FICHE TP 1

Premiers pas avec Scilab

L'objectif de ce TP est de se familiariser avec le logiciel Scilab (qui est libre). C'est un interpréteur de commandes écrites en langage Scilab. Ce logiciel a été conçu pour faciliter les opérations sur les vecteurs, matrices et tableaux.
Pour les probabilités, on utilisera les fonctions du module « stixbox » que l'on peut facilement télécharger, qui est disponible le jour de l'agrégation.

Lorsqu'on lance Scilab une fenêtre s'ouvre, comportant la console où sont tapées les instructions et où s'affichent les résultats. On peut aussi, si elles ne sont pas déjà visibles, à partir de l'onglet « Applications », faire apparaître les fenêtres avec l'« historique des commandes », avec le « navigateur de variables » (qui montre les variables utilisées), ainsi que l'éditeur « Scinotes » qui servira beaucoup.
Enfin on peut changer si nécessaire le répertoire courant avec l'onglet « Fichier ».

L'historique des commandes permet de récupérer des instructions déjà saisies : en double cliquant sur une de ces instructions passées, on la réexécute.

Pour sauver certaines commandes de l'historique dans un fichier, on peut par exemple sélectionner des entrées dans cette fenêtre faire un clic droit et les ouvrir dans l'éditeur. Attention l'historique peut dépendre de l'ordinateur sur lequel on travaille. Le jour de l'oral de l'agrégation, par exemple, bien penser à sauver toutes les commandes que vous voulez montrer au jury dans un fichier...

$\uparrow$, $\downarrow$, $\rightarrow$, $\leftarrow$ permettent de se déplacer dans les lignes de commandes tapées dans la console. Cela évite de retaper des instructions !
La tabulation permet quand on a commencé de taper une commande de rechercher une complétion automatique.

Enfin on peut changer si nécessaire le répertoire courant avec l'onglet « Fichier ».
On peut afficher l'aide à l'aide de F1. De manière générale, `help nom` fournit de l'aide sur le mot clé « nom ».

1. Trouver en utilisant l'aide une fonction qui donne les valeurs propres d'une matrice.

a. Créations et Manipulations de variables sous Scilab.

`a=4;b=%pi` La valeur 4 est affectée à a et la valeur π à b.
Le point virgule a pour effet que matlab n'affiche pas le résultat. Remarque : %pi et %i contiennent respectivement la constante π et le nombre imaginaire i.

`A=[1 2 3;4 5 6; 7 8 9]`	Création « à la main » de la matrice A.
`C=A'`	C est la matrice adjointe de A

Les colonnes des matrices sont séparées par des espaces (ou des virgules), les lignes par des points virgule, ou par des retours à la ligne.
`A(2,3)` désigne l'élément de la deuxième ligne et de la troisième colonne de A.

2. A votre avis que font les commandes suivantes ? Vérifier.

```
[A,[0 0 0]']
[A;[0,0,0]]
```

b. Utilisation des deux points (:)

`X=[1:20]`	X est le vecteur ligne contenant les entiers de 1 à 20.
`X=[X,21]`	rajoute un élément au vecteur X.
`Y=[1:2:20]`	Y est le vecteur ligne contenant les entiers $1, 3, \cdots, 19$.
`Z=linspace(1,5,50)`	Z est un vecteur ligne contenant 50 valeurs régulièrement réparties entre 1 et 5.

3. A votre avis que font les commandes suivantes ? Vérifier.

```
2+1:5
2+(1:5)
[2:1/10:%pi]
```

`b=A(2,:)`	b est le vecteur ligne contenant la deuxième ligne de A.
`c=A(:,3)`	c est le vecteur colonne contenant la troisième colonne de A.
`B=A(1:2,1:2)`	B est la première sous-matrice carrée d'ordre 2 de A .

4. Obtenir la matrice contenant les deux premières colonnes de A.

c. Opérations élémentaires sur les matrices et les vecteurs, opérateurs logiques.

`length(A)`	donne le nombre d'éléments de A
`size(A)`	donne la taille de A
`A+C,A*C,2*A`	Addition et multiplications matricielles classiques
`A+1`	ajoute 1 à tous les éléments de A.
`n=4;I4=eye(n,n)`	Création de la matrice identité d'ordre 4
`B=zeros(n,n)`	B est la matrice nulle 4×4
`D=ones(3,2)`	D est la matrice 3×2 ne contenant que des 1 .
`p=2; A^p`	Calcul de A^2
`det(A),trace(A),spec(A)`	Déterminant, trace et valeurs propres de A
`sum(X)`	Somme des éléments d'un vecteur
`cumsum(X)`	Vecteur $[X_1, X_1 + X_2, \ldots, X_1 + \ldots X_n]$ (sommes cumulées) : très utile !

Effectuez les questions ci-dessous sans faire de boucle :

5. Obtenir un vecteur colonne de taille 10 dont tous les éléments valent 5.
6. Calculer le produit scalaire des deux premiers vecteurs colonnes de A (penser à utiliser la transposée)
7. Calculer la somme des 50 premiers entiers pairs.

La plupart des fonctions Scilab peuvent s'appliquer à des matrices ou à des vecteurs, et s'effectuent élément par élément : c'est le cas par exemple de `cos`, `sin`, `sqrt` et `exp`. Attention : si on veut calculer l'exponentielle d'une matrice il faut utiliser `expm`.
Cela marche aussi avec les opérateurs logiques : `|` pour « ou », `&` pour « et ».
essayer `(A<3)|(A>7)`, qui renvoie une matrice ne contenant que des F et des T.
Pour transformer les « F » et « T » en 0 et 1, la méthode la plus rapide consiste en fait à multiplier par 1. Essayer `1*(A>2)`.
Remarque : `sum` ajoute directement les « F » et les « T » comme si c'étaient des 0 et des 1.

8. Construire un vecteur contenant les racines carrées des entiers pairs de 2 à 10.
9. Construire un vecteur contenant les cosinus des entiers de 1 à 100. Utiliser `sum` et un opérateur logique pour savoir combien d'entiers de 1 à 100 ont un cosinus supérieur à 0.5.

d. Utilisation du point (.) L'opérateur point (`.`) permet de transformer une commande matricielle (typiquement, une opération « puissance ») en une commande élément par élément. Comparer par exemple `A^2` et `A.^2`, `A*C` et `A.*C`
Remarque : si X est un vecteur, `X^2` effectue directement la multiplication coordonnée par coordonnée.

Remarque : un entier suivi d'un point est compris comme une valeur réelle. Donc si on veut que le point soit compris comme ayant un lien avec l'opération qui suit, il faut un espace ! Exemple : comparer `1./A` et `1 ./A`

10. Construire (sans faire de boucle) un vecteur contenant les carrés des entiers de 1 à 10.

11. Construire (sans faire de boucle) un vecteur contenant les inverses des entiers de 1 à 10.

12. Construire (sans faire de boucle) un vecteur contenant les sommes harmoniques $1, 1+\frac{1}{2}, \ldots, 1+\frac{1}{2}+\ldots\frac{1}{10}$. (penser à `cumsum`)

e. Premières commandes statistiques. Rappel : le générateur aléatoire fournit par Scilab ne fait que « semblant » d'être aléatoire; en fait, les nombres fournis par des appels successifs à la fonction `rand` ne sont pas indépendants.

`rand`	Générateur aléatoire associé à la loi uniforme $\mathcal{U}([0,1])$.
`X=rand(1,1000);`	X est un vecteur ligne de 1000 réalisations i.i.d. $\mathcal{U}([0,1])$
`D=rand(2,3);`	D est une matrice 2×3 dont les éléments sont i.i.d. $\mathcal{U}([0,1])$
`m=mean(X)`	Calcule la moyenne empirique (i.e. arithmétique) de X
`v=variance(X)`	Calcule la variance empirique de X
`sigma=stdev(X)`	Calcule l'écart-type empirique de X

Un vecteur comportant n réalisations indépendantes d'une même loi sera aussi appelé n-échantillon.

13. Générer un 2000-échantillon $(X_n)_{1\leq n\leq 2000}$ de loi uniforme sur $[0,1]$. Comparer sa moyenne empirique avec $\mathbb{E}[X_1]$.
14. Comparer la moyenne empirique de X^3 avec $\mathbb{E}[X_1^3]$
15. Générer un 20-échantillon de loi uniforme sur $[0,1]$ et déterminer le nombre N d'éléments de votre échantillon qui sont inférieurs à 0.3. N est donc aléatoire. Quelle est sa loi ?

`grand(n,m,'nom',paramètres)` est la commande qui permet de simuler les lois usuelles.

16. Faire `help grand`. Trouver comment simuler une loi normale, une loi exponentielle, une loi de Poisson. Attention : pour Scilab, le paramètre de la loi exponentielle est son espérance, et le deuxième paramètre de la loi normale est son écart-type, pas sa variance.

17. Générer un 100 échantillon d'une loi normale centrée de variance 4. Obtenir sa moyenne empirique et sa variance empirique.

18. Générer un 100 échantillon d'une loi de Poisson de paramètre 2. Obtenir sa moyenne empirique et sa variance empirique.

f. Boucles. Toutes ces commandes permettent bien souvent de ne pas avoir à faire de boucle. Néanmoins ce n'est pas toujours possible... La syntaxe pour une boucle est :

```
for k=1:10
INSTRUCTIONS
end
```

Il est fortement déconseillé de taper des boucles directement dans la console. La manière la plus pratique dès qu'on commence à avoir des suites d'instructions à effectuer est d'ouvrir un fichier dans l'éditeur et d'y mettre ces instructions (le fichier sera un script « .sce »). On peut mettre des commentaires (c'est très utile quand on doit montrer son programme par exemple à un jury) en commençant une ligne par //

19. Créer un script *blabla.sce* qui construit la matrice 3×3 dont l'élément (i,j) est $\frac{1}{i+j-1}$ (utiliser deux boucles), calcule son déterminant, et son inverse.
Pour exécuter le script, il suffit normalement dans l'éditeur de faire F5, ou de sélectionner « exécuter »dans l'un des menus ou dans la barre des taches.
Par défaut, « exécuter » fonctionne comme s'il y avait des points virgule partout (les commandes sont effectuées, mais rien ne s'affiche). Pour changer cela, on peut sélectionner « exécuter avec echo », ou utiliser `disp` dans le script pour faire afficher les variables.

g. Représentations graphiques. Si `x` et `y` sont des vecteurs de même taille, la commande `plot2d(x,y)` fournit le graphe formé à partir des points $[x_i, y_i]$, par défaut reliés par des segments. On peut mettre ensuite des options : en

particulier, les entiers positifs correspondent à des couleurs, les entiers négatifs à des symboles pour les points (il n'y a alors plus de segment tracé). Ainsi, par exemple :

`t=linspace(-2,2,50); plot2d(t,exp(t)+3,5)`	graphe en rouge de la fonction $t \mapsto \exp(t)+3$.
`xtitle(''graphe'', ''temps'', ''fn'')`	Titre et légendes des axes

Si on fait plusieurs `plot2d` à la suite, ils seront sur la même figure, à moins de l'effacer avant avec `clf`. `figure`, ou ici `scf(2)`, ouvre une nouvelle fenêtre pour une nouvelle figure (si la fenêtre 2 n'existe pas ; sinon `scf(2)` utilise la fenêtre 2 pour la figure)

20. Obtenir sur le même graphe les courbes représentatives de *cos*, et de $x \mapsto 1 - \frac{x^2}{2}$. sur $[-\pi, \pi]$. Essayez d'avoir des couleurs différentes. Rajouter des légendes aux axes, Obtenir encore sur le même graphe une droite horizontale d'équation $y = 0.5$ et une droite verticale $x = \frac{\pi}{2}$.

h. Les fonctions. Une fonction Scilab est une procédure qui peut prendre en entrée des arguments, effectuer des instructions, et peut fournir un ou des résultats en sortie.

Remarque : une fonction peut aussi éventuellement n'avoir aucun argument et ne fournir aucun résultat.

Une fonction s'écrit soit dans un script, soit dans un fichier .sci à part. Un même fichier peut comporter plusieurs fonctions, et peut s'appeler comme on le souhaite. On peut par exemple mettre les commandes suivantes dans un fichier « truc.sci » :

```
function y=toto(t);
// Ligne de commentaire éventuelle
y=t^2-3;
z=y+1;
disp(z);
endfunction
```

Il faut ensuite exécuter le fichier qui contient la fonction, par exemple avec F5 (ou avec le menu Exécuter). On peut ensuite faire appel à la fonction. Même si on n'a pas mis de point virgule dans la fonction, Scilab fait comme s'il y en avait. Si on veut forcer la fonction à afficher une valeur pendant son exécution, il faut utiliser `disp(nomdelavariable)`.

Ici par exemple `toto(3)` affiche 7 et renvoie la valeur 6. `w=toto(3);` affiche 7 et affecte 6 à la variable w.

L'argument de toto peut aussi être un vecteur : essayer `toto([1:10])`

Une fonction peut aussi renvoyer plusieurs résultats. Par exemple pour une fonction « bidule » qui doit prendre en argument un réel et renvoyer deux réels, on a deux solutions :

• Soit on la déclare sous la forme `function [y1,y2]=bidule(x)`. Dans ce cas là, si on effectue `u=bidule(1)`, à la variable u sera uniquement affectée la valeur de $y1$. Pour obtenir les deux valeurs il faut effectuer `[u,v]=bidule(1)`.

• On peut aussi déclarer `function y=bidule(x)`, mais en faisant en sorte (dans la fonction) que y soit bien égal au vecteur $[y1, y2]$. Dans ce cas la fonction ne renvoie en fait qu'une variable, qui est un vecteur. Si on effectue `u=bidule(1)`, à u est affecté le vecteur $[y1, y2]$.

21. Créer une fonction qui prend en entrée un vecteur et renvoie la moyenne empirique et l'écart-type empirique du vecteur.
22. Créer une fonction qui prend en entrée x (réel ou vecteur) et un réel l et renvoie $l\exp(-lx)$.
23. Créer une fonction qui prend en entrée un réel l et dessine le graphe de la densité d'une loi exponentielle de paramètre l.