Technische Universität München
Fakultät für Mathematik

Tensor-Network-Methods for Simulating Infinite 1-dimensional Quantum-Many-Body Systems

Bachelorarbeit von Alexander Müller-Hermes

angefertigt am Max-Planck-Institut für Quantenoptik

Aufgabensteller: Prof. Dr. Folkmar Bornemann
Betreuer: Dr. Mari-Carmen Banũls

Abgabetermin: 15. Juni 2010

Ich erkläre hiermit, daß ich diese Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

(Garching b. München den 15. Juni 2010, Alexander Müller-Hermes)

Abstract

In this thesis we will introduce the so called tensor-network methods for the simulation of infinite 1-dimensional quantum-many-body systems. These numerical methods are, in the 1-dimensional case, based on the formalism of matrix product states. This formalism allows an efficient representation of states in 1-dimensional systems. We will use this representation to formulate numerical algorithms for the simulation of 1-dimensional quantum-many-body systems. In this work we will focus on the TEBD-algorithm for the simulation of real time-evolution and present some improvements with respect to its original formulation. Furthermore we will show how to calculate ground states of 1-dimensional systems using the imaginary time-evolution. This technique will be used to calculate ground state properties of the 1-dimensional Ising model in transverse field and the dipolar XXZ model. We will also comment the implementation of these methods in MATLAB.

Zusammenfassung

In dieser Arbeit stellen wir sogenannte Tensor-Netzwerk-Verfahren zur Simulation unendlicher 1-dimensionaler Quanten-Vielteilchen Systeme vor. Diese numerischen Verfahren basieren in 1-dimensionalen Systemen auf dem Formalismus der Matrix-Produkt-Zustände. Dieser Formalismus ermöglicht eine effiziente Darstellung der Zustände eines solchen Quantensystems. Mithilfe des Matrix-Produkt-Formalismus können numerische Verfahren zur Simulation von 1-dimensionalen Quanten-Vielteilchen Systemen konstruiert werden. Wir werden den TEBD-Algorithmus vorstellen und Verbesserungen gegenüber seiner ursprünglichen Formulierung darlegen. Zusätzlich zeigen wir, wie dieser Algorithmus dazu benutzt werden kann Grundzustände eines 1-dimensionalen Systems zu berechnen. Diese Technik werden wir dann dazu benutzen Eigenschaften der Grundzustände des 1-dimensionalen Ising Modells in transversalem Feld und des XXZ-Modells zu berechnen. Wir werden außerdem die Implementierung der vorgestellten Verfahren in MATLAB erläutern.

Acknowledgments

I am deeply grateful to Dr. Mari-Carmen Banũls for her constant support and help concerning this thesis. She answered all my questions with great patience and provided interesting topics for my work at the MPQ.

I also thank Prof. Dr. Ignacio Cirac for the opportunity to work at the MPQ and to do this thesis in his group. Finally I also want to thank Prof. Dr. Folkmar Bornemann for helpful advices and supervising this thesis.

Contents

1. Introduction and Overview

Computer simulations of physical systems have become an important tool in modern science and provide new possibilities in theoretical as well as experimental areas. For theorists there is the possibility to test a new model in an easily controllable environment. On the other hand there is the possibility to investigate the interesting parameters, in order to design an experiment in the real world. Finally computer simulations are often the only possible way to obtain results in the absence of an analytical solution. Quantum Theory suffers from the problem, that its underlying structure makes it difficult to simulate general quantum-many-body systems in an efficient way. A main source of complexity is the exponential growth of the dimension of the total Hilbert Space, w.r.t the number of subsystems needed to describe a composed quantum system. There has been major research in this area in recent years which led to the development of new algorithms for the simulation of such systems. Important steps in this direction were the development of the formalism of Matrix Product States(MPS) [FNW92] and as a major event the development of the Density Matrix Renormalization Group ($DMRG$) [Whi92b] [Whi92a]. The formalism of Matrix Product States led to a better understanding of DMRG [OR95] [RO97], and quantum information provided a new perspective [VPC04] [Vid03], which allowed the development of new algorithms, like TEBD [Vid03], for the simulation of the time-evolution of 1-dimensional systems. In this thesis we are concerned with the simulation of infinite 1-dimensional quantum-many-body systems using algorithms based on MPS.

In Chapter 2 we will introduce some basic concepts and motivate the following chapters. We will also introduce the valence bond construction. In Chapter 3 we will give an introduction to the Matrix Product Formalism which is used in the algorithm. In Chapter 4 we describe the TEBD-Algorithm in detail and the possibility to calculate ground states with it. We will also present improvements of the original algorithm by the use of translationally invariant MPO representations of the occurring operators and the use of MPS of higher periodicity than one. In Chapter 6 we test the power of the algorithm on two examples, namely the 1-dimensional Ising model and a spin chain with a XXZ-Hamiltonian with long-term interaction. In Chapter 5 we will say a few words about the efficient implementation of the described methods using MATLAB. In Chapter 7 we finish the thesis with a short conclusion.

2. Motivation and the Valence Bond Construction

We will begin this chapter with a short discussion of general problems occur

In the following we will always consider quantum many-body systems in one spatial dimension, which are composed of subsystems referred to as sites. The state of each site is described by a vector on a d-dimensional Hilbert space denoted by $\mathcal{H}_k$ where k is the index of the site. We will require that the local dimension d is finite and site independent. As an example one can think of a spin chain of N particles each of which has spin $\frac{d-1}{2}$.

For such a system of N sites the Hilbert space of the whole system takes the form $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_N$.

A general state $|\psi\rangle \in \mathcal{H}$ is of the following form

$$|\psi\rangle = \sum_{i_1,\cdots,i_N=1}^{d} c_{i_1,\ldots,i_N} |i_1, i_2, \ldots, i_N\rangle \ , \tag{2.1}$$

with $c_{i_1,\ldots,i_N} \in \mathbb{C}$ and $\{|i_1, i_2, \ldots, i_N\rangle\}$ denoting the computational basis. This form indicates a major problem, if we want to do numerical simulations of quantum many-body systems. To describe a general state $|\psi\rangle \in \mathcal{H}$ we will need to save the d^N numbers $c_{i_1,\ldots,i_N}$. This means that the amount of memory and the computational cost, needed to simulate the system, scales exponentially with the size N of the system.

In the following chapters we will describe algorithms solving two problems. First to simulate the time-evolution of very large quantum many-body systems and second to find ground states of certain systems. These two tasks are connected and can basically be solved by the same algorithm. We will focus on the simulation of systems in the thermodynamic limit, i.e. systems in the limit $N \to \infty$.

The idea that makes these simulations feasible is that the states which are physical relevant are not arbitrary states in the Hilbert space. It can be shown that the most important states, such as ground states or thermal states, occurring in a quantum system in nature can be described by a reduced number of parameters, which grows polynomially with the size N of the system [CV09]. Therefore only a small subset of the whole Hilbert space is relevant for the study of a physical system. The problem lies in finding an efficient parametrization of this subset, which also allows the efficient calculation of properties, such as expectation values, of states in this set.

To characterize the subset of physically relevant states, we can look for a special property, which can be used for this purpose. This property is the amount of entanglement in a state.

Definition 1. *A pure state $|\psi\rangle \in \mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$ is called a* ***product state*** *if it has the form*

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \ , \tag{2.2}$$

with states $|\phi_k\rangle \in \mathcal{H}_k$.

If $|\psi\rangle \in \mathcal{H}$ is not of the above form it is called ***entangled****.*

This definition can be generalized to bigger systems. For further analysis it is important to quantify the amount of entanglement in a particular state of the form (2.1). This can be done using the Schmidt decomposition:

Theorem 1. (*See [NC07]*)

Let $|\psi\rangle \in \mathcal{H}$ be a pure state and $\mathcal{H} = \mathcal{H}_A \otimes \mathcal{H}_B$ a partition of the total system in two subsystems with finite dimensions d_A and d_B. Then there exists a decomposition, called ***Schmidt decomposition****, of the form*

$$|\psi\rangle = \sum_{i=1}^{\chi} \lambda_i |i_A\rangle |i_B\rangle \ , \tag{2.3}$$

with $\lambda_i \in \mathbb{R}^+ \setminus \{0\}$, called ***Schmidt coefficients****, satisfying $\sum_i \lambda_i^2 = 1$. Here $|i_A\rangle$ and $|i_B\rangle$ denote orthonormal states for the subsystems A and B enumerated by i. The numbers χ and λ_i in this decomposition are uniquely determined by $|\psi\rangle$. χ is called the* ***Schmidt rank*** *of the state $|\psi\rangle$ according to the splitting A:B. To emphasize the dependence on the splitting, we will sometimes write $\chi_{A:B}[\psi]$ instead of χ.*

Proof. $|\psi\rangle$ can be written in the following form for orthonormal bases $|j\rangle$ and $|k\rangle$ for the systems A and B:

$$|\psi\rangle = \sum_{j,k} c_{j,k} |j\rangle |k\rangle \tag{2.4}$$

As c is a $d_A \times d_B$ matrix, we can perform a singular value decomposition $c = usv$, with diagonal matrix s and unitary matrices u and v. Then we can sum over the now separated indices j and k, which corresponds to a unitary transformation of the orthonormal bases $|j\rangle$ and $|k\rangle$ which preserves orthonormality:

$$|\psi\rangle = \sum_{j,i,k} u_{j,i} s_{i,i} v_{i,k} |j\rangle |k\rangle \tag{2.5}$$

$$= \sum_{i} s_{i,i} |i_A\rangle |i_B\rangle \ . \tag{2.6}$$

We finally set $\lambda_i = s_{i,i}$ and observe that the Schmidt rank and coefficients are uniquely determined, because s is uniquely determined by the theorem of the singular value decomposition. □

Using the Schmidt rank we can define the function $E_\chi : \mathcal{H} \to \mathbb{R}$ by for a Hilbert space $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_N$ by

$$E_\chi(\psi) = \log_d \left(\max_{A:B} \left(\chi_{A:B}[\psi] \right) \right) \ . \tag{2.7}$$

This function has some important properties following from the Schmidt decomposition:

Corollary 1. (*See [Vid03]*)
Let $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_N$ and $E_\chi : \mathcal{H} \to \mathbb{R}$ be defined as above, then

1. *$E_\chi(\psi) \geq 0$ for all $|\psi\rangle \in \mathcal{H}$.*
2. *$|\psi\rangle$ is product state iff $E_\chi(\psi) = 0$.*
3. *$E_\chi(\phi_1 \otimes \phi_2) = E_\chi(\phi_1) + E_\chi(\phi_2)$ for all tensor products of states $\phi_1 \otimes \phi_2 \in \mathcal{H}$.*
4. *$E_\chi(\psi)$ is invariant under local unitary transformations of $|\psi\rangle$.*
5. *$E_\chi(\psi)$ cannot be increased under arbitrary local transformations of $|\psi\rangle$.*

A transformation is called local, if it acts nontrivial on only one of the two spaces $\mathcal{H}_A$ and $\mathcal{H}_B$.

These properties indicate that we can take the E_χ to quantify entanglement, as entanglement behaves the same under local transformations. A function which fulfills these properties is called an **entanglement measure**. We will use E_χ in the following to quantify entanglement in our states. For more details of entanglement and entanglement measures see [HHHH07] and [Per05].

2.1. The Valence Bond Picture

We will now introduce a description of states, which leads to an efficient representation for states with bounded amount of entanglement. To do so we will introduce the valence bond picture, which leads to such a representation. This construction is equivalent to the matrix product states in the case of finite systems. For infinite systems the matrix product formalism provides a generalization of valence bond states. This chapter will therefore provide a picture, which shows where matrix product states arise and which alleviates their understanding.

The valence bond states were first introduced by Affleck, Kennedy, Lieb and Tasaki [AKLT88] to study ground state properties of Quantum Antiferromagnets.

Let $|\psi\rangle \in \mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_N$ be a pure state of a finite 1-dimensional system composed of N sites. In the valence bond picture we assign to each site two virtual spins with states in auxiliary Hilbert spaces $\tilde{H}_k$ and $\tilde{H}_{k+1}$ of dimension D_k and D_{k+1}, where k is the number of the site. Now let each pair of neighboring virtual spins which correspond to different sites be in a maximally entangled state $|I\rangle = \sum_{b=1}^{D_{k+1}} |b, b\rangle$, which is called an entangled bond (See Fig.2.1). Note that the dimensions D_k of the entangled bonds have no physical meaning. They are parameters of the model, which we can choose arbitrary. To get back to the physical state, we define maps $\mathbf{A}_k : \tilde{H}_k \otimes \tilde{H}_{k+1} \to H_k$ on every site via

$$\mathbf{A}_k = \sum_{i=1}^{d} \sum_{a=1}^{D_k} \sum_{b=1}^{D_{k+1}} A[k]^i_{a,b} \, |i\rangle \langle a, b| \ , \tag{2.8}$$

with $A[k]^i \in \mathbb{C}^{D_k \times D_{k+1}}$. These maps translate between the model and the physical system. We can now define states by stating their maps in the valence bond picture. States which are defined this way are called **valence bond states**. By fixing the maximal occurring dimension D_k, we obtain sets of states, which have bounded amount of entanglement. As we will see later, the maximal occurring dimension D_k is also connected to the number of parameters, which are needed to describe the state. This can be used to describe states with a certain amount of entanglement in an efficient way.

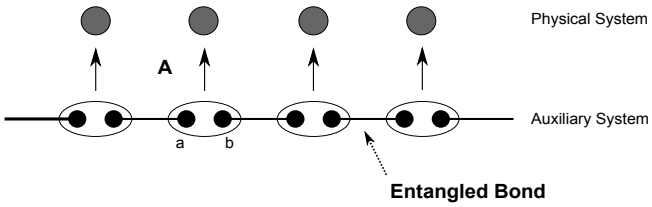

Figure 2.1.: Valence Bond Picture. Each dot in the auxiliary system represents a virtual spin.

For a finite system we have to define boundary conditions for this construction. One possibility is to put only one virtual spin at each of the two ends of the virtual system (See Picture 2.2). This means that at the boundaries we have

$$\mathbf{A}_1 = \sum_{i=1}^{d} \sum_{b=1}^{D_2} A[1]_b^i \, |i\rangle \langle b| \ , \tag{2.9}$$

$$\mathbf{A}_N = \sum_{i=1}^{d} \sum_{a=1}^{D_N} A[N]_a^i \, |i\rangle \langle a| \ , \tag{2.10}$$

with vectors $A[1]^i$ and $A[N]^i$. We will call states of this form, valence bond states in **open boundary conditions**. If we now apply all maps $\mathbf{A}_k$ to the auxiliary system, we obtain a state of the form

$$|\psi\rangle = \sum_{i_1,\ldots,i_N} A[1]^{i_1} A[2]^{i_2} \cdots A[N-1]^{i_{N-1}} A[N]^{i_N} \, |i_1, i_2, \ldots, i_N\rangle \ , \tag{2.11}$$

where $A[k]^i$ denotes the defining matrices for the map $\mathbf{A}_k$. We can therefore express the state also through the defining matrices $A[k]^i$. We will call a state of the form (2.11) a matrix product state with open boundary condition.

It will be useful to introduce another kind of boundary condition for studying states, which are invariant under certain cyclic permutations of the sites. In such cases we can think of the finite system as closed, forming a ring of sites. We therefore make the identification $H_k = H_{\mathrm{k \bmod N}}$ for $k \in \mathbb{Z}$ and N the total number of sites. Then we can think of states defined on this ring, which are invariant under translations of k sites. We will call such states k-periodic and

a 1-periodic state **translationally invariant**. To represent k-periodic states in the valence bond construction we can put an entangled bond into the auxiliary system connecting the first and the last particle (See Picture 2.2). We will refer to this conditions as **periodic boundary conditions** and they will be important when we deal with infinite systems. In the next chapter we will show that we can represent each translationally invariant state as a valence bond state with site-independent maps $\mathbf{A}$.

If we again apply all maps $\mathbf{A}_k$ to the auxiliary system, we obtain a state of the form

$$|\psi\rangle = \sum_{i_1,\ldots,i_N=1}^{d} Tr\left[A\left[1\right]^{i_1} A\left[2\right]^{i_2} \cdots A\left[N\right]^{i_N}\right] |i_1, i_2, \ldots, i_N\rangle \ , \tag{2.12}$$

where the trace arises from the summation over the entangled bond connecting the two ends. We can of course also write valence bond states with open boundary conditions this way. States of the form (2.12) are called matrix product states and we will use the next chapter to develop the matrix product formalism in detail.

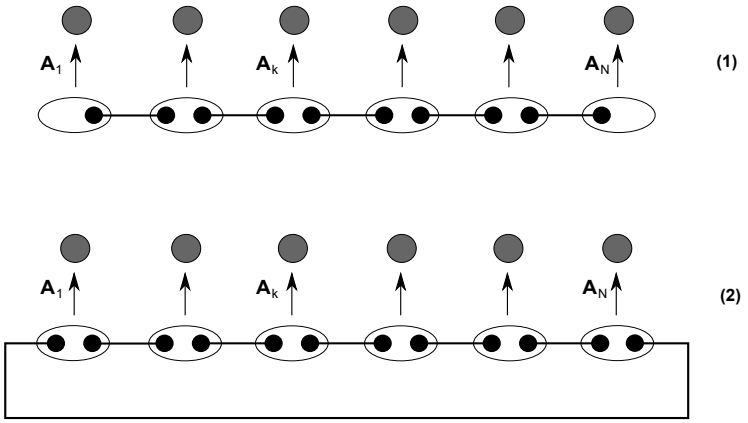

Figure 2.2.: (1) Open boundary conditions. (2) Periodic boundary conditions.

3. Matrix Product States and Matrix Product Operators

This chapter provides the theoretical framework, which we will use to develop numerical algorithms for the simulation of 1-dimensional many-body quantum systems. We will introduce the Matrix Product Formalism, which allows us to represent states of the physical system and operators corresponding to observables of the physical system in an efficient way.

In the following we are dealing with the same kind of physical system as in the last chapter, i.e. a 1-dimensional quantum many-body system composed of N sites. The full Hilbert space has the form $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_N$, with $\mathcal{H}_k$ as defined above. Again we assume $dim\left(\mathcal{H}_k\right) = d < \infty$ for all k.

Definition 2. (*See* [*PGVWC07*])

1. *A* ***Matrix Product State***(MPS) *in* ***periodic boundary condition*** (PBC) *is a pure state* $|\psi\rangle \in \mathcal{H}$ *of the form*

$$|\psi\rangle = \sum_{i_1,\ldots,i_N=1}^{d} Tr\left[A\left[1\right]^{i_1} A\left[2\right]^{i_2} \cdots A\left[N\right]^{i_N}\right] |i_1, i_2, \ldots, i_N\rangle \ , \tag{3.1}$$

   where $A\left[k\right]^i$ *is a* $D_k \times D_{k+1}$ *matrix corresponding to site* $k \in \{1, \cdots, N\}$ *with* $D_{N+1} = D_1$. $D = \max_n D_n$ *is called the* ***bond dimension*** *of the MPS.*

2. *If* $D_1 = D_{N+1} = 1$ *the MPS is said to be with* ***open boundary conditions*** (OBC).

3. *We call a MPS in periodic boundary conditions with site-independent matrices,* $A\left[k\right]^i = A^i$ *for all k,* ***translationally invariant*** (TI).

One can think of a MPS in PBC as defined on a ring of particles, as introduced in the valence bond picture. Therefore we can make the identification $A\left[N+1\right]^i = A\left[1\right]^i$.

We can now count the number of parameters needed to describe a state as a MPS. Instead of the d^N parameters which we need in the ordinary representation, we are now using up to dD^2N complex parameters in the MPS representation. This means that there is an advantage of this representation compared to a direct representation in the computational basis as in (2.1), if the bond dimension D is not too big. In the valence bond picture, D defines the maximal dimension of occurring entangled bonds. Therefore states with lower amount of entanglement need smaller bond dimension when represented in this way. In this chapter we will show this property also for MPS. But first let us start with a few examples.

Example 1. *Every product state can be expressed as a MPS in OBC with bond dimension D=1. Take a product state*

$$|\psi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_N\rangle \ , \tag{3.2}$$

with $|\phi_k\rangle = \sum_{i_k} c_{i_k} |i_k\rangle$ *and* $c_{i_k} \in \mathbb{C}$. *Therefore we can write*

$$|\psi\rangle = \sum_{i_1,\ldots,i_N=1}^{d} c_{i_1} c_{i_2} \cdots c_{i_N} |i_1, i_2, \ldots, i_N\rangle \ . \tag{3.3}$$

Example 2. *As a further example one can consider an entangled state like the unnormalized GHZ state* $|000\cdots0\rangle + |111\cdots1\rangle$ *which can be written as a TI MPS with* $D = 2$ *and site independent matrices*

$$A^1 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, A^2 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

In the following we will show that every state can be written as a MPS in OBC with bond dimension depending on the amount of entanglement in the state.

3.1. Some Important Results on MPS in Open Boundary Condition

We will now present some results to MPS in OBC. First we will show that the set of all MPS in OBC is the full Hilbert space, i.e. every state can be written as a MPS in OBC. Then we will discuss some properties of MPS in OBC such as the connection between bond dimension and entanglement. We will state an efficient method for the calculation of expectation values of physical observables and finally define a canonical form for MPS.

Theorem 2. (*See* [*PGVWC07*])
Every pure state $|\psi\rangle \in \mathcal{H}$ *can be written as a MPS in open boundary conditions with bond dimension* $D \leq d^{\frac{N}{2}}$.

Proof. Start with the state $|\psi\rangle \in \mathcal{H}$ in the computational basis:

$$|\psi\rangle = \sum_{i_1,\ldots,i_N=1}^{d} c_{i_1,\ldots,i_N} |i_1, i_2, \ldots, i_N\rangle \ . \tag{3.4}$$

We obtain the $A[k]$ of the MPS representation by doing successive reduced singular value

decompositions (SVD) and recombinations in the following way:

$$\begin{aligned}
c_{i_1,\ldots,i_N} = c_{i_1,(i_2,\ldots,i_N)} &= \sum_{\alpha_1} U[1]_{i_1,\alpha_1} S[1]_{\alpha_1,\alpha_1} V[1]_{\alpha_1,(i_2,\ldots,i_N)} \\
&= \sum_{\alpha_1} A[1]^{i_1}_{\alpha_1} V[1]_{\alpha_1,(i_2,\ldots,i_N)} \\
&= \sum_{\alpha_1} A[1]^{i_1}_{\alpha_1} V[1]_{(\alpha_1,i_2)(i_3\cdots,i_N)} \\
&= \sum_{\alpha_1,\alpha_2} A[1]^{i_1}_{\alpha_1} U[2]_{(\alpha_1,i_2),\alpha_2} S[2]_{\alpha_2,\alpha_2} V[2]_{\alpha_2,(i_2,\ldots,i_N)} \\
&= \sum_{\alpha_1,\alpha_2} A[1]^{i_1}_{\alpha_1} A[2]^{i_2}_{\alpha_1,\alpha_2} V[2]_{\alpha_2,(i_2,\ldots,i_N)} \\
&= \cdots \\
&= \sum_{\alpha_1,\ldots,\alpha_N} A[1]^{i_1}_{\alpha_1} A[2]^{i_2}_{\alpha_1,\alpha_2} \cdots A[N-1]^{i_{N-1}}_{\alpha_{N-1},\alpha_N} A[N]^{i_N}_{\alpha_N}
\end{aligned}$$

This is the desired decomposition and one obtains a MPS in OBC. As in each SVD of a $m \times n$-matrix the number of singular values is at most $\min(m,n)$ the bond dimension satisfies $D \leq d^{\frac{N}{2}}$. □

Note that the upper bound on the bond dimension $D \leq d^{\frac{N}{2}}$ does not show that we can represent states more efficient compared to the ordinary representation. We need dD^2N parameters to represent a state as a MPS and for general states we will not save memory by doing this. But as we will show, the bond dimension is connected to the entanglement in the state. Therefore we get an efficient representation for states for which the amount of entanglement is not too high.

It is also important to note that we have just used basic properties of $\mathcal{H}$ as a tensor product to obtain the matrix product form. This means that the above construction works for all spaces with tensor product structure. Therefore it will also work for operators as the space $L(\mathcal{H})$ of linear operators on $\mathcal{H}$ is also a tensor product. We will use this later in order to construct a corresponding representation for operators.

There is an equivalent form of MPS introduced by G.Vidal [Vid03] for which the connection between the bond dimension and entanglement is clearer. The tensor c in the above expression is decomposed in the following way

$$c_{i_1,\ldots,i_N} = \sum_{\alpha_1,\ldots,\alpha_{N-1}} \Gamma[1]^{i_1}_{\alpha_1} \lambda[1]_{\alpha_1} \Gamma[2]^{i_2}_{\alpha_1,\alpha_2} \lambda[2]_{\alpha_2} \cdots \Gamma[N]^{i_N}_{\alpha_{N-1}} \quad . \tag{3.5}$$

This form can be obtained as in the above proof but now setting $U[k]^{i_k}_{\alpha_{k-1},\alpha_k} = \Gamma[k]^{i_k}_{\alpha_{k-1},\alpha_k}$ and $S[k]_{\alpha_k,\alpha_k} = \lambda[k]_{\alpha_k}$ instead of combining them.

In this form the interpretation of the SVDs as Schmidt Decompositions of the physical system

is evident, as the Schmidt decomposition at a site k of a state of this form reads as

$$|\psi\rangle = \sum_{\alpha_k} \lambda\left[k\right]_{\alpha_k} |1\cdots k-1;\alpha_k\rangle\, |k\cdots N;\alpha_k\rangle \ , \tag{3.6}$$

with basis vectors

$$\begin{aligned}
&|1\cdots k-1;\alpha_k\rangle \\
&= \sum_{i_1,\ldots,i_{k-1}} \sum_{\alpha_1,\ldots,\alpha_{k-1}} \Gamma\left[1\right]^{i_1}_{\alpha_1} \lambda\left[1\right]_{\alpha_1} \Gamma\left[2\right]^{i_2}_{\alpha_1,\alpha_2} \lambda\left[2\right]_{\alpha_2} \cdots \Gamma\left[k\right]^{i_2}_{\alpha_{k-1},\alpha_k} |i_1, i_2, \ldots, i_{k-1}\rangle \\
&|k\cdots N;\alpha_k\rangle \\
&= \sum_{i_k,\ldots,i_N} \sum_{\alpha_{k+1},\ldots,\alpha_N} \Gamma\left[k+1\right]^{i_k}_{\alpha_k,\alpha_{k+1}} \lambda\left[k+1\right]_{\alpha_{k+1}} \cdots \Gamma\left[N\right]^{i_N}_{\alpha_{N-1}} |i_k, i_{k+1}, \ldots, i_N\rangle \ .
\end{aligned}$$

We can use the entanglement measure defined above to connect the MPS representation with the entanglement of a state. As the maximal occurring number of singular values in each of the above SVDs is equal to χ (see definition), we obtain $D \leq \chi$. Because we use $\mathcal{O}\left(dND^2\right)$ parameters in the MPS representation as stated above, we need $\mathcal{O}\left(dN^3\right)$ parameters if we assume that $E_\chi = \mathcal{O}\left(\log_d\left(N\right)\right)$. Thus we need only polynomially, w.r.t. the system size, many parameters in the MPS representation, if the entanglement grows logarithmically in the system size.

We have seen so far, that the matrix product formalism provides an efficient representation of states, for which the amount of entanglement is not too high. Now we want to show, that we can also calculate expectation values of certain observables in an efficient way, if we are given a MPS.

Theorem 3. (*See [Per05]*)
Given an operator of the form

$$O = O_1 \otimes O_2 \otimes \cdots \otimes O_N : \mathcal{H} \to \mathcal{H} \ , \tag{3.7}$$

where $O_k : \mathcal{H}_k \to \mathcal{H}_k$ is a local operator acting on site k, its expectation in a normalized state $|\psi\rangle$ written as MPS of the form (3.1) *is given by*

$$\langle\psi| O |\psi\rangle = Tr\left(E_{O_1} \cdots E_{O_N}\right) \ . \tag{3.8}$$

Each $E_{O_k} \in \mathbb{C}^{(D_k)^2 \times (D_{k+1})^2}$ is of the form

$$E_{O_k} = \sum_{i_k,j_k=1}^{d} \langle j_k| O_k |i_k\rangle \left(A\left[k\right]^{j_k} \otimes \overline{A}\left[k\right]^{i_k}\right) \ , \tag{3.9}$$

and called transfer matrix of the MPS ψ corresponding to O_k. If $O_k = \mathbb{1}$ we will just write E_k for $E_{\mathbb{1}_k}$ or just E if the index does not matter.

Proof. By inserting (3.1) we obtain

$$\langle\psi|\, O\, |\psi\rangle = \sum_{i_1,\ldots,i_N=1}^{d} \sum_{j_1,\ldots,j_N=1}^{d} Tr\left(\prod_{k=1}^{N} A\,[k]^{i_k}\right) Tr\left(\prod_{k=1}^{N} \overline{A}\,[k]^{j_k}\right) \prod_{k=1}^{N} \langle j_k|\, O_k\, |i_k\rangle \ .$$

Using $Tr\,(A\otimes B) = Tr\,(A)\,Tr\,(B)$ and $(A_1\otimes B_1)\,(A_2\otimes B_2) = A_1A_2\otimes B_1B_2$ we obtain

$$\begin{aligned}
\langle\psi|\, O\, |\psi\rangle &= \sum_{i_1,\ldots,i_N=1}^{d} \sum_{j_1,\ldots,j_N=1}^{d} Tr\left[\prod_{k=1}^{N} \left(A\,[k]^{i_k}\otimes \overline{A}\,[k]^{j_k}\right)\right] \prod_{k=1}^{N} \langle j_k|\, O_k\, |i_k\rangle \\
&= Tr\left[\sum_{i_1,\ldots,i_N=1}^{d} \sum_{j_1,\ldots,j_N=1}^{d} \prod_{k=1}^{N} \langle j_k|\, O_k\, |i_k\rangle \left(A\,[k]^{i_k}\otimes \overline{A}\,[k]^{j_k}\right)\right] \\
&= Tr\left[\prod_{k=1}^{N} \sum_{i_k,j_k=1}^{d} \langle j_k|\, O_k\, |i_k\rangle \left(A\,[k]^{i_k}\otimes \overline{A}\,[k]^{j_k}\right)\right] \\
&= Tr\,(E_{O_1}\cdots E_{O_N}) \ .
\end{aligned}$$

Here we have used the fact that the trace, the sum over the sites and the product commute. □

As a special case of the above theorem, we obtain a formula for the norm of a MPS by setting $O_k = \mathbb{1}$ for all k. In this case, the above theorem gives

$$\langle\psi|\psi\rangle = Tr\,(E_1\cdots E_N) \ . \tag{3.10}$$

In the following chapters we will need some further properties of the transfer matrices E, corresponding to the unit operator $\mathbb{1}$, which we will derive now.

Consider a TI MPS with matrices A^i and bond dimension D. The following calculation will work for MPS in OBC as well, but let us forget about the indices for a moment. Let E be the corresponding transfer matrix

$$E = \sum_{i}^{d} A^i\otimes \overline{A}^i \ . \tag{3.11}$$

Now consider a vector $|r\rangle \in \mathbb{C}^{D^2}$. We can write the multiplication of $|r\rangle$ with E in the following

way

$$\begin{aligned}(E\,|r\rangle)_{(\alpha\alpha')} &= \sum_{(\beta\beta')} E_{(\alpha\alpha')(\beta\beta')}\,|r\rangle_{(\beta\beta')} \\ &= \sum_{(\beta\beta')}\sum_{i} A^{i}_{\alpha\beta}\overline{A}^{i}_{\alpha'\beta'}\,|r\rangle_{(\beta\beta')} \\ &= \sum_{i}\sum_{(\beta\beta')} A^{i}_{\alpha\beta}\,|r\rangle_{(\beta\beta')}\,A^{i\dagger}_{\beta'\alpha'} \\ &= \left(\sum_{i} A^{i} R A^{i\dagger}\right)_{(\alpha\alpha')} ,\end{aligned}$$

with $R \in \mathbb{C}^{D\times D}$ being a matrix with the elements of $|r\rangle$ as components indicated by the indices above. For a left vector $\langle l|$ we can do this in an analogous way to obtain

$$\begin{aligned}(\langle l|\,E)_{(\beta\beta')} &= \left(\sum_{i} A^{i\dagger} L^{\dagger} A^{i}\right)_{(\beta\beta')} \\ &= \lambda L^{\dagger}_{(\beta\beta')} ,\end{aligned}$$

with $L \in \mathbb{C}^{D\times D}$ being the matrix corresponding to $\langle l|$.

Therefore the multiplication of E with a vector can be rewritten as an application of the map $\mathcal{E} : \mathbb{C}^{D\times D} \to \mathbb{C}^{D\times D}$ given by

$$\mathcal{E}\,(X) = \sum_{i} A^{i} X A^{i\dagger} , \tag{3.12}$$

to the reshaped vector. For the left vector we can define a similar map, as shown above. A map of the above form on finite dimensional spaces is called a **completely positive map** (CP-Map) [Cho75] with so called Kraus operators A^i. Therefore we can translate certain properties of the transfer matrix E into the setting of CP-Maps.

For example an eigenvector with eigenvalue $\lambda \in \mathbb{C}$ in the setting of the transfer-matrix corresponds via

$$(E\,|r\rangle)_{(\alpha\alpha')} = \left(\sum_{i} A^{i} R A^{i\dagger}\right)_{(\alpha\alpha')} \tag{3.13}$$

$$= \lambda R_{(\alpha\alpha')} \tag{3.14}$$

to eigenvectors of $\mathcal{E}$. The above expression shows, that for eigenvalue $\lambda = 1$ the eigenvector corresponds to a fixed point of the CP-Map $\mathcal{E}$. In the infinite case the MPS is normalized if and only if the dominant eigenvalue of E is equal to 1. Therefore the fixed points of the corresponding CP-Map will play an important role.

Fixed points of CP-Maps have nice properties, which will become important, when we develop

the algorithm. For example it can be shown that a fixed point R of such a map is positive semidefinite [EHK77] and therefore also hermitian. We will use this property later.

Note that we have freedom in the choice of the matrices for a given MPS. If we insert an invertible matrix M and its inverse in between any pair of sites by defining $A[k]^i \to A[k]^i M$ and $A[k+1]^i \to M^{-1}A[k+1]^i$, we obtain the same MPS and thus the same state. We will call such a transformation of the MPS a **gauge transformation**. In order to give a unique representation, we can use the correspondence to CP-Maps to define a gauge condition for the MPS, which we will call the canonical form.

Theorem 4. (*See [PGVWC07]*)
Any state $|\psi\rangle \in \mathcal{H}$ can be represented as a MPS in OBC with matrices $A[k]^i$ fulfilling the following gauge conditions for all k

1. $\sum_{i=1}^{d} A[k]^i A[k]^{i\dagger} = \mathbb{1}_{D_k}$
2. $\sum_{i=1}^{d} A[k]^i \Lambda[k-1] A[k]^{i\dagger} = \Lambda[k]$
3. $\Lambda[k] \in \mathbb{C}^{D_{m+1} \times D_{m+1}}$ *is diagonal, positive, has full rank and satisfies* $Tr(\Lambda[k]) = 1$

*A state fulfilling the above conditions is said to be in **canonical form**. The canonical form is unique up to permutations in the Schmidt coefficients.*

We note, that the conditions of the canonical form correspond to certain conditions of fixed points of the CP-Map $\mathcal{E}$ defined above. As shown in [Per05] the first condition for the canonical form implies that the state represented by the MPS is normalized. Therefore it will turn out useful in the algorithms to obtain the canonical form of the MPS in order to be sure that the state is normalized. Furthermore we can easily compare two given MPS in canonical form, because it is unique.

Theorem 5. (*See [PGVWC07]*)
Let $|\psi\rangle$ be a MPS in OBC of the form

$$|\psi\rangle = \sum_{i_1,\ldots,i_N} A[1]^{i_1} A[2]^{i_2} \cdots A[N-1]^{i_{N-1}} A[N]^{i_N} |i_1, i_2, \ldots, i_N\rangle \ , \tag{3.15}$$

representing a normalized state.

There exist, not necessarily square, matrices L_j, R_j with $R_j L_j = \mathbb{1}$ for $j \in \{1, \ldots, N-1\}$, such that the gauge transformation

$$B[1]^i = A[1]^i R_1, B[N]^i = L_{N-1} A[N]^i \tag{3.16}$$

$$B[k]^i = L_{k-1} A[k]^i R_k, \qquad \text{if } 2 \leq k \leq N-2 \tag{3.17}$$

will establish the canonical form of the state.

3.2. Matrix Product Operators and Tensor Diagrams

After introducing the basic formalism of MPS in open boundary conditions, we will now develop a formalism for the operators acting on the states of a one dimensional system written as a MPS. In the following we will consider operators acting on the Hilbert space of a finite chain $\mathcal{H} = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_N$, where the spaces H_k all have the same dimension $d < \infty$.

Definition 3. (*See* [*PGVWC07*])

1. *A* ***Matrix Product Operator*** (MPO) *in* ***periodic boundary condition*** (PBC) *is an operator* $O : \mathcal{H} \to \mathcal{H}$ *of the form*

$$O = \sum_{i_1,\ldots,i_N=1}^{d} Tr\left[C\left[1\right]^{i_1} C\left[2\right]^{i_2} \cdots C\left[N\right]^{i_N}\right] X_1^{i_1} \otimes X_2^{i_2} \otimes \cdots \otimes X_N^{i_N} \ , \tag{3.18}$$

   where $C\left[k\right]^i$ *is a* $D_k \times D_{k+1}$ *matrix corresponding to the operator* O_k *acting on site* $k \in \{1, \ldots, N\}$. $D = \max_n D_n$ *is called the bond dimension of the MPO.*

2. *If* $D_1 = D_{N+1} = 1$ *the MPO is said to be with* ***open boundary conditions*** (OBC).

3. *We call a MPO in periodic boundary conditions with site-independent matrices,* $C\left[k\right]^i = C^i$ *for all* k, ***translationally invariant*** (TI).

Note that the concept of MPO is the same as MPS, but on a different space, namely the vector space of linear operators. We therefore obtain similar results.

Theorem 6. *Every operator* $O : \mathcal{H} \to \mathcal{H}$ *is a MPO in open boundary conditions with bond dimension* $D \leq d^N$.

Proof. As the space $L\left(\mathcal{H}\right) = \{O : \mathcal{H} \to \mathcal{H} : O \ \ linear\}$ is a vector space and is also the tensor product of the spaces $L\left(\mathcal{H}_k\right)$ with dimensions d^2, we can use the same construction as in Theorem 2. □

In fact the construction in the proof of Theorem 6 is not very useful, as there are often better ways to construct an equivalent MPO representation than the way used to construct a MPS. From numerical considerations it is often convenient if the matrices of the MPO representation have special properties, such as symmetry or translational invariance. We will see later that we can obtain MPO representations with such properties for time-evolution operators arising from Hamilton operators of quantum spin chains. Next we will see how the MPS and the MPO formalism fit together.

Theorem 7. (*See* [*MCPV08*])

Given a MPS $|\psi\rangle \in \mathcal{H}$ *of the form* (3.1) *with bond dimension* D *and a MPO* $O : \mathcal{H} \to \mathcal{H}$ *of the form* (3.18) *with bond dimension* K*, then the state* $O\left|\psi\right\rangle$ *is a MPS with bond dimension*

$\tilde{D} = DK$ *and matrices*

$$\tilde{A}[n]^i = \sum_{j,k} C[n]^j \otimes A[n]^k \langle i| X_n^j |k\rangle \ . \tag{3.19}$$

Proof. Following the same calculations as in the proof of Theorem 3 and inserting a

$$\mathbb{1} = \sum_{k_1,\ldots,k_N} |k_1, k_2, \ldots, k_N\rangle \langle k_1, k_2, \ldots, k_N| \ ,$$

to get the basis right. The new bond dimension is clear after obtaining the new matrices. □

In order to alleviate the calculations we can contract the tensors C and X at the j bound to obtain $\tilde{C}^{i,k} = \sum_j C^j \langle i| X^j |k\rangle$. This allows us to write (3.19) in a shorter way

$$\tilde{A}[n]^i = \sum_k \tilde{C}[n]^{i,k} \otimes A[n]^k \ . \tag{3.20}$$

For the algorithms we will use the MPO representation of time-evolution operators in order to simulate 1-dimensional quantum many-body systems. In this case we only need the tensors $A[n]$ in the MPS representation of an initial state and the $\tilde{C}[n]$ in the MPO representation of the time-evolution operator in order to describe the evolution of the system. To visualize the methods and alleviate the understanding of calculations in this formalism we can use a graphical representation in form of tensor diagrams. Therefore we only look at the coefficients, necessary to describe the occurring states and operators, and not at the whole state. In these diagrams we will represent tensors corresponding to a MPS by squares and the ones corresponding to a MPO by circles. Each leg in the diagram corresponds to an index of a tensor. If two tensors are connected by a leg in the diagram, this means that one has to contract these tensors along the shared index represented by the leg. In the diagram vertical legs correspond to physical indices of dimension d and horizontal legs to auxiliary indices with dimension equal to the bond dimension of the MPS or MPO. As another convention we will draw the MPS representing $|\psi\rangle \in \mathcal{H}$ with legs pointing up and the MPS representing $\langle\psi| \in \mathcal{H}'$ with legs pointing down (See Fig.3.1 and Fig.3.2).

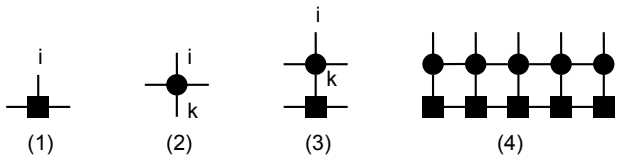

Figure 3.1.: (1) A tensor corresponding to a single site of a MPS. The index i has dimension d. The other two indices have the bond dimension. (2) A tensor corresponding to a local MPO. (3) A local MPO acting on a tensor of a MPS corresponding to one site. There is a contraction along index k with dimension d. (4) A MPS in OBC for a system of 5 particles after applying a MPO. Each line connecting two tensors corresponds to a contraction.

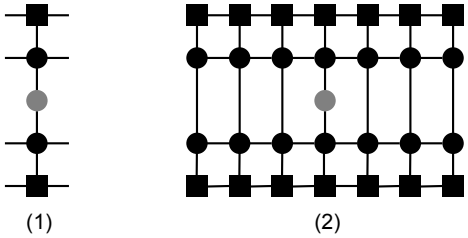

Figure 3.2.: (1) Transfer matrix of a local operator acting on one tensor of a MPS after the application of a MPO. (2) The expectation value of a local operator.

3.3. Infinite Matrix Product States

We will now switch to a system of infinite size. Let $|\psi\rangle \in \mathcal{H}$, where $\mathcal{H}$ is now the Hilbert space of an infinite 1-dimensional system and has the form $\mathcal{H} = \bigotimes_{k\in\mathbb{Z}} H_k$, where H_k is the d-dimensional Hilbert space corresponding to one site of the system. We will only look at TI states of such systems, because it is impossible to simulate a system in an efficient way, if the amount of information we have to store is infinite. We will see later that this is actually not so restrictive, because many systems of physical interest are translationally invariant.

Definition 4. *An **infinite MPS** ($iMPS$) with bond dimension D is a pure TI state $|\psi\rangle \in \mathcal{H}$ represented by site-independent $D \times D$ matrices A^i, for which the transfer matrix, see Theorem 3, has 1 as a non-degenerate eigenvalue and all other eigenvalues have modulus smaller than 1.*

This definition has to be read in the sense of a limit of finite TI-MPS. It is in fact not trivial to extend finite MPS to the infinite limit. That this is possible in a mathematical way has been shown by Fannes, et al. [FNW92]. They use CP-Maps on more general algebras as the starting point, while we use the transfer matrix with certain restrictions to the eigenvalues. Their approach is of course more general, but as we only work on special Hilbert spaces, we can define iMPS in the above way.

The conditions stated in the definition ensure that the limit is well-defined and that we can calculate expectation values. In this sense the given definition suffices for our purposes and most of the results we obtained for finite systems are still valid in the infinite case. The MPO formalism can be carried over to the infinite case by introducing translationally invariant infinite MPOs [OV08]. We will see later that we can express the time-evolution operators of the models we are going to observe in a translationally invariant form. The formula to apply a MPO to a MPS stays the same in the infinite case [MCPV08]. To calculate expectation values of infinite MPOs or of local operators at a finite number of sites, we have to be careful as the calculation of expectation values corresponds to the contraction of an infinite tensor network, which in our case is well-defined as we see in the following.

Theorem 8. *For an iMPS $|\psi\rangle \in \mathcal{H}$ with matrices A^i for $i \in \{1, \ldots, d\}$ and for any local*

observable $O : H_k \to H_k$ *the limit*

$$\langle\psi| O |\psi\rangle = \lim_{k\to\infty} Tr\left(E^k E_O E^k\right) \tag{3.21}$$

exists and gives the expectation value, where E_O *is the transfer matrices corresponding to the operator O, see Theorem* (3)*. Here the expression* $\langle\psi| O |\psi\rangle$ *for a local operator O is a shorthand for* $\langle\psi| \cdots \otimes \mathbb{1} \otimes O \otimes \mathbb{1} \otimes \cdots |\psi\rangle$

Proof. Expression (3.21) is the limit of the formula to compute the expectation value in the finite case.
For the proof of the existence of the limit we use the theorem of the power method, see Theorem 5.6 in [QSS07]. As in our case the leading eigenvalue of the matrix E is non-degenerated, equal to 1 and all other eigenvalues have smaller modulus, we get $E^k \to |R\rangle\langle L|$ for $k \to \infty$. Here $|R\rangle$ and $\langle L|$ denote the right and left eigenvector corresponding to the maximal eigenvalue 1. □

The proof of Theorem 8 leads to a method for calculating such expectation values directly by computing the right and left eigenvector $|R\rangle$ and $\langle L|$ with the leading eigenvalue 1 and inserting them into the expression (3.21) as the limit of the matrix powers. Using the properties of the trace, we get the expectation value

$$\langle\psi| O |\psi\rangle = \langle L| E_O |R\rangle \ . \tag{3.22}$$

The condition that 1 is the maximal eigenvalue implies that an iMPS in our definition is normalized. Now remember the connection between the transfer matrix E and the CP-Map $\mathcal{E} : \mathbb{C}^{D\times D} \to \mathbb{C}^{D\times D}$ given by

$$\mathcal{E}(X) = \sum_i A^i X A^{i\dagger} \ . \tag{3.23}$$

We have seen, that eigenvectors to eigenvalue 1 of the transfer matrix E correspond to fixed points of the map $\mathcal{E}$. The importance of the results on fixed points of CP-Maps become more apparent now.

As for finite MPS, we can now define a gauge condition for iMPS, which leads to the canonical form of iMPS. We want to use

$$\sum_{i=1}^{d} A^i A^{i\dagger} = \mathbb{1}_D \ , \tag{3.24}$$

as the canonical form. This canonical form has several advantages. For example the calculation of expectation values is easy, as we know the leading eigenvector of the transfer matrix to be the vector, that corresponds to the reshaped identity matrix $\mathbb{1}$. Also it will improve the stability of the algorithm. There are other possibilities for the canonical form, see [OV08].

As we will see later it is always possible to write an iMPS in canonical form. This is more difficult compared to the finite case, because we cannot use the boundary of a finite chain. We can only use gauge transformations, i.e. the insertion of terms like $\mathbb{1} = XX^{-1}$ in the states

in a way that the translational invariance is preserved. In the next section we will describe algorithms for obtaining the canonical form of an iMPS in an efficient way.

As an example we show, how an iMPS from a state can be explicitly constructed. We will follow the construction given by Vidal [Vid07]. Start with a pure translationally invariant state $|\psi\rangle \in \mathcal{H}$ on a spin chain. The Schmidt decomposition of the chain in two parts at the site r has the form

$$|\psi\rangle = \sum_{\alpha=1}^{D} \lambda_\alpha^{[r]} \left|\Phi_\alpha^{[\lhd r]}\right\rangle \left|\Phi_\alpha^{[r+1\rhd]}\right\rangle \ . \tag{3.25}$$

Here $\left|\Phi_\alpha^{[\lhd r]}\right\rangle$ and $\left|\Phi_\alpha^{[r+1\rhd]}\right\rangle$ denote orthonormal basis vectors of the left and the right sublattice. By applying the Schmidt decomposition we have assumed that the Schmidt rank D is finite. This is needed if we want to describe the state in the valence bond picture with finite interaction at one bond. We can express these basis vectors through the basis vectors of one site and basis vectors of shifted semi-infinite sublattices and obtain

$$\left|\Phi_\alpha^{[\lhd r]}\right\rangle = \sum_{i=1}^{d}\sum_{\beta=1}^{D} \lambda_\beta^{[r-1]} \Gamma_{i,\beta,\alpha}^{[r]} \left|\Phi_\beta^{[\lhd r-1]}\right\rangle \left|i^{[r]}\right\rangle \ , \tag{3.26}$$

$$\left|\Phi_\alpha^{[r+1\rhd]}\right\rangle = \sum_{i=1}^{d}\sum_{\beta=1}^{D} \Gamma_{i,\beta,\alpha}^{[r+2]} \lambda_\beta^{[r+2]} \left|\Phi_\beta^{[\lhd r-1]}\right\rangle \left|i^{[r+1]}\right\rangle \ . \tag{3.27}$$

By inserting (3.26) in (3.25) we obtain

$$|\psi\rangle = \sum_{i=1}^{d}\sum_{\alpha,\beta=1}^{D} \lambda_\beta^{[r-1]} \lambda_\alpha^{[r]} \Gamma_{i,\beta,\alpha}^{[r]} \left|\Phi_\alpha^{[\lhd r-1]}\right\rangle \left|i^{[r]}\right\rangle \left|\Phi_\beta^{[r+1\rhd]}\right\rangle \ . \tag{3.28}$$

For extending the decomposition in the other direction we use (3.27). But this is not necessary since we can use the translational invariance to see that all the tensors occurring in (3.28) are site independent. Therefore we can set $A_{\alpha,\beta}^{i} = \lambda_\alpha^{[r]} \Gamma_{i,\beta,\alpha}^{[r]}$ and obtain the final iMPS.

Within our algorithms it will be useful to think of one site of the system in a more general way. In the infinite case we can form equal-sized blocks of neighboring physical sites to obtain a translationally invariant system. With this technique it is possible to deal with periodic forms of iMPS.

An example would be the case where we take p different matrices and repeat them periodically over the chain. These states are no iMPS according to Definition 4, but by contracting blocks of p neighboring physical sites together we can solve this problem and obtain an iMPS on the chain build up by blocks of size p. For each local observable O acting on one physical site we get a corresponding operator by taking the mean over all possible sites it can act on. We obtain the operator

$$\frac{1}{p}\left[O \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \mathbb{1} \otimes O \otimes \cdots \otimes \mathbb{1} + \cdots + \mathbb{1} \otimes \cdots \otimes \mathbb{1} \otimes O\right] \ , \tag{3.29}$$

which has to be applied to a block of size p. In order to calculate expectation values with Theorem 8 one has to take these operators instead of the ones acting on physical sites. We will use these more general iMPS later to build certain periodicities inside the iMPS.

4. Time Evolving Block Decimation and its application to Ground State Calculation

In this chapter we will state an efficient algorithm for the simulation of 1-dimensional quantum many-body systems using the matrix product formalism introduced in the previous chapter. This algorithm is the Time Evolving Block Decimation ($TEBD$) developed by Vidal [Vid03]. Later we will describe changes to the original algorithm, which leads to an implementation for longer range interaction.

We want to consider a general quantum system with corresponding Hilbert space $\mathcal{H}$ first. Let $H : \mathcal{H} \to \mathcal{H}$ be the Hamiltonian of the system. To calculate the time evolution of such a system one has to solve the Schrödinger equation

$$i\hbar \frac{\partial}{\partial t} |\psi(t)\rangle = H |\psi(t)\rangle \tag{4.1}$$

$$|\psi(0)\rangle = |\psi_0\rangle \ . \tag{4.2}$$

This equation is equivalent to the operator equation

$$i\hbar \frac{\partial}{\partial t} U(t) = HU(t) \ , \tag{4.3}$$

for a time dependent operator $U(t) : \mathcal{H} \to \mathcal{H}$ with the property

$$|\psi(t)\rangle = U(t) |\psi_0) \ . \tag{4.4}$$

The operator U is called the **Time-Evolution Operator** of the system. If H is time independent the time-evolution operator takes the form

$$U(t) = \exp\left(-\frac{i}{\hbar} Ht\right) \ . \tag{4.5}$$

To obtain the time-evolution of a quantum system one has to apply the time evolution operator to the initial state of the system. There are some problems in using this technique for the simulation of big many-body quantum systems with time independent Hamiltonians. The first problem is that it is not clear how to calculate the matrix exponential of a Hamiltonian acting on an infinite system. If we can calculate the time evolution operator the second problem is to apply it efficiently to a state of the system. In the following we will use the methods developed in the first chapter to solve these problems and thereby obtain an algorithm for simulating the time evolution of a infinite 1-dimensional many-body quantum system.

4.1. Trotter Decomposition

First we have to introduce the Trotter Decomposition which approximates the exponential of a sum of non-commuting operators as a product of exponentials. This is a fundamental tool used in the following.

Theorem 9. *Let $A, B : \mathcal{H} \to \mathcal{H}$ for a separable Hilbert space $\mathcal{H}$ be hermitian operators. Then*

$$\exp\left(i\left(A+B\right)t\right) = \left(\exp\left(\frac{iAt}{n}\right)\exp\left(\frac{iBt}{n}\right)\right)^n + \mathcal{O}\left(\frac{1}{n}\right) \ , \tag{4.6}$$

for all $t \in \mathbb{R}$.

Proof. We will follow the proof of [NC07]. The matrix exponential has the form

$$\exp\left(\frac{iAt}{n}\right) = \mathbb{1} + \frac{iAt}{n} + \mathcal{O}\left(\frac{1}{n^2}\right) \ . \tag{4.7}$$

Here we have to show that the definition of the matrix exponential of a hermitian operator by the above power series is valid, i.e. that $\exp\left(\frac{iAt}{n}\right)|\psi\rangle$ is defined for $|\psi\rangle \in \mathcal{H}$. We denote by $\{|k\rangle\}$ the orthonormal basis of $\mathcal{H}$ built out of eigenvectors of A corresponding to eigenvalues λ_k. This ONB exists because A is hermitian. For $|\psi\rangle = \sum_{k\in\mathbb{N}} \mu_k |k\rangle$ we can write

$$\begin{aligned}
\exp\left(\frac{iAt}{n}\right)|\psi\rangle &= \sum_{n\in\mathbb{N}} \frac{(iAt)^n}{n!} \sum_{k\in\mathbb{N}} \mu_k |k\rangle \\
&= \sum_{n\in\mathbb{N}} \sum_{k\in\mathbb{N}} \frac{(it\lambda_k)^n}{n!} \mu_k |k\rangle \\
&= \sum_{k\in\mathbb{N}} \mu_k \exp\left(\frac{i\lambda_k t}{n}\right)|k\rangle \in \mathcal{H} \ .
\end{aligned}$$

Here we have used that $\left|\exp\left(\frac{i\lambda_k t}{n}\right)\right| = 1$ for all $k \in \mathbb{N}$.

Using (4.7) for A and B, we obtain

$$\exp\left(\frac{iAt}{n}\right)\exp\left(\frac{iBt}{n}\right) = \mathbb{1} + \frac{i\left(A+B\right)t}{n} + \mathcal{O}\left(\frac{1}{n^2}\right) \ . \tag{4.8}$$

Taking the n-th power shows

$$\left(\exp\left(\frac{iAt}{n}\right)\exp\left(\frac{iBt}{n}\right)\right)^n = \mathbb{1} + \sum_{k=1}^{n} \binom{n}{k} \frac{1}{n^k} \left[i\left(A+B\right)t\right]^k + \mathcal{O}\left(\frac{1}{n}\right) \ . \tag{4.9}$$

With $\binom{n}{k}\frac{1}{n^k} = \left(1 + \mathcal{O}\left(\frac{1}{n}\right)\right)/k!$ we get

$$\left(\exp\left(\frac{iAt}{n}\right)\exp\left(\frac{iBt}{n}\right)\right)^n = \sum_{k=0}^{n}\frac{\left[i\left(A+B\right)t\right]^k}{k!}\left(1+\mathcal{O}\left(\frac{1}{n}\right)\right)+\mathcal{O}\left(\frac{1}{n}\right) \tag{4.10}$$

$$= \exp\left(i\left(A+B\right)t\right) - \sum_{k=n}^{\infty}\frac{\left[i\left(A+B\right)t\right]^k}{k!} + \mathcal{O}\left(\frac{1}{n}\right) \tag{4.11}$$

$$= \exp\left(i\left(A+B\right)t\right) + \mathcal{O}\left(\frac{1}{n}\right) \ . \tag{4.12}$$

□

We have proved the above theorem for the special case of hermitian operators on a separable Hilbert space. The theorem is also true in a more general formulation proved by Trotter [Tro59]. For example, the above decomposition is also true if $t \in i\mathbb{R}$ and the spectrum of A and B is bounded from below.

For the numerics it is useful to state some similar formulas of higher order introduced by M. Suzuki [Suz84].

Theorem 10. (*See [Suz84]*)
For any hermitian operators A and B on a separable Hilbert space and $t \in \mathbb{R}$

$$\exp\left(i\left(A+B\right)t\right) = \left(\exp\left(\frac{iAt}{2n}\right)\exp\left(\frac{iBt}{n}\right)\exp\left(\frac{iAt}{2n}\right)\right)^n + \mathcal{O}\left(\frac{1}{n^2}\right) \tag{4.13}$$

To obtain further formulas of higher order one can use a technique proposed by M. Suzuki in [Suz90], which leads to the so called fractal decomposition. If we want to compute for example a third order decomposition for $\exp\left(i\left(A+B\right)t\right)$, we set $S\left(t\right) = \exp\left(iAt/2\right)\exp\left(iBt\right)\exp\left(iAt/2\right)$. This is the second order approximant. Then for a parameter $s \in \mathbb{R}$, we get

$$\exp\left(i\left(A+B\right)t\right) = \exp\left(i\left(A+B\right)st\right)\exp\left(i\left(A+B\right)\left(1-2s\right)t\right)\exp\left(i\left(A+B\right)st\right) \ . \tag{4.14}$$

Now we can substitute the second order approximant $S\left(t\right)$in in each factor of the last equation to obtain

$$S_3\left(t\right) = S\left(st\right)S\left(\left(1-2s\right)t\right)S\left(st\right) \ . \tag{4.15}$$

It is now possible to choose the s in a way, that the third order terms in the above equation vanishes. It can be shown that this happens for $s = \frac{1}{2-\sqrt[3]{2}}$ [Suz90]. This method can be used recursively to obtain approximants of higher order by using the general ansatz,

$$S_m\left(t\right) = S_{m-1}\left(s_m t\right)S_{m-1}\left(\left(1-2s_m\right)t\right)S_{m-1}\left(s_m t\right) \ , \tag{4.16}$$

for the m-th order approximant S_m. For more details on this method and a discussion of the fractal structure of these decompositions see, e.g. [Suz90].

4.2. Time Evolving Block Decimation

In the following we will focus on infinite systems, and introduce the infinite Time Evolving Block Decimation ($iTEBD$) algorithm. This algorithm can be used to simulate infinite 1-dimensional quantum many-body systems. An algorithm for finite systems in PBC can obtained in a similar way [PWE10].

Consider an infinite system as in chapter 1, i.e. a system composed of subsystems with corresponding Hilbert spaces $\mathcal{H}_k$. The full system is required to be translationally invariant as explained in the previous chapter such that the corresponding Hilbert space of the full system has the form $\mathcal{H} = \bigotimes_{k\in\mathbb{Z}} \mathcal{H}_k$ where $\mathcal{H}_k = \mathcal{H}_m = \mathcal{H}_{loc}$ for all $k, m \in \mathbb{Z}$. We will make some assumptions on the Hamiltonian of the system. We will first consider a Hamiltonian of the form

$$H = \sum_{k\in\mathbb{Z}} h^{[k,k+1]} \ . \tag{4.17}$$

Here $h^{[k,k+1]}$ is shorthand for $(\otimes_{l<k}\mathbb{1}_l) \otimes h \otimes (\otimes_{l>k+1}\mathbb{1}_l)$ where $h : H_{loc} \otimes H_{loc} \to H_{loc} \otimes H_{loc}$ denotes a hermitian operator corresponding to a nearest-neighbor interaction of the 1-dimensional system. This interaction is the same for each pair of sites. It is not necessary to assume this special form of the Hamiltonian, but it simplifies the calculation, if we assume only two-body terms. In order to use iMPS it is necessary that the time-evolution operator preserves the form of the iMPS. Therefore we need some kind of translational invariance. We could have also used invariance under translation of any finite number k of sites. Then we would take blocks of k physical sites together, to make the system translational invariant. We will see some examples in Chapter 5.

Suppose we want to compute the time-evolution of an initial state $|\psi_0\rangle$, which can be written as an iMPS with bond dimension D_0. For calculating the state after a time t using the time-evolution operator, one has to compute

$$|\psi(t)\rangle = \exp\left(-\frac{i}{\hbar}Ht\right)|\psi_0\rangle \tag{4.18}$$

$$= \exp\left(-\frac{i}{\hbar}\sum_{k\in\mathbb{Z}} h^{[k,k+1]}t\right)|\psi_0\rangle \ . \tag{4.19}$$

We will use the matrix product formalism to compute the above expression efficiently. The idea is to restrict the set of states, which are used during the simulation to iMPS with bond dimension D depending on the computational power available. For these states the representation is efficient. We have to express the time-evolution operator in such a way that it can be applied to the matrices of an iMPS directly. This can be achieved by using the Trotter-decomposition for this step and writing the time-evolution operator as an iMPO. After applying the time-evolution operator to the state we obtain an iMPS, which has in general a bigger bond dimension than the initial iMPS. To stay efficient we have to approximate this iMPS by an iMPS with the bond dimension D. A sketch of the algorithm is the following:

1. *The initial state is an iMPS with bond dimension D_0*

2. *Use the Trotter-decomposition to decompose the time-evolution operator in such a way that it can be applied to the matrices of the iMPS.*

3. *Apply the time-evolution operator to the iMPS. As we will see, the bond dimension of the resulting iMPS is in general bigger than the original one.*

4. *Approximate the state in 3. by an iMPS with bond dimension D, which is a chosen maximal dimension.* (**Truncation**)

By repeating the steps 3. and 4. we obtain an approximation of the time-evolution of the state.

In the following sections we will discuss the steps of the algorithm in detail. Remember that we have discussed step 1. already in Chapter 2.

In order to apply the algorithm, the system has to satisfy some conditions. Firstly the time-evolution must preserve the translational invariance. This is guaranteed if the Hamiltonian of the system is translationally invariant. Secondly we require the time-evolution operator to be decomposable in a way that it can be applied to an iMPS directly. In the next sections we will discuss some ways to achieve this.

4.2.1. Decomposition of the Time-Evolution Operator: Even-Odd-Ansatz

In this section we explain how to do the step 2 in the algorithm presented before, i.e. the decomposition of the time-evolution operator. We will express the time-evolution operator as a composition of local operations which can be applied to the iMPS directly. There are different approaches to do this decomposition. We will first discuss the original approach stated by Vidal [Vid07] in the first formulation of the iTEBD method and second a decomposition stated in [MCPV08], which improves the first method in some cases, using the MPO formalism.

To follow the idea of Vidal, we first remember that we are dealing with a Hamiltonian of the form

$$H = \sum_{k\in\mathbb{Z}} h^{[k,k+1]} \ , \tag{4.20}$$

with hermitian operators $h^{[k,k+1]}$, which are the same for each pair $\{k, k+1\}$. The corresponding time-evolution operator has the form

$$U(t) = \exp\left(-\frac{i}{\hbar}\sum_{k\in\mathbb{Z}} h^{[k,k+1]}t\right) \ . \tag{4.21}$$

If we evaluate the time-evolution operator, we have the problem that in general

$$\left[h^{[k-1,k]}, h^{[k,k+1]}\right] \neq 0 \ . \tag{4.22}$$

Therefore we cannot write the time-evolution operator directly as a product of local operators. To compute the decomposition we write the Hamiltonian as a sum of two terms

$$H = H_e + H_o \ , \tag{4.23}$$

with an even part $H_e = \sum_{k\in\mathbb{Z}} h^{[2k,2k+1]}$ and an odd part $H_o = \sum_{k\in\mathbb{Z}} h^{[2k-1,2k]}$. The time-evolution operator then has the form

$$U(t) = \exp\left(-\frac{i}{\hbar}\left(H_e + H_o\right)t\right) \ . \tag{4.24}$$

We can use the first order trotter decomposition(Theorem 9), because H_e and H_o are hermitian operators, and get

$$U(t) = \left(\exp\left(-\frac{iH_e t}{n}\right)\exp\left(-\frac{iH_o t}{n}\right)\right)^n + \mathcal{O}\left(\frac{1}{n}\right) \ . \tag{4.25}$$

We could have used a Trotter-decomposition of a different order at this step, but then the method would work the same. The above expression means that instead of applying $U(t) = \exp\left(-\frac{i}{\hbar}\sum_{k\in\mathbb{Z}} h^{[k,k+1]}t\right)$ once, we decompose $[0,t]$ into n discrete steps $\left[\frac{mt}{n}, \frac{(m+1)t}{n}\right]$ and apply the time-evolution operators corresponding to H_e and H_o for the time-step $\frac{t}{n}$ n times.
Using

$$\left[h^{[2k-1,2k]}, h^{[2k+1,2k+2]}\right] = 0 \tag{4.26}$$

$$\left[h^{[2k,2k+1]}, h^{[2k+2,2k+3]}\right] = 0 \ , \tag{4.27}$$

because the operators are acting on different spaces, we can write the components of the above decomposition as a product of 2-site operators. We get

$$U_e\left(\frac{t}{n}\right) = \exp\left(-\frac{iH_e t}{n}\right) = \prod_{k\in\mathbb{Z}} \exp\left(-\frac{ih^{[2k,2k+1]}t}{n}\right) \ , \tag{4.28}$$

$$U_o\left(\frac{t}{n}\right) = \exp\left(-\frac{iH_o t}{n}\right) = \prod_{k\in\mathbb{Z}} \exp\left(-\frac{ih^{[2k-1,2k]}t}{n}\right) \ . \tag{4.29}$$

We can finally use Theorem 6 to obtain a MPO-representation of the 2-site operators

$$\exp\left(-\frac{ih^{[2k,2k+1]}t}{n}\right) = \sum_{i_1,i_2}^{d} Tr\left[C[1]^{i_1} C[2]^{i_2}\right] X_{2k}^{i_1} \otimes X_{2k+1}^{i_2} \tag{4.30}$$

$$\exp\left(-\frac{ih^{[2k-1,2k]}t}{n}\right) = \sum_{i_1,i_2}^{d} Tr\left[C[1]^{i_1} C[2]^{i_2}\right] X_{2k-1}^{i_1} \otimes X_{2k}^{i_2} \tag{4.31}$$

in the above expression. Note that the k dependence is due to the fact that we apply the same

operators to different sites.

If we put everything together, we obtain an iMPO representation for the full time-evolution operator, where we used one approximation in the Trotter decomposition. We apply this operator to the iMPS by applying the operator $\left(U_e(\frac{t}{n})U_o(\frac{t}{n})\right)^n$. This can be done by applying the operators (4.30) and (4.31) as MPOs directly to the iMPS (see Fig.4.1). We can therefore calculate the time-evolution of the state.

The described method has disadvantages. In order to apply the time-evolution operator corresponding to the even and odd part one has to break translational invariance (see Fig.4.1). This is often not a big problem as we obtain an iMPS by contracting the matrices corresponding to two neighboring sites together.

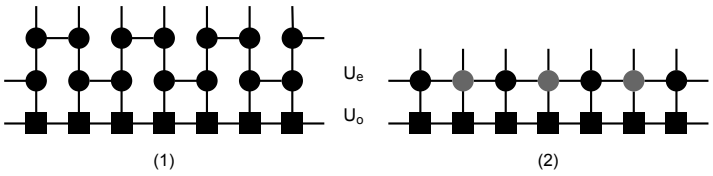

Figure 4.1.: (1) Application of the time-evolution operators corresponding to the even and the odd part of the decomposition. (2) One step contracted.

Another problem of the method is that it is not easy to decompose the time-evolution operator if we have Hamiltonians with longer range interaction. For such interactions we cannot use the even-odd decomposition technique presented above. We will see in the next section, that there is a way to do the decomposition in a better way, which works in many cases.

4.2.2. Decomposition of the Time-Evolution Operator: Translationally Invariant MPO

Here we will present an alternative approach to the decomposition presented in the last section. We will write the time-evolution operator as a translationally invariant MPO. With this representation we do not have to break translational symmetry. This method will not work in general and we have to make some constraints on the Hamiltonian. Following [MCPV08] suppose we have a Hamiltonian of the form

$$H = \sum_{k\in\mathbb{Z}} \left[h_1^{[k,\ldots,k+r_1]} + h_2^{[k,\ldots,k+r_2]} \right] , \tag{4.32}$$

for hermitian operators $h_1^{[k,\ldots,k+r_1]}$ and $h_2^{[k,\ldots,k+r_2]}$, with

$$\left[h_1^{[k,\ldots,k+r_1]}, h_1^{[l,l+r_1]} \right] = \left[h_2^{[k,\ldots,k+r_2]}, h_2^{[l,\ldots,l+r_2]} \right] = 0 , \tag{4.33}$$

for all $k, l \in \mathbb{Z}$, i.e the Hamiltonian is built of commuting terms. We will also assume that the $h_1^{[k,\ldots,k+r_1]}$ are the same for all k acting only on different spaces, and the same for $h_2^{[k,\ldots,k+r_2]}$.

Using the first order Trotter-decomposition we obtain

$$U(t) = \exp\left(-\frac{it}{\hbar}\sum_{k\in\mathbb{Z}}\left[h_1^{[k,\ldots,k+r_1]} + h_2^{[k,\ldots,k+r_2]}\right]\right) \tag{4.34}$$

$$= \left(\exp\left(-\frac{it}{n\hbar}\sum_{k\in\mathbb{Z}} h_1^{[k,\ldots,k+r_1]}\right)\exp\left(-\frac{it}{n\hbar}\sum_{k\in\mathbb{Z}} h_2^{[k,\ldots,k+r_2]}\right)\right)^n + \mathcal{O}\left(\frac{1}{n}\right) \tag{4.35}$$

$$= \left(U_1\left(\frac{t}{n}\right)U_2\left(\frac{t}{n}\right)\right)^n + \mathcal{O}\left(\frac{1}{n}\right)\ . \tag{4.36}$$

We will show how to decompose $U_1\left(\frac{t}{n}\right)$. Of course this works for $U_2\left(\frac{t}{n}\right)$ in the same way and therefore we just write $h^{[k,\ldots,k+r]}$ and leave out the index. In addition we will only show the decomposition for two body operators $h^{[k,k+1]}$. The decomposition works for longer range interaction in the same way. As another shorthand we will write $\epsilon = -\frac{it}{n\hbar}$.

Using the condition (4.33) we can write

$$U_1\left(\frac{t}{n}\right) = \exp\left(\epsilon\sum_{k\in\mathbb{Z}} h^{[k,k+1]}\right) = \prod_{k\in\mathbb{Z}}\exp\left(\epsilon h^{[k,k+1]}\right)\ . \tag{4.37}$$

Now we take the MPO representation of $\exp\left(\epsilon h^{[k,k+1]}\right)$ and build the MPO representation of the full operator out of it. We can write

$$\exp\left(\epsilon h^{[k,k+1]}\right) = \sum_{i,j}\left(B_i^T D_j\right) X_i^{[k]}\otimes X_j^{[k+1]}\ , \tag{4.38}$$

for bases of operators $X_i^{[k]} : H_k \to H_k$ and $X_i^{[k+1]} : H_{k+1} \to H_{k+1}$ and complex vectors B_i and D_i.

We will develop a MPO representation of the above time-evolution operator on a finite chain of N sites in PBC. Therefore we identify the site N+1 with site 1. Using (4.37) we obtain the following:

$$\begin{aligned} U_1\left(\frac{t}{n}\right) &= \prod_{k\in\mathbb{Z}}\exp\left(\epsilon h^{[k,k+1]}\right) \\ &= \sum_{i_1 j_1, i_2 j_2, \ldots, i_N j_N}\left(B_{i_1}^T D_{j_1}\right)\left(B_{i_2}^T D_{j_2}\right)\cdots\left(B_{i_N}^T D_{j_N}\right) X_{j_N}^{[1]} X_{i_1}^{[1]}\otimes X_{j_1}^{[2]} X_{i_2}^{[2]}\otimes\cdots \\ &= \sum_{i_1 j_N, i_2 j_1, \ldots} Tr\left(\left(D_{j_N} B_{i_1}^T\right)\left(D_{j_1} B_{i_2}^T\right)\cdots\left(D_{i_{N-1}} B_{j_N}^T\right)\right) X_{j_N}^{[1]} X_{i_1}^{[1]}\otimes X_{j_1}^{[2]} X_{i_2}^{[2]}\otimes\cdots \quad (\star) \end{aligned}$$

At the last step, we have used the fact that the trace is invariant under cyclic permutations of the arguments. We can express the products of the basis operators $X_j^{[k]} X_i^{[k]}$ in the operator

basis at one site. Therefore we obtain

$$X_j^{[k]} X_i^{[k]} = \sum_l \mu_{ijl} X_l^{[k]} \ , \tag{4.39}$$

where the numbers $\mu_{ijl} \in \mathbb{C}$ are site-independent, because we have assumed all operators to act in the same way on different sites. Therefore we obtain

$$\begin{aligned}(\star) &= \sum_{i_1 j_N l_1, i_2 j_1 l_2, \ldots} Tr\left(\left(D_{j_N} B_{i_1}^T\right)\left(D_{j_1} B_{i_2}^T\right)\cdots\left(D_{i_{N-1}} B_{j_N}^T\right)\right) \mu_{i_1 j_N l_1} X_{l_1}^{[1]} \otimes \mu_{i_2 j_1 l_2} X_{l_2}^{[2]} \otimes \cdots \\ &= \sum_{l_1, l_2, \ldots, l_N} Tr\left(\left(\sum_{i_1 j_N} \mu_{i_1 j_N l_1} D_{j_N} B_{i_1}^T\right)\left(\sum_{i_2 j_1} \mu_{i_2 j_1 l_2} D_{j_1} B_{i_2}^T\right)\cdots\right) X_{l_1}^{[1]} \otimes X_{l_2}^{[2]} \otimes \cdots \ ,\end{aligned}$$

where we put the sum inside the trace. We can now rename the occurring matrices as

$$C^l = \sum_{i_k j_{k-1}} \mu_{i_k j_{k-1} l} D_{j_{k-1}} B_{i_k}^T \ . \tag{4.40}$$

Note that the matrices C^l are indeed site-independent, according to our assumptions. Therefore we get

$$U_1\left(\frac{t}{n}\right) = \sum_{l_1, l_2, \ldots, l_N} Tr\left(C^{l_1} C^{l_2} \cdots C^{l_N}\right) X_{l_1}^{[1]} \otimes X_{l_2}^{[2]} \otimes \cdots \otimes X_{l_N}^{[N]} \ . \tag{4.41}$$

This is a MPO representation of $U_1\left(\frac{t}{n}\right)$ for N sites and PBC. As the representation is TI, we can take the limit $N \to \infty$ and obtain an iMPO representation for the time-evolution operator.

The advantage of this representation compared to the even-odd-Ansatz described before, is that it is translationally invariant. It therefore preserves the translational invariance of the iMPS we have introduced earlier.

As an example of this decomposition, take the Hamiltonian of the 1-dimensional Ising model in a transverse field

$$H = \sum_{k \in \mathbb{Z}} -J\left(\sigma_z^{[k]} \sigma_z^{[k+1]} + g\sigma_x^{[k]}\right) \ , \tag{4.42}$$

with real parameters J and g. In this case we have

$$h_2^{[k,k+1]} = -J\sigma_z^{[k]} \sigma_z^{[k+1]} \tag{4.43}$$

$$h_1^{[k]} = -Jg\sigma_x^{[k]} \tag{4.44}$$

We will consider this model later, when speaking about applications. By using the Trotter

decomposition, as showed above, we obtain

$$U(t) = \exp\left(-\frac{it}{\hbar}\sum_{k\in\mathbb{Z}}\left(h_2^{[k,k+1]} + h_1^{[k]}\right)\right) \tag{4.45}$$

$$= \left(U_2\left(\frac{t}{n}\right)U_1\left(\frac{t}{n}\right)\right)^n + \mathcal{O}\left(\frac{1}{n}\right) \ . \tag{4.46}$$

As $h_1^{[k]}$ is a single-body operator, U_1 is a product of local terms. Each of the terms corresponds to $\exp(\epsilon - Jg\sigma_x)$ on the local Hilbert spaces. Therefore we obtain a MPO with bond dimension $D = 1$ and site-independent matrices

$$M^0 = \cosh(-Jg\epsilon) \tag{4.47}$$

$$M^1 = \sinh(-Jg\epsilon) \ , \tag{4.48}$$

in the basis $X_0^{[k]} = \mathbb{1}_k$, $X_1^{[k]} = \sigma_x^{[k]}$.

To obtain the MPO representation of U_2, we start with

$$\exp\left(-J\epsilon\sigma_z^{[k]}\sigma_z^{[k+1]}\right) = \cosh(-J\epsilon)\,\mathbb{1}^{[k]}\otimes\mathbb{1}^{[k+1]} + \sinh(-J\epsilon)\,\sigma_z^{[k]}\otimes\sigma_z^{[k+1]} \tag{4.49}$$

$$= \sum_{i,j=0}^{1}\left(B_i^T B_j\right)X_i^{[k]}\otimes X_j^{[k+1]} \ , \tag{4.50}$$

where $X_0^{[k]} = \mathbb{1}_k$, $X_1^{[k]} = \sigma_z^{[k]}$ and

$$B_0 = \begin{pmatrix}\sqrt{\cosh(-J\epsilon)} & 0\end{pmatrix} \tag{4.51}$$

$$B_1 = \begin{pmatrix}0 & \sqrt{\sinh(-J\epsilon)}\end{pmatrix} \ . \tag{4.52}$$

By doing the calculation explained above, we finally obtain the site-independent matrices

$$C^0 = \begin{pmatrix}\cosh(-J\epsilon) & 0\\ 0 & \sinh(-J\epsilon)\end{pmatrix} \tag{4.53}$$

$$C^1 = \begin{pmatrix}0 & \sqrt{\sinh(-J\epsilon)\cosh(-J\epsilon)}\\ \sqrt{\sinh(-J\epsilon)\cosh(-J\epsilon)} & 0\end{pmatrix} \ . \tag{4.54}$$

We obtained an iMPO representation of the time-evolution operator for the 1-dimensional Ising model. Note also that the matrices in the representation are symmetric. This is a useful property, which we can use, to make our computations faster. In Chapter 5 we will discuss the application of the iTEBD algorithm to the Ising model, and we will use the above iMPO to simulate this model.

4.2.3. Truncation

In the last two subsections we have seen two possibilities to decompose the time-evolution operator, in order to apply it to the iMPS efficiently. Recall the formula for applying a MPO to a MPS, which is

$$\tilde{A}^i = \sum_{j,k} C^j \otimes A^k \langle i| X_n^j |k\rangle \ . \tag{4.55}$$

Assume that the bond dimension of the initial MPS is $D_0 \geq 1$ and the bond dimension of the MPO is $k \geq 1$. Then we obtain the final bond dimension $kD_0 \geq D_0$. Given that computational resources are limited, we have to keep the bond dimension of the represented states controlled. After a number of MPO applications, the bond dimension will exceed some maximal bond dimension D, which we can choose within the algorithm. We will therefore approximate the MPS after the application of a MPO, by a MPS with that maximal bond dimension D. This MPS should be as close as possible to the original one, w.r.t the norm of the Hilbert space. The approximated state will thus have similar expectation values of local observables, as the initial MPS. We will call this approximation the truncation step.

The idea of the truncation methods, we are going to describe, is to impose the canonical form (3.24) in the iMPS. This will lead to a stable algorithm. Remember that a gauge transformation is a transformation of the form

$$\tilde{A}^i = M A^i M^{-1} \tag{4.56}$$

for an invertible matrix M. We can then change this transformation in a certain way to obtain an iMPS with matrices $\tilde{A}^i$ of smaller bond dimension as approximation of the previous state.

Suppose after applying one of the MPOs for a step in the time-evolution, we are left with an iMPS given by tensors A^i of bond dimension Dk, where D was the bond dimension of the previous iMPS and k the bond dimension of the applied MPO.

We will introduce two methods for the truncation, which work well in different settings. The first method can be used for iMPS with real symmetric matrices and was introduced by Murg et al. [MCPV08]. It is faster than the second method, but restricted to the real symmetric iMPS. The second method works for general iMPS and was introduced by Orus et al. [OV08].

4.2.4. Method for Symmetric Matrix Product States

We suppose $\tilde{A}^i$ to be symmetric for all i.

For this truncation method we will use the following gauge condition,

$$\sum_{i=1}^{d} A^i \Lambda A^{i\dagger} = \Lambda \ , \tag{4.57}$$

with a diagonal matrix Λ. This can be easily transformed into the canonical form for iMPS and leads to an efficient method.

To do the truncation take the transfer-matrix of the iMPS

$$E = \sum_i A^i \otimes \overline{A}^i \ , \tag{4.58}$$

where we have used Theorem 3. We know that $E \in \mathbb{R}^{(Dk)^2 \times (Dk)^2}$ is also real symmetric. Let $|x\rangle \in \mathbb{R}^{(Dk)^2}$ be the leading right eigenvector with eigenvalue 1, ensured by normalization. To compute $|x\rangle$ one should use an iterative method for eigenvector calculation, which uses the symmetry and the sparseness of E, for example the Lanczos method. Because of the symmetry, $\langle x|$ is the leading left eigenvector. This is important and allows us, to make this truncation method faster than the general one, described in the next section.

As we showed earlier, the reshaped leading eigenvector $X \in \mathbb{R}^{(Dk)\times(Dk)}$ of the transfer operator is positive semidefinite and thus is diagonalizable with orthonormal eigenvectors. By diagonalization we obtain

$$X = USU^\dagger \ , \tag{4.59}$$

with unitary matrix $U \in \mathbb{C}^{Dk\times Dk}$ and diagonal matrix $S \in \mathbb{C}^{Dk\times Dk}$ with the singular values in the diagonal sorted by decreasing magnitude. If we would now construct new matrices A^i by

$$A^i_{new} = U^\dagger A^i U \ , \tag{4.60}$$

we would get

$$\begin{aligned}
\mathcal{E}_{new}(S) &= \sum_i A^i_{new} S A^{i\dagger}_{new} && (4.61)\\
&= \sum_i U^\dagger A^i U S U^\dagger A^{i\dagger} U && (4.62)\\
&= U^\dagger \sum_i \left[A^i X A^{i\dagger}\right] U && (4.63)\\
&= U^\dagger X U && (4.64)\\
&= S \ . && (4.65)
\end{aligned}$$

In an analogous calculation by using the symmetry, one shows, that $S^\dagger$ is a fixed point of the map corresponding to the left site. This transformation achieves the canonical form, but does not reduce the bond dimension, as the matrices A^i_{new} have the same dimension as the original ones.

Note that A_{new} was obtained by a gauge transformation of the initial state, because U is unitary, and therefore the represented state has not changed. We have constructed the new iMPS in a way that the diagonal matrices S and $S^\dagger$ are fixed points of the corresponding CP-Maps. This means, as showed above, that S is an eigenvector of the transfer-matrix, if we reshape it as a vector. As stated above this is the condition on the matrices stated in the canonical form.

Now it is not difficult to compute the truncated iMPS of given dimension D. The idea is, that the matrix X can be approximated by $\tilde{X}$, with

$$\tilde{X} = \tilde{U}\tilde{S}\tilde{U}^\dagger \ , \tag{4.66}$$

with $\tilde{U}$ consisting of the first D rows of U and $\tilde{S}$ is the diagonal matrix with the D biggest singular values in the diagonal sorted by magnitude as before. It can be shown that $\tilde{X}$ is the best approximating matrix of rank D in the frobenius norm with respect to the matrix X [Ste93]. We can now define the truncated matrices through

$$A^i_{trunc} = \tilde{U}^\dagger A^i \tilde{U} \ . \tag{4.67}$$

This iMPS approximates the initial iMPS in a satisfactory way, as we will see in the applications.

The full algorithm would have the following form:

1. *Start with a iMPS given by symmetric $Dk \times Dk$ matrices A^i.*
2. *Calculate the transfer-matrix $E = \sum_i A^i \otimes \overline{A}^i$ corresponding to the iMPS.*
3. *Compute the leading eigenvector $|x\rangle$ of E.*
4. *Reshape $|x\rangle$ to a positive semidefinite matrix X and diagonalize it: $X = USU^\dagger$.*
5. *Define $\tilde{U}$ by the first D rows of U.*
6. *Compute truncated iMPS: $A^i_{trunc} = \tilde{U}^\dagger A^i \tilde{U}$.*

We observe that this algorithm can be implemented with computational costs $\mathcal{O}\left((Dk)^6\right)$, as we have to compute the singular value decomposition and the eigenvector of a $(Dk)^2 \times (Dk)^2$ matrix. Note that the eigenvector calculation is the most expensive step here. We can use, that E often is sparse, to accelerate the computation of the eigenvector $|x\rangle$. Later we will show some techniques to reduce the computational costs of the truncation.

4.2.5. Method for the General Case

In general the matrices A^i are not symmetric and thus we have to think about a different way to truncate them, because we cannot use that the leading eigenvectors on both sites are the same. We have to use the different leading left and right eigenvector in this method. This causes higher computation costs, because we have to compute two eigenvectors instead of one, of the big matrix E. This will later turn out to be the bottleneck of our algorithm with respect to computation time.

It is in general not possible to impose that $\mathbb{1}$ is a fixed point for both CP-Maps corresponding to the left and the right site. We will therefore use the special canonical form with $\mathbb{1}$ as fixed point only on the right site and the more general one on the left site. It will turn out, that this leads to a useful method.

First we take the transfer matrix $E \in \mathbb{C}^{(Dk)^2 \times (Dk)^2}$ given by

$$E = \sum_i A^i \otimes \overline{A}^i \ . \tag{4.68}$$

Because E is not symmetric, we have to compute the leading left and right eigenvectors $|x\rangle \in \mathbb{C}^{(Dk)^2}$ and $|y\rangle \in \mathbb{C}^{(Dk)^2}$ to the leading eigenvalue 1. We know that we obtain fixed points of the corresponding CP-Maps, if we reshape the vectors $|x\rangle$ and $|y\rangle$ to matrices $X \in \mathbb{C}^{(Dk)\times(Dk)}$ and $Y \in \mathbb{C}^{(Dk)\times(Dk)}$. As fixed points of CP-Maps X and Y are positive semidefinite. We can therefore compute the square roots of these matrices, which are unique positive semidefinite matrices $\sqrt{X} \in \mathbb{C}^{(Dk)\times(Dk)}$ and $\sqrt{Y} \in \mathbb{C}^{(Dk)\times(Dk)}$ fulfilling

$$X = \sqrt{X}^\dagger \sqrt{X} \tag{4.69}$$

$$Y = \sqrt{Y}^\dagger \sqrt{Y} \ . \tag{4.70}$$

Now we can compute the following singular value decomposition

$$\sqrt{Y}^T \sqrt{X} = USV^\dagger \ , \tag{4.71}$$

with unitary matrices $U, V \in \mathbb{C}^{(Dk)\times(Dk)}$ and a diagonal matrix $S \in \mathbb{C}^{(Dk)\times(Dk)}$ containing the singular values of $\sqrt{Y}^T \sqrt{X}$ in decreasing order.

As in the truncation method for the symmetric case, we can use the above singular value decomposition, to define truncated matrices $\tilde{U}, \tilde{V} \in \mathbb{C}^{(Dk)\times D}$ and $\tilde{S} \in \mathbb{C}^{D\times D}$. $\tilde{U}$ and $\tilde{V}$ consist of the first D rows of U and V. $\tilde{S}$ is the diagonal matrix with the biggest D singular values of $\sqrt{Y}^T \sqrt{X}$ in the diagonal.

We can now define the truncated iMPS by the matrices

$$A^i_{trunc} = \tilde{S}^{-1} \tilde{U}^\dagger \sqrt{Y}^T A^i \sqrt{X} \tilde{V} \ . \tag{4.72}$$

Note first, that

$$\sqrt{X} V S^{-1} U^\dagger \sqrt{Y}^T = \sqrt{X} \left(\sqrt{Y}^T \sqrt{X} \right)^{-1} \sqrt{Y}^T \tag{4.73}$$

$$= \mathbb{1} \ . \tag{4.74}$$

Note that we have to compute the inverse of the diagonal matrix $\tilde{S}$. It can happen that, the diagonal elements are too small and therefore lead to numerical problems. In this case, one can set diagonal entries of $\tilde{S}^{-1}$ to zero, if the corresponding elements in $\tilde{S}$ smaller than a certain threshold. This corresponds to the calculation of the Moore Pseudo-Inverse of the matrix $\sqrt{Y}^T \sqrt{X}$ and works well in praxis. To make the method more efficient it is also possible to take instead of the fixed final bond dimension D the number of singular values bigger than the chosen threshold for the Pseudo-Inverse. In this case we keep exactly the part of the matrix

S, which is big enough for inverting. Also this leads to an adaptive method as the it keeps the bond dimension as small as needed to describe the state faithfully.

We have therefore used a gauge transformation and obtained an approximation of the initial state. The proof that the transformed iMPS fulfills the canonical form can be done using a tensor diagram (See Fig.4.2). Note that a rotation of a matrix in the diagram corresponds to a commutation of the indices, i.e. a transposition of the matrix. The left site works in the same way and one obtains, that $S^\dagger S$ is the fixed point of the corresponding CP-Map.

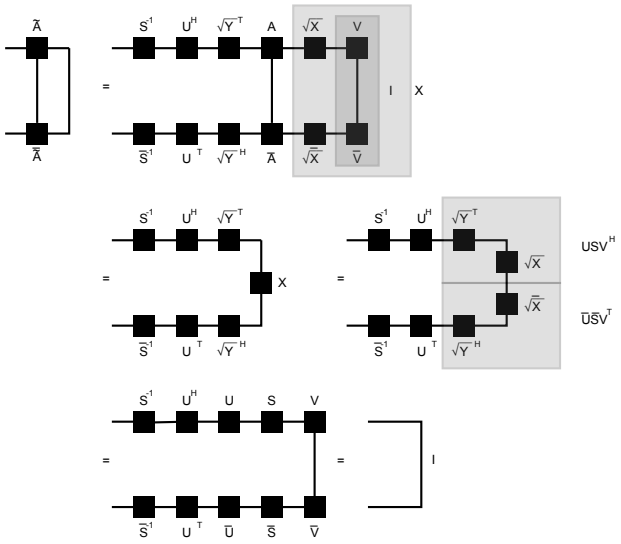

Figure 4.2.: Proof of the right hand side canonical form.

The full truncation algorithm would have the following form:

1. *Start with a general iMPS given by $Dk \times Dk$ matrices A^i.*
2. *Calculate the transfer-matrix $E = \sum_i A^i \otimes \overline{A}^i$ corresponding to the iMPS.*
3. *Compute the leading left and right eigenvectors $|x\rangle$ and $|y\rangle$ of E.*
4. *Reshape $|x\rangle$ and $|y\rangle$ to positive semidefinite matrices X and Y and calculate their square roots: $\sqrt{X}$ and $\sqrt{Y}$*
5. *Compute the following ordered singular value decomposition: $\sqrt{Y}^T\sqrt{X} = USV^\dagger$*
6. *Define $\tilde{U}$ and $\tilde{V}$ by the first D rows of U and V. Also define $\tilde{S}$ as the diagonal matrix with the biggest D singular values of $\sqrt{Y}^T\sqrt{X}$ in the diagonal*
7. *Compute truncated iMPS: $A^i_{trunc} = \tilde{S}^{-1}\tilde{U}^\dagger\sqrt{Y}^T A^i \sqrt{X}\tilde{V}$.*

We notice that the above algorithm can be implemented with computational costs $\mathcal{O}\left((Dk)^6\right)$, but we have to do more operations than in the symmetric case. For example we have to do the

very expensive eigenvector calculation twice. The method can be implemented less costly by exploiting the MPS structure, as we will see later.

4.3. Ground State Calculation

In the following we will focus on one application of the algorithm introduced in the last sections, namely the calculation of ground states of infinite 1-dimensional quantum many-body systems. The idea is very simple and is called **imaginary time-evolution**.

Note that we calculate the time-evolution of an initial state by applying the time-evolution operator

$$U(t) = \exp\left(-\frac{i}{\hbar}Ht\right) \ , \tag{4.75}$$

to the state, where H is the Hamiltonian of the physical system, and $t \in \mathbb{R}$ the time. The idea is now, to make the transformation to imaginary time $t \to -i\tau$ with $\tau \in \mathbb{R}$. This transformation gives the operator

$$U_i(\tau) = \exp\left(-\frac{H\tau}{\hbar}\right) \ . \tag{4.76}$$

This operator is not unitary anymore, and therefore we have to care about normalization. If we apply this operator to a state $\psi \in \mathcal{H}$, we obtain

$$U_i(\tau)|\psi\rangle = \sum_i \exp\left(-\frac{E_i\tau}{\hbar}\right)|i\rangle \ , \tag{4.77}$$

where we have used, that the eigenvectors $\{|i\rangle\}$ to the energy eigenvalues $\{E_i\}$ of the Hamiltonian H form an orthonormal basis of $\mathcal{H}$ as H is hermitian.

Let now $E_0 < E_i$ for all $i \neq 0$ be the energy of the ground state. Then we obtain

$$\exp\left(\frac{E_0\tau}{\hbar}\right)U_i(\tau)|\psi\rangle = |0\rangle + \sum_{i\neq 0}\exp\left(-\frac{(E_i - E_0)\tau}{\hbar}\right)|i\rangle \tag{4.78}$$

$$\longrightarrow |0\rangle \ , \tag{4.79}$$

for $\tau \to \infty$. Therefore we get by normalizing

$$\frac{U_i(\tau)|\psi\rangle}{\langle\psi|U_i^\dagger(\tau)U_i(\tau)|\psi\rangle} \longrightarrow |0\rangle \ . \tag{4.80}$$

This means, that, provided the initial state had non-zero overlap with the ground state, the state will evolve to the ground state, if we do imaginary time-evolution and normalize each step. The speed of convergence depends on the energy gap between the ground state and the first excited state of the Hamiltonian. If the time is longer than $\frac{\hbar}{E_1 - E_0}$ the convergence will be exponentially fast. This means that the computation is harder for smaller gaps. A system for which the gap goes to zero is called **critical**. In the last chapter we will apply the method

also at the critical point of the Ising model in transverse field and observe this problems in an example.

The only step in our algorithm were we can have a problem by simulating the imaginary time-evolution, is the Trotter decomposition. But if we look at the proof of theorem 9, we see, that it also works for $t \to -i\tau$, if the spectrum of A and B in the decomposition H = A + B, is bounded from below. In this cases the definition of the exponential function via the series is well defined, and the proof works in the same way, as mentioned.

To calculate the ground state using iTEBD we can start with a random initial iMPS, a fixed bond dimension and a fixed step size for the Trotter decomposition. We then apply the imaginary time-evolution operator using the method developed before. In each step of the method, we compute the energy.

As the imaginary time-evolution goes on, the energy decreases and converges after a certain time to some limit energy. The limit state will give us an upper bound to the ground state energy. The distance to the true energy depends on the Trotter error and the current bond dimension, which determines the set of states we can express. We can lower the Trotter error by decreasing the step size in the Trotter decomposition. The error in the truncation method is usually small enough to be neglected here. It can be also controlled over the bond dimension of the state, as the truncation error depends on the weights of the singular values which we neglected. This weight is determined by the number of singular values which can be kept, i.e. the bond dimension. This means that the two parameters, step size in the Trotter decomposition and bond dimension, control the final error.

If we detect the convergence in our method, we can lower the step size in order to reduce the trotter error. This will lead to a further decrease of energy, as the state can be described more accurately. Note that imaginary time-evolution is very robust in the sense, that errors made at the beginning do not matter. As long as the errors in the implementation get small, we achieve exponential convergence to the ground state.

We can also increase the bond dimension, which also leads to a decrease of the energy, as we can approximate the ground state in a better way. In practice we will use both possibilities. One will see, that the distance between two energy limits gets smaller as the bond dimension used for the two limits gets higher. This indicates, that the bond dimension approaches the value, we need, to describe the ground state as an iMPS. The Fig.4.3 shows the typical behaviour of energy in an imaginary time evolution. Convergence is apparently attained with given D and delta. Then, a further increase of D or reduction of delta produces a drop in energy, and several iterations are required to reach convergence again. We started with a step size of $\Delta\tau = 0.1$ and a bond dimension $D = 5$. After 60 steps we decreased the step size to $\Delta\tau = 0.05$. Then after 70 steps we increased the bond dimension to $D = 10$. We repeated this one more time up to $\Delta\tau = 0.025$ and $D = 20$.

This leads to the following method for ground state calculation:

1. *Define initial parameters: Bond dimension D, stepsize $\Delta\tau$ for Trotter decomposition*

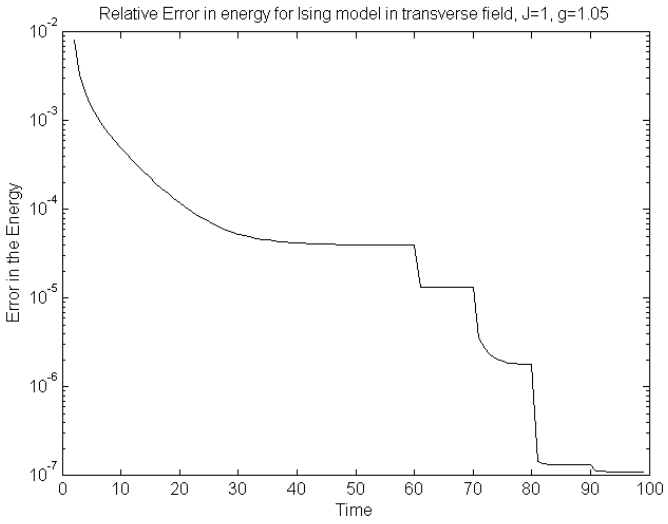

Figure 4.3.: Typical behaviour of the relative error in energy compared to the exact ground state energy during imaginary time-evolution.

2. *Start with a random iMPS with bond dimension D*

3. *Repeat until convergence of energy:*

   a) *Use iTEBD-algorithm, with parameters D and* $\Delta\tau$*, to apply the imaginary time-evolution operator* $U_i(\tau)$*for a chosen time-step* τ *to the iMPS*

   b) *Calculate the energy of the current state and the relative error compared to the last step.*

4. *If the energy converges: Increase the bond dimension or decrease the step-size in the Trotter decomposition and go to step 3.*

An important question is, how high the bond dimension has to get for a particular system, in order to approximate the ground state faithfully. There exists many results in this direction. It is known that for finite critical systems, and hence also for noncritical systems, that the bond dimension, needed for a MPS representation of the ground state, scales only polynomially with the system size [VC06] [Has07]. Therefore if the entanglement in the system does not grow too much and the gap between the ground state energy and the first excited state scales at most polynomially with the system size, we can find the ground state efficiently using our method(for rigorous conditions see [VC06]).

In the infinite case there exist results for special systems. For example it is known [VC06], that for the Hamiltonian

$$H = -\sum_{k\in\mathbb{Z}} \Delta\sigma_z^{[k]}\sigma_z^{[k+1]} + \sigma_x^{[k]}\sigma_x^{[k+1]} + \sigma_y^{[k]}\sigma_y^{[k+1]} \ , \tag{4.81}$$

for $\Delta < -1$ the ground state can be represented by an iMPS efficiently. More rigorous the error between the exact ground state and the iMPS approximant is bounded from above by an expression decaying faster than any inverse power of the bound dimension D. This Hamiltonian is a special point of the XXZ-model, which we are going to observe numerically in Chapter 5.

We have seen, that we can simulate the imaginary time-evolution and therefore calculate the ground state of a quantum many-body system using the iTEBD algorithm. In the next chapter we will do some concrete applications in this area.

5. Application to the 1-Dimensional Ising- and XXZ-Model

In this chapter we will use the iTEBD algorithm to calculate properties of the ground states in two situations. At first we will examine the 1-dimensional Ising model, which is exactly solvable. We can therefore compare the exact solution to the one we obtain with our method. After this we will look at a longer range interaction in the case of a XXZ spin chain.

5.1. The 1-Dimensional Ising Model in a Transverse Field

In this section we will illustrate the application of imaginary time-evolution in an iTEBD algorithm by calculating ground states of the infinite 1-dimensional Ising model in a transverse field.

Consider an infinite chain of particles with spin 1/2. Therefore the dimension of each local Hilbert space is $d = 2$, and the full Hilbert space has the form $\mathcal{H} = \bigotimes_{k\in\mathbb{Z}} \mathbb{C}^2$. The Hamiltonian of the system is

$$H = -J \sum_{k\in\mathbb{Z}} \left(\sigma_z^{[k]} \sigma_z^{[k+1]} + g\sigma_x^{[k]} \right) , \tag{5.1}$$

with real parameters J and g and Pauli matrices σ_z and σ_x. The goal of this section is to compute the ground state of such a system.

The problem of calculating ground states in this model has been solved exactly [Pfe70]. Therefore we can compare the results of our method with the exact results.

The MPO description of the time-evolution operator corresponding to the Ising model was already introduced in the last chapter. We saw that the matrices in this MPO were symmetric. Therefore we can do the simulation completely with symmetric iMPS and can also use the truncation method for symmetric iMPS within our method.

Using this method for the Ising model with J=1, we computed the ground states for various values of g, which corresponds to different external magnetic fields. According to [Pfe70] the exact energy E_0 of the ground state for the Ising-model in transverse field is given by

$$E_0 = -\frac{1}{4\pi} \int_{-\pi}^{\pi} 2J\sqrt{1 + g^2 - 2g\cos x}\; dx \tag{5.2}$$

We can compute this integral numerically and calculate the relative error between the energies we calculated and the exact energy of the ground state. Therefore we can compare the convergence of our method for the different values of g (See Fig.5.1). For this plot we started in the initial iMPS given by matrices $A^1 = 1$ and $A^2 = 0$ with bond dimension 1. We started also with a stepsize of $\Delta\tau = 0.1$ in the Trotter decomposition. When detecting convergence

the method divides the stepsize by 2, or doubles the bond dimension (See Appendix A). Each drop in the error corresponds to one of these events. One can see that the method converges very fast if g is not near 1. At $g = 1$ there is a critical point in the ground state energy, i.e. a point for which the energy gap tends to zero. Therefore the convergence is much slower than for the non-critical points. It is also a feature of critical points that the bond dimension has to be higher to describe the ground state well [VCM09].

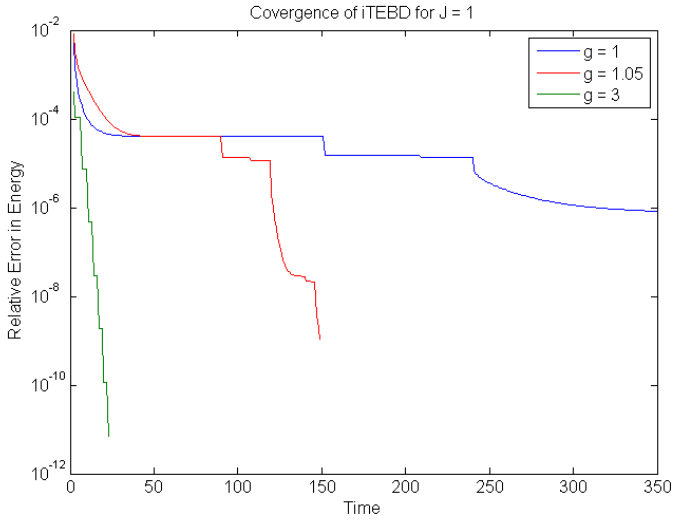

Figure 5.1.: Relative error in the energy for the Ising-model with parameters J=1, g=1,1.05,3.

We have seen, that we can approximate the ground state of the 1-dimensional Ising model in transverse field quite well using the iTEBD algorithm with imaginary time-evolution.

Now we can use the obtained states to calculate other observables in this system. This can be done efficiently, because we are given the states written as iMPS. We can for example calculate 2-point correlations in the ground states of the Ising model in x-direction. This important observable can be used to identify quantum phase transitions. In x-direction it is given by the expression

$$C_2(k) = \langle 0| \sigma_x^{[0]} \sigma_x^{[k]} |0\rangle - \left(\langle 0| \sigma_x^{[0]} |0\rangle \right)^2 , \tag{5.3}$$

and gives the correlation between two spins in x-direction, which are k sites apart in the spin chain. These functions have been calculated exactly [Pfe70] [BM71] for the Ising model in transverse field, and we can again compare the results of our method with the exact ones. As the 2-point correlator is built up out of expectation values of products of local operators, we can use Theorem 8 to calculate these functions. We have to evaluate combinations of transfer matrices of the form

$$\langle 0| \sigma_x^{[0]} \sigma_x^{[k]} |0\rangle = \lim_{l \to \infty} \frac{Tr\left(E^l E_{\sigma_x} E^k E_{\sigma_x} E^l\right)}{Tr\left(E^l E^{k+2} E^l\right)} , \tag{5.4}$$

where the fraction comes from normalizing. If we now introduce the spectral decomposition

$$E = \sum_i \lambda_i \left|R_i\right\rangle \left\langle L_i\right| \ , \tag{5.5}$$

with eigenvalues $\lambda_0 \geq \lambda_1 \geq \cdots$ and orthonormal left and right eigenvectors $|L_i\rangle$ and $|R_i\rangle$, we obtain

$$\left\langle 0\right| \sigma_x^{[0]} \sigma_x^{[k]} \left|0\right\rangle = \frac{\left\langle L_0\right| E_{\sigma_x} E^k E_{\sigma_x} \left|R_0\right\rangle}{\left\langle L_0\right| E^{k+2} \left|R_0\right\rangle} \ , \tag{5.6}$$

where the left and right eigenvectors $|L_0\rangle, |R_0\rangle$ corresponding to the eigenvalue of maximal modulus, as in the proof of Theorem 8. Using

$$E^k = \sum_i \lambda_i^k \left|R_i\right\rangle \left\langle L_i\right| \ , \tag{5.7}$$

we get

$$\left\langle 0\right| \sigma_x^{[0]} \sigma_x^{[k]} \left|0\right\rangle = \sum_i \left(\frac{\lambda_i}{\lambda_0}\right)^k \frac{\left\langle L_0\right| E_{\sigma_x} \left|R_i\right\rangle \left\langle L_i\right| E_{\sigma_x} \left|R_0\right\rangle}{\lambda_0^2 \left\langle L_0 | R_0\right\rangle} \ . \tag{5.8}$$

Therefore we get

$$C_2\left(k\right) = \left\langle 0\right| \sigma_x^{[0]} \sigma_x^{[k]} \left|0\right\rangle - \left(\left\langle 0\right| \sigma_x^{[0]} \left|0\right\rangle\right)^2 = \mathcal{O}\left(\left(\frac{\lambda_1}{\lambda_0}\right)^k\right) \ . \tag{5.9}$$

This means that $C_2(k)$ decays exponentially for $k \to \infty$, if $\lambda_0 \neq \lambda_1$. If $\lambda_0 = \lambda_1$ we have correlations of infinite range. As for iMPS always $\lambda_0 \neq \lambda_1$ is fulfilled, it is a general feature of iMPS, that the correlations of the above form, decay exponentially.

We can compute the correlations at the critical point of the Ising model in transverse field, i.e. for $J = 1$ and $g = 1$. At this point the correlation in x-direction is given by the expression

$$C_2\left(r\right) = \frac{4}{\pi^2\left(4r^2 - 1\right)} \tag{5.10}$$

Note that this correlator decays only polynomially and therefore the iMPS approximating the ground state cannot reproduce this quantity exactly. The polynomially decay of correlation functions is a feature of critical points, which makes them hard to approximate by iMPS. If we calculate the 2-point correlator $C_2(k)$ with our method, we see that the iMPS reproduces the correlation function well up to a certain point depending on the bond dimension. For bigger bond dimensions the results get more accurate. After a certain point the correlation function of the iMPS decays exponentially(See Fig. 5.2).

5.2. Long Range Interactions

The applicability of the techniques described before is not limited to nearest-neighbor interactions. As an illustration, we show here the case of long range XXZ-type interactions for a

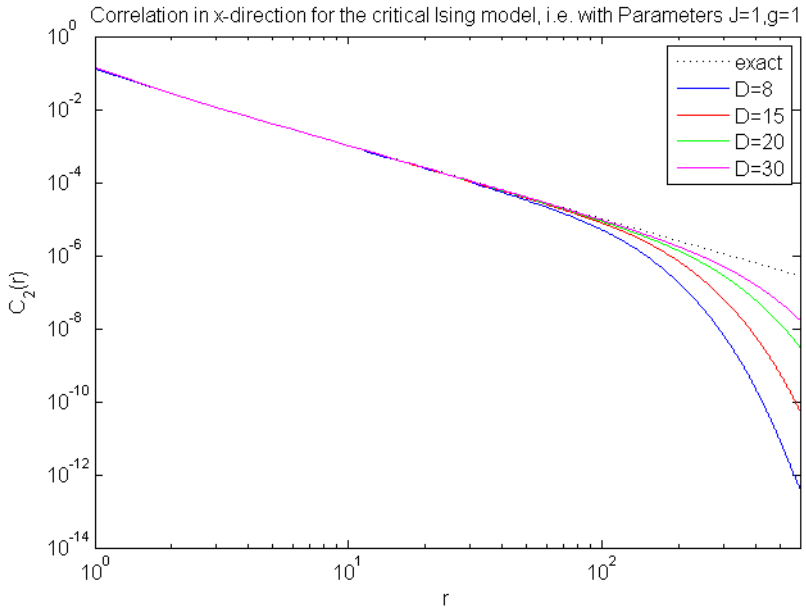

Figure 5.2.: Comparision between the 2-point correlators for different bond dimensions.

chain of spin 1/2. This kind of Hamiltonian can be relevant to describe the physics of ultracold dipolar gases trapped in an optical lattice.

We consider the Hamiltonian

$$H = \sum_{k\in\mathbb{Z}} \left(\sum_{r=1}^{l} \frac{1}{r^3} \left(\frac{\cos(\theta)}{4} \sigma_z^{[k]} \sigma_z^{[k+r]} + \frac{\sin(\theta)}{4} \left(\sigma_x^{[k]} \sigma_x^{[k+r]} + \sigma_y^{[k]} \sigma_y^{[k+r]} \right) \right) - \frac{\mu}{2} \sigma_z^{[k]} \right) , \quad (5.11)$$

with parameters $\theta \in [-\pi, \pi]$ and $g \in \mathbb{R}$ and Pauli matrices σ_i. $l \in \mathbb{N}$ determines the length of the interaction.

Notice that, when $l > 1$, the Hamiltonian contains long-range terms, which are not nearest-neighbor, whose strength decays as a power law. In this cases the decomposition of the time-evolution operator with the even-odd-ansatz explained above, is not possible. But we can use the decomposition into a translationally invariant MPO in this case. The matrices of the corresponding MPOs are not symmetrical anymore. Therefore we cannot use the fast truncation-method for symmetrical matrices, but have to use the general one.

As the interaction in the above Hamiltonian is not nearest-neighbor, for $l > 1$, it seems to be reasonable to break translational invariance and use iMPS with more than 1 tensor. If we simulate system with, for example, $l = 2$, we have for every $k \in \mathbb{Z}$ three particles which interact with each other. Therefore we can use iMPS, with the same symmetry, i.e. iMPS consisting of three tensors $A[j]$ for $j \in \{1, 2, 3\}$ repeated periodically over the infinite chain (See Fig.5.3).

To implement the imaginary time-evolution for the use of arbitrary periodicities we had to modify the methods to deal with more than one tensor. Let N denote the number of different

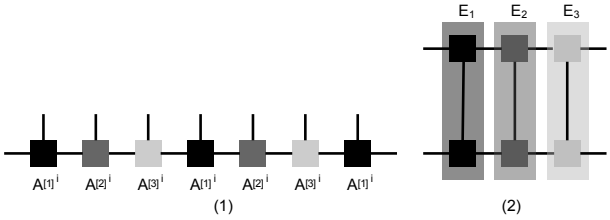

Figure 5.3.: (1)Periodic iMPS with N=3. (2)Transfer Matrix after putting N=3 sites together.

tensors used in the iMPS. If we want to calculate an expectation value of a k-body operator acting on the iMPS, it is not enough to calculate the expectation value at k neighboring sites if we use more than one tensor. Instead we compute the mean over all possibilities the operator can act on k out of the N tensors. For a local operator this would be the mean over all local expectation values corresponding to the N tensors.

To modify the truncation method for iMPS with N tensors $\{A[j]\}$, we first remember how to truncate in the translationally invariant case. Applying the truncation method gives us two matrices $\tilde{X}$ and $\widetilde{X^{-1}}$, which are approximations of the matrix X and its inverse X^{-1}, as explained in the Section 4.2.3. For translationally invariant iMPS we would have multiplied the matrix $\tilde{X}$ to the left and $\widetilde{X^{-1}}$ to the right side of the matrices A^i to obtain the truncated iMPS in canonical form.

In the case of iMPS with N tensors we can define the canonical form by collecting a block of N sites together and using the standard canonical form defined for translationally invariant states. We can therefore also use the truncation method for translationally invariant iMPS, to truncate iMPS with N tensors. Note that we have to take into account that there are different possibilities to collecting a block of N sites corresponding to cyclic permutations of the sites. Therefore we use the standard truncation to calculate two matrices $\tilde{X}[l]$ and $\widetilde{X^{-1}}[l]$ for each cyclic permutation numerated by $l \in \{1, \ldots, N\}$. Than we perform the transformations

$$A(k)^i \rightarrow \tilde{X}[k] A[k]^i \widetilde{X^{-1}}[k+1] \ , \tag{5.12}$$

for all i(See Fig.5.4). For the exact matrices $X[k]$ and $X^{-1}[k]$ this defines a gauge transformation of the iMPS and it also establishes the canonical form. This method leads to the desired truncation (See Appendix A).

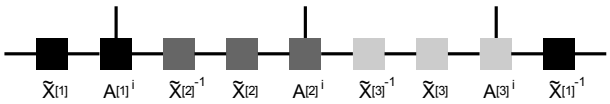

Figure 5.4.: Truncation of an iMPS with 3 tensors.

As application of our method we can start by computing the ground state energy of points in this model. We see that the energy converges against some limit, which varies for different

bond dimensions. Because we cannot solve this model analytically, we have to compare the limites obtained for different bond dimension to estimate how far we are from the exact ground state energy. For example, we can compute the energies for $\theta = \frac{\pi}{4}$, $\mu = 1$ and range $l = 2$ using periodic iMPS with 3 tensors and fixed steprange $\delta t = 0.025$. In this case we obtain convergence for different bond dimensions and observe, that the limit energies for different bond dimensions converge against a final energy, which is the energy of the ground state(See Fig.5.5).

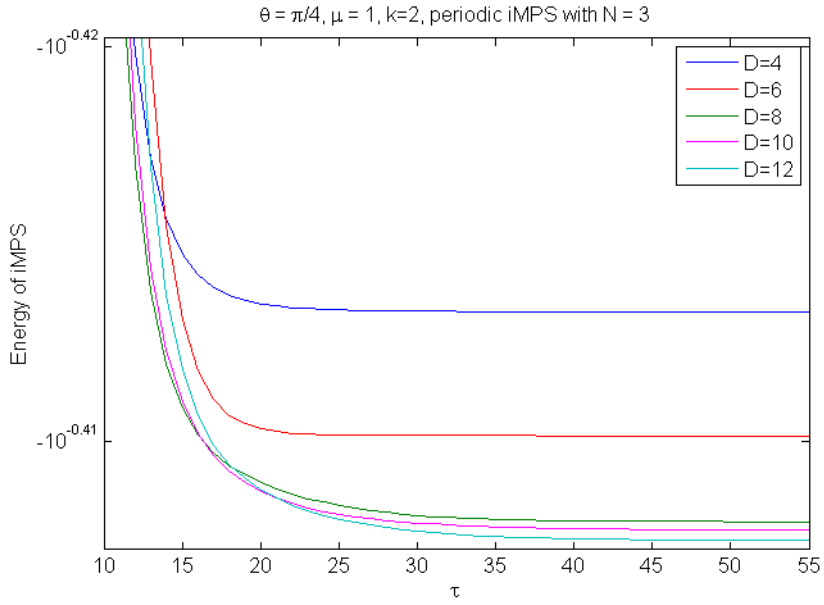

Figure 5.5.: Comparision between limit energies obtained for different bond dimensions.

As another application of the method, we can compute the mean magnetization in z-direction of the ground state in an efficient way using the techniques introduced before. For $\theta \in [-\pi, \pi]$ and $\mu \in [0, 1.5]$ we obtain the following plot(See Fig.5.6). We see that the magnetization shows interesting structures. The equipotential regions can be visualized in a contourplot(See Fig.5.7).

In this section we have seen, that we can use the iTEBD algorithm for the study of systems with longer range interaction. We obtained interesting structures for the values of local observables and showed how to use periodic iMPS of higher periodicity than 1 for the simulation of such systems.

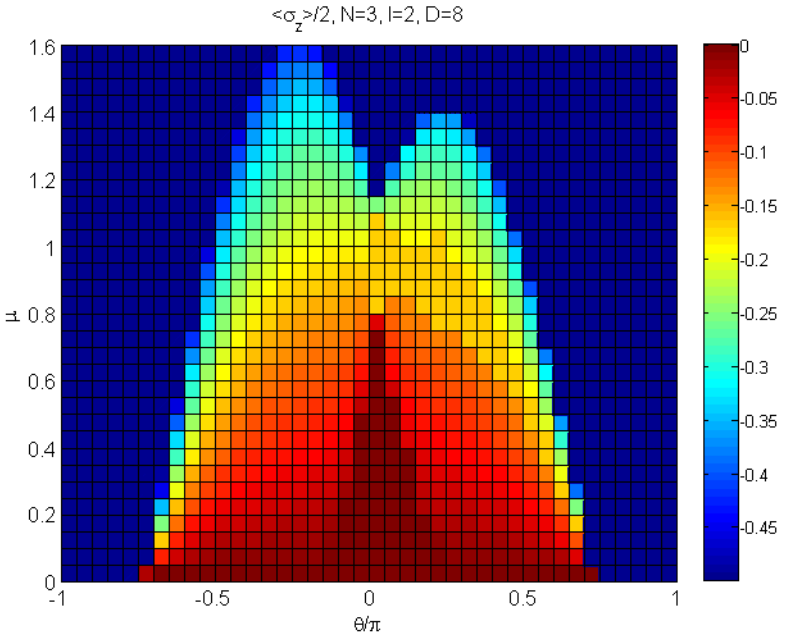

Figure 5.6.: Mean magnetization in z-direction of the ground state.

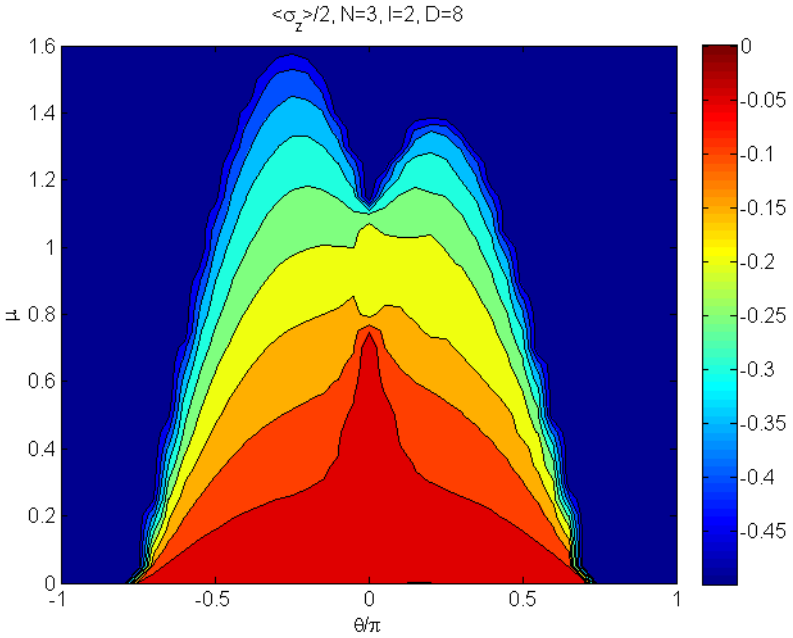

Figure 5.7.: Mean magnetization in z-direction of the ground state.

6. Implementation

In this chapter we will discuss some issues of the implementation of the iTEBD algorithm in MATLAB. We will focus on the contraction of tensor networks, which is the very basis of the algorithms. There are, broadly speaking, two kinds of contractions which we have to perform in the application of our methods. On one hand there are contractions of physical indices, which correspond to vertical contractions in a tensor diagram and on the other hand there are contractions of virtual indices, which correspond to horizontal contractions in a diagram(See Fig6.1). We will see that one should not be that restrictive in the classification and that it is necessary to choose the right order of contracting different indices in order to obtain an efficient method.

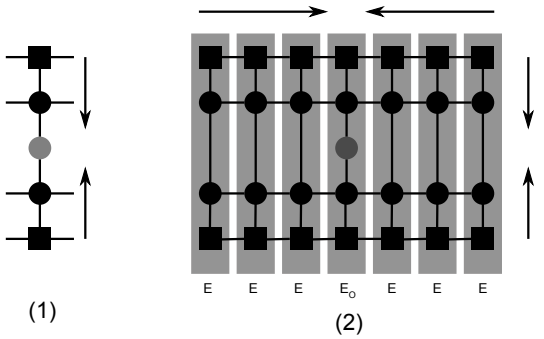

Figure 6.1.: (1) Vertical Contraction: Transfer Matrix. (2) Vertical and Horizontal Contractions: Expectation Value

6.1. Vertical Contractions

Within the iTEBD algorithm there are many cases where we have to contract the tensor network in the vertical direction. For example if we calculate transfer matrices or apply a MPO to a MPS. We will now give some general notes of how to implement these methods efficiently using MATLAB.

To determine the transfer matrix for an iMPS with matrices A^i, we have to calculate the expressions of the following kind

$$E_O = \sum_{i,j=1}^{d} \langle j| O |i\rangle \left(A^i \otimes \overline{A}^j \right) \ , \tag{6.1}$$

for a local operator O.

Let us now look at the special case $O = \mathbb{1}$. For general operators O we use a similar method, which is contained in the program (See Appendix A). In the special case $O = \mathbb{1}$ the above equation simplifies to

$$E = \sum_{i}^{d} A^i \otimes \overline{A}^i \ . \tag{6.2}$$

One possibility of calculating this, would be to use the MATLAB-function *kron* in the following way:

```
function E = transferOpWithKron(A)

    d = size(A,3);%physical dimension
    D = size(A,1); %bound dimension
    E = zeros(D^2,D^2);
    for k=1:d
        E = E + kron(A(:,:,k),conj(A(:,:,k)));
    end

end
```

But this implementation is not efficient. A more efficient way to compute the equation is the following, by using the MATLAB-function *reshape*

```
function E = transferOp(A)

    d = size(A,3);%physical dimension
    D = size(A,1); %bound dimension

    E = permute(A,[3 1 2]);
    E = reshape(E,d,D*D);
    E = E.'*conj(E);
    E = reshape(E,D,D,D,D);
    E = permute(E,[3 1 4 2]);
    E = reshape(E,D*D,D*D);

end
```

The error between the two methods is of the order of the machine precision. If we plot the computation time of the two methods for random complex iMPS of different bond dimensions, we see, that the method *transferOp* scales better for big bond dimensions (See Fig.6.2).

As the application of a MPO to a MPS is very similar from the point of view of the matrix operations involved to the computation of the transfer matrix, we just refer to Appendix A, where we stated the complete program. The function *mpoAppl* contains our implementation of this method.

6.2. Horizontal Contractions

There are also many parts in the algorithm, where we have to contract tensor networks horizontally. For example in order to calculate expectation values for iMPS with more than one tensor. We showed in chapter 2, that we can introduce iMPS, which are invariant under a translation of $N > 1$ sites, by putting N physical sites together and treating them as a single site in the algorithm. In Chapter 5 we used such states to compute ground state properties in the case of long range interactions in the Hamiltonian. If we want to truncate such iMPS or calculate expectation values, we have to construct the transfer matrix in the following way(See Fig.5.3). We

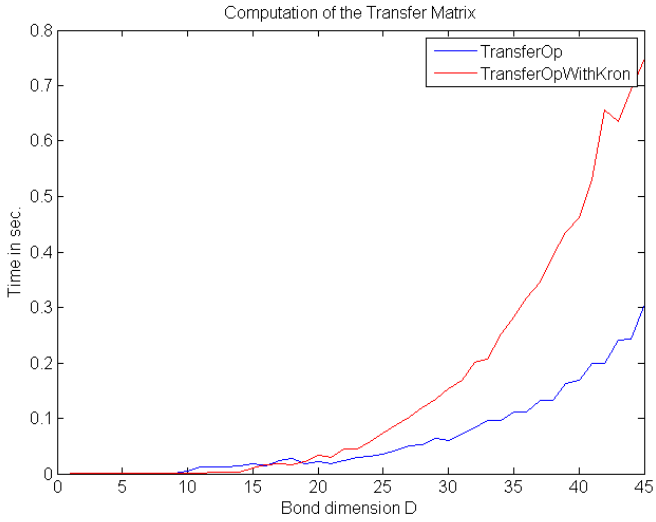

Figure 6.2.: Computation Time of the two methods for the calculation of E

have to do both vertical and horizontal contractions in this case and we can save computation time by choosing the proper order of this contractions.

There are many ways to contract the tensor network above. One can do it by constructing the transfer matrices corresponding to the matrices $A[1]^i, A[2]^i, \ldots, A[k]^i$ separately and than multiplying the big matrices $E_1, E_2, \ldots, E_k$, i.e. contracting first the physical indices and later the virtual ones. This would lead to a complexity of $\mathcal{O}\left(kD^6\right)$ and is thus not very efficient for big D. But note that it scales polynomially with the periodicity k. One can do a bit better by first computing the leftmost transfer matrix and then contracting the two indices going to the next tensors A separately. By repeating this over the chain, we calculate the complete truncation with $\mathcal{O}\left(d^2D^5\right)$ (See Fig.6.3). There is another way, which is a bit better in the cases, which we are going to observe. If we first contract the virtual indices and then the physical ones, we obtain a complexity of $\mathcal{O}\left(d^kD^4\right)$ (See Fig.6.4). This seems to be problematic, because of the exponential scaling in k, but as in our simulations k is fixed and $d << D$ it will be more efficient, if $d^k < d^2D$. As we are simulating spin-1/2 chains, we have d=2. For the values of k, we are going to take, the latter way will be therefore better.

We used the last technique in the program for the Ising case to calculate 2-point correlators (See next chapter) and in the program for the XXZ-model to calculate transfer matrices for periodic iMPS.

In the infinite case we have also the case, where we contract infinite networks in one direction. This is done by deriving the eigenvectors of the corresponding transfer matrices. As already mentioned in the section about the truncation method, one should use a vector iteration method here, which uses the properties of the iMPS, for example the Lanczos method for symmetric iMPS. We use the command *eigs* in order to compute the eigenvectors in this case, which

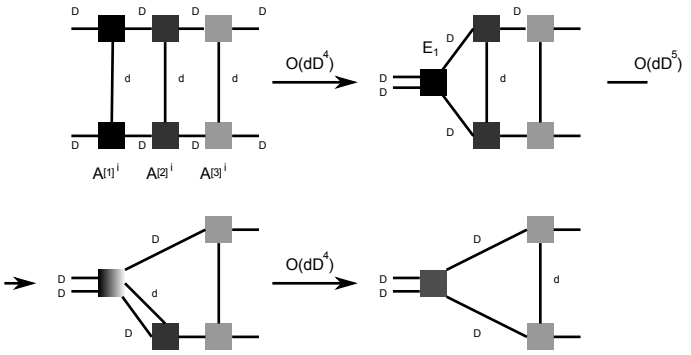

Figure 6.3.: Contraction in $\mathcal{O}\left(kD^5\right)$

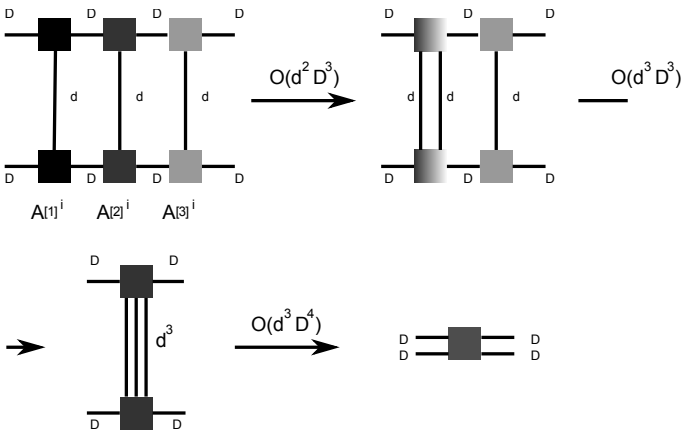

Figure 6.4.: Contraction in $\mathcal{O}\left(d^kD^4\right)$

chooses between different iterative methods, depending on the input.

There is also a way to accelerate the vector iteration used for eigenvector calculation. We have seen in chapter 1, that the application of the transfer matrix E to a vector can be written as the application of the completely positive map $\mathcal{E}$ given by

$$\mathcal{E}(X) = \sum_{i=1}^{d} A^i X A^{i\dagger} \tag{6.3}$$

to the vector reshaped as a matrix. Applying the map $\mathcal{E}$ can be done with costs $\mathcal{O}\left(dD^3\right)$, as it involves only multiplications of $D \times D$ matrices. Compared to that the multiplication of a vector with the $D^2 \times D^2$ transfer matrix needs $\mathcal{O}\left(D^4\right)$ operations. Therefore we can save time by doing the vector iteration with the map $\mathcal{E}$. This can be implemented using the *eigs* method and defining a different function for the iteration(See Appendix A: function *eigMPS*).

For the complete code see Appendix A.

7. Conclusion

We saw that the matrix product formalism represents states with bounded amount of entanglement efficiently. This picture is not limited to the 1-dimensional case and MPS can be generalized to higher dimensional systems. This leads in multi-dimensional systems to the so called projected entangled pair states(PEPS), which are a generalization of the MPS [VCM09]. By developing tensor-network methods for these more general states, efficient simulations become possible also in higher dimensional systems. As mentioned in the introduction, one can use MPS to understand the DMRG method, which is a very efficient algorithm for ground state calculation in finite 1-dimensional systems. It can be shown that the states constructed by DMRG are MPS and therefore the MPS also give DMRG its power. DMRG can also deal with infinite systems by increasing the system size until convergence. One problem of DMRG was, that it could not deal with real time-evolution in a satisfying way. The TEBD algorithm led to major improvements in this are. By combining TEBD and DMRG it is now possible to do real time-evolution in a satisfying way using DMRG.

The goal of this thesis was, to introduce algorithms for the simulation of infinite 1-dimensional quantum-many-body systems. Therefore we introduced the matrix product formalism first for finite and then for infinite systems. This formalism gives us a way to represent states of bounded amount of entanglement efficiently as MPS. It also can be applied to operators which leads to the MPO representation. Furthermore we showed how to calculate observables of such states in an efficient way. In section 3 we introduced the TEBD-algorithm, which uses the matrix product formalism to simulate 1-dimensional quantum-many-body systems efficiently. We also presented improvements to its original formulation. This algorithm can also be used to find ground states of 1-dimensional systems by imaginary time-evolution. We demonstrated this application for two physical systems. First we presented the calculation of ground state properties for the 1-dimensional Ising model in transverse field and second we showed that it can also be used in systems with longer range interaction. In the last case, we calculated the magnetization in the dipolar XXZ-model. We also showed, that we can impose certain structures to the iMPS to approximate the ground state in a more reliable way. Finally we presented the MATLAB-Code which we used for the simulations.

A. Matlab Code

A.1. Code for Ising model and symmetric iMPS

The following MATLAB-Code is the program for the imaginary time-evolution of the Ising model in transverse field.

```
function [erg,fin_err] = ITE_Ising()
        format long;


    %Ising-Hamiltonian: H = -J*sum(sig^z_i*sig^z_i+1 + g*sig^x_i)

    %Parameters:
        J = 1;
        g = 1.05;

    accuracy = 1e-10; %Accuracy for the detection of convergence and when to break up the
        optimization
    maxsteps = 100;   %Final imaginary time to break up.


    steprange = 0.025;%size of single time-step in the Trotter decomposition
    D = 5; %Bond dimension of MPS

        %Initial state in MPS-Form: |psi> = sum(tr((A_i1)(A_i2)...)|i1,i2,...>) = |00000...>
        %translational invariant: A_i1[1]=A_i2[2]=...

      A(:,:,1)=1; %Initial State
    A(:,:,2)=0;

    en = zeros(maxsteps,1);
    time = 2:1:maxsteps;

    err = zeros(maxsteps,1);

    exactEnerg = exactIsingEnergy(J,g);

    en(1)=exactEnerg+1;
    err(1)=1;

    minsteprange = 1e-5;
    maxdim = 20;

    range_count = 0;
    dim_count = 0;

    h=2;
    while (err(h-1)>=accuracy && h<maxsteps)
       A = evolve(A,D,1,steprange,J,g); %Calculates 1 time-step in imaginary time
       [A,v] = normal(A); %Normalizes the state
       energ = energyIs(A,v,J,g); %Calculates energy of the state
       en(h) = energ;
       err(h)= abs((energ-exactEnerg)/exactEnerg); %Relative error compared to exact energy
       abs((energ-exactEnerg)/exactEnerg)

       %Optimization of the Parameters:
       if abs(err(h)-err(h-1))<accuracy %Convergence Detection
            range_count = range_count + 1;
            if range_count==2 && steprange>minsteprange %Decrease stepsize of Trotter decomposition
               steprange = steprange/2
               range_count = 0;
               dim_count = dim_count+1;
            end
            dim_count = 0;
              if dim_count == 3 && D<maxdim %Increases bond dimension
                  D = 2*D
```

```
60                      dim_count = 0;
61                  end
62
63              end
64
65              h = h+1;
66
67          end
68
69          R = 10^3;
70          sigx = [0,1;1,0];
71          cor = 1:R;
72          t2 = 1:R;
73          for i=1:R
74              cor(i) = correlator2(A,i,sigx);
75          end
76
77          enf = en;
78          corrf = cor;
79
80          save(strcat(sprintf('en%d%d%d%d%d%d%d',round(100*J),round(100*g),D)),'enf');
81          save(strcat(sprintf('corr%d%d%d%d%d%d%d',round(100*J),round(100*g),D)),'corrf');
82
83          if h>=maxsteps
84              disp('Maxstep reached');
85          end
86
87          erg = en(h-1);
88          fin_err = abs(en(h-1)-exactEnerg)
89
90          semilogy(time,err(2:end));
91
92      end
93
94
95      function A = evolve(A_in,D,t_fin,steprange,J,g)
96
97          %Evolves the start iMPS given by A_in and base in imaginary time until t_fin
98          %in single time steps of length steprange for a one-dimensional Ising model in
99          %thermodynamic limit(translational invariant)with the Hamiltonian
100
101         % H = -J*sum(sig(i)_z*sig(i+1)_z + g*sig(i)_x)
102
103
104         N = t_fin/steprange; %Number of time steps
105
106         persistent T;
107         persistent stepr_temp;
108
109
110         %MPO's
111
112         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
113
114         %Time-evolution operator in MPO-Form
115         if isempty(T) || stepr_temp ~= steprange
116
117             cg = cosh(steprange*J*g/2);
118             sg = sinh(steprange*J*g/2);
119
120             c = cosh(steprange*J);
121             s = sinh(steprange*J);
122
123             T(:,:,1,1) = cg*cg*[c,sqrt(s*c);sqrt(s*c),s] + sg*sg*[c,-sqrt(s*c);-sqrt(s*c),s];
124             T(:,:,2,1) = 2*cg*sg*[c,0;0,s];
125             T(:,:,1,2) = 2*cg*sg*[c,0;0,s];
126             T(:,:,2,2) = cg*cg*[c,-sqrt(s*c);-sqrt(s*c),s] + sg*sg*[c,sqrt(s*c);sqrt(s*c),s];
127
128         end
129
130
131
132         stepr_temp = steprange;
133
134         %Evolution
135
136         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
137         A = mpoAppl(A_in,T);
138         A = truncation(A,D);
```

```
139     for k=2:N
140         A = mpoAppl(A,T); %Application of iMPO
141         A = truncation(A,D);%Truncation to dimension D
142     end
143
144 end
145
146
147 function erg = truncation(C,D)
148
149     %Takes an iMPS with symmetric C(:,:,i) as input and returns a truncated
150     %iMPS with final bound dimension D in canonical form.
151
152     opts.disp = 0;
153     opts.issym = 1;
154     opts.isherm = 1; %symmetric and real-valued
155
156     % C:Big MPS-tensor, D: Final bond dimension
157
158     dBig = size(C,1);
159     d = size(C,3);
160
161     if(dBig<=D)     %C already small enough
162         erg = C;
163     else
164         E = transferOp(C,eye(d));
165
166         [x,e] = eigs(E,1,'lm',opts);
167
168         xr = reshape(x,dBig,dBig);
169
170         [V,S] = svd(xr);
171
172         P = V(1:end,1:D);
173
174         C = permute(C,[1 3 2]);
175         C = reshape(C,dBig*d,dBig);
176         C=C*P;
177         C = reshape(C,dBig,d,D);
178         C = permute(C,[1 3 2]);
179         C = reshape(C,dBig,D*d);
180         C = P'*C;
181         erg = reshape(C,D,D,d);
182         erg = erg/sqrt(e);
183
184     end
185
186 end
187
188 function erg = energyIs(A,v,J,g)
189
190     %Calculates the energy of a iMPS for the one-dimensional Ising model in the
191     %thermodynamic limit (translational invariant).
192
193     % H = -J*sum(sig(i)_z*sig(i+1)_z + g*sig(i)_x)
194
195
196     opts.disp = 0;
197     opts.issym = 1;
198     opts.isherm = 1; %symmetric and real-valued
199
200
201     d = size(A,3); %dimension of one site
202
203     sigx = [0,1;1,0];
204     sigz = [1,0;0,-1];
205
206     E = transferOp(A,eye(d));
207     vr = v;                             %Use the given eigenvector here, to save time
208     [vl,el] = eigs(E',1,'lm',opts);
209
210     %Calculation of the x-part of the energy
211     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
212
213     Eox=transferOp(A,sigx);
214     energx = vl'*Eox*vr;
215     energx = energx/(vl'*E*vr);
216
217     %Calculation of the z-part of the energy
```

```
218     %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
219
220     Eoz = transferOp(A,sigz);
221     energz = vl'*Eoz*Eoz*vr;
222     energz = energz/(vl'*E*E*vr);
223
224
225     erg = -J*(energz+g*energx);
226
227 end
228
229
230 function erg = correlator2(A,r,C)
231
232     %Calculates the 2-point correlators of a state written as an iMPS with
233     %matrices A, of distance r, and with operator C
234
235
236     opts.disp = 0;
237     opts.issym = 1;
238     opts.isherm = 1; %symmetric and real-valued
239
240     d = size(A,3); %dimension of one site
241     D = size(A,1);
242
243     E = transferOp(A,eye(d));
244
245     %Eigenvectors for the boundaries:
246     [vr,er] = eigs(E,1,'lm',opts);
247     [vl,el] = eigs(E',1,'lm',opts);
248
249
250     Eoc = transferOp(A,C);
251
252     b = vl'*Eoc*vr/(vl'*E*vr);
253
254     E_fin = Eoc;
255
256     a = 1;
257     for j = 1:(r-1) %Finding a period which is efficient
258         if(d^j < D^2)
259             if(mod(r-1,j)==0)
260                 a = j;
261             end
262         else
263              break;
264         end
265     end
266     %Contracting the tensor network:
267     Atmp = reshape(permute(A,[2 1 3]),D,d*D).';
268     for p=2:a
269         Atmp = Atmp*reshape(A,D,d*D);
270         Atmp = reshape(permute(reshape(Atmp,(d^(p-1))*D,D,d),[2 1 3]),D,(d^p)*D).';
271     end
272
273     Atmp = permute(reshape(Atmp.',D,D,d^a),[2 1 3]);
274
275     Eper = transferOp(Atmp,eye(d^a));
276
277     k = 1;
278
279     while k<=((r-1)/a)
280       E_fin = E_fin*Eper;
281       k=k+1;
282     end
283
284     E_fin = E_fin*Eoc;
285
286     erg = vl'*E_fin*vr/(vl'*E*E*vr)-b*b;
287
288 end
289
290 function E = transferOp(A,O)
291
292     %Calculates the transfer-operator of the one-site operator O and the MPS A
293
294     d = size(A,3);%physical dimension
295     D = size(A,1); %bond dimension
296
```

```
297     O = O.';
298
299     K = reshape(A,D^2,d);
300     K = repmat(K,1,d);
301
302     elem = reshape(O,d^2,1);
303     elem = repmat(elem.',D^2,1);
304
305     K = K.*elem;
306
307     R = reshape(conj(A),D^2,d);
308     R = repmat(R,1,d);
309     R = reshape(R,D^2,d,d);
310     R = permute(R,[1 3 2]);
311     R = reshape(R,D^2,d^2);
312
313     E = R*K.';
314
315     E = reshape(E,D,D,D,D);
316     E = permute(E,[1 3 2 4]);
317     E = reshape(E,D^2,D^2);
318
319 end
320 function erg = mpoAppl(A,C)
321
322     %Applies the iMPO with matrices C to the iMPS with matrices A
323
324     d = size(A,3);%physical dimension
325     D = size(A,1); %bond dimension of iMPS
326     Dc = size(C,1);%bond dimension of iMPO
327
328     L = reshape(A,D*D,d);
329
330     T = reshape(C,Dc*Dc*d,d);
331
332     K = L*T.';
333
334     erg = reshape(K,D,D,Dc,Dc,d);
335
336     erg = permute(erg,[3 1 4 2 5]);
337     erg = reshape(erg,D*Dc,D*Dc,d);
338 end
339
340 function [erg,v] = normal(A)
341
342 %Normalization of the translational invariant iMPS given by A
343 %It also calculates the eigenvector to the biggest eigenvalue of the
344 %transfer-operator of A. This vector is used in the function energyIs to
345 %save time.
346
347   opts.disp = 0;
348   opts.issym = 1;
349   opts.isherm = 1; %symmetric and real-valued
350
351   d = size(A,3);
352
353   E = transferOp(A,eye(d));
354
355   [v,e] = eigs(E,1,'lm',opts);
356
357   erg = A/sqrt(e);
358 end
359
360 function erg = norm2(A)
361
362 %Calculates the norm^2 of the translational invariant iMPS given by A
363
364     opts.disp = 0;
365     opts.issym = 1;
366     opts.isherm = 1; %symmetric and real-valued
367
368     d = size(A,3);
369
370     E = transferOp(A,eye(d));
371
372     [vr,er] = eigs(E,1,'lm',opts);
373     [vl,el] = eigs(E',1,'lm',opts);
374
375     erg = vl'*E*vr;
```

```
376
377 end
378
379 function E = exactIsingEnergy(J,g)
380
381     %Calculates the exact energy of the ground state
382
383     E = -1/(4*pi)*quad(@F,-pi,pi,1E-10);
384
385     function y=F(x)
386         y=2*J*sqrt(1+g*g-2*g*cos(x));
387     end;
388
389 end
```

A.2. Code for general iMPS

For the XXZ-model most parts of the program are the same as for the Ising model with minor changes to use iMPS of higher periodicity. We therefore only present the methods for the calculation of the transfer operator and the application of the iMPO.

```
1  function A = mpoApTrunc(A,C,Dtr)
2      opts.isreal = false;
3      opts.disp = 0;
4      opts.issym = false;
5      opts.isherm = false;
6          % C:Big MPS-tensor, D: Final bond dimension
7
8      d = size(A,3);%physical dimension
9      D = size(A,1); %bound dimension
10     Dc = size(C,1);
11     N = size(A,4);
12
13         %Application of the iMPO:
14     A = reshape(A,D*D,d,N);
15     a = num2cell(A,[1 2]);
16     A = sparse(blkdiag(a{:}));
17     C = reshape(C,Dc*Dc*d,d);
18     C = repmat(C,1,N);
19     A = A*C.';
20     A = permute(reshape(A,D*D,N,Dc*Dc*d),[1 3 2]);
21     A = reshape(A,D,D,Dc,Dc,d,N);
22     A = permute(A,[3 1 4 2 5 6]);
23     A = reshape(A,Dc*D,Dc*D,d,N);
24
25
26     cutoff = 1e-8;%Zero for the Pseudo-Inverse
27     DBig = size(A,1);
28
29     xr = rand(DBig*DBig,N);
30     xl = rand(DBig*DBig,N);
31
32     if(isvector(A)==0 && DBig>Dtr)%Dimension small enough?
33
34         ordN = circshift((1:N).',(N+1)-N);
35
36
37         for q = 1:N %For all cyclic permutations
38             ord = circshift((1:N).',(N+1)-q);
39
40             %Contraction of N tensors:
41             Atmp = reshape(permute(A(:,:,:,ord(1)),[2 1 3]),DBig,d*DBig).';
42             for p=2:N
43                 Atmp = Atmp*reshape(A(:,:,:,ord(p)),DBig,d*DBig);
44                 Atmp = reshape(permute(reshape(Atmp,(d^(p-1))*DBig,DBig,d),[2 1 3]),DBig,(d^p)*DBig)
                       .';
45             end
46
47             Atmp = permute(reshape(Atmp.',DBig,DBig,d^N),[2 1 3]);
48
49             %Eigenvector calculation:
50             [xr(:,q),e] = eigMPS(Atmp);
```

```
51              [xl(:,q),e2] = eigMPS(permute(conj(Atmp),[2 1 3]));
52
53
54              X = reshape(xr(:,q),DBig,DBig);
55              Y = reshape(xl(:,q)',DBig,DBig);
56
57              Y = sqrtm(Y);
58              X = sqrtm(X);
59
60              [Utmp,Stmp,Vtmp] = svd(Y.'*X);
61
62                      %Calculation of the Pseudo-Inverse.
63                      %Determine the biggest bond dimension k<Dtr which is necessary
64              if(q == 1)
65                  for k=1:Dtr
66                      if(Stmp(k,k)<cutoff)
67                          break
68                      end
69                  end
70                  tr1 = sparse(zeros(2*N*k,2*DBig*N));
71                  tr2 = sparse(zeros(2*DBig*N,k));
72              end
73
74              S = Stmp(1:k,1:k);
75              U = Utmp(1:end,1:k);
76              V = Vtmp(1:end,1:k);
77
78              %Build the transformation matrices:
79              tr1(((2*(q-1))*k+1):((2*(q-1)+1)*k),((2*(q-1))*DBig+1):((2*(q-1)+1)*DBig)) = (1/sqrt(e))
                    *S\U'*Y.';
80              tr1(((2*(q-1)+1)*k+1):(2*q*k),((2*(q-1)+1)*DBig+1):(2*q*DBig)) = (1/sqrt(e))*S\U'*Y.';
81
82
83              tr2(((2*(ordN(q)-1))*DBig+1):((2*(ordN(q)-1)+1)*DBig),:) = X*V;
84              tr2(((2*(ordN(q)-1)+1)*DBig+1):(2*ordN(q)*DBig),:) = X*V;
85
86
87          end
88
89          %Transformation of A:
90          A = reshape(A,DBig,DBig,d*N);
91          a = num2cell(A,[1 2]);
92          A = sparse(blkdiag(a{:}));
93          A = tr1*A*tr2;
94          A = full(A);
95          A = reshape(A.',k,k,d,N);
96          A = permute(A,[2 1 3 4]);
97
98      end
99  end
100
101
102 function [erg,e] = eigMPS(A)
103
104         %Computes the eigenvector and eigenvalue of the
105         %transfer matrix of an iMPS by using the CP-map directly
106
107         d = size(A,3);%physical dimension
108     D = size(A,1); %bond dimension of iMPS
109
110     opts.disp = 0;
111     opts.isreal = false;
112
113     [erg,e] = eigs(@Av,D*D,1,'lr',opts);%Using eigs for calculating fixed point of the CP-Map
114
115         function vnew = Av(v) % Apply MPS to vector
116             v = reshape(v,D,D);
117             vnew = zeros(D,D);
118
119             %Application of CP-Map:
120             for k=1:d
121                 vnew = vnew + A(:,:,k)*v*A(:,:,k)';
122             end
123
124             vnew = reshape(vnew,D^2,1);
125         end
126
127 end
128
```

```
129
130 function E = transferOp(A,O)
131
132     %Calculates the transfer-operator of the one-site operator O and the
133     %iMPS A with N tensors
134
135     %O = 1, calculates transfer matrix for the identity
136
137     d = size(A,3);%physical dimension
138     D = size(A,1); %bound dimension
139     N = size(A,4); %Number of tensors
140
141     O = O.';
142
143     if N==1
144         if O==1
145            E = permute(A,[3 1 2]);
146            E = reshape(E,d,D*D);
147            E = E.'*conj(E);
148            E = reshape(E,D,D,D,D);
149            E = permute(E,[3 1 4 2]);
150            E = reshape(E,D*D,D*D);
151         else
152             E = reshape(A,D^2,d);
153             E = repmat(E,1,d);
154
155             elem = reshape(O,d^2,1);
156             elem = repmat(elem.',D^2,1);
157
158             E = E.*elem;
159
160             R = reshape(conj(A),D^2,d);
161             R = repmat(R,1,d);
162             R = reshape(R,D^2,d,d);
163             R = permute(R,[1 3 2]);
164             R = reshape(R,D^2,d^2);
165
166             E = R*E.';
167
168             E = reshape(E,D,D,D,D);
169             E = permute(E,[1 3 2 4]);
170             E = reshape(E,D^2,D^2);
171         end
172     else
173
174         if O==1
175             E = permute(A,[3 1 2 4]);
176             E = reshape(E,d,D*D,N);
177             a = num2cell(E,[1 2]);
178             E = sparse(blkdiag(a{:}));
179
180             E = E.'*conj(E);
181             E = full(E);
182
183             E = reshape(E,D*D*N,D*D,N);
184             E = squeeze(sum(E,3));
185             E = reshape(E.',D*D,D*D,N);
186             E = permute(E,[2 1 3]);
187
188             E = reshape(E,D,D,D,D,N);
189             E = permute(E,[3 1 4 2 5]);
190             E = reshape(E,D*D,D*D,N);
191         else
192
193             E = reshape(A,D*D,d,N);
194             a = num2cell(E,[1 2]);
195             E = sparse(blkdiag(a{:}));
196
197             C = repmat(O,1,N);
198             C = reshape(C,d,d,N);
199             c = num2cell(C,[1 2]);
200             C = sparse(blkdiag(c{:}));
201
202             C = E*C;
203
204             E = C*E';
205
206             E = full(E);
207             E = reshape(E,D*D*N,D*D,N);
```

```
208                 E = squeeze(sum(E,3));
209                 E = reshape(E.',D*D,D*D,N);
210                 E = permute(E,[2 1 3]);
211 
212                 E = reshape(E,D,D,D,D,N);
213                 E = permute(E,[3 1 4 2 5]);
214                 E = reshape(E,D*D,D*D,N);
215             end
216         end
217 end
218 
219 function A = normal(A)
220 
221         %Normalization of the periodical iMPS A of period N in canonical form
222 
223         opts.disp = 0;
224         opts.isreal = false;
225         N = size(A,4);
226         D = size(A,1);
227         d = size(A,3);
228 
229         for k = 1:N
230             n = 0;
231             for l = 1:d
232                 n = n + A(:,:,l,k)*A(:,:,l,k)';
233             end
234 
235             f = sum(diag(n))/D;
236             A(:,:,:,k) = A(:,:,:,k)/sqrt(f);
237         end
238 
239 end
```

Bibliography

[AKLT88] I. Affleck, T. Kennedy, E.H. Lieb, and H. Tasaki. Valence bond ground states in isotropic quantum antiferromagnets. *Commun. Math. Phys.*, 115:477, 1988.

[BM71] E. Barouch and B.M. McCoy. Statistical Mechanics of the XY Model. II. Spin-Correlation Functions. *Phys. Rev. A*, 3(2):786–804, 1971.

[Cho75] M.-D. Choi. Completely Positive Linear Maps on Complex Matrices. *Lin. Alg. and its Appl.*, 10(285), 1975.

[CV09] J.I. Cirac and F. Verstraete. Renormalization and tensor product states in spin chains and lattices. *arXiv:0910.1130v1*, 2009.

[EHK77] D.E. Evans and R. Hoegh-Krohn. Spectral Properties of Positive Maps on C*-Algebras. *J. Lond. Math. Soc.*, 2(17):345–355, 1977.

[FNW92] M. Fannes, B. Nachtergaele, and R.F. Werner. Finitely Correlated States on Quantum Spin Chains. *Lett.Math.Phys.*, 25:249, 1992.

[Has07] M.B. Hastings. An area law for one-dimensional quantum systems. *J. Stat. Mech.*, (P08024), 2007.

[HHHH07] R. Horodecki, P. Horodecki, M. Horodecki, and K. Horodecki. Quantum Entanglement. *arxiv:quant-ph/0702225*, 2007.

[MCPV08] V. Murg, J.I. Cirac, B. Pirvu, and F. Verstraete. Matrix product operator representations. *New J. Phys.*, 12(025012), 2008.

[NC07] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, 2007.

[OR95] S. Ostlund and S. Rommer. Thermodynamic Limit of Density Matrix Renormalization. *Phys.Rev.Lett*, 75:3537, 1995.

[OV08] R. Orus and G. Vidal. Infinite time-evolving block decimation algorithm beyond unitary evolution. *Phys.Rev.B*, 78:155117, 2008.

[Per05] M. Perotti. Matrix Product Formalism. Diplomarbeit, Technische Universität München, 2005.

[Pfe70] P. Pfeuty. The One-Dimensional Ising Model with a Transverse Field. *Ann. Phys. (N.Y.)*, 57:79–90, 1970.

[PGVWC07] D. Perez-Garcia, F. Verstraete, M.M. Wolf, and J.I. Cirac. Matrix Product State Representations. *Quantum Inf. Comput.*, 7(401), 2007.

[PWE10] P. Pippan, S.R. White, and H.G. Evertz. Efficient matrix-product state method for periodic boundary conditions. *Phys.Rev.B*, 81(081103), 2010.

[QSS07] A. Quarteroni, R. Sacco, and F. Saleri. *Numerical Mathematics*. Springer, 2007.

[RO97] S. Rommer and S. Ostlund. Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group. *Phys.Rev.B*, 55:2164, 1997.

[Ste93] G.W. Stewart. On the Early History of the Singular Value Decomposition. *SIAM Rev.*, 35(551), 1993.

[Suz84] M. Suzuki. Decomposition formulas of exponential operators and Lie exponentials with some applications to quantum mechanics and statistical physics. *J. Math. Phys.*, 26(4), 1984.

[Suz90] M. Suzuki. Fractal Decomposition of Exponential Operators with Applications in Many-Body Theories and Monte-Carlo Simulations. *Phys. Lett. A*, 146(6), 1990.

[Tro59] H.F. Trotter. On the product of semi-groups of operators. *Proc. Am. Math. Soc.*, 10(4):545, 1959.

[VC06] F. Verstraete and J.I. Cirac. Matrix product states represent ground states faithfully. *Phys. Rev. B*, 73(094423), 2006.

[VCM09] F. Verstraete, J.I. Cirac, and V. Murg. Matrix Product States, Projected Entangled Pair States, and variational renormalization group methods for quantum spin systems. *arXiv:0907.2796v1*, 2009.

[Vid03] G. Vidal. Efficient Classical Simulation of Slightly Entangled Quantum Computations. *Phys.Rev.Lett.*, 91:147902, 2003.

[Vid07] G. Vidal. Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension. *Phys.Rev.Lett.*, 98:070201, 2007.

[VPC04] F. Verstraete, D. Porras, and J.I. Cirac. Density Matrix Renormalization Group and Periodic Boundary Conditions: A Quantum Information Perspective. *Phys.Rev.Lett.*, 93:227205, 2004.

[Whi92a] S.R. White. Density-matrix algorithms for quantum renormalization groups. *Phys.Rev.B*, 48:10345, 1992.

[Whi92b] S.R. White. Density matrix formulation for quantum renormalization groups. *Phys.Rev.Lett*, 69:2863, 1992.