La simulation parfaite

N.T.Viet M.Paul Encadrant : Christian Robert

29 juin 2007

Résumé

La simulation parfaite est une technique permettant d'obtenir des échantillons suivant une distribution (ou loi) donnée. On l'appelle "parfaite" par opposition aux méthodes approchées comme MCMC qui fonctionnent avec des chaînes de Markov.
Dans le chapitre 1, on commencera par rappeller quelques résultats sur les chaînes de Markov, puis l'on décrira deux algorithmes approchés basés sur la méthode MCMC. Dans le chapitre 2, on décrira deux algorithmes de simulation "parfaits". On s'intérressera ensuite dans le chapitre 3 à étudier les temps d'exécution de ces algorithmes. On terminera dans le chapitre 4 par une application au modèle d'Ising, et quelques expériences que l'on a réalisé.

Table des matières

1 Aperçu

1.1 Chaînes de Markov

Bien que cela soit bien connu, nous donnons ici la définition des chaînes de Markov et quelques résultats importants pour la clareté du travail. Il est essentiel de garder à l'esprit que certains résultats annoncés ici ne sont vrais que dans le cas d'un ensemble d'états fini (qui est le seul cas auquel on s'intérressera). Pour des preuves détaillées de ces résultats et pour des résultats plus généraux sur les chaînes de Markov et les chaînes de Markov réversibles, vous pouvez en lire plus dans des livres sur les chaînes de Markov.

Définition 1 *Soit $\mathbb{P}$ une $E \times E$ matrice, où E est un ensemble fini d'objets. On dit que $\mathbb{P}$ est une matrice stochastique sur E si elle satisfait :*
(i) $0 \leq \mathbb{P}(x, y) \leq 1 \qquad \forall x, y \in E$
(ii) $\forall x \in E, \qquad \sum_{y \in E} \mathbb{P}(x, y) = 1$

Définition 2 *Soit $\mathbb{P}$ une matrice stochastique (voir déf.1) sur E, et soit $(X_n)_{n \in \mathbb{N}}$ un processus aléatoire à valeurs dans E . On dit que $(X_n)_{n \in \mathbb{N}}$ est une chaîne de Markov de matrice de transition (ou noyau) $\mathbb{P}$ si $\forall n \in \mathbb{N}$, la loi conditionnelle de X_{n+1} sachant $(X_0, X_1, ..., X_n)$ est $\mathbb{P}(X_n, .)$. Ou de façon équivalente :*

$$P(X_{n+1} = y | X_0 = x_0, X_1 = x_1, ..., X_n = x_n) = \mathbb{P}(x_n, y)$$

pour tout $x_0, x_1, ..., x_n, y$ satifaisant $P(X_0 = x_0, X_1 = x_1, ..., X_n = x_n) > 0$

Remarque : Par convention on écrit $P_x(A)$ pour $P(A|X_0 = x)$ et $\mathbb{P}_n$ pour $\mathbb{P}^n$, ou $\mathbb{P}_n = \mathbb{P}_{n-1} * \mathbb{P}$. Ainsi

$$P(X_n = y | X_0 = x) = P_x(X_n = y) = \mathbb{P}_n(x, y)$$

Une autre notion très utile est le temps d'arrêt dont la définition est la suivante :

Définition 3 *Soit $\mathcal{F}_n$ la tribu engendrée par $X_0, X_1, ..., X_n$. Alors une variable aléatoire T à valeurs entières est appelée temps d'arrêt si pour tout $n \in \mathbb{N}$:*

$$\{T = n\} \in \mathcal{F}_n$$

Définition 4 *Une chaîne est dite irréductible si pour tout $x, y \in E$, $\exists n \in \mathbb{N}$ tel que $\mathbb{P}_n(x, y) > 0$.*
Pour un état $x \in E$ on pose $N_x = \sum_{n=0}^{\infty} 1_{\{X_n = x\}}$, et $H_x = min\{n | X_n = x\}$. On a alors :
(i) Si $P_x(H_x < \infty) = 1$ alors $N_x = \infty$ $\quad P_x - p.s$ et x est appelé état récurrent. Si de plus $E_x(H_x) < \infty$ alors x est appelé récurrent positif, sinon il est appelé récurrent nul.
(ii) Si $P_x(H_x < \infty) < 1$ alors $N_x < \infty$ $\quad P_x - p.s$ et x est dit transitoire.
(iii) On pose $L_x = \{n \geq 0 : \mathbb{P}_n(x, x) > 0\}$. Alors la période $d(x)$ de x est le pgcd de L_x.

Ici, dans notre cas, rappelez-vous que E est fini. Nous avons ainsi le théorème :

Théorème 1 *Pour une chaîne de Markov irréductible, tous les états sont récurrents positifs. De plus, ils ont la même période d (qui est appelée période de la chaîne). Si $d = 1$, la chaîne est dite apériodique.*

Définition 5 *Soit π une mesure positive sur E telle que $\forall x \in E, \pi(x) < \infty$ et π n'est pas partout nul. On dit que π est une mesure invariante de matrice de transition $\mathbb{P}$ si c'est le vecteur propre à gauche de la valeur propre $\lambda = 1$ de la matrice $\mathbb{P}$, ou autrement dit :*

$$\forall y \in E \qquad \pi(y) = \sum_{x \in E} \pi(x) \mathbb{P}(x, y)$$

π sera appelé mesure de probabilité invariante si de plus on a :

$$\sum_{x \in E} \pi(x) = 1$$

Théorème 2 *Supposons que nous avons une chaîne récurrente positive sur un espace fini E de noyau $\mathbb{P}$. Alors il existe une unique mesure de probabilité invariante (qui est toujours notée π).*

Les deux théorèmes suivants sont des plus importants, et sont fondamentaux pour la méthode MCMC.

Théorème 3 *Supposons que l'on ait une chaîne récurrente positive et apériodique sur un espace fini d'états E, de noyau $\mathbb{P}$. Alors $\forall x \in E$:*

$$\sum_{y \in E} |\mathbb{P}_n(x, y) - \pi(y)| \underset{n \to \infty}{\longrightarrow} 0$$

Théorème 4 *Supposons que l'on ait une chaîne récurrente positive et apériodique sur un espace fini d'états E, de noyau $\mathbb{P}$, et h une fonction sur E telle que $\sum_{x \in E} h(x)\pi(x) < \infty$ (ou autrement dit $E_\pi(h(X_0)) < \infty$). Alors,*

$$\frac{1}{n} \sum_{k=0}^{n} h(X_k) \underset{n \to \infty}{\longrightarrow} E_\pi(h(X_0))$$

Remarque : En fait c'est juste un corollaire du théorème ergodique plus général. Mais l'on voit que pour la plupart des applications (au moins dans cet exposé), c'est suffisant.

Parfois, il est bien pratique d'utiliser ceci :

Définition 6 *Soit μ une mesure positive sur un espace E telle que $\forall x \in E, \mu(x) < \infty$ et μ n'est pas toujours nul. On dit que μ est réversible si*

$$\forall x, y \in E, \qquad \mu(x)\mathbb{P}(x,y) = \mu(y)\mathbb{P}(y,x)$$

Cela est appelé la condition de balance détaillée.

Proposition 1 *Toute mesure réverible est invariante.*

Définition 7 *Soit $P(.)$ et $Q(.)$ deux mesures de probabilité sur E. Alors*

$$|P - Q|_{var} = \sup_{S \subset E} |P(S) - Q(S)| \tag{1}$$

$$= \frac{1}{2} \sum_{x \in E} |P(x) - Q(x)| \tag{2}$$

Définition 8 *Pour une chaîne de Markov, on definit :*

$$d(k) = \max_{x \in E} |P^k(x,.) - \pi|_{var}$$

$$\overline{d}(k) = \max_{x,y \in E} |P^k(x,.) - P^k(y,.)|_{var}$$

$$\overline{s}(k) = \max_{x,y \in E} \left\{ 1 - \frac{P^k(x,y)}{\pi(y)} \right\}$$

Lemme 1 *Pour toute chaîne de Markov, $\overline{d}(.)$, $d(.)$ et $\overline{s}(.)$ sont sous-multiplicatives et pour une chaîne de Markov reversible :*

$$d(k) \leq \overline{d}(k) \leq 2d(k)$$

$$\overline{d}(k) \leq \overline{s}(k)$$

$$\overline{s}(2k) \leq 1 - (1 - \overline{d}(k))^2$$

1.2 Monte Carlo par chaînes de Markov

D'abord, il faut que nous parlions de la méthode Monte Carlo. Une des applications les plus simples est le calcul d'une intégrale finie, que l'on peut voir comme l'espérance d'une variable aléatoire, ou encore le calcul de l'espérance elle-même. On peut faire cela de manière deterministe, ou approchée par des méthodes analytiques, mais ces méthodes sont souvent trop coûteuses. Mais grâce à la loi des grands nombres, on a :

$$\frac{1}{n} \sum_{k=0}^{n} f(X_k) \xrightarrow[n \to \infty]{p.s} E(f(X_0))$$

où $\{X_n\}_{n\in\mathcal{N}}$ est une suite de variables aléatoires i.i.d. et f est une fonction intégrable.

Ainsi, il suffit de d'engendrer suffisament de variables aléatoires X_i pour approcher $E(f(X))$ par la moyenne de $f(X_i)$. C'est l'idée de la méthode Monte Carlo.

Ainsi, le problème le plus important pour appliquer la méthode Monte Carlo est de savoir comment engendrer X_i. Afin d'engendrer ces échantillons, nous avons besoin la *fonction de répartition* (fr) F ou bien de la *densité de probabilité* (dp) f de la variable aléatoire. Mais parfois, avec un f donné, il est très difficile d' engendrer des échantillons (il arrive souvent que f ne soit connue qu'à une constante de normalisation près, qui soit très difficile à calculer). La méthode MCMC est une solution à ce problème. Dans ce travail, nous nous concentrons uniquement sur le cas d'un ensemble d'états fini. Plutôt que des tirages indépendants à partir de f, on peut à la place utiliser une chaîne ne Markov dont la mesure invariante est f, qui commence avec un X_0 convenablement choisi. Alors, d'après le théorème 4, on peut utiliser la moyenne de cette suite comme une approximation non biaisée de ce que l'on veut.

Remarque : En fait, notre chaîne de Markov doit satisfaire des conditions plus fortes que l'irréductibilité et l'apériodicité. Dans notre cas, où l'espace d'états est en fait fini, ces deux conditions sont suffisantes. Grâce au théorème 1, une chaîne ayant un nombre fini d'états,et irréductible est aussi récurrente positive, et ainsi nous pouvons appliquer le théorème ergodique.

C'est juste l'idée de comment cela fonctionne. En fait, les échantillons initiaux peuvent introduire un grand biais, puisque les différences entre les distributions des premiers échantillons et f peut être très grande. Mais grâce au théorème 3, la différence entre la distribution de X_n et f tend vers 0 quand n tend vers ∞. Donc il suffit d'attendre jusqu'à un n assez grand, d'oublier les n premiers échantillons, et utiliser le reste de la chaîne comme approximation de la variable aléatoire dont la dp est f.

Maintenant, voici les deux algorithmes les plus importants qui utilisent les chaînes de Markov Monte Carlo.

1.2.1 Metropolis-Hastings

Maintenant, supposons que nous ayons une dp π sous la main, et que nous voulions tirer des échantillons de cette dp. Mais parfois il est difficile de faire cela directement (en utilisant l'inverse de la fonction de dp correspondante par exemple) alors voici une solution. Rappelons-nous que nous avons toujours considéré un espace d'états fini (peut-être gros mais fini). Soit T une matrice stochastique sur $E \times E$, et $T(x, y) > 0$ quand $T(y, x) > 0$ pour toute paire de x, y dans E. Alors soit une chaîne de Markov ayant la matrice de transition $\mathcal{P}$ suivante :

$$\mathcal{P}(x, y) = T(x, y)min\{1, \frac{\pi(y)T(y, x)}{\pi(x)T(x, y)}\} \qquad \forall x, y \in E, x \neq y$$

$$\mathcal{P}(x, x) = 1 - \sum_{y\neq x} \mathcal{P}(x, y) \qquad \forall x \in E$$

Proposition 2 *La matrice $\mathcal{P}$ définie ci-dessus est stochastique, et une chaîne qui a pour matrice de transition $\mathcal{P}$ est réversible et a π comme distribution stationnaire.*

D'abord, nous voyons que $\forall x, y \in E, x \neq y$ entraîne $\mathcal{P}(x,y) \leq T(x,y)$. Ainsi,

$$\sum_{y \neq x} \mathcal{P}(x,y) \leq \sum_{y \neq x} T(x,y) < 1$$

ce qui signifie que $0 \leq T(x,x) \leq \mathcal{P}(x,x)$ et donc $\mathcal{P}$ est une matrice stochastique. Pour le reste, il est facile de vérifier que $\mathcal{P}$ satisfait la condition de balance détaillée et a pour distribution stationnaire π (Grâce à la proposition 1). Nous avons donc l'algorithme suivant pour obtenir des échantillons :

Algorithm METROPOLIS-HASTLING :

Commencer avec x
Etape de mise à jour
Proposer y selon $T(x,y)$
Accepter y avec probabilité $\min(1, \frac{\pi(y)T(y,x)}{\pi(x)T(x,y)})$

Si la matrice T proposée est symétrique, alors on obtient l'algorithme de Metropolis original.

1.2.2 Simulateur de Gibbs

Un autre algorithme peut être utilisé pour le problème du dernier paragraphe si l'échantillon est un vecteur ayant plusieurs composantes et si l'on peut effectivement échantillonner à partir de la distribution conditionnelle d'une composante quand toutes les autres sont fixées, c'est l'algorithme de Gibbs. Plus formellement, donnons quelques définitions.

D'abord, nous supposerons que notre échantillon est un vecteur $\mathbf{x} = (x_1, x_2, ..., x_n)$ de dp $\pi(\mathbf{x})$ et nous noterons $\pi(.|\mathbf{x}_{-i})$ la distribution conditionnnelle du *i-ème* élément de $\mathbf{x}$ quand tous les autres éléments sont fixés (comme nous l'avons supposé, on peut effectivement échantillonner à partir de cette distribution). Considérons les matrices stochastiques P_i définies par :

$$P_i(\mathbf{x}, \mathbf{y}) = \pi(y_i|\mathbf{x}_{-i}) \qquad \text{si } \mathbf{x}, \mathbf{y} \text{ ne diffèrent que par leur } i\text{-ème élément}$$

et $P_i(\mathbf{x}, \mathbf{y}) = 0$ sinon.

On voit facilement que chaque P_i a pour distribution stationnaire π. Mais malheureusement, aucune d'entre elles n'est irréductible. Nous devons donc les rendre irréductibles en considérant :

$$P_r = \frac{1}{n} \sum_{i=1}^{n} P_i$$

$$P_s = \prod_{i=1}^{n} P_i$$

Proposition 3 *P_r et P_s sont irréductibles, apériodiques et ont pour distribution stationnaire π. De plus, P_r est réversible.*

Ainsi P_r est la matrice de transition correspondant à un algorithme d'échantillonnage appelé *random Gibbs sampler*, et P_s correspond à l'algorithme *sequential Gibbs sampler*. Dans *random Gibbs sampler*, on commence avec un vecteur initial $\mathbf{x}_0$, et on met alors à jour le *i-ème* élément de $\mathbf{x}_t$ suivant $\pi(.|\mathbf{x}_{-i})$. Dans

sequential Gibbs sampler, on fait presque la même chose, sauf qu'à la place d'engendrer un indice aléatoire i, on prend $i = t \bmod n$ de façon déterministe.

Algorithme GIBBS :(version aléatoire)
Commencer avec $\mathbf{x} = (x_1, x_2, ..., x_n)$
Etape de mise à jour :
Tirer i de façon uniforme dans $\{1, 2, ..., n\}$
Tirer y_i selon $\pi(.|\mathbf{x}_{-i})$
Soit $\mathbf{y} = (x_1, ..., y_i, ..., x_n)$

Algorithme GIBBS :(version séquentielle)
Commencer avec $\mathbf{x} = (x_1, x_2, ..., x_n)$, $i = 0$
Etape de mise à jour :
$i := (i + 1) \mod n$
Tirer y_i selon $\pi(.|\mathbf{x}_{-i})$
Soit $\mathbf{y} = (x_1, ..., y_i, ..., x_n)$

1.2.3 Taille d'échantillon effective

Comme expliqué avant, nous pouvons utiliser $\frac{1}{n}\sum_{k=0}^{n} f(X_k)$ comme estimation non biasée de $E(X)$. Et pour la variance ? Nous avons que

$$var\left\{\frac{\sum_{k=0}^{n} f(X_k)}{n}\right\} = \frac{1}{n^2}(\sum_{k=0}^{n} var\{f(X_k\} + 2 \cdot \sum_{0 \le i < j \le n} cov\{h(X_i), h(X_j)\})$$

Supposons que la chaîne soit équilibrée, et $var\{f(X_0)\} = \sigma^2$, $corr\{f(X_0), f(X_k)\} = \rho_k$ nous avons

$$var\{f(X_k)\} = var\{f(X_0)\} = \sigma^2$$
$$cov\{f(X_i), f(X_j)\} = cov\{f(X_0), f(X_{j-i})\} = \sigma^2 \rho_{j-i}$$

En substituant dans la formule précedente :

$$var\left\{\frac{\sum_{k=0}^{n} f(X_k)}{n}\right\} = \frac{1}{n}\sigma^2 \left[1 + 2 \cdot \sum_{k=1}^{n-1}\left(1 - \frac{k}{n}\right)\rho_k\right] \quad (3)$$
$$\approx \frac{1}{n}\sigma^2 \left[1 + 2\sum_{k=1}^{\infty}\rho_k\right] \quad (4)$$

Définissons le temps d'autocorrélation intégré de $h(x)$ comme

$$\tau_{int}(h) = \frac{1}{2} + \sum_{k=1}^{\infty}\rho_k$$

Remarquez que cette quantité dépend de la chaîne de Markov canonique et de la fonction h. Appelons $\hat{h}$ la valeur estimée de $E(h)$, alors nous avons

$$var(\hat{h}) = \frac{2\tau_{int}(h)}{m}\sigma^2$$

Ainsi au sens de la variance, nous pouvons traiter m échantillons engendrés par une chaîne de Markov comme $\frac{m}{[2\tau_{int}(h)]}$ échantillons iid de la même distribution.

$\frac{m}{[2\tau_{int}(h)]}$ est souvent ce à quoi l'on réfère par "Taille d'échantillon éffective" (tee), qui est habituellement une utile mesure d'éffectivité de l'algorithme. Cette valeur dépend à la fois de la chaîne canonique et de la fonction h. Cette valeur peut être bornée. En fait, on peut prouver que pour un échantillon avec un nombre fini d'états, il existe

$$\tau_{int} = sup_{h \in L^2_\pi} \tau_{int}(h) = \frac{1+\lambda}{2(1-\lambda)}$$

où λ est la deuxième plus grande valeur propre de la matrice de transition (la plus grande est bien entendu 1).

1.2.4 Quelques exemples

Exemple 1 *Un exemple de chaîne de Markov.*

Considérons la matrice de transition suivante : $\begin{pmatrix} 2/3 & 2/9 & 1/9 \\ 1/3 & 1/2 & 1/6 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$

ce qui correspond à une chaîne de Markov avec 3 états numérotés 1,2,3. Cette chaîne est clairement irréductible et donc récurrente positive. Après quelques calculs, il apparaît que cette chaîne est réversible et sa distribution stationnaire est $(\frac{1}{2}, \frac{1}{3}, \frac{1}{6})$.
Il n'est pas difficile de voir que cette chaîne est obtenue en appliquant l'algorithme Metropolis-Hastling avec la matrice $\begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{pmatrix}$.

Exemple 2 *Distribution de Poisson :*

Dans cet exemple, nous allons trouver un moyen d'échantillonner suivant une distribution de Poisson de paramètre $\lambda < 1$, arrondie à n (c'est-à-dire $\pi(x) = P(X = x)$ égal à $P_{Poisson_\lambda}(x)$ pour $x \leq n$ et $P_{Poisson_\lambda}([x, \infty])$ pour $x = n$).
Pour ce faire, considérons la matrice T proposée avec $T(x, x+1) = T(x, x-1) = \frac{1}{2}$ pour $0 < x < n$, $T(0,0) = T(0,1) = T(n,n) = T(n,n-1) = \frac{1}{2}$. Ainsi, en appliquant Metropolis-Hastling, nous obtenons la matrice de transition :

$$\mathcal{P}(x, x-1) = T(x, x-1) \min\left\{1, \frac{\pi(x-1)}{\pi(x)}\right\} = \frac{1}{2} \qquad x > 0$$

$$\mathcal{P}(x, x+1) = T(x, x+1) \min\left\{1, \frac{\pi(x-1)}{\pi(x)}\right\} = \frac{\lambda}{2(n+1)} \qquad x < n-1$$

$$\mathcal{P}(n-1, n) = T(x, x-1) \min\left\{1, \frac{\pi(x-1)}{\pi(x)}\right\} = \frac{1}{2} \sum_{k=n}^{\infty} \frac{n! \lambda^{k-n+1}}{k!}$$

les autres valeurs de $\mathcal{P}$ (c'est-à-dire $\mathcal{P}(x, x)$) peuvent être calculée à partir de la formule.

2 La simulation parfaite

On a vu comment l'on pouvait obtenir de manière approchée des échantillons suivant une distribution π donnée, avec la méthode MCMC. La simulation parfaite permet d'obtenir des échantillons répartis exactement suivant la distribution souhaitée. L'idée des algorithmes "parfaits" est de faire tourner une chaîne de Markov, comme avec MCMC, et de trouver un moyen de savoir combien de temps il faut faire tourner notre chaîne de Markov pour qu'elle devienne distribuée exactement suivant π. L'idée de la coalescence permet de faire cela, et nous donnera l'algorithme de Propp-Wilson, et l'idée de Fill d'accepter ou non une chaîne de Markov, en faisant un algorithme de rejet, sera également une solution.

2.1 Algorithme de Propp-Wilson

L'idée de cet algorithme est de calculer simultanément N copies de la chaîne de Markov, pour chaque état initial possible, en prenant la même fonction de mise à jour, et la même suite de nombre aléatoires, et d'attendre que ces chaînes aient coalescé, c'est-à-dire qu'elle aient atteint le même état. Quand deux chaînes ont coalescé, elles suivent exactement le même chemin. On a ainsi perdu l'information de l'endroit d'où l'on était partit.
Mais cela ne fonctionne pas directement, comme le montre l'exemple suivant : Considérons deux chaînes de Markov sur l' ensemble d'états $\{0, 1\}$, et de matrice transition $\begin{pmatrix} 0.5 & 0.5 \\ 1.0 & 0.0 \end{pmatrix}$, l'une partant de 0 et l'autre de 1. On voit facilement que ces deux chaînes ne peuvent coalescer qu'en 0, ce qui donnerait donc la distribution $\pi(0) = 1$ et $\pi(1) = 0$. Pourtant, on a $\pi(0) = 2/3$ et $\pi(1) = 1/3$.
La solution consiste à faire fonctionner la chaîne de Markov du passé vers le présent, et non du présent vers le futur. Plus précisément, si l'on appelle $\varphi : S \times \mathcal{U} \to S$ la fonction de mise à jour (et U est une variable aléatoire à valeur dans $\mathcal{U}$), telle que $Prob(\varphi(x, U) = y) = \mathcal{P}(x, y)$, et si l'on définit récursivement F par :

$$F_0(x) = x$$

$$F_{t-1}(x, U_0, U_{-1}, ..., U_{t+1}, U_t) = F_t(\varphi(x, U_t), U_0, U_{-1}, ..., U_{t+1})$$

alors il y a coalescence de toutes les chaines de Markov à partir du temps t si et seulement si la fonction $x \to F_t(x, U_0, ..., U_{t+1})$ est constante. Et dans ce cas, on a pour tout $s \leq t$ et tout x, y, $F_s(x, U_0, U_{-1}, ..., U_{s+1}) = F_s(y, U_0, U_{-1}, ..., U_{s+1})$, et donc on a bien la distribution stationnaire, puisque $F_{-\infty}$ l'a.
Il est cependant impossible en pratique de réaliser cet algorithme tel quel, puisque généralement l'ensemble S des états est trop grand (sinon, on pourrait directement tirer un échantillon suivant la distribution de probabilité voulue). Pour mettre en pratique cet algorithme, on fait l'hypothèse (raisonnable) que l'ensemble des états est partiellement ordonné par $\leq$, tel que la fonction de mise à jour soit croissante (pour tous u,x,y, $\varphi(x, u) \leq \varphi(y, u)$ dès que $x \leq y$), et qu'il existe un état maximal et un état minimal. Il suffit alors de calculer seulement deux chaînes de Markov, l'une partant du minimum, et l'autre partant du maximum, et de leur appliquer l'algorithme. Quand ces deux chaînes ont coalescé, alors toutes les autres aussi, et donc on a encore le résultat souhaité.

Algorithme de Propp-Wilson :

$k \leftarrow 0$
Faire
$U(k) \leftarrow Rand()$ // nombre aléatoire entre 0 et 1
$X \leftarrow 0$// minimum
$Y \leftarrow 1$ // maximum
Pour i allant de k à 0
$X \leftarrow \varphi(X, U(i))$
$Y \leftarrow \varphi(Y, U(i))$
Fin pour
$k \leftarrow k + 1$
Tant que $X \neq Y$
Rendre X

2.2 Preuve de convergence de l'algorithme

Intuitivement, nous pouvons supposer qu'il y a une chaîne imaginaire, mise à jour suivant la même condition que nos deux chaînes, et qui commence au temps $-\infty$ dans un état distribué suivant π. Alors au temps présent, l'état de cette nouvelle chaîne sera également distribué suivant π. De plus, si nos deux chaînes ont coalescé, alors par monotonie de la fonction de mise à jour, et avec l'hypothèse que toutes les chaînes suivent la même condition, on a que l'état courant de nos chaînes est le même que celui de la chaîne imaginaire, donc sera distribué également suivant π (puisque le temps de coalescence est presque sûrement fini).
Plus formellement, nous allons prouver que la distribution que nous obtenons est arbitrairement proche de π. Appelons π' la distribution que nous obtenons par l'algorithme de Propp-Wilson, T le temps de coalescence, et X l'échantillon que l'on obtient. Donc $X \sim \pi'$. Comme $T < \infty$ p.s., pour tout $\epsilon > 0$ on peut trouver un t tel que $P[T > t] < \epsilon$.
Maintenant, considérons une chaîne imaginaire qui commence au temps $-t$ avec l'etat initial distribué suivant π, alors l'état courant de la chaîne virtuelle est $X_0 \sim \pi$.
Sur l'évènement $T \leq t$, l'état dans lequel était la chaîne imaginaire au temps $-T$ n'a aucune importance (puisque $T \leq t$, cela arrive après le début de la chaîne), par monotonie de la mise à jour, $X_0 = X$. Donc $\{X_0 \neq X\} \subset \{T > t\}$. Ainsi :

$$P(X_0 \neq X) \leq P(T > t) < \epsilon$$

Il est bien connu que $|P - Q|_{var} = \min_{Y \sim P, X \sim Q} P(X \neq Y)$ donc

$$\epsilon > P(X \neq X_0) > \min_{Y \sim \pi, X \sim \pi'} P(X \neq Y) = |\pi - \pi'|_{var}$$

Donc π' est aussi proche que l'on veut de π, donc est égal à π.

2.3 Algorithme de Fill

L'algorithme de Propp-Wilson permet de tirer des échantillons indépendants suivant la distribution de probabilité π, mais il faut pour cela attendre la coalescence. Si l'on arrête l'algorithme au bout d'un temps fixé à l'avance, les résultats

qu'il donnera seront biaisés en faveur des échantillons qui coalescent plus rapidement. L'algorithme de Fill permet d'éviter ce problème : il donne encore des échantillons indépendants distribués exactement suivant la distribution de probabilités π, et il donne des résultats non biaisés si on l'interrompt. Afin de présenter cet algorithme, nous aurons besoin de quelques notations et résultats, et tout d'abord de l'algorithme de rejet.

2.3.1 Algorithme de rejet

L'algorithme de rejet permet de tirer un échantillon avec une dp g, sachant que l'on sait tirer un échantillon avec une dp f toujours non nulle, et que l'on sait calculer $\frac{g(x)}{L \cdot f(x)}$ pour un x donné (L est une constante que l'on choisis suffisamment grande ($\geq \sup_y \frac{g(y)}{f(y)}$)). L'algorithme est le suivant :
On tire un échantillon x suivant la dp f.
On accepte x avec la probabilité $\frac{g(x)}{L \cdot f(x)}$, sinon on recommence.

Cela produit bien un échantillon avec la dp g, car si l'on appelle p_x la probabilité d'obtenir l'échantillon x avec cet algorithme, on a :

$$p_x = \frac{f(x) \cdot \frac{g(x)}{L \cdot f(x)}}{\sum_y f(y) \cdot \frac{g(y)}{L \cdot f(y)}} = g(x)$$

2.3.2 L'algorithme

L'idée de l'algorithme de Fill est de corriger l'algorithme MCMC pour qu'il devienne parfait, en faisant un algorithme de rejet. On supposera pour cela que notre chaîne de Markov admet un renversement de temps.

Notation 1 *Nous noterons $\tilde{P}(x,y) = \frac{\pi(y) \cdot P(y,x)}{\pi(x)}$ le renversement de temps de la matrice de transition P de distribution stationnaire π.*

Notation 2 *Si la matrice $\tilde{P}$ est croissante (i.e. pour tous x,y, $x \leq y \Rightarrow \tilde{P}(x,.) \leq \tilde{P}(y,.)$), alors il existe un "haut noyau" $K_{x,y}(.,.)$ tel que $\{y' \in S | x' \leq y'\}$ est $K_{x,y}(x',.)$-mesurable, et on a l'identité :*

$$\tilde{P}(y,y') = \sum_{x'} \tilde{P}(x,x') \cdot K_{x,y}(x',y')$$

Pour pouvoir utiliser l'algorithme de Fill, on devra supposer que l'ensemble des états est partiellement ordonné, de telle façon que $\tilde{P}$ soit croissante, et qu'il existe un unique minimum $\hat{0}$ et un unique maximum $\hat{1}$.

L'algorithme fonctionne en deux temps : On commence par prendre une chaîne de Markov X de matrice de transition P et partant de $\hat{0}$, que l'on calcule jusqu'au temps t, en conservant en mémoire la trajectoire $(X_0, ..., X_t)$. Notons $\tilde{X} = (X_t, ..., X_0)$ la chaîne renversée (qui est une chaîne de Markov de matrice de transition $\tilde{P}$ par hypothèse). On considère maintenant une chaîne de Markov $\tilde{Y}$ partant de $\hat{1}$. On passe d'un état y à un état y' avec la probabilité $K_{x,y}(x',y')$, où $x \to x'$ est la transition observée pour $\tilde{X}$. Si $\tilde{Y}_t = \hat{0}$, alors la simulation est finie, et l'on rend X_t, sinon, on recommence, en augmentant éventuellement t.

Pour démontrer la validiter de cet algorithme, on peut voir cet algorithme comme un algorithme de rejet, où $g = \pi$, et $f(z) = P^t(\hat{0}, z) = Prob(X_t = z | X_0 = \hat{0})$.
Il faut alors vérifier que $Prob(\tilde{Y}_t = 0 | \tilde{Y}_0 = \hat{1}, X_0 = \hat{0}, X_t = z)$ est bien de la forme $\frac{\pi(z)}{L \cdot P^t(\hat{0},z)}$.

Lemme 2 *Pour tout t, x, et y, on a* $\tilde{P}^t(x,y) = \frac{\pi(y) \cdot P^t(y,x)}{\pi(x)}$.
Si $\tilde{P}$ *est croissant, alors* $\tilde{P}^t$ *aussi.*

D'après ce lemme, on a $\frac{\pi(z)}{P^t(\hat{0},z)} = \frac{\pi(\hat{0})}{\tilde{P}^t(z,\hat{0})} \leq \frac{\pi(\hat{0})}{\tilde{P}^t(\hat{1},\hat{0})} = L$, car l'on suppose que $\tilde{P}$ est croissant.
On a alors $\frac{\pi(z)}{L \cdot P^t(\hat{0},z)} = \frac{\tilde{P}^t(\hat{1},\hat{0})}{\tilde{P}^t(z,\hat{0})}$, et donc la convergence de l'algorithme de Fill se déduit du Lemme suivant :

Lemme 3 $Prob(\tilde{Y}_t = \hat{0} | \tilde{X}_0 = z, \tilde{X}_t = \hat{0}, \tilde{Y}_0 = \hat{1}) = \frac{\tilde{P}^t(\hat{1},\hat{0})}{\tilde{P}^t(z,\hat{0})}$

Preuve : On a

$$Prob(\tilde{Y}_t = \hat{0} | \tilde{X}_0 = z, \tilde{X}_t = \hat{0}, \tilde{Y}_0 = \hat{1}) = \frac{Prob(\tilde{Y}_t = \hat{0}, \tilde{X}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1})}{Prob(\tilde{X}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1})} \tag{5}$$

$$= \frac{Prob(\tilde{Y}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1})}{Prob(\tilde{X}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1})} \tag{6}$$

Or,

$$Prob(\tilde{Y}_t = y' | \tilde{Y}_0 = y, \tilde{X}_0 = x, \tilde{X}_t = x') = \sum_{x'} \tilde{P}(x,x') \cdot K_{x,y}(x',y') = \tilde{P}(y,y')$$

puisque $(\tilde{X}, \tilde{Y})$ est une chaîne de Markov telle que

$$Prob(\tilde{X}_{t+1} = x', \tilde{Y}_{t+1} = y' | \tilde{X}_t = x, \tilde{Y}_t = y) = \tilde{P}(x,x') \cdot K_{x,y}(x',y')$$

Donc $Prob(\tilde{Y}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1}) = \tilde{P}^t(\hat{1},\hat{0})$.
Comme on a aussi $Prob(\tilde{X}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1}) = \tilde{P}^t(z,\hat{0})$, on a finalement $\frac{Prob(\tilde{Y}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1})}{Prob(\tilde{X}_t = \hat{0} | \tilde{X}_0 = z, \tilde{Y}_0 = \hat{1})} = \frac{\tilde{P}^t(\hat{1},\hat{0})}{\tilde{P}^t(z,\hat{0})}$ comme souhaité.

Algorithme de Fill :

$k \leftarrow 0$
Faire
$X[0] \leftarrow 0$
Pour i allant de 1 à k
$U[i] \leftarrow Rand()$
$X[i] \leftarrow \varphi(X[i-1], U[i])$
Fin pour
$R \leftarrow X[k]$
$Y \leftarrow 1$
Pour i allant de k à 0
$Y \leftarrow \tilde{\varphi}(Y, X, U, i)$
Fin pour

$k \leftarrow 2 \cdot k$
Tant que $Y \neq 0$
Rendre R

3 Temps d'exécution

Nous commençons par faire les quelques hypothèses que notre chaîne de Markov est réversible de matrice de transition P sur un espace d'échantillon E de taille n, et a π pour distribution stationnaire. Il est aussi raisonnable de supposer que E est muni d'une structure d'ordre, avec un maximum $\hat{1}$ et un minimum $\hat{0}$.

3.1 Propp-Wilson

Dans cette partie, on appellera T_c le temps de coalescence de l'algorithme. Par la propriété stationnaire de la chaîne, ce temps T_c est le même en moyenne que le temps de coalescence du present vers le futur (le temps de coalescence dans l'algorithme est celui du passé vers le présent). Comme celui-ci est plus facile à manipuler, c'est celui que nous considèrerons. C'est un algorithme basé sur de l'aléatoire, et donc le temps d'exécution sera aussi une variable aléatoire. Il est donc raisonnable d'analyser le temps d'exécution de l'algorithme en considérant son espérance ($E[T]$ i.e. le temps d'exécution moyen) et sa répartition (i.e. $P[T \geq t]$). Dans le cadre de l'algorithme de Propp-Wilson, nous adoptons une méthode d'essai et réessai, qui double le temps de départ à chaque échec d'acceptation de l'état courant. Nous accepterons quand $T \geq T_c$, donc si nous commençons avec $T = 1$, alors le temps réel d'exécution sera :

$$\sum_{k=0}^{\lceil \log T_c \rceil} 2^k = 2^{\lceil \log T_c \rceil + 1} - 1$$

Dans la suite, puisque l'algorithme de Propp-Wilson nécessite une structure d'ordre sur E, nous pourrons considérer la plus grande des longueurs de chaînes de E, que l'on appellera d.

Lemme 4

$$\overline{d}(k) \leq P(T_c \geq k) \leq d\overline{d}(k)$$
$$P(T_c \geq k + n) \leq P(T_c \geq k) \cdot P(T_c \geq n)$$
$$E(T_c) \leq \frac{k}{1 - d\overline{d}(k)}$$

Preuve : Pour la première partie, il est bien connu que $|P - Q|_{var} = \min P(X_1 \neq X_2)$, où le minimum est pris sur l'ensemble des couple de variables aléatoires (X_1, X_2) qui ont comme distributions marginales respectivement P et Q. Considérons deux autres chaînes X'_1, X'_2 que l'on calcule en parrallèle avec les chaînes originales, en commençant dans n'importe quels états x et y. Par monotonie de la mise à jour, si les deux chaînes originales ont coalescé, alors les deux chaînes en parrallèles aussi, donc

$$P(X'^k_1 \neq X'^k_2) \leq P(X^k_1 \neq X^k_2) = P(T_c \geq k)$$

Mais $(X_1'^k, X_2'^k)$ ont pour distributions marginales $P^k(x,.)$ et $P^k(y,.)$ respectivement, donc

$$|P^k(x,.) - P^k(y,.)|_{var} \leq P(X_1'^k \neq X_2'^k)$$

ce qui entraîne :

$$|P^k(x,.) - P^k(y,.)|_{var} \leq P(T_c \geq k) \qquad \forall x, y \in E$$

donc :

$$\overline{d}(k) \leq P(T \geq k)$$

Pour la deuxième partie de la double inégalité, pour tout x dans E, on appelle $h(x)$ la plus longue chaîne qui a pour élément maximal x. Il est facile de voir que $h(\hat{0}) = 0$ et $h(\hat{1}) = d$, et pour $x < y$, on a $h(x) \leq h(y) + 1$, donc

$$\begin{aligned}
P(T_c \geq k) &= P(X_1^k \neq X_2^k) && (7)\\
&= E(\mathbb{1}_{X_1^k < X_2^k}) && (8)\\
&\leq E(h(X_1^k) - h(X_2^k)) && (9)\\
&= \sum_{y \in E} h(y) P^k(\hat{1}, y) - \sum_{y \in E} h(y) P^k(\hat{0}, y) && (10)\\
&= \sum_{y \in E} h(y)(P^k(\hat{1}, y) - P^k(\hat{0}, y)) && (11)\\
&\leq \sum_{y \in E : P^k(\hat{1},y) > P^k(\hat{0},y)} h(y)(P^k(\hat{1}, y) - P^k(\hat{0}, y)) && (12)\\
&\leq \sum_{y \in E : P^k(\hat{1},y) > P^k(\hat{0},y)} d(P^k(\hat{1}, y) - P^k(\hat{0}, y)) \qquad \text{car } h(y) \leq d && (13)\\
&= d|P^k(\hat{1},.) - P^k(\hat{0},.)|_{var} && (14)\\
&\leq d\overline{d}(k) && (15)
\end{aligned}$$

Pour la deuxième inégalité, on a

$$\begin{aligned}
P(T_c \geq k+n) &= P(F^0_{-k-n} \text{ n'est pas constant}) && (16)\\
&\leq P(F^{-n}_{-k-n} \text{ n'est pas constant et } F^0_{-n} \text{ n'est pas constant}) && (17)\\
&= P(F^{-n}_{-k-n} \text{ n'est pas constant}) \cdot P(F^0_{-n} \text{ n'est pas constant}) && (18)\\
&= P(T_c \geq k) \cdot P(T_c \geq n) && (19)
\end{aligned}$$

Pour la dernière inégalité, on a :

$$E(T_c) = \sum_{t=0}^{\infty} P(T_c \geq t) \tag{20}$$

$$\leq k \sum_{t=0}^{\infty} P(T_c \geq tk) \tag{21}$$

$$\leq k \sum_{t=0}^{\infty} P(T_c \geq k)^t \qquad \text{par la deuxième inégalité} \tag{22}$$

$$= \frac{k}{1 - P(T_c \geq k)} \tag{23}$$

$$\leq \frac{k}{1 - d\overline{d}(k)} \qquad \text{par la première inégalité} \tag{24}$$

□

Avec ce lemme à disposition, nous pouvons estimer $E(T_c)$. Appelons τ_1 le temps de mixage de la chaîne de Markov canonique (i.e. $\tau_1 = \min\{k : \overline{d}(k) \leq e^{-1}\}$). Posons maintenant $k = \tau_1(\lceil \ln 2d \rceil)$. Alors, par sous-multiplicativité de $\overline{d}(.)$, nous avons

$$\overline{d}(k) \leq \overline{d}(\tau_1)^{\lceil \ln d \rceil} \tag{25}$$

$$\leq \exp(-\ln 2d) \tag{26}$$

$$= \frac{1}{2d} \tag{27}$$

Donc

$$E(T_c) \leq \frac{k}{1 - d\overline{d}(k)} \leq \frac{k}{1 - d\frac{1}{2d}} = 2k$$

Donc le temps de coalescence sera en $O(\tau_1 \ln d)$. Ainsi le temps moyen d'exécution est aussi en $O(\tau_1 \ln d)$.

3.2 Algorithme de Fill

Dans cette partie, nous gardons l'approche précedente, qui consiste à essayer de borner l'espérance de la durée d'exécution de l'algorithme.

3.2.1 T variable

Considérons pour commencer quelques notations : Soit K le nombre (aléatoire) d'itérations de l'algorithme de Fill jusqu'à obtention du résultat. Alors le temps d'exécution sera :

$$T_r = \sum_{i=0}^{K-1} T 2^{i+1} \tag{28}$$

$$= 2T(2^K - 1) \tag{29}$$

Tout d'abord, nous voyons que l'évènement $\{K \geq i\}$ est équivalent à l'évènement que l'algorithme exécute la i-ème itération. Donc :

$$P(K > i | K > i - 1) = 1 - \frac{\tilde{\mathbf{P}}^{2^{i-1}}(\hat{0}, \hat{1})}{\pi(\hat{1})} \qquad \{i > 0\}$$

$$P(K > 0) = 1$$

qui est la probabilité de rejet de la i-ème itération. Soit τ_2 le seuil de séparation modifié de la chaîne de Markov canonique (i.e $\tau_2 = \min\{k : \overline{s}(k) \leq 2^{-1}\}$). Nous avons le lemme suivant :

Lemme 5

$$P(T_r > 2(2^i - 1)T) \leq \prod_{g=0}^{i-1} \overline{s}(2^g)$$

$$E(T_r) < 16\tau_2 T$$

Preuve
Il est facile de voir que $\{T_r > 2(2^i - 1)T\} = \{K > i\}$, donc :

$$P(T_r > 2(2^i - 1)T) = P(K > i) = \prod_{g=1}^{i} P(K > g | K > g - 1)$$

Mais comme nous l'avons dit avant :

$$P(K > g | K > g - 1) = 1 - \frac{\tilde{\mathbf{P}}^{2^{g-1}}(\hat{0}, \hat{1})}{\pi(\hat{1})}$$

Et ainsi, par définition :

$$P(K > g | K > g - 1) \leq \overline{s}(2^{g-1})$$

Et cela prouve la première inégalité. Pour la seconde, c'est un peu plus compliqué : Appelons $h = \lceil \lg \tau_2 \rceil$:

$$E(T_r) = \sum_{i=0}^{\infty} 2(2^i - 1)TP(T_r = 2(2^i - 1)T) \tag{30}$$

$$= \sum_{i=0}^{\infty} 2(2^{i+1} - 2^i)TP(T_r > 2(2^i - 1)T) \tag{31}$$

$$= 2T \sum_{i=0}^{\infty} 2^i P(T_r > 2(2^i - 1)T) \tag{32}$$

$$= 2T \left(\sum_{i=0}^{h} 2^i P(T_r > 2(2^i - 1)T) + \sum_{i=h+1}^{\infty} 2^i P(T_r > 2(2^i - 1)T) \right) \tag{33}$$

Nous bornons maintenant le premier terme :

$$\sum_{i=0}^{h} 2^i P(T_r > 2(2^i - 1)T) \leq \sum_{i=0}^{h} 2^i \leq 2^{h+1} < 4\tau_2$$

et le deuxième terme :

$$\sum_{i=h+1}^{\infty} 2^i P(T_r > 2(2^i-1)T) \leq \sum_{i=h+1}^{\infty} 2^i \prod_{g=0}^{i-1} \overline{s}(2^g) \tag{34}$$

$$\leq \sum_{i=h+1}^{\infty} 2^i \prod_{g=h}^{i-1} \overline{s}(2^g) \qquad \text{puisque } \overline{s}(2^g) \leq 1, \tag{35}$$

on 'maximise' $\overline{s}(2^g)$ pour tout $g < h$ (36)

$$\leq \sum_{i=h+1}^{\infty} 2^i \prod_{g=h}^{i-1} \overline{s}(2^h)^{2^{g-h}} \qquad \text{par sous-multiplicativité} \tag{37}$$

$$\leq \sum_{i=h+1}^{\infty} 2^i \prod_{g=h}^{i-1} 2^{-2^{g-h}} \qquad \text{par définition de } h \tag{38}$$

$$= \sum_{i=h+1}^{\infty} 2^i 2^{-\sum_{g=h}^{i-1} 2^{g-h}} \tag{39}$$

$$= \sum_{i=h+1}^{\infty} 2^i 2^{-2^{i-h}-1} \tag{40}$$

Par un simple changement de variable, on obtient :

$$\sum_{i=h+1}^{\infty} 2^i P(T_r > 2(2^i-1)T) \leq 2^h (\sum_{i=1}^{\infty} 2^{i-2^i+1}) < 4\tau_2$$

En combinant ces deux inégalités :

$$E(T_r) < 2T(4\tau_2 + 4\tau_2) = 16\tau_2 T$$

□

D'après ce lemme, nous avons que le temps moyen d'exécution de l'algorithme de Fill dans ce cadre est en $O(\tau_2)$.

3.2.2 T fixé

Il y a un autre cadre pour l'algorithme de Fill, dans lequel nous fixons un temps limite T et nous réessayons encore et encore jusqu'à accepter l'échantillon. Cette fois-ci, nous appellons K le nombre de réessais jusqu'à l'acceptation de l'échantillon. Dans ce cas, le temps d'exécution de l'algorithme est KT et le temps moyen d'exécution est $E[K]T$. Pour déterminer $E[K]$, rappelons-nous que l'algorithme de Fill est basé sur l'algorithme de rejet, où la deuxième phase est la phase de rejet. Ici, nous connaissons la "constante de couverture" qui est : $c = \pi(\hat{0})/\tilde{\mathcal{P}}^T \leq (1 - \overline{s}(T))^{-1}$.
Il est bien connu que dans un algorithme de rejet, le nombre de rejets attendu est la constante de couverture, donc le temps d'exécution de l'algorithme de Fill sera cT. Ainsi, si nous pouvions magiquement obtenir la valeur de τ_2, le temps d'exécution de l'algorithme de Fill dans ce cadre serait encore en $O(\tau_2)$, puisque

$c \leq (1 - \overline{s}(T))^{-1} = 2$. L'algorithme dans ce cadre n'est donc pas plus lent que dans le cadre précédent.
Et nous voyons apparaître un compromis : plus on laisse de temps au simulateur, plus on a de chance que l'échantillon obtenu soit bon, et donc moins on aura besoin de réessayer. Il faut donc ajuster le temps limite pour minimiser le temps d'exécution. La plupart du temps, le temps limite idéal est très difficile à connaître.

3.3 MCMC

Dans cette partie, nous allons examiner le temps d'exécution d'un algorithme MCMC. Il est essentiel de garder à l'esprit que MCMC est un algorithme approché, et que le temps d'exécution dépend donc de la précision du résultat souhaitée. Plus nous laissons tourner notre chaîne, plus nous approchons la distribution souhaitée. Le problème est que l'on ne sait pas à priori quand stopper, ou combien de temps suffit à obtenir un exemple convenable. Tout d'abord, supposons que nous ayons un chaîne dans un état x, et que nous voulions avoir un échantillon proche de π au sens de la variance :

$$\forall x \in E, \|\mu_x^k - \pi\|_{var} < c$$

ou autrement dit, nous voulons $d(k) < c$. Soit $\tau_3 = \{k : d(k) < e^{-1}\}$. Alors, par sous-multiplicativité, il est suffisant de prendre $k \geq -\tau_3 \lceil \ln c \rceil$. Ici, $-\lceil \ln c \rceil$ ne dépend pas de la chaîne de Markov canonique. Nous pouvons donc conclure que le temps d'exécution de l'algorithme est en $O(\tau_3)$. Notons que ce qui viens d'être dit n'a de sens que si l'on peut connaître (peut-être magiquement) la valeur de τ_3.
En réalité, il est impossible de connaître exactement cette valeur, ou même un majorant raisonnable de cette valeur. Beaucoup d'efforts ont été fournis pour trouver un moyen de connaître cette valeur, ou un critère pour savoir quand s'arrêter. Un critère est donné par l'idée de la coalescence. Il s'agit de faire tourner autant de chaînes que possible, avec la même fonction de mise à jour, et les mêmes variables aléatoires uniformes de mise à jour et d'accepter le résultats dès que toutes les chaînes ont coalescé (intuitivement, cela signifie que l'état initial n'a plus d'incidence sur l'état courant, et donc l'état courant sera proche de la distribution stationnaire). Dans le cas d'une fonction de mise à jour monotone, il est suffisant de ne calculer que deux chaînes, l'une partant de $\hat{0}$ et l'autre de $\hat{1}$. Dans ce cas, nous pouvons à nouveau utiliser le résultat du lemme 3.1 pour voir que le temps moyen de coalescence est en $O(\tau_1 \ln d)$.
Remarque : Bien que présenté ici, cette méthode est parfois incorrecte (voir exemple donné dans la présentation de l'algorithme Propp-Wilson).

3.4 Comparaison des algorithmes

Tout d'abord, résumons ce que nous avons fait. Le temps moyen d'exécution de : Propp-Wilson : $O(\tau_1 \ln d)$
Fill : $O(\tau_2)$
MCMC : $O(\tau_3)$
Coalescence : $O(\tau_1 \ln d)$
Ainsi, sous l'hypothèse que chaque étape de mise à jour coûte autant pour chaque

algorithme, pour comparer ces algorithmes en terme de temps d'exécution, nous devons comparer les trois quantités τ_1,τ_2,τ_3. On a :

Lemme 6 *Pour toute chaîne de Markov :* $\tau_3 \leq \tau_1 \leq 2\tau_3$ *et* $\tau_3 \leq \tau_2$.
Pour toute chaîne de Markov réversible : $\tau_3 \leq \tau_2 \leq 4\tau_1$.

preuve
Par définition, on a :

$$\overline{d}(\tau_1) \leq e^{-1}$$

de plus :

$$d(k) < \overline{d}(k) \qquad \forall k$$

donc :

$$d(\tau_1) < \overline{d}(\tau_1) \leq e^{-1}$$

Ainsi, par définition de τ_3, on a $\tau_3 < \tau_1$. Maintenant, comme on sait que $2d(K)$ est sous-multiplicative et que $2d(\tau_3) \leq 2/e$, on a :

$$\overline{d}(2\tau_3) \leq 2d(2\tau_3) \leq (2d(\tau_3)^2) \leq (\frac{2}{e})^2 < \frac{1}{2}$$

Ainsi, par définition, on a aussi $\tau_1 \leq 2\tau_3$. Par le lemme 1 dans le chapitre des chaînes de Markov : $d(k) < \overline{s}(k)$, donc

$$d(\tau_2) \leq \overline{s}(\tau_2) \leq \frac{1}{2}$$

et donc $\tau_3 \leq \tau_2$. De plus :

$$\begin{aligned}
\overline{s}(4\tau_1) &\leq 1 - (1 - \overline{d}(2\tau_1))^2 && (41)\\
&\leq 1 - (1 - \overline{d}(\tau_1)^2)^2 && (42)\\
&\leq 1 - (1 - \frac{1}{2}^2)^2 && (43)\\
&< \frac{1}{2} && (44)
\end{aligned}$$

Donc $\tau_2 \leq 4\tau_1$
□

Avec ce lemme, nous pouvons voir des relations étroites entre les coûts de ces algorithmes. Tout d'abord, dans le cas réversible (qui apparaît comme étant le cas le plus important), on peut réecrire les coûts de Fill et MCMC comme : $O(\tau_1)$. Ainsi, nous pouvons dire que les temps d'exécution des algorithmes de Fill et MCMC sont simplement les mêmes, et qu'il est plus mauvais pour l'algorithme de Propp-Wilson avec un facteur $\ln d$.
Pour le cas non réversible, il est bien connu qu'il n'existe pas de constante indépendante de la chaîne canonique qui majore le rapport $\frac{\tau_1}{\tau_2}$, et donc il n'y a pas grand chose à dire sur la comparaison entre Fill et Propp-Wilson. Mais la relation entre Propp-Wilson et MCMC reste inchangée, tandis que dans ce cas, Fill est plus mauvais que MCMC (puisque $\tau_2 \geq \tau_1$ et $\tau_1 = O(\tau_3)$).
Il est honnête de remarquer que cela est vrai seulement sous (*). Particulièrement pour le cas de l'algorithme de Fill où le problème est qu'échantillonner à partir du haut noyau est d'habitude plus complexe qu'échantillonner à partir de la matrice de transition. En fait, il y a des exemples pour lesquels à partir du

haut noyau transition est considérablement plus coûteux qu'un échantillonnage simple.
Une autre remarque est que bien que les simulateurs parfaits soient légèrement plus lent que MCMC, dans la plupart des cas, il est préferable, puisqu'il permet de produire la distribution exacte, tandis que MCMC ne peut que donner une distribution approchée. De plus, jusqu'à maintenant, il n'y a pas de méthode générale pour savoir quand les échantillons donnés par MCMC sont des approximations acceptables de la distribution souhaitée.

4 Le modèle d'Ising

4.1 Introduction

Le modèle d'Ising à d'abord été introduit pour modéliser les matériaux ferromagnétiques. Dans le modèle d'Ising, il y a n particules magnétiques qui intéragissent entre elles. ces particules forment un réseau (ou une grille), et chaque particule peut être dirigée vers le haut ou vers le bas, avec des voisinages qui préfèrent avoir le même état. Plus formellement :

Définition 9 *Un modèle d'Ising est un système qui contient n sites. Chaque site peut prendre la valeur 1 (haut) ou -1 (bas). Entre chaque paire i, j de sites, il y a une force d'interaction $\alpha_{i,j} \geq 0$ et en chaque site i, il y a une force d'intéraction exterieur β_i. L'énergie totale du système dans l'état $\mathbf{x} = (x_1, x_2, ..., x_n)$($x_i$ est la valeur du site i) est définie par :*

$$E(\mathbf{x}) = -\sum_{0<i<j\leq n} \alpha_{i,j} x_i x_j - \sum_{0<i\leq n} \beta_i x_i$$

et la probabilité de $\mathbf{x}$ est

$$P(\mathbf{x}) = \pi(\mathbf{x}) = c^{-1} \exp(-E(\mathbf{x}))$$

où c est une constante de normalisation.

Notons que bien que la distribution de probabilité de $\mathbf{x}$ ait une forme canonique, il est habituellement très difficile d'échantillonner directement à partir de cette distribution, à cause de la constante de normalisation c qui est inconnue. Souvent, les particules magnétiques sont organisées en réseau, et $\alpha_{i,j} > 0$ seulement pour les sites voisins dans le réseau. Nous considèrerons par la suite un exemple simple de modèle d'Ising dans lequels tous les sites sont dans une grille de taille $n \times n$, la force d'interaction de toutes les paires de sites voisins (les voisins de (i, j) sont $(i, j + 1), (i, j - 1), (i + 1, j), (i - 1, j)$, si une de ces coordonnées de sites est en dehors des bornes, i.e. $> n$ ou < 1, alors on ne prend simplement pas en compte ce site) est égale à α et la force exterieur est nulle pour tous les sites.

4.2 Simulateur de Gibbs et monotonie

4.2.1 Simulateur de Gibbs

On peut maintenant considérer un état $\mathbf{x}$ du système comme étant un vecteur ayant n^2 composantes, chaque composante prenant les valeurs 1 ou -1, et

étant indexée par les coordonnées du site. Ainsi, $\mathbf{x} = (x_{i,j})_{0<i,j\leq n}$. Pour chaque i, j soit $\pi(\mathbf{x}|.)_{i,j}$ la distribution conditionnelle de $x_{i,j}$ sachant toutes les autres composantes de $\mathbf{x}$. On a :

$$\pi(\mathbf{x}|1)_{i,j} = 1 - \pi(\mathbf{x}|-1)_{i,j}$$

$$\pi(\mathbf{x}|-1)_{i,j} = \frac{\pi(\mathbf{x}_{-1,i,j})}{\pi(\mathbf{x}_{-1,i,j}) + \pi(\mathbf{x}_{1,i,j})}$$

Où $\mathbf{x}_{-1,i,j}(\text{res.}\mathbf{x}_{1,i,j})$ sont les nouveaux états obtenus à partir de x en assignant sa composante $x_{i,j}$ à -1(res. 1). Notons que l'on peut réecrire $\pi(\mathbf{x}|-1)_{i,j}$ comme

$$\pi(\mathbf{x}|-1)_{i,j} = \frac{1}{1 + \frac{\pi(\mathbf{x}_{-1,i,j})}{\pi(\mathbf{x}_{1,i,j})}}$$

Remarquons que

$$\frac{\pi(\mathbf{x}_{-1,i,j})}{\pi(\mathbf{x}_{1,i,j})} = \frac{c^{-1}\exp[-E(\mathbf{x}_{-1,i,j})]}{c^{-1}\exp[-E(\mathbf{x}_{1,i,j})]} = \exp[E(\mathbf{x}_{1,i,j}) - E(\mathbf{x}_{-1,i,j})]$$

Ainsi, on peut éviter la problématique constante c et ainsi échantillonner à partir de $\pi(\mathbf{x}|.)_{i,j}$ deviens possible. Considérons l'échantillonneur de Gibbs appliqué à ce cas. En appliquant la description de l'échantillonneur de Gibbs, on a l'algorithme suivant :

Algorithme Heath bath

Etape 1 : Engendrer aléatoirement i,j avec une distribution uniforme sur $\{1, 2, ..., n\}$.
Etape 2 : Engendrer une variable aléatoire U uniforme sur $[0, 1]$.
Etape 3 : Si $U \leq \pi(\mathbf{x}\| - 1)_{i,j}$ alors assigner $x_{i,j} = -1$,$f((U, i, j), \mathbf{x}) = \mathbf{x}_{-1,i,j}$, sinon $x_{i,j} = 1$,$x_{i,j} = -1$,$f((U, i, j), \mathbf{x}) = \mathbf{x}_{1,i,j}$.

4.2.2 Monotonie

On définit un ordre sur notre espace d'états par : $\mathbf{x} > \mathbf{y}$ si $\mathbf{x} \neq \mathbf{y}$ et pour tout site (i, j) $x_{i,j} \geq y_{i,j}$.
Nous affirmons maintenant que l'échantillonneur de Gibbs a la propriété d'être croissante. Nous devons d'abord calculer explicitement la valeur de $\pi(\mathbf{x}\| - 1)$. Tout d'abord, nous avons, par la formule de la fonction énergie E :

$$E(\mathbf{x}_{1,i,j}) - E(\mathbf{x}_{-1,i,j}) = 2\alpha(x_{i+1,j} + x_{i-1,j} + x_{i,j+1} + x_{i,j-1})$$

ici, la valeur de n'importe quel $x_{i,j}$ est égale à 0 si ses coordonnées sortes de la grille. Ainsi, pour $\mathbf{x} > \mathbf{y}$ on a $E(\mathbf{x}_{1,i,j}) - E(\mathbf{x}_{-1,i,j}) \geq E(\mathbf{y}_{1,i,j}) - E(\mathbf{y}_{-1,i,j})$.
Donc $\frac{\pi(\mathbf{x}_{-1,i,j})}{\pi(\mathbf{x}_{1,i,j})} \geq \frac{\pi(\mathbf{y}_{-1,i,j})}{\pi(\mathbf{y}_{1,i,j})}$ puisque exp est une fonction croissante, donc $\pi(\mathbf{x}\| - 1)_{i,j} \leq \pi(\mathbf{y}\| - 1)_{i,j}$.
Maintenant, nous considérons la fonction de mise à jour de l'échantillonneur de Gibbs de la forme $f((U, i, j), \mathbf{x})$, définie dans la deuxième étape de l'algorithme heath bath :
Si $f((U, i, j), \mathbf{x}) = \mathbf{x}_{-1,i,j}$ alors $U \leq \pi(\mathbf{x}\| - 1)_{i,j} \leq \pi(\mathbf{y}\| - 1)_{i,j}$ donc nous avons aussi $f((U, i, j), \mathbf{y}) = \mathbf{y}_{-1,i,j} \leq \mathbf{x}_{-1,i,j}$.
Si $f((U, i, j), \mathbf{y}) = \mathbf{y}_{1,i,j}$ alors $U \geq \pi(\mathbf{y}\| - 1)_{i,j} \geq \pi(\mathbf{x}\| - 1)_{i,j}$ donc nous avons

aussi $f((U,i,j),\mathbf{x}) = \mathbf{x}_{1,i,j} \geq \mathbf{y}_{1,i,j}$.
Pour les autres cas (i.e $\pi(\mathbf{x}\| - 1)_{i,j} \leq U \leq \pi(\mathbf{y}\| - 1)_{i,j}$), on peut vérifier facilement que nous avons toujours $f((U,i,j),\mathbf{y}) \leq f((U,i,j),\mathbf{x})$.
Donc nous avons prouvé que cette fonction de mise à jour est croissante et a donc une chance que l'on puisse lui appliquer les échantillonneurs parfaits.

4.2.3 Approche de Propp-Wilson

Considérons maintenant l'application de l'algorithme de Propp-Wilson. L'implémentation est naturelle : on commence avec deux états initiaux $\hat{0}$ et $\hat{1}$, au temps $-T$ dans le passé. Nous calculons nos deux chaînes jusqu'au présent et vérifions si elles ont coalescé ou non. Si c'est le cas, nous acceptons l'état courant comme résultat, sinon, nous doublons T et recommençons.
Une chose importante est que nous devons utiliser les mêmes variables aléatoires uniformes à chaque fois. C'est-à-dire que si dans un premier temps, au temps $-t$ nous utilisons la variable aléatoire $(U,i,j)_t$, et que nous n'avons encore coalescé alors nous doublons T et recommençons, et cet fois, quand la chaîne arrive au temps $-t$, nous devons prendre la même valeur aléatoire $(U,i,j)_t$.

4.2.4 Approche de Fill

Il est facile de vérifier que la chaîne de Markov canonique est réversible (elle satisfait la condition de balance détaillée). Donc nous pouvons appliquer l'algorithme de Fill avec un peu d'avance (puisque nous pouvons maintenant réutiliser la fonction de mise à jour pour la chaîne renversée). Pour appliquer Fill, c'est un peu plus compliqué, puisque cela nécessite d'échantillonner à partir du "haut noyau".
Supposons pour la première étape au temps t que l'état de la chaîne renversée est $\mathbf{X})_t$. Nous notons simplement la chaîne $\tilde{\mathbf{X}}$ parce-que cela est pratique, mais nous conservons le temps en indice, donc $\tilde{\mathbf{X}}$ commencera au temps T et stoppera au temps 0. En fait $\tilde{\mathbf{X}}_t = \mathbf{X}_t$. Appelons $\mathbf{Y}$ la chaîne renversée qui commence en $\hat{1}$ au temps T. Cette chaîne est également indexée en sens inverse (donc de T à 0). Fill a proposé la méthode suivante pour effectivement échantillonner à partir du "haut noyau" dans le cas du modèle d'Ising :
Au temps t dans la phase inversée, considérons les deux cas :
Cas $\tilde{\mathbf{X}}_{t-1}$ diffère de $\tilde{\mathbf{X}}_t$ à la coordonnée (i,j)
si $\tilde{\mathbf{X}}_{t-1}(i,j) = 1$, on prend $\mathbf{Y}_{t-1}(i,j) = 1$ par croissance.
si $\tilde{\mathbf{X}}_{t-1}(i,j) = -1$ on fixe $\mathbf{Y}_{t-1}(i,j) = -1$ avec probabilité $\frac{\pi(\mathbf{Y}_t|-1)_{i,j}}{\pi(\tilde{\mathbf{X}}_t|-1)_{i,j}}$ et on fixe $\mathbf{Y}_{t-1}(i,j) = -1$ avec la probabilité restante.
Cas $\tilde{\mathbf{X}}_{t-1} = \tilde{\mathbf{X}}_t$:
Ici, comme dit dans la description de l'algorithme, nous devrions calculer la distribution conditionnelle de la déviation sous la condition $\tilde{\mathbf{X}}_{t-1} = \tilde{\mathbf{X}}_t$. Cependant, nous pouvons éviter ce calcul fastidieux en sauvegardant simplement les variables aléatoires uniformes utilisées dans la première étape et les réutiliser dans l'étape inversée.

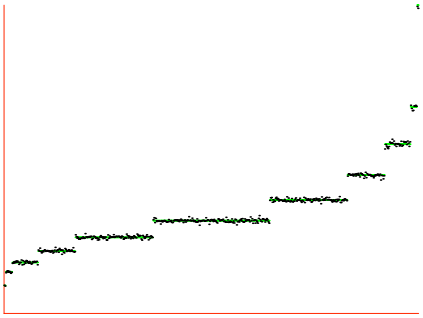
4.2.5 Temps d'exécution

Comme la plus longue chaîne dans ce cadre est n^2, le temps d'exécution est ici en $O(\tau_1 \log n)$ pour Propp-Wilson, en $O(\tau_2)$ pour Fill et en $O(\tau_3)$ pour MCMC normal.

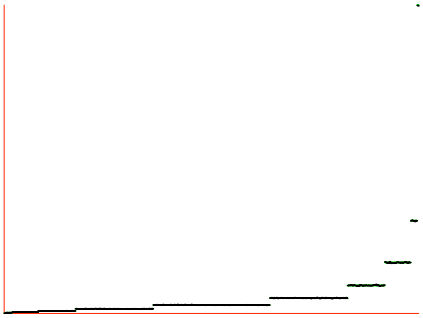
4.3 Expériences

4.3.1 Grille 3×3

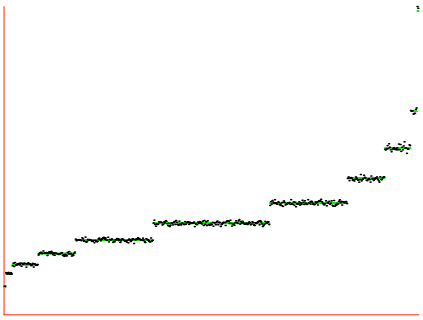
Nous avons réalisé quelques expériences avec une grille 3×3, de façon à ce que la taille soit suffisemment petite pour que l'on puisse effectivement calculer la distribution théorique. Et ici, nous avons obtenu quelques images qui expliquent l'experience. Le résultat est indexé par ordre croissant de probabilité théorique. L'experience à été faite avec 2000000 échantillons. On peut voir la distribution théorique en vert et celle donnée par chaque algorithme en noir.



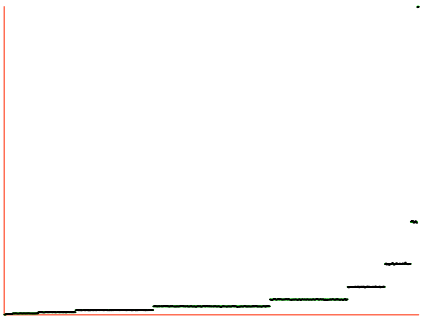
Fill $\alpha = 0.1$



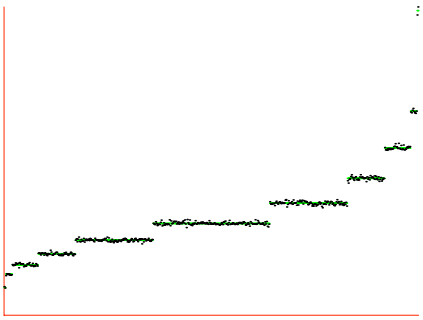
Fill $\alpha = 0.3$



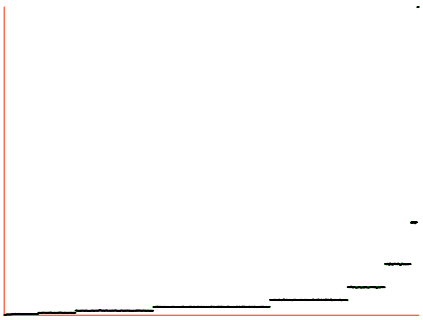
Propp-Wilson $\alpha = 0.1$



Propp-Wilson $\alpha = 0.3$



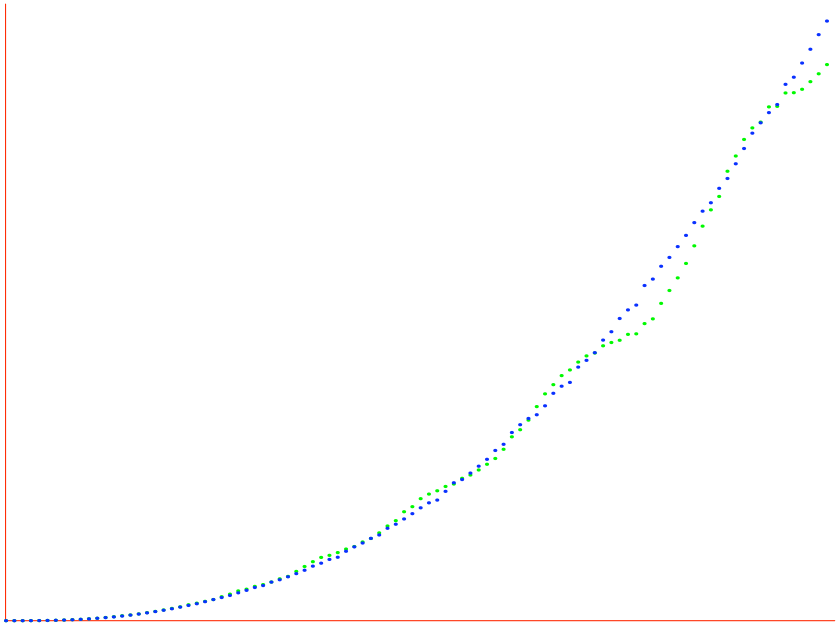
MCMC $\alpha = 0.1$



MCMC $\alpha = 0.3$

4.3.2 Temps d'exécution

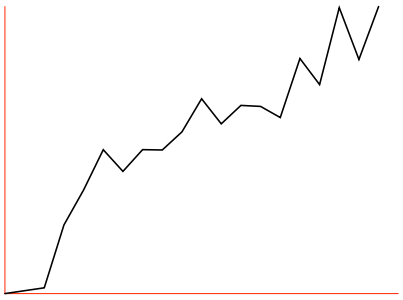
Pour comparer les temps d'exécution des algorithmes, nous avons fait plusieurs expériences. Pour Propp-Wilson et Fill, nous avons fait tourner les deux algorithmes beaucoup de fois (1000 fois), avec beaucoup de valeurs de la taille n de la grille (de 1 à 100), et nous calculions le temps moyen pour chaque algorithme. Le résultat apparaît sur cette image, où les courbes représente le temps moyen (en ordonnée) pour chaque valeur de n (en abscisse), en vert pour Propp-Wilson, et en bleu pour Fill.



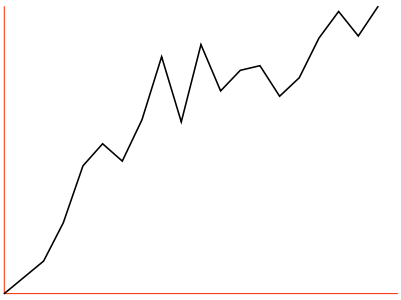
Propp-Wilson (vert) et Fill (bleu) pour $\alpha = 0.3$

D'après ce graphe, on peut voir que les temps d'exécution des algorithmes de Propp-Wilson et de Fill sont très proches, ce qui semble normal, puisque les résultats que l'on a établi ne sont que des majorations, et puis on a ici que des échantillons avec des tailles inférieures à 100. Pour $n > 100$, nous n'avons rien pour conclure.

Pour comparer MCMC avec les algorithmes "parfaits", ce n'est pas très clair, puisque la philosophie de MCMC est "plus c'est long, meilleur c'est". Nous avons donc exécuté l'algorithme MCMC pour chaque taille n, avec le temps moyen T_n donné par les simulateurs parfaits, et comparé les résultats. Ici, nous avons utilisé la distance au sens de la variance [def. 7] pour mesurer l'écart entre les résultats. Nous avons réalisé ces expériences avec $\alpha = 0.1$ et 100000 itérations par taille n, et nous montrons ici le graphe de l'écart en fonction de la taille (de 2 à 20).



Ecart entre MCMC et Propp-Wilson



Ecart entre MCMC et Fill

Références

1. http ://dbwilson.com/exact/
2. Explaining the perfect sampler, G. Casella, M. Lavine et C.P. Robert, The American Statistician, 55(4), 2001, 299 - 305
3. J.S. Liu, Monte Carlo Strategies in Scientific Computing, Springer Verlag, New York, 2001.
4. James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Structures and Algorithms, 9(1&2) :223–252, 1996.
5. James A. Fill. An interruptible algorithm for perfect sampling via Markov chains. The Annals of Applied Probability, 8(1) :131–162, 1998. (Postscript.) An extended abstract appeared in the Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing.
6. David J. Aldous and James A. Fill. Reversible Markov Chains and Random Walks on Graphs. Book in preparation, http ://www.stat.berkeley.edu/ aldous/book.html.
7. Morten Fismen. Exact simulation using Markov chains. Technical Report 6/98, Institutt for Matematiske Fag, 1998. Diploma-thesis. Advisor : Håvard Rue