

Introduction

L'introduction à Scilab est donnée dans le chapitre 2 du poly disponible en ligne sur le site et vous avez été invité à l'étudier chez vous en tapant des commandes directement sur une console scilab, Scilab devant être installée sur votre machine personnelle. Scilab étant librement utilisable.

On donne ici un certain nombre de remarques et d'instructions figurant aussi dans le poly. Vous êtes donc invités à rentrer les instructions qui vont suivre. L'interpréteur de Scilab vous indique immédiatement les erreurs. Vous pouvez bien entendu essayer des variantes des instructions. (Attention cependant ne modifiez pas les variables utilisées dans l'énoncé, utilisez d'autres variables sinon les opérations suivantes données dans l'énoncé peuvent vous indiquer des erreurs (et ce serait normal).

Ce document a été prévu pour Scilab 5.xx pour les utilisateurs de la version 6.xx, il y a quelques différences mineures, mais certaines instructions qui généraient des erreurs dans la version 5, sont interprétés différemment dans la version 6 et n'en produisent plus..

Le but n'est pas de taper les commandes le plus vite possible mais de comprendre ce qu'elles font !

On utilise `//` pour indiquer un commentaire (C'est la syntaxe d'un commentaire pour scilab). Une instruction terminée par un `" ;"` n'affiche pas le résultat à l'écran. Il y a un help en ligne accessible : (ou bien l'icône "?" de la console scilab)

```
>help
>help "instruction qu'on veut connaître"
```

Préliminaires - Quelques instructions Observez les effets des instructions suivantes :

Pour toutes les commandes qui suivent ; indiquer sur ce papier ce qu'elles font (et ne vous contentez pas de reporter le résultat affiché)

```
>x=4:12
>x(2)
>x(2)=100
>x=1:10;      // rien ??
>x            // noter l'effet du ;
>x=1:2:10
>x=6:0.25:8

>size(x)
>length(x)

>x(5)
>x(12)        // ??
>x(12)=3      // !!

>x=10:1        // !!
>x=10:-1:1
>x=10:-2:1
>x=1:-1:10     // !!

>x=1:2:7
```

```
>y=10:20:70
```

```
>z=[x y]
```

```
>z=[x;y]
```

```
>x+2*y
```

```
>x*y'
```

```
>x'*y
```

```
>x*y
```

La dernière instruction génère une erreur! En effet Scilab essaye d'effectuer une multiplication matricielle $(4,1) \times (4,1)$ qui n'est pas définie! Les premières instructions effectuaient les multiplications $(1,4) \times (4,1)$ et $(4,1) \times (1,4)$ qui donnent des matrices de type $(1,1)$ et $(4,4)$. On peut cependant vouloir effectuer des multiplications ou divisions ou autre opérations "éléments par éléments" en écrivant :

```
>y.*x
```

```
>y./x
```

```
>y.^x
```

Le "." indique que les opérations s'effectuent élément par élément.

Attention sous Scilab si V est un vecteur $1/V$ n'est pas le vecteur contenant les inverses des éléments de V . N'utilisez jamais cette expression " $1/V$ " qui a un tout autre sens. A la console faites le test simple :

```
>V=1:5
```

```
>1/V
```

```
>1/V'
```

Pour obtenir un vecteur de même type que V comprenant les inverses des éléments de V utilisez l'une des expressions suivantes : (le "." indiquant donc que l'on effectue une opération élément par élément)

```
>W=1../V
```

```
>W=(1)./V
```

```
>W=V.^(-1)
```

```
>W=V.^-1
```

Attention ">W=1./V" ne donne pas le bon résultat car le "." est interprété comme le "." décimal.

(En fait le comportement est différent avec la version 6 de Scilab, mais il vaut mieux éviter ces écritures ambiguës sources d'erreurs)

- **Quelques autres instructions utiles en vrac - testez les! Retenez les**

Pour toutes les commandes qui suivent; indiquer sur ce papier ce qu'elles font (et ne vous contentez pas de reporter le résultat affiché)

```
>A=[1 4 7 6; 2 9 8 3;0 1 -8 4]
```

```
>A(2,3)
```

```
>A(2,3)=-5
```

```
>A(2,6) // !! erreur
```

```
>A(2,6)= 13 // !!
```

```
>A(3,[4 5 6])=[ 1 2 3]
```

```
>A(2,[5 1])=[-10 -20 -30] // erreur !!
```

```
>A(2,[5 1])=[ -10 -20]
```

```
>A([2 3],4)
```

```
>[100 200]
```

```
>A([2 3],4)=[100 200] // erreur !!
```

```
// (scilab 5)
```

```
>A([2 3],4)=[100 200]' // !!
```

```
>A(15) // !!
```

```
>A(15)=13 // !!
```

```
>A(2:7) // !!
```

```
>B=[ 1 2 3;6.1 ... // taper les ...
```

```

                                puis ENTREE
7 8 ; 9 10 11]                //

>A=zeros(4,5)
>A=ones(3,4)
>A=eye(3,3)
>A=rand(4,5);
>A=round(100*rand(4,4))
>A=round(100*rand(4,4)-50)

>a=A(2,3)
>b=A(2,4)

>c=A([2 4],3)
>d=A([2 4],:)
>h=A(:,[3 2])
>e=A(4:-1:1,3)
>e=A(4:-1:1,:)

>E=triu(A)
>F=tril(A)
>G=A'
>A=rand(5,5)
>X=(1:5)
>X=(1:5)'

>B=A*X
>Y=B/A    // !! erreur
>Y=A\B    // la "division à gauche" !
            cela équivaut à multiplier à gauche par l'inverse de A
            (si A est effectivement inversible)
            On peut donc ainsi vérifier une résolution de système
            linéaire Ax=B
>Y-X      // Pourquoi n'est ce pas nul ?

>B=rand(5,1) // Autre résolution, cette
>X=A\B       // fois on ne connaissait
              // pas déjà la solution
>A*X-B       // Pourquoi n'est ce pas nul ?

>A=matrix(1:28,4,7)
>length(A)
>size(A)
>A(:,7:-1:1)
>A(4:-1:1,:)
>A([1 3],[6 2])

```

- L'instruction "diag"

```

>U=1:5
>diag(U)
>V=1:4

```

```

>diag(V,1)
>diag(V,-1)
>diag(U)+diag(V,-1)+diag(V,1)
>A=matrix(1:16,4,4)
>diag(A)           // !
>diag(diag(A))     // !!

```

- L'instruction "*linspace*" complémentaire du ":".

```

>t= linspace(2,7,11)    // remarquez les répartitions de valeurs
>t= linspace(2,7,10)    // et leurs nombres
>t= linspace(7,2,21)    // selon ces trois instructions

```

- Un tracé simple

```

>t=linspace(0,2*%pi,100); // n'oubliez pas le ";"
>x=cos(t);
>clf;
>plot(t,x) // ne pas fermer la fenêtre graphique
>y=sin(t);
>plot(t,y,'r') // ne pas fermer la fenêtre graphique

>scf(2); // bouger cette fenêtre !
>y=sin(t);
>plot(x,y,'k'); // ne pas fermer la fenêtre
>plot(x,y,'r+');
>q=gca()
>q.isoview="on" // quelle différence !!
>q.isoview="off" // quelle différence !!
>clf;
>xdel(0);
>xdel(2);

```

- un autre tracé

```

>clf;
>x=linspace(-1,1,11);
>y=x.*x
>plot(x,y,'r+')

```

- Scilab a une notion de répertoire courant qu'on peut afficher avec l'instruction ">pwd". On peut changer de répertoire courant avec l'instruction ">cd" ou avec l'aide du menu "*Fichier*".
- L'éditeur peut être lancé par l'intermédiaire de l'icône en haut à gauche de la console. Les scripts peuvent être lancés et les fonctions chargées à partir de cet éditeur par l'icône en forme de triangle usuel.
- Les fonctions scilab sont traitées dans le chapitre 3 du poly scilab. La syntaxe est simple :

```

function [Variables de sortie]=nom_de_la_fonction(Variables d'entrée)
    "instructions"
endfunction

```

N.B. Attention au mot clef "function" et non pas "fonction".

Les variables d'entrée (entre crochets) et de sortie (entre parenthèses) sont séparées les unes des autres par des virgules. On peut mettre une variable en entrée et en sortie. Les variables du programme principal sont connues dans la fonction, mais n'y sont pas modifiables : les modifications sont "oubliées" dès la sortie de la fonction de même que toute modification des variables d'entrée ne figurant pas en sortie. On peut considérer que les variables figurant simultanément en entrée et en sortie sont passées par adresse.

- La structure d'une boucle "for" classique est la suivante : (La première boucle est ascendante, la seconde est descendante)

<pre>n=10 for i=1:n "instructions" end</pre>	<pre>n=10 for i=n:-1:1 "instructions" end</pre>
---	--

Vous devez connaître et mémoriser ce que font toutes les instructions de base données ci dessus, c'est la seule façon de pouvoir commencer à maîtriser le langage Scilab, ceci pourra de plus faire l'objet d'un exercice de contrôle en DS ou en examen.

Un tracé plus complexe

Ce qui suit est plus compliqué et donne l'occasion d'écrire un script avec des fonctions. Vous devez comprendre l'utilité des différentes instructions même s'il ne vous est pas demandé de mémoriser les différentes subtilités pour le tracé graphique.

On a donc ci dessous un script scilab simple pour la représentation de deux fonctions de \mathbb{R}^2 dans \mathbb{R} sur deux figures séparées. Ouvrir l'éditeur Scilab (SciNotes) (l'icône en haut à droite dans la console scilab) saisissez le code suivant, enregistrer le ("monscript.sce" ou "monscript.sci") dans votre répertoire et lancer son exécution depuis SciNotes - (l'icône de lancement est le triangle).

```
// représentation d'une fonction de R^2 dans |R.
clear

function val=mafonctiona(x,y)
// une cloche
    val=16*x*(x-1)*y*(y-1)
endfunction
function val=mafonctionb(x,y)
// oscillations ...
    val=sin(2*%pi*x)*sin(3*%pi*y)
endfunction

a=0;
b=1;
N=40;
x=linspace(a,b,N);
y=linspace(a,b,N);

za=feval(x,y,mafonctiona);
zb=feval(x,y,mafonctionb);
// z_ij=mafonction(xi,yj) pour tous les couples (x_i,y_j)
// voir le help de feval
```

```
// za ou zb sera donc ici un tableau NxN
mb=min(zb);
Mb=max(zb);
ma=min(za);
Ma=max(za);

ma_boite_a_couleur=jetcolormap(256);
clf(1);
g=scf(1); // fenêtre 1
g.color_map=ma_boite_a_couleur; // définition d'une carte de couleurs
colorbar(ma,Ma);
plot3d1(x,y,za);
h=get("hdl");
h.color_mode=-2;

clf(2);
g=scf(2) // fenêtre 2

g.color_map=ma_boite_a_couleur; // définition d'une carte de couleurs
colorbar(mb,Mb);
plot3d1(x,y,zb);
h=get("hdl");
h.color_mode=-2;
```

———— • **FIN** • ————