Feuille de TP n° 6

Fonctions en Scilab

Ce TP accompagne le chapitre 3 (Informatique et Algorithmique) : **Fonctions en Scilab**.
Créez le dossier `..\ECS1B_TPInfo\TP6\` et faites-en le répertoire courant de Scilab. Tous les scripts devront être sauvegardés dans ce dossier.

I Entrainement

Prérequis : Vecteurs Scilab, Structures conditionnelles, Structures répétitives.

Nous avons vu que Scilab dispose de nombreuses commandes et fonctions prédéfinies. Nous allons à présent voir comment créer nos propres fonctions.

• On peut coder des fonctions de la variable réelle à valeurs réelles. Par exemple la fonction $f : x \longmapsto \dfrac{\sin(x)}{1+x^2}$, s'implémente comme cela en Scilab :

```
function y=fct1(x)
    y=sin(x)/(1+x^2);
endfunction
```

Recopier ce code dans un script Scilab. En écrivant `y=fct1(x)`, nous avons choisi de :

- stocker dans la variable `x` l'argument d'entrée de la fonction.
- stocker dans la variable `y` l'argument de sortie de la fonction
- appeler `fct1` la fonction.

Il faut absolument sauvegarder ce script sous le nom `fct1.sci` (`fct1` car c'est le nom que l'on a donné à la fonction et `.sci` car c'est l'extension pour les fonctions Scilab).

Exécuter le script sans écho dans la console. Désormais `fct1` s'ajoute à la liste des fonctions Scilab. Tester les commandes suivantes :

```
-->clear
-->fct1(0), fct1(%pi), fct1(1)
-->x
-->y
-->y=fct1(-1); y
```

• Constater que les variables `x` et `y` ne sont pas affectées. C'est normal ce sont des variables muettes.

• On a vu dans le TP1 que les fonctions usuelles (`exp, log, sqrt, ans, floor, sin, cos, tan`) peuvent prendre en entrée des vecteurs et renvoient un vecteur de même taille dont chaque coordonnée est évaluée par la fonction. Est-ce que cela fonctionne toujours avec la fonction que nous avons définie ?

Exécuter la commande `fct1([1,2,3])` dans la console.

Elle renvoie un message d'erreur. C'est normal car notre fonction n'est pas compatible avec les opérations algébriques sur les vecteurs. Faites les modifications nécessaires (c'est-à-dire remplacer `*` par '`.*`, `/` par `./` et $\wedge$ par `.`$\wedge$) et tester-la (après l'avoir enregistrée et exécutée à nouveau) avec la commande `fct1([1,2,3])`.

• Dans le corps d'une fonction Scilab, tout comme dans un programme, on peut écrire plusieurs instructions et utiliser une ou plusieurs structures conditionnelles et répétitives. Par exemple la fonction $f : x \longmapsto \begin{cases} \frac{\sin(x)}{x} & \text{si} \quad x > 0, \\ e^x & \text{si} \quad x \leqslant 0. \end{cases}$ s'implémente comme cela en Scilab :

```
function y=fct2(x)
     if   x<>0   then
            y=sin(x)./x;
     else
            y=exp(x);
     end
endfunction
```

Sauvegarder ce script (sous le nom `fct2.sci`), exécuter-le sans écho dans la console et tester les commandes suivantes :

```
-->fct2(0), fct2(%pi), fct2(%pi/2)
-->x=[-2,-1,0]; y=fct2(x)
-->x=[1,2,3]; y=fct2(x)
-->x=[%pi/2,%pi,0]; y=fct2(x)
```

• Remarquer que la dernière commande ne renvoie pas ce qu'on pourrait attendre. Le problème vient du fait que la condition `x>0` n'est pas vraie pour tous les éléments du vecteur. C'est donc l'instruction du `else` qui s'applique. Pour contourner ce problème, on peut créer une boucle qui va appliquer la fonction coordonnée par coordonnée :

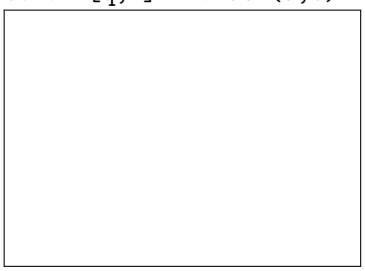
```
function y=fct2(x)
    l=length(x); y=zeros(1:l);//On crée un vecteur nul de même taille que x.
    for   i=1:l//On applique la fonction coordonnée par coordonnée.
          if   x(i)>0   then
                 y(i)=sin(x(i))/x(i);
          else
                 y(i)=exp(x(i));
          end
    end
endfunction
```

Faites les modifications et tester à nouveau la commande

```
-->x=[%pi/2,%pi,0]; y=fct2(x)
```

• On peut aussi créer des fonctions qui prennent plusieurs arguments d'entrée et/ou plusieurs arguments de sortie. Nous allons écrire une fonction qui prend en arguments deux entiers naturels et qui renvoie le quotient et le reste de la division euclidienne de a par b.

```
function [q,r]=DivEucl(a,b)
```



```
endfunction
```

Compléter la fonction (on pourra s'aider de la partie entraînement du TP5). Sauvegarder-la et exécuter-la sans écho dans la commande. Tester les commandes suivantes :

```
-->DivEucl(60,7)
-->v=DivEucl(60,7); v
-->[q,r]=DivEucl(60,7)
```

Tournez SVP

II Exercices

Commencez par taper `clear` *dans la console. Pour vous y retrouver plus facilement, veuillez écrire*

- *en commentaire au début de chaque fonction le numéro du TP et de l'exercice concerné.*
- *sur cette feuille de TP, le nom que vous avez donné à a fonction (extension en* `.sci`*).*

Vous testerez dans la console Scilab toutes les fonctions que vous écrirez (sans oublier de les exécuter au préalable).

Exercice 1. Écrire une fonction qui prend en entrée deux entiers naturels n et k (pas forcément tels que $k \leqslant n$) et qui calcule $\binom{n}{k}$. On se donnera la contrainte de ne pas utiliser de factorielles. La fonction répondra un message d'erreur si n ou k ne sont pas des entiers naturels.

Exercice 2. Écrire une fonction en Scilab qui prend en entrée un entier n et quatre réels a, b, x, y et qui renvoie la valeur du $n^{\text{ième}}$ terme de la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = x$, $u_1 = y$ et pour tout $n \in \mathbb{N}$, $u_{n+2} = au_{n+1} + bu_n$. *On utilisera la formule de récurrence et une boucle* `for`.

Tester cette fonction avec $(n, a, b, x, y) = (10, 6, 7, -2, -1)$ et comparer avec la valeur théorique donnée par une formule du cours.

Exercice 3. Implémenter en Scilab :

1) une fonction `somme.sci` qui prend en entrée un vecteur et qui calcule la somme de ses coordonnées.
2) une fonction `minmax.sci` qui prend en entrée un vecteur composé de nombres réels et qui renvoie en sortie la plus grande et la plus petite de ses coordonnées.

Scilab disposait déjà de fonctions permettant de faire cela : il s'agit de `sum`*,* `max` *et* `min`*.*

Exercice 4. Implémenter en Scilab les fonctions suivantes (de telle sorte qu'elles soient compatibles avec un traitement vectoriel).

$$f : x \longmapsto \frac{e^x - e^{-x}}{e^x + e^{-x}}, \qquad g : x \longmapsto \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right), \qquad h : x \longmapsto \begin{cases} 1 - |x| & \text{si} \quad x \in [-1, 1] \\ 0 & \text{si} \quad x \notin [-1, 1] \end{cases},$$

$$\phi : x \longmapsto \begin{cases} \dfrac{\sin(x)}{x} & \text{si} \quad x \neq 0 \\ 1 & \text{si} \quad x = 0 \end{cases} \qquad \text{et} \qquad \psi : x \longmapsto \begin{cases} \sin(x)\ln(-x) & \text{si} \quad x < 0 \\ 0 & \text{si} \quad x = 0 \\ x^{-3/2}(1 - \cos(2x)) & \text{si} \quad x > 0 \end{cases}.$$

Pour la fonction h, on pourra s'aider de la fonction `max` *pour éviter le recours à une boucle* `for`*.*

Exercice 5. Implémenter en Scilab la fonction f qui est 2-périodique sur $\mathbb{R}$ et qui vérifie

$$\forall x \in [-1, 1], \qquad f(x) = x\sqrt{1 - x^2}.$$

Exercice 6. Pour tout $n \in \mathbb{N}$, notons S_n la fonction définie par

$$\forall x \in \mathbb{R}, \qquad S_n(x) = \sum_{k=0}^{n} \frac{(-1)^k x^{2k}}{(2k)!}.$$

1) Écrire une fonction en Scilab nommée *sommecos.sci* qui prend en entrée $(x, n) \in \mathbb{R} \times \mathbb{N}$ et qui renvoie la valeur de $S_n(x)$.
   On rendra la fonction compatible avec un traitement vectoriel pour la première coordonnée. On n'utilisera pas le symbole $\wedge$*.*
2) Comparer les fonctions $\cos$ et S_n sur $[-k\pi, k\pi]$ pour différentes valeurs des entiers k et n.