Statistiques Bayésiennes

Hicham Janati

hjanati@insea.ac.ma

INSEA

1. Pourquoi Monte-Carlo ? (Exemple de modèle hiérarchique)
2. Introduction à la méthode Monte-Carlo (historique, PRNG)
3. Algorithmes de simulation i.i.d (PRNG, transformation, rejet)
4. Méthodes MCMC (Gibbs, Metropolis)
5. Diagonstics de convergence MCMC
6. Méthodes MCMC avancées (Langevin, HMC, NUTS)

INSEA

Pourquoi les

II. Méthodes de Monte-Carlo

?

On souhaite modéliser la fréquence des sinistres d'un ensemble de conducteurs:

TD1

Modèle (vraisemblance)

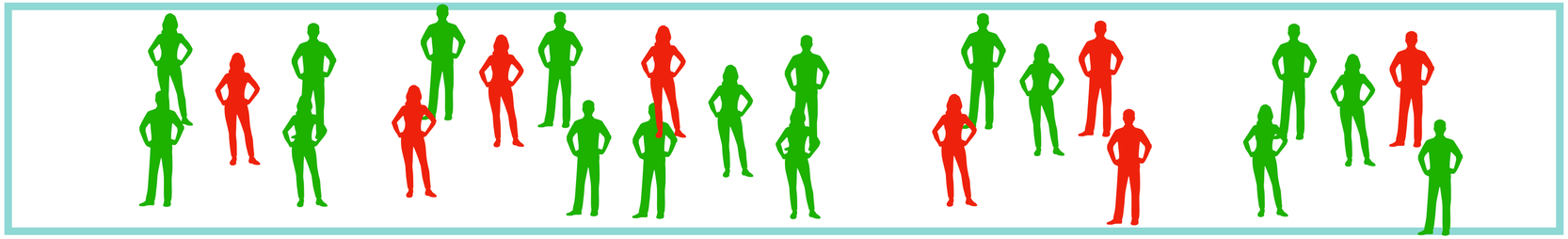
$$\text{Nombre de sinistres par années } N_i|\lambda_i \sim \mathcal{P}(\lambda_i)$$

a priori

$$\lambda_i \sim \text{Gamma}(\alpha, \beta)$$

issu de données historiques sur tous les clients

Or les conducteurs peuvent être bons ou mauvais:



Inconvénients: tous les conducteurs ont la même loi a priori

Sur-estimer le risque des bons conducteurs / sous-estimer le risque des mauvais conducteurs

$$\text{On note } z_i = 1 \text{ (bon) et } z_i = 0 \text{ (mauvais)}$$

On souhaite modéliser la fréquence des sinistres d'un ensemble de conducteurs:

TD1

Modèle (vraisemblance)

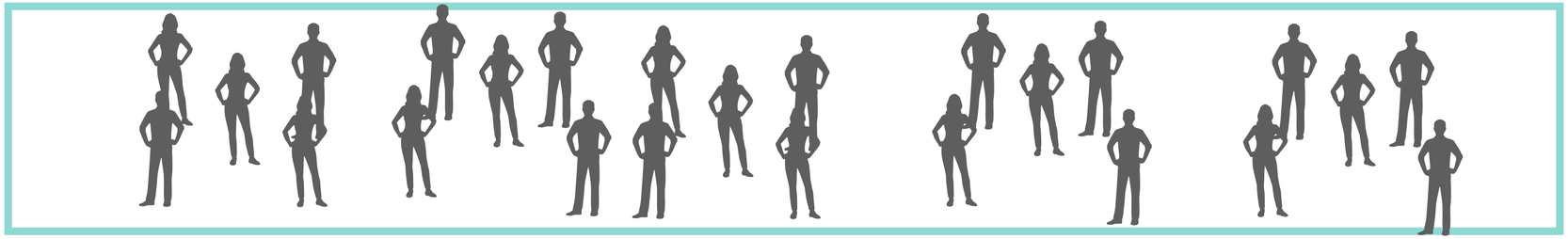
$$\text{Nombre de sinistres par années } N_i|\lambda_i \sim \mathcal{P}(\lambda_i)$$

a priori

$$\lambda_i \sim \text{Gamma}(\alpha, \beta)$$

issu de données historiques sur tous les clients

Or les conducteurs peuvent être bons ou mauvais:



Inconvénients: tous les conducteurs ont la même loi a priori

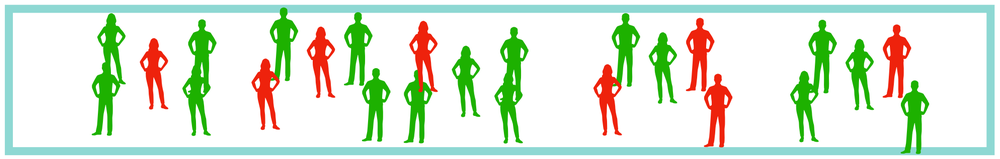
Sur-estimer le risque des bons conducteurs / sous-estimer le risque des mauvais conducteurs

$$\text{On note } z_i = 1 \text{ (bon) et } z_i = 0 \text{ (mauvais)}$$

$$\text{Mais les } z_i \text{ ne sont pas observées...}$$

Comment peut-on adapter le modèle ?

Modèle bayésien hiérarchique



$$N_i|\lambda_i \sim \mathcal{P}(\lambda_i)$$

On note $z_i = 1$ (bon) et $z_i = 0$ (mauvais)

Comment peut-on modéliser la variable z_i ?

$$z_i|p \sim \text{Bernoulli}(p)$$

Lois a priori des bons et des mauvais différentes:

$$\lambda_i|z_i = 0 \sim \text{Gamma}(\alpha_0, \beta_0) \text{ et } \lambda_i|z_i = 1 \sim \text{Gamma}(\alpha_1, \beta_1)$$

Comment choisir les paramètres ?

Avec des *hyperpriors*:

$$\alpha_0, \alpha_1, \beta_0, \beta_1 \sim \text{Gamma}(\tau, \delta)$$

$$p \sim \beta(a, b)$$

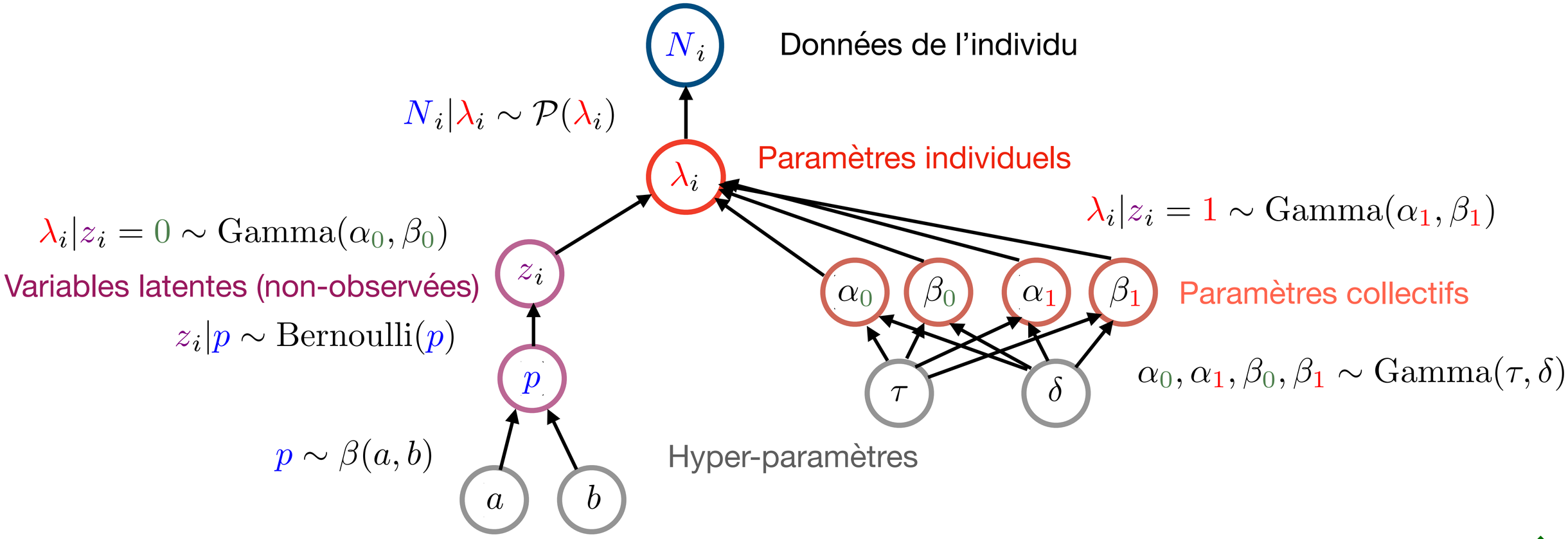
L'hyperprior Gamma "lie" les deux groupes

Comment choisir les hyper-paramètres ?

Ceci est un exemple de Modèle bayésien hiérarchique

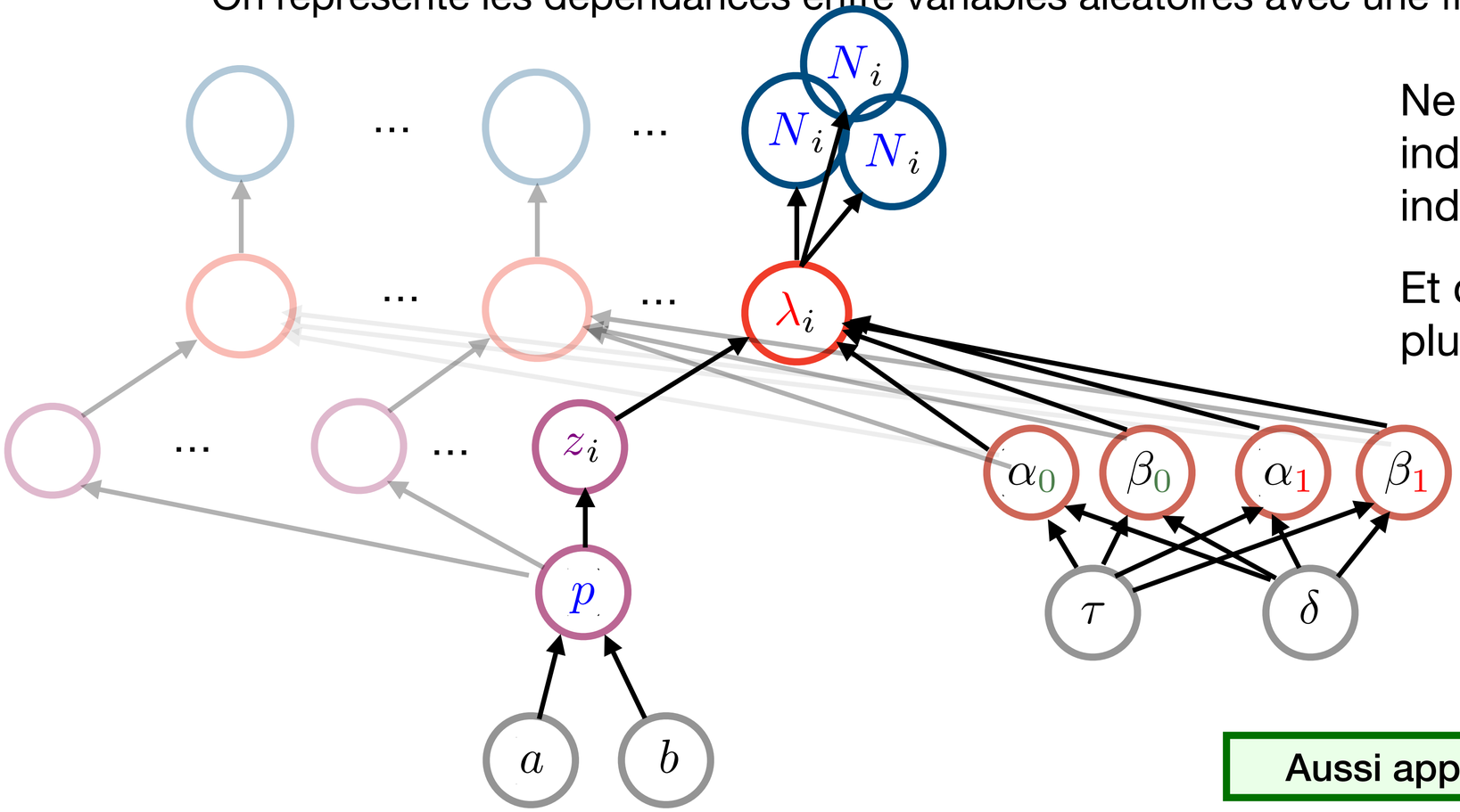
Modèle bayésien hiérarchique comme un PGM (*Probabilistic graphical model*)

On représente les dépendances entre variables aléatoires avec une flèche:



Modèle bayésien hiérarchique comme un PGM (*Probabilistic graphical model*)

On représente les dépendances entre variables aléatoires avec une flèche:



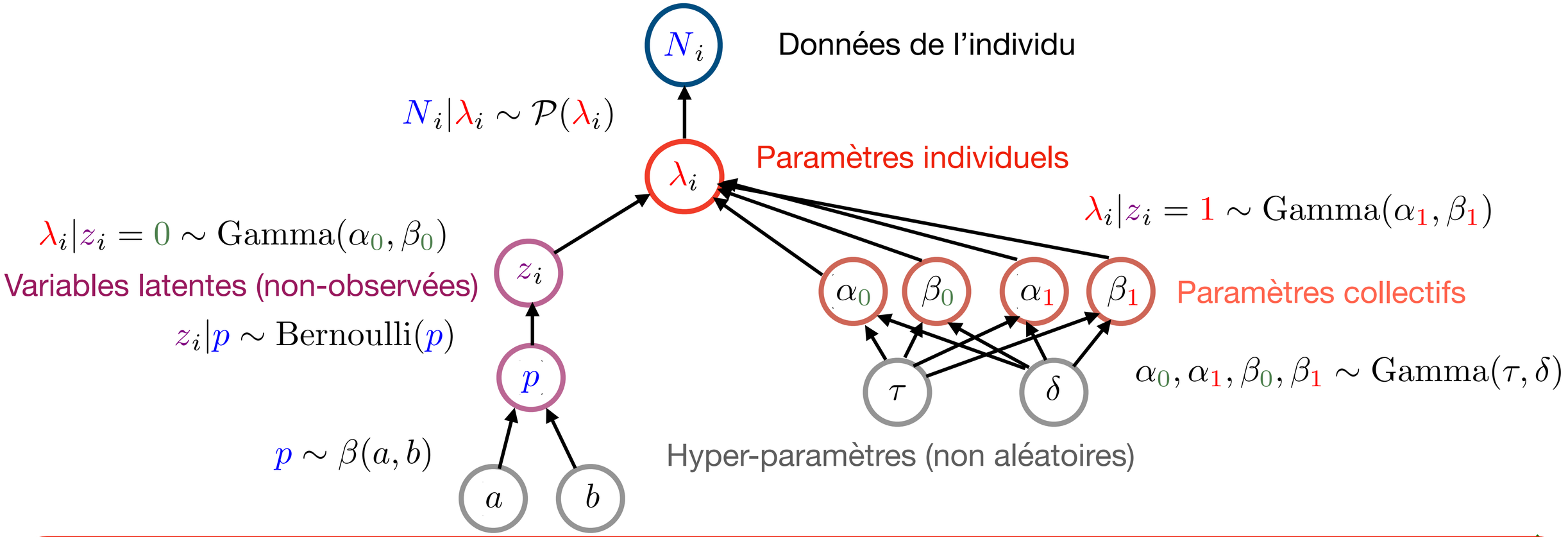
Ne pas oublier que chaque individu a son paramètre individuel…

Et chaque individu peut avoir plusieurs observations …

Aussi appelé un “Réseau bayésien”

Modèle bayésien hiérarchique comme un PGM (*Probabilistic graphical model*)

On représente les dépendances entre variables aléatoires avec une flèche:



Question: trouver la loi a posteriori jointe de tous les paramètres $(\{\lambda_i\}_{i=1}^n, \{z_i\}_{i=1}^n, p, \alpha_0, \alpha_1, \beta_0, \beta_1) \,|\{N_i\}_{i=1}^n$.

La loi jointe $(\{\lambda_i\}_{i=1}^n, \{z_i\}_{i=1}^n, p, \alpha_0, \alpha_1, \beta_0, \beta_1) \,|\{N_i\}_{i=1}^n$ est

$$\propto \prod_{i=1}^{n} \left\{ \mathbb{P}\Big(N_i \mid \lambda_i\Big)\, \mathbb{P}\Big(\lambda_i \mid z_i,\, \alpha_0,\, \beta_0,\, \alpha_1,\, \beta_1\Big)\, \mathbb{P}\Big(z_i \mid p\Big) \right\} \mathbb{P}(p)\, \mathbb{P}(\alpha_0)\, \mathbb{P}(\beta_0)\, \mathbb{P}(\alpha_1)\, \mathbb{P}(\beta_1).$$

$$\propto \prod_{i=1}^{n} \left\{ \frac{{\lambda_i}^{N_i} e^{-\lambda_i}}{N_i!} \left(\frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} {\lambda_i}^{\alpha_0 - 1} e^{-\beta_0 \lambda_i} \right)^{1-z_i} \left(\frac{\beta_1^{\alpha_1}}{\Gamma(\alpha_1)} {\lambda_i}^{\alpha_1 - 1} e^{-\beta_1 \lambda_i} \right)^{z_i} p^{z_i} (1-p)^{1-z_i} \right\} p^{a-1} (1-p)^{b-1} \left[\alpha_0^{\tau-1} e^{-\delta\, \alpha_0} \beta_0^{\tau-1} e^{-\delta\, \beta_0} \alpha_1^{\tau'-1} e^{-\delta'\, \alpha_1} \beta_1^{\tau'-1} e^{-\delta'\, \beta_1} \right].$$

Comment obtenir les lois a posteriori d'intérêt: $\lambda_j | \{N_i\}_{i=1}^n$? $z_j | \{N_i\}_{i=1}^n$?

Il faut marginaliser: intégrer la loi jointe par rapport à toutes les autres variables: impossible analytiquement !

Solution: Obtenir des échantillons "simulés" issus de la loi a posteriori et construire des estimateurs empiriques

Et pour cela on utilise les méthodes de simulation "Monte-Carlo"

INSEA

Pourquoi les

2. Méthodes de Monte-Carlo

?

1. Modèles Bayésiens: Simuler à partir d'une loi a posteriori intractable
2. Résoudre des équations différentielles / intégrales en Physique / Finance / Météo
3. Incontournables en IA / ML (Bayesian Optimization, Generative AI, Reinforcement learning …)

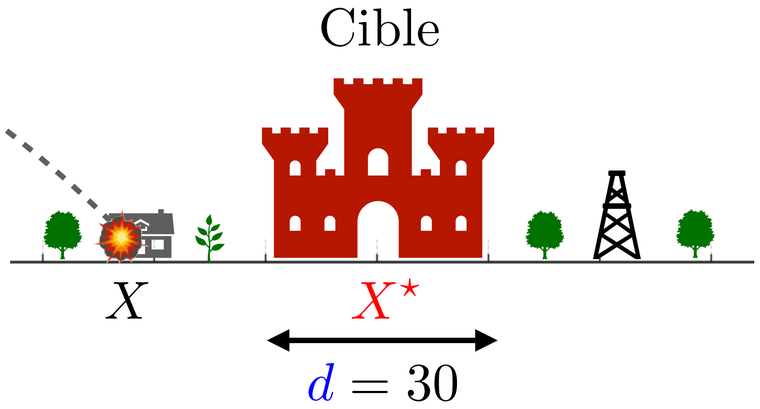
INSEA

II Méthodes de Monte-Carlo

1. Introduction

Calcul d'intégrale pre-WWII

Nous sommes en 1902



On note X et $X^\star$ les coordonnées du projectile et de la cible.

On modélise $X \sim \mathcal{N}(X^\star, \sigma^2)$ avec $\sigma = 50$.

On note $d = 30$ le diamètre de la cible.

Pour détruire la cible, il faut en moins 40 impacts directs, combien de projectiles devez-vous lancer ?

On pose $Z \overset{\text{def}}{=} \frac{X - X^\star}{\sigma} \sim \mathcal{N}(0,1)$.

$$\mathbb{P}(\text{Impact}) = \mathbb{P}(|X - X^\star| \leq d) = \mathbb{P}(|Z| \leq \tfrac{d}{\sigma}) = \Phi(\tfrac{d}{\sigma}) - \Phi(-\tfrac{d}{\sigma})$$

$$= 2\Phi(\tfrac{d}{\sigma}) - 1 = 2\Phi(\tfrac{30}{50}) - 1 \approx 0.45$$

$$\mathbb{P}(\widehat{\text{Impact}}) = \frac{40}{N} \Rightarrow N = \frac{40}{\mathbb{P}(\widehat{\text{Impact}})} \approx \frac{40}{0.45} \approx 89$$

Comment on calcule $\Phi(\frac{30}{50})$ en 1902 ?

INSEA

Calcul d'intégrale pre-WWII

Comment on calcule $\Phi(\frac{30}{50})$ en 1902 ?

$$\Phi(x) \stackrel{\text{def}}{=} \mathbb{P}(Z \leq x) = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{x} e^{-\frac{x^2}{2\sigma^2}} \, \mathrm{d}x$$

Comment calculer π ?

1. $\pi = \arctan(1)$ et $\arctan(x) = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots$

La racine ?

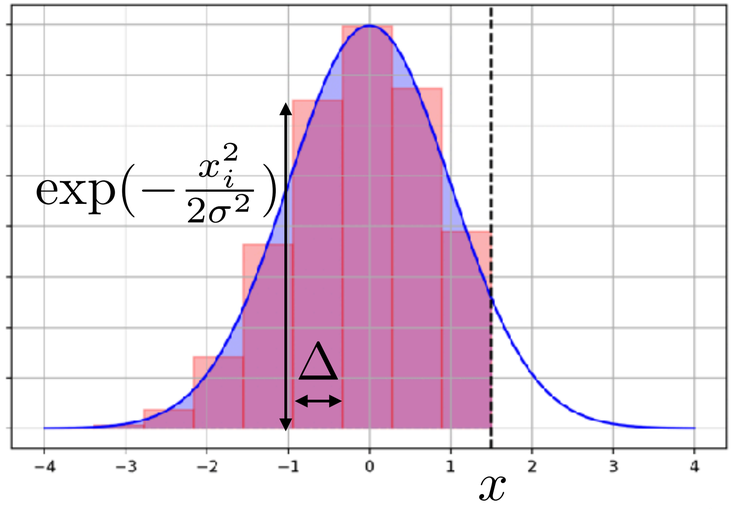
2. La suite de Newton $a_{n+1} = \frac{1}{2}\left(a_n + \frac{A}{a_n}\right)$ converge vers $\sqrt{A}$.

l'intégrale ?

3. Approcher l'intégrale avec des rectangles $\approx \sum_{i=1}^{10} \Delta \exp(-\frac{x_i^2}{2\sigma^2})$

l'exponentielle ?

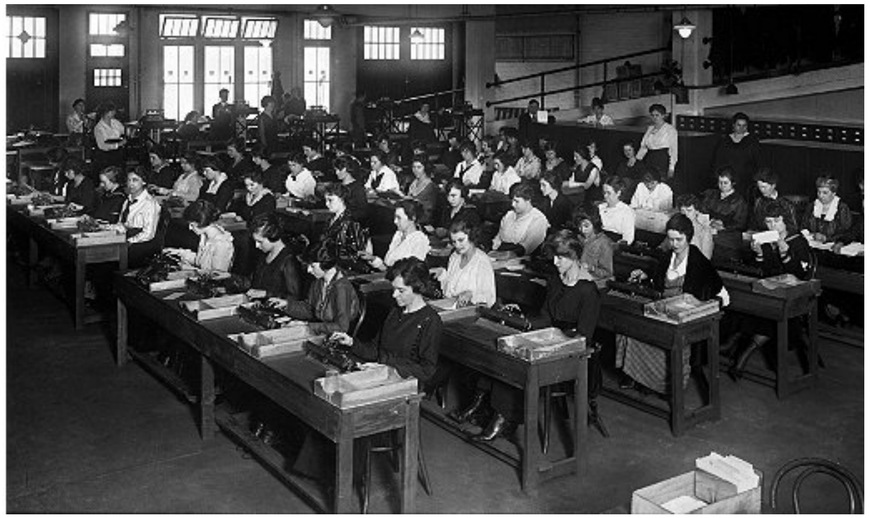
4. Calculer les exp: $\exp(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$



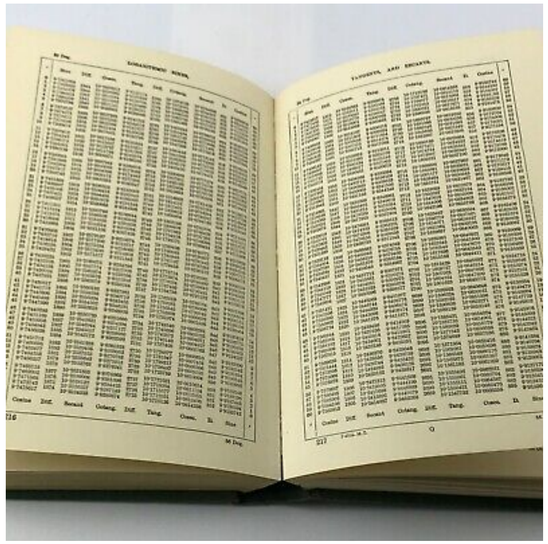
Beaucoup de calculs à la main…

Calcul d'intégrale pre-WWII

"Un *computer*" à l'époque était un travail:

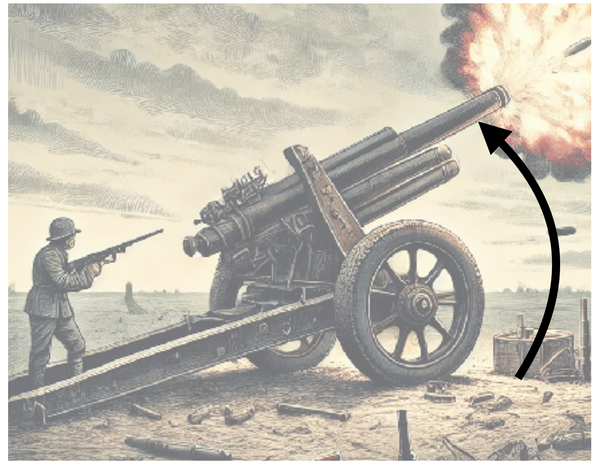


1] produire des livres avec des tables de calcul précises cosinus, sinus, exp, logarithme, racines, puissances, les constantes, intégrales…



2] Résoudre manuellement (algorithmes) des équations différentielles en mécanique (trajectoires de projectiles) en utilisant des calculatrices mécaniques de bureau

3] Produire des tables de calculs des trajectoires en fonction de l'angles de tir, vitesse et direction du vent …



INSEA

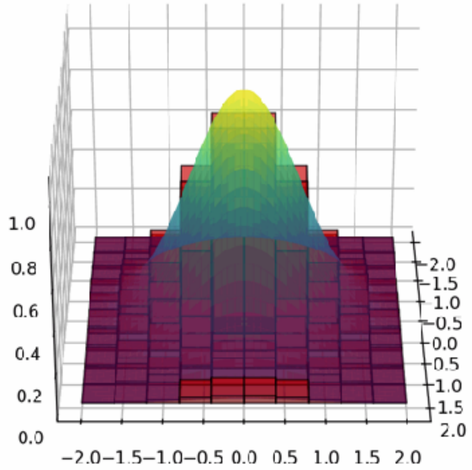
ENIAC et méthodes de quadrature

1946: Premier ordinateur électronique **ENIAC**
(Electronic Numerical Integrator and Computer)

“Human computers” ENIAC

Calcul d’une trajectoire: 40 heures ⟶ 30 secondes

Et si on utilisait ENIAC pour calculer des intégrales doubles ? triples ?



Applications: Prévision météo, Finance, Physique nucléaire …

La première prévision météo (ENIAC) du lendemain a nécessité 24h de calcul !

Les méthodes de **quadrature** sont lentes en **grande** dimension

L’idée “Monte-Carlo”

Stan Ulam: Physicien / mathématicien membre du Manhattan project (bombe atomique)

À l’hôpital, essaie de calculer la probabilité de gagner au solitaire en analysant toutes les combinaisons de cartes possibles

“Au lieu d’étudier tous les cas possibles, je peux jouer 1000 parties et calculer le pourcentage de parties gagnées…”

*“Et si **ENIAC** peut faire ça à ma place …”*

ATTENDS ! Au lieu de calculer toutes les trajectoires de neutrons possibles avec les quadratures, je pourrais les *simuler* et avoir une approximation de:

$$\int_0^{\infty} \int_{4\pi} \left[\sum_{\ell=0}^{L} \frac{2\ell+1}{4\pi} \left(\sum_{m=1}^{M} A_{\ell,m}(\mathbf{r}) \, \exp\left(-B_{\ell,m}(\mathbf{r}) \, |E' - E|\right) \right) P_{\ell}\left(\boldsymbol{\Omega}' \cdot \boldsymbol{\Omega}\right) \right] \phi(\mathbf{r}, \boldsymbol{\Omega}', E') \, \mathrm{d}\boldsymbol{\Omega}' \, \mathrm{d}E'.$$

L’ordinateur était toujours considéré comme un “calculateur déterministe”. L’utiliser pour “simuler” des nombres selon des “probabilités” était une idée révolutionnaire

INSEA

L'idée "Monte-Carlo"

Ulam, Metropolis et Von Neumann proposent une méthode de calcul d'une intégrale quelconque de la forme:

$$\int_0^1 g(x)\mathrm{d}x$$

Comment peut-on écrire cette intégrale comme une espérance ?

$$\int_0^1 g(x)\mathrm{d}x = \mathbb{E}_{X\sim\mathcal{U}([0,1])}(g(X))$$

Comment approximer cette espérance ?

$$\approx \frac{1}{N}\sum_{i=1}^{N} g(X_i) \qquad \text{Avec } X_1,\ldots,X_N \text{ i.i.d } \sim\mathcal{U}([0,1])$$

Metropolis propose de nommer leur article "*The Monte-Carlo Method*" inspiré par l'oncle d'Ulam qui était accro au célèbre *casino de Monte-Carlo*.

$$\text{Comment générer } X_1,\ldots,X_N \text{ i.i.d } \sim\mathcal{U}([0,1]) \text{ ?}$$

Le premier PRNG

Comment générer $x \sim \mathcal{U}([0,1])$?

Première idée de Newman:

1. *Choisir un nombre à 10 chiffres: 4924748149*
2. *Calculer son carré: 24253144331078926201*
3. *Prendre 10 chiffres au milieu: 24253144331078926201*
4. *Diviser par 10^{10}: x = 1443310789 / 10000000000*
5. *Pour un second chiffre, calculer 1443310789^2 … et ainsi de suite*

La suite à 10 chiffres est une suite périodique (déterministe) avec un comportement qui "semble" aléatoire

On appelle ce type d'algorithme: PRNG (Pseudo-random numbers generators)

INSEA

Les PRNG aujourd'hui

Les PRNG d'aujourd'hui:

1. Se basent sur un mélange F d'opérations (addition, multiplication, reste de division, shift de bits ...)
2. Définissent toujours une suite d'entiers périodique $x_{n+1} = F(x_n)$ avec $x_i \in [0, M]$, avec M très grand et une période très très grande (même avec 10^{12} chiffres / seconde, il faut 10^{18} années pour voir toute la séquence).
3. Définir $u_i = \frac{x_i}{M}$ permet d'obtenir des nombres qui "semblent" suivre une loi uniforme $\mathcal{U}([0, 1])$.
4. Il existe plusieurs tests qui permettent de mesurer la qualité statistique d'un PRNG (corrélation, biais de sélection...)
5. On appelle x_0 *la graine (seed)* du PRNG: si on la connait, on peut déterminer toute la suite.
6. Fixer la seed permet de garantir la reproducibilité d'un programme avec des nombres (pseudo) aléatoires.
7. Les PRNG les plus connus sont:
   (a) *Mersenne Twister (1997)*: utilisé par défaut `Numpy <= 1.16`. np.random.RandomState
   (b) *PCG64 (2014)*: adopté par `Numpy >= 1.17` en 2019. np.random.default_rng
8. Ne doivent pas être utilisés en cryptographie (générer des clés privées / mots de passe)

INSEA

Algorithmes de simulation

Les PRNG permettent de générer des échantillons i.i.d $X_1, \dots, X_n \sim \mathcal{U}([0,1])$.

Qu'en est-il de

Basés sur des échantillons i.i.d

$X_1, \dots, X_n \sim \mathcal{U}([a,b])$?

$X_1, \dots, X_n \sim \mathcal{B}(p)$?

$X_1, \dots, X_n \sim Bin(p,n)$?

$X_1, \dots, X_n \sim \mathcal{E}xp(\lambda)$?

$X_1, \dots, X_n \sim \text{Gamma}(k,\theta)\ k \in \mathbb{N}_\star$?

Résultats théorie des probabilités (au tableau / Python)

$X_1, \dots, X_n \sim \mathcal{N}(\mu, \sigma^2)$? Box-Muller, Marsaglia, Ziggurat

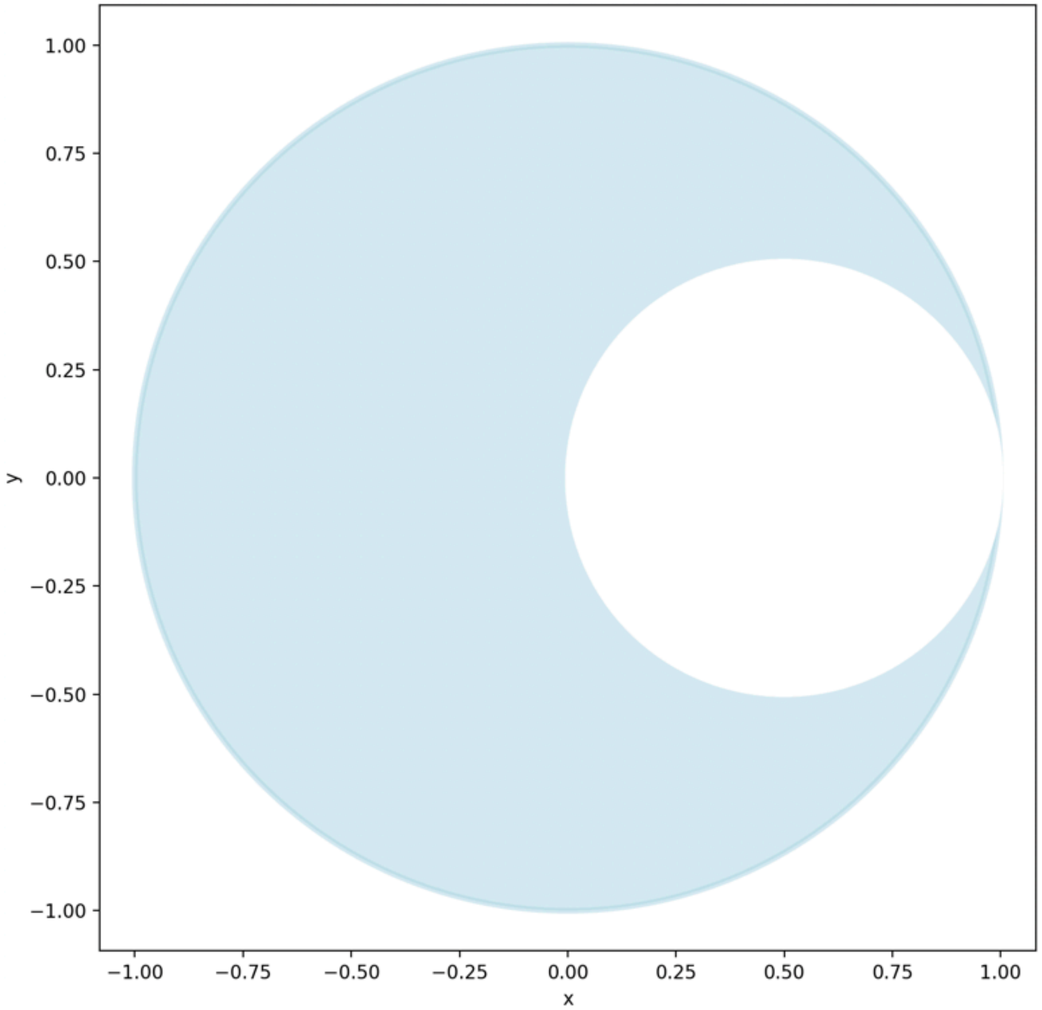
$X_1, \dots, X_n \sim \frac{f}{\int f}$? Rejection sampling

Basés sur des échantillons non i.i.d

$X_1, \dots, X_n \sim \propto g$? Algorithmes MCMC (Markov-Chain Monte-Carlo)

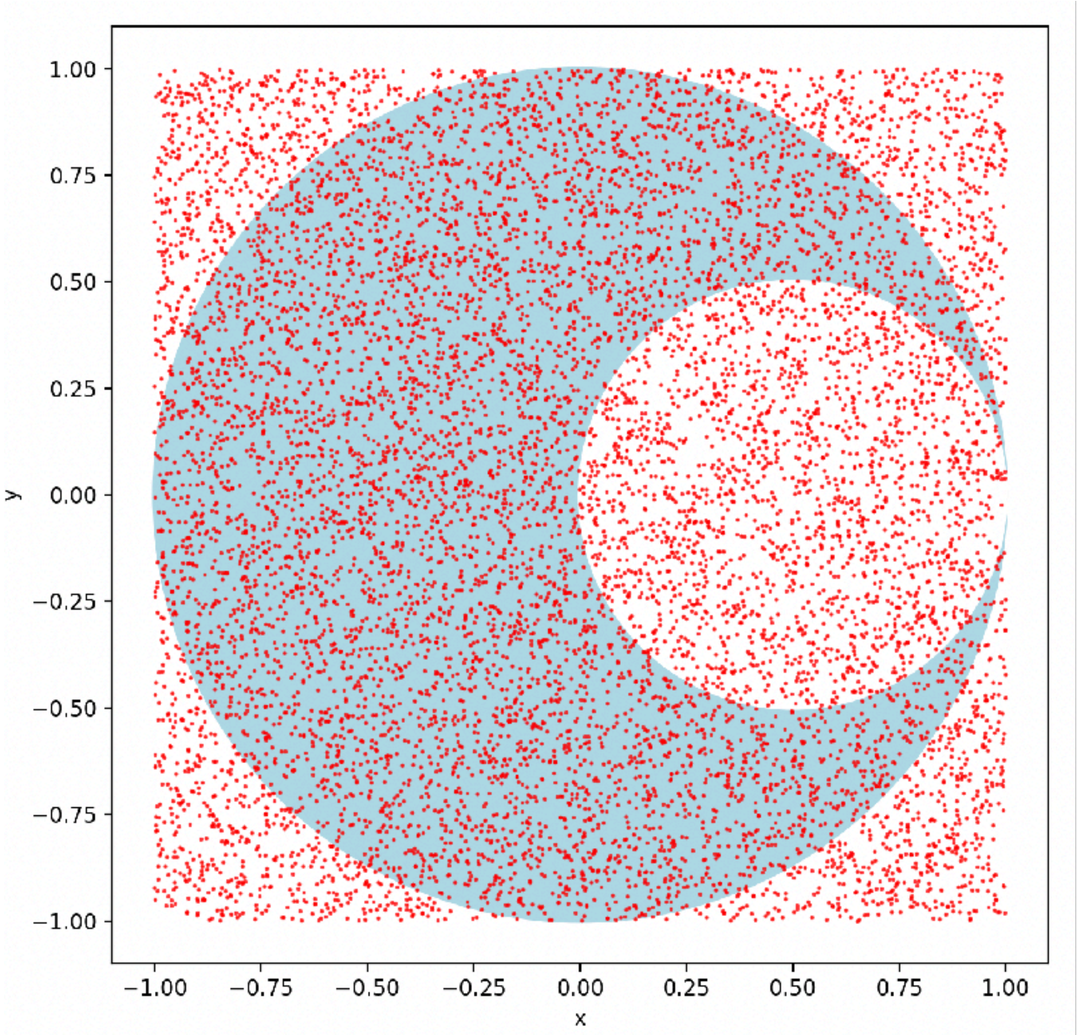
INSEA

Les PRNG permettent de générer des échantillons i.i.d $X_1, \ldots, X_n \sim \mathcal{U}([0,1])$.



Comment simuler des points à l'intérieur de la surface bleue ?

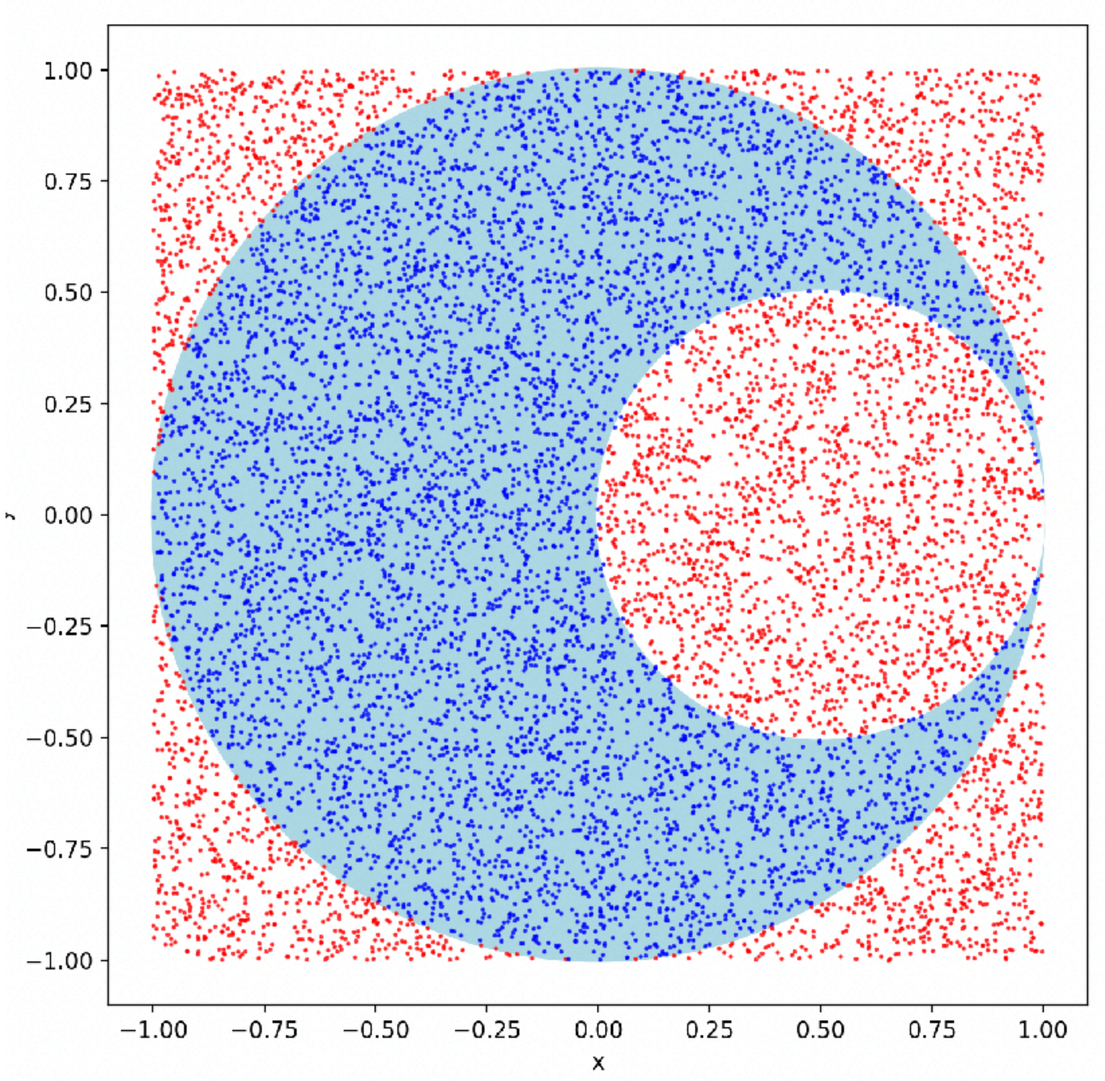
Les PRNG permettent de générer des échantillons i.i.d $X_1, \ldots, X_n \sim \mathcal{U}([0,1])$.



On simule des échantillons uniformes dans le carré $[0, 1]^2$

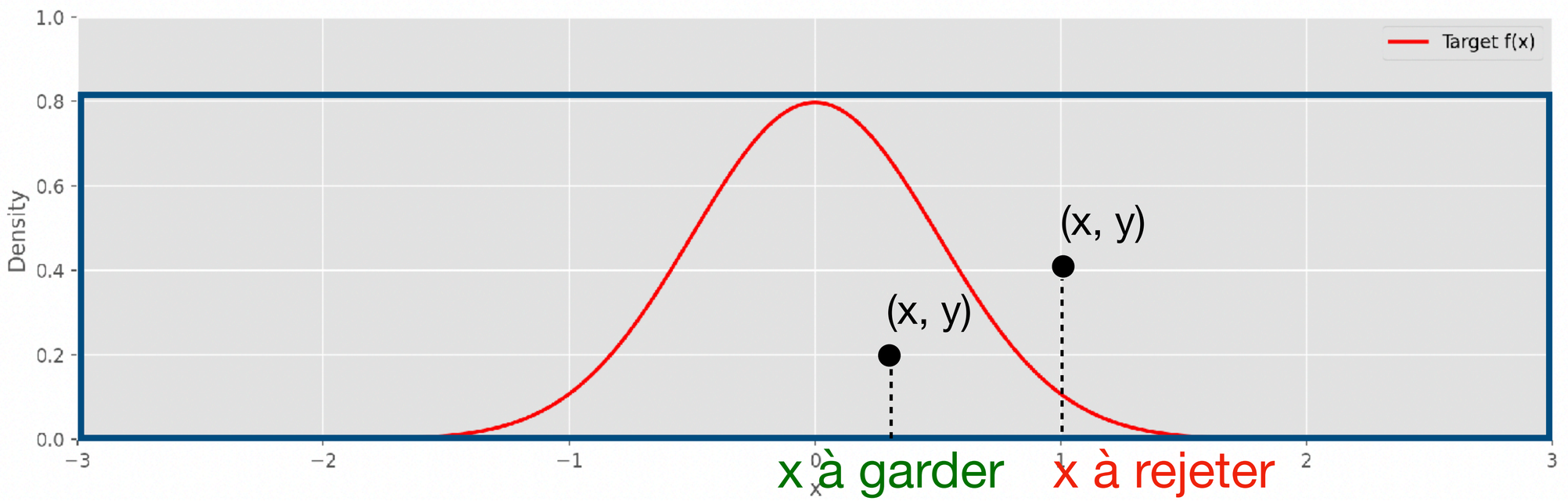
Rejection sampling

Les PRNG permettent de générer des échantillons i.i.d $X_1, \ldots, X_n \sim \mathcal{U}([0,1])$.



On “rejette” les échantillons à l’extérieur de la surface.

On souhaite adapter cette méthode pour avoir des échantillons qui suivent une densité f: définie sur [-3, 3]:

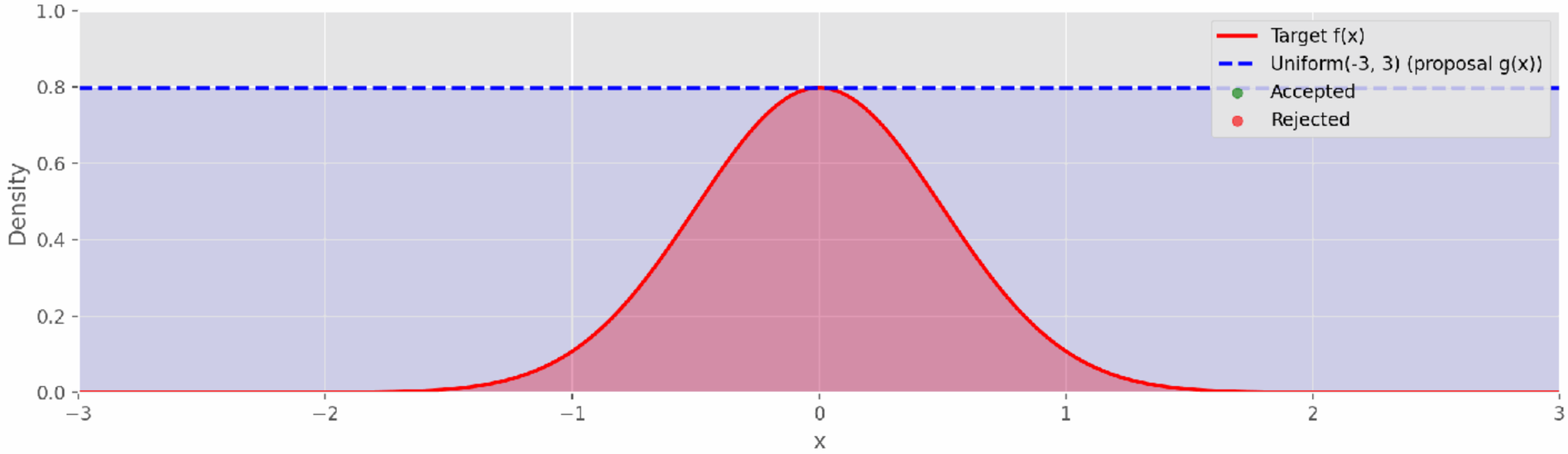


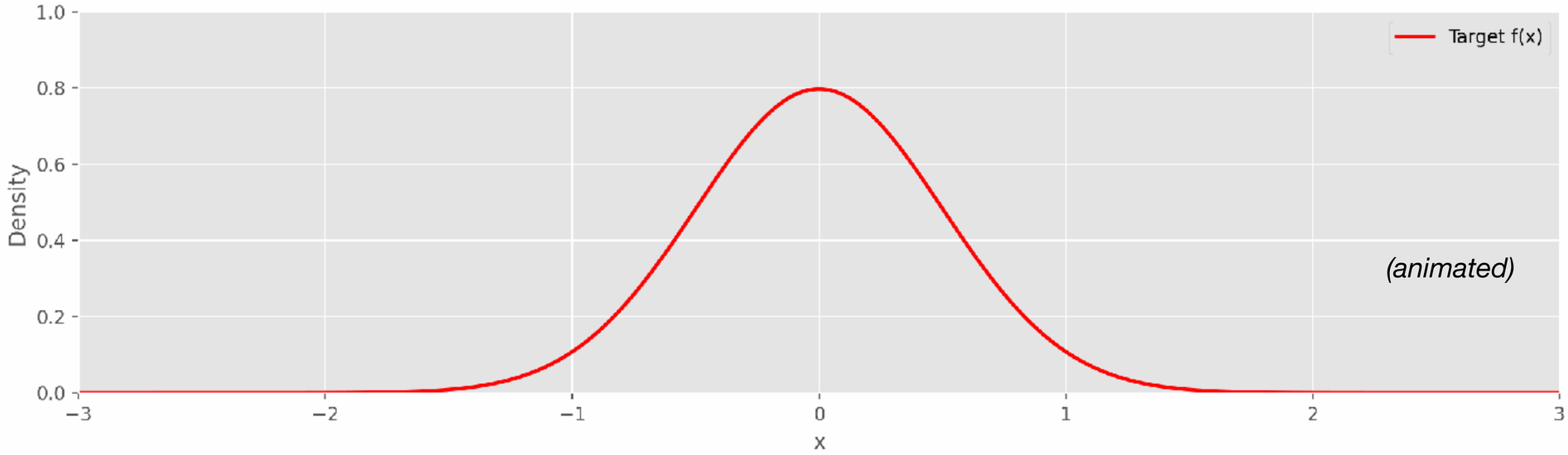
Comment peut-on simuler des échantillons à partir de sa distribution avec des échantillons uniformes ?

Simuler uniformément sur tout le rectangle un point (x, y)

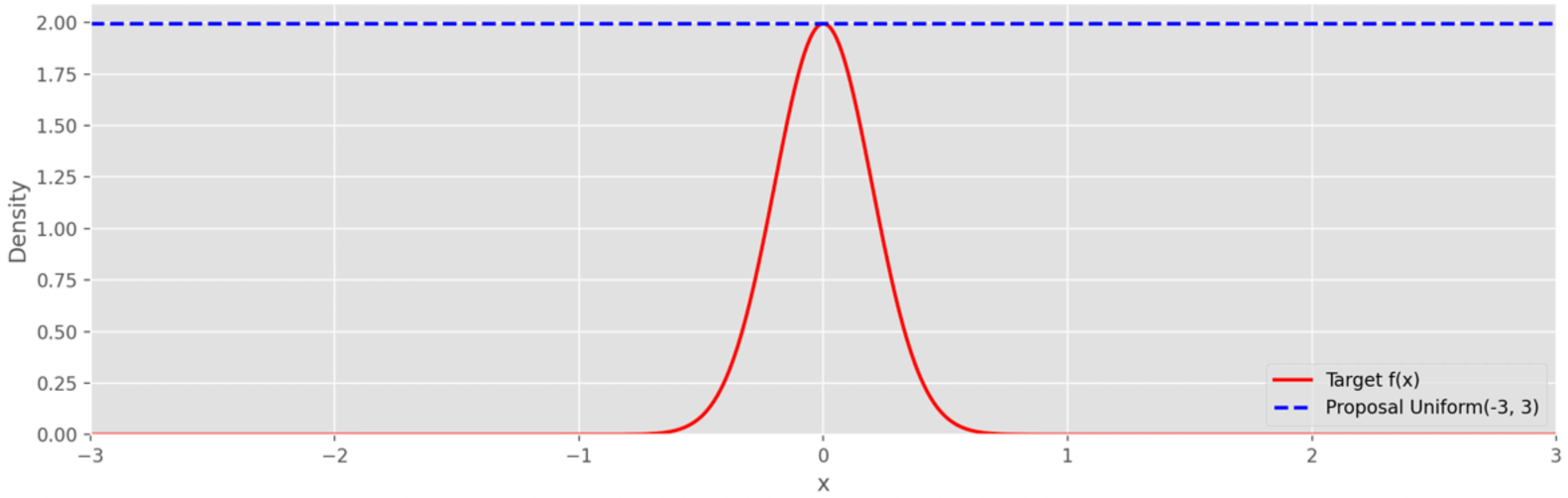
Garder x si le point est sous la densité: si y < f(x)

Rejection sampling

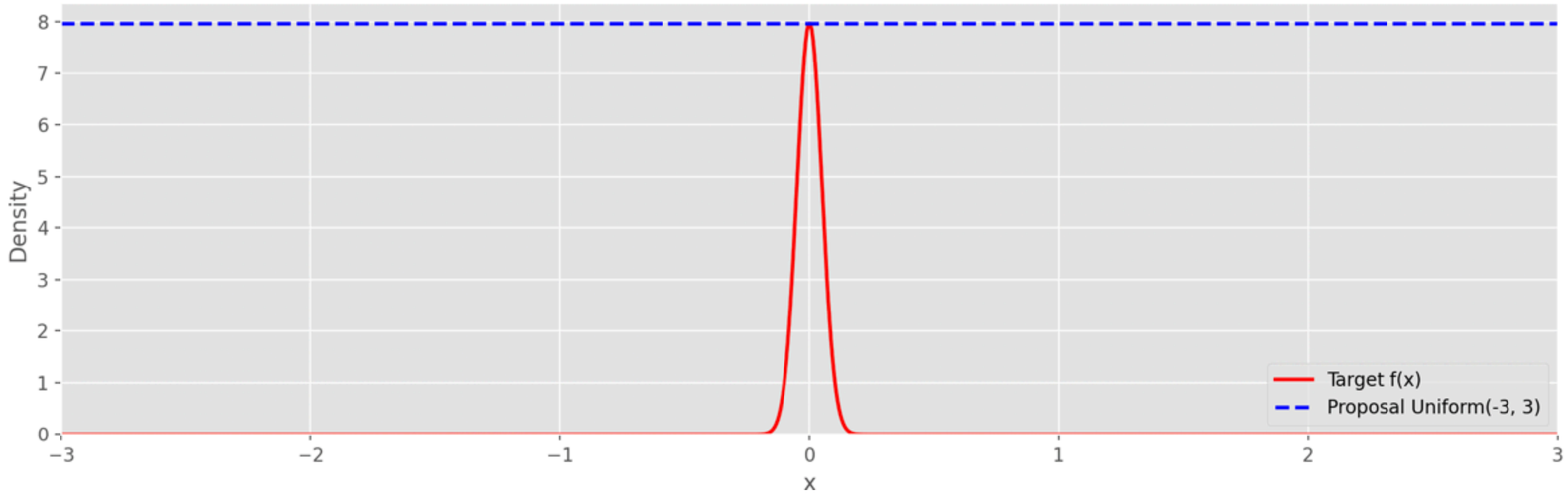




Et si on avait cette densité comme target ?

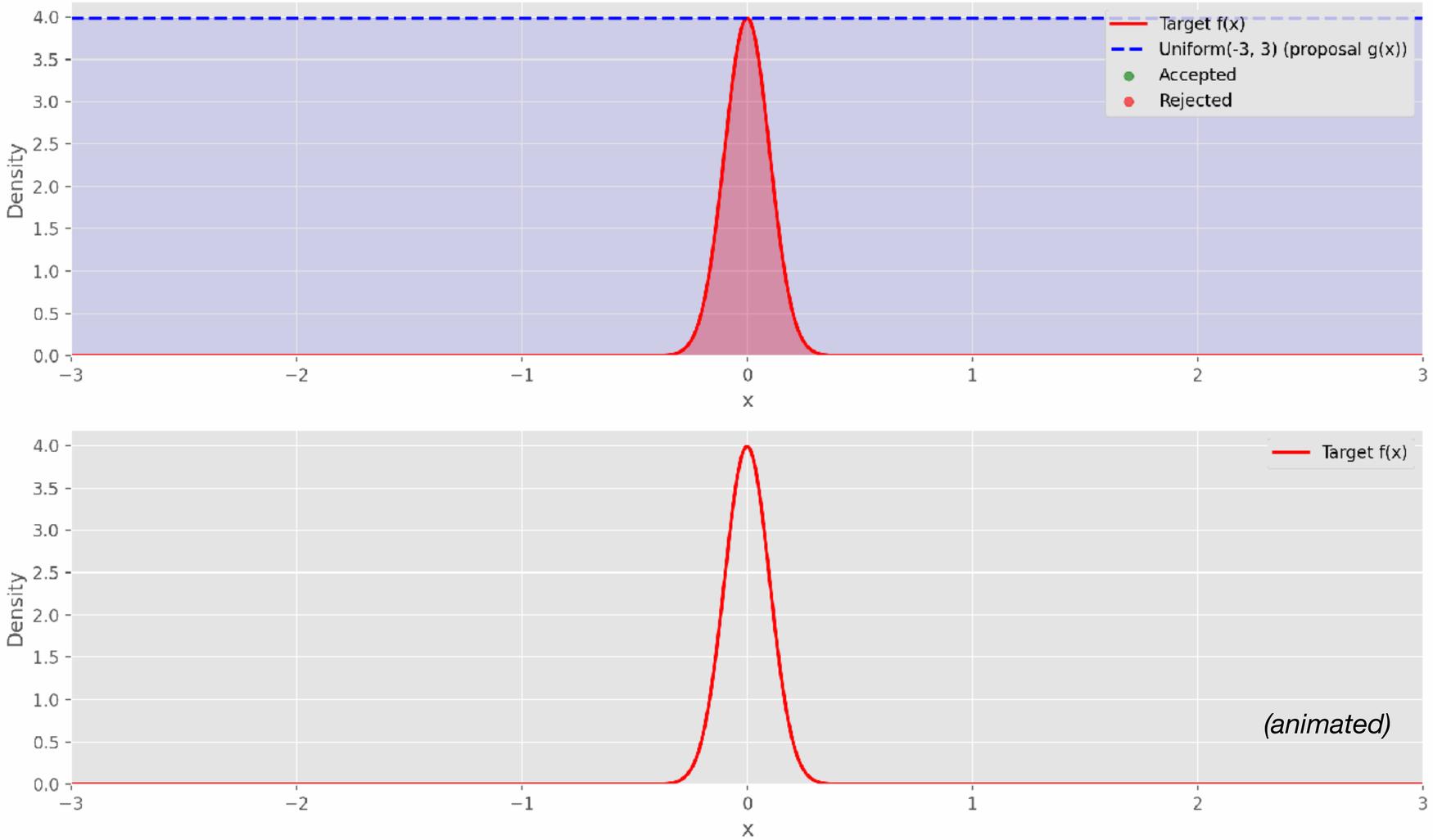


Ou celle-ci ?



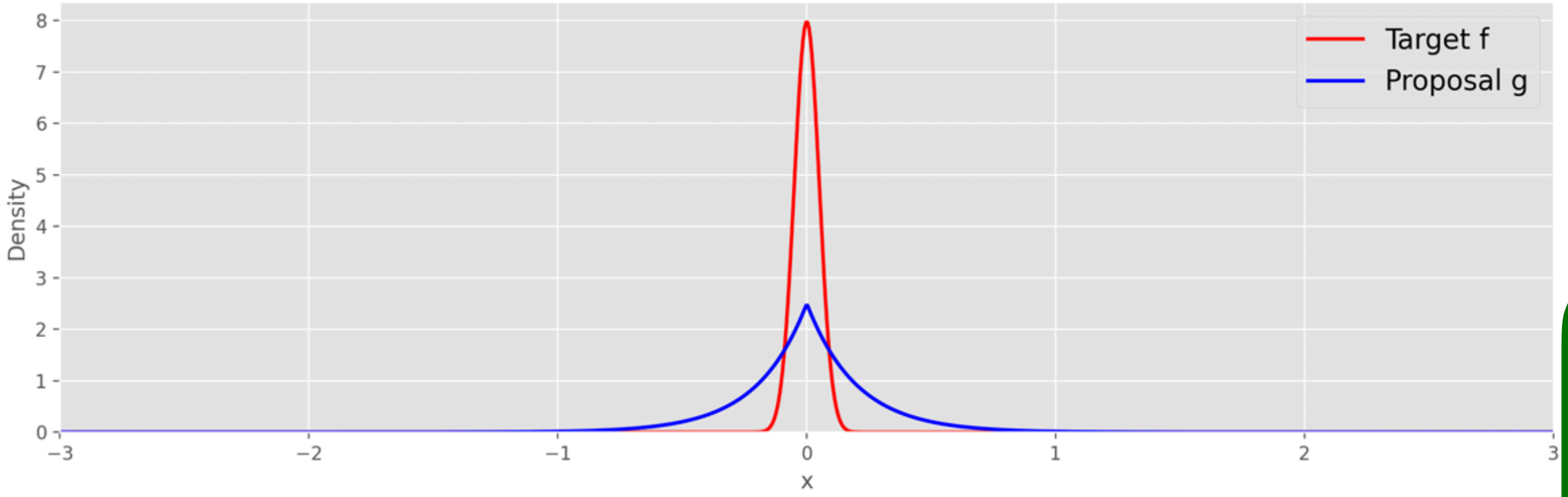
On rejette beaucoup trop d'échantillons.

Comment y remédier ?

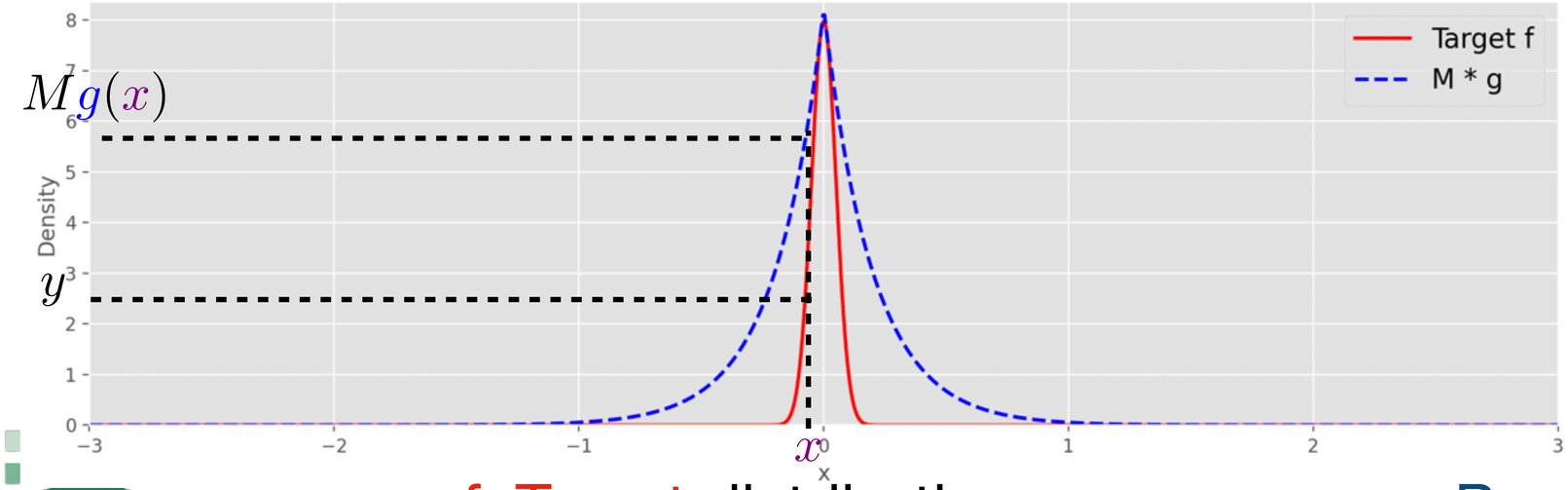


Idée: prendre une courbe bleue plus proche de la densité cible f

Supposons qu'on sait générer à partir d'une loi à densité g:



On cherche alors une constante M telle que f < Mg:



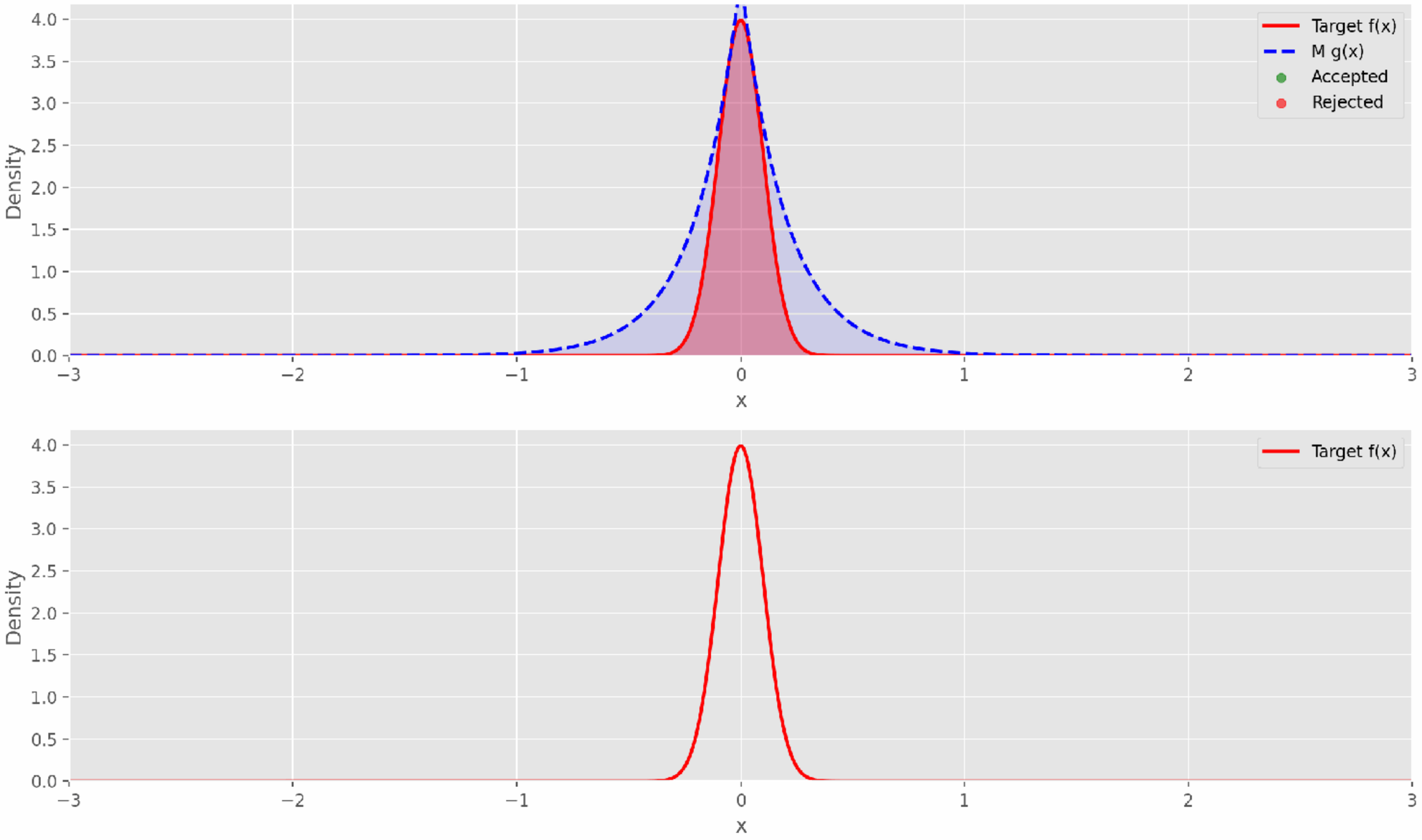
Rejection sampling

1. Sample $x \sim g$
2. Sample $y \sim \mathcal{U}([0, Mg(x)])$
3. If $y \leq f(x)$: Accept x else Reject x.

f: Target distribution

g: Proposal distribution

Rejection sampling



Algorithme de rejet:

1. On souhaite simuler à partir d'une densité f.
2. On cherche une densité g selon à laquelle on sait générer des échantillons.
3. Il faut dominer f par g en trouvant $M > 0$ tel que: $f \leq Mg$ partout.
4. La probabilité d'acceptation est donnée par $\frac{1}{M}$: on veut le plus petit M possible (borne serrée)
5. On itère par la suite l'algorithme:
    1. Sample $x \sim g$
    2. Sample $y \sim \mathcal{U}([0, Mg(x)])$
    3. If $y \leq f(x)$: Accept x else Reject x.

Quelles sont les limites de cette méthode ?

1. Il faut connaître la densité f normalisée !
2. En grande dimension, la zone de rejet explose

Solution: Les méthodes MCMC

INSEA

II Méthodes de Monte-Carlo

1. Introduction
2. Markov Chain Monte-Carlo (MCMC)
3. Algorithmes MCMC avancés

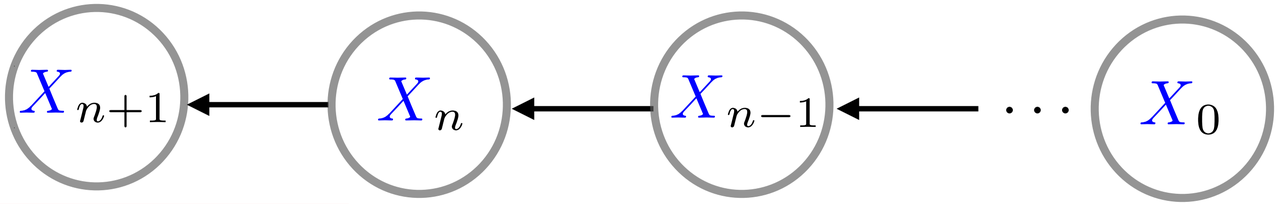
Définition: Chaîne de Markov

Soit $(X_n)_{n\geq 0}$ une suite de variables aléatoires

On dit que $(X_n)_{n\geq 0}$ vérifie la propriété de Markov si son état futur $n+1$ ne dépend que du présent n et non pas du passé $(n-1\ ..\ 0)$:

$$\mathbb{P}(X_{n+1}=x \mid X_n=x_n,\ldots,X_0=x_0) = \mathbb{P}(X_{n+1}=x \mid X_n=x_n)$$

Quel graphe probabiliste permet de visualiser cette suite ?



Définition: Chaîne de Markov homogène

On dit que $(X_n)_n$ est homogène si les variables $X_{n+1} \mid X_n$ ont la même loi $\forall n$.

Dans toute la suite, les chaînes de Markov sont supposées homogènes

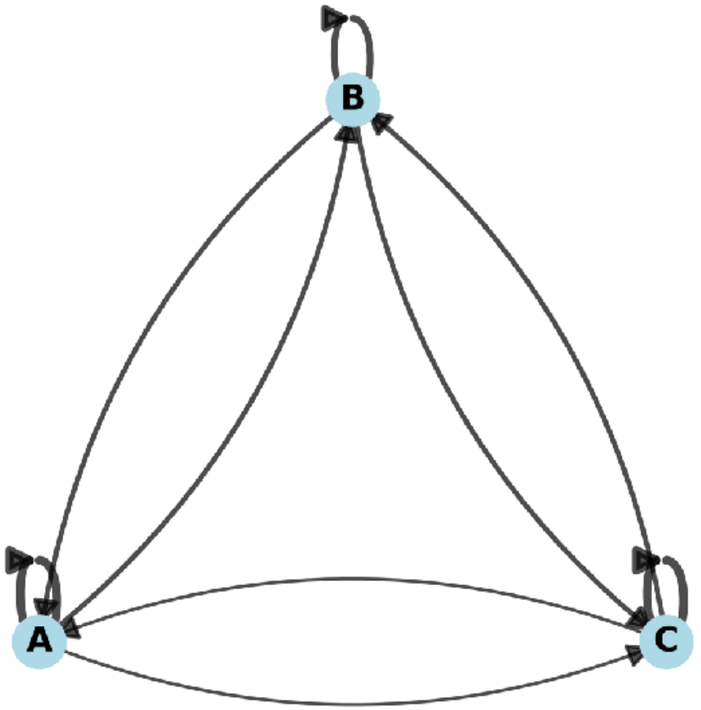
Exemple

Soit $(X_n)_{n\geq 0}$ une chaîne de Markov à valeurs dans $\mathcal{S} = \{A, B, C\}$.

X_n peut valoir A, B ou C et X_{n+1} ne dépend que de X_n avec les probabilités conditionelles de transition:

	To A	**To B**	**To C**
From A	0.5	0.3	0.2
From B	0.3	0.4	0.3
From C	0.2	0.3	0.5

La matrice de transition P



On note la distribution de X_n par le vecteur de probabilités $\pi_n = \begin{bmatrix}\mathbb{P}(X_n = A) & \mathbb{P}(X_n = B) & \mathbb{P}(X_n = C)\end{bmatrix}^\top \in \mathbb{R}^3$.

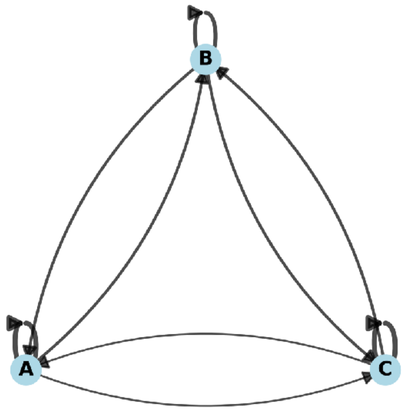
1. Déterminez π_{n+1} en fonction de P et de π_n.
2. On suppose que X_n a une distribution asymptotique π, comment peut-on l'obtenir ?

Stationnarité: cas discret

Soit $(X_n)_{n\geq 0}$ une chaîne de Markov à valeurs dans un ensemble fini $\mathcal{S}$.

Alors:

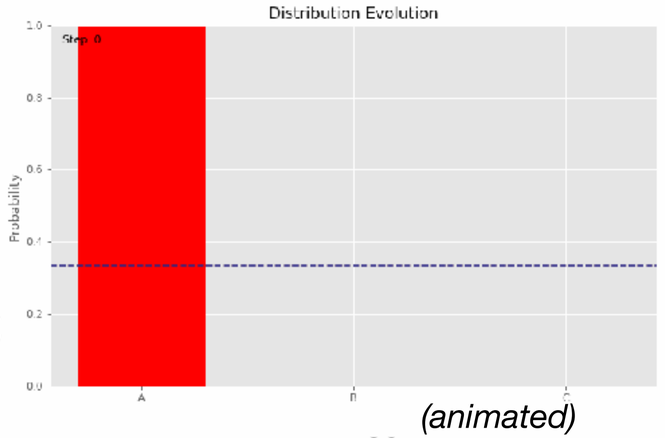
1. $(X_n)_{n\geq 0}$ est dite **irréductible** si $\forall i, j \in \mathcal{S}$, $\exists n \in \mathbb{N}$ tel que on peut atteindre j à partir de i après n étapes.
2. $(X_n)_{n\geq 0}$ est dite **apériodique** si la chaîne ne présente pas de comportement cyclique.
3. $(X_n)_{n\geq 0}$ est dite **récurrente positive** si le temps de retour à n'importe quel $i \in \mathcal{S}$ est fini.
4. $(X_n)_{n\geq 0}$ est dite **ergodique** si elle est à la fois **irréductible**, **apériodique** et **récurrente positive**.
5. Une chaîne $(X_n)_{n\geq 0}$ **ergodique** admet toujours une distribution stationnaire unique $\pi = \lim_{n\to+\infty} \pi_n$.



Histogramme

$\longrightarrow$

des états A, B, C



(animated)

Stationnarité: cas continu

Soit $(X_n)_{n\geq 0}$ une chaîne de Markov à valeurs dans $\mathbb{R}^d$.

La matrice de transition P devient un "noyau": $(x, y) \mapsto K(x, y)$ qui correspond à la densité de $X_{n+1} = y | X_n = x$

Si $X_n \sim \pi_n$ alors $X_{n+1} \sim$? $\pi_{n+1}(y) = \int K(x, y)\pi_n(x)\mathrm{d}x$

Une condition suffisante pour avoir une distribution stationnaire π est le principe "detailed balance":

Detailed balance

Si π vérifie la condition:

$$K(x, y)\pi(x) = K(y, x)\pi(y) \qquad \forall x, y$$

Alors $(X_n)_n$ a une distribution stationnaire π.

Interprétez cette équation en l'intégrant sur des régions A et B contenant x et y respectivement.

On obtient: $\mathbb{P}(X_{n+1} \in A, X_n \in B) = \mathbb{P}(X_{n+1} \in B, X_n \in A)$

"autant de particules vont de A à B que de B à A"

INSEA

Algorithmes MCMC

Soit g une densité (non-normalisée). On veut simuler un échantillon x selon la densité $\pi \overset{\text{def}}{=} \frac{g}{\int g}$.

Soit $(X_n)_n$ une chaîne de Markov admettant une distribution stationnaire à densité π.

Alors pour simuler $x \sim \pi$ Il suffit de prendre $x = X_n$ avec $n \to \infty$.

Pour générer un échantillon, il faut attendre la convergence de la chaîne

Et pour estimer une moyenne a posteriori, il faut plusieurs échantillons i.i.d !

Mais les itérés de la suite ne sont jamais i.i.d …

Théorème Ergodique

Soit $(X_n)_n$ une chaîne de Markov ergodique ayant pour distribution stationnaire π. Alors:

$$\frac{1}{n}\sum_{i=1}^{n} f(X_i) \overset{n\to+\infty}{\longrightarrow} \mathbb{E}_\pi(f(X)) = \int f(x)\pi(x)\mathrm{d}x$$

"Loi des grands nombres" pour des échantillons non i.i.d

Algorithmes MCMC

Soit g une densité (non-normalisée). On veut simuler un échantillon x selon la densité $\pi \overset{\text{def}}{=} \frac{g}{\int g}$.

Soit $(X_n)_n$ une chaîne de Markov admettant une distribution stationnaire à densité π.

Alors pour simuler $x \sim \pi$ Il suffit de prendre $x = X_n$ avec $n \to \infty$.

Comment construire la suite $(X_n)_n$?

Foundation of MCMC

1. Gibbs Sampling
2. Metropolis
3. Metropolis-Hastings (MH)

Physics inspired algorithms

4. Langevin dynamics
5. Hamiltonian Monte-Carlo (HMC)
6. No-U-Turn Sampler (NUTS)

Default algorithm in pyMC

On veut souvent des échantillons suivant une loi a posteriori multivariée $f_{(\theta,\mu,...,\sigma^2)}$.

Où on sait générer $\theta|\mu, ..., \sigma^2$ et $\mu|\theta, \ldots, \sigma^2$ et ainsi de suite

Peut-on générer un paramètre à la fois ?

Oui ! Grâce à l'algorithme de Gibbs.

Algorithme de Gibbs (2D)

On souhaite simuler $\mathbf{X} = (X_1, X_2) \in \mathbb{R}^2$ d'une distribution jointe $\mathbb{P}_{(X_1,X_2)}$. Soit $x_2 \in \mathbb{R}$

1. On sait générer $x_1 \sim X_1$ avec X_2 fixée selon la loi conditionnelle $X_1|X_2 = x_2$
2. On sait générer X_2 avec X_1 fixée selon la loi conditionnelle $X_2|X_1 = x_1$

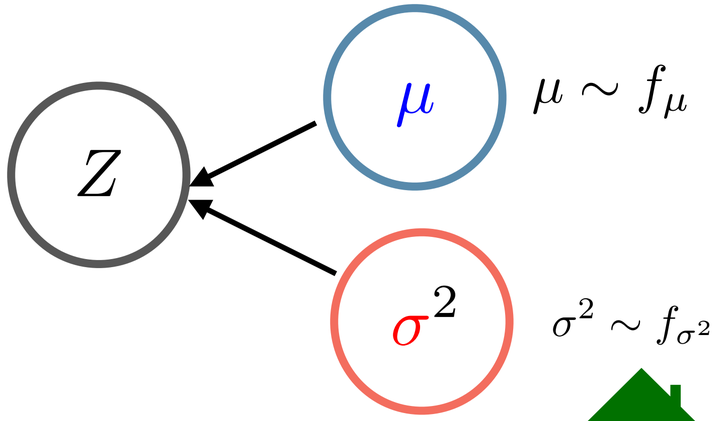
Alors en partant d'un quelconque $x_2 \in \mathbb{R}$, la suite $\mathbf{X}_n$ définie par ces itérés est une chaîne de Markov avec $\mathbb{P}_{(X_1,X_2)}$ comme distribution stationnaire.

Application (Ex 3. TD 1)

Soit $Z|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$. μ et σ^2 ont des densités a priori indépendantes f_μ et f_{σ^2}.

1. Dessiner le graphe probabiliste du modèle.
2. Déterminez la loi a posteriori jointe $(\mu, \sigma^2)|Z$.
3. Déterminez les lois conditionnelles $\mu|\sigma^2, Z$ et $\sigma^2|\mu, Z$.
4. Quelles lois a priori f_μ et f_{σ^2} devrait-on prendre pour avoir des lois conditionnelles usuelles ?

$$Z|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2)$$



Algorithme de Gibbs (général)

On souhaite simuler $\mathbf{X} = (X_1, \ldots, X_d) \in \mathbb{R}^d$ d'une distribution jointe $\mathbb{P}_{(X_1,\ldots,X_d)}$. Soit $x_2, \ldots, x_d \in \mathbb{R}$.

1. Générer $x_1 \sim X_1$ avec $X_2, \ldots, X_d$ fixés selon la loi conditionnelle $X_1 | X_2 = x_2, \ldots X_d = x_d$
2. Générer (et mettre à jour) $x_2 \sim X_2$ avec $X_1, X_3, \ldots, X_d$ fixés selon la loi conditionnelle $X_2 | X_1 = x_1, X_3 = x_3, \ldots, X_d = x_d$
3. Générer (et mettre à jour) $x_3 \sim X_3$ avec $X_1, X_2, \ldots, X_d$ fixés selon la loi conditionnelle $X_3 | X_1 = x_1, X_2 = x_2, \ldots, X_d = x_d$
4. ...

La suite $\mathbf{X}_n$ définie par ces itérés est une chaîne de Markov avec $\mathbb{P}_{(X_1,\ldots,X_n)}$ comme distribution stationnaire.

Avantages:

1. Simple et facile à mettre en oeuvre
2. Efficace pour simuler les lois a posteriori d'un modèle hiérarchique

Inconvénients:

1. Nécessite de savoir simuler les lois conditionnelles
2. Convergence lente en grande dimension
3. Convergence très lente si les composantes sont corrélées

L’algorithme de Gibbs permet de découper le problème de simulation d’une loi jointe en lois conditionnelles en 1D.

Et si la loi conditionnelle en 1D n’est pas usuelle ?

C’est là où intervient l’algorithme de Metropolis.

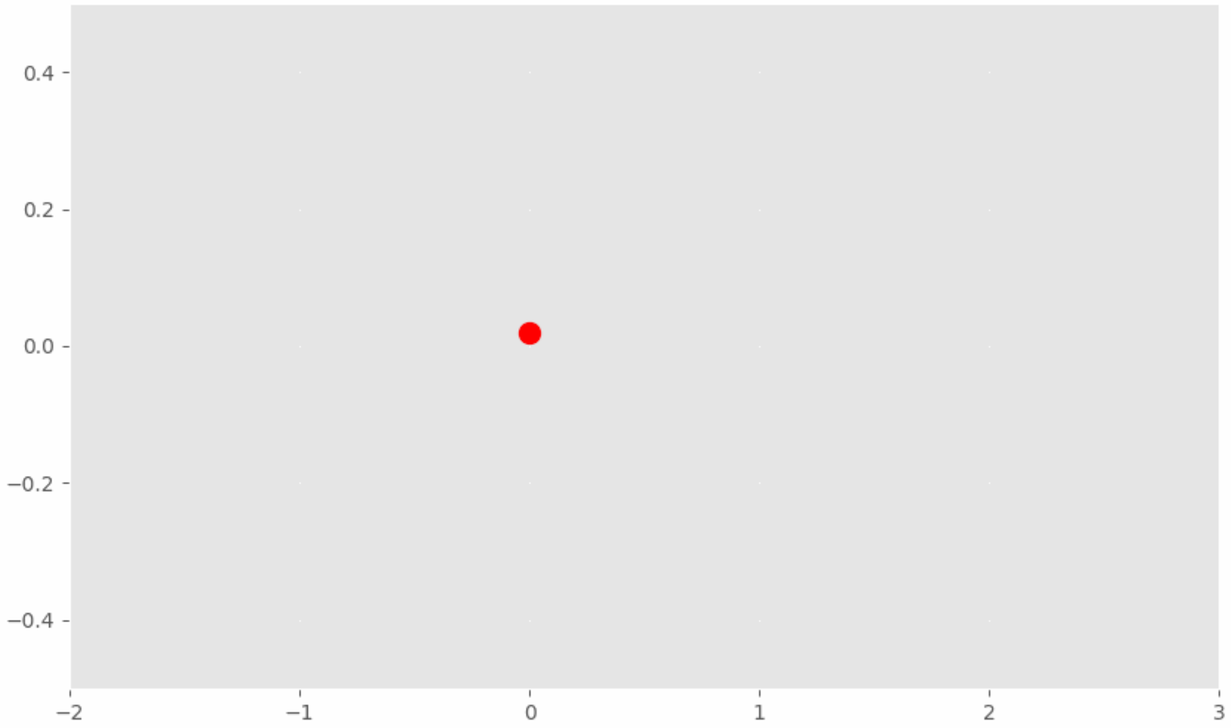
Random Walk

Une chaîne de Markov simple est donnée par une marche aléatoire: $X_{n+1} = X_n + \varepsilon$ avec $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

Admet-elle une distribution stationnaire ?

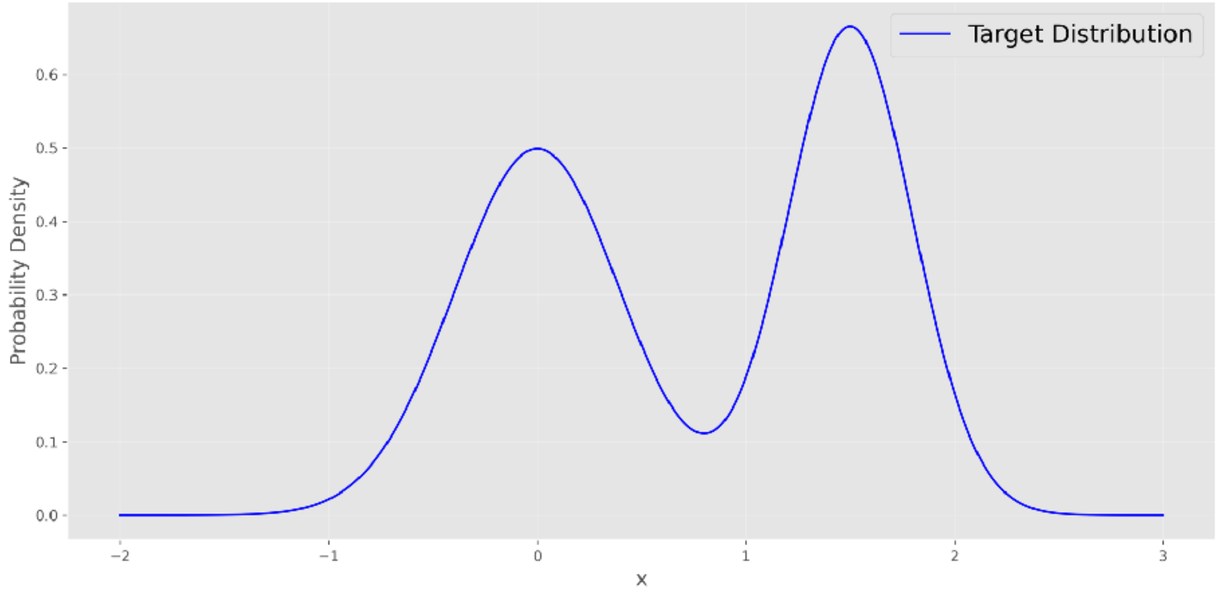
Non ! Elle a une variance divergente (calcul simple)

Visuellement, c'est une suite qui "explore" l'espace:



L'algorithme

On souhaite créer une chaîne de Markov avec la distribution stationnaire:



Quel est le problème avec cet algorithme ?

Risque de rester coincé autour du maximum !

Idée de Metropolis:

1. Explorer l'espace avec une marche aléatoire $X_{n+1} = X_n + \varepsilon$ avec $\varepsilon \sim \mathcal{N}(0, \sigma^2)$

2. Utiliser la densité cible f pour accepter ou rejeter des "sauts":

Accepter de sauter à X_{n+1} s'il est "meilleur" que X_n: ~~$f(X_{n+1}) \geq f(X_n)$~~ avec probabilité $\frac{f(X_{n+1})}{f(X_n)}$

Sinon $X_{n+1} = X_n$ et refaire la marche aléatoire

Comment l'implémenter en pratique ?

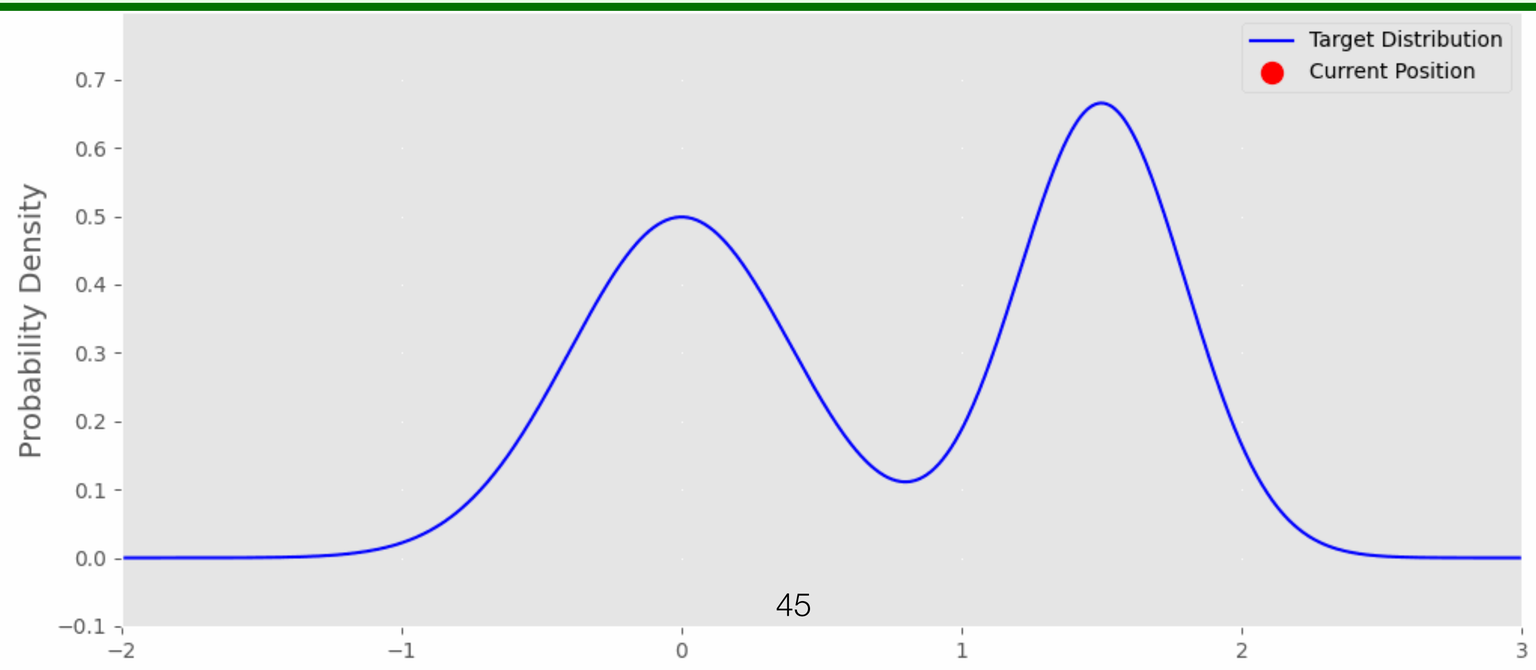
INSEA

Algorithme de Metropolis (Gaussien)

Soit f une densité de probabilité. On suppose que X_n est déjà généré. X_{n+1} est défini par:

1. Générer $y \sim \mathcal{N}(X_n, \sigma^2)$ équivalent à $y = X_n + \varepsilon$ et générer $\varepsilon \sim \mathcal{N}(0, \sigma^2)$
2. Générer $u \sim \mathcal{U}([0,1])$. équivalent à accepter y avec probabilité $\min(1, \frac{f(y)}{f(X_n)})$
3. Si $u < \min(1, \frac{f(y)}{f(X_n)})$ alors $X_{n+1} = y$, sinon $X_{n+1} = X_n$.

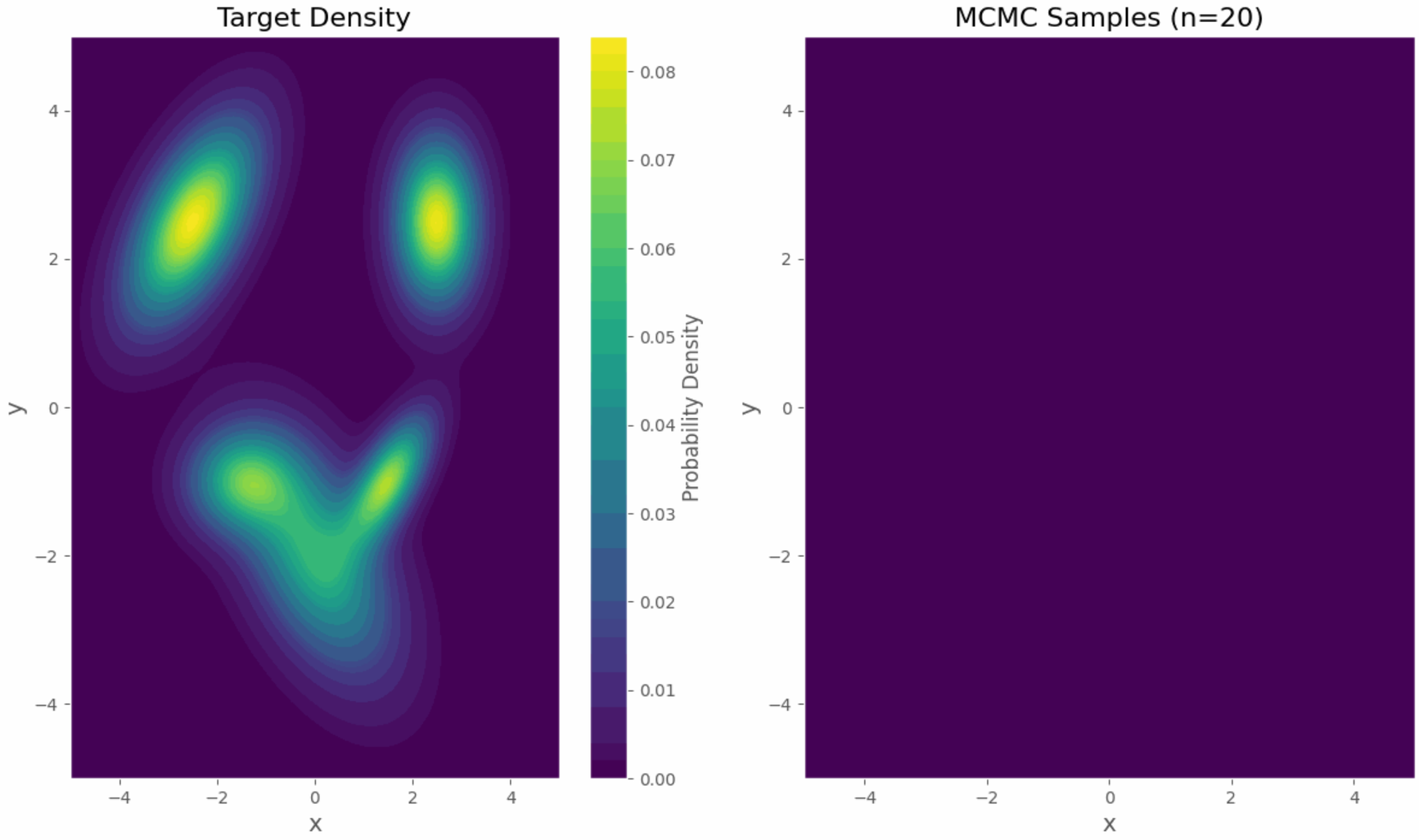
La suite $(X_n)_n$ obtenue admet une distribution stationnaire donnée par la densité f.



INSEA

Metropolis in action

Exemple en deux dimensions:



Algorithme de Metropolis (Gaussien)

Soit f une densité de probabilité. On suppose que X_n est déjà généré. X_{n+1} est défini par:

1. Générer $y \sim \mathcal{N}(X_n, \sigma^2)$
2. Générer $u \sim \mathcal{U}([0,1])$.
3. Si $u < \min(1, \frac{f(y)}{f(X_n)})$ alors $X_{n+1} = y$, sinon $X_{n+1} = X_n$.

La suite $(X_n)_n$ obtenue admet une distribution stationnaire donnée par la densité f.

Peut-on utiliser cet algorithme si on a uniquement accès à $g \propto f$?

Oui ! $f = \frac{g}{\int g}$ et la constante de normalisation $\int g$ disparaît dans le rapport $\frac{f(y)}{f(X_n)} = \frac{g(y)}{g(X_n)}$

Quel est l'effet de σ^2 ?

σ^2 contrôle le trade-off "exploration-exploitation"

Algorithme de Metropolis (Gaussien)

Soit f une densité de probabilité. On suppose que X_n est déjà généré. X_{n+1} est défini par:

1. Générer $y \sim \mathcal{N}(X_n, \sigma^2)$
2. Générer $u \sim \mathcal{U}([0,1])$.
3. Si $u < \min(1, \frac{f(y)}{f(X_n)})$ alors $X_{n+1} = y$, sinon $X_{n+1} = X_n$.

La suite $(X_n)_n$ obtenue admet une distribution stationnaire donnée par la densité f.

Avantages:

1. Il suffit de savoir calculer une densité (même non-normalisée) f pour l'implémenter
2. Généralisable facilement en dimension > 1

Inconvénients:

1. La variance doit être assez petite pour éviter trop de rejets (*random walk behavior*)
2. La variance doit être assez grande pour bien explorer l'espace (distributions multimodales)
3. Le pourcentage d'acceptation tend vers 0 en grande dimension

Qu’en est-il d’une densité cible discrète ?

Hastings généralise l’algorithme Metropolis où on peut choisir la distribution qui explore l’espace (marche aléatoire).

Hastings généralise l'algorithme de Metropolis en prenant une distribution de transition (exploration) quelconque:

Algorithme de Metropolis-Hastings

Soit f une densité de probabilité et $q(y|x)$ une densité de transition (proposal) de x à y.
On suppose que X_n est déjà généré. X_{n+1} est défini par:

1. Générer $y \sim q(.|X_n)$
2. Générer $u \sim \mathcal{U}([0,1])$.
3. Si $u < \min(1, \frac{f(y)q(X_n|y)}{f(X_n)q(y|X_n)})$ alors $X_{n+1} = y$, sinon $X_{n+1} = X_n$.

La suite $(X_n)_n$ obtenue admet une distribution stationnaire donnée par la densité f.

Avantages:

1. Utile si la distribution cible f a un domaine borné (par ex. [0, 1], on peut choisir Q = Beta)
2. Utile si la distribution cible f est discrète (Q uniforme sur ensemble fini)
3. On peut utiliser une densité q non normalisée (Gaussienne tronquée)

Inconvénients: Identiques à ceux de l'algorithme de Metropolis

Remarque: si q est symétrique, on retrouve l'algorithme de Metropolis

INSEA

Monte-Carlo: récap des séances passées

On est souvent confronté à calculer une quantité du type: $I \overset{\text{def}}{=} \int \varphi(x) f(x) \mathrm{d}x$.

Par exemple, avec:

1. f = densité non normalisée et $\varphi = 1$: I = constante de normalisation
2. f = densité normalisée et $\varphi = \mathbb{1}_A$: I = Probabilité d'un évenement
3. f = densité normalisée et $\varphi : x \mapsto x^k$: I = moment d'ordre k de loi a posteriori.
4. f = densité à posteriori normalisée et $\varphi : x \mapsto x$: I = Estimateur de Bayes.

Monte-Carlo: récap des séances passées

On est souvent confronté à calculer une quantité du type: $I \overset{\text{def}}{=} \int \varphi(x) f(x) \mathrm{d}x$.

En pratique on utilise:

1. Un estimateur Monte-Carlo "brut" *(Crude Monte-Carlo)*

Avec des échantillons $X_1, \ldots, X_n$ i.i.d $\sim f$ on peut estimer I avec : $\widehat{I} = \frac{1}{n} \sum_{i=1}^{n} \varphi(X_i)$.

Il suffit de simuler (générer) les échantillons et calculer la moyenne

Pour simuler les X_i, on utilise des transformations d'échantillons uniformes, Rejection sampling ...

1.bis: Importance Sampling (préférentiel), Variables de contrôle ... (à voir en *Méthodes de simulation*)

2. L'estimateur Markov-Chain Monte-Carlo

Avec une chaîne de Markov ergodique $(X_i)_i$ à distribution stationnaire f, on peut estimer I avec : $\widehat{I} = \frac{1}{n} \sum_{i=1}^{n} \varphi(X_i)$.

Pour simuler les X_i, on utilise:

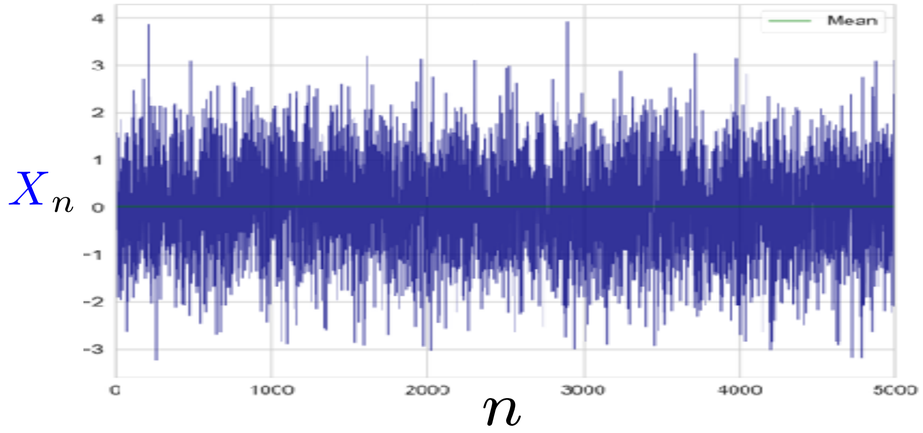
1. Metropolis-Hastings (Exploration aléatoire + rejet)
2. Gibbs (loi jointe $\rightarrow$ lois conditionnelles)

Il faut vérifier la convergence de la chaîne de Markov

Soit $(X_n)_{n\geq 0}$ une chaîne de Markov donnée par un algorithme MCMC.

L'estimateur $\widehat{I} = \frac{1}{n}\sum_{i=1}^{n} \varphi(X_i)$ est de bonne qualité si:

1. Converge rapidement: les X_i utilisés ont atteint le régime stationnaire.



Régime stationnaire = $X_n \sim f$:
la chaîne semble "mélangée",
pas de tendance, absence de "pattern",
ne reste pas bloquée.

2. L'estimateur a une petite variance: $\mathbb{V}(\widehat{I}) = \frac{1}{n^2}\left[\sum_{i=1}^{n} \mathbb{V}(\varphi(X_i)) + 2\sum_{i<j} \mathbb{C}\mathrm{ov}(\varphi(X_i), \varphi(X_j))\right]$

Faible variance = faibles corrélations $\mathbb{C}\mathrm{ov}(X_i, X_j)$.

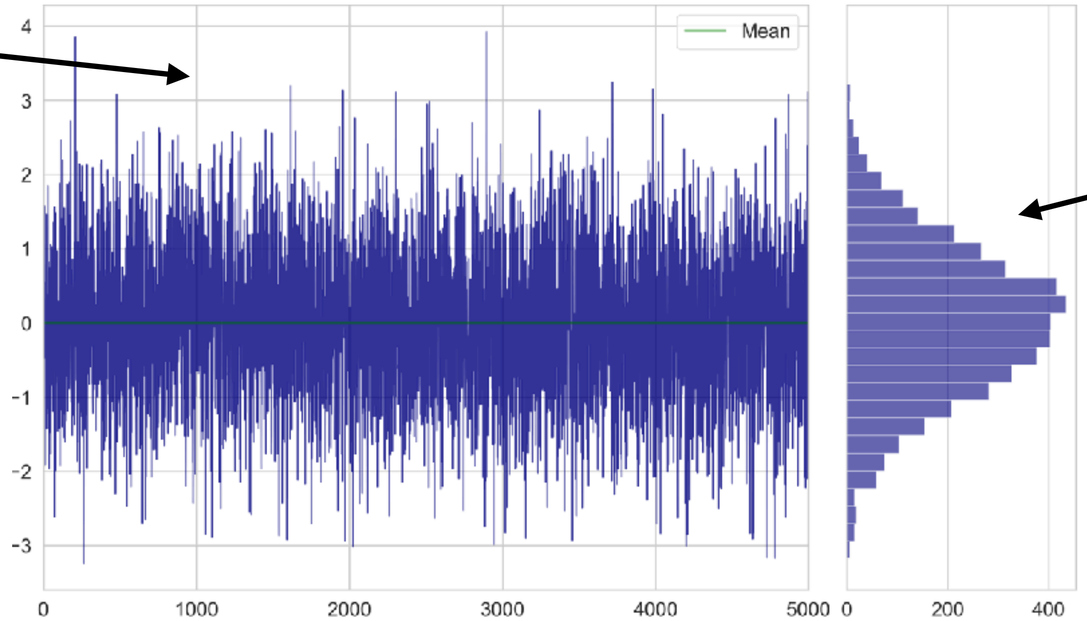
Auto-corrélation avec lag k = $\mathbb{C}\mathrm{ov}(X_n, X_{n+k})$.

Décroissante en fonction de k.

Empiriquement avec N échantillons:
$\mathrm{AutoCorr}(k) = \mathrm{Corr}([X_0, X_1, \ldots, X_{N-k}], [X_k, X_{1+k}, \ldots, X_N])$

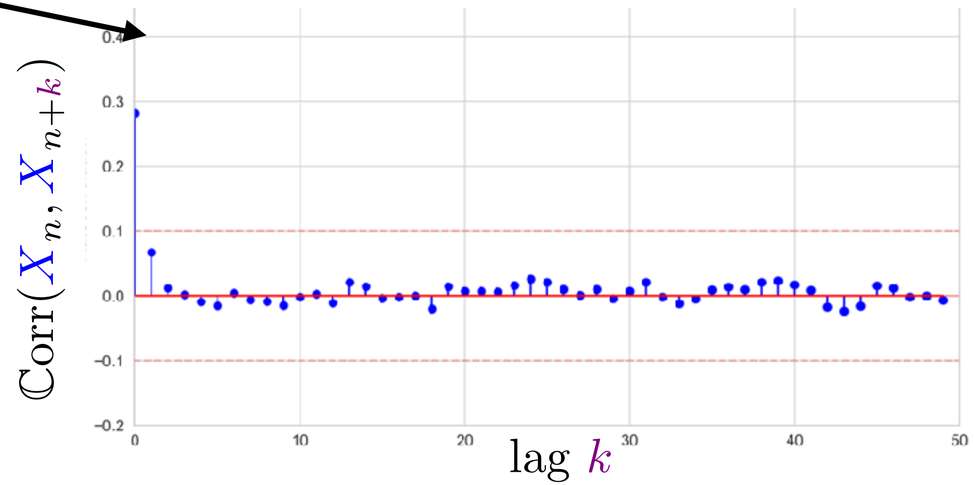
Visuals

“Trace plot”



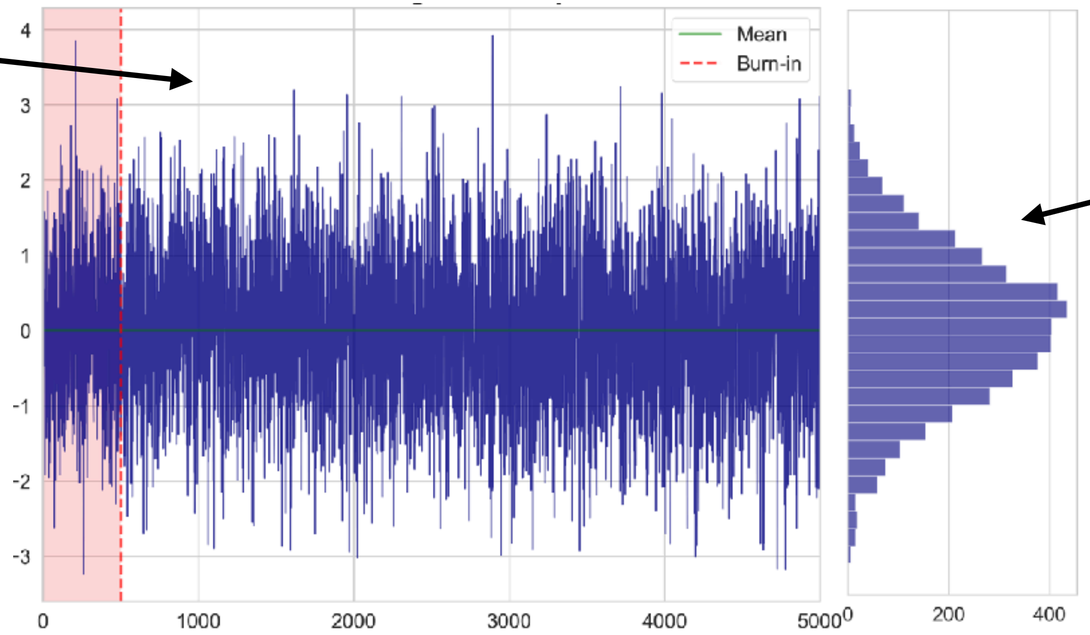
Histogramme des échantillons

“Auto-correlation plot”



Visuals

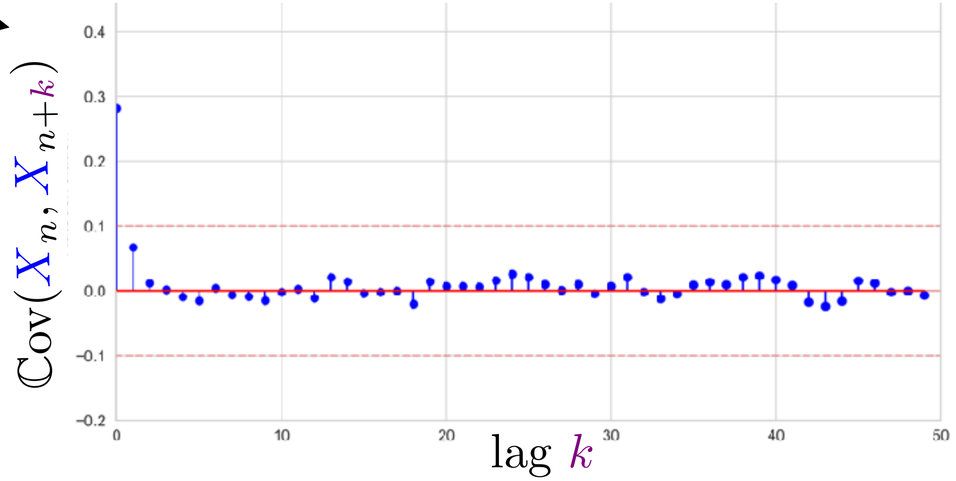
“Trace plot”



Histogramme des échantillons

En pratique, on “jette” la première partie des échantillons pour être dans le régime stationnaire: période de “chauffe” / *Burn-in.*

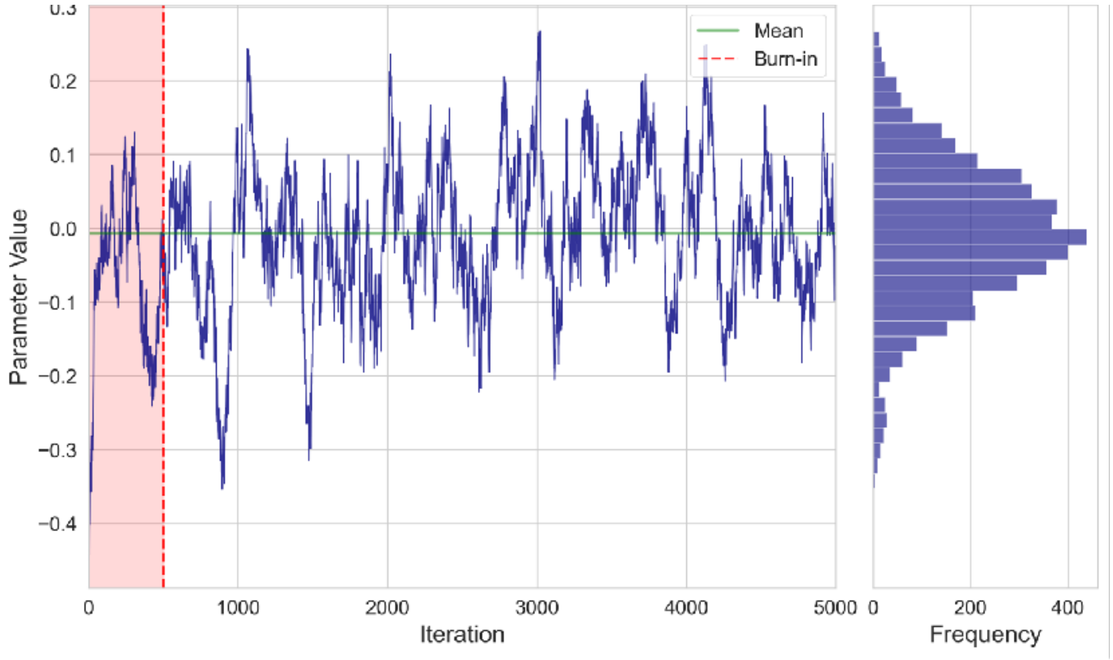
“Auto-correlation plot”



La corrélation entre deux échantillons consécutifs est ~ 0.3. On peut prendre un échantillon sur deux ou trois pour réduire les corrélations: c’est le *thinning*

MCMC diagnostics in 1D

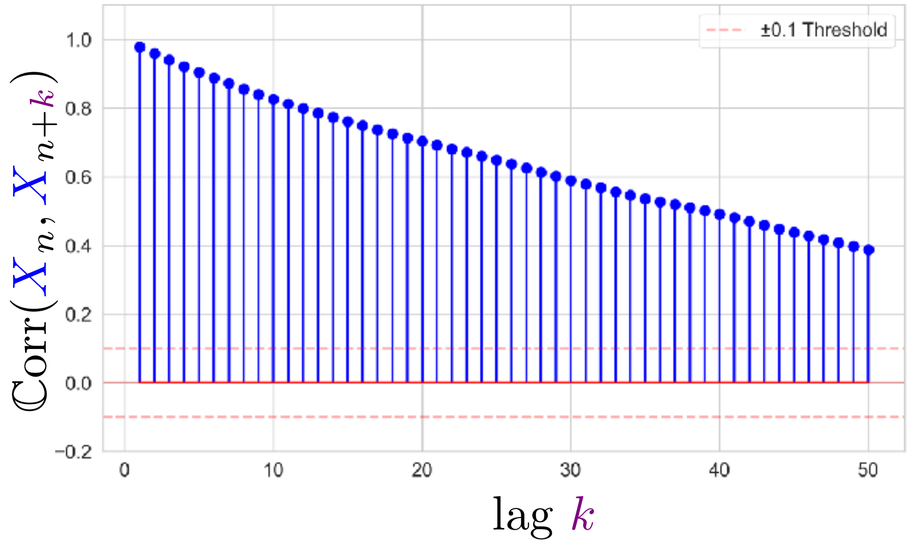
Visuals



Est-ce une bonne chaîne MCMC ?

Comment expliquer ce comportement ?

Auto-corrélations excessives:

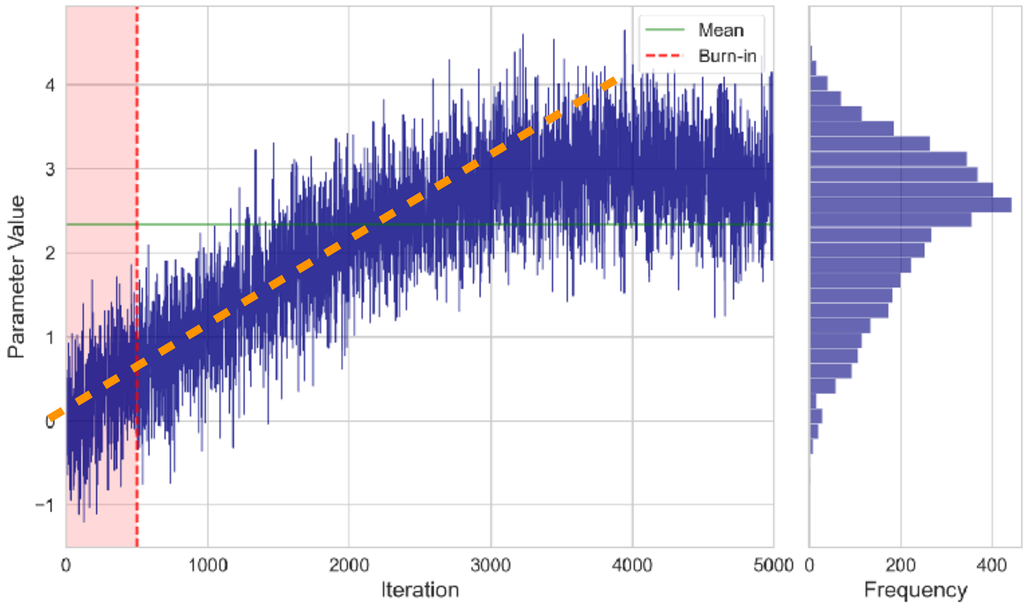


Raison principale ?

Metropolis: On n'explore pas assez l'espace

MCMC diagnostics in 1D

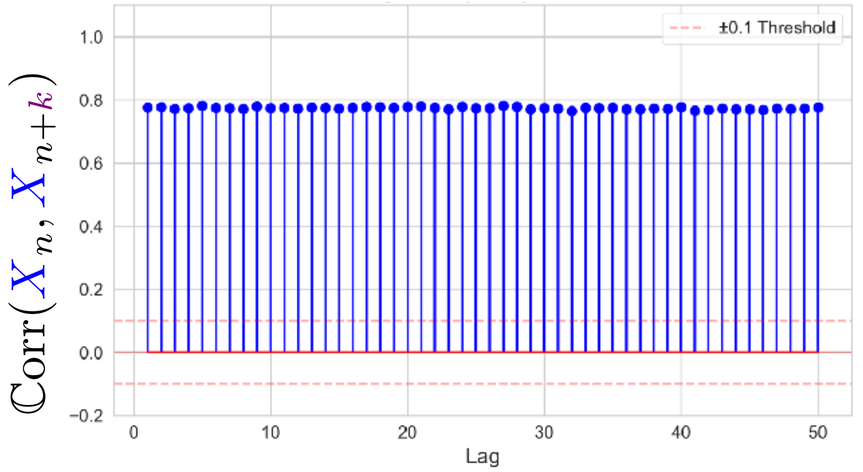
Visuals



Est-ce une bonne chaîne MCMC ?

Drift excessif !

(autocorr inutile ici, on s'y attendait vu le trace plot)



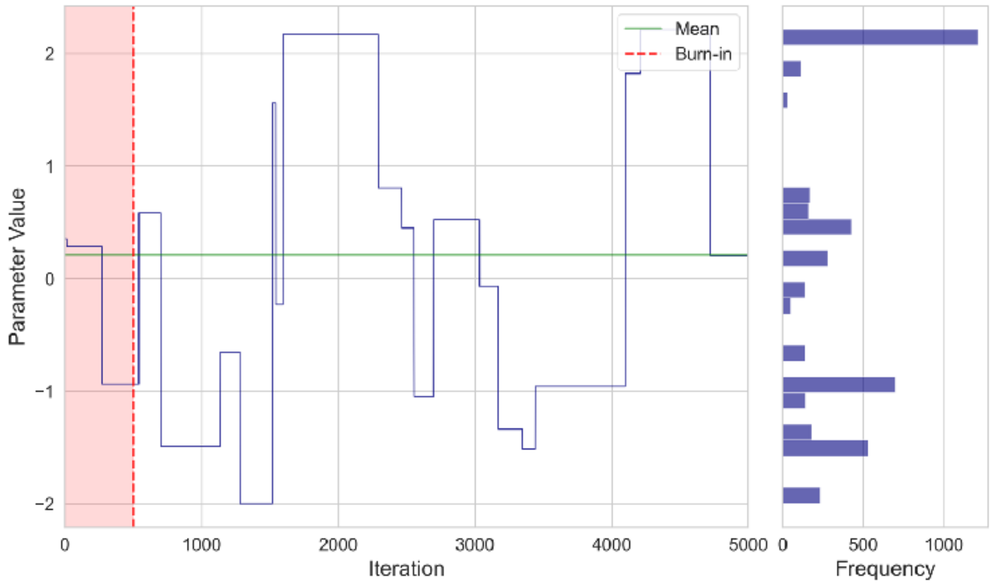
Explications principales ?

Très probablement:

1. Erreur d'implémentation
2. Loi a priori extrêmement loin des données
3. Convergence très très lente

MCMC diagnostics in 1D

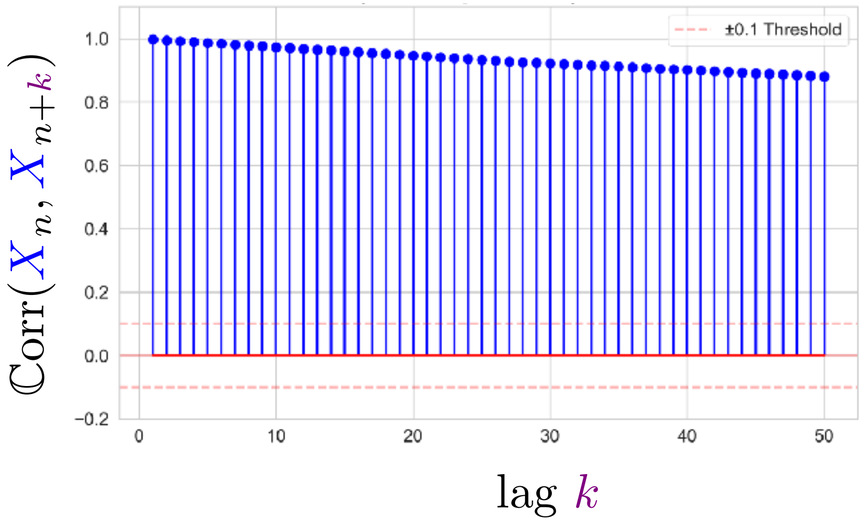
Visuals



Est-ce une bonne chaîne MCMC ?

Aucun mélange visible, Chaîne bloquée !

(autocorr inutile ici, on s'y attendait vu le trace plot)



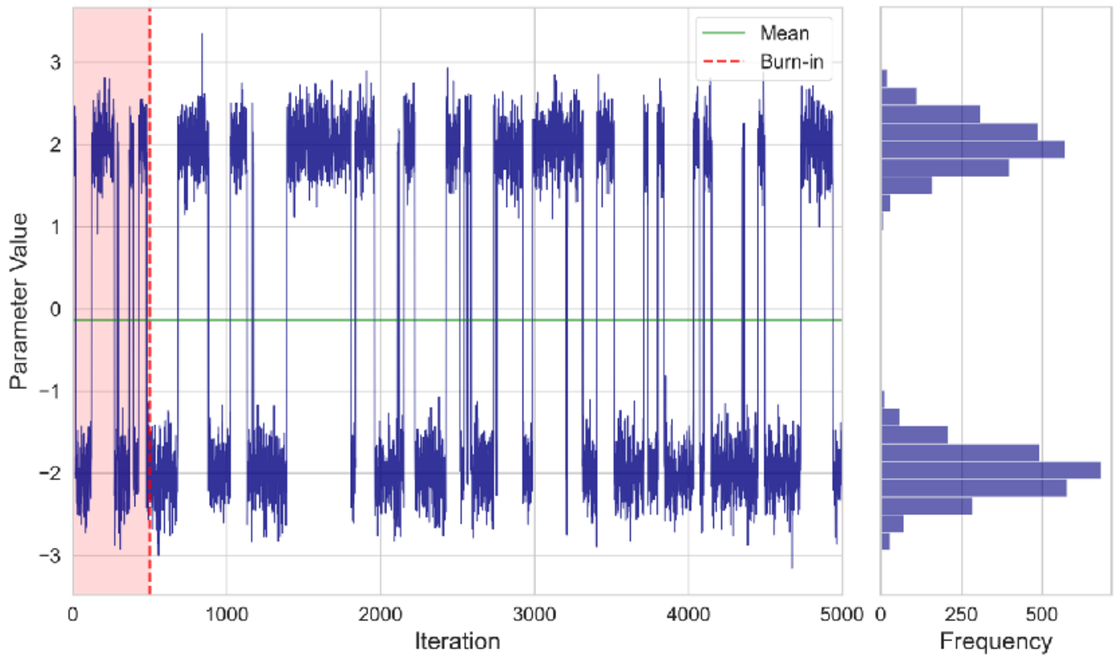
Explications principales ?

Très probablement:

1. Erreur d'implémentation
2. Mauvais choix de la distribution de proposition (exploration très limitée)

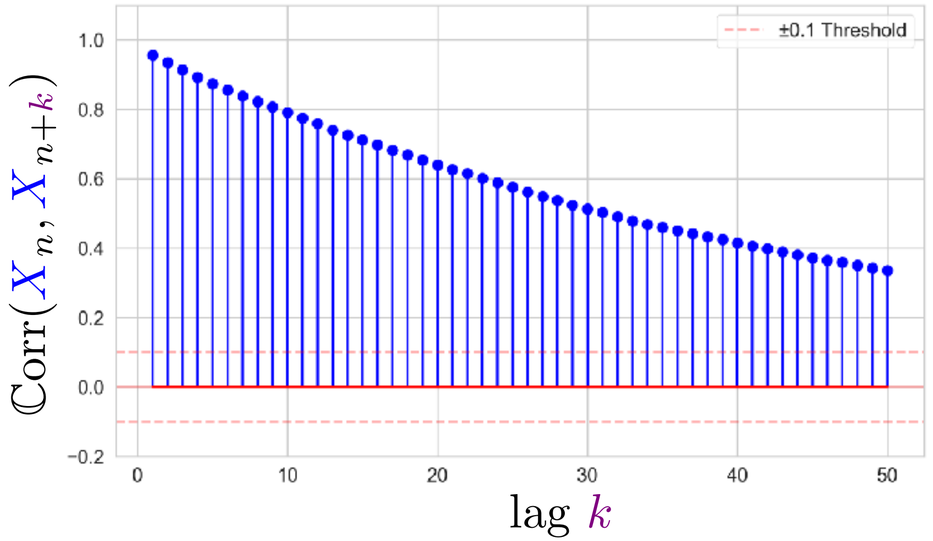
MCMC diagnostics in 1D

Visuals



Est-ce une bonne chaîne MCMC ?

Oui ! Mais la distribution cible est multimodale …



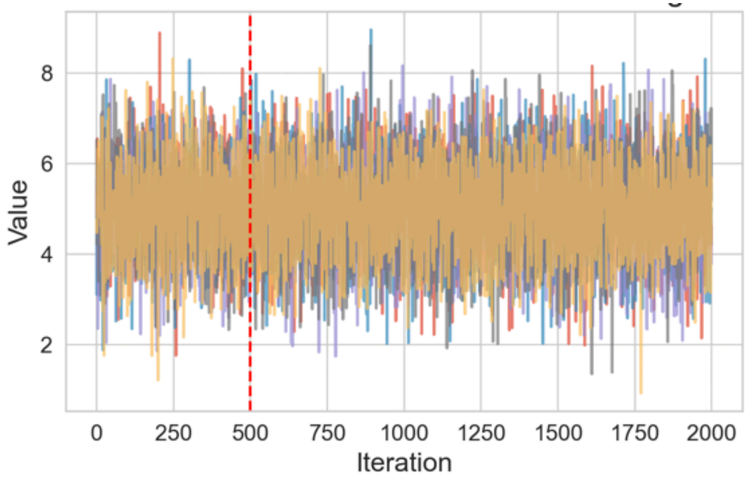
En pratique:

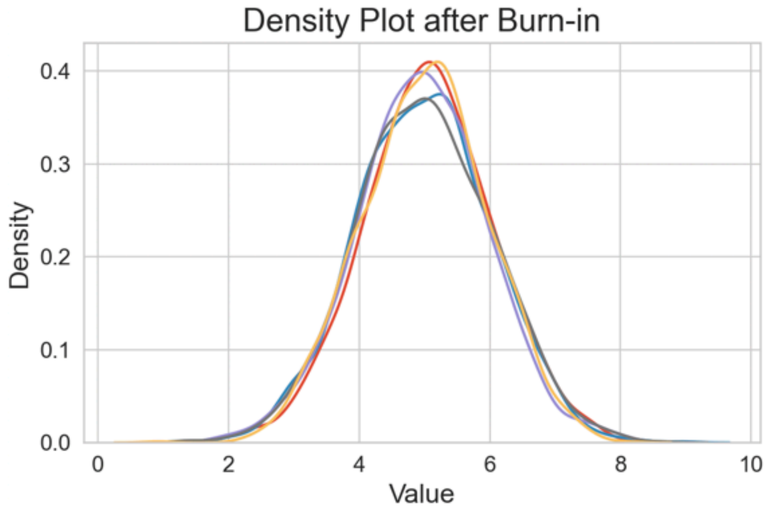
1. Très rare d'avoir une loi a posteriori avec "un désert" entre les modes
2. Si c'est le cas, il vaut mieux changer de modèle: Ici par ex, ajouter une variable latente (si possible) Z = 0 ou 1 pour distinguer les modes

INSEA

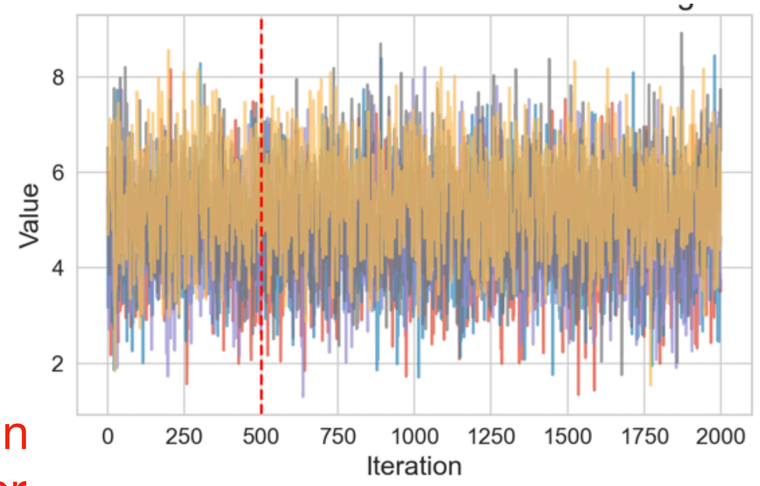
Lancer plusieurs chaînes avec des initialisations X_0 différentes:

Bon exemple

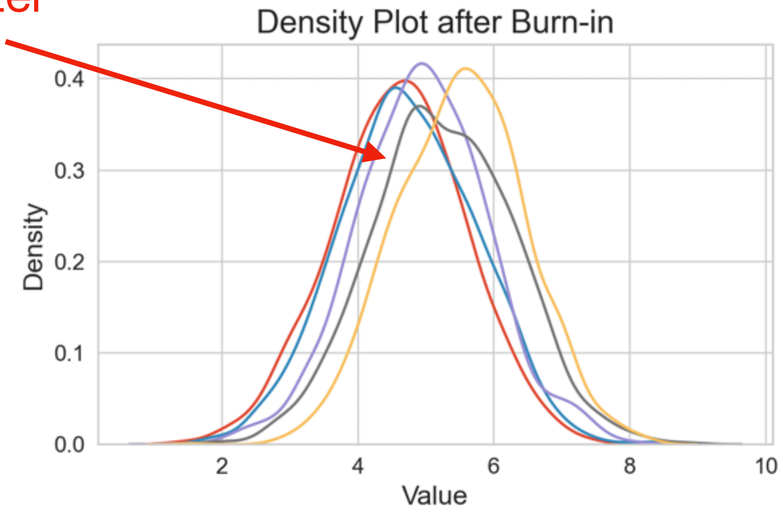




Mauvais exemple



Comment définir “un score” pour détecter ces différences ?



Metrics: R hat de Gelman-Rubin

Moyenne et variance de chaque chaîne

$$\text{Chaîne 1} : X_1^{(1)}, \ldots, X_n^{(1)} \longrightarrow \overline{\mathbf{X}}^{(1)} \quad , \quad \sigma^{\mathbf{2}(1)}$$

$$\vdots$$

$$\text{Chaîne j} : X_1^{(j)}, \ldots, X_n^{(j)} \longrightarrow \overline{\mathbf{X}}^{(j)} \quad , \quad \sigma^{\mathbf{2}(j)}$$

$$\vdots$$

$$\text{Chaîne m} : X_1^{(m)}, \ldots, X_n^{(m)} \longrightarrow \overline{\mathbf{X}}^{(m)} \quad , \quad \sigma^{\mathbf{2}(m)}$$

Moyenne des moyennes $\overline{\overline{\mathbf{X}}}$

Variance des moyennes “between” $B = \frac{n}{m-1} \sum_j^m (\overline{\mathbf{X}}^{(j)} - \overline{\overline{\mathbf{X}}})^2$

Moyenne des variances “within” $W = \frac{1}{m} \sum_{j=1}^m \sigma^{\mathbf{2}(j)}$

Pourquoi n ?

car $\mathbb{V}(\overline{\mathbf{X}}^{(j)}) = \frac{\sigma^2}{n}$

Si les chaînes convergent vers la même moyenne: $\frac{B}{n} \to 0$

Sinon $\frac{B}{n} \to b^\star > 0$

“R hat” de Gelman-Rubin (1992)

$$\widehat{R} \overset{\text{def}}{=} \sqrt{\frac{\frac{n-1}{n} W + \frac{1}{n} B}{W}} \to r \begin{cases} = 1 \text{ si } \frac{B}{n} \to 0 \\ > 1 \text{ sinon} \end{cases}$$

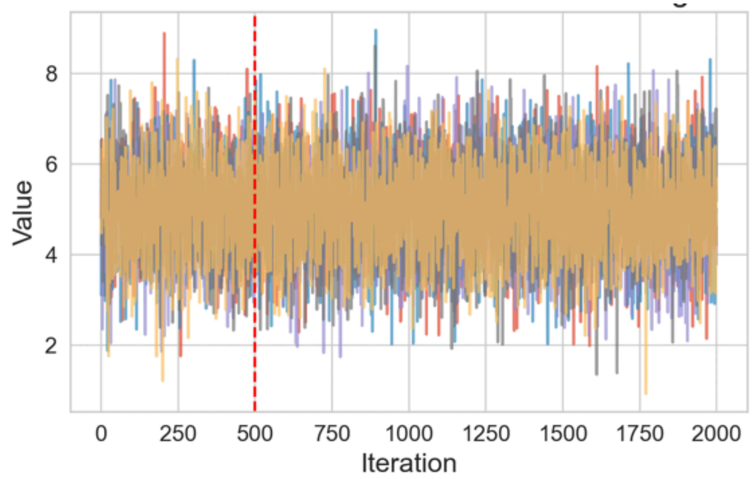
Détails en TD

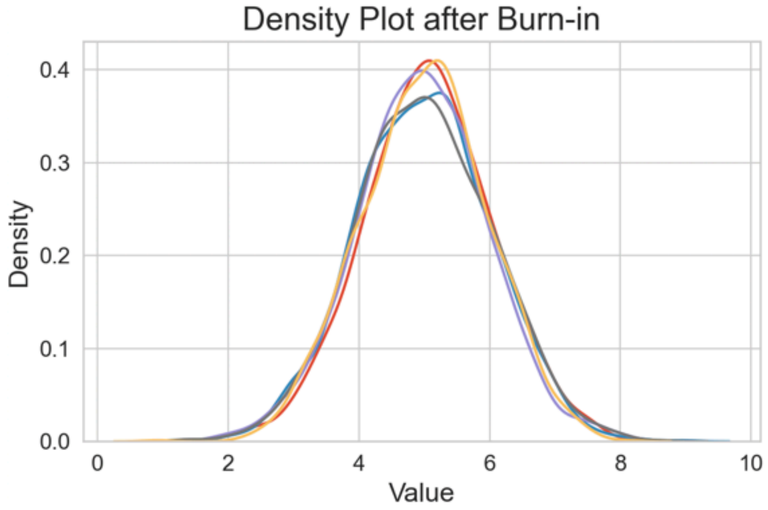
$$\widehat{R} \overset{\text{def}}{=} \sqrt{\frac{\frac{n-1}{n} \sigma^2_{within} + \frac{1}{n} \sigma^2_{between}}{\sigma^2_{within}}}$$

En pratique on veut R < 1.01

MCMC diagnostics in 1D

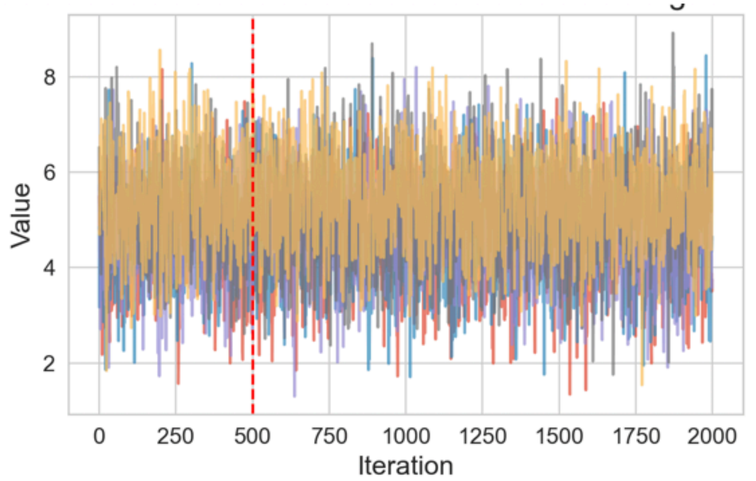
Metrics

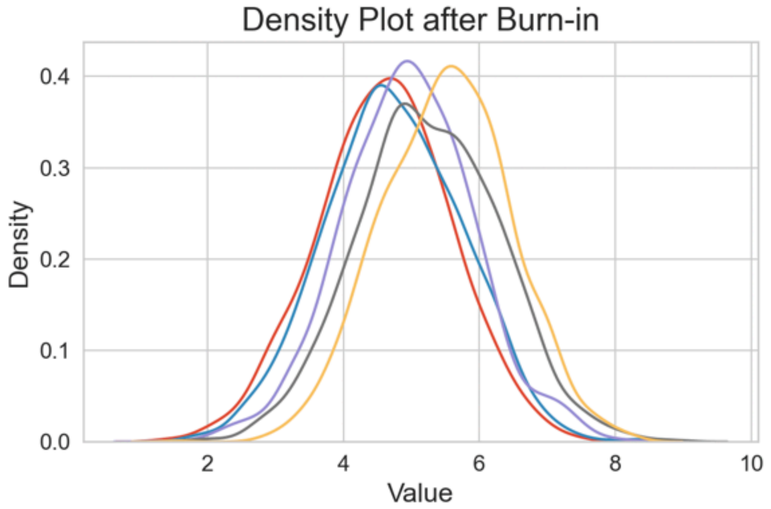




$\widehat{R}$ = 1.005

Converging





$\widehat{R}$ = 1.025

Not converging

On estime la moyenne d'une loi π avec deux estimateurs:

1. I_0 : Moyenne de N_0 échantillons i.i.d $\sim \pi$
2. I_1 : Moyenne de N_1 échantillons $\sim \pi$.

Quel est le meilleur estimateur ?

On suppose que $\mathbb{V}(I_0) = \mathbb{V}(I_1)$. On peut en déduire que $N_0 \geq N_1$ ou $N_0 \leq N_1$?

ESS: Effective sample size (Hastings 1970)

Étant donnée une MC $X_1, \ldots, X_{N_1}$, ESS correspond au nombre N_0 tel que: $\mathbb{V}(I_0) = \mathbb{V}(I_1)$

$$\text{ESS} = \frac{N_1}{1 + 2\sum_{t=1}^{\infty} \text{AutoCorr}(\text{lag t})} \leq N_1$$

En général, sauf cas spéciaux (autocorr < 0) (variables antithétiques à voir en Méthodes de simulation)

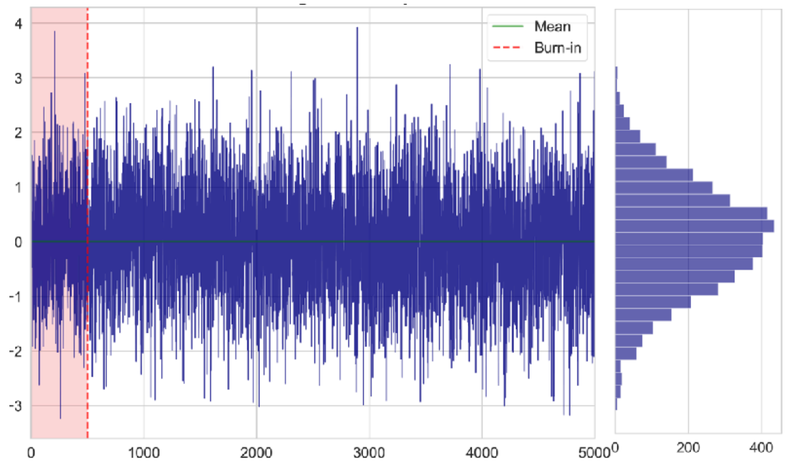
Preuve: il suffit d'égaliser la variance des deux estimateurs en supposant le régime stationnaire atteint

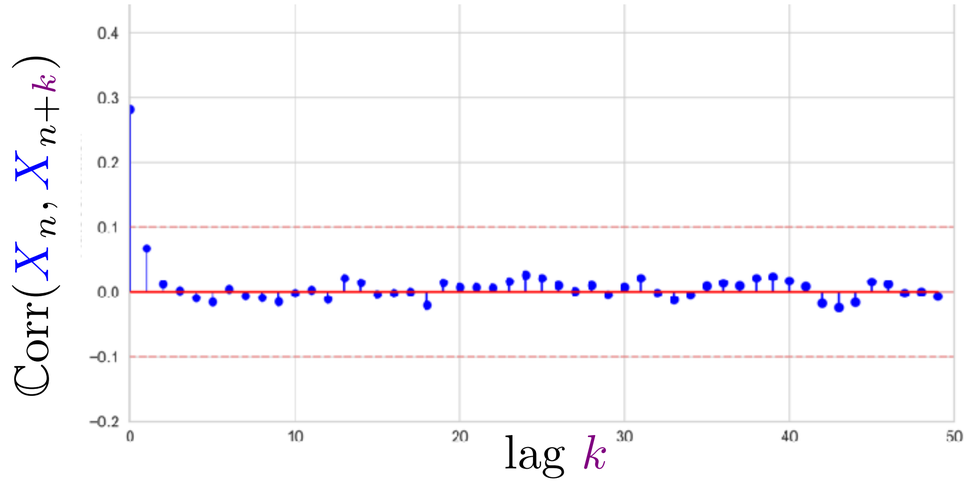
Remarques

1. On interprète ESS comme le nombre d'échantillons "réellement" utilisés (en moins ~ 400)
2. On calcule une version améliorée qui prend en compte plusieurs chaînes (Gelman, 2013)

MCMC diagnostics in 1D

Metrics

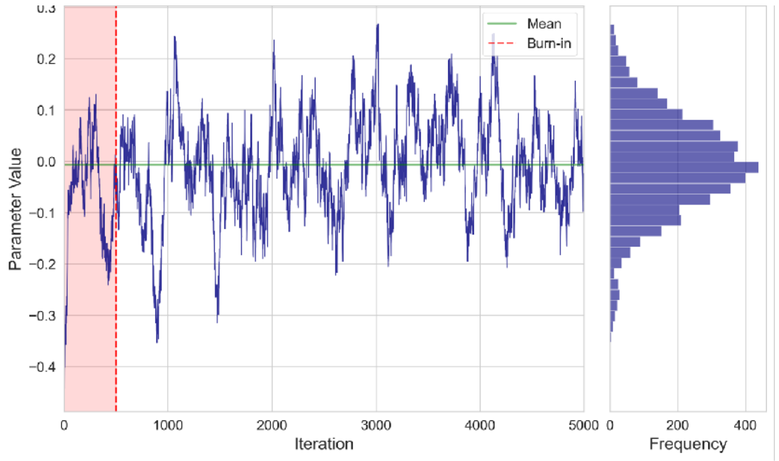


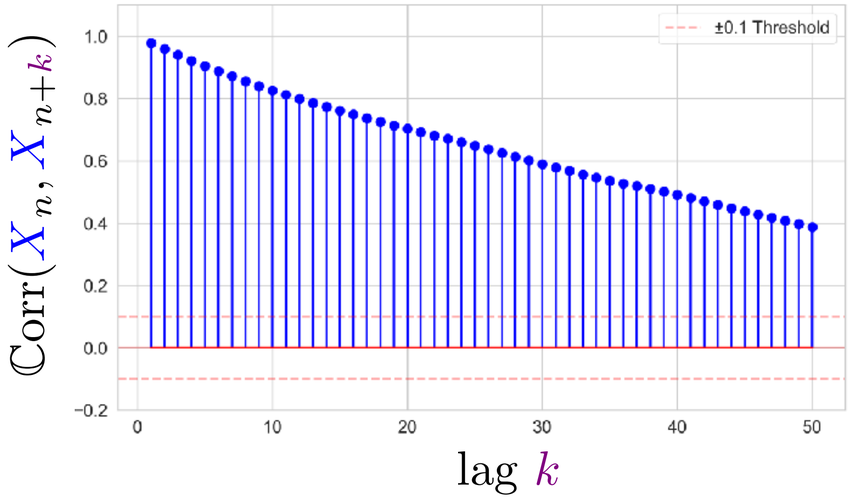


N = 5500

ESS = 5149

échantillons “efficaces”





N = 5500

ESS = 145 << N

Notre estimation a la qualité d’une moyenne i.i.d avec 145 échantillons seulement

Inconvénient: quantifie la qualité de l’estimation de la moyenne uniquement (ESS mean)

On peut définir d’autres ESS pour l’estimation des quantiles (ESS median, ESS Q95%..)

INSEA

MCMC diagnostics in 1D

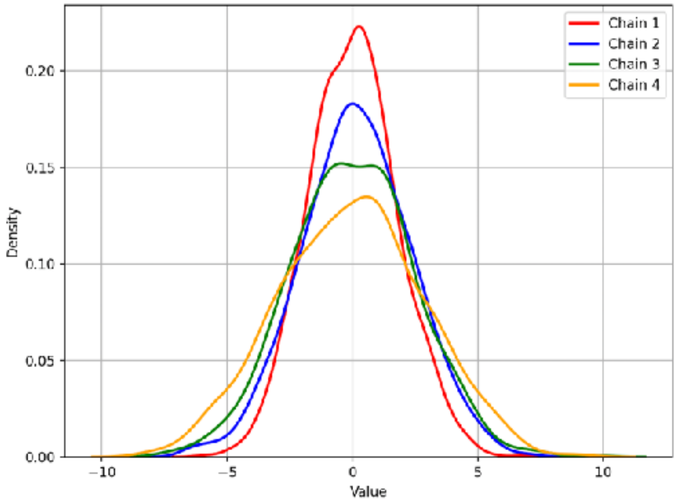
Metrics

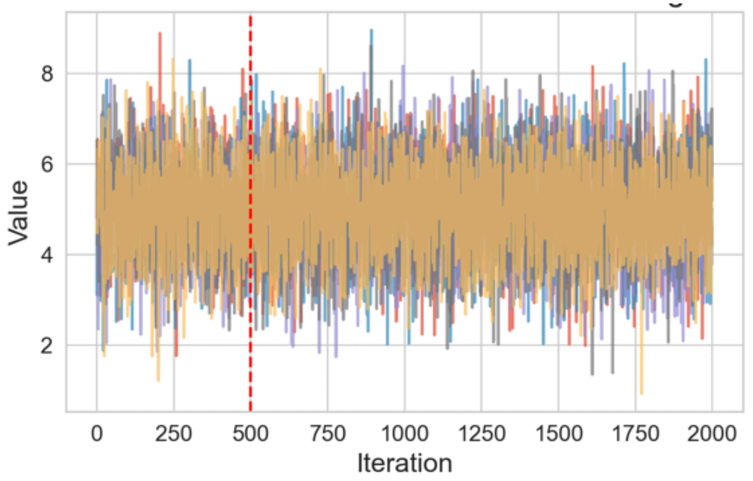
$$\widehat{R} \overset{\text{def}}{=} \sqrt{\frac{\frac{n-1}{n}\sigma^2_{within} + \frac{1}{n}\sigma^2_{between}}{\sigma^2_{within}}}$$

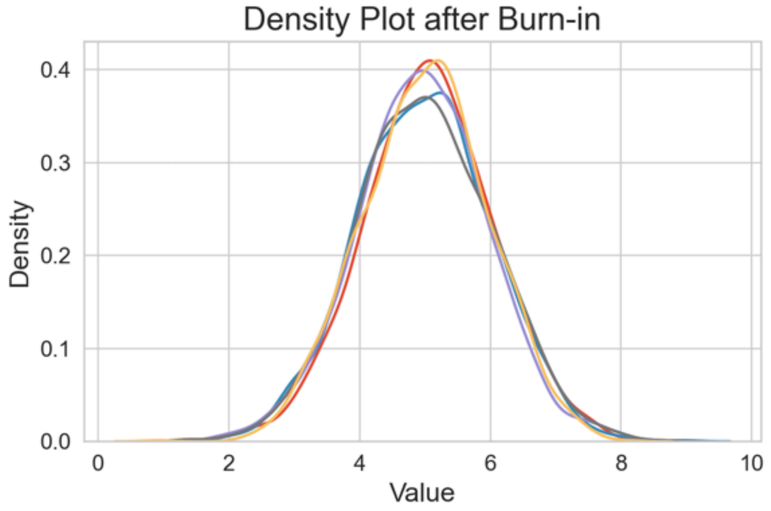
Inconvénient:
On peut avoir R ~ 1 alors que les chaînes sont très différentes. Comment construire un tel exemple ?

$$R \approx 1 \Leftrightarrow \sigma^2_{between} \approx 0$$

Il suffit d'avoir des chaînes différentes mais de mêmes moyennes !

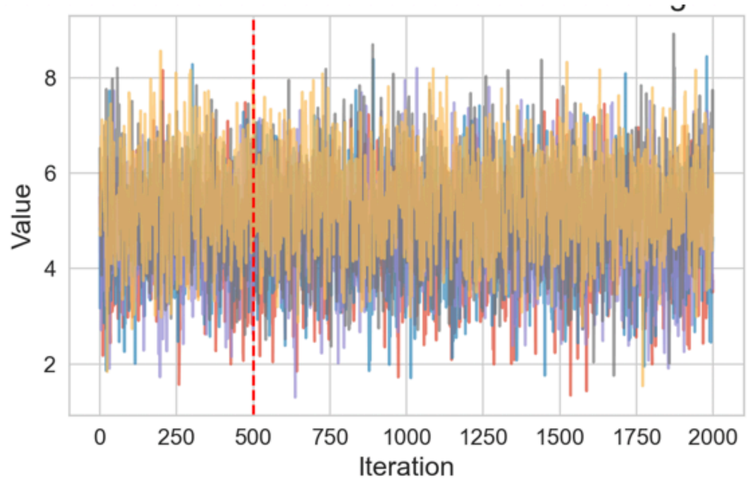


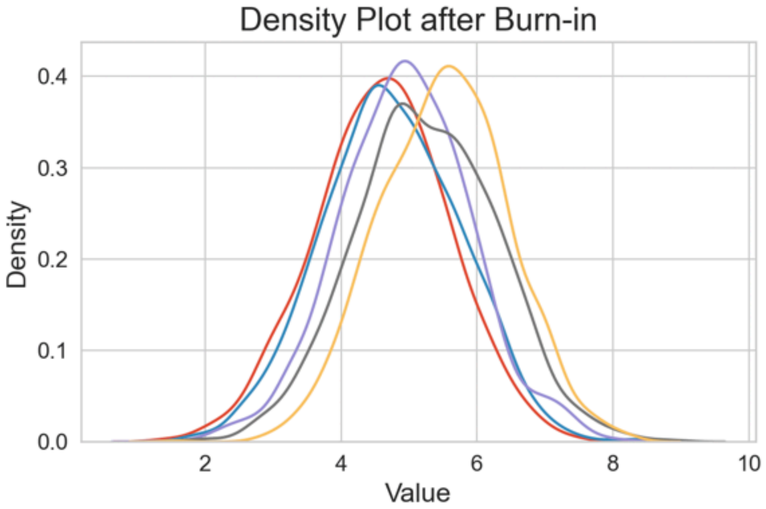




$\widehat{R}$ = 1.005 ESS = 4500

Converging





$\widehat{R}$ = 1.025

Not converging

“Rank-Normalization” (Vehtari et al, 2021)

1. Trier toutes les observations en calculant $R_i^{(j)} \overset{\text{def}}{=} \text{rank}(X_i^{(j)}) \in [|1, mn|]$
2. Les projeter sur $[0,1]$: $u_i^{(j)} \overset{\text{def}}{=} \frac{R_i^{(j)}}{mn}$
3. Si les chaînes sont bien “mélangées” alors $u_i^{(j)} \sim \mathcal{U}([0,1])$
4. Avec le théorème d’inversion $Z_i^{(j)} \overset{\text{def}}{=} \Phi^{-1}(u_i^{(j)}) \sim \mathcal{N}(0,1)$, Φ: Fct de répart. de $\mathcal{N}(0,1)$.
5. Calculer $\widehat{R}$ classique sur $Z_i^{(j)}$ au lieu des $X_i^{(j)}$.

Remarques:

1. Rhat plus sensible à la variabilité entre les chaînes
2. Moins sensible aux outliers
3. Cette transformation est utilisée pour calculer ESS également: “ESS bulk” (moyenne) et “ESS tail” (quantile 0.95)
4. Méthode par défaut en Python (Arviz)

Récap de la séance passée

1. Le but des algorithmes MCMC est de simuler des échantillons suivant une loi à densité connue $\propto f$.
2. Un algorithme MCMC crée une chaîne de Markov dont la distributions asymptotique est f (stationnaire).
3. Ces algorithmes sont principalement basés sur une exploration aléatoire et un mécanisme de rejet.
4. En pratique, on jette les k premiers échantillons pour être dans le régime stationnaire: burn-in / tuning.
5. En pratique, la chaîne obtenue n'est pas forcément convergente, pour cela on doit effectuer un diagnostic de convergence.
6. Chaque chaîne dépend de son initialisation, pour cela on lance plusieurs chaînes pour comparer leur distribution asymptotique.
7. Visuellement, les chaînes doivent être bien mélangées et des autocorrélations faibles (< 0.1).
8. On calcule le $\hat{R}$ de Gelman-Rubin qui mesure la variabilité entre ces chaînes. Il faut avoir un $\hat{R} < 1.01$.
9. On calcule l'ESS (effective sample size) qui donne une mesure du nombre d'échantillons "réllement utilisés".
10. ESS bulk donne le nombre d'échantillons pour l'estimateur de la moyenne.
11. ESS tail donne le nombre d'échantillons pour l'estimateur des quantiles 0.05 et 0.95 (le min des deux ESS).
12. En pratique il faut des ESS > 400 pour avoir une estimation de bonne qualité.

INSEA

Monte-Carlo Standard Error (MCSE): L'écart type d'un estimateur Monte-Carlo

$$\text{On suppose le régime stationnaire atteint } X_i \sim f. \quad \text{On pose } \sigma \overset{\text{def}}{=} \mathbb{V}(X_i)$$

L'estimateur de la moyenne est: $\frac{1}{n}\sum_{i=1}^{n} X_i$

Sa variance: $\frac{1}{n^2}\left[\sum_{i=1}^{n}\mathbb{V}(X_i) + 2\sum_{i<j}\mathbb{C}\text{ov}(X_i, X_j)\right] = \frac{\sigma^2}{n_{ESS}}$

$$\text{MCSE} = \frac{\hat{\sigma}}{\sqrt{n_{ESS}}}$$

MCSE de la moyenne

$$\hat{\sigma} = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$$

Avec le même principe, on peut définir le MCSE de la médiane, de l'écart type lui même etc …

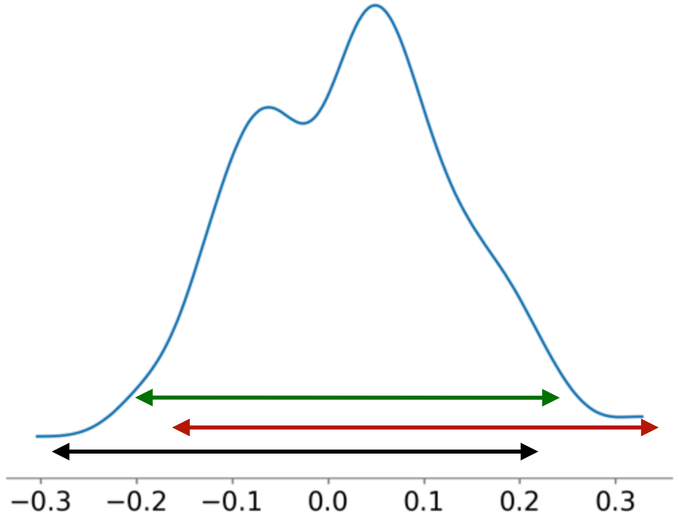
HDI: High density intervals

High density interval (HDI) ou *Credible Interval* (CI):
équivalent de l'intervalle de confiance en statistiques bayésiennes

Comment définir un intervalle à haute densité de niveau 90% ?

$$\text{Un intervalle } I \text{ tel que } \mathbb{P}_f(X \in I) \geq 0.9$$

$$\int_I f \geq 0.9$$



On peut en trouver plusieurs !

Lequel faut-il choisir ?

$$\text{HDI de niveau } 1-\alpha = \text{Le plus petit intervalle } I \text{ telle que } \mathbb{P}_f(X \in I) \geq 1-\alpha$$

Comment peut-on les comparer aux intervalles de confiance d'un point de vue statistique ?

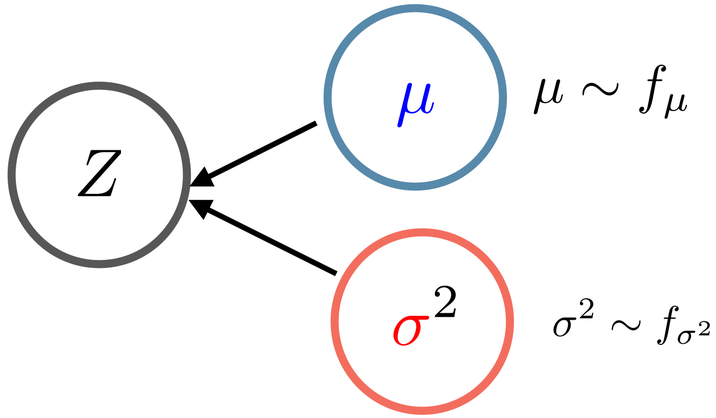
Application (Python)

Application (MCMC Diagnostics)

Soit $Z|\mu,\sigma^2 \sim \mathcal{N}(\mu,\sigma^2)$. μ et σ^2 ont des densités a priori indépendantes f_μ et f_{σ^2}.

1. Dessiner le graphe probabiliste du modèle.
2. Déterminez la loi a posteriori jointe $(\mu,\sigma^2)|Z$.
3. Déterminez les lois conditionnelles $\mu|\sigma^2,Z$ et $\sigma^2|\mu,Z$.
4. Quelles lois a priori f_μ et f_{σ^2} devrait-on prendre pour avoir des lois conditionnelles usuelles ?

$$Z|\mu,\sigma^2 \sim \mathcal{N}(\mu,\sigma^2)$$



Simuler 4 chaînes en numpy (Gibbs) et faîtes le diagnostique avec Arviz:

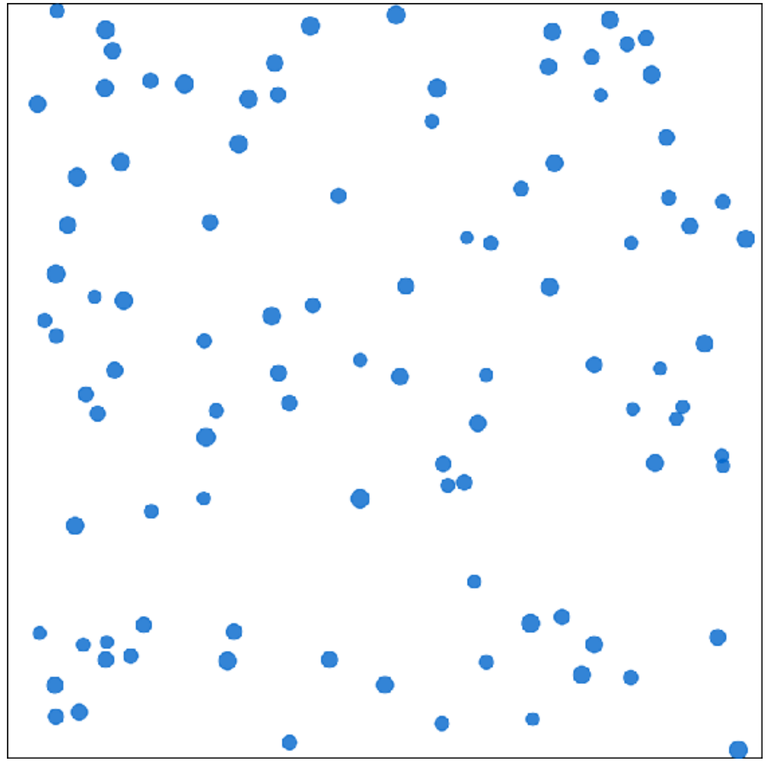
```
import arviz as az
chains_data_1d = .... # np array de taille (n_chains, n_samples)
i_data = az.convert_to_inference_data(dict(var_name=chains_data_1d)) # on construit l'objet inf_data nécessaire pour arviz

az.plot_trace(i_data) # visualiser les chaines / densités
az.plot_autocorr(i_data) # visualiser auto_corr
print(az.summary(kind="diagnostics")) # afficher les métriques ESS, R

az.plot_posterior(i_data) # on visualise la densité a posteriori
```

INSEA

II Méthodes de Monte-Carlo

1. Introduction
2. Markov Chain Monte-Carlo (MCMC)
3. Algorithmes MCMC avancés

Langevin Dynamics



Des particules d'un fluide dont le mouvement est expliqué par:

1. Des forces déterministes (gravitation) $F(X_t)$
2. Forces aléatoires (chaleur thermique) $G(X_t)$
3. Frottements / amortissement $-\gamma \frac{\mathrm{d}X_t}{\mathrm{d}t}$

Loi de Newton généralisée:

The Langevin Equation (1908)

$$m\frac{\mathrm{d}^2X_t}{\mathrm{d}t^2} = -\gamma\frac{\mathrm{d}X_t}{\mathrm{d}t} + F(X_t) + G(X_t)$$

$$m\frac{\mathrm{d}^2X_t}{\mathrm{d}t^2} \approx 0$$

La masse d'une particule est négligeable

$$F(X_t) = -\nabla U(X_t)$$

Définition de l'énergie potentielle U

$$G(X_t) = \sqrt{2\gamma kT}\xi_t \qquad \xi_t \sim \mathcal{N}(0,1)$$

Moyenne des forces aléatoires modélisée par une Gaussienne (thm. central)

k: constante de Boltzmann

Ainsi:

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = -\frac{1}{\gamma}\nabla U(X_t) + \sqrt{\frac{2kT}{\gamma}}\xi_t$$

INSEA

Langevin Dynamics

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = -\frac{1}{\gamma}\nabla U(X_t) + \sqrt{\frac{2kT}{\gamma}}\xi_t$$

À l'équilibre, la distribution de ces particules est celle de Boltzmann: $f(x) \propto \exp\left(-\frac{U(x)}{kT}\right)$

$$\Rightarrow \log\left(f(x)\right) = -\frac{U(x)}{kT} + \text{cte} \qquad \Rightarrow \nabla \log\left(f(x)\right) = -\frac{\nabla U(x)}{kT}$$

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = \frac{kT}{\gamma}\nabla \log\left(f(X_t)\right) + \sqrt{\frac{2kT}{\gamma}}\xi_t$$

Coefficient de diffusion d'Einstein

$$D \overset{\text{def}}{=} \frac{kT}{\gamma}$$

Stochastic differential equation (SDE)

$$\frac{\mathrm{d}X_t}{\mathrm{d}t} = D\nabla \log\left(f(X_t)\right) + \sqrt{2D}\xi_t$$

Pouvez-vous deviner un moyen de générer des échantillons suivant une densité f ?

Si on arrive à résoudre cette équation différentielle alors on aura un processus ~ f

Langevin Dynamics

$$\mathrm{d}X_t = D\nabla \log\left(f(X_t)\right)\mathrm{d}t + \sqrt{2D}\xi_t \mathrm{d}t$$

Comment résoudre cette SDE numériquement ?

$$\int_t^{t+h} \mathrm{d}X_s = D\int_t^{t+h} \nabla \log\left(f(X_s)\right)\mathrm{d}s + \sqrt{2D}\int_t^{t+h} \xi_s \mathrm{d}s$$

ξ_t: processus aléatoire $\sim \mathcal{N}(0,1)$

$$\approx$$

Avec $h \to 0$:

"Somme de h variables $\mathcal{N}(0,1)$ i.i.d en t" $= \sqrt{h}\xi_t$

$$X_{t+h} - X_t = Dh\nabla \log\left(f(X_t)\right)\mathrm{d}t + \sqrt{2Dh}\xi_t$$

Avec $D = 1$ et $\varepsilon = h \approx 0$, on définit la suite:

C'est la méthode d'Euler

$$X_{n+1} = X_n + \varepsilon\nabla \log\left(f(X_n)\right) + \sqrt{2\varepsilon}\xi_n \qquad \xi_n \sim \mathcal{N}(0,1)$$

La chaîne de Markov $(X_n)_n$ a pour distribution stationnaire une densité $\propto f$.

Langevin Dynamics

Unadjusted Langevin Algorithm (ULA)

$$X_{n+1} = X_n + \varepsilon \nabla \log\left(f(X_n)\right) + \sqrt{2\varepsilon}\xi_n \qquad \xi_n \sim \mathcal{N}(0,1)$$

Interpréter cette suite. Quel est le lien avec Metropolis ?

1. Dirige la suite vers les régions à haute densité en explorant l'espace
2. Si on enlève le terme du gradient, on retrouve l'étape d'exploration de Metropolis

Pas d'étape de rejet: risque de non-convergence due aux erreurs de discrétisation

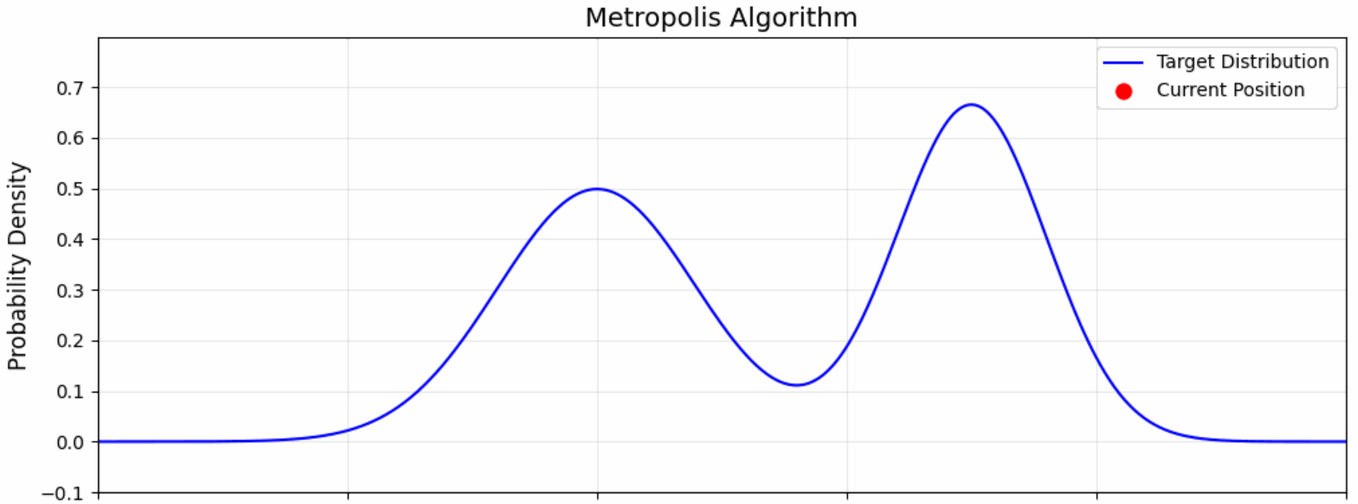
Metropolis Adjusted Langevin Algorithm (MALA)

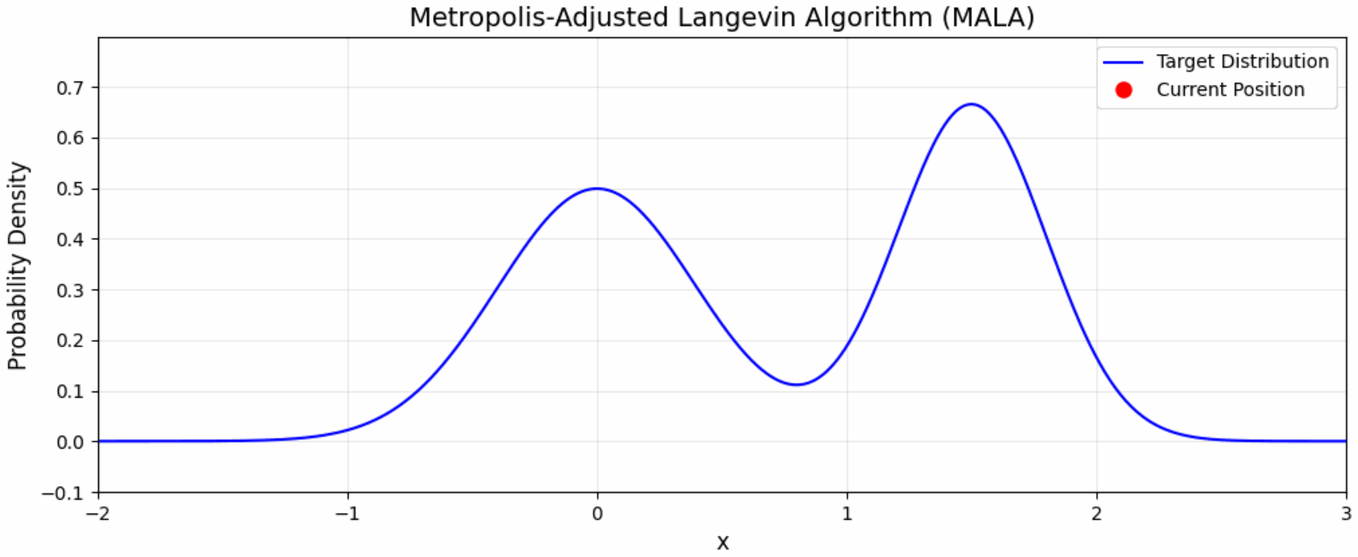
Soit f une densité de probabilité et $q(y|x) \propto \exp(-\frac{1}{4\varepsilon}\|y - x - \varepsilon\nabla \log f(x)\|^2)$. On suppose que X_n est déja généré. X_{n+1} est défini par:

1. Générer $y = X_n + \varepsilon\nabla \log f(X_n) + \mathcal{N}(0, 2\varepsilon)$.
2. Générer $u \sim \mathcal{U}([0,1])$.
3. Si $u < \min(1, \frac{f(y)q(X_n|y)}{f(X_n)q(y|X_n)})$ alors $X_{n+1} = y$, sinon $X_{n+1} = X_n$.

La chaîne $(X_n)_n$ obtenue admet une distribution stationnaire donnée par la densité f.

Metropolis vs MALA

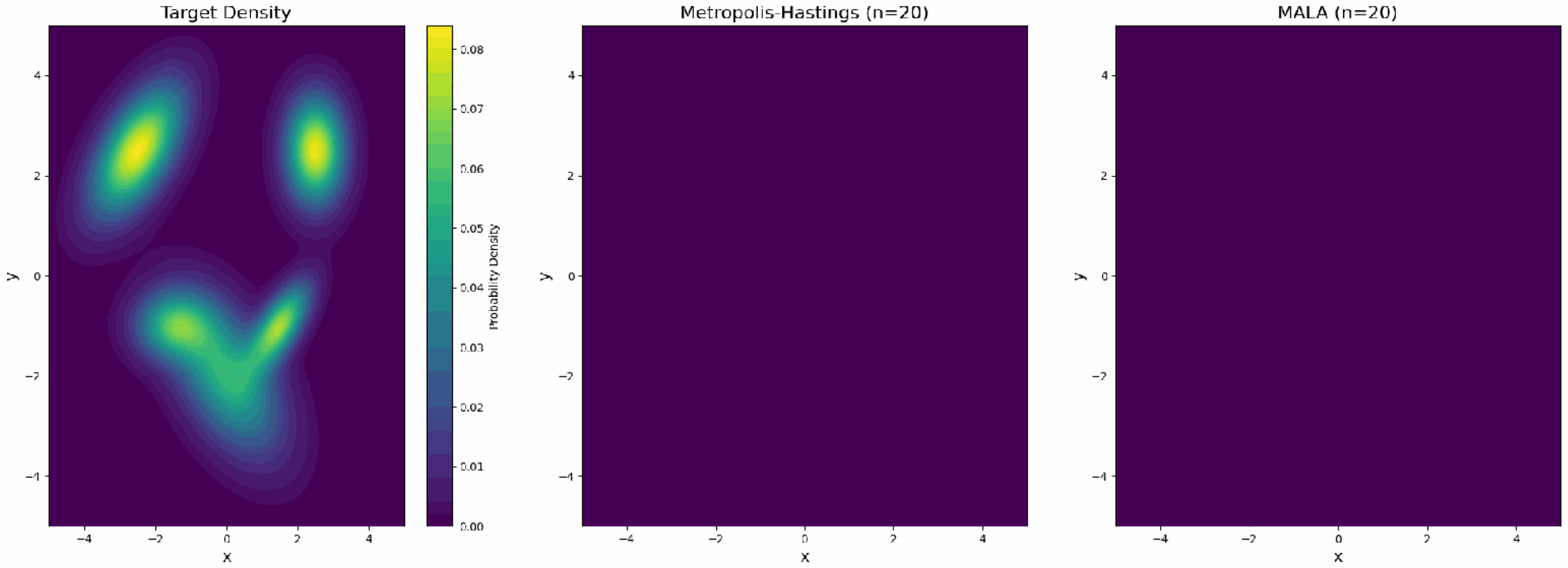




INSEA

(animated)

Metropolis vs MALA



Quand faut-il utiliser MALA au lieu de Metropolis ?

(animated)

Metropolis vs MALA

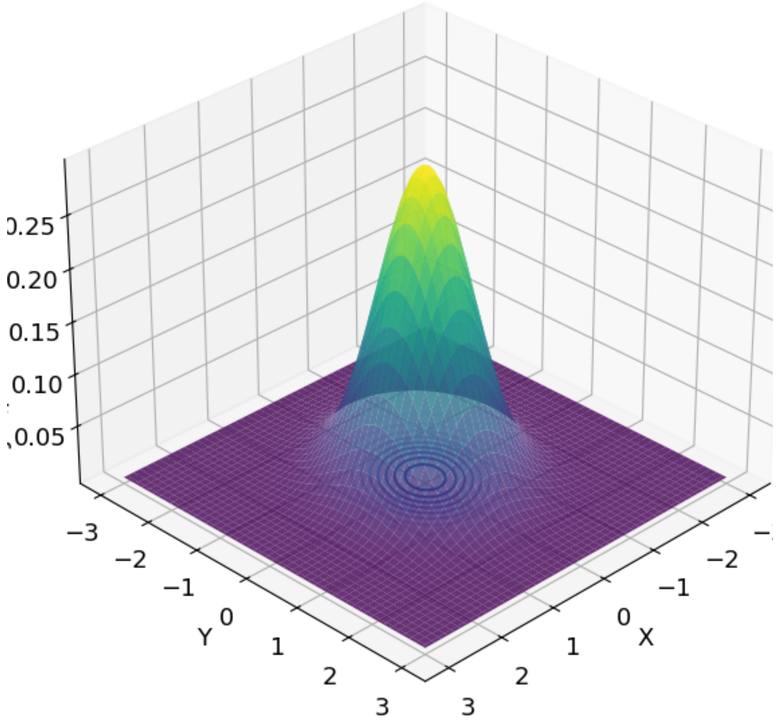
La densité de Boltzmann donne une intuition très importante

$$f(x) \propto \exp\left(-\frac{U(x)}{kT}\right)$$

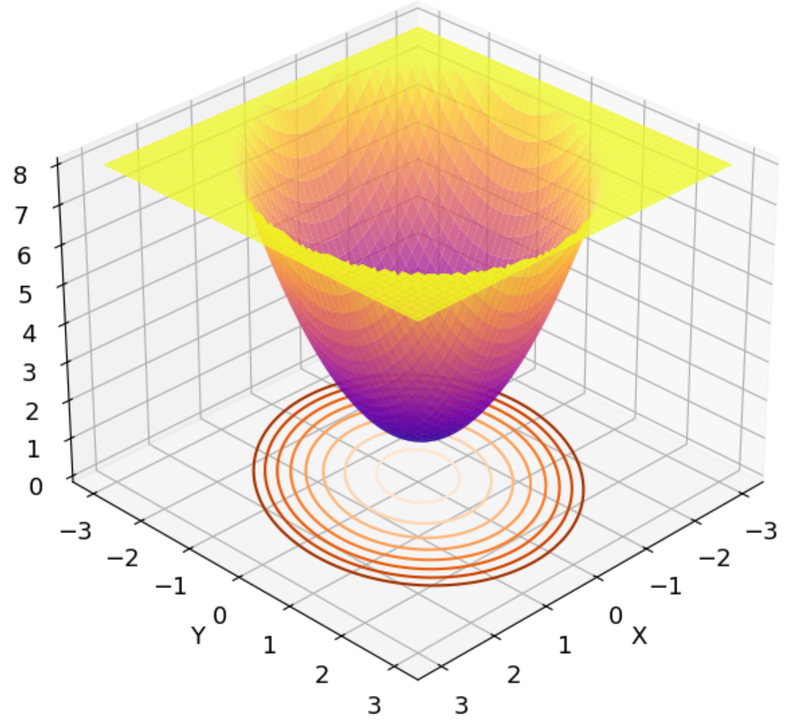
Comment interpréter “l’énergie potentielle” U ?

haute énergie potentielle U(x) = hauteur (gravité)

Densité f



Énergie potentielle -log(f)

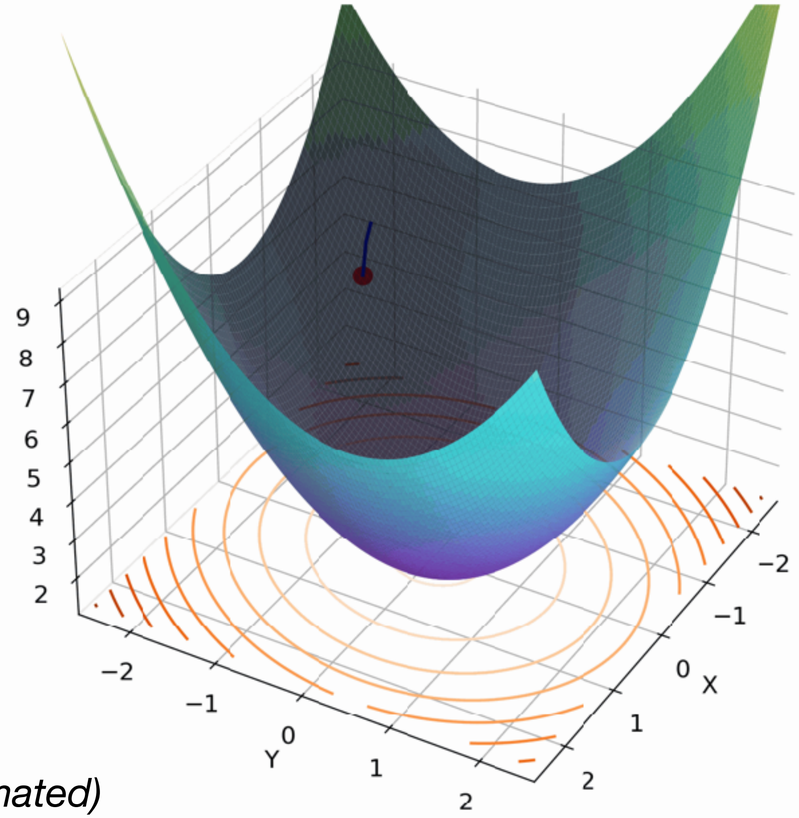


Hamiltonian Dynamics

En grande dimension, les algorithmes Metropolis / MALA explorent inefficacement l'espace.

Idée: au lieu d'explorer avec une position aléatoire x, simuler une vitesse initiale aléatoire.

En physique, on parle du moment: $\vec{p} = m\vec{v}$

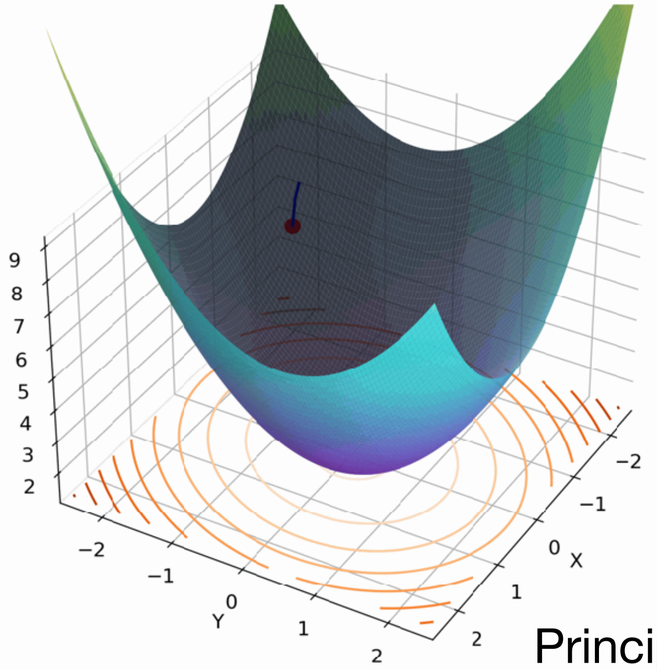


(animated)

Hamiltonian Monte-Carlo *(Duane 1987, Neal 1996)*

1. Simuler un vecteur de **moment p'** (direction et vitesse)
2. Déterminer la trajectoire
3. Suivre la trajectoire et s'arrêter (**nouveau x'**)
4. Accepter ou rejeter (**x'**, **p'**)
5. Répéter

Hamiltonian Dynamics



Hamiltonian Monte-Carlo *(Duane 1987, Neal 1996)*

1. Simuler un vecteur de **moment p'** (direction et vitesse)
2. Déterminer la trajectoire
3. Suivre la trajectoire et s'arrêter (**nouveau x'**)
4. Accepter ou rejeter (**x'**, **p'**)
5. Répéter

Avec un moment aléatoire p', comment obtenir la trajectoire ?

Principe de conservation de l'Énergie totale = Énergie potentielle + Énergie cinétique

Énergie totale = l'opérateur Hamiltonien: $H(x,p) = U(x) + K(p)$

Conservation de l'énergie dans le temps: $\frac{\mathrm{d}H}{\mathrm{d}t}(x,p) = 0$

Pour trouver la trajectoire, on discrétise l'équation différentielle à p' fixé avec un pas $\varepsilon \overset{\text{def}}{=} \Delta t$

On s'arrête après un nombre de pas égal à L

En pratique, L et ε sont difficiles à choisir

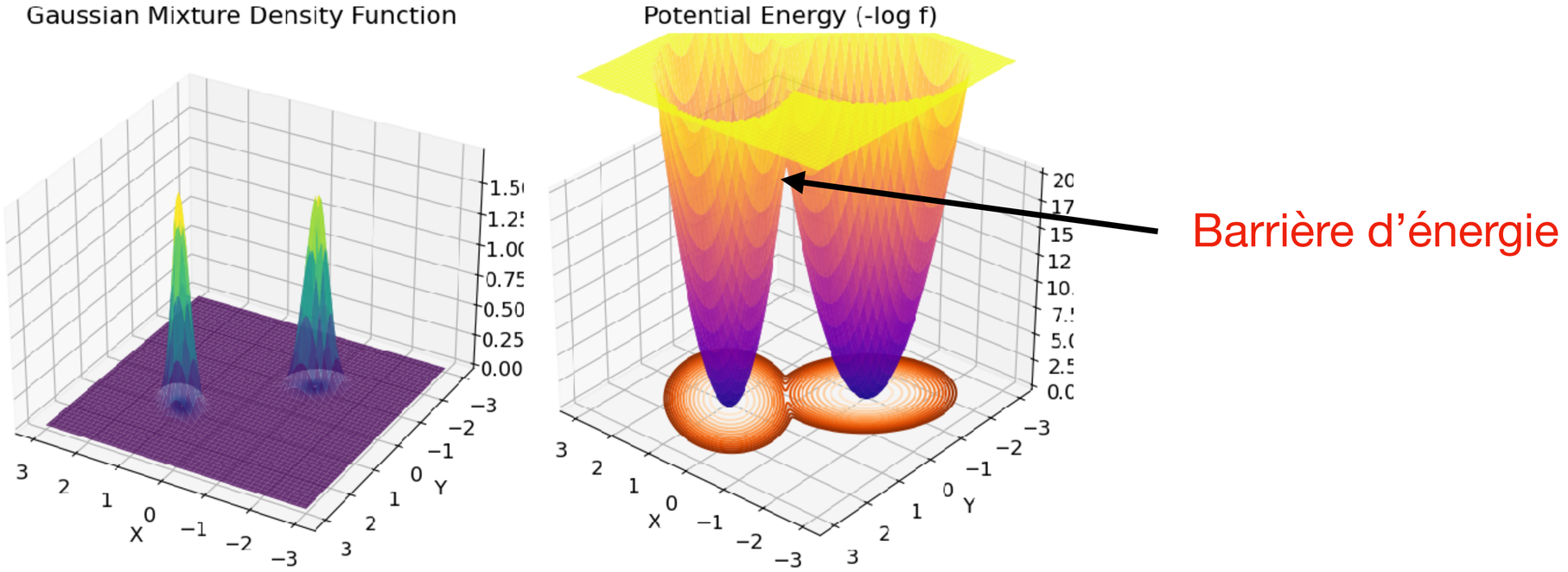
No-U-Turn-Sampler (NUTS) *(Hoffman and Gelman, 2014)*

HMC avec:

1. La variance d'exploration (pour générer les moments) est adaptée pendant le burn-in.
2. L est optimisé pour éviter un U-turn (demi-tour).
3. ε est modifié de façon adaptative pour un taux d'acceptation cible.

1. NUTS est l'algorithme "état-de-l'art" pour simuler en grande dimension (100-1000)
2. Aucun paramètre à choisir
3. Nécessite le gradient du log de la densité (distributions continues uniquement)
4. Par défaut dans PyMC pour les variables continues

Hamiltonian Dynamics



Est-il possible qu'une chaîne simulée par NUTS ne découvre qu'un seul mode de cette distribution ?

Oui ! Pour sortir de l'un des deux puits, il faut simuler un moment (vitesse) très grand

On veut simuler une loi a posteriori $(\alpha, m|\text{data})$ avec α continue et m discrète. Comment faire ?

Algorithme de Gibbs:

Simuler $\alpha_n \sim \alpha|m_n$ avec une itération de NUTS / MALA

Simuler $m_{n+1} \sim m|\alpha_n$ avec une itération de Metropolis-Hastings

Géré automatiquement par PyMC: à chaque variable (ou groupe de variables) est associé le meilleur algorithme (exemple en TP).

Résumé

Algorithme	Méthode	Utilité
Gibbs	Alterne entre les lois conditionnelles	Lois conditionnelles simples Alterner entre des variables discrètes / continues
Metropolis	Marche aléatoire avec densité de proposition symétrique	Densité non différentiable symétrique
Metropolis-Hastings	Marche aléatoire avec densité de proposition quelconque	Densité non différentiable (discrète) et asymétrique comme la LogNormal, Gamma
Metropolis Adjusted Langevin Algorithm (MALA)	Marche aléatoire Gaussienne guidée par le gradient de la densité	Variable continue en faible dimension
Hamiltonian Monte-Carlo (HMC)	Exploration avec des moments aléatoires et des trajectoires à énergie totale constante	Variable continue en grande dimension
No-U-Turn Sampler (NUTS)	HMC avec des trajectoires optimisées	Variable continue en grande dimension, pas de paramètres à fixer manuellement

Chapitre 3. Applications et thématiques avancées

1. Modèles Bayésiens hiérarchiques (Assurance / Biostats)
2. Bayesian Machine learning

Bibliography

- Gelman, A et al. (2013). *Bayesian Data Analysis, Third Edition.* Chapman and Hall/CRC.
- Kruschke, J. K. (2010). *Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan.* Academic Press.
- Robert, C. P., & Casella, G. (2004). *Monte Carlo Statistical Methods.* Springer.
- Castillo, I (2017) *Introduction aux statistiques baysiennes*, UPMC.
- Vehtari, A. et al. (2021). *Rank-normalization, folding, and localization: An improved $\hat{R}$ for assessing convergence of MCMC.*
- Vets D. and Knudson C. (2020). *Revisiting the Gelman-Rubin Diagnostic.*
- Betancourt, M. (2018). *A Conceptual Introduction to Hamiltonian Monte Carlo.*
- Hoffman, M. D., & Gelman, A. (2014). *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo.* Journal of Machine Learning Research, 15(1), 1593-1623.
- Gelman, A., & Rubin, D. B. (1992). *Inference from Iterative Simulation Using Multiple Sequences.* Statistical Science, 7(4), 457-472.
- Hastings, W.K. (1970). *Monte Carlo Sampling Methods Using Markov Chains and Their Applications.* Biometrika, 57 (1): 97:109.
- Metropolis, N. et al. (1953). *Equation of State Calculations by Fast Computing Machines.* The Journal of Chemical Physics, 21(6), 1087-1092.

INSEA