ALGORITHMS FOR THE ISING MODEL

MARIO ULLRICH

Abstract. We describe some algorithms that output random spin configurations of the Ising model whose distribution is close to (or exactly) the Gibbs distribution using Markov chains.
Therefore we first introduce the Ising model and the related random cluster model and show that sampling on both state spaces is equivalent. Nevertheless simulation on the random cluster model seems to be easier, because since the Gibbs distribution at low temperature is multimodal, the random cluster distribution is (almost) unimodal. So we hope that simulation on the random cluster model is tractable, in contrast to simulation of the Ising model.
Next we describe some algorithms on both models algorithmically and show that they have the right stationary distribution. Additionally we introduce two methods that allow us to generate exactly distributed states. The methods are the Propp-Wilson (or Coupling-From-The-Past) algorithm and the Bounding chain technique.
Then we introduce techniques to evaluate integrals on the Ising model. As a consequence of the introduced algorithms we will see that we could make our estimators unbiased and so we save a lot of computing time.
Finally we show some results of simulations that were done with the self-made Matlab programs that are available on my homepage.

Date: February 9, 2011.

Contents

1. Introduction 3
2. Connection between Ising and Random Cluster model 7
3. Algorithms 13
3.1. Heat bath algorithm 16
3.2. Metropolis algorithm 18
3.3. Single-bond algorithm 23
3.4. Swendsen-Wang algorithm 28
3.5. Wolff's algorithm 31
3.6. Propp-Wilson algorithm 33
3.7. Bounding Chains for Swendsen-Wang 51
3.8. Conclusion 56
4. Approximation of Integrals 58
4.1. Spontaneous Magnetization 61
References 66

1. Introduction

First we describe the used models. For the *Ising model* consider a collection of vertices $V = \{1, \dots, N\}$ and a set of edges $E \subseteq \{\{x, y\} : x, y \in V,\ x \neq y\}$. In most cases these sets forms a regular lattice graph (V, E) with periodic boundary conditions.
A *spin configuration* σ is an assignment of spins, either -1 or 1, to each vertex, so $\sigma \in \{-1, 1\}^V =: \Sigma$. The energy of such configurations is defined by the Hamiltonian

$$H(\sigma) \;=\; J \sum_{\{x,y\} \in E} \mathbb{1}(\sigma_x \neq \sigma_y)\,,$$

where $\mathbb{1}(A)$ is the indicator function of the event A. Therefore $H(\sigma)$ is the number of edges e with unequal endpoints multiplied by J.
If $J > 0$, we call the model ferromagnetic.
The probability $\pi_\beta(\sigma)$ that an Ising spin configuration takes the value σ is given by the *Gibbs measure*

$$\pi_\beta(\sigma) \;=\; \frac{e^{-\beta H(\sigma)}}{Z}\,,$$

where $\beta = \frac{1}{k_B T}$ is the inverse temperature ($k_B =$ Boltzmann constant) and

$$Z \;=\; \sum_{\sigma} e^{-\beta H(\sigma)}$$

is the normalizing constant referred to as the *partition function.*

Remark 1. If we take $\widetilde{H}(\sigma) = -\widetilde{J} \sum_{\{x,y\} \in E} \sigma_x \cdot \sigma_y$ as the Hamiltonian, then we can take $J = 2\widetilde{J}$ in the original notation to get an equivalent model with the same probabilities for each spin configuration.
If we have for example $J = \widetilde{J}$, we get the same, if we change the value of β to $\frac{\beta}{2}$.

The question is, whether a phase transition for different temperatures exists or not. For the two-dimensional Ising model this is solved since 1944 due to Onsager. He solved the expected magnetization of the Ising model depending on the temperature for infinite regular square

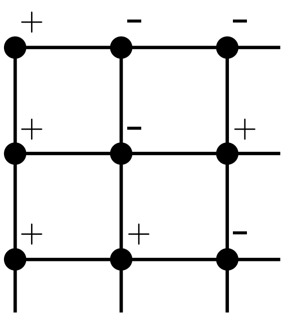


FIGURE 1. An Ising configuration with $H(\sigma) = 10$.

lattices ($N \to \infty$).

Let $m_k(T)$ denote the expected magnetization on a $k \times k$ lattice V, i.e.,

$$m_k(T) \;:=\; \frac{1}{k^2} \sum_{\sigma \in \{-1,1\}^{k^2}} \left| \sum_{v \in V} \sigma_v \right| \cdot \pi_\beta(\sigma),$$

where $\beta = \frac{1}{k_B T}$ and $\frac{1}{k^2} | \sum_{v \in V} \sigma_v |$ is the magnetization of a given spin configuration σ.

The expected magnetization of a infinite square lattice is than defined as

$$m(T) \;:=\; \lim_{k \to \infty} m_k(T).$$

Onsager [O] has solved the problem of a concrete representation of $m(T)$ in 1944. The solution is

$$m(T) \;=\; \begin{cases} \left(1 - \left(\sinh(\frac{J}{k_B T})\right)^{-4}\right)^{\frac{1}{8}}, & \text{if } T \leq T_c \\ 0, & \text{otherwise.} \end{cases}$$

Therefore we get the critical temperature

$$T_c \;=\; \frac{J}{\ln(1+\sqrt{2})\, k_B} \;\approx\; 1.1345926 \,\frac{J}{k_B}$$

and so

$$\beta_c \;=\; \frac{1}{J}\, \ln(1+\sqrt{2}) \;\approx\; 0.8813736 \,\frac{1}{J}$$

is called *Onsager's critical value.* At this value we have a phase transition in the two-dimensional Ising model.

Figure 2 shows the behavior of the expected (absolute) magnetization against the temperature. The dashed line is the critical temperature from above (we set $J = 1$).

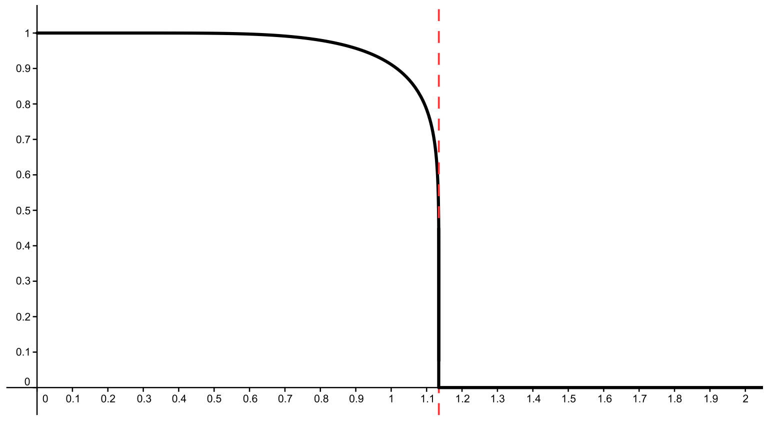


FIGURE 2. Magnetization against the temperature for $N = \infty$ (dashed line: critical temperature)

We see that at high temperatures the expected magnetization is zero, while at low temperatures we have a nonzero magnetization. We will see that this value T_c has a large significance for the below introduced algorithms. I.e., almost all algorithms are much faster above the critical temperature.

One extension of the Ising model is the *Potts model*. Here, each state is assigned a value from $1, \dots, Q$. If $Q = 2$ we just have the standard Ising model. Again $H(x)$ is the number of edges in a configuration with differently labeled endpoints times J, and the Gibbs distribution is the same as before.

Now we introduce the *random cluster model* as another way of looking at the Potts model, which was introduced by Fortuin and Kasteleyn [FK]. Suppose that the graph is $G = (V, E)$. Then the state space of the random cluster model consists of all spanning subgraphs $\omega \subseteq E$ (or $G_\omega = (V, \omega)$). Spanning means that we have the same set of vertices in the subgraph. Let $\Omega := \{\omega : \omega \subseteq E\}$ be the set of all subsets of E. The probability assigned to ω is

$$\mu_{p,Q}(\omega) \;=\; \frac{1}{Z_{\mathrm{RC}}}\, p^{|\omega|}\, (1-p)^{|E|-|\omega|}\, Q^{C(\omega)},$$

where $Z_{\mathrm{RC}} = Z_{\mathrm{RC}}(p, Q)$ is again the normalizing constant that makes this a probability distribution, so

$$Z_{\mathrm{RC}} = \sum_{\omega \in \Omega} p^{|\omega|} (1-p)^{|E|-|\omega|} Q^{C(\omega)}.$$

In addition p is a parameter of the model which varies from 0 to 1 and $C(\omega)$ is the number of connected components of the subgraph formed by ω (note that isolated vertices count as a component in this scheme).

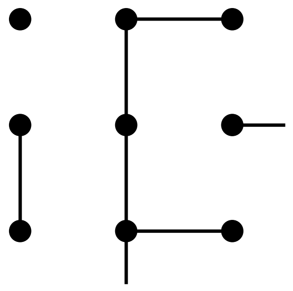


FIGURE 3. A random cluster with $|\omega| = 7$ and $C(\omega) = 3$.

Roughly speaking, larger p yields more edges.
For $Q = 2$ we will see that for $p = 1 - e^{-\beta J}$ (or $\beta = -\frac{1}{J}\ln(1-p)$), there is a tight connection between the random cluster model and the Ising model, which we present in the next section. Hence we call the random-cluster distribution μ_p instead of $\mu_{p,2}$.

2. Connection between Ising and Random Cluster model

We want to describe an algorithm that, given a random cluster state ω with respect to μ_p, outputs a random spin configuration, that is distributed according to π_β, and vice versa. Therefore we choose $Q = 2$ and $p = 1 - e^{-\beta J}$.

For a subset of edges $\omega \in \Omega$, let $\mathcal{C}_\omega$ denote the set of all connected components in (V, ω) and C_v denote the lowest numbered vertex in the component that contains v (recall that $V = \{1, \ldots, N\}$).
Moreover we have available a uniform distributed random variable U on the set $\Sigma = \{-1, 1\}^V$, i.e., $U(v) \sim \text{Uniform}\{-1, 1\}, \forall v \in V$.

Now we define a function $f : \Omega \times \Sigma \mapsto \Sigma$ by

$$f(\omega, U)(v) \;=\; U(C_v). \tag{1}$$

That is, we assign spins uniformly to each of the vertices so that all vertices within the same connected component of ω get the same spin. For computation the algorithm works as follows. We assume that we have a random cluster state ω, then

```
Step 1:   Enumerate all connected components of $\omega$ by
              $\mathcal{C}_\omega = \{\omega_1, \ldots, \omega_{C(\omega)}\}$.
Step 2:   Generate $U_1, \ldots, U_{C(\omega)}$ with
              $\mathbb{P}(U_i = -1) = \mathbb{P}(U_i = 1) = \frac{1}{2}$.
Step 3:   For all $v \in V$:
            If $v \in \omega_i$, then assign $U_i$ to $\sigma_v$, i.e. $\sigma_v = U_i$.
```

In Figure 4 we show how the algorithm works in an example.
The first picture shows the initial random cluster state. The second shows again the connected components together with the used random

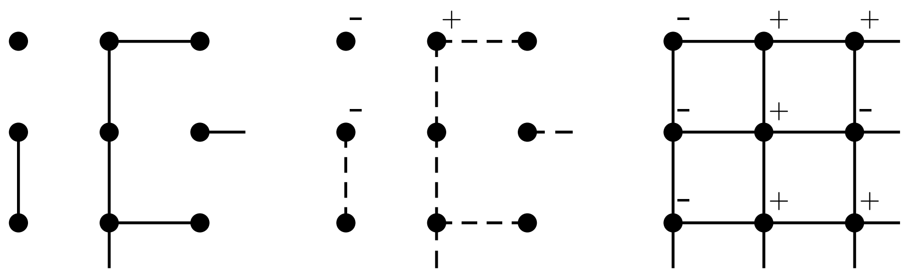


FIGURE 4. Transformation of a random cluster configuration.

spins that are assigned to the lowest numbered vertex in each component. The last picture is the resulting spin configuration.

We want to show that this function produces a state according to the Gibbs distribution, if the distribution of ω is μ_p. First, we state the following essential Lemma.

Lemma 2. *Let $p = 1 - e^{-\beta J}$. Then $Z = Z_{RC}$.*

Proof. The proof is based on a recursion relation for Z. See e.g. Fortuin, Kasteleyn [FK] or Grimmett [GG, (1.10)]. $\square$

Remark 3. If we take the Hamiltonian $\widetilde{H}$ from Remark 1, we can prove with the same techniques that $Z = e^{\beta \widetilde{J}|E|}\, Z_{\mathrm{RC}}$.
We will see, that this has no effect to the further propositions.

Now we are in the position to prove that this algorithm works.

Theorem 4. *Let $p = 1 - e^{-\beta J}$. Moreover let $Y \sim \mu_p$ and $U \sim$ Uniform(Σ) and assume that the function f is defined as above. Then*

$$X := f(Y, U) \sim \pi_\beta.$$

Proof. For $\sigma \in \Sigma$ let $C^*(\sigma)$ denote the number of edges in E with unequal ends, i.e., $C^*(\sigma) = \sum_{\{x,y\}\in E} \mathbb{1}(\sigma_x \neq \sigma_y) = \frac{1}{J} H(\sigma)$. Also let the set of random cluster states $A_\sigma := \{\omega \in \Omega : \exists u \in \Sigma : f(\omega, u) = \sigma\}$.

By the structure of the function $f(\omega, u)$ it is easy to see that we only need $C(\omega)$ (number of connected components) values of the vector u, so the other $|V| - C(\omega)$ components are arbitrary. Hence we get for any $\sigma \in \Sigma$

$$\begin{aligned}
\mathbb{P}(X = \sigma) &= \mathbb{P}(f(Y,U) = \sigma) = \sum_{\substack{(\omega,u)\in\Omega\times\Sigma: \\ f(\omega,u)=\sigma}} \mathbb{P}(Y = \omega, U = u) \\
&= \sum_{\substack{(\omega,u)\in\Omega\times\Sigma: \\ f(\omega,u)=\sigma}} \mathbb{P}(Y = \omega) \cdot \mathbb{P}(U = u) \\
&= \sum_{\omega\in A_\sigma} \mathbb{P}(Y = \omega) \sum_{\substack{u\in\Sigma: \\ f(\omega,u)=\sigma}} \frac{1}{2^{|V|}} \\
&= \sum_{\omega\in A_\sigma} \mu_p(\omega) \frac{2^{|V|-C(\omega)}}{2^{|V|}} = \frac{1}{Z_{\mathrm{RC}}} \sum_{\omega\in A_\sigma} p^{|\omega|} (1-p)^{|E|-|\omega|}.
\end{aligned}$$

The next to last equation results by applying the fact that for fixed ω exactly $|V| - C(\omega)$ components are arbitrary.
Now let us calculate the first sum. Therefore we need to know, which random cluster states ω are in A_σ for a given σ. It is easy to see that ω can contain at most $|E| - C^*(\sigma)$ edges, because if one of the other $C^*(\sigma)$ edges is in ω, the endpoints must have the same spin, in contrast to the definition of $C^*(\sigma)$.
Moreover there exist $\binom{|E|-C^*(\sigma)}{i}$ random cluster states ω with $|\omega| = i$. Therefore

$$\begin{aligned}
\mathbb{P}(X = \sigma) &= \frac{1}{Z_{\mathrm{RC}}} \sum_{\omega\in A_\sigma} p^{|\omega|} (1-p)^{|E|-|\omega|} \\
&= \frac{1}{Z} \sum_{i=0}^{|E|-C^*(\sigma)} \binom{|E| - C^*(\sigma)}{i} p^i (1-p)^{|E|-i} \\
&= \frac{(1-p)^{C^*(\sigma)}}{Z} \sum_{i=0}^{|E|-C^*(\sigma)} \binom{|E| - C^*(\sigma)}{i} p^i (1-p)^{|E|-C^*(\sigma)-i} \\
&= \frac{(1-p)^{C^*(\sigma)}}{Z} (p+1-p)^{|E|-C^*(\sigma)} = \frac{1}{Z} (e^{-\beta J})^{C^*(\sigma)} \\
&= \frac{1}{Z} e^{-\beta H(\sigma)} = \pi_\beta(\sigma).
\end{aligned}$$

This proves the claim. □

This theorem shows that it is enough to find a generator for random-cluster states that are distributed like μ_p to have a generator for (exactly) distributed random spin configurations with respect to the Gibbs distribution π_β.

Now we go the reverse way and define an algorithm that produces a random cluster state ω according to the distribution μ_p, if the given spin configuration σ is distributed according to π_β.
Therefore we need a random variable $\widetilde{U}$ with a product binomial distribution, i.e. $\widetilde{U} \sim \big(\mathrm{Bin}(1,p)\big)^E$, where $p = 1 - e^{-\beta J}$ like above. This means that $\widetilde{U}$ is a function on E, which assigns the values 1 or 0 to each edge $e \in E$ with probability p or $1-p$, respectively. Then we define the function $g : \Sigma \times \{0,1\}^E \mapsto \Omega$ by

$$g(\sigma, \widetilde{U}) \;=\; \big\{\{x,y\} \in E :\; \sigma_x = \sigma_y,\; \widetilde{U}(\{x,y\}) = 1\big\}. \tag{2}$$

Hence the algorithm for generating an $\omega \sim \mu_p$ from a spin configuration $\sigma \sim \pi_\beta$ works as follows.

```
Step 1:    Let $\Omega_\sigma := \big\{\{x,y\} \in E :\; \sigma_x = \sigma_y\big\}$

Step 2:    For all $e \in \Omega_\sigma$:
             Generate $\widetilde{U}(e)$ with
                $\mathbb{P}(\widetilde{U}(e) = 1) \;=\; 1 - \mathbb{P}(\widetilde{U}(e) = 0) \;=\; p.$

Step 3:    Output $\omega = \big\{e \in \Omega_\sigma : \widetilde{U}(e) = 1\big\}$.
```

Again Figure 5 shows how the algorithm works in an example.
The first picture is the initial spin configuration. The second picture shows all possible edges for the generated random cluster state, i.e., all edges which endpoints have a equal spin. The last picture shows the resulting random cluster state after Step 3 of the algorithm.

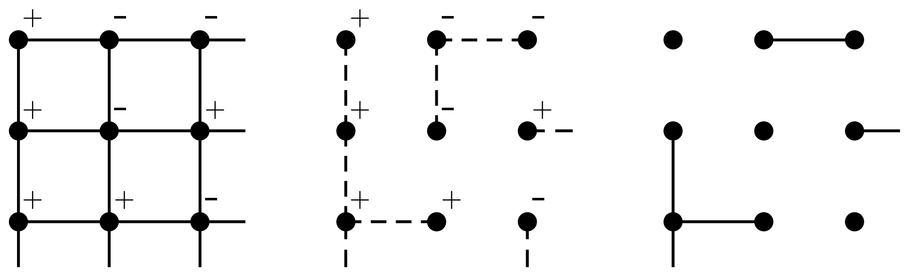


FIGURE 5. Transformation of an Ising configuration.

Now we prove that this algorithm is correct, too.

Theorem 5. *Let $p = 1 - e^{-\beta J}$. Moreover let $X \sim \pi_\beta$ and $\widetilde{U} \sim \big(Bin(1,p)\big)^E$ and assume that the function g is defined as above. Then*

$$Y := g(X, \widetilde{U}) \sim \mu_p.$$

Proof. Again for $\sigma \in \Sigma$ let $C^*(\sigma)$ denote the number of edges in E with unequal ends and C_σ denote $\{\{x,y\} \in E : \sigma_x = \sigma_y\}$. Also let the set of spin configurations $A_\omega := \{\sigma \in \Sigma : \exists \widetilde{u} \in \{0,1\}^E : g(\sigma, \widetilde{u}) = \omega\}$. Therefore $|A_\omega| = 2^{C(\omega)}$.
It is easy to see that we only need $|C_\sigma| = |E| - C^*(\sigma)$ (number of edges with equal ends) values of the vector $\widetilde{u}$, so the other $C^*(\sigma)$ components are arbitrary. Hence we get for any $\omega \in \Omega$

$$\begin{aligned}
\mathbb{P}(Y = \omega) &= \mathbb{P}(g(X,\widetilde{U}) = \sigma) = \sum_{\substack{(\sigma,\widetilde{u}) \in \Sigma \times \{0,1\}^E: \\ g(\sigma,\widetilde{u}) = \omega}} \mathbb{P}(X = \sigma, \widetilde{U} = \widetilde{u}) \\
&= \sum_{\substack{(\sigma,\widetilde{u}) \in \Sigma \times \{0,1\}^E: \\ g(\sigma,\widetilde{u}) = \omega}} \mathbb{P}(X = \sigma) \cdot \mathbb{P}(\widetilde{U} = \widetilde{u}) \\
&= \sum_{\sigma \in A_\omega} \pi_\beta(\sigma) \sum_{\substack{\widetilde{u} \in \{0,1\}^E: \\ g(\sigma,\widetilde{u}) = \omega}} \mathbb{P}(\widetilde{U} = \widetilde{u}).
\end{aligned}$$

Because of the fact that for a given random cluster state ω all values of $\widetilde{U}(e)$ for $e \in C_\sigma$ are determined for each σ and arbitrary for each

$e \in E \setminus C_\sigma$, we get the following.

$$\begin{aligned}
\mathbb{P}(Y=\omega) \;&=\; \sum_{\sigma \in A_\omega} \pi_\beta(\sigma) \sum_{\substack{\widetilde{u} \in \{0,1\}^E \\ g(\sigma,\widetilde{u})=\omega}} \mathbb{P}(\widetilde{U}=\widetilde{u}) \\
&=\; \sum_{\sigma \in A_\omega} \pi_\beta(\sigma) \cdot \\
&\qquad \sum_{\widetilde{u} \in \{0,1\}^E} \mathbb{P}(\widetilde{U}=\widetilde{u},\; \widetilde{u}(e)=1\,\forall e \in \omega,\; \widetilde{u}(e')=0\,\forall e' \in C_\sigma \setminus \omega) \\
&=\; \sum_{\sigma \in A_\omega} \pi_\beta(\sigma) \sum_{i=0}^{C^*(\sigma)} \binom{C^*(\sigma)}{i} p^{|\omega|+i}\,(1-p)^{|C_\sigma \setminus \omega|+C^*(\sigma)-i} \\
&=\; \frac{1}{Z} \sum_{\sigma \in A_\omega} e^{-\beta J C^*(\sigma)}\, p^{|\omega|}\,(1-p)^{|E|-C^*(\sigma)-|\omega|}\; 1^{C^*(\sigma)} \\
&=\; \frac{1}{Z_{\mathrm{RC}}} \sum_{\sigma \in A_\omega} p^{|\omega|}\,(1-p)^{|E|-|\omega|} \;=\; \frac{1}{Z_{\mathrm{RC}}}\, p^{|\omega|}\,(1-p)^{|E|-|\omega|} \sum_{\sigma \in A_\omega} 1 \\
&=\; \frac{1}{Z_{\mathrm{RC}}}\, p^{|\omega|}\,(1-p)^{|E|-|\omega|}\; 2^{C(\omega)} \;=\; \mu_p(\omega)
\end{aligned}$$

This proves the claim. □

Now we know that the random cluster model and the Ising model are tightly related in both directions. Therefore we can analyze different algorithms on both state spaces to obtain an efficient procedure for generating random spin configurations according to the (exact or an approximate) Gibbs distribution.

3. Algorithms

First of all we have to say that one can download all algorithms of this article from my homepage.
We want to describe some algorithms that produce in a sufficiently large number of iterations an approximate sample according to the desired distribution. I.e., we define different Markov chains with the right stationary distribution to get an efficient algorithm for generating such samples. Finally we describe possibilities to use such Markov chains to produce an exact sample according to the desired distribution.

Therefore we need the notation of *update rules* of a Markov chain $X = (X_t)_{t=0,1,\dots}$. An update rule ϕ is a deterministic function with two parameters. The first argument is the initial state and the second is a random variable U, mostly uniformly distributed in $[0,1]$, so that for all $x, y \in \Sigma$ and for all $t \in \mathbb{N}_0$

$$\mathbb{P}\big(\phi(x,U) = y\big) \;=\; q(x,y) \;=\; \mathbb{P}(X_{t+1} = y \mid X_t = x)\;,$$

where q is the matrix of transition probabilities for our discrete Markov chain.
Assume we have available the i.i.d. random variables $U_1, U_2, \dots$. For simplicity we define the functions $\phi_t(x) = \phi(x, U_t)$, then one step of the Markov chain is defined as

$$X_t \;=\; \phi_t(X_{t-1}) \quad \forall t \in \mathbb{N}.$$

So we have $X_t = \phi_t \circ \phi_{t-1} \circ \cdots \circ \phi_1(X_0)$ and the chain only depends on initial value X_0 and the random variables $U_1, U_2, \dots$. Hence we can say that the U_t's are the source of all the randomness used by such an algorithm.

Our goal is to define a Markov chain, which has stationary distribution π_β or μ_p, respectively. Therefore it is enough to show their reversibility with respect to the stationary distribution, i.e. $\pi_\beta(x)q(x,y) = \pi_\beta(y)q(y,x)\ \forall x, y \in \Sigma$ etc..
First we look at *single flip* algorithms. These algorithms mostly work

as follows: From a given configuration first choose one component (a vertex for the Ising model or an edge for the random cluster model) at random or deterministically and change the value of the state only on this component to get a new configuration, then accept this configuration with specified probability. In detail we have available a proposal matrix $\widetilde{q}$ and want to generate a matrix α of acceptance probabilities, which guarantees the desired reversibility. Therefore we prove the following lemma, which is applicable to all presented single flip algorithms.

Lemma 6. *Assume the proposal matrix $\widetilde{q}$ is symmetric. Let S be a finite state space and $\pi > 0$ a distribution on S. In addition let $b : (0,\infty) \mapsto [0,1]$ be a function satisfying $b(x) = x\, b(\frac{1}{x})$ for all $x \in S$. Then the transition matrix*

$$q(x,y) \;=\; \widetilde{q}(x,y)\,\alpha(x,y) \;+\; \delta_{x,y} \sum_{z\in S} \widetilde{q}(x,z)\,(1-\alpha(x,z))$$

with $\alpha(x,y) = b\big(\pi(y)/\pi(x)\big)$ is reversible with respect to the distribution π.

Proof. For $x = y$ the reversibility is trivial. So it is enough to prove it for $x \neq y$.

$$\begin{aligned}\pi(x)\,q(x,y) \;&=\; \pi(x)\,\widetilde{q}(x,y)\,\alpha(x,y) \;=\; \pi(x)\,\widetilde{q}(y,x)\,b\big(\pi(y)/\pi(x)\big) \\ &=\; \pi(x)\,\widetilde{q}(y,x)\,\frac{\pi(y)}{\pi(x)}\,b\big(\pi(x)/\pi(y)\big) \;=\; \pi(y)\,q(y,x)\end{aligned}$$

□

Now let us discuss in detail why this Lemma works for all single flip algorithms in this section. Therefore we first present the basic idea of such algorithms.

Let S (e.g. V or E) and M (e.g. $\{-1,1\}$) be finite sets. For simplicityn let $|M| = 2$. We number all elements of S serially, i.e., $S = \{1,\ldots,N\}$. Suppose we have $N = |S|$ proposal matrices $\widetilde{q}_i$, $i = 1,\ldots,N$, which takes only the ith coordinate of M^S deterministically and change the value on i, e.g. from -1 to 1, with probability 1. It is obvious that the matrices $\widetilde{q}_i$ are symmetric, because for $x, y \in M^S$ we have

$$\widetilde{q}_i(x,y) \begin{cases} 1, & \text{if } x \text{ and } y \text{ differs only in the } i\text{th coordinate} \\ 0, & \text{otherwise.} \end{cases}$$

In this section we describe only two types of single flip algorithms. The first is the random choice flip algorithm, that chooses one coordinate of M^S uniformly at random and changes the value only on this coordinate, and the second is the sweep algorithm. *Sweep algorithms* run deterministically through all coordinates S and so each coordinate of a state will be changed in a fixed time.
Since each of the proposal matrices $\widetilde{q}_i$ is symmetric we can apply Lemma 6 to all used proposal matrices to generate transition matrices that are reversible with respect to the distribution π, say q_i.
We use this transition matrices, which operates only on one cordinate, to produce the transition matrices for the random choice and the sweep algorithm. The transition matrix for the random choice flip algorithm fulfills

$$q^{\text{ran}} = \frac{1}{N} \sum_{i=1}^{N} q_i$$

and for the sweep algorithm we have (suppose without loss of generality that we go in increasing order through S)

$$q^{\text{det}} = q_1 \cdot q_2 \cdot \ldots \cdot q_N.$$

For q^{ran} it is easy to prove that it is reversible with respect to π. For the transition matrix of the sweep algorithm we lost the reversibility in general, but we keep the stationarity. Recall that π is the stationary distribution of all q_i and therefore we see this by

$$\pi\, q^{\text{det}} = \pi \cdot q_1 \cdot q_2 \cdot \ldots \cdot q_N = \pi \cdot q_2 \cdot \ldots \cdot q_N = \pi.$$

Now it is clear that all below presented single flip algorithms have the right stationary distribution and so they are working correctly.

In the following we want to present a few algorithms exactly and want to prove that they are converge to the desired distributions.

3.1. **Heat bath algorithm.** The *heat bath algorithm* is one of the most used algoritms in statistical physics and is a procedure on the set Σ of all spin configuration of an Ising system. In this algorithm we choose a vertex $v \in V$ at random (or sometimes determined so that each vertex is chosen at a fixed time) and update the spin configuration σ only at this vertex according to the conditional probability of π_β. I.e., produce $\widetilde{\sigma}$ according to

$$\begin{aligned}\pi_\beta\big(\widetilde{\sigma}_v = 1 \mid \widetilde{\sigma}_w = \sigma_w \ \forall w \neq v\big) \ &= \ \frac{\pi_\beta(\sigma^{v+})}{\pi_\beta(\sigma^{v+}) + \pi_\beta(\sigma^{v-})} \\ &= \ \left(1 + \exp\left[2\beta \sum_{w\colon \{v,w\}\in E} \left(\mathbb{1}(\sigma(w) \neq 1) - \frac{1}{2}\right)\right]\right)^{-1}\end{aligned}$$

and

$$\begin{aligned}\pi_\beta\big(\widetilde{\sigma}_v = -1 \mid \widetilde{\sigma}_w = \sigma_w \ \forall w \neq v\big) \ &= \ \frac{\pi_\beta(\sigma^{v-})}{\pi_\beta(\sigma^{v+}) + \pi_\beta(\sigma^{v-})} \\ &= \ \left(1 + \exp\left[2\beta \sum_{w\colon \{v,w\}\in E} \left(\mathbb{1}(\sigma(w) \neq -1) - \frac{1}{2}\right)\right]\right)^{-1},\end{aligned}$$

where σ^{v+} denote the spin configuration with $\sigma^{v+}(v) = 1$ and $\sigma^{v+}(w) = \sigma(w)$ for all $w \neq v$ (equivalent for σ^{v-}). Then, if we have selected the vertex v deterministicly, our update rule is

$$\phi^{\mathrm{hb},v}(\sigma, U) \ = \ \begin{cases} \sigma^{v+}, & \text{if } \ U < \frac{\pi_\beta(\sigma^{v+})}{\pi_\beta(\sigma^{v+})+\pi_\beta(\sigma^{v-})} \\ \sigma^{v-}, & \text{otherwise.} \end{cases} \tag{3}$$

This means we take the function $a(x) = \frac{x}{1+x}$ as acceptance function in Lemma 6.

If we want to choose a vertex v at random, we must include this step in the update rule ϕ, too. Therefore we could take $U \sim \mathrm{Uniform}[0,1]^2$ and use the first component as before whereas the second is to choose a random v. We will analyze this algorithm, because it is easier to understand how it works and we define the update rule of this algorithm as

$$\phi^{\mathrm{hb}} \ := \ \phi^{\mathrm{hb},\,\mathrm{Uniform}(V)}. \tag{4}$$

Heat bath algorithm

```
Input     Initial spin configuration $\sigma \in \Sigma$ and $t_{\max} \in \mathbb{N}$

Step 1:   Set $t = 1$.

Step 2:   Choose $v \in V$ uniformly at random and generate
            $U_t \sim \text{Uniform}[0,1]$.

Step 3:   If $U_t < \pi_\beta(\sigma^{v+})/\big(\pi_\beta(\sigma^{v+}) + \pi_\beta(\sigma^{v-})\big)$:
            Set $\sigma_v = 1$.
          Else: Set $\sigma_v = -1$.

Step 4:   If $t < t_{\max}$:
            Set $t = t+1$ and go to Step 2.
```

Reversibility of this chain with respect to π_β is proven by Lemma 6.

It is well-known that the heat bath algorithm is rapidly mixing (i.e. it has a mixing time bounded above by a polynomial in N) when $J\beta < J\beta_c$, where β_c is Onsager's critical value (for an square lattice $J\beta_c = \ln(1+\sqrt{2}) \approx 0.8813736$). This is proven in Martinelli, Olivieri [MO, Theorem 5.1]. However, this is only an existence statement.
For $J\beta > J\beta_c$ the behavior of the mixing time is not proven at the moment, but we are sure (and see it in simulations) that at least for very low temperatures we have exponentially increase of the mixing time and so this algorithm is slow mixing. We call this problem the *critical slowing down.*
The reason for this behavior is that for large β a spin configuration is more likely to be "all spins up" or "all spins down", but with the heat bath algorithm it is improbable to go from one of these states to the other.

Later we will discuss algorithms on another state space, which are (hopefully) not affected by the critical slowing down that way. However, they are partially very close to the heat bath algorithm.

3.2. **Metropolis algorithm.** The *Metropolis algorithm* works exactly like the heat bath algorithm, but we take the function $a(x) = \min\{1, x\}$ as acceptance function and so

$$\alpha(\sigma, \tau) \;=\; \min\left\{1, \frac{\pi_\beta(\tau)}{\pi_\beta(\sigma)}\right\}.$$

Let σ^v denote the configuration with $\sigma^v(v) = -\sigma(v)$ and $\sigma^v(w) = \sigma(w)$ for all $w \neq v$. Then we get

$$\alpha(\sigma, \sigma^v) \;=\; \min\left\{1,\ \exp\left[2\beta \sum_{w\colon \{v,w\}\in E} \left(\mathbb{1}(\sigma(w) \neq \sigma(v)) - \frac{1}{2}\right)\right]\right\}$$

and so for the deterministic choice of the vertex v the update rule

$$\phi^{\mathrm{M},v}(\sigma, U) \;=\; \begin{cases} \sigma^v, & \text{if } U < \alpha(\sigma, \sigma^v) \\ \sigma, & \text{otherwise.} \end{cases} \tag{5}$$

Again we define ϕ^M as in (4).
The Metropolis algorithm works as follows.

Metropolis algorithm

```
Input     Initial spin configuration $\sigma \in \Sigma$ and $t_{\max} \in \mathbb{N}$

Step 1:   Set $t = 1$.

Step 2:   Choose $v \in V$ uniformly at random and generate
            $U_t \sim \mathrm{Uniform}[0,1]$.

Step 3:   If $U_t < \exp\left[2\beta \sum_{w\colon \{v,w\}\in E} \left(\mathbb{1}(\sigma(w) \neq \sigma(v)) - \frac{1}{2}\right)\right]$:
            Set $\sigma_v = -1 \cdot \sigma_v$.

Step 4:   If $t < t_{\max}$:
            Set $t = t + 1$ and go to Step 2.
```

The probability of the Metropolis algorithm to change the sign of a spin σ_v is always larger than for the heat bath algorithm, so the Metropolis algorithm should be faster.

Now we want to present some Ising states that are generated with the Metropolis algorithm on the Ising model. States that are generated with the heat bath algorithm look very similar. In each case the initial state is the "all spins up" (all white) state.
First we show in Figure 6 an Ising state after 100 steps of the random choice and the sweep update algorithms at the critical temperature to show that the sweep algorithms are more efficient in the case of the Ising model. One step of the random choice algorithm means that the algorithm was applied N times, so that both update versions handled with the same number of vertices in each step. Another advantage is that one sweep is much faster in simulations than a random choice step, because we can use matrix operations for update.

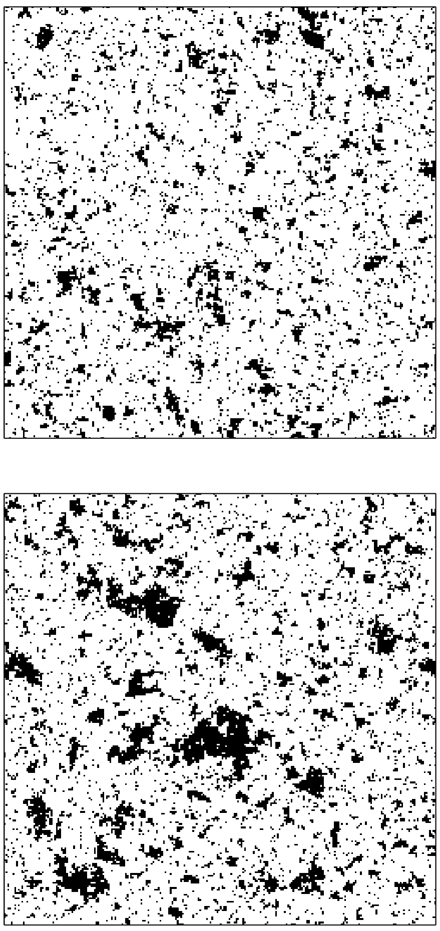


FIGURE 6. An Ising state after 100 steps (up - random choice / down - sweep)

From now we only show states that are generated with the sweep algorithm.
Next in Figure 7 we show a state in the high temperature case, i.e., the stationary disribution is near to the uniform one. Here the algorithms on the Ising model, like Metropolis, are very fast and so we need only 10 sweeps to get a good result, i.e. we have magnetization 0 and nearly uniform distributed spins.

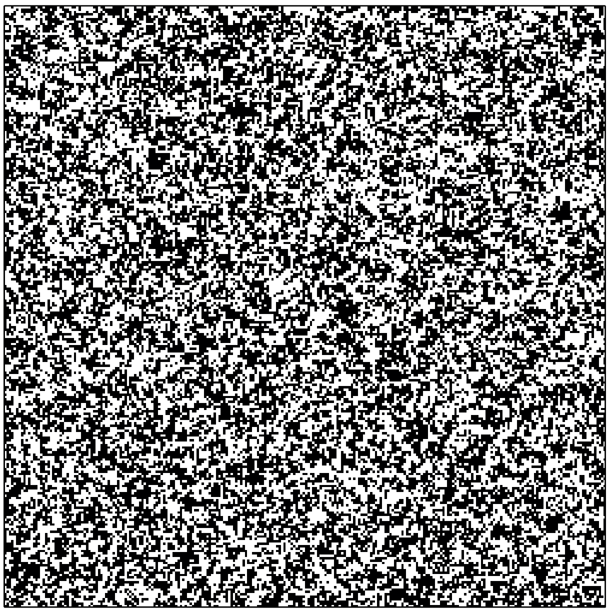


FIGURE 7. An Ising state for $\beta = \frac{1}{2}\beta_c$ after 10 sweeps

In such cases it is not necessary to use other algorithms or other state spaces, because the convergence to equilibrium is fast enough, also for large lattice sizes.
In the low temperature case we are in another situation. As mentioned above, the probability to go from the "all spins up" to the "all spins down" state is very low and in practice we will never see this transition. Therefore this algorithms on the Ising model are useless to generate a sample for calculation of expectations. Figure 8 shows a Ising state after 100000 steps with start in the "all spins up" (white) state. We see that even after a very large number of steps, we are still near the initial state.

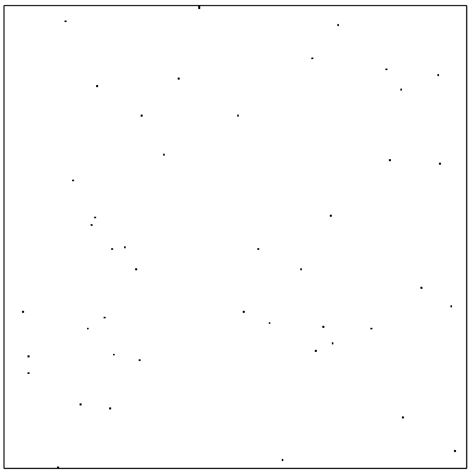


FIGURE 8. An Ising state for $\beta = 2\beta_c$ after 100000 sweeps

On the other hand it is interesting to see, what happens, if we start the algorithm in another initial state. For example in a state that is distributed like in the high temperature case. This situation corresponds to the cooling-down of a ferromagnet. Figure 9 shows such a situation.

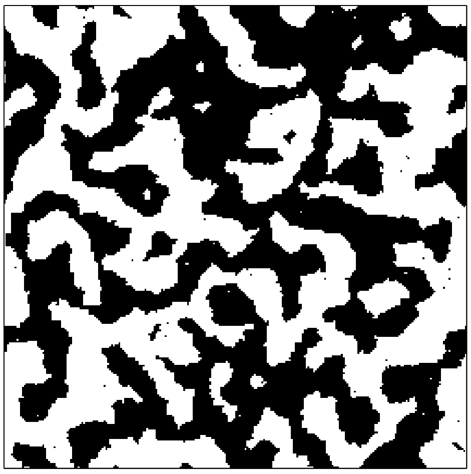


FIGURE 9. An Ising state for $\beta = 2\beta_c$ after 10 sweeps with start in a uniform distributed state

We see that we have "white" and "black" areas with relative sharp bounds. These areas are called *Weiss domains.* Such an Ising state can keep this structure for a long time and in some cases the Weiss domains never disappears. Figure 10 shows such a "metastable" state from which we never converge to equilibrium. This state has also after 100000 steps still an almost zero magnetization.

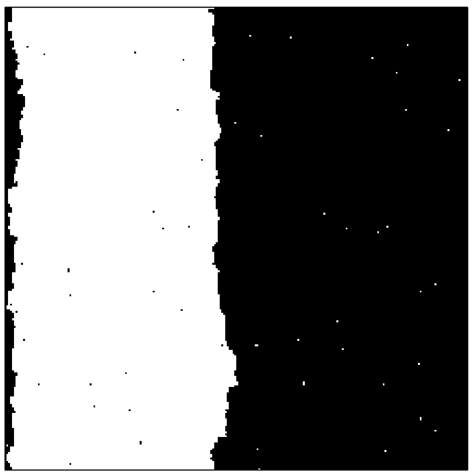


FIGURE 10. An Ising state for $\beta = 2\beta_c$ after 100000 sweeps with start in a uniform distributed state

Finally in Figure 11 we show a Ising state at the critical value β_c after 10000 steps. This looks to be close to the stationary distribution, as we will see later.

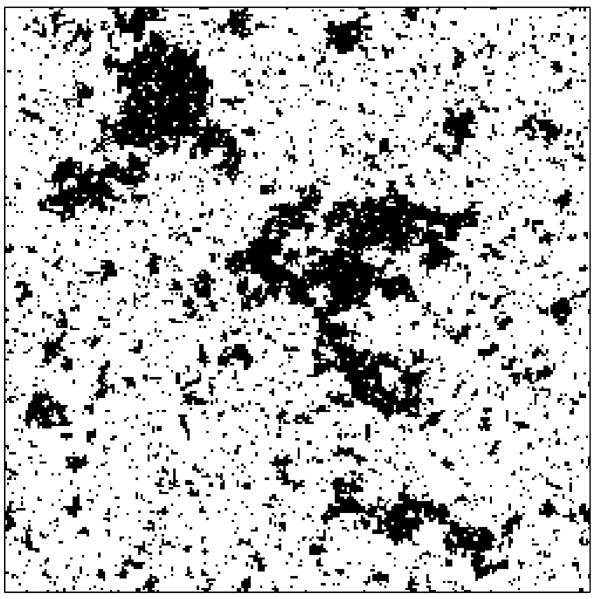


FIGURE 11. An Ising state at β_c after 100000 sweeps

3.3. **Single-bond algorithm.** The *single-bond algorithm* is the heat bath algorithm on the random cluster model. Hence it is also a local algorithm, that adds or removes only a single edge per step.
To do one step of the chain we first choose one of the edges uniformly at random (or deterministically) and add or remove this edge according to the conditional probabilities

$$\mu_p\big(\widetilde{\omega} = \omega \cup \{e\} \mid \widetilde{\omega} \ominus \omega \subseteq \{e\}\big)$$

and

$$\mu_p\big(\widetilde{\omega} = \omega \setminus \{e\} \mid \widetilde{\omega} \ominus \omega \subseteq \{e\}\big),$$

where $\ominus$ is the symmetric difference.
It is easy to calculate that these probabilities are p or $\frac{p}{2-p}$ depending on whether the number of connected components changes or not. So together with the random variable $U \sim \text{Uniform}[0,1]$ we get the update rule for the case of deterministic choice of an edge e

(6)
$$\phi^{\text{sb,e}}(\omega, U) \;=\; \begin{cases} \omega \cup \{e\}, & \text{if } U < \frac{p}{2-p} \\ \omega \cup \{e\}, & \text{if } U < p \;\text{ and } C(\omega \setminus \{e\}) = C(\omega \cup \{e\}) \\ \omega \setminus \{e\}, & \text{otherwise.} \end{cases}$$

As we see on this update rule, we now have another problem during the simulation of such a Markov chain. In each step of the chain we have to check twice, whether the two endpoints of the chosen edge are in the same connected component, i.e., whether the endpoints are connected by edges in the clusters ω and $\omega \setminus \{e\}$. This problem is very complex and computational expensive. In my Matlab code this step takes circa 95% of the computing time.
In our case it is enough to check whether the number of connected components changes, if we add or remove the chosen edge. This is a problem that can be implemented that way that it has a much smaller computing time than the general algorithm, that determines whether two points are connected by edges in a cluster or not.
This problem and possible solutions will be presented later.

Again we want to describe the full algorithm.

Single-bond algorithm

```
Input    Initial cluster $\omega \in \Omega$ and $t_{\max} \in \mathbb{N}$

Step 1:  Set $t = 1$.

Step 2:  Choose $e \in E$ uniformly at random and generate $U_t \sim$
         Uniform$[0, 1]$.

Step 3:  Set $\omega = \omega \setminus \{e\}$. (To decrease the number of "'Else"'-
         calls)

Step 4:  If $U_t < \frac{p}{2-p}$:
             Set $\omega = \omega \cup \{e\}$.
         Else: If $U_t < p$ and $C(\omega) = C(\omega \cup \{e\})$:
             Set $\omega = \omega \cup \{e\}$.

Step 5:  If $t < t_{\max}$:
             Set $t = t + 1$ and go to Step 2.
```

The reversibility of this chain with respect to μ_p can be proven exactly like for the heat bath algorithm.

It is not obvious at the moment, how fast this chain converges to its stationary distribution μ_p depending on p, but in the article [PW] from Propp and Wilson they use this chain for their coupling from the past algorithm to generate an exact sample from an Ising state according to the Gibbs distribution at the critical temperature. Therefore one may hope, that this chain is not so much affected by the critical slowing down like the heat bath algorithm. In their algorithm they also use sweeps of the single-bond algorithm, i.e., the connection of $|E|$ single-bond steps for each edge in E. So we (probably) lost the reversibility of the chain, but we keep the stationarity with respect to the random-cluster distribution μ_p.

Now we show some states generated by the sampling method above. Again we restrict ourself to the sweep algorithm and we start in the empty set. All presented states are generated in 100 sweeps.
As we can see by the definition of μ_p, the value of β is a indicator for the average size of the connected components. So we expect that in the high temperature case (low β) we have much less edges than in the low temperature case and so much more small components.
First Figure 12 we show a random cluster state at high temperature together with two Ising states that could result out of this state.

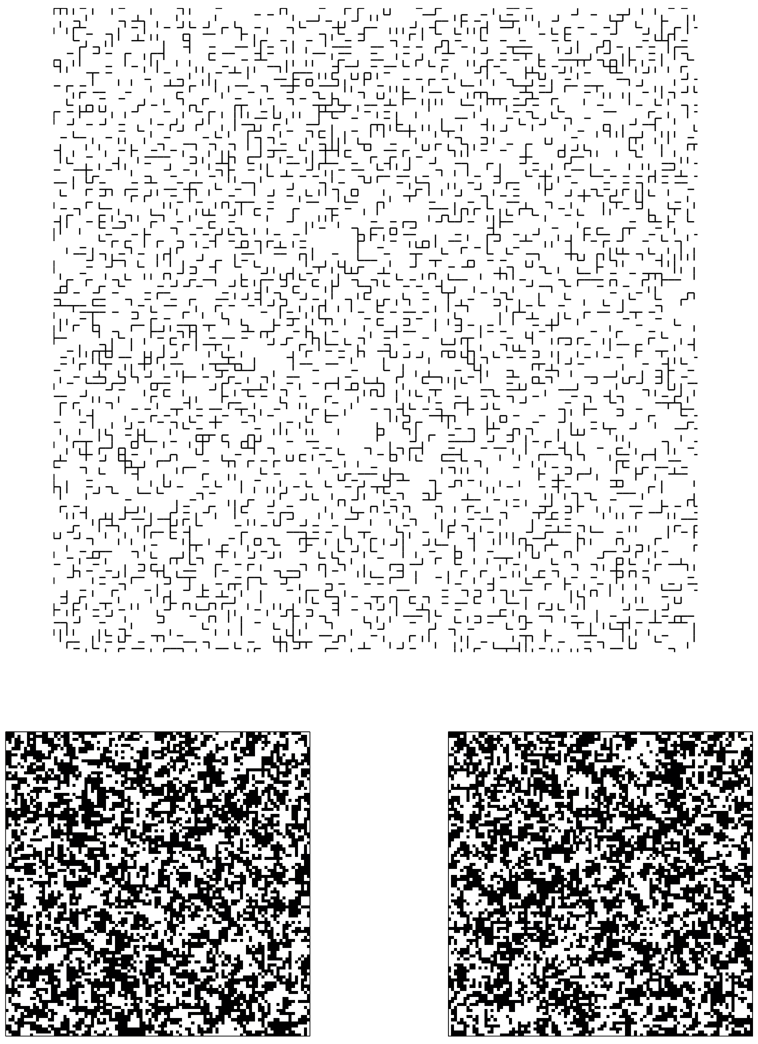


FIGURE 12. An random cluster state for $\beta = \frac{1}{2}\beta_c$ after 100 sweeps

In contrast the next Figure 13 shows the low temperature case. We see that in this state we have much more edges and so the resulting Ising states have the same spin on nearly all vertices. But we can see that one random cluster state can produce both states, the almost "all spins up" and the almost "all spins down" state, while the algorithms on the Ising model could not.

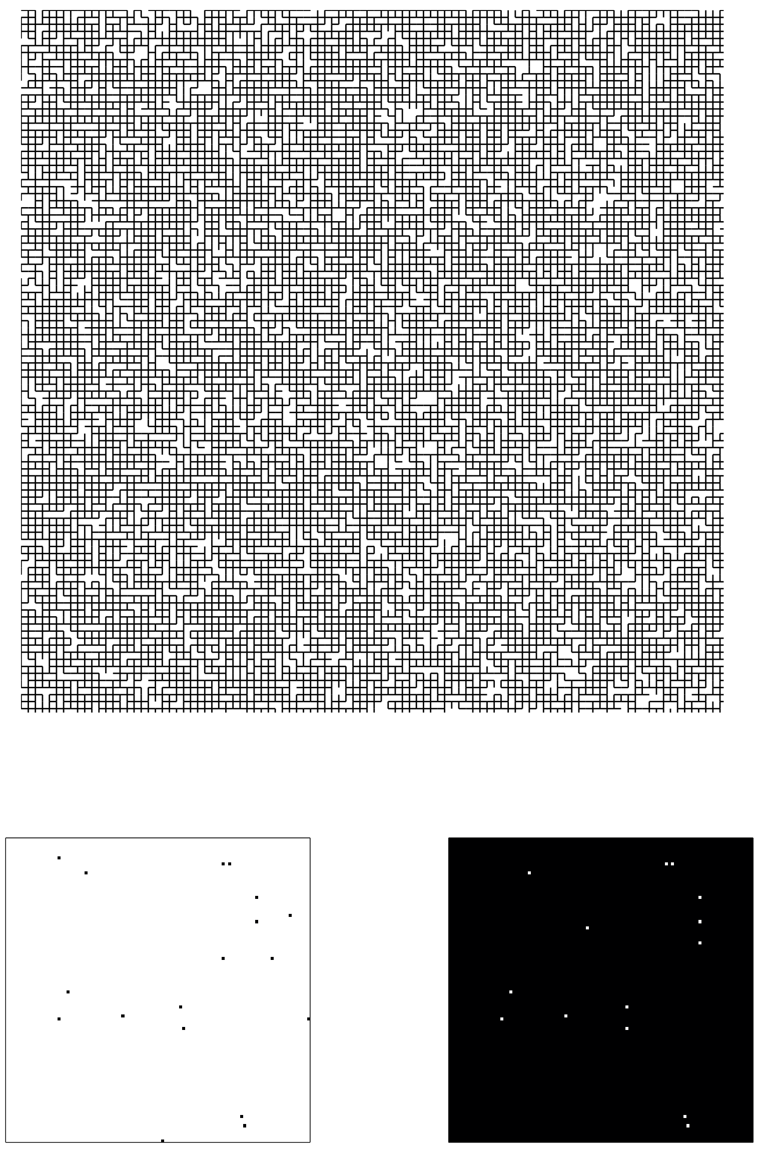


FIGURE 13. An random cluster state for $\beta = 2\beta_c$ after 100 sweeps

The last picture of this section in Figure 14 shows a random cluster state at the critical value β_c. At first sight we see hardly any differences to Figure 13, but if we look at the resulting Ising states we see that the results are similar to the Ising states at the critical temperature from Figure 11.

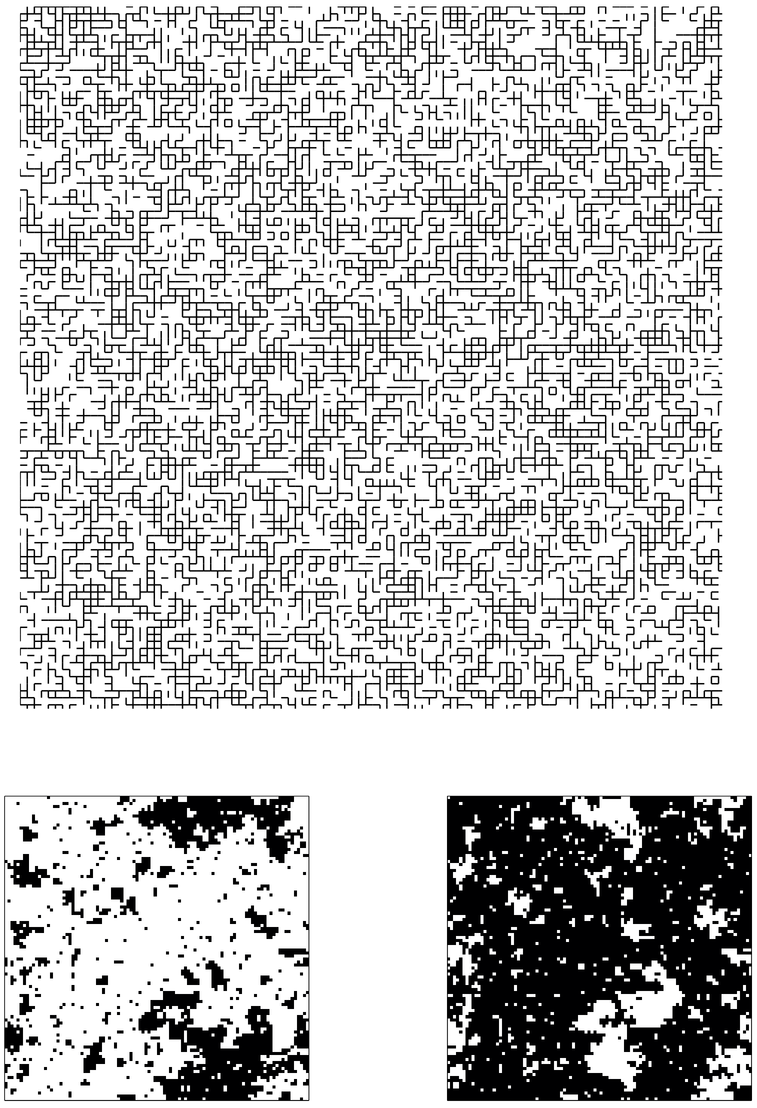


FIGURE 14. A random cluster state at β_c after 100 sweeps

3.4. **Swendsen-Wang algorithm.** The *Swendsen-Wang algorithm* can be defined on both state spaces equivalently. Therefore we use the functions f from (1) and g from (2) alternating to define the update rules for the Swendsen-Wang algorithm on the Ising and random cluster model, respectively.
Let us recall that the function f generates a spin configuration according to the Gibbs distribution, if the given random cluster state is distributed with respect to μ_p, where $p = 1 - e^{-\beta J}$, and vice versa for the function g.
Now for $\sigma \in \Sigma$ and $\omega \in \Omega$ we take the update rules

$$\phi^{\mathrm{sw}}(\sigma, U) \;=\; f\big(g(\sigma, U_1), U_2\big) \;\in \Sigma$$

and

$$\phi^{\mathrm{sw2}}(\omega, U) \;=\; g\big(f(\omega, U_2), U_1\big) \;\in \Omega,$$

where we have $U = (U_1, U_2)$ as well as $U_1 \sim \big(\mathrm{Bin}(1,p)\big)^E$ and $U_2 \sim \mathrm{Uniform}\{0,1\}^V$.

Now we show that the Markov chains defined by this update rules have the desired stationary distribution and so the algorithm works correctly.

Theorem 7. *Let $X_t = \phi_t^{sw1}(X_{t-1}) \in \Sigma$ and $Y_t = \phi_t^{sw2}(Y_{t-1}) \in \Omega$ with random variables U_t defined as U above. Then the Markov chains $X = (X_t)_{t\in\mathbb{N}}$ and $Y = (Y_t)_{t\in\mathbb{N}}$ are irreducible, aperiodic and reversible with respect to π_β or μ_p, respectively.*

Proof. First let us prove the irreducibility and the aperiodicity. By the construction of f from (1) it is obvious that $f(\varnothing, u)$ can reach every $\sigma \in \Sigma$ for a certain value of u. Moreover $g(\sigma, \widetilde{u})$ can produce the empty set $\varnothing$ for each input σ, e.g. for $\widetilde{u} = (0, \ldots, 0)$. Therefore every $\sigma \in \Sigma$ is reachable from each state in one step of the chain X and so X is irreducible and aperiodic. It is easy to see, that this also holds for the chain Y. Especially we get that matrices of the transition probabilities of both chains are strictly positive.

Now let us prove the reversibility with respect to π_β. For $U = (U_1, U_2)$ with $U_1 \sim \big(\mathrm{Bin}(1,p)\big)^E$ and $U_2 \sim \mathrm{Uniform}\{0,1\}^V$ we get for each $\sigma_1, \sigma_2 \in \Sigma$

$$\begin{aligned}\pi_\beta(\sigma_1)\, \mathbb{P}\big(\phi^{\mathrm{sw1}}(\sigma_1, U) = \sigma_2\big) \;&=\; \pi_\beta(\sigma_1)\, \mathbb{P}\big(f(g(\sigma_1, U_1), U_2) = \sigma_2\big)\\ &=\; \pi_\beta(\sigma_1) \sum_{\omega\in\Omega} \mathbb{P}\big(f(\omega, U_2) = \sigma_2,\; g(\sigma_1, U_1) = \omega\big)\\ &=\; \pi_\beta(\sigma_1) \sum_{\omega\in\Omega} \mathbb{P}\big(f(\omega, U_2) = \sigma_2\big)\, \mathbb{P}\big(g(\sigma_1, U_1) = \omega\big).\end{aligned}$$

Just like in the proofs of the Theorems 4 and 5 we get

$$\begin{aligned}&\pi_\beta(\sigma_1)\, \mathbb{P}\big(\phi^{\mathrm{sw1}}(\sigma_1, U) = \sigma_2\big)\\ &\quad=\; \pi_\beta(\sigma_1) \sum_{\omega\in\Omega} \frac{1}{2^{C(\omega)}}\, p^{|\omega|}\, (1-p)^{|E|-|\omega|-C^*(\sigma_1)}.\\ &\qquad\qquad \cdot \mathbb{1}\,(\exists u : f(\omega, u) = \sigma_2)\; \mathbb{1}\,(\exists \widetilde{u} : g(\sigma_1, \widetilde{u}) = \omega)\\ &\quad=\; \frac{1}{Z}\, e^{-\beta H(\sigma_1)} \sum_{\omega\in\Omega} \frac{1}{2^{C(\omega)}}\, p^{|\omega|}\, (1-p)^{|E|-|\omega|}\, e^{+\beta J C^*(\sigma_1)}.\\ &\qquad\qquad \cdot \mathbb{1}\,(\exists u : f(\omega, u) = \sigma_2)\; \mathbb{1}\,(\exists \widetilde{u} : g(\sigma_1, \widetilde{u}) = \omega)\\ &\quad=\; \frac{1}{Z} \sum_{\omega\in\Omega} \frac{1}{2^{C(\omega)}}\, p^{|\omega|}\, (1-p)^{|E|-|\omega|}.\\ &\qquad\qquad \cdot \mathbb{1}\,(\exists u : f(\omega, u) = \sigma_2)\; \mathbb{1}\,(\exists \widetilde{u} : g(\sigma_1, \widetilde{u}) = \omega)\\ &\quad=\; \frac{1}{Z}\, e^{-\beta H(\sigma_2)} \sum_{\omega\in\Omega} \frac{1}{2^{C(\omega)}}\, p^{|\omega|}\, (1-p)^{|E|-|\omega|}\, e^{+\beta J C^*(\sigma_2)}.\\ &\qquad\qquad \cdot \mathbb{1}\,(\exists \widetilde{u} : g(\sigma_2, \widetilde{u}) = \omega)\; \mathbb{1}\,(\exists u : f(\omega, u) = \sigma_1)\\ &\quad=\; \pi_\beta(\sigma_2)\, \mathbb{P}\big(\phi^{\mathrm{sw1}}(\sigma_2, U) = \sigma_1\big).\end{aligned}$$

Here the next to last equality holds because

$$\begin{aligned}\mathbb{1}\big(\exists u : f(\omega, u) = \sigma\big) \;&=\; \mathbb{1}\,\big(\omega \subseteq \big\{\{x,y\} \in E : \sigma_x = \sigma_y\big\}\big)\\ &=\; \mathbb{1}\big(\exists \widetilde{u} : g(\sigma, \widetilde{u}) = \omega\big)\end{aligned}$$

and the last equality comes by doing the same steps backwards. Therefore we have reversibility of the Markov chain X.

For the chain Y we get with the same techniques for each $\omega_1, \omega_2 \in \Omega$

$$\begin{aligned}
&\mu_p(\omega_1)\, \mathbb{P}\big(\phi^{\mathrm{sw2}}(\omega_1, U) = \omega_2\big) \\
&\quad= \mu_p(\omega_1) \sum_{\sigma\in\sigma} \mathbb{P}\big(g(\sigma, U_1) = \omega_2\big)\, \mathbb{P}\big(f(\omega_1, U_2) = \sigma\big) \\
&\quad= \mu_p(\omega_1) \sum_{\sigma\in\sigma} p^{|\omega_2|}\, (1-p)^{|E|-|\omega_2|-C^*(\sigma)}\, \frac{1}{2^{C(\omega_1)}} \cdot \\
&\qquad\quad \cdot\ \mathbb{1}\,(\exists \widetilde{u} : g(\sigma, \widetilde{u}) = \omega_2)\ \mathbb{1}\,(\exists u : f(\omega_1, u) = \sigma) \\
&\quad= \frac{1}{Z}\, p^{|\omega_1|}\, (1-p)^{|E|-|\omega_1|} \sum_{\sigma\in\sigma} p^{|\omega_2|}\, (1-p)^{|E|-|\omega_2|-C^*(\sigma)} \cdot \\
&\qquad\quad \cdot\ \mathbb{1}\,(\exists \widetilde{u} : g(\sigma, \widetilde{u}) = \omega_2)\ \mathbb{1}\,(\exists u : f(\omega_1, u) = \sigma) \\
&\quad= \frac{1}{Z}\, p^{|\omega_2|}\, (1-p)^{|E|-|\omega_2|} \cdot \\
&\qquad\quad \cdot \sum_{\sigma\in\sigma} p^{|\omega_1|}\, (1-p)^{|E|-|\omega_1|-C^*(\sigma)}\ 2^{C(\omega_2)}\, \frac{1}{C(\omega_2)} \cdot \\
&\qquad\quad \cdot\ \mathbb{1}\,(\exists u : f(\omega_2, u) = \sigma)\ \mathbb{1}\,(\exists \widetilde{u} : g(\sigma, \widetilde{u}) = \omega_1) \\
&\quad= \mu_p(\omega_2)\, \mathbb{P}\big(\phi^{\mathrm{sw2}}(\omega_2, U) = \omega_1\big).
\end{aligned}$$

This proves the claim. □

The Swendsen-Wang algorithm generates a Markov chain that can jump from every state to any other in one step and this in both state spaces, the Ising and the random cluster model. Therefore one may hope, that this chain is rapidly mixing, hopefully for all values of β.

Another advantage of the Swendsen-Wang algorithm is that it works very fast compared to other algorithms on the random cluster model. While 100 sweeps of the single-bond algorithm on a 100×100 lattice takes circa 100 minutes, the same number of steps with the Swendsen-Wang algorithm takes only 10 seconds.
However, we have again a problem with large lattices. The reason is the absence of a proper data structure (in Matlab).

3.5. **Wolff's algorithm.** The algorithm of Wolff works almost like the Swendsen-Wang algorithm on the Ising model. The difference between these two algorithms is that in the Swendsen-Wang algorithm every connected component of the generated cluster could change the spin, while in *Wolff's algorithm* only one connected component will be changed surely. For a detailed introduction, see e.g. [W].
In summary one step works that way that we first choose a vertex v uniformly at random and then build a connected component starting from this vertex. We add every neighbor with the same spin value with probability $p = 1 - e^{-J\beta}$. Next we go to each vertex that we have connected with v and make the same step. We repeat this until no connected vertex remains.
Finally we change the spin of all connected components with probability 1.

Let us describe the algorithm in detail. Recall that $p = 1 - e^{-\beta J}$

Wolff's algorithm step

```
Input    Initial spin configuration $\sigma \in \Sigma$

Step 1:  Choose $v^* \in V$ uniformly at random and set $A = \{v^*\}$
         and $C = \emptyset$.

Step 2:  Repeat
           For all $v \in A \setminus C$:
             Set $C = C \cup \{v\}$.
             For all $w \in V \setminus C$ with $w \sim v$ and $\sigma_w = \sigma_v$:
               Generate $U \sim \text{Uniform}[0,1]$.
               If $U < p$:
                 Set $A = A \cup \{w\}$.
         until $A = C$

Step 3:  For all $v \in A$:
           Set $\sigma_v = -\sigma_v$.
```

Note that in one step of Wolff's algorithm from a spin configuration σ, we can reach only such spin configurations $\widetilde{\sigma}$ that differs only by one connected component in E. However, this connected component can be very large.

Next we prove the reversibility of Wolff's algorithm.

Theorem 8. *Wolff's algorithm is irreducible, aperiodic and reversible with respect to the distribution π_β.*

Proof. The first two statements are easy to see. As we mentioned before, the probability $W(\sigma, \widetilde{\sigma})$ to go in one step with Wolff's algorithm from a spin configuration σ to $\widetilde{\sigma}$ is nonzero if and only if the two configurations differs only by one connected component in E, say A. I.e. $A = \{v \in V : \sigma_v \neq \widetilde{\sigma}_v\}$ is a connected component in E. We define the boundary of A by

$$\delta A := \big\{\{x,y\} \in E : \ x \in A, \ y \notin A\big\}.$$

Note that $\delta A \subseteq E$. Now let σ and $\widetilde{\sigma}$ a pair of spin configurations that are reachable in one step of Wolff's algorithm. Then the probabilities $W(\sigma, \widetilde{\sigma})$ and $W(\widetilde{\sigma}, \sigma)$ differ only by a factor that depends on the values of the initial spin configuration on the endpoints of edges in δA. If the spins are equal on such endpoints we have the probability $1 - p = e^{-\beta J}$ that we do not add the second endpoint to A, otherwise this probability is 1. Therefore we get

$$\begin{aligned}
\frac{W(\sigma,\widetilde{\sigma})}{W(\widetilde{\sigma},\sigma)} &= \prod_{\{x,y\}\in\delta A} \frac{1 - p \cdot \mathbb{1}(\sigma_x = \sigma_y)}{1 - p \cdot \mathbb{1}(\widetilde{\sigma}_x = \widetilde{\sigma}_y)} = \prod_{\{x,y\}\in\delta A} \frac{e^{-\beta J \cdot \mathbb{1}(\sigma_x = \sigma_y)}}{e^{-\beta J \cdot \mathbb{1}(\widetilde{\sigma}_x = \widetilde{\sigma}_y)}} \\
&= \exp\Big(-\beta J \sum_{\{x,y\}\in\delta A} \big[\mathbb{1}(\sigma_x = \sigma_y) - \mathbb{1}(\widetilde{\sigma}_x = \widetilde{\sigma}_y)\big]\Big) \\
&= \exp\Big(-\beta J \sum_{\{x,y\}\in E} \big[\mathbb{1}(\sigma_x = \sigma_y) - \mathbb{1}(\widetilde{\sigma}_x = \widetilde{\sigma}_y)\big]\Big) \\
&= \exp\Big(-\beta J \sum_{\{x,y\}\in E} \big[\mathbb{1}(\widetilde{\sigma}_x \neq \widetilde{\sigma}_y) - \mathbb{1}(\sigma_x = \sigma_y)\big]\Big) \\
&= \frac{\pi_\beta(\widetilde{\sigma})}{\pi_\beta(\sigma)}.
\end{aligned}$$

This proves the claim. □

3.6. **Propp-Wilson algorithm.** The *Propp-Wilson algorithm*, also called *Coupling From The Past*, is not an algorithm like the others above. This algorithm does not generate a Markov chain, but it uses one of these to generate a sample according *exactly* to the stationary distribution of the chain.
This algorithm has two major advantages. The first is that the output is exactly distributed according to the the stationary distribution of the underlying Markov chain and the second is that the algorithm determines automatically when to stop, so we can do our simulation without calculating any Markov chain convergence rates before.
However it is very interesting to know how long the algorithm needs to stop.

First we explain the basic idea of the Propp-Wilson algorithm and then we define the algorithm in general. Next we show that if we have a special structure of the state space and the update rule we can reduce the computing time extremely.

Suppose we have available an update rule $\phi(\cdot,\cdot)$ with stationary distribution π on the state space S and a sequence of uniform distributed random numbers $U_{-1}, U_{-2}, \dots$ (We will see in a moment, why we count into the past). Recall that we use the notation $\phi_t(\cdot) := \phi(\cdot, U_t)$, $\forall t < 0$.
Now we define the *run from* t_1 *to* t_2 *with start in* i as the composition

$$F_{t_1}^{t_2}(i) \;=\; \phi_{t_2-1} \circ \phi_{t_2-2} \circ \cdots \circ \phi_{t_1}(i). \tag{7}$$

We look at the run F_{-t}^0, $t > 0$, and want to find the smallest t^* that makes $F_{-t^*}^0$ a constant mapping (for the given random numbers), i.e. $F_{-t^*}^0(i) = F_{-t^*}^0(j)$ for all $i, j \in S$. It is easy to see that if $F_{-t^*}^0$ is constant, all F_{-s}^0 with $s > t^*$ are constant, too.
We will see that such a constant mapping guarantees that the distribution of the output is exactly the stationary one. But first we have to demonstrate how the algorithm works.
We start some Markov chains (defined by the update rule) in each state of the state space S at a time in the past and run to time 0. If all Markov chains are coalesce, i.e., end up in the same state, we stop the algorithm. Otherwise we go further into the past and start again.

Let S be any state space. The algorithm works as follows.

Propp-Wilson algorithm

```
Input    An update function $\phi$ for an ergodic Markov chain with
         stationary distribution $\pi$ and an increasing sequence of
         positive integers $(N_0 = 1, N_1, N_2, \dots)$.

Step 1:  Set $m = 1$.

Step 2:  Generate random numbers $U_{-N_m+1}, \dots, U_{-N_{m-1}}$ with
         $U_i \sim \text{Uniform}[0,1]$.

Step 3:  For each $i \in S$:
           Simulate the Markov chain starting at time $-N_m$ in
         state $i$, and running up to 0 using update function $\phi$
         and random numbers $U_{-N_m+1}, U_{-N_m+2}, \dots, U_{-1}$ (these
         are the same for each of the $|S|$ chains).

Step 4:  If all chains end up in the same state $j$:
           Output $j$ and stop.
         Else: Continue with Step 5.

Step 5:  Set $m = m + 1$ and go to Step 2.
```

The first question is, whether this algorithm ever stops or not. Since we have an ergodic Markov chain, this depends only on the choice of the update rule. To see this, look at the following simple example and the Lemma after it.

Example 9. Suppose the state space $S = \{1, 2\}$ and the transition matrix

$$q = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}.$$

One possible update rule is

$$\phi(i,U) \;=\; \begin{cases} 1, & \text{if } U < \frac{1}{2} \\ 2, & \text{if } U \geq \frac{1}{2} \end{cases}$$

for $i = 1, 2$. It is easy to see that if we use this update rule the algorithm terminates after one step.
Another update rule that fulfills the transition probabilities above is

$$\phi^*(1,U) \;=\; \begin{cases} 1, & \text{if } U < \frac{1}{2} \\ 2, & \text{if } U \geq \frac{1}{2} \end{cases}$$

and

$$\phi^*(2,U) \;=\; \begin{cases} 2, & \text{if } U < \frac{1}{2} \\ 1, & \text{if } U \geq \frac{1}{2}. \end{cases}$$

In each step both chains either transpose or keep their values. So with this update rule the algorithm never terminates.

□

In the following we assume that we have chosen an update rule so that it is possible that all chains coalesce. Then the algorithm stops with probability 1.

Lemma 10. *Suppose that it is possible that all $|S|$ ergodic Markov chains with update rule ϕ coalesce. Then the Propp-Wilson algorithm terminates with probability 1.*

Proof. From the assumption we get that there is an $L \in \mathbb{N}$ such that $\mathbb{P}\left(F^0_{-L}(\cdot) \text{ is constant}\right) > 0$. Hence for each t, F^{-t}_{-t-L} has a positive probability of being constant. Since each of the maps $F^0_{-L}, F^{-L}_{-2L}, \ldots$ has some positive probability $\varepsilon > 0$ of being constant and all these events are independent, it will happen with probability 1 that one of these maps is constant. In this case F^0_{-M} is constant for sufficiently large M.

□

Another important fact is that we have to go from the past to 0 and not from 0 to the future, because it turns out that this gives biased samples in general. Therefore see the following example.

Example 11. Again suppose the state space $S = \{1, 2\}$, but now the transition matrix

$$q = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ 1 & 0 \end{pmatrix}.$$

Clearly, the chain is reversible with stationary distribution

$$\pi = (\pi_1, \pi_2) = \left(\frac{2}{3}, \frac{1}{3}\right).$$

Suppose the two copies of the chain that run from time 0 to the future coalesce for the first time at some random time T. By the definition of T the two chains cannot be in the same state at time $T-1$. Hence one of the chains is in state 2 at time $T-1$. But the transition matrix tells us that this chain will with probability 1 be in state 1 in time T. So this algorithm outputs the state 1 with probability 1. But this is not the stationary distribution of the chain.

□

Also it is necessary in the algorithm to reuse the random numbers $U_{-N_m+1}, U_{-N_m+2}, \ldots, U_0$ when restarting the chains and again come to time $-N_m + 1$, because otherwise we get also biased samples. The following example shows the problem.

Example 12. Again look at the situation of Example 11 with the update rule

$$\phi^{**}(i, U) = \begin{cases} 1, & \text{if } U < q_{i1} \\ 2, & \text{if } U \geq q_{i1}. \end{cases}$$

We take the Propp-Wilson algorithm for this chain with starting points $(N_1, N_2, ...) = (1, 2, 4, 8, ...)$, but modify it in that way that we use new random numbers in each round. Let Y denote the output of this modified algorithm, and define M as the largest integer m for which the algorithm decides to start the chains at time $-N_m$. Furthermore let $X_t^{(m)}, t = -N_m, \ldots, 0$ denote the coupling used in the modified Propp-Wilson algorithm. Hence we have $X_{-N_m}^{(m)} = (1,2)$. Then we get

$$\begin{aligned}
\mathbb{P}(Y=1) &= \sum_{m=1}^{\infty} \mathbb{P}(M=m,\, Y=1) \\
&\geq \mathbb{P}(M=1,\, Y=1) \;+\; \mathbb{P}(M=2,\, Y=1) \\
&= \mathbb{P}\left(X_0^{(1)} = (1,1)\right) \;+\; \mathbb{P}\left(X_0^{(1)} \in \{(1,2),(2,1)\},\; X_0^{(2)} = (1,1)\right) \\
&= \frac{1}{2} \;+\; \frac{1}{2} \cdot \mathbb{P}\Big(\big(X_{-1}^{(2)}, X_0^{(2)}\big) \in \big\{\big((1,1),(1,1)\big),\; \big((2,1),(1,1)\big)\big\}\Big) \\
&= \frac{1}{2} \;+\; \frac{1}{2} \cdot \frac{1}{2} \\
&= \frac{3}{4} \;>\; \frac{2}{3}
\end{aligned}$$

The fourth line occurs because the $X^{(m)}$ are independent if we take new random numbers in each round. Hence, the distribution of the output Y does not agree with the distribution π from Example 11 and so this algorithm is incorrect.

□

After we have discussed the details of the algorithm we want to prove that it works correctly. For an alternative prove see e.g. Propp and Wilson [PW].

The intuition behind this is that if we take a random variable X with distribution π, then for all $t \geq 0$ the random variable $Y_t = F_{-t}^0(X)$ has the same distribution π. But if we take t large enough, so that $F_{-t}^0(\cdot)$ is a constant mapping, we cannot distinguish whether we start in X or in any other state. Hence the output of the algorithm is distributed according to π.

Theorem 13. *Let ϕ be an update rule for an ergodic Markov chain with state space $S = \{1, \ldots, K\}$ and stationary distribution π. Furthermore let $N_1, N_1, \ldots$ be an increasing sequence of positive integers.*
Suppose that the Propp-Wilson algorithm terminates with probability 1, and write Y for its output. Then we have

$$\mathbb{P}(Y = i) \;=\; \pi(i)\,, \qquad \forall i \in S.$$

Proof. By the assumption that the algorithm terminates with probability 1, we can pick a sufficiently large $M \in \mathbb{N}$ such that for any $\varepsilon > 0$

$$\mathbb{P}\left(F^0_{-N_M}(\cdot) \text{ is constant}\right) \;\geq\; 1 - \varepsilon. \tag{8}$$

Suppose we have a random variable $X \sim \pi$ and so $\widetilde{Y} = F^0_{-N_M}(X)$ is the random variable that we get after N_M steps with the update rule ϕ and the same random numbers $U_{-N_M}, \ldots, U_{-1}$ as in the algorithm, but with start in the stationary distribution. Therefore $\widetilde{Y} \sim \pi$. We know that $Y = \widetilde{Y}$ holds if the event in (8) happens, so that

$$\mathbb{P}(Y \neq \widetilde{Y}) \;\leq\; \varepsilon.$$

We get for all $i \in S$

$$\begin{aligned}\mathbb{P}(Y = i) - \pi(i) \;&=\; \mathbb{P}(Y = i) - \mathbb{P}(\widetilde{Y} = i)\\ &\leq\; \mathbb{P}(Y = i,\, \widetilde{Y} \neq i) \;\leq\; \mathbb{P}(Y \neq \widetilde{Y})\\ &\leq\; \varepsilon\end{aligned}$$

and

$$\begin{aligned}\pi(i) - \mathbb{P}(Y = i) \;&=\; \mathbb{P}(\widetilde{Y} = i) - \mathbb{P}(Y = i)\\ &\leq\; \mathbb{P}(\widetilde{Y} = i,\, Y \neq i) \;\leq\; \mathbb{P}(Y \neq \widetilde{Y})\\ &\leq\; \varepsilon.\end{aligned}$$

Hence, we have

$$|\mathbb{P}(Y = i) - \pi(i)| \;\leq\; \varepsilon\;, \quad \forall \varepsilon > 0, \quad \forall i \in S$$

and so

$$\mathbb{P}(Y = i) \;=\; \pi(i)\;, \quad \forall i \in S.$$

□

Now we know that the algorithm works, but there is another problem one can object:
Suppose the state space S is very large. How can we then run Markov chains from all possible states? This would make the algorithm not very useful.

The answer is that there are techniques to reduce the number of chains that have to simulate to decide whether all chains are coupled or not. One of the most important techniques uses some kind of monotonicity of the state space and the used Markov chain, this is the so-called *sandwiching.*

3.6.1. *Sandwiching.* Suppose that the state space S admits a natural partial ordering $\preceq$, and that the update rule ϕ of our Markov chain has the property

$$i \preceq j \implies \phi(i,U) \preceq \phi(j,U) \tag{9}$$

almost surely with respect to U. In addition we assume that there exist the elements $\hat{0}$ and $\hat{1}$, so that $\hat{0} \preceq i \preceq \hat{1}$ for all $i \in S$.
Like above we assume that the sequence $U_{-1}, U_{-2}, \dots$ of random numbers is given. Then the monotonicity property ensures that if we have a number t^* such that $F^0_{-t^*}(\hat{0}) = F^0_{-t^*}(\hat{1})$, we get that $F^0_{-t^*}(\cdot)$ is a constant mapping, i.e., $F^0_{-t^*}(i) = F^0_{-t^*}(j)$ for all $i, j \in S$.
Therefore it is enough to consider only two chains instead of the $|S|$ possible ones and so the algorithm become applicable and simplifies to the following.

Propp-Wilson algorithm (sandwiching)

```
Input    An update function $\phi$ with property (9) for an ergodic
         Markov chain with stationary distribution $\pi$ and an in-
         creasing sequence of positive integers ($N_0 = 1, N_1, \dots$).
         Additionally the top and bottom element $\hat{0}$ and $\hat{1}$.

Step 1:  Set $m = 1$.

Step 2:  For $t = -N_m + 1, \dots, -N_{m-1}$: Generate random num-
         bers $U_t$ with $U_t \sim$ Uniform$[0,1]$ and set $\phi_t(\cdot) := \phi(\cdot, U_t)$.

Step 3:  If $F^0_{-N_m}(\hat{0}) = F^0_{-N_m}(\hat{1})$:
           Output $F^0_{-N_m}(\hat{0})$ and stop.
         Else: Set $m = m + 1$ and go to Step 2.
```

Now we want to apply the Propp-Wilson algorithm to the Ising model.

3.6.2. *Ising model.* We show that we can use the sandwiching method for the heat bath algorithm to reduce the computing time extremely. Therefore we define the partial ordering for $\sigma, \tau \in \Sigma := \{-1,1\}^V$ as

$$\sigma \preceq \tau \;:\Longleftrightarrow\; \sigma(v) \leq \tau(v)\,, \quad \forall v \in V.$$

So our top and bottom elements are $\hat{0} = (-1)^V$ and $\hat{1} = (1)^V$.
Recall that σ^{v+} is the Ising configuration with $\sigma^{v+}(v) = 1$ and $\sigma^{v+}(w) = \sigma(w)$ for all $w \neq v$ (equivalently for σ^{v-}).
Then it is easy to see (like in Section 3.1) that for $\sigma, \tau \in \Sigma$ with $\sigma \preceq \tau$ and for all $A \subseteq V$ as well as each vertex v

$$\sum_{w \in A} \mathbb{1}(\sigma(w) \neq 1) \;\geq\; \sum_{w \in A} \mathbb{1}(\tau(w) \neq 1).$$

Hence, we get

$$\begin{aligned}\frac{\pi_\beta(\sigma^{v-})}{\pi_\beta(\sigma^{v+})} &= \exp\left[2\beta \sum_{w\colon \{v,w\} \in E} \left(\mathbb{1}(\sigma(w) \neq 1) - \frac{1}{2}\right)\right] \\ &\geq \exp\left[2\beta \sum_{w\colon \{v,w\} \in E} \left(\mathbb{1}(\tau(w) \neq 1) - \frac{1}{2}\right)\right] \;=\; \frac{\pi_\beta(\tau^{v-})}{\pi_\beta(\tau^{v+})}\end{aligned}$$

and so

$$\frac{\pi_\beta(\sigma^{v+})}{\pi_\beta(\sigma^{v+}) + \pi_\beta(\sigma^{v-})} \;\leq\; \frac{\pi_\beta(\tau^{v+})}{\pi_\beta(\tau^{v+}) + \pi_\beta(\tau^{v-})}.$$

Now, if we look at the update rule of the heat bath algorithm from (3), we see that for $\sigma \preceq \tau$

- $U < \dfrac{\pi_\beta(\sigma^{v+})}{\pi_\beta(\sigma^{v+}) + \pi_\beta(\sigma^{v-})} \implies \phi^{\text{hb},v}(\sigma, U) = \sigma^{v+}$
  $\implies \phi^{\text{hb},v}(\tau, U) = \tau^{v+}$

- $U \geq \dfrac{\pi_\beta(\tau^{v+})}{\pi_\beta(\tau^{v+}) + \pi_\beta(\tau^{v-})} \implies \phi^{\text{hb},v}(\tau, U) = \tau^{v-}$
  $\implies \phi^{\text{hb},v}(\sigma, U) = \sigma^{v-}$

- otherwise $\implies \phi^{\text{hb},v}(\sigma, U) = \sigma^{v-}$ and $\phi^{\text{hb},v}(\tau, U) = \tau^{v+}$.

Therefore we get

$$\sigma \preceq \tau \quad \Longrightarrow \quad \phi^{\mathrm{hb},v}(\sigma, U) \preceq \phi^{\mathrm{hb},v}(\tau, U) \qquad \text{a.s.}$$

and we can use the sandwiching technique to reduce the computing time. So it is enough to start two Markov chains, one in the "all spnis up" and one in the "all spins down" state, and search for a time in the past such that these two chains coalesce.

Remark 14. The Metropolis algorithm with the update rule from (5) does not have such a monotonicity property, because there is the possibility that both, the upper and the lower chain, change their value on one vertex at the same time. This avoids that this update rule is monotone.

Note that the sweep algorithm of the heat bath algorithm is also monotone, because the composition of monotone update rules is monotone.

In the next subsection we analyze the heat bath algorithm on the random cluster model.

3.6.3. *Random cluster model.* On the the random cluster model with state space $\Omega = \{\omega : \omega \subseteq E\}$ we have the natural partial ordering "$\subseteq$" with the top element $\hat{1} = E$ and bottom element $\hat{0} = \varnothing$.
Now if we look at the number of connected components $C(\cdot)$ we get for each edge e and $\omega, \eta \in \Omega$ with $\omega \subseteq \eta$

$$C(\omega \setminus \{e\}) = C(\omega \cup \{e\}) \quad \Longrightarrow \quad C(\eta \setminus \{e\}) = C(\eta \cup \{e\}).$$

In other words, if two vertices are connected in $\omega \setminus \{e\}$ then they are connected in $\eta \setminus \{e\}$, too.
And so, if we look at the update rule of the single bond algorithm from (6), we have again three cases. Suppose $\omega \subseteq \eta$, then

- $U < \dfrac{p}{2-p} \implies \phi^{\mathrm{sb},e}(\omega,U) = \omega \cup \{e\}$ and $\phi^{\mathrm{sb},e}(\eta,U) = \eta \cup \{e\}$

- $U \ge p \implies \phi^{\mathrm{sb},e}(\omega,U) = \omega \setminus \{e\}$ and $\phi^{\mathrm{sb},e}(\eta,U) = \eta \setminus \{e\}$

- otherwise: $\phi^{\mathrm{sb},e}(\eta,U) = \eta \setminus \{e\} \implies \phi^{\mathrm{sb},e}(\omega,U) = \omega \setminus \{e\}$ and $\phi^{\mathrm{sb},e}(\omega,U) = \omega \cup \{e\} \implies \phi^{\mathrm{sb},e}(\eta,U) = \eta \cup \{e\}$

So we have again the monotonicity property

$$\omega \subseteq \eta \implies \phi^{\mathrm{sb},e}(\omega,U) \subseteq \phi^{\mathrm{sb},e}(\eta,U) \qquad \text{a.s.}$$

for all $\omega, \eta \in \Omega$ and we can apply the sandwiching technique also on the random cluster model.

Like for the heat bath algorithm we can use the sweep algorithm, because it is monotone if all single flip rules are monotone.

Remark 15. Note that for $U < \frac{p}{2-p}$ and for $U \ge p$ the chains coalesce at the chosen edge, so if there would be no difference between them, the chains would be coalesce completely in one step (It happens so for $p = 0$ and $p = 1$). This suggests that the value of p with the largest gap between p and $\frac{p}{2-p}$ is the "critical value" for the random cluster model.
An interesting fact is that this critical value is $p_c = 2 - \sqrt{2}$ and this corresponds exactly to the critical value of the Ising model β_c, i.e., we have $p_c = 1 - \exp(\beta_c)$.

As mentioned before, it is not clear whether the single bond algorithm is affected by somewhat like critical slowing down or not. However this algorithm was used by Propp and Wilson [PW] to generate an exactly distributed 4200×4200 Ising state, so we do it in the same manner.

After we have shown that the presented single flip algorithms fulfills the required property, one may ask whether such a method exists for the cluster algorithms, like Swendsen-Wang, above. Because it is obvious that the presented cluster algorithms do not have something like the monotonicity property.
However, the answer is yes. Especially for the Swendsen-Wang algorithm there exists the so-called *bounding chain* method, as described in the next section.

But before we show some Figures of exactly distributed states. Note that the required starting time take the largest value at the critical temperature.
First in Figure 15 we show an exact distributed random cluster state in the low temperature case $\beta = 2\beta_c$. We supress the resulting Ising states, because they look very similar to them in Figure 13.

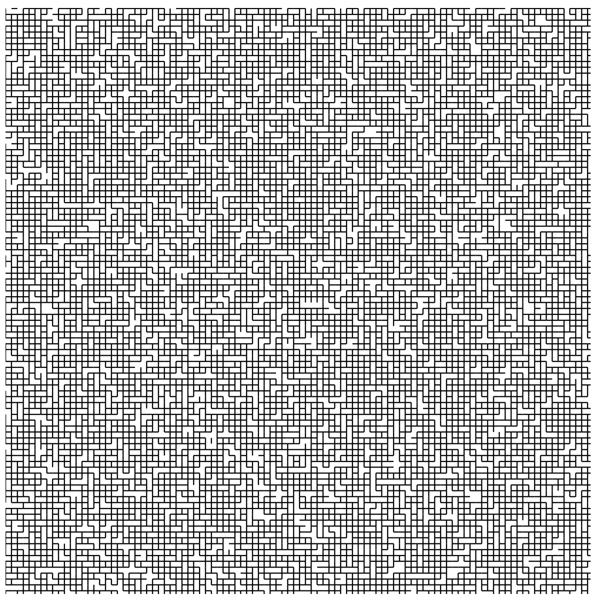


FIGURE 15. An exactly distributed 100×100 random cluster state with $\beta = 2\beta_c$ and start at time -4

We see hardly any differences to Figure 13, but it is interesting that it is enough to start at time -4 in the Propp-Wilson algorithm to get an exact sample.

For high temperatures we have nearly the same result. Also a start at -4 suffice. In Figure 16 we show a random cluster state generated from the Propp-Wilson algorithm and to Ising configurations that could result from this state.

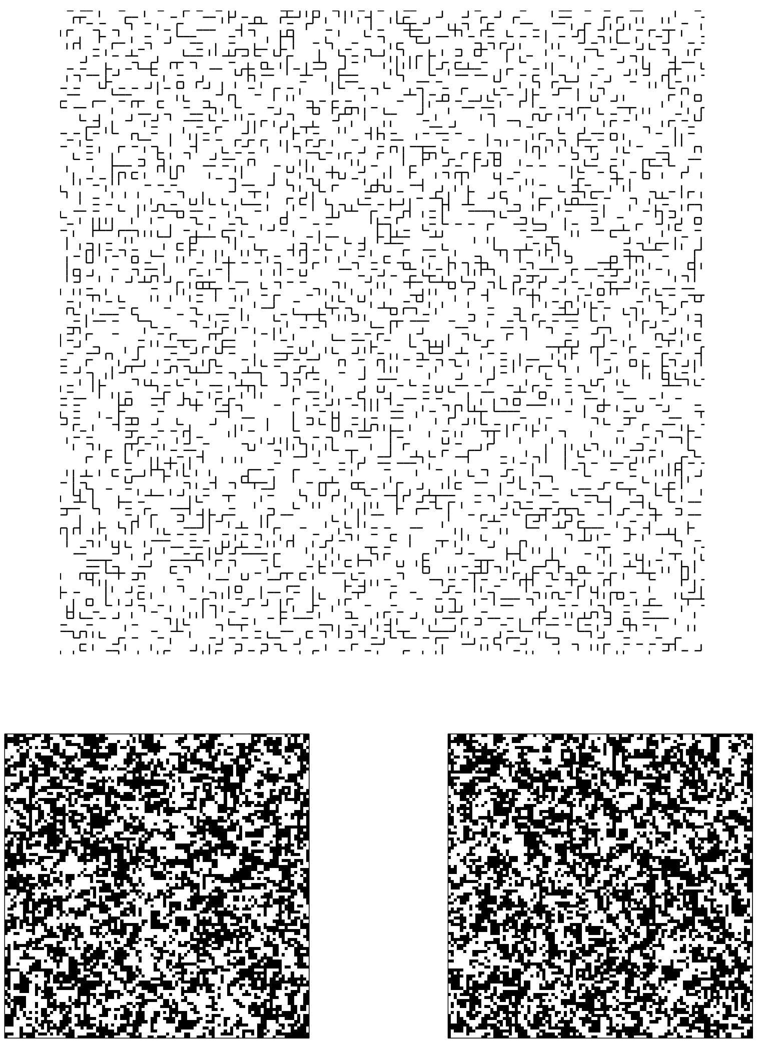


FIGURE 16. An exactly distributed 100×100 random cluster state with $\beta = \frac{1}{2}\,\beta_c$ and start at time -4

In contrast to the two figures above the next Figure 17 shows an exact distributed state at the critical value β_c (or p_c). In this case we had to start at time -32 in order that the algorithm stops.

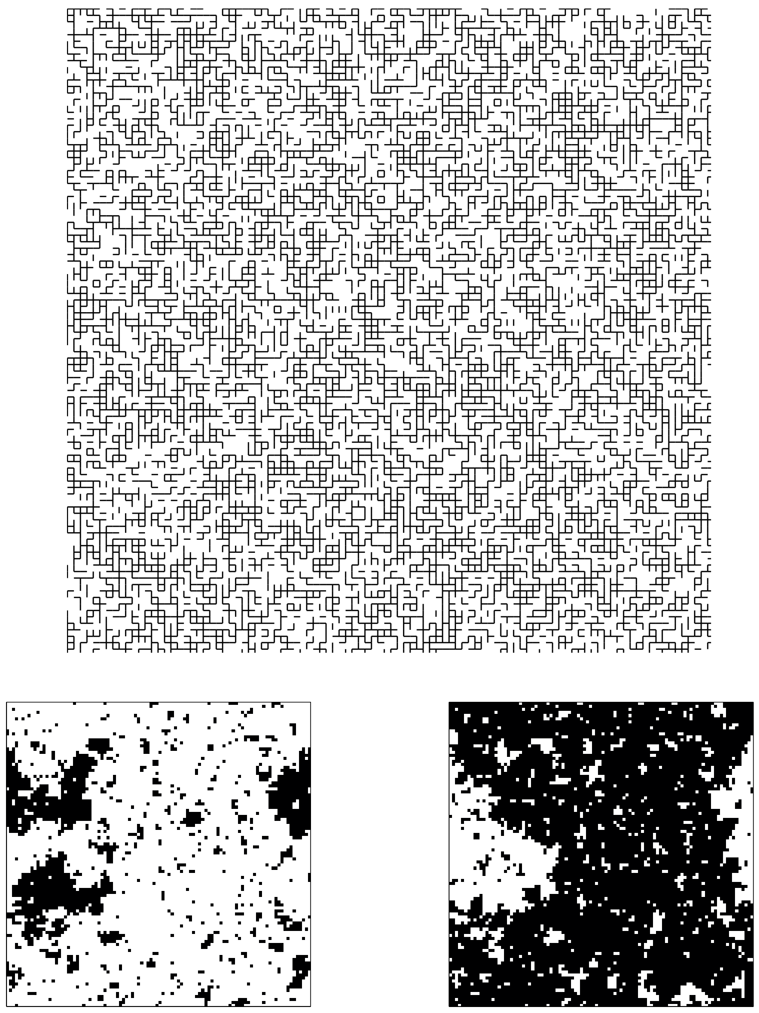
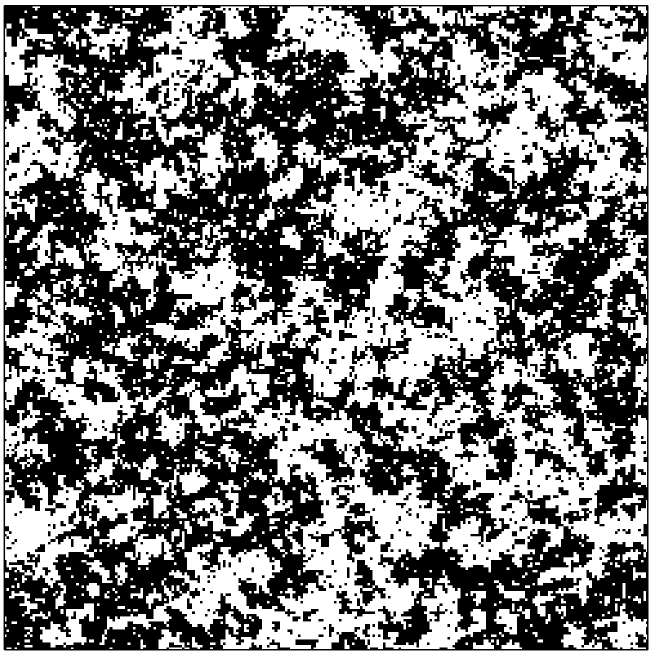


Figure 17. An exactly distributed 100×100 random cluster state at β_c and start at time -32

Before we discuss a technique to optimize the computing time of the algorithm of Propp and Wilson in the next subsection, we show some exactly distributed Ising configurations on a 250×250 lattice around the critical inverse temperature β_c. This configurations again were generated from a random cluster state, but we would see no differences to the random cluster states above. Hence we show only one resulting Ising configuration in each case.
We will see that even the smallest change of the temperature can cause huge differences in the behavior of the model.



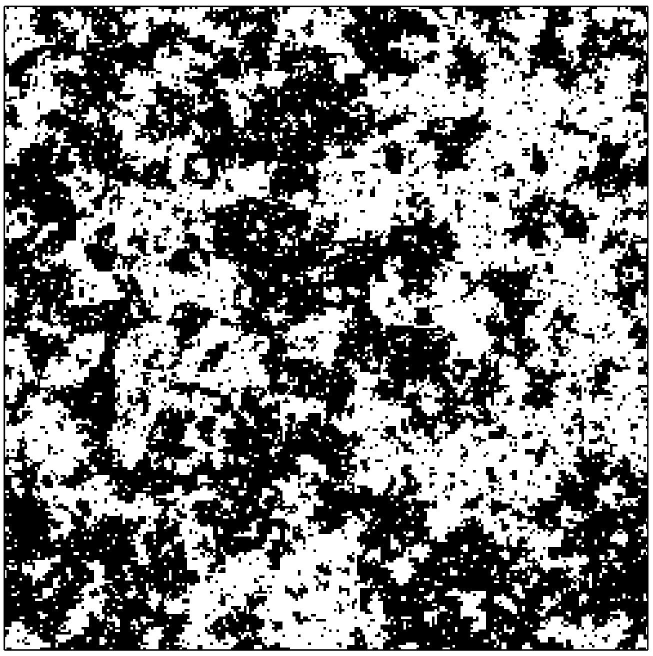


FIGURE 18. Exactly distributed 250×250 Ising configurations at $0.9\,\beta_c$ (up) and $0.95\,\beta_c$ (down) with start at time -16 (both)

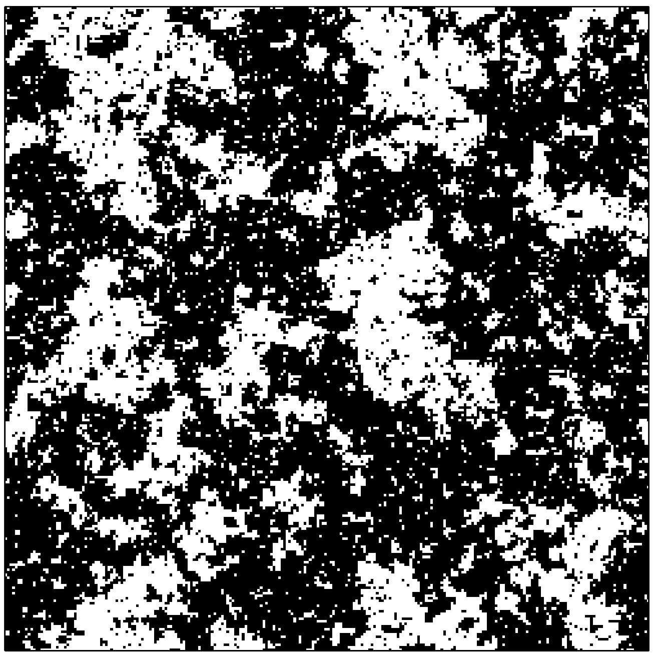
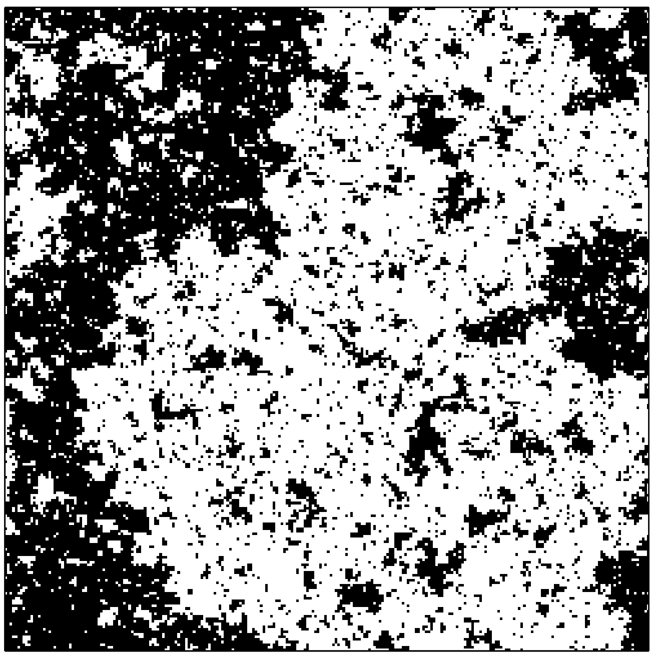
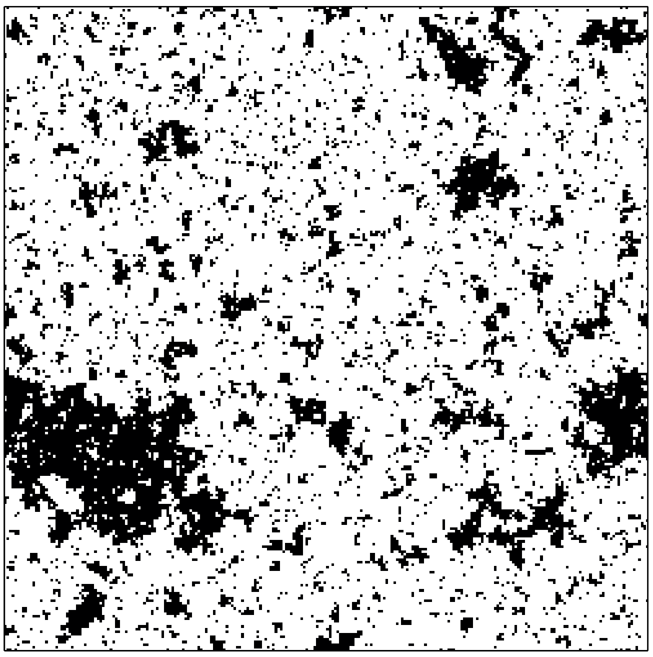



FIGURE 19. Exactly distributed 250×250 Ising configurations at $0.98\,\beta_c$ (up) and $0.99\,\beta_c$ (down) with start at time -16 and -32, respectively.



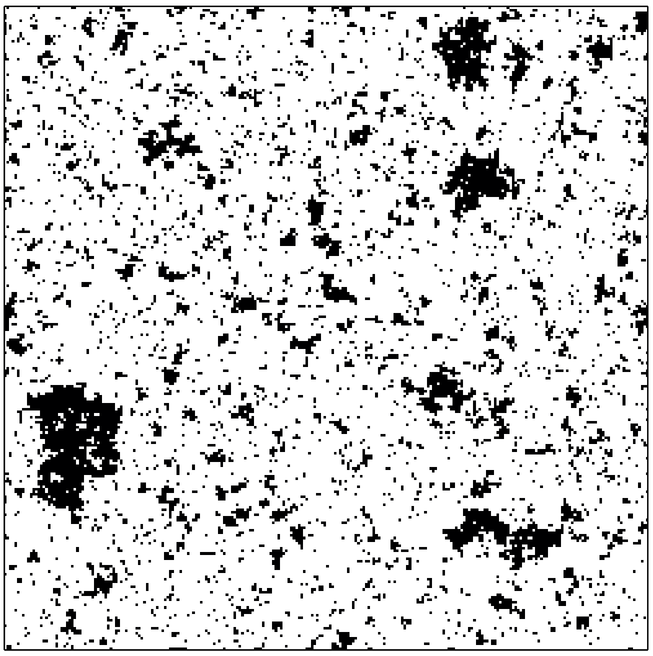
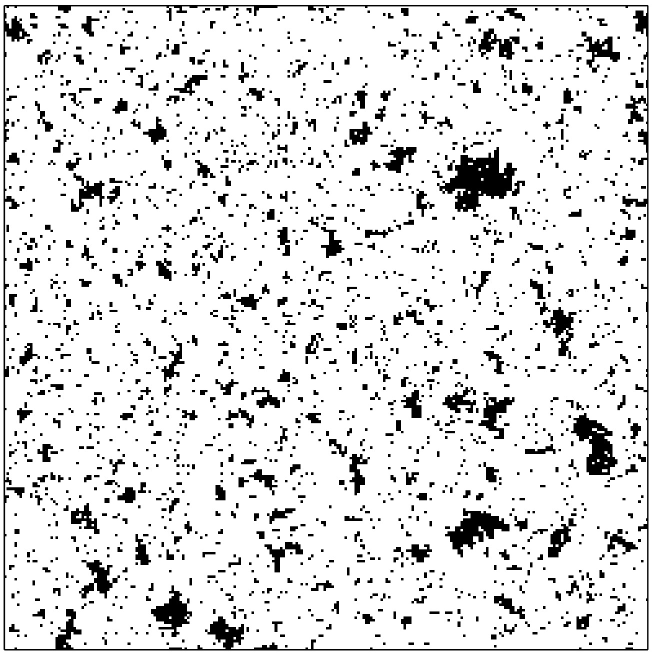


Figure 20. Exactly distributed 250 × 250 Ising configurations at β_c (up) and $1.01\,\beta_c$ (down) with start at time -16 (both)



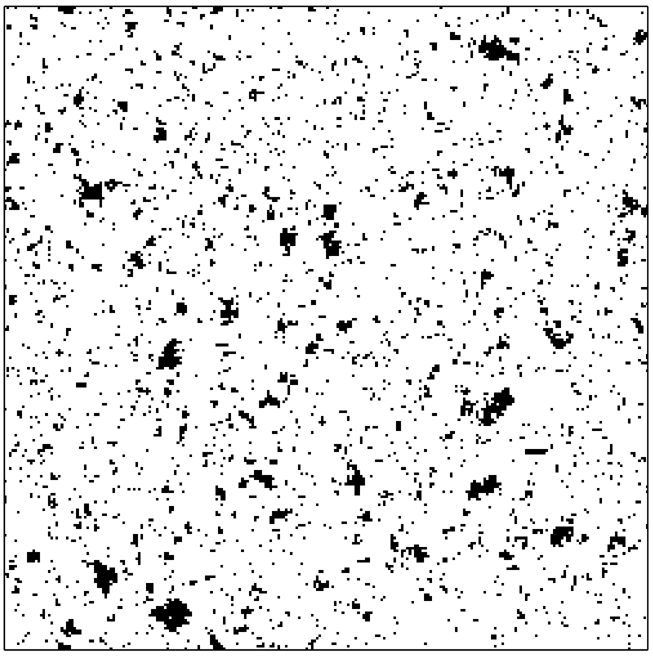


FIGURE 21. Exactly distributed 250×250 Ising configurations at $1.02\,\beta_c$ (up) and $1.05\,\beta_c$ (down) with start at time -16 (both)

We see that the behavior of the model changes dramatically from $0.9\,\beta_c$ to $1.05\,\beta_c$. Near $0.9\,\beta_c$ we have an almost zero relative magnetization whereas at $1.05\,\beta_c$ the relative (and absolute) magnetization is near 1. For infinite lattices we know that above the critical temperature, the expected magnetization is zero and we know the behavior below the critical temperature, too. But for finite lattices we don't.
The guess is that behavior of the characteristics (like the magnetization) of the model is more "smooth" than for infinite lattices. Especially one may expect that the magnetization is a differentiable function of the temperature.
In the next chapter we will see that simulations suggest that this is true.

3.6.4. *Optimization.* Now we describe a slightly optimization of the Propp-Wilson algorithm. In the algorithm we always need an increasing sequence of positive integers $N_1, N_2, \ldots$ to define the possible starting points of the chains. The question is how to choose these values.
Suppose that we are able to use the sandwiching method and note that it is not necessary to get the smallest t^* such that $F^0_{-t^*}$ is constant. An overestimate is as good as the correct value.
Assume that t^* is that correct value. Then if we take the easiest variant of starting points $N_i = i$, we have to make $2\,\frac{t^*(t^*+1)}{2}$ simulation steps, where the two in front comes from the fact that we have two copies of the chain. On the other hand, if we take $N_i = 2^i$ and we assume that $2^{k-1} < t^* \leq 2^k$, we have only to make $2\,(1 + 2 + 4 + \cdots + 2^k) < 2^{k+2}$ simulation steps. That means that we do not need to make more than $4\,t^*$ steps and so this choice is much better than the other one.
For a prove of optimality and other kinds of optimization see Propp and Wilson [PW, Section 5.2].

3.7. **Bounding Chains for Swendsen-Wang.** Bounding chains are helpful tools for detecting, whether a mapping F_{-t}^{0} from (7) is constant or not, especially in the case of update rules without the monotonicity property that is required for the sandwiching technique.
The trick is that we take another chain additionally to the original one. These chain keep track of all possible values of the original chain at each time and in each component.

First we define bounding chains in detail. Therefore we use the notation of Huber [Hu1]. An improved version of the results of this article is given in Huber [Hu2].
Let $X = (X_t)_{t\in\mathbb{N}}$ be a Markov chain on the Ising model, i.e., with state space $\Sigma = \{-1,1\}^V$ and $V = \{1,\dots,N\}$. Then the chain $B = (B_t)_{t\in\mathbb{N}}$ with state space $(2^{\{-1,1\}})^V$ is called a *bounding chain for* X if for all $v \in V$ and $t \in \mathbb{N}$

$$X_t(v) \in B_t(v) \quad\Longrightarrow\quad X_{t+1}(v) \in B_{t+1}(v).$$

(At the moment we look at $t \in \mathbb{N}$, but for $t \in -\mathbb{N}$ as we need later, this works the same way.)
Now, if we recall that there exists an update rule such that $X_t = F_0^t(X_0)$ (the random numbers are implicitly given), and we set $B_0(v) = \{-1,1\}$ for all $v \in V$, we know that $F_0^t(\Sigma)(v) = \{X_t(v) : X_0 \in \Sigma\} \subseteq B_t(v)$ for all t. That is, the chain B contains all possible values for each component of the chain X. And so it is easy to see that if $|B_t(v)| = 1$ for all $v \in V$, then the mapping $F_0^t(\cdot)$ is constant.

One possible bounding chain for the Swendsen-Wang algorithm is given below.

Recall that $\mathcal{C}_A$ denote the set of all connected components in (V,A), $A \subseteq E$. In addition $C_v \in V$ is the lowest numbered vertex in the connected component that contains v (where $V = \{1,\dots,N\}$) and $v \overset{D}{\longleftrightarrow} w$ means that v and w are connected with edges in D.

Bounding Chain Step for Swendsen-Wang

```
Input     The value of $B_t$ and $p$.

Step 1:   Let $A = \big\{\{v,w\} \in E : |B_t(v)| = |B_t(w)| = 1,$
                              $B_t(v) = B_t(w)\big\}$.

Step 2:   Let $D = \big\{\{v,w\} \in E : |B_t(v) \cap B_t(w)| \geq 1\big\}$.

Step 3:   For each edge $e$ generate a random numbers $U(e)$ with
          $U(e) \sim \text{Uniform}[0,1]$.

Step 4:   For each vertex $v$ generate a random numbers $Q(v)$ with
          $k(v) \sim \text{Uniform}\{-1,1\}$.

Step 5:   For each edge $e \in A$:
            If $U(e) < 1-p$:
              Set $A = A \setminus \{e\}$.
              Set $D = D \setminus \{e\}$.

Step 6:   For all $C \in \mathcal{C}_A$:
            Set $Z(w) = k(C_w)$ for all $w \in C$.

Step 7:   Choose a total order uniformly at random for $v \in V$.

Step 8:   For all $v \in V$:
            Set $B_{t+1}(v) = \bigcup_{w \overset{D}{\leftrightarrow} v:\, C_w \leq C_v} Z(w)$.
```

The intuition behind this chain is the following. The set $B_t(v)$ contains all possible values for v at the t-th step of the Swendsen-Wang chain $X = (X_t)_t$, i.e., $X_t(v) \in B_t(v)$ for all $v \in V$ and all t. Therefore A from Step 1 is a subset of $\big\{\{v,w\} \in E : X_t(v) = X_t(w)\big\}$ as in the Swendsen-Wang algorithm. If $|B_t(v) \cap B_t(w)| \geq 1$ for an edge $\{v,w\} \in E$, then it is possible that $X_t(v) = X_t(w)$ and so there might be an edge in random cluster state, that will temporary generated by the Swendsen-Wang algorithm, between v and w.

This means that A is the set of edges that are definitely included in the Swendsen-Wang step and D is the set of all possible edges.
Therefore if $D \setminus A$ is the empty set, one step of the bounding chain is exactly the same as in the original Swendsen-Wang chain, except for Step 7. This step could be omitted in practice, but it is necessary for some proves of running times of the algorithm.
In the case that $D \setminus A$ is not empty, edges of this set might need to be included. This yields to the fact that some components receive the color of another component. And so the set possible colors $B_{t+1}(v)$ at time $t+1$ for a vertex v is just the union over all the colors of the components that are connected to v in D. Moreover, only the components are included that are "lower numbered", because the Swendsen-Wang algorithm tells us that a component will receive the color of the lowest numbered vertex.

Now if we suppose that we start one Swendsen-Wang chain in each state of the state space Σ and the above described bounding chain simultaneously with the same random numbers, we are able to keep track of the possible values of all Swendsen-Wang chains at each time if we look at the bounding chain. So if $|B_t(v)| = 1$ for each vertex v and some t all chains are coupled and we get that the mapping $F_0^t(\cdot)$ from (7) is constant.
One could use this together to the idea of coupling from the past from the Propp-Wilson algorithm to produce samples that are exactly distributed according to the stationary distribution π_β of the chain, where as always $\beta = -\frac{1}{J}\ln(1-p)$.
We do this as before, but this time we only have to start on chain: the bounding chain. We start the bounding chain at some time $-t$ in the past (e.g. $t = 1$) and run it to zero by saving all used random numbers. If $|B_0(v)| = 1$ for each vertex v then B_0 is our sample. If not we start again at time $-2\,t$ by reusing all used random numbers, when we come back to t. Again we prove, whether all chains have coupled or not. And so on.
Since this procedure is not so intuitive like the Propp-Wilson algorithm, we finish this section with a simple example.

Example 16. Suppose a 2×2 lattice without periodic boundary conditions and $p = \frac{1}{2}$. For simplification we do the algorithm without Step 7 and choose the natural order of the vertices, that is 1, 2 in the upper row and 3, 4 in the lower, and we enumerate the edges in the same way, so up 1, left 2, right 3 and down 4.
As always our start value of the bounding chain is $B_0(v) = \{+1, -1\}$ for all $v \in V$ and in the figures below $+$, $-$ or $\pm$ stand for $\{+1\}$, $\{-1\}$ or $\{+1, -1\}$, respectively. Additionally all solid lines are edges in A from Step 1 or 5 and all dashed lines are edges in D from Step 2 or 5. (Recall that $A \subseteq D$.)

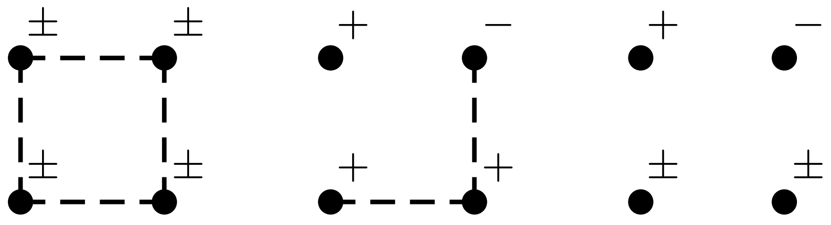


Figure 22. Left: Bounding chain step B_0 with edges from Step 1 (solid) and edges from Step 2 (dashed). Center: Edges (A and D) and spins (Z) after Step 6. Right: New Bonding chain step B_1.

We see in Figure 22 that at the beginning $A = \varnothing$ and $D = E$. We use the random numbers $U = (0.4, 0.4, 0.6, 0.7)$ and $k = (+1, -1, +1, +1)$, so only edge 1 and 2 will be removed, as we see in the center. Since $A = \varnothing$ we have only isolated vertices as connected components in A and vertex 1 is not connected to another vertex in D, so vertex 1 must have spin $+1$. Vertex 2 is the lowest numbered vertex of the rest of the vertices and so it can also get its own spin -1. All other vertices are connected to vertex 2 and so they could receive all possible spin values.

The Bounding chain step in Figure 23 is almost the same as in the first step, but we take the random numbers $U = (0.1, 0.1, 0.4, 0.9)$ and $k = (-1, +1, -1, +1)$. Therefore only edge 4 survive and vertex 3 is the lowest numbered in the connected component, so only vertex 4 could receive another spin than its own.

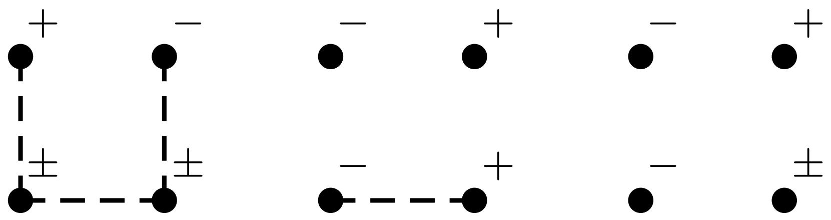


FIGURE 23. Left: Bounding chain step B_1 with edges from Step 1 (solid) and edges from Step 2 (dashed). Center: Edges (A and D) and spins (Z) after Step 6. Right: New Bonding chain step B_2.

The next Figure 24 shows that the number of undecidable spins could be larger after each step. The random numbers are $U = (0.5, 0.2, 0.6, 0.8)$ and $k = (-1, -1, +1, +1)$.

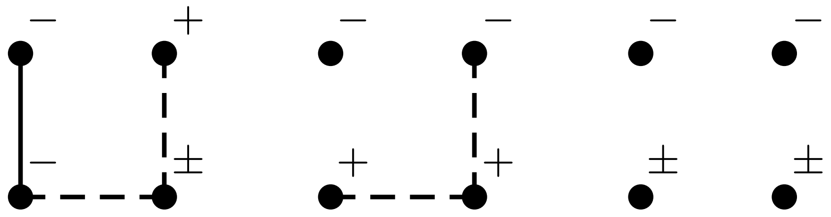


FIGURE 24. Left: Bounding chain step B_2 with edges from Step 1 (solid) and edges from Step 2 (dashed). Center: Edges (A and D) and spins (Z) after Step 6. Right: New Bonding chain step B_3.

After the next step we see that $|B_4(v)| = 1$ for all $v \in V$ and so we are finish. We have $U = (0.8, 0.2, 0.4, 0.3)$ and $k = (+1, -1, -1, +1)$.

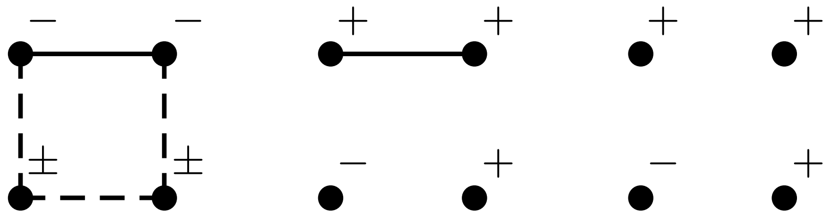


FIGURE 25. Left: Bounding chain step B_3 with edges from Step 1 (solid) and edges from Step 2 (dashed). Center: Edges (A and D) and spins (Z) after Step 6. Right: New Bonding chain step B_4.

Therefore the spin configuration in the right of Figure 25 is the only possible, no matter, what we have as initial configuration. And so the mapping $F_0^4(\cdot)$ (with the Swendsen-Wang update rule and the above defined random numbers) is constant.
However this state is not distributed according to the stationary distribution. To get such a sample we have to use some techniques like coupling from the past.

$\square$

3.8. **Conclusion.** In this section we have seen how to generate approximately and exactly distributed samples for the Ising model. How fast the given algorithms are theoretically is not completely known, but there are some estimates. See e.g. [Hu1], [LPW] or [MO]. For instance, the algorithms on the Ising model, e.g. heat bath or Metropolis, are "fast" above the critical temperature and "slow" otherwise. For the algorithms on the random cluster model we do not know how fast they are theoretically, but for simulations they are unfavorable, because there are some difficulties with the implementation of some of the algorithms. For example, to determine whether two vertices in the random cluster model are connected or not is a very hard problem. Hence (my implementation of) the heat bath algorithm on the random cluster model is very slow. However, we need this algorithm for the Propp-Wilson algorithm, because the other algorithms don't provide a monotone update rule and so we cannot use the sandwiching. So, as far as I know, the fastest method to generate an exact sample of the Ising model is to use the Propp-Wilson algorithm with the heat bath algorithm on the random cluster model, i.e. the single-bond algorithm. For my calculations on a 250×250 lattice we only have to start at time -16 to get an exactly distributed state at the critical temperature. If we choose the temperature higher or lower we need less steps than for the critical temperature. This suggests that the heat bath algorithm on the random cluster model is theoretically faster also below the critical temperature and so a better choice for simulation than the same algorithm on the Ising model, but we do not know much about that.

For the approximately sampling, i.e. for running a Markov chain where we need every state, I would prefer the Swendsen-Wang algorithm for small lattices ($N < 10^4$), because it is easy to implement and it takes a few steps per second. Additionally we don't have the problem at low temperatures that we get either an "all spins up" or an "all spins down" state after a long time, because we can jump between both states in each step. However, for larger lattices we have some difficulties with our data structure and the problem of connectedness that avoids us from sampling efficiently from the random cluster model. For example, one step of the Swendsen-Wang algorithm at the critical temperature for a 500×500 lattice takes about one minute.
But this is only a problem of the implementation, since my connectivity algorithm in `Connected.mat` from my homepage is computationally too expensive. Probably we get a huge improvement of computing time if we use some kind of *dynamic connectivity algorithm* that takes care of the connected vertices in each step and so we do not have to calculate it every time. One of the first ideas of a dynamic algorithm is given in Sweeny [S], where he uses that it is only necessary to take care of the number of connected components if we delete or add an edge. Further improvements are given in Henzinger and King [HK] and in [A] an implementation of the "Dynamic Connectivity Algorithm" of [HK] for *C++* is given in detail. However, to use it with my programs, I need an implementation for Matlab.
For large lattices the sweep versions of the Metropolis and heat bath algorithm are good choices for the simulation above the critical temperature, because at this region the stationary distribution is "unimodal" and so small steps in the state space are enough to reach stationarity in polynomial time. Additionally one can use matrix operations to speed up the calculations and so we can update all spins in a configuration in the same time than the random choice flip algorithms make only a few steps. The problem is the low temperature region where the distribution is "multimodal" and so the algorithms are surely intractable. We describe this in detail in another section.

In the next section we describe the evaluation of expectations, i.e. integrals, on the Ising model.

4. Approximation of Integrals

Now we assume that we have a function $f : \Sigma \mapsto \mathbb{R}$ on the Ising model and we want to calculate the expectation with respect to the Gibbs measure π_β, i.e. we want to calculate

$$S(f) \ := \ \mathbb{E}_{\pi_\beta}(f) \ = \ \int_\Sigma f \, d\pi_\beta.$$

Since we do not know the distribution (i.e. the normalizing constant) exactly, we cannot calculate it analytically. Therefore we use *Markov Chain Monte Carlo* to approximate the integral. Let us describe this in detail.
Assume we have an ergodic Markov chain $X = (X_t)_{t=0,1,\ldots}$ with state space $\Sigma = \{-1, 1\}^V$, $V = \{1, \ldots, N\}$, and stationary distribution π_β. Then one could use the estimator

$$S_n(f) \ = \ \frac{1}{n} \sum_{i=1}^{n} f(X_i)$$

to approximate the integral. The ergodic theorem (see e.g. [K, Satz 20.16]) yields that

$$\lim_{n\to\infty} S_n(f) \ = \ S(f)$$

and so the algorithm is correct. The question is, how fast this convergence is and how is the dependence on N. Therefore let q be the matrix of transition probabilities of the Markov chain X and $\lambda_0, \lambda_1, \ldots, \lambda_{2^N-1}$ its eigenvalues (ordered by size). The corresponding eigenfunctions (or eigenvectors) are $\psi_0, \ldots, \psi_{2^N-1}$. Then we know from the fact that X is ergodic and the Perron-Frobenius theorem that

$$1 \ = \ \lambda_0 \ > \ \lambda_1 \ \geq \ \ldots \ \geq \ \lambda_{2^N-1} \ > \ -1.$$

Since q is a stochastic matrix ψ_0 is always constant 1.
The most important factor for the calculation of the error of our approximation scheme is the second largest eigenvalue λ_1 or equivalently the *spectral gap* of the Markov chain $1-\lambda_1$, as we will see in a moment.
From now we assume that the first state of our Markov chain was chosen with respect to the stationary distribution, i.e. $X_0 \sim \pi_\beta$, to avoid the additional error that results from the (possibly bad) initial distribution. Additionally, only in this case the estimator is unbiased.

The error of our method S_n for the approximation of the integral of a function $f \in \mathbb{R}^\Sigma$ is defined as the *mean square error*

$$e(S_n, f) \;=\; \Big(\mathbb{E}_q\big(S_n(f) - S(f)\big)^2\Big)^{1/2},$$

where $\mathbb{E}_q$ is the expectation over all possible paths of the Markov chain induced by the transition matrix q, i.e.

$$\begin{aligned}\mathbb{E}_q(S_n(f)) \;&=\; \sum_{(x_1,\dots,x_n)\in\Sigma^n} \Big(\frac{1}{n}\sum_{i=1}^{n} f(x_i)\Big)\; \mathbb{P}\big((X_1,\dots,X_n) = (x_1,\dots,x_n)\big)\\ &=\; \sum_{(x_1,\dots,x_n)\in\Sigma^n} \Big(\frac{1}{n}\sum_{i=1}^{n} f(x_i)\Big)\; \pi_\beta(x_1)\cdot q(x_1,x_2)\cdots q(x_{n-1},x_n).\end{aligned}$$

Again due to the start with respect π_β we know an explicit error formula, as stated in the next theorem. See e.g. Rudolf [Rud].

Theorem 17. *Let $f \in \mathbb{R}^S$, where S is the state space of a finite ergodic Markov chain $X = (X_t)_{t=0,1,\dots}$, which is reversible with respect to the measure π and have the transition matrix q. Additionally let $X_0 \sim \pi$. Then*

$$e(S_n, f)^2 = \frac{1}{n^2} \sum_{k=1}^{|S|-1} a_k^2\, W_n(\lambda_k) \tag{10}$$

with

$$W_n(\lambda_k) = \frac{n - 2\lambda_k - n\lambda_k^2 + 2\lambda_k^{n+1}}{(1-\lambda_k)^2} \tag{11}$$

and

$$a_k = \langle f, \psi_k\rangle_\pi.$$

Proof. Let $g := f - S(f) \in \mathbb{R}^S$. With the same notation as above $f(x) = \sum_{i=0}^{|S|-1} a_k\psi_k(x)$ and so $g(x) = \sum_{i=1}^{|S|-1} a_k\psi_k(x)$, because $a_0 = 0$ since $S(g) = 0$.

As mentioned above, S_n is unbiased. So

$$\begin{aligned} e(S_n,f)^2 &= \mathbb{E}_q\Big(S_n(f) - \mathbb{E}_q\big(S_n(f)\big)\Big)^2 \\ &= \mathbb{E}_q\left(\frac{1}{n}\sum_{i=1}^n f(X_i) - S(f)\right)^2 = \mathbb{E}_q\left(\frac{1}{n}\sum_{i=1}^n g(X_i)\right)^2 \\ &= \frac{1}{n^2}\,\mathbb{E}_q\left(\sum_{i=1}^n g(X_i)\right)^2 = \frac{1}{n^2}\sum_{i,j=1}^n \mathbb{E}_q\Big(g(X_i)g(X_j)\Big)\,. \end{aligned}$$

Now with $i \le j$:

$$\begin{aligned} \mathbb{E}_q\, g(X_i)g(X_j) &= \mathbb{E}_q\left[\left(\sum_{k=1}^{|S|-1} a_k\psi_k(X_i)\right)\left(\sum_{l=1}^{|S|-1} a_l\psi_l(X_j)\right)\right] \\ &= \sum_{k=1}^{|S|-1}\sum_{l=1}^{|S|-1} a_k a_l\,\mathbb{E}_q\psi_k(X_i)\,\psi_l(X_j) = \sum_{k=1}^{|S|-1}\sum_{l=1}^{|S|-1} a_k a_l\,\langle\psi_k,\, q^{j-i}\psi_l\rangle_\pi \\ &= \sum_{k=1}^{|S|-1}\sum_{l=1}^{|S|-1} a_k a_l\,\lambda_l^{j-i}\,\langle\psi_k,\,\psi_l\rangle_\pi = \sum_{k=1}^{|S|-1} a_k^2\,\lambda_k^{j-i}\,. \end{aligned}$$

The last equality comes from the orthonormality of the eigenfunctions. Now with the identity

$$\sum_{1\le i<j\le n} x^{j-i} = \frac{(n-1)x - nx^2 + x^{n+1}}{(1-x)^2}\,,$$

we get

$$\begin{aligned} e(S_n,f)^2 &= \frac{1}{n^2}\sum_{k=1}^{|S|-1} a_k^2\left(n + 2\sum_{1\le i<j\le n}\lambda_k^{j-i}\right) \\ &= \frac{1}{n^2}\sum_{k=1}^{|S|-1} a_k^2\left(n + 2\frac{(n-1)\lambda_k - n\lambda_k^2 + \lambda_k^{n+1}}{(1-\lambda_k)^2}\right) \\ &= \frac{1}{n^2}\sum_{k=1}^{|S|-1} a_k^2\,W_n(\lambda_k)\,. \end{aligned}$$

□

It is easy to see that the function $W_n(x)$ from (11) is monotone increasing for $x \in [-1,1)$ and so we get the following corollary for the worst-case-error of the estimator S_n for functions from the unit ball of $L_2(\pi)$.

Corollary 18. *With the assumptions of Theorem 17 we get*

$$\sup_{||f||_{2,\pi}\leq 1} e(S_n, f)^2 \;=\; e(S_n, \psi_1)^2 \;=\; \frac{1+\lambda_1}{n(1-\lambda_1)} \;-\; \frac{2\lambda_1(1-\lambda_1^n)}{(n(1-\lambda_1))^2}.$$

Therefore we see that maximal error of such a method is completely determined by the second largest eigenvalue of the transition matrix of the underlying Markov chain, but unfortunately this is unknown in our case.

Remark 19. In the case of an arbitrary initial distribution (not the stationary one) we have also an explicit (but more complicated) formula of the error. This formula depends additionally on some kind of distance between the initial and the stationary distribution of the chain and this could be very large, especially in the case of starting in a single point. For details and associated estimates see again [Rud].

Additionally to the error of a Markov Chain Monte Carlo method, such as S_n, the "distance" of the stationary distribution to the n-step distribution of the Markov chain is also determined by the second largest eigenvalue of the associated transition matrix. Hence one could assume that the running time of the Propp-Wilson algorithm is also determined by this value.
In the next subsection we present some simulation results that show the large improvement of the algorithm, if we start the Markov chain in the stationary distribution.

4.1. **Spontaneous Magnetization.**

This subsection contains some results of the simulations that I have made with the self-implemented Matlab programs. If you want to check this results or make your own simulations, you can download all files on my homepage.
Now let the function that we want to integrate fixed to be the expected (absolute) magnetization per spin

$$M(\sigma) \;:=\; \frac{1}{N}\left|\sum_{v\in V} \sigma_v\right|, \qquad \sigma\in\Sigma. \tag{12}$$

Recall that $V = \{1,\ldots,N\}$. We see that $M(\cdot) \in [0,1]$.

If (V, E) is a square lattice with only nearest neighbors and $N = k^2$ for some $k \in \mathbb{N}$, then we know the exact expectation if k goes to infinity, as we have seen in Section 1.
For finite lattices it is still an open problem to find an explicit representation of the exact solution of the expected magnetization. Therefore we use simulations to get an approximation of the solution, nevertheless we do not know how exact this approximations are, because we do not know the eigenvalues of the corresponding Markov chains.

To get an impression of the exact solution, we show our "best" approximation first. In Figure 26 we see the expected magnetization against the temperature for different lattice sizes. Since a larger lattice causes that we need more time for each step of the Markov chain (especially for the Propp-Wilson algorithm), we made a different number of steps for each size. However, this simulation takes about 3 month.

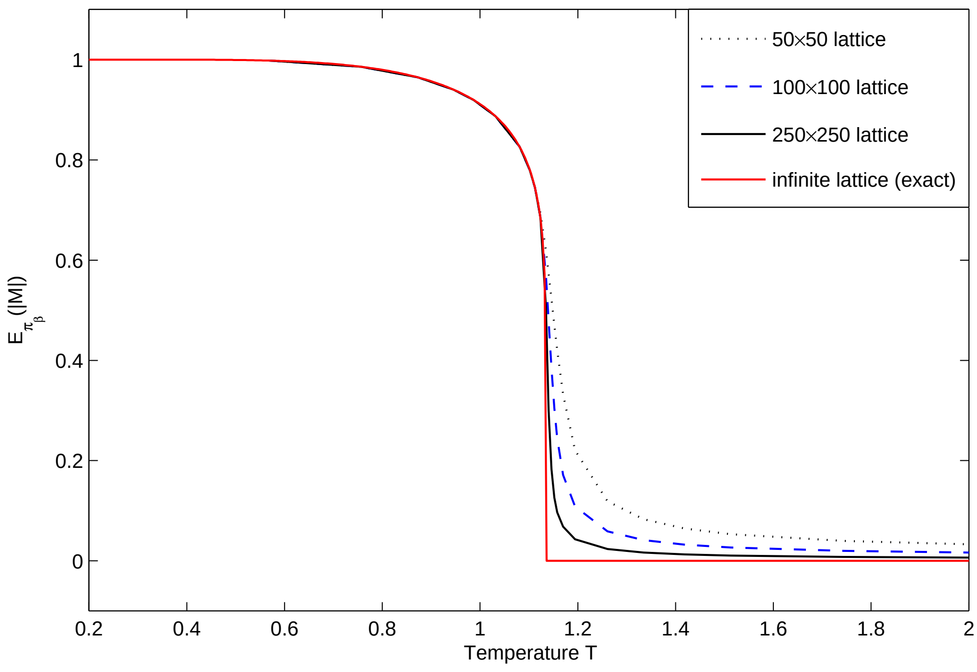


FIGURE 26. Simulations of the expected magnetization for different lattice sizes. Number of simulation steps: $k = 50$: $2 \cdot 10^8$, $k = 100$: 10^8 and $k = 250$: $3 \cdot 10^7$

We see that, since the exact solution for an infinite lattice is not differentiable at the critical temperature, the smaller the lattice the "smoother" is the approximation. In the low temperature region the curves are anyhow close together.
It is very likely that this results are close to the exact solutions, because, additionally to the huge number of steps, we take exactly distributed initial states and so the used algorithm is unbiased.
Now we want to show what happens, if we do not start at the stationary distribution. Therefore we first show in Figure 27 the magnetization of some exactly distributed states that come from the Propp-Wilson algorithm. These are also the initial states of our simulation above.

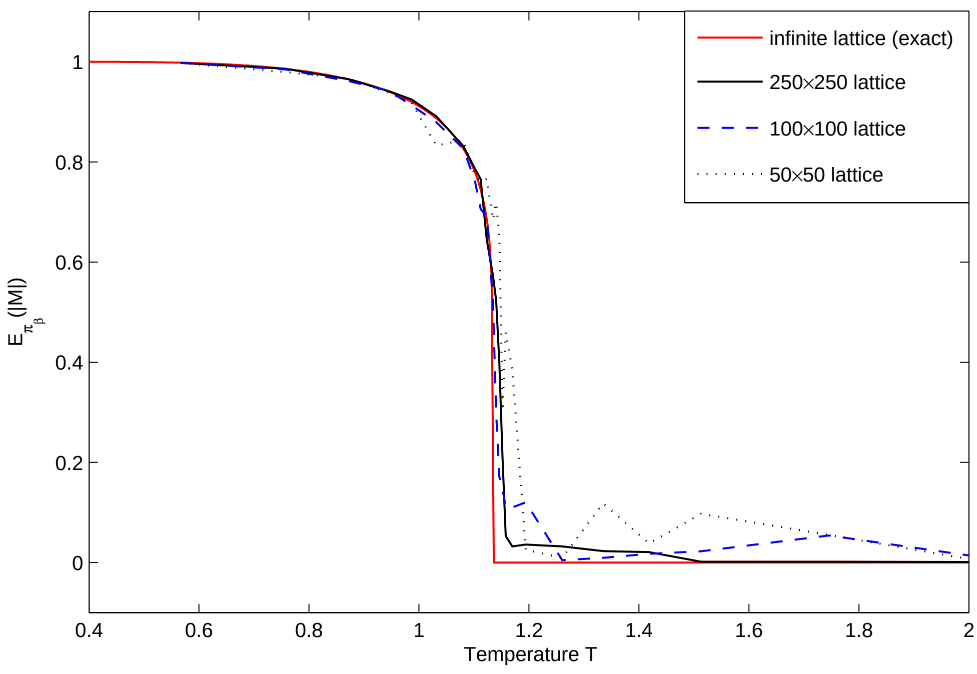


FIGURE 27. Absolute magnetization of the initial states

As expected, the magnetization of this states is close to the results of the simulation, because they have the right distribution. Therefore the Markov chain do not have to change the characteristics of the initial states too much. Note that (for this function) it is equal whether we have much more "spin up" or "spin down" vertices.

Now we want show that the convergence of the used Markov chain is much faster above the critical temperature. Therefore we take other initial states and restrict ourself to the 250×250 lattice, because the smaller lattices show the same behavior.
First we use uniform distributed initial states, i.e. to each vertex we assign a spin with equal probability $\frac{1}{2}$. It is easy to see that the magnetization of such a configuration is close to zero and so we are in the situation that only below (and a bit above) the critical temperature we are far away from the solution. In Figure 28 we show the results of the simulation after a different number of steps.

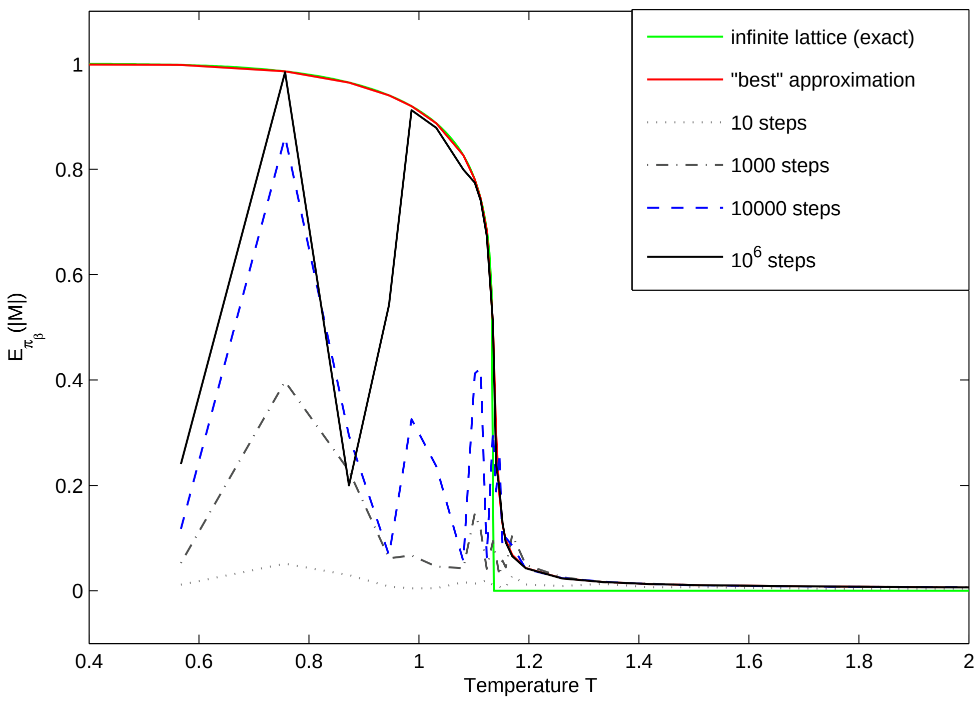


FIGURE 28. Simulation of the expected magnetization with uniform distributed initial states

We see that even after 10^6 steps we are still far far away from the correct solution at some sampling points. Additionally there are points where the distance became bigger (like the peak of the black curve). The reason for this behavior could be that we reach a metastable state, where we have almost zero magnetization, but most of the vertices have the same spin as their neighbors. We have talked about this in Section

3.2 and called the opposite regions of the state Weiss domains.
Figure 28 let us suggest that the Metropolis algorithm (or each single spin flip algorithm) is very slow in the low temperature region and possibly never converges. Since the absolute magnetization does not depend on the sign of the configuration, this is another problem than the two peaks of the distribution π_β and so it is unlikely to find positive results for the order of convergence below the critical temperature for other (more complicated) functions.
Now we take the "all spins up" state as the initial state at all temperatures. This state has magnetization 1 and so we get the correct solution for low temperatures. In Figure 29 we show again the results of one simulation after different numbers of steps. We will see that the behavior is completely different in this case.

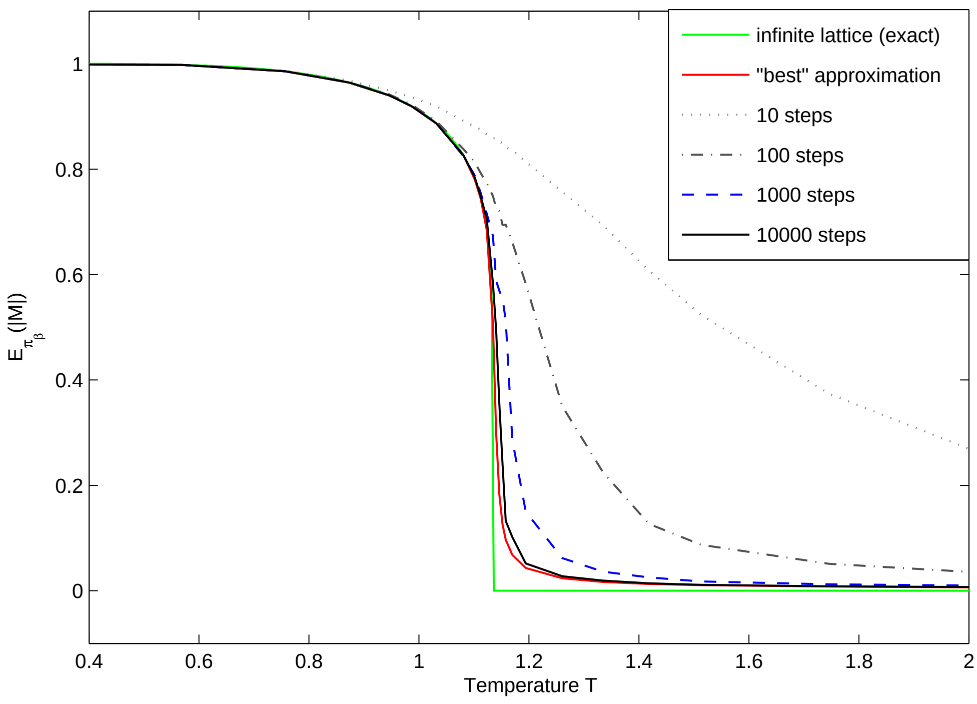


FIGURE 29. Simulation of the expected magnetization with "all spins up" initial states

First note that the simulation steps which are plotted are much smaller than above. Another observation is that we have a monotone convergence in every point. Hence we will definitely reach the correct solution.

Additionally we are close to our "best" approximation already after 10000 steps.
The results of this section affirm our guess that single spin flip algorithms, like the Metropolis algorithm, for the Ising model are useful only for high temperatures. So we have to analyze other methods to find an algorithm that is fast (hopefully) at all temperatures.
The first idea is to analyze the above described cluster algorithms in detail to find bounds on the mixing time. For the Swendson-Wang algorithm there is something known, see e.g. [Hu2]. But most of the known results are not satisfactory, because they deal with other structures of the underlying graph. For example graphs with higher degree or trees.
Another (and by me preferred) idea is to analyze the random cluster model. Since we make a random walk on subsets of a given set (here the set of edges E), this is equivalent to a random walk on the discrete cube $\{0,1\}^{|E|}$, which is exactly the Ising model, but now with $|E|$ instead of $|V|$ vertices. The difference is the distribution on the "new Ising model" and we hope that this distribution is easier to analyze. The reason for this guess is that apparently the bottleneck of the random cluster distribution is not such significant as by the Boltzmann distribution.

References

[A] D. Alberts. Implementation of the Dynamic Connectivity Algorithm by Monika Rauch Henziger and Valerie King, *http://ftp.mi.fu-berlin.de/reports/tr-b-95-10.ps.gz, Reports TU Berlin, 1995*

[B] E. Behrends. Introduction to Markov Chains, *Vieweg-Verlag, Braunschweig/Wiesbaden, 2000*

[ES] R. Edwards, A. Sokal. Generalization of the Fortuin-Kasteleyn-Swendsen-Wang representation and Monte Carlo algorithm, *Physical Review D, Vol. 38, No. 6, 1988*

[FK] C.M. Fortuin, P.W. Kasteleyn. On the Random-cluster model I. Introduction and Relation to other models, *Physica 57, 1972, p. 536-564*

[GG] G. Grimmett. The Random-Cluster Model, *Springer-Verlag, Berlin, 2006*

[Hä] O. Häggström. Finite Markov Chains and Algorithmic Applications, *Cambridge University Press, 2002*

[HK] M. Henzinger, V. King. Randomized Fully Dynamic Graph Algorithms with Polylogarithmic Time per Operation, *Journal of the ACM, Vol. 46, 1999, p. 502-516*

[Hu1] M. Huber. Efficient Exact Sampling From the Ising Model Using Swendsen-Wang, *2000*

[Hu2] M. Huber. A Bounding Chain for Swendsen-Wang, *Random Struct. Alg., 22: 43-59, 2003*

[K] A. Klenke. Wahrscheinlichkeitstheorie, *Springer-Verlag, Berlin, 2006*

[LPW] D.A. Levin, Y. Peres, E.L. Wilmer. Markov Chains and Mixing Times, *American Mathematical Society, Providence, RI ,2009*

[MO] F. Martinelli, E. Olivieri. Approach to Equilibrium of Glauber Dynamics in the One Phase Region. I. The Attractive Case, *Commun. Math. Phys. 161, 447-486 (1994)*

[O] L. Onsager. Crystal statistics I. A two dimensional model with order-disorder transition, *Physical Review, Vol. 65, 1944, p. 117-149*

[PW] J.G. Propp, D.B. Wilson. Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics, *1996*

[Rud] D. Rudolf. Error bounds for computing the expectation by Markov Chain Monte Carlo, *arXiv e-prints, 2009*

[S] M. Sweeny. Monte Carlo study of weighted percolation clusters relevant to the Potts models, *Physical Review, Vol. 27, 1983, p. 4445-4455*

[W] U. Wolff. Collective Monte Carlo Updating for Spin Systems, *Physical Review, Vol. 62, No. 4, 1989*

FRIEDRICH SCHILLER UNIVERSITY JENA
homepage: `HTTP://USERS.MINET.UNI-JENA.DE/~ULLRICH/`

E-mail address: `mario.ullrich@uni-jena.de`