Université de Paris Diderot **2007-2008**
Analyse Numérique Matricielle **L3 Option Maths appli**

Fonctions et graphiques sous Scilab

---

(1) Nous allons dans ce Tp écrire des fonctions en les sauvant dans des fichiers. Pour cela cliquez sur le bouton "Editor" et un éditeur de texte se lance. Créez un répertoire *tpscilab* et sauvegardez-y un fichier qui s'appelle $exemple - 1.sci$. Si l'éditeur inclus dans Scilab ne fonctionne pas, il faut taper dans un terminal X (une fenêtre de l'environnement) la commande "emacs exemple-1.sci".

(2) Dans le fichier $exemple - 1.sci$, ecrivez le texte:

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

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

Dans Scilab, lancer

```
getf ./tpscilab/exemple-1.sci
poly(3)
g(1,2)
poly(%pi)
```

Entre les commandes $function$ et $endfunction$, on met d'abord la commande $r = poly(x)$. Dans ce cas r sera le résultat de la fonction et doit être défini dans le corps de la fonction. *poly* est le nom de la fonction et x le nom de la variable que l'utilisateur entrera quand il appelera la fonction. Dans la commande $poly(3)$, le programme lancera la fonction *poly* en remplacant partout x par 3. Les lignes précedées de // sont des commentaires, utiles pour comprendre les programmes de autres et même les siens, mais qui ne sont pas lus par la machien.
La commande "getf" quant à elle permet de charger le fichier, ici le fichier se trouve dans le sous-répertoire "tpscilab" et s'appelle "exemple-1.sci". **Attention** , à chaque fois que vous modidifierez le fichier, il faudra le recharger avec la commande "getf".

(3) **Les boucles** for. A la suite de votre fichier exemple-1.sci, écrivez La fonction suivante qui calcule la racine carrée d'un nombre (tournez la page):

```
//algorithme sumero babylonien de calcul de la racine carre
function x=sumbab(a)
   x=1
   n=5
   for i=1:n
      x=(x+a/x)/2
   end
endfunction
```

Que fait cet algorithme? Testez cette fonction avec plusieurs exemples.

(4) Les boucle *while*. Toujours à la suite de votre fichier, tapez la fonction suivante:

```
//algorithme sumero babylonien de calcul de la racine carre
function x=sumbab2(a)
x=1
i=0
while abs(x^2-a)>1.E-14
   x=(x+a/x)/2;
   i=i+1 //nombre d'iterations
   else  x //on imprime le resultat
end
endfunction
```

Testez cette fonction avec plusieurs exemples. Comment obtenir après un calcul le nombre d'itération utilisées?

(5) Structure if/then/else/elseif..

Dans un fichier "exemple-2.sci" tapez le code suivant :

```
function resultat=racine(a,b,c)
d=b^2-4*a*c;
if a==0
   resultat='degenere'
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
```

Que fait cet algorithme?

(6) **Les graphiques sous Scilab: plot**

Pour afficher une courbe sous scilab, la commande est *plot2d*. Le premier argument de cette fonction est un vecteur (ligne ou colonne mais de préference colonne) contenant des abscisses des points et le deuxième un vecteur contenant les ordonnées des points. Essayez (sans refermer la figure):

```
X=0:0.01:%pi;
plot2d(X,sin(X))
plot2d(X',sin(2*X))
plot2d(X,sin(3*X))
clf()
plot2d(X,sin(X))
```

(7) On peut afficher plusieurs courbes en même temps à condition que le vecteur des abscisses soit un vecteur colonne et de ranger les ordonnées des fonctions à tracer en colonnes dans la matrice des ordonnées. Parmi les instructions ci-dessous, quelle est celle qui fonctionne? Pourquoi ?

```
X=[0:0.01:%pi];
plot2d(X,[sin(X),sin(2*X)])
plot2d(X,[sin(X);sin(2*X)])
plot2d(X',[sin(X)',sin(2*X)'])
plot2d(X',[sin(X)';sin(2*X)'])
```

(8) On peut choisir certains modes d'affichage, comme le style de la courbe (pointillés, plein,, ect...) la couleur et la taille de la fenêtre d'affichage. Tapez la commande ci-dessous et expliquer les options.

```
X=[0:0.01:%pi];
plot2d(X',[sin(X)',sin(2*X)',sin(3*X)'],style=[1,-1,6],rect=[-1,-2,6,2])
```

(9) Enfin lancez la commande *plot2d* toute seule pour avoir une démonstration du foncitonnement de la commande. Il existe encore les commandes *plot2d1*,*plot2d2*,*plot2d3* et *plot2d4*. Lancez les avec un exemple et aidez vous de l'aide en ligne pour comprendre leur fonctionnement.

(10) Lancez la commande *plot3d* et analysez la syntaxe de la démo.

(11) Soit A une matrice symmétrique (A=A^T) définie positive (i.e. $v^T Av \geq 0$ pour tout v). On pose $X_0 = Id$ et pour tout n

$$X_{n+1} = \frac{1}{2}(X_n + X_n^{-1} A)$$

-Montrer par récurrence que cet algorithme est bien défini (c'est à dire que X_n^{-1} existe) en montrant que $X_n A = A X_n$ et que $X_n^T = X_n$ et enfin que si $X_{n+1} v = 0$ alors $v = 0$.
-En supposant que la suite X_n converge vers une matrice X_∞, montrez que $X_\infty = X_\infty^T$ et que $X_\infty^2 = A$ (X_∞ est appellée "racine carrée de A)"
-A l'aide de cet algorithme , écrire dans Scilab une fonction *Racinemat* qui donne la racine carrée d'une matrice quand on lui donne une matrice symétrique, et qui répond que la matrice n'est pas symétrique sinon.