Feuille de TP n° 1-Introduction à Scilab

1 Description

Scilab est un *logiciel libre* basé sur le principe que tout calcul, programmation ou tracé graphique peut se faire à partir de matrices. En scilab tout est matrice (scalaires, vecteurs lignes et colonnes...) Il a été développé conjointement par l'INRIA et l'ENPC. Vous trouverez à l'URL

`http://ljk.imag.fr/membres/Bernard.Ycart/`

ainsi qu'à l'URL

`http://www.iecn.u-nancy.fr/~pincon/scilab/scilab.html`

des manuels pour débuter en Scilab. Vous pouvez aussi télécharger Scilab à l'adresse suivante :

`http://www.scilab.org/`

Dans toute la suite vous exécuterez chaque ligne de commande à et tiendrez comptes des annotations. On pourra écrire les lignes de code dans l'éditeur scipad en tapant par exemple `scipad();` dans scilab et les executer ensuite dans Scilab.

2 Premières manipulations

2.1 Matrices

▷ `M=[1,2,3,4;5,6,7,8;9,10,11,12]` // Définit une matrice 3×4.
Réessayez avec un point virgule à la fin de la ligne. Quelle différence ?

▷ `M` // Affiche M.

▷ `a=M(2,3); L=M(2,:); C=M(:,3); d=M(1,$); dd=M([2,3],$); E=M([1,2,3],[2,3,4]); l=L([1,4]); c=C([1,2]);`
Affichez chaque variable et trouvez à quoi correspondent ces manipulations.

▷ `M(3,4)=0; M(:,1)=[1;2;3]; E([1,2],:)=[]; N=M';`
Même question.

▷ `U1=ones(M); U2=ones(a); U3=ones(1,2); Z1=zeros(M); Z2=zeros(a); Z3=zeros(2,2); s=size(M); t=size(a);`
Que font ces commandes ?

▷ `x0=[1:3]; x1=[0:0.2:1]; P=[-9,x0;x0,-9];` // Fabrication de vecteurs et de matrices par blocs.

2.2 Opérations et Booléens

On dispose des opérations usuelles sur les matrices $+, -, *$ ainsi que la puissance `^` et la division par un scalaire /. En faisant précéder d'un point ces opérations on les effectue sur tous les coeficients de la matrice.

▷ `E^3, E.^3, 1 ./ L, 1 ./ C, M.*M, M./M, 100 + M, L./[1:4]`

Les fonctions habituelles sont déjà implémentées.

▷ `sin(%pi/6), cos(M), log(L), exp(C), floor([1.5,-0.8])`

▷ `B=(M>4), R=1*B`

Que font les commandes suivantes ?

▷ `x=[-2:0.5:2]; b=(x>-1)&(x<1); y=x(b); x(b)=%pi; x`

2.3 Aide Scilab

L'aide en ligne est appelée par la commande `help`. Prenez le temps d'en parcourir les pages en exécutant les commandes suivantes :

▷ `help help`

▷ `help matrix; help bool` // Renvoie respectivement à toutes les rubriques portant sur les matrices et sur les booléens.

▷ `help ones; help zeros; help sum; help cumsum; help sort; help linspace; help eye; help diag` etc...

Avant de poursuivre :

▷ `who`

Affiche toutes les variables.

▷ `clear`

Efface toutes les variables crées lors de la session.

Remarque : Il suffit d'appuyer sur la touche ↑ pour faire défiler les commandes précédentes.

2.4 Exercices

► `z=cos([1:100]);` Trier par ordre croissant les éléments positifs ou nuls de z.

► `u=sin([1:1000]); v=cos([1:1000]);` Calculer le produit scalaire de u et v puis calculer, uniquement à l'aide de la fonction `ones`, la somme des éléments du vecteurs u. En effectuant le moins d'opérations possibles compter le nombre d'éléments de u inférieurs ou égaux à $\frac{1}{2}$ en valeur absolue et calculer leur somme.

► `w=sin([1:10]);` Construire un vecteur W tel que sa n-ème coordonnée, $1 \leq n \leq 10$, soit égale à $\frac{w_1+\cdots w_n}{n}$.

3 Générateur pseudo-aléatoire

3.1 La fonction *rand*

Cette fonction utilise le générateur congruentiel linéaire suivant :

$$X_{n+1} = aX_n + b\,[m].$$

Avec par exemple $m = 2^{31}$, $a = 843314861$ et $c = 453816693$.
$X_0 \in [|0, m-1|]$ est appelé le germe. Le premier nombre donné par la machine est $\frac{X_1}{m}$, le $n$ème est $\frac{X_n}{m}$. Ainsi une suite « aléatoire »sur un ordinateur n'est rien d'autre qu'une suite parfaitement déterministe. La qualité d'un générateur pseudo-aléatoire est donnée par un ensemble de testes statistiques. `rand` est un générateur pseudo-aléatoire « uniforme »sur $[0, 1[$. Par défaut le germe est souvent $X_0 = 0$. Ainsi le premier appel à la fonction `rand` donnera toujours

$$X_1 = \frac{C}{m} \simeq 0.2113249.$$

▷ `rand`
On peut changer le germe à tout moment.

▷ `rand("seed",germe)` // Où « germe »est un nombre entier compris entre 0 et $m-1$.

▷ `rand(2,3), l=[2008,2009;1983,3006]; rand(l), rand([1:5])`

3.2 La fonction *grand*

La fonction suivante est fondamentale quand on veut faire des probabilités avec Scilab.

▷ `help grand`

▷ `a=grand(1,10,'nor',0,1)` // Génère 10 réalisations « indépendantes »d'une « loi normale centrée réduite ». Essayez avec d'autres lois.

Rappels : Une variable aléatoire X suit une loi normale centrée réduite si elle admet pour densité sur $\mathbb{R}$ la fonction définie par

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp \frac{-x^2}{2}.$$

Donnez son espérance et sa variance. A quoi est égale sa fonction caractéristique ?

3.3 Exercices

► A l'aide de la fonction `rand` simuler une loi uniforme sur l'intervalle $[-1, 1]$, une loi uniforme sur l'ensemble $\{0, \cdots, 9\}$, une loi de Bernoulli de paramètre $p = \frac{1}{3}$, une loi binomiale de paramètre $p = \frac{1}{3}$ et $N = 10$, ainsi que 5 réalisations "indépendantes" de la loi précédente (en une seule ligne !)

► Une particule initialement en $x = 0$ se déplace sur $\mathbb{Z}$ aléatoirement avec une probabilité $p = \frac{13}{25}$ d'un pas sur la droite et une probabilité $q = 1 - p$ d'un pas sur la gauche, les sauts étant indépendants. Simulez 1000 déplacements de cette particule. Compter la proportion de temps passé à droite et à gauche de 0. Qu'en pensez vous ?

4 Graphisme

4.1 Fonctions

▷ `x=linspace(0,2*%pi,5); y=x.*sin(x); plot(x,y);`
`xx=linspace(0,2*%pi,100); yy=xx.*sin(xx); plot(xx,xy)`
// Les deux courbes sont affichées sur le même graphique avec la même couleur.
▷ `xbasc()` // Efface la fenêtre graphique courante.
▷ `plot2d(x,y,1); plot2d(xx,yy,5)`
// Affiche les deux courbes avec des couleurs différentes.
▷ `xbasc(); plot2d(x,y); xset('window',1); plot2d(xx,yy,5)`
// Affiche les deux courbes dans des fenêtres distinctes.
▷ `xbasc([0,1]); xset('window',0); plot2d(xx',[yy;2*yy]')`
// Efface les fenêtres 0 et 1 et affiche deux courbes simultanément dans la fenêtre 0.

Pour plus de détails et de fonctionnalités on se reportera à l'aide de `xset`, `plot2d` `plotframe` et `subplot`.

▷ `x=[1:100]; y=cumsum(-1 + 2*grand(x,'bin',1,0.5)); xbasc([0,1]);`
`xset('window',0); plot2d2(x,y)`
Que simule ce graphe ?
▷ `xset('window',1); z=y./x; plot2d(x, z)`
Même question.
▷ `xbasc([0,1]); xset('windows',0); g=1 + 5*grand(1,1000,'nor',0,1);`
`histplot(100,g); xtitle('Histogramme')`
// Trace un histogramme dans la fenêtre 0 et affiche le titre « histogramme ».
Se reporter à l'aide en ligne pour la fonction `histplot`.

4.2 Exercices

▶ Dessiner la densité d'une loi normale centrée réduite sur $[-10, 10]$ et y superposer des histogrammes obtenus à partir d'échantillon de loi normale de longeur $100, 1000, 10000$. Que remarquez vous ? Tracer des histogrammes des lois usuelles que vous connaissez.

▶ r=1grand(1000,10,'nor',0,1) ; xbasc([0,1]) ; xset('window',0) ;+
Que représente les commandes suivantes :
`plot2d([1:1000]',cumsum(r,'c')./([1:1000]'*ones(1,10)));`
`plotframe([0,-1.5,1000,3.5],[10,5,10,5])`

▶ `R=grand(1000,1000,'bin',1,0.5);`
`h=1000^0.5*(-0.5 + sum(R,'c')./1000 ); histplot(100,h,5)`
Que remarquez vous ?