Simulation de variables aléatoires

Prépa Agreg Option A

Nathan Huguenin

Introduction

Ce document a pour but de présenter les différentes méthodes de simulation au programme de l'Option A de l'agrégation. Le langage utilisé est Python. On utilise ici les sous-modules `numpy.random` et `scipy.stats` de Python. On pourra par exemple les appeler au début des programmes par `import numpy.random as rd` et `import scipy.stats as st`. Le poly est très largement inspiré de l'ouvrage de Luc Devroye, "Non-uniform random variate generation", disponible sur internet http://luc.devroye.org/rnbookindex.html. Citons aussi (en moins détaillé mais qui couvre tout le programme de l'option et qui est disponible dans toutes les bibliothèques dont celle de l'I2M et celle du lycée où aura lieu l'oral, contrairement au Devroyre qui est dur à trouver) "Probabilités et Statistiques pour l'épreuve de modélisation à l'agrégation de Mathématiques" de Marie-Line Chabanol et Jean-Jacques Ruch.

Pour une liste de fonctions disponibles (et plus que ça), consulter la doc du module `numpy.random` https://numpy.org/doc/stable/reference/random/index.html. La doc du module `scipy.stats` est disponible à l'adresse https://docs.scipy.org/doc/scipy/reference/stats.html.

1 Simulation sans inversion de la fonction de répartition

On considère que la commande `rd.random(size=n)` renvoie un échantillon de n réalisations de la loi $\mathcal{U}([0,1))$. Elle peut servir à simuler très simplement des réalisations de lois usuelles.

Loi de Bernoulli de paramètre p. Il suffit de partager l'intervalle $[0,1]$ en deux sous-intervalles de longueurs respectives p et $1-p$:

```
u = rd.random()
x = 0
if u < p:
    x = 1
```

ou encore,

```
u = rd.random()
x = u < p
```

Dans le second exemple `x` est considéré comme un booléen (`true` pour 1 ou `false` pour 0). En Python il est possible de faire des opérations entre des entiers et des booléens, et le booléen `x == 1` vaut `true` si `x` vaut `true`.

Dé déséquilibré. Si on souhaite simuler un dé déséquilibré à n faces dont les probabilités d'apparition sont données par $p_1, ..., p_n$ avec $p_1 + ... + p_n = 1$, on procède de la même manière en partageant l'intervalle $[0,1]$ en n sous-intervalles de longueurs respectives $p_1, ..., p_n$. Par exemple pour un dé à 4 faces numérotées $-1, 0, 3, 5$ de probas respectives $1/2, 1/4, 1/6, 1/12$:

```
u = rd.random()
x = 5
if u < 1/2:
    x = -1
elif u < 3/4:
    x = 0
elif u < 11/12:
    x = 3
```

Selon les cas, il est possible d'écrire un code plus concis (loi uniforme, par exemple...).

Loi binomiale de paramètres (n, p)**.** C'est la même méthode que pour la loi de Bernoulli, on commencera simplement par générer un échantillon de taille n :

```
u = rd.random(n)
y = u < p
x = y.sum()
```

Loi géométrique de paramètre p**.** C'est la loi du nombre d'expériences de Bernoulli de paramètre p nécessaires à l'obtention d'un premier succès :

```
x = 1
u = rd.random()
while u > p:
    x += 1
```

Exercise 1.1. *Adapter les algorithmes Python précédents pour simuler plusieurs réalisations des v.a. en même temps.*

2 Simulation par inversion de la fonction de répartition

Un rappel de la définition d'une fonction de répartition.

Definition 2.1. *La fonction de répartition F d'une loi de probabilité $\mathbb{P}$ définie sur $(\mathbb{R}, \mathcal{B}(\mathbb{R}))$ est la fonction qui à tout réel t associe $\mathbb{P}((-\infty, t])$. La fonction de répartition F_X d'une variable aléatoire réelle X de loi $\mathbb{P}$ est la fonction qui à un réel t associe $\mathbb{P}(X \leq t)$.*

Une fonction de répartition est aussi caractérisée par les propriétés suivantes :

- elle est croissante
- elle est continue à droite en tout point
- elle tend vers 0 en $-\infty$ et vers 1 en $+\infty$.

Definition 2.2 (π-système)**.** *Un π-système est une classe $\mathcal{C}$ de parties d'un ensemble, stable par intersection finie, i.e.*

$$A, B \in \mathcal{C} \Rightarrow A \cap B \in \mathcal{C}.$$

Definition 2.3 (Classe monotone)**.** *Une classe (ensemble de parties) $\mathcal{M}$ d'un ensemble Ω est dite monotone si elle contient Ω, qu'elle est stable par différence et par réunion croissante :*

$$A, B \in \mathcal{M}, A \subset B \Rightarrow B \setminus A \in \mathcal{M}$$

$$\forall n \geq 0, A_n \in \mathcal{M}, A_n \subset A_{n+1} \Rightarrow \cup_{n \geq 0} A_n \in \mathcal{M}.$$

Lemma 2.4 (des classes monotones)**.** *La plus petite classe monotone contenant le π-système $\mathcal{C}$ est la tribu engendrée par $\mathcal{C}$.*

Lemma 2.5. *Deux mesures de probabilités sont égales si et seulement si elles ont même fonction de répartition.*

Proof. C'est le lemme des classes monotones (ou théorème d'unicité des probas) appliqué à la classe $\mathcal{C} = \{(-\infty, t], t \in \mathbb{R}\}$. On vérifie que c'est un π-système, c'est-à-dire que la classe est stable par intersection. Supposons que les deux mesures de probas $\mathbb{P}$ et $\mathbb{Q}$ ont la même fonction de répartition. Cela revient à dire qu'elles sont égales sur le π-système $\mathcal{C}$. Soit $\mathcal{M}$ la classe des boréliens sur lesquels $\mathbb{P}$ et $\mathbb{Q}$ sont égales. On vérifie facilement que $\mathcal{M}$ est une classe monotone (ou λ-système), c'est-à-dire qu'elle est stable par différence, par union croissante, et qu'elle contient $\mathbb{R}$ (c'est vrai car $\mathbb{P}(\mathbb{R}) = \mathbb{Q}(\mathbb{R}) = 1$). De plus $\mathcal{M}$ contient $\mathcal{C}$. Donc $\mathcal{M}$ contient la plus petite classe monotone contenant $\mathcal{C}$. Par le lemme des classes monotones, $\mathcal{M}$ contient donc la tribu engendrée[1] par $\mathcal{C}$, c'est-à-dire $\mathcal{B}(\mathbb{R})$, et donc $\mathbb{P}$ et $\mathbb{Q}$ sont égales. La réciproque est évidente. □

Proposition 2.6. *Soit F une fonction de répartition, et soit F^- la fonction quantile*

$$F^-(u) = \inf\{t \ : \ F(t) \geq u\} \quad \textit{pour } u \in (0,1)$$

Si U est une variable aléatoire de loi uniforme sur $[0,1]$, alors $X := F^-(U)$ a pour fonction de répartition F.

Proof. Pour $u \in (0,1)$, notons A_u l'ensemble $\{t \ : \ F(t) \geq u\}$. On veut montrer que

$$F^-(u) \leq x \Leftrightarrow u \leq F(x).$$

En effet si c'est vrai alors $\mathbb{P}(F^-(U) \leq x) = \mathbb{P}(U \leq F(x)) = F(x)$. Montrons donc cette équivalence. De droite à gauche c'est facile, si $u \leq F(x)$ alors $x \in A_u$ et donc $x \geq F^-(u)$. Dans l'autre sens, supposons que $F^-(u) \in A_u$, alors $F(F^-(u)) \geq u$, mais comme F est croissante et que $F^-(u) \leq x$ alors $F(F^-(u)) \leq F(x)$ et donc $u \leq F(x)$. Supposons maintenant que $F^-(u) \notin A_u$, par la propriété de la borne inférieure il existe une suite (t_n) d'éléments de A_u qui converge vers $F^-(u)$. On $F(t_n) \geq u$, et comme F est continue à droite $F(F^-(u)) \geq u$ et donc $F^-(u) \in A_u$ ce qui est le contraire de ce qu'on avait supposé. On a donc montré la proposition. □

Ces deux résultats montrent qu'on peut simuler des réalisations d'une variable aléatoire X de fonction de répartition F, à condition de savoir calculer (à la main ou par un algorithme) la fonction quantile associée. Par exemple si X est à densité cela revient à inverser la fonction de densité. Voici quelques exemples (non-exhaustif).

La loi $\mathcal{U}([a,b])$. Il suffit de faire une transformation affine :

```
u = rd.random()
x = (b-a)*u + a
```

La loi exponentielle $\mathcal{E}(\lambda)$. La fonction de répartition de la loi exponentielle est la fonction $t \mapsto (1 - e^{-\lambda t})\mathbb{1}_{t \geq 0}$. Sa réciproque est la fonction $u \mapsto -\lambda^{-1}\ln(1-u)$. Le code suivant permet donc de simuler une réalisation de la loi exponentielle (si U est une v.a. uniforme sur $[0,1]$, U a la même loi que $1-U$) :

```
u = rd.random()
x = -np.log(u)/lambda
```

(`np` désigne le package `numpy`, dont le logarithme sert quand on souhaite l'appliquer directement à un vecteur.)

[1] Tribu : ensemble de parties non-vide, stable par complémentaire et union dénombrable. Tribu engendrée par $\mathcal{C}$: plus petite tribu contenant $\mathcal{C}$ (= intersection des tribus contenant $\mathcal{C}$).

La loi de Cauchy de paramètre σ. Sa fonction de répartition est donnée par $t \mapsto \frac{1}{2} + \frac{1}{\pi}\arctan(\frac{x}{\sigma})$, sa réciproque est la fonction $u \mapsto \sigma\tan(\pi(u - \frac{1}{2}))$.

Loi de Bernoulli de paramètre p. On obtient pour la réciproque la fonction $u \mapsto \lfloor p + u \rfloor$, on peut vérifier que le code qui en résulte est équivalent à celui de la section précédente.

Exercise 2.7. *Ecrire et adapter les codes précédents pour simuler directement un nombre n de variables i.i.d (on écrira une fonction qui prend* `n` *en paramètre).*

Loi du max. Supposons qu'on veuille simuler une réalisation de la loi du maximum de n variables aléatoires i.i.d. On peut bien sûr (si on sait faire) simuler n réalisations puis prendre le maximum. Supposons qu'on sache résoudre l'équation $F_X(t)^n = u$ (ce qui est le cas dès qu'on a une réciproque de la fonction de répartition F_X), alors on peut aussi simuler une seule réalisation de la loi uniforme et appliquer la méthode d'inversion !

Exercise 2.8. *Pourquoi ?*

3 Méthodes de rejet

Theorem 3.1. *Soit (X_n) une suite de vecteurs aléatoires i.i.d. à valeurs dans $\mathbb{R}^d$, et soit A un Borélien de $\mathbb{R}^d$ tel que $\mathbb{P}(X_1 \in A) = p > 0$. Soit $T = \min\{n > 0 \ : \ X_n \in A\}$. Alors :*

- *T suit une loi géométrique de paramètre p, donc est finie p.s.*
- *la loi de X_T est déterminée par*

$$\mathbb{P}(X_T \in B) = \frac{\mathbb{P}(X_1 \in A \cap B)}{p}, \quad B \in \mathcal{B}(\mathbb{R}^d)$$

Proof. Soit $n > 0$, on a

$$\mathbb{P}(T = n) = \mathbb{P}(X_1 \notin A, ..., X_{n-1} \notin A, X_n \in A) = (1-p)^{n-1}p$$

Ce qui montre le premier point. Soit B un Borélien. Alors

$$\begin{aligned}\mathbb{P}(X_T \in B) &= \sum_{i=1}^{\infty} \mathbb{P}(X_1 \notin A, ..., X_{i-1} \notin A, X_i \in A \cap B) \\ &= \sum_{i=1}^{\infty} (1-p)^{i-1}\mathbb{P}(X_1 \in A \cap B) \\ &= \frac{1}{1-(1-p)}\mathbb{P}(X_1 \in A \cap B)\end{aligned}$$

d'où le second point. □

En particulier, si X_1 suit une loi uniforme sur A_0 avec $A \subset A_0$, alors X_T suit une loi uniforme sur A (facile à voir). Il est donc possible, en sachant générer des points uniformes dans un Borélien donné, d'en générer dans un Borélien plus petit (et a priori plus compliqué). Par exemple le code suivant permet de simuler un point uniforme sous la courbe de x^2 entre les abscisses 0 et 1 (donc $d = 2$) :

```
x = rd.random()
y = rd.random()
while y > x^2:
    x = rd.random()
    y = rd.random()
```

Ici, $A_0 = [0,1)^2$ et $A = \{(x,y) \ : \ x \in [0,1), 0 \leq y \leq x^2\}$. On peut par exemple calculer une intégrale de cette manière (Monte-Carlo). Cependant le véritable intérêt de ce résultat n'apparaît qu'une fois combiné avec le théorème fondamental suivant.

Theorem 3.2. *Soit X un vecteur aléatoire de densité f sur $\mathbb{R}^d$. Soit U une variable aléatoire uniforme sur $[0,1]$ indépendante de X. Alors $(X, cUf(X))$ est uniformément distribué sur $A = \{(x,u) \in \mathbb{R}^d \times \mathbb{R} \ : \ 0 \leq u \leq cf(x)\}$, avec $c > 0$ une constante arbitraire. Réciproquement, si (X,U) est un vecteur aléatoire de $\mathbb{R}^d \times \mathbb{R}$ uniformément distribué sur A, alors X a pour densité f sur $\mathbb{R}^d$.*

Proof. Pour le premier point, on remarque facilement que $(X, cUf(X))$ est à valeurs dans A et que $|A| = c$ (où $|\cdot|$ désigne la mesure de Lebesgue). Soit $B \subset A$ un Borélien. Alors par Fubini-Tonelli,

$$\mathbb{P}((X, cUf(X)) \in B) = \int \left(\int_{\{u \ : \ (x,u) \in B\}} \frac{du}{cf(x)} \right) f(x)dx = \frac{1}{c} \int_B dudx$$

Pour la réciproque, soit B un Borélien, et soit B_0 l'ensemble $\{(x,u) \ : \ x \in B, 0 \leq u \leq cf(x)\}$. Alors,

$$\mathbb{P}(X \in B) = \mathbb{P}((X,U) \in B_0) = \frac{\int_{B_0} dudx}{\int_A dudx} = \frac{1}{c} \int_B cf(x)dx = \int_B f(x)dx$$

ce qui conclut la preuve. □

On souhaite simuler une réalisation d'une variable aléatoire de densité donnée f. Le théorème 3.2 dit qu'on peut alors appliquer la méthode dictée par le théorème 3.1 à l'ensemble $A = \{(x,y) \ : \ 0 \leq y \leq f(x)\}$. Il faut pour cela savoir calculer la densité f, et savoir trouver un ensemble $A_0 \supset A$ "simple" (en pratique, un pavé). En particulier cela n'est pas possible si f a un support non borné, ce qui est souvent le cas. Supposons en revanche que l'on connaisse un réel $c > 0$ et une densité g que l'on sait simuler (par la méthode d'inversion par exemple), tels que $f \leq cg$. Soient les ensembles $A = \{(x,y) \ : \ 0 \leq y \leq f(x)\}$ et $A_0 = \{(x,y) \ : \ 0 \leq y \leq cg(x)\}$. Le premier point du théorème 3.2 nous permet de simuler des points uniformes dans A_0, puis, le théorème 3.1 nous permet d'obtenir un point uniforme dans A (à vitesse "géométrique"). Enfin, le second point du théorème 3.2 nous assure qu'on simule de cette façon une réalisation d'une v.a. de densité f. Plus précisément :

Theorem 3.3. *Soit f une densité de probabilités. On suppose qu'il existe $c > 0$ et g une densité de probabilités tels que $f \leq cg$. Soit (X_n) une suite de vecteurs aléatoires i.i.d. de densité g, et (U_n) une suite de v.a. i.i.d. uniformes sur $[0,1]$ indépendantes de (X_n). Alors*

- *$T = \min\{n > 0 \ : \ cU_n g(X_n) < f(X_n)\}$ suit une loi géométrique de paramètre $\frac{1}{c}$*
- *X_T a pour densité f.*

En pratique, l'algorithme suivant permet de mettre en œuvre la méthode du rejet.

```
def simul_g():
    # fonction qui simule une réalisation d'une v.a. de densité g, stockée dans x
    return x

def simul_f():
    loop = True
    while loop:
        x = simul_g()
        u = rd.random()
        if c*u*g(x) < f(x):
            loop = False
    return x
```

La méthode du rejet est exacte, et nécessite un nombre d'itérations qui décroît exponentiellement vite (géométrique). Cependant elle peut être assez inefficace si la fonction g et le réel c sont mal choisis. En pratique on essayera de trouver une fonction g proche de f, au sens où c, qui est un majorant du ratio $\frac{f}{g}$, doit pouvoir être choisi le plus petit possible. Il y a plusieurs manières de trouver g et c, selon la situation, et plusieurs façons d'améliorer l'algorithme. On renvoie pour ça au livre de Luc Devroye (ou autre). L'exercice suivant montre comment on peut simuler une loi normale à partir d'une loi de Laplace.

Exercise 3.4 (Simulation d'une loi normale)**.** *On souhaite simuler une réalisation d'une v.a. de loi normale $\mathcal{N}(0,1)$, de densité notée f, par la méthode du rejet.*

- *Montrer qu'on peut majorer la densité f par une densité exponentielle (préciser le paramètre ainsi qu'une constante c efficace.*
- *La majoration par une loi exponentielle à l'avantage de pouvoir être simulée facilement, mais est très mauvaise pour les abscisses négatives. Proposer une meilleure majoration et un algorithme de simulation (utiliser le lemme de simulation des mélanges de loi). Donner la constante c dans ce cas-ci.*
- *Compléter l'algorithme pour simuler la loi normale.*

Correction. On souhaite simuler une réalisation d'une v.a. de loi normale $\mathcal{N}(0,1)$, de densité notée f. Il s'agit de majorer la densité de la loi normale, donc de minorer $\frac{x^2}{2}$. En complétant le carré on obtient $\frac{x^2}{2} + \frac{1}{2} - x = \frac{1}{2}(x-1)^2 \geq 0$ et donc $\frac{x^2}{2} \geq x - \frac{1}{2}$, puis $f(x) \leq (2\pi)^{-\frac{1}{2}} e^{\frac{1}{2}-x} = cg(x)$ avec $g(x) = e^{-x}$ qui est la densité d'une loi exponentielle de paramètre 1 que l'on sait simuler, et $c = \sqrt{\frac{e}{2\pi}}$. C'est une majoration efficace pour les x positifs, mais très mauvaise pour les x négatifs... En revanche on peut remarquer que rien ne change dans l'argument si on remplace x par sa valeur absolue, et on obtient $f(x) \leq cg(x)$ avec $c = \sqrt{\frac{2e}{\pi}}$ et $g(x) = \frac{1}{2}e^{-|x|}$, qui est la densité d'une loi de Laplace de moyenne nulle et de diversité 1 (variance 2). On peut simuler une v.a. de Laplace à partir d'une v.a. exponentielle en jetant une pièce : si c'est pile, on garde la réalisation, si c'est face, on prend son opposé (c'est un cas de mélange de lois, voir la partie suivante). L'algorithme suivant met en œuvre la méthode du rejet pour simuler une v.a. $\mathcal{N}(0,1)$.

```
def simul_laplace():
    u = rd.random()
    x = -np.log(u)
    u = rd.random()
    if u < 1/2:
        x = -x
    return x

def rejet_gaussienne():
    loop = True
    while loop:
        x = simul_g()
        u = rd.random()
        if (abs(x)-1)^2 < -2*log(u):
            loop = False
    return x
```

La condition pour boucler a été simplifiée, la condition initiale $\frac{u}{\sqrt{2\pi}} e^{\frac{1}{2}-|x|} \geq \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ se réduisant à $(|x|-1)^2 \geq -2\ln u$. □

4 Simulation d'un couple de variables aléatoires, méthode de Box-Muller

Mélange de lois. On souhaite simuler une réalisation d'un couple (X,Y) de variables aléatoires de loi jointe $\mathbb{P}_{(X,Y)}$. Le lemme élémentaire suivant permet de simplifier le problème dans le cas de ce qu'on appelle un mélange de lois.

Lemma 4.1. *Si, X suit la loi marginale $\mathbb{P}_X$ et qu'indépendamment, Y suit la loi conditionnelle $\mathbb{P}_{Y|X}$, alors le couple (X,Y) suit la loi jointe $\mathbb{P}_{(X,Y)}$.*

Ainsi pour simuler une réalisation du couple (X,Y), il suffit de simuler une réalisation x d'une loi $\mathbb{P}_X$, puis une réalisation de la loi $\mathbb{P}_{Y|X=x}$. On a déjà vu un exemple avec la loi de Laplace, en voici un autre (tiré du poly de Chevallier-Rossignol-Coquille) : pour simuler une v.a. X de densité $f_X(x) = p\mathbb{1}_{[1,2]} + (1-p)\mathbb{1}_{[0,1]}$, où $p \in [0,1]$, on simule le couple (i,X) où $i \sim \mathcal{B}(p)$ et $\mathbb{P}_{X|i} = \mathcal{U}([i,i+1])$ pour $i = 0,1$.

Exercise 4.2 (Inversion sur $\mathbb{R}^d$)**.** *Montrer ce résultat (tiré du livre de Bercu et Chafaï) : soit μ une loi de proba sur $\mathbb{R}^d$. Alors il existe une fonction mesurable (pour Lebesgue) f_μ continue p.p. telle que $f_\mu(U_1,...,U_d)$ est de loi μ si $U_1,...,U_d$ sont i.i.d. de loi $\mathcal{U}([0,1])$.*

Correction. On raisonne par récurrence sur d. Pour $d=1$ c'est la méthode d'inversion. Pour d quelconque, soit donc $(X_1,...,X_d)$ un vecteur aléatoire de loi μ. D'après le lemme 4.1 il suffit de résoudre la question pour la loi marginale $\mathbb{P}_{X_d}$ ainsi que la loi conditionnelle $\mathbb{P}_{(X_1,...,X_{d-1})|X_d}$. Pour la seconde on utilise l'hypothèse de récurrence, quant à la première on utilise la méthode d'inversion, ce qui prouve le résultat. □

Simulation d'un couple de Gaussiennes, méthodes de type Box-Muller. Il existe plusieurs méthodes pour simuler des familles de Gaussiennes i.i.d. Nous présentons ici les méthodes dites de Box-Muller et de Marsaglia.

Theorem 4.3. *Soient U et V deux variables i.i.d. uniformes sur $[0,1]$. On note $R = \sqrt{-2\log U}$ et $\Theta = 2\pi V$. Alors les variables*

$$X = R\cos\Theta \quad ; \quad Y = R\sin\Theta$$

forment un couple de Gaussiennes centrées réduites i.i.d.

Proof. Calculons dans un premier temps la densité de R. On a

$$\mathbb{P}(R \le t) = \mathbb{P}(U \ge e^{-t^2/2}) = 1 - \mathbb{P}(U \le e^{-t^2/2}) = (1-e^{-t^2/2})\mathbb{1}_{t\ge 0}$$

Ainsi $f_R(t) = te^{-t^2/2}$ pour $t \ge 0$. La densité de Θ est quant à elle constante égale à $f_\Theta(t) = 1/2\pi$ pour $0 \le t \le 2\pi$. Les variables R et Θ sont bien-sûr indépendantes et donc

$$f_{(R,\Theta)}(s,t) = f_R(s)f_\Theta(t) = \frac{s}{2\pi}e^{-\frac{s^2}{2}}.$$

Soit $g : (s,t) \mapsto (\sigma,\tau) = (s\cos t, s\sin t)$ la transformation en coordonnées polaires. On a $(X,Y) = g(R,\Theta)$ et donc

$$f_{(X,Y)}(\sigma,\tau) = \frac{1}{|\det J_g(\sigma,\tau)|} f_{(R,\Theta)} \circ g^{-1}(\sigma,\tau) \tag{4.1}$$

En effet, pour un Borélien quelconque $A \subset [0,\infty[\times[0,2\pi[$, on a

$$\mathbb{P}((X,Y) \in g(A)) = \mathbb{P}((R,\Theta) \in A) = \int_A f_{(R,\Theta)}(s,t)dsdt.$$

Par changement de variable, on voit que cette dernière intégrale vaut

$$\int_{g(A)} f_{(R,\Theta)} \circ g^{-1}(\sigma,\tau)|\det J_{g^{-1}}(\sigma,\tau)|d\sigma d\tau.$$

On a donc

$$\mathbb{P}((X,Y) \in g(A)) = \int_{g(A)} f_{(X,Y)}(\sigma,\tau) d\sigma d\tau$$

avec $f_{(X,Y)}(\sigma,\tau) := f_{(R,\Theta)} \circ g^{-1}(\sigma,\tau) |\det J_{g^{-1}}(\sigma,\tau)|$, ce qui montre bien l'égalité des densités (4.1). On voit facilement que le déterminant de la Jacobienne de g est donné par $\sqrt{\sigma^2+\tau^2} = s$, et ainsi

$$f_{(X,Y)}(\sigma,\tau) = \frac{1}{s} \cdot \frac{s}{2\pi} e^{-s^2/2} = \frac{1}{\sqrt{2\pi}} e^{-\sigma^2/2} \cdot \frac{1}{\sqrt{2\pi}} e^{-\tau^2/2} = f_X(\sigma) f_Y(\tau)$$

ce qui conclut la preuve. □

L'exemple de code suivant permet d'implémenter la méthode de Box-Muller en Python.

```
def box-muller_gen():
    u = rd.random()
    v = rd.random()
    r = np.sqrt(-2*np.log(u))
    theta = 2*pi*v
    return (r*np.cos(theta),r*np.sin(theta))
```

La seconde version de la méthode de Box-Muller, connue sous le nom de méthode de Marsaglia, permet d'éviter l'évaluation des fonctions trigonométriques, au prix de l'implémentation d'une méthode de rejet, ce qui est plus efficace quand la dimension est petite (en pratique pour $d \leq 5$, voir le livre de Devroye pour des estimations en espérance).

Theorem 4.4. *Soit (X_1, X_2) un point uniforme dans le disque unité. On note $S = \sqrt{X_1^2 + X_2^2}$. Alors les variables*

$$N_1 = \sqrt{-4 \log S} \frac{X_1}{S} \quad ; \quad N_2 \sqrt{-4 \log S} \frac{X_2}{S}$$

forment un couple de Gaussiennes centrées réduites i.i.d.

En pratique, on simulera X_1 et X_2 à partir de variables uniformes sur $[-1,1]$ grâce à la méthode du rejet.

Exercise 4.5. *Prouver le théorème.*

Correction. Il suffit de montrer les trois points suivants :

- S^2 suit une loi uniforme sur $[0,1]$
- $\left(\frac{X_1}{S}, \frac{X_2}{S}\right)$ suit une loi uniforme sur le cercle unité
- S et $\left(\frac{X_1}{S}, \frac{X_2}{S}\right)$ sont indépendants.

En effet, si ces points sont vrais alors les mêmes arguments que pour Box-Muller permettent de conclure ("racine d'une exponentielle × point uniforme indépendant sur le cercle = couple de Gaussiennes indépendantes"). Pour simplifier les notations, appelons $\mathbf{X}$ le point (X_1, X_2).

Pour le deuxième point, on sait que $\mathbf{X}$ suit la loi uniforme sur le disque, en particulier, la loi de $\mathbf{X}$ est invariante sous les transformations orthogonales[2]. Soit A une telle transformation. Alors $A\mathbf{X}/||\mathbf{X}||$ a la même loi que $A\mathbf{X}/||A\mathbf{X}||$, qui a la même loi que $\mathbf{X}/||\mathbf{X}||$, ce qui montre que $\mathbf{X}/||\mathbf{X}||$ est également invariant sous les transformations orthogonales. Comme $\mathbf{X}/||\mathbf{X}||$ est de norme 1, sa loi est la loi uniforme sur le cercle unité (s'en convaincre).

[2]Matrice orthogonale : $A^t A = I$, elles sont de déterminant ± 1, et préservent la norme euclidienne. La loi uniforme est invariante par transformation orthogonale (et une loi invariante par transformation orthogonale sur le cercle est uniforme), la loi normale est également invariante par transformation orthogonale.

Pour le premier point, si $r \geq 0$,

$$\mathbb{P}(S^2 \leq r) = \mathbb{P}(S \leq \sqrt{r}) = \int_{||\mathbf{X}|| \leq \sqrt{r}} dx = \int_{y \leq \sqrt{r}} 2y dy = r$$

donc S^2 suit bien la loi uniforme sur $[0,1]$.

Enfin, pour prouver l'indépendance, soit $t \in (0,1]$ et A un arc du cercle unité (il suffit de considérer un arc). On note $l(A)$ la longueur d'arc, α l'angle du secteur C délimité par l'arc. Alors :

$$\begin{aligned}
\mathbb{P}(S \leq t, \mathbf{X}/S \in A) &= \mathbb{P}(\mathbf{X} \in C \cap D(0,t)) \\
&= |C \cap D(0,t)|/\pi \\
&= \frac{1}{2} t^2 \alpha / \pi \\
&= t \cdot l(A)/2\pi \\
&= \mathbb{P}(S \leq t)\mathbb{P}(\mathbf{X}/S \in A)
\end{aligned}$$

Ce qui montre l'indépendance. □

Exercise 4.6. *Mettre en œuvre un programme Python qui simule deux v.a. Gaussiennes i.i.d. en utilisant la méthode de Marsaglia. Et pour des lois $\mathcal{N}(\mu, \sigma^2)$ quelconques ?*

Exercise 4.7 (Méthode de Bell)**.** *Montrer que si (X_1, X_2) est un point uniforme du disque unité, et E une variable exponentielle indépendante, alors les variables*

$$\sqrt{2E}\frac{X_1^2 - X_2^2}{S^2} \quad ; \quad \sqrt{2E}\frac{2X_1X_2}{S^2}$$

forment un couple de Gaussiennes centrées réduites i.i.d. Cette méthode permet d'éviter l'évaluation d'une racine carrée.

Correction. Clairement si U suit la loi uniforme sur $[0,1]$ alors $(\cos 2k\pi U, \sin 2k\pi U)$ suit la loi uniforme sur le cercle unité pour k entier. En appliquant les formules

$$\cos 4\pi U = \cos^2 2\pi U - \sin^2 2\pi U$$

et

$$\sin 4\pi U = 2 \cos 2\pi U \sin 2\pi U$$

On voit que $\frac{1}{S^2}(X_1^2 - X_2^2, 2X_1X_2)$ a la même loi que $(\cos 4\pi U, \sin 4\pi U)$, soit uniforme sur $\partial\mathbb{D}$. On conclut de la même manière que pour la méthode de Marsaglia. □