TD machine 1 - Mose 1003

E. Ringeisen

Octobre 2015

Découverte de Scilab

La console Scilab

Une fois l'ordinateur en route, cherchez le programme Scilab dans les menus accessibles depuis le coin de l'écran (il se trouve normalement dans le menu Cremi), et lancez le programme en cliquant. Un certain temps de chargement est nécessaire avant de voir apparaitre une fenêtre ressemblant à la figure 1.

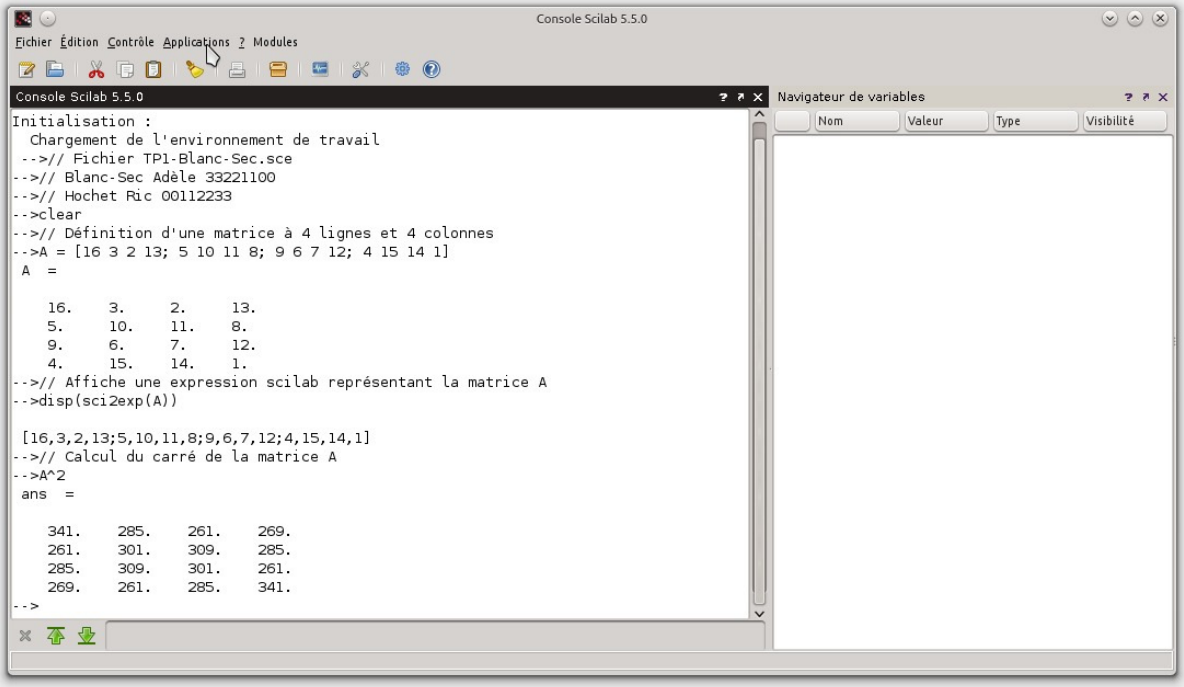

FIGURE 1 – Console Scilab

Dans la partie gauche de la console, on voit une flèche `-->` qu'on appelle une *invite de commandes*. A côté de cette flèche, on peut écrire des commandes dans un langage particulier qui est celui de Scilab, ces commandes sont interprétées et exécutées lorsqu'on appuie sur la touche entrée.
Pour commencer, créez un dossier dans lequel sera stocké votre travail, pour cela, écrivez sans erreur dans la console Scilab la commande

```
unix_w("cd && mkdir TPmose1003")
```

et appuyez sur la touche entrée. Si tout se passe bien, il y a maintenant un dossier `TPmose1003` dans votre dossier utilisateur. Vous pouvez obtenir le chemin de votre dossier utilisateur par la commande

```
unix_w("cd && pwd")
```

L'éditeur SciNotes

Il s'agit d'une fenêtre permettant d'écrire des suites de commandes et de les sauvegarder dans un fichier pour les exécuter et les modifier autant de fois qu'on veut.
Cette fenêtre s'ouvre en cliquant sur `Applications -> SciNotes` dans la console Scilab. Elle ressemble à celle de la figure 2.
Pour enregistrer votre premier fichier dans SciNotes, maintenez enfoncée la touche `CTRL` et appuyez sur la touche `S`. Une fenêtre de dialogue apparait pour enregistrer un fichier. Naviguez jusqu'à votre dossier `TPmose1003` et choisissez un nom de fichier qui se termine par `.sce` Dans la figure 2, on voit que l'étudiante Adèle Blanc-Sec a choisi d'appeler son fichier `TP1-Blanc-Sec.sce` Evitez de préférence les espaces dans des noms de fichiers, et choisissez un nom qui vous différencie (au cas où l'enseignant vous demande de rendre le fichier).
Chaque fois que vous utiliserez la séquence de touches `CTRL S`, le fichier sera enregistré. Pensez à le faire régulièrement pendant votre travail pour ne pas risquer de perdre des données.
Pour exécuter les commandes contenus dans le fichier, on utilisera la séquence `CTRL L`. Le résultat de l'exécution apparait alors dans la console scilab.

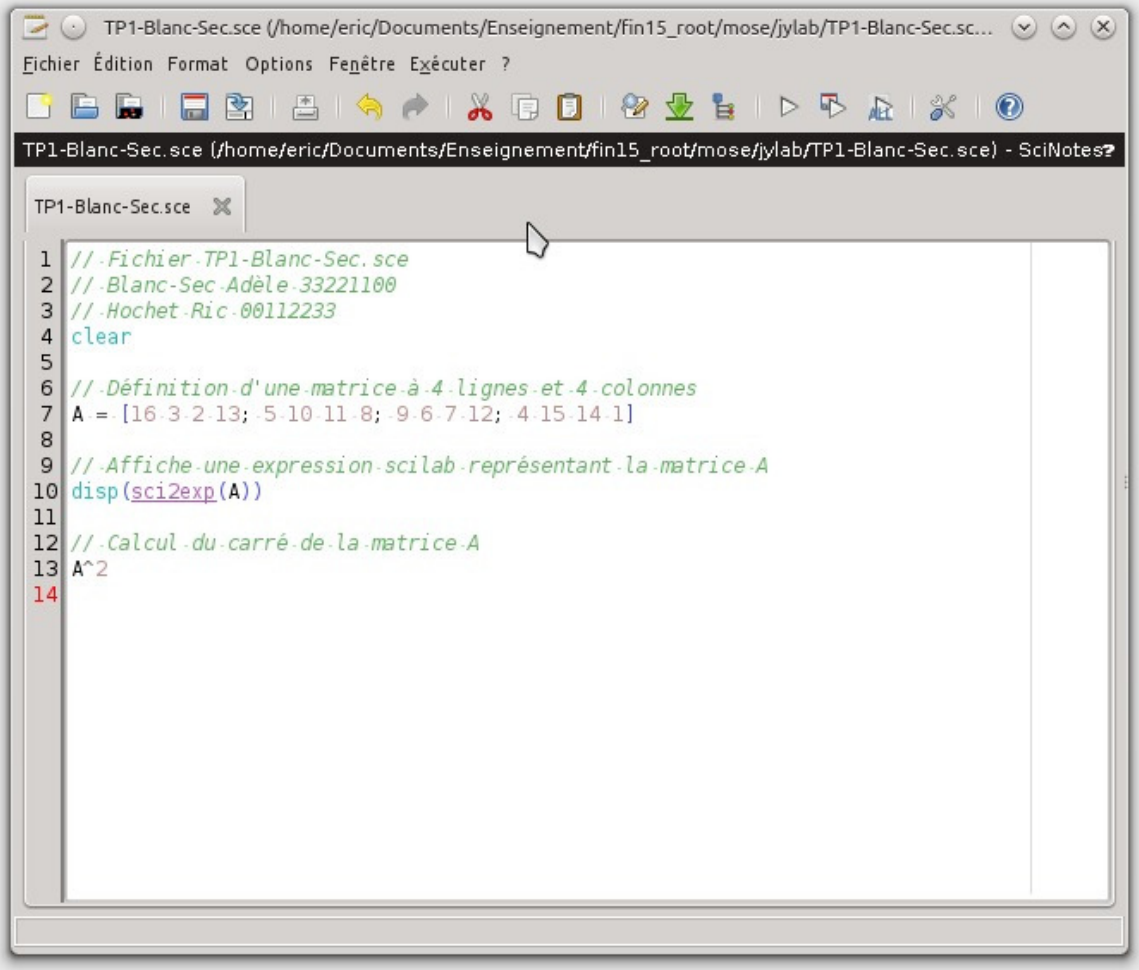

FIGURE 2 – Editeur SciNotes

Par exemple si on écrit dans SciNotes

```
clear
A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
```

et qu'on utilise la séquence `CTRL L`, la console Scilab affiche le résultat suivant

```
-->A = [16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1]
 A  =

    16.    3.     2.     13.
    5.     10.    11.    8.
    9.     6.     7.     12.
    4.     15.    14.    1.
-->
```

qui montre que vous venez de définir une matrice nommée `A`, qui possède 4 lignes et 4 colonnes. L'instruction `clear` en haut du fichier permet de garantir que chaque exécution commence dans un environnement *propre*, c'est à dire sans souvenir des exécutions précédentes.
Dans la figure 2, remarquez les lignes en haut du fichier d'Adèle Blanc-Sec :

```
// Blanc-Sec Adèle 33221100
// Hochet Ric 00112233
```

Ces lignes commencent par les deux caractères `//` : ce sont des lignes de *commentaire* qui ne sont pas exécutées par Scilab. Elles permettent d'écrire dans le fichier des informations destinées au lecteur et non pas à l'interpréteur de commandes. Ici Adèle a écrit son nom et son numéro d'étudiante ainsi que ceux de son binôme. Faites de même dans votre fichier.

Construction et manipulation des matrices

Dans votre fichier de commandes, entrez l'une après l'autre chacune des commandes suivantes et observez son effet dans la console Scilab. Pour chacune d'entre elles, *mettez un commentaire dans le fichier*, juste avant cette commande, qui explique en détail ce qu'elle fait.

```
A ^ 2
5 * A
A * 5
A'
B = 3 * A - A'
A(2, 3)
A(4, 1) + A(3, 2) + A(2, 3)
A(2,:)
A(:,2)
A([2 3], [3 4])
```

Scilab possède quelques fonctions pour construire des matrices particulières : continuez à commenter les commandes suivantes

```
eye(A)
eye(3, 3)
eye(2, 4)
Z = zeros(4, 2)
size(Z)
size(A)
length(Z)
length(A)
ones(2, 3)
15 * ones(3, 2)
diag([3 2 5])
B = 10:20
B = 10:-2:4
X = 0:3:10
x = linspace(-2, 2, 7)
x = linspace(-2, 2, 30)
x = linspace([-1;-2;-3], [0;0;0], 7)
```

En cas de doute, on peut utiliser `help` pour obtenir des informations sur une fonction, par exemple la commande

```
help linspace
```

affichera l'aide de la fonction `linspace` dans une fenêtre à part (il y a un certain délai avant que la fenêtre n'apparaisse la première fois). N'hésitez pas à ajouter vos propres commandes dans le fichier pour tester le comportement de Scilab
Exécutez et commentez maintenant l'une après l'autre les commandes

```
A(1:2, 3:4)
B = ones(3, 3) + diag([2 1 5])
C = [A(1:3,2) B(:,3) A(2:4,1)]
diag(C)
diag(diag(C))
[A(:,1:3) ; B]
```

On peut modifier les valeurs contenues dans une matrice. Commentez

```
B(3,2) = 100
truc = [B]
B(3,2) = -%pi
truc
B(:,2) = []
F = resize_matrix(B, 3, 4)
G = resize_matrix(F, 4, 3, "", -10)
B
```

Des opérations terme à terme

Commentez

```
A .* A
A .^ 3
[1 2; 3 4] ./ [4 3; 2 1]
```

Des opérations globales

Commentez

```
sum(C)
sum(C, 'r')
sum(C, 'c')
min(C)
max(C)
inv(C)
inv(C) * C
```

Des boucles de calcul

La construction `for...end` permet de faire un calcul itératif sur les colonnes d'une matrice. Commentez l'exécution de la boucle suivante

```
for m = A
    disp("La valeur de m est: ");
    disp(m);
end

disp("---------------------");
```

Pour itérer sur les lignes, on peut ruser en utilisant une matrice d'indices, telle que `1:4` (que vaut cette matrice ?)

```
for m = 1:4
    L = A(m,:);
    mprintf("La ligne numéro %d de A est %s\n", m, sci2exp(L));
end
```

On a utilisé ici la fonction `mprintf` qui permet de faire des sorties formattées dans la console Scilab. Essayez par exemple la commande suivante qui imprime π dans un champ de longueur 7 avec 5 décimales, et π^2 dans un champ de longueur 10 avec 3 décimales.

```
mprintf("La valeur de pi est %7.5f, son carré est %10.3f\n", %pi, %pi^2);
```

On peut utiliser deux boucles `for...end` imbriquées pour fixer tous les éléments d'une matrice. Quelle matrice le code suivant construit-elle ? Commentez.

```
M = zeros(3, 3)
for i = 1:3
    for j = 1:3
        M(i, j) = %pi / (i*j);
    end
end

disp("La matrice M est");
disp(M);
```

Que se passe-t-il si on retire le point-virgule en fin de ligne `M(i, j)=...` dans la boucle précédente ? Commentez l'effet d'un point-virgule en fin de ligne.

Appliquer des fonctions terme à terme

Commentez les instructions

```
sin(M)
cos(2*M)
```

Patchwork matriciel

Exécutez et commentez

```
UUU = repmat(['X'], 2, 2)
VVV = repmat(['*'], 2, 2)
WWW = [UUU, VVV; VVV, UUU]
repmat(WWW, 2, 3)
```

Question subsidiaire

Que fait la boucle suivante, qui contient la structure `if...else...end`

```
x = rand(1, 10) // matrice ligne aléatoire de longueur 10
y = zeros(x);

for i = 1:length(x)
    if i == 1
        y(i) = x(i);
    else
        y(i) = max(y(i-1), x(i));
    end
end
y
```