Le langage Scilab

I. Les objets du langage(constantes ou variables)

1) Les types élémentaires

• Les nombres entiers (signés ou non, codés sur 1 octet, 2 octets ou 4 octets).
Par exemple, un entier codé sur 2 octets et signé est un entier appartenant à l'intervalle des entiers [-32768;32767]. Lorsqu'il n'est pas signé, il sera dans l'intervalle [0;65535].
• Les nombres réels et complexes
• Les booléens
Scilab possède un type logique de booléens %t et %T pour vrai (True) et %f et %F pour faux (False).
• Les polynômes
• Les chaînes de caractères

2) Les types structurés

• Les listes
• Les tableaux (ou matrices)

II. Les actions élémentaires

1) L'instruction

L'instruction la plus simple est celle qui comporte une expression unique. Cette expression doit se terminer par un passage à la ligne, une virgule (séparateur d'instructions), trois points suivis d'un passage à la ligne (permettant de continuer l'expression sur la ligne suivante) ou un point virgule (dans ce dernier cas, le résultat de l'expression n'est pas affiché sur l'écran).

1) L'affectation

L'affectation est l'instruction qui permet de donner à une variable une valeur qui est le résultat d'un calcul. L'opération d'affectation est représentée par le symbole =. La forme la plus simple est *nom = expression*.

2) L'appel d'une fonction définie par l'utilisateur.

III. Les opérateurs

1) Les opérateurs numériques

Opérateurs unaires : identité (+), opposé (-)
Opérateurs arithmétiques : addition (+), soustraction (-), multiplication (*), division (/), division inversée (\)

2) Les opérateurs de comparaison

Egalité = =, non égalité <> ou ~ =, inégalités strictes < et >, inégalités larges <= et >=.
Le résultat d'une comparaison est un booléen.

3) Opérateurs logiques

La négation ~, la conjonction &, la disjonction |

P	~ P
T	F
F	T

&	T	F
T	T	F
F	F	F

|	T	F
T	T	T
F	T	F

où T : True et F : False

4) Ordre de priorité

Une expression peut parfois devenir relativement complexe, du fait de la présence de multiples opérateurs au sein de l'expression. Le langage Scilab a déterminé un ordre de priorité par défaut :

Priorité	Opérateurs
1. (La plus haute)	~(non) ^
2.	&(et) * /
3.	|(ou) + -
4. (la plus basse)	== ~> < <= > >=

IV. Les actions composées

1) Le bloc d'instructions (ou séquence d'instructions)

Un bloc d'instructions se compose d'une ou plusieurs instructions placées les unes après les autres (et séparées par des virgules ou des points-virgules) suivant que l'on souhaite un affichage ou non.

Dans la suite une instruction pourra représenter une instruction simple ou un bloc d'instructions.

2) Les conditionnelles

L'instruction *if* permet l'exécution conditionnelle de séquences d'instructions. Sa syntaxe la plus simple est : *if expression then instructions end.*

Lorsque l'expression *expression* est vraie, la séquence d'instructions *instructions* sera exécutée.

```
-->a=2 ; if a= =2 then 3+4, end
  ans = 7.
-->a=5 ; if a= =2 then 3+4, end
-->
```

Dans le deuxième exemple, le résultat du test est faux ; l'addition n'est donc pas exécutée et il n'y a pas d'impression du résultat.

Il est possible de préciser une instruction à exécuter lorsque le test est faux en utilisant le mot-clef *else*. La syntaxe de l'expression est alors la suivante :

if expression then instructions1 else instruction2 end

Lorsque l'expression *expression* est vraie *instruction1* est réalisée sinon c'est *instruction2*.

La conditionnelle multiple

La conditionnelle multiple ou instruction de sélection permet d'envisager plusieurs résultat à une expression :

```
select expression0
  case expression1 then instruction1
  case expression2 then instruction2
  case expression3 then instruction3
  else instruction4
end
```

La valeur d'*expression0* est calculée puis comparée successivement aux valeurs d'*expression1*, *expression2*, *expression3*, … Dès que l'une de ces comparaisons rend vrai, l'action correspondante s'effectue et les autres expressions ne sont plus testées. Si aucune de ces comparaisons ne retourne vrai, l'action introduite par le mot-clef else est exécutée.

On peut toujours traduire les conditionnelles multiples par des imbrications de conditionnelles :

```
if expression0= =expression1 then instruction1
  elseif expression0= =expression2 then instruction2
  elseif expression0= =expression3 then instruction3
  else expression4
end
```

3) Les boucles (ou instructions répétitives)

Elles ont pour but d'exécutée plusieurs fois une séquence d'instructions.

L'instruction ***for***

La boucle for permet de répéter une instruction alors qu'une variable particulière, la variable de boucle, décrit un ensemble de valeurs.

for variable=valeurs do instruction end

Valeurs est la liste des valeurs que va décrire la variable de boucle *variable*.

D'une manière générale, les valeurs sont définies par une valeur de début, une valeur de fin et d'un pas de progression (ou d'incrémentation) entre ces bornes.

Le mot-clef do peut être remplacé par un passage à la ligne, une virgule ou un point virgule.

Notons encore que la variable de contrôle est une variable locale et n'a pas d'existence à l'extérieur de la boucle.

L'instruction **while**

La boucle while va répéter une instruction tant qu'une condition est réalisée.

while expression do instruction end

Il faut dans ce cas que expression fournisse un booléen (donnant le résultat du test).

L'instruction while permet une autre forme dont la syntaxe est la suivante :

while expression do instruction1 else instruction2 end

Cette autre écriture permet de préciser une instruction (ou bloc d'instructions) qui sera exécutée à la fin de la boucle while lorsque la condition testée sera de venue fausse.

L'instruction **break**

Utilisée à l'intérieur de la boucle for ou while, l'instruction break permet d'interrompre la boucle et de sauter immédiatement à l'instruction qui suit cette boucle dans le texte du programme en cours d'exécution.

4) Identificateurs

Scilab permet de définir des variables, qui peuvent être locales à une procédure, résider dans l'espace de travail, ou encore être globales.

Un identificateur est représenté par une suite de caractères, dont le premier est une lettre ou l'un des six caractères % _ # ! $? ; le reste du nom peut être composé de ces caractères, de lettres ou de chiffres. Scilab ne prend en compte que les 24 premiers caractères d'un identificateur.

5) Variables et constantes prédéfinies

Scilab met à la disposition de l'utilisateur quelques variables et constantes prédéfinies :

• Inf dénote ∞. Il peut être utilisé dans certaines opérations :

```
--> 1/%inf
ans =
  0
```

• Nan (Not A Number) est le résultat typiquement renvoyé par ∞/∞, par exemple.

• Pi (obtenu par %pi) dont la valeur vaut approximativement pi.
• i (%i) le nombre complexe tel que $i^2 = -1$
• %T et %F sont les variables bouléennes signifiant respectivement vrai (pour true) et faux pour (false).
• La tolérance de "zéro machine" est fixée par la variable standart eps. C'est-à-dire que pour certaines opérations telles que la recherche de zéros de fonctions, par exemple, les nombres x tels que -eps < x < eps seront considérés comme nuls.
En modifiant cette borne, vous pouvez renforcer ou abaisser la sévérité des tests de nullité. La valeur standard vaut :

```
-->%eps
%eps =
   2.220E-16
```

V. Fonctions mathématiques prédéfinies

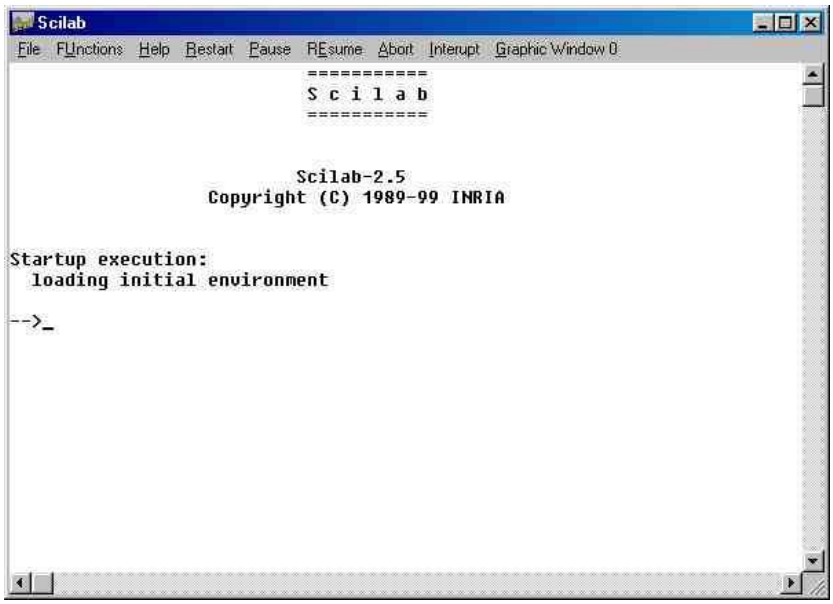
Opération	description
abs(x)	valeur absolue
atan(x)	arc tangente
ceil(x)	partie entière, arrondie vers +∞
conj(x)	conjugué de x
cos(x)	cosinus
disp(x,y,…)	affichage des valeurs de paramètres x, y …
exp(x)	exponentielle
fix(x)	partie entière, arrondie vers 0
floor(x)	partie entière, arrondie vers -∞
format(n)	sélection de n chiffres significatifs pou l'affichage des nombres
imag(x)	partie imaginaire de x
int(x)	partie entière, arrondie vers 0 (comme fix)
log(x)	logarithme népérien
modulo(x,y)	reste de la division de x par y
rand()	nombre aléatoire compris entre 0 et 1
real(x)	partie réelle de x
round(x)	partie entière, arrondie vers l'entier le plu proche
sin(x)	sinus
sqrt(x)	racine carrée
string(x)	chaîne de caractères représentant l paramètre
tan(x)	tangente

VI Principales commandes de Scilab

Help	Aide en ligne : sans paramètre, ouvre une fenêtre décrivant la commande help. Suivi du nom d'une fonction du langage, ouvre une fenêtre décrivant cette fonction
apropos	Aide en ligne : suivi d'un mot ou d'une chaîne de caractères, ouvre une fenêtre contenant un bref résumé des aides en ligne contenant ces mots.
what	liste des mots-clefs du langage
who	liste des variables définies
clear	Avec des paramètres, efface les variables correspondantes
global	déclaration de variables globales
clearglobal	effacement de variable globale
edit	Appel d'un éditeur de texte (créer ou modifier des fichiers de script)
quit	Arrêt de Scilab

Débuter avec Scilab

Lancement du programme
Après le lancement du programme, une fenêtre de dialogue apparaît :



La suite de caractère "-->" imprimée par le logiciel est le prompt. Elle signale que Scilab est en attente d'une commande de l'utilisateur. Scilab est un interprète (c'est-à-dire que chaque commande introduite par l'utilisateur est immédiatement exécutée à la différence d'un langage compilé - comme Turbo-Pascal - dans lequel le programme complet est analysé, transformé en un langage directement utilisable par la machine).

Premier exercice
Tapez a = 2 + 3 puis la touche entrée :
--> a = 2 + 3
a =
5.
La réponse du système consiste en la ligne a = suivie de la valeur de l'expression introduite. Le résultat sera alors conservée dans la variable de nom a.

On peut se contenter d'écrire :
--> 2 + 3
ans =
5.
Dans ce cas, l'expression a été calculée mais le résultat est affecté, par défaut, dans la variable nommée ans (pour answer). Cette dernière pourra être utilisée, tout comme la variable a dans le premier cas, ultérieurement dans toute expression :
--> ans + 3
ans =
8.

Par défaut, Scilab imprime la valeur calculée. Il est possible de supprimer cette impression en faisant suivre l'expression par un point virgule. Ce caractère permet également d'introduire plusieurs expressions sur une ligne.
Vous pouvez alors expliquer la ligne suivante :

```
--> 2 + 3 ; ans + 4 ; a = ans + 10
a =
   19.
```

La virgule peut servir de séparateur d'expressions. Les résultats des différentes expressions sont alors imprimées :

```
--> a = 5 , b = 3
a =
   5.
b =
   3.
```

Il est possible de saisir une commande sur plusieurs lignes, à condition d'utiliser 3 points (…) comme caractères de suite. Par exemple :

```
--> 1/2 + 1/3 + 1/4 …
--> 1/5 + 1/6
ans =
   1.45
```

Utilisation de l'aide en ligne
Elle s'obtient en cliquant sur le bouton help de la fenêtre Scilab.

En utilisant la commande Help Dialog, une nouvelle fenêtre nommée Scilab Help apparaît. Le partie du milieu de cette nouvelle fenêtre correspond au classement de toutes les fonctions en un certain nombre de rubriques alors que la partie du haut donne la liste de toutes les fonctions de la rubrique choisie au milieu. Pour obtenir le détail d'une fonction, il suffit de cliquer sur le bouton show. Il apparaît alors une nouvelle fenêtre qui donne les détails voulus.

La commande Topic du menu Help permet de demander des détails en donnant directement des mots clefs. (ce qui correspond à la commande A propos de la fenêtre Scilab Help).

Enfin, vous pouvez, à l'invite du prompt, taper help suivi du sujet sur lequel vous désirez des détails. Si le mot clef donné existe dans sa liste, il vous donnera la fiche correspondante dans une nouvelle fenêtre.

Fermeture de Scilab
La session peut se terminer en fermant la fenêtre comme toute fenêtre sous Windows ou en tapant la commande :

```
--> exit
```

Premiers exercices sous Scilab

Ces exercices ont pour but de vous permettre d'utiliser Scilab comme une calculette, en même temps qu'ils vous incitent à consulter la documentation en ligne.

Respectez l'ordre des exercices.

1) -2^3+9

2) 3/4*5-5^2*2-3

3) ans

4) // ans

5) Quelle est la différence entre round, ceil et floor ?
x=sqrt(12),round(x),floor(x),ceil(x)
y= sqrt(13),round(y),floor(y),ceil(y)

6) A=12;a=13;A,a

7) Quelle est la surface d'un disque de 12 cm de diamètre ?

8) Calculez N=1/3. Quelle valeur s'affiche pour N ?
Recopiez cette valeur, multipliez-la par 3. Quel est le résultat ?
Calculez maintenant N*3 ; qu'en déduisez-vous ?
Utilisez la commande format pour modifier le format d'affichage.

9) Soient les nombres X=1E30, Y=1E10.
Calculez X+Y-X puis X-X+Y. Qu'en déduisez-vous ?
Si l'on choisit Y=1, qu'elle est la valeur minimale de X pour laquelle le phénomène se produit ?

10) Chercher dans la documentation comment tracer le graphe de la fonction sinus entre 0 et π

11) Donner un nombre aléatoire entre 0 et 1.
Simuler le tirage d'un dé à six faces.

12) x=2;y=3;y=x;x=y;
x,y
Que se passe-t-il ?
D'une manière plus générale, comment échanger les valeurs de deux identificateurs sans avoir à connaître ces valeurs (*ie* sans manipuler ces nombres) ?

13) Soit C=3+4*%i ; comment obtenir les parties réelles et imaginaires de ce nombre ? Comment en calculer ρ et θ, c'est-à-dire le module et l'argument ?
Réciproquement, étant données ces deux dernières valeurs, comment obtenir le nombre complexe correspondant ?

La gestion des fichiers sous Scilab : Partie 1

Lorsque vous travaillez sous Scilab, vous pouvez :

a) Imprimer pour avoir une première trace de l'ensemble de vos instructions

b) Sauvegarder votre travail avec l'instruction *save*.
Mais avant cela, vous devez préparer votre enregistrement :

Il faut, à chaque début de séance, indiquer quel est votre répertoire de travail, vous devez pour cela le donner grâce à la commande *Change Directory* du menu *File*. Par exemple, indiquez, votre chemin sous la forme : C:\Mes documents\Scilab\Jeudi Matin

Après un exercice ou une session quelconque, vous pouvez enregistrer votre travail. La fenêtre qui apparaît ne vous demande que l'identificateur du fichier. Il ajoute lui-même l'extension du fichier *.bin

Vous pouvez alors demander à reprendre ce fichier. Si la fenêtre ne voit pas ce fichier, vous pouvez demander le *type* tout fichier (*All files*) et vous pourrez alors le désigner. Vous remarquerez qu'il ne se passe rien ; le chargement du fichier s'est effectué pour reprendre les variables et leur contenu et non pas toutes les instructions tapées. Ceci permet de reprendre un calcul en conservant les valeurs déjà affectées.

La gestion de la mémoire
Il suffit de taper la commande who

```
-->who
your variables are...

 b          a          startup    ierr       MODE_X     scicos_pal
 %helps     MSDOS      home       PWD        TMPDIR     percentlib
 fraclablib            soundlib   xdesslib   utillib    tdcslib    siglib
 s2flib     roblib     optlib     metalib    elemlib    commlib    polylib
 autolib    armalib    alglib     intlib     mtlblib    WSCI       SCI
 %F         %T         %z         %s         %nan       %inf       old
 newstacksize          $          %t         %f         %eps       %io
 %i         %e
 using       5028 elements  out of    1000000.
          and          48 variables out of       1071
```

pour faire apparaître les variables que vous avez entrées dans l'ordre inverse de leur création ou modification. Cette liste est utile lorsque l'on souhaite connaître la liste des identificateurs utilisés (par exemple, au moment de créer une nouvelle variable).

Il faut passer par un éditeur pour sauvegarder l'ensemble des instructions.

c) Utilisation d'un éditeur de texte :
Ouvrez le *bloc-note* de Windows (il se trouve dans la rubrique
Démarrer/Programmes/Accessoires/.
Vous pouvez alors taper l'ensemble de vos instructions. Vous devez enregistrer votre texte en un fichier du type *.sce. Il faut pour cela demander un enregistrement avec le type *All files* puis à spécifier le nom et l'extension comme, par exemple, test.sce
(dans la pratique, il suffit de suffixer en *.txt)

Attention, n'oubliez pas d'enregistrer votre travail dans le bon répertoire, de préférence dans celui choisi dans Scilab. De plus, la dernière ligne du fichier doit obligatoirement se terminer par un retour-chariot pour être prise en compte.

Vous devez retourner dans la fenêtre de travail de Scilab puis à demander l'exécution de ce programme par l'instruction *exec* dans le menu *file*. Il vous suffit de préciser le fichier. Si le fichier n'apparaît pas immédiatement, comme pour la commande load, vous devez demander la lecture des fichiers de tout type puis à parcourir le contenu du répertoire pour trouver le bon fichier (ici le fichier test.sce).

Cette gestion des fichiers a le double avantage de sauvegarder l'ensemble de vos instructions et celui d'une gestion plus globale de votre programme.

TD : Algorithmes sur les nombres

1) La suite (u_n) est définie par $u_0 = 0$ et $u_{n+1} = \frac{1}{2}u_n + 3$. Faire afficher les termes u_1 à u_n, n étant saisi au clavier.

2) Conjecture Syracuse (ou de Kollek, ou de Collatz) :
La suite (u_n) est définie par l'entier naturel u_0 et par la relation suivante :

si u_n est impair $u_{n+1} = 3u_n + 1$ et si u_n est pair $u_{n+1} = \frac{u_n}{2}$, u_0 étant saisi au clavier.

Faire afficher tous les termes de la suite jusqu'à ce que l'un d'eux soit égal à 1.

3) Algorithme Babylonien ou méthode de Newton pour calculer $\sqrt{a}$

En partant de a et u_0 saisis au clavier, avec la formule de récurrence $u_{n+1} = \frac{1}{2}(u_n + \frac{a}{u_n})$, donner le premier terme de la suite vérifiant : $\left|\frac{u_{n+1} - u_n}{u_n}\right| < \text{eps}=10^{-5}$

4) La suite de Fibonacci est définie par $u_0 = u_1 = 1$ et par la relation : $u_{n+2} = u_{n+1} + u_n$.
Faire afficher les termes u_2 à u_n, n étant saisi au clavier.

5) Faire afficher la somme $S_1(n) = 1 + 2 + ... + n$, n étant saisi au clavier.
Reprendre l'exercice avec la somme $S_2(n)$ des carrés puis la somme $S_3(n)$ des cubes des entiers de 1 à n.

6) On démontre que la limite de la somme $S(n)$ suivante, lorsque n tend vers $+\infty$, est égale à $\frac{\pi}{4}$:

$$S(n) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ... + \frac{(-1)^n}{2n+1}$$

Faire afficher une valeur approchée de π en calculant $S(n)$, n étant saisi au clavier.

7) Un nombre est parfait s'il est égal à la somme de ses autres diviseurs. C'est le cas, par exemple, de 6 = 1 + 2 + 3 ou de 28 = 1 + 2 + 4 + 7 + 14.
Ecrire un programme testant si un nombre est parfait.
Modifier ce programme pour faire afficher la liste des nombres parfaits compris entre deux entiers saisis au clavier.

8) Un nombre est premier s'il admet exactement deux diviseurs : lui-même et l'unité.
Ecrire un programme testant si un nombre est premier.
Modifier le programme précédent pour faire afficher la liste des nombres premiers compris entre deux entiers saisis au clavier.