Master Ingénierie Mathématique pour la Science des Données

Orientation Modélisation, Calcul, Aide à la Décision

Université de Lorraine

Entreprise encadrante : Université de Lorraine

Sujet de stage : Simulation parfaite de processus stochastiques spatio-temporels

Antonin LAURENT

18 septembre 2019

Table des matières

Résumé / Préface

L'IECL Nancy comprend de nombreux chercheurs dans tout domaine, notamment dans notre cas les mathématiques. Algèbre, analyse, probabilités,etc de nombreux sujets de recherches sont disponibles.
On m'a proposé de travailler dans le domaine des probabilités et statistiques, à propos d'un problème classique du domaine : la génération d'échantillons selon une loi de probabilité ciblée.
L'objectif de ce stage et ultimement de ce mémoire est le suivant : présenter une introduction précise aux méthodes classiques de simulation de tels échantillons, ainsi que fournir des exemples d'applications.
Ce stage est basé principalement sur le livre de Mark Huber :
"Perfect Simulation", où ce mémoire fournit non seulement une traduction d'une partie de ce livre mais aussi :

- l'implémentation sous R de la plupart des algorithmes énoncés
- une révision de certains résultats
- une comparaison des méthodes données

Les pages des théorèmes démontrés par Huber seront fournies avant de les énoncer, si aucune source n'est fournie c'est un résultat qui a été modifié,ajouté dans le cadre de ce stage ou qui n'a pas été démontré.
On présentera dans ce mémoire :

- L'avant simulation exacte : méthodes approchées
- Acceptation/Rejet : premiers pas de la simulation exacte
- CFTP : première méthode avancée
- Bounding chain : évolution du CFTP
- Comparaison de résultats : une analyse concise

Mots clés : simulation exacte, méthode de rejet,chaine de Markov, Coupling From The Past, graphes, modèles d'Ising, modèle hardcore gas, modèle shift

1 Chapitre 1 : Introduction

1.1 Définitions et théorèmes

Définition 1 (Temps d'arrêt)**.** *On dit que T est un temps d'arrêt pour une suite $X_1, X_2, \ldots$ si, connaissant les valeurs de $X_1, X_2, \ldots, X_n$ on peut déterminer si $T \leq n$.*

Définition 2 (Fonction calculable)**.** *Une fonction est dite calculable (computable en anglais) si il existe un algorithme capable de retourner le résultat de la fonction.*

Définition 3 (Algorithme probabiliste)**.** *Soit $\mathcal{I}$ un ensemble d'indices tel que pour tout $I \in \mathcal{I}$, il existe une distribution π_I sur un espace d'états Ω_I. On se donne une suite de variables aléatoires $X_1, X_2, \ldots$ où les $X_I \in S_I$.*
Un algorithme probabiliste est une famille de temps d'arrêts $\{T_I\}_{I\in\mathcal{I}}$ et et de fonctions calculables $\{f_{I,t}\}_{i\in\mathcal{I},t\in\{1,2,\ldots\}}$. La sortie de l'algorithme est $f_{I,T_I}(X_1, \ldots, X_{T_I})$.

Définition 4 (Algorithme de simulation parfaite)**.** *Un algorithme de simulation parfaite est un algorithme probabiliste dont la sortie est une variable aléatoire qui provient d'une distribution cible.*

Les algorithmes de simulation parfaite sont une sous-classe des algorithmes de simulation exacte, algorithmes qui tirent d'une distribution ciblée.
Cependant les algorithmes dont le temps d'arrêt T_I est déterministe (algorithmes tournant selon un nombre fini de choix aléatoires) sont généralement considérés comme algorithmes de simulation exacte mais pas comme algorithmes de simulation parfaite.

Théorème 1 (Théorème fondamental de la simulation parfaite, Huber p.4)**.** *On suppose que pour $U_1, U_2, \ldots$ iid tel que $U_i \sim$ Unif([0,1]) , il existe des fonctions calculables b, g et f telles que la fonction b ait pour image $\{0, 1\}$ et $\mathbb{P}(b(U) = 1) > 0$.*
Pour une variable aléatoire X qui vérifie :

$$X \sim b(U)g(U) + (1 - b(U))f(X, U), \tag{1}$$

soit $T = inf\{t : b(U_t) = 1\}$. On a alors que :

$$Y = f(\ldots f(f(g(U_T), U_{T-1}), U_{T-2}), \ldots, U_1)$$

a la même distribution que X et on a $\mathbb{E}[T] = \frac{1}{\mathbb{P}(b(U)=1)}$.

Démonstration. Soient $X_0, X_1, \ldots$ des tirages indépendants chacun distribués selon X, et $U_1, U_2, \ldots$ iid tel que $U_i \sim$ Unif([0,1]). Pour X_t, fixons $X_{t,t} = X_t$ et récursivement, on pose :

$$X_{t,i} = b(U_{i+1})g(U_{i+1}) + (1 - b(U_{i+1}))f(X_{t,i+1}, U_{i+1}),$$

pour $i \in \{0, \ldots, t - 1\}$.

On a alors d'après la relation (1) : $X_{t,0} \sim X$. On montre ce résultat pour les premiers indices t, les suivants sont prouvés de la même manière.

t=0

$$X_{0,0} = X_0$$

comme on l'a posé précédemment, d'où

$$X_{0,0} \sim X.$$

t=1

$$X_{1,0} = b(U_1)g(U_1) + (1 - b(U_1))f(X_{1,1}, U_1)$$

Or,

$$X_{1,1} = X_1 \sim X.$$

En remplaçant dans l'équation précédente, on obtient :

$$X_{1,0} = b(U_1)g(U_1) + (1 - b(U_1))f(X_1, U_1)$$

D'après la relation (1), on obtient alors $X_{1,0} = X_1 \sim X$.

t=2

$$\begin{aligned} X_{2,0} &= b(U_1)g(U_1) + (1 - b(U_1))f(X_{2,1}, U_1) \\ X_{2,1} &= b(U_2)g(U_2) + (1 - b(U_2))f(X_{2,2}, U_2) \\ &= b(U_2)g(U_2) + (1 - b(U_2))f(X_2, U_2) \end{aligned}$$

À nouveau, par la relation (1), on a : $X_{2,1} = X_2$ On remplace donc par cette valeur dans $X_{2,0}$ et on obtient :

$$X_{2,0} = b(U_1)g(U_1) + (1 - b(U_1))f(X_2, U_1)$$

D'après la relation (1), on obtient alors $X_{2,0} = X_2 \sim X$.

On a donc $X_{0,0}, X_{1,0}, X_{2,0}, \ldots$ de même distribution que X mais pas forcément indépendants. On considère à présent la variable Y énoncée dans le théorème. On a alors la relation suivante : $X_{t,0} = Y$ si $t \geq T$. On illustre ce résultat avec comme exemple $T = 2$ et $t = 2, 3$:
On a donc :

$$Y = f(g(U_2), U_1)$$

Montrons que $X_{2,0} = X_{3,0} = Y$. Étant donné que $T = 2$, on a $b(U_1) = 0$ et $b(U_2) = 1$, d'où :

$$\begin{aligned} X_{2,0} &= f(X_{2,1}, U_1) \\ X_{2,1} &= g(U_2) \end{aligned}$$

on remplace dans la première équation et on obtient

$$X_{2,0} = f(g(U_2), U_1)$$

D'où le résultat pour t=2. Voyons pour t=3.

De même, puisque $T = 2$, on a $b(U_1) = 0$ et $b(U_2) = 1$, d'où :

$$\begin{aligned} X_{3,0} &= f(X_{3,1}, U_1) \\ X_{3,1} &= g(U_2) \end{aligned}$$

on remplace dans la première équation et on obtient

$$X_{3,0} = f(g(U_2), U_1)$$

D'où le résultat.
Ensuite, puisque les U_i sont indépendants, on a que : $\mathbb{P}(T > t) = (1 - \mathbb{P}(b(U) = 1))^t$ et puisque, par hypothèse, $\mathbb{P}(b(U) = 1) > 0$, on a la relation qui tend vers 0 pour t tendant vers l'infini. Il ne reste plus qu'à montrer $Y \sim X$. Pour tout t, pour tout ensemble C :

$$\mathbb{P}(Y \in C) = \mathbb{P}(Y \in C, t \geq T) + \mathbb{P}(Y \in C, t < T)$$

puisque $X_{t,0} = Y$ si $t \geq T$, on a

$$\begin{aligned} &= \mathbb{P}(X_{t,0} \in C, t \geq T) + \mathbb{P}(Y \in C, t < T) \\ &= \mathbb{P}(X_{t,0} \in C) - \mathbb{P}(X_{t,0} \in C, t < T) + \mathbb{P}(Y \in C, t < T) \end{aligned}$$

puisque $X_{t,0} \sim X$, on a

$$= \mathbb{P}(X \in C) - \mathbb{P}(X_{t,0} \in C, t < T) + \mathbb{P}(Y \in C, t < T)$$

Les deux derniers termes sont bornés par $\mathbb{P}(T > t) = (1 - \mathbb{P}(b(U) = 1))^t$, et puisque l'équation est vraie pour n'importe quel t, on obtient : $\mathbb{P}(Y \in C) = \mathbb{P}(X \in C)$ pour tout ensemble C , donc $Y \sim X$. Le fait que $\mathbb{E}[T] = \frac{1}{\mathbb{P}(b(U)=1)}$ provient du fait que T suit une loi géométrique de paramètre $\mathbb{P}(b(U) = 1)$

□

Définition 5 (Algorithme interruptible)**.** *Dans le théorème fondamental de la simulation parfaite, si X et T sont indépendants, on dit que l'algorithme est interruptible, sinon il est non-interruptible.*

Définition 6 (Algortihme de simulation parfaite à lecture unique)**.** *Un algorithme de simulation parfaite qui utilise $X \sim b(U)g(U) + (1 - b(U))f(X, U)$ est un algorithme à lecture unique si $f(X, u) = f(X, u')$ pour tout u,u'. Sinon, c'est un algorithme à lecture double.*

En général, un algorithme interruptible est préférable à un algorithme non-interruptible, et un algorithme à lecture unique est préférable à un algorithme à lecture double.

1.2 Modèles étudiés

1.2.1 Champs aléatoires de Markov

On considère un graphe $G = (V, E)$. V est l'ensemble des nœuds et E est l'ensemble des arêtes. On notera par la suite Δ le degré du graphe (égal au degré du nœud ayant un nombre de voisins maximal). On notera par Ω l'ensemble des labels des nœuds.

Définition 7 (Ensemble séparant)**.** *On dit qu'un sous-ensemble de nœuds S sépare les nœuds i et j si tout chemin du graphe menant i à j passe par S.*

Définition 8 (Champ aléatoire de Markov)**.** *Pour un graphe $G = (V, E)$ et un ensemble de labels Ω, on dit que la distribution de X sur Ω est un champ aléatoire de Markov si, pour tous les nœuds i et j, pour tous les ensembles S séparant i de j, on a : $[X(i)|X(j), X(S)] \sim [X(i)|X(S)]$. Un état $x \in \Omega$ est appelé une configuration.*

Définition 9 (Clique)**.** *Une clique est un sous-ensemble de nœuds du graphe tel que chaque paire de nœuds soit connectée par une arête.*

Théorème 2 (Théorème de Hammersley-Clifford)**.** *Pour un graphe fini $G = (V, E)$, la distribution π est un champ aléatoire de Markov si elle a pour densité f_X et qu'il existe des fonctions ϕ_C pour toute clique C telles que f_X puisse s'écrire :*

$$f_X(x) = \frac{1}{Z} \prod_{C \in cliques(G)} \phi_C(x)$$

Définition 10 (Auto-modèle)**.** *On dit qu'un champ aléatoire de Markov est un auto-modèle, si il existe des fonctions f_i et g_i telles que la densité de $X \sim \pi$ peut-être écrite sous la forme :*

$$f_X(x) = \frac{1}{Z}\left[\prod_{i\in V} f_i(X(i))\right]\left[\prod_{\{i,j\}\in E} g_{\{i,j\}}(X(i), X(j))\right]$$

Dans la suite, on définit des exemples bien connus d'auto-modèles que l'on étudiera par la suite.

Exemple 1 : Modèle d'Ising

Le modèle d'Ising est un auto-modèle de paramètres μ et β , où $\Omega = \{-1, 1\}^V$, et $f(c) = exp(\mu c)$, $g(c_1, c_2) = exp(\beta \mathbf{1}(c_1 = c_2))$. Dans la littérature, on appelle μ magnétisation et β la température inverse. Lorsque $\beta > 0$, on dit que le modèle est ferromagnétique. Lorsque $\mu = 0$, on dit que le modèle est sans champ magnétique.

Exemple 2 : Modèle Hard-core gas

Le modèle hard-core gas est un auto-modèle défini sur $\{0, 1\}^V$ et de paramètre $\lambda > 0$ où $f(c) = \lambda^c$ et $g(c_1, c_2) = 1 - c_1 c_2$. Lorsqu'un noœud ν a pour label 1, on dit que le nœud est occupé, sinon, il est inoccupé.

Exemple 3 : Modèle de Strauss

Le modèle de Strauss est un auto-modèle sur $\{0, 1\}^V$ de paramètres $\lambda > 0$ et $\gamma \in [0, 1]$ où $f(c) = \lambda^c$ et $g(c_1, c_2) = 1 + (\gamma - 1)c_1 c_2$.Lorsqu'un noœud ν a pour label 1, on dit que le nœud est occupé, sinon, il est inoccupé.

1.2.2 Permutations

Un problème classique de distribution sur les permutations est lié à la recherche du permanent d'une matrice non-négative.

Exemple

Pour une matrice non-négative $w(i, j)$, notons :

$$w(\sigma) = \prod_{i=1}^{n} w(i, \sigma(i))$$

Tant qu'il existe au moins une permutation σ telle que $w(i, \sigma(i)) > 0$ pour tout i, cela donne une densité non normalisée sur l'ensemble des permutations. La constante de normalisation pour cette densité est alors appelée le permanent de la matrice $w(i, j)$. Si

aucun σ ne vérifie cette relation, le permanent est 0.
Une autre distribution importante que l'on va considérer est la distribution uniforme sur les permutations où certains objets doivent avoir une position plus faible que les autres.

Définition 11 (Relation d'ordre partiel)**.** *Soit un ensemble P. Une relation d'ordre partiel sur P est une relation binaire $\preceq$ telle que pour tout $a, b, c \in P$, la relation est :*

1. *(Réflexive) $a \preceq a$*
2. *(Antisymétrique) Si $a \preceq b$ et $b \preceq a$, alors $a = b$*
3. *(Transitive) Si $a \preceq b$ et $b \preceq c$, alors $a \preceq c$*

Un ensemble disposant d'une relation d'ordre partiel est parfois appelé poset d'après l'anglais partially ordered set.

Définition 12 (Extension linéaire d'un ordre partiel)**.** *Une extension linéaire d'un ordre partiel sur $1, \ldots, n$ est une permutation pour laquelle si i et j sont tels que $\sigma(i) \prec \sigma(j)$, alors $i < j$.*

1.3 Chaînes de Markov et simulation approchée

Jusqu'au développement des algorithmes de simulation parfaite/exacte, la manière principale d'obtenir une réalisation d'une distribution ciblée était une méthode d'approche. Plusieurs algorithmes et méthodes ont été mis en place et l'ensemble de ces méthodes porte le nom de Chaîne de Markov Monte Carlo (Markov Chain Monte Carlo, MCMC dans la littérature).
On ne rappellera pas ici les principales définitions pour les chaînes de Markov mais d'autres seront nécessaires pour la suite. On s'intéressera notamment aux chaînes de Harris et au théorème ergodique associé.

Définition 13 (Chaîne de Harris)**.** *Une chaîne de Markov $\{X_t\}$ sur un espace d'état Ω est une chaîne de Harris si il existe des ensembles mesurables $A, B \in \Omega$ et $\epsilon > 0$ pour $x \in A$ et $y \in B$, et une mesure de probabilité ρ où $\rho(B) = 1$ tels que l'on ait :*

1. *Pour $T_A = inf\left\{t \geq 0 : X_t \in A\right\}, (\forall z \in \Omega)(\mathbb{P}(T_A < \infty | X_0 = z) > 0)$.*
2. *Si $x \in A$ et $C \subseteq B$, alors $\mathbb{P}(X_1 \in C | X_0 = x) \geq \epsilon\rho(C)$*

Définition 14 (Chaîne récurrente)**.** *Soit $R = \inf\{n > 0 : X_n \in A\}$. On dit qu'une chaîne de Harris est une chaîne récurrente si pour tout $x \in A$, $\mathbb{P}(R < \infty | X_0 = x) = 1$. Une chaîne qui n'est pas récurrente est dite transiente.*

Définition 15 (Chaîne apériodique)**.** *Une chaîne de Harris récurrente est apériodique si pour tout $x \in \Omega$, il existe n tel que pour tout $n' \geq n$, $\mathbb{P}(X_{n'} \in A | X_0 = x) > 0$*

Théorème 3 (Théorème ergodique pour les chaînes de Harris)**.** *Soit X_n une chaîne de Harris récurrente et apériodique de distribution stationnaire π. Si $\mathbb{P}(R < \infty | X_0 = x) = 1$ pour tout x, alors, pour $t \to \infty$, pour tout ensemble mesurable C et pour tout x :*

$$|\mathbb{P}(X_t \in C | X_0 = x) - \pi(C)| \to 0$$

Ce théorème est le cœur des méthodes MCMC, puisqu'il "suffit" de construire une chaîne de Harris ayant pour distribution stationnaire la distribution souhaitée et de la faire avancer pendant un nombre infini de pas. Cependant, n'ayant pas un temps infini, les utilisateurs font tourner leurs algorithmes durant un grand nombre de pas et espèrent arriver dans la distribution stationnaire.

Nous pourrons cependant déterminer à quel point la chaîne est proche de la loi stationnaire à l'aide du concept de couplage (voir chapitre 3 pour une définition et application étendue).

Définition 16 (Couplage)**.** *Supposons que* $\{X_t\} \sim \nu_X$ *et* $\{Y_t\} \sim \nu_Y$*. Un couplage de* $\{X_t\}$ *et* $\{Y_t\}$ *est un processus bivarié* $\left\{(X'_t, Y'_t)\right\}$ *tel que* $\left\{X'_t\right\} \sim \nu_X$ *et* $\left\{Y'_t\right\} \sim \nu_Y$*.*

Théorème 4 (Lemme de Couplage)**.** *Soit* $Y_0 \sim \pi$ *et* $X_0 = x_0$ *tels que les deux variables évoluent de manière couplée. Alors, pour tout mesurable* C *:*

$$|\mathbb{P}(X_t \in C | X_0 = x) - \pi(C)| \leq \mathbb{P}(X_t \neq Y_t).$$

1.4 Création de chaînes de Markov

Afin d'utiliser les méthodes MCMC, il faut créer des chaînes de Harris qui convergent vers la distribution π ciblée. Il est généralement mieux de créer des chaînes réversibles plutôt que des chaînes simplement stationnaires.
Nous utiliserons la notation suivante pour la définition de réversibilité : $\pi(dx) = f(x)dx$. Et donc, pour tout mesurable A, $\pi(A) = \int_{x \in A} \pi(dx) = \int_{x \in A} f(x)dx$.
Nous utiliserons l'équation de balance détaillée pour en déduire la réversibilité :

Définition 17 (Équation de balance détaillée)**.** *Une distribution* π *est réversible selon une chaîne de Markov* $\{X_t\}$ *en particulier si :* $\pi(dx)\mathbb{P}(X_{t+1} \in dy | X_t = x) = \pi(dy)\mathbb{P}(X_{t+1} \in dx | X_t = y)$.

Lemme 1 (Huber p.16, lemme 1.2)**.** *Si* π *est réversible, alors* π *est stationnaire.*

Démonstration. Soit Ω l'espace d'état de la chaîne de Markov $\{X_t\}$ considérée et π réversible pour cette chaîne. Pour $X_t \sim \pi$, et C un ensemble mesurable, alors on a :

$$\begin{aligned}
\mathbb{P}(X_{t+1} \in C) &= \mathbb{E}[\mathbf{1}(X_{t+1} \in C)] \\
&= \mathbb{E}[\mathbb{E}[\mathbf{1}(X_{t+1} \in C)|X_t]] \\
&= \int_{x\in\Omega} \mathbb{E}[\mathbf{1}(X_{t+1} \in C)|X_t = x]\pi(dx) \\
&= \int_{x\in\Omega} \mathbb{P}(X_{t+1} \in C|X_t = x)\pi(dx) \\
&= \int_{x\in\Omega}\int_{y\in C} \mathbb{P}(X_{t+1} \in dy|X_t = x)\pi(dx) \\
&= \int_{y\in C}\int_{x\in\Omega} \mathbb{P}(X_{t+1} \in dx|X_t = y)\pi(dy) \\
&= \int_{y\in C} \mathbb{P}(X_{t+1} \in \Omega|X_t = y)\pi(dy) \\
&= \int_{y\in C} \pi(dy) \\
&= \pi(C)
\end{aligned}$$

D'où la stationnarité de π.

□

Plusieurs types de chaînes réversibles existent telles que l'échantillonnage de Gibbs (Gibbs sampler), Metropolis-Hastings,etc. Ces chaînes sont présentées dans la suite.

1.4.1 Échantillonnage de Gibbs

On présente ici un échantillonneur de Gibbs, qui agit sur un espace d'états de la forme C^V. On appelle $\nu \in V$ une dimension du problème. Pour $X_t = x$, l'échantillonneur choisit une dimension ν uniformément sur V. Soit $L(x, \nu)$ l'ensemble des états qui sont exactement les nœuds de la configuration x sauf en ν, écrit de la manière suivante :
$L(x, v) = \{y : (\forall w \in V \setminus \{v\})(y(w) = x(w))\}$. Pour $X_t = x$, l'état suivant X_{t+1} est choisi selon π conditionné à être dans $L(x, v)$.
On présente comme exemple le modèle d'Ising précédemment vu. Une dimension est alors un nœud de la configuration. On choisit donc un nœud uniformément dans V, et on considère les états qui sont exactement x en tous les autres nœuds autres que ν. Pour le modèle d'Ising, la valeur en ν de x est 1 ou -1. On note ces configurations $x_{\nu\to 1}$ et $x_{\nu\to -1}$. On choisit alors l'état suivant entre $x_{\nu\to 1}$ et $x_{\nu\to -1}$, où le choix est fait proportionnellement à π. On a donc :

$$\mathbb{P}(X_{t+1} = x_{\nu\to 1}) = \frac{\pi(\{x_{\nu\to 1}\})}{\pi(\{x_{\nu\to 1}\}) + \pi(\{x_{\nu\to -1}\})}$$

Or, pour le modèle d'Ising,

$$\pi(\{x\}) = \frac{1}{Z}\prod_{i\in V} exp(\mu X(i)) \prod_{\{i,j\}\in E} exp(\beta \mathbf{1}(x(i) = x(j)).$$

Après simplification, on obtient :

$$\begin{aligned}
&\mathbb{P}(X_{t+1} = x_{\nu\to 1}) = \\
&= \frac{exp(\mu)\prod_{\{\nu,j\}\in E} exp(\beta\mathbf{1}(x(j)=1))}{exp(\mu)\prod_{\{\nu,j\}\in E} exp(\beta\mathbf{1}(x(j)=1)) + exp(-\mu)\prod_{\{\nu,j\}\in E} exp(\beta\mathbf{1}(x(j)=-1))} \\
&= \frac{exp(\beta n_1 + \mu)}{exp(\beta n_1 + \mu) + exp(\beta n_{-1} - \mu)}
\end{aligned}$$

Où n_c est le nombre de voisins de ν de label c. Par exemple, si ν est adjacent à trois nœuds de label 1 et un nœud de label -1, alors la probabilité que ν se voit labelliser 1 est $exp(3\beta + \mu)/(exp(3\beta + \mu) + exp(\beta - \mu))$. Mettre ensuite en place cette méthode algorithmiquement est très simple.
Vérifions la réversibilité de la chaîne.

Démonstration. Les cas où la chaîne ne change pas d'état après un pas,(ie $X_t = x_{\nu\to 1}$ et $X_{t+1} = x_{\nu\to 1}$ ou $X_t = x_{\nu\to -1}$ et $X_{t+1} = x_{\nu\to -1}$) nous donnent immédiatement l'équation de balance détaillée, il ne reste alors qu'à traiter deux cas.

Premier cas : $X_t = x_{\nu\to 1}$ et $X_{t+1} = x_{\nu\to -1}$.

On a alors :

$$\pi(\{x_{\nu\to 1}\})\mathbb{P}(X_{t+1} = x_{\nu\to -1}|X_t = x_{\nu\to 1}) = \pi(\{x_{\nu\to 1}\})\frac{exp(-\mu + \beta n_{-1})}{exp(-\mu + \beta n_{-1}) + exp(\mu + \beta n_1)}$$

D'autre part :

$$\pi(\{x_{\nu\to -1}\})\mathbb{P}(X_{t+1} = x_{\nu\to 1}|X_t = x_{\nu\to -1}) = \pi(\{x_{\nu\to -1}\})\frac{exp(\mu + \beta n_1)}{exp(-\mu + \beta n_{-1}) + exp(\mu + \beta n_1)}$$

On aura donc l'égalité entre ces termes si et seulement si on a :

$$\pi(\{x_{\nu\to 1}\})exp(-\mu + \beta n_{-1}) = \pi(\{x_{\nu\to -1}\})exp(\mu + \beta n_1)$$

Les simplifications suivantes s'opèrent dans les termes en π :
— simplification des $\frac{1}{Z}$
— simplification des $\prod_{i\in V} exp(\mu X(i))$ sauf au nœud ν
— simplification des $\prod_{\{i,j\}\in E} exp(\beta\mathbf{1}(x(i) = x(j))$ sauf aux arêtes ayant ν comme extrémité

On obtient alors :

$$exp(\mu + \beta n_1)exp(-\mu + \beta n_{-1}) = exp(-\mu + \beta n_{-1})exp(\mu + \beta n_1)$$

On a donc bien vérifié l'égalité et donc l'équation de balance détaillée.
Le deuxième cas : $X_t = x_{\nu \to -1}$ et $X_{t+1} = x_{\nu \to 1}$ se vérifie de la même manière.

Ayant vérifié tous les cas possibles, on obtient donc l'équation de balance détaillée qui conduit bien à la réversibilité de la chaîne.

□

On propose une mise en place de l'échantillonneur de Gibbs pour le modèle d'Ising sous R en annexe (6.1). Voici, un exemple de sortie pour $\lambda = 0.5$ ou $\lambda = 1$ et $\mu = 0$ sur un graphe carré de taille 10×10 et $\epsilon = 0.1$.

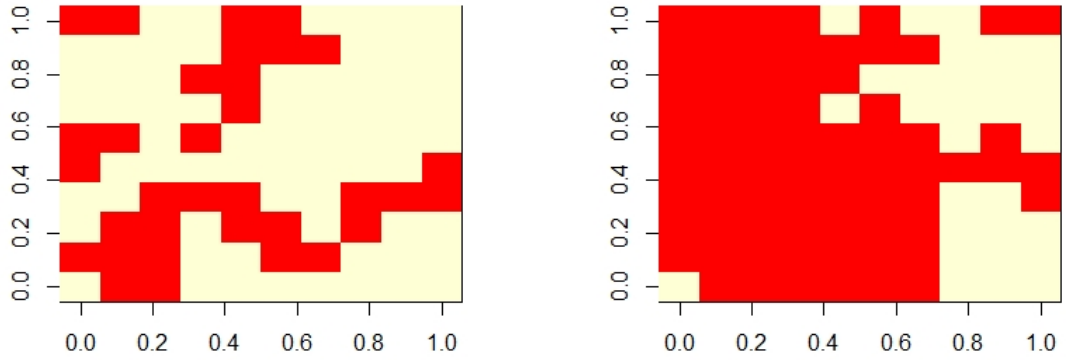


TABLE 1 – Exemple de sortie de IsingGibbsconditionnalstop pour $\mu = 0$, $\epsilon = 0.1$ et $\lambda = 0.5$ (à gauche) et $\lambda = 1$ (à droite). Les cases blanches sont des 1, et les cases rouges des -1.

1.4.2 Metropolis-Hastings

Pour la méthode de Metropolis-Hastings, pour chaque configuration x de la chaîne, on a une densité q_x. On propose ensuite un état suivant y, selon la densité q_x. L'état suivant sera donc x ou y. La probabilité avec laquelle est choisie la configuration y est donnée de telle sorte que la réversibilité s'applique. On la donne ci-dessous. Notez que, pour que cela fonctionne, il est nécessaire que si $q_x(y) > 0$, alors $q_y(x) > 0$ aussi. On a alors le changement vers y avec probabilité :

$$min \left\{ 1, \frac{f_\pi(y) q_y(x)}{f_\pi(x) q_x(y)} \right\}$$

On notera aussi que f_π peut-être normalisée ou non.
En exemple, on considère à nouveau le modèle d'Ising. A chaque pas, on choisit un nœud uniformément dans V puis un label candidat pour ce nœud est choisi uniformément dans $\{-1, 1\}$. On a alors $q_x(y) = q_y(x) = 1/2$. On calcule ensuite, pour n_c le nombre de voisins de ν de label c (où c est le label proposé) et $n_{x(\nu)}$ le nombre de voisins de ν de même label que ν :

$$\frac{f_\pi(y)}{f_\pi(x)} = \frac{exp(n_c\beta + c\mu)}{exp(n_{x(\nu)}\beta + x(\nu)\mu)}$$

L'algorithme est alors très simple à mettre en place. D'abord, vérifions la réversibilité de la chaîne.

Démonstration. Les cas où la chaîne ne change pas d'état après un pas,(ie $X_t = x_{\nu\to 1}$ et $X_{t+1} = x_{\nu\to 1}$ ou $X_t = x_{\nu\to -1}$ et $X_{t+1} = x_{\nu\to -1}$) nous donnent immédiatement l'équation de balance détaillée, il ne reste alors qu'à traiter deux cas.
Premier cas : $X_t = x_{\nu\to 1}$ et $X_{t+1} = x_{\nu\to -1}$.

On a alors :

$$\pi(\{x_{\nu\to 1}\})\mathbb{P}(X_{t+1} = x_{\nu\to -1}|X_t = x_{\nu\to 1}) = \pi(\{x_{\nu\to 1}\})min\left\{1, \frac{exp(-\mu + \beta n_{-1})}{exp(\mu + \beta n_1)}\right\}$$

D'autre part :

$$\pi(\{x_{\nu\to -1}\})\mathbb{P}(X_{t+1} = x_{\nu\to 1}|X_t = x_{\nu\to -1}) = \pi(\{x_{\nu\to -1}\})min\left\{1, \frac{exp(\mu + \beta n_1)}{exp(-\mu + \beta n_{-1})}\right\}$$

On aura donc l'égalité entre ces termes si et seulement si on a :

$$\pi(\{x_{\nu\to 1}\})min\left\{1, \frac{exp(-\mu + \beta n_{-1})}{exp(\mu + \beta n_1)}\right\} = \pi(\{x_{\nu\to -1}\})min\left\{1, \frac{exp(\mu + \beta n_1)}{exp(-\mu + \beta n_{-1})}\right\}$$

Les simplifications suivantes s'opèrent dans les termes en π :
- simplification des $\frac{1}{Z}$
- simplification des $\prod_{i\in V} exp(\mu X(i))$ sauf au nœud ν
- simplification des $\prod_{\{i,j\}\in E} exp(\beta \mathbf{1}(x(i) = x(j))$ sauf aux arêtes ayant ν comme extrémité

On obtient alors :

$$exp(\mu + \beta n_1)min\left\{1, \frac{exp(-\mu + \beta n_{-1})}{exp(\mu + \beta n_1)}\right\} = exp(-\mu + \beta n_{-1})min\left\{1, \frac{exp(\mu + \beta n_1)}{exp(-\mu + \beta n_{-1})}\right\}$$

Par propriété du min, on obtient alors l'égalité entre les termes, qui conduit à l'équation de balance détaillée.

Le deuxième cas : $X_t = x_{\nu\to -1}$ et $X_{t+1} = x_{\nu\to 1}$ se vérifie de la même manière.

Ayant vérifié tous les cas possibles, on obtient donc l'équation de balance détaillée qui conduit bien à la réversibilité de la chaîne.

□

On propose une mise en place pour la méthode Métropolis-Hastings pour le modèle d'Ising sous R en annexe (6.1). Voici un exemple de sortie pour $\lambda = 0.5$,$\lambda = 1$ et $\mu = 0$ sur un graphe carré de taille 10×10 et $\epsilon = 0.1$.

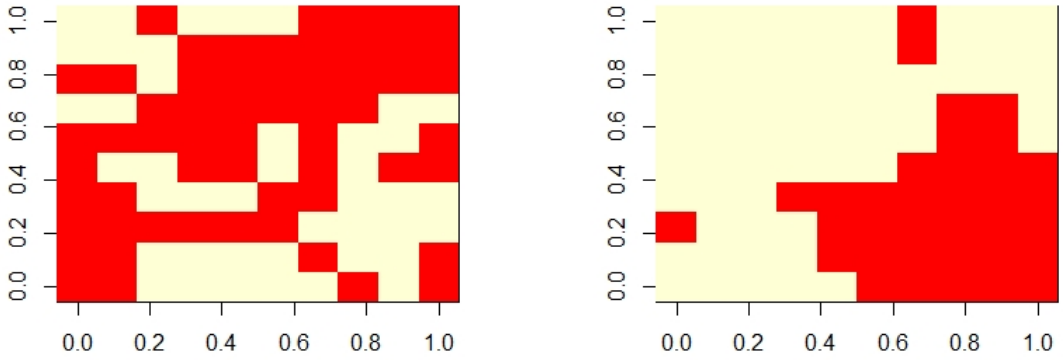


TABLE 2 – Exemple de sortie de MetropolisHastingsconditionnalstop pour $\mu = 0$, $\epsilon = 0.1$ et $\lambda = 0.5$ (à gauche) et $\lambda = 1$ (à droite). Les cases blanches sont des 1, et les cases rouges des -1.

Discussion du critère d'arrêt dans "IsingGibbsconditionnalstop" et "MetropolisHastingsconditionnalstop"

Le premier critère pour l'arrêt du programme est si le nombre de pas i vérifie : $i >= ((log(n^2) * n^2)/\epsilon)$, où n^2 est la taille du graphe carré considéré et ϵ est la "précision" que l'utilisateur souhaite.
Le terme en $n^2 * log(n^2)$ provient du problème du collectionneur. En effet, on souhaite, pour atteindre un certain équilibre, avoir donné la chance à chacun des nœuds d'avoir subi un changement. Cela revient à avoir sélectionné chacun des nœuds au moins une fois, ce qui revient au problème du collectionneur, où en moyenne le temps d'obtention de tous les éléments de la collection se fait en $n * log(n)$ si n est la taille de la collection complète.
Le terme en ϵ permet de s'assurer d'avoir obtenu la visite complète de tous les nœuds, mais servira plus amplement dans l'autre condition d'arrêt présentée ci-après.
Le second critère est si le vecteur des ratios de 1 vérifie :

$$|Y[i] - \frac{1}{\lfloor \frac{n}{\epsilon} \rfloor} \sum_{k=i+1-\lfloor \frac{n}{\epsilon} \rfloor}^{i} Y[k]| < \epsilon$$

C'est-à-dire, si le ratio de 1 n'a pas évolué de plus de ϵ en moyenne depuis le temps $i + 1 - \lfloor \frac{n}{\epsilon} \rfloor$. On parle donc de "précision" ϵ puisque plus cette valeur est petite, plus le temps requis pour que peu de changements s'effectuent en moyenne est grand.
Le terme en n (où n^2 est la taille de la matrice) est ajouté pour obtenir une dépendance à la dimension dans la précision.

1.4.3 Variables aléatoires auxiliaires / Auxiliary random variables

Dans la plupart des applications du MCMC, la densité ciblée X a une structure multiplicative. Pour ces densités, il est possible d'ajouter un vecteur Y de variables aléatoires supplémentaires telles que (X, Y) a une distribution jointe plus simple. Par exemple, on considère à nouveau le modèle d'Ising avec $\mu = 0$. Pour chaque arête $\{i, j\}$, on crée une variable aléatoire auxiliaire $Y(\{i, j\})$ telle que sa distribution conditionnée sur X soit uniforme sur $[0, 1]$ si $X(i) \neq X(j)$ et uniforme sur $[0, exp(\beta)]$ si on a l'égalité.
La densité jointe est alors uniforme sur :

$$X \in \{0,1\}^V \text{ et } Y \in \{[0,\infty) : (\forall \{i,j\})(y\{i,j\}) \leq min(exp(\beta), 1))\}$$

La chaîne de Markov est alors la suivante : pour X donné, il suffit de choisir une nouvelle valeur de Y sachant X, puis choisir un nouveau X conditionné sur Y.
De par la construction de la chaîne, tirer Y sachant X est direct. Cependant, tirer X sachant Y est une autre histoire : prenons pour exemple le modèle d'Ising avec $\beta > 0$. On a alors $exp(\beta) > 1$. Lorsque $y(\{i, j\}) \in [0, 1]$, alors il est possible que $x(i) = x(j)$ ou que $x(i) \neq x(j)$, mais lorsque $y(\{i, j\}) > 1$, alors on a forcément que $x(i) = x(j)$. Les arêtes avec $y(\{i, j\}) > 1$ séparent le graphe en groupes de composants connectés tels que chacun des composants doivent avoir le même label.
Il suffit donc de séparer le graphe en groupes de composants connectés en se servant des arêtes $y(\{i, j\}) > 1$, puis il faut choisir un label uniformément sur $\{-1, 1\}$ pour les composantes de ces groupes.

1.4.4 Shift Chains

Un autre type de chaîne utile pour les modèles répulsifs est le shift. Pour illustrer le modèle, on considère le modèle hard-core où chaque nœud du graphe est soit occupé ($x(\nu) = 1$) soit inoccupé ($x(\nu) = 0$). Chaque nœud occupé contribue d'un facteur λ à la densité. Construire un pas à l'aide de l'échantillonneur de Gibbs est alors très simple.

HCGM-Gibbs Entrée : état x Sortie : nouvel état x

1: Tirer $\nu \leftarrow Unif(V)$
2: Tirer $U \leftarrow Unif([0, 1])$
3: $N_1 \leftarrow$ nombre de voisins de ν de label 1 dans x
4: **if** $U < \lambda/(\lambda + 1)$ et $n_1 = 0$ **then**
5: $\quad x_\nu \leftarrow 1$
6: **end if**

Dès lors qu'un des voisins de ν est occupé, la chaîne ne peut changer la valeur de ν.
Le shift autorise un échange de label entre ν et son voisin occupé avec une certaine probabilité p_{shift}. Lorsque deux voisins de ν ou plus sont occupés, aucun changement n'est possible et ν reste inoccupé. On obtient alors l'algorithme suivant.

HCGM-Shift Entrée : état x Sortie : nouvel état x

Tirer $\nu \leftarrow Unif(V)$
Tirer $U \leftarrow Unif([0,1])$
$S \leftarrow Bern(p_{shift})$
$N_1 \leftarrow$ nombre de voisins de ν de label 1 dans x
if $U < \lambda/(\lambda+1)$ et $n_1 = 0$ **then**
    $x_\nu \leftarrow 1$
else if $U < \lambda/(\lambda+1)$ et $n_1 = 1$ et $S = 1$ **then**
    $w \leftarrow$ l'unique voisin de ν de label 1
    $x(\nu) \leftarrow 1$
    $x(w) \leftarrow -1$
end if

2 Chapitre 2 : Procédures de rejet

2.1 Théorème et exemple simple

Théorème 5 (Huber, p.25,théorème 2.1)**.** *Soit ν une mesure finie sur un ensemble B et soit A un sous-ensemble de B tel que $\nu(A) > 0$. Soit $X_1, X_2, \ldots$ des variables aléatoires iid tirées selon ν sur l'ensemble B. Soit enfin $T = inf\{t : X_t \in A\}$. Alors :*

$$X_T \sim [X_1 | X_1 \in A]$$

C'est à dire, si on tire successivement des valeurs dans B selon ν et que l'on rend la première valeur dans A, alors le résultat provient de la mesure ν conditionnée à être dans A.

Démonstration. Cela suit du théorème fondamental de la simulation parfaite en ayant $U_1, U_2, \ldots$ des tirages dans B selon ν, $b(U) = \mathbf{1}(U \in A)$,$g(U) = U$ et $f(X, U) = X$. □

Le nombre moyen de pas nécessaire pour obtenir un tirage dans A dépend de la taille de A par rapport à la taille de B.

Théorème 6 (Huber, p.26, lemme 2.1)**.** *La probabilité qu'un tirage X dans B selon ν soit dans A est $\nu(A)/\nu(B)$. Le nombre moyen de tirages nécessaires est $\nu(B)/\nu(A)$.*

Démonstration. La première moitié du théorème découle simplement de la définition de mesure de probabilité. La seconde provient du fait que T soit une loi géométrique de paramètre $\nu(A)/\nu(B)$, et donc sa moyenne est $\nu(B)/\nu(A)$. □

Exemple : tirage uniforme sur le cercle unité

On choisit $B = [-1, 1] \times [-1, 1]$ et $A = \{(x, y) \in \mathbb{R}^2 | x^2 + y^2 \leq 1\}$.
L'aire de B est 4 tandis que l'aire du cercle unité est π. Le nombre moyen de lancers est alors $2 * (4/\pi) = 2 * 1.273$, où le facteur 2 provient du nombre d'uniformes nécessaires pour obtenir un tirage dans B.
On propose une mise en place sous R en fin de chapitre.
On déduit de cette méthode un moyen de générer une réalisation de Cauchy standard :

Théorème 7 (Voir source n°3)**.** *Si (X, Y) est un tirage aléatoire sur le cercle unité de $\mathbb{R}^2$, alors X/Y est un tirage aléatoire selon la loi de Cauchy standard.*

Démonstration. Soit ϕ une fonction borélienne bornée sur $\mathbb{R}$. On a, pour (X, Y) variable aléatoire uniforme sur le disque unité :

$$\mathbb{E}\left[\phi\left(\frac{Y}{X}\right)\right] = \frac{1}{\pi}\int_{\mathbb{R}^2} \phi\left(\frac{y}{x}\right) \mathbf{1}_{\{x^2+y^2\leq 1\}} dxdy$$

On passe en coordonnées polaires : $y = r\sin(\theta)$, $x = r\cos(\theta)$. On a alors $dxdy = rdrd\theta$ et on passe d'une intégrale sur $\mathbb{R}^2$ vers une intégrale sur $\mathbb{R}^+ \times [0, 2\pi]$, d'où :

$$\begin{aligned}\mathbb{E}\left[\phi\left(\frac{Y}{X}\right)\right] &= \frac{1}{\pi}\int_{\mathbb{R}^+\times[0,2\pi]}\phi(\tan(\theta))\mathbf{1}_{\{r\leq 1\}}rdrd\theta \\ &= \frac{1}{2\pi}\int_0^{2\pi}\phi(\tan(\theta))d\theta \\ &= 2*\frac{1}{2\pi}\int_{-\pi/2}^{\pi/2}\phi(\tan(\theta))d\theta\end{aligned}$$

On pose ensuite le changement de variables $u = \tan(\theta)$ pour enfin obtenir :

$$\mathbb{E}\left[\phi\left(\frac{Y}{X}\right)\right] = \int_{\mathbb{R}^+}\phi(u)\frac{1}{\pi(1+u^2)}du$$

Ce qui donne bien la densité d'une loi de Cauchy standard. On obtient aussi le résultat pour X/Y puisque l'inverse de la loi de Cauchy standard reste la loi de Cauchy standard. □

2.2 Méthode de rejet pour variables aléatoires à densité

La méthode de rejet de base n'inclut pas les variables aléatoires à densité. Cependant, celles-ci sont traitées aisément à l'aide des variables aléatoires auxiliaires. Pour cela, on considère le théorème suivant, provenant de la théorie de la mesure.

Théorème 8 (Théorème fondamental de la simulation de Monte Carlo)**.** *Soit X variable aléatoire de densité (possiblement non-normalisée) f_X par rapport à une mesure ν sur Ω. Si on a $[Y|X] \sim Unif([0, f_X])$, alors (X, Y) est un lancer provenant de la mesure produit $\nu \times$ Unif sur l'ensemble*
$\{(x, y) : x \in \Omega, 0 \leq y \leq f_x\}$.

Essentiellement, ce théorème nous dit que pour obtenir un lancer selon une certaine densité, il suffit de tirer selon une loi uniforme sur un espace plus grand. De plus, la densité de X n'est pas forcément normalisée, ce qui sera essentiel dans les applications qui suivent.

Pour comprendre comment cela fonctionne, nous allons traiter un exemple en toute généralité, puis un exemple concret.
Soit μ une mesure sur un ensemble Ω. On suppose que, pour une densité g, il est possible de générer un lancer provenant de la densité produit $\mu \times$ Unif sur $\Omega_g = \{(x, y) : x \in \Omega, 0 \leq y \leq g\}$.
Une fois que ce lancer est obtenu, X provient alors de la densité g. Mais supposons ici que le but est d'obtenir un lancer selon une densité f.
Notons d'abord que, si (X, Y) est de mesure $\mu \times$ Unif sur Ω_g, alors $(X, 2Y)$ est de mesure $\mu \times$ Unif sur $\Omega_{2g} = \{(x, y) : x \in \Omega, 0 \leq y \leq 2g\}$. On peut même généraliser : pour toute constante $c > 0$, (X, cY) est de mesure $\mu \times$ Unif sur Ω_{cg}.
Supposons alors que $c \geq sup_{x\in\Omega} f(x)/g(x)$. Alors :

$$\Omega_f \subseteq \Omega_{cg}$$

On peut alors utiliser une méthode de rejet pour obtenir une réalisation provenant de la densité f. Il suffit de générer X selon la densité g, puis de tirer uniformément Y sur $[0, c * g(X)]$. Si Y est aussi dans $[0, f(X)]$ (de sorte que $(X, Y) \in \Omega_f$) alors on accepte (X, Y), ce qui signifie que X est un lancer selon la densité f.
Pour tester si le lancer uniforme de $[0, c*g(X)]$ est dans $[0, f(X)]$, il n'est pas nécessaire de vérifier si l'uniforme tombe effectivement dedans : on peut simplement tirer une variable aléatoire C suivant une loi de Bernouilli de paramètre $f(X)/[c * g(X)]$. Si $C = 1$, on considère que X est un lancer provenant de la densité f.
La probabilité d'acceptation varie inversement avec c, le meilleur choix possible pour c étant alors $sup_{x\in\Omega} f(x)/g(x)$. La valeur de c n'est pas toujours évidente à calculer. Dans ce cas, une majoration suffit : l'algorithme fera simplement plus de lancers en moyenne.
Nous allons maintenant calculer la probabilité d'acceptation d'un lancer.

Lemme 2 (Huber, p.29,lemme 2.2)**.** *Soit $Z_f = \int_{x\in\Omega} f(x)\nu(dx)$ et $Z_g = \int_{x\in\Omega} g(x)\nu(dx)$, les constantes de normalisation pour les densités (possiblement) non-normalisées f et g. Alors, pour la méthode par rejet présentée précédemment, la probabilité d'acceptation est $Z_f/[c * Z_g]$ et le nombre moyen de lancers est $c * Z_g/Z_f$.*

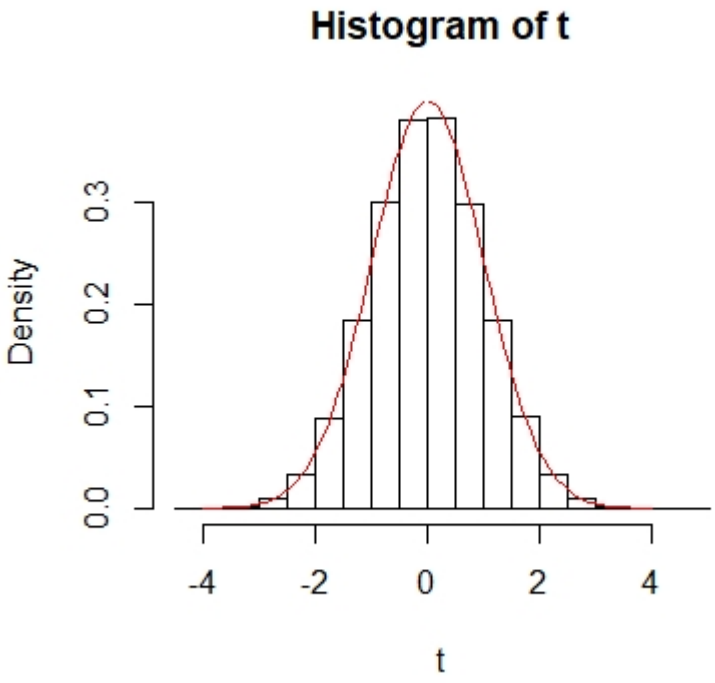
Démonstration. Pour C, la loi de Bernouilli présentée précédemment, le problème est de calculer $\mathbb{P}(C = 1)$.

$$\begin{aligned}
\mathbb{P}(C = 1) &= \mathbb{E}[\mathbb{E}[\mathbf{1}(C = 1)|X]] \\
&= \int_{x\in\Omega} \mathbb{P}(C = 1|X = x)\mathbb{P}(X \in dx) \\
&= \int_{x\in\Omega} f(x)/[cg(x)](g(x)/Z_g)\nu(dx) \\
&= [cZ_g]^{-1} \int_{x\in\Omega} f(x)\nu(dx) \\
&= Z_f/[c * Z_g]
\end{aligned}$$

□

2.2.1 Exemple : D'une Cauchy vers une Normale

Un exemple simple est la situation suivante : on suppose qu'un utilisateur peut générer une réalisation d'une loi de Cauchy standard de densité $g(x) = [\pi(1 + x^2)]^{-1}$. Cet utilisateur souhaite générer une réalisation d'une loi Normale standard de densité $f(x) = \frac{1}{\sqrt{2\pi}} \exp(-x^2/2)$. D'abord, la normalisation n'est pas nécessaire, donc soit $g_1(x) = (1 + x^2)^{-1}$ et $f_1(x) = \exp(-x^2/2)$. On a que $f_1(x)/g_1(x) \leq 2, \forall x$.
On présente une implémentation de l'algorithme sous R en annexe (6.2).
L'histogramme rendu après un lancer du programme est le suivant :



On obtient bien l'allure de la densité d'une loi normale standard.
On récupère aussi le nombre moyen de lancers requis afin d'obtenir une réalisation : 2.50588. On peut retrouver ce résultat de manière théorique en calculant la probabilité d'acceptation :

$$\begin{aligned}\mathbb{P}(C=1) &= \int_{x\in\mathbb{R}} \mathbb{P}(X \in dx, C=1) \\ &= \int_{x\in\mathbb{R}} g(x)\frac{1}{2}(1+x^2)\exp(-x^2/2)dx \\ &= (2\pi)^{-1/2}\end{aligned}$$

Le nombre moyen de lancers est donc $\sqrt{2\pi} = 2.506\ldots$
Il est possible d'accélérer l'algorithme : on a en fait $sup_x(1+x^2)\exp(-x^2/2) = 2/\sqrt{e}$. En remplaçant dans l'algorithme (mais aussi en calculant exactement l'intégrale), le nombre moyen de lancers devient $\sqrt{2\pi/e} = 1.520\ldots$

2.3 Union d'ensembles

Une autre manière d'utiliser les méthodes de rejet est de prendre en compte les multiples manières d'obtenir une réalisation. Par exemple, on considère le problème de génération selon une mesure ν sur $S_1 \cup \ldots \cup S_n$, où les ensembles S_i sont de mesure finie $\nu(S_i)$.
On considère l'algorithme suivant : on tire I dans $\{1, 2, \ldots, n\}$ de telle sorte que

$\mathbb{P}(I = i) \propto \nu(S_i)$. Puis, conditionnellement sur I, on tire X dans S_I selon la mesure ν. Le problème est que X n'est pas tiré selon ν sur $S_1 \cup \ldots \cup S_n$, car il y a plusieurs manières pour lesquelles X aurait pu être choisi. Par exemple, si $X \in S_1$ et $X \in S_2$ mais $X \notin S_3 \ldots S_n$, alors I aurait pu être 1 ou 2 lors du choix de X.

Lemme 3 (Huber, p.30, lemme 2.3)**.** *Pour la procédure précédente, X est un tirage selon ν sur $S_1 \cup \ldots \cup S_n$ avec densité $f(x) = \#\{i : x \in S_i\} / \nu(S_1 \cup \ldots \cup S_n)$*

Démonstration. Soit $x \in S_1 \cup \ldots \cup S_n$ et $C = \nu(S_1 \cup \ldots \cup S_n)$ Alors :

$$\begin{aligned}
\mathbb{P}(X \in dx) &= \sum_{i=1}^{n} \mathbb{P}(X \in dx, I = i) \\
&= \sum_{i=1}^{n} \mathbb{P}(I = i)\mathbb{P}(X \in dx | I = i) \\
&= \sum_{i=1}^{n} C^{-1}\nu(S_i)\mathbf{1}(x \in S_i)\nu(dx)\nu(S_i)^{-1} \\
&= C^{-1}\#\{i : x \in S_i\}
\end{aligned}$$

□

Par exemple, supposons que l'on veuille tirer uniformément sur les 3 cercles unités respectivement centrés en $(0,0)$,$(1/2,0)$ et $(1/2,1/2)$.
On tire alors $I \sim Unif(\{1,2,3\})$. Si $I = 1$, on tire X sur le cercle unité centré en $(0,0)$. Pour $I = 2$ ou $I = 3$, on tire X sur le cercle centré en $(1/2,0)$ et $(1/2,1/2)$ respectivement. Après, avoir tiré I puis X conditionnellement à I, on accepte X avec probabilité égale à l'inverse du nombre de cercles où se situe X.

2.4 Simulation à l'aide des inégalités de Markov et Chernoff

2.4.1 L'inégalité de Markov en tant que procédure de rejet

Lemme 4 (Inégalité de Markov)**.** *On considère une variable aléatoire X non-négative avec probabilité 1 et d'espérance finie $\mathbb{E}[X]$. Alors, pour tout $a > 0$:*

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}[X]}{a}$$

Du point de vue simulation, l'inégalité de Markov nous permet de tirer depuis $[X|X \geq a]$. C'est à dire, si X à pour densité $f_X(x)$ alors le but est de tirer depuis la densité non-normalisée $f_X(x)\mathbf{1}(x > a)$. Pour pouvoir appliquer une méthode de rejet, on utilise la densité non-normalisée $xf_X(x)$. On a alors que $f_X(x)\mathbf{1}(x \geq a)/[xf_X(x)]$ vaut 0 lorsque $x < a$, et $1/x$ lorsque $x \geq a$. Le produit n'est donc jamais supérieur à $1/a$ ce qui permet d'utiliser la méthode de rejet suivante :

Méthode rejet Inégalité de Markov Entrées : f,a Sortie : $X \sim f$ sachant $X > a$

while $C \neq 1$ **do**
  Tirer X selon la densité non-normalisée $xf(x)$
  Tirer $C \sim Bern(a\mathbf{1}(X \geq a)/x)$
end while

On remarque que la constante de normalisation pour la densité $xf(x)$ est simplement : $\int xf(x)\nu(dx) = \mathbb{E}[X]$. On a alors :

$$\mathbb{P}(C = 1) = \int_{x \geq 0} \frac{xf_X(x)}{\mathbb{E}[X]} \frac{a\mathbf{1}(X > a)}{x} = \int_{x \geq a} \frac{af_X(x)}{\mathbb{E}[X]} = \frac{a\mathbb{P}(X > a)}{\mathbb{E}[X]}$$

Puisque la chance d'accepter est au plus 1, cette fraction est aussi au plus 1, ce qui montre l'inégalité de Markov.

2.4.2 Les inégalités de Chernoff en tant que procédure de rejet

Les inégalités de Chernoff donnent des limites supérieures sur la probabilité qu'une somme de variables aléatoires est supérieure (ou inférieure) à une certaine valeur. Soit $S_n = X_1 + \ldots + X_n$, où $X_1, \ldots, X_n$ sont iid ; alors le but est de générer un tirage selon S_n tel que $S_n \geq a$ ou $S_n \leq a$.
L'inégalité de Markov peut aussi être appliquée à une somme de variables aléatoires, mais la borne donnée n'est pas aussi précise que celle obtenue avec les inégalités de Chernoff.

Lemme 5 (Inégalités de Chernoff, Huber, p.40, lemme 2.9)**.** *Soit une variable aléatoire X telle que $\mathbb{E}[e^{tX}]$ soit finie pour $t \in [a, b]$, où a est négatif et b positif. Alors pour un certain $c \in \mathbb{R}$, et en notant par $mgf_X(t)$ la fonction génératrice des moments pour la variable aléatoire X,*

$$\mathbb{P}(X \geq c) \leq mgf_X(t)/\exp(tc), \ \forall t \in [0, b]$$
$$\mathbb{P}(X \leq c) \leq mgf_X(t)/\exp(tc), \ \forall t \in [a, 0]$$

Démonstration. Tout d'abord : $\mathbb{P}(X \geq c) = \mathbb{P}(tX \geq tc)$ pour tout t positif. De plus, $\mathbb{P}(tX \geq tc) = \mathbb{P}(\exp(tX) \geq \exp(tc))$. Enfin, on applique l'inégalité de Markov pour obtenir $\mathbb{P}(\exp(tX) \geq \exp(tc)) \leq \mathbb{E}[\exp(tX)]/\exp(tc)$. Le résultat pour l'autre inégalité est démontré de la même manière. □

Nous considérons à présent ce qu'il se passe lorsque l'on considère les inégalités de Chernoff pour une somme de variables aléatoires indépendantes. On rappelle d'abord un résultat sur les fonctions génératrices des moments.

Lemme 6. *Soit $S_n = X_1 + \ldots + X_n$, où les $\{X_i\}$ sont iid de fonction génératrice des moments finie. Alors on a :* $\mathbb{E}[\exp(tS_n)] = [\mathbb{E}[\exp(tX_i)]]^n$

On applique ce lemme aux inégalités de Chernoff pour obtenir :

Lemme 7. *Soit $S_n = X_1 + \ldots + X_n$, où les $\{X_i\}$ sont iid de fonction génératrice des moments $mgf_{X_i}(t)$ finie pour $t \in [a, b]$, où a est négatif et b positif. Alors on a :*

$$\mathbb{P}(S_n \geq \alpha n) \leq \left(\frac{mgf_{X_i}(t)}{\exp(t\alpha)}\right)^n, \ \forall t \in [0, b]$$
$$\mathbb{P}(S_n \leq \alpha n) \leq \left(\frac{mgf_{X_i}(t)}{\exp(t\alpha)}\right)^n, \ \forall t \in [a, 0]$$

Supposons que $X_i \sim f_X$. Le but est d'utiliser la fonction génératrice des moments pour obtenir un meilleur algorithme de rejet. Pour cela, il doit être possible de générer une réalisation selon la densité $g_t(x) \propto e^{tx} f_x(x)$. Lorsque t est grand, cette densité aura tendance à prendre de plus grandes valeurs. Lorsque t est grand dans les négatifs, g_t tendra à prendre des valeurs plus faibles.
Soit $t > 0$. Pour $x \geq a$, alors $g_t(x) = e^{tx} f_X(x) \geq e^{ta} f_X(x)$. Alors une réalisation de g_t peut être acceptée comme réalisation de f_X avec probabilité e^{ta}/e^{tx}. Si la probabilité que x soit bien plus grand que a est faible, alors la probabilité d'acceptation sera très proche de 1. Une méthode similaire s'applique lorsque $t < 0$. On en déduit alors l'algorithme suivant :

Méthode rejet Inégalités de Chernoff
Entrées : f_X,a,t
Sortie : $S_n = X_1 + \ldots + X_n$ sachant $S_n \geq a$ (lorsque $t > 0$) ou sachant $S_n \leq a$ (lorsque $t < 0$)

Si $t > 0$, alors $A \leftarrow [a, \infty)$, sinon $(-\infty, a]$
while $C \neq 1$ **do**
  Tirer $X_1, \ldots, X_n$ iid selon la densité non-normalisée $e^{tx} f(x)$
  $S_n \leftarrow X_1 + \ldots + X_n$
  Tirer $C \sim Bern(\exp(t(a - S_n))\mathbf{1}(S_n \in A))$
end while

Lemme 8 (Huber, p.41, propriété 2.5)**.** *Supposons que* $mgf_{X_i}(t)\exp(-ta/n) < 1$*. Alors l'algorithme précédent génère une réalisation de* $[S_n|S_n > a]$ *lorsque* $t > 0$ *ou de* $[S_n|S_n < a]$ *lorsque* $t < 0$*.*

Démonstration. On considère tout vecteur $(x_1, \ldots, x_n)$ tel que $\sum x_i = s$. On considère $X_1, \ldots, X_n$ iid de densité $f_X(x)$ et $X'_1, \ldots, X'_n$ iid de densité $\exp(tx)f_X(x)$. Alors la densité de $S'_n = X'_1 + \ldots + X'_n$ est simplement la densité de $S_n = X_1 + \ldots + X_n$ avec un facteur $\exp(tx_1)\exp(tx_2)\ldots\exp(tx_n) = \exp(ts)$.
On génère dans l'algorithme une variable aléatoire de densité $g(s) = \exp(ts)f_{S_n}(s)$ et la densité ciblée est $f(s) = f_{S_n}(s)\mathbf{1}(s \in A)$. On a alors $f(s)/g(s)$ qui est soit 0 lorsque $s \notin A$ soit $\exp(-ts)$ lorsque $s \in A$.
Si $A = [a, \infty)$, alors $t > 0$ et $\exp(-ts) \leq \exp(-ta)$. De même, lorsque $A = (-\infty, a]$, alors $t < 0$ et $\exp(-ts) \leq \exp(-ta)$. On choisit donc $c = \exp(-ta)$ et on obtient $f(s)/[cg(s)] = \exp(ta)\exp(-ts)\mathbf{1}(s \in A)$ de même que dans l'algorithme. □

On propose une mise en place de l'algorithme précédent sous R en annexe sur un exemple simple.
On utilise l'exemple suivant pour tester nos algorithmes : pour $n = 10$ et $p = 0.1$, pour X la binomiale de paramètres précisés précédemment, on veut générer une réalisation de $[X|X >= 5]$.
La méthode de base nous donne un nombre moyen expérimental de lancers de 610.9838 tandis que la méthode à l'aide des inégalités de Chernoff nous donne un nombre moyen expérimental de lancers de 3.698136.
D'où l'efficacité de cette méthode.

2.5 Défaut des méthodes de rejet

Le principal défaut des méthodes de rejet concerne l'approche des problèmes considérés. On prend pour exemple la génération de la variable aléatoire uniforme dans la boule de dimension n.
La méthode présentée précédemment dans le cas $n = 2$ nécessite de tirer uniformément sur le carré $[-1, 1] \times [-1, 1]$. La probabilité d'acceptation p est alors l'aire du cercle sur l'aire du carré, ie, $p = \pi/4$.
On peut généraliser cette méthode aux dimensions supérieures : on tire uniformément dans l'hypercube unité de dimension n : $[-1, 1]^n$, le but étant d'obtenir un tirage uniforme dans la boule unité de même dimension.

Théorème 9 (Volume de la boule unité (voir source n°4))**.** *Le volume de la boule unité en dimension* n *est* $\frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}$*, où* Γ *désigne la fonction Gamma.*

Démonstration. Notons par $V^{(n)}[1]$ le volume de la boule unité de dimension n et de rayon 1. On a d'abord $V^{(1)}[1] = 2$, puis pour tout $n \geq 1$, on a par récurrence, en utilisant la relation suivante : pour $B_d(0, 1)$ la boule unité de dimension d,

$$B_d(0,1) = \left\{(x_1,\dots,x_{d-1},h) | h \in [-1,1], (x_1,\dots,x_{d-1}) \in B_{d-1}(0,\sqrt{1-h^2})\right\}$$

Et en utilisant le théorème de Fubini,

$$V^{(n+1)}[1] = \int_{-1}^{1} V^{(n)}[1] \left(\sqrt{1-x^2}\right)^n dx = V^{(n)}[1] * 2 \int_0^1 \left(1-x^2\right)^{n/2} dx$$

On effectue ensuite le changement de variables $u = x^2$ ce qui nous donne $x = \sqrt{u}$ et $dx = \frac{du}{2\sqrt{u}}$ pour obtenir :

$$V^{(n+1)}[1] = V^{(n)}[1] * 2 \int_0^1 \left(1-x^2\right)^{n/2} dx = V^{(n)}[1] \int_0^1 (1-u)^{n/2}\, u^{-1/2} du$$

L'intégrale à droite est connue comme la fonction bêta, d'où :

$$V^{(n+1)}(1) = V^{(n)}[1] B\left(\frac{n}{2}+1, \frac{1}{2}\right)$$

Or on peut exprimer la fonction bêta par rapport à la fonction gamma pour obtenir :

$$V^{(n+1)}(1) = V^{(n)}[1] \frac{\Gamma(\frac{n}{2}+1)\Gamma(\frac{1}{2})}{\Gamma(\frac{n}{2}+\frac{3}{2})}$$

Enfin, en utilisant le fait que $\Gamma(\frac{1}{2}) = \sqrt{\pi}$, et à l'aide d'une simple récurrence, on a enfin que :

$$V^{(n)}[1] = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}$$

□

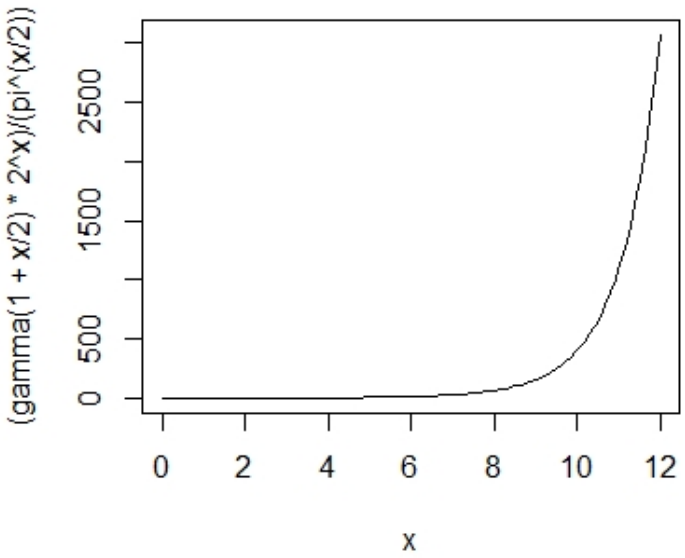
La probabilité d'acceptation p d'un tirage aléatoire dans l'hypercube unité de dimension n en tant que tirage dans la boule unité de même dimension est donc le volume de la boule sur le volume de l'hypercube, soit encore :

$$p = \frac{V^{(n)}[1]}{2^n} = \frac{\pi^{\frac{n}{2}}}{2^n \Gamma(\frac{n}{2}+1)}$$

On en déduit le nombre moyen k de lancers nécessaires à l'acceptation (par simple étude de loi géométrique) :

$$k(n) = \frac{1}{p} = \frac{2^n \Gamma(\frac{n}{2}+1)}{\pi^{\frac{n}{2}}}$$

On trace ci-dessous k en fonction de n pour $n \leq 12$:



On remarque une augmentation très rapide du nombre de lancers nécessaires à l'obtention d'une réalisation selon que la dimension croît.
La méthode de rejet n'est donc plus efficace pour des problèmes dont l'approche n'est pas connue, une étude approfondie étant nécessaire.
Le code permettant d'obtenir cette courbe ainsi qu'une mise en place du problème sous R est disponible en annexe.

3 Chapitre 3 : Coupling From The Past

Définition : Fonction de mise à jour

On dit que $\phi : \Omega \times [0,1] \to \Omega$ est une fonction de mise à jour pour une chaîne de Markov $\{X_t\}$ si, pour $U \sim$ Unif[0,1], $[X_{t+1}|X_t] \sim \phi(X_t, U)$.
La fonction ϕ est déterministe : tout l'aléatoire est contenu dans la variable U.
Toute chaîne qui peut-être simulée sur ordinateur est un exemple de fonction de mise à jour.
Une même chaîne de Markov peut-être représentée par plusieurs fonctions de mise à jour : la fonction de mise à jour n'est pas forcément unique.
À l'aide d'une fonction de mise à jour ϕ, on peut représenter la trajectoire d'une chaîne de Markov $\{X_t\}$. En effet, soient $U_0, U_1, U_2, \ldots$ iid $\sim$ Unif([0,1]). Pour un état initial x_0, on a : $X_1 = \phi(x_0, U_0)$ puis pour $i > 1$, $X_i = \phi(X_{i-1}, U_{i-1})$.
On notera alors la trajectoire jusqu'au temps t sous la forme :

$$\phi_t(x_0, U) = \phi(\phi(\phi(\ldots(\phi(x_0, U_0), U_1), \ldots, U_{t-1}))$$

Ensuite, pour n'importe quels états x_0 et y_0 dans Ω, on définit pour une fonction de mise à jour ϕ : $X_t = \phi_t(x_0, U)$ et $Y_t = \phi_t(y_0, U)$ (en utilisant les mêmes valeurs de U). On appelle ce procédé un couplage. Notons qu'avec ce couplage, si il existe $t \geq 0$ tel que $X_t = Y_t$, alors on dit que les processus ont fusionné (ou se sont rejoints,etc).

Définition : Couplage

Soit $\mathcal{S}$ un ensemble de processus stochastiques définis sur un même ensemble d'indices $\mathcal{I}$ et un même espace d'états Ω. Si il existe un indice $i \in \mathcal{I}$ et un état $x \in \Omega$ tels que pour tout $S \in \mathcal{S}$, on aie $S_i = x$, alors on dit que les processus stochastiques ont coalescé (ou se sont rejoints,couplés,etc).

3.1 CFTP : approche trajectorielle (Baccelli / Brémaud)

Soit P une matrice de transition ergodique sur l'espace d'état fini $E = \{1, \ldots, r\}$, de distribution stationnaire π. Comme définie précédemment, à l'aide d'une fonction de mise à jour h, on peut implémenter la chaîne de Markov de la manière suivante :

$$X_{n+1} = h(X_n, \xi_n)$$

pour une suite de variables aléatoires $\{\xi_n\}_{n\geq 1}$ iid uniformes sur $[0,1]$ et indépendantes de l'état initial.
On considère à présent un tableau de choix aléatoires $\{\xi_k(i)\}_{k\in\mathbb{Z}, i\in E}$ iid uniformes sur $[0,1]$. Pour tout $k \in \mathbb{Z}$ et tout $i \in E$, soit $\left\{X_n^k(i)\right\}_{n\geq k}$ définie par récurrence :

$$X_{n+1}^k(i) = h(X_n^k(i), \xi_n(X_n^k(i)),\ n \geq k,$$

avec pour condition initiale $X_k^k(i) = i$, et h comme définie précédemment.
Pour tout k et i, $\left\{X_n^k(i)\right\}_{n\geq k}$ est une chaîne de Markov homogène de matrice de transition P. Par la structure de récurrence stochastique sous-jacente, les chaînes de la famille définie précédemment sont telles que : pour tout $k \in \mathbb{Z}$, $X_n^k(i) = X_n^k(j)$ implique que $X_m^k(i) = X_m^k(j)$, pour tout $m \geq n$.
On note par :

$$N^+ = \inf\left\{n \geq 0 | X_n^0(1) = X_n^0(2) = \ldots = X_n^0(r)\right\}$$

($= +\infty$ si la condition n'est jamais satisfaite) le temps de coalescence forwards de la chaîne. On notera le temps de coalescence backwards de la chaîne par :

$$N^- = \inf\left\{n \geq 1 | X_0^{-n}(1) = X_0^{-n}(2) = \ldots = X_0^{-n}(r)\right\}$$

($= +\infty$ si la condition n'est jamais satisfaite)

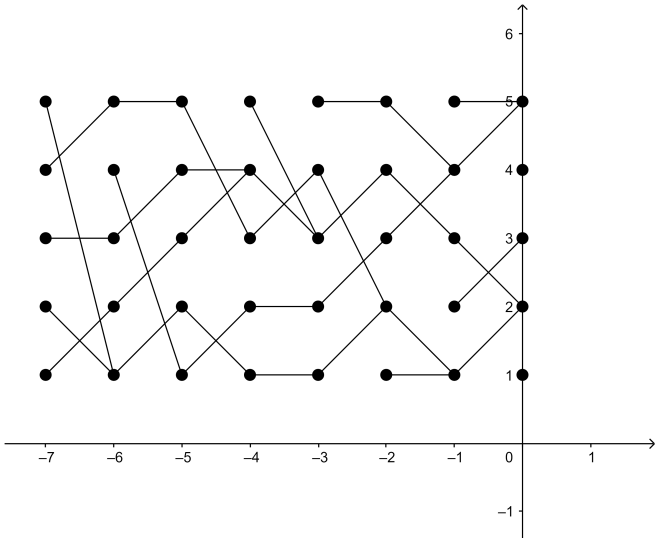


FIGURE 1 – Exemple de temps de coalescence backwards, $N^- = 7$

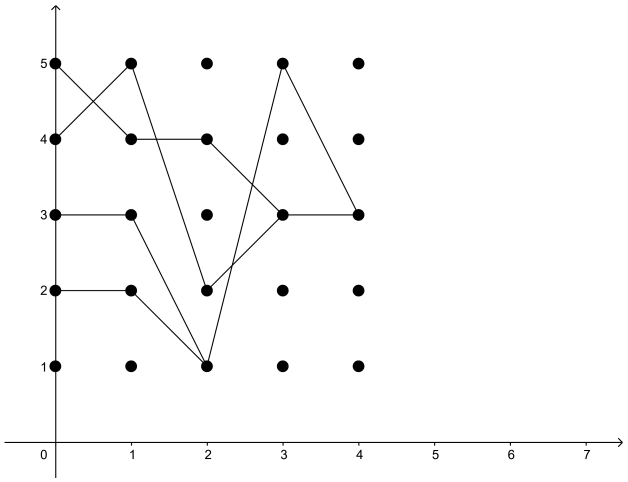


FIGURE 2 – Exemple de temps de coalescence forwards, $N^+ = 4$

Théorème 10 (Baccelli/Brémaud, p.110, propriété 2.5.1)**.** *Le temps de coalescence forwards N^+ est presque sûrement fini.*

Démonstration. Il suffit de prouver le résultat dans le cas de r chaînes de Markov homogènes, complètement indépendantes les unes des autres, de même matrice de transition. Nous n'avons pas l'hypothèse d'indépendance dans la construction des chaînes de Markov donnée précédemment. Cependant, la probabilité de coalescence (probabilité que N^+ soit fini) dans notre situation est bornée inférieurement par la probabilité de coalescence dans le cas complètement indépendant. Pour mieux le comprendre, on construit d'abord le modèle de chaînes indépendantes :

$$\overline{X}_{n+1}(i) = h(\overline{X}_{n+1}(i), \overline{\xi}_{n,i}),\ n \geq 0,$$

(avec pour condition initiale $\overline{X}_0(i) = i$), qui utilise r composantes de mise à jour iid $\left\{\overline{\xi}_{n,i}\right\}$.
La différence entre ce modèle et celui que l'on a introduit réside dans le nombre de mise à jour trop élevé de notre modèle. Afin de construire un ensemble de r chaînes semblable à celui de notre modèle, il suffit d'utiliser les mêmes mise à jour pour deux chaînes dès lors qu'elles se rencontrent. Il est alors clair que le temps de coalescence forwards du modèle ainsi modifié est plus petit ou égal à celui du modèle complètement indépendant.
Il reste alors à prouver que pour un nombre fini de chaînes de Markov homogènes, ergodiques et indépendantes, elles finiront par se rencontrer. Cela suit du fait que le produit de chaînes ergodiques indépendantes est une chaîne ergodique.

□

Théorème 11 (Baccelli/Brémaud, p.111, propriété 2.5.2)**.** *Les variables aléatoires N^+ et N^- ont la même distribution.*

Démonstration. Soit $k \in \mathbb{N}$. On considère le modèle modifié obtenu en remplaçant $\xi_{-k+l}(i)$ par $\xi_l(i)$,pour tout l tel que $0 \leq l \leq k$,et pour $i \in E$. Notons par $N^{'}$ le

temps de coalescence backwards du modèle modifié. Clairement N^- et $N^{'}$ ont la même distribution.

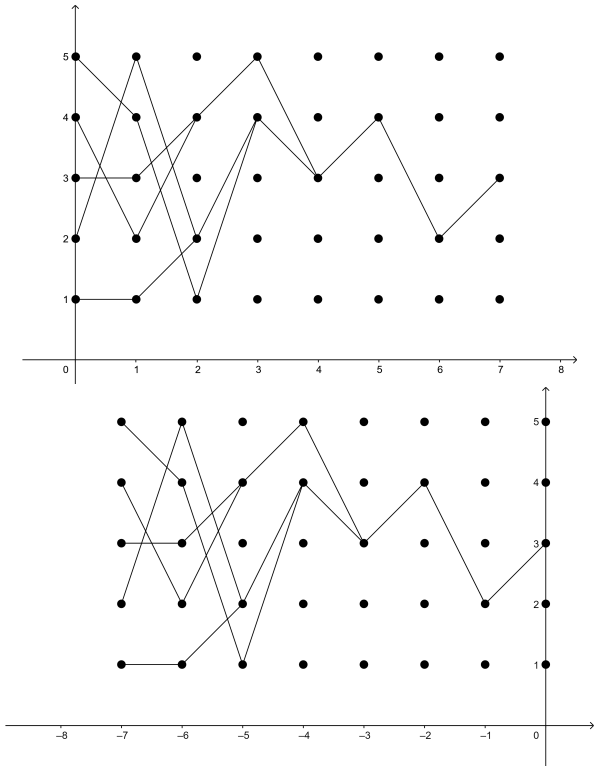


FIGURE 3 – $N^+ \leq k$ implique $N^{'} \leq k$

On suppose maintenant que $N^+ \leq k$. Alors, dans le modèle modifié, les chaînes commençant au temps $-k$ depuis les états $1, \ldots, r$ coalescent au temps $-k + N^+ \leq 0$ (voir figure précédente), et par conséquent $N^{'} \leq k$. Donc, $N^+ \leq k$ implique $N^{'} \leq k$, puis :

$$\mathbb{P}(N^+ \leq k) \leq \mathbb{P}(N^{'} \leq k) = \mathbb{P}(N^- \leq k)$$

D'autre part, on suppose que $N^{'} \leq k$. Alors, dans le modèle original, les chaînes commençant depuis les états $1, \ldots, r$ au temps $k - N^{'}$ coalesceront au temps k. On en déduit donc $N^+ \leq k$ (voir figure suivante). On a alors que $N^{'} \leq k$ implique $N^+ \leq k$, puis :

$$\mathbb{P}(N^- \leq k) = \mathbb{P}(N^{'} \leq k) \leq \mathbb{P}(N^+ \leq k)$$

Grâce aux deux inégalités démontrées précédemment, on en déduit le résultat.

□

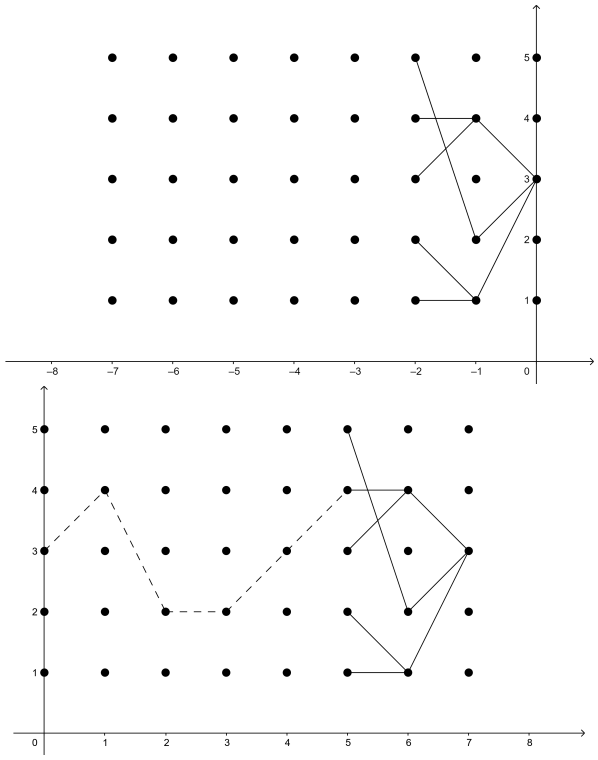


FIGURE 4 – $N^{'} \leq k$ implique $N^{+} \leq k$

On définit à présent la variable aléatoire suivante :

$$Z = Z(i) = X_0^{-N^-}(i)$$

(la variable aléatoire Z est indépendante de $i \in E$; dans la figure 1, on a $Z = 2$).
On a le théorème suivant :

Théorème 12 (Baccelli/Brémaud, p.112, théorème 2.5.1)**.** *Le temps de coalescence backwards N^- est presque sûrement fini. De plus, la variable aléatoire Z admet π comme distribution.*

Démonstration. Puisque, pour tout $k \in \mathbb{N}$, $\mathbb{P}(N^- \leq k) = \mathbb{P}(N^+ \leq k)$ (d'après le résultat précédent), le fait que N^- soit fini provient directement du fait que N^+ soit fini (théorème 10).
D'autre part, puisque pour $n \geq N^-$, on a $X_0^{-n}(i) = Z$,

$$\mathbb{P}(Z = j) = \lim_{n\to\infty} \mathbb{P}(X_0^{-n}(i) = j) = \lim_{n\to\infty} p_{ij}(n) = \pi(j),$$

où la dernière égalité provient du théorème ergodique pour les chaînes de Markov. □

Exemple : distribution stationnaire et temps de coalescence forwards

Le théorème précédent montre qu'en utilisant le temps de coalescence backwards, on obtient une réalisation de la loi stationnaire par la méthode CFTP. La question est alors la suivante : pourquoi ne pas utiliser le temps de coalescence forwards ? La réponse à cette question s'obtient en étudiant l'exemple simple suivant :
On considère $\{X_n\}$ la chaîne de Markov à deux états de matrice de transition $P = \begin{pmatrix} 0 & 1 \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$.
On remarque que cette chaîne a pour loi stationnaire $\pi = (\frac{1}{3}\ \frac{2}{3})$.
On remarque ensuite que, pour N^+ le temps de coalescence forwards, on a $X_0^{N^+}(1) = X_0^{N^+}(2) = 2$ presque sûrement. En effet, par l'absurde, si les deux chaînes issues des états 1 et 2 s'étaient rejointes au temps N^+ à l'état 1, alors on aurait deux cas possibles :

- Au temps $N^+ - 1$, la chaîne issue de l'état 1 se déplace à l'état 1 au temps suivant : une absurdité par la structure de la chaîne ($P_{1,1} = 0$)
- Au temps $N^+ - 1$, les deux chaînes ont déjà coalescé, d'où la coalescence au temps N^+ : une absurdité par la définition de N^+

Cependant le temps de coalescence forwards n'est pas une cause perdue : certains modèles permettent d'utiliser la méthode CFTP en conjugaison avec celui-ci pour obtenir une réalisation de la loi stationnaire, comme le montre l'exemple suivant :

On considère $\{X_n\}$ la chaîne de Markov à deux états de matrice de transition $P = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$.
On remarque facilement que la loi stationnaire est $\pi = (\frac{1}{2}\ \frac{1}{2})$, puis que, au temps N^+ :

$$\mathbb{P}(X_0^{N^+}(1) = X_0^{N^+}(2) = 1) = \mathbb{P}(X_0^{N^+}(1) = X_0^{N^+}(2) = 2) = \frac{1}{2}$$

car les deux évènements sont équiprobables et que l'on sait que l'on coalesce forcément pour la première fois au temps N^+.

La méthode présentée précédemment est particulièrement coûteuse lorsque l'espace d'états est grand (puisqu'il faut faire évoluer des trajectoires à partir de chaque état). On propose une mise en place de l'algorithme sous R en annexe, partie 3. Cependant, si la méthode ne requerrait la coalescence que de 2 chaînes et non plus r, celle-ci serait bien plus efficace. Propp et Wilson ont montré comment utiliser cela dans le cas monotone suivant.
On suppose qu'il existe une relation d'ordre partiel sur l'espace d'état E que l'on notera par $\preceq$.
On suppose ensuite que la fonction de mise à jour préserve cette relation d'ordre, ie,

$$i \preceq j \Rightarrow h(i,\xi) \preceq h(j,\xi) \quad \forall \xi$$

On suppose ensuite que $1 \preceq 2 \preceq \ldots \preceq r$ et on considère le modèle avec une seule mise à jour commune (ie, $\xi_n(i) = \xi_n$ pour tout n et tout i). On définit à présent le temps de coalescence backwards monotone :

$$M = \inf\left\{n \geq 1 | X_0^{-n}(1) = X_0^{-n}(r)\right\}$$

($= +\infty$ si, pour tout $n \geq 1$ la condition n'est pas satisfaite). La procédure de samplage correspondant est appelée algorithme monotone de Propp-Wilson.

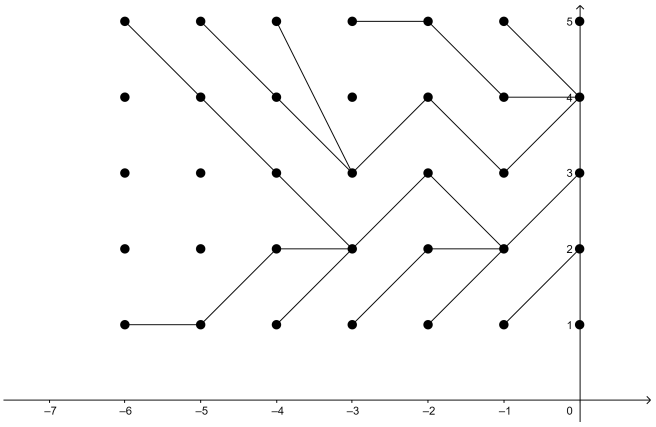


FIGURE 5 – Algorithme monotone de Propp-Wilson ; ici $M = 6$

Théorème 13 (Baccelli/Brémaud, p.113, théorème 2.5.2)**.** *Le temps de coalescence backwards monotone M est presque sûrement fini. De plus, les variables aléatoires $X_0^{-M}(1) = X_0^{-M}(r)$ sont de distribution π.*

Démonstration. On appelle N^+ le temps de couplage forwards des deux chaînes $\{X_n^0(1)\}$ et $\{X_n^0(r)\}$. La preuve que N^+ est fini est immédiate dans notre cas puisque N^+ est majoré par le premier temps $n \geq 0$ tel que $X_n^0(r) = 1$ qui est fini. Ayant cela, la preuve se déroule comme dans celle du théorème 12 grâce à la propriété de funneling de la chaîne, ie, pour tout $n \in \mathbb{Z}$, pour tout $i \in E$, $X_n^k(1) \preceq X_n^k(i) \preceq X_n^k(r)$ (voir notamment la figure précédente). □

3.2 CFTP : approche par blocs (Huber)

On considère un état stationnaire X. Puis, pour t fixé, soit $Y = \phi_t(X, U)$, où ϕ est la fonction de mise à jour de la chaîne considérée et $U \in [0,1]^t$ sont les choix aléatoires effectués. Alors Y est aussi stationnaire, car c'est la composition de t états stationnaires. La sortie de l'algorithme de CFTP sera toujours Y. On considère ensuite
$W = (U_1, U_2, \ldots, U_t)$ uniforme sur $[0,1]^t$ puis un ensemble mesurable $A \subset [0,1]^t$. Alors soit $W \in A$, soit $W \notin A$. D'où :

$$Y = \phi_t(X, W)\mathbf{1}(W \in A) + \phi_t(X, W)\mathbf{1}(W \notin A) \tag{2}$$

Supposons qu'on puisse trouver un ensemble A tel que lorsque $W \in A$, $\phi(X, W)$ ne dépende pas de X. C'est à dire, dès lors que $W \in A$, on ait $\phi(x, w) = \phi(x', w)$ pour tout $x, x' \in \Omega$. Une telle chaîne à "oublié" son point de départ lors de son déplacement. Si cela arrive, il n'y a alors pas besoin de connaître la valeur de X pour obtenir Y dans la formule précédente.
Pour voir cela, on considère l'exemple suivant :

Exemple

Soit $\Omega = \{0, 1, 2\}$. Soit une fonction de mise à jour ϕ telle que :

$$\phi(x, U) = x + \mathbf{1}(x < 2, U > \frac{1}{2}) - \mathbf{1}(x > 0, U \leq \frac{1}{2}).$$

Soient $\{X_t\}$ et $\{Y_t\}$ deux chaînes de Markov ayant ϕ comme fonction de mise à jour et telles que $X_0 = 0$ et $Y_0 = 2$. On suppose que $U_0 = 0.64, U_1 = 0.234$ et $U_2 = 0.1$. On a donc les trajectoires suivantes : $(X_0, X_1, X_2, X_3) = (0, 1, 0, 0)$ et $(Y_0, Y_1, Y_2, Y_3) = (2, 2, 1, 0)$.
Les deux chaînes ont donc fusionné au temps $t = 3$.
Pour cet exemple, on a donc :

$$\phi_3(0, W) = \phi_3(1, W) = \phi_3(2, W) = 0$$

D'où : $\phi_3(\{0, 1, 2\}, W) = \{0\}$
Supposons que les 3 premiers pas étaient (haut,bas,bas). Cette suite de mouvements correspond à l'ensemble des valeurs appartenant à $A_1 = (\frac{1}{2}, 1] \times [0, \frac{1}{2}] \times [0, \frac{1}{2}]$. Si $W \in A_1$, alors $\phi_3(\{0, 1, 2\}, W) = \{0\}$.
Une autre suite de mouvements valide est (bas,bas,bas) ce qui correspond à $A_2 = [0, \frac{1}{2}] \times [0, \frac{1}{2}] \times [0, \frac{1}{2}]$. Bien sûr, si A_1 et A_2 fonctionnent, alors $A_3 = A_1 \cup A_2$ fonctionne aussi. D'où $\phi_3(\{0, 1, 2\}, W) = \{0\}$ pour tout $W \in A_3$.
Le but ici est de montrer que A n'a pas besoin d'être exactement l'ensemble de tous les mouvements qui coalesceront en un point. Cependant, plus A sera grand, plus la probabilité que W soit dans A sera grande.
Retour à l'équation (2). Dans le premier terme, la valeur de X n'est pas importante, alors on peut écrire : $\phi_t(X, W)\mathbf{1}(W \in A) = \phi_t(x_0, W)\mathbf{1}(W \in A)$, où x_0 est un élément arbitraire de l'espace d'états. On a alors :

$$Y = \phi_t(x_0, W)\mathbf{1}(W \in A) + \phi_t(X, W)\mathbf{1}(W \notin A)$$

Donc lorsque $W \in A$, il n'y a pas besoin de connaître la valeur de X afin de calculer Y. Cependant, lorsque $W \notin A$, on doit évaluer $\phi_t(X, W)$ pour obtenir Y. L'idée principale du CFTP est de trouver $X \sim \pi$ en appelant récursivement le CFTP (c'est la partie "from the past" de l'algorithme) puis en calculant Y comme précédemment. Alors par définition de Y, on générera éventuellement un $W \in A$ tel qu'on ait pas besoin de connaître X, et il ne restera plus qu'à dérouler tous les appels récursifs effectués.
Pour comprendre cela en pratique, on altère légèrement notre notation en ajoutant un indice de temps. Soit $Y_0 = Y$, $W_0 = W$, et $Y_{-1} = W$. Avec cette notation :

$$Y_0 = \phi_t(x_0, W_0)\mathbf{1}(W_0 \in A) + \phi_t(Y_{-1}, W_0)\mathbf{1}(W_0 \notin A)$$

Donc si $W_0 \in A$, on s'arrête, sinon, on doit générer Y_{-1}. Ce sera fait de la même manière que lorsque l'on à généré Y_0 :

$$Y_{-1} = \phi_t(x_0, W_{-1})\mathbf{1}(W_{-1} \in A) + \phi_t(Y_{-2}, W_1)\mathbf{1}(W_1 \notin A)$$

où $W_{-1} \sim Unif([0,1])^t$ et est indépendant de W_0.
En général,

$$Y_{-i} = \phi_t(x_0, W_{-i})\mathbf{1}(W_{-i} \in A) + \phi_t(Y_{-i-1}, W_{-i})\mathbf{1}(W_{-i} \notin A)$$

Théorème 14 (Théorème CFTP (Coupling From The Past), Huber, p.46, théorème 3.1)**.** *On suppose que ϕ est une fonction de mise à jour pour une chaîne de Markov définie sur Ω, telle que pour $U = (U_0, U_{-1}, \ldots, U_{-t-1}) \sim Unif([0,1]^t)$ on ait :*

- *Pour $Y \sim \pi, \phi(Y, U_0) \sim \pi$*
- *Il existe un ensemble $A \subseteq [0,1]^t$ tel que $\mathbb{P}(U \in A) > 0$ et $\phi_t(\Omega, A) = \{x\}$ pour un certain $x \in \Omega$*

Posons alors, pour tout $x \in \Omega$,

$$\begin{aligned} Y_0 = \phi_t(x_0, U_0)\mathbf{1}(U_0 \in A) + \phi_t(\phi_t(x_0, U_{-1}), U_0)\mathbf{1}(U_{-1} \in A) + \\ \phi_t(\phi_t(\phi_t(x_0, U_{-2}), U_{-1}), U_0)\mathbf{1}(U_{-2} \in A) + \ldots \end{aligned}$$

Alors $Y_0 \sim \pi$.

Démonstration. Soit $x_0 \in \Omega$. Le résultat est immédiat en utilisant le théorème fondamental de la simulation parfaite, en posant $g(U) = \phi_t(x_0, U)$,$b(U) = \mathbf{1}(U \in A)$, et $f(X, U) = \phi_t(X, U)$. □

L'idée clé qu'ont eu Propp et Wilson est qu'il n'est pas nécessaire de connaître tous les termes de la suite pour trouver Y_0. Il suffit en effet de connaître $U_0, \ldots, U_{-T}$, où $U_T \in A$. Tant que $\mathbb{P}(U \in A) > 0$, alors T suivra une loi géométrique de paramètre $\mathbb{P}(U \in A)$. Le pseudo-code suivant accomplit alors cette tâche :

Coupling-from-the-past Sortie : $Y \sim \pi$

1: Tirer $U \sim Unif([0,1]^t)$
2: **if** $U \in A$ **then**
3: Rendre $\phi_t(x_0, U)$ (où x_0 est un élément arbitraire de Ω)
4: **else**
5: $X \leftarrow$ Coupling-from-the-past
6: Rendre $\phi_t(X, U)$
7: **end if**

Cet algorithme est un algorithme récursif, autorisé à s'appeler à nouveau en ligne 5. Lorsque cela arrive, les valeurs de U ne sont pas passées en paramètre : le nouvel appel génère en ligne 1 ses propres valeurs de U indépendantes des précédentes. C'est une étape importante, car sinon le théorème fondamental de la simulation parfaite ne s'appliquerait pas. On propose une mise en place de cet algorithme sur un exemple simple en annexe, partie 3.
On considère à présent le temps d'exécution de l'algorithme mesuré par le nombre d'appels à ϕ.

Lemme 9 (Huber, Adaptation du lemme 3.1, p.48)**.** *Le nombre d'appels à ϕ dans l'algorithme Coupling-from-the-past est en moyenne $\frac{t}{\mathbb{P}(U \in A)}$.*

Démonstration. Soit $T = \inf\{\tau \geq 1 | U_\tau^t \in A\}$. T suit par définition la loi géométrique de paramètre $\mathbb{P}(U \in A)$
Alors, l'algorithme s'arrête au temps T après être passé $T-1$ fois dans la partie ELSE puis une fois dans la première partie.
Dans la partie ELSE, l'algorithme fait appel t fois à ϕ, d'où : $(T-1) * t$ appels à ϕ.
La première partie fait aussi t appels à ϕ. En sommant, on obtient le nombre total d'appels : $T * t$.
Or, on connaît la loi de T, donc, en prenant l'espérance, on obtient :

$$\mathbb{E}[T * t] = t * \mathbb{E}[T] = \frac{t}{\mathbb{P}(U \in A)}$$

□

Pour garder le temps d'exécution suffisamment bas, il est important que t soit suffisamment grand pour que $\mathbb{P}(U \in A)$ soit raisonnablement élevé. Dès lors que cette condition est achevée, t devra être le plus petit possible, afin de garder un nombre de mise à jour faible.
Cependant, il existe un moyen de contourner ce problème.

Variation de la taille des blocs

Lors de la construction du CFTP, une fonction de mise à jour à été composée avec elle-même t fois. On appelle cet ensemble de t pas un bloc.

On suppose qu'il y ait un seuil t' tel que $\mathbb{P}(U \in A) = 0$ si $t < t'$. En pratique, un tel seuil est très difficile à calculer : il est donc commun d'utiliser des blocs de taille qui augmente afin d'être sûr que t dépassera éventuellement ce seuil.
En général, il n'y a aucune raison pour que le même nombre de pas soit utilisé pour chaque bloc.
Le théorème fondamental de la simulation parfaite s'adapte facilement au cas où b, g et f changent à chaque étape de la récursion.

Théorème 15 (Théorème fondamental de la simulation parfaite (seconde forme), Huber, p.48, théorème 3.2)**.** *On suppose que pour $U_1, U_2, \ldots$ iid, on ait des suites de fonctions calculables $\{b_t\}$,$\{g_t\}$ et $\{f_t\}$ telles que chaque $\{b_t\}$ ait pour image $\{0,1\}$ et que $\prod_{t=1}^{\infty} \mathbb{P}(b_t(U) = 0) = 0$. Soit X la variable aléatoire telle que, pour tout t :*

$$X \sim b_t(U)g_t(U) + (1 - b_t(U))f_t(X, U)$$

Soit $T = \inf\{t : b_t(U_t) = 1\}$. Alors,

$$Y = f_0(\ldots f_{T-2}(f_{T-1}(g_T(U_T), U_T), U_{T-1}), \ldots, U_1)$$

a la même distribution que X et $\mathbb{E}[T] = \frac{1}{\mathbb{P}(b_t(U_t)=1)}$

Démonstration. La preuve est presque identique à celle du théorème fondamental de la simulation parfaite présentée dans le chapitre 1. □

On utilise cette forme du théorème fondamental de la simulation parfaite pour obtenir le résultat suivant :

Théorème 16. *Soit $t(0) < t(1) < t(2) < \ldots$ des entiers positifs. Soit $W_0, W_1, \ldots$ indépendants tels que $W_{-i} \sim Unif([0,1]^{t(i)})$, et $A_{t(i)}$ un ensemble mesurable de $[0,1]^{t(i)}$ tel que $\phi_{t(i)}(\Omega, W_{-i}) = \{y\}$. On suppose que $\prod_{i=1}^{\infty} \mathbb{P}(W_{-i} \in A) = 0$. Alors pour tout $x_0 \in \Omega$,*

$$\begin{aligned} Y_0 = {} & \phi_{t(0)}(x_0, W_0)\mathbf{1}(W_0 \in A_{t(0)}) + \phi_{t(0)}(\phi_{t(1)}(x_0, W_{-1}), W_0)\mathbf{1}(W_{-1} \in A_{t(1)}) \\ & + \phi_{t(0)}(\phi_{t(1)}(\phi_{t(2)}(x_0, W_{-2}), W_{-1}), W_0)\mathbf{1}(W_{-2} \in A_{t(2)}) + \ldots \end{aligned}$$

est tel que $Y_0 \sim \pi$.

Un choix usuel est de poser $t(i) = 2 * t(i-1)$ afin que le nombre de pas double à chaque étape de l'algorithme. Si le t initial est $t(0) = 1$, alors doubler le nombre de pas à chaque itération nous permettra d'atteindre t' après un nombre logarithmique de récursions. De plus le travail total effectué dans tous les blocs sera de $1 + 2 + 4 + \ldots + t' = 2 * t' - 1$.
On utilise ce choix de $t(i)$ pour obtenir l'algorithme suivant :
On notera que le choix $t(i) = 2 * t(i-1)$ n'est pas mandatoire, d'autres choix sont valides. Seul le nombre de pas pris par l'algorithme (en moyenne) sera affecté.

Doubling-Coupling-from-the-past Entrée : t, Sortie : $Y \sim \pi$

1: Tirer $U \sim Unif([0,1]^t)$
2: **if** $U \in A_t$ **then**
3: Rendre $\phi_t(x_0, U)$ (où x_0 est un élément arbitraire de Ω)
4: **else**
5: $X \leftarrow$ Doubling-Coupling-from-the-past$(2 * t)$
6: Rendre $\phi_t(X, U)$
7: **end if**

3.3 CFTP monotone

La partie difficile permettant d'utiliser le CFTP est de créer les ensembles $A_{t(i)}$ ainsi qu'une méthode permettant de déterminer si $U \in A_{t(i)}$.
C'est cette difficulté qui empêche d'utiliser le CFTP dans toutes les applications de Monte Carlo.
On présentera ici une méthode permettant de trouver de tels $A_{t(i)}$ dans différentes applications.
L'idée est la suivante. On suppose que Ω est un espace d'états admettant un ordre partiel (voir définition 11).
Comme exemple, on considère le modèle d'Ising et deux configurations x et y. On dit que $x \preceq y$ si pour tout nœud ν on a $x(\nu) \leq y(\nu)$. Par exemple, si on a 4 nœuds, $(-1,-1,-1,1) \preceq (-1,-1,1,1)$. D'autre part, les configurations $(-1,-1,1,1)$ et $(1,1,-1,-1)$ ne sont pas comparables.
Ensuite, on supposera que Ω est doté d'un plus grand élément x_{max}, et d'un plus petit élément x_{min}. Pour continuer avec l'exemple du modèle d'Ising, la configuration n'ayant que des -1 est le plus petit élément, celle n'ayant que des 1, le plus grand.

Définition 18 (Fonction de mise à jour monotone)**.** *Soit ϕ une fonction de mise à jour pour un pas d'une chaîne de Markov. Si ϕ est telle que, pour tout $x \preceq y$, et $u \in [0,1]$, on ait $\phi(x,u) \preceq \phi(y,u)$, alors on dit que ϕ est monotone.*

On considère à présent deux chaînes de Markov $\{X_t\}$ et $\{Y_t\}$ telles que $X_0 = x_0$ et $Y_0 = y_0$ où $x_0 \preceq y_0$. On suppose qu'à chaque pas, $X_{t+1} = \phi(X_t, U_t)$ et $Y_{t+1} = \phi(Y_t, U_t)$ où $U_1, U_2, \ldots$ sont iid telles que $U_1 \sim Unif([0,1])$.
On en déduit simplement que $X_{t+1} \preceq Y_{t+1}$ pour tout t.
On suppose de plus que $x_0 = x_{min}$ et $y_0 = x_{max}$. Alors tout autre état w_0 vérifie $x_0 \preceq w_0 \preceq y_0$ et on construit alors de manière similaire une chaîne de Markov W_t qui donnera $X_t \preceq W_t \preceq Y_t$ pour tout t.
On suppose maintenant que $X_t = Y_t$. Les propriétés d'ordre partiel impliquent alors que $X_t = W_t = Y_t$. L'état de départ de W_t était arbitraire, donc pour tout état de départ $w_0 \in \Omega$, $\phi_t(w_0, u) = X_t = Y_t$
En d'autres termes, si après un nombre fixé de pas, les états extrémaux ont atteint le même état, alors tous les autres états ont été "piégés" entre ceux-ci et ont donc aussi atteint le même état.
On obtient alors une variante importante du CFTP : le CFTP monotone.

Monotonic-Coupling-from-the-past Entrée : t Sortie : Y

1: Tirer $U \sim Unif([0,1]^t)$
2: Soit $X_t \leftarrow \phi_t(x_{max}, U)$ et $Y_t \leftarrow \phi_t(x_{min}, U)$
3: **if** $X_t = Y_t$ **then**
4: Rendre X_t
5: **else**
6: Tirer $X \leftarrow$ Monotonic-Coupling-from-the-past$(2 * t)$
7: Rendre $\phi_t(X, U)$
8: **end if**

3.3.1 Modèle D'Ising

On considère la mise à jour de Gibbs pour le modèle d'Ising de la section 1.4.1, pour $\mu = 0$
Lors de la mise à jour, un nœud ν était choisi uniformément aléatoirement et $U \sim Unif([0,1])$. On note N_c le nombre de voisins de ν de label c. Si $U < \exp(N_1\beta)/[\exp(N_1\beta + \exp(N_{-1}\beta)]$, alors le nœud devenait de label 1, sinon de label -1.
Afin de pouvoir utiliser cette mise à jour pour la simulation parfaite, on doit l'écrire comme fonction de mise à jour, c'est-à-dire, l'état actuel et les choix aléatoires doivent être des paramètres d'entrée.

Ising-Gibbs Entrée : $x \in \{-1,1\}^V, u \in [0,1], \nu \in V$ Sortie : $x \in \{-1,1\}^V$

1: $N_1 \leftarrow$ nombre de voisins de ν de label 1 dans x
2: $N_{-1} \leftarrow$ nombre de voisins de ν de label -1 dans x
3: **if** $U < \exp(N_1\beta)/[\exp(N_1\beta + \exp(N_{-1}\beta)]$ **then**
4: $x(\nu) \leftarrow 1$
5: **else**
6: $x(\nu) \leftarrow -1$
7: **end if**

Lemme 10 (Huber, p.51, propriété 3.1)**.** *La fonction créée précédemment est monotone lorsque* $\beta > 0$

Démonstration. Soit $x \preceq y$. Soit ν un nœud du graphe considéré et la fonction $N_c(z)$ notant le nombre de voisins de ν de label c dans la configuration z. Alors tout voisin de ν dans x de label 1 est aussi de label 1 dans y, donc $N_1(x) \leq N_1(y)$. De manière similaire, on a $N_{-1}(x) \leq N_{-1}(y)$.
Soit $f(n_1, n_{-1}) = \exp(N_1(x)\beta)/[\exp(N_1(x)\beta + \exp(N_{-1}\beta)]$. Puisque $\beta > 0$, $f(n_1, n_{-1})$ est croissant en n_1 et décroissant en n_{-1}. Donc si $U < f(N_1(x), N_{-1}(x))$ alors on aura $U < f(N_1(y), N_{-1}(y))$. C'est-à-dire, si $x(\nu)$ est changé en 1, alors $y(\nu)$ aussi.
De même, si $U > f(N_1(y), N_{-1}(y))$, alors $U > f(N_1(x), N_{-1}(x))$. C'est-à-dire, si $y(\nu)$ devient 0, alors $x(\nu)$ aussi.

Enfin, si $f(N_1(x), N_{-1}(x)) < U < f(N_1(y), N_{-1}(y))$, alors seul le nœud $x(\nu)$ est changé en -1, $y(\nu)$ devient 1, l'ordre n'est alors pas changé.
Donc, peu importe le choix de ν et U dans la mise à jour, si $x \preceq y$, alors $\phi(x, \nu, U) \preceq \phi(y, \nu, U)$. □

Cela signifie que le CFTP monotone peut-être utilisé pour générer un échantillon aléatoire de la distribution stationnaire. On propose une mise en place du pseudo-code suivant sous R en annexe, partie 3.

Monotonic-Ising-Gibbs Entrée : t Sortie :$X \sim \pi$

Tirer $(U_1, \ldots, U_t) \leftarrow Unif([0,1]^t)$
Tirer $(V_1, \ldots, V_t) \leftarrow Unif(V^t)$
$x_{max} \leftarrow (1, \ldots, 1)$, $x_{min} \leftarrow (0, \ldots, 0)$
for i allant de 1 à T **do**
    $x_{max} \leftarrow$ Ising-Gibbs(x_{max},U_i,V_i)
    $x_{min} \leftarrow$ Ising-Gibbs(x_{min},U_i,V_i)
end for
if $x_{max} \neq x_{min}$ **then**
    $x_{max} \leftarrow$ Monotonic-Ising-Gibbs($2 * t$)
    for i allant de 1 à t **do**
        $x_{max} \leftarrow$ Ising-Gibbs(x_{max},U_i,V_i)
    end for
end if
$X \leftarrow x_{max}$

3.3.2 Modèle gas Hard-core sur graphe biparti

On considère maintenant la mise-à-jour HCGM-Gibbs de la section 1.4.4. Cette fonction de mise-à-jour n'est pas monotone en utilisant l'ordre partiel $x \preceq y \leftrightarrow (\forall \nu \in V)(x(\nu) \leq y(\nu))$.
Il est alors nécessaire de trouver une autre fonction de mise-à-jour ou un autre ordre partiel. Le problème est que la chaîne est répulsive,ie, deux nœuds voisins ne peuvent avoir le label 1. Cependant, si le graphe est biparti, il est possible de construire un nouvel ordre partiel tel que l'on ait une fonction de mise-à-jour monotone.

Définition 19 (Graphe biparti)**.** *On dit qu'un graphe est biparti lorsque l'ensemble des nœuds V peut être partitionné en deux ensembles V_1 et V_2 tels que pour toute arête e, un nœud appartienne à V_1 et l'autre à V_2.*

Pour les configurations sur graphe biparti, on utilise l'ordre partiel suivant (similaire à l'ordre partiel défini précédemment) :

$$x \preceq y \leftrightarrow (\forall \nu_1 \in V_1)(\forall \nu_2 \in V_2)(x(\nu_1) \leq y(\nu_1) \wedge x(\nu_2) \geq y(\nu_2))$$

Lemme 11 (Huber,p.52,propriété 3.2)**.** *HCGM-Gibbs est monotone relativement à l'ordre partiel ci-dessus*

Démonstration. Si $x \preceq y$ et le nœud $i \in V_2$ est choisi pour être mis-à-jour, alors si un voisin w de i est tel que $x(w) = 1$, alors $y(w) = 1$ aussi. Donc $x(i)$ et $y(i)$ resteront tous deux à 0. La seule manière pour que $y(i)$ soit changé en 1 est si $y(w) = 0$ pour tout voisin w de i. Dans ce cas, tout voisin vérifie aussi $x(w) = 0$. D'où : $x(i) = y(i) = \mathbf{1}(U < \lambda/(\lambda + 1))$.
L'analyse pour $i \in V_1$ est similaire. □

L'état "le plus grand" est alors celui dont tous les nœuds de V_2 sont de label 1 et tous les nœuds de V_1 sont de label 0.
L'état "le plus petit" voit ses labels inversés par rapport au plus grand. On propose une mise en place sous R pour ce modèle en annexe, partie 3.

3.3.3 Monotonie et temps de mélange

Une des raisons pour lesquelles la monotonie est importante est que le CFTP monotone ne prendra pas plus de temps à fonctionner que le temps de mélange de la chaîne de Markov.

Définition 20 (Chaîne en ordre partiel)**.** *Une chaîne de longueur l en ordre partiel est un sous-ensemble de $l + 1$ éléments tels que $x_0 \prec x_1 \prec x_2 \prec \ldots \prec x_l$*

On présente ensuite une propriété importante de la distance en variation totale : celle-ci permet la représentation de deux variables aléatoires en utilisant une troisième variable aléatoire commune.

Lemme 12 (Adaptation de la propriété 3.3, p.53, Huber)**.** *On suppose que $d_{TV}(X, Y) = d$, où X, Y sont des variables aléatoires de densités respectives f_X et f_Y par rapport à la mesure μ sur Ω. Alors il existe des variables aléatoires W_1, W_2, W_3 et $B \sim Bern(d)$ indépendantes telles que :*

$$X \sim (1 - B)W_1 + BW_2$$
$$Y \sim (1 - B)W_1 + BW_3$$

Démonstration. On procède ici par disjonction de cas sur d.

- $d = 0$. Alors les variables aléatoires X et Y sont égales, on pose $W_1 = X(= Y)$ et W_2, W_3 ont n'importe quelle distribution (car $B = 0$).
- $d = 1$. Il suffit alors de poser $W_2 = X$ et $W_3 = Y$ puisque $B = 1$.
- $0 < d < 1$. On pose alors $g(s) = \min(f_X(s), f_Y(s))$ puis à W_1 on donne la densité $g/\int_\Omega g d\nu$,à W_2 la densité $[f_X(s) - g(s)]/\int_\Omega (f_X(r) - g(r))d\nu(r)$ et à W_3 la densité $[f_Y(s) - g(s)]/\int_\Omega (f_Y(r) - g(r))d\nu(r)$.
  Il reste à voir maintenant que $\int_\Omega g d\nu = 1 - d$. Pour le montrer, on rappelle d'abord que la distance en variation totale est $d = \sup_D |\mathbb{P}(Y \in D) - \mathbb{P}(X \in D)|$. Puisque $|\mathbb{P}(Y \in D) - \mathbb{P}(X \in D)| = |\mathbb{P}(Y \in D^C) - \mathbb{P}(X \in D^C)|$, le supremum peut être

considéré sur les ensembles D tels que $\mathbb{P}(Y \in D) \geq \mathbb{P}(X \in D)$.
Soit à présent $C = \{s \in \Omega | f_X(s) \leq f_Y(s)\}$. Soit D un ensemble mesurable tel que $\mathbb{P}(Y \in D) \geq \mathbb{P}(X \in D)$. On a, pour toute variable aléatoire W, $\mathbb{P}(W \in D) = \mathbb{P}(W \in C) - \mathbb{P}(W \in C \cap D^C) + \mathbb{P}(W \in C^C \cap D)$.
Pour voir cela, il suffit de voir que, pour une mesure μ et des ensembles mesurables C et D :

$$\mu(D \cap C) = \mu(C \setminus (D^C \cap C)) = \mu(D \setminus (D \cap C^C))$$

puis que,

$$\mu(C) - \mu(D^C \cap C) = \mu(D) - \mu(D \cap C^C)$$

On définit ensuite, pour toute variable aléatoire W, $h(W) = \mathbb{P}(W \in C \cap D^C) - \mathbb{P}(W \in C^C \cap D)$. On a alors :

$$\mathbb{P}(Y \in D) - \mathbb{P}(X \in D) = \mathbb{P}(Y \in C) - \mathbb{P}(X \in C) - h(Y) + h(X)$$

Or,

$$-h(Y)+h(X) = [\mathbb{P}(X \in C \cap D^C) - \mathbb{P}(Y \in C \cap D^C)] + [\mathbb{P}(Y \in C^C \cap D) - \mathbb{P}(X \in C^C \cap D)]$$

De la manière dont C a été choisi, les deux termes entre crochets sont non-positifs ou nuls, on a donc :

$$0 \leq \mathbb{P}(Y \in D) - \mathbb{P}(X \in D) \leq \mathbb{P}(Y \in C) - \mathbb{P}(X \in C)$$

Puisque D était un ensemble arbitraire tel que $\mathbb{P}(Y \in D) \leq \mathbb{P}(X \in D)$, la distance de variation totale est alors $d = \mathbb{P}(Y \in C) - \mathbb{P}(X \in C)$.
On considère à présent la constante de normalisation de W_2.

$$\begin{aligned}
\int_\Omega f_X(s) - g(s) d\nu(s) &= \int_C f_X(s) - g(s) d\nu(s) + \int_{C^C} f_X(s) - g(s) d\nu(s) \\
&= 0 + \int_{C^C} f_X(s) - f_Y(s) d\nu(s) \\
&= \mathbb{P}(X \in C^C) - \mathbb{P}(Y \in C^C) = \mathbb{P}(Y \in C^C) - \mathbb{P}(X \in C^C) = d
\end{aligned}$$

La densité de $(1-B)W_1 + BW_2$ est :

$$(1-d)\frac{\min(f_X(s), f_Y(s))}{1-d} + d\frac{f_X - \min(f_X(s)(s), f_Y(s))}{d} = f_X(s)$$

la même densité que X. Un résultat similaire vaut pour Y.

□

Nous utilisons ce résultat dans le théorème suivant :

Théorème 17 (Propp et Wilson (Huber, p.54, théorème 3.4)**.** *Pour un ensemble discret Ω, soit l la longueur de la chaîne la plus longue en ordre partiel de Ω. Soit p la probabilité qu'un appel à Monotonic-coupling-from-the-past(t) doive s'appeler récursivement. Notons $d_t = \sup_{x\in\Omega} d_{TV}([X_t|X_0 = x], \pi)$. On a alors :*

$$\frac{p}{l} \leq d_t \leq p$$

Démonstration. Le fait que $d_t \leq p$ provient du Lemme de couplage (voir chapitre 1 théorème 4).
On va montrer ici que $\frac{p}{l} \leq d_t$. Pour un élément $x \in \Omega$, soit $h(x)$ la longueur de la chaîne la plus longue ayant pour plus grand élément x. Si X_t est la chaîne commençant en x_{min}, et Y_t celle commençant en x_{max}, alors on aura toujours $X_t \preceq Y_t$. On notera que si $X_t \prec Y_t$, alors $h(X_t) + 1 \leq h(Y_t)$, mais si $X_t = Y_t$ alors $h(X_t) = h(Y_t)$. On en conclut donc que $h(Y_t) - h(X_t)$ est une variable aléatoire entière positive ou nulle qui vaut au moins 1 lorsque $X_t \neq Y_t$. On obtient, d'après l'inégalité de Markov :

$$p = \mathbb{P}(X_t \neq Y_t) \leq \mathbb{E}[h(Y_t) - h(X_t)]$$

En utilisant le lemme 12, on sait qu'il existe W_1, W_2, W_3 et $B \sim Bern(d_t)$ indépendants tels que $X_t \sim (1 - B)W_1 + BW_2$ et $Y_t \sim (1 - B)W_1 + BW_3$. D'où :

$$p \leq \mathbb{E}[h(Y_t) - h(X_t)] = \mathbb{E}[B(h(W_2) - h(W_3))] = d_t l$$

□

La distance de variation totale tend à décroître très rapidement car c'est un exemple de fonction sous-multiplicative.

Lemme 13. *La distance de variation totale d_t vérifie $d_{t+s} \leq d_t d_s$.*

La preuve comprend l'utilisation du lemme de couplage cité dans le théorème 17. La propriété énoncée permet de montrer que la taille d'un bloc dans le CFTP monotone nécessaire à la coalescence n'est pas plus grand que le temps que met la chaîne de Markov à commencer à se mélanger.

Lemme 14 (Huber, p.54, lemme 3.3)**.** *On suppose que $d_t \leq e^{-1}$. Alors pour un bloc de taille $t[k + ln(l)]$, la probabilité p qu'un bloc ne coalesce pas est au plus e^{-k}.*

Démonstration. Par la propriété de sous-multiplicativité, $d_{t[k+ln(k)]} \leq [e^{-1}]^{k+ln(l)} = l^{-1}e^{-k}$. En utilisant le théorème précédent cela donne : $p \leq ll^{-1}e^{-k} = e^{-k}$. □

Lemme 15 (Huber, p.54, lemme 3.4)**.** *On suppose que dans le CFTP monotone lancé pour un bloc de taille t, on ait $\mathbb{P}(X_t \neq Y_t) \leq e^{-1}$. Alors le nombre moyen de pas de la chaîne de Markov utilisé par Monotonic-Coupling-from-the-past(t) est au plus de* $19.2t$.

Démonstration. Un appel à Monotonic-Coupling-from-the-past avec pour entrée s requiert au plus $2s$ pas de la chaîne de Markov : s pas pour déterminer si $X_t = Y_t$ puis s

autres pas après l'appel récursif si $X_t = Y_t$
Notons R_k l'évènement suivant : le $k$ème appel à Monotonic-Coupling-from-the-past ne vérifie pas $X_t = Y_t$. Si le premier appel utilise un seul pas et que le nombre de pas double à chaque appel, alors après $\lceil log_2(t) \rceil$ appels, le prochain appel a un paramètre d'au moins t, et n'échoue donc qu'avec probabilité au plus e^{-1}. D'où :

$$\begin{aligned}
\mathbb{E}[2S] &= \sum_{k=1}^{\infty} 2 * 2^k \mathbb{P}(R_1 R_2 \dots R_{k-1}) \\
&\leq 2 * (2t) + \sum_{k=\lceil log_2(t) \rceil + 1}^{\infty} 2^{k+1} (e^{-1})^{k-1-\lceil log_2(t) \rceil} \\
&\leq 4t + 4t/(1 - 2/e) \leq 19.2t
\end{aligned}$$

□

Comme exemple, considérons le modèle d'Ising. La taille maximale d'une chaîne est $\#V$, le nombre de nœuds du graphe. Donc, si t est le temps nécessaire pour que la distance de variation totale descende en dessous de e^{-1}, alors $O(tln(\#V))$ pas sont nécessaires en moyenne pour générer une simulation exacte du modèle. D'autre part, o(t) pas sont aussi nécessaires pour générer une sortie avec la simulation exacte : le CFTP ne fait donc pas converger la chaîne de Markov plus vite qu'il ne faut.
La partie importante du CFTP ici est que la connaissance du t pour lequel la distance de variation totale est au plus e^{-1} n'est pas nécessaire pour utiliser la procédure. Lorsque l'algorithme s'arrête, le résultat est garanti d'avoir été généré selon la distribution ciblée.

3.4 Inconvénients du CFTP

Bien que le CFTP soit un outil puissant pour créer des algorithmes de simulation exacte, il a quand même des défauts. Les deux principaux problèmes sont la non-interruptibilité et la lecture double.

3.4.1 Non-interruptibilité

Considérons d'abord un exemple. On veut générer une loi géométrique à partir de $B_1, B_2, \dots$ des réalisations de $Bern(p)$, puis en posant $T = \inf\{t | B_t = 1\}$ et
$f_T(B_1, \dots, B_T) = T$. Alors $T \sim Geom(p)$. Bien sûr ici T et f_T ne sont pas indépendants puisqu'ils sont de même valeur. Donc, d'après la définition 5 du chapitre 1, cet algorithme est non-interruptible.
On considère ce problème du point de vue d'une personne souhaitant simuler ce problème. Peut-être que celle-ci, en mettant en place ce problème veut inconsciemment que si l'algorithme prend trop de temps (ou de pas, par exemple, 5 millions de pas), alors la personne fait preuve d'impatience et interrompt l'algorithme.
Cette décision est en fait une partie non-reconnue de l'algorithme, et signifie que l'on

obtient pas d'échantillon suivant une loi $Geom(p)$ mais plutôt une loi $Geom(p)$ conditionné à être dans $\{1, \ldots, 5 * 10^6\}$. Dans notre cas, c'est peut-être anodin, sauf si p est très petit. Le potentiel est là, et la volonté de la personne à vouloir arrêter l'algorithme fait que celui-ci n'est plus un algorithme de simulation exacte.
On donne à présent un exemple d'algorithme interruptible. Considérons la méthode de rejet permettant de tirer selon une mesure ν sur un ensemble A sachant que l'on sait tirer selon la mesure ν sur B, et sachant que A est contenu dans B.

Lemme 16 (Huber, p.58, lemme 3.5)**.** *On suppose que ν est une mesure finie sur B, que $A \subset B$ et $\nu(A) > 0$. Pour des variables iid$X_1, X_2, \ldots \sim \nu(B)$ et $T = \inf\{t | X_t \in A\}$, on a que T et $f_T(X_1, \ldots, X_T) = X_T$ sont des variables aléatoires indépendantes. Et donc, la méthode de rejet est un algorithme interruptible.*

Démonstration. Soit C un sous-ensemble mesurable de A et $i \in \{1, 2, \ldots\}$. Alors :

$$\begin{aligned}
\mathbb{P}(X_T \in C, T = i) &= \mathbb{P}(X_1, \ldots, X_{i-1} \notin C, X_i \in C) \\
&= \left(1 - \frac{\nu(A)}{\nu(B)}\right)^{i-1} \left(\frac{\nu(C}{\nu(B)}\right) \\
&= \left(1 - \frac{\nu(A)}{\nu(B)}\right)^{i-1} \left(\frac{\nu(A)}{\nu(B)}\right) \left(\frac{\nu(C)}{\nu(A)}\right) \\
&= \mathbb{P}(X_1, \ldots, X_{i-1} \notin C, X_i \in A)\mathbb{P}(X_T \in C) \\
&= \mathbb{P}(T = i)\mathbb{P}(X_T \in C)
\end{aligned}$$

où $\mathbb{P}(X_T \in A) = \frac{\nu(A)}{\nu(B)}$ est le théorème 6 du chapitre 2.

□

L'avantage d'un algorithme interruptible est que l'utilisateur peut l'arrêter puis le relancer sans changer la sortie de l'algorithme. L'utilisateur n'a donc pas à se soucier de possibles limitations (connues ou non) sur le temps de calcul de l'algorithme.
Le problème est le suivant : en général, le CFTP est non-interruptible.
On rappelle d'abord que le CFTP décrit la distribution ciblée π comme un mélange de deux autres distributions. Soit A l'évènement tel que si $U \in A$, alors $\phi(x, U) = \{y\}\ \forall x \in \Omega$. Alors, pour $Y \sim \pi$ et pour tout ensemble mesurable C,

$$\mathbb{P}(Y \in C) = \mathbb{P}(Y \in C|A)\mathbb{P}(A) + \mathbb{P}(Y \in C|A^C)\mathbb{P}(A^C)$$

Lorsque que A se réalise, $\phi(x, U) = \{y\}$, et on rend y. Lorsque A^C se réalise, on utilise une récursion pour tirer $X \sim \pi$ puis $\phi(X, U)$ donne $[Y|A^C]$.
On suppose à présent que l'utilisateur n'a pas le temps d'effectuer la récursion. Alors le résultat n'est pas $Y \sim \pi$ mais plutôt $[Y|A]$. Comme exemple, on considère le problème de tirer uniformément sur $\{1, 2, 3\}$ en utilisant la fonction de mise à jour de la chaîne de Markov suivante :

$$\phi_1(x, U) = x + \mathbf{1}(x < 3, U > 1/2) - \mathbf{1}(x > 1, U \leq 1/2)$$

On suppose que $\phi_2(x, U_1, U_2) = \phi_1(\phi_1(x, U_1), U_2)$,ie, ϕ_2 effectue deux pas dans la chaîne de Markov. Dans quel cas avons nous $\phi_2(\Omega, U_1, U_2) = \{y\}$? Seulement dans le cas où U_1 et U_2 tombent tout deux dans $(1/2, 1]$ ou $[0, 1/2]$. Dans le premier cas $\phi_2(\Omega, U_1, U_2) = \{3\}$ et dans le second $\phi_2(\Omega, U_1, U_2) = \{1\}$. En aucun cas avons nous $\phi_2(\Omega, U_1, U_2) = \{2\}$ donc le résultat ne peut-être uniforme sur $\{1, 2, 3\}$.
Des raisons connues (temps de calcul, temps autorisé par l'utilisateur) et des raisons inconnues (coupure de courant) signifient qu'en somme, chaque simulation est lancée avec une borne supérieure aléatoire sur le temps d'arrêt de l'algorithme. Si il y a une limite sur le temps pour lequel l'algorithme peut-être lancé, celui-ci cesse d'être un algorithme de simulation exacte.

3.4.2 Lecture double

L'autre problème du CFTP provient de sa structure. Lorsque les variables aléatoires U sont générées, si la récursion s'opère, alors ces variables u doivent être utilisées à nouveau. Donc, elle doivent en théorie être stockées, ce qui peut imposer un problème sur la mémoire.
Il y a deux moyens de pallier à ce défaut. Premièrement, on peut simplement ne garder en mémoire que la "seed" utilisée pour générer les choix alétoires.
De plus, il existe une variante du CFTP telle que les choix aléatoires n'ont pas à être stockés. Cette variante s'appelle le CFTP à lecture unique, que le livre d'Huber (Chapitre 5) aborde.

4 Chapitre 4 : Bounding chains

4.1 Qu'est-ce qu'une bounding chain ?

On suppose que les états de la chaîne de Markov sous-jacente possèdent des nœuds, eux-même ayant chacun un label.
La bounding chain (littéralement chaîne liante ou chaîne bornante) tient compte de tous les labels possibles pour chaque nœud. Cette bounding chain est liée à la chaîne de Markov sous-jacente si il est possible de s'assurer que les labels de chaque nœud de la chaîne sous-jacente tombent dans l'ensemble des labels de la bounding chain.
Soit $\Omega = C^V$ l'espace d'états de la chaîne de Markov sous-jacente. Par exemple, pour le modèle d'Ising, $C = \{-1, 1\}$ et V est la configurations des arêtes (et donc nœuds) du graphe.
À chaque nœud de V, la bounding chain contient un sous-ensemble de C qui représentent les labels permis pour la chaîne de Markov sous-jacente.
Par exemple, pour le modèle d'Ising, une bounding chain peut avoir $\{-1\}$,$\{1\}$ ou $\{-1, 1\}$ comme label pour un nœud. Le véritable challenge est de mettre à jour la bounding chain.
Soit X_t l'état de la chaîne de Markov sous-jacente au temps t et Y_t l'état de la bounding chain au temps t. Si $X_t(\nu) \in Y_t(\nu)$ pour chaque nœud ν, alors on voudra aussi que $X_{t+1}(v) \in Y_{t+1}(\nu)$ pour chaque ν. Soit 2^C l'ensemble des parties de C.

Définition 21. *On dit que $y \in (2^C)^V$ lie $x \in C^V$ si pour tout $\nu \in V$, $x(\nu) \in y(\nu)$.*

Définition 22 (Bounding chain)**.** *On dit que le processus Markovien Y_t sur $(2^C)^V$ est une bounding chain pour la chaîne de Markov $\{X_t\}$ sur C^V si il existe un couplage en X_t et Y_t tel que si Y_t lie X_t, alors Y_{t+1} lie X_{t+1}.*

La manière la plus simple de créer un tel couplage est d'utiliser une fonction de mise-à-jour.

Théorème 18 (Construction de bounding chain)**.** *Soit $X_{t+1} = \phi(X_t, U_t)$ pour tout t, où $U_0, U_1, \ldots$ sont iid de loi $Unif([0, 1])$. On créé alors $y' = \phi_t^{BC}(y, u)$ comme suit. Pour tout $\nu \in V$, on pose :*

$$y'(\nu) = \{c | \exists x \text{ lié par } y \text{ tel que } \phi_t(x, u) \text{ ait le label } c \text{ au nœud } \nu\}$$

Alors ϕ^{BC} donne une bounding chain pour X_t

En d'autres termes, on considère tous les états possibles x qui sont liés par l'état y. On effectue un pas dans chacun de ces chaînes, l'ensemble des labels résultants pour ν devient le nouvel état $y(\nu)$.
Lorsque la fonction de mise-à-jour a été créée, le CFTP peut être utilisé comme dans l'algorithme ci-après :
Le cadre général maintenant mis en place, on va considérer différents exemples.

Bounding-chain-cftp Entrée : t Sortie : X

Tirer $U_1, U_2, \ldots, U_t$
Pour tout $\nu \in V$, poser $Y(\nu) = C$
for i de 1 à t **do**
  $Y \leftarrow$ Bounding-chain-update(Y,U_i)
end for
if $\#Y(\nu) > 1$ pour un certain ν **then**
  $X \leftarrow$ Bounding-chain-cftp($2t$)
  Pour tout ν, $Y(\nu) \leftarrow \{X(\nu)\}$
  for i de 1 à t **do**
    $Y \leftarrow$ Bounding-chain-update(Y,U_i)
  end for
else
  Pour tout ν, $X(\nu)$ est l'unique élément de $Y(\nu)$
end if

4.2 Modèle hard-core gas

On considère le modèle hard-core gas (vu au chapitre 1, section 1.2) où chaque nœud peut-être de label 0 ou 1, deux nœuds adjacents ne peuvent être tout deux de label 1 et le poids d'une configuration dans la densité est proportionnel à λ puissance le nombre de nœuds de label 1. Lorsque le graphe est biparti, on a vu qu'il était possible d'utiliser une chaîne monotone en utilisant le bon ordre partiel. On considère maintenant un graphe non-biparti.
Dans la mise-à-jour de Gibbs vue section 1.4.4, un nœud était choisi uniformément ainsi que $U \sim Unif([0,1])$. Si $U < \lambda/(1+\lambda)$, et qu'aucun voisin du nœud choisi n'est de label 1, alors le nœud choisi obtient le label 1, sinon, il reste ou devient 0.
Pour la bounding chain, les nœuds sont labellisés $\{0\}$,$\{1\}$ ou $\{0,1\}$. Si un nœud ν a un voisin w de label $\{1\}$, alors cela signifie pour la bounding chain que $y(\nu) = \{0\}$. Si tous les voisins de ν sont de label $\{0\}$, alors seul U déterminera si $y(\nu)$ sera de label $\{0\}$ ou $\{1\}$.
Mais si certains voisins sont de label $\{0\}$ et d'autres sont de label $\{0,1\}$ alors on ne peut avoir que $y(\nu) = \{0,1\}$. En effet, il y aurait un état x lié par y tel que $x(w) = 1$ et un autre x' où $x'(w) = 0$ (on admet que $y(w) = \{0,1\}$). Donc selon la situation, on aurait $x(\nu) = 0$ ou $x'(\nu) = 1$ d'où $y(\nu) = \{0,1\}$.
Le pseudo-code suivant formalise cette idée. On propose une mise en place sous R d'une méthode de bounding chain pour le modèle HCGM en annexe, partie 4.
On peut alors utiliser la bounding chain avec le CFTP pour générer des échantillons exactement selon la loi cible. On veut à présent savoir combien de temps met le CFTP dans ce cas pour fonctionner. Pour cela, on doit borner le temps moyen pour que tous les labels $\{0,1\}$ disparaissent dans la bounding chain.

HCGM-bounding-chain-Gibbs-update Entrée : y,$\nu \in V$,$U \in [0,1]$ Sortie : y

$N_{\{1\}} \leftarrow$ nombre de voisins de ν de label $\{1\}$ dans y
$N_{\{0,1\}} \leftarrow$ nombre de voisins de ν de label $\{0,1\}$ dans y
if $U > \lambda/(1+\lambda)$ ou $N_{\{1\}} > 0$ **then**
    $y(\nu) \leftarrow \{0\}$
else if $N_{\{0,1\}} = 0$ **then**
    $y(\nu) \leftarrow \{0\}$
else
    $y(\nu) \leftarrow \{0,1\}$
end if

Lemme 17 (Huber, p.63, lemme 4.1)**.** *Notons ϕ_t^{BC} comme étant t pas effectués à l'aide de HCGM-bounding-chain-Gibbs-update. Pour $\lambda < 1/(\Delta-1)$,où Δ est le degré du graphe, la probabilité pour que la bounding chain lie plus d'un état est au plus de*

$$\#V \exp(-t(\#V)^{-1}(1-\lambda\Delta(1+\lambda)))$$

Démonstration. Pour voir la preuve complète, on se réfère au lemme 4.1 du livre d'Huber. La preuve étudie le comportement de la variable aléatoire
$W_t = \{\nu | Y_t(\nu) = \{0,1\}\}$ qui compte le nombre de nœuds ayant pour label $\{0,1\}$. Il suffit alors d'énumérer les cas pouvant se produire : sélection d'un nœud ayant le label $\{0,1\}$, tentative de modification de label,etc et de majorer l'espérance conditionnelle $\mathbb{E}[W_{t+1}|W_t]$
La preuve se termine à l'aide de l'inégalité de Markov avec $\mathbb{P}(W_t > 0) \leq \mathbb{E}[W_t]$ □

Shift chain

On considère le modèle shift introduit en 1.4.4. La chaîne introduite peut donner le label 1 à un nœud ν ayant exactement un voisin w de label 1 selon une Bernouilli $S \sim Bern(p_{swap})$. Lorsque celle-ci vaut 1, on a alors $x(\nu) = 1$ et $x(w) = 0$. Lorsque $S = 1$, la valeur de $x(w)$ est déplacée/shiftée vers $x(\nu)$.
Pour la bounding chain y, cela soulève de nombreux cas à gérer. L'ensemble de ces cas sont traités dans le tableau suivant. On notera $(N_{\{0\}}, N_{\{1\}}, N_{\{0,1\}})$ respectivement le nombre de voisins du nœud ν sélectionné de label $\{0\}$,$\{1\}$ et $\{0,1\}$. Lorsque $N_{\{1\}}$ ou $N_{\{0,1\}}$ vaut 1, on notera w l'unique voisin de ν ayant ce label. Le caractère $*$ désigne un caractère joker, pouvant se voir affecter n'importe quel entier positif ou nul inférieur ou égal à Δ. On propose une mise en place de méthode de bounding chain sous R pour ce modèle en annexe, partie 4.

Cas n°	$(N_{\{0\}}, N_{\{1\}}, N_{\{0,1\}})$	$U < \lambda/(1+\lambda)$?	S	Bounding chain
1	$(*,*,*)$	Non	0 ou 1	$y(\nu) = \{0\}$
2	$(*,0,0)$	Oui	0 ou 1	$y(\nu) = \{1\}$
3	$(*,0,1)$	Oui	0	$y(\nu) = \{0,1\}$
4	$(*,0,1)$	Oui	1	$y(\nu) = \{1\}$,$y(w_{\{0,1\}}) = \{0\}$
5	$(*,1,*)$	Oui	0	$y(\nu) = \{0\}$
6	$(*,1,0)$	Oui	1	$y(\nu) = \{1\}$,$y(w_{\{1\}}) = \{0\}$
7	$(*,1,\geq 1)$	Oui	1	$y(\nu) = \{0,1\}$,$y(w_{\{1\}}) = \{0,1\}$
8	$(*,\geq 2,*)$	Oui	0 ou 1	$y(\nu) = \{0\}$

Lemme 18 (Huber, p.65, lemme 4.2)**.** *Notons ϕ_t^{BC} comme étant t pas effectués par la bounding chain du modèle shift avec $p_{swap} = 1/4$. Si $\lambda < 2/(\Delta - 2)$, alors la probabilité que la bounding chain lie plus d'un état est d'au plus :*

$$\#V \exp(-t(\#V)^{-1}(1 - \lambda\Delta/[2(1+\lambda)])$$

Démonstration. La preuve complète se situe dans le livre d'Huber : lemme 4.2. La preuve se déroule de la même façon, en étudiant $W_t = \{\nu : Y_t(\nu) = \{0,1\}\}$ à l'aide du tableau de cas précédent. □

4.3 Modèle d'Ising

On considère le modèle d'Ising (vu au chapitre 1, section 1.2). Ayant vu maintenant quelques disjonctions de cas, la mise à jour du modèle d'Ising s'effectue de la manière suivante : pour un nœud $\nu \in V$, et pour $U \sim Unif([0,1])$, si $U < \exp(\beta * N_1 + \mu)/[\exp(\beta * N_1 + \mu) + \exp(\beta * N_{-1} - \mu)]$, alors ν prend le label 1, sinon, il prend le label -1.
Ici, à cause des labels d'états inconnus $\{-1, 1\}$, on ne peut pas appliquer directement cette mise à jour. On peut cependant procéder de la manière suivante : si au moins un des voisins du nœud ν est de label inconnu, il suffit alors de vérifier la condition suivante : si $U < \exp(\beta * N_1 + \mu)/[\exp(\beta * N_1 + \mu) + \exp(\beta * (N_{-1} + N_{-1,1}) - \mu)]$, alors ν prend le label 1 sinon, si $U > \exp(\beta * (N_1 + N_{-1,1}) + \mu)/[\exp(\beta * (N_1 + N_{-1,1}) + \mu) + \exp(\beta * N_{-1} - \mu)]$, alors ν prend le label -1, sinon, ν prend le label inconnu. C'est-à-dire : lorsqu'on choisit un nœud ayant au moins un voisin de label inconnu, on se place dans les cas extrémaux qui nous permettent de déduire des informations. D'une part, tous les labels inconnus sont de label -1,d'autre part tous les labels inconnus sont de label 1. Les cas intermédiaires ne nous permettent pas de déduire d'informations sur le prochain état puisque l'on remarque simplement que :

$$\frac{\exp(\beta * N_1 + \mu)}{\exp(\beta * N_1 + \mu) + \exp(\beta * (N_{-1} + N_{-1,1}) - \mu)} < \ldots < \frac{\exp(\beta * (N_1 + N_{-1,1}) + \mu)}{\exp(\beta * (N_1 + N_{-1,1}) + \mu) + \exp(\beta * N_{-1} - \mu)}$$

où ... représente toutes les possibilités pour les labels $\{-1, 1\}$.
On en déduit alors le pseudo-code suivant :
On propose une mise en place de méthode de bounding chain pour le modèle d'Ising sous R en annexe, partie 4.

Bounding-chain-update-Ising-Gibbs Entrée : y,$\nu \in V$,$U \in [0,1]$ Sortie : y

$N_{\{1\}} \leftarrow$ nombre de voisins de ν de label $\{1\}$ dans y
$N_{\{-1,1\}} \leftarrow$ nombre de voisins de ν de label $\{-1,1\}$ dans y
$N_{\{-1\}} \leftarrow$ nombre de voisins de ν de label $\{-1\}$ dans y
if $U < \exp(\beta * N_1 + \mu)/[\exp(\beta * N_1 + \mu) + \exp(\beta * (N_{-1} + N_{-1,1}) - \mu)]$ **then**
    $y(\nu) \leftarrow \{1\}$
else if $U > \exp(\beta * (N_1 + N_{-1,1}) + \mu)/[\exp(\beta * (N_1 + N_{-1,1}) + \mu) + \exp(\beta * N_{-1} - \mu)]$ **then**
    $y(\nu) \leftarrow \{-1\}$
else
    $y(\nu) \leftarrow \{-1,1\}$
end if

4.4 Problème d'initialisation

Pour beaucoup de modèles, l'état initial de la bounding chain est clair : imposer le label comportant tous les états possibles à chaque nœud. Le problème survient lorsque qu'il faut avancer dans la chaîne : comment avancer sans aucune information à propos de la chaîne ?
Pour certains modèles, ce problème n'existe pas, comme par exemple le modèle gas hardcore pour un λ tel que $\mathbb{P}(U > \lambda/(1+\lambda))$ (pour $U \sim Unif([0,1])$) soit suffisamment grand. En effet, au vu de l'algorithme de mise à jour de la bounding chain, on obtiendrait couramment de nouvelles informations sur la configuration en cours, dans notre cas, des obtentions d'état $\{0\}$.
Le problème se pose notamment dans le modèle que l'on étudie dans la suite.

Modèles d'orientation sink-free

On considère un graphe non-dirigé $G = (V, E)$. Les arêtes sont des sous-ensembles de taille deux de l'ensemble des nœuds, où l'ordre n'a pas d'importance. Ainsi, l'arête $\{i, j\}$ et l'arête $\{j, i\}$ sont identiques. Orienter une arête signifie lui donner une direction, soit de i vers j, soit de j vers i.

Définition 23. *L'orientation d'un graphe est une labellisation où chaque arête est soit de label (i, j) soit de label (j, i). On dit qu'une arête de label (i, j) est dirigée de i vers j. On appelle alors i la queue et j la tête.*

Définition 24. *On dit qu'une orientation possède un puits (venant de sink = couler) au nœud i si chacune des arêtes de la forme $\{i, j\}$ est orientée (j, i). Une orientation où aucun des nœuds n'est un puits est dite sink-free.*

Lorsque le degré d'un nœud est 1, il n'y a qu'une seule manière d'orienter l'arête afin de ne pas créer un puits. On oriente alors cette arête de la dite manière puis on ne prend plus en considération ces nœud et arête pour la suite. On répète ce processus pour chaque nœud de degré 1 ainsi créé. On peut donc considérer en toute généralité les graphes ou

chaque nœud est de degré au moins 2.
De la même manière, si un graphe est déconnecté, alors chaque composante sera orientée séparément. On peut donc considérer en toute généralité seulement les graphes connectés.
On considère l'ensemble des orientations sink-free d'un graphe. Un sampler de Gibbs choisit une arête uniformément au hasard puis choisit une orientation pour l'arête choisie uniformément dans les orientations qui ne créé pas de puits. Un pseudo-code pour cette mise-à-jour est comme suit :

Sink-free-orientation-Gibbs-chain Entrée : x,$\{i,j\} \in E$,$B \in \{0,1\}$ Sortie : x

1: Si $B = 1$ alors $a \leftarrow (i,j)$, sinon, $a \leftarrow (j,i)$
2: $n_{out} \leftarrow \#\{w | x(\{a(2),w\}) = (a(2),w)\}$
3: Si $n_{out} \geq 1$, alors $x(\{i,j\}) \leftarrow a$

C'est-à-dire : si l'orientation choisie laisse au moins une autre arête quitter la tête de l'arête choisie, alors on effectue le changement dans l'orientation.
Bubley et Dyer ont notamment montré que si deux chaînes X et X' sont passées à travers le pseudocode précédent en utilisant le même choix d'arête et de B, alors après $O(\#E^3)$ pas, il y a une forte probabilité que les deux chaînes aient coalescé (voir lemme 4.5 du livre d'Huber).
Il est donc possible de faire coalescer deux états. Cependant, pour utiliser le CFTP, il est nécessaire de faire coalescer tous les états. Le problème de la chaîne que l'on considère est qu'il n'est pas possible d'utiliser les bounding chain dans leur état actuel : à l'état initial, la bounding chain que l'on considère aura chaque arête de la forme (i,j) ayant le label $\{(i,j),(j,i)\}$. Avancer dans la chaîne n'aide pas à réduire le nombre de ces labels : le fait que les orientations des arêtes soient inconnues empêche de déduire des informations sur la bounding chain.
La solution est alors la suivante : d'abord faire coalescer les états en 2 états puis utiliser le résultat qu'on montré Bubley et Dyer pour coalescer ces deux états.
Pour cela, choisissons une arête $\{i,j\}$. On note par Ω_1 l'ensemble des orientations où l'arête $\{i,j\}$ est orientée (i,j). On note de plus par Ω_2 l'ensemble des orientations telles que l'arête $\{i,j\}$ soit orientée (j,i).
La chaîne choisira alors uniformément aléatoirement une arête autre que $\{i,j\}$ puis choisira uniformément dans les orientations où cette arête ne créé pas de puits.
Puisque l'arête $\{i,j\}$ ne sera jamais choisie, la bounding chain pour Ω_1 peut maintenant progresser puisque l'orientation de $\{i,j\}$ est fixée à (i,j), et alors le nœud i ne sera jamais un puits peu importe l'orientation des arêtes voisines.
On effectue de même cette procédure pour la bounding chain dans Ω_2. Après $O(\#E^3)$ pas, il est très probable que chacune des deux chaînes aie coalescé vers une configuration chacune. On utilise alors le résultat montré par Bubley et Dyer pour coalescer ces deux états vers un seul.
On accomplit l'opération précédente à l'aide du pseudo-code suivant :

Lemme 19 (Huber, p.78, lemme 4.6)**.** *On suppose qu'il existe une arête e telle que $\#Y_0(e) = 1$, que $e_1, e_2, \ldots, e_t$ soient iid uniformes sur $E \setminus \{e\}$, et que $B_1, \ldots, B_t$*

Sink-free-orientation-bounding-chain Entrée : y,$\{i,j\} \in E$,$B \in \{0,1\}$ Sortie : y

1: Si $B = 1$ alors $a \leftarrow (i,j)$, sinon, $a \leftarrow (j,i)$
2: $n_{out} \leftarrow \#\{w | y(\{a(2),w\}) = (a(2),w)\}$
3: $n_{inconnu} \leftarrow \#\{w | y(\{a(2),w\}) = \{(a(2),w),(w,a(2))\}\}$
4: Si $n_{out} \geq 1$, alors $y(\{i,j\}) \leftarrow a$
5: Sinon, si $n_{inconnu} > 0$, $y(\{i,j\}) \leftarrow y(\{i,j\}) \cup \{(a(2),a(1))\}$

soient iid uniformes sur $\{0,1\}$*. Alors, si* Y_{t+1} = *Sink-free-orientation-bounding-chain*(Y_t,e_{t+1},B_{t+1})*, la probabilité qu'il existe une arête* e' *telle que* $\#Y_t(e') > 1$ *est au plus*

$$\exp\left(-t\left[\frac{1}{2\#E^3} + \frac{1}{24\#E^4}\right]\right)$$

En particulier, avoir que la probabilité qu'une arête reste inconnue soit au plus δ *requiert* $(2\#E^3 + O(\#E))\log(\delta^{-1})$ *pas.*

Démonstration. Voir lemme 4.6 du livre d'Huber pour plus de détails. La preuve étudie le comportement de la variable aléatoire $D_t = \#\{e | \#Y(e) > 1\}$ qui compte le nombre d'arête de label non unique dans la bounding chain. □

5 Chapitre 5 : Comparaison de méthodes

5.1 CFTP : Comparaison entre la méthode de Baccelli/Brémaud et Huber

La comparaison ici est très simple : l'approche de Baccelli/Brémaud demande de calculer toutes les trajectoires à partir de tous les états initiaux possibles, et demande donc un nombre de calculs bien plus élevé que la méthode d'Huber.
La méthode d'Huber ne requiert de connaître qu'une simple condition sur les choix aléatoires effectués.
Bien qu'au final il faille calculer la trajectoire complète, il n'est pas nécessaire de calculer de trajectoire à chaque fois que l'on tire des choix aléatoires, d'où un nombre de calculs et finalement un temps de calcul réduit.
De plus, la méthode de Baccelli/Brémaud n'est pas adaptée à tous les problèmes : certains d'entre eux sont trop compliqués pour qu'on puisse considérer tous les états initiaux possibles (par exemple, tout modèle se basant sur des graphes aura un nombre d'états initiaux augmentant exponentiellement selon le nombre de nœuds).
Cependant, la mise en place de Baccelli/Brémaud est conceptuellement plus simple que celle d'Huber : il suffit de connaître tous les états possibles, de connaître la fonction de mise à jour et de faire alors évoluer toutes les trajectoires selon des choix aléatoires ; la méthode de Huber requiert de connaître suffisamment le comportement de la chaîne étudiée afin de déterminer un ensemble dans lequel les choix aléatoires effectués permettent d'obtenir le même état final, peu importe le choix initial effectué.
En somme, si la chaîne est suffisamment simple et que vous ne la connaissez pas suffisamment, utilisez plutôt la méthode Baccelli/Brémaud, sinon, si vous avez la connaissance de l'ensemble des choix aléatoires résultant en un même état final, utilisez la méthode d'Huber.
La comparaison n'est au final valable que pour les modèles "simples" (comme l'exemple avec les états $\{0, 1, 2\}$ donné par Huber) puisque ensuite se retrouve avec la méthode de CFTP monotone ou les bounding chains que l'on analyse dans la suite.

5.2 Comparaison entre CFTP monotone et bounding chains

Pour comparer les deux méthodes, il faut d'abord pouvoir adapter un modèle aux deux méthodes.
On considère donc ici le modèle hardcore gas sur graphe biparti, dont on peut échantillonner de la distribution stationnaire selon les deux méthodes : CFTP monotone ou bounding chain. On considèrera ensuite le modèle d'Ising avec $\mu = 0$.

5.2.1 Modèle HCGM sur graphe biparti

Une analyse plus extensive serait nécessaire pour confirmer l'intuition que nous donne les modèles étudiés ici : nous allons nous intéresser à deux graphes bipartis tels que $\#V = 8$.
On considère d'abord le graphe suivant :

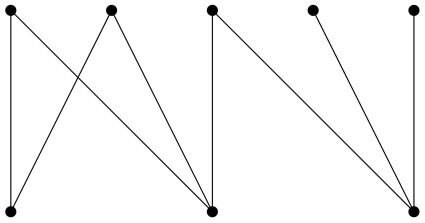


FIGURE 6 – Graphe biparti à nœuds numérotés de 1 à 8 : 1 à 5 (en haut), 6 à 8 (en bas)

De matrice d'adjacence $A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}$.

On compare les méthodes de CFTP et de bounding chain en lançant un nombre élevé de fois chacun des programmes qui rendra alors le nombre de lancers uniformes effectués sur la plus grande récursion (le travail effectué est alors de deux fois ce nombre d'uniformes moins un). Une mise en place sous R de la méthode de bounding chain pour le modèle HCGM sur graphe biparti est proposée en annexe, partie 5.
Dans les graphes qui suivent, suite à un trop grand travail effectué dans la méthode CFTP, on a décidé ici d'arrêter le programme si le nombre d'uniformes lancées est 8192, on assume donc que le nombre d'uniformes nécessaire pour l'obtention d'un échantillon de la loi stationnaire dépasserait ce montant. Par exemple, pour $\lambda = 0.5$, sur 10^3 lancers on obtient, pour la méthode CFTP :

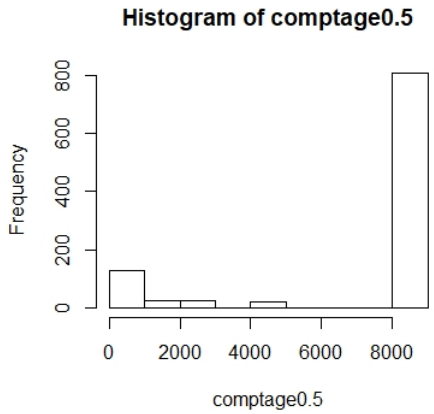


FIGURE 7 – Histogramme du nombre de lancers aléatoires nécessaires à l'obtention d'une sortie pour la méthode CFTP

8	16	32	64	128	256	512	1024	2048	4096	8192
1	22	21	24	22	16	22	23	22	19	808

TABLE 3 – Données de l'histogramme précédent

Alors que, pour $\lambda = 0.5$, pour la méthode de bounding chains, sur 10^4 lancers, on obtient :

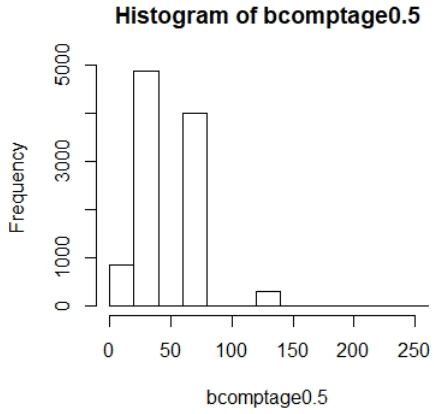
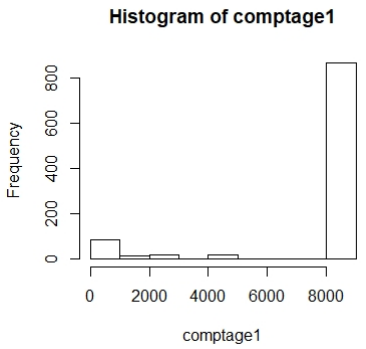


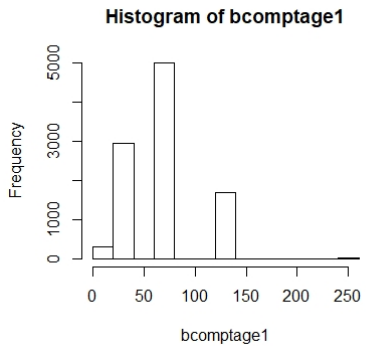
FIGURE 8 – Histogramme du nombre de lancers aléatoires nécessaires à l'obtention d'une sortie pour la méthode de bounding chains

8	16	32	64	128	256
2	832	4870	3998	297	1

TABLE 4 – Données de l'histogramme précédent

- On effectue de même pour $\lambda = 1$ pour obtenir :

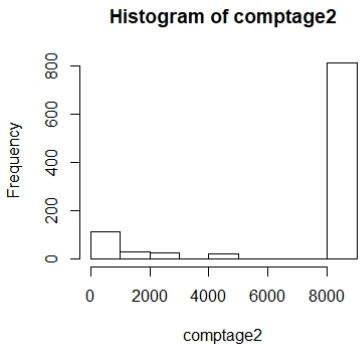


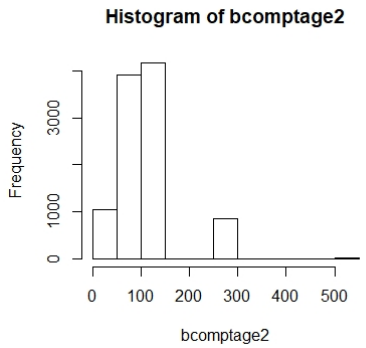


16 à 256	512	1024	2048	4096	8192
67	17	14	18	15	869

8	16	32	64	128	256
2	832	4870	3998	297	1

- Puis enfin pour $\lambda = 2$:





16 à 256	512	1024	2048	4096	8192
90	23	30	25	20	812

16	32	64	128	256	512
52	991	3907	4166	859	25

- On effectue ensuite une analyse similaire pour le graphe suivant :

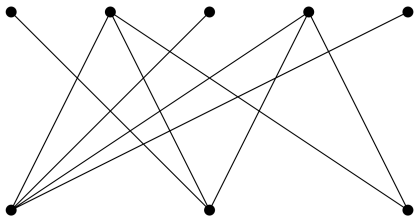
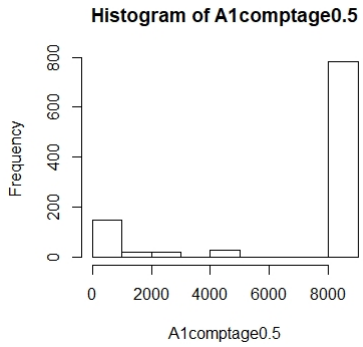
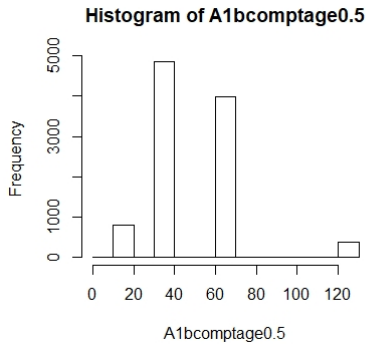


FIGURE 9 – Graphe biparti à nœuds numérotés de 1 à 8 : 1 à 5 (en haut), 6 à 8 (en bas)

De matrice d'adjacence $A1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$.

- On obtient alors, pour $\lambda = 0.5$:

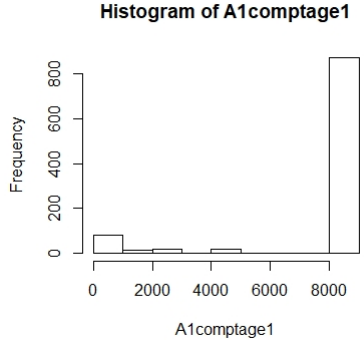


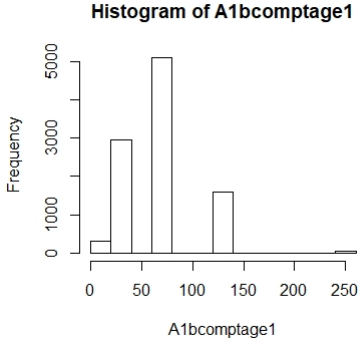


16 à 256	512	1024	2048	4096	8192
123	25	20	21	29	782

8	16	32	64	128
1	781	4860	3997	361

- Puis, pour $\lambda = 1$:

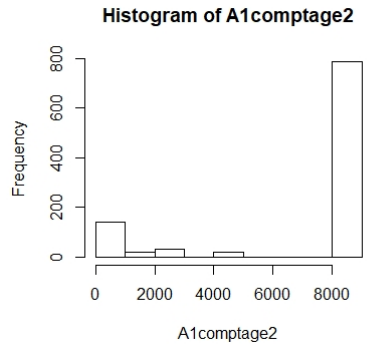


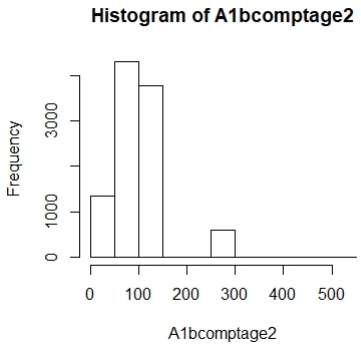


8 à 256	512	1024	2048	4096	8192
59	21	14	16	17	873

16	32	64	128	256
311	2956	5097	1592	44

- Et enfin, pour $\lambda = 2$:





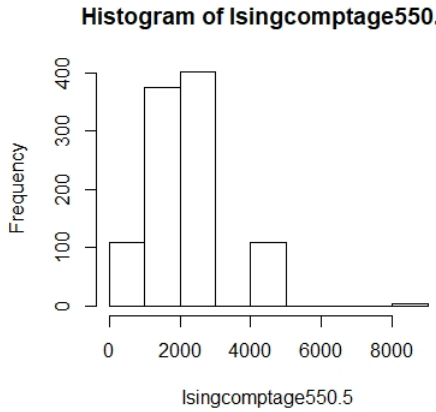
16 à 256	512	1024	2048	4096	8192
100	39	21	31	21	788

16	32	64	128	256	512
82	1265	4301	3763	584	5

Au vu des résultats précédents, nous pouvons émettre la supposition suivante : la méthode de bounding chains semble être bien meilleure que la méthode de CFTP pour le modèle hardcore gas sur graphe biparti. D'autres test seraient nécessaire pour confirmer cette supposition (notamment en augmentant la taille du graphe, en testant la plupart des configurations possible, en variant plus λ,etc).

5.2.2 Modèle d'Ising

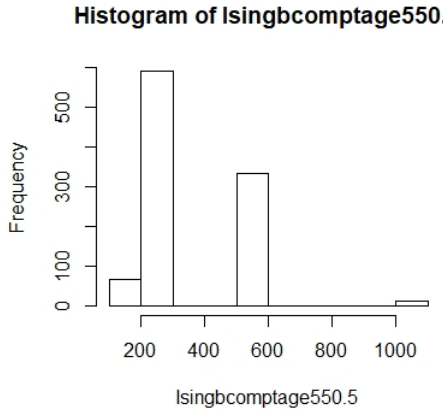
On considère maintenant le modèle d'Ising ayant pour paramètres $\mu = 0$ et $\beta > 0$ (tel que la fonction de mise à jour soit monotone, voir section 3.3.1). On utilise pour la méthode CFTP la fonction de mise à jour présentée en 3.3.1, puis pour la méthode de bounding chain, la fonction de mise à jour présentée en 4.3.
Une analyse plus poussée serait nécessaire pour affiner les résultats suivants. On considèrera différentes tailles de graphes rectangulaires (variant entre 5×5 et 8×8) ainsi que différents β (0.5,1, et 1.5).
D'abord pour un graphe de taille 5×5 et $\beta = 0.5$, pour le modèle CFTP on obtient :



256	512	1024	2048	4096	8192
7	102	376	402	109	4

TABLE 5 – Données de l'histogramme précédent

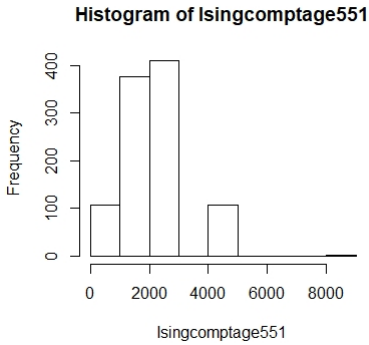
Alors que pour la méthode de bounding chains, on obtient :



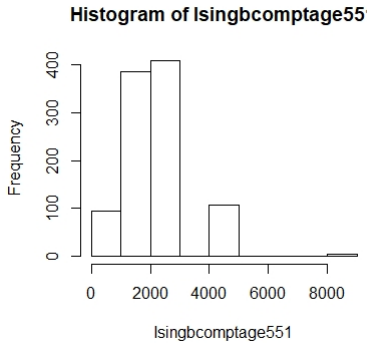
128	256	512	1024
65	590	333	12

TABLE 6 – Données de l'histogramme précédent

- Puis, pour $\beta = 1$:

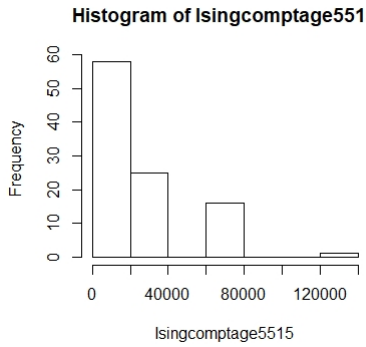


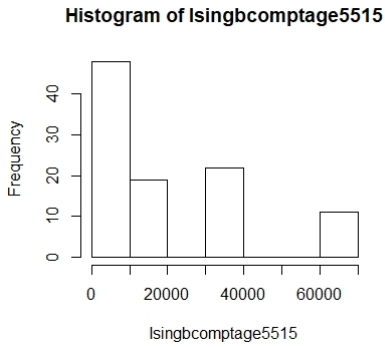
256	512	1024	2048	4096	8192
6	100	376	410	107	1



256	512	1024	2048	4096	8192
7	87	386	409	107	4

- Et enfin, pour $\beta = 1.5$:

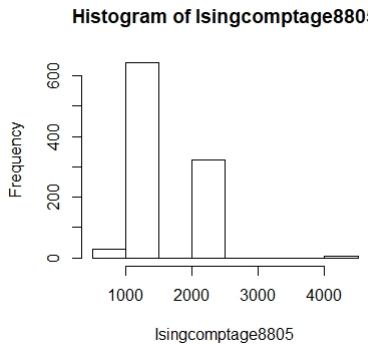


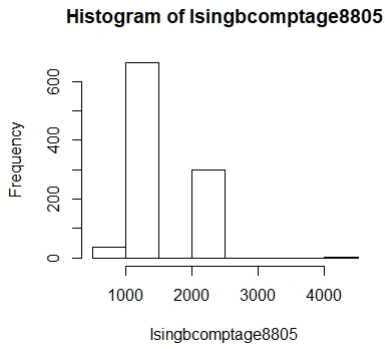


2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	2^{17}
3	15	22	18	25	16	1

2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
3	3	12	30	19	22	11

• On effectue une analyse similaire pour les graphes rectangulaires de taille 8×8. D'abord, pour $\beta = 0.5$:

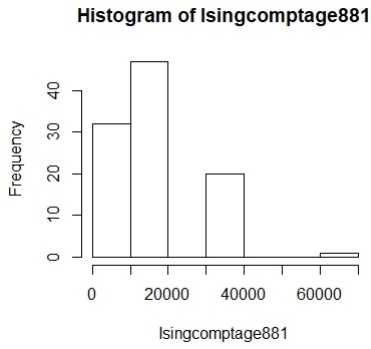




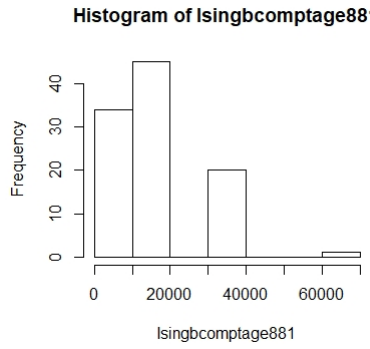
512	1024	2048	4096
29	643	322	6

512	1024	2048	4096
36	663	298	3

•Puis, pour $\beta = 1$:



2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
7	25	47	20	1



2^{12}	2^{13}	2^{14}	2^{15}	2^{16}
5	29	45	20	1

On n'effectuera pas de comparaison pour $\beta = 1.5$. En effet, les algorithmes prennent beaucoup trop de temps d'exécution pour obtenir une seule sortie.
Le problème pour la méthode de CFTP survient lorsque la taille du graphe augmente : les états extrémaux prennent trop de temps pour se rejoindre. Le problème pour la méthode de bounding chain est le suivant : pour démarrer, n'ayant que des états inconnus dans la bounding chain, il faut absolument que $U < \exp(\beta * N_1 + \mu)/[\exp(\beta * N_1 + \mu) + \exp(\beta * (N_{-1} + N_{-1,1}) - \mu)]$ ou que $U > \exp(\beta*(N_1+N_{-1,1})+\mu)/[\exp(\beta*(N_1+N_{-1,1})+\mu)+\exp(\beta*N_{-1}-\mu)]$ afin de pouvoir en déduire des informations sur la chaîne. N'ayant que des états inconnus, et ayant $\lambda = 1.5$, ces conditions sont alors, pour un graphe carré : $U < \frac{1}{1+\exp^{1.5*4}} = 0.002472623$ ou encore $U > \frac{exp^{4*1.5}}{exp^{4*1.5}+1} = 0.9975274$. Ces conditions sont difficiles à atteindre, et donc augmentent le temps d'obtention d'une sortie pour la méthode de bounding chains.

La comparaison entre les deux méthodes pour le modèle d'Ising est peu concluante, les résultats indiquant que les deux méthodes sont plus ou moins équivalentes.

Conclusion personnelle

Les méthodes présentées dans ce mémoire ne sont que les prémices des méthodes de simulation parfaite : d'autres méthodes plus efficaces existent, d'autres méthodes adaptées à des modèles plus compliqués existent,etc.
Ce mémoire aide à entrer dans ce monde aussi complexe qu'intéressant, de nombreuses applications peuvent découler de ces méthodes et idées exposées ici.
Ce stage m'a permis d'en apprendre plus sur le travail de recherche ainsi que sur le domaine de la simulation, j'ai pu faire la connaissance de personnes expérimentées qui m'ont apporté de nombreux conseils avisés que je tente au mieux de suivre. J'ai appris énormément de ce stage, que ce soit au niveau des modèles jusqu'alors inconnus ou que ce soit des techniques et méthodes de simulation.
J'ai aussi pu approfondir les résultats présentés par Huber et j'ai apporté une certaine analyse des méthodes proposées.

6 Annexe : programmes

6.1 Chapitre 1

Programmation sous R : échantillonneur de Gibbs

```
## X = graphe (rectangulaire ou carré), u = lancer uniforme sur [0,1], v = noeud de X ##
IsingGibbsUpdate <- function(X,u,v,beta,mu){ ## Permet de faire un pas de la chaîne de Markov associée selon le Gibbs Sampler##

  ##On entoure la matrice de valeurs aberrantes pour la manipuler plus facilement##
  ##On change les indices du point considéré pour se conformer à la nouvelle matrice##

  i = v[1] + 1
  j = v[2] + 1

  X = rbind(rep(0,dim(X)[2]),X)
  X = rbind(X,rep(0,dim(X)[2]))
  X = cbind(rep(0,dim(X)[1]),X)
  X = cbind(X,rep(0,dim(X)[1]))

##Les indicatrices permettent de vérifier quelles valeurs ont les voisins du point considéré ##

  if(u< exp(mu + beta*((X[i+1,j]==1)*1 + (X[i-1,j]==1)*1 +(X[i,j+1]==1)*1 + (X[i,j-1]==1)*1))/((exp(mu + beta*((X[i+1,j]==1)*1 + (X[i-1,j]==1)*1 +(X[i,j+1]==1)*1 + (X[i,j-1]==1)*1))) + exp(-mu + beta*((X[i+1,j]==-1)*1 + (X[i-1,j]==-1)*1 +(X[i,j+1]==-1)*1 + (X[i,j-1]==-1)*1)))){
    X[i,j] = 1
  }
  else{
    X[i,j] = -1
  }

  ##On supprime les valeurs insérées##
  X=X[,-1]
  X=X[,-dim(X)[2]]
  X=X[-1,]
  X=X[-dim(X)[1],]

  X
}
```

```
IsingGibbssteps <- function(X,beta,mu,n){ ##effectue n pas de la chaîne de
    Markov pour le sampler de Gibbs, pour un graphe rectangulaire X##

  for (i in 1:n){

  ##choix uniforme du noeud considéré##
    k = sample(1:dim(X)[1],1)
    l = sample(1:dim(X)[2],1)
  ##lancer uniforme sur [0,1]##
    u = runif(1)
  ##mise à jour du point##
    X=IsingGibbsUpdate(X,u,c(k,l),beta,mu)

  }

  X
}
```

```
IsingGibbsconditionnalstop <- function(X,beta,epsilon){ ##effectue un nombre aléatoire de pas de la chaîne de Markov pour le sampler de Gibbs, pour un graphe carré X##

  ##On créé le vecteur Y qui contiendra les ratios de 1 pour chaque graphe##
  Y=c(mean(X==1))
  bol = TRUE
  i=1

  while (bol){

  ##Choix uniforme du noeud considéré##
    k = sample(1:dim(X)[1],1)
    l = sample(1:dim(X)[2],1)

  ##lancer uniforme sur [0,1]##
    u = runif(1)

    ##Mise à jour du noeud considéré##
    X=IsingGibbsUpdate(X,u,c(k,l),beta)

    ##Mise à jour du vecteur des ratios##
    Y = c(Y,mean(X==1))

    ##Si la ligne est décommentée, on obtient une représentation de la configuration X tout les (dim(X)[1]**2)/epsilon pas##
    ##if ((i%%((dim(X)[1]**2)/epsilon)==0)){image(X)}

  ##Si on a effectué un nombre de pas suffisant, et que le nombre de 1 n'a pas varié énormément durant un certain nombre d'itérations, on s'arrête##
    if ((i>=((log(dim(X)[1]**2)*dim(X)[1]**2)/epsilon)) && ((mean(Y[i+1-floor((dim(X)[1]**(1/2))/epsilon):i])-Y[i])<epsilon)){
      break()
    }
    i=i+1
  }

  return(list(X,i))

}
```

Programmation sous R : Métropolis-Hastings

```
## X = graphe (rectangulaire ou carré), u = lancer uniforme sur [0,1], v = noeud de X ##

MetropolisHastingsUpdate <- function(X,u,v,beta){ ## Permet de faire un pas de la chaîne de Markov associée selon Metropolis-Hastings

  ##On entoure la matrice de valeurs aberrantes pour la manipuler plus facilement##
  ##On change les indices du point considéré pour se conformer à la nouvelle matrice##

  i = v[1] + 1
  j = v[2] + 1

  X = rbind(rep(0,dim(X)[2]),X)
  X = rbind(X,rep(0,dim(X)[2]))
  X = cbind(rep(0,dim(X)[1]),X)
  X = cbind(X,rep(0,dim(X)[1]))

  ##On choisit une couleur candidate pour le noeud v##

  c=sample(c(-1,1),1,prob=c(1/2,1/2))

  ##Les indicatrices permettent de vérifier quelles valeurs ont les voisins du point considéré ##

  if(u< exp(beta*((X[i+1,j]==c)*1 + (X[i-1,j]==c)*1 +(X[i,j+1]==c)*1 + (X[i,j-1]==c)*1))/exp(beta*((X[i+1,j]==X[i,j])*1 + (X[i-1,j]==X[i,j])*1 +(X[i,j+1]==X[i,j])*1 + (X[i,j-1]==X[i,j])*1))){
    X[i,j] = c
  }

  ##On supprime les valeurs insérées##
  X=X[,-1]
  X=X[,-dim(X)[2]]
  X=X[-1,]
  X=X[-dim(X)[1],]

  X
}
```

```
MetropolisHastingssteps <- function(X,beta,n){ ##effectue n pas de la chaîne de
    Markov pour Metropolis-Hastings, pour un graphe rectangulaire X##

  for (i in 1:n){

  ##choix uniforme du noeud##
    k = sample(1:dim(X)[1],1)
    l = sample(1:dim(X)[2],1)

  ##lancer uniforme sur [0,1]##
    u = runif(1)

  ##mise à jour de la configuration X##
    X=MetropolisHastingsUpdate(X,u,c(k,l),beta)
  }

  X

}
```

```
1
2  MetropolisHastingsconditionnalstop <- function(X,beta,epsilon){ ##effectue un
       nombre aléatoire de pas de la chaîne de Markov pour Metropolis-Hastings,
       pour un graphe carré X##
3
4   ##On créé le vecteur Y qui contiendra les ratios de 1 pour chaque graphe##
5    Y=c(mean(X==1))
6    bol = TRUE
7    i=1
8
9
10   while (bol){
11
12   ##Choix uniforme du noeud considéré##
13     k = sample(1:dim(X)[1],1)
14     l = sample(1:dim(X)[2],1)
15
16   ##lancer uniforme sur [0,1]##
17     u = runif(1)
18
19   ##Mise à jour du noeud considéré##
20     X=MetropolisHastingsUpdate(X,u,c(k,l),beta)
21
22   ##Mise à jour du vecteur des ratios##
23     Y = c(Y,mean(X==1))
24
25   ##Si la ligne est décommentée, on obtient une représentation de la
         configuration X tout les (dim(X)[1]**2)/epsilon pas##
26   ##if ((i%%((dim(X)[1]**2)/epsilon)==0)){image(X)}##
27
28   ##Si on a effectué un nombre de pas suffisant, et que le nombre de 1 n'a pas
         varié énormément durant un certain nombre d'itérations, on s'arrête##
29     if ((i>=((log(dim(X)[1]**2)*dim(X)[1]**2)/epsilon)) && ((mean(Y[i+1-floor((
           dim(X)[1]**(1/2)))/epsilon:i])-Y[i])<epsilon)){
30       break()
31     }
32     i=i+1
33   }
34
35   return(list(X,i))
36
37 }
```

6.2 Chapitre 2

```
normalfromcauchy <- function(){## Fonction permettant de générer une réalisation
     de la loi normale standard à partir de la loi de Cauchy standard par
    méthode de rejet ##

##penser à changer le return pour obtenir une réalisation (return(x)) ou le
    temps mis à l'obtenir (return(n))##

  ##On initialise les valeurs ##
  C=0
  x=0
  n=0

  ##Tant que la bernouilli n'est pas 1##
  while(C!=1){

    ##On génère une Cauchy standard##
    x = rcauchy(1)

    ##On génère une Bernouilli du paramètre indiqué##
    C = rbinom(1,1,((sqrt(exp(1))/2)*(1+x^2)*exp(-(1/2)*x^2)))

    ##mise à jour du nombre de lancers##
    n=n+1
  }
  return(x)
}



##On calcule le nombre moyen de lancers nécessaires pour atteindre la sortie de
    l'algorithme précédent##

##On initialise le nombre moyen##
s=0

##On lance 10^6 fois l'algorithme qui retourne le nombre de lancers effectué ##
for (i in 1:10^6) {
  s=s+normalfromcauchy()
}

##On moyenne sur les 10^6 lancers
s = s/10^6



##On créé un histogramme illustrant 10^5 lancers de l'algorithme rendant une
    réalisation de la normale standard##
```

```
43
44 ##On initialise le tableau qui contiendra les 10^5 lancers
45 t = matrix(0,1,10^5)
46
47 ##On remplit le tableau en lançant 10^5 fois l'algorithme##
48 for (i in 1:10^5) { #changer le return pour obtenir des réalisations
49   t[i]=normalfromcauchy()
50
51 }
52
53 ##On produit l'histogramme du tableau t et on rajoute la courbe de la densité de
       la loi normale standard##
54 hist(t,prob=T)
55
56 curve(dnorm(x, 0, 1), col="red",xlim=c(-4,4),add=T)
```

Mise en place de la méthode de rejet sour R :

Inégalités de Chernoff et queue de loi binomiale

```
AR_chernoffs <- function(n,p,a){ ##fonction permettant de générer une
    réalisation de la loi binomiale de paramètres n et p, conditionnellement à
    être supérieure où égale à a en utilisant les inégalités de Chernoff##


  ##On initialise la Bernouilli##
  C=0

  ##On créé la valeur gamma présentée précédemment##
  gam = (a/n)*(1-p)/(p*(1-(a/n)))

  ##On initialise la Binomiale de paramètres n et p
  t = 0

  ##On initialise le nombre de lancers##
  k = 0

  ##Tant que la Bernouilli n'est pas 1##
  while (C != 1){

    ##On génère n Bernouillis de paramètres a/n##
    t = rbinom(n,1,a/n)

    ##On effectue la somme pour obtenir la binomiale considérée##
    x = sum(t)

    ##On génère une réalisation de la Bernouilli du paramètre indiqué
    C = rbinom(1,1,(x>=a)*1*gam**(a-x))

    ##On met à jour le nombre de lancers effectués##
    k=k+1
  }

  ##On rend soit le nombre de lancers, soit la réalisation de la loi demandée (
      ie, k ou x))##
  return(k)

}


##On calcule ici le nombre moyen de lancers nécessaires pour sortir de l'
    algorithme précédent

s2=0
```

```
41 for (i in 1:10^5) { #changer le return pour obtenir le nombre moyen de lancers
       nécessaires
42   s2=s2 + AR_chernoffs(10,0.1,5)
43 }
44 s2 = s2/10^5
```

```r
AR_basic <- function(n,p,a){##fonction permettant de générer une réalisation de la loi binomiale de paramètres n et p, conditionnellement à être supérieure où égale à a en utilisant la méthode de rejet basique##

  ##On initialise le critère d'arrêt##
  C=0

  ##On initialise la Binomiale de paramètres n et p##
  t = 0

  ##On initialise le nombre de lancers##
  k = 0

  ##Tant que le critère d'arrêt n'est pas 1##
  while (C != 1){

    ##On génère une réalisation de la Binomiale de paramètres n et p##
    t = rbinom(1,n,p)

    ##On met à jour le critère d'arrêt selon que la binomiale soit supérieure à a ou non##
    C = (t>=a)*1

    ##On met à jour le nombre de lancers##
    k=k+1
  }

  ##On rend soit le nombre de lancers, soit la réalisation de la loi demandée (ie, k ou t))##
  return(k)

}


##On calcule ici le nombre moyen de lancers nécessaires pour sortir de l'algorithme précédent
s1=0
for (i in 1:10^6) { #changer le return pour obtenir le nombre moyen de lancers nécessaires
  s1=s1+ AR_basic(10,0.1,5)
}
s1= s1/10^6
```

```r
AR_boule_unite <- function(n){ ##Génère un tirage alétoire sur la boule unité de dimension n selon la méthode de rejet présentée dans le chapitre 2##

  u = runif(n,-1,1)
  s = u**2
  s = sum(s)
  i=1
  while(s>1){
    u = runif(n,-1,1)
    s=u**2
    s = sum(s)
    i=i+1
  }

  ##return u pour avoir l'échantillon, return i pour avoir le nombre de lancers nécessaires à l'obtention de l'échantillon##
  return(u)


}


s3=0
for (i in 1:10^6) { #changer le return pour obtenir le nombre moyen de lancers nécessaires
  s3=s3+ AR_boule_unite(5)
}
s3= s3/10^6


##Trace la courbe de croissance du temps d'obtention d'un échantillon selon la dimension##
curve((gamma(1+x/2)*2**x)/(pi**(x/2)),0,12, xlab = NULL)
```

6.3 Chapitre 3

```r
simple_update <- function(x,u,t){##fonction d'update présentée comme exemple par Huber##

  for (i in 1:t){

    if (u[i]<=0.5 && x>0){
      x = x-1
    }
    else if(u[i]>0.5 && x<2){
      x = x+1
    }

  }
```

```
13
14    return(x)
15
16  }
```

```
1   Coupling_from_the_past_Bacc <- function(){ ##effectue le CFTP pour la fonction d
        'update donnée plus haut, par la méthode donnée dans Baccelli et Bremaud##
2
3     ##changer le return pour obtenir une réalisation ou le nombre de pas utilisé (
          ie, a ou dim(u)[2])##
4
5
6     u = NULL ##on initalise les choix aléatoires##
7
8
9     while (TRUE) {
10
11      u = cbind(t(t(runif(3))),u) ##on ajoute de nouveaux choix alétoires##
12
13      ## on réinitialise les états de départ ##
14      a=0
15      b=1
16      c=2
17
18
19      ##on met à jour les états de départ selon les choix aléatoires dans u et la
            fonction d'update##
20      for (i in 1:dim(u)[2]){
21
22        a = simple_update(a,u[a+1,i],1)
23        b = simple_update(b,u[b+1,i],1)
24        c = simple_update(c,u[c+1,i],1)
25
26      }
27
28      ##si les trajectoires se sont rejointes en un même point, on rend ce point (
            ou le nombre de choix aléatoires pour l'obtenir##
29      if(a==b && b==c){
30        return(dim(u)[2])
31      }
32
33    }
34
35  }
```

```
Coupling_from_the_past_Huber <- function(){ ##effectue le CFTP pour la fonction
    d'update donnée plus haut, par la méthode donnée par Huber##

  ##Cet algorithme rend une réalisation de CFTP##

  u = runif(2)

  if (u[1]<0.5 && u[2]<0.5){

    return(0)

  }
  else{

    x = Coupling_from_the_past_Huber()
    return(simple_update(x,u,2))
  }

}

Coupling_from_the_past_counting <- function(i=2){##effectue le CFTP pour la
    fonction d'update donnée plus haut, par la méthode donnée par Huber##


  ##Cet algorithme rend le nombre d'appels à la fonction d'update nécessaires à
      l'obtention d'une réalisation de CFTP##

  u = runif(2)

  if (u[1]<0.5 && u[2]<0.5){

    return(i)

  }
  else{

    i = Coupling_from_the_past_counting(i+2)
    return(i)
  }

}
```

```
IsingGibbsUpdate <- function(X,u,v,beta){  ## X = graphe (rectangulaire ou
    carré), u = lancé uniforme sur [0,1], v = noeud de X ##

  ##On entoure la matrice de valeurs aberrantes pour la manipuler plus
      facilement##
```

```
4     ##On change les indices pour s'y conformer##
5
6     i = v[1] + 1
7     j = v[2] + 1
8
9     X = rbind(rep(0,dim(X)[2]),X)
10    X = rbind(X,rep(0,dim(X)[2]))
11    X = cbind(rep(0,dim(X)[1]),X)
12    X = cbind(X,rep(0,dim(X)[1]))
13
14    ##On applique la fonction de mise à jour selon le sampler de Gibbs, donnée par
          Huber##
15    if(u< exp(beta*((X[i+1,j]==1)*1 + (X[i-1,j]==1)*1 +(X[i,j+1]==1)*1 + (X[i,j
          -1]==1)*1))/((exp(beta*((X[i+1,j]==1)*1 + (X[i-1,j]==1)*1 +(X[i,j+1]==1)*1
           + (X[i,j-1]==1)*1))) + exp(beta*((X[i+1,j]==-1)*1 + (X[i-1,j]==-1)*1 +(X[
          i,j+1]==-1)*1 + (X[i,j-1]==-1)*1)))){
16      X[i,j] = 1
17    }
18    else{
19      X[i,j] = -1
20    }
21
22    ##On retire les valeurs aberrantes puis on rend la matrice##
23    X=X[,-1]
24    X=X[,-dim(X)[2]]
25    X=X[-1,]
26    X=X[-dim(X)[1],]
27
28    return(X)
29  }
30
31  Monotonic_Ising_Gibbs <- function(t,a,b,bet){ ##t = nombre de pas effectués, a =
          dim(X)[1], b = dim(X)[2]
32
33
34    ##On tire t choix aléatoires##
35    u = runif(t)
36
37    ##On tire t noeuds du graphe rectangulaire##
38    i = sample(1:a,t,replace = TRUE)
39    j = sample(1:b,t,replace = TRUE)
40
41    ##On initialise les états extremaux##
42    xmin = matrix(-1,a,b)
43    xmax = list(matrix(1,a,b),t)
44
45    ##On fait évoluer les états extrémaux selon les mêmes choix aléatoires##
46    for (k in 1:t){
47      xmax[[1]] = IsingGibbsUpdate(xmax[[1]],u[k],c(i[k],j[k]),bet)
```

```
    xmin = IsingGibbsUpdate(xmin,u[k],c(i[k],j[k]),bet)
  }
  ##si les fins de trajectoires résultant des mises à jour des états extrémaux
      sont différentes, on fait une récursion##
  if (sum(xmax[[1]] != xmin)>=1){
    xmax = Monotonic_Ising_Gibbs(2*t,a,b,bet)

    ##après récursion, xmax est un état tel que les états extrémaux se sont
        rejoints, on le met alors à jour selon les choix aléatoires effectués
        pré-récursion##
    for (m in 1:t){

      xmax[[1]] = IsingGibbsUpdate(xmax[[1]],u[m],c(i[m],j[m]),bet)

    }

  }

  return(xmax)

}

##permet d'effectuer les histogrammes obtenus en partie 5##
Isingcomptage8815 = NULL
for(i in 1:10^2){

  X = Monotonic_Ising_Gibbs(1,8,8,1.5)[[2]]
  Isingcomptage8815 = c(Isingcomptage8815,X)
  print(i)
}

table(Isingcomptage8815)
hist(Isingcomptage8815)
```

```
voisins_1 <- function(Y,A,k){ ## Entrée : état du modèle HCGM,matrice d'
    adjacence,numéro du noeud du graphe. Sortie : nombre de voisins de label 1
    du noeud choisi##

  X = Y
  V1 = X[[1]]
  V2 = X[[2]]
  N1 = 0

  ##numvoisin contiendra les numéros des noeuds voisins de k##
  numvoisin = (A[k,] == 1)
  a = 1:dim(A)[2]
  numvoisin = a[numvoisin]
```

```
12
13    ##On compte le nombre de 1 autour du noeud considéré selon qu'il soit dans une
          partie du graphe ou l'autre##
14
15    if(k <= dim(V1)[2]){
16
17      for (i in numvoisin){
18
19        N1 = N1 + 1*(V2[i - length(V1)]==1)
20
21      }
22    }else{
23
24      for (i in numvoisin){
25
26        N1 = N1 + 1*(V1[i]==1)
27
28      }
29    }
30
31    return(N1)
32
33  }
34
35  HCGM_Gibbs_update <- function(X,A,lambda,u){ ##met à jour un état X de la chaine
        pour le modèle HCGM biparti de matrice d'adjacence A
36
37    for(i in 1: length(u)){
38      ##tirage d'un noeud aléatoire uniformément##
39      ij = sample(1:dim(A)[1],1,replace = TRUE)
40
41      ##On compte les voisins de label 1##
42      N1 = voisins_1(X,A,ij)
43
44      ##On effectue la mise à jour pour le modèle HCGM donnée par Huber##
45      if((u[i] < lambda/(lambda +1)) && (N1==0)){
46        if(ij <= length(X[[1]])){X[[1]][ij] = 1}else{X[[2]][ij - length(X[[1]])] =
              1}
47      }else{if(ij <= length(X[[1]])){X[[1]][ij] = 0}else{X[[2]][ij - length(X
            [[1]])] = 0}}
48    }
49    return(X)
50  }
51
52  Monotonic_HCGM_Bipartite_Gibbs <- function(t,A,lV1,lV2,lambda){ ##t = nombre de
      pas effectués, A = matrice d'adjacence du graphe biparti, V1 = ensemble des
      noeuds d'un côté du graphe biparti, V2 = ensemble des noeuds de l'autre côté
       du graphe biparti
53
```

```
54    ##On effectue les choix aléatoires##
55    u = runif(t)
56
57    ##On initialise les états extremaux##
58    V1 = matrix(1,1,lV1)
59    V2 = matrix(0,1,lV2)
60    V3 = matrix(0,1,lV1)
61    V4 = matrix(1,1,lV2)
62
63    xmin = list(V1,V2)
64    xmax = list(V3,V4)
65
66    ##On fait évoluer les trajectoires selon les choix aléatoires##
67      xmin = HCGM_Gibbs_update(xmin,A,lambda,u)
68      xmax = HCGM_Gibbs_update(xmax,A,lambda,u)
69
70    ##si les fins de trajectoires résultant des mises à jour des états extrémaux
          sont différentes, on fait une récursion##
71    if ((sum(xmin[[1]]!=xmax[[1]])>=1 | sum(xmin[[2]]!=xmax[[2]])>=1) && t<8192){
72
73      xmax = Monotonic_HCGM_Bipartite_Gibbs(2*t,A,lV1,lV2,lambda)
74
75      ##après récursion, on met à jour la sortie selon les choix aléatoires
            effectués pré-récursion##
76      xmax[1:2] = HCGM_Gibbs_update(xmax,A,lambda,u)
77
78
79    }
80    return(c(xmax,t,(sum(xmin[[1]]!=xmax[[1]])>=1 | sum(xmin[[2]]!=xmax[[2]])>=1))
          )
81
82  }
83
84  ##On compte le nombre de choix aléatoires nécessaires à la sortie d'un
        algorithme, permet d'obtenir les histogrammes données en chapitre 5##
85
86  A1comptage2 = NULL
87  for(i in 1:10^3){
88
89    X = Monotonic_HCGM_Bipartite_Gibbs(t,A1,lV1,lV2,2)
90    if(X[[4]] == TRUE){A1comptage2 = c(A1comptage2,8192)}
91    else{A1comptage2= c(A1comptage2,X[[3]]) }
92    print(i)
93  }
94
95  table(A1comptage2)
96  hist(A1comptage2)
```

6.4 Chapitre 4

```
Bounding_chain_voisins_1 <- function(Y,k,l){ ## Entrée : bounding chain, coordonnées d'un noeud du graphe. Sortie : nombre de voisins de label 1 du noeud choisi##

  X = Y

  ##On entoure la matrice de valeurs aberrantes##
  X = rbind(rep(-99,dim(X)[2]),X)
  X = rbind(X,rep(-99,dim(X)[2]))
  X = cbind(rep(-99,dim(X)[1]),X)
  X = cbind(X,rep(-99,dim(X)[1]))

  ##On change les coordonnées du noeud d'entrée en conséquence##
  k = k+1
  l = l+1

  ##On compte le nombre de 1 autour du noeud considéré##

  N1 = 1*(X[k-1,l]==1) + 1*(X[k,l+1]==1) + 1*(X[k+1,l]==1) + 1*(X[k,l-1]==1)

  return(N1)

}

Bounding_chain_voisins_01 <- function(Y,k,l){ ## Entrée : bounding chain, coordonnées d'un noeud du graphe. Sortie : nombre de voisins de label 1 du noeud choisi##


  X = Y

  ##On entoure les matrices de valeurs aberrantes##
  X = rbind(rep(-99,dim(X)[1]),X)
  X = rbind(X,rep(-99,dim(X)[2]))
  X = cbind(rep(-99,dim(X)[1]),X)
  X = cbind(X,rep(-99,dim(X)[1]))

  ##On change les coordonnées du noeud d'entrée en conséquence##
  k = k+1
  l = l+1

  ##On compte le nombre de 2 autour du noeud considéré##

  N01 = 1*(X[k-1,l]==2) + 1*(X[k,l+1]==2) + 1*(X[k+1,l]==2) + 1*(X[k,l-1]==2)

```

```
42    return(N01)
43
44
45  }
46
47
48
49  Bounding_chain_update_HCGM <- function(Y,u,lambda){##mise à jour d'un noeud pour
        la bounding chain du modèle hard core gas##
50
51    for(i in 1:length(u)){
52
53      ##On tire un noeud uniformément dans le graphe##
54      k = sample(1:dim(Y)[1],1)
55      l = sample(1:dim(Y)[2],1)
56
57      ##On compte le nombre de voisins de label {1} et de label {0,1} = 2##
58
59      N1 = Bounding_chain_voisins_1(Y,k,l)
60
61      N01 = Bounding_chain_voisins_01(Y,k,l)
62
63      ##mise à jour selon le pseudo code fourni par Huber##
64
65      if((u[i] > lambda/(1+lambda)) | (N1>0)){
66
67        Y[k,l] = 0
68
69      }else if(N01 == 0){
70
71        Y[k,l] = 1
72
73      }else{
74
75        Y[k,l] = 2
76
77      }
78    }
79
80    return(Y)
81
82  }
83
84
85  Bounding_chain_cftp_HCGM <- function(t,lambda,Nrows,Ncols){##Effectue la méthode
        de bounding chains pour l'obtention d'un échantillon de la loi stationnaire
        pour le modèle hardcore gas de paramètre lambda et une mise à jour de type
      Gibbs pour un graphe de taille Nrows*Ncols##
86
```

```
  a = 1
  ##tirage des aléatoires##
  u = runif(t)

  ##état initial de la bounding chain##
  Y = matrix(2,Nrows,Ncols)

  ##on met à jour la bounding chain selon les choix aléatoires u##

  Y = Bounding_chain_update_HCGM(Y,u,lambda)


  ##si la bounding chain a plus d'un état sur au moins un des noeuds##

  if(sum(Y==2)!=0){


    ##après récursions, X sera la bounding chain où tout noeud n'aura qu'un
        élément##
    X = Bounding_chain_cftp_HCGM(2*t,lambda,Nrows,Ncols)
    a = 2*X[[2]]

    ##on représente la bounding chain par une matrice remplie de 0,1 et 2, où 2
        représente le label {0,1}$##
    Y = X[[1]]

    Y = Bounding_chain_update_HCGM(Y,u,lambda)

  }
    return(list(Y,a))

}
```

```
Bounding_chain_update_shift <- function(Y,u,lambda,pshift){##mise à jour d'un
    noeud pour la bounding chain du modèle hard core gas shift##


  for(i in 1:length(u)){

    ##On génère une réalisation de la Bernouilli de paramètre pshift##
    S = rbinom(1,1,pshift)

    ##On tire un noeud uniformément dans le graphe##
    k = sample(1:dim(Y)[1],1)
    l = sample(1:dim(Y)[2],1)

    ##On compte le nombre de voisins (du noeud de coordonnées k,l) de label {1}
```

```
    et de label {0,1}##

N1 = Bounding_chain_voisins_1(Y,k,l)

NO1 = Bounding_chain_voisins_01(Y,k,l)

##On vérifie le tableau des cas donné dans le chapitre d'Huber associé##

if(u[i]>=lambda/(1 + lambda)){

  Y[k,l] = 0

}else if((N1 == 0) && (NO1 == 0)){

  Y[k,l] = 1

}else if((N1 == 0) && (NO1 == 1) && (S == 0)){

  Y[k,l] = 2

}else if ((N1 == 1) && (S==0)){

  Y[k,l] = 0

}else if(N1>=2){

  Y[k,l] = 0

}else if((N1 == 0) && (NO1 == 1) && (S==1)){

  k = k+1
  l = l+1

  X = Y

  X = rbind(rep(-99,dim(X)[1]),X)
  X = rbind(X,rep(-99,dim(X)[2]))
  X = cbind(rep(-99,dim(X)[1]),X)
  X = cbind(X,rep(-99,dim(X)[1]))

  w = c(k-1,l)*(X[k-1,l] == 2) + c(k,l+1)*(X[k,l+1] == 2) + c(k+1,l)*(X[k+1,
      l] == 2) + c(k,l-1)*(X[k,l-1] == 2)

  w = w - c(1,1)
  k=k-1
  l=l-1
  Y[w[1],w[2]] = 0
  Y[k,l] = 1

```

```
61      }else if((N1==1) && (NO1 == 0) && (S == 1)){
62
63        k = k+1
64        l = l+1
65
66        X = Y
67
68        X = rbind(rep(-99,dim(X)[1]),X)
69        X = rbind(X,rep(-99,dim(X)[2]))
70        X = cbind(rep(-99,dim(X)[1]),X)
71        X = cbind(X,rep(-99,dim(X)[1]))
72
73        w = c(k-1,l)*(X[k-1,l] == 1) + c(k,l+1)*(X[k,l+1] == 1) + c(k+1,l)*(X[k+1,
              l] == 1) + c(k,l-1)*(X[k,l-1] == 1)
74
75        w = w - c(1,1)
76        k = k-1
77        l = l-1
78        Y[w[1],w[2]] = 0
79        Y[k,l] = 1
80
81
82      }else if((N1==1) && (NO1 >= 1) && (S == 1)){
83
84        k = k+1
85        l = l+1
86
87        X = Y
88
89        X = rbind(rep(-99,dim(X)[1]),X)
90        X = rbind(X,rep(-99,dim(X)[2]))
91        X = cbind(rep(-99,dim(X)[1]),X)
92        X = cbind(X,rep(-99,dim(X)[1]))
93
94        w = c(k-1,l)*(X[k-1,l] == 1) + c(k,l+1)*(X[k,l+1] == 1) + c(k+1,l)*(X[k+1,
              l] == 1) + c(k,l-1)*(X[k,l-1] == 1)
95
96        w = w - c(1,1)
97        k =k-1
98        l =l-1
99        Y[w[1],w[2]] = 2
100       Y[k,l] = 2
101
102     }
103
104   }
105
106   return(Y)
107
```

```
}

Bounding_chain_cftp_shift <- function(t,lambda,pshift,Nrows,Ncols){##Effectue la
     méthode de bounding chains pour l'obtention d'un échantillon de la loi
    stationnaire pour le modèle hardcore gas shift de paramètre lambda et une
    mise à jour de type Gibbs pour un graphe de taille Nrows*Ncols##
  a = 1
  ##tirage des aléatoires##
  u = runif(t)

  ##état initial de la bounding chain##
  Y = matrix(2,Nrows,Ncols)

  ##on met à jour la bounding chain selon les choix aléatoires u##

  Y = Bounding_chain_update_shift(Y,u,lambda,pshift)


  ##si la bounding chain a plus d'un état sur au moins un des noeuds##

  if(sum(Y==2)!=0){


    ##après récursions, X sera la bounding chain où tout noeud n'aura qu'un
        élément##
    X = Bounding_chain_cftp_shift(2*t,lambda,pshift,Nrows,Ncols)
    a = 2*X[[2]]
    ##on représente la bounding chain par une matrice remplie de 0,1 et 2, où 2
        représente le label {0,1}$##
    Y = X[[1]]

    Y = Bounding_chain_update_shift(Y,u,lambda,pshift)

  }
  return(list(Y,a))

}
```

```
Bounding_chain_update_ising <- function(Y,u,mu,beta){

  for(i in 1:length(u)){

    ##On tire un noeud uniformément dans le graphe##
    k = sample(1:dim(Y)[1],1)
    l = sample(1:dim(Y)[2],1)

    ##On compte le nombre de voisins de label {1} et de label {-1,1} = 2##
```

```
10
11      N1 = Bounding_chain_voisins_1(Y,k,l)
12
13      NO1 = Bounding_chain_voisins_01(Y,k,l)
14
15
16      ##On prépare l'entourage de la matrice par des valeurs aberrantes##
17      k = k+1
18      l = l+1
19
20      X = Y
21
22      X = rbind(rep(-99,dim(X)[1]),X)
23      X = rbind(X,rep(-99,dim(X)[2]))
24      X = cbind(rep(-99,dim(X)[1]),X)
25      X = cbind(X,rep(-99,dim(X)[1]))
26
27
28      ##On calcule le degré du noeud choisi en comptant les valeurs aberrantes
            autour de celui-ci##
29      w = 1*(X[k-1,l] == -99) + 1*(X[k,l+1] == -99) + 1*(X[k+1,l] == -99) + 1*(X[k
            ,l-1] == -99)
30
31
32      ##On met à jour selon le degré du noeud, le nombre de valeurs aberrantes et
            le nombre d'états inconnus##
33      if(NO1 == 0){
34
35        if(u[i] < exp(mu + beta*N1)/(exp(mu + beta*N1) + exp(beta*(4 - N1 - w) -
              mu))){
36
37          Y[k-1,l-1] = 1
38
39        }else{
40
41          Y[k-1,l-1] = -1
42
43        }
44
45
46      }else if(w == 0){
47
48        if(u[i] < exp(beta*N1 + mu)/(exp(beta*N1 + mu) + exp(beta*(4-N1) - mu))){
49
50          Y[k-1,l-1] = 1
51
52        }else if(u[i] > exp(beta*(N1 + NO1) + mu)/(exp(beta*(N1+NO1) + mu) + exp(
              beta*(4-N1-NO1) - mu))){
53
```

```
54          Y[k-1,l-1] = -1
55
56        }else{
57
58          Y[k-1,l-1] = 2
59
60        }
61
62
63      }else if (w==1){
64
65        if(u[i] < exp(beta*N1 + mu)/(exp(beta*N1 + mu) + exp(beta*(3-N1) - mu))){
66
67          Y[k-1,l-1] = 1
68
69        }else if(u[i] > exp(beta*(N1 + NO1) + mu)/(exp(beta*(N1+NO1) + mu) + exp(
              beta*(3-N1-NO1) - mu))){
70
71          Y[k-1,l-1] = -1
72
73        }else{
74
75          Y[k-1,l-1] = 2
76
77        }
78
79      }else if (w==2){
80
81        if(u[i] < exp(beta*N1 + mu)/(exp(beta*N1 + mu) + exp(beta*(2-N1) - mu))){
82
83          Y[k-1,l-1] = 1
84
85        }else if(u[i] > exp(beta*(N1 + NO1) + mu)/(exp(beta*(N1+NO1) + mu) + exp(
              beta*(2-N1-NO1) - mu))){
86
87          Y[k-1,l-1] = -1
88
89        }else{
90
91          Y[k-1,l-1] = 2
92
93        }
94
95      }
96    }
97
98    return(Y)
99  }
100
```

```
Bounding_chain_cftp_ising <- function(t,mu,beta,Nrows,Ncols){##Effectue la
    méthode de bounding chains pour l'obtention d'un échantillon de la loi
    stationnaire pour le modèle d'Ising de paramètre mu et beta et une mise à
    jour de type Gibbs pour un graphe rectangulaire de taille Nrows*Ncols##

  a = 1

  ##tirage des aléatoires##
  u = runif(t)

  ##état initial de la bounding chain##
  Y = matrix(2,Nrows,Ncols)

  ##on met à jour la bounding chain selon les choix aléatoires u##

  Y = Bounding_chain_update_ising(Y,u,mu,beta)


  ##si la bounding chain a plus d'un état sur au moins un des noeuds##

  if(sum(Y==2)!=0){
    ##après récursions, X sera la bounding chain où tout noeud n'aura qu'un
        élément##
    X = Bounding_chain_cftp_ising(2*t,mu,beta,Nrows,Ncols)
    a = 2*X[[2]]
    ##on représente la bounding chain par une matrice remplie de -1,1 et 2, où 2
         représente le label {-1,1}##
    Y = X[[1]]

    Y = Bounding_chain_update_ising(Y,u,mu,beta)

  }
  return(list(Y,a))

}

##permet l'obtention des histogrammes obtenus pour la comparaison de méthodes##
Isingbcomptage881 = NULL
for(i in 1:10^2){

  X = Bounding_chain_cftp_ising(1,0,1.5,8,8)[[2]]
  Isingbcomptage881 = c(Isingbcomptage881,X)
  print(i)
}

table(Isingbcomptage881)
hist(Isingbcomptage881)
```

6.5 Chapitre 5

```
Bounding_chain_voisins_1_bipartite <- function(Y,A,k){ ## Entrée : bounding chain, coordonnées d'un noeud du graphe. Sortie : nombre de voisins de label 1 du noeud choisi##

    ##On récupère des informations sur l'état actuel##
    X = Y
    V1 = X[[1]]
    V2 = X[[2]]

    ##On initialise le compteur d'états 1##
    N1 = 0

    ##numvoisin contiendra les numéros des noeuds voisins de k##
    numvoisin = (A[k,] == 1)
    a = 1:dim(A)[2]
    numvoisin = a[numvoisin]

    ##On compte le nombre de 1 autour du noeud considéré selon qu'il soit dans une partie du graphe ou l'autre##

    if(k <= dim(V1)[2]){

      for (i in numvoisin){

        N1 = N1 + 1*(V2[i - length(V1)]==1)

      }
    }else{

      for (i in numvoisin){

        N1 = N1 + 1*(V1[i]==1)

      }
    }

    return(N1)

}


Bounding_chain_voisins_01_bipartite <- function(Y,A,k){ ## Entrée : bounding chain, coordonnées d'un noeud du graphe. Sortie : nombre de voisins de label 1 du noeud choisi##
```

```
41
42      ##On récupère des informations sur l'état actuel##
43      X = Y
44      V1 = X[[1]]
45      V2 = X[[2]]
46
47      ##On initialise le compteur d'états inconnus##
48      NO1 = 0
49
50      ##numvoisin contiendra les numéros des noeuds voisins de k##
51      numvoisin = (A[k,] == 1)
52      a = 1:dim(A)[2]
53      numvoisin = a[numvoisin]
54
55      ##On compte le nombre de 1 autour du noeud considéré selon qu'il soit dans
            une partie du graphe ou l'autre##
56
57      if(k <= dim(V1)[2]){
58
59        for (i in numvoisin){
60
61          NO1 = NO1 + 1*(V2[i - length(V1)]==2)
62
63        }
64      }else{
65
66        for (i in numvoisin){
67
68          NO1 = NO1 + 1*(V1[i]==2)
69
70        }
71      }
72
73      return(NO1)
74
75  }
76
77
78  Bounding_chain_update_HCGM_bipartite <- function(Y,A,u,lambda){
79
80    for(i in 1:length(u)){
81
82      ##On tire un noeud uniformément dans le graphe##
83      ij = sample(1:dim(A)[1],1,replace = TRUE)
84
85      ##On compte le nombre de voisins de label {1} et de label {0,1} = 2##
86
87      N1 = Bounding_chain_voisins_1_bipartite(Y,A,ij)
88
```

```
89      N01 = Bounding_chain_voisins_01_bipartite(Y,A,ij)
90
91      ##mise à jour selon le pseudo code fourni par Huber##
92
93      if((u[i] > lambda/(1+lambda)) | (N1>0)){
94
95        if(ij <= length(Y[[1]])){Y[[1]][ij] = 0}else{Y[[2]][ij - length(Y[[1]])] =
              0}
96
97      }else if(N01 == 0){
98
99        if(ij <= length(Y[[1]])){Y[[1]][ij] = 1}else{Y[[2]][ij - length(Y[[1]])] =
              1}
100
101     }else{
102
103       if(ij <= length(Y[[1]])){Y[[1]][ij] = 2}else{Y[[2]][ij - length(Y[[1]])] =
              2}
104
105     }
106   }
107
108   return(Y)
109
110 }
111
112
113
114 Bounding_chain_HCGM_Bipartite <- function(t,A,lV1,lV2,lambda){ ##t = nombre de
      pas effectués, A = matrice d'adjacence du graphe biparti, V1 = ensemble des
      noeuds d'un côté du graphe biparti, V2 = ensemble des noeuds de l'autre côté
       du graphe biparti
115
116   #compteur d'uniformes utilisées##
117   a = 1
118
119   ##Initialisation du graphe##
120   V1 = matrix(2,1,lV1)
121   V2 = matrix(2,1,lV2)
122
123   ##tirage des aléatoires##
124   u = runif(t)
125
126   ##on représente le graphe par une liste de 2 vecteurs##
127   Y = list(V1,V2)
128
129   ##On met à jour le graphe selon les choix aléatoires##
130   Y = Bounding_chain_update_HCGM_bipartite(Y,A,u,lambda)
131
```

```
132
133   if (sum(Y[[1]]==2)>=1 | sum(Y[[2]]==2)>=1){ ##s'il existe encore au moins un
          noeud de label inconnu##
134
135     ##la récursion s'opère##
136     X = Bounding_chain_HCGM_Bipartite(2*t,A,lV1,lV2,lambda)
137
138     ##On met à jour le compteur##
139     a = 2*X[[3]]
140
141     ##On met à jour la chaîne résultant de la récursion selon les choix
            aléatoires effectués##
142     Y = Bounding_chain_update_HCGM_bipartite(X[1:2],A,u,lambda)
143   }
144
145   ##On rend le résultat##
146   return(c(Y,a))
147
148 }
149
150 ##permet l'obtention des histogrammes obtenus pour la comparaison de modèles##
151 A1bcomptage2 = c()
152 for (i in 1:10^4){A1bcomptage2 = c(A1bcomptage2,Bounding_chain_HCGM_Bipartite(t,
        A1,lV1,lV2,2)[[3]])
153 print(i)}
154 mean(A1bcomptage2)
155 table(A1bcomptage2)
156 hist(A1bcomptage2)
```

7 Bibliographie

— Mark L. Huber, *Perfect Simulation*, CRC Monographs on Statistics & Applied Probability, ISBN : 978-1482232448, 2016

— François Baccelli et Pierre Brémaud, *Elements of queuing theory*, Ed. Springer Verlag (1994, seconde édition en 2003)

— `http://lmrs.math.cnrs.fr/Persopage/Chagny/AgregTP3ScilabCorrection2016_2017.pdf`, exercice 11, preuve du théorème 7.

— `https://math.unice.fr/~rcatelli/Cours_2018/Integration/DM2_20172018_correction.pdf`, volume de la boule unité en dimension d