Travaux Pratiques de traitement du signal

Vous prendrez soin de faire un compte-rendu de TP indiquant les points jugés essentiels, les difficultés rencontrées et les solutions alors trouvées. Attention à bien séparer les points liés au logiciel Octave des points liés à la discipline étudiée (le traitement du signal).

Ce compte-rendu doit être rédigé de façon à vous permettre de refaire très facilement le TP et d'en maîtriser toutes les notions abordées. Il est donc fortement recommandé pour chaque question d'indiquer : la(les) commande(s) entrée(s), les copies d'écran des figures, et les commentaires/conclusions.

Certaines parties devront être préparées préalablement : les questions correspondantes sont soulignées en rouge dans le texte.

---SEANCE 1---

L'objectif de cette séance est double : découvrir et prendre en main le logiciel Octave tout en réalisant les premières applications de traitement du signal.

La première partie consiste à découvrir l'environnement Octave (et les différentes fenêtres associées de l'interface) et à utiliser quelques commandes de base.

Dans la deuxième partie, un premier programme Octave sera réalisé en suivant les recommandations de programmation et de présentation données.

1. Environnement de travail

a) Connectez-vous sur une station de travail.

b) Démarrer le logiciel Octave.

c) Repérer le répertoire où stocker vos fichiers.

d) Identifiez les différentes fenêtres de l'environnement proposé par Octave. Il sera important en fin de TP d'avoir retenu le rôle et l'intérêt de chacune de ses fenêtres.

2. Commandes de bases

La commande la plus importante à retenir est *help* ; celle-ci vous servira régulièrement.

A l'aide de Octave, répondre aux questions qui suivent.

a) Calculs élémentaires – Octave utilisé en « simple calculatrice »

- Taper dans la fenêtre de commandes : *help sqrt*

Pour toutes les fonctions Octave, une telle aide est disponible. Pour plus d'informations, il existe aussi une aide en ligne plus complète qui fournit notamment des exemples.

- Calculer $\sqrt{10}$ et vérifier que $\left(\sqrt{10}\right)^2 = 10$.
  Fonction *sqrt*
- Afficher j et vérifier que $j^2 = -1$ (remarque : le même résultat est obtenu avec i).
  Fonction « ^ »
- Calculer $cos\left(\frac{\pi}{2}\right)$, $sin\left(\frac{\pi}{2}\right)$ et $exp\left(j\frac{\pi}{2}\right)$.
  Fonction *pi*

b) « Nettoyage » de l'environnement de travail

- Effacer l'ensemble des variables générées.
  Fonction *clear all*
- Effacer les affichages en lignes de commande.
  Fonction *clc*
- Complément : pour fermer les fenêtres de figures créées.
  Fonction *close all*

c) Création de variables

Pour créer une variable, il suffit d'écrire : *nom_de_la_variable* = *valeur*.

ATTENTION : les noms de variable doivent être différents des noms de fonctions Octave.

- Par exemple, entrer et exécuter les lignes suivantes (noter l'impact du « ; ») :
  - a = 3
  - b = -1;
  - maVariable = $3*cos\left(\frac{\pi}{3}\right)$;
- Calculer $x = (1 + 3i)(2 + 2i)(3 + i)$
- En déduire son module, noté *mod*.
  Fonction *abs*
- En déduire son argument, noté *ang*, en degrés
  Fonction *angle*

d) Création de vecteurs

- Entrer et exécuter les lignes suivantes (noter l'impact du « ; ») :
  - v1 = [5 ; 6 ; -3 ; 2];
  - v2 = [5 6 -3 2];
  - v3 = [5 , 6 , -3 , 2];
- Identifier quels sont les vecteurs *ligne* et les vecteur *colonne* en utilisant le 'workspace'. Retenir la syntaxe correspondante dans chacun des trois cas.

3. Programmes Octave

Plutôt que de taper pas à pas les commandes dans le terminal (c'est-à-dire dans la fenêtre de commande), l'ensemble des commandes sont écrites dans un fichier texte d'extension « .m », appelé fichier script, qui est exécuté dans Octave.

a) Illustration par un premier exemple

L'exemple en annexe est une illustration d'un programme Octave correctement écrit dont il faudra s'inspirer de la forme ; remarquer notamment les nombreux commentaires et la séparation des différentes parties.

- Sans le recopier, ni l'exécuter, seulement en le lisant, déduire les différentes fonctions réalisées.

b) Premier programme et vecteurs

- Créer un fichier script *TP1question3b.m* et repérer le répertoire où il est créé.

Dans ce fichier, le programme va être écrit : ce sont des lignes de code (comme présentées en annexe) qui s'exécutent séquentiellement (c'est-à-dire, les unes après les autres, dans l'ordre : de haut en bas, de gauche à droite).

REMARQUES :

i- A tout moment, il est possible d'exécuter (c'est-à-dire lancer) le programme pour vérifier que le code s'exécute comme attendu. Le programme peut (et doit) être complété en vérifiant son bon fonctionnement au fur et à mesure.

ii- Penser à vérifier dans la fenêtre de commande si aucune erreur de syntaxe n'est signalée ; si une erreur arrive, c'est dans cette fenêtre que l'utilisateur est informé.

iii- Le cas échéant, corriger les erreurs et ré-exécuter le programme.

Ecrire les instructions qui permettent de répondre aux questions qui suivent.

- Commencer par « nettoyer » l'environnement de travail.

___Calculs avec les vecteurs___

- Construire le vecteur ligne *v* contenant les nombres entiers pairs de 1 à 20.
  Fonction « : »
- Utiliser le vecteur *v* pour calculer la racine carrée des nombres entiers pairs de 1 à 20.
- Ajouter 1 à chaque élément de *v* et sauvegarder ce résultat dans un vecteur (ou variable) *w*.
- Calculer le vecteur *u* contenant tous les éléments de *v* au carré.
  Fonction « .^ »
- Calculer le vecteur *prod* contenant les éléments de *v* et de *w* multipliés terme à terme.
  Fonction « .* »

A noter :

Au fur et à mesure des questions qui précèdent, il est important de vérifier les résultats et notamment utiliser les informations donnés dans le 'workspace'. Un double-clic sur la variable ou vecteur permet de le lire.

___Vecteur ligne et vecteur colonne___

- Transformer le vecteur ligne *v* en vecteur colonne.
  Fonction « .' »

La fonction « ' » (sans le point) réalise la même opération mais en conjuguant les valeurs si elles sont complexes. Pour le vérifier :

- Générer un vecteur colonne *A* constitué des valeurs suivantes : $1 + i$; 5 ; $3 - i$.

- Calculer et comparer : B = A' et C = A.'.

___Affichage graphique___

- Tracer le vecteur *v*. Remarquer l'abscisse mise par défaut.
  Fonction *plot*
- Dans une autre figure, tracer le vecteur *w* (ordonnée) en fonction du vecteur *v* (abscisse).
  Fonction *figure*
- Sur cette dernière figure, ajouter un titre, et la signification de l'abscisse et de l'ordonnée.
  Fonctions *title* et *xlabel, ylabel*
- Dans une autre figure, tracer les vecteurs w et u en fonction du vecteur v ; distinguer les courbes par deux couleurs différentes et en affichant un symbole sur chaque point ; ajouter la légende correspondante.
  Fonction *legend*

4. Lecture et caractérisation d'un signal sonore de type « .wav »

- Créer un nouveau fichier script et repérer le répertoire où il est créé.
- Placer le fichier « *song335*.wav » (disponible dans Claroline) dans ce même répertoire de travail.
- Ecrire dans ce fichier un programme réalisant les fonctions suivantes et permettant de répondre aux questions :

- Nettoyage de l'environnement de travail.

À l'aide des commandes *audioread* et *audioinfo* :

- Générer le vecteur de valeurs correspondant au signal enregistré, noté s.
- S'agit-il d'un vecteur ligne ou d'un vecteur colonne ?
- Donner la valeur de la fréquence d'échantillonnage du signal, notée Fe.
- Donner la valeur de la période d'échantillonnage notée Te.
- Donner le nombre de bits de quantification, noté Nq.
- Donner le nombre total de points du signal noté N.
- Donner la durée du signal, notée D.

Aide : lire un fichier au format « .wav »

Soit un fichier « *nom_fichier.wav* ». La commande *audioread* appliquée comme illustré permet d'obtenir *x* le vecteur de valeurs correspondant et *FS* la fréquence d'échantillonnage :

```
[x, FS] = audioread('nom_fichier');
```

Il est aussi possible d'avoir accès à plusieurs informations, comme suit :

```
info = audioinfo('nom_fichier');  % création de la variable « info » qui contient ces informations
NBITS = info.BitsPerSample;    % nombre de bits de quantification
T = info.Duration;             % durée du signal en seconde
NECH = info.TotalSamples;      % nombre total d'échantillons
```

Pour plus d'informations sur les fonctions *audioread* et *audioinfo*, consulter l'aide de Octave

Pour la suite, le programme est complété (et exécuté) au fur et à mesure des questions.

- Représenter le signal s en fonction du temps t sur une figure qui comportera un titre et indiquera que l'abscisse est le temps.
- À partir de la représentation obtenue, que pouvez-vous dire sur le signal *song335* ?

Pour aller plus loin :

- Extraire du signal *s*, un signal noté *s2*, égal à *s* pendant les 2 premières secondes.
- Extraire du signal *s*, un signal noté *s3*, recopiant 1000 échantillons de *s*, à partir de l'échantillon 10.
- Représenter les signaux s, *s2* et *s3* en fonction du temps t, chacun dans un graphe, et les trois graphes sur une même figure (à l'aide de la Fonction *subplot*).

___Fin de la séance 1___

ANNEXE : exemple de programme Octave

```
%% --- Travaux Pratiques de Traitement du Signal ---%
%
% ANNEXE du sujet de TP rappelant les grandes lignes
% pour structurer correctement un programme Matlab
%_____________________________________________________


%% --- Démarrer avec un environnement "propre"
clear all; % effacer toutes les variables de l'espace de travail (workspace)
close all; % fermer toutes les figures ouvertes
clc;       % nettoyer la fenêtre ligne de commande (command window)



%% --- Quelques recommandations à suivre à travers un exemple simple
a = [0:1:99]     % à éviter (voir suite)

N = 100;         % utiliser un paramètre qui sera facile à modifier

a = [0:1:N-1]    % ajouter un ";" pour ne pas afficher les valeurs en
a = [0:1:N-1];   % ligne de commande et surcharger l'exécution du programme



%% --- Quelques fonctions utiles pour les graphiques
figure(1); plot(a)

grid;
title('Un exemple illustrant les recommandations à suivre')
xlabel('sans abscisse spécifiée dans plot, indice de 1 à length(a)')
ylabel('vecteur a')

% Observer les différences avec les options qui suivent
% Choisir l'option adéquate selon les cas
figure(2); stem(a)

figure(3);
subplot(211); plot(a)
subplot(212); plot(a,'rx')

%% --- COMMENTER et STRUCTURER les programmes !!
```

---SEANCE 2---

L'objectif de cette deuxième séance est :

i) d'apprendre à générer un signal de caractéristiques données sous Octave (versus utiliser un fichier de type « .wav », comme lors de la séance 1) ;

ii) d'étudier la convolution et la corrélation à partir des deux programmes Octave fournis.

a) Génération d'un signal et caractérisation dans le domaine temporel

- Créer un fichier script *TP1.m* et repérer le répertoire où il est créé.
- Ecrire (en suivant les recommandations précédentes) dans ce fichier un programme réalisant les fonctions suivantes :
  - o Nettoyage de l'environnement de travail.
  - o Création d'un vecteur de 1024 points contenant un signal sinusoïdal de fréquence 100 Hz échantillonné à 1000 Hz.
  - o Affichage de ce signal en fonction du numéro des points abscisse.
  - o Affichage de ce signal en fonction du temps en seconde avec un titre et une légende à chaque axe.
- Exécuter (lancer) le programme ; vérifier dans la fenêtre de commande si aucune erreur de syntaxe n'est signalée ; si une erreur est signalée, la corriger et ré-exécuter le programme ; si aucune erreur n'est signalée, en créer une volontairement et observer le message indiqué.
- Modifier la fréquence du signal sinusoïdal et observer le résultat.

Avant-propos : partie à réaliser à l'aide du logiciel Matlab

Télécharger le fichiers *audio_conv.m* et *audio_corr.m* disponibles sur Claroline.

b) Etude de la convolution à partir du programme fourni

- Lancer le script *audio_conv.m*.
- Essayer différentes combinaisons de signaux et retrouver (comprendre) le principe du calcul de la convolution (ici décomposé et illustré).
- Dans le cas où y représenterait la réponse impulsionnelle d'un filtre :

  + observer le résultat de la convolution pour les deux sinus ; en déduire quel est le type de filtre appliqué ?

  + Quelle est la transformée de Fourier d'un sinus cardinal ? En déduire la réponse en fréquence du filtre. Est-ce cohérent avec le résultat précédent ?

  + Appliquer en entrée un Dirac. Quel est le signal obtenu en sortie ?

c) Etude de la corrélation à partir du programme fourni

- Lancer le script *audio_corr.m*.
- Essayer différentes combinaisons de signaux et retrouver (comprendre) des résultats connus.

___Fin de la séance 2___

---SEANCE 3---

L'objectif de cette troisième séance est d'étudier l'influence des différents paramètres rentrant en jeu lors de la génération d'un signal ainsi que la représentation fréquentielle des signaux. Plusieurs types de signaux seront générés et observés en temps et en fréquence.

Il est conseillé d'écrire un programme Octave pour chaque partie indépendante.

1. Synthèse de signaux et échantillonnage

a) Etude de signaux échantillonnés et paramètres associés

- Ecrire un programme Octave permettant de générer les signaux suivants et d'afficher sur une même figure les 100 premiers échantillons de chaque signal par rapport au temps.

Signal #	F_e	Fréquence	Nombre de points
1	1000	100	512
2	1000	200	1024
3	2000	200	512
4	2000	200	1024
5	44 100	50	44 100
6	44 100	20 000	44 100
7	5000	4000	2048

- Exécuter le programme réalisé ; corriger les éventuelles erreurs.

- Comparer les différents cas de figures rencontrés et conclure.

- Ecouter les signaux 1 et 2 avec la commande *wavplay*. Varier la fréquence d'échantillonnage. Comparer le signal 1 écouté avec $F_e = 1000\ Hz$ et le signal 2 avec $F_e = 2000\ Hz$. Conclure.

b) Génération et affichage de signaux particuliers

(i) Signal composite

- Ecrire un programme Octave qui permet de créer et d'afficher un signal contenant deux sinus de même amplitude et de fréquence respective $F_1 = 100\ Hz$ et $F_2 = 200\ Hz$ (avec $F_e = 1000\ Hz$).

- Exécuter le programme réalisé ; corriger les éventuelles erreurs.

(ii) Signal aléatoire

- Ecrire un programme Octave qui permet de créer et d'afficher un signal aléatoire dont l'amplitude est comprise entre -1 et +1. Pour cela, voir dans l'aide comment créer un nombre aléatoire. Quelle est la fréquence d'échantillonnage correspondante ?

- Exécuter le programme réalisé ; corriger les éventuelles erreurs.

2. Représentation spectrale de signaux – TFD et FFT

a) Etude de la représentation spectrale d'un signal composite

- A l'aide des fonctions *fft* et *abs*, afficher le module de la transformée de Fourier (appelé spectre) du signal de la question précédente contenant les deux sinus.

- Où se situe la fréquence $f = 0$?

- Quel doit être l'axe en fréquence du spectre ? Créer cet axe et afficher le spectre en fonction de la fréquence. Conclure par rapport à la théorie.

- Il est possible « artificiellement » de retrouver l'allure obtenue dans le cas continu. Placer la fréquence $f = 0$ au centre de l'affichage à l'aide de la fonction *fftshift*.

b) Prolongement par des zéros ('zero-padding')

- Reprendre les questions de la partie 2a) en considérant le même signal mais auquel auront été ajoutées M valeurs nulles ; autrement dit la taille du vecteur signal est augmentée en ajoutant des zéros par exemple comme illustré dans les deux cas suivants :

(i) $\left[\underbrace{x \quad \dots \quad x}_{vecteur\ initial} \quad \underbrace{0 \quad 0 \quad 0 \quad \cdots \quad 0}_{vecteur\ de\ M\ z\acute{e}ros}\right]$

(ii) $\left[\underbrace{0 \quad \cdots \quad 0}_{\substack{vecteur\ de \\ M/2\ z\acute{e}ros}} \quad \underbrace{x \quad \dots \quad x}_{\substack{vecteur \\ initial}} \quad \underbrace{0 \quad \cdots \quad 0}_{\substack{vecteur\ de \\ M/2\ z\acute{e}ros}}\right]$

- Commenter et analyser les résultats obtenus.

- Relancer les programmes réalisés (sans et avec prolongement par zéros) en modifiant les fréquences du signal tel que : $F_1 = 180\ Hz$ et F_2 inchangée. Observer l'impact de la valeur de M.

- Quels sont les avantages / inconvénients du prolongement par zéros ?

c) Etude d'un signal porte

- Créer un signal porte. Afficher ce signal, son spectre et sa phase (fonction *angle*). Conclure.

- Créer un nouveau signal porte, retardé par rapport au précédent. Afficher ce signal, son spectre et sa phase. Conclure.

Pour aller plus loin :

- Représenter la transformée de Fourier discrète du signal « song335.wav » étudié lors de la séance 1.

- Quelles sont les trois fréquences physiques les plus grandes contenues dans le signal ?

- Représenter le spectrogramme du signal s en considérant un calcul de *FFT* par tranche de 256 échantillons avec un recouvrement de 64 échantillons. Commenter la figure obtenue.

Rappel : la commande Octave est : *spectrogram* (voir aide de Octave pour plus d'informations)

___Fin de la séance 3___