Analysis of the Ising and Kitaev Model Using the CRBM Aided Monte Carlo Method

Master Thesis

in the
Degree Program
"Master of Science"
in Physics

at the Faculty of Physics and Astronomy
of the Ruhr-Universität Bochum

by
Daniel Alcalde Puente

from
Madrid, ES

Bochum 2020

Abstract

Machine learning is becoming widely used in analyzing the thermodynamics properties of many-body condensed matter systems. Restricted Boltzmann Machine (RBM) aided Monte Carlo simulations have sparked interest recently, as they manage to speed up classical Monte Carlo simulations of condensed matter systems. Here we employ the Convolutional Restricted Boltzmann Machine (CRBM) method and show that using it helps to drastically reduce the number of parameters to be learned by taking advantage of translation invariance. By doing this, we show that it is possible to train the CRBM at smaller lattice sizes and apply it to larger lattice sizes, which enables faster sampling at big lattice sizes. To demonstrate the efficiency of CRBM, we apply it not only to the paradigmatic Ising and Kitaev models in two-dimensions but also to an extension of the Kitaev model. Thanks to the ability to sample at larger lattice sizes, we were able to compute critical exponents for the extended Kitaev model, a computation that thus far has not been carried out. To achieve this, two novel correction methods are developed for the CRBM; the Parallel RBM Tempering method and the Local Metropolis RBM method.

Zusammenfassung

Maschinelles Lernen wird zunehmend zur Analyse der thermodynamischen Eigenschaften von Mehrkörpersystemen für kondensierte Materie eingesetzt. Restricted Boltzmann Machine (RBM) unterstützte Monte-Carlo-Simulationen haben in letzter Zeit Interesse geweckt, da sie es ermöglichen, klassische Monte-Carlo-Simulationen von kondensierte Materie zu beschleunigen. Wir verwenden die CRBM-Methode (Convolutional Restricted Boltzmann Machine) und zeigen, dass ihre Verwendung dazu beiträgt, die Anzahl der Parameter, die gelernt werden müssen, durch Ausnutzung von Translationsinvarianz drastisch zu reduzieren. Darüber hinaus zeigen wir, dass es möglich ist, das CRBM bei kleineren Gittergrößen zu trainieren und auf größere Gittergrößen anzuwenden, was eine schnelleres Sampling bei großen Gittergrößen ermöglicht. Um die Effizienz von CRBM zu demonstrieren, wenden wir es nicht nur auf die zweidimensionalen paradigmatischen Ising- und Kitaev-Modelle an, sondern auch auf eine Erweiterung des Kitaev-Modells. Aufgrund der Möglichkeit, bei größeren Gittergrößen zu samplen, konnten wir kritische Exponenten für das erweiterte Kitaev-Modell berechnen, eine Berechnung, die bisher noch nicht durchgeführt wurde. Um dies zu erreichen, wurden zwei neuartige Korrekturmethoden für die CRBM entwickelt; die Parallele RBM Temperierungsmethode und die Lokale Metropolis RBM Methode.

Preface

It makes me proud to present my master thesis, which is the result of my research at Ruhr-Universität Bochum from 2018 to 2020. I investigated how to accelerate Monte Carlo simulations of condensed matter models using machine learning. Due to my preexisting interest in physics and machine learning, I suggested this topic to my supervisor Prof. I. Eremin and was glad when he accepted it for a master's thesis, as it lies at the nexus of two of my passions.

First of all, I sincerely thank my parents, Isabel and Pablo, on whose continuing support I am always able to build on. The journey of this thesis has been eventful. From sickness to health. From love to marriage. It has been a remarkable time. I deeply thank Salma, my wife, for being supportive in the hard moments and being part of the good ones. And lastly from life to death, to the memory of my grandfather.

I would like to thank my supervisor Prof. I. Eremin for his guidance and Prof. N. B. Perkins (University of Montana) for the stimulating discussions. I wish to express my sincere thanks to Noemi Roggero, Salma Alsayyad, and Yasmine Alsayyad for proofreading this work.

Daniel Alcalde Puente

Daniel.AlcaldePuente@rub.de

14th May 2020

Table of Content

Abbreviations

MC	Monte Carlo
MCMC	Markov chain Monte Carlo
PT	parallel tempering
RBM	Restricted Boltzmann Machine
CRBM	Convolutional Restricted Boltzmann Machine
LCRBM	Local Convolutional Restricted Boltzmann Machine
PRBM	Parallel RBM Tempering Update
LMRBM	Local Metropolis RBM Update
MRBM	Metropolis RBM Update

CHAPTER 1

Introduction

The main aim of statistical mechanics is to compute different properties of condensed matter systems. These computations should not be too difficult since interactions between particles are linear. This means that a two-particle system might be understood with high precision. While the exact dynamics are much more difficult to calculate in systems where hundreds of particles interact, the properties of such systems can still be predicted. To put this in context, there are around 10^{22} particles in one liter of gas, and while the exact dynamics are impossible to compute, describing the properties of this system is not a challenge. Statistical mechanics reformulates the problem so that we are no longer interested in the dynamics of the system but in the probability that the system is in a given state. The probability of the system being in a given state $x \in X$ is governed by the Boltzmann distribution $P(x) \propto e^{\frac{-E(x)}{k_{\mathrm{B}}T}}$, where $E(x)$ is the energy of the state, T is the temperature, X is the state space and k_{B} is the Boltzmann constant. The same applies for a quantum mechanical systems: $P(x) \propto \sum_{x\in X} \langle x| \, e^{-\frac{\hat{H}}{k_{\mathrm{B}}T}} \, |x\rangle$.
An observable f can be computed by a sum:

$$\langle f \rangle = \sum_{x \in X} f(x) P(x). \tag{1.1}$$

In practice, this sum can only be computed for small state spaces X.

In this thesis, the thermodynamics of two spin models, the Ising and the Kitaev model, will be analyzed. In the ferromagnetic Ising model, a state is a configuration of spins that interact with their nearest neighbor on a lattice. In this model, each spin is classical and can be in one of two states, up or down, which are encoded as ± 1. In the Kitaev

model, spins interact in a honeycomb lattice and fractionalize into Majorana fermions and a topological quantum spin liquid emerges in the ground state. It is possible to reformulate the Hamiltonian into classical gauge fields that can take values ± 1 so that classical computations are possible.

If states are two-dimensional and each component of these states is binary, then X is the space of all L long two-dimensional tensors x, with each component x_{ij} being a statistical variable that can take one of two possible states, ± 1, $X = \{-1, +1\}^{L^2}$. With lattice size $L = 10$, the system has a hundred particles. It is unfeasible to compute the sum in Eq. 1.1, as it would have $2^{L^2} \approx 10^{30}$ terms, making an alternative necessary.

In an Monte Carlo (MC) simulation, the sum $\sum_{x \in X} f(x)P(x)$ is reinterpreted to compute the expectation value $\langle f(x) \rangle_{x \in P(x)}$. M samples are drawn from the probability distribution $P(x)$ and the expectation value is calculated as $\langle f \rangle = \frac{1}{M} \sum_i^M f(x^{(i)})$. Sampling from a given probability distribution is not a trivial task. Different Monte Carlo sampling techniques try to tackle this difficulty. Commonly, Markov chain Monte Carlo (MCMC) techniques are used to sample from a probability distribution. Instead of drawing a new sample x, from $P(x)$ every time, a chain of states is employed, such that the n-th state $x^{(n)}$ is generated from the $(n-1)$-th state, $x^{(n-1)}$. Practically, this means that the first state $x^{(1)}$ is chosen randomly, and the second state $x^{(2)}$ is generated from the first state using a conditional distribution $P(x^{(2)}|x^{(1)})$ (see Eq. 1.2).

$$x^{(1)} \xrightarrow{P(x^{(2)}|x^{(1)})} \dots \xrightarrow{P(x^{(n)}|x^{(n-1)})} x^{(n)} \tag{1.2}$$

If the conditional probability distribution $P(x'|x)$ is chosen properly (see Sec. 3.1.1), then the states obtained from the chain will be distributed according to the probability distribution.

The change of the state $x^{(i)}$ in each update can be small or large depending on the conditional distribution $P(x'|x)$. Common MCMC algorithms such as the Metropolis-Hasting algorithm (see Sec. 3.1.2) have a crucial shortcoming. They have a large autocorrelation time τ. This means that it takes a lot of samples until $x^{(1)}$ and $x^{(n)}$ are uncorrelated. This is the case, as for most probability distributions the only Metropolis update that is available does not change the state much in each update (local Metropolis). At most, it will flip one component of x in each MCMC update.

In this thesis, a Restricted Boltzmann Machine (RBM) (see Chap. 3.3), which is a model that originated in machine learning, is employed to try to find a Markov chain with a smaller autocorrelation time. The RBM is a probabilistic model with two main features. First, it is possible to sample states from the model's probability distribution through a known conditional probability distribution that performs global updates. Second, the unnormalized probability distribution $\hat{P}_{\mathrm{RBM}}(x;W)$ is known. This distribution depends on a matrix W. Accordingly, different matrices W define different probability distributions. One needs to choose a W such that $P_{\mathrm{RBM}}(x;W) \approx P_{\mathrm{target}}(x) \quad \forall x$ and then use the global updates from the RBM to sample. How one finds a W that fulfills the previous equation will be further explored in Sec. 3.3.4. For now, this process will be referred to as training or learning.

To summarize, two distinct steps are performed. First, the probability distribution $P_{\mathrm{RBM}}(x)$ for the RBM is trained to be close to the physical distribution. Second, the conditional distribution of the RBM is used to sample, as its Markov chain performs global updates, which leads to a smaller autocorrelation time τ. W can never be chosen perfectly. During sampling, different sampling strategies are employed to correct for discrepancies between P_{RBM} and P_{target} (see Sec. 3.4.2). Two new correction strategies, Parallel RBM Tempering Update (PRBM) and Local Metropolis RBM Update (LMRBM) tempering are introduced.

For the Ising and Falicov-Kimball models, Ref. [1, 2] succeeded to train an RBM such that it would represent their corresponding physical distribution. This was done for small lattice sizes $L = 8$ and $L = 10$. The training is slow and major difficulties arise when training the RBM for larger lattice sizes. There is a trick on how RBMs can be applied at larger lattice sizes. This is achieved by exploiting the property that physical models are often translation invariant. The matrix W is to be chosen such that the probability distribution $P_{\mathrm{RBM}}(x;W)$ is translation invariant. This is done by using a Convolutional Restricted Boltzmann Machine (CRBM). CRBMs are not only translation invariant, but they have an interesting property. Once parameters are set, the lattice size can be increased. Thus, for a nearest-neighbor interaction as the one in the Ising model, the parameters of the CRBM can be learned with a small lattice size $L = 3$ and will generalize for larger lattice sizes. This will be further explored in Sec. 3.3.2. This change makes it possible to train the CRBM for the Ising model in seconds rather than hours. As proof of concept, the performance of the CRBM is compared to the local Metropolis method for the Ising model in Sec. 4.2. Not only can a CRBM be applied at bigger lattice sizes, but also, due to the

parallel nature of convolutions when dealing with larger lattice sizes, the process can be parallelized by the GPU.

This thesis answers the three main questions:

- Can a CRBM sample accurately on lattice sizes it was not trained on?
- Is the CRBM method faster than Metropolis?
- Can the method be used to gain new insights into physical systems?

To answer the first question the CRBM is tested on the Ising model. The Ising model is just a proof of concept, as there are much faster algorithms than the CRBM to perform Metropolis. This is not the case for more complex interactions.

To answer the second and third questions, the Kitaev model, a quantum spin liquid, is simulated with the help of the CRBM (see Chap. 5.1). We can perform Monte Carlo simulations on larger lattice sizes than found in the literature. This result is still unsatisfactory, as the normal Kitaev model has no phase transition. As a result, the specific heat almost does not change when increasing the lattice size. No new physical knowledge was gained.

This is why an extension for the Kitaev model [3] is considered (see Chap. 6). This model has a phase transition and thanks to the CRBM we can simulate larger lattices. In Ref. [3], it was only possible to perform simulations of not more than $L = 18$, which is also the upper limit we were able to achieve with Metropolis. With the CRBM method, simulations with a lattice size up to $L = 36$ could be performed. This makes it possible to apply finite-size scaling to compute critical exponents.

CHAPTER 2

Literature Review

In recent years, machine learning and specifically Restricted Boltzmann Machines (RBMs) have been used in the field of many-body quantum systems to explicitly parametrize a probability distribution function of a quantum many-body state [4]. For example, RBMs have been found to be a powerful tool to obtain the ground state of quantum models through variational Monte Carlo (MC) [5] and to reconstruct quantum states from a set of qubit measurements [6, 7]. Neural networks have been shown to be useful in the classification of phases of matter in MC simulations [8], which has awakened interest in the classification of topological and nematic phases [9–11] since finding a suitable order parameter for phase transitions can be challenging. Increasing the resolution of already sampled states [12] has been another area of study.

Another promising avenue in the application of the machine learning techniques is the sizeable speeding up of the quantum MC simulations obtained via the so-called self-learning Monte Carlo (SLMC) method [13–17] applied to the models of interacting fermionic systems where the Trotter decomposition can be employed. In this method, the effective energy is cheap to compute and is mostly composed of two-particle interactions, which enable cluster updates. The strength of these interactions is then learned by applying the linear regression method. For more complex systems, neural networks are employed to model the effective energy [18, 19]. This, however, makes cluster updates very hard to realize. At the same time, RBMs can be used as an alternative to SLMC for models where no Trotter decomposition is needed, as the former can model more complex interactions than the original SLMC (learned by linear regression), and they are faster to sample from than SLMC with neural networks, due to Gibbs sampling.

For the Ising and Falicov-Kimball models, [1, 2] succeeded to train an RBM such that it would represent the physical distribution. This was done for small lattice sizes $L = 8$ and $L = 10$. Training is slow and major difficulties arise when training the RBM for larger lattice sizes.

This thesis introduces new concepts in three areas. First, in the architecture of the model, exploiting the CRBM translational invariance to be able to train at small lattices sizes and sample at bigger. Second, a modified training scheme is introduced. Third, two novel methods for correcting the states sampled from RBMs, Parallel RBM Tempering, and Local Metropolis RBM, are presented.

CHAPTER 3

Methods

3.1. Markov Chain Monte Carlo

Markov chains enable the sampling of a probability distribution $P(x) = \frac{e^{-\beta E(x)}}{Z}$, where $Z = \sum_x e^{-\beta E(x)}$ is the partition function, even if Z is unknown. A Markov chain converges to a given probability distribution when certain conditions are met.

3.1.1. Necessary Conditions for a Markov Chain

The difficulty to define a Markov chain that makes it possible to efficiently sample from a probability distribution is threefold. First, the resulting probability distribution $P(x)$ of the Markov process has to match the desired distribution. Second, the MC has to be ergodic, and third but not less important, a technical constraint: the conditional distribution $P(x'|x)$ needs to be easy to sample from.

A Markov chain is ergodic if all states are reachable by the MC. If an MC is reversible, aperiodic, and positive recurrent, then it is also ergodic.

A Markov chain is reversible if $P(x|x')P(x') = P(x'|x)P(x)$. To understand why, the probability $P(x, x')$ is needed. $P(x, x')$ describes the process of x being sampled from $P(x)$ and x' being sampled from $P(x'|x)$. This yields $P(x, x') = P(x'|x)P(x)$. Reversibility is

achieved if a pair of states are equally likely regardless of the order:

$$\begin{aligned} P(x', x) &= P(x, x') \\ P(x|x')P(x') &= P(x'|x)P(x) \end{aligned} \tag{3.1}$$

This is often also called the detailed balance condition.

3.1.2. Metropolis-Hasting Algorithm

The Metropolis algorithm is one of the easiest ways to sample from a given distribution using Markov chains. In the Metropolis algorithm, the conditional distribution is divided into two: First, a state x' is drawn from a proposal distribution $g(x'|x)$. Second, this state is accepted with probability $A(x', x)$ so that $P(x'|x) = A(x', x)g(x'|x)$. This equation can be inserted into the detailed balance condition:

$$\begin{aligned} P(x|x')P(x') &= P(x'|x)P(x) \\ A(x', x)g(x'|x)P(x') &= A(x, x')g(x|x')P(x) \\ \frac{A(x', x)}{A(x, x')} &= \frac{P(x')g(x|x')}{P(x)g(x'|x)}. \end{aligned}$$

The Metropolis choice for the acceptance rate is $A(x', x) = \min\left(1, \frac{P(x')g(x|x')}{P(x)g(x'|x)}\right)$. Given a probability distribution $P(x) = \frac{e^{-\beta E(x)}}{Z}$, the acceptance rate is:

$$A(x', x) = \min\left(1, e^{-\beta(E(x')-E(x))}\frac{g(x|x')}{g(x'|x)}\right). \tag{3.2}$$

To have a nonvanishing acceptance rate, the new state should have a similar energy to the old one. This can be achieved by either modifying x slightly or exploiting symmetries in $E(x)$.

The easiest choice for $g(x'|x)$ is to randomly flip a component of x. This is called the local

Metropolis algorithm. In this case $\frac{g(x'|x)}{g(x|x')} = 1$ and so:

$$\begin{aligned} A(x', x) &= \min\left(1, \frac{P(x')}{P(x)}\right) \\ &= \min\left(1, e^{-\beta(E(x')-E(x))}\right) \\ &= \min\left(1, e^{-\beta\Delta E(x',x)}\right) \end{aligned} \tag{3.3}$$

If the new state has lower energy than the previous one it is automatically accepted, else it is accepted with the probability $A(x', x)$. In the case of the Ising model, a better $g(x'|x)$ is possible using cluster algorithms. These, instead of just flipping one vector component, flip clusters of spins. However, for most interactions, no such technique exists. Note that the autocorrelation time for a Metropolis update scales with $\tau \approx N_p^c$, where N_p is the number of lattice sites or particle number, and c typically lies between 1 and 2 depending on model and temperature.

3.2. Phase Transitions

A phase transition describes a process in which there is a transition, between different states of matter. Examples for phase transitions can be found all around in nature. From the freezing and boiling of liquids to the magnetization of iron. Phase transitions can happen when parameters of a physical system are altered. Most notably a phase transition might occur when a system is heated or cooled. The temperature at which the transition occurs is called critical temperature T_c. Phase transitions are of interest, as the properties of the physical system are different in the ordered phase ($T < T_c$) than in the disordered phase ($T > T_c$).

In many-body physics, both the concept and the study of phase transitions have been of great importance. Several mathematical tools have been developed to describe them [20]. In Landau theory, each phase transition is associated with an order parameter ψ, which corresponds to a symmetry of a given system. Above T_c the order parameter will be 0, but when the temperature drops under T_c the symmetry is broken and the order parameter is

no longer 0.

$$|\psi| = 0 \quad \text{when} \quad T > T_c$$
$$|\psi| > 0 \quad \text{when} \quad T < T_c$$

In magnets, for example, the order parameter is the magnetization m, which turns to 0 when the system is heated over T_c. Here all the spins go from pointing in the same direction (ordered phase) to pointing in different directions (disordered phase). A more rigorous theoretical treatment requires the renormalization group to describe how the microscopic dynamics of the Hamiltonian induce the macroscopic effect of a phase transition.

It can be derived that close to T_c thermodynamic, constants follow power laws:

$$C_v \propto |T - T_c|^{-\alpha} \tag{3.4}$$
$$\psi \propto \begin{cases} (T - T_c)^{\beta} & T > T_c \\ 0 & T < T_c \end{cases} \tag{3.5}$$
$$\chi \propto |T - T_c|^{-\gamma} \tag{3.6}$$
$$\xi \propto |T - T_c|^{-\nu} \tag{3.7}$$

where C_v is the specific heat, ψ is the order parameter, χ is the magnetic susceptibility, and ξ is the correlation length. α,β, γ, and ν are the critical exponents that characterize the transition. Interestingly, critical exponents are found to be universal. Models of the same universality class have the same critical exponents, independently of the microscopic physics, which make them a wonderful tool to characterize physical models.

Note that if $\alpha, \gamma, \nu > 0$ then a singularity is encountered at criticality ($T = T_c$). Of course, in a finite system thermodynamic, constants can not be singular. Therefore, the correlation length ξ can not be larger than the lattice size L. This means that at finite lattice sizes the critical temperature is lattice size dependent $T_c(L)$. Based on this it can be derived (see Ref. [21]):

$$C_v(T, L) = L^{\frac{\alpha}{\nu}} \bar{C}(L^{\frac{1}{\nu}}(T - T_c)). \tag{3.8}$$

The specific heat at finite lattice sizes scales with $L^{\frac{\alpha}{\nu}}$, but it reaches its maximum when

$\bar{C}(x)$ is maximal at x_0, and so:

$$L^{\frac{1}{\nu}}(T_c(L) - T_c) = x_0$$
$$C_v^{\max}(L) = L^{\frac{\alpha}{\nu}}\bar{C}(x_0)$$

which leads to:

$$T_c(L) - T_c \propto L^{-\frac{1}{\nu}} \tag{3.9}$$
$$C_v^{\max}(L) \propto L^{\frac{\alpha}{\nu}}. \tag{3.10}$$

Eq. 3.9 and Eq. 3.10 will be used in Sec. 6.9 to compute the critical exponents of the extended Kitaev model.

3.3. Restricted Boltzmann Machine

3.3.1. Fully Connected RBM

A Restricted Boltzmann Machine is a probabilistic model that can learn an approximated probability distribution and then sample from it using Gibbs sampling.

The model has two distinct groups of statistical variables. The visible variables v and the hidden variables h. In the one-dimensional case, v is a vector of length L, where each component takes either the value 0 or 1, $v \in \{0,1\}^L$. The same goes for $h \in \{0,1\}^M$. The probability distribution over both visible and hidden variables is $P_{\mathrm{RBM}}(x,h;W)$. Summing over the hidden variables, the probability distribution over the visible units $P_{\mathrm{RBM}}(x;W) = \sum_h P_{\mathrm{RBM}}(x,h;W)$ is obtained. This will approximate the target distribution, while the hidden units just act as an instrument to make Gibbs sampling possible. Visible variables are not connected to each other, neither are the hidden variables. Only visible and hidden variables are connected through a matrix W. The probability distribution over both visible and hidden variables for the fully connected RBM is:

$$P_{\mathrm{RBM}}(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{3.11}$$
$$E(v,h) = -\sum_i h^i_{\mathrm{bias}} h^i - \sum_j v^j_{\mathrm{bias}} v^j - \sum_{ij} h^i W_{ij} v^j \tag{3.12}$$

where Z is the normalization constant and W, v_{bias}, and h_{bias} are model parameters.

W_{ij} is the weight matrix component that connects hidden unit h_j and visible unit v_i and has dimensions $M \times L$. There are also bias weights v_{bias}(L long vector) for the visible units and h_{bias}(M long vector) for the hidden units.

The probability over the visible units is:

$$P_{\text{RBM}}(v) = \sum_{h \in \{0,1\}^M} P(v,h) = \frac{e^{-F(v)}}{Z} \tag{3.13}$$

$$F(v) = -\sum_j v^j_{\text{bias}} v^j - \sum_i \log\Big(1 + e^{h^i_{\text{bias}} + \sum_j W_{ij} v^j}\Big). \tag{3.14}$$

In the following text $P_{\text{RBM}}(v)$, $P_{\text{RBM}}(v; W)$ and $P_{\text{RBM}}(v; W, v_{\text{bias}}, h_{\text{bias}})$ will be used interchangeably depending on the context.

The conditional probabilities $P(v|h) = \frac{P(v,h)}{P(h)}$ used in Gibbs sampling are:

$$P(h_i = 1|v) = \frac{e^{h^i_{\text{bias}} + \sum_j W_{ij} v_j}}{1 + e^{h^i_{\text{bias}} + \sum_j W_{ij} v_j}} \tag{3.15}$$

$$= \sigma(h^i_{\text{bias}} + \sum_j W_{ij} v^j) \tag{3.16}$$

$$P(v_j = 1|h) = \sigma(v^j_{\text{bias}} + \sum_i h_i W_{ij}). \tag{3.17}$$

The conditional probabilities describe a binomial distribution that depends either on h or on v. The binomial distribution is easy to sample from. To update a state $v^{(1)}$, a bipartite Markov chain is constructed: First, a sample $h^{(1)}$ is drawn from the conditional distribution $P(h^{(1)}|v^{(1)})$, and second, $v^{(2)}$ is drawn from $P(v^{(2)}|h^{(1)})$:

$$v^{(1)} \xrightarrow{P(h^{(1)}|v^{(1)})} h^{(1)} \xrightarrow{P(v^{(2)}|h^{(1)})} v^{(2)} \tag{3.18}$$

Comparing it to a normal Markov chain, the only difference is the middle step where h is sampled. The RBM can be reformulated so that it is equivalent to a normal Markov chain:

$$P(v^{(2)}|v^{(1)}) = \sum_h P(v^{(2)}|h) P(h|v^{(1)}).$$

Note that this quantity can not be computed analytically. It can be proven that this conditional probability fulfills the detailed balance equation (see App. A.1). This means that the RBM describes an ergodic Markov chain.

3.3.2. Convolutional RBM

A Convolutional Restricted Boltzmann Machine (CRBM) [22] is a special type of RBM. It has been used in machine learning for image generation and recognition. The only difference between an RBM and a CRBM is that the weight parameters are fixed in such a way that the free energy $F(v; W)$ is translation invariant. This is achieved by connecting the visible and the hidden units using a convolution. Then the probability distribution $P_{\mathrm{RBM}}(v) = \frac{e^{-F(v)}}{Z_{\mathrm{RBM}}}$ is invariant under translation. The same CRBM can be applied at different lattice sizes, this makes it possible to learn the probability distribution using states v of any lattice size bigger than the interaction length. Small lattice sizes can be used for model training, which in the case of the Ising model can reduce training time from hours to seconds. The CRBM can then be used to generate samples of larger lattice sizes. This is not possible with a fully connected RBM. With a fully connected RBM the lattice size can't be changed. To make the boundaries of the lattice periodic, a wrap around condition is introduced for the convolution.

Since we are interested in a two-dimensional model, a 2D CRBM is presented. The visible units now have two-dimensions v_{ij}, and the hidden units have three, h^k_{ij}, where i and j correspond to the x- and y-direction and k is an extra dimension so that there is more than one hidden unit per visible unit. The energy function as given in Ref. [22] is:

$$\begin{aligned} E(v,h) = &- \sum_{k=1}^{K} \sum_{i,j=1}^{L} \sum_{r,s=1}^{L_W} h_{ij} W^k_{rs} v_{(i+r-1, j+s-1) \bmod L} \\ &- \sum_{k=1}^{K} h^k_{\mathrm{bias}} \sum_{i,j=1}^{L} h^k_{ij} - v_{\mathrm{bias}} \sum_{i,j=1}^{L} v_{ij} \end{aligned} \tag{3.19}$$

Note that the visible units have dimension $L \times L$ and the hidden ones have dimension $K \times L \times L$. They are connected through K convolutional kernels W^k of dimensions $L_W \times L_W$. The periodic boundary condition is enforced through the mod operator in Eq. 3.19. The CRBM has $KL_W^2 + K + 1$ parameters.

The Ising model can be described by a two-particle interaction $L_W = 2$ and it is found empirically that $K = 2$ kernels can learn the model. The model only has to learn 11 parameters, which stands in striking contrast with the $ML^2 + M + L^2$ parameters that need to be learned by a fully connected RBM. For more complex models, like the Kitaev model, more and larger kernels will be needed.

To simplify the notation we define the symbol $*$ as a wrap around convolution between lattice and kernel. The symbol $\bullet$ is defined as the dot product between lattices. With this change, the equations read:

$$\begin{aligned} E(v,h) = &- \sum_k h^k \bullet (W^k * v) \\ &- \sum_k h^k_{\text{bias}} \sum_{i,j} h^k_{ij} - v_{\text{bias}} \sum_{i,j} v_{ij} \end{aligned} \tag{3.20}$$

and the free energy is:

$$F(v) = -v_{\text{bias}} \sum_{k,i,j} v_{ij} - \sum_{i,j} \log\Big(1 + e^{(v*W^k)_{ij} + h^k_{\text{bias}}}\Big). \tag{3.21}$$

Each element in $(v * W^k)_{ij}$ represents the interaction between neighboring points in the lattice, connected by the convolutional kernel. So the free energy F is translation invariant. Note that if the lattice size is increased, the new lattice points will still interact with their neighbors in the same way. This is represented for the 1D case in Fig. 3.1.

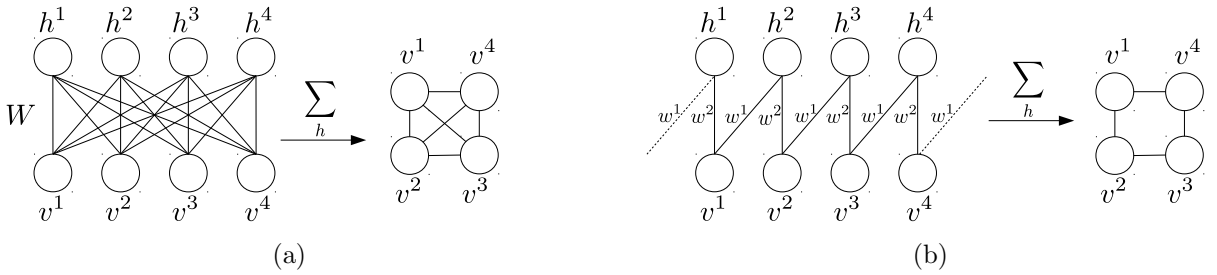


Figure 3.1.: The one-dimensional representation of fully connected RBM and CRBM. (a) (left) Connection between visible and hidden units representing the energy of the fully connected RBM. (right) Effective connections between visible units after summing over h, representing the free energy (see Eq. 3.21). All the visible units interact with each other in the free energy. (b) Only nearest neighbors are connected through h in the CRBM and so only nearest neighbors interact.

As with fully connected RBMs, block Gibbs sampling can be performed. The conditional

distributions are:

$$P(h_{ij}^k = 1|v) = \sigma((W^k * v)_{ij} + h_{\text{bias}}^k)) \tag{3.22}$$

$$P(v_{ij} = 1|h) = \sigma((\sum_k \bar{W}^k * h^k)_{ij} + v_{\text{bias}})). \tag{3.23}$$

$\bar{W}$ indicates that the kernel needs to be spatially inverted before the convolution is performed.

3.3.3. Locally Connected RBM

The Local Convolutional Restricted Boltzmann Machine (LCRBM) is a small modification to the CRBM that can deal with open boundary conditions. It is similar to the CRBM in that in the middle of the lattice translation invariance (weights are shared) is assumed. The two models are however different in that for the LCRBM new unshared kernels are introduced at the edges (Fig. 3.2). If a periodic CRBM has already been trained, the middle kernel can be used, and only the boundaries will have to be trained.

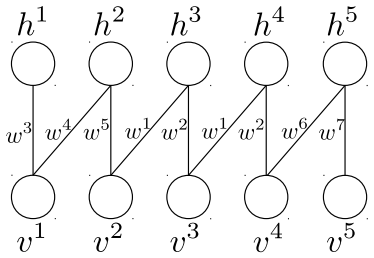


Figure 3.2.: Translation invariant interactions only in the middle, not at the boundaries.

Note that for the 2D Kitaev model only the boundary conditions in one-dimension are chosen to be open. This means that for each lattice point at the boundary a different convolution kernel is used. If a boundary size b is chosen, there will be b different kernels for the right boundary, 1 kernel for the middle region, and b different kernels at the left boundary. In total $2b + 1$ different kernels are needed. In the direction with periodic boundary conditions, the kernels are shared. Note that our implementation for the LCRBM is approximately 10 times slower than the one for the periodic CRBM.

3.3.4. Training

The physical model with energy $E_{\text{phys}}(x)$ will be approximated by the free energy $F_{\text{RBM}}(x)$. Note that states in the physical model are encoded as -1 and $+1$, but that the RBM encodes its states as 0 and 1. While sampling and training, a conversion between the two is necessary.

The RBM can be trained both unsupervised [1] or supervised [2]. When training unsupervised, the fact that we have access to the energy of the target distribution is not used. This explains why it is slower to train unsupervised and faster to train supervised. Training is done supervised similarly to Ref. [2] but with some modifications. We train by minimizing the loss:

$$\text{loss}(W) = \frac{1}{M}\sum_{i=1}^{M}(E_{\text{phys}}(x^{(i)})\beta - F_{\text{RBM}}(x^{(i)};W) - C(W))^2 \tag{3.24}$$

where M is the batch size, β is the inverse temperature and C is a value that can be chosen freely, as the probability function is invariant under the addition of a constant to the energy. The $x^{(i)}$ are the states, which will be trained. C is chosen such that the loss is minimal:

$$C(W) = \frac{1}{M}\sum_{i}^{M} E_{\text{phys}}(x^{(i)})\beta - F_{\text{RBM}}(x^{(i)};W). \tag{3.25}$$

Note that if a W_{trained} is found where $\text{loss}(W_{\text{trained}}) = 0$ then $F_{\text{RBM}}(x;W_{\text{trained}}) = E_{\text{phys}}(x)\beta - C \ \forall x$, and so:

$$\begin{aligned} P_{\text{RBM}}(x;W_{\text{trained}}) &= \frac{\exp\{-F(x;W_{\text{trained}})\}}{\sum_x \exp\{-F(x;W_{\text{trained}})\}} \\ &= \frac{\exp\{-E(x)\beta + C\}}{\sum_x \exp\{-E(x)\beta + C\}} \\ &= P_{\text{phys}}(x) \end{aligned}$$

The only question that remains is what states $x^{(i)}$ do we train with. Ref. [2] has proposed to train with states sampled from the physical distribution. The problem this loss poses is that if a state x_{unlikely} has a small realization probability $P_{\text{phys}}(x_{\text{unlikely}}) \ll 1$, then the RBM will not learn the state x_{unlikely}, as the RBM will never have seen it. This leaves low

probability regions of P_{phys} undefined.

The solution is to not only train with samples from P_{phys} but to also include samples from the RBM. This will suppress the development of high probability regions of P_{RBM} in areas where no states are sampled. In practice, two steps are performed. First, states from the physical distribution are sampled through Metropolis. Second, in each training step, we sample from the RBM using Gibbs sampling, combine it with samples obtained with Metropolis, and then perform one training step.

We use ADAM batch-gradient descent to minimize the loss. Since after each training step P_{RBM} changes, new samples need to be generated from the RBM in each epoch. Not only that but in each training step, $E_{\text{phys}}(x)$ for the new samples needs to be computed. This is expensive for the Kitaev model.

To alleviate the computational load, two tricks are employed. First, a pre-training step is performed where completely random states are generated and trained on until the loss of the states generated from the physical distribution no longer decreases. This reduces the training time because $E_{\text{phys}}(x)$ only needs to be computed once for the random states. The second trick is to use a buffer of past RBM samples where instead of generating new samples in each training step, only a portion of the used states are new, and the rest are reused from previous training steps. For our experiments, the update size for the buffer is set to $ub = 200$ and the buffer size is set to the size of the training set. More details can be found in Alg. 1 and Alg. 2.

Algorithm 1 Functions to Train the RBM

function UPDATE BUFFER
update $\bar{x}_{\mathrm{RBM}}$ with k steps of Gibbs sampling
$x_{\mathrm{RBM}} = x_{\mathrm{RBM}} \cup \bar{x}_{\mathrm{RBM}}$
calculate $E_{\mathrm{phys}}(\bar{x}_{\mathrm{RBM}})$
if size(x_{RBM}) > N **then**
remove last elements of x_{RBM}
end if
end function

function LOSS(x)
$\mathrm{diff}^{(i)} = E_{\mathrm{phys}}(x^{(i)})\beta - F_{\mathrm{RBM}}(x^{(i)}; param)$
$C = \frac{1}{M}\sum_{i=1}^{M} \mathrm{diff}^{(i)}$
$\mathrm{loss} = \frac{1}{M}\sum_{i=1}^{M}(\mathrm{diff}^{(i)} - C)^2$
return loss
end function

function TRAIN(x')
for x = (M elements of x') **do**
loss = LOSS(x)
$param \leftarrow param - \mathrm{ADAM}\left(\frac{\partial \mathrm{loss}}{\partial param}\right)$
end for
return loss
end function

Algorithm 2 Train the RBM

$N \leftarrow$ train size ▷ Initialization
$M \leftarrow$ batch size
$K \leftarrow$ number of kernels
$L_W \leftarrow$ kernel size
$ub \leftarrow$ Buffer update size
initialize $W \sim \mathcal{N}\left(m = 0, \sigma = \frac{2}{KM^2}\right)$ with size $K \times L_W \times L_W$
initialize $v_{\text{bias}} = 0$ with size 1
initialize $h_{\text{bias}} = 0$ with size K
$param = [W, h_{\text{bias}}, v_{\text{bias}}]$
sample N samples $x_{\text{phys}} \sim P_{\text{phys}}$ with Metropolis
calculate $E_{\text{phys}}(x_{\text{phys}})$
initialize $\bar{x}_{\text{RBM}} \sim \text{Binomial}(\text{size} = (ub, L, L), p = 0.5)$
initialize $x_{\text{rand}} \sim \text{Binomial}(\text{size} = (N, L, L), p = 0.5)$

while LOSS(x_{phys}) does decrease **do** ▷ Pre-training
    TRAIN(x_{rand})
end while

iterate UPDATE BUFFER until size(x_{RBM})= N ▷ Filling the Buffer

while loss $> \epsilon$ **do** ▷ Training
    UPDATE BUFFER
    $x_{\text{tot}} = x_{\text{RBM}} \cup x_{\text{phys}}$
    loss = TRAIN(x_{tot})
end while

This is the most important drawback of using fully connected RBMs. To train them, a normal Monte Carlo simulation (expensive) needs to be performed in advance. This usually makes the use of RBMs redundant. Often a similar amount of samples are needed to train an RBM as it would have been necessary to compute expectation values. The second drawback is that with larger lattice size, the number of parameters that need to be learned scales with L^4 in 2D. So the bigger L, the more training time is required. The CRBM solves this problem, as the training can be done at smaller lattice sizes and so both problems can be circumvented.

3.3.5. Regularization

There are RBMs with different parameters that describe the same probability distribution. It would be convenient to find a way to obtain the RBM that describes the Markov chain with the lowest autocorrelation time τ. If the conditional distribution $P(v_j = 1|h) = \sigma(v^j_{\text{bias}} + \sum_i h_i W_{ij})$ is as random as possible, then the autocorrelation will be small. If all parameters are 0 then $P(v_j = 1|h) = \frac{1}{2}$, then the conditional distribution is completely random and the autocorrelation time is minimal. The easiest way to get close to this is to add a L2 regularization term to the loss:

$$\text{loss}_{\text{regu}}(W, v_{\text{bias}}, h_{\text{bias}}) = \text{loss}(W, v_{\text{bias}}, h_{\text{bias}}) + \lambda_{\text{regu}} \left(v^2_{\text{bias}} + \sum_k^K (h^k_{\text{bias}})^2 + \sum_k^K \sum_{ij}^{L_W} (W^k_{ij})^2 \right) \tag{3.26}$$

where λ_{regu} is a constant that sets how much the loss is regularized. Note that the added term can not equal 0, as this would only be possible if the target probability distribution is constant. Practically, we first train the RBM with $\lambda_{\text{regu}} = 10^{-4}$ until $\text{loss}_{\text{regu}}$ no longer decreases. After that λ_{regu} is set to zero and the RBM is trained until completion. Note that a regularization term was only used for the Ising model.

3.4. Sampling

This chapter describes different sampling techniques that try to decrease the autocorrelation time for both the local Metropolis and the RBM. Error correction schemes based on Metropolis are employed for the RBM.

3.4.1. Parallel Tempering

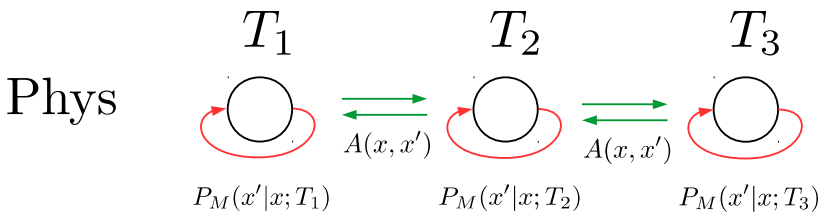


Figure 3.3.: Parallel tempering step where $P_M(x'|x)$ represents a local Metropolis step and $A(x', x)$ (Eq. 3.27) is the probability that the two temperatures exchange states. Green represents a cheap operation. Red represents an expensive operation in the case of the Kitaev model.

Often sizeable problems plague Monte Carlo simulations at small temperatures, as acceptances rates are very low. A method commonly used to tackles this is parallel tempering (PT), also called replica exchange [23, 24]. Parallel tempering takes advantage of the fact that probability distributions at similar temperatures overlap. It introduces an exchange between high and low temperatures so that the low temperature dynamics can take advantage of the high acceptance rates at high temperatures. States are exchanged between temperatures T_1 and T_2 with Metropolis acceptance probability:

$$A(x_1, x_2) = \min(1, e^{\Delta E}) \tag{3.27}$$

$$\Delta E = F_1(x_1)\beta_1 + F_2(x_2)\beta_2 - F_1(x_2)\beta_1 - F_2(x_1)\beta_2. \tag{3.28}$$

First, a Metropolis step and then a state exchange with the neighboring temperature is performed (Fig. 3.3).

Note that for the Kitaev model, exchanges are computationally cheap, as the eigenproblem for both states has already been solved.

3.4.2. Statistical Corrections

The RBM is sampled through Gibbs sampling as stated in Eq. 3.22 and 3.23. This Markov chain only converges to the physical distribution if the loss is exactly 0. We will see that in the case of the Ising model no corrections are needed, as the loss is very small. For the Kitaev model, the loss is too large to be neglected and corrections are necessary. Three different correction schemes are presented.

3.4.2.1. Metropolis RBM Update (MRBM)

The most straight forward way to correct the RBM is by applying a Metropolis acceptance step between Gibbs sampling steps as in Ref. [2]. After k Gibbs sampling steps the state is accepted with the Metropolis acceptance probability:

$$A(x', x) = \min\left(1, \frac{P_{\text{phys}}(x')}{P_{\text{phys}}(x)} \frac{P_{\text{RBM}}(x|x')}{P_{\text{RBM}}(x'|x)}\right). \tag{3.29}$$

The RBM fulfills the detailed balance condition as proven in App. A.1, and so:

$$\frac{P_{\text{RBM}}(x|x')}{P_{\text{RBM}}(x'|x)} = \frac{P_{\text{RBM}}(x)}{P_{\text{RBM}}(x')} \tag{3.30}$$

from this the followings is obtained:

$$A(x', x) = \min\left(1, e^{\Delta E(x', x)}\right) \tag{3.31}$$

$$\Delta E(x', x) = F_{\text{RBM}}(x') + E_{\text{phys}}(x)\beta - F_{\text{RBM}}(x) - E_{\text{phys}}(x')\beta. \tag{3.32}$$

The closer the RBM is to the target distribution the larger the acceptance rate. In practice, a parallel tempering step is done after this correction step. Note that if a state is rejected all the computations done for the k Gibbs steps will be wasted. This is why we introduce a modified version of parallel tempering.

3.4.2.2. Parallel RBM Tempering Update (PRBM)

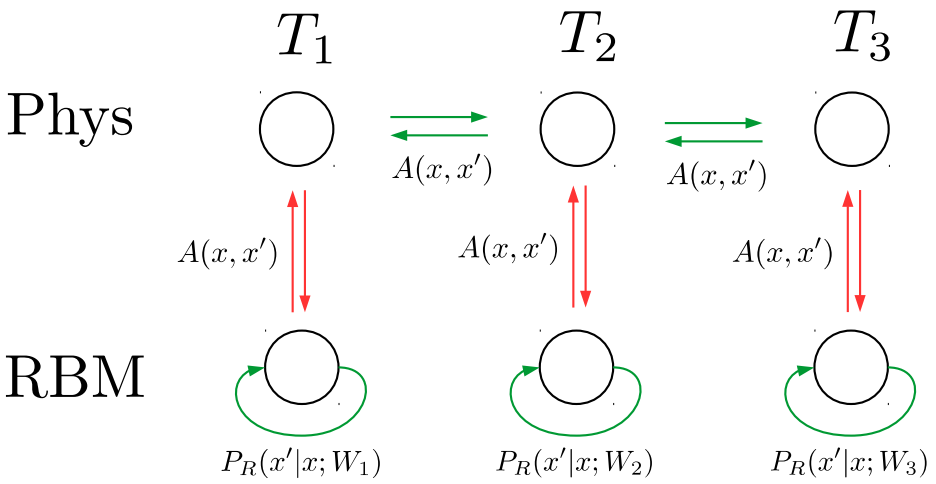


Figure 3.4.: Parallel RBM tempering where instead of accepting states with Metropolis an exchange with the RBM is introduced. The conditional distribution P_R represents k Gibbs steps.

A small modification leads to the Parallel RBM Tempering Update (PRBM). Instead of using Metropolis for the updates, we introduce the RBM as a different probability distribution that the physical states can exchange with.

$$\Delta E = F_{\text{RBM}}(x_{\text{RBM}}) + E_{\text{phys}}(x_{\text{phys}})\beta - F_{\text{RBM}}(x_{\text{phys}}) - E_{\text{phys}}(x_{\text{RBM}})\beta \tag{3.33}$$

The more the approximate RBM distribution and physical distribution overlap, the larger the acceptance rate is (see Eq. 3.33). Sampling consists of three steps as seen in Fig. 3.4. First, the RBM states are updated through Gibbs sampling a set amount of times, for example, 10^3 times. Second, the RBM states are exchanged with the physical distribution with probability $A(x_{\text{RBM}}, x_{\text{phys}})$. Lastly, a normal parallel tempering exchange is performed between different temperatures. Each time a state is accepted, it is uncorrelated to the previous one. This lies in stark contrast with normal Metropolis where only one lattice point is updated each time a new state is accepted. Another advantage of this technique is that it gives another opportunity for parallelization, as the RBM Gibbs update can be performed in parallel with the correction step.

3.4.2.3. Local Metropolis Update (LMRBM)

Remember that after training the RBM on small lattice sizes the objective is to sample from larger lattice sizes. The question that remains is how the loss behaves when the lattice size is increased. Assuming that the physical interactions do not change much when the lattice size is increased the loss per lattice site ΔE should remain constant. This means that the loss should scale with L^2, and so the mean acceptance rate dependent on the lattice size is $A(L) = e^{-\Delta E L^2}$. As $L \to \infty$ the acceptance rate tends to zero exponentially fast. To solve this problem, instead of updating all lattice sites, only a subset of the lattice is updated with Gibbs sampling between Metropolis acceptance steps. The method is named Local Metropolis RBM Update (LMRBM). The algorithm is as follows: First, randomly choose a subset U of elements of x, next, update them k times using Gibbs sampling, after that perform one Metropolis correction step. In App. A.1.2 we prove that, since the conditional distribution can be written as $P(x|h) = \prod_i P(x_i|h)$, this procedure does fulfill the detailed balance condition. Note that the same subset U is updated k times. Two options to randomly choose the update set U come to mind. A square can be placed at random on the lattice, this will be called window LMRBM. Another option is to choose U to be completely random, which will be referred to as random LMRBM. In Sec. 6.4 we will compare Parallel RBM Tempering, Metropolis RBM, window Local Metropolis RBM, and random Local Metropolis RBM.

Algorithm 3 random LMRBM

initialize $x \sim$ Binomial(size $= (L, L), p = 0.5$)
for 0 to M **do**
  U = u randomly chosen elements from $[0, L^2]$
  x_{new} = copy x
  for 0 to k **do**
    x'_{new} = Gibbs Sample x_{new}
    x_{new} = combine U lattice sizes of x'_{new} with not U lattice size of x_{new}
  end for
  x = Acceptance(x_{new}, x)
  save x
end for

Now the question that remains is: How do we decide the size u of the subset U that will be updated? Assuming that all states are accepted, the smaller u the larger is the autocorrelation time $\tau = (\frac{L^2}{u})^c$ where u is the size of the update U, and c is a constant

that typically lies between 1 and 2, depending on the temperature. It was assumed that a system with update size u and lattice size L has $\frac{L^2}{u}$ effective lattices. When the acceptance rate is smaller than one the autocorrelation time is:

$$\tau = \frac{(\frac{L^2}{u})^c}{A(u)}$$
$$\tau = \left(\frac{L^2}{u}\right)^c e^{\Delta E u}$$

minimizing the autocorrelation time with respect to u, the optimal update size is obtained:

$$u = \frac{c}{\Delta E}. \tag{3.34}$$

So the optimal acceptance rate is:

$$A_{\text{opti}} = e^{-c}$$

since in practice c usually lies between 1 and 2, the update size should be tuned so that the acceptance rate has a value between 0.36 and 0.14.

The best autocorrelation time one can obtain is:

$$\tau_{\text{opti}} = \left(\frac{\Delta E L^2}{c}\right)^c e^c.$$

The maximal expected speedup for the use of an RBM in contrast to Metropolis is:

$$\begin{aligned}\frac{\tau_{\text{metro}}(L)}{\tau_{\text{opti}}(L)} &= \left(\frac{\Delta E L^2}{c}\right)^{-c} e^{-c} L^{2c} \\ &= \left(\frac{\Delta E}{c}\right)^{-c} e^{-c} \\ &= \left(\frac{c}{\Delta E e}\right)^c.\end{aligned}$$

Note that in this analysis we have assumed that after k Gibbs steps, samples are uncorrelated to each other. This is mostly the case, as one Metropolis correction step for the Kitaev model is expensive and so a lot of Gibbs steps are performed between each correction step.

3.4.3. Symmetry Sampling

Note that the CRBM was restricted such that it is translation invariant in its probability and conditional distribution. The Ising model, however, has other symmetries. Our hypothesis is that these symmetries are learned by the probability distribution of the CRBM, but that the conditional probabilities used for sampling can break these symmetries. After many sampling steps, all symmetries are restored. This could be exploited to lower the autocorrelation time.

Before h is sampled from the conditional distribution $P(h|v^{(1)})$, $v^{(1)}$ is transformed via a symmetry. A new state $v^{(2)}$ is generated from $P(v^{(2)}|h)$ and is then transformed back via the same symmetry. Each symmetry is applied with probability $\frac{1}{3}$. The Ising model with no magnetization has 3 symmetries: spin-flip symmetry $v_s = 1 - v$, rotational symmetry, and mirroring symmetry. This modified Markov chain is given the name symmetry Gibbs sampling. Its main advantage is that different conditional distributions are used in each sampling step.

In Sec. 4.2 it will be shown that it only gives a small improvement over normal Gibbs sampling, that is why it will not be used for the Kitaev model.

CHAPTER 4

Ising Model

In this chapter, we will use the Ising model as a proof of concept for the Convolutional Restricted Boltzmann Machine (CRBM). In order to do this, Monte Carlo (MC) simulations of the two-dimensional Ising model will be sped up with the use of CRBM, a type of RBM. The probability distribution of the Ising model was already learned in the past, in Ref. [1], for a small lattice size $L = 10$ with a fully connected RBM, but not with a CRBM. With CRBM, it is possible to train at a small lattice size $L = 3$ and sample at larger lattice sizes. In this chapter, it will be proven that a CRBM can successfully sample on lattice sizes it was not trained on and that using this method outperforms the local Metropolis algorithm.

Using this method, an autocorrelation time that is 80 times smaller than the one attained by the local Metropolis algorithm is achieved on a lattice size $L = 100$ near the critical temperature. A Jupyter Notebook with a small demo can be found in supplementary information [25, 26].

4.1. The Model

The Ising model is a physical model proposed to study the behavior of ferromagnetic materials. It is a paradigmatic model of statistical mechanics since it is not only one of the simplest statistical models to show a phase transition, but it has an exact analytical solution. This makes the Ising model perfect to test new computational approaches.

Ising's model was invented by physicist Wilhelm Lenz (1920), who conceived it as a problem for his student Ernst Ising to demonstrate that the system presented a phase transition. Ising, in his thesis (1924), showed that in one-dimension there was no such phase transition. Ising's two-dimensional square grid model is much more difficult, and it was only given an analytical description much later, by Lars Onsager (1944), which showed that statistical physics was capable of describing phase transitions, which finally consolidated statistical mechanics.

The model is described by its energy:

$$E_{\text{Ising}}(s) = -\frac{J}{2}\sum_{\langle ij\rangle} s_i s_j \tag{4.1}$$

and its Boltzmann distribution is:

$$P_{\text{Ising}}(s) = \frac{e^{-\beta E_{\text{Ising}}(s)}}{Z} \tag{4.2}$$

where $s_i = \pm 1$ and $\langle ij\rangle$ is the sum over all nearest neighbors. The temperature T is in units of J and $k_{\text{B}} = 1$. If two spins point in the same direction, this gives a negative contribution to the energy. The state s that minimizes the energy is the one where all spins point in the same direction, a magnet. At a temperature of $T_c = 2.269$, the system undergoes a phase transition from a ferromagnetic to a disordered phase.

For the Monte Carlo simulations, four thermodynamic constants are of interest: mean energy $\langle E\rangle$, specific heat C_v, magnetization $\langle M\rangle$, and susceptibility χ:

$$\langle E\rangle = -\frac{\partial \ln(Z)}{\partial \beta} = \sum_s E(s)P(s) \tag{4.3}$$

$$C_v = \frac{\partial\langle E\rangle}{\partial T} = \frac{1}{T^2}\left(\langle E^2\rangle - \langle E\rangle^2\right) \tag{4.4}$$

$$\langle M\rangle = \sum_s M(s)P(s) \tag{4.5}$$

$$\chi = \frac{1}{T}(\langle M^2\rangle - \langle M\rangle^2). \tag{4.6}$$

To be able to do the MC simulation the local Metropolis algorithm is used. It flips a

random spin at site i:

$$s'_j = \begin{cases} +s_j & , j \neq i \\ -s_j & , i = j. \end{cases} \tag{4.7}$$

For the Ising model, the energy difference only depends on the surrounding spins:

$$\begin{aligned} \Delta E &= E(s') - E(s) \\ \Delta E &= +(-s_i) \sum_{\langle ij \rangle} s_j - s_i \sum_{\langle ij \rangle} s_j \\ \Delta E &= -2 s_i \sum_{\langle ij \rangle} s_j \end{aligned} \tag{4.8}$$

as the rest cancels out. This energy difference can then be used to either accept or discard the state with the flipped spin.

4.2. Monte Carlo Simulations of the Ising Model

The python library Theano [27] is used for optimization and sampling. Differentiation of the loss is performed automatically by the library.

Calculations for the Metropolis-Hasting algorithm were performed using a CPU and the convolutions needed for the CRBM were computed using a GPU.

The CRBM consists of 2 kernels with size 2×2. First, it is trained for the temperature $T = 2.2$. The model has $2 \times 2 \times 2$ weights, one visible bias, and two hidden biases. These are 11 free parameters for the free energy $F_{\mathrm{CRBM}}(x)$. The CRBM is trained with $L = 3$. Since there are only $2^{3^2} = 512$ different states, the training is done with all the available states. A loss of 10^{-16} is achieved. Note that it is indeed possible to find the optimal parameters analytically, which is why a small loss can be achieved. For more complex models, an analytical solution is not possible.

The values for the trained parameters can be seen in Fig. 4.1. In the Ising model, only neighboring lattice points interact with each other, and so as expected, the kernels only connect neighboring lattice points. Not only that but all the values of W_1 have the same sign. This makes it favorable for spins to align. For an antiferromagnetic interaction, the

values in one kernel would have a different sign. The two kernels W_1 and W_2 have opposite signs to each other.

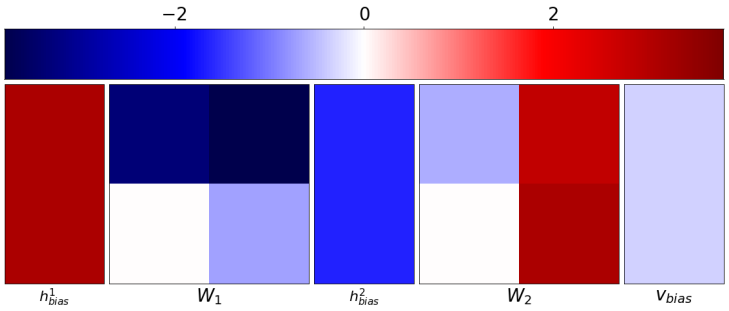


Figure 4.1.: Convolution kernels W_i trained at $L = 3$ for the Ising model with temperature $T = 2.2$. See Eq. 3.21 for more details. They represent a ferromagnetic nearest-neighbor interaction.

The CRBM is trained using L2 regularization and the sampling is performed using symmetry Gibbs sampling. Later we will discuss what performance boost these two methods give. Note that one Metropolis step takes less time to compute than a CRBM step. With $L = 100$, $k_h = 10^3$ Metropolis steps take the same time as $k_h = 17$ CRBM steps.

To get a visual representation of how long it takes for the CRBM and Metropolis to warm up: first, a random initial state is generated and next, $k_h 10^n$ states are generated with the CRBM and with Metropolis. In Fig. 4.2 it can be seen how the CRBM reaches equilibrium after around 17×10^3 steps. In contrast, the Metropolis algorithm takes around 100 times longer, needing $10^3 \times 10^5$ iterations to reach equilibrium.

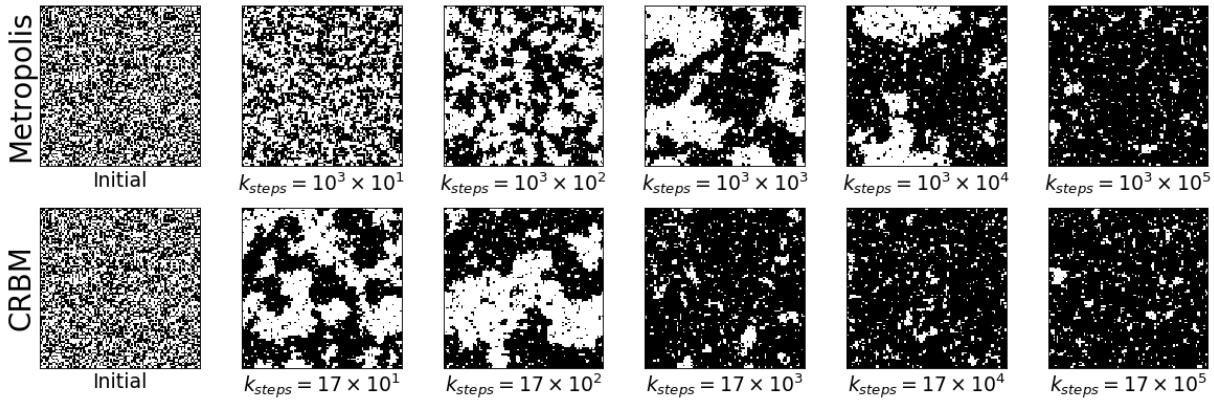


Figure 4.2.: States generated after k_{steps} with temperature $T = 2.2$ using the Metropolis algorithm and the CRBM with $L = 100$.

The thermodynamic constants are computed for different lattice sizes with constant tem-

perature $T = 2.4$. For the Metropolis algorithm, $k_h = 10^3$ steps are performed between each recorded step. For the CRBM, k_h is adapted for different lattice sizes so that both methods take the same time. Before thermodynamic constants can be recorded, states need to be updated many times so that the chain reaches equilibrium, this is called warm up phase. In practice, the first 10% of the states sampled are discarded.

10^5 steps are recorded, and then thermodynamic constants seen in Fig. 4.3 are computed. To compute the error bars, batch means (see App. A.2) with a batch size of 10 is employed.

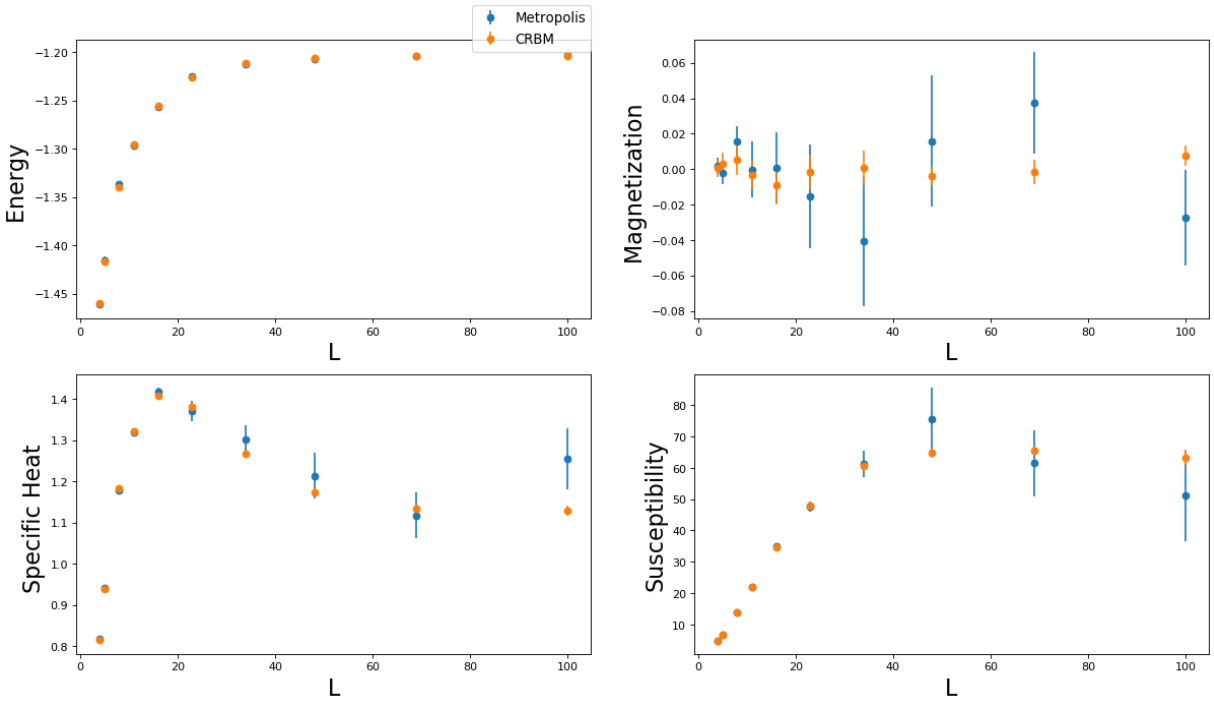


Figure 4.3.: Thermodynamic constants of the Ising model varying L at a constant temperature $T = 2.4$ using Metropolis and CRBM to generate 10^5 samples.

One can observe that the error increases as the lattice size increases. The CRBM can not only reproduce the thermodynamic constants but has a lower error than Metropolis. This is the case, especially when computing the susceptibility and specific heat, where Metropolis fails to give the right value at large lattice sizes. The autocorrelation τ for the magnetization is depicted in Fig. 4.4. The CRBM has a higher autocorrelation time than Metropolis for lattice sizes smaller than 8. After that, the CRBM has a considerably smaller autocorrelation time.

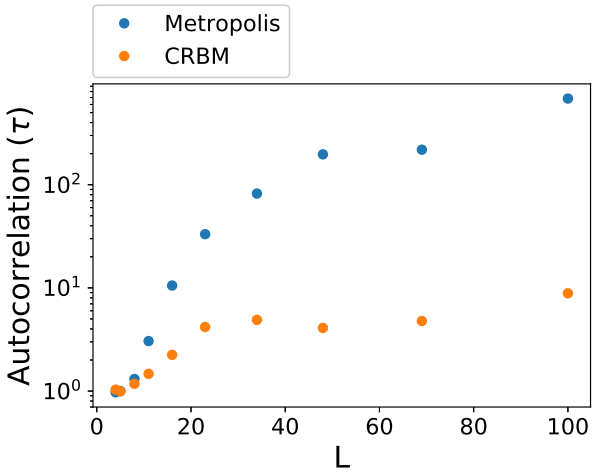


Figure 4.4.: Autocorrelation time of the energy varying L at a constant temperature $T = 2.4$ using Metropolis and CRBM to generate samples.

When working with small lattices, the Metropolis-Hasting algorithm has the advantage that its mixing rate is only based on single spin flips. As a result, the smaller the grid, the larger the percentage of spin flips per step. On the other hand, the CRBM has a slower mixing rate. This is because Gibbs sampling has a slow update percentage per lattice site, independent of the lattice size.

This reverses at large lattices. The Metropolis algorithm still can only flip single spins, but the CRBM can update several spins at a time. This means that for large lattice sizes, the CRBM has a faster mixing rate than Metropolis. For $L = 100$, the CRBM is 80 times faster than the Metropolis algorithm.

Let's now take a look at the thermodynamical constants at different temperatures with $L = 100$ (see Fig. 4.5). Four approaches are compared: Metropolis Hasting, CRBM without regularization, CRBM with regularization (see Sec. 3.3.5), and CRBM with regularization and symmetry sampling. Note that parallel tempering was not used for the Ising model. Close to the critical temperature $T_c = 2.269$, the susceptibility is not approximated accurately by the Metropolis algorithm. On the other hand, the three CRBMs have no problem approximating it. At other temperatures, the CRBM methods and the Metropolis method accurately describe the thermodynamic constants.

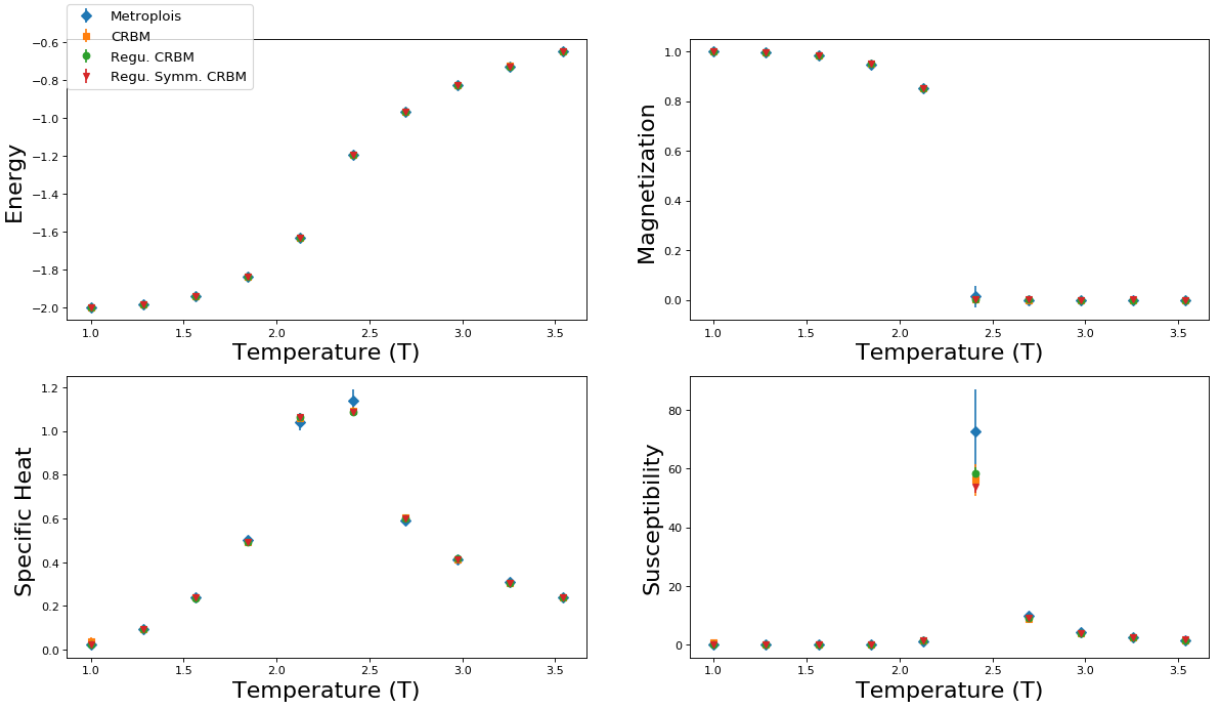


Figure 4.5.: Thermodynamic constants varying temperature with $L = 100$ using Metropolis and different CRBMs to generate 10^5 samples.

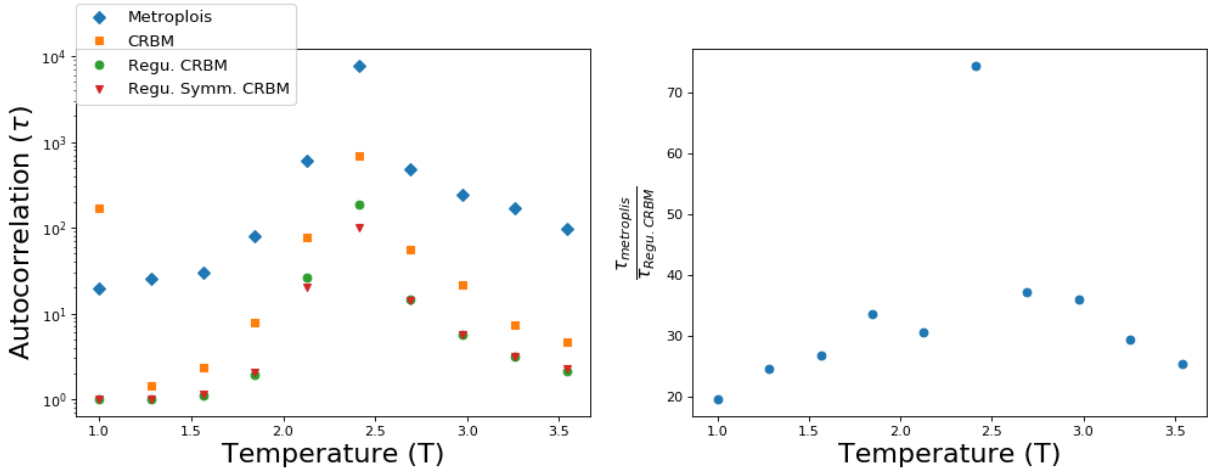


Figure 4.6.: (Left) Autocorrelation time for different temperatures at $L = 100$ using Metropolis and different CRBMs to generate samples. (Right) The ratio of the autocorrelation of Metropolis and the regularized CRBM.

In Fig. 4.6, the autocorrelation τ of the magnetization is presented. The three CRBMs have a lower autocorrelation τ than Metropolis. The autocorrelation of all models increases as the critical temperature is approached. Far away from the critical temperature, the best CRBM performs around 30 times better than Metropolis. Close to T_c, the performance

increases to around 70. The regularized CRBM performs better than the regularized one, and the symmetry CRBM performs slightly better when close to the phase transition.

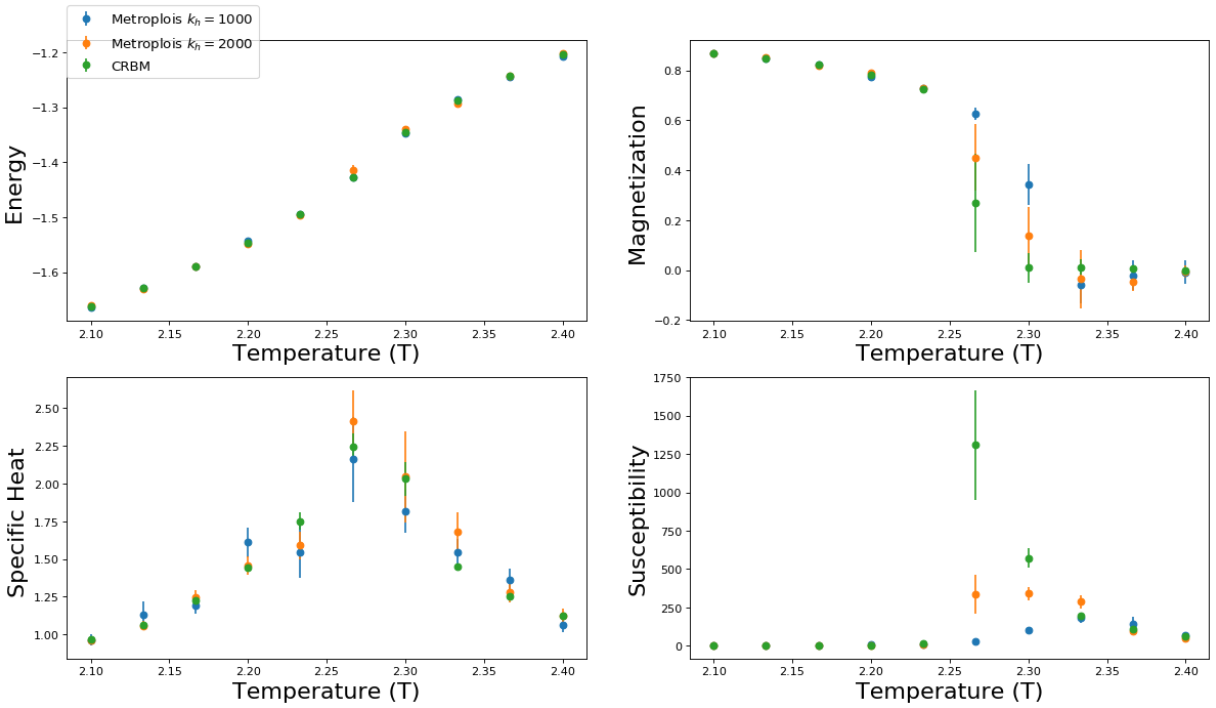


Figure 4.7.: Thermodynamic constants at different temperatures T that are close to the phase transition, with $L = 100$ using Metropolis and a CRBM to generate samples.

The same has been done for temperatures close to T_c (see Fig. 4.7). Metropolis fails to converge with $k_h = 1000$ steps between samples. The CRBM, on the other hand, estimates the thermodynamic constants more accurately. To check this, Metropolis is run again with $k_h = 2000$. This gives results that are closer to the ones obtained with the CRBM but still have large error bars. We conclude that the CRBM can not only capture correctly the phase transition of the Ising model but also converges faster than the Metropolis algorithm at criticality.

4.3. Summary

A Convolutional Restricted Boltzmann Machine (CRBM) was employed to sample from the two-dimensional Ising model as a proof of concept. In the past, fully connected RBMs have been employed [1]. However, fully connected RBMs suffer from long training times

when the lattice size is increased. In contrast, a CRBM can be trained in a matter of seconds for any lattice size.

For the Ising model, the CRBM could successfully reproduce thermodynamic constants accurately on lattice sizes that it was not trained on. With small lattice sizes $L < 8$, the CRBM has a larger autocorrelation time than the Metropolis algorithm. However, this reverses when $L > 8$. With $L = 100$, the CRBM has an up to 80 times smaller autocorrelation time. Since it was shown that both regularization and symmetry sampling give only marginally better results, the two methods will not be used in future chapters.

In conclusion, CRBMs seem to be a useful tool to perform Monte Carlo simulations. There are better techniques to perform MC for the Ising model, like cluster algorithms, but for other models like the Kitaev model, such algorithms do not exist. In the two following chapters, the usefulness of the CRBM for Monte Carlo simulations will be explored.

CHAPTER 5

Kitaev Model

This chapter has three objectives. The first objective is to test our training method for the first time. This training method has not been applied to the Ising model as it is a model that can be learned trivially. The second objective is to test the Parallel RBM Tempering Update (PRBM). The last objective is to prove that a CRBM can outperform Metropolis for physical models where no better update method is known. The Kitaev honeycomb model is a good candidate, as MC calculations over $L = 18$ are hard to perform.

5.1. The Hamiltonian

Kitaev's honeycomb lattice model [28] is an analytically tractable liquid spin model that gives rise to quasi-particles. Spins interact in a honeycomb lattice and fractionalize into Majorana fermions. A topological quantum spin liquid emerges in the ground state. Kitaev established that anyons could be used for fault-tolerant quantum computing. Anyonic quantum computing could help sidestep the problem of quantum errors introduced by the environment by protecting the sates topologically. If realized experimentally, it could serve as quantum memory or as a quantum computer.

Its ground state can be computed analytically and its specific heat can be calculated using a classical Monte Carlo simulation. Its Hamiltonian is:

$$H = -J_x \sum_{x \text{ bonds}} \sigma_i^x \sigma_j^x - J_y \sum_{y \text{ bonds}} \sigma_i^y \sigma_j^y - J_z \sum_{z \text{ bonds}} \sigma_i^z \sigma_j^z \tag{5.1}$$

where σ_i are Pauli matrices and the J_αs are the interaction strengths. Each black lattice point is connected to three white lattice points through x,y, and z (see Fig. 5.1).

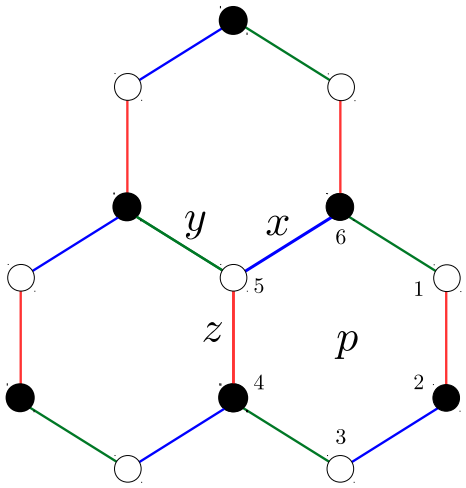


Figure 5.1.: Honeycomb lattice. Particles 5 and 6 are connected with strength J_x (blue), 4 and 5 with J_z (red), and 6 and 1 with J_y (green). Each combination of white and black in the x-y-direction constitutes a unit cell.

It is a frustrated magnetic system. A spin cannot arrange its orientation such that its three neighbors profit from the interaction, hence its ground state is a spin liquid.

5.2. Formalisms and Boundary Conditions

Two similar formalisms, that make use of Majorana fermions, can be used to solve the Hamiltonian. They both can be found in the literature. Equivalence between the two formalisms was found with open boundary conditions. This question is of interest, as formalism 1 is easier to learn for the CRBM but formalism 2 makes analytical calculations easier.

5.2.1. Formalism 1

This formalism can be found in the supplementary material of Ref. [29]. The Jordan Wigner transformation is applied:

$$\sigma^x_{m,n}\sigma^x_{m,n+1} = (S^+_{m,n} + S^-_{m,n})(S^+_{m,n+1} + S^-_{m,n+1}) \tag{5.2}$$
$$\sigma^y_{m,n}\sigma^y_{m,n+1} = -(S^+_{m,n} - S^-_{m,n})(S^+_{m,n+1} - S^-_{m,n+1}) \tag{5.3}$$
$$\sigma^z_{m,n}\sigma^z_{m',n'} = (2n_{m,n} - 1)(2n_{m',n'} - 1) \tag{5.4}$$

inserting $S^+_{m,n} = a^\dagger_{m,n} e^{-i\pi \sum_{i<m,j<n} n_{i,j}}$ and $S^-_{m,n} = a_{m,n} e^{i\pi \sum_{i<m,j<n} n_{i,j}}$ yields:

$$\sigma^x_{m,n}\sigma^x_{m,n+1} = -(a_{m,n} - a^\dagger_{m,n})(a_{m,n+1} + a^\dagger_{m,n+1}) \tag{5.5}$$
$$\sigma^y_{m,n}\sigma^y_{m,n+1} = (a_{m,n} + a^\dagger_{m,n})(a_{m,n+1} - a^\dagger_{m,n+1}) \tag{5.6}$$
$$\sigma^z_{m,n}\sigma^z_{m',n'} = (2n_{m,n} - 1)(2n_{m',n'} - 1) \tag{5.7}$$

with Majorana transformation:

$$c^w = (a^w - a^{w\dagger})/i \qquad \bar{c}^w = a^w + a^{w\dagger} \tag{5.8}$$
$$c^b = (a^b + a^{b\dagger}) \qquad \bar{c}^b = (a^b - a^{b\dagger})/i \tag{5.9}$$

where w,b stand for white and black lattice points. This gives the Hamiltonian:

$$H = -iJ_x \sum_{\langle ij\rangle_x} c^b_i c^w_j - iJ_y \sum_{\langle ij\rangle_y} c^b_i c^w_j - iJ_z \sum_{\langle ij\rangle_z} u^z_{i,j} c^b_i c^w_j \tag{5.10}$$

where $u^z = i\bar{c}^b\bar{c}^w$ are defined on each z bond. u_z commute with the Hamiltonian $[H, \bar{c}^b\bar{c}^w] = 0$ and with each other. The eigenstates of u_z are also eigenstates of H. Since $(i\bar{c}^b\bar{c}^w)^2 = 1$ they must have eigenvalues ± 1. Accordingly, the operator u^z can be substituted by a Z_2 gauge filed that can take values ± 1.

The Hamiltonian is then rewritten as:

$$H = \sum_{ij} \frac{i}{4} A_{ij} c_i c_j \tag{5.11}$$

with

$$A_{ij} = 2 \begin{cases} -, & \text{if i is black} \\ +, & \text{if i is white} \end{cases} \begin{cases} J_\alpha, & \text{if } i,j \text{ connected through } \alpha = \{x,y\} \\ J_z u^z_{ij}, & \text{if } i,j \text{ connected through z} \\ 0, & \text{otherwise} \end{cases} \tag{5.12}$$

Note that the matrix A is anti-symmetric $A_{ij} = -A_{ji}$. A more detailed description off the filling of the matrix can be found in App. B.1. Now the Hamiltonian is quadratic, and so it can be diagonalized (see App. B). Note that in App. B.2 a method that yields 6 times faster computations, that is not found in the literature, is introduced.

The final Hamiltonian is:

$$\mathcal{H}(u^z) = \sum_\lambda \epsilon_\lambda(u^z)(a^\dagger_\lambda a_\lambda - \frac{1}{2}) \tag{5.13}$$

where ϵ_λ are the eigenvalues obtained from diagonalization for a specific gauge configuration. The partition function now reads:

$$\begin{aligned} Z &= \mathrm{Tr}_{u^z} \mathrm{Tr}_{a_\lambda} e^{-\beta \mathcal{H}(u^z)} && (5.14) \\ &= \mathrm{Tr}_{u^z} e^{-\beta F(u^z)} && (5.15) \end{aligned}$$

and so the probability and free energy for an u^z configuration are:

$$\begin{aligned} P(u^z) &= \frac{e^{-\beta F(u^z)}}{Z} && (5.16) \\ F(u^z) &= -T \ln\left(\mathrm{Tr}_{a_\lambda} e^{-\beta \mathcal{H}(u^z)}\right) && (5.17) \\ &= -T \sum_\lambda \ln(2\cosh(\beta\epsilon_\lambda/2)). && (5.18) \end{aligned}$$

The mean energy of the system is $E = -\frac{\partial \ln Z}{\partial \beta} = \langle E_f \rangle_{\mathrm{MC}}$ with:

$$\begin{aligned} E_f(u^z) &= \frac{\partial (F(u^z)\beta)}{\partial \beta} \\ &= -\sum_\lambda \frac{\epsilon_\lambda}{2} \tanh \frac{\beta \epsilon_\lambda}{2}. \end{aligned} \tag{5.19}$$

The specific heat is:

$$C_v = \frac{\partial E}{\partial T} = \frac{1}{T^2}\left(\langle E_f^2\rangle_{\mathrm{MC}} - \langle E_f\rangle_{\mathrm{MC}}^2 - \left\langle \frac{\partial E_f}{\partial \beta}\right\rangle_{\mathrm{MC}}\right) \tag{5.20}$$

with:

$$\frac{\partial E_f}{\partial \beta} = -\sum_{\lambda} \frac{\epsilon_\lambda^2}{2\cosh(\beta\epsilon_\lambda)+2}. \tag{5.21}$$

More detailed calculations can be found in App. B.

Note that this is the formalism that will be used for the Monte Carlo simulations. Nevertheless, it is also important to understand formalism 2, and not only formalism 1, which is discussed in this section, as not only does formalism 2 make analytical calculations easier, but it also is used in part of the literature.

5.2.2. Formalism 2

Formalism 2 was used by Kitaev *et al.* [28]. Four Majorana operators are introduced for each site, so two fermions per spin site:

$$b^x = (d + d^\dagger) \qquad b^y = (d - d^\dagger)/i \tag{5.22}$$

$$b^z = (f - f^\dagger)/i \qquad c = f + f^\dagger \tag{5.23}$$

the spin operators are then defined as $\sigma_i^\alpha = i b_i^\alpha c_i$. For them to satisfy all algebraic relations of the spin operator, $\sigma_i^x\sigma_i^y\sigma_i^z/i = 1 = D_i$ with $D_i = b_i^x b_i^y b_i^z c_i$ has to hold.

Not all states fulfill $D_i|\Phi\rangle = |\Phi\rangle$. There are two physical states $|\uparrow\rangle = |10\rangle, |\downarrow\rangle = |01\rangle$ and two nonphysical states $|00\rangle, |11\rangle$ per spin site. After the transformation this gives the Hamiltonian:

$$H = -iJ_x \sum_{\langle ij\rangle_x} u_{i,j}^x c_i^b c_j^w - iJ_y \sum_{\langle ij\rangle_y} u_{i,j}^y c_i^b c_j^w - iJ_z \sum_{\langle ij\rangle_z} u_{i,j}^z c_i^b c_j^w \tag{5.24}$$

where $u_{i,j}^\alpha = i b_i^\alpha b_j^\alpha$ are $\mathcal{Z}_2$ variables. This Hamiltonian is equivalent to the first formalism if only configurations with $u_{i,j}^x = u_{i,j}^y = 1$ are considered.

5.2.3. Wilson Operator

The Wilson operator is defined as the multiplication of the spin operators around a plaquette p:

$$W_p = \sigma_1^x \sigma_2^y \sigma_3^z \sigma_4^x \sigma_5^y \sigma_6^z \tag{5.25}$$

it does not only commute with the Hamilton operator $[W_p, H]$ but also squares to one $W_p^2 = 1$. This means that it has eigenvalues $w_p = \pm 1$. To rewrite the Wilson loop in the new basis, the following relationship is helpful:

$$\begin{aligned} \sigma_i^\alpha &= i b_i^\alpha c_i \\ &= i b_i^\alpha c_i D_i D_i \\ &= i b_i^\alpha c_i b_i^x b_i^y b_i^z c_i D_i \\ \sigma_i^\alpha &= -i \epsilon_{\alpha\beta\gamma} b_i^\beta b_i^\gamma D_i. \end{aligned} \tag{5.26}$$

As only physical states are considered $D_i = 1$:

$$\sigma_i^\alpha = -i \epsilon_{\alpha\beta\gamma} b_i^\beta b_i^\gamma$$

and so inserting this into the Wilson loop yields:

$$\begin{aligned} W_p &= \sigma_1^x \sigma_2^y \sigma_3^z \sigma_4^x \sigma_5^y \sigma_6^z \\ &= (-i)^6 (b_1^y b_1^z)(b_2^z b_2^x)(b_3^x b_3^y)(b_4^y b_4^z)(b_5^z b_5^x)(b_6^x b_6^y) \\ &= i^6 (b_1^z b_2^z)(b_2^x b_3^x)(b_3^y b_4^y)(b_4^z b_5^z)(b_5^x b_6^x)(b_1^y b_6^y) \\ &= u_{12}^z u_{23}^x u_{34}^y u_{45}^z u_{56}^x u_{16}^y. \end{aligned} \tag{5.27}$$

We arrive at the equation found in the literature. For the first formalism $u_{ij}^x = u_{ij}^y = 1$ and so $W_p = u_{12}^z u_{45}^z$. One Wilson loop configuration has different possible $\mathcal{Z}_2$ gauge field configurations. In the case of formalism 1, one Wilson loop configuration corresponds to $\frac{2^{L^2}}{2^{L(L-1)}} = 2^L$ u configurations. In the case of formalism 2, $\frac{2^{3L^2}}{2^{L^2}} = 2^{2L^2}$.

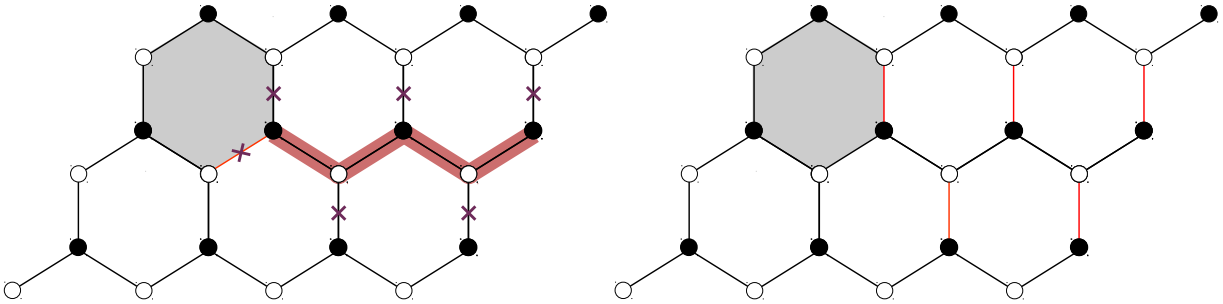
However, do all $\mathcal{Z}_2$ gauge field corresponding to one Wilson loop configuration have the same free energy? To answer this let us first establish a mapping between formalism 2 and formalism 1.

5.2.4. Mapping between the Formalisms

The eigenvalues $u_{ij}^{\alpha} = \pm 1$ of the link operators can be thought of as a classical $\mathcal{Z}_2$ gauge field. The operators D_i perform a local gauge transformation. When D_i is applied to an eigenstate, all u_{ij}^{α} that are linked with the site i invert their sign [30]:

$$D_i u_{ij} D_i = -u_{ij}$$

Applying this transformation to u does not change the free energy of the configuration. Now, using this transformation, we can map any state u to a state u' where $u'^{x}_{i,j} = u'^{y}_{i,j} = 1$. This is only possible with an open boundary condition in the x-y-direction. The strategy is conceived as follows: when $u_{i,j}^{x \text{ or } y} = -1$, the operator D_i is applied across all sites in the x-y-direction until the end of the lattice is reached. All u_{ij}^{α} that lie in the path where D_i is applied change their sign twice, which cancels out. But all u_{ij}^{α} that touch the path change their sign once. In Fig. 5.2, this is exemplified for a configuration where all u_{ij}^{α} are 1 except one x-link.



(a) (b)

Figure 5.2.: Grey colored honeycombs represent $w_p = -1$. (a) u configuration where all u_{ij}^{α} are 1 except the red x-link. The thick red line represents the path where the D operators are applied, and accordingly the links with violet crosses will change their sign. (b) After the operation all $u_{i,j}^{x \text{ or } y} = 1$. Note that the free energy of the configuration did not change.

Note that this operation is not possible with periodic boundary conditions in x-y-direction, as the algorithm would also change the left y-link and so not all $u^y = 1$.

5.2.5. Wilson Loop with unique Free Energy

So do all $\mathcal{Z}_2$ gauge fields corresponding to one Wilson loop configuration have the same free energy? The answer is yes if both z and x-y-directions are chosen to be open, then all u configurations that lead to a Wilson loop configuration can be reached through applying the Operator D_i and so all gauge fields that correspond to one Wilson loop configuration share the same free energy.

If one of the boundary conditions is chosen to be open and the other one periodic not all gauge configurations that lead to a Wilson loop configuration can be reached through applying D_i, which means that two groups remain unconnected, and so there are two possible free energies for one Wilson loop configuration. In the case of choosing periodic boundary conditions in both directions, there are four groups of distinct u configurations for one w configuration. This generates non-local effects that disappear when the lattice size is increased. To reach this conclusion, 100 random Wilson loop configurations were generated. Then for each Wilson loop configuration, the four different gauge field configurations are generated, and their free energies are compared. In Fig. 5.3, one can see that the absolute energy difference between them decreases exponentially with the lattice size.

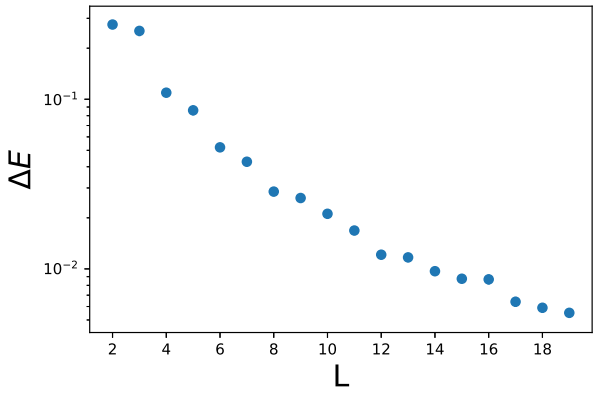


Figure 5.3.: Absolute energy difference between the four gauge fields that correspond to one Wilson loop configuration at different lattice sizes with periodic boundary conditions.

To confirm the previous statements, the exact specific heat for a lattice with size $L = 2$ is computed for both periodic and open boundary conditions in the x-y-direction, and periodic boundary conditions in the z-direction. As expected (see Fig. 5.4a), both formalisms give a different specific heat, while with open boundary conditions in x-y-direction, both formalisms yield the same result.

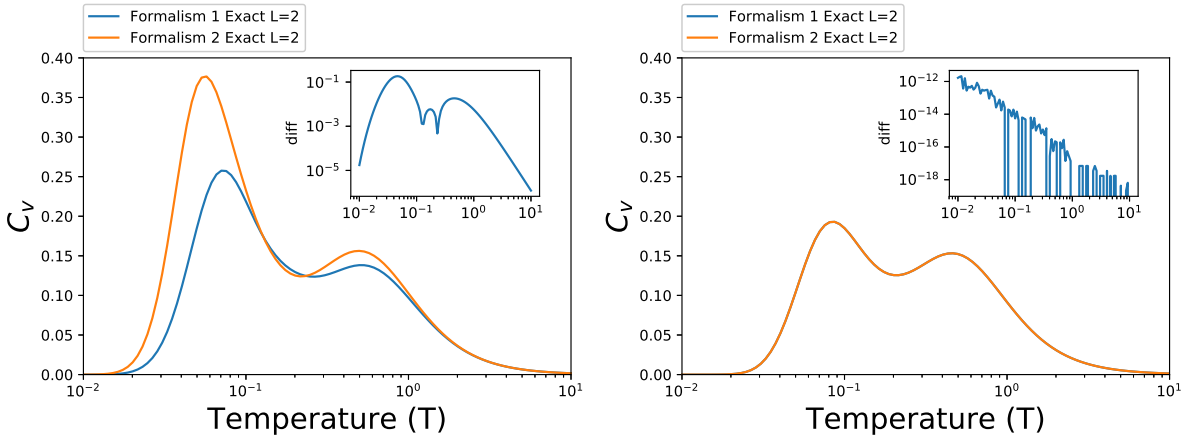


Figure 5.4.: (a) Exact specific heat for $L = 2$ with $N_p = 2L^2$ particles and $\alpha = 1$, computed through direct summation with periodic boundary conditions in x-y-direction. (b) Exact specific heat for $L = 2$ for open boundary conditions.

At higher L, both formalisms converge as can be seen in Fig. 5.5, where the specific heat is compared between both at $L = 12$.

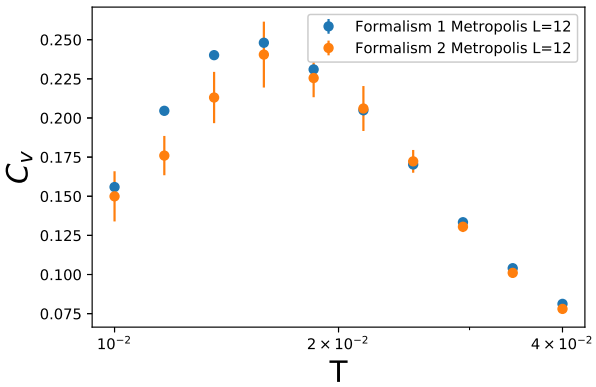


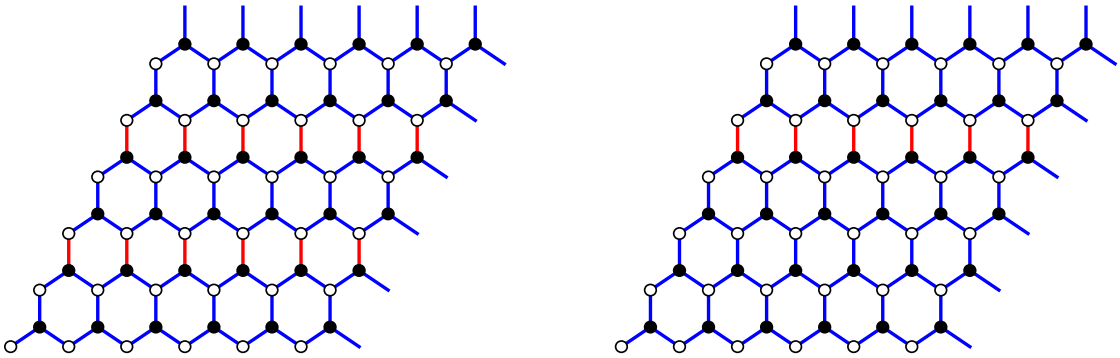
Figure 5.5.: MC simulation with $L = 12$ with periodic boundary conditions for both formalisms.

Note that the blue graph in Fig. 5.4b can also be found in the supplementary material of Ref. [29].

5.2.6. Ground State

A theorem by Lieb [31] was used by Kitaev to prove that the ground state for this Hamiltonian is the one where all Wilson loops take eigenvalues $w_p = 1$, this is called the zero

flux phase. As discussed before, there are two distinct representations for this Wilson loop configuration in the x-y open boundary case. In Fig. 5.6a, the one with the lowest energy is represented. Even though the configuration in Fig. 5.6b is also flux 0, it has a different value for the free energy.



(a) (b)

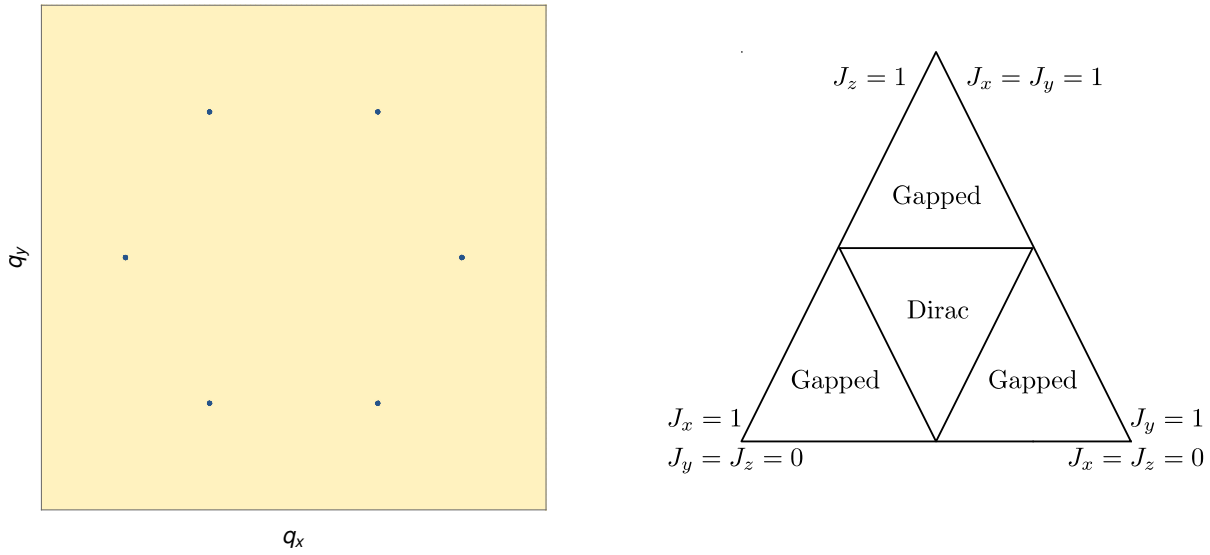
Figure 5.6.: Blue represents a gauge configuration component with -1 and red represents a +1 component. White Wilson loops represent +1 and grey loops represent -1. In this case, since it is a zero flux configuration all Wilson loops are +1. (a) Ground state for the Kitaev model at $L = 6$. (b) State with the same Wilson loop configuration but different free energy.

This is a good opportunity to further understand the non-localities that arise. For the zero flux phase, all u^z in a line need to have the same sign. All the zero flux states with an even amount of +1 lines have the same free energy. This free energy is different than the one for states with an odd amount of +1 lines. If the number of lines with positive signs is even then it is a ground state. If not, then the state has the other possible value for the free energy. This is a non-local effect. Non-localities are non-physical. This is why as discussed before this effect disappears at large lattice sizes or if the boundary conditions are chosen to be open.

The natural question to ask is whether this ground state is gaped or not. It was shown analytically through Fourier transformation of the Hamiltonian: that for the gauge field configuration $u^{\alpha}_{ij} = 1$ its energy spectrum is:

$$\begin{aligned}\epsilon(q) =& 2|J_x e^{iqn_1} + J_y e^{iqn_2} + J_z| \\ =& \pm 2\sqrt{J_x^2 + J_y^2 + J_z^2 + 2J_xJ_y\cos(q(n_1 - n_2)) + 2J_z(J_x\cos(qn_1) + J_y\cos(qn_2))}\end{aligned} \tag{5.28}$$

with basis vectors $n_1 = (\frac{1}{2}, \frac{\sqrt{3}}{2}), n_2 = (-\frac{1}{2}, \frac{\sqrt{3}}{2})$. The spectrum has a gap $\epsilon(q) = 0$ only if the triangle equations are satisfied $|J_x| \leq |J_y| + |J_z|$, $|J_y| \leq |J_x| + |J_z|$, $|J_z| \leq |J_x| + |J_y|$ (see Fig. 5.7b). The gap appears at six discrete Dirac points (see Fig. 5.7a).

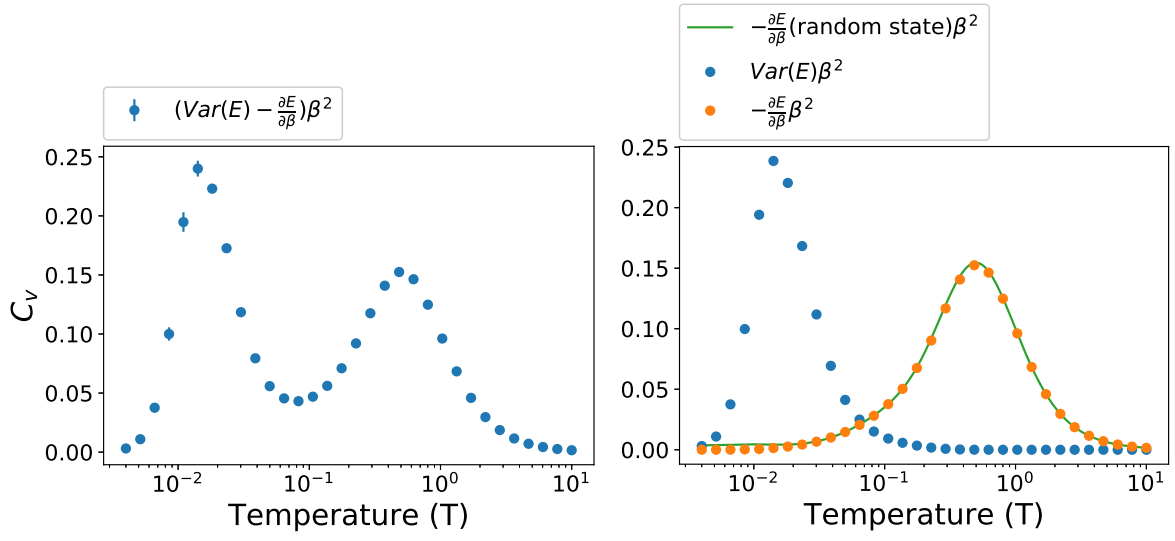


(a) (b)

Figure 5.7.: (a) Dirac points for the Fermi surface for the ground state described by Eq. 5.28 with $\alpha = 1$. (b) Phase diagram on the plane $J_x + J_y + J_z = 1$ for the zero flux state.

5.2.7. Specific Heat

Let's take a look at the specific heat for $\alpha = 1 (J_x = J_y = \frac{\alpha}{3}, J_z = 1 - \frac{2\alpha}{3})$. In Fig. 5.8a one can see that the specific heat becomes maximal at two distinct peaks.



(a) (b)

Figure 5.8.: Results of Monte Carlo simulation at $\alpha = 1$ with $L = 8$. (a) Specific heat defined in Eq. 5.20. (b) (Blue) The variance component of the specific heat and (orange) the derivative component. The green line is the derivative component calculated for a constant gauge configuration, instead of running a Monte Carlo simulation.

Next, the origin of the two peaks will be investigated. In order to do this, the specific heat is separated into its two components as in Ref. [32], variance $\mathrm{Var}(E) = \langle E_f^2 \rangle_{\mathrm{MC}} - \langle E_f \rangle_{\mathrm{MC}}^2$ and the energy derivative $-\frac{\partial E}{\partial \beta} = -\left\langle \frac{\partial E_f}{\partial \beta} \right\rangle_{\mathrm{MC}}$. In Fig. 5.8b, the two are plotted independently from each other. Each term is responsible for one peak. If instead of running an MC, the energy derivative is computed for only one state that is chosen randomly, a similar peak appears. This is why in further calculations we will focus only on the first peak of the specific heat.

5.3. Computational Complexity

Monte Carlo simulations of the Kitaev model have two major difficulties with large lattice sizes L. First, the computational complexity of the free energy scales with $\mathcal{O}(L^6)$ and

second, the autocorrelation time $\tau_{\text{metro}}(L)$ of the Metropolis algorithm increases with bigger L. In total, the complexity is $\mathcal{O}(\tau_{\text{metro}}(L)L^6)$. This means that, with our setup, and with parallelization at different temperatures, simulations for Metropolis $L = 16$ already take around 40 hours. With $L = 24$ it is even hard to know how many Metropolis steps are needed for it to converge properly.

The CRBM tackles the slowness of the Metropolis algorithm. Instead of having to do expensive Metropolis steps, we do a lot of cheap CRBM steps (convolution) $\mathcal{O}(L^2)$ until the original state is uncorrelated to the new state. This new state is corrected through Parallel CRBM Tempering. The correction step involves computing the expensive free energy $\mathcal{O}(L^6)$. This means that if the CRBM is close enough to the physical distribution, Parallel RBM Tempering has complexity $\mathcal{O}(L^6 + \tau_{CRBM}(L)L^2)$.

We know that $\tau_{CRBM}(L) < \tau_{\text{metro}}(L)$, as CRBM does global updates. So we conclude that a well trained CRBM will do the Monte Carlo with complexity $\mathcal{O}(L^6)$ instead of $\mathcal{O}(\tau_{\text{metro}}(L)L^6)$. This is not the case if the CRBM is not well trained. A more detailed description of the errors of the CRBM can be found in Sec. 3.4.2.

5.4. Monte Carlo Simulations of the Kitaev Model

The temperature is in units of the interaction strengths J_α and $\hbar = k_{\text{B}} = 1$. The usual MC simulation is performed following Ref. [29] using periodic boundary conditions in the z-direction. However, in the x-y-direction, periodic boundary conditions lead to an unwanted term in the Hamiltonian, which is in agreement with our analysis in Sec. 5.2. As proof of concept, we will first focus on using periodic boundary conditions for both directions as to not break translation invariance. Afterward, an open boundary condition is explored.

It is important to note that the free energy of the Kitaev model is expensive to compute, which lies in contrast with the Ising model were computing the energy is cheaper than one CRBM Gibbs step. For the Kitaev model, computing the free energy with our setup is as expensive as 10^3 CRBM Gibbs updates at $L = 8$. Note that the calculations performed for this chapter were done with the slower of the two methods to compute the free energy. The faster method (App. B.2), which we did not discover at the time of writing this chapter, is employed in the next chapter. For this faster method, 160 Gibbs updates are equivalent

to computing the free energy one time at $L = 8$. The amount of Gibbs steps performed for each parallel tempering (PT) step can be found in App. A.6. Experiments were run on a computer with an Intel(R) Xeon(R) CPU E5-2620 v4 running at 2.10GHz.

5.4.1. Periodic Boundary Conditions

For each temperature, a different CRBM with kernel size $4 \times 6 \times 6$ was trained at $L = 8$, as it is still fast to compute the free energy at this lattice size. This means that a 6×6 field of u^z_{ij} interact. The trained kernel for $T = 0.3$ can be seen in Fig. 5.9. The effective interaction in the z-direction is of shorter range than in the x-y-direction.

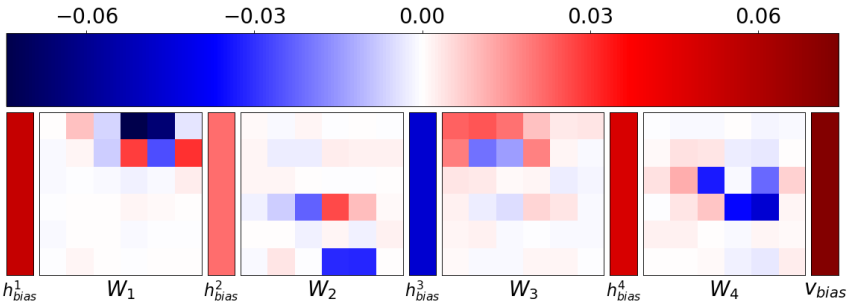
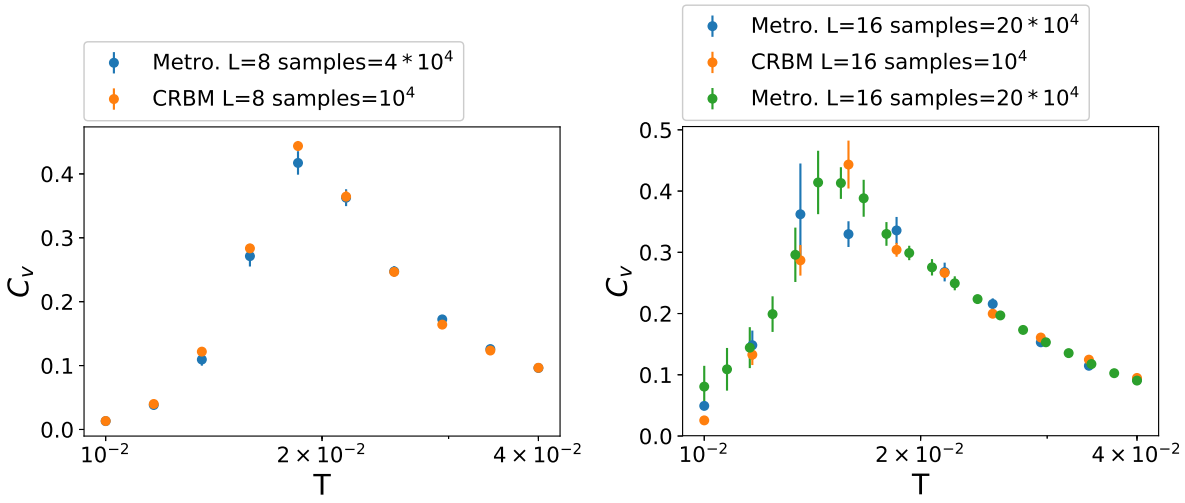


Figure 5.9.: Convolution kernels W_i trained at $L = 8$ for the Kitaev model at $T = 0.3$ and $\alpha = 1$ with periodic boundary conditions. See Eq. 3.21 for more details. The lattice points u^z_{ij} interact with range 4×3.

Next, we analyze the specific heat at different temperatures. We focus only on the first of the two peaks in the specific heat since the second one does not pose any problem for the MC. Note that Metropolis sampling was performed using parallel tempering. For the CRBM, a modified version of parallel tempering is used PRBM (see App. 3.4.1).

For the trained lattice size $L = 8$ (see Fig. 5.10a), both the CRBM and Metropolis match, however, in order to get a small enough error, the Metropolis MC needs 4 times more samples than the CRBM. Nevertheless, the CRBM has a much smaller error as shown in Fig. 5.10a. This is because each sample for the CRBM is almost completely uncorrelated with the previous one.



(a) (b)

Figure 5.10.: Specific heat on lattice size (a) $L = 8$ and (b) $L = 16$ for the Kitaev model with periodic boundary conditions zoomed around the first peak with $\alpha = 1$ at different temperatures. (Blue) Metropolis parallel tempering with 10 temperature points. (Orange) CRBM parallel tempering with 10 temperature points. (Green) Metropolis parallel tempering with 20 temperature points.

For $L = 16$, Metropolis MC acquires problems in converging at temperatures close to the peak as can be seen in Fig. 5.10b. In order to better take advantage of parallel tempering, so that the results are more stable, the temperature points have to be increased by a factor of two. In contrast, the CRBM works well with 20 times fewer samples and half as many temperature points, even though it was trained at $L = 8$. Note that the use of a fully connected RBM would not be possible here due to the ever increasing need for samples due to the larger lattice size. Not only that, but it would also need to learn from the poorly performing Metropolis.

5.4.2. Open Boundary Conditions

To treat the open boundary conditions we adapt the CRBM by using the Local Convolutional Restricted Boltzmann Machine (LCRBM) (see Sec. 3.3.3). Our LCRBM can learn the free energy of a system with open boundary conditions in x-y-direction and periodic boundary conditions in the z-direction. The LCRBM is similar to the CRBM in that in the middle of the lattice we assume translation invariance, i.e. the weights are shared. The two models are however different in that for the LCRBM new unshared kernels are

introduced at the edges. For our purposes, three unshared kernels are introduced on the left and right sides to learn boundary effects. Instead of training a new LCRBM, we use the values of the CRBM used for periodic boundaries and then retrain it at $L = 12$, which converges after only a few iterations. The kernels before training and after are very similar. For $L = 16$ the specific heat is compared between LCRBM and Metropolis (see Fig 5.11a).

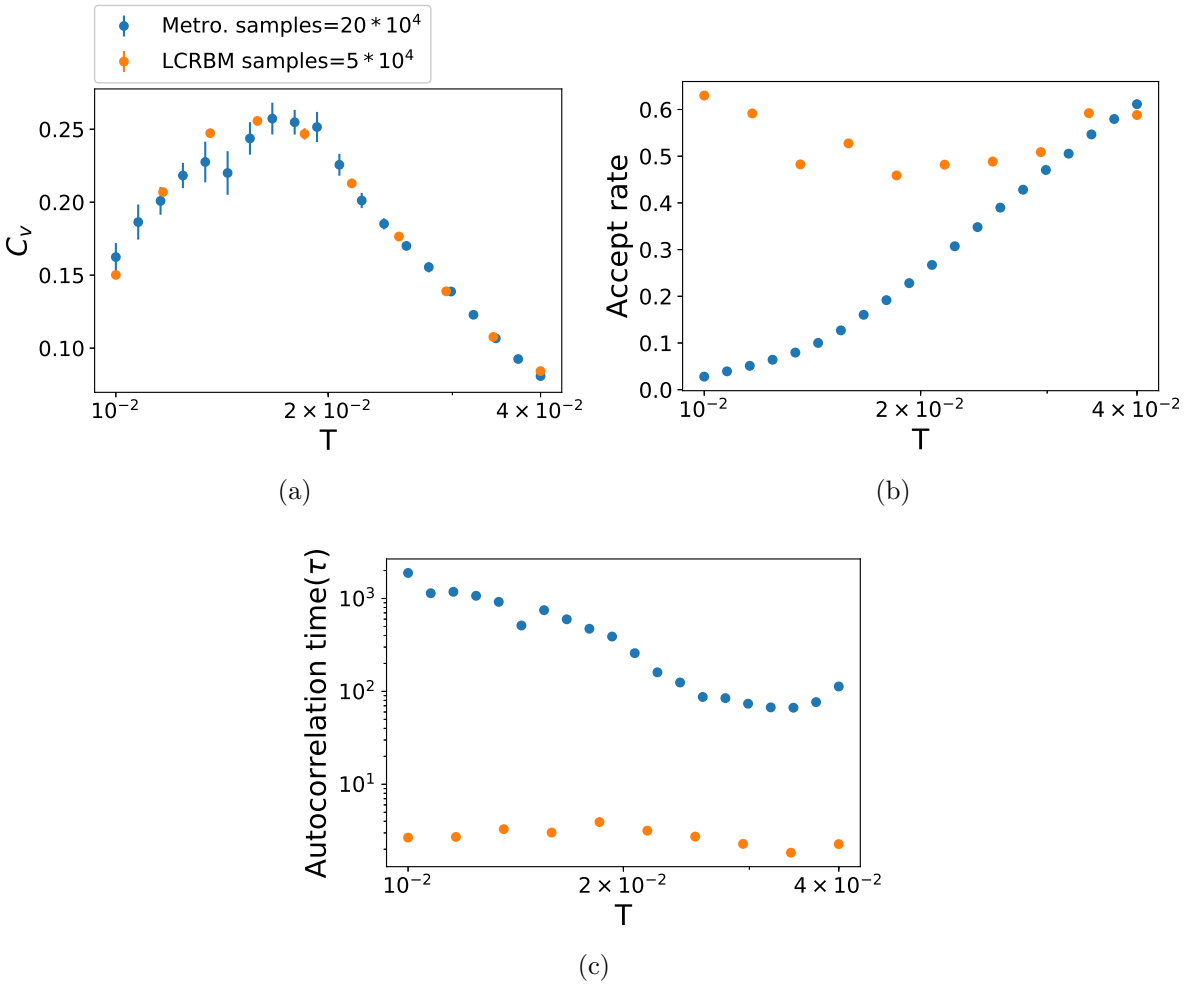
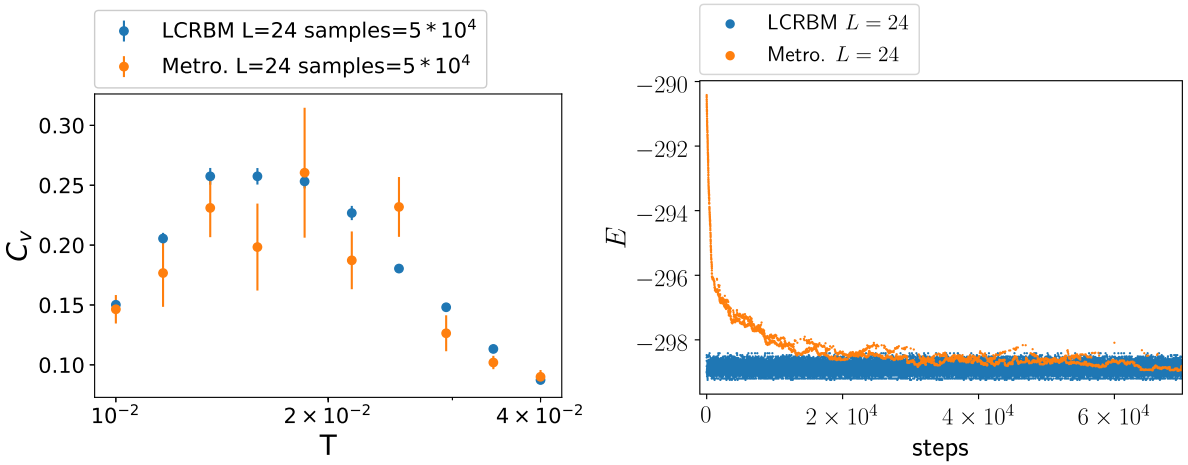


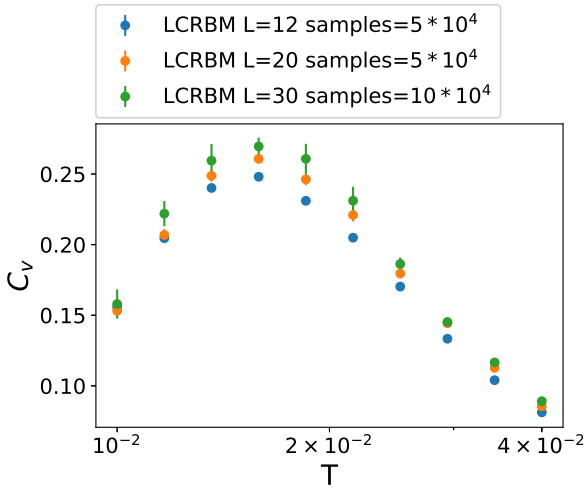
Figure 5.11.: Kitaev model with $\alpha = 1$ and $L = 16$ with open boundary conditions. (Blue) Metropolis parallel tempering with 20 temperature points. (Orange) CRBM parallel tempering with 10 temperature points with (a) Specific heat. (b) Acceptance rate of the parallel tempering exchange between the physical distribution and the LCRBM. (c) Autocorrelation time of the energy.

The LCRBM has an extremely low error even though eight times fewer number of computations were performed. This is especially noticeable at lower temperatures and close to the peak. Another aspect of interest is the acceptance rate and the autocorrelation time (see Fig. 5.11b,c). The acceptance rate for Metropolis increases from almost zero to 0.6 with rising temperatures. This is expected, as bigger β translate directly to smaller accep-

tance rates. For the LCRBM the acceptance rate is not strongly temperature dependent, as it is only influenced by how well the LCRBM fits the physical probability distribution. One has to keep in mind that each time a state is accepted by Metropolis, only one variable has been flipped. In contrast, each state that was accepted from the LCRBM is almost independent of its predecessor. This is confirmed by the autocorrelation time. The LCRBM has an average of ~ 30 times smaller τ. Taking into account that more temperature points are needed, this is equivalent to a ~ 60 times faster simulation.



(a) (b)



(c)

Figure 5.12.: (a) Specific heat around the first peak with $\alpha = 1$ and $L = 24$ at different temperatures with open boundary conditions. (b) Energies of samples during the MC, for Metropolis and the LCRBM, at $T = 10^{-2}$. (c) Specific heat around the first peak with $\alpha = 1$ at different temperatures with open boundary conditions using LCRBM parallel tempering at $L = \{12, 16, 20, 24\}$.

Increasing the lattice size to $L = 24$, the Metropolis MC does not converge, whereas the LCRBM converges easily (see Fig. 5.12a). For Metropolis the warm up phase had to be increased to 5×10^4. This can be traced back to the high autocorrelation time. In Fig. 5.12b the energies during the MC at low temperatures are shown. The Metropolis MC takes a long time until equilibrium is reached and one can see strong correlations between samples. In contrast, the LCRBM is in equilibrium after the first LCRBM parallel tempering step, as one LCRBM parallel tempering step consists of 10^3 Gibbs steps and one corrections step. Also, the samples seem to be mostly uncorrelated to each other.

The specific heat for $L = 12, 16, 20, 24$ as shown in Fig. 5.12c only show small differences between each other in agreement with Motome *et al.* [32]. For 2D there is no phase transition, as the specific heat is not singular at $L \to \infty$.

5.5. Summary

We showed that not only can a CRBM reproduce thermodynamic observables accurately, but it can also produce results with smaller errors for both periodic and open boundary conditions for the Kitaev model. The LCRBM, which was introduced for the open boundary case, achieves around 60 times faster simulation at $L = 16$. Simulations at $L = 30$, which were out of our reach with the Metropolis method could be simulated thanks to the LCRBM.

Our results confirm that the Kitaev model does not possess a phase transition. This makes the analysis uninteresting from a physical point of view, as increasing the lattice size does not change the specific heat. In the next chapter a new term, which will lead to a phase transition is added to the Hamiltonian.

CHAPTER 6

Extension of the Kitaev Model

In this chapter, a term is added to the Hamiltonian of the Kitaev honeycomb model as in Ref. [3], which leads to a phase transition. This extended Kitaev model will be analyzed with the help of the CRBM. Accordingly, the two main objectives of this chapter are: First, to apply and evaluate the methods of Local Metropolis RBM and the Parallel RBM Tempering in order to find out which one performs best under the harder circumstances of a phase transition. While doing this, the possibilities of beta interpolation and retraining are introduced.

The second objective is to show that the CRBM can sample at larger lattice sizes even with a phase transition. Thanks to the simulations at larger lattice sizes, we were able to compute critical exponents, which could not be achieved with Metropolis.

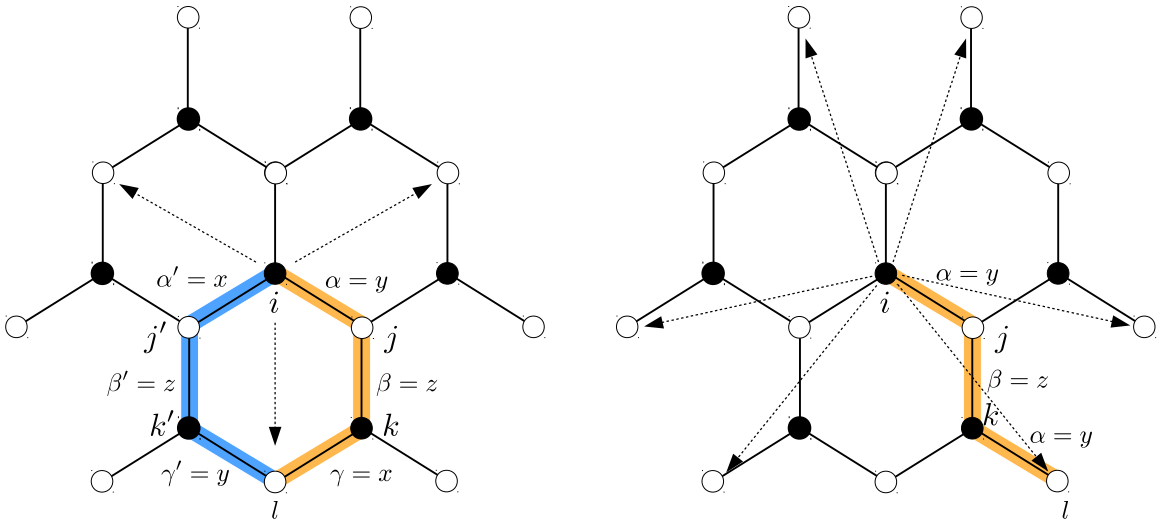
6.1. The Hamiltonian

Until now, the Kitaev honeycomb model has not been observed experimentally. If it is observed, other spin liquid phases with other properties might be seen, which is why extensions are useful. One of these extensions adds a four spin interaction to the Hamiltonian $\mathcal{H} = \mathcal{H}_{\text{kitaev}} + \mathcal{H}_{K_3}$ [3]. This extension is of interest as new ground states arise, and the Hamiltonian still can be rewritten in terms of gauge fields so that a classical Monte Carlo

simulation can be performed. The extension is:

$$\mathcal{H}_{K_3} = K_3 \sum_{\langle ijkl\rangle_{\alpha\beta\gamma}} \sigma_i^\alpha \sigma_j^\gamma \sigma_k^\alpha \sigma_l^\gamma - K_3' \sum_{\langle ijkl\rangle_{\alpha\beta\alpha}} \sigma_i^\alpha \sigma_j^\gamma \sigma_k^\gamma \sigma_l^\alpha. \tag{6.1}$$

For the first term, $\langle ijkl\rangle_{\alpha\beta\gamma}$ is the path from particle i to particle l through particle j and k. $\alpha(\beta, \gamma)$ is the type of connection between i and j (j and k, k and l). For example, in Fig. 6.1a, i and l are connected through $\langle ijkl\rangle_{yzx}$ and through $\langle ij'k'l\rangle_{xzy}$. The same principle applies to the second term. See Fig. 6.1b for an example of $\langle ijkl\rangle_{\alpha\beta\alpha}$.



(a) (b)

Figure 6.1.: Interactions of the extended Kitaev model. (a) i and l are connected through $\langle ijkl\rangle_{yzx}$ and through $\langle ij'k'l\rangle_{xzy}$. This represents the first term in Eq. 6.1. (b) i and l are connected through $\langle ijkl\rangle_{yzx}$. This represents the second term in Eq. 6.1.

Both terms are rewritten in the gauge basis using 5.26:

$$\begin{aligned}\sigma_i^\alpha\sigma_j^\gamma\sigma_k^\alpha\sigma_l^\gamma &= i^4(b_i^\alpha c_i)(\epsilon_{\gamma\alpha\beta}b_j^\alpha b_j^\beta)(\epsilon_{\alpha\beta\gamma}b_k^\beta b_k^\gamma)(b_l^\gamma c_l) \\ &= -i^4\epsilon_{\alpha\beta\gamma}^2 c_i c_l (b_i^\alpha b_j^\alpha)(b_j^\beta b_k^\beta)(b_k^\gamma b_l^\gamma) \\ &= ic_i c_l u_{ij}^\alpha u_{kj}^\beta u_{kl}^\gamma\end{aligned} \tag{6.2}$$

$$\begin{aligned}\sigma_i^\alpha\sigma_j^\gamma\sigma_k^\gamma\sigma_l^\alpha &= i^4(b_i^\alpha c_i)(\epsilon_{\gamma\alpha\beta}b_j^\alpha b_j^\beta)(\epsilon_{\gamma\beta\alpha}b_k^\beta b_k^\alpha)(b_l^\alpha c_l) \\ &= i^4\epsilon_{\gamma\alpha\beta}\epsilon_{\gamma\beta\alpha}c_i c_l (b_i^\alpha b_j^\alpha)(b_k^\beta b_j^\beta)(b_k^\alpha b_l^\alpha) \\ &= -ic_i c_l u_{ij}^\alpha u_{kj}^\beta u_{kl}^\alpha.\end{aligned} \tag{6.3}$$

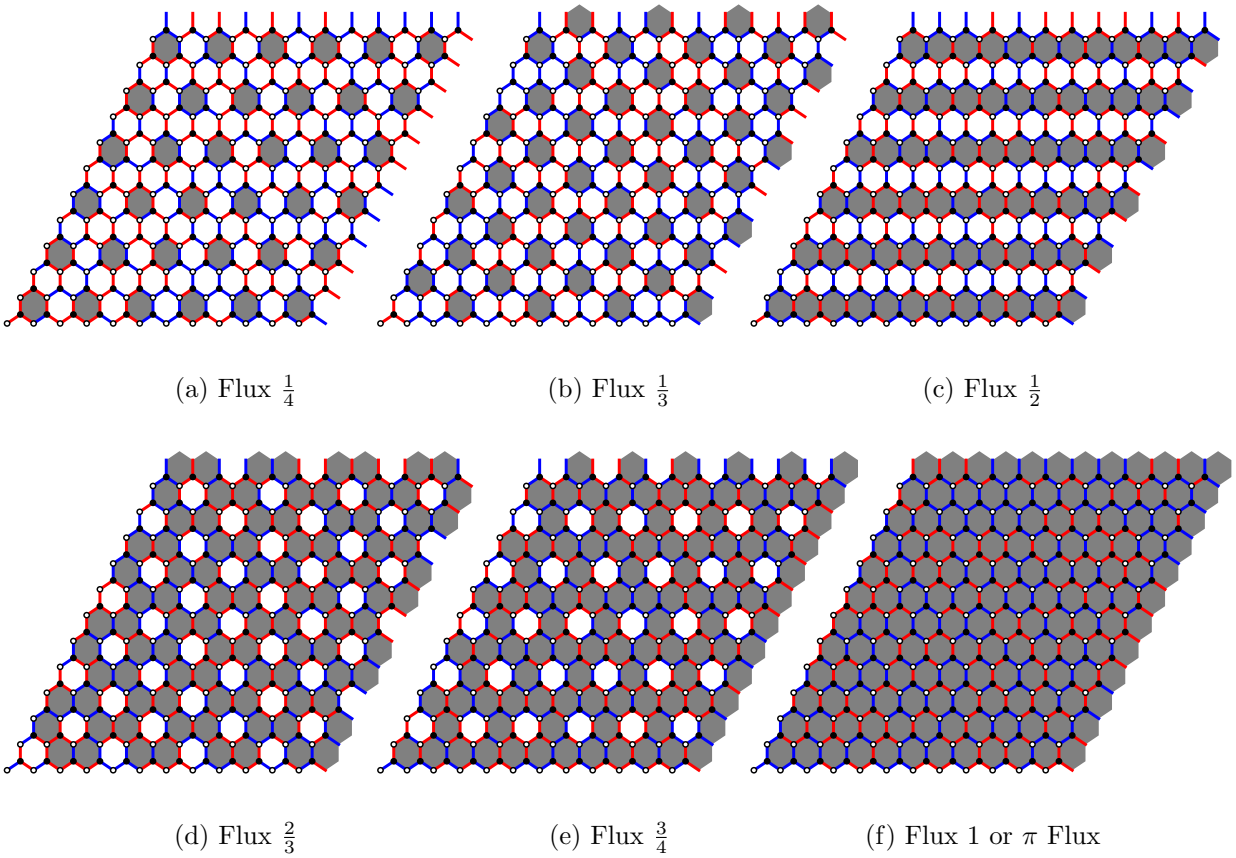
The final Hamiltonian coincides with Ref. [3]:

$$\mathcal{H} = iK_1\sum_{\langle ij\rangle} u_{ij}^\alpha c_i c_j + iK_3\sum_{\langle ijkl\rangle_{\alpha\beta\gamma}} u_{ij}^\alpha u_{kj}^\beta u_{kl}^\gamma c_i c_l + iK_3'\sum_{\langle ijkl\rangle_{\alpha\beta\alpha}} u_{ij}^\alpha u_{kj}^\beta u_{kl}^\alpha c_i c_l. \tag{6.4}$$

This Hamiltonian can now be diagonalized in the same way the original Kitaev model was. For a detailed description of how this is done numerically, see App. B.3. In Ref. [3], formalism 2 was used. In Sec. 5.2.2, we proved that formalism 1 and 2 are equivalent when using open boundary conditions in x-y-direction. Formalism 1 $u_{ij}^x = u_{ij}^y = 1$, with open boundary conditions in x-y-direction, will be used for Monte Carlo simulations.

6.2. Phase Diagram

Before running any Monte Carlo simulations on the extended Kitaev model, let us first take a closer look at how the ground state changes when changing parameters K_3 and K_3'. Six new ground states are found. The ground states are no longer flux 0 (all the Wilson loops are +1) like in the conventional Kitaev model. Since the new Hamiltonian introduces new interactions between three neighboring gauge fields, all ground states are either periodic in two or three. This is why all ground states have at least a periodicity of six (see Fig. 6.2).

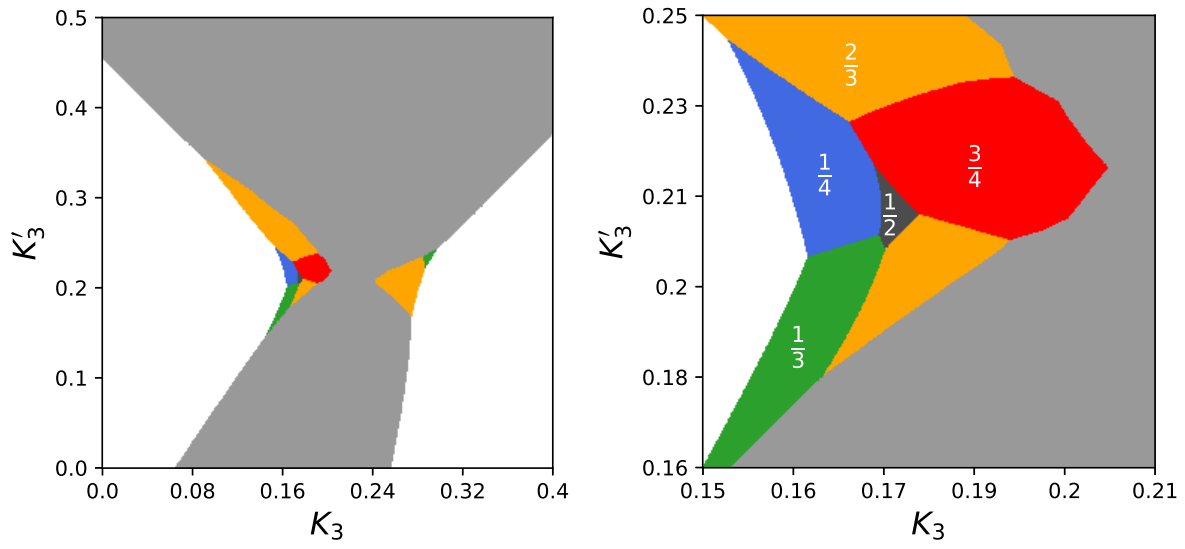


(a) Flux $\frac{1}{4}$ (b) Flux $\frac{1}{3}$ (c) Flux $\frac{1}{2}$

(d) Flux $\frac{2}{3}$ (e) Flux $\frac{3}{4}$ (f) Flux 1 or π Flux

Figure 6.2.: Ground states of the extended Kitaev model for $L = 12$ obtained from computing the phase diagram. Blue represents a gauge configuration component with -1 and red represents a +1 component. White Wilson loops represent +1 and grey loops represent -1.

These new ground states arise when the parameters of the Hamiltonian are tuned. They can be found in the region $K_3 = [0.15, 0.2], K_3' = [0.15, 0.25]$ (see Fig. 6.3).



(a) (b)

Figure 6.3.: Phase diagram with flux phases: 0 (white), $\frac{1}{4}$ (blue), $\frac{1}{3}$ (green), $\frac{1}{2}$ (dark grey), $\frac{2}{3}$ (yellow), $\frac{3}{4}$ (red), and 1 (grey), at different parameters K_3, K_3'.

Small differences can be observed when comparing the phase diagram to Ref. [3]. This might be due to the choice of boundary conditions or the choice of the lattice shape. With $K_3 = K_3' = 0$, the Hamiltonian is equivalent to the pure Kitaev model. This means that the flux of the ground state is 0, which is the dominant ground state. The flux 1 phase emerges when, due to the increased K_3, the two paths around the flux sector interfere constructively, and so $W_p = -1$ becomes more energetically favorable. Different flux patterns emerge at the edge between the flux 0 and the flux 1 phase when both K_3 and K_3' have similar values. How the phase diagram was obtained is further explained in App. C.1. In the following sections, different methods will be applied to the extended Kitaev model with flux $= \frac{3}{4}$ ($K_3 = 0.1, K_3' = 0.22$).

Note that free energies in this chapter are computed using the faster method (App. B.2).

6.3. Training

For the extended Kitaev model, 10 kernels with size 6×6 were trained at $L = 12$. First, 5×10^4 states were sampled with Metropolis. To lower the correlation between samples, only every fifth sample was used for training. This leaves 10^4 samples for training. Note

that the CRBM is first pre-trained using random states as described in Sec. 3.3.4. In Fig. 6.4, one of the trained kernels was plotted.

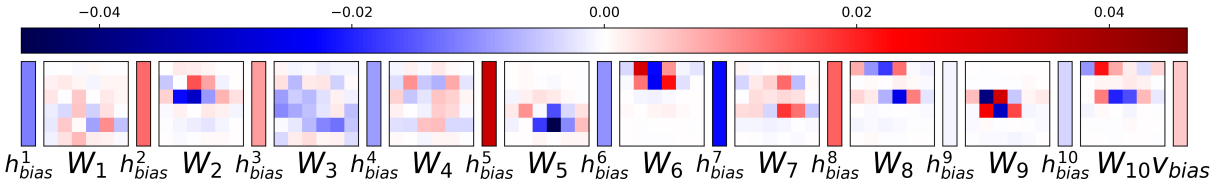


Figure 6.4.: Convolution kernels W_i trained at $L = 12$ for the extended Kitaev model at $T = 0.078$ in the flux $\frac{3}{4}$ phase. See Eq. 3.21 for more details. The lattice points u^z_{ij} interact with a field of 4×3 gauge fields.

One gauge field component seems to be interacting with a 4×3 field of neighbors. Note that W_1, W_3, and W_4 don't have strong features. It could be that the extended Kitaev model has a weak long-range interaction. However, it could just be an artifact that would disappear if we would have regularized the CRBM or would have trained fewer kernels.

6.4. Beta Interpolation

Until now, in order to be able to simulate a specific inverse temperature β with an RBM, it was necessary to first train it on states sampled at the specific β. There is a way to reuse RBMs trained for other temperatures. In order to do this, the free energy for the Kitaev model is rewritten to:

$$F_{\text{kitaev}}(u^z) = \sum_{\lambda} \left[\frac{\epsilon_\lambda}{2} + \frac{1}{\beta} \log\left(1 + e^{-\beta\epsilon_\lambda}\right) \right]. \tag{6.5}$$

Afterward, a temperature is introduced into the joint probability distribution of the RBM:

$$P_{\text{RBM}}(v, h) = \frac{e^{-\beta E(v,h)}}{Z}.$$

The RBMs free energy and probability distribution are:

$$F_{\text{RBM}}(v) = -\sum_j v^j_{bias} v^j - \frac{1}{\beta} \sum_i \log\left(1 + e^{\beta(h^i_{bias} + \sum_j W_{ij} v^j)}\right) \tag{6.6}$$

$$P_{\text{RBM}}(v) = \frac{e^{-\beta F_{\text{RBM}}(v)}}{Z}. \tag{6.7}$$

Due to the clear resemblance between the two free energies, if an RBM was trained for specific β, this RBM is expected to perform well even when β is changed slightly. Let's put this to the test. When an RBM for a specific β is not known, the closest trained RBM is taken and its inverse temperature is changed. This gives a new set of RBMs that are then used for sampling.

In Fig. 6.5, one can see that the interpolated RBMs yield a specific heat consistent with the trained RBMs. Note that the acceptance rate stays close to the one achieved with the RBM originally trained. Moving forward, beta interpolation will be used.

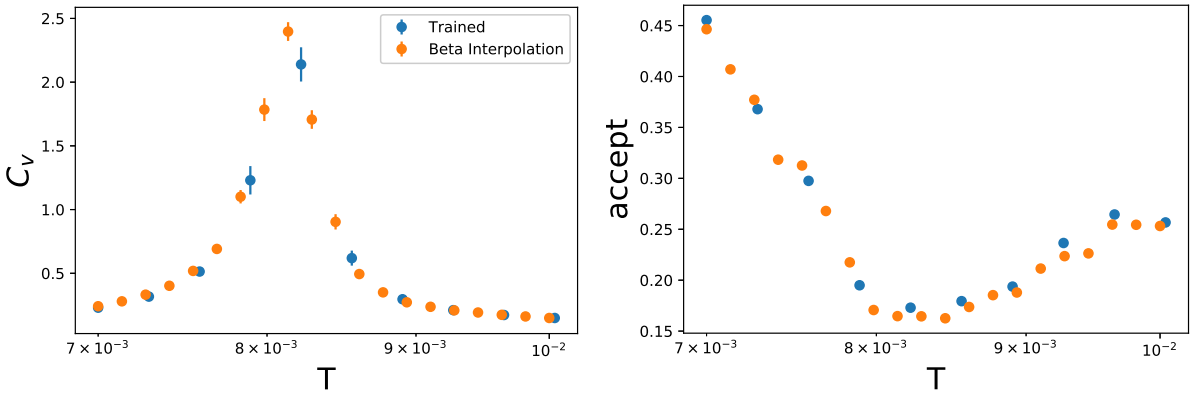


Figure 6.5.: Comparing trained vs interpolated RBMs at $L = 18$. The LCRBM was trained at $L = 12$.

6.5. Loss Prediction

First, let us remember the theoretical predictions from Sec. 3.4.2.3. It is expected that the acceptance rate decreases with lattice size as $A(L) = e^{-\Delta E L^2}$. The objective of this section will be to determine the ideal update size for the Local Metropolis RBM update (LMRBM). As an example, we will analyze simulations at $K_3 = 0.1, K_3' = 0.22$, which lies in the flux $\frac{3}{4}$ phase. As expected, the acceptance rate decreases when the lattice size is increased (see Fig. 6.6). The acceptance rate is at its lowest close to criticality.

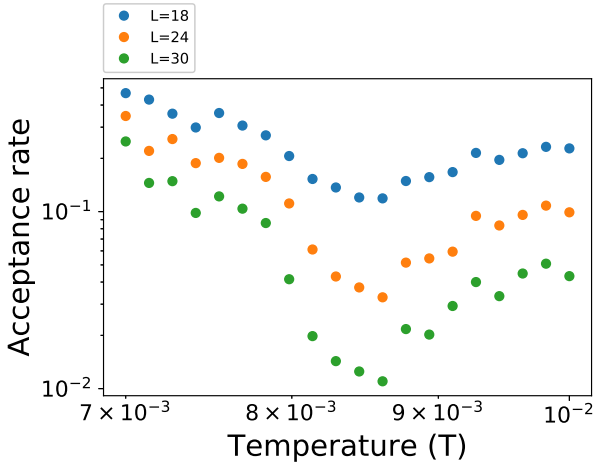


Figure 6.6.: Acceptance rates on lattice sizes $L = \{18, 24, 30\}$ close to T_c with 10^4 samples generated through Gibbs sampling at flux $\frac{3}{4}$.

Now, let's compute ΔE:

$$\frac{A(L_1)}{A(L_2)} = e^{-\Delta E(L_1^2 - L_2^2)}$$
$$\Delta E = \frac{\ln(A(L_2)) - \ln(A(L_1))}{L_1^2 - L_2^2} \tag{6.8}$$

and with $L_1 = 18$ and $L_2 = 24$, Fig. 6.7 is obtained.

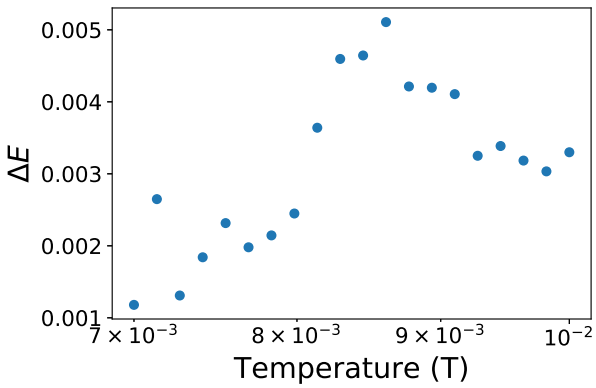


Figure 6.7.: Effective loss per lattice site calculated with Eq. 6.8.

As expected, the effective loss reaches its maximum near to the phase transition. Now, the acceptance rate for $L_3 = 30$ can be predicted using $A(L_3) = A(L_2)e^{-\Delta E(L_3^2 - L_2^2)}$:

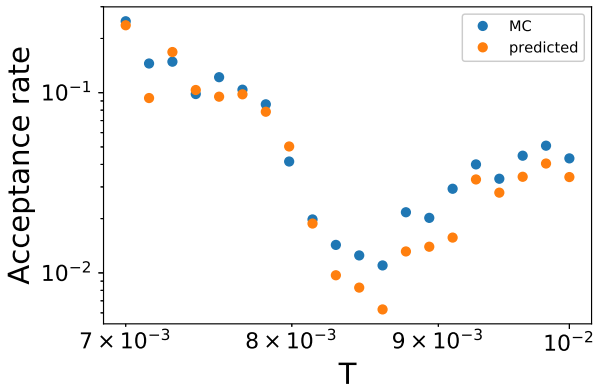


Figure 6.8.: The predicted and measured acceptance rate for $L = 30$ with flux $\frac{3}{4}$.

The predictions for the acceptance rate match with the ones observed (see Fig. 6.8). Having computed ΔE, the optimal update size for the Local Metropolis RBM update (LMRBM) can be calculated, $u = \frac{c}{\Delta E}$ (see Sec. 3.4.2.3). c is estimated to be between 1 and 2, which matches results obtained from simulations at $L = 12$ and $L = 18$. Note that many more simulations would need to be done to compute c exactly. In Fig. 6.9, the optimal u is calculated for both $c = 1$ and $c = 2$.

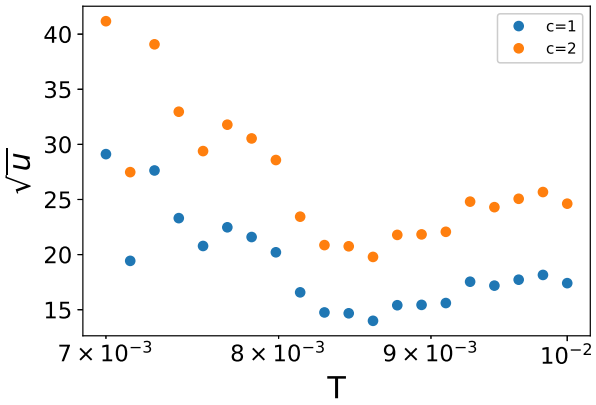


Figure 6.9.: Best expected update size u at different temperatures.

Close to criticality, the best update size seems to lie between 15^2 and 20^2. It is important to note that since the corrected states are exchanged through parallel tempering, bigger update sizes might be optimal. The small update size at temperatures away from criticality might slow down the simulation. For the following experiment, an update size of 15^2 is going to be employed.

6.6. Comparing Sampling Methods

In this section, the Parallel RBM Tempering Update (PRBM), the Metropolis RBM Update (MRBM), and the Local Metropolis RBM Update (LMRBM) are compared. MRBM and PRBM differ in that PRBM has two Markov chains for each temperature, one for the physical distribution and one for the RBM, while for the MRBM there is only one chain. The difference between the MRBM update and the LMRBM update is that for the latter only a set of U variables is updated in each step, while for the former all variables are updated. For the LMRBM, two approaches for the local update are compared. First, only a randomly placed square with dimension $\sqrt{u} \times \sqrt{u}$ is updated (window method), and second, a completely random set U is chosen (random method) and then updated. The amount of Gibbs steps performed between each parallel tempering (PT) step can be found in App. A.6.

Note that in order to compare the different sampling methods, different estimators were used. The slower the simulation, the noisier these estimators become. The autocorrelation time is the noisiest estimator, yet the most useful, and will thus be used when possible. Two estimators are more stable than the autocorrelation time: the error of the specific heat, and the error of the energy. Both are computed using batch means. The error of the specific heat is fourth-order and so it is noisier than the error of the energy, which is second-order. They will be used as alternatives when the autocorrelation time is not a good approximation for the accuracy of the simulation.

6.6.1. The L = 18 Case

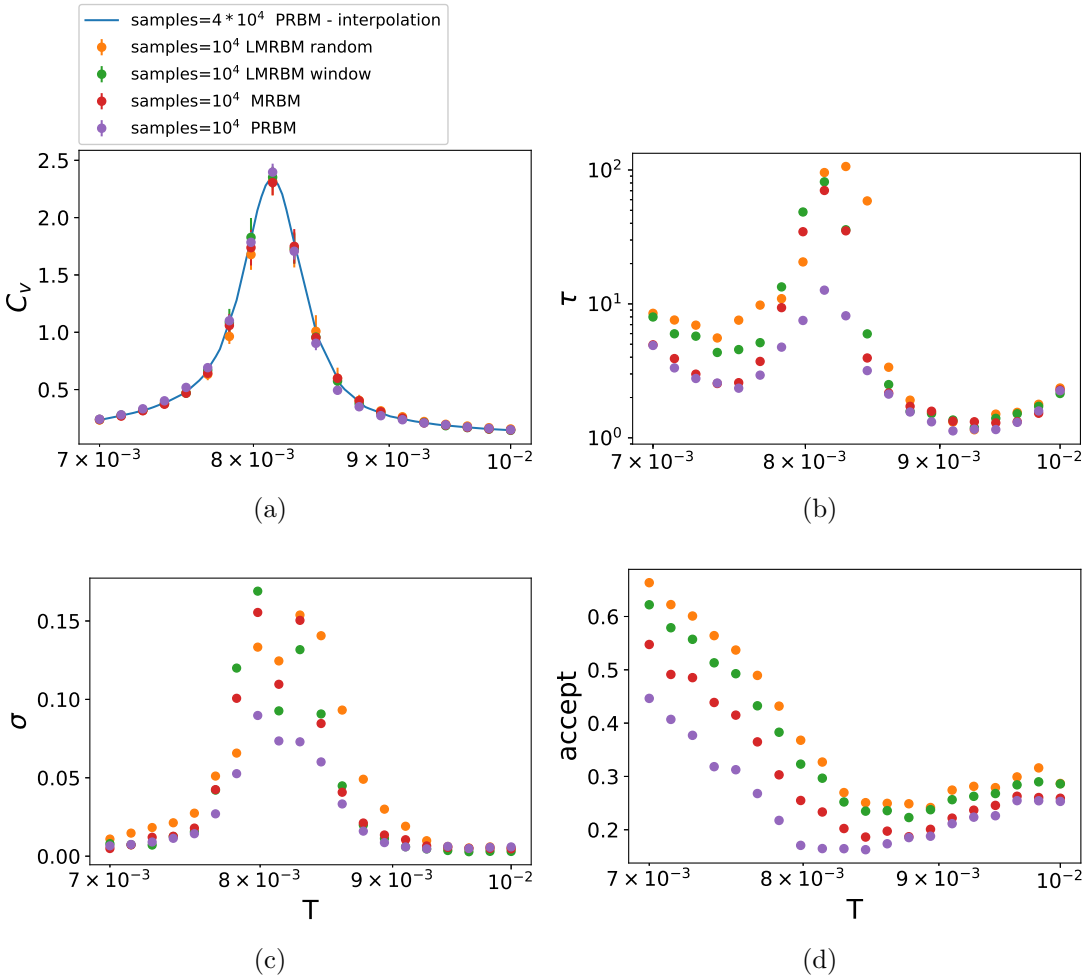


Figure 6.10.: Comparing PRBM and LMRBM at flux $\frac{3}{4}$ with $L = 18$. The update size for both LMRBM approaches is chosen to be $u = 15^2$ as suggested by the data compiled in the previous chapter. The blue line is an accurate simulation with more samples. (Orange) is computed using the window LMRBM method, (green) the random LMRBM, (red) the MRBM approach (equivalent to LMRBM with $u = 18^2$), and (violet) the PRBM. (a) Specific heat, (b) the autocorrelation τ, (c) the error bar size for the specific heat and (d) the acceptance rate.

In Fig. 6.10a, one can see that all methods show consistent values for the specific heat for $L = 18$. However, the performance of the methods is not the same. The autocorrelation time (see Fig. 6.10b) and the errors of the specific heat (see Fig. 6.10c) are similar for all approaches at high temperatures. At low temperatures, random LMRBM does worst followed by a similar performance for window LMRBM, MRBM, and PRBM. Close to the phase transition, PRBM performs best and the other methods perform similarly worse.

The key to understanding these results lies in the acceptance rate. In section 3.4.2.3, we

calculated that the optimal acceptance rate is $A_{\text{opt}} = e^{-c}$, where c lies around 2 close to the phase transition, and around 1 everywhere else. The acceptance rate should lie between 0.16 and 0.36. The closer to the phase transition, the lower should the ideal acceptance rate be. The PRBM yields acceptance rates closer to these values, which is why it performs best for $L = 18$. The PRBM yields a smaller acceptance rate, as states that are rejected are not discarded like in the LMRBM and MRBM methods. Instead, they are continuously updated. In the PRBM method, states that the RBM exchanges with the physical distribution have changed more and so are less correlated to each other. The less correlated the states are to each other, the larger their energy difference becomes and so the acceptance rate gets smaller. With larger lattice sizes, this will become a sizeable problem for the PRBM as the acceptance rate keeps getting smaller. At this size, however, it is the ideal update.

Comparing the LMRBM methods, random (orange), and window (green), it becomes clear why the window update performed better: the acceptance rate of the window approach is smaller. This means that states are less correlated after a window update compared to after a random update, which leads to better results. Important to note here is that the update size of $u = 15^2$ is too small, as the MRBM method, which is equivalent to $u = 18^2$ performs better than both LMRBMs. Since the two approaches perform similarly, when using LMRBM the random variant will be employed going forward.

6.6.2. The L = 24 Case

The update size of the LMRBM is adapted so that the acceptance rate is the same regardless of temperature. It is adapted during the warm up phase. Three methods are compared: PRBM and LMRBM with acceptance rates 0.22 and 0.28.

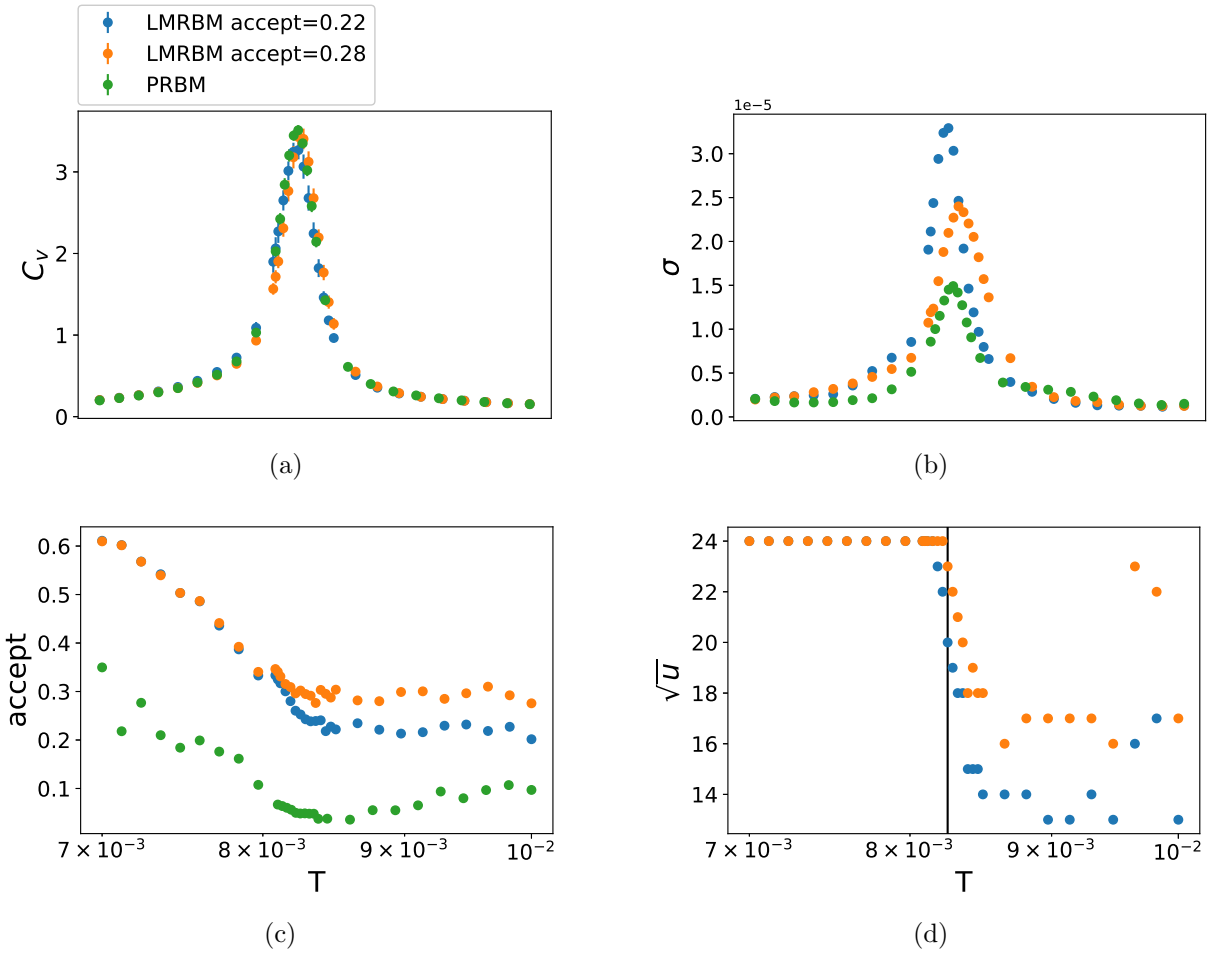
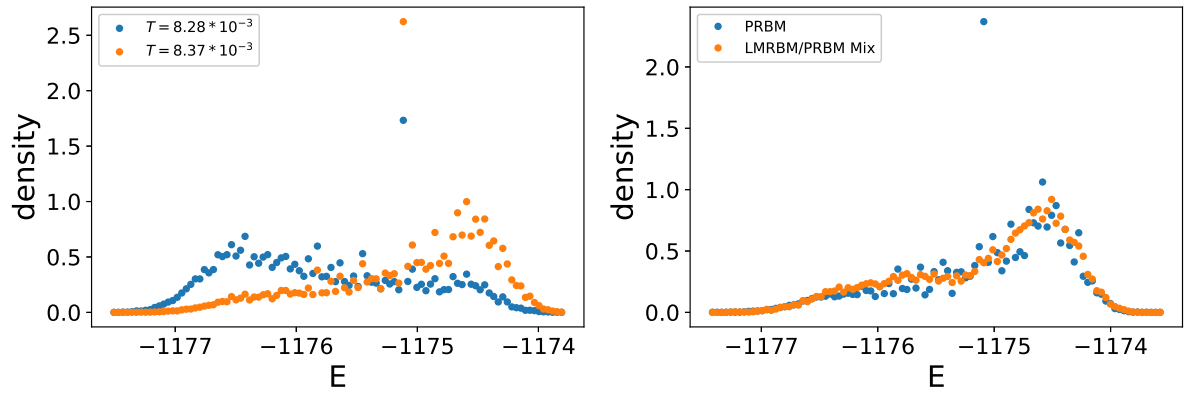


Figure 6.11.: Comparing PRBM and random LMRBM at flux $\frac{3}{4}$ with $L = 18$. The update size for both LMRBM approaches has been chosen such that the accept rate is constant. (Blue) The minimal acceptance rate is set to 0.22 and (orange) it is set to 0.28. For (green), the PRBM method is used. (a) Specific heat, (b) the error bar size for the mean energy, (c) acceptance rate and (d) update size. The horizontal line marks T_c. 4×10^4 samples were generated

In Fig. 6.11, the specific heat coincides for the three simulations. However, when looking at the error bars of the mean energy, one finds that PRBM still performs much better than the two LMRBM methods. The minimal acceptance rate of 0.28 performs best. As expected, the PRBM has a much lower acceptance rate than the LCRBM. What was not expected is that even though the minimum of the acceptance rate of the PRBM is 3.5%, it still performs best. It seems that the low acceptance rate is balanced out by the fact that states are not discarded. It is most difficult to run the MC close to T_c. The method that performs best close to criticality will do best. Close to T_c, one can see that the update size is close to 24^2. This means that the full state is being updated. We know that if the full state is updated, PRBM will do better than LMRBM, which coincides with the results

obtained. Note that the decreasing acceptance rate of the PRBM might lead to artifacts. This will be explored in the next section.

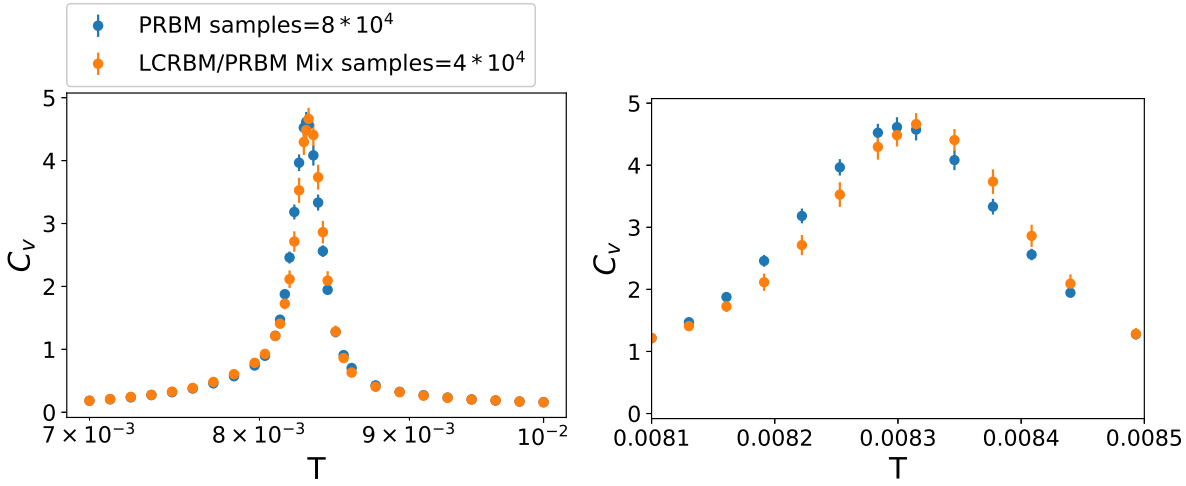
6.6.3. The L = 30 Case



(a) (b)

Figure 6.12.: Histogram of the energies obtained from MC. For (a) the PRBM before and after criticality and (b) comparing PRBM and LMRBM/PRBM Mix approach at $T = 8.28 \times 10^{-3}$.

To get a better sense of what is happening, the energy distribution of the MC at $L = 30$, the energy histogram, is analyzed (see Fig. 6.12a). There is a peak in the histogram at an unexpected energy $E = -1175.1$ at $T = 8.37 \times 10^{-3}$. Surprisingly, this peak appears also at other temperatures. After closer inspection, it is discovered that a state was "trapped" in parallel tempering. No exchange with this state was accepted. This can happen since the acceptance rate of the PRBM is so low. To solve this, a mixed method is introduced where the LMRBM and the PRBM methods are combined. From now onwards this mixed method will be referred to as the LMRBM/PRBM Mix. We take advantage of the faster Markov chain of the PRBM method, which has the downside of low acceptance rates and combines it with the slower LMRBM method, which has higher acceptance rates. The two methods are alternated. This removes artifacts as can be seen in Fig. 6.12b.

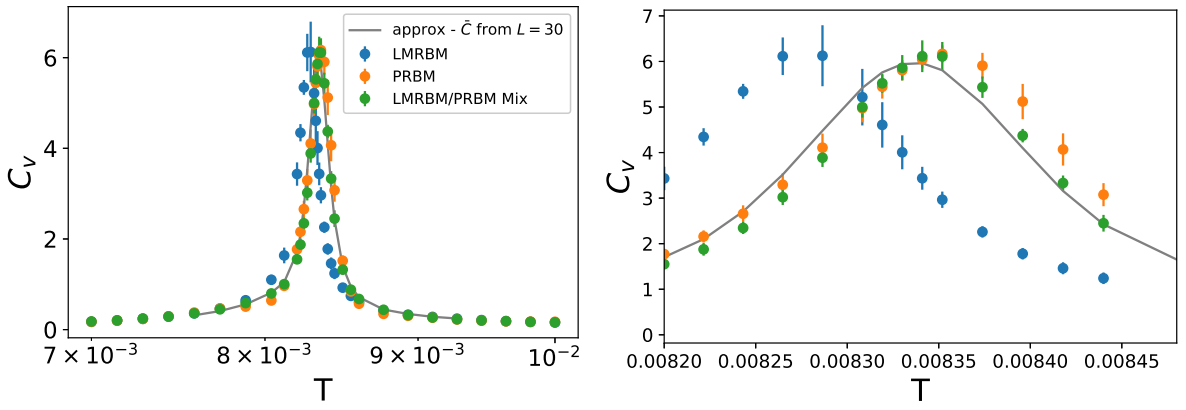


(a) (b)

Figure 6.13.: Comparing PRBM and LMRBM/PRBM Mix at flux $\frac{3}{4}$ with $L = 30$.

In Fig. 6.13a, one can see that the two methods yield slightly different results for the specific heat. Because of time constraints, the LMRBM/PRBM Mix method was not repeated with 8×10^4 samples and the LMRBM method was not tested at $L = 30$.

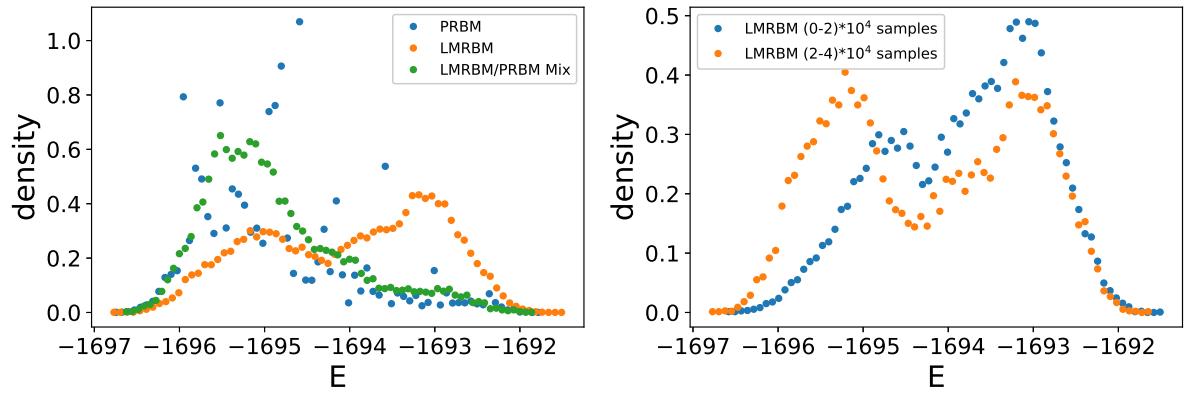
6.6.4. The L = 36 Case



(a) (b)

Figure 6.14.: Comparing PRBM, LMRBM and LMRBM/PRBM Mix at $L = 36$. 4×10^4 samples were generated. The black line is the specific heat approximated through finite-size scaling using the $\bar{C}$ obtained from $L = 30$ and applying it to $L = 36$.

PRBM, LMRBM, and LMRBM/PRBM Mix are compared for the $L = 36$ case. The grey line is the specific heat approximated through finite-size scaling using the $\bar{C}$ obtained from $L = 30$ and applying it to $L = 36$ (see App. A.5). The LMRBM/PRBM Mix approach has the lowest error and comes closest to the expected C_v. As expected, the PRBM method performs a bit worse but still produces coherent results. In contrast, the LMRBM method provides completely different values for the specific heat.



(a) (b)

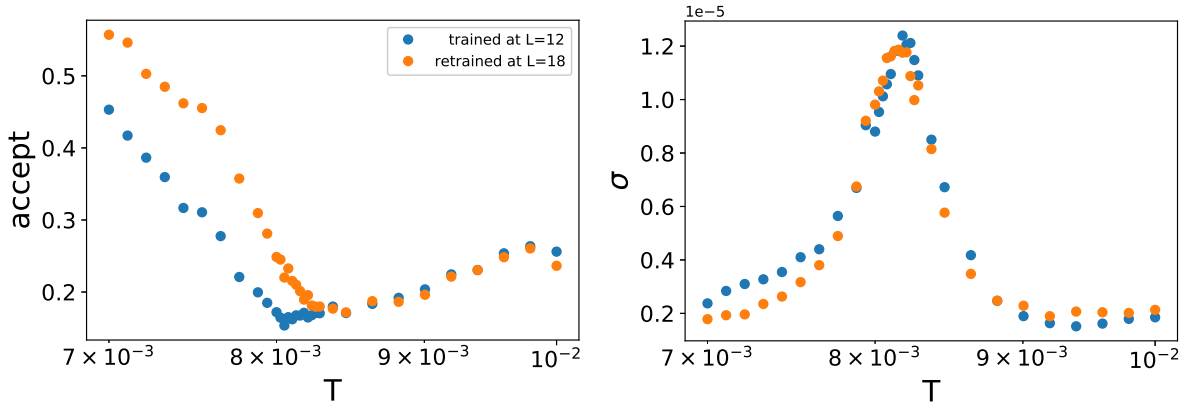
Figure 6.15.: Histogram of the energies obtained from the Monte Carlo simulations.
(a) Comparing PRBM and LMRBM/PRBM Mix approach at $T = 8.29 \times 10^{-3}$.
(b) Comparing the first 2×10^4 samples to the last 2×10^4 samples

The histogram of the energy tells a similar story to the one for $L = 30$. The PRBM method has a lot of trapped states that can not exchange with the RBM. This creates the spikes in Fig. 6.15a. The histogram for the LCRBM is shifted due to the slower Markov chain. In Fig. 6.15b the histogram of the first 2×10^4 samples is compared to the one obtained from the last 2×10^4 samples. It is clear that the Markov chain has not reached equilibrium for the LMRBM. Combining PRBM and LCRBM to the LMRBM/PRBM Mix method solves the challenges both approaches face.

6.7. Retraining

Until now, we have always employed LCRBMs trained at $L = 12$. It is reasonable to expect that if we trained an LCRBM at larger lattice sizes we could obtain smaller errors and bigger acceptance rates. The strategy is to sample at $L = 18$ from the LCRBM trained at $L = 12$, then fine-tune the LCRBM with the samples obtained. This can be repeated

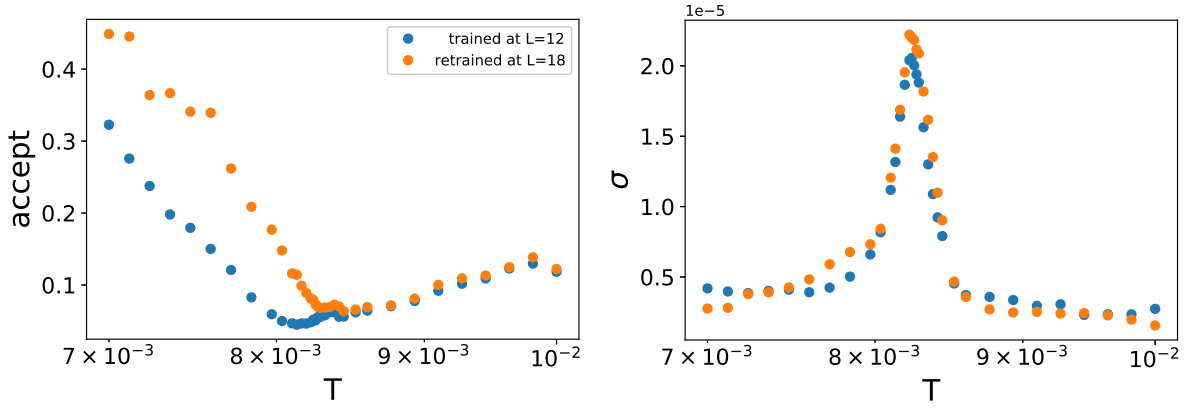
for different lattice sizes. Note that the learning rate is decreased from $\lambda = 10^{-3}$, which is used for training, to $\lambda = 10^{-4}$, which is used for retraining.



(a) (b)

Figure 6.16.: Comparing simulations for $L = 18$ made with an LCRBM trained at $L = 12$ (blue) and one retrained $L = 18$ (orange). PRBM was employed for statistical corrections.(a) The acceptance rate and (b) the error bars of the mean energy.

After retraining, the acceptance of the LCRBM increased before T_c but did not change after (see Fig. 6.16a). Our preliminary hypothesis is that in the ordered phase the LCRBM can continue to improve through training, which is probably a side effect of the fact that samples are less distinct in the ordered phase. This also affects the error bars of the mean energy, which decrease slightly in the ordered phase. The simulations are repeated for $L = 24$ with similar results (see Fig. 6.17). The only difference is that for this case the error actually increases slightly for the retrained LCRBM.



(a) (b)

Figure 6.17.: Comparing simulations for $L = 24$ made with an LCRBM trained at $L = 12$ (blue) and one retrained $L = 18$ (orange). PRBM was employed for statistical corrections.(a) The acceptance rate and (b) the error bars of the mean energy.

It could well be that the retrained LCRBM does not sample better than the one trained at $L = 12$, but that the higher acceptance rate is an artifact stemming from an increased autocorrelation time. In Sec. 3.3.5, it was established that a smaller L2 norm will lead to smaller autocorrelation times for the RBM.

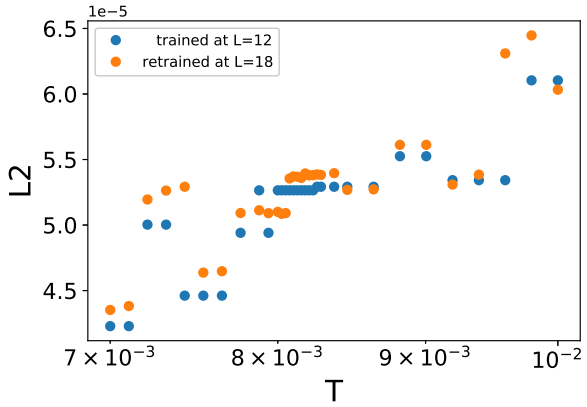


Figure 6.18.: L2 norm of kernels trained at $L = 12$ and retrained at $L = 18$.

Fig. 6.18 shows that retraining the LCRBM increased its L2 norm. This supports the claim that the retrained LCRBM does not perform significantly better, contrary to what would be expected from the acceptance rate. For the next sections, the LCRBM retrained at $L = 18$ will be used, but no further retraining will be done as further training will only bias results.

6.8. Calculating the Critical Temperature

In this section, the simulations made with the RBM are compared to the ones made with Metropolis for $L = 18, 24$. Since Metropolis can not sample at system sizes larger then $L = 24$, the specific heat is approximated through finite-size scaling for $L = 30, 36$. This method is further explained in App. A.5.

6.8.1. The L = 18 Case

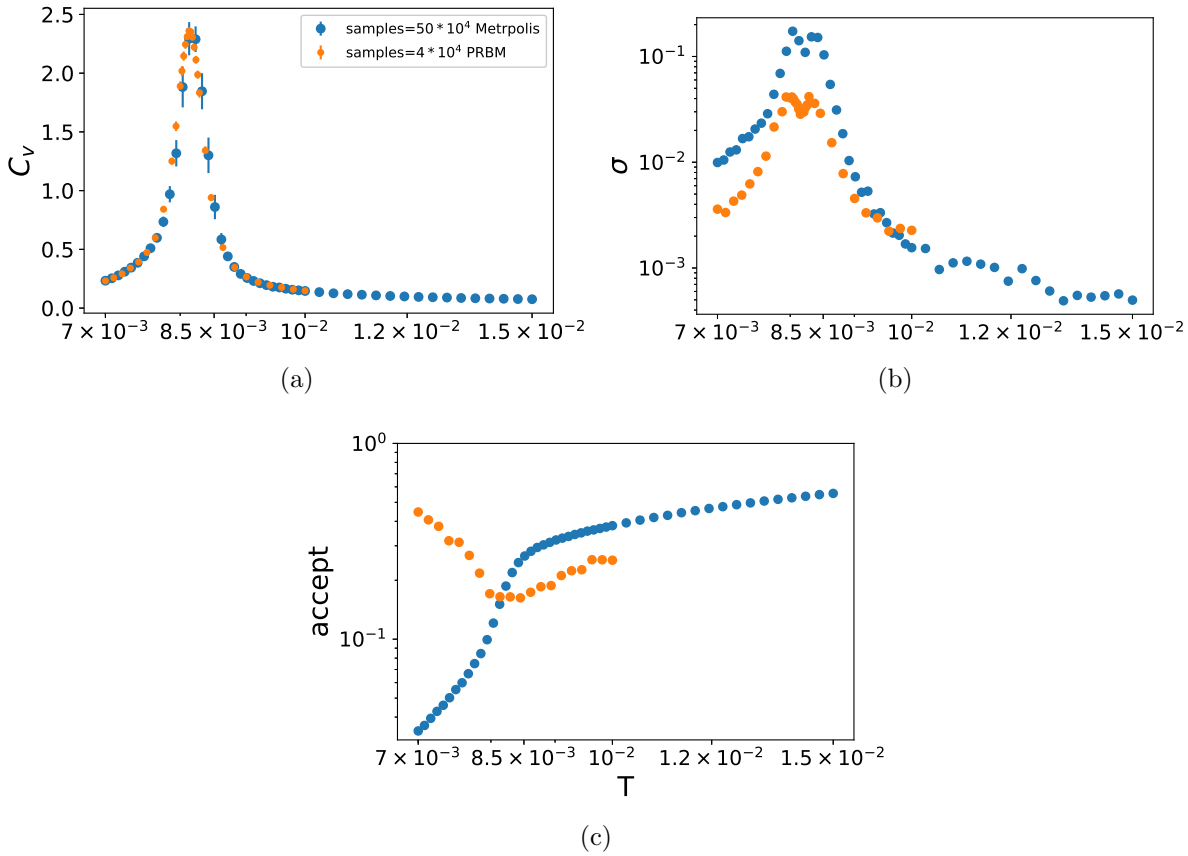
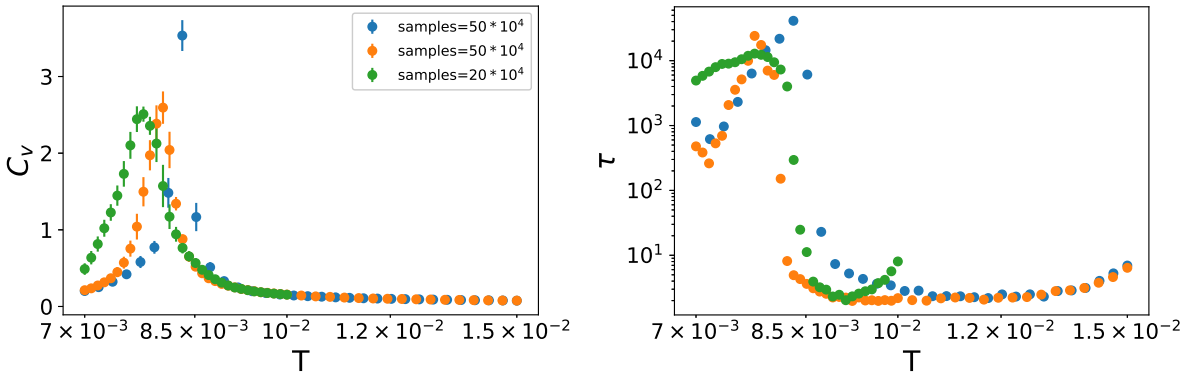


Figure 6.19.: Comparing Metropolis (blue) and PRBM (orange) at flux $\frac{3}{4}$ with $L = 18$. Metropolis has been sampled ~ 10 more times and uses more temperature points. (a) Specific heat, (b) the error bar size of the specific heat and (c) the acceptance rate.

To sample with the Metropolis method, additional temperatures that are not relevant for this study need to be simulated. This is done in order to take advantage of parallel tempering, as acceptance rates are low at small temperatures (see Fig. 6.19c). Even then, a

ten times longer simulation needs to be performed. The RBM not only matches the specific heat, but the error bars are so small that the marker size needed to be decreased. The error bars for the RBM are around four times smaller than for Metropolis (see Fig. 6.19b).

6.8.2. The L = 24 Case



(a) (b)

Figure 6.20.: Three different Metropolis simulation, at $L = 24$ and flux $\frac{3}{4}$, at different temperatures. (a) Specific heat and (b) integrated autocorrelation time. Depending on the choice of temperatures, different T_c are obtained. This is due to the huge autocorrelation time.

Performing Metropolis at $L = 24$ is very hard because a lot of temperature points and update steps are required so that the Markov chain converges. Moreover, these steps are $(\frac{24}{18})^6 = 5.6$ times more expensive than for $L = 18$. Three different Monte Carlo simulations were performed varying both the temperature range that was simulated and the number of temperature points used. The specific heats do not coincide. The autocorrelation time close to the critical temperature lies around 10^4. Given that 50×10^4 samples are generated, this yields optimistically 50 uncorrelated samples. Interestingly, the blue simulation in Fig. 6.20 coincides with the results obtained with the LCRBM (see Fig. 6.21). Further simulations, adjusting parameters in parallel tempering, would have to be made in order to understand why the blue simulation performs better. It is proposed that since the temperature points are more spread out, parallel tempering exchange acceptance rates are lower. This might increase the probability that a state from the disordered phase exchanges with the ordered phase. Ref. [33] concludes that to prevent this, more Metropolis updates need to be performed between parallel exchanges.

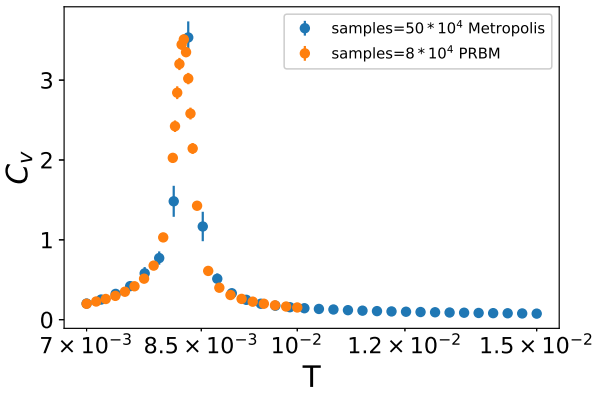


Figure 6.21.: The Metropolis and the PRBM method are compared for $L = 24$.

The LCRBM retrained at $L = 18$ was used. It is a good sign that one of the Metropolis simulations coincides with the LCRBM, which has no problem sampling at $L = 24$.

6.8.3. The L = {30, 36} Case

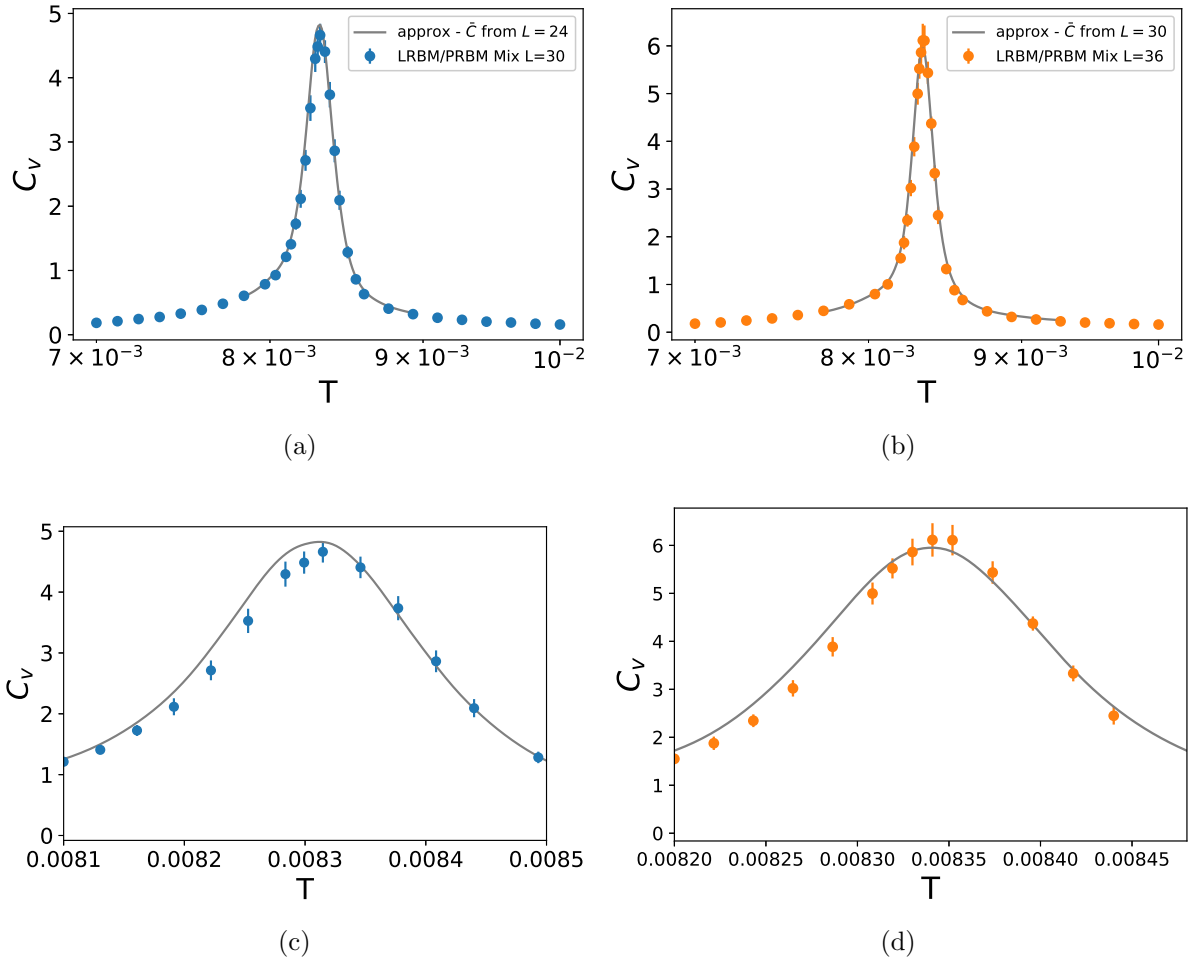


Figure 6.22.: Specific heat (grey) approximated through finite-size scaling using the $\bar{C}$ obtained from (a,c) $L = 24$ and applied at $L = 30$ and (b,d) obtained from $L = 30$ and applied at $L = 36$. The MC calculations for both where performed with the LMRBM/PRBM Mix method.

The grey line in Fig. 6.22 is the specific heat approximated through finite-size scaling using the $\bar{C}$ obtained from a smaller lattice size (see App. A.5 for more details).

The specific heat values for both $L = 30, 36$ coincide roughly with the one predicted by finite-size scaling. Remember that the lowest acceptance rate for $L = 30$ was 2% and for $L = 36$ it is 0.5%. This is why we used the better LMRBM/PRBM Mix method (see Sec. 6.6.3). However, even when using the Mix method for $L = 36$, the error bars continue to be quite large. To get more accurate results, a longer simulation would need to be run. Note that performing the $L = 36$ simulations takes 6 days with our setup.

Accordingly, computing the specific heat for larger lattice sizes would be hard. Experiments were run on a computer with an Intel(R) Xeon(R) CPU E5-2620 v4 running at 2.10GHz.

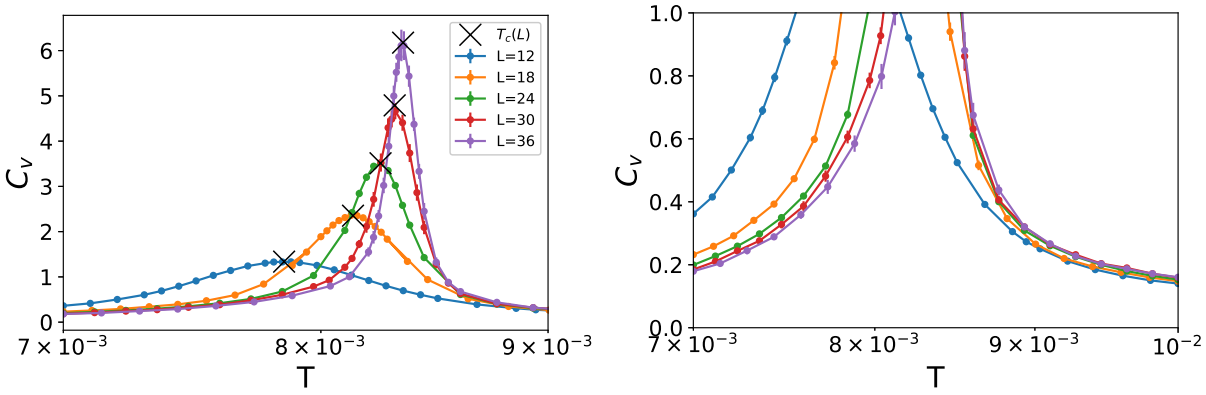
6.9. Finite-Size Scaling

As discussed in Sec. 3.2, the critical temperature and maximal specific heat at finite lattice sizes are:

$$T_c(L) - T_c(\infty) \propto L^{-\frac{1}{\nu}} \tag{6.9}$$

$$C_v^{\max}(L) \propto L^{\frac{\alpha}{\nu}}. \tag{6.10}$$

In Fig. 6.23, the specific heat values for $L = \{12, 18, 24, 30, 36\}$ are plotted. One can see that values for the specific heat slowly converge far from T_c.



(a) (b)

Figure 6.23.: (a) The specific heat for different lattice sizes. The black crosses are the expected values for $T_c(L)$ that will be computed in this section. (b) Zoom in to show the convergence of the specific heat far from $T_c(L)$.

The critical temperature $T_c(L)$ and the maximum of the specific heat $C_v^{\max}(L)$, which are obtained from the previous section, are fitted with weighted least squares to Eq. 6.9 and Eq. 6.10. The maximum of the specific heat was obtained by fitting a Gaussian curve to temperatures close to $T_c(L)$. This is explained further in App. A.4.

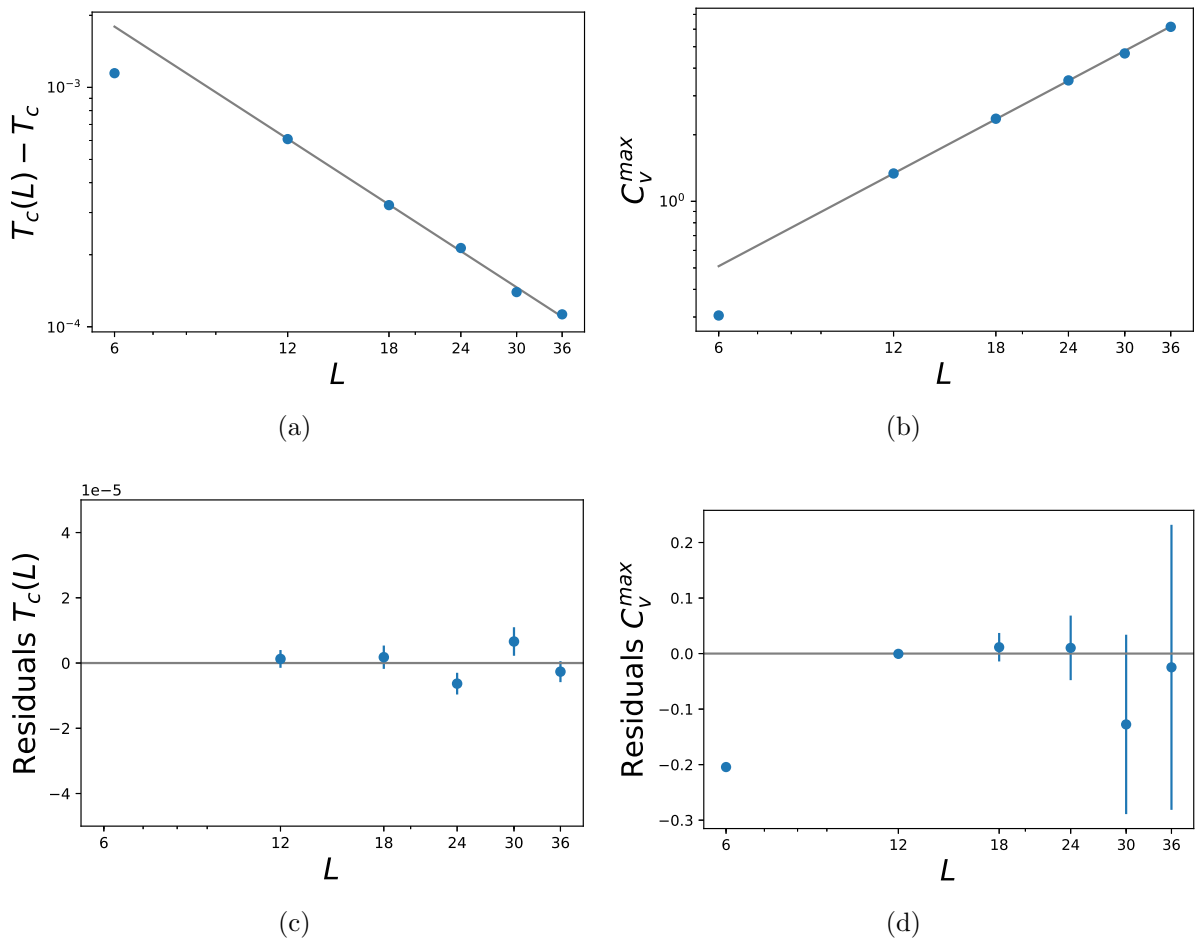


Figure 6.24.: Finite-size scaling for (a) the critical temperature and (b) the maximum of the specific heat. The $L = 6$ values are left out of the fit. (c,d) are the residuals of the fit.

The resulting critical exponents are:

$$T_c = (8.458 \pm 0.012) \times 10^{-3}$$
$$\nu = 0.646 \pm 0.026$$
$$\frac{\alpha}{\nu} = 1.393 \pm 0.016$$

and α can then be calculated to be:

$$\alpha = 0.90 \pm 0.04\,.$$

Note that the errors of the computed constants were obtained from the covariance (inverse hessian) of the fit parameters and that the $L = 6$ data point was not used in the fit. Since finite-size scaling is just an approximation at this relatively small lattice sizes, this

will bias the result. There might also be additional statistical errors not accounted for. This is why the same fit is repeated leaving different data points out.

L	$T_c \times 10^3$	ν	$\frac{\alpha}{\nu}$
None	8.458 ± 0.012	0.646 ± 0.026	1.393 ± 0.016
12	8.481 ± 0.036	0.730 ± 0.120	1.367 ± 0.043
18	8.472 ± 0.018	0.680 ± 0.410	1.388 ± 0.018
24	8.451 ± 0.011	0.623 ± 0.026	1.390 ± 0.020
30	8.453 ± 0.012	0.640 ± 0.025	1.399 ± 0.017
36	8.473 ± 0.022	0.672 ± 0.042	1.394 ± 0.017

Table 6.1.: Critical exponents leaving one of the lattice sizes out.

$\frac{\alpha}{\nu}$ and T_c do not change much between the different fits. In contrast, ν increases by 16% from its lowest to its highest value. The weighted mean and standard deviation of Tab. 6.1 is computed:

$$T_c = (8.458 \pm 0.015) \times 10^{-3} \tag{6.11}$$

$$\nu = 0.646 \pm 0.035 \tag{6.12}$$

$$\frac{\alpha}{\nu} = 1.393 \pm 0.020 \tag{6.13}$$

$$\alpha = 0.90 \pm 0.05\,. \tag{6.14}$$

This only raises the errors slightly and does not change the critical exponents. T_c could be computed to an accuracy of 0.2%. Even though $\frac{\alpha}{\nu}$ was computed with an accuracy of 1.4%, α has an inaccuracy of 5.5%. This is a result of the large error of 5.5% of ν. Unknown statistical errors might increase these figures.

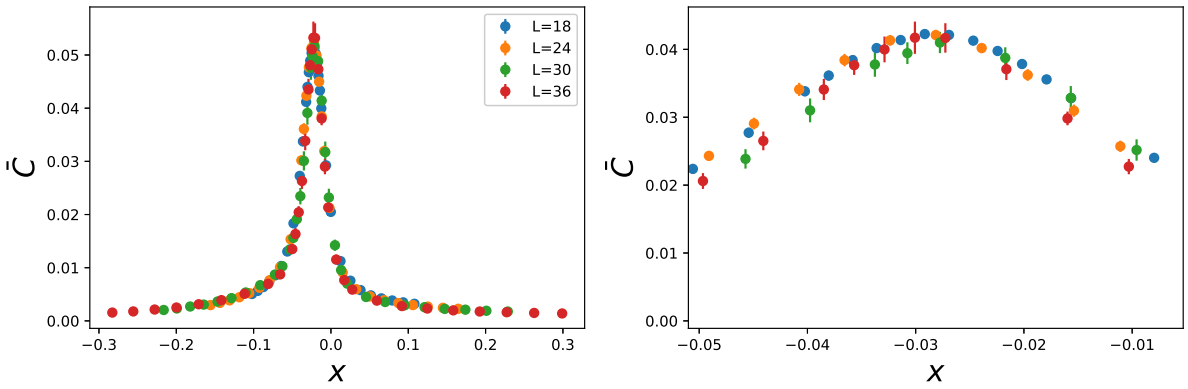
In Ref. [3], the authors conjectured that the phase transition could belong to the universality class of the two-dimensional 3-state Potts model. However, they were not able to compute the critical exponents. This conjecture is disproved thanks to the use of the LCRBM.

The results are double-checked with the help of another finite-size method. The specific heat C_v at finite lattice sizes can be described by:

$$C_v(T, L) = L^{\frac{\alpha}{\nu}} \bar{C}(L^{\frac{1}{\nu}}(T - T_c)). \tag{6.15}$$

With the critical exponents previously computed, $\bar{C}$ can be obtained for the different lattice

sizes (see Fig. 6.25). If the critical exponents were calculated accurately then $\bar{C}$ should be lattice size independent.



(a) (b)

Figure 6.25.: $\bar{C}(x)$ for $L = \{18, 24, 30, 36\}$ with $x = L^{\frac{1}{\nu}}(T - T_c)$.

While all the $\bar{C}$s are similar, small differences arise since this is just an approximation.

In App. A.5, $\bar{C}$ will be used to approximate the specific heat without performing a Monte Carlo simulation.

6.10. Summary

In this chapter, beta interpolation was introduced, which made it possible to sample at temperatures at which no RBM was trained on without decreasing the acceptance rate. Retraining the RBM at larger lattice sizes was considered, but the results for the retraining were inconclusive. Accordingly, this procedure is not recommended.

We confirmed that the acceptance rate decreases exponentially with increasing lattice size as $\log(A) \propto -L^2$ for the PRBM method. The LMRBM method was introduced to avoid small acceptance rates. For $L = 18$ and $L = 24$, the PRBM method outperformed the LMRBM method. This is because at these lattice sizes the PRBM method can produce states that are less correlated to each other. The LMRBM method continues to perform poorly even with its higher acceptance rates at $L = 36$. Due to the low acceptance rate of the PRBM method, artifacts arise. Mixing the two methods provides good results.

Metropolis was compared to the RBM for $L = 18, 24$. Metropolis not only needs to sample additional temperatures that are not relevant for this analysis, but it also needs 10 times more samples than the RBM to obtain stable results. Even with the increased sample size, the error is four times smaller for the RBM at $L = 18$. At $L = 24$, several Metropolis simulations were performed, but only one matched the specific heat of the RBM. Accordingly, one can say that running Metropolis at system sizes bigger than $L = 18$ is hard. As an alternative, simulations carried out at $L = 30$ and $L = 36$ were compared with results obtained from finite-size scaling. Satisfactory results are obtained for both. Computing the specific heat for lattice sizes larger then $L = 36$ would be hard since the simulation already takes 6 days at $L = 36$.

Finite-size scaling was performed with the data obtained from the simulation performed with the LCRBM. Thanks to that, the critical exponents could be calculated successfully. The most considerable error is 6% for α and ν. It is important to note that systematic errors could increase the uncertainty of all values. In Ref. [3], the authors conjectured that the phase transition could belong to the universality class of the two-dimensional 3-state Potts model. However, they were not able to compute the critical exponents. This conjecture is disproved in this chapter thanks to the use of the LCRBM.

CHAPTER 7

Conclusion

Machine learning methods are gaining traction in the condensed matter physics community, from calculating ground states of quantum mechanical systems to learning how to classify phases of matter. This research aimed to increase the speed of classical Monte Carlo (MC) simulations of condensed matter physics systems using a Convolutional Restricted Boltzmann Machine (CRBM). Making an MC simulation faster does not only save time, but it makes it possible to perform simulations on lattice sizes unobtainable otherwise.

This research builds on Ref. [1, 2], who both used fully connected RBMs for MC simulations. This method, however, has two substantial limitations: First, it is slow to train, and second, a Metropolis simulation needs to be performed beforehand, which is a problem if Metropolis is too slow. In this work, we solve these problems using a CRBM. Contrary to using fully connected RBMs, CRBM training can be done at small lattice sizes, where Metropolis still performs well. The same CRBM can then be applied at larger lattices. Another advantage is that fewer parameters need to be learned, which makes training faster.

As a proof of concept, the CRBM was tested on the two-dimensional Ising model. Thermodynamic constants could be reproduced and the CRBM achieved up to 80 times smaller autocorrelation time than Metropolis at $L = 100$, even though it was only trained at $L = 3$. This itself is not a great achievement since cluster algorithms that would outperform the CRBM exist for the Ising model. However, it shows that a CRBM can sample on lattice sizes that it was not trained at.

After successfully testing the CRBM on the Ising model, the CRBM was applied to the Kitaev honeycomb model since no cluster algorithms exist for this model. To be able to apply the CRBM, a new training procedure and a new method to correct errors, the PRBM method, were introduced. These newly introduced methods perform well and not only can they reproduce thermodynamics, but they can also perform around 60 times faster at $L = 16$. Even though Metropolis struggled at $L = 24$, a lattice size of $L = 30$ could be simulated with the CRBM. Larger simulations are not of interest, as the Kitaev model has no phase transition. This means that the specific heat does not change much when the lattice size is increased.

An extension of the Kitaev model, which has a phase transition, was introduced in Ref. [3]. However, the authors were not able to perform simulations of more than lattice size $L = 18$, which is also the upper limit we were able to achieve with Metropolis. Thanks to the CRBM, lattice sizes up to $L = 36$ could be simulated. This made it possible to gain new insight into the flux $\frac{3}{4}$ phase. Critical exponents could be estimated with the help of finite-size scaling. In the same paper, Ref. [3], the authors conjectured that the phase transition could belong to the universality class of the two-dimensional 3-state Potts model. However, they were not able to compute the critical exponents. This conjecture is disproved in the present work thanks to the use of the LCRBM.

We conclude that CRBMs can increase the speed of Monte Carlo simulations and, by doing so, makes it possible to gain insights into models where interesting physics happens at large lattice sizes.

This work finds two limitations: The first limitation is that a correcting step, which scales with complexity $\mathcal{O}(L^6)$, needs to be performed while running the simulations. Considering that it already took 6 days to perform simulations at $L = 36$ with an 8 core CPU, larger lattices would be hard to compute with our hardware setup.

The second limitation is that the acceptance rate of the PRBM method decreases as $\log(A) \propto -L^2$. Since the smallest acceptance rate at $L = 36$ is 0.5%, this is a problem. The LMRBM method is introduced to achieve larger acceptance rates, but it comes with the caveat of a much slower Markov chain. The two methods are combined into a new method, which we called the LMRBM/PRBM Mix method, with some success. This is because this newly introduced LMRBM/PRBM Mix method combines the high acceptance rates of LMRBM and the fast Markov chain of the PRBM. However, the problem of low

acceptance rates of the PRBM persists. More research is needed in order to address these two limitations.

Several avenues remain to be explored in future work. In our application on the Kitaev model, we sampled the gauge configurations u_{ij}^z. In future research, it would be of interest to instead sample directly Wilson loop configurations. This might decrease the complexity of the learned convolutional kernels. This was not done in the present study since each Wilson loop configuration has two possible free energies, but since this effect disappears at larger lattice sizes, it should be considered.

We showed that RBMs can be applied to temperatures they were not trained at through beta interpolation. For further work, it would be of interest to find a way to train a CRBM that would fit all temperatures. This could be done by parameterizing the convolutional kernels so that they are temperature dependent $W_i = f_i(\beta)$. This could vastly improve training time.

In the present work, only the critical exponents of the flux $\frac{3}{4}$ phase of the extended Kitaev model were computed. In further research, it would be of interest to compute the critical exponents of the other phases found in the extended Kitaev model. With this information, two things could be determined: First, whether all phases belong to the same universality class, and second, which universality class they belong to.

Other research avenues could also be explored. The CRBM could, for instance, be employed to model ground states of quantum mechanical models as in Ref. [5]. The matrix W learned in that paper seems to be both translation invariant and to not have long-range correlations. This would make the use of the CRBM of interest for this application. With this said, we hope that applications carried out in the present work are beneficial for opening further avenues in the machine learning and physics nexus.

Bibliography

[1] Giacomo Torlai and Roger G Melko. "Learning thermodynamics with Boltzmann machines". In: *Physical Review B* 94.16 (2016), p. 165134.

[2] Li Huang and Lei Wang. "Accelerated Monte Carlo simulations with restricted Boltzmann machines". In: *Physical Review B* 95.3 (2017), p. 035105.

[3] Shang-Shun Zhang et al. "Vison Crystals in an Extended Kitaev Model on the Honeycomb Lattice". In: *Phys. Rev. Lett.* 123 (5 July 2019), p. 057201. DOI: 10.1103/PhysRevLett.123.057201. URL: https://link.aps.org/doi/10.1103/PhysRevLett.123.057201.

[4] Juan Carrasquilla. "Machine Learning for Quantum Matter". In: *arXiv:2003.11040* (2020).

[5] Giuseppe Carleo and Matthias Troyer. "Solving the quantum many-body problem with artificial neural networks". In: *Science* 355.6325 (2017), pp. 602–606.

[6] Giacomo Torlai et al. "Neural-network quantum state tomography". In: *Nature Physics* 14.5 (2018), pp. 447–450.

[7] Dan Sehayek et al. "Learnability scaling of quantum states: Restricted Boltzmann machines". In: *Phys. Rev. B* 100 (19 Nov. 2019), p. 195125. DOI: 10.1103/PhysRevB.100.195125. URL: https://link.aps.org/doi/10.1103/PhysRevB.100.195125.

[8] Juan Carrasquilla and Roger G Melko. "Machine learning phases of matter". In: *Nature Physics* 13.5 (2017), pp. 431–434.

[9] Yi Zhang, Roger G. Melko, and Eun-Ah Kim. "Machine learning $\mathbb{Z}_2$ quantum spin liquids with quasiparticle statistics". In: *Phys. Rev. B* 96 (24 Dec. 2017), p. 245119. DOI: 10.1103/PhysRevB.96.245119. URL: https://link.aps.org/doi/10.1103/PhysRevB.96.245119.

[10] Jonas Greitemann, Ke Liu, and Lode Pollet. "Probing hidden spin order with interpretable machine learning". In: *Phys. Rev. B* 99 (6 Feb. 2019), p. 060404. DOI: 10.1103/PhysRevB.99.060404. URL: https://link.aps.org/doi/10.1103/PhysRevB.99.060404.

[11] Joaquin F Rodriguez-Nieva and Mathias S Scheurer. "Identifying topological order through unsupervised machine learning". In: *Nature Physics* 15.8 (2019), pp. 790–795. URL: https://www.nature.com/articles/s41567-019-0512-x.

[12] Stavros Efthymiou, Matthew JS Beach, and Roger G Melko. "Super-resolving the Ising model with convolutional neural networks". In: *Physical Review B* 99.7 (2019), p. 075113.

[13] Junwei Liu et al. "Self-learning monte carlo method". In: *Physical Review B* 95.4 (2017), p. 041101.

[14] Xiao Yan Xu et al. "Self-learning quantum Monte Carlo method in interacting fermion systems". In: *Phys. Rev. B* 96 (4 July 2017), p. 041119. DOI: 10.1103/PhysRevB.96.041119. URL: https://link.aps.org/doi/10.1103/PhysRevB.96.041119.

[15] Huitao Shen et al. "Self-Learning Monte Carlo Method in Fermion Systems". In: *Bulletin of the American Physical Society* 62 (2017).

[16] Yuki Nagai et al. "Self-learning Monte Carlo method: Continuous-time algorithm". In: *Phys. Rev. B* 96 (16 Oct. 2017), p. 161102. DOI: 10.1103/PhysRevB.96.161102. URL: https://link.aps.org/doi/10.1103/PhysRevB.96.161102.

[17] Chuang Chen et al. "Symmetry-enforced self-learning Monte Carlo method applied to the Holstein model". In: *Phys. Rev. B* 98 (4 July 2018), p. 041102. DOI: 10.1103/PhysRevB.98.041102. URL: https://link.aps.org/doi/10.1103/PhysRevB.98.041102.

[18] Huitao Shen, Junwei Liu, and Liang Fu. "Self-learning Monte Carlo with deep neural networks". In: *Physical Review B* 97.20 (2018), p. 205140.

[19] Shaozhi Li et al. "Accelerating lattice quantum Monte Carlo simulations using artificial neural networks: Application to the Holstein model". In: *Physical Review B* 100.2 (2019), p. 020302.

[20] Piers Coleman. *Introduction to Many-Body Physics*. Cambridge University Press, 2015. DOI: 10.1017/CBO9781139020916.

[21] M Newman and G Barkema. *Monte carlo methods in statistical physics chapter 8*. Oxford University Press: New York, USA, 1999.

[22] Honglak Lee et al. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. Montreal, Quebec, Canada: Association for Computing Machinery, 2009, pp. 609–616. ISBN: 9781605585161. DOI: 10.1145/1553374.1553453. URL: https://doi.org/10.1145/1553374.1553453.

[23] Robert H. Swendsen and Jian-Sheng Wang. "Replica Monte Carlo Simulation of Spin-Glasses". In: *Phys. Rev. Lett.* 57 (21 Nov. 1986), pp. 2607–2609. DOI: 10.1103/PhysRevLett.57.2607. URL: https://link.aps.org/doi/10.1103/PhysRevLett.57.2607.

[24] Koji Hukushima and Koji Nemoto. "Exchange Monte Carlo Method and Application to Spin Glass Simulations". In: *Journal of the Physical Society of Japan* 65.6 (1996), pp. 1604–1608. DOI: 10.1143/JPSJ.65.1604. eprint: https://doi.org/10.1143/JPSJ.65.1604. URL: https://doi.org/10.1143/JPSJ.65.1604.

[25] *Ising model demo - Google Colab*. https://colab.research.google.com/github/danielalcalde/MCMC_CRBM/blob/master/Ising_CRBM.ipynb. 2019.

[26] *Ising model demo - Github*. https://github.com/danielalcalde/MCMC_CRBM/blob/master/Ising_CRBM.ipynb. 2019.

[27] Rami Al-Rfou et al. "Theano: A Python framework for fast computation of mathematical expressions". In: *arXiv:arXiv:1605.02688* (2016).

[28] Alexei Kitaev. "Anyons in an exactly solved model and beyond". In: *Annals of Physics* 321.1 (2006), pp. 2–111.

[29] Joji Nasu, Masafumi Udagawa, and Yukitoshi Motome. "Vaporization of Kitaev spin liquids". In: *Physical review letters* 113.19 (2014), p. 197205.

[30] Jiannis K Pachos. *Introduction to topological quantum computation*. Cambridge University Press, 2012.

[31] Elliott H Lieb. "Flux phase of the half-filled band". In: *Condensed Matter Physics and Exactly Soluble Models*. Springer, 2004, pp. 79–82.

[32] Joji Nasu, Masafumi Udagawa, and Yukitoshi Motome. "Thermal fractionalization of quantum spins in a Kitaev model: Temperature-linear specific heat and coherent transport of Majorana fermions". In: *Phys. Rev. B* 92 (11 Sept. 2015), p. 115122. DOI: 10.1103/PhysRevB.92.115122. URL: https://link.aps.org/doi/10.1103/PhysRevB.92.115122.

[33] Elmar Bittner, Andreas Nußbaumer, and Wolfhard Janke. "Make Life Simple: Unleash the Full Power of the Parallel Tempering Algorithm". In: *Phys. Rev. Lett.* 101 (13 Sept. 2008), p. 130603. DOI: 10.1103/PhysRevLett.101.130603. URL: https://link.aps.org/doi/10.1103/PhysRevLett.101.130603.

[34] A Sokal. "Monte Carlo methods in statistical mechanics: foundations and new algorithms". In: *Functional integration*. Springer, 1997, pp. 131–192.

[35] George Fishman. *Discrete Event Simulation*. Springer, 2001.

Appendices

APPENDIX A

Monte Carlo

A.1. Detailed Balance

In this section it will be proven that the Gibbs sampling of the RBM and the Local Metropolis RBM Update (LMRBM) both satisfy the detailed balance condition:

$$P(v|v')P(v') = P(v'|v)P(v).$$

A.1.1. RBM Update

The conditional probability for an RBM is $P(v'|v) = \sum_h P(v'|h)P(h|v)$ and $P(h|v)P(v) = P(v|h)P(h)$. Using these two equations:

$$\begin{aligned} P(v'|v)P(v) &= \sum_h P(v'|h)P(h|v)P(v) \\ &= \sum_h \frac{P(v',h)P(v,h)}{P(h)} \\ &= \sum_h \frac{P(h|v')P(v')P(v,h)}{P(h)} \\ &= \sum_h P(v|h)P(h|v')P(v') \\ &= P(v|v')P(v'). \end{aligned}$$

A.1.2. Local Metropolis RBM Update

The procedure to do a Local Metropolis RBM update $g_L(v'|v)$ has two steps: First, a normal RBM Gibbs steps is done $P(v''|v) = \sum_h P(v''|h)P(h|v)$ and then a subset U of the elements of x is updated, whereas the subset NU is not updated. Note that $U \cup NU$ are all elements of x. This operation has the conditional probability:

$$g_U(v'|v'', v) = \prod_{i \in NU} \delta(v'_i - v_i) \prod_{i \in U} \delta(v'_i - v''_i)$$

, which is just a set of delta functions. Note that the RBM update can be rewritten as:

$$P(v''|v) = \sum_h \prod_i P(v''_i|h)P(h|v).$$

So the local update is:

$$\begin{aligned}
g_L(v'|v) &= \sum_{v''} P(v''|v) g_U(v'|v'', v) \\
&= \sum_{v''} \prod_{i \in NU} \delta(v'_i - v_i) \prod_{i \in U} \delta(v'_i - v''_i) \sum_h P(v'|h) \sum_h \prod_i P(v''_i|h)P(h|v) \\
&= \sum_h \sum_{v''} \prod_{i \in NU} \delta(v'_i - v_i) \prod_{i \in U} \delta(v'_i - v''_i) \prod_i P(v''_i|h)P(h|v) \\
&= \sum_h \prod_{i \in U} P(v'_i|h)P(h|v) \prod_{i \in NU} \delta(v'_i - v_i).
\end{aligned}$$

Now let's prove that this conditional distribution fulfills the detailed balance condition:

$$\begin{aligned}
g_L(v'|v)P(v) &= \sum_h \prod_{i\in U} P(v'_i|h) \prod_{i\in NU} \delta(v'_i - v_i) P(h|v)P(v) \\
&= \sum_h \prod_{i\in U} P(v'_i|h) \prod_{i\in NU} \delta(v'_i - v_i) P(v|h)P(h) \\
&= \sum_h \prod_{i\in U} P(v'_i|h) \prod_{i\in NU} \delta(v'_i - v_i) \prod_i P(v_i|h)P(h) \\
&= \sum_h \prod_{i\in U} P(v'_i|h) \prod_{i\in NU} \delta(v'_i - v_i) \prod_{i\in U} P(v_i|h) \prod_{i\in NU} P(v'_i|h)P(h) \\
&= \sum_h \prod_{i} P(v'_i|h) \prod_{i\in NU} \delta(v'_i - v_i) \prod_{i\in U} P(v_i|h)P(h) \\
&= \sum_h P(v'|h)P(h) \prod_{i\in NU} \delta(v'_i - v_i) \prod_{i\in U} P(v_i|h) \\
&= \sum_h P(h|v')P(v') \prod_{i\in NU} \delta(v'_i - v_i) \prod_{i\in U} P(v_i|h) \\
&= g_L(v|v')P(v').
\end{aligned}$$

A.2. Batch Means

Monte Carlo simulations are statistical processes, and as such their expectation values are not exact. A comprehensive review for error estimation in Monte Carlo methods can be found in Ref. [34] and Ref. [35].

Batch means is a method to estimate the error of Markov chain Monte Carlo (MCMC)s. The idea is to split the values into k batches X_j, then compute the expectation values of each batch:

$$\bar{f}_j = \sum_{x\in X_j} f(x). \tag{A.1}$$

After that, the standard deviation of the $\bar{f}_j$ is computed and divided by $\sqrt{k}$. The final error is:

$$\sigma = \frac{\mathrm{std}(\bar{f}_j)}{\sqrt{k}}. \tag{A.2}$$

For our purposes, $k = 10$ is chosen.

A.3. Autocorrelation Time

When performing Monte Carlo simulations the error of the observable decreases as more samples are drawn from the probability distribution. If all samples are independent the error σ of an expectation value $\langle f \rangle$ is:

$$\sigma = \sqrt{\frac{1}{N}\mathrm{Var}(f)} \tag{A.3}$$

where N is the number of drawn samples. The error σ decreases with $\sqrt{\frac{1}{N}}$. When samples are dependent on each other, as is the case in MCMC simulations, the error is:

$$\sigma = \sqrt{\frac{\tau}{N}\mathrm{Var}(f)} \tag{A.4}$$

where τ is the autocorrelation time. This means that the effective amount of samples is $\frac{N}{\tau}$. The autocorrelation $\rho(t)$ is a measure of how correlated a sample is to another sample after t MC steps. First, the mean is subtracted from the data (Eq. A.5). Second, the correlation between samples that are t steps apart is computed (Eq. A.6), and lastly, it is normalized (Eq. A.7).

$$\bar{f}_n = f_n - \langle f \rangle \tag{A.5}$$

$$c(t) = \frac{1}{N-t}\sum_{n=1}^{N-t} \bar{f}_n \bar{f}_{n+t} \tag{A.6}$$

$$\rho(t) = \frac{c(t)}{c(0)} \tag{A.7}$$

Note that $c(t)$ can be computed by using a Fourier transform and that if the samples are independent of each other, $\rho(t)$ should be 1 for $t = 0$ and 0 for $t > 0$. With infinite samples the autocorrelation decreases exponentially:

$$\rho(t) = e^{-\frac{2|t|}{\tau}}.$$

In the finite case, the larger t is, the noisier $\rho(t)$ becomes. This is because fewer samples are used to estimate bigger t. One way to compute τ is to construct a sum:

$$\tau = \sum_{t=-\infty}^{\infty} \rho(t). \tag{A.8}$$

Since only a finite amount of samples are available, τ will be estimated as:

$$\tau(M) = 1 + 2\sum_{t=1}^{M} \rho(t). \tag{A.9}$$

Note that the larger M is, the noisier the estimator becomes. A compromise between using the information in $\rho(t)$ (big M) and reducing the noise (small M) needs to be found. Sokal [34] proposes an automatic window procedure. The smallest M that fulfills $M > C\tau(M)$ is chosen. Sokal proposes $C = 5$.

Algorithm 4 Integrated autocorrelation time

samples f_n
$\langle f \rangle = \frac{1}{N}\sum_n f_n$
Fast Fourier Transform $\mathcal{F}$
$\bar{f}_n \leftarrow (f - \langle f \rangle)$
$g_n \leftarrow \mathcal{F}\bar{f}_n$
$c_t \leftarrow \mathcal{F}^{-1}(g_n g_n^*)$
$\rho_t \leftarrow \frac{c_t}{c_0}$
$\tau_m \leftarrow 2\sum_t^m \rho_t - 1$
$M \leftarrow$ smallest m where $m < \tau_m C$
$\tau \leftarrow \tau_M$

A.4. Computing $\mathbf{T_c(L)}$

In this section, the maximum values for the specific heat $C_v^{\max}$ and $T_c(L)$ will be calculated for several lattice sizes. For this the function:

$$f(T) = C_v^{\max} e^{a|T-T_c|^k} \tag{A.10}$$

is fitted to the vales close to $T_c(L)$. Note that the errors were calculated using the covariance of the fit parameters. The fits and the obtained values can be found in Fig. A.1 and Tab. A.1.

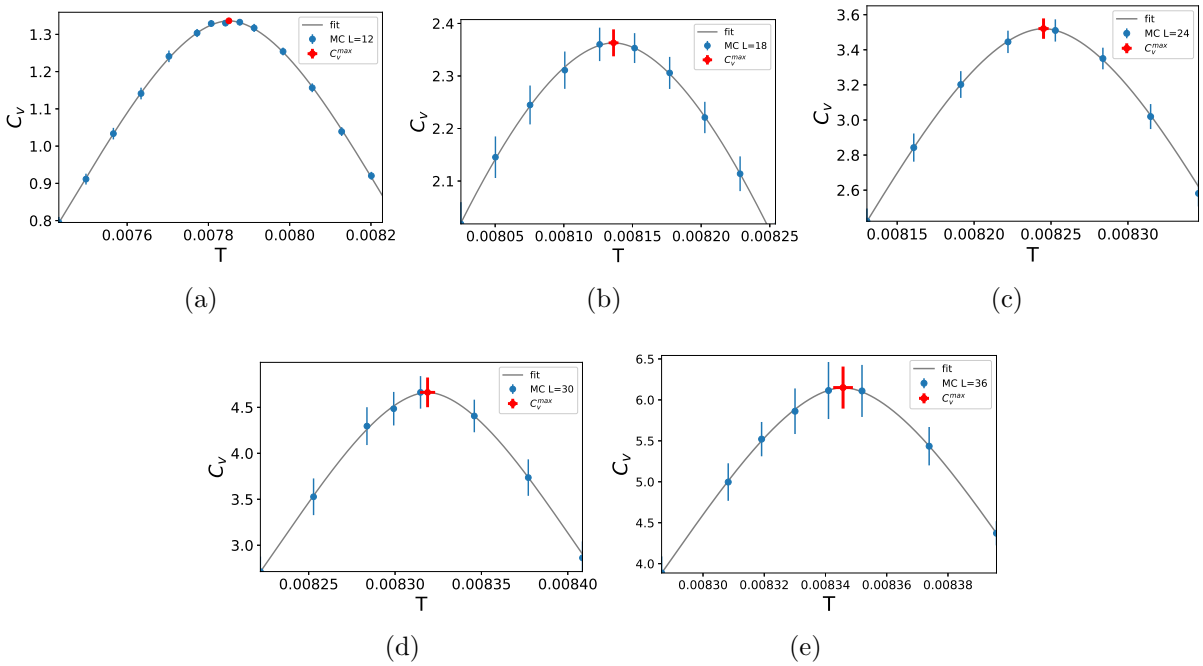


Figure A.1.: Fitting the specific heat at different lattice sizes to Eq. A.10. (a) $L = 12$, (b) $L = 18$, (c) $L = 24$, (d) $L = 30$ and (e) $L = 36$.

L	$T_c(L) \times 10^3$	$C_v^{\max}$	
12	7.850 ± 0.0027	1.336	± 0.005
18	8.136 ± 0.0036	2.363	± 0.026
24	8.245 ± 0.0033	3.52	± 0.06
30	8.319 ± 0.0044	4.66	± 0.16
36	8.346 ± 0.0032	6.15	± 0.25

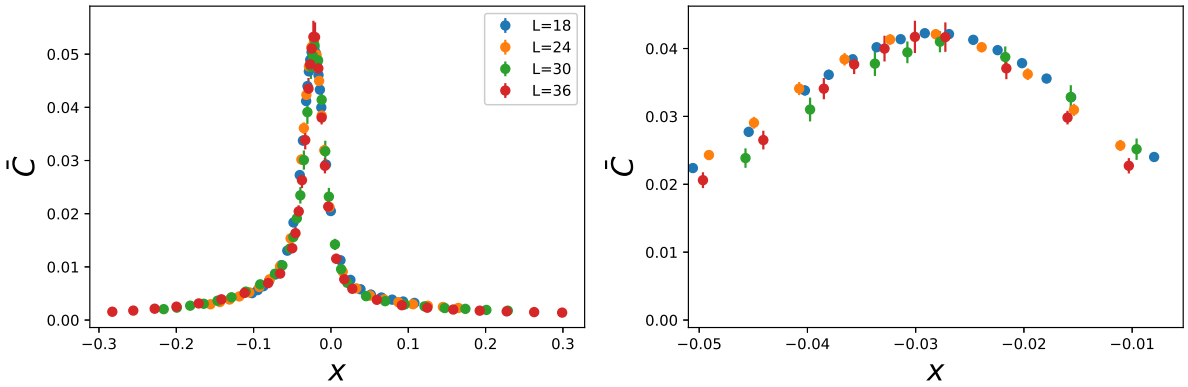
Table A.1.: Position of the maximum of the specific heat $T_c(L)$ and the maximal value $C_v^{\max}$.

A.5. Approximating the Specific Heat with Finite-Size Scaling

The specific heat C_v at finite lattice sizes can be described by:

$$C_v(T, L) = L^{\frac{\alpha}{\nu}} \bar{C}(L^{\frac{1}{\nu}}(T - T_c)). \tag{A.11}$$

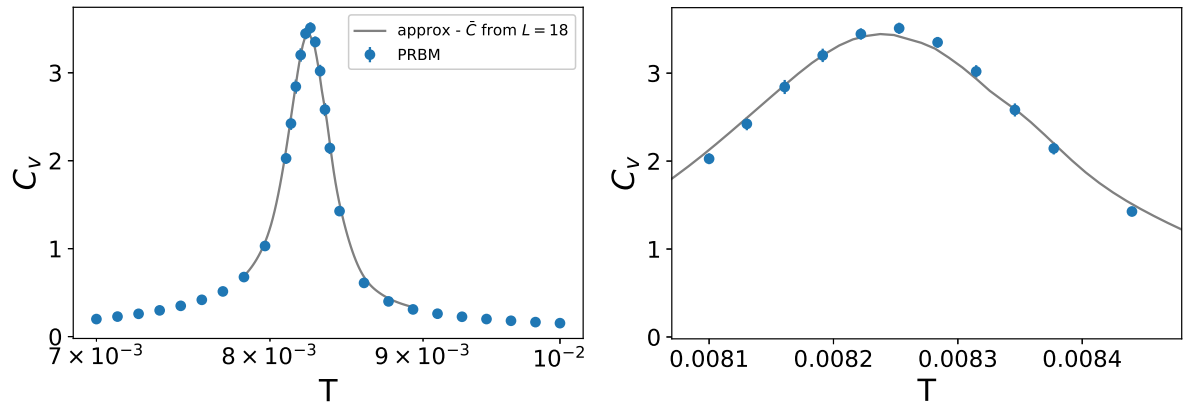
T_c, α, and ν are known from Sec. 6.9. With those values, $\bar{C}$ can be computed for different lattice sizes. In Fig. A.2 this is done for different lattice sizes.



(a) (b)

Figure A.2.: $\bar{C}(x)$ for $L = \{18, 24, 30, 36\}$ with $x = L^{\frac{1}{\nu}}(T - T_c)$.

While all the $\bar{C}$ are similar, small differences arise. This knowledge can be used to approximate the specific heat without sampling. This was used in Sec. 6.8 to approximate C_v. As an example, let's approximate $L = 24$ using $\bar{C}$ for $L = 18$.



(a) (b)

Figure A.3.: (a) The specific heat at $L = 24$ computed with (blue) PRBM Monte Carlo and (black) specific heat approximated through the $\bar{C}$ obtained from $L = 18$. (b) Zoom in on the peak of (a).

The two specific heats do not coincide exactly, but given that the approximation is just valid at large lattice sizes the results are satisfactory.

A.6. Training and Sampling Parameters

In this section, you will find the parameters used for both training and sampling in this thesis.

Ising Model The Ising model is trained on all 512 possible states with $L = 3$ with 2 kernels of size 2×2. Boundary conditions are chosen to be periodic. Note that no statistical correction was necessary since it learned the distribution exactly.

Kitaev Model The training for the Kitaev model was done at $L = 8$ with 4 kernels of size 6×6.
5×10^4 samples were produced with Metropolis. Only every fifth sample will be used for training. In total 10^4 samples are used for training. The buffer update size is set to $ub = 200$. The learning rate is set to $\lambda = 10^{-3}$.

For sampling, the PRBM method is used. For all lattice sizes, 10^3 Gibbs steps were performed between correction steps. Note that the faster method (App. B.2) to compute the free energy was not discovered at the time of writing the Kitaev model chapter.

Extended Kitaev Model An LCRBM is trained for the extended Kitaev model at $L = 12$ with 10 kernels of size 6×6, and a boundary size of 3. Boundary conditions are chosen to be periodic in z-direction and open in x-y-direction. The buffer update size is set to $ub = 200$. The learning rate is set to $\lambda = 10^{-3}$ and $\lambda = 10^{-4}$ when retraining at $L = 18$.

For sampling, the PRBM and LMRBM methods are used. Instead of setting a fixed amount of Gibbs steps per correction step like for the Kitaev model, the amount of Gibbs steps is chosen so that the Gibbs steps take the same amount of time as a correction step. Note that the update sizes are much smaller for the extended Kitaev model since the LCRBM implementation is 10 times slower than the CRBM and the 6 times faster method is used to compute the free energy. For $L = 18$ we used 37 Gibbs steps, for $L = 24$: 40 steps, $L = 30$: 62 steps, and $L = 36$: 150 steps. Experiments were run on a computer with an Intel(R) Xeon(R) CPU E5-2620 v4 running at 2.10GHz.

APPENDIX B

Free Energy of the Kitaev Model

Because of the antisymmetry of the matrix, A can be Shur decomposed:

$$A = Q \begin{pmatrix} 0 & \epsilon_1 & & & \\ -\epsilon_1 & 0 & & & \\ & & \ddots & & \\ & & & 0 & \epsilon_m \\ & & & -\epsilon_m & 0 \end{pmatrix} Q^T \tag{B.1}$$

where Q is a unitary matrix. With the new basis:

$$(b'_1, b''_1, ..., b'_m, b''_m) = (c^b_1, c^w_1, ..., c^b_m, c^w_m)Q \tag{B.2}$$

the Hamiltonian is obtained:

$$\mathcal{H}(u^z) = \frac{i}{2} \sum_k \epsilon_\lambda b'_\lambda b''_\lambda \tag{B.3}$$

which can then be transformed back from Majorana representation to fermion representation with $a^\dagger = \frac{1}{2}(b' - ib'')$, which yields the final Hamiltonian:

$$\mathcal{H}(u^z) = \sum_\lambda \epsilon_\lambda (a^\dagger_\lambda a_\lambda - \frac{1}{2}). \tag{B.4}$$

The partition function now reads:

$$Z = \mathrm{Tr}_{u^z} \mathrm{Tr}_{n_\lambda} e^{-\beta \mathcal{H}(u^z)} \tag{B.5}$$

$$\mathrm{Tr}_{n_\lambda} e^{-\beta \sum_\lambda \epsilon_\lambda (n_\lambda - \frac{1}{2})} = \mathrm{Tr}_{n_\lambda} \prod_\lambda e^{-\beta \epsilon_\lambda (n_\lambda - \frac{1}{2})} \tag{B.6}$$

$$= \prod_\lambda \mathrm{Tr}_{n_\lambda} e^{-\beta \epsilon_\lambda (n_\lambda - \frac{1}{2})} \tag{B.7}$$

$$= \prod_\lambda \left(e^{-\frac{\beta \epsilon_\lambda}{2}} + e^{\frac{\beta \epsilon_\lambda}{2}} \right) \tag{B.8}$$

$$= \prod_\lambda \left(2 \cosh \frac{\beta \epsilon_\lambda}{2} \right) \tag{B.9}$$

and so the free energy is:

$$F(u^z) = -T \ln\left(\mathrm{Tr}_{a_\lambda} e^{-\beta \mathcal{H}(u^z)} \right) \tag{B.10}$$

$$= -T \sum_\lambda \ln(2 \cosh(\beta \epsilon_\lambda / 2)). \tag{B.11}$$

B.1. Filling the Matrix

To construct A, Fig. B.1 is useful. Each black and white lattice point is mapped to a $2 \times L \times L$ lattice that is connected through the gauge fields u^α_{ij}. Since the lattice is two-dimensional and there are two particles in a unit cell, i is expanded to $i = (x_1, x_2, (b \text{ or } w))$. The gauge fields $u^\alpha_{x_1 x_2, i, z_1, z_2, j} = -u^\alpha_{z_1 z_2, j, x_1, x_2, i}$ are mapped to:

$$u^x_{x_1, x_2, b, x_1 \quad, x_2 \quad, w} = u^x_{x_1, x_2} \tag{B.12}$$

$$u^y_{x_1, x_2, b, x_1 \quad, x_2+1, w} = u^y_{x_1, x_2} \tag{B.13}$$

$$u^z_{x_1, x_2, b, x_1+1, x_2 \quad, w} = u^z_{x_1, x_2}. \tag{B.14}$$

Note that the new u are defined as a connection from black to white.

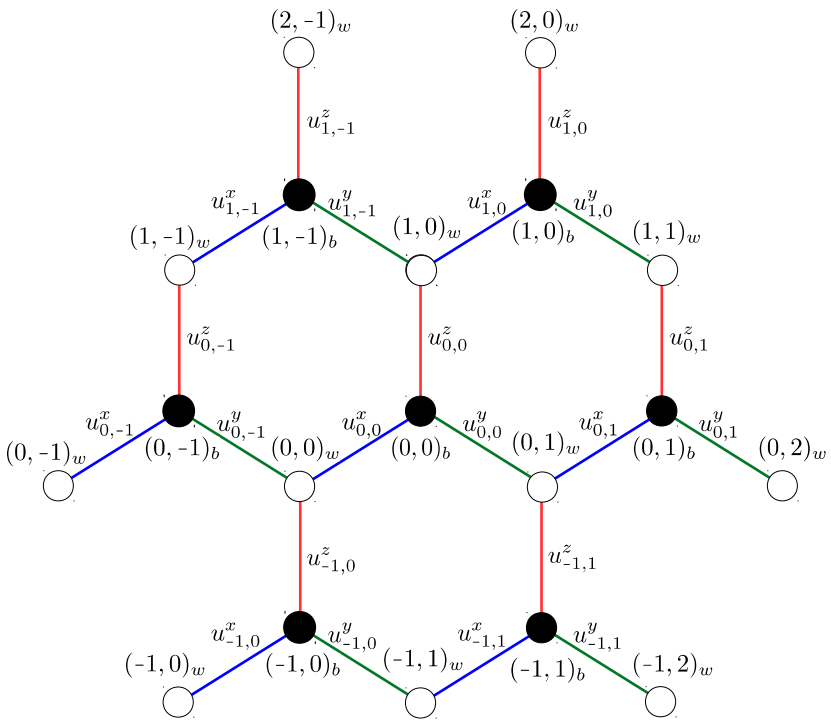


Figure B.1.: Honeycomb lattice. The lattice points and the gauge fields are numbered. Note that blue represents a x, green a y, and red a z connection.

Now the matrix A is filled with the gauge fields. First, a tensor A with dimensions $2 \times L \times L \times 2 \times L \times L$ is created. 0 represents black and 1 represents white. A is filled:

$$A[i,j,0,i,j,1] = u^x_{ij} J_x \tag{B.15}$$
$$A[i,j,1,i,j,0] = -u^x_{ij} J_x \tag{B.16}$$

$$A[i,j,0,i,j+1,1] = u^y_{ij} J_y \tag{B.17}$$
$$A[i,j+1,1,i,j,0] = -u^y_{ij} J_y \tag{B.18}$$

$$A[i,j,0,i+1,j,1] = u^z_{ij} J_z \tag{B.19}$$
$$A[i+1,j,1,i,j,0] = -u^z_{ij} J_z \tag{B.20}$$

or in table form:

	$c^w_{i+\Delta i,j+\Delta j}$		$u^\alpha_{i+\Delta i,j+\Delta j}$		
Strength	Δi	Δj	α	Δk	Δl
J_x	0	0	x	0	0
J_y	0	+1	y	0	0
J_z	+1	0	z	0	0

Table B.1.: Interactions of type $c^b_{ij}c^w_{i+\Delta i,j+\Delta j}u^\alpha_{i+\Delta k,j+\Delta l}$.

After the filling, A is reshaped into a $2L^2 \times 2L^2$ matrix and the eigenvalues are computed.

B.2. A Faster Method

A method that speeds up the calculation of the free energy 6 times, was discovered after the writing of Chap. 5 and was first used for Chap. 6.

Since the matrix A has no connections between operators of the same color, so no black-black connections and no white-white connections are present. This means that:

$$A = \begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \tag{B.21}$$

and so the eigenvalue equation is:

$$\begin{pmatrix} 0 & -B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix} \tag{B.22}$$

where λ is an eigenvalue and (x, y) is its eigenvector. This yields the eigenvalue equation:

$$BB^T x = (i\lambda)^2 x = \lambda_B x. \tag{B.23}$$

After computing the eigenvalues λ_B of BB^T, the eigenvalues of the system are straightforward to compute:

$$\begin{aligned} \lambda &= \pm i\epsilon \\ \epsilon &= \sqrt{\lambda_B}. \end{aligned}$$

In practice, this gives a 6 times speedup over a direct diagonalization. This method can

be both applied to the Kitaev model and the extension presented in this thesis but can not be used for interactions with three spin terms $\sigma^\alpha\sigma^\beta\sigma^\alpha$. These are used to model a magnetic field, and when expanded into gauge configuration they lead to white-white and black-black terms in the Hamiltonian, which makes the use of the faster method not possible.

B.3. Interactions of the Extended Model

For the extended model, there is a three gauge field interactions of the form:

$$c^b_{ij}c^w_{i+\Delta i,j+\Delta j}u^\alpha_{i,j}u^\beta_{i+\Delta k,j+\Delta l}u^\gamma_{i+\Delta k,j+\Delta l}.$$

All contributions to the Hamiltonian are listed in Tab. B.2.

	$c^w_{i+\Delta i,j+\Delta j}$		$u^\alpha_{i+\Delta i,j+\Delta j}$			$u^\beta_{i+\Delta k,j+\Delta l}$			$u^\gamma_{i+\Delta k,j+\Delta l}$		
Strength	Δi	Δj	α	Δi	Δj	β	Δk	Δl	γ	Δk	Δl
K_3	-1	$+1$	y	0	0	z	-1	$+1$	x	-1	$+1$
			x	0	0	z	-1	0	y	-1	0
	$+1$	$+1$	y	0	0	x	0	$+1$	z	0	$+1$
			z	0	0	x	$+1$	0	y	$+1$	0
	$+1$	-1	x	0	0	y	0	-1	z	0	-1
			z	0	0	y	$+1$	-1	x	$+1$	-1
K_3'	-1	$+2$	y	0	0	z	-1	$+1$	y	-1	$+1$
	0	$+2$	z	0	0	y	0	$+1$	z	0	$+1$
	$+2$	0	z	0	0	y	$+1$	0	z	$+1$	0
	$+2$	-1	z	0	0	y	$+1$	-1	z	$+1$	-1
	0	-1	x	0	0	y	0	-1	x	0	-1
	-1	0	x	0	0	z	-1	0	x	-1	0

Table B.2.: Interactions described in Sec. 6.1. The operator c^b_{ij} interacts with $c^w_{i+\Delta i,j+\Delta j}$ with the sign of the product of the three gauge fields $u^\alpha_{i,j}u^\beta_{i+\Delta k,j+\Delta l}u^\gamma_{i+\Delta k,j+\Delta l}$.

APPENDIX C

Other Methods

C.1. Phase Diagram

The objective of this section is to present the algorithm to compute the phase diagram from Sec. 6.2. The gauge field u that minimizes $F(u; K_3, K_3')$ needs to be found. In essence, a lot of small Monte Carlo simulations using Metropolis are done with different parameters K_3 and K_3'. A random state is initialized, then k Metropolis steps at the inverse temperature β_0 are performed. Then β is increased to $\beta * c$, and k more Metropolis steps are performed. This is repeated until the free energy of the state did not reduce in l steps. In practice, $\beta_0 = 10$ and $c = 1.00693$ are chosen such that there are 10^3 steps between β_0 and $\beta = 10^3$. Before saving the new state, the free energy of ground states obtained for other parameters is computed for the current parameters, and if the computed state is indeed the smallest for the parameters K_3 and K_3', it is saved.

This needs to be done for an interval $[\bar{K_3}, K_3] \times [\bar{K_3'}, K_3']$. Doing this to high resolution is expensive. First, a coarse calculation is done. The interval is split into $2^3 \times 2^3$ pixels. After that, the ground states are obtained through the previous method. Next, the lattice resolution is increased by a factor of two. Instead of performing the optimization procedure for all new pixels, only the ones that are at a boundary between two phases are computed again. This is repeated until the desired accuracy is obtained. Note that once it is believed that all possible ground states have been obtained, it is possible to stop running Monte Carlo simulations and instead just test, which of the saved ground states has the lowest free energy for the parameters.

List of Figures