# Algorithme de Métropolis Hasting Pseudo Marginal : Théorie et implémentation

Enzo Cabaret

# I Introduction aux chaines de Markov

L'algorithme de Métropolis Hasting fait partie des méthodes dites "Markov Chains Monte Carlo" (MCMC), ces méthodes permettent de simuler des lois de probabilité dont on ne connaitrait pas l'intégralité des paramètres. Ces méthodes ont deux aspects, Monte Carlo : il s'agit d'estimer une quantité à partir de tirage aléatoires, Markov Chains: on va chercher à simuler les réalisations d'une chaine de Markov dont la loi limite est notre loi cible. Dans cette première section on donne les éléments de base pour comprendre les aspects des chaines de markovs utiles pour apréhender les MCMC.

On se place dans un espace mesurable $(\mathrm{X}, \mathcal{X})$

**Definition 1**
*On dit que* $P : \mathrm{X} \times \mathcal{X} \to \mathbb{R}^+$ *est un kernel de Markov si* $\forall (x, A) \in \mathrm{X}, \mathcal{X}$ *on a:*

- $\mathrm{X} \ni y \mapsto P(y, A)$ *est* $\mathcal{B}(\mathbb{R}^+)$ *mesurable*
- $\mathcal{X} \ni B \mapsto P(x, B)$ *est une mesure de probabilité sur* $(\mathrm{X}, \mathcal{X})$

**Definition 2**
*Soit* $(X_k)_{k\in\mathbb{N}}$ *une séquence de variables aléatoires sur un même espace* $(\Omega, \mathcal{G}, \mathbb{P})$ *et prenant des valeurs dans* $\mathrm{X}$ *on dit que* $(X_k)_{k\in\mathbb{N}}$ *est une chaine de markov si et seulement si on a:*

- $\forall (k, A) \in \mathbb{N} \times \mathcal{X}$ *on a* $\mathbb{P}(X_{k+1} \in A | X_{0:k}) = P(X_k, A)$ $\mathbb{P}$*-ps*
- $\mathbb{P}(X_0 \in A) = \nu(A)$ *avec* $\nu \in M_1(\mathcal{X})$

## Notations additionnelles

On introduit ici des notations en lien avec les kernels de markovs pour simplifer l'expression des calculs par la suite.

Soit $\mu \in M_1(\mathcal{X})$, $P, Q$ des kernel de markov et $h$ une fontion bornée sur $\mathrm{X}$ on note :
- $\mu P$: $\mathcal{X} \ni A \mapsto \mu P(A) = \int \mu(dx) P(x, A)$
- $PQ$ le kernel de markov: $(x, A) \mapsto \int_{\mathrm{X}} P(x, dy) Q(y, A)$

**Lemma 1**
$(X_k)_{k\in\mathbb{N}}$ *une chaine de markov sur* $(\Omega, \mathcal{G}, \mathbb{P})$ *et qui prend des valeurs dans* $\mathrm{X}$ *avec un kernel* $P$ *et une distribution initiale* $\nu$ *alors* $\forall n \in \mathbb{N}$ *la loi de* $X_{0:n}$ *est* $\nu(dx_0) \prod_{i=1}^{n-1} P(x_i, dx_{i+1})$*. De meme* $X_n$ *suit la loi* $\nu P^n$

**Proof**

*Facile. Pour la deuxième assertion il suffit d'integrer le résultat de la premiere.* ■

# Probabilités invariantes

**Definition 3**
*On appelle probabilité invariante pour le kernel de Markov $P$ toute mesure $\pi$ vérifiant : $\pi = \pi P$*

**Definition 4**
*Soit $\pi$ une mesure de probabilité et $P$ un kernel de markov on dit que $P$ et $\pi$-reversible si*

$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx) \tag{1}$$

**Proposition 2**
*Soit $P$ un kernel de Markov et $\pi$ une probabilité. Si $P$ est $\pi$-reversible alors $\pi$ est invariante pour $P$.*

# II Algorithme d'Acceptation Rejet et Métropolis Hastings

## Rappels des simulations de variables aléatoires

On note ici $X$ une variable aléatoire (v.a.) réele et $F$ sa fonction de répartition : $F(x) := \mathbb{P}(X \leq x)$

On rappelle que $F$ est croissante, continue à droite et limité à gauche.

**Definition 5**
*On appelle inverse généralisée de F ou fontion quantile la fonction définie par :*

$$\forall u \in ]0,1[ \quad F^{-1}(u) = \inf x \in \mathbb{R} \mid F(x) \geq u \tag{2}$$

**Theorem 3**
*Soient $X$ une v.a. réele et $U$ une v.a. de loi uniforme sur $]0,1[$. Alors $F^{-1}(U)$ et $X$ ont la même loi.*

**Proof**
*(si on a le temps de l'écrire)* ∎

## Algorithme d'acceptation rejet

La méthode d'acceptation rejet s'explique simplement. On suppose que l'on sait générer un vecteur aléatoire $M_1$ de $\mathbb{R}^d$ suivant une loi $\mu$. On génère une suite iid $M_1, \dots, M_n, \dots$ en s'arretant au premier d'entre eux qui vérifie une certaine condition$\mathcal{H}_0$. On note $T$ l'indice correspondant. le vecteur aléatoire $M_T$ suit une nouvelle loi $\nu$. Si l'on parvient facilement à simuler $M$ et à verifier la condition $\mathcal{H}_0$ on possède donc une méthode pour simuler une variable suivant la loi $\nu$.

**Proposition 4**
*Soit $(M_n)_{n\in\mathbb{N}}$ une suite de vecteur aléatoires $\Omega \to \mathbb{R}^d$ indépendant de même loi $\mu$. Soit $B \subset \mathbb{R}^d$ un borélien tel que $\mu(B) > 0$. $\forall \omega \in \Omega$ on pose :*

$$T(\omega) \triangleq \inf\{i \in \mathbb{N}*; M_i(\omega) \in B\} \tag{3}$$

*Avec la convention $\inf\varnothing = +\infty$. On définit $M_T : \Omega \to \mathbb{R}^d$ par*

$$M_T(\omega) \triangleq \begin{cases} M_T(\omega) \text{ si } T(\omega) < +\infty \\ 0 \text{ sinon} \end{cases} \tag{4}$$

*Dans ces conditions :*

- *a) $T$ est une va suivant la loi géométrique de paramètre $p = \mu(B)$*
- *b) $M_T$ est un vecteur aléatoire de loi $\nu$ donnée par*

$$\forall A \in \mathrm{Bor}(\mathbb{R}^d), \quad \nu(A) = \frac{\mu(A \cap B)}{\mu(B)} \tag{5}$$

**Proposition 5**
*Soit $f$ une densité de probabilité sur $\mathbb{R}^d$ et $G$ son hypographe. Soit $M = (Z, Y)$ un vecteur aléatoire de $\mathbb{R}^d \times \mathbb{R}$ de loi uniforme sur G. Alors $Z$ a pour densité $f$ par rapport à $\lambda_d$*

**Proposition 6**
*Soit $X : \Omega \to \mathbb{R}^d$ un vecteur aléatoire dont la loi a pour densité $g$ par rapport à $\lambda_d$. Posons*

$$M \triangleq (X, cg(X)U) \tag{6}$$

*où $U$ est une variable aléatoire uniforme sur $[0,1]$ indépendante de $X$ et $c > 0$ une constante. on note $H$ l'hypographe de $cg$. alors $M$ suit la loi uniforme sur $H$.*

Dans le cas d'une variable à densité l'algorithme d'acceptation rejet se décrit de la manière suivante. On suppose que l'on sait simuler un vecteur de densité $g$ et on souhaite simuler un autre vecteur de densité $f$. Supposons que l'on ait $c$ une constante positive telle que $f < cg$. On a $\lambda_{d+1}(G) = \int_{\mathbb{R}^d} f \, \mathrm{d}\lambda_d = 1$ et $\lambda_{d+1}(H) = \int_{\mathbb{R}^d} cg \, \mathrm{d}\lambda_d = c$. On en déduit que $\lambda_{d+1}(H \backslash G) = \int_{\mathbb{R}^d} cg - f \, \mathrm{d}\lambda_d$, ce qui nous permet de considerer que $\lambda_{d+1}(H \backslash G) > 0$ car le cas échéant $f = cg = g$ car $f, g$ sont des densités de probabilité. Par hypothèse on sait simuler $(X_i)_{i \geq 1}, (U)_{i \geq 1}$, la première car c'est un vecteur de densité $g$ la deuxième car elle est de loi uniforme sur $[0, 1]$ en considérant $M_i \triangleq (X_i, g(X_i)U_i)$ qui est uniforme sur $H$ par la proposition 6. Posons:

$$T(\omega) \triangleq \inf\{i \in \mathbb{N}*; M_i(\omega) \in G\} \tag{7}$$

et

$$M_T(\omega) \triangleq \begin{cases} M_T(\omega) & \text{si } T(\omega) < +\infty \\ 0 & \text{sinon} \end{cases} \tag{8}$$

Les propositions 4 et 5 nous permettent alors de conclure que $Z = X_T$ est un vecteur aléatoire de densité $f$

# Algorithme de Métropolis Hasting

Dans les méthodes MCMC nous allons chercher à simuler les réalisations d'une chaine de Markov de kernel $P$ acceptant pour probabilité invariante la loi que l'on cherche à simuler $\pi$. Toute la problématique revient à trouver le kernel $P$ adéquat. L'approche de Métropolis Hasting nous permet de construire ce kernel.

Comme dans la méthode d'acceptation rejet on suppose que l'on dispose d'une densité nous permettant de générer des candidats pour l'état suivant de notre chaine de markov $y$ à l'état courant $x$ que l'on note $q(x, y)$. Si $q$ vérifie la condition de réversibilité (inserer ref) notre objectif est déja atteint. Cependant il n'y a pas de raison que ce soit le cas. Supposons $x, y \in X$ tels que :

$$\pi(x)q(x, y) > \pi(y)q(y, x) \tag{9}$$

Dans ce cas la notre chaine de Markov bouge de x vers y trop souvent on introduit donc une probabilité $\alpha(x, y) < 1$ pour que y soit pris comme état suivant moins fréquement. Cela revient à choisir comme kernel de Markov le kernel de densité.

$$p_{MH}(x, y) = q(x, y)\alpha(x, y) \text{ avec } x \neq y \tag{10}$$

On souhaite donc introduire $\alpha(x, y)$ et $\alpha(y, x)$ tels que l'équation d'equilibre (inserer ref) soit vérifiée cad :

$$\pi(x)q(x, y)\alpha(x, y) = \pi(y)q(y, x)\alpha(y, x) \tag{11}$$

ce qui nous donne l'identité suivante pour définir $\alpha$:

$$\alpha(x, y) = \frac{\pi(y)q(y, x)}{\pi x q x, y}\alpha(y, x) \tag{12}$$

Dans l'algorithme de métropolis hasting on cherche à ce que $\alpha(x, y)$ soit maximale pour une exploration des nouveaux états plus rapides ce qui revient donc à :

$$\alpha(x, y) = \min(\frac{\pi(y)q(y, x)}{\pi x q x, y}, 1) \tag{13}$$

Pour finir la définition du Kernel de Métropolis Hasting on doit considérer le cas ou notre chaine de Markov reste dans le même état car actuellement on se plaçait dans le cas $x \neq y$. En utilisant notre raisonnement précédent la probabilité que la chaine reste dans le même état est :

$$r(x) = 1 - \int_{\mathrm{X}} q(x, y)\alpha(x, y)dy \tag{14}$$

Finallement on définit le kernel de Métropolis Hasting par :

$$P_{MH}(x, y) = q(x, y)\alpha(x, y)dy + \left[1 - \int_{\mathrm{X}} q(x, y)\alpha(x, y)dy\right] \delta_x(dy) \tag{15}$$

Ce qui nous amène à l'implémentation de l'algorithme de Métropolis Hasting :

-Initialiser $x_0$ arbitrairement.
-Pour $i = 0, \dots, n-1$ faire :
  -Tirer $y$ à avec $q(x, \cdot)$ et $u$ avec $\mathcal{U}(0, 1)$
  -Si $u < \alpha(x_i, y)$ alors $x_{i+1} = y$
  -Sinon $x_{i+1} = x_i$
-Fin Pour
-Retourner $x_0, \dots, x_n$

Nous illustrons l'algorithme par un exemple simple :
On se place dans le cas d'une distribution normale centrée réduite perturbée, le problème ici c'est que l'on ne peut pas exprimer la constante de normalisation toutefois l'algorithme de métropolis hasting nous permet quand meme de génerer des tirages.

-fonction cible : $\pi(x) \propto \sin^2(x) \times \sin^2(2x) \times \exp(-x^2/2)$
-kernel : $q : x \mapsto \mathcal{U}(x - \mu, x + \mu)$ avec $\mu > 0$

In [2]:

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.random as npr
import scipy.stats as stats
from scipy.stats import expon, geom, norm
from math import pi
from pylab import *
% matplotlib inline
```

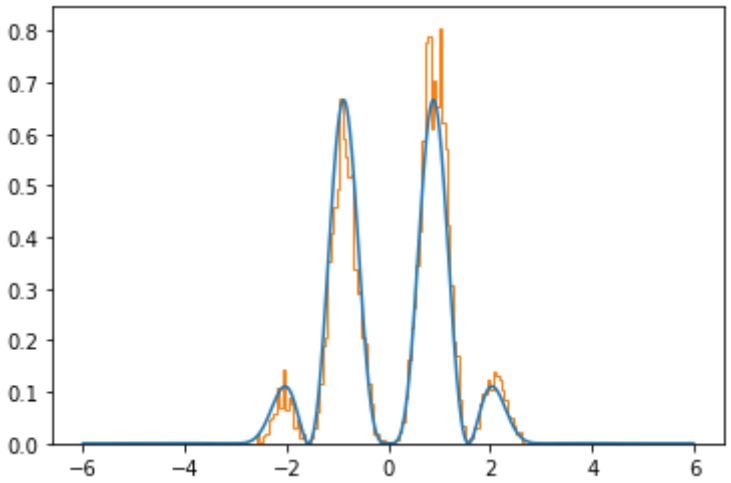
```
UsageError: Line magic function `%` not found.
```

In [3]:

```
import numpy as np
accpeted = []
def pi(x):
    return (np.sin(x)**2)*(np.sin(2*x)**2)*np.exp(-(x**2)/2)

def metropolis_hastings(x_0,n,mu=1):
    X = np.empty(n)
    X[0] = x_0
    for i in range(n-1):
        y = np.random.uniform(X[i]-mu,X[i]+mu)
        u = np.random.uniform(0,1)
        if u < min(pi(y)/pi(X[i]),1):
            X[i+1] = y
        else:
            X[i+1] = X[i]
    return X
```

In [4]:

```
import matplotlib.pyplot as plt

#paramètres
x_0 = 3
n = 10**4


x_MH = metropolis_hastings(x_0,n)


#graphe de la fonction cible :
x = np.arange(-6,6,0.01)
y = pi(x)
y = [i/(sum(y)*0.01) for i in y]
plt.plot(x,y)

#hisogramme des échantillons obtenus par MH
n,bins,patches = plt.hist(x_MH,100,density = True,histtype = 'step')


plt.show()
```



# II. Metropolis Hastings Méthode pseudo-marginales

# Généralisation de l'algorithme MH

Soit $\pi \in M_1(X)$ et $Q$ un noyau de Markov sur $X \times \chi$. Dans la section précédente on a regardé l'algorithme Metropolis-Hastings quand $\pi$ et $Q(x)$ ont toutes les deux des densités par rapport à $\lambda$. Dans cette partie on ne fait pas cette supposition et l'expression de $\alpha^{MH}$, n'est plus disponible. La supposition que nous feront est la suivante :

$$\mu_0(dxdy) = \pi(dx)Q(x, dy) \tag{16}$$

$$\mu_1(dxdy) = \pi(dy)Q(x, dy) \tag{17}$$

**Proposition 7**
*Il existe une fonction* $(x, y) \Longrightarrow r(x, y)$ *tel que* $r(x, y) > 0, \mu_o$ *a.s et* $\forall \quad h \in F_+(X^2)$, $\int h(x, y)\mu_1(dx, dy) = \int h(x, y)r(x, y)\mu_0(dx, dy)$

On montre grâce à cette propriété que ces deux mesures sont équivalentes. Dans ce cas, on remplace dans la version du noyau MH où $\alpha^{MH}$ est donnée dans le **Lemme 5**, par $\alpha(x, y) = r(x, y) \wedge 1$ qui est $\pi$-reversible.

**Lemma 8**
*On assumant la propriété 1, on pose* $\alpha(x, y) = r(x, y)$, *le moyau de Markov est alors :* $P^{MH}_{\langle \pi, Q \rangle} = Q(x, dy)\alpha(x, y) + \overline{\alpha}(x)\delta_x(dy)$ *ou* $\overline{\alpha}(x) = 1 - \int_X Q(x, dy)\alpha(x, y)$ *est* $\pi$*-reversible*

*Idée de la preuve* : Détailler la condition d'équilibre (1).

# Techniques pseudo marginales de Monte Carlo

**Proof**

*On assume dans cette partie que* $\pi$ *et* $Q$ *sont dominées par une mesure commune* $\lambda$ *et on note* $\pi(dx) = \pi(x)\lambda(dx)$ *et* $Q(x, y) = q(x, y)\lambda(dx)$.

*On a normalement besoin (à une constante multipplicative près) d'une expression explicite de la distribution* $\pi(x)\ \forall x \in X$ *comme distribution cible de l'algorithme de Metropolis-Hastings. Néanmoins il peut arriver qu'on ne puisse pas calculer la distribution* $\pi(x)$ *même pas à une constante multiplicative près. Dans ce cas là,* ***on assume alors que l'on peut avoir un estimateur non-biaisé de la distribution*** $\pi(x)$.

*Pour obtenir cet estimateur non-biaisé, on tire* $W \sim R(x, dw)$ *où* $R$ *est un noyau de Markov de* $X \longrightarrow \mathbb{R}^+$ *tel que la condition d'estimateur non-biaisé soit respectée.* ■

On obtient alors l'algorithme MH Pseudo marginal suivant :

**Input :** $n$

**Output :** $X_0, \ldots, X_n$

À $t = 0$, tirage de $X_0$ selon une distribution arbitraire $W_0 \sim R(X_0, .)$

**for** $t \leftarrow 0$ **to** $n - 1$ **do**

- Tirage de $\tilde{X}_{t+1} \sim Q(X_t, .)$ et après $\tilde{W}_{t+1} \sim \tilde{R}(X_{t+1}, .)$

- On pose $(X_{t+1}, W_{t+1}) = \begin{cases} (\tilde{X}_{t+1}, \tilde{W}_{t+1}) \text{ avec prob. } \frac{\tilde{W}_{t+1} q(\tilde{X}_{t+1}, X_t)}{W_t q(X_t, \tilde{X}_{t+1})} \wedge 1 \\ (X_t, W_t) \text{ avec prob. } 1 - \frac{\tilde{W}_{t+1} q(\tilde{X}_{t+1}, X_t)}{W_t q(X_t, \tilde{X}_{t+1})} \wedge 1 \end{cases}$

Cette version de l'algorithme de Metropolis-Hastings est très proche de la version du chapitre précédent mais en remplaçant $\pi(x)$ par un estimateur non-biaisé. On justifie l'utilisation des techniques Pseudo-marginales de Monte-Carlo en montrant que c'est en fait un algorithme MH généralisé en considérant la chaîne de Markov $(\bar{X}_k)_{k\in\mathbb{N}} = (X_k, W_k)_{k\in\mathbb{N}}$ sur un nouvel espace $\bar{X} = X \times \mathbb{R}_*^+$, avec une nouvelle distribution cible $\Pi(d\bar{x}) = \Pi(dxdw) = wR(x, dw)\lambda(dx).\backslash$ On pose $\mu_0(d\bar{x}, d\bar{x'}) = wR(x, dw)\lambda(dx)Q(x, dx')R(x', dw')$ et $\mu_1(d\bar{x}, d\bar{x'}) = w' R(x', dw')\lambda(dx)Q(x', dx)R(x, dw)$ pour satisfaire la propriété 1.\ On écrit ensuite $Q(x, dy) = q(x, y)\lambda(dy)$, on obtient $\forall h \in F_X(\bar{X}^2)$ :

$$\$ \int_{\bar{X}^2} h(\bar{x}, \bar{x'})\mu_1(d\bar{x'}, d\bar{x'}) = \int_{\bar{X}^2} h(\bar{x}, \bar{x'})r(\bar{x}, \bar{x'})\mu_0(dx, dy) \$ \tag{18}$$

Comme $r > 0$, on applique le lemme 1 avec $\alpha(\bar{x}, \bar{x'}) = r(\bar{x}, \bar{x'}) \wedge 1$ on a que $P^{MH}_{\langle \pi, Q \rangle}(\bar{x}, d\bar{x'})$ est $\Pi$-reversible. En appliquant l'algorithme 2 on a que $P^{MH}_{\langle \pi, Q \rangle}(\bar{x}, d\bar{x'})$ est un noyau de Markov.\ **En conclusion :** $(\bar{X}_k)_{k \in \mathbb{N}} = (X_k, W_k)_{k \in \mathbb{N}}$ produite par l'algorithme MH Pseudo marginal où la distribution cible est $\Pi$ admet $\pi$ comme distribution marginale pour $X_k$, $(X_k)_{k \in \mathbb{N}}$ n'est plus une chaîne de Markov mais $(\bar{X}_k)_{k \in \mathbb{N}}$ oui.

# Méthode pseudo marginale avec un mélange de k gaussiennes

## Mélange de gaussienne avec un estimateur sans biais

In [19]:

```
#gaussian mixture of 5 gaussian
mu = [-3,0,3]
sig = [0.5,0.5,0.5]

def pi(x):
    result = 0
    for m,s in zip(mu,sig):
        result+= np.exp(-((x-m)**2)/(2*s**2))
    return result

#estimateur sans biais de pi(x)
def unbiaised_estimator(pi,x):
    return np.random.exponential()*pi(x)

#pseudo marginal metropolis hastings
def metropolis_hastings(x_0,n,pi_hat,mu=2.5):
    X = np.empty(n)
    X[0] = x_0
    for i in range(n-1):
        y = np.random.uniform(X[i]-mu,X[i]+mu)
        u = np.random.uniform(0,1)
        if u < min(pi_hat(y)/pi_hat(X[i]),1):
            X[i+1] = y
        else:
            X[i+1] = X[i]
    return X
```

In [20]:

```
pi_hat = lambda x: unbiaised_estimator(pi,x)

x_MH = metropolis_hastings(0,10**4,pi_hat)

#graphe de la fonction cible :
x = np.arange(-6,6,0.01)
y = pi(x)
y = [i/(sum(y)*0.01) for i in y]
plt.plot(x,y)

#hisogramme des échantillons obtenus par MH
n,bins,patches = plt.hist(x_MH,100,density = True,histtype = 'step')


plt.show()
```