PHYS-338: Physique Statistique **2024-2025**

Lecture 7: Échantillonage et Algorithmes de Monte-Carlo

Professeur: Florent Krzakala *Scribes: o*

7.1 Motivation

Dans les chapitres précédents, on s'est intéressé au formalisme de la mécanique statistique en étudiant des systèmes simples où on a le plus souvent supposé que les particules n'interagissaient pas entre elles. Il va de soi que ces suppositions ne tiennent plus dans la plupart des systèmes réels et que dans ces derniers, les particules interagissent entre elles. Malheureusement, lorsque les particules sont en interaction, la fonction de partition devient considérablement plus compliquée a calculer et il sera donc plus raisonnable de l'estimer numériquement plutôt que de la calculer directement.

Pour illustrer comment on peut estimer une fonction de partition, on considère un système à N particules avec un hamiltonien $\mathscr{H}$ qui est une fonction "compliquée" des configurations.[1] La fonction de partition s'écrit alors :

$$Z(N,V,T) = \int d\vec{x}_1 \dots d\vec{x}_N \; e^{-\beta \mathscr{H}(\vec{x}_1,\dots,\vec{x}_N)}. \tag{7.1}$$

Souvent ce qui nous intéresse dans un système, c'est les valeurs moyennes d'une observable O. Cette valeur se calcule suivant :

$$\langle O \rangle = \frac{1}{Z} \int d\vec{x}_1 \dots d\vec{x}_N \; O(\vec{x}_1,\dots,\vec{x}_N) e^{-\beta \mathscr{H}(\vec{x}_1,\dots,\vec{x}_N)}. \tag{7.2}$$

Si on est capable de calculer notre observable pour $\mathscr{N}$ configurations et en supposant que $\mathscr{N}$ est assez grand, on sait par la loi des grands nombres, qu'on peut écrire la valeur moyenne d'une observable comme

$$\langle O \rangle \approx \frac{1}{\mathscr{N}} \sum_{i=1}^{\mathscr{N}} O\left(\vec{x}_1^{(i)},\dots,\vec{x}_N^{(i)}\right), \tag{7.3}$$

où les $\left(\vec{x}_1^{(i)},\dots,\vec{x}_N^{(i)}\right)$ dénotent des points de l'espace des configurations. On pose

$$X^{(i)} := \begin{vmatrix} \vec{x}_1^{(i)} \\ \vdots \\ \vec{x}_N^{(i)} \end{vmatrix}, \tag{7.4}$$

où $X^{(i)}$ dénote alors la i-ème configuration étudiée. Notre objectif principal est donc d'échantillonner notre espace proportionnellement à la probabilité associée à chacune des configurations. En d'autres mots, on cherche à générer des variables aléatoires $X^{(i)}$ telles que

$$X^{(i)} \sim \frac{1}{Z} e^{-\beta \mathscr{H}\left(\vec{x}_1^{(i)},\dots,\vec{x}_N^{(i)}\right)}. \tag{7.5}$$

Toute cette procédure de génération de variables aléatoires qu'on appelle plus communément échantillonage est résolue par les simulations Monte-Carlo. Mais avant ça, on va s'intéresser à comment échantilloner des distributions plus simples.

[1] L'idée se généralise facilement si on considère aussi les impulsions.

7.2 Échantillonage de lois discrètes unidimensionnelles

Nous allons étudier un système à N états avec un ensemble de probabilités associées à chaque état $\{p_i\}_{i=1}^{N}$. Par exemple, cherchons à répondre à la question “Que faire ce soir ?" et on imagine les réponses suivantes :

1. “Fêter Halloween" avec $p_1 = 0.5$
2. “Lire un livre" avec $p_2 = 0.1$
3. “Aller dans un bar" avec $p = 0.2$
4. “Regarder la télévision" avec $p = 0.2$

Dans les sections suivantes, nous illustrerons cet exemple avec des différents algorithmes. Avant de poursuivre, nous devons supposer qu'un ordinateur peut calculer des réalisations d'une loi de distribution uniforme.

7.2.1 Algorithme de Rejet (Rejection Sampling)

Pour implémenter un algorithme de rejet, on dresse un tableau de taille 1 par 1 contenant des rectangles représentant chaques évènements (ou états). Ces rectangles doivent avoir une aire proportionnelle à la probabilité de réalisation de l'évènement qu'ils représentent. Un exemple d'un tel montage est représenté à la figure 7.1.

L'idée de l'algorithme de rejet est de générer deux variables aléatoires indépendantes X, Y suivant une loi de distribution uniforme $U[0,1]$ et de regarder où la coordonée d'une réalisation de ces deux variables aléatoires (x, y) a atteri sur le tableau dressé précédemment.

Un exemple d'un tel algorithme pourrait être :

Algorithme 1.

```
ok = 0
while(ok = 0){
     x ~ U(0,1)
     y ~ U(0,1)
     if ((x,y) ∈ Histogram){ok = 1}
}
Return(int) 4x+1
```

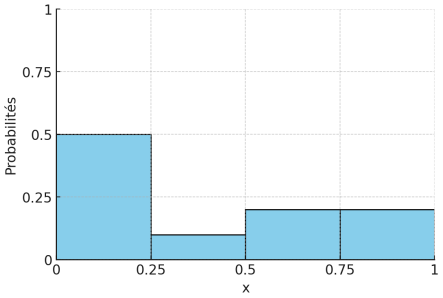

Figure 7.1: Illustration d'un tableau pour un algorithme de rejet.

Pour améliorer cet algorithme on va chercher à réduire le nombre de rejets. On remarque alors facilement que pour ce faire, on peut simplement reduire l'intervalle de la loi de distribution de la variable aléatoire Y à $U\left(0, \max(p_i)\right)$. Procéder ainsi nous permet de retirer toute l'aire inutile qui se trouve au dessus des rectangles.

7.2.2 Algorithme de la Cummulative (Tower sampling)

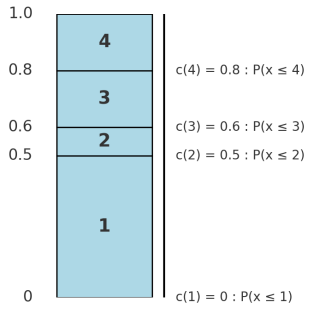

Figure 7.2: Illustration d'un tableau pour un algorithme de rejet.

Pour implémenter un algorithme de la cumulative, on dresse un axe avec des segments mis bout-à-bout où ces segments on une taille égale à la probabilité de l'évènement qu'ils représentent. Un exemple est donné à la figure 7.2. L'idée de l'algorithme de la cumulative est de générer une variable aléatoire $X \sim U(0,1)$ et ensuite regarder sur quel segment une réalisation de cette variable aléatoire est arrivée. Pour ce faire on doit également définir les bords des segments $c(i)$ comme illustré sur la figure ci-contre.

Un exemple d'un tel algorithme pourrait être :

Algorithme 2.

```
x ~ U[0,1]
ok = 0
i = 2
while(x > c(i)){ i ← i + 1 }
Return i
```

L'algorithme de la cumulative a l'avantage sur l'algorithme de rejet de ne pas avoir à générer plusieurs fois une variable aléatoire. On dira alors que l'algorithme de la cumulative est "rejection free" ce qui est computationellement nettement meilleur.

On remarque également qu'une version plus rapide de l'algorithme 2 consite à faire une recherche de la réalisation de la variable aléatoire par dichotomie.

7.2.3 Algorithme de Walker

Cet algorithme, un peu plus dur à implémenter, cherche à construire un tableau avec une aire proportionnelle à la propabilité de l'évènement associé mais cette fois sans avoir de possibilité de rejet. La figure 7.3 illustre un tel montage.

Il nous suffit, une fois le tableau dressé, de générer deux variables aléatoires X, Y et noter la surface sur laquelle elle est arrivée.

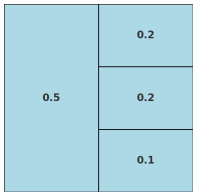

Figure 7.3: Illustration de l'algorithme de Walker en 1D avec les probabilités données.

7.3 Echantillonage de loi continue unidimensionnelle

Dans cette section, on s'intéresse à différentes méthodes pour échantillonner des variables aléatoires X suivant une distribution continue et unidimensionnelle $X \sim P_X$, où P_X est une densité de probabilité.

7.3.1 Algorithme de rejet (Rejection sampling)

Imaginons qu'on connaisse le graphe de la densité de probabilité $P_X(x)$. On génère deux variables aléatoires X, Y telles que X est uniforme sur l'intervalle qu'on considère, *i.e.* $X \sim U(x_{\min}, x_{\max})$, et $Y \sim U(0,c)$, où $c \in \mathbb{R}_{>0}$ est un paramètre de notre implémentation.

L'algorithme de rejet consiste à observer où le couple (x, y) d'une réalisation de ces deux variables aléatoires arrive sur le graphe. Si le couple désigne un point sous le graphe on accepte sinon on rejette.

Un exemple du graphe que donnerait une implémentation de l'algorithme de rejet est donné à la figure 7.4.

Algorithme 3.

```
ok = 0
while(ok = 0){
     x ~ U[0,1]
     y ~ c · U[0,1]
     if (y < P_X(x)){ok = 1}
}
Return x
```

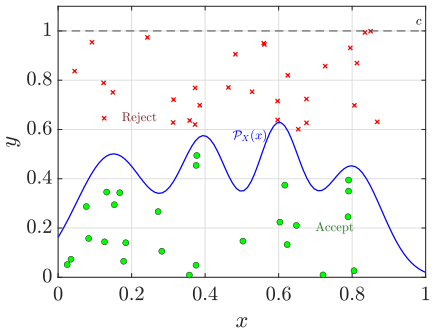

Figure 7.4: Illustration d'un algorithme de rejet avec $c = 1$.

On peut noter que faire tourner l'algorithme un grand nombre de fois nous donne l'aire sous la courbe, qui n'est autre que la valeur de l'intégrale $I = (c-0)(x_{\max} - x_{\min})P(\text{accept})$.

Comme pour dans le cas discret, on veut réduire à un minimum le nombre de rejets de cet algorithme. Pour ce faire on peut penser au premier abord à restreindre notre variable aléatoire Y a suivre une loi uniforme sur l'intervalle $U(0, c)$, où $c = \max(P_X(x))$ comme nous l'avions vu pour échantillonner une loi discrète. Il existe cependant une méthode qui peut s'avérer encore meilleure pour certaines situations :

On suppose qu'on sait générer des variables aléatoires selon une autre loi de distribution continue que la loi uniforme que l'on notera Q_X et on suppose aussi que $Q_X(x) \geq P_X(x), \forall x$. Alors on voit simplement qu'on peut générer $X \sim U(0,1)$ et $Y \sim Q_X$ et alors regarder si le couple (x, y) est arrivé sous la courbe de P_X.

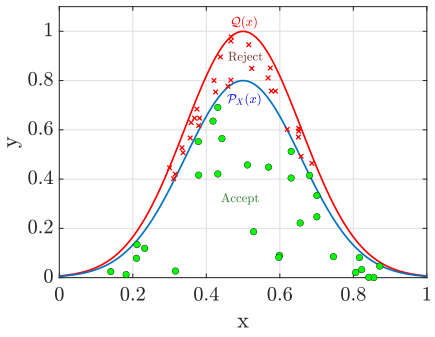

Figure 7.5: Illustration d'un algorithme de rejet avec $c = \max(Q(x))$.

La figure 7.5 illustre un exemple d'échantillonnage efficace de P_X, où on a pris Q_X comme étant une Gaussienne.

Algorithme 4.

```
ok = 0
while(ok = 0){
     x ~ U[0,1]
     y ~ Q(x)
     if (y < P_X(x)){ok = 1}
}
Return x
```

7.3.2 Méthode cumulative inverse

Soit $x \sim P(x)$. On cherche une transformation T tel que $x = T(u) \sim P(\omega)$, $T^{-1}(T(x)) = x$ (*i.e.* T admet un inverse unique) avec u qui suit une loi uniforme. Pour cela nous utilisons la cumulative $F_X(x)$:

$$F_X(x) = \int_{-\infty}^{x} P_X(\theta)\mathrm{d}\theta = P(X \leq x) = P(T(u) \leq x) = P(u \leq T^{-1}(x)) = T^{-1}(x) \tag{7.6}$$

Où nous avons utilisé pour la dernière égalité le fait que, pour une loi uniforme $u[0,1]$, $P((x \sim u[0,1]) \leq a) = a$.

Ainsi, si on prend $x = T(u)$ avec $T(\cdot) = F_X^{-1}(\cdot)$ alors $x \sim P_X(x)$.

Exemple: Prenons $x \sim e^{-x}, x \geq 0$, alors on a $P_X(x) = e^{-x}$ et $F_X(x) = 1 - e^{-x}$ On en tire

$$u = 1 - e^{-x} \implies x = -\log(1-u)$$

Ainsi, si on prend u uniforme, alors x suivra bien la distribution exponentielle. Cela montre que l'on sait échantillonner les lois exponentielles.

R Si on avait pris $x = \log(u)$ cela aurait très bien fonctionné car $1-u$ et u sont également distribués uniformément. Attention cependant numériquement: lorsque u est proche de 0 ou 1, il peut y avoir des problèmes dus au logarithme. Il faut ainsi choisir judicieusement selon le problème si l'on choisit $x = \log(u)$ ou $x = \log(1-u)$

7.4 Transformations utiles

Dans cette section, quelques transformations de loi de probabilité permettant d'échantillonner sur des espaces importants vont être mises en avant :

7.4.1 Le cercle

Pour ce premier format, l'objectif est d'échantillonner des points à l'intérieur du cercle unité, dans ce but nous allons lister trois méthodes dont deux fonctionnent :

Méthode 1

Pour le premier algorithme on va générer deux variables aléatoires indépendantes X, Y suivant une loi de distribution uniforme $U[-1,1]$ et puis on garde ceux qui tombent dans le cercle unité.

L'algorithme est le suivant :

Algorithme 1.

```
x ~ U(-1,1)
y ~ U(-1,1)
if x^2 + y^2 ≥ 1 on rejette
```

Cette méthode permet aussi de calculer la valeur de π en utilisant le fait que la probabilité de tomber dans le cercle est $P_{in} = \frac{\text{aire cercle}}{\text{aire carré}} = \frac{\pi}{4} = \frac{N_{in}}{N_{total}}$ où N_{in} est le nombre de points tombés dans le cercle et N_{total} le nombre total de points, cela implique:

$$\pi = \frac{4N_{in}}{N_{total}} \tag{7.7}$$

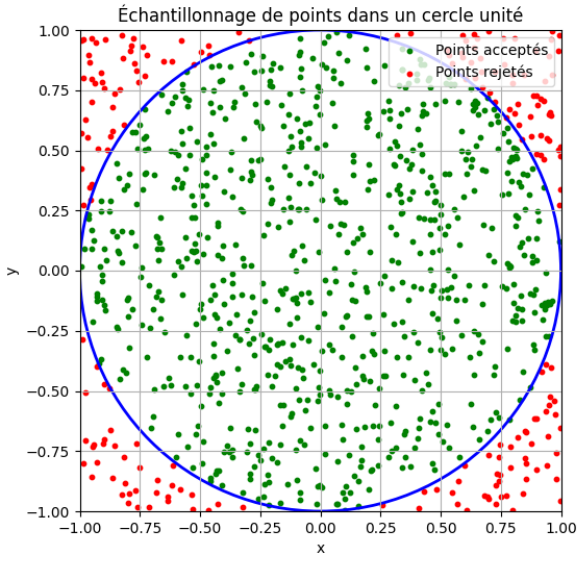

Figure 7.6: Échantillonnage du cercle avec rejet

Méthode 2

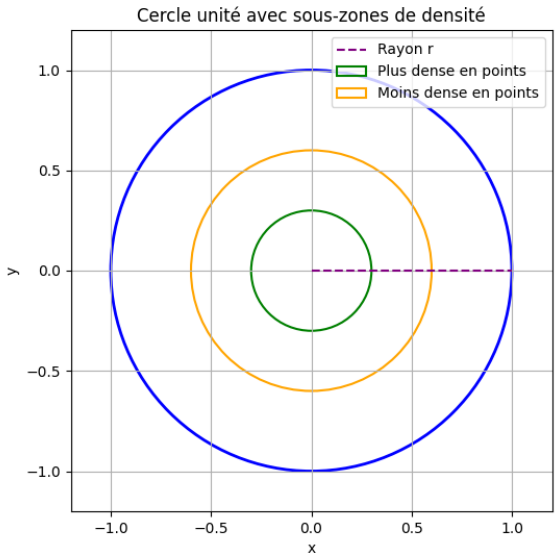

Figure 7.7: Echantillonnage du cercle en coordonnées polaires

Pour le second algorithme on va générer des points en échantillonnant avec r suivant une loi de distribution uniforme $U[0,1]$ et $\theta \sim$ U[0,2π] cela nous donne des couples de points dans le cercle $(x,y) = (rcos\theta, rsin\theta)$:

Algorithme 2.

```
r ~ U(0,1)
θ ~ U(0,2π)
Return (rcosθ,rsinθ)
```

En échantillonnant de la sorte, on voit tout de suite que plus la valeur de r est proche de 1 moins les points seront denses. Donc l'échantillonnage n'est pas équitablement répartie sur le cercle entier

⚠ Attention cette méthode ne fonctionne pas.

Méthode 3

La troisième méthode qui permet bien d'échantillonner uniformément le cercle est de partir de la loi jointe de x et y :

$$P(x,y) = \frac{1}{\pi}, \text{ si } x^2+y^2 \leq 1 \tag{7.8}$$

$$= 0, \text{ sinon} \tag{7.9}$$

On peut donc écrire en passant en coordonnées polaires :

$$P(x,y)dxdy = \frac{1}{\pi}dxdy = \frac{r}{\pi}drd\theta \tag{7.10}$$

On trouve donc ensuite :

$$P(x,y)dxdy = \left(\frac{1}{2\pi}d\theta\right)(2rdr) = U(0,2\pi) \times P_r(r)drd\theta \tag{7.11}$$

Il faut trouver un moyen d'échantillonner une loi $P_r(r) = 2r$, cela se fait en utilisant la variable Y ~ U(0,1) et en posant $Y = r^2$, on obtient bien :

$$P(r) = \left|\frac{dY}{dr}\right| = 2r \tag{7.12}$$

On voit donc qu'on peut échantillonner uniformément des points dans le cercle en choisissant les variables suivantes :

$$\theta \sim U(0,2\pi) \tag{7.13}$$

$$Y \sim U(0,1) \tag{7.14}$$

$$r = \sqrt{Y} \tag{7.15}$$

Cela nous donne donc des couples (x,y) avec :

$$x = \sqrt{Y} cos\theta \tag{7.16}$$

$$y = \sqrt{Y} sin\theta \tag{7.17}$$

Contrairement à la deuxième méthode, la troisième fonctionne bien. Cela est du au fait que cette fois ci, on a inclus dans nos calculs le jacobien de la transformation en coordonnées polaires.

7.4.2 La gaussienne

Il nous serait très utile de savoir échantillonner des variables de distribution Gaussienne. En effet, toutes les distributions "smooth" peuvent être écrite comme une superposition de Gaussienne. Savoir échantillonner selon une Gaussienne permet donc d'échantillonner selon toutes les distributions voulues.

Pour ce faire, on utilise la méthode de Box-Muller. En partant de la fonction de densité de probabilité jointe, on a :

$$P_X(x,y)dxdy = \frac{e^{-x^2/2}}{\sqrt{2\pi}} \frac{e^{-y^2/2}}{\sqrt{2\pi}} = \frac{d\theta}{2\pi} \cdot re^{-r^2/2}dr \tag{7.18}$$

où l'on a procédé au changement en coordonnées polaires.

En posant $Y = \frac{r^2}{2}$, on obtient :

$$\left(\frac{1}{2\pi}d\theta\right)\left(e^{-Y}dY\right) = U_\theta[0,2\pi]d\theta \cdot \exp(Y)dY \tag{7.19}$$

On peut alors en partant de la loi uniforme $U[0,1]$, que l'on sait échantillonner numériquement, échantillonner des variables aléatoires qui suivent une distribution Gaussienne. On procède de la manière suivante :

1. D'abord on échantillone θ et Y à l'aide de la loi uniforme et des relations suivantes :

$$\theta = 2\pi \cdot U[0,1] \tag{7.20}$$

$$Y = -\log(U[0,1]) \tag{7.21}$$

2. On peut ensuite tirer x et y à l'aide du changement de coordonnées polaire :

$$x = \sqrt{2\pi}\cos\theta \tag{7.22}$$

$$y = \sqrt{2\pi}\sin\theta \tag{7.23}$$

De cette manière, on tire directement deux variables indépendantes x et y qui suivent les deux une distribution Gaussienne.

7.5 Chaîne de Markov Monte-Carlo (MCMC)

Le MCMC est un algorithme créant une séquence de variables aléatoires en espérant qu'elle converge vers une variable aléatoire avec la probabilité désirée. Un exemple de MCMC est l'échantillonage de l'intérieur d'un carré

de côté qui vaut 2 centré en l'origine. On commence à un certain point à l'intérieur du carré puis à chaque pas de temps on avance vers un autre point, l'algorithme s'exprime de la façon suivante :

$$x^{t+1} = x^t + \epsilon U[-1,1] \tag{7.24}$$

$$y^{t+1} = y^t + \epsilon U[-1,1] \tag{7.25}$$

si au temps $t+1$ le point est en dehors du carré trois choix s'offrent à nous :

1.Je relance jusqu'à ce que j'obtienne un point dans le carré puis je vais à ce point.

2.Je vais au point et je continue à avancer jusqu'à ce que je rentre à nouveau dans le carré puis je supprime tous les points obtenus en dehors du carré.

3.Je reste à ma position du temps t, je compte un point en plus à cette même position puis je relance.

Le troisième algorithme s'avère être le meilleur, c'est l'algorithme de Metropolis.

7.5.1 Exemple avec une grille à 9 états

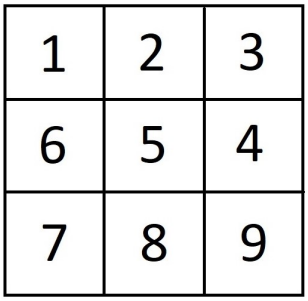

Figure 7.8: Grille pour le MCMC, on peut se déplacer seulement horizontalement ou verticalement avec une certaine probabilité

Une **chaîne de Markov** est un processus stochastique où la probabilité de transition vers l'état suivant dépend uniquement de l'état actuel et non des états précédents. Mathématiquement, cela s'exprime par :

$$P(X_{t+1} = x \mid X_t = x_t, X_{t-1} = x_{t-1}, \ldots, X_0 = x_0) = P(X_{t+1} = x \mid X_t = x_t). \tag{7.26}$$

Dans le cas discret, on peut imaginer une grille, comme représentée à la figure 7.8, contenant 9 états où l'on peut se déplacer d'un état a à un état b seulement horizontalement ou verticalement avec un taux de transition $P_{a\to b}$.

7.5.2 Matrice de Transition

L'évolution de la chaîne de Markov est décrite par une matrice de transition P, où P_{ij}, est la probabilité de passer de l'état j à l'état i :

$$P_{ij} = P(X_{t+1} = i \mid X_t = j). \tag{7.27}$$

Une chaine de Markov est entièrement déterminée par cette matrice de transfert car elle contient la probabilité d'être dans un état i au temps $t+1$ si on est dans un état j au temps t pour tout i, j et t.

7.5.3 Conservation de probabilité

Le but est que pour t grand, l'algorithme visite tous les états avec une probabilité donnée $\vec{\pi}^{eq}$. Cela impose certaines conditions. La première est la conservation de probabilité :

$$1 = P_{j\to j} + \sum_{i\,\mathrm{d}\,j} P_{j\to i} \tag{7.28}$$

en effet, les seules possibilités sont soit de rester dans l'état dans lequel on était déjà, soit on bouge vers un autre état.

7.5.4 Équation maîtresse

La deuxième condition est l'équation maitresse, elle exprime la probabilité d'être dans l'état j au temps $t+1$

$$\pi_j^{t+1} = \pi_j^t P_{j\to j} + \sum_{i\neq j} \pi_i^t P_{i\to j} \tag{7.29}$$

le premier terme est simplement la probabilité d'être dans l'état j au temps t multiplié par la probabilité d'y rester et le deuxième terme est la somme des probabilités d'être dans un état i au temps t multiplié par la probabilité d'aller de i à j.

7.5.5 Distribution d'Équilibre, bilan global et détaillé

La chaîne de Markov possède une distribution d'équilibre π^* si elle satisfait :

$$\pi_j^* = \sum_i \pi_i^* P_{ij} \quad \text{pour tout } j. \tag{7.30}$$

On peut multiplier par 1 des deux côtés et utiliser le fait que $\sum_i P_{ji} = 1$:

$$1 \times \pi_j^* = \sum_i \pi_i^* P_{ij} \tag{7.31}$$

$$\sum_i \pi_j^* P_{ji} = \sum_i \pi_i^* P_{ij} \tag{7.32}$$

où la dernière égalité est le bilan global qui est la dernière condition, à l'équilibre, on voudrait échantillonner cette loi. Il faut que les taux de transition satisfassent cette loi, sans ça, il n'est pas possible de converger vers les probabilités recherchées. Etant donné que les vecteurs $\vec{\pi}_i$ ont souvent une grande dimension, il est nécessaire de faire une simplification, pour se faire on peut regarder une sous-classe d'algorithmes qui satisfait le bilan détaillé :

$$\pi_j^* P_{ji} = \pi_i^* P_{ij} \tag{7.33}$$

Attention: Le bilan détaillé implique le bilan global mais le contraire est faux. Il existe de nombreux travaux qui proposent des algorithmes sans bilan détaillé! Mais nous allons contenter ici de l'approche la plus classique.

7.6 Algorithme de Metropolis-Hastings

L'**algorithme de Metropolis-Hastings** est une méthode MCMC pour générer des échantillons d'une distribution de probabilité cible $\pi(x)$ connue jusqu'à un facteur de normalisation. L'algorithme de Metropolis fonctionne comme cela: (i) on propose un changement possible, au hasard, et (ii) on décide d'accepter ce changement avec une certaine probabilité. Il faut donc bien décomposer la probabilité, on a:

$$P_{a\to b} = P\text{proposer ce mouvement} \times P\text{accepter ce mouvement} \tag{7.34}$$

Typiquement, nous allons utiliser quelque chose de simple pour Pproposer ce mouvement, typiquement on peut choisir dans notre problème de bouger en haut, en bas, a droite ou a gauche, de façon équiprobable.

Il faut ensuite décider si l'on accepte ou non ce mouvement! C'est la que Metropolis nous donne une règle qui satisfait automatiquement le Bilan détaille:

7.6.1 Description de l'Algorithme

1. Initialiser avec un état initial x_0.
2. Pour chaque étape t :
   (a) Générer une proposition x' à partir de l'état actuel x_t en utilisant une distribution de proposition $q(x' \mid x_t)$.
   (b) Calculer le taux d'acceptation :
   $$\alpha = \min\left(1, \frac{\pi(x')q(x_t \mid x')}{\pi(x_t)q(x' \mid x_t)}\right). \tag{7.35}$$
   (c) Accepter le mouvement avec probabilité α.

Il est assez simple de vérifier que cette regle verifie le Bilan detaillée. En particulier, si les mouvement "q sont equiprobabble parmis un certains nombre de choix, on trouve que règle de Metropolis s'écrit comme :

$$P_{a\to b} = \min\left[1, \frac{\pi_b^{eq}}{\pi_a^{eq}}\right] \tag{7.36}$$

Grâce à cette règle, les taux de transitions sont seulement donnés que par des rapports de probabilités! Pourquoi est-ce important ? Imaginons un système décrit par $X^{(1)}$ et l'on cherche à se rendre à l'état $X^{(2)}$ par un mouvement de Monte-Carlo :

$$X^{(1)} = \begin{vmatrix} \vec{x}_1^{(1)} \\ \vdots \\ \vec{x}_N^{(1)} \end{vmatrix} \to X^{(2)} = \begin{vmatrix} \vec{x}_1^{(2)} \\ \vdots \\ \vec{x}_N^{(2)} \end{vmatrix}$$

Dans le formalisme canonique, les probabilités à l'équilibre sont données par :

$$\pi_1^{eq} = \frac{e^{-\beta \mathrm{H}(X^{(1)})}}{Z} \quad , \quad \pi_2^{eq} = \frac{e^{-\beta \mathrm{H}(X^{(2)})}}{Z}$$

On a alors pour la règle de Metropolis dans le cas canonique la relation suivante :

$$P_{1\to 2} = \min\left[1, e^{-\beta(E_2 - E_1)}\right] \tag{7.37}$$

La règle de Metropolis permet donc de calculer les taux de transitions sans calculer ou même connaître la fonction Z. Cette règle permet donc de simplifier à la fois les calculs et le bilan détaillé. En effet, en respectant la règle de Metropolis, on est assuré de respecter le bilan détaillé.

7.6.2 Metropolis pour notre probleme

Appliquons metropolis a notre probleme simple! On va simplement chosir les moiuvement de facon equiprobable parmis 4 possible : haut, bas, droite, gauche. QUand accepte-t on ? Et bien, toujours, puisque les probabliite π sont toute les memes, on trouvera donc $\alpha = 1$ **SAUF** si l'on propose de sortir de la grille, auquel cas $\alpha = 0$.

7.6.3 La convergence de Metropolis

Il reste alors une question à se poser : la vitesse de convergence de l'algorithme de Metropolis.

Tout d'abord, voyons pourquoi nous sommes sûr que la matrice converge. La matrice de transition T est droite-stochastique (la somme sur les lignes vaut toujours 1). Si la dynamique respecte les conditions d'ergodicité et d'irréductibilité, alors il existe une seule valeur propre $|\lambda|=1$ et toutes les autres ont valeurs $|\lambda_i<1|$. De plus, le vecteur propre associé à $|\lambda|=1$ est $\vec{\pi^{eq}}$. Voyons pourquoi ceci assure la convergence de la matrice :

Soit λ_i et e_i les valeurs propres et vecteurs propres de la matrice T. On peut alors écrire :

$$\vec{\pi}^{t=0} = \sum_i \alpha_i e_i$$

On décrit l'évolution de $\vec{\pi}$ comme :

$$\vec{\pi}^{t=t} = T\vec{\pi}^{t=t-1} = T^t\vec{\pi}^{t=0} = \sum_i \alpha_i \lambda_i^t e_i$$

Mais comme toutes les valeurs propres autres que $|\lambda_1|=1$ sont plus petites que 1, leurs valeurs tend vers 0 lorsque t devient grand. On a donc la relation :

$$\vec{\pi}^{t=t} = \vec{\pi}^{eq} \approx \vec{\pi}^{eq} + \alpha_2\lambda_2^t\vec{\pi}^2 \tag{7.38}$$

On voit alors que le temps de convergence dépend uniquement de la vitesse de convergence vers 0 de λ_2.

On peut réecrire λ_2^t comme :

$$\lambda_2^t = e^{t\log\lambda_2} = e^{-t\log\frac{1}{\lambda_2}} = e^{-t/\frac{1}{\log\frac{1}{\lambda_2}}}$$

On définit alors $\tau = \frac{1}{\log\frac{1}{\lambda_2}}$ et c'est ce paramètre qui définit la vitesse de convergence. Plus λ_2 sera proche de 1, plus τ sera grand et donc la convergence longue. Dans la pratique, il est très difficile de connaître λ_2 mais la convergence est toujours garantie.

7.6.4 Les conditions nécessaires à la convergence

Voyons plus en détails les conditions nécessaires à la convergence de la chaîne de Markov. Elles sont au nombre de trois:

(i) Tout d'abord, il faut respecter le bilan global. C'est bien cas pour nous

(ii) Il faut être ergodique, c'est à dire que n'importe quel point est atteignable depuis n'importe quel point. Imaginons une grille pour le MCMC de taille 2x2 numérotée de 1 à 4. Un exemple de matrice non-ergodique est pour ce système est :

$$T = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

Dans ce cas là, la matrice T décrit deux sous-systèmes indépendants qui ne communique pas. On reste soit dans les points 1 et 2, soit dans les points 3 et 4. Mathématiquement, on peut diagonaliser par bloc cette matrice avec chaque bloc ayant une valeur propre $\lambda = 1$.

(iii) La dernière condition est d'être apériodique: il ne faut pas que l'algorithme soit pige dans une boucle deterministe qui se repete infiniment! En pratique, cela n'arrive pratiquement jamais puisque e nous utilisons des algo avec de l'aleatoire. Voyons quand même un cas de matrice périodique. Imaginons une grille pour le MCMC de seulement 2 cases numérotées 1 et 2. On considère la matrice de transition suivante :

$$T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

On a ici un exemple de système périodique, connaissant le point de départ on sait exactement où l'on se trouve au temps t. On voit mathématiquement que le théorème n'est pas valide car les 2 valeurs propres de cette matrice sont $\lambda_{1,2} = \pm 1$.

7.7 Algorithme de Metropolis-Hastings en Physique Statistique

L'un des contextes historiques et les plus importants d'application de l'algorithme de Metropolis-Hastings est la **physique statistique**. Dans ce cadre, le but est d'échantillonner les configurations microscopiques d'un système de N particules selon la *distribution canonique de Boltzmann* :

$$\pi^{eq}(X) = \frac{1}{Z} e^{-\beta \mathscr{H}(X)}, \tag{7.39}$$

où $\mathscr{H}(X)$ est l'énergie du système, $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ représente les positions des particules, $\beta = 1/(k_B T)$ est l'inverse de la température, et Z la fonction de partition :

$$Z = \int dX \, e^{-\beta \mathscr{H}(X)}.$$

Le calcul de Z étant impossible pour un grand nombre de particules, l'algorithme de Metropolis permet de générer directement des configurations distribuées selon $\pi^{eq}(X)$ sans jamais avoir besoin de connaître Z.

7.7.1 Principe général

On part d'une configuration initiale $X^{(0)} = \{\boldsymbol{x}_i^{(0)}\}$, puis à chaque étape, on propose une petite modification du système. Par exemple, on choisit une particule au hasard i et on propose un déplacement :

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \epsilon \, \boldsymbol{u},$$

où $\boldsymbol{u}$ est un vecteur aléatoire choisi uniformément dans une petite sphère (ou cube) centrée en zéro, et ϵ fixe l'amplitude typique du déplacement.

On calcule alors la variation d'énergie

$$\delta E = \mathscr{H}(X') - \mathscr{H}(X^{(t)}), \tag{7.40}$$

et on accepte ce nouveau point avec la probabilité de Metropolis :

$$P_{\text{accept}}(X^{(t)} \to X') = \min\left(1, e^{-\beta \delta E}\right). \tag{7.41}$$

Si le mouvement est accepté, on remplace $X^{(t+1)} = X'$; sinon, on garde $X^{(t+1)} = X^{(t)}$.

Cette règle satisfait automatiquement le **bilan détaillé** :

$$\pi^{eq}(X^{(t)}) P(X^{(t)} \to X') = \pi^{eq}(X') P(X' \to X^{(t)}),$$

ce qui garantit la convergence vers la distribution de Boltzmann.

7.7.2 Le choix de la taille des déplacements : la “règle du 1/2”

Un aspect souvent négligé mais crucial du bon fonctionnement du Metropolis est le **choix de l'amplitude** ϵ des déplacements proposés. Ce paramètre contrôle le compromis entre exploration et acceptation :

- Si ϵ est trop petit, alors les changements proposés sont minuscules, l'énergie varie peu, et la probabilité d'acceptation est proche de 1. Le système évolue lentement, explore mal l'espace des configurations et reste fortement corrélé.
- Si au contraire ϵ est trop grand, la plupart des propositions augmentent fortement l'énergie, les mouvements sont presque toujours refusés, et le système ne bouge presque plus.

Le bon choix est donc un compromis : il faut que les déplacements soient suffisamment grands pour explorer l'espace, mais pas trop pour que la plupart soient encore acceptés. Empiriquement, on cherche à ajuster ϵ de sorte que la **probabilité moyenne d'acceptation** soit environ :

$$\boxed{\langle P_{\text{accept}} \rangle \approx \frac{1}{2}.}$$

Cette **“règle du 1/2”** n'est pas stricte mais donne une indication très utile :

- Si le taux d'acceptation est trop élevé (~ 0.9), on peut augmenter ϵ pour accélérer l'exploration.
- S'il est trop bas (~ 0.1), on doit réduire ϵ pour éviter de rester bloqué.

Dans les systèmes de grande dimension, des analyses plus précises montrent que le taux optimal d'acceptation tend vers une valeur proche de 0.234 pour les mouvements gaussiens isotropes (résultat de Roberts et al., 1997), mais la règle empirique du 1/2 reste un excellent guide pour la physique statistique.

7.7.3 Moyennes thermodynamiques et thermalisation

Une fois la chaîne de Markov convergée vers l'équilibre, on peut estimer des observables $\mathcal{O}(X)$ (par exemple l'énergie moyenne, la position moyenne d'une particule, etc.) par une moyenne temporelle :

$$\langle \mathcal{O} \rangle = \frac{1}{M} \sum_{t=t_0}^{t_0+M} \mathcal{O}(X^{(t)}), \tag{7.42}$$

où t_0 est le temps de *thermalisation* (nombre d'itérations nécessaires pour oublier l'état initial) et M le nombre d'échantillons effectivement utilisés.

7.7.4 Remarques pratiques

- La bonne estimation des observables nécessite de vérifier que la chaîne est bien **mélangée** (mixing time suffisant).
- Les configurations successives sont souvent corrélées : il faut parfois ne conserver qu'une configuration sur k pour obtenir des échantillons quasi-indépendants.
- Le choix du pas ϵ est essentiel : il influence directement la vitesse de convergence.

7.7.5 Résumé

L'algorithme de Metropolis appliqué à un système de particules dans un potentiel :

- permet d'échantillonner la distribution de Boltzmann sans connaître Z ;
- satisfait automatiquement le bilan détaillé ;
- converge si la chaîne est ergodique et apériodique ;
- est efficace seulement si le taux d'acceptation est bien calibré — typiquement autour de 50%.