Introduction à l'analyse numérique
Année : 2017-2018
Formation : L2 Mathématiques

TP1 : Initiation au logiciel Scilab

1. Dans Scilab les variables sont définies très naturellement. Essayez par exemple dans la fenêtre de commande ("Console Scilab") les commandes suivantes :

```
alpha = 1.54
b = 1.32;
alpha
alpha;
b
```

2. Pour créer des tableaux, vous pouvez utiliser les commandes suivantes :

```
A = [1 2 3]  ,  B = [2,3,4]
C = [1;3;5]  ,  D = [1:4]
E = [1:10]  ,  F = [1:2:10]
G = [1:3:10]  ,  H = [1:4:10]
 I = [1:4]  ,  J = [1,2,3;4,5,6;7,8,9]
 K = [3 4 ; alpha b]
 G = K', H=C'
 size(C), D(4), J(2,2)
```

   La commande `E(1,3)` retourne le coefficient de la ligne 1, colonne 3 du tableau `E`.

3. Créez un tableau `x` contenant les réels de 0 à 1 avec un pas de 0.1 entre deux valeurs consécutives.
   Remarque importante pour la suite des TP : pour définir un élément u_i d'une suite de valeur $(u_i)_{i\geq 0}$, on utilisera donc dans le programme la syntaxe `u(i)`.
4. Les opérations d'addition, soustraction et multiplication sont définies comme suit :

```
x = b + alpha
y = alpha -b
xx = alpha/x
yy = x*y
z = x**3
z = (x**2)**(1/2)
z = x**y
x^y
```

   Comme vous pouvez remarquer, pour l'opération a^b il y a deux possibilités : `a^b` ou `a**b`.
5. Les constantes mathématiques comme π or i ou e sont obtenues par `%pi`, `%i` et `%e`. Essayez les commandes suivantes :

```
%i*%i
%e**(%i*%pi)
```

6. **Concaténation**
   La concaténation consiste à créer de nouveaux tableaux à partir de tableaux existants en les agglomérant. Essayez les commandes suivantes :

```
A = [1,2;6,7] , B = [3,4,5;8,9,10]
C = [11,12], D = [13,14,15]
AA = [A,B;C,D] , BB = [B,A;C,D]
CC = [A,D;C,B]
T = 0:0.1:1
T1 = [T,1.1], T2=[T1;T1]
X = 0:0.1:2*%pi
[X,sin(X)], [X;sin(X)]
Y = [], Z = [Y,1], Z = [Z,2,3], Z = [Z,4,5]
Z = [Z;Z]
```

7. Affichage graphique

Vérifiez d'abord que `X` et `T` sont bien définis dans Scilab et essayez les commandes suivantes :

```
plot2d(X,cos(X))
clf()
plot2d(T',[exp(T'),exp(2*T')])
clf()
plot2d(X,cos(X),-3,rect=[0,-2,2*%pi,2])
plot2d(X',[sin(X'),sin(2*X'),sin(3*X')],[1,2,3],leg="sin(x)@sin(2x)@sin(3x)", ...
...rect=[0,-2,2*%pi,2])
A_1 = [0:1:5]
A_2 = [6:1:12]
[X,Y] = meshgrid(A_1,A_2)
Z = cos(X.^2 + Y.^2)
mesh(X,Y,Z)
surf(X,Y,Z)
```

Testez avec d'autres fonctions. Pour plus d'options d'affichage, consultez la documentation de Scilab.

8. Déclaration en ligne de fonctions

```
deff('[f]=f(z)','f=sin(z)./(1+abs(z)).^2-cos(z).*log(1+abs(z).^4)');
format(7)
f(rand(3,3,2)+ %i*rand(3,3,2))
```

Testez la commande `format()` avec d'autres valeurs pour comprendre sa fonction.

9. Maintenant, nous allons voir comment écrire des fonctions dans Scilab. Cliquez sur le bouton "Editor" et un éditeur de texte spécifique à Scilab va apparaitre. Créez un répertoire, par exemple "TPscilab", et dans ce répertoire créez et sauvez un fichier intitulé "exemple1.sci". Ouvrez ce fichier avec l'éditeur de texte de Scilab, et écrivez dedans :

```
// premiere fonction
function r=poly(x)
   r= x^3-2*x+1
endfunction

// deuxieme fonction
function resultat=g(toto)
   resultat = 2*toto
endfunction
```

Vérifiez que dans Scilab, vous êtes bien positionnés dans le répertoire "TPscilab" et lancez l'exécution du fichier en cliquant sur le bouton "Exécuter" ou "Enregistrer et exécuter". Puis tapez dans la fenêtre de commande

```
poly(3)
g(1)
poly(%pi)
```

Si vous observez la première fonction, `r` est le résultat de la fonction, `poly` est le nom de la fonction et `x` le nom de la variable à laquelle l'utilisateur doit fournir une valeur quand il utilise la fonction.

Les commentaires sont écrits après `//`. Le bouton "Exécuter" ou "Enregistrer et exécuter" sert à charger dans Scilab le contenu du fichier "exemple1.sci". Chaque fois que vous changez ce fichier, vous devez le sauvegarder et le re-charger en cliquant sur le bouton, avant de pouvoir l'utiliser.

10. Boucle "for"

Dans le même fichier, écrivez la fonction suivante, qui calcule avec un algorithme une approximation de la racine carrée d'un nombre.

```
// Algorithme sumerien-babylonien pour approx racine carree
function x = sumbab(a)
   x=1
   n=5
   for i=1:n
      x = (x+a/x)/2
   end
endfunction
```

Testez cette fonction sur plusieurs exemples. Faites varier le paramètre `n`.

11. Boucle "while"

Toujours dans le même fichier, écrivez la fonction suivante :

```
// Algorithme sumerien-babylonien pour approx racine carree v2
function x = sumbab2(a)
   x=1
   i=0
   while abs(x^2-a) > 1.E-14
    x = (x+a/x)/2;
i=i+1 // iterations number
   else // we print the result
        x
   end
endfunction
```

Testez cette fonction sur plusieurs exemples, comparez-la à la version précédente.

12. Comment stopper un calcul en cours

Il arrive qu'un calcul soit trop long, ou qu'une erreur algorithmique nous force à interrompre le calcul en cours. Dans ce cas, taper la commande `CTRL+C` suivi de la commande `abort`. Lancez une boucle de calcul très longue et faites un essai.

13. Structure "if/then/else/elseif

Dans un autre fichier intitulé "exemple2.sci", écrivez le code suivant :

```
function resultat = racine(a,b,c)
d = b^2-4*a*c;
if a==0
     resultat = 'degenerated'
elseif d>0 then
     x1 = (-b-sqrt(d))/(2*a);
               x2 = (-b+sqrt(d))/(2*a);
    resultat = [x1 x2];
```

```
elseif d==0 then
     x1 = -b/(2*a);
     resultat = [x1 x1];
else
     x1=(-b-%i*sqrt(-d))/(2*a);
     x2=(-b+%i*sqrt(-d))/(2*a);
     resultat = [x1 x2];
end
endfunction
```

A quoi sert cet algorithme ?

14. Il est possible de lancer une série de commandes directement à partir d'un fichier où elles sont écrites. Cela permet de gagner du temps en évitant de les écrire plusieurs fois de suite dans la fenêtre de commande. Ouvrez un nouveau fichier avec l'éditeur de texte Scilab et écrivez y les commandes suivantes : Traditionnellement, en Scilab les fichiers avec l'extension ".sci" contiennent des fonctions, et les fichiers avec l'extension ".sce" contiennent des suites de commandes. Ce n'est pas obligatoire, mais pour la lisibilité de vos programme vous êtes très fortement encouragés à le faire.

```
L1 = [];
L2 = [];
n=16
  for i=1:n
x = sumbab(i);
y = sumbab2(i);
L1 = [L1,x];
L2 = [L2,y];
end

L1
L2
```

Sauvez-le sous le nom "script.sce", puis exécutez son contenu en ligne de commande avec le bouton "Exécuter".

15. La conjecture de Syracuse

Soit $\phi : \mathbb{N}^* \to \mathbb{N}^*$ l'application telle que $\phi(n) = \frac{n}{2}$ si n est pair et $\phi(n) = 3n + 1$ si n est impair. On appelle suite de Syracuse toute suite $(s_n)_{n\in\mathbb{N}}$ définie par son premier terme $s_0 \in \mathbb{N}^*$ et par la relation $\forall n \in \mathbb{N}$, $s_{n+1} = \phi(s_n)$.
La conjecture de Syracuse postule que $(s_n)_{n\in\mathbb{N}}$ finit toujours par atteindre la valeur 1 et répète le motif 1, 4, 2, 1, 4, 2.... En dépit de la simplicité de son énoncé, cette conjecture n'a pas encore été prouvée. Elle mobilisa tant les mathématiciens américains durant les années 1960, en pleine guerre froide, qu'une plaisanterie courut selon laquelle ce problème faisait partie d'un complot soviétique visant à ralentir la recherche américaine (source Wikipedia).

(a) Ecrivez une fonction `function u = syracuse(a,n)` qui calcule le n-ième terme de la suite de Syracuse initiée par a.

(b) Combien de temps faut-il pour atteindre 1 et quelle est la valeur maximale de la suite lorsque $s_0 = 15$ et $s_0 = 127$? Modifiez la fonction précédente pour répondre à ces questions.

16. Méthode de Monte-Carlo

La méthode de Monte-Carlo vise à calculer de manière approchée l'intégrale d'une fonction en utilisant un procédé aléatoire. Le principe est le suivant : on sait que l'intégrale d'une fonction f sur l'intervalle

$[a, b]$ est directement reliée à la valeur moyenne de cette fonction sur cet intervalle. En effet, si on note M cette valeur moyenne, on a

$$M(b-a) = \int_a^b f(x)\,dx$$

L'idée est de tirer aléatoirement (avec une probabilité uniforme) des valeurs $(x_1, x_2, ..., x_N)$ comprises entre a et b, et d'approximer (d'après la loi des grands nombres) cette valeur moyenne M par la moyenne des valeurs $(f(x_1), f(x_2), ..., f(x_N))$:

$$M \approx \frac{1}{N} \sum_{i=1}^{N} f(x_i)$$

On obtient donc l'approximation de l'intégrale de f suivante :

$$\int_a^b f(x)\,dx \approx \frac{(b-a)}{N} \sum_{i=1}^{N} f(x_i)$$

(a) Ecrivez une fonction `function E = montecarlo(f,N)` qui calcule l'approximation de l'intégrale de la fonction f sur l'intervalle $[0, 1]$, à partir de N tirs aléatoires compris entre 0 et 1, par la méthode de Monte Carlo.

(b) Par un choix approprié de la fonction f, utilisez la méthode de Monte Carlo pour calculer une approximation du nombre π.