Chapitre 1

Présentation de Scilab

1.1 Introduction

Scilab est un logiciel de calcul numérique fournissant un environnement de calcul pour des applications scientifiques. L'environnement Scilab met à disposition un ensemble de méthodes (routines) pré-intégrées usuellement utilisées en sciences physique :

- Résolution de systèmes linéaires, d'équations différentielles,
- Manipulation matricielle et vectorielle optimisée (valeurs propres, etc.),
- Transformées de Fourier rapides,
- Algorithmes d'optimisation.

Scilab est équipé d'une interface graphique rendant le logiciel convivial et permettant la génération de figures (2D et 3D) incorporables dans les documents (word, latex). Un atout important de la programmation *Scilab* est qu'elle ne nécessite pas l'appel de librairies spécifiques et dépendantes de l'environnement utilisé (Windows, XP, 2000, Linux, Unix, etc.). Le formalisme et la programmation *Scilab* sont les mêmes quel que soit le système d'exploitation étudié. Ainsi, les programmes sont entièrement compatibles d'un système d'exploitation à un autre.

Scilab est développé depuis 1990 par des chercheurs de l'INRIA et de l'ENPC. *Scilab* est devenu depuis mai 2003 un Consortium, développé et maintenu par l'INRIA. *Scilab* est distribué gratuitement avec son code source via l'Internet depuis 1994, il est disponible précompilé pour un grand nombre d'architectures. Néanmoins il ne s'agit ni d'un logiciel Open source selon l'Open Source Initiative ni d'un Logiciel libre. En effet, la licence de *Scilab* n'autorise pas la distribution commerciale d'une version modifiée. Selon la classification de la FSF, il s'agirait donc plutôt d'un logiciel semi-libre.

1.2 Démarrage de *Scilab*

Pour avoir *Scilab*, on peut le télécharger sur le site : http://www.scilab.org/, on l'installe puis on exécute.

Pour lancer l'exécution de *Scilab* :

- sous Windows, il faut cliquer sur Démarrage, ensuite Programme, ensuite *Scilab*,

– sous d'autres systèmes, se référer au manuel d'installation.

L'invite '–>' de *Scilab* doit alors apparaître, à la suite duquel on entrera les commandes (voir la figure (1.1)).

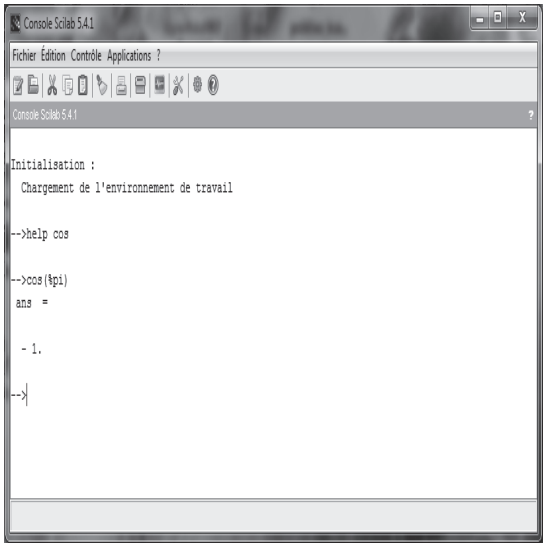


FIG. 1.1 – *Fenêtre Scilab*

La fonction quit permet de quitter *Scilab* :

```
-->quit
```

La commande help permet de donner l'aide sur un problème donné. Par exemple :

```
--> help cos
```

cette commande va ouvrir le navigateur d'aide (voir la figure (1.2)).

Elle s'obtient en cliquant sur le bouton Help de la fenêtre *Scilab*... Apparaît alors une nouvelle fenêtre (intitulée Scilab Help Panel) qui se décompose en trois parties (plus le bouton done qui détruit cette fenêtre). La partie du milieu correspond àă un classement de toutes les fonctions en un certain nombre de rubriques (Scilab Programming, Graphic Library, Utilities and Elementary functions,...) alors que la première partie donne la liste de toutes les fonctions de la rubrique qui apparaît en inverse vidéo dans la deuxième partie. Pour changer de rubrique il suffit de cliquer une seule fois sur la rubrique désirée.

Pour obtenir le détail d'une fonction il suffit de cliquer (toujours une seule fois) sur son intitulé : apparaît alors une autre fenêtre donnant ces détails. Pour fermer cette dernière fenêtre, il faut cliquer sur le bouton close window (mais garder la, tant que vous en avez besoin). Lorsque l'on ne sait pas où chercher on peut écrire un mot clé dans la troisième partie intitulée Apropos puis appuyer sur la touche Return pour lancer la recherche.

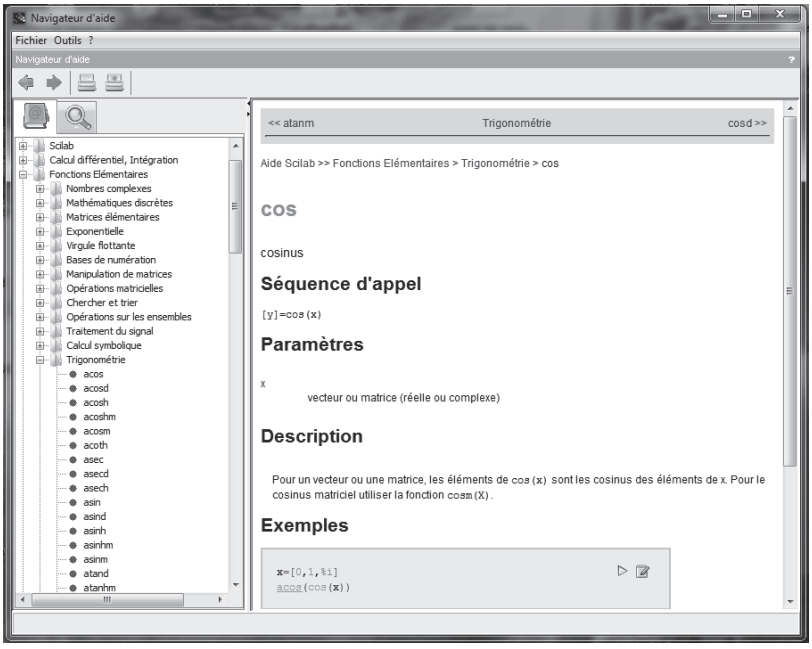


FIG. 1.2 – *Fenêtre aide*

Si la recherche échoue, vous avez le message « no info for topics ... back to chapter one » qui apparaît (signifiant que la première partie de la fenêtre d'aide affiche les fonctions de la rubrique 1 (*Scilab* Programming)). En cas de succès, la première partie affiche toutes les fonctions qui contiennent ce mot clé dans leur intitulé (c-à-d dans le nom de la fonction ou dans sa brève description). Par exemple, la recherche sur le mot clé int renvoie énormément de fonctions car la chaîne de caractères int est présente dans les mots interrupt, interface, printing, points, etc. Si je fais suivre int par un caractère blanc, on sélectionne déjà moins de fonctions et l'on voit alors clairement apparaître cette fonction dans la liste sélectionnée.

Pour tout renseignement consulter la « Scilab home page » :

http://www-rocq.inria.fr/scilab/

à partir de laquelle vous avez en particulier accès à différentes documentations, aux contributions des utilisateurs, etc... Le « Scilab Group » écrit depuis le mois de Décembre 99 un article mensuel dans « Linux magazine ». Plusieurs aspects de Scilab (dont la plupart ne sont pas évoqués dans cette introduction) y sont présentés.

Scilab dispose aussi d'un forum usenet qui est le lieu adéquat pour poser des questions, faire des remarques, rapporter un bug, apporter une solution à une question préalablement posée, etc.: comp.sys.math.scilab

Tous les messages qui ont été postés dans ce forum sont archivés et accessibles à partir de la « home page » *Scilab*. Pour terminer, je signale un nouveau document intéressant « Scilab Bag Of Tricks » élaboré par Lydia E. van Dijk et Christoph L. Spiel qu'on trouve à l'adresse :

http://www.lightlink.com/lydia/sci-bot/book1.htm/

et qui est disponible sous plusieurs formats (HTML, SGML, postscript, PDF).

1.3 Super calculette

Pour faire du calcul arithmétique, les opérations usuelles sont :

+ : plus ; - : moins ; / : division ; * : multiplication ; ^ : puissance ; $\%pi = \pi$.

Par exemple :

```
-->x=2

x =
    2
-->P=(4*x^2-2*x+3)/(x^3+1)

P =
    1.6667
```

Soit à calculer le volume suivant : $V = \frac{4}{3}\pi R^3$ où $R = 4$ cm. Pour calculer V, on exécute les commandes suivantes :

```
-->R=4

R =
    4
-->V=4/3*pi*R^3

V =
    268.0826
```

1.4 Fonctions mathématiques

Les fonctions trigonométriques sont :

sin(x), cos(x), tan(x), asin(x), acos(x), atan(x), atan2(x,y), sinh(x), cosh(x), tanh(x), asinh(x), acosh(x), atanh(x).

Les autres fonctions mathématiques (élémentaires) sont :

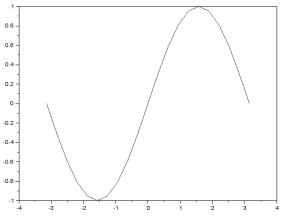
abs(x)	Valeur absolue de x
sqrt(x)	Racine carrée de x
real(x)	Partie réelle de la valeur complexe de x
imag(x)	Partie imaginaire de la valeur complexe de x
conj(x)	Complexe conjugué de x
round(x)	Arrondi à l'entier le plus proche de x
fix(x)	Partie entière la plus proche du réel x
floor(x)	Partie entière par excès du réel x
ceil(x)	Arrondi au voisinage de du réel x
sign(x)	=+1 si $x > 0$; =-1 si $x < 0$
exp(x)	Exponentielle (de base e)
log(x)	Logarithme népérien
log10(x)	Logarithme décimal

1.5 Génération de graphique avec *Scilab*

Scilab est un outil très puissant et très convivial pour la gestion des graphiques, que ce soit en une dimension, en deux dimensions ou en trois dimensions.

Pour tracer une courbe $y = \sin(x)$ par exemple, où $x = 0 : 50$; il suffit de faire :

```
-->x= -pi:pi/10:pi;
y=sin(x),
plot(x,y)
```



On a déjà vu l'instruction plot toute simple. Cependant si l'on veut dessiner plusieurs courbes, il vaut mieux se concentrer sur plot2d. L'utilisation la plus basique est la suivante :

```
-->x=linspace(-1,1,61)'; // les abscisses (en vecteur colonne)

-->y = x.^3; // les ordonnees (aussi en vecteur colonne)

-->plot2d(x,y)
```

On rajoute maintenant une autre courbe :

```
-->ybis = 1 - x.^3;

-->plot2d(x,ybis)
```

```
-->xtitle("Courbes...")  // on rajoute un titre
```

Tout se passe bien car *Scilab* choisit la même échelle, mais si on continue avec :

```
-->yter = 3*y;

-->plot2d(x,yter)
```

On remarque que *Scilab* change l'échelle et cette nouvelle courbe se confond avec la première. Si on veut afficher les trois courbes simultannément :

```
-->clf() // pour effacer

-->plot2d([x x x],[y ybis yter]) // concatenation de matrices ...

-->xtitle("Courbes...","x","y") // un titre plus une legende pour
                                // les deux axes
```

Pour afficher simultannément plusieurs courbes, l'instruction prend la forme plot2d(Mx,My) où Mx et My sont deux matrices de tailles identiques, le nombre de courbes étant égal au nombre de colonnes nc, et la i$^{\text{ème}}$ courbe est obtenue à partir des vecteurs $Mx(:,i)$ (ses abscisses) et $My(:,i)$ (ses ordonnées). La syntaxe générale est :

```
plot2d(Mx,My,[style,strf,leg,rect,nax])
```

où les arguments optionnels ont la signification suivante :

- style est un vecteur ligne de dimension nc, sa i$^{\text{ème}}$ composante spécifiant le style de la i$^{\text{ème}}$ courbe. Sur un terminal couleur, si style(i) est un entier strictement positif k, la courbe correspondante sera dessinée avec la couleur numéro k.
- strf est une chaîne de trois caractères "xyz" telle que :
  si x est le caractère 1 alors une légende pour chacune des courbes (donnée par leg qui doit alors être présent dans la liste des arguments) sera affichée ; pour les autres valeurs de x la légende n'est pas affichée ; le caractère y spécifie le calcul de l'échelle (c-à-d le rectangle xmin,ymin,xmax,ymax qui va déterminer la frontière du graphe) :
  - pour $y = 0$ on utilise l'échelle précédente (celle qui a été déterminée par un appel précédent) ;
  - pour $y = 1$ l'échelle est donnée par l'argument rect ;
  - pour $y = 2$ *Scilab* calcule automatiquement l'échelle (en prenant les max et min de Mx et My) ;
  - avec $y = 3$ et $y = 4$, on peut obtenir une échelle isométrique ;

  pour les autres possibilités voir le Help ; le caractère z permet de régler le pourtour du graphe :
  - $z = 1$, le graphe est entouré d'une boîte (dont les dimensions sont $xmin$,$ymin$, $xmax$,$ymax$) et les côtés ouest (l'axe des y) et sud (l'axe des x) de la boîte sont gradués (la graduation est automatique ou est spécifiée par le paramètre nax) ;

  - pour $z = 2$, le graphe est simplement entouré de la boîte (xmin,ymin,xmax, ymax) ; pour les autres valeurs, il n'y a ni boîte, ni axes.
- Le paramètre leg est une chaîne de caractère qui donne une légende (cf $x = 1$) pour chacune des courbes 'y1@y2@...', chaque légende étant séparée de la suivante par le caractère @ ; si on fournit n légendes alors qu'il y a nc courbes ($n < nc$) seules les n premières courbes auront une légende ;
- rect = [xmin,ymin,xmax,ymax] sert à spécifier l'échelle (cf $y = 1$) ;
- nax = [nx,Nx,ny,Ny] sert à choisir les graduations (cf $z = 1$) ; Nx étant le nombre d'intervalles en « x » (on obtient alors Nx+1 valeurs numériques), nx étant le nombre de sous-intervalles par intervalle.

On donne un premier exemple avec presque tous les paramètres (voir figure 1.3) :

```
-->t = linspace(0,2*%pi,60)';

-->x1 = 2*cos(t); y1 = sin(t); // pour une ellipse

-->x2 = cos(t); y2 = y1; // pour un cercle

-->x3 = linspace(-2,2,60)'; y3 = erf(x3); // la fct erreur

-->rect = [-2.05,-1.4,2.05,1.4]; // la fenetre

-->leg="ellipse@cercle@fct erreur"; // les legendes

-->plot2d([x1 x2 x3],[y1 y2 y3],[1:3],"111",leg,rect)
```

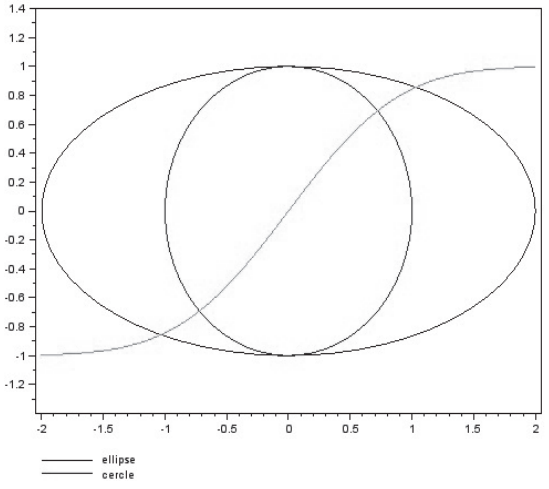


FIG. 1.3 – *Concaténation des trois fonctions*

D'une manière générale, on utilisera l'instruction plot2d de la façon suivante (où

nc est le nombre de courbes) :

```
plot2d(Mx,My,[1:nc],"121","leg_1@....@leg_nc")
```

qui permet de repérer chacune des différentes courbes par une légende, tout en calculant automatiquement l'échelle.

On fait ci-dessous un petit résumé des différentes commandes concernant le traçage des graphiques et la manipulation des axes et des échelles :

- xlabel("temps") : donne un titre à l'axe x,
- ylabel("vitesse") : donne un titre à l'axe y,
- title('evolution de la vitesse') : donne un titre au graphique,
- set(gca(),"grid",[1 1]) : affiche le quadrillage dans le graphique,
- set(gca(),"auto_clear",[-1 -1]) : masque le quadrillage dans le graphique,
- clf : efface le graphique,
- plot(x,[y z w]) : trace y, z et w en fonction de x sur le même graphe,
- polarplot(x,y) : trace la courbe y en fonction de x en coordonnées polaires,
- plot(x,y,'+g') : trace y en fonction de x avec des marques '+' en couleur verte (g relative à « green »),