Année 2021-2022

Simulation Monte Carlo

Madalina Deaconu

Institut Élie Cartan & Inria Nancy - Grand Est
Université de Lorraine
Email : madalina.deaconu@univ-lorraine.fr

Table des matières

Chapitre 1

Introduction

L'objectif de ce cours est d'introduire les outils théoriques et d'illustrer, à travers des problèmes réels, comment la modélisation stochastique et les méthodes de Monte Carlo permettent d'apporter des réponses pour des questions issues de nombreux domaines applicatifs.

La modélisation probabiliste est fondamentale dans tous les domaines d'application.

Les outils probabilistes que nous développons, comme les méthodes de Monte Carlo et le mouvement brownien, sont des outils généraux utilisés dans de nombreux domaines comme en physique (écoulement d'un fluide, trajectoire d'un avion), en biologie (mutation du genôme), en chimie (coagulation des polymères), en climatologie (modélisation du vent, de la pluie etc), en informatique, en médicine, en finance (évaluation des produits dérivés), en science du vivant, en assurance, en linguistique, en sociologie... Les modèles que nous étudions sont inspirées du monde de la finance, des files d'attente, etc.

Nous débuterons ce cours par des rappels sur les méthodes numériques permettant de simuler des variables aléatoires.

Ensuite nous étudions des méthodes numériques basées sur les tirages successifs de nombres aléatoires, méthodes appelées généralement méthodes de Monte Carlo.

En outre, nous aborderons l'étude de certains processus stochastiques essentiels dans la modélisation, tels que le mouvement brownien. Ce processus est fondamental en finance.

Chapitre 2

Introduction

L'objectif de ce cours est d'introduire les outils théoriques et d'illustrer, à travers des problèmes réels, comment la modélisation stochastique et les méthodes de Monte Carlo permettent d'apporter des réponses pour des questions issues de nombreux domaines applicatifs.

La modélisation probabiliste est fondamentale dans tous les domaines d'application.

Les outils probabilistes que nous développons, comme les méthodes de Monte Carlo et le mouvement brownien, sont des outils généraux utilisés dans de nombreux domaines comme en physique (écoulement d'un fluide, trajectoire d'un avion), en biologie (mutation du genôme), en chimie (coagulation des polymères), en climatologie (modélisation du vent, de la pluie etc), en informatique, en médicine, en finance (évaluation des produits dérivés), en science du vivant, en assurance, en linguistique, en sociologie... Les modèles que nous étudions sont inspirées du monde de la finance, des files d'attente, etc.

Nous débuterons ce cours par des rappels sur les méthodes numériques permettant de simuler des variables aléatoires.

Ensuite nous étudions des méthodes numériques basées sur les tirages successifs de nombres aléatoires, méthodes appelées généralement méthodes de Monte Carlo.

En outre, nous aborderons l'étude de certains processus stochastiques essentiels dans la modélisation, tels que le mouvement brownien. Ce processus est fondamental en finance.

Chapitre 3

Simulation de nombres aléatoires

Nous allons voir dans ce qui suit que, à la base de toute méthode de simulation d'une variable aléatoire, se trouve la loi uniforme. En outre, nous montrerons comment on peut construire des générateurs de nombres aléatoires.

3.1 Simulation d'une loi uniforme sur $[0, 1]$

Dans tous les logiciels classiques dédiés au calcul numérique, il existe de très bons générateurs de nombres aléatoires.

Nous présentons leur principe général de fonctionnement.

La méthode la plus couramment utilisée est la méthode des congruences linéaires.

On simule un nombre aléatoire sur $[0, 1]$ comme suit :

On construit une suite $(x_n)_{n\geq 0}$ de nombres entiers compris entre 0 et $m-1$ de la façon suivante :

1. On choisit un nombre entier quelconque, appelé valeur initiale

$$x_0 = \text{ valeur initiale.} \tag{3.1}$$

2. On construit la suite x_n par récurrence

$$x_{n+1} = ax_n + b \,(\text{modulo } m), \tag{3.2}$$

avec a, b, m des entiers positifs, qu'il faudra choisir soigneusement, si l'on veut que les caractéristiques de la suite soient performantes.

Sedgewick préconise le choix suivant :

$$\begin{cases} a = 31415821 \\ b = 1 \\ m = 10^8. \end{cases}$$

Cette méthode permet de simuler des entiers pseudo aléatoires entre 0 et $m-1$.

3. Pour obtenir un nombre aléatoire entre 0 et 1 on divise le résultat, cet entier aléatoire ainsi généré, par m

$$u = \frac{x_n}{m}. \tag{3.3}$$

On dit que de tels nombres sont (pseudo-)aléatoires pusiqu'ils sont obtenus par une règle arithmétique. Mais la grande périodicité de m et un choix judicieux de a et b permettent d'affirmer (cf. Knuth) que leur loi de distribution est "indistinguable presque sûrement" de vrais nombres aléatoires.

Le générateur précédent fournit des résultats acceptables dans les cas courants. Cependant, sa période (ici $m = 10^8$) peut se réveler parfois insuffisante.

On peut alors, obtenir des générateurs de nombres aléatoires de période arbitrairement longue en augmentant m.

Chapitre 4

Simulation de variables aléatoires

4.1 Introduction

On présentera dans cette partie un panel de méthodes pour la simulation de variables aléatoires. Nous commençons par rappeler les principes de simulation pour les lois classiques. Ensuite nous introduisons les méthodes générales de simulation pour les variables aléatoires à valeurs discrètes et continues.

Soit $(\Omega, \mathcal{F}, \mathbb{P})$ un espace de probabilité et X une variable aléatoire réelle sur cet espace. On notera F sa fonction de répartition et f sa densité. Plus précisément, pour tout $x \in \mathbb{R}$:

$$F(x) = \mathbb{P}(X \leq x)$$

et

$$f(x) = \frac{\mathrm{d}}{\mathrm{d}x} F(x).$$

Principe *Pour simuler des variables aléatoires de loi quelconque, l'idée est de se ramener à la loi uniforme sur* $[0,1]$. *La loi uniforme sur* $[0,1]$ *est donc à la base de toute simulation de variable aléatoire. Rappelons ses propriétés.*

Soit U une variable aléatoire de loi uniforme sur $[0,1]$, $U \sim \mathcal{U}[0,1]$. Dans ce cas :

$$F_U(x) = \begin{cases} 0 & \text{si} \quad x < 0 \\ x & \text{si} \quad 0 \leq x \leq 1 \\ 1 & \text{si} \quad x > 1 \end{cases}$$

et

$$f_U(x) = \mathbb{1}_{\{x \in [0,1]\}} = \begin{cases} 1 & \text{si } x \in [0,1] \\ 0 & \text{si } x \notin [0,1]. \end{cases}$$

Cadre général : On suppose qu'on dispose d'un générateur de variables aléatoires de loi uniforme sur $[0,1]$ indépendantes.

L'hypothèse d'indépendance des valeurs est une des conditions essentielles pour la validité de la plupart des algorithmes présentés dans la suite.

Exemple 1. MATLAB possède une fonction `rand()` dont les appels successifs fournissent une suite de variables aléatoires "indépendantes" et identiquement distribuées, de loi uniforme sur $[0,1]$. Nous ne nous intéresserons pas ici à la conception d'une telle fonction.

Les générateurs sont souvent construits à partir d'une relation de congruence sur de nombres de grande dimension et initialisés par exemple à partir de l'horloge de la machine. Il s'agit des générateurs de nombres pseudo-aléatoires, notamment les valeurs obtenues ne sont qu'apparemment indépendantes.

Remarque 1. *Une variable aléatoire U de loi uniforme sur $[0,1]$ est différente de 0 et 1 presque surement. En particulier, si $\Lambda(dx)$ note la mesure de Lebesgue, on a*

$$\mathbb{P}(U \in \{0,1\}) = \int_{\{0,1\}} \mathbb{1}_{[0,1]} d\Lambda = \Lambda(\{0,1\}) = 0.$$

Remarque 2. *En* MATLAB *la fonction* rand *ne renvoie que des valeurs différentes de 0 et 1.*

Objectif du cours : Vous apprendre à construire des méthodes pour simuler une variable aléatoire ou un vecteur aléatoire suivant une loi donnée, différents de la loi uniforme sur $[0,1]$.

4.2 Simulation de lois classiques

4.2.1 Loi de Bernoulli de paramètre p

Soit $p \in [0,1]$. On veut simuler une variable aléatoire $X \sim \mathcal{B}(p)$ de loi de probabilité

$$\mu(x) = p\delta_1(x) + (1-p)\delta_0(x). \tag{4.1}$$

On peut exprimer cela aussi sous la forme X est une variable aléatoire à valeurs dans $\{0,1\}$ et sa loi est donnée par $p_0 = \mathbb{P}(X=1) = p$ et $p_1 = \mathbb{P}(X=0) = 1-p$. On remarque que $p_0 + p_1 = 1$.

Algorithme Pour simuler X on tire U une uniforme sur $[0,1]$ et on définit

$$X = \begin{cases} 1 & \text{si } U \leq p \\ 0 & \text{sinon,} \end{cases}$$

c'est-à-dire $X = \mathbb{1}_{\{U \leq p\}}$.

On propose deux fonctions pour simuler cette variable aléatoire. La première ne rend qu'une réalisation d'une variable aléatoire de la loi de Bernoulli de paramètre p, la seconde renvoie un k-échantillon, qui sera utilisé par la suite dans la simulation de la loi binomiale :

Algorithmes Matlab

```
 function x=bernoulli1(p)
% tirage suivant la loi de Bernoulli de parametre p
  x=0;
if rand()<p
  x=1;
end

function x=bernoulli2(k,p)
% tirage d'un k-echantillon suivant la loi de Bernoulli de parametre p
  y=rand(1,k),
x=(y<p);
end
```

Exemple 2. Pile ou face

On veut simuler une variable aléatoire de loi "Pile ou face" i.e. $X \sim \mathcal{B}(\frac{1}{2})$. On tire U une uniforme sur $[0,1]$ et on définit

$$X = \begin{cases} \text{"Pile"} & \text{si } U \leq \dfrac{1}{2} \\ \text{"Face"} & \text{sinon,} \end{cases}$$

formellement on peut aussi écrire $X = \mathbb{1}_{\{U \leq \frac{1}{2}\}}$.

4.2.2 Loi binomiale de paramètres (n, p)

Soient $n \in \mathbb{N}^*$ et $p \in [0,1]$. On veut simuler $X \sim \mathcal{B}(n,p)$ donc de loi

$$\mu(x) = \sum_{k=0}^{n} \binom{n}{k} p^k (1-p)^{n-k} \delta_k(x). \tag{4.2}$$

Donc X est une variable aléatoire à valeurs dans $\{0, \ldots, n\}$ de loi donnée par les probabilités $p_k = \mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$,pour tout $k \in \{0, \ldots, n\}$.
On sait qu'une variable aléatoire binomiale peut être représentée comme la somme de n variables aléatoires indépendantes de Bernoulli de paramètre p. Plus précisément on a le résultat suivant :

Lemme 1. *Soient $(Y_i)_{1 \leq i \leq n}$ des variables aléatoires indépendantes et identiquement distribuées de loi de Bernoulli de paramètre p, alors $X = \sum_{i=1}^{n} Y_i$ suit une loi binomiale de paramètres (n,p).*

Algorithme

Il suffit donc de simuler n variables aléatoires indépendantes de loi $\mathcal{B}(p)$ et d'en faire la somme. On simule $(U_i)_{1 \leq i \leq n}$, n v.a.i.i.d. de loi uniforme sur $[0,1]$ et on définit

$$X = \sum_{i=1}^{n} \mathbb{1}_{\{U_i \leq p\}}.$$

Algorithme Matlab

Nous utilisons ici la fonction Matlab définie précédemment qui simule un n échantillon de loi de Bernoulli de paramètre p.

```
function x=binomiale(n,p)
% tirage suivant la loi binomiale de parametres (n,p)
  x=sum(bernoulli2(n,p));
```

4.2.3 Loi uniforme sur $\{0, 1, ..., n-1\}$

On veut simuler X de loi uniforme sur $\{0, 1, ..., n-1\}$, donc de loi

$$\mu(x) = \frac{1}{n} \sum_{k=0}^{n-1} \delta_k(x). \tag{4.3}$$

Méthode 1 : Nous allons utiliser le résultat suivant :

Lemme 2. *Si U est une variable aléatoire de loi uniforme sur $[0,1]$ alors la partie entière $X = [nU]$ suit la loi uniforme sur $\{0,1,...,n-1\}$.*

Remarque 3. *En admettant ce résultat, l'algorithme s'écrit :*
Algorithme Matlab

```
function x=uniforme1(n)
  % tirage suivant la loi uniforme sur {0,...,n-1}
  x=floor(n*rand());
end
```

Démonstration : On note X la variable aléatoire rendue par cette fonction.
Comme $\mathbb{P}(0 \leq \texttt{rand()} < 1) = 1$, on a $\mathbb{P}(0 \leq \texttt{n*rand()} < n) = 1$ et

$$\mathbb{P}(\texttt{floor(n*rand())} \in \{0,1,...,n-1\}) = 1.$$

Donc X prend ses valeurs dans $\{0,1,...,n-1\}$.

Maintenant, soit $k \in \{0,1,...,n-1\}$:

$$\begin{aligned}\mathbb{P}(X=k) &= \mathbb{P}(\texttt{floor(n*rand())} = k) = \mathbb{P}(k \leq \texttt{n*rand()} < k+1) \\ &= \mathbb{P}\left(\frac{k}{n} \leq \texttt{rand()} < \frac{k+1}{n}\right) = \frac{1}{n}.\end{aligned}$$

Ce qui prouve le résultat souhaité. ■

Méthode 2 : En utilisant la méthode générale de simulation d'une variable aléatoire discrète, voir 4.3.1.

4.2.4 Loi uniforme sur $[a,b]$

On souhaite simuler X une variable aléatoire de loi uniforme sur l'intervalle $[a,b]$ pour $a < b$, avec densité

$$f(x) = \frac{1}{b-a}\mathbb{1}_{[a,b]}(x) = \begin{cases} \dfrac{1}{b-a} & \text{si } x \in [a,b] \\ 0 & \text{si } x \notin [a,b]. \end{cases} \tag{4.4}$$

Lemme 3. *Si U suit la loi uniforme sur $[0,1]$, alors, si $a < b$ et a, $b \in \mathbb{R}$, la variable aléatoire $X = a + (b-a)U$ suit la loi uniforme sur $[a,b]$.*

Démonstration : Soit $\varphi : \mathbb{R} \to \mathbb{R}$ une fonction continue bornée. En faisant le changement de variable $x = a + (b-a)u$, on a

$$\mathbb{E}(\varphi(a+(b-a)U)) = \int_0^1 \varphi(a+(b-a)u)\mathrm{d}u = \int_a^b \varphi(x)\frac{1}{b-a}\mathrm{d}x,$$

ce qui signifie que X est une variable aléatoire de densité $f(x) = \frac{1}{b-a}\mathbb{1}_{[a,b]}(x)$, et on reconnaît ainsi la densité de la loi uniforme sur $[a,b]$. ■

L'algorithme associé s'écrit alors :

Algorithme Matlab :

```
function x=uniforme2(a,b)
% tirage suivant la loi uniforme sur [a,b]
  x=a+(b-a)*rand();
end
```

4.2.5 Loi exponentielle de paramètre λ

Soit $\lambda > 0$. On souhaite simuler une variable aléatoire X de loi exponentielle de paramètre λ, avec

$$f(x) = \lambda \exp(-\lambda x) \mathbb{1}_{\mathbb{R}_+}(x) = \begin{cases} \lambda \exp(-\lambda x) & \text{si } x \geq 0 \\ 0 & \text{si } x < 0. \end{cases} \tag{4.5}$$

Sa fonction de répartition est $F(x) = \mathbb{P}(X \leq x) = 1 - \exp(-\lambda x)$. Cette fonction est une bijection de $]0, +\infty[$ dans $]0, 1[$, d'inverse

$$G(u) = -\frac{1}{\lambda} \ln(1 - u). \tag{4.6}$$

Lemme 4. *Si U est de loi uniforme sur $[0, 1]$, alors $G(U)$ suit la loi exponentielle de paramètre λ.*

En utilisant ce lemme on déduit l'algorithme suivant pour la simulation de la loi exponentielle :

Algorithme Matlab :

```
function x=exponentielle(a)
% tirage suivant la loi exponentielle de parametre a
  x=- log(rand())/a;
end
```

Ce procédé annonce la méthode de simulation par inversion de la fonction de répartition, que nous présenterons plus tard.

Remarque 4. *Pourquoi a-t-on remplacé $1 - U$ par U dans l'algorithme ?*

4.2.6 Loi géométrique de paramètre p

Soit $p \in]0, 1]$. On veut simuler une variable aléatoire géométrique de paramètre p, donc de loi

$$\mu(x) = \sum_{k=1}^{+\infty} (1 - p)^{k-1} p \delta_k(x). \tag{4.7}$$

On a donc dans ce cas les probabilités $p_k = \mathbb{P}(X = k) = (1-p)^{k-1} p$ pour tout $k \geq 1$ Plusieurs méthodes sont possibles.

Méthode 1 : En utilisant la loi exponentielle :

Lemme 5. *Si U suit la loi uniforme sur $[0, 1]$, alors $X = 1 + \left[\frac{\ln U}{\ln(1-p)}\right]$ est une variable aléatoire qui suit la loi géométrique de paramètre p.*

Démonstration : Notons $X = 1 + \left[\frac{\ln U}{\ln(1-p)}\right]$. Par définition de la partie entière, X prend ses

valeurs dans $\mathbb{N}^*$. Soit maintenant $k \in \mathbb{N}^*$. Calculons :

$$\begin{aligned}
\mathbb{P}(X=k) &= \mathbb{P}\left(1+\left[\frac{\ln U}{\ln(1-p)}\right]=k\right) \\
&= \mathbb{P}\left(\left[\frac{\ln U}{\ln(1-p)}\right]=k-1\right) \\
&= \mathbb{P}\left(k-1 \leq \frac{\ln U}{\ln(1-p)} < k\right) \\
&= \mathbb{P}\left(k\ln(1-p) < \ln U \leq (k-1)\ln(1-p)\right) \\
&= \mathbb{P}\left((1-p)^k < U \leq (1-p)^{k-1}\right) \\
&= (1-p)^{k-1} - (1-p)^k \\
&= (1-p)^{k-1}p.
\end{aligned}$$

■

L'algorithme correspondant s'écrit :

Algorithme Matlab :

```
function x=geometrique2(p)
% tirage suivant la loi geometrique de parametre p
  x=1+floor(log(rand())/log(1-p));
```

Méthode 2 : En utilisant la loi de Bernoulli :

Lemme 6. *Si $(X_i)_{i\in\mathbb{N}^*}$ sont des variables aléatoires i.i.d. de loi de Bernoulli de paramètre p, alors $N = \min\{i : X_i = 1\}$ est une variable aléatoire qui suit une loi géométrique de paramètre p.*

L'algorithme correspondant s'écrit.

```
function x=geometrique1(p)
% tirage suivant la loi geometrique de parametre p
  x=1;
  while rand()>p, x=x+1;
 end
```

Méthode 3 : En utilisant la méthode générale pour la simulation des variables aléatoires discrètes, voir 4.3.1.

4.2.7 Loi de Poisson de paramètre λ

Soit $\lambda \in \mathbb{R}_+$. On veut simuler une variable aléatoire $X \sim \mathcal{P}(\lambda)$, donc de loi donnée par

$$\mu(x) = \sum_{k=0}^{+\infty} \exp(-\lambda)\frac{\lambda^k}{k!}\delta_k(x). \tag{4.8}$$

On est ici dans la situation $p_k = \mathbb{P}(X=k) = \frac{\lambda^k}{k!}e^{-\lambda}$, pour tout $k \geq 0$.
Une variable aléatoire poissonienne ne prend pas ses valeurs dans un ensemble fini, mais on peut utiliser la méthode générale pour les variables aléatoires discrètes présentée dans le

paragraphe 4.3.1. La méthode proposée ici annonce la méthode de la fonction inverse.

Méthode 1 : Cumul des durées exponentielles.

Une première méthode pour la simulation de variables aléatoires de Poisson est issue d'une des propriétés des processus de Poisson, il s'agit du cumul de durées exponentielles.

On sait que si des événements surviennent à des dates séparées par des durées exponentielles de paramètre λ, le nombre d'événements survenant en une unité de temps suit une loi de Poisson de même paramètre.

Lemme 7. *Soient $(X_i)_{i\in\mathbb{N}^*}$ des variables aléatoires i.i.d. exponentielles de paramètre λ. La variable aléatoire définie par*

$$N = \begin{cases} 0 & \text{si } X_1 \geq 1 \\ \max\{i \geq 1 : \sum_{j=1}^{i} X_j \leq 1\} & \text{sinon,} \end{cases}$$

suit une loi de Poisson de paramètre λ.

Démonstration : Soit $k \in \mathbb{N}$. On évalue :

$$\begin{aligned}
\mathbb{P}(N=k) &= \mathbb{P}\left(\textstyle\sum_{j=1}^{k} X_j \leq 1 < \sum_{j=1}^{k+1} X_j\right) \\
&= \int_{\mathbb{R}_+^{k+1}} \lambda^{k+1} \exp(-\lambda(x_1+\cdots+x_{k+1}))\mathbb{1}_{\{x_1+\cdots+x_k\leq 1\}} \\
&\qquad \times \mathbb{1}_{\{x_{k+1}>1-(x_1+\cdots+x_k)\}} \mathrm{d}x_1 \ldots \mathrm{d}x_{k+1} \\
&= \int_{\mathbb{R}_+^{k}} \lambda^{k} \exp(-\lambda(x_1+\cdots+x_k))\mathbb{1}_{\{x_1+\cdots+x_k\leq 1\}} \\
&\qquad \times \left(\int_{1-(x_1+\cdots+x_k)}^{+\infty} \lambda \exp(-\lambda x_{k+1}) \mathrm{d}x_{k+1}\right) \mathrm{d}x_1 \ldots \mathrm{d}x_k \\
&= \int_{\mathbb{R}_+^{k}} \lambda^{k} \exp(-\lambda(x_1+\cdots+x_k))\mathbb{1}_{\{x_1+\cdots+x_k\leq 1\}} \exp(-\lambda) \\
&\qquad \times \exp(\lambda(x_1+\cdots+x_k)) \mathrm{d}x_1 \ldots \mathrm{d}x_k \\
&= \lambda^k \exp(-\lambda) \textstyle\int_{\mathbb{R}_+^k} \mathbb{1}_{\{x_1+\cdots+x_k\leq 1\}} \mathrm{d}x_1 \ldots \mathrm{d}x_k.
\end{aligned}$$

Pour calculer la dernière intégrale, on effectue le changement de variable $s_1 = x_1$, $s_2 = x_1 + x_2$, …, $s_k = x_1 + \cdots + x_k$:

$$\begin{aligned}
\int_{\mathbb{R}_+^k} \mathbb{1}_{\{x_1+\cdots+x_k\leq 1\}} \mathrm{d}x_1 \ldots \mathrm{d}x_k &= \int_{\mathbb{R}_+^k} \mathbb{1}_{\{s_1\leq s_2\leq\cdots\leq s_k\leq 1\}} \mathrm{d}s_1 \ldots \mathrm{d}s_k \\
&= \int_0^1 \mathrm{d}s_k \int_0^{s_k} \mathrm{d}s_{k-1} \ldots \int_0^{s_2} \mathrm{d}s_1 = \frac{1}{k!}.
\end{aligned}$$

On voit donc que $\mathbb{P}(N=k) = \frac{\lambda^k}{k!}e^{-\lambda}$. Ce qui termine la preuve. ■

Remarque 5. *Il faut donc simuler des variables aléatoires exponentielles de paramètre λ et compter le nombre de simulations nécessaires pour dépasser 1, ou bien simuler des variables aléatoires exponentielles de paramètre 1 et compter le nombre de simulations nécessaires pour dépasser λ.*

Remarque 6. *Soient $(U_i)_{i\in\mathbb{N}^*}$ des variables aléatoires i.i.d. de loi uniforme sur $[0,1]$, alors si $X_i = -\frac{1}{\lambda}\ln(U_i)$, les $(X_i)_{i\in\mathbb{N}^*}$ sont des variables aléatoires i.i.d. exponentielles de paramètre λ, et*

$$\sum_{j=1}^{i} X_j \leq 1 \Leftrightarrow -\frac{1}{\lambda}\ln\left(\prod_{j=1}^{i} U_j\right) \leq 1 \Leftrightarrow \prod_{j=1}^{i} U_j \geq \exp(-\lambda).$$

L'algorithme s'écrit :

Algorithme Matlab :

```
function x=poisson(a)
% tirage suivant la loi de Poisson de parametre a
  test=exp(-a);x=0;prod=rand();
  while (prod>=test), x=x+1; prod=prod*rand(); end;
```

Méthode 2 : En utilisant la méthode générale pour les variables aléatoires discrètes, voir le paragraphe 4.3.1.

On sait que

$$X \sim \mathcal{P}(\lambda) \Leftrightarrow p_k = \mathbb{P}\{X = k\} = \frac{\lambda^k}{k!}e^{-\lambda} \quad (\text{pour } k \in \mathbb{N})$$

ce qui implique que

$$p_{k+1} = \frac{\lambda}{k+1}p_k,$$

et en notant q_k la somme des p_j pour $0 \leq j \leq k$, $(q_k = \sum_{j=0}^{k} p_j)$, on a

$$q_{k+1} = q_k + \frac{\lambda}{k+1}p_k.$$

On simule donc une variable de Poisson de paramètre λ en prenant

$$X = \sum_{k\geq 0} k\mathbb{1}_{\{q_{k-1}<U\leq q_k\}}$$

avec la convention $q_{-1} = 0$.

Cet algorithme est assez simple à programmer.

4.2.8 Loi gaussienne centrée réduite : Méthode de Box-Muller

La loi normale n'a pas une densité à support compact et on ne connaît pas d'expression simple de l'inverse de sa fonction de répartition. Nous utiliserons le résultat suivant pour sa simulation :

Lemme 8. *Soient U_1 et U_2 deux variables aléatoires de loi uniforme sur $[0,1]$, supposées de plus indépendantes. Alors, si on pose $X = \sqrt{-2\ln U_1}\cos(2\pi U_2)$ et $Y = \sqrt{-2\ln U_1}\sin(2\pi U_2)$, les variables aléatoires X et Y sont i.i.d. de loi gaussienne centrée réduite.*

Ceci conduit à une méthode de simulation simple pour une loi gaussienne, basée sur la simulation de deux variables aléatoires uniformes sur $[0,1]$. C'est la méthode de Box-Muller.

Algorithme Matlab :

```
function [x,y]=boxmuller
% tirage de deux N(0,1) independantes
  r=sqrt(-2*log(rand())); t=2*pi*rand();
  x=r*cos(t); y=r*sin(t);
end
```

4.3 Méthodes générales

4.3.1 Méthode générale pour une variable aléatoire discrète

Soit X une variable aléatoire sur un espace de probabilité $(\Omega, \mathcal{F}, \mathbb{P})$, telle que $X(\Omega) = \{x_0, x_1, ...\} = \{x_i\}_{i\in I}$ avec $I = \mathbb{N}$ ou $I = \{0, 1, \dots, n\}$. Pour tout $i \in I$, $p_i = \mathbb{P}(X = x_i)$. Considérons le cas $I = \{0, \dots, n\}$ fini, le second se traitant de la même manière. La loi de probabilité de X est donnée par

$$\mu(x) = \sum_{i=0}^{n} p_i \delta_{x_i}(x)$$

où

$$p_i = \mathbb{P}(X = x_i) = \mu(x_i), \forall i \in I \text{ et } \sum_{i=0}^{n} p_i = 1. \tag{4.9}$$

Notons par q_k le cumul des p_i, pour $0 \leq i \leq k$, *i.e.*

$$q_k = \sum_{i=0}^{k} p_i. \tag{4.10}$$

On a alors $q_0 = p_0$ et $q_n = 1$.

Nous souhaitons simuler une variable aléatoire de même loi que X. Pour ce faire on va construire une variable aléatoire Y telle que

$$\mathbb{P}(X = x_i) = \mathbb{P}(Y = x_i) = p_i, \quad \forall i \in I,$$

à l'aide d'une variable aléatoire U de loi uniforme sur $[0, 1]$.

Algorithme L'algorithme s'écrit : on tire U une uniforme sur $[0, 1]$ et on pose

$$\begin{cases} Y = x_0 \text{ si } U \leq p_0 = q_0 \\ Y = x_k \text{ si } q_{k-1} < U \leq q_k, \text{ pour } k \in \{0, \dots, n-1\} \end{cases}$$

ce qui s'exprime également sous la forme

$$Y = x_0 \mathbb{1}_{\{U \leq q_0\}} + \sum_{i=1}^{n} x_i \mathbb{1}_{\{q_{i-1} < U \leq q_i\}}.$$

Remarque 7. *Cette méthode générale peut s'appliquer pour tous les variables aléatoires discrètes particulières que nous avons étudiées.*

L'algorithme suivant simule une variable représentative pour cette loi :

Algorithme Matlab :

```
function y=simuldiscrete(x,p)
% simulation d'une v.a. de loi discrete
% x=vecteur des valeurs prises, p=vecteur des probabilites
  u=rand(); q=p(1); i=0;
  while (u>q); i=i+1; q=q+p(i); end;
  y=x(i);
end
```

Démonstration : En effet, on a : $\mathbb{P}(X = x_0) = \mathbb{P}(\texttt{rand()} \leq p_0) = p_0$ et pour $k \geq 1$,

$$\mathbb{P}(X = x_k) = \mathbb{P}\left(\sum_{i=0}^{k-1} p_i < \texttt{rand()} \leq \sum_{i=0}^{k} p_i\right) = p_k.$$

■

Remarque 8. *Le nombre N de tests nécessaires satisfait $N = 1$ ssi $u \leq p_0$, et pour $i > 1$,*

$$N = i \Leftrightarrow q_{i-1} < u \leq q_i.$$

On a donc intérêt à réordonner les $(x_i)_{i\geq 0}$ dans l'ordre des $(p_i)_{i\geq 0}$ décroissants.

4.3.2 Méthode de simulation par inversion de la fonction de répartition

On veut simuler une variable aléatoire X de fonction de répartition F.

La méthode utilisée pour simuler une loi exponentielle est en fait générale : dès que l'on sait inverser une fonction de répartition F, il est très facile de simuler une variable aléatoire de fonction de répartition F.

Lemme 9. *Soit U une variable aléatoire qui suit la loi uniforme sur $[0, 1]$, et F une fonction de répartition bijective de $]a, b[$ dans $]0, 1[$ d'inverse F^{-1}. Alors $F^{-1}(U)$ est une variable aléatoire de fonction de répartition F.*

Démonstration : On pose $X = F^{-1}(U)$. Cette variable aléatoire prend ses valeurs dans $]a, b[$. Remarquons que nécessairement F est strictement croissante de $]a, b[$ dans $]0, 1[$. Soit $t \in]a, b[$:

$$\mathbb{P}(X \leq t) = \mathbb{P}(F^{-1}(U) \leq t) = \mathbb{P}(U \leq F(t)) = F(t).$$

Donc la fonction de répartition de X est bien F. ■

Remarque 9. *L'hypothèse de la connaissance de F^{-1} n'a de sens que si F est strictement croissante. Cependant, même dans ce cas, il se peut que F^{-1} n'ait pas d'expression analytique simple, c'est le cas par exemple pour la loi normale.*

Si on note $\phi(x)$ la fonction de répartition de la loi normale centrée réduite,

$$\phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-\frac{t^2}{2}} \mathrm{d}t,$$

il n'existe pas de formulation simple de $\phi(x)$ et encore moins de $\phi^{-1}(x)$, la méthode de la fonction inverse ne peut donc pas s'appliquer directement à la loi normale.

Il existe cependant des polynômes donnant de bonnes approximations de $\phi(x)$ et de $\phi^{-1}(x)$ qui permettent donc d'appliquer la méthode de la fonction inverse à la loi normale moyennant cette approximation.

4.3.3 Algorithme par rejet

On veut simuler une variable aléatoire X de densité f et de fonction de répartition F.

Exemple 3. Commençons par un exemple très simple : comment simuler une loi uniforme sur le disque unité $\{x^2+y^2 \leq 1\}$?

```
function X=disque
% simule un point uniformement sur le disque unite
  X=2*rand(1,2)-[1,1];
while (norm(X)>1),
    X=2*rand(1,2)-[1,1];
end
```

L'idée est la suivante : on tire des points uniformément dans le carré $[0,1]\times[0,1]$, et on les jette jusqu'à en obtenir un qui tombe dans le disque. La loi du point obtenue est la loi d'un point tiré uniformément dans le carré conditionnellement à être dans le disque, ce qui est encore la loi uniforme sur le disque.

Remarque 10. *Quelle est la loi du nombre N de passages dans la boucle ?*

Lemme 10. *Soit X une variable aléatoire de densité f (sur $\mathbb{R}$) à simuler. On suppose qu'il existe une constante $k>0$ et une densité g (sur $\mathbb{R}$ aussi, facile à simuler) tels que*

$$\forall x, \quad f(x) \leq kg(x).$$

Soit U une variable aléatoire de loi uniforme sur $[0,1]$ et Z une variable aléatoire, indépendante de U, de densité g. On pose $V = kUg(Z)$. Alors, la loi de Z conditionnellement à l'événement $\{V < f(Z)\}$ a pour densité f.

Remarque 11. *Notons que nécessairement $k \geq 1$ (car f, g sont des densités).*

Remarque 12. *Il est très important de noter qu'on doit choisir la constante k la plus petite possible pour minimiser le nombre de rejets : plus la majoration est grossière, plus il faut des tirages pour obtenir une valeur acceptable.*

Démonstration : Notons que pour tout $z\in\mathbb{R}$, $\frac{f(z)}{kg(z)} \leq 1$. On a tout d'abord

$$\begin{aligned}
\mathbb{P}(V < f(Z)) &= \mathbb{P}(kUg(Z) < f(Z)) \\
&= \int_{\mathbb{R}} g(z) \left(\int_0^1 \mathbb{1}_{\{kug(z)<f(z)\}} \mathrm{d}u \right) \mathrm{d}z \\
&= \int_{\mathbb{R}} g(z) \frac{f(z)}{kg(z)} \mathrm{d}z \\
&= \frac{1}{k}.
\end{aligned}$$

Évaluons ensuite

$$\begin{aligned}
\mathbb{P}(\{Z \leq t\} \cap \{V < f(Z)\}) &= \int_{-\infty}^{t} g(z) \left(\int_0^1 \mathbb{1}_{\{kug(z)<f(z)\}} \mathrm{d}u \right) \mathrm{d}z \\
&= \int_{-\infty}^{t} g(z) \frac{f(z)}{kg(z)} \mathrm{d}z \\
&= \frac{1}{k} \int_{-\infty}^{t} f(z) \mathrm{d}z.
\end{aligned}$$

Ce qui montre que $\mathbb{P}(Z \leq t | V < f(Z)) = \displaystyle\int_{-\infty}^{t} f(z)\mathrm{d}z$, donc la loi conditionnelle de Z sachant que $\{V < f(Z)\}$ a bien pour densité f.
Ce résutltat se généralise à $\mathbb{R}^d$. ■

On obtient donc l'algorithme de simulation par rejet (on suppose qu'on possède une fonction `simulg` qui simule une variable aléatoire de densité g) :

Algorithme :

```
function z=simulf
% simule par rejet une va de densite f
  u=rand();
  z=simulg;
  v=k*u*g(z);
  while (v>=f(z));
      u=rand();
      z=simulg;
      v=k*u*g(z);
end
```

Démonstration : Notons N le nombre de tests fait lors de cette fonction. N est une variable aléatoire, à valeurs dans $\mathbb{N}^*$. Notons $(U_n)_{n\geq 1}$ la suite des appels à la fonction `rand()`, et $(Z_n)_{n\geq 1}$ la suite des appels à la fonction `simulg`. Toutes ces variables aléatoires sont indépendantes, les premières de loi uniforme sur $[0,1]$, les secondes de densité g. On note $V_n = kU_n g(Z_n)$, et on note X la sortie de la fonction.

Soit $t \in \mathbb{R}$. Evaluons

$$\mathbb{P}(X \leq t \text{ et } N = 1) \;=\; \mathbb{P}(V_1 < f(Z_1) \text{ et } Z_1 \leq t) = \frac{1}{k}\int_{-\infty}^{t} f(z)\mathrm{d}z$$

par la démonstration précédente. Soit maintenant $i \geq 2$. Par indépendance, et comme précédemment :

$$\begin{aligned}
&\mathbb{P}(X \leq t \text{ et } N = i)\\
&= \mathbb{P}(V_1 \geq f(Z_1),\ V_2 \geq f(Z_2),\ \ldots,\ V_{i-1} \geq f(Z_{i-1}),\ V_i < f(Z_i),\ Z_i \leq t)\\
&= \mathbb{P}(V_1 \geq f(Z_1))\mathbb{P}(V_2 \geq f(Z_2))\ldots\mathbb{P}(V_{i-1} \geq f(Z_{i-1}))\mathbb{P}(V_i < f(Z_i),\ Z_i \leq t)\\
&= \left(1 - \frac{1}{k}\right)^{i-1}\frac{1}{k}\int_{-\infty}^{t} f(z)\mathrm{d}z.
\end{aligned}$$

Finalement (rappelons que $k \geq 1$) :

$$\begin{aligned}
\mathbb{P}(X \leq t) &= \sum_{i=1}^{+\infty}\mathbb{P}(X \leq t \text{ et } N = i)\\
&= \sum_{i=1}^{+\infty}\left(1 - \frac{1}{k}\right)^{i-1}\frac{1}{k}\int_{-\infty}^{t} f(z)\mathrm{d}z\\
&= \int_{-\infty}^{t} f(z)\mathrm{d}z,
\end{aligned}$$

donc la densité de X est bien f. ■

4.3.4 Simulation par composition

Exemple 4. Soit F et G deux fonctions de répartition sur $\mathbb{R}$. On construit une variable aléatoire X de la façon suivante : on lance une pièce qui tombe sur pile avec probabilité $1/3$ et sur face avec probabilité $2/3$, si pile sort, on tire un nombre au hasard suivant la loi donnée par F, sinon, on tire un nombre au hasard suivant la loi donnée par G. Déterminer la fonction de répartition H de X.

L'exemple précédent est un exemple de mélange de variables aléatoires. On suppose maintenant qu'on veut simuler une variable aléatoire X de fonction de répartition $F = \sum_{i=1}^{n} \theta_i F_i$, où les θ_i sont des poids : $\theta_i \geq 0$ et $\sum_{i=1}^{n} \theta_i = 1$ et les F_i sont des fonctions de répartition dont les lois sont faciles à simuler. On suppose qu'on a à notre disposition des fonctions `simulFi` qui simulent des variables aléatoires de fonction de répartition F_i et une fonction `simulTheta` qui simule une variable aléatoire Θ à valeur dans $\{1, ..., n\}$ telle que $\mathbb{P}(\Theta = i) = \theta_i$.

```
function x = melange
% simulation par melange
  i=simulTheta;
  x=simulFi;
end
```

Cette méthode se généralise immédiatement à un nombre infini dénombrable de poids $(\theta_i)_{i\in\mathbb{N}}$, et même à un mélange "continu" de lois : on suppose qu'on veut simuler une variable aléatoire X de densité $f(z) = \int_\Theta g(\theta) f_\theta(z) \mathrm{d}\theta$, où g est une densité de probabilité, ainsi que tous les f_θ. L'algorithme est alors simple : on tire θ suivant g, puis x suivant f_θ.

4.4 Simulation de vecteurs aléatoires

4.4.1 Cas indépendant

Supposons qu'on souhaite simuler Z un vecteur aléatoire de $\mathbb{R}^d$. Si ses composantes sont indépendantes, on est ramené au cas de la simulation de variables aléatoires réelles indépendantes traité dans les sections précédentes (on rappelle que les sorties successives de `rand` donnent des variables aléatoires *indépendantes* de loi uniforme sur $[0, 1]$). Le problème est différent quand les coordonnées ne sont pas indépendantes.

4.4.2 Vecteur gaussien

Un vecteur gaussien dans $\mathbb{R}^d$ est caractérisé par un vecteur moyenne $m \in \mathbb{R}^d$, et une matrice de covariance K, de taille $d \times d$, symétrique et positive. On suppose pour l'instant que K est *définie positive*.

Théorème 1 (Décomposition de Cholesky)**.** *Si K est une matrice symétrique définie positive de taille $d \times d$, il existe au moins une matrice réelle triangulaire inférieure L telle que :*

$$K = L^t L.$$

On peut également imposer que les éléments diagonaux de la matrice L soient tous strictement positifs, et la factorisation correspondante est alors unique.

En pratique on cherche L par coefficients indéterminés et identification, colonne par colonne. Voir l'exemple juste après.

Remarque 13. *Si K est seulement semi-définie positive, la décomposition de Cholesky existe encore mais elle n'est plus unique.*

Lemme 11. *Soit T un vecteur gaussien de dimension d, centré et de matrice de covariance I_d (dont les coordonnées sont des variables aléatoires i.i.d. normales centrées réduites). Soit $m \in \mathbb{R}^d$, et K une matrice définie positive de taille $d \times d$. Soit L la matrice triangulaire inférieure donnée par la factorisation de Cholesky de K.*

Alors le vecteur aléatoire $Z = m + LT$ est un vecteur gaussien de moyenne m et de matrice de covariance K.

Démonstration : Comme L est une application linéaire de $\mathbb{R}^d$ dans $\mathbb{R}^d$, Z est encore un vecteur gaussien d-dimensionnel. Pour l'espérance :

$$\mathbb{E}(Z) = \mathbb{E}(m + LT) = m + L\mathbb{E}(T) = m,$$

puisque T est centré. Pour la matrice de covariance, comme celle de T est I_d,

$$\mathbb{E}((Z-m)^t(Z-m)) = \mathbb{E}(LT^tT^tL) = L\mathbb{E}(T^tT)^tL = LI_d^tL = L^tL = K.$$

■

Exemple 5. On veut simuler le vecteur gaussien Z de $\mathbb{R}^3$ de moyenne m et de matrice de covariance K avec

$$m = \begin{pmatrix} 1 \\ -2 \\ 4 \end{pmatrix} \text{ et } K = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 5 & 6 \\ 0 & 6 & 10 \end{pmatrix}.$$

On commence par chercher la matrice de factorisation de Cholesky L par coefficients indéterminés et identification :

$$L = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix}.$$

On calcule LL^t et on identifie, colonne par colonne :

1. $l_{11}^2 = k_{11} = 1 \Rightarrow l_{11} = 1$.
2. $l_{11}l_{21} = k_{21} = -1 \Rightarrow l_{21} = -1$.
3. $l_{11}l_{31} = k_{31} = 0 \Rightarrow l_{31} = 0$.
4. $l_{21}^2 + l_{22}^2 = k_{22} = 5 \Rightarrow l_{22} = 2$.
5. $l_{21}l_{31} + l_{22}l_{32} = k_{32} = 6 \Rightarrow l_{32} = 3$.
6. $l_{31}^2 + l_{32}^2 + l_{33}^2 = k_{33} = 1 \Rightarrow l_{33} = 1$.

Donc

$$Z = \begin{pmatrix} 1 \\ -2 \\ 4 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ -1 & 2 & 0 \\ 0 & 3 & 1 \end{pmatrix} T,$$

où T est un vecteur gaussien de $\mathbb{R}^3$ dont les trois composantes sont i.i.d. centrées réduites.

4.4.3 Cas général

1. Le cas d'une v.a. discrète à valeurs dans $\mathbb{R}^d$ se traite comme celui d'une variable aléatoire réelle discrète.

2. La méthode de rejet a été présentée dans la cas d'une variable aléatoire à valeurs dans $\mathbb{R}^d$.

3. On peut encore utiliser les lois conditionnelles. Nous allons illustrer cette méthode, dite **méthode récurrente**, par un exemple :

Exemple 6. On veut simuler un vecteur (X, Y) de loi uniforme sur le triangle ABC avec $A = (0, 0)$, $B = (1, 1)$ et $C = (0, 1)$.

On commence par déterminer la densité de cette loi :

$$f(x, y) = 2\mathbb{1}_{0\leq x\leq 1}\mathbb{1}_{0\leq y\leq 1}\mathbb{1}_{x\leq y}.$$

On peut alors calculer la densité de X :

$$f_X(x) = 2(1 - x)\mathbb{1}_{0\leq x\leq 1},$$

puis la densité de la loi conditionnelle de Y sachant X :

$$f_{Y|X}(y|x) = \frac{1}{1 - x}\mathbb{1}_{x\leq y\leq 1}.$$

Pour la simulation, on procède maintenant de la façon suivante : on simule X suivant sa densité (en utilisant par exemple la méthode de la fonction de répartition), on obtient une valeur x, puis on simule Y suivant la densité $f_{Y|X}(y|x) = \frac{1}{1-x}\mathbb{1}_{x\leq y\leq 1}$ (on reconnaît par exemple une loi usuelle).

Remarque 14. *Remarquons que la seconde étape ressemble beaucoup à la simulation par mélange.*

4.5 Exercices

Exercice 1. Soit X une variable aléatoire de loi de Bernoulli de paramètre p.

1. Proposer une fonction qui construit une réalisation de X.
2. Proposer une fonction qui renvoie un k-échantillon de cette variable aléatoire.
3. Tester ces résultats en Matlab pour différentes valeurs de p et k.

Exercice 2. Simuler une variable aléatoire X de loi binomiale de paramètres (n, p) pour différentes valeurs de n et p.

Exercice 3. Soit X une variable aléatoire de loi uniforme sur $\{1, \dots n\}$. Proposer un algorithme en Matlab pour la simulation de cette loi :

1. En utilisant la simulation d'une loi de probabilité discrète sur un ensemble fini.
2. En utilisant le résultat suivant : Si U est une variable aléatoire de loi uniforme sur $[0, 1]$ alors $[nU] + 1$ est une variable aléatoire de loi uniforme sur $\{1, \dots, n\}$.

Simuler les résultats pour différentes valeurs de n. Comparer les deux approches.

Exercice 4. Soit X une variable aléatoire de loi uniforme sur $[a, b]$ avec $a, b \in \mathbb{R}$, $a < b$. Proposer un algorithme pour la simulation de cette loi et tester le pour différentes valeurs de a et b.

Exercice 5. Soit X une variable aléatoire de loi exponentielle de paramètre λ. Proposer un algorithme pour la simulation de cette loi en utilisant la méthode de la fonction inverse.

Exercice 6. Soit X une variable aléatoire géométrique de paramètre p. Proposer un algorithme pour la simulation de cette loi :

1. En utilisant le résultat : Si $(Y_i)_{i\in\mathbb{N}^*}$ sont des variables aléatoires i.i.d. de loi de Bernoulli de paramètre p, alors $N = \min\{i; Y_i = 1\}$ suit une loi géométrique de paramètre p.
2. En utilisant le résultat : Si U suit une loi uniforme sur $[0, 1]$ alors $G = 1 + \left[\dfrac{\ln U}{\ln(1-p)}\right]$ suit la loi géométrique de paramètre p.
3. Comparer les résultats en insistant sur le temps de calcul.

Exercice 7. Soit Y une variable aléatoire de loi exponentielle de paramètre λ, avec $\lambda > 0$. Sa densité est donnée par

$$f(x) = \lambda e^{-\lambda x} \mathbb{1}_{\{x>0\}}.$$

On définit une variable aléatoire $Z = 1 + [Y]$, où, pour tout $x \in \mathbb{R}$, $[x]$ note la partie entière de x.

1. Quelle est la loi de Z ? Justifier votre réponse.
2. Soient $p \in]0, 1[$ et U une variable aléatoire de loi uniforme sur $[0, 1]$. Trouver la constante λ telle que
$$1 + \left[-\frac{\ln U}{\lambda}\right] \sim \mathcal{G}(p).$$
3. Utiliser ce résultat pour écrire un algorithme pour la simulation d'une variable aléatoire X de loi géométrique de paramètre p.
4. Donner des résultats numériques de votre algorithme pour $p = 0.1$, $p = 0.5$ et $p = 0.9$ (4 valeurs pour chaque choix de p).

Exercice 8. Soit X une variable aléatoire de loi de Poisson de paramètre λ. Simuler cette variable aléatoire en utilisant le résultat suivant :
Soit $(X_i)_{i\in\mathbb{N}^*}$ une suite de variables aléatoires i.i.d. de loi exponentielle de paramètre λ. Alors la variable aléatoire :

$$N = \begin{cases} 0, & \text{si } X_1 \geq 1 \\ \max\{i \geq 1;\ \sum_{j=1}^{i} X_j \leq 1\}, & \text{sinon} \end{cases}$$

suit une loi de Poisson de paramètre λ.

Exercice 9. Soit X une variable aléatoire de loi normale de moyenne m et variance σ^2. Simuler cette variable aléatoire en utilisant la méthode de Box-Muller.

Exercice 10. En utilisant la méthode de la fonction de répartition, simuler une loi de Cauchy. Soit X une variable aléatoire de loi de Cauchy de paramètre $a > 0$. Sa densité est donée par :

$$f(x) = \frac{a}{\pi(x^2 + a^2)}, \quad x \in \mathbb{R}.$$

1. Vérfier que f est une densité.
2. Calculer sa fonction de répartition F.
   (*Indication :* $F(x) = \frac{1}{\pi}\arctan(\frac{x}{a}) + \frac{1}{2}$.)
3. En déduire G l'inverse de la fonction F.
4. Soit U une variable aléatoire de loi uniforme sur $[0, 1]$. En utilisant la périodicité de la fonction tan, montrer que

   $$\tan\left(\pi\left(U - \frac{1}{2}\right)\right) = \tan(\pi U), \quad \text{en loi .}$$

   (*Indication :* Montrer que pour toute fonction φ continue et bornée on a
   $\mathbb{E}\left\{\varphi\left(\tan\left(\pi\left(U - \frac{1}{2}\right)\right)\right)\right\} = \mathbb{E}\{\varphi(\tan(\pi U))\}$.)
5. Écrire un algorithme pour simuler une variable aléatoire de loi de Cauchy de paramètre a.
6. Donner des réalisations de votre code Matlab pour $a = 10$ et $a = 1$ (4 valeurs numériques pour chaque choix de a).
7. Tracer la vraie densité et un histogramme à l'aide de cet algorithme pour $a = 10$ et $a = 1$. Utiliser un échantillon de taille 10000.

Exercice 11. Simulation de la gaussienne par rejet par rapport à la double exponentielle. On pose

$$f(z) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{z^2}{2}\right) \text{ et } g(z) = \frac{1}{2}\exp(-|z|).$$

1. Montrer que g est bien une densité sur $\mathbb{R}$.
2. Déterminer une constante k satisfaisant $\forall z \in \mathbb{R},\ f(z) \leq kg(z)$. On aura intérêt à prendre k la plus petite possible : $[k = \sqrt{\frac{2e}{\pi}} = 1,3155]$.
3. Écrire un algorithme de simulation d'une loi gaussienne centrée réduite par rejet.

Exercice 12. Considérons l'algorithme de la méthode par rejet. Montrer que le nombre de passage dans la boucle suit une loi géométrique dont le paramètre ne dépend que de k. Justifier alors l'intérêt de prendre k la plus petite possible.

Exercice 13. Simuler un mélange d'exponentielles $F(x) = \alpha(1 - \exp(-ax)) + (1 - \alpha)(1 - \exp(-bx))$, avec $\alpha \in [0, 1]$.

Exercice 14. Simuler une variable aléatoire de loi $\frac{1}{2}\delta_0(x) + \frac{1}{2}e^{-x}\mathbb{1}_{\mathbb{R}_+}(x)\mathrm{d}x$.

Exercice 15. Simuler le vecteur gaussien Z de $\mathbb{R}^3$ de moyenne m et de matrice de covariance K avec

$$m = \begin{pmatrix} 1 \\ -2 \\ 4 \end{pmatrix} \text{ et } K = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 5 & 6 \\ 0 & 6 & 10 \end{pmatrix}.$$

Exercice 16. Simuler un vecteur (X, Y) de loi uniforme sur le triangle ABC avec $A = (0, 0)$, $B = (1, 1)$ et $C = (0, 1)$.

Exercice 17. Simuler un vecteur de loi $f(x, y, z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$, où

$$f(x, y, z) = 6\mathbb{1}_{x>0,y>0,z>0}\mathbb{1}_{x+y+z<1} :$$

1. En utilisant les lois conditionnelles.

2. Par la méthode du rejet.

3. Comparer.

Exercice 18. Simuler un vecteur de loi $f(x, y)\mathrm{d}x\mathrm{d}y$, où

$$f(x, y) = \frac{1}{2}(x + y)e^{-(x+y)}1_{x>0}1_{y>0}.$$

Simulation de variables aléatoires

Dans tout ce qui suit U note une v.a. de loi uniforme sur $[0,1]$ et $(U_n)_{n\geq 1}$ une suite de v.a. i.i.d., avec U_1 de loi uniforme sur $[0,1]$.

Loi	**Méthode de simulation**
Bernoulli $\mathcal{B}(p)$, $p \in [0,1]$	$\mathbb{1}_{\{U\leq p\}}$
Binomiale $\mathcal{B}(n,p)$, $n \in \mathbb{N}^*$, $p \in [0,1]$	$\displaystyle\sum_{i=1}^{n} \mathbb{1}_{\{U_i\leq p\}}$
Uniforme sur $\{0,\ldots,n-1\}$	$[nU]$
Uniforme sur $[a,b]$	$a+(b-a)U$
Exponentielle $\mathcal{E}(\lambda)$	$\displaystyle -\frac{\ln(U)}{\lambda}$
Géométrique $\mathcal{G}(p)$	$\displaystyle 1+\left[\frac{\ln(U)}{\ln(1-p)}\right]$
Poisson $\mathcal{P}(\lambda)$	$\displaystyle \min\left\{n\in\mathbb{N} : \prod_{i=1}^{n} U_i \leq e^{-\lambda}\right\}$
Gaussienne $\mathcal{N}(m,\sigma^2)$	$m+\sigma\sqrt{-2\ln(U_1)}\sin(2\pi U_2)$
Cauchy $\mathcal{C}(a)$	$a\tan(\pi U)$