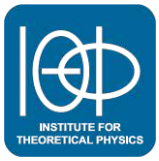


DIPLOMARBEIT

Using Modern Equivariant Machine Learning Architectures as Effective Hamiltonians for Monte Carlo Simulations of Quantum Spin Systems

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

066 461
Masterstudium Technische Physik

eingereicht von

Felix Schlemmer
Matrikelnummer 11906276

ausgeführt am Institut für Theoretische Physik
der Fakultät für Physik der Technischen Universität Wien

Betreuung
Betreuer: Privatdoz. Dipl.-Ing. Dr.techn. Andreas Ipp

Wien, 31.08.2025

(Unterschrift Verfasser)

(Unterschrift Betreuer)

Kurzfassung

Monte-Carlo-Methoden spielen eine fundamentale Rolle in der numerischen Simulation physikalischer Systeme, insbesondere in der statistischen Mechanik, der Quantenmechanik und der Festkörperphysik. In dieser Arbeit werden verschiedene Monte-Carlo-Techniken und deren Anwendung auf unterschiedliche Hamiltonians untersucht, mit besonderem Fokus auf das Ising-Modell, das Ising-ähnliche Plaquettenmodell und das Double-Exchange-Modell. Ein zentrales Ziel ist der Vergleich verschiedener Update-Strategien, darunter lokale Metropolis-Updates, globale Wolff-Cluster-Updates und Self Learning Monte Carlo (SLMC) Updates. Nach der Demonstration der Wirksamkeit von SLMC anhand einer Proof-of-Concept-Studie des Ising-ähnlichen Plaquettenmodells wird gezeigt, dass SLMC in Kombination mit netzwerkbasierten effektiven Hamiltonians die Simulationseffizienz komplexer Quantenspinsysteme signifikant verbessern kann. Zukünftige Arbeiten können sich darauf konzentrieren, den vorgestellten Formalismus auf komplexere Systeme wie die $SU(3)$-Symmetriegruppe der QCD zu erweitern.

Abstract

Monte Carlo methods play a fundamental role in the numerical simulation of physical systems, particularly in statistical mechanics, quantum mechanics, and condensed matter physics. This study explores various Monte Carlo techniques and their application to different Hamiltonians, with a focus on the Ising model, the Ising-like plaquette model, and the Double Exchange model. A key objective is to compare different update strategies, including local Metropolis updates, global Wolff cluster updates, and Self Learning Monte Carlo (SLMC) updates. After demonstrating the effectiveness of SLMC using a proof-of-concept study of the Ising-like plaquette model, it is shown that SLMC in combination with network-based effective Hamiltonians can improve the simulation efficiency of complex quantum spin systems significantly. Future work might focus on extending the presented formalism to more complex systems such as the $SU(3)$ symmetry group of QCD.

Contents

1. Introduction

Monte Carlo methods have long been a cornerstone in the numerical simulation of physical systems, providing powerful tools for investigating statistical mechanics, quantum mechanics, and condensed matter physics [1]. This work explores various Monte Carlo techniques and their application to different Hamiltonians, with a particular focus on the Ising model [2], the Ising-like plaquette model [3], and the Double Exchange model [4–6]. The study aims to compare different update methods, including local and global techniques, and to assess the effectiveness of Self Learning Monte Carlo (SLMC) [3] updates in improving computational efficiency for complex quantum systems.

The motivation behind this research lies in the necessity of efficient simulation techniques for complex systems. Many-body problems, particularly those involving interactions between classical and quantum mechanical degrees of freedom, pose significant computational challenges. The Double Exchange model, which describes the coupling of itinerant electrons to localized spins, exemplifies such a system. Traditional Monte Carlo methods, while effective, often suffer from critical slowing down, limiting their applicability near phase transitions. By incorporating cluster update methods, such as the Wolff algorithm, and exploring the potential of SLMC, this work seeks to mitigate these limitations and enhance the performance of Monte Carlo simulations.

This thesis is structured as follows: Chapter 2 provides a detailed description of the Hamiltonians analyzed in this work, starting with the classical Ising model, followed by the Ising-like plaquette model, and concluding with the more complex Double Exchange model. Chapter 3 introduces the fundamental principles of Monte Carlo simulations and explores several update strategies, including local Metropolis updates, Wolff cluster updates, and the Self Learning Monte Carlo (SLMC) method. Chapters 4 and 5 discuss the construction and training of effec-

tive Hamiltonians using neural network architectures, such as transformer-based and fully connected models, with an emphasis on numerical stability and expressivity. Chapter 6 presents a comprehensive comparison of simulation performance across models and update schemes, evaluating acceptance rates, autocorrelation lengths, and accuracy of observable estimation. Finally, Chapter 7 concludes with a summary of key findings and an outlook on future directions.

2. Exact Hamiltonians

This chapter provides an overview of the Hamiltonians investigated in this study. Beginning with the Ising model, a well-established framework in statistical mechanics, its formulation and key properties will be discussed. After the Ising model, the Ising-like plaquette model is introduced, which is used to validate the Self Learning Monte Carlo implementation. The chapter then introduces the Double Exchange model, which describes the interaction between itinerant electrons and classical spins, requiring a more intricate treatment due to the coupling of classical and quantum degrees of freedom. The mathematical structure of these models is presented, along with relevant approximations and observables that will be used in subsequent Monte Carlo simulations.

2.1. Ising Model

In preparation for simulating the Double Exchange model, a simpler model, namely the Ising model, is considered. The Ising model is a simplification of the more generalized classical Heisenberg model [7]

$$_{Heisenberg}\mathcal{H}(\{\boldsymbol{S}\}) = -\frac{1}{2}\sum_{i,j} J_{ij}\,\boldsymbol{S}_i \cdot \boldsymbol{S}_j. \tag{2.1}$$

The Heisenberg model describes classical spins $\boldsymbol{S}_i$ of unit length, which are placed on a lattice. The interaction strength between the spins on the lattice sites i and j is given by the coupling constant J_{ij}. The coupling constant fulfills $J_{ij} = J_{ji}$ and to prevent self interactions of the spins, $J_{ii} = 0$ holds. A common simplification of the Heisenberg model is to pin the spins to one direction, most commonly the

z-direction. The resulting Hamiltonian

$$_{Ising}\mathcal{H}(\{\boldsymbol{S}\}) = -\frac{1}{2}\sum_{i,j} J_{ij}\, S_i^z \cdot S_j^z \tag{2.2}$$

describes the Ising model without an external field.
The model was first introduced by Ernst Ising and Wilhelm Lenz in 1920 [2], and in 1925 an exact solution for the 1d-Ising model was found by Ising and subsequently published in his thesis [8]. The one dimensional model does not show a phase transition. The two dimensional model was solved only approximately 20 years later in 1942 by Lars Onsager [9, 10]. In two or more dimensions, the Ising model shows a phase transition between an unordered and ordered phase. The ordered phase shows ferro- or antiferromagnetic order, depending on the sign of the coupling parameter J.
The expectation values for observables in the Ising model can be calculated using

$$\langle \mathcal{O} \rangle_\beta = \frac{1}{\mathcal{Z}_\beta} \int \mathrm{d}\boldsymbol{S}\, e^{-\beta \mathcal{H}(\{\boldsymbol{S}\})}\, \mathcal{O}(\{\boldsymbol{S}\}), \tag{2.3}$$

$$\mathcal{Z}_\beta = \int \mathrm{d}\boldsymbol{S}\, e^{-\beta \mathcal{H}(\{\boldsymbol{S}\})}, \tag{2.4}$$

where $\mathcal{Z}$ is the partition function. Common observables are the magnetization and energy [7].

2.1.1. Mean Field Approximation with External Field

A common method for calculating the partition function $\mathcal{Z}_\beta$ of the Ising model is the mean field approximation. Starting from the Ising model with an external field h (2.5)

$$_{Ising}\mathcal{H}(\{\boldsymbol{S}\}) = -\frac{J}{2}\sum_{\langle i,j \rangle} S_i^z S_j^z - h\sum_i S_i^z, \tag{2.5}$$

the nearest neighbour interaction $S_i^z S_j^z$ is expanded around the expectation value $\langle S_i^z \rangle$ and higher order terms of the fluctuation $(\langle S_i^z \rangle - S_i)$ are dropped, as given

by

$$\begin{aligned} S_i^z S_j^z &= (\langle S_i^z \rangle + S_i^z - \langle S_i^z \rangle) \cdot (\langle S_j^z \rangle + S_j^z - \langle S_j^z \rangle) \\ &= \langle S_i^z \rangle \langle S_j^z \rangle + \langle S_i^z \rangle (S_j^z - \langle S_j^z \rangle) + \langle S_j^z \rangle (S_i^z - \langle S_i^z \rangle) + \\ &\quad + \cancel{(S_i^z - \langle S_i^z \rangle)(S_j^z - \langle S_j^z \rangle)}. \end{aligned} \tag{2.6}$$

The resulting mean field (MFT) Hamiltonian is

$$_{MFT}\mathcal{H}(\{\boldsymbol{S}\}) = \frac{1}{2} m^2 N q J - \sum_i S_i^z \left(h + qJm\right), \tag{2.7}$$

where $m = \langle S_i^z \rangle$, N is the total number of spins in the system, q is the number of nearest neighbours, and $S_i^z \in \{-1, 1\}$. The partition function $\mathcal{Z}_\beta$ evaluates to

$$\mathcal{Z}_\beta = \sum_{S_i^z \in \{-1,1\}} \left[e^{-\beta \, _{MFT}\mathcal{H}(\{\boldsymbol{S}\})} \right] = \left(e^{-\frac{1}{2}\beta m^2 q J} 2 \cosh\left[\beta(h + qJm)\right] \right)^N. \tag{2.8}$$

With $m = \frac{1}{\beta N} \frac{\partial}{\partial h} \log \mathcal{Z}_\beta$, an implicit equation for the magnetization of the system can be calculated [7]

$$m = \tanh\left[\beta(h + qJm)\right]. \tag{2.9}$$

A plot of the magnetization is given in figure 2.1.

2.2. Ising-Like Plaquette Model

An extension of the Ising model considered in this thesis is an Ising-like plaquette model. The Hamiltonian is first introduced in [3] and used to demonstrate the effectiveness of the Self Learning Monte Carlo method (SLMC). The Hamiltonian – given in equation (2.10) – is a direct extension of the Ising model and introduces a non-linear fourth order spin term. As for the Ising model, the degrees of freedom considered are one dimensional spins $S_i^z \in \{+1, -1\}$. A similar Hamiltonian could be constructed using $O(3)$ spins, however, such a Hamiltonian would need to consider permutations of the plaquette spins. If the plaquette coupling t is taken to

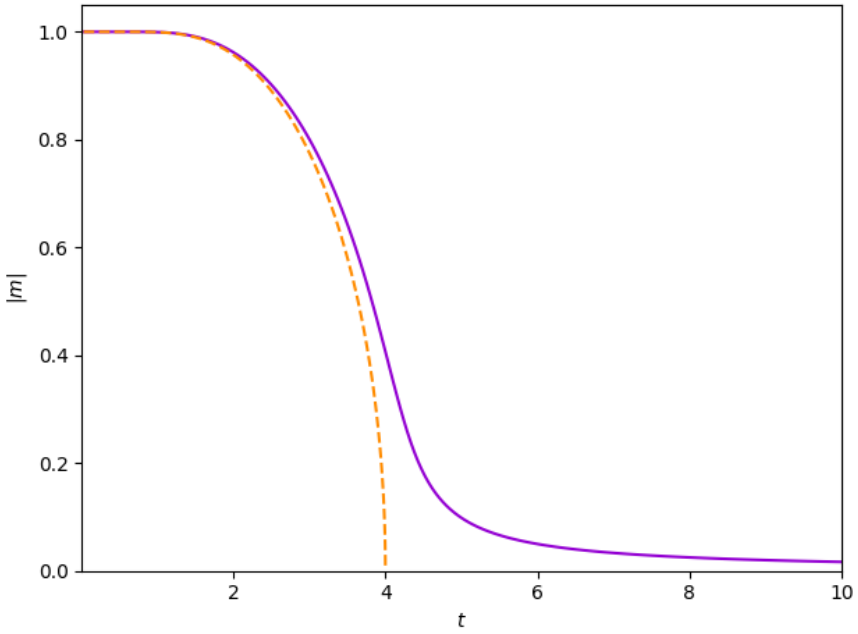


Figure 2.1.: *The dashed line corresponds to an external field of $h = 0$, the solid line to an external field $h = 0.1$. In both cases, the number of nearest neighbours is $q = 4$ and the coupling constant is $J = 4$.*

be $t = 0$, the model reduces to the ordinary Ising model, given by

$$_{PQ}\mathcal{H}(\{\boldsymbol{S}\}) = -J \sum_{\langle i,j \rangle} S_i^z S_j^z - t \sum_{ijkl \in \square} S_i^z S_j^z S_k^z S_l^z, \tag{2.10}$$

with a constant coupling J and only first order nearest neighbour interactions.
In the context of this thesis, the model is used to validate the implementation of the Self Learning Monte Carlo method (for details see section 3.4) and familiarize oneself with training effective models.
The Ising-like plaquette model is suitable for these tasks, as the non-linear nature of the model cannot be reproduced exactly by linear nearest neighbour interactions of higher orders, and thus if the observables of the Hamiltonian can be reproduced correctly using SLMC, the robustness of the reweighting is demonstrated.
The formalism developed for the Ising model (section 2.1) can be applied to the Ising-like plaquette model without major modification.

2.3. Double Exchange Model

The Double Exchange model – also known as spin fermion model – was first proposed by Clarence Zehner [5] and describes the interaction between itinerant electrons and stationary, classical spins. It therefore describes an interaction between classical and quantum-mechanical degrees of freedom [6].

2.3.1. Hamiltonian and Ladder Operators

The Hamiltonian consists of three parts, the hopping term, the spin-fermion interaction and the occupation number of electrons per spin site i and is given by

$$_{DE}\mathcal{H}(\{\boldsymbol{S}\}) = \underbrace{-t\sum_{\langle i,j\rangle,\sigma}\left(\hat{c}^{\dagger}_{i,\sigma}\hat{c}_{j,\sigma}+\hat{c}^{\dagger}_{j,\sigma}\hat{c}_{i,\sigma}\right)}_{\text{hopping term}} + \underbrace{\frac{J}{2}\sum_{i,\sigma,\sigma'}\hat{c}^{\dagger}_{i,\sigma}(\boldsymbol{S}_i\cdot\boldsymbol{\sigma})_{\sigma\sigma'}\hat{c}_{i,\sigma'}}_{\text{spin-fermion interaction}} - \underbrace{\mu\sum_{i,\sigma}\hat{c}^{\dagger}_{i,\sigma}\hat{c}_{i,\sigma}}_{\mu\hat{\mathcal{N}}}\,. \tag{2.11}$$

The hopping term allows a fermion to jump from the lattice site i to a neighbouring lattice site j – indicated by $\langle i,\, j\rangle$ – and vice versa. t is the hopping constant, which sets the energy scale of the system. The interaction strength between the classical spins $\boldsymbol{S}_i$ and the fermions is set by the coupling constant J and a fermion with spin σ is created (annihilated) on lattice site i through the application of $\hat{c}^{\dagger}_{i,\sigma}$ ($\hat{c}_{i,\sigma}$) on the vacuum wave function $|\emptyset\rangle$. μ is the chemical potential of the system, most commonly chosen to be 0. The classical spins are placed on a 2D $N \times N$ square lattice and are normalized to $|\boldsymbol{S}| = 1$. $\boldsymbol{\sigma}$ are the Pauli matrices $\{\sigma_x,\, \sigma_y,\, \sigma_z\}$ [4, 11].

Creation and Annihilation Operators

The challenge when simulating the Double Exchange model are the creation and annihilation operators of the fermions. The creation/annihilation operators for

one spin up or spin down particle are given by

$$\hat{c}^\dagger = \tfrac{1}{2}\left(\sigma_x + i\sigma_y\right), \tag{2.12a}$$

$$\hat{c} = \tfrac{1}{2}\left(\sigma_x - i\sigma_y\right). \tag{2.12b}$$

If the Fock-states (2.17) are considered, the tensor product of the two Hilbert spaces must be taken and thus the ladder operators for a 2 particle system have the form [12]

$$\hat{c}_\uparrow^{(\dagger)} = \hat{c}^{(\dagger)} \otimes \mathbb{I}, \tag{2.13a}$$

$$\hat{c}_\downarrow^{(\dagger)} = \mathbb{I} \otimes \hat{c}^{(\dagger)}. \tag{2.13b}$$

Matrix Representation

To calculate the matrix representation of the ladder operators, the identity matrix and the Pauli matrices

$$\mathbb{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.14}$$

must be inserted into (2.12) and (2.13). The ladder operators are then given by

$$\hat{c}^\dagger = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}, \qquad \hat{c} = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}. \tag{2.15}$$

If the state of the single particle can be $|\mathbb{1}\rangle = (1,\, 0)^T$ and $|\emptyset\rangle = (0,\, 1)^T$, the ladder operators act in the following way:

$$\hat{c}^\dagger |\emptyset\rangle = |\mathbb{1}\rangle, \qquad \hat{c}^\dagger |\mathbb{1}\rangle = 0, \qquad \hat{c} |\emptyset\rangle = 0, \qquad \hat{c} |\mathbb{1}\rangle = |\emptyset\rangle. \tag{2.16}$$

Furthermore, the anticommutation relation $\{\hat{c},\, \hat{c}^\dagger\} = \mathbb{I}$ holds and $(\hat{c}^{(\dagger)})^2 = 0$.

For a two particle system, represented by the states

$$|\uparrow\downarrow\rangle = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \qquad |\uparrow\rangle = \begin{pmatrix}0\\1\\0\\0\end{pmatrix}, \qquad |\downarrow\rangle = \begin{pmatrix}0\\0\\1\\0\end{pmatrix}, \qquad |\emptyset\rangle = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}, \tag{2.17}$$

the matrix representation of the ladder operators is

$$\hat{c}_\uparrow^\dagger = \begin{pmatrix}0&0&1&0\\0&0&0&1\\0&0&0&0\\0&0&0&0\end{pmatrix}, \qquad \hat{c}_\uparrow = \begin{pmatrix}0&0&0&0\\0&0&0&0\\1&0&0&0\\0&1&0&0\end{pmatrix}, \tag{2.18}$$

$$\hat{c}_\downarrow^\dagger = \begin{pmatrix}0&1&0&0\\0&0&0&0\\0&0&0&1\\0&0&0&0\end{pmatrix}, \qquad \hat{c}_\downarrow = \begin{pmatrix}0&0&0&0\\1&0&0&0\\0&0&0&0\\0&0&1&0\end{pmatrix}. \tag{2.19}$$

For the ladder operators, the relations

$$\begin{aligned}
&\hat{c}_\uparrow^\dagger|\uparrow\downarrow\rangle = 0, && \hat{c}_\uparrow^\dagger|\uparrow\rangle = 0, && \hat{c}_\uparrow^\dagger|\downarrow\rangle = |\uparrow\downarrow\rangle, && \hat{c}_\uparrow^\dagger|\emptyset\rangle = |\uparrow\rangle,\\
&\hat{c}_\uparrow|\uparrow\downarrow\rangle = |\downarrow\rangle, && \hat{c}_\uparrow|\uparrow\rangle = |\emptyset\rangle, && \hat{c}_\uparrow|\downarrow\rangle = 0, && \hat{c}_\uparrow|\emptyset\rangle = 0,\\
&\hat{c}_\downarrow^\dagger|\uparrow\downarrow\rangle = 0, && \hat{c}_\downarrow^\dagger|\uparrow\rangle = |\uparrow\downarrow\rangle, && \hat{c}_\downarrow^\dagger|\downarrow\rangle = 0, && \hat{c}_\downarrow^\dagger|\emptyset\rangle = |\downarrow\rangle,\\
&\hat{c}_\downarrow|\uparrow\downarrow\rangle = |\uparrow\rangle, && \hat{c}_\downarrow|\uparrow\rangle = 0, && \hat{c}_\downarrow|\downarrow\rangle = |\emptyset\rangle, && \hat{c}_\downarrow|\emptyset\rangle = 0
\end{aligned} \tag{2.20}$$

hold.

n-Particle Operators

For the n-particle case, the construction of the ladder operators works in a similar fashion. The ladder operator consists of tensor products of the unit matrix with the corresponding 2-particle (2p) ladder operator on the corresponding index. For example, if the operator $\hat{c}_{\uparrow,(3)}^\dagger$ should be constructed for a system with 4 lattice

sites, the operator reads

$$\hat{c}^{\dagger}_{\uparrow,(3)} = \mathbb{I} \otimes \mathbb{I} \otimes \hat{c}^{\dagger}_{\uparrow} \otimes \mathbb{I}, \tag{2.21}$$

where $\mathbb{I}$ indicates the 4×4 unit matrix. Thus, the total matrix dimension of the operator is 4^4. In general, the matrix dimension scales like $4^{\mathbb{L}}$, where $\mathbb{L}$ is the lattice size $N \times N$. This is a problem, since the dimension of the ladder operator determines the dimension of the Hamiltonian. As will be outlined in the next section, the eigenvalues – e.g. the energy values – of the Hamiltonian are needed to calculate observables. If the dimension of the Hamiltonian gets too large, the system can not be simulated within a reasonable timeframe [4, 6, 11].

2.3.2. 1-Particle Reduced Hamiltonian

To solve the problem of the dimensionality of the Hamiltonian, the fact that the fermions in (2.11) are non-interacting can be exploited. The non-interaction of the fermions implies that the full n-particle Hamiltonian can be replaced by the one-particle operator. In contrast to the dimension of the full Hamiltonian ($4^{\mathbb{L}}$), the dimension of the 1-particle (1p)Hamiltonian scales like $2 \cdot \mathbb{L}$. The drastic difference of these scaling laws is visualized in figure 2.2. To construct the 1p-Hamiltonian, the one particle entries must be projected out from the full Hamiltonian. This is done by calculating matrix elements of the form $\langle \Psi_{\mathcal{A}} |\, {}_{DE}\mathcal{H}(\{\boldsymbol{S}\})\, | \Psi_{\mathcal{B}} \rangle$, where $\Psi_{\mathcal{A}}$ and $\Psi_{\mathcal{B}}$ are one particle wave functions, which can be constructed using $\hat{c}^{\dagger}_{\sigma} |\emptyset\rangle$, with $|\emptyset\rangle$ being the vacuum state. The result of the calculation is given by

$$\begin{aligned} {}^{1p}_{DE}\mathcal{H}_{nm\rho\rho'}(\{\boldsymbol{S}\}) &= \langle \emptyset\, \hat{c}_{n\rho} |\, {}_{DE}\mathcal{H}(\{\boldsymbol{S}\})\, | \hat{c}^{\dagger}_{m\rho'}\, \emptyset \rangle \\ &= -2t\delta_{\rho\rho'} \sum_{\langle i,\, j \rangle} \delta_{nj}\delta_{im} + \frac{J}{2}\delta_{nm}(\boldsymbol{S}_m \cdot \boldsymbol{\sigma})_{\rho\rho'} - \mu\delta_{nm}\delta_{\rho\rho'}. \end{aligned} \tag{2.22}$$

This result is obtained if all expressions are simplified using the anticommutation relations for the ladder operators

$$\begin{aligned} \{\hat{c}_{\sigma},\, \hat{c}^{\dagger}_{\sigma'}\} &= \delta_{\sigma\sigma'}, \\ \{\hat{c}^{\dagger}_{\sigma},\, \hat{c}^{\dagger}_{\sigma'}\} &= \{\hat{c}_{\sigma},\, \hat{c}_{\sigma'}\} = 0. \end{aligned} \tag{2.23}$$

The detailed derivation of the 1-particle reduced Hamiltonian, including the application of anticommutation relations and matrix element evaluations, is provided in appendix A.

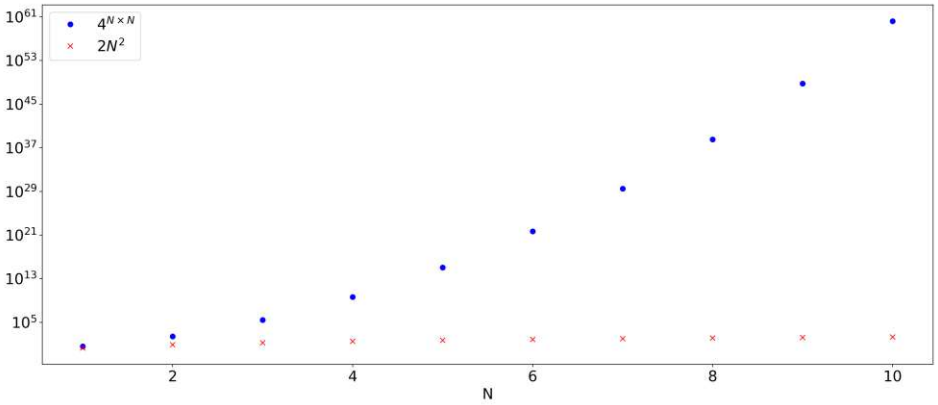


Figure 2.2.: *Given the lattice has $\mathbb{L} = N \times N$ sites, the dimension of the Hamiltonian scales differently for the full- and the 1p-Hamiltonian. The two scaling laws are compared in this figure. Especially noteworthy is the logarithmic scale on the y-axis.*

2.3.3. Observables

Similar to the Ising model, observables are calculated through the use of the partition function $\mathcal{Z}_\beta$. While the integration over the spin configurations was sufficient for the Ising model, the quantum-mechanical degrees of freedom in the Double Exchange model require an additional trace [6]. In general, this trace calculation is highly non-trivial, however, using the fact that the fermions are non interacting and treating them with the grand canonical ensemble, a simple expression can be obtained

$$\mathcal{Z}_\beta = \int \mathrm{d}\boldsymbol{S}\, \mathrm{Tr}\left[e^{-\beta\mathcal{H}(\{\boldsymbol{S}\})}\right] = \int \mathrm{d}\boldsymbol{S} \prod_\nu \left[1 + e^{-\beta\varepsilon_\nu(\{\boldsymbol{S}\})}\right], \tag{2.24}$$

where ε_ν are the eigenenergies of the one particle Hamiltonian. Observables can be calculated using

$$\langle\mathcal{O}\rangle_\beta = \frac{1}{\mathcal{Z}_\beta} \int \mathrm{d}\boldsymbol{S}\, \mathrm{Tr}\left[e^{-\beta\mathcal{H}(\{\boldsymbol{S}\})}\right] \mathcal{O}(\{\boldsymbol{S}\}). \tag{2.25}$$

Common observables are the magnetization

$$|\,\boldsymbol{M}\,| = \frac{1}{\mathcal{Z}_\beta}\int \mathrm{d}\boldsymbol{S}\,\mathrm{Tr}\left[e^{-\beta\mathcal{H}(\{\boldsymbol{S}\})}\right]\left|\frac{1}{\mathbb{L}}\sum_i \boldsymbol{S}_i\right|, \tag{2.26}$$

the staggered magnetization

$$|\,\boldsymbol{M}_s\,| = \frac{1}{\mathcal{Z}_\beta}\int \mathrm{d}\boldsymbol{S}\,\mathrm{Tr}\left[e^{-\beta\mathcal{H}(\{\boldsymbol{S}\})}\right]\left|\frac{1}{\mathbb{L}}\sum_i \boldsymbol{S}_i(-1)^{\sum_{j=1}^{d} x_j}\right|, \tag{2.27}$$

where x_j in the sum in the exponent indicates the site index in the j-th dimension, and the mean energy [4, 6]

$$\langle E\rangle = -\frac{\partial \ln\left(\mathcal{Z}_\beta\right)}{\partial\beta} = \frac{1}{\mathcal{Z}_\beta}\int \mathrm{d}\boldsymbol{S}\,\mathrm{Tr}\left[e^{-\beta\mathcal{H}(\{\boldsymbol{S}\})}\right]\sum_\nu \frac{\varepsilon_\nu}{1+e^{\beta\varepsilon_\nu}}. \tag{2.28}$$

How these integrals are calculated in the framework of Monte Carlo simulations will be described in detail in the following sections.

3. Monte Carlo Methods

Monte Carlo methods serve as essential tools for the numerical study of statistical and quantum-mechanical systems. This chapter introduces the fundamental principles of Monte Carlo sampling and discusses various update methods, including the local Metropolis algorithm, global Wolff cluster updates, and the Self Learning Monte Carlo algorithm. Additionally, the Hybrid Monte Carlo method is discussed, particularly in the context of constrained spin manifolds. Lastly, the No-U-Turn sampler is introduced, a state of the art extension of the Hybrid Monte Carlo method.

3.1. Introduction to Monte Carlo Methods

The Monte Carlo method was first used by Nicholas Metropolis et al. [1] to calculate observables of an ensemble. The main challenge when calculating observables is the calculation of the high dimensional integrals over the phase space, given by

$$\langle \mathcal{O} \rangle = \frac{1}{\mathcal{Z}_\beta} \int \mathrm{d}\boldsymbol{p}\mathrm{d}\boldsymbol{q}\, \mathcal{O}(\boldsymbol{p},\, \boldsymbol{q})\, e^{-\beta \mathcal{H}(\boldsymbol{p},\, \boldsymbol{q})}. \tag{3.1}$$

The trick used to calculate the given integral is to generate configurations $\{\boldsymbol{p},\, \boldsymbol{q}\}$ according to the probability distribution $e^{-\beta\mathcal{H}}$ [1]. The configurations following this distribution can be generated using a Markov chain obeying detailed balance and ergodicity. If a set $\{\mathfrak{x}_n\}_{n\geq 0}$ of stochastically distributed random configurations and its transition probabilities T fulfil the condition

$$T(\mathfrak{x}_{n+1}|\mathfrak{x}_n,\, \mathfrak{x}_{n-1},\, \ldots,\, \mathfrak{x}_0) = T(\mathfrak{x}_{n+1}|\mathfrak{x}_n) \quad \forall n \geq 0, \tag{3.2}$$

the set is called a Markov chain [13]. This means that each generated configuration $\mathfrak{x}_i \equiv \mathfrak{x}_i(\boldsymbol{p}, \boldsymbol{q})$ only depends on the previous configuration (no memory). If this condition is satisfied,

$$p_n(\mathfrak{x}) = p_{n-1}(\mathfrak{x}) + \int \mathrm{d}\mathfrak{x}'\, T(\mathfrak{x}|\mathfrak{x}')p_{n-1}(\mathfrak{x}') - \int \mathrm{d}\mathfrak{x}'\, T(\mathfrak{x}'|\mathfrak{x})p_{n-1}(\mathfrak{x}) \tag{3.3}$$

is fulfilled for the configuration probabilities $p_n(\mathfrak{x})$. In the limit for large n, $p_n(\mathfrak{x}) \approx p_{n-1}(\mathfrak{x}) = p_{eq}(\mathfrak{x})$ and thus a sufficient condition for equation (3.3) is

$$T(\mathfrak{x}|\mathfrak{x}')p_{eq}(\mathfrak{x}') = T(\mathfrak{x}'|\mathfrak{x})p_{eq}(\mathfrak{x}). \tag{3.4}$$

The condition (3.4) is called detailed balance. In Monte Carlo simulations, $p_{eq}(\mathfrak{x})$ is most commonly chosen to be $p_{eq}(\mathfrak{x}) = e^{-\beta\mathcal{H}[\mathfrak{x}(\boldsymbol{p},\boldsymbol{q})]}$.
Ergodicity is the condition that each configuration $\mathfrak{x}_i$ can theoretically be reached from each other configuration $\mathfrak{x}_j$. This condition highly depends on the specific update method used. If both detailed balance and ergodicity are fulfilled and the random configurations $\mathfrak{x}_i$ are generated according to $p_{eq}(\mathfrak{x})$, then the stochastic expectation value $\langle\mathcal{O}\rangle$ can be calculated using

$$\langle\mathcal{O}\rangle = \frac{1}{N}\sum_i \mathcal{O}_i, \tag{3.5}$$

where N is the total number of configurations and $\mathcal{O}_i$ is the value of the observable associated with the configuration $\mathfrak{x}_i$ [14].
How the configurations $\mathfrak{x}_i$ are generated in detail depends on the specific update method used. A few of these update methods will be discussed in the following.

3.2. Local Updates – Metropolis Algorithm

One of the most flexible update methods is the Metropolis algorithm. Here, only one degree of freedom is changed to step from configuration $\mathfrak{x}_i$ to $\mathfrak{x}_{i+1}$. To determine if the new configuration is accepted, the transition probability T is written as $T(\mathfrak{x}'|\mathfrak{x}) = g(\mathfrak{x}'|\mathfrak{x}) \cdot A(\mathfrak{x}'|\mathfrak{x})$, where $g(\mathfrak{x}'|\mathfrak{x})$ is the proposal distribution and $A(\mathfrak{x}'|\mathfrak{x})$ is

the acceptance probability. $g(\mathfrak{x}'|\mathfrak{x})$ is the probability to propose the configuration $\mathfrak{x}'$, given the current configuration is $\mathfrak{x}$. The new configuration $\mathfrak{x}'$ is then chosen with the acceptance probability $A(\mathfrak{x}'|\mathfrak{x})$. If the decomposition of T is inserted in the detailed balance condition,

$$\frac{A(\mathfrak{x}'|\mathfrak{x})}{A(\mathfrak{x}|\mathfrak{x}')} = \frac{p_{eq}(\mathfrak{x}')}{p_{eq}(\mathfrak{x})} \frac{g(\mathfrak{x}|\mathfrak{x}')}{g(\mathfrak{x}'|\mathfrak{x})} \tag{3.6}$$

is obtained. A common choice for the acceptance ratio, which fulfills equation (3.6) is

$$A(\mathfrak{x}'|\mathfrak{x}) = \min\left(1, \frac{p_{eq}(\mathfrak{x}')}{p_{eq}(\mathfrak{x})} \frac{g(\mathfrak{x}|\mathfrak{x}')}{g(\mathfrak{x}'|\mathfrak{x})}\right). \tag{3.7}$$

This method is known as the Metropolis algorithm. For local updates, the distribution $g(\mathfrak{x}'|\mathfrak{x})$ is constant, since the generation of the configuration $\mathfrak{x}'$ is purely random [13, 14]. For this case, the Metropolis algorithm simplifies to

$$A(\mathfrak{x}'|\mathfrak{x}) = \min\left(1, e^{-\beta\Delta E}\right), \tag{3.8}$$

where ΔE is the energy difference of the configurations $\mathfrak{x}$ and $\mathfrak{x}'$. The main drawback of this update method is that near the critical point, where the correlation length diverges, a lot of samples must be generated to reach a statistically significant result. This limitation can be mitigated with nonlocal update methods. Pseudocode for the Metropolis algorithm is given in appendix B.1.

3.3. Global Updates

To mitigate the problems of local update methods, global update methods can be developed. These update methods are designed for a specific Hamiltonian and are thus not that versatile in comparison to local updates. One prominent example will be discussed in the following.

3.3.1. Wolff Cluster Updates

The Wolff update is one example for a cluster update. In contrast to local updates, cluster updates update significantly more degrees of freedom compared to the

single one in local updates. The Wolff update method was first developed in 1989 to combat the critical slowing down in Ising-like models [15]. The idea behind the Wolff algorithm is to construct a cluster of spins and flip all of the involved spins. The update has an acceptance probability of 1, since detailed balance is considered when constructing the cluster.

Algorithm

Before discussing the Wolff algorithm, the notion of a spin flip must be generalized. In the Ising model, spins can usually only take the values $S^z = \pm 1$. There, a spin flip is trivial. In the case where the spins are elements of $SO(n)$, the spins can be reflected along the reflection vector $\boldsymbol{r}$ using

$$R(\boldsymbol{r})\boldsymbol{S}_i = \boldsymbol{S}_i - 2(\boldsymbol{S}_i \cdot \boldsymbol{r})\boldsymbol{r}. \tag{3.9}$$

The algorithm to construct a Wolff cluster will be described in the following.

1. choose a random reflection vector $\boldsymbol{r}$ and a random lattice site i
2. add the spin at site i to the cluster $\mathcal{C}$
3. connect a neighbouring site $\boldsymbol{S}_j$ to the cluster with probability

$$P(\boldsymbol{S}_i, \boldsymbol{S}_j) = 1 - \exp\left\{\min\left[0,\, 2\beta(\boldsymbol{r} \cdot R(\boldsymbol{r})\boldsymbol{S}_i)(\boldsymbol{r} \cdot \boldsymbol{S}_j)\right]\right\}$$

   and if the connection is performed, add $\boldsymbol{S}_j$ to the cluster $\mathcal{C}$
4. repeat for all unvisited neighbouring lattice sites
5. flip all spins contained in the cluster $\forall \boldsymbol{S}_i \in \mathcal{C}\colon\ \boldsymbol{S}_i \mapsto R(\boldsymbol{r})\boldsymbol{S}_i$

Ergodicity is fulfilled, since a cluster can be constituted by only a single spin, and the reflection vector is chosen at random. Detailed balance is also fulfilled, since the transition probabilities fulfil the relation [15]

$$\frac{T(\mathfrak{x}'|\mathfrak{x})}{T(\mathfrak{x}|\mathfrak{x}')} = e^{-\beta\Delta E}. \tag{3.10}$$

Pseudocode for the Wolff cluster algorithm is given in appendix B.2.

3.4. Self Learning Monte Carlo Method

A major limitation of global update methods is that they must be designed carefully for each set of problems. An effort to mitigate this limitation is done by the Self Learning Monte Carlo (SLMC) method. Here, two Hamiltonians are involved, the exact Hamiltonian $\mathcal{H}$ and an effective Hamiltonian $\mathcal{H}_{\text{eff}}$. The effective Hamiltonian is preferably chosen in such a way that there exists an efficient global update method. The effective Hamiltonian then 'learns' the exact Hamiltonian and Monte Carlo updates are performed using the effective Hamiltonian.

Algorithm

The algorithm consists of the following steps:

1. Learning Process
   - 1.1 Perform a trial simulation using local updates and the exact Hamiltonian $\mathcal{H}$ to generate training data.
   - 1.2 Using the data generated in the previous step, an effective Hamiltonian $\mathcal{H}_{\text{eff}}$ is learned.
2. Simulation
   - 2.1 Using $\mathcal{H}_{\text{eff}}$, propose new configurations.
   - 2.2 Determine, whether the proposed configuration should be kept using the original Hamiltonian $\mathcal{H}$.

The acceptance ratio of a proposed configuration is computed following these steps: Since the new configurations are proposed following the effective Hamiltonian $\mathcal{H}_{\text{eff}}$, the proposal distribution obeys

$$\frac{g(\mathfrak{x}'|\mathfrak{x})}{g(\mathfrak{x}|\mathfrak{x}')} = \frac{p_{eq,\,\text{eff}}(\mathfrak{x}')}{p_{eq,\,\text{eff}}(\mathfrak{x})}. \tag{3.11}$$

Inserting this into (3.7),

$$A(\mathfrak{x}'|\mathfrak{x}) = \min\left(1, \frac{p_{eq}(\mathfrak{x}')}{p_{eq}(\mathfrak{x})}\frac{p_{eq,\,\text{eff}}(\mathfrak{x})}{p_{eq,\,\text{eff}}(\mathfrak{x}')}\right) \tag{3.12}$$

is obtained. This is the acceptance ratio for an SLMC update [3, 4]. Pseudocode for the SLMC algorithm is given in appendix B.3.

3.5. Hybrid Monte Carlo Method

A global update method which is relatively system independent is the Hybrid – or Hamiltonian – Monte Carlo method. The main idea is to generate new configurations through integrating Hamilton's equations.
Suppose the target distribution for the Markov chain is $f(\mathfrak{x})$. By adding auxiliary momentum variables $\mathfrak{p}$, the Hamiltonian transforms to

$$\mathcal{H} \to \mathbb{H} = \mathcal{U}(\mathfrak{x}) + \mathcal{T}(\mathfrak{x}, \mathfrak{p}), \tag{3.13}$$

where $\mathcal{U}(\mathfrak{x}) = -\ln f(\mathfrak{x})$, which follows from $f(\mathfrak{x}) \to f(\mathfrak{x})e^{-\mathcal{T}} = e^{-\mathbb{H}} = p(\mathfrak{x}, \mathfrak{p})$. With this new Hamiltonian $\mathbb{H}$, new configurations $\mathfrak{x}'$ can be calculated using

$$\frac{\mathrm{d}\mathfrak{x}}{\mathrm{d}t} = \frac{\partial \mathbb{H}}{\partial \mathfrak{p}} = \frac{\partial \mathcal{T}}{\partial \mathfrak{p}}, \tag{3.14}$$

$$-\frac{\mathrm{d}\mathfrak{p}}{\mathrm{d}t} = \frac{\partial \mathbb{H}}{\partial \mathfrak{x}} \tag{3.15}$$

To calculate the acceptance ratio of a proposed configuration, the proposal probability $g(\mathfrak{x}', \mathfrak{p}'|\mathfrak{x}_0, \mathfrak{p}_0)$ must be considered. If a usual integrator is chosen that performs the transformation $(\mathfrak{x}_0, \mathfrak{p}_0) \to (\mathfrak{x}_T, \mathfrak{p}_T)$ after the integration time T, then the proposal probability is $g(\mathfrak{x}', \mathfrak{p}'|\mathfrak{x}_0, \mathfrak{p}_0) = \delta(\mathfrak{x}' - \mathfrak{x}_T)\delta(\mathfrak{p}' - \mathfrak{p}_T)$.
Inserting this into the acceptance ratio for the Metropolis algorithm (3.7), the acceptance ratio is always 0, since the proposal probability is not reversible, which can be seen from

$$\begin{aligned} A(\mathfrak{x}_T, \mathfrak{p}_T|\mathfrak{x}_0, \mathfrak{p}_0) &= \min\left(1, \frac{p(\mathfrak{x}_T, \mathfrak{p}_T)}{p(\mathfrak{x}_0, \mathfrak{p}_0)} \frac{g(\mathfrak{x}_0, \mathfrak{p}_0|\mathfrak{x}_T, \mathfrak{p}_T)}{g(\mathfrak{x}_T, \mathfrak{p}_T|\mathfrak{x}_0, \mathfrak{p}_0)}\right) \\ &= \min\left(1, \frac{p(\mathfrak{x}_T, \mathfrak{p}_T)}{p(\mathfrak{x}_0, \mathfrak{p}_0)} \frac{\delta(2\mathfrak{x}_T - \mathfrak{x}_0)\delta(2\mathfrak{p}_T - \mathfrak{p}_0)}{\delta(\mathfrak{x}_T - \mathfrak{x}_T)\delta(\mathfrak{p}_T - \mathfrak{p}_T)}\right) \\ &= \min\left(1, \frac{p(\mathfrak{x}_T, \mathfrak{p}_T)}{p(\mathfrak{x}_0, \mathfrak{p}_0)} \frac{0}{1}\right) = 0. \end{aligned} \tag{3.16}$$

The expressions $\delta(2\mathfrak{x}_T - \mathfrak{x}_0)$ and $\delta(2\mathfrak{p}_T - \mathfrak{p}_0)$ come from the fact that after two steps, the new configuration $(\mathfrak{x}', \mathfrak{p}')$ is $(\mathfrak{x}', \mathfrak{p}') = (2\mathfrak{x}_T, 2\mathfrak{p}_T)$ and the target configuration is $(\mathfrak{x}_0, \mathfrak{p}_0)$.

If the transition is modified to be reversible, which corresponds to the sign flip $(\mathfrak{x},\, \mathfrak{p}) \to (\mathfrak{x},\, -\mathfrak{p})$ in the integrator, and modifying $g(\mathfrak{x}',\, \mathfrak{p}'|\mathfrak{x}_0,\, \mathfrak{p}_0)$ to read

$$g(\mathfrak{x}',\, \mathfrak{p}'|\mathfrak{x}_0,\, \mathfrak{p}_0) = \delta(\mathfrak{x}' - \mathfrak{x}_T)\delta(\mathfrak{p}' + \mathfrak{p}_T), \tag{3.17}$$

the acceptance probability becomes non-zero and is given by

$$\begin{aligned} A(\mathfrak{x}_T,\, -\mathfrak{p}_T|\mathfrak{x}_0,\, \mathfrak{p}_0) &= \min\left(1,\, \frac{p(\mathfrak{x}_T,\, -\mathfrak{p}_T)}{p(\mathfrak{x}_0,\, \mathfrak{p}_0)} \frac{g(\mathfrak{x}_0,\, \mathfrak{p}_0|\mathfrak{x}_T,\, -\mathfrak{p}_T)}{g(\mathfrak{x}_T,\, -\mathfrak{p}_T|\mathfrak{x}_0,\, \mathfrak{p}_0)}\right) \\ &= \min\left(1,\, \frac{p(\mathfrak{x}_T,\, -\mathfrak{p}_T)}{p(\mathfrak{x}_0,\, \mathfrak{p}_0)} \frac{\delta(\mathfrak{x}_0 - \mathfrak{x}_0)\delta(\mathfrak{p}_0 - \mathfrak{p}_0)}{\delta(\mathfrak{x}_T - \mathfrak{x}_T)\delta(-\mathfrak{p}_T + \mathfrak{p}_T)}\right) \\ &= \min\left(1,\, e^{-\mathbb{H}(\mathfrak{x}_T,\, -\mathfrak{p}_T)+\mathbb{H}(\mathfrak{x}_0,\, \mathfrak{p}_0)}\right). \end{aligned} \tag{3.18}$$

A popular choice for the kinetic term $\mathcal{T}$ is $\mathcal{T}(\mathfrak{x},\, \mathfrak{p}) = \frac{1}{2}\mathfrak{p}^T\mathcal{M}^{-1}\mathfrak{p}$, where $\mathcal{M}$ is the mass matrix. This choice is known as Euclidian-Gaussian kinetic energy [16]. Pseudocode for the Hybrid Monte Carlo algorithm is given in appendix B.4.

3.5.1. Hybrid Monte Carlo on the Sphere

Since the degrees of freedom considered in the Ising model and Double Exchange model are elements of $SO(n)$, it is natural to construct a Hybrid Monte Carlo method on the sphere. The following discussion will focus on spins $\boldsymbol{S} \in \mathbb{S}^2$.

Derivatives on the Sphere

A crucial component of the Hybrid Monte Carlo method are derivatives. The problem with cartesian derivatives of a spin is that the norm may not be conserved. To construct a norm conserving derivative, a new spin $\boldsymbol{S}'$ is considered. $\boldsymbol{S}'$ is constructed by rotating the original spin $\boldsymbol{S}$ around the rotation axis $\boldsymbol{r}$ by an infinitesimal angle ϑ. The rotation around $\boldsymbol{r}$ can be expressed using the exponential map $S \to e^{\hat{\boldsymbol{r}}}S = e^{r_i \boldsymbol{L}_i}S$, where $\boldsymbol{L}_i$ are the generators of $\mathfrak{so}(3)$. The transformed spin thus reads

$$\boldsymbol{S}' = \boldsymbol{S} + \delta\boldsymbol{S} = e^{\vartheta r_i \boldsymbol{L}_i}\boldsymbol{S} = \boldsymbol{S} + \vartheta r_i \boldsymbol{L}_i \boldsymbol{S}. \tag{3.19}$$

Using the representation $(\boldsymbol{L}_i)_{jk} = -\varepsilon_{ijk}$, $\delta\boldsymbol{S}$ is given as

$$\delta S_j = \vartheta[r_i \boldsymbol{L}_i]_{jk} S_k = -\vartheta r_i \varepsilon_{ijk} S_k = \vartheta(\boldsymbol{r} \times \boldsymbol{S})_j. \tag{3.20}$$

Expanding a function $f(\boldsymbol{S}')$ for small changes, the result

$$f(\boldsymbol{S}') = f(\boldsymbol{S} + \delta\boldsymbol{S}) = f(\boldsymbol{S}) + \delta\boldsymbol{S}\nabla f(\boldsymbol{S}) = f(\boldsymbol{S}) + \vartheta(\boldsymbol{r} \times \boldsymbol{S}) \cdot \nabla f(\boldsymbol{S}) \tag{3.21}$$

is obtained.
With the obtained expressions, the directional derivative $\mathcal{D}_{\boldsymbol{r}} f(\boldsymbol{S})$ can be defined as

$$\mathcal{D}_{\boldsymbol{r}} f(\boldsymbol{S}) = \lim_{\vartheta \to 0} \frac{f(\boldsymbol{S} + \delta\boldsymbol{S}) - f(\boldsymbol{S})}{\vartheta} = \boldsymbol{r} \cdot \big[\boldsymbol{S} \times \nabla f(\boldsymbol{S})\big], \tag{3.22}$$

and the components of the derivative can be identified as

$$\mathcal{D}_{\boldsymbol{r}} f(\boldsymbol{S}) = \boldsymbol{r} \cdot \mathcal{D} f(\boldsymbol{S}), \tag{3.23}$$

$$\mathcal{D} f(\boldsymbol{S}) = \boldsymbol{S} \times \nabla f(\boldsymbol{S}). \tag{3.24}$$

Some properties of $\mathcal{D}_{\boldsymbol{r}} f(\boldsymbol{S})$ are:

- The main advantage of this derivative is that it is – in contrast to the derivative in spherical coordinates – defined everywhere, even at the poles.
- For $\boldsymbol{r} = \boldsymbol{S}$, $\mathcal{D}_{\boldsymbol{S}} f(\boldsymbol{S}) = 0$. This is expected, since the rotation of a vector around itself does not change the vector.
- The derivative $\mathcal{D} f(\boldsymbol{S})$ is part of the tangent space of the $\mathbb{S}^2$.

Equations of Motion

To introduce an $SO(3)$ symmetry into the Hybrid Monte Carlo method, the normalization condition $\|\boldsymbol{S}\| = 1$ is added to the partition function and the auxiliary momentum variables $\mathfrak{p}$ are introduced [17]. The result is

$$\begin{aligned} \mathcal{Z} &= \int \mathcal{D}\,\boldsymbol{S} e^{-\beta\mathcal{H}(\boldsymbol{S})} = \int \mathrm{d}^3\boldsymbol{S}\, \delta(\|\boldsymbol{S}\| - 1) e^{-\beta\mathcal{H}(\boldsymbol{S})} \\ &= \frac{1}{\mathcal{Z}_{\mathfrak{p}}} \int \mathrm{d}^3\boldsymbol{S} \int \mathrm{d}^3\mathfrak{p}\, \delta(\|\boldsymbol{S}\| - 1) e^{-\underbrace{[\beta\mathcal{H}(\boldsymbol{S}) + \frac{1}{2}\|\mathfrak{p}\|^2]}_{\mathbb{H}(\boldsymbol{S},\mathfrak{p})}} \end{aligned} \tag{3.25}$$

and the equations of motion (EOMs) read

$$\frac{\mathrm{d}\boldsymbol{S}}{\mathrm{d}t} = \frac{\partial \mathbb{H}(\boldsymbol{S}, \mathfrak{p})}{\partial \mathfrak{p}} = \mathfrak{p} \times \boldsymbol{S}, \qquad \frac{\mathrm{d}\mathfrak{p}}{\mathrm{d}t} = -\frac{\partial \mathbb{H}(\boldsymbol{S}, \mathfrak{p})}{\partial \boldsymbol{S}} = -\beta \mathcal{D}\mathcal{H}(\boldsymbol{S}). \tag{3.26}$$

Using these equations, symplectic integrators can be adapted to conserve the norm of the spins [17, 18].
To derive the EOM for $\mathfrak{p}$, the derivative of $\mathbb{H}(\boldsymbol{S}, \mathfrak{p})$ with respect to $\boldsymbol{S}$ must be taken. As shown previously, the partial derivative with respect to $\boldsymbol{S}$ can be calculated using equation (3.23).

$$-\frac{\partial \mathbb{H}(\boldsymbol{S}, \mathfrak{p})}{\partial \boldsymbol{S}} = -\mathcal{D}\mathbb{H}(\boldsymbol{S}, \mathfrak{p}) = -\beta \mathcal{D}\mathcal{H}(\boldsymbol{S}) = -\beta \boldsymbol{S} \times \nabla \mathcal{H}(\boldsymbol{S}) = \dot{\mathfrak{p}}. \tag{3.27}$$

A possible integration scheme for equation (3.27) is

$$\mathfrak{p}(t + \Delta t) = \mathfrak{p}(t) + \Delta t \dot{\mathfrak{p}} = \mathfrak{p}(t) + \Delta t \beta \nabla \mathcal{H}\big(\boldsymbol{S}(t)\big) \times \boldsymbol{S}(t). \tag{3.28}$$

The EOM for the spins $\boldsymbol{S}$ carry a bit more subtly. The derivative $\frac{\partial \mathbb{H}(\boldsymbol{S}, \mathfrak{p})}{\partial \mathfrak{p}}$ is not performed directly, however, auxiliary fields $\mathfrak{K}$ are introduced. These fields parametrize the spins $\{\boldsymbol{S}\}$ with respect to a reference spin $\boldsymbol{S}^0$. The parametrization reads

$$\boldsymbol{S} = e^{\mathfrak{K}_i \boldsymbol{L}_i} \boldsymbol{S}^0. \tag{3.29}$$

Using this parametrization, the EOM for $\mathfrak{K}$ can be calculated to be

$$\dot{\mathfrak{K}} = \frac{\partial \mathbb{H}(\boldsymbol{S}, \mathfrak{p})}{\partial \mathfrak{p}} = \mathfrak{p}. \tag{3.30}$$

A possible integration scheme again is

$$\mathfrak{K}(t + \Delta t) = \mathfrak{K}(t) + \Delta t \dot{\mathfrak{K}} = \mathfrak{K}(t) + \Delta t \mathfrak{p}(t). \tag{3.31}$$

If the $\mathfrak{so}(3)$ generators are multiplied from the right hand side and the resulting equation is exponentiated, an expression for the time evolution of $\boldsymbol{S}$ is re-

trieved

$$\underbrace{e^{\mathfrak{K}(t+\Delta t)\boldsymbol{L}}}_{\boldsymbol{S}(t+\Delta t)} = \underbrace{e^{\mathfrak{K}_i(t)\boldsymbol{L}_i}}_{\boldsymbol{S}(t)} e^{\Delta t \mathfrak{p}_j(t)\boldsymbol{L}_j}. \tag{3.32}$$

Using equations (3.29) and (3.30) an explicit expression for $\dot{\boldsymbol{S}}$ can be calculated. It reads

$$\begin{aligned} \dot{\boldsymbol{S}}_i(t) &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\left[e^{\mathfrak{K}_j(t)\boldsymbol{L}_j}\right]_{ik}\boldsymbol{S}_k^0\right) = \\ &= \left[\dot{\mathfrak{K}}_j(t)\boldsymbol{L}_j\right]_{ik}\boldsymbol{S}_k(t) = \\ &= \left[\mathfrak{p}_j(t)\boldsymbol{L}_j\right]_{ik}\boldsymbol{S}_k(t) = \\ &= -\mathfrak{p}_j(t)\varepsilon_{jik}\boldsymbol{S}_k(t) = \\ &= (\mathfrak{p}\times\boldsymbol{S})_i \end{aligned} \tag{3.33}$$

and matches precisely with the expression given in equation (3.26).
To summarize, the equations of motion read

$$\dot{\boldsymbol{S}} = \mathfrak{p}\times\boldsymbol{S}, \qquad\qquad \dot{\mathfrak{p}} = -\beta\boldsymbol{S}\times\nabla\mathcal{H}(\boldsymbol{S}), \tag{3.34}$$

and possible integration steps are

$$\boldsymbol{S}(t+\Delta t) = e^{\Delta t\mathfrak{p}(t)\cdot\boldsymbol{L}}\boldsymbol{S}(t), \tag{3.35}$$

$$\mathfrak{p}(t+\Delta t) = \mathfrak{p}(t) + \Delta t\beta\nabla\mathcal{H}\big(\boldsymbol{S}(t)\big)\times\boldsymbol{S}(t). \tag{3.36}$$

3.5.2. Dual Averaging

When performing Hamiltonian Monte Carlo simulations the chosen step size Δt and trajectory length L have a profound impact on the simulation efficiency. If the step size is chosen to be too small, the computational cost is very high without proportional gains in sampling efficiency. Conversely, a very large step size leads to numerical instability and poor acceptance rates.
Thus, the choice of Δt and L must be made deliberately to maximize the efficiency. This, however, is not always trivial or even possible. A possibility for optimizing the parameters is to adapt the step size Δt dynamically. The *Dual Averaging* scheme is such an algorithm [19].

Overview of the Algorithm

Dual averaging is a stochastic optimization method for solving convex optimization problems. The core idea is to adapt the step size until the target acceptance ratio is reached. This is achieved through maintaining a running average of a quantity related to the discrepancy between the observed acceptance ratio and the target acceptance rate. Let H_m denote the error at iteration m

$$H_m = \delta - \alpha_m, \tag{3.37}$$

where δ is the target acceptance rate and α_m is the observed acceptance ratio at iteration m. The update to the step size Δt is performed using

$$\bar{H}_m = \tfrac{1}{m+t_0} H_m + \left(1 - \tfrac{1}{m+t_0}\right) \bar{H}_{m-1}, \qquad \bar{H}_0 = 0, \tag{3.38}$$

$$\log \Delta t_m = \mu - \tfrac{\sqrt{m}}{\gamma} \bar{H}_m, \tag{3.39}$$

$$\log \overline{\Delta t}_m = \tfrac{1}{m^\kappa} \log \Delta t_m + \left(1 - \tfrac{1}{m^\kappa}\right) \log \overline{\Delta t}_{m-1}, \qquad \overline{\Delta t}_0 = 1, \tag{3.40}$$

where μ is a user-defined parameter related to the initial guess for the step size, γ is a learning rate parameter, and $\bar{H}_m$ is the running average of the acceptance error. κ and t_0 are adaptions to the original algorithm proposed in [19], which weigh recent configurations stronger and prevent a step size which tends to 0, respectively. The dual averaging update rule incorporates a decaying step size schedule to ensure convergence, while mitigating the effect of early noise in the sampling process. Usually, the update is only performed during the warm-up phase, upon completeness, $\overline{\Delta t}_m \to \Delta t$.
Using this update procedure for the step size an overall acceptance ratio of δ can usually be achieved. Dual averaging is a core component in the advanced No–U–Turn Sampler (NUTS) algorithm, which ensures optimal computational efficiency [20].

3.5.3. The No-U-Turn Sampler

The No-U-Turn Sampler (NUTS) is an advanced version of the HMC algorithm, which dynamically builds the trajectory until the proposed trajectory starts to

turn back on itself. The main advantage of NUTS is that the optimal trajectory length L is determined during the simulation and no manual tuning is needed. The necessity of tuning the step size Δt can be eliminated by the previously described dual averaging approach. Thus, both, L and Δt will be determined automatically to reach the target acceptance ratio and maximize the computational efficiency of the simulation [20].

Overview of the Algorithm

NUTS augments the Hamiltonian model $p(\mathfrak{x}, \mathfrak{p}) \propto e^{-\mathbb{H}(\mathfrak{x}, \mathfrak{p})}$ by introducing a slice variable u. The joint probability of the position $\mathfrak{x}$, momentum $\mathfrak{p}$, and slice variable u is given by

$$p(\mathfrak{x}, \mathfrak{p}, u) \propto \mathcal{I}\left\{u \in [0, e^{-\mathbb{H}(\mathfrak{x}, \mathfrak{p})}]\right\}, \tag{3.41}$$

where $\mathcal{I}\{\cdot\}$ is defined as

$$\mathcal{I}\{w\} = \begin{cases} 1, & w \text{ is True} \\ 0, & \text{otherwise} \end{cases}. \tag{3.42}$$

After integrating out u, the marginal probability of $\mathfrak{x}$ and $\mathfrak{p}$ reduces to $p(\mathfrak{x}, \mathfrak{p}) = e^{-\mathbb{H}(\mathfrak{x}, \mathfrak{p})}$, as in standard HMC. The conditional distributions $p(u|\mathfrak{x}, \mathfrak{p})$ and $p(\mathfrak{x}, \mathfrak{p}|u)$ are uniform within the slice and the following conditions hold:

- $p(u|\mathfrak{x}, \mathfrak{p})$ is uniform over the interval $[0, e^{-\mathbb{H}(\mathfrak{x}, \mathfrak{p})}]$,
- $p(\mathfrak{x}, \mathfrak{p}|u)$ is uniform if and only if $u \leq e^{-\mathbb{H}(\mathfrak{x}, \mathfrak{p})}$.

To add further flexibility to the sampling process, two sets of configurations are introduced. The set $\mathcal{B}$, containing all configurations generated during the NUTS iteration and the set $\mathcal{C}$, which is a subset of $\mathcal{B}$. The configurations for $\mathcal{B}$ are generated by integrating randomly in the forward or backward direction, whereas the configurations in $\mathcal{C}$ are chosen deterministically from $\mathcal{B}$.

The key here is that the set $\mathcal{C}$ must satisfy a set of conditions to ensure that the overall procedure leaves the joint distribution $p(\mathfrak{x}, \mathfrak{p}, u, \mathcal{B}, \mathcal{C}|\Delta t)$ invariant. The conditions are

1. transformations of $\mathfrak{x}$, $\mathfrak{p}$ when adding a new state to $\mathcal{C}$ must preserve volume, i.e., the Jacobian determinant must be 1,

2. $p((\mathfrak{x},\, \mathfrak{p}) \in \mathcal{C}|\mathfrak{x},\, \mathfrak{p},\, u,\, \Delta t) = 1$, e.g, the current configuration $(\mathfrak{x},\, \mathfrak{p})$ is included in $\mathcal{C}$,
3. all states in $\mathcal{C}$ must satisfy $u \leq e^{-\mathbb{H}(\mathfrak{x}_0, \mathfrak{p}_0)}$,
4. the conditional distribution $p(\mathcal{B},\, \mathcal{C}|\mathfrak{x},\, \mathfrak{p},\, u,\, \Delta t)$ must be invariant under any permutation of $\mathcal{C}$.

To choose a new configuration during the NUTS update, the following steps are taken:

1. sample $\mathfrak{p} \sim \mathcal{N}(0,\, 1)$,
2. sample $u \sim \mathcal{U}(0,\, e^{-\mathbb{H}(\mathfrak{x}, \mathfrak{p})})$ from a uniform distribution,
3. sample $\mathcal{B}$ and $\mathcal{C}$ from the corresponding conditional probability,
4. sample the new position $\mathfrak{x}_{m+1}$ and momentum $\mathfrak{p}_{m+1}$ using a transition kernel $\mathcal{T}(\mathfrak{x}^m,\, \mathfrak{p}, \mathcal{C})$ that leaves the uniform distribution over $\mathcal{C}$ invariant.

To leave the uniform distribution invariant, the transition kernel $\mathcal{T}$ must satisfy

$$\sum_{(\mathfrak{x},\, \mathfrak{p}) \in \mathcal{C}} \mathcal{T}(\mathfrak{x}',\, \mathfrak{p}'|\mathfrak{x},\, \mathfrak{p},\, \mathcal{C}) = \mathcal{I}\{(\mathfrak{x}',\, \mathfrak{p}') \in \mathcal{C}\}. \tag{3.43}$$

This ensures that each state in $\mathcal{C}$ has an equal probability of being chosen.

The specific probability distribution $p(\mathcal{B},\, \mathcal{C}|\mathfrak{x},\, \mathfrak{p},\, u,\, \Delta t)$ used by NUTS is defined by a binary tree of configurations $(\mathfrak{x},\, \mathfrak{p})$. At the start, the tree contains a single node corresponding to the current state $(\mathfrak{x},\, \mathfrak{p})$. Each subsequent iteration of the algorithm doubles the size of the tree by adding more position-momentum states. The tree expansion consists of:

- A direction v_j is randomly chosen from $\{+1,\, -1\}$, and 2^j integrations steps are taken, where j is the current depth of the tree.
- If $v_j = +1$, the left half of the tree remains unchanged, while the right half corresponds to a balanced binary tree of height j. If $v_j = -1$, the situation is reversed.
- This process is repeated recursively until either of the stopping criteria is satisfied.

The two primary stopping conditions are:

- **Stopping due to retracing:** The algorithm stops if, for any subtree, the position-momentum states at the leftmost $(-)$ and rightmost $(+)$ leaves sat-

isfy

$$(\mathfrak{x}^+ - \mathfrak{x}^-) \cdot \mathfrak{p}^- < 0 \quad \text{or} \quad (\mathfrak{x}^+ - \mathfrak{x}^-) \cdot \mathfrak{p}^+ < 0.$$

This condition ensures that the trajectory has made a 'U-turn', indicating that further exploration would revisit previously sampled regions in parameter space.

- **Stopping due to low probability:** The tree expansion stops if the probability falls below a threshold, indicating that further exploration would result in states with extremely low probability

$$e^{\Delta_{\max} - \mathbb{H}} < u,$$

where $\Delta_{\max}$ is a user-defined threshold, commonly chosen to be ~ 1000.

Once the trajectory has been fully explored and the doubling process is halted, the next state is sampled uniformly from the set $\mathcal{C}$, the collection of valid position-momentum states that satisfy the slice condition. This ensures that the distribution of sampled states remains consistent with the target distribution.

When implementing NUTS for the simulation of a process, the sampling stage and tree building can be combined, as described in [20]. Pseudocode for the advanced algorithm is given in algorithm 1.

Algorithm 1 *Pseudocode for the No-U-Turn Sampler with Dual Averaging [20]*

Given $\mathfrak{x}^0$, δ, $\mathcal{H}(\{\boldsymbol{S}\})$, N, N_{adapt}:
Initialize variables:
$\Delta t_0 = \text{ProposeStepSize}(\mathfrak{x})$, $\mu = \log(10\Delta t_0)$, $\overline{\Delta t}_0 = 1$, $\bar{H}_0 = 0$, $\gamma = 0.05$, $t_0 = 10$, $\kappa = 0.75$.
for m=1 to N **do**
  Sample $\mathfrak{p}^0 \sim \mathcal{N}(0, 1)$.
  Sample $u \sim \mathcal{U}[0, e^{-\mathbb{H}(\mathfrak{x}^{m-1}, \mathfrak{p}^0)}]$.
  Initialize $\mathfrak{x}^m = \mathfrak{x}^- = \mathfrak{x}^+ = \mathfrak{x}^{m-1}$, $\mathfrak{p}^- = \mathfrak{p}^+ = \mathfrak{p}^0$, $j = 0$, $n = s = 1$.
  while $s = 1$ **do**
    Choose random direction $v_j \sim \mathcal{U}(\{-1, 1\})$.
    if $v_j = 1$ **then**
      $-, -, \mathfrak{x}^+, \mathfrak{p}^+, \mathfrak{x}', n', s', \alpha, n_\alpha \leftarrow \textsc{BuildTree}(\mathfrak{x}^-, \mathfrak{p}^-, u, v_j, j, \Delta t_{m-1}, \mathfrak{x}^{m-1}, \mathfrak{p}^0)$.
    else
      $\mathfrak{x}^-, \mathfrak{p}^-, -, -, \mathfrak{x}', n', s', \alpha, n_\alpha \leftarrow \textsc{BuildTree}(\mathfrak{x}^-, \mathfrak{p}^-, u, v_j, j, \Delta t_{m-1}, \mathfrak{x}^{m-1}, \mathfrak{p}^0)$.
    end if
    if $s' = 1$ **then**
      With probability $\min(1, \frac{n'}{n})$: $\mathfrak{x}^m \leftarrow \mathfrak{x}'$.
    end if
    $n \leftarrow n + n'$.
    $s \leftarrow s'\mathcal{I}\{(\mathfrak{x}^+ - \mathfrak{x}^-) \cdot \mathfrak{p}^- \geq 0\}\mathcal{I}\{(\mathfrak{x}^+ - \mathfrak{x}^-) \cdot \mathfrak{p}^+ \geq 0\}$.
    $j \leftarrow j + 1$.
  end while
  if $m \leq N^{\text{adapt}}$ **then**
    Set $\bar{H}_m = \frac{1}{m+t_0}\left(\delta - \frac{\alpha}{n_\alpha}\right) + \left(1 - \frac{1}{m+t_0}\right)\bar{H}_{m-1}$.
    Set $\log \Delta t_m = \mu - \frac{\sqrt{m}}{\gamma}\bar{H}_m$.
    Set $\log \overline{\Delta t}_m = \frac{1}{m^\kappa}\log \Delta t_m + \left(1 - \frac{1}{m^\kappa}\right)\log \overline{\Delta t}_{m-1}$.
  else
    Set $\Delta t_m = \overline{\Delta t}_{N^{\text{adapt}}}$.
  end if
end for
function $\textsc{BuildTree}(\mathfrak{x}, \mathfrak{p}, u, v, j, \Delta t, \mathfrak{x}^0, \mathfrak{p}^0)$
  if $j = 0$ **then**
    $\mathfrak{x}', \mathfrak{p}' \leftarrow \textsc{Propagate}(\mathfrak{x}, \mathfrak{p}, v\Delta t)$.
    $n' = \mathcal{I}\{u \leq e^{-\mathbb{H}(\mathfrak{x}', \mathfrak{p}')}\}$.
    $s' = \mathcal{I}\{u < e^{\Delta_{\max} - \mathbb{H}(\mathfrak{x}', \mathfrak{p}')}\}$.
    $\alpha = \min\left(1, e^{\mathbb{H}(\mathfrak{x}^0, \mathfrak{p}^0) - \mathbb{H}(\mathfrak{x}', \mathfrak{p}')}\right)$.
    return $\mathfrak{x}', \mathfrak{p}', \mathfrak{x}', \mathfrak{p}', \mathfrak{x}', n', s', \alpha, 1$.
  else
    $\mathfrak{x}^-, \mathfrak{p}^-, \mathfrak{x}^+, \mathfrak{p}^+, \mathfrak{x}', n', s', \alpha', n'_\alpha \leftarrow \textsc{BuildTree}(\mathfrak{x}, \mathfrak{p}, u, v, j-1, \Delta t, \mathfrak{x}^0, \mathfrak{p}^0)$.
    if $s' = 1$ **then**
      if $v = 1$ **then**
        $--, --, \mathfrak{x}^+, \mathfrak{p}^+, \mathfrak{x}'', n'', s'', \alpha'', n''_\alpha \leftarrow \textsc{BuildTree}(\mathfrak{x}^+, \mathfrak{p}^+, u, v, j-1, \Delta t, \mathfrak{x}^0, \mathfrak{p}^0)$.
      else
        $\mathfrak{x}^-, \mathfrak{p}^-, --, --, \mathfrak{x}'', n'', s'', \alpha'', n''_\alpha \leftarrow \textsc{BuildTree}(\mathfrak{x}^-, \mathfrak{p}^-, u, v, j-1, \Delta t, \mathfrak{x}^0, \mathfrak{p}^0)$.
      end if
      With probability $\frac{n''}{n'+n''}$: $\mathfrak{x}' \leftarrow \mathfrak{x}''$.
      $\alpha' \leftarrow \alpha' + \alpha''$.
      $n'_\alpha \leftarrow n'_\alpha + n''_\alpha$.
      $s' \leftarrow s''\mathcal{I}\{(\mathfrak{x}^+ - \mathfrak{x}^-) \cdot \mathfrak{p}^- \geq 0\}\mathcal{I}\{(\mathfrak{x}^+ - \mathfrak{x}^-) \cdot \mathfrak{p}^+ \geq 0\}$.
      $n' \leftarrow n' + n''$.
    end if
    return $\mathfrak{x}^-, \mathfrak{p}^-, \mathfrak{x}^+, \mathfrak{p}^+, \mathfrak{x}', n', s', \alpha', n'_\alpha$.
  end if
end function

4. Effective Hamiltonians

In this chapter, the basic properties of the effective Hamiltonians – namely invariance and equivariance – will be discussed on the basis of global $SO(3)$ rotations. Once the importance of equivariance is demonstrated, the effective Hamiltonians used for later analysis will be introduced and motivated.

4.1. General Properties

Before the considered effective Hamiltonians are discussed, general properties of the Hamiltonians will be considered. These properties apply to the effective, as well as the exact Hamiltonians. These considerations are important because a well designed effective Hamiltonian can reproduce the exact behaviour better than a randomly chosen one. The properties considered are (group) invariance and (group) equivariance.

4.1.1. Invariance

In physics, quantities that remain unchanged under a defined set of transformations are invariant with respect to these transformations. Such invariants – energy, charge, angular momentum – form the bedrock of physical law because they encode universal, observer-independent statements about a system. Noether's theorem shows that each continuous symmetry is linked to an invariant quantity of the system. In other words, each transformation that leaves the system invariant is linked to a conserved quantity [21]. When constructing effective Hamiltonians, insisting on the same invariance properties that the exact Hamiltonian possesses guarantees that the low-energy description respects the underlying symmetries of the full theory.

Mathematically, invariance can be defined in the following way.
A function $\mathcal{F} : X \to Y$, $\Phi \mapsto \mathcal{F}(\Phi)$ is invariant with respect to the group transformations $g \in G$, if

$$\mathcal{F}(D_X[g]\Phi) = \mathcal{F}(\Phi), \tag{4.1}$$

where $\Phi \in X$, $\mathcal{F}(\Phi) \in Y$ and $D_X[g]$ is the representation of g in X.
For the exact Hamiltonians considered in chapter 2 it can be verified that global spin flips are invariant transformations for the Ising model and the Ising-like plaquette model. For the Double Exchange model, global $SO(3)$ rotations leave the observables invariant. Consequently, all constructed effective Hamiltonians should include a global $SO(3)$ invariance. This can be achieved if spin-spin interactions are considered, since the expression $\boldsymbol{S}_i \cdot \boldsymbol{S}_j$ is $SO(3)$ invariant, as shown by

$$\boldsymbol{S}_i \cdot \boldsymbol{S}_j \to (R\boldsymbol{S}_i) \cdot (R\boldsymbol{S}_j) = \boldsymbol{S}_i^{\mathsf{T}} \underbrace{R^{\mathsf{T}} R}_{\mathbb{I}} \boldsymbol{S}_j = \boldsymbol{S}_i^{\mathsf{T}} \boldsymbol{S}_j = \boldsymbol{S}_i \cdot \boldsymbol{S}_j. \tag{4.2}$$

4.1.2. Equivariance

In contrast to invariance, equivariance is the property that the output of a transformation $\mathcal{F}$ transforms in the same way as the input. Mathematically, a function $\mathcal{F} : X \to Y$, $\Phi \mapsto \mathcal{F}(\Phi)$ is equivariant with respect to the group transformations $g \in G$, if

$$\mathcal{F}(D_X[g]\Phi) = D_Y[g]\mathcal{F}(\Phi), \tag{4.3}$$

where $\Phi \in X$, $\mathcal{F}(\Phi) \in Y$ and $D_X[g]$, $D_Y[g]$ is the representation of g in X and Y, respectively.
In the context of effective Hamiltonians, equivariance guarantees that any term of the form $\boldsymbol{S}_i^{\text{eff}} \cdot \boldsymbol{S}_j^{\text{eff}}$ is invariant under transformations $\boldsymbol{S} \to \boldsymbol{S}^{\text{eff}} = \mathcal{F}(\boldsymbol{S})$, provided that $\mathcal{F}^{\mathsf{T}} = \mathcal{F}^{-1}$.
Equivariance in the context of machine learning allows models to naturally handle transformations, ensuring that the model's output transforms in a predictable way when the input undergoes these transformations. This leads to improved parameter efficiency, as the network can share filters across different transforma-

tions, reducing the number of parameters needed and enhancing generalization. Additionally, equivariance makes the network more robust to variations in the input, such as changes in orientation or viewpoint, which is especially valuable in domains like object recognition, medical imaging, and robotics where such variations are common. Group equivariant Convolutional Neural Networks represent a widespread model class which is used for these applications [22].

4.2. Neural Networks as Effective Hamiltonians

The preceding sections have established that any effective Hamiltonian should respect the global $SO(3)$ invariance dictated by the exact model. Beyond this symmetry requirement, the practical task is to choose a functional form that is sufficiently expressive to reproduce the low-energy physics while remaining tractable for optimisation. In the following, four complementary ansätze are explored. They are loosely arranged by increasing structural complexity: a strictly linear nearest-neighbour model, a transformer-like architecture that learns an effective spin field, a fully-connected interaction matrix, and a convolutional-attention hybrid inspired by modern deep-learning practice. Each construction is accompanied by a discussion of its symmetry properties, parameter count, and expected range of validity.

4.2.1. Linear Nearest Neighbour Interactions

The most basic effective Hamiltonians considered in this work are

$$\mathcal{H}_{\text{eff}}^{\mathfrak{L}} = E_0 - \sum_{k,\langle i,j\rangle_k} J_k^{\text{eff}} \boldsymbol{S}_i \cdot \boldsymbol{S}_j \tag{4.4}$$

and

$$\mathcal{H}_{\text{eff}}^{\mathfrak{L}z} = E_0 - \sum_{k,\langle i,j\rangle_k} J_k^{\text{eff}} S_i^z S_j^z. \tag{4.5}$$

They consist of n nearest-neighbour interaction terms and a constant energy offset E_0. Each of the k-th-nearest-neighbour interactions has its own coupling constant

J_k^{eff}. Invariance of the Hamiltonian is guaranteed because of the spin-spin interaction $\boldsymbol{S}_i \cdot \boldsymbol{S}_j$. For some Hamiltonians, e.g., Ising model and Ising-like plaquette model, the 3d-spins get restricted to a reference axis – most commonly chosen to be the z axis – which has the consequence that not the full inner product of the spins $\boldsymbol{S}_i \cdot \boldsymbol{S}_j$ is needed, however, the z-component $S_i^z S_j^z$ is sufficient to capture the model's behaviour. This is equivalent to using 1d-spins $S_i^z \in \{+1, -1\}$ instead of 3d-spins $\boldsymbol{S} \in O(3)$.
The Hamiltonian $\mathcal{H}_{\text{eff}}^{\mathfrak{L}}$ is used for the Double Exchange model, while $\mathcal{H}_{\text{eff}}^{\mathfrak{L}z}$ is used for approximating the Ising-like plaquette model. Often – e.g., for the Ising-like plaquette model – only first order nearest-neighbour interactions must be considered and higher order interactions can be neglected. If this is done, the Hamiltonian reduces to

$$\mathcal{H}_{\text{eff}}^{\mathfrak{L}z} = E_0 - J^{\text{eff}} \sum_{\langle i,\, j \rangle} S_i^z S_j^z. \tag{4.6}$$

The number of trainable parameters scales with the nearest-neighbour order considered and is $c_p = 1 + n$. One parameter encodes the constant energy offset E_0, while the other n parameters describe the n coupling parameters J_k^{eff}.
The main advantage of this class of effective Hamiltonian is that in order to capture more complex behaviour it is often sufficient to include nearest-neighbour terms of higher order.

4.2.2. Spin-Transformer

A natural extension of the nearest-neighbour interactions is the use of an effective spin field $\{\boldsymbol{S}^{\text{eff}}\}$ instead of the exact one $\{\boldsymbol{S}\}$. Based on [23], a transformer-like architecture is chosen to generate the effective spin field. The adapted nearest-neighbour Hamiltonian is given by

$$\mathcal{H}_{\text{eff}}^{\mathfrak{T}} = E_0 - \sum_{k,\, \langle i,\, j \rangle_k} J_k^{\text{eff}} \mathbf{S}_i^{\text{eff}} \cdot \mathbf{S}_j^{\text{eff}}. \tag{4.7}$$

The effective spin field $\{\boldsymbol{S}^{\text{eff}}\}$ is built up iteratively using a neural network consisting of $\mathfrak{L}$ attention layers. The network architecture is given by

$$\begin{aligned} \mathbf{S}^{(0)} &\equiv \mathbf{S}, && (4.8)\\ \mathbf{S}^{(l)} &= \mathcal{N}\left[\mathbf{S}^{(l-1)} + \mathcal{A}^{\mathbf{S}}_{\theta^{(l)}}(\mathbf{S}^{(l-1)})\right], && (4.9)\\ \mathbf{S}^{\text{eff}} &\equiv \mathbf{S}^{(\mathfrak{L})}, && (4.10) \end{aligned}$$

where l is the layer index, $\mathcal{N}$ normalizes each of the spins and $\mathcal{A}^{\mathbf{S}}_{\theta^{(l)}}$ is a neural network map between spin configurations. $\theta^{(l)}$ represents all trainable parameters of the l-th layer. The structure of the neural network map $\mathcal{A}$ closely follows the structure introduced in [24] and represents an attention layer. The precise form of the attention mechanism is

$$\begin{aligned} \mathcal{A}^{\mathbf{S}}_{\theta^{(l)}}(\mathbf{S}) &= \hat{\mathcal{M}}\mathbf{S}^{\hat{V}},\\ \hat{\mathcal{M}}_{ij} &= \sigma\left(\tfrac{1}{\sqrt{3}}\mathbf{S}_i^{\hat{Q}} \cdot \mathbf{S}_j^{\hat{K}}\right), \qquad (4.11)\\ \theta^{(l)} &= \{Q^{(l)}, K^{(l)}, V^{(l)}\}. \end{aligned}$$

$\hat{\mathcal{M}}_{ij}$ is the attention matrix, a $\mathbb{L}\times\mathbb{L}$ matrix which encodes all spin-spin interactions. $\hat{Q}$, $\hat{K}$ and $\hat{V}$ are the sets of query, key and value weights, which can be interpreted as the representation of a local operator given by

$$\begin{aligned} \mathbf{S}^{\hat{\alpha}} &\equiv \hat{\alpha}\mathbf{S},\\ \mathbf{S}_i^{\hat{\alpha}} &\equiv \sum_{k,\langle i,j\rangle_k} \alpha_k \mathbf{S}_j. \qquad (4.12) \end{aligned}$$

The local operator calculates a new spin field $\boldsymbol{S}^{\hat{\alpha}}$ using up to n-th order nearest-neighbour interactions. The function $\sigma(\cdot)$ is a non-linear activation function [25], chosen to be the ReLU in this thesis.

To test if the proposed effective Hamiltonian could be suitable for approximating the Double Exchange model it must be shown that the effective Hamiltonian $\mathcal{H}^{\mathfrak{T}}_{\text{eff}}$ is invariant under global $SO(3)$ rotations. Furthermore, it must be shown that the proposed attention mechanism is equivariant under global rotations.

The invariance with respect to global rotations follows directly from the form of

the interaction term $\boldsymbol{S}_i^{\text{eff}} \cdot \boldsymbol{S}_j^{\text{eff}}$, provided the attention mechanism is equivariant. This means that the condition

$$(R\boldsymbol{S})^{\text{eff}} = R\boldsymbol{S}^{\text{eff}} \tag{4.13}$$

must hold if R is an $SO(3)$ rotation. From the structure of the local operator $\boldsymbol{S}^{\hat{\alpha}}$, it follows that the operator is equivariant, e.g. $(R\boldsymbol{S})^{\hat{\alpha}} = R\boldsymbol{S}^{\hat{\alpha}}$, and thus the attention matrix is invariant under global $SO(3)$ rotations. If the attention matrix is invariant under global rotations it immediately follows that the attention mechanism $\mathcal{A}^{\boldsymbol{S}}_{\theta^{(l)}}$ is equivariant and thus the effective Hamiltonian is invariant under global rotations. In a similar way the invariance of the effective Hamiltonian with respect to global lattice translations can be shown [4].
The transformer-like effective Hamiltonians shows a promising scaling law for the number of trainable parameters. If nearest-neighbours up to n_1-th order are considered for spin-spin interaction, the local operator considers nearest-neighbour interactions up to n_2-th order and the number of attention layers is $\mathfrak{L}$ then the total number of trainable parameters is $c_p = 1 + n_1 + 3 \times \mathfrak{L} \times (n_2 + 1)$.

4.2.3. Fully Connected

The previous two effective Hamiltonians used nearest-neighbour interactions of k-th order as a basic building block. Although this is a somewhat canonical choice for lattice systems, it nevertheless is a strong restriction. A generalised approach would be to allow arbitrary spin-spin interactions without limiting them to nearest neighbour interactions. An implicit advantage of the nearest-neighbour interactions is the lattice size independence of the constructed Hamiltonians. A general interaction term of the form $x^{\mathsf{T}}Ax$ would, however, be interesting to study.
A natural choice for such a Hamiltonian applied to spin systems is given by

$$\mathcal{H}^{\mathfrak{F}}_{\text{eff}} = E_0 + \text{Tr}\left[\mathbf{S}^{\mathsf{T}}\hat{\mathcal{J}}_{\mathbb{L}}\mathbf{S}\right]. \tag{4.14}$$

E_0 again is a constant energy offset, $\hat{\mathcal{J}}_{\mathbb{L}}$ is the $\mathbb{L} \times \mathbb{L}$ coupling matrix for a lattice of size $\mathbb{L} = N \times N$ and Tr is the trace operator. Invariance with respect to global $SO(3)$ rotations is again guaranteed by the form of the spin-spin interaction.

Equivariant spin transformations, similar to the attention mechanism previously described, could be added to the model, unfortunately, the scope of the thesis did not allow for this.
Such a fully connected effective Hamiltonian should reproduce complex emergent behaviour relatively well, since each coupling can be adjusted individually to allow for fine-grained control. A major disadvantage in comparison to the previously discussed Hamiltonians is the dependence on the lattice size. If a simulation for a different lattice size should be performed, the presently proposed Hamiltonian must be retrained. The number of trainable parameters is $c_p = 1 + \mathbb{L}^2$, which is significantly higher than for the previously proposed models.

4.2.4. Spin-CNN

The last effective Hamiltonian considered in this thesis takes inspiration from convolutional neural networks and transformer architectures. The main idea is that the query, key and value matrices [24] get generated by a convolution. E.g. the local operator from the transformer-like Hamiltonian (see equation (4.12)) gets replaced by a convolution.
The convolution $\Gamma(\boldsymbol{S})_i$ of order n will perform a weighted sum of the $n \times n$ spins surrounding the spin $\boldsymbol{S}_i$, where $n = 2k + 1$. A visual representation of the kernel is given in figure 4.1, where the spins are represented by the black dots, the highlighted spin is the spin for which the convolution is performed and the blue square visualizes the convolution kernel. Mathematically, the convolution is defined as

$$(\omega * \boldsymbol{S})_i = \Gamma_{(\omega)}(\mathbf{S})_i = \mathcal{N}\Big[\sum_j \omega_j \mathbf{S}_{i+j}\Big], \tag{4.15}$$

where ω is the convolution kernel and j sums over the neighbouring spins in the $n \times n$ region.
The Hamiltonian consists of a constant energy offset E_0 and the sum over all inner products of the current spin field and a convoluted spin field scaled with an effective coupling constant J^{eff}. Additionally, the actual spin field gets replaced by

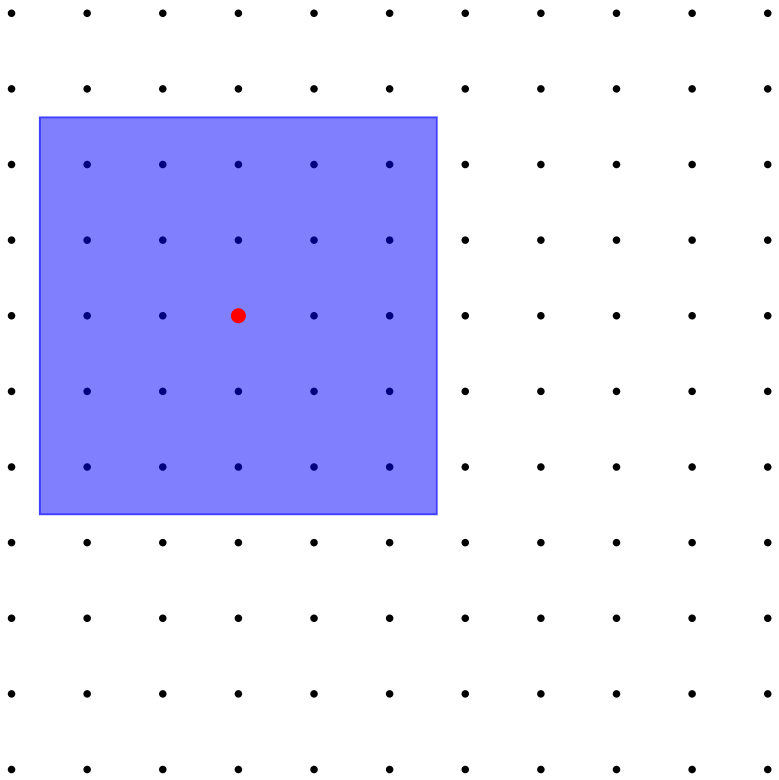


Figure 4.1.: *Visualization of a 2D spin convolution.*

an effective spin field $\boldsymbol{S}^{\text{eff}}$. The resulting Hamiltonian is

$$\mathcal{H}^{\mathfrak{C}}_{\text{eff}} = E_0 + J^{\text{eff}} \sum_i \mathbf{S}^{\text{eff}}_i \cdot \Gamma_{(\omega)}(\mathbf{S}^{\text{eff}})_i. \tag{4.16}$$

The effective spin field is constructed using a combination of convolution layers Γ and attention layers $\mathfrak{A}$. The attention layer is defined in a similar way as described previously, however, $\boldsymbol{S}^{\hat{\alpha}} \to \Gamma^{(\alpha)}(\boldsymbol{S})$. Again, ReLU is used as a non-linear activation function and the resulting attention spin field is added to the original field.
If the layer number of the model is $\mathfrak{L}$, the original spin field gets transformed using $\mathfrak{L}$ convolution layers, followed by $\mathfrak{L}$ attention layers. In addition to the growing number of layers, the convolution kernel for each layer depends on the layer index, namely the kernel size for layer ℓ is $n_\ell = [(2\ell + 1) \mod (2\mathfrak{L} + 1)]$.

The full definition of the effective spin field is given by

$$\begin{aligned} \boldsymbol{S}^{\text{eff}} &= \mathfrak{A}^{\mathfrak{L}}(\Gamma^{\mathfrak{L}}(\mathbf{S})), \\ \mathfrak{A}(\mathbf{S})_i &= \mathcal{N}\left(\mathbf{S}_i + \sum_j \sigma\left[\tfrac{1}{\sqrt{3}}\Gamma_{(Q)}(\mathbf{S})_i \cdot \Gamma_{(K)}(\mathbf{S})_j\right] \cdot \Gamma_{(V)}(\mathbf{S})_j\right), \\ \mathcal{O}^n &= \underbrace{\mathcal{O} \circ \mathcal{O} \circ \cdots \circ \mathcal{O}}_{\text{n-times}}. \end{aligned} \tag{4.17}$$

Regarding the symmetries of the spin-CNN, it is apparent that the defined convolution is equivariant with respect to global $SO(3)$ rotations

$$(\omega * R\boldsymbol{S}) = R(\omega * \boldsymbol{S}), \tag{4.18}$$

and thus the attention layer $\mathfrak{A}$ transforms equivariantly and the effective Hamiltonian $\mathcal{H}^{\mathfrak{L}}_{\text{eff}}$ is invariant under global $SO(3)$ rotations. The number of trainable parameters scales like $c_p = 2 + n_\omega^2 + \mathfrak{L} + \frac{4}{3}\mathfrak{L}(\mathfrak{L}+1)(\mathfrak{L}+2)$, where n_ω is the shape of the outermost convolution. Within the scope of this thesis, $n_\omega = 3$ was chosen. The parameter count depends highly on the model depth ($\mathfrak{L}^3$) and scales worse than the previous models. Furthermore, the model does not display the global $\mathcal{C}_4$ lattice symmetry. Thus, it is to be expected that the effective spin-CNN performs worse than the previously proposed models.

5. Training Effective Hamiltonians

This chapter addresses the training of effective Hamiltonians introduced in the previous section. While their construction defines the functional form and symmetry properties, it remains necessary to determine suitable parameters such that the effective model reproduces the behaviour of the exact Hamiltonian. To this end, different training strategies are employed, ranging from supervised fits of energy spectra to unsupervised approaches based on reweighting techniques. The following discussion illustrates these procedures and highlights numerical aspects relevant for stability and efficiency.

5.1. Plaquette Model

The training procedure for the Ising-like plaquette model is heavily inspired by [3]. In preparation for training the effective Hamiltonian, training data must be generated. Therefore, the simulation is performed using local updates and all accept/reject steps are performed using the exact Hamiltonian. The simulation is performed near the critical point and the parameters of the Hamiltonian were set to $t/J = 0.2$ and $T = \sqrt{10}$.
Once the training data is generated, the model parameters of the effective Hamiltonian can be fitted to the data. $\mathcal{H}_{\text{eff}}^{\mathfrak{L}_z}$ given in equation (4.6), was chosen as an effective Hamiltonian. As shown in [3], higher order interaction terms can be neglected. The loss function for training is defined to be

$$\text{loss}(\{\boldsymbol{S}\}) = \left[{}_{PQ}\mathcal{H}(\{\boldsymbol{S}\}) - \mathcal{H}_{\text{eff}}^{\mathfrak{L}_z}(\{\boldsymbol{S}\}) \right]^2, \tag{5.1}$$

which is the mean square error. The resulting trained parameters are $E_0 = 1.181$ and $J^{\text{eff}} = 1.116$. Thus, the trained effective Hamiltonian is

$$\mathcal{H}_{\text{eff}}^{\mathfrak{L}_z} = 1.181 - 1.116 \sum_{\langle i,j \rangle} S_i^z S_j^z. \tag{5.2}$$

The similarity of the obtained parameters E_0, J^{eff} and the parameters stated in [3] shows that the training procedure yields sensible results. The described procedure is a supervised training procedure [26], which will be adapted for the Double Exchange model.

5.2. Double Exchange Model

As discussed in the previous section, only one effective Hamiltonian was trained to approximate the exact Ising-like plaquette Hamiltonian. For the Double Exchange model, different effective Hamiltonians will be trained and consequently compared.
The general training procedure is the same for all effective Hamiltonians and will be discussed in the following. Additionally, considerations regarding numerical stability will be made.

5.2.1. Numerical Stability

In contrast to the Ising model or the plaquette model, the energy calculation for the Double Exchange model is more involved. The reason being that the Hamiltonian describes the interaction of fermions with classical spins $\{\boldsymbol{S}\}$. Thus, the Hamiltonian is a matrix, as described in section 2.3.
When calculating observables, the trace over the Boltzmann weight, $\mathcal{W} = \text{Tr}\left[e^{-\beta\mathcal{H}}\right]$, must be calculated. For a scalar valued Hamiltonian this operation is trivial, however, for matrix valued Hamiltonians the calculation is more involved. Using the grand canonical ensemble and the fact that the fermions are non-interacting, the

trace can be rewritten as

$$\mathrm{Tr}\left[e^{-\beta\mathcal{H}}\right] = \prod_{\nu}\left[1 + e^{-\beta\varepsilon_\nu}\right]. \tag{5.3}$$

The product in equation (5.3) is performed over all eigenenergies of the Hamiltonian. This implies that for each local update the eigenvalues of $\mathcal{H}$ must be calculated, which is numerically expensive. In addition, the expression $e^{-\beta\varepsilon_\nu}$ is numerically unstable for large β. If one of the factors diverges, which could happen if the energy is negative and $\beta > 10^3$, then $\mathcal{W}$ diverges, which is unphysical and a problem for calculating the acceptance ratio.
A possible solution for the divergence is to work with the logarithmic Boltzmann weight $\ln \mathcal{W}$. Using the logarithmic expression, the Boltzmann weight reads

$$\ln \mathcal{W} = \ln\left\{\prod_{\nu}\left[1 + e^{-\beta\varepsilon_\nu}\right]\right\} = \sum_{\nu}\ln\left[1 + e^{-\beta\varepsilon_\nu}\right]. \tag{5.4}$$

This expression still has problems if $\varepsilon_\nu < 0$ and $\beta > 10^3$. This can be fixed if an approximation is used for $\ln(1 + e^x)$. If x is large, $1 + e^x \approx e^x$ and if x is very negative, $1 + e^x \approx 1$. If x is $\mathcal{O}(1)$, the exact expression must be used. The behaviour of this expression can be captured using a piecewise function

$$e(x) = \ln(1 + e^x) \approx \begin{cases} x, & x > \theta \\ 0, & x < -\theta \\ \ln(1 + e^x), & \text{otherwise} \end{cases}, \tag{5.5}$$

where θ is a cutoff parameter which must be chosen to minimize the error. In the implementation a value of $\theta = 50$ was chosen. If this expression is used during the calculation of $\ln \mathcal{W}$, the logarithmic Boltzmann weight is numerically stable. In addition to the function $e(x)$, the derivative $e'(x)$ must also be stabilized, since it

is used for backpropagation [27]. The stabilized derivative is given by

$$e'(x) = \frac{1}{1+e^{-x}} \approx \begin{cases} 1, & x > \theta \\ 0, & x < -\theta \\ \frac{1}{1+e^{-x}}, & \text{otherwise} \end{cases} . \tag{5.6}$$

As described in chapter 3, the acceptance ratio of the newly generated configuration is calculated using the ratio of Boltzmann weights, where $\mathcal{W}(\mathfrak{x}) = p_{eq}(\mathfrak{x})$. To prevent redundant exponentiation of $\ln \mathcal{W}$ and reduce the thus introduced errors, the logarithm of the acceptance ratio can be calculated. Using local updates, the acceptance ratio is

$$A(\mathfrak{x}'|\mathfrak{x}) = \min\left(1, \frac{\mathcal{W}(\mathfrak{x}')}{\mathcal{W}(\mathfrak{x})}\right), \tag{5.7}$$

and upon applying the logarithm to the equation, the expression reads

$$\ln A(\mathfrak{x}'|\mathfrak{x}) = \min\left(0, \ln \mathcal{W}(\mathfrak{x}') - \ln \mathcal{W}(\mathfrak{x})\right). \tag{5.8}$$

In addition to the acceptance ratio, the uniformly distributed random number $\alpha \in [0, 1)$ must also be logarithmized. Thus, the new configuration is accepted if $\ln \alpha < \ln A(\mathfrak{x}'|\mathfrak{x})$.
For SLMC updates, the acceptance ratio using the logarithmized Boltzmann weight reads

$$\ln A(\mathfrak{x}'|\mathfrak{x}) = \min\left(0, \ln \mathcal{W}(\mathfrak{x}') - \ln \mathcal{W}(\mathfrak{x}) + \ln \mathcal{W}_{\text{eff}}(\mathfrak{x}) - \ln \mathcal{W}_{\text{eff}}(\mathfrak{x}')\right). \tag{5.9}$$

Using these equations, the acceptance ratio can be calculated with a high degree of numerical stability.

5.2.2. Supervised and Unsupervised Training

Training an effective Hamiltonian is split into two parts, a supervised training stage followed by an unsupervised training procedure. All training is performed for a 6×6 lattice setup. This strikes a balance between computational complexity and complex emergent behaviour.

Supervised Training

The supervised training [26] is carried out in a similar way as described for the Ising-like plaquette model. At first, training data is generated using local updates and the exact Double Exchange Hamiltonian. Afterwards, the relative mean square error (MSE) over all training configurations

$$\mathrm{MSE}(\{\mathfrak{x}\}) = \frac{1}{N_{\mathfrak{x}}} \sum_{\{\mathfrak{x}\}} \left[\frac{\ln \mathcal{W}(\mathfrak{x}) - \ln \mathcal{W}_{\mathrm{eff}}(\mathfrak{x})}{\ln \mathcal{W}(\mathfrak{x})} \right]^2 \tag{5.10}$$

is minimized. $N_{\mathfrak{x}}$ is the number of training configurations, $\mathfrak{x}$ is one set of spins $\{\boldsymbol{S}\}$ and $\mathcal{H}_{\mathrm{eff}} \to \ln \mathcal{W}_{\mathrm{eff}}$ is the effective Hamiltonian which should be trained. The parameters chosen for training were $t = 1$, $J = -1$, $\mu = 0$ and $\beta = 20$.

Unsupervised Training

Following the supervised phase, the effective Hamiltonians undergo unsupervised refinement [28] using the Self Learning Monte Carlo method. In this phase, the model actively participates in generating new spin configurations through Markov chains and is continuously updated at each reweighting step. This iterative process allows the model to adapt to configurations it may not have encountered during initial training, improving its accuracy and generalization. By leveraging feedback from SLMC acceptance ratios, the model parameters are fine-tuned to better approximate the exact Boltzmann distribution, particularly in regions of configuration space that are critical for capturing the physical behaviour of the Double Exchange model.

5.2.3. Training Results

An overview of the trained models and their corresponding MSE is given in table 5.1. The number in brackets indicates the number of layers used to generate the effective spin field $\{\boldsymbol{S}^{\mathrm{eff}}\}$. The linear model – which corresponds to a transformer-like model with 0 attention layers – was trained using nearest neighbour interactions of order 6. Similarly, for the local operators $\boldsymbol{S}^{\hat{\alpha}}$, $n_2 = 6$ was used. As can be seen from the table, the MSE of the linear model as well as all transformer models

is approximately equal. This indicates that the attention layers do not drastically improve the model's performance.

Model	**MSE** $[10^{-7}]$	**Parameter Count** c
Linear	4.747	7
Transformer (1)	4.468	25
Transformer (2)	4.431	43
Transformer (3)	4.571	61
Transformer (4)	4.638	79
Transformer (5)	4.435	97
Transformer (6)	4.712	115
Transformer (7)	4.699	133
Transformer (8)	4.591	151
Transformer (9)	4.455	169
Fully Connected	2.259	1297
Convolutional (0)	39.433	11
Convolutional (1)	497.183	47
Convolutional (2)	156.494	147
Convolutional (3)	177.245	343

Table 5.1.: *Mean squared error for different model architectures, scaled by* 10^{-7} *and the model size.*

A plot of the true value $\ln \mathcal{W}$ over the predicted value $\ln \mathcal{W}_{\text{eff}}$ is given in figure 5.1 for the linear model and in figure 5.2 for the transformer models. Compared to the models including only nearest neighbour interactions, the relative MSE of the fully connected effective Hamiltonian improves by a factor of ~ 2. A plot comparing the true and predicted values is given in figure 5.3.

In contrast to the linear, transformer-like and fully connected effective Hamiltonian, the convolution-like effective Hamiltonian struggles with reproducing the behaviour of the exact Hamiltonian. While the linear convolution network – corresponding to $\mathfrak{L} = 0$ – manages to reproduce the behaviour approximately, the effective models involving at least one convolution and convolutional attention layer (4.15 & 4.17) struggle to reproduce the results. The models involving $\mathfrak{L} \geq 2$ display a hard cutoff for $\ln \mathcal{W}_{\text{eff}}$ and struggle to converge. Consequently, the relative MSE is $1 - 2$ orders of magnitude higher than for the previously discussed

models.

The source for this discrepancy could have different reasons. Firstly, the chosen Hamiltonian could be designed poorly, resulting in the effective Hamiltonian not being able to capture the correct behaviour. Secondly, there could be a problem with the convolutions (4.15), from which the discrepancy arises. Lastly, the model could have lacked training time or training data, which could be fixed with longer training cycles or more diverse training data.

As part of this work, it was decided to focus on the effective Hamiltonians which have already been shown to capture the behaviour of the exact model. However, future work should investigate the convolution-like models further.

A plot of the training result for the linear convolution-like effective Hamiltonian is given in figure 5.4. It can be observed that the distribution of datapoints is significantly wider than for the more effective models. The plots for the convolution-like models with $1-3$ layers is given in figure 5.5. In addition to the whole data range, a zoomed in view to the relevant area is provided.

In conclusion, the linear, fully connected, and transformer-like effective Hamiltonians can be trained to reproduce the behaviour of the exact Hamiltonian, while the convolution-like Hamiltonians fail to do so.

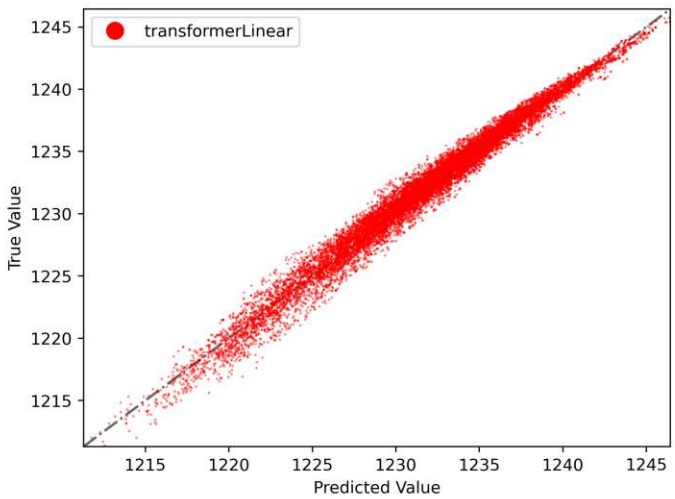


Figure 5.1.: *A plot of the true value over the predicted value for the linear effective Hamiltonian.*

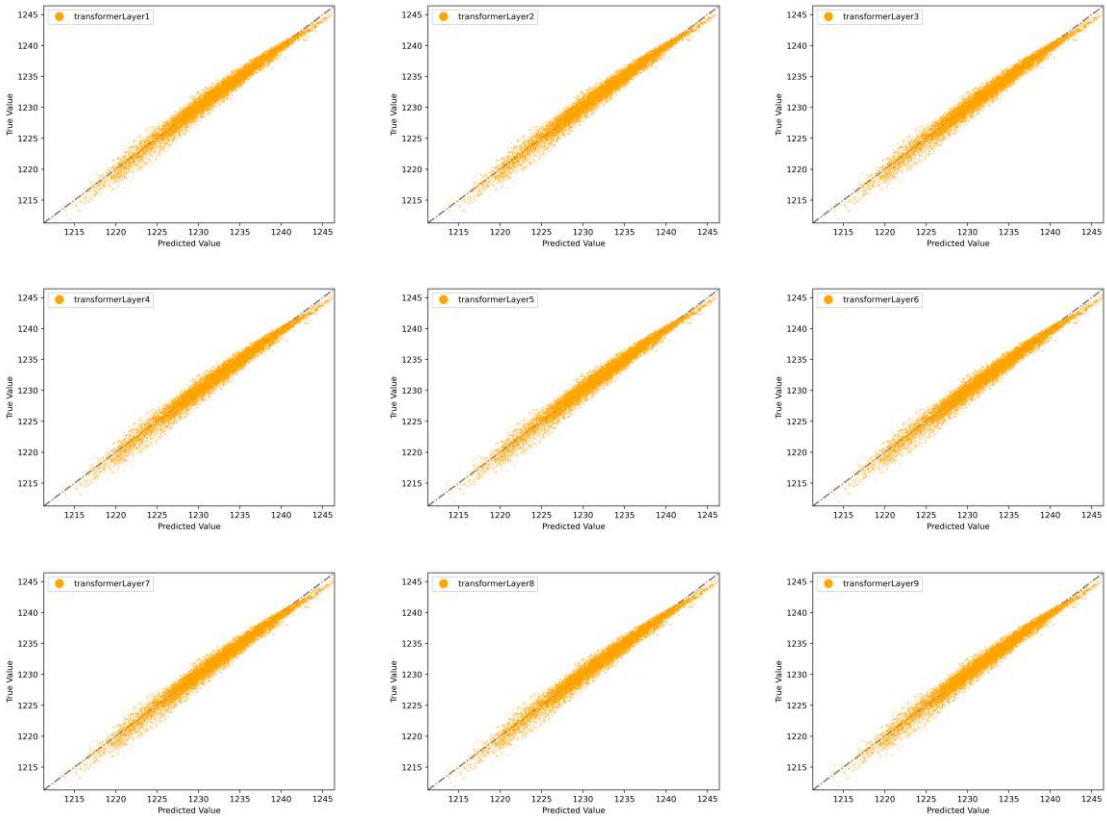


Figure 5.2.: *A plot of the true value over the predicted value for the transformer-like effective Hamiltonian with 1-9 attention layers.*

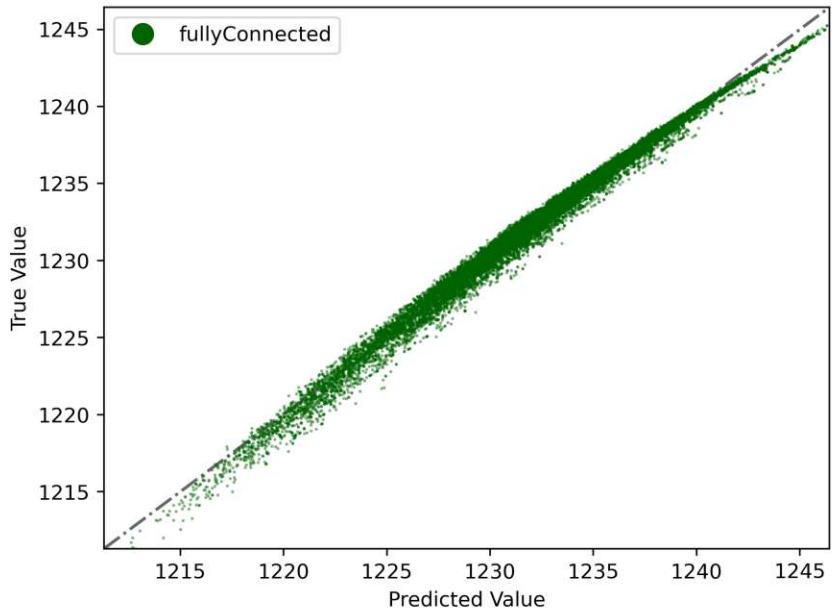


Figure 5.3.: *A plot of the true value over the predicted value for the fully connected effective Hamiltonian.*

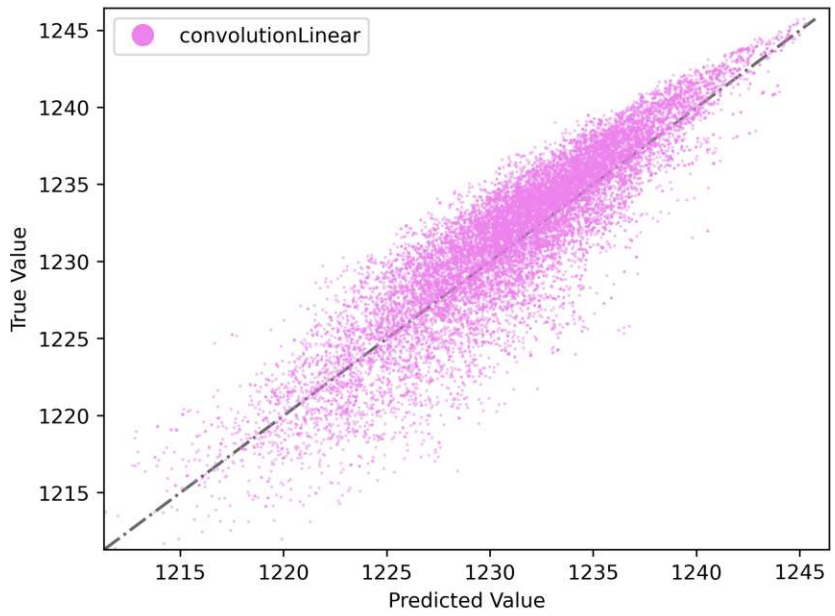


Figure 5.4.: *A plot of the true value over the predicted value for the linear convolution-like effective Hamiltonian.*

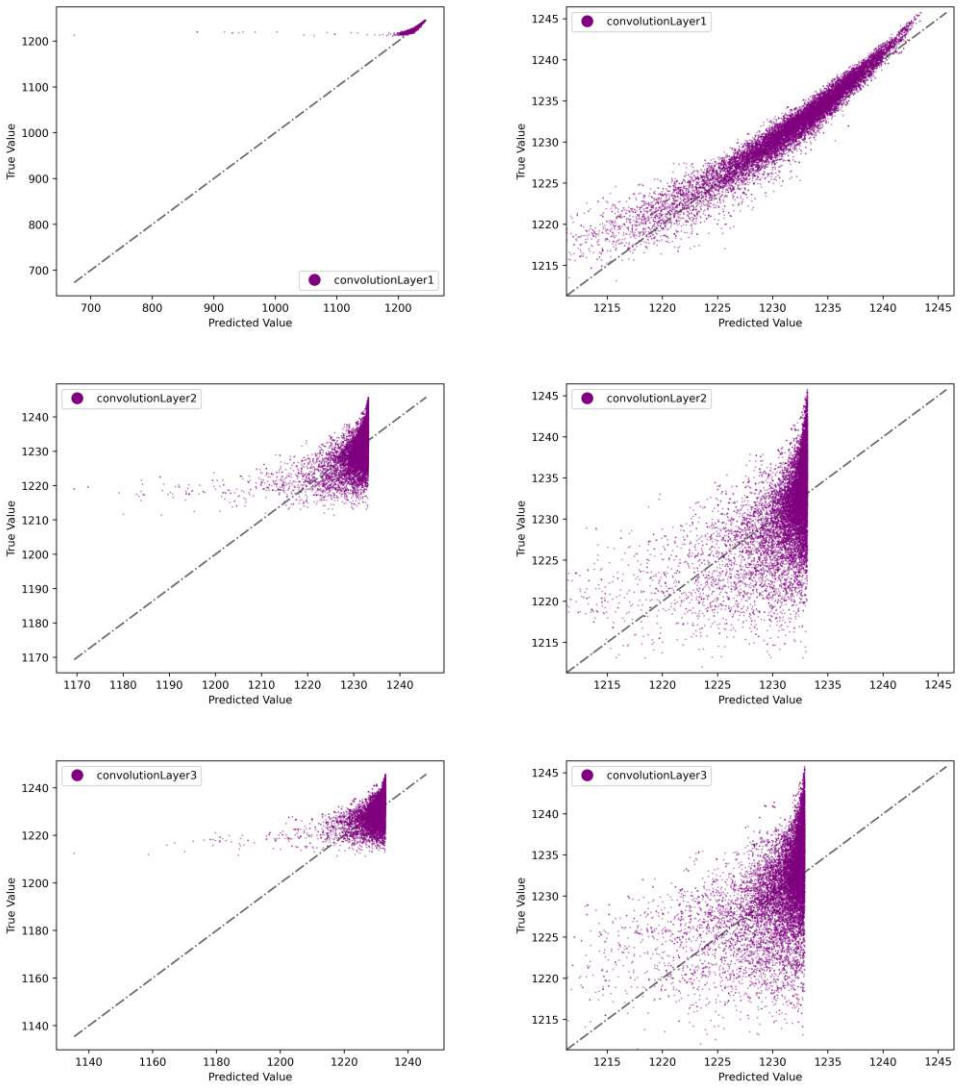


Figure 5.5.: *A plot of the true value over the predicted value for the convolution-like effective Hamiltonian with a layer count of 1-3.*

6. Comparison of Update Methods and Effective Hamiltonians

Having established the relevant Hamiltonians, Monte Carlo techniques and machine learning aspects, this chapter presents a systematic comparison of the different update strategies. Special attention is given to the applicability of the SLMC approach, particularly to the Double Exchange model. The results highlight both the strengths and limitations of the considered update methods, providing a foundation for future work.

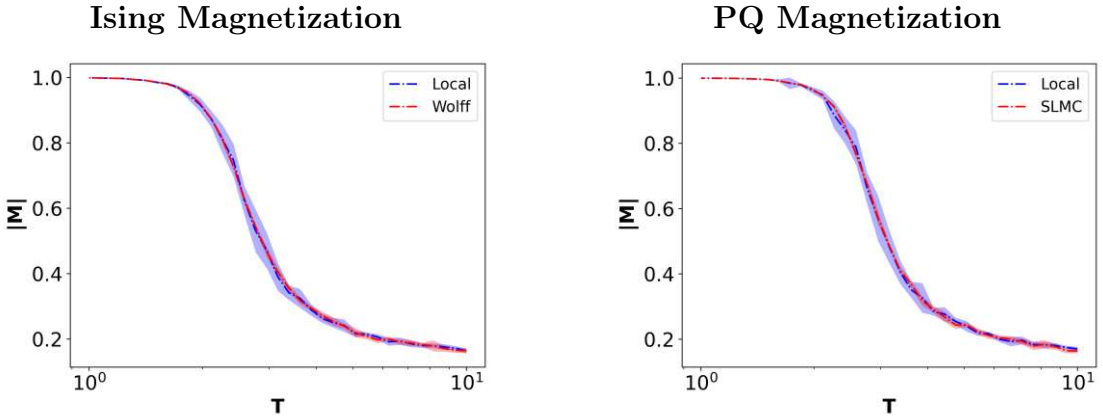
6.1. Ising Model

The Ising model was mainly considered as a preparation for the Double Exchange model, since Wolff updates are designed to work with Ising-like models. For the Ising model, local- and Wolff updates yield the same results for the magnetization and staggered magnetization over a large temperature range (figures 6.1a & 6.2a). This indicates that the Wolff updates were implemented correctly and can be used as an efficient update method for the Self Learning Monte Carlo method.
It is notable that the acceptance ratio for the Wolff updates (figure 6.3a) is equivalent to one over the whole temperature range. This is expected, since the proposed configuration after the cluster growth is always accepted. The opaquely filled region in the plots indicates the standard deviation of the observable.

6.2. Ising-like Plaquette Model

As in the Ising model, multiple Markov chains were calculated and the results aggregated at the end. From figure 6.2b it is obvious that the simulation has not yet fully reached the equilibrium, however, for the low temperature regime and the phase transition, the convergence was sufficient.

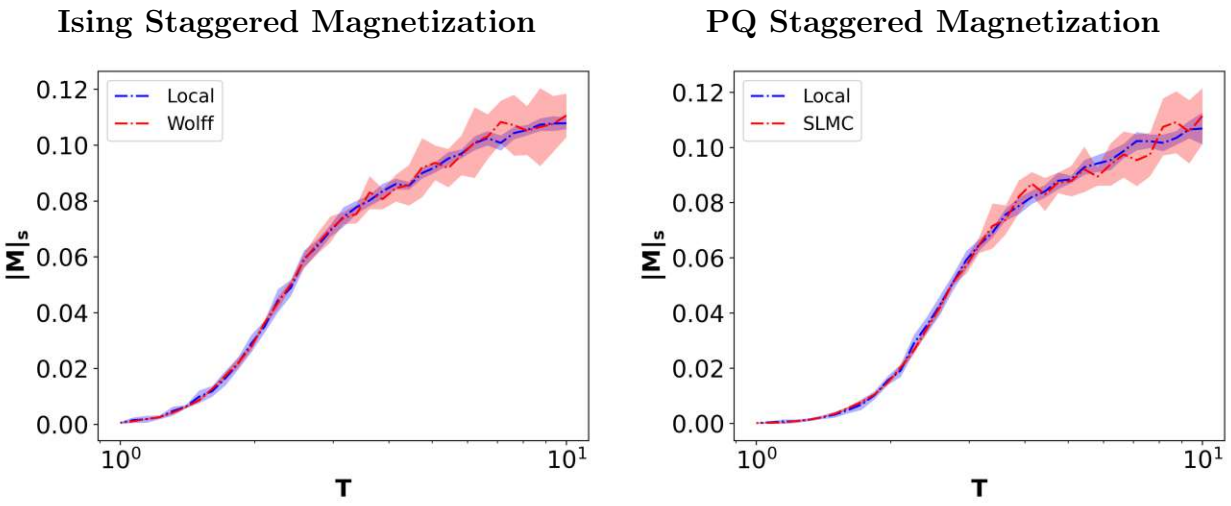
A comparison of the magnetization and staggered magnetization (figures 6.1b & 6.2b) shows that the two update methods, local updates and Self Learning Monte Carlo updates, yield the same observable over a large temperature range. For the Self Learning Monte Carlo updates, the Hamiltonian described in section 5.1 was used. This behaviour demonstrates that a simple effective model can be used to simulate the Ising-like plaquette model (PQ) within the framework of the Self Learning Monte Carlo method. Since the results match with [3] it can be deduced that the algorithm was implemented correctly and can be extended to more complex models. In addition to the observables, the acceptance ratio (figure 6.3b) shows a rather interesting behaviour. In comparison to local updates, the acceptance ratio for the SLMC updates is significantly higher across the whole temperature range. Furthermore, in the low and high temperature regions, the acceptance ratio tends towards one. This indicates that in these regions, the two models – Ising-like plaquette model and Ising model – show the same behaviour.



(a) *Comparison of local and Wolff updates.*

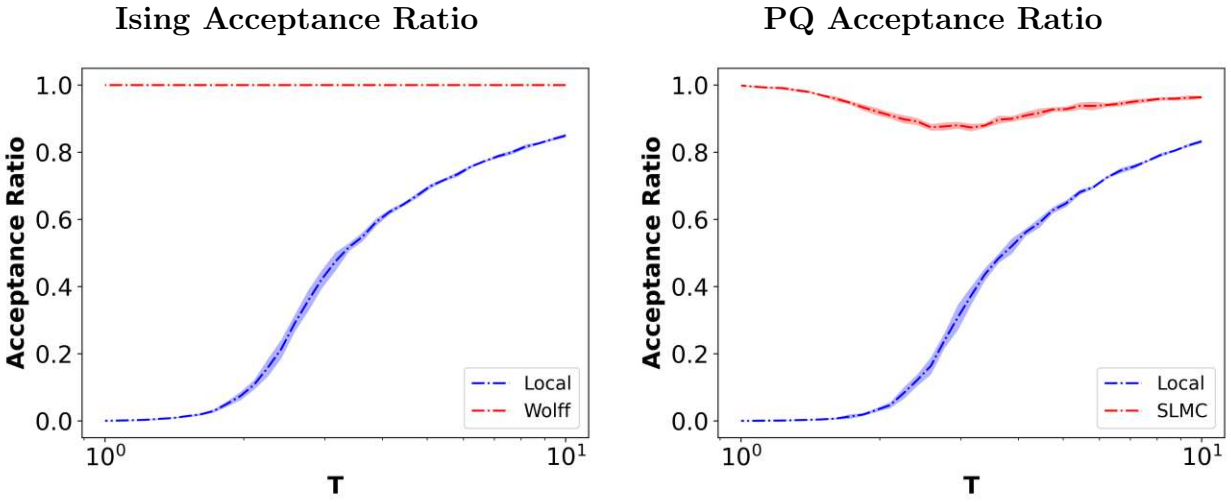
(b) *Comparison of local and SLMC updates.*

Figure 6.1.: *Magnetization for the Ising model and the Ising-like plaquette model.*



(a) Comparison of local and Wolff updates. *(b) Comparison of local and SLMC updates.*

Figure 6.2.: *Staggered magnetization for the Ising model and the Ising-like plaquette model.*



(a) Comparison of local and Wolff updates. *(b) Comparison of local and SLMC updates.*

Figure 6.3.: *Acceptance ratio for the Ising model and the Ising-like plaquette model.*

6.3. Double Exchange Model

Having discussed the results for the Ising and Ising-like plaquette model, the results for the Double Exchange model will be presented. The main focus will be on the correctness of the observables calculated by the effective Hamiltonians, the correlation length of the different effective Hamiltonians, as well as the behaviour of selected effective Hamiltonians in dependency of the lattice size.

6.3.1. Observables

As discussed in section 3.4, the accept/reject step in the Self Learning Monte Carlo method

$$A(x'|x) = \min\left(1, \frac{p_{eq}(x')}{p_{eq}(x)} \frac{p_{eq,\,\mathrm{eff}}(x)}{p_{eq,\,\mathrm{eff}}(x')}\right) \tag{6.1}$$

should be able to correct faulty distributions of the effective Hamiltonian. For Markov chains of infinite length, any effective Hamiltonian should in theory reproduce the results of the exact Hamiltonian. In practice, however, the probability distribution generated by the effective Hamiltonian and the exact Hamiltonian should be sufficiently similar. To test the performance of the effective Hamiltonians compared to the exact simulation, 7 Markov chains of length 5000 were generated for each temperature. From the low standard deviation in the figures 6.4, 6.5 and 6.6 it can be deduced that the simulation reached the equilibrium and all Markov chains produced a similar result.

From the figures 6.4a and 6.5a it can be seen that the transformer-like and fully connected effective Hamiltonians can reproduce the exact behaviour of the Double Exchange model to a high degree of agreement. The convolution-like effective Hamiltonian with 0 convolution or attention layer also reproduces the result of the exact Hamiltonian, however, the discrepancy to the exact Hamiltonian is larger compared to the other effective Hamiltonians, especially near the phase transition.

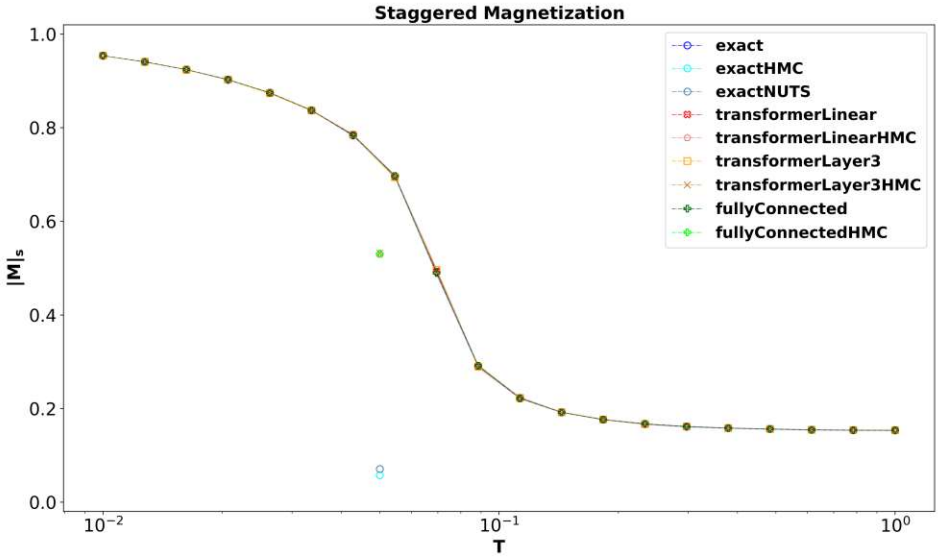
The convolution-like effective Hamiltonian with 3 convolution and attention layers fails to accurately reproduce the behaviour of the exact Hamiltonian, as can be seen in the figures 6.4b and 6.5b.

This model behaviour reflects the relative MSE values given in table 5.1, where the relative MSE values for the transformer-like, and fully connected effective Hamiltonians were approximately equal, whilst the values for the convolutional models were $1-2$ orders of magnitude higher.

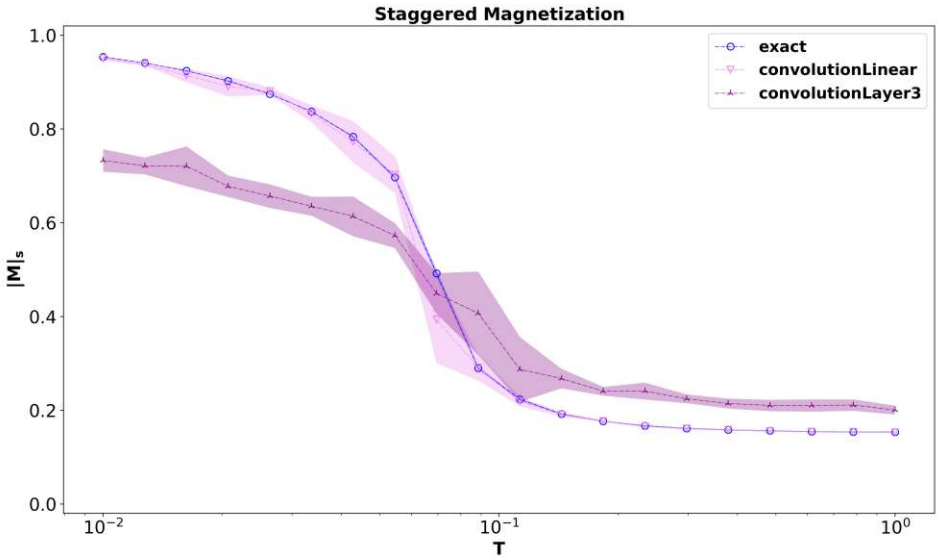
In addition to the whole temperature range, simulations at $\beta = 20$ – which roughly corresponds to the critical point – were performed. Simulations at this temperature were performed using local-, SLMC-, as well as, HMC-updates. As displayed in the figures 6.4a and 6.5a, the HMC-like update methods – HMC and NUTS – fail to explore the phase space in a representative way. Furthermore, due to the model's nature of needing to diagonalize the exact Hamiltonian for each update step, the computational complexity for the HMC updates and NUTS updates is higher compared to local updates. The SLMC updates did not manage to correct the faulty configuration distribution and thus, the observables generated using a Hybrid Self Learning Monte Carlo update do not agree with the exact result. Thus, HMC in the here presented and implemented way is not suitable for simulating the Double Exchange model. It is possible that an error in the implementation causes these discrepancies; however, due to the computational complexity the advantages of HMC are diminishingly small.

The acceptance ratio given in the figure 6.6 shows an interesting behaviour. For the convolution-like effective Hamiltonians the acceptance ratio is worse in comparison to the exact model (figure 6.6b), which is consistent with the models failing to reproduce the behaviour of the exact Hamiltonian.

Analysing the acceptance ratio for the transformer-like model and the fully connected effective Hamiltonian, a dip near the critical point can be observed. This indicates that the effective models can approximate the exact model in the high and low temperature regime very accurately, as indicated by the high acceptance ratio of $> 60\%$, and struggles near the phase transition. Furthermore, the performance of the fully connected effective Hamiltonian must be emphasized, as it outperforms the transformer-like effective Hamiltonian significantly near the critical point. The comparison of the transformer-like model with 0 and 3 attention layers also shows no significant difference. This indicates that the attention layers do not significantly improve the effective model.

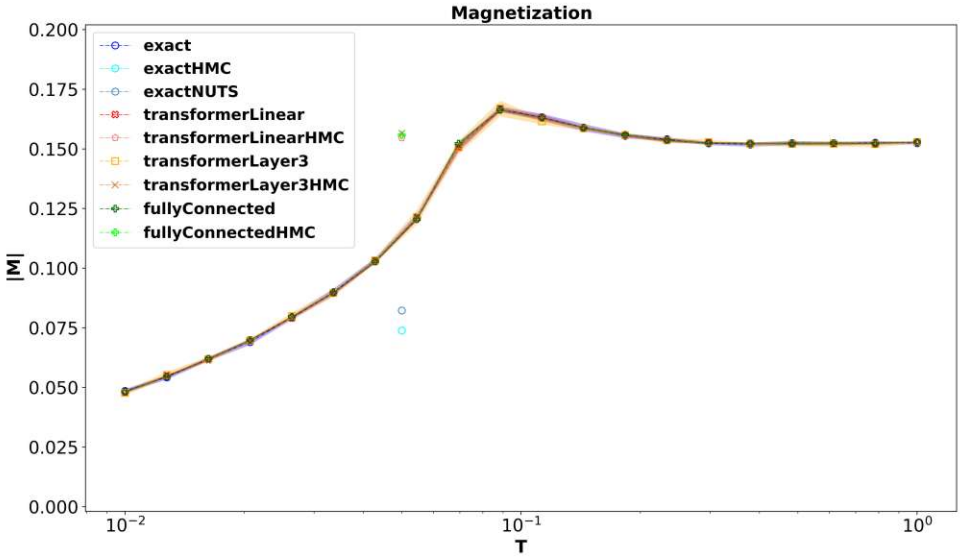


(a)

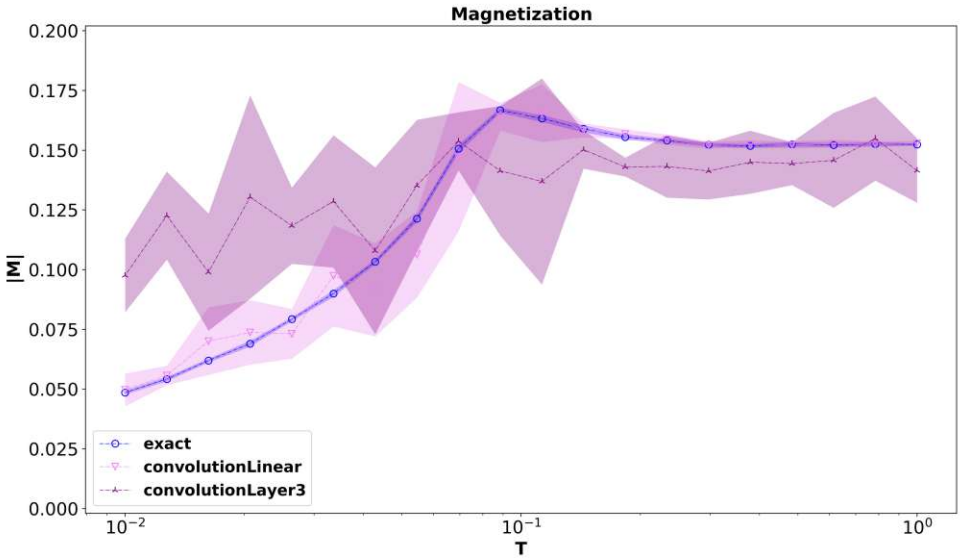


(b)

Figure 6.4.: *A plot of the staggered magnetization over a temperature range of* $T = 0.01t$ *to* $T = 1t$ *for multiple (effective) Hamiltonians.*

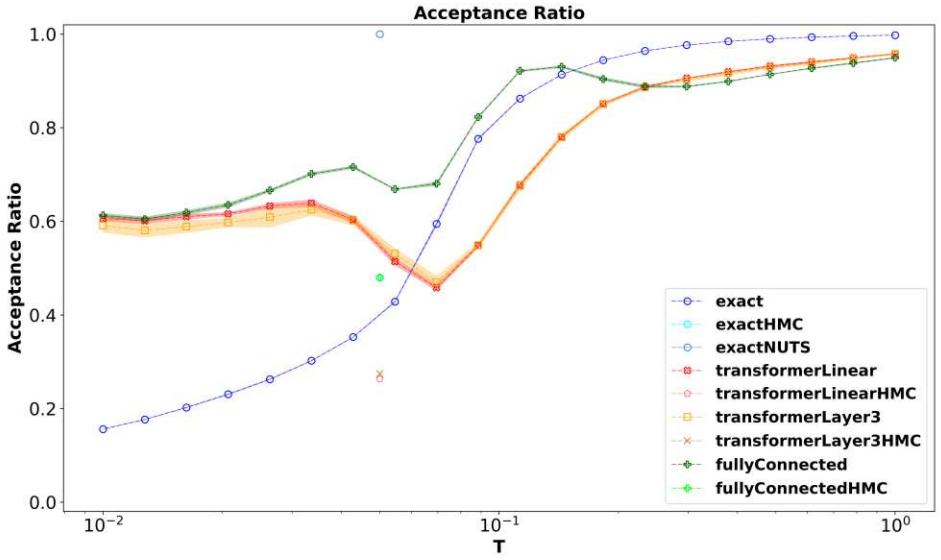


(a)

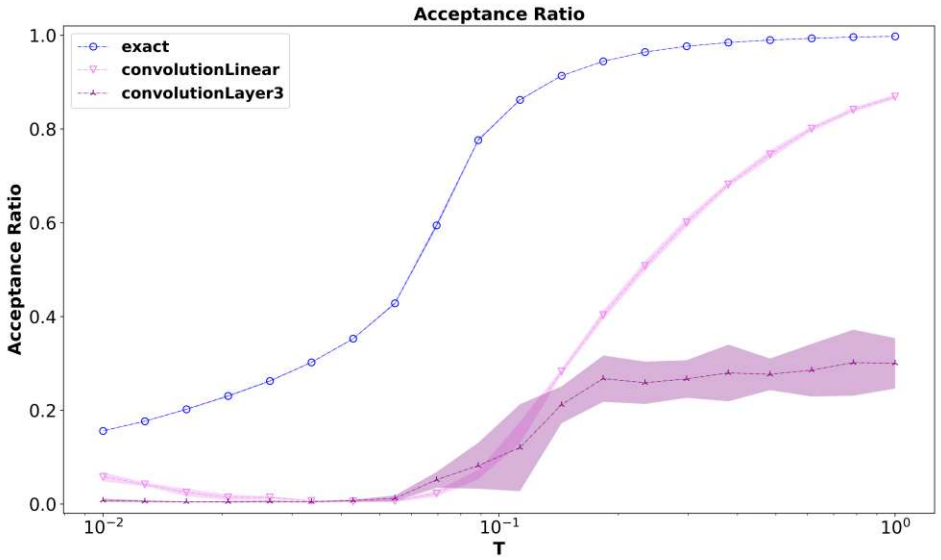


(b)

Figure 6.5.: *A plot of the magnetization over a temperature range of $T = 0.01t$ to $T = 1t$ for multiple (effective) Hamiltonians.*



(a)



(b)

Figure 6.6.: *A plot of the acceptance ratio over a temperature range of $T = 0.01t$ to $T = 1t$ for multiple (effective) Hamiltonians.*

6.3.2. Autocorrelation Length

An important quantity within the framework of Monte Carlo simulations is the autocorrelation length $\hat{\tau}$. The autocorrelation can be used to quantify how similar two consecutive configurations are. The total length ℓ of the Markov chain should be significantly larger than the autocorrelation time, $\ell \gg \hat{\tau}$. This guarantees that the number of independent samples $N = \ell/\hat{\tau}$ is large enough to sufficiently approximate the target distribution.
The autocorrelation length is calculated using

$$\hat{\tau} = 1 + 2\sum_{t=1}^{M} \hat{\rho}(t), \tag{6.2}$$

where $\hat{\rho}(t)$ is the normalized autocorrelation function given by

$$\hat{\rho}(t) = \frac{\hat{c}(t)}{\hat{c}(0)}, \qquad \hat{c}(t) = \frac{1}{N-t}\sum_{n=1}^{N-t}(x_n - \hat{\mu})(x_{n+t} - \hat{\mu}), \qquad \hat{\mu} = \frac{1}{N}\sum_{n=1}^{N} x_n \tag{6.3}$$

and $M \ll N$. If the sum would range up to N, fluctuations in the longer lags would result in a noisy estimate for $\hat{\tau}$. M is commonly chosen to fulfil $M \geq C\hat{\tau}(M)$, with constant $C \sim 5$ [29].
In table 6.1, the values of $\hat{\tau}$ for each considered (effective) Hamiltonian is given. The autocorrelation was calculated at $\beta = 20$, which roughly corresponds to the critical point. As the operator considered, the staggered magnetization $|M|_s$ was chosen. It can be observed that local updates yield a very high autocorrelation length for the exact Hamiltonian. Using a transformer-like – Linear, Transformer (i) – or fully connected effective Hamiltonian with SLMC updates, $\hat{\tau}$ is reduced by a factor of ~ 20. This indicates that Self Learning Monte Carlo updates drastically improve the simulation performance of a quantum spin system near the critical point. In agreement with the poor model performance indicated by the deviating observable values, the autocorrelation length for the convolutional Hamiltonians is approximately equal or even worse than for local updates of the exact Hamiltonian. This is an indication that the convolutional models are not suited for simulating the Double Exchange model.
An interesting observation is that the autocorrelation length for the simulations using HMC updates does not improve drastically, except for the exact Hamiltonian. This indicates that SLMC has the highest impact for reducing the correlation

length. Interestingly, the correlation length is largest for NUTS updates of the exact Hamiltonian.

The best performing model, in agreement with the acceptance ratio values, is the fully connected effective Hamiltonian. This can also be observed in the inset of figure 6.7. In figure 6.7, a comparative plot of the autocorrelations of the different Hamiltonians is given.

Model	$\hat{\tau}$ **[iterations]**
Exact	266.542
Exact (HMC)	31.511
Exact (NUTS)	613.562
Linear	11.978
Linear (HMC)	17.794
Transformer (3)	12.627
Transformer (3) (HMC)	20.354
Fully Connected	8.373
Fully Connected (HMC)	9.870
Convolutional (0)	233.784
Convolutional (3)	707.452

Table 6.1.: *The autocorrelation length of different (effective) Hamiltonians for a Markov chain of length* $N = 5000$ *and* $\beta = 20$.

6.3.3. Lattice Size Analysis

As discussed in chapter 4, not all of the considered effective Hamiltonians are a priori lattice size independent. By design, only the transformer- and convolution-like effective Hamiltonians can be used to simulate lattice sizes other than the size they are trained on. For the transformer-like effective Hamiltonian with 0 and 3 attention layers simulations for different lattice sizes and temperatures were performed.

The results for $\beta = 20$ and $N = 4$ up to $N = 15$ are given in the figures 6.9a, 6.10a and 6.11a. A remarkable result is that the staggered magnetization shows a different behaviour in even lattice sizes compared to uneven ones. For uneven lattice sizes, frustration effects significantly reduce the alignment of spins and thus reduce

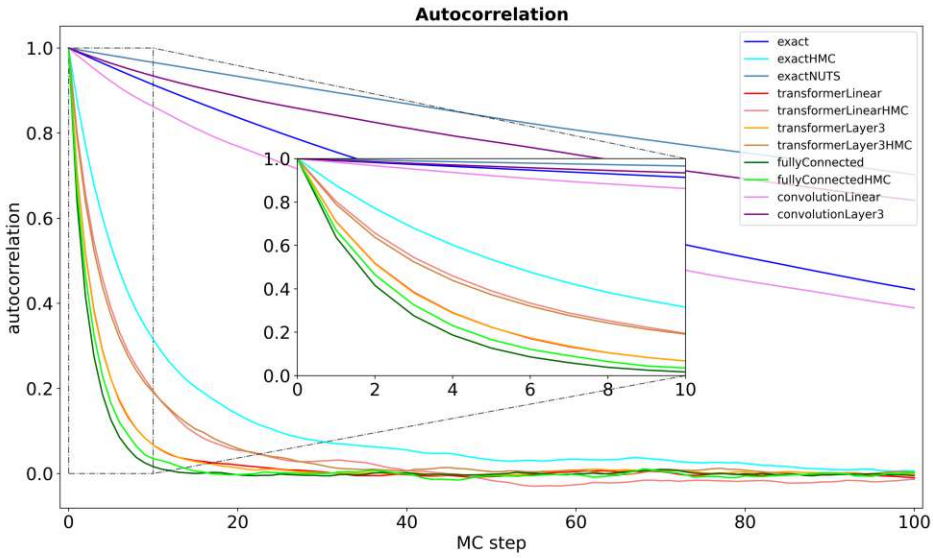


Figure 6.7.: *The autocorrelation for different (effective) Hamiltonians and update methods for* $\beta = 20$.

the staggered magnetization (figure 6.9a). The magnetization shows a similar behaviour, however, the effect is less pronounced (figure 6.10a). The acceptance ratio also fluctuates with the lattice size. For uneven lattice sizes, the exact simulation using local updates provides higher acceptance ratios than SLMC updates. The reason could again be the frustration effects, which hinder the spin order. If a frustrated spin is updated, the energy of the configuration hardly changes and the proposed configuration will be accepted with high probability. This would also explain the sinking acceptance ratio with growing N (figure 6.11a).

A similar behaviour can be observed for $\beta = 200$, although the acceptance ratio for the SLMC updates is consistently $\sim 50\%$ higher than for local updates (figure 6.11b). This is explainable by the fact that the system is in an ordered phase for $\beta = 200$ and thus, local updates significantly impact the energy of the system. The discrepancy in the magnetization (figure 6.10b) can be explained by poor statistics, indicated by the low acceptance ratio. The staggered magnetization again is different for even and uneven lattice sizes N (figure 6.9b).

In general it was demonstrated that the transformer-like effective Hamiltonians

can be used to simulate configurations of arbitrary size, even though training was performed only for one specific lattice size.

Lattice Size Independent Fully Connected Effective Hamiltonian

For the fully connected effective Hamiltonian an adaptation is proposed to restore the lattice size independence.
As described previously, $\mathcal{H}_{\text{eff}}^{\mathfrak{F}}$ is trained for the lattice size $\mathbb{L} = N \times N$. The trained coupling matrix is $\hat{\mathcal{J}}_{\mathbb{L}}$. To use the trained Hamiltonian for a different lattice size $\mathbb{L}' = N' \times N'$, the original coupling matrix $\hat{\mathcal{J}}_{\mathbb{L}}$ gets tiled, cropped and rescaled, to obtain the new coupling matrix $\hat{\mathcal{J}}_{\mathbb{L}'}$. Rescaling by N/N' aims to keep the trace at a similar order of magnitude. The algorithm for this tiling is given in algorithm 2. Once the new effective coupling matrix is generated, it can be used to simulate lattices of size $\mathbb{L}'$.
The results for $\beta = 20$ are given in the figures 6.9a, 6.10a and 6.11a. Although the observables can be reproduced correctly for lattice sizes smaller than 8, the acceptance ratio is very low for lattice sizes N' and thus, the correlation length is rather long. Consequently, the simulation would need a very long Markov chain to reach the equilibrium and thus, the fully connected effective Hamiltonian is not suitable for this task. For simulating different lattice sizes than the training lattice size, the transformer-like effective Hamiltonians are more suitable than the adapted fully connected effective Hamiltonian. A comparison of the autocorrelation for $N = 4$, $N = 6$ and $N = 10$ of the staggered magnetization for the transformer-like effective Hamiltonian with 3 attention layers and the fully connected effective Hamiltonian is given in figure 6.8.

Algorithm 2 *The tiling algorithm to generate a coupling matrix for arbitrary lattice sizes.*

function TILECOUPLINGMATRIX($\hat{\mathcal{J}}_{\mathbb{L}}$, N')
  $N \leftarrow$ number of rows in $\hat{\mathcal{J}}_{\mathbb{L}}$
  $f = \frac{N}{N'}$
  if $N' < N$ **then**
    $s = \lfloor \frac{N-N'}{2} \rfloor$
    $\hat{\mathcal{J}}_{\mathbb{L}'} = f \cdot \hat{\mathcal{J}}_{\mathbb{L}}[s : s+N', s : s+N']$
  else
    num_tiles $= 2 \cdot \lfloor \frac{N'}{N} \rfloor + 1$
    tiled $= \hat{\mathcal{J}}_{\mathbb{L}}$.repeat(num_tiles, num_tiles)
    $M =$ num_tiles $\cdot N$
    $s = \lfloor \frac{M-N'}{2} \rfloor$
    $\hat{\mathcal{J}}_{\mathbb{L}'} = f \cdot$ tiled$[s : s+N', s : s+N']$
  end if
  return $\hat{\mathcal{J}}_{\mathbb{L}'}$
end function

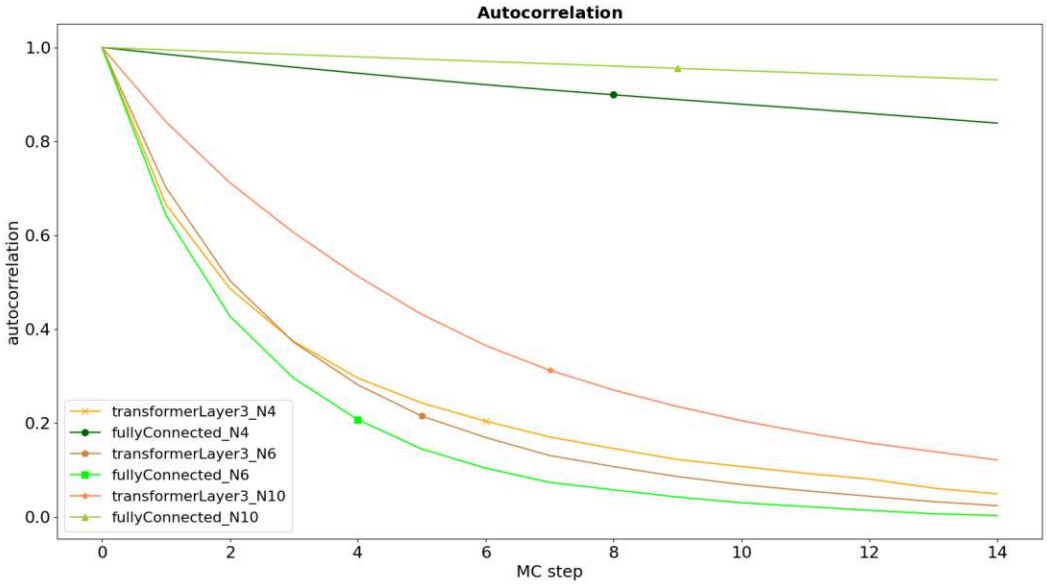
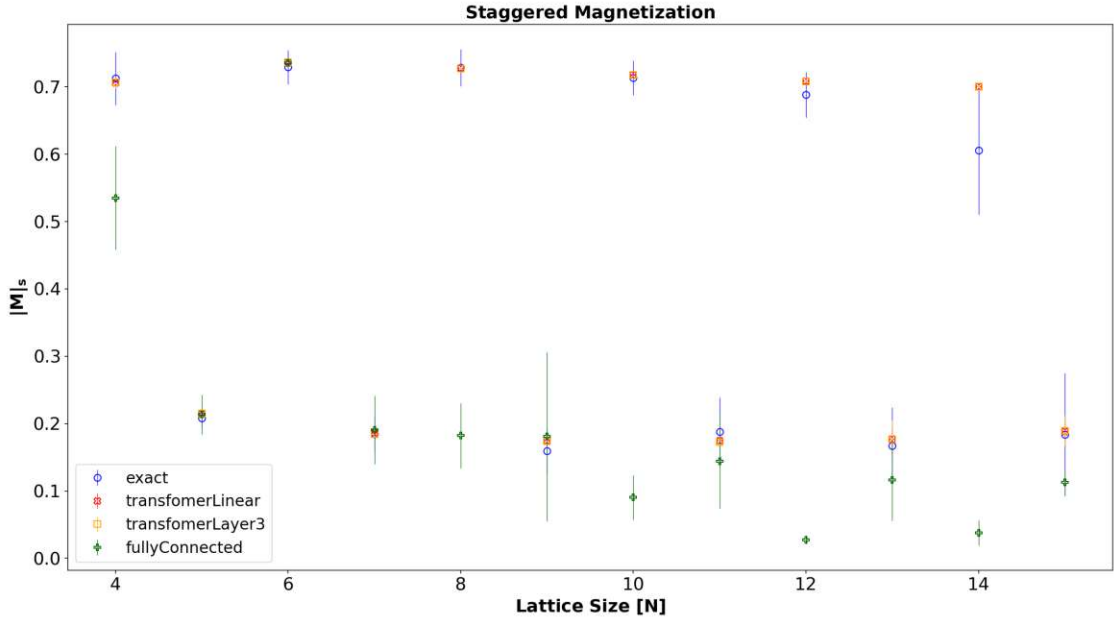
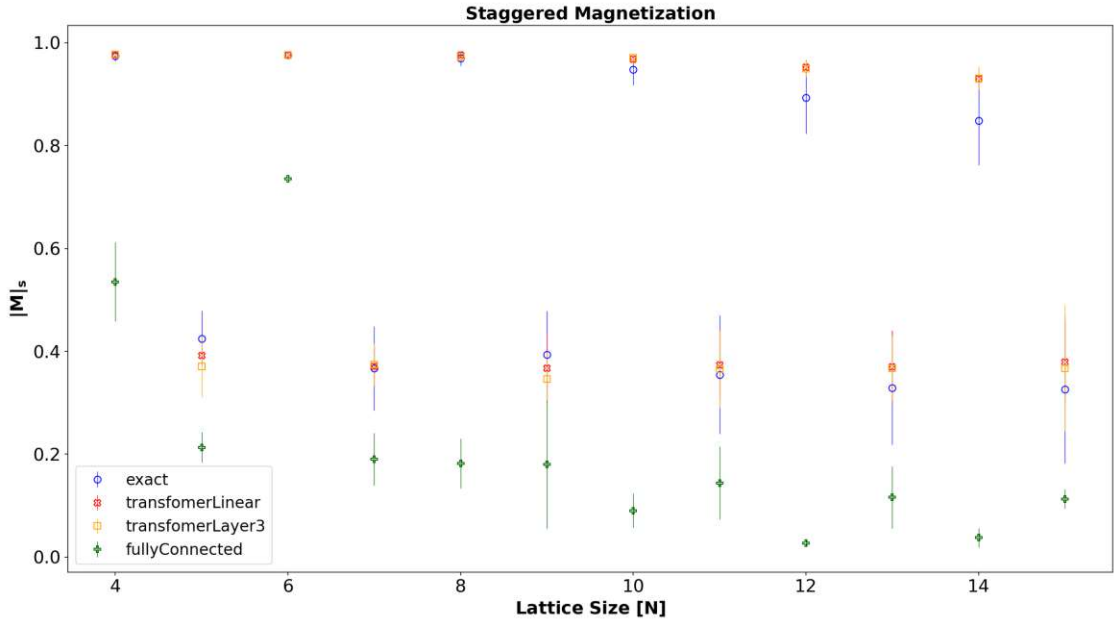


Figure 6.8.: *Autocorrelation for the transformer-like effective Hamiltonian with 3 attention layers and the fully connected effective Hamiltonian for lattice sizes $N = 4, 6, 10$.*

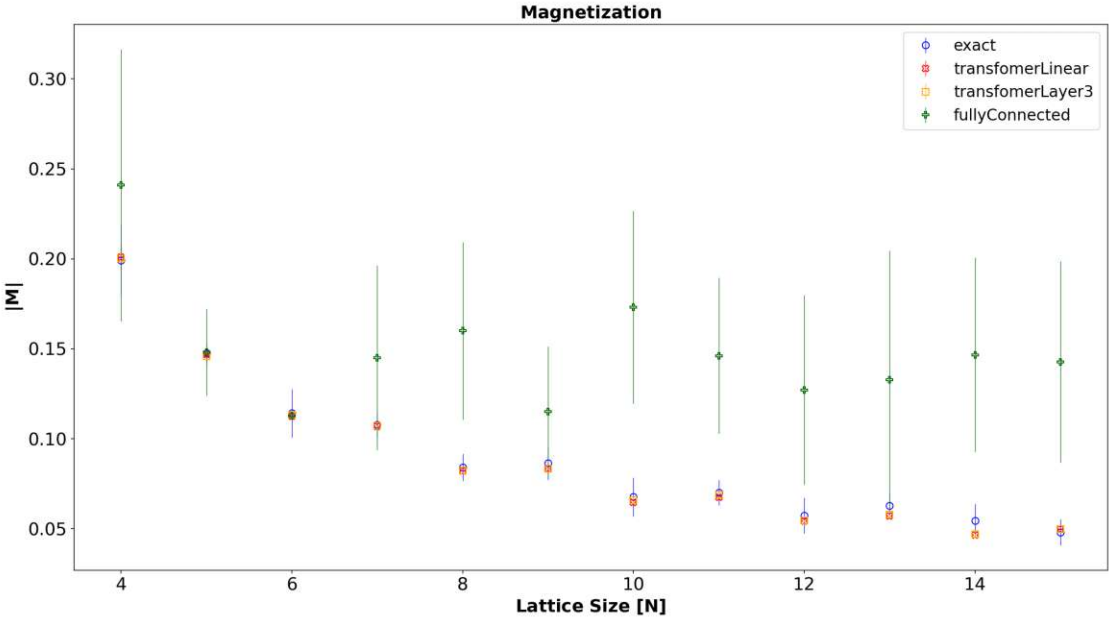


(a)

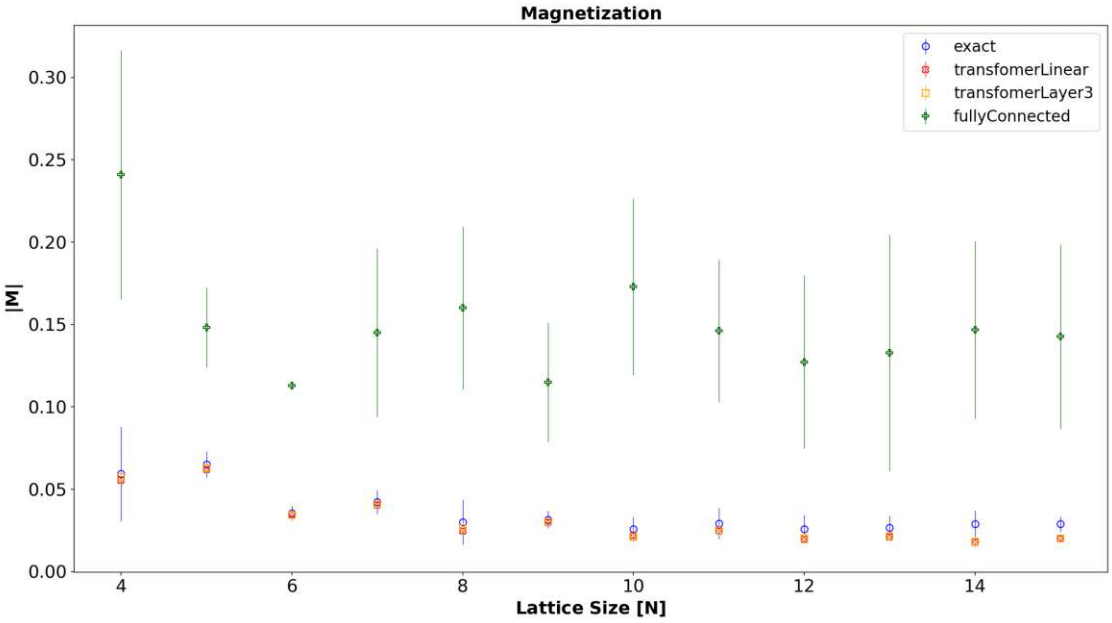


(b)

Figure 6.9.: *The staggered magnetization for $\beta = 20$* ***(a)*** *and $\beta = 200$* ***(b)*** *for different lattice sizes ranging from $N = 4$ to $N = 15$.*

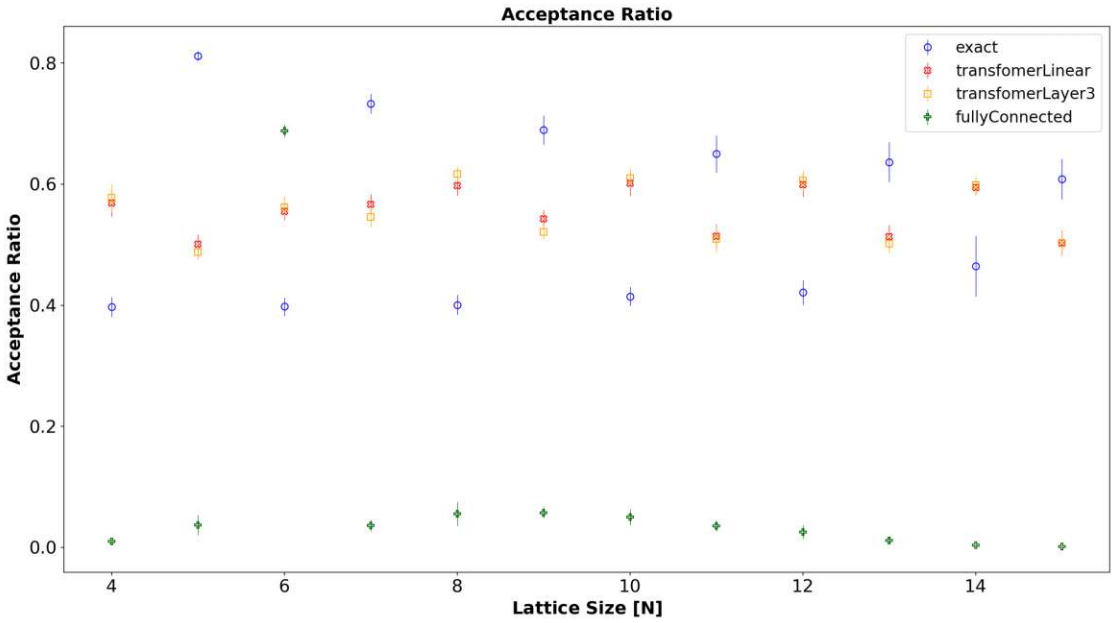


(a)

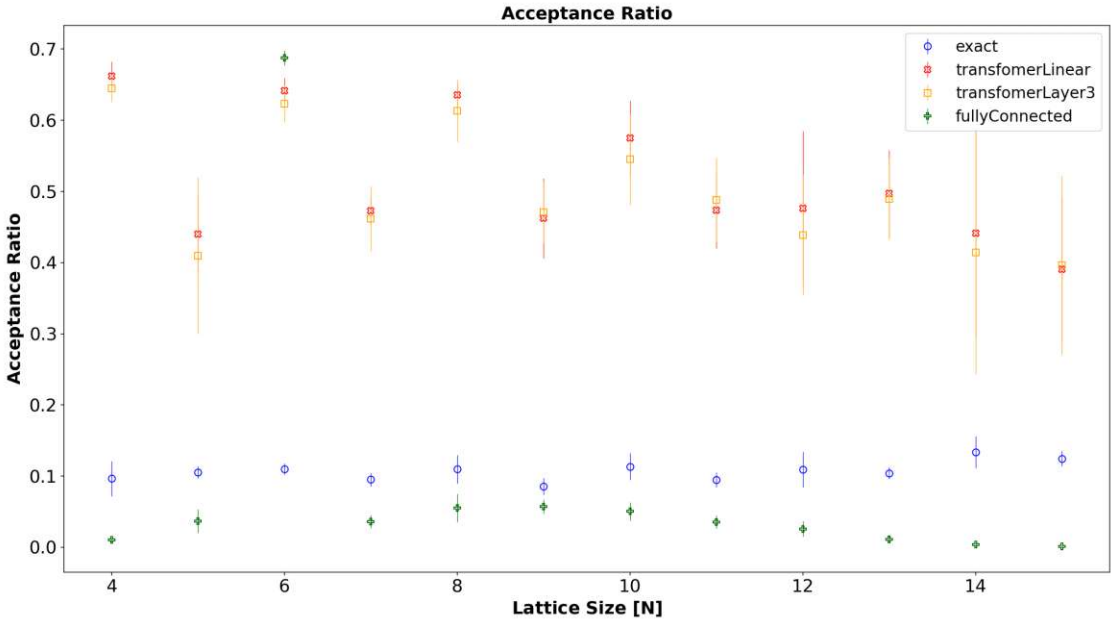


(b)

Figure 6.10.: *The magnetization for $\beta = 20$* ***(a)*** *and $\beta = 200$* ***(b)*** *for different lattice sizes ranging from $N = 4$ to $N = 15$.*



(a)



(b)

Figure 6.11.: *The acceptance ratio for $\beta = 20$* ***(a)*** *and $\beta = 200$* ***(b)*** *for different lattice sizes ranging from $N = 4$ to $N = 15$.*

7. Conclusion and Outlook

In this thesis, a comprehensive exploration of various Monte Carlo simulation techniques was undertaken, with a focus on the development and evaluation of the Self Learning Monte Carlo (SLMC) method applied to complex spin systems. Three models were considered in increasing order of complexity: the Ising model, the Ising-like plaquette model, and the Double Exchange model. Through this progression, both classical and machine learning-inspired update strategies were systematically compared, enabling a nuanced understanding of their respective strengths and limitations.

For the Ising and plaquette models, local and cluster update methods – particularly the Wolff algorithm – were validated and shown to produce consistent thermodynamic observables across a wide range of temperatures. The implementation of SLMC updates yielded a significant improvement in sampling efficiency, as evidenced by higher acceptance ratios and reduced autocorrelation lengths, especially near the critical region where local updates typically suffer from critical slowing down. These results laid the foundation for applying SLMC to more intricate models.

The main focus of the thesis was the application of SLMC techniques to the Double Exchange model, which represents a challenging system due to the interplay of classical spin degrees of freedom and quantum mechanical fermions. Here, multiple effective Hamiltonians were designed using neural network architectures of varying complexity, including linear, transformer-based, fully connected, and convolution-like models. The models were trained through a combination of supervised and unsupervised learning procedures. Training stability was achieved by introducing numerically robust formulations of the logarithmic Boltzmann weight and its derivatives, thereby enabling stable backpropagation even at low temperatures.

The training results revealed that transformer-based and fully connected effective Hamiltonians achieved the lowest relative mean squared error (MSE), successfully approximating the exact Boltzmann distribution. Interestingly, the addition of

multiple attention layers in the transformer architectures did not yield a significant performance improvement, suggesting a redundancy in expressivity for the considered lattice size. The fully connected Hamiltonian, although computationally more expensive and lacking lattice-size generalizability, outperformed all other models near the critical point. In contrast, convolution-like Hamiltonians consistently underperformed both in terms of MSE and in reproducing thermodynamic observables, likely due to their limited symmetry properties and poor numerical conditioning.

If lattice size independence is not needed for the analysis of a system, the fully connected effective Hamiltonian proved to outperform more carefully constructed models. This is rather surprising since an interaction of the form $\mathrm{Tr}\left[\boldsymbol{S}(\hat{\mathcal{J}}\boldsymbol{S})^{\mathsf{T}}\right]$ is a rather simple and general interaction term. For lattice size dependent simulations it would be interesting to introduce different lattice sizes into the training data and study the behaviour of the resulting transformer-like effective Hamiltonian. Additionally, a comparative analysis of two Hamiltonians trained using different lattice sizes could yield further insight. Furthermore, an expansion to even more complex exact Hamiltonians and symmetries – like $SU(3)$ used in QCD – is desirable.

To conclude, this study demonstrates that advanced Monte Carlo techniques using machine learning, when carefully adapted to specific models, can significantly enhance simulation efficiency and provide deeper insights into complex interacting systems such as quantum spin systems.

A. 1 Particle Reduced Hamiltonian - Calculations

In this appendix, detailed calculations for the derivation of the 1-particle reduced Hamiltonian used for the Double Exchange model are provided. The goal is to reduce the computational complexity by projecting the full Hamiltonian onto a one-particle subspace. The key steps involve the application of ladder operator anticommutation relations and the evaluation of matrix elements between single-particle states. The starting point is the full Hamiltonian provided in equation (2.11). Using the one-particle wave functions $|\hat{c}^\dagger_{n\sigma}\,\emptyset\rangle$, the one-particle Hamiltonian is projected from the full Hamiltonian,

$$\begin{aligned}
{}^{1p}_{DE}\mathcal{H}_{nm\rho\rho'}(\{\boldsymbol{S}\}) &= \langle\emptyset\,\hat{c}_{n\rho}|\,{}_{DE}\mathcal{H}(\{\boldsymbol{S}\})\,|\hat{c}^\dagger_{m\rho'}\,\emptyset\rangle \\
&= -2t\,\langle\emptyset\,\hat{c}_{n\rho}|\sum_{\langle i,j\rangle,\sigma}\left(\hat{c}^\dagger_{i,\sigma}\hat{c}_{j,\sigma}+\hat{c}^\dagger_{j,\sigma}\hat{c}_{i,\sigma}\right)|\hat{c}^\dagger_{m\rho'}\,\emptyset\rangle \\
&\quad + \frac{J}{2}\,\langle\emptyset\,\hat{c}_{n\rho}|\sum_{i,\sigma,\sigma'}\hat{c}^\dagger_{i,\sigma}\underbrace{(\boldsymbol{S}_i\cdot\boldsymbol{\sigma})_{\sigma\sigma'}}_{\phi^i_{\sigma\sigma'}}\hat{c}_{i,\sigma'}\,|\hat{c}^\dagger_{m\rho'}\,\emptyset\rangle \\
&\quad - \mu\,\langle\emptyset\,\hat{c}_{n\rho}|\sum_{i,\sigma}\hat{c}^\dagger_{i,\sigma}\hat{c}_{i,\sigma}\,|\hat{c}^\dagger_{m\rho'}\,\emptyset\rangle \\
&= -t\mathcal{A}+\frac{J}{2}\mathcal{B}-\mu\mathcal{C},
\end{aligned} \tag{A.1}$$

where the parameters $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are given by the following equations.

- Term $\mathcal{A}$:

$$\begin{aligned}\langle \emptyset\, \hat{c}_{n\rho} |\, \hat{c}^\dagger_{i\sigma}\hat{c}_{j\sigma}\, |\hat{c}^\dagger_{m\rho'}\, \emptyset\rangle &= \langle \emptyset\, \hat{c}_{n\rho}\hat{c}^\dagger_{i\sigma}\big(\delta_{jm}\delta_{\sigma\rho'} - \cancel{\hat{c}^\dagger_{m\rho'}\hat{c}_{j\sigma}}\big)\, \emptyset\rangle \\ &= \langle \emptyset\, \delta_{jm}\delta_{\sigma\rho'}\big(\delta_{in}\delta_{\sigma\rho} - \cancel{\hat{c}^\dagger_{i\sigma}\hat{c}_{n\rho}}\big)\, \emptyset\rangle \\ &= \delta_{jm}\delta_{\sigma\rho'}\delta_{in}\delta_{\sigma\rho}\end{aligned} \tag{A.2}$$

$$\begin{aligned}\Rightarrow \mathcal{A} &= \sum_{\langle i,j\rangle,\sigma}\big(\delta_{jm}\delta_{\sigma\rho'}\delta_{in}\delta_{\sigma\rho} + \delta_{im}\delta_{\sigma\rho'}\delta_{jn}\delta_{\sigma\rho}\big) = \delta_{\rho\rho'}\sum_{\langle i,j\rangle}\big(\delta_{jm}\delta_{in} + \delta_{im}\delta_{jn}\big) \\ &= 2\delta_{\rho\rho'}\sum_{\langle i,j\rangle}\delta_{nj}\delta_{im}\end{aligned} \tag{A.3}$$

- Term $\mathcal{B}$:

$$\begin{aligned}\langle \emptyset\, \hat{c}_{n\rho} |\, \hat{c}^\dagger_{i\sigma}\phi^i_{\sigma\sigma'}\hat{c}_{i\sigma'}\, |\hat{c}^\dagger_{m\rho'}\, \emptyset\rangle &= \phi^i_{\sigma\sigma'}\langle \emptyset\, \hat{c}_{n\rho}\hat{c}^\dagger_{i\sigma}\big(\delta_{im}\delta_{\sigma'\rho'} - \cancel{\hat{c}^\dagger_{m\rho'}\hat{c}_{i\sigma'}}\big)\, \emptyset\rangle \\ &= \phi^i_{\sigma\sigma'}\delta_{im}\delta_{\sigma'\rho'}\langle \emptyset\, \big(\delta_{in}\delta_{\sigma\rho} - \cancel{\hat{c}^\dagger_{i\sigma}\hat{c}_{n\rho}}\big)\, \emptyset\rangle \\ &= \phi^i_{\sigma\sigma'}\delta_{im}\delta_{\sigma'\rho'}\delta_{in}\delta_{\sigma\rho}\end{aligned} \tag{A.4}$$

$$\Rightarrow \mathcal{B} = \sum_{i,\sigma,\sigma'}\big(\phi^i_{\sigma\sigma'}\delta_{im}\delta_{\sigma'\rho'}\delta_{in}\delta_{\sigma\rho}\big) = \phi^m_{\rho\rho'}\delta_{nm} \tag{A.5}$$

- Term $\mathcal{C}$:

$$\begin{aligned}\langle \emptyset\, \hat{c}_{n\rho} |\, \hat{c}^\dagger_{i,\sigma}\hat{c}_{i,\sigma}\, |\hat{c}^\dagger_{m\rho'}\, \emptyset\rangle &= \langle \emptyset\, \hat{c}_{n\rho}\hat{c}^\dagger_{i,\sigma}\big(\delta_{im}\delta_{\sigma\rho'} - \cancel{\hat{c}^\dagger_{m\rho'}\hat{c}_{i,\sigma}}\big)\, \emptyset\rangle \\ &= \langle \emptyset\, \delta_{im}\delta_{\sigma\rho'}\big(\delta_{in}\delta_{\sigma\rho} - \cancel{\hat{c}^\dagger_{i,\sigma}\hat{c}_{n\rho}}\big)\, \emptyset\rangle \\ &= \delta_{im}\delta_{\sigma\rho'}\delta_{in}\delta_{\sigma\rho}\end{aligned} \tag{A.6}$$

$$\Rightarrow \mathcal{C} = \sum_{i,\sigma}\delta_{im}\delta_{\sigma\rho'}\delta_{in}\delta_{\sigma\rho} = \delta_{nm}\delta_{\rho\rho'} \tag{A.7}$$

If all terms are simplified, the one-particle Hamiltonians is given by

$${}^{1p}_{DE}\mathcal{H}_{nm\rho\rho'}(\{\boldsymbol{S}\}) = -2t\delta_{\rho\rho'}\sum_{\langle i,j\rangle}\delta_{nj}\delta_{im} + \frac{J}{2}\delta_{nm}(\boldsymbol{S}_m\cdot\boldsymbol{\sigma})_{\rho\rho'} - \mu\delta_{nm}\delta_{\rho\rho'}. \tag{A.8}$$

B. Pseudoalgorithms for Monte Carlo Update Methods

This appendix collects the pseudo-code for the Monte Carlo update algorithms discussed in chapter 3. These pseudoalgorithms provide step-by-step procedures for implementing the Metropolis, Wolff cluster, Self-Learning Monte Carlo, and Hybrid Monte Carlo methods. They serve both as a reference for algorithmic details and as a guide for potential reimplementations.

B.1. Metropolis Algorithm

Algorithm 3 *Pseudocode for the Metropolis algorithm.*

Initialize state $\mathfrak{x}$
for $i = 1$ to N **do**
    Generate candidate $\mathfrak{x}'$
    Compute the acceptance probability $\alpha = \min\left(1,\, e^{-\beta \Delta E}\right)$
    Choose $r \sim \mathcal{U}(0, 1)$
    if $r < \alpha$ **then**
        Accept the move: $\mathfrak{x} \leftarrow \mathfrak{x}'$
    else
        Reject the move: Keep $\mathfrak{x}$ unchanged
    end if
    Compute the observable $\mathcal{O}_i$ for the configuration $\mathfrak{x}$
end for
Calculate the ensemble average $\langle \mathcal{O} \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}_i$

B.2. Wolff Clustering Algorithm

Algorithm 4 *Pseudocode for the Wolff clustering algorithm.*

Initialize spin state x
for $t = 1$ to N **do**
    Choose a random initial spin $\boldsymbol{S}_i$
    Choose a random reflection vector $\boldsymbol{r}$
    Initialize an empty cluster and add $\boldsymbol{S}_i$ to it
    Mark $\boldsymbol{S}_i$ as visited
    Initialize a stack with $\boldsymbol{S}_i$
    while stack is not empty **do**
        Pop a spin $\boldsymbol{S}'$ from the stack
        for each unvisited neighbour $\boldsymbol{S}_j$ of $\boldsymbol{S}'$ **do**
            Compute bond activation probability: $p = 1 - e^{\min[0, -2\beta(\boldsymbol{r}\cdot\boldsymbol{S}')(\boldsymbol{r}\cdot\boldsymbol{S}_j)]}$
            Mark $\boldsymbol{S}_j$ as visited
            Draw $r \sim \mathcal{U}(0, 1)$
            if $r < p$ **then**
                Add $\boldsymbol{S}_j$ to the cluster
                Push $\boldsymbol{S}_j$ onto the stack
            end if
        end for
    end while
    Flip all spins in the cluster
end for

B.3. Self Learning Monte Carlo Algorithm

Algorithm 5 *Pseudocode for the SLMC algorithm.*

Perform trial simulation using local updates and $\mathcal{H}$ to generate training data
Train an effective Hamiltonian $\mathcal{H}_{\text{eff}}$
Initialize state $\mathfrak{x}$.
for $i = 1$ to N **do**
    Propose a new configuration $\mathfrak{x}'$ using $\mathcal{H}_{\text{eff}}$
    Compute acceptance probability $\alpha = \min(1, \frac{p_{eq}(\mathfrak{x}')}{p_{eq}(\mathfrak{x})} \frac{p_{eq,\,\text{eff}}(\mathfrak{x})}{p_{eq,\,\text{eff}}(\mathfrak{x}')})$
    Choose $r \sim \mathcal{U}(0, 1)$
    if $r < \alpha$ **then**
        Accept the move: $\mathfrak{x} \leftarrow \mathfrak{x}'$
    else
        Reject the move: Keep $\mathfrak{x}$ unchanged
    end if
    Compute the observable $\mathcal{O}_i$ for the configuration $\mathfrak{x}$
end for
Calculate the ensemble average $\langle \mathcal{O} \rangle = \frac{1}{N} \sum_{i=1}^{N} \mathcal{O}_i$

B.4. Hybrid Monte Carlo Algorithm

Algorithm 6 *Pseudocode for the Hybrid Monte Carlo algorithm.*

Initialize state $\mathfrak{x}$
for $i = 1$ to N **do**
    Sample momentum $\mathfrak{p} \sim \mathcal{N}(0, \mathcal{M})$
    Compute initial energy $E = \mathbb{H}(\mathfrak{x}, \mathfrak{p}) = \mathcal{H}(\mathfrak{x}) + \frac{1}{2}\mathfrak{p}^T\mathcal{M}^{-1}\mathfrak{p}$
    Compute new configuration $\mathfrak{x}'$, $\mathfrak{p}'$ using Hamilton's equations
    Compute new energy $E' = \mathbb{H}(\mathfrak{x}', \mathfrak{p}')$
    $\mathfrak{p}' \leftarrow -\mathfrak{p}'$
    Compute the acceptance probability $\alpha = \min(1, e^{-\beta(E'-E)})$
    Choose $r \sim \mathcal{U}(0, 1)$
    if $r < \alpha$ **then**
        Accept the move: $\mathfrak{x} \leftarrow \mathfrak{x}'$
    else
        Reject the move: Keep $\mathfrak{x}$ unchanged
    end if
    Compute the observable $\mathcal{O}_i$ for the configuration $\mathfrak{x}$
end for
Calculate the ensemble average $\langle\mathcal{O}\rangle = \frac{1}{N}\sum_{i=1}^{N}\mathcal{O}_i$

Bibliography

[1] Nicholas Metropolis et al. "Equation of State Calculations by Fast Computing Machines". In: *The Journal of Chemical Physics* 21.6 (June 1953). ISSN: 0021-9606. DOI: 10.1063/1.1699114.

[2] W Lenz. "Beitrag zum Verständnis der magnetischen Erscheinungen in festen Körpern". In: *Z. Phys.* 21 (1920), pp. 613–615. URL: https://cds.cern.ch/record/460663.

[3] Junwei Liu et al. "Self-learning Monte Carlo method". In: *Phys. Rev. B* 95 (4 Jan. 2017). DOI: 10.1103/PhysRevB.95.041101.

[4] Yuki Nagai and Akio Tomiya. *Self-learning Monte Carlo with equivariant Transformer*. 2024. arXiv: 2306.11527 [cond-mat.str-el]. URL: https://arxiv.org/abs/2306.11527.

[5] Clarence Zener. "Interaction between the *d*-Shells in the Transition Metals. II. Ferromagnetic Compounds of Manganese with Perovskite Structure". In: *Phys. Rev.* 82 (3 May 1951), pp. 403–405. DOI: 10.1103/PhysRev.82.403.

[6] Georgios Stratis et al. "Sample generation for the spin-fermion model using neural networks". In: *Phys. Rev. B* 106 (20 Nov. 2022), p. 205112. DOI: 10.1103/PhysRevB.106.205112.

[7] Franz Schwabl. *Statistical Mechanics*. Advanced Texts in Physics. Springer Berlin, Heidelberg, 2006. ISBN: 978-3-540-36217-3. DOI: 10.1007/3-540-36217-7.

[8] Ernst Ising. "Beitrag zur Theorie des Ferromagnetismus". In: *Zeitschrift für Physik* 31.1 (1925). ISSN: 0044-3328. DOI: 10.1007/BF02980577.

[9] Lars Onsager. "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition". In: *Phys. Rev.* 65 (3-4 Feb. 1944). DOI: 10.1103/PhysRev.65.117.

[10] Stephen G. Brush. "History of the Lenz-Ising Model". In: *Rev. Mod. Phys.* 39 (4 Oct. 1967). DOI: 10.1103/RevModPhys.39.883.

[11] J.L. Alonso et al. "Hybrid Monte Carlo algorithm for the double exchange model". In: *Nuclear Physics B* 596.3 (2001), pp. 587–610. ISSN: 0550-3213. DOI: 10.1016/S0550-3213(00)00681-7.

[12] P. Jordan and E. Wigner. "Über das Paulische Äquivalenzverbot". In: *Zeitschrift für Physik* 47.9 (Sept. 1928), pp. 631–651. ISSN: 0044-3328. DOI: 10.1007/BF01331938.

[13] Pierre Brémaud. *Markov Chains. Gibbs Fields, Monte Carlo Simulation and Queues*. 2nd ed. Texts in Applied Mathematics. Springer Cham, 2020. ISBN: 978-3-030-45982-6. DOI: 10.1007/978-3-030-45982-6.

[14] David P. Landau and Kurt Binder. *A Guide to Monte Carlo Simulations in Statistical Physics*. 5th ed. Cambridge University Press, 2021. ISBN: 978-1-108-78034-6. DOI: 10.1017/9781108780346.

[15] Ulli Wolff. "Collective Monte Carlo Updating for Spin Systems". In: *Phys. Rev. Lett.* 62 (4 Jan. 1989). DOI: 10.1103/PhysRevLett.62.361.

[16] Michael Betancourt. *A Conceptual Introduction to Hamiltonian Monte Carlo*. 2018. arXiv: 1701.02434 [stat.ME]. URL: https://arxiv.org/abs/1701.02434.

[17] David Müller. *Notes on the $O(3)$ non-linear σ-model*. Personal email correspondence, December 2024. Email: dmueller@hep.itp.tuwien.ac.at.

[18] Liane Backfried. "Machine learning a perfect lattice action for the O(3) non-linear sigma-model". MA thesis. Wien: Technische Universität Wien - E136 - Institut für Theoretische Physik, 2025. DOI: 10.34726/hss.2025.124787.

[19] Yurii Nesterov. "Primal-dual subgradient methods for convex problems". In: *Mathematical Programming* 120.1 (2009). ISSN: 1436-4646. DOI: 10.1007/s10107-007-0149-x.

[20] Matthew D. Hoffman and Andrew Gelman. *The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo*. 2011. arXiv: 1111.4246 [stat.CO]. URL: https://arxiv.org/abs/1111.4246.

[21] Emmy Noether. "Invariante Variationsprobleme". In: *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse* 23 (1918), pp. 235–257.

[22] Taco S. Cohen and Max Welling. *Group Equivariant Convolutional Networks*. 2016. arXiv: 1602.07576 [cs.LG]. URL: https://arxiv.org/abs/1602.07576.

[23] Akio Tomiya and Yuki Nagai. "Equivariant transformer is all you need". In: *PoS* LATTICE2023 (2023). DOI: 10.22323/1.453.0001.

[24] Ashish Vaswani et al. *Attention Is All You Need*. 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.

[25] Shiv Ram Dubey, Satish Kumar Singh, and Bidyut Baran Chaudhuri. "Activation functions in deep learning: A comprehensive survey and benchmark". In: *Neurocomputing* 503 (2022). ISSN: 0925-2312. DOI: 10.1016/j.neucom.2022.06.111.

[26] Frank Rosenblatt. "The perceptron: A probabilistic model for information storage and organization in the brain". In: *Psychological Review* 65.6 (1958). DOI: 10.1037/h0042519.

[27] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (1986). DOI: 10.1038/323533a0.

[28] G. E. Hinton and R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786 (2006), pp. 504–507. DOI: 10.1126/science.1127647.

[29] Alan D. Sokal. "Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms Note to the Reader". In: 1996. URL: https://api.semanticscholar.org/CorpusID:14817657.