University of West Bohemia
Faculty of Applied Sciences
Department of Mathematics

# BACHELOR THESIS

## Propp-Wilson Algorithm for Perfect Simulations and its Modifications

Pilsen 2013 Kristýna Chrbolková

**Abstract**

The aim of the thesis is to study Propp-Wilson algorithm and its modifications. This algorithm, also called coupling from the past, is used for the so called perfect simulation that allows us in favourable cases to sample exactly from the stationary distribution of a Markov chain. We study the original algorithm, its monotone "sandwiching" modification for Markov chains with an ordering on the state space and the so called Wilson's or read-once randomness modification. And finally we apply the algorithm to the Ising model.

I would like to thank my thesis supervisor, Ing. Jan Pospíšil, Ph.D., for his support and guidance. I would also like to thank RNDr. Pavel Popela, Ph.D. from VUT in Brno for his advise and motivation.
My one week internship at VUT in Brno was supported by the project A-Math-Net - knowledge transfer network in applied mathematics (project no. CZ.1.07/2.4.00/17.0100). This project is co-financed by the European Social Fund and the state budget of the Czech Republic.

I hereby declare that this bachelor thesis is completely my own work and that I used only the cited sources.

Plzeň, June 27, 2013

# Contents

# Chapter 1

# Introduction

Coupling from the Past (CFTP), a method generating independent draws from a stationary distribution of a Markov chain, was invented in 1996 by James Propp and David Wilson. It was published in their paper Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics [7]. After that many scientists used it, modified it and also researched some of its aspects (accuracy, efficiency).

Before we get to the structure of this thesis let us talk a bit about the motivation for CFTP. Why do we need such a method? It is quite simple – it can serve as a random number generator, producing draws from a given distribution in case, when we are not able to sample from this distribution for example by transforming uniformly distributed random variable into one with the desired distribution. In such a case we can solve our task by formulating an ergodic Markov chain, that has the desired distribution as its stationary distribution, and simulating it till we get a draw from the stationary distribution.

This can be done approximately by Monte Carlo methods, that had been known long before CFTP occurred. In Monte Carlo we just run the Markov chain for a sufficiently long time period, getting a draw from the stationary distribution as an output. But this method does not give us the exact distribution of outputs and is also very demanding in terms of determining for how long do we have to run the chain and when to stop.

These problems can be avoided by using the Propp-Wilson algorithm, that is unlike the usual Monte Carlo methods able to identify on its own when the stationary distribution was reached and the outputs are distributed exactly according to the stationary distribution.

In Chapter 2 of this thesis we will present the crucial definitions and theorems needed in the rest of this text. They were put together from [6], [9] and [3]. We will

also introduce a trivial Markov chain with a very small state space on that we will show all of the properties necessary for CFTP. This example will be used throughout the whole thesis and all of the basic simulations.

Next chapter is about the computer simulation of Markov chains. We will explain some theory from [3] and also explain what method of updating the chain we will use for our example.

Chapter 4 sums up the basic theory of the Propp-Wilson algorithm according to [3], [5] and [8] and shows some outputs of the simulations.

After that we will focus on two modifications of the basic algorithm, implement them again for the example introduced in Chapter 2 and see the results. The first modification is so called sandwiching and it is the topic of Chapter 5, for that we used [3]. Chapter 6 is about the Read-Once Randomness modification according to [3], [10] and [2].

Finally in Chapter 7 we abandon the basic example and apply one of the modifications of the algorithm on Ising model. The theory of the Ising model and the notions how to implement CFTP for this model we get from [1], [4] and [3].

# Chapter 2

# Preliminaries

In this chapter we will introduce some basic theory that we will need in the rest of this thesis. Because CFTP is a method for sampling from the stationary distribution of Markov chains first thing we need to define is a Markov chain.

**Definition 2.1.** Let $(\Omega, A, P)$ be a probability space and let $T \subset R$. A collection of real random variables $\{X_t, t \in T\}$ defined on $(\Omega, A, P)$ is a *random (stochastic) process*.

In this work we will consider only random processes with discrete time $n \in \mathbf{Z}$ and also discrete and finite state space $S \subset \mathbf{Z}$. In addition to that we will require a few more restrictions.

**Definition 2.2.** Discrete-time *Markov chain* is a random process $(X_n, n \in \mathbf{Z})$ with state space $S \subset \mathbf{Z}$ meeting the so called Markov property, saying, that the conditional probability of the chain being in state $i$ in time $n+1$, when we know all the states it has been in before, is dependant only on the state in which it has been in the previous moment $n$:

$$P(X_{n+1} = j | X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = P(X_{n+1} = j | X_n = i).$$

First we need the chain to be time-homogeneous, that means that the probability of going from state $i$ to state $j$ is the same in any time:

$$P(X_{n+1} = j | X_n = i) = P(X_{m+1} = j | X_m = i) \quad \forall n, m \in \mathbf{Z}.$$

When a Markov chain is time-homogeneous we can create a *transition-probability matrix* $P$, in that we collect all the transition probabilities between any pair of states in $S$, defined as the conditional probabilities of going from state $i$ to state $j$:

$$p_{i,j} := P(X_{n+1} = j | X_n = i) \quad \forall i, j \in S.$$

Example 1:
Let us consider a Markov chain with state space $S = \{s_1, \ldots, s_5\}$ and transition-probability matrix

$$P = \begin{bmatrix} 1/3 & 2/3 & 0 & 0 & 0 \\ 1/3 & 0 & 2/3 & 0 & 0 \\ 0 & 1/3 & 0 & 2/3 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix}$$

Such a Markov chain can be interpreted as a ladder walk. We have a ladder and we can go only one step up or one step down. When we are all the way down, we can go only up or stay where we are, the same applies to the upper state, from where we can go only down or stay there. In Figure 2.1 this idea is demonstrated on a transition graph.

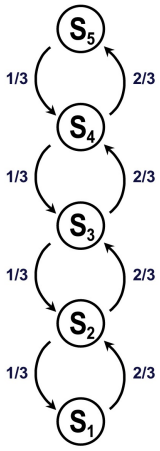


**Figure 2.1**: Transition Graph for Example 1

Second we need the Markov chain to be ergodic.

**Definition 2.3.** A Markov chain $(X_n, n \in \mathbf{Z})$ is said to be *aperiodic*, when all states $S$ are aperiodic, i.e. if

$$\forall i \in S: \quad gcd(n : P(X_n = i | X_0 = i) > 0) = 1.$$

**Definition 2.4.** A Markov chain $(X_n, n \in \mathbf{Z})$ is called *irreducible*, when all its states are irreducible:

$$\forall i, j \in S \quad \exists n > 0: \quad p_{i,j}^{(n)} := P(X_n = j | X_0 = i) > 0.$$

**Definition 2.5.** A Markov chain $(X_n, n \in \mathbf{Z})$ is *ergodic* if it is aperiodic and irreducible.

Next property we need the chain to have is reversibility.

**Definition 2.6.** Markov chain $(X_n, n \in \mathbf{Z})$ with transition matrix $P = (p_{i,j})$ is *reversible* if there exists a probability distribution over states $\pi = (\pi_i)$ such that

$$\forall i, j \in S : \pi_i p_{i,j} = \pi_j p_{j,i}$$

And finally we can get to the stationary distribution of Markov chain.

**Definition 2.7.** A row vector $\pi = (\pi_i)$ is a *stationary distribution* of Markov chain $(X_n, n \in \mathbf{Z})$ with state space $S = (s_i)$ if it is a probability distribution ($\pi_i \geq 0$ and $\sum_i \pi_i = 1$) and it also satisfies

$$\pi P = \pi.$$

**Theorem 2.1.** *For every ergodic Markov chain there exists at least one stationary distribution* $\pi$.

The proof of this theorem can be found for example in [3].

**Example 1(continued):** Stationary distribution of the Markov chain from Example 1 can be easily found:

$$\begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 \end{bmatrix} * \begin{bmatrix} 1/3 & 2/3 & 0 & 0 & 0 \\ 1/3 & 0 & 2/3 & 0 & 0 \\ 0 & 1/3 & 0 & 2/3 & 0 \\ 0 & 0 & 1/3 & 0 & 2/3 \\ 0 & 0 & 0 & 1/3 & 2/3 \end{bmatrix} = \begin{bmatrix} \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 \end{bmatrix}$$

By computing this we get the system of five linear equations

$$1/3\pi_1 + 1/3\pi_2 = \pi_1$$
$$2/3\pi_1 + 1/3\pi_3 = \pi_2$$
$$2/3\pi_2 + 1/3\pi_4 = \pi_3$$
$$2/3\pi_3 + 1/3\pi_5 = \pi_4$$
$$2/3\pi_4 + 2/3\pi_5 = \pi_5$$

for five unknowns $\pi_1, \ldots, \pi_5$.
Non-trivial solution of this system is

$$\pi_2 = 2\pi_1, \pi_3 = 4\pi_1, \pi_4 = 8\pi_1, \pi_5 = 16\pi_1,$$

where $\pi_1$ is arbitrary.
By adding the property of any probability distribution, that the sum of all the probabilities in this distribution has to be equal to 1

$$\pi_1(1 + 2 + 4 + 8 + 16) = 1,$$

and so

$$\pi_1 = 1/31$$

we get the stationary distribution of our chain in this form

$$\pi = \begin{bmatrix} 1/31 & 2/31 & 4/31 & 8/31 & 16/31 \end{bmatrix}.$$

Now we can check if the Markov chain in our example has all the needed properties.
1. Aperiodicity:
for state $s_1 \in S : gcd(1, 2, 4, 6, \dots) = 1$
for state $s_2 \in S : gcd(2, 3, 4, 5, \dots) = 1$
for state $s_3 \in S : gcd(2, 4, 5, 6, \dots) = 1$
for state $s_4 \in S : gcd(2, 3, 4, 5, \dots) = 1$
for state $s_5 \in S : gcd(1, 2, 4, 6, \dots) = 1$

2. Irreducibility:
it is obvious (see Picture 1), that from any state we can get to any other state with probability greater than zero.

3. Reversibility:
we will only do this for a few randomly chosen couples $i, j$.
for states $s_1, s_2 \in S : 1/31 * 2/3 = 2/31 * 1/3$
for states $s_2, s_4 \in S : 2/31 * 0 = 8/31 * 0$
for states $s_3, s_4 \in S : 4/31 * 2/3 = 8/31 * 1/3$
by repeating this procedure for every combination of $i$ and $j$ we can check this property for the whole chain.

In the rest of this text we will consider only time-homogeneous ergodic Markov chains with discrete time and finite and discrete state space, so every time we mention a Markov chain all these properties will be anticipated.

# Chapter 3

# Computer Simulation of Markov Chain

Next we need a way to simulate Markov chains using a computer. For that we need a programming language with a (pseudo) random numbers generator that will produce a sequence $U_0, U_1, \ldots$ of random variables with the uniform distribution on $(0,1)$. We will use the `rand()` function in Matlab, that implements the Mersenne-Twister algorithm.

You can see [3] for more information about the *initial function*, however, in the Propp-Wilson algorithm there is no need for that since we will be starting from all possible states and because of that we do not need to generate one randomly chosen initial state.

To move from one state to the other with passing time we construct a piecewise constant function $\phi$ called an *update function*, that gives us the state of the Markov chain in time $n$ based on the state in time $n-1$ and a random number $U_n$. In order for that function to be valid we need the total length of intervals for which the input state is $s_i$ and the output state $s_j$ be equal to $p_{i,j}\ \forall s_i, s_j \in S$.

**Example 1 (cont.):** For our example of ladder walk we can use this update function:

$$\phi(s_1, x) = \begin{cases} s_1, & x \in [0, 1/3) \\ s_2, & x \in [1/3, 1] \end{cases}$$

$$\phi(s_2, x) = \begin{cases} s_1, & x \in [0, 1/3) \\ s_3, & x \in [1/3, 1] \end{cases}$$

$$\phi(s_3, x) = \begin{cases} s_2, & x \in [0, 1/3) \\ s_4, & x \in [1/3, 1] \end{cases}$$

$$\phi(s_4, x) = \begin{cases} s_3, & x \in [0, 1/3) \\ s_5, & x \in [1/3, 1] \end{cases}$$

$$\phi(s_5, x) = \begin{cases} s_4, & x \in [0, 1/3) \\ s_5, & x \in [1/3, 1] \end{cases}$$

This function can be formulated in many different ways. For example if we define

$$\phi(s_1, x) = \begin{cases} s_2, & x \in [0, 1/3) \\ s_1, & x \in [1/3, 1] \end{cases}$$

and the rest of the update function stays the same, it would be also a valid update function.

There is of course a lot of other versions of update functions for Markov chains with bigger state spaces and non-zero transition probabilities as we will see for example in Chapter 7. Also, there exists a lot of combinations of these parts, so it is possible to create a function $\phi$ for that the algorithm never terminates (coalescence can not be reached). To avoid this complication we will use only the first version, where the smallest random numbers indicate transition to the first state we can reach and the largest random numbers send the chain to the last reachable state of the state space.

In our implementation we will use a matrix of row cumulative sums of $P$. We will label it $Q = (q_{i,j})$. Every time we need to update the current state $s_i \in S$ of the chain we will go through the $i$-th row of $Q$ and compare the value of the random number used in this update to values $q_{i,j}$. The first $j$ for that the random number is smaller than $q_{i,j}$ we output state $s_j$ as the new state of the Markov chain. This would work for any update function with the property mentioned in the last paragraph.

**Example 1 (cont):** For our example we get matrix $Q$ in form:

$$Q = \begin{bmatrix} 1/3 & 1 & 1 & 1 & 1 \\ 1/3 & 1/3 & 1 & 1 & 1 \\ 0 & 1/3 & 1/3 & 1 & 1 \\ 0 & 0 & 1/3 & 1/3 & 1 \\ 0 & 0 & 0 & 1/3 & 1 \end{bmatrix}$$

And if we run the chain for ten steps we can get for example the trajectory in Figure 3.1.

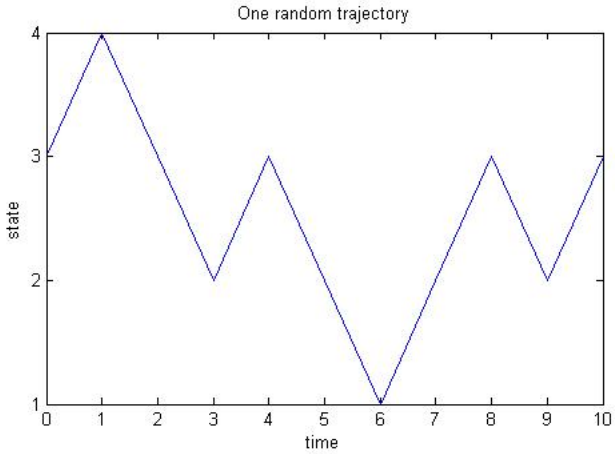


**Figure 3.1**: Ten steps of a random trajectory

# Chapter 4

# Propp-Wilson Algorithm

Propp-Wilson Algorithm also known as Coupling from the Past is an algorithm, that has a sample from its stationary distribution as an output. The input can be for example the transition probability matrix (which we use) or some other characteristic properties of the Markov chain. Propp-Wilson algorithm will continue as long as needed and end, when the stationary distribution has been achieved.

The main difference between Propp-Wilson algorithm and standard Markov chain Monte Carlo methods is, that we do not just simulate one copy of Markov chain at a time, we run several copies at once and we wait, until all the chains get to the same state at the same time (we call this event coalescence of the Markov chain) and then they will all stay in one state, given the way we run the algorithm. This idea of evolving together in the same way until reaching coalescence is also called coupling.

How many copies of the Markov chain do we run and what is the difference among them? We have to run as many copies as is the number of states, the cardinality of the state space. And the trick is to start one copy of the Markov chain from any possible starting state. This means, we will have to run as many chains as is the number of states. We call this coupling. But the trick is not to run these into the future, but from some point in the past to time zero. Because the algorithm determines when to stop on its own, the running time is random.

We use the algorithm to simulate from the probability distribution $\pi$ on a finite state space $S = \{s_1, \ldots, s_k\}$. We create a Markov chain $(X_n, n \in \mathbf{Z})$, that is reversible, irreducible and aperiodic with respect to $\pi$. $P = (p_{i,j})$ is the transition probability matrix of that Markov chain and $\phi : S \times [0,1] \to S$ is the update function.

After that we just need to create a strictly increasing sequence of natural numbers $N_1, N_2, N_3, \ldots$, that will serve as starting times for the coupling (with opposite sign). In the original paper [7] authors of the algorithm considered sequence $N_k =$

$2^{k-1}$. And the last thing we need to prepare before starting the algorithm is a sequence of independent random variables identically distributed on $(0,1)$ that we denote $U_0, U_{-1}, U_{-2}, \ldots$.

**Propp-Wilson Algorithm:**

**1.** We set $m = 1$.

**2.** For every $s \in \{s_1, \ldots, s_k\}$ we simulate a Markov chain starting at state s and running from time $-N_m$ to time 0. To simulate the chain we use update function $\phi$ and sequence of random numbers $U_{-N_m+1}, U_{-N_m+2}, \ldots, U_{-1}, U_0$ – it is very important to reuse the same numbers $U_{-N_{m-1}+1}, \ldots, U_0$ for every earlier starting time and only add the newest numbers $U_{-N_m+1}$ to $U_{m-1}$, otherwise the results would be biased. Also we have to use the same sequence $U_{-N_m+1}, \ldots, U_0$ for all the chain with all possible starting states. If we used different numbers for every copy of the chain, not only the samples would be biased but the termination of the algorithm could be threatened.

**3.** If we get the same final state $s$ for every copy of the Markov chain in step 2., we terminate the algorithm and $s$ is the state we were looking for, otherwise we increment $m$ and repeat step 2.

**Example 1 (cont.)** Now we can return to our example. We use sequence of starting times $N_k = 2^{k-1}$, so first we start from time -1 ($k = 1$; $-N_k = -1$). On the first picture nn Figure 4.1 we see the situation when $U_0$, our only random number, was greater than $1/3$. The values of the chain at 0 are:

$$\phi(s_1, U_0) = s_2$$

$$\phi(s_2, U_0) = s_3$$

$$\phi(s_3, U_0) = s_4$$

$$\phi(s_4, U_0) = s_5$$

$$\phi(s_5, U_0) = s_5$$

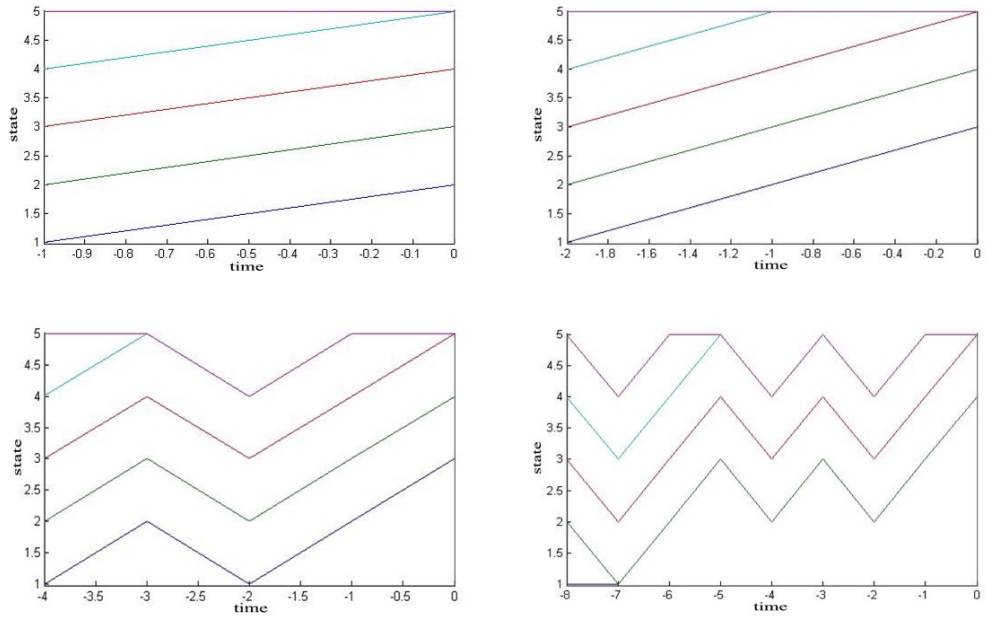


**Figure 4.1**: First four non-coalescent restarts

Figures 4.1 above shows the trajectories in the first four unsuccessful (not coalescent) restarts of the algorithm. The upper left picture starts at time -1 and goes one state up. It is impossible to reach coalescence in one step in this example (the least number of steps needed is 4 so we can get from the lowest state to the highest). This means we can not reach coalescence even in the second case starting at -2. The other two restarts are apparently also unsuccessful. Please note that for example in the lower right picture starting at -8 we can clearly see, that the chains evolve the same from time -4, as do the ones in the previous picture (lower left) because the same random numbers are being reused.

In Figure 4.2 there is the final coalescent restart. It is again visible, that the random numbers are reused.

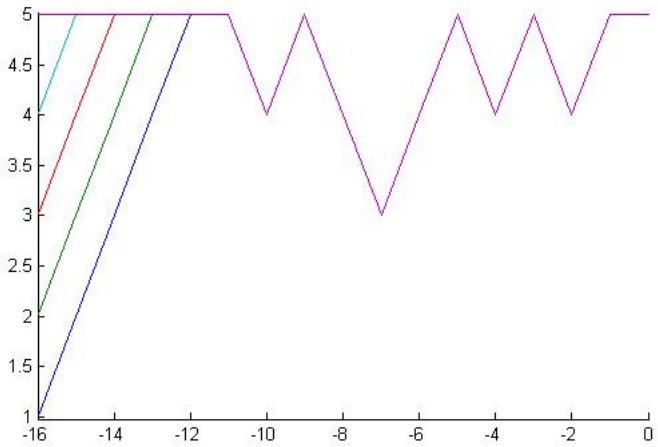


**Figure 4.2**: Coalescent restart

Next Figure 4.3 shows the fit to the theoretically counted stationary distribution (for 100 outputs).

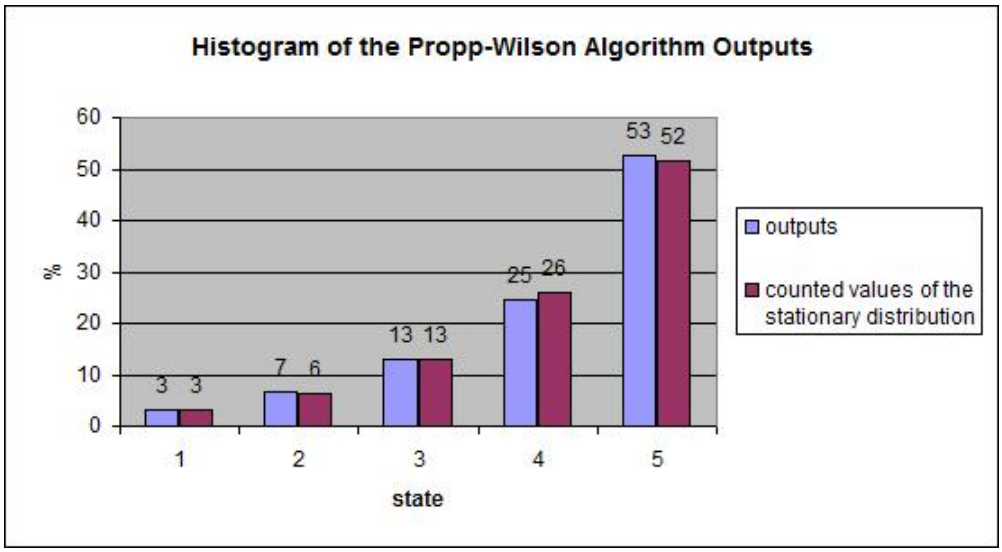


**Figure 4.3**: Histogram

We can see that it is pretty close but there are some differences. So the question is how many time would we have to repeat the algorithm to achieve some given error?

We tried to answer that by an experiment and found out, the numbers are very diverse. If we want the error to be smaller than $10^{-2}$ we can achieve this in less than 400 draws but it can also take us more than 10 000 simulations. We tried it 1000 times and from these the mean number of simulations was 2262. In Figure 4.4 we can se the evolution of the error.

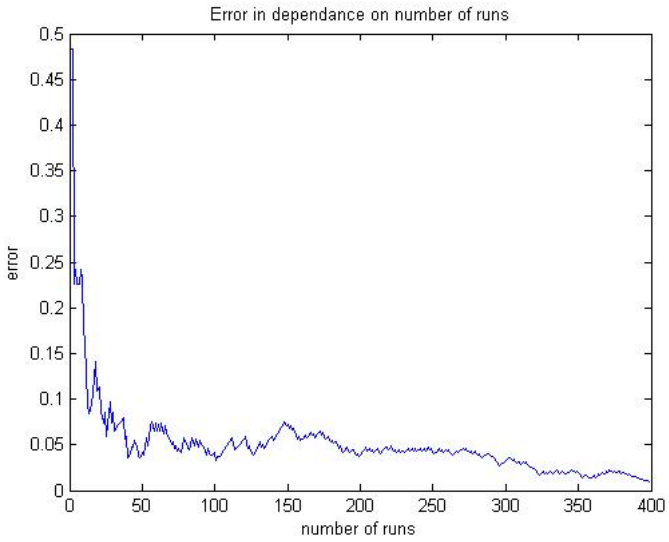


**Figure 4.4**: Error

Let us focus on the choice of the sequence of the starting times. In the last chapter we mentioned that we are using the sequence $2^{k-1}$.

**Theorem 4.1.** *If the algorithm for our Markov chain terminates with probability 1 and we denote the output of the algorithm Y, than for $\forall s_i \in S$ the probability*

$$P(Y = s_i) = \pi_i,$$

*where $\pi = (\pi_i)$ is the stationary distribution.*

For proof see [3].

Can we choose some other sequence of initial times, for example $\{1, 2, 3, 4, ...\}$? It is of course possible but if we assign $N$ to the time from which (with the minus sign) we have to start the simulation for the algorithm to terminate and we count all the steps done in all the restarts until coalescence, it would be $1 + 2 + 3 + 4 + \cdots + N = \frac{N(N+1)}{2}$ for the sequence $\{1, 2, 3, 4...\}$ and only $1 + 2 + 4 + 8 + \cdots + N = 2N - 1$ for sequence $2^{k-1}$. So it is logical to choose the sequence for which the number of steps grows only linearly rather than the one for that it grows like $N^2$.

# Chapter 5

# Sandwiching

We call the Propp-Wilson algorithm a perfect simulation because it produces exactly the corresponding probability distribution we need. This accuracy however comes at a high price. First there is no guarantee the algorithm will ever terminate (the chains will reach coalescence in finite time) for a general function $\phi$. This can be solved by some additional requirements for the function $\phi$.

Second disadvantage of this algorithm is its high computational complexity. It is very demanding to run all the copies of Markov chains with very big state spaces and also to store all the random numbers $U_{-N_m+1}, \ldots, U_0$ when we have a long time to coalescence.

The first problem regarding big state spaces can be solved easily if the Markov chain satisfies the so called sandwich (or monotone) property. It basically means that there exists an ordering on the state space $S = \{s_1, \ldots, s_k\}$, $s_1 < s_2 < \cdots < s_k$, that is preserved also by the update function $\phi$.

For that we do not need to simulate all the chains, corresponding to all possible initial states, in fact only two chains would be enough. Which two? It is obvious that all the chains, starting from a somehow higher state will be always higher than these starting from a lower state and the same applies in the reversed case, when all chains starting lower will never get higher than a chain starting from a higher state. This means that all chains in the middle will be always closed between the chain starting in the lowest state $s_1$ and the chain starting in the highest state $s_k$. When this two chains meet, all the chains between them will be in the same state.

The Markov chain from our example has the ordering, needed for this modification, so in Figure 5.1 we can observe how it works. All the chains evolve according to the same random numbers, so in this case they all go one state up or one state down at a time, because the transition probabilities are the same for all the states and the

update function is designed that way. That means that the chains starting at $s_2, s_3$ and $s_4$ are sandwiched between the chains starting at $s_1$ and $s_5$ (the bolder black lines in the picture). At one point all the trajectories meet and they all continue the same way till time 0.

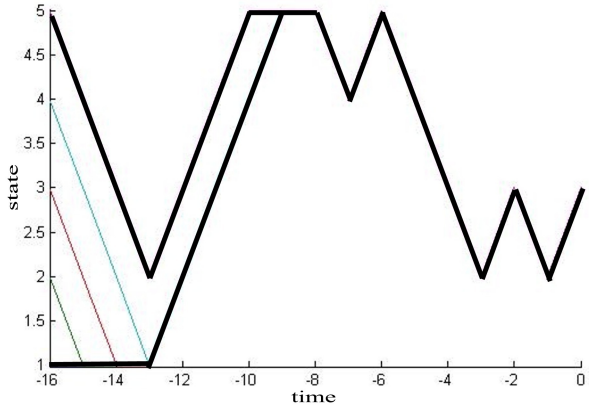


**Figure 5.1**: Trajectories of CFTP and use of sandwiching

Figure 5.2 shows that this modification will not change the fit to the stationary distribution. Also the speed of convergence to this stationary distribution will be obviously the same, because the only thing that changes in this modification is the computational complexity.

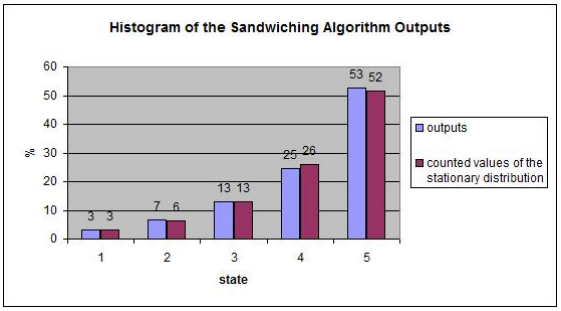


**Figure 5.2**: Histogram

This modification can help a lot with chains with a big state space, but it can be used only for a relatively small number of chains, because not all chains have this sandwich property.

# Chapter 6

# Propp-Wilson Algorithm with Read-Once Randomness

At the beginning of the previous chapter we have mentioned two essential weaknesses of the algorithm and we have shown a way to avoid one of these at least for some special chains. Here we will try to avoid the second problem – the huge amount of random numbers $U_{-N_m+1}, \ldots, U_0$ that has to be stored and reused again and again.

After the Propp-Wilson algorithm was published, many people were trying to deal with this disadvantage of the algorithm. Some ideas concerning adjustments of the seed of a random number generator occurred, but these algorithms were not able to produce unbiased results.

**Algorithm 1** In 2000, David Wilson published a paper [10] where he modified the original Propp-Wilson algorithm by introducing the so called read-once randomness. Later in the literature this modification has been referred to as Wilson's modification, we will use the term Propp-Wilson with read-once randomness. It is not only different from the initial algorithm because we do not reuse the random numbers, but it also runs from zero to the future, not from the past to zero. Here we will use the explanation from [3] where the idea is described by gradually modifying the original algorithm.

Let $N_1, N_2, N_3, \ldots$ be a strictly increasing sequence of random natural numbers independent on $U_0, U_{-1}, U_{-2}, \ldots$. Next we realize that when a chain reaches coalescence at time zero, it would end in the exactly same state when we run it from a random earlier time. By running the coupling to the future algorithm we get a sequence of independent natural random numbers $N_1^*, N_2^*, \ldots$, times needed for coalescence of CTTF. Then we obtain the sequence $N_1, N_2, N_3, \ldots$ by this summing:

$N_1 = N_1^*$

$N_2 = N_1^* + N_2^*$

$N_3 = N_1^* + N_2^* + N_3^*$ etc.

It is also possible to prove (see [3]), that the probability of getting coalescence before or exactly at time zero, with the starting time $-N_1 = -N_1^*$ is at least $1/2$. To show this, imagine, we can start the Propp-Wilson algorithm at time $-N_1$ and continue, until we reach coalescence (even past zero if necessary). Let $M_1$ be the number of steps needed to get coalescence in that case. Then $M_1$ and $N_1^*$ have the same distribution and are independent. From this we get that

$$\mathbf{P}(M_1 \leq N_1^*) = \mathbf{P}(M_1 \geq N_1^*)$$

and we also know that

$$\begin{aligned} \mathbf{P}(M_1 \leq N_1^*) + \mathbf{P}(M_1 \geq N_1^*) &= 1 - \mathbf{P}(M_1 > N_1^*) + 1 - \mathbf{P}(M_1 < N_1^*) \\ &= 2 - (\mathbf{P}(M_1 > N_1^*) + \mathbf{P}(M_1 < N_1^*)) \\ &= 2 - \mathbf{P}(M_1 \neq N_1^*) \\ &\geq 2 - 1 = 1 \end{aligned}$$

If we combine these two pieces of information, we get that $\mathbf{P}(M_1 \leq N_1^*) \geq 1/2$. Of course we would rather work with probability exactly equal to $1/2$. The only problem is the case when $M_1 = N_1^*$. That we can solve by tossing a coin whenever this happens. With probability $1/2$ we get that the chain starting at $N_1$ was *–successful, that means it reached coalescence in less than $M_1$ steps or exactly in $M_1$ steps and the coin toss came up heads. In the opposite case the restart is called *–failing.

Moreover it is possible to prove that when the first restart is not successful, the second one starting from time $-N_2$ will be *–successful (now it has to reach coalescence in time $-N_1$) with conditional probability $1/2$ etc. The number of *–failing restarts $Y$ we have to do before getting a *–successful restart has a geometrical distribution with parameter $1/2$. So when we start in time $-N_{Y+1}$ we get coalescence at zero.

Now we can run two independent copies of coupling to the future algorithm [1] until both copies reach coalescence. We will call this a twin run. We denote the first coalescent copy the winner and the second one the loser. We again toss a coin whey they reach coalescence at the same time. So after the twin run we let the chain evolve in the same way as did the winner of the twin run from time $-N_{Y+1}$ to time $-N_Y$.

[1] It is similar to the coupling to the future algorithm but we start at zero and continue until we reach coalescence. This algorithm, unlike coupling from the past, does not produce the right distribution of outcomes.

Now we generate the geometrically distributed random variable $Y$ and we repeat this procedure with the twin run $Y$ times. The only difference is that now, we will let the chain evolve as the loser and go only for the same amount of step that needed the winner to get coalescence. And in zero we get an unbiased sample from the stationery distribution of our Markov chain.

**Algorithm 2**
However, there are different ways to simulate Propp-Wilson algorithm with read-once randomness and we use a different one in the simulation shown bellow. In this version we fix an arbitrary number of steps $T$ (that has to be at least equal to the smallest number of steps in which we can get from any state $s_i \in S$ to any other state $s_j \in S$ so it would be possible to get coalescence in this time; also it should not be too large so we will not waste time and resources for running the chain longer than necessary) and we run a coupling into the future algorithm for exactly T steps. We repeat this until we get an evolution of the chain that is coalescent in time T (it does not matter if the chain reaches coalescence before T or exactly at T). Then we let our chain evolve this way for the first T steps.

Now we repeat the into the future algorithm for T steps but we do exactly the opposite – if the coupling is coalescent we end the algorithm and will not use this last evolution (leading to coalescence). On the contrary if the restart is not coalescent at T we use this trajectory to evolve our chain for next T steps. One trajectory of the chain is shown in Figure 6.1.

In our example the smallest choice of T possible would be 4 (the number of steps needed to get from $s_1$ to $s_5$ and vice versa). The simulation is run with T=7 because it seems to be a efficient compromise between too small (we have to run the coupling to the future too many times to get one coalescent copy) and too large (just one run of the coupling would be unnecessarily long).

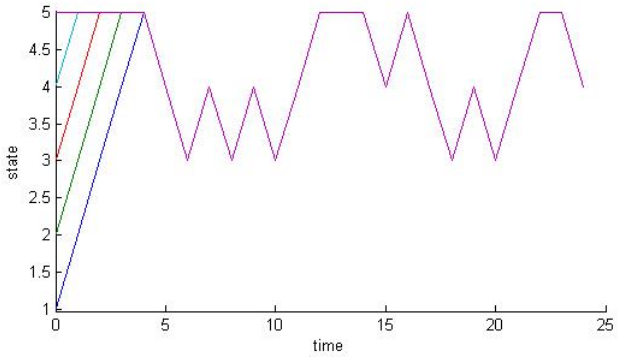


**Figure 6.1**: One random trajectory of read-once randomness algorithm 2 (T=5)

Figure 6.2 shows us the distribution of outcomes produced by this algorithm, compared to the theoretically obtained stationary distribution of the chain.

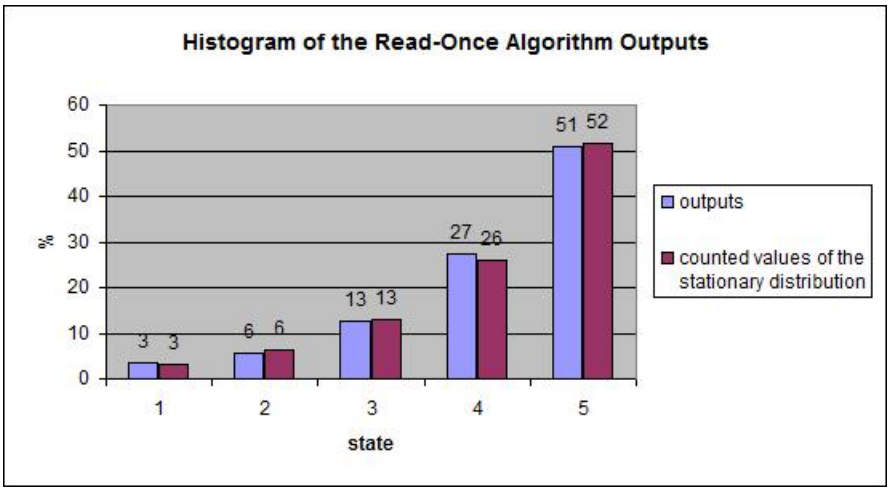


**Figure 6.2**: Histogram

Figure 6.3 shows the evolution of the difference between the computed stationary distribution and the distribution of the simulation outputs until it is equal or less than $10^{-2}$. Again from 1000 repetitions we got the mean number of simulations needed equal to 2246, which is slightly less than in the CFTP algorithm, but on the other hand it took a little more time.

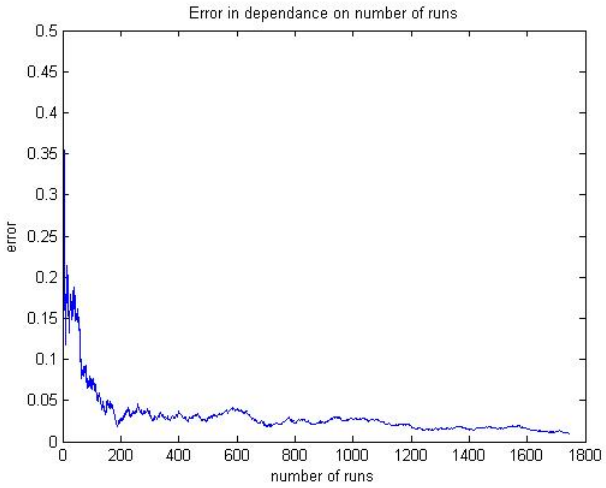


**Figure 6.3**: Error of the Read-Once Randomness modification

# Chapter 7

# Ising Model

Ising model is designed to demonstrate the behaviour of an ferromagnetic material at given temperature. It can be solved in any number of dimensions, we will consider only a two dimensional square example and try to get unbiased samples of possible solutions at different temperatures.

What does the model look like? In our case it is going to be a square lattice. We can describe it as a graph $G = (V, E)$. To every vertex (atom) from $V$ a number $\xi \in \{-1, 1\}$ (spin) should be assigned. How do we assign this number? Randomly with accordance to the probability measure $\pi_{G,\beta}(\xi)$

$$\pi_{G,\beta}(\xi) = \frac{1}{Z_{G,\beta}} exp(-\beta H(\xi)) = \frac{1}{Z_{G,\beta}} exp(\beta \sum_{\langle x,y \rangle \in E} \xi(x)\xi(y))$$

Where $\beta \geq 0$ is the inverse temperature, $H(\xi)$ stands for Hamiltonian or the energy of the model and $Z_{G,\beta}$ is a constant that makes the sum of all the probabilities equal to 1.

$$\begin{aligned} H(\xi) &= -\sum_{\langle x,y \rangle \in E} \xi(x)\xi(y) \\ Z_{G,\beta} &= \sum_{\eta \in \{-1,1\}^V} exp(\beta H(\eta)t) \end{aligned}$$

Now we get to the Propp-Wilson algorithm. We will consider any spin configuration a state of a Markov chain. It is possible to show that there does not exist an ordering on the state space but we can find a minimal (all $-1$s) and maximal (all $+1$s) spin configuration and if we let two copies evolve form the maximal and the minimal state in parallel, the minimal will at any time in the future remain smaller than the maximal, unless they are equal. That equality would mean coalescence. This property is sufficient for us to use the sandwiching algorithm.

The updates will be done by randomly choosing one vertex $x \in V$ at a time and changing or leaving its value. So next thing we need if we want to use the Propp-Wilson algorithm is to be able to tell the next value at this randomly chosen vertex $X_{n+1}(x)$ with knowledge of the previous value $X_n(x)$ and a uniformly distributed random number $U_{n+1}$.

$$X_{n+1}(x) = \begin{cases} +1 & if U_{n+1} < \frac{exp(2\beta(k_+(x,\xi)-k_-(x,\xi)))}{exp(2\beta(k_+(x,\xi)-k_-(x,\xi)))+1} \\ -1 & otherwise, \end{cases}$$

where $k_+(x,\xi)$ is the number of neighbour of $x$ having the spin $+1$ and $k_-(x,\xi)$ is the number of neighbours of $x$ with spin $-1$.

And last but not least we need to deal with the fact that some of the vertices have less than four neighbours. The easiest way is to introduce some additional edges between the vertices on the boundary making the vertexes on the top neighbour to the vertexes on the bottom and the same with left and right side.

In the following figures we can observe the behaviour of the model at different inverse temperatures. Close to zero there is nearly the same amount of black and white (-1s and +1s) as in Figure 7.1 at inverse temperature $\beta$ = 0.05. The closer we get to the so called critical temperature $\beta_c = \frac{1}{2}log(1+\sqrt{2})$ the more the neighbours tend to have the same spin, some groups are formed and one colour (spin) dominates the other. This is only slightly noticeable in Figure 7.2 at $\beta = 0.25$, but evident at the inverse temperature 0.4 in Figure 7.3.

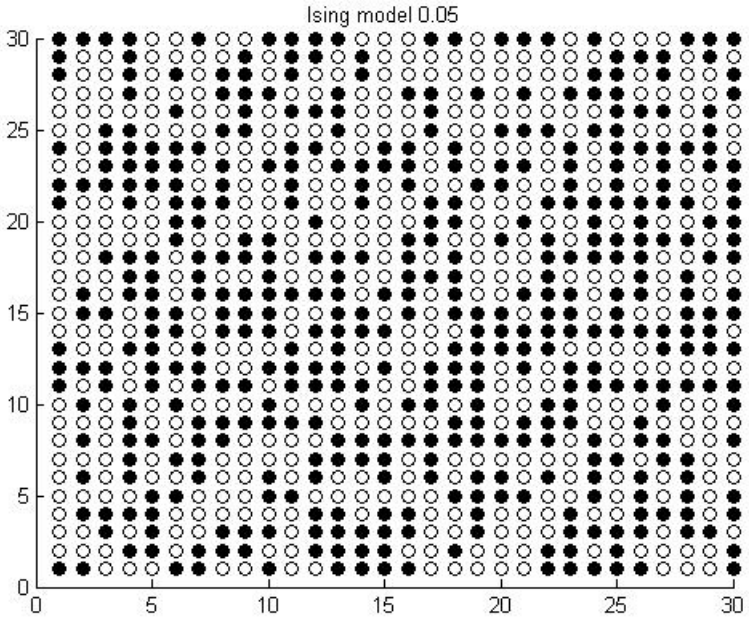


**Figure 7.1**: One randomly generated draw from the stationary distribution of the Ising model at inverse temperature $\beta = 0.05$

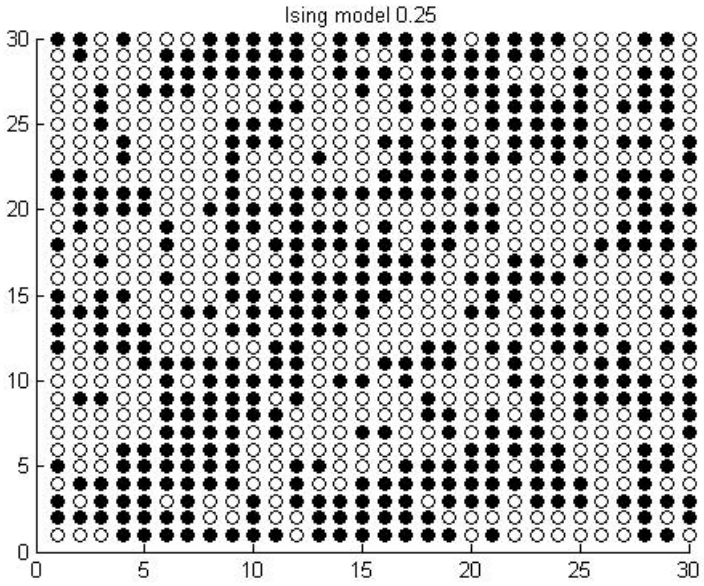


**Figure 7.2**: One randomly generated draw from the stationary distribution of the Ising model at inverse temperature $\beta = 0.25$

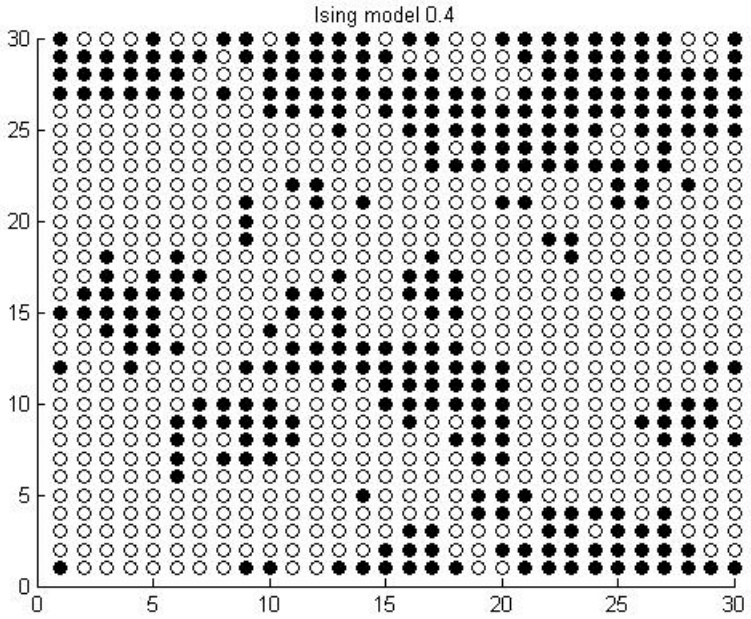


**Figure 7.3**: One randomly generated draw from the stationary distribution of the Ising model at inverse temperature $\beta = 0.4$

# Chapter 8

# Conclusion

The aim of the thesis was to explain and implement the Propp-Wilson algorithm and two of its modifications. In this section we summarize the results for chosen examples.

For the Ising model we used only the monotone sandwiching modification of CFTP, because it is very difficult to simulate the chain from all possible initial states, considering the number of possible states grows as a square of the number of vertices. For a square grid $n \times n$ in will be $n^4$ chains and that is very hard to simulate even for very small grids like $10 \times 10$. Because of that for this type of examples with larger state spaces the sandwiching modification is practically the only applicable alternative. Of course only in case there exists an ordering on the state space allowing us to use this algorithm.

Considering a simple Markov chain in Example 1 we applied all three algorithms: CFTP, sandwiching and read-once randomness CFTP and measured, how many simulations we have to make until we get the stationary distribution with an error smaller than $10^{-2}$. Surprisingly the mean number of simulations needed ranged for all the algorithms between 2240 and 2280 and therefore the accuracy of the algorithms seems very similar. Also the mean computational time it took to get the desired accuracy was not significantly different. Since the chain from Example 1 is in fact very small (only 5 states), the difference in time needed in CFTP and sandwiching is not nearly as big as it would be for example for the Ising model.

So which one of these algorithms should we choose? If the chain has an ordering on the state space, the sandwiching algorithm is the best possibility. If we can not use sandwiching, we can choose any from the others. The read-once randomness CFTP can be implemented using less memory, but that is not usually an issue with modern computers. On the other hand the original CFTP took slightly less time for our example. The accuracy of both algorithms was comparable. So probably the original Propp-Wilson algorithm would be a better option.

# Appendix

## Content of the CD

```
/Thesis/.............includes all the LaTex files for generating
                     the thesis (texts and settings)

/Matlab/.............includes all the Matlab codes used for gene-
                     rating figures or other outputs

/Figures/............includes all the figures used in the thesis

ChrbolkovaBT.pdf....text of the thesis

README.txt..........this file
```

# Bibliography


[1] B. Cipra, *An Introduction to the Ising Model*, Amer. Math. Monthly (1987).

[2] S. Connor, *Coupling: Cutoffs, CFTP and Tameness*, Ph.D. thesis, University of Warwick, 2007.

[3] O. Häggström, *Finite Markov Chains and Algorithmic Applications*, London Mathematical Society Student Texts, Cambridge University Press, Cambridge, 2002.

[4] M. Novotny, *Introduction to the Propp-Wilson Method of Exact Sampling for the Ising Model*, Condensed Matter Physics (1999).

[5] Z. Pawlas, *Metody MCMC*, *http://www.karlin.mff.cuni.cz/pawlas/2007/STP139/mcmc.pdf*.

[6] Z. Prášková and Lachout P., *Základy náhodných procesů*, Karolinum, 1998.

[7] J. Propp and D. Wilson, *Exact Sampling with Coupled Markov Chains and Applications to Statistical Mechanics*, Proceedings of the Seventh International Conference on Random Structures and Algorithms (Atlanta, GA, 1995), 1996.

[8] ———, *Coupling from the Past: a User's Guide*, Microsurveys in discrete probability (Princeton, NJ, 1997), DIMACS Ser. Discrete Math. Theoret. Comput. Sci., Amer. Math. Soc., Providence, RI, 1998.

[9] M. Wennlund, *Methods for Checking Coupling from the Past*, (2011).

[10] D. Wilson, *How to Couple from the Past Using a Read-Once Source of Randomness*, Random Structures Algorithms (2000).