TP 2

# Révisions : Les fonctions en Scilab



**Compétences attendues.**

- ✓ Définir une fonction en Scilab.
- ✓ Tracer la courbe représentative d'une fonction ou les premiers termes d'une suite.
- ✓ Calculer une valeur approchée de la racine d'une équation du type $f(x) = 0$ par la méthode de dichotomie.
- ✓ Calculer une valeur approchée d'un point fixe d'une fonction à l'aide d'une suite récurrente.


Anthony Mansuy
Professeur de Mathématiques en deuxième année de CPGE filière ECE au Lycée Clemenceau (Reims)
Page personnelle : `http://anthony-mansuy.fr`
E-mail : `mansuy.anthony@hotmail.fr`

# 1 Les fonctions

**Définition.**

- **Fonctions usuelles :** `log` (logarithme népérien ln), `exp`, `sqrt` (racine carrée), `floor` (partie entière) et `abs` (valeur absolue).
- **Syntaxe pour définir une fonction réelle d'une variable réelle $f$ :**

```
function y=f(x)
      y=...
endfunction
```

- **Plus généralement, syntaxe pour définir une fonction `Scilab` :**

```
function [y1,y2,...,yp]=nom_fonction(x1,x2,...,xn)
      instructions
endfunction
```

**Exemple.** Pour définir la fonction $x \mapsto (1+3x+2x^2-x^4)e^{-x}$, recopier le script suivant dans `SciNotes` :

```
function y=P(x)
      y=(1+3*x+2*X^2-x^4)*exp(-x)
endfunction
```

Après avoir exécuté le fichier dans `Scilab`, on peut alors utiliser cette fonction dans la console. Taper par exemple :

```
--> P(1),P(-1),P(sqrt(3))
```

**Remarques**

- Une fois qu'une fonction a été chargée (le script qui la contient a été sauvé et exécuté), elle est disponible jusqu'à la fin de la session `Scilab`.
- Les variables utilisées dans le corps d'une fonction sont des variables locales : elles n'existent pas à l'extérieur de la fonction.

**Attention.**

Inutile d'ajouter les fonctions d'entrée et de sortie `input` et `disp`. L'utilisateur est sensé connaitre l'argument à rentrer pour la fonction. `Scilab` lui, renverra le contenu des variables de sorties `y` ou `[y1,y2,...,yp]`.

Dans l'exemple précédent, l'utilisateur rentre donc un réel à la fonction `P`, et `Scilab` renvoie le contenu de la variable `y` en sortie.

**Exemple.** Recopier la fonction suivante dans `SciNotes` :

```
// Plus grand et plus petit de deux nombres
// m: le plus petit, M: le plus grand
function  [m,M] = min_max(x,y)
if x<=y then
      m=x;
      M=y;
      else m=y;
           M=x;
end
endfunction
```

Enregistrer et exécuter cette fonction. Tester cette fonction sur l'exemple suivant :

```
-->min_max(2.3,-4.7)
```

**Remarque.** Si la fonction possède plusieurs paramètres de sortie, la syntaxe

```
nom_fonction(valeurx1,...,valeurxn)
```

ne rend que la valeur de `y1`. Pour obtenir toutes les valeurs de sortie, il faut utiliser la syntaxe

```
[nomvar1,...,nomvarp]=nom_fonction(valeurx1,...,valeurxn).
```

**Exemple.** Taper l'instruction suivante :

```
-->[m,M]=min_max(2.3,-4.7)
```

**Remarques.**

- Les paramètres d'entrée `x1,x2,...,xn` et les paramètres de sortie `y1,y2,...,yp` peuvent être de tout type (réel, vecteur, matrice, etc...).
- Si la fonction ne nécessite pas de paramètre de sortie (la fonction ne fait que des actions), on écrira :

  ```
  function []=nom_fonction(x1,x2,...,xn).
  ```

  Si la fonction ne nécessite pas de paramètre d'entrée, on écrira :

  ```
  function y=nom_fonction().
  ```

# 2 Les représentations graphiques

## 2.1 Fonction `plot2d`

Pour représenter une courbe, `Scilab` trace des points et les relie par des lignes droites.

**Définition.**

Soit `x` et `y` deux vecteurs-lignes (ou deux vecteurs-colonnes) de même taille.
La commande `plot2d(x,y)` trace une ligne brisée entre les points dont les abscisses sont données par le vecteur `x` et les ordonnées sont données par le vecteur `y`.

**Exemple.** Taper les instructions suivantes :

```
-->x=[-1, 0, 1, 2]; y=[2, -1, 0, 3]; plot2d(x,y)
```

**Remarques.**

- On peut aussi utiliser l'instruction `plot` qui diffère de `plot2d` uniquement par les options possibles. Consulter l'aide pour cela.

  Dans la pratique, plutôt que d'utiliser des options, on effectuera les modifications désirées directement dans la fenêtre graphique.
- Si on trace successivement plusieurs courbes, elles se superposent dans la même fenêtre.
  Pour supprimer les graphiques précédents, on utilise l'instruction `clf()` (clear figure).
- Si on désire superposer plusieurs représentations graphiques, on peut utiliser
  `plot2d(x,[y1,y2,...,yn])` où `x,y1,y2,...,yn` sont des vecteurs-**colonnes**.
  On trace ainsi les lignes brisées correspondant aux couples `(x,y1)`, `(x,y2)`, ..., `(x,yn)` sur un même graphique.

## 2.2 Tracé de la courbe représentative d'une fonction

Pour tracer la courbe représentative d'une fonction $f$ sur un intervalle $[a,b]$, on peut utiliser l'instruction `plot2d`. Pour cela, on précise le vecteur `x` des abscisses à considérer et le vecteur `y` des images par $f$ :

- Pour le vecteur des abscisses, on prend une « discrétisation » de l'intervalle $[a,b]$ avec un pas petit. Par exemple, si $[a,b]=[0,1]$, on peut taper l'instruction (pour un pas de $1/100$) :

```
x=linspace(0,1,100)
```

- Pour le vecteur des images par $f$ :
  - Si la fonction $f$ est usuelle ou si sa définition autorise un vecteur en paramètre d'entrée alors on peut l'appliquer directement au vecteur `x`.
  - Sinon, il faut l'appliquer à `x` à l'aide de `feval`.

**Exemples.** Taper les instructions suivantes.

```
-->x=linspace(-1,3,100); y=exp(x); plot2d(x,y)
```

Taper ensuite les instructions suivantes :

```
-->clf()
-->x=linspace(-1,3,100); y=P(x); plot2d(x,y)
```

On notera alors un message d'erreur. Pourquoi ?

Taper l'instruction suivante :

```
-->x=linspace(-1,3,100); y=feval(x,P); plot2d(x,y)
```

**Remarque.** Il est possible d'éviter le message d'erreur en changeant la définition de $P$, et en autorisant les opérations pour les vecteurs :

```
function y=P(x)
      y=(1+3*x+2*x.^2-x.^4).*exp(-x)
endfunction
```

Généralement, il est plus simple d'utiliser l'instruction `fplot2d`.

**Définition.**

Soit `x` un vecteur et `f` une fonction.
`fplot2d(x,f)` trace une ligne brisée entre les points dont les abscisses sont données par le vecteur `x` et les ordonnées sont les images de `x` par `f`.

**Exemple.** Taper les instructions suivantes :

```
-->clf()
-->x=linspace(-1,3,100); fplot2d(x,P)
```

**Remarque.** Dans le cas où plusieurs représentations graphiques se superposent, on peut les différencier par couleur en utilisant par exemple `plot2d(x,y,3)` ou `fplot2d(x,P,5)`

# 3 Exercices

**Exercice 1**

Compléter la fonction suivante afin qu'elle retourne le couple $a,b$, où $a=\sum_{i=1}^{n}\sqrt{i}$ et $b=\sum_{i=1}^{n}\lfloor\ln(i)\rfloor$.

```
function [a,b] = sommes(n)
      a = ...
      b = ...
endfunction
```

## Exercice 2

1. On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = 2nu_n + 3$.

   Écrire une fonction en `Scilab` ayant en entrée un entier naturel $n$ et qui renvoie la valeur de $u_n$.

2. Même question avec $(v_n)_{n\in\mathbb{N}}$ définie par $v_0 = 1$, $v_1 = -2$ et pour tout $n \in \mathbb{N}$, $v_{n+2} = 2v_{n+1} - v_n$.

3. On considère les suites $(a_n)_{n\in\mathbb{N}}$ et $(b_n)_{n\in\mathbb{N}}$ définies par $a_0 = 1$, $b_0 = 2$ et les relations :

$$\forall n \in \mathbb{N},\ a_{n+1} = \sqrt{a_n b_n} \quad \text{et} \quad b_{n+1} = \frac{a_n + b_n}{2}.$$

   Écrire une fonction en `Scilab` ayant en entrée un entier naturel $n$ et qui renvoie le couple $a_n$, $b_n$.

---

## Exercice 3

1. Définir en `Scilab` la fonction $f$ suivante :

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

2. Écrire les commandes `Scilab` permettant de tracer la courbe représentative de $f$ sur $[-2, 2]$

---

## Exercice 4

1. On considère la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par : $f(x) = \dfrac{e^x - e^{-x}}{2}$.

   Montrer que $f$ réalise une bijection de $\mathbb{R}$ dans $\mathbb{R}$.

2. Définir une subdivision de l'intervalle $I = [-2, 2]$ en 100 sous-intervalles de même longueur.

3. Écrire les commandes permettant de tracer sur un même graphique la courbe représentative de la fonction $f$ définie sur $I$ ainsi que celle de sa fonction réciproque (sans chercher à déterminer cette dernière).

---

## Exercice 5

On considère la fonction $f$ définie pour tout $x \in \mathbb{R}$ par $f(x) = e^x + x - 2$.

1. Montrer qu'il existe un réel $c \in [0, 1]$ tel que $f(c) = 0$.

2. Écrire les commandes permettant de tracer la courbe représentative de $f$ sur $[0, 1]$. Obtenir à partir du graphique une approximation de $c$.

3. Écrire un programme qui, étant donné $\varepsilon > 0$, permet de calculer une approximation de $c$ à $\varepsilon$ près en utilisant la méthode de dichotomie.

---

## Exercice 6

1. On considère la fonction $f(x) = e^{-x} + 1$.

   (a) Étudier les variations de $f$ et montrer que l'intervalle $[1, 2]$ est stable par $f$.

   (b) Montrer que l'équation $f(x) = x$ admet une unique solution dans l'intervalle $[1, 2]$ notée $\alpha$.

   (c) Écrire une fonction `f` sur `Scilab` qui prend en entrée un réel $x$ et qui calcule $f(x)$.

   (d) Écrire les commandes `Scilab` permettant de tracer sur un même graphique la courbe représentative de $f$ et la droite d'équation $y = x$ sur $[1, 2]$. Obtenir à partir du graphique une approximation de $\alpha$.

2. On définit la suite $(u_n)_{n\in\mathbb{N}}$ par $u_0 = 1$ et la relation : $\forall n \in \mathbb{N}$, $u_{n+1} = f(u_n)$.

   (a) Montrer que, pour tout $n \in \mathbb{N}$, $u_n \in [1, 2]$.

   (b) En utilisant la fonction `f` précédente, écrire une fonction `SuiteU` qui prend en entrée un entier naturel $n$ et qui calcule $u_n$.

(c) On considère le programme `Scilab` suivant :

```
n=20
absc=0:n-1
ord=zeros(1,n)
ord(1)=1
for k = 1:n-1
    ord(k+1) = f(ord(k))
end
plot(absc, ord, '+')
```

Recopier ce programme et l'exécuter. Que réalise ce programme ?

3. (a) Montrer que : $\forall n \in \mathbb{N}$, $|u_{n+1} - \alpha| \leq \dfrac{1}{e} |u_n - \alpha|$.

(b) En déduire que : $\forall n \in \mathbb{N}$, $|u_n - \alpha| \leq \dfrac{1}{e^n}$. Puis obtenir $\lim\limits_{n \to +\infty} u_n$.

(c) En utilisant la fonction `SuiteU` précédente, écrire un programme `Scilab` qui donne une approximation de $\alpha$ à $\varepsilon$ près pour $\varepsilon > 0$ donné.

---

## Exercice 7

On note, pour tout $n \in \mathbb{N}^*$, $(E_n)$ l'équation : $x^n + x - 1 = 0$.

1. Soit $n \in \mathbb{N}^*$. Étudier les variations sur $\mathbb{R}^+$ de la fonction $x \mapsto x^n + x - 1$.

   En déduire que l'équation $(E_n)$ admet une unique solution sur $\mathbb{R}^+$ que l'on note $u_n$.

2. Recopier et compléter la fonction `Scilab` suivante afin que, prenant en argument un entier $n \in \mathbb{N}^*$, elle renvoie une valeur approchée de $u_n$ à $10^{-3}$ près, obtenue à l'aide de la méthode par dichotomie.

```
function u = valeur_approchee(n)
    a = 0
    b = 1
    while ...
        c = (a+b)/2
        if (c^n+c-1)>0 then
            ....
        else
            ....
        end
    u = ....
    end
endfunction
```

3. On considère le programme `Scilab` suivant :

```
n = 100
absc = 1:n
ord = zeros(1,n)
for k = 1:n
    ord(k) = valeur_approchee(k)
end
plot(absc, ord, '+')
```

Recopier ce programme et l'exécuter. Que réalise ce programme ? Quelles conjectures peut-on faire sur la suite $(u_n)$ concernant sa monotonie, sa convergence et son éventuelle limite ?

---