# MT12 - TP1 : familiarisation avec Scilab

---

Ce document part de bases très simples pour aller vers des subtilités de Scilab afin que vous soyez préparés pour les TPs suivants.

---

L'objectif de ce TP est de sa familiariser avec Scilab, outil que nous utiliserons au cours de ce semestre.

## 1 Elements de base de Scilab : Variables, fonctions, boucles

1. Scilab est un programme informatique permettant d'effectuer la plupart des opérations mathématiques classiques. Le code suivant instancie un réel $a$, crée la variable $b$ qui vaut $a - 3$. Affichez les valeurs de $a$ et de $b$. Scilab défini un certain nombre de constantes, introduites par %. Ainsi %pi correspond à une approximation de $\pi$, %T correspond à la valeur logique "vrai" (true) et %F correspond à "faux" (false). Vous pouvez retrouver toutes les variables actuellement définies dans le navigateur de variables en haut à droite.

```
--> a = 1
a  =

 1.

--> b = a - 3
b  =

 -2.

--> b
b  =

 -2.

--> a
a  =

 1.
```

2. Un certain nombre de fonctions sont déjà implémentées dans Scilab, par exemple, la fonction valeur absolue est notée abs. Pour appliquer la fonction valeur absolue à a, il suffit de l'appeler en mettant entre parenthèse la variable sur laquelle on veut l'appliquer. Calculez la valeur absolue de a, la valeur absolue de b. Vous pouvez appliquer la plupart des fonctions habituelles à un nombre : sqrt, exp, abs.

*Attention* : si vous définissez une variable ou une fonction avec le même nom qu'une fonction interne, vous masquez la fonction interne. La commande clear() permet de supprimer toutes les définitions de variables et de fonctions.

```
--> abs(a)
ans  =

1.

--> abs(b)
ans  =

2.
```

3. Scilab propose un ensemble d'instructions conditionnelles. La syntaxe est la suivante (pour l'inscrire dans votre console, tapez entrée après then, vous verrez que l'interpréteur comprend que vous n'avez pas terminé) :

```
--> if a < b then
> disp("true")
> else disp("false")
> end
```

Ce que dit l'instruction précédente, c'est que si a est inférieur à b, alors elle affiche vrai, sinon elle affiche faux. On peut bien entendu mettre n'importe quelle proposition à la place de $a < b$, exécuter n'importe quel nombre de lignes de code à la place de disp("true") et n'importe quel nombre de lignes de code à la place de disp("false").

4. Cet ensemble de commandes est déjà fastidieux à écrire, peut-être même avez-vous dû corriger une erreur de recopiage et cela était encore plus ennuyeux. Pour cela, ouvrez sci-notes à l'aide du premier icône dans la barre des icônes.

```
clear

// Initialisation des variables
a = 1
b = -2

// Debut du script
if a < b then
    disp("true")
else
    disp("false")
end
```

Vous aurez noté la fonction *clear()* au début qui permet de restaurer l'espace de travail. Cela permet d'assurer que nous n'allons pas utiliser des variables issues d'autres scripts au moment où on lance le notre. Si vous appuyez sur F5 ou l'icône de compilation, vous obtiendrez le résultat.

5. Proposez un script qui détermine le minimum de deux nombres à l'aide du script précédent.
6. Ce script étant utile, on souhaite pouvoir l'utiliser comme une fonction, c'est à dire en tapant dans la console Scilab une expression comme $my\_min(a, b)$, obtenir le minimum de deux nombres a et b. Le script suivant définit une fonction qui retourne la somme de deux nombres. Pour vous en convaincre,

exécutez-ce bout de code. Vous verrez qu'il ne se passe rien dans un premier temps, en revanche, à l'issu de son exécution, si vous tapez $my\_sum(a, b)$, le script devrait vous retourner le résultat attendu. Utilisez sa syntaxe et les questions précédentes pour produire une fonction qui retourne le minimum entre deux entiers.

```
function res = my_sum(a, b)
    res = a + b
endfunction

```

7. Le code ci-dessous crée une fonction qui renvoie la somme et la différence de deux nombres.

```
function [sum, diff] = operations(a, b)
    sum = a + b
    diff = a - b
endfunction

```

On pourra appeler la fonction de plusieurs manières

```
> operations(2, 4)
> 6 // ne renvoie que le premier retour
> [sum, difference] = operations(2, 4)
> sum =
>   6
> difference =
>   -2

```

Adaptez votre code afin que votre fonction min(a, b), renvoie un duet $(is\_min, min)$, avec $is\_min$ qui indique si a est plus petit que b, et min le minimum des deux nombres.

## 2 Fonctions mathématiques, affichage de graphiques, ensembles de départ et d'arrivée

Dans cet exercice, on souhaite afficher la fonction exponentielle, son DL à l'ordre 2 et l'erreur de cette même approximation (cf figure 2).
Pour effectuer l'affichage d'une fonction, il nous faut constituer deux vecteurs, un vecteurs contenant tous les antécédents, et un vecteur contenant toutes les images par la fonction exponentielle de ces antécédents.
Ainsi, si l'on veut afficher 100 points uniformément répartis entre $-1$ et 1, correspondant aux valeurs de la fonction exponentielle, il faut :

1. Générer un vecteur contenant 100 nombres uniformément répartis entre $-1$ et 1 et stocker le résultat dans un vecteur $X$
2. Appliquer la fonction exp() à ce vecteur et stocker le résultat dans un vecteur $Y$
3. Effectuer un affichage en utilisant les deux vecteurs précédents.

Le premier point s'effectue en utilisant l'une des deux fonctions suivantes :

```
X = borne_min:pas:borne_max
X = linspace(borne_min, borne_max, nb_points)

```

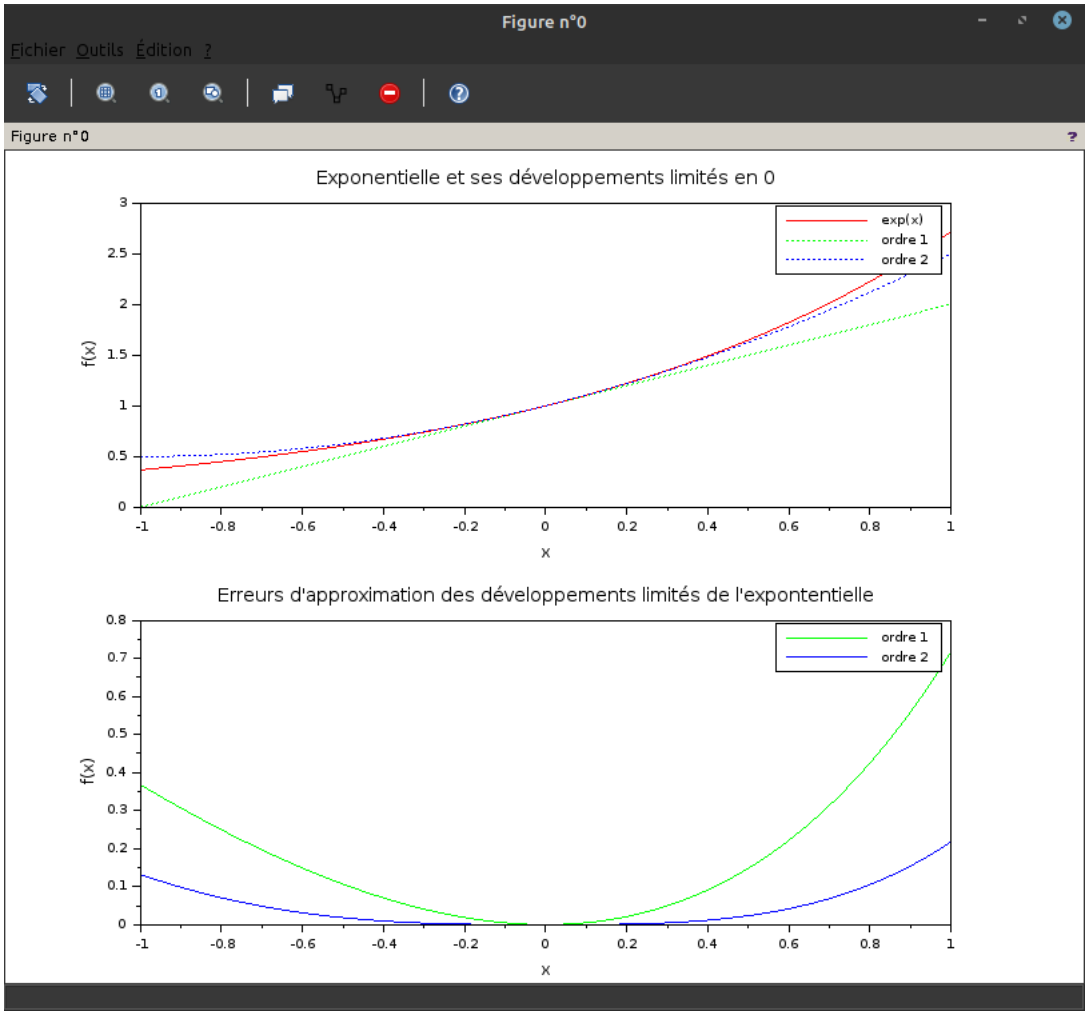


FIGURE 1 – Resultat souhaité

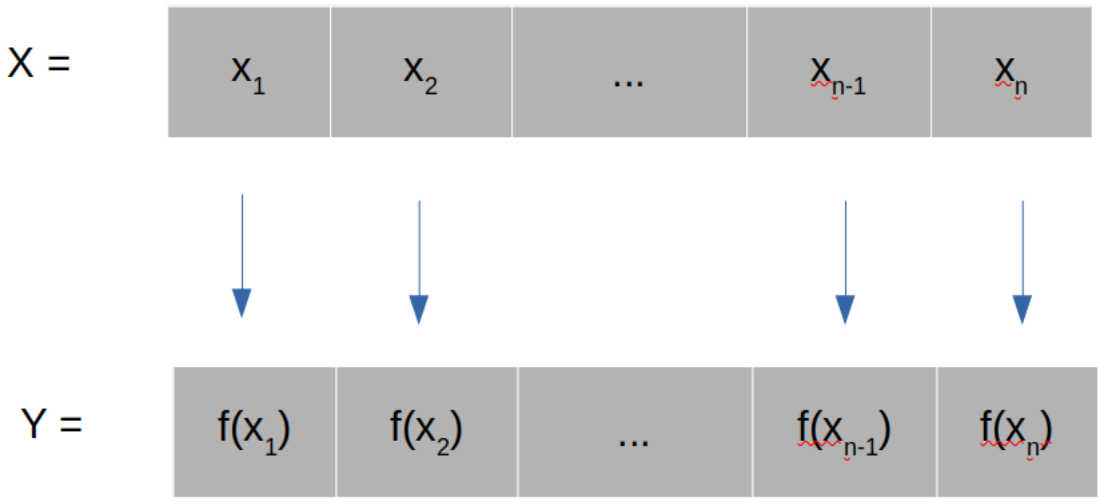


FIGURE 2 – Vecteurs à produire pour afficher une fonction dans Scilab

Vous pouvez utiliser l'une ou l'autre de ces fonctions, selon les informations dont vous disposez. Pour appliquer la fonction à votre vecteur, rien de plus simple, il vous suffit d'entrer la commande $Y = \exp(X)$. Pour effectuer l'affichage il vous suffit d'utiliser la fonction *plot* : plot$(X, Y)$.

8. Affichez le graphe de la fonction exponentielle.
9. Ce graphe n'est pas très beau, il lui faudrait un titre. Utilisez la documentation Scilab (–> help plot) pour trouver comment ajouter un titre à votre graphique.
10. Maintenant que l'affichage est correct, nous pouvons nous attaquer au développement limité d'ordre 1 de cette fonction. Proposez une fonction Scilab qui permet de calculer le développement limité d'ordre 1 de exp. Vérifiez en affichant les deux courbes sur le même graphique que l'approximation semble fonctionner. On utilisera dans la fonction la valeur du développement limité : $f(x) \approx 1 + x$
11. Proposez une fonction Scilab qui permet de calculer le développement limité d'ordre 2 de la fonction $x \mapsto \exp(x)$. On utilisera dans la fonction la valeur approchée du DL à l'ordre 2, à savoir : $f(x) \approx 1 + x + (x * x)/2$
12. Essayez d'afficher votre fonction, *cela ne devrait pas fonctionner*. Pourquoi ?
13. Le problème vient de l'opération $x * x$. Cette opération prend votre vecteur à 100 points et le multiplie par lui-même, sans succès. Les opérateurs .* et ./ permettent d'effectuer des opérations élément par élément dans un vecteur tandis que les opérations * et / effectuent des opérations matricielles. Corrigez votre code afin qu'il fonctionne avec n'importe quel ensemble de départ. L'opération .^2 calcul le carré d'un vecteur élément par élément.
14. Le code qui suit permet de séparer votre figure en six parties pour afficher différents éléments, utilisez le même procédé pour séparer votre figure en deux parties et afficher en haut la fonction exponentielle ainsi que ses deux premiers DL, en bas les erreurs d'approximation des premier et deuxième développements. On veillera à ce que les couleurs des courbes d'erreur correspondent aux couleurs des courbes initiales (cf figure 1)

```
1                X = linspace(-1,1, 100)
2
3                subplot(3,2, 1) // On se place dans la premiere figure d'un espace
divise en 2 lignes 3 colonnes
4                plot(X, sin(X)) // On affiche sin(X)
```

```
                plot(X, cos(X)) // On affiche cos(X)
                title("Le titre de la partie 1")

                subplot(3,2, 2)
                plot(X, [sin(X) cos(X)]) // On cree un vecteur de deux cases, dans chaque vecteur un ensemble d'arrivee
                title("Le titre de la partie 2")


                subplot(3,2, 3)
                plot(X, sin(X), "-r") // Pour plus d'infos : help  LineSpec
                plot(X, cos(X), ":b")
                title("Le titre de la partie 3")

                subplot(3,2, 4)
                plot(X, sin(X))
                plot(X, cos(X))
                title("Le titre de la partie 4")

                subplot(3,2, 5)
                plot(X, sin(X))
                plot(X, cos(X))
                xlabel("axe x")
                title("Le titre de la partie 5")

                subplot(3,2, 6)
                plot(X, sin(X))
                plot(X, cos(X))
                ylabel("axe y")
                title("Le titre de la partie 6")

```

---

## 3 Mesure du temps d'exécution d'un programme, calcul mathématiques, vecteurs, itération

Dans cette partie de l'exercice, on cherche d'une part à introduire la boucle for et d'autre part à bannir son usage autant que faire se peut dans des programmes de calcul. On se propose d'afficher les milles premiers termes de la série des inverses des carrés et de vérifier expérimentalement qu'elle converge vers $\frac{\pi^2}{6}$.

15. On a vu précédemment deux manière de faire des vecteurs, mais voici d'autres manières de créer des vecteurs.

---

```
                    --> X = 1:10
                    --> X = [1 2 5 7] // cree un vecteur d'entiers entre 1 et 10
                    --> X = zeros(1, 10) // cree un vecteur contenant dix zeros
                    --> X = ones(1, 10) // cree un vecteur contenant dix un
                    --> X = rand(1, 10, "normal") // cree un vecteur de dix nombres aleatoires suivant une loi normale centree reduite
                    --> X(3)
                    ans =
                        5.
```

Il est possible de transposer un vecteur à l'aide de l'opérateur ' afin d'obtenir un vecteur colonne. Toutes les opérations usuelles sur des vecteurs sont définies dans Scilab. Essayez ces différentes manières de faire des vecteurs.

16. On rappelle que le produit scalaire (canonique) de deux vecteurs $X = (x_1, x_2, \ldots, x_n)$ et $Y = (y_1, y_2, \ldots, y_n)$ de $\mathbb{R}^n$ est calculé comme :

$$< X, Y >= X \cdot Y' = x_1 \cdot y_1 + x_2 \cdot y_2 + \ldots + x_n \cdot y_n.$$

Proposer une fonction $scalaire(a, b)$ qui calcule le produit scalaire de deux vecteurs. Proposez une fonction $orthogonal(vect1, vect2)$ qui prend deux vecteurs de même dimension et retourne *True* si ils sont orthogonaux, *False* sinon.

17. On note que le vecteur

$$X = \left(\frac{1}{1^2}, \frac{1}{2^2} \ldots, \frac{1}{n^2}\right),$$

peut être construit à partir d'opérations élément par élément à partir du vecteur $V = (1, 2, \ldots, n)$. Proposer une fonction suite(n) qui crée un tableau contenant les $n$ premiers éléments de la suite en utilisant un vecteur de taille $n$ :

$$U(n) = \frac{1}{n^2}.$$

18. à l'aide de la documentation de sum (help sum) et votre fonction précédente, proposez une fonction Serie(n) qui calcule les n premiers éléments de la suite et les somme entre eux pour obtenir

$$S(n) = \sum_{i=1}^{n} \frac{1}{i^2}.$$

19. La boucle for (cf syntaxe ci-après, issue de la documentation de Scilab) permet d'effectuer un ensemble d'opérations similaires plusieurs fois d'affilé. Par exemple la boucle for ci-après effectue 5 fois l'opération disp(i), en modifiant à chaque fois la valeur de $i$. Chacune des valeurs de $i$ étant prise dans le tableau 1 : 5. Son retour est 1, 2, 3, 4 et 5. La deuxième fonction affiche $i+1$. Elle affiche donc 2, 3, 5 et 6. En utilisant une boucle for, proposez une autre fonction $S\_2(n)$ qui calcule la même série que précédemment.

```
for i = 1:5 // Affiche les entiers de 1 a 5
disp(i);
end

X = [1 2 4 5]
for i = X
disp(i + 1); // Affiche les nombres dans le vecteur X
end

```

20. La fonction timer renvoie le temps qui s'est écoulé depuis son dernier appel.

```
timer();
A=rand(100,100);
time_elapsed=timer()

```

Le code précédent (issu de la documentation de Scilab) stocke dans la variable *time_elapsed* le temps écoulé pendant la création du tableau A. Proposez un code qui permet de comparer les temps d'exécution de vos deux fonctions. Quelle est la fonction la plus rapide ? À votre avis, pourquoi ?

# 4 Mesure du temps d'exécution d'un programme, calcul mathématiques, vecteurs, itération (Deuxième exemple)

21. Pour manipuler des éléments d'un vecteur, on peut utiliser leurs indices.

```
--> a = 1:10
--> a(2:4)
 ans  =
2.   3.   4.

```

Que fait le bout de code ci-dessus ? Proposer une fonction qui prend en argument un tableau et renvoie un tableau contenant toutes les valeurs en dessous de la médiane et toutes les valeurs au dessus de la médiane dans deux tableaux différents. On utilisera la fonction gsort(tableau_a_trier, "g", "i") pour trier et size(tableau, 2) pour connaitre la longueur du tableau donné en argument.

22. Soient $X$ un vecteur et $Y$ le vecteur donné par $f(X)$ pour $f$ une fonction connue. Proposer une fonction deriv(Y), qui prend en entrée ce vecteur $Y$ et renvoie un vecteur contenant une approximation de la dérivée définie comme $f'(n) = \frac{f(X_{n+1})-f(X_n)}{Xn+1-X_n}$ *en utilisant une boucle for.*
23. Créez un vecteur $X$ de 4000 points répartis uniformément dans l'intervalle $[-10, 10]$. Afficher sur un même graphique la fonction de terme général $f(x) = x^4$ et ses quatre premières dérivées approchées.
24. Proposez un code qui calcule le temps écoulé pour calculer la dérivée d'un vecteur donné.
25. Proposez une fonction qui calcule la dérivée approchée d'un vecteur $X(n)$ *sans utiliser de boucle for.*
26. Comparez les temps d'exécution de vos deux méthodes avec une fonction $f$ définie par $f(x) = x^3 + x^2$ en utilisant comme vecteur de départ un vecteur à $n$ points répartis dans l'intervalle $[-1000, 1000]$. Commencez avec un $n$ petit et augmentez-le jusqu'à obtenir une différence significative (cela dépend de la puissance de votre machine). Ne l'augmentez pas trop sinon votre programme risque de manger la mémoire de votre machine (et vous n'avez pas envie qu'il le fasse). À votre avis, pourquoi ces résultats sont-ils différents des précédents ?

# 5 Masques

27. Appliquer l'opérateur de comparaison à un tableau crée un tableau contenant comme uniques valeurs vrai et faux. Il contient vrai là où le tableau est vrai, faux là où il est faux. Par exemple :

```
--> a = 1:10
--> a > 5
     ans  =
    F F F F F T T T T T

```

Appliquer un tel tableau comme argument de sélection dans un tableau permet de sélectionner uniquement les valeurs pour lesquelles celui-ci vaut T. Par exemple :

```
--> a(a > 5)
ans  =
    6.   7.   8.   9.   10.

```

Les opérateurs binaires permettent de composer les conditions, par exemple :

```
--> a(a > 5 & a < 8)
ans  =
```

```
                    6. 7.
--> a(a > 5 | a < 8)
ans =
1. 2. 3. 4. 5. 6. 7. 8. 9. 10
```

---

## 6 Exercices : écriture de quelques fonctions

28. Ajouter 0 à un booléen lui donnera la valeur 1 si il valait vrai, 0 sinon. À l'aide des questions précédentes, proposez en une ligne une définition de la fonction de Heaviside et tracez-la

$$H(x) = \begin{cases} 1 & \text{si } x > 0, \\ 0 & \text{sinon.} \end{cases}$$

29. Proposez une représentation graphique pour $x \in [-2, 2]$ de la fonction

$$f(x) = \begin{cases} \exp(x) - \exp(-1) & \text{si } -1 \leq x < 0, \\ \exp(-x) - \exp(-1) & \text{si } 0 \leq x < 1, \\ 0 & \text{sinon.} \end{cases}$$

30. Modifiez uniquement la commande d'affichage pour restreindre celui-ci au support de la fonction