Feuille de travaux pratiques # 2
Remise en forme Scilab

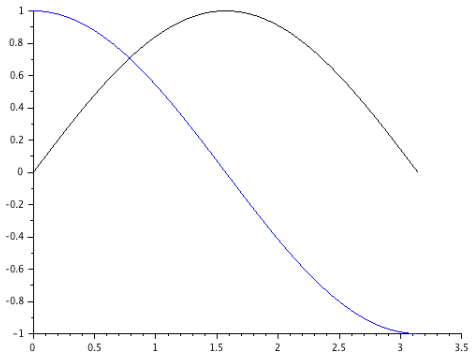
1 Illustrations graphiques avec Scilab

1.1 Tracés en dimension deux

La commande de base pour tracer des courbes planes avec Scilab est `plot2d`. Les variantes `plot2d2` et `plot2d3` permettent de tracer des courbes en escalier ou des diagrammes en bâtons. La commande `plot2d` prend en arguments les vecteurs des abscisses et des ordonnées d'une suite de points et représente la ligne brisée joignant ces points. De nombreux arguments facultatifs permettent de règler l'affichage (couleur, échelle, axes etc.), d'ajouter un titre ou des légendes : voir la section 1.3 ci-dessous.

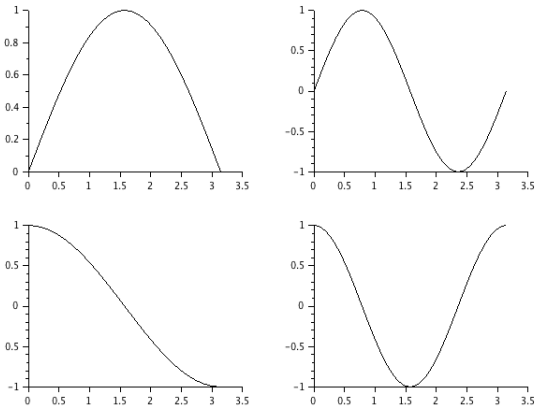
Si on veut représenter plusieurs courbes sur le même graphique, on peut simplement itérer la commande `plot2d` ou alors lui donner les différentes courbes en argument, par exemple :

```
x=linspace(0,%pi);
plot2d([x',x'],[sin(x'),cos(x')])
```



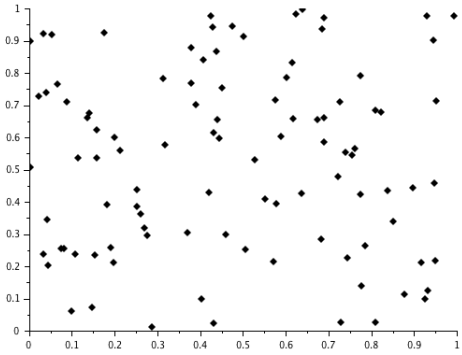
Au contraire, si l'on ne veut pas que les différentes courbes apparaissent dans la même fenêtre, on peut utiliser la commande `clf()` qui efface la fenêtre graphique courante, ou la commande `figure` qui ouvre une nouvelle fenêtre. Si l'on veut représenter plusieurs graphiques distincts dans la même fenêtre graphique, la commande dédiée est la commande `subplot` dont voici un exemple et la miniature du résultat :

```
x=linspace(0,%pi);
subplot(2,2,1)
plot2d(x,sin(x))
subplot(2,2,2)
plot2d(x,sin(2*x))
subplot(2,2,3)
plot2d(x,cos(x))
subplot(2,2,4)
plot2d(x,cos(2*x))
```



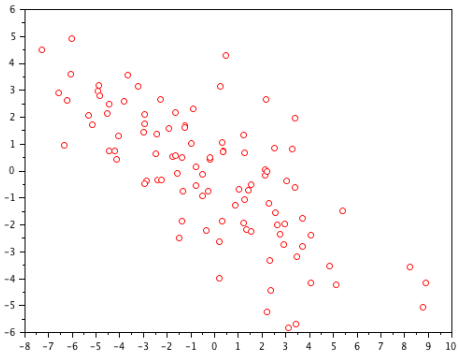
Pour représenter un nuage de points ou une courbe dont les points ne sont pas reliés entre eux avec la commande `plot2d`, il faut spécifier `style=-n` dans les options, où `n` est un entier entre 1 et 10, qui représente la marque utilisée pour représenter chaque point.

```
x=rand(2,100) ;
plot2d(x(1,:),x(2,:),style=-4)
```



Il existe aussi une commande `plot`, dont la syntaxe est plus souple. Elle permet de représenter directement le graphe d'une fonction en donnant son nom, ou de choisir style et couleur de représentation par des paramètres simples. Par exemple, ci-dessous on représente le nuage de points dont les coordonnées sont données dans la matrice xy, chaque point étant représenté par un petit cercle (`'o'`), de couleur rouge (`'r'`).

```
xy=[3 -1;-1 2]*rand(2,100,'n');
plot(xy(1,:),xy(2,:),'or')
```



Exercice 1 Télécharger le fichier donnees2.xls dans votre espace de travail. Importer le tableau dans Scilab, et tracer la première colonne en fonction de la seconde.

Exercice 2 Télécharger le fichier donnees3.xls dans votre espace de travail. Importer le document dans Scilab. Extraire la deuxième feuille excel du document, et tracer le nuage de points dont les abscisses et ordonnées sont stockées dans le tableau. Les points seront matérialisés par des triangles.

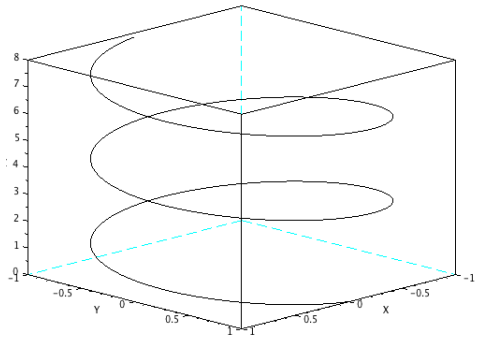
Exercice 3 Dans une même fenêtre graphique, tracer les unes en dessous des autres les trois fonctions trigonométriques usuelles (sin, cos et tan) sur une période et en regard, sur la droite, leur fonctions réciproques (là où elles sont définies !).

Exercice 4 Générer une fonction qui prend en entrée un entier n et qui en sortie trace la fonction étagée binomiale correspondante i.e. la fonction qui vaut C_n^k sur l'intervalle $[k, k+1]$, pour k variant de 0 à $n-1$.

1.2 Tracés en dimension trois

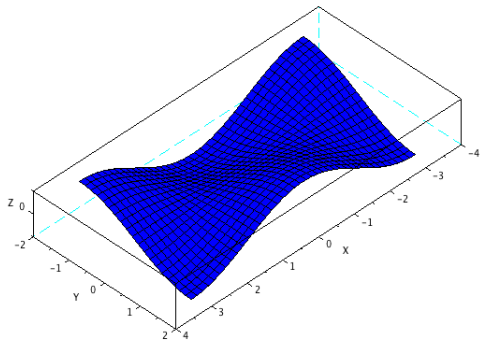
La commande pour tracer une courbe paramètrée en dimension trois est la commande `param3d` ou encore `param3d1`. Elle prend en entrée trois vecteurs ou matrices de même taille représentant les valeurs des projections de la courbe suivant les 3 axes.

```
t=0:0.1:5*%pi;
param3d(sin(t),cos(t),t/2)
```



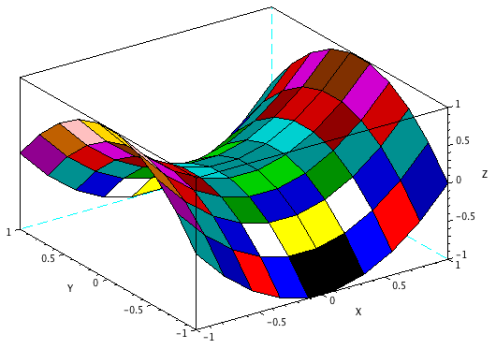
La commande de base pour représenter une surface est `plot3d` . Dans sa version la plus simple, la surface d'équation $z = f(x, y)$ est obtenue en entrant les intervalles de définition discrétisés de x et y, de longueurs respectives n et m, et une matrice z à n lignes et m colonnes donnant les valeurs de la fonction en les points de la grille ainsi définie.

```
x=linspace(-%pi,%pi,40);
y=linspace(-%pi/2,%pi/2,20);
z=cos(x)'*sin(y);
plot3d(x,y,z);
```



Une variante est donnée par la commande `surf` via la grille d'interpolation `meshgrid` :

```
X=linspace(-1,1,10);
Y=linspace(-1,1,10);
[x,y]=meshgrid(X,Y);
z=x.^2-y.^2;
surf(x,y,z)
```



Là encore, il existe d'innombrables options pour les couleurs, les ombrages etc. Il est fortement conseillé de se reporter à l'aide en ligne pour les exemples illustratifs en tapant `help plot3d` ou `help surf` ou encore `help surface properties`.

Exercice 5 Télécharger le fichier donnees4.xls dans votre espace de travail. Importer le document dans Scilab. Supprimer la première ligne et la première colonne du tableau et tracer la surface correspondant aux entrées du tableau.

1.3 Options graphiques

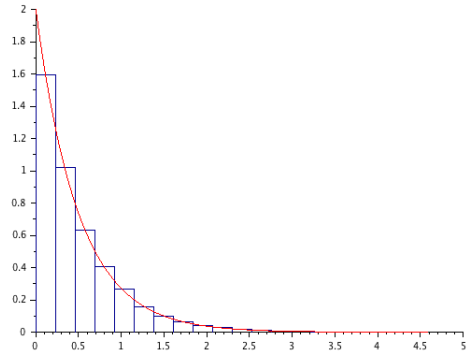
Les possibilités graphiques de Scilab sont impressionnantes et les différentes commandes graphiques comme `plot2d` ou `plot3d` admettent de très nombreuses options. Pour s'y retrouver dans la jungle des possibilités, vous pouvez consulter l'aide associée à l'item `Graphics Entities` ainsi que l'aide associée aux commandes suivantes :

`gca()`	gestion des axes sur une figure
`gcf()`	gestion de la figure elle-même
`set`	attribution d'une valeur à un paramètre d'option
`get`	obtenir la valeur d'un paramètre d'option

Lorsque vous présentez une illustration graphique au jury, vous **devez** naturellement préciser ce qu'elle représente i.e. qu'est ce qui est tracé, en fonction de quoi, quels sont les paramètres etc. Pour faciliter la compréhension et la lisibilité de vos illustrations, vous **devez** ainsi vous habituer à les "décorer" en ajoutant un titre général, un également pour chaque axe. Lorsqu'il y a plusieurs courbes, une légende permet d'afficher la correspondance entre les couleurs/motifs et les courbes représentées.

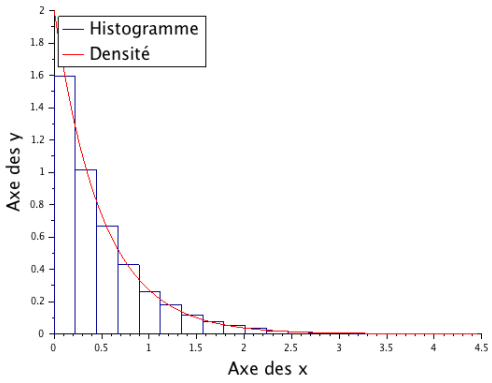
Avant

```
X=grand(1,10000,'exp',0.5)
// échantillon de variables exponentielles
clf;
histplot(20,X,9)
// histogramme associé
x=0:0.01:max(X);
plot2d(x,2*exp(-2*x),5)
// densité de la loi associée
```



Après

```
X=grand(1,10000,'exp',0.5)
// échantillon de variables exponentielles
clf;
histplot(20,X,9)
// histogramme associé
x=0:0.01:max(X);
plot2d(x,2*exp(-2*x),5)
title('Adéquation échantillon/densité',
'fontsize',4) ;
xlabel('Axe des x','fontsize',4)
ylabel('Axe des y','fontsize',4)
legends(['Histogramme';'Densité'],[9,5],
opt=2, font_size=4 )
```



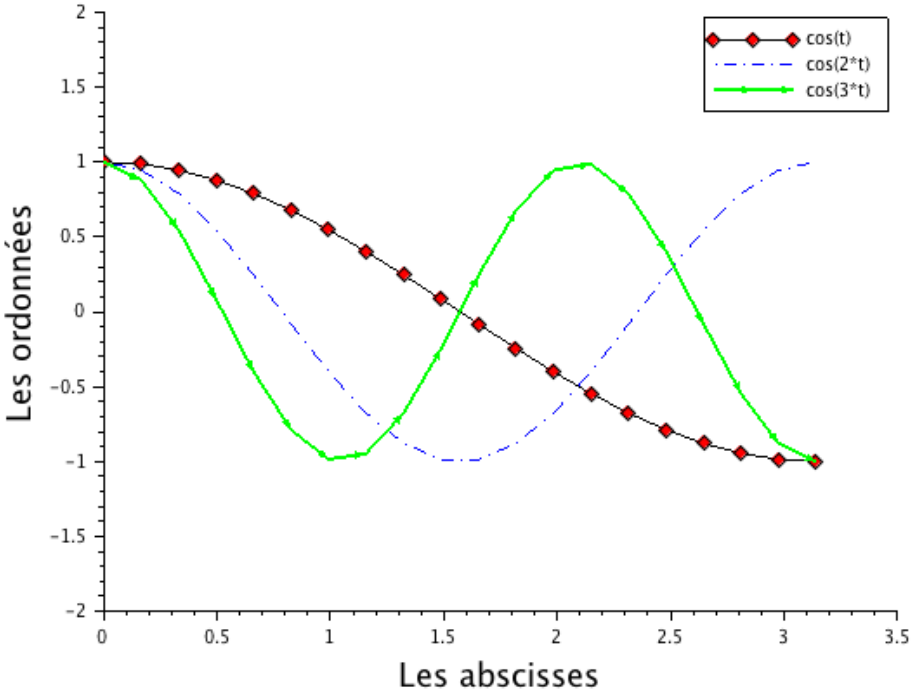
Autre exemple issu de l'aide en ligne en utilisant les commandes `gca` et `gce` :

```
t=linspace(0,%pi,20);
a=gca();
a.data_bounds=[t(1) -1.8;t($) 1.8];

plot2d(t,[cos(t'),cos(2*t'),cos(3*t')],[-5,2 3]);
xlabel('Les abscisses','fontsize',4)
ylabel('Les ordonnées','fontsize',4)
e=gce();
e1=e.children(1);
e1.thickness=2;
e1.polyline_style=4;
e1.arrow_size_factor = 1/2;
e.children(2).line_style=4;
e3=e.children(3);
e3.line_mode='on';
e3.mark_background=5;

hl=legend(['cos(t)';'cos(2*t)';'cos(3*t)']);
```

et le résultat :



2 Gestion des variables aléatoires

Dans cette section, on revient rapidement sur les commandes implémentées/implémentables dans Scilab permettant de générer/manipuler des variables aléatoires.

2.1 Les générateurs d'aléa de Siclab

Il existe dans Scilab deux principaux générateurs de nombres pseudo-aléatoires : un premier générateur simple basé sur les congruences arithmétiques permettant de générer des variables de loi uniforme et gaussienne, et un second générateur beaucoup plus complet et sophistiqué permettant de générer la plupart des lois usuelles, des chaînes de Markov etc.

2.1.1 Le générateur `rand`

La commande `x=rand(n,m)` renvoie une matrice `x` de taille $n \times m$ dont les coefficients sont des nombres pseudo-aléatoires indépendants uniformément distribués sur l'intervalle $[0, 1]$. Si A est une matrice de taille $n \times m$, alors `rand(A)` équivaut à `rand(n,m)`. On peut également obtenir un échantillon de variables de loi normale centrée réduite via la commande `x=rand(n,m,'n')`. La commande `rand('n')` permet de modifier le comportement par défaut du générateur `rand` de manière à obtenir systématiquement des nombres suivant cette loi normale.

Dans le cas uniforme, `rand` est un générateur de type "congruentiel linéaire" de paramètres $a = 843314861$, $c = 453816693$, $m = 2^{31}$, i.e. la suite x_n des nombres renvoyés par la commande `rand` est de la forme $x_n = u_n/m$, où la suite d'entiers u_n satisfait à la relation de récurrence $u_{n+1} = au_n + c \mod m$, et $u_0 = 0$ par défaut au démarage de Scilab. On peut obtenir la valeur courante de u_n par la commande `rand('seed')` et changer sa valeur par la commande `rand('seed',k)`. Cette commande est utile pour répéter à l'identique un tirage aléatoire et pouvoir ainsi contrôler ses résultats.

2.1.2 Le générateur `grand`

À préférer à `rand`, la commande `grand` permet de simuler directement la plupart des lois usuelles. Elle fait appel à différents générateurs très performants, ce qui constitue une raison supplémentaire de l'utiliser. Elle permet en outre de simuler l'évolution de chaînes de Markov à espace d'états fini, ainsi que de générer des permutations aléatoires.

Exercice 6 À l'aide de la fonction `grand`, simuler une matrice $(M_{ij})_{i=1...100,j=1...200}$ contenant 100 échantillons de taille 200 de variables aléatoires de loi exponentielle de paramètre 1. Pour illustrer le théorère limite central relatif aux sommes $\sum_j M_{i,j}$, après centrage et renormalisation, tracer l'histogramme associé et sur le même graphique la densité gaussienne correspondante.

Exercice 7 À l'aide de la fonction `grand`, simuler les 100 premiers pas d'une chaîne de Markov dont la matrice de transition est donnée par

$$P = \begin{pmatrix} 1/3 & 1/3 & 1/6 & 1/6 \\ 0 & 1/2 & 0 & 1/2 \\ 1/4 & 1/4 & 1/4 & 1/4 \\ 1/3 & 1/3 & 1/3 & 0 \end{pmatrix}$$

Estimer la probabilité invariante à partir des temps d'occupation empiriques.

2.2 Simulation par fonction de répartition inverse

On ne peut faire appel au générateur `grand` pour simuler une variable aléatoire réelle dont la loi n'est pas une loi classique. Cependant, si l'on connaît la fonction de répartition de loi cible, il est toujours possible de l'inverser pour arriver à ses fins.

Proposition 1 *Soit X une variable aléatoire réelle de fonction de répartition F. Pour $u \in [0,1]$, on désigne par $F^{-1}(u) := \inf\{x \in \mathbb{R},\, F(x) \geq u\}$ l'inverse généralisée de la fonction de répartition F. Si $U \sim \mathcal{U}_{[0,1]}$ alors $F^{-1}(U)$ a pour fonction de répartition F.*

Exercice 8 Simuler les variables aléatoires décrites ci-dessous sans faire appel à `grand` :

1. À partir de variables uniformes sur $[0,1]$, générer un $n-$échantillon de variables de Bernoulli de paramètre $1/2$ à valeurs dans $E = \{0,1\}$. Même question lorsque $E = \{-1,1\}$.
2. Simuler une variable aléatoire Y de loi uniforme à valeurs dans $\{0,1,2,3\}$.
3. À partir de variables uniformes, simuler un $n-$échantillon $(X_1, \ldots, X_n)$ de variables aléatoires de loi exponentielle de paramètre 2. Mettre en évidence la loi des grands nombres en traçant la fonction $k \mapsto (X_1 + \ldots + X_k)/k$ pour k allant de 1 à n.

2.3 Simulation par rejet

Proposition 2 *Soient A, $D \in \mathcal{B}(\mathbb{R}^d)$ tels que $A \subset D$. Soit $(X_i)_{i\in\mathbb{N}}$ une suite i.i.d. de variables aléatoires définies sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$ uniformes sur D. Introduisons le temps $\tau := \inf\{i \in \mathbb{N}^*,\, X_i \in A\}$, alors X_τ est uniformément distribuée dans A, i.e. pour $B \in \mathcal{B}(\mathbb{R}^d)$, $B \subset A$, $\mathbb{P}(X_\tau \in B) = |B|/|A|$. Le nombre de tirages avant l'obtention d'une réalisation de la loi uniforme sur A par ce procédé suit une loi géométrique $\mathcal{G}(p)$, avec $p = |A|/|D|$. En particulier le nombre moyen de tirages nécessaires est $|D|/|A|$.*

De la proposition, on déduit la méthode de simulation suivante. Soit X une variable aléatoire réelle de loi μ_X possèdant une densité continue f_X à support compact inclus dans $[a,b] \subset \mathbb{R}$ et telle que son graphe soit inclus dans $[a,b] \times [0,M]$, pour un certain $M \in \mathbb{R}^+$. On considère une suite $(Z_i)_{i\in\mathbb{N}} = (X_i, Y_i)_{i\in\mathbb{N}}$ de variables aléatoires uniformes dans $[a,b] \times [0,M]$ et l'on définit $\tau = \inf\{i \in \mathbb{N},\, f(X_i) > Y_i\}$. Alors X_τ a pour loi μ_X.

Application : utiliser la méthode de simulation ci-dessus pour générer une variable aléatoire de densité $x \mapsto \frac{\pi}{2}\sin(\pi x)$ sur l'intervalle $[0,1]$.

On peut généraliser la méthode de rejet de la façon suivante. Soit X une variable aléatoire réelle de loi μ_X qui possède une densité continue f_X majorée uniformément par une densité g, i.e. il existe une constante $C \geq 1$ telle que

$$\forall x \in \mathbb{R},\, f_X(x) \leq Cg(x), \quad \text{avec} \quad \int g(y)dy = 1.$$

On suppose que l'on sait facilement simuler des variables aléatoires de loi de densité g. Soient donc $(X_i)_{i\in\mathbb{N}}$ une suite i.i.d. de variables aléatoires de loi de densité g et $(U_i)_{i\in\mathbb{N}}$ une suite i.i.d. de variables aléatoires uniformes sur l'intervalle $[0,1]$, indépendantes de $(X_i)_{i\in\kappa}$. Si l'on considère le temps $\tau := \inf\{i \geq 1,\, f(X_i) > Cg(X_i)U_i\}$, alors X_τ a pour loi μ_X.

Exercice 9 On souhaite simuler une variable aléatoire X de loi $\mathcal{N}(0,1)$. On montre sans trop de difficulté la majoration suivante :

$$\forall x \in \mathbb{R},\quad \underbrace{\left(\frac{e^{-x^2/2}}{\sqrt{2\pi}}\right)}_{:=f_X(x)} \leq \underbrace{\sqrt{\frac{2e}{\pi}}}_{:=C} \times \underbrace{\left(\frac{1}{2}e^{-|x|}\right)}_{:=g(x)}.$$

1. Montrer que si ε suit une loi de Bernoulli de paramètre $1/2$ dans $\{-1,1\}$ et si Z est une variable exponentielle de paramètre 1 indépendante de ε, alors εZ a pour densité la fonction g sur $\mathbb{R}$;
2. Implémenter un algorithme qui simule une variable gaussienne via la méthode de rejet ;
3. Simuler 1000 variables selon cet algorithme et vérifier que l'histogramme correspondant approche bien la densité gaussienne.

2.4 Simulation de loi conditionnelle

Si A est un événement de probabilité strictement positive associé à une certaine expérience aléatoire et si on veut simuler sous la probabilité conditionnelle $\mathbb{P}(\cdot|A)$ une variable aléatoire dépendant de cette expérience, on effectue une suite de tirages successifs sous la probabilité initiale $\mathbb{P}$ et on conserve uniquement les tirages pour lesquels l'événement A est réalisé.

Exercice 10 Soit X une variable aléatoire réelle de loi de Pareto, admettant la densité

$$f_X(x) = 2/x^3, \quad \text{sur} \quad [1,+\infty[.$$

La fonction de répartition $F_X(x) = 1 - x^{-2}$ s'inverse facilement de sorte que X a même loi que $(1-U)^{-1/2}$ ou encore $U^{-1/2}$ où U suit la loi uniforme sur $[0,1]$. Une propriété remarquable d'une telle variable est que, pour tout réel $c \geq 1$, on a $\mathbb{E}[X|X \geq c] = 2c$. Vérifier empiriquement cette propriété en simulant un échantillon de grande taille de cette loi et en ne conservant que les valeurs supérieures à c.

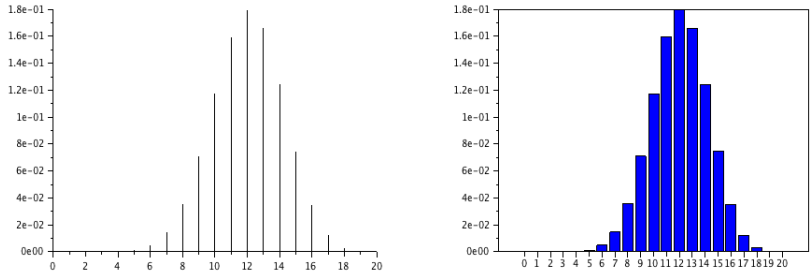
3 Représentation des variables aléatoires

Dans cette dernière section, nous rappelons les commandes utiles pour représenter graphiquement les variables aléatoires générées avec Scilab.

3.1 Lois discrètes

Une loi discrète sur une partie finie de $\mathbb{R}$ peut se représenter par un diagramme en bâtons ou par un diagramme en barres. On dispose pour cela des commande `plot2d3` et `bar` :

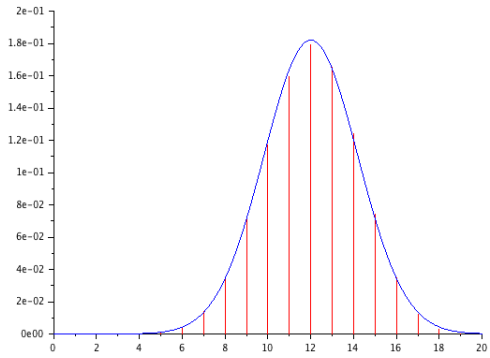
```
x=binomial(0.6,20);
subplot(121);
plot2d3(0:20,x);
subplot(122);
bar(0:20,x)
```



3.2 Lois continues

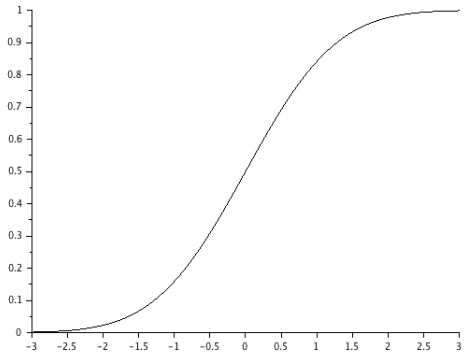
Pour représenter graphiquement la densité d'une loi à densité sur $\mathbb{R}$, il suffit d'utiliser la commande `plot2d`. On peut ainsi par exemple illustrer la proximité d'une loi binomiale à la loi normale correspondante :

```
x=binomial(0.6,20);
plot2d3(0:20,x,style=5);
x1=linspace(0,20);m=12;v=4.8;
plot2d(x1,exp(-(x1-m).^2/2/v)/sqrt(2*%pi*v),
style=2);
```



Pour représenter la fonction de répartition d'une loi à densité, on peut intégrer numériquement cette densité via la fonction `integrate`. Mais la plupart des fonctions de répartition classiques sont déjà implémentées dans Scilab. Les commandes associées commencent toutes par `cdf` (Cumulative Distribution Function), par exemple `cdfbet` pour les lois beta, `cdfnor` pour les lois normales etc. Le premier argument en entrée est la chaîne de caractères `'PQ'` si on veut évaluer la fonction de répartition elle-même. On peut ainsi tracer le graphe de la fonction de répartition de la loi normale réduite comme suit :

```
x=linspace(-3,3);
y=cdfnor('PQ',x,zeros(x),ones(x));
plot2d(x,y)
```



Les commandes `cdf` permettent également d'inverser les fonctions de répartition et ainsi de déterminer les quantiles des lois usuelles. Il faut pour cela donner comme premier argument en entrée la chaîne de caractères `'X'`. Par exemple, si X suit une loi normale centrée réduite, pour déterminer le réel x tel que $\mathbb{P}(|X| > x) = \alpha$, on écrit

```
alpha=0.05;
x=cdfnor('X',0,1,1-alpha/2,alpha/2)
```

Pour une loi du χ^2 à 7 degrés de liberté :

```
alpha=0.05;
x=cdfchi('X',7,alpha,1-alpha)
```

Ces commandes seront particulièrement utiles lorsque l'on voudra représenter des intervalles de confiances lors de l'estimation de paramètres des modèles.

3.3 Lois empiriques

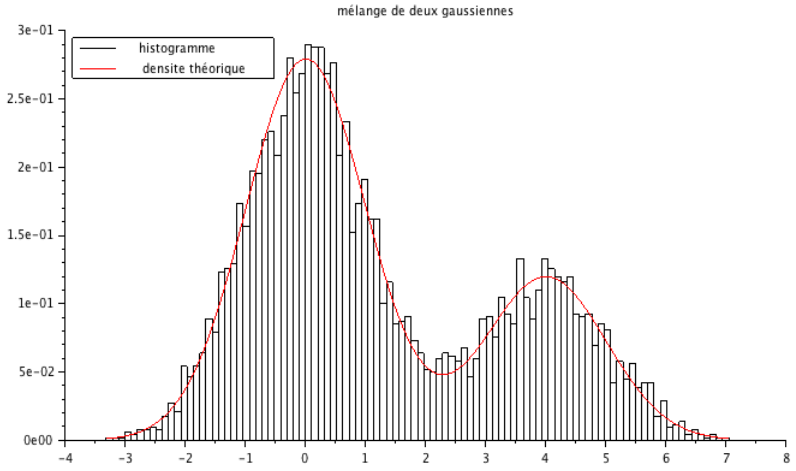
Pour représenter les lois empiriques associées à une expérience aléatoire, on peut au choix tracer l'histogramme empirique associé aux réalisations, ou encore la fonction de répartition empirique.

3.3.1 Histogramme empirique

L'histogramme d'un échantillon est un diagramme constitué de barres verticales juxtaposées, chacune de ces barres représentant le nombre de termes de l'échantillon appartenant à une classe donnée. Pour représenter un histogramme d'un échantillon d'une loi réelle, on commence donc par répartir cet échantillon en classes, chacune de ces classes correspondant aux valeurs appartenant à un certain intervalle de $\mathbb{R}$, et on représente cette classe par un rectangle vertical dont l'aire est proportionnelle à l'effectif de la classe. On normalise souvent ces aires de façon à ce que l'aire totale soit égale à 1. La commande `histplot` permet de tracer directement un tel histogramme. Son premier argument est soit un entier (nombre de classes) soit un vecteur dont les composantes donnent les limites des classes. Son second argument est le vecteur des données ; on peut aussi ajouter des arguments de style comme dans `plot2d`.

```
// melange de lois normales
N=5000; p=0.7;
X=grand(1,N,'bin',1,p);
Y1=grand(1,N,'nor',0,1);
Y2=grand(1,N,'nor',4,1);
Z=X.*Y1 +(1-X).*Y2;
histplot(100,Z);
function y=f(x) // la densité du mélange
y=(p/sqrt(2*%pi))*exp(-x.^2/2)+((1-p)/sqrt(2*%pi))*exp(-(x-4).^2/2);
endfunction;
u=linspace(min(Z),max(Z),200);
plot2d(u,f(u),5,"000");
legends(['histogramme';'densite théorique'],[1,5],2);
xtitle('mélange de deux gaussiennes');
```

Le résultat :



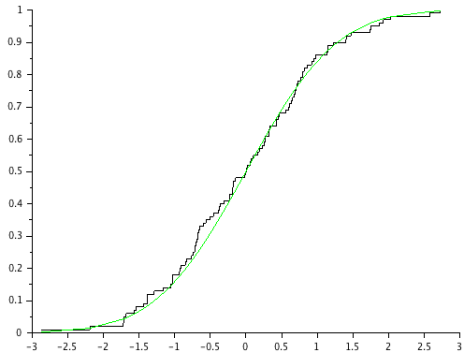
3.3.2 Fonction de répartition empirique

La fonction de répartion empirique d'un n-échantillon $(X_1, \ldots, X_n)$ est la fonction croissante $F_n : \mathbb{R} \to [0, 1]$ définie par

$$F_n(y) = \frac{1}{n} \sum_{k=1}^{n} \mathbb{1}_{X_k \leq y}$$

Si on dispose d'un échantillon de taille n, on peut tracer le graphe de la fonction de répartition empirique de cet échantillon en ordonnant préalablement cet échantillon puis en utilisant la fonction `plot2d2`. L'exemple suivant tire un échantillon de taille 100 d'une loi normale réduite, trace le graphe de sa fonction de répartition empirique et la compare à la fonction de répartition de la loi normale

```
n=100;
x=rand(1,n,'n');
xx=gsort(x,'g','i');
plot2d2(xx,(1:n)/n);
y=cdfnor('PQ',xx,zeros(xx),ones(xx));
plot2d(xx,y,style=3)
```



3.4 Quelques exercices pour s'entraîner

Exercice 11 La loi de Fréchet de paramètres $a > 0$ et $\alpha > 0$ est la loi sur $[0, +\infty[$ de fonction de répartition $F_X(x) = \exp(-(a/x)^\alpha)$. Simuler par inversion un échantillon de grande taille de cette loi. Représenter sur un même graphique la fonction de répartition empirique de cet échantillon et la fonction de répartition théorique de la loi.

Exercice 12 On rappelle que la distance en variation totale entre deux lois de probabilité μ et ν supportées par un ensemble au plus dénombrable E est définie par

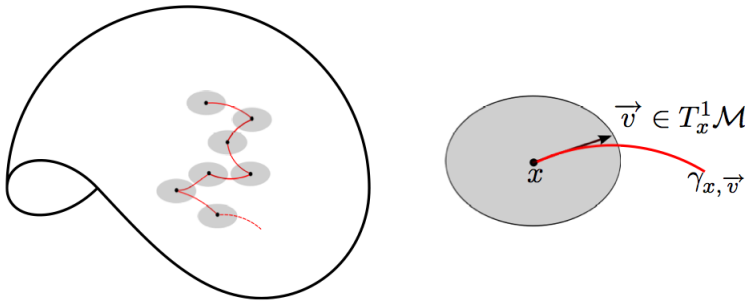
$$\mathrm{d}_{TV}(\mu, \nu) = \sum_{x \in E} |\mu(\{x\}) - \nu(\{x\})|.$$

Evaluez numériquement la distance en variation totale entre la loi de Poisson de paramètre λ et la loi binomiale de paramètres n et $p = \lambda/n$ pour différentes valeurs de n. Représentez graphiquement ces deux lois sur une même figure, puis sur deux figures juxtaposées.

Exercice 13 Estimer par simulation la probabilité $p(n)$ que le déterminant d'une matrice aléatoire A de taille n dont les coefficients sont des variables aléatoires i.i.d. de loi uniforme sur $\{-1, 1\}$, soit nul. On estimera $p(n)$ pour n variant de 2 à 20 et on tracera la courbe représentative de $p(n)$ en fonction de n. Estimer par simulation l'espérance de $\det(A)^2$ pour $n = 2 \ldots 5$. Énoncer, puis démontrer une conjecture sur l'espérance de ce déterminant.

4 Pour celles et ceux qui sont très à l'aise avec Scilab

Exercice 14 On se propose de simuler le mouvement brownien sur la sphère $\mathbb{S}^2 \subset \mathbb{R}^3$, que l'on munira de la métrique euclidienne induite. Comme le mouvement brownien euclidien est la limite d'échelle "universelle" de marches aléatoires symétriques, on peut montrer que le mouvement brownien sphérique peut être obtenu comme la limite d'échelle d'une marche aléatoire géodésique par morceaux, marche illustrée ci-dessous et que l'on peut décrire comme suit :



On fixe ainsi un paramètre $\varepsilon > 0$ (le pas de notre marche) et on note X_n la position de la marche au temps n.

- Si $X_n = x \in \mathbb{S}^2$,
- on choisit une direction v_x uniformément dans $T^1_x\mathbb{S}^2 = \{v \in T_x\mathbb{S}^2, ||v|| = 1\}$,
- sur une distance $\varepsilon > 0$, on suit la géodésique issue de x et dirigée selon v_x, autrement dit, on suit le grand cercle passant par x tangent au vecteur v_x,
- on arrive ainsi à nouveau point $y \in \mathbb{S}^2$, on pose $X_{n+1} = y$ et on recommence...

1. Implémenter un algorithme qui, étant donné X_n, fournit une réalisation de X_{n+1}.
2. Représenter la marche obtenue sur la sphère de dimension 2.
3. Ensuivant la même démarche, simuler le mouvement brownien sur le tore plat $\mathbb{T}^2 \subset \mathbb{R}^3$.