# PHYSIQUE, ELECTRONIQUE ET SYSTEMES

FISA - Semestre 5

2024-2025

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.

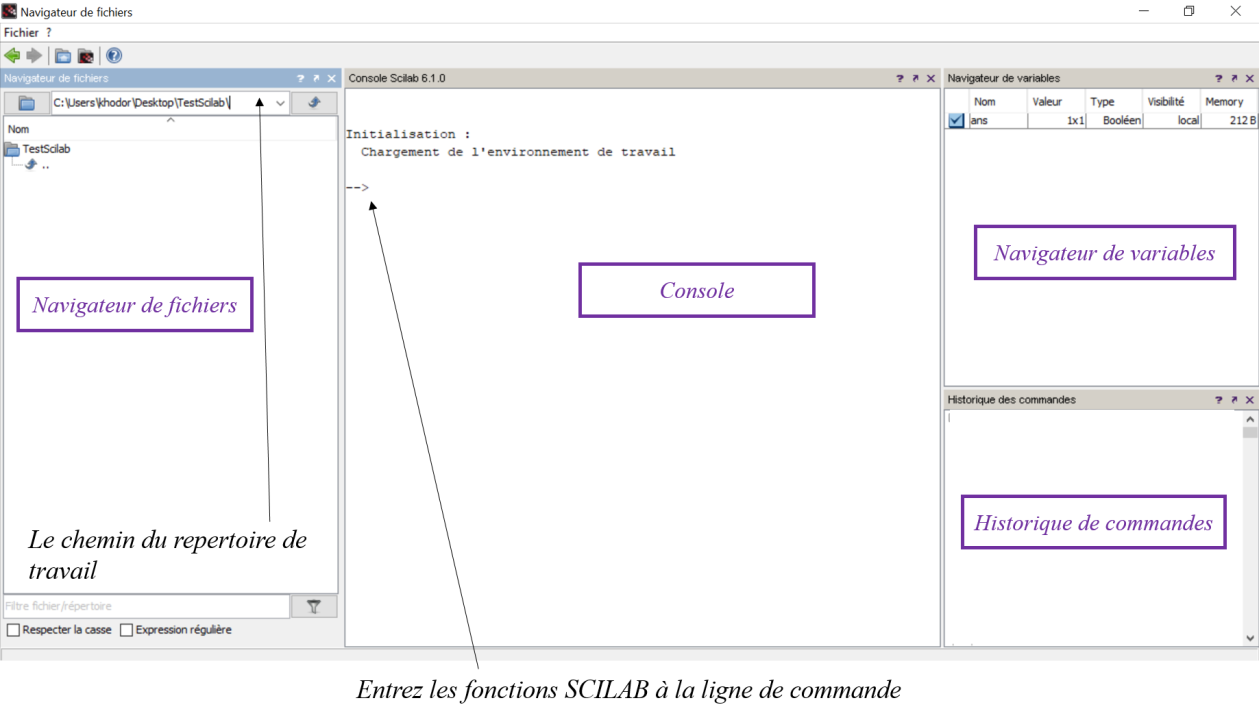
# Objectif du TDAO

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.**

# Bref descriptif de Scilab

Lorsque vous démarrez Scilab, différentes fenêtres apparaissent :



* *« 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.

## 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;

Créer une variable pour nom « x » avec pour valeur « 1 » de type réel.

--> b=14

Créer une variable pour nom « b » avec pour valeur « 14 » de type réel.

--> exp(1)

Donne la valeur de l’exponentielle de 1 : « 2.7182818 ».

--> x= [7 2 9]

Créer un tableau de taille 1x3 (1 ligne x 3 colonnes) avec pour valeurs « 7. 2. 9. »

--> y= [7;2;9]

Créer un tableau de taille 3x1 (3 lignes x 1 colonne) avec pour valeurs «  »

--> M=0:9 ;

Créer un tableau de taille 1x10 (1 ligne x 10 colonnes) avec pour valeurs 0 à 9, 1 unité par 1 unité « 0. 1. 2. 3. 4. 5. 6. 7. 8. 9. »

--> M

Affiche la variable (ici un tableau) M.

-->mean(M)

La commande « mean » affiche la moyenne d’une suite de nombre, ici « 4.5 »

--> N=[7 2 9 ; 4 3 1 ; 6 5 8]

Créer un tableau de taille 3x3 (3 lignes x 3 colonnes) «   »

--> N’

Inversion des lignes – colonnes avec la matrice originale N. ATTENTION N’ devient N, N’ n’est pas une nouvelle variable mais une modification de N.

--> N(1,2)

Accède à la valeur située à la colonne 1, à la ligne 2. Ici la valeur « 2 »

--> log(y)

Calcule chaque logarithme des valeurs inclues dans la variable « y ».

--> disp(M)

Affiche le contenu de la variable « M ».

--> Pi=3.14 ;

Créer une variable pour nom « Pi » avec pour valeur « 3.14 » de type réel.

--> z=sin(Pi)

Retourne la valeur sinus de Pi « 0.0015927 »

-->% pi

Retourne la valeur exacte de Pi « 3.1415927 ».

--> abs(-4)

Abs retourne la valeur absolue du nombre, ici « 4 »

--> // c’est un commentaire

Inclus un commentaire mais n’instaure pas de variable.

* Commenter les résultats de chaque commande et conclure sur l’utilité de « ; »

« ; » peut servir à séparer plusieurs commandes sur une seule ligne

* Afficher le 4ème élément du vecteur *M*.

  Le 4ème élément est « 2 ». On met donc la commande « N(1,2) pour l’afficher.

* Taper *N( :,2)* et commenter

Cette commande permet d’afficher la deuxième colonne de la matrice M.

**1.1.2.** Taper la commande *who\_user*, et commenter.

« who\_user » permet d’afficher les variables instaurées.

* Taper la commande *clc* et commenter.

« clc » permet de supprimer l’historique de la console.

* Taper *clear N* etcommenter.

Supprime la variable nommée « N ».

- Taper *clear* et commenter

« Clear » supprime toutes les variables instaurées.

- Taper *help min et* commenter

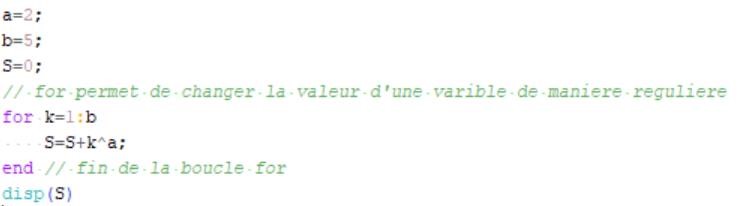
Affiche le guide du logiciel.

## 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 :



* 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.

Initialise 5 variables réelles : a=2 ; b=5 ; S=0 ;

Début de la boucle for : pour k allant de 1 jusqu’à b (5)

S = ValeurdeS + kvaleurdea ⬄ S = ValeurdeS + k2

Soit :

(k=1)

S = 0 + 52 ⬄ 0 + 25 = 25

(k=2)

S = 25 + 52 ⬄ 25 + 25 = 50

(k=3)

S = 50 + 52 ⬄ 50 + 25 = 75

(k=4)

S = 75 + 52 ⬄ 75 + 25 = 100

(k=5)

S = 100 + 52 ⬄ 100 + 25 = 125

D’où Scilab qui nous retourne la valeur 125.

**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*

=

*nom*

*dela*

*fonction*

(

*arg1*

,

*arg2*

*,…*

)

La définition commence donc par le nom

*function*

,

*var*

:

nom de la variable de sortie

*nomdela*

*fonction*

:

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’extensio

n

.

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

:

Exécuter

la fonction pour x

1

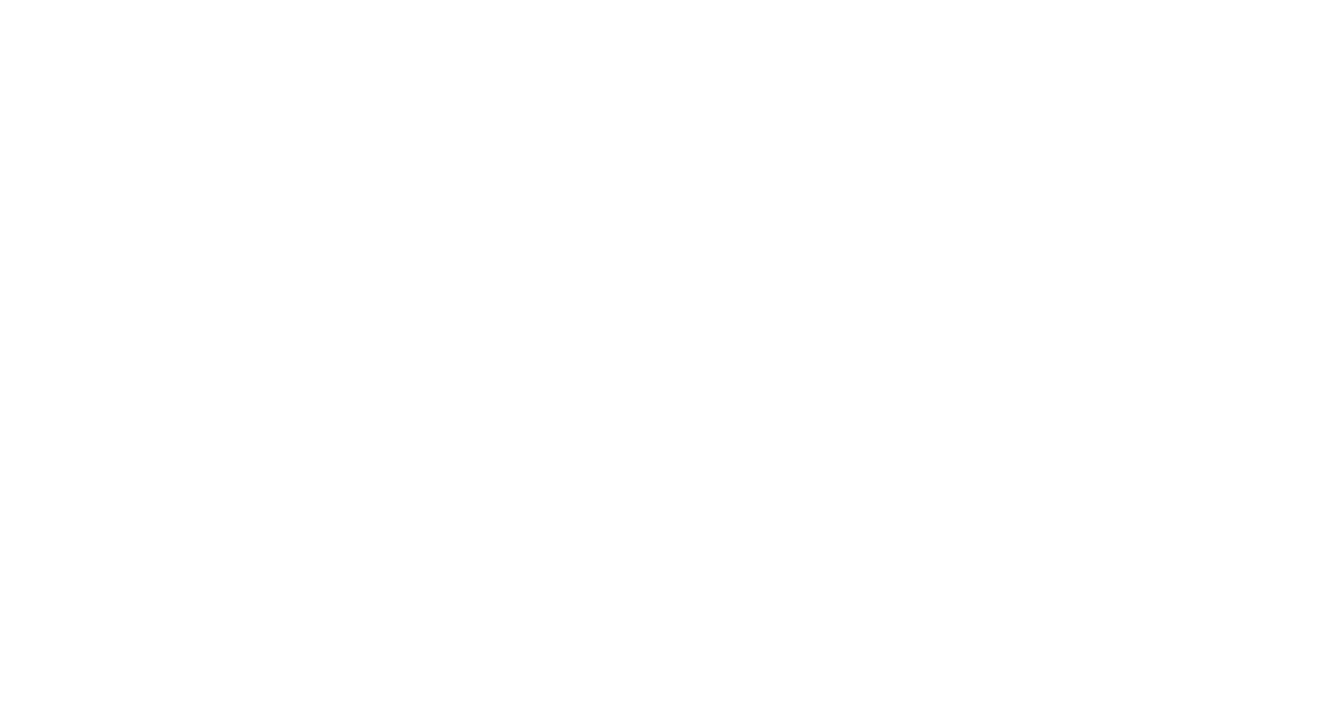
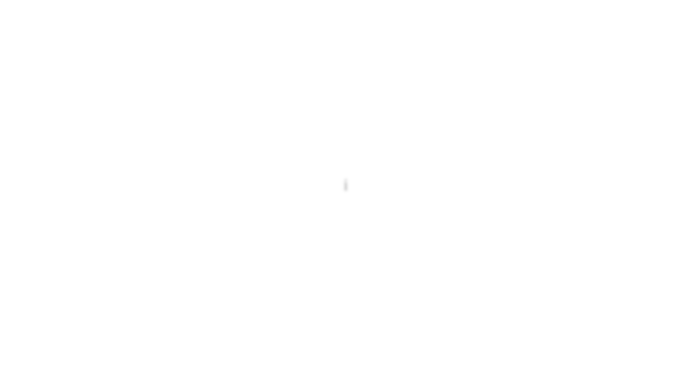
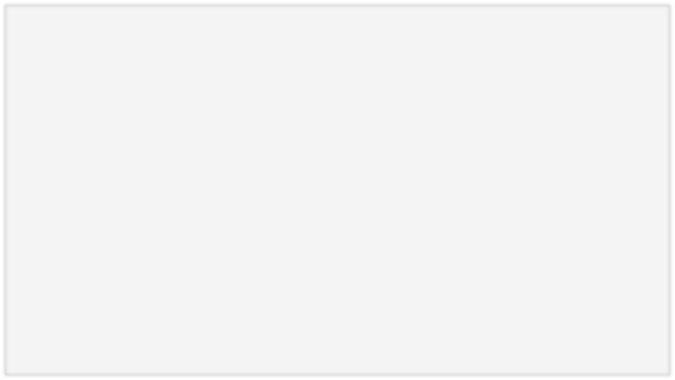
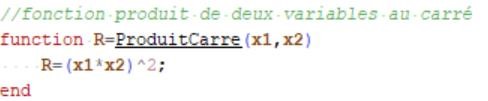
et x

=100

2

=40

.



i

*// fonction produit de deux variables au carré*

x1=100;

x2=40;

function **R**=ProduitCarre(**x1**, **x2**)

**R**=(**x1**\***x2**)^2;

endfunction

*// appel de la fonction*

R=ProduitCarre(x1,x2);

disp(R);

## 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])*

1

1.5

2

2.5

3

3.5

4

1

2

3

4

5

6

7

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’)*

# Exercices

## Exercice 1: Signal sinusoïdal

a) 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 (*f0*=1/la période) et son amplitude *A*:

### y=A sin(2πf0t)

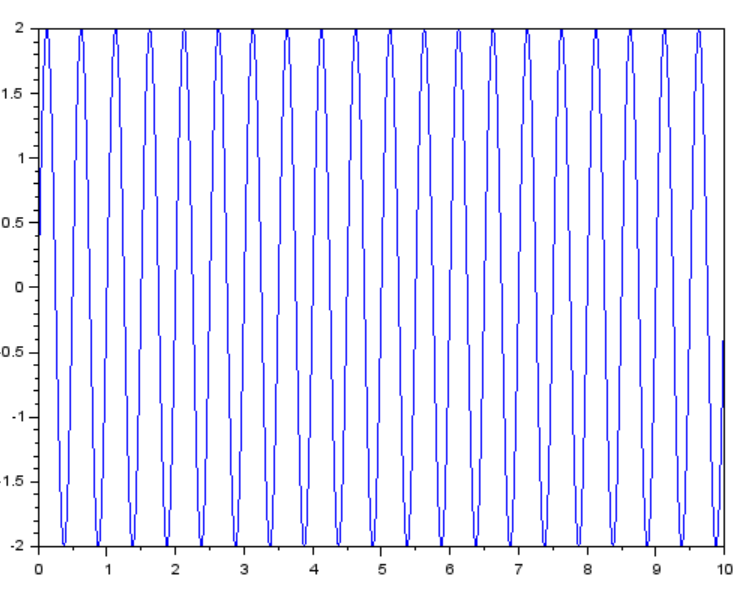
Générer un signal sinusoïdal d’amplitude *A*=2 et de fréquence *f0* =2 Hz.

On définit la variable de l’amplitude : A = 2 ;

On définit la variable de fréquence : f0 = 2 ;

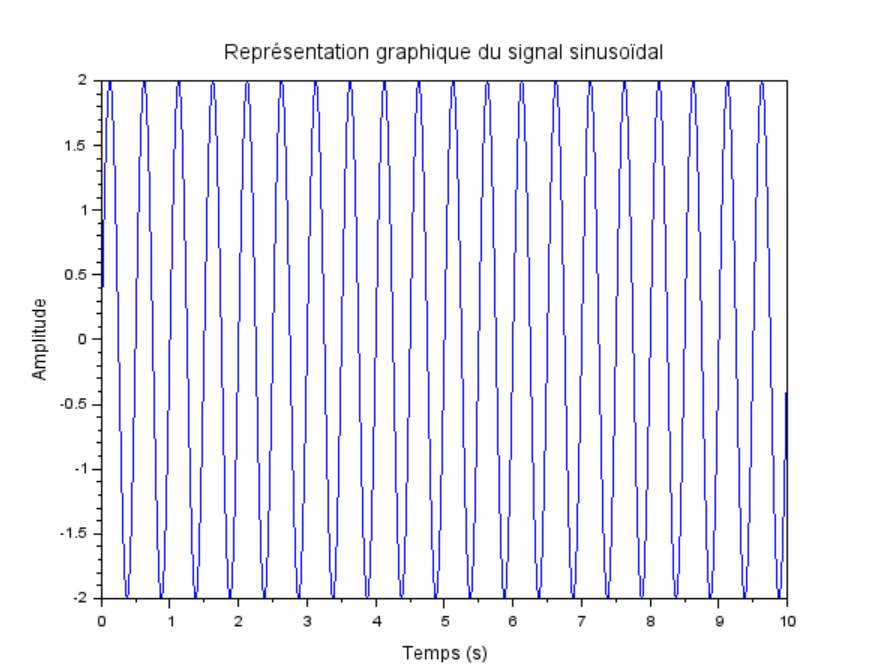
On effectue : y = A \* sin(2 \* %pi \* f0 \* t) ;

1. Tracer le vecteur signal *y* en fonction du vecteur temps *t* en utilisant *plot*.



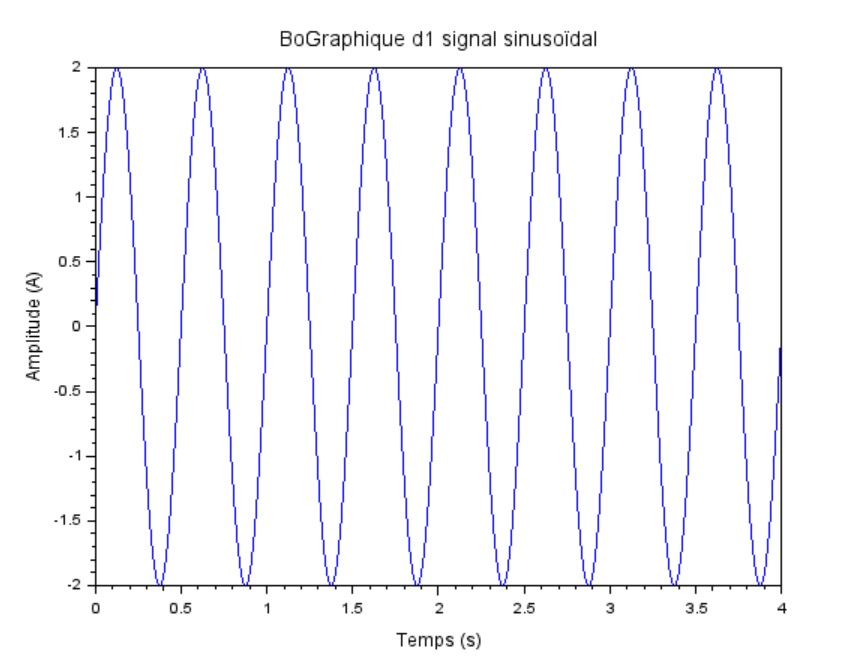
Affichage avec « plot (t, y) ; »

1. 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

1. 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.
2. Exécuter la fonction afin de tracer un signal sinusoïdal avec *A*=3, durée=4s, pas=0.001 et *f0*=2Hz. Vérifier graphiquement la période du signal.

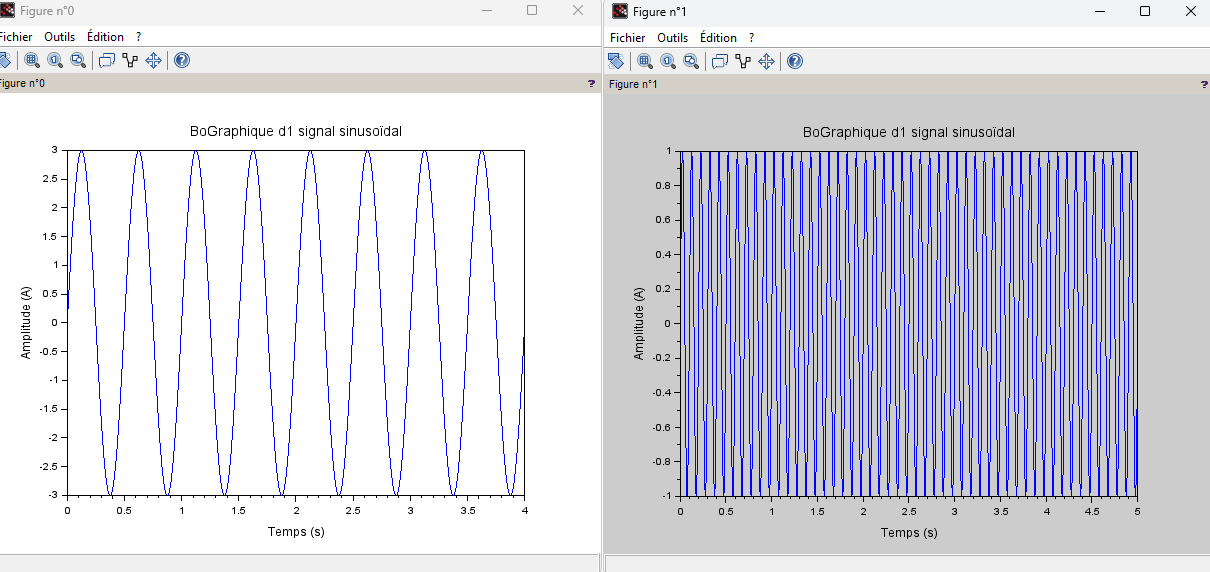


On sait que :

Avec , on calcule : secondes

On confirme via le graphique que le signal sinusoïdal est périodique avec un cycle de 0.5 secondes.

1. Ajouter une nouvelle figure en utilisant la fonction *figure* et tracer un autre signal sinusoïdal *y1* avec *A*=1, durée=5s, pas=0.0001 et *f0*=10Hz. Vérifier graphiquement la période du signal. Comparer la fréquence et la période de ces deux signaux.



On peut voir sur les représentations graphiques les différences suivantes :

* Le signal 1 a une durée de 4 secondes contre 5 secondes pour le signal 2
* Le signal 1 a une fréquence plus basse que le signal 2
* Le signal 1 a une amplitude de 3 contre 1 pour le signal 2
* Le signal 1 a une période de 0.5 contre 0.1 pour le signal 2

1. Déterminer dans la fenêtre de commande, le maximum, le minimum, la moyenne, la médiane et la dispersion de 𝑦1.