Initiation à Scilab

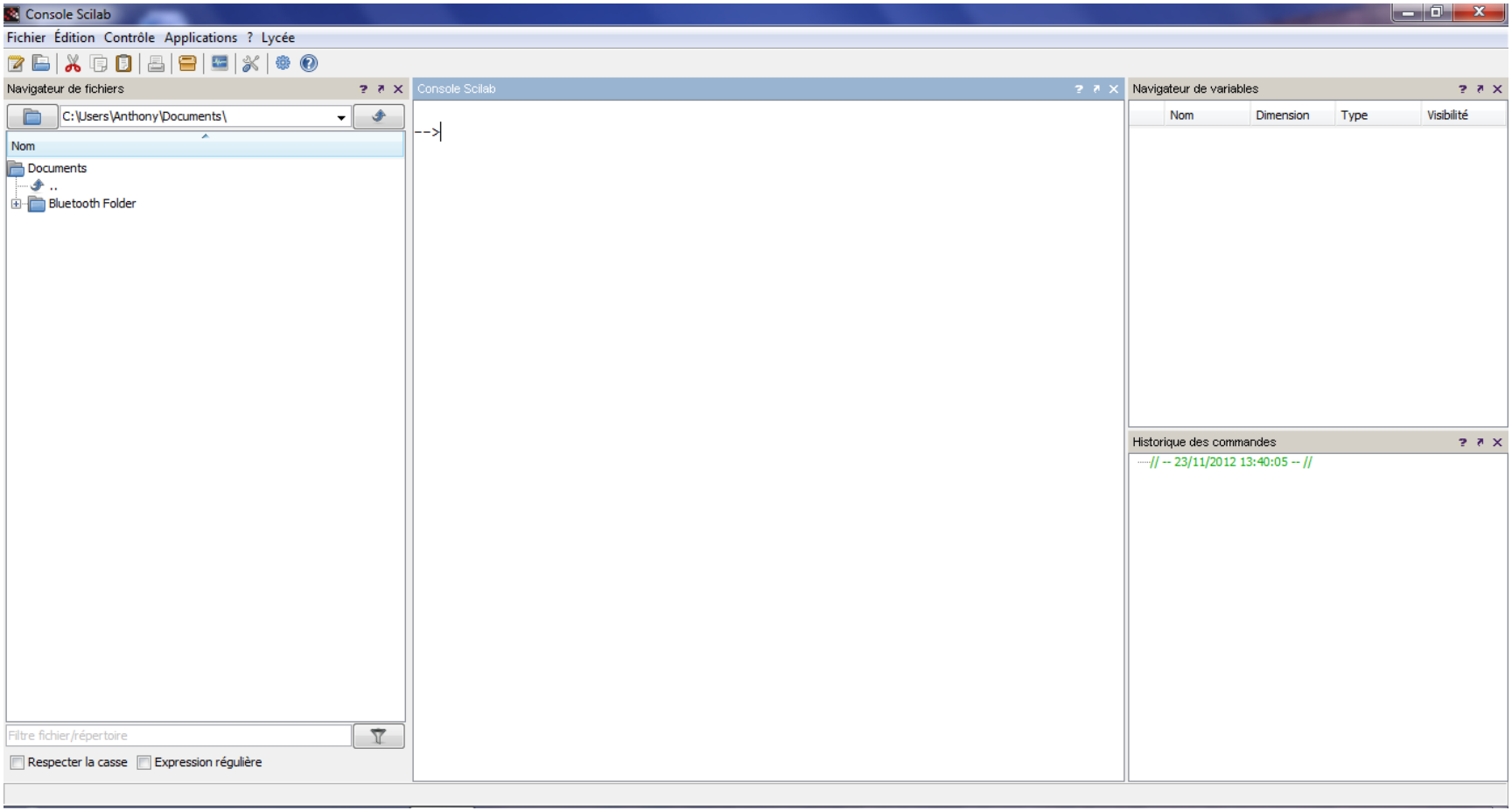
Anthony Mansuy (Académie de Reims)

6 et 13 Février 2013

- Le consortium *Scilab*, constitué d'industriels et de partenaires académiques, a été créé en 2003 à l'initiative de l'Institut National de Recherche en Informatique et en Automatique.
- La société Scilab Enterprises est créée en juin 2010 avec le soutien d'Inria, pour garantir la pérennité du logiciel Scilab. Scilab Enterprises a intégralement pris en charge le développement du logiciel depuis juillet 2012.
- Le but : proposer un outil de calcul numérique libre et gratuit. Il est fait pour les applications des mathématiques par exemple en optimisation, en transformation de Fourier, en statistique...
- La dernière version disponible est *Scilab* 5.4.0, disponible sur le site officiel `http://www.scilab.org/`.

Au lancement de *Scilab*, 4 fenêtres s'ouvrent :

1. Navigateur de fichiers : Il permet d'accéder aux fichiers *Scilab* déjà enregistrés,
2. Console *Scilab* : C'est "la feuille de travail" qui permet de taper des commandes et d'analyser les résultats,
3. Navigateur de variables : Il permet de parcourir et de modifier les variables stockées en mémoire,
4. Historiques des commandes : Toutes les commandes tapées dans la console lors de sessions *Scilab* précédentes sont gardées en mémoire.


Console Scilab
Fichier Édition Contrôle Applications ? Lycée
Navigateur de fichiers
C:\Users\Anthony\Documents\
Nom
Documents
..
Bluetooth Folder
Filtre fichier/répertoire
Respecter la casse
Expression régulière
Console Scilab
-->
Navigateur de variables
Nom
Dimension
Type
Visibilité
Historique des commandes
// -- 23/11/2012 13:40:05 -- //

Scilab fournit une aide intégrée illustrée d'exemples d'utilisation, accessible depuis :

- le menu de la console dans la rubrique " ?",
- la console, en tapant la commande

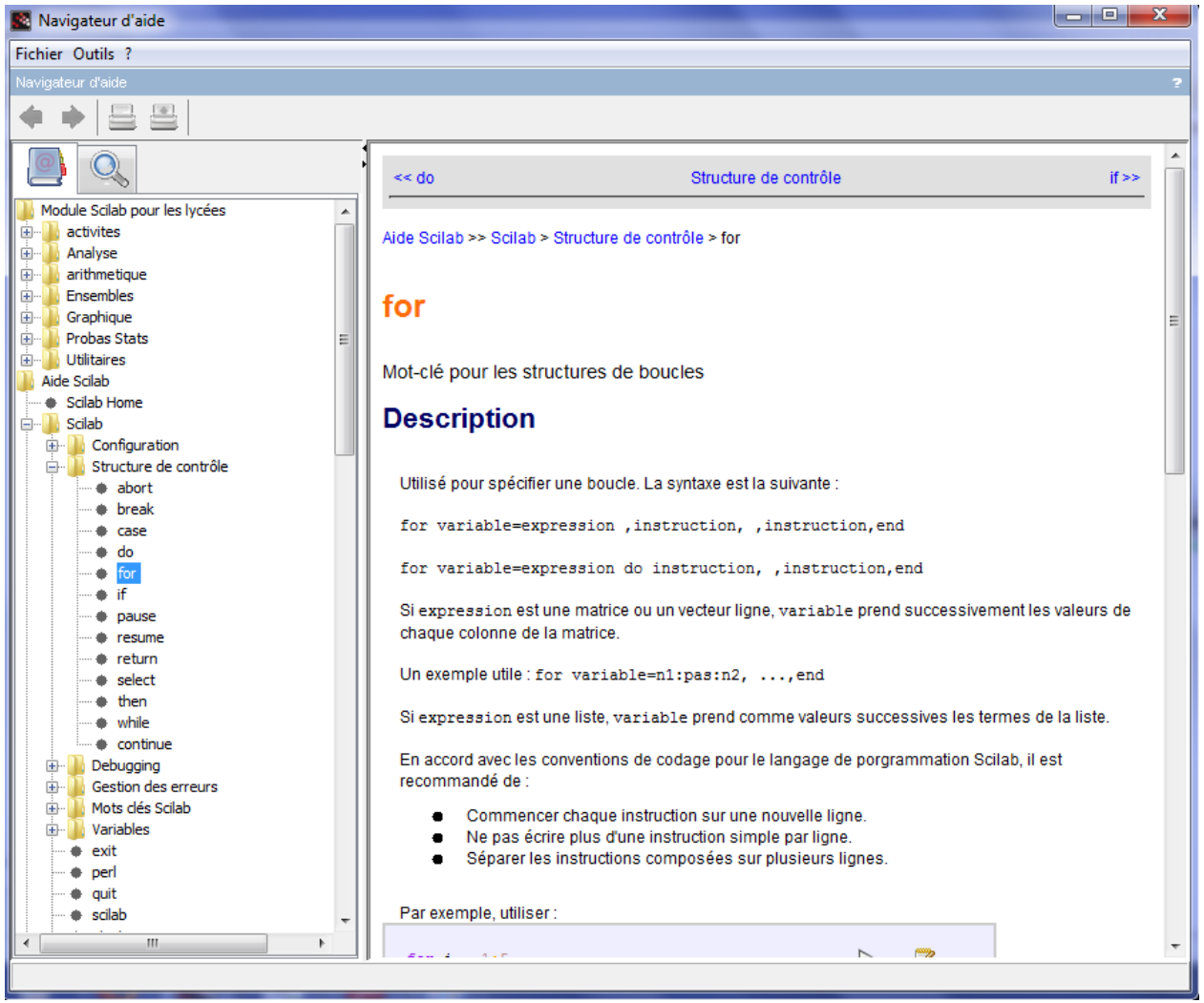
`help` < *commande* >

Exemple

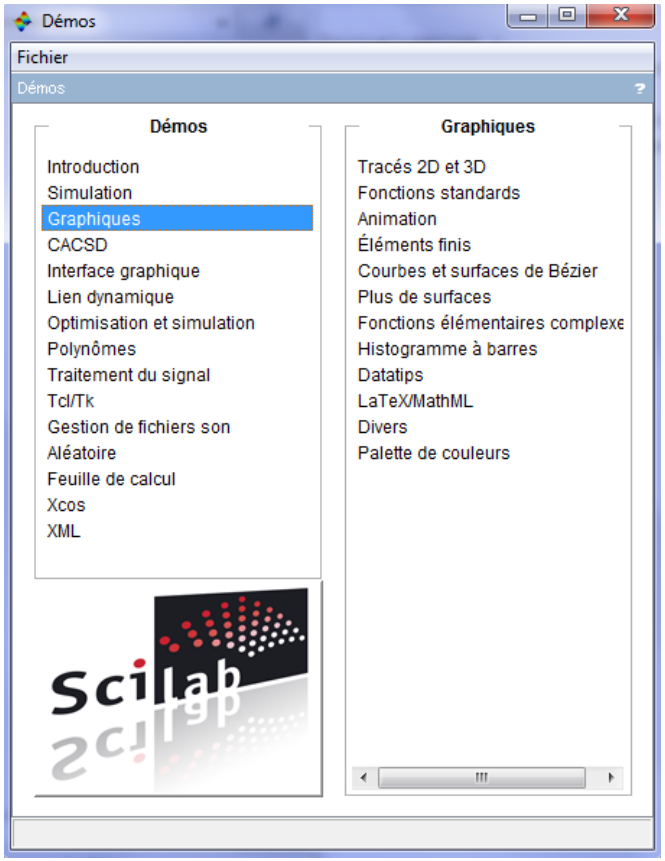
Si on tape la commande

```
help for
```

le navigateur d'aide s'ouvre directement sur la page qui correspond au mot-clef `for`.


Navigateur d'aide
Fichier Outils ?
Navigateur d'aide
Module Scilab pour les lycées
activites
Analyse
arithmetique
Ensembles
Graphique
Probas Stats
Utilitaires
Aide Scilab
Scilab Home
Scilab
Configuration
Structure de contrôle
abort
break
case
do
for
if
pause
resume
return
select
then
while
continue
Debugging
Gestion des erreurs
Mots clés Scilab
Variables
exit
perl
quit
scilab
<< do
Structure de contrôle
if >>
Aide Scilab >> Scilab > Structure de contrôle > for
for
Mot-clé pour les structures de boucles
Description
Utilisé pour spécifier une boucle. La syntaxe est la suivante :
for variable=expression ,instruction, ,instruction,end
for variable=expression do instruction, ,instruction,end
Si expression est une matrice ou un vecteur ligne, variable prend successivement les valeurs de chaque colonne de la matrice.
Un exemple utile : for variable=n1:pas:n2, ...,end
Si expression est une liste, variable prend comme valeurs successives les termes de la liste.
En accord avec les conventions de codage pour le langage de porgrammation Scilab, il est recommandé de :
Commencer chaque instruction sur une nouvelle ligne.
Ne pas écrire plus d'une instruction simple par ligne.
Séparer les instructions composées sur plusieurs lignes.
Par exemple, utiliser :

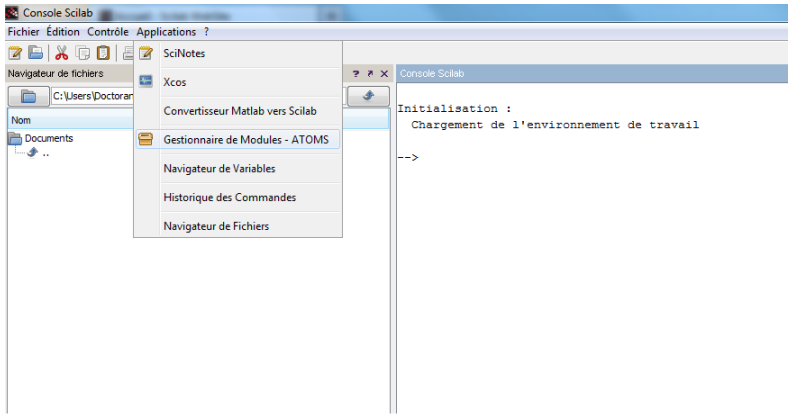
Scilab propose aussi des démonstrations d'utilisation qui permettent de découvrir le logiciel. Celles-ci sont accessibles depuis le menu de la console dans la rubrique " ?".

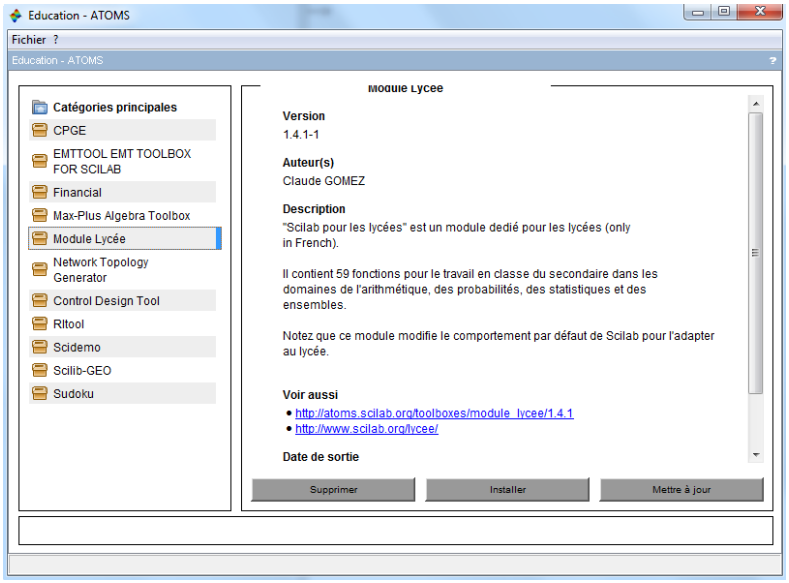


Ce module est spécialement conçu pour une utilisation de *Scilab* au lycée (documentation en ligne : `http://www.scilab.org/lycee/`). Il modifie le comportement par défaut de *Scilab* pour l'adapter au lycée :

- Il modifie le format d'affichage par défaut (plus de décimales).
- Il contient un ensemble de fonctions francisées :
  - en analyse (`ln`)
  - en arithmétique (`pair`,`pgcd`, `reste`...)
  - en dénombrement (`factorielle`,`arrangement`...)
  - utilitaires (`taille`, `trier`...)

L'installation se fait à l'aide du gestionnaire de modules ATOMS : Accessible depuis le menu de la console à la rubrique *Applications*, dans le fichier *Éducation*.



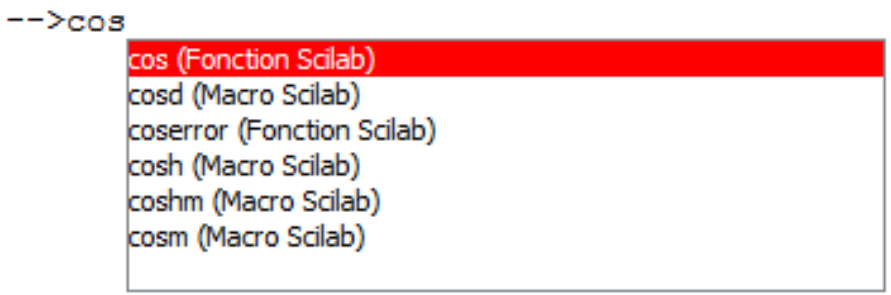


Après installation, le module lycée se charge automatiquement à chaque utilisation de *Scilab*.

Attention : Dans la suite, c'est le comportement standard de *Scilab* qui est décrit sauf indications contraires.

Pour taper des commandes/instructions, on utilise la console. Il existe quelques aides au travail :

- Il est possible de rappeler d'anciennes commandes en pressant les flèches HAUT et BAS.
- Il existe une saisie semi-automatique avec la touche *Tabulation* : Une liste déroulante indique alors toutes les fonctions/variables correspondantes. Par exemple, après la saisie de "cos",



Les intructions sont validées par un *retour à la ligne* ou *retour chariot*.

On peut écrire plusieurs instructions sur une même ligne mais, dans ce cas, chaque instruction doit être suivie par

- une virgule : dans ce cas l'instruction est effectuée et le résultat affiché,
- un point-virgule : dans ce cas l'instruction est effectuée mais le résultat n'est pas affiché.

Pour écrire des commentaires, on utilisera la commande `//` : tout ce qui est à droite de ce symbole ne sera pas "lu" par *Scilab*.

On peut effacer le contenu de la console et de l'historique des commandes à partir du menu dans la rubrique *Édition*.

Gestion des variables

- Pour affecter une valeur à une variable, on utilise le symbole d'affectation `=`.
- Pour effacer une variable de l'environnement de travail, on utilise la commande `clear`.

Le type de la variable est automatiquement défini lors de l'affectation.

Scilab est capable de reconnaître le type d'une variable :

- Les réels
- Les chaînes de caractères
- Les booléens
- Les matrices...

Les réels

Par défaut, toutes les valeurs entrées dans la console sont considérées comme réelles et 10 caractères sont affichés (16 avec le module lycée).

Quelques commandes classiques

- `format` : modifie le nombre de caractères affichés
- `^` : puissance (ex : `2^3`)
- `sqrt` : racine carrée (ex : `sqrt(4)`)
- `round` : arrondi à l'entier le plus proche
- Les fonctions classiques : `exp`, `log`, `cos`, `sin`, `tan`, `acos`, `asin`, `atan`
- `sign` : retourne -1 ou 1
- `rand()` : génère un réel aléatoire $\in [0, 1[$

Les constantes

Les variables précédées d'un % sont des constantes. Elles ne peuvent être modifiées. En voici quelques unes :

`%pi`	$\pi \simeq 3.1415927$
`%e`	$e \simeq 2.7182818$
`%i`	nombre complexe i
`%inf`	$+\infty$
`%nan`	"Not A Number"

Exemple

```
-->0*%inf
 ans =
  Nan
```

```
-->clear %pi
 !--error    13
redefining permanent variable
```

Scilab travaille avec des nombres décimaux approchés. La constante `%eps = 2.220446049D-16` détermine la plus petite précision relative que l'on puisse espérer dans le calcul (environ 16 chiffres).

`%eps` est la plus grande valeur de x telle que *Scilab* considère que $1 + x = 1$.

Exemple

```
-->sin(%pi)
 ans =
   1.224646799D-16
```

La valeur de $\sin(\pi)$ n'est pas nulle... Mais on la considère comme nulle car elle est inférieure à `%eps`.

Les chaînes de caractères

Une chaîne de caractère est un mot délimité par des guillemets.

- La longueur d'une chaîne de caractères est donnée par la commande `length`.
- La concaténation de chaînes de caractères se fait par le symbole $+$.

Exemple

```
-->"sujet "+"verbe "+"complement"
 ans =
sujet verbe complement
```

Les booléens

Une variable booléenne peut prendre deux valeurs vrai ou faux représentées sous *Scilab* par deux constantes `%T` et `%F`.

Voici quelques opérateurs sur les booléens :

- $\sim$: la négation d'un booléen,
- `&` : le ET logique,
- `|` : le OU logique,
- `==`, `<>`, `<`, `>`, `<=`, `>=` : opérateurs de comparaison (utiles pour définir des booléens).

Exemple

```
-->(~(1==2))&(2>3)
 ans =
  F
```

Un vecteur ligne est défini entre crochets, les valeurs séparées par un espace ou une virgule (ex : `[1 2 3]` ou `[1,2,3]`).

Un vecteur colonne est aussi défini entre crochets, les lignes séparées par un point-virgule (ex : `[1;2;3]`).

On définit une matrice en combinant les deux syntaxes (ex : `[1 2 3;4 5 6;7 8 9]`).

Exemple

```
-->[1 2 3;4 5 6;7 8 9]
 ans  =

    1.    2.    3.
    4.    5.    6.
    7.    8.    9.
```

Construction d'une matrice

- `a:c:b` : Vecteur ligne contenant les valeurs comprises entre *a* et *b* avec un pas de *c* éventuellement négatif (par défaut, l'incrément *c* est égal à 1).
- `linspace(a,b,c)` : Vecteur ligne contenant les *c* valeurs équiréparties entre *a* et *b*.
- `zeros(n,m)` (resp. `ones(n,m)`) : Matrice $n \times m$ dont les coefficients sont égaux à 0 (resp. 1).
- `rand(n,m)` : Matrice $n \times m$ de coefficients aléatoires compris entre 0 et 1.
- `diag(u)` : Matrice diagonale dont la diagonale est *u* (avec *u* un vecteur ligne).

Les fonctions `length` et `size` permettent d'obtenir des informations sur la taille d'une matrice A :

- `length(A)` renvoie le nombre d'éléments de A,
- `size(A)` renvoie dans un vecteur ligne le nombre de lignes et de colonnes de A.

On peut accéder à l'élément a_{ij} de A avec la commande $A(i, j)$.

Opérations matricielles

- `A+B` : Addition des matrices A et B
- `A*B` : Multiplication des matrices A et B
- `inv(A)` : Inverse d'une matrice A inversible
- `A.'` : Matrice transposée de A
- `A'` : Matrice adjointe (ou transconjuguée) de A
- `det(A)` : Déterminant de A

En particulier, `X'*Y` est le produit scalaire de deux vecteurs colonnes X et Y.

Opérations sur les coefficients d'une matrice

- Si A et B sont deux matrices $n \times m$, les instructions $C = A.*B$, $D = A./B$ et $E = A.\wedge B$ sont respectivement définies par

$$c_{ij} = a_{ij} * b_{ij}, \qquad d_{ij} = \frac{a_{ij}}{b_{ij}}, \qquad e_{ij} = a_{ij}^{b_{ij}}.$$

- Les fonctions numériques usuelles s'appliquent aussi aux matrices : si f est une fonction *Scilab* et A une matrice, $f(A) = (f(a_{ij}))_{ij}$.

Pour représenter des fonctions, on utilisera la fonction `plot` :

- Si x et y sont deux vecteurs de même longueur, $plot(x, y)$ trace y en fonction de x.
- A l'exécution de la commande `plot`, une fenêtre graphique s'ouvre automatiquement avec la courbe demandée.

Il est possible de tracer plusieurs courbes en même temps.

Exemple

```
x=linspace(-%pi/2,%pi/2,100);
plot(x,cos(x),x,sin(x),x,x^2);
```

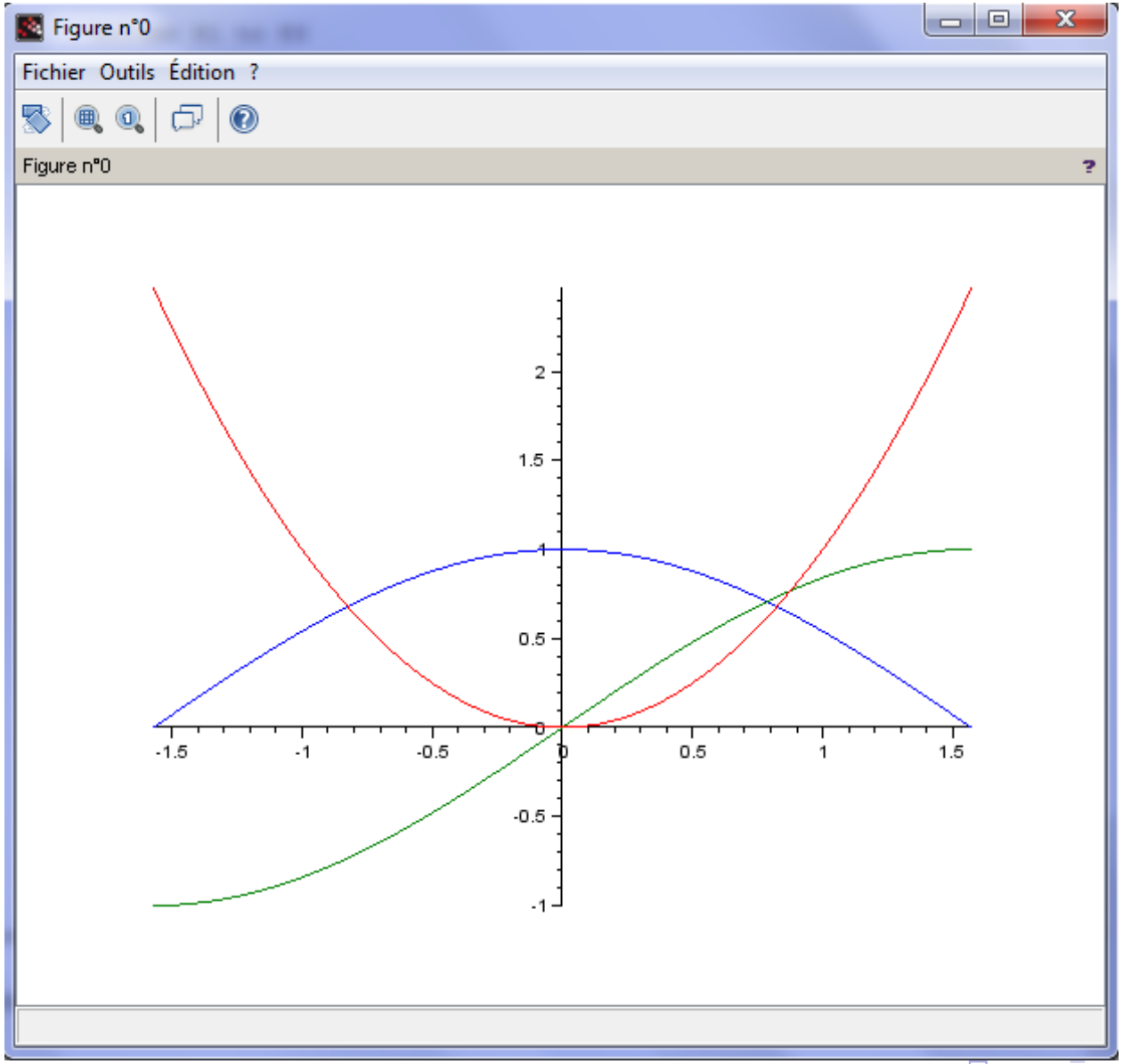


Figure n°0
Fichier Outils Édition ?
Figure n°0
2
1.5
1
0.5
0
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

Remarquons que si plusieurs tracés sont demandés, ils sont placés dans la même figure (et chacune a une couleur différente). Par défaut :

- Les axes sont affichés
- L'échelle est configurée automatiquement (en fonction des valeurs)
- Les tracés sont réalisés sous forme de lignes

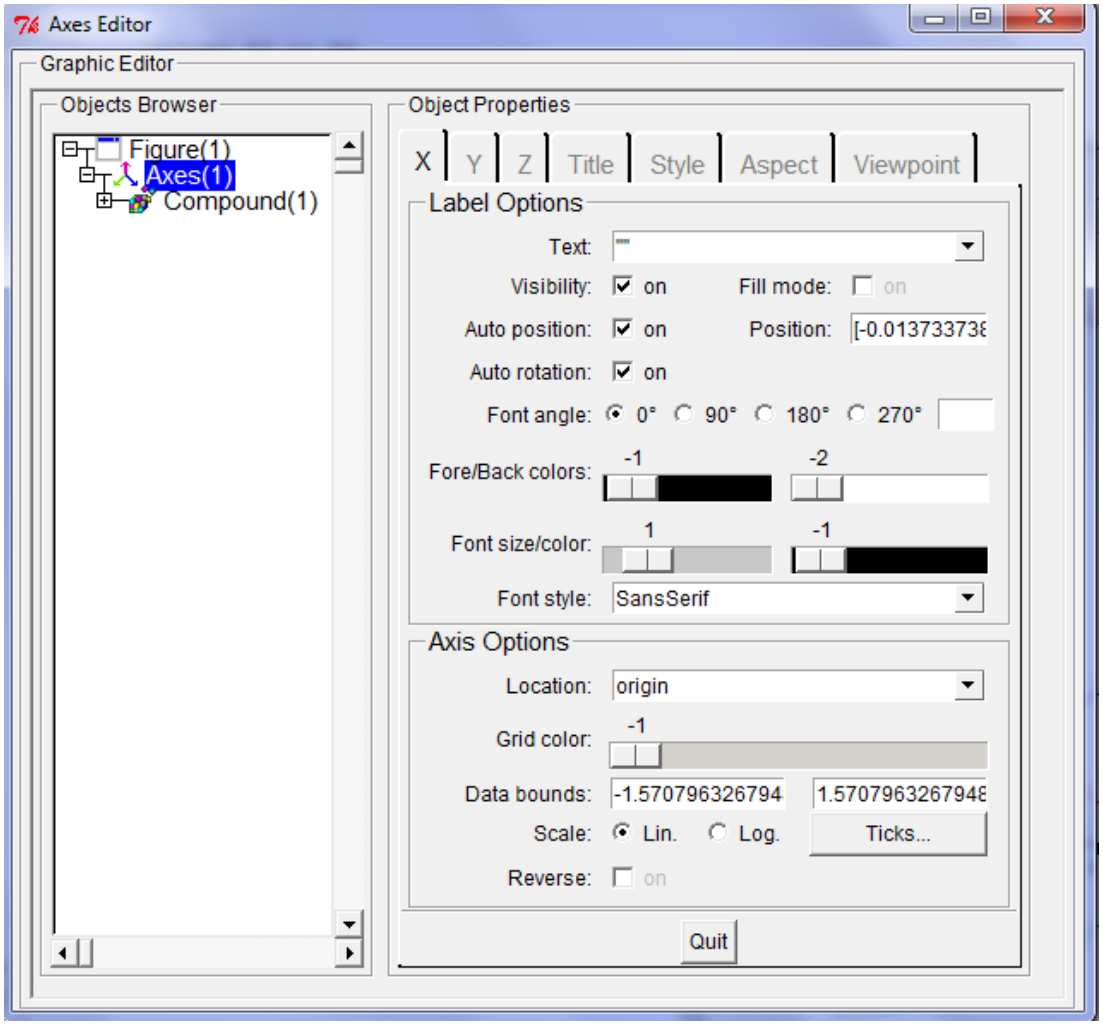
Notons l'existence de la fonction `clf` qui permet d'effacer les courbes de la figure courante.

On peut enregistrer les tracés au format *Scilab* ou les exporter sous forme d'images.

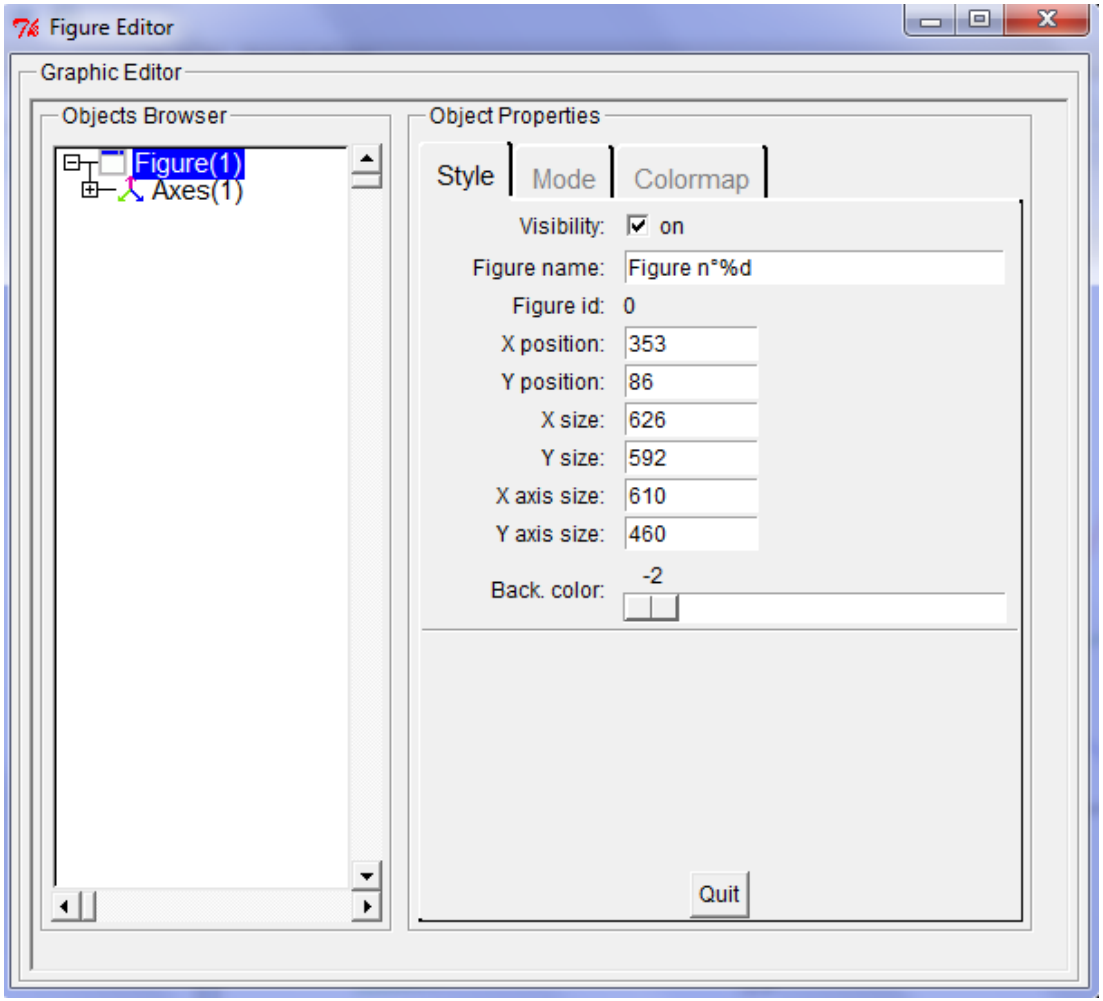
Depuis la fenêtre graphique à la rubrique *Edition*, il est possible de configurer un ensemble d'options :

- les axes (échelles, unités...) avec la rubrique *Propriétés des axes*,
- les courbes (style du tracé, couleur...) avec la rubrique *Propriétés de la figure*.

Éditeur d'axes



Éditeur de figure



Il est aussi possible de modifier directement l'apparence des tracés dans la fonction `plot` en ajoutant un troisième paramètre : une chaîne de caractère correspondant à

- la couleur,
- le style de la courbe,
- les marqueurs.

$\Rightarrow$ Les caractères pour les couleurs :

Caractère	Couleur	Caractère	Couleur
r	rouge	g	vert
b	bleu	c	cyan
m	magenta	y	jaune
k	noir	w	blanc

⇒ Les caractères pour le style de la courbe :

Caractère	Style	Caractère	Style
−	ligne simple	:	pointillés

⇒ Les caractères pour les marqueurs :

Caractère	Marqueur	Caractère	Marqueur
+	plus	*	astérisque
o	rond	.	point
x	croix	s	carré

Exemple

```
clf;
x=linspace(-%pi/2,%pi/2,100);
plot(x,cos(x),"-g",x,sin(x),":b",x,x^2,"-xr")
```

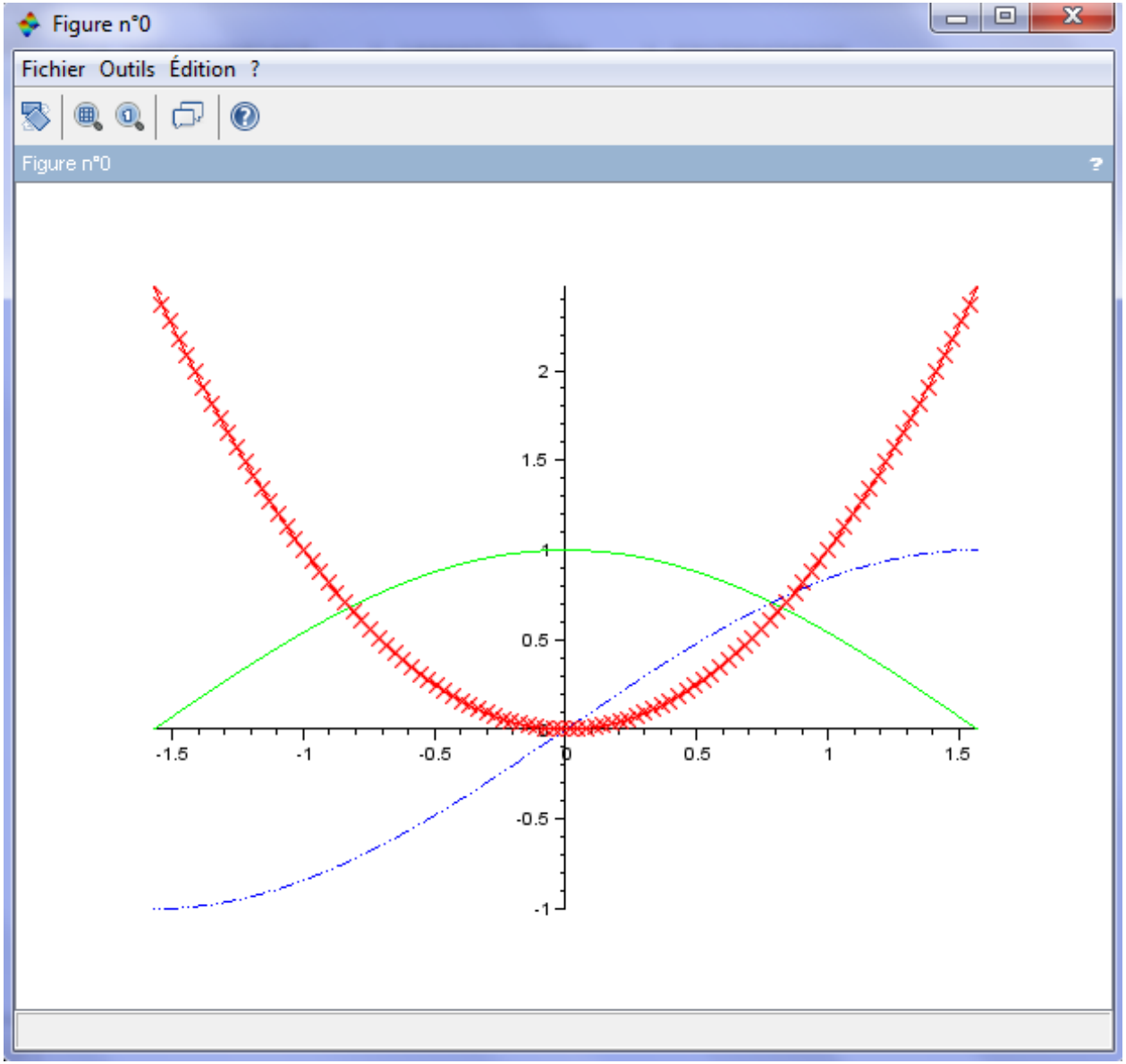


Figure n°0
Fichier Outils Édition ?
Figure n°0
2
1.5
1
0.5
-0.5
-1
-1.5
-1
-0.5
0
0.5
1
1.5

On peut ajouter un titre à un graphique avec la commande `xtitle(titre,[titrex,titrey])` :

- `titre` est une chaîne de caractères qui servira de légende au graphique
- `titrex` et `titrey` désignent respectivement les légendes de l'axe des abscisses et des ordonnées.

Enfin, dans le cas où on fait plusieurs tracés, il est possible de séparer les figures en utilisant la fonction `set`.

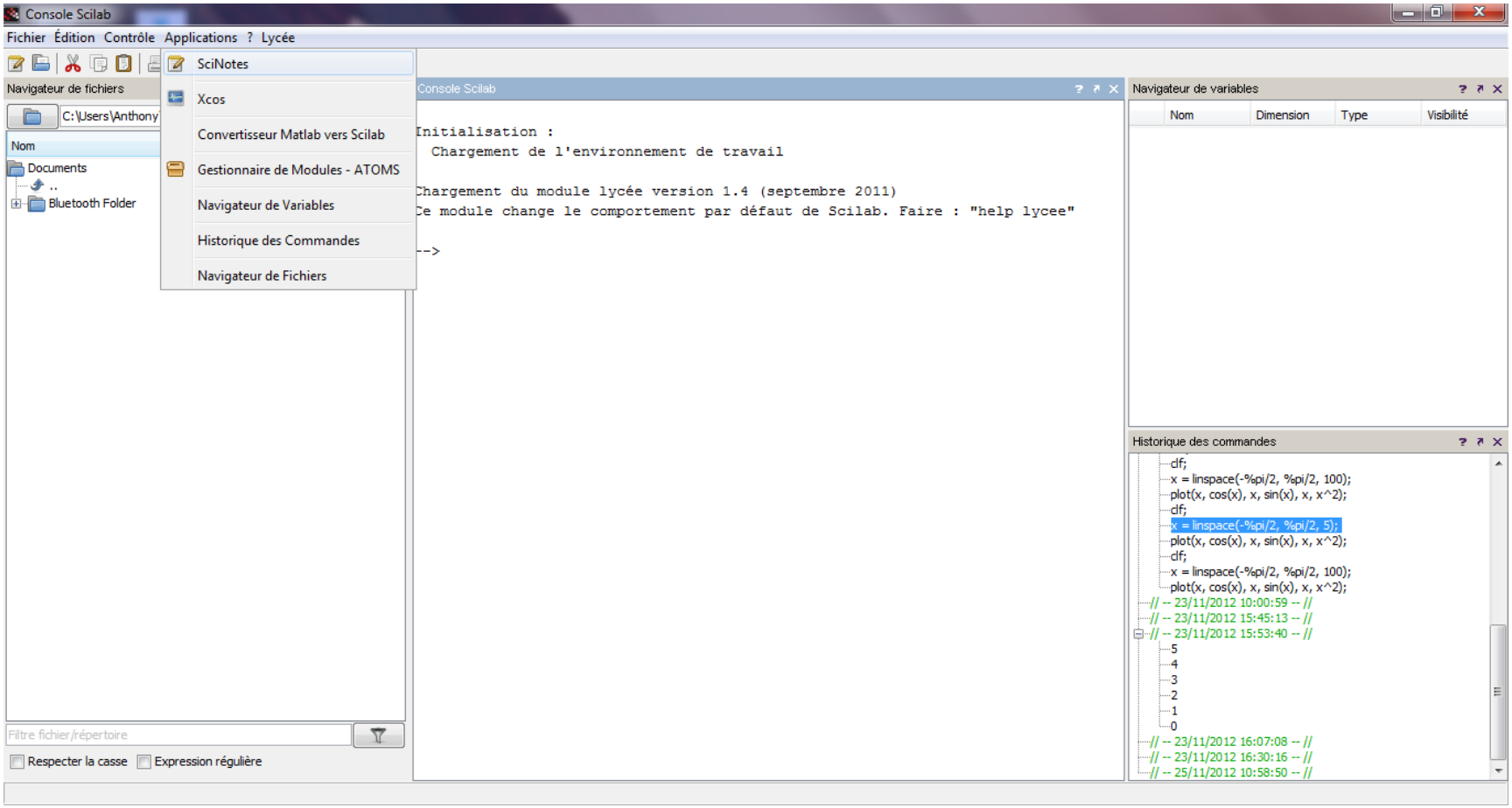
Exemple

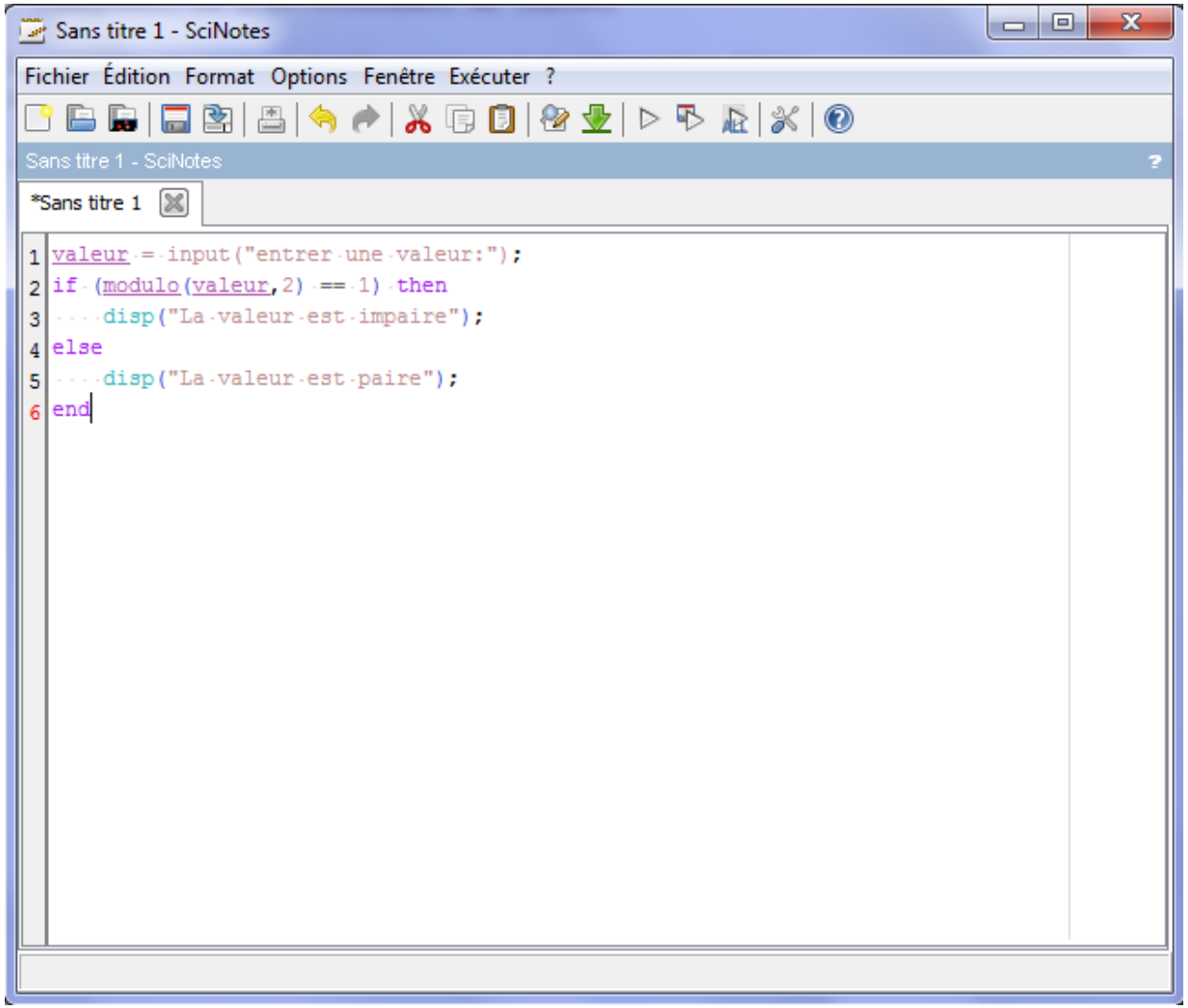
```
x=linspace(-2,2,100);
set("current_figure",1);
plot(x,x^2);
set("current_figure",2);
plot(x,x^3);
```

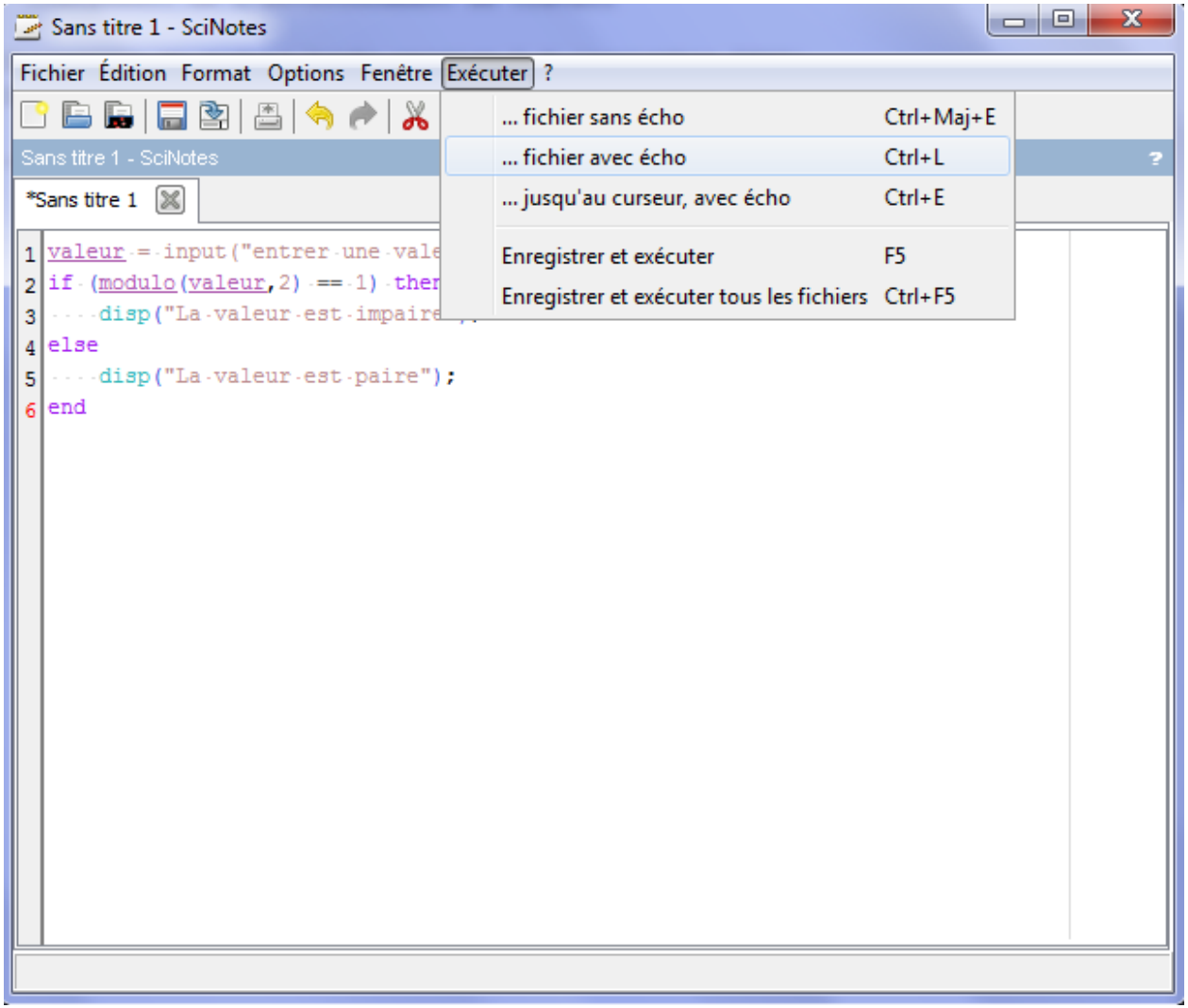
Un éditeur de texte, SciNotes, est intégré à *Scilab*.

- Il permet de créer et d'éditer des programmes *Scilab*.
- Il reconnait le langage *Scilab* (mise en couleur des mots-clef, indentation automatique du code).
- Il permet d'utiliser directement le code dans la console.

L'éditeur de texte SciNotes est accessible depuis la console dans la rubrique *Applications*.


Console Scilab
Fichier Édition Contrôle Applications ? Lycée
SciNotes
Xcos
Convertisseur Matlab vers Scilab
Gestionnaire de Modules - ATOMS
Navigateur de Variables
Historique des Commandes
Navigateur de Fichiers
Navigateur de fichiers
C:\Users\Anthony
Nom
Documents
Bluetooth Folder
Filtre fichier/répertoire
Respecter la casse
Expression régulière
Console Scilab
Initialisation :
Chargement de l'environnement de travail
Chargement du module lycée version 1.4 (septembre 2011)
Ce module change le comportement par défaut de Scilab. Faire : "help lycee"
-->
Navigateur de variables
Nom
Dimension
Type
Visibilité
Historique des commandes
clf;
x = linspace(-%pi/2, %pi/2, 100);
plot(x, cos(x), x, sin(x), x, x^2);
clf;
x = linspace(-%pi/2, %pi/2, 5);
plot(x, cos(x), x, sin(x), x, x^2);
clf;
x = linspace(-%pi/2, %pi/2, 100);
plot(x, cos(x), x, sin(x), x, x^2);
// -- 23/11/2012 10:00:59 -- //
// -- 23/11/2012 15:45:13 -- //
// -- 23/11/2012 15:53:40 -- //
5
4
3
2
1
0
// -- 23/11/2012 16:07:08 -- //
// -- 23/11/2012 16:30:16 -- //
// -- 25/11/2012 10:58:50 -- //


Sans titre 1 - SciNotes
Fichier Édition Format Options Fenêtre Exécuter ?
Sans titre 1 - SciNotes
*Sans titre 1
1 valeur = input("entrer une valeur:");
2 if (modulo(valeur,2) == 1) then
3 disp("La valeur est impaire");
4 else
5 disp("La valeur est paire");
6 end


Sans titre 1 - SciNotes
Fichier Édition Format Options Fenêtre Exécuter ?
... fichier sans écho Ctrl+Maj+E
... fichier avec écho Ctrl+L
... jusqu'au curseur, avec écho Ctrl+E
Enregistrer et exécuter F5
Enregistrer et exécuter tous les fichiers Ctrl+F5
Sans titre 1 - SciNotes
*Sans titre 1
1 valeur = input("entrer une vale
2 if (modulo(valeur,2) == 1) ther
3 disp("La valeur est impaire
4 else
5 disp("La valeur est paire");
6 end

Le fichier doit être enregistré avant son exécution. Il existe deux modes d'exécution :

- fichier sans écho : *Scilab* exécute les instructions dans la console en affichant uniquement les entrées/sorties.
- fichier avec écho : *Scilab* exécute les instructions dans la console en affichant chaque ligne de code. Ce mode est utilisé lors d'un processus de débogage.

Il est possible de définir ses propres fonctions. Dans ce cas, les instructions sont entrées :

- soit dans l'éditeur de texte,
- soit directement dans la console

Structure générale

```
function res=nom(arg_1,...,arg_n)
  instructions
endfunction
```

- $arg_1, \ldots, arg_n$ sont les arguments appelés dans la fonction
- *res* est le résultat de la fonction affiché après exécution

Pour appeler la fonction, on utilise la commande

$$nom\,(arg_1, \ldots, arg_n)$$

- Avantage : Une fois chargée, la fonction peut être directement utilisée dans la console ou dans un script.
- Remarque : Dans le cas où la fonction à définir est compliquée, on préférera utiliser l'éditeur de texte dont l'utilisation est plus simple et plus souple par rapport à la console.

Exemple : Une fonction permettant de tester la parité

```
function resultat=estPair(n)
  if modulo(n,2)==0 then
    resultat=%T;
  else
    resultat=%F;
  end
endfunction
```

On réservera l'utilisation de la console pour définir des fonctions simples, comme par exemple des fonctions polynomiales.

Exemple

Considérons la fonction $f(x) = x^2 + x - 1$.

```
function y=f(x)
  y=x^2+x-1;
endfunction
```

On peut alors effectuer le tracé de $\mathcal{C}_f$:

```
clf;
x=linspace(-3,3,50);
plot(x,f);
```

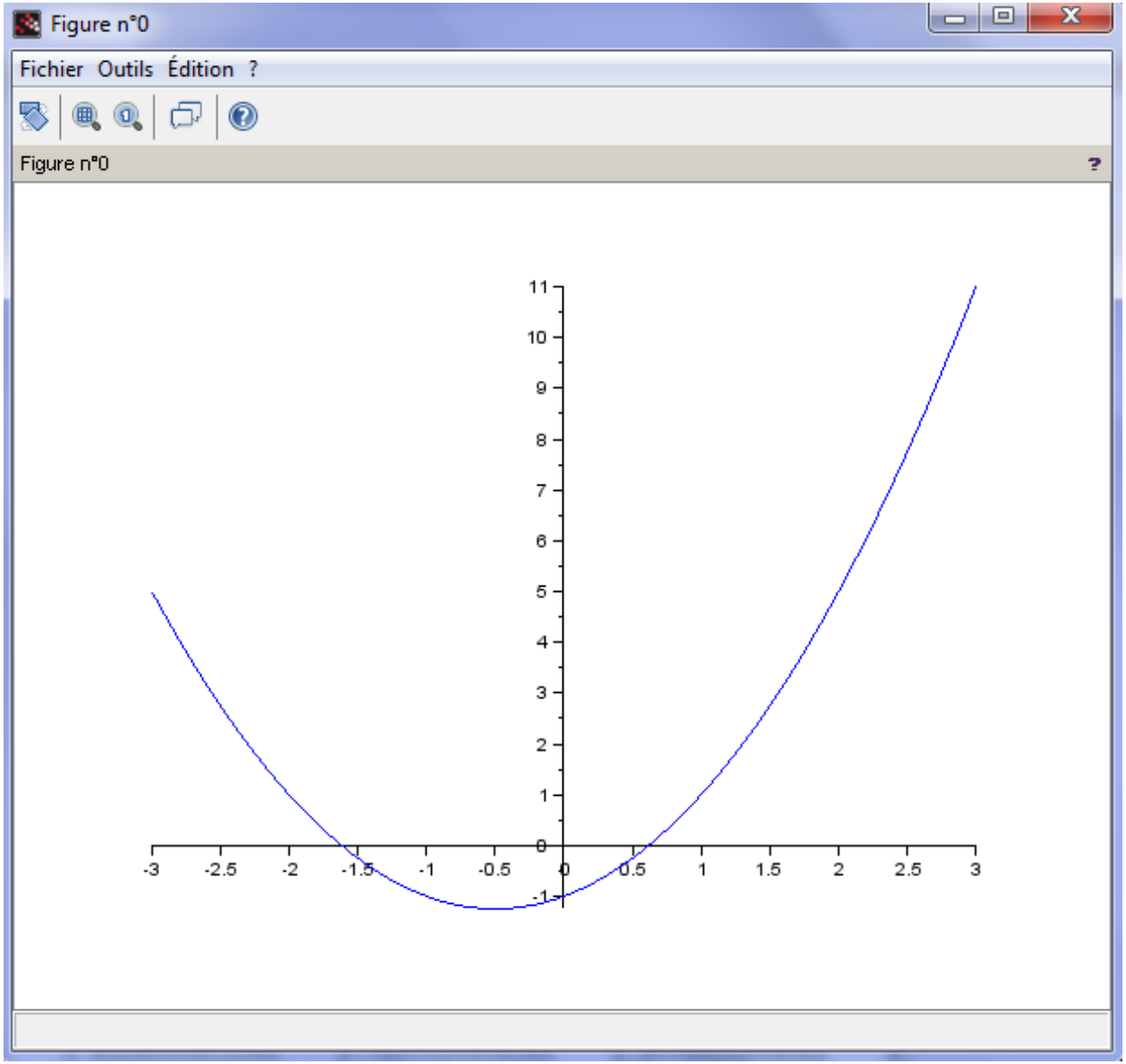


Figure n°0
Fichier Outils Édition ?
Figure n°0
11
10
9
8
7
6
5
4
3
2
1
0
-1
-3
-2.5
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
2.5
3

Pour les entrées et les sorties, on utilisera les deux fonctions suivantes :

- `input("texte")` : permet de saisir une valeur après l'affichage dans la console du *texte*.
- `disp(`$var_1, \ldots, var_k$`)` : affichage dans la console de $var_k, \ldots, var_1$ (Attention à l'ordre des variables).

Exemple

```
x=input("Saisissez x");
y=input("Saisissez y");
disp(x+y,x*y);
```

Structures conditionnelles

Boucle *if*

```
if condition_1 then
  instruction_1 ;
elseif condition_2 then
  instruction_2 ;
...
else
  instruction_k
end
```

`elseif` et `else` sont facultatifs. Les *conditions* sont des expressions booléennes. Seules les instructions correspondant à la première condition vérifiée sont exécutées.

Exemple

Une procédure avec en entrée a et b et en sortie $max(a, b)$:

```
a=input("Saisissez a:");
b=input("Saisissez b:");
if a>b then
  disp(a);
else
  disp(b);
end
```

Voici un deuxième exemple où on veut définir une fonction définie par morceaux.

Exemple

```
function y=f(x)
  if x<0 then
    y=sin(x);
  else
    y=sqrt(x);
  end
endfunction
clf;
x=linspace(-%pi/2,%pi/2,50);
plot(x,f);
```

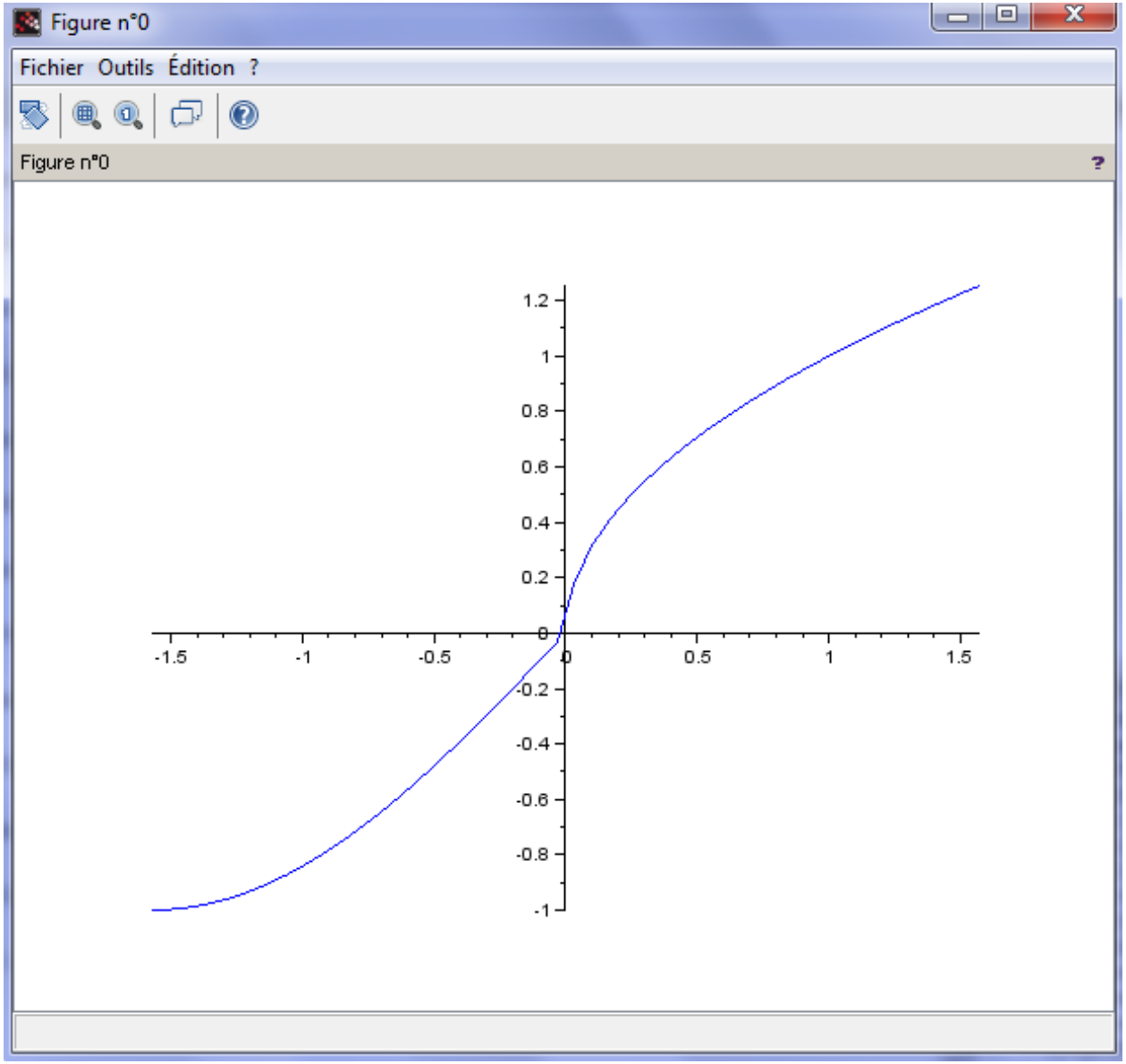


Figure n°0
Fichier Outils Édition ?
Figure n°0
?
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.5
-1
-0.5
0
0.5
1
1.5

Structures itératives

Boucle *for*

```
for var = ini : pas : fin do
  instruction_1 ;
  ...
end
```

var est la variable de la boucle, *ini* la borne de départ, *pas* le pas d'incrémentation (il est facultatif et vaut par défaut 1) et *fin* la borne d'arrivée.

Les *instructions* sont exécutées un nombre déterminé de fois (dépendant de *ini*, *pas* et *fin*) inconditionnellement.

Signalons l'existence de la commande `break` qui permet de sortir d'une boucle.

Exemple

Une procédure avec en entrée *a* et *b* et en sortie les nombres premiers compris entre *a* et *b* (la fonction `premier` fait partie du module lycée) :

```
a=input("Saisissez a:");
b=input("Saisissez b:");
for i=a:b do
  if premier(i) then
    disp(i);
  end
end
```

Boucle *while*

```
while condition do
  instruction_1 ;
  ...
end
```

La *condition* est une expression booléenne. Les *instructions* sont exécutées tant que la *condition* est vraie.

Attention : Il faut veiller à ce que la boucle se termine (et donc que la *condition* soit fausse) en un nombre fini d'étapes.
On peut *Interrompre* ou *Abandonner* une procédure à partir du menu de la console dans la rubrique *Contrôle*.

Exemple

Une procédure avec en entrée *a* et *b* et en sortie les nombres premiers compris entre *a* et *b* :

```
a=input("Saisissez a:");
b=input("Saisissez b:");
i=a;
while i<=b do
  if premier(i) then
    disp(i);
  end
  i=i+1;
end
```

Références

- Le site officiel de *Scilab* :

  `http://www.scilab.org`

- Le site officiel du module lycée :

  `http://www.scilab.org/lycee/`

  On y trouve des livrets d'exercices corrigés pour les classes de Lycée ainsi qu'une présentation détaillée de toutes les commandes *Scilab* utiles au Lycée.