Introduction

MATLAB, pour MATrix LABoratory, est un logiciel de calcul numérique, qui permet de traiter des problèmes complexes dans de nombreux domaines scientifiques, d'exploiter et de visualiser des données en 1D/2D/3D, ..., dans un environnement interactif.

Il est en particulier couramment utilisé par les scientifiques et les ingénieurs, pour réaliser les différentes tâches de traitement du signal.

De nombreuses fonctions sont disponibles dans la boîte à outils *Signal Processing* que nous exploiterons lors de ces séances de TD.

La boîte à outils *Signal Processing* comprend des fonctions et des interfaces graphiques (Apps) pour générer, visualiser, transformer, filtrer des signaux. Des algorithmes sont mis à votre disposition pour :

- approcher le spectre de signaux par transformée de Fourier discrète ;
- estimer les spectres de puissance de signaux aléatoires ;
- concevoir et analyser des filtres ;
- estimer des modèles paramétriques de signaux aléatoires ;
- réaliser une prévision de séries temporelles.

Cette boîte à outils permet donc d'analyser et de comparer des signaux dans les domaines temporel, fréquentiel et temps-fréquence, identifier des motifs et des tendances, extraire des caractéristiques, et développer et valider des algorithmes personnalisés afin d'obtenir des informations sur vos données.

Vous allez réaliser des traitements sur des données audio. Apportez vos écouteurs afin de pouvoir entendre l'effet de vos traitements numériques.

TD 1

Jeux de mots et série de Fourier

Objectifs

— Découvrir et utiliser l'interface interactive **Live Editor**
— Se rappeler des objets manipulés sous Matlab
— Segmenter les mots d'une phrase puis les réorganiser pour générer une autre phrase
— Reconstruire un signal créneau à partir de sa décomposition en série de Fourier

Téléchargez le fichier TD1.zip sur le site web du cours.

1.1 Découverte de Live Editor

Depuis 2016, Matlab intègre une interface interactive, baptisée **Live Editor** qui permet d'écrire, d'exécuter et de modifier un programme Matlab (voir Figure 1.1). Les fichiers créés avec cet éditeur interactif sont sauvegardés avec l'extension .mlx.

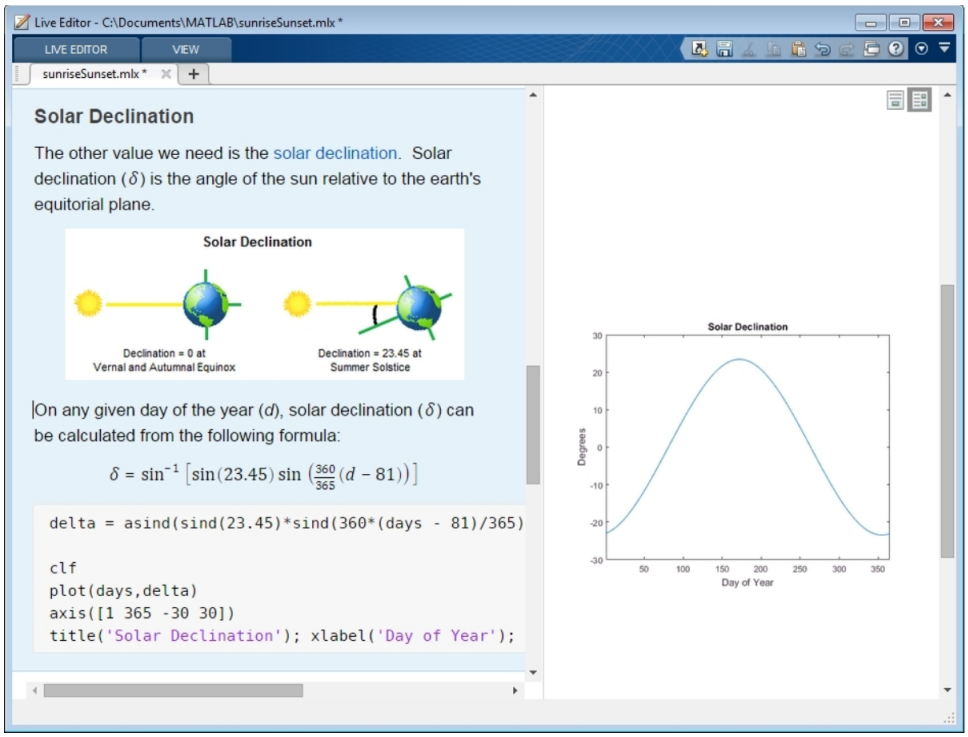

Figure 1.1 – L'interface interactive **Live Editor** de Matlab

Ce nouvel éditeur interactif facilite l'analyse et l'exploration d'un programme en offrant la possibilité d'y intégrer du texte, des hyperliens, des équations mathématiques à l'aide de com-

mandes latex, des schémas ou figures.... A partir du fichier développé, il est ensuite facile de convertir le document en .pdf ou .html, lisible et partageable pour d'autres.
Nous allons découvrir et exploiter cette interface interactive qui permet donc de faire fonctionner Matlab «en ligne» lors des séances de TD.
Visionnez la version courte du webinar de présentation de cette interface interactive disponible à partir du lien suivant :

- `https://fr.mathworks.com/videos/using-the-live-editor-117940.html` (version courte)
- `https://fr.mathworks.com/videos/introducing-the-matlab-live-editor-119100.html` (version longue)

1.2 Rappels : objets manipulés sous MATLAB

1.2.1 Matrice, vecteur, scalaire

Sous Matlab, on travaille essentiellement avec un seul type d'objet : une **matrice** qui peut contenir des valeurs entières, réelles ou complexes.
On peut retenir que dans MATLAB tout est **matrice**.

Lancez MATLAB. Placez-vous sous votre répertoire de travail. Créez un répertoire *TdS* puis placez-vous sous ce répertoire.
Saisissez les différents exemples proposés dans la suite pour bien comprendre la manipulation des vecteurs et matrices sous Matlab.
Attention si vous copiez les commandes depuis le fichier pdf de l'énoncé de TD, il faudra parfois retaper certains caractères (apostrophe en particulier) qui seront soulignés en rouge dans le fichier .mlx.

Sauvegardez l'ensemble des commandes dans un programme dénommé td1_1.mlx qui débutera ainsi :

Manipulation d'objets sous MATLAB

Fichier td1_1.mlx
Votre prénom nom
Date du jour

```
clear
clc
close all
```

1.2.2 Construction d'un vecteur

On peut définir une liste de nombres en donnant à la suite ses éléments séparés par des espaces ou des virgules. Un vecteur est délimité par des crochets.
Exemple

```
a=[2,4,6,8]
```
ou bien
```
a=[2 4 6 8]
a =
2 4 6 8
```
a est considéré comme un vecteur (ligne) comportant une ligne et quatre colonnes : 2 4 6 8

Accès aux éléments d'un vecteur

On peut isoler des éléments d'une liste. On accède à l'élément d'indice i de la liste a avec $a(i)$.
Exemple :
```
a(1)
ans =
2
```
On accède au premier élément du vecteur `a`.
Important : Il faut bien noter que les indices d'un vecteur commence à 1 sous Matlab et non 0.
```
a(0)
??? Subscript indices must either be real positive integers or logicals.
```
La demande d'accès à un élément d'indice nul ou négatif conduira donc toujours à une erreur !
```
a(1:3)
ans =
2 4 6
```
On récupère les trois premiers éléments de a.
```
a(1:2:4)
ans =
2 6
```
On récupère les deux premiers éléments aux indices impairs 1 et 3.
Rappels :

- L'expression `x=x0 :pas :xf` crée un vecteur dont les éléments vont de `x0` à `xf` avec un incrément de `pas` ;
- Lorsqu'on ne donne pas le pas, la valeur du pas est par défaut 1.

Exemple :
```
a(1:4)
a =
2 4 6 8
```

1.2.3 Construction d'une matrice

On peut créer des matrices de nombres avec plusieurs lignes en donnant chaque ligne séparée par un point-virgule " ;" .
Exemple :
```
X=[1 2 3 4;5 6 7 8]
X =
1 2 3 4
```

```
5 6 7 8
```

Accès aux éléments d'une matrice

De même, on peut extraire des parties d'une matrice. Le premier nombre désigne la ligne et le deuxième nombre la colonne où se trouve l'élément :
Exemple :

```
 X(1,2)
ans =
2 (élément à la ligne 1 et à la colonne 2)
 X(2,1)
ans =
5 (élément à la ligne 2 et à la colonne 1)

X(:,1)
ans =
1
5 (tous les éléments de la colonne 1)

 X(1,:)
ans =
1 2 3 4 (tous les éléments de la ligne 1)

 X(1,2:4)
ans =
2 3 4
```

Concaténation de matrices

On peut mettre bout à bout plusieurs matrices. Il faut vérifier que la matrice ajoutée a le même nombre de lignes pour une concaténation en colonnes, où le même nombre de colonnes pour une concaténation en ligne.
Exemple :

```
b=[10 12 14 16] ;
c=[a  b]
ans =
2 4 6 8 10 12 14 16  (mise bout à bout sur la même ligne des 2 vecteurs)

Y=[9 10 11 12]
[X ; Y]
ans =
1 2 3 4
5 6 7 8
9 10 11 12 (mise l'une en dessous de l'autre des 2 matrices)
```

Essayez à présent la commande suivante. Que se passe-t-il ? Expliquez.

```
[X  Y]
```

Ce type d'erreur est classique lorsqu'on débute sous Matlab. Pour obtenir la taille d'une matrice ou d'une liste, il est utile d'exploiter les commandes `size` ou `length` présentées ci-dessous.

Dimension d'un vecteur ou d'une matrice

Pour accéder au nombre d'éléments contenus dans une liste et donc à sa longueur, on utilise la commande `length`. Pour accéder au nombre de lignes et de colonnes d'une matrice, on utilise la commande `size`. Exemple :

```
length(a)
ans =
4 (nombre d'éléments de a)
size(X)
ans =
2 4
size(X,1)
ans =
2 (nombre de lignes de X)
size(X,2)
ans =
4 (nombre de colonnes de X)
```

Opération sur les matrices

Toutes les opérations usuelles sur les matrices sont disponibles (addition, multiplication, transposition etc.). Matlab permet aussi d'appliquer les principaux opérateurs éléments par éléments, il faut alors faire précéder l'opérateur voulu par un point « . .
Créez deux vecteurs :
`x=[1 2 3];`
`y=[4 5 6];`
Observez ce que donnent les opérations suivantes :
`x-y`
`x*y`
`x.*y`
`x^2`
`x.^2`
`x./y`

Modifiez à présent `x=[0 1 2 3]`.
`x` et `y` n'ont alors plus la même taille. Que se passe-t-il au niveau des opérations ci-dessus ? Conclusion ?

1.3 Jeux de mots

1.3.1 Enregistrement d'une courte phrase à l'aide de votre smartphone

Si vous n'avez pas de smartphone, vous pouvez passer à la section suivante sinon enregistrez-vous à l'aide de votre téléphone en prononçant lentement l'une des deux phrases suivantes :

— « Rien ne sert de courir, il faut partir à point.

— «J'entends, j'oublie. Je vois, je me souviens. Je fais, je comprends.

Une fois enregistré au format .wav ou mp3, envoyez votre fichier sur votre boîte mail et sauvegardez-le sur votre répertoire de travail. Chargez alors votre phrase dans MATLAB à l'aide de la commande suivante (à adapter si votre fichier est au format wav.) :

```
[x,fs] = audioread('maphrase.mp3');
t = (0:length(x)-1)' * 1/fs;
```

Les vecteurs colonne `x` et `t` contenant les échantillons de voix et les instants d'enregistrement. La variable `fs` contient la valeur de la fréquence d'échantillonnage de la phrase.

1.3.2 Visualisation du signal enregistré

Si vous n'avez pas de smartphone, téléchargez le fichier RienNeSertDeCourir.mat sur le site web du cours. Chargez la phrase dans MATLAB en saisissant la commande

```
load RienNeSertDeCourir;\\
fs=22050;
```

Dans ce fichier, se trouvent les vecteurs colonne `x` et `t` contenant les échantillons de voix et les instants d'enregistrement.

Tracez le signal enregistré en fonction du temps.

```
plot(t,x),
shg
```

La commande `shg` pour SHow Graphic permet de mettre au premier plan la figure après une commande de tracé de type plot, stem, ...

1.3.3 Ecoute du signal enregistré

Ecoutez la phrase enregistrée en saisissant la commande :

```
sound(x, fs)
```

Cette commande permet d'écouter la phrase à sa fréquence d'échantillonnage d'enregistrement. Ecoutez la phrase en modifiant la fréquence d'échantillonnage à double ou deux fois plus petite pour me/vous faire parler comme comme « Terminator » ou « Donald Duck ».
Modifier la fréquence d'échantillonnage revient à appliquer un changement d'échelle $y(t) =$

$x(at)$ au signal initial. En fonction de la valeur du facteur d'échelle, cela revient à opérer une compression ou une dilatation du spectre initial d'où la version plus grave ou plus aigüe du signal écouté comme illustré sur la figure 1.2 dans le cas d'une fenêtre rectangulaire.

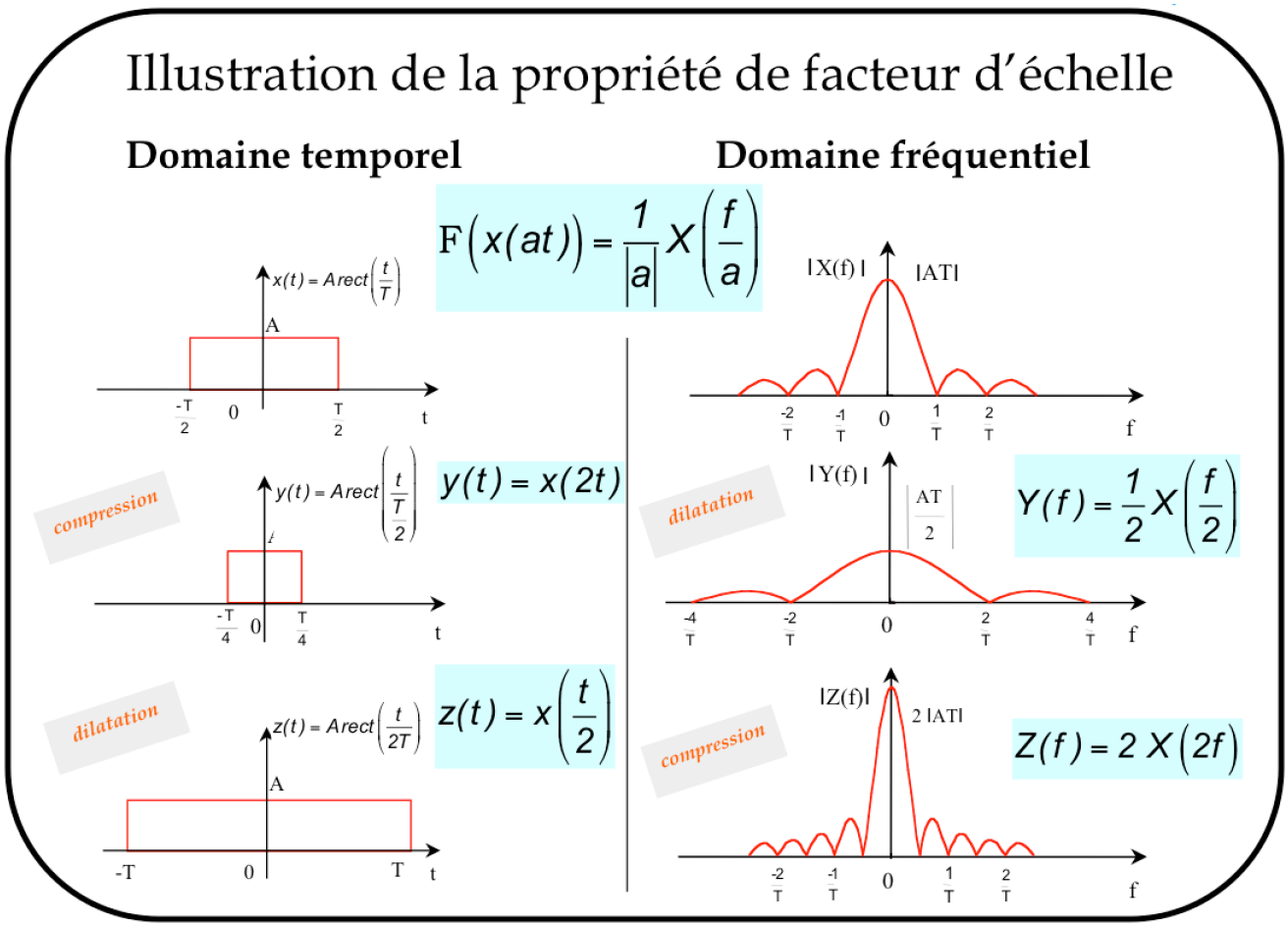

Figure 1.2 – Illustration de l'effet d'un changement d'échelle dans le domaine fréquentiel

Déterminez pour les deux nouvelles fréquences d'échantillonnage la valeur du facteur d'échelle a.

1.3.4 Segmentation de la phrase enregistrée

Tracez le signal en fonction des indices du vecteur x

```
plot(x),shg
```

Essayez de repérer les indices de début et de fin du début de la phrase «Rien ne sert de ».
Pour segmenter le premier mot, il faut par exemple créer un vecteur `riennesertde` contenant les 38000 premières valeurs du signal enregistré `x`.

```
riennesertde = x(1:38000);
```

N'oubliez pas de mettre un « ; » à la fin de la ligne ci-dessus pour exécuter la commande sans afficher le résultat dans la fenêtre commande de Matlab.
Pour écouter le mot segmenté :

```
sound(riennesertde, 22050);
```

Segmentez la phrase initiale en créant les variables suivantes : `riennesertde`, `courir`, `ilfaut`, `partirapoint`. Certaines parties de la phrase peuvent être plus difficiles que d'autres à segmenter ...
Sauvegardez l'ensemble de vos commandes dans un programme dénommé td1_2.mlx qui débutera ainsi :

Jeux de mots
Fichier td1_2.mlx
Votre prénom nom
Date du jour

```
clear
clc
close all

[x,fs] = audioread('maphrase.mp3');
t= (0:length(x)-1)' * 1/fs);
%load RienNeSertDeCourir;
riennesertde = x(1:...);
courir = x(...:...);
etc
```

1.3.5 Synthèse d'une nouvelle phrase

Réorganisez les mots pour générer la phrase suivante : « Rien ne sert de partir à point, il faut courir ! »
Notez que le signal initial de parole est un vecteur colonne contenant un certain nombre de valeurs (length(x)).
Pour produire un signal réarrangé, vous devez construire un autre vecteur colonne.
Pour écouter «Rien ne sert de courir», il faut mettre bout à bout les 2 vecteurs verticaux contenant le début de la phrase à l'aide de la commande :

```
sound([riennesertde ; courir], fs);
```

Faites valider votre programme `td1.mlx` et la nouvelle phrase synthétisée par l'enseignant.

1.4 Reconstitution d'un signal créneau à partir de sa décomposition en série de Fourier

Soit un signal créneau $s(t)$ de période T_o d'amplitude A représenté sur la figure 1.3.
La décomposition en série de Fourier de ce signal créneau ne comprend que des harmoniques de rang impair et s'écrit sous la forme :

$$s(t) = \sum_{k=0}^{+\infty} \frac{4A}{(2k+1)\pi} \sin\left((2k+1)2\pi f_o t\right) = \frac{4A}{\pi}\sin(2\pi f_o t) + \frac{4A}{3\pi}\sin(2\pi \times 3 f_o t) + \dots$$

Ecrire un programme Matlab creneau.mlx sous Live Editor qui permet de reconstruire un signal créneau (pour $A = 2$ et $T_o = 2$s) sur 3 périodes à partir de sa décomposition en

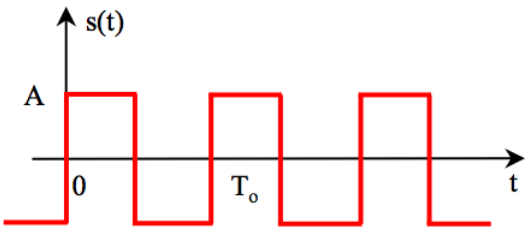

Figure 1.3 – Signal créneau

série de Fourier et de tracer sur une même figure le signal créneau original et le signal reconstruit à partir de n harmoniques. Tracer également les harmoniques successifs sur 3 périodes.

Rappels et conseils :

— Un signal sinusoïdal non retardé est défini par :

$$s(t) = A \sin(2\pi f_o t)$$

Pour le créer sous Matlab, il faudra choisir avec attention la période d'échantillonnage T_e (pas de discrétisation du vecteur temps `t`).

— Tracez dans un premier temps le signal créneau et le signal reconstruit à partir de l'harmonique fondamental puis des 3 premiers harmoniques, avant de considérer le cas général à l'aide d'une boucle `for ... end`. Vous pouvez vous inspirer des commandes ci-dessous :

```
A=2;
To=2 ;
fo=1/To;
fe=100*fo; % Théorème de Shannon fe > 2*fmax
Te=1/fe;
t=0 :Te :3*To ;
cren=A*square(2*pi*fo*t); % Signal creneau
h1=4*A/pi*sin(2*pi*fo*t); % harmonique de rang 1
plot(t,cren,t,h1)
h3=...
plot(t,cren,t,h1+h3)
```

— Jusqu'à quel harmonique, la fréquence d'échantillonnage vérifie-t-elle le théorème de Shannon. Justifier.

Faire valider votre programme creneau.mlx par l'enseignant.