Université Lille 1 **U.F.R. de Mathématiques**

Licence MASS S6 **Année 2015-2016**

Formulaire

Petits rappels

- Pour démarrer SCILAB : cliquer sur l'icone correspondante ou en tapant `Scilab&` dans une fenêtre de commande.
- N'hésitez pas à cliquer sur `Help` et `Apropos` pour avoir des explications et des exemples sur les mots clés donnés dans cette fiche.
- `help nomcommande` donne aussi accès à l'aide.
- Pour faire des programmes, cliquer sur `Editor` et taper les commandes correspondantes dans la fenêtre de l'éditeur de programme. Sauver ce programme (en cliquant sur `File` et `Save`) sous le nom `toto.sce` puis cliquer sur `Execute` et `Load into scilab` pour l'exécuter.
- Pour ne pas afficher le résultat, il faut mettre un point virgule à la fin de chaque ligne de commande.
- Pour forcer l'affichage, on peut utiliser la commande `disp`.
- Pour mettre un commentaire, utilisez `//`.

Commandes usuelles de Scilab : vecteurs et matrices

- création d'une variable `x=nombre`
- e est noté `%e`, π est noté `%pi` et i est noté `%i`.
- les fonctions $\sqrt{}$, exp et ln sont notées `sqrt`, `exp` et `log`.
- crépation d'un vecteur `x=[nombre1,nombre2,...]` pour un vecteur ligne, `x=[nombre1;nombre2;nombre3]` pour un vecteur colonne.
- création d'un vecteur séquence : `x=[nombre1:nombre2:nombre3]`. Crée un vecteur allant de *nombre*1 à *nombre*3 par pas *nombre*2.
- longueur d'un vecteur : `length(x)`.
- accès à un élément d'un vecteur : `x(n° élément)`.
- concaténation de deux vecteurs : `x=[u,v]`.
- création d'une matrice : `A=[[n1,n2,n3];[n4,n5,n6]]` (les lignes sont séparées par des points-virgules).
- `zeros(n,p)` crée une matrice ne contenant que des 0.

— dimension d'une matrice : `size(A)`.
— accès à un élément d'une matrice : `A(n° ligne,n° colonne)`.
— concaténation de deux matrices : `C=[A,B]` (concaténation côte à côte) ou `C=[A;B]` (concaténation l'une en dessous de l'autre).

quelques manipulations

— les multiplications et les puissances sont des multiplications matricielles. Pour faire une multiplication ou une division terme à terme, on utilise les commandes `.*`, `./` ou `.^` pour les puissances.
— arrondi : Les fonctions `round`, `ceil` et `floor` permettent de donner des arrondis : `round` arrondi à l'entier le plus proche, `ceil` à l'entier supérieur, `floor` à l'entier inférieur.
— comparaison : les comparaisons `<`, `<=` (...) se font composante par composante. La commande `x<5` renvoie un vecteur de même longueur que x. On utilise la commande `==` pour tester l'égalité.
— algèbre : `modulo(a,b)` donne le reste de la division de a par b, `pmodulo(a,b)` calcule la partie entière de a divisé par b.
— `factorial(k)` calcule $k!$.
— tri des données : `gsort`. Pour trier un vecteur, on utilise `gsort(x,'g','i')`. $'g'$ indique que l'on trie tous les éléments, $'i'$ que l'on trie dans l'ordre croissant. Pour trier une matrice, on utilise `gsort(A,'r','i')` pour trier les lignes de A dans l'ordre croissant, `gsort(A,'c','i')` pour trier les colonnes de A dans l'ordre croissant. .
— les fonctions `max`, `min`, `sum`, `prod` s'utilisent aussi bien pour les vecteurs que pour les matrices. `sum(A)` calcule la somme de tous les éléments de A, `min(A,'c')` calcule le minimum pour chaque colonne de A.
— la fonction `unique` ne garde qu'une fois les éléments répétés d'un vecteur.
— la fonction `find` permet de ne garder que les éléments d'un vecteur qui vérifient une condition.

1 Probabilités

Les fonctions `mean`, `variance`, `covar`, `median` s'utilisent de la même manière que `min` et `max`.

Nom des lois usuelles :

— Loi normale : `nor`,

— loi Gamma : `gam`
— loi binomiale : `bin`
— loi de Poisson : `poi`
— loi du χ^2 : `chi`
— loi de Student : `t`
— loi exponentielle : `exp`
— loi uniforme : `unf`

Fonction de répartition et fonction quantile Scilab connaît la plupart des lois de probabilités usuelles. Nous allons utiliser la commande `cdfnomdelaloi`.

Elle permet de calculer la fonction de répartition, les quantiles, et d'autres choses bizarres. La fonction de répartition d'une binomialbe $\mathcal{B}(n,p)$ en un point x, dépend de (n,p), du point x où on la calcule, et dans ce cas sa valeur est unique. On peut donc calculer la valeur de la fonction de répartition en un point, ou si on connaît sa valeur et tous les autres paramètres : calculer x ou n ou p. On commence par dire ce que l'on cherche, puis l'on donne les autres paramètres.

— `cdfbin('PQ',x,n,p,1-p)` calcule la valeur de la fonction de répartition d'une $\mathcal{B}(n,p)$ au point x.
— `cdfbin('S',n,p,1-p,alpha,1-alpha)` calcule le quantile d'ordre α d'une $\mathcal{B}(n,p)$.

Pour la loi normale :

— `cdfnor('PQ',x,mean,std)` calcule la valeur de la fonction de répartition d'une $\mathcal{N}(mean, std^2)$ au point x.
— `cdfnor('S',x,mean,std,alpha,1-alpha)` calcule le quantile d'ordre α d'une $\mathcal{N}(mean, std^2)$.

Pour la loi de Poisson :

— `cdfpoi('PQ',S,p)` calculer la fonction de répartition d'une loi de Poisson de paramètre $Xlam$ au point S.
— `cdfpoi('S',p,alpha,1-alpha)` calcule le quantile d'ordre *alpha* d'une loi de Poisson de paramètre p.

Génération de variables aléatoires

La fonction `grand(nombre1,nombre2,'nomdelaloi',paramètres de la loi)` génère une matrice $m \times n$ de nombres aléatoires i.i.d générés suivant la loi demandée.

2 Graphiques

Il existe différents type de graphiques sous `scilab` : les graphes classiques, les histogrammes, les boîtes à moustaches . . . .

Pour les graphes les plus simples, on utilise les trois commandes `plot(x,y,options)`, `points(x,y,options)` et `lines(x,y,options)`. Ces trois commandes permettent de tracer y en fonction de x.
On utilise toujours `plot(x,y,'options')` tracer un graphique. Les options principales sont les couleurs et les styles de points. Les graphes se superposent naturellement, et Scilab recalcule la fenêtre automatiquement.

La fonction `bar` trace l'histogramme d'un vecteur.

La fonction `boxplot` trace la boîte à moustache d'un vecteur. Elle permet de tracer soit la boîte à moustache d'un seul vecteur, soit la boîte à moustache de plusieurs vecteurs côte à côte. La fonction `qqplot` permet de tracer le graphe quantile/quantile de deux vecteurs (X, Y). Les deux vecteurs n'ont pas besoin d'être de même longueur. Avec `qqline`, on peut ajouter la droite qui passe par les quartiles $Q1$ et $Q3$. La fonction `qqnorm` trace les quantiles empiriques d'un vecteur X par rapport aux quantiles théoriques de la loi normale $\mathcal{N}(0, 1)$.

On peut aussi tracer plusieurs graphiques côte à côte : pour cela on commence par partager la fenêtre des graphes en plusieurs cases grâce à la commande `split.screen(c(i,j))` (partage en i lignes et j colonnes). Pour tracer un graphe dans une case, on peut donner le numéro de la case : `screen(k)` et on trace ensuite le graphique grâce aux commandes habituelles. Il faut fermer les graphes à la fin, sinon tout se superpose. Pour cela, on utilise la commande `close.screen(all=TRUE)`.

3 Un peu de programmation

Le logiciel `Scilab` n'aime pas du tout les boucles. À chaque fois que cela est possible, il faut passer par les matrices et les opérations sur les vecteurs. (Une boucle pour 5 ou 10 opérations n'est bien sûr pas génante, une boucle for de 1000 itérations peut énormément ralentir le programme).

Structure d'une boucle for :

```
for (k=1:n)
  lignes codes
end
```

Conditions

```
if condition then
  lignes codes
elseif condition then
  lignes codes
else
  lignes codes
end
```

On n'est pas obligé de mettre un `elseif` ou un `else`. `Scilab` ne sait comparer pas utiliser la condition `if` sur un vecteur. La structure `if/then/else` s'utilise quand on a qu'une seule variable à tester. Pour faire des opérations conditionnelles sur un vecteur, on peut simplement utiliser les fonctions `<` ou `==`.

Boucles while

```
while condition
  lignes codes
end
```

Fonctions

```
function variablesortie=nomfonction(variables d'entrée)
  lignes de code (dont variablesortie= )
endfunction
```

Exemple :

```
function y=f(x);
    y=sin(x);
endfunction
```