L'Algorithme Metropolis-Hastings

Projet de recherche CRSNG

Vanessa Bergeron Laperrière

(supervisée par Mylène Bédard)

Été 2010

Département de Mathématiques et Statistique
Université de Montréal

1 Introduction

Ce rapport conclut un stage de recherche de premier cycle du CRSNG, effectué sous la supervision de la professeure Mylène Bédard. Le but du projet était de comprendre les mécanismes de base des méthodes de Monte Carlo par chaînes de Markov, utilisées dans le cadre de la simulation de variables aléatoires. Plus spécialement, nous voulions parvenir à programmer divers algorithmes de Metropolis-Hastings afin d'en remarquer les forces et les faiblesses, pour peut-être y voir des améliorations possibles. Avant de se lancer dans de tels objectifs, il fallait d'abord connaître un peu de théorie sur la simulation ainsi que sur les chaînes de Markov, outils nécessaires à la compréhension et à l'étude de l'algorithme Metropolis-Hastings. Ce rapport fait d'abord un survol théorique des notions vues durant le stage. Il développe ensuite les résultats obtenus dans le but d'optimiser l'algorithme Metropolis-Hastings, et propose finalement des améliorations à celui-ci.

2 Méthodes de base de simulation

L'objectif de base des méthodes de simulation est de produire des nombres pseudo-aléatoires, ou des variables aléatoires distribuées selon une certaine fonction de densité d'intérêt f. Beaucoup de raisons peuvent pousser à l'utilisation dans la pratique de telles méthodes, par exemple les jeux sur support électronique, ou encore le cryptage de messages, etc. D'un point de vue plus théorique (et c'est à celui-ci que nous nous intéresserons), la simulation servira dans le cadre du problème général de l'évalutation de

$$\mathbb{E}_f[h(X)] = \int_{supp\ f} h(x)f(x)dx\ ,$$

pour n'importe quelle variable aléatoire X suivant une fonction de densité f et pour toute fonction réelle h, dans les cas où l'intégrale ne peut être calculée de manière analytique, notamment en grandes dimensions.

2.1 Intégration de Monte Carlo

La méthode classique d'intégration de Monte Carlo fut développée en 1949 par Stan Ulam et Nicholas Metropolis, avec la collaboration de John von Neumann, suite au développement du premier ordinateur électronique, ENIAC, qui permettait pour la première fois de l'histoire de se débarrasser des calculs longs et répétitifs (Metropolis 1987). Elle s'énonce comme suit. Supposons que nous ayons un échantillon $(X_1, X_2, \ldots, X_N)$ généré selon la densité f (nous nous intéresserons plus loin à la production de cet échantillon). Il est alors possible d'approximer $\mathbb{E}_f[h(X)]$ par l'estimateur de Monte Carlo

$$\hbar_N = \frac{1}{N}\sum_{j=1}^{N} h(x_j),$$

étant donné que $\hbar_N$ converge presque sûrement vers $\mathbb{E}_f[h(X)]$ par la Loi forte des grands nombres.

Bien qu'il soit parfois difficile de produire un échantillon selon f, cette difficulté peut heureusement être contournée par la méthode dite d'échantillonnage préférentiel (en anglais, *importance sampling*) permettant de générer l'échantillon $(X_1, X_2, \ldots, X_N)$

non pas de la distribution d'intérêt f, mais plutôt d'une autre densité g telle que *supp* $f \subseteq$ *supp* g. On utilisera alors le fait que

$$\mathbb{E}_f[h(X)] = \int_{supp\ f} h(x)\frac{f(x)}{g(x)}g(x)dx$$

et $\mathbb{E}_f[h(X)]$ sera estimée par

$$\tilde{h}_N = \frac{1}{N}\sum_{j=1}^{N}\frac{f(x_j)}{g(x_j)}h(x_j).$$

Ainsi, à condition de pouvoir générer l'échantillon selon la distribution voulue, le problème du calcul de $\mathbb{E}_f[h(X)]$ est résolu.

2.2 Uniformes

Nous pouvons maintenant songer à la production d'un échantillon $(X_1, \ldots, X_n)$ suivant une densité f. Nous aurons avant tout besoin de pouvoir simuler certaines variables aléatoires suivant des distributions simples. Logiquement, la toute première forme de variable aléatoire qu'il serait utile de simuler est sans aucun doute les uniformes sur un intervalle [a, b], et plus spécifiquement les $\mathcal{U}_{[0,1]}$ parce qu'elles fournissent une représentation des probabilités et parce que nous devront faire appel aux uniformes pour simuler des variables provenant d'autres distributions. On appelle *générateur de nombres uniformes pseudo-aléatoire* tout algorithme qui, à partir d'une valeur initiale u_0 et suivant une transformation D, produit une suite $(u_i) = D^i(u_0)$ de valeurs dans $[0, 1]$ satisfaisant la propriété qui suit. Pour n'importe quel n, les valeurs $(u_1, \ldots, u_n)$ se comportent, si soumises à certains tests, comme un échantillon $(U_1, \ldots, U_n)$ de $\mathcal{U}_{[0,1]}$ iid (indépendentes et identiquement distribuées). Un algorithme de base utiliserait par exemple une transformation de la forme $D(x) = \frac{ax+b}{M+1} \bmod 1$, avec $a, b, M \in \mathbb{N}$. Certains algorithmes sont souvent préprogrammés dans les logiciels et compilateurs, leur permettant ainsi d'être utilisés par une simple commande plutôt que d'avoir à en construire le code.

2.3 Variables aléatoires discrètes

Grâce à l'utilisation d'uniformes, la simulation de variables aléatoires discrètes devient assez facile. En effet, il suffit de calculer les valeurs que prend la fonction de répartition $F(k) = P(X \leq k) = p_k$, pour toutes les valeurs k que peut prendre X. Ensuite, une valeur $u \sim \mathcal{U}_{[0,1]}$ est générée. Pour respecter la fonction de répartition F, il suffit donc que X_i prenne la valeur du k tel que $p_{k-1} < u \leq p_k$. En recommençant ces étapes pour i entre 0 et N, on obtient un échantillon $(X_1, \ldots, X_N)$ de taille N distribué selon la fonction de masse choisie.

2.4 Méthodes de transformations générales

Le problème se complique rapidement lorsque l'on cherche à simuler des variables aléatoires continues. Heureusement, plusieurs se génèrent en utilisant quelques transformations directes à partir d'uniformes. Définissons d'abord, pour une fonction non-décroissante F sur $\mathbb{R}$ (une fonction de répartition par exemple), son *inverse généralisée*, F^-. Elle sera la fonction définie par

$$F^-(u) = \inf \{x : F(x) \geq u\} .$$

On pourra utiliser cette inverse pour faire un premier type de transformation. En effet, si $U \sim \mathcal{U}_{[0,1]}$, alors la variable aléatoire $F^-(U)$ suit la distribution F. En d'autre termes, pour générer une variable $X \sim F$, il suffit de générer U selon une $\mathcal{U}_{[0,1]}$ puis d'effectuer la transformation $x = F^-(u)$. Par exemple, pour générer $X \sim \mathcal{E}xp(1)$, donc avec $F(x) = 1 - e^{-x}$, on pourra utiliser $x = -\log(1-u)$. Si $U \sim \mathcal{U}_{[0,1]}$ ($1-U$ est donc aussi uniforme sur $[0, 1]$), alors la variables $X = -\log U$ aura la distribution exponentielle voulue.

Lorsqu'inverser la fonction de répartition se révèle impossible, nous pourrons utiliser d'autres opérations sur les variables aléatoires pour nous permettre d'en atteindre de nouvelles. Ainsi, en additionnant correctement des exponentielles, ou en utilisant les fonctions trigonométiques, on peut générer, toujours à partir d'uniformes, certains

type de variables Chi carrées, Gamma, Beta, Normales, Poisson, etc. Par exemple, l'algorithme suivant, dit Box-Muller, permet de générer des variables suivant des $\mathcal{N}(0,1)$:

1. Générer U_1, U_2, deux $\mathcal{U}_{[0,1]}$ iid ;
2. Définir
$$\begin{cases} x_1 = \sqrt{-2\log(u_1)}\cos(2\pi u_2) \\ x_2 = \sqrt{-2\log(u_1)}\sin(2\pi u_2) \end{cases} ;$$
3. Prendre x_1 et x_2 comme étant deux tirages indépendants de $\mathcal{N}(0,1)$.

Il existe bien entendu d'autres méthodes de simulation plus complètes et offrant beaucoup plus de possibilités en terme de densités à simuler. Nommons par exemple la méthode d'acceptation-rejet, qui permet de simuler un échantillon provenant de virtuellement n'importe quelle densité f connue à une constante multiplicative près, en utilisant une seconde densité g, dite instrumentale. Permettons-nous de ne pas développer ici cette méthode, mais de seulement mentionner que le choix de cette seconde densité donne vite beaucoup de fil à retordre, surtout dans les cas de grandes dimensions, et de passer tout de suite à la famille de méthodes qui nous intéresse ici, les méthodes MCMC.

3 Méthodes MCMC

3.1 Théorie sur les chaînes de Markov

Avant de discuter de cette nouvelle classe de méthodes, nous aurons besoin d'un nouvel outil : les chaînes de Markov. Par soucis de concision, nous ne ferons qu'un survol théorique des propriétés dont nous aurons besoin aux sections suivantes.

Une chaîne de Markov peut être vue comme une suite de variables aléatoires $\{X_n, n = 0, 1, 2, \dots\}$ évoluant dans le temps, avec des probabilités de transition dépendant uniquement de l'état particulier dans lequel la chaîne se trouve. Si $X_t = i$, on dira que la chaîne est à l'état i au temps t. Définissons S comme étant l'espace d'états de la chaîne de Markov (ici discret), c'est-à-dire l'ensemble de tous les états

qu'elle peut prendre. Nous noterons par $\mathbb{P}_{ij}$ la probabilité de passer de l'état i à l'état j en une seule étape, dite probabilité de transition. Pour que la suite $\{X_n\}$ soit une chaîne de Markov, on doit donc avoir, pour tout n,

$$\begin{aligned}\mathbb{P}_{ij} &= P\left(X_{n+1} = j \,|X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0\right) \\ &= P\left(X_{n+1} = j \,|X_n = i\right)\end{aligned}$$

avec

$$\mathbb{P}_{ij} \geq 0 \quad \forall i, j \in S \quad \text{et} \quad \sum_{j \in S} \mathbb{P}_{ij} = 1 \quad \forall i \in S$$

En d'autres termes, seule la connaissance de l'état actuel de la chaîne (au moment présent) nous permet de connaître les probabilités de transition vers chacun des autres états. La connaissance du chemin parcouru précédemment ne nous donne pas d'information supplémentaire sur le futur. On peut aussi s'intéresser aux probabilités de transition en plusieurs étapes. La probabilité de passer de l'état i à l'état j en n étapes sera notée

$$\mathbb{P}_{ij}^n = P\left(X_{k+n} = j \,|X_k = i\right) \quad \forall i, j \in S \text{ avec } n, k \geq 0.$$

Ces probabilités sont liées entre elles par l'équation de Chapman-Kolmogorov :

$$\mathbb{P}_{ij}^{n+m} = \sum_{k \in S} \mathbb{P}_{ik}^n \mathbb{P}_{kj}^m \qquad \forall n, m \geq 0 \text{ et } \forall i, j \in S$$

où les états k agissent comme états transitaires.

Certaines des propriétés des chaînes de Markov nous assureront de l'efficacité des méthodes élaborées dans les prochaines sections. C'est le cas notamment de l'**irréductibilité**. Cette notion exige d'abord de poser la définition suivante : L'état j est dit *accessible* à partir de i s'il existe un $n \geq 0$ tel que $\mathbb{P}_{ij}^n > 0$, c'est-à-dire que, la chaîne étant d'abord à i, il sera possible en un certain nombre n d'étapes qu'elle rejoigne l'état j. Si deux états sont accessibles l'un à l'autre (possiblement en un nombre d'étapes différent), on dit qu'ils communiquent et tous les états communiquant entre eux forment une *classe*. Deux classes de la même chaîne de Markov sont donc soit identiques, soit disjointes. Une chaîne composée d'une seule classe est dite *irréductible*.

La prochaine propriété que nous rechercherons dans une chaîne de Markov sera l'**apériodicité**. Si d est le plus grand commun diviseur de l'ensemble des $m \geq 1$ tels que $\mathbb{P}_{ii}^m > 0$, on dira alors que i a une *période* de d. C'est-à-dire que i ne sera revisité qu'après un minimum de d étapes supplémentaires. La période se rapproche de la notion de cycle et est une propriété de classe. Ceci signifie qu'une chaîne de Markov irréductible n'en aura qu'une seule, et dans le cas où $d = 1$, la chaîne sera dite apériodique.

Ce qui représente le plus d'intérêt dans une chaîne de Markov est sans aucun doute sa **distribution stationnaire**. Si l'on pose

$$\pi_j = \lim_{n\to\infty} \mathbb{P}_{ij}^n \ , \qquad j \in S$$

(indépendante de i), alors π_j représente la proportion à long terme du temps que le processus passera dans l'état j (la *probabilité limitante*) et est la solution non-négative de

$$\pi_j = \sum_{i\in S} \pi_i \mathbb{P}_{ij} \quad \text{et} \quad \sum_{j\in S} \pi_j = 1 \ .$$

Pour des chaînes de Markov irréductibles et apériodiques, cette proportion π_j est aussi appelée *probabilité stationnaire*, car

$$\lim_{n\to\infty} P(X_n = j) = \pi_j, \quad j \in S,$$

pour presque n'importe quelle valeur de départ. L'ensemble de ces probabilités forme la distribution stationnaire.

La dernière propriété qui nous intéressera ici sera la **réversibilité** de la chaîne. Une chaîne de Markov sera réversible par rapport à une distribution $\{\pi_i\}_{i\in S}$ si

$$\pi_i \mathbb{P}_{ij} = \pi_j \mathbb{P}_{ji}$$

pour tout i, j dans S. Il s'agit en quelque sorte d'une symétrie entre la chaîne et son processus inverse (avec le temps à reculons). Ceci est une condition suffisante (mais non nécessaire), parfois facile à vérifier, pour que π soit une distribution stationnaire de la chaîne.

Toutes les propriétés précédentes ont été énoncées dans le cadre des chaînes de Markov à espace discret, mais les concepts peuvent se généraliser à un espace continu. Cependant, leur présente forme est suffisante à la compréhension (moins rigoureuse) de ce qui suit.

3.2 Méthodes MCMC

Nous avons désormais accès à un nouveau type de méthodes de simulation, les méthodes de Monte Carlo par chaînes de Markov, ou méthodes MCMC. Ce nom désigne toute méthode qui, dans le but de simuler des variables d'une distribution f, produit une chaîne de Markov irréductible et apériodique dont la distribution stationnaire est f. Une fois cette chaîne produite, nous obtiendrons ainsi un échantillon de variables distribuées selon f et nous pourrons ainsi utiliser l'estimateur de Monte Carlo pour approximer $\mathbb{E}_f[h(x)]$. La chaîne produite devra donc posséder les propriétés énoncées ci-haut.

3.3 Algorithme Metropolis-Hastings

De toutes les familles de méthodes MCMC, la plus générale est sans doute l'algorithme Metropolis-Hastings, dans le sens qu'il impose le moins de conditions sur la densité cible. Cet algorithme fut d'abord publié sous une première forme par Metropolis et al. (1953), puis généralisé par Hastings (1970). À partir de la densité cible $\pi(x)$ (possiblement en grandes dimensions), on choisit une densité instrumentale conditionnelle $q(x, y) = q(y|x)$ à partir de laquelle il est assez facile de simuler. Commençant avec une valeur (possiblement vectorielle) x_0, l'algorithme passe au travers des étapes suivantes à chaque itération. Sachant que la chaîne est à l'état x_t à la t^e itération,

1. Générer $y_{t+1} \sim q(x_t, \cdot)$
2. Calculer la *probabilité d'acceptation*

$$\alpha(x_t, y_{t+1}) = \min\left[\frac{\pi(y_{t+1})q(y_{t+1}, x_t)}{\pi(x_t)q(x_t, y_{t+1})}, 1\right]$$

3. Prendre $x_{t+1} = \begin{cases} y_{t+1} & \text{avec probabilité } \alpha \\ x_t & \text{avec probabilité } 1-\alpha \end{cases}$

En recommençant ces étapes pour t allant de 0 à N, on construit un échantillon à partir duquel il sera possible d'estimer $\mathbb{E}_f[h(x)]$ avec $\hbar_N$. Toutefois, pour se convaincre de la validité de cette affirmation, vérifions d'abord que la chaîne générée par l'algorithme possède les propriétés nous assurant de l'existence de sa distribution stationnaire et de la convergence de celle-ci vers la densité cible $\pi(x)$.

On voit rapidement que la chaîne est apériodique puisque l'évènement $X_{t+1} = X_t$ est possible pratiquement à tout moment. En effet, chaque état peut donc être visité à deux itérations consécutives, d'où $\mathbb{P}^1_{xx} > 0$, leur période étant ainsi de 1. De plus, la chaîne est irréductible car $q(x, y) > 0 \ \forall \ x, y$ dans $\mathcal{E} \times \mathcal{E}$ (où $\mathcal{E}$ est le support de $\pi(x)$), si le support de q inclut $\mathcal{E}$, ce qui devrait être le cas puisque sinon q a été mal choisie [1]. Les états communiquent donc tous entre eux et la chaîne est ainsi composée d'une seule classe, apériodique. Finalement, la probabilité d'acceptation a été construite pour respecter la condition de réversibilité de la chaîne. En effet, la chaîne est réversible et π est sa distribution stationnaire si

$$\begin{aligned} & \pi(x)\mathbb{P}_{xy}dxdy = \pi(y)\mathbb{P}_{yx}dydx \\ \Leftrightarrow\ & \pi(x)q(x,y)\alpha(x,y)dxdy = \pi(y)q(y,x)\alpha(y,x)dxdy \\ \Leftrightarrow\ & \frac{\alpha(x,y)}{\alpha(y,x)} = \frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}. \end{aligned}$$

En choisissant $\alpha(x, y) = \min\left[\frac{\pi(y)q(y,x)}{\pi(x)q(x,y)}, 1\right]$, la relation précédente est automatiquement respectée. La chaîne possède donc les bonnes propriétés pour converger vers la distribution cible.

1. Il se peut que le support de q n'inclut pas d'emblée $\mathcal{E}$, mais que q ait la possibilité de se déplacer partout suivant la valeur actuelle de la chaîne. C'est le cas par exemple de certaines densités instrumentales centrées sur x_t. L'irréductibilité demeure tout de même respectée.

Il existe plusieurs sous-catégories d'algorithmes Metropolis-Hastings, suivant le type de distribution q choisie. D'abord, elle pourrait être indépendante de x_t, ie. $q(x, y) = q(y)$. La difficulté de cette version est de trouver une courbe qui épousera bien la densité cible et qui permettra une bonne exploration de l'espace. Lorsque nous cherchons une densité instrumentale aussi globale, nous sommes souvent bloqués par la complexité et par notre connaissance de la distribution cible, surtout dans les cas de grandes dimensions. C'est pourquoi il est souvent préférable (entendre plus facile) de se tourner vers une approche plus locale de l'exploration du support. Pour ce faire, on crée l'algorithme de sorte à générer une chaîne de Markov du type *marche aléatoire*, c'est-à-dire que la densité intrumentale q sera centrée sur la valeur courante de la chaîne et sera symétrique. La valeur proposée y_{t+1} suivra donc la forme $x_t + \varepsilon_t$, où ε_t est une perturbation aléatoire distribuée selon q et indépendante de x_t. Autrement dit, q sera maintenant de la forme $q(|y - x_t|)$. Par exemple, nous pourrions utiliser $q \sim \mathcal{N}(\vec{x}_t, \Sigma = \mathbf{I}_d \cdot \sigma^2)$ (où $\mathbf{I}_d$ représente la matrice identité en d dimensions) pour simuler $f \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$. [2]

Au delà de l'utilisation en pratique de l'algorithme Metropolis-Hatings, on peut s'intéresser d'un œil théorique à optimiser son efficacité. Plusieurs critères s'offrent à nous pour mesurer le rendement des algorithmes. Nous avons choisi dans notre présente étude de considérer la variance de l'estimateur $\hbar_N$, ainsi qu'une valeur mesurant l'exploration du support par la chaîne créée par l'algorithme, la moyenne des distances au carré des sauts, l'ASJD (de l'anglais *Average Square Jumping Distance*). Celle-ci est définie par

$$ASJD = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{d} (X_{i,j} - X_{i-1,j})^2$$

où d est le nombre de dimensions de la distribution cible et N est le nombre d'itérations faites. On cherchera à la fois à minimiser la variance de l'estimateur en même tant qu'à maximiser l'ASJD (on veut que la chaîne ait exploré au maximum l'espace), ce

2. Cet exemple n'a bien sûr aucune valeur en pratique, puisque la modélisation directe de f serait plus facile que celle de q, mais il constitue un bon point de départ pour tester la théorie et amadouer l'algorithme.

qui s'avérera équivalent. Un résultat théorique connu (Roberts et al. 1997) stipule que, pour des distributions cibles en grandes dimensions avec composantes iid et pour des densités intrumentales symétriques, le meilleur taux d'acceptation des valeurs proposées dans l'algorithme (nombre de valeurs acceptées / nombres d'itérations) est d'environ 25% (0.234 serait la valeur théorique précise, mais le fait de viser 25% suffit dans la pratique) [3]. La recherche de ce taux nous dirigera dans le choix des paramètres de la fonction instrumentale de notre algorithme. Par exemple, si l'on cherche à simuler $f \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_{100})$ en utilisant q de la forme $\mathcal{N}(\mathbf{x_t}, \mathbf{I}_{100} \cdot \sigma^2)$, le choix de σ^2 maximisant l'efficacité de notre algorithme serait 0,0566. Plus généralement, en dimension d, le σ^2 optimal sera de la forme

$$\sigma^2 = \frac{5,66}{d}.$$

Pour le vérifier, il suffit de faire fonctionner l'algorithme (avec un nombre d'itérations fixe) plusieurs fois en boucle en faisant varier la valeur de σ à chaque fois que l'algorithme recommence. On pourra ainsi trouver la valeur (ou au moins un intervalle dans lequel elle devrait se retrouver) de σ correspondant au meilleur ASJD.

Nous avons vu, pour une certaine densité cible et après avoir choisi une famille de densité instrumentale, qu'il est possible d'optimiser l'algorithme Metropolis-Hastings dans sa forme originale en choisissant les paramètres de celui-ci. Cependant, des améliorations à l'algorithme lui-même peuvent mener à des résultats encore plus efficaces. C'est ce qui pousse des chercheurs à proposer de nouveaux algorithmes qui le surpasseront peut-être. Le *Metropolis-Hastings avec rejet retardé* en est un exemple.

3. La preuve de ce résultat exige des notions dépassant largement le cadre de ce texte, nous l'accepterons donc tel quel. Intuitivement, on comprendra qu'un taux d'acceptation trop petit mène à un échantillon contenant trop peu de valeurs différentes pour que l'estimation soit efficace, alors qu'un taux trop élevé suggère que les propositions n'étaient pas assez audacieuses et que la chaîne n'explore probablement pas les endroits à probabilités plus faible, ou encore qu'elle reste coincée dans un mode.

3.4 Algorithme Metropolis-Hastings avec rejet retardé

Développé par Mira (2001), l'algorithme Metropolis-Hastings avec rejet retardé (en anglais, *with delayed rejection*) a pour but de réduire la variance de l'estimateur $\hbar_N$. La modification proposée s'attaque au problème de la répétition d'un même état pour la chaîne lorsque la valeur proposée par l'algorithme est rejetée. Intuitivement, le fait de rester au même état trop souvent suggère que l'algorithme échoue à explorer tout l'espace et que la correlation entre les valeurs augmente. De plus, l'information sur la valeur rejetée dans l'algotithme Metropolis-Hastings régulier ne peut être utilisée subséquemment, afin de préserver les propriétes de la chaîne de Markov. L'algorithme avec rejet retardé (ARR) contourne cette difficulté. Le principe choisi ici pour réduire le nombre de valeurs rejetées est le suivant : lorsque la valeur proposée y_1 est refusée, plutôt que d'avancer le temps en gardant la même valeur, proposons-en une seconde, y_2. Si cette valeur est à son tour rejetée, nous pourrions décider soit de garder la valeur de l'itération précedente et d'avancer le temps, soit de proposer une troisième valeur, et ainsi de suite. Les probabilités d'acceptation à chaque étape devront être ajustées (séparément) afin de préserver la propriéte de réversibilité de la chaîne. L'ARR permet donc d'utiliser l'information acquise par les rejets à l'intérieur d'une même itération, offrant donc la possibilié d'un ajustement local de la distribution instrumentale.

Nous élaborerons ici la version de l'algorithme avec seulement deux étapes par itération.[4] Soit $\pi(x)$ la densité cible. Supposons d'abord que la chaîne soit à l'état $X_t = x$ au temps t.

1. Générer $y_1 \sim q_1(x, \cdot)$
2. Calculer la probabilité d'acceptation

$$\alpha_1(x, y_1) = \min\left[\frac{\pi(y_1)q_1(y_1, x)}{\pi(x)q_1(x, y_1)}, 1\right]$$

3. Prendre $X_{t+1} = y_1$ avec probabilité α_1.

   Si y_1 est rejetée,

4. Pour une description plus générale, voir Mira 2001

(a) Générer $y_2 \sim q_2(x, y_1, \cdot)$

(b) Calculer la nouvelle probabilité d'acceptation

$$\alpha_2(x, y_1, y_2) = \min\left[\frac{\pi(y_2)q_1(y_2, y_1)q_2(y_2, y_1, x)[1 - \alpha_1(y_2, y_1)]}{\pi(x)q_1(x, y_1)q_2(x, y_1, y_2)[1 - \alpha_1(x, y_1)]}, 1\right]$$

(c) Prendre $X_{t+1} = \begin{cases} y_2 & \text{avec probabilité } \alpha_2 \\ x & \text{avec probabilité } 1 - \alpha_2 \end{cases}$

Le rejet de y_1 suggère que la distibution instrumentale q_1 était probablement mal adaptée localement, et q_2 devrait être construite en considération de cela. Cette amélioration permettra une meilleure exploration du support, et produira des valeurs d'ASJD plus grandes que le Metropolis-Hastings régulier, avec la densité intrumentale de la première étape de l'ARR correspondant à celle du Metropolis-Hastings standard. Par contre, le temps d'exécution et de programmation pour un même nombre d'itérations sera sensiblement plus long pour l'ARR.

En cherchant à optimiser l'algorithme avec rejet retardé, on trouve, dans le cas des densités intrumentales symétriques et en grandes dimensions, que le meilleur taux d'acceptation à la première proposition est d'environ 25% (ce qui est cohérent avec les résultats obtenus pour l'algorithme régulier) et qu'il en va de même pour la seconde étape. Un exemple d'utilisation de l'algorithme serait, pour simuler une $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, d'utiliser $q_1 \sim \mathcal{N}(\mathbf{x_t}, \mathbf{I}_d \cdot \sigma_1^2)$ et $q_2 \sim \mathcal{N}(\mathbf{x_t}, \mathbf{I}_d \cdot \sigma_2^2)$, où d est la dimension de la densité cible. Intuitivement, il paraît logique de choisir $\sigma_2 < \sigma_1$, puisque le rejet de la première valeur proposée porte à croire que la proposition était probablement trop «agressive» (trop loin de la valeur courante de la chaîne). La seconde proposition aurait ainsi avantage à être plus «conservatrice». Dans le cas exposé, on peut démontrer théoriquement que les valeurs de variances optimales seront de la forme

$$\sigma_1^2 = \frac{5,66}{d} \qquad \text{et} \qquad \sigma_2^2 = \frac{5,66^2}{d^2}$$

(voir Bédard et al. 2010). Bien que cette forme de l'algorithme fonctionne assez bien en pratique (c'est-à-dire, en dimensions finies et pas trop grandes), la forme optimale de σ_2 pose cependant problème en très grandes dimensions, car les valeurs proposées à

partir de q_2 sont tellement proches de la valeur courante de la chaîne que la différence devient négligeable et que l'amélioration apportée par la deuxième étape face à l'algorithme régulier ne vaut peut-être plus dans ce cas le temps de calcul supplémentaire nécessaire.

4 Discussion

Nous avons vu que l'algorithme Metropolis-Hastings avec rejet retardé, bien qu'il puisse être plus efficace en pratique que le Metropolis-Hastings régulier, pose des problèmes d'ordre théorique lorsqu'on cherche à en trouver les paramètres optimaux puisqu'il faut alors considérer ses comportements lorsque le nombre de dimensions tend vers l'infini. Tout porte à croire que de nouvelles versions pourraient être développées dans le but d'améliorer encore plus l'algorithme, tout en évitant ces difficultés théoriques et en essayant d'amoindrir le temps de calcul nécessaire. Bédard et al. (2010) développent une version de l'algorithme qui fixe la direction des propositions pour chaque itération selon la première proposition dans celle-ci. Par exemple, on pourrait choisir que si $Y_1 = x + \sigma_1 Z$ avec Z une $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, ie. $Y_1 \sim \mathcal{N}(x, \mathbf{I}_d \cdot \sigma_1^2)$, alors $Y_2 = x + \sigma_2 Z$, tandis que l'ARR original aurait quant à lui des propositions de la forme $x + \sigma_1 Z_1$ et $x + \sigma_2 Z_2$. La proposition y_2 serait donc dans la même direction que y_1. L'algorithme demanderait ainsi moins de temps d'exécution que l'ARR, puisque les deux propositions reposeraient sur la même valeur Z. Les probabilités d'acceptation devraient bien sûr encore une fois être adaptées pour conserver les propriétés de la chaîne de Markov créée. Selon la même logique, on pourrait plutôt décider que la seconde proposition serait faite dans le sens opposé à la première, avec $Y_1 = x + \sigma_1 Z$ et $Y_2 = x - \sigma_2 Z$. Les mêmes tests pourront être effectués avec ces nouvelles versions d'algorithmes pour en trouver les paramètres optimaux. Bien entendu, d'autres suggestions seront probablement faites au cours des prochaines années dans le but de rendre les méthodes MCMC toujours plus efficaces, plus rapides et plus utilisées.

Références

[1] Bédard, M., Douc, R., Fort, G. et Moulines, E. (2010) *Scaling Analysis of Delayed Rejection MCMC Methods*, rapport technique.

[2] Hastings, W. K. (1970) *Monte Carlo sampling methods using Markov chains and their applications*, **Biometrika**, Vol. 57, pp. 97-109.

[3] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., et Teller, E. (1953) *Equations of state calculations by fast computing machines*, **Journal of Chemical Physics**, Vol. 21, pp. 1087-1092.

[4] Metropolis, N. (1987) *The Beginning of the Monte Carlo Method*, rapport technique no LA-UR-88-9067, Los Alamos Nationale Laboratory.

[5] Mira, A. (2001), *On Metropolis-Hastings Algorithms with Delayed Rejection*, **Metron**, Vol. LIX, n. 3-4, pp. 231-241.

[6] Robert, C. P. et Casella, G. (2004), *Monte Carlo Statistical Methods*, 2^e édition, Springer, New York.

[7] Roberts, G. O., Gelman, A. et Gilks, W. R. (1997), *Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms*, **Ann. Appl. Probab.**, Vol. 7, pp. 110-120.

[8] Ross, S. M. (2003), *Introduction to Probability Models*, 8^e édition, Academic Press, San Diego.