Feuille 1 (TP 1 et 2)- Débuts sous Scilab

1 Premiers pas

1. Après avoir lancé une session Scilab, taper les commandes suivantes : `2+2`, `2*%pi`, `4/%e`, `1-%eps`, `%i^2`, `ans+1`, `ans+1`, `exp(2)`, `sin(%pi/4)`, `sqrt(2)/2`, `sqrt(-1)`, `sqrt(%i)`. Réessayer quelques unes de ces commandes en ajoutant un point-virgule.
2. Taper la commande `rand ()`. Comparer avec $453816693/2^{31}$ (D'où vient ce facteur 2^{31} ?). Réessayer cette commande `rand()` plusieurs fois de suite (penser à utiliser la touche "flèche haut"). Tester les commandes `rand(1,1)`, `rand(1,2)`, `rand(2,2)`, `rand(2,3)`.
3. Pour affecter une valeur à la variable `u`, il suffit d'entrer `u=` la valeur, par exemple `u=rand()`. Taper alors les commandes `a=1`, `a=a+1`.
4. Les constantes booléennes Vrai et Faux sont `%t`, `%f` sous Scilab. `|` et `&` correspondent au Ou et Et logiques. `~` correspond à la négation logique. Scilab affiche les variables booléennes sous la forme `T` et `F` pour Vrai et Faux.
   (a) Taper `a=3>2`, `b=1>2`, `a|b`, `a&b`, `~a`.
   (b) Deviner à quoi correspondent en logique `1`, `0`, `==`, `~=` en testant les commandes `1==0`, `1~=0`. Quelle valeur Scilab devrait elle retourner si on tape `1==0&1~=0` et `1==0|1~=0` ? Le vérifier. Tester la commande `bool2s` et deviner à quoi elle sert.
5. Pour utiliser l'aide sous Scilab on peut :
   — cliquer sur le carré avec un `?`,
   — cliquer sur le `?` dans le bandeau supérieur puis sur Aide Scilab,
   — ou taper directement `help` suivi du nom de la commande (si vous la connaissez...).

   Chercher l'aide sur `rand` ainsi que sur quelques commandes.

2 Calcul matriciel

2.1 Vecteurs

1. La création manuelle d'un vecteur ligne se fait en tapant les valeurs entre crochet séparées par des virgules, ex. `[1,2,3]`. Les virgules servent à rendre la lecture plus simple, mais ne sont pas indispensables, ex. `[1 2 3]`.
2. Création automatique d'un vecteur ligne : tester les commandes `zeros(1,5)`, `ones(1,10)`, `[4:9]`, `[4:2:9]`, `[0:.1:1]`, `[9:-1:4]`.
3. Après avoir affecté à `x` le vecteur lignes de composantes 1, 4 et 3, tester les commandes suivantes :
   `x(2) ones(x) x' min(x) max(x) [m k]=max(x) mean(x) x+1 2*x length(x) exp(x) log(x) sqrt(x) sum(x) cumsum(x) sort(x) -sort(-x) [s k]=gsort(x) gsort(x,"g","i") x($) x($-1) floor(1.5*x)`.
   Regarder l'aide relative aux commandes `sort` et `gsort` (regarder en particulier les options et en tester quelques unes), ainsi qu'à celles qui ne vous semblent pas claires au premier coup d'oeil. Comment crée-t-on automatiquement des vecteurs colonnes ?

2.2 Matrices

1. La création manuelle de matrices se fait de la même manière que pour les vecteurs, en séparant les lignes par un point-virgule. Affecter à `A` la matrice dont les lignes sont 1 2 3, 4 5 6, 7 8 9 puis tester :
   `A(2,3) length(A) size(A) size(x) A^2 A.^2 A*x' A.*ones(A) A*ones(A)`.
2. Extraction de lignes, colonne et sous matrices d'une matrice : tester (et comprendre !) `l=A(2,:) c=A(:,1) B=A(1:2,:)`.
3. Essayer de deviner ce que fait la commande `A(1,:)=[]`, puis le tester.
4. Création d'une matrice par blocs : Tester `P=[x 2*x; A ones(A);3*x zeros(x)]`.
5. Scilab permet de faire des tests directement sur les composantes de vecteurs ou matrices : Tester `[4 0 2 1 0] == 0`, `[4 0 2 1 0] > 0`.

3 Lois de probabilité usuelles et graphiques

1. Question préliminaire : à l'aide de ce que vous avez vu jusqu'à présent, écrire une commande qui crée un vecteur ligne comprenant 100 réalisations d'une loi de Bernoulli $\mathcal{B}(1/2)$ (Pensez à utiliser `bool2s`).
2. En fait, Scilab permet de simuler facilement des lois classiques, sans avoir à écrire des commandes (comme dans la question précédente) pour des lois de base. Tester les commandes `grand(1,5,"exp",2)` `grand(1,5,"nor",1,2)`. Que signifient d'après vous les paramètres 2 et 1,2 après exp et nor ? Consulter l'aide sur `grand` et vérifier.
3. Taper les commandes suivantes : `histplot(100,grand(1,1000,"exp",2))` `xtitle('Histogramme loi exponentielle')`. Consulter l'aide sur `histplot`.
4. Expérimenter et comprendre la fonction `plot2d` sur les exemples suivants : `plot2d(cumsum(rand(1,1000))./[1:1000])`
   `clf; plot2d(cumsum(rand(1,1000))./[1:1000])`
   `plot2d([1:1000],cumsum(rand(1,1000))./[1:1000],5)`
   `x=[1:100]; plot2d(x',[(100*x)' (x^2)'])`
   On peut utiliser `getcolor()` afin de changer les couleurs des courbes dans un même graphe. Superposer le graphe de la densité de la loi exponentielle de bon paramètre sur l'histogramme précédent. Que pouvez vous dire ? Que se passe-t-il si on augmente la taille de l'échantillon ?
5. Refaire la même manipulation avec la loi $\mathcal{N}(1, 2)$.

4 Fonctions et programmation

L'interface que vous avez utilisée jusqu'à présent est telle que Scilab exécute les commandes au fur et à mesure que vous les rentrez. Il est possible de créer des fonctions que vous utiliserez alors par la suite autant de fois que vous le voulez. Un éditeur est mis à votre disposition pour écrire ces fonctions : cliquez sur Editeur. Une nouvelle fenêtre s'ouvre alors : vous pouvez alors taper vos fonctions et commandes, puis les sauvegarder (penser à le faire fréquemment) avec l'extension .sci (si le fichier ne contient que des fonctions) ou .sce (si le fichier ne contient que des lignes de commande). Pour compiler votre fichier, cliquez sur Execute puis load into Scilab (ou ctrl-l).

4.1 Fonctions

La syntaxe pour créer une fonction est
`function` *valeurs renvoyées* `= nom de la fonction (`*arguments)*

suivi des instructions. Une fonction se termine par `endfunction`. Les valeurs renvoyées se présentent sous forme de vecteur. Après avoir chargé la fonction dans Scilab, la fonction s'utilise comme tout autre fonction du logiciel. Tester par exemple

```
function [x,y]=additionsoustraction(a,b)
    x=a+b
    y=a-b
endfunction;
```

1. Ecrire une fonction créant un vecteur aléatoire de taille n (pris en argument) d'échantillon de quelques lois quelconques (uniforme, exponentielle,...).
2. En utilisant `rand` et `bool2s`, écrire une fonction créant un vecteur aléatoire de taille n (pris en argument) d'échantillon de loi de Bernoulli de paramètre p (également pris en argument).

4.2 Boucles et instructions conditionnelles

Les instructions les plus utilisées sont `for` et `while`. Tester

```
for i=1:10 do
  disp(i)
end;
m=10
while m>0 do
  disp(m)
  m=m-1
end;
```

Ecrire une fonction calculant $n!$ (n pris en argument) grâce à `for` et `while`. Après avoir consulté l'aide sur `if... then... else`, écrire la fonction sous forme récursive. On en profitera ensuite pour consulter l'aide sur `factorial`.

4.3 Exercices

1. Créer une fonction qui compare les probabilités théoriques et observées pour la loi binomiale. Faites apparaître les valeurs théoriques (penser à reconsulter l'aide sur `grand` et `binomial`) sous forme de croix (avec `plot2d(x,y,-2)`) et les valeurs observées sous la forme d'un diagramme bâton (avec `plot2d3`). Idem avec la loi géométrique.
2. Ensuite, recommencez en simulant vous même la loi binomiale (utiliser `rand`) et la loi géométrique. Comparer les temps d'exécution avec `timer`.