Simulation stochastique 1
Licence MIASHS - L3
Année 2015 - 2016
Jung Jonathan - Sophie Mercier

TP n° 1 : Introduction à Scilab

La première séance est dédiée à la prise en main du logiciel libre *Scilab* de calcul numérique.

1 Création et enregistrement d'un fichier *Scilab*

— Mettez-vous en binômes.
— Connectez-vous sous *Linux*.
— Créez dans votre espace personnel (répertoire /S) un dossier *SimuStochastique1* dans lequel vous mettrez tous vos fichiers *Scilab*.
— Ouvrir *Scilab*, allez dans *"Démarer SciNotes"* puis cliquez sur *Fichier/Nouveau*.
— Allez immédiatement dans *Fichier/Enregistrer sous* et sauvegardez le fichier vide sous *TP1-Intro-Scilab.sce* dans le dossier *SimuStochastique1*.

2 Quelques règles de base...

Pour effacer toutes les variables de la console, tapez

```
clear
```

Pour introduire une ligne de commentaire dans votre fichier, commencez la ligne par un // :

```
// Votre commentaire
```

Pour exécuter votre fichier *SciNotes*, appuyez sur l'icône "Exécuter".
Pour affecter à la variable a la valeur 2, tapez

```
a = 2
```

On notera que, lors de l'exécution de cette instruction, le contenu de la variable n'est pas affiché. Pour afficher la valeur de a, tapez dans la console

```
a
```

Pour réinitialiser toutes les variables précédemment utilisées, tapez

```
clear
```

Pour affecter à x le vecteur ligne $(1, 2, 3, 4, 5, 6)$, tapez

```
x=[1,2,3,4,5,6]
```

ou

```
x=1:1:6
```

car $a : p : b$ correspond au vecteur ligne commençant en a et s'arrêtant à b avec un pas de p.
Pour afficher le deuxième élément du vecteur x, tapez

```
x(2)
```

Pour afficher les éléments entre les positions 3 et 5, tapez

```
x(3:5)
```

Pour transposer la matrice x, tapez

```
x=x'
```

Pour accéder à l'**aide** d'une commande, il faut aller dans *?/Aide de Scilab*. Par exemple, pour accéder à l'aide de la commande *sum*, faites une recherche sur `sum`.

3 Quelques exercices

Exercice 1. Créer les vecteurs lignes suivants :

(a) $a = (0, 1, ..., 10)$,

(b) $b = (1, 3, ..., 11, 13)$ (le pas vaut 2),

(c) $c = (1, 1.1, 1.2, ..., 9.9, 10)$ (le pas vaut 0.1),

(d) $d = (1^2, ..., 20^2)$,

```
d=1:1:20                                    d=d.*d
```

(e) $e = (\ln(1), ..., \ln(15))$ (utiliser `log`)

(f) $f = (1 * \ln(1), ..., 15 * \ln(15))$,

(g) $g = \left(1^1, 2^2, ..., 9^9, 10^{10}\right)$,

(h) $h =$ vecteur ligne formé de 10 "0", puis de 10 "1" (utiliser `zeros` et `ones`)

(i) $i =$ vecteur colonne formé de $(1, 2, 3, 4, 5)$, puis de $(0, 2, ..., 12, 14)$ (pas de 2).

Exercice 2. Créer les matrices suivantes :

(a) $A =$ matrice d'ordre 6 dont toutes les lignes sont égales au vecteur des entiers de 1 à 6 (utiliser `repmat`)

(b) $B = (i + j)_{1 \leq i,j \leq 6}$,

(c) $C = \left(e^{i+j}\right)_{1 \leq i,j \leq 6}$ (utiliser `exp`),

(d) $D = e^A = \sum_{k=0}^{+\infty} \frac{A^k}{k!}$ (utiliser `expm`),

(e) $E =$ matrice diagonale d'ordre 6 dont la diagonale contient les entiers de 1 à 6 (utiliser `diag`),

(f) $F =$ matrice d'ordre 6 contenant les entiers de 1 à 36, rangés par colonne (utiliser `matrix`),

(g) $G =$ matrice d'ordre 6 contenant les entiers de 1 à 36, rangés par ligne (utiliser `matrix`),

Exercice 3. Soit $A = (i - j)_{1 \leq i,j \leq 5}$.

(a) Créer la matrice A,

(b) Calculer le vecteur ligne $b = (b_j)_{1 \leq j \leq 5}$ avec $b_j = \sum_{i=1}^{5} a_{i,j}$ (à l'aide de la commande `sum`),

(c) Calculer le vecteur colonne $c = (c_i)_{1 \leq i \leq 5}$ avec $c_i = \sum_{j=1}^{5} a_{i,j}$ (à l'aide de la commande `sum`),

(d) Calculer le vecteur ligne $d = (d_j)_{1 \leq j \leq 5}$ avec $d_j = \sum_{k=1}^{j} b_k$ (à l'aide de la commande `cumsum`),

(e) Calculer le vecteur ligne $e = (e_i)_{1 \leq i \leq 5}$ avec $e_i = \sum_{k=1}^{i} c_k$ (à l'aide de la commande `cumsum`),

(f) Calculer la matrice $f = (f_{i,j})_{1 \leq i,j \leq 5}$ avec $f_{i,j} = \sum_{k=1}^{i} a_{k,j}$ (à l'aide de la commande `cumsum`),

(g) Calculer la matrice $g = (g_{i,j})_{1 \leq i,j \leq 5}$ avec $g_{i,j} = \sum_{k=1}^{j} a_{i,k}$ (à l'aide de la commande `cumsum`).

Exercice 4. Écrire un programme Scilab qui calcule $S_n = \sum_{i=1}^{n} i^2$:

(a) avec une boucle `for`,

(b) sans boucle (à l'aide de la commande `sum`).

Comparer les temps d'éxécution des deux méthodes pour de grandes valeurs de n. (On utilisera la commande `timer()`).

Exercice 5. Écrire un programme Scilab qui calcule $S_n = \prod_{i=1}^{n} \left(1 + \frac{1}{i^2}\right)$:

(a) avec une boucle `for`,

(b) sans boucle (à l'aide de la commande `prod`).

Comparer les temps d'éxécution des deux méthodes pour de grandes valeurs de n.

Exercice 6. On pose $S_n = \sum_{i=1}^{n} ie^i$. Déterminer $n_0 = \min\left(n \geq 1 \text{ tel que } S_n \geq 10^5\right)$. (On utilisera une boucle `while` et on vérifiera le résultat obtenu).

Exercice 7. On pose

$$f(i) = \begin{cases} i+1 \text{ si } i \text{ est multiple de } 3 \\ i^2+1 \text{ sinon} \end{cases}$$

pour $i \in \mathbb{N}$. Ecrire un programme qui calcule et stocke les valeurs des $f(i)$ pour $0 \leq i \leq 20$,puis qui affiche le vecteur f. (On utilisera une boucle `for`, l'instruction `if then else` et la fonction `pmodulo`).

Exercice 8. Soient

$$f(x) = x \sin(x) \text{ et } g(x) = x \cos(x)$$

Tracer les fonctions f et g sur $[0, 10]$ sur un même graphique, avec un pas de temps égal à 0.1. Mettre les légendes f et g sur le graphique dans le coin supérieur gauche. (On utilisera `plot` et `legend`).

Exercice 9. On pose

$$f(x) = x \cos(x)$$

pour tout $x \in \mathbb{R}$. Créer une fonction scilab qui prend x en entrée et retourne $f(x)$. Calculer $(f(i))_{0 \leq i \leq 5}$.

Exercice 10. On pose

$$f(x, y) = x \cos(y)$$

pour tout $x, y \in \mathbb{R}$. Créer une fonction scilab qui prend (x, y) en entrée et retourne $f(x, y)$. Calculer $(f(i, j))_{0 \leq i,j \leq 5}$.

Exercice 11. Pour $a = [-1, -0.5, 0, 0.5]$, on pose

$$f_a(x) = x^a e^{-x}$$

pour tout $x \in \mathbb{R}$. Créer une fonction scilab et tracer $f_a(x)$ pour les différentes valeurs de a.

Exercice 12. Ecrire les fonctions suivantes, sans utiliser de boucle. Toutes prennent en entrée un vecteur colonne $v = (v_i)$ et un vecteur ligne $w = (w_j)$ et retourne en sortie une matrice $A = (a_{i,j})$ qui a autant de lignes que v et de colonne que w.

(a) `matV` : $a_{i,j} = v_i$,

(b) `matW` : $a_{i,j} = w_j$,

(c) `produit` : $a_{i,j} = v_i * w_j$,

(d) `echiquier` : $a_{i,j} = v_i$ si $i + j$ est pair, w_j sinon.