Une introduction à Scilab

Algèbre matricielle numérique

Math 307

A. Matos, C. Roland, A. Tinzefté, F. Wielonsky

September 27, 2005

Scilab 3.0 est un logiciel libre, téléchargeable à l'adresse : <http://scilabsoft.inria.fr/>
Une documentation (114 pages) est téléchargeable à l'adresse :
www.iecn.u-nancy.fr/~pincon/scilab/docA4.pdf

Scilab est un logiciel de calcul scientifique développé par l'INRIA et l'ENPC depuis 1990. Comme Matlab (Matrix Laboratory), Scilab est basé sur la philosophie selon laquelle toute expérimentation numérique peut s'écrire et être programmée à l'aide du calcul matriciel. Scilab est un langage interprété, donc a priori moins performant qu'un langage compilé comme Fortran ou C. Mais Scilab est également un langage évolué. Cela permet d'éviter la programmation de toutes les routines de base, ce qui le rend un outil indispensable dans les domaines de la simulation et de la modélisation.

Le logiciel Scilab accepte des commandes sous deux formes possibles :

- de façon interactive au clavier dans la fenêtre de base
- sous la forme d'une ou plusieurs fonctions écrites dans un ou des fichiers de commande (scripts - nom-de-fichier.sce).

Remarques :

- Dans la fenêtre de commandes:

- le bouton " Help" permet d'obtenir entre autres une documentation des commandes et l'item "A propos" permet d'obtenir de l'aide sur une commande que l'on tape dans la nouvelle fenêtre qui s'ouvre . A utiliser aussi souvent que possible.
- le bouton "Editor" fait appel à un éditeur qui vous permettra d'écrire vos scripts
- "Demos" permettra d'exécuter une démonstration de Scilab (conseillé pour se rendre compte des possibilités du logiciel)

- Les flèches tapées dans la fenêtre de commande permettent de se déplacer dans l'historique des instructions précédentes.

- La suite de symboles // met une ligne en commentaire

Premières commandes

<code>sqrt(-2)</code>	Scilab connaît les nombres complexes
<code>2+3; 2*3; 2^3; 2**3</code>	Les opérations élémentaires
<code>2+3</code>	Quelques commandes à essayer
<code>2+3;</code>	
<code>2+3; 3+4</code>	
<code>2+3, 3+4</code>	
<code>a=1, clear a, a</code>	La fonction clear désaffecte la variable a
<code>sin(2), abs(-2), imag(2+%i),</code>	Les fonctions élémentaires
<code>help eye</code>	Aide en ligne
<code>help matrice</code>	

Construction de vecteurs et de matrices; opérations de base

<code>x=2:6</code>	Un vecteur
<code>0:2:8</code>	
<code>[0:2:8]</code>	
<code>[5,3,2]</code>	
<code>linspace(0,20,5)</code>	vecteur de 5 points équidistribués entre 0 et 20
<code>A=[2,3;4,5;6,7]</code>	Une matrice de type (3,2), chaque lignes séparées par des “;”
<code>B=zeros(5,20)</code>	matrice de zéros
<code>C=ones(2,5)</code>	matrice de 1
<code>D=eye(2,5)</code>	matrice identité de taille (2,5)
<code>diag([0:2:8])</code>	matrice diagonale
<code>rand(3,4)</code>	génère une matrice de taille (3,4) de nombres aléatoires compris entre 0 et 1
<code>size(B), size(B,1), size(B,2)</code>	dimensions de B, nombre de lignes, nombre de colonnes
<code>A'</code>	Transposée
<code>C+D, E=A*A', inv(E)</code>	Addition, produit et inverse matriciels
<code>trace(E)</code>	trace de A
<code>spec(E), rank(E), det(E)</code>	les valeurs propres, le rang, le déterminant
<code>clean(spec(E))</code>	arrondi à zéro les nombres inférieurs en valeur absolue à $10^{(-10)}$
<code>cond(E)</code>	le conditionnement
<code>norm(E,1), norm(E,'inf'), norm(E,'fro')</code>	différentes normes matricielles
<code>triu(C)</code>	extraît la partie triangulaire supérieure de la matrice C
<code>tril(C)</code>	extraît la partie triangulaire inférieure de la matrice C

<code>E(1,2)</code>	l'élément E_{12}
<code>E(:,3), E(3,:)</code>	troisième colonne de E, troisième ligne de E
<code>E(:,1:2)</code>	les deux premières colonnes
<code>E(:, \$)</code>	la dernière colonne de E
<code>E([1,2],[2,3])</code>	sous-matrice de taille 2×2
<code>diag(E)</code>	vecteur colonne des éléments diagonaux de E
<code>diag(diag(E))</code>	
<code>E([2,1,3],:)</code>	permutation des 2 premières lignes de E

D'autres opérations sur les matrices

Certaines opérations s'appliquent sur les matrices (par ex. la multiplication si les dimensions sont convenables) et certaines opérations n'ont pas de sens (par ex. la division). Si on veut appliquer ces opérations terme à terme, on les précède par un point. Voici quelques exemples.

```
v=[1,2,3]
v/2
v./2
w=[1,2,3]
v./w
v./(1:1:3)
v./(1:3)
v(1:1:3)./(1:1:3)
(0:0.1:1).^2
v/w
```

Cette division n'a pas de sens mathématique. Cependant, en Scilab, cette commande résout l'équation $x * w = v$. Ce sont les dimensions de v et w qui déterminent les dimensions de la réponse x. Le système peut être sous- ou sur-déterminé

```
[1,2,3]/[1,3]
[1,2]/[1,1]
```

L'équation à résoudre n'a pas de solution. Pourtant Scilab renvoie un resultat. Une idée de ce que représente la réponse ? Faire attention à l'utilisation de cette commande

```
w\v
E+1
sin([%pi,%pi/2;%pi/2,%pi])
abs(E), imag(E+%i)
[m,k] = max(E)
```

résout l'équation $w * x = v$

Une addition possible en Scilab

Les fonctions élémentaires s'appliquent aussi aux matrices

Qu'obtient-on dans les variables m et k ?

Le graphique

La représentation d'une fonction de $\mathbb{R}$ dans $\mathbb{R}$ commence par la création d'un vecteur d'abscisses, auquel on applique la fonction pour créer le vecteur des ordonnées.

<code>x=[0:0.1:2*pi]</code>	
<code>plot2d(sin(x))</code>	Trace les pts d'ordonnées <code>sin(x)</code> correspondant aux abscisses <code>(1:size(x,'c'))</code>
<code>plot2d(x,sin(x))</code>	Trace les pts d'ordonnées <code>sin(x)</code> correspondant aux abscisses <code>x</code>
<code>plot2d([0:0.5:2*pi],sin([0:0.5:2*pi]))</code>	La même courbe avec moins de points
<code>clf</code>	On efface le contenu dans la fenêtre de plot
<code>plot2d(x,[sin(x') sin(2*x') sin(3*x')])</code>	Si le 2ieme argument est une matrice, <code>plot2d</code> trace le graphe de chaque colonne

Les structures de contrôle et les opérateurs logiques

<code>for i=1:n,for j=1:n,a(i,j)=1/(i+j-1);end;end</code>	boucle <code>for</code> ; la boucle peut être interrompue par une instruction <code>break</code> "
<code>a=10, while a > 1, a=a/2; end; a</code>	le <code>while</code>
<code>clear a, i=1, j=2</code>	
<code>if i==j then a(i,j)=2; else a(i,j)=3; end,</code>	execution conditionnelle <code>if then else</code>
<code>%t, %f</code>	les variables booléennes
<code>1==1 & 0==1, 1 & 2,</code>	le <code>et</code> , les arguments peuvent être matriciels
<code>1 0</code>	le <code>ou</code> , les arguments peuvent être matriciels
<code>~[%t %t %f]</code>	la négation (avec un argument matriciel)

Les fonctions

En utilisant l'éditeur de Scilab, on écrit dans un fichier `nomDeFichier.sce` (script) une ou plusieurs fonctions implémentant les suites d'instructions réalisant une tâche - les algorithmes - dont on a besoin. La syntaxe est

```
function [x,y]=myfct(a,b)
// suite d instructions
x=a+b, y=a-b
endfunction
```

On sauvegarde le fichier (avec le menu ou en tapant `Ctrl-s`). Ensuite, le fichier peut être chargé dans la session Scilab

- 1) avec la commande `exec ~/nomDeFichier.sce` (on indique le chemin pour accéder au fichier, `~` désigne le répertoire de l'utilisateur).
- 2) ou en allant dans le menu `Execute` puis en faisant `Load into Scilab` (si ça marche)
- 3) ou en tapant directement `Ctrl-l` dans l'éditeur (si ça marche)

Pour executer la fonction, on tape dans la fenêtre de commande :

`myfct(1,2)` renvoie la valeur de x (le 1er élément de la liste des variables en sortie)
`[x,y] = myfct(1,2)` renvoie les valeurs de x et y

L'éditeur

L'éditeur permet de taper ses programmes dans un ou plusieurs fichiers que l'on peut charger ensuite dans la session Scilab. De nombreux raccourcis permettent de travailler plus rapidement :

`Ctrl-a` met le curseur en début de ligne
`Ctrl-e` met le curseur en fin de ligne
`Ctrl-p` met le curseur sur la ligne précédente
`Ctrl-k` supprime une ligne
`Ctrl-s` sauvegarde du fichier
`Ctrl-l` chargement du fichier dans la session Scilab

Les entrées sorties

Quand on fait du calcul matriciel, les données sont vite nombreuses et donc pour être efficace, on utilise des fichiers de données et de résultats.

– pour la lecture :

```
exec donnees.sce
```

 le fichier contient `a=[1,2,3;4,5,6]`

– pour l'écriture :

à l'écran :

```
print(%io(2), 'norme infinie = ', nor)
```

 écrit à l'écran `norme infinie = 10` si la valeur de la variable `nor` est 10

```
write(%io(2),s,'(e11.4)')
```

 écrit la valeur de la variable `s` à l'écran avec un format de 4 chiffres significatifs

dans un fichier :

```
a=eye(2,2),b=ones(a)
```

```
print('fichier',a,b)
```

 sauvegarde des variables `a` et `b` dans le fichier 'fichier' (dans le format Scilab lisible par l'utilisateur)

```
save('result',a,b)
```

 sauvegarde (en binaire) des variables `a` et `b` dans le fichier `result`