

## SIGNAL DETERMINISTE ET ANALYSE DE FOURIER

### TDAO 1 : Introduction à Scilab

---

## Introduction

Le but de ce TDAO est de faciliter l'accès aux techniques numériques du traitement du signal en faisant liaison entre la théorie et la pratique et en mettant à la portée des étudiants des résultats utiles dans ce domaine. Le logiciel utilisé pour les travaux pratiques est Scilab, qui a été conçu pour le calcul matriciel. Scilab est un logiciel gratuit et très intéressant pour développer des calculs numériques. Il intègre un environnement graphique et ses atouts principaux sont :

- La surprenante simplicité d'utilisation provenant de la vectorisation,
- La puissance de calcul,
- La précision des résultats.

Scilab est enrichi par ses bibliothèques spécialisées appelées boîtes à outils (*toolboxes*) qui seraient automatiquement chargées au démarrage du logiciel et qui évitent, dans de nombreux cas, de programmer les techniques numériques les plus courantes. Dans cette séance et dans les séances qui suivent, la boîte à outils (*toolbox*) la plus utilisée est « *Signal Processing* ». Cependant pour les développements spécifiques, il offre les mêmes possibilités de programmation structurée que les langages scientifiques courants (C, FORTRAN...).

### Installation:

Scilab est un logiciel gratuit disponible sous Windows, Linux et MacOSX, il est téléchargeable à l'adresse suivante: <http://www.scilab.org/>

### Conventions:

- Les principales fonctions Scilab utilisées sont écrites en italique.
- Les commandes et les instructions Scilab sont écrites en italique.

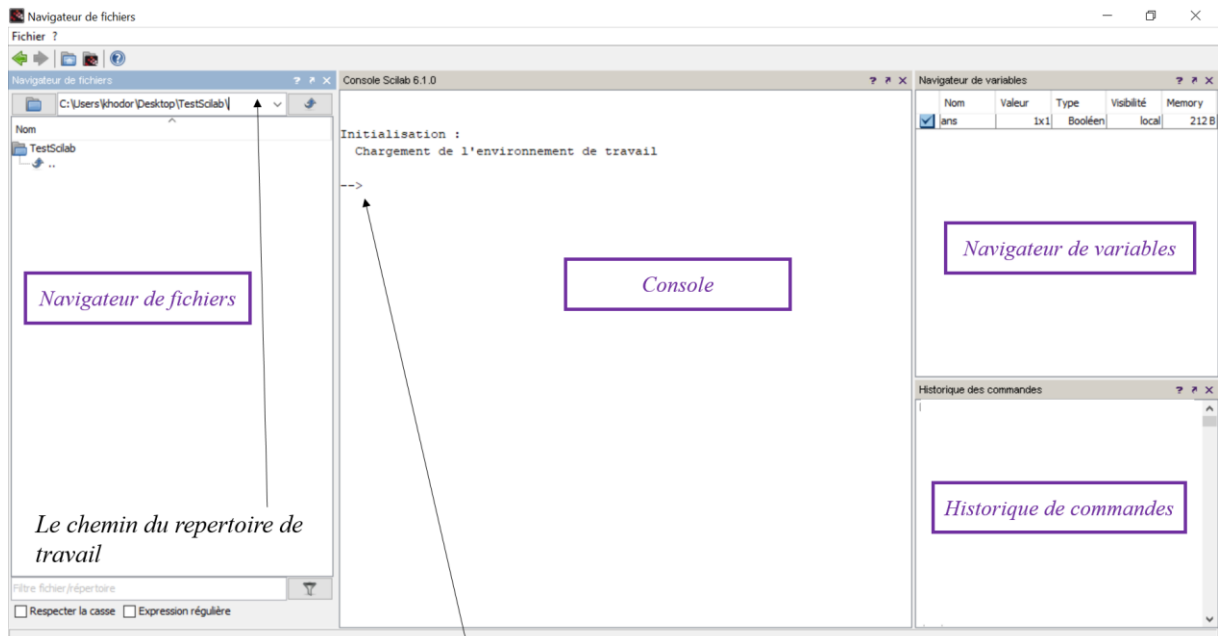
## Objectif du TDAO

L'objectif de TDAO1 est de se familiariser avec les bases de Scilab. La première partie est un bref descriptif du logiciel et la deuxième partie se compose de plusieurs exercices.

**Tout d'abord, vous créer un fichier dans votre répertoire de travail où vous sauvegardez vos programmes Scilab.**

## 1. Bref descriptif de Scilab

Lorsque vous démarrez Scilab, différentes fenêtres apparaissent :



Entrez les fonctions SCILAB à la ligne de commande

- « *Console* » : la fenêtre de commande, c'est la fenêtre la plus importante est dans laquelle vous tapez des commandes Scilab et voyez les résultats de vos commandes. Attention : les commandes écrites dans cette fenêtre ne sont pas sauvegardées dans un fichier
- « *Historique des commandes* » : affiche la liste des commandes exécutés dans la console
- « *Navigateur de variables* » : affiche les variables déclarés dans console avec des informations sur chaque variable.

### 1.1 Manipulation des variables et des matrices

Scilab est un interpréteur de commandes dont l'élément de base est la matrice.

**1.1.1.** Taper dans la *console* les commandes suivantes :

```
--> x=1;  
--> b=14  
--> exp(1)  
--> x= [7 2 9]  
--> y= [7;2;9]
```

```

--> M=0:9 ;
--> M
--> mean(M)
--> N=[7 2 9 ; 4 3 1 ; 6 5 8]
--> N'
--> N(1,2)
--> log(y)
--> disp(M)
--> Pi=3.14 ;
--> z=sin(Pi)
--> % pi
--> abs(-4)
--> // c'est un commentaire

```

- Commenter les résultats de chaque commande et conclure sur l'utilité de « ; »
- Afficher le 4<sup>ème</sup> élément du vecteur *M*.
- Taper *N( :,2)* et commenter


### 1.1.2.

- Taper la commande *who\_user*, et commenter.
- Taper la commande *clc* et commenter.
- Taper *clear N* et commenter.
- Taper *clear* et commenter
- Taper *help min* et commenter

## 1.2 Scripts et fonctions

### 1.2.1 Scripts

Pour pouvoir réutiliser les lignes de commandes, il est utile de les mettre dans un **script**. C'est une suite de commandes qui se trouvent dans un fichier dont l'extension est '.sci' ou '.sce' que Scilab pourra lire et exécuter:

- Ouvrir l'éditeur de scripts de Scilab en cliquant sur l'icone  de la barre d'outils.
- Créer le script suivant :

```

a=2;
b=5;
S=0;
//for permet de changer la valeur d'une variable de maniere reguliere
for k=1:b
    S=S+k^a;
end //fin de la boucle for
disp(S)

```

- Enregistrer le fichier sous le nom : *Testscript* . Le fichier portera l'extension *.sce*
- Exécuter *Testscript.sce* en tapant *exec Testscript.sce* dans la console, ou en cliquant sur le bouton « enregistrer et exécuter » dans la barre d'outil du script.
- Expliquer le calcul effectué par le script.

### 1.2.2. Fonctions

- Pour créer une fonction :
  - 1- Créer un nouveau fichier de script (il faut un fichier séparé pour chacune des fonctions que l'on définit)
  - 2- Définir la fonction dans la première ligne comme la syntaxe suivant:

`function var=nomdelafunction (arg1,arg2,...)`

La définition commence donc par le nom *function*,

*var* : nom de la variable de sortie

*nomdelafunction* : le nom de la fonction

*arg1,arg2,...* : les variables d'entrée

- 3- Ecrire le corps de la fonction (terminer par *end*)
- 4- Sauvegarder le fichier. Le fichier doit impérativement porter **le même nom que fonction** (suivi de l'extension *.sce*)

Une fois le fichier sauvegardé, il est possible d'utiliser la fonction après son exécution avec la commande *exec*. On peut ensuite appeler la fonction directement dans la console en tapant son nom et en précisant les arguments d'entrée et de sortie.

- Réaliser la fonction suivante :

```

//fonction produit de deux variables au carré
function R=ProduitCarre(x1,x2)
    R=(x1*x2)^2;
end

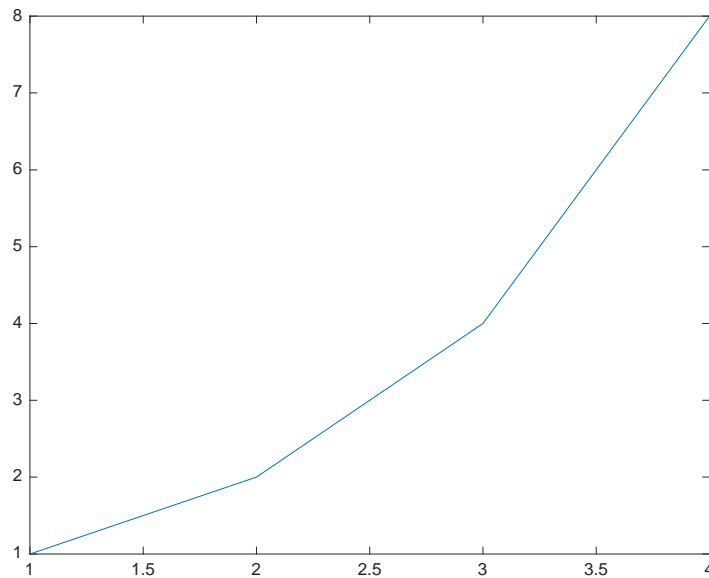
```

Exécuter la fonction pour  $x_1=100$  et  $x_2=40$ .

## 1.3 Représentation graphique

On peut utiliser Scilab pour représenter graphiquement des fonctions. Les courbes créées sont à base de vecteur. La commande de base qui permet de dessiner des graphes est *plot*

Exemple: *plot ([1, 2, 4, 8])*



La commande *help plot* donne plus de détails, en particulier la liste des codes permettant de choisir la couleur du tracé, la forme des marqueurs des points (croix, cercle, ...) et le type de trace (pointillé, plein, tirets, etc.)

On peut également (et c'est ce que l'on va utiliser le plus souvent par la suite) représenter les valeurs d'un vecteur *y* en fonction d'un vecteur *x* :

```
--> x = 0:0.1:1
--> y = x.^2 // carré élément par élément
--> y1=x.^3 ; // cube élément par élément
--> plot(x, y)
--> figure ; //pour ouvrir une nouvelle figure
--> plot(x,y,'o')
--> set(gca(),"auto_clear","off")//pour dessiner sur la même figure
>> plot(x,y1,'r')
```

## 2. Exercices

### Exercice 1: Signal sinusoïdal

- Définir un vecteur *t* contenant les valeurs entre 0 et 10 espacées de 0.001 (ce sera le vecteur du temps en fonction duquel on crée et on trace notre signal). La durée du signal est donc de 10 sec avec un pas de 0.001 :

$$t=0:0.001:10$$

- b) Un signal sinusoïdal est caractérisé par sa fréquence ( $f_0=1/\text{la période}$ ) et son amplitude  $A$ :

$$y=A \sin(2\pi f_0 t)$$

Générer un signal sinusoïdal d'amplitude  $A=2$  et de fréquence  $f_0=2$  Hz.

- c) Tracer le vecteur signal  $y$  en fonction du vecteur temps  $t$  en utilisant *plot*.  
d) Les fonctions *ylabel* et *xlabel* : permettent d'étiqueter les axes. La fonction *title* permet de donner un titre à la figure. Utiliser *ylabel*, *xlabel* et *title* pour nommer les axes et donner un titre à votre figure. (Regarder l'aide de ces fonctions pour la syntaxe).

## Exercice 2: Signal sinusoïdal en utilisant une fonction

- a) En vous inspirant de l'exercice ci-dessus, créer une fonction *dessiner\_sinus.sce* qui permet de générer et dessiner un signal sinusoïdal en déterminant le pas, la fréquence, l'amplitude et la durée du signal comme arguments d'entrées.  
b) Exécuter la fonction afin de tracer un signal sinusoïdal avec  $A=3$ , durée=4s, pas=0.001 et  $f_0=2$ Hz. Vérifier graphiquement la période du signal.  
c) Ajouter une nouvelle figure en utilisant la fonction *figure* et tracer un autre signal sinusoïdal  $y_1$  avec  $A=1$ , durée=5s, pas=0.0001 et  $f_0=10$ Hz. Vérifier graphiquement la période du signal. Comparer la fréquence et la période de ces deux signaux.  
d) Déterminer dans la fenêtre de commande, le maximum, le minimum, la moyenne, la médiane et la dispersion de  $y_1$ .