École Normale Supérieure de Cachan
Département de Mathématiques

Mémoire de stage de fin de $3^{ème}$ année de Licence

Sous la direction de :
Lionel MOISAN
Université Paris Descartes
et Bruno GALERNE
Centre de Mathématiques et de Leurs Applications

SYNTHÈSE DE TEXTURES PAR LES CHAÎNES DE MARKOV

Massé Pierre-Yves
pierre-yves.masse@ens-cachan.fr
Meiniel William
william.meiniel@ens-cachan.fr
Roussillon Pierre
pierre.roussillon@ens-cachan.fr

Résumé

Dans cet article, nous synthétisons des textures à l'aide de chaînes de Markov. Nous partons de modèles très simples, mais obtenons des images d'une complexité surprenante à cet égard. Nombre d'entre elles sont implémentées grâce à l'algorithme de Metropolis-Hastings. Nous présentons également une version modifiée de cet algorithme, moins coûteuse en temps de calcul, mais qui produit des textures très similaires. Lors des simulations, nous sommes amenés à nous demander si les chaînes que nous observons ont convergé, sachant qu'il n'existe pas de critère numérique "d'entrée en convergence" réellement satisfaisant. Nous observons que certains algorithmes produisent des textures intéressantes avant d'avoir convergé, alors que les distributions stationnaires ne le sont pas.

Paris, Cachan, Juin 2011

Table des matières

I Théorie **7**

1 Chaînes de Markov **8**
1.1 Définitions 8
1.2 Propriétés 13
1.2.1 Définition de la distance et théorème de convergence 13
1.3 Démonstration du théorème de convergence 14
1.3.1 Existence et unicité de la distribution stationnaire 14
1.3.2 Convergence de la chaîne vers sa distribution stationnaire 15

2 Champ de Markov et champ de Gibbs **16**

II De la distribution de Gibbs aux algorithmes **19**

3 Algorithmes de l'échantilloneur de Gibbs et de Metropolis-Hastings **20**
3.1 L'échantilloneur de Gibbs 20
3.2 Metropolis-Hastings et ses variantes 21
3.2.1 Metropolis-Hastings classique 21
3.2.2 Metropolis-Hastings avec échange 23

4 Un exemple concret : le modèle d'Ising **24**
4.1 Le modèle d'Ising 24
4.1.1 Modèle physique 24
4.1.2 Modèle mathématique de l'algorithme de Metropolis-Hastings associé à Ising 25
4.2 Convergence de l'algorithme de Metropolis appliqué à Ising 25
4.3 Images obtenues 27
4.4 Étude de cas 29
4.4.1 Algorithme "Ising modifié" 29
4.4.2 Etude mathématique de cet algorithme 29
4.4.3 Caractérisation du modèle d'Ising classique 30
4.4.4 Ising modifié n'est pas un modèle d'Ising 31

III Problèmes de convergence **33**

IV Des algorithmes vers les distributions **36**

5 Algorithme "Écart quadratique au point central local" **38**
5.1 Description du modèle 38
5.2 Description en pseudo-code 38
5.3 Description mathématique 39
5.4 Propriétés mathématiques 39

6 Algorithme "Écart quadratique au point central global“ **41**
6.1 Description du modèle 41
6.2 Propriétés mathématiques 42

7 Algorithme "p-écart au point central local" **43**
7.1 Description du modèle 43

8 Algorithme "p-Écart au point central modifié" **44**
8.1 Description du modèle 44
8.2 Description en pseudo-code 44
8.3 Description mathématique 45
8.4 Observations 46

9 Algorithme du patch **48**
9.1 Observations 52

Appendices **57**

Table des figures

1 Metropolis-Hastings classique, 10000 itérations. 27
2 Metropolis-Hastings avec échange, 10000 itérations. 28
3 Metropolis-Hastings classique, $\lambda = 0.4$ 28
4 Comparaison des procédés classique et modifié, $\lambda = 0.7$, 1000 itérations. . . . 29
5 $z = \phi(p, \lambda)$ 32
6 Représentation de ϕ en niveaux de gris : zoom sur le coin inférieur droit . . . 33
7 Courbe d'énergie, 5000 itérations, $\lambda = 0.4$ 34
8 Somme des valeurs, 1000 itérations, $\lambda = 0.4$ 35
9 Comparaison des deux conventions, $\lambda = 0.7$ 36
10 Energies correspondantes, 10000 itérations 37
11 "Écart quadratique au point central local", 1000 itérations, rayon = 3 40
12 "Écart quadratique au point central local", rayon = 5 41
13 "Écart quadratique au point central local", $r = 10$ 42
14 "Écart quadratique au point central global", 50 itérations, rayon = 3 43
15 "Écart quadratique au point central global", 50 itérations, rayon = 5 43
16 "Écart quadratique au point central global", 50 itérations, rayon = 10 44
17 "Écart quadratique au point central global", 1000 itérations, rayon = 5, $\lambda = 10$ 44
18 "p-Écart au point central", p = 1, rayon = 3 45
19 "p-Écart au point central", p = 1, rayon = 5 46
20 "p-Écart au point central", p = 1, rayon = 10 47
21 "p-Écart au point central", $p = \frac{1}{2}$, rayon = 3 47
22 "p-Écart au point central", $p = \frac{1}{2}$, rayon = 5 48
23 "p-Écart au point central", $p = \frac{1}{2}$, rayon = 10 49
24 "p-Écart au point central modifié", p = 2, rayon = 5 50
25 "p-Écart au point central modifié", p = 2, rayon = 10 50
26 "p-Écart au point central modifié", p = 2, rayon = 15 51
27 "p-Écart au point central modifié", p = 2, 10000 itérations, $\lambda = 10$ 51
28 "p-Écart au point central modifié", $p = \frac{1}{2}$, 1000 itérations, $\lambda = 10$ 52
29 Images obtenues à partir de différents patchs, $n = 500$ 53
30 Une structure diagonale obtenue avec un patch différent de celui utilisé dans la Figure 29, image (c) 53
31 Différentes textures obtenues avec des patchs 5×5 54

Introduction

Nous avons dans nos travaux tenté d'obtenir des "textures intéressantes". Obtenir signifie bien sûr utiliser un programme tel que ses sorties soient les images voulues, mais les deux autres termes n'ont pas de définition formelle satisfaisante. Par texture, nous entendons donc comme le veut l'usage des images présentant une certaine périodicité due à la répétition de motifs qui en constitueraient les éléments de base.

"Intéressante" est plus délicat à définir. Le point de départ était d'utiliser des modèles très simples, permettant une description mathématique impossible pour des modèles plus élaborés (voir par exemple [2]). Plutôt que leur intérêt intrinsèque - nombre d'entre elles se sont cependant, de manière paradoxale, révélées assez riches - c'est cette possibilité de description qui fait la valeur des textures étudiées. En particulier, il n'était pas du tout dans nos objectifs de tenter de reproduire des textures existantes, par exemple inspirées du monde réel.

En fait, nous ne cherchons pas à obtenir des images isolées, mais plutôt des ensembles d'images, tels que chacun soit consitué d'images ayant de manière "évidente" un lien entre elles, sans qu'elle ne soient pourtant identiques. Nous cherchons donc à simuler des images aléatoires, et la propriété qui les unit est leur distribution : au sein d'un même ensemble, les images observées sont simulées selon une distribution identique. Ce sont donc en fin de compte plutôt ces ensembles que les images prises individuellement que nous appellerons textures.

Cette simulation est réalisée au moyen d'algorithmes. Mais aussi bien pour écrire ces derniers que pour discuter le caractère intéressant ou non des images obtenues, nous avons besoin d'un modèle mathématique tel que les images observées soient des réalisations du modèle. Le modèle que nous utilisons est celui des chaînes de Markov. Les images que nous obtenons sont donc des représentations de chaînes de Markov à valeurs dans le plan (ou plutôt dans des pavés *finis* du plan).

Bien sûr, nous voulons *a priori* (cette nuance sera justifée par la suite) étudier des images dont la chaîne de Markov sous-jacente a convergé. C'est là un premier problème, car il n'existe pas à ce jour de critère d'arrêt réellement satisfaisant pour déterminer si une chaîne de Markov a convergé lors d'une simulation numérique. Différents procédés permettent de discuter le sujet. Nous n'avons pas ici cherché à explorer de manière exhaustive ceux existant, mais avons utilisé quelques statistiques que nous présentons dans la Partie IV.

Pour autant, avant de savoir si les chaînes que nous étudions, et partant les algorithmes que nous programmons, ont convergé, il faut écrire ces derniers. Pour cela, nous avons procédé de deux manières différentes. Nous sommes d'abord partis de modèles de textures, qui décrivent les distributions de ces dernières. Ces modèles sont notamment inspirés de la physique. Le but est alors d'écrire les algorithmes qui simulent ces modèles, puis de prouver que ces algorithmes implémentent bien le modèle que nous voulions. Enfin, nous pouvons discuter des propriétés des textures obtenues. C'est l'objet de la partie III.

Nous avons ensuite, dans la Partie V, adopté la démarche inverse : partant d'algorithmes, nous avons essayé de décrire les distributions correspondant à l'état stationnaire. Cependant, une description analytique comme dans la partie précédente est bien souvent impossible. Une texture est donc définie comme la limite du procédé qui génère la suite d'images. Pour implémenter nos algorithmes, nous utilisons notamment un procédé dérivé de l'algorithme de Metropolis-Hastings que nous appelons Metropolis-Hastings local. Il permet, dans le cadre du modèle d'Ising, d'obtenir des textures très proches de celles obtenues par l'algorithme de

Metropolis “classique”, tout en étant beaucoup moins coûteux en temps de calcul. De même, il produit des résultats intéressants lorsque nous l'utilisons dans des algorithmes non décrits initialement en termes de champs de Gibbs.

Enfin, la Partie II est consacrée à la théorie dont nous avions besoin pour modéliser nos algorithmes. Nous présentons donc les chaînes de Markov, ainsi que les notions nécessaires à leur description, puis énonçons le théorème général de convergence d'une chaîne de Markov à valeurs dans un espace dénombrable. Cependant, attendu que les chaînes avec lesquelles nous travaillons sont à valeurs dans des espaces finis, nous ne démontrons le théorème mentionné que dans ce cas particulier, ce qui permet de simplifier les arguments.

Les algorithmes ont été écrits en C, et compilés grâce au logiciel Megawave2.

Première partie

Théorie

1 Chaînes de Markov

Les algorithmes utilisés pour notre travail de recherche sont à valeurs dans un espace dit espace de phases, représentant l'image. Chaque étape de l'algorithme ne dépend que de l'état de l'image à l'étape précédente. Les variables aléatoires représentant les différentes itérations de l'algorithme sont alors une chaîne de Markov. Nous allons introduire dans cette partie les diverses notions nécessaires pour la compréhension mathématique des algorithmes, puis les différentes propriétés qui nous permettrons de conclure quant à leur convergence.

1.1 Définitions

Remarque 1.1. Dans toute la suite, nous allons travailler dans un espace probabilisé $(\Omega_0, \sigma(U_n)_{n\geqslant 0}, P)$.

Définition 1.1 (**Chaînes de Markov Homogènes**). *Soit $(U_n)_{n\geq 0}$ une suite de variables aléatoires à valeurs dans un espace dénombrable E. Si pour tout entier $n \geq 0$ et tous états $i_0, i_1, \ldots, i_{n-1}, i, j$,*

$$P(U_{n+1} = j | U_n = i, U_{n-1} = i_{n-1}, \ldots, U_0 = i_0) = P(U_{n+1} = j | U_n = i)$$

alors la suite $\{U_n\}_{n\geq 0}$ est appelée une chaîne de Markov. *Si, de plus, le terme de droite est indépendant de n, on parle de* chaîne de Markov homogène *(HMC).*

Nous fixons ici les notations utilisées par la suite. Ainsi les états correspondront à des images, c'est-à-dire un tableau de la taille celle de l'image, et dont la valeur de chaque composante est comprise entre 0 et 255, représentant le niveau de gris du pixel considéré. La variable aléatoire U_n représente l'état de l'image après n itérations de l'algorithme, en partant de l'image initiale U_0.

Définition 1.2 (**Matrice de Transition**). *La matrice $P = \{p_{ij}\}_{i,j\in E}$, où*

$$p_{ij} = P(U_{n+1} = j | U_n = i),$$

est la matrice de transition *de la HMC.*

Définition 1.3 (**Distribution Stationnaire**). *Une loi de probabilité π satisfaisant*

$${}^t\pi = {}^t\pi P$$

est appelée une distribution stationnaire *de la matrice de transition P, ou de la HMC correspondante.*

Remarque 1.2. Cela signifie que lorsque l'on itère cette distribution avec la chaîne de Markov, elle reste constante.

Définition 1.4 (**Irréductibilité**). *Deux états communiquent s'il existe un chemin de l'un à l'autre et réciproquement. Cette relation définit une partition de l'espace des états. Lorsqu'il n'y a qu'une seule classe, on parle de chaîne irréductible.*

Remarque 1.3. Pour une HMC irréductible, il est possible d'aller d'un état i à un état j avec une probabilité non nulle en un nombre fini d'étapes. Mathématiquement, cela s'écrit :

$$\forall i, j \in E,\ \exists N \in \mathbb{N} \text{ tel que } P(U_{n+N} = j | U_n = i) > 0$$

Remarque 1.4. Il est important de noter que la notion d'irréductibilité dépend de l'espace dans lequel la chaîne de Markov prend ses valeurs.

Définition 1.5 (**Récurrence**). *Un état est dit* récurrent *si on y revient presque sûrement en temps fini sachant que la chaîne y était à l'origine. Un état est* récurrent positif *si l'espérance du temps de retour est finie. Lorsque tous les états de la chaîne sont récurrents (resp.* récurrents positifs*), on dit que la chaîne est* récurrente *(resp.* récurrente positive*)*

Remarque 1.5. *1.* Autrement dit, un état i est récurrent si, et seulement si

$$P_i(T_i < \infty) = 1$$

où P_i représente la mesure de probabilité sachant que la chaîne débute à l'état i, et T_i le temps de retour en i.

2. Un état i est récurrent positif si, et seulement si

$$\begin{cases} P_i(T_i < \infty) = 1 \\ E_i[T_i] < \infty \end{cases}$$

Remarque 1.6. Le caractère *récurrent* est une *propriété de classe*. Ainsi si deux états i et j communiquent ($\exists N \in \mathbb{N}$ tel que $P(U_{n+N} = j | U_n = i) > 0$), et si l'un est récurrent, alors l'autre l'est aussi. De même pour la récurrence positive.

Définition 1.6 (**Période**). *La* période d_i *d'un état $i \in E$ est*

$$d_i = pgcd\{n \geqslant 0\ ;\ p_{ii}(n) > 0\}$$

où $p_{ii}(n)$ représente la probabilité, partant de l'état i, de retourner à l'état i en n étapes. Si $d_i = 1$, l'état i est dit apériodique*.*

Théorème 1.1. *Si deux états i et j communiquent, alors ils ont la même période.*

Remarque 1.7. Ainsi, par définition d'une chaîne irréductible, tous les états possibles de la chaîne ont la même période. On peut alors parler de la *période d'une HMC irréductible*

Un exemple : la marche aléatoire sur $\mathbb{Z}$ On pose $E = \mathbb{Z}$ et on considère $(U_n)_{n\in\mathbb{N}}$ une chaîne de Markov telle que

$$P(U_{n+1} = i + 1 | U_n = i) = P(U_{n+1} = i - 1 | U_n = i) = \frac{1}{2}$$

symbolisant une marche aléatoire symétrique sur l'ensemble des entiers relatifs. $(U_n)_{n\in\mathbb{N}}$ est alors évidemment une HMC. Elle est de plus irréductible et récurrente, mais n'est pas récurrente positive.

1. La chaîne est irréductible :

   Démonstration. Soit $i, j \in \mathbb{Z}$, avec $i \leqslant j$, on a alors

   $$P(U_{n+j-i} = j | U_n = i) = \left(\frac{1}{2}\right)^{(j-i)} > 0$$

   Il existe donc un chemin de probabilité non nulle reliant chaque état possible de la chaîne. D'où l'irréductibilité de cette chaîne.

   □

2. La chaîne est récurrente :
   Cette démonstration est inspirée de la partie "marches aléatoires sur Z" du site maths.ens

   Démonstration. Pour prouver que la chaîne est récurrente, il faut montrer que le temps de retour (à l'état 0 par exemple), est presque sûrement fini. Cela se fait en plusieurs étapes :

   (a) Calcul de la probabilité que $S_n = 0$
   Il est tout d'abord utile de remarquer que la parité de S_n est celle de la parité de n. Ainsi S_n ne peut valoir 0 que pour les valeurs paires de n, et on a :

   $$P_0(S_{2n} = 0) = \frac{\binom{2n}{n}}{2^{2n}}$$

   En effet, pour la chaîne retourne en 0 sachant qu'elle y a débuté, il faut qu'il y ait autant de montées que de descentes. Or il y a $\binom{2n}{n}$ façons de choisir exactement n montées et donc exactement n descentes. Il est alors nécessaire de diviser ce nombre de possibilités par le nombre de chemins total possibles pour la chaîne soit 2^{2n}, car il y a deux choix possibles à chaque étape. D'où le résultat annoncé.

   (b) Calcul de la probabilité du temps de retour en n étape
   Il est maintenant question de calculer la probabilité d'atteindre l'état 0 à l'étape $2n$ et seulement à cette étape. Pour cela, nous allons différencier deux cas symétriques : si $S_1 = 1$ ou $S_1 = -1$.
   Supposons alors que $S_1 = 1$.
   Comme le temps de retour est de $2n$, il est alors nécessaire que $S_{2n-1} = 1$ (la chaîne n'atteint pas 0 avant l'étape $2n$). Il y a de plus $\binom{2n-2}{n-1}$ chemins reliant l'état 1 à l'étape 1 à l'état 1 à l'étape $2n-1$.
   Il s'agit maintenant, parmi ces chemins, de dénombrer ceux passant par 0. Or il est possible de montrer que le nombre de chemins reliant ces deux états, et passant au moins une fois par 0, est égal au nombre de chemins reliant l'état 1 à l'étape 1 à l'état -1 à l'étape $2n-1$. C'est le principe de réflexion. (Ce principe se démontre en effectuant une symétrisation des chemins à partir du moment où ils atteignent l'état 0.)
   On peut alors calculer ce nombre, et on obtient $\binom{2n-2}{n-2}$.
   D'où le nombre de chemins reliant 1 à l'étape 1 et 1 à l'étape $2n-1$: $\binom{2n-2}{n-1} - \binom{2n-2}{n-2}$, qui correspond aussi au nombre de chemins partant de 0 et dont le temps de retour en 0 est exactement $2n$.

Il faut alors diviser ce nombre par le nombre de chemins possibles, soit 2^{2n}, et en traitant le cas symétrique $S_1 = -1$, on obtient

$$P_0(T_0 = 2n) = 2\frac{\binom{2n-2}{n-1} - \binom{2n-2}{n-2}}{2^{2n}} = \frac{\binom{2n-2}{n-1} - \binom{2n-2}{n-2}}{2^{2n-1}}$$

Après calcul, on obtient le résultat final

$$P_0(T_0 = 2n) = \frac{(2n-2)!}{2^{2n-1}n!(n-1)!}$$

qui est aussi la valeur de $P_0(S_{2n-2} = 0) - P_0(S_{2n} = 0)$
Donc $P_0(T_0 = 2n) = P_0(S_{2n-2} = 0) - P_0(S_{2n} = 0)$.

(c) Calcul de $P_0(T_0 < \infty)$

$$\begin{aligned}
\sum_{n=1}^{m} P_0(T_0 = 2n) &= \sum_{n=1}^{m} P_0(S_{2n-2} = 0) - P_0(S_{2n} = 0) && (1)\\
&= P_0(S_0 = 0) - P_0(S_{2m} = 0) && (2)\\
&= 1 - \frac{\binom{2m}{m}}{2^{2m}} && (3)
\end{aligned}$$

Or, la formule de Stirling permet de montrer que $\frac{\binom{2m}{m}}{2^{2m}} \xrightarrow[m\to\infty]{} 0$ et donc, on obtient

$$P_0(T_0 < \infty) = 1$$

Donc la chaîne est bien récurrente.

□

3. La chaîne n'est pas récurrente positive :

Démonstration. Voir démonstration dans [1], Example 7.6, page 88. □

4. La chaîne est de période 2 :

Démonstration. Comme elle est irréductible, il suffit de montrer que l'état 0 est de période 2 (en vertu de la définition 2.6 et remarque 2.4). Or, en partant de l'état 0, il faut nécessairement un nombre pair d'étapes pour retourner en 0, donc 2 divise $d_0 = pgcd\{n \geqslant 0\ ;\ p_{00}(n) > 0\}$.
Et comme $p_{00}(2) = \frac{1}{4}$, on a par définition de d_0 que d_0 divise 2. Donc $d_0 = 2$.
Donc l'état 0 est de période 2.
Donc la chaîne est de période 2.

□

Par la suite, nous allons avoir besoin de faire converger nos algorithmes vers une distribution stationnaire. Encore faut il qu'elle existe. La propriété ci-dessous, bien que peu utilisée dans notre travail, donne des conditions d'existence de telles distributions dans un cadre général et met en lumière l'importance des notions d'*irréductibilité* et de *récurrence positive*.

Propriété 1.1 (**Condition d'existence et d'unicité de la distribution stationnaire**). *Une chaîne de Markov homogène irréductible est récurrente positive si, et seulement si, il existe une distribution stationnaire. De plus la distribution stationnaire π, lorsqu'elle existe, est unique, et $\pi > 0$*

Démonstration. Voir la démonstration dans [1], Theorem 3.1, page 104. □

Remarque 1.8. L'irréductibilité et la récurrence positive sont bien évidemment essentielles dans l'équivalence, comme le montrent les contre-exemples ci dessous.

Exemple de HMC irréductible, récurrente mais non récurrente positive, n'ayant pas de distribution stationnaire. Reprenons l'exemple de la marche aléatoire symétrique sur $\mathbb{Z}$, HMC irréductible, récurrente mais non récurrente positive. Montrons qu'elle n'admet pas de distribution stationnaire.

Démonstration. Une solution de la distribution stationnaire π vérifie l'équation suivante

$$\pi(i) = \frac{1}{2}\pi(i-1) + \frac{1}{2}\pi(i+1) \quad \forall i \geqslant 0$$

soit

$$\pi(i+1) - \pi(i) = \pi(i) - \pi(i-1) = \pi(1) - \pi(0)$$

On a donc

$$\pi(i) = \pi(0) + (\pi(1) - \pi(0))i$$

Mais comme $\pi(i) \in [0,1]$, on a nécessairement $\pi(1) - \pi(0) = 0$. Ainsi $\pi(i)$ est constant égal à $\pi(0)$, et comme la masse totale de π doit être finie (c'est une distribution de probabilité), on a $\pi(0) = 0 = \pi(i) \ \ \forall i \geqslant 0$, ce qui est une contradiction si on veut que π soit une distribution de probabilité.

□

Exemple de HMC récurrente positive mais non irréductible, ayant une infinité de distributions stationnaires.

Démonstration. On suppose dans cet exemple que E est un espace fini. On peut alors considérer la matrice de transition P d'une HMC. Posons $P = \mathrm{Id}$. P est bien une matrice de transition d'une HMC, et chaque état possible est récurrent positif, puisque si la chaîne démarre à un état donné, elle y reste presque sûrement. Mais on a alors que toute distribution de probabilité est une distribution stationnaire. □

1.2 Propriétés

Prenons pour la suite une chaîne de Markov irréductible et récurrente positive. Si la distribution initiale est la distribution stationnaire de la chaîne, alors elle gardera la même distribution au cours des itérations.
Quelques questions arrivent alors naturellement : Quel est le comportement de la chaîne, partant d'une distribution quelconque ? Peut on parler de convergence ? Comment définir une notion de convergence satisfaisante pour une distribution ?

1.2.1 Définition de la distance et théorème de convergence

Définition 1.7 (**Distance en variation**). *Soit E un espace dénombrable et α et β des distributions de probabilité sur E. La* distance en variation $d_v(\alpha, \beta)$ *entre α et β est définie par*

$$d_v(\alpha, \beta) = \frac{1}{2}|\alpha - \beta| = \frac{1}{2}\sum_{i \in E} |\alpha(i) - \beta(i)|$$

Ceci définit une notion de distance sur les distributions.

Remarque 1.9. Cette définition s'applique dans un cadre général. Ici nous ne travaillerons qu'avec des espaces E finis, et donc cette distance peut se simplifier. Ainsi la convergence en distance en variation est équivalente à une convergence coordonnée par coordonnée.

Remarque 1.10. La distance en variation entre deux variables aléatoires X et Y est la distance en variation entre les deux distributions associées.

Cette notion de distance permet d'étudier la convergence des chaînes de Markov grâce au théorème suivant :

Théorème 1.2. *Soit $(U_n)_{n \geqslant 0}$ une HMC irréductible, récurrente positive, sur un espace E. Il existe alors une unique distribution stationnaire π. De plus, pour toute distribution de probabilité μ, on note ${}^t\mu_n = {}^t\mu P^n$, où P est la matrice de transition de U. On a alors le résultat suivant :*

$$d_V(\mu_n, \pi) \xrightarrow[n \to \infty]{} 0$$

Démonstration. Voir démonstration dans [1], Theorem 2.1, page 130. □

Remarque 1.11. Ce théorème nous permet de dire que si l'on itère une chaîne de Markov irréductible, récurrente positive et apériodique à partir d'une distribution initiale quelconque, alors l'itérée des distributions va converger vers la distribution stationnaire de la chaîne. Ainsi, les itérées des algorithmes utilisés dans le stage représentant une chaîne de Markov, on peut obtenir des résultats de convergence vers une distribution stationnaire, ce qui valide le travail sur ces programmes.

Ce théorème s'applique à un espace E pouvant être dénombrable. Cependant, la récurrence positive, en hypothèse nécessaire à son application, peut être délicate à démontrer. Or, dans notre travail, l'espace E sera toujours fini et il est alors possible de simplifier ce théorème de la façon suivante :

Théorème 1.3 (**convergence vers l'état stationnaire**). *Supposons $(U_n)_{n\geq 0}$ irréductible, à valeurs dans E fini. Alors $(U_n)_{n\geq 0}$ admet une unique distribution stationnaire, et la distribution de U_n converge vers celle-ci quand $n \to \infty$.*

Remarque 1.12. Par la suite, l'espace E sera l'ensemble des images possibles pour l'algorithme, qui est un espace fini. Ainsi il sera simplement nécessaire de vérifier que la chaîne de Markov associée à l'algorithme soit irréductible et apériodique pour se placer dans les hypothèses du théorème 1.3

Cette simplification rejoint la proposition suivante dans le cadre fini :

Propriété 1.2. *Soit E un espace* fini. *Alors une HMC irréductible est récurrente positive.*

Cependant, dans la démonstration proposée ci-après pour le théorème 1.3, on n'utilise que l'irréductibilité de la chaîne, et non la récurrence positive. Le cadre fini nous permet alors de ne pas se soucier d'hypothèses utiles dans le cadre général.

Nous allons maintenant démontrer le théorème 1.3

1.3 Démonstration du théorème de convergence

1.3.1 Existence et unicité de la distribution stationnaire

Remarque 1.13. Les arguments utilisés dans cette sous-section sont librement inspirés de ceux utilisés dans la démonstration du théorème de Perron-Froebenius de la page en.wikipedia correspondante.

Nous voulons prouver la proposition suivante :

Propriété 1.3. *$(U_n)_{n\geqslant 0}$ admet une unique distribution stationnaire.*

Nous aurons besoin des deux lemmes suivants. Dans toute la suite, P est la matrice de transition de la chaîne.

Lemme 1.1. *Soit F l'espace propre de P associé à 1. Soit X non nul tel que $X \in F$. Supposons que toutes les coordonnées de X sont positives. Alors elles sont toutes strictement positives.*

Démonstration du lemme 1.1. $X \neq 0$ donc par exemple $x_0 \neq 0$.
Soit x_i une coordonnée quelconque de X. P est irréductible donc on peut trouver $n \geq 0$ tel que $P^n(i,0) > 0$.
Mais $X = P^n X$ donc

$$x_i = \sum_{k\in E} P^n(i,k)x_k \geq P^n(i,0)x_0 > 0.$$

□

Remarque 1.14. On voit dans la preuve que 1 n'a pas de rôle particulier et peut être remplacé par n'importe quel réel strictement positif.

Lemme 1.2. *En gardant les notations précédentes, on suppose que F contient un vecteur e dont toutes les coordonnées sont positives. Alors F est de dimension* 1.

Démonstration du lemme 1.2. Soit donc Y non nul vecteur de F. Quitte à changer Y en $-Y$, on peut supposer que Y a une coordonnée strictement positive. Montrons alors que Y est colinéaire à e.
$\{\alpha \in \mathbb{R}_+ | e - \alpha Y$ a toutes ses coordonnées positives$\}$ est un fermé borné non vide (il contient 0) donc admet un maximum α_0.
Une des coordonnées au moins de $X = e - \alpha_0 Y$ est nulle par maximalité de α_0. En fait, elles le sont toutes.
En effet, sinon on aurait $X \neq 0$ et $X \in F$ (car $Y \in F$) donc le lemme 1.1 s'appliquerait et contredirait l'assertion précédente. Donc Y est bien colinéaire à e et F est bien de dimension 1. □

Nous pouvons maintenant démontrer la proposition 1.3.

Démonstration de la proposition 1.3. Une distribution stationnaire est un vecteur X de norme 1 vérifiant ${}^tXP = {}^t X$ soit ${}^tPX = X$. Il est donc équivalent de prouver que 1 est valeur propre de ${}^tP$ et que l'espace propre associé est de dimension 1. Nous allons en fait le prouver pour P, pour laquelle nous avons plus d'informations que pour sa transposée. C'est légitime car une matrice et sa transposée ont même valeurs propres et les espaces propres associés sont de même dimension.
Or P est stochastique, donc le vecteur e dont toutes les coordonnées sont égales à 1 est vecteur propre pour la valeur propre 1.
Donc d'après le lemme 1.2 il est bien de dimension 1. □

Remarque 1.15. On a donc montré que si une HMC à valeurs dans un espace fini était irréductible, alors elle admettait une distribution stationnaire. En combinant ce résultat à la proposition 1.1 on a alors démontré par la même occasion que cette chaîne était récurrente positive et donc prouvé la proposition 1.2.

Nous pouvons maintenant prouver la convergence de la chaîne vers sa distribution stationnaire, que nous notons π.

1.3.2 Convergence de la chaîne vers sa distribution stationnaire

Nous voulons prouver la proposition suivante

Propriété 1.4. *Quelque soit U_0, la chaîne $(U_n)_{n\geqslant 0}$ converge vers la distribution stationnaire π.*

Démonstration. Notons $d = |E|$. La suite $(U_n)_{n\geqslant 0}$ vérifie la relation de récurrence

$$U_{n+1} = PU_n$$

où P est la matrice de transition de la chaîne.
La suite $(U_n)_{n\geqslant 0}$ est de plus à valeurs dans la sphère unité de $\mathbb{R}^d$. (U_0 est une probabilité, et P étant stochastique, par récurrence tous les U_n sont des probabilités.) Or la sphère est compacte (c'est un fermé borné et on est en dimension finie). Pour prouver la convergence de la suite, il est donc équivalent de montrer qu'elle admet une unique valeur d'adhérence.
Or, par le théorème de Bolzano-Weierstrass, elle en admet une, que l'on note λ. Mais λ est alors un point fixe de $X \mapsto PX$ (continuité d'un endomorphisme en dimension finie), donc un vecteur propre de P pour la valeur propre 1. λ est donc colinéaire à π. Mais les deux

sont à coefficients positifs et de norme 1, donc $\lambda = \pi$. Donc $(U_n)_{n\in\mathbb{N}}$ converge coordonnée par coordonnée vers π, or grâce à la remarque 1.9, cela est équivalent à une convergence en distance en variation.

□

Nous avons donc prouvé le Théorème 1.3.

2 Champ de Markov et champ de Gibbs

La notion de chaîne de Markov est un outil utile pour représenter une évolution temporelle, itérative. Nous allons ainsi les utiliser pour modéliser l'évolution des algorithmes. Cependant, l'espace des phases sur lequel nous travaillons est à formaliser, mathématiquement et algorithmiquement.

Définition 2.1 (**Champ aléatoire**). *Soit Ω un espace fini, dont les éléments (appelés sites) seront notés s, et soit Λ un espace fini appelé l'espace de phase. Un champ aléatoire sur Ω, de phases à valeurs dans Λ est une famille $U = \{U(s)\}_{s\in\Omega}$ de variables aléatoires $U(s)$ à valeurs dans Λ.*

Remarque 2.1. Concrètement, l'espace Ω sera l'ensemble des pixels de l'image, et un site s représentera donc un pixel. L'espace Λ représentera les niveaux de gris d'un pixel considéré, et donc l'espace Λ^Ω l'ensemble des images possibles, qui est bien un espace fini puisque chaque pixel a un nombre fini de valeurs possibles.

Pour pouvoir synthétiser une texture, les algorithmes que nous étudierons font dépendre la valeur d'un pixel de son voisinage, qu'il faut donc définir formellement.

Définition 2.2 (**Voisinage**). *Un système de voisinage sur Ω est une famille $N = \{\mathcal{N}_s\}_{s\in\Omega}$ de sous-ensembles de Ω telle que pour tout $s \in \Omega$ on ait les deux propriétés suivantes :*

1. $s \notin \mathcal{N}_s$
2. $t \in \mathcal{N}_s \Rightarrow s \in \mathcal{N}_t$

Un sous-ensemble $\mathcal{N}_s$ est appelé voisinage du site s. Le couple (Ω, N) est appelé un graphe ou une topologie.

Cette définition correspond avec l'intuition que l'on peut avoir des voisinages : deux sites sont liés si, et seulement si, ils sont au voisinage l'un de l'autre.

Définition 2.3 (**Champ aléatoire de Markov**). *Un champ aléatoire est appelé un champ de Markov aléatoire (RMF) associé au système de voisinages N si pour tout site $s \in \Omega$ les variables aléatoires $U(s)$ et $U(\Omega\backslash\tilde{\mathcal{N}_s})$ sont indépendants sachant $U(\mathcal{N}_s)$.*

En reformulant en langage mathématique, on obtient alors :

$$P\Big(U(s) = u(s)|U(\Omega\backslash s) = u(\Omega\backslash s)\Big) = P\Big(U(s) = u(s)|U(\mathcal{N}_s) = u(\mathcal{N}_s)\Big)$$

En poursuivant le parallèle fait avec les images, pour un champ de Markov, la valeur d'une image en un pixel dépend du voisinage de ce pixel, mais pas de l'image entière. Un champ de Markov aléatoire dépend donc du système de voisinage que l'on a précédemment choisi. De manière évidente, la taille du voisinage doit rester bien inférieure à celle de l'image.

Les programmes utilisés sont à valeurs dans un espace de réalisation de champ de Markov. En effet, la valeur d'un pixel dépendra en probabilité de la valeur des pixels de son voisinage. Ainsi nous obtenons une chaîne de Markov indexée par le nombre d'itérations de l'algorithme et dont chaque variable aléatoire est un champ de Markov. La réalisation de ce champ de Markov donne alors une image.

Remarque 2.2. Une distribution de probabilité sur la chaîne de Markov représentant l'algorithme est une mesure de probabilité définie sur l'ensemble des images possibles pour l'algorithme.

La notion de champ de Gibbs, présentée par la suite, fait appel à une intuition physique. La distribution de probabilité sur les états possibles est en effet dépendante d'une énergie que l'on aura introduite et définie. Cela permet, notamment pour les images, de partir de modèles physiques concrets pour obtenir une distribution connue et voulue. Un exemple clé de ce rapport sera le modèle d'Ising, inspiré de l'étude de l'orientation des pierres ferromagnétiques. Mais avant d'expliciter ce modèle de façon plus poussée, il nous faut quelques outils mathématiques :

Définition 2.4 (**Cliques**). *Tout singleton $\{s\}$ est une clique. Un ensemble $C \subset \Omega$ contenant plus d'un élément est appelé une clique du graphe (Ω, N) si, et seulement si, deux éléments distincts de C sont voisins.*

Définition 2.5 (**Potentiel de Gibbs**). *Un potentiel de Gibbs sur Λ^Ω (l'ensemble des images possibles) associé au système N est une collection $\{V_C\}_{C\subset\Omega}$ de fonctions V_C telles que*

1. *$V_C = 0$ si C n'est pas une clique,*
2. *pour tout $u, u' \in \Lambda^\Omega$ et pour tout $C \subset \Omega$,*

$$(u(C) = u'(C)) \Rightarrow (V_C(u) = V_C(u'))$$

La fonction d'énergie $\mathcal{E}$ dérive du potentiel $\{V_C\}_{C\subset S}$ si

$$\mathcal{E}(u) = \sum V_C(u)$$

Et alors,

$$\pi_T(u) = \frac{1}{Z_T} e^{-\frac{1}{T}\mathcal{E}(u)}$$

est appelée distribution de Gibbs sur Λ^Ω associée au potentiel $\{V_C\}_{C\subset\Omega}$.

Remarque 2.3.

1. u et u' représentent donc une image donnée et $\pi_T(u)$ la probabilité d'avoir l'image u
2. Le constante Z_T est une constante de normalisation, mais est en pratique difficile à expliciter. Ceci peut rendre difficile à exploiter un champ de Gibbs. Cependant nous verrons un algorithme permettant de s'affranchir du calcul de cette constante en faisant le quotient de deux distributions. (Algorithme de Metropolis-Hastings)

Théorème 2.1 (**Hammersley & Clifford (1968)**). *Les descriptions en termes de champ de Gibbs ou de champ de Markov sont équivalentes.*

1. *Tout champ de Gibbs est un champ de Markov. En conservant les notations précédentes, la spécification locale, définie par $\pi_T(u(s)|u(\mathcal{N}s))$, est donnée par la formule :*

$$\pi_T(u(s)|u(\mathcal{N}s)) = \frac{e^{-\sum_{C\ni s} V_C(u)}}{\sum_{\lambda\in\Lambda} e^{-\sum_{C\ni s} V_C(\lambda, u(S\setminus s))}}$$

2. *Tout champ de Markov est un champ de Gibbs.*

Démonstration. Voir démonstration dans [1], Theorem 2.1, page 260. □

L'équivalence ainsi obtenue entre les deux principes. En effet, l'idée de créer un algorithme à partir d'une distribution de Gibbs est, comme nous allons le voir dans la partie suivante, d'une grande simplicité. La force de ce théorème est de dire que la réciproque est vraie. Ainsi, si l'on décrit une texture à partir d'une chaîne de Markov, il est possible de l'exprimer en tant que champ de Gibbs. Nous allons dans un premier lieu étudier l'aspect direct du théorème, avant de nous concentrer sur la réciproque, qui est le coeur de notre stage.

Deuxième partie

De la distribution de Gibbs aux algorithmes

Dans cette partie, nous cherchons à synthétiser des textures de distribution donnée. Nous présentons d'abord des algorithmes en ce sens, puis nous intéressons à leur application à un modèle concret : le modèle d'Ising, qui est l'un des exemples les plus connus de champs de Gibbs.

3 Algorithmes de l'échantilloneur de Gibbs et de Metropolis-Hastings

Les algorithmes de Metropolis-Hastings et de l'échantilloneur de Gibbs sont des algorithmes utilisant une technique de MCMC (Monte Carlo Markov Chain), c'est-à-dire la convergence d'une chaîne de Markov vers une distribution stationnaire voulue, par itération de cette chaîne. Cette méthode s'appuie donc sur un résultat de convergence en distribution des chaînes de Markov, reposant sur certaines hypothèses étudiées auparavant (irréductibilité, apériodicité, récurrence positive,...).

3.1 L'échantilloneur de Gibbs

L'objectif est maintenant de générer une chaîne de Markov qui converge vers la distribution souhaitée pour la synthèse de texture.

Le principe de l'échantillonage de Gibbs est le suivant. On considère U_n l'image (la texture) obtenue à la n-ème itération de l'algorithme, et on regarde la transition de $U_n = x$ à $U_{n+1} = y$. L'état y est obtenu à partir de l'étape x en changeant ou non la valeur d'un site uniquement. Ce site est choisi avec une probabilité q_s parmi les autres sites. La configuration x est changée en configuration y comme suit :
$y(S\backslash s) = x(S\backslash s)$ et la nouvelle valeur de $y(s)$ est choisie avec une probabilité $\pi(y(s)|x(S\backslash s))$

Pour une description plus formelle, on introduit la quantité :

$$Q(u,x) = \frac{e^{-\sum_{C\ni x} V_C(u)}}{\sum_{\omega\in\Lambda} e^{-\sum_{C\ni x} V_C(\omega,u(\mathcal{N}_x))}} = \pi(u(x)|u(y), y \neq x)$$

appelée spécification locale de l'image u au point x.
On note aussi $Q_\lambda(u,x)$ la quantité $Q(u,x)$ évaluée en remplaçant la valeur de $u(x)$ par λ.

Description en pseudo-code

L'algorithme s'écrit en pseudo-code, pour l'initialisation par un bruit blanc :

pour tout pixel $x \in \Omega$:
    tirer aléatoirement $u(x)$ uniformément dans Λ
itérer :
    tirer aléatoirement x dans Ω et λ dans Λ
    calculer $p_\lambda = Q_\lambda(u,x)$
    tirer q uniformément sur $[0,1]$
    SI $q < p_\lambda$ faire $u(x) \leftarrow \lambda$

Description mathématique

On construit une suite de variables aléatoires $(U_n)_{n\geq 0}$ à valeur dans Ω de la manière suivante :

– U_0 est un bruit blanc.

– pour tout $n \geq 1$,

$$D_n = (\lambda_n - 1)U_{n-1}(X_n)\mathbf{1}_{X_n}, \tag{4}$$

$$U_n = U_{n-1} + \mathbf{1}_{\epsilon_n < t_n} D_n, \tag{5}$$

où $(X_n)_{n\geqslant 0}$ est une suite de variables aléatoires uniformes sur Ω, $(\lambda_n)_{n\geqslant 0}$ est une suite de variables aléatoires, indépendantes, identiquement distribuées uniformes sur Λ, $(\epsilon_n)_{n\geqslant 0}$ est une suite de v.a. i.i.d. uniformes sur $[0, 1]$, et

$$t_n = Q_{\lambda_n}(U_n, X_n) \tag{6}$$

Les variables U_0, $(X_n)_{n\geq 1}$ et $(\epsilon_n)_{n\geq 1}$ sont indépendantes.
$\mathbf{1}_{X_n}$ est l'image comportant des 0 en chaque site, sauf au site X_n, qui prend la valeur 1.

3.2 Metropolis-Hastings et ses variantes

3.2.1 Metropolis-Hastings classique

Cet algorithme repose sur un principe simple : en partant d'une image initiale (bruit blanc, par exemple), l'algorithme propose un changement d'état de cette image, c'est-à-dire une modification de l'image. Ce changement est accepté avec une probabilité fonction de la nouvelle et de l'ancienne image. Par itérations successives, on peut obtenir convergence vers une distribution stationnaire, comme pour l'algorithme de échantilloneur de Gibbs. De façon plus formelle, Metropolis-Hastings peut se décrire de la manière suivante :

Soit π la distribution que l'on souhaite obtenir. Pour cela, il faut exhiber, comme pour l'échantilloneur de Gibbs, une chaîne de Markov dont les itérations convergeront vers cette distribution.

On pose Q la matrice appelée *matrice de proposition*, dont les coefficients q_{ij} sont indexés par les différents états possibles de l'image. Ainsi l'état i correspond à une image. q_{ij} représente la probabilité pour l'algorithme, sachant que l'on est à l'état i, de proposer l'état j comme nouvel état. On définit alors la chaîne de Markov de la façon suivante :

$$U_0 = \mu \text{ où } \mu \text{ est une distribution initiale quelconque}$$

$$P(U_{n+1} = j | U_n = i) = \underbrace{q_{ij}}_{\substack{\text{probabilité de proposer le changement} \\ \text{de l'état } i \text{ à } j}} \overbrace{\min\left(\frac{\pi(j)q_{ij}}{\pi(i)q_{ji}}, 1\right)}^{\substack{\text{probabilité d'acceptation de ce changement}}} .$$

Le plus souvent, Q sera symétrique, et le quotient se simplifie alors

$$P(U_{n+1} = j | U_n = i) = q_{ij} \min\left(\frac{\pi(j)}{\pi(i)}, 1\right).$$

Ceci définit bien une chaîne de Markov, convergeant en probabilité vers la distribution stationnaire π. (admis) Si la distribution est une distribution de Gibbs, alors la distribution est de la forme

$$\pi(i) = \frac{1}{Z} e^{-\mathcal{E}(i)}.$$

Le quotient des deux distributions apparaissant dans la probabilité conditionnelle devient alors

$$\frac{\pi(j)}{\pi(i)} = e^{\mathcal{E}(i)-\mathcal{E}(j)}$$

qui est supérieur à 1 si, et seulement si $\mathcal{E}(i) \geqslant \mathcal{E}(j)$, et donc si, et seulement si l'énergie diminue après le changement d'état. L'algorithme de Metropolis-Hastings autorise donc avec une probabilité 1 les changements d'état diminuant l'énergie. A noter tout de même qu'il autorise avec une probabilité non nulle un changement d'état même si ce dernier ne diminue pas l'énergie. Il n'est donc pas déterministe et permet un certain aléa d'où l'irréductibilité nécessaire à la convergence (voir théorème 1.3). Cependant la tendance globale de cet algorithme est de diminuer l'énergie de l'image.

Description mathématique

Concrètement, la matrice Q de proposition sera choisie de telle sorte qu'elle simule un choix suivant une loi uniforme sur l'ensemble des pixels de l'image. Et même, $q_{ij} = 0$ si i et j diffèrent de plus d'un pixel. Le changement proposé ne change donc l'image que d'un pixel, choisit au hasard uniformément sur l'ensemble des pixels. On note $u(x)$ la valeur du pixel au point x.

On reprend les même notations que pour l'algorithme du l'échantilloneur Gibbs :
- $(U_n)_{n\in\mathbb{N}}$ suite de variables aléatoires, indépendantes, identiquement distribuées, à valeurs dans Λ^Ω.
- $(X_n)_{n\in\mathbb{N}}$suite de v.a. i.i.d. à valeurs dans Ω.
- $(Z_n)_{n\in\mathbb{N}}$ suite de v.a. i.i.d., à valeurs dans $\{0,\dots,255\}$.
- $(U_n)_{n\in\mathbb{N}}$ suite de v.a., représentant l'image à la n-ième iération.
- $(\varepsilon_n)_{n\in\mathbb{N}}$ suite de v.a. i.i.d., suivant une loi uniforme sur $[0,1]$.
- U_0 est une image initiale quelconque (bruit blanc uniforme sur les niveaux de gris,...)

$$\forall x \in \Omega : \quad V_n(x) = \begin{cases} Z_n & \text{si } x = X_n \\ U_n(x) & \text{sinon} \end{cases} \tag{7}$$

$$p_n = \min\left(\frac{\pi(V_n)}{\pi(U_n)}, 1\right) \tag{8}$$

$$U_{n+1} = \mathbf{1}_{\varepsilon_n < p_n} V_n + \mathbf{1}_{\varepsilon_n > p_n} U_n \tag{9}$$

Pseudo-code de l'algorithme

pour tout pixel $x \in \Omega$:
    tirer aléatoirement $u(x)$ uniformémentdans l'espace Λ des valeurs possibles pour un pixel

itérer :
    $i =$ état initial de l'image
    tirer aléatoirement x uniformément dans Ω
    tirer aléatoirement $u'(x)$ dans Λ, l'état proposé est noté j
    $p \leftarrow \min\left(\frac{\pi(j)}{\pi(i)}, 1\right)$
    avec probabilité p, faire $u(x) \leftarrow u'(x)$ et $i = j$

3.2.2 Metropolis-Hastings avec échange

Définition 3.1 (**Histogramme**). *C'est la donnée, pour chaque $\lambda \in \Lambda$ du nombre de sites ayant cette valeur λ.*

Cet algorithme est très proche de celui de Metropolis-Hastings où l'on ne modifie l'image que d'un pixel à chaque itération, à ceci près que l'on décide de conserver l'histogramme de l'image initiale. Pour cela, nous allons, à chaque étape de l'algorithme, échanger deux sites plutôt qu'en écraser un.

Description mathématique

On construit une suite de variables aléatoires $(U_n)_{n\geqslant 0}$ de la manière suivante :
- $(X_n)_{n\in\mathbb{N}}, (Y_n)_{n\in\mathbb{N}}$ suite de variables aléatoires, indépendantes, identiquement distribuées, à valeurs dans Ω.
- $(U_n)_{n\in\mathbb{N}}$ suite de v.a., représentant l'image à la n-ième itération.
- $(\varepsilon_n)_{n\in\mathbb{N}}$ suite de v.a.i.i.d., suivant une loi uniforme sur $[0, 1]$.
- U_0 est une image initiale quelconque (bruit blanc)
- pour tout $n \geqslant 1$,

$$
\begin{aligned}
V_n &= U_n && (10)\\
V_n(X_n) &= U_n(Y_n) && (11)\\
V_n(Y_n) &= U_n(X_n) && (12)\\
p_n &= \min\left(\frac{\pi(V_n)}{\pi(U_n)}, 1\right) && (13)\\
D_n &= (U_{n-1}(X_n) - U_{n-1}(Y_n))(\mathbf{1}_{Y_n} - \mathbf{1}_{X_n}) && (14)\\
U_n &= U_{n-1} + \mathbf{1}_{\varepsilon_n < p_n} D_n && (15)
\end{aligned}
$$

Pseudo-code de l'algorithme

pour tout pixel $x \in \Omega$:
tirer aléatoirement $u(x)$ uniformément dans l'espace Λ des valeurs possibles pour un pixel

itérer :
$i =$ État initial de l'image
tirer aléatoirement x et y uniformément dans Ω
SI x et y ne sont pas voisins :
inverser les valeurs $u(x)$ et $u(y)$, le nouvel état est noté j
$p \leftarrow \min\left(\frac{\pi(j)}{\pi(i)}, 1\right)$
avec probabilité p, faire $i = j$

Remarque 3.1. Comme il est dit précédemment, cet algorithme conserve l'histogramme. Il est alors nécessaire, lorsque que l'on étudiera la chaîne de Markov associée, de faire attention à l'espace dans lequel le champ de Markov est à valeurs. En effet, nous travaillerons dans l'ensemble des images réalisables avec l'histogramme initial.

4 Un exemple concret : le modèle d'Ising

4.1 Le modèle d'Ising

4.1.1 Modèle physique

Soit Ω un pavé de $\mathbb{Z}^2$ (le domaine des images), typiquement

$$\Omega = \{0, \dots, M-1\} \times \{0, \dots, N-1\}.$$

On appelle image binaire toute fonction $u : \Omega \to \{-1, 1\}$. Notons E l'ensemble des images binaires. Le voisinage $V(x)$ d'un point $x \in \Omega$ est un multi-ensemble (ensemble avec possibilité de points répétés) de cardinal 4 constitué de tous les points y de Ω vérifiant $|y-x| = 1$, chacun étant répété 2 fois si son symétrique $2x - y$ n'est pas dans Ω (convention de symétrisation au bord de type Neumann). On considère alors le modèle d'Ising (M1) spécifié par la probabilité

$$p(u) = \frac{1}{Z} e^{-\mathcal{E}(u)},$$

(Z étant une constante de normalisation), où l'énergie est donnée par

$$\mathcal{E}(u) = \lambda \sum_{x \in \Omega} \sum_{y \in V(x)} u(x)u(y) \tag{16}$$

(λ est un paramètre réel, positif ou négatif).

Remarque 4.1. Ce modèle d'Ising est un cas particulier d'un modèle plus général, dont l'énergie s'exprime de la façon suivante

$$\mathcal{E}(u) = -\frac{J}{k} \sum_{x \in \Omega} \sum_{y \in V(x)} u(x)u(y) - \frac{H}{k} \sum_{x \in \Omega} u(x)$$

où k représente la constante de Boltzmann, et H et J respectivement le champ magnétique externe et l'énergie interne d'un dipôle magnétique. On a donc pris pour la suite $\frac{J}{k} = \lambda$, où λ est un paramètre, et $H = 0$.

Il fut introduit en 1925 par Ising pour comprendre les phases de transitions dans les roches ferromagnétiques. Il y a donc bien une réalité physique derrière ce modèle.

4.1.2 Modèle mathématique de l'algorithme de Metropolis-Hastings associé à Ising

Pour étudier nos algorithmes, il est nécessaire de formaliser leur représentation mathématique.

On appelle (A1) l'algorithme suivant, qui construit une suite d'images aléatoires $(U_n)_{n\geq 0}$:

– U_0 est une image aléatoire (distribution initiale à préciser).

– pour tout $n \geq 1$,

$$D_n = -2U_{n-1}(X_n)\mathbf{1}_{X_n}, \tag{17}$$

$$U_n = U_{n-1} + \mathbf{1}_{\varepsilon_n < t_n} D_n, \tag{18}$$

où X_n est une variable aléatoire uniforme sur Ω, $\mathbf{1}$ est la fonction indicatrice usuelle (d'un point ou d'une condition), ε_n est une suite de variables aléatoires uniformes sur $[0,1]$, et

$$t_n = e^{(\mathcal{E}(U_{n-1}) - \mathcal{E}(U_{n-1}+D_n))^-} \tag{19}$$

(avec la définition usuelle $t^- = \min(0,t)$). Les variables U_0, $(X_n)_{n\geq 1}$ et $(\varepsilon_n)_{n\geq 1}$ sont indépendantes. La suite $(U_n)_{n\geqslant 0}$ ainsi définie est une chaine de Markov homogène.

L'algorithme (A1) s'écrit, en pseudo-code (pour l'initialisation par un bruit blanc) :

> pour tout pixel $x \in \Omega$:
> tirer aléatoirement $u(x)$ uniformément dans $\{-1, 1\}$
> calculer $E = \mathcal{E}(u)$
> itérer :
> tirer aléatoirement x uniformément dans Ω
> $\delta \leftarrow 2\lambda u(x) \cdot \sum_{y\in V(x)} u(y)$
> avec probabilité e^{δ^-}, faire $u(x) \leftarrow -u(x)$ et $E \leftarrow E - \delta$

L'algorithme (A1) est en fait un algorithme de Metropolis-Hastings classique avec Ising. Cet algorithme converge bien vers un modèle d'Ising, comme nous allons le montrer.

4.2 Convergence de l'algorithme de Metropolis appliqué à Ising

Pour montrer la convergence de cet algorithme, on utilise le théorème 1.3.

Montrons que ce théorème s'applique bien ici.

Propriété 4.1. *La chaîne $U = (U_n)_{n\geqslant 0}$ est irréductible.*

Démonstration. Soient $u_1, u_2 \in E$.

On pose $A = \{x \in \Omega | u_1(x) \neq u_2(x)\} = \{x \in \Omega | u_1(x) = -u_2(x)\}$.

Soit $x \in A$. Il existe $p > 0$ tel que x soit choisit avec une probabilité p.

Soit alors $\delta = 2\lambda u_1(x) \sum_{y\in V(x)} u(y)$.

On effectue le changement $u_1(x) \leftarrow -u_1(x) = u_2(x)$ avec probabilité e^{δ^-}.

Ainsi, on change $u_1(x)$ en $u_2(x)$ avec une probabilité $pe^{\delta^-} > 0$. Ceci étant valable pour tous $x \in A$, il existe $p' > 0$ (produit fini de facteurs strictement positifs) tel que u_1 deviennent u_2.

La chaîne est bien irréductible.

□

Donc d'après le théorème 1.3, l'algorithme (A1) converge vers un certain modèle (M)

Propriété 4.2. *L'algorithme (A1) converge vers le modèle (M1).*

Pour cela, nous allons montrer que la distribution de probabilités spécifiant le modèle (M1) est stationnaire pour la chaîne de Markov associée à l'algorithme (A1), et donc, par unicité de la distribution stationnaire, que l'algorithme (A1) converge bien vers le modèle (M1).

Nous allons utiliser pour ce faire le théorème suivant :

Théorème 4.1. *Soit π une distribution de probabilité sur E. Soit P une matrice de transition. Si P est réversible, i.e. s'il existe Q telle que $\forall i, j \in E$*

$$\pi(i)q_{ij} = \pi(j)p_{ji}$$

alors π est une distribution stationnaire de P

que nous admettons.

Démontrons maintenant la convergence vers (M1)

Démonstration. Soient $u_1, u_2 \in E$.

Montrons que $\pi(u_2)P(U_{n+1} = u_1|U_n = u_2) = \pi(u_1)P(U_{n+1} = u_2|U_n = u_1)$

Si u_1 et u_2 diffèrent de plus d'un pixel,

$$P(U_{n+1} = u_1|U_n = u_2) = P(U_{n+1} = u_2|U_n = u_1) = 0$$

Sinon, $\exists x_0 \in \Omega / u_1(x_0) = -u_2(x_0)$. Alors :

$$\pi(u_2)P(U_{n+1} = u_1|U_n = u_2) = \frac{p}{Z}\exp[-\lambda \sum_{x\in\Omega}\sum_{y\in V(x)} u_2(x)u_2(y) + 2\lambda u_2(x_0)\sum_{y\in V(x_0)} u_2(y)]$$

or :

$$\begin{aligned}
-\lambda \sum_{x\in\Omega}\sum_{y\in V(x)} u_2(x)u_2(y) + 2\lambda u_2(x_0)\sum_{y\in V(x_0)} u_2(y) = & - \lambda \sum_{\substack{x\in\Omega \\ x\notin \tilde{V}(x_0)}}\sum_{y\in V(x)} \underbrace{u_2(x)}_{=u_1(x)}\overbrace{u_2(y)}^{=u_1(y)} \\
& + 2\lambda \underbrace{u_2(x_0)}_{=-u_1(x_0)}\sum_{y\in V(x_0)} u_2(y) \\
& - \lambda \sum_{i=1}^{4} \underbrace{u_2(x_i)}_{=u_1(x_i)}\sum_{y\in V(x_i)} u_2(y) \\
& - \lambda \underbrace{u_2(x_0)}_{=-u_1(x_0)}\sum_{y\in V(x_0)} u_2(y) \\
& - \lambda \sum_{i=1}^{4} u_1(x_i)\sum_{y\in V(x_i)} u_1(y) \\
& + \lambda \sum_{i=1}^{4} u_1(x_i)\sum_{y\in V(x_i)} u_1(y)
\end{aligned}$$

$$
\begin{aligned}
&= -\lambda \sum_{x\in\Omega} \sum_{y\in V(x)} u_1(x)u_1(y) + \lambda \sum_{i=1}^{4} u_1(x_i) \sum_{y\in V(x_i)} \underbrace{(u_1(y) - u_2(y))}_{=\mathbf{1}_{\{y=x_0\}}} \\
&= -\lambda \sum_{x\in\Omega} \sum_{y\in V(x)} u_1(x)u_1(y) + 2\lambda u_1(x_0) \sum_{y\in V(x_0)} u_1(y)
\end{aligned}
$$

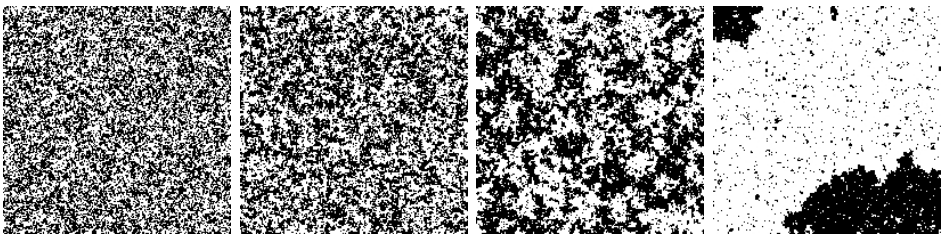
Conclusion, la chaîne est bien réversible, et d'après le théorème 4.1, la distribution spécifiant la méthode (M1) est stationnaire pour la chaîne associée à l'algorithme (A1) convergeant vers (M), et par unicité d'une telle distribution, (M) = (M1) □

On a ainsi justifié l'utilisation de l'algorithme de Metropolis-Hastings, au moins pour le modèle d'Ising, car il y a bien convergence vers la distribution stationnaire de la chaîne, dont la distribution n'est autre que celle du modèle d'Ising.

4.3 Images obtenues

Dans cette partie, nous allons nous intéresser à des chaînes en convergence. Savoir si les chaînes ont convergé reste un problème ouvert à l'heure actuelle, et nous essaierons d'aborder dans la partie quatre diverses techniques pouvant donner une indication quant à la convergence des images obtenues.

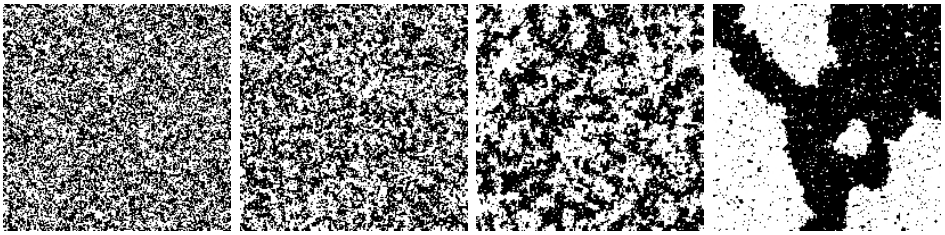
Remarque 4.2. Les images suivantes sont de tailles 200×200. Il est possible, et nous l'avons fait au début de notre travail, d'obtenir des images plus importantes (800×600), mais le calcul est alors beaucoup plus long, pour des algorithmes déjà peu rapides.



(a) $\lambda = 0.2$ (b) $\lambda = 0.3$ (c) $\lambda = 0.4$ (d) $\lambda = 0.5$

FIGURE 1 – Metropolis-Hastings classique, 10000 itérations.

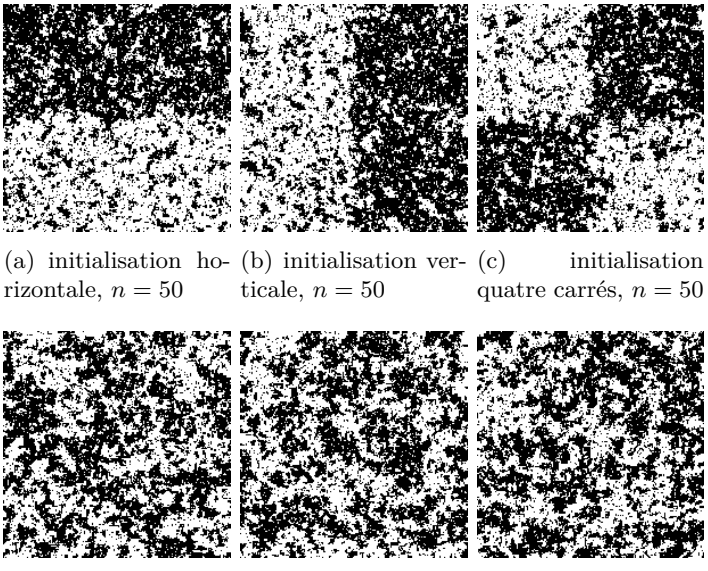
La première chose frappante dans les diverses images obtenues est que la valeur de λ influence grandement l'échelle des formes synthétisées. Ainsi, à partir d'une certaine valeur de λ, celle-ci variant selon la taille de l'image, les taches noires obtenues sont beaucoup plus grandes à l'échelle de l'image que pour un λ plus petit. De plus, on observe une relative régularité des formes. Cette régularité à un rôle dans les images obtenues en fonction de la condition aux bords choisie. Ainsi, si on choisit la condition aux bords de Neumann, qui impose une symétrie pixels aux bords (réflexion), il est nécessaire, pour conserver la régularité de la courbure des formes, que ces dernières arrivent orthogonalement aux bords. Cela s'observe bien sur les images ci dessus et ne se vérifie plus si l'on se place dans la condition aux bords torique. (voir partie sur la convergence.)



(a) $\lambda = 0.2$ (b) $\lambda = 0.3$ (c) $\lambda = 0.4$ (d) $\lambda = 0.5$

FIGURE 2 – **Metropolis-Hastingss avec échange, 10000 itérations.** On remarque la similitude avec les images (a), (b) et (c) de la figure 1. Pour les images (d), la conservation de l'histogramme empêche la disparition de l'une des deux couleurs.

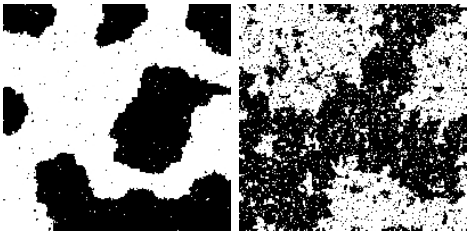
Une deuxième remarque très importante peut être faite lorsque l'on regarde les images (d) des figures 1 et 2. En effet, on observe avec l'algorithme de Metropolis-Hastings classique une dégénérescence de l'une des deux couleurs pour une valeur de λ suffisamment grande. En effet, la chaîne de Markov converge vers une image uniformément noire (ou blanche, les deux cas sont symétriques) légèrement bruitée de quelques pixels blancs, dus à l'aléa du procédé. Cette dégénérescence entraine une trivialisation des distributions stationnaires pour une large palette de paramètres. La mise en place du procédé d'échange prend alors tout son sens. La conservation de l'histogramme empêche la "disparition" de l'une des deux couleurs, et les textures seront aussi intéressantes après la convergence qu'au cours des itérations (voir remarque 4.3). Par la suite, c'est le cadre des algorithmes avec échange qui prévaudra.



(a) initialisation horizontale, $n = 50$ (b) initialisation verticale, $n = 50$ (c) initialisation quatre carrés, $n = 50$

(d) initialisation horizontale, $n = 500$ (e) initialisation verticale, $n = 500$ (f) initialisation quatre carrés, $n = 500$

FIGURE 3 – **Metropolis-Hastings classique,** $\lambda = 0.4$ Illustration de l'évolution de la chaîne à l'aide de différentes initialisations.



(a) Algorithme classique (b) Algorithme modifié, $p = 0.1$

FIGURE 4 – Comparaison des procédés classique et modifié, $\lambda = 0.7$, 1000 itérations.

4.4 Étude de cas

Comme nous l'avons vu précédemment, l'irréductibilité est une condition indispensable à la convergence des algorithmes. Cependant, il arrive parfois de rencontrer une chaîne associée à un algorithme qui ne soit pas irréductible. Pour la rendre irréductible, il est alors possible d'introduire un paramètre d'aléa, ce qui signifie qu'à chaque étape, on applique l'algorithme avec une probabilité donnée et sinon, on remplace un pixel par une valeur aléatoire. Cette technique permet de rendre nombre d'algorithmes convergents, et, même si l'exemple précédent était convergeant, nous avons essayé d'étudier mathématiquement ce que donnerait un modèle d'Ising modifié de la sorte.

Au vu de la figure 4, l'algorithme "Ising modifié" semble conduire à un algorithme de Metropolis-Hastings classique adapté au modèle d'Ising, en changeant simplement le paramètre λ. Nous allons montrer dans cette partie que notre intuition se révèle être fausse mathématiquement parlant.

4.4.1 Algorithme "Ising modifié"

On construit maintenant un nouvel algorithme (A2) en modifiant l'étape (18) de l'algorithme (A1). Pour tout n, on décide d'appliquer la formule (18) avec probabilité $1-p$, et dans le cas contraire de choisir $U_n(X_n)$ aléatoirement et uniformément dans $\{-1, 1\}$. Ceci nous donne un algorithme appliquant avec une probabilité $1-p$ l'algorithme d'Ising classique, et avec une probabilité p changeant un pixel aléatoirement. En itérant cet algorithme à partir d'une distribution initiale quelconque, on obtient alors des textures très proches des textures obtenues avec le modèle d'Ising classique. Il parait alors légitime de se demander si le modèle d'Ising modifié n'est pas un autre modèle d'Ising, avec un paramètre différent. C'est ce que nous allons étudier.

4.4.2 Etude mathématique de cet algorithme

Tout d'abord, la description mathématique de (A2) donne :
- U_0 est une image aléatoire (distribution initiale à préciser).
- pour tout $n \geq 1$,

$$\begin{aligned} D_n &= -2U_{n-1}(X_n)\mathbf{1}_{X_n}, \\ U_n &= U_{n-1} + (\mathbf{1}_{Y_n>p}\mathbf{1}_{\varepsilon_n<t_n} + \mathbf{1}_{Y_n\leqslant p}\mathbf{1}_{Z_n<\frac{1}{2}})D_n, \end{aligned}$$

où X_n est une variable aléatoire uniforme sur Ω, $\mathbf{1}$ est la fonction indicatrice usuelle (d'un point ou d'une condition), ε_n, Y_n, Z_n sont des suites de variables aléatoires uniformes sur $[0,1]$, et

$$t_n = e^{(\mathcal{E}(U_{n-1})-\mathcal{E}(U_{n-1}+D_n))^-}$$

Pour voir si l'algorithme (A2) et l'algorithme (A1) sont similaires, il faut essayer de mettre l'algorithme (A2) sous la même forme, ainsi mettons (t_n) sous une forme semblable à celle de l'équation 19. On peut réécrire (A2) de la manière suivante :

$$\begin{aligned} D_n &= -2U_{n-1}(X_n)\mathbf{1}_{X_n}, \\ U_n &= U_{n-1} + \mathbf{1}_{\varepsilon_n<\tilde{t}_n} D_n, \end{aligned}$$

avec

$$\tilde{t}_n = (1-p)t_n + \frac{p}{2}$$

On se ramène alors à une définition mathématique plus proche de celle de l'algorithme (A1). Cependant ce n'est pas suffisant pour conclure. Avant d'aller plus loin, il faut caractériser le modèle d'Ising classique pour pouvoir comparer efficacement avec ce modèle, et de prouver ou non que c'est un modèle d'Ising.

4.4.3 Caractérisation du modèle d'Ising classique

Propriété 4.3. *Un modèle d'Ising au sens de (16) est caractérisé par l'équation*

$$p(u(x)=1|u(y), y\neq x) = \frac{1}{1+F\left(\sum_{y\in V(x)} u(y)\right)}, \tag{20}$$

où les 5 réels $F(i)$, $i\in\{-4,-2,0,2,4\}$ sont en progression géométrique.

Pour démontrer ceci, on utilise le théorème suivant :

Théorème 4.2. *Deux distributions pour un RMF sur un espace fini Λ^Ω qui ont même spécification locale sont identiques.*

que l'on admet.

Démonstration. – La spécification locale d'un modèle d'Ising s'écrit de la manière suivante :

$$\pi(u(x)|u(y), y\neq x) = \frac{e^{\lambda u(x)\sum_{y\in V(x)} u(y)}}{e^{\lambda\sum_{y\in V(x)} u(y)} + e^{-\lambda\sum_{y\in V(x)} u(y)}}$$

On a donc, en particulier :

$$\begin{aligned} p(u(x)=1|u(y), y\neq x) &= \frac{e^{\lambda\sum_{y\in V(x)} u(y)}}{e^{\lambda\sum_{y\in V(x)} u(y)} + e^{-\lambda\sum_{y\in V(x)} u(y)}} \\ p(u(x)=1|u(y), y\neq x) &= \frac{1}{1+e^{-2\lambda\sum_{y\in V(x)} u(y)}} \end{aligned}$$

Et donc

$$F : i \mapsto e^{-2\lambda i}$$

– Réciproquement, supposons que l'on a le modèle suivant :

$$p(u(x) = 1 | u(y), y \neq x) = \frac{1}{1 + F(\sum_{y \in V(x)} u(y))}$$

où F est de la forme $i \mapsto a.q^i$ et $a, q \neq 0$. Alors :

$$\begin{aligned} p(u(x) = 1 | u(y), y \neq x) &= \frac{1}{1 + a.q^{\sum_{y \in V(x)} u(y)}} \\ p(u(x) = 1 | u(y), y \neq x) &= \frac{1}{1 + e^{\ln q \sum_{y \in V(x)} u(y) + \ln a}} \end{aligned}$$

Ce qui correspond à un modèle d'Ising, de paramètre $\lambda = \ln q$ et de terme constant non nul. Enfin, d'après le théorème 4.2, le modèle proposé est bien un modèle d'Ising.

□

Nous avons ainsi caractérisé le modèle d'Ising classique et nous allons pouvoir étudier le modèle d'Ising modifié pour vérifier sa similitude avec le modèle d'Ising classique.

4.4.4 Ising modifié n'est pas un modèle d'Ising

Propriété 4.4. *Le modèle avec aléa n'est pas un modèle d'Ising.*

On a :

$$p(u(x) = 1 | u(y), y \neq x) = \frac{1 - p}{1 + e^{-2\lambda \sum_{y \in V(x)} u(y)}} + \frac{p}{2}$$

Supposons qu'il existe a et q tels que :

$$p(u(x) = 1 | u(y), y \neq x) = \frac{1}{1 + a.q^{\sum_{y \in V(x)} u(y)}}$$

Posons $i = \sum_{y \in V(x)} u(y)$, si bien que $i \in \{-4, -2, 0, 2, 4\}$.

Soit alors f telle que $\forall i$:

$$(1 - p)\frac{1}{1 + e^{-2\lambda \sum_{y \in V(x)} u(y)}} + \frac{p}{2} = \frac{1}{1 + f(i)} \Leftrightarrow f(i) = \frac{1}{\frac{1-p}{1+e^{-2\lambda i}} + \frac{p}{2}} - 1$$

Montrons que f n'est pas géométrique. Pour cela, nous allons montrer, à l'aide d'une étude numérique, que :

$$\frac{f(2)}{f(0)} \neq \frac{f(4)}{f(2)}$$

Soit alors $\phi : (p, \lambda) \mapsto (f(4) - f(2))^2$. (En effet, $f(0) = 1$).
Il s'agit de démontrer que $\phi \neq 0$ dès lors que $p \in]0; 1[$ et $\lambda > 0$

On trace alors la surface représentant le graphe de ϕ, et on obtient la figure (1). Mais celle-ci n'est pas exploitable.

Une deuxième idée est de représenter la matrice des valeurs de $\phi(p, \lambda)$ en niveaux de gris, tels que le blanc représente la valeur 0. Pour plus de clarté, on effectue une transformation affine après chaque zoom. Ainsi, le blanc représente toujours la valeur 0, et le noir représente la valeur minimale de l'image. Les résultats de différents zooms sont affichés en figure (2). On peut ici remarquer que dans le cadre choisi, la fonction ϕ est bien non nulle, et on peut conclure.

Remarque : les échelles utilisées pour le calcul numérique sont les suivantes :

– λ : $[0.1 : 0.1 : 10]$

– p : $|0.01 : 0.01 : 1]$

Et l'utilisation d'autres échelles donne le même résultat.

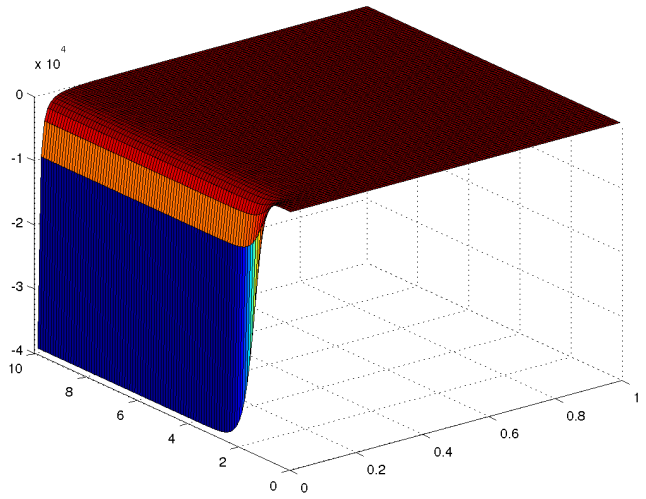
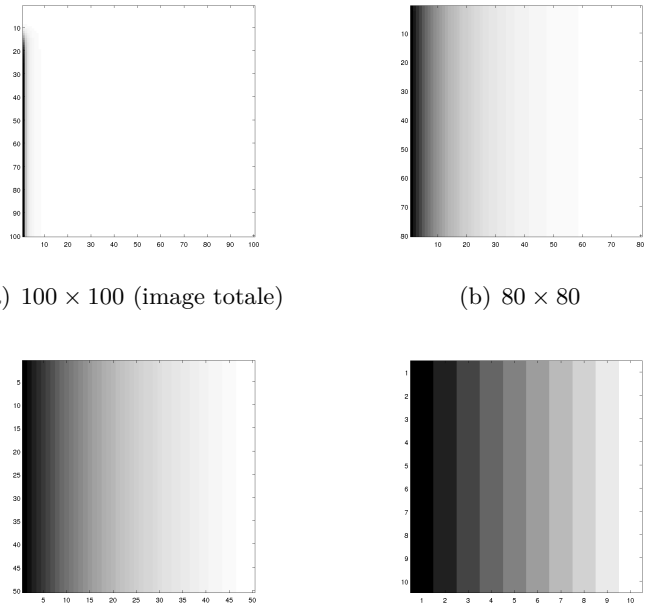


FIGURE 5 – $z = \phi(p, \lambda)$



(a) 100×100 (image totale)

(b) 80×80

(c) 50×50

(d) 10×10

FIGURE 6 – Représentation de ϕ en niveaux de gris : zoom sur le coin inférieur droit

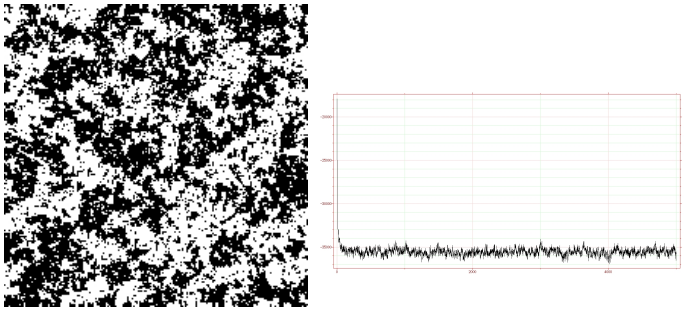
Troisième partie

Problèmes de convergence

Comme dit précédemment, les algorithmes de Metropolis et de l'échantilloneur de Gibbs converge dans notre cadre de travail. Cependant cette convergence est purement théorique, mais nous n'avons aucune information sur la vitesse de la convergence. L'espace E d'étude étant fini, il est possible d'étudier les valeurs propres de la matrice de transition associée à la chaîne de Markov pour obtenir des majorations de la vitesse de convergence (voir [1] chapitre 6). Hélas l'étude mathématique de cette matrice est très lourde, puisque nous avons affaire à des matrices de taille $\Lambda^\Omega \times \Lambda^\Omega$ ($2^{4000} \times 2^{4000}$ pour des images 200×200 dans le cadre du modèle d'Ising !) . Il nous faut alors utiliser des critères purement algorithmiques, que nous allons détailler ici.

L'énergie

Une première idée, naturelle dans le cadre d'algorithmes obtenus à partir de distribution de Gibbs, est celle du calcul de l'énergie. En effet, nos algorithmes tendent à faire diminuer l'énergie, mais une fois l'état stationnaire atteint, celle-ci reste aussi stationnaire. On admettra alors que l'état stationnaire est atteint dès lors que la courbe d'énergie *oscille autour d'une valeur moyenne*. La figure 7 représente notamment l'énergie en fonction du nombre d'itérations.



(a) Image de la distribution stationnaire

(b) Courbe d'énergie

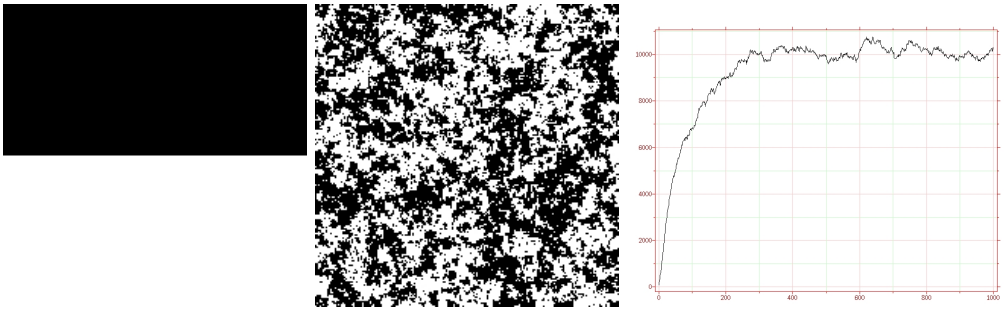
FIGURE 7 – Courbe d'énergie, 5000 itérations, $\lambda = 0.4$

Valeur d'une partie de l'image

S'en suit une multitude d'exemples prenant en compte l'évolution des valeurs des sites de l'image au cours des itérations. Ces méthodes sont surtout efficaces dans le cadre d'algorithmes d'échange. En effet, l'histogramme de l'image restant constant, il est pertinent d'étudier la valeur d'une zone de l'image. Nous avons mis en oeuvre l'un de ces exemples adapté au modèle d'Ising, et on utilise l'algorithme "Ising modifié" :

- On choisit comme initialisation l'image (a), de la figure 8
- A chaque itération de l'algorithme, on calcule la somme des valeurs des sites sur le rectangle supérieur.
- On trace l'évolution de cette valeur au cours des itérations.

Encore une fois, on admettra que l'on a atteint l'état stationnaire dès lors que la courbe oscille autour d'une valeur moyenne.



(a) Initialisation (b) Image de la distribution stationnaire (c) Somme des valeurs

FIGURE 8 – Somme des valeurs, 1000 itérations, $\lambda = 0.4$

Condition aux bords

Condition symétrique

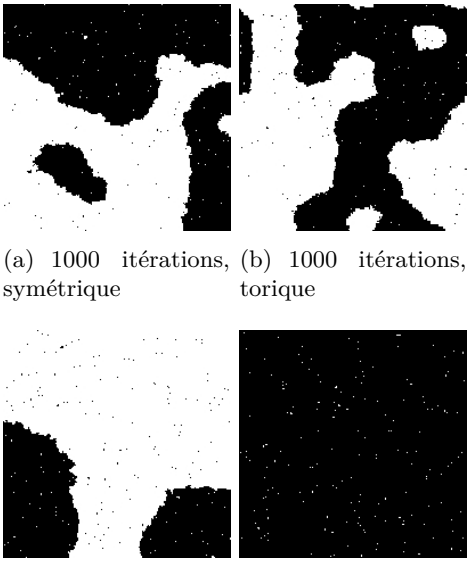
Tous nos algorithmes prennent en compte dans les calculs le voisinage du site étudié. Or, il existe des sites pour lesquels le voisinage est mal défini, ce sont les sites situés au bord de l'image, de sorte que le rayon du voisinage soit supérieur à la distance du point étudié au bord. Nous devons alors établir une convention afin de résoudre ce problème. On parle de condition aux bords.

Dans tout la plupart de nos algorithmes, nous utilisons la condition aux bords dite symétrique, déjà vue plus haut. Elle consiste en l'apposition d'un miroir au niveau des bords de l'image. Ainsi, la valeur d'un pixel s situé en dehors des bornes de l'image sera égale à la valeur du pixel t de l'image tel que s et t sont symétriques par rapport au site placé sur le bord de l'image.

Condition torique

Une autre convention est la condition torique. Dans ce cadre, on décide d'assimiler l'image à un tore, si bien qu'il y a continuité entre un bord de l'image et le bord opposé. Ainsi, la valeur d'un pixel s de coordonnées (x, y) situé en dehors des bornes de l'image sera égale à la valeur du pixel t de coordonnées $(x \mod nx, y \mod ny)$ où nx et ny sont les dimensions de l'image.

On remarque que cette méthode entraîne en général une accélération de la convergence de nos algorithmes, en particulier lorsque l'on utilise des voisinages réduits. Cela s'explique probablement par le fait que, dans le cas de la convention symétrique, certains pixels "comptent double", et donc la "disparition" des éléments de grande envergure est plus difficile aux bords de l'image.



(a) 1000 itérations, symétrique

(b) 1000 itérations, torique

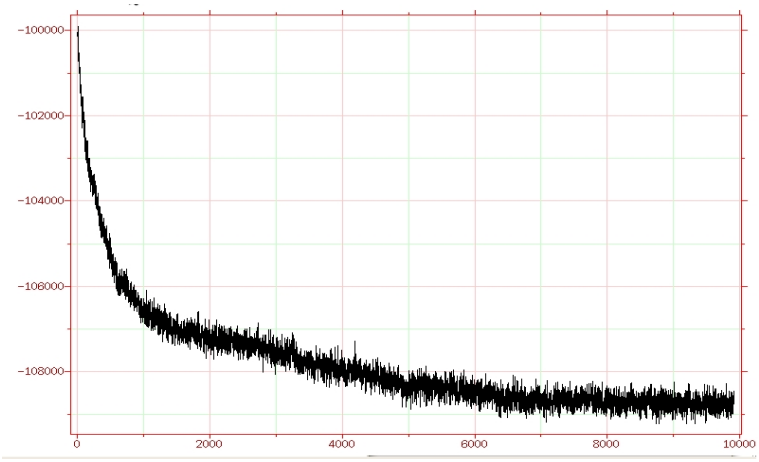
(c) 10000 itérations, symétrique

(d) 10000 itérations, torique

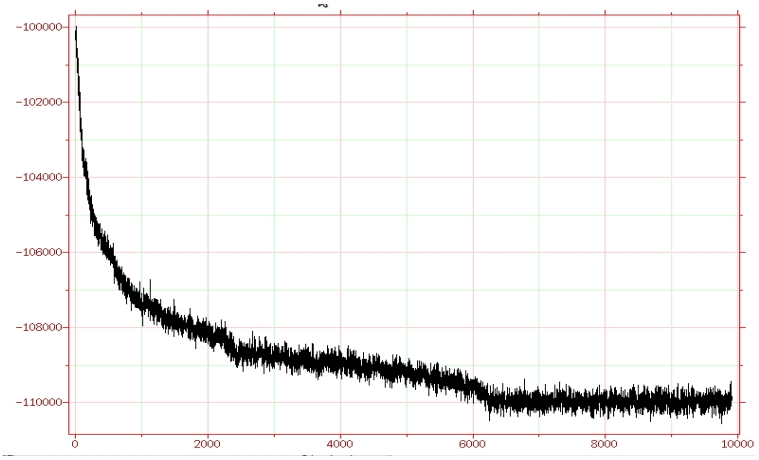
FIGURE 9 – **Comparaison des deux conventions,** $\lambda = 0.7$ On remarque en particulier que dans le cas la convention torique, l'image a atteint sa distribution stationnaire, contrairement au cas de la convention symétrique.

Quatrième partie

Des algorithmes vers les distributions



(a) Symétrique



(b) Torique

FIGURE 10 – **Energies correspondantes,** 10000 **itérations** La courbe d'énergie (a) oscille autour de la valeur -110000 après 6000 itérations, alors que la courbe (b) reste en moyenne au-dessus de cette valeur après 10000 itérations. On remarque que les deux courbes ont un comportement en paliers, et si la courbe (b) n'évoluera plus, la courbe (a) va encore diminuer jusqu'à atteindre un profil similaire à la courbe (b). Il faudra néanmoins un très grand nombre d'itération, dû à la difficulté d' "absorber" les coins de l'image.

Nous nous intéressons maintenant au problème inverse de la troisième partie. Nous partons en effet de descriptions de textures uniquement en termes d'algorithmes, et nous essayons de trouver des informations sur les propriétés des chaînes de Markov ainsi modélisées.

Remarque 4.3. Dans la plupart des algorithmes étudiés, on observe un phénomène de dégénérescence de l'image au cours des itérations. Ainsi, bien que les chaînes soient irréductibles, les probabilités de retour sont souvent très faibles. Par conséquent, pour la plupart des paramètres utilisés, la distribution supposée stationnaire étudiée sera triviale. Nous allons bien sûr nous y intéresser en premier lieu, mais nous allons aussi nous pencher plus en avant sur les images en cours de convergence. Ainsi, il n'est pas rare de pouvoir observer des images plus intéressantes au cours des itérations, plutôt qu'après la convergence. Il nous a donc semblé intéressant, au fil des expériences, de nous considérer aussi les images n'ayant pas convergé.

5 Algorithme "Écart quadratique au point central local"

5.1 Description du modèle

On considère un pavé $\Omega = \{0, ..., M\} \times \{0, ..., N\}$, et une image est une fonction $u : \Omega \to \{0, ..., 255\}$. Le voisinage $V(x)$ d'un point $x \in \Omega$ est constitué des points $y \in \Omega$ tels que $|x - y| < r$, où r est un entier fixé, avec convention de symétrisation de type Neumann. Dans la suite, on sera amené à utiliser la quantité

$$Q_2(u, x, z) = \sum_{y \in V(x)} (u(y) - u(z))^2$$

qui est l'écart quadratique au centre calculé dans le disque de centre x où on a remplacé x par z. Q_2 est une mesure de la proximité des niveaux de gris des pixels du voisinage de x par rapport à la valeur en x. L'algorithme présenté ci-dessous est un algorithme d'échange. Ayant tiré deux points au hasard, on les échange si l'échange fait diminuer la somme des quantités Q_2 calculées en chaque point. Sinon, on échange quand même avec une certaine probabilité. Cet algorithme est à comparer avec l'algorithme de "Metropolis-Hastings", car son fonctionnement en est proche, à ceci près que la notion d'énergie est remplacée par la notion d'écart quadratique, et que l'on ne prend en compte que l'énergie locale dans le calcul, et non l'énergie totale de l'image modifiée.

5.2 Description en pseudo-code

L'algorithme s'écrit en pseudo-code, pour l'initialisation par un bruit blanc :

pour tout pixel $x \in \Omega$:
    tirer aléatoirement $u(x)$ uniformément dans $[0, 255]$
itérer :
    tirer aléatoirement x_1 et x_2 dans Ω
    SI x_1 et x_2 ne sont pas voisins et $u(x_1) \neq u(x_2)$
        calculer $e_{1,1} = Q_2(u, x_1, x_1)$, $e_{1,2} = Q_2(u, x_1, x_2)$, $e_{2,2} = Q_2(u, x_2, x_2)$, $e_{2,1} = Q_2(u, x_2, x_1)$
        calculer $q = (e_{1,2} + e_{2,1}) - (e_{1,1} + e_{2,2})$
        SI $q < 0$ échanger $u(x_1)$ et $u(x_2)$
        SINON tirer t uniformément sur $[0, 1]$
            SI $t < e^{-\lambda q}$ faire l'échange quand même FIN SI
        FIN SI
    FIN SI

où λ est une constante positive.

5.3 Description mathématique

On construit une suite de variables aléatoires $(U_n)_{n\geq 0}$ à valeur dans Ω de la manière suivante :

- U_0 est un bruit blanc.
- pour tout $n \geq 1$,

$$D_n = (U_{n-1}(X_n) - U_{n-1}(Y_n))(\mathbf{1}_{Y_n} - \mathbf{1}_{X_n}), \tag{21}$$

$$U_n = U_{n-1} + (\mathbf{1}_{t_n>1} + \mathbf{1}_{t_n<1}\mathbf{1}_{\varepsilon_n<e^{-\lambda t_n}})D_n, \tag{22}$$

où (X_n, Y_n) sont des variables aléatoires uniformes sur l'ensemble des couples de points non voisins de Ω, ε_n est uniforme sur $[0, 1]$ et

$$t_n = (Q_2(U_{n-1}, X_n, Y_n) + Q_2(U_{n-1}, Y_n, X_n)) - (Q_2(U_{n-1}, X_n, X_n) + Q_2(U_{n-1}, Y_n, Y_n)) \tag{23}$$

Les variables U_0, $(X_n, Y_n)_{n\geq 1}$, et $(\varepsilon_n)_{n\geq 1}$ sont indépendantes.

5.4 Propriétés mathématiques

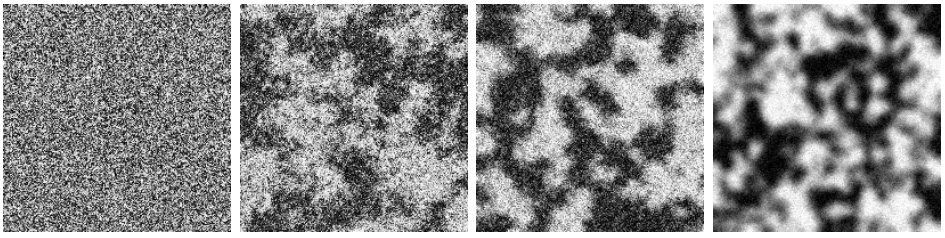
La suite $(U_n)_{n\geq 0}$ est une chaîne de Markov.

Propriété 5.1. *La chaîne $(U_n)_{n\geq 0}$ est irréductible.*

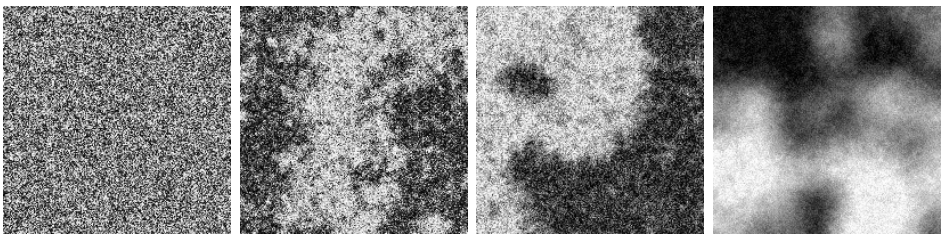
Démonstration. L'algorithme qui modélise la chaîne étant un algorithme d'échange, l'histogramme (i.e. la donnée pour chaque couleur du nombre de pixels de cette couleur) est constant.
Deux états différents de la chaîne correspondent donc à une permutation de l'ensemble des pixels disponibles. Si donc i et j sont deux images différentes, on peut écrire $j = \sigma(i)$ avec σ une permutation, que l'on peut à son tour écrire comme produit de transpositions.
Si chaque transposition avait une probabilité strictement positive de survenir (i.e. $\forall \tau_{x,y}$ $P(U_{n+1} = \tau_{x,y}(U_n)|U_n) > 0$, avec $\tau_{x,y}(u)$ l'image u où on a échangé les pixels x et y), on aurait $\exists N > n, P(U_N = j|U_n = i) > 0$ qui est ce que l'on cherchait.
Le problème est que certaines transpositions ne peuvent pas se produire : celles entre deux voisins - on ne s'occupe bien sûr pas des permutations entre deux pixels identiques. En revanche, toute transposition entre deux pixels non voisins a une probabilité strictement positive de survenir.
Pour conclure, il est donc suffisant de montrer que l'on peut écrire une transposition entre deux voisins comme produit de transpositions ne faisant pas intervenir de pixels voisins.
Pour cela, on suppose que l'on veut échanger les pixels x et y, où x est noir et y blanc, et x et y sont voisins. Pour cela, on cherche un point qui ne soit voisin ni de x ni de y. On travaille avec des voisinages suffisamment petits pour qu'il en existe un, et quitte à travailler avec y, on peut supposer que l'on a z tel que z ne soit voisin ni de x ni de y et ne soit pas noir. On échange alors x et z, puis x et y. On termine alors en échangeant z et y - si besoin. □

Remarque 5.1. Cette propriété sera vraie pour tous les algorithmes dans la suite.

Les images présentées sur les figures 11 à 13 nous permettent de tirer quelques conclusions quant au comportement des distributions.

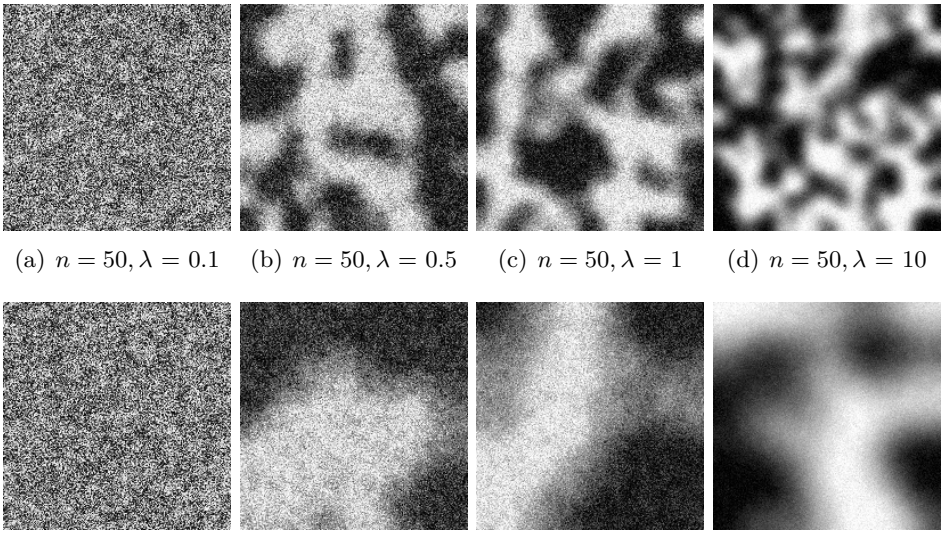


(a) $n = 50, \lambda = 0.1$ (b) $n = 50, \lambda = 0.5$ (c) $n = 50, \lambda = 1$ (d) $n = 50, \lambda = 10$



(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 11 – **"Écart quadratique au point central local", 1000 itérations, rayon = 3.** Lorsque λ augmente, le caractère bruité se fait de moins en moins présent, ce qui est à relier avec l'algorithme de Metropolis-Hastings appliqué à Ising. Dans le cadre d'images en niveaux de gris, l'augmentation de la valeur de λ tend à "lisser" les formes. Ceci s'explique en reprenant le pseudo code de l'algorithme : lorsque p est positif, c'est à dire lorsque le changement de pixel augmente localement l'écart quadratique, et donc ne lisse pas l'image, on l'accepte quand même avec une probabilité $e^{-\lambda p}$. Ainsi plus λ augmente, plus $e^{-\lambda p}$ devient petit quand p est positif, et donc moins le changement est probable. Quand λ augmente, on se rapproche donc d'un algorithme n'acceptant que les changement baissant l'écart quadratique, et donc lissant l'image.



(a) $n = 50, \lambda = 0.1$ (b) $n = 50, \lambda = 0.5$ (c) $n = 50, \lambda = 1$ (d) $n = 50, \lambda = 10$

(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 12 – **"Écart quadratique au point central local", rayon = 5.**

– le paramètre λ semble plutôt réduire la dimension des structures, contrairement à l'algorithme de "Metropolis-Hastings" appliqué à Ising, dans lequel l'effet inverse se produit.

Présenter des algorithmes différents n'a d'intérêt que si leurs propriétés sont différentes (notamment par exemple en matière de vitesse de convergence). Réciproquement, il ne sert à rien d'avoir des algorithmes différents si ils ont le même comportement. Nous allons donc voir par la suite s'il ne serait pas possible de modéliser cet algorithme avec un formalisme proche de celui de Metropolis-Hastings.

6 Algorithme "Écart quadratique au point central global"

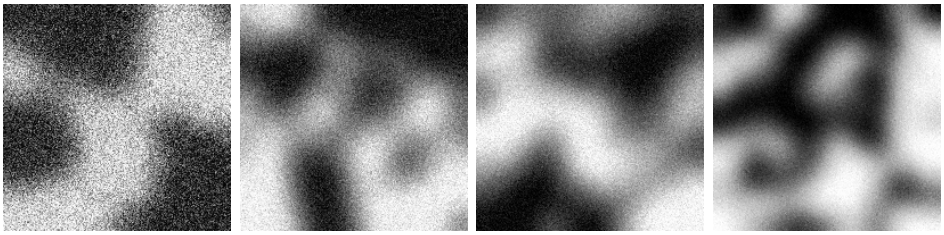
6.1 Description du modèle

L'algorithme que nous allons étudier pourrait s'intituler "pseudo Metropolis-Hastings", car son fonctionnement est similaire à celui de l'algorithme précédent. Cependant, celui-ci est plus fidèle à la représentation de l'algorithme de Metropolis-Hastings, dans le sens où l'on va maintenant considérer la variation d'énergie totale induite par un échange, plutôt que la simple variation locale. Si en théorie il parait plus justifié de s'intéresser à la variation d'énergie globale plutôt que locale, le calcul de cette dernière a un prix et rend cet algorithme beaucoup plus lent que le précédent.

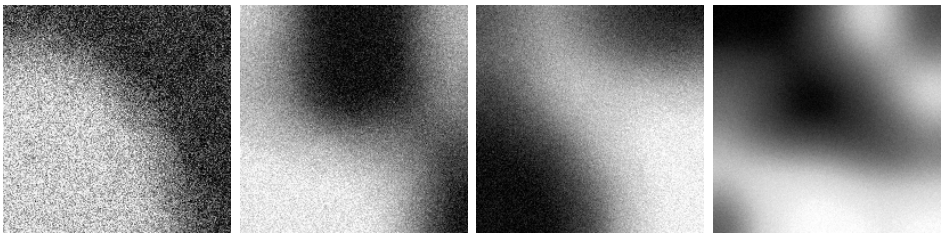
Nous allons ici utiliser la fonction suivante, qui somme la quantité Q_2 sur toute l'image.

$$Q(u, x, z) = \sum_{\substack{y \in V(x) \\ u(x) \leftarrow u(z)}} \sum_{y' \in V(y)} (u(y') - u(y))^2$$

De cette manière, les descriptions de l'algorithme "Écart quadratique au point central global" sont définies rigoureusement de la même manière que dans la section précédente, en remplaçant partout la fonction Q_2 par la fonction Q.



(a) $n = 50, \lambda = 0.1$ (b) $n = 50, \lambda = 0.5$ (c) $n = 50, \lambda = 1$ (d) $n = 50, \lambda = 10$



(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

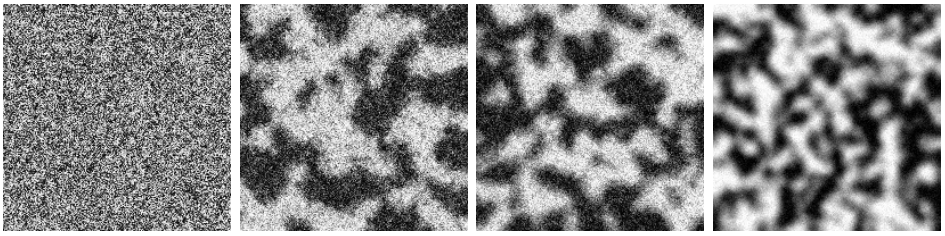
FIGURE 13 – **"Écart quadratique au point central local",** $r = 10$. Lorsque r augmente, on observe un effet de zoom (pour 50 itérations) par rapport au plus petit rayon, mais cette différence entre les images s'estompe lorsque le nombre d'itération augmente (1000 itérations). Ceci s'explique par le fait que l'échange se fait en fonction de l'écart quadratique au point central par rapport au voisinage choisi, et tend à diminuer cette quantité. Concrètement, l'algorithme lisse donc l'image sur le rayon choisi pour voisinage. Il est donc normal de retrouver une continuité sur un rayon plus grand lorsque l'on augmente la taille du voisinage. Cependant, asymptotiquement, on se rapproche d'un lissage sur toute l'image, quelque soit le voisinage choisi, ce qui estompe les différences entre les images de divers rayons.

Remarque 6.1. Au vue des valeurs prises par la quantité Q, on l'a divisé par l'ordre de grandeur de ces valeurs prises, de sorte que l'on puisse comparer de façon efficace l'influence du paramètre λ sur les images obtenues.

6.2 Propriétés mathématiques

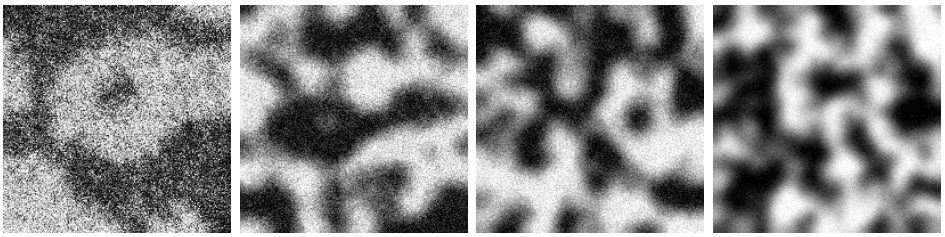
L'étude des images présentées sur les figures 14 à 16 induit quelques commentaires quant à l'évolution de la distribution :

- Le rôle des différents paramètres est le même que dans le cas de l'algorithme local.
- Les images sont sensiblement les mêmes que dans le cas de l'algorithme local. Ceci quelque soit le nombre d'itérations (nous n'avons affiché qu'une seule image après 1000 itérations, la figure 17, car le temps de calcul est proportionnel à r^4, de sorte qu'une itération pour $r = 15$ a un temps de calcul de 10 minutes.) Si ce résultat semble décevant en premier lieu, il a en réalité une conséquence importante : L'algorithme de Metropolis-Hastings pourrait être simplifié, puisque l'on obtient sensiblement les mêmes résultats en considérant uniquement la variation d'énergie locale dans le calcul. Le temps de calcul est ainsi divisé par un facteur r^2.



(a) $\lambda = 0.1$ (b) $\lambda = 0.5$ (c) $\lambda = 1$ (d) $\lambda = 10$

FIGURE 14 – "Écart quadratique au point central global", 50 itérations, rayon = 3



(a) $\lambda = 0.1$ (b) $\lambda = 0.5$ (c) $\lambda = 1$ (d) $\lambda = 10$

FIGURE 15 – "Écart quadratique au point central global", 50 itérations, rayon = 5

7 Algorithme "p-écart au point central local"

7.1 Description du modèle

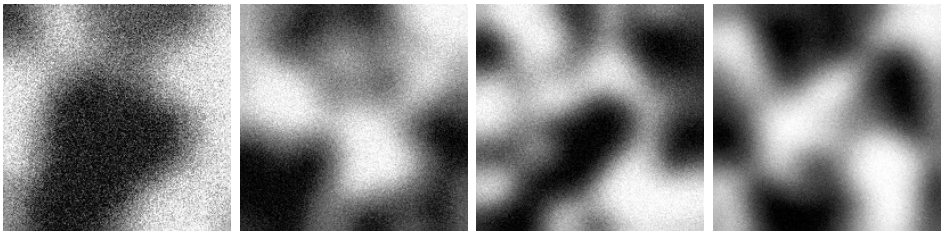
Ici, nous allons généraliser le principe de l'algorithme "Écart quadratique au point central" en modifiant la fonction Q_2 de la manière suivante :

$$Q_p(u, x, z) = \sum_{y \in V(x)} |u(y) - u(z)|^p$$

où p est une réel strictement positif. Les descriptions de l'algorithme s'en déduisent ainsi aisément puisqu'il suffit de remplacer la fonction Q_2 par la fonction Q_p dans les descriptions de la partie 5. De même, les caractéristiques de type irréductibilité ou convergence de l'algorithme restent vraies.

Remarque 7.1. Nous procédons de même que pour la remarque 6.1.

Nous allons présenter les résultats pour différentes valeurs de λ, de r et de p dans les figures 18 à 23.



(a) $\lambda = 0.1$ (b) $\lambda = 0.5$ (c) $\lambda = 1$ (d) $\lambda = 10$

FIGURE 16 – "Écart quadratique au point central global", 50 itérations, rayon = 10

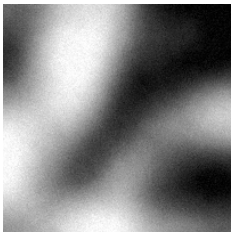


FIGURE 17 – **"Écart quadratique au point central global", 1000 itérations, rayon = 5,** $\lambda = 10$**.** Cette image ne diffère pas de celle obtenue par l'algorithme précédent (voir figure 12, (h)), alors que le temps de calcul est bien plus important.

8 Algorithme "p-Écart au point central modifié"

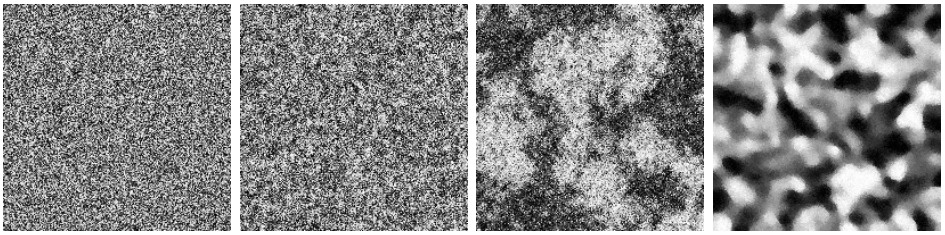
8.1 Description du modèle

Dans nos recherches sur l'algorithme d'écart quadratique au point central, nous avons à un moment fait une erreur dans le code, mais le code "buggué" a produit des textures intéressantes. Nous essayons dans la partie suivante de les retrouver en présentant un code "non buggué".

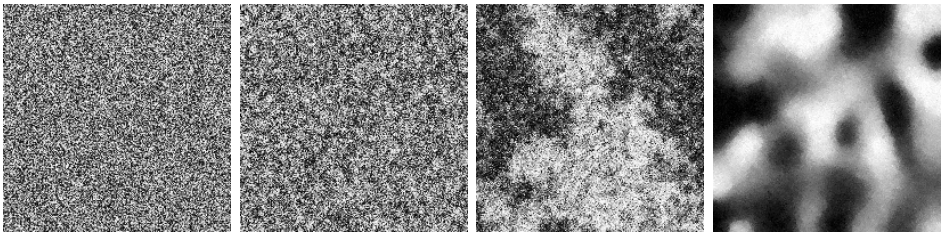
Nous allons ici utiliser de nouveau la fonction Q_p, et l'algorithme sera très proche de celui vu dans la partie 5. Cependant, cet algorithme visera à favoriser les augmentations d'énergie, plutôt que la diminution, que l'on étudie en général. Enfin, c'est ici le quotient entre la nouvelle énergie et l'ancienne qui est étudié, plutôt que la différence. Les descriptions sont donc modifiées de la manière suivante :

8.2 Description en pseudo-code

L'algorithme s'écrit en pseudo-code, pour l'initialisation par un bruit blanc :



(a) $n = 100, \lambda = 0.1$ (b) $n = 100, \lambda = 0.5$ (c) $n = 100, \lambda = 1$ (d) $n = 100, \lambda = 10$



(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 18 – **"p-Écart au point central", p = 1, rayon = 3.** Lorsque l'exposant p augmente, on observe une augmentation de la régularité des formes obtenues.

pour tout pixel $x \in \Omega$:
    tirer aléatoirement $u(x)$ uniformément dans $[0, 255]$
itérer :
    tirer aléatoirement x_1 et x_2 dans Ω
    SI x_1 et x_2 ne sont pas voisins et $u(x_1) \neq u(x_2)$
        calculer $e_{1,1} = Q_p(u, x_1, x_1)$, $e_{1,2} = Q_p(u, x_1, x_2)$, $e_{2,2} = Q_p(u, x_2, x_2)$, $e_{2,1} = Q_p(u, x_2, x_1)$
        calculer $q = \frac{e_{1,2}+e_{2,1}}{e_{1,1}+e_{2,2}}$
        SI $q > 1$ échanger $u(x_1)$ et $u(x_2)$
        SINON tirer t uniformément sur $[0, 1]$
            SI $t < e^{-\lambda q}$ faire l'échange quand même FIN SI
        FIN SI
    FIN SI

où λ est une constante positive.

8.3 Description mathématique

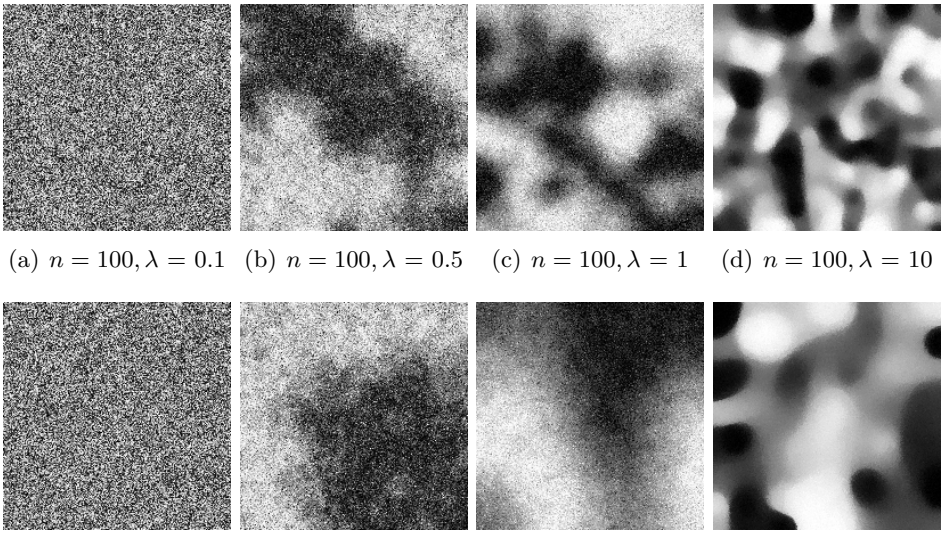
On construit une suite de variables aléatoires $(U_n)_{n\geq 0}$ à valeur dans Ω de la manière suivante :

- U_0 est un bruit blanc.
- pour tout $n \geq 1$,

$$D_n = (U_{n-1}(X_n) - U_{n-1}(Y_n))(\mathbf{1}_{Y_n} - \mathbf{1}_{X_n}), \tag{24}$$

$$U_n = U_{n-1} + (\mathbf{1}_{t_n>1} + \mathbf{1}_{t_n<1}\mathbf{1}_{\varepsilon_n<e^{-\lambda t_n}})D_n, \tag{25}$$

où (X_n, Y_n) sont des variables aléatoires uniformes sur l'ensemble des couples de points



(a) $n = 100, \lambda = 0.1$ (b) $n = 100, \lambda = 0.5$ (c) $n = 100, \lambda = 1$ (d) $n = 100, \lambda = 10$

(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 19 – **"p-Écart au point central", p = 1, rayon = 5.** Lorsque le rayon augmente, et pour les mêmes raisons que pour les algorithmes précédents, l'échelle des structures augmente.

non voisins de Ω, ε_n est uniforme sur $[0, 1]$ et

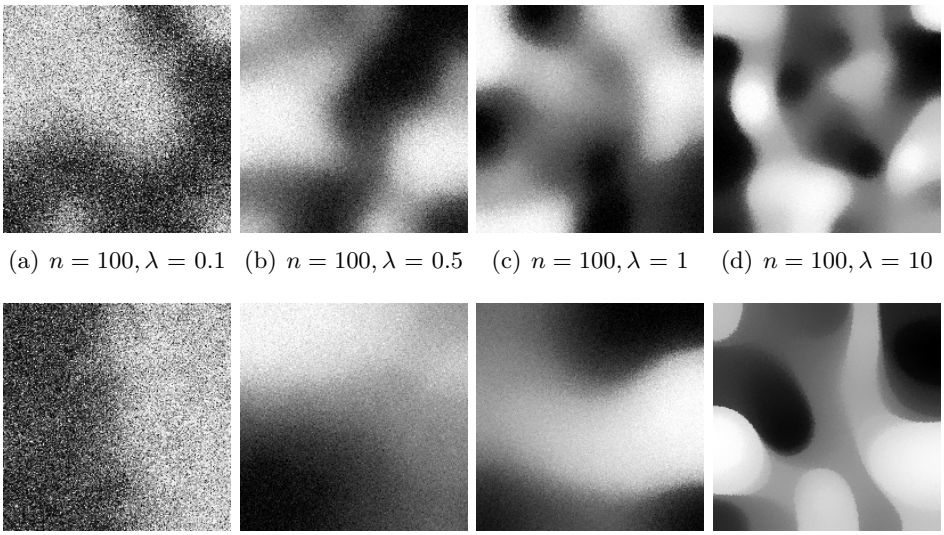
$$t_n = \frac{Q_p(U_{n-1}, X_n, Y_n) + Q_p(U_{n-1}, Y_n, X_n)}{Q_p(U_{n-1}, X_n, X_n) + Q_p(U_{n-1}, Y_n, Y_n)} \tag{26}$$

Les variables U_0, $(X_n, Y_n)_{n\geq 1}$, et $(\varepsilon_n)_{n\geq 1}$ sont indépendantes.

8.4 Observations

Les images obtenues sur les figures 24 à 28 sont assez différentes de celles obtenue jusqu'à présent, de part leur caractère très géométrique. En fait, on remarque que cet algorithme n'est intéressant que lorsque λ tend vers l'infini. En effet, celui-ci ayant tendance à augmenter la variance, les conséquences naturelles sont des images de bruit blanc, signe d'une régularité très faible.

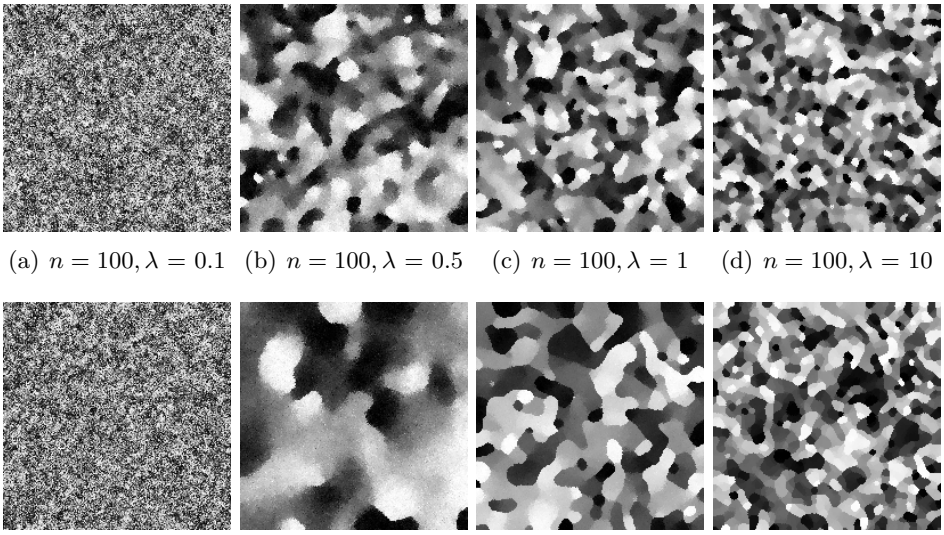
Dans le cas $p = 2$, on remarque que pour un nombre faible d'itérations, les structures sont celles de cercles noirs et blancs de rayon approximativement r. Mais lorsque le nombre d'itérations augmente (figure 27), on tend vers des images ayant des structures rayées, ce qui s'explique par le "rapprochement" des cercles entre eux. Dans le cas $p = \frac{1}{2}$, (figure 28) on n'observe plus seulement des éléments noirs ou blancs, mais aussi des formes d'un niveau gris moyen. Autrement dit, on retrouve des images localement proches de celles obtenues avec l'algorithme "p-écart au point central" lorsque p varie (visuel "treillis" pour $p = \frac{1}{2}$ par exemple).



(a) $n = 100, \lambda = 0.1$ (b) $n = 100, \lambda = 0.5$ (c) $n = 100, \lambda = 1$ (d) $n = 100, \lambda = 10$

(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

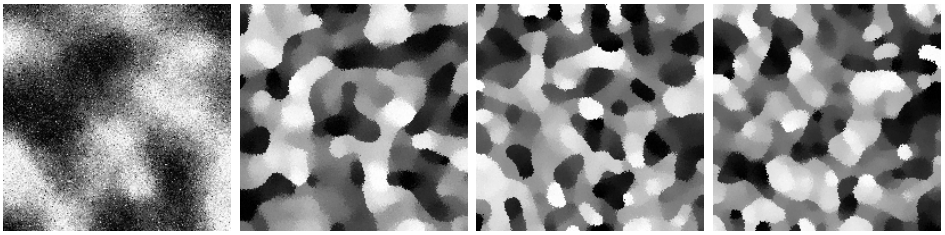
FIGURE 20 – "p-Écart au point central", p = 1, rayon = 10



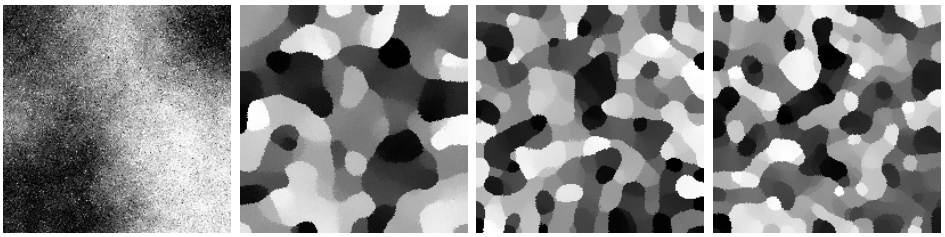
(a) $n = 100, \lambda = 0.1$ (b) $n = 100, \lambda = 0.5$ (c) $n = 100, \lambda = 1$ (d) $n = 100, \lambda = 10$

(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 21 – "p-Écart au point central", $p = \frac{1}{2}$, rayon = 3



(a) $n = 100, \lambda = 0.1$ (b) $n = 100, \lambda = 0.5$ (c) $n = 100, \lambda = 1$ (d) $n = 100, \lambda = 10$



(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 22 – "p-Écart au point central", $p = \frac{1}{2}$, rayon = 5

9 Algorithme du patch

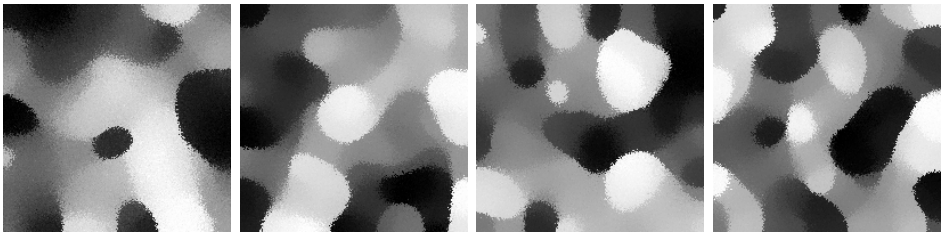
Remarque 9.1. *Nous avons manqué de temps pour étudier cet algorithme.*

Cet algorithme est un peu différent des précédents. On travaille toujours avec un pavé Ω du plan, chaque pixel pouvant être blanc ou noir. Une image est donc une fonction $u : \Omega \to \{-1, 1\}$. L'algorithme utilise une image initiale de petite taille appelée patch. Le voisinage d'un pixel est de la taille du patch, centré en ce pixel (on peut toujours centrer, car les dimensions du patch sont choisies impaires à cet effet). Lors de son déroulement, on cherche à maximiser la ressemblance entre le voisinage du pixel courant et le patch ou son inverse.
On note M_0 le patch (P étant déjà utilisé pour les matrices de transition) et pour $x \in \Omega$, $M(x)$ une zone de la taille du patch centrée en x.
$< M_0, M(x) >$ est le produit scalaire usuel entre M_0 et $M(x)$ considérés comme des matrices.
On implémente le modèle avec l'algorithme de Metropolis-Hasting, en utilisant la quantité :

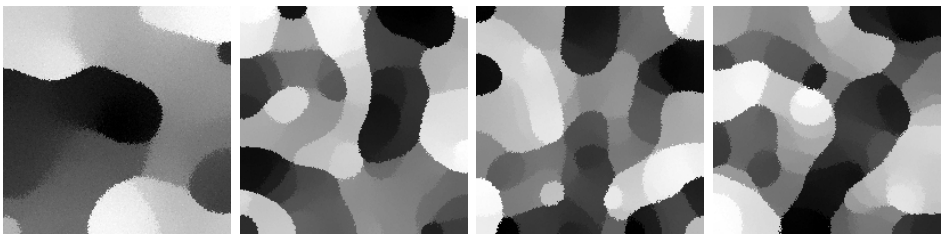
$$\mathcal{E}'(u) = -\lambda \sum_{x \in \Omega} | < M_0, M(x) > |$$

comme si c'était une énergie. On ne sait pas si c'est vraiment le cas, et comme nous le dirons dans la conclusion, ce pourrait être une nouvelle piste d'étude.

La chaîne construite est irréductible, la démonstration est similaire à la démonstration de l'irréductibilité de l'algorithme de Metropolis-Hastings dans le cadre du modèle d'Ising.

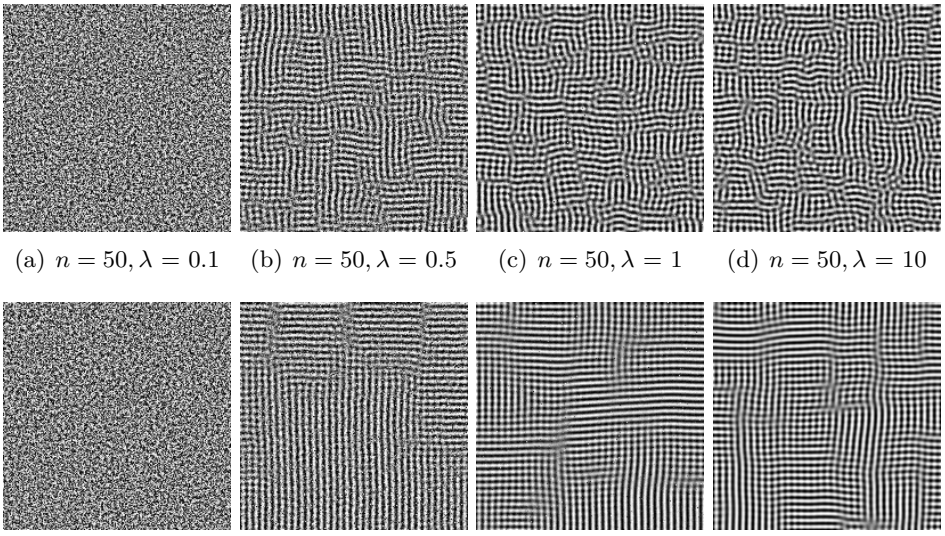


(a) $n = 100, \lambda = 0.1$ (b) $n = 100, \lambda = 0.5$ (c) $n = 100, \lambda = 1$ (d) $n = 100, \lambda = 10$



(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

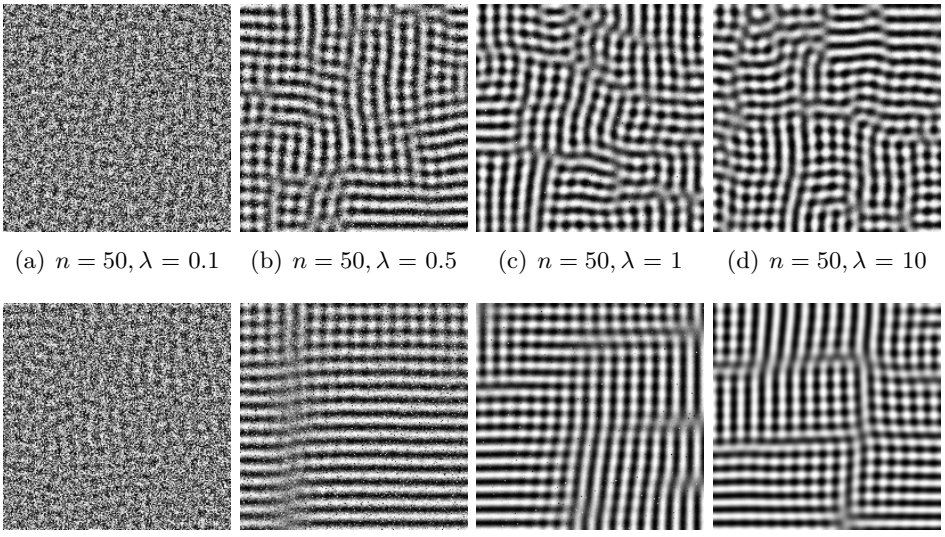
FIGURE 23 – **"p-Écart au point central", $p = \frac{1}{2}$, rayon = 10.** On observe avec $p = \frac{1}{2}$, des structures moins continues, de la taille du rayon choisi pour le voisinage. De plus on peut remarquer un effet de transparence : même si les formes sont d'une échelle donnée, chacune semble "se prolonger" sous une autre, de telle sorte qu'il y a une impression de continuité des structures, et ceux même si le niveau de gris n'est pas constant sur chaque forme. Si l'on ne sait pas expliquer la dernière remarque, on peut donner un début d'explication au lissage plus faible réalisé par cet algorithme. Sachant qu'il y a un nombre fini de valeurs prises pour un pixel, il y a aussi un nombre fini de valeurs prises par la quantité Q_p. Mais si l'exposant p n'influence pas la positivité de q, il influence la distribution des valeurs prises par Q_p et donc l'aléa permis par l'algorithme si q est positif.



(a) $n = 50, \lambda = 0.1$ (b) $n = 50, \lambda = 0.5$ (c) $n = 50, \lambda = 1$ (d) $n = 50, \lambda = 10$

(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

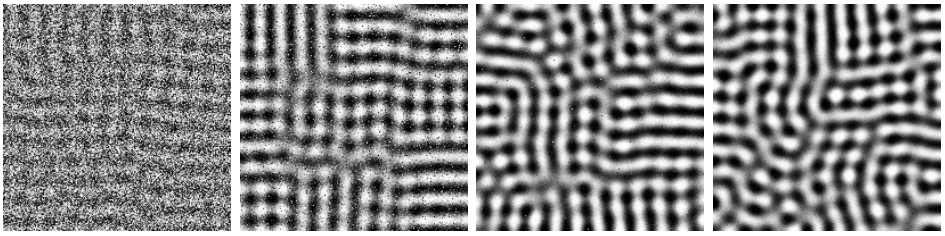
FIGURE 24 – "p-Écart au point central modifié", p = 2, rayon = 5



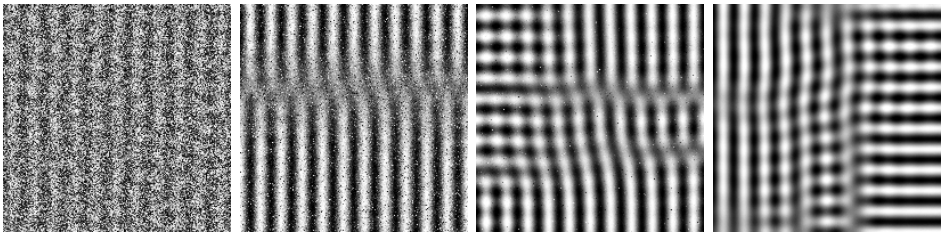
(a) $n = 50, \lambda = 0.1$ (b) $n = 50, \lambda = 0.5$ (c) $n = 50, \lambda = 1$ (d) $n = 50, \lambda = 10$

(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 25 – "p-Écart au point central modifié", p = 2, rayon = 10

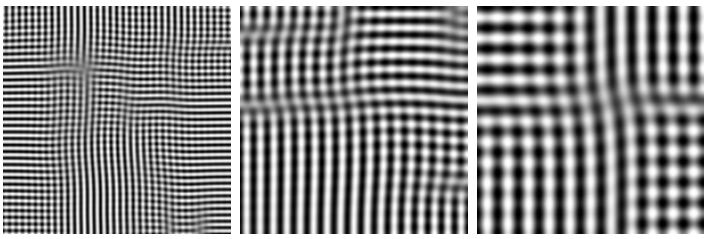


(a) $n = 50, \lambda = 0.1$ (b) $n = 50, \lambda = 0.5$ (c) $n = 50, \lambda = 1$ (d) $n = 50, \lambda = 10$



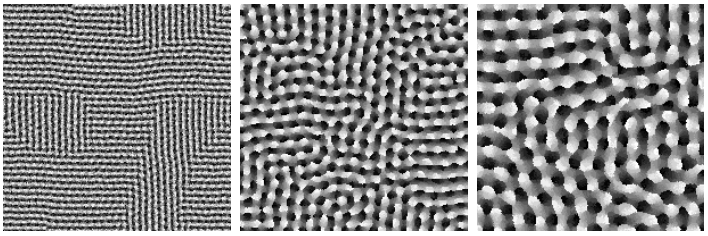
(e) $n = 1000, \lambda = 0.1$ (f) $n = 1000, \lambda = 0.5$ (g) $n = 1000, \lambda = 1$ (h) $n = 1000, \lambda = 10$

FIGURE 26 – "p-Écart au point central modifié", p = 2, rayon = 15



(a) $r = 5$ (b) $r = 10$ (c) $r = 15$

FIGURE 27 – "p-Écart au point central modifié", p = 2, 10000 itérations, $\lambda = 10$



(a) $r = 5$ (b) $r = 10$ (c) $r = 15$

FIGURE 28 – **"p-Écart au point central modifié", $p = \frac{1}{2}$, 1000 itérations, $\lambda = 10$.** Les remarques sur l'influence du rayon sont toujours les mêmes que précédemment. On observe ici des formes beaucoup plus droites qu'auparavant (voir la partie observation). Les formes géométriques obtenues s'expliquent par le fait que l'algorithme accepte avec une probabilité 1 les échanges de pixel *augmentant* localement l'écart quadratique et, si le paramètre λ est pris suffisamment grand, accepte avec une probabilité quasi-nulle les autres échanges. Ainsi, le meilleur moyen d'augmenter cette écart est de faire des formes géométriques (lignes verticales, ...) bien délimitées par un contour contrasté. Pour le cas $p = \frac{1}{2}$, on peut se ramener à la remarque sur la figure 23.

9.1 Observations

On observe que les images observées diffèrent grandement selon le patch utilisé, sans que nous puissions dégager un lien évident entre les deux.

On remarque cependant que dans le cas d'un patch de la forme

$$M_0 = \begin{pmatrix} 1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & -1 & 1 \\ 1 & -1 & 1 \end{pmatrix},$$

on obtient une texture avec des motifs verticaux (Figure 29, image (a)).

De même, avec la transposée du patch ci-dessus, on obtient des motifs horizontaux (Figure 29, image (b)).

La matrice $2I_3 - J_3$ donne une structure diagonale (Figure 29, image (c)), où on note J_3 la matrice 3×3 dont tous les coefficients sont égaux à 1.

Le patch

$$M_0 = \begin{pmatrix} -1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & -1 \end{pmatrix},$$

donne une sorte de labyrinthe (Figure 29, image (d)).

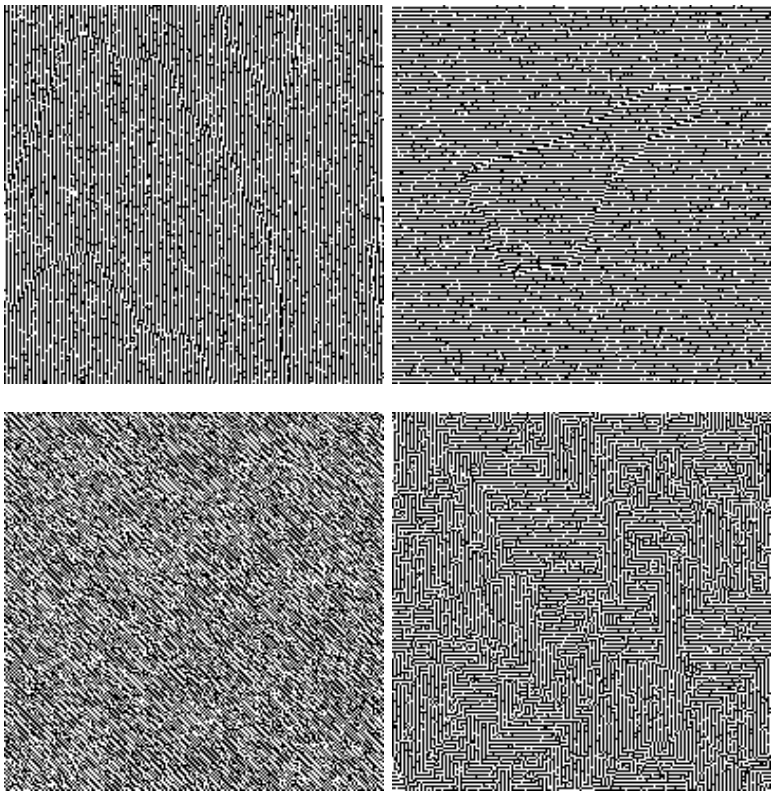


FIGURE 29 – Images obtenues à partir de différents patchs, $n = 500$

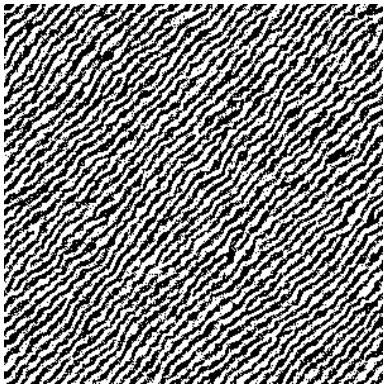


FIGURE 30 – Une structure diagonale obtenue avec un patch différent de celui utilisé dans la Figure 29, image (c)

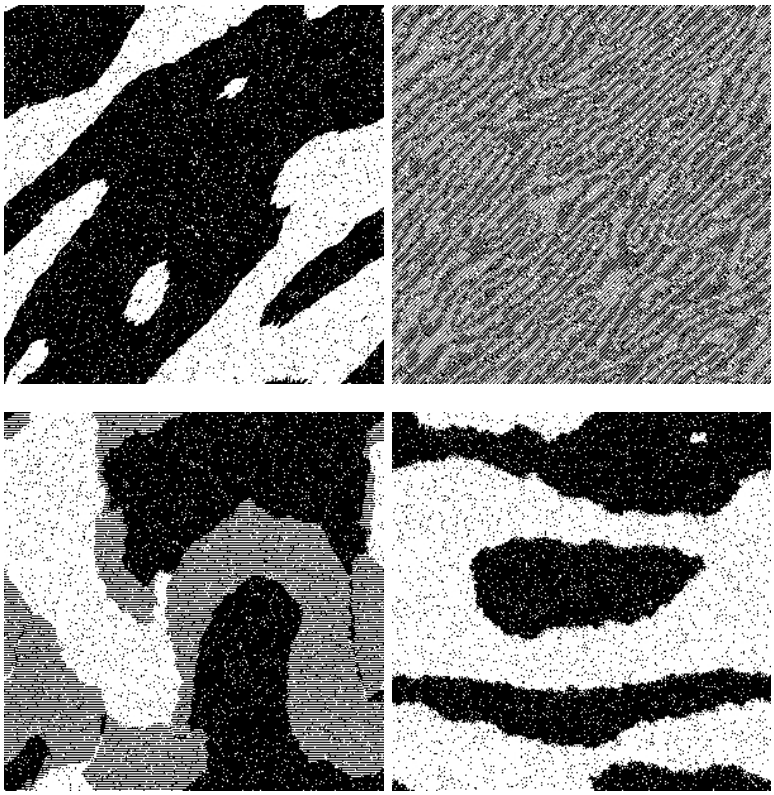


FIGURE 31 – Différentes textures obtenues avec des patchs 5×5

Conclusion

En l'état de nos travaux, quelques pistes semblent en constituer la continuation naturelle. Les deux premières concernent l'étude de l'algorithme de Metropolis-Hastings "local".

Nous avons vu en effet qu'il produit des textures intéressantes. Mais il serait surtout intéressant d'explorer ses liens avec l'algorithme de Metropolis-Hastings "local". En effet, nous avons constaté empiriquement que les textures obtenues avec les deux algorithmes sont très proches. Mais la version "locale" est beaucoup plus rapide que la version "classique" et ainsi, démontrer que les textures synthétisées sont *in fine* les mêmes, permettrait de réaliser des gains de temps importants lors des simulations.

Ensuite, il faudrait prouver que les textures générées par la version "locale" réalisent bien un champ aléatoire de Markov. Ceci déboucherait alors sur des recherches pour exprimer l'énergie associée. Encore une fois, c'est la proximité avec l'algorithme de Metropolis classique, bien associé lui à une énergie, qui peut donner l'espoir d'obtenir des résultats dans cette voie.

Le modèle du patch semble prometteur. En effet, il faudrait d'abord essayer de dégager des liens entre les motifs du patch et les textures obtenues. Nous avons aussi remarqué qu'avec certains patchs - et seulement certains - la texture semble se translater au cours des itérations. Il pourrait être intéressant de comprendre cet effet de déplacement.

Nous avons uniquement travaillé avec des voisinages circulaires. Or, de nombreuses formes sur les textures obtenues sont vaguement circulaires. Cela semble "normal", mais il serait alors intéressant d'étudier les résultats obtenus avec d'autres formes de voisinage.

La dernière piste à laquelle nous pensons à présent a trait à la convergence. Nous avons en effet discuté du phénomène de seuil qui apparaît dans la synthèse de nombreuses textures, et qui semble être le moment où celles-ci sont le plus intéressantes. Il faudrait essayer de comprendre pourquoi elles le sont plus à ce moment plutôt qu'en convergence, et de trouver des indicateurs de l'arrivée dans cette zone de "seuil".

Références

[1] Brémaud, Pierre, *Markov Chains - Gibbs Fields, Monte Carlo simulation, and Queues*, Springer, 1999.

[2] Efros, Alexei A. and Leung, Thomas K., *Texture Synthesis by Non-parametric Sampling*, September 1999

[3] Pérez, Patrick, *Markov Random Fields and Images*, 1998

Appendices

Exemple de code utilisé. Algorithme "Écart quadratique au point central local"

```
#include <stdio.h>
#include <math.h>
#include <time.h>
#include <unistd.h>
#include "mw.h"

Cimage metro2( int n, int *nx, int *ny, double r, double k)
{
  Cimage out;
  int i,j,x1,x2,y1,y2,xx1,xx2,yy1,yy2,dx,dy,ir,adr,k1,k2,va1,va2,v1,v2,v11,v22,pp;
  double r2,p,nbr,aux;
  int val1[10000];
  int val2[10000];

  /*** Initialize random seed ***/
  srand48( (long int) time (NULL) + (long int) getpid() );

  ir = (int)ceil(r);
  r2 = (r)*(r);

  /* create output image */
  out = mw_change_cimage(NULL,*ny,*nx);

  /* create a pure noise image */
  for (adr=0;adr<*nx*(*ny);adr++)
    out->gray[adr] = (int)floor(256.*drand48());

  /* main loop */
  for (i=0;i<n;i++) {
    printf("iteration %d\n",i+1);

    /* one iteration = nx * ny random steps */
    for (j=0;j<*nx*(*ny);j++) {

      /* choose two random pixels */
      x1 = (int)floor((double)(*nx)*drand48());
      x2 = (int)floor((double)(*nx)*drand48());
      y1 = (int)floor((double)(*ny)*drand48());
      y2 = (int)floor((double)(*nx)*drand48());

    /* proceed only if points are not neighbours */

        if ((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2) > r2) {
```

```
    if (out->gray[y1*(*nx)+x1] != out->gray[y2*(*nx)+x2]) {

  /* computes variances as they are and in case of switch */
  k1 = v1 = v11 = 0;
  k2 = v2 = v22 = 0;
  va1= out->gray[y1*(*nx)+x1] ;
  va2= out->gray[y2*(*nx)+x2] ;
  for (dx=-ir;dx<=ir;dx++)
    for (dy=-ir;dy<=ir;dy++)
      if ((double)(dx*dx+dy*dy)<r2) {
        xx1 = x1+dx;
        xx2 = x2 + dx;
        yy1 = y1+dy;
        yy2 = y2+dy;
        if (xx1<0) xx1 = -xx1-1;
        else if (xx1>=*nx) xx1 = 2*(*nx)-xx1-1;
        if (yy1<0) yy1 = -yy1+1;
        else if (yy1>=*ny) yy1 = 2*(*ny)-yy1-1;
        if (xx2<0) xx2 = -xx2-1;
        else if (xx2>=*nx) xx2 = 2*(*nx)-xx2-1;
        if (yy2<0) yy2 = -yy2+1;
        else if (yy2>=*ny) yy2 = 2*(*ny)-yy2-1;
        val1[k1] = (int)out->gray[yy1*(*nx)+xx1];
        val2[k2] = (int)out->gray[yy2*(*nx)+xx2];
        v1 += (val1[k1]-va1)*(val1[k1]-va1);
        v11 += (val1[k1]-va2)*(val1[k1]-va2); /* if v2 replaces v1 */
        v2 += (val2[k2]-va2)*(val2[k2]-va2);
        v22 += (val2[k2]-va1)*(val2[k2]-va1); /* conversely*/
        k1++;
        k2++;
      }

  /* switch if variances decrease globally, else do it with probability p */
    p = (v11+v22) - (v1+v2) ;
    aux = exp(-k*p/100000) ; /* because p has an order of 100000 */
    if (p < 0) {
    out->gray[y1*(*nx)+x1] = va2 ;
    out->gray[y2*(*nx)+x2] = va1 ;
             }
    else { nbr = drand48() ;
            if (nbr < aux) {
                            out->gray[y1*(*nx)+x1] = va2 ;
                            out->gray[y2*(*nx)+x2] = va1 ;
                             }
       }
}
}
```

```
  }
}
return(out);
}
```