Introduction à Scilab

Nicolas Kielbasiewicz*

21 juin 2007

Scilab est un logiciel gratuit développé à l'INRIA (l'Institut National de Recherche en Informatique et Automatique) sous Windows, Linux et Mac, et dédié au calcul numérique. Pour le télécharger et obtenir de plus amples informations, consulter :

http ://www.scilab.org

Scilab dispose d'un langage de programmation basé essentiellement sur le calcul matriciel, avec des fonctionnalités mathématiques et graphiques étendues.

L'objectif de ce document n'est pas de donner une liste exhaustive des diverses fonctionnalités de Scilab , mais plutôt de donner un aperçu des commandes de base et des fonctions les plus couramment utilisées.

Table des matières

*Unité de Mathématiques Appliquées, École Nationale Supérieure de Techniques Avancées

1 Premiers contact avec Scilab

1.1 La Fenêtre Scilab

Pour démarrer, il suffit de cliquer sur l'icône de Scilab ou de taper **scilab &** dans une fenêtre de terminal. Le programme se lance et une fenêtre graphique s'ouvre.

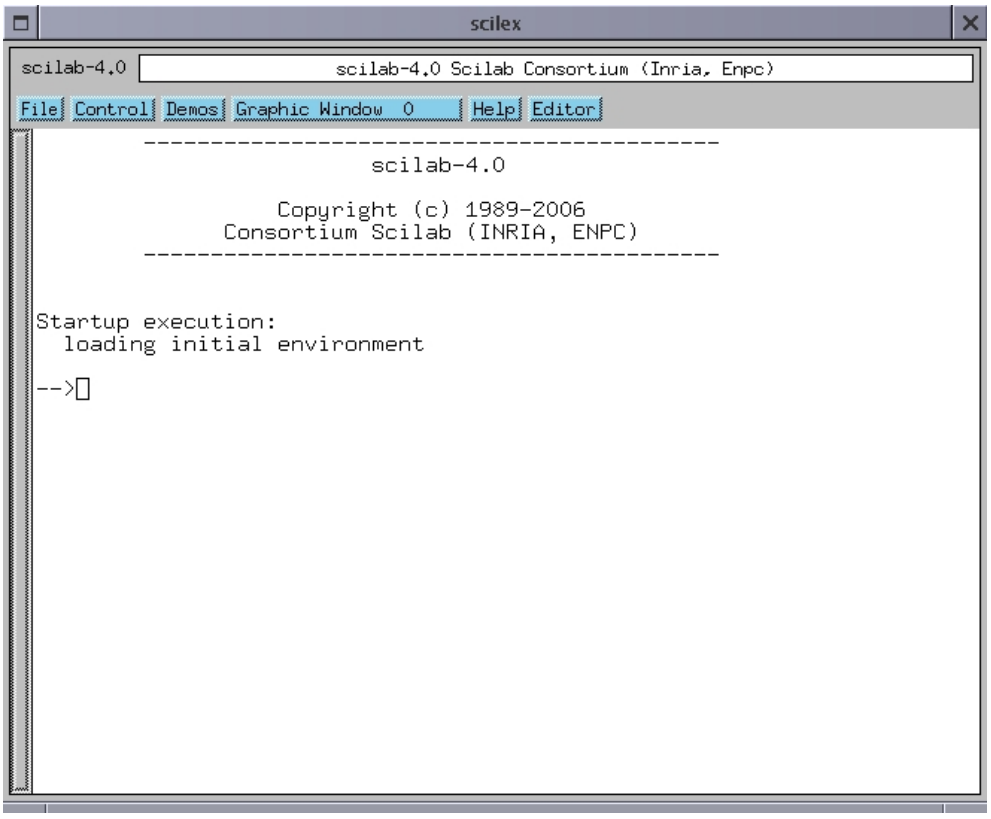

Fig. 1 – La fenêtre Scilab

Cette fenêtre assez simple d'utilisation comporte un menu très pratique contenant en particulier :

file permet d'accéder aux commandes de compilation des librairies et d'exécution des scripts.

help permet d'accéder à l'aide

editor permet de lancer l'éditeur de texte de Scilab

1.2 Les fichiers Scilab

Bien que Scilab utilise des fichiers texte, et que l'extension n'a à priori aucune importance, je conseille néanmoins fortement de suivre quelques conseils sur la définition des différents fichiers.

1.2.1 Les librairies

Ce sont des fichiers du type **file.sci**. Ces fichiers sont dédiés à la définition des différentes fonctions. Pour être utilisés, ces fichiers doivent être compilés au préalable. Pour cela, on utilise la commande **getf** :

getf("fichier.sci") ;

On peut exécuter cette commande dans la fenêtre SCILAB ou dans un script.

1.2.2 Les scripts

Ce sont des fichiers du type **file.sce**. Ils vont être exécutés par SCILAB grâce à la commande **exec** :

exec("fichier.sce") ;

Ces fichiers contiennent une suite d'instructions. Ces instructions pourraient tout aussi bien être exécutés directement dans la fenêtre SCILAB . On peut donc y charger des librairies, effectuer des opérations d'entrées/sorties, des calculs, exécuter des commandes et des fonctions, ...

1.3 L'aide

Hormis le menu **help**, il existe plusieurs moyens de trouver des informations :
- La commande **help()** retourne des informations générales sur l'aide en ligne.
- La commande **help nom _fct** effectue une recherche dans l'aide sur la commande nom_fct.
- La commande **apropos chaine** effectue une recherche dans l'aide avec le mot-clé chaine.

2 Les types de données

2.1 Les constantes spéciales

La liste complète des constantes spéciales est obtenue grâce à la commande **who**. Elles sont toutes précédées du caractère %. En voilà les plus courantes :

%nan	NotANumber
%inf	l'infini
%t	le booléen true
%f	le booléen false
%eps	le zéro machine
%i	le nombre imaginaire
%e	le nombre exponentiel
%pi	le nombre π

2.2 Vecteurs et matrices

Pour définir un vecteur dont les valeurs sont uniformément réparties, il existe deux méthodes :
- **u=deb :pas :fin ;**
- **u=linspace(deb,fin,nbval) ;**

Pour définir un vecteur ou une matrice élément par élément, il faut procéder comme suit :
- Les crochets [] servent à encadrer la matrice ;
- un espace ou une virgule , sépare deux éléments d'une même ligne ;
- un point-virgule ; sépare les lignes entre elles.

Exemples :
- **[2 3 -5] ;** donne **! 2. 3. -5. !**

– **[3 ;1] ;** donne **!3 !**
!1 !

Les opérations matricielles sont définies dans le tableau suivant :

symbole	définition
[]	définition matricielle et concaténation
;	séparateur de colonne
()	extraction/insertion d'un élément
'	transposition
+	addition
-	soustraction
*	produit matriciel
\	division à gauche
/	division à droite
^	puissance
.*	produit élément par élément
.\	division à gauche élément par élément
./	division à droite élément par élément
.^	puissance élément par élément

3 Fonctions ou macros

Comme il a été dit précédemment, il est préférable de définir des fonctions dans des fichiers de librairies plutôt que dans les scripts eux-mêmes. Voici la syntaxe pour définir une fonction :

function [y1, y2, . . . , yn]=nom_fct(x1, x2, . . . , xp)
. . .
. . .
. . .
endfunction

Pour définir une fonction en ligne de commande, on peut utiliser la syntaxe suivante :

deff("[y]=nom_fct(x1,x2)","y=cos(x1+x2)") ;

4 Boucles et instructions de contrôle

4.1 Les boucles

Il en existe de deux types en Scilab : les boucles for et les boucles while.
Les boucles for s'écrivent de la façon suivante :

for i=deb :pas :fin
. . .
. . .
end

Quand le pas n'est pas précisé, il est à la valeur 1 par défaut.
Il existe une autre manière d'écrire une boucle for dont l'indice de boucle parcourt les éléments d'un vecteur v :

for i=v
. . .
. . .
end

Les boucles while s'écrivent de la façon suivante :

while *expression*
...
...
end

4.2 Les instructions de contrôle

Pour effectuer un test, on peut utiliser la combinaison classique if-then-else :

if *condition* **then**
...
else
...
end

On peut également avoir à utiliser select-case dans le cas où le test comprend au moins trois possibilités :

select *var*
case *val1* **then**
...
case *val2* **then**
...
case *val3* **then**
...
end

4.3 Les opérateurs logiques

Voici la liste des opérateurs logiques servant entre autres à écrire les conditions :

==	égal à
~= ou <>	différent de
<	inférieur à
<=	inférieur ou égal à
>	supérieur à
>=	supérieur ou égal à
&	et
|	ou
~	non

5 Entrées/Sorties

5.1 Ouvrir un fichier

La commande la plus courante est :

[fid,error]=mopen('fichier.xxx',mode) ;

La variable *mode* peut être "r" (lecture), "w" (écriture), "b" (binaire), ou une combinaison des 3.

5.2 Lire dans un fichier

La commande la plus courante est :

x=mfscanf(n,fid,format) ;

n correspond aux nombre d'occurrences de données de type $format$ à extraire. Si $n = -1$, la commande récupère le maximum de données possibles jusqu'à la fin du fichier.

La variable $format$ est une chaine de caractères constituée de caractères du type %s (caractère), %d (entier), %f (flottant), ...

5.3 Ecrire dans un fichier

La commande la plus courante est :

x=mfprintf(fid,format,a1, . . . , an) ;

5.4 Fermer un fichier

mclose(fid) ;

5.5 Cas particulier des entrées/sorties FORTRAN

Il est préférable d'utiliser les commandes **file**, **read** et **write**.

6 Les graphiques

6.1 Tracés en 2D

6.1.1 plot : la commande de base

La commande **plot** est la fonction d'affichage graphique la plus élémentaire. Elle a subi des modifications substantielles depuis SCILAB 4.0 pour ressembler à son équivalent dans MATLAB. Elle permet maintenant la même gestion des couleurs et des styles de lignes.

valeur	correspondance	valeur	correspondance
-	ligne continue	r	rouge
–	ligne discontinue	g	vert
: ou -.	ligne pointillée discontinue	c	cyan
+	signes plus	m	magenta
o	cercles	y	jaune
*	astérisques	k	noir
.	points	w	blanc
x	croix		
s	carrés		
d	diamant		
p	pentagramme		
^	triangles vers le haut		
v	triangles vers le bas		
<	triangles vers la gauche		
>	triangles vers la droite		

6.1.2 La commande plot2d : structure la plus courante

C'est la commande traditionnelle de SCILAB pour les tracés en 2d.

plot2d(x,[y1 y2 ... yn], style=*style*, **strf=***chaine*, **leg=***legende*, **rect=***cadre*) ;

- les y_i sont des vecteurs colonnes, n désigne le nombre de courbes à tracer.
- *style* est un vecteurs d'entiers relatifs de taille n. Si cet entier est strictement positif, alors il caractérise une ligne continue colorée, sinon une ligne de symboles noirs et blancs.

valeur	correspondance	valeur	correspondance
1	noir	0	.
2	bleu	-1	+
3	vert	-2	×
4	cyan	-3	⊙
5	rouge	-4	♦
6	magenta	-5	◊
7	jaune	-6	△
8	blanc	-7	▽
9	bleu marine	-8	♣
26	marron	-9	○
29	rose		
32	jaune orangé		

- *chaine* est un argument du type 'xyz'
  - x=0 : pas de légende
  - x=1 : affiche la légende
  - y=0 : bornes courantes
  - y=1 : bornes définies dans *cadre*
  - y=2 : bornes ajustées
  - z=0 : pas d'entourage graphique
  - z=1 : axes gradués
  - z=2 : cadre autour du graphique
- *legende* est une chaine de caractère du style 'leg1@leg2@...', qui contient la liste des légendes pour chacune des courbes.
- *cadre* détermine les bornes des échelles graphiques. C'est un argument de la forme [**xmin ymin xmax ymax**].

6.1.3 Commandes plus spécifiques

- La commande **fplot2d** permet de tracer une courbe définie par une fonction. Ses arguments sont les mêmes que **plot2d** à ceci près que la matrice des ordonnées est remplacée par le nom de la fonction.
- Les commandes **xpoly** et **xpolys** permettent de tracer des lignes brisées et par extension des polygones. La seconde est notamment utilisée pour afficher des maillage.
- D'autres fonctions : **graypolarplot**, **grayplot**, **contour2d**, ...

6.2 Tracés en 3D

6.2.1 La commande plot3d : structure la plus courante

plot3d(x,y,z, theta=*theta*, **alpha=***alpha*, **leg=***legende*, **flag=**[*mode, type, box*], **ebox=***cadre*) ;

- x et y sont des vecteurs de tailles nx et ny. z est une matrice de taille (nx,ny).

- *theta* et *alpha* sont des paramètres angulaires permettant de définir l'angle de vue. Les valeurs par défaut sont 45 et 35.
- *legende* est une chaine de type **'xlabel@ylabel@zlabel'**.
- *mode* définit la couleur
  - mode=-n : la couleur n est choisie, mais la grille n'est pas tracée
  - mode=0 : seule la grille est tracée
  - mode=n : la couleur n est choisie et la grille est tracée
- *type* définit les paramètres d'échelle
  - type=0 : bornes courantes
  - type=1 : bornes définies avec *cadre*
  - type=2 : bornes ajustées
- *box* définit l'entourage graphique
  - box=0 : pas d'entourage graphique
  - box=1 : pas d'entourage graphique
  - box=2 : seuls les axes derrière la surface sont affichés
  - box=3 : une boite et les légende sont affichées
  - box=4 : une boite, les légendes et les axes sont affichés
- *cadre* détermine les bornes du graphique. C'est un argument de la forme [**xmin xmax ymin ymax zmin zmax**].

6.2.2 Commandes plus spécifiques

- La commande **param3d** permet d'afficher des courbes paramétrées. Contrairement à **plot3d**, son troisième argument z n'est pas une matrice, mais un vecteur de la même dimension que x et y. On peut également l'utiliser pour afficher des nuages de points.
- La commande **eval3dp** permet de définir les facettes d'un volume défini à partir d'une fonction. Elle prend comme arguments le nom de la fonction ainsi que ses deux arguments.
- D'autres fonctions : **eval3d**, **fplot3d**, **surf**, ...