
Informatique - T.P n°1

À la découverte de *SciLab*

On commencera par créer, dans son dossier personnel, un dossier intitulé "TP1" dans lequel on pourra enregistrer tous les programmes (au format *.sce*) ou fonctions (au format *.sci*) relatifs à cette première feuille d'exercices.

1 Calculs non programmés

Quand les commandes à taper sont simples et peu nombreuses, on peut directement utiliser la feuille de travail *SciLab* (appelée *console*).

L'invite `-->` de la console indique que *SciLab* est prêt pour exécuter de nouvelles commandes.

Exercice 1. (*SciLab* comme une calculatrice)

- (1) Calculer le résultat des opérations suivantes

$$\frac{1}{2 \times 3}; \quad \frac{1}{2} \times 3; \quad 7^4; \quad \frac{1 + \sqrt{5}}{2}; \quad \frac{1}{1 + \frac{1}{1+2}}; \quad \left| \frac{2}{3} - \frac{\sqrt{2}}{2} \right|.$$

- (2) Déterminer la valeur du plus grand entier naturel inférieur ou égal à $\ln(385)$.
(3) Quelle est la valeur de π ? De e ? De $\pi \times e$?
(4) (i) Calculer $\ln(3) \cdot e^2$.
(ii) Puis, à l'aide de la variable `ans`, calculer

$$\frac{\ln(3) \cdot e^2}{1 + \ln\left(\frac{7}{8}\right)}.$$

- (iii) Peut-on encore utiliser `ans` pour calculer $\ln(3) \cdot e^2 \cdot \sqrt{0,1}$? Taper la commande nécessaire au calcul de l'expression précédente (on pourra utiliser les touches \uparrow ou \downarrow).

Exercice 2. (Affectation)

- (1) (i) Créer une variable `a` puis lui affecter la valeur 10. Taper ensuite dans l'invite de commande `a=a*10`. Que vaut maintenant `a`?
(ii) Créer et affecter la valeur $5e$ à une variable `b`. Calculer $a + b$.
(iii) Taper la commande `clear` puis à nouveau `a`. Que se passe-t-il?
- (2) (i) Écrire une suite de deux instructions qui affecte à la variable `a` le nombre 10, calcule le nombre $(1 + \frac{1}{a})^a$ et fait afficher cette valeur.
(ii) En rappelant les instructions précédentes dans la ligne de commande, faire afficher le nombre $(1 + \frac{1}{100})^{100}$, puis $(1 + \frac{1}{1000})^{1000}$.
(iii) Faire afficher la valeur du nombre e . Que peut-on conjecturer?

Remarque. (Erreurs d'arrondi)

Les nombres réels sont représentés par des *flottants* (sorte d'équivalent informatique de l'écriture scientifique). Cela dit, il n'est pas possible de représenter tous les réels exactement sur un ordinateur (par exemple, 0,2 s'écrit - en base 2 - 0,00110011...). La machine arrondit donc tout résultat vers le plus proche nombre flottant. La précision de cet arrondi est définie par l' ϵ -*machine*; c'est la distance entre 1 et le plus proche nombre flottant qui lui est supérieur. Il vaut

$$\%eps = 2.220E-16$$

En particulier, aucun nombre compris entre 1 et $1 + \frac{\epsilon}{2}$ n'est représentable. Il y a donc un arrondi dans la représentation machine et des erreurs d'arrondi qui peuvent s'accumuler au fur et à mesure du calcul.

2 Premiers programmes

Un programme est une suite d'instructions regroupées dans un fichier. On écrira les programmes avec l'éditeur de textes **SciNotes**, intégré à **SciLab**.

La commande **input** permet de faire lire la valeur d'une variable au cours de l'exécution d'un programme (on peut par exemple écrire la commande `n=input('entrez un nombre entier')` pour que le programme demande à l'utilisateur de rentrer un entier dont la valeur sera affecté à la variable n).

On utilisera la commande **disp** (pour *display*) pour afficher la valeur d'une variable (ou une chaîne de caractères).

Il ne faut pas hésiter à utiliser le symbole `//` afin de commenter les lignes de code pour s'y retrouver!

Enfin, on signale que les eux opérateurs logiques à connaître sont **&** (signifiant "et") et **|** (signifiant "ou").

La syntaxe des alternatives `if...then` et des boucles `for` et `while` sera précisée au cours du TP.

Exercice 3. (Petits programmes entre amis)

- (1) Écrire un programme qui renvoie le pourcentage d'évolution entre deux valeurs rentrées par l'utilisateur.
- (2) Écrire un programme qui demande un nombre réel x et, après discussion, renvoie la racine carrée de celui-ci quand cela est possible.
- (3) Écrire un programme qui, en fonction des trois paramètres a, b, c rentrés par l'utilisateur, renvoie les solutions réelles de l'équation $ax^2 + bx + c = 0$.
- (4) Écrire un programme qui demande un nombre réel x et renvoie 1 si $x \in [0; 1]$ et 0 sinon.
- (5) Écrire un programme qui demande un nombre réel x et renvoie 1 si $x \in]-\infty; -2] \cup]\ln(2); +\infty[$ et 0 sinon.
- (6) Écrire un programme, faisant intervenir une boucle `for`, qui demande un entier $n \geq 0$, calcule et affiche $n! = 1 \times 2 \times 3 \times \dots \times n$. (On rappelle - notamment pour l'initialisation - que, par convention, $0! = 1$.)
- (7) Écrire un programme, faisant intervenir une boucle `while`, qui demande un nombre positif t , calcule et affiche en combien d'années une valeur subissant une hausse annuelle de $t\%$ aura doublé.

Exercice 4. (Un comportement asymptotique)

Écrire un programme qui demande un entier $n > 0$ et qui, en utilisant une boucle `for`, calcule et affiche

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} - \ln(n).$$

Faire tourner ce programme pour de plus en plus grandes valeurs de n . Que remarque-t-on?

Lorsque l'on a besoin de calculer une quantité qui dépend d'un paramètre plusieurs fois (et d'éventuellement s'en servir dans d'autres calculs), on écrit souvent le programme sous forme de fonction. En enregistrant le programme (qui débute par `function y=nomdelafonction(x)` et se termine toujours par `endfunction`) au format `.sci`, on peut ensuite charger la fonction dans l'environnement pour `y` faire appel.

Exercice 5.

(1) Écrire une fonction `SciLab` correspondant à

$$f(x) = \frac{\ln(x + \sqrt{x^2 + 1})}{x^2 + x + 1}$$

- (i) *Quel est le domaine de définition de f ?*
- (ii) Conjecturer (à l'aide de `SciLab` puis le *justifier*) sur la valeur de

$$\lim_{x \rightarrow +\infty} f(x).$$

(2) Programmer les fonctions

$$\begin{array}{l} g_1 : \mathbb{Z} \longrightarrow \mathbb{N} \\ n \longmapsto \begin{cases} 1, & \text{si } n \text{ est pair} \\ 0, & \text{sinon} \end{cases} \end{array}, \quad \begin{array}{l} g_2 : \mathbb{R} \longrightarrow \mathbb{N} \\ x \longmapsto \begin{cases} 1, & \text{si } x \in \mathbb{Z} \\ 0, & \text{sinon} \end{cases} \end{array}.$$

(3) Écrire les programmes des (4) et (5) de l'Exercice 3 sous forme de fonctions.

- (4) (i) Utiliser les instructions du programme du (6) de l'Exercice 3 pour écrire une fonction intitulée `factorielle` prenant en argument un entier $n \geq 0$ et renvoyant la valeur $n!$.
- (ii) Utiliser cette fonction pour écrire un programme qui demande à l'utilisateur un nombre réel positif x et renvoie le plus petit entier naturel n tel que $n! \geq x$. (On utilisera une boucle `while`.)

3 Premières représentations graphiques

`SciLab` propose de dessiner des "lignes brisées" grâce à la fonction `plot2d()`. Plus précisément, l'instruction `plot2d(X,Y)` permet de relier les points $\{(x_i, y_i) : i = 1, \dots, n\}$, où on a défini au préalable les deux listes (abscisses et ordonnées des points à relier) $X = [x_1, x_2, \dots, x_n]$ et $Y = [y_1, y_2, \dots, y_n]$.

Pour représenter une fonction, on utilisera donc `plot2d()` en prenant pour liste des abscisses une *subdivision* de l'intervalle sur lequel on veut représenter la fonction.

Exercice 6.

(1) Que renvoient les instructions

- (i) `--> X=-1:0.1:1?`
- (ii) `--> X=linspace(-1,1,21)?`

(iii) Expliciter comment fonctionnent les deux instructions précédentes dans le but de créer des subdivisions d'intervalle.

- (i) Définir, dans la console, la fonction f où $f(x) = xe^{-x}$.
- (ii) Créer une liste de nombres correspondant aux images par la fonction f des réels de la liste `X` de **deux manières différentes** (à l'aide d'une boucle et avec la commande `feval()`).

- (iii) Représenter la fonction f sur l'intervalle $[-1; 1]$.
- (2) Définir, dans la console, la fonction g où $g(x) = x^2 e^{-x}/2$.
- (3) Représenter simultanément les fonctions f et g sur l'intervalle $[0; 5]$.