

# Quelques commandes de base en SCILAB

Jean-Paul Chehab  
Laboratoire de Mathématiques Paul Painlevé  
Equipe AN-EDP, Bât. M2  
Université des Sciences et Technologies de Lille  
F-59655 Villeneuve d'Ascq Cedex - France  
mél : [chehab@math.univ-lille1.fr](mailto:chehab@math.univ-lille1.fr)

SCILAB est un logiciel de calcul scientifique (du type de MATLAB), développé par l'INRIA, dans lequel sont implantés des outils voués à des simulations en analyse numérique (matricielle notamment) et en statistiques ; des interfaces en Fortran et en C sont prévues. Les calculs sont effectués en double précision (16 chiffres significatifs), ce n'est donc pas un logiciel de calcul formel. Ce logiciel est à diffusion non commerciale, il ne coûte pas un euro. Il existe des versions Windows et Linux de SCILAB, qu'on peut télécharger aisément en se connectant sur le site ouèbe

<http://www.scilab.org>

La documentation complète est également disponible sur ce site ; on peut télécharger sa version "postscript" ou bien la consulter en ligne (version "html").

On donne ici quelques commandes de bases en SCILAB ; il est recommandé d'utiliser le **help** et de s'inspirer des démos pour effectuer des manipulations plus sophistiquées techniquement. Ce qui suit n'est donc pas un manuel de SCILAB mais simplement un *vade-mecum* pour débutants.

D'autres commandes de SCILAB seront introduites avec les feuilles d'exercices suivant les besoins. On abordera des problèmes d'analyse numérique et de proba stats ; les exercices portant sur des problèmes de probabilités et de statistiques sont dus à Olivier Garet (Laboratoire de Mathématiques, Applications et Physique Mathématique d'Orléans UMR 6628).

# Contents

<b>1 Les types de données simple</b>	<b>3</b>
1.1 les constantes réelles . . . . .	3
1.2 Les booléens . . . . .	3
1.3 Les complexes . . . . .	4
1.4 Fonctions prédéfinies . . . . .	4
<b>2 Les types matrices et vecteurs</b>	<b>5</b>
2.1 Les vecteurs . . . . .	5
2.2 Les matrices . . . . .	5
2.2.1 Définitions et fonctions prédéfinies . . . . .	5
2.2.2 Quelques opérations . . . . .	6
<b>3 Les polynômes et les fonctions réelles</b>	<b>7</b>
3.1 Polynômes simples . . . . .	7
3.2 Matrices de polynômes . . . . .	8
3.3 Les fonctions réelles . . . . .	8
<b>4 La programmation en SCILAB</b>	<b>9</b>
4.1 Les opérateurs de comparaison . . . . .	9
4.2 Les boucles . . . . .	9
4.3 Les tests . . . . .	9
4.4 Les sorties de boucles . . . . .	10
<b>5 Les fichiers de commandes</b>	<b>10</b>
5.1 Les fichiers de type <i>script</i> . . . . .	10
5.2 Les sous-programmes <i>function</i> . . . . .	11
<b>6 Représentations graphiques en SCILAB</b>	<b>11</b>
6.1 Graphiques 2D . . . . .	11
6.1.1 Plusieurs courbes dans un même graphique . . . . .	12
6.1.2 Différentes courbes dans différents graphiques . . . . .	12
6.1.3 Paramètres de la commande <i>plot2d</i> . . . . .	12
6.1.4 Titres et légendes . . . . .	12
6.2 La commande <i>plot3d</i> . . . . .	12
6.3 Sauver un graphique . . . . .	13
<b>7 Liste des fonctions élémentaires</b>	<b>14</b>

# 1 Les types de données simple

## 1.1 les constantes réelles

On distingue deux types de constantes :

- celles prédéfinies

SCILAB	%pi	%e	%eps
math	$\pi$	e	$\epsilon$ -machine

- celles définies par affectation
- ```
--> a=3; // ces deux barres permettent de placer un commentaire
--> A=5; // SCILAB distingue majuscules et minuscules
--> a+A // une opération simple
ans=
8.
--> 1+%eps // definition de l'epsilon machine
ans=
1.
```

On peut effectuer plusieurs commandes sur une même ligne : pour cela on sépare deux commandes successives par un ';' si on ne désire pas que le résultat s'affiche à l'écran (lorsqu'on effectue un calcul intermédiaire par exemple). Si au contraire on veut voir le résultat affiché, il faut séparer les deux commandes par ', '

```
--> a=2; b=2;3*a+8*b^3 // le resultat de ce dernier calcul sera affiche
ans=
70.
```

Réciproquement une commande peut s'étaler sur plusieurs lignes. Pour cela on place ' ... ' à la fin de la ligne que l'on désire prolonger

```
--> x=10*(a*sin(b)+...
--> 3*b^2);
```

Cela est utile lorsqu'une commande est trop longue pour tenir en une seule ligne, par exemple quand on écrit une expression arithmétique kilométrique.

## 1.2 Les booléens

Les constantes booléennes sont  
%t et %f.

```
-->%f
%f =
```

F

```
-->%t
%t =
```

T

### exemple :

On définit la suite de nombres entiers  $3, \dots, 15$  par

```
-->suite=3:15;  
  
-->suite>7// on donne une valuation a la proposition "terme(suite)>7"  
ans =  
  
! F F F F F T T T T T T T !
```

## 1.3 Les complexes

On définit

```
--> %i ; // la racine de -1 ; c'est une constante predefinie  
--> z=3+5*%i; // affectation
```

A l'instar des réels on effectue les opérations algébriques usuelles avec les complexes définis par SCILAB.

```
--> z1=3+5*%i;  
--> z2=2+%i;  
--> z1+z2 // addition de deux complexes  
ans =  
5+6*%i  
--> z1*z2 // multiplication  
ans=  
1.+13*%i
```

## 1.4 Fonctions prédéfinies

On retrouve les fonctions usuelles (comme sur une calculatrice) :

`sin, cos, tan, exp, log, abs, sqrt`  
Ces fonctions auront pour argument un réel.

Pour les complexes les fonctions suivantes sont prédéfinies :

`abs` module  
`real` partie réelle  
`imag` partie imaginaire  
`sqrt` racine carrée  
Par exemple  
--> `sqrt(2*%i)` // si si ca marche !  
ans=  
1+%i

## 2 Les types matrices et vecteurs

### 2.1 Les vecteurs

On peut définir un vecteur ligne par affectation :

```
--> v=[1 2 3] // ne pas oublier l'espace entre chaque nombre !
v=
! 1. 2. 3.!
--> v' // vecteur colonne (tranpose de v)
v'=
! 1. !
! 2. !
! 3. !
```

Si  $v$  et  $w$  sont deux vecteurs de même taille et de même type (ligne ou colonne), on peut considérer les opérations suivantes

```
--> v+w ; // la somme des vecteurs v et w
--> v'*w ; // le produit scalaire (v et w sont des vecteurs lignes ici)
--> abs(v) ; // le vecteur ligne [abs(v1), ..., abs(vn)]
```

**Remarque 1** On peut également définir un vecteur à l'aide d'incrément.

```
--> v=5:-0.5:3
v=
! 5. 4.5 4. 3.5 3. !
```

### 2.2 Les matrices

#### 2.2.1 Définitions et fonctions prédéfinies

```
--> A=[1 2 ; 3 4] // affectation
A=
! 1. 2. !
! 3. 4. !
```

Il existe des matrices constantes en SCILAB. Regardez la description des commandes **ones** (matrice n'ayant que des 1 pour coefficient), de **zeros** (idem mais avec des 0) et **eye** (matrice identité). Il est à noter que si  $A$  a été définie comme matrice  $2 \times 3$  en SCILAB la commande

```
--> eye(A)
ans=
! 1. 0. 0. !
! 0. 1. 0. !
définira la matrice identité de même taille ; c'est pratique.
```

On peut effectuer les opérations algébriques usuelles :

```
-->A*B ; // multiplication des matrices
```

```
-->A*b' ; // multiplication matrice - vecteur
-->A+B ; // addition de matrices.
```

Enfin, et cela est important dans la pratique, les fonctions usuelles d'algèbre linéaire sont préprogrammées en SCILAB

| Algèbre linéaire              | SCILAB              |
|-------------------------------|---------------------|
| déterminant de H              | <b>det(H)</b>       |
| conditionnement de H          | <b>cond(H)</b>      |
| valeurs propres de H          | <b>spec(H)</b>      |
| polynôme caractéristique de H | <b>poly(H, 's')</b> |
| inverse de H                  | <b>inv(H)</b>       |
| trace de H                    | <b>trace(H)</b>     |
| rang de H                     | <b>rank(H)</b>      |
| norme de H                    | <b>norm(H,flag)</b> |
| $\ H\ _{frobenius}$           | si flag='fro'       |
| $\ H\ _\infty$                | si flag='inf'       |
| $\ H\ _2$                     | par défaut          |
| taille de H                   | <b>size(H)</b>      |

Exemple : les commandes suivantes sont équivalentes :

```
--> det(%s*eye(H)-H) ; // %s signifie que s est utilisee comme variable
et
--> poly(H,'s');
```

**Remarque 2** L'application d'une fonction numérique prédéfinie à une variable multidimensionnelle (vecteur, matrice) a pour effet d'agir sur chaque composante de cette variable. Par exemple, **exp(H)** est la matrice dont les coefficients sont les nombres  $e^{H_{i,j}}$ . L'exponentielle matricielle de H est calculée au moyen de la commande **expm(H)**.

Dans le même ordre d'idées, l'instruction

**H+1**

a pour effet d'ajouter 1 à chaque coefficient de H.

## 2.2.2 Quelques opérations

### a) Extraction de matrices

- i.  $H(3,4)$  donne le coefficient  $H_{3,4}$ .
- ii.  $H(:,4)$  donne la quatrième colonne de H
- iii.  $H(4,:)$  donne la quatrième ligne de H

iv.  $\text{H}(6:9,5:6)$  donne la matrice extraite de  $\text{H} : H_{i,j}, i = 6, \dots, 9, j = 5, 6$ .

v.  $\text{H}(:, \$)$  : dernière colonne de  $H$

Il existe des fonctions d'extraction incorporées ; on pourra consulter dans le *help* la description des commandes `diag`, `tril`, `triu`

### b) Les différents produits

Par commodité, les matrices sont notées par des majuscules et les vecteurs par des minuscules.

i.  $\text{M*N}$  : produit matriciel  $M * N$

ii.  $\text{M}\backslash\text{b}$  : la solution du système  $Mx = b$

iii.  $\text{M}\backslash\text{N}$  : le produit matriciel  $M^{-1}.N$

iv.  $\text{M}/\text{b}$  : la solution de  $x.M = b$

v.  $\text{M}/\text{N}$  : le produit matriciel  $M.N^{-1}$ .

vi.  $\text{M}.*\text{N}$  : produit d'hadamard des matrices  $M$  et  $N$ , i.e. composantes par composantes.

vii.  $\text{M}./\text{N}$  : division composantes par composantes, i.e. cette commande renvoie la matrice de coefficients  $M_{i,j}/N_{i,j}$ .

viii.  $\text{M}.^*\text{N}$  renvoie la matrice de coefficients  $M_{i,j}^{N_{i,j}}$

## 3 Les polynômes et les fonctions réelles

### 3.1 Polynômes simples

L'objet polynôme existe en SCILAB. On peut le définir directement de deux manières :

- A partir des racines

```
--> p=poly([ 1 2 ],'s') // polynome de la variable s dont les racines sont 1
et 2,
p=
2-3s+s^2
```

- A partir des coefficients

```
--> q=poly([1 2 ],'s','c')
q=
1+2s
```

Les opérations usuelles avec les polynômes sont prévues par SCILAB :

```
--> p*q ; // multiplication
--> p+q ; // addition
```

```
--> p/q ; // division
```

**Remarque** On peut définir en SCILAB une variable s comme étant le polynôme égal à s :

```
--> poly(0,'s')
```

La recherche des racines d'un polynôme est importante. Aussi en SCILAB est intégré une procédure numérique qui permet de déterminer, de manière approchée, les racines sous forme de vecteur colonne. Il s'agit de la commande **roots**. Exemple

```
--> p=poly([2 3 1],'s','c');  
--> roots(p)  
ans=
```

```
! 1. !  
! 2. !
```

Cela est aussi valable quand il existe des racines complexes.

On pourra consulter dans le **help** la description des commandes **derivat**, **degree**, **horner**, **pdiv**

**Remarque 3** *On peut faire quelques calculs de type formel sur les polynômes : changements de variables, développements et simplification, dérivation (cf plus bas). Par exemple, exécutez la suite d'instructions*

```
-->poly(0,'s')  
-->w=s*s+%pi*(1-3*s)^2  
-->p=poly([1 2 3 4 5],'s')  
-->horner(p,w)
```

### 3.2 Matrices de polynômes

Exemple : tapez les commandes suivantes

```
-->p=poly([1 2 3 4],'s','c')  
-->H=[ p derivat(p) ; p-1 derivat(p)*p/(p^2+1)]  
-->H('num')  
-->H('den')  
-->derivat(H)
```

### 3.3 Les fonctions réelles

Voici quelques opérations que l'on peut effectuer avec une fonction  $f$

i. Intégration

-->4\*integrate('sqrt(1-x^2)', 'x', 0, 1) donne une valeur approchée de  $4 \int_0^1 \sqrt{1 - x^2} dx$ .

Il est à noter que **integrate** est utilisé ici comme "boîte noire", la méthode d'intégration n'est pas spécifiée.

- ii. Evaluation  
 $\rightarrow \text{horner}(\pi/4, \sin(x))$  retourne une valeur approchée de  $\sin(\frac{\pi}{4})$ .

## 4 La programmation en SCILAB

### 4.1 Les opérateurs de comparaison

| SCILAB                     | math   |
|----------------------------|--------|
| <code>== ou =</code>       | =      |
| <                          | <      |
| >                          | >      |
| <code>&gt;=</code>         | $\geq$ |
| <code>&lt;=</code>         | $\leq$ |
| <code>&lt;&gt; ou =</code> | $\neq$ |

### 4.2 Les boucles

On distingue essentiellement deux types de boucles :

- la boucle de type `for`

```
--> for k=1:3, x=2*k,end
x=
2.
x=
4.
x=
6.
```

- la boucle de type `while` (tant que)

```
--> x=1; while x<4, x=3*x,end
x=
3.
x=
6.
```

### 4.3 Les tests

Le test `if-then-else` est incorporé en SCILAB

Qu'il effectue la suite de commandes suivantes

```
--> if x < 0 then , y=-x,else, y=x,end
```

#### 4.4 Les sorties de boucles

Elles sont très utiles pour arrêter le cours d'un processus itératif (précision souhaité atteinte avant le nombre maximum d'itérations). A cet effet, on utilisera la commande **break**. Exemple

```
--> a=5;
--> for i=1:100, a=(a+2/a)/2, ...
--> if abs(a-sqrt(2)) < 10(-6) then break, ...
--> end;
```

Exercice :

- Quelle méthode est mise en œuvre à l'aide de ces commandes ?
- Le test de sortie vous paraît-il utile ?

### 5 Les fichiers de commandes

Jusqu'à présent, SCILAB a été utilisé de manière conversationnelle, c'est à dire que les résultats des commandes apparaissent au fur et à mesure qu'on les valide. Il est possible d'écrire ces suites d'instructions dans des fichiers qui pourront être appelés durant une session. On distingue deux types de fichiers :

#### 5.1 Les fichiers de type *script*

Ce sont des fichiers de caractères ascii dans lesquels sont écrites des suites d'instructions SCILAB, telles que celles rencontrées plus haut. On utilise un éditeur de texte, comme par exemple emacs, pour écrire dans ces fichiers.

Pour faire executer la suite d'instructions contenue dans le fichier 'toto' dans une session SCILAB, il faut taper la commande

```
exec('toto')
```

Toutes les variables définies dans 'toto' seront alors en mémoire dans la session.

**exemple :**

```
i=1
x=0.
d=10.
while i < 100 & d > 10^(-6)
x=x-horner(p,x)/horner(derivat(p),x)
d=abs(horner(p,x))
i=i+1
end
```

Quelle tâche effectue cet algorithme ?

## 5.2 Les sous-programmes function

On les édite comme les fichiers de type *script*. Les fonctions SCILAB se présentent sous la forme :

```
function[y1,...,yn]=nom(x1,...,xm)
liste d'instructions
end
```

Ici  $x_1, \dots, x_m$  est la liste des arguments de la fonction 'nom' et  $y_1, \dots, y_n$  la liste des sorties. Pour que cette fonction puisse être utilisée dans une session SCILAB, on a recours à la l'instruction

```
getf('nom-fichier.sci')
```

exemple :

```
function sortie=mca(n)
j=0
for i=1:n
rand('uniform')// les variables aleatoires suivront une loi uniforme sur [0,1]
a=rand()
b=rand()
if a < sqrt(1-b^2) then
j=j+1
end
sortie=j/n
```

Qu'effectue d'après-vous cette fonction ? Calculer  $4.*mca(100)$ . Le résultat est-il surprenant ?

## 6 Représentations graphiques en SCILAB

### 6.1 Graphiques 2D

Pour pouvoir obtenir une représentation graphique en SCILAB il faut se donner

- un vecteur d'abscisses, par exemple à l'aide d'incrément :  $t=0:0.1:2*\pi$
- un vecteur d'ordonnées. On peut le construire à l'aide d'une fonction comme fonction d'un vecteur ; on rappelle qu'en SCILAB  $f(v)=[f(v_1), f(v_2), \dots, f(v_n)]$  si  $v=[v_1, v_2, \dots, v_n]$

Exemples

```
t=(0:0.1:2*pi)
plot2d(t,sin(t)) // courbe simple

plot2d(sin(t),cos(t)) // courbe paramétrée
```

### 6.1.1 Plusieurs courbes dans un même graphique

On peut tracer plusieurs courbes sur un même graphique. Pour cela il faut que le vecteur d'ordonnées soit composé de deux vecteurs de données. Par exemple

```
plot2d([t,t], [f1(t) f2(t)])
```

### 6.1.2 Différentes courbes dans différents graphiques

Pour pouvoir réaliser cela en SCILAB, il faut préciser le point d'origine de chaque courbe ainsi que la longueur des axes dans la direction x et y. A cet effet on utilise la commande `xsetech`.

Exemples On désire tracer dans 2 graphiques différents mais dans une même fenêtre graphique les courbes de  $f(t) = \sin(t)$  et  $g(t) = \exp(t)$

```
t=0.:0.1:2*pi
xsetech([0.,0.,0.6,0.3],[-1,1,-1,1])
plot2d(t,sint(t))
xsetech([0.5,0.3,0.4,0.6],[-1,1,-1,1])
plot2d(t,exp(t))
```

On pourra utiliser la commande `xgrid()` pour superposer un réseau (une grille) sur le graphe.

### 6.1.3 Paramètres de la commande `plot2d`

La commande `plot2d`, selon que l'on ajoute à la fin les chiffres 1,2,3,4 ou qu'on l'utilise telle quelle produit différents types de graphiques.

| Commande SCILAB      | type de courbes                     |
|----------------------|-------------------------------------|
| <code>plot2d</code>  | linéaire par morceaux               |
| <code>plot2d1</code> | idem mais avec échelle log possible |
| <code>plot2d2</code> | constante par morceaux              |
| <code>plot2d3</code> | barres verticales                   |
| <code>plot2d4</code> | flèches                             |

### 6.1.4 Titres et légendes

Pour placer un titre au milieu d'un graphique : `titlepage('titre')`

Pour placer un titre au bas d'un graphique : `xtitle('titre')`.

Ces deux commandes sont à placer après l'ordre `plot2d`.

## 6.2 La commande `plot3d`

SCILAB offre la possibilité de représenter des graphiques 3D. La commande de base est

```
plot3d(x,y,z) où x, y et z sont trois matrices.
```

### 6.3 Sauver un graphique

On utilise la commande

```
xsave('nom-fichier' [, numero-fenetre] )
```

et pour charger le graphique 'nom-fichier' on effectue

```
xload('numero-fenetre' [, numero-fenetre] )
```

## 7 Liste des fonctions élémentaires

|             |              |           |           |            |
|-------------|--------------|-----------|-----------|------------|
| abs         | acos         | acosh     | acoshm    | acosm      |
| addf        | addmenu      | adj2sp    | amell     | asin       |
| asinh       | asinhm       | asinm     | atan      | atanh      |
| atanhm      | atanm        | besseli   | besselj   | besselk    |
| bessely     | bloc2exp     | bloc2ss   | c_link    | calerf     |
| cmb_lin     | conj         | convstr   | cos       | cosh       |
| coshm       | cosm         | cotg      | coth      | cothm      |
| cumprod     | cumsum       | debug     | dec2hex   | delip      |
| delmenu     | demos        | diag      | dlgamma   | log        |
| edit        | emptystr     | erf       | erfc      | erfcx      |
| eval        | execstr      | full      | gamma     | gammaln    |
| getvalue    | gsort        | halt      | hex2dec   | havewindow |
| input       | interp       | intsplin  | intrap    | integrate  |
| isdef       | isinf        | isnan     | kron      | ldivf      |
| linspace    | log10        | interp1n  | logm      | logspace   |
| macr2lst    | macrovar     | manedit   | mean      | median     |
| modulo      | mult         | nnz       | norm      | pen2ea     |
| pertrans    | prod         | rdivf     | readc_    | readmps    |
| sci2exp     | sci2map      | setmenu   | sin       | sinh       |
| sinhm       | sinm         | smooth    | solve     | sort       |
| sp2adj      | sparse       | speye     | spcompack | spget      |
| splin       | spones       | sprand    | spzeros   | sqrt       |
| sqrtm       | ssprint      | ssrand    | G_make    | strcat     |
| stripblanks | subf         | strindex  | strsubst  | sum        |
| sysconv     | sysdiag      | syslin    | tan       | tanh       |
| tanhm       | tanm         | timer     | toeplitz  | trfmod     |
| trianfml    | tril         | trisolve  | triu      | typeof     |
| unsetmenu   | st_deviation | x_choices | zeros     | x_choose   |
| x_dialog    | x_matrix     | x_mdialog | x_message | x_getfile  |