RICM4		Traitement du signal
TP Image et Musique, Fréquences, FFT, et filtrage		
2017		

I. Analyse sur une image

Ouvrez le fichier TP_imageFFT.py, lancez-le. (pas de programmation dans cet exercice)

1) Sur la première image (le sinus2D simple), donnez la longueur en pixel des périodes suivant l'axe des lignes et suivant l'axe des colonnes. Expliquez alors la fft de cette image et donnez une explication à la position des deux pics. Après le fftshift, la même fft est alors affichée avec le zéro fréquentiel placé au milieu de la fenêtre.
2) On compose une image avec 3 sinus différents. Repérez dans le programme les différentes périodes et les différentes amplitudes de ces trois sinus. Retrouvez-les sur la fft (justifiez). Que pouvez-vous dire quand N=N0 sur les périodes de votre image? Retrouvez-vous vos 3 fréquences ? Le fenêtrage est-il utile ? Testez maintenant N=625, une valeur arbitraire. Mêmes questions que précédemment, en particulier le fenêtrage est-il utile, si oui pourquoi ?

II. Analyse sur un signal de son

Dans de nombreux systèmes physiques (optique, acoustique, radio-astronomie, communication …), la simple représentation temporelle des signaux se révèle insuffisante. Aussi utilise-t-on souvent une représentation fréquentielle. L'analyse de spectre fournit cette représentation sous forme d'une répartition de la puissance du signal en fonction de la fréquence.

Dans le cadre de ce TP, vous allez travailler avec différents fichiers sons correspondant à des notes de musique. L'objectif de ce TP est de repérer correctement les fréquences présentes dans ce fichier, d'étudier les problèmes de résolutions puis de filtrer une fréquence particulière. Une analyse des résultats obtenus et la justification des choix effectués seront les bienvenus.

Pour ouvrir un fichier son avec python, vous pouvez utiliser la fonction wavread de scipy.io
```
from scipy.io import wavfile
[Fs,y] = wavfile('note.wav')
```
Y est le signal échantillonné, Fs, la fréquence d'échantillonnage.
Ouvrir et Lire le fichier python d'introduction fourni TP_son_etu.py

Analyses de notes de musiques :
Une note pure est une vibration sonore de type sinusoïdal à une fréquence. On vous donne le tableau des fréquences dans l'énoncé. Le La de référence est ici de 440 Hz, mais souvent les instruments sont encore accordés à 437Hz, et parfois l'accord n'est pas parfait…
Un instrument de musique classique produit pour chaque note un fondamental f_0 et des harmoniques $f_n=2*f_{n-1}=2^n*f_0$ pour n>0.
Souvent la qualité d'un instrument repose sur sa capacité à produire de bons harmoniques. La différence entre une note grave et aigüe vient simplement du fait que l'harmonique la plus puissante est celle de l'octave considéré.

Tableau des fréquences de notes musicales :

$f_{réf}$ (Hz)	440

Ton	Note	Octave 1	2	3	4	5	6	7	8	9
1	do	32,703196	65,406391	130,81278	261,62557	523,25113	1046,5023	2093,0045	4186,009	8372,0181
2	do#	34,647829	69,295658	138,59132	277,18263	554,36526	1108,7305	2217,461	4434,9221	8869,8442
3	ré	36,708096	73,416192	146,83238	293,66477	587,32954	1174,6591	2349,3181	4698,6363	9397,2726
4	ré#	38,890873	77,781746	155,56349	311,12698	622,25397	1244,5079	2489,0159	4978,0317	9956,0635
5	mi	41,203445	82,406889	164,81378	329,62756	659,25511	1318,5102	2637,0205	5274,0409	10548,082
6	fa	43,653529	87,307058	174,61412	349,22823	698,45646	1396,9129	2793,8259	5587,6517	11175,303
7	fa#	46,249303	92,498606	184,99721	369,99442	739,98885	1479,9777	2959,9554	5919,9108	11839,822
8	sol	48,999429	97,998859	195,99772	391,99544	783,99087	1567,9817	3135,9635	6271,927	12543,854
9	sol#	51,913087	103,82617	207,65235	415,3047	830,6094	1661,2188	3322,4376	6644,8752	13289,75
10	la	55	110	220	440	880	1760	3520	7040	14080
11	sib	58,27047	116,54094	233,08188	466,16376	932,32752	1864,655	3729,3101	7458,6202	14917,24
12	si	61,735413	123,47083	246,94165	493,8833	987,7666	1975,5332	3951,0664	7902,1328	15804,266

La à 437 octave 4

Re#	Mi	Fa	Fa#	Sol	Sol#	La	Sib	Si	do	Do#	Re	Re#
309	327.38	346.85	367.47	389.32	412.47	437	462.99	490.52	519.68	550.59	583.325	618.01

A titre purement d'information et pour votre curiosité que je sais grande ;-), le calcul des fréquences en fonction du La de départ et de l'octave choisi est donné dans le fichier TP_son_etu.py

1 Une note

Retrouver, à l'aide du tableau ci-dessus, la note jouée. Fichier 'une_note.wav'.

Ecrire un programme qui affiche le signal en temporel, le spectre d'amplitude du signal ainsi que le spectre en dB. On fera très attention à la bonne définition des axes. Le temps est en seconde et l'axe des fréquences en Hz. Voir fichier TP_son_etu.py qui vous donne le début du programme.

Pourquoi le spectre semble-t-il être nul à 16kHz alors que la fréquence d'échantillonnage est de 44,1 kHz ? Que pouvez-vous en déduire ?

Etudiez en fonction du nombre de points (Nbr) la valeur de fréquence principale f_0 trouvée. En particulier, étudiez $Nbr=x*1/f_0$ où x vaut par exemple 1,4, 64, 128. Etudiez ensuite un nombre de point quelconque. Expliquez vos résultats. Notez pour chaque valeur de Nbr la résolution fréquentielle que vous obtenez.

2 Deux notes

- Etablir par le calcul, le nombre de points nécessaires pour séparer des fréquences distantes de 30, 40, 50, 60 Hz.
- Choisissez et justifiez votre nombre de points minimum pour séparer deux fréquences de musique proches.
- A partir du fichier ' Deux_notes_attenuees.wav ', retrouver, à l'aide du tableau, les deux notes jouées. Pour ceci utilisez la fenêtre rectangulaire et la fenêtre de Hamming.
- Expliquer en choisissant N=4096, puis N=8192 vos résultats pour les deux fenêtres.
- Exemple de code :

```
##fenetre de hamming"
fen1 = np.hamming(N);
```

```
fen = np.reshape(fen1,N)
s1fen = s1*fen;
```

3 Deux notes dont une à restituer : Filtrage

Utiliser le fichier ‘deux_notes.wav’, où les deux notes sont d’amplitude similaire.
Quelles sont les deux notes présentes ? Proposez un filtrage dans le domaine fréquentiel permettant d’isoler le fondamentale d’une note. Tracez le signal filtré, son spectre. Ecoutez la restitution, commentaire ?

Réalisation du filtre dans le domaine fréquentiel :
Il faut définir un filtre dont les fréquences de coupure seront choisies en fonction de ce que vous voulez extraire. Sous scipy, on peut utiliser différents types de filtre, l’un des plus classiques utilise la version numérique du filtre analogique de Tchebytchev.
Pour réaliser ceci avec numpy, on utilise che2ord, cheby2 pour créer le filtre numérique dans le domaine de la TZ à partir des spécifications analogiques. Par exemple un filtre passe-bande laissant passer les pulsations entre 0.2*fe/2 et 0.5*fe/2 et coupant à -40dB : à 0.1*fe/2 et à 0.6*fe/2 est programmé ainsi :

```
N, Wn = sc.cheb2ord([0.2, 0.5],[0.1, 0.6], 3, 40)
z,p,k = sc.cheby2(N, 40.0, Wn, 'bandpass',output=’zpk’)
```

Ayant ainsi les pôles, les zéros et le gain de la TZ, écrire la fonction filtre qui vous donnera la réponse fréquentielle (voir template dans le fichier fourni)

```
H=filtre(z,p,k,Nbr)   # fonction à écrire
```

- Tracer le module de la réponse fréquentielle du filter H.
- Appliquer ensuite votre filtre à votre signal (dans le domaine de Fourier), tracer le signal résultat et son spectre
- Commentaire sur la note restituée ?