TP Traitement du signal

Dr. TALEB Sarra

1.0 2023/2024

Université de Tlemcen

Table des matières

Objectifs

Ce TP est destiné aux étudiants de troisième année licence en télécommunications. Les textes de travaux pratiques présentés sont constitués d'un rappel théorique suivi d'une série de manipulations numériques. Ces manipulations permettent à l'étudiant de :

- **Niveau connaissance** : Vérifier pratiquement sur machine les différentes notions théoriques présentées dans le cours "traitement du signal".
- **Niveau compréhension** : Intégrer l'étudiant avec les techniques de traitement numérique du signal.
- **Niveau d'application** : Pratiquer l'étudiant à la programmation.
- **Niveau d'application :** Appliquer des opérations mathématiques élémentaires sur les signaux temporels pour les modifier.
- **Niveau d'évaluation** : Évaluer l'effet des opérations appliquées sur les propriétés des signaux temporels.

Introduction

Un signal est la représentation d'une mesure d'un phénomène physique. A l'heure du tout numérique, les notions de traitement numérique du signal attirent de plus en plus l'intérêt des praticiens. De ce fait, cette discipline mérite une attention particulière avec le souci de simplicité dans sa diffusion auprès des étudiants. Le traitement du signal vise à comprendre, modifier et exploiter les informations contenues dans les signaux.

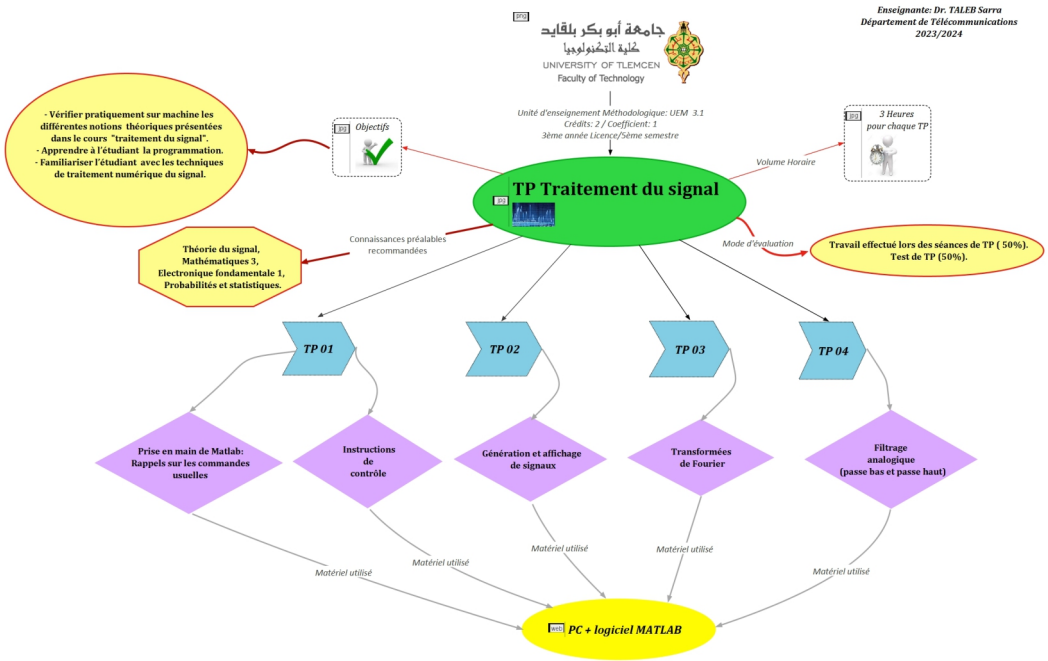
Le traitement du signal est un domaine vaste et crucial qui s'intéresse à l'analyse, à la manipulation et à la transformation de signaux. Il intervient dans une multitude d'applications scientifiques et industrielles, touchant des domaines aussi divers que les télécommunications, l'électronique, le traitement d'images, la reconnaissance vocale, la médecine et bien d'autres encore.

Ainsi c'est un domaine en constante évolution avec de nombreuses nouvelles applications et techniques émergentes. Il joue un rôle crucial dans l'ère numérique en permettant de manipuler et d'exploiter des informations précieuses contenues dans les signaux qui nous entourent.

Ce manuel comporte quatres TP qui résument l'ensemble des connaissances de base de traitement du signal using Matlab, un outil largement utilisé dans divers domaines tels que l'électronique, les télécommunications, la robotique, le traitement d'images et l'analyse de données.

Pour commencer, TP I se divise en deux parties "Prise en main de Matlab: Rappels sur les commandes usuelles" et "Instructions de contrôle". De fait, ce chapitre constitue une base de révision pour que l'étudiant puisse aborder plus aisément les TP suivants. Vient ensuite, le TP II "Génération et affichage de signaux" par exemple :Sinusoïdaux, impulsion, rectangulaire, etc . Le TP III "Transformée de Fourier" étudie la transformée de Fourier appliquée à un signal. Dans le chapitre IV "Filtrage analogique" on s'intéresse à l'étude des filtres passe bas et passe haut.

La figure suivante montre la structure de ce manuel "**carte conceptuelle**" :



TP 1.1 : Prise en main de Matlab: Rappels sur les commandes usuelles

« Matlab » est un logiciel conçu pour les calculs numériques. Le nom « deux mots « **Mat**rix **Lab**oratory ». Comme ce nom indique, Matlab est basé sur le traitement des matrices. Autrement dit, Matlab manipule les données,les variables,les image,... comme étant des matrices. Ce logiciel est developpé par la société américainne « mathworks » (le site www.mathworks.com). Le logiciel Matlab permet d'analyser les données , de developper les algorithmes,de faire les calculs mathématiques , de faire des simulations et des modélisations des différents systémes et de representer graphiquement les résultats obtenus *[1]*.

Ce TP doit permettre à tout étudiant de maîtriser l'environnement du logiciel de manière à pouvoir travailler de manière autonome et plus précis dans le domaine du traitement du signal.

1. Objectifs spécifiques

Ce TP a pour objectifs de :

- **Niveau compréhension** : Utiliser les commandes usuelles de Matlab, un langage de programmation et un environnement de calcul numérique largement utilisé dans les domaines scientifiques et d'ingénierie.
- **Niveau d'application** : Exercer des opérations arithmétiques et logiques.
- **Niveau d'évaluation** : Vérifier par des tests les connaissances acquises lors de ce TP1.1.

2. Instructions élémentaires utilisées dans MATLAB

Manipulations de variables et opérations arithmétiques sur les matrices

Après avoir cliqué sur l'icône MATLAB dans WINDOWS, vous vous trouvez dans la feuille de calcul de MATLAB que l'on appellera fenêtre MATLAB. Le curseur se situe derrière 'l'invite' représenté par ce symbole: >>

A cet endroit vous pouvez utiliser MATLAB aussi simplement qu'une machine à calculer.

Taper par exemple 2+3 puis **Entrée**, le résultat apparaît directement à la suite.

Le grand avantage de MATLAB est d'offrir à l'utilisateur un grand nombre de fonctions dédiées soit au calcul numérique, soit au traitement du signal, soit à l'automatique ou encore aux réseaux de neurones.

Assignation d'une variable : taper **a=2** puis '**Entrée**' (pour valider votre opération) et la variable a vaudra 2.

Attention MATLAB fait la différence entre les majuscules et les minuscules.

Taper par exemple a et le contenu de la variable a sera affiché, c'est à dire 2. Si vous tapez A aucune valeur ne sera affichée puisque la variable A n'est pas allouée. Vous pouvez fort bien ré allouer une autre valeur à la variable a. Taper par exemple: **a=[1 2; 3 4]**.

La variable a est ainsi devenue une matrice 2 x 2. Dans la définition de la matrice a donnée ci dessus, les crochets servent à englober tous les éléments et le point virgule permet de séparer chaque ligne. A noter que les indices partent obligatoirement de 1, ils ne sont pas paramétrables comme dans d'autres

langages. Ainsi si vous tapez **a(1,1)**, le résultat affiché sera 1. Le chiffre 1 sera donc l'élément (1,1) de la matrice a; on ne pourra pas changer cela pour qu'il devienne l'élément (0,0). Tester ces instructions en tapant **a(1, :)** et **a(:,2).**

Pour transposer la matrice, il suffit de taper a'. Les transpositions peuvent être utiles pour transformer un vecteur ligne en vecteur colonne, ou inversement. Pour créer une matrice, nous avons vu qu'il était possible de la définir à partir de l'ensemble de ses éléments mis entre crochets en séparant les lignes par des points virgules. Dans le cas de matrices élémentaires, Matlab permet de définir celles-ci de manière simple, pour définir une matrice x ne contenant que des 0 par exemple, il suffit de taper : **x=zeros(m,n)** où m et n représentent respectivement le nombre de lignes et le nombre de colonnes.

Exemple **x=zeros(3,2)** ou **x=ones(4,3)** ;

Passons à présent aux opérations matricielles (voir le tableau suivant) :

Addition:	taper: b=2+a et vérifier l'opération réalisée. taper: c=a+b et vérifier le résultat.
Soustraction:	taper: b=b - 2 et vérifier que b vaut à présent a. taper: b=b - a et vérifier que les éléments de b sont nuls.
Multiplication:	taper: b=2*a et noter ce que vaut b taper: c=a.*b et relever les valeurs de c pour comprendre comment les éléments de a et b sont multipliés entre eux. taper: d=a*b et donner la différence par rapport à l'opération précédente. Attention cette dernière opération n'est pas commutative contrairement au .*. Le produit b*a est donc différent de d (sauf cas particulier comme ici).
Division:	taper: c./a et comparer le résultat avec b. taper: c./b et comparer le résultat avec a. taper: d/b et vérifier que le résultat correspond au produit de d par l'inverse de b.
Puissance :	taper: a.^2 et expliquer le type d'opération effectuée. taper: a^2 et comparer ce résultat avec le précédent.

Nous venons de créer un certain nombre de variables. Contrairement aux langages de programmation les plus courants, il n'y a pas besoin de déclarer les variables ainsi que leur type (entier, réel ou autre). Tapez la commande **who** pour connaître les différentes variables utilisées. Pour effacer les variables, tapez **clear *[2]*.**

3. Génération de matrices ou de vecteurs et calculs sur les nombres complexes en traitement du signal

En traitement du signal, nous travaillerons surtout avec des vecteurs qui correspondent à des matrices à une seule ligne ou une seule colonne. Ces vecteurs permettent de simuler un signal échantillonné.

Prenons l'exemple de la fonction : y(t)=sin(2πt) (évitez cette écriture sur Matlab, suivez les instructions suivantes).

Nous définissons tout d'abord un vecteur temps par l'instruction suivante :

t=t0 :te :tf avec t0 : l'instant initial choisi, te : la période d'échantillonnage, tf : le temps correspondant au dernier échantillon. Une autre écriture possible est : t=[t0 :te :tf].

Pour les simulations, prenez par exemple : t=0 :0.1 :1

Pour définir y, il suffit ensuite de taper : y=sin(2*pi*t). La variable pi est prédéfinie dans Matlab et correspond bien sur à la valeur de π.

Si vous voulez avoir de l'aide sur une fonction dont vous connaissez le nom, il suffit de taper **help** suivi du nom de l'instruction dans la fenêtre de travail pour avoir des renseignements sur la syntaxe et le rôle de cette instruction *[2]*.

3.1. Graphisme

Afin de vous guider nous présenterons en premier lieu la fonction graphique élémentaire: **plot**

Utilisation de plot pour tracer un graphe en 2 dimensions :

Reprenons les données t et y générées auparavant.

Le vecteur y contenant des nombres complexes, si vous tapez plot(y), le graphe obtenu représentera la partie imaginaire de y tracée en fonction de la partie réelle.

Si vous voulez tracer la fonction sinus, tapez plot(imag(y)) ou plot(t,imag(y)) si vous voulez avoir en abscisse le vecteur temps t.

Si vous voulez tracer la fonction sinus et la fonction cosinus sur un même graphe, tapez plot(t,real(y),t,imag(y)).

Tapez la suite d'instructions suivantes :

t=0 :0.001 :1 ;

y1=cos(2*pi*t) ;

plot(t,y1)

hold on

y2=cos(4*pi*t) ;

plot(t,y2)

Vous pouvez fort bien spécifier la couleur et le style de tracé en rajoutant une chaîne de caractères S dans l'instruction: plot(X,Y,S). Par exemple, si vous tapez: plot(t,real(y),'b+'), vous tracerez des + de couleur bleue. Pour connaître les différentes possibilités de tracé, tapez help plot.

Tracé de plusieurs graphes dans la même fenêtre :

Grâce à l'instruction subplot, il est possible de spécifier le nombre de graphes que l'on veut tracer dans une fenêtre. subplot(M,N,P) permet de diviser la fenêtre en M x N, et à la prochaine instruction plot, le graphe sera tracé à l'emplacement n° P. A titre d'exemple taper:

subplot(1,3,1)

plot(t,real(y))

subplot(1,3,2)

```
plot(t,imag(y))
subplot(1,3,3)
plot(real(y),imag(y))
```

3.2. Programmation structurée, et instructions conditionnelles

L'écriture de fichiers de commandes ou scripts :

Un fichier de commandes ou script est une séquence d'instructions Matlab. Ces fichiers portent toujours l'extension **.m**. Les variables créées dans ces fichiers sont globales, ce qui signifie que depuis la fenêtre Matlab vous pouvez accéder aux contenus de ces variables. Les fichiers de commandes permettent la saisie de données grâce à l'instruction **input** utilisée dans l'exemple qui va vous être présenté.

Nous allons écrire un script qui permet le tracé d'une fonction $y=x^2+5$. En haut de la fenêtre Matlab, cliquez sur **File** puis **New** et **M-File**. Puis reportez dans l'éditeur qui vient de s'ouvrir (MATLAB Editor/Debugger) la suite d'instructions suivantes:

```
clc
T=input('Durée du signal = ');
N=input('Nombre de points = ');
t=0:T/N:T-T/N;
y=t.^2+5;
plot(t,y);
```

Pour sauvegarder ce script, il faut au préalable créer un répertoire de travail sur le disque dur, nommez le par exemple test. Cliquez ensuite sur **File** puis **Save as** et donnez le nom suivant: **courbe1** ainsi que le répertoire que vous venez de créer. Revenez dans la fenêtre de Matlab et tapez: **courbe1**.

Vous devrez donner une durée du signal et un nombre de points avant de visualiser le tracé de la fonction.

L'écriture de fichiers de fonctions :

Vous pouvez créer de nouvelles fonctions spécifiques à votre domaine de travail qui auront le même statut que toutes les autres fonctions Matlab, comme par exemple les fonctions sin, cos, sinc,...

L'avantage de fonctions par rapport aux scripts est le paramétrage facile d'entrées et de sorties, ce qui permet une utilisation commode et une intégration possible dans un script.

A titre d'application, nous allons écrire une fonction pour générer un tableau de N nombres correspondant au calcul de la fonction $y=x^2+5$ sur une durée T.

Les fichiers de fonctions portent également l'extension m et sont créés avec le même éditeur que celui utilisé précédemment pour générer des fichiers de commandes. Voici la suite d'instructions que l'on vous demande de taper dans un fichier de fonctions que vous nommerez **signal1**:

```
function res=signal1(T,N)
% res: vecteur correspondant au signal y=x^2+5
% T: durée du signal
% N: nombre de points temps=0:T/N:T-T/N;
res=temps.^2+5;
```

La première ligne déclare le nom de la fonction, les arguments d'entrée et de sortie. Sans cette première ligne, le fichier correspondrait plutôt à un fichier de commandes ou script. Pour exécuter cette fonction après l'avoir enregistré, tapez **y=signal1(1,50)** dans la fenêtre Matlab. La variable y comprendra alors 50 valeurs calculées à partir d'un vecteur temps allant de 0 à 1seconde. Les lignes commençant par % dans un fichier de fonctions ou un script permettent de donner des commentaires.

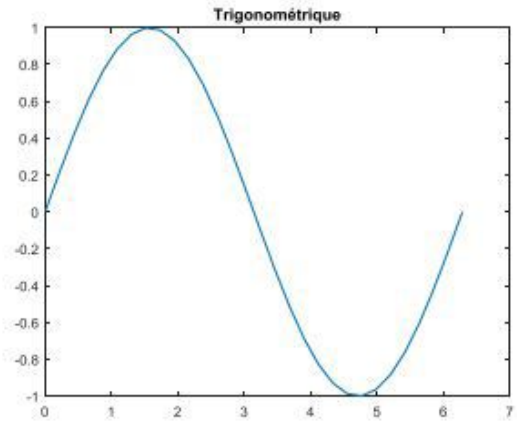
4. Représentation graphique sous Matlab

Les fonctionnalités graphiques de Matlab se séparent en trois catégories principales : les représentations 2D regroupées sous la terminologie **graph2d**, les représentations 3D regroupées sous la terminologie **graph3d** *[3]*.

4.1. Graphiques à 2D

L'instruction graph2d permet de connaître la liste des graphiques 2D et tableaux des fonctions disponibles. Le programme suivant illustre un exemple :

```
x = linspace(0,2*pi,30);
y = sin(x);
figure(1);
plot(x,y, '-');
title('Trigonométrique');
```

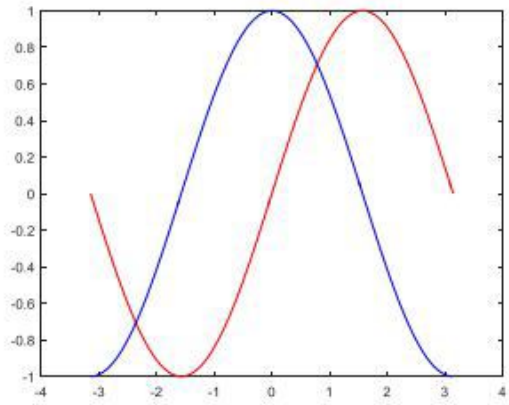


De nombreuses options existent sous Matlab pour contrôler l'affichage des courbes (Axe, couleur, commentaires, légende, échelle logarithmique,... etc).

Pour ajouter une courbe à un graphique existant, il suffit d'utiliser l'instruction **hold on**. Dans ce cas, les instructions graphiques qui suivent seront réalisées sur la même figure.

Par exemple, le programme suivant permet de tracer graphiquement en mode superposé les fonctions sin(x) et cos(x) sur l'intervalle [−π, π] avec 201 points.

```
x = -pi:pi/100:pi;
y = sin(x);
z = cos(x);
plot(x,y, 'r');
hold on;
plot(x,z, 'b');
hold off;
```



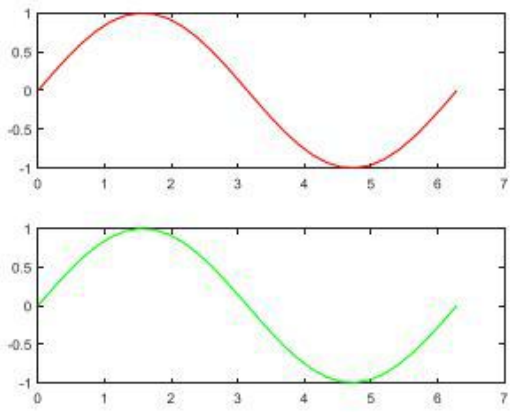
Pour subdiviser la fenêtre graphique et donc tracer plusieurs graphiques sur la même figure, nous utilisons l'instruction **subplot** (sous-graphique). Sa syntaxe est :

subplot(nombre lignes, nombre colonnes, numéro subdivision)

Les subdivisions sont numéroter de 1 à nombre lignes * nombre colonnes, de la gauche vers la droite puis de haut en bas.

Un exemple est donné ci-dessous pour illustrer le fonctionnement de cette instruction :

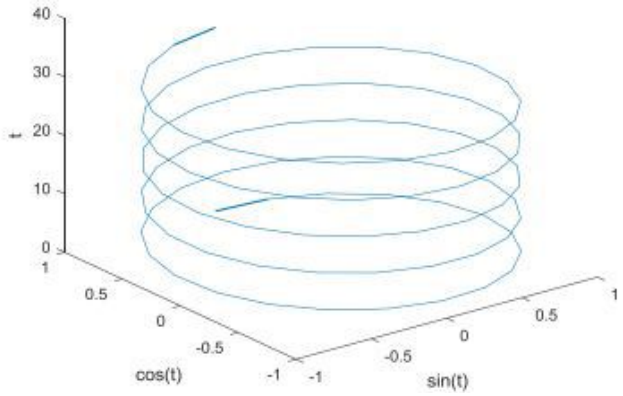
```
x = linspace(0,2*pi,30);
y = sin(x);
z = cos(x);
subplot(2,1,1);
plot(x,y, 'r');
subplot(2,1,2);
plot(x,y, 'g');
```



4.2. Graphiques à 3D

Pour tracer une courbe paramétrée avec 3 coordonnées, nous utilisons l'instruction plot3.

```
t = linspace(0,10*pi);
x = sin(t);
y = cos(t);
plot3(x,y,t);
xlabel('sin(t)');
ylabel( 'cos(t)');
zlabel('t');
```



5. Exercice à faire :

A titre d'exercice utiliser les instructions if et for pour créer une matrice X de dimension 5 x 5 où les éléments de la diagonale sont égaux à 5 et les autres à 1. La syntaxe de ce type d'instructions (for, while, if) est décrite dans l'aide (option Help, puis Help Window).

6. Solution :

```
% Initialiser une matrice 5x5 avec des éléments nuls
X = zeros(5, 5);
% Utiliser une boucle for pour parcourir les éléments de la matrice
for i = 1:5
    for j = 1:5
        % Utiliser une instruction if pour vérifier si l'élément est sur la diagonale
        if i == j
            % Si l'élément est sur la diagonale, attribuer la valeur 5
            X(i, j) = 5;
        else
            % Sinon, attribuer la valeur 1
            X(i, j) = 1;
        end
    end
end
% Afficher la matrice résultante
disp(X);

```

```
>> solTP01
     5     1     1     1     1
     1     5     1     1     1
     1     1     5     1     1
     1     1     1     5     1
     1     1     1     1     5
```

Ce TP vous permettra de faire vos premiers pas dans le monde de Matlab et de découvrir les commandes usuelles de ce langage puissant. En vous familiarisant avec les concepts de base, vous serez en mesure d'aborder des tâches plus complexes et de développer vos compétences en programmation et en calcul numérique.