ENSG	Unité : Traitement du signal TP n°1 - Filtrage élémentaire	Master PPMD

Sujet de J.-F. BERCHER – Professeur ESIEE

ÉNONCÉ

TP SOUS PYTHON

Le TP sera réalisé sous Python. Sous Python, on dispose d'un certain nombre de bibliothèques scientifiques qui nous seront utiles, notamment *numpy*, *scipy* et *matplotlib*. Le module *pylab* permet de charger des commandes simplifiées analogues à celles que l'on trouve dans Matlab.

Spyder

Pour ces TP, vous utiliserez l'éditeur Spyder, qui précharge les modules scientifiques et le mode Pylab. Dans *Outils>Préférences>Console IPython>Graphiques*, vous vérifierez que le mode graphique (Matplotlib) est activé, que le chargement automatique de Numpy et Pylab est coché, et enfin que la sortie graphique est positionnée sur « automatique ». Dans *Démarrage*, vous cocherez « ouvrir une console IPython au démarrage ».
Pour ceux qui voudraient installer un environnement équivalent sur un ordinateur personnel, il vous suffit

- sous Windows, de télécharger et installer une distribution comme Pythonxy ou WinPython
- sous Linux debian (ou ubuntu) : `sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy spyder python-nose python-pygments python-zmq python-setuptools python-sphinx`
- la distribution Python Anaconda est paraît-il très bien (sous Linux, mac, Windows)

Dans Spyder, vous pouvez entrer vos commandes soit directement dans la console, soit créer un script (enregistrez le nom sous un nom différent de temp) et l'exécuter par la touche de raccourci F5, ou n'exécuter que les lignes sélectionnées par F9.

Aide

Vous disposez d'une aide en ligne via l'inspecteur d'objet, qui est également automatiquement activé lorsque vous entrez des commandes dans l'éditeur ou la console ; si vous entrez `plot(` par exemple, l'aide sur la fonction sera affichée dès que vous aurez entré la parenthèse ouvrante. Vous disposez également d'une aide en ligne par `help(nom_de_la fonction)`.

Imports

Vous aurez besoin à la fois d'une fonction de filtrage, `lfilter`, et des fonctions de transformée de Fourier (`fft` et `ifft`). Les imports se font par :

```
# On a besoin de la fonction de filtrage
import scipy from scipy.signal import lfilter
# On aura besoin des fonctions fft
from numpy.fft import fft, ifft
# Pour ne pas s'embêter avec les divisions
from __future__ import division
from __future__ import print_function
```

En outre, on fournit un fichier `defs_tp_filt.py`, qui contient quelques définitions, que vous téléchargerez et exécuterez une fois.

Tracer des graphiques

La commande `figure()` ouvre une nouvelle figure graphique. Si nécessaire `clf()` efface la figure courante. la commande `plot(x,y,label='texte', color='nom_de_couleur')` représente y en fonction de x. La commande `legend()` ajoute une légende construite avec les différents labels ; les paramètres labels, color, etc sont tous optionnels. La commande `stem`, avec les mêmes paramètres que plot, permet d'afficher une courbe en batons (points non reliés). Vous pourrez par exemple tester les lignes suivantes :

```
# Représentation
N=100
stem(range(N),dirac(N))
title("Dirac $\delta(n)$")
xlabel("n")
ylim([0, 1.2])    # Pour mieux voir l'impulsion xlim([-5, 10])
```

Pour les représentations en fréquence, prendre garde au fait que
(i) les TF sont complexes (prendre le module – `abs` le cas échéant), et
(ii) le résultat de la TF est rendu sur une période allant de 0 à 1 ;
il est possible de recentrer le résultat, et voir les fréquences négatives, en utilisant un décalage, fonction `fftshift`.

Le but de ce TP est de concevoir et appliquer différents filtres numériques sur un signal périodique de fréquence fondamentale f_0=200 Hz, échantillonné à la fréquence F_e=8000 Hz.
Ce signal est stocké dans le vecteur x sauvegardé dans le fichier [sig1.npz][1]. On peut le récupérer sous Python par l'instruction : ‘`f=numpy.load('sig1.npz')`‘
[1] : http://www.esiee.fr/~bercherj/ppmd/sig1.npz

I – ANALYSE DES DES DONNÉES :

On peut afficher le signal, en temps et en fréquence, par les instructions suivantes :

```
# Pour charger le truc :
f=numpy.load('sig1.npz')
m=f['m']
x=f['x']

Fe=8000
Te=1/Fe

# Affichage  temps
figure(1)
t=arange(len(x))*Te
plot(t,x,t,m)
title('Signal avec dérive lente')
show()
```

et

```
# Affichage fréquence
figure(2)
N=len(x)
f=freq(N)
plot(f,abs(fftshift(fft(x))))
xlim([-0.5, 0.5])
title('TF du signal (module)')
```

Commentez et interprétez.
On souhaite à présent modifier le contenu spectral du signal x par différents filtres numériques de fonction de transfert $H(z) = B(z)/A(z)$. Une fonction standard Python vous sera très utile :

— *lfilter* qui implémente l'équation aux différences~ : cette fonction calcule le vecteur y des sorties d'un filtre numérique spécifié par le vecteur B des coefficients du numérateur $B(z)$, le vecteur A des coefficients du dénominateur $A(z)$, pour un vecteur d'entrées x, suivant l'instruction~ : `y=filter(B,A,x)`

II – CALCUL DES FILTRES MOYENNEUR ET SOUSTRACTEUR

Le signal est affecté par une dérive lente de son niveau moyen. On cherche alors à extraire cette valeur moyenne à variations lentes, d'une part, et à calculer le signal débarrassé de cette dérive d'autre part. On notera $M(n)$ la dérive, et $x_c(n)$ le signal centré.

II.1 ***Partie théorique :***

1) Quelle expression permet de calculer la moyenne du signal x sur une période ?
2) En déduire un filtre, de réponse impulsionnelle $g(n)$, qui calcule cette moyenne $M(n)$.
3) En déduire un autre filtre, de réponse impulsionnelle $h(n)$, qui élimine cette moyenne : $x_c(n) = x(n) - M(n) = x(n) * h(n)$. Donner l'expression de $h(n)$.

Donner les expressions de $G(z)$ et de $H(z)$.

II.2 ***Partie pratique***

Pour le filtre moyenneur, puis pour le filtre soustracteur :

— Créer et tracer les deux réponses impulsionnelles (on pourra se servir de l'instruction `ones(L)` qui crée un vecteur ligne de L uns.
— Tracer les réponses en fréquence de ces filtres. Vous pourrez utiliser la fonction `fft` qui calcule la TF, et tracer le module (`abs`) du résultat.
— Filtrer le signal x par ces filtres. Tracer les signaux de sortie ainsi que leurs spectres en fréquence. Conclure.

III – ACCENTUATION D'UNE BANDE DE FRÉQUENCE

On souhaite maintenant accentuer (très nettement) la zone de fréquence autour de 1000 Hz sur le signal initial.

III.1 Partie théorique

Après un rappel éventuel de l'enseignant sur les filtres rationnels, déterminez les pôles p_1 et p_2 d'un filtre permettant de réaliser cette accentuation. Calculer la fonction de transfert, $H(z)$, correspondante ainsi que la réponse impulsionnelle $h(n)$.

III.2 Partie pratique

— Le vecteur de coefficients du dénominateur $A(z)$ sera calculé par : ***A=poly([p1,p2])***
et vous vérifierez que vous obtenez les coefficients déterminés "à la main".
— Tracer la réponse fréquentielle du filtre
— Calculer sa réponse impulsionnelle par :

```
# calcul de la RI
d=zeros(300)
d[1]=1
h_accentue=lfilter([1],a,d)
```

(réponse à une impulsion de dirac calculée sur 300 points). La tracer.
— Calculer et tracer la réponse impulsionnelle obtenue avec la formule théorique. La comparer à la précédente.
— Calculer et tracer la sortie du filtre soumis à l'entrée x_c ainsi que son spectre. Conclure.
— Comment peut-on simultanément amplifier autour de 1000 Hz et supprimer la dérive ? Proposer un filtre qui réalise les deux opérations.

IV – FILTRAGE PASSE-BAS [0- 250 HZ] PAR LA MÉTHODE DE LA FENÊTRE

On cherche maintenant à ne conserver que les basses fréquences (0 à 250 Hz) du signal x_c en filtrant celui-ci par un filtre passe-bas FIR à N=101 coefficients.

IV.1 Partie théorique

On considère un filtre passe-bas idéal dont le module de la fonction de transfert, $H(f)$, est une fonction rectangulaire. Calculer la réponse impulsionnelle (infinie) du filtre numérique qui réaliserait ce passe-bas de façon idéale.

IV.2 Partie pratique

a. On veut limiter le nombre de coefficients à L (RIF). Calculer le vecteur h à L coefficients représentant cette réponse pondérée (tronquée) par une fenêtre rectangulaire rect$_T$(t) où $T = L * Te$.

b. Tracer la réponse fréquentielle de ce filtre.

c. Calculer et tracer la sortie de ce filtre soumis à l'entrée x_c ainsi que son spectre.

d. Observer le temps de propagation de groupe de la réponse fréquentielle : `plot(f,grpdelay(B,A,N))`. Commenter.

V – CONSTRUCTION DE LA TFD. DUALITÉ ECHANTILLONNAGE-PÉRIODISATION

V.1 Préalable

1) Créer une fonction *repeat(x,n)* permettant de périodiser un signal. Vous pourrez utiliser pour ce faire le fait que sous Python, [(x)*n]=((x) (x) ... (x)) , où x est une liste quelconque.
2) Tester cette fonction, par exemple en prenant *x=randn(10)*, ou *x=hann(50)*,
3) Créer une fonction *echant(x,step)* permettant de sous échantilloner un signal, en ne retenant qu'un point tous les -step_.
4) Tester cette fonction, par exemple en prenant *x=range(100)*, *x=randn(10)*, ou *x=hann(50)*

V.2 Manipulations

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.