Traitement du Signal
- Travaux Pratiques -

Thème : "Echantillonnage, Transformée de Fourier d'un signal échantillonné"

Modalités de déroulement

Travail en binôme
- Sauf autorisation

Evaluation
- Rapport de synthèse (à remettre 1 semaine après la dernière séance)
- Compréhension des exercices (évaluation continue)
- Travail effectué

Logicie/langage utilisé
- le logiciel/langage de programmation utilisé est Matlab
- *Remarque :* il existe un logiciel libre développé sur le modèle de Matlab par l'INRIA (laboratoire de recherche français) : Scilab. Sa syntaxe est pratiquement identique à celle de Matlab. Il diffère de Matlab du point de vue de ses fonctions spécialisées (et notamment des fonctions de Traitement du Signal).
- Une petite aide à l'utilisation de Matlab est donnée en annexe

1) Prise en main de Matlab (durée recommandée : 30mn)

En dernière page de ce document est donnée une liste de commandes destinée à la découverte de Matlab par la pratique directe, de manière progressive. Si vous n'êtes pas encore familiarisé avec Matlab, il est conseillé de toutes les tester (utiliser pour cela la fenêtre de commande de Matlab) et d'analyser leurs effets, pour aborder plus facilement les programmes de la suite.

Pour des explications sur les commandes Matlab, il existe la commande `help` suivi du nom de la commande. Il est conseillé de commenter chacune de ces commandes (symbole de début de commentaire : `%`); cela peut s'avérer utile pour la suite.

Pour cette partie, il n'y a pas de travail de synthèse à effectuer.

2) Génération et affichage de signaux

A partir de cet exercice, il est conseillé d'utiliser le mode "fichier" de Matlab (menu Fichier, option Nouveau, etc).

On souhaite générer afficher un signal sinusoïdal d'équation

$$s(t) = A.\sin(\omega_0 t) = A.\sin(2\pi f_0 t)$$

où A est son amplitude et f_0 sa fréquence (ω_0 sa pulsation).

Les différentes étapes ci-dessous permettent de réaliser cette opération.

2.1) Pour connaître les instants des échantillons de ce signal, il faut générer un vecteur dont les éléments représentent des instants régulièrement espacés (on l'appellera "vecteur de temps"), par exemple de la manière suivante :

```
...
t=(0:N-1)*Te;       %N : nombre d'éch. total ; Te : période d'échantillonnage
...
```

Le signal peut alors être généré à l'aide de la commande suivante :

```
...
s=a*sin(2*pi*f0*t);
...
```

On choisit les paramètres suivants :

- fréquence d'échantillonnage : f_e=100Hz, 1000Hz et 10000Hz,
- fréquence du signal : f_0=10Hz,
- amplitude du signal : a=1.

Ecrire un programme, utilisant les 2 lignes de code ci-dessus, permettant de générer (et d'afficher) une période de signal.

2.2) On souhaite avoir toujours un nombre entier de périodes dans les N échantillons (on verra pourquoi dans l'analyse de la TFD). Modifier ce programme pour avoir un paramètre `np` (nombre de périodes). Le tester pour `np`=1.

2.3) Modifier le programme pour générer 3 périodes de signal, les afficher en rouge et donner un titre au graphique.

2.4) Proposer une solution pour que cet axe soit gradué avec le temps correspondant au signal (préciser comment est gradué l'axe des abscisses sinon).

3) Etude de l'échantillonnage

Remarque : cet exercice reprend une partie de l'exercice 2 du TD3 (partie "Echantillonnage").

Introduction

Avec Matlab utilisé seul, on ne peut pas réaliser d'échantillonnage proprement dit (sauf si l'on dispose d'un matériel d'acquisition de signaux géré par Matlab).

On ne contrôle pas les paramètres d'échantillonnage, puisqu'on travail directement sur des signaux numériques (par exemple, un fichier son, un signal biomédical dont les échantillons sont stockés dans un fichier, etc). On ne peut donc pas étudier directement les conséquences du sur-échantillonnage et du sous-échantillonnage.

Par contre, on peut les simuler sur un signal numérique. En effet, sous-échantillonner un signal discret provoque sur le signal discrétisé les mêmes effets que le sous-échantillonnage d'un signal analogique (voir cours).

3.1) Echantillonnage d'un signal de test

Génération du signal

On considère un signal sinusoïdal défini par :

$$s(t) = \sin(2\pi f_0 t) \qquad \text{avec } f_0=1\text{Hz}.$$

3.1.1) Rappeler (sans démonstration) l'expression théorique de son spectre (complexe) d'amplitude et le représenter.

3.1.2) On choisit d'échantillonner ce signal à la fréquence f_e=4 Hz. Vérifier que cette fréquence d'échantillonnage est correcte, du point de vue théorique (en justifiant la réponse). Ecrire un programme permettant de générer et d'afficher quelques périodes du signal s(t).

Affichage du spectre

3.1.3) Représenter le spectre théorique du signal sur un intervalle allant de $-f_e$ à $2f_e$ (en considérant que sa durée est infinie).

3.1.4) Afficher son spectre à l'aide de la fonction fft (l'algorithme FFT, pour Fast Fourier Transform, est un algorithme de calcul rapide de la TFD, Transformée de Fourier Discrete), par exemple de la manière suivante :

```
tfd=fft(s,N)/N;     %TFD du signal s, sur N échantillons
```

Remarque : `tfd` est une variable, on peut l'appeler comme on veut !

Sous-échantillonnage

3.1.5) On simule un sous-échantillonnage de ce signal modifiant la fréquence d'échantillonnage. Donner l'expression théorique du signal non-tronqué échantillonné.

3.1.6) Observer l'effet du sous-échantillonnage en faisant varier la fréquence du signal aux valeurs suivantes :

$f_0=f_e/10$ $f_0=f_e/4$ $f_0=f_e/2$ $f_0=f_e*3/4$ $f_0=f_e*10/9$

Interpréter les résultats en raisonnant sur le spectre (pour le 3^e cas, on précisera quelles sont les valeurs théoriques des échantillons).

Filtrage anti-repliement

Pour simuler le filtrage anti-repliement, le signal va d'abord être sur-échantillonné, puis sous-échantillonné, d'abord sans puis avec filtrage.

3.1.7) Avec f_0=0,4Hz, afficher le signal sinusoïdal précédent sur-échantillonné d'un facteur 5 (en précisant f_e).

3.1.8) Pour le sous-échantillonnage, on peut utiliser par exemple les lignes de code suivantes :

```
j=1;
for i=1:N
   if mod(i,se)==0
      signal2(j)=signal(i);
      j=j+1;
   end;
end;
```

où la variable `signal` est le vecteur des échantillons du signal, et `se` est le facteur de sous-échantillonnage.

Analyser ce que font ces lignes de code, et afficher le signal ainsi sous-échantillonné (par exemple avec les valeurs suivantes de `se` : 2, 5, 7 et 9, ou d'autres valeurs), avec l'axe du temps correctement gradué. Interpréter ces résultats en mettant en évidence le problème posé par l'échantillonnage.

3.1.9) Recommencer les tests précédents en filtrant (passe-bas) le signal préalablement à son sous-échantillonnage, au moyen de la fonction `filter` et d'un filtre de Butterworth (voir exemple ci-dessous). On choisira la fréquence de coupure et l'ordre du filtre judicieusement. Reprendre les tests de la question précédente et mettre en évidence l'amélioration apportée par ce filtrage.

Exemple de programmation d'un filtre de Butterworth (et affichage de sa réponse en fréquence) :

```
...
[b,a]=butter(N,Wn);            %définir N et Wn
[H,phase]=freqz(b,a,F);         %F vecteur de fréquences à définir
semilogx(F,20*log(abs(H)));
```

3.1.10) Recommencer la même chose avec la fonction pré-définie de Matlab `decimate`.

3.2) Application à un signal sonore

On souhaite reprendre les tests précédents sur un cas concret.

On considère un signal sonore stocké dans un fichier son de type WAV. Dans les fichiers utilisés pour cette séance de TP, ce son a été échantillonné à une fréquence 44,1kHz (fréquence pour une qualité haute-fidélité, utilisée pour les CD).

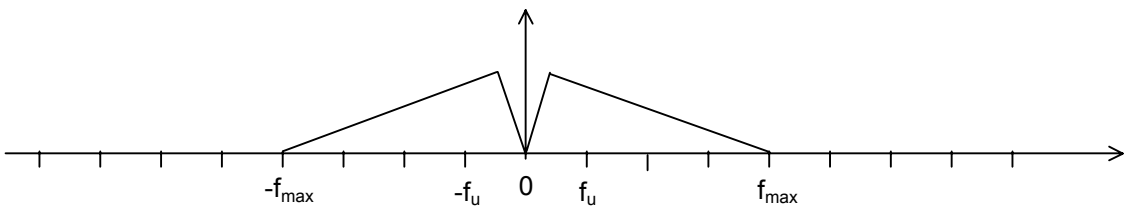
Dans une application de téléphonie, on se limite à une bande passante de 8kHz.

3.2.1) Choisir l'un des fichiers WAV proposés sur la page Internet du TP. Charger les échantillons de ce fichier en mémoire (commande `wavread`). Par tâtonnement, rechercher une partie du signal présentant une périodicité, et l'afficher en mode temporel (quelques périodes) ainsi qu'en mode fréquentiel (fonction FFT), avec les graduations correctes des axes.

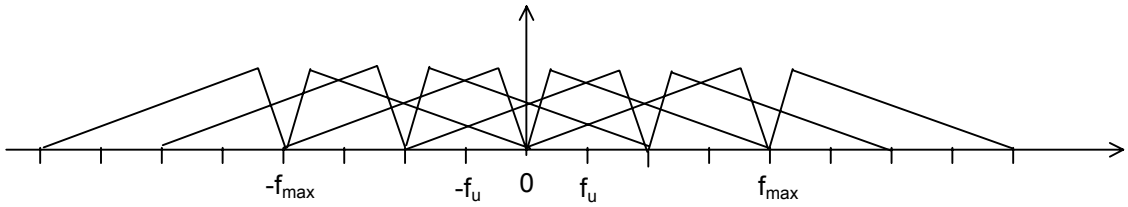
3.2.2) Relever la fréquence fondamentale (approximative) du signal, à la fois en mode temporel et en mode fréquentiel.

3.2.3) Le spectre du signal audio est celui représenté ci-dessous, avec f_{max}=22050Hz. On suppose que l'on travaille sur une application de téléphonie et que la bande utile du signal est de 8kHz ; pour simplifier, on prendra pour limite de bande utile f_u=22050/4=5512,5Hz.

On considère que le signal de 44100Hz présent dans les fichiers WAV utilisés est la version sur-échantillonnée du signal utile, pouvant être échantillonné lui à 11025Hz.



Le spectre du signal échantillonné à $2f_u$ est le suivant :



S'il n'est pas préalablement filtré, le chevauchement des spectres se traduira par des signaux parasites supplémentaires dans la bande utile.

Réaliser un sous-échantillonnage du signal chargé à partir d'un fichier WAV échantillonné à 44100Hz d'un facteur 4, et mettre en évidence le problème de repliement des spectres en comparant le signal original et le signal sous-échantillonné (si l'on dispose d'un casque, on pourra également écouter l'effet sur le son).

Remarque : cette question reprend l'exercice 1 du TD3.

3.2.4) Résoudre le problème en filtrant le signal avec un filtre de Butterworth dont les paramètres seront judicieusement choisis.

Annexe : Prise en main de Matlab

1) Introduction

Matlab est un langage interprété, c'est à dire que si l'on écrit une suite de commandes, celles-ci sont traitées une à une. Les variables utilisées dans une ligne de commande restent alors en mémoire. La liste des commandes peut également être copiée dans un fichier texte (portant l'extension .m) ; il est recommandé de le faire lorsque celle-ci s'allonge. La structure de ces programmes peut alors être très proche d'un programme en C, les déclarations de variables et les allocations mémoire en moins.

Matlab fonctionne sur le principe du calcul matriciel. Toutes les variables sont vues comme des matrices. Une variable simple (ce qu'on appelle "un scalaire") est vue comme une matrice à une ligne et une colonne ; un vecteur de N éléments est vu comme une matrice à une colonne et N lignes.

2) Pour démarrer avec Matlab

2.1) Mode "Commandes"

Le liste des commandes suivantes permettent de se familiariser à la syntaxe de Matlab.

```
help
help signal
x=2;
x
clear x
x
x=3
clear x
x=[1 2 3];
x
clear x
x=[1 2 3]          %noter la différence par rapport à la 1ère commande
y=[4;5;6]
x'
y'
clear y
y=[4 5 6]
z=[x y]
clear z
z=[x;y]
t=[0:9]
t=(0:9)
size(t)
t=[0:0.1:0.9]
t=[0:9]'
size(t)
pi
clear
for i=1:10 s(i)=i; end
s
clear
t=0:0.1:2*pi
s=sin(t)           %observer ces valeurs
```

```
plot(s)
help
plot(t,s)
clear
N=10
e=0.1
t=(0:N-1)*e
m=ones(2,3)
zeros(3,2)
eye(3,3)
m(:,2)=0
m(2,:)=0
x=[1 2 3]
y=[4;5;6]
x*y
dot(x.y')
subplot(2,1,1);                    %recommencer (les 2 lignes) avec (1,2,1)
stem(sin([0:0.1:2*pi]))
subplot(2,1,2);                    %recommencer (les 2 lignes) avec (1,2,2)
stem(sin(2*pi*100*[0:99]/10000))
```

2.2) Mode "Fichiers" (.m)

Le mode "Fichiers"

Pour sauvegarder une liste de commandes Matlab dans un fichier (qui doit porter l'extension .m), il faut ouvrir l'éditeur de commandes de Matlab. L'ouverture et la création de fichiers .m d'effectue à partir du menu *File* de la fenêtre de commandes. On appellera ces fichiers "programme Matlab".

Exécuter un programme Matlab

Pour exécuter un programme Matlab, on peut utiliser le menu *Debug*, option *Run*, de l'éditeur texte de Matlab.

Obtenir de l'aide

Complète :
help

Sur une boîte à outils (exemple : Signal Processing Toolbox) :

help signal

Sur une fonction particulière :
help fonction (remplacer fonction par un nom de fonction Matlab)

Afficher le résultat d'une commande

Il suffit de ne pas mettre de point virgule à la fin de la ligne.

2) Fonctions utiles

Remarque préalable : la plupart des fonctions Matlab peuvent comporter un nombre d'arguments variable ; de même, elles peuvent renvoyer plusieurs variables, qui peuvent être utilisées ou non selon les besoins.

`wavread, wavwrite`	Lecture et écriture de foichiers WAV
`fft`	Fast Fourier Transform : algorithme rapide de calcul de la TFD (Transformée de Fourier Discrète)
`plot`	Affichage graphique (2D) d'un tableau (sous forme de courbe)
`subplot`	Permet d'avoir plusieurs graphes sur la même figure
`stem`	Graphe d'un signal (style "discret")
`freqz`	Réponse en fréquence d'un filtre.
`Filter`	Calcul de la sortie d'un filtre
`fir1`	Conception d'un filtre à réponse impulsionnelle finie
`buttord,cheb1ord,cheb2ord,ellipord`	Synthèse des filtres numériques
`resample`	Ré-échantillonnage d'un tableau
`square`	Génération d'un signal carré
`abs, angle, real, imag, conj`	Pour des variables complexes, respectivement : module, argument, partie réelle, partie imaginaire, conjugué.