Université de Montréal

Recyclage des candidats dans l'algorithme Metropolis à essais multiples

par

Assia Groiez

Département de mathématiques et de statistique
Faculté des arts et des sciences

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en statistiques

21 mars 2014

Université de Montréal

Faculté des études supérieures

Ce mémoire intitulé

Recyclage des candidats dans l'algorithme Metropolis à essais multiples

présenté par

Assia Groiez

a été évalué par un jury composé des personnes suivantes :

Pierre Duchesne

(président-rapporteur)

Mylène Bédard

(directeur de recherche)

David Haziza

(membre du jury)

Mémoire accepté le

12 Mars 2014

SOMMAIRE

Les méthodes de Monte Carlo par chaînes de Markov (MCCM) sont des méthodes servant à échantillonner à partir de distributions de probabilité. Ces techniques se basent sur le parcours de chaînes de Markov ayant pour lois stationnaires les distributions à échantillonner. Étant donné leur facilité d'application, elles constituent une des approches les plus utilisées dans la communauté statistique, et tout particulièrement en analyse bayésienne. Ce sont des outils très populaires pour l'échantillonnage de lois de probabilité complexes et/ou en grandes dimensions.

Depuis l'apparition de la première méthode MCCM en 1953 (la méthode de Metropolis, voir [10]), l'intérêt pour ces méthodes, ainsi que l'éventail d'algorithmes disponibles ne cessent de s'accroître d'une année à l'autre.

Bien que l'algorithme Metropolis-Hastings (voir [8]) puisse être considéré comme l'un des algorithmes de Monte Carlo par chaînes de Markov les plus généraux, il est aussi l'un des plus simples à comprendre et à expliquer, ce qui en fait un algorithme idéal pour débuter. Il a été sujet de développement par plusieurs chercheurs. L'algorithme Metropolis à essais multiples (MTM), introduit dans la littérature statistique par [9], est considéré comme un développement intéressant dans ce domaine, mais malheureusement son implémentation est très coûteuse (en termes de temps).

Récemment, un nouvel algorithme a été développé par [1]. Il s'agit de l'algorithme Metropolis à essais multiples revisité (MTM revisité), qui définit la méthode MTM standard mentionnée précédemment dans le cadre de l'algorithme Metropolis-Hastings sur un espace étendu.

L'objectif de ce travail est, en premier lieu, de présenter les méthodes MCCM, et par la suite d'étudier et d'analyser les algorithmes Metropolis-Hastings ainsi que le MTM standard afin de permettre aux lecteurs une meilleure compréhension de l'implémentation de ces méthodes. Un deuxième objectif est d'étudier les perspectives ainsi que les inconvénients de l'algorithme MTM revisité afin de voir s'il répond aux attentes de la communauté statistique. Enfin, nous tentons de

combattre le problème de sédentarité de l'algorithme MTM revisité, ce qui donne lieu à un tout nouvel algorithme. Ce nouvel algorithme performe bien lorsque le nombre de candidats générés à chaque itérations est petit, mais sa performance se dégrade à mesure que ce nombre de candidats croît.

Mots clé : algorithme Metropolis-Hastings, marche aléatoire, échantillonneur indépendant, échantillonneur de Gibbs, algorithme à essais multiples, réversibilité, probabilité d'acceptation.

SUMMARY

Markov Chain Monte Carlo (MCMC) algorithms are methods that are used for sampling from probability distributions. These tools are based on the path of a Markov chain whose stationary distribution is the distribution to be sampled. Given their relative ease of application, they are one of the most popular approaches in the statistical community, especially in Bayesian analysis. These methods are very popular for sampling from complex and/or high dimensional probability distributions.

Since the appearance of the first MCMC method in 1953 (the Metropolis algorithm, see [10]), the interest for these methods, as well as the range of algorithms available, continue to increase from one year to another.

Although the Metropolis-Hastings algorithm (see [8]) can be considered as one of the most general Markov chain Monte Carlo algorithms, it is also one of the easiest to understand and explain, making it an ideal algorithm for beginners. As such, it has been studied by several researchers. The multiple-try Metropolis (MTM) algorithm , proposed by [9], is considered as one interesting development in this field, but unfortunately its implementation is quite expensive (in terms of time).

Recently, a new algorithm was developed by [1]. This method is named the revisited multiple-try Metropolis algorithm (MTM revisited), which is obtained by expressing the MTM method as a Metropolis-Hastings algorithm on an extended space.

The objective of this work is to first present MCMC methods, and subsequently study and analyze the Metropolis-Hastings and standard MTM algorithms to allow readers a better perspective on the implementation of these methods. A second objective is to explore the opportunities and disadvantages of the revisited MTM algorithm to see if it meets the expectations of the statistical community. We finally attempt to fight the sedentarity of the revisited MTM algorithm, which leads to a new algorithm. The latter performs efficiently when the number of generated candidates in a given iteration is small, but the performance

of this new algorithm then deteriorates as the number of candidates in a given iteration increases.

Keywords : Metropolis-Hastings algorithm, random walk, independent sampler, Gibbs sampler, multiple-try Metropolis algorithm, reversibility, acceptance probability.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES TABLEAUX

DÉDICACE

Je dédie cet humble travail,

À Mes Très Chers Parents,

Tous les mots du monde ne sauraient exprimer l'immense amour que je vous porte, ni la profonde gratitude que je vous témoigne pour tous les efforts et les sacrifices que vous n'avez jamais cessé de consentir pour mon instruction et mon bien-être. J'espère avoir répondu aux espoirs que vous avez fondés en moi. Je vous rends hommage par ce modeste travail en guise de ma reconnaissance éternelle et de mon infini amour.

À Maryam,

Mon trésor le plus précieux, si tu savais comme tu me rends heureuse, je suis une autre femme depuis ta naissance. Tu ne sais pas encore la force que tu détiens, qui me rend plus forte au quotidien. Ma fille, je serai là pour panser tes blessures, pour te guider vers un avenir plus sûr, pour te consoler, te réconforter, quand le chagrin fera son entrée.

À Mohamed,

Ton aide, ta générosité, ta patience et ton soutien ont été pour moi une source de courage et de confiance. Qu'il me soit permis aujourd'hui de t'assurer mon respect et ma grande reconnaissance. J'implore Dieu qu'il t'apporte bonheur et t'aide à réaliser tous tes voeux.

À ma soeur Mounira,

Je ne peux exprimer à travers ces lignes tous mes sentiments d'amour et de tendresse envers toi. Puisse la fraternité nous unir à jamais. Merci pour ta précieuse aide à la réalisation de ce travail.

À mes frères,

Aucune dédicace ne saurait exprimer tout ce que je ressens pour vous. Je vous remercie pour tout le soutien exemplaire et l'amour exceptionnel que vous me portez depuis mon enfance.

À tous ceux ou celles qui me sont chers et que j'ai omis involontairement de citer.

REMERCIEMENTS

On dit que le trajet est aussi important que la destination. Mes années d'études en maîtrise m'ont permis de bien comprendre la signification de cette expression. Ce parcours, en effet, ne s'est pas réalisé sans défi et sans soulever de nombreuses questions pour lesquelles les réponses nécessitent de longues heures de travail.

En premier lieu, j'exprime toute ma gratitude à Mme Mylène Bédard, ma directrice de recherche, pour sa patience, sa persévérance dans le suivi, son enthousiasme et sa capacité à toujours trouver une solution à tout. J'ai par-dessus tout apprécié de travailler avec quelqu'un dont le mot d'ordre est « on y arrivera ». Merci Mylène de m'avoir donné l'opportunité de faire ce mémoire à vos côtés. Malheureusement les mots ne sont pas assez forts pour remercier comme il se doit ma directrice de recherche.

J'adresse également mes remerciements, à tous mes enseignants, qui m'ont donné la base de la science.

Enfin, je remercie les membres du jury qui ont accepté de corriger ce travail. Je veux de plus leur manifester ma très sincère reconnaissance quant au sérieux et à l'esprit critique dont ils ont fait preuve dans cet exercice.

INTRODUCTION

Les techniques de Monte Carlo par chaînes de Markov (MCCM) ont été introduites dans les années 1950. Ces méthodes sont utilisées pour échantillonner à partir de distributions de probabilité d'intérêt, distributions qui sont souvent complexes ou encore comportent plusieurs dimensions. Ces algorithmes se basent sur le parcours de chaînes de Markov ayant, pour lois stationnaires, les distributions à échantillonner. Le développement de cette méthodologie n'a pas seulement changé les solutions aux problèmes, mais a également modifié la façon d'aborder ces problèmes. Aujourd'hui, elles constituent un centre d'intérêt pour les chercheurs.

Les méthodes MCCM sont reliées de très près aux méthodes de Monte Carlo (voir [4] et [7]). Ces dernières sont utilisées dans le but d'obtenir des descriptions adéquates d'une variable aléatoire en considérant sa fonction de densité. Elles visent à obtenir un échantillon aléatoire de la distribution cible (d'intérêt) ; par la suite, en se servant de l'échantillon généré, les quantités voulues pourront être estimées de façon empirique. Cette approche est basée, d'une part, sur la capacité de facilement générer des valeurs de la distribution en question et, d'autre part, sur la capacité de produire un gros échantillon afin d'obtenir des résultats fiables.

Les méthodes MCCM ont été créées peu après l'apparition des méthodes de Monte Carlo, mais leur impact dans le domaine de la statistique n'a pas été vraiment senti jusqu'au début des années 1990. Aujourd'hui, malgré le nombre considérable d'algorithmes de simulation disponibles, les méthodes de Monte Carlo par chaîne de Markov constituent une des approches les plus utilisées dans la communauté statistique. Ces méthodes sont utilisées lorsque les algorithmes de Monte Carlo ne sont pas accessibles, c'est-à-dire dans les cas où il est difficile/impossible de générer des observations directement de la distribution cible.

L'algorithme Metropolis introduit par [10] est considéré comme le premier algorithme MCCM ; il a ensuite été généralisé par [8], résultant en l'algorithme

Metropolis-Hastings. Cette méthode nécessite la sélection d'une distribution instrumentale, qui servira à proposer des candidats pour l'échantillon. Les candidats seront ensuite soumis à une probabilité d'acceptation. Le processus résultant constitue une chaîne de Markov dont la distribution stationnaire est en fait la distribution cible. Les principales caractéristiques de cette méthode sont la facilité d'application à des distributions cibles complexes et/ou en grandes dimensions, ainsi que la non spécification de la constante de normalisation de la densité cible. Ceci consiste en un avantage important pour la communauté bayésienne, puisque cette constante est généralement difficile à obtenir pour une fonction de densité à postériori.

Le but de ce mémoire est d'analyser, et ultimement d'améliorer, une variation de l'algorithme Metropolis-Hastings récemment introduite par [1]. Cette méthode, que nous appellerons l'algorithme Metropolis à essais multiples revisité, est en fait reliée à l'algorithme Metropolis à essais multiples (MTM) de [9]. Ces deux méthodes visent à améliorer l'exploration de l'espace en générant plusieurs candidats par itération. L'algorithme MTM revisité se distingue de l'algorithme MTM standard par le fait qu'il tente d'atteindre ce but en recyclant des candidats précédemment utilisés.

Le premier chapitre de ce mémoire sera consacré à la description de l'approche MCCM. Nous introduirons les bases nécessaires pour comprendre les algorithmes, une étape importante pour se familiariser avec ces techniques. Tout d'abord, nous fournirons une description générale de la construction des algorithmes MCCM. Par la suite, nous aborderons les algorithmes Metropolis-Hastings élémentaires en détail.

Le deuxième chapitre présentera l'algorithme Metropolis à essais multiples (MTM) développé par [9]. Il sera question des généralisations et comportements de cette méthode. Nous aborderons également le concept de réversibilité dans ce contexte.

Le troisième chapitre traitera de l'algorithme Metropolis à essais multiples revisité présenté par [1]. Le but de ce chapitre sera de décrire cette méthode d'un point de vue théorique.

Étant donné le caractère novateur de cet algorithme (MTM revisité), nous nous intéressons plus particulièrement à sa performance et convergence. Ces aspects seront discutés au dernier chapitre de ce mémoire (chapitre 4). En premier lieu, nous établirons que l'algorithme MTM revisité ne se réduit pas à l'algorithme Metropolis-Hastings standard lorsqu'un seul candidat est généré à l'intérieur d'une itération donnée. Ceci nous mènera à identifier les forces et faiblesses

de cette approche, et nous tenterons finalement de la modifier dans le but d'améliorer sa performance.

Dans ce contexte, nous proposerons un tout nouvel algorithme, que nous appellerons l'algorithme Metropolis à essais multiples revisité amélioré ; cet échantillonneur vise en fait à améliorer l'algorithme MTM revisité proposé par [1] en recyclant moins de candidats à l'intérieur d'une itération donnée. Lorsque le nombre de candidats générés dans une itération donnée est petit, cette nouvelle méthode est plus performante du fait qu'elle nous permet d'affronter la sédentarité de l'algorithme MTM revisité. Cette supériorité s'estompe cependant à mesure que le nombre de candidats générés à l'intérieur d'une itération croît. Des exemples d'application de chacune des méthodes discutées seront présentés dans les chapitres correspondants.

Chapitre 1

ALGORITHMES DE MONTE CARLO PAR CHAîNE DE MARKOV (MCCM)

Les algorithmes Monte Carlo par chaînes de Markov (MCCM) sont des méthodes permettant d'échantillonner à partir de distributions de probabilité. Ils représentent une démarche alternative pour l'échantillonnage de variables aléatoires ayant des densités complexes, ainsi que pour les problèmes numériques en grandes dimensions.

Dans ce chapitre, nous décrirons brièvement la construction des algorithmes MCCM ; par la suite, nous présenterons les algorithmes MCCM élémentaires (algorithmes de Metropolis-Hastings) tout en se basant sur la théorie présentée par [16] et nous expliquerons brièvement les mesures d'efficacité utilisées dans les exemples pratiques. Enfin, nous conclurons avec quelques exemples d'application de ces méthodes.

1.1. Construction des algorithmes MCCM

Soit une fonction de densité $\pi_u(\cdot)$ par rapport à une certaine mesure μ sur un certain espace $\mathcal{X}$, qui peut être non normalisée, mais au moins satisfait la condition suivante :

$$0 < \int_{\mathcal{X}} \pi_u(x)\mu(dx) < \infty.$$

Dans le cas le plus usuel, $\mathcal{X}$ est un sous-ensemble ouvert de $\mathbb{R}^n$ et les densités sont définies par rapport à la mesure de Lebesgue. Cette mesure, qui est très importante en analyse et en probabilité, étend le concept intuitif de volume à possiblement plusieurs dimensions. Dans ce qui suit, nous supposons que $\pi_u(\cdot)$

est une densité par rapport à la mesure de Lebesgue. Par conséquent, la mesure de probabilité $\pi(\cdot)$ sur l'espace $\mathcal{X}$ est définie par :

$$\pi(A) = \frac{\int_A \pi_u(x)dx}{\int_{\mathcal{X}} \pi_u(x)dx}, \quad \forall A \subseteq \mathcal{X}.$$

Dans le reste de ce mémoire, la notation $\pi(\cdot)$ sera utilisée interchangeablement pour dénoter la densité cible normalisée ainsi que la distribution cible. Cet abus de notation est courant dans la littérature sur les méthodes MCCM.

Habituellement, les méthodes MCCM sont utilisées dans le but d'estimer l'espérance d'une fonction $f : \mathcal{X} \to \mathbb{R}$ par rapport à $\pi(\cdot)$, c'est-à-dire :

$$\pi(f) = E_\pi[f(X)] = \frac{\int_{\mathcal{X}} f(x)\pi_u(x)dx}{\int_{\mathcal{X}} \pi_u(x)dx}. \tag{1.1.1}$$

Dans le cas où $\mathcal{X}$ est de grande dimension et $\pi_u(.)$ est une fonction d'une grande complexité, une intégration directe pour les intégrales en (1.1.1) est irréalisable. La solution classique par l'approche de Monte Carlo est de générer des observations provenant de variables aléatoires indépendantes et identiquement distribuées (i.i.d.), $Z_1, Z_2, \ldots, Z_N \sim \pi(\cdot)$ et d'estimer par la suite $\pi(f)$ par

$$\hat{\pi}(f) = \frac{1}{N}\sum_{i=1}^{N} f(z_i).$$

Il en résulte un estimateur sans biais ayant un écart-type de l'ordre $O(N^{-1/2})$. Cependant, le problème qui se pose dans ce cas est que si $\pi(x)$ est compliquée, il sera alors très difficile de simuler des observations indépendantes directement de $\pi(\cdot)$.

Une solution serait d'utiliser l'approche MCCM qui consiste à construire une chaîne de Markov $\{X[n]; n \geq 0\}$ sur l'espace $\mathcal{X}$, ayant $\pi(\cdot)$ comme loi stationnaire. Il s'agit donc de définir les probabilités de transition $P(x, dy) = P(X[n+1] \in dy \mid X[n] = x)$ pour $x, y \in \mathcal{X}$, telles que :

$$\int_{x\in\mathcal{X}} \pi(dx)P(x, dy) = \pi(dy).$$

Si la chaîne de Markov est simulée pendant un grand nombre d'itérations, alors la distribution de $X[n]$ (où n est grand) sera approximativement stationnaire : $\mathcal{L}(X[n]) \simeq \pi(\cdot)$. Notons que ce résultat est valide en présence de certaines

conditions (irréductibilié et apériodicité) qui ne seront pas présentées dans ce mémoire, mais qui sont virtuellement toujours satisfaites dans le cas des algorithmes MCCM.

En pratique, il est possible d'obtenir la première observation de l'échantillon souhaité en sélectionnant le n-ième état de la chaîne de Markov. Les autres observations sont ensuite obtenues de la même manière, à partir de chaînes de Markov distinctes. Le processus s'achève lorsqu'on obtient la taille échantillonnale souhaitée.

Une pratique moins coûteuse consiste à sélectionner les états $B+1, B+2, \ldots, B+N$ pour obtenir un échantillon de taille N, où B est appelé la période de chauffe (burn-in period). La valeur B est choisie suffisamment grande de telle sorte que $\mathcal{L}(X[B]) \approx \pi(\cdot)$. Les valeurs ainsi générées sont alors corrélées, ce qui ne pose cependant pas de problème lors du calcul d'une espérance :

$$N^{-1} \sum_{i=B+1}^{B+N} f(X[i]).$$

La construction d'une telle chaîne de Markov peut sembler difficile, mais n'est certainement pas impossible. En fait, en introduisant le concept de la réversibilité, nous pourrons bâtir cette chaîne.

Définition 1.1.1. *Une chaîne de Markov avec probabilités de transition $P(x, dy)$, $x, y \in \mathcal{X}$ est réversible par rapport à la fonction de densité $\pi(\cdot)$ si*

$$\pi(dx)P(x, dy) = \pi(dy)P(y, dx) \quad \forall x, y \in \mathcal{X}. \tag{1.1.2}$$

Notons qu'en intégrant (1.1.2) par rapport à x, on obtient :

$$\begin{aligned} \int_{x\in\mathcal{X}} \pi(dx)P(x, dy) &= \int_{x\in\mathcal{X}} \pi(dy)P(y, dx) \\ &= \pi(dy) \int_{x\in\mathcal{X}} P(y, dx) \\ &= \pi(dy). \end{aligned}$$

Donc, nous pouvons conclure que si une chaîne de Markov est réversible par rapport à la distribution $\pi(\cdot)$, alors celle-ci est stationnaire pour cette chaîne. Pour construire un algorithme MCCM, il suffira donc de créer une chaîne de Markov qui est réversible par rapport à la distribution $\pi(\cdot)$ et qui peut être facilement simulée. La façon la plus simple d'atteindre ce but est d'utiliser l'algorithme Metropolis-Hastings, sujet de la section suivante.

1.2. Algorithme Metropolis-Hastings

Proposé par [10] dans un contexte de physique statistique, l'algorithme Metropolis-Hastings a ensuite été généralisé par [8] et popularisé dans la communauté statistique par [6] et [19]. Cette méthode repose sur l'utilisation d'une densité instrumentale $q(x;\cdot)$, où x est l'état actuel de la chaîne de Markov. Celle-ci sert à proposer des candidats pour la chaîne de Markov, ou autrement dit, à proposer des valeurs pour l'échantillon. L'algorithme Metropolis-Hastings ne peut être mis en pratique que si $q(x;\cdot)$ est simulable rapidement, et est soit disponible analytiquement (à une constante près), soit symétrique (c'est-à-dire $q(x;y) = q(y;x)$).

Le principe est d'utiliser ce noyau de transition instrumental pour proposer un candidat, et par la suite d'accepter ou de rejeter ce candidat selon une certaine probabilité. Lorsque le candidat est rejeté, l'état actuel est répété dans l'échantillon (c'est-à-dire la chaîne de Markov ne bouge pas).

L'algorithme Metropolis-Hastings associé à la densité d'intérêt $\pi(\cdot)$ (appelée densité cible) et à la densité instrumentale $q(x;\cdot)$ produit une chaîne de Markov $\{X[n]; n \geq 0\}$ fondée sur la transition suivante :

Algorithme 1.2.1 (Metropolis-Hastings générique)**.**
Étant donné $X[n] = x$, l'état de la chaîne de Markov au temps n,

(1) Générer une observation $Y[n+1] = y$ de la distribution instrumentale $q(x;\cdot)$;

(2) Calculer la probabilité d'acceptation :

$$\alpha(x;y) = \min\left\{1, \frac{\pi(y)q(y;x)}{\pi(x)q(x;y)}\right\}. \tag{1.2.1}$$

(3) Poser $X[n+1] = \begin{cases} y & \text{avec probabilité } \alpha(x;y); \\ x & \text{avec probabilité } 1-\alpha(x;y). \end{cases}$

Il s'agit donc, en premier lieu, de choisir un état initial $X[0] = x_0$. Au temps n, $X[n] = x$ et un nouvel état $Y[n+1] = y$ est proposé selon la loi $q(x;.)$. Une variable indépendante $Z[n+1]$ est ensuite générée, où $Z[n+1] \sim Bernoulli(\alpha(x;y))$ avec probabilité de succès définie en (1.2.1). Si $Z[n+1] = 1$, nous nous retrouvons dans le cas d'acceptation du candidat y, qui devient le nouvel état de la chaîne ($X[n+1] = y$). Sinon, $Z[n+1] = 0$ et nous rejetons $Y[n+1] = y$; $X[n] = x$ est alors posé comme nouvel état ($X[n+1] = X[n]$). En fait, cet algorithme produit une chaîne de Markov $\{X[n]; n \geq 0\}$ qui est caractérisée par sa réversibilité par rapport à $\pi(\cdot)$. Cette réversibilité est assurée par la variable aléatoire $Z[n+1]$ et la probabilité de succès $\alpha(\cdot;\cdot)$ fournie en (1.2.1).

On peut facilement vérifier la réversibilité de la chaîne de Markov comme suit : on suppose que $x \neq y$ (le cas $x = y$ est trivial) et on obtient :

$$\begin{aligned}\pi(dx)P(x,dy) &= \pi(x)\ dx\ q(x;y)\ \alpha(x;y)\ dy \\ &= \pi(x)\ dx\ q(x;y) \min\left\{1, \frac{\pi(y)\ q(y;x)}{\pi(x)\ q(x;y)}\right\}\ dy \\ &= \min\left\{\pi(x)\ q(x;y), \pi(y)\ q(y;x)\right\}\ dx\ dy\end{aligned}$$

qui est symétrique en x et y.

1.3. Algorithmes Metropolis-Hastings élémentaires

Il existe plusieurs façons de construire une distribution instrumentale, ce qui donne lieu à différentes versions de la méthode Metropolis-Hastings. Dans le cadre de cette étude, nous allons citer les plus usuelles.

1.3.1. Algorithme Metropolis-Hastings de type marche aléatoire

Un cas particulier de l'algorithme Metropolis-Hastings est l'algorithme de type marche aléatoire, pour lequel il faut prendre en compte la valeur présente de la chaîne de Markov afin de générer un candidat pour le prochain intervalle de temps. Il s'agit de simuler le candidat $Y[n+1]$ de la manière suivante :

$$Y[n+1] = X[n] + \epsilon[n+1]$$

où $\epsilon[n+1]$ est une perturbation aléatoire indépendante de $X[n]$. Par exemple, $\epsilon[n+1] \sim Unif(-\sigma, \sigma)$, ou encore $\epsilon[n+1] \sim N(0, \sigma^2)$. La densité instrumentale d'un tel algorithme est de la forme $q(x;y) = q(y-x)$.

La chaîne de Markov formée par les candidats $\{Y[n]; n \geq 1\}$ est alors une marche aléatoire, mais la chaîne de Markov de l'algorithme Metropolis-Hastings $\{X[n]; n \geq 0\}$ n'en est pas une à cause de l'étape d'acceptation de l'algorithme Metropolis-Hastings. Si, de plus, la densité instrumentale est symétrique, c'est-à-dire, $q(x;y) = q(|x-y|)$, alors la probabilité d'acceptation s'en trouve simplifiée.

L'algorithme associé à cette approche (marche alétoire symétrique) est l'algorithme original proposé par [10].

Algorithme 1.3.1 (Metropolis-Hastings symétrique de type marche aléatoire)**.** *Étant donné $X[n] = x$, l'état de la chaîne de Markov au temps n,*

(1) Générer une observation $Y[n+1] = y$ de la densité instrumentale $q(x;\cdot)$

(2) Calculer la probabilité d'acceptation :

$$\alpha(x;y) = \min\left\{1, \frac{\pi(y)}{\pi(x)}\right\}$$

(3) Prendre $X[n+1] = \begin{cases} y & \textit{avec probabilité} \quad \alpha(x;y); \\ x & \textit{avec probabilité} \quad 1-\alpha(x;y). \end{cases}$

Il est clair que la probabilité d'acceptation ne dépend pas de la densité instrumentale. Ceci signifie que pour une densité cible $\pi(\cdot)$ et un couple $(X[n], Y[n+1])$ donnés, la probabilité d'acceptation est la même quelle que soit la densité instumentale utilisée pour générer $Y[n+1]$, en autant que cette densité soit symétrique par rapport à $X[n]$. Autrement dit, générer $Y[n+1]$ selon une loi normale $N(X[n], \sigma^2)$ ou selon toute autre loi symétrique (par exemple $Unif(X[n] - \sigma, X[n] + \sigma)$) donnerait lieu à la même probabilité d'acceptation.

1.3.2. Algorithme Metropolis-Hastings indépendant

L'échantillonneur indépendant (voir [14]) est un cas particulier de l'algorithme Metropolis-Hastings dans lequel la densité instrumentale ne dépend pas de l'état présent de la chaîne de Markov, c'est-à-dire $q(x;y) = q(y)$. On obtient alors un cas particulier de l'algorithme générique, qui se résume ainsi :

Algorithme 1.3.2 (Échantillonneur indépendant)**.**
Étant donné $X[n] = x$*, l'état de la chaîne de Markov au temps* n*,*

(1) Générer une observation $Y[n+1] = y$ *de la densité instrumentale* $q(x;\cdot)$

(2) Calculer la probabilité d'acceptation :

$$\alpha(x;y) = \min\left\{1, \frac{\pi(y)q(x)}{\pi(x)q(y)}\right\}$$

(3) Prendre $X[n+1] = \begin{cases} y & \textit{avec probabilité} \ \ \alpha(x;y); \\ x & \textit{avec probabilité} \ \ 1-\alpha(x;y). \end{cases}$

Bien que les candidats $Y[n+1]$ $(n = 0, 1, \ldots)$ soient générés de manière indépendante, l'échantillon obtenu à l'aide d'un algorithme Metropolis-Hastings indépendant n'est pas i.i.d., étant donné que la probabilité d'acceptation de $Y[n+1]$ dépend de $X[n]$. Une exception est le cas d'un échantillonneur indépendant pour lequel $q = \pi$, ce qui revient à échantillonner directement de la densité cible et ainsi à toujours accepter les états proposés.

1.3.3. Échantillonneur de Gibbs (Gibbs Sampler)

L'échantillonneur de Gibbs (voir [18]) est une méthode populaire qui fait partie des algorithmes MCCM. Il constitue une bonne manière d'échantillonner les distributions conjointes de plusieurs variables, en autant que les distributions conditionnelles soient disponibles pour chacune des variables de la distribution cible. En effet, il est plus simple de générer des observations à partir de densités conditionnelles plutôt qu'à partir de densités conjointes. L'échantillonneur de Gibbs est basé sur cette approche.

Par exemple, des observations d'une distribution bivariée pour (X, Y), peuvent être facilement simulées avec l'échantillonneur de Gibbs, en utilisant les distributions conditionnelles plutôt que la distribution conjointe. En commençant par un choix arbitraire pour $X[0]$ et $Y[0]$, $X[1]$ est simulé à partir de la distribution conditionnelle de $X \mid Y[0]$, et $Y[1]$ est simulé à partir de la distribution conditionnelle $Y \mid X[1]$. L'alternance entre les deux distributions conditionnelles, dans les étapes subséquentes, génère un échantillon qui provient de la distribution conjointe de X et Y.

L'échantillonneur de Gibbs peut comporter deux étapes seulement, ou peut être généralisé à plusieurs étapes. Même si le premier est un cas particulier du second, l'échantillonneur à deux étapes a de meilleures propriétés de convergence, tel que mentionné dans [13]. De manière plus formelle, si deux variables X et Y ont pour densité conjointe $f(x, y)$, avec les densités conditionnelles $f_{Y|X}$ et $f_{X|Y}$, l'échantillonneur de Gibbs à deux étapes génère une chaîne de Markov $\{(X[n], Y[n]); n \geq 0\}$ selon l'algorithme suivant :

Algorithme 1.3.3 (Échantillonneur de Gibbs)**.**
Étant donné $X[n] = x$ et $Y[n] = y$, l'état de la chaîne de Markov au temps n,

(1) Générer une observation $X[n+1] = x'$ de la densité $f_{X|Y}(\cdot \mid y)$;

(2) Générer une observation $Y[n+1] = y'$ de la densité $f_{Y|X}(\cdot \mid x')$.

Notons que l'échantillonneur de Gibbs est une composition de plusieurs algorithmes de type Metropolis-Hastings avec probabilités d'acceptation uniformément égales à 1.

1.4. Différences entre les algorithmes Metropolis-Hastings de type marche aléatoire et indépendant

L'algorithme Metropolis-Hastings de type marche aléatoire décrit dans l'algorithme 1.3.1 est l'une des méthodes MCCM les plus utilisées. Il apparaît souvent

comme un algorithme générique qui convient pour la plupart des distributions cibles. Ce type d'algorithme nécessite parfois un très grand nombre d'itérations afin de faire face à certaines difficultés potentielles de convergence. De plus, à cause de sa symétrie, il passe environ la moitié des itérations à visiter des régions déjà exploitées.

Le seul paramètre à ajuster dans cette méthode est l'échelonnage de la distribution instrumentale $q(x;\cdot)$ qui est un paramètre d'implémentation. La figure 1.1 montre l'évolution des trajectoires d'une chaîne de Markov simulée à partir d'un algorithme Metropolis-Hastings de type marche aléatoire avec une loi instrumentale $N(x,\sigma)$ pour différentes valeurs de σ (grand, petit et moyen).

Les résultats montrent clairement la différence entre les chaînes de Markov obtenues. Si σ est trop grand, presque tous les déplacements proposés sont refusés (car $\pi(y) \ll \pi(x)$) et la chaîne de Markov reste longtemps au même point. Si σ est trop petit, les candidats sont presque tous acceptés (car $\pi(y) \approx \pi(x)$), mais la chaîne de Markov se déplace très lentement et elle mettra beaucoup de temps à converger vers $\pi(\cdot)$.

Comme on le remarque dans cet exemple, calibrer l'échelle σ de la marche aléatoire est crucial pour obtenir une approximation de la densité cible en un nombre d'itérations raisonnable ; nous y reviendrons d'ailleurs dans la section 1.6. Par contre, la probabilité d'acceptation ne dépend pas de la densité instrumentale, ce qui signifie que la probabilité d'acceptation est toujours la même étant donné le couple $(X[n], Y[n+1]) = (x, y)$ et la densité cible $\pi(\cdot)$. Évidemment, le fait de modifier la densité cible aura un impact sur la probabilité d'acceptation des candidats.

L'implémentation d'un échantillonneur indépendant nécessite davantage d'information au sujet de la densité cible, puisque la densité instrumentale est indépendante de l'état présent. De façon générale, plus la densité instrumentale se rapproche de la densité cible, plus la convergence de l'algorithme sera rapide. Par conséquent, dans le but de bénéficier d'une convergence efficace, une connaissance plus appronfondie de la distribution cible est nécessaire afin de choisir et de calibrer judicieusement la distribution instrumentale. En effet, avec un choix adéquat de la densité instrumentale, la convergence de l'algorithme de type indépendant est meilleure (plus rapide) que celle de l'algorithme de type marche aléatoire.

1.5. Les mesures d'efficacité

Afin de comparer les résultats obtenus avec les différentes méthodes MCCM, il importe d'introduire des façons de mesurer et de comparer l'efficacité de différents

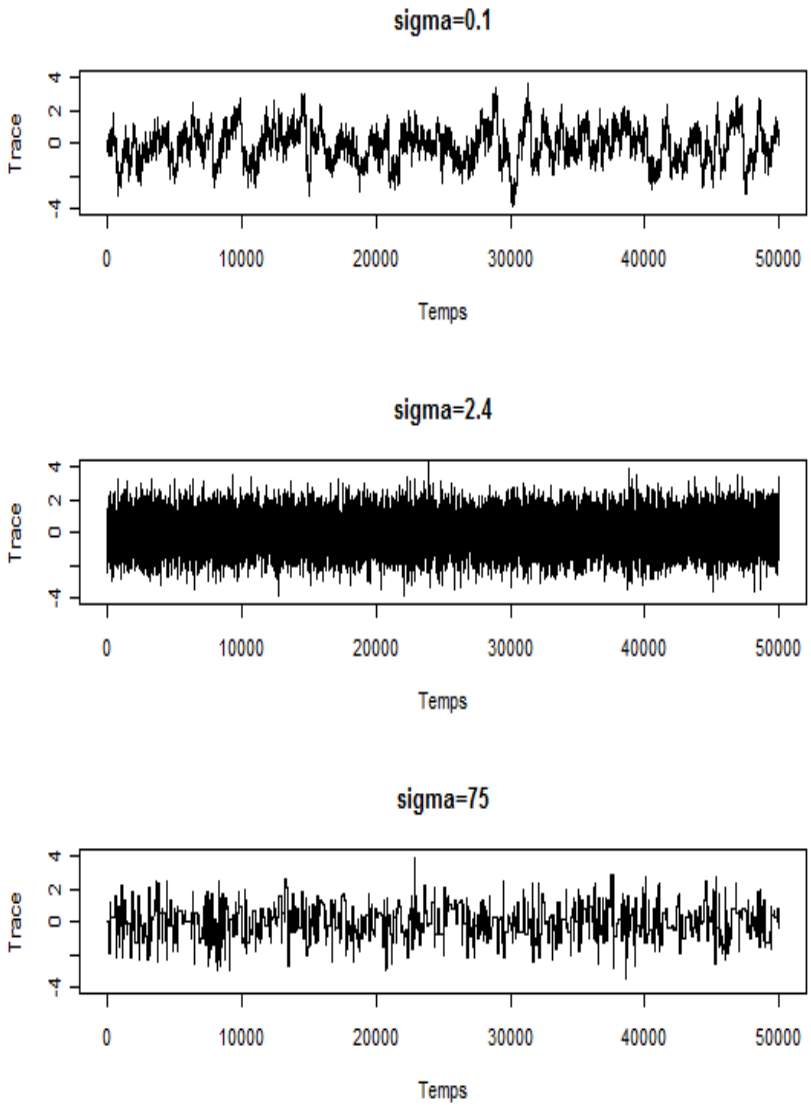

FIGURE 1.1. Réalisation d'une chaîne de Markov d'un algorithme Metropolis-Hastings par marche aléatoire pour une loi instrumentale $N(x, \sigma)$ et une densité cible normale pour σ petit (σ=0.1), σ moyen (σ=2.4) et σ grand (σ=75).

algorithmes. Dans ce cadre, nous allons présenter quelques mesures d'efficacité couramment utilisées.

1.5.1. Erreur quadratique moyenne de Monte Carlo

L'erreur quadratique moyenne de Monte Carlo comporte deux termes : un terme de biais et un terme de variance (voir [3] pour un exemple d'application).

Supposons que nous cherchons à générer des valeurs d'une certaine variable β. Afin de calculer le terme de biais, il est nécessaire de connaître $E[\beta]$. Le biais consiste alors à calculer la différence entre $\overline{b}_{..}$, un estimateur de β (basé sur l'ensemble de la chaîne de Markov) et l'espérance théorique $E[\beta]$.

Pour calculer le terme de variance, nous devons diviser la chaîne de Markov en τ sous-échantillons de taille η. En dénotant l'estimateur de β basé sur le i-ième sous-échantillon par $\overline{b}_{i.}$, le terme de variance mesure la variabilité entre les estimateurs $\overline{b}_{i.}, i = 1, \ldots, \tau$ et l'estimateur global $\overline{b}_{..}$.

L'erreur quadratique moyenne de Monte Carlo est exprimée comme suit :

$$EQM_{MC}(\beta) = (\overline{b}_{..} - E[\beta])^2 + \frac{1}{\tau - 1} \sum_{i=1}^{\tau} (\overline{b}_{i.} - \overline{b}_{..})^2,$$

où $\overline{b}_{i.} = \sum_{j=1}^{\eta} b_i[j]/\eta$ pour $i = 1, 2, \ldots, \tau$ et $\overline{b}_{..} = \sum_{i=1}^{\tau} \sum_{j=1}^{\eta} b_i[j]/(\tau\eta)$. Nous cherchons à minimiser cette quantité.

1.5.2. La variation quadratique moyenne

La variation quadratique moyenne (voir [11]) est une autre mesure d'efficacité, qui a l'avantage d'être une fonction de l'échantillon seulement (et non d'un estimateur en particulier). La variation quadratique moyenne est obtenue à partir de la formule suivante :

$$AQV = \frac{1}{N} \sum_{i=1}^{d} \sum_{j=1}^{N} (b^{(i)}[j] - b^{(i)}[j-1])^2,$$

où N est le nombre d'itérations, d est la dimension de la chaîne de Markov et $b[j] = (b^{(i)}[j], \cdots, b^{(d)}[j])$ représente la j-ième itération de la chaîne de Markov générant des observations d'un paramètre (multidimensionnel) β. Plus cette mesure est élevée, plus la chaîne explore efficacement son espace.

1.5.3. La fonction d'autocorrélation

La fonction d'autocorrélation (acf) mesure la dépendance linéaire entre une valeur du processus d'une série chronologique au temps t et les valeurs passées ou futures de ce processus. Elle est définie comme suit :

$$\rho(s,t) = \frac{\gamma(s,t)}{\sqrt{\gamma(s,s)\gamma(t,t)}} \quad \forall \ \ s,t,$$

tel que

$$\begin{aligned}\gamma(s,t) &= cov\{y_t, y_s\},\\ &= E\{(y_t - \mu_t)(y_s - \mu_s)\},\end{aligned}$$

est la fonction d'autocovariance de $\{y_t\}$ et $\mu_t = E(y_t)$.

Un processus est dit stationnaire si $\mu_t = \mu$ pour tout t et la fonction d'autocovariance dépend uniquement de $|s-t|$. Dans ce cas, nous pouvons décrire la fonction d'autocovariance comme une fonction du délai h. Elle peut être exprimée comme suit :

$$\gamma(h) = cov\{y_t, y_{t-h}\}.$$

Ainsi, la fonction d'autocorrélation d'un processus stationnaire est définie également en fonction du délai h :

$$\rho(h) = \frac{\gamma(h)}{\gamma(0)}$$

et possède les caractéristiques suivantes :

- $\rho(h)$ prend les valeurs dans l'intervalle $[-1, 1]$;
- $\rho(h) = \rho(-h)$;
- $\rho(h) = 0$ si y_t est indépendant de y_{t-h}.

Pour plus de détails, nous référons le lecteur à [12] et [17].

Sous l'hypothèse de stationnarité de notre chaîne de Markov, il est possible d'obtenir des estimateurs convergents pour la fonction d'autocorrélation. L'estimateur naturel de la fonction d'autocovariance est

$$\hat{\gamma}(h) = \frac{1}{n}\sum_{i=1}^{n-h}(y_i - \hat{\mu}_n)(y_{i+h} - \hat{\mu}_n),$$

ce qui nous mène à la fonction d'autocorrélation empirique $\hat{\rho}(h) = \hat{\gamma}(h)/\hat{\gamma}(0)$.

La mesure d'efficacité qui sera utilisée dans ce mémoire est la mesure de la fonction d'autocorrélation et nous utiliserons la fonction acf dans le logiciel statistique **R** pour obtenir les estimateurs.

1.6. Exemples pratiques

Dans cette section, nous présenterons des exemples en appliquant différents types d'algorithmes Metropolis-Hastings avec différents paramètres. Pour ce faire, nous avons programmé les algorithmes à l'aide du logiciel statistique `R` (programmes en annexe).

Tout d'abord, nous allons présenter trois exemples pour l'algorithme Metropolis-Hastings par marche aléatoire avec une densité cible normale, $N(0,1)$, et une innovation normale (générée à partir d'une normale) en utilisant différentes valeurs du paramètre σ. Par la suite, nous allons présenter deux exemples pour l'algorithme Metropolis-Hastings indépendant avec une densité cible normale, $N(0,1)$, et une innovation normale en utilisant deux valeurs différentes de la moyenne μ.

Les résultats obtenus sont présentés dans les figures qui suivent. Chaque figure contient trois graphes : celui du haut représente l'extrait (ou la trace) de la chaîne de Markov ; celui du centre illustre l'histogramme des observations ; celui du bas présente la fonction d'autocorrélation (acf).

La figure 1.2 a été obtenue à l'aide d'un algorithme Metropolis-Hastings avec marche aléatoire pour une densité cible $N(0,1)$ en générant 50 000 itérations à partir d'une densité intrumentale $N(x, \sigma = 0.1)$. La figure 1.3 a été obtenue à l'aide du même algorithme, mais avec un paramètre d'échelonnage différent : cette fois-ci, nous avons généré les candidats à partir d'une densité instrumentale normale $N(x, 2.4)$. De façon similaire, la figure 1.4 a été obtenue en utilisant une instrumentale $N(x, 75)$. En comparant les trois résultats, nous remarquons que l'échelle de la marche aléatoire est cruciale pour obtenir une bonne approximation de la loi cible. En effet, nous observons clairement la différence entre les chaînes obtenues avec les paramètres 0.1, 2.4 et 75. Pour le premier paramètre, la chaîne de Markov se déplace lentement et elle mettra longtemps à converger vers $\pi(\cdot)$. Dans le cas du deuxième paramètre, qui est le plus approprié, la chaîne de Markov se déplace très rapidement et elle mettra peu de temps à converger vers $\pi(\cdot)$. Pour le troisième paramètre, presque tous les déplacements proposés sont refusés ; par conséquent, la chaîne de Markov reste au même état sur de long intervalles de temps.

Les figures 1.5 et 1.6 illustrent les résultats obtenus à l'aide de 50 000 itérations d'algorithmes Metropolis-Hastings indépendants pour une densité cible normale,

$N(0,1)$. La figure 1.5 est obtenue en générant des candidats à partir d'une densité instrumentale normale $N(1.2,1)$; la figure 1.6 est obtenue en générant des candidats à partir d'une densité instrumentale normale, $N(0.25,1)$. Il est clair que la chaîne de Markov obtenue à partir d'une instrumentale $N(1.2,1)$ se déplace très lentement et elle mettra beaucoup de temps à converger vers $\pi(\cdot)$. La chaîne de Markov obtenue à partir d'une $N(0.25,1)$ se déplace plus rapidement et elle mettra moins de temps à converger vers la densité cible $\pi(\cdot)$. Ce phénomène n'est pas surprenant puisque la deuxième densité instrumentale se rapproche davantage de la densité cible.

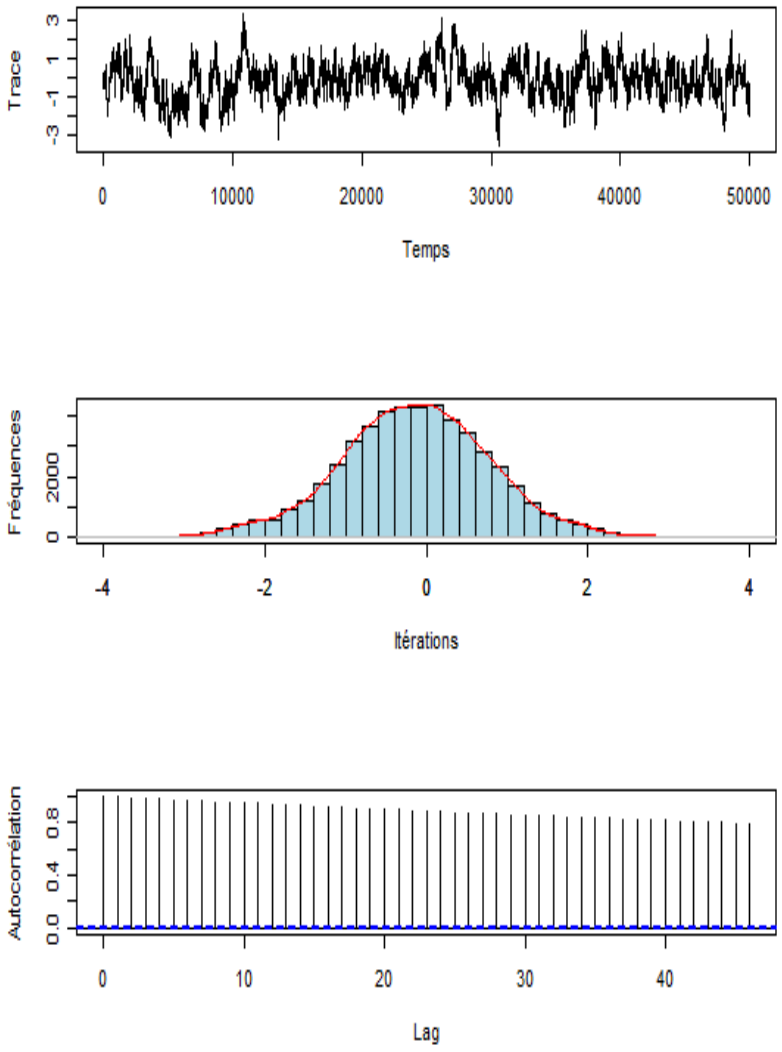

FIGURE 1.2. Algorithme Metropolis-Hastings par marche aléatoire pour une loi instrumentale $N(x, 0.1)$ et une densité cible normale, $N(0, 1)$.

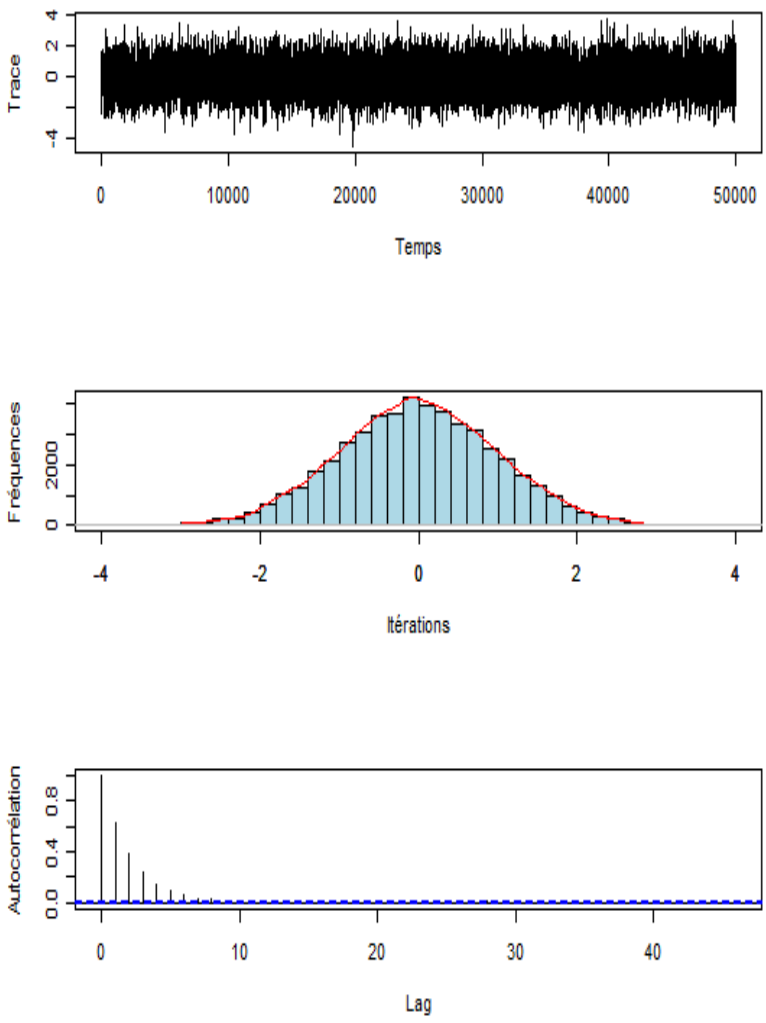

FIGURE 1.3. Algorithme Metropolis-Hastings par marche aléatoire pour une loi instrumentale $N(x, 2.4)$ et une densité cible normale, $N(0, 1)$.

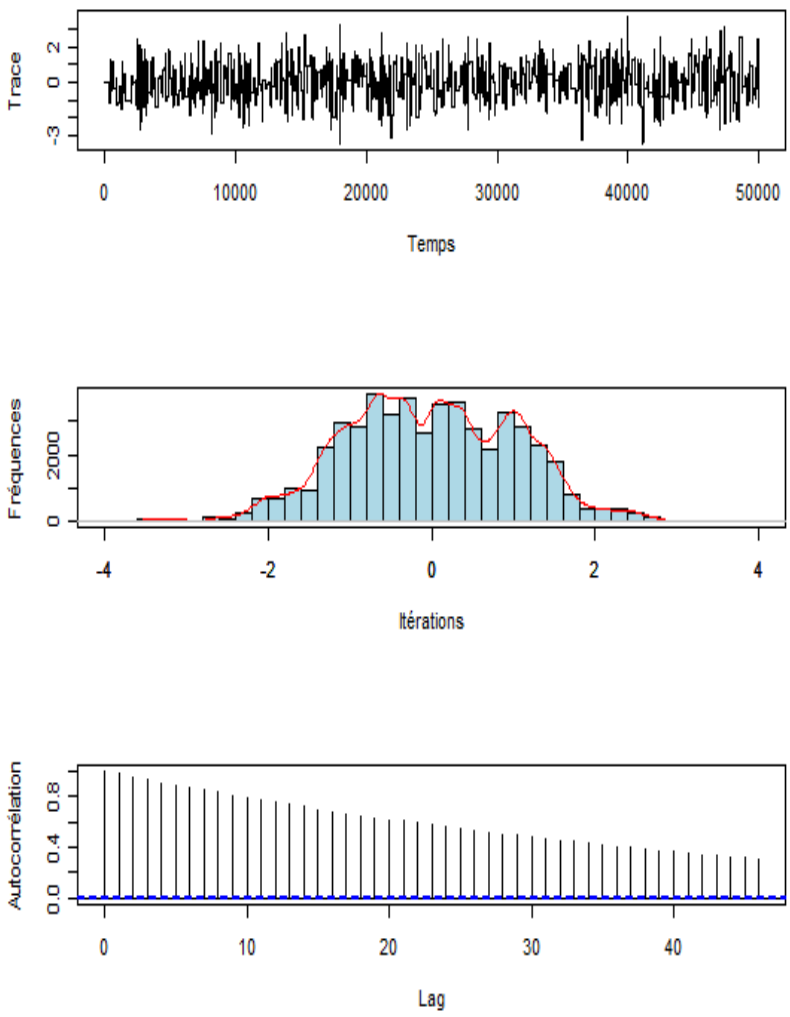

FIGURE 1.4. Algorithme Metropolis-Hastings par marche aléatoire pour une loi instrumentale $N(x, 75)$ et une densité cible normale, $N(0, 1)$.

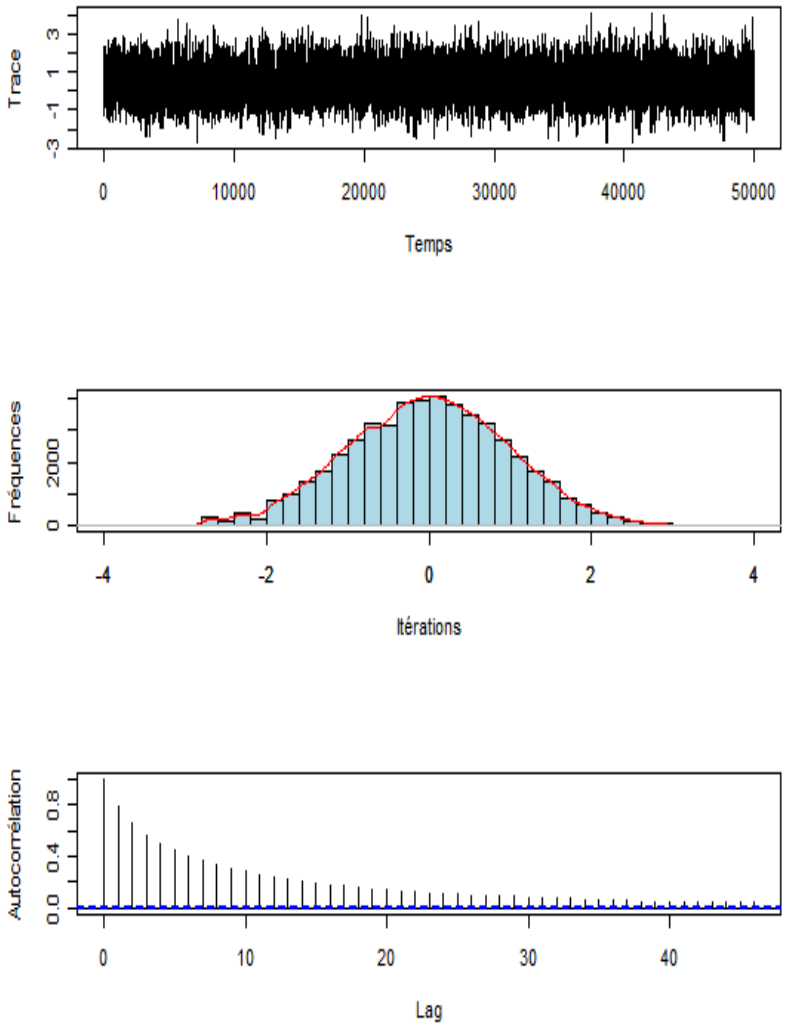

FIGURE 1.5. Algorithme Metropolis-Hastings indépendant pour une loi instrumentale $N(1.2, 1)$ et une densité cible normale, $N(0, 1)$.

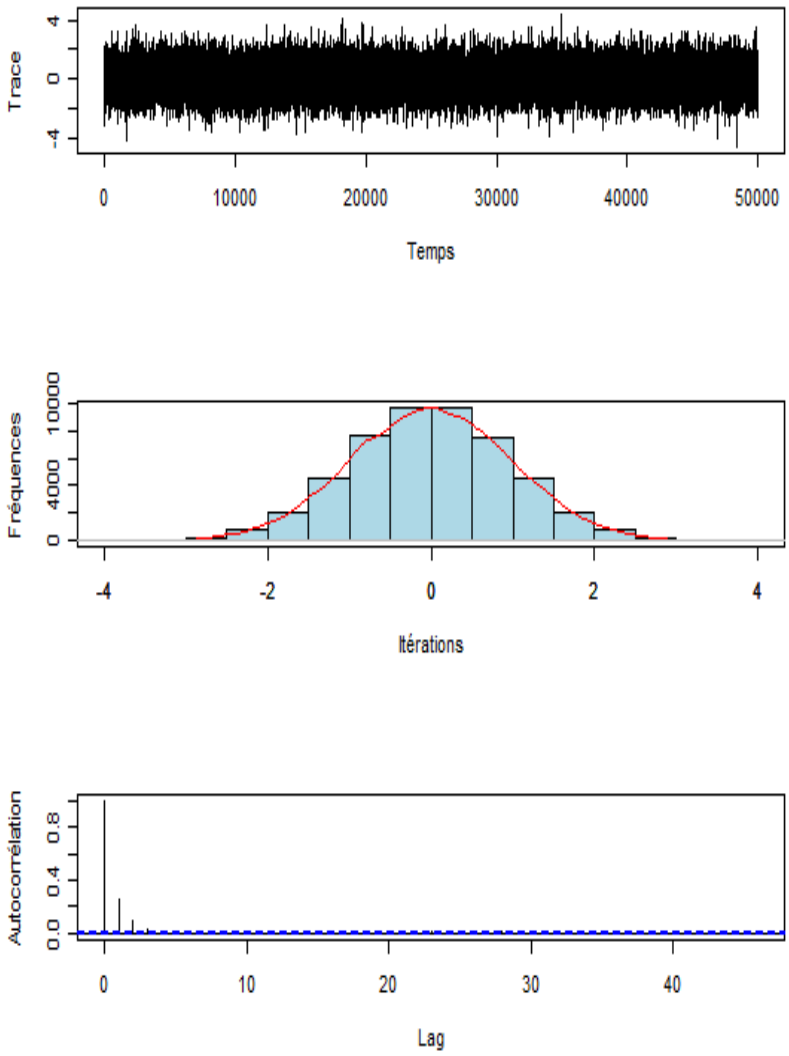

FIGURE 1.6. Algorithme Metropolis-Hastings indépendant pour une loi instrumentale $N(0.25, 1)$ et une densité cible normale $N(0, 1)$.

Chapitre 2

ALGORITHME METROPOLIS À ESSAIS MULTIPLES

L'algorithme Metropolis à essais multiples (multiple-try Metropolis - MTM) est une variation de la méthode Metropolis-Hastings, qui a été introduite par [9] dans la littérature statistique. Cet échantillonneur vise à améliorer l'exploration de l'espace dans une itération donnée, en intégrant la notion d'optimisation locale. Ceci est accompli en générant plusieurs candidats de façon simultanée à l'intérieur d'une itération ; par la suite, un seul de ces candidats est sélectionné, et proposé comme prochain état potentiel pour la chaîne de Markov. Naturellement, cette amélioration de l'algorithme Metropolis-Hastings n'est pas implémentée à coût nul puisqu'elle nécessite la génération d'un plus grand nombre de valeurs.

Dans ce chapitre, nous décrirons l'algorithme Metropolis à essais multiples ; par la suite, nous discuterons des généralisations de cet algorithme et nous présenterons son comportement asymptotique avant de faire le point sur les algorithmes réversibles. Enfin, nous conclurons avec quelques exemples d'application de cette méthode.

2.1. Implémentation de l'algorithme

L'algorithme MTM modifie l'algorithme Metropolis-Hastings standard en remplaçant la proposition unique y par un ensemble de K propositions i.i.d. $y_1, y_2, \ldots, y_K$. Ces K propositions sont générées d'une distribution instrumentale avec densité $q(x, \cdot)$. Par la suite, un seul candidat parmi ces K propositions est retenu ; ce candidat sera alors accepté avec une certaine probabilité.

Supposons que $q(y; x) > 0$ si et seulement si $q(x; y) > 0$. Soit $\lambda(x; y)$, une fonction non négative et symétrique en x et y telle que $\lambda(x; y) > 0$ lorsque $q(x; y) > 0$. Sous ces suppositions, nous définissons la fonction de poids suivante :

$$w(x;y) = \pi(x)q(x;y)\lambda(x;y). \tag{2.1.1}$$

Une itération de l'algorithme MTM est maintenant décrite.

Algorithme 2.1.1 (Metropolis à essais multiples)**.**
Étant donné $X[n] = x$, l'état de la chaîne de Markov au temps n,

(1) – *(a) Générer K propositions i.i.d. $y_1, y_2, \ldots, y_K$ de la densité $q(x;\cdot)$.*
– *(b) Calculer $w(y_i;x)$ pour $i = 1, 2, \ldots, K$.*

(2) – *(a) Sélectionner $Y[n+1] = y$ parmi les propositions $y_1, y_2, \ldots, y_K$, avec probabilité proportionnelle à $w(y_i;x)$, $i = 1, 2, \ldots, K$.*
– *(b) Générer $x_1^*, x_2^*, \ldots, x_{K-1}^*$ de la densité $q(y;\cdot)$ et poser $x_K^* = x$.*
– *(c) Calculer ${w_i}^* = w(x_i^*;y)$, $i = 1, 2, \ldots, K$.*

(3) Calculer la probabilité d'acceptation :

$$\begin{aligned}\alpha(\boldsymbol{x}_{1:K}^*;\boldsymbol{y}_{1:K}) &\equiv \alpha(x_1^*,\ldots,x_K^*;y_1,\ldots,y_K)\\ &= \min\left\{1, \frac{w(y_1;x)+\cdots+w(y_K;x)}{w(x_1^*;y)+\cdots+w(x_K^*;y)}\right\}.\end{aligned}$$

(4) Prendre $X[n+1] = \begin{cases} y & \text{avec probabilité } \alpha(\boldsymbol{x}_{1:K}^*;\boldsymbol{y}_{1:K});\\ x & \text{avec probabilité } 1-\alpha(\boldsymbol{x}_{1:K}^*;\boldsymbol{y}_{1:K}).\end{cases}$

La condition de réversibilité est également satisfaite dans le cas de l'algorithme MTM. En effet, la génération d'un échantillon de variables auxiliaires $x_1^*, \ldots, x_K^*$ fait en sorte que la règle de transition satisfait la condition d'équilibre, et donc induit une chaîne de Markov réversible par rapport à la densité cible $\pi(\cdot)$. La démonstration a été publiée par [9], et est présentée ci-dessous.

Soit $P(x,y)$, la probabilité de transition pour se déplacer de l'état $X[n] = x$ à l'état y à l'aide d'un échantillonneur MTM. Supposons que $x \neq y$ et posons I, l'indicatrice que la proposition y_j est sélectionnée. Étant donné

$$w(y;x) = \pi(y)q(y;x)\lambda(y;x)$$

et puisque les y_j sont permutables, alors nous obtenons :

$$\begin{aligned}\pi(x)P(x,y) &= \pi(x)P\left[\cup_{i=1}^{K}[(Y_i = y)\cap(I=i)] \mid x\right] \\ &= K\pi(x)P\left[(Y_K = y)\cap(I=K) \mid x\right] \\ &= K\pi(x)\int\cdots\int q(x;y)q(x;y_1)\cdots q(x;y_{K-1}) \\ &\times \frac{w(y;x)}{w(y;x)+\sum_{i=1}^{K-1}w(y_i;x)} \\ &\times \min\left(1, \frac{w(y;x)+\sum_{i=1}^{K-1}w(y_i;x)}{w(x;y)+\sum_{i=1}^{K-1}w(x_i^*;y)}\right) \\ &\times q(y;x_1^*)\cdots q(y;x_{K-1}^*)dy_1\cdots dy_{K-1}dx_1^*\cdots dx_{K-1}^*.\end{aligned}$$

En utilisant le fait que

$$\frac{w(x;y)}{\lambda(x;y)} = \pi(x)q(x;y),$$

nous pouvons écrire

$$\begin{aligned}\pi(x)P(x,y) = K\frac{w(x;y)w(y;x)}{\lambda(x;y)}\int\cdots\int & \\ \min\left(\frac{1}{w(y;x)+\sum_{i=1}^{K-1}w(y_i;x)}, \frac{1}{w(x;y)+\sum_{i=1}^{K-1}w(x_i^*;y)}\right) & \\ \times q(x;y_1)\cdots q(x;y_{K-1})q(y;x_1^*)\cdots q(y;x_{K-1}^*) & \\ \times dy_1\cdots dy_{K-1}dx_1^*\cdots dx_{K-1}^*. &\end{aligned} \tag{2.1.2}$$

Puisqu'on a $\lambda(x;y) = \lambda(y;x)$, alors l'expression (2.1.2) est symétrique en x et y, donc

$$\pi(x)P(x,y) = \pi(y)P(y,x).$$

Plusieurs options sont possibles pour le choix de la fonction symétrique $\lambda(x;y)$ de l'algorithme MTM. Par exemple, le cas le plus simple est de choisir $\lambda(x;y) \equiv 1$. Un autre exemple serait de prendre :

$$\lambda(x;y) = \frac{2}{q(x;y)+q(y;x)},$$

avec $q(x;y)$ symétrique. Ce dernier choix de $\lambda(x;y)$ implique que $w(x;y) = \pi(x)$ et mène à l'algorithme suivant :

Algorithme 2.1.2 (Metropolis à essais multiples symétrique)**.**
Étant donné $X[n] = x$, l'état de la chaîne de Markov au temps n,

(1) Générer K propositions i.i.d. $y_1, y_2, \ldots, y_K$ de la densité symétrique $q(x;\cdot)$.

(2) – *(a) Sélectionner* $Y[n+1] = y$ *parmi les propositions* $y_1, y_2, \ldots, y_K$*, avec probabilité proportionnelle à* $\pi(y_i)$*,* $i = 1, 2, \ldots, K$*.*
– *(b) Générer* $x_1^*, x_2^*, \ldots, x_{K-1}^*$ *de la densité symétrique* $q(y; \cdot)$ *et poser* $x_K^* = x$*.*

(3) Calculer la probabilité d'acceptation :

$$\begin{aligned}\alpha(\boldsymbol{x}_{1:K}^*; \boldsymbol{y}_{1:K}) &= \alpha(x_1^*, \ldots, x_K^*; y_1, \ldots, y_K) \\ &= \min\left\{1, \frac{\pi(y_1) + \cdots + \pi(y_K)}{\pi(x_1^*) + \cdots + \pi(x_K^*)}\right\}.\end{aligned}$$

(4) Prendre $X[n+1] = \begin{cases} y & \text{avec probabilité } \alpha(\boldsymbol{x}_{1:K}^*; \boldsymbol{y}_{1:K}); \\ x & \text{avec probabilité } 1 - \alpha(\boldsymbol{x}_{1:K}^*; \boldsymbol{y}_{1:K}). \end{cases}$

2.1.1. Généralisations

En se référant au développement de la réversibilité présenté à la section précédente, il est facile d'observer que les K propositions $y_1, y_2, \ldots, y_K$ générées à partir de la loi instrumentale $q(x; \cdot)$ n'ont pas besoin d'être indépendantes. En effet, si la fonction de transition dépend d'une autre variable aléatoire, disons e, dont la distribution échantillonnale est $f_x(e)$, alors nous pouvons écrire

$$q(x; y) = \int q_e(x; y) f_x(e) de.$$

Il est alors possible de générer e en premier, par la suite, nous pourrons générer l'ensemble des K propositions $y_1, y_2, \ldots, y_K$ à partir de $q_e(x; \cdot)$. Dans ce cas, les y_j sont encore permutables, mais ils ne sont plus indépendants.
Définissons

$$w_e(x; y) = \pi(x) f_x(e) q_e(x; y) \lambda_e(x; y),$$

où $\lambda_e(x; y)$ est une fonction positive symétrique en x et y.

En conséquence, la règle de transition de l'algorithme Metroplis à essais multiples peut être modifiée de la façon suivante :

Algorithme 2.1.3 (Algorithme Metropolis à essais multiples généralisé)**.**
Étant donné $X[n] = x$*, l'état de la chaîne de Markov au temps* n*,*

(1) – *(a) Générer une observation* e *de la densité* $f_x(\cdot)$ *;*
– *(b) Générer* K *propositions i.i.d.* $y_1, y_2, \ldots, y_K$ *de la densité* $q_e(x; \cdot)$*.*

(2) – *(a) Sélectionner* $Y[n+1] = y$ *parmi les propositions* $y_1, y_2, \ldots, y_K$*, avec probabilité proportionnelle à* $w_e(y_i; x)$*,* $i = 1, 2, \ldots, K$*.*

– *(b) Générer* $x_1^*, x_2^*, \ldots, x_{K-1}^*$ *de la densité* $q_e(y;\cdot)$ *et poser* $x_K^* = x$.

(3) Calculer la probabilité d'acceptation :

$$\begin{aligned}\alpha(\boldsymbol{x}_{1:K}^*; \boldsymbol{y}_{1:K}) &= \alpha(x_1^*, \ldots, x_K^*; y_1, \ldots, y_K) \\ &= \min\left\{1, \frac{w_e(y;x) + \sum_{i=1}^{K-1} w_e(y_i;x)}{w_e(x;y) + \sum_{i=1}^{K-1} w_e(x_i{}^*;y)}\right\}\end{aligned} \tag{2.1.3}$$

(4) Prendre $X[n+1] = \begin{cases} y & \textit{avec probabilité } \alpha(\boldsymbol{x}_{1:K}^*; \boldsymbol{y}_{1:K}); \\ x & \textit{avec probabilité } 1 - \alpha(\boldsymbol{x}_{1:K}^*; \boldsymbol{y}_{1:K}). \end{cases}$

La probabilité d'acceptation présentée en (2.1.3) peut être simplifiée lorsque la distribution échantillonnale de e est indépendante de la position de x ou de y. Dans ce cas, nous ignorons $f_x(e)$ dans la définition de $w_e(x;y)$, qui devient alors

$$w_e(x;y) = \pi(x) q_e(x;y) \lambda_e(x;y).$$

Il existe d'autres généralisations possibles de l'algorithme Metropolis à essais multiples. On peut citer, à titre d'exemple, le cas où l'ensemble des propositions $y_1, y_2, \ldots, y_K$ peuvent être générées à partir de la chaîne de Markov $\{X[n]; n \geq 0\}$. Mentionnons également les versions introduites dans [5] et [2], qui considèrent des propositions dépendantes. Dans tous ces cas, il est évidemment nécessaire de modifier la façon d'obtenir les variables auxiliaires $x_1^*, x_2^*, \ldots, x_K^*$ afin de préserver la propriété de réversibilité de la chaîne de Markov.

2.1.2. Comportement asymptotique

Comme nous le savons déjà, l'échantillonneur MTM génère plusieurs candidats simultanément à l'intérieur d'une seule itération. L'objectif est de choisir, par la suite, un seul de ces candidats qui sera proposé comme le prochain état potentiel pour la chaîne de Markov.

Il serait intéressant de mieux comprendre le comportement de cet échantillonneur, dont la performance peut être ajustée à l'aide de la variance instrumentale σ^2. Celle-ci devrait donc être choisie de manière à obtenir une chaîne de Markov qui explore l'espace le plus rapidement possible.

À cet effet, des résultats asymptotiques liés à l'efficacité de l'échantillonneur MTM pour des distributions cibles de grande dimension et formées de composantes i.i.d. ont été obtenus récemment.

Considérons la densité cible de dimension d suivante :

$$\pi(x) = \prod_{i=1}^{d} f(x_i),$$

où la densité f doit satisfaire certaines conditions de régularité (pour plus de détails, nous référons le lecteur à [2] et [3]).

Pour exprimer le résultat en question, une formulation alternative de la variance instrumentale se révèle utile :

$$\sigma^2 = \frac{\tilde{\ell}^2}{\mathtt{I}d},$$

où $\widetilde{\ell}$ représente la constante d'échelle et $\mathtt{I} = \int_{\mathcal{X}} f(x) \, |[\ln f]'(x)|^2 \, dx$. Le terme $\mathtt{I}$ est une mesure de la rugosité de la densité f.

En étudiant le comportement limite de la chaîne de Markov lorsque $d \to \infty$, il est généralement possible de montrer que les échantillonneurs par marche aléatoire impliquant plusieurs propositions convergent faiblement vers un processus de diffusion Langevin (l'article [2] donne plus de détails concernant ce point ; l'article [15] est le premier résultat théorique de ce genre à avoir été publié dans la littérature).

La mesure de vitesse du processus de diffusion (le seul terme qui est fonction de $\widetilde{\ell}$, et donc de la variance instrumentale) est la seule mesure d'efficacité disponible dans un contexte asymptotique ; toutes les autres mesures d'efficacité possibles en dimension finie convergent vers la vitesse de diffusion lorsque $d \to \infty$. Pour optimiser la performance de l'algorithme, il suffit donc d'opter pour la variance instrumentale qui maximise la vitesse de diffusion. Bien entendu, ceci nous mène à une variance instrumentale asymptotiquement optimale. Cependant, ce type de résultat donne généralement de bons résultats dans des contextes de dimension finie, ce qui nous permet de les utiliser plus largement.

Dans ce contexte, nous allons reproduire un tableau présenté dans [2] qui illustre l'efficacité de l'algorithme MTM. Dans leurs exemples, [2] ont utilisé des valeurs pour le nombre de candidats K allant de 1 à 5. Il a été établi à partir de ces valeurs que l'efficacité théorique de l'algorithme MTM par marche aléatoire avec des propositions indépendantes augmente avec K. Le gain le plus important en terme d'efficacité théorique est obtenu lors du passage de $K = 1$ (efficacité proportionnelle relative de 1.32) à $K = 2$ (efficacité proportionnelle de 2.24). Cela représente une amélioration de 70% par rapport à l'algorithme Metropolis

par marche aléatoire.

TABLE 2.1. L'efficacité de l'algorithme Metropolis à essais multiples représentée par la constante d'échelle asymptotiquement optimale ($\tilde{\ell}^*$), la vitesse de diffusion optimale (λ^*) et le taux d'acceptation asymptotiquement optimal (a^*).

K	1	2	3	4	5
$\tilde{\ell}^*$	2.38	2.64	2.82	2.99	3.12
λ^*	1.32	2.24	2.94	3.51	4.00
a^*	0.23	0.32	0.37	0.39	0.41

De façon intuitive, l'efficacité de l'échantillonneur MTM repose, d'une part, sur le calibrage du paramètre d'échelonnage de la distribution instrumentale ainsi que sur le nombre d'essais K et, d'autre part, sur la forme de la densité cible.

2.1.3. Formulation générale des algorithmes réversibles

Dans le but d'obtenir une chaîne de Markov réversible par rapport à la densité cible $\pi(\cdot)$, il suffit que la fonction de transition actuelle de x à y, pour $x \neq y$, soit de la forme suivante :

$$P(x, y) = \pi(y)\delta(x; y), \tag{2.1.4}$$

où $\delta(x; y)$ est une fonction non négative et symétrique en x et y. Il est possible de vérifier que la fonction de transition donnée en (2.1.4) satisfait la condition d'équilibre détaillée.

La fonction de transition $P(x, y)$ doit cependant satisfaire la contrainte suivante

$$\int_{y \neq x} P(x, y)dy \leq 1 \ \ \forall x.$$

On peut donc conclure que le choix de la fonction $\delta(x; y)$ ne peut être complètement arbitraire, contrairement au choix de la fonction $\lambda(x; y)$ dans l'échantillonneur Metropolis à essais multiples, qui est presque arbitraire.

De toute évidence, dans un algorithme Metropolis-Hastings, la fonction de transition $P(x, y)$ est de la forme suivante :

$$P(x, y) = \pi(y)\delta_{MH}(x; y),$$

où

$$\delta_{MH}(x; y) = \min\left\{\frac{q(x; y)}{\pi(y)}, \frac{q(y; x)}{\pi(x)}\right\}, \tag{2.1.5}$$

qui est évidemment symétrique en x et y.

En revanche, la fonction de transition asymptotique $P(x, y)$ dans un algorithme Metropolis à essais multiples (lorsque $K \to \infty$) correspond à la forme suivante :

$$P(x; y) = \pi(y)\delta_{MTM}(x; y),$$

où la fonction $\delta_{MTM}(x; y)$ est obtenue en généralisant l'équation (2.1.5) pour K essais. Tel que mentionné dans [9] nous avons donc,

$$\begin{aligned}\delta_{MTM}(x; y) &= q(x; y)q(y; x)\lambda(x; y) \\ &\quad \times \min\left\{\frac{1}{E^{\pi}[q(x; Y)q(Y; x)\lambda(x; Y)]}, \frac{1}{E^{\pi}[q(X; y)q(y; X)\lambda(X; y)]}\right\}.\end{aligned}$$

Dans un algorithme MTM, nous réalisons donc que la forme de $\lambda(x; y)$ aura possiblement un impact sur la performance de l'algorithme MTM. Dans ce travail, nous allons nous concentrer sur la fonction :

$$\lambda(x; y) = \left(\frac{2}{q(x; y) + q(y; x)}\right)$$

pour laquelle $w(y_i; x) = \pi(y_i)$. Cette forme de $\lambda(x; y)$ donne lieu à un algorithme MTM intuitif : le candidat dans une itération donnée est choisi parmi les K propositions générées, avec une probabilité proportionnelle à la densité cible évaluée à chacune des K propositions.

2.2. Exemples pratiques

Cette section présente des exemples pratiques dans le but de comparer des algorithmes Metropolis à essais multiples pour différentes valeurs de K. Pour ce faire, nous avons programmé les algorithmes à l'aide du logiciel statistique `R` (programmes en annexe).

Nous présenterons quatre exemples pour l'algorithme MTM avec une innovation normale en utilisant le paramètre d'échelonnage de la densité instrumentale le plus approprié selon les résultats obtenus à la section 2.1.2, soit $\sigma = 2.4$. Pour chaque exemple, nous utiliserons un nombre de candidats différent.

Les résultats obtenus sont présentés dans les figures qui suivent. Chaque figure contient trois graphes : celui du haut représente l'extrait (ou la trace) de la chaîne de Markov ; celui du centre illustre l'histogramme des échantillons et celui du bas présente la fonction d'autocorrélation (acf).

La figure 2.1 représente un algorithme MTM pour une densité cible normale, $N(0, 1)$ en générant 50 000 itérations à partir d'une densité intrumentale $N(x, 2.4)$ en utilisant deux candidats ($K = 2$). Les figures 2.2, 2.3 et 2.4 ont été obtenues à l'aide du même algorithme, mais en utilisant $K = 5$, $K = 10$ et $K = 30$, respectivement. En comparant les résultats obtenus, on observe clairement que les chaînes générées se déplacent rapidement (graphique de la trace). De même, la fonction d'autocovariance s'estompe rapidement, ce qui laisser présager une exploration efficace de l'espace. Autrement dit, si la chaîne est stationnaire, alors le fait qu'il y ait peu de corrélation entre deux points éloignés dans le temps indique que la chaîne explore l'espace efficacement. S'il y a trop de corrélation entre 2 points, alors l'exploration de l'espace est plus lente, c'est-à-dire la chaîne stagne. Les quatre exemples semblent indiquer (bien que ce soit assez subtil, voir la fonction d'autocorrélation) que plus le nombre de candidats est grand, plus la chaîne de Markov générée est proche de la densité cible et converge rapidement. Notons que ce phénomène serait plus facilement observable dans le cas d'une densité cible complexe, présentant certaines difficultés de convergence lors de l'utilisation d'un algorithme Metropolis-Hastings standard.

Les taux d'acceptation obtenus dans chaque cas sont présentés dans le tableau qui suit. Ces taux augmentent avec K puisque lorsqu'il y a plusieurs candidats dans une itération, il devient plus probable de sélectionner un candidat qui sera par la suite accepté.

TABLE 2.2. Les taux d'acceptation obtenus pour les exemples de l'algorithme Metropolis à essais multiples.

K	2	5	10	30
Taux d'acceptation	0.60	0.75	0.82	0.89

Il est important de noter que dans le cas où $K = 1$ l'algorithme MTM se réduit à un algorithme Metropolis-Hastings dont les exemples étaient présentés au chapitre précédent.

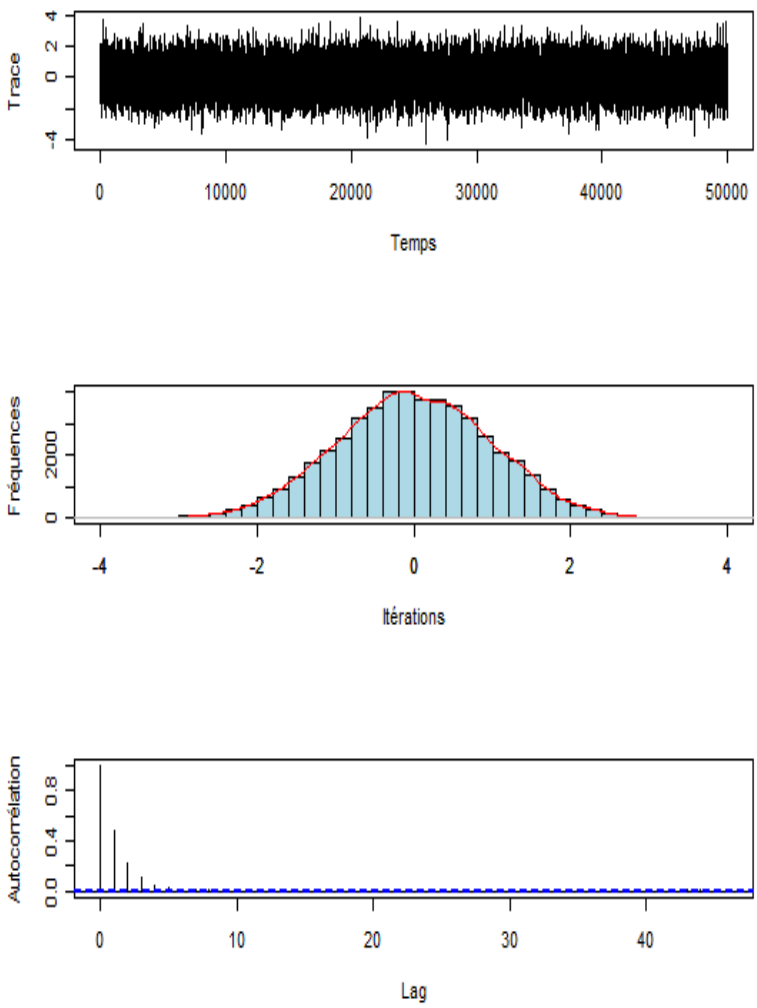

FIGURE 2.1. Algorithme MTM pour une loi instrumentale $N(x, 2.4)$ et une densité cible $N(0, 1)$ avec $K = 2$.

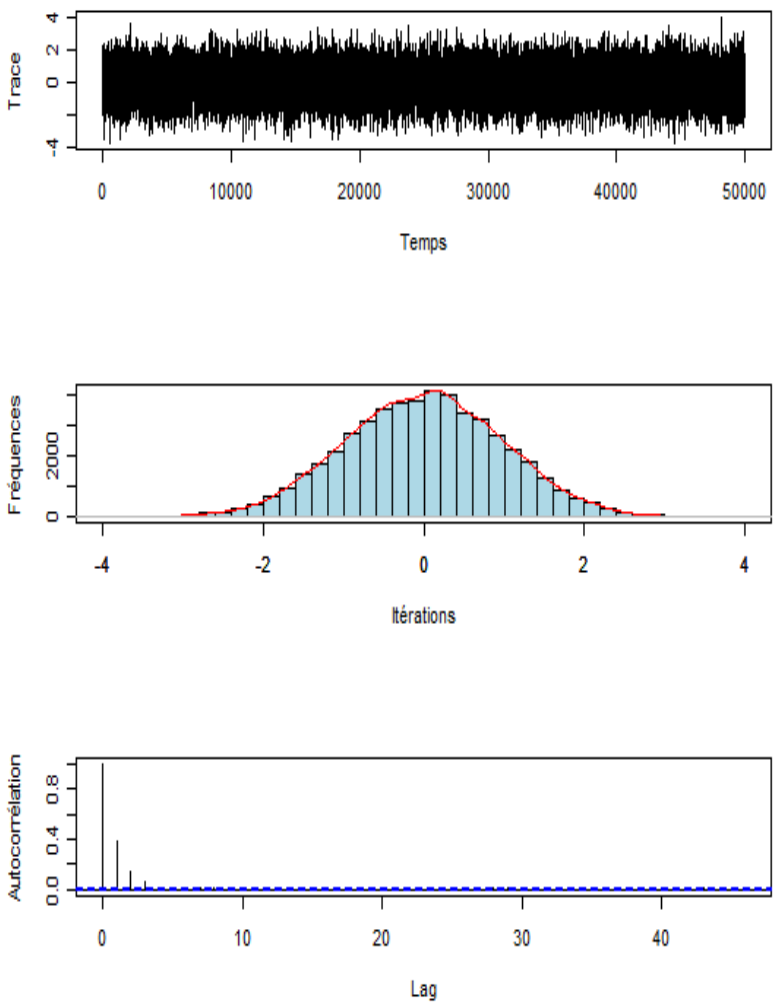

FIGURE 2.2. Algorithme MTM pour une loi instrumentale $N(x, 2.4)$ et une densité cible $N(0, 1)$ avec $K = 5$.

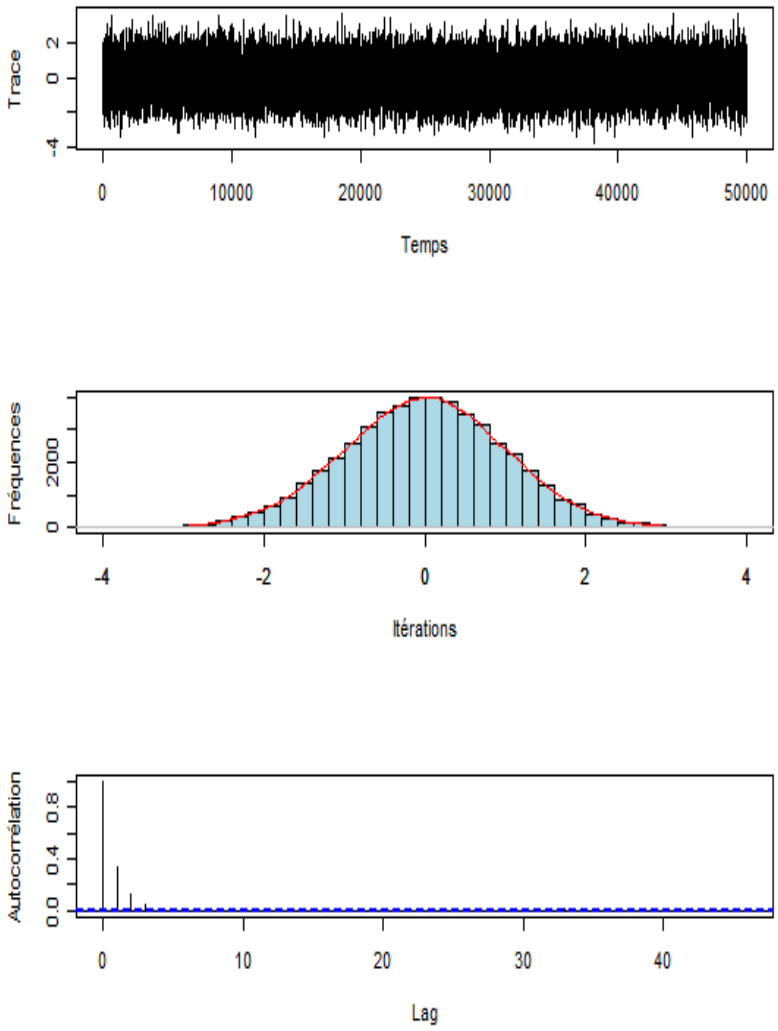

FIGURE 2.3. Algorithme MTM pour une loi instrumentale $N(x, 2.4)$ et une densité cible $N(0, 1)$ avec $K = 10$.

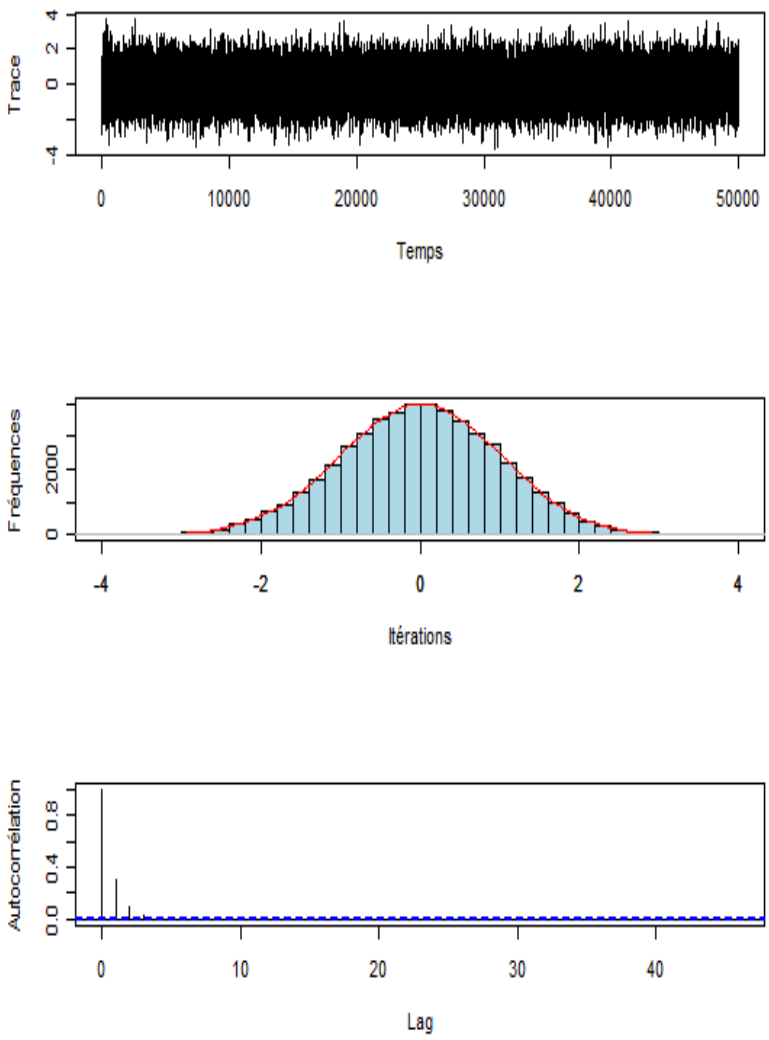

FIGURE 2.4. Algorithme MTM pour une loi instrumentale $N(x, 2.4)$ et une densité cible $N(0, 1)$ avec $K = 30$.

Chapitre 3

ALGORITHME MTM : RECYCLAGE DES CANDIDATS

L'algorithme MTM présenté au chapitre précédent a été repensé par [1] ; leur approche permet de recycler des propositions déjà générées. L'idée est de définir la méthode MTM dans le cadre de l'algorithme Metropolis-Hastings standard, mais sur un espace étendu. À cette fin, une nouvelle densité cible qui est une fonction de toutes les propositions générées dans une itération donnée a été identifiée, un aspect crucial permettant de réexprimer l'algorithme MTM standard sur un espace étendu.

Cette nouvelle version a l'avantage d'être moins coûteuse par rapport à l'algorithme MTM standard, du point de vue de l'implémentation. En effet, puisque les propositions déjà générées sont recyclées, il n'est donc pas nécessaire de générer autant de valeurs dans une itération et d'évaluer la densité cible à ces points. En outre, cette nouvelle formulation mène à diverses généralisations tel que mentionné dans [1].

Grâce à cette reformulation sous la forme d'un algorithme Metropolis-Hastings standard, il devient clair que de nouveaux schémas sont accessibles afin de mettre à jour les états de la chaîne de Markov. Ceux-ci permettent, notamment, de réutiliser en tout ou en partie les propositions générées antérieurement.

Dans ce chapitre, nous allons présenter la méthode principale, que nous appellerons l'algorithme MTM revisité. La première section réintroduira l'algorithme MTM dans ce nouveau contexte d'espace étendu ; à la deuxième section, nous comparerons ce nouvel algorithme avec l'algorithme MTM standard. Enfin, à la dernière section, nous présenterons des exemples pratiques afin d'illustrer la performance de l'algorithme MTM revisité.

3.1. L'algorithme MTM revisité

Comme nous l'avons mentionné au chapitre précédent, l'algorithme MTM peut être vu comme une variation de l'algorithme Metropolis-Hastings, nécessitant la génération de variables auxiliaires et visant à échantillonner d'une densité cible $\pi(\cdot)$. Nous répétons maintenant l'approche de [1] pour montrer que :

(1) L'échantillonneur MTM peut être interprété comme un algorithme Metropolis-Hastings standard qui vise à prélever un échantillon d'une nouvelle distribution cible définie sur un espace étendu.

(2) La nouvelle distribution cible n'admet pas $\pi(\cdot)$ comme une densité marginale, par contre l'une de ses densités conditionnelles correspond à $\pi(\cdot)$.

(3) Selon cette nouvelle formulation, il n'est pas nécessaire de générer deux échantillons à chaque itération (c'est-à-dire un échantillon de propositions et un autre de variables auxiliaires).

3.1.1. Configuration du MTM revisité

De manière similaire à [1], par souci de clarité, nous nous concentrons sur le cas des distributions cibles discrètes. Nous supposons que toutes les fonctions de probabilité considérées admettent une densité par rapport à une mesure dominante commune. Les généralisations au cas continu sont directes.

Considérons la distribution cible suivante, définie sur l'espace étendu $\{1, 2, \ldots, K\} \times \mathbb{Z}^K$ pour $K \geq 1$

$$\begin{aligned} \widetilde{\pi}^K(k, x_1, x_2, \ldots, x_K) &= \frac{1}{K}\pi(x_k) \prod_{i=1, i\neq k}^{K} q(x_k; x_i) \\ &= \frac{1}{K}\frac{\pi(x_k)}{q(x_k; x_k)} \prod_{i=1}^{K} q(x_k; x_i), \end{aligned} \tag{3.1.1}$$

avec une convention évidente (c'est-à-dire 0) dans l'équation (3.1.1) lorsque le ratio n'est pas défini.

On définit $\pi(\cdot)$ comme étant la fonction de probabilité cible qui nous intéresse réellement et $q(x_k; x_i)$ comme étant une fonction de probabilité instrumentale utilisée pour générer des candidats pour $\pi(\cdot)$.
La densité $\pi(\cdot)$ n'est pas l'une des distributions marginales de $\widetilde{\pi}^K(k, x_1, x_2, ..., x_K)$. Par contre, on remarque que

$$\tilde{\pi}^K(x_k|k) = \pi(x_k), \quad \forall k \in \{1, 2, \ldots, K\}$$

et donc $\pi(\cdot)$ peut être exprimée comme étant une distribution conditionnelle de $\tilde{\pi}^K(\cdot)$. Comme nous le verrons, cette relation représente la propriété fondamentale exploitée par l'algorithme MTM afin d'échantillonner directement de $\pi(\cdot)$.

Afin d'échantillonner de $\widetilde{\pi}^K$ à l'aide d'un échantillonneur Metropolis-Hastings standard, nous devons faire le choix d'une densité instrumentale quelconque. Dans le but de définir cette densité instrumentale, rappelons-nous le concept de fonction de poids introduit au chapitre 2.
Soit la fonction

$$w(x; y) : \mathbb{Z}^2 \to [0, \infty),$$

qui peut être considérée comme une fonction de poids positive. Dans l'équation (2.1.1), on suggère de considérer la fonction de poids suivante

$$w(x; y) = \pi(x)q(x; y)\lambda(x; y),$$

pour $(x; y) \in \mathbb{Z}^2$ et pour toute fonction positive et symétrique $\lambda : \mathbb{Z}^2 \to [0, \infty)$. On note que, dans le cas particulier où

$$\lambda(y; x) = (q(y; x)q(x; y))^{-1},$$

la fonction de poids prend la forme de l'échantillonnage d'importance et est donnée par :

$$w(y; x) = \frac{\pi(y)}{q(x; y)}.$$

Rappelons que dans le cas qui nous intéresse, où

$$\lambda(y; x) = \frac{2}{q(y; x) + q(x; y)}$$

et $q(x, y)$ est symétrique en x et y, alors la fonction de poids est proportionnelle à la fonction de probabilité cible $w(y; x) = \pi(y)$.

Afin de mettre à jour les composantes de $\widetilde{\pi}^K(\cdot)$ dans l'équation (3.1.1), nous appliquons d'abord une transformation déterministe aux variables aléatoires impliquées dans l'algorithme Metropolis-Hastings. La mise-à-jour considérée dans ce contexte consiste à fixer une paire de composantes (x_k, x_l) du modèle étendu

tout en actualisant les autres composantes $x_i, i \neq k, l$. Les composantes fixées échangent alors leurs rôles ; spécifiquement, pour une paire $(k, l) \in \{1, 2, ..., K\}$, nous cherchons à mettre à jour l'état actuel,

$$(k, \mathbf{x}) = (k, x_1, x_2, \ldots, x_k = y_k, \ldots, x_l = y_l, \ldots, x_K).$$

à l'aide de l'état proposé :

$$(l, \mathbf{y}) = (l, y_1, y_2, \ldots, y_k = x_k, \ldots, y_l = x_l, \ldots, y_K).$$

Étant donné l'état actuel $(k, \mathbf{x})$, cette mise à jour ne nécessite d'échantillonner que $(l, \mathbf{y}_{-k,l})$. Afin de générer $(l, \mathbf{y}_{-k,l})$, la distribution instrumentale suivante est adoptée :

$$q(l, \mathbf{y}_{-k,l} | k, \mathbf{x}) = \frac{w(x_l; x_k)}{\sum_{m=1}^{K} w(x_m; x_k)} \prod_{m=1, m\neq k,l}^{K} q(y_l = x_l; y_m).$$

Une itération de l'algorithme MTM revisité, tel que présenté par [1], est détaillée comme suit.

Algorithme 3.1.1 (Algorithme Metropolis à essais multiples revisité)**.**
Étant donné $(k, \boldsymbol{x})$, l'état de la chaîne de Markov au temps n,

(1) Échantillonner $l \mid (k, \boldsymbol{x}) \sim (\widetilde{w}(x_1; x_k), \widetilde{w}(x_2; x_k), \ldots, \widetilde{w}(x_K; x_k))$, où

$$\widetilde{w}(x_i; x_k) = \frac{w(x_i; x_k)}{\sum_{m=1}^{K} w(x_m; x_k)}.$$

(2) Poser $y_k = x_k$, $y_l = x_l$ et proposer un échantillon de la distribution conditionnelle :

$$\boldsymbol{y}_{-k,l} \mid (l, y_k, y_l) \sim \prod_{m=1, m\neq k,l}^{K} q(y_l = x_l; y_m).$$

(3) Calculer la probabilité d'acceptation

$$\alpha\left((k, \boldsymbol{x}); (l, \boldsymbol{y})\right) = \min\left(1, \frac{\pi(y_l) q(y_l; y_k) \frac{w(y_k; y_l)}{\sum_{m=1}^{K} w(y_m; y_l)}}{\pi(x_k) q(x_k; x_l) \frac{w(x_l; x_k)}{\sum_{m=1}^{K} w(x_m; x_k)}}\right). \tag{3.1.2}$$

(4) L'état de la chaîne de Markov au temps $n+1$ devient

$$\begin{cases} (l, \boldsymbol{y}) & \text{avec probabilité } \alpha\left((k, \boldsymbol{x}); (l, \boldsymbol{y})\right); \\ (k, \boldsymbol{x}) & \text{avec probabilité } 1 - \alpha\left((k, \boldsymbol{x}); (l, \boldsymbol{y})\right). \end{cases}$$

En se référant à la forme générale de la fonction de poids fournie dans l'équation (2.1.1), le ratio d'acceptation spécifié dans l'équation (3.1.2) prend la forme simplifiée suivante :

$$\alpha\left((k,\mathbf{x});(l,\mathbf{y})\right)=\min\left(1,\frac{\sum_{m=1}^{K}w(x_m;x_k)}{\sum_{m=1}^{K}w(y_m;y_l)}\right). \tag{3.1.3}$$

Il est intéressant de remarquer que ce ratio simplifié est en fait le ratio d'acceptation de l'algorithme Metropolis à essais multiples (MTM) décrit au chapitre 2.

Afin d'obtenir un échantillon de $\pi(\cdot)$, il suffit de considérer le processus $\{X_k[n]; n \geq 0\}$, c'est-à-dire de recueillir les observations pour x_k après une certaine période de chauffe.

3.2. Comparaison : MTM revisité vs MTM standard

Nous comparons maintenant les algorithmes MTM standard et MTM revisité, en les exprimant tous deux dans un contexte d'échantillonneur de type Metropolis-Hastings sur un espace étendu. Pour ce faire, nous cherchons à obtenir un échantillon de la densité cible étendue fournie en (3.1.1). Dans ce contexte, l'algorithme MTM revisité a déjà été décrit à la section précédente. Afin d'obtenir un échantillonneur équivalent au MTM standard (c'est-à-dire un algorithme qui ne recycle pas les candidats antérieurs), il suffit d'ajouter une étape au MTM revisité, qui consiste à rafraîchir l'état actuel $(k,\mathbf{x})$ à l'aide d'un échantillonneur de Gibbs.

Une itération de l'algorithme MTM standard est détaillée comme suit :

Algorithme 3.2.1 (Algorithme MTM standard)**.**
Étant donné $(k,\boldsymbol{x})$, l'état de la chaîne de Markov au temps n,

(1) Mettre à jour l'état actuel à l'aide de l'échantillonneur de Gibbs, c'est-à-dire échantillonner $\boldsymbol{x}_{-k}^{'} \mid (k,x_k) \sim \prod_{m=1,m\neq k}^{K} q(x_k;x_m)$ et poser $x_k^{'}=x_k$.

(2) Échantillonner $l \mid (k,\boldsymbol{x}^{'}) \sim \left(\tilde{w}(x_1^{'};x_k^{'}),\tilde{w}(x_2^{'};x_k^{'}),\ldots,\tilde{w}(x_K^{'};x_k^{'})\right)$ où

$$\tilde{w}(x_i^{'};x_k^{'})=\frac{w(x_i^{'};x_k^{'})}{\sum_{m=1}^{K}w(x_m^{'};x_k^{'})}.$$

(3) Poser $y_k=x_k^{'}$, $y_l=x_l^{'}$ et proposer un échantillon de la distribution conditionnelle :

$$\boldsymbol{y}_{-k,l} \mid (l,y_k,y_l) \sim \prod_{m=1,m\neq k,l}^{K} q(y_l=x_l^{'};y_m).$$

(4) Calculer la probabilité d'acceptation

$$\alpha\left((k,\boldsymbol{x});(l,\boldsymbol{y})\right)=\min\left(1,\frac{\pi(y_l)q(y_l;y_k)\frac{w(y_k;y_l)}{\sum_{m=1}^{K}w(y_m;y_l)}}{\pi(x_k^{'})q(x_k^{'};x_l^{'})\frac{w(x_l^{'};x_k^{'})}{\sum_{m=1}^{K}w(x_m^{'};x_k^{'})}}\right).$$

(5) L'état de la chaîne de Markov au temps $n+1$ devient

$$\begin{cases}(l,\boldsymbol{y}) & \textit{avec probabilité } \alpha\left((k,\boldsymbol{x}^{'});(l,\boldsymbol{y})\right)\\ (k,\boldsymbol{x}^{'}) & \textit{avec probabilité } 1-\alpha\left((k,\boldsymbol{x}^{'});(l,\boldsymbol{y})\right).\end{cases}$$

En comparant les deux échantillonneurs, c'est-à-dire l'algorithme MTM revisité (section 3.1.1) et l'algorithme Metropolis à essais multiples classique (chapitre 2), on remarque que la seule différence réside dans l'interprétation des rôles respectifs de $\mathbf{x}$ et $\mathbf{y}$.

Considérons la probabilité d'acceptation fournie en (3.1.3). Dans l'algorithme MTM classique, $\mathbf{y}$ est interprété comme une variable auxiliaire de l'échantillon, nécessaire pour préserver la réversibilité de la chaîne, alors que $\mathbf{x}$ est interprété comme un candidat pour le prochain état de la chaîne. Dans l'algorithme MTM revisité, $\mathbf{x}$ représente plutôt l'état actuel de la chaîne de Markov, alors que $\mathbf{y}$ représente le candidat proposé. La différence principale entre les deux approches est donc que l'algorithme MTM standard rafraîchit $\mathbf{x}_{-k}$ avant de sélectionner la composante active l, contrairement à l'algorithme MTM revisité. Ceci explique que le MTM standard soit plus coûteux à implémenter. Les différences de performance entre les deux approches seront discutées plus en détail dans le prochain chapitre.

3.3. Exemples pratiques

Cette section présentera des exemples pratiques pour l'algorithme Metropolis à essais multiples revisité avec différents paramètres. Pour ce faire, nous avons programmé les algorithmes à l'aide du logiciel statistique **R** (programmes en annexe).

En se fiant aux résultats obtenus dans le deuxième chapitre, nous présenterons quatre exemples illustrant la performance de l'algorithme MTM revisité. Nous échantillonnerons d'une densité cible normale, en générant des candidats à partir d'une densité instrumentale normale. Nous utiliserons le paramètre d'échelonnage de la densité instrumentale le plus approprié (soit $\sigma = 2.4$ tel que mentionné au chapitre 2) ; à chaque exemple, le nombre de candidats (K) sera différent.

Les résultats obtenus sont présentés dans les figures qui suivent. Chaque figure contient trois graphes : celui du haut représente l'extrait (ou la trace) de la chaîne

de Markov ; celui du centre illustre l'histogramme des échantillons et celui du bas présente la fonction d'autocorrélation (acf).

La figure 3.1 a été obtenue à l'aide d'un algorithme MTM revisité pour une densité cible normale, $N(0,1)$ en générant 50 000 itérations à partir d'une densité intrumentale $N(x_l, 2.4)$ et en se basant sur deux candidats ($K = 2$). Les figures 3.2, 3.3 et 3.4 ont été obtenues de façon similaire en utilisant $K = 5$, $K = 10$ et $K = 30$, respectivement. En comparant les résultats obtenus, on observe clairement que les chaînes résultantes se déplacent de plus en plus rapidement à mesure que K augmente. De même, la fonction d'autocovariance s'estompe plus rapidement lorsque K croît, ce qui laisse présager d'une meilleure convergence vers $\pi(\cdot)$. Par conséquent, plus le nombre de candidats est grand, plus la chaîne de Markov obtenue semble efficace pour échantillonner de la densité cible.

En comparant ces résultats avec ceux obtenus avec l'algorithme MTM standard (dans le chapitre précédent), on observe que bien que l'algorithme MTM revisité mène à un processus d'échantillonnage valide, ce dernier est moins efficace que l'algorithme MTM standard (pour un même nombre de candidats). Par contre, l'exécution du MTM standard ne se basant pas sur des candidats recyclés, celle-ci est plus laborieuse que pour le MTM revisité.

Les taux d'acceptation obtenus dans chaque cas sont présentés dans le tableau qui suit. Le taux est élevé lorsque $K = 2$; ceci est dû au fait que l'état présent figure parmi les candidats, ce qui a pour conséquence de faire augmenter la proportion d'états acceptés. Lorsque K croît, l'impact de cette particularité s'estompe et l'algorithme se comporte comme l'algorithme MTM standard. En effet, en présence de plusieurs candidats, le fait que l'état présent figure dans le groupe des candidats n'a que peu d'impact sur la chaîne générée.

TABLE 3.1. Les taux d'acceptation obtenus pour les exemples de l'algorithme Metropolis à essais multiples revisité.

K	2	5	10	30
Taux d'acceptation	0.87	0.81	0.84	0.89

Dans le prochain chapitre, nous identifierons les raisons pour lesquelles la convergence s'effectue plus lentement lors du recyclage de candidats. Nous tenterons également d'améliorer la performance de l'algorithme MTM revisité. L'algorithme que nous proposerons sera plus performant que le MTM revisité pour de petites valeurs de K, mais moins performant lorsque K est grand.

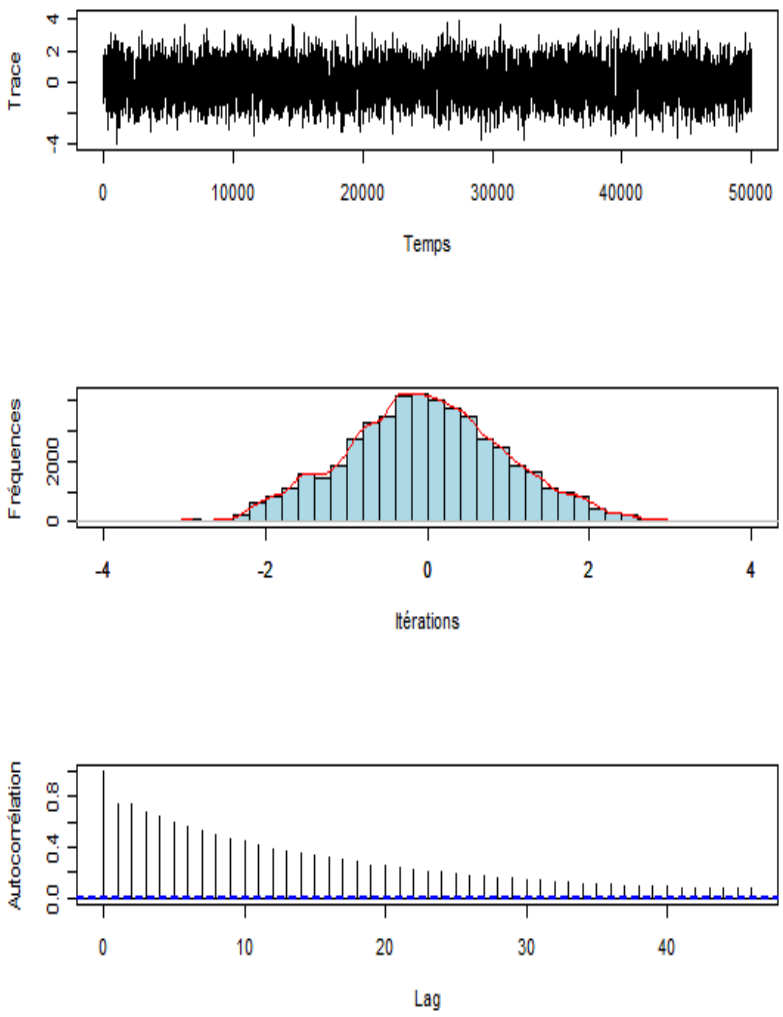

FIGURE 3.1. Algorithme MTM revisité pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 2$.

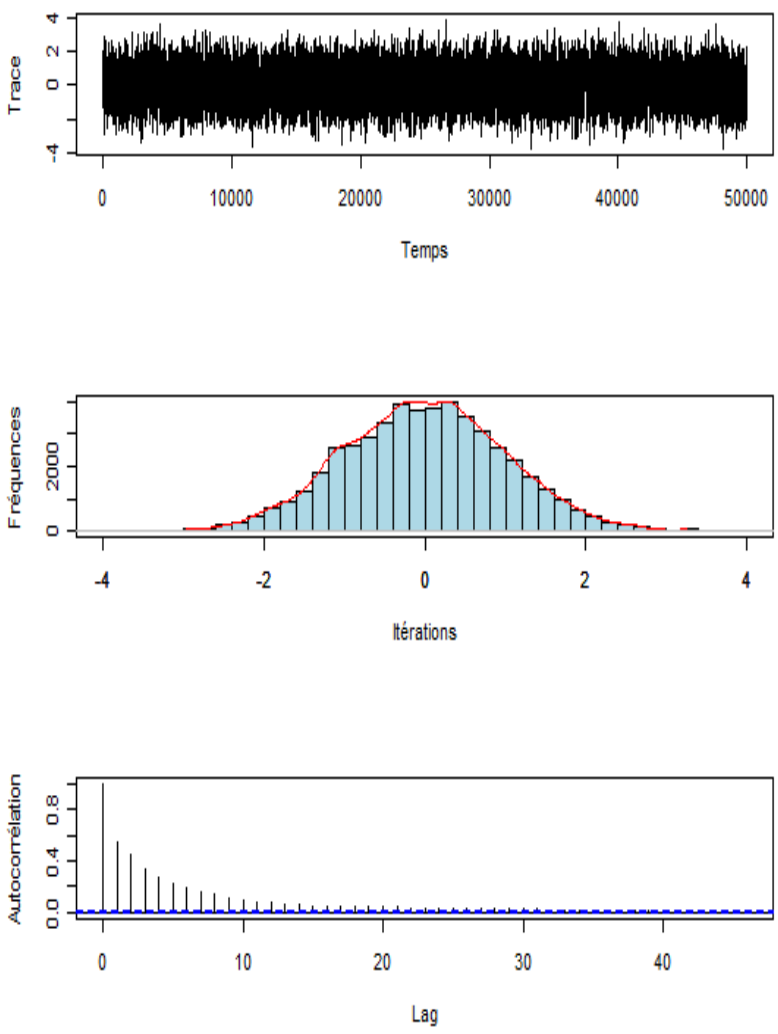

FIGURE 3.2. Algorithme MTM revisité pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 5$.

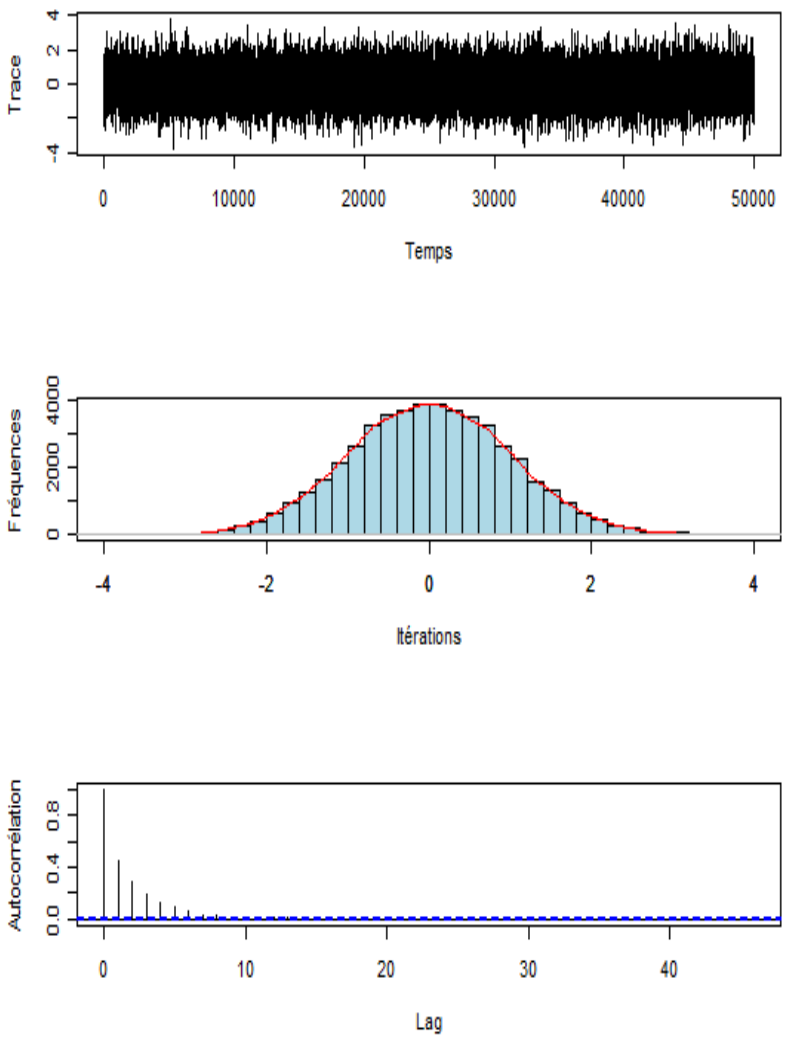

FIGURE 3.3. Algorithme MTM revisité pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 10$.

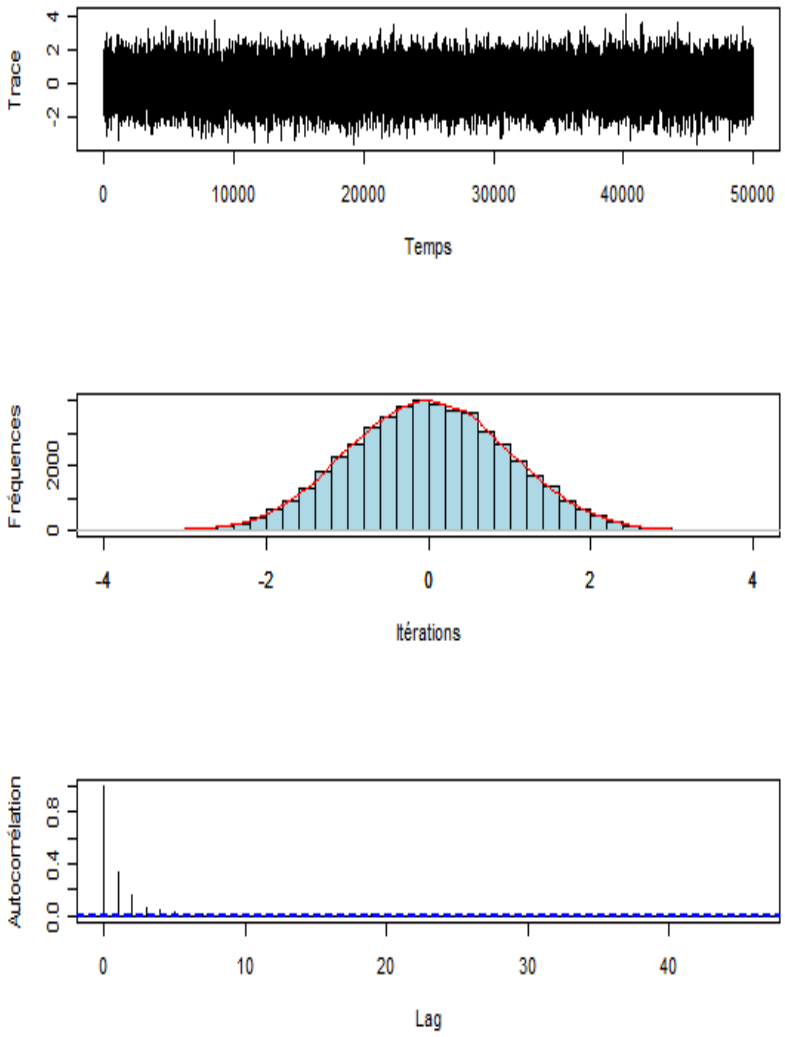

FIGURE 3.4. Algorithme MTM revisité pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 30$.

Chapitre 4

AMÉLIORATION DE L'ALGORITHME MTM REVISITÉ

Nous consacrerons ce chapitre à l'analyse de l'algorithme MTM revisité par [1]. Cette analyse consistera à étudier la performance de cette méthode, et ensuite à proposer une amélioration.

Comme nous l'avons déjà présenté au chapitre précédent, l'algorithme MTM revisité est un algorithme inspiré de l'algorithme MTM standard, introduit dans la littérature statistique par [9], qui est lui-même une variation de l'algorithme Metropolis-Hastings. Il est évident que l'algorithme MTM standard se réduit à l'algorithme Metropolis-Hastings dans le cas où $K = 1$ c'est-à-dire quand le nombre de candidats générés dans une itération donnée se réduit à un seul candidat. Nous aimerions maintenant vérifier si l'algorithme MTM revisité se réduit également à l'algorithme Metropolis-Hastings lorsque nous utilisons un seul candidat.

La première section de ce chapitre consistera à analyser l'algorithme MTM revisité en utilisant un seul candidat ($K = 1$). La deuxième section sera consacrée à une analyse similaire, mais cette fois-ci nous utiliserons deux candidats ($K = 2$). Par la suite, nous proposerons une mise-à-jour nécessaire afin d'assurer la réduction à l'algorithme Metropolis-Hastings lorsque $K = 1$ et nous développerons un nouvel algorithme inspiré de l'algorithme MTM revisité. Ce nouvel algorithme démontre une performance améliorée lorsque de petites valeurs de K sont utilisées. Finalement, nous conclurons avec quelques exemples pratiques pour illustrer la performance de cette modification.

4.1. Analyse de l'algorithme MTM revisité

L'approche du MTM revisité a été basée sur l'idée de redéfinir la méthode MTM standard dans le cadre de l'algorithme Metropolis-Hastings. Il serait donc

approprié que cette nouvelle version se réduise, tout comme le MTM standard, à l'algorithme Metropolis-Hastings de base.

4.1.1. MTM revisité lorsque le nombre de candidats $K = 1$

Reformulons maintenant l'algorithme MTM revisité lorsque le nombre de candidats est $K = 1$.

Algorithme 4.1.1 (Algorithme MTM revisité avec $K = 1$)**.**
Étant donné $(k, \boldsymbol{x}) = (1, x_1)$, l'état de la chaîne de Markov au temps n,

(1) Échantillonner $l \mid (k, \boldsymbol{x})$; nécessairement, $l = k = 1$.

(2) Poser $y_k = x_k$, $y_l = x_l$; nécessairement, $y_k = y_l = y_1 = x_1$. Normalement nous devrions générer un échantillon $\boldsymbol{y}_{-k,l}$, mais dans ce cas-ci il n'y a pas d'autres candidats à générer.

(3) Calculer la probabilité d'acceptation

$$\alpha\left((k, \boldsymbol{x}); (l, \boldsymbol{y})\right) = \alpha\left((1, x_1); (1, y_1)\right). \tag{4.1.1}$$

Puisque $y_1 = x_1$, alors

$$\alpha\left((1, x_1); (1, y_1)\right) = 1.$$

(4) L'état de la chaîne de Markov au temps $n + 1$ demeure donc $(1, x_1)$.

Il est maintenant clair qu'en n'utilisant qu'un seul candidat ($K = 1$) dans l'algorithme MTM revisité, la chaîne de Markov résultante ne dépendra alors que de l'état initial. Autrement dit, à chaque itération, l'état initial est répété indéfiniment, ce qui signifie que l'algorithme ne convergera tout simplement pas. Par contre, dans le cas de la méthode Metropolis-Hastings habituelle, la probabilité d'acceptation dépend d'un nouvel état proposé et l'algorithme roulera jusqu'à la convergence vers la distribution cible.

Par conséquent, il est clair que l'algorithme MTM revisité ne se réduit pas à l'algorithme Metropolis-Hastings lorsque $K = 1$.

4.1.2. MTM revisité lorsque le nombre de candidats $K = 2$

Supposons maintenant que le nombre de candidats proposés est égal à $K = 2$. Dans ce cas, nous aurons deux possibilités :

(1) $k \neq l$

La transition de l'algorithme MTM revisité génère un échantillon de candidats $\mathbf{y}_{-k,l}$. Dans le cas où $k = 1$ et $l = 2$ (ou $k = 2$ et $l = 1$), il n'y aura donc aucun candidat à générer.
La probabilité d'acceptation est donnée par

$$\begin{aligned}\alpha((k,\mathbf{x});(l,\mathbf{y})) &= \alpha((1,x_1,x_2);(2,y_1,y_2)) \\ &= \min\left(1, \frac{w(x_1,x_1)+w(x_2,x_1)}{w(y_1,y_2)+w(y_2,y_2)}\right).\end{aligned}$$

En utilisant la fonction de poids définie en (2.1.1), la fonction

$$\lambda(x;y) = \frac{2}{q(x;y)+q(y;x)}$$

et une densité instrumentale symétrique en x et y, alors $w(x,y) = \pi(x)$ et nous obtenons

$$\alpha((k,\mathbf{x});(l,\mathbf{y})) = \min\left(1, \frac{\pi(x_1)+\pi(x_2)}{\pi(y_1)+\pi(y_2)}\right).$$

Dans le présent cas, puisque $x_1 = y_1$ et $x_2 = y_2$, la probabilité d'acceptation devient égale à 1. Il est cependant important de mentionner que, même si aucun candidat n'a été généré, la chaîne ne restera pas immobile. En effet, les x resteront les mêmes, mais la composante active passera de x_1 à x_2.

(2) $k = l$

La transition de l'algorithme MTM revisité génère un échantillon $\mathbf{y}_{-k,l}$. Dans le cas, par exemple, où $k = l = 1$, un nouveau candidat y_2 sera généré, et nous poserons $y_1 = x_1$. En utilisant les mêmes hypothèses que dans le cas (1) pour la fonction de poids et la densité instrumentale, la probabilité d'acceptation prendra la forme suivante :

$$\begin{aligned}\alpha((k,\mathbf{x});(l,\mathbf{y})) &= \alpha((1,x_1,x_2);(1,y_1,y_2)) \\ &= \min\left(1, \frac{\pi(x_1)+\pi(x_2)}{\pi(x_1)+\pi(y_2)}\right).\end{aligned}$$

Dans ce cas, la probabilité d'acceptation n'est pas nécessairement égale à 1. Si l'état proposé est accepté, les x seront modifiés. Cependant, il est important de

spécifier qu'indépendamment de l'acceptation du nouvel état de la chaîne, la composante active restera la même puisque $k = l$.

Rappelons que du point de vue de l'algorithme MTM, c'est-à-dire pour échantillonner de la densité $\pi(\cdot)$, c'est la trajectoire de la composante active qui importe (puisque $\widetilde{\pi}^K(x_k \mid k) = \pi(x_k)$ tel que mentionné à la section 3.1.1).

D'après cette analyse de l'algorithme MTM revisité lorsque le nombre de candidats $K = 2$, nous observons clairement que lorsque $k \neq l$ alors le groupe de candidats n'est pas rafraîchi. Lorsque $k = l$, l'un des deux candidats est rafraîchi, mais la composante active ne bouge pas.

Le fait de recycler des candidats semble, du moins lorsque K est petit, favoriser la sédentarité de la chaîne de Markov. D'ailleurs, le fait que les taux d'acceptation soient nettement plus élevés pour le MTM revisité comparativement au MTM standard (voir tables 2.2 et 3.3) est dû au fait que des états préalablement inclus dans la chaîne font partie de l'état proposé. Ces états sont donc généralement favorisés comparativement aux autres candidats, ce qui par le fait même favorise le phénomène de sédentarité tout en accroissant les taux d'acceptation.

4.2. Mise-à-jour proposée

Dans le but d'obtenir une probabilité d'acceptation qui dépend d'un nouvel état à chaque itération (contrairement à l'ancienne méthode, dont le groupe de candidats inclut toujours l'état initial), nous suggérons une modification à l'algorithme MTM revisité proposé par [1]. Cette modification vise à contrer la sédentarité de l'algorithme de [1] en rafraîchissant, à chaque itération, un candidat supplémentaire. L'idée est en fait d'éliminer x_k, l'état de la composante active au temps n, du groupe de candidats $(l, y_1, \ldots, y_K)$ au temps $n + 1$.

La mise-à-jour considérée dans ces circonstances consiste donc à conserver l'idée de fixer une paire de composantes (x_k, x_l) du modèle étendu tout en actualisant les autres composantes $x_i, i \neq k, l$ (tel que proposé dans la version de [1]). Les composantes fixées échangeront toujours leurs rôles, donc pour une paire $(k, l) \in \{1, 2, ..., K\}$, la mise à jour de l'état actuel,

$$(k, \mathbf{x}) = (k, x_1, x_2, \ldots, x_k = y_k, \ldots, x_l = y_l, \ldots, x_K)$$

est obtenue à l'aide de l'état proposé :

$$(l, \mathbf{y}) = (l, y_1, y_2, \ldots, y_k = x_k, \ldots, y_l = x_l, \ldots, y_K).$$

Cependant, avant d'effectuer ces étapes, une étape préalable est nécessaire afin que le nouvel algorithme se réduise à l'algorithme de Metropolis-Hastings lorsque $K = 1$.

Dans un premier temps, avant d'inverser les rôles et de rafraîchir les composantes autres que k et l, il faut d'abord commencer par rafraîchir la composante active x_k. Par la suite, on peut se permettre de choisir l. Finalement, on peut échanger les rôles et rafraîchir le reste des composantes.

Une légère modification dans la notation de la distribution de la proposition est nécessaire afin de faire la distinction entre la composante active actuelle et la composante active rafraîchie, notées respectivement x_k et y_k^*. La distribution de la proposition devient donc :

$$\begin{aligned}&\tilde{q}\left((k, y_k^*, x_1, \ldots, x_k); (l, y_1, \ldots, y_k)\right)\\ &\qquad = \begin{cases} \frac{w(x_l, x_k)}{w(y_k^*, x_k) + \sum_{i=1, i\neq k}^{K} w(x_i, x_k)} \prod_{i=1, i\neq l}^{K} q(y_l = x_l; y_i) & \text{si } \ l \neq k; \\ \frac{w(y_k^*, x_k)}{w(y_k^*, x_k) + \sum_{i=1, i\neq k}^{K} w(x_i, x_k)} \prod_{i=1, i\neq l}^{K} q(y_l = y_k^*; y_i) & \text{si } \ l = k. \end{cases}\end{aligned} \tag{4.2.1}$$

Afin de lutter contre la sédentarité de l'algorithme MTM revisité, nous allons exclure la composante active actuelle x_k du prochain état proposé $(l, y_1, \ldots, y_K)$. Maintenant, x_l est la composante active proposée, représentant ainsi la seule valeur actuelle qui est reportée sur la valeur proposée.

La transition de cet algorithme MTM revisité amélioré peut être mise en oeuvre comme suit. Notons que cette transition se base sur les hypothèses mentionnées à la section 4.1.2 pour les fonctions $\lambda(\cdot;\cdot), q(\cdot;\cdot)$ et $w(\cdot;\cdot)$.

Algorithme 4.2.1 (Algorithme MTM revisité amélioré)**.**
Étant donné $(k, x_1, \ldots, x_K)$, l'état actuel de la chaîne de Markov au temps n,

(1) Rafraîchir la composante active actuelle en générant $y_k^* \sim q(x_k, \cdot)$.

(2) Sélectionner l avec une probabilité proportionnelle à $w(y_k^*, x_k)$ et $w(x_j, x_k)$ pour $j = 1, \ldots, K$, $j \neq k$. Selon les hypothèses mentionnées à la section 4.1.2, ces probabilités sont proportionnelles à $\pi(y_k^*), \pi(x_j), j \neq k$.

(3) – Si $l = k$, alors on génère $\boldsymbol{y}_{-k} \sim q(y_k^*; \cdot)$ et on pose $y_k = y_k^*$.
– On accepte $(l, \boldsymbol{y})$ avec une probabilité d'acceptation :

$$\alpha((k, \boldsymbol{x}); (l, \boldsymbol{y})) = \min\left\{1, \frac{\pi(y_k) + \sum_{i\neq k} \pi(x_i)}{\pi(x_k) + \sum_{i\neq k} \pi(y_i)}\right\}. \tag{4.2.2}$$

(4) – *Si* $l \neq k$, *alors on génère* $\boldsymbol{y}_{-k,l} \sim q(x_l;\cdot)$ *et on pose* $y_l = x_l$, $y_k = y_k^*$.
– *On accepte* $(l, \boldsymbol{y})$ *avec probabilité d'acceptation :*

$$\alpha((k,\boldsymbol{x});(l,\boldsymbol{y})) = \min\left\{1, \frac{\pi(y_k)}{\pi(x_k)} \frac{q(y_l,y_k)\prod_{i\neq k,l} q(y_k,x_i)}{\prod_{i\neq k} q(x_k,x_i)} \left[\frac{\pi(y_k)+\sum_{i\neq k}\pi(x_i)}{\pi(x_l)+\sum_{i\neq l}\pi(y_i)}\right]\right\}. \tag{4.2.3}$$

Il s'agit donc, en premier lieu, de rafraîchir la composante d'intérêt actuelle x_k selon la loi $q(x_k;\cdot)$. L'étape suivante serait d'utiliser la composante d'intérêt rafraîchie, y_k^*, afin de sélectionner l, qui devrait être choisi avec une probabilité proportionnelle à $w(y_k^*, x_k)$ et $w(x_j, x_k)$. Ensuite, nous échangerons les rôles en posant $y_l = x_l$ et $y_k = y_k^*$ (ou $y_l = y_k^*$ si $l = k$).

À présent, nous devrons nous assurer que la chaîne de Markov obtenue est caractérisée par sa réversibilité par rapport à $\pi(\cdot)$. Cette réversibilité devrait être assurée, en principe, par les probabilités de succès $\alpha(\cdot;\cdot)$ fournies dans l'algorithme 4.2.1.

Si la chaîne de Markov obtenue par l'algorithme 4.2.1 est réversible, alors

$$\begin{aligned}&\tilde{\pi}^K(k,x_1,x_2,\ldots,x_K)P((k,x_1,x_2,\ldots,x_K),(l,y_1,y_2,\ldots,y_K))\\ &\qquad = \tilde{\pi}^K(l,y_1,y_2,\ldots,y_K)P((l,y_1,y_2,\ldots,y_K),(k,x_1,x_2,\ldots,x_K)),\end{aligned} \tag{4.2.4}$$

où $P(\cdot,\cdot)$ est la probabilité de transition et $\tilde{\pi}^K(\cdot,\cdot)$ est la distribution cible définie sur $\{1,\ldots,K\}\times\mathbb{Z}^K$ pour tout $K \geq 1$, donnée par :

$$\tilde{\pi}^K(k,x_1,x_2,\ldots,x_k) = \frac{1}{K}\pi(x_k)\prod_{i\neq 1, i\neq k}^{K} q(x_k;x_i).$$

Ici, $\pi(\cdot)$ et $q(\cdot;\cdot)$ sont respectivement la densité cible et la densité instrumentale utilisées dans l'algorithme MTM standard.

Il est important de se rappeler que $\pi(\cdot)$ n'est pas une distribution marginale de $\tilde{\pi}^K(\cdot)$, mais $\tilde{\pi}^K(x_k \mid k) = \pi(x_k)$ pour tout $k \in \{1,\ldots,K\}$.

Pour vérifier que les probabilités d'acceptation fournies en (4.2.2) et (4.2.3), donneront une chaîne de Markov réversible, nous allons vérifier la relation en (4.2.4) pour $K = 1, 2, 3$ candidats, ainsi que pour le cas général de K candidats par itération.

4.2.1. Le nombre de candidats utilisés est $K = 1$

Ce cas-ci est trivial, car $k = l = 1$. En simplifiant l'équation (4.2.4), nous obtiendrons :

$$\pi(x_1)q(x_1;y_1)\alpha(x_1;y_1) = \pi(y_1)q(y_1;x_1)\alpha(y_1;x_1).$$

Ceci nous mène à la relation suivante :

$$\alpha(x_1; y_1) = \frac{\pi(y_1)}{\pi(x_1)} \frac{q(y_1; x_1)}{q(x_1; y_1)} \alpha(y_1; x_1).$$

Il est donc évident que l'expression

$$\alpha(x_1; y_1) = \min \left\{ 1, \frac{\pi(y_1)}{\pi(x_1)} \frac{q(y_1; x_1)}{q(x_1; y_1)} \right\}$$

satisfait la condition de réversibilité.

4.2.2. Le nombre de candidats utilisés est $K = 2$

Lorsque le nombre de candidats est $K = 2$, nous aurons plusieurs possibilités pour les composantes k et l. Tout d'abord, nous pouvons avoir $k = 1$ et $l = 2$ ou bien l'inverse, c'est-à-dire $k = 2$ et $l = 1$; les deux cas mèneront à un résultat similaire. Ensuite, nous avons le cas où $k = l = 1$ ou encore $k = l = 2$, qui mèneront tous les deux à un même résultat également.

(**1**) $k = 1$ et $l = 2$

$$\begin{aligned}
&\tilde{\pi}^K(1, x_1, x_2) P((1, x_1, x_2), (2, y_1, y_2)) \\
&= \frac{1}{2} \pi(x_1) q(x_1; x_2) \left[q(x_1; y_1) \frac{w(x_2, x_1)}{w(y_1, x_1) + w(x_2, x_1)} q(x_2; y_2) \right] \alpha\left((1, x_1, x_2); (2, y_1, y_2)\right).
\end{aligned}$$

Rappelons que si

$$w(y, x) = \pi(y) q(y; x) \lambda(y, x), ; \quad \lambda(y, x) = \left\{ \frac{q(x; y) + q(y; x)}{2} \right\}^{-1}$$

et $q(x; y)$ est symétrique en x et y, alors $w(y, x) = \pi(y)$.

Nous obtiendrons dans ce cas :

$$\begin{aligned}
&\frac{1}{2} \pi(x_1) q(x_1; x_2) \left[q(x_1; y_1) \frac{\pi(x_2)}{\pi(y_1) + \pi(x_1)} q(x_2; y_2) \right] \alpha\left((1, x_1, x_2), (2, y_1, y_2)\right) \\
&\quad = \frac{1}{2} \pi(y_2) q(y_2; y_1) \left[q(y_2; x_2) \frac{\pi(y_1)}{\pi(y_1) + \pi(x_2)} q(y_1; x_1) \right] \alpha\left((2, y_1, y_2), (1, x_1, x_2)\right).
\end{aligned}$$

On déduit alors

$$\begin{aligned}
\alpha((1, x_1, x_2); (2, y_1, y_2)) &= \min \left\{ 1, \frac{\pi(y_2) q(y_2; y_1) \left[q(y_2; x_2) \frac{\pi(y_1)}{\pi(y_1) + \pi(x_2)} q(y_1; x_1) \right]}{\pi(x_1) q(x_1; x_2) \left[q(x_1; y_1) \frac{\pi(x_2)}{\pi(y_1) + \pi(x_1)} q(x_2; y_2) \right]} \right\} \\
&= \min \left\{ 1, \frac{\pi(y_2)}{\pi(x_1)} \frac{q(y_2; y_1)}{q(x_1; x_2)} \frac{\pi(y_1)}{\pi(x_2)} \left[\frac{\pi(y_1) + \pi(x_1)}{\pi(y_1) + \pi(x_2)} \right] \right\}.
\end{aligned}$$

Si $x_l = y_l$ (c'est-à-dire $x_2 = y_2$ dans ce cas-ci), alors

$$\alpha((1, x_1, x_2); (2, y_1, y_2)) = \min\left\{1, \frac{\pi(y_1)}{\pi(x_1)} \frac{q(y_2; y_1)}{q(x_1; x_2)} \left[\frac{\pi(y_1) + \pi(x_1)}{\pi(y_1) + \pi(x_2)}\right]\right\}. \quad (4.2.5)$$

D'après le résultat obtenu en (4.2.5), il faudra évaluer la fonction $\pi(\cdot)$ à trois valeurs : x_1, x_2 et y_1. Théoriquement, nous disposons déjà de deux valeurs, $\pi(x_1)$ et $\pi(x_2)$, qui proviennent de l'itération précédente. Il suffira donc de calculer $\pi(y_1)$, ce qui donne un algorithme computationnellement moins coûteux que le MTM standard, pour lequel il faudra évaluer trois valeurs, soit $\pi(y_1)$, $\pi(y_2)$ et $\pi(x^*)$.

(**2**) $k = l = 1$

$$\frac{1}{2}\pi(x_1)q(x_1; x_2)\left[q(x_1; y_1)\frac{w(y_1, x_1)}{w(y_1, x_1) + w(x_2, y_1)}q(y_1; y_2)\right]\alpha\left((1, x_1, x_2); (1, y_1, y_2)\right)$$
$$= \frac{1}{2}\pi(y_1)q(y_1; y_2)\left[q(y_1; x_1)\frac{w(x_1, y_1)}{w(x_1, y_1) + w(y_2, y_1)}q(x_1; x_2)\right]\alpha\left((1, y_1, y_2); (1, x_1, x_2)\right).$$

Sous les mêmes hypothèses qu'auparavant, nous avons

$$\frac{1}{2}\pi(x_1)q(x_1; x_2)\left[q(x_1; y_1)\frac{\pi(y_1)}{\pi(y_1) + \pi(x_2)}q(y_1; y_2)\right]\alpha\left((1, x_1, x_2); (1, y_1, y_2)\right)$$
$$= \frac{1}{2}\pi(y_1)q(y_1; y_2)\left[q(y_1; x_1)\frac{\pi(x_1)}{\pi(x_1) + \pi(y_2)}q(x_1; x_2)\right]\alpha\left((1, y_1, y_2); (1, x_1, x_2)\right).$$

Nous pouvons en déduire que la probabilité d'acceptation

$$\alpha\left((1, y_1, y_2); (1, x_1, x_2)\right) = \min\left\{1, \frac{\pi(y_1)q(y_1; y_2)\left[q(y_1; x_1)\frac{\pi(x_1)}{\pi(x_1)+\pi(y_2)}q(x_1; x_2)\right]}{\pi(x_1)q(x_1; x_2)\left[q(x_1; y_1)\frac{\pi(y_1)}{\pi(y_1)+\pi(x_2)}q(y_1; y_2)\right]}\right\}$$
$$= \min\left\{1, \frac{\pi(y_1) + \pi(x_2)}{\pi(x_1) + \pi(y_2)}\right\}$$

satisfait la condition de réversibilité pour la chaîne de Markov.

Dans cette situation, nous ne pouvons pas poser $x_l = y_l$ (dans le cas présent, $x_1 = y_1$). Ceci impliquerait que l'état présent de la composante active est inclus dans le groupe des candidats, et on serait alors dans le cas du MTM revisité. Pour calculer la probabilité d'acceptation dans ce cas, il faudra évaluer la fonction $\pi(\cdot)$ à quatre valeurs : x_1, x_2, y_1 et y_2. Théoriquement, nous disposons déjà des deux valeurs $\pi(x_1)$ et $\pi(x_2)$ provenant de l'itération précédente. Il suffira donc de calculer $\pi(y_1)$ et $\pi(y_2)$, contrairement à l'algorithme MTM standard, pour lequel il faudra évaluer $\pi(\cdot)$ à trois valeurs, soit $\pi(y_1)$, $\pi(y_2)$ et $\pi(x^*)$.

4.2.3. Le nombre de candidats utilisés est $K = 3$

Dans cette section nous traiterons le cas où $K = 3$; nous procéderons de la même façon qu'à la section précédente. Lorsque le nombre de candidats est $K = 3$, nous aurons plusieurs possibilités pour les composantes k et l. Tout d'abord, nous pouvons avoir $k = 1$ et $l = 2$, ce qui correspond tout simplement au cas $k \neq l$. Un deuxième regroupement de cas possibles est lorsque $k = l$.

(**1**) $k = 1$ et $l = 2$

$$\begin{aligned}
&\frac{1}{3}\pi(x_1)q(x_1;x_2)q(x_1;x_3)\left[q(x_1;y_1)\frac{w(x_2,x_1)}{w(y_1,x_1)+w(x_2,x_1)+w(x_3,x_1)}q(x_2;y_2)q(x_2,y_3)\right]\\
&\qquad\qquad\times\alpha\left((1,x_1,x_2,x_3),(2,y_1,y_2,y_3)\right)\\
&=\frac{1}{3}\pi(y_2)q(y_2;y_1)q(y_2;y_3)\left[q(y_2;x_2)\frac{w(y_1,y_2)}{w(y_1,y_2)+w(x_2,y_2)+w(y_3,y_2)}q(y_1;x_1)q(y_1,x_3)\right]\\
&\qquad\qquad\times\alpha\left((2,y_1,y_2,y_3),(1,x_1,x_2,x_3)\right).
\end{aligned}$$

Sous les mêmes hypothèses qu'avant pour $w(\cdot,\cdot)$, $\lambda(\cdot,\cdot)$ et $q(\cdot,\cdot)$, on obtient

$$\begin{aligned}
&\frac{1}{3}\pi(x_1)q(x_1;x_2)q(x_1;x_3)\left[q(x_1;y_1)\frac{\pi(x_2)}{\pi(y_1)+\pi(x_2)+\pi(x_3)}q(x_2;y_2)q(x_2;y_3)\right]\\
&\qquad\qquad\times\alpha\left((1,x_1,x_2,x_3);(2,y_1,y_2,y_3)\right)\\
&=\frac{1}{3}\pi(y_2)q(y_2;y_1)q(y_2;y_3)\left[q(y_2;x_2)\frac{\pi(y_1)}{\pi(y_1)+\pi(x_2)+\pi(y_3)}q(y_1;x_1)q(y_1;x_3)\right]\\
&\qquad\qquad\times\alpha\left((2,y_1,y_2,y_3);(1,x_1,x_2,x_3)\right).
\end{aligned}$$

On en déduit la probabilité d'acceptation suivante

$$\begin{aligned}
&\alpha((1,x_1,x_2,x_3);(2,y_1,y_2,y_3))\\
&=\min\left\{1,\frac{\pi(y_2)}{\pi(x_1)}\frac{q(y_2,y_1)q(y_2,y_3)}{q(x_1,x_2)q(x_1,x_3)}\frac{\pi(y_1)}{\pi(x_2)}\left[\frac{\pi(y_1)+\pi(x_2)+\pi(x_3)}{\pi(y_1)+\pi(x_2)+\pi(y_3)}\right]\frac{q(y_1;x_3)}{q(x_2;y_3)}\right\}.
\end{aligned}$$

Si $x_l = y_l$ (c'est-à-dire $x_2 = y_2$ dans ce cas-ci), alors

$$\begin{aligned}
&\alpha((1,x_1,x_2,x_3);(2,y_1,y_2,y_3))\\
&\qquad=\min\left\{\frac{\pi(y_1)}{\pi(x_1)}\frac{q(y_2;y_1)}{q(x_1;x_2)}\frac{q(y_1;x_3)}{q(x_1;x_3)}\left[\frac{\pi(y_1)+\pi(x_2)+\pi(x_3)}{\pi(y_1)+\pi(x_2)+\pi(y_3)}\right]\right\}.
\end{aligned}$$

Dans ce cas, il faudra évaluer la fonction $\pi(\cdot)$ à 6 valeurs : y_1, y_2, y_3, x_1, x_2 et x_3. Théoriquement, on connait déjà $\pi(x_1), \pi(x_2)$ et $\pi(x_3)$. De plus, puisque

$\pi(x_2) = \pi(y_2)$, il ne reste qu'à calculer $\pi(y_1)$ et $\pi(y_3)$.

(**2**) $k = l = 1$

$$\frac{1}{3}\pi(x_1)q(x_1;x_2)q(x_1;x_3)\left[q(x_1;y_1)\frac{\pi(y_1)}{\pi(y_1)+\pi(x_2)+\pi(x_3)}q(y_1;y_2)q(y_1;y_3)\right] \times \alpha\left((1,x_1,x_2,x_3);(1,y_1,y_2,y_3)\right)$$
$$= \frac{1}{3}\pi(y_1)q(y_1;y_2)q(y_1;y_3)\left[q(y_1;x_1)\frac{\pi(x_1)}{\pi(x_1)+\pi(y_2)+\pi(y_3)}q(x_1;x_2)q(x_1;x_3)\right] \times \alpha\left((1,y_1,y_2,y_3);(1,x_1,x_2,x_3)\right).$$

On obtient alors la probabilité d'acceptation suivante :

$$\alpha((1,x_1,x_2,x_3);(1,y_1,y_2,y_3)) = \min\left\{1, \frac{\pi(y_1)+\pi(x_2)+\pi(x_3)}{\pi(x_1)+\pi(y_2)+\pi(y_3)}\right\}.$$

De même, comme le cas précédent, la fonction $\pi(\cdot)$ doit être évaluée a 6 valeurs y_1, y_2, y_3, x_1, x_2 et x_3. Dans ce cas, nous aurons à calculer $\pi(y_1), \pi(y_2)$ et $\pi(y_3)$, puisque les autres valeurs sont déjà calculées à l'itération précédente. Pour le MTM standard avec $K = 3$, nous aurions à évaluer la fonction $\pi(\cdot)$ à cinq nouvelles valeurs.

4.2.4. Le nombre de candidats utilisés est K

Nous allons maintenant généraliser notre analyse pour K candidats. Comme auparavant, nous distinguerons deux possibilités, $k \neq l$ et $k = l$.

(**1**) $k \neq l$

$$\frac{1}{K}\pi(x_k)\prod_{i\neq k}q(x_k;x_i)\left[q(x_k;y_k)\frac{\pi(x_l)}{\pi(y_k)+\sum_{i\neq k}\pi(x_i)}\prod_{i\neq k}q(x_l;y_i)\right] \times \alpha\left((k,x_1,x_2,\ldots,x_K),(l,y_1,y_2,\ldots,y_K)\right)$$
$$= \frac{1}{K}\pi(y_l)\prod_{i\neq l}q(y_l;y_i)\left[q(y_l;x_l)\frac{\pi(y_k)}{\pi(x_l)+\sum_{i\neq l}\pi(y_i)}\prod_{i\neq l}q(y_k;x_i)\right] \times \alpha\left((l,y_1,y_2,\ldots,y_K),(k,x_1,x_2,\ldots,x_K)\right).$$

On déduit alors que

$$\alpha((k,x_1,x_2,\ldots,x_K);(l,y_1,y_2,\ldots,y_K)) = \min\left\{1, \frac{\pi(y_l)}{\pi(x_k)}\frac{\prod_{i\neq l}q(y_l;y_i)}{\prod_{i\neq k}q(x_k;x_i)}\frac{\pi(y_k)}{\pi(x_l)}\left[\frac{\pi(y_k)+\sum_{i\neq k}\pi(x_i)}{\pi(x_l)+\sum_{i\neq l}\pi(y_i)}\right]\frac{\prod_{i\neq k,l}q(y_k;x_i)}{\prod_{i\neq k,l}q(x_l;y_i)}\right\}.$$

Si $x_l = y_l$, alors

$$\alpha((k, x_1, x_2, \ldots, x_K); (l, y_1, y_2, \ldots, y_K)) = \min\left\{1, \frac{\pi(y_k)}{\pi(x_k)} \frac{q(y_l; y_k)}{\prod_{i\neq k} q(x_k; x_i)} \left[\frac{\pi(y_k) + \sum_{i\neq k} \pi(x_i)}{\pi(x_l) + \sum_{i\neq l} \pi(y_i)}\right] \prod_{i\neq k,l} q(y_k; x_i)\right\}.$$

Dans ce cas, il faudra évaluer la fonction $\pi(\cdot)$ à $K-1$ nouvelles valeurs pour $y_j, j = 1, 2, \ldots, K$ pour tout $j \neq l$. Dans l'algorithme MTM standard, nous aurions à évaluer la fonction $\pi(\cdot)$ à $2K-1$ nouvelles valeurs par itération.

(**2**) $k = l$

$$\begin{aligned}
&\frac{1}{K}\pi(x_k) \prod_{i\neq k} q(x_k; x_i) \left[q(x_k; y_k) \frac{\pi(y_k)}{\pi(y_k) + \sum_{i\neq k} \pi(x_i)} \prod_{i\neq k} q(y_k; y_i)\right] \\
&\qquad \times \alpha\left((k, x_1, x_2, \ldots, x_K), (k, y_1, y_2, \ldots, y_K)\right) \\
&= \frac{1}{K}\pi(y_k) \prod_{i\neq k} q(y_k; y_i) \left[q(y_k; x_k) \frac{\pi(x_k)}{\pi(x_k) + \sum_{i\neq k} \pi(y_i)} \prod_{i\neq k} q(x_k; x_i)\right] \\
&\qquad \times \alpha\left((k, y_1, y_2, \ldots, y_K), (k, x_1, x_2, \ldots, x_K)\right).
\end{aligned}$$

D'où on obtient

$$\alpha((k, x_1, x_2, \ldots, x_K); (k, y_1, y_2, \ldots, y_K)) = \min\left\{1, \frac{\pi(y_k) + \sum_{i\neq k} \pi(x_i)}{\pi(x_k) + \sum_{i\neq k} \pi(y_i)}\right\}.$$

Pour calculer la probabilité d'acceptation dans ce cas, il faudra évaluer la fonction $\pi(\cdot)$ à K nouvelles valeurs, contrairement à l'algorithme MTM standard, pour lequel il faudra évaluer $\pi(\cdot)$ à $2K-1$ nouvelles valeurs.

4.3. Exemples pratiques

Cette section présentera des exemples pratiques pour l'algorithme Metropolis à essais multiples revisité amélioré avec différents paramètres. Pour ce faire, nous avons programmé les algorithmes à l'aide du logiciel statistique **R** (programmes en annexe).

Les résultats obtenus sont présentés dans les figures qui suivent. Chaque figure contient trois graphes : celui du haut représente l'extrait (ou la trace) de la chaîne de Markov ; celui du centre illustre l'histogramme des échantillons et celui du bas présente la fonction d'autocorrélation (acf).

Nous présenterons quatre exemples illustrant la performance de l'algorithme MTM revisité amélioré. Nous échantillonnerons d'une densité cible normale, en

générant des candidats à partir d'une densité instrumentale normale. Nous utiliserons le paramètre d'échelonnage de la densité instrumentale le plus approprié (soit $\sigma = 2.4$ tel que mentionné au chapitre 2) ; à chaque exemple, le nombre de candidats (K) sera différent.

La figure 4.1 a été obtenue à l'aide d'un algorithme MTM revisité amélioré pour une densité cible normale, $N(0, 1)$ en générant 50 000 itérations à partir d'une densité intrumentale $N(x_l, 2.4)$ et en se basant sur deux candidats ($K = 2$). Les figures 4.2, 4.3 et 4.4 ont été obtenues de façon similaire en utilisant $K = 5$, $K = 10$ et $K = 30$, respectivement. En comparant les résultats obtenus avec ceux des chapitres 2 et 3, nous observons clairement que les chaînes résultant de l'algorithme MTM revisité amélioré convergent vers la densité cible $\pi(\cdot)$ plus rapidement lorsque $K = 2$ et $K = 5$, que les chaînes obtenues à l'aide de l'algorithme MTM revisité (voir les figures 3.1, 4.1 et 3.2, 4.2). La performance de ce nouvel algorithme est par contre moins bonne que le MTM revisité dans le cas $K = 10, 30$. En fait, à mesure que K croît, la performance de notre algorithme semble empirer. Ceci est possiblement dû au fait que lorsque K est grand et que $l = k$, alors les candidats sont générés à partir de la valeur rafraîchie y_k^* (qui n'a pas encore été acceptée dans notre chaîne de Markov), ce qui fait en sorte que le groupe de candidats est moins approprié que le groupe de candidats de l'algorithme MTM revisité (qui est formé de candidats générés à partir de l'état actuel x_k). Ce phénomène devra être étudié plus en détails dans le futur ; des méthodes pourraient également être proposées afin de contrer ce problème pour de grandes valeurs de K. En particulier, il serait intéressant de voir quelle proportion des itérations favorisent le cas $l = k$ par opposition au cas où l est une composante différente de k ; une proportion élevée confirmerait l'hypothèse exposée précédemment. Les résultats obtenus à l'aide de l'algorithme MTM standard (voir les figures 2.1, 2.2, 2.3 et 2.4) restent plus performants, en terme d'exploration de l'espace, que ceux obtenus à l'aide de l'algorithme MTM revisité amélioré (voir les figures 4.1, 4.2, 4.3 et 4.4). Par contre, en terme de temps, le nouvel algorithme est moins demandant. Une étude plus approfondie sera nécessaire afin de déterminer si la performance nette du nouvel algorithme, c'est-à-dire en tenant compte de l'effort computationnel et de l'exploration de l'espace, est meilleure ou moins concluante que celle de l'algorithme MTM standard.

Les taux d'acceptation obtenus dans chaque cas sont présentés dans le tableau qui suit. Notons que les taux d'acceptation obtenus pour le MTM revisité amélioré sont nettement moins élevés comparativement à ceux obtenus pour le MTM revisité et le MTM standard. Ceci est justifié par le fait que les groupes de candidats générés par le nouvel algorithme sont moins appropriés que les groupes

de candidats générés par le MTM standard, ce qui fait en sorte que pour un même paramètre d'échelonnage σ, les états proposés sont acceptés moins fréquemment dans le nouvel algorithme. Pour ce qui est de l'algorithme MTM revisité, le fait que l'état actif x_k fasse partie du groupe de candidats a pour effet de gonfler artificiellement le taux d'acceptation pour de petites valeurs de K, tel que mentionné précédemment. Pour de grandes valeurs de K, le comportement de l'algorithme MTM revisité est similaire à celui du MTM standard. Une étude plus approfondie du nouvel algorithme sera nécessaire afin de pouvoir analyser plus précisément les causes exactes de la décroissance des taux d'acceptation lorsque K croît.

TABLE 4.1. Les taux d'acceptation obtenus pour les exemples de l'algorithme Metropolis à essais multiples revisité amélioré.

K	2	5	10	30
Taux d'acceptation	0.42	0.37	0.31	0.21

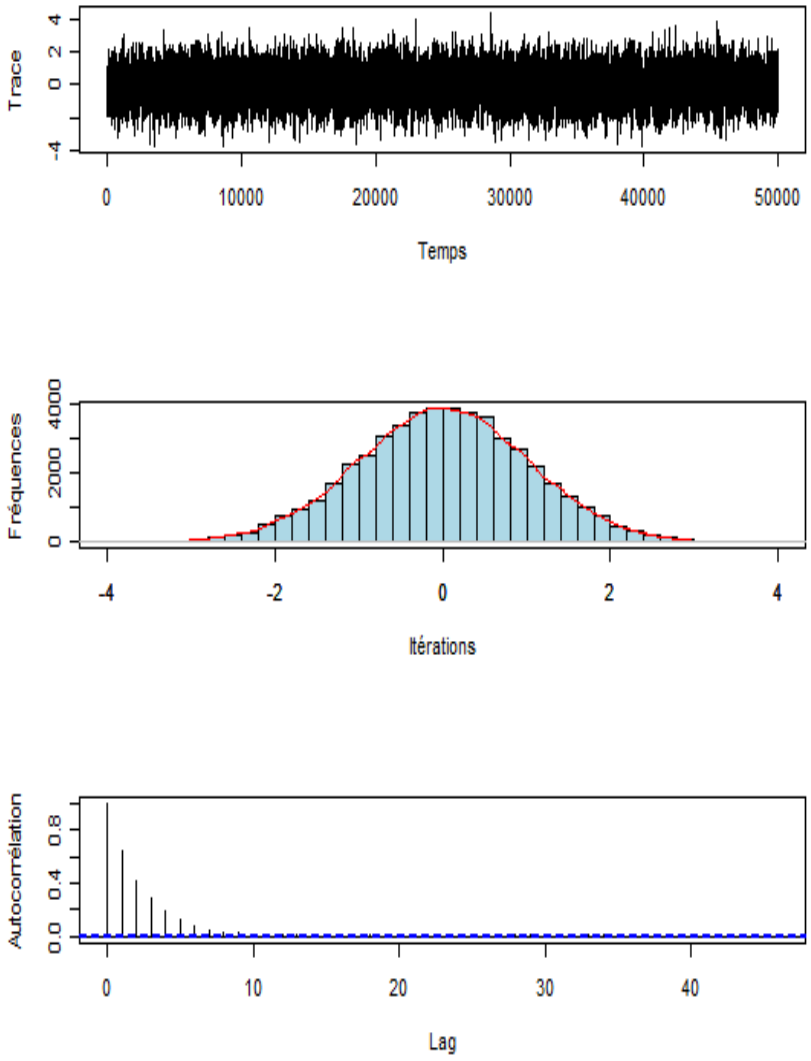

FIGURE 4.1. Algorithme MTM revisité amélioré pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 2$.

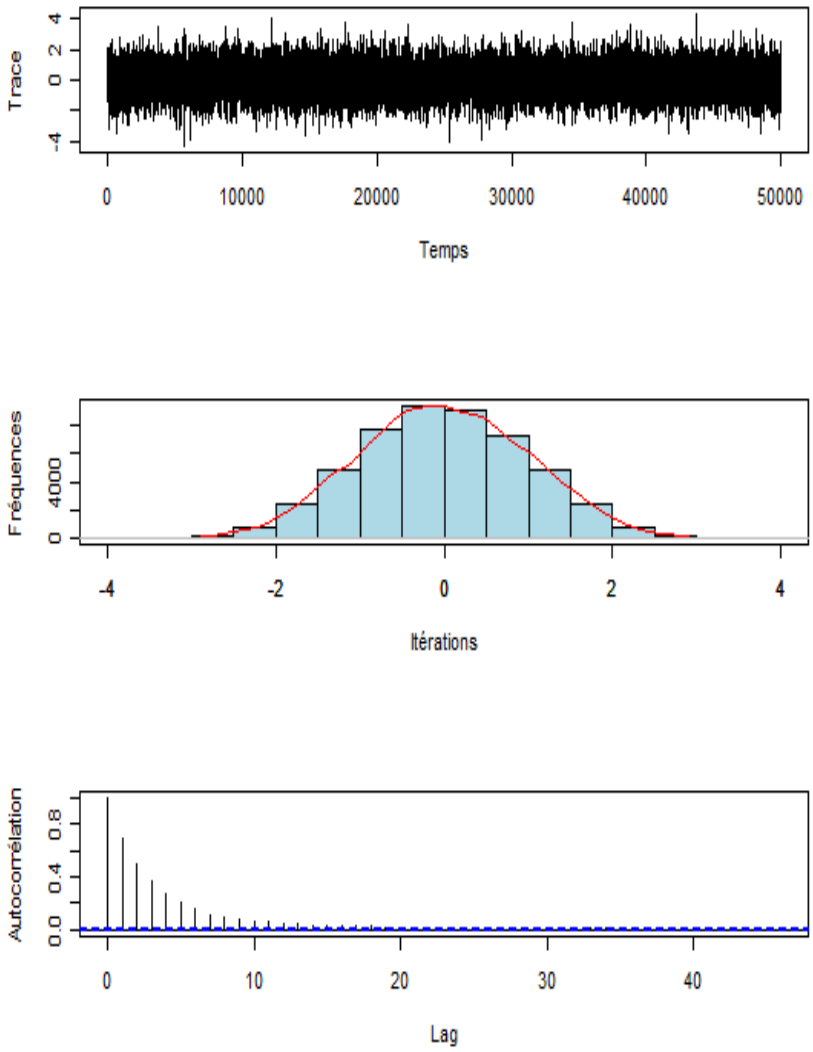

FIGURE 4.2. Algorithme MTM revisité amélioré pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 5$.

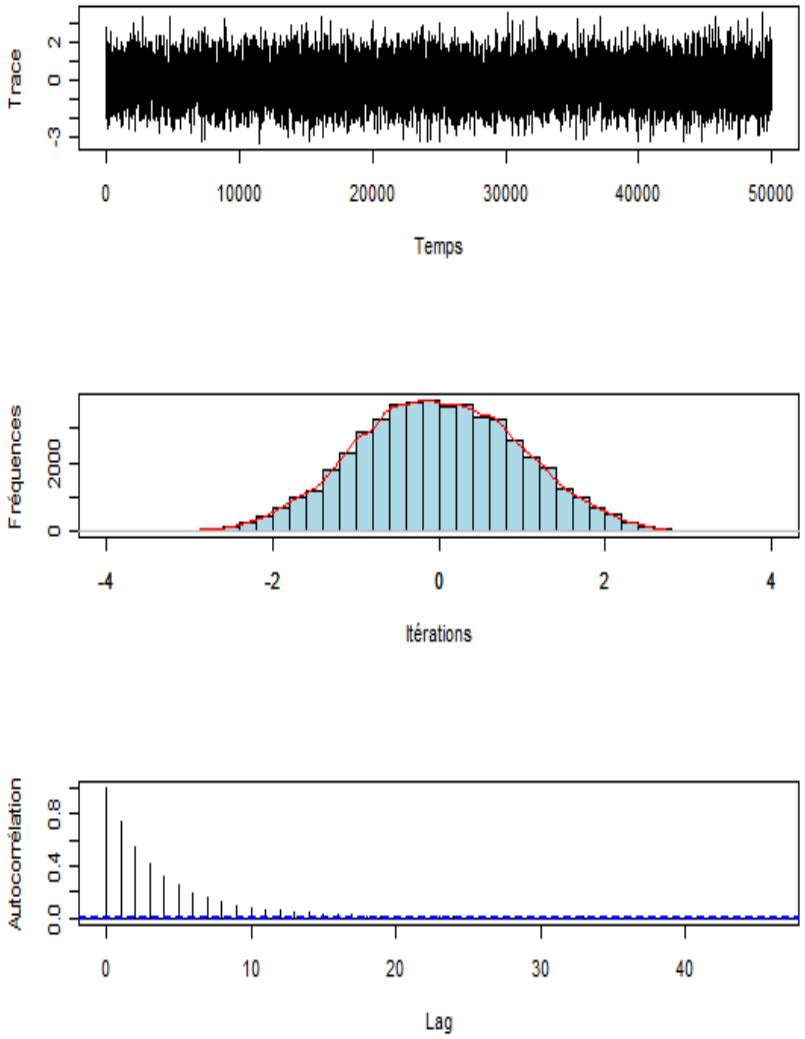

FIGURE 4.3. Algorithme MTM revisité amélioré pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 10$.

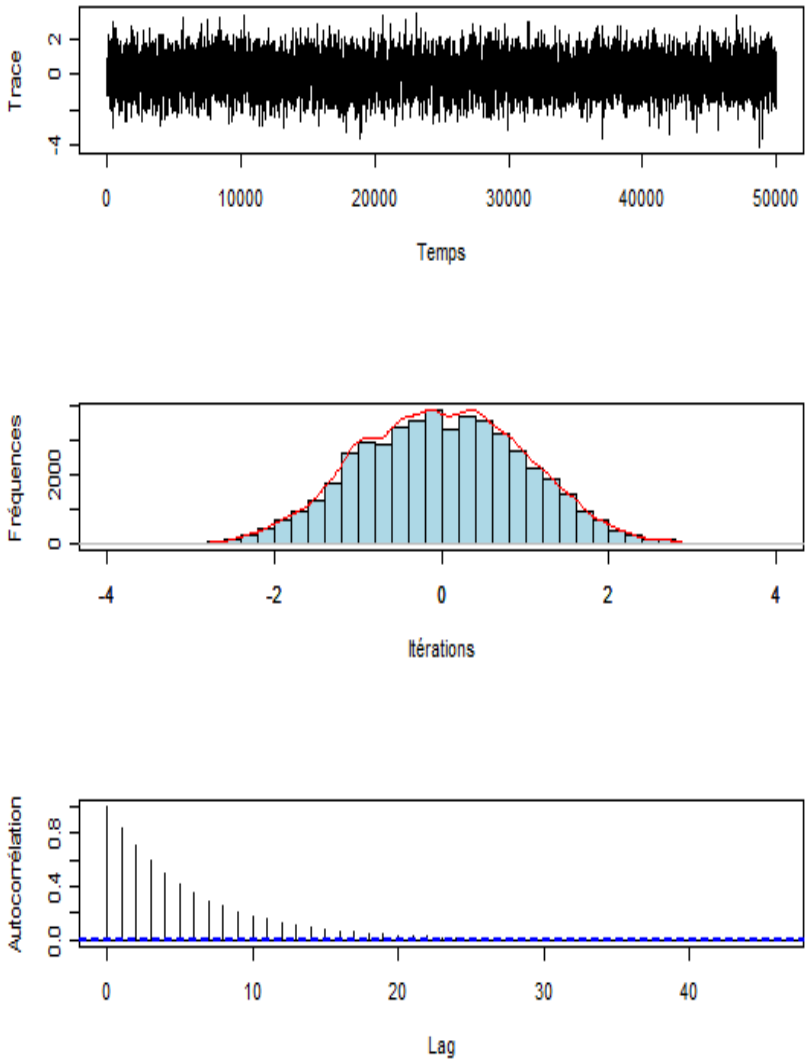

FIGURE 4.4. Algorithme MTM revisité amélioré pour une loi instrumentale $N(x_l, 2.4)$ et une densité cible normale $N(0, 1)$ avec $K = 30$.

CONCLUSION

Nous avons vu dans ce document une introduction sur les algorithmes de Monte Carlo par chaînes de Markov. De plus, nous avons discuté les grands résultats concernant l'algorithme Metropolis à essais multiples. Nous avons également introduit l'algorithme Metropolis à essais multiples revisité. Finalement, nous avons analysé et developpé ce dernier algorithme (MTM revisité) et nous avons présenté une amélioration que nous avons jugée nécessaire pour le bon fonctionnement de cet algorithme. Nous avons donc accompli nos objectifs présentés en introduction.

En premier lieu, nous avons tenté de donner une idée sur les algorithmes MCCM, et de présenter les algorithmes de type Metropolis-Hastings afin de souligner l'importance et la facilité d'application des ces méthodes, ce qui explique leur popularité dans les communautés statistique et bayésienne. Nous avons également montré qu'un bon choix de l'échelle de la marche aléatoire σ est crucial pour obtenir une bonne approximation de la loi cible.

Ensuite, nous avons présenté avec précision l'algorithme Metropolis à essais multiples discuté dans [9], qui est considéré comme une amélioration de l'algorithme Metropolis-Hastings. Cette méthode intègre la notion d'optimisation locale dans un algorithme MCCM et vise à améliorer l'exploration de l'espace dans une itération donnée. Les résultats obtenus par l'implémentation de cette méthode ont montré que l'algorithme répond parfaitement a nos besoins. Effectivement, les échantillons réalisés convergent vers la densité cible prédéfinie. Nous avons montré aussi que plus le nombre de candidats est grand, plus la chaîne de Markov obtenue est proche de la densité cible et converge plus rapidement. Par contre, l'inconvénient de cette méthode était le coût d'implémentation supplémentaire par rapport à la méthode Metropolis-Hastings.

Notre intérêt s'est dirigé, par la suite, vers l'algorithme Metropolis à essais multiples revisité, développé par [1]. Cet algorithme est en fait une nouvelle version de l'algorithme MTM, définie cette fois-ci dans le cadre de l'algorithme Metropolis-Hastings standard. Étant donné que l'algorithme MTM standard a

l'inconvénient d'être coûteux, cette nouvelle version a été mise en place afin d'éviter ce problème. En effet, la méthode MTM revisité est considérée moins coûteuse par rapport à la méthode MTM standard, du point de vue de l'implémentation, puisqu'elle permet de recycler des propositions déjà générées.

L'application de cette méthode a montré que la chaîne de Markov obtenue converge effectivement vers la densité cible. L'inconvénient que présente cette approche se résume dans le nombre d'itérations nécessaires afin de bien explorer l'espace, qui est significativement plus grand qu'avec l'approche Metropolis à essais multiples standard.

Enfin, dans le dernier chapitre, nous avons essayé d'analyser la version MTM revisité afin de cerner les points négatifs qui causent l'inconvénient de cette approche (le temps d'exploration trop long). Tout d'abord, nous avons vérifié si l'algorihme MTM revisité se réduit à l'algorithme Metropolis-Hastings standard, lorsqu'il n'y a qu'un seul candidat. L'analyse a révélé que l'algorithme MTM revisité ne se réduit pas à l'algorithme Metropolis-Hastings standard. En effet, lorsqu'on n'utilise qu'un seul candidat ($K = 1$) dans la version MTM revisité, la chaîne de Markov obtenue stagne à l'état initial à chaque itération, ce qui signifie que l'algorithme ne converge pas.

Notre objectif était, finalement, de trouver une solution qui permettrait la réduction de l'algorithme MTM revisité vers l'algorithme Metropolis-Hastings standard. Dans ce cadre, nous avons montré que le fait d'opter pour l'algorithme 4.2.1, qui rafraîchit la composante active actuelle x_k afin de l'utiliser dans la sélection de l à l'étape suivante, nous permettait d'atteindre ce but. Ce nouvel algorithme nous permet d'affronter la sédentarité de l'algorithme MTM revisité. Une étude plus approfondie nous permettra d'en apprendre davantage au sujet de la performance et des caractéristiques de cette méthode.

Bibliographie

[1] Andrieu, C., Doucet, A. et Holenstein, R. (August 2007, updated June 2009). A Note on the MTM Algorithm.

[2] Bédard, M., Douc, R. et Moulines, E. (2012). Scaling Analysis of Multiple-Try MCMC Methods. *Stochastic Processes and their Applications*, **122**, 758−786.

[3] Bédard, M. et Mireuta, M. (2013). On the Empirical Efficiency of Local MCMC Algorithms with Pools of Proposals. *Canadian Journal of Statistics*, **41**, 657 − 678.

[4] Carroll, R.J., Liu, C., Liang, F. et Wong, W.H. (2010). *Advanced Markov Chain Monte Carlo Methods : Learning From Past Samples*. Wiley : New-York.

[5] Craiu, R.V. et Lemieux, C. (2007). Acceleration of the Multiple-Try Metropolis Algorithm Using Antithetic and Stratified Sampling. *Statistics and Computing*, **17**, 109 − 120.

[6] Gelfand, A.E., et Smith, A.F.M. (1990). Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, **85**, 398 − 409.

[7] Hammersley, J.M. (1967). *Les Méthodes de Monte Carlo*. Dunod : Paris.

[8] Hastings, W.K. (1970). Monte Carlo Sampling Methods Using Markov Chain and Their Applications. *Biometrika*, **57**, 97 − 109.

[9] Liu, S., Liang, F. et Wong, W.H. (2000). The Multiple-Try Method and Local Optimization in Metropolis Sampling. *Journal of the American Statistical Association*, **95**, 121 − 134.

[10] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H. et Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *Journal of Chemical Physics*, **21**, 1087 − 1092.

[11] Pasarica, C., et Gelman, A. (2010). Adaptively Scaling The Metropolis Algorithm Using Expected Squared Jumped Distance. *Statistica Sinica*, **20**, 343 − 364.

[12] Prado, R. et West, M. (2010). *Time Series Modeling, Computation and Inference*. CRC Press : Londres.

[13] Robert, C. et Casella, G. (2010). *Introducing Monte Carlo Methods with R*. Springer : New-York.

[14] Robert, C. et Casella, G. (2009). *Monte Carlo Statistical Methods*. Springer-Vertag : New York, second edition.

[15] Roberts, G. O., Gelman, A. et Gilks, W. R. (1997). Weak Convergence and Optimal Scaling of Random Walk Metropolis Algorithms. *Annals of Applied Probability*, **7**, $110 - 120$.

[16] Robert, G.O. et Rosental, J.S. (2004). General State Space Markov Chains and MCMC Algorithms. *Probability Surveys*, **1**, $20 - 71$.

[17] Shumway, R.H., et Stoffer, D.S. (2010). *Time Series Analysis and Its Applications : With R Examples*. Springer : New-York, third edition.

[18] Tanner, M.A. et Wong, W.H. (1987). The Calculation of Posterior Distributions by Data Augmentation. *Journal of the American Statistical Association*, **82**, $528 - 540$.

[19] Tierney, L. (1994). Markov Chains for Exploring Posterior Distributions (With Discussion). *The Annals of Statistics*, **22**, $1701 - 1728$.

Annexe A

PROGRAMMES R

A.1. Programme R pour l'exemple du chapitre 1

```
norm<-function (n, sigma)
{
        vec <- vector("numeric", n)
        x <- 0
        vec[1] <- x
        for (i in 1:n) {
                innov <- rnorm(1, 0, sigma)
                can <- x + innov
                aprob <- min(1, dnorm(can)/dnorm(x))
                u <- runif(1)
                if (u < aprob)
                        x <- can
                vec[i] <- x
          }
        vec
}


MH<-norm(50000,0.1)
MH1<-norm(50000,2.4)
MH2<-norm(50000,75)
par(mfrow=c(3,1))
plot(ts(MH), xlab = "Temps", ylab = "Trace")
plot(ts(MH1), xlab = "Temps", ylab = "Trace")
plot(ts(MH2), xlab = "Temps", ylab = "Trace")
```

A.2. Programme **R** pour l'algorithme Metropolis-Hastings par marche aléatoire

```
norm<-function (n, sigma)
{
        vec <- vector("numeric", n)
        x <- 0
        vec[1] <- x
        for (i in 1:n) {
                innov <- rnorm(1, 0, sigma)
                can <- x + innov
                aprob <- min(1, dnorm(can)/dnorm(x))
                u <- runif(1)
                if (u < aprob)
                        x <- can
                vec[i] <- x
          }
        vec
}


MH<-norm(50000,0.1)
par(mfrow=c(3,1))
plot(ts(MH), xlab = "Temps", ylab = "Trace")
hist(MH ,main="", xlim = c(-4, 4), breaks = 30, col = "lightblue",
     xlab = "Itérations", ylab = "Fréquences")
par(new = TRUE)
den=density(MH)
plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
     col = "red")
acf(MH,main="", xlab = "Lag", ylab = "Autocorrélation")


MH1<-norm(50000,2.4)
par(mfrow=c(3,1))
plot(ts(MH1), xlab = "Temps", ylab = "Trace")
```

```
hist(MH1,main="",xlim = c(-4, 4), breaks = 30, col = "lightblue",
     xlab = "Itérations", ylab = "Fréquences")
par(new = TRUE)
den=density(MH1)
plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
     col = "red")
acf(MH1,main="", xlab = "Lag", ylab = "Autocorrélation")

MH2<-norm(50000,75)
par(mfrow=c(3,1))
plot(ts(MH2), xlab = "Temps", ylab = "Trace")
hist(MH2,main="",xlim = c(-4, 4), breaks = 30, col = "lightblue",
     xlab = "Itérations", ylab = "Fréquences")
par(new = TRUE)
den=density(MH2)
plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
     col = "red")
acf(MH2, main="",xlab = "Lag", ylab = "Autocorrélation")
```

A.3. Programme **R** pour l'algorithme Metropolis-Hastings indépendant

```
norm3<-function (n,  m, sigma)
{
        vec <- vector("numeric", n)
        x <- 0
        vec[1] <- x
        for (i in 1:n) {
                can <- rnorm(1,  m, sigma)
                aprob <- min(1, (dnorm(can)*dnorm(x, m, sigma))/
                                (dnorm(x)*dnorm(can, m, sigma)))
                u <- runif(1)
                if (u < aprob)
                    x <- can
                vec[i] <- x
          }
        vec
}
```

```
MH<-norm3(50000,1.2, 1)
par(mfrow=c(3,1))
plot(ts(MH), xlab = "Temps", ylab = "Trace")
hist(MH,main="", xlim = c(-4, 4), breaks = 30, col = "lightblue",
     xlab = "Itérations", ylab = "Fréquences")
par(new = TRUE)
den=density(MH)
plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
     col = "red")
acf(MH,main="", xlab = "Lag", ylab = "Autocorrélation")


MH1<-norm3(50000,0.25, 1)
par(mfrow=c(3,1))
plot(ts(MH1), xlab = "Temps", ylab = "Trace")
hist(MH1,main="", xlim = c(-4, 4), breaks = 30, col = "lightblue",
     xlab = "Itérations", ylab = "Fréquences")
par(new = TRUE)
den=density(MH1)
plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
     col = "red")
acf(MH1, main="", xlab = "Lag", ylab = "Autocorrélation")
```

A.4. Programme **R** pour l'algorithme Metropolis à essais multiples

```
function (it = 50000, k = 2, dim = 1, vsd = 2.4)
{

    mu <- 0
    sig <- 1

    x <- rnorm(dim, mu, sd = sig)
    xp <- x
    sx <- rep(0,dim)
```

```
accr <- 0
av <- c()
xvec <- c()

for(i in 1:it){

  y <- rep(x, k) + rnorm(dim * k, 0, sd = vsd)
  sumy <- diff(c(0, cumsum((y - mu)^2)[(1:k) * dim]))
  num <- sum(exp(-sumy/2/sig^2))
  pos <- sample(1:k, 1, replace=TRUE, exp(-sumy/2/sig^2))
  ychosen <- y[(pos - 1) * dim + (1:dim)]
  xstar <- c(x, rep(ychosen, k-1) + rnorm(dim * (k-1), 0, sd = vsd))
  sumxstar <- diff(c(0, cumsum((xstar - mu)^2)[(1:k) * dim]))
  den <- sum(exp(-sumxstar/2/sig^2))

  alpha <- min(1, num/den)
  rr <- runif(1)
  x[rep(rr, dim) < rep(alpha, dim)] <- ychosen[rep(rr, dim) <
       rep(alpha, dim)]

  sx <- sx + (x  - xp)^2
  accr <- accr + (rr < alpha)
  av <- c(av, x)
  xvec <- c(xvec, x[1])
  xp <- x

  print(i)

}

accr <- accr/it
sx <- sum(sx)/it
av <- sum(av)/dim/it


print("MTM")
print(cat("Proposal var = ", vsd))
print(cat("k = ", k))
```

```
    print(cat("Dim = ", dim))
    print(cat("Iterations = ", it))
    print(cat("Acceptance rate = ", accr))
    print(cat("Mean Theta = ", av))
    print(cat("ASJD = ", sx))
    par(mfrow=c(3,1))
    plot(ts(xvec), xlab = "Temps", ylab = "Trace")
    hist(xvec, main="",xlim = c(-4, 4), breaks = 30, col = "lightblue",
          xlab = "Itérations", ylab = "Fréquences")
    par(new=TRUE)
    den=density(xvec)
    plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
         col = "red")
    acf(xvec, main="", xlab = "Lag", ylab = "Autocorrélation")

}
```

A.5. Programme **R** pour l'algorithme Metropolis à essais multiples revisité

```
function (it=50000, k=2, dim=1, vsd = 2.4)
{
    mu <- 0
    sig <- 1

    u <- sample(1:k, 1, replace = TRUE, rep(1/k,k))
    vectu <- (u-1) * dim + (1:dim)
    xu <- rnorm(dim, mu, sd = sig)

    xt <- rep(xu, k) + rnorm(dim * k, 0, sd = vsd)
    xt[vectu] <- xu

    sx <- rep(0,dim)
    accr <- 0
    av <- c()
    xvec <- c()

    for(i in 1:it){
```

```
 sumxt <- diff(c(0, cumsum((xt - mu)^2)[(1:k) * dim]))
 num <- sum(exp(-sumxt/2/sig^2))

 l <- sample(1:k, 1, replace=TRUE, exp(-sumxt/2/sig^2))
 vectl <- (l-1) * dim + (1:dim)
 xl <- xt[vectl]

 yt <- rep(xl, k) + rnorm(dim * k, 0, sd = vsd)
 yt[vectu] <- xt[vectu]
 yt[vectl] <- xl

 sumyt <- diff(c(0, cumsum((yt - mu)^2)[(1:k) * dim]))
 den <- sum(exp(-sumyt/2/sig^2))

 alpha <- min(1, num/den)
 rr <- runif(1)
 xt[rep(rr, dim * k) < rep(alpha, dim * k)] <- yt[rep(rr, dim * k) <
         rep(alpha, dim * k)]
 u[rr < alpha] <- l[rr < alpha]

 sx <- sx + (xt[vectl] - xt[vectu])^2 * (rep(rr, dim) < rep(alpha,dim))
 accr <- accr + (rr < alpha)
 av <- c(av, xt[(vectl * (rep(rr, dim) < rep(alpha, dim)) + vectu *
          (rep(rr, dim) >= rep(alpha, dim)))])

 vectu[rep(rr, dim) < rep(alpha, dim)] <- vectl[rep(rr, dim) <
         rep(alpha, dim)]
 xvec <- c(xvec, xt[vectu[1]])

 print(i)

}

accr <- accr/it
sx <- sum(sx)/it
av <- sum(av)/dim/it
```

```
    print("MTM Andrieu & Doucet")
    print(cat("Proposal var = ", vsd))
    print(cat("k = ", k))
    print(cat("Dim = ", dim))
    print(cat("Iterations = ", it))
    print(cat("Acceptance rate = ", accr))
    print(cat("Mean Theta = ", av))
    print(cat("ASJD = ", sx))
    par(mfrow=c(3,1))
    plot(ts(xvec), xlab = "Temps", ylab = "Trace")
    hist(xvec, main="",xlim = c(-4, 4), breaks = 30, col = "lightblue",
         xlab = "Itérations", ylab = "Fréquences")
    par(new=TRUE)
    den=density(xvec)
    plot(den, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "", main = "",
         col = "red")
    acf(xvec, main="", xlab = "Lag", ylab = "Autocorrélation")

}
```

A.6. Programme **R** pour l'algorithme Metropolis à essais multiples revisité amélioré

```
function (it = 50000, k = 2, dim = 1, vsd = 2.4)
{

mu <- 0
sig <- 1

u <- sample(1:k, 1, replace = TRUE, rep(1/k,k))
vectu <- (u-1) * dim + (1:dim)
xu <- rnorm(dim, mu, sd=sig)

xt <- rep(xu, k) + rnorm(dim * k, 0, sd = vsd)
xt[vectu] <- xu
xtp <- xt

sx <- rep(0, dim)
```

```
accr <- 0
av <- c()
xvec <- c()

for(i in 1:it){

yu <- rnorm(dim, xt[vectu], sd = vsd)
xtr <- xt
xtr[vectu] <- yu

sumxtr <- diff(c(0, cumsum((xtr - mu)^2)[(1:k) * dim]))
num <- sum(exp(-sumxtr/2/sig^2))

l <- sample(1:k, 1, replace = TRUE, exp(-sumxtr/2/sig^2))
vectl <- (l-1) * dim + (1:dim)
xl <- xtr[vectl]

yt <- rep(xl, k) + rnorm(dim * k, 0, sd = vsd)
yt[vectu] <- xtr[vectu]
yt[vectl] <- xl

sumyt <- diff(c(0, cumsum((yt - mu)^2)[(1:k) * dim]))
den <- sum(exp(-sumyt/2/sig^2))

if(u == l){

yts <- yt
yts[vectu] <- xt[vectu]
sumyts <- diff(c(0, cumsum((yts - mu)^2)[(1:k) * dim]))
dens <- sum(exp(-sumyts/2/sig^2))
alpha <- min(1, num/dens)

}

if(u != l){

num1 <- sum((xtr[vectu] - mu)^2)/2/sig^2
den1 <- sum((xt[vectu] - mu)^2)/2/sig^2
```

```
num2 <- sum((yt[vectu] - yt[vectl])^2)/2/vsd^2
num3 <- sum((xt[-c(vectu, vectl)] - rep(yt[vectu], k-2))^2)/2/vsd^2
den2 <- sum((xt[-vectu] - rep(xt[vectu], k-1))^2)/2/vsd^2
alpha <- min(1, exp(den1 + den2 - num1 - num2 - num3) * num/den)

}

rr <- runif(1)
xt[rep(rr, dim * k) < rep(alpha, dim * k)] <- yt[rep(rr, dim * k) <
                    rep(alpha, dim * k)]
u[rr < alpha] <- l[rr < alpha]

sx <- sx + (xt[vectl] - xtp[vectu])^2 * (rep(rr, dim) < rep(alpha, dim))
accr <- accr + (rr < alpha)
av <- c(av, xt[vectl] * (rep(rr, dim) < rep(alpha, dim)) +
                    xtp[vectu] * (rep(rr, dim) >= rep(alpha, dim)))
xvec <- c(xvec, xt[vectl[1]] * (rr < alpha) +
                    xtp[vectu[1]] * (rr >= alpha))
vectu[rep(rr, dim) < rep(alpha, dim)] <- vectl[rep(rr, dim) <
                    rep(alpha, dim)]
xtp <- xt

print(i)

}

accr <- accr/it
sx <- sum(sx)/it
av <- sum(av)/dim/it

print("MTM Andrieu & Doucet modified")
print(cat("Proposal var = ", vsd))
print(cat("k = ", k))
print(cat("Dim = ", dim))
print(cat("Iterations = ", it))
print(cat("Acceptance rate ", accr))
print(cat("Mean Theta = ", av))
print(cat("ASJD = ", sx))
```

```
par(mfrow = c(3, 1))
plot(ts(xvec), xlab = "Temps", ylab = "Trace")
hist(xvec, main="", xlim = c(-4, 4), breaks = 30, col = "lightblue",
             xlab = "Itérations", ylab = "Fréquences")
par(new = TRUE)
densi <- density(xvec)
plot(densi, xlim = c(-4, 4), xlab = "", yaxt = "n", ylab = "",
             main = "", col = "red")
acf(xvec, main = "", xlab = "Lag", ylab = "Autocorrélation")

}
```