Florian Maislinger, BSc.

A Comparison of Recently Developed Time-Evolution Algorithms for One-Dimensional Systems with Long-Range Interactions

DIPLOMA THESIS

written to obtain the academic degree of

Diplom-Ingenieur

Master's programme; Technical Physics

submitted to

Graz University of Technology

Supervisor:
Ao.Univ.-Prof. Dipl.-Phys. Dr.rer.nat. Hans Gerd Evertz

Institute of Theoretical and Computational Physics

Graz, May 2016

EIDESSTATTLICHE ERKLÄRUNG

AFFIDAVIT

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Das in TUGRAZonline hochgeladene Textdokument ist mit der vorliegenden Masterarbeit identisch.

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master's thesis.

Datum/Date	Unterschrift/Signature

To Franz, Anita, Barbara, Stefanie, and Jean-Luc.

There are four lights!

Abstract

In recent decades the research of strongly correlated materials has been of great popularity within the physics research community. To properly describe these materials we cannot discard electron-electron interactions and thus the physical models of these materials are of great mathematical complexity. To study and numerically simulate strongly correlated electrons, very precise methods are available for one-dimensional systems, namely the Density-Matrix Renormalization Group (DMRG) for static quantities and the more recent methods of Time-Evolving Block Decimation (TEBD) and time-dependent DMRG (tDMRG) for time evolutions. Both algorithms work within the framework of Matrix Product States (MPS) that is best suited for one-dimensional models. For years, TEBD and the related tDMRG have been the first choice to compute time evolutions of a quantum system. Their main drawback is that they can only deal with nearest neighbor models (or finite short range), i.e. electrons only interact with each other if they are located on the same atom or located on neighboring atoms. This means that if long-ranged electron interactions are important to describe strongly correlated materials, we are unable to capture the physics of said materials with TEBD or tDMRG. To overcome this problem, two new classes of algorithms were developed in recent years. In 2011, Jutho Haegeman et. al. introduced the Time-Dependent Variational Principle for quantum lattices (TDVP). This algorithm is able to calculate the time evolution of quantum systems in the thermodynamic limit and was originally formulated for nearest neighbor interactions. In the course of this thesis this algorithm was extended to work with exponentially decaying long-ranged interactions. There is also a version of TDVP for finite lattices. This version can inherently work with long-ranged interactions. Furthermore, in 2014, Michael Zaletel et. al. introduced an algorithm to calculate time evolutions of long-ranged interactions that is based on Matrix Product Operators. Both algorithms have advantages and drawbacks. In the present thesis these two new algorithms are quantitatively compared for the one-dimensional transverse field Ising model (TFI) with exponentially decaying long-range interactions. The results show that if the interactions decay fast with respect to distance, the algorithm based on Matrix Product Operators is a viable alternative to TDVP. If the interactions decay slowly with respect to distance, TDVP should be used and is orders of magnitudes better than the other method.

Zusammenfassung

Seit Jahrzenten stößt die Erforschung von stark korrelierten Materialien auf großes Interesse in der physikalischen Forschungsgemeinde.
Die zugehörigen physikalischen Modelle sind in der Regel mathematisch sehr komplex, da die Elektron-Elektron-Wechselwirkung nicht vernachlässigt werden kann um diese Materialien zu beschreiben. Für eindimensionale Modelle sind zur Erforschung und Simulation von stark korrelierten Elektronen sehr genaue Verfahren verfügbar, nämlich die Density-Matrix Renormalization Group (DMRG) zur Errechnung von statischen Messgrößen und Time-Evolving Block Decimation (TEBD) und time-dependent DMRG (tDMRG) für Zeitentwicklungen. Beide Algorithmen arbeiten dabei mit Matrixproduktzuständen (MPS), die wiederum am besten geeignet zur Untersuchung von eindimensionalen Modellen sind. Jahre lang waren TEBD und das verwandte tDMRG die erste Wahl, wenn es darum ging Zeitentwicklung von Quantensystemen zu berechnen. Dabei haben diese Verfahren den Nachteil, dass es damit nur möglich ist eine Nächste-Nachbar-Wechselwirkung zu simulieren (oder endliche kurzreichweitige Wechselwirkungen). Das heißt, dass die Elektronen nur wechselwirken, wenn sie sich auf dem selben Atom oder benachbarten Atomen befinden. Damit sind wir nicht in der Lage diese Materialen mit TEBD oder tDMRG zu untersuchen, falls langreichweitige Elektronenwechselwirkungen wichtig sind um stark korrelierte Materialien zu beschreiben. Um dieses Problem zu lösen wurden in den letzten Jahren zwei neue Klassen von Algorithmen entworfen. Im Jahre 2011 haben Jutho Haegeman et. al. das sogenannte Time-Dependent Variational Principle (TDVP) vorgestellt. Mit diesem Algorithmus ist es möglich, Zeitentwicklungen von Quantensystem im thermodynamischen Limes zu berechnen und wurde ursprünglich ebenfalls nur für eine Nächste-Nachbar-Wechselwirkung entworfen. Im Zuge dieser Diplomarbeit wurde diese Methode erweitert, um auch mit exponentiell-abfallenden langreichweitigen Wechselwirkungen zu arbeiten. Außerdem gibt es eine Version von TDVP, mit der es möglich ist, Zeitentwicklungen auf endlichen Gittern zu berechnen. Diese Version kann in ihrer ursprünglichen Formulierung bereits mit langreichweitigen Wechselwirkungen rechnen. Weiters haben Michael Zaletel et. al. in Jahre 2014 einen Algorithmus zur Zeitentwicklung von langreichweitigen System vorgestellt, der auf dem Matrixprodukt-operatorformalismus basiert. Beide Methoden haben ihre Vor- und Nachteile. Im Zuge dieser Diplomarbeit wurden beide Algorithmen quantitativ verglichen mit einer abgeänderten Variante des transverse field Ising Modells (TFI) mit exponentiell-abfallenden langreichweitigen Wechselwirkungen. Die Ergebnisse zeigen, wenn diese Wechselwirkungen schnell abfallen bezüglich des Abstands, dann ist die operatorbasierende Methode eine brauchbare alternative zu TDVP. Fallen die Wechselwirkung jedoch langsam ab, so sollte TDVP benutzt werden, da es in diesem Fall um Größenordnungen besser als das andere Verfahren. german

Contents

1 Introduction and Physical Background **1**
1.1 Models . . . 2

2 Matrix Product States and Operators **7**
2.1 Basics . . . 7
2.2 Graphical representation . . . 12
2.3 Normalization Conditions . . . 13
2.4 Uniform Matrix Product States . . . 18
2.5 Tangent Space . . . 24

3 Density-Matrix Renormalization Group **28**
3.1 Finite DMRG . . . 28
3.2 Infinite DMRG . . . 31

4 Time-Evolving Block Decimation **33**

5 Time-Dependent Variational Principle **35**
5.1 Basic Idea . . . 35
5.2 Finite Lattice TDVP . . . 35
5.3 Infinite Lattice TDVP . . . 42

6 Time Evolution with Matrix Product Operators **58**
6.1 Basics . . . 58
6.2 W^I . . . 59
6.3 W^{II} . . . 62
6.4 Second Order . . . 66

7 Matrix Product Operators of Fermionic Systems **68**

8 Results: Ground State Search in the Thermodynamic Limit **71**

9 Results: Long Range Transverse Field Ising Model **74**

10 Conclusions **88**

A Appendix **89**
A.1 Calculation of the geometric series of the transfer matrix . . . 89
A.2 Linearly independent parameters of B(x) . . . 90
A.3 Calculation of K_{OS}, K_{NN}, K_l^n, K_r^n, and K_{LR} . . . 90
A.4 Computational Cost of Operator Application and TDVP . . . 91
A.5 Plots of the Fits . . . 93

1 Introduction and Physical Background

In trecent decades, strongly correlated materials have been a topic of great interest within the physics research community. To understand these materials, electron-electron interactions cannot be neglected. The problem here is that the study of strongly correlated electrons is of great analytic and numerical complexity, so up until this day many effects are not completely or not at all understood. With the discovery of yttrium barium copper oxide (YBCO) high temperature superconductors in 1986 [1] and the giant magnetoresistance in 1988 [2], [3] understanding these materials has also become of great technological and economical relevance.

The first approach to calculate the ground state or the time evolution of a system is exact diagonalization. Here we try to diagonalize the Hamiltonian of the system, i.e. find the full set of eigenvectors and eigenvalues. This method is severely limited by system size, since the dimensions of the Hamiltonian grows like d^N, where d is the local Hilbert space dimension and N is the number of lattice sites. Exact diagonalization by hand usually fails after a couple of lattice sites and numerical diagonalization still fails after ≈ 15 lattice sites. Hence, investigation of larger systems and systems in the thermodynamic limit is out of the question. In 1950, Cornelius Lanczos introduced an algorithm to find the k eigenvectors corresponding to the k lowest eigenvalues of Hermitian matrices [4]. The advantage of this method is that we do not need to know the Hamiltonian $\hat{H}$ itself, but only the action of the Hamiltonian on an arbitrary state $\hat{H}\,|\psi\rangle$. This allows the use of sparse matrices and similar tools. The Lanczos method is still limited by lattice size as we need to keep at least two vectors with d^N entries in our computer memory. Another class of algorithms consists of the so called quantum Monte Carlo algorithms [5], [6]. The idea here is to calculate expectation values via importance sampling. One of the major drawbacks of quantum Monte Carlo simulations is the so called sign problem [6]. As an interesting side note: In [7] and [6] it has been shown that many physical problems are in the complexity class NP-complete and a general solution to the sign problem would imply P = NP. This means, if it will ever be proven that P $\neq$ NP, there can not be a general solution to the fermionic sign problem!

Another idea is to represent the coeffecients of a quantum state as a product of matrices (the used methods in this thesis are based upon this idea). This resulting construct is called Matrix Product State, and with tools called DMRG and TEBD we can calculate very precise ground states and time evolutions of quantum systems (see below) [8]. It can also be shown that the Matrix Product State formalism has problems too, if P $\neq$ NP [9] (although we have in general no sign problem within the MPS and MPO formalism). In contrast to all the methods above the MPS formalism allows investigation of one dimensional system with a high number of lattice sites and even systems in the thermodynamic limit. Furthermore we can calculate time evolution of systems in non-equilibrium [10]. The standard methods to calculate ground states and time evolutions are DMRG and TEBD (see section 3 and section 4). The one limiting factor of TEBD is that we can simulate only models with nearest neighbor interactions and that may not be enough to capture the physics of the real world (see below).

In the first chapter of this thesis, we will give a brief introduction to the physical models used to test the algorithms described below. In section 2 we will go through the basics of the Matrix Product State formalism and the concept of tangent states that are important for TDVP. In section 4 and section 3 there is a very short introduction to the two standard methods for time evolutions (Time-Evolving Block Decimation; TEBD) and ground state searches (Density-Matrix Renormalization Group; DMRG). The basics of the finite and infinite TDVP will be discussed in section 5. In section 6 the ideas of time evolutions with Matrix Product Operators will be introduced. Furthermore, for fermionic systems we need to change the Matrix Product Operator of our system to include the fermionic anticommutator relations. This adaptation will be sketched in section 7. In sections 8 and 9 the results of the numerical comparison between TDVP and the Matrix Product Operator will be shown.

1.1 Models

Note that in this thesis we will use atomic units: $\hbar = m_e = (4\pi\epsilon_0)^{-1} = e = 1$. We start with the many body Hamiltonian for electrons in a crystal:

$$\begin{aligned}
\hat{H}_{kin} &= -\frac{1}{2}\sum_i \nabla_i^2 \\
\hat{H}_e &= \frac{1}{2}\sum_{\substack{i,j \\ i\neq j}} \frac{1}{|x_i - x_j|} \\
\hat{H}_{pot} &= \hat{V}(x_1, x_2, \ldots) \\
\hat{H} &= \hat{H}_{kin} + \hat{H}_e + \hat{H}_{pot}
\end{aligned}$$

One possible way to solve this problem is to neglect the electron-electron interaction:

$$\begin{aligned}
\forall \phi, \psi : |\langle\phi|\hat{H}_e|\psi\rangle| \ll |\langle\phi|\Big(\hat{H}_{kin} + \hat{H}_{pot}\Big)|\psi\rangle| \\
\Rightarrow \quad \hat{H} \approx \hat{H}_{kin} + \hat{H}_{pot}
\end{aligned}$$

This is the nearly free electron model and many physical phenomena of solids can be qualitatively explained with this Hamiltonian e.g. the band structure of metals [11], [12]. However there is a class of materials, called strongly correlated materials, where we can not neglect electron-electron interactions to properly describe these materials. The effects of these materials include high temperature superconductivity, the Kondo effect, charge ordering, spin-charge-separation, and many more [12].
One important simplification of the original Hamiltonian yields the so called Hubbard model. Here we transform the Hamiltonian into second quantization, assume a strong electronic shielding of the potential of the nuclei and that the electronic wave functions decay exponentially with respect to position. Furthermore we neglect all core electrons and consider only one valence band (or more). This yields the Hubbard model. It was

introduced by Rudolph Pariser et. al. in 1953 [13], [14] and is named after John Hubbard [15].

Definition 1.1: Hubbard model

Let V be the set of all lattice sites and $\Sigma = \{\uparrow, \downarrow\}$. The Hamiltonian of the Hubbard model is given by:

$$\hat{H} = \sum_{\substack{i,j \in V \\ \sigma \in \Sigma}} t_{ij} c^\dagger_{i\sigma} c_{j\sigma} + U \sum_{i \in V} n_{i\uparrow} n_{i\downarrow}$$

where $\bar{t}_{ij} = t_{ji}$. If the lattice is a simple one dimensional chain with open boundary conditions and $t_{ij} = -e^{-\alpha|j-i|}$ the model will be called long range Hubbard model in this thesis.

The first term of the Hubbard Hamiltonian describes a hopping of electrons between different lattice sites due to the kinetic energy term of the original Hamiltonian. The second term is the repulsive force between electrons of different spin, due to the electron potential. Note that the Pauli principle prohibits two electrons of the same spin occupying on the same site. This is mathematically ensured with the electronic anticommutator relations.

Definition 1.2: Anticommutator relations for electronic creation and annihilation operators

$$\{c_{i\sigma}, c_{j\sigma'}\} = 0 \qquad \{c^\dagger_{i\sigma}, c^\dagger_{j\sigma'}\} = 0 \qquad \{c_{i\sigma}, c^\dagger_{j\sigma'}\} = \delta_{i,j}\delta_{\sigma,\sigma'}$$

The Hubbard model with only nearest neighbor hopping $t_{ij} = \tilde{t}(\delta_{i,j+1} + \delta_{i,j-1})$ and $U \neq 0$ has been solved in one dimension [16]. Furthermore many interesting results about the ground state of the Hubbard model have been obtained [17], [18]. The electronic wave functions centered on one atomic site usually decay with the distance r like $e^{-\alpha r}$ and the tight-binding approximation states that interactions further than the nearest neighbor can be neglected. This approximation may or may not produce qualitatively wrong results. Thus there is a need to investigate systems with long-ranged interactions.

Another important use case for time evolution of systems with long ranged Hamiltonians is the dynamic mean field theory. Here the Hubbard model gets mapped to the Anderson impurity model [19].

Definition 1.3: Anderson impurity model

Let $\Sigma = \{\uparrow, \downarrow\}$, and K be the set of all k-points in K-space. The Hamiltonian of the Anderson impurity model is given by:

$$\hat{H}_{imp} = \epsilon_0 \sum_{\sigma\in\Sigma} n_{0\sigma} + U n_{0\uparrow} n_{0\downarrow}$$
$$\hat{H}_{bath} = \sum_{\substack{k\in K\\ \sigma\in\Sigma}} (\epsilon_k - \mu) n_k$$
$$\hat{H}_{hyb} = \sum_{\substack{k\in K\\ k\neq 0}} \sum_{\sigma\in\Sigma} V_k \left(c^\dagger_{0\sigma} c_{k\sigma} + c^\dagger_{k\sigma} c_{0\sigma} \right)$$
$$\hat{H} = \hat{H}_{imp} + \hat{H}_{bath} + \hat{H}_{hyb}$$

The quantity that needs to be calculated here is the retarded impurity Green function:

$$G_\sigma(t) = -i\theta(t) \langle \{ c_{0\sigma}(t), c^\dagger_{0\sigma}(0) \} \rangle$$

To do that a time evolution of the system needs to be done. Besides total diagonalization, there are several algorithms to calculate the Green function e.g. several quantum Monte Carlo algorithms [5]. It has already been shown that this is also possible within the Matrix Product State formalism [20], [21]. A question of future research will be how the Matrix Product State formalism holds up against these quantum Monte Carlo methods with more orbitals.
It is possible to use the Hubbard model at half filling and a strong repulsion U. Then the electrons will be fully located on one lattice site and we can only consider the spin. One of these models is the transverse field Ising model. This is very similar to the classical Ising model, except that the magnetic field does not point in the same direction as the spin-spin interaction.

Definition 1.4: Transverse field Ising model

Let V be the set of all lattice sites. The Hamiltonian of the transverse field Ising model is given by:

$$\hat{H} = \sum_{i,j\in V} J_{ij} \hat{S}^x_i \hat{S}^x_j - \sum_{i\in V} h_i \hat{S}^z_i$$

In this thesis the model will be called long range transverse field Ising model, if the lattice is a simple one dimensional chain and

$$J_{ij} = \begin{cases} 0 & \text{if } j \leq i \\ -e^{-\alpha(j-i)} & \text{otherwise} \end{cases}$$

In the case of one dimension and only nearest neighbor interactions the model has been solved and has a phase transition in the thermodynamic limit with respect to h [22]. The model also shows some interesting connections to spinless tight-binding fermions [23].

Another interesting model for us with respect to this thesis is the Haldane-Shastry model. This model is similar to the quantum Heisenberg model. The key difference lies within the interaction range of the spins. This interaction range is not limited for the Haldane-Shastry model. This model is especially useful for us, because there is an analytic solution for correlations of the ground state in the thermodynamic limit [24].

Definition 1.5: Haldane-Shastry model

Let V be the set of all lattice sites and let the lattice be a one dimensional chain with open boundary conditions. The Hamiltonian of the Haldane-Shastry model is given by:

$$\hat{H} = \sum_{\substack{i,j \in V \\ j>i}} |j-i|^{-2} \left(\frac{1}{2} \left(\hat{S}_i^+ \hat{S}_j^- + \hat{S}_i^- \hat{S}_j^+ \right) + \hat{S}_i^z \hat{S}_j^z \right)$$

Note that with the methods used in this thesis interactions that decay like r^-2 are not usable directly. We will approximate powers like this with sums of exponential functions that proof to be a very good approximation [25].

$$r^{-2} \approx \sum_n \beta_n e^{-\alpha_n r}$$

Another important use case of long-ranged interactions is the investigation of quasi one dimensional systems [26]. In this systems the second dimensions spans only a couple of lattice sites and the first dimension is longer. In this case we can map the two dimensional system onto a one dimensional system and some of the nearest neighbor interactions of the two dimensional system will be long-ranged in the one dimensional system (see fig. 1.1)
To investigate systems with the the Hamiltonians above there is a range of usable algorithms, most important of which are TEBD (section 4) and DMRG (section 3). In recent years two new algorithms have emerged: TDVP (section 5) and an approximation of the time evolution operator with Matrix Product Operators (section 6). The problem of TEBD is that it relies on a Lie-Trotter decomposition of the system and thus no long-ranged interactions can be examined [27]. DMRG is suitable for long-ranged interactions but mathematically TDVP can find a better approximation of the ground state within the matrix dimension of the matrices we use to describe the state [27]. In the course of this thesis the TDVP for infinite lattices was expanded to calculate the time evolution of systems with long ranged interactions. Furthermore the Finite lattice TDVP and the time evolution with approximate Matrix Product Operators were quantitatively compared.

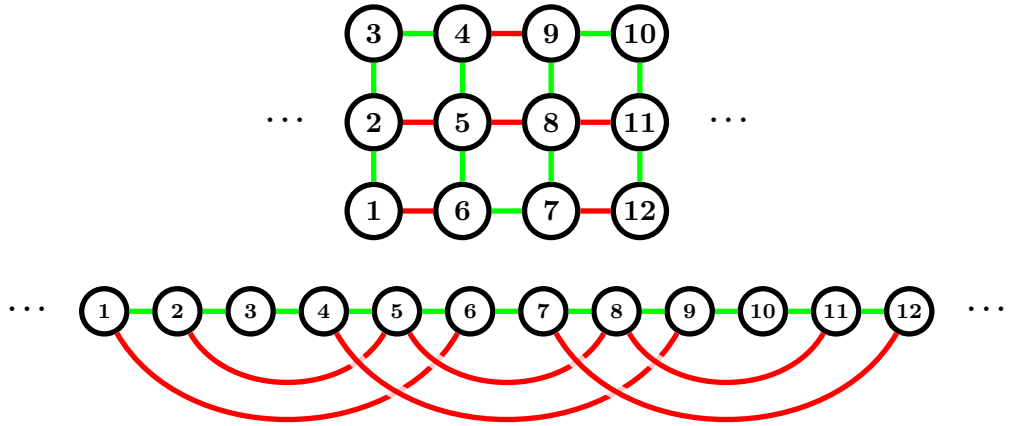

Figure 1.1: Mapping of a two dimensional system onto a one dimensional system. The red lines mark interactions that are nearest neighbor in the two dimensional case but are long-ranged in the one dimensional case.

2 Matrix Product States and Operators

Matrix Product States (MPS) are a powerful and versatile tool to describe and calculate quantum states [8]. When doing numerical calculations about quantum systems it is impossible to store all information about a complicated state in memory, because of the sheer size of the state space. One has to make compromises and only store the most relevant parts of the state. The state is stored as a product of different matrices. The advantage of this approach lies in the fact, that it is possible to reduce the matrix size without loss of relevant information. As a general rule of thumb this is possible in one dimensional systems, but fails in two or more dimensions. This fact is linked to the so called area law of entanglement entropy [8]. The matrix size reduction is done with tools like singular value decomposition (SVD) or QR decomposition. It is also possible to write a quantum operator in a similar fashion. This is necessary if e.g. a certain operator has to be applied to a state. The resulting construct is called Matrix Product Operator (MPO). Several important algorithms have been developed for time evolution (e.g. TEBD) and ground state calculations (e.g. DMRG) and many more applications [8], [28].

2.1 Basics

In the most general case a quantum state of a system with N lattice sites can be written as

$$|\psi\rangle = \sum_{s_1,s_2,\dots s_N=1}^{d} c_{s_1,s_2,\dots,s_N} |s_1 s_2 \dots s_N\rangle$$

Where d is the local Hilbert space dimension, and c is a complex tensor with d^N entries, $c \in \mathbb{C}^{d^N}$. If the system size N is sufficiently large it is not possible to store all entries of d in memory, because the number of elements of d grows exponentially with N. To overcome this problem c can be approximated by a product of matrices.

Definition 2.1: Matrix Product States

A quantum state written or stored in the form

$$|\psi\rangle = \sum_{s_1,s_2,\dots s_N=1}^{d} A^{s_1}_{[1]} A^{s_2}_{[2]} \dots A^{s_N}_{[N]} |s_1 s_2 \dots s_N\rangle$$

is called a Matrix Product State. All A are complex tensors, $A_{[j]} \in \mathbb{C}^{d \times \chi_{L,j} \times \chi_{R,j}}$, where $\chi_{L,j}$ and $\chi_{R,j}$ must be chosen in a way that the matrix product $A^{s_j}_{[j]} A^{s_{j+1}}_{[j+1]}$ is well defined, i.e. $\chi_{R,j} = \chi_{L,j+1}$.

To represent the same state as in the usual notation the condition $A^{s_1}_{[1]} A^{s_2}_{[2]} \dots A^{s_N}_{[N]} = c_{s_1,s_2,\dots,s_N}$ must be true. For every lattice site, there are d different matrices. In practice

the subscript index of the Matrix Product State tensors is omitted and implied by the index of the superscript $A^{s_j} \equiv A^{s_j}_{[j]}$.

Example 2.1: A state with four lattice sites with a singlet on site 3 and 4

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\uparrow\uparrow\downarrow\rangle - |\uparrow\uparrow\downarrow\uparrow\rangle\right)$$

$$A^{\uparrow}_{[1]} = \frac{1}{\sqrt{2}} \qquad A^{\uparrow}_{[2]} = 1 \qquad A^{\uparrow}_{[3]} = \begin{pmatrix} 1 & 0 \end{pmatrix} \qquad A^{\uparrow}_{[4]} = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$$

$$A^{\downarrow}_{[1]} = 0 \qquad A^{\downarrow}_{[2]} = 0 \qquad A^{\downarrow}_{[3]} = \begin{pmatrix} 0 & 1 \end{pmatrix} \qquad A^{\downarrow}_{[4]} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Example 2.2: A state with four lattice sites with a singlet on site 1 and 4

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\uparrow\uparrow\downarrow\rangle - |\downarrow\uparrow\uparrow\uparrow\rangle\right)$$

$$A^{\uparrow}_{[1]} = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \qquad A^{\uparrow}_{[2]} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad A^{\uparrow}_{[3]} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \qquad A^{\uparrow}_{[4]} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$A^{\downarrow}_{[1]} = \begin{pmatrix} 0 & \frac{-1}{\sqrt{2}} \end{pmatrix} \qquad A^{\downarrow}_{[2]} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad A^{\downarrow}_{[3]} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \qquad A^{\downarrow}_{[4]} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

The accuracy of Matrix Product States is limited by the matrix dimension. If the matrix size is too small a Matrix Product State may only be an approximation to the real state $A^{s_1}A^{s_2}\ldots A^{s_N} \approx c_{s_1,s_2,\ldots,s_N}$. However this may not be a problem, if the state is approximated well enough. For a fixed matrix size the limiting factor is the fraction of the state space that is used to describe a state. This fraction is usually very small for one dimensional chains and thus Matrix Product States work so well for these kind of systems. For two, or more dimensions this usually does not hold true. The only exceptions are quasi one dimensional chains, where the second and third dimensions span only over a limited number of sites. One important fact to note is that the MPS representation of a state is not unique. It is possible to multiply an invertible matrix G to A^{s_i} from the right and G^{-1} to $A^{s_{i+1}}$ from the left and the state is not changed.

$$
\begin{aligned}
|\psi\rangle &= \sum_{s_1,s_2,\dots s_N=1}^{d} \dots \tilde{A}^{s_i}\tilde{A}^{s_{i+1}} \dots |s_1 s_2 \dots s_N\rangle \\
&= \sum_{s_1,s_2,\dots s_N=1}^{d} \dots A^{s_i} \underbrace{GG^{-1}}_{=1} A^{s_{i+1}} \dots |s_1 s_2 \dots s_N\rangle \\
&= \sum_{s_1,s_2,\dots s_N=1}^{d} \dots A^{s_i}A^{s_{i+1}} \dots |s_1 s_2 \dots s_N\rangle
\end{aligned}
$$

There are several conventions that take advantage of this gauge invariance, the most important of which are left-canonical, right-canonical, mixed-canonical, and canonical Matrix Product States. This gauges are very important when doing numerical calculations to save computation time. More on that is to be read in the chapter of normalization.
The idea of Matrix Product States can also be applied to operators. In the most general form an operator is written as

$$
\hat{O} = \sum_{s_1,s_1',\dots=1}^{d} c_{s_1,s_1',\dots,s_L,s_N'} |s_1 s_2 \dots s_N\rangle \langle s_1' s_2' \dots s_N'|
$$

Definition 2.2: Matrix Product Operators

An operator written or stored in the form

$$
\hat{O} = \sum_{s_1,s_1',\dots=1}^{d} W_{[1]}^{s_1,s_1'} W_{[2]}^{s_2,s_2'} \dots W_{[N]}^{s_N,s_N'} |s_1 s_2 \dots s_N\rangle \langle s_1' s_2' \dots s_N'|
$$

is called a Matrix Product Operator. All W are complex tensors, $W_{[j]} \in \mathbb{C}^{d\times d\times \chi_{L,j}\times\chi_{R,j}}$, where $\chi_{L,j}$ and $\chi_{R,j}$ must be chosen in a way that the matrix product $W_{[j]}^{s_j,s_j'} W_{[j]}^{s_{j+1},s_{j+1}'}$ is well defined, i.e. $\chi_{R,j} = \chi_{L,j+1}$.

Like with Matrix Product States, the subscript of the tensors W is usually omitted and implied by the index of the superscript, $W^{s_j,s_j'} \equiv W_{[j]}^{s_j,s_j'}$. The needed matrix dimensions of $W_j^{s_j,s_j'}$ is determined by the interaction range of the operator $\hat{O}$. Note that this just a rule of thumb, as there are Matrix Product Operators with infinite range that have a relatively small size.

Example 2.3: Anisotropic quantum Heisenberg model

Let us take a look at the Hamiltonian for the anisotropic quantum Heisenberg model.

$$\hat{H} = \sum_{i=1}^{N-1} \frac{J_{xy}}{2} \left(\hat{S}_i^+ \hat{S}_{i+1}^- + \hat{S}_i^- \hat{S}_{i+1}^+ \right) + J_z \hat{S}_i^z \hat{S}_{i+1}^z - \sum_{i=1}^{N} h_i \hat{S}_i^z$$

In MPO notation this Hamiltonian can be written as

$$W^{s_1,s_1'} = \begin{pmatrix} 1 & J_z \hat{S}^z & \frac{J_{xy}}{2} \hat{S}^+ & \frac{J_{xy}}{2} \hat{S}^- & -h_i \hat{S}^z \end{pmatrix}$$

$$W^{s_i,s_i'} = \begin{pmatrix} 1 & J_z \hat{S}^z & \frac{J_{xy}}{2} \hat{S}^+ & \frac{J_{xy}}{2} \hat{S}^- & -h_i \hat{S}^z \\ 0 & 0 & 0 & 0 & \hat{S}^z \\ 0 & 0 & 0 & 0 & \hat{S}^- \\ 0 & 0 & 0 & 0 & \hat{S}^+ \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \forall 1 < i < N$$

$$W^{s_N,s_N'} = \begin{pmatrix} -h_i \hat{S}^z \\ \hat{S}^z \\ \hat{S}^- \\ \hat{S}^+ \\ 1 \end{pmatrix}$$

Example 2.4: TFI model with long range interactions

The transverse field Ising model with infinite exponential decaying interaction range.

$$\hat{H} = J \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \hat{S}_i^x \hat{S}_j^x e^{-\alpha(j-i)} - h \sum_{i=1}^{N} \hat{S}_i^z$$

Where α is a positive real number. In MPO notation this Hamiltonian can be written as

$$W^{s_1,s_1'} = \begin{pmatrix} 1 & Je^{-\alpha} \hat{S}^x & \hat{S}^z \end{pmatrix}$$

$$W^{s_i,s_i'} = \begin{pmatrix} 1 & Je^{-\alpha} \hat{S}^x & \hat{S}^z \\ 0 & e^{-\alpha} & \hat{S}_x \\ 0 & 0 & 1 \end{pmatrix} \forall 1 < i < N$$

$$W^{s_N,s_N'} = \begin{pmatrix} \hat{S}^z \\ \hat{S}_x \\ 1 \end{pmatrix}$$

Note that the usual convention to write the matrix $W^{s_i,s_i'}$ can be a bit confusing. The

operators as matrix elements do not mean that $W^{s_i,s_i'}$ is in block form, but that the indices s_i, s_i' get "carried" into the matrix elements. Single scalars are treated as scalar times identity matrix. The representation depends also in the chosen basis. For example the the bulk element from last Hamiltonian in z-basis:

$$W^{\uparrow,\downarrow} = \begin{pmatrix} 1\cdot\mathbb{1}(\uparrow,\downarrow) & Je^{-\alpha}\hat{S}^x(\uparrow,\downarrow) & \hat{S}^z(\uparrow,\downarrow) \\ 0 & e^{-\alpha}\cdot\mathbb{1}(\uparrow,\downarrow) & \hat{S}_x(\uparrow,\downarrow) \\ 0 & 0 & 1\cdot\mathbb{1}(\uparrow,\downarrow) \end{pmatrix} = \begin{pmatrix} 0 & Je^{-\alpha}\frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 \end{pmatrix}$$

From the second to the third equal sign we used $\mathbb{1}(\uparrow,\downarrow) = \hat{S}^z(\uparrow,\downarrow) = 0$ and $\hat{S}^x(\uparrow,\downarrow) = \frac{1}{2}$. Note that these exact relations only hold true in z-basis. For the x- and y-basis different representations emerge.
How can an MPO be applied to an MPS? For the first step we write $\hat{O}\,|\psi\rangle$ in their explicit matrix product forms and use the fact that $\langle \boldsymbol{s'}|\boldsymbol{s''}\rangle = \delta_{\boldsymbol{s'},\boldsymbol{s''}}$.

$$\begin{aligned}
\hat{O}\,|\psi\rangle &= \sum_{s_1,s_1',s_1''\ldots=1}^{d} \left(\prod_i W^{s_i,s_i'}\right)\left(\prod_i A^{s_i''}\right)|s_1 s_2 \ldots s_N\rangle\,\langle s_1' s_2' \ldots s_N'|s_1'' s_2'' \ldots s_N''\rangle \\
&= \sum_{s_1,s_1'\ldots=1}^{d} \left(\prod_i W^{s_i,s_i'}\right)\left(\prod_i A^{s_i'}\right)|s_1 s_2 \ldots s_N\rangle \\
&= \sum_{s_1,s_1'\ldots=1}^{d} \sum_{a_1,a_2,\ldots b_1,b_2\ldots} W_{1,b_1}^{s_1,s_1'} W_{b_1,b_2}^{s_2,s_2'} \ldots W_{b_{N-1},1}^{s_N,s_N'} A_{1,a_1}^{s_1'} A_{a_1,a_2}^{s_2'} \ldots A_{a_{N-1},1}^{s_N'}\,|s_1 s_2 \ldots s_N\rangle
\end{aligned}$$

In the third line we rewrote the matrix products in its explicit form. In the next step we realign the different tensors in way so that they form again a Matrix Product State.

$$\begin{aligned}
\hat{O}\,|\psi\rangle &= \sum_{s_1,s_1'\ldots=1}^{d} \sum_{a_1,a_2,\ldots b_1,b_2\ldots} \left(W_{1,b_1}^{s_1,s_1'} A_{1,a_1}^{s_1'}\right)\left(W_{b_1,b_2}^{s_2,s_2'} A_{a_1,a_2}^{s_2'}\right)\ldots\left(W_{b_{N-1},1}^{s_N,s_N'} A_{a_{N-1},1}^{s_N'}\right)|s_1 \ldots\rangle \\
&= \sum_{s_1,s_2\ldots s_N=1}^{d} \tilde{A}^{s_1}\tilde{A}^{s_2}\ldots\tilde{A}^{s_N}\,|s_1 s_2 \ldots s_N\rangle
\end{aligned}$$

where $\tilde{A}^{s_i}_{\tilde{a}_{i-1},\tilde{a}_i} = \sum_{s_i'=1}^{d} W^{s_i,s_i'}_{b_{i-1},b_i} A^{s_i'}_{a_{i-1},a_i}$, and $\tilde{a}_i = (b_i, a_i)$ as a combined new tensor index. The last line of the equation above is by definition a Matrix Product State. Note that we inflated the bond dimension of the new MPS tensors by the bond dimension of the MPO tensor. If a numerical calculation applies an MPO to an MPS several times, the bond dimension of the MPS needs to be reduced after every step or the bond dimension grows without limit. This can be done with a singular value decomposition. Note that

this approach is in general computationally to expensive and other methods to apply an MPO to an MPS must be used [29].

2.2 Graphical representation

There is a graphical representation for Matrix Product States and operators. This is used, because in graphical notation it is sometimes easier to see what is calculated. Matrices and Vectors are represented by circles, where horizontal lines stand for the matrix and vector indices, while vertical lines represent the spin index.

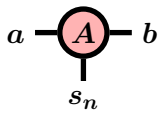

Figure 2.1: Graphical representation of $A^{s_n}_{a,b}$.

Every matrix and vector which are connected with lines are multiplied with each other. Connected lines mean that there is a summation over this index. This holds true for matrix indices and spin indices.

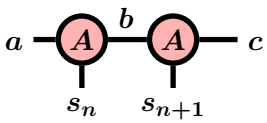

Figure 2.2: Graphical representation of $\sum_b A^{s_n}_{a,b} A^{s_{n+1}}_{b,c} = A^{s_n} A^{s_{n+1}}$.

If the line for the spin shows upwards, the matrix is complex conjugated.

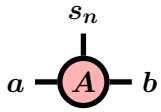

Figure 2.3: Graphical representation of $\bar{A}^{s_n}_{a,b}$.

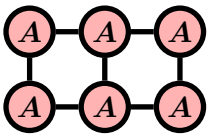

Figure 2.4: Graphical representation of $\langle\psi|\psi\rangle$ with three lattice sites. The usual convention is that lattice sites with a lower index are to the left.

In fig. 2.4 the norm squared of a state with three lattice sites can be seen. In mathematical terms this tensor network reads as:

$$\begin{aligned}\langle\psi|\psi\rangle &= \sum_{a,b}\sum_{\bar{a},\bar{b}}\sum_{s_1,s_2,s_3} A_a^{s_1} A_{a,b}^{s_2} A_c^{s_3} \bar{A}_{\bar{a}}^{s_1} \bar{A}_{\bar{a},\bar{b}}^{s_2} \bar{A}_{\bar{c}}^{s_3} \\ &= \sum_{s_1,s_2,s_3} \left(A^{s_1} A^{s_2} A^{s_3}\right)\left(\bar{A}^{s_1} \bar{A}^{s_2} \bar{A}^{s_3}\right) \\ &= \sum_{s_1,s_2,s_3} \left(A^{s_1} A^{s_2} A^{s_3}\right)\left(A^{s_3\dagger} A^{s_2\dagger} A^{s_1\dagger}\right)\end{aligned}$$

With the same rules as with Matrix Product States we can represent Matrix Product Operators. In fig. 2.5 an example of a tensor of a Matrix Product Operator can be seen.

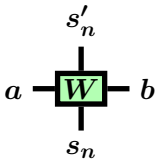

Figure 2.5: Graphical representation of $W_{a,b}^{s_n,s'_n}$ of a Matrix Product Operator.

2.3 Normalization Conditions

Up to this point we never calculated the norm squared $\langle\psi|\psi\rangle$ of a Matrix Product State. Usually in numerical calculations the norm of a Matrix Product State is fixed to 1. This is ensured with normalization conditions. There is various number of different normalization conditions for Matrix Product States. Note that it is possible and often necessary to move from one normalization to another one (e.g. calculate the mixed-canonical representation from the left-canonical representation). The transformation from one representation to another one is usually done by a series of repeated singular value decompositions. All normalization conditions have in common that the norm squared $\langle\psi|\psi\rangle$ is 1, if the tensors and matrices satisfy certain conditions. Since a Matrix Product State consist of tensors it is a good starting point to fix the normalization there. We start by defining left- and right-normalized tensors.

Definition 2.3: Left- and right-normalized tensors

If the a tensor A of a Matrix Product States satisfies the condition

$$\sum_s A^{s\dagger} A^s = \mathbb{1}$$

it is called left-normalized. Similarly, if the a tensor A of a Matrix Product States satisfies the condition

$$\sum_s A^s A^{s\dagger} = \mathbb{1}$$

it is called right-normalized. In this thesis left-normalized tensors are denoted by L, right-normalized by R.

Figure 2.6: Graphical representation of the conditions for left-normalized tensors and right-normalized tensors.

The left- and right-normalization are very important and handy properties, because they save computation time. As can be seen in fig. 2.6 the tensor network collapses on its own if the tensors are normalized. Now we want to take a look what happens to the norm of a Matrix Product State if all its tensors are normalized.

Definition 2.4: Left- and right-canonical MPS

A left-canonical Matrix Product State consists only of left-normalized tensors. A right-canonical Matrix Product State consists only of right-normalized tensors.

Theorem 2.1

The norm squared $\langle\psi|\psi\rangle$ of every left-canonical and right-canonical Matrix Product State is 1.

Proof. Let $|\psi\rangle$ be a left-canonical Matrix Product State.

$$\begin{aligned}
\langle\psi|\psi\rangle &= \sum_{\boldsymbol{s},\boldsymbol{s'}} \left(L^{s'_N\dagger}\dots L^{s'_2\dagger}L^{s'_1\dagger}\right)\left(L^{s_1}L^{s_2}\dots L^{s_N}\right)\langle\boldsymbol{s'}|\boldsymbol{s}\rangle \\
&= \sum_{s_2,\dots,s_N} L^{s_N\dagger}\dots L^{s_2\dagger}\underbrace{\left(\sum_{s_1} L^{s_1\dagger}L^{s_1}\right)}_{=\mathbb{1}} L^{s_2}\dots L^{s_N} \\
&= \sum_{s_3,\dots,s_N} L^{s_N\dagger}\dots\underbrace{\left(\sum_{s_2} L^{s_2\dagger}L^{s_2}\right)}_{=\mathbb{1}}\dots L^{s_N} \\
&\dots \\
&= \sum_{s_N} L^{s_N\dagger}L^{s_N} = 1
\end{aligned}$$

It can be shown in the same way that the norm of a right-canonical state is 1. □

This is a very important theorem. If we ensure that all tensors are right- or left-normalized during our numerical computation, we know that the our Matrix Product State will be normed to 1, without explicitly calculating the norm. However it is not possible to keep the state right-canonical or left-canonical all the time, if we e.g. apply an operator or calculate the ground state. Another downside is that it is computationally very expensive to compute expectation values $\langle\psi|\hat{O}|\psi\rangle$ of operators with this normalization. Thus there is a need for other normalizations.

Definition 2.5: Schmidt decomposition

A quantum lattice is split up into two separate systems $\mathcal{A}$ and $\mathcal{B}$. Let $|a\rangle_{\mathcal{A}}$ and $|a\rangle_{\mathcal{B}}$ be orthonormal basis sets in each sublattice. The Schmidt decomposition of state $|\psi\rangle$ is given by

$$|\psi\rangle = \sum_a \lambda_a |a\rangle_{\mathcal{A}} |a\rangle_{\mathcal{B}}$$

where every $\lambda_a \in \mathbb{R}$ and $\lambda \geq 0$. The λ_a are called Schmidt coefficients.

Theorem 2.2

A state $|\psi\rangle$ is normed to 1, if and only if the sum of all squared Schmidt coefficients λ_a^2 in the Schmidt decomposition is equal to 1.

$$\sum_a \lambda_a^2 = 1 \quad \Leftrightarrow \quad \langle\psi|\psi\rangle = 1$$

Proof.

$$\langle\psi|\psi\rangle = \sum_{a,a'} \lambda_{a'}\lambda_a \langle a'|a\rangle_{\mathcal{A}} \langle a'|a\rangle_{\mathcal{B}}$$

By definition $|a\rangle_{\mathcal{A}}$ and $|a\rangle_{\mathcal{B}}$ are orthonormal basis sets, thus $\langle a'|a\rangle_{\mathcal{A}} = \langle a'|a\rangle_{\mathcal{B}} = \delta_{a,a'}$

$$\langle\psi|\psi\rangle = \sum_a \lambda_a^2 = 1$$

□

We will use the Schmidt decomposition to calculate the norm of mixed-canonical Matrix Product States that are defined below. In general the Schmidt decomposition can be used to calculate the reduced density matrices of the subsystems $\mathcal{A}$ and $\mathcal{B}$, and the von Neumann entropy of the entanglement [8].

Definition 2.6: Mixed-Canonical MPS

A Matrix Product State written or stored in the form

$$|\psi\rangle = \sum_{s_1,s_2,\ldots s_N=1}^{d} L^{s_1}L^{s_2}\ldots L^{s_{i-1}}SR^{s_i}\ldots R^{s_N}\,|s_1 s_2\ldots s_N\rangle$$

where every tensor L is left-normalized, every tensor R is right-normalized, and S is a matrix, is called mixed-canonical Matrix Product State. S will be denoted as the center matrix.

Theorem 2.3

Let $|\psi\rangle$ be a mixed-canonical Matrix Product State, and S, its center matrix, be diagonal with only real non-negative elements. The diagonal entries $S_{a,a}$ are then the Schmidt coefficients λ_a of the state $|\psi\rangle$.

Proof. Define $|a\rangle_{\mathcal{A}} = \sum_{s_1,s_2,\ldots,s_{i-1}} (L^{s_1}L^{s_2}\ldots L^{s_{i-1}})_{1,a}\,|s_1 s_2\ldots s_{i-1}\rangle$ and $|a\rangle_{\mathcal{B}} = \sum_{s_i,s_{i+1},\ldots,s_N} (R^{s_i}R^{s_{i+1}}\ldots R^{s_N})_{a,1}\,|s_i s_{i+1}\ldots s_N\rangle$. The mixed-canonical state then transforms into the same form as the Schmidt decomposition.

$$|\psi\rangle = \sum_a S_{a,a}\,|a\rangle_{\mathcal{A}}\,|a\rangle_{\mathcal{B}}$$

Now it is sufficient to show that $|a\rangle_{\mathcal{A}}$ and $|a\rangle_{\mathcal{B}}$ form orthonormal basis sets. Then the equation above is indeed the Schmidt decomposition and the diagonal elements

$S_{a,a}$ are the Schmidt coefficients.

$$\begin{aligned}
\langle a'|a\rangle &= \sum_{\boldsymbol{s'},\boldsymbol{s}} \left(L^{s'_{i-1}\dagger}\dots L^{s'_2\dagger}L^{s'_1\dagger}\right)_{a',1} \left(L^{s_1}L^{s_2}\dots L^{s_{i-1}}\right)_{1,a} \langle \boldsymbol{s'}|\boldsymbol{s}\rangle \\
&= \sum_{s_2,s_3,\dots,s_{i-1}} (L^{s_{i-1}\dagger}\dots L^{s_2\dagger} \underbrace{\left(\sum_{s_1} L^{s_1\dagger}L^{s_1}\right)}_{=\mathbb{1}} L^{s_2}\dots L^{s_{i-1}})_{a',a} \\
&= \sum_{s_3,\dots,s_{i-1}} (L^{s_{i-1}\dagger}\dots \underbrace{\left(\sum_{s_2} L^{s_2\dagger}L^{s_2}\right)}_{=\mathbb{1}} \dots L^{s_{i-1}})_{a',a} \\
&= (\sum_{s_{i-1}} L^{s_{i-1}\dagger}L^{s_{i-1}})_{a',a} \\
&= (\mathbb{1})_{a',a} = \delta_{a',a}
\end{aligned}$$

In the same way it can be shown, that the $|a\rangle_{\mathcal{B}}$ form also a orthonormal basis. □

Theorem 2.4

Let $|\psi\rangle$ be a mixed-canonical Matrix Product State, and S, its center matrix, be diagonal with only real non-negative elements. The state is normed to 1, if and only if the sum over all squared diagonal entries of S is 1.

$$\sum_a S_{a,a}^2 = 1 \quad \Leftrightarrow \quad \langle\psi|\psi\rangle = 1$$

Proof. We showed in thm. 2.3 that the diagonal elements of S are equal to the Schmidt coefficients of $|\psi\rangle$. According to thm. 2.2 the state is normed, if the sum of the squared Schmidt coefficients is 1. □

Note that for an arbitrary mixed-canonical Matrix Product State one can shift the site where the center matrix is located to the right or to the left by performing a singular value decomposition (for the exact procedure see [8]). If the center matrix is pushed to the boundary of the system, we will have a left-canonical or right-canonical Matrix Product State. This is how we can jump between those three normalizations. Another option to write a Matrix Product State is the $\lambda\Gamma$-notation introduced by Guifré Vidal in 2003 [30].

Definition 2.7: Canonical MPS

A Matrix Product State written or stored in the form

$$|\psi\rangle = \sum_{s_1,s_2,...s_N=1}^{d} \lambda_0\Gamma^{s_1}\lambda_1\Gamma^{s_2}\lambda_2\Gamma^{s_3}...\lambda_{N-1}\Gamma^{s_N}\lambda_N\,|s_1 s_2...s_N\rangle$$

is called Matrix Product State in $\lambda\Gamma$-notation. Here every λ_i is a square diagonal matrix with real entries. The two matrices at the boundaries are scalars and fixed to the value 1, $\lambda_0 = \lambda_N = 1$. If for every i the tensor $A^{s_i} = \lambda_{i-1}\Gamma^{s_i}$ is left-normalized and $A^{s_i} = \Gamma^{s_i}\lambda_i$ is right-normalized it is called a canonical Matrix Product State.

Theorem 2.5

The norm squared $\langle\psi|\psi\rangle$ of every canonical Matrix Product State is 1.

Proof. Let $|\psi\rangle$ be a canonical Matrix Product State with matrices λ_i and Γ^{s_i}. By definition we can construct an equivalent left-canonical Matrix Product State by computing a left-normalized tensor for every lattice site: $L^{s_i} = \lambda_{i-1}\Gamma^{s_i}$. We have shown above that every left-canonical Matrix Product State has a norm of 1. □

The $\lambda\Gamma$-notation is the most flexible of the representations defined above, because we can calculate the other three normalization just with matrix multiplications. Furthermore calculating expectation values of operators $\langle\psi|\hat{O}|\psi\rangle$ is very fast in this representation, if the operator acts only locally.

2.4 Uniform Matrix Product States

Uniform Matrix Product States are a subset of conventional Matrix Product State that describe states in the thermodynamic limit [27]. They are essential for the Time-Dependent Variational Principle (TDVP) on infinite lattices and as we can see below are by definition translational invariant.

Definition 2.8: uMPS

A Matrix Product State $|\psi\rangle$ of a system in the thermodynamic limit ($N \to \infty$), where every tensor is the same $A_{[i]} = A_{[j]}\ \forall i, j$, is called uniform Matrix Product State (uMPS).

$$|\psi(A)\rangle = \sum_{\{\boldsymbol{s}\}} v_l^\dagger \left(\prod_{n\in\mathbb{Z}} A^{s_n}\right) v_r\,|\boldsymbol{s}\rangle$$

The two objects v_l and v_r are complex column vectors. The set of integers $\mathbb{Z}$ indicate the infinite number of lattice sites.

Note that there is the same gauge invariance as with finite Matrix Product States. If A^s is multiplied from both sides with an arbitrary invertible matrix G, $A^s \to G^{-1}A^sG$ the same state $|\psi\rangle$ arises.

$$\begin{aligned}&...G^{-1}A^{s_{-2}}G\ G^{-1}A^{s_{-1}}G\ G^{-1}A^{s_0}G\ G^{-1}A^{s_1}G\ G^{-1}A^{s_2}G...\\ =&...A^{s_{-2}}A^{s_{-1}}A^{s_0}A^{s_1}A^{s_2}...\end{aligned}$$

We are going to use different normalization conditions for uniform Matrix Product States, although it is possible to define the same normalization conditions as for finite Matrix Product States. To ensure normalization for uniform Matrix Product States we will fix the dominant eigenvalue of the transfer matrix.

Definition 2.9: Transfer matrix

The transfer matrix of a uniform Matrix Product State is defined as

$$T = \sum_{s=1}^{d} A^s \otimes \bar{A}^s$$

For this thesis it is assumed that its dominant left and right eigenvectors denoted as $\langle l|$ and $|r\rangle$ are unique (i.e. the dominant eigenvalue is not degenerate).

With the dominant eigenvalues of the transfer matrix we define our set where the uniform matrix product are from.

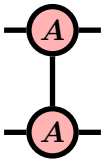

Figure 2.7: Graphical representation of the transfer matrix.

The usage of the transfer matrix may seem a little bit odd at first, but we will link the norm of a uniform Matrix Product State to the dominant eigenvalue of T down below. This becomes more clear with a look an fig. 2.8.
The transfer matrix can be calculated explicitly as follows

$$T_{(a,b),(c,d)} = \sum_{s=1}^{d} A^s_{a,c} \cdot \bar{A}^s_{b,d}$$

Where (a, b) means that the two indices a and b get grouped into a new index. The transfer matrix in its explicit form is a $\chi^2 \cdot \chi^2$ matrix. For that reason it is in general not a good idea to construct the explicit form of the transfer matrix and do calculations with it. The computational cost is too high (see below).

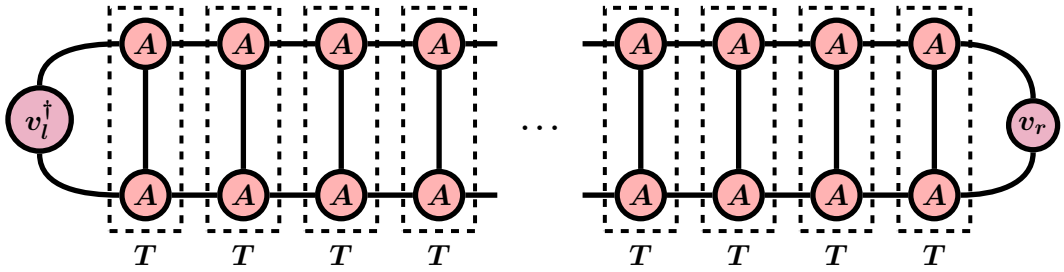

Figure 2.8: Norm squared of a uniform Matrix Product State. Note the emergence of a product of transfer matrices.

Because of the special structure of T it is possible to find alternative methods to calculate e.g. a vector matrix product. In general a vector that can be applied to the transfer matrix from the left or from the right has χ^2 entries. It is possible to transform these vectors into $\chi \cdot \chi$ matrices with the transformation

$$\langle p|_{1,(a,b)} = p_{b,a} \qquad |q\rangle_{(a,b),1} = q_{a,b}$$

where $\langle p|$ is an arbitrary row vector and p its matrix representation and similar for the column vector $|q\rangle$. Note that the order of indices changed in the row vector to matrix transformation. The process of applying the transfer matrix to a row vector transforms then into

$$\begin{aligned}
\Big(\langle p|\, T\Big)_{1,(c,d)} &= \sum_{a,b} \langle p|_{1,(a,b)}\ T_{(a,b),(c,d)} \\
&= \sum_{a,b} p_{b,a}\ T_{(a,b),(c,d)} \\
&= \sum_{s,a,b} \bar{A}^s_{b,d}\ p_{b,a}\ A^s_{a,c} \\
&= \Big(\sum_s A^{s\dagger} p A^s\Big)_{d,c}
\end{aligned}$$

In the same manner the process of applying T to a column vector $T\,|q\rangle$ can be transformed into $\sum_s A^s q A^{s\dagger}$. Maybe the most important use case for this are the eigenvalue equations for the transfer matrix.

$$\begin{aligned}
\langle l|\, T = \eta_l\, \langle l| \quad &\Leftrightarrow \quad \sum_{s=1}^{d} A^{s\dagger} l A^s = \eta_l l \\
T\,|r\rangle = \eta_r\,|r\rangle \quad &\Leftrightarrow \quad \sum_{s=1}^{d} A^s r A^{s\dagger} = \eta_r r
\end{aligned} \tag{2.1}$$

If one wants to solve the eigenproblem for the transfer matrix it is best to use the two equations on the right hand side, because they scale with a computational complexity like $\mathcal{O}(\chi^3)$ in comparison to $\mathcal{O}(\chi^6)$ for the two equations on the left hand side.
Another important transformation is the one of the vector product of a row and column vector.

$$\begin{aligned}\langle p|q\rangle &= \sum_{a,b} \langle p|_{1,(a,b)} \ \ |q\rangle_{(a,b),1} \\ &= \sum_{a,b} p_{b,a} q_{a,b} \\ &= \mathrm{tr}(pq)\end{aligned}$$

Theorem 2.6

Let T be the transfer matrix of a uniform Matrix Product State and let $\langle l|$ and $|r\rangle$ be the eigenvectors corresponding to its dominant eigenvalue η. Let $\eta = 1$. The matrix representations r and l of the two vectors can be chosen to be hermitian.

Proof. Let $\tilde{l}$ be a matrix that is not hermitian, but saves the eigenvalue equation for T. If we take the conjugate transpose of the eigenvalue equation for $\tilde{l}$

$$\sum_s A^{s\dagger} \tilde{l} A^s = \tilde{l} \quad \Leftrightarrow \quad \sum_s A^{s\dagger} \tilde{l}^\dagger A^s = \tilde{l}^\dagger$$

we see that $\tilde{l}^\dagger$ also solves the eigenvalue equation. By assumption the dominant eigenvalue of T is not degenerate. Thus $\tilde{l}$ and $\tilde{l}^\dagger$ can only be different by a phase factor.

$$\tilde{l}^\dagger = e^{i\varphi} \tilde{l}$$

If we set $l = e^{i\frac{1}{2}\varphi} \tilde{l}$, we found a matrix that clearly solves the eigenequation and is hermitian.

$$l^\dagger = e^{-i\frac{1}{2}\varphi} \tilde{l}^\dagger = e^{-i\frac{1}{2}\varphi} e^{i\varphi} \tilde{l} = e^{i\frac{1}{2}\varphi} \tilde{l} = l$$

In the same way it can be shown that r can be chosen to be hermitian. □

In numerical calculations when we find a matrix $\tilde{l}$ that solves the eigenequation but is not hermitian, we do not need to calculate φ from the proof above. We can calculate the hermitian matrix with $l = \frac{1}{c}\tilde{l}$, where c is the diagonal element of $\tilde{l}$ with the greatest absolute value. The same holds true for the right hand side eigenvector. Theoretically we don't need to take the maximum absolute value of the diagonal and divide $\tilde{l}$ by any of its diagonal elements. The problem with that approach is that we might use a diagonal element that is almost zero and thus amplify numerical noise. It is going to be important for the Time-Dependent Variational Principle algorithm, that we will discuss later, that the matrix representations of the eigenvectors will be hermitian.

Theorem 2.7

Let T be the transfer matrix of a uniform Matrix Product State and let $\langle l|$ and $|r\rangle$ be the eigenvectors corresponding to its dominant eigenvalue η. Let $\eta = 1$. Let the matrix representations of the vectors l and r be hermitian. The vector product $\langle l|r\rangle$ then must be real.

Proof. The vector product in matrix representation is

$$\mathrm{tr}(lr) = c$$

If we look at the complex conjugate of this we see that

$$c^* = \sum_{a,b} l^*_{a,b} r^*_{b,a} = \sum_{a,b} l^\dagger_{b,a} r^\dagger_{a,b} = \mathrm{tr}(r^\dagger l^\dagger) = \mathrm{tr}(lr) = c$$

After the fourth equation sign we used the cyclic invariance of the trace and that the matrices are hermitian. c must be real, because $c = c^*$. □

That the product $\langle l|r\rangle$ is real will be important later, because we will link the norm of a uniform Matrix Product State to this product, which in fact needs to be real. If we have a uniform Matrix Product State with a tensor A and the dominant eigenvalue is not equal to 1, $\eta \neq 1$, there is a simple renormalization procedure.

$$A_{\mathrm{new}} = \frac{1}{\sqrt{\eta}} A_{\mathrm{old}} \tag{2.2}$$

This division only changes the norm of the state, as desired.

Theorem 2.8

Let $|\psi\rangle$ be a uniform Matrix Product State and let T be its transfer matrix. The norm $\sqrt{\langle\psi|\psi\rangle}$ is finite, if and only if the dominant eigenvalue η is equal to 1.

Proof. The norm squared in a more explicit form is

$$\langle\psi|\psi\rangle = \lim_{N\to\infty} \sum_{\boldsymbol{s}} \left(\bar{v}_l^\dagger \bar{A}^{s_a} \bar{A}^{s_b} \dots \bar{A}^{s_z} \bar{v}_r\right) \left(v_l^\dagger A^{s_a} A^{s_b} \dots A^{s_z} v_r\right)$$

where a, b,... are just some arbitrary index names. If we rearrange the sum, we can see that the transfer matrix emerges. This is easier to see in fig. 2.8.

$$\langle\psi|\psi\rangle = \lim_{N\to\infty} \left(v_l^\dagger \otimes \bar{v}_l^\dagger\right) \left(\sum_{s_a} A^{s_a} \otimes \bar{A}^{s_a}\right) \left(\sum_{s_b} A^{s_b} \otimes \bar{A}^{s_b}\right) \dots \left(\sum_{s_z} A^{s_z} \otimes \bar{A}^{s_z}\right) \left(v_r \otimes \bar{v}_r\right)$$

Now we can see that in the middle of the equation above there is by definition a product of transfer matrices.

$$\langle\psi|\psi\rangle = \lim_{N\to\infty} \left(v_l^\dagger \otimes \bar{v}_l^\dagger\right) T^N \left(v_r \otimes \bar{v}_r\right)$$

For the next step we perform a total diagonalization of the transfer matrix T.

$$\langle\psi|\psi\rangle = \lim_{N\to\infty}\Big(v_l^\dagger \otimes \bar{v}_l^\dagger\Big) P\lambda^N P^{-1}\Big(v_r \otimes \bar{v}_r\Big)$$

Here λ is the diagonal matrix of eigenvalues. Now we can see that the limit $\lim_{N\to\infty}\lambda^N$ and with it the norm will not converge if the dominant eigenvalue is greater than 1. Similarly the norm will be zero in the limit if all eigenvalues are smaller than 1. □

Now we have shown that only the eigenvectors of the dominant eigenvalue will survive when we calculate the norm of uniform Matrix Product States. This is why we can set the norm squared of the state to be $\langle\psi|\psi\rangle = \langle l|r\rangle = \mathrm{tr}(lr) = 1$. Any other positive real value would be equally valid, but would cancel out anyway if calculate the interesting quantities of the form $\frac{\langle\psi|\hat{O}|\psi\rangle}{\langle\psi|\psi\rangle}$. In the proof above we can also see why the the two boundary vectors v_l and v_r don't contribute to the physics of the system like it was claimed at the beginning of this section. When calculating the norm or expectation values only the overlap with the dominant eigenvectors "survives" the multiplication with $\lim_{N\to\infty} T^N$.

Algorithm 2.1: Normalizing a uniform Matrix Product State

In general when one starts with a random or arbitrary uniform Matrix Product State (i.e. a random complex tensor A), the normalization procedure is as follows.

1. *Calculate dominant eigenvalue η of the transfer matrix T. This is to be done with the matrix form of the eigenvalue equation $\sum_s A^{s\dagger} l A^s = \eta l$ to save computation time. For example in Python this can be done with the package scipy using scipy.sparse.linalg.LinearOperator and scipy.sparse.linalg.eigs [31].*
2. *Divide A by $\sqrt{\eta}$. This ensures that the dominant eigenvalue is equal to 1.*
3. *Compute the dominant eigenvectors $\langle l|$ and $|r\rangle$ and ensure that their matrix representations are hermitian. This again needs to be done with the eigenvalue equation in matrix form, $\sum_s A^{s\dagger} l A^s = l$ and $\sum_s A^s r A^{s\dagger} = r$.*
4. *Divide r and l by a certain factor so that $\langle l|r\rangle = \mathrm{tr}(lr) = 1$.*

If one has stored a normed uMPS state an expectation value of an operator acting on k neighboring sites can be calculated like this:

$$\frac{\langle\psi|\hat{O}|\psi\rangle}{\langle\psi|\psi\rangle} = \sum_{\{s_n\},\{t_n\}=1}^{d} \langle s_1 s_2 \dots s_k|\hat{O}|t_1 t_2 \dots t_k\rangle \ \langle l|\left[(A^{t_1}A^{t_2}\dots A^{t_k})\otimes(\bar{A}^{s_1}\bar{A}^{s_2}\dots\bar{A}^{s_k})\right]|r\rangle$$
$$= \sum_{\{s_n\},\{t_n\}=1}^{d} \langle s_1 s_2 \dots s_k|\hat{O}|t_1 t_2 \dots t_k\rangle \ \mathrm{tr}\left(A^{s_k\dagger}\dots A^{s_2\dagger}A^{s_1\dagger}lA^{t_1}A^{t_2}\dots A^{t_k}r\right)$$

2.5 Tangent Space

The tangent space is an important concept for the Time-Dependent Variational Principle. In this section we use the short notation $\partial_i = \frac{\partial}{\partial A^i}$, where i is a grouped index $i = (s_n, a_{n-1}, a_n)$ and $A^{s_n}_{a_{n-1},a_n} = A^i$.

Definition 2.10: Variational manifold of a (uniform) Matrix Product State

Let $\mathcal{A}$ be the set of tensors $A \in \mathbb{C}^{d\times\chi\times\chi}$ for which the transfer matrix T has a non-degenerate dominant eigenvalue, if $|\psi\rangle$ is a uniform Matrix Product State. If $|\psi\rangle$ is a finite Matrix Product State, let $\mathcal{A}$ be the set of all tuples $A_\psi = (A_{[1]}, A_{[2]}, \dots)$ with a maximum bond dimension of χ. The variational manifold $\mathcal{M}_A$ is the set of all states $|\psi\rangle$ that can be represented by a (uniform) Matrix Product State with bond dimension χ [32].

$$\mathcal{M}_A = \{|\psi(A)\rangle \,|A \in \mathcal{A}\}$$

If we assume that the bond dimension of a (uniform) Matrix Product State is fixed, the matrix entries are functions of time. In the example below we are going to see that the derivative with respect to time of such a state has a special form that can be formalized with so called tangent states and spaces.

Example 2.5: Derivative with respect to time of a (uniform) Matrix Product State

Let the Hilbert space of a system be fully parametrized with three parameters. The derivative with respect to time of a state in this space will have the following form:

$$|\psi\rangle = f(a,b,c)$$
$$\frac{d}{dt}|\psi\rangle = \frac{\partial f(a,b,c)}{\partial a}\frac{da}{dt} + \frac{\partial f(a,b,c)}{\partial b}\frac{db}{dt} + \frac{\partial f(a,b,c)}{\partial c}\frac{dc}{dt}$$

Where $a, b, c \in \mathbb{C}$. If we consider now a Matrix Product State with a finite fixed bond dimension we cannot map onto the whole Hilbert space anymore. This is indicated with the subscript A. The fact that we cannot map onto the whole Hilbert Space anymore is indicated with the last argument of the function c which is set

permanently to zero.

$$
\begin{aligned}
|\psi\rangle_A &= f(a,b,0) \\
\frac{d}{dt}\left(|\psi\rangle_A\right) &= \frac{\partial f(a,b,0)}{\partial a}\frac{da}{dt} + \frac{\partial f(a,b,0)}{\partial b}\frac{db}{dt} \\
\Rightarrow \frac{d}{dt}\left(|\psi\rangle_A\right) &\neq \frac{d}{dt}|\psi\rangle
\end{aligned}
$$

As a consequence the derivatives of a Matrix Product State with fixed bond dimension and a general state are not equal anymore. It can be seen now that the derivative with respect to time of a Matrix Product State will have the form:

$$\tilde{a}\ \frac{\partial f(a,b,0)}{\partial a} + \tilde{b}\ \frac{\partial f(a,b,0)}{\partial b}$$

where $\tilde{a}$ and $\tilde{b}$ are arbitrary parameters.

In the example above it can be seen that the derivative with respect to time of a Matrix Product State is always in the same form. This idea can be formalized with tangent states and spaces.

Definition 2.11: Tangent states and tangent space

Let $|\psi\rangle$ be a Matrix Product State and $\mathcal{A}_\psi$ the tuple of all the tensors used to represent the state, $\mathcal{A}_\psi = (A_{[1]}, A_{[2]}, \ldots)$. Let the set of all lattice sites be denoted as L. Let $\mathcal{B}$ be a tuple of arbitrary complex tensors, where every n-th element B^{s_n} of $\mathcal{B}$ has the same dimensions as the n-th element of $\mathcal{A}_\psi$. A tangent state $|\phi\rangle$ of $|\psi\rangle$ is defined as:

$$|\phi(\mathcal{B},\mathcal{A}_\psi)\rangle = \sum_{n\in L}\sum_{\{\boldsymbol{s}\}}\left(\prod_{m<n} A^{s_m}\right) B^{s_n}\left(\prod_{m>n} A^{s_m}\right)|\boldsymbol{s}\rangle$$

For uniform Matrix Product States there is the same definition just that there are only two tensors A and B.

$$|\phi(B,A))\rangle = \sum_{n\in\mathbb{Z}}\sum_{\{\boldsymbol{s}\}} v_l^\dagger\left(\prod_{m<n} A^{s_m}\right) B^{s_n}\left(\prod_{m>n} A^{s_m}\right) v_r\,|\boldsymbol{s}\rangle$$

The tangent plane or tangent space $\mathcal{T}_A\mathcal{M}$ of a Matrix Product State is defined as the set of all possible tangent states.

$$\mathcal{T}_A\mathcal{M} = \{|\phi(\mathcal{B},\mathcal{A}_\psi)\rangle\,|\forall i : \dim(B_{[i]}) = \dim(A_{[i]})\}$$

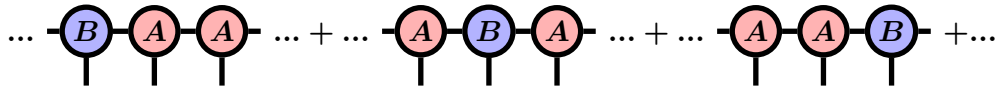

Figure 2.9: Graphical representation of a tangent state.

And similar for uniform Matrix Product States:

$$\mathcal{T}_A\mathcal{M} = \{|\phi(B, A)\rangle \,|B \in \mathbb{C}^{d\times\chi\times\chi}\}$$

An equivalent definition used in [27] is given by:

$$\mathcal{T}_A\mathcal{M} = \text{span}\{|\partial_i\psi\rangle \,|i \in \{1, 2, \ldots\}\}$$

If we assume that the tensors A of a (uniform) Matrix Product State is time dependent and reshape every tensor so that $A^i = A^{s_n}_{a_{n-1},a_n}$ with $i = (s_n, a_{n-1}, a_n)$ the derivative with respect to time of the state is a tangent state with $B^{s_n} = \dot{A}^{s_n}$.

$$\frac{d}{dt}|\psi(A)\rangle = \sum_i \dot{A}^i |\partial_i\psi(A)\rangle = |\phi(\dot{A}, A)\rangle$$
$$\Rightarrow \frac{d}{dt}|\psi(A)\rangle \in \mathcal{T}_A\mathcal{M}$$

Definition 2.12: Gram matrix and inverse Gram matrix

The Gram matrix $G_{i,j}$ of a (uniform) Matrix Product State is the overlap of its tangent basis states.

$$G_{i,j} = \langle\partial_i\psi(A)|\partial_j\psi(A)\rangle$$

Where $\partial_i = \frac{\partial}{\partial A^i}$.
The inverse Gram matrix $G^{i,j}$ (note the superscripts instead of the subscripts) is defined by the following property:

$$G^{i,j}G_{j,k} = \delta^i_k$$

Finally, we are going to need a projection operator that projects an arbitrary state to the tangent space. The naive choice $\hat{P} = \sum_i |\partial_i\psi\rangle\langle\partial_i\psi|$ does not work here because the tangent basis states $|\partial_i\psi\rangle$ do not form a orthonormal basis.

Definition 2.13: Projection operator onto the tangent space

Let $|\psi(A)\rangle$ be a uniform Matrix Product State. Its projection operator onto its tangent space $\hat{P}_{\mathcal{T}_A\mathcal{M}}$ is defined by:

$$\hat{P}_{\mathcal{T}_A\mathcal{M}} = \sum_{i,j} |\partial_i \psi(A)\rangle \, G^{i,j} \, \langle \partial_j \psi(A)|$$

3 Density-Matrix Renormalization Group

This section follows [8] and the doctoral thesis of Martin Ganahl [33]. The Density-Matrix Renormalization Group (DMRG) is a method to simulate strongly correlated quantum systems [8]. It was introduced by Steven White in 1992 [34]. The main focus of DMRG is to find the ground state of a system, where unimportant states are omitted. For infinite systems the algorithm is mostly referred to as iDMRG. In general the DMRG formalism is independent from the MPS formalism. It is based on calculating the density matrix of the state of interest and eliminate states that have a negligible weight in the eigenbasis of the density matrix. In the present thesis we will omit this approach and refer to [8]. Instead, we will directly take a look at the DMRG algorithm written in the MPS formalism.

3.1 Finite DMRG

Let us consider that we want to calculate the ground state of a Hamiltonian $\hat{H}$ and that we know that MPO representation of $\hat{H}$. The general idea behind DMRG is to start with a random Matrix Product State and do a variational ground state search. To do that we fix the tensors for every lattice except the first two neighboring tensors from the left. Then we will find the minimum value of $\frac{\langle\psi|\hat{H}|\psi\rangle}{\langle\psi|\psi\rangle}$ with respect to the two tensors that are not fixed. This process is repeated for every pair of lattice sites until we reach right the boundary of the system and then repeated back to the start again. This is called one sweep. After several sweeps the calculated state should be a good approximation of the ground state. To minimize the expectation value with respect to the two free tensors $A_{[n]}$ and $A_{[n+1]}$ we combine these two tensors $A^{s_n,s_{n+1}} = A^{s_n}A^{s_{n+1}}$ and use a Lagrange multiplier.

$$\mathcal{L}(A_{[n,n+1]}, \bar{A}_{[n,n+1]}, \lambda) = \langle\psi|\,\hat{H}\,|\psi\rangle + \lambda(\langle\psi|\psi\rangle - 1)$$

Now we need to solve $\nabla_{A_{[n,n+1]},\bar{A}_{[n,n+1]},\lambda}\mathcal{L} = 0$. In this chapter we will use the short notation $\partial_x = \frac{\partial}{\partial x}$. The two equations $\partial_{A_{[n,n+1]}}\mathcal{L} = 0$ and $\partial_{\bar{A}_{[n,n+1]}}\mathcal{L} = 0$ are form equivalent and solving them will yield the same result for $A_{[n,n+1]}$. Thus we will omit the former equation. Note that we used another short notation here. The derivative with respect to the tensor is to be read as

$$\partial_{A_{[n,n+1]}}\mathcal{L} = 0 \quad \Rightarrow \quad \partial_{A^{s_n,s_{n+1}}_{a_{n-1},a_{n+1}}}\mathcal{L} = 0 \quad \forall a_{n-1}, a_{n+1}, s_n, s_{n+1}$$

The equation $\partial_\lambda\mathcal{L} = \langle\psi|\psi\rangle - 1 = 0$ imposes normalization and this is assured with the normalization condition for mixed-canonical Matrix Product States (see thm. 2.4). This is possible, because we will always hold the Matrix Product State in mixed-canonical form during the ground state search.
When calculating the value of $\langle\psi|\,\hat{H}\,|\psi\rangle$ we must contract the tensor network to the left and right of the free tensors $A_{[n]}$ and $A_{[n+1]}$. This can be seen in fig. 3.1 and it is the best way to do this with an recursive calculation.

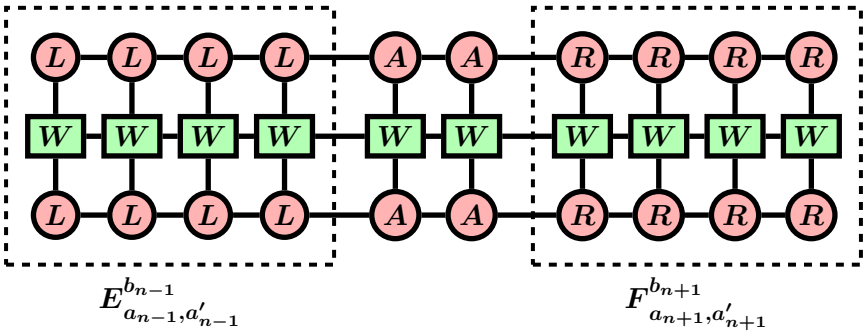

Figure 3.1: Graphical representation of $\langle\psi|\,\hat{H}\,|\psi\rangle$. The two free tensors are the two denoted with A. The left- and right-normalized tensors denoted with L and R are fixed.

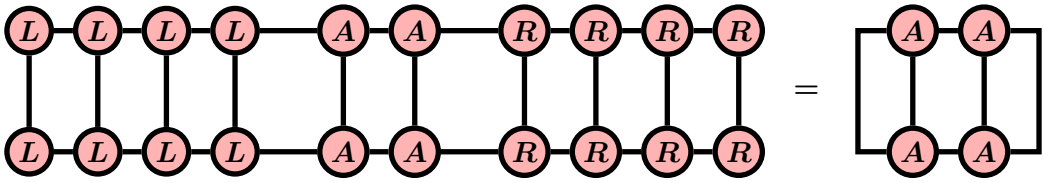

Figure 3.2: Graphical representation of $\langle\psi|\psi\rangle$. The two free tensors are the two denoted with A. The left- and right-normalized tensors denoted with L and R are fixed.

$$E^{b_1}_{a_1,a'_1} = \sum_{s_1,s'_1} L^{s_1}_{1,a_1} \bar{L}^{s'_1}_{1,a'_1} W^{s'_1,s_1}_{1,b_1}$$

$$E^{b_n}_{a_n,a'_n} = \sum_{s_n,s'_n} \sum_{a_{n-1}} \sum_{a'_{n-1}} \sum_{b_{n-1}} E^{b_{n-1}}_{a_{n-1},a'_{n-1}} L^{s_n}_{a_{n-1},a_n} \bar{L}^{s'_n}_{a'_{n-1},a'_n} W^{s'_n,s_n}_{b_{n-1},b_n}$$

$$F^{b_{N-1}}_{a_{N-1},a_{N-1}} = \sum_{s_N,s'_N} R^{s_N}_{a_{N-1},1} \bar{R}^{s'_N}_{a'_{N-1},1} W^{s'_N,s_N}_{b_{N-1},1}$$

$$F^{b_n}_{a_n,a'_n} = \sum_{s_n,s'_n} \sum_{a_{n+1}} \sum_{a'_{n+1}} \sum_{b_{n+1}} F^{b_{n+1}}_{a_{n+1},a'_{n+1}} R^{s_n}_{a_n,a_{n+1}} \bar{R}^{s'_n}_{a'_n,a'_{n+1}} W^{s'_n,s_n}_{b_n,b_{n+1}}$$

The calculation of $\langle\psi|\psi\rangle$ is very simple, because we will hold the Matrix Product State in mixed-canonical form and thus the tensor network collapses to the two free tensors $A_{[n]}$ and $A_{[n+1]}$ (see fig. 3.2). Finally, we define a combined index $\alpha = (s_{n+1}, s_n, a_{n+1}, a_{n-1})$, an effective Hamiltonian $h_{\alpha',\alpha}$, and a vector X_α that is just the two site tensor $A_{[n,n+1]}$ reshaped into vector form.

$$h_{\alpha',\alpha} = \sum_{b_{n-1},b_n,b_{n+1}} E^{b_{n-1}}_{a_{n-1},a'_{n-1}} W^{s'_n,s_n}_{b_{n-1},b_n} W^{s'_{n+1},s_{n+1}}_{b_n,b_{n+1}} F^{b_{n+1}}_{a_{n+1},a'_{n+1}}$$

$$X_\alpha = \sum_{a_n} A^{s_n}_{a_{n-1},a_n} A^{s_{n+1}}_{a_n,a_{n+1}}$$

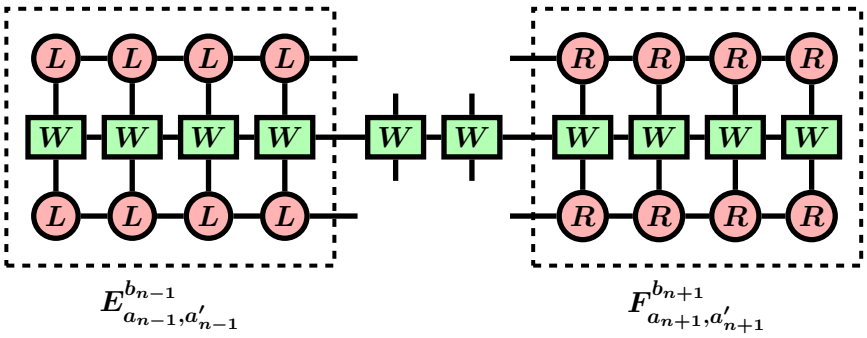

Figure 3.3: Graphical representation of $h_{\alpha,\alpha'}$.

With that transformations in mind the Lagrange equations transform into

$$\begin{aligned}\partial_{\bar{X}_{\alpha'}} \mathcal{L} &= \partial_{\bar{X}_{\alpha'}} \left(\sum_{\alpha,\alpha''} \bar{X}_{\alpha''} h_{\alpha'',\alpha} X_\alpha + \lambda \sum_{\alpha''} \bar{X}_{\alpha''} X_{\alpha''} \right) \\ &= \sum_\alpha h_{\alpha',\alpha} X_\alpha + \lambda X_{\alpha'} = 0\end{aligned}$$

The last line is an eigenequation $hX = -\lambda X$, where $-\lambda$ is the ground state energy with respect to the free tensor $A_{[n,n+1]}$. This eigenproblem can be solved numerically with e.g. the Lanczos method. For the last step we reshape the calculated X back into $A^{s_n,s_{n+1}}_{a_{n-1},a_{n+1}}$ and split that tensor into the product $L^{s_n} S R^{s_{n+1}}$. Now we have a valid mixed-canonical Matrix Product State, can reduce the bond dimension if it is necessary, and fix the norm of the state according to the normalization condition for mixed-canonical Matrix Product States. Depending on the sweep direction we multiply the center matrix S to the left or right tensor for the next minimization step.

$$L^{s_n} S R^{s_{n+1}} = A^{s_n} R^{s_{n+1}} = L^{s_n} A^{s_{n+1}}$$

Algorithm 3.1: Two-site DMRG in Matrix Product State notation

1. *Generate a random Matrix Product State.*
2. *Gauge transform into a right-canonical Matrix Product State.*
3. *Compute* $F^{b_n}_{a_n,a'_n}$ *for every* $n \in \{N-1, N-2, \ldots, 3, 2\}$.
4. *For every* $n \in \{1, 2, \ldots, N-1\}$*:*
   (a) *Solve eigenproblem* $\sum_\alpha h_{\alpha',\alpha} X_\alpha = -\lambda X_{\alpha'}$.
   (b) *Split* X *into a left-normalized* L^{s_n}*, and* $A^{s_{n+1}}$*. This is done with a singular value decomposition. Reduce the bond dimension and fix the norm of the state if necessary.*
   (c) *Calculate the left contraction* $E^{b_n}_{a_n,a'_n}$.
5. *For every* $n \in \{N-1, N-2, \ldots, 2, 1\}$*:*
   (a) *Solve eigenproblem* $\sum_\alpha h_{\alpha',\alpha} X_\alpha = -\lambda X_{\alpha'}$.
   (b) *Split* X *into a* A^{s_n}*, and right-normalized* $R^{s_{n+1}}$*. This is done with a singular value decomposition. Reduce the bond dimension and fix the norm of the state if necessary.*
   (c) *Calculate the right contraction* $F^{b_n}_{a_n,a'_n}$.
6. *If the energy* $(-\lambda)$ *is not converged, go back to step 4.*

3.2 Infinite DMRG

The general idea behind the infinite version of DMRG (iDMRG) is very similar to its finite counter part. We want to find the ground state for a Hamiltonian $\hat{H}$ that describes a system with an arbitrary number of lattice sites. We assume that we already know the MPO representation of $\hat{H}$. We start to find the matrix product ground state $|\psi\rangle$ for two lattice sites in its mixed-canonical representation.

$$|\psi\rangle = \sum_{\boldsymbol{s}} L^{s_1^{\mathcal{A}}} S_{[1]} R^{s_1^{\mathcal{B}}} |\boldsymbol{s}\rangle$$

For the next step we insert two new lattice sites between the two systems $\mathcal{A}$ and $\mathcal{B}$, fix all tensors in these systems, and minimize the value of $\frac{\langle\psi|\hat{H}|\psi\rangle}{\langle\psi|\psi\rangle}$ with respect to the tensor A.

$$|\psi\rangle = \sum_{\boldsymbol{s}} L^{s_1^{\mathcal{A}}} A^{s_2^{\mathcal{A}}, s_2^{\mathcal{B}}} R^{s_1^{\mathcal{B}}} |\boldsymbol{s}\rangle$$

It is assumed that A takes the place of the center matrix S. This minimization is done

with the same method as described in the section for finite DMRG. For four lattice sites we arrive at the new state

$$|\psi\rangle = \sum_{\boldsymbol{s}} L^{s_1^{\mathcal{A}}} L^{s_2^{\mathcal{A}}} S_{[2]} R^{s_2^{\mathcal{B}}} R^{s_1^{\mathcal{B}}} |\boldsymbol{s}\rangle$$

This process gets repeated for 6, 8, 10,... lattice sites. So, in general one iteration step starts with a state of this form

$$|\psi\rangle = \sum_{\boldsymbol{s}} \dots L^{s_{n-2}^{\mathcal{A}}} L^{s_{n-1}^{\mathcal{A}}} A^{s_n^{\mathcal{A}}, s_n^{\mathcal{B}}} R^{s_{n-1}^{\mathcal{B}}} R^{s_{n-2}^{\mathcal{B}}} \dots |\boldsymbol{s}\rangle$$

and $\frac{\langle\psi|\hat{H}|\psi\rangle}{\langle\psi|\psi\rangle}$ is minimized with respect to A to derive a new mixed-canonical state of the form

$$|\psi\rangle = \sum_{\boldsymbol{s}} \dots L^{s_{n-2}^{\mathcal{A}}} L^{s_{n-1}^{\mathcal{A}}} L^{s_n^{\mathcal{A}}} S_{[n]} R^{s_n^{\mathcal{B}}} R^{s_{n-1}^{\mathcal{B}}} R^{s_{n-2}^{\mathcal{B}}} \dots |\boldsymbol{s}\rangle$$

Note that we need to reduce the bond dimension of the matrices if it gets to large. This process is repeated until convergence, or the desired system size is reached. One option is to stop at a certain lattice size and use the calculated state as initial state for finite DMRG instead of a random state. The other option is to assume the state can be described as an infinite length Matrix Product State with a two site unit cell (similar to a uniform Matrix Product State).

4 Time-Evolving Block Decimation

The Time-Evolving Block Decimation (TEBD) was first introduced by Guifré Vidal in 2003 [28]. In general it is used to compute the time evolution of a one dimensional chain. When doing a time evolution the principal problem is to calculate the time evolution operator $e^{-it\hat{H}}$. In this chapter we assume the Hamiltonian to be of the form:

$$\hat{H} = \sum_{j=0}^{N-1} \hat{h}_{j,j+1}$$

where $\hat{h}_{j,j+1}$ acts only on site j and $j+1$ and N is the number of lattice sites.
If we assume that all $\hat{h}_{j,j+1}$ commute with each other the computation of the time evolution operator is easy.

$$\hat{H} = \sum_{j} \hat{h}_{j,j+1} \ \wedge \ [\hat{h}_{j,j+1}, \hat{h}_{j+1,j+2}] = 0 \ \forall j \quad \Rightarrow \quad e^{-it\hat{H}} = \prod_{j} e^{-it\hat{h}_{j,j+1}}$$

However, in most cases the commutator at hand will not vanish. The basic idea behind TEBD is to split the system up into two parts, even and odd, where every two-site-Hamiltonian will commute with each other within said part.

$$\hat{H} = \hat{H}_{\text{even}} + \hat{H}_{\text{odd}} = \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor} \hat{h}_{2j,2j+1} + \sum_{j=0}^{\lfloor \frac{N}{2} \rfloor} \hat{h}_{2j+1,2j+2}$$

As mentioned above $\forall m, n : \ [\hat{h}_{2j,2j+1}, \hat{h}_{2m,2m+1}] = [\hat{h}_{2j+1,2j+2}, \hat{h}_{2m+1,2m+2}] = 0$, but $[\hat{H}_{\text{even}}, \hat{H}_{\text{odd}}] \neq 0$. Because every operator within even and odd commutes it is trivial to calculate the time evolution within these systems.

$$e^{-it\hat{H}_{\text{even}}} = \prod_{\hat{h} \in \hat{H}_{\text{even}}} e^{-it\hat{h}}$$
$$e^{-it\hat{H}_{\text{odd}}} = \prod_{\hat{h} \in \hat{H}_{\text{odd}}} e^{-it\hat{h}}$$

To calculate the time evolution of the whole system we define the time step $\Delta t = \frac{t}{n}$ with $n \in \mathbb{N}$ and use the Baker-Hausdorff formula to calculate:

$$
\begin{aligned}
e^{-it\hat{H}} &= e^{-it(\hat{H}_{\text{even}}+\hat{H}_{\text{odd}})} \\
&= \left(e^{-i\Delta t(\hat{H}_{\text{even}}+\hat{H}_{\text{odd}})}\right)^n \\
&= \left(e^{-i\Delta t\hat{H}_{\text{even}}}e^{-i\Delta t\hat{H}_{\text{odd}}}(1+\mathcal{O}(\Delta t^2))\right)^n \\
&= \left(e^{-i\Delta t\hat{H}_{\text{even}}}e^{-i\Delta t\hat{H}_{\text{odd}}}e^{-i\Delta t\hat{H}_{\text{even}}}e^{-i\Delta t\hat{H}_{\text{odd}}}...e^{-i\Delta t\hat{H}_{\text{even}}}e^{-i\Delta t\hat{H}_{\text{odd}}}\right)(1+\mathcal{O}(\Delta t))
\end{aligned}
$$

Algorithm 4.1: First order time evolving block decimation

Let $|\psi\rangle$ be an arbitrary Matrix Product State and let $\hat{U}_{even/odd} = e^{-i\Delta t\hat{H}_{even/odd}}$ (see above). $|\psi(\Delta t)\rangle$ is approximated with:

1. $|\tilde{\psi}\rangle = \hat{U}_{even}|\psi\rangle$
2. $|\psi(\Delta t)\rangle = \hat{U}_{odd}|\tilde{\psi}\rangle$

5 Time-Dependent Variational Principle

This section will follow [27] and [35]. The Time-Dependent Variational Principle for uniform Matrix Product States was introduced by Haegeman et. al. [27] in 2011. It is a versatile tool to do time evolutions and ground state calculations of Matrix Product States on infinite one dimensional lattices. Later on, the algorithm was extended to work with Matrix Product Operators on finite lattices [35]. In the original article [27] only Hamiltonians of the form $\hat{H} = \sum_i \hat{h}_{i,i+1}$ were considered (i.e. only nearest neighbor interactions). The present thesis modifies this approach to include exponentially decaying long ranged interactions. This extension to TDVP was done in email collaboration with Valentin Zauner (now Valentin Stauber; University of Vienna). For an in-depth discussion about TDVP and the concept of tangent states and spaces see [32].

5.1 Basic Idea

The basic idea behind TDVP is not to solve the Schrödinger equation $|\dot{\psi}\rangle = -i\hat{H}\,|\psi\rangle$ in the whole state space but to find the best approximation within the variational manifold spanned by matrices with a certain maximum size χ. This is in contrast to algorithms like the two-site DMRG or TEBD where for every two new lattice sites added the new matrix dimension is at first a multiple of the local Hilbert space dimension and is reduced only later [8]. For TDVP, to find the best approximation within the desired space, the right hand side of the Schrödinger equation is projected onto the tangent space of the current MPS manifold [35]. This means, we need to solve the following equation:

$$|\dot{\psi}\rangle_A = -i\hat{P}_{\mathcal{T}_A\mathcal{M}}\hat{H}\,|\psi\rangle_A$$

Here the subscript A explicitly denotes that $|\psi\rangle_A$ is a (uniform) Matrix Product State, and $\hat{P}_{\mathcal{T}_A\mathcal{M}}$ is the projection operator onto the tangent space of $|\psi\rangle$ (see def. 2.13). An illustration of the process can be seen in fig. 5.1. If this differential equation is solved and $|\psi\rangle_A$ updated accordingly, we are never going to leave the initial manifold of $|\psi\rangle$.

5.2 Finite Lattice TDVP

For the Finite lattice TDVP we need to find a representation of the projection operator onto the tangent space $\hat{P}_{\mathcal{T}_A\mathcal{M}}$ in the MPS basis [35]. The projection will be derived for an arbitrary state $|\xi\rangle$ (not necessarily an MPS) by minimizing the norm squared of the difference between $|\xi\rangle$ and an arbitrary tangent state (eq. (5.1)).

First we rewrite the definition of tangent states (def. 2.11).

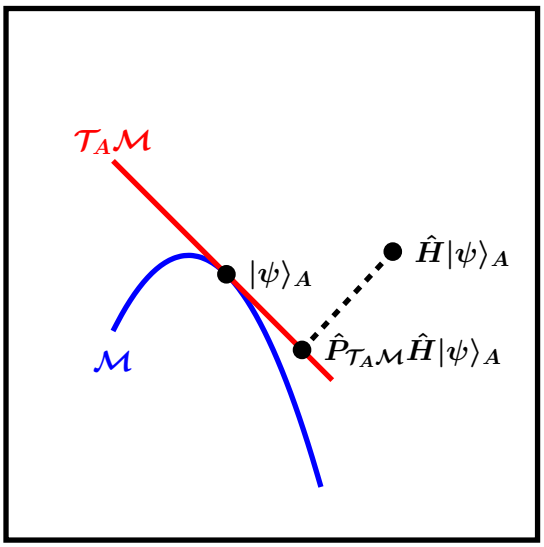

Figure 5.1: Graphical representation of the projection of $\hat{P}_{\mathcal{T}_A\mathcal{M}}\hat{H}\,|\psi\rangle_A$ onto the tangent space.

$$
\begin{aligned}
|\phi(\mathcal{B},\mathcal{A}_\psi)\rangle &= \sum_{n\in L}\sum_{\{\boldsymbol{s}\}}\left(\prod_{m<n}A^{s_m}\right)B^{s_n}\left(\prod_{m>n}A^{s_m}\right)|\boldsymbol{s}\rangle \\
&= \sum_{n\in L}\sum_{\{\boldsymbol{s}\}}\left(\prod_{m<n}L^{s_m}\right)G_{l,[n]}B^{s_n}G_{r,[n+1]}\left(\prod_{m>n}R^{s_m}\right)|\boldsymbol{s}\rangle \\
&= \sum_{n\in L}\sum_{\{\boldsymbol{s}\}}\left(\prod_{m<n}L^{s_m}\right)\tilde{B}^{s_n}\left(\prod_{m>n}R^{s_m}\right)|\boldsymbol{s}\rangle \\
&= |\tilde{\phi}(\tilde{\mathcal{B}},\mathcal{A}_\psi)\rangle
\end{aligned}
$$

In the second line we transformed every tensor left of B^{s_n} into its left-normalized representation and the every tensor of B^{s_n} into its right-normalized representation. The two matrices $G_{l,[n]}$ and $G_{r,[n+1]}$ are the factors that remain after the normalization and are multiplied into B^{s_n}, resulting in $\tilde{B}^{s_n} = G_{l,[n]}B^{s_n}G_{r,[n+1]}$. The problem is now that there is no one-to-one correspondence between the derivation of a Matrix Product State and the modified tangent state $|\tilde{\phi}(\tilde{\mathcal{B}},\mathcal{A}_\psi)\rangle$, but the derivative of the projection operator is easier this way. For the rest of this chapter we are going to omit the two tildes and the second argument of $|\tilde{\phi}(\tilde{\mathcal{B}},\mathcal{A}_\psi)\rangle \to |\phi(\mathcal{B})\rangle$ for the sake of brevity. Furthermore, we introduce $|a_n\rangle$ and $|b_n\rangle$ that are orthonormal basis states of the sublattices to the left and to the right of site n.

$$
\begin{aligned}
|a_n\rangle &= \sum_{s_1,s_2,\ldots,s_n} \Big(\prod_{m\leq n} L^{s_m} \Big)_{1,a} \ |s_1 s_2 \ldots s_n\rangle \\
|a_{n+1}\rangle &= \sum_{a',s_{n+1}} L^{s_{n+1}}_{a',a} \, |a'_n\rangle \, |s_{n+1}\rangle \\
|b_n\rangle &= \sum_{s_n,s_{n+1}\ldots,s_N} \Big(\prod_{m\geq n} R^{s_m} \Big)_{b,1} \ |s_n s_{n+1} \ldots s_N\rangle
\end{aligned}
$$

Keep the recursion relation for $|a_n\rangle$ in mind. This is going to be important later. With these definitions the tangent state $|\phi(\mathcal{B})\rangle$ can be written as:

$$
|\phi(\mathcal{B})\rangle = \sum_n \sum_{a,b,s_n} B^{s_n}_{a,b} \, |a_{n-1}\rangle \, |s_n\rangle \, |b_{n+1}\rangle
$$

Note that a similar gauge invariance exists for the tangent states as for regular Matrix Product States. To tangent states defined by the matrices $B^{s'_n}$ and B^{s_n} are the same, when the two tensors are related by:

$$
B^{s'_n} = B^{s_n} + X_{[n-1]} R^{s_n} - L^{s_n} X_{[n]} \quad \Rightarrow \quad |\phi(\mathcal{B}')\rangle = |\phi(\mathcal{B})\rangle
$$

Here the $X_{[n]}$ are arbitrary complex matrices that match the dimensions of the original Matrix Product State. Thus we can fix a gauge for the different B^{s_n}. There are several equivalent choices for the gauge. Here we are going to impose the condition $\forall n \neq N : \sum_{s_n} L^{s_n \dagger} B^{s_n} = 0$. This comes in very handy when calculating the overlap of two tangent states.

$$
\langle \phi(\mathcal{B}_1) | \phi(\mathcal{B}_2) \rangle = \sum_n \sum_{s_n} \mathrm{tr}(B_1^{s_n \dagger} B_2^{s_n})
$$

This is the form of an Euclidean inner product and thus corresponds to an orthonormal basis. The orthonormal basis guarantees that a solution to $|\phi(\mathcal{B})\rangle = \hat{P}_{\mathcal{T}_\mathcal{A}\mathcal{M}} \, |\xi\rangle$ (with an arbitrary state $|\xi\rangle$) is also a solution to:

$$
\min_{\mathcal{B}} \| |\phi(\mathcal{B})\rangle - |\xi\rangle \|^2 = \min_{\mathcal{B}} \langle \phi(\mathcal{B}) | \phi(\mathcal{B}) \rangle - \langle \phi(\mathcal{B}) | \xi \rangle - \langle \xi | \phi(\mathcal{B}) \rangle + \langle \xi | \xi \rangle \tag{5.1}
$$

We can omit the term $\langle \xi | \xi \rangle$ in the equation above, because it is just a constant with respect to $\mathcal{B}$. Next, we are going to look onto the second term of the minimization.

$$
\begin{aligned}
\langle \phi(\mathcal{B}) | \xi \rangle &= \sum_n \sum_{a,b,s_n} B^{s_n \dagger}_{b,a} \, \langle a_{n-1} s_n b_{n+1} | \xi \rangle \\
&= \sum_n \mathrm{tr}(B^{s_n \dagger} F^{s_n})
\end{aligned}
$$

Where $F^{s_n}_{a,b} = \langle a_{n-1} s_n b_{n+1} | \xi \rangle$.

Then, in matrix notation this boils down to:

$$\begin{aligned}
&\min_{\mathcal{B}} \sum_n \sum_{s_n} \mathrm{tr}\left(B^{s_n\dagger}B^{s_n} - F^{s_n\dagger}B^{s_n} - B^{s_n\dagger}F^{s_n}\right) \\
&\sum_{s_n} L^{s_n\dagger}B^{s_n} = 0 \quad \forall n \neq N
\end{aligned}$$

We can solve this with Lagrange multipliers:

$$\mathcal{L} = \sum_{n=1}^{N}\sum_{s_n}\sum_{a,b} \bar{B}^{s_n}_{b,a}B^{s_n}_{b,a} - \bar{F}^{s_n}_{b,a}B^{s_n}_{b,a} - \bar{B}^{s_n}_{b,a}F^{s_n}_{b,a} + \sum_{n=1}^{N-1}\sum_{a,b,c}\sum_{s_n} \lambda^n_{a,c}\bar{B}^{s_n}_{b,a}L^{s_n}_{b,c}$$

Where we used the hermitian conjugate of the constraint.

$$\frac{\partial\mathcal{L}}{\partial\bar{B}^{s_n}_{b,a}} = B^{s_n}_{b,a} - F^{s_n}_{b,a} + \sum_c \lambda^n_{a,c}L^{s_n}_{b,c} \overset{!}{=} 0 \quad \Rightarrow \quad B^{s_n}_{b,a} = F^{s_n}_{b,a} - \sum_c \lambda^n_{a,c}L^{s_n}_{b,c}$$

Now we need to plug this solution into the side condition:

$$\begin{aligned}
0 \overset{!}{=} \sum_{s_n,b} \bar{B}^{s_n}_{b,a}L^{s_n}_{b,c} &= \sum_{s_n,b}\left(\bar{F}^{s_n}_{b,a}L^{s_n}_{b,c} - \sum_{c'}\bar{\lambda}^n_{a,c'}\underbrace{\bar{L}^{s_n}_{b,c'}L^{s_n}_{b,c}}_{=\delta_{c,c'}}\right) \\
&= \sum_{s_n,b}\left(\bar{F}^{s_n}_{b,a}L^{s_n}_{b,c}\right) - \bar{\lambda}^n_{a,c} \\
\Rightarrow \quad \lambda^n_{a,c} &= \sum_{s_n,b}\left(F^{s_n}_{b,a}\bar{L}^{s_n}_{b,c}\right)
\end{aligned}$$

So we get our final result for the B in matrix notation:

$$B^{s_n} = \begin{cases} F^{s_n} - L^{s_n}\sum_{t_n} L^{t_n\dagger}F^{t_n} & \text{if } n < N \\ F^{s_n} & \text{if } n = N \end{cases}$$

For the next step we define $G = \sum_{t_n} L^{t_n\dagger}F^{t_n}$ and calculate it in braket notation.

$$G_{a,b} = \sum_{t_n} L^{t_n\dagger}_{a,a'} \langle a'_{n-1}| \langle t_n b_{n+1}|\xi\rangle = \langle a_n b_{n+1}|\xi\rangle$$

And finally we plug the minimum solution of $\mathcal{B}$ into our tangent state:

$$
\begin{aligned}
|\phi(\mathcal{B})\rangle &= \sum_n \sum_{a,b,s_n} B^{s_n}_{a,b} \, |a_{n-1} s_n b_{n+1}\rangle \\
&= \sum_{n=1}^{N} \sum_{s_n,a,b} F^{s_n}_{a,b} \, |a_{n-1} s_n b_{n+1}\rangle - \sum_{n=1}^{N-1} \sum_{s_n,a,a',b} L^{s_n}_{a',a} G_{a,b} \, |a'_{n-1} s_n b_{n+1}\rangle \\
&= \sum_{n=1}^{N} \sum_{s_n,a,b} |a_{n-1} s_n b_{n+1}\rangle \langle a_{n-1} s_n b_{n+1}| \;\; |\xi\rangle - \sum_{n=1}^{N-1} |a_n b_{n+1}\rangle \langle a_n b_{n+1}| \;\; |\xi\rangle \\
&= \left(\sum_{n=1}^{N} \hat{P}^L_{n-1} \otimes \mathbb{1}_n \otimes \hat{P}^R_{n+1} - \sum_{n=1}^{N-1} \hat{P}^L_n \otimes \hat{P}^R_{n+1} \right) |\xi\rangle \\
&\overset{!}{=} \hat{P}_{\mathcal{T}_\mathcal{A}\mathcal{M}} \, |\xi\rangle
\end{aligned}
$$

This is the representation of the projection operator in Matrix Product State basis.

Result 5.1: Tangent space projector $\hat{P}_{\mathcal{T}_\mathcal{A}\mathcal{M}}$ in MPS notation

$$
\begin{aligned}
|a_n\rangle &= \sum_{s_1,s_2,\ldots,s_n} \Big(\prod_{m \le n} L^{s_m} \Big)_{1,a} \;\; |s_1 s_2 \ldots s_n\rangle \\
|b_n\rangle &= \sum_{s_n,s_{n+1}\ldots,s_N} \Big(\prod_{m \ge n} R^{s_m} \Big)_{b,1} \;\; |s_n s_{n+1} \ldots s_N\rangle \\
\hat{P}^L_n &= \sum_a |a_n\rangle \langle a_n| \\
\hat{P}^R_n &= \sum_b |b_n\rangle \langle b_n| \\
\hat{P}_{\mathcal{T}_\mathcal{A}\mathcal{M}} &= \sum_{n=1}^{N} \hat{P}^L_{n-1} \otimes \mathbb{1}_n \otimes \hat{P}^R_{n+1} - \sum_{n=1}^{N-1} \hat{P}^L_n \otimes \hat{P}^R_{n+1}
\end{aligned}
$$

Finite One Site TDVP Algorithm

Now we want to use the projection operator and solve the modified Schrödinger equation $|\psi\rangle_A = -i\hat{P}_{\mathcal{T}_\mathcal{A}\mathcal{M}}\hat{H} \, |\psi\rangle_A$. We will see that the emerging algorithm is almost identical to the one site DMRG algorithm with a few steps modified. The idea is to time evolve every lattice site separately. We assume that $\hat{H}$ is known in Matrix Product Operator form $\hat{H} = \sum_{\boldsymbol{s},\boldsymbol{s'}} \prod_n W^{s_n,s'_n} |\boldsymbol{s}\rangle \langle \boldsymbol{s'}|$, and $|\psi\rangle$ is in mixed-canonical form. Similar as in section 3 we define the tensors F and E defined by:

$$E^{b_1}_{a_1,a'_1} = \sum_{s_1,s'_1} L^{s_1}_{1,a_1} \bar{L}^{s'_1}_{1,a'_1} W^{s'_1,s_1}_{1,b_1}$$
$$E^{b_n}_{a_n,a'_n} = \sum_{s_n,s'_n} \sum_{a_{n-1}} \sum_{a'_{n-1}} \sum_{b_{n-1}} E^{b_{n-1}}_{a_{n-1},a'_{n-1}} L^{s_n}_{a_{n-1},a_n} \bar{L}^{s'_n}_{a'_{n-1},a'_n} W^{s'_n,s_n}_{b_{n-1},b_n}$$

$$F^{b_{N-1}}_{a_{N-1},a_{N-1}} = \sum_{s_N,s'_N} R^{s_N}_{a_{N-1},1} \bar{R}^{s'_N}_{a'_{N-1},1} W^{s'_N,s_N}_{b_{N-1},1}$$
$$F^{b_n}_{a_n,a'_n} = \sum_{s_n,s'_n} \sum_{a_{n+1}} \sum_{a'_{n+1}} \sum_{b_{n+1}} F^{b_{n+1}}_{a_{n+1},a'_{n+1}} R^{s_n}_{a_n,a_{n+1}} \bar{R}^{s'_n}_{a'_n,a'_{n+1}} W^{s'_n,s_n}_{b_n,b_{n+1}}$$

Furthermore we define the combined index $\alpha = (s_n, a_{n-1}, a_n)$ and the effective Hamiltonian:

$$h_{\alpha',\alpha} = \sum_{b_{n-1},b_n} E^{b_{n-1}}_{a_{n-1},a'_{n-1}} W^{s'_n,s_n}_{b_{n-1},b_n} F^{b_n}_{a_n,a'_n}$$

One term of the first sum of the modified Schrödinger equation now reads as:

$$\sum_{\boldsymbol{s}} \dots L^{s_{n-1}} \dot{A}^{s_n} R^{s_{n+1}} \dots |\boldsymbol{s}\rangle = -i \sum_{\boldsymbol{s},\alpha,\alpha'} \left(\dots L^{s'_{n-1}} \right)_{1,a'_{n-1}} h_{\alpha',\alpha} A_\alpha \left(R^{s'_{n+1}} \dots \right)_{a'_n,1} |\boldsymbol{s'}\rangle$$

This yields:

$$\dot{A}_{[n]} = -i h_{[n]} A_{[n]} \quad \Rightarrow \quad A_{[n]}(t) = e^{-ith_{[n]}} A_{[n]}(0)$$

where we explicitly wrote the lattice site. Note that in this notation $A_{[n]}$ is a vector: $A_\alpha = A^{s_n}_{a_{n-1},a_n}$. Similarly, on term of the second sum of the modified Schrödinger equation reads as:

$$\sum_{\boldsymbol{s}} \dots L^{s_n} S R^{s_{n+1}} \dots |\boldsymbol{s}\rangle = i \sum_{\boldsymbol{s},\beta,\beta'} \left(\dots L^{s'_{n-1}} \right)_{1,a'_{n-1}} k_{\beta',\beta} S_\beta \left(R^{s'_{n+1}} \dots \right)_{a'_n,1} |\boldsymbol{s'}\rangle$$

where we defined a new combined index $\beta = (a_n, \tilde{a}_n)$ the effective zero-site Hamiltonian:

$$k_{\beta,\beta'} = \sum_{\substack{a_{n-1},a'_{n-1} \\ s_n,s'_n}} h_{(s_n,\tilde{a}_n,a_{n-1}),(s'_n,\tilde{a}'_n,a'_{n-1})} L^{s_n}_{a_{n-1},a_n} \bar{L}^{s'_n}_{a'_{n-1},a_n}$$

This yields the differential equation:

$$\dot{S}_{[n]} = i k_{[n]} S_{[n]} \quad \Rightarrow \quad S_{[n]}(t) = e^{itk_{[n]}} S_{[n]}(0)$$

All in all one integration step with a time step of Δt is:

Algorithm 5.1: One site finite TDVP

1. *Start with arbitrary state* $|\psi\rangle$ *at* $t = 0$.
2. *Gauge transform into a right-canonical Matrix Product State.*
3. *Compute* $F^{b_n}_{a_n,a'_n}$ *for every* $n \in \{N-1, N-2, \ldots, 3, 2\}$.
4. *For every* $n \in \{1, 2, \ldots, N\}$*:*
   (a) *Compute* $A_{[n]}(\frac{\Delta t}{2}) = e^{-i\frac{\Delta t}{2}h_{[n]}}A_{[n]}(0)$.
   (b) *Calculate left-normalized tensor* L^{s_n} *and center matrix* $S_{[n]}$*:* $A^{s_n} = L^{s_n}S_{[n]}$.
   (c) *Calculate the left contraction* $E^{b_n}_{a_n,a'_n}$.
   (d) *Compute* $S_{[n]}(0) = e^{i\frac{\Delta t}{2}k_{[n]}}S_{[n]}(\frac{\Delta t}{2})$.
   (e) *Multiply* S *into the next tensor:* $A^{s_{n+1}} = S_{[n]}R^{s_{n+1}}$.
5. *For every* $n \in \{N, N-1, \ldots, 2, 1\}$*:*
   (a) *Compute* $A_{[n]}(\Delta t) = e^{-i\frac{\Delta t}{2}h_{[n]}}A_{[n]}(\frac{\Delta t}{2})$.
   (b) *Calculate right-normalized tensor* R^{s_n} *and center matrix* $S_{[n-1]}$*:* $A^{s_n} = S_{[n-1]}R^{s_n}$.
   (c) *Calculate the right contraction* $F^{b_{n-1}}_{a_{n-1},a'_{n-1}}$.
   (d) *Compute* $S_{[n-1]}(\frac{\Delta t}{2}) = e^{i\frac{\Delta t}{2}k_{[n]}}S_{[n-1]}(\Delta t)$.
   (e) *Multiply* S *into the next tensor:* $A^{s_{n-1}} = L^{s_{n-1}}S_{[n-1]}$.

Finite Two site TDVP Algorithm

It is also possible to do the same calculation as above with a two site tangent space. Note that now always two sites are combined into one tensor $A_{[n,n+1]}$.
We have now the same effective Hamiltonian as in the two site DMRG algorithm:

$$h^2_{\alpha',\alpha} = \sum_{b_{n-1},b_n,b_{n+1}} E^{b_{n-1}}_{a_{n-1},a'_{n-1}} W^{s'_n,s_n}_{b_{n-1},b_n} W^{s'_{n+1},s_{n+1}}_{b_n,b_{n+1}} F^{b_{n+1}}_{a_{n+1},a'_{n+1}}$$

Algorithm 5.2: Two site finite TDVP

1. *Start with arbitrary state* $|\psi\rangle$ *at* $t = 0$.

2. *Gauge transform into a right-canonical Matrix Product State.*
3. *Compute* $F^{b_n}_{a_n,a'_n}$ *for every* $n \in \{N-1, N-2, \ldots, 3, 2\}$.
4. *For every* $n \in \{1, 2, \ldots, N-1\}$*:*
   (a) *Compute* $A_{[n,n+1]}(\frac{\Delta t}{2}) = e^{-i\frac{\Delta t}{2}h^2_{[n,n+1]}} A_{[n,n+1]}(0)$.
   (b) *Split* $A_{[n,n+1]}$ *into:* $A^{s_n,s_{n+1}} = L^{s_n} A^{s_{n+1}}$.
   (c) *Calculate the left contraction* $E^{b_n}_{a_n,a'_n}$.
   (d) *Compute* $A_{[n+1]}(0) = e^{i\frac{\Delta t}{2}h_{[n]}} A_{[n+1]}(\frac{\Delta t}{2})$.
5. *For every* $n \in \{N, N-1, \ldots, 2\}$*:*
   (a) *Compute* $A_{[n-1,n]}(\Delta t) = e^{-i\frac{\Delta t}{2}h^2_{[n]}} A_{[n-1,n]}(\frac{\Delta t}{2})$.
   (b) *Split* $A_{[n,n+1]}$ *into:* $A^{s_{n-1},s_n} = A^{s_{n-1}} R^{s_n}$.
   (c) *Calculate the right contraction* $F^{b_{n-1}}_{a_{n-1},a'_{n-1}}$.
   (d) *Compute* $A_{[n-1]}(\frac{\Delta t}{2}) = e^{i\frac{\Delta t}{2}h_{[n]}} A_{[n-1]}(\Delta t)$.

5.3 Infinite Lattice TDVP

For the first step the state is assumed to be in uMPS form $|\psi\rangle \to |\psi(A)\rangle$. Not that A is dependent on time. We will temporarily rewrite the matrix A^s_{ab} as A^j with the tuple $j = (s, a, b)$. The Schrödinger equation now reads as:

$$B^j \, |\partial_j \psi\rangle = -i\hat{H} \, |\psi\rangle \tag{5.2}$$

where $\partial_j = \frac{\partial}{\partial A^j}$ and $B = \dot{A}$. The challenge is to find the matrix B which solves the equation at a given time t. In general there is no solution to this equation within the matrix dimension χ. Within a fixed matrix dimension χ there can only be an approximate solution by minimizing the expression

$$|| B^j \, |\partial_j \psi\rangle + i\hat{H} \, |\psi\rangle \, ||$$

This minimization is done by projecting eq. (5.2) onto the tangent plane of $|\psi(A)\rangle$.

$$\bar{B}^{\bar{k}} \, \langle \partial_{\bar{k}} \psi | \partial_j \psi \rangle \, B^j = -i\bar{B}^{\bar{k}} \, \langle \partial_{\bar{k}} \psi | \, \hat{H} \, |\psi\rangle \tag{5.3}$$

What the TDVP algorithm does is to find a solution of eq. (5.3) with respect to B. Now that we have laid out the general idea, we can transform A^j and B^j back to A^s_{ab} and

B^s_{ab}. The derivative of $|\psi\rangle$ can now be written as:

$$|\dot{\psi}\rangle = \sum_{n\in\mathbb{Z}} \sum_{\{s\}=1}^{d} v_l^\dagger ... A^{s_{n-1}} B^{s_n} A^{s_{n+1}} ... v_r \, |s\rangle$$

Definition 5.1

We will consider Hamiltonians of the following form:

$$\hat{H} = \sum_i \hat{h}_i^1 + \sum_i \hat{h}_{i,i+1}^2 + \sum_{m,n,j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)}$$

i and j will denote the different lattice sites, $i, j \in \mathbb{Z}$. The operator $\hat{h}^1$ acts non trivially only on one site and will be called on-site potential. Similarly $\hat{h}^2$ is an operator that acts non trivially only on two neighboring sites and will be called nearest neighbor interaction. The last term of the Hamiltonian will be called long range interaction. Every $\hat{M}^m$ and $\hat{N}^m$ is a single site operator, where we sum over different pairs of operators with $m \in \mathbb{N}$. β_n and α_n are real scalars. By definition we assume that $\alpha_n > 0 \ \forall n$.

We will split eq. (5.3) up into its several terms.

$$\begin{aligned} \bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \partial_j \psi \rangle \, B^j = -i \bar{B}^{\bar{k}} \Bigg[& \langle \partial_{\bar{k}} \psi | \sum_i \hat{h}_i^1 \, |\psi\rangle + \langle \partial_{\bar{k}} \psi | \sum_i \hat{h}_{i,i+1}^2 \, |\psi\rangle \\ & + \langle \partial_{\bar{k}} \psi | \sum_{m,n,j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)} \, |\psi\rangle \Bigg] \end{aligned} \tag{5.4}$$

To calculate all the several terms we will define several tensors, that occur through the process.

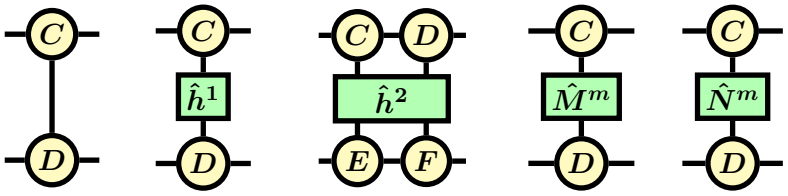

Figure 5.2: Graphical representation of T_D^C, h_D^C, h_{EF}^{CD}, M_D^C, and N_D^C.

Definition 5.2

$$
\begin{aligned}
T_D^C &= \sum_{s=1}^{d} C^s \otimes \bar{D}^s \\
h_D^C &= \sum_{s,t=1}^{d} \langle s| \hat{h}^1 |t\rangle \ \ (C^t \otimes \bar{D}^s) \\
h_{EF}^{CD} &= \sum_{s,t,u,v=1}^{d} \langle st| \hat{h}^2 |uv\rangle \ \ [(C^u D^v) \otimes (\bar{E}^s \bar{F}^t)] \\
M_D^C &= \sum_{s,t=1}^{d} \langle s| \hat{M}^m |t\rangle \ \ (C^t \otimes \bar{D}^s) \\
N_D^C &= \sum_{s,t=1}^{d} \langle s| \hat{N}^m |t\rangle \ \ (C^t \otimes \bar{D}^s)
\end{aligned}
$$

Note that T_A^A is the transfer matrix T.

Left Hand Side

The left hand side of the equation is a sum of tensor networks, where B and $\bar{B}$ take every possible position. This can be seen in fig. 5.3. The left and right eigenvectors of the transfer matrix "collapse" to the first appearance of either B or $\bar{B}$. There are three different cases to examine:

1. B and $\bar{B}$ are on the same position.
2. B is to the left of $\bar{B}$.
3. B is to the right of $\bar{B}$.

For point 2 and 3, one has to sum over all possible distances between B and $\bar{B}$. Every lattice site between them is represented by the transfer matrix. With that in mind the left hand side transforms to:

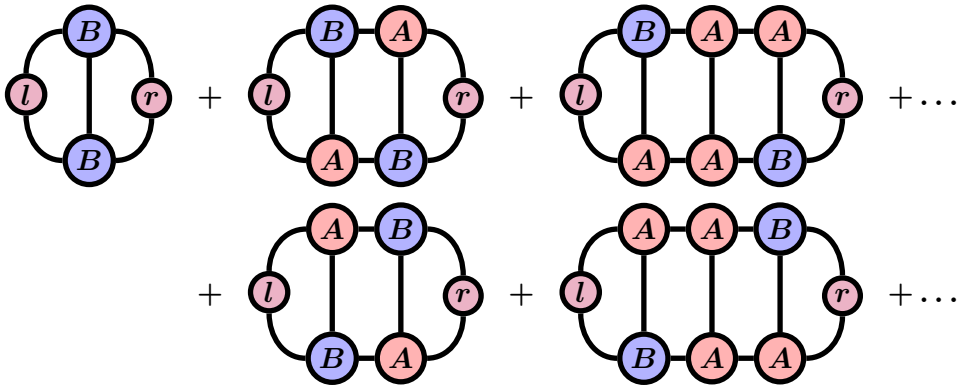

Figure 5.3: Graphical representation of $\frac{1}{|\mathbb{Z}|}\bar{B}^{\bar{k}}\langle\partial_{\bar{k}}\psi|\partial_j\psi\rangle B^j$. Note that this sum of tensor networks gets repeated for every lattice site (hence the factor $\frac{1}{|\mathbb{Z}|}$).

$$\begin{aligned}\bar{B}^{\bar{k}}\langle\partial_{\bar{k}}\psi|\partial_j\psi\rangle B^j = |\mathbb{Z}| \quad &\Big[\langle l|T_B^B|r\rangle + \langle l|T_A^BT_B^A|r\rangle + \langle l|T_A^BTT_B^A|r\rangle + \langle l|T_A^BT^2T_B^A|r\rangle + \dots \\ &+ \langle l|T_A^BT_B^A|r\rangle + \langle l|T_A^BTT_B^A|r\rangle + \langle l|T_A^BT^2T_B^A|r\rangle + \dots\Big] \\ = |\mathbb{Z}| \quad &\Big[\langle l|T_B^B|r\rangle + \langle l|T_A^B\Big(\sum_{n=0}^{\infty}T^n\Big)T_B^A|r\rangle + \langle l|T_B^A\Big(\sum_{n=0}^{\infty}T^n\Big)T_A^B|r\rangle\Big]\end{aligned} \tag{5.5}$$

The sums over all powers of T are geometric series. The usual formula to calculate this, fails for T.

$$\sum_{n=0}^{\infty}x^n = \frac{1}{1-x} \quad \forall|x|<1$$

The series will not converge for $\forall|x|\geq 1$. As can be seen in section 2 the transfer matrix T of a uniform Matrix Product State with a norm of 1 has a dominant eigenvalue of 1. Thus, for a normed state it is not possible to calculate the geometric series for the transfer matrix. There is however a projection operator Q that solves the problem.

$$Q = 1 - |r\rangle\langle l|$$

With this projector it is possible to calculate the geometric series of the transfer matrix.

$$\sum_{n=0}^{\infty}T^n = Q\left(1-QTQ\right)^{-1}Q + \sum_{n=0}^{\infty}|r\rangle\langle l|$$

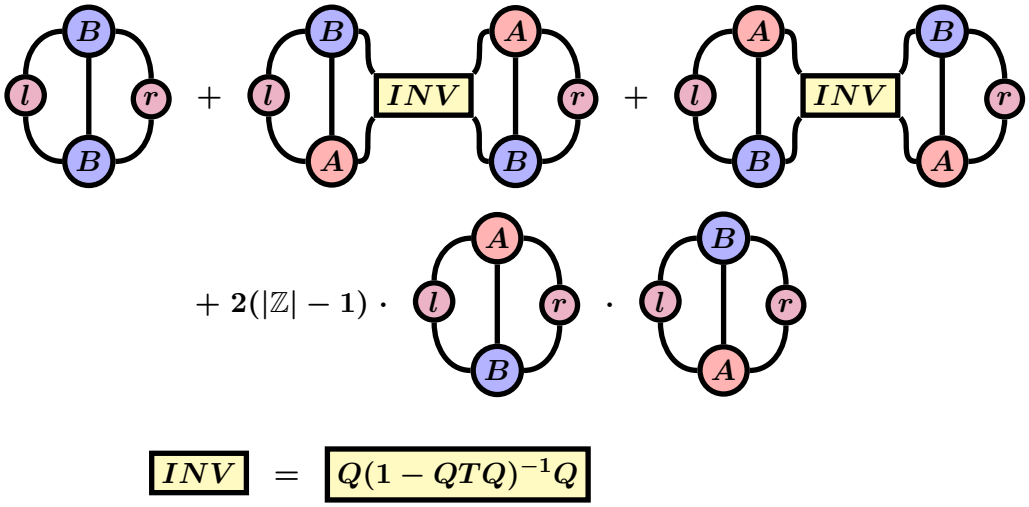

Figure 5.4: Graphical representation of $\frac{1}{|\mathbb{Z}|}\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi | \partial_j \psi \rangle B^j$ after the geometric series has been calculated. Note that this sum of tensor networks gets repeated for every lattice site (hence the factor $\frac{1}{|\mathbb{Z}|}$).

The whole calculation can be found in appendix A.1. Note that $Q\,(1 - QTQ)^{-1}\,Q$ is the pseudo inverse of $(1 - T)$. Now eq. (5.5) can be transformed into this expression.

$$\begin{aligned}\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi | \partial_j \psi \rangle B^j = |\mathbb{Z}| \;\Big[& \langle l |\, T_B^B \,| r \rangle + \langle l |\, T_A^B Q(1 - QTQ)^{-1} Q T_B^A \,| r \rangle \\ & + \langle l |\, T_B^A Q(1 - QTQ)^{-1} Q T_A^B \,| r \rangle \\ & + 2(|\mathbb{Z}|-1)\, \langle l |\, T_B^A \,| r \rangle \, \langle l |\, T_A^B \,| r \rangle \Big]\end{aligned}$$

The first three terms are all computable. The only problem arises if one wants to calculate the last term, because of the extra factor of $\mathbb{Z}$. This can be solved with a special parametrization of B. The extra factor of $|\mathbb{Z}|$ which is multiplied with all terms is there because of the infinite number of lattice sites. It will not prove to be a problem, because the same factor arises for all terms of eq. (5.3).

On-site Potential

The right hand side of eq. (5.3) will be split up into the three parts of the Hamiltonian and can be dealt with the same approach. Again, there a three cases to consider:

1. $\bar{B}$ sits on the same place as $\hat{h}^1$.
2. $\bar{B}$ is to the left of $\hat{h}^1$.
3. $\bar{B}$ is to the right of $\hat{h}^1$.

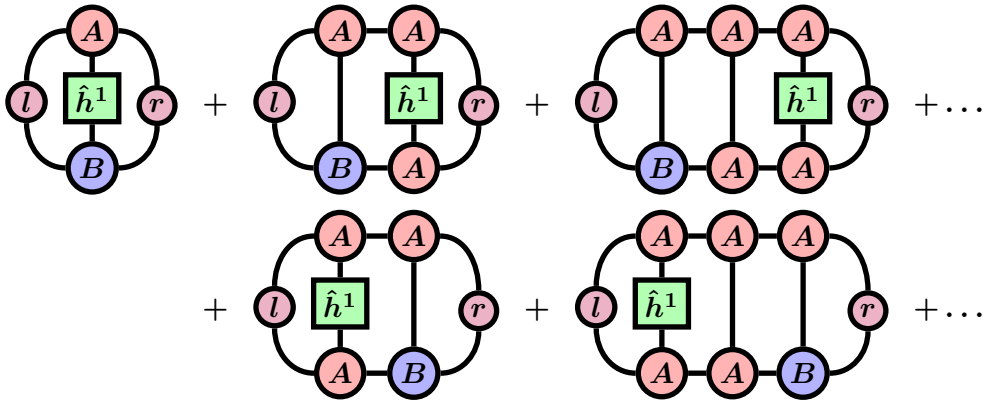

Figure 5.5: Graphical representation of $\frac{1}{|\mathbb{Z}|}\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi| \sum_i \hat{h}_i^1 |\psi\rangle$. Note that this sum of tensor networks gets repeated for every lattice site (hence the factor $\frac{1}{|\mathbb{Z}|}$).

Like before, in point 2 and 3 a summation over all different distances between $\bar{B}$ and $\hat{h}^1$ has to be done.

$$\begin{aligned}\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi| \sum_i \hat{h}_i^1 |\psi\rangle = |\mathbb{Z}| \; & \Big[\langle l| h_B^A |r\rangle + \langle l| T_B^A \Big(\sum_{n=0}^{\infty} T^n\Big) h_A^A |r\rangle + \langle l| h_A^A \Big(\sum_{n=0}^{\infty} T^n\Big) T_B^A |r\rangle\Big] \\ = |\mathbb{Z}| \; & \Big[\langle l| h_B^A |r\rangle + \langle l| T_B^A Q(1-QTQ)^{-1} Q h_A^A |r\rangle \\ & + \langle l| h_A^A Q(1-QTQ)^{-1} Q T_B^A |r\rangle \\ & + 2(|\mathbb{Z}|-1) \langle l| h_A^A |r\rangle \langle l| T_B^A |r\rangle\Big]\end{aligned}$$

Nearest Neighbor Interaction

In the case of the two site interaction, there are four cases:

1. $\bar{B}$ is on the left side, of the two sides, where $\hat{h}^2$ operates.
2. $\bar{B}$ is on the right side, of the two sides, where $\hat{h}^2$ operates.
3. $\bar{B}$ is to the left of $\hat{h}^2$.
4. $\bar{B}$ is to the right of $\hat{h}^2$.

This time there is a sum over all different distances between $\bar{B}$ and $\hat{h}^2$ in point 3 and 4.

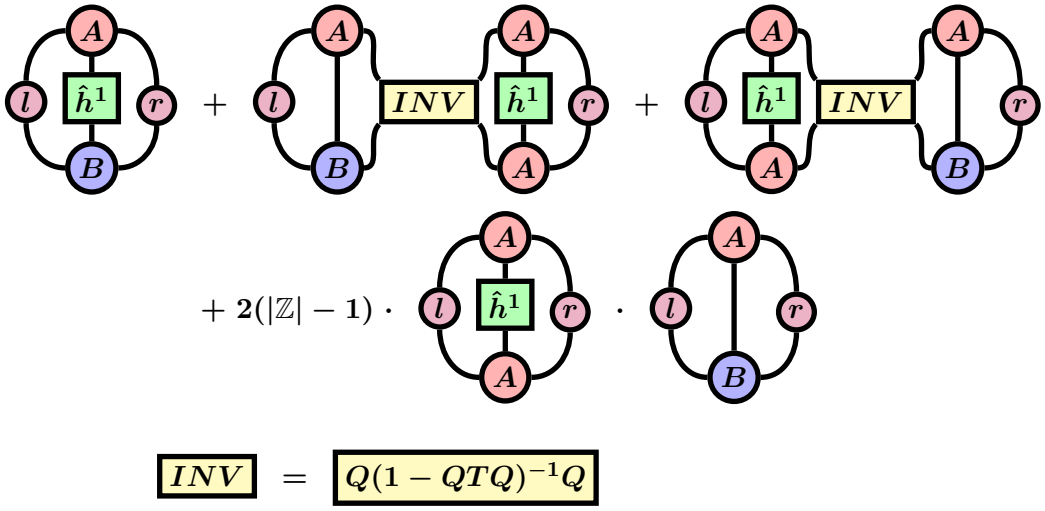

Figure 5.6: Graphical representation of $\frac{1}{|\mathbb{Z}|}\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi| \sum_i \hat{h}_i^1 |\psi\rangle$ after the geometric series has been calculated. Note that this sum of tensor networks gets repeated for every lattice site (hence the factor $\frac{1}{|\mathbb{Z}|}$).

$$\begin{aligned}
\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi| \sum_i \hat{h}_{i,i+1}^2 |\psi\rangle = |\mathbb{Z}| \; & \Big[\langle l| h_{BA}^{AA} |r\rangle + \langle l| h_{AB}^{AA} |r\rangle + \langle l| T_B^A \Big(\sum_{n=0}^{\infty} T^n \Big) h_{AA}^{AA} |r\rangle \\
& + \langle l| h_{AA}^{AA} \Big(\sum_{n=0}^{\infty} T^n \Big) T_B^A |r\rangle \Big] \\
= |\mathbb{Z}| \; & \Big[\langle l| h_{BA}^{AA} |r\rangle + \langle l| h_{AB}^{AA} |r\rangle \\
& + \langle l| T_B^A Q(1 - QTQ)^{-1} Q h_{AA}^{AA} |r\rangle \\
& + \langle l| h_{AA}^{AA} Q(1 - QTQ)^{-1} Q T_B^A |r\rangle \\
& + 2(|\mathbb{Z}|-2) \langle l| h_{AA}^{AA} |r\rangle \langle l| T_B^A |r\rangle \Big]
\end{aligned}$$

Long Range Interaction

For the long ranged interaction, there are five cases to examine:

1. $\hat{B}$ is to the left of both $\hat{M}$ and $\hat{N}$.
2. $\hat{B}$ is on the same place as $\hat{M}$.
3. $\hat{B}$ is between $\hat{M}$ and $\hat{N}$.
4. $\hat{B}$ is on the same place as $\hat{N}$.

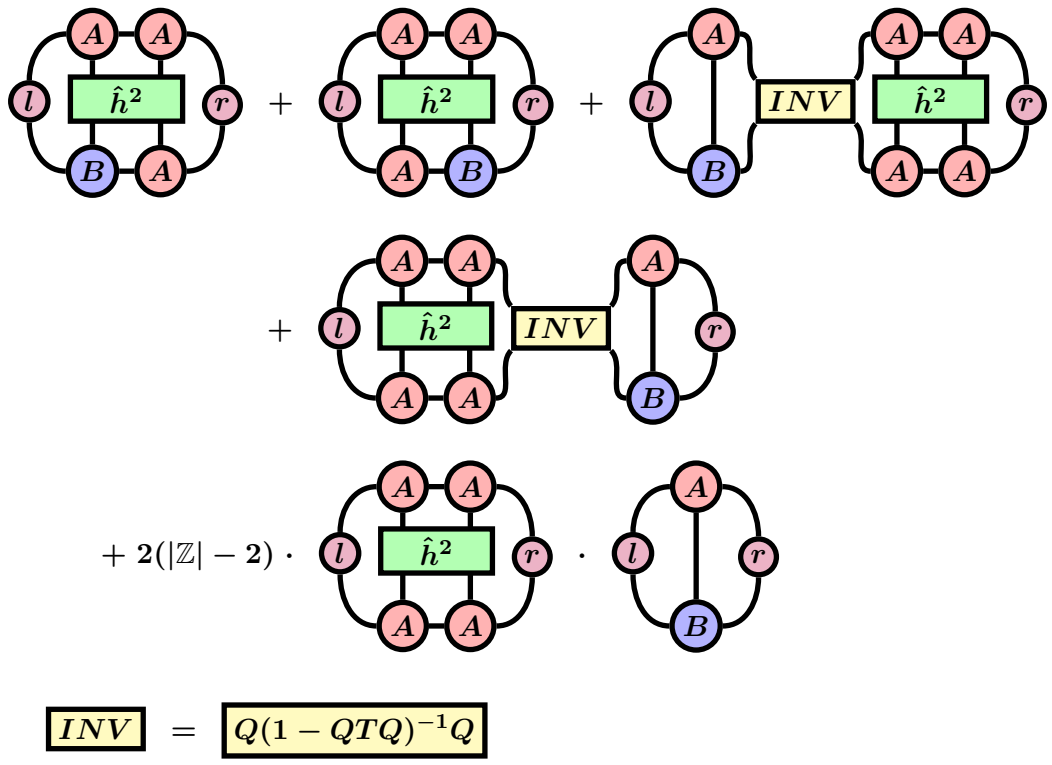

Figure 5.7: Graphical representation of $\frac{1}{|\mathbb{Z}|}\bar{B}^{\bar{k}}\langle\partial_{\bar{k}}\psi|\sum_i \hat{h}^2_{i,i+1}|\psi\rangle$ after the geometric series has been calculated. Note that this sum of tensor networks gets repeated for every lattice site (hence the factor $\frac{1}{|\mathbb{Z}|}$).

5. $\hat{B}$ is to the right of both $\hat{M}$ and $\hat{N}$.

In case 1 and 5 there is a sum over all distances between $\bar{B}$ and either $\hat{M}$ or $\hat{N}$. In every case an extra summation over all distances between $\hat{M}$ and $\hat{N}$ has to be done.

$$\sum_m \bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_{n,j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)} |\psi\rangle = *$$

$$\begin{aligned} * = |\mathbb{Z}| \sum_{m,n} \beta_n e^{-\alpha_n} \quad & \Big[\langle l| T_B^A \Big(\sum_{l=0}^{\infty} T^l \Big) M_A^A \Big(\sum_{l=0}^{\infty} (e^{-\alpha_n} T)^l \Big) N_A^A |r\rangle \\ & + \langle l| M_B^A \Big(\sum_{l=0}^{\infty} (e^{-\alpha_n} T)^l \Big) N_A^A |r\rangle \\ & + e^{-\alpha_n} \langle l| M_A^A \Big(\sum_{l=0}^{\infty} (e^{-\alpha_n} T)^l \Big) T_B^A \Big(\sum_{l=0}^{\infty} (e^{-\alpha_n} T)^l \Big) N_A^A |r\rangle \\ & + \langle l| M_A^A \Big(\sum_{l=0}^{\infty} (e^{-\alpha_n} T)^l \Big) N_B^A |r\rangle \\ & + \langle l| M_A^A \Big(\sum_{l=0}^{\infty} (e^{-\alpha_n} T)^l \Big) N_A^A \Big(\sum_{l=0}^{\infty} T^l \Big) T_B^A |r\rangle \Big] \end{aligned}$$

The geometric series of $e^{-\alpha} T$ converges without the use of QTQ, because by assumption $\alpha > 0 \Rightarrow e^{-\alpha} < 1$. T has exactly one eigenvalue of 1 and all other eigenvalues have a smaller absolute value, thus $e^{-\alpha} T$ has only eigenvalues with absolute values smaller than 1. This allows the geometric series to converge. There is an extra factor of $e^{-\alpha_n}$ in the fourth line of the equation. This is because we omitted this because we pulled this factor away from T_B^A in front of the term.

$$\sum_m \bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_{n,j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)} |\psi\rangle = *$$

$$\begin{aligned} * = |\mathbb{Z}| \sum_{m,n} \beta_n e^{-\alpha_n} \quad & \Big[\langle l| T_B^A Q (1 - QTQ)^{-1} Q M_A^A (1 - e^{-\alpha_n} T)^{-1} N_A^A |r\rangle \\ & + \langle l| M_B^A (1 - e^{-\alpha_n} T)^{-1} N_A^A |r\rangle \\ & + e^{-\alpha_n} \langle l| M_A^A (1 - e^{-\alpha_n} T)^{-1} T_B^A (1 - e^{-\alpha_n} T)^{-1} N_A^A |r\rangle \\ & + \langle l| M_A^A (1 - e^{-\alpha_n} T)^{-1} N_B^A |r\rangle \\ & + \langle l| M_A^A (1 - e^{-\alpha_n} T)^{-1} N_A^A Q (1 - QTQ)^{-1} Q T_B^A |r\rangle \\ & + 2(|\mathbb{Z}| - 2) \langle l| T_B^A |r\rangle \langle l| M_A^A (1 - e^{-\alpha_n} T)^{-1} N_A^A |r\rangle \Big] \end{aligned}$$

In summary, if we can find a $B = \dot{A}$ that solves eq. (5.4) we can numerically integrate the Schrödinger equation and compute $\psi(t)$. The single terms of this equation were expanded in this chapter.

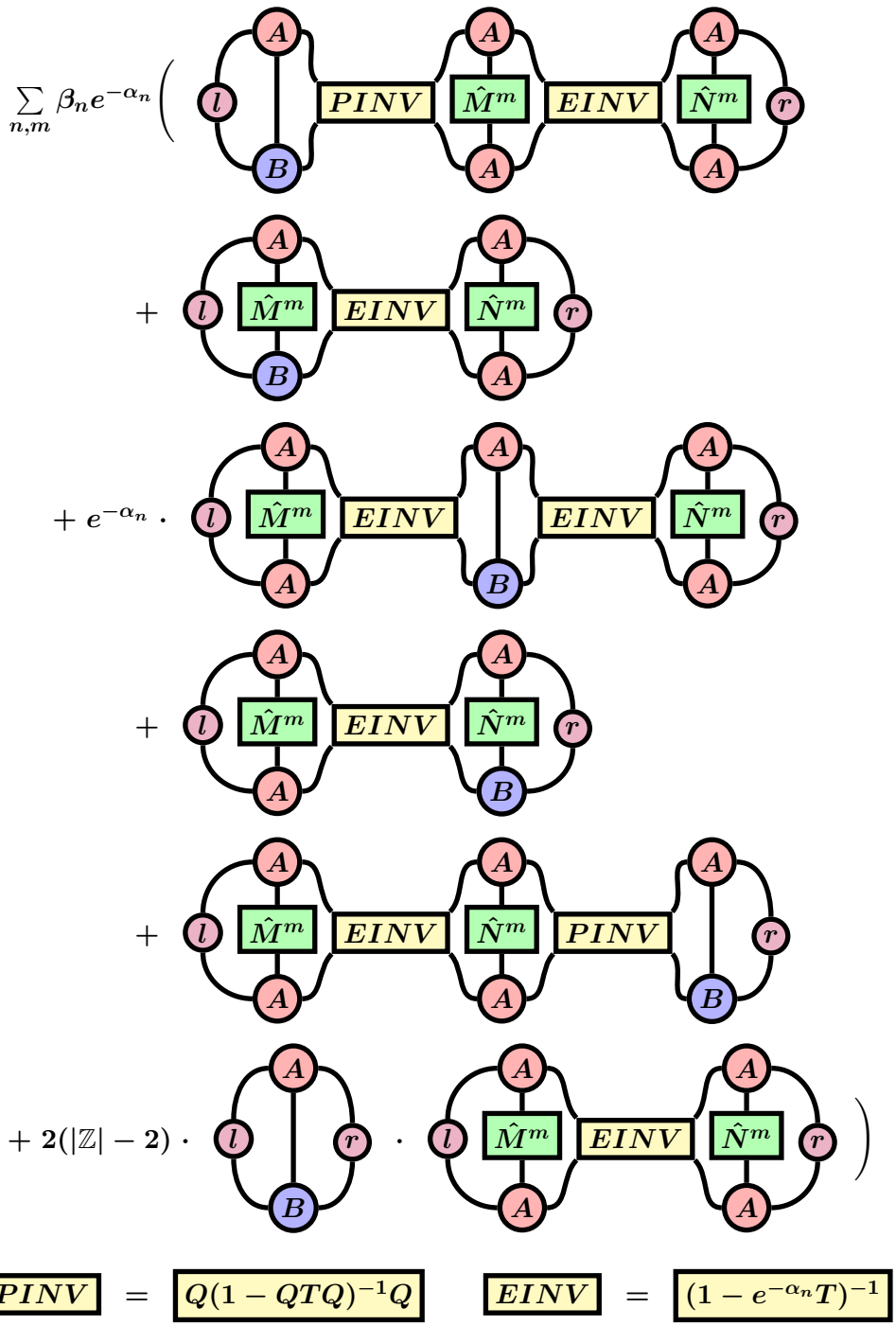

Figure 5.8: Graphical representation of $\frac{1}{|\mathbb{Z}|}\bar{B}^{\bar{k}} \langle \partial_{\bar{k}}\psi | \sum_{l,j>i} \hat{M}_i^l \hat{N}_j^l \beta e^{-\alpha(j-i)} |\psi\rangle$ after the geometric series has been calculated. Note that this sum of tensor networks gets repeated for every lattice site (hence the factor $\frac{1}{|\mathbb{Z}|}$).

Computation

As mentioned above, the task at hand now is to find a B so that eq. (5.4) is satisfied. At first a suitable gauge for B has to be found, so that the diverging factors of this equation can be eliminated. One possible way to fix the gauge of B is:

$$\begin{aligned} \langle l| T_A^B &= 0 \\ \sum_s A^{s\dagger} l B^s &= 0 \end{aligned}$$

This gauge has several advantages. The norm of the state is preserved and the left eigenvector does not change in first order. Furthermore all terms with $\langle l| T_A^B$ or $\langle l| T_B^A$ in it will vanish. This is especially important for terms, that have an extra factor of $\mathbb{Z}$. With this gauge eq. (5.4) transforms into:

$$\begin{aligned} \langle l| T_B^B |r\rangle = -i\Bigg(& \langle l| h_B^A |r\rangle + \langle l| h_A^A Q(1-QTQ)^{-1} QT_B^A |r\rangle \\ & + \langle l| h_{BA}^{AA} |r\rangle + \langle l| h_{AB}^{AA} |r\rangle \\ & + \langle l| h_{AA}^{AA} Q(1-QTQ)^{-1} QT_B^A |r\rangle \\ & + \sum_{m,n} \Big(\langle l| M_B^A (1-e^{-\alpha_n T})^{-1} N_A^A |r\rangle \\ & + e^{-\alpha_n} \langle l| M_A^A (1-e^{-\alpha_n T})^{-1} T_B^A (1-e^{-\alpha_n T})^{-1} N_A^A |r\rangle \\ & + \langle l| M_A^A (1-e^{-\alpha_n T})^{-1} N_B^A |r\rangle \\ & + \langle l| M_A^A (1-e^{-\alpha_n T})^{-1} N_A^A Q(1-QTQ)^{-1} QT_B^A |r\rangle \Big) \Bigg) \end{aligned}$$

Now we want to find a parametrization for $B = B(x)$ so that $\langle l| T_A^{B(x)} = 0$. The parametrization for B used in [27] is defined as:

Definition 5.3: Parametrization of B

$$B^s(x) = l^{-\frac{1}{2}} V^s x r^{-\frac{1}{2}} \tag{5.6}$$

where

$$
\begin{aligned}
V^s_{a,b} &= V_{(sa),b} = \mathrm{Null}\,(L) \\
\sum_s V^{s\dagger} V^s &= 1 \\
L_{c,(sa)} &= \left(A^{s\dagger} l^{\frac{1}{2}} \right)_{c,a}
\end{aligned}
$$

If we plug eq. (5.6) into the the gauge condition, we can see that the condition is in fact fulfilled.

$$
\begin{aligned}
0 &= \sum_s A^{s\dagger} l \; B^s \\
&= \sum_s A^{s\dagger} l \; l^{-\frac{1}{2}} V^s x r^{-\frac{1}{2}} \\
&= \sum_s \underbrace{A^{s\dagger} l^{\frac{1}{2}} V^s}_{=0,\text{by def.}} x r^{-\frac{1}{2}}
\end{aligned}
$$

In general $x \in \mathbb{C}^{(d-1)\chi\times\chi}$ (see appendix A.2).
To simplify the equations for x we will define column and row vectors that include the inverse and pseudoinverse of $(1-T)$ and $(1-e^{-\alpha_n}T)$.

Definition 5.4: K

$$
\begin{aligned}
\langle K_{OS}| &= \langle l|\, h^A_A Q (1-QTQ)^{-1} Q \\
\langle K_{NN}| &= \langle l|\, h^{AA}_{AA} Q (1-QTQ)^{-1} Q \\
\langle K^n_l| &= e^{-\alpha_n} \;\; \langle l|\, M^A_A (1-e^{-\alpha_n}T)^{-1} \\
|K^n_r\rangle &= e^{-\alpha_n} \;\; (1-e^{-\alpha_n}T)^{-1} N^A_A\, |r\rangle \\
\langle K_{LR}| &= \sum_n \beta_n \langle K^n_l|\, N^A_A Q (1-QTQ)^{-1} Q
\end{aligned}
$$

These vectors also have a matrix representation exactly like $\langle l|$ and $|r\rangle$. Ant the same rules for products like $\langle l|\,T$ (see section 2) apply for e.g. $\langle K_{OS}|\,T$. In general it is not a good idea to calculate these vectors directly, because the computational cost of calculating the inverse of T goes like $\mathcal{O}(\chi^6)$. An iterative process of calculating these vectors can be found (see appendix A.3).
Furthermore we will combine every other tensor than x in a single matrix F. We will define this in advance:

Definition 5.5: F

$$
\begin{aligned}
F_{OS} &= \sum_{s,t} V^{s\dagger} l^{-\frac{1}{2}} \Big(\langle s| \hat{h}^1 |t\rangle \; l \; A^t + K_{OS} \; A^t \Big) r^{\frac{1}{2}} \\
C^{s,t} &= \sum_{u,v} \langle s,t| \hat{h}^2 |u,v\rangle \; A^u A^v \\
F_{NN} &= \sum_{s,t} V^{s\dagger} l^{\frac{1}{2}} \; C^{s,t} \; r \; A^{t\dagger} r^{-\frac{1}{2}} + \sum_{s} V^{s\dagger} l^{-\frac{1}{2}} \Big(\sum_{t} A^{t\dagger} \; l \; C^{t,s} + K_{NN} \; A^s \Big) r^{\frac{1}{2}} \\
F_{LR} &= \sum_{m} \Bigg(\sum_{s,t} \langle s| \hat{M}^m |t\rangle \, V^{s\dagger} l^{\frac{1}{2}} A^t \Big(\sum_{n} \beta_n K_r^n \Big) r^{-\frac{1}{2}} \\
&\qquad + \sum_{s} \sum_{n} \; \beta_n V^{s\dagger} l^{-\frac{1}{2}} K_l^n A^t K_r^n r^{-\frac{1}{2}} \\
&\qquad + \sum_{s,t} \langle s| \hat{N}^m |t\rangle \, V^{s\dagger} l^{-\frac{1}{2}} \Big(\sum_{n} \beta_n K_l^n \Big) A^t r^{\frac{1}{2}} \\
&\qquad + \sum_{s} V^{s\dagger} l^{-\frac{1}{2}} K_{LR} A^s r^{\frac{1}{2}} \Bigg) \\
F &= F_{OS} + F_{NN} + F_{LR}
\end{aligned}
$$

With these definitions eq. (5.4) will take the form of a simple matrix equation. We will deal with its left hand side, the on-site potential, the nearest neighbor term, and the long range interaction separately. In every calculation below we will use the cyclic invariance of the trace and pull $x^\dagger$ to the left.

Left Hand Side

$$
\begin{aligned}
\bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \partial_j \psi \rangle \, B^j &= |\mathbb{Z}| \;\; \langle l| \, T^{B(x)}_{B(x)} \, |r\rangle \\
&= |\mathbb{Z}| \;\; \mathrm{tr}\Big(\sum_s B^{s\dagger}(x) \; l \; B^s(x) \; r \Big) \\
&= |\mathbb{Z}| \;\; \mathrm{tr}\Big(\sum_s r^{-\frac{1}{2}} x^\dagger V^{s\dagger} l^{-\frac{1}{2}} \; l \; l^{-\frac{1}{2}} V^s x r^{-\frac{1}{2}} \; r \Big) \\
&= |\mathbb{Z}| \;\; \mathrm{tr}\Big(x^\dagger \underbrace{\sum_s V^{s\dagger} V^s}_{=1} x \Big) \\
&= |\mathbb{Z}| \;\; \mathrm{tr}\Big(x^\dagger x \Big)
\end{aligned}
$$

On-site Potential

$$\begin{aligned}
\bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_i \hat{h}_i^1 | \psi \rangle &= |\mathbb{Z}| \ \Big[\langle l | \, h^A_{B(x)} \, | r \rangle + \langle l | \, h^A_A Q (1 - QTQ)^{-1} Q T^A_{B(x)} \, | r \rangle \Big] \\
&= |\mathbb{Z}| \ \Big[\langle l | \, h^A_B \, | r \rangle + \langle K_{OS} | \, T^A_{B(x)} \, | r \rangle \Big] \\
&= |\mathbb{Z}| \ \Big[\sum_{s,t} \langle s | \, \hat{h}^1 \, | t \rangle \, B^{s\dagger}(x) \ l \ A^t \ \ r + B^{s\dagger}(x) \ K_{OS} \ A^t \ r \Big] \\
&= |\mathbb{Z}| \ \ \mathrm{tr} \Big(\sum_{s,t} x^\dagger V^{s\dagger} l^{-\frac{1}{2}} \Big(\langle s | \, \hat{h}^1 \, | t \rangle \, l \ A^t + K_{OS} \ A^t \Big) r^{\frac{1}{2}} \Big) \\
&= |\mathbb{Z}| \ \ \mathrm{tr} \Big(x^\dagger F_{OS} \Big)
\end{aligned}$$

Nearest Neighbor Interaction

$$\begin{aligned}
\bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_i \hat{h}^2_{i,i+1} | \psi \rangle &= |\mathbb{Z}| \ \ \Big[\langle l | \, h^{AA}_{B(x)A} \, | r \rangle + \langle l | \, h^{AA}_{AB(x)} \, | r \rangle \\
&\qquad + \langle l | \, h^{AA}_{AA} Q (1 - QTQ)^{-1} Q T^A_{B(x)} \, | r \rangle \Big] \\
&= |\mathbb{Z}| \ \ \mathrm{tr} \Big(\langle l | \, h^{AA}_{B(x)A} \, | r \rangle + \langle l | \, h^{AA}_{AB(x)} \, | r \rangle + \langle K_{NN} | \, T^A_{B(x)} \, | r \rangle \Big) \\
&= |\mathbb{Z}| \ \ \mathrm{tr} \Big(\sum_{s,t,u,v} \langle s,t | \, \hat{h}^2 \, | u,v \rangle \, A^{t\dagger} B^{s\dagger}(x) \ l \ A^u A^v \ r \\
&\qquad + \sum_{s,t,u,v} \langle t,s | \, \hat{h}^2 \, | u,v \rangle \, B^{s\dagger}(x) A^{t\dagger} \ l \ A^u A^v \ r \\
&\qquad + \sum_s B^{s\dagger}(x) \ K_{NN} \ A^s \ r \Big)
\end{aligned}$$

$$\begin{aligned}
\bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_i \hat{h}^2_{i,i+1} | \psi \rangle &= |\mathbb{Z}| \ \ \mathrm{tr} \Big(\sum_{s,t} x^\dagger V^{s\dagger} l^{\frac{1}{2}} \ C^{s,t} \ r \ A^{t\dagger} r^{-\frac{1}{2}} \\
&\qquad + \sum_{s,t} x^\dagger V^{s\dagger} l^{-\frac{1}{2}} A^{t\dagger} \ l \ C^{t,s} \ r^{\frac{1}{2}} \\
&\qquad + \sum_s x^\dagger V^{s\dagger} l^{-\frac{1}{2}} \ K_{NN} \ A^s \ r^{\frac{1}{2}} \Big) \\
&= |\mathbb{Z}| \ \ \mathrm{tr} \Big(x^\dagger F_{NN} \Big)
\end{aligned}$$

Long Range Interaction

$$\sum_{m,n} \bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_{j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)} |\psi\rangle = *$$

$$\begin{aligned} * = |\mathbb{Z}| \; \sum_{m,n} \beta_n e^{-\alpha_n} \; \Big[& \langle l| \, M^A_{B(x)} (1 - e^{-\alpha_n} T)^{-1} N^A_A \, |r\rangle \\ & + \; e^{-\alpha_n} \, \langle l| \, M^A_A (1 - e^{-\alpha_n} T)^{-1} T^A_{B(x)} (1 - e^{-\alpha_n} T)^{-1} N^A_A \, |r\rangle \\ & + \; \langle l| \, M^A_A (1 - e^{-\alpha_n} T)^{-1} N^A_{B(x)} \, |r\rangle \\ & + \; \langle l| \, M^A_A (1 - e^{-\alpha_n} T)^{-1} N^A_A Q (1 - QTQ)^{-1} Q T^A_{B(x)} \, |r\rangle \Big] \end{aligned}$$

$$\sum_{m,n} \bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_{j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)} |\psi\rangle = *$$

$$\begin{aligned} * = |\mathbb{Z}| \; \sum_{m} \; \Bigg[& \langle l| \, M^A_{B(x)} \Big(\sum_n \; \beta_n \, |K^n_r\rangle \Big) \\ & + \; \sum_n \; \beta_n \, \langle K^n_l| \, T^A_{B(x)} \, |K^n_r\rangle \\ & + \; \Big(\sum_n \; \beta_n \, \langle K^n_l| \Big) N^A_{B(x)} \, |r\rangle \\ & + \; \langle K_{LR}| \, T^A_{B(x)} \, |r\rangle \Bigg] \end{aligned}$$

$$\sum_{m,n} \bar{B}^{\bar{k}} \langle \partial_{\bar{k}} \psi | \sum_{j>i} \hat{M}_i^m \hat{N}_j^m \beta_n e^{-\alpha_n (j-i)} |\psi\rangle = *$$

$$\begin{aligned} * = |\mathbb{Z}| \; \sum_{m} \mathrm{tr} \Bigg(& \sum_{s,t} \langle s| \, \hat{M}^m \, |t\rangle \, B^{s\dagger}(x) l A^t \Big(\sum_n \beta_n K^n_r \Big) \\ & + \; \sum_s \sum_n \; \beta_n B^{s\dagger}(x) K^n_l A^t K^n_r \\ & + \; \sum_{s,t} \langle s| \, \hat{N}^m \, |t\rangle \, B^{s\dagger}(x) \Big(\sum_n \beta_n K^n_l \Big) A^t r \\ & + \; \sum_s B^{s\dagger}(x) K_{LR} A^s r \Bigg) \\ = |\mathbb{Z}| \; \mathrm{tr} \Big(& x^\dagger F_{LR} \Big) \end{aligned}$$

Projected Schrödinger Equation

Finally, if we plug those expressions into eq. (5.4), the projected Schrödinger equation, we arrive at:

$$\mathrm{tr}\Big(x^\dagger x\Big) = -i\ \mathrm{tr}\left((x^\dagger(F_{OS}+F_{NN}+F_{LR})\right) = -i\ \mathrm{tr}(x^\dagger F)$$

This is an equation that can be solved with ease:

$$\mathrm{tr}\Big(x^\dagger x + ix^\dagger F\Big) = 0$$
$$\mathrm{tr}\Big(x^\dagger(x+iF)\Big) = 0$$

And we arrive at the final solution:

Result 5.2

$$x = -iF \tag{5.7}$$

With this x we can calculate $\dot{A} = B(x)$. Now there is a map where $\dot{A}(t)$ can be calculated for an arbitrary $A(t)$. The whole process boils down to use a numerical integrator to do the time evolution. The simplest numerical integrator is the so called Euler integrator:

$$A(t+\Delta t) = A(t) + \Delta t\dot{A}(t)$$

According to [27] this is sufficient for imaginary time evolutions (ground state search), but for real time evolutions a better integrator must be used.

Algorithm 5.3: Ground state search with TDVP and imaginary time evolution

1. *Start with a random uMPS A.*
2. *Calculate the dominant eigenvalue. (eq. (2.1), right hand side)*
3. *Renormalize A. (eq. (2.2))*
4. *Compute left and right eigenvector. (eq. (2.1), right hand side)*
5. *Compute V. (def. 5.3)*
6. *Compute K_{OS}, K_{NN}, K_l^n, K_r^n, and K_{LR}. (def. 5.4)*
7. *Compute F_{OS}, F_{NN}, and F_{LR}. (def. 5.5)*
8. *Compute x. (eq. (5.7))*
9. *Compute B. (def. 5.3)*
10. *Compute $A(t+\Delta t) = A - i\Delta t B$.*
11. *If not converged go back to 2.*

6 Time Evolution with Matrix Product Operators

In 2014 Zaletel et al. introduced a new way to calculate the time evolution of a Matrix Product State [36]. The main idea behind this approach is to find an approximate representation of the time evolution operator $e^{-it\hat{H}}$ in Matrix Product Operator form. After the introduction of the Time-Dependent Variational Principle, this is the second algorithm that is capable of time evolving systems with long ranged interactions and still maintain a constant error per site. This new idea integrates well with the existing framework of Matrix Product States and operators, because one only needs to apply an operator to conduct a time evolution.

6.1 Basics

It is possible to rewrite the Hamiltonian of a one dimensional chain with long ranged interactions into this form:

$$\hat{H} = \hat{H}_{L_i} \otimes \mathbb{1}_{R_i} + \mathbb{1}_{L_i} \otimes \hat{H}_{R_i} + \sum_{a_i=1}^{N_i} \hat{h}_{L_i,a_i} \otimes \hat{h}_{R_i,a_i}$$

Here we have split the chain into two separate chains at sites i and $i+1$. The term $\hat{H}_{L_i}$ are the parts of the Hamiltonian that act only on sites to the left of site i and $\hat{H}_{R_i}$ are the parts that are to the right of site $i+1$. The summands of H that act on both subchains are denoted by the sum $\sum_{a_i=1}^{N_i} \hat{h}_{L_i,a_i} \otimes \hat{h}_{R_i,a_i}$, where N_i is the number of interactions.

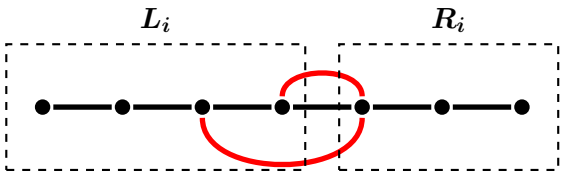

Figure 6.1: Graphical representation of the split into two chains. The red lines mark interactions between sites on both subchains. Thus, in this case $N_i = 2$.

If we want to move the location of the split to the left we can do so by calculating the following matrix product:

$$\begin{pmatrix} \hat{H}_{R_{i-1}} \\ \hat{h}_{R_{i-1}} \\ \mathbb{1}_{R_{i-1}} \end{pmatrix} = \begin{matrix} 1 \\ N_{i-1} \\ 1 \end{matrix} \overset{\begin{matrix} 1 & N_i & 1 \end{matrix}}{\begin{pmatrix} \hat{\mathbb{1}} & \hat{C} & \hat{D} \\ 0 & \hat{A} & \hat{B} \\ 0 & 0 & \hat{\mathbb{1}} \end{pmatrix}}_{[i]} \otimes \begin{pmatrix} \hat{H}_{R_i} \\ \hat{h}_{R_i} \\ \hat{\mathbb{1}}_{R_i} \end{pmatrix} \tag{6.1}$$

Where the first matrix on the right side of the equation (in this paragraph called $W_{[i]}$) is in block form. The small indices outside of the matrix denote the size of the block. So $\hat{A}$ is a $N_{i-1} \times N_i$ matrix of operators, $\hat{C}$ is $1 \times N_i$ row vector, $\hat{B}$ is a $N_{i-1} \times 1$ column vector, and $\hat{D}$ is a single operator. Note that all operators contained in $\hat{A}$, $\hat{B}$, $\hat{C}$, and $\hat{D}$ only act on site i. Because we can iterate through the whole chain that way, it follows that the first matrix on the right hand side of the equation above is the Matrix Product Operator form of the Hamiltonian $\hat{H}$.

$$\hat{H} = \sum_{\boldsymbol{s},\boldsymbol{s'}} \prod_i W_{[i]}^{s_i,s_i'} \left|\boldsymbol{s'}\right\rangle \left\langle \boldsymbol{s}\right|$$

The idea presented in [36] is to find an approximation of the time evolution operator $\hat{U}(t) = e^{-it\hat{H}}$ that can be written in Matrix Product Operator form and calculated from the matrices $\hat{A}$, $\hat{B}$, $\hat{C}$, and $\hat{D}$.
In general the time evolution operator can be written as Taylor expansion with respect to t:

$$\hat{U}(\tau) = e^{\tau\hat{H}} = 1 + \tau \sum_x \hat{H}_x + \frac{1}{2}\tau^2 \sum_{x,y} \hat{H}_x \hat{H}_y + \dots$$

Here $\tau = -it$ and $\hat{H} = \sum_x \hat{H}_x$, where the $\hat{H}_x$ are some arbitrary operators (not necessarily acting just on one site). If we want to express $U(\tau)$ in terms of a Matrix Product Operator the first problems arise in the second order of τ. There are two important simplifications to the Taylor series, where a MPO can be calculated. The Matrix Product Operators that emerge from those simplifications are called W^I and W^{II}.

6.2 W^I

The first way to simplify the Taylor series of $\hat{U}(\tau) = e^{\tau\hat{H}}$ is to discard all terms in second order where there is no partition into two subchains where $\hat{H}_x$ acts only on the left subchain and $\hat{H}_y$ acts only on the right side.

$$\hat{U}(\tau) \approx \hat{U}^I(\tau) = 1 + \tau \sum_x \hat{H}_x + \tau^2 \sum_{x<y} \hat{H}_x \hat{H}_y + \tau^3 \sum_{x<y<z} \hat{H}_x \hat{H}_y \hat{H}_z + \dots$$

Here $\sum_{x<y} \hat{H}_x \hat{H}_y$ is defined as sum over all summands of $\hat{H}$, where $\hat{H}_x$ acts only sites that are strictly to the left of all sites that are affected by $\hat{H}_y$.
For this simplification of the time evolution operator an exact MPO representation can be found and is named $\hat{W}^I$. If we take a look at the original structure of the Hamiltonian, we see that all of its terms are of one of the following forms:

1. Identity operator on every site except i, where the on site term $\hat{D}$ is.

$$\hat{\mathbb{1}}_{[1]} \dots \hat{\mathbb{1}}_{[i-1]} \hat{D}_{[i]} \hat{\mathbb{1}}_{[i+1]} \dots \hat{\mathbb{1}}_{[N]}$$

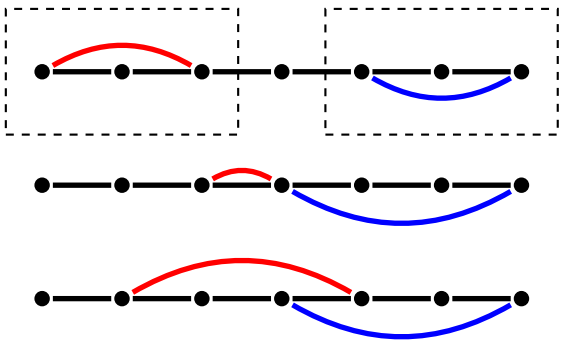

Figure 6.2: Hypothetical interactions terms of $\hat{H}_x$ (red) and $\hat{H}_y$ (blue). The second and the third combinations are discarded because the red interactions are not strictly to the left of the blue interactions.

2. Chain of the form $\hat{C}\hat{A}\ldots\hat{A}\hat{B}$, where the number of matrices $\hat{A}$ is greater or equal to zero.

$$\hat{\mathbb{1}}_{[1]}\ldots\hat{\mathbb{1}}_{[i-1]}\hat{C}_{[i]}\hat{A}_{[i+1]}\hat{B}_{[i+2]}\hat{\mathbb{1}}_{[i+3]}\ldots\hat{\mathbb{1}}_{[N]}$$

With this we can see that the summands of $\hat{U}^I$ as defined above look like this:

$$\hat{\mathbb{1}}_{[1]}\overbrace{\hat{D}_{[2]}}^{\hat{H}_x}\hat{\mathbb{1}}_{[3]}\ldots\hat{\mathbb{1}}_{[i-1]}\overbrace{\hat{C}_{[i]}\hat{A}_{[i+1]}\hat{B}_{[i+2]}}^{\hat{H}_y}\hat{\mathbb{1}}_{[i+3]}\ldots\hat{\mathbb{1}}_{[j-1]}\overbrace{\hat{C}_{[j]}\hat{B}_{[j+1]}}^{\hat{H}_z}\hat{\mathbb{1}}_{[i+2]}\ldots\hat{\mathbb{1}}_{[N]}$$

Here the condition $x < y < z$ holds. Note that there can not be products of the form $\hat{D}_{[i]}\hat{C}_{[i]}$ (same site) or similar terms, because that would violate the assumption that $\hat{H}_x$ and $\hat{H}_y$ do not overlap. If we describe operator strings of the product above with a finite state machine, we can see that there are two different states at an arbitrary site i:

1. The number of occurrences left of site i are equal for $\hat{B}$ and $\hat{C}$.
2. $\hat{C}$ appeared once more than $\hat{B}$ left of i.

For the five different matrices, there are the following transitions between the states:

- $\hat{\mathbb{1}}$: Can only occur at state 1 and does not change the number of appearances of $\hat{B}$ and $\hat{C}$. Thus, the state does not change, $1 \to 1$.
- $\hat{D}$: Can only occur at state 1 and does not change the number of appearances of $\hat{B}$ and $\hat{C}$. Thus, the state does not change, $1 \to 1$.
- $\hat{C}$: Can only occur at state 1 and increases the number of appearances of $\hat{C}$. Thus, there is a state transition, $1 \to 2$.
- $\hat{A}$: Can only occur at state 2 and does not change the number of appearances of $\hat{B}$ and $\hat{C}$. Thus, the state does not change, $2 \to 2$.

- $\hat{B}$: Can only occur at state 2 and increases the number of appearances of $\hat{B}$. Thus, there is a state transition, $2 \to 1$.

With these transitions the Matrix Product Operator can be derived.

Definition 6.1: W^I

$$
\begin{aligned}
W^I_{[1]}(\tau) &= \begin{pmatrix} 1+\tau\hat{D} & \sqrt{\tau}\hat{C} \end{pmatrix} \\
W^I_{[n]}(\tau) &= \begin{pmatrix} 1+\tau\hat{D} & \sqrt{\tau}\hat{C} \\ \sqrt{\tau}\hat{B} & \hat{A} \end{pmatrix} \quad \forall n : 1 < n < N \\
W^I_{[N]}(\tau) &= \begin{pmatrix} 1+\tau\hat{D} \\ \sqrt{\tau}\hat{B} \end{pmatrix}
\end{aligned}
$$

Theorem 6.1

W^I is the Matrix Product Operator of $\hat{U}^I$.

Proof. We are going to look at the same example as above. For example, one summand of $\tau^3 \sum_{x<y<z} \hat{H}_x \hat{H}_y \hat{H}_z$ will look like this:

$$
\tau^3 \hat{\mathbb{1}}_{[1]} \overbrace{\hat{D}_{[2]}}^{\hat{H}_x} \hat{\mathbb{1}}_{[3]} \dots \hat{\mathbb{1}}_{[i-1]} \overbrace{\hat{C}_{[i]} \hat{A}_{[i+1]} \hat{B}_{[i+2]}}^{\hat{H}_y} \hat{\mathbb{1}}_{[i+3]} \dots \hat{\mathbb{1}}_{[j-1]} \overbrace{\hat{C}_{[j]} \hat{B}_{[j+1]}}^{\hat{H}_z} \hat{\mathbb{1}}_{[i+2]} \dots \hat{\mathbb{1}}_{[N]}
$$

Now we split the power of τ and move it into the matrix product and remove the site indices for the sake of brevity.

$$
\hat{\mathbb{1}}(\tau\hat{D})\hat{\mathbb{1}} \dots \hat{\mathbb{1}}(\sqrt{\tau}\hat{C})\hat{A}(\sqrt{\tau}\hat{B})\hat{\mathbb{1}} \dots \hat{\mathbb{1}}(\sqrt{\tau}\hat{C})(\sqrt{\tau}\hat{B})\hat{\mathbb{1}} \dots \hat{\mathbb{1}}
$$

This means that we need to add the factors τ and $\sqrt{\tau}$ to the transitions above (that correspond to the MPO). The first two cases mark transitions between the same state and can be combined into one single transition. Thus $W^I_{1,1} = \hat{\mathbb{1}} + \tau\hat{D}$. With the same principle the other elements of W^i can be derived. The transitions with the prefactors can be seen in fig. 6.3. □

The authors of the paper [36] claim that the error of $\hat{U}^I(\tau)$ with respect to $\hat{U}(\tau)$ goes with order $\mathcal{O}(\tau^2 N)$, where N is the number of lattice sites. However, careful inspection of the sum where the first errors occur $\sum_{x<y} \hat{H}_x \hat{H}_y$ shows that this holds only true, where the interaction range is small compared to the system size and if there are only interactions between two lattice sites. Thus, for the most interesting physical cases their claim should be a good approximation.

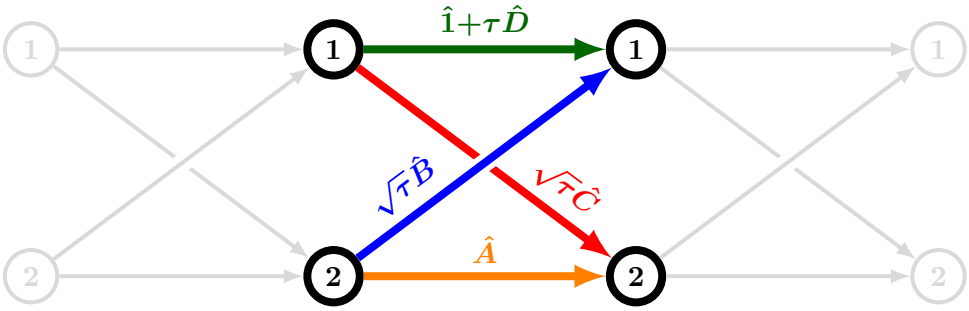

Figure 6.3: Representation of W^I as graph. See [37] on how to read such a graph.

6.3 W^{II}

There is a better approximation to $\hat{U}$ called $\hat{U}^{II}$ that eliminates some of the problems $\hat{U}^I$ has.

$$\hat{U}(\tau) \approx \hat{U}^{II}(\tau) = 1 + \tau \sum_x \hat{H}_x + \tau^2 \sum_{<x,y>} \hat{H}_x \hat{H}_y + \tau^3 \sum_{<x,y,z>} \hat{H}_x \hat{H}_y \hat{H}_z + \dots \tag{6.2}$$

Here $\sum_{<x,y,\dots>} \hat{H}_x \hat{H}_y \dots$ is defined as the sum over all terms, where no interaction term crosses a nearest neighbor bond.

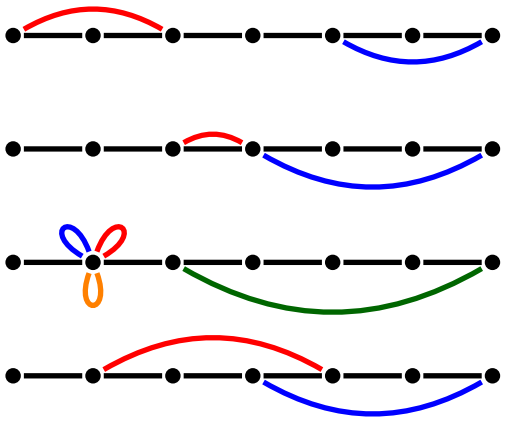

Figure 6.4: Hypothetical interactions terms of $\hat{U}(\tau)$. Only the last term is discarded for $\hat{U}^{II}(\tau)$, because there are two interaction terms that cross the fourth bond. Note that there can be an arbitrary number of on-site interactions on one lattice site. This can be seen in the third line.

To calculate the Matrix Product Operator of $\hat{U}^{II}$ we will use the multi-dimensional version of the Gaussian integral:

$$\frac{1}{\pi^N}\int e^{x\bar{\phi}-\bar{\phi}\phi+\phi y}\; d\bar{\phi}\; d\phi = e^{xy}$$

where $x, y, \phi \in \mathbb{C}^N$ and xy is the inner product. With the Gaussian integral we can find an exact representation of $\hat{U}(\tau)$ if every term of $\hat{H}$ commutes with all other terms. Because generally this is not the case we will have to do an error estimation. The strategy is to perform a split between sites i and $i+1$ into two subchains like mentioned above.

$$\begin{aligned}
e^{\tau\hat{H}} &= e^{\tau\hat{H}_{L_i}+(\sqrt{\tau}\hat{h}_{L_i})(\sqrt{\tau}\hat{h}_{R_i})+\tau\hat{H}_{R_i}} \\
&= e^{\tau\hat{H}_{L_i}} e^{(\sqrt{\tau}\hat{h}_{L_i})(\sqrt{\tau}\hat{h}_{R_i})} e^{\tau\hat{H}_{R_i}} + \mathcal{O}(\tau^2) \\
&= e^{\tau\hat{H}_{L_i}} \left(\int \mathcal{D}[\phi_i,\bar{\phi}_i] e^{\sqrt{\tau}\hat{h}_{L_i}\bar{\phi}_i} e^{-\bar{\phi}_i\phi_i} e^{\sqrt{\tau}\hat{h}_{R_i}}\right) e^{\tau\hat{H}_{R_i}} + \mathcal{O}(\tau^2)
\end{aligned}$$

where $\mathcal{D}[\phi_i,\bar{\phi}_i] = \prod_{a_i} d\bar{\phi}_{i,a_i}\; d\phi_{i,a_i}\pi^{-1}$. We used this relationship to merge and split the exponential functions:

$$e^{\tau_1 x+\tau_2 y} = e^{\tau_1 x} e^{\tau_2 y} + \mathcal{O}\left(\tau_1\tau_2[x,y]\right)$$

$$e^{\tau\hat{H}} = \int \mathcal{D}[\phi_i,\bar{\phi}_i] e^{\tau\hat{H}_{L_i}+\sqrt{\tau}\hat{h}_{L_i}\bar{\phi}_i}\; e^{-\bar{\phi}_i\phi_i}\; e^{\sqrt{\tau}\hat{h}_{R_i}+\tau\hat{H}_{R_i}} + \mathcal{O}\left(\tau^2\right) \tag{6.3}$$

Note that there is also an error term $\mathcal{O}\left(\tau^{\frac{3}{2}}\phi_i\right)$ in the equation above that vanishes due to the integral.
With the recursion relation laid out above (eq. (6.1)) we can move the the border of the subsystems to the right.

$$\begin{aligned}
\hat{H}_{R,i} &= \hat{H}_{R,i+1} + \hat{C}_{[i+1]}\hat{h}_{R,i+1} + \hat{D}_{[i+1]} \\
\hat{h}_{R,i} &= \hat{A}_{[i+1]}\hat{h}_{R,i+1} + \hat{B}_{[i+1]} \\
\sqrt{\tau}\phi_i\hat{h}_{R,i} + \tau\hat{H}_{R,i} &= \sqrt{\tau}\phi_i\left(\hat{A}_{[i+1]}\hat{h}_{R,i+1} + \hat{B}_{[i+1]}\right) + \tau\left(\hat{H}_{R,i+1} + \hat{C}_{[i+1]}\hat{h}_{R,i+1} + \hat{D}_{[i+1]}\right)
\end{aligned}$$

With this recursion we can replace the last factor of the integral in eq. (6.3) by:

$$\begin{aligned}
e^{\sqrt{\tau}\phi_i\hat{h}_{R,i}+\tau\hat{H}_{R,i}} &= e^{\sqrt{\tau}\phi_i\left(\hat{A}_{[i+1]}\hat{h}_{R,i+1}+\hat{B}_{[i+1]}\right)+\tau\left(\hat{H}_{R,i+1}+\hat{C}_{[i+1]}\hat{h}_{R,i+1}+\hat{D}_{[i+1]}\right)} \\
&= e^{\tau\hat{D}_{[i+1]}+\sqrt{\tau}\phi_i\hat{B}_{[i+1]}+\left(\phi_i\hat{A}_{[i+1]}+\sqrt{\tau}\hat{C}_{[i+1]}\right)\sqrt{\tau}\hat{h}_{R,i+1}+\tau\hat{H}_{R,i+1}} \\
&= e^{\tau\hat{D}_{[i+1]}+\sqrt{\tau}\phi_i\hat{B}_{[i+1]}} e^{\left(\phi_i\hat{A}_{[i+1]}+\sqrt{\tau}\hat{C}_{[i+1]}\right)\sqrt{\tau}\hat{h}_{R,i+1}} e^{\tau\hat{H}_{R,i+1}} + \mathcal{O}\left(\tau^2\right) \\
&= \int \mathcal{D}[\phi_{i+1},\bar{\phi}_{i+1}]\hat{U}_{\phi_i,\bar{\phi}_{i+1}} e^{-\bar{\phi}_{i+1}\phi_{i+1}} e^{\sqrt{\tau}\phi_{i+1}\hat{h}_{R,i+1}+\tau\hat{H}_{R,i+1}} + \mathcal{O}\left(\tau^2\right)
\end{aligned}$$

where $\hat{U}_{\phi_i,\bar{\phi}_{i+1}} = e^{\tau\hat{D}_{[i+1]}+\sqrt{\tau}\phi_i\hat{B}_{[i+1]}+\phi_i\hat{A}_{[i+1]}\bar{\phi}_{i+1}+\sqrt{\tau}\hat{C}_{[i+1]}\bar{\phi}_{i+1}}$. We used the multidimensional Gaussian integral in the last line of the equation above. Note that there should be an error of $\mathcal{O}\left(\tau^{\frac{3}{2}}\phi_i\right)$, but this gets integrated out again. We can repeat this process on every lattice site and get our next intermediate result:

$$e^{\tau\hat{H}} = \int \mathcal{D}[\phi,\bar{\phi}]\left[\ldots e^{-\bar{\phi}_i\phi_i}\hat{U}_{\phi_i,\bar{\phi}_{i+1}}e^{-\bar{\phi}_{i+1}\phi_{i+1}}\hat{U}_{\phi_{i+1},\bar{\phi}_{i+2}}e^{-\bar{\phi}_{i+2}\phi_{i+2}}\ldots\right] + \mathcal{O}\left(\tau^2\right) \tag{6.4}$$

For the next step we rewrite every $\hat{U}_{\phi_i,\bar{\phi}_{i+1}}$ as a Taylor series.

$$\hat{U}_{\phi_i,\bar{\phi}_{i+1}} = \sum_{\{n_i\},\{\bar{n}_{i+1}\}} \hat{U}_{n_i,\bar{n}_{i+1}} \frac{\phi_i^{n_i}\bar{\phi}_{i+1}^{\bar{n}_{i+1}}}{\sqrt{|n_i!\,||\bar{n}_{i+1}!\,|}}$$

where $|n_i!\,| = \prod_{a_i}(n_{i,a_i}!\,)$. Note that the integral in eq. (6.4) now transforms into a sum:

$$\int \mathcal{D}[\phi_i,\bar{\phi}_i]\frac{\phi_i^{n_i}\bar{\phi}_i^{\bar{n}_i}}{\sqrt{|n_i!\,||\bar{n}_i!\,|}}e^{-\bar{\phi}_i\phi_i} = \delta_{n_i,\bar{n}_i} \tag{6.5}$$

Now we get our final result for the time evolution operator:

$$e^{\tau\hat{H}} = \sum_{\{n\}}\left[\ldots\hat{U}_{n_i,n_{i+1}}\hat{U}_{n_{i+1},n_{i+2}}\hat{U}_{n_{i+2},n_{i+3}}\ldots\right] + \mathcal{O}\left(\tau^2\right) \tag{6.6}$$

To get the matrix operator representation of $\hat{U}^{II}(\tau)$ (eq. (6.2)), we need to eliminate some of the terms in eq. (6.6).

Definition 6.2: W^{II}

$$\begin{aligned}
W^{II}_{[1]}(\tau) &= \begin{pmatrix} W^{II}_D & W^{II}_C \end{pmatrix}_{[1]} \\
W^{II}_{[i]}(\tau) &= \begin{pmatrix} W^{II}_D & W^{II}_C \\ W^{II}_B & W^{II}_A \end{pmatrix}_{[i]} \quad \forall i : 1 < i < N \\
W^{II}_{[N]}(\tau) &= \begin{pmatrix} W^{II}_D \\ W^{II}_B \end{pmatrix}_{[N]}
\end{aligned}$$

Here $W^{II}_{A,[i]}$, $W^{II}_{B,[i]}$, $W^{II}_{C,[i]}$, and $W^{II}_{D,[i]}$ are defined in terms of a Taylor expansion of $\hat{U}_{\phi_{i-1},\bar{\phi}_i}$:

$$\hat{U}_{\phi_{i-1},\bar{\phi}_i} = W^{II}_{D,[i]} + \phi_{i-1}W^{II}_{B,[i]} + W^{II}_{C,[i]}\bar{\phi}_i + \phi_{i-1}W^{II}_{A,[i]}\bar{\phi}_i + \mathcal{O}(\phi^2_{i-1,1}\ldots\bar{\phi}^2_{i,1}\ldots)$$

Theorem 6.2

W^{II} is a possible Matrix Product Operator form of $\hat{U}^{II}$ within $\mathcal{O}(\tau^3)$.

Proof. Several interactions crossing the same bond appear in eq. (6.4) as higher orders of ϕ_i and $\bar{\phi}_i$. Thus eliminating higher orders of $\hat{U}_{\phi_{i-1},\bar{\phi}_i}$ gets rid of those interactions. Furthermore we need to proof that the Taylor expansion of $\hat{U}_{\phi_{i-1},\bar{\phi}_i}$ yields the same product of operators as W^{II}. To do that we inspect one part of the integral eq. (6.4).

$$
\begin{aligned}
&\ldots \int \mathcal{D}[\phi_i, \bar{\phi}_i] \hat{U}_{\phi_{i-1},\bar{\phi}_i} e^{-\bar{\phi}_i \phi_i} \hat{U}_{\phi_i,\bar{\phi}_{i+1}} \ldots \\
&\approx \ldots \int \mathcal{D}[\phi_i, \bar{\phi}_i] \left(W^{II}_{D,[i]} + \phi_{i-1} W^{II}_{B,[i]} + W^{II}_{C,[i]} \bar{\phi}_i + \phi_{i-1} W^{II}_{A,[i]} \bar{\phi}_i \right) e^{-\bar{\phi}_i \phi_i} \\
&\qquad \left(W^{II}_{D,[i+1]} + \phi_i W^{II}_{B,[i+1]} + W^{II}_{C,[i+1]} \bar{\phi}_{i+1} + \phi_i W^{II}_{A,[i+1]} \bar{\phi}_{i+1} \right) \ldots \\
&= \ldots W^{II}_{D,[i]} W^{II}_{D,[i+1]} + W^{II}_{D,[i]} W^{II}_{C,[i+1]} \bar{\phi}_{i+1} + \phi_{i-1} W^{II}_{B,[i]} W^{II}_{D,[i+1]} \\
&\qquad + \phi_{i-1} W^{II}_{B,[i]} W^{II}_{C,[i+1]} \bar{\phi}_{i+1} + W^{II}_{C,[i]} W^{II}_{B,[i+1]} + W^{II}_{C,[i]} W^{II}_{A,[i+1]} \bar{\phi}_{i+1} \\
&\qquad + \phi_{i-1} W^{II}_{A,[i]} W^{II}_{B,[i+1]} + \phi_{i-1} W^{II}_{A,[i]} W^{II}_{A,[i+1]} \bar{\phi}_{i+1} \ldots
\end{aligned}
$$

Which has in fact all the terms of the matrix product $W^{II}_{[i]} W^{II}_{[i+1]}$. From the first to the second line we used the Taylor expansion of $\hat{U}_{\phi_{i-1},\bar{\phi}_i}$ and from the second to the last line we used eq. (6.5). For the fact that this is only exact up to $\mathcal{O}(\tau^3)$, we refer to [36]. □

Note that W^{II} is only the correct representation of $\hat{U}^{II}$ up to $\mathcal{O}(\tau^3)$. This does not matter, because the error of $\hat{U}^{II}$ with respect to the real time evolution operator is only correct up to $\mathcal{O}(\tau^2)$. To explicitly calculate W^{II} and eliminate the higher orders of ϕ and $\bar{\phi}$ there is a convenient mathematical trick. We can redefine the ϕ_{i,a_i} to be operators instead of scalars with the property $\phi^2_{i,a_i} = 0$. Likewise for $\bar{\phi}_{i,a_i}$. This automatically eliminates higher orders. We can use the well known boson creation and annihilation operator for this with a maximum occupation number of 1.

Definition 6.3: Auxiliary boson creation and annihilation operators

$$
c^\dagger = \begin{pmatrix} c^\dagger_1 & c^\dagger_2 & \ldots & c^\dagger_{N_i} \end{pmatrix} \qquad \bar{c}^\dagger = \begin{pmatrix} \bar{c}^\dagger_1 \\ \bar{c}^\dagger_2 \\ \ldots \\ \bar{c}^\dagger_{N_i} \end{pmatrix}
$$

Where the $c^\dagger_j$ are represented in matrix notation as:

$$c_j^\dagger = \begin{matrix} \\ \langle 0| \\ \langle 1| \end{matrix} \begin{matrix} \begin{matrix} |0\rangle & |1\rangle \end{matrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{matrix} \qquad \bar{c}_j^\dagger = \begin{matrix} \\ \langle \bar{0}| \\ \langle \bar{1}| \end{matrix} \begin{matrix} \begin{matrix} |\bar{0}\rangle & |\bar{1}\rangle \end{matrix} \\ \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} \end{matrix}$$

and $c_j = (c_j^\dagger)^\dagger$.
The vacuum state of this auxiliary bosons will be denoted as $|0\rangle$ *for* $c^\dagger$ *and* $|\bar{0}\rangle$ *for* $\bar{c}^\dagger$

Note that the $c_j^\dagger$ span their own auxiliary space that is independent of the physical Hilbert space. We can use the properties of the creation and annihilation operator to numerically calculate the matrix elements of W^{II}.

Algorithm 6.1: Computation of W^{II}

$$\hat{U}_{c^\dagger,\bar{c}^\dagger,[i]} = e^{\tau \hat{D}_{[i]} \,+\, \sqrt{\tau} c^\dagger \otimes \hat{B}_{[i]} \,+\, c^\dagger \otimes \hat{A}_{[i]} \otimes \bar{c}^\dagger \,+\, \sqrt{\tau} \hat{C}_{[i]} \otimes \bar{c}^\dagger}$$

$$\begin{aligned} W^{II}_{A,[i],j,k} &= \langle 0\bar{0}|\, c_j \bar{c}_k \hat{U}_{c^\dagger,\bar{c}^\dagger,[i]} \,|0\bar{0}\rangle \\ W^{II}_{B,[i],j,1} &= \langle 0\bar{0}|\, c_j \hat{U}_{c^\dagger,\bar{c}^\dagger,[i]} \,|0\bar{0}\rangle \\ W^{II}_{C,[i],1,k} &= \langle 0\bar{0}|\, \bar{c}_k \hat{U}_{c^\dagger,\bar{c}^\dagger,[i]} \,|0\bar{0}\rangle \\ W^{II}_{D,[i]} &= \langle 0\bar{0}|\, \hat{U}_{c^\dagger,\bar{c}^\dagger,[i]} \,|0\bar{0}\rangle = e^{\tau \hat{D}_{[i]}} \end{aligned}$$

Note that $\hat{U}_{c^\dagger,\bar{c}^\dagger,[i]} \in \mathcal{H}_c \otimes \mathcal{H}_{\bar{c}} \otimes \mathcal{H}_{\text{phys}}$, where $\mathcal{H}_c$ and $\mathcal{H}_{\bar{c}}$ are the auxiliary spaces of the bosons, and $\mathcal{H}_{\text{phys}}$ is the physical operator space. $W^{II}_{A,[i],j,k}$ is entry j,k of W^{II}_A on site i.

6.4 Second Order

In the previous subsections we showed the error of $\hat{U}^I(\tau)$ and $\hat{U}^{II}(\tau)$ goes with $\mathcal{O}(\tau^2)$. There is an easy way to lower the error by one order with two time evolutions with different time steps τ_1 and τ_2.

$$\begin{aligned} \hat{U}^{I/II}(\tau) &= 1 + \tau \hat{H} + \tau^2 \sum_{x<y} \hat{H}_x \hat{H}_y + \mathcal{O}(\tau^3) \\ \hat{U}^{I/II}(\tau_1)\hat{U}^{I/II}(\tau_2) &= 1 + (\tau_1 + \tau_2)\,\hat{H} + \tau_1\tau_2 \hat{H}^2 + \left(\tau_1^2 + \tau_2^2\right) \sum_{x<y} \hat{H}_x \hat{H}_y + \mathcal{O}(\tau^3) \\ &\stackrel{!}{=} 1 + \tau \hat{H} + \frac{1}{2}\tau^2 \hat{H}^2 + \mathcal{O}(\tau^3) \end{aligned}$$

From this we get equations for τ_1 and τ_2:

$$\begin{aligned}\tau_1 + \tau_2 &= \tau \\ \tau_1\tau_2 &= \frac{1}{2}\tau^2 \\ \tau_1^2 + \tau_2^2 &= 0\end{aligned}$$

These equations have solutions for $\tau_1 = \frac{1}{2}(1-i)\tau$ and $\tau_2 = \frac{1}{2}(1+i)\tau$.

Proof.

$$\begin{aligned}\tau_1 + \tau_2 &= \frac{1}{2}\tau\left(1 - i + 1 + i\right) = \tau \\ \tau_1\tau_2 &= \frac{1}{4}\tau^2\left((1-i)(1+i)\right) = \frac{1}{2}\tau^2 \\ \tau_1^2 + \tau_2^2 &= \frac{1}{4}\tau^2\left((1-i)^2 + (1+i)^2\right) = \frac{1}{4}\tau^2\left((1-2i-1) + (1+2i-1)\right) = 0\end{aligned}$$

□

Note that we apply $\hat{U}^{I/II}(\tau_1)$ and $\hat{U}^{I/II}(\tau_2)$ one after another and there is a possible error due to the finite bond dimension of our Matrix Product State $|\psi\rangle$. According to [36] this does not proof to be a problem.

Algorithm 6.2: Second order time evolution with Matrix Product Operators

We want to calculate the time evolution of $|\psi(\tau_0)\rangle$ with $\hat{U}^I$ or $\hat{U}^{II}$ with a time step of τ.

1. $$\tau_1 = \frac{1}{2}(1-i)\tau \qquad \tau_2 = \frac{1}{2}(1-i)\tau$$

2. $$|\psi(\tau_1 + \tau_0)\rangle = \hat{U}^{I/II}(\tau_1)\,|\psi(\tau_0)\rangle$$

3. $$|\psi(\tau + \tau_0)\rangle = \hat{U}^{I/II}(\tau_2)\,|\psi(\tau_1 + \tau_0)\rangle$$

7 Matrix Product Operators of Fermionic Systems

If we want to want to calculate the ground state or time evolution of a fermionic system we must take the fermionic anticommutator relations (def. 1.2) into account. One must choose a convention for the order how the creation operators get applied to the vacuum state to represent the Fock space. Every choice is equally valid as long as we stay consistent while doing our computation. For the entire thesis the following convention is assumed:

Definition 7.1: Order convention of the fermionic Fock Space

We consider a fermionic system with N lattice sites. Let $|\psi\rangle$ be an arbitrary state of the Fock space and let $n_{i,s}$ be the occupation number of site i and spin s. The order convention in this thesis is:

$$|\psi\rangle = \left(c^\dagger_{1,\uparrow}\right)^{n_{1,\uparrow}} \left(c^\dagger_{1,\downarrow}\right)^{n_{1,\downarrow}} \dots \left(c^\dagger_{N,\uparrow}\right)^{n_{N,\uparrow}} \left(c^\dagger_{N,\downarrow}\right)^{n_{N,\downarrow}} |0\rangle$$

The problem regarding Matrix Product Operators is in a very strict sense operators like $c_{i,s}$ are not local to one lattice site, because we may get a different sign stemming from the anticommutator relations.

$$\begin{aligned} c_{i,\uparrow}|\psi\rangle &= c_{i,\uparrow} \left(c^\dagger_{1,\uparrow}\right)^{n_{1,\uparrow}} \left(c^\dagger_{1,\downarrow}\right)^{n_{1,\downarrow}} \dots \left(c^\dagger_{i,\uparrow}\right)^{n_{i,\uparrow}} \dots \left(c^\dagger_{N,\uparrow}\right)^{n_{N,\uparrow}} \left(c^\dagger_{N,\downarrow}\right)^{n_{N,\downarrow}} |0\rangle \\ &= \delta_{n_{i,\uparrow},1}(-1)^\kappa \left(c^\dagger_{1,\uparrow}\right)^{n_{1,\uparrow}} \left(c^\dagger_{1,\downarrow}\right)^{n_{1,\downarrow}} \dots \left(c^\dagger_{i,\uparrow}\right)^{0} \dots \left(c^\dagger_{N,\uparrow}\right)^{n_{N,\uparrow}} \left(c^\dagger_{N,\downarrow}\right)^{n_{N,\downarrow}} |0\rangle \end{aligned}$$

where $\kappa = \sum_{m<i}(n_{m,\uparrow} + n_{m,\downarrow})$. The idea is to find an operator that calculates the product $(-1)^{n_{1,\uparrow}+n_{1,\downarrow}}(-1)^{n_{2,\uparrow}+n_{2,\downarrow}} \dots$ for us. The resulting operator for one lattice site is called parity operator and reads as:

Definition 7.2: Parity operator

$$P_i = \begin{array}{c} \\ \langle 0| \\ \langle 0|c_\uparrow \\ \langle 0|c_\downarrow \\ \langle 0|c_\downarrow c_\uparrow \end{array} \begin{array}{c} \begin{array}{cccc} |0\rangle & c^\dagger_\uparrow|0\rangle & c^\dagger_\downarrow|0\rangle & c^\dagger_\uparrow c^\dagger_\downarrow|0\rangle \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}_i \end{array}$$

Note that $P_i^2 = \mathbb{1}_i$. With this definition we can transform the creation and annihilation operators into:

$$c^{(\dagger)}_{i,s} \quad \rightarrow \quad P_1 P_2 \dots P_{i-1} c^{(\dagger)}_{i,s}$$

This transforms products of creation and annihilation operators into:

$$\begin{aligned}\forall i<j: \quad c^\dagger_{i,s}c_{j,s'} \quad &\rightarrow \quad c^\dagger_{i,s}P_iP_{i+1}\dots P_{j-1}c_{j,s'}\\ \forall i>j: \quad c^\dagger_{i,s}c_{j,s'} \quad &\rightarrow \quad -c_{j,s'}P_jP_{j+1}\dots P_{i-1}c^\dagger_{j,s}\end{aligned}$$

These kind of transformations are known as Wigner-Seitz transformation. With that knowledge we can find a Matrix Product Operator representation of e.g. the Hubbard model:

Example 7.1: Long range Hubbard model in MPO notation

$$\hat{H} = -\sum_{\substack{i<j\\ s}} e^{-\alpha(j-i)}\left(c^\dagger_{is}c_{js} - c_{is}c^\dagger_{js}\right) + U\sum_i n_{i\uparrow}n_{i\downarrow}$$

$$W^{s_1,s_1'} = \begin{pmatrix}1 & -e^{-\alpha}c^\dagger_\uparrow P & -e^{-\alpha}c^\dagger_\downarrow P & e^{-\alpha}c_\uparrow P & e^{-\alpha}c_\downarrow P & Un_\uparrow n_\downarrow\end{pmatrix}$$

$$W^{s_i,s_i'} = \begin{pmatrix}1 & -e^{-\alpha}c^\dagger_\uparrow P & -e^{-\alpha}c^\dagger_\downarrow P & e^{-\alpha}c_\uparrow P & e^{-\alpha}c_\downarrow P & Un_\uparrow n_\downarrow\\ 0 & e^{-\alpha}P & 0 & 0 & 0 & c_\uparrow\\ 0 & 0 & e^{-\alpha}P & 0 & 0 & c_\downarrow\\ 0 & 0 & 0 & e^{-\alpha}P & 0 & c^\dagger_\uparrow\\ 0 & 0 & 0 & 0 & e^{-\alpha}P & c^\dagger_\downarrow\\ 0 & 0 & 0 & 0 & 0 & 1\end{pmatrix} \quad \forall 1<i<N$$

$$W^{s_N,s_N'} = \begin{pmatrix}Un_\uparrow n_\downarrow\\ c_\uparrow\\ c_\downarrow\\ c^\dagger_\uparrow\\ c^\dagger_\downarrow\\ 1\end{pmatrix}$$

Infinite TDVP is not based on the Matrix Product Operator formalism, but we still need to do the same adjustments with the parity operator. If we want to use it to calculate time evolutions all of the derivations used above are still valid. A fermionic Hamiltonian (def. 5.1) of TDVP reads after a Wigner-Seitz transformation as:

$$\hat{H} = \sum_i \hat{h}^1_i + \sum_i \tilde{\hat{h}}^2_{i,i+1} + \sum_{m,n,j>i} \hat{M}^m_i P_i\dots P_{j-1}\hat{N}^m_j\beta_n e^{-\alpha_n(j-i)}$$

Where $\tilde{\hat{h}}^2_{i,i+1}$ is the Wigner-Seitz transformation of $\hat{h}^2_{i,i+1}$. The only other step that is modified in regard to the original algorithm is the calculation of the geometric series that stems from the summation over the infinite pairs i and j.

$$(1-e^{-\alpha_n T})^{-1} \quad \rightarrow \quad (1-e^{-\alpha_n \tilde{T}})^{-1}$$

where $\tilde{T} = \sum_s P^{s,s} \left(A^s \otimes \bar{A}^s\right)$. Furthermore, we need to multiply every $\hat{M}_i^m$ with P from the right: $\tilde{\hat{M}}_i^m = \hat{M}_i^m P$.

8 Results: Ground State Search in the Thermodynamic Limit

As mentioned above, the original formulation of the Time-Dependent Variational Principle on uniform infinite lattices [27] was formulated to work only with nearest neighbor interactions and in the course of this thesis this was extended to compute the time evolution of Hamiltonians with exponentially decaying long range interactions. To test this new version of the algorithm an imaginary time evolution of the Haldane-Shastry model (def. 1.5) and the long range Hubbard model (def. 1.1) was performed.

Haldane-Shastry model

$$\hat{H} = \sum_{j>i} |j-i|^{-2} \left(\frac{1}{2} \left(\hat{S}_i^+ \hat{S}_j^- + \hat{S}_i^- \hat{S}_j^+ \right) + \hat{S}_i^z \hat{S}_j^z \right)$$

Note that the interaction of the Haldane-Shastry model follows a power law and it is not possible to use the matrix product formalism in that case. To circumvent this problem the power law was approximated with a sum of ten exponentials [25].

$$r^{-2} \approx \sum_i \beta_i e^{-\alpha_i r}$$

In fig. 8.1 the difference between the power law and the sum of exponentials can be seen.

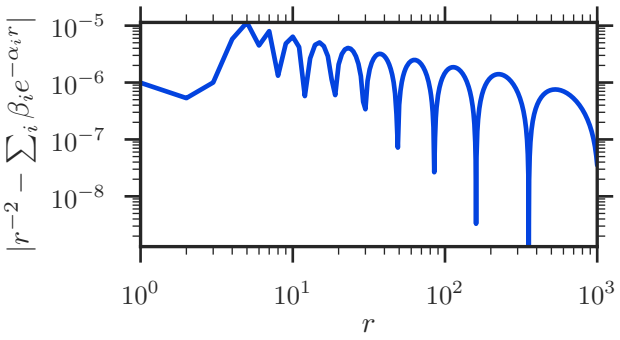

Figure 8.1: Absolute difference between r^{-2} and $\sum_i \beta_i e^{-\alpha_i r}$.

The Haldane-Shastry model is really useful here, because in 1993 F. D. M. Haldane and M. R. Zirnbauer [24] found a closed form solution for correlations of the form $\langle GS|\hat{S}_n^a(t)\hat{S}_{n+\Delta n}^b(t')|GS\rangle$, where $a, b \in \{x, y, z\}$ and $|GS\rangle$ is the ground state of the system in the thermodynamic limit.

$$\langle GS|\hat{S}_n^a(t)\hat{S}_{n+\Delta n}^b(t')|GS\rangle = \delta_{a,b}\frac{1}{16}(-1)^{\Delta n}\int_{-1}^{1} d\lambda_1 \int_{-1}^{1} d\lambda_2 e^{i\pi\lambda_1\lambda_2\Delta n - \frac{1}{4}\pi^2(t-t')(\lambda_1-\lambda_2)^2}$$

The integral of the solution was performed with the trapezoid rule with 30000 nodes and $t = t' = 0$. The results are:

$$\begin{aligned}\langle GS|\hat{S}^z_n\hat{S}^z_{n+1}|GS\rangle &= -0.14737\ldots\\ \langle GS|\hat{S}^z_n\hat{S}^z_{n+2}|GS\rangle &= 0.05642\ldots\\ \langle GS|\hat{S}^z_n\hat{S}^z_{n+3}|GS\rangle &= -0.04442\ldots\end{aligned}$$

The imaginary time evolution was done with a random initial state with a bond dimension of $\chi = 32$ and was carried out twice with two different time steps $\Delta\tau$. The results of the time evolution can be seen in fig. 8.2, where

$$\epsilon = |\langle\psi(\tau)|\hat{S}^z_n\hat{S}^z_{n+\Delta n}|\psi(\tau)\rangle - \langle GS|\hat{S}^z_n\hat{S}^z_{n+\Delta n}|GS\rangle|$$

It shows that the error goes to zero as the imaginary time τ goes to infinity, except some constant error term that is expected due to the finite bond dimension χ and the approximation of the power law with exponential functions. This holds true for both time step sizes and shows that it is possible to use TDVP in the thermodynamic limit with long range interactions.

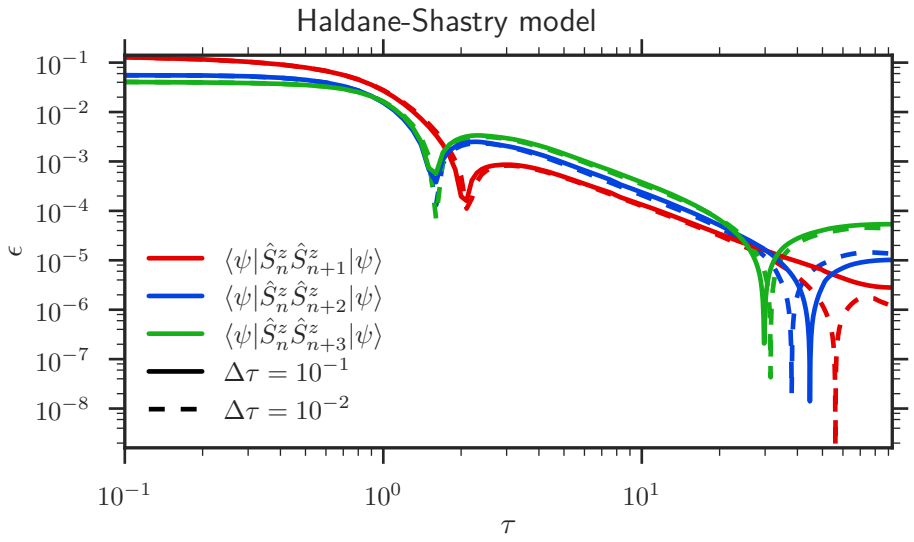

Figure 8.2: Error as a function of imaginary time.

Long Range Hubbard model

$$\hat{H} = -\sum_{\substack{i<j \\ s}} e^{-\alpha(j-i)} \left(c_{is}^{\dagger} c_{js} - c_{is} c_{js}^{\dagger} \right) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}$$

For the long range Hubbard model no analytic solution is known and iDMRG did not converge. In fig. 8.3 the calculated expectation values as a function of imaginary time τ can be seen. Some non-rigorous tests on finite lattice showed that the calculated expectation values in the thermodynamic limit are close to the expectation values on small finite lattices. In fig. 8.3 can also be seen that the expectation values of $\hat{n}$, $\hat{n}_{\uparrow}$, and $\hat{n}_{\downarrow}$ clearly converge.

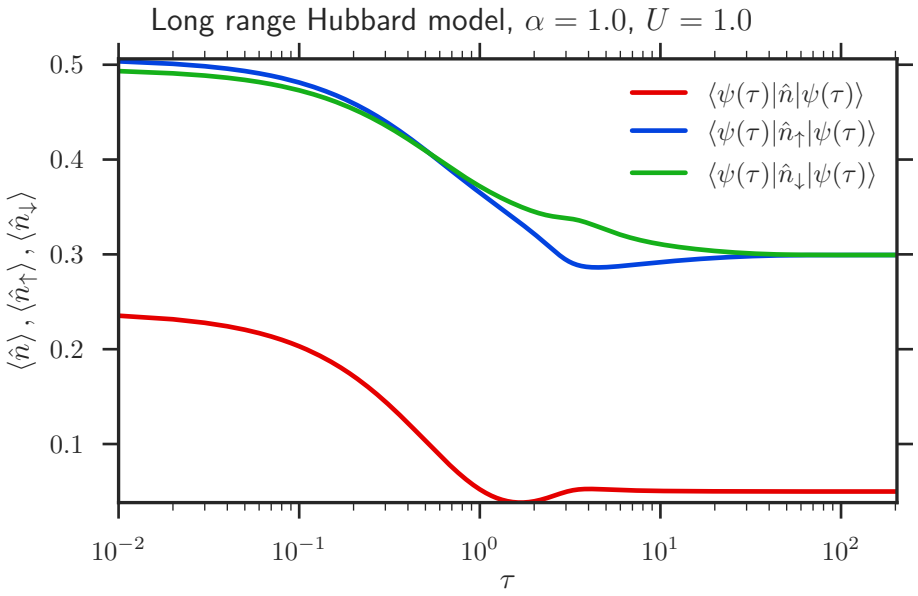

Figure 8.3: Expectation values as a function of imaginary time.

9 Results: Long Range Transverse Field Ising Model

In this section the Time-Dependent Variational Principle (TDVP) is compared to Matrix Product Operator based time evolution (W^I and W^{II}). In this section the one site version of TDVP will be called **TDVP1**, while the two site version will be called **TDVP2**. The two operator based methods will be dubbed **W1** and **W2**. Both algorithms have a computational cost of $\mathcal{O}(\chi_S^3 \chi_O)$, where χ_S is the bond dimension of the Matrix Product State and χ_O is the bond dimension of the Matrix Product Operator (appendix A.4). W1 and W2 were used in second order. All computations in this section were done with julia 0.4.2. [38] and were performed on a Toshiba Satellite U500 [39] (2 cores with 2Ghz clock rate). At first, a system with nine lattice sites was investigated. This was done to compare the resulting states of the different algorithms to a state calculated via total diagonalization. Next, a system with 41 lattice sites was simulated to see how the algorithms perform for bigger system sizes. As reference model the long range transverse field Ising model (def. 1.4) was used:

$$\hat{H} = -\sum_{i<j} e^{-\alpha|j-i|} \hat{S}_i^x \hat{S}_j^x - h \sum_i \hat{S}_i^z$$

In this section the magnetic field will always be equal to $h = 0.45$, and the parameter α will be set to either $\alpha = 0.1$ (long range) or $\alpha = 1.0$ (short range). The parameter α indicates how fast the interactions between the different lattice sites decay with respect to distance and a higher value of α corresponds to a faster decaying interaction. In fig. 9.1 a comparison between those two interactions can be seen.

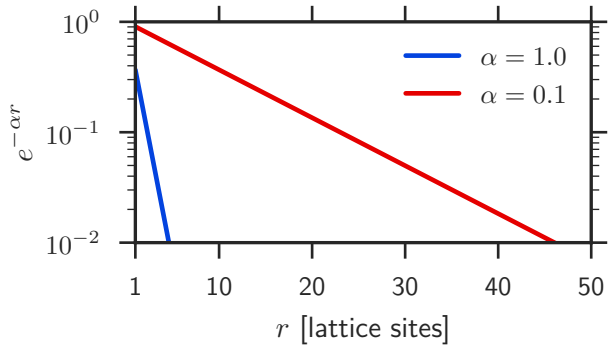

Figure 9.1: Comparison between the interactions of the lattice sites for $\alpha = 1.0$, and $\alpha = 0.1$.

Furthermore, we need a way to measure the error of a state that was numerically calculated with respect to the analytic solution or a good approximation of the analytic solution (called reference state). To do this, the overlap of the state and the reference state, and the difference in expectation values were used:

Definition 9.1: Error measures

Let $|\psi\rangle$ be the state that was calculated with a numerical method of our choice and let $|\phi\rangle$ be a reference state. As reference state we will either use the analytical solution or a numerical solution that has a negligible deviance to the analytically solution in comparison to $|\psi\rangle$. The two error measures used in this section are defined as:

$$\epsilon_1 = 1 - |\langle\phi|\psi\rangle|$$
$$\epsilon_2 = \sum_i |\langle\phi|\hat{S}_i^x|\phi\rangle - \langle\psi|\hat{S}_i^x|\psi\rangle|$$

For every α and system size the following initial state was used:

$$|\psi(t=0)\rangle = |\uparrow\uparrow \ldots \uparrow\downarrow\uparrow \ldots \uparrow\uparrow\rangle_x$$

Based on this initial state an approximation for $|\psi(t=1)\rangle$ was calculated with the different algorithms. These simulations were repeated for different time steps Δt to get $\epsilon_{1/2}$ as a function of Δt.

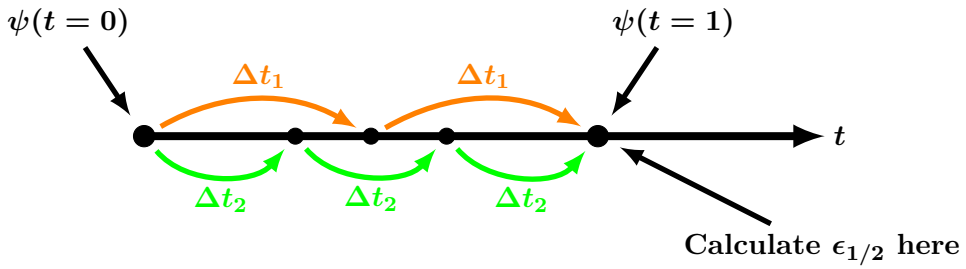

Figure 9.2: Illustration of the performed simulations. The orange and green lines stand for time evolutions done with one of the two algorithms.

After every computation of $|\psi(t=1)\rangle$ the elapsed real time was saved. In the following figures this quantity will be denoted with the name wall clock. This size is very important in practical terms. One algorithm may perform better than the other at the same time step Δt, but may need much more time to compute the result. Because of this, one wants to take a look at the error as a function of computation time. It can be seen down below that this is indeed the case with TDVP1/2 and W1/2. As a rule of thumb TDVP1/2 performs better at the same time step Δt as W1/2, but the simulation takes more time to complete. The question is, if Δt is lowered for W1/W2 so that it produces the same error as TDVP1/2, will the former be faster than the latter?

N = 9 Lattice Sites

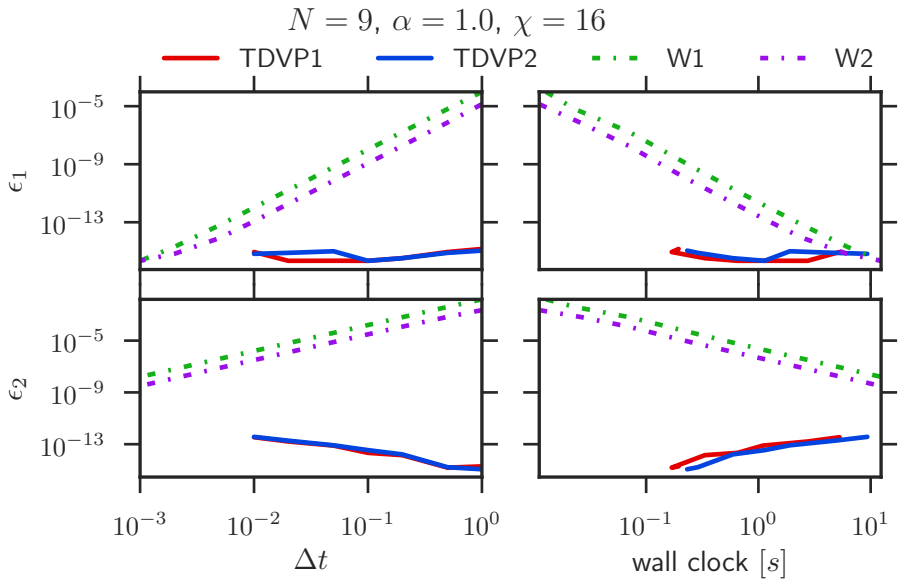

Figure 9.3: Error of $|\psi(t=1)\rangle$ for a system with nine lattice sites and $\alpha = 1.0$ (short range). The reference state was calculated via total diagonalization.

In figs. 9.3 and 9.4 the results for a system with nine lattice sites can be seen. The bond dimension $\chi = 16$ was chosen, because for nine lattice sites the Matrix Product State can map to the whole Hilbert space of 2^9 basis states. It can be seen that TDVP1 and TDVP2 are basically down to machine precision at every time step Δt. An oddity is that ϵ_2 increases with a lower time step Δt for TDVP1/2. This seems to be somehow linked to the Matrix Product State formalism. It can be seen that the overlap error ϵ_1 is in many cases down to machine precision, while the expectation value error ϵ_2 is still orders of magnitudes higher. If we had an analytic solution of the state and two states had an overlap of one, every operator must have the same expectation value. Thus, it must be a numerical problem.

$$\forall \psi, \phi, \hat{O}: \quad \langle\psi|\psi\rangle = \langle\phi|\phi\rangle = \langle\psi|\phi\rangle = 1 \quad \Rightarrow \quad \langle\psi|\hat{O}|\psi\rangle = \langle\phi|\hat{O}|\phi\rangle$$

In fig. 9.4 it can also be seen that the error of W1/2 is almost constant for $\alpha = 0.1$ and large time steps. This effect can be seen better on larger systems and seems to increase with system size.

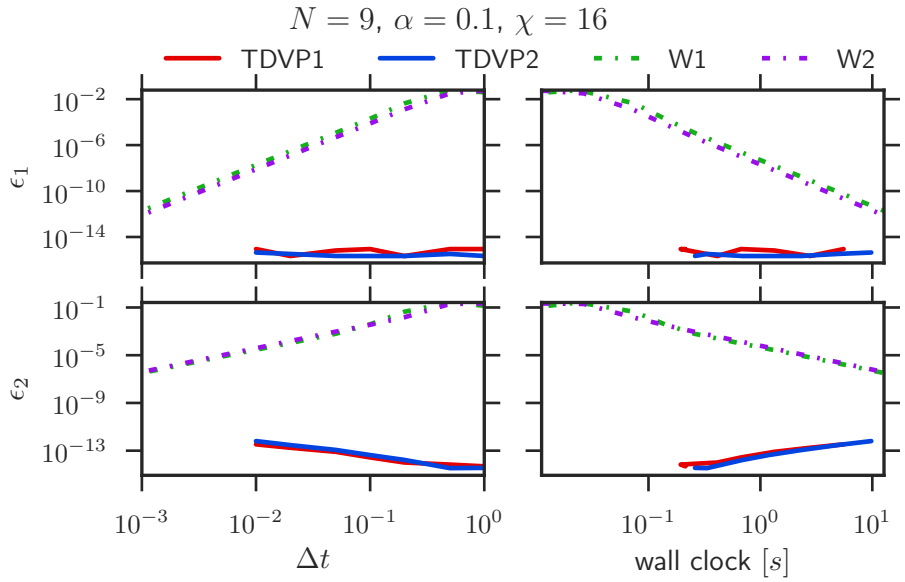

Figure 9.4: Error of $|\psi(t = 1)\rangle$ for a system with nine lattice sites and $\alpha = 0.1$ (long range). The reference state was calculated via total diagonalization.

N = 41 Lattice Sites, time reversal

For a system with 41 lattice sites it is not possible to calculate the time evolution of the system via total diagonalization due to the sheer size of the state space ($2^{41} \approx 2 \cdot 10^{12}$). Thus, the first simulations on this large system were done under time reversal.

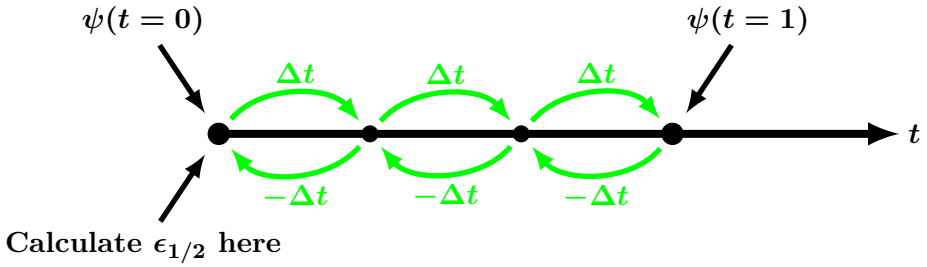

Figure 9.5: Illustration of the performed simulations under time reversal. With all of the algorithms $|\psi(t=1)\rangle$ was computed and then back evolved to $t = 0$.

The initial state was time evolved to $t = 1$ and then back evolved to $t = 0$. An illustration of this process can be seen in fig. 9.5. As reference state the original initial state was used to calculate $\epsilon_{1/2}$.

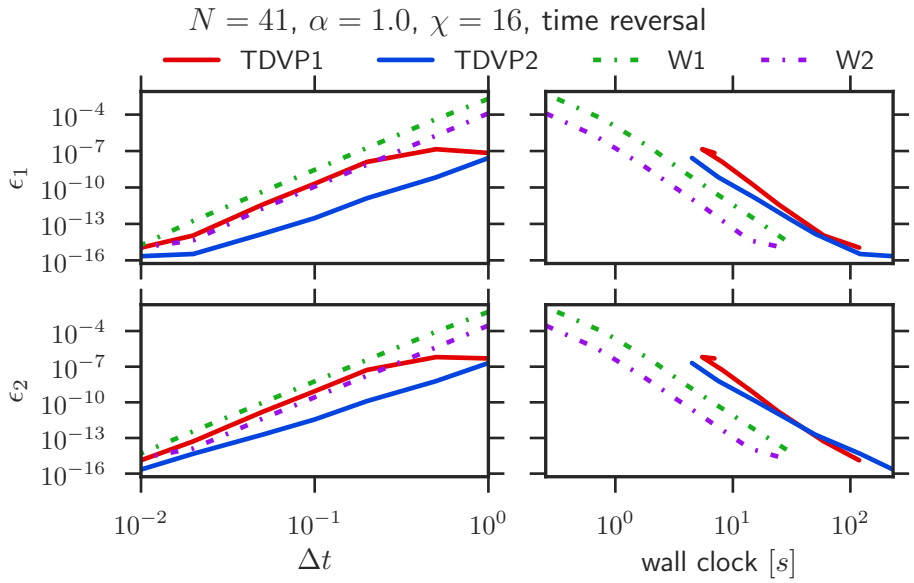

Figure 9.6: Error of the algorithms for a system with 41 lattice sites and $\alpha = 1.0$ (short range) under time reversal.

The results of this computation can be seen in figs. 9.6 and 9.7. What we can see again in the case of $\alpha = 0.1$ that there is a growing range of time steps, where the error of W1/2

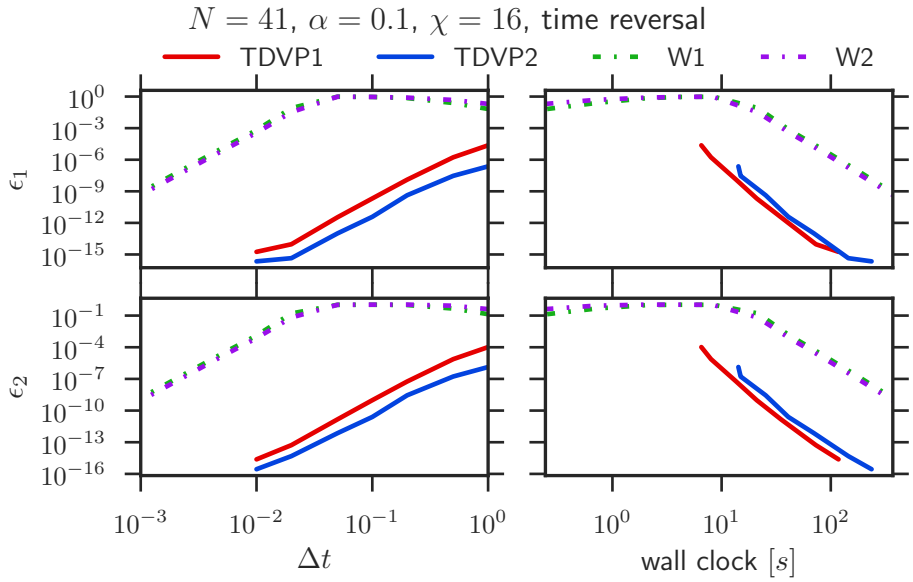

Figure 9.7: Error of the algorithms for a system with 41 lattice sites and $\alpha = 0.1$ (long range) under time reversal.

is almost constant. Furthermore, we see that in the case of time reversal and $\alpha = 1.0$ W1/2 is much faster than TDVP1/2 with the same error. This is because some of the errors seem to cancel themselves, when we do a time evolution back in time. Nevertheless we see that $\epsilon_{1/2}$ as a function of the time step Δt is the smallest for TDVP2. For that reason TDVP2 was chosen to calculate the reference state for the next subsection.

N = 41 Lattice Sites, Small Bond Dimension

In figs. 9.8 and 9.9 the results for a system with 41 lattice sites can be seen. The bond dimension was capped at $\chi = 4$ to see how the algorithms perform, when the error is dominated by a low bond dimension. It can be seen best in fig. 9.9 ($\alpha = 0.1$) that TDVP1/2 reaches the lowest possible ϵ at far smaller time steps Δt and at the same computation time TDVP1/2 has an error orders of magnitudes lower than the error of W1/2. In the case of $\alpha = 1.0$, where the interactions decay much faster with respect to distance, it seems that the algorithms are not well separated anymore with respect to computation time. Especially, it is notable that the lines of TDVP2 and W2 for ϵ_1 as a function of computation time lie almost on top of each other.

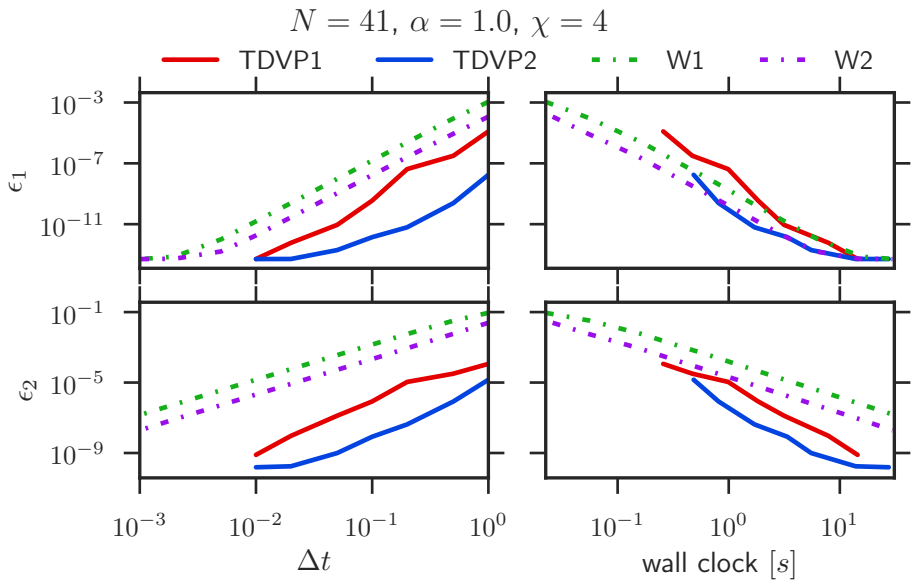

Figure 9.8: Error of $|\psi(t = 1)\rangle$ for a system with 41 lattice sites and $\alpha = 1.0$ (short range). The reference state was calculated with TDVP2. Note the small bond dimension.

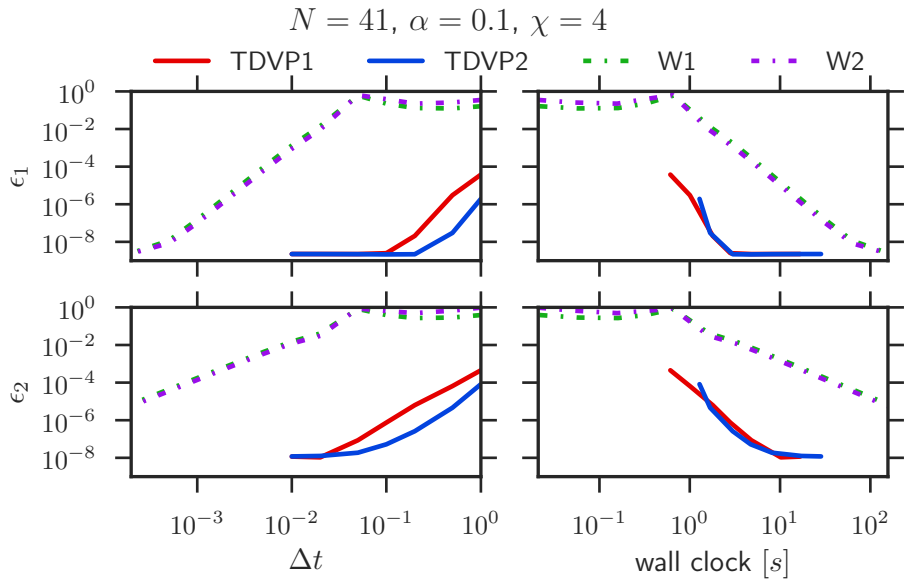

Figure 9.9: Error of $|\psi(t = 1)\rangle$ for a system with 41 lattice sites and $\alpha = 0.1$ (long range). The reference state was calculated with TDVP2. Note the small bond dimension.

N = 41 Lattice Sites

In figs. 9.10 and 9.11 the results for a system with 41 lattice sites and a fixed bond dimension of $\chi = 16$ can be seen. It can clearly be seen that $\epsilon_{1/2}$ as a function of Δt has a steeper slope for TDVP1/2 than W1/2. The measured slope can be seen below. We have almost the same results as in the case of $\chi = 4$. For shorter ranges ($\alpha = 1.0$) TDVP2 and W2 lie almost on top of each other for ϵ_1 as a function of the elapsed real time. For longer effective ranges ($\alpha = 0.1$) TDVP1/2 produces an error orders of magnitudes lower than the error of W1/2 with the same computation time. It can also be seen that W1 and W2 produce a vastly increased error with a longer effective interaction range (α decreases). This behavior is not really surprising, because those two algorithms discard multiple interactions crossing the same bond within the higher powers $\hat{H}$ in the Taylor expansion of $e^{-it\hat{H}}$. As α gets decreased those interactions get more important relative to the on-site potential. It can not be seen very well, but the same happens to TDVP1 and TDVP2. This probably stems from the finite bond dimension χ. The reference states were computed with a maximum bond dimension of $\chi = 64$. While for $\alpha = 1.0$ a bond dimension of $\chi = 18$ was enough, for $\alpha = 0.1$ the maximum $\chi = 64$ was not big enough and in that case the discarded weight was not equal to zero. For the simulations in figs. 9.10 and 9.11 a fixed bond dimension of $\chi = 16$ was used. This explains the increasing error of TDVP1 and TDVP2 with decreasing α.

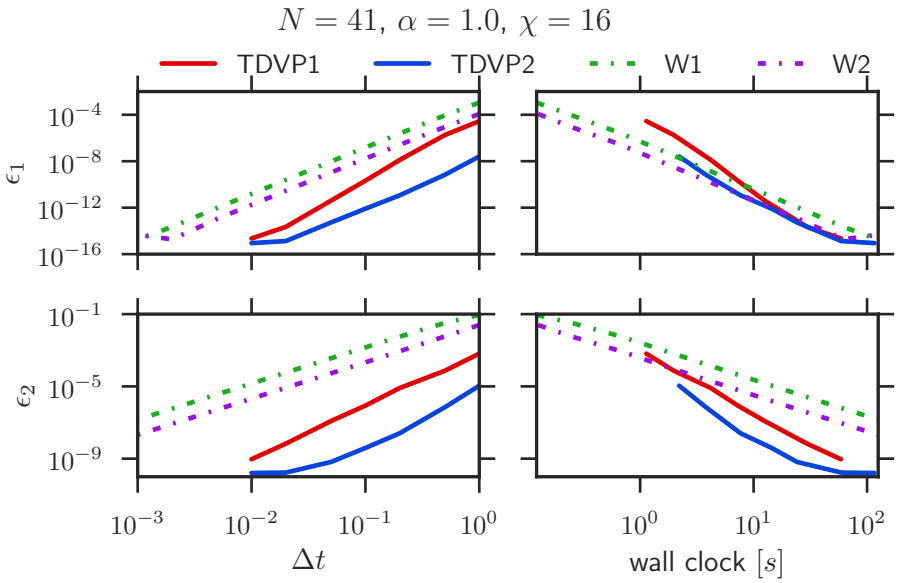

Figure 9.10: Error of $|\psi(t = 1)\rangle$ for a system with 41 lattice sites and $\alpha = 1.0$ (short range). The reference state was calculated with TDVP2.

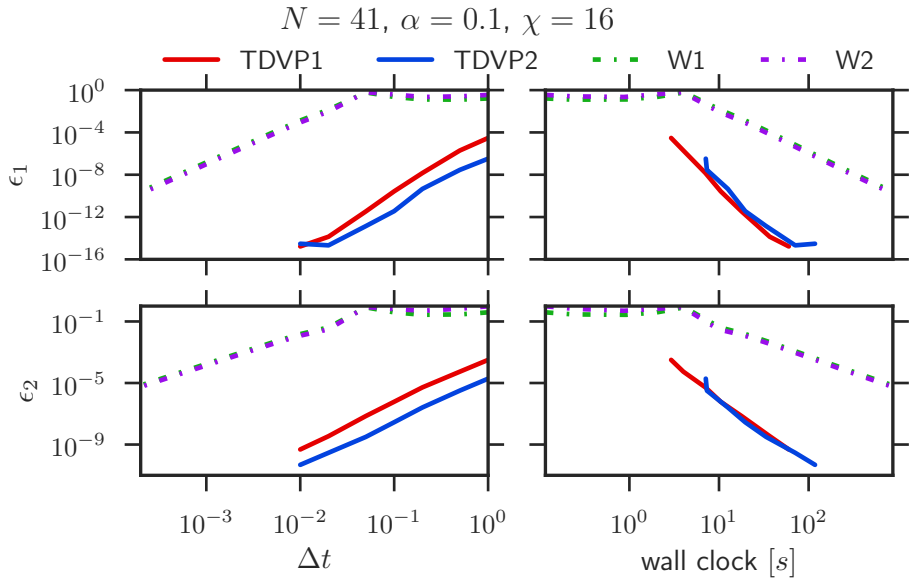

Figure 9.11: Error of $|\psi(t = 1)\rangle$ for a system with 41 lattice sites and $\alpha = 0.1$ (long range). The reference state was calculated with TDVP2.

Fits

For the case of $N = 41$ and $\chi = 16$ a fit for the graphs was performed for ϵ_1 and ϵ_2. The fits were done in the ranges, where $\epsilon_{1/2}$ as a function of the time step or the elapsed real time (wall clock) is approximately linear in the log-log plot. This implies a power function:

$$\log y = \tilde{a} + b \cdot \log x = \tilde{a} + \log x^b \quad \Rightarrow \quad y = e^{\tilde{a} + \log x^b} = e^{\tilde{a}} \cdot e^{\log x^b} = a \cdot x^b$$

The data points that apparently deviated from the linear behavior were simply discarded. These fits may provide unreliable information - especially for TDVP1/2 - because the graphs are not linear enough (for a prime example see ϵ_2 as a function of Δt for TDVP2 in fig. 9.10). The data can be seen in tables 9.1 and 9.2 and the responding plots can be seen in appendix A.5. In all cases TDVP1 and TDVP2 have a better behavior than W1 and W2 for small Δt. From the tables we can gather the same information, we already saw in the figures above. All algorithms perform worse with decreasing α. Note that the procedure that performs the exponential of the large matrix was custom build and an average time of 90% was spent in this routine douring the simulations. Thus, TDVP may be sped up significantly and the results may be skewed.

Table 9.1: Fits for the system with 41 lattice sites and $\chi = 16$, where $\epsilon_{1/2}$ was fitted as a function of the time step Δt. The graphs on the log-log plot are not always straight lines and the fits may be unreliable, so no uncertainty is provided here.

Algorithm	α	$\epsilon_1 = a \cdot t^b$		$\epsilon_2 = a \cdot t^b$	
		a	b	a	b
TDVP1	1.0	1.8e−4	5.9	1.4e−3	3.2
TDVP2	1.0	1.5e−8	4.4	1.1e−5	3.9
W1	1.0	1.5e−3	4.0	1.4e−1	2.0
W2	1.0	1.5e−4	3.9	2.3e−2	2.0
TDVP1	0.1	1.5e−4	5.7	2.9e−3	3.1
TDVP2	0.1	6.5e−7	4.5	1.1e−5	3.2
W1	0.1	1.5e+5	4.0	2.6e−3	2.0
W2	0.1	9.7e+4	4.0	3.8e−4	2.0

Table 9.2: Fits for the system with 41 lattice sites and $\chi = 16$, where $\epsilon_{1/2}$ was fitted as a function of the elapsed real time t_w. The graphs on the log-log plot are not always straight lines and the fits may be unreliable, so no uncertainty is provided here.

Algorithm	α	$\epsilon_1 = a \cdot t_w^b$		$\epsilon_2 = a \cdot t_w^b$	
		a	b	a	b
TDVP1	1.0	6.4e−4	−7.5	2.9e−3	−4.1
TDVP2	1.0	1.6e−6	−5.8	6.0e−4	−5.1
W1	1.0	5.0e−7	−4.1	2.6e−3	−2.0
W2	1.0	5.2e−8	−4.1	3.8e−4	−2.1
TDVP1	0.1	5.5	−10.1	2.2e−1	−5.4
TDVP2	0.1	1.3e−1	−7.7	7.1e−2	−4.9
W1	0.1	3.8e+2	−4.2	8.4	−2.1
W2	0.1	2.1e+2	−4.2	6.3	−2.1

N = 41 Lattice Sites, longer time

In this subsection are the results for a time evolution to $t = 15$ computed with TDVP2 and a time step of $\Delta t = 0.1$ and a fixed bond dimension of $\chi = 32$. One can clearly see in fig. 9.12 that for the shorter effective interaction range ($\alpha = 1.0$) the spin flip term due to the magnetic field in z-direction dominates and the lattice gets flipped almost simultaneously on all lattice sites, while for longer effective interaction range ($\alpha = 0.1$, fig. 9.13) the spin-spin interaction dominates. Note that the behavior seems to be qualitatively vastly different in these two cases, because in the case of $\alpha = 0.1$ the system never performs a global spin flip in the simulated time frame and the expectation value $\langle S_x \rangle$ seems to stay near the initial values. This may stem from a too short simulation time or some kind of "phase transition". Either way, further research in this area should proof to be interesting.

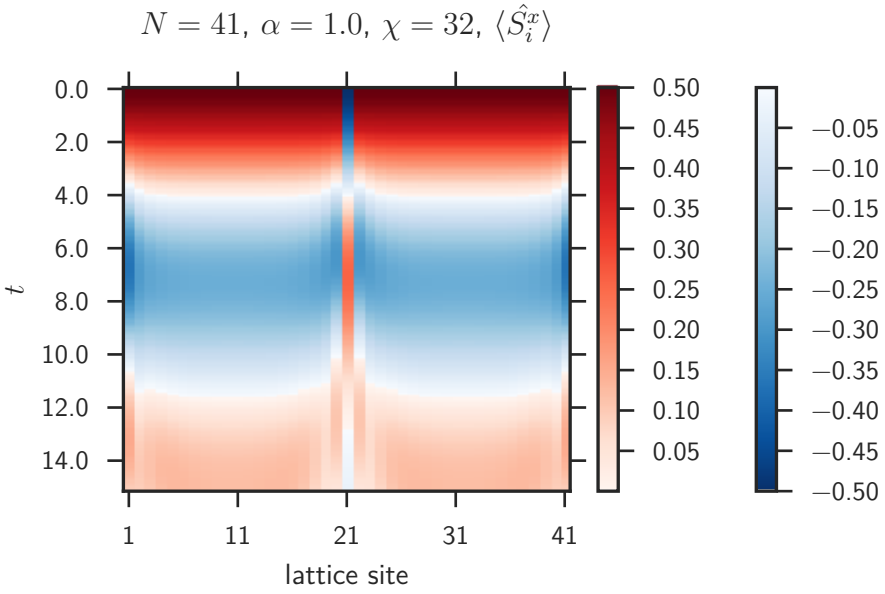

Figure 9.12: $\langle \hat{S}_i^x \rangle$ as a function of time for the initial state $|\uparrow\uparrow \ldots \uparrow\downarrow\uparrow \ldots \uparrow\uparrow\rangle_x$ and $\alpha = 1.0$ (short range).

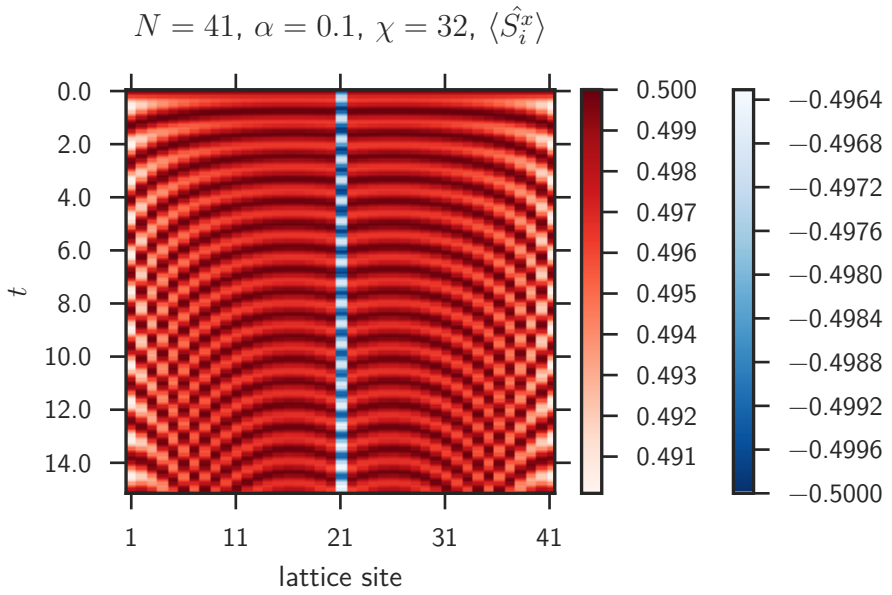

Figure 9.13: $\langle \hat{S}_i^x \rangle$ as a function of time for the initial state $|\uparrow\uparrow \ldots \uparrow\downarrow\uparrow \ldots \uparrow\uparrow\rangle_x$ and $\alpha = 0.1$ (long range).

10 Conclusions

In the course of this thesis, the two algorithms introduced by Zaletel et al. [36] (MPO based) and Haegeman et. al. [35] (TDVP) on a modified version of the transverse field Ising model with exponentially decaying long range interactions were compared. The main result of the simulation is contained in figs. 9.10 and 9.11. They show that the MPO based method is a viable alternative to TDVP, if the interactions of the lattice sites decay fast with respect to distance. On the other hand, if the interactions decay slowly, TDVP should be used and produces an error orders of magnitudes lower. In tables 9.1 and 9.2 it can be seen that TDVP has in general a much better asymptotic behavior with respect to the used time step.

The computational bottleneck of Finite lattice TDVP is to calculate an exponential of a matrix with a very high number of elements. Further research in this area could additionally speed up TDVP significantly.

Furthermore, the Infinite lattice TDVP [27] was extended in the present thesis, such that it can also work with exponentially decaying long range interactions. In section 8 it can be seen that this method can be used to calculate the ground state of long-ranged models with imaginary time evolution.

A Appendix

A.1 Calculation of the geometric series of the transfer matrix

In this subsection we will calculate the geometric series of the transfer matrix T. We assume that the transfer matrix has exactly one dominant eigenvalue with value 1, where $|r\rangle$ is the corresponding right hand side eigenvector and $\langle l|$ is the left hand side eigenvector. Furthermore it was assumed that $\langle l|r\rangle = 1$. The projector Q was defined as:

$$Q = 1 - |r\rangle \langle l|$$

This operator has some helpful properties. As a projection operator it is idempotent:

$$\begin{aligned} Q^2 &= ((1 - |r\rangle \langle l|) ((1 - |r\rangle \langle l|) \\ &= 1 - |r\rangle \langle l| - |r\rangle \langle l| + |r\rangle \underbrace{\langle l|r\rangle}_{=1} \langle l| \\ &= 1 - |r\rangle \langle l| \\ &= Q \end{aligned}$$

Furthermore, Q commutes with the transfer matrix:

$$\begin{aligned} [Q, T] &= ((1 - |r\rangle \langle l|) T - T ((1 - |r\rangle \langle l|) \\ &= T - |r\rangle \langle l| T - T + T |r\rangle \langle l| \\ &= T - T + |r\rangle \langle l| - |r\rangle \langle l| \\ &= 0 \end{aligned}$$

The powers $T = QTQ + |r\rangle \langle l|$ have a plain and simple form:

$$\begin{aligned} T^2 &= (QTQ + |r\rangle \langle l|) (QTQ + |r\rangle \langle l|) \\ &= (QTQ)^2 + |r\rangle \langle l|r\rangle \langle l| + QT \underbrace{Q |r\rangle}_{=0} \langle l| + |r\rangle \underbrace{\langle l| Q}_{=0} TQ \\ &= (QTQ)^2 + |r\rangle \langle l| \\ &\dots \\ T^n &= (QTQ)^n + |r\rangle \langle l| \end{aligned}$$

Because QTQ has only eigenvalues with an absolute value < 1, it is possbile to calculate the geometric series for it. Now we have everything we need to calculate the geometric series of the transfer matrix.

$$\begin{aligned}\sum_{n=0}^{\infty} T^n &= \sum_{n=0}^{\infty} (QTQ + |r\rangle\langle l|)^n \\ &= \sum_{n=0}^{\infty} \left((QTQ)^n + |r\rangle\langle l|\right) \\ &= Q\left(\sum_{n=0}^{\infty}(QTQ)^n\right)Q + \sum_{n=0}^{\infty}|r\rangle\langle l| \\ &= Q\left(1 - QTQ\right)^{-1}Q + \sum_{n=0}^{\infty}|r\rangle\langle l|\end{aligned}$$

This proofs the claim from the main text. In the third line of the equation above we used $Q^2 = Q$ and $[T, Q] = 0$.

A.2 Linearly independent parameters of B(x)

This section follows [27].
How many linearly independent parameters does B have? In general a tangent state $|\phi(A, B)\rangle$ will have a non trivial null space. This stems from the gauge invariance of $|\psi(A)\rangle$ with respect to A. If we look at an auxiliary state defined as

$$\tilde{A}^s = e^{\epsilon X} A^s e^{-\epsilon X}$$

where $\epsilon \in \mathbb{R}$ and $X \in \mathbb{C}^{\chi\times\chi}$. Because $e^{\epsilon X}e^{-\epsilon X} = 1$, $\tilde{A}$ and A describe the same state (see section 2). Note that $X \in \mathbb{C}^{\chi\times\chi}$. Thus, the state is independent of ϵ and its derivative with respect to ϵ must vanish.

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\epsilon}\tilde{A}^s &= X\tilde{A}^s - \tilde{A}^s X \\ \frac{\mathrm{d}}{\mathrm{d}\epsilon}|\psi(\tilde{A})\rangle &= 0 \\ \frac{\mathrm{d}\tilde{A}^j}{\mathrm{d}\epsilon}|\partial_j\psi(\tilde{A})\rangle &= 0\end{aligned}$$

The last line of the equation above is of the same form of the derivative of $|\psi(A)\rangle$ with respect to time! With that obervation we know that every $B^s_X = XA^s - A^sX$ produces a zero norm state. So, every $B' = B + B_X$ will result in the same tangent state as B. This eliminates χ^2 degrees of freedom from the general $d\chi^2$ parameters of B. Thus, a appropriate parametrization of B will have $(d-1)\chi^2$ linearly indepentent parameters.

A.3 Calculation of K_{OS}, K_{NN}, K_l^n, K_r^n, and K_{LR}

This section follows [27].

In general calculating the these vectors directly is too expensive computational wise. However, by plugging in the definitions it is possible to find matrix equations that can be solved.

$$\begin{aligned}
\langle K_{OS}| &= \langle l|\, h_A^A Q(1-QTQ)^{-1}Q \\
\langle K_{OS}|\,(1-QTQ) &= \langle l|\, h_A^A Q \\
\langle K_{OS}|\,(1-T+|r\rangle\langle l|) &= \langle l|\, h_A^A(1-|r\rangle\langle l|) \\
\langle K_{OS}| - \langle K_{OS}|\,T + \langle K_{OS}|r\rangle\langle l| &= \langle l|\, h_A^A - \langle l|\, h_A^A\,|r\rangle\langle l|
\end{aligned}$$

From the first to the second line we used $[T, Q] = 0$. Now we can rewrite this with our usual rules into a matrix equation:

$$K_{OS} - \sum_s A^{s\dagger} K_{OS} A^s + \mathrm{tr}(K_{OS} r)\; l = \sum_{s,t} \langle s|\,\hat{h}^1\,|t\rangle \left[A^{s\dagger} l A^t - \mathrm{tr}(A^{s\dagger} l A^t r) l\right] \tag{A.1}$$

The last line can be solved like a matrix equaitn $Ax = b$. Most modern computer algebra systems are able to this with a complexity like $\mathcal{O}(\chi^3)$. Likewise for the next neighbor term K_{NN} we need to solve the equation:

$$\begin{aligned}
K_{NN} - \sum_s A^{s\dagger} K_{NN} A^s + \mathrm{tr}(K_{NN} r)\; l = \sum_{s,t,u,v} &\langle s,t|\,\hat{h}^2\,|u,v\rangle \\
&\left[A^{t\dagger} A^{s\dagger} l A^u A^v - \mathrm{tr}(A^{t\dagger} A^{s\dagger} l A^u A^v r) l\right]
\end{aligned} \tag{A.2}$$

And similar for the long range case, we need to solve:

$$K_l^n - e^{-\alpha_n} \sum_s A^{s\dagger} K_l^n A^s = e^{-\alpha_n} \sum_{s,t} \langle s|\,\hat{M}^m\,|t\rangle\, A^{s\dagger} l A^t \tag{A.3}$$

$$K_r^n - e^{-\alpha_n} \sum_s A^s K_r^n A^{s\dagger} = e^{-\alpha_n} \sum_{s,t} \langle s|\,\hat{N}^m\,|t\rangle\, A^t l A^{s\dagger} \tag{A.4}$$

$$\begin{aligned}
K_{LR} - \sum_s A^{s\dagger} K_{LR} A^s + \mathrm{tr}(K_{LR} r) l = \sum_n \beta_n \sum_{s,t} &\langle s|\,\hat{N}^m\,|t\rangle \\
&\left[A^{s\dagger} K_l^n A^t - \mathrm{tr}(A^{s\dagger} K_l^n A^t r) l\right]
\end{aligned} \tag{A.5}$$

A.4 Computational Cost of Operator Application and TDVP

In this section we are going to briefly discuss the computational cost of applying a Matrix Product Operator to a Matrix Product State and the cost of one TDVP sweep on a finite lattice. Currently the most effective algorithm to perform an MPO-MPS application is the zip-up algorithm [29]. Here, the most expensive step is performing a singular value decomposition of the combined tensor depicted in fig. A.1.

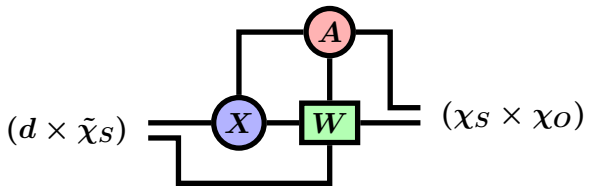

Figure A.1: Graphical representation of the tensor to perform an SVD on. A is a tensor of the Matrix Product State, W a tensor of the Matrix Product Operator, and X is a $\tilde{\chi}_S \times \chi_S \times \chi_O$ tensor.

So, we must compute the SVD of a $(d \cdot \tilde{\chi}_S) \times (\chi_S \cdot \chi_O)$ matrix, where $\tilde{\chi}_S \propto \chi_S$. We assume that the bond dimension of the state is much greater than the local Hilbert space dimension and the bond dimension of the operator. So the computational cost of this SVD and thus the MPO-MPS application is $\mathcal{O}(\chi_S^3 d^2 \chi_O)$ [40]. For TDVP the most expensive step is the computatation of the matrix exponential $e^{-i\Delta t \hat{H}_{eff}}$ (fig. 3.3). For TDVP we only need to compute the product of the exponential with a vector. This can be effectively calculated with a Lanczos scheme, where we only need to know the action of $\hat{H}_{eff}$ on an arbitrary vector v [41]. The trick here is to never explicitly build $\hat{H}_{eff}$, but to calculate the product $w = \hat{H}_{eff} v$ recursively.

$$
\begin{aligned}
\tilde{w}^1_{a_{n-1},a'_{n-1},b_n,s_n,s'_n} &= \sum_{b_{n-1}} E^{b_{n-1}}_{a_{n-1},a'_{n-1}} W^{s_n,s'_n}_{b_{n-1},b_n} \\
\tilde{w}^2_{a'_{n-1},b_n,s_n,s'_{n+1},a_{n+1}} &= \sum_{s'_n,a_{n-1}} \tilde{w}^1_{a_{n-1},a'_{n-1},b_n,s_n,s'_n} v_{s'_n,s'_{n+1},a_{n-1},a_{n+1}} \\
\tilde{w}^3_{a'_{n-1},s_n,a_{n+1},s_{n+1},b_{n+1}} &= \sum_{b_n,s'_{n+1}} \tilde{w}^2_{a'_{n-1},b_n,s_n,s'_{n+1},a_{n+1}} W^{s_{n+1},s'_{n+1}}_{b_n,b_{n+1}} \\
w_{s_n,s_{n+1},a'_{n-1},a'_{n+1}} &= \sum_{b_{n+1},a_{n+1}} \tilde{w}^3_{a'_{n-1},s_n,a_{n+1},s_{n+1},b_{n+1}} F^{b_{n+1}}_{a_{n+1},a'_{n+1}}
\end{aligned}
$$

where E and F are MPS-MPO-MPS contractions (section 3). The most expensive computations here are line 2 and 4 which cost $\mathcal{O}(\chi_S^3 d^3 \chi_O)$ for the two site TDVP and $\mathcal{O}(\chi_S^3 d^2 \chi_O)$ for the one site version of TDVP. This operation needs to be done a number of times that depends on how exact we want to evaluate $e^{-i\Delta t \hat{H}_{eff}}$.

A.5 Plots of the Fits

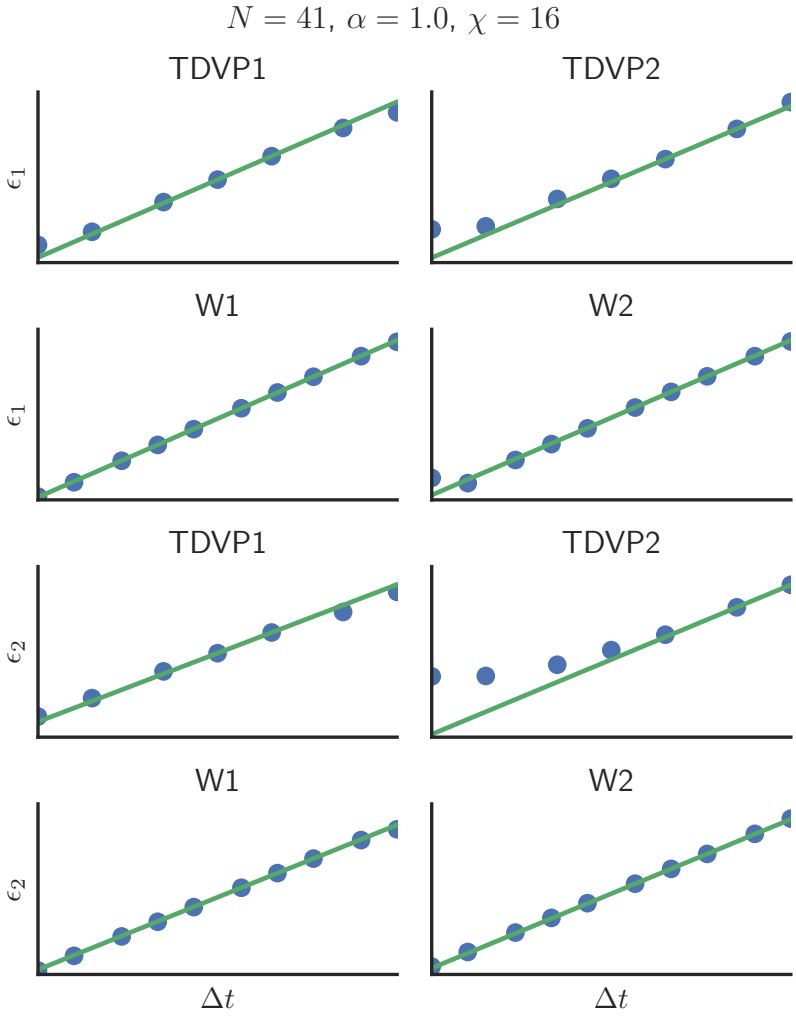

Figure A.2: Fits of the error as a function of the time step for $\alpha = 1.0$. The actual data points are the blue dots. The fits are represented by green lines. Note that the plots are not in scale with each other. (original plot: fig. 9.10; fitdata: table 9.1)

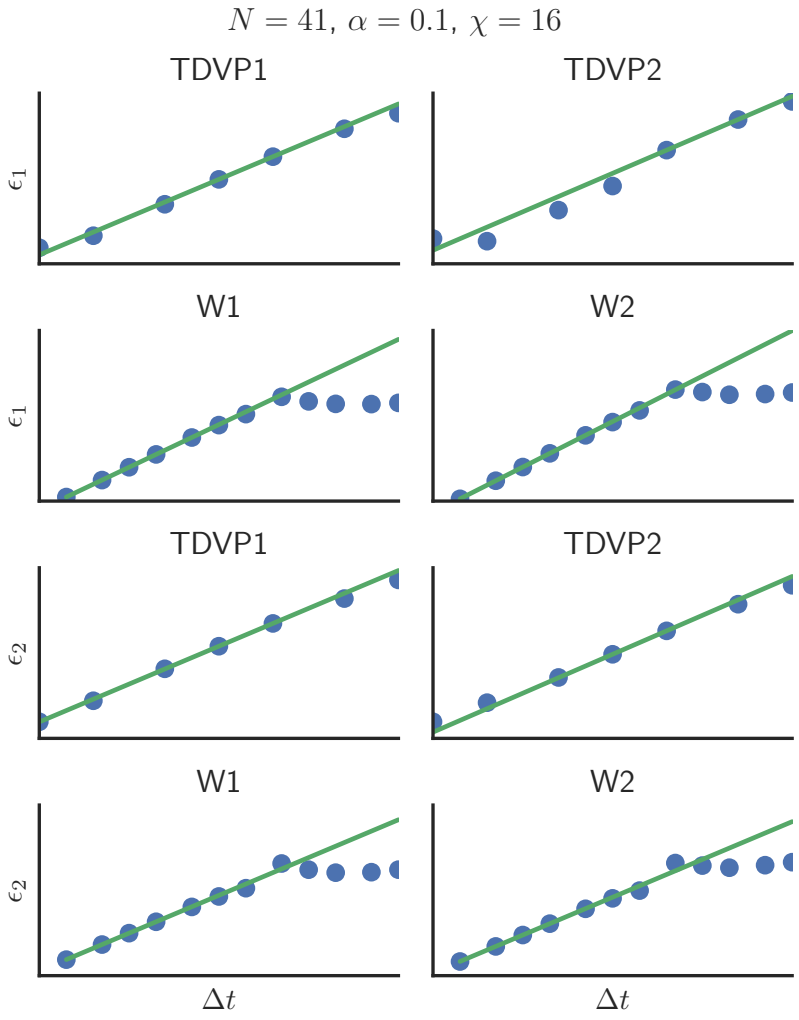

Figure A.3: Fits of the error as a function of the time step for $\alpha = 0.1$. The actual data points are the blue dots. The fits are represented by green lines. Note that the plots are not in scale with each other. (original plot: fig. 9.11; fitdata: table 9.1)

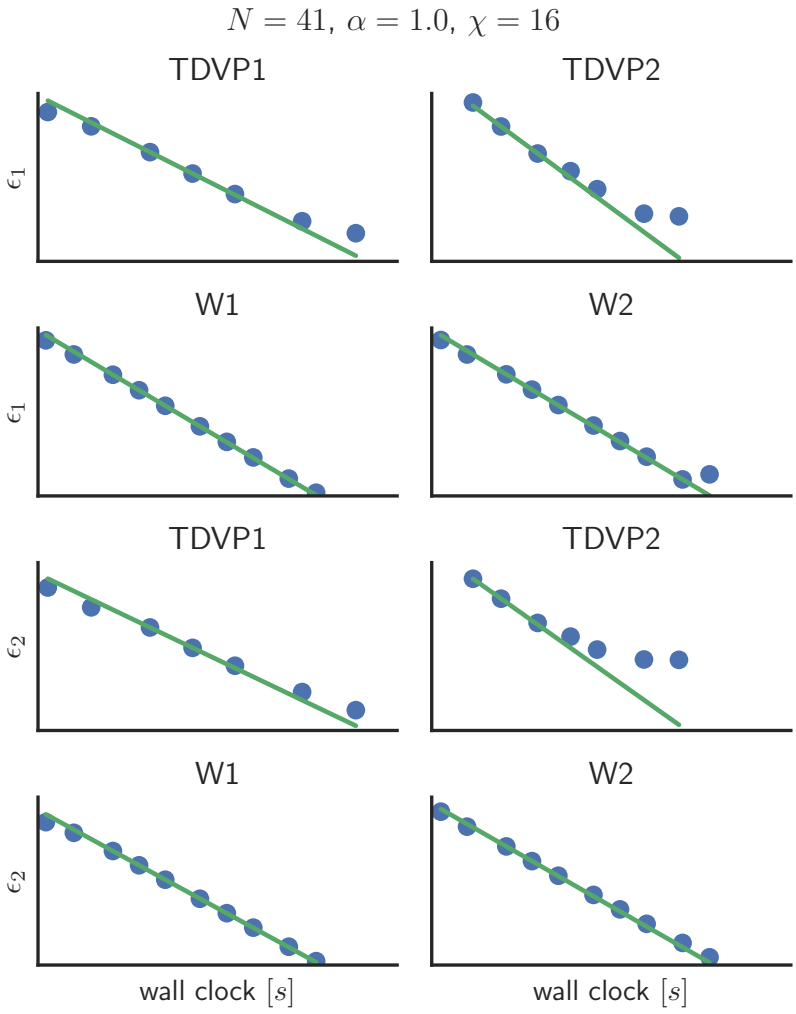

Figure A.4: Fits of the error as a function of the computation time for $\alpha = 1.0$. The actual data points are the blue dots. The fits are represented by green lines. Note that the plots are not in scale with each other. (original plot: fig. 9.10; fitdata: table 9.2)

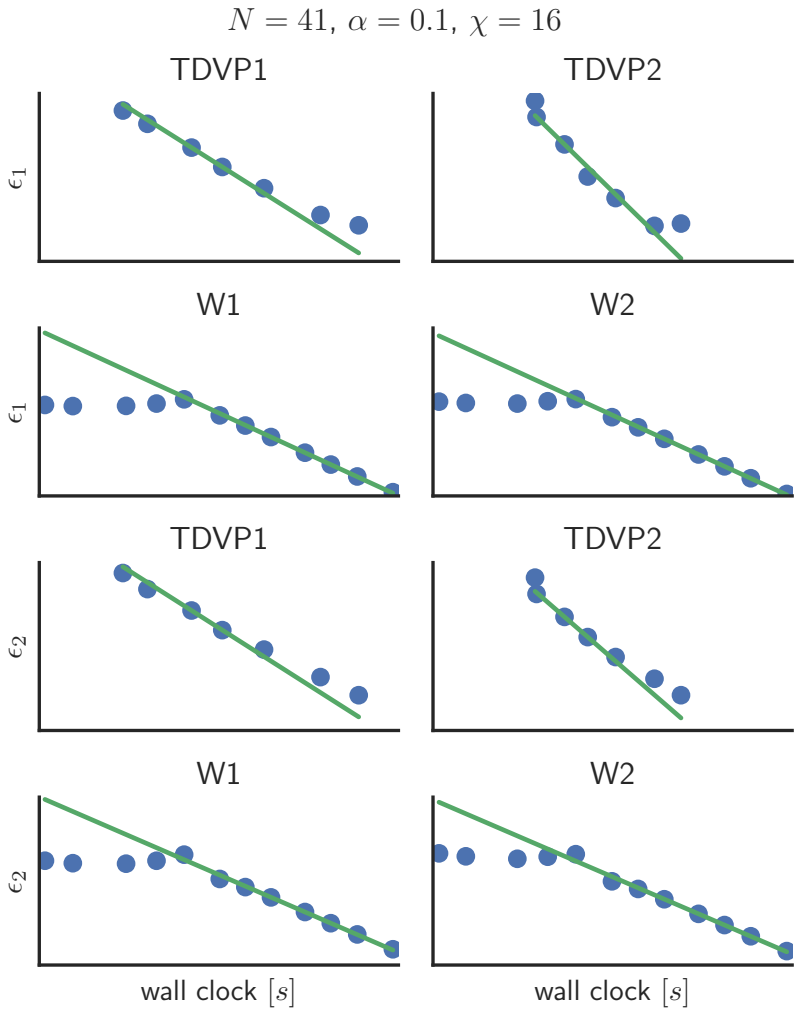

Figure A.5: Fits of the error as a function of the computation time for $\alpha = 0.1$. The actual data points are the blue dots. The fits are represented by green lines. Note that the plots are not in scale with each other. (original plot: fig. 9.11; fitdata: table 9.2)

References

[1] J. G. Bednorz and K. A. Müller, "Possible high t c superconductivity in the ba—la—cu—o system," in *Ten Years of Superconductivity: 1980–1990*, Springer, 1986, pp. 267–271.

[2] G. Binasch, P. Grünberg, F Saurenbach, and W Zinn, "Enhanced magnetoresistance in layered magnetic structures with antiferromagnetic interlayer exchange," *Physical review B*, vol. 39, no. 7, p. 4828, 1989.

[3] M. N. Baibich, J. M. Broto, A. Fert, F. N. Van Dau, F. Petroff, P Etienne, G Creuzet, A Friederich, and J Chazelas, "Giant magnetoresistance of (001) fe/(001) cr magnetic superlattices," *Physical review letters*, vol. 61, no. 21, p. 2472, 1988.

[4] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[5] E. Gull, A. J. Millis, A. I. Lichtenstein, A. N. Rubtsov, M. Troyer, and P. Werner, "Continuous-time monte carlo methods for quantum impurity models," *Reviews of Modern Physics*, vol. 83, no. 2, p. 349, 2011.

[6] M. Troyer and U.-J. Wiese, "Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations," *Physical review letters*, vol. 94, no. 17, p. 170 201, 2005.

[7] F. Barahona, "On the computational complexity of ising spin glass models," *Journal of Physics A: Mathematical and General*, vol. 15, no. 10, p. 3241, 1982.

[8] U. Schollwöck, "The density-matrix renormalization group in the age of matrix product states," *Annals of Physics*, vol. 326, no. 1, pp. 96–192, 2011.

[9] N. Schuch, I. Cirac, and F. Verstraete, "Computational difficulty of finding matrix product ground states," *Physical review letters*, vol. 100, no. 25, p. 250 501, 2008.

[10] T. Prosen and M. Žnidarič, "Matrix product simulations of non-equilibrium steady states of quantum spin chains," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2009, no. 02, P02035, 2009.

[11] C. Kittel, *Introduction to solid state physics*. Wiley, 2005.

[12] P. Fulde, P. Thalmeier, and G. Zwicknagl, "Strongly correlated electrons," *arXiv preprint cond-mat/0607165*, 2006.

[13] R. Pariser and R. G. Parr, "A semi-empirical theory of the electronic spectra and electronic structure of complex unsaturated molecules. i.," *The Journal of Chemical Physics*, vol. 21, no. 3, pp. 466–471, 1953.

[14] R. Pariser and R. G. Parr, "A semi-empirical theory of the electronic spectra and electronic structure of complex unsaturated molecules. ii," *The Journal of Chemical Physics*, vol. 21, no. 5, pp. 767–776, 1953.

[15] J. Hubbard, "Electron correlations in narrow energy bands," in *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, The Royal Society, vol. 276, 1963, pp. 238–257.

[16] E. H. Lieb and F. Wu, "Absence of mott transition in an exact solution of the short-range, one-band model in one dimension," *Physical Review Letters*, vol. 20, no. 25, p. 1445, 1968.

[17] E. H. Lieb, "Two theorems on the hubbard model," in *Condensed Matter Physics and Exactly Soluble Models*, Springer, 2004, pp. 55–58.

[18] W. Metzner, M. Salmhofer, C. Honerkamp, V. Meden, and K. Schönhammer, "Functional renormalization group approach to correlated fermion systems," *Reviews of Modern Physics*, vol. 84, no. 1, p. 299, 2012.

[19] A. Georges, G. Kotliar, W. Krauth, and M. J. Rozenberg, "Dynamical mean-field theory of strongly correlated fermion systems and the limit of infinite dimensions," *Reviews of Modern Physics*, vol. 68, no. 1, p. 13, 1996.

[20] M. Ganahl, M. Aichhorn, P. Thunström, K. Held, H. G. Evertz, and F. Verstraete, "Efficient dmft impurity solver using real-time dynamics with matrix product states," *arXiv preprint arXiv:1405.6728*, 2014.

[21] F. A. Wolf, A. Go, I. P. McCulloch, A. J. Millis, and U. Schollwöck, "Imaginary-time matrix product state impurity solver for dynamical mean-field theory," *Physical Review X*, vol. 5, no. 4, p. 041 032, 2015.

[22] P. Pfeuty, "The one-dimensional ising model with a transverse field," *ANNALS of Physics*, vol. 57, no. 1, pp. 79–90, 1970.

[23] V. Zauner, M. Ganahl, H. G. Evertz, and T. Nishino, "Time evolution within a comoving window: scaling of signal fronts and magnetization plateaus after a local quench in quantum spin chains," *arXiv preprint arXiv:1207.0862*, 2012.

[24] F. Haldane and M. Zirnbauer, "Exact calculation of the ground-state dynamical spin correlation function of a s= 1/2 antiferromagnetic heisenberg chain with free spinons," *Physical review letters*, vol. 71, no. 24, p. 4055, 1993.

[25] G. M. Crosswhite, A. C. Doherty, and G. Vidal, "Applying matrix product operators to model systems with long-range interactions," *Physical Review B*, vol. 78, no. 3, p. 035 116, 2008.

[26] P. Corboz, S. R. White, G. Vidal, and M. Troyer, "Stripes in the two-dimensional t-j model with infinite projected entangled-pair states," *Physical Review B*, vol. 84, no. 4, p. 041 108, 2011.

[27] J. Haegeman, J. I. Cirac, T. J. Osborne, I. Pižorn, H. Verschelde, and F. Verstraete, "Time-dependent variational principle for quantum lattices," *Physical review letters*, vol. 107, no. 7, p. 070 601, 2011.

[28] G. Vidal, "Efficient simulation of one-dimensional quantum many-body systems," *Physical review letters*, vol. 93, no. 4, p. 040 502, 2004.

[29] E. Stoudenmire and S. R. White, "Minimally entangled typical thermal state algorithms," *New Journal of Physics*, vol. 12, no. 5, p. 055 026, 2010.

[30] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Physical Review Letters*, vol. 91, no. 14, p. 147 902, 2003.

[31] (2015). Scipy.org, [Online]. Available: www.scipy.org/ (visited on 01/10/2015).

[32] J. Haegeman, T. J. Osborne, and F. Verstraete, "Post-matrix product state methods: to tangent space and beyond," *Physical Review B*, vol. 88, no. 7, p. 075 133, 2013.

[33] M. J. Ganahl, "Dynamics of strongly correlated one-dimensional quantum systems using matrix product states," PhD thesis, Graz University of Technology, 2014.

[34] S. R. White, "Density matrix formulation for quantum renormalization groups," *Physical Review Letters*, vol. 69, no. 19, p. 2863, 1992.

[35] J. Haegeman, C. Lubich, I. Oseledets, B. Vandereycken, and F. Verstraete, "Unifying time evolution and optimization with matrix product states," *arXiv preprint arXiv:1408.5056*, 2014.

[36] M. P. Zaletel, R. S. Mong, C. Karrasch, J. E. Moore, and F. Pollmann, "Time-evolving a matrix product state with long-ranged interactions," *Physical Review B*, vol. 91, no. 16, p. 165 112, 2015.

[37] G. M. Crosswhite and D. Bacon, "Finite automata for caching in matrix product algorithms," *Physical Review A*, vol. 78, no. 1, p. 012 356, 2008.

[38] (2016). Http://julialang.org/, [Online]. Available: http://julialang.org/ (visited on 04/19/2016).

[39] (2016). Http://www.toshiba.at/discontinued-products/satellite-u500-119/, [Online]. Available: http://www.toshiba.at/discontinued-products/satellite-u500-119/ (visited on 04/19/2016).

[40] L. N. Trefethen and D. Bau III, *Numerical linear algebra*. Siam, 1997, vol. 50.

[41] L. Orecchia, S. Sachdeva, and N. K. Vishnoi, "Approximating the exponential, the lanczos method and an o(m)-time spectral algorithm for balanced separator," *arXiv preprint arXiv:1111.1491*, 2011.