{}0:N/2+1{}
   ```

   Une fois que la boucle sur le temps aura été effectuée, et que vous aurez correctement initialisé les différentes variables (notamment celle-ci : `Xwf=zeros((N/2+1,Nx))+0j`) :
3. Il ne restera plus alors qu'à afficher la matrice résultante sous la forme d'une image imshow(abs(Xwf), aspect='auto', origin='lower',extent=(0, Nx, 0, 0.5))

Vous pourrez tester ce spectrogramme sur différents signaux : un cri (bat.wav) de chauve-souris, un son (gong.wav) de gong ou un chant (bruant.wav) de bruant.
Pour ce faire, vous importerez le module qui va bien

```
from scipy.io.wavfile import read as wavread, write as wavwrite
```

et lirez le fichier par :

```
(r, x) =wavread('bat.wav')
```

Vous représenterez à la fois le spectrogramme et le signal en fonction du temps.