Scilab

Skander Zannad et Judicaël Courant

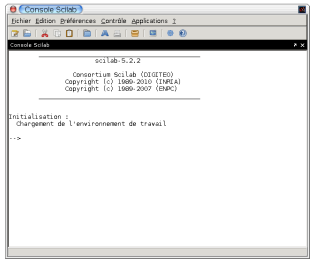
2014-01-18

1 Scilab

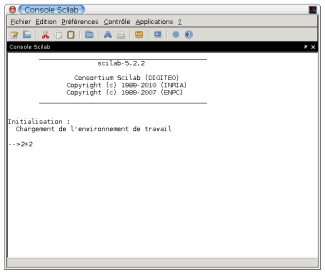
1. Logiciel libre.
2. Usages :
   - calcul numérique
   - modélisation problèmes SI
   - ~~programmation générale~~
   - ~~calcul algébrique~~.
3. Proche de Matlab (logiciel privateur).
4. Fonctionne sur de nombreux systèmes d'exploitation.

2 Console Scilab

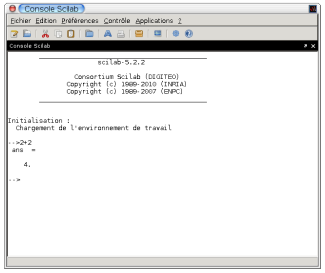
Est à Scilab ce que le shell python est à Python dans IDLE.



On entre une expression :



Et Scilab répond :



(`ans` : abréviation de *answer*)

3 Données manipulées

Essentiellement : Scilab manipule des matrices.

```
-->x = [3 5 7 11] // vecteur ligne
 x  =

    3.    5.    7.    11.

-->y = [1, 4, 9, 16] // autre façon d'écrire
 y  =

    1.    4.    9.    16.

-->z = [11; 22; 33; 44] // vecteur colonne
 z  =

    11.
```

```
    22.
    33.
    44.

-->x+y
 ans  =

    4.    9.    16.    27.

-->x-y
 ans  =

    2.    1.  - 2.  - 5.

-->x + z
      !--error 8
Addition incohérente.
```

Addition/soustraction matricielle : addition/soustraction terme à terme, notés +/-.
Multiplication :

- terme à terme : `.*`
- matricielle : `*`

Division :

- terme à terme : `./`
- «matricielle» : `a / b` «solution» de `x * b = a`.

3.1 Accès aux coefficients

```
-->x(1,2)
 ans  =

    5.

-->z(2,1)
 ans  =

    22.

-->x(2)
 ans  =

    5.
```



```
-->z(2)
 ans  =

    2.
```

Affectation d'une case d'un tableau :

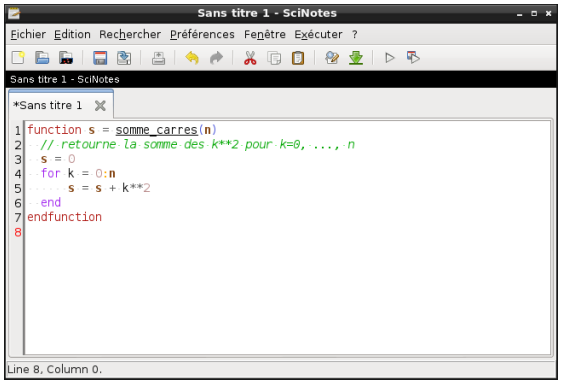
```
-->x(1,2) = 33
 x  =

    3.    33.    7.    11.
```

4 Programmation

4.1 Éditeur

Scilab dispose d'un éditeur (SciNotes), qu'on peut ouvrir avec le menu Applications/SciNotes :



On peut sauvegarder le contenu de l'éditeur sous forme d'un fichier (menu «Fichier/Enregistrer») et en transférer le contenu dans la console (F5 ou menu «Exécuter/Enregistrer et lancer»).

4.2 Syntaxe

En python : indentations significatives.
En Scilab :

1. Indentations non significatives (mais utiles pour le lecteur)
2. Les blocs d'instructions délimités explicitement.

Exemples :

```
u = 0
v = 1
for k = 0:n-1
  // u == Fibonacci(k) et v == Fibonacci(k+1)
  w = u + v
  u = v
  v = w
end
// ici u == Fibonacci(n)

u0 = 42
k = 0
u = u0
while u <> 1
  // u est le terme de rang k de la suite de Syracuse
 // commençant à u0
  if pmodulo(u, 2) == 0 then
    u = 3*u + 1
  else
    u = u / 2
  end
end
```

Fonctions :

```
function [m, v] = stat(t)
  // retourne la moyenne m et la variance v
  // des valeurs du tableau t
  n = length(t)
  s = sum(t)
  m = s / n
  s2 = sum(t .** 2) / n
  v = s2 - s**2
endfunction
```

5 Aide

Une aide très fournie est disponible par le menu « ?/Aide de Scilab» ou par la commande «help» («`help` *sujet*»).

6 Utilisation pour des simulations numériques

Scilab sait notamment résoudre numériquement des équations différentielles de la forme $y' = F(t, y(t))$ avec une condition initiale $y(t_0) = y_0$.

Exemple : résolution numérique de l'équation différentielle

$$y' = y^2 + cos(\pi t) \tag{1}$$

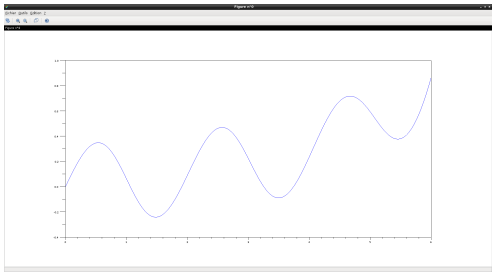
avec la condition initiale $y(0) = 1$.

En scilab :

```
function yp = F(t, y)
  yp = y ** 2 + cos(%pi*t)
endfunction

// conditions initiales
t0 = 0
y0 = 0

// on veut la valeur de y en 0, 0.01, 0.02, ..., 6 :
t = 0:0.01:6
// vecteur des valeurs de la solution calculée :
y = ode(y0, t0, t, F)
// affichage :
plot(t, y)
```



Annexe : Un regard critique

Certains choix de conception sont manifestement erronés :

1. Gestion des variables locales conduisant à des captures de variables.
2. Les tableaux ne sont pas des vrais tableaux dynamiques : l'ajout d'un élément coûte toujours un temps proportionnel à la taille du tableau.

3. Les indices de tableau commencent à 1.
4. Le choix d'un langage très spécifique.

Explication pour les trois premiers points : Scilab a été écrit par des numériciens, pas par des spécialistes de la programmation [1].
Pour le dernier :

1. Volonté de compatibilité (partielle) avec Matlab.
2. Méconnaissance des langages existants qui auraient pu être utilisés [2] ?
3. Sous-estimation du coût du développement d'un langage ?

Remarque :

1. Il existe de nombreux outils avec des langages spécialisés : Maple, Mathematica, Maxima, XCas, shell-scripts, ...
2. Dans presque tous les cas, le langage est difficile à manipuler dès qu'on sort de son domaine d'application.
3. Aujourd'hui, on sait construire des langages généralistes suffisamment souples pour remplacer ces outils par des bibliothèques du langage. Intérêt : moins de langages à apprendre, puissance plus grande de ces langages, facilité d'ajout d'un nouveau domaine d'application.
4. Exemple avec Python : scipy pour remplacer Matlab/Scilab, sympy ou SAGE pour remplacer Maple, Mathematica, Maxima et XCas.

1. Un informaticien spécialiste de la programmation est à un programmeur ce qu'un linguiste est à une personne polyglotte.
2. Python était trop jeune à l'époque mais on aurait pu envisager d'autres langages, par exemple LISP.