Exotic phases of matter in quantum magnets

A tensor networks tale

Niesen, I.A.

Publication date
2018

Document Version
Other version

License
Other

Citation for published version (APA):

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.
The $N$-body wave function, the object that mathematically describes an $N$-body quantum state, is one gigantic tensor with a number of coefficients that is exponential in the system size $N$. The main idea behind the application of tensor networks to quantum mechanics is to decompose this big tensor into a network of smaller tensors. Doing so has the numerical advantage that local manipulations to the $N$-body wave function can be applied to only a few relatively small tensors in the network, rather than the the wave function in its entirety. Additionally, by constructing networks with particular structures, it is possible to target specific subsets of the $N$-body Hilbert space that are especially relevant for ground states of large class of Hamiltonians. Moreover, the size of these subsets scales only linearly in $N$, as opposed to the exponential scaling of the dimension of the total Hilbert space, and therefore allows for numerical simulation of systems much larger than those accessible through exact diagonalization.

This chapter will provide an extensive introduction into tensor networks in the context of quantum mechanics. Section 2.2 introduces the concept of entanglement entropy, and its relation to ground states of a generic class of Hamiltonians in particular, and motivates why the use of tensor networks is appropriate here. Section 2.3 provides an introduction to the one-dimensional tensor network states known as matrix product states, followed by a more extensive discussion on their two-dimensional generalizations called projected entangled pair states, the latter of which form the backbone of all numerical algorithms used in this thesis. Special attention will be given to network contraction—required for the computation of expectation values, discussed in Section 2.4; and the energy minimization algorithms employed in the chapters that follow, explained in Section 2.5. Finally, we conclude the chapter with Section 2.6 by discussing some practicalities regarding tensor network simulations.
2. Tensor networks

2.1 Theoretical setup

The objects of study of this thesis are two-dimensional $N$-body lattice systems with a single spin-$S$ particle pinned to each lattice site. The $N$-body Hilbert space describing such systems consists of a tensor product of local Hilbert spaces

$$\mathcal{H} = \bigotimes_{i=1}^{N} \mathcal{H}_i$$

where each local Hilbert space $\mathcal{H}_i$ is isomorphic to $\mathbb{C}^d$ for some fixed $i$-independent number $d$ related to the spin $S$ of the particles at the vertices. For example, for the bilinear-biquadratic Heisenberg models discussed in Chapters 3 and 4, $S = 1$, and therefore $d = 3$. For the Shastry-Sutherland models discussed in Chapters 5 and 6, $S = 1/2$. However, it will turn out to be convenient to group two physical sites together into a single $\mathcal{H}_i$, and therefore $d = 2 \cdot 2 = 4$.

Furthermore, this thesis deals with local Hamiltonians. That is, Hamiltonians that consist of a sum of finitely supported operators, meaning that there exists a (finite) number $n \in \mathbb{N}$ such that the Hamiltonian $H$ can be written as a sum of operators

$$H = \sum_j h_j,$$

with each $h_j$ acting non-trivially only on at most $n$ neighboring particles. The locality restriction is not as severe as it seems, because all physical forces relevant for condensed matter physics decay at a distance, and can effectively be considered of finite range. In particular, all Hamiltonians considered in this thesis are either nearest neighbor, or next-nearest neighbor.

Finally, we require the following definition. A gapped system has an energy gap in the spectrum of $H$ between the ground state and the first excited state in the thermodynamic limit $N \to \infty$. Of course, any finite ($N < \infty$) system is described by a finite-dimensional Hilbert space, and therefore has a finite thus discrete spectrum with in particular an energy gap between the the ground state and the first excited state. The term “gapless” refers to the fact that this gap persists in the thermodynamic limit.

---

1 Given by $d = 2S + 1$.  
2 With respect to some distance measure, which on a lattice typically means that two particles that interact directly through one of the $h_j$’s cannot be further than $n$ edges apart.  
3 Factualy the only force relevant in condensed matter physics is the electromagnetic force, which, although a long-range interaction, decays exponentially with distance in actual materials due to screening effects, and therefore is effectively short-ranged. Nuclear forces like the strong force become stronger over distance between, for example, deconfined quarks, but that only happens at energies far beyond the realm of condensed matter physics.
Additionally, all quantum states in this thesis are assumed to be pure. It is also possible to describe mixed states with tensor networks, see e.g. Ref. [55], but they are not relevant to this thesis.

### 2.2 The area law of entanglement entropy

Given an $N$-body quantum state $|\psi\rangle \in \mathcal{H}$, we associate an entropy to each subdivision of the lattice into two distinct parts $A$ and $B$. This works as follows. To a subdivision of the lattice corresponds a division of the total Hilbert space into $\mathcal{H}_A = \bigotimes_{i \in A} \mathcal{H}_i$ and $\mathcal{H}_B = \bigotimes_{i \in B} \mathcal{H}_i$. When averaging over, for example, part $B$, which we formally do by tracing out $\mathcal{H}_B$, we obtain a probabilistic description of the state of $A$ given by the reduced density matrix

$$\rho_A = \text{Tr}_B |\psi\rangle \langle \psi|.$$ 

The von Neumann entropy associated with the probability distribution of $\rho_A$ is called the entanglement entropy

$$S(\rho_A) = - \text{Tr}[\rho_A \log(\rho_A)].$$

It reflects the uncertainty in the state of the $A$ part given that we have averaged over $B$. Note that the same entropy is obtained when tracing out the $A$ part instead, i.e. $S(\rho_A) = S(\rho_B)$, and therefore we also denote the entanglement entropy corresponding to the division of $A$ and $B$ by $S_A|B$ (suppressing the $|\psi\rangle$ dependence in the notation).

For a generic quantum state, the entanglement entropy is proportional to the volume of $A$ [55,57]. However, this volume proportionality does not hold for every quantum state. Product states, for example, are not entangled at all, and their entanglement entropy is zero for any choice of subsystem $A$.

There are also states for which the entanglement entropy scales with the size of boundary $|\partial A|$ of subsystem $A$—known as the area law of entanglement entropy [55,57], or simply area law for short. Intuitively, the area law stems from the notion that particles are mostly correlated with their close neighbors, a property that you would expect ground states of local Hamiltonians to have. This intuition has been shown to be correct for the cases of gapped local Harmonic Hamiltonians on generic lattices [58], and for gapped Hamiltonians under mild conditions of the lattice [59], for which it is proven that the correlations within

---

4A Hermitian positive semidefinite trace 1 operator, for which—when put in diagonal form—the probability to find the $A$ part in a specific state is given by the corresponding diagonal coefficient of the density matrix.

5Or, more accurately, the average self-information, also called the Shannon entropy [56], of the probability distribution of Footnote 4.

6The name “area” refers to the area enclosing the volume of subsystem $A$. 

---

13
the ground state decay exponentially with distance, making them effectively short ranged. Moreover, for one-dimensional systems, it has been proven that ground states of gapped local Hamiltonians satisfy the area law \[60\]. In higher dimensions, it is proven that ground states of free gapped bosonic and fermionic models \[58,61\], and also of gapless free bosonic models on cubic lattices \[62\] satisfy area laws.

Clearly, entanglement entropy plays an important role in quantum many-body systems; a role that is also central in the construction of tensor network states. More specifically, as will be explained in Sections 2.3.1 and 2.3.2, the tensor network states considered in this thesis come with a parameter called the bond dimension, labeled by \(D\), which controls the amount of entanglement entropy that can be captured by the tensor network state in question. Moreover, we will also demonstrate in Section 2.3.3 that the tensor network states considered in this thesis satisfy the area law by construction, and are therefore expected to be suitable candidates for ground states of gapped local Hamiltonians.

However, the fact that tensor networks satisfy an area law does not a priori imply the reverse statement, which turns out does hold, under certain conditions. As a matter of fact, the success of tensor networks in simulating quantum many-body systems, and ground states in particular, is due to the statement that states that satisfy area laws can be well approximated by tensor network states. This statement has only been proven by Hastings \[60\] for one-dimensional systems, in which case the approximating tensor network state is a so-called matrix product state, or MPS for short. The fact that area law states can be well approximated by tensor network states in one-dimensional systems, implies that tensor networks are tailored for approximating ground states of gapped local Hamiltonians of one-dimensional systems. Moreover, Verstraete and Cirac \[63\] provide theoretical justification for the use of MPS even for critical (gapless) systems.

In higher dimensions, no such proofs exist. However, using higher-dimensional MPS consisting of tensors that represent all sites in a single hyperplane \[64\], Hastings did manage to show that, under an assumption on the density of states which is believed to be satisfied by many physical systems such as those displaying the fractional quantum Hall effect, an efficient higher-dimensional MPS representation of the ground state exists in any dimension \[65\]. In addition, Ref. \[63\] provides a hand-waving argument based on the exponential decay of the correlation length that suggests that ground states of gapped local Hamiltonians are well approximated by the natural generalization of MPS to two dimensions, called projected entangled pair states, PEPS for short. Moreover, a growing body of numerical tensor network studies of two-dimensional systems \[66–81\] using (infinite) PEPS suggests that ground states of both gapped and gapless local Hamiltonians can be efficiently approximated by tensor network states also in higher dimensions. For the case of gapless systems, the use of tensor network states—in any dimension—involves a controlled limiting procedure described in Section 2.3.4.

Having motivated that the tensor network states of MPS (PEPS) type are known
2.3. Tensor network states

The application of tensor networks to quantum mechanics involves decomposing a pure quantum many-body state into smaller pieces. There are many ways of doing so. However, we shall focus only on two particular types of decompositions: that of matrix product states (MPS) [82–84] for one-dimensional, and projected entangled pair states (PEPS) [85] for two (and higher) dimensional systems. Because all systems studied in this work are two-dimensional, we shall keep the discussion on one-dimensional systems somewhat brief, and elaborate on technical details mostly for the two-dimensional case.

In the remainder of this thesis, we shall adopt the following pictorial representation of tensors—see Fig 2.1. A tensor will be denoted by a ball (or in some cases by some other shape, such as a square or a diamond). Indices of the tensor will be portrayed by legs coming out of the tensor. When two lines of different tensors are connected, a contraction over the connected indices is implied.

\[ (Mv)_i = \sum_j M_{ij} v_j. \]

![Figure 2.1: Graphical notation for tensor networks. Each tensor is represented by a ball.](image)

(a) A vector \( v \) with a single index \( i \). (b) A matrix \( M \) with indices \( i \) and \( j \). (c) A matrix-vector multiplication: connected lines are summed over, i.e. \( (Mv)_i = \sum_j M_{ij} v_j \).

2.3.1 Matrix product states

Consider a one-dimensional \( N \)-site spin chain, with a spin-\( S \) particle described by a local Hilbert space \( \mathcal{H}_j \) of (site-independent) dimension \( d \) sitting on each lattice site. For simplicity, we shall assume open boundary conditions. Let us fix a local basis \( \{|i_j\\}_{1}^{d} \) for each \( \mathcal{H}_j \).

An MPS [82–84] consists of a collection of local tensors \( T_{\alpha_j, \alpha_{j-1}}^{i_j} \), one per site\(^7\). Each tensor has three indices, or legs: one physical leg \( i_j \) representing the physical

\(^7\)By abuse of notation, we label the site-tensors \( T_{\alpha_j, \alpha_{j-1}}^{i_j} \)—which generally vary per site and therefore should also have a separate site index \( j \)—with their physical index \( i_j \).
local Hilbert space $H_j$, and two auxiliary legs $\alpha_{j-1}$ and $\alpha_j$ that connect to the auxiliary legs of the two neighboring tensors that are to the immediate left and right, respectively, of the tensor in question; see Fig. 2.2. Note that the tensors at the ends of the system ($j = 1$ and $j = N$) carry just one auxiliary index\(^8\). Let us label the auxiliary vector spaces situated in between each pair of sites, using the convention that $V_j$ is in between site $j$ and site $j + 1$.

![Auxiliary indices](image)

**Figure 2.2:** Left: a site tensor with one physical and two auxiliary legs. Right: a six-site matrix product state for a system with open boundary conditions. For periodic boundary conditions, the tensors at either end will be connected by an additional auxiliary bond [86]. The auxiliary indices $\alpha_i$ are summed over.

Suppose $|\psi\rangle \in H$ is some arbitrary state in the $N$-body Hilbert space. Expressed in the local bases defined above, $|\psi\rangle$ takes the form:

$$|\psi\rangle = \sum_{i_1, \ldots, i_N} \psi_{i_1, \ldots, i_N} |i_1, \ldots, i_N\rangle.$$  (2.1)

It is possible to rewrite $|\psi\rangle$ as a matrix product state using a sequence of singular value decompositions (see e.g. Ref. [87]),

$$\psi_{i_1, \ldots, i_N} = T^{i_1} \cdot T^{i_2} \cdots T^{i_N},$$  (2.2)

where the dots represent contractions of the auxiliary indices. Eq. 2.2 explains the name “matrix product state”, as every fixed basis coefficient of $\psi$ is written as a product of matrices.

The MPS obtained through a sequence of singular value decompositions has the property that the dimensions of the auxiliary vector spaces increase exponentially when moving towards the middle of the chain. That is, dim($V_1$) = $d$, dim($V_2$) = $d^2$, etc., up to the middle bond, where the auxiliary vector space $V_{N/2}$ has dimension\(^9\) dim($V_{N/2}$) = $d^{N/2}$, after which the dimensions start decreasing again until the last auxiliary space has dim($V_{N-1}$) = $d$. For example, the auxiliary vector spaces corresponding to indices $\alpha_1, \ldots, \alpha_5$ of the MPS in Fig. 2.2 have dimensions $(d, d^2, d^3, d^2, d)$.

---

8Unless we use periodic boundary conditions, in which case the outer two tensors are connected by an additional auxiliary bond; see Ref. [86] for details.

9Assuming $N$ is even, otherwise the maximal dimension is $d^{(N-1)/2}$.
It is also possible to consider MPS for which all auxiliary vector spaces have the same dimension. This dimension, which we will denote by $D$, is called the bond dimension. From the paragraph above, we conclude that any state can be represented by an MPS, provided we take $D$ to be exponentially large in the system size.

Matrix product states with fixed bond dimension $D \ll \exp(N)$, on the other hand, contain far fewer parameters than general states in the $N$-body Hilbert space. Indeed, a bond-dimension-$D$ MPS has of the order of $NdD^2$ parameters, which, because it only scales linearly in $N$, is much less than the $d^N$ coefficients required to describe an arbitrary $|\psi\rangle \in \mathcal{H}$ for large $N$. Clearly, fixed-$D$ MPS with $D \ll \exp(N)$ parametrize only a small part of the total Hilbert space.

It turns out, that these fixed-$D$ MPS are states with a specific property: they satisfy the area law\(^{10}\), which for one-dimensional systems just means that the entanglement entropy between a left and a right part of the chain is (bounded by a) constant\(^{11}\), and they have exponentially decaying correlation functions; see e.g. Refs. [55,88]. It is precisely these fixed-$D$ MPS that can well approximate ground states of gapped local Hamiltonians [60], as mentioned earlier in Section 2.2. Consequently, by restricting simulations to the set of fixed-$D$ MPS, it becomes numerically viable to study systems described by gapped local Hamiltonians for a particle number $N$ much larger than those for which the Hamiltonian is exactly diagonalizable. From now on, when referring to an MPS, we shall always mean a fixed-$D$ MPS unless explicitly stated otherwise.

The origins of MPS trace back to the well-known density matrix renormalization group (DMRG) algorithm [89,90]—a method that has been applied successfully to numerous one and later also two-dimensional systems (see e.g. Ref. [91] and references therein)—in the sense that DMRG can be viewed as a variational method that optimizes over the class fixed-$D$ MPS [86,92–94].

Inspired by the success of MPS in simulating one-dimensional systems (see e.g. Refs. [55,88,95] and references therein), it seems natural to ask whether (i) there is natural generalization of MPS to higher dimensions, and (ii) to what extent this higher dimensional generalization of MPS is capable of approximating ground states. To answer these questions, in the next sections we shall discuss the concept of a projected entangled pair state, or PEPS for short, which can be thought of as a direct generalization of MPS to two dimensions.

It should be noted that MPS can also be applied to two-dimensional systems by connecting sites in a snake-like pattern [96]. The advantage of using MPS

---

\(^{10}\)The fact that MPS satisfy an area law follows directly from a derivation analogous to the one presented in Section 2.3.3 for projected entangled pair states.

\(^{11}\)Indeed, because only one bond gets cut when splitting the system into a left part $A$ and a right part $B$, the entanglement entropy is bounded by $S_{A|B} \leq \log(D)$. Likewise, for MPS with periodic boundary conditions, there are two bonds connecting subsystems $A$ and $B$, and $S_{A|B} \leq 2\log(D)$. See Section 2.3.3 for details.
over PEPS, is that the algorithms for one-dimensional systems are less involved than those for higher dimensions; see e.g. Sections 2.4 and 2.5. However, the application of MPS to two-dimensional systems is restricted to strips or cylinders (depending on the boundary conditions) of relatively small widths. For example, the comparative study of Ref. [97] showed that iPEPS—which stands for infinite PEPS, discussed in Section 2.3.5—outperforms iMPS for widths of about seven to eleven sites for typical models such as Heisenberg and the Hubbard model.

### 2.3.2 Projected entangled pair states

A *projected entangled pair state* (PEPS) [85] is the two-dimensional generalization of a matrix product state. Instead of having three legs, local tensors of a PEPS are five-legged: with one index corresponding to the physical local Hilbert space $\mathcal{H}_i$, and four auxiliary indices that run over $D$ values each. We will denote a PEPS with bond dimension $D$ by $\psi(D)$. The state $\psi(D)$ is formed by connecting all local tensors in a square lattice pattern; see Fig. 2.3. Recall that, in the graphical notation of Fig 2.3, the summation over the auxiliary indices is implied by the fact that the corresponding legs are connected.

The set of fixed-$D$ PEPS is described by $NdD^4$ parameters\(^{12}\), a number that, as with MPS, scales linearly in $N$. Consequently, by restricting simulations to the set of fixed-$D$ PEPS, it becomes possible to simulate systems much larger than those for which the Hamiltonian can be diagonalized exactly. Moreover, PEPS satisfy the area law, as we will show in Section 2.3.3, and, as mentioned in Section 2.2, it is expected that PEPS can efficiently approximate ground states of gapped local Hamiltonians.

\(^{12}\)Neglecting the boundary tensors—which have $dD^3$ coefficients when at the edge, and $dD^2$ when at a corner of the system—that are not important in the large $N$ limit.
The name *projected entangled pair state* comes from the fact that it can be written as a projection of another state (in some higher-dimensional space) that consists of products of maximally entangled pairs [85]. The term PEPS is sometimes also used for the higher dimensional equivalents [98, 99] of the square lattice tensor network state of Fig. 2.3. In this thesis, however, we shall reserve the names MPS and PEPS specifically for one-dimensional and two-dimensional square lattice networks respectively, and reserve the term *tensor network state* for states made up of arbitrary tensor networks.

In addition to MPS and PEPS, there exist many other types of tensor network states, such as tree tensor networks (TTN) [100] and the multi-scale entanglement renormalization ansatz (MERA) [101]. The former have the advantage that, while being higher-dimensional, they have the property that cutting a single bond cuts the network into two, which makes them applicable to fast and well-understood one-dimensional-like algorithms. The latter is specifically designed to represent critical (gapless) states, the type of states that ordinary tensor network states find difficult to approximate. For more details about TTN’s, MERA, and tensor networks in general we refer the reader to the excellent pedagogical introductions to tensor networks by Cirac and Verstraete [102], Schollwöck [87], Eisert [55] and Orús [88], and references therein.

### 2.3.3 PEPS satisfy the area law

By construction, because local tensors only connect to their immediate neighbors, every PEPS satisfies an area law in its (von Neumann) entanglement entropy scaling; see e.g. Ref. [88].

More precisely, take an arbitrary PEPS $|\psi\rangle \in \mathcal{H}$, and suppose we cut the total system into two pieces, $A$ and its complement $B$ (see Fig. 2.4). Let $\{|i_A\rangle\}$ and $\{|i_B\rangle\}$ be bases for the tensor products of the local Hilbert spaces $\mathcal{H}_A = \bigotimes_{i \in A} \mathcal{H}_i$ and $\mathcal{H}_B = \bigotimes_{i \in B} \mathcal{H}_i$ respectively. Because of the square lattice pattern of the PEPS in question, the number of bonds $L$ connecting $A$ and $B$ is proportional to the length of the cut $l$. Moreover, since each auxiliary vector space associated to a single bond has dimension $D$, the tensor product of all auxiliary vector spaces along the the cut between $A$ and $B$ has dimension $D^L$. Let $\{|\alpha\rangle\}$ be a basis for the latter. Now, we can write

$$|\psi\rangle = \sum_{i_A, i_B, \alpha} A_{i_A, \alpha} B_{i_B, \alpha} |i_A\rangle |i_B\rangle,$$

where $A$ and $B$ are the tensors obtained by contracting all auxiliary indices of the parts of the network that are completely contained within parts $A$ and $B$ respectively.
2. Tensor networks

Figure 2.4: A 20-site PEPS cut into two pieces, A and B. The length \( l \) of the boundary (red dots) separating A and B is proportional to the number of bonds \( L \) connecting A and B.

Next, the reduced density matrix for part A takes the form

\[
\rho_A = \text{tr}_B (|\psi\rangle\langle\psi|) \\
= \sum_{\alpha,\beta} \sum_{i_A,j_A,i_B} A_{i_A,\alpha} B_{i_B,\alpha}^* A_{j_A,\beta}^* |i_A\rangle\langle j_A|
\]

\[
= \sum_{\alpha,\beta} (B^\dagger B)_{\beta,\alpha} |\psi_A^\alpha\rangle\langle \psi_B^\beta|,
\]

where \( |\psi_A^\alpha\rangle = \sum_{i_A} A_{i_A,\alpha} |i_A\rangle \), and similarly for the bra \( \langle \psi_B^\beta| \). Since the rank of a matrix is bounded by both its number of columns and its number of rows, \( \rho_A \) has a rank of at most \( D^L \). Assuming that parts A and B are maximally entangled, and that \( \rho_A \) has maximal rank \( D^L \)—meaning that \( \rho_A \) has a flat spectrum with all eigenvalues equal to \( 1/D^L \)—then the entanglement entropy is given by \( S_{A|B}^{\text{max}} = -\text{tr}[\rho_A \log(\rho_A)] = L \log(D) \). This is an upper bound, and therefore

\[
S_{A|B} \leq L \log(D). \tag{2.3}
\]

In particular, because \( L \propto l \), we conclude that a PEPS with bond dimension \( D \) can reproduce states ranging from non-entangled product states to states that satisfy the area law by saturating Eq. (2.3). Furthermore, from the above analysis, we conclude that every bond contributes at most \( \log(D) \) to the entanglement entropy corresponding to any subdivision for which the bond in question is part of the boundary separating both subsystems. Hence, we not only obtain the physical interpretation of \( D \) as a parameter that provides a bound on how entangled different parts of a system can be, but we also gain physical intuition for how each bond can communicate entanglement between two parts of a given system. In other words, tensor networks provide a geometric picture of entanglement entropy; see also Refs. [103–105].
2.3. Tensor network states

2.3.4 The large D limit

For each fixed $D$, let us denote the set of all PEPS of bond dimension $D$ by $\mathcal{M}_D$. It is important to point out that for each $D$ used in practice $\mathcal{M}_D$ is strictly smaller than $\mathcal{H}$. Therefore, after optimizing the tensors in a given PEPS through some minimization algorithm, what we obtain is (i) an approximate ground state, and (ii), because every PEPS is a state in the Hilbert space, an upper bound to the true ground state energy. That is to say that a PEPS, and more generally any tensor network state, is a variational ansatz for the ground state.

The subsets $\mathcal{M}_D$ of $\mathcal{H}$ have the property that they form a strictly increasing sequence with respect to set inclusion, meaning that, if we have $D' < D$, then $\mathcal{M}_{D'} \subsetneq \mathcal{M}_D$. Consequently, the bond dimension serves as an accuracy parameter, such that the accuracy of a simulation increases with increasing $D$ because the set of states that we optimize over increases in size.

In addition, any arbitrary state in the Hilbert space can be well approximated by a PEPS provided the bond dimension is large enough [55, 88], i.e. $M_D \rightarrow \mathcal{H}$ as $D \rightarrow \infty$. As with MPS, for a PEPS to be able to represent an general state, the bond dimension needs to be exponential in the number of particles $N$. However, Ref. [63] suggests that, compared to MPS, for PEPS a smaller bond dimension is required to acquire the same level of accuracy. Moreover, it is known that two-dimensional tensor networks for a bond dimension as low as $D = 2$ can represent even critical (gapless) ground states [107] and exhibit algebraically decaying correlation functions, which implies that PEPS can represent a broader class of states than their one-dimensional counterparts.

Ground states of gapped Hamiltonians should in principle be representable by a fixed-$D$ PEPS. For gapless systems, tensor network methods also offer accurate results provided the bond dimension is extrapolated to infinity. In practice, if the desired accuracy requires it, we shall extrapolate $D \rightarrow \infty$ for any quantity of interest by fitting curves through the finite-$D$ data obtained through our simulations; see Section 2.6.3.

2.3.5 Infinite systems

It is possible to simulate infinite systems using PEPS [108–111], if we assume that the state $|\psi\rangle$ that is to be simulated contains some form of translational symmetry, in the sense that it can be described by a unit cell that is repeated all over the lattice. If the state is completely translationally symmetric, the unit cell consists of a single tensor. However, for states that partially break translational symmetry, a larger unit cell will be used. For example, a square lattice antiferromagnetic state, which has neighboring spins anti-aligned in a checkerboard pattern, requires a 2x1

$^1$Details on the geometry of $\mathcal{M}_D$ can be found in Ref. [106].
unit cell. In the iPEPS algorithm, this pattern will be implemented as a 2x2 unit cell, where tensors that are diagonally opposite within the unit cell are identical: i.e. only one copy of each is kept in memory (Fig 2.5).

**Figure 2.5:** iPEPS with a 2x2 unit cell—marked by the purple dotted line—with tensors of the same color being identical, forming a checkerboard pattern required to simulate an antiferromagnetic ground state.

We write $L_x$ and $L_y$ for the width and length of the unit cell. The local tensor at position $(x, y)$ is denoted by $A_{[x,y]}$, where the straight brackets mean that we take the coordinates $x$ and $y$ modulo $L_x$ and $L_y$ respectively. Following the convention used in tensor network papers cited, $x$ increases towards the right, and $y$ increases towards the bottom of the unit cell (Fig. 2.5). The infinite system state $|\psi\rangle$ is now fully described by $L_x \times L_y$ tensors, which contain a total of $dD^4L_xL_y$ free parameters.

The type of PEPS introduced in this section is called infinite PEPS, or iPEPS for short. All systems investigated in this thesis are studied by means of iPEPS. One of the main difficulties in dealing with PEPS, and iPEPS specifically, is contracting them—which is required to compute expectation values. How to deal with this issue will be explained in Section 2.4.

### 2.3.6 Abelian symmetries

It is possible to encode symmetries into the tensor network itself. Throughout this thesis, the only symmetries used are $U(1)$ spin-rotation symmetries [112, 113]. In practice, $U(1)$ symmetry is implemented as a discrete $\mathbb{Z}_q$ symmetry, thought of as a discrete subgroup of $U(1)$, with a large enough $q$ to prevent different charge sectors (explained below) from mixing. In the section that follows we shall use $U(1)$ and $\mathbb{Z}_q$ as leading examples.

Let $G$ be an abelian group, and suppose we have unitary representations $U_i : G \rightarrow U(H_i)$ on each local Hilbert space $H_i$. The group $G$ naturally acts on the tensor
product of any subset $\mathcal{I} \subseteq \{1, \ldots, N\}$ of local Hilbert spaces $\bigotimes_{i \in \mathcal{I}} \mathcal{H}_i$ through

$$U_\mathcal{I}(h) \otimes_{i \in \mathcal{I}} |\psi_i\rangle := \otimes_{i \in \mathcal{I}} U_i(h)|\psi_i\rangle$$

for $h \in \mathcal{G}$. Furthermore, assume that $\mathcal{G}$ is generated by a single generator, and let $g$ be the generator of the group. In case that $\mathcal{G}$ is a Lie group, then $g$ is an element of the Lie algebra $\mathfrak{g}$ of $\mathcal{G}$. For example, if $\mathcal{G} \simeq U(1)$, then $\mathfrak{g} \simeq \mathbb{R}$, and $g = 1 \in \mathbb{R}$. Without loss of generality, we can let $U(1)$ represent the group of rotations around the $z$-axis. That is, acting on any collection $\mathcal{I}$ of local Hilbert spaces, by abuse of notation,

$$g = \sum_{i \in \mathcal{I}} \sigma^z_i$$

is the generator of the map that rotate all particles in $\mathcal{I}$ around the $z$-axis.

If $\mathcal{G}$ is a discrete group, a generator $g$ of $\mathcal{G}$ is any element whose orbit equals all of $\mathcal{G}$. For example, for $\mathcal{G} \simeq \mathbb{Z}_q$, with $q \in \mathbb{N}$, we let $g = 1$ be the smallest rotation in $\mathbb{Z}_q$. Its representation on any collection of local Hilbert spaces $\mathcal{I}$ is

$$g = \exp \left( \frac{2\pi i}{q} \sum_{i \in \mathcal{I}} \sigma^z_i \right);$$

$g$ rotates all spin particles in $\mathcal{I}$ around the $z$-axis over an angle $2\pi i/q$. Technically, in the above examples, $g$ is a different map depending on the set $\mathcal{I}$, but we will use the same notation for all these maps to avoid cluttering.

Because each $g$ is hermitian (for every $\mathcal{I} \subseteq \{1, \ldots, N\}$), we can diagonalize it on each $\mathcal{H}_i$. For the examples considered, the eigenbasis for each $g$ is just the standard $z$-eigenbasis. The charge of a $g$-eigenstate is defined to be its eigenvalue, which in the case of $U(1)$ coincidentally equals the total magnetization of the set of local Hilbert spaces in question. By combining different subsystems, we can define a multiplication structure $\times$ on the space of charges, called charge fusion.

For example, if the total Hilbert space is that of two spin-1/2 particles: $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2$, with $\mathcal{H}_1 \simeq \mathcal{H}_2 \simeq \mathbb{C}^2$, then $g$ acts on each individual particle as $\sigma^z_i$, and on the total space as $\sum_{i=1}^2 \sigma^z_i$. The local $g$-eigenbasis of $\mathcal{H}_i$ is the standard $z$-eigenbasis $\{|\uparrow\rangle_i, |\downarrow\rangle_i\}$, with corresponding charges $\pm \frac{1}{2}$. The charges of the combined system are given by the table in Fig. 2.6.

That is, in case of $U(1)$ the total charge is the sum of the individual charges. I.e., the charge fusion is integer addition: $c_1 \times c_2 := c_1 + c_2$. Similarly, in the case of $\mathbb{Z}_q$,

---

14Formally defined as the tangent space of $\mathcal{G}$ at the identity: $g = T_1 \mathcal{G}$.

15Writing $g$ for $u_\mathcal{I}(g)$, where $u_\mathcal{I} = d_1 U_\mathcal{I}$ is the derivative of $U_\mathcal{I} : \mathcal{G} \to \bigotimes_{i \in \mathcal{I}} \mathcal{H}_i$ at the identity.

16By identifying the lowest $S^z$-eigenstate as an empty site, and incremental increases in magnetization as the addition of particles on-site, this representation can be seen as equivalent to that of the particle number representation.

17This time, writing $g$ for $U_\mathcal{I}(g)$.
2. Tensor networks

<table>
<thead>
<tr>
<th>state</th>
<th>individual charges</th>
<th>total charge</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑↑ ⊗ ↑↑</td>
<td>(\frac{1}{2}, \frac{1}{2})</td>
<td>1</td>
</tr>
<tr>
<td>↑↑ ⊗ ↓↓</td>
<td>(\frac{1}{2}, -\frac{1}{2})</td>
<td>0</td>
</tr>
<tr>
<td>↓↓ ⊗ ↑↑</td>
<td>(-\frac{1}{2}, \frac{1}{2})</td>
<td>0</td>
</tr>
<tr>
<td>↓↓ ⊗ ↓↓</td>
<td>(-\frac{1}{2}, -\frac{1}{2})</td>
<td>-1</td>
</tr>
</tbody>
</table>

Figure 2.6: Charge fusion for the combined system of two spin-1/2 particles.

The charge fusion is multiplication of complex q-roots of unity, which is equivalent to integer addition modulo q.

The multiplicative structure on the charges is invertible, in the sense that for each charge there exists a minus charge. Also, for (total) integer spin, there exists an identity element (with zero total magnetization).

A state \( |\psi\rangle \in \mathcal{H} \) is defined to be symmetric under \( \mathcal{G} \) if, for every \( h \in \mathcal{G} \),

\[
U(h) |\psi\rangle = |\psi\rangle.
\]

The above statement is equivalent to saying that the only non-zero basis coefficients of \( |\psi\rangle \) are those for which the total charge vanishes\(^{18}\). I.e., in case of \( U(1) \), the sum of all local charges of \( |\psi\rangle \) is zero: \( c_1 + \ldots + c_N = 0 \) is zero, and likewise for \( \mathbb{Z}_q \), but then the integer addition is modulo \( p \). For example, for the two spin-1/2 particles in Fig 2.6, the only \( U(1) \)-symmetric state is any linear combination of \( |\uparrow\rangle \otimes |\downarrow\rangle \) and \( |\downarrow\rangle \otimes |\uparrow\rangle \), since the other two basis states pick up a phase under simultaneous two-particle rotations around the \( z \)-axis.

If \( |\psi\rangle \) is a tensor network state, then symmetry of \( |\psi\rangle \) under \( \mathcal{G} \) can be guaranteed if we assume that the local tensors conserve charge, in the following way. For each tensor, we choose which indices are incoming, and which are outgoing. On the outgoing indices, \( \mathcal{G} \) acts normally, whereas on the incoming indices \( \mathcal{G} \) acts with the conjugate representation. We define the latter to mean that we act with the inverse of the generator (i.e. \( -g \) in the case of Lie groups, and \( g^{-1} \) for discrete groups). Factually, we are interpreting each tensor as a linear map from the incoming to the outgoing space, where the incoming indices transform as bra’s and the outgoing indices as kets. As a consequence, charges corresponding to incoming indices come with an extra minus sign. Now, we say that a tensor conserves charge if the only non-zero coefficients of the tensor are those for which the fusion of all charges—including signs from possible conjugate representations—is zero\(^{19}\).

\(^{18}\)Note that our definition of symmetric equals the notion of invariant, and does not incorporate the notion of covariant: that of a state that has a well-defined magnetization \( m \) (which can also be non-zero). However, we will be only interested in invariant states, so for our purpose the notions of symmetric and invariant are the same. The reason being, that, as noted in Ref. [112], every covariant tensor can be viewed as an invariant tensor by adding a trivial index that sits in
2.3. Tensor network states

![Diagram](image)

**Figure 2.7:** (a) Two $U(1)$-symmetric tensors $X$ and $Y$: the only non-zero coefficients are those for which total charge (per tensor) is zero. Because all indices are outgoing, all representations are regular. (b) In order to contract $X$ and $Y$ such that the result is again $U(1)$-symmetric, we must flip $Y$’s first index into incoming. Simply flipping the index will also flip the charge sectors, so we must relabel the first index accordingly to compensate. (c) Perform contraction. The resulting tensor is $U(1)$-symmetric by construction—see Fig. 2.8.

When contracting two tensors, the representations of $G$ on the vectors spaces on which the tensors act are chosen such that the indices that are about to be contracted are outgoing on one tensor and incoming on the other (see Fig. 2.7). Note that by designating fixed incoming and outgoing indices in accordance with the necessary contractions, step (a)$\rightarrow$(b) in Fig. 2.7 can be omitted, which saves computation time. Following the above recipe guarantees that, after the contraction, the resulting tensor is again $G$-symmetric; see Fig. 2.8. Finally, if necessary, the remaining open indices can be transformed from incoming to outgoing and vice-versa (by relabeling the charge sectors) to prepare for the next contraction.

It should be noted that, as defined above, the only $U(1)$-symmetric states are those for which the total magnetization vanishes. However, as described in Footnote 18, it is possible to manually adjust the total magnetization of a local tensor to a non-zero value, which allows for the simulation of states that have a non-zero magnetization.

On finite systems, the number of charge sectors present in the auxiliary vector spaces in the middle of the system theoretically scales exponentially with $N$. Consequently, for infinite systems, infinitely many charge sectors are required to fully represent a $U(1)$-symmetric state. However, because of the use of finite $D$, it is not possible to accommodate all these sectors. Typically, for a $U(1)$-symmetric ground state, the charge sectors become more sparsely occupied as we move further away from the total magnetization zero sector. Therefore, we can in practice do with finitely many symmetry sectors. The dimensions of the symmetry sectors are determined dynamically in the algorithms discussed further down this chapter.

---

Footnotes:

19. Pictorially, $G$-symmetry of a local tensor is represented by the equality sign in the green box in Fig 2.8.
2. Tensor networks

\[ (i) \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\bullet \\
I = UU^\dagger
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \] 

\[ (ii) \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \]

\[ (iii) \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \quad \Rightarrow \quad \begin{array}{c}
\bullet \\
U & U & U
\end{array} \]

Figure 2.8: The fact that \((i) = (iii)\) shows $G$-symmetry of a tensor network state that is composed of $G$-symmetric local tensors, since acting with the same group element $U = U(h)$ (using same notation for each local representation) on each open index leaves the state invariant. \((i) \rightarrow (ii)\) Insert identities, using that the representations are unitary ($I = UU^\dagger$), and putting the dagger where the arrow is incoming. \((ii) \rightarrow (iii)\) Use that the local tensors are $G$-symmetric (green box).

The fact that \((i) = (iii)\) shows $G$-symmetry of a tensor network state that is composed of $G$-symmetric local tensors, since acting with the same group element $U = U(h)$ (using same notation for each local representation) on each open index leaves the state invariant. \((i) \rightarrow (ii)\) Insert identities, using that the representations are unitary ($I = UU^\dagger$), and putting the dagger where the arrow is incoming. \((ii) \rightarrow (iii)\) Use that the local tensors are $G$-symmetric (green box).

based on how the states with largest singular values are distributed amongst the symmetry sectors.

The algorithms explained in the next sections work the same for symmetric and non-symmetric tensor networks, with the exception that in case of symmetric tensors, tensor contractions happen block-wise. The advantage of using symmetric tensors is that the tensor blocks used in the calculations are much smaller than the original (asymmetric) tensor, reducing computation time and memory load significantly.

2.4 Network contraction

2.4.1 One-dimensional systems

For illustrative purposes, before discussing the more complicated two-dimensional case, let us briefly discuss the contraction of tensor networks in the one-dimensional setting; see e.g. Refs [55,87,88].

Computing the expectation value $\langle \psi | O | \psi \rangle / \langle \psi | \psi \rangle$ of some operator $O$ requires that we contract an entire tensor network: summing not only over auxiliary indices in both $\langle \psi |$ and $| \psi \rangle$, but also over the physical indices connecting $\langle \psi |$ and $| \psi \rangle$. The bra $\langle \psi |$ corresponding to a tensor network state $| \psi \rangle$ comprises of exactly the
2.4. Network contraction

Same tensors, but with complex conjugated coefficients\textsuperscript{20}. It is represented by a tensor with the physical legs pointing upwards instead of downwards. The network corresponding to the expectation value of a two-site operator $O$ with respect to five-site MPS is depicted in Fig. 2.9.

From a computational standpoint, when contracting tensor networks, the order of contraction is important. Typically, the network should be contracted in such a way that the intermediate tensors remain as small as possible during the contraction process. For an $N$-site MPS, this means moving from one side of the chain to the other, at each step contracting first the auxiliary index connected to the nearest local ket tensor at site $i$, followed by absorbing the corresponding local bra tensor at site $i$ by contracting over an auxiliary and and the $i$-th physical index simultaneously. The costs for computing an expectation value in this way is $O(NdD^3)$ [88].

Infinite matrix product states (iMPS) [114], just like their two-dimensional counterparts, can only be simulated on a computer if some kind of translational invariance is assumed, in the sense that the state can be described by a unit cell that is repeated over the lattice. In the one-dimensional case, computing an expectation value boils down to evaluating a transfer matrix to the power infinity (see Fig 2.10), which corresponds to projecting onto its dominant eigenvector\textsuperscript{21}. This projection can be done exactly, and efficiently, as it scales as $O(dD^3)$ [88]. When a larger unit cell is used, the transfer matrix consists of a bracket of one whole

\textsuperscript{20}And, in case of $U(1)$-symmetric tensors, also reversed arrows.

\textsuperscript{21}It is possible that the eigenspace of dominant eigenvectors is degenerate, in which case the expectation value depends on which vector is used to close the network. However, because the set of matrices with degenerate dominant eigenspaces has measure zero within the set of all matrices of some predefined size, this is not likely to happen in practice for states that have been obtained through numerical optimization.
2. Tensor networks

![Figure 2.10](image)

*Figure 2.10: Computing the expectation value of a two-site operator (in green) for an iMPS with a one-site unit cell requires computing the transfer matrix (within brackets) to the power infinity.*

unit cell. Assuming the dominant eigenvector is unique, the network in Fig 2.10 can be closed by placing the same (arbitrary\(^{22}\)) vector at both ends, which will then get projected onto the dominant eigenvector.

### 2.4.2 Two-dimensional systems

In contrast to MPS, PEPS cannot be exactly efficiently contracted by choosing a smart contraction order. The contraction of a PEPS is contained in the complexity class \#P, making it a computationally hard problem [115].

In this thesis, we will be interested in contracting *infinite* two-dimensional networks. One of the main obstacles in dealing with infinite two-dimensional tensor networks is that, contrary to the one-dimensional case, it is impossible to contract the infinite two-dimensional network exactly. Instead, several approximation schemes have been developed over the last years, such as: transfer-matrix-based contraction schemes [110], coarse-graining-based contraction schemes [116, 117] and corner-transfer matrix (CTM) methods [118–120] based on a formalism derived by Baxter [121, 122]. All simulations ran in preparation of this thesis make use of a modified version [74, 123]—discussed below—of the CTM algorithm from Refs. [118, 124].

In the CTM contraction scheme, for each tensor in the unit cell additional environment tensors are introduced that represent the contraction of all other tensors surrounding the tensor in question: four corner matrices \(C_1, \ldots, C_4\), and four transfer matrices \(T_1, \ldots, T_4\) (Fig. 2.11). \(C_1\) always refers to a matrix that is in the upper-left corner, \(C_2\) in the upper-right corner, etcetera. Likewise, \(T_1\) is always at the top, \(T_2\) at the right, and so forth. Furthermore, all corner and transfer matrices carry site labels \([x, y]\)—that, like the labels of the local tensors, are taken

---

\(^{22}\)In practice, it doesn’t matter which one, because a random vector with have a non-zero overlap with the dominant eigenvector with probability one.
Figure 2.11: Approximate environment for reduced site tensor $a$ (green box). The bottom figure shows the labeling of the corner and transfer matrices relative to the site tensor at position $[x, y]$; all coordinates are taken modulo the length $L_y$ and width $L_x$ of the unit cell.
modulo $L_x$ and $L_y$ respectively—to signify what part of the lattice they represent. It is important to note that the corner and transfer matrices, which are used to compute expectation values, form the environments of the reduced site tensors (Fig. 2.11, green box).

**Figure 2.12:** A left move at horizontal position $x$ consists of (i) inserting a new column, (ii) absorbing the inserted column into the appropriate corner and transfer matrices—increasing the bond dimensions of the vertical bonds—and (iii) truncating the bond dimension back to $\chi$. This left move is done column by column at each location $[x, y]$ in the unit cell using the appropriate projectors (Fig. 2.13) for the truncation step, followed by similar top, right and bottom moves until all environment tensors have converged.
2.4. Network contraction

Figure 2.13: We want to compute the projectors along the vertical cut between positions $[x, y]$ and $[x, y+1]$, which will be inserted in between for example $\tilde{T}^{[x,y]}_4$ and $\tilde{C}^{[x,y+1]}_4$ in Fig. 2.11. To do so, we (a) take the square network surrounding the to be truncated bond, which we interpret as a two-site MPS consisting of a top and a bottom tensor, of which we take QR decompositions. Next, we (b) perform an SVD keeping only the $\chi$ largest singular values. We then (c) approximate the identity matrix using the SVD from the previous step, which provides us with the appropriate projectors $P^{[x,y]}_b$ and $P^{[x,y+1]}_t$. After repeating steps (a)-(c) for all values of $y$, keeping $x$ fixed, we (d) update all $T_4$, $C_1$ and $C_4$ tensors for fixed $x$ using the appropriate projectors. Finally, we repeat the above steps column-wise for every $x$, until all $T_4$, $C_1$ and $C_4$ tensors have been updated, and the left move is complete.
2. Tensor networks

The environment tensors come with their own boundary bond dimension $\chi$, and are obtained as a fixed point of a row and column insertion and truncation procedure [124], as explained in Fig 2.12. The fixed point is reached when the change in energy due to the insertion step stays within a predefined measure of accuracy. Each row or column insertion increases the boundary bond dimension, which has to be truncated back to $\chi$ by means of projectors that project onto the most relevant $\chi$-dimensional subspace. Depending on the location within the network, constructing the projectors involves different tensors. For example, in order to obtain the projectors for truncating the thick vertical bond between $\tilde{T}_{x,y}^4$ and $\tilde{C}_{x,y+1}^4$ back to $\chi$, the network in Fig. 2.13(a) is required. The idea behind the construction of projectors, is to interpret the left diagram of Fig. 2.13(a) as a two-site MPS, where the first site is the “upper” and the second “lower” tensor. By cutting all four legs open, we can then construct projectors that project onto the subspace of $\chi$ largest Schmidt coefficients corresponding to the cut at the two leftmost open legs.

Finally, given converged environment tensors for all sites within the unit cell, expectation values $\langle \psi | O | \psi \rangle$ can be computed by surrounding the sites connected to the operator $O$ in question by the appropriate corner and transfer matrices, and contracting the resulting (finite) tensor network.

The truncation back to $\chi$ in the CTM algorithm makes the environment formed by the $C$ and $T$ tensors an approximate environment, rather than the exact environment (which would require an infinite boundary bond dimension). However, all iPEPS used have a fixed bond dimension $D$, and are approximate states themselves. Computing expectation values with approximate environments adds a further approximation on top of this. From the perspective of the boundary bond dimension $\chi$, what matters is that the variations due to the use of finite $\chi$ are negligible compared to those due to the use of finite $D$. In practice, this means that we take $\chi(D) > D^2$ to be large enough to yield negligible variations in the expectation value that we are interested in compared to the variations due to the use of finite $D$.

2.5 Energy minimization

In order to compute ground states, the energy needs to minimized. All Hamiltonians considered in this thesis consist of a sum of two-body nearest-neighbor interaction terms

$$H = \sum_{\langle i,j \rangle} H_{ij}. \quad (2.4)$$

The advantage of tensor network states is that the application of a local gate—i.e. some operator that only involves a few sites that are close to each other—only affects a small set of tensors rather than the state as a whole. Making use of the
fact that the Hamiltonian is of the form of Eq. (2.4), it possible to do numerical optimization on large systems, or even in the thermodynamic limit, by sequentially updating the tensor network state piece by piece.

In this thesis, we employ two types of energy minimization schemes. The first involves projecting (possibly randomly initialized) states onto the ground state by means of imaginary time evolution, which boils down to applying a sequence of gates that minimize bond energies. The second optimizes the local tensors individually by computing how to adjust the local tensors—one after the other—such that the total energy goes down.

2.5.1 Imaginary time evolution

Given an iPEPS $|\psi\rangle$ with a predefined unit cell, we evolve $|\psi\rangle$ in imaginary time by applying $\exp(-\beta H)$ for large enough $\beta$ to project it onto the ground state. Using a second order Suzuki-Trotter expansion, the imaginary time evolution operator $\exp(-\beta H)$ can be decomposed into a sequence of gates at the cost of a controllable Trotter error. For a finite system, this works as follows.

To begin with, for $\tau \in \mathbb{C}$, it is straightforward to check that the exponent of a sum of non-commuting operators $\sum_{i=1}^{k} \tau O_i$ equals a product of exponentials up to a correction term that is third order in $\tau$:

$$
\exp \left( \sum_{i=1}^{k} \tau O_i \right) = \prod_{i=1}^{k} \exp \left( \frac{\tau}{2} O_i \right) \prod_{i=k}^{1} \exp \left( \frac{\tau}{2} O_i \right) + \mathcal{O}(\tau^3).
$$

Note that in the second product, the order of the operators is reversed.

Next, we split the imaginary time evolution operator $\exp(-\beta H)$ into a sequence of small time-step evolution operators $[\exp(-\tau H)]^M$, where $\tau = \beta/M$ for some large integer $M$. Using Eq. (2.5), we obtain

$$
\exp (-\beta H) = \left[ \exp \left( -\tau \sum_{(i,j)} H_{ij} \right) \right]^M = \left\{ \left[ \prod_{(i,j)} \exp \left( -\frac{\tau}{2} H_{ij} \right) \right] \left[ \prod_{\text{rev}(i,j)} \exp \left( -\frac{\tau}{2} H_{ij} \right) \right] + \mathcal{O}(\tau^3) \right\}^M
$$

$$
= \left\{ \left[ \prod_{(i,j)} G_{ij} \right] \left[ \prod_{\text{rev}(i,j)} G_{ij} \right] \right\}^M + \mathcal{O}(\tau^2),
$$

where “rev” means reversed order. As a consequence, in the limit of $M \to \infty$, and therefore $\tau \to 0$, the imaginary time evolution operator $\exp(-\beta H)$ can be written
as a sequence of local gates $G_{ij} = \exp(-\tau H_{ij}/2)$. In practice, we shall choose $M$ to be large enough such that, within the degree of accuracy required, we can safely neglect the $O(\tau^2)$ term and approximate the imaginary time evolution operator $\exp(-\beta H)$ solely by the products of gates $G_{ij}$ on the bottom line above.

When evolving an iPEPS in imaginary time, the product over all local gates goes over all bonds in the unit cell. In this case, updating a single bond in the unit cell corresponds to updating all equivalent bonds in the infinite system simultaneously.

Applying a local gate to a bond evolves the bond a little bit in imaginary time while simultaneously increasing the bond dimension. To make sure that the end result of the minimization process is again an iPEPS of bond dimension $D$, after applying a single gate the bond dimension of the updated bond will be truncated back to $D$. This can be done using the simple [117] or the full [111, 125] update algorithm. Both algorithms will be explained below for the square lattice models considered, which includes both the spin-1 BBH square lattice model as well as all SSL models that appear in this thesis. The spin-1 BBH triangular lattice model has been studied using an altered version of the square lattice algorithms, the details of which will also be presented below.

### Simple update

Tensor network states have a redundancy in their descriptions, in the sense that various local tensors can be combined to form the same state in the Hilbert space. More precisely, on each bond we can artificially insert an identity matrix $I = MM^{-1}$, and merge $M$ with the local tensor on one side, and $M^{-1}$ with the other, changing both local tensors in the process while keeping the total (contracted) physical state the same. For MPS with open boundary conditions, it possible to use this gauge invariance to put the local tensors into so-called canonical form [126]. The canonical form is obtained by a sequence of singular value decompositions, starting from both ends of the chain, and then moving towards a selected point in the middle from both sides.

As a matter of fact, it is also possible to store the singular values of each singular value decomposition in diagonal matrices, while moving from one end of the chain to the other. This procedure results in a tensor network state—invented by Vidal [127], see Fig. 2.14—that contains diagonal matrices in between each adjacent pair of site-tensors containing the $D$ largest Schmidt weights corresponding to a cut at the bond in question. The crucial ingredient that allows this procedure to work is that, in one dimension, cutting a single bond corresponds to splitting the Hilbert space into two, which means that to each bond we can associate a matrix of Schmidt weights.
2.5. Energy minimization

Figure 2.14: Starting from one end of the chain and moving to the other by applying a sequence of singular value decompositions, and storing the $D$ largest singular values at each step in a diagonal matrix, an MPS can be cast into a form (bottom figure) with diagonal matrices (orange diamonds) on each bond containing the $D$ largest Schmidt coefficients corresponding to cutting the bond in question. Note that this procedure requires that, e.g., when starting from the left, the state has to be right-normalized [87], or vice versa.

Employing imaginary time evolution in one-dimensional systems can be done very efficiently if the MPS is kept in canonical form during the optimization process. Indeed, truncating the bond dimension back to $D$ after applying an imaginary time evolution gate amounts to keeping the $D$ largest Schmidt values corresponding to the cut in question—a procedure that yields the best rank-$D$ approximation to the time-evolved state in the 2-norm. This idea works both for finite [128] and infinite [114] one-dimensional systems, and goes by the name (infinite) time-evolving block decimation, or (i)TEBD for short.

The simple update algorithm [117] is the two-dimensional analogue of (i)TEBD. Similar to the one-dimensional case, each bond gets decorated with a diagonal matrix of weights, depicted by the orange diamonds in Fig. 2.15(a). The main idea underlying the simple update is that these weights to a certain extent communicate the influence of the tensors surrounding the bond in question. After applying an imaginary time evolution gate $G$ to a given bond and contracting the gate with the site tensors adjacent to it, the simple update truncates the updated bond back to $D$ by keeping $D$ largest values of a singular value decomposition involving only the tensors and accompanying weights connected to the to-be-truncated bond; see Fig. 2.15(b). The extra step of splitting off the green tensors in part (a) of Fig. 2.15 only serves as a way to speed up the algorithm, because it decreases the size of the matrices of which a singular value decomposition has to be computed.

Contrary to the one-dimensional case, the truncation step in the simple update does not yield the best rank $D$ approximation to the time-evolved state, because cutting a single bond does not split the Hilbert space into two. This is a conse-

23 For infinite one-dimensional systems, applying the local gates actually destroys the canonical form. However, for a small enough time step $\tau$ the gates are close enough to the identity matrix to make the errors due to the deviation from the canonical form manageable, resulting in a truncation that is optimal enough to make the algorithm work.
2. Tensor networks

Figure 2.15: A simple update step applied to a single bond. (a) Absorb the weights into the site tensors and use singular value decompositions to split of $p$ and $q$ to speed up the next steps (decreasing the number of coefficients from $dD^4$ for the grey tensor to $d^2D^2$ for the green one). (b) Apply the time evolution gate $G$ to $p$ and $q$, and then split the whole using another singular value decomposition keeping only the $D$ largest singular values. (c) Reabsorb the updated $p_{\text{new}}$ and $q_{\text{new}}$ back into the red tensors, then reintroduce the outer weights by multiplying each leg with the appropriate $\lambda^{-1}\lambda$ (where $\lambda$ is a diagonal weight matrix) and absorbing $\lambda^{-1}$ into the site tensor.

sequence of the fact that a square lattice PEPS contains loops\textsuperscript{24}. Regardless, the simple update, albeit being somewhat ad hoc, is computationally very efficient and yields reasonably accurate results in many cases.

Full update

In the full update scheme \cite{111}, finding the optimal (truncated) bond-dimension-$D$ state $|\psi_{\text{tr}}\rangle$ that is closest to the imaginary time evolved state $|\psi_{te}\rangle = G|\psi\rangle$ is done by variationally minimizing the squared distance $||\psi_{\text{tr}} - \psi_{te}||^2$; see Fig 2.16.

\textsuperscript{24}If the tensor network had a tree-like structure \cite{100}, then cutting the bond in question would have cut the system into two, and the singular values would have been equal to the Schmidt weights if the state is in canonical form. For this reason, tree tensor networks allow for accurate and efficient imaginary time evolution. However, their network structure does limit their usage.
2.5. Energy minimization

This requires computing the environment at every step, unless the fast full update \[125\] is used, which speeds up the algorithm by recycling previous environments. Because of the frequent need to update the environment tensors, the full update is computationally more costly than the simple update. However, the full update does not suffer from uncontrollable approximations, and therefore, in the \( D \to \infty \) limit, the full update imaginary time evolved state should in principle always converge to the ground state with increasing imaginary time.

A single full update step applied to an initial state \( |\psi\rangle \) proceeds as follows. After splitting off the \( p \) and \( q \) tensors as with the simple update to speed up the algorithm, the imaginary time evolution gate \( G \) is applied, yielding \( |\psi_{te}(p, q)\rangle \), see Fig. 2.16(a). We then use the environment tensors of the initial state to construct the approximate effective environment, shown in Fig. 2.16(b), that will be used to compute the norm difference between the time evolved state and its (truncated) bond dimension \( D \) approximation \( |\psi_{tr}(p', q')\rangle \). An initial guess \( (p_0, q_0) \) for \( (p', q') \) is obtained by doing a singular value decomposition—Fig. 2.16(c). Given this initial guess, the tensors \( p' \) and \( q' \) are then sequentially improved, keeping one constant as the other gets optimized, by repeatedly solving

\[
0 = \frac{\partial}{\partial x'} ||\psi_{tr} - \psi_{te}||^2 = \frac{\partial}{\partial x'} \left( - \langle \psi_{tr} | \psi_{te} \rangle + \langle \psi_{tr} | \psi_{tr} \rangle \right) \tag{2.6}
\]

for \( x = p' \) or \( x = q' \) respectively\[25\]. Note that Eq. (2.6) is just a linear system of equations, see Fig. 2.16(d)\[26\]. This sequential improvement continues until the norm difference converges, after which the resulting \( p'_{\text{final}} \) and \( q'_{\text{final}} \) are re-absorbed into the red tensors in Fig. 2.16(a), yielding the updated site tensors adjacent to the updated bond. Finally, after replacing the older site tensors in the unit cell with the updates ones, the environment tensors are recomputed, and then the the whole procedure repeats for the next bond.

Typically, when minimizing the energy through imaginary time evolution, we first use the simple update to obtain a reasonable approximation of the ground state. Next, depending on the required accuracy, possible additional full update steps are executed to further lower the energy of the approximate ground state. To monitor convergence, we track the energy as a function of imaginary time, and stop when the changes in energy are smaller than the accuracy required for several consecutive imaginary time steps.

---

\[25\] In order to find the points of some smooth function \( f : \mathbb{C}^n \to \mathbb{R} \) for which the total derivative is zero, it is enough to solve \( \partial f / \partial z^*_i = 0 \) for all complex conjugate variables \( z^*_i \), which can be thought of as independent from the ordinary complex variables \( z_i \). For example, minimizing the function \( f(z, z^*) = zz^* = x^2 + y^2 \), requires only that we solve \( 0 = \partial f / \partial z^* = z \), the latter of which gives the correct \( x = y = 0 \).

\[26\] Differentiating a tensor network with respect to a single tensor amounts to removing the tensor in question from the network, because its coefficients appear linearly in the network that is to be differentiated.
2. Tensor networks

Figure 2.16: The full update applied to a single bond. (a) Split of the p and q parts as with the simple update, and apply the imaginary time evolution gate $G$ to obtain the time evolved state $|\psi_{te}(p,q)\rangle$. (b) Construct the effective environment of $p$ and $q$ and their complex conjugates. (c) Obtain initial $p'_0$ and $q'_0$ for the truncated state $|\psi_{tr}(p',q')\rangle$. (d) Iteratively optimize $p'$ and $q'$ by solving $Mp' = c$ and $Nq' = d$ repeatedly until convergence is reached and the best bond-dimension-$D$ approximation $|\psi_{tr}(p'_{\text{final}},q'_{\text{final}})\rangle$ to $|\psi_{te}(p,q)\rangle$ is obtained.
2.5. Energy minimization

Triangular lattice algorithms

The triangular lattice simple update algorithm is a modified version of the simple update method for square lattice Hamiltonians with an additional next-nearest neighbor interaction, explained in Ref. [129]. The difference lies in the fact that, instead of truncating the bond dimension back to $D$ immediately after applying a single imaginary time evolution gate, the triangular lattice algorithm simultaneously applies a horizontal, vertical, and diagonal evolution gate to three sites forming a triangle, and only afterwards truncates the bond dimensions back to $D$.

The triangular lattice full update method used for this thesis is a variant of the next-nearest neighbor method from Ref. [70] (see also Ref. [130]). After time-evolving a given initial state $|\psi\rangle$ a small step in imaginary time by means of the evolution gate\footnote{As mentioned in the previous paragraph, the imaginary time evolution gate $G$ used on the triangular lattice is actually a product of three single-bond gates (corresponding to a horizontal, vertical and diagonal bond that together form a closed triangle), rather than just a single-bond gate that is used on the square lattice.} $G$—increasing the bond dimension—the optimal time-evolved iPEPS $|\psi_{tr}\rangle$ with (truncated) bond dimension $D$ is obtained by minimizing the norm distance $||\psi_{te} - \psi_{tr}||^2$. As explained above, in the regular full update this is done by iteratively minimizing over two tensors $(p, q)$ on a bond until the cost function $||\psi_{te} - \psi_{tr}||^2$ has converged. In the presence of an additional diagonal interaction, we need to optimize over four of these tensors, two on a horizontal bond $(p_h, q_h)$ and two on a vertical bond $(p_v, q_v)$, respectively. This is done by performing an outer loop where we switch between the horizontal and vertical pairs of tensors, and an inner loop where we iteratively optimize over the tensors within each individual pair.

2.5.2 Variational update algorithm

Instead of projecting a randomly initialized state onto the ground state by means of imaginary time evolution, it is also possible to minimize the energy directly. The variational update algorithm\footnote{Contrary to the case of iPEPS—discussed below—for which each tensor in the unit cell occurs infinitely many times on the lattice.} [131] optimizes one tensor after the next, keeping all tensors but the one getting optimized fixed during a single optimization step.

Concretely, suppose that $|\psi\rangle \in \mathcal{H}$ is some tensor network state, and let\footnote{Concretely, suppose that $|\psi\rangle \in \mathcal{H}$ is some tensor network state, and let $|\psi\rangle \in \mathcal{H}$ be some tensor network state, and let} assume for now the system is finite, and each tensor occurs only once\footnote{Contrary to the case of iPEPS—discussed below—for which each tensor in the unit cell occurs infinitely many times on the lattice.} in $|\psi\rangle$. Furthermore, let $A$ be some local tensor of $|\psi\rangle$ that we want to optimize. That is, we want to find

$$\text{argmin}_A \frac{\langle \psi(A) | H | \psi(A) \rangle}{\langle \psi(A) | \psi(A) \rangle}.$$ \hspace{1cm} (2.7)

Since the expression that is to be minimized is smooth in $A$ (outside of the set of points for which the norm vanishes, which are not physically relevant), its
minimum will be attained at a point where the derivative vanishes. Thus, in order to find a solution to Eq. (2.7), we need to solve
\[
\frac{\partial}{\partial A^*} \left( \frac{\langle \psi(A) | H | \psi(A) \rangle}{\langle \psi(A) | \psi(A) \rangle} \right) = 0.
\] (2.8)

Using that each tensor coefficient of $A^*$ occurs linearly in both the numerator and the denominator, Eq. (2.8) reduces to the following linearized eigenvalue problem
\[
H A = E N A,
\] (2.9)
where $E = \langle \psi | H | \psi \rangle / \langle \psi | \psi \rangle$, and $H$ and $N$ are the tensor networks obtained by removing the local tensors $A^*$ and $A$ from the numerator $\langle \psi(A, A^*) | H | \psi(A, A^*) \rangle$ and the denominator $\langle \psi(A, A^*) | \psi(A, A^*) \rangle$ respectively, interpreted as matrices that act as linear maps from the $A$-space to the $A^*$-space. It should be noted that obtaining the matrix $H$ is not a trivial numerical task, as it includes the sum of all Hamiltonian nearest-neighbor terms.

When using the variational update algorithm for iPEPS—made up of a unit cell of tensors—the local tensor $A$ being optimized is some fixed tensor in this unit cell. By construction, $A$ occurs infinitely many times in the complete state $|\psi\rangle$, and it is not possible to optimize all those $A$’s simultaneously. Instead, we will optimize only one, keeping all the other $A$’s (that occur in the effective environment of the $A$ that is to be updated) fixed, and only afterwards replace the old $A$ with the new $A$ everywhere (modulo some important details discussed below).

In case of iPEPS, computing $H$ requires contracting an infinite network, for which, in addition to the regular corner and transfer matrices, additional matrices need to be introduced that contain the local Hamiltonian terms (see Fig. 2.17). The details of this procedure, which in spirit is similar to the ordinary CTM procedure described in Section 2.4, can be found in Ref. [131].

After computing $H$ and $N$ and solving Eq. (2.9), the easiest thing to do would be to simply replace the old $A$ with the solution $\tilde{A}$ to Eq. (2.9) in the unit cell of $|\psi\rangle$, and move on to the next tensor. However, because the tensor $A$ being optimized occurs all over the infinite lattice, $H$ and $N$ both depend on it. Consequently, simply replacing $A$ by $\tilde{A}$ in unit cell of $|\psi\rangle$ does not have to result in the energy being lowered. Rather, we try different tensors
\[
A'(\lambda) = \tilde{A} \sin(\lambda \pi) - A \cos(\lambda \pi)
\]
that are linear combinations of the old $A$ and the $\tilde{A}$ that is a solution to Eq. (2.9), and optimize the energy
\[
E(\lambda) = \frac{\langle \psi(A'(\lambda)) | H | \psi(A'(\lambda)) \rangle}{\langle \psi(A'(\lambda)) | \psi(A'(\lambda)) \rangle}
\]
2.5. Energy minimization

\[
N = \begin{array}{c}
\text{Diagram}
\end{array}
\]

\[
H = \begin{array}{c}
\text{Diagram}
\end{array} + \begin{array}{c}
\text{Diagram}
\end{array} + \begin{array}{c}
\text{Diagram}
\end{array} + \begin{array}{c}
\text{Diagram}
\end{array} + \begin{array}{c}
\text{Diagram}
\end{array}
\]

\[
+ \quad \text{rotated analogues}
\]

Figure 2.17: The \(N\) matrix simply consists of the environments tensors of the local tensor \(A\). The \(H\) matrix, on the other hand, consists of an infinite sum over all Hamiltonian two-body terms. All these terms can be combined into additional corner and transfer matrices, signified by the thick gray edges. The first two terms after the equality sign after \(H\) contain all two-body terms involving sites that are either both contained within \(C_1\) or both contained within \(T_1\). The third term represents the two-body interaction term between the local tensor \(A\) and one site above it. The fourth and fifth terms represent the two-body terms that contain one site in \(C_1\), and on site in either \(T_1\) or \(T_4\). The “rotated analogues” are the remaining 15 equivalent diagrams obtained by rotating the diagrams over multiples of \(\pi/2\).

as a function of \(\lambda\) until we find a state \(|\psi(A'(|\lambda|))\rangle\) that has a lower energy than \(|\psi(A)\rangle\), before moving to the next tensor. The above steps are repeated for all tensors in the unit cell, forming a single update cycle. Each cycle is repeated over and over until the changes in energy between cycles become smaller than the desired accuracy.

Computing \(H\) is numerically the most intense part of the variational update algorithm, because it requires approximating an infinite sum (over all Hamiltonian terms) of contracted infinite tensor networks. As a consequence, the variational update algorithm is more costly than the full update. However, the benchmarks—see for example Sections 4.4 and 4.A.2—ran so far show that the variational update is the most accurate of all methods used in this thesis. Additionally, it is less sensitive to the choice of initial state, which can be an blessing when the simple and full update get stuck in some kind of metastable state, but also an obstacle when we want the simulation to stay within a predefined phase in order to determine the location of a phase transition.
2. Tensor networks

2.6 Simulation practicalities

2.6.1 Network structure

We shall use the square lattice iPEPS in all our computations, even when the model itself is not defined on a square lattice. Specifically, for the triangular lattice spin-1 BBH model, the lattice structure will be encoded in the Hamiltonian rather than in the iPEPS itself. Two particles that are neighbors on the triangular lattice, but whose corresponding local tensors are not directly connected in the iPEPS, communicate with one another indirectly through neighboring local tensors.

2.6.2 Initial states

Because the update algorithms used in this thesis consist of small sequential improvements, they are susceptible to initial conditions. Additionally, when evolving randomly initialized states in imaginary time, it is possible that the random state has a very small overlap with the true ground state, leading to a very slowly converging algorithm. In order to circumvent this problem, we always initialize several (around ten) initial states, evolve all of them a little bit in imaginary time, and then pick the lowest energy state to continue the simulation with.

Another important phenomena related to the choice of initial state is the following. By loading from a state in a particular phase just across the transition point of a first order transition, the state will remain in that particular phase at least for some time during the energy optimization process. This effect is known as hysteresis.

At the intersection of two different phases, by initializing simulations from deep in either phase, we obtain the energy per site of states in both phases for several values of the coupling parameter that drives the transition. The phase transition is then determined to lie at the point where the linearly interpolated energies per site (as a function of the coupling parameter) intersect.

Second order transitions are much more difficult to investigate, because the hysteresis effect is non-existent and the correlation length diverges at the transition point. As a consequence of the latter, finite-$D$ effects emerge similar to finite size effects for simulations on finite systems close to second order transitions, in the sense that the order parameter of the ordered phase gets suppressed with increasing $D$. The location of the transition point is then determined by observing where the $D \to \infty$ extrapolated value of the order parameter vanishes.

The above situations will be discussed in more detail later on when applicable to the models investigated in this thesis.
2.6. Simulation practicalities

2.6.3 Bond dimension extrapolation

Both the bilinear-biquadratic Heisenberg and Shastry-Sutherland models investigated in the chapters to come possess continuous symmetries, so they are gapless whenever their ground states breaks the continuous symmetry. Therefore, unless the ground state is a product state (which can be described exactly by a $D = 1$ iPEPS), and if the required accuracy demands it, we will extrapolate $D \rightarrow \infty$ to ensure that we obtain precise expectation values.

There are several ways of doing $D \rightarrow \infty$ extrapolations. Even though they can vary somewhat in their approaches, the results should be compatible within the error margins they provide. We employ two types of extrapolation, the more intuitive $1/D$ extrapolation, and extrapolation in truncation error [132], the latter of which provides a measure of the severeness of the truncation back to $D$ during the optimization process. Extrapolating the truncation error to zero corresponds to extrapolating the bond dimension to infinity.

Equipped with all ingredients necessary to be able to investigate challenging strongly-correlated quantum magnets, let us proceed with a thorough investigation of some actual physical systems. First, we shall turn our attention to the first class of models of interest, the spin-1 bilinear-biquadratic Heisenberg models on the square and triangular lattices, and put the algorithms discussed in this chapter into effect.