TD 1 en Scilab

E.Ringeisen

Novembre 2016

Découverte de Scilab

La console Scilab

Une fois l'ordinateur en route, le programme Scilab est accessible depuis le menu en haut de l'écran (dans le sous-menu Cremi). Un certain temps de chargement est nécessaire avant de voir apparaitre une fenêtre ressemblant à la figure 1.

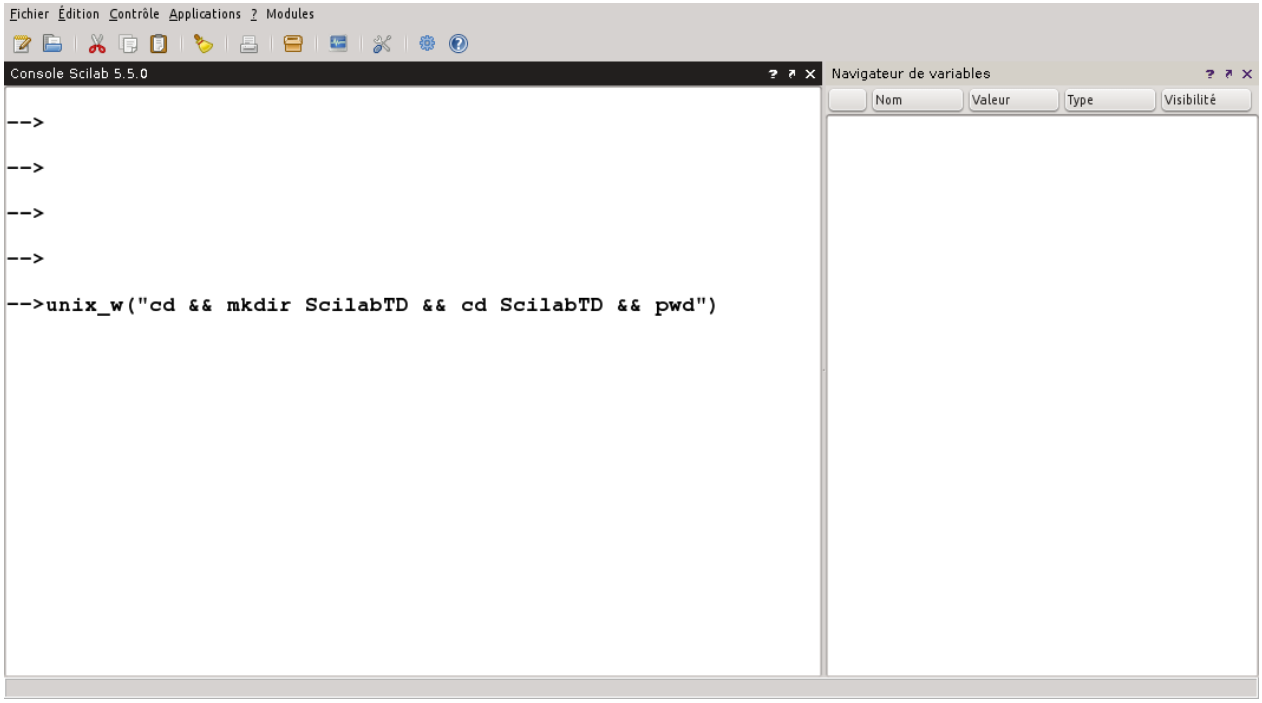

FIGURE 1 – Console Scilab

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.
Tapez soigneusement la commande indiquée dans la console et exécutez là. Elle crée un dossier nommé `ScilabTD` pour stocker votre travail et elle affiche le chemin de ce dossier.

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

`ScilabTD` et choisissez un nom de fichier qui se termine par `.sce`, par exemple `TD1-Nom-Prénom.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).

Sauvegarde : 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.

Exécution 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.

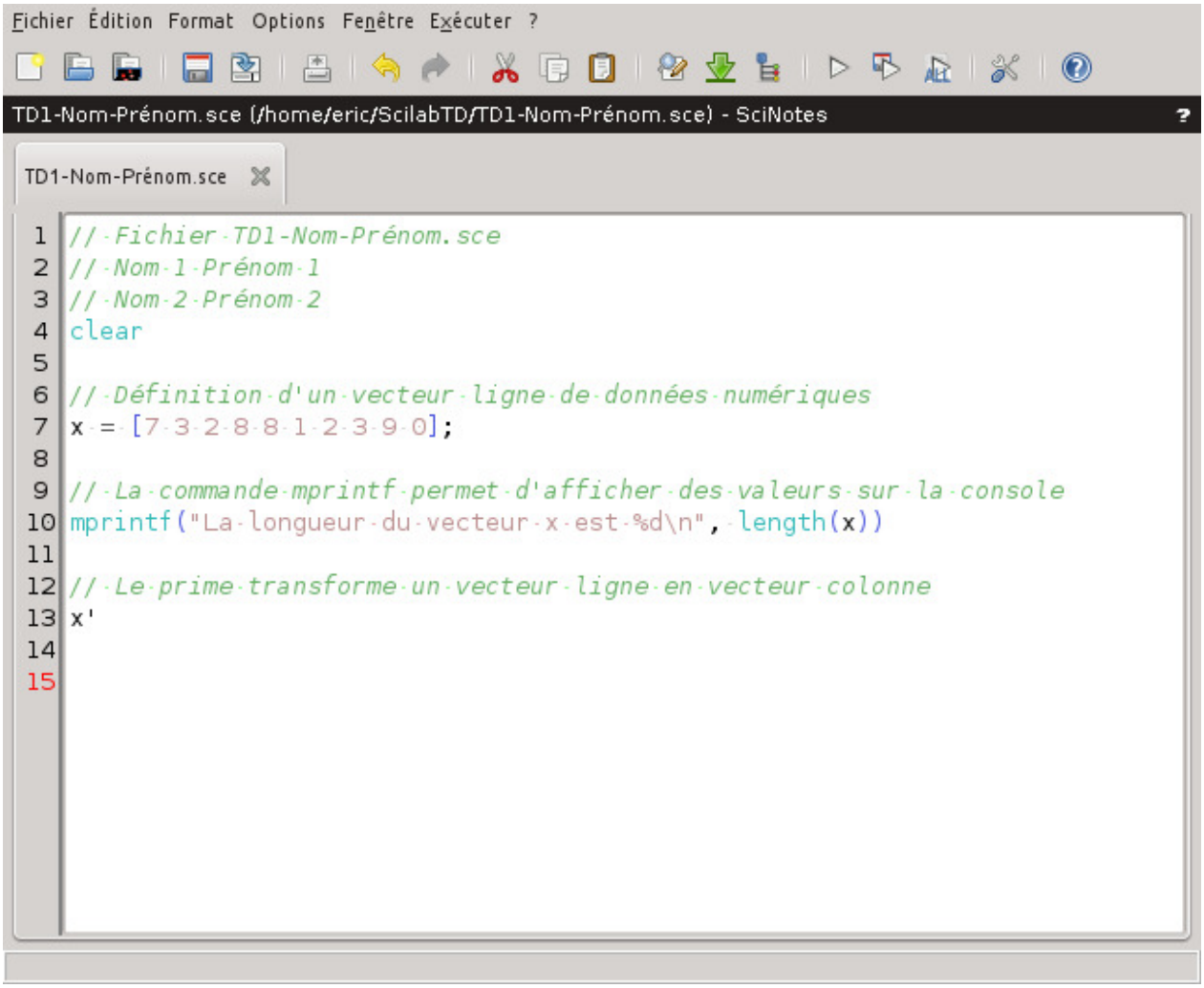

FIGURE 2 – Editeur SciNotes

Les lignes commençant par deux barres obliques `//` 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. En haut de votre fichier, mettez votre nom et celui de votre binôme.
Recopiez dans votre fichier et exécutez des instructions similaires à celles de la figure 2 pour voir comment définir un vecteur de données.

Vecteurs numériques

L'instruction `zeros(1, 10)` permet de créer un vecteur ligne de 8 valeurs nulles. Testez le, ainsi que `y = ones(1, 10)`. Commentez dans le programme. Pour créer un vecteur colonne, on peut utiliser par exemple `zeros(10, 1)`. Exécutez `zeros(3, 5)` : cette commande crée une famille de 3 vecteurs lignes de taille 5, ou 5 vecteurs colonnes de taille 3, ce qu'on appelle une matrice de valeurs.

Testez l'expression `rand(1, 3)`. Observez que le résultat change à chaque exécution de votre fichier. Quel est l'effet de la commande ? Remarquez la différence à l'affichage lorsqu'on tape `y = rand(1, 10)`. Scilab se débrouille pour découper la ligne de valeurs à afficher lorsqu'elle est trop grande.
Que fait la commande `y = linspace(0, 50, 10)` ? Faites varier les valeurs des paramètres pour comprendre. Commentez. Pour en savoir plus, tapez (sans la flèche)

```
-->help(linspace)
```

dans la console scilab et attendez que la fenêtre d'aide de scilab s'affiche, avec la documentation de la commande linspace (en anglais peut-être...)

L'opérateur : Testez et commentez l'instruction `z = 3:5:30` Faites varier les paramètres pour comprendre. Que se passe-t-il si on omet le dernier paramètre ?

Tranches de vecteurs Expliquer l'effet des expressions `x(1)`, `x(2)`, `x(6)`, `x(3:8)`, `x(1:2:10)`

Modifications des valeurs On peut changer une seule valeur dans le vecteur `x` par l'instruction `x(7) = -5`. Expliquez l'effet de `x(3:5) = [-1 -3 -2]`

Combinaison linéaire Après `y = 1:10`, calculer `z = 3*x - 5*y`. Expliquez d'où proviennent les données du vecteur `z`. Observez de même le résultat de `x-3`.

Produit, puissance Calculez de même `x .* y` (on prendra soin de mettre le point, et pas seulement `*`). Quel est l'effet produit ?. Calculez `x .^ 2` et `y .^ 3` Quel est l'effet produit ?

Opérations globales Calculez et expliquez `sum(x)`, puis `min(x)` et `max(x)` et `abs(x-3)`.

Fonctions usuelles Calculez `exp(x)`. Que contient le vecteur résultat ? Comparez avec les valeurs que donne votre calculatrice.
En scilab, le nombre π s'écrit `%pi`. Exécutez le code suivant

```
a = [0 %pi/6 %pi/4 %pi/3 %pi/2 2*%pi/3]
cos(a)
```

Remarquez que scilab ne donne pas 0 pour $\cos(\pi/2)$ mais `6.123D-17`, ce qui signifie 6.123×10^{-17}, c'est à dire un nombre très petit. Cette erreur d'arrondi vient du fait que le nombre π utilisé est arrondi à la précision de la machine alors que le nombre π mathématique contient une infinité de décimales.
Parmi les nombreuses fonctions usuelles de scilab, citons la racine carrée `sqrt`, le logarithme naturel `log` (c'est le ln des calculatrices) et le logarithme de base 10 noté `log10`. Il y a aussi `sin`, `cos`, `tan`, et leurs fonctions inverses `acos`, `asin`, `atan`. Il y aussi des versions `sind`, `cosd`, `tand` avec des angles en degrés, par exemple `cosd(60)` donne 0.5.

Définir soi-même une fonction

La définition d'une nouvelle fonction peut se décrire sur un exemple

```
function r = truc(t)
    r = (1 - t) .* sin(2 .* t);
endfunction
```

La première ligne est une ligne de déclaration, introduite par le mot clef *function*. Elle dit ici qu'on s'apprête à définir une fonction nommée `truc` qui prend un paramètre d'entrée `t` et qui a un paramètre de sortie (un résultat) `r`. Les lignes suivantes sont des instructions scilab, il peut y en avoir plusieurs, qui servent à calculer la valeur du résultat `r` à partir du paramètre `t`. Enfin, le mot clef *endfunction* termine obligatoirement la définition de la fonction.
La fonction définie ci-dessus correspond mathématiquement à

$$\text{truc}(t) = (1-t)\sin(2t)$$

En programmation, les noms de fonctions sont généralement des mots plutôt que des lettres comme f ou g, l'alphabet étant trop petit.
On peut maintenant utiliser la fonction `truc` comme on utilise `sin` ou `cos`. Par exemple exécutez les deux instructions

```
x = linspace(-5, 5, 100);
y = truc(x);
```

et expliquez ce que contiennent les vecteurs `x` et `y`.

Remarque Le rôle du point-virgule à la fin d'une ligne d'instruction de scilab comme ci-dessus est d'inhiber l'affichage du résultat dans la console à l'exécution. Lorsqu'il y a des grands vecteurs, comme c'est le cas ici, cela permet d'éviter de saturer l'affichage avec des valeurs numériques.

Tracé de fonctions

Pour tracer des fonctions, il faut utiliser une fenêtre graphique, qui, en scilab, s'appelle une figure. Ajoutez dans votre programme les commandes

```
xdel(winsid()); // détruit les fenêtres graphiques entre deux exécutions
fenetre = figure("Figure_name", "Mon premier tracé", "position", [100 50 1000 600]);
fenetre.background = color("aliceblue");
set("current_figure", fenetre);
```

La fonction `plot2d()` permet de tracer la courbe. Il faut lui passer en paramètre un vecteur colonne contenant les abscisses et un vecteur colonne contenant les ordonnées des points de la courbe à relier entre eux. On peut ajouter des paramètres permettant de styler un peu le résultat :

```
plot2d(x', y', style=[color("forestgreen")]);
```

Observez ce qui se passe si on change les valeurs du paramètre `"position"` de la fenêtre graphique (elles représentent des nombres de points sur l'écran).

Stylage du tracé

L'objet scilab contenant les courbes s'appelle un système de coordonnées (*axes* en jargon scilab). On peut obtenir le système de coordonnées courant comme ceci

```
coor = get("current_axes");
```

Il peut servir à **donner un titre au dessin**, avec une certaine taille des caractères par exemple

```
coor.title.text = "Quelques courbes";
coor.title.font_size = 3;
```

Pour **mettre une couleur de fond**, on utilisera

```
coor.filled = "on";
coor.background = color("seashell1");
```

On peut également mettre un **titre sur l'axe** des x et sur l'axe des y en modifiant les propriétés `x_label` et `y_label`

```
coor.x_label.text= "Axe des abscisses";
coor.x_label.font_size = 2;
coor.x_label.font_color = color("dark violet");

coor.y_label.text= "Axe des ordonnées";
coor.y_label.font_size = 2;
coor.y_label.font_color = color("dark violet");
```

On peut **positionner un axe** pour le faire passer par l'origine en modifiant la propriété `x_location` (dont les valeurs possibles sont `"bottom", "top", "middle"` et `"origin"`)

```
coor.x_location = "origin";
coor.x_label.position = [4, -0.5];
```

Enfin, on peut **ajouter une grille** par dessus le dessin si on veut

```
coor.grid = [-1, color("blue"), -1];
```

Les 3 valeurs sont les couleurs de la grille selon les 3 axes Ox, Oy et Oz, bien que l'axe Oz n'intervienne pas dans un dessin 2D. Une valeur de -1 indique l'absence de grille dans cette direction. Testez différentes valeurs pour voir l'effet produit.

Zoom

Un clic `Outils->Zoomer` de la fenêtre graphique déclenche un outil permettant en deux clics de sélectionner une zone rectangulaire du dessin sur laquelle on peut zoomer.
Un clic `Fichier->Exporter` de la fenêtre graphique permet de créer une image à partir du contenu de la fenêtre graphique. Taper par exemple le nom de fichier `mon-image.jpg`.

Complément 1

Ajouter au tracé la courbe représentative de la fonction

$$\text{plaf}\,(t) = \sin\,(3t) - t\cos\,(5t)$$

Améliorez le tracé en augmentant le nombre de points de discrétisation (c'est à dire le nombre d'abscisses contenues dans le vecteur `x`).

Complément 2

Calculer à la main la dérivée de la fonction $\text{truc}\,(t)$ et obtenir de scilab le tracé de cette dérivée sur le même dessin. On tracera la courbe en rouge.

Complément 3

Modifier le programme pour qu'il ouvre deux fenêtres graphiques et que le tracé du complément 1 se fasse sur la deuxième fenêtre entre les abscisse 0 et 6 seulement. On pourra choisir `fen2` comme nom de variable pour la deuxième fenêtre.