TP1 : Scilab – Prise en main et comment tracer le graphe d'une fonction

Henri GAUTHIER

3 octobre 2017

Sommaire

I Énoncé **2**

1 Prise en main du logiciel **2**
1.1 Qu'est-ce que Scilab ? . . . 2
1.2 L'environnement de travail . . . 2

2 Scilab comme une machine à calculer... **3**
2.1 La base ! . . . 3
2.2 Notion de variables . . . 3

3 ...avec des tableaux **4**
3.1 Créer un tableau . . . 4
3.1.1 À la main . . . 4
3.1.2 Formes prédéfinies . . . 5
3.1.3 Listes à coefficients regulièrement espacés . . . 5
3.1.4 Concaténation de tableaux de tailles compatibles . . . 6
3.2 Calcul avec des tableaux . . . 7

4 Graphe de la fonction sinus **7**

5 SciNotes **9**

6 Dichotomie **9**
6.1 Boucles «tant que » . . . 9
6.2 Branchement «si » . . . 10

II Suggestion de Correction **12**

Première partie
Énoncé

1 Prise en main du logiciel

1.1 Qu'est-ce que Scilab ?

Scilab (Scientific Laboratory) est un système interactif de calcul numérique utilisable comme une calculette et qui dispose d'un très grand nombre de fonctions, d'un langage de programmation et d'outils de visualisation graphique.

Pour lancer Scilab (sous Windows) cliquer sur l'icône sur le bureau ou chercher "Scilab" dans le menu démarrer.

Pour plus d'informations, en particulier pour l'installer sur votre ordinateur personnel, consultez la page `http://www.scilab.org`. Vous pouvez aussi consulter le cours en ligne suivant : `https://openclassrooms.com/courses/prise-en-main-de-scilab`.

1.1. Scilab est un logiciel open-source et gratuit. Son équivalent industriel que vous serez certainement amenés à utiliser en tant qu'ingénieur s'appelle Matlab. Les deux logiciels se ressemblent énormément. Vous aurez peu de peine à passer de l'un à l'autre.

1.2 L'environnement de travail

La première fois que vous ouvrez Scilab, cela devrait ressembler à ça :

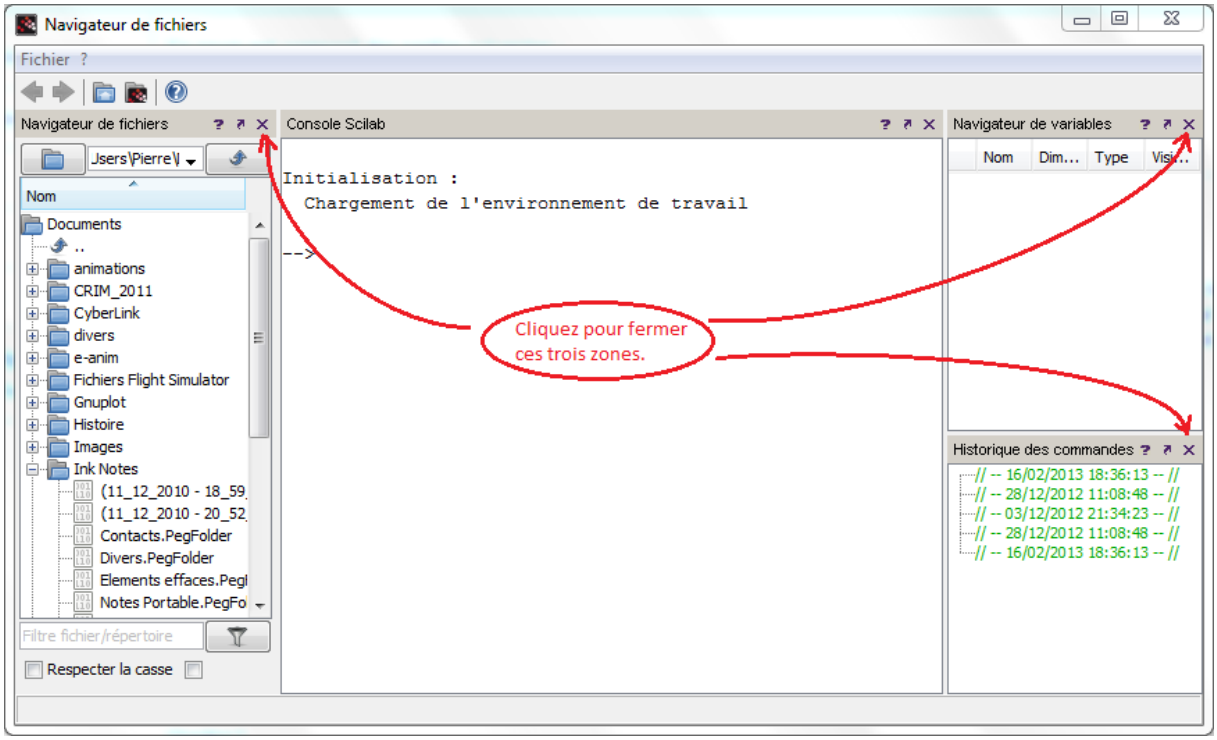

FIGURE 1 – Scilab au démarrage

On ferme les trois zones suivantes : `Navigateur de fichier`, `Navigateur de variables` et `Historique des commandes` qui ne vont pas du tout nous servir au courant de l'année.

Il ne nous reste donc plus que la `Console`.

2 Scilab comme une machine à calculer...

2.1 La base !

Scilab peut fonctionner comme une machine à calculer et fonctionne avec les règles de priorité que vous avez apprises. (Il fait les multiplications et les divisions avant les additions et les soustractions).

Addition `+`

Soustraction `-`

Multiplication `*`

Division `/`

Élévation à une puissance C'est le ^ qui est sur la touche où il y a le 9, on le choppe en maintenant la touche `Alt Gr` enfoncée et en appuyant sur la touche 9 (au dessus des lettres).

Racine carrée `sqrt(_)` : Remplacer _ par le nombre dont vous voulez la racine carrée.

Parenthèse `(bla bla bla)`

Tapez sur la touche `Entrée` pour obtenir le résultat.

Danger : les nombres à virgule s'écrivent avec des points.

Question 1 :

Intuitez les résulats des calculs suivants, puis vérifier à l'aide de la Console en les tapant.

```
1+2
1*37
1 - 1*37
(1-1)*37
sqrt(4)-1
sqrt(4-1)
```

Question 2 :

Toujours dans la console calculez donc ce truc infâme !

$$\frac{\sqrt{42} - 64,2 + 12^3}{\frac{37-5\times6}{\sqrt{12}}}$$

(On trouve 826.5746)

À la manière de `sqrt`, on peut utiliser les fonctions classiques suivantes :

Sinus, Cosinus, Tangente `sin(_)`, `cos(_)`, `tan(_)` Attention, angle en radians !

Arcsinus, Arccosinus, Arctangente `asin(_)`, `acos(_)`, `atan(_)`

Sinus hyperbolique, Cosinus hyperbolique, Tangente hyperbolique `sinh(_)`, `cosh(_)`, `tanh(_)`

Exponentielle, Logarithme néperien `exp(_)`, `log(_)`

Valeur absolue, Signe `abs(_)`, `sgn(_)`

Partie entière `floor(_)`

2.2 Notion de variables

Pour résoudre des problèmes de maths, vous avez appris à donner des noms aux inconnues. De même, en programmation, il est pratique de donner des noms à certains objets, surtout si ceux-ci varient au cours d'un programme. Pour cela, on utilise ce que l'on appelle des *variables*.

Un exemple : Julie a 12 bonbons. Elle en mange 2. On peut savoir combien il lui reste de bonbons grâce à une savante soustraction.

On écrira : `bonbons = 12`

`bonbons = bonbons - 2`

Le signe = fait ce qu'on appelle une *affectation*.

Maintenant, si à un moment vous voulez connaître la valeur de bonbons, il suffit de taper :

`bonbons` puis entrée.

Pour ne pas afficher les résultats des calculs intermédiaires, on utilise le caractère ; à la fin des phrases.

ça donne ça :

```
-->bonbons = 12;

-->bonbons = bonbons -2;

-->bonbons
```

Essayez !

Attention, les noms des variables peuvent contenir toutes les lettres **sans les accents**, majuscule et minuscule, les chiffres et les caractères suivants % _ # ? $!

Les noms des variables doivent commencer par une lettre ou un des caractères listés au dessus sauf le !

Seuls les 24 premiers caractères du nom d'une variable sont pris en compte, autrement dit cho6 et cho7 sont bien deux variables différentes mais anticonstitutionnellement1 et anticonstitutionnellement2 représentent la même variable pour Scilab.

Scilab est sensible à la casse, c'est à dire qu'il fait la différence entre les majuscules et les minuscules. Ainsi, les variables Veronique et veronique ne sont pas les mêmes.

Un petit conseil : avant de vous attaquer à un problème, faites un petit `clear` pour être sûr de partir de rien.

Certaines constantes sont toujours accessibles dans Scilab, comme par exemple les nombres π et e. Pour utiliser le nombre π, on tape %pi. Pour e, c'est %e.

Question 3 :

Donner une valeur approchée de $\pi + e$.

Vous aurez remarqué que, quand Scilab effectue un calcul, il répond un truc du genre : `ans=blablabla`. `ans` est aussi une *variable*. On peut l'appeler et l'utiliser comme un nombre. En fait, `ans` contient toujours le résultat du dernier calcul effectué.

Question 4 :

Intuitez la valeur de ans +3, puis vérifiez.

3 ...avec des tableaux

Scilab a été conçu pour calculer avec des tableaux.

Lorsque deux tableaux ont la même taille, on peut les additionner, les soustraire, les multiplier, les diviser, et même leur appliquer une fonction. Pour cela, on utilise les symboles et les fonctions que l'on a déjà appris.

C'est bien joli tout ça, mais comment crée-t-on un tableau ?

3.1 Créer un tableau

Alors évidement, il y a tout plein de méthodes, suivant ce qu'on veut faire.

Chaque tableau a un nombre de ligne(s) et un nombre de colonne(s).

3.1.1 À la main

Un bon exemple vaut mieux qu'un long discours.

Un tableau à une ligne et 3 colonnes : `u = [1 2 3]` ou bien `[1, 2, 3]`. Les tableaux à une ligne ont leur petit nom bien à eux : ce sont des *listes*.

Un tableau à 3 lignes et une colonne : `v = [4; 5; 6]`.

Un tableau à 2 lignes et 3 colonnes : `T = [1 2 3; 4 5 6]`.

On peut définir aussi un tableau en lui affectant chacune de ses valeurs. Les lignes et les colonnes sont numérotés à partir de 1. Par exemple, `T(2,3)` désigne le coefficient de la deuxième ligne et de la troisième colonne du tableau `T`.

On pourra donc définir un tableau `S` de la manière suivante :

```
S(1,1)=37
S(1,2)=12
S(1,3)=42
S(1,4)=%pi
S(2,1)=5.12
```

```
S(2,2)=sqrt(8)
S(2,3)=-1
S(2,4)=4.^5
```

Question 5 : Intuitez la taille de `S`. En utilisant la fonction `size`, vérifier ! Scilab donne d'abord le nombre de lignes puis le nombre de colonnes.

3.1. Si on omet de donner une valeur à certains coefficients d'un tableau, Scilab leur affecte la valeur `0`.

La fonction `length` permet de connaître le nombre total de cases dans un tableau.

L'instruction `[l,c]=size(S)` affecte à la variable `l` (respectivement `c`) le nombre de lignes (respectivement de colonnes) de `S`.

Si on ne devait retenir qu'une chose, c'est :

`T = [1 2 3 ; 4 5 6]` correspond au tableau ayant 2 lignes et 3 colonnes suivant : $\begin{matrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{matrix}$

3.1.2 Formes prédéfinies

Scilab dispose de fonctions définissant des tableaux comme par exemple :

`ones` un tableau avec que des `1`.

`zeros` un tableau avec que des `0`.

Pour savoir s'en servir je vous donne un truc de professionnel :

Question 6 : Dans la Console, tapez `help zeros`. Une fenêtre d'aide s'ouvre. Et tout est expliqué dedans ! Avec même des exemples.

Vous pouvez utiliser la fonction `help` avec toutes les commandes dont vous avez oublié le fonctionnement.

Personnellement, je trouve ça un peu relou d'avoir plusieurs fenêtres Scilab, et là aussi j'ai un truc de pro, pour mettre toutes les fenêtres en une : réduire la fenêtre si elle est en plein écran, puis cliquez sur le liseré noir ou bleu où il y a écrit `navigateur d'aide` et maintenez le clique de la souris enfoncé. Baladez la souris sur votre fenêtre principale de Scilab. Vous voyez des cadres apparaître. Si vous lâchez la souris à ce moment-là, votre fenêtre d'aide se mettra dans le cadre en question. Personnellement, je préfère lâcher la souris au milieu de l'écran. Je ne vois alors plus que l'aide, mais un liseré gris est apparu en bas de ma fenêtre : en cliquant sur `Console`, je retrouve mon environnement de calcul. À vous de choisir votre configuration préférée. Tout à droite du liseré noir ou bleu, il y a un point d'interrogation, une petite flèche et une croix. Le point d'interrogation, je ne sais pas à quoi il sert... La flèche sert à désolidariser la fenêtre. La croix sert à la fermer, évidemment.

Du coup, créez-moi un tableau `Uns` avec que des `1` de la même taille que `S`.

3.1.3 Listes à coefficients regulièrement espacés

Vous allez voir que celles-là, elles servent tout le temps !

On dénote par «`debut` » le premier nombre de la liste et par «`fin` » le dernier. On appelle «`pas` » l'écart entre deux nombres consécutifs de la liste que l'on veut créer. La syntaxe est alors : `liste=debut:pas:fin`.

Par exemple, `u= -10:4:2` correspond à `u=[-10 -6 -2 2]`.

3.2. `u= -10:4:%pi` correspond aussi à `u=[-10 -6 -2 2]` : si `fin` ne tombe pas tout pile, la liste s'arrête juste avant.

Si `pas` vaut 1, pas besoin de l'écrire : `v=1:1:37` est exactement la même liste que `v=1:37`.

Bien sûr, le `pas` peut être négatif si ça vous amuse.

Question 7 : D'après vous, que donne `w=2:5` ? Vérifiez.

D'après vous, que donne `x=5:2` ? Vérifiez.

Question 8 : Créez une liste `l` qui va de -12 à 51 avec 7 comme pas. Quelle est sa taille ?

Si on ne devait retenir qu'une chose, c'est :

$$\texttt{a = 1 : 3 : 22} \text{ correspond à } 1.\ 4.\ 7.\ 10.\ 13.\ 16.\ 19.\ 22.$$

3.1.4 Concaténation de tableaux de tailles compatibles

Je vous donne un exemple :

```
-->M=[ones(2,3) zeros(2,2) ; 1:3 [1 -1]]
 M  =

    1.    1.    1.    0.    0.
    1.    1.    1.    0.    0.
    1.    2.    3.    1.  - 1.
```

Question 9 : À vous de jouez : faites-moi ça :

$$\begin{pmatrix} -37 & -30 & -23 & -16 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$$

On avait vu que pour accéder au coefficient de la `i`ème ligne et `j`ème ligne du tableau `T`, on devait taper `T(i,j)`. Plus généralement, on peut remplacer `i` et `j` par des listes. Si `I` et `J` désignent des listes d'entiers strictement positifs, `T(I,J)` désigne le tableau extrait de `T` contenant les lignes indicées par `I` dans l'ordre où `I` les indice, et de même les colonnes indicées par `J`.

Du coup, je vous donne une autre méthode pour répondre à la question 9, mais en plusieurs lignes de code via l'affectation.

```
Q9=zeros(3,7);
Q9(1,1:4)=-37:7:-16;
Q9(1,5:$)=1:3;
Q9(2:$,1:4)=ones(2,4);
Q9
```

`$` signifie la dernière ligne ou colonne suivant sa position.

Vous avez compris comment ça marche ? En premier on met les lignes concernées, en deuxième les colonnes concernées.

3.3. Pour considérer toutes les lignes, ou toutes les colonnes d'un tableau, on peut utiliser `:` au lieu de `1:$`.

Question 10 : Refaites le tableau `M` avec cette méthode.

Si la variable `u` est une liste déjà définie, on peut même faire des trucs du genre : `u = [3 u]`. Cela revient à rajouter 3 au début de `u`.

Mettons qu'on ait déjà un tableau existant. Par exemple, `M`. On veut construire `N` de la façon suivante :

— `N` est un tableau de taille 2×2,

— La première colonne de `N` est constituée des cases 1 et 3 de la colonne 3 de `M`,

— Sa deuxième colonne est constituée des cases 1 et 3 de la dernière colonne de `M`.

Alors, on fait comme ça :

```
N=M([1 3],[3 $])
```

3.2 Calcul avec des tableaux

Soient `A` et `B` deux tableaux de même taille. Alors,
— `A+B` désigne leur somme terme à terme.
— `A-B` désigne leur différence terme à terme.
— `A.*B` désigne leur produit terme à terme. Attention à ne pas oublier le point avant *, sinon, il s'agira du produit entre matrices.
— etc... `A./B`, `A.^ B`, `sin(A)`, ...

On peut aussi multiplier un tableau par un nombre : `37.*T`. Dans ce cas, tous les coefficients de `T` sont multipliés par `37`.

Question 11 : L'autre jour Alice m'a téléphoné parce qu'elle avait un problème de maths (true story !). Elle est à la fac, et elle avait ses notes de partiels et ses notes totales. Elle sait que ses notes totales sont calculées de la manière suivante : la note de partiel compte pour 40% du total, et la note d'exam compte pour 60% du total. Sa grand-mère lui demandait ses notes d'exam. Et, là, la pauvre Alice ne savait pas répondre. Donc, elle a choisi l'option *appel à un ami* et c'est tombé sur moi. Bon, là du coup c'est vous ! Je vous donne ses notes de partiels et ses notes totales, et à vous de jouer. Un conseil : au lieu de faire dix fois le même calcul, calculez directement avec les listes.

```
partiels=[12 14 9 15 10 7 16 18 4];
totaux=[10.2 12.2 13.2 14.4 13.6 10.6 9.4 16.8 8.2];
```

4 Graphe de la fonction sinus

En fait, on ne peut pas tracer de courbes avec Scilab ! Mais on peut tracer des segments. Mis à la suite, s'ils sont suffisamment petits, cela donne effectivement une courbe.

Question 12 : Tapez la commande `plot2d([0 1],[2 3])`. Quel segment a-t-on tracé ?

On a donc la syntaxe suivante : `plot2d(liste des abscisses,liste des ordonnées)`.

Pour effacer la figure, on utilise la commande `clf`. Nettoyez votre figure. On va maintenant tracer une ligne brisée de deux segments :

Question 13 : Tracer sur une feuille le résultat que vous attendez pour la commande `plot2d([0 1 0],[0 2 1])`. Puis, vérifiez.

Ainsi, Scilab trace des lignes brisées. Par conséquent, pour tracer une fonction, on se donne une liste d'abscisses `X`, puis la liste d'ordonnées associées `Y`.

On construit donc `X` comme une liste régulièrement espacée. On va tracer la fonction sinus entre 0 et 2π.

Question 14 : On choisit un pas `h=1`. Créez la liste `X` d'abscisses correspondante. Créez la liste `Y` associée. Puis, tracez le graphe correspondant.

Il est clair que le résultat obtenu n'est pas satisfaisant. On va devoir changer le pas.

Je vous donne un petit truc pour perdre moins de temps : pour retaper une commande que vous avez déjà tapée, utilisez les flèches directionnelles.

Question 15 : Refaites la question 14 avec `h=0.01`.

Vous remarquerez que Scilab n'a pas ouvert de nouvelle figure, mais qu'il a fait son tracé sur la même figure, sans effacer le tracé précédent. Si on avait voulu l'effacer, on aurait utilisé `clf`.

Vous me direz, on pourrait avoir besoin de faire plusieurs figures. Pour créer une nouvelle figure, on utilise la commande `figure(_)` où _ représente un entier naturel. Bon, si vous avez la flemme de taper des longs trucs, `scf` fait exactement là même chose. Dans ce cas, `clf` nettoie la dernière figure. Si vous voulez en nettoyer une autre, utilisez `clf(_)` où _ représente le numéro de la figure à nettoyer. Pour basculer d'une figure existante à l'autre, on utilise aussi `figure` (ou `scf`). Pour fermer une figure, on utilise `close(figure(_))`, où _ désigne le numéro de la figure à fermer.

Maintenant, on va apprendre à faire des trucs jolis.

Par exemple, j'aimerais beaucoup me placer dans un repère orthonormé. Pour cela, il faut rajouter un *argument* dans la fonction `plot2d` :

```
plot2d(X,Y,frameflag=4)
```

Alors, `frame` en anglais ça veut dire cadre. `flag` ça veut dire drapeau. Et donc on choisit le drapeau numéro 4 pour le cadre, on a des axes orthonormés. Bon, je suis d'accord c'est impossible de se rappeler que c'est 4 qu'il faut. Du coup, n'oubliez pas la fonction `help`. Isometric scale = repère orthonormé ! De plus, j'aimerais bien donner des noms aux axes. Par exemple, comme ça :

```
xlabel("x");
ylabel("y");
```

On aime bien aussi donner un titre à ses graphiques :

```
title("le titre de votre choix sans accent sans apostrophe and so on")
```

On peut aussi choisir la couleur de la courbe. Pour cela, dans `plot2d`, on rajoute un argument :

```
plot2d(X,Y,style=color("le_nom_de_la_couleur"));
```

Vous trouverez la liste des couleurs disponibles via `help color_list`.

Question 16 : (facultative) Fermez la figure 0, puis dans la figure 37, tracez le graphe de la fonction sinus entre -2π et 2π dans un repère orthonormé. De plus, je voudrais que l'axe des abscisses s'appelle *carotte* et que l'axe des ordonnées s'appelle *poireau*. Ok, ça n'a pas vraiment de sens mais c'est rigolo ! Et en plus, j'adorerais que la courbe soit violette. Comme titre, je souhaite lui donner *Le plus joli titre de tous les temps*.

Si l'on veut tracer plusieurs figures sur le même graphique, la syntaxe est la suivante :

```
plot2d(X,[Y1 Y2 Y3],style=[couleur1 couleur2 couleur3])

MAIS ATTENTION: Contrairement au cas d'une seule courbe, où X peut être au choix une liste
ou un tableau à une colonne du moment que Y a la même taille que X, ici, X doit être un
tableau à une colonne et les Yi doivent être des tableaux de la même taille que X.
Pour transformer un tableau de taille n*m en son symétrique par rapport à la diagonale qui
est de taille n*m (transposer), on utilise le symbole .' :
A.' transforme la ligne A en une colonne.
```

Question 17 : Sur une même figure, tracer les fonctions $x \mapsto \sqrt{x}$, $x \mapsto x$ et $x \mapsto x^2$ entre 0 et $1,5$.

Indication : On commence par taper :
x = 0 : 00.1 : 1.5 ça, vous avez compris pourquoi.
Puis on transforme cette liste en un tableau à une colonne en tapant :
x = x.'

Après avoir représenté une fonction, on peut limiter les valeurs de x et de y en écrivant :
zoom_rect([a,b,c,d])
la courbe se limitera aux x compris entre a et c et aux y compris entre b et d.

Question 18 : Poser `X=[0:0.01:10]`. Utiliser ce `X` pour tracer le graphe de la fonction $x \mapsto \tan(2x)$.

Elle a une sale tête, n'est-ce pas ?
C'est pourquoi on va limiter les X et Y.
Débrouillez vous pour que X varie entre 3 et 7, et y entre -10 et 10.

C'est-y pas plus bioutifull comme ça ?

Si on ne devait retenir qu'une chose, c'est :

On définit la liste x ; on définit la liste y = f(x) .
On tape : plot2d(x,y)

5 SciNotes

Dans ce paragraphe, vous allez apprendre à enregistrer ce que l'on appelle un *script*. Il s'agit d'un document texte qui contient vos lignes de code.

Pour créer un nouveau script, cliquez sur l'icône `Démarrer SciNotes` ou dans le menu `Applications` cliquez sur `SciNotes`. Vous pouvez placer la nouvelle fenêtre où vous voulez via la méthode du glisser-déposer que l'on a déjà vue.

Au début de chacun de vos scripts, tapez toujours les trois lignes suivantes :

```
mode(2)
clear
xdel(winsid())
```

La première sert à s'assurer que Scilab affiche bien les résultats de vos calculs.

La deuxième, vous savez déjà à quoi elle sert.

La troisième sert à fermer toutes les figures ouvertes.

Les scripts ont plusieurs intérêts. Déjà, vous allez pouvoir enregistrer votre travail en cliquant sur la disquette.

En plus, vous pouvez écrire pleins de conneries, ou plutôt des *commentaires* (grâce au symbole `//`).

Par exemple, dans les quatre lignes suivantes, ce qui est après `//` n'est que du commentaire.

```
mode(2) // sert à faire en sorte que la Console affiche les résultats
clear // sert à effacer toutes les variables
xdel(winsid()) // sert à fermer toutes les figures
1+1
```

Quand Scilab exécute ce script, il ignore tout ce qu'il y a derrière le symbole `//` sur chaque ligne.

Pour (enregistrer puis) exécuter un script, il suffit de cliquer sur le bouton avec un symbole "play" et une disquette.

Question 19 : Recopiez le script précédent. Enregistrez-le. Intuitez ce que Scilab va renvoyer à l'exécution. Puis, exécutez-le pour vérifier.

Désormais, vous travaillerez de préférence dans des scripts.

6 Dichotomie

Nous allons mettre en place ici la méthode de la dichotomie pour trouver le zéro d'une fonction monotone.

Avant cela, je vais devoir vous introduire deux notions : celles de boucles «tant que » (while) et de branchement «si » (if).

6.1 Boucles «tant que »

On a envie de répéter une instruction plusieurs fois jusqu'à ce qu'une certaine condition ne soit plus vérifiée. Par exemple, on pose `n=0` et on veut exécuter `n=n+1` tant que `n<37`.

Les boucles «tant que » permettent d'effectuer des *itérations* tant qu'une certaine condition est vérifiée. On ne connaît pas le nombre d'itérations à effectuer, mais à chaque itération, on vérifie si la condition est vraie ou fausse. Dès que cette condition est fausse, on sort de la boucle.

C'est donc cet outil que l'on va utiliser.

La syntaxe est la suivante :

```
while bla_bla
  faire ceci cela;
end
```

`bla_bla` désigne une condition. Elle est soit vrai soit fausse.

Dans le cas présent cela donnerait :

```
n=0;
while n<37
  n=n+1;
end
```

Question 20 : Dans un script, recopiez ces instructions. Intuitez la valeur de `n` après exécution. Vérifiez.

Dans la condition, on peut utiliser >,<,== (Attention, il faut faire la différence entre == (deux signes = qui se suivent) qui est utilisé dans une condition, et = qui est utilisé pour une affectation.),<= (≤), >= (≥) ou bien <> (≠).

Question 21 : Utilisez une boucle «tant que » pour calculer 6! (factoriel). Pour cela, utiliser deux variables : `n` et `fact6`. Autrement dit, traduisez en Scilab l'algorithme suivant :

```
fact6=1
n=1
tant que n<=6 faire
  fact6=fact6*n
  n=n+1
fin faire
Renvoyer fact6
```

6.1. Pour ce genre de question, il existe une autre forme de boucle plus adaptée : la boucle «pour » (for).

On ferait alors ça :

```
fact6=1;
for n=[1:6]
  fact6=fact6.*n;
end
fact6 // renvoie la valeur obtenue
```

En français ça donne :

```
fact6=1
Pour n allant de 1 à 6 faire
  fact6=fact6*n
fin faire
Renvoyer fact6
```

Question 22 : Toujours avec une boucle «tant que », calculez 2^{11}.

6.2 Branchement «si »

Le branchement «si » sert à n'exécuter une instruction que si une certaine condition est vérifiée.

En Scilab, cela donne :

```
if condition
  instruction
end
```

En français (algorithmique) :

```
Si condition faire
  instruction
fin faire
```

La condition et l'instruction prennent la même forme que dans une boucle «tant que ». La seule différence c'est que l'instruction sera exécutée une ou zéro fois suivant que la condition est vérifiée ou ne l'est pas.

On peut aussi utiliser un branchement «sinon ».

```
Si condition faire
  instruction
Sinon
  instruction
fin faire
```

Soit en Scilab :

```
if condition
  instruction1
else
  instruction2
end
```

On peut aussi utiliser des schémas de la forme :

```
if condition1
  instruction1
elseif condition2
  instruction2
elseif condition3
  intruction3
(else
  instruction4)
end
```

Dans ce cas, si la `condition1` est vérifiée (que la `condition2` et la `condition3` le soit ou pas) l'`instruction1` sera exécutée. Si la `condition1` n'est pas vérifiée et si la `condition2` est vérifiée, c'est l'`instruction2` qui sera exécutée. Si la `condition1` et la `condition2` ne sont pas vérifiées mais la `condition3` l'est, alors c'est l'`instruction3` qui sera exécutée. Et, pour finir, si aucune des conditions n'est vérifiée, l'`instruction4` sera exécutée.

Question 23 : Soit $f : \left| \begin{array}{ccc} [1\,;5] & \to & \mathbb{R} \\ x & \mapsto & x^2 - x - 1 \end{array} \right.$. Tracer la fonction f dans une figure. Observer que f s'annule. Sur une feuille, dérivez f pour vérifier qu'elle est croissante et donc ne s'annule qu'une seule fois. Le x pour lequel elle s'annule s'appelle le *nombre d'or*. Toujours sur une feuille, trouver la valeur exacte du nombre d'or.

Nous allons utiliser la méthode de la dichotomie pour donner une valeur approchée du nombre d'or à 10^{-5} près.

Question 24 : À la variable `epsilon`, affectez la valeur 10^{-5}. Poser `a=1` et `b=5`
L'algorithme de la dichotomie se fait de la manière suivante :

```
Tant que (b-a>epsilon) faire
  c=(a+b)/2
  Si f(c)<=0 faire
    a=c
  Sinon faire
    b=c
  fin faire
fin faire
Renvoyer c
```

À vous de jouer !

Question 25 : Soit $f : \left| \begin{array}{ccc} [0\,;5] & \to & \mathbb{R} \\ x & \mapsto & x^3 - 7x^2 - 10x + 16 \end{array} \right.$. Montrer que f est décroissante et s'annule une seule fois. Tracez-la. Adaptez le script précédent pour trouver une valeur approchée de son zéro. Puis, sur une feuille, trouvez la valeur exacte de ce zéro.

Deuxième partie

Suggestion de Correction

Question 1 :

```
-->1+2
 ans  =

    3.

-->1.*37
 ans  =

    37.

-->1-1.*37
 ans  =

  - 36.

-->(1-1).*37
 ans  =

    0.

-->sqrt(4)-1
 ans  =

    1.

-->sqrt(4-1)
 ans  =

    1.7320508
```

Question 2 :

```
-->(sqrt(42)-64.2+12.^3)./((37-5.*6)./sqrt(12))
 ans  =

    826.5746
```

Question 3 :

```
-->%pi+%e
 ans  =

    5.8598745
```

Question 4 :

```
-->ans + 3
 ans  =

    8.8598745
```

Question 5 :

```
-->size(S)
 ans  =

    2.    4.
```

Question 6 :

```
-->help zeros

-->Uns=ones(S)
 Uns  =

    1.    1.    1.    1.
    1.    1.    1.    1.
```

Question 7 :

```
-->w=2:5
 w  =

    2.    3.    4.    5.

-->x=5:2
 x  =

     []
```

C'est la liste vide !!!!!!!!!!!!

Question 8 :

```
-->l=-12:7:51
 l  =

  - 12.  - 5.    2.    9.    16.    23.    30.    37.    44.    51.

-->length(l)
 ans  =

    10.
```

Question 9 :

```
-->[-37:7:-16 1:3 ; ones(2,4) zeros(2,3)]
 ans  =

  - 37.  - 30.  - 23.  - 16.    1.    2.    3.
    1.     1.     1.     1.     0.    0.    0.
    1.     1.     1.     1.     0.    0.    0.
```

Question 10 :

```
-->M=zeros(3,5);

-->M(1:2,1:3)=ones(2,3);

-->M($,1:3)=1:3;
```

```
-->M($,4:$)=[1 -1];

-->M
 M  =

    1.    1.    1.    0.    0.
    1.    1.    1.    0.    0.
    1.    2.    3.    1.  - 1.
```

Question 11 :

```
-->exams=(totaux - 0.4.*partiels)./0.6
 exams  =

    9.    11.    16.    14.    16.    13.    5.    16.    11.
```

Question 12 : Réponse figure 2.

Question 13 : Réponse figure 3.

Question 14 :

```
-->clf

-->h=1;

-->X=0:h:(2.*%pi);

-->Y=sin(X);

-->plot(X,Y);
```

Résultat figure 4.

Question 15 :

```
-->h=0.01;

-->X=0:h:(2.*%pi);

-->Y=sin(X);

-->plot(X,Y);
```

Résultat figure 5

Question 16 :

```
-->close(figure(0))

-->figure(37);

-->X=-2.*%pi:0.01:2.*%pi;

-->plot2d(X,sin(X),frameflag=4,style=color("purple"));
```

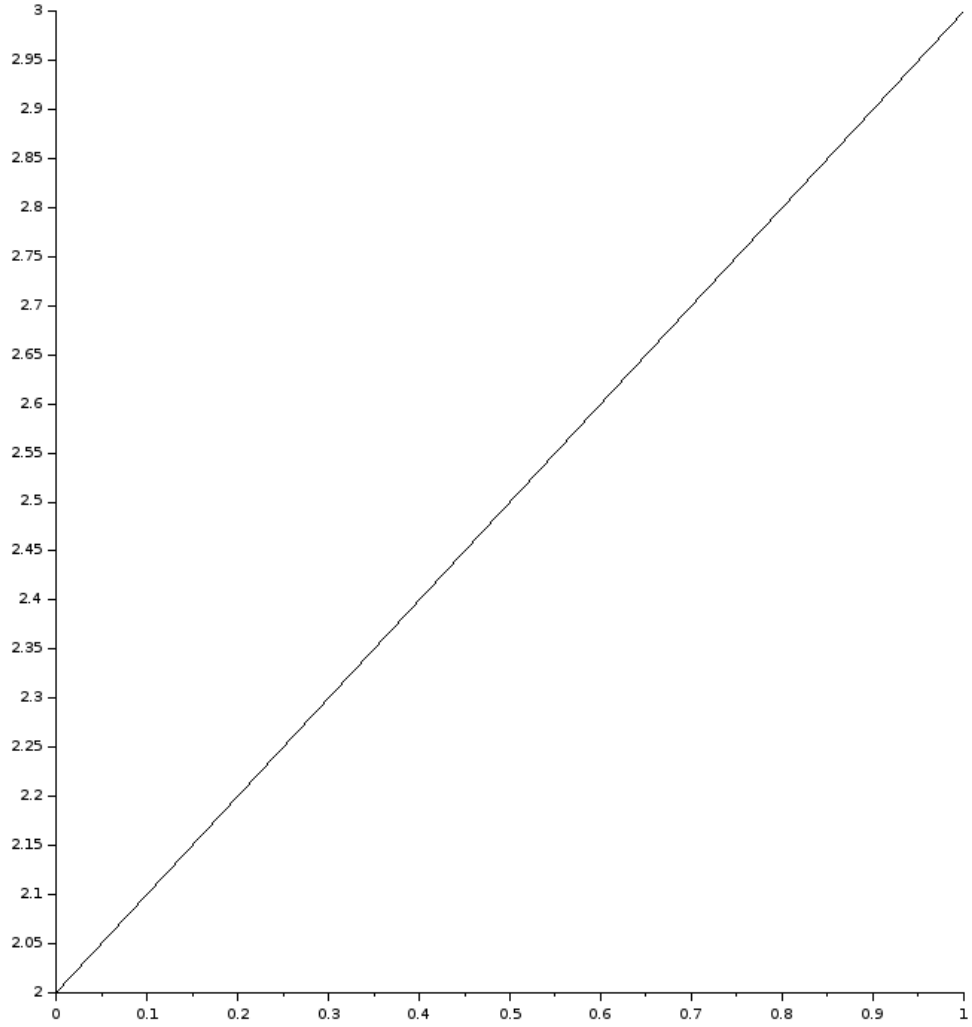

FIGURE 2 – Segment [(0,2),(1,3)]

```
-->xlabel("carotte")

-->ylabel("poireau");

-->title("le plus joli titre de tous les temps")
```

Question 17 : –>x=[0 :0.01 :1.5] ;
y=sqrt(x) ;
z=x.^2 ;
plot2d(x,y)
plot2d(x,x)
plot2d(x,z)

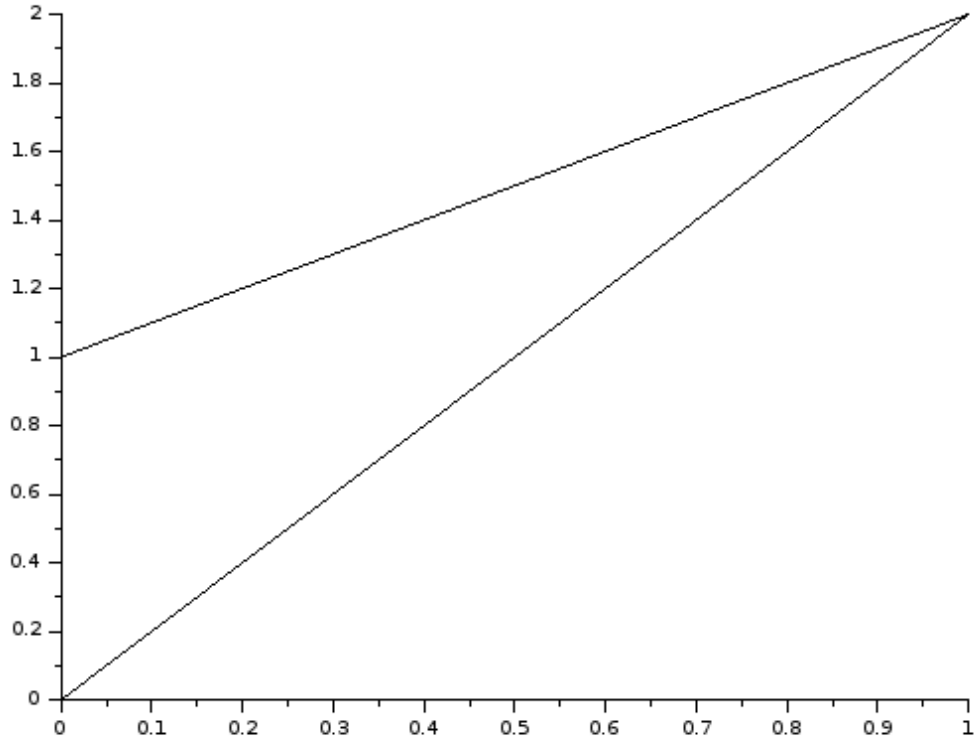

FIGURE 3 – Ligne brisée

Question 18 : –>x=[0 :0.01 :10] ;
–>y=tan(2*x) ;
–>plot2d(x,y)
–>zoom_rect([3,-10,7,10])

Question 19 : trop facile !

Question 20 : On trouve n = 37

Question 21 : mode(2)

```
clear
xdel(winsid())
fact6=1 ;
n=1 ;
while n<6
n=n+1 ;
fact6=fact6*n ;
end
fact6
```

Question 22 :

```
deuxpuiss=1 ;
n=1 ;
while n<9
deuxpuiss=deuxpuiss*2 ;
```

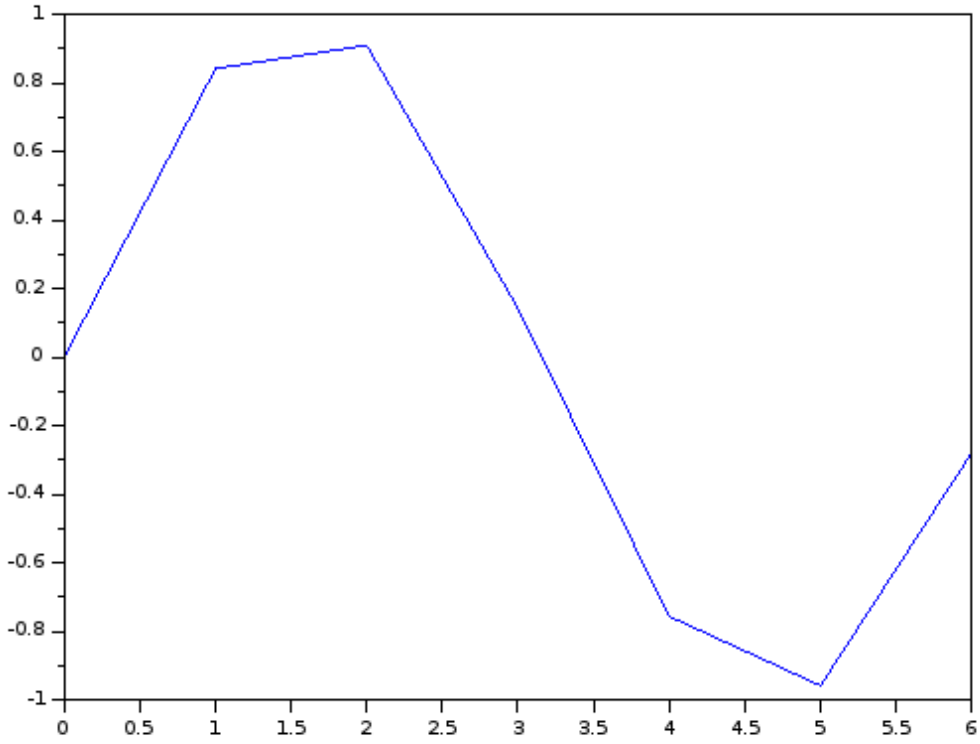

FIGURE 4 – Houla ! Le pas est beaucoup trop grand !

```
n=n+1 ;
end
deuxpuiss
```

Question 23 : x=[1 :0.01 :5] ;

```
y=x.^2-x-1 ;
plot2d(x,y)
```

Question 24 : mode(2)

```
clear
xdel(winsid())

eps=10.^(-5) ;
a=1 ;
b=5 ;
while b-a>eps
c=(a+b)/2 ;
if c.^2-c-1<=0
a=c ;
else b=c ;
end
end
c
```

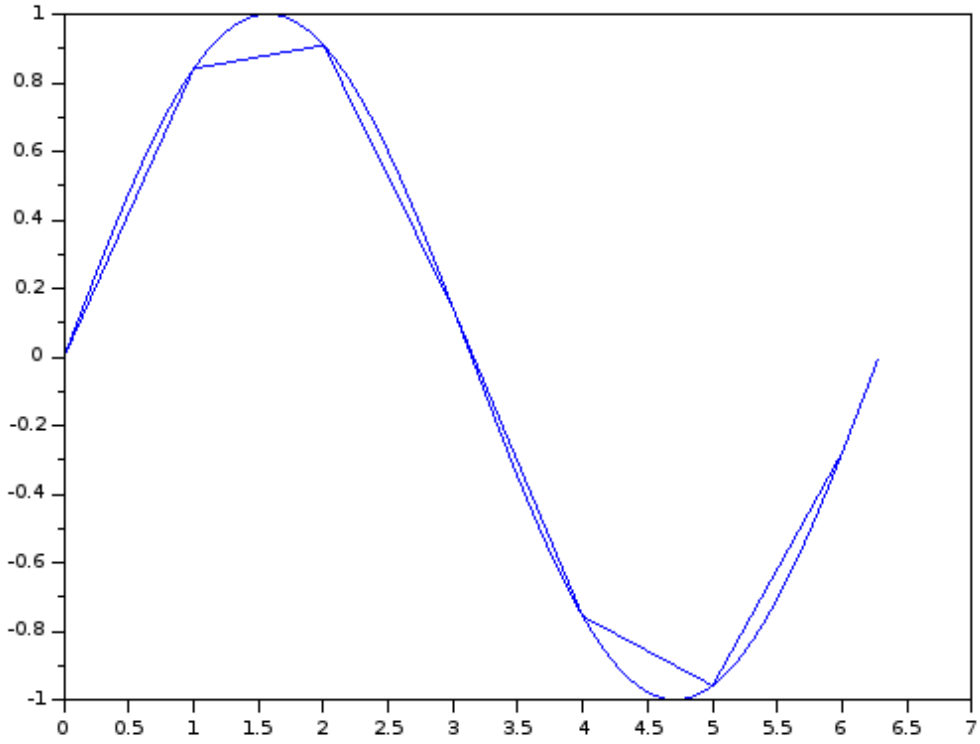

FIGURE 5 – Ah ! Là c'est mieux !

Question 25 : eps=10.^(-5) ;
a=0 ;
b=5 ;
while b-a>eps
c=(a+b)/2 ;
if c.^3-7*c.^2-10*c+16<=0
b=c ;
else a=c ;
end
end
c