TP 2

Révisions. Les fonctions en Scilab.

Compétences attendues.

✓ Définir une fonction en `Scilab`.

✓ Tracer la courbe représentative d'une fonction.

Mathieu Mansuy

Professeur de Mathématiques en deuxième année de CPGE filière ECS au Lycée Louis Pergaud (Besançon)

Page personnelle : `http://mathieu-mansuy.fr/`

E-mail : `mansuy.mathieu@hotmail.fr`

1 Les fonctions

Définition.

- **Fonctions usuelles :** `log` (logarithme népérien ln), `exp`, `cos`, `sin`, `tan`, `sqrt` (racine carrée), `floor` (partie entière) et `abs`.

- **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 $x \mapsto (1 + 3x + 2x^2 - x^4)e^{-x}$, taper dans un fichier `P.sce` :

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

Pour charger la fonction dans `Scilab`, taper la commande :

```
-->exec("P.sce")
```

On peut alors utiliser cette fonction dans la console comme s'il s'agissait d'une autre fonction `Scilab`. 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.

Mise en garde.

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

Exemple. Entrer la fonction suivante.

```
// 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 charger cette fonction. Exécuter 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, complexe, vecteur, matrice, etc...).
- Si la fonction ne nécessite pas de paramètre de sortie (la fonction ne fait que des actions), écrire `function []=nom_fonction(x1,x2,...,xn)`.
  Si la fonction ne nécessite pas de paramètre d'entrée, écrire `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.

`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(-1,3,6); 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 différentier par couleur en utilisant par exemple `plot2d(x,y,3)` ou `fplot2d(x,P,5)`

3 Exercices

Exercice 3.1 (★)

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 3.2 (★★)

1. Montrer que la série $\sum \frac{(-1)^n}{n^2}$ est convergente.
2. Créer un vecteur ligne `u`, tel que pour tout $1 \leq i \leq 50$, `u(i)` soit égal à $\frac{(-1)^i}{i^2}$. *On pourra utiliser pour cela une boucle for.*
3. Créer un vecteur `v` tel que pour $1 \leq i \leq 50$, `v(i)` soit égal à $\sum_{k=1}^{i} \frac{(-1)^k}{k^2}$.
4. Illustrer la convergence de la série en représentant graphiquement la suite de ses sommes partielles, et donner une valeur approchée à 10^{-1} près de $\sum_{n=1}^{+\infty} \frac{(-1)^n}{n^2}$.

Exercice 3.3 (★)

1. Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = 2nu_n + 3$. Écrire une fonction en `Scilab` ayant pour paramètre un entier $n \geq 0$ et qui renvoie la valeur de u_n.
2. Même question avec $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$, $u_1 = -2$ et pour tout $n \in \mathbb{N}$, $u_{n+2} = 2u_{n+1} - u_n$.

Exercice 3.4 (★)

Représenter sur une même figure les représentations graphiques des fonctions sin, $x \mapsto x$, $x \mapsto x - \frac{x^3}{3!}$, $x \mapsto x - \frac{x^3}{3!} + \frac{x^5}{5!}$ sur l'intervalle $[-\pi, \pi]$. Que remarque-t-on ?

Exercice 3.5 (Fonction réciproque - ★★)

1. On considère la fonction $f : \mathbb{R} \to \mathbb{R}$ définie par $f(x) = \frac{e^x - e^{-x}}{2}$ (appelée *sinus hyperbolique*, et souvent notée *sh*). 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 une ligne de 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 3.6 (★★ - Étude d'une suite récurrente)

1. Soit f la fonction définie pour tout $x \in \mathbb{R}_+^*$ par $f(x) = \frac{2}{x} + \ln x$.

   (a) Tracer le graphe de f ainsi que la droite d'équation $y = x$ à l'aide de `Scilab`, puis vérifier graphiquement que :

- l'intervalle $\left[\frac{3}{2}, 2\right]$ est stable par f ;
- il existe un unique $\alpha \in \left[\frac{3}{2}, 2\right]$ tel que $f(\alpha) = \alpha$, et en déterminer une valeur approchée.

(b) Démontrer l'ensemble des résultats observés à la question précédente.

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

(a) Montrer que la suite $(u_n)_{n\in\mathbb{N}}$ est bien définie et pour tout $n \in \mathbb{N}$, $u_n \in \left[\frac{3}{2}, 2\right]$.

(b) Écrire une fonction `u = suite(n)` ayant pour paramètre d'entrée $n \geq 0$ et qui renvoie la valeur de u_n.

(c) Écrire une fonction `[x,y] = escargot(n)` renvoyant le vecteur `x` des abscisses et celui `y` des ordonnées de la suite de points constituant le diagramme en escargot pour les termes $u_0, \ldots, u_n$ de la suite (u_n).

(d) Tracer, toujours sur la même fenêtre graphique que f, le diagramme en escargot de la suite (u_n) pour différentes valeurs de n, puis conjecturer le comportement asymptotique de (u_n).

3. On souhaite démontrer que la suite (u_n) converge vers α.

(a) Montrer que pour tout $(x, y) \in \left[\frac{3}{2}, 2\right]^2$, $|f(x) - f(y)| \leq \frac{2}{9}|x - y|$.

(b) Montrer que pour tout $n \in \mathbb{N}$, $|u_{n+1} - \alpha| \leq \frac{2}{9}|u_n - \alpha|$ puis que $|u_n - \alpha| \leq \left(\frac{2}{9}\right)^n |u_0 - \alpha|$.

(c) En déduire que la suite $(u_n)_{n\in\mathbb{N}}$ converge vers α.

4. Écrire une fonction `Scilab` qui donne une approximation de α à ε près pour $\varepsilon > 0$ donné.

   En déduire une valeur approchée de α à 10^{-3} près.

Exercice 3.7 (Fonction de répartition de la loi normale centrée réduite - ★★★)

Pour tout $x \in \mathbb{R}$, on note $\varphi(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right)$ la densité de probabilité de la loi normale centrée réduite $\mathcal{N}(0, 1)$, et Φ sa fonction de répartition, définie par :

$$\forall x \in \mathbb{R}, \quad \Phi(x) = \int_{-\infty}^{x} \varphi(t)\,\mathrm{d}t.$$

1. Créer la fonction φ en `Scilab` (on permettra que la variable d'entrée soit un vecteur). Représenter-la graphiquement.

2. (a) On rappelle que si f est une fonction continue sur $[a, b]$ alors on a :

$$\int_a^b f(t)\,\mathrm{d}t = \lim_{x\to+\infty} \frac{b-a}{n} \sum_{k=1}^{n} f\left(a + k\frac{b-a}{n}\right).$$

Soit $x \in [0, +\infty[$. Pour a « petit » et n « grand » avec $\frac{x-a}{n}$ « petit », on a alors :

$$\frac{x-a}{n} \sum_{k=1}^{n} \varphi\left(a + k\frac{x-a}{n}\right) \approx \int_{-\infty}^{x} \varphi(t)\,\mathrm{d}t = \Phi(x).$$

Écrire une fonction `Phi` en `Scilab` qui prend comme argument des réels x et a et un entier n, et renvoie une approximation de $\Phi(x)$.

(b) En utilisant la fonction `Phi`, tracer la représentation graphique de Φ (choisir a et n de façon « adéquate »).

3. On va vérifier graphiquement la qualité de notre représentation graphique. Pour cela, on utilise la commande `Scilab` suivante.

   `[P,Q]=cdfnor("PQ",x,m,sigma)` avec `x`, `m` et `sigma` vecteurs de même taille, donne $P = F(x)$ où F est la fonction de répartition de la loi $\mathcal{N}(m, \sigma^2)$, et $Q = 1 - P$.

   (a) Calculer $\Phi(0), \Phi(1), \Phi(1,96)$.

   (b) Tracer la fonction de répartition Φ de la loi normale centrée réduite à l'aide de la commande `cdfnor`. Comparer les deux courbes (faites varier les paramètres a et n).
