TP5 : Fonctions et fonctions

1. Définition de fonctions *Scilab*

Une fonction *Scilab* est une variable très particulière, elle permet d'effectuer plusieurs instructions à son appel et doit être déclarée avec la syntaxe suivante :

```
function [arguments de sortie]=nom(arguments d'entrée)
   instructions
   ...
endfunction
```

Exemple (à tester et comprendre - écrire dans SciNotes) :

```
function y=double(x)
   y=2*x
endfunction
```

Charger la fonction (touche F5) puis taper dans la console :

```
-->double(7).
```

Remarques :

(a) En tant que variable une fonction *Scilab* peut donc être passée en argument d'une autre fonction (une fonction peut donc en appeler une autre).

(b) Les crochets ne sont pas obligatoires lorsque l'argument de sortie est unique.

(c) À la différence d'une fonction mathématique, une fonction *Scilab* peut très bien retourner des arguments de sortie sans aucun argument d'entrée.

Exemple (à ajouter à la suite de la fonction `double` *dans le même fichier)* :

```
function y=quadruple(x)
   y=double(double(x))
endfunction
```

Charger la fonction, puis taper dans la console :

```
-->quadruple(25)
-->quadruple(%pi)
```

Exemple (à taper dans SciNotes) :

```
function stom=envers(mots)
   n=length(mots)  //n : nombre de caractères dans mots
   stom=""  //stom : chaîne de caractères vide
   for i=n:-1:1
      stom=stom+part(mots,i)
      //on parcourt les lettres de mots à l'envers
      //et on les enregistre dans stom
   end
endfunction
```

Charger la fonction (F5), puis essayer dans la console :

```
-->envers("Cette fonction est vraiment très utile !")
```

Posez-vous les questions suivantes :
- Quel est le nom de la fonction ?
- Quel est le nom de la variable d'entrée ?
- Quel est le nom de la variable de sortie ?
- Que fait cette fonction ?

Remarque :
L'intérêt de créer des fonctions se situe dans les programmes longs qui exécutent des tâches variées. Il est alors pratique de créer une fonction pour chacune des ces tâches. Cela permet entre autres : de rendre le code plus lisible, de repérer plus facilement les bugs, de pouvoir réutiliser certaines fonctions dans d'autres programmes.

2. Fonctions récursives et itératives

En informatique, une fonction récursive est une fonction qui s'appelle elle-même. A contrario, une fonction qui ne s'appelle pas elle même sera dite itérative (ou encore impérative).

Exemple (La factorielle itérative - tester et comprendre) :

```
function f=factoIt(n)
   f=1
   for i=1:n
         f=f*i
   end
endfunction
```

Exemple (La factorielle récursive - tester et comprendre) :

```
function f=factoRe(n)
   if n<=1 then  //si n=0 ou 1 alors n!=1
      f=1
   else  //si n>1 alors n!=n*(n-1)!
      f=n*factoRe(n-1)
   end
endfunction
```

Remarque :
En *Scilab*, la méthode la plus simple pour obtenir $n!$ est encore :

```
-->prod(1:n)  //n doit évidemment être définie
```

3. Définition de fonctions au sens 'mathématiques'

On peut principalement définir des fonctions 'mathématiques' de deux manières :

(a) À partir des fonctions *Scilab*

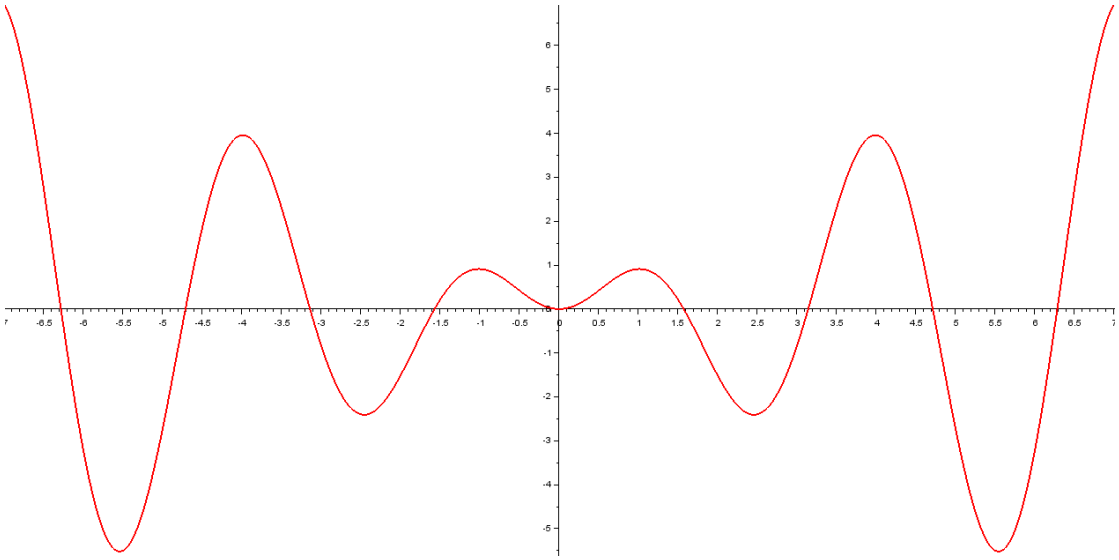
Exemple ($x \mapsto x\sin(2x)$) à taper dans SciNotes :

```
function y=f(x)
    y=sin(2*x).*x
endfunction
```

Charger la fonction (touche F5) puis taper dans la console :

```
-->x=linspace(-7,7,1000);
-->y=f(x);
-->plot2d(x,y)
```

On obtient alors le graphe de la fonction $x \mapsto x\sin(2x)$ sur l'intervalle $[-7, 7]$ comme ci-dessous (à la position des axes près).



(b) À l'aide de la commande `deff`

Syntaxe :

```
deff("[y1,y2,...]=NomFonction(x1, x2,...)","description"
)
```

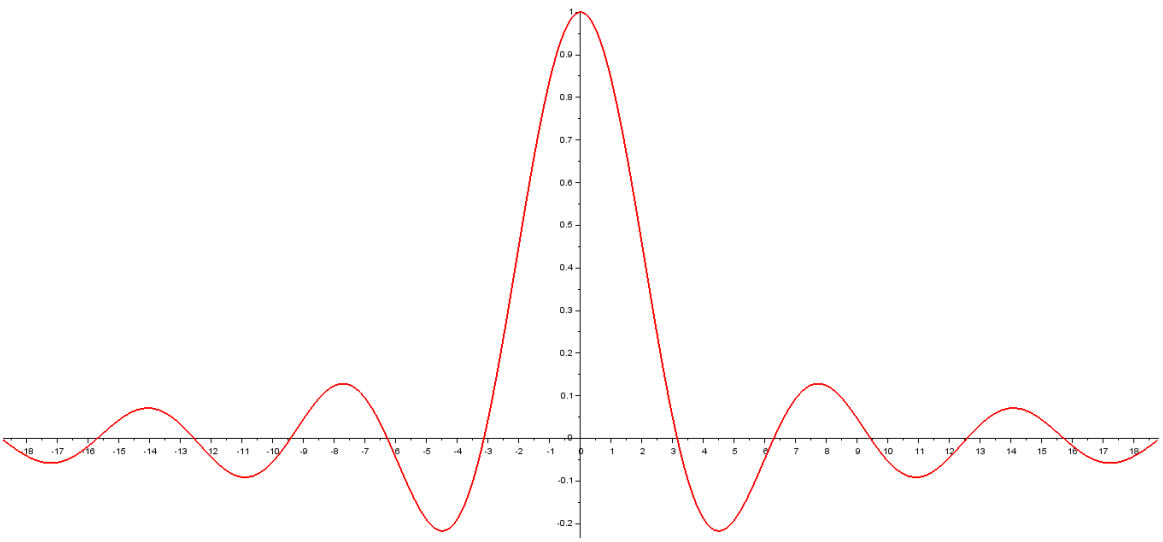
où `description` est un vecteur colonne de chaînes de caractères, constituant les instructions successives de la fonction, mais rien ne vaut un exemple :

Exemple (Sinus cardinal : $x \mapsto \frac{\sin(x)}{x}$) :

```
-->deff("[y] = sincard(x)", "y = sin(x)./x")
```

Puis taper dans la console :

```
-->x=-6*%pi:0.1:6*%pi;
-->y=sincard(x);
-->plot2d(x,y)
```



4. Exercices

Exercice 1. Graphe d'une fonction

Créer une fonction du nom de votre choix qui prendra en argument trois paramètres : a, b et h et qui tracera ensuite le graphe de la fonction

$$f : x \longmapsto \frac{(4x^3 - 6x^2 + 1)\sqrt{\sin(x^2) + 1}}{1 + x^2}$$

sur $[a, b]$ et avec un pas de h.

Exercice 2. Conjecture de Syracuse - Collatz

1. Créer une fonction `syracuse` qui prendra en argument d'entrée un entier u_n et qui renverra $\frac{u_n}{2}$ si u_n est pair, $3u_n + 1$ si u_n est impair.
2. Créer un programme qui demandera à l'utilisateur un nombre u et qui affichera ensuite les termes de la suite $(u_n)_{n\in\mathbb{N}^*}$ définie par $u_1 = u$ et pour $n > 0$:

$$u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{, si } u_n \text{ est pair} \\ 3u_n + 1 & \text{, si } u_n \text{ est impair} \end{cases}$$

   tant que $u_n \neq 1$ (on utilisera, bien évidemment, la fonction syracuse précédente).
3. Modifier ce dernier programme pour qu'il affiche aussi l'entier n du rang d'arrêt.
   Le mathématicien Lothar Collatz conjecture en 1928 que, quelque soit le choix de l'entier u_1, la suite $(u_n)_{n\in\mathbb{N}^*}$ finit toujours par atteindre 1 puis devenir cyclique (1,4,2,1,...). À l'heure actuelle personne n'a réussi à confirmer ou infirmer cette conjecture.

Exercice 3. Suite de Fibonacci

Créer une fonction `fibonacci` qui prendra en argument un entier n et qui renverra les n premiers termes de la suite de Fibonacci.
Pour rappel la suite de Fibonacci est une suite récurrente linéaire d'ordre 2, définie par $u_1 = u_2 = 1$ et pour tout entier $n \in \mathbb{N}^*$

$$u_{n+2} = u_{n+1} + u_n$$

À partir de quel rang n a-t-on $u_n > 10^6$?

Exercice 4. Série et exponentielle

1. Créer dans SciNotes une fonction `factorielle` qui prendra en argument d'entrée un entier naturel k et qui renverra $k!$.
2. Ajouter dans le fichier déjà créé une fonction `puissance` qui prendra en arguments d'entrée deux paramètres, le premier un nombre x et le second un entier naturel k et qui renverra le nombre x^k sans utiliser l'opération $\widehat{}$.
3. Ajouter à la suite des deux fonctions précédentes une fonction `expo` qui prendra en arguments d'entrée deux paramètres, un nombre x et un entier naturel n et qui renverra la somme $\sum_{k=0}^{n} \frac{x^k}{k!}$ (on utilisera les fonctions `factorielle` et `puissance`). Comparer avec la valeur de e^x. Que remarquez-vous ?

Exercice 5. Triangle de Pascal

Soit $n \in \mathbb{N}$. On rappelle que si $k \in \mathbb{Z}$ et ($k < 0$ ou $k > n$) alors $\binom{n}{k} = 0$. De plus $\binom{n}{0} = \binom{n}{n} = 1$ et pour $k \in [\![1, n]\!]$, $\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$ (Formule du triangle de Pascal).

1. Écrire une fonction `triangle` qui prend en argument un vecteur `c=[c1,c2,...,cn]` et qui renvoie le vecteur `d=[d1,...,d(n+1)]` de taille `n+1` défini par :
$$d_1 = d_{n+1} = c_1, \qquad \forall k \in [\![2, n]\!], \quad d_k = c_{k-1} + c_k$$
2. Écrire une fonction `pascal` qui prend en argument un entier naturel n et qui affiche le triangle de Pascal jusqu'à la ligne n. (On utilisera la fonction triangle).

Exercice 6. Minimum et maximum

On interdit l'utilisation des fonctions `min` et `max` de *Scilab* dans cet exercice.

1. Créer une fonction `maximum` qui prend en argument un vecteur `v` et qui renvoie le plus grand de ses coefficients.
2. Faire de même avec une fonction `minimum` qui renvoie le plus petit de ses coefficients. (On pourra utiliser la fonction maximum).
3. Créer une fonction `MatMax` qui prend en argument une matrice `A` de taille $n \times p$ et qui retourne le plus grand de ses coefficients. Faire de même avec une fonction `MatMin` retournant le plus petit de ses coefficients.
4. Créer une fonction `MinMaxLi` qui prend en argument une matrice `A` de taille $n \times p$ et qui renvoie le plus petit des maximums de chaque ligne.
5. Faire de même avec une fonction `MinMaxCo` pour les colonnes.
6. On renverse le problème : créer deux fonctions `MaxMinLi` et `MaxMinCo` qui retournent respectivement le plus grand des minimums de chaque ligne/colonne.

Exercice 7. Méthode des rectangles

On rappelle que la somme de Riemann $S_{n,a,b}(f)$ associée à une fonction continue f sur $[a,b]$ est donnée par :

$$S_{n,a,b}(f) = \frac{1}{n}\sum_{k=1}^{n} f\left(a + k\frac{b-a}{n}\right)$$

et dans ce cas, on a : $\lim\limits_{n\to+\infty} S_{n,a,b}(f) = \displaystyle\int_a^b f(t)\mathrm{d}t$.

1. Créer une fonction *scilab* `rectangle` qui prendra en paramètres d'entrée n, a, b et renverra la valeur de $S_{n,a,b}(\arctan)$ (pour mémoire : `atan` en *scilab*).
2. Calculer (papier,stylo) la valeur exacte de $\displaystyle\int_0^1 \arctan(x)\mathrm{d}x$.
3. Vérifier la cohérence de vos résultats avec la fonction `rectangle`.
4. Mêmes questions avec la fonction $x \mapsto \frac{1}{\cos(x)}$ et $\displaystyle\int_0^{\pi/4} \frac{\mathrm{d}x}{\cos(x)}$.