LM206 INITIATION À SCILAB - ÉLÉMENTS DE PROGRAMMATION

1 Environnement de travail

Une fois lancé (en cliquant sur une icône ou par la commande `scilab`), le logiciel se présente sous la forme d'une fenêtre de commande, avec une barre de menu en environnement graphique.

```
 ---------------------------------------
                 Scilab-4.0

           Copyright (c) 1989-2006
                 (INRIA, ENPC)
 ---------------------------------------


Startup execution:
  loading initial environment

-->
```

On peut alors entrer au fur et à mesure des commandes (instructions) après la flèche `-->` (prompt, ou invite), et obtenir en réponse le résultat de celles-ci.

Exemple 1

```
2**(1/2)
sqrt(2)
(1-%i)^2
1/3
format ('v',16); 1/3
sin(%pi/4)
```

Notes :

- on peut se déplacer dans l'historique des commandes avec les flèches ↑ ↓ ;
- les espaces ne sont pas "lus" par SCILAB ; libre à vous d'en insérer ou non, afin de garantir une bonne lisibilité du code ;
- `ans`, pour *answer*, réponse, est le résultat de votre dernière commande ; vous pouvez le réutiliser dans la commande suivante ;
- une commande qui se termine par un point-virgule (`;`) est traitée, mais son résultat n'est pas affiché ;

– la virgule (`,`) permet d'enchainer plusieurs commandes sur la même ligne ;
– une commande trop longue pour tenir sur une ligne peur se terminer par trois points (`...`) et reprendre sur la ligne suivante ;
– `**` et ^ sont équivalents.

1.1 Documentation

Il est essentiel d'utiliser intensément la documentation incluse ans le logiciel. Elle est accessible à partir de la commande `help` (ou par le bouton "Help" du menu[1]). Vous pouvez alors l'explorer selon le type d'information que vous recherchez, ou bien accéder directement à la documentation d'une commande précise (`help who` ou `help format` par exemple). On peut aussi chercher des mots-clés, avec `apropos` (`apropos absolute`, `apropos rounding`). De nombreux programmes de démonstration sont aussi accessibles par le menu "Demos".

Exercice 1

- *Cherchez dans l'aide en ligne la documentation sur `format`, `ieee`, `exec`, `symbols`, `keyboard` et ce que vous voudrez.*
- *Explorez les démonstrations des sections "simulations" et "graphics".*

1.2 Variables

Comme tout logiciel numérique, SCILAB fait des calculs approchés, donc ...faux. La précision, qui dépend de la machine, est enregistrée dans `%eps`, le plus grand ε tel que $1 + \varepsilon/2 = 1$! D'autres valeurs sont préenregistrées, comme `%e`, `%pi`, `%i`.

Exemple 2

```
%eps
sqrt(-1)
(-1)**(1/2)
X = 1E30; Y=1E10;
X + Y - X
X - X + Y
```

Remarquez que le resultat obtenu est parfois un nombre tres petit (inférieur à ε) là où on attend 0. De même l'ordre des opérations est important si ces opérations mettent en jeu des nombres de grandeurs très différentes.

Vous pouvez vous aussi définir vous-même ces variables, ie des données, dans la mémoire de l'ordinateur, auxquelles vous associez un nom. `who` vous indique la liste des variables en mémoire (y compris les variables internes automatiquement créées par SCILAB) et la place disponible en mémoire, et `clear` réinitialise la mémoire en supprimant toutes les variables.

[1] Le détail du menu dépend de la version de SCILAB.

Exemple 3

```
a = 5
b = a + 7
a = 0
b
blablabla = 'une longue ligne de texte'
who
```

1.3 Scripts

Lorsque vous quittez puis reprenez votre travail, ou lorque vous élaborez une séquence d'instructions, il est utile de les sauvegarder. Vous pouvez écrire les instructions dans un fichier texte, à l'aide de l'éditeur de texte de votre choix (`scipad` par exemple), puis exécuter ces instructions à l'aide de la commande `exec` depuis SCILAB, obtenir les résultats ou les erreurs, corriger, améliorer, recommencer.

Exercice 2

- *Créez un dossier sur l'ordinateur, dans lequel vous créez un fichier `2deg.sci` contenant des instructions pour calculer les racines d'un polynome du second degré. Les paramêtres sont donnés dans ce fichier, et on n'a bas desoin de commandes interactives pour les définir. Testez, corrigez, testez...*
- *Puis obtenez le module m de $3 + 5i$, et calculez la surface du disque de rayon m.*

Plus tard, l'utilisation de fonctions nous permettra d'écrire des instructions génériques, utilisables pour plusieurs valeurs des paramêtres.

2 Vecteurs et matrices

Les matrices sont esentielles dans SCILAB. En fait, tout (ou presque) y est un tableau de nombres. Une matrice 5×5 est un tableau à 5 lignes et 5 colonnes, un vecteur est un tableau à une seule colonne, et une simple nombre est en fait un tableau à une ligne et une colonne. Le format essentiel de données est le tableau de nombre à virgule.

De nombreuses syntaxes sont possibles pour créer, manipuler, modifier ces matrices. Ces syntaxes doivent être exploitées au mieux pour avoir des instructions efficaces, car il est beaucoup plus rapide dáppliquer une fonction à un tableau de 100 données que 100 fois une fonction à une seule donnée.

2.1 Création et opérations simples

Il existe de multiples façons de créer ces matrices et vecteurs.

– Création explicite, par la liste des éléments. On les sépare par une virgule `,` ; un point-virgule `;` indique un passage à la ligne suivante.

Exemple 4

```
a = [1,3,6,2,-3,4]
b = [1;3;6;2;-3;4]
```

```
c = [1,3,6;2,-3,4]
x = 5
y = [5]
z = []
```

- Utilisation des constructeurs élémentaires `ones`, `zeros`, `eye`, `rand`
- Utilisation de l'opérateur `:`, opérateur de progression arithmétique. `a:b` signifie $a, a+1, a+2, ...b$. `a:i:b` signifie $a, a+i, a+2i, ...b$.

Exemple 5

```
a = [1:10]
b = [1:5:13]
c = [9:-4:-20]
c = [0:%pi/8:%pi]
```

- On peut aussi crér des matrices puis les transformer par des fonctions, ou des opérateurs tels que `+`, `-`, `*`, `/`, etc.

Exemple 6

```
c = [0:%pi/8:%pi]
c = sin(c)
```

- On peut enfin les concaténer, les assembler.

Exemple 7

```
a0 = zeros(1, 5)
a1 = ones(1, 5)
a2 = 2 * ones(1, 5)
a = [a0;a1;a2]
```

2.2 Extraction, assignement

Une fois une matrice créée, on peut accéder à ses éléments en indiquant les indices entre parenthèses.

Exemple 8

```
a = [0:10:100]
b = [a;2*a;3*a;4*a]
a(3)
b(2,4), b(4,2)
```

On peut aussi utiliser dans ces indices l'opérateur arithmétique `:`. Le signe dollar `$` est aussi un raccourci pratique ; il correspond toujours au plus grand indice possible.

Exemple 9

```
a($), a($-1), a($-2)
b(2,1:$)
b(1:2:$,1:2:$)
```

Ces indices permettent aussi de modifier certains éléments de la matrice, en y assignant une valeur. On peut notamment modifier *plusieurs éléments d'un tableau en une seule instruction*

Exemple 10

```
b(2,1:$) = 0
c = [1;2] * [0:5]
b(1:2:$,1:2:$) = c
```

Exercice 3

- *Expliquez les différences entre les opérateurs* `*`, `.*` *et* `.*.`.
- *Construisez de deux manières différentes le vecteur contenant les* n *premiers carrés des nombres entiers.*
- *Construisez de deux manières différentes une matrice de taille* 12×5 *ne contenant que des zéros, sauf sur les premières et dernières lignes et colonnes qui contiennent des* 1.

2.3 Transformations élémentaires

Les tableaux et matrices bénéficient de nombreuses fonction implémentéees dans la langage de programmation de SCILAB, dont l'extraction des matrices triangulaires inférieures ou supérieures (`triu`, `tril`), de la diagonale (`diag`), ou bien la transposition (opérateur `'`).

Dans la liste des manipulation simples on peut aussi citer `matrix`, pour modifier les dimensions d'un vecteur ou d'une matrice, `size` et `length` pour obtenir ces dimensions et le nombre total d'éléments, ou `cumprod` et `cumsum` pour calculer le produit ou la somme des éléments d'une tableau, avec les produits ou sommes intermédiaires. Pensez aussi aux classiques `max` et `min`. Enfin, `sort` trie le tableau, `unique` n'en conserve que les éléments uniques, et `find` permet d'obtenir les indices des éléments d'un tableau correspondant à une certaine condition.

Exercice 4

- *Générez un tableau de départ* a*, de 10 valeurs aléatoires entre* -1 *et* 1*, dont vous calculez la moyenne.*
- *Comptez le nombre d'éléments positifs/négatifs de* a.
- *Produire plusieurs tableaux d'analyse de* a *:*
  - *un tableau* b *contenant les différences entre deux éléments successifs de* a *;*
  - *un tableau contenant les indices* i *tels que* $a_{i+1} > a_i$ *;*
  - *un tableau* c *de taille* 10×10 *tel que* $c_(i, j) = 1$ *si si* $a_i > a_j$, 0 *sinon (commencez par détailler ce que contiendra* c*, élément par élément) ;*
- *Visualisez* c *avec la fonction* `Matplot` *(il sera nécessaire de modifier* c *pour obtenir un résultat visuellement intéressant).*
- *Refaites ces opérations après avoir préalablement ordonné* a *(et recalculé* b *et* c*).*

On complète cette liste des outils standard par les fonctions d'algèbre linéaire comme `det`, `spec` et `rank` pour le déterminant, les valeurs propres et le rang d'une matrice.

De plus, `x = A \ b` et `linsolve(A, b)` calculent respectivement les solutions de $A \times x = b$ et $A \times x + b = 0$ (dans ces opérations, `1` est interprété comme la matrice unité, ce qui permet d'écrire `x = 1/A`) et `inv(A)` est la matrice inverse de `A`.

Exercice 5
À partir de matrices `A` et `b` de votre choix, vérifiez le résultat des fonctions indiquées ci-dessus. Résolvez le système de trois équations $3x - 2y - z = 7$, $5x + y - 3z = 2$, $-2x - y = 5$ (intersection de trois plans dans l'espace, par exemple).

Exercice 6
Avec `linsolve`, déterminez une base du plan dans $\mathbb{R}^3$ vérifiant $x-2y = 0$, puis un point et un vecteur de la droite dans $\mathbb{R}^4$ vérifiant $x-2y = 0, x+y+z-1 = 0, 3x-y+z-t = 0$.

Exercice 7
À l'aide de `timer`, comparez le temps d'execution de ces trois fonctions, ainsi que le calcul du déterminant et des valeurs propres et ordonnez les par rapidité. En approximation grossière, lorsqu'on double la taille d'une matrice, est-ce que ces temps de calcul sont multipliés par 2, 4, 8 ?

3 Fonctions et tests

Le mot-clé `function` permet de définir nos propres fonctions, aisément utilisables ensuite. Ces fonctions peuvent être définies en ligne de commande, ou dans un script sur un fichier. Elles prennent un ou plusieurs paramètres, et peuvent renvoyer un ou plusieurs résultats comme dans l'exemple suivant :

Exemple 11
```
function [x,y]=truc(a,b)
  x=a+b
  y=a-b
endfunction
```

Exercice 8
Avec des tests (`if`, `then`, `else`, `end`), construisez une fonction calculant la factorielle d'un nombre entier.

Exercice 9
Construisez une fonction calculant les nombres de Fibonacci $f_i = f_{i-1} + f_{i-2}$, avec $f_0 = f_1 = 0$.

Notes :
- on teste l'égalité de deux valeurs par `if x == y then ...` ;
- pour faire un test sur deux conditions, on utilise
  - `|` pour un test *condition1 OU condition2*,
  - `&` pour un test *condition1 ET condition2*,

  par exemple sous la forme `if x == y | x > z then ...`.

4 Graphiques

4.1 2D

L'outil principal pour tracer des courbes, dans un repère de $\mathbb{R}^2$, est `plot2d()`. Si `x` et `y` sont des vecteurs de même dimension, `plot2d(x, y)` trace la courbe (polygonale) reliant les points (x_i, y_i). Ainsi, pour tracer le graphe de cos entre 0 et π, avec un espacement des points de 0.1 en abscisse, on utilisera `[0:0.1:%pi]` pour `x` et `cos([0:0.1:%pi])` pour `y`. Les tracés de plusieurs commandes `plot2d` se superposent ; il faut donc utiliser `clf` pour effacer la fenêtre graphique.

Exemple 12

```
plot2d([0:0.1:%pi], cos([0:0.1:%pi]))
clf
function [y] = f(x); y = x^3-2*x; endfunction
X = [-2:0.1:2]
plot2d(X, f(X))
```

On peut aussi tracer plusieurs courbes à la fois, par exemple avec `plot2d(x, y)` où `x` est un vecteur des abscisses, et `y` est une matrice dont chaque colonne correspond aux ordonnées pour une courbe. `help plot2d` vous indiquera toutes les possibilités, dont certaines sont présentées ci-après :

Exemple 13

```
plot2d([1, 3, -5, 2, 4, 7])
clf
X = [-%pi:%pi/32:%pi]'
plot2d(X, [cos(X), cos(2 * X), cos(3 * X)])
```

`plot2d` accepte des options complémentaires, permettant d'adapter le résultat graphique aux information ou paramètres souhaités, sous la forme `option=valeur` :

- `style=...` permet de choisir la couleur ou le format de tracé pour chacune des courbes, en indiquant un vecteur dont chaque element correspond à une couleur ;
- `rect=[xmin, ymin, xmax, ymax]` permet de spécifier les intervalles d'abscisses et ordonnées sur lesquelles le graphes est tracé ;
- `axesflag=...` permet de décider de la manière dont les axes figurent sur l'image
- `leg=...` indique la légende des diverses courbes.

Toutes ces options sont détaillées dans l'aide, qui propose également des exemples d'application.

Il existe quelques variantes de `plot2d`, à utiliser pour des besoins spécifiques :

- plot`plot2d2` trace des graphes constants par morceaux (ie "en escalier") ;
- plot`plot2d2` trace des graphes par lignes verticales ;
- plot`plot2d4` trace des graphes composés de flèches ;
- enfin, `histplot` permet de tracer des histogrammes.

Exemple 14

```
X = [-%pi:%pi/32:%pi]'
```

```
clf; plot2d2(X, [cos(X), cos(2 * X), cos(3 * X)])
clf; plot2d3(X, [cos(X), cos(2 * X), cos(3 * X)])
clf; plot2d4(X, [cos(X), cos(2 * X), cos(3 * X)])
```

Exercice 10
Tracez la courbe $y = cos(x)$, puis
- $y = 1$
- $y = 1 - x^2/4$
- $y = 1 - x^2/4 + x^4/24$
- $y = 1 - x^2/4 + x^4/24 - x^6/720$

en spécifiant correctement les couleurs, intervalles et labels, de manière à y observer la convergence de $\sum_{n=1}^{k}(-1)^n \frac{x^{2n}}{2n!}$ vers $cos(x)$.

Exercice 11
Déterminez une approximation graphique de la solution de $cos(x) = x$, en utilisant `plot2d` et les possibilités de zoom.

Exercice 12
Écrivez une fonction utilisant `rand` et `int` pour simuler le tirage aléatoire d'un dé à six faces. Utilisez ensuite une boucle `for` pour effectuer un grand nombre de tirages, puis regroupez ces tirages par trois. Affichez alors avec `histplot` les probabilites d'obtenir un total donné entre 3 et 18 dans le cadre de votre simulation.

4.2 3D

De même que `plot2d` permet de tracer des courbes correspondant à des fonctions de $\mathbb{R}$ dans $\mathbb{R}$, `plot3d` permet de tracer des surfaces correspondant à des fonctions de $\mathbb{R}^2$ dans $\mathbb{R}$. Ici, `plot3d(x, y, z, options)` trace la surface liant les points $(x_i, y_j, z_{i,j})$. Les options sont semblables à celles de `plot2d`.

Exercice 13
À titre d'exercice, on peut tracer la surface d'équation $z = f(x, y) = cos(x)sin(y)$.