Cluster Expansion of Thermal States using Tensor Networks

David Devoogdt
Student number: 01608249

Supervisors: Dr. Laurens Vanderstraeten, Prof. Frank Verstraete
Counsellor: Bram Vanhecke

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Engineering Physics

Academic year 2020-2021

Cluster Expansion of Thermal States using Tensor Networks

David Devoogdt
Student number: 01608249

Supervisors: Dr. Laurens Vanderstraeten, Prof. Frank Verstraete
Counsellor: Bram Vanhecke

GHENT
UNIVERSITY

Foreword

During the preliminary first meeting, the quantum group promised me if I'd do a master's thesis in there, I'd get a challenging subject tailored to my interests. I can say that this has absolutely been the case. The quantum group gave me a taste of real research, and introduced me to many exciting (and for someone with an engineering background sometimes unknown) topics during the Friday "lunch talks". While this is an individual task, I certainly was not on my own. Firstly I want to thank my supervisors Laurens Vanderstraeten, Frank Verstraete and my counsellor Bram Vanhecke for the interesting discussions and feedback given during the year. I also want to thank my friends and fellow students with who I spend many memorable nights in Ghent. They made my students years truly unforgettable. Also BEST Ghent deserves a mention, because despite the Covid-19 pandemic, my board year is something I won't forget. But most importantly, I want to thank my parents for giving me the chance to study in the first place.

David Devoogdt Ghent, 29$^{\text{th}}$ May 2021

Permission of use on loan

The author(s) gives (give) permission to make this master dissertation available
for consultation and to copy parts of this master dissertation for personal use.
In all cases of other use, the copyright terms have to be respected, in particular
with regard to the obligation to state explicitly the source when quoting results
from this master dissertation.

David Devoogdt, May 2021

Cluster Expansion of Thermal States using Tensor Networks

David Devoogdt [1]

Supervisors: Dr. Laurens Vanderstraeten, Prof. Frank Verstraete
Counsellors: Bram Vanhecke

Faculty of Engineering and Architecture Ghent University

Abstract

Simulating many-body quantum systems is one of the challenges faced by modern physics. One particular problem is constructing the operator $e^{-\beta \hat{H}}$, which is used to compute the partition function of a quantum models. This is needed to make predictions about the macroscopic behaviour of quantum systems. Of particular interest are the phase transitions. This dissertation introduces the rich field of Tensor Networks as a method to perform these calculations. The physics of strongly correlated matter is discussed. The main contribution of this work, the novel cluster expansions, are described, together with the computational tools to solve the equations efficiently. The results are promising. Large imaginary time steps can be taken. Calculating phase transitions of the Transverse Field Ising model in 2D produces results that can challenge recent developments in literature.

Keywords

Tensor networks, Thermal States, Cluster Expansions, Phase transitions, Transverse Field Ising model

---

[1]Student number: 01608249

Cluster Expansion of Thermal States using Tensor Networks

David Devoogdt

Academic year 2020-2021

Abstract

Simulating many-body quantum systems is one of the challenges faced by modern physics. One particular problem is constructing the operator $e^{-\beta \hat{H}}$, which is used to compute the partition function of a quantum models. This is needed to make predictions about the macroscopic behaviour of quantum systems. Of particular interest are the phase transitions. This dissertation introduces the rich field of Tensor Networks as a method to perform these calculations. The physics of strongly correlated matter is discussed. The main contribution of this work, the novel cluster expansions, are described, together with the computational tools to solve the equations efficiently. The results are promising. Large imaginary time steps can be taken. Calculating phase transitions of the Transverse Field Ising model in 2D produces results that can challenge recent developments in literature.

1 Overview

Understanding the many-body quantum problem remains a challenge. Strongly correlated matter has many interesting phases which are not yet understood, including high-T superconductors, topological ordered phases and quantum spin liquids [2]. One method to simulate these materials numerically are tensor networks (TN). This work introduces a new TN method to calculate the exponential of a Hamiltonian with local interactions. This allows to evolve a quantum state in time and calculate the macroscopic properties at a finite temperature. As an example, the new method is used to calculate phase transitions of the Transverse Field Ising (TFI) model.

2 Tensor Networks

2.1 MPS

A general quantum state is given by

$$|\Psi\rangle = \sum_{i_1 i_2 \cdots i_n} C^{i_1 i_2 \cdots i_n} |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle. \quad (1)$$

The tensor C needs d^n numbers to describe the full wave function, where d is the dimension of the local Hilbert space. Luckily, the relevant low energy states for many systems found in nature obey the so called 'area law', which only make up a very small corner of the full Hilbert space. In uniform matrix prodcut states (MPS), the tensor $C^{i_1 i_2 \cdots i_n}$ is subdivided into the product of n tensors C and a matrix M that contains the boundary conditions

$$C^{i_1 i_2 \cdots i_n} = Tr(C^{i_1} C^{i_2} \cdots C^{i_n} M). \quad (2)$$

If M is the identity matrix, the chain is closed cyclically. Tensor networks are typically denoted in their graphical form (see table 1 for examples). External lines denote free indices, connected lines implies a summation over the shared index, analogous to matrix multiplication. Equation (2) is in this graphical notation becomes
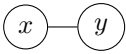

$$(3)$$

An MPS has 2 dimension, the physical dimension of the particles d and the dimension χ of the bonds between the tensors. The cluster expansion will rely

Table 1: Examples of graphical notation.

conventional	Einstein	tensor notation
$\vec{x}$	x_α	
M	$M_{\alpha\beta}$	
$\vec{x}\cdot\vec{y}$	$x_\alpha y_\alpha$	

on "virtual levels". This is the division of the MPS in blocks, analogous to dividing a matrix into block matrices. Every virtual level has its own associated dimension. The dimensions sum up to the total bond dimension χ.

2.2 MPO

A matrix procuct operator (MPO), is similar to an MPS but has 2 physical legs i and j. The following compact notation is used in this paper

$$O^{00} = \quad \begin{matrix} i \\ 0 \quad \bigcirc \quad 0 \\ j \end{matrix} \quad = \bigcirc . \tag{4}$$

This is the MPO with virtual level 0 and physical indices i and j, which will both be omitted. Non-zero virtual indices are shown. Summation over shared virtual index 1 on 2 neighbouring sites is denoted as

$$O^{01}O^{10} = \bigcirc \!\!-\!\!\overset{1}{}\!\!-\!\! \bigcirc , \tag{5}$$

while contraction over all possible virtual levels on 3 sites is denoted by

$$\bigcirc \!\!-\!\!-\!\! \bigcirc \!\!-\!\!-\!\! \bigcirc . \tag{6}$$

3 Cluster Expansion

The novel method to construct the operator $e^{-\beta\hat{H}}$ with cluster expansions was first introduced in [3].

The goal is capture the exponential of a Hamiltonian operator $\hat{H}$ of the form

$$\hat{H} = -J \left(\sum_{\langle ij \rangle} H_2^i H_2^j + \sum_i H_1^i \right) . \tag{7}$$

This Hamiltonian consists of 1 and 2 site operators. The first summation $\langle ij \rangle$ runs over all neighbouring sites. The precise form of the Hamiltonian is not important. The notation for the contraction of the tensor network will also be used to denote the Hamiltonian evaluated on the given geometry

$$
\begin{aligned}
H \left(\bigcirc \!\!-\!\! \bigcirc \!\!-\!\! \bigcirc \right) = & H_1 \otimes 1 \quad \otimes 1 \\
& + 1 \quad \otimes H_1 \otimes 1 \\
& + 1 \quad \otimes 1 \quad \otimes H_1 \\
& + H_2 \otimes H_2 \otimes 1 \\
& + 1 \quad \otimes H_2 \otimes H_2 .
\end{aligned} \tag{8}
$$

3.1 Idea

The main idea is to make an extensive expansion by adding blocks which solve the model exactly on a local patch. Crucially, the expansion is not in the inverse temperature β but in the size of the patches. The local patches are separated by a virtual level 0 bond. To make this somewhat more precise, the first steps of the expansion are shown here. The smallest patch, i.e. 1 site, encodes the exponential of that Hamiltonian

$$\bigcirc = \exp\left(-\beta H(\bigcirc)\right) . \tag{9}$$

If there were no 2 site interactions, this already captures the full diagonalisation. The next step is to introduce 2 site interactions, where the one site interactions are subtracted from the diagonalised Hamiltonian.

$$
\begin{aligned}
\bigcirc \!\!-\!\!\overset{1}{}\!\!-\!\! \bigcirc &= \exp -\beta H(\bigcirc \!\!-\!\! \bigcirc) \\
&- \bigcirc \!\!-\!\!\overset{0}{}\!\!-\!\! \bigcirc
\end{aligned} \tag{10}
$$

Contraction of larger network lead to many terms, such as

$$\bigcirc \!\!-\!\!\overset{1}{}\!\!-\!\! \bigcirc \!\!-\!\!\overset{0}{}\!\!-\!\! \bigcirc \!\!-\!\!\overset{0}{}\!\!-\!\! \bigcirc \!\!-\!\!\overset{0}{}\!\!-\!\! \bigcirc \!\!-\!\!\overset{1}{}\!\!-\!\! \bigcirc \!\!-\!\!\overset{0}{}\!\!-\!\! \bigcirc . \tag{11}$$

The beauty of this lays in the fact that disconnected regions(regions separated by level 0) combine in exactly the right way to capture all the terms appearing in the series expansion of the exact tensor exponential with as largest patch 2. In other words it is exact up to order 2 in β [3]. Only the terms of the exponential which acts on 3 or more neighbouring sites at once, are not accounted for.

Notice that in eq. (10), 2 new blocks are introduced. The dimension of virtual level 1 needs to be d^2, with d the dimension of physical level. Although the types already differ in the next step, one more blocks is shown to make the notation clear:

$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc = \exp-\beta H(\bigcirc\!-\!\!-\!\bigcirc\!-\!\!-\!\bigcirc)
$$
$$
-\bigcirc\!\overset{0}{-}\!\bigcirc\!\overset{0}{-}\!\bigcirc
$$
$$
-\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{0}{-}\!\bigcirc
$$
$$
-\bigcirc\!\overset{0}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc \tag{12}
$$
$$
= \exp-\beta H(\bigcirc\!-\!\!-\!\bigcirc\!-\!\!-\!\bigcirc)
$$
$$
-\bigcirc\!-\!\!-\!\bigcirc\!-\!\!-\!\bigcirc .
$$

This is called a cluster expansion of order 3, because there are 3 connected sites solved exactly. The right-hand side of eq. (12) can be omitted, as it is just evaluating the exponentiated Hamiltonian on the same geometry as the left-hand side and subtracting all possible contractions of the blocks which were added previously. This very compact notation will be able to capture the essence of the different constructions. Because it is important for the remainder of the chapter, it is stressed that for an equation similar to

$$
\boxed{\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc}, \tag{13}
$$

the right-hand side of eq. (12) is implied. This fully defines one new block as in eq. (12), or the contraction of 2 blocks as in eq. (10). In the following section, different types will be discussed. For every geometry, one or multiple new blocks are defined.

3.2 1D

3.2.1 Type A

The first few blocks in the cluster expansion are

$$
\bigcirc \tag{14a}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc \tag{14b}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc \tag{14c}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc \tag{14d}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc . \tag{14e}
$$

Virtual level l needs a bond dimension of d^{2l} to solve the equations exactly. The introduction of O^{nn} blocks could lead to long chains in the contraction, e.g. $\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{1}{-}\!\bigcirc$. The results of this will be discussed in section 4.1.

3.2.2 Type E

To remedy this behaviour, type E only has these blocks

$$
\bigcirc \tag{15a}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc \tag{15b}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{1'}{-}\!\bigcirc \tag{15c}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{1'}{-}\!\bigcirc \tag{15d}
$$
$$
\bigcirc\!\overset{1}{-}\!\bigcirc\!\overset{2}{-}\!\bigcirc\!\overset{2'}{-}\!\bigcirc\!\overset{1'}{-}\!\bigcirc . \tag{15e}
$$

The bond dimension χ is twice as large, because for every virtual level there is also a primed level. With these blocks, it is impossible to make a patch longer than the order of the largest chain. This generalises well to higher dimensions.

3.2.3 Type F

Both type A and F have potentially ill conditioned inverses. The blocks of type F are

$$
\bigcirc \tag{16a}
$$

$$
\bigcirc \overset{1'}{-} \bigcirc + \bigcirc \overset{1'}{-} \bigcirc \tag{16b}
$$

$$
\bigcirc \overset{1}{-} \bigcirc \overset{1}{-} \bigcirc \tag{16c}
$$

$$
\bigcirc \overset{1}{-} \bigcirc \overset{2}{-} \bigcirc \overset{1}{-} \bigcirc +
$$

$$
\bigcirc \overset{1}{-} \bigcirc \overset{2'}{-} \bigcirc \overset{1}{-} \bigcirc \tag{16d}
$$

$$
\bigcirc \overset{1}{-} \bigcirc \overset{2}{-} \bigcirc \overset{2}{-} \bigcirc \overset{1}{-} \bigcirc . \tag{16e}
$$

The blocks O^{nn+1} are chosen unitary up to a constant factor. The primed blocks solve the chains of even order. This requires twice the bond dimension of type A, but is guaranteed to have well conditioned inverses.

3.3 2D

In hindsight of the results, the construction in 2D is a generalisation of type A. The linear blocks, for which the defining equation is described by a tree graph, will resemble the blocks previously introduced. The non-linear blocks are used to account for loops.

3.3.1 Linear blocks

The extension of an MPO to 2D is called a PEPO (Projected Entangled Pair Operator). These PEPO's are graphically depicted in this work by the same symbol

$$
O^{0000} = \bigcirc = \overset{0}{\underset{j\,0}{\bigcirc}} \overset{i}{-} 0 \quad . \tag{17}
$$

The construction starts off as

$$
\bigcirc \overset{1}{-} \bigcirc \qquad 1 \overset{}{\mid} \bigcirc . \tag{18}
$$

For the order 3 blocks, 6 different options are possible. They are

$$
\tag{19}
$$

and their rotations over 90 degrees. Solving all blocks, imposing rotation symmetry over 90 degrees

$$
\overset{b}{\underset{j\,d}{\overset{a}{\bigcirc}}} \overset{i}{-}c = \overset{c}{\underset{j\,a}{\overset{b}{\bigcirc}}} \overset{i}{-}d \tag{20}
$$

reduces the number of blocks that need to be solved significantly. Also, $\perp$ and $+$ blocks should be added

$$
\tag{21}
$$

From here on, it can again be generalised to higher orders. Care has to be taken that for before solving a new patch, all smaller patches that fit the geometry are solved first.

3.3.2 Nonlinear blocks

Not all finite size patches are covered with the blocks introduced in the previous section. The lowest order blocks not covered are

$$
\tag{22}
$$

$$
\tag{23}
$$

These are nonlinear equations, and will be treated separately in section 3.4. All virtual levels for solving loops are denoted by Greek letters. For $d = 2$, the bond dimension of α can be as low as 6. To connect the loops to the linear blocks, corner pieces similar to eq. (23) are used. Directly connecting virtual level 1 to the loops with $O^{\alpha\alpha 01}$ also solves the local

x

patch, but cause diverging results when 2 or more corner pieces with extensions connect. One corner can connect to 2 chains, with for each chain a maximum length equal to the order of the linear blocks. Going further is possible, but comes at an ever-increasing bond dimension cost.

3.4 Solvers

3.4.1 Linear Solver

Now that it is clear how the construction works, we focus on how to solve them numerically. Take as example $X = O^{1111}$ from eq. (21). The involved tensors are reshaped and reordered to bring it in the following standard form

$$\tag{24}$$

Here B is the exponentiated Hamiltonian on the $+$ geometry minus the contraction of all previous added blocks. ϵ is the residual error, and should be zero up to machine precision when the solver is finished. Solving this equation by inverting the matrices A^i separately results in numerical unstable results, due to the ill-conditioned blocks A^i inherent to the construction. In practice this instability starts happening at virtual level 2. Taking the pseudoinverse (with all singular values $\sigma_0 < 10^{-12}$ taken to be exactly zero) of the tensor product $A = A^1 \otimes A^2 \otimes A^3 \otimes A^4$ resolves the problem, but is computationally too expensive for long side chains. The problem is resolved by taking the SVD decomposition of each matrix $A^i = U^i \Sigma^i V^{i\dagger}$. The unitary matrices are inverted by taking the Hermitian transpose, and the matrix $\Sigma = \Sigma^1 \otimes \Sigma^2 \otimes \Sigma^3 \otimes \Sigma^4$ is pseudoinverted. This is fast because Σ is very sparse. The procedure also works for e.g. the corner pieces of eq. (23), and by extension every connected patch. When the patch contains 2 tensors such as eq. (14d), it is split with an SVD decomposition.

3.4.2 Nonlinear Solver

A nonlinear solver takes steps in the direction that lowers the residual error ϵ. This procedure can be sped up if the gradient is known. Inspection of eq. (24) shows that the derivative $\frac{\partial \epsilon_{(I^1 I^2 I^3 I^4)j}}{X_{(\alpha_1 \alpha_2 \alpha_3 \alpha_1)j}} = A_{(I^1 I^2 I^3 I^4)(\alpha_1 \alpha_2 \alpha_3 \alpha_4)}$. Or more simply put, the contraction of the network but with X removed. This is extended to more complex situation with the chain rule for derivatives. The solver handles multiple blocks which are identical up to a permutation.

3.4.3 Sequential Linear Solver

Another useful way to solve the nonlinear equations is by employing the linear solver described before, where each appearing tensor is solved individually. The algorithm keeps sweeping over all tensors until convergence is reached. It's necessary to choose a suitable step size.

4 Results

The cluster expansions are tested in 2 ways. In 1D and 2D, the error ϵ is calculated by comparing the contracted PEPO against the exact exponential. In 2D, some critical temperatures are cacluated for the TFI model.

4.1 Exact exponentials

In 1D, the error is calculated on a cyclic system of 11 sites. The results are shown in fig. 1. All the cluster expansion get better with increasing order. Type A outperforms the other 2 types by quite some margin. This conclusion is also true for the Heisenberg model and random 2 site Hamiltonians (not shown here). The results in 2D show a similar trend. The error in 2D is of the same order of magnitude. A plaquette term eq. (22) is needed, the extensions eq. (23) are optional. The results show that real time evolution ($t = -i\beta$) also performs well.
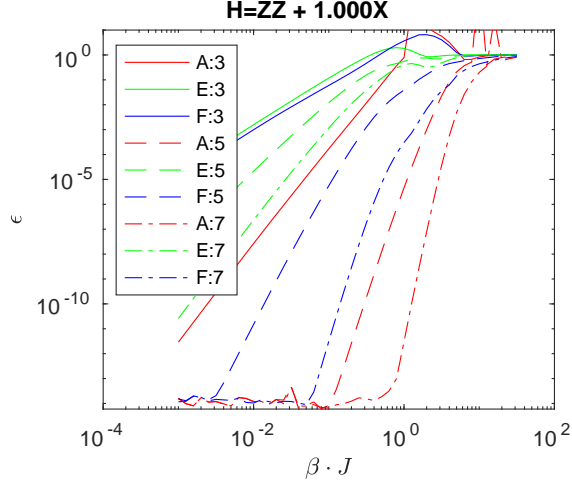
Figure 1: Comparison type A, E and F for TFI. Error evaluated on cyclic chain.

Figure 2: Data collapse for $g = 2.5$ phase transition of TFI Model. Data points are taken from $T \in [T_c - 0.08, T_c + 0.08]$.

4.2 2D TFI model

The TFI model is of the form eq. (7), with $H_1 = g\sigma^x$ and $H_2 = \sigma^z$. For low temperature, the spins tend to align forming a ferromagnet. For high T, the magnetisation disappears. The transversal field g causes the spins to align in the direction of the transversal field. A cluster expansion of order 5 with loops is used to simulate the phase transition of the TFI model at g=0 (classical Ising) and g=2.5. The simulated magnetisation $\langle m^z \rangle$ and the data collapse of m, entropy S and correlation length ξ for g=2.5 is shown in fig. 2. δ is a measure for the (inverse) size of the system. Figure 2 shows a very clear data collapse. For g=0, the fitted critical temperature is $T_c = 2.691(9)$, in agreement with Onsager's analytical solution $T_c = 2.69185$. For $g = 2.5$, the fitted value $T_c = 1.2736(6)$ agrees well with values from literature, e.g. $T_c = 1.2737(2)$ obtained with a competing tensor network technique and $T_c = 1.2737(6)$ obtained with quantum Monte Carlo. [1] Higher precision can be achieved with cluster expansions of higher order. The properties were calculated for infinite lattices using VUMPS.
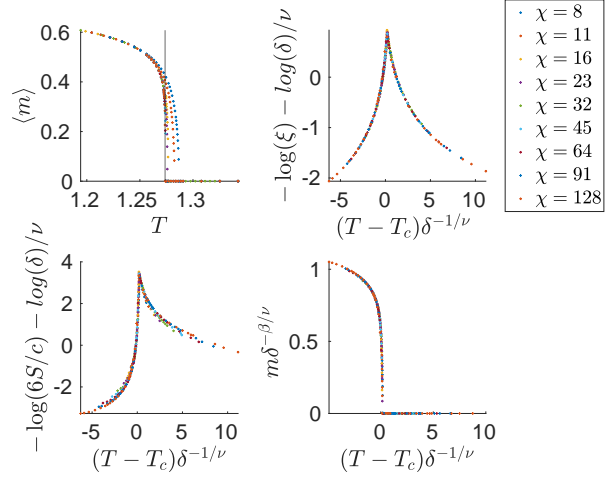
5 Conclusion and Outlook

The results agree well with exact exponentiation and with critical values from literature, proving that these cluster expansion can compete with other methods. With the current framework, it is possible to simulate the quantum critical point of the Ising model. The linear blocks and plaquette term seem to be sufficient to make an accurate cluster expansion, making the generalisation to 3D within reach.

References

[1] Piotr Czarnik and Philippe Corboz. Finite correlation length scaling with infinite projected entangled pair states at finite temperature. *Physical Review B*, 99:245107, 2019.

[2] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states, oct 2014.

[3] Bram Vanhecke, Laurens Vanderstraeten, and Frank Verstraete. Symmetric cluster expansions with tensor networks. *Physical Review A*, 103(2), 2021.

Contents

Acronyms

MPO Matrix Product Operator.

MPS Matrix Product State.

PEPO Projected Entangled Pairs Operator.

PEPS Projected Entangled Pairs State.

SVD Singular Value Decomposition.

TEBD Time Evolving Block Decimation.

TFI Transverse Field Ising.

TN Tensor Network.

VUMPS variational uniform Matrix Product State algorithm.

Chapter 1

Introduction

There is nothing new to be
discovered in physics now. All
that remains is more and more
precise measurement.

Lord Kelvin, 1900

1.1 Introduction

In 2015, there were about 5.6 million known physics papers in literature. At the
current rate, this number doubles every 18.7 years [1]. Despite this enormous
body of literature, there are a lot of things which are not completely under-
stood. Some examples include a self-consistent theory of quantum gravity, the
need for dark energy and matter in cosmology, the arrow of time, the matter-
antimatter asymmetry. There even is no interpretation of quantum mechanics
where everyone agrees upon [2]. But certainly not all open problems have to
do with 'new' physics. In many areas of physics, computing the implications of
relatively simple laws becomes exceedingly difficult for many particles. Of his-
torical importance is the classical three-body problem, describing the trajectory
of 3 gravitational bodies such as the earth, moon and sun. The general case is
not solved, despite developments over the last 300 years [3]. In reality, the real
challenge is to model the macroscopic properties of quantum many-body system
with around 10^{23} particles. This broad field is called condensed matter physics
and is important across many disciplines, including physics, chemistry, biol-
ogy and engineering. Its practitioners include those who discover and develop
new materials, those who seek to understand such materials at a fundamental
level through experiments and theoretical analysis, and those who apply the
materials to create new devices and technologies. [4] In computational chem-
istry, the many-body problem is tackled with methods which fall in one of the
following categories: ab-initio methods, density functional theory (DFT), (semi-

)empirical methods and large scale force-field methods [5]. The ab-initio methods such as (post-) Hartree-Fock methods are successful at calculating molecule geometries, molecule energies, thermodynamic properties, etc. [5] Still, these methods are not fully able to capture all the properties of the so-called strongly correlated matter. They are defined in [6] as "a correlated electron problem is one in which interactions are so strong or have a character such that theories based on the underlying original "bare" particles fail even qualitatively to describe the material properties". There exist different methods to investigate these exciting materials. A very limited number of models are quantum integrable, meaning they can be solved in a non pertubative way. Also, some properties of models near criticality can be determined exactly with conformal field theory (CFT). But for most systems, we can only simulate the behaviour with numerical techniques. To make progress, new fast and accurate numerical methods are needed, because exact diagonalisation becomes unfeasible for large systems. Some examples of such numerical techniques, which will not be discussed here, are: Dynamical Mean Field (DMFT) / Dynamical Cluster Approximation (DCA), Series expansion, Density Matrix Embedding Theory (DMET), Fixed-node Monte Carlo, Diagrammatic Monte Carlo, Variational Monte Carlo, Functional renormalization group (FRG) and Coupled-cluster methods. [7]. In this thesis, a technique is proposed that builds on the broad field of Tensor Networks. Strongly correlated electron systems host a tremendous variety of fascinating macroscopic phenomena including high-temperature superconductivity, quantum spin-liquids, fractionalized topological phases, and strange metals [6].

1.2   Tensor networks

One problem that makes simulation of large quantum systems difficult is the exponentially large size of the Hilbert space. This is often referred to as the curse of dimensionality. But it turns out that Hilbert space of a many-body quantum system is far too large: all physical states, that is, all states that can ever be created, live on a tiny submanifold of measure zero [8]. The physical states obey the so called 'area law', which states that the entanglement of a system scales according to its area and not the volume. Tensor Networks (TNs) automatically have this property, making them an efficient description of a quantum system.

1.3   Outline thesis

This master's thesis is organised as follows. Chapter 2 gives a practical introduction to TNs. Some important TN algorithms are discussed, including the VUMPS algorithm to contract infinite lattices. Chapter 3 treats phases of matter and criticality and some selected quantum models. The importance of operator exponentials is explained. Chapter 4 introduces the central topic of this thesis: TN cluster expansions to simulate operator exponentials. Chapter 5 treats the implementation aspects of the cluster expansions, in particular the

numerical solvers. Chapter 6 measures the accuracy of the cluster expansion in 1D and 2D. Also, some phase transitions of the 2D Transverse Field Ising (TFI) model are calculated and compared with results from literature. Chapter 7 summarises the main findings and gives an outlook for future work.

Chapter 2

Tensor networks

> We could, of course, use any
> notation we want; do not laugh at
> notations; invent them, they are
> powerful. In fact, mathematics is,
> to a large extent, invention of
> better notations.
>
> Richard Feynman

2.1   Introduction

TNs form a vast subject. This chapter gives a very brief introduction into the
field of TNs. First, the graphical notation used to represent these tensors is
explained. The most relevant kinds of TNs for this dissertation are introduced.
A practical introduction is given how to manipulate these networks. Next, a se-
lected number of Matrix Product State (MPS) algorithms are given. The focus
will be to understand the intuition behind them, but not to provide a mathe-
matical rigorous treatment. The next section treats the contraction of infinite
dimensional 2D networks. This is chosen both to improve the understanding of
TNs through example, and because one of the algorithms, variational uniform
Matrix Product State algorithm (VUMPS), will be widely used in to calculate
2D phase transition later on.
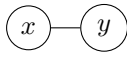
2.2   Tensor networks

2.2.1   Graphical notation

Before explaining TNs, some graphical notation should be introduced. This
really is a way to conveniently write vectors, matrices and in general tensors
without the need to introduce many labels. A tensor T is represented by a

circle with a number of external legs, according to the number of external indices. Connected legs are summed over, similar to matrix multiplication. Some examples are shown in table 2.1.

Table 2.1: Examples of graphical notation.

conventional	Einstein	tensor notation
$\vec{x}$	x_α	x —
M	$M_{\alpha\beta}$	— M —
$\vec{x} \cdot \vec{y}$	$x_\alpha y_\alpha$	x — y

2.2.2   Representing a quantum state

TNs come in many shapes and forms. They are used to represent a tensor with many legs. A general quantum state with N sites can be described in a given basis $|i\rangle$ in the following way:

$$|\Psi\rangle = \sum_{i_1 i_2 \cdots i_n} C^{i_1 i_2 \cdots i_n} |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle . \tag{2.1}$$

Here the tensor C holds all the information of the quantum state. The graphical representation can be seen in fig. 2.1. This requires an exponential number d^n

Figure 2.1: Exmple of a tensor C with multiple physical legs.

of coefficients C where d is the dimension of basis $|i\rangle$. In order to make the problem tractable, the following form is proposed as wave function:

$$C^{i_1 i_2 \cdots i_n} = C^{1\,i_1}_{\quad \alpha_1} C^{2\,i_2}_{\quad \alpha_1 \alpha_2} \cdots C^{n\,i_n}_{\quad \alpha_{n-1}} \tag{2.2}$$
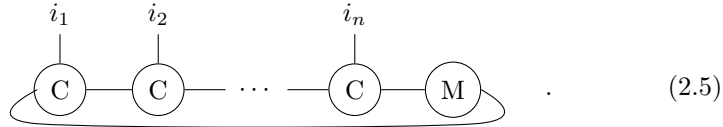
$$\tag{2.3}$$

where summation over shared indices is implied. It is always possible to find such a representation by means of matrix decomposition (see section 2.3). The summation bond α_i is called a virtual bond and their dimension is denoted by χ. At this point, this is not yet an improvement because the bond dimension needs to be exponentially large in order to represent the tensor C exactly. To make the chain translation invariant, all the tensors $C^i_{\alpha\beta}$ are the same tensor C. The chain is closed by a matrix M which contains the boundary conditions. Setting $\alpha_n = \alpha_0$. We can now write this as a trace over matrix products

$$|\Psi\rangle = \text{Tr}\big(C^{i_1} C^{i_2} \cdots C^{i_n} M\big) |i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_n\rangle, \qquad (2.4)$$

or in graphical notation

$$(2.5)$$

2.2.3 Classification

TNs come in many shapes and many forms. During this thesis, we will mainly encounter the types explained here.

2.2.3.1 MPS
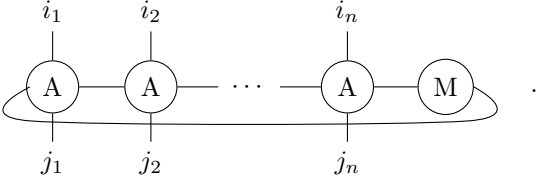
A general MPS is of the form eq. (2.5). For an infinite extended chain with translation invariance, it is logical to assume a unit cell:

$$(2.6)$$
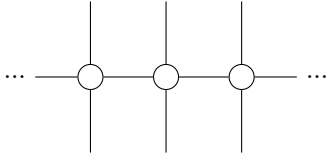
This will be the form we will work with in this thesis. Of course, larger unit cells are also possible. MPSs are dense. This means that (with a bond dimension exponential in the system size) the MPS can represent every state in the full Hilbert space [9]. However, the low energy states can typically be represented by a much lower bond dimension. This can be explained in terms of the so-called 'area law': the entropy low energy states often scale with the boundary area of volume. MPSs exactly follow this area law.

2.2.3.2   MPO
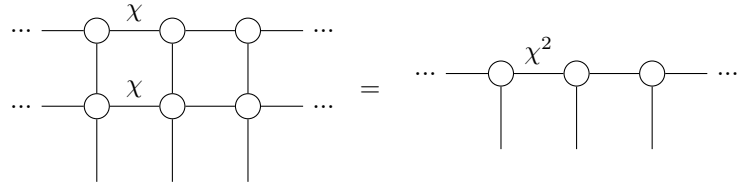
Similar to an MPS, a Matrix Product Operator (Matrix Product Operator (MPO)) is defined by

$$\hat{O} = \sum Tr(A^{i_1 j_1} A^{i_2 j_2} \cdots A^{i_n j_n} M)$$
$$\times |i_1\rangle \langle j_1| \otimes |i_2\rangle \langle j_2| \otimes \cdots \otimes |i_n\rangle \langle j_n|$$

(2.7)

The matrix M contains the boundary conditions of the operator. It's clear that this can again be made translationally invariant A uniform MPO is denoted by

(2.8)

This can also be reinterpreted as an MPS with physical bond dimension D^2, by grouping the i and j indices together per site. Acting with an MPS on a MPO $\hat{O}|\Psi\rangle$ again results in an MPS

(2.9)

The new MPS has the same physical dimension, but the bond dimension is increased from χ to χ^2.

2.2.3.2.1   Example of MPO Hamiltonian
As an example of an MPO representation of a Hamiltonian, consider the TFI Hamiltonian with exponentially decaying interactions:

$$\hat{H} = -J \sum_j \sum_{n>0} \lambda^{n-1} X_j X_{j+n} - h \sum_j Z_j.$$

(2.10)

The MPO is

$$\hat{O} = \begin{bmatrix} I & 0 & 0 \\ -JX & \lambda I & 0 \\ -hZ & X & I \end{bmatrix}$$

$$\hat{w}_l = \begin{bmatrix} -hZ & X & I \end{bmatrix}$$

$$\hat{w}_r = \begin{bmatrix} I & -JX & -hZ \end{bmatrix}^T$$

(2.11)

with X and Z the Pauli matrices, and $\hat{w}_l$ and $\hat{w}_l$ the boundary vectors [10]. Indeed, explicitly performing the multiplication for 3 sites gives

(2.12)

$$= \begin{bmatrix} -hZ_1 & X_1 & I \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ -JX_2 & \lambda I & 0 \\ -hZ_2 & X_2 & I \end{bmatrix} \begin{bmatrix} I \\ -JX_3 \\ -hZ_3 \end{bmatrix}$$

(2.13)
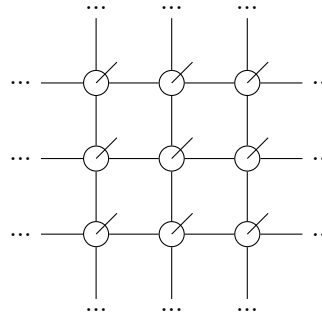
$$= \begin{bmatrix} (-hZ_1 - JX_1X_2 - hZ_2) & (-\lambda X_1 + X_2) & I \end{bmatrix} \begin{bmatrix} I \\ -JX_3 \\ -hZ_3 \end{bmatrix}$$

(2.14)

$$= -h(Z_1 + hZ_2 + hZ_3) - J(X_1X_2 + X_2X_3) - J\lambda X_1X_3$$

(2.15)

as expected.

2.2.3.3 PEPS

Projected entangled pair states (Projected Entangled Pairs State (PEPS)) are the 2D equivalent of MPS. In the visualisation, the physical indices come out of the plane.
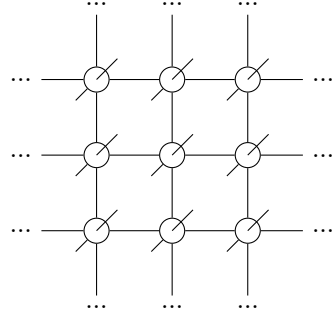
(2.16)

PEPSs also obey the area law, making them efficient parametrisations of the relevant corner of the full Hilbert space.

2.2.3.4   PEPO

The operator version of PEPS is called PEPO. The depiction looks as follows:

$$(2.17)$$

2.2.3.5   Others

To show there exist many more TNs beside the ones treated in this thesis, 2 examples are shown in fig. 2.2. Two Tree Tensor Networks (TTNs) are shown. The red indices are physical. multi-scale entanglement renormalization ansatz (MERA) (c) is also shown. These TNs are special because they can be contracted exactly, as opposed to for instance PEPS.
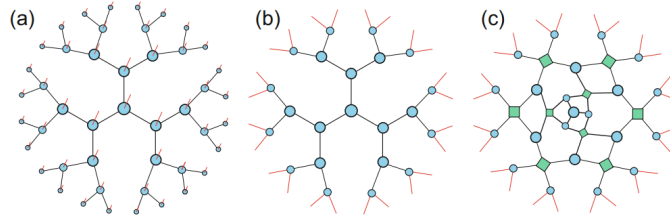
Figure 2.2: (a) and (b) two different TTNSs and (c) MERA. Figure taken from [11].

2.3   TN manipulations

This section serves as an introduction of TN manipulations. The overview mainly focusses on MPS/MPO networks, but most of the operations translate to the 2D case. The MPSs are processed by transforming the tensor into a matrix, performing some matrix calculations and casting it back into its original form. In this way, the standard methods from linear algebra can be used. This section gives some examples how this is done in practice.

2.3.1 Basics

2.3.1.1 Grouping legs

One of the most basic manipulations is to group some legs of a tensor into one leg:

$$
\begin{aligned}
T^{i_1 i_2 j_1 j_2} &= \boxed{\quad T \quad} \\
&\cong (i_1 j_1) \boxed{\quad T \quad} (i_2 j_2) \\
&= T^{(i_1 j_1)(i_2 j_2)}
\end{aligned}
\tag{2.18}
$$

The dimension of the new leg is the product of the dimension of the individual legs. Contracting legs separately or in merged form gives the same result. The 4 leg tensor and matrix contain exactly the same information. Manipulating this in memory requires both permute and reshape commands. This requires some time as the internal representation of the matrix changes.

2.3.1.2 Virtual levels

A matrix can be split in smaller block matrices. Similarly, a leg of a tensor can be split in virtual levels. Each virtual level has its own dimension. Contracting a common leg means summing over all the virtual levels, and performing a normal contraction for each virtual level. These virtual level will be useful in chapter 4.

2.3.1.3   Decomposition

The grouping above can be applied to decompose a tensor into 2 tensors with matrix techniques. An example is

$$
\begin{aligned}
&= O^{i_1 i_2 j_1 j_2}_{\alpha_u \gamma_w} \\[4pt]
&\cong O^{uw}_{(\alpha_u i_1 j_1)(\gamma_w i_2 j_2)} \\
&= O^{uv}_{(\alpha_u i_1 j_1)\alpha_v} O^{vw}_{\alpha_v (\alpha_w i_2 j_2)} \\
&\cong
\end{aligned}
\tag{2.19}
$$

The indices U, V and W represent virtual levels. Step 2 reshapes and groups the indices into one index on the left and one on the right. The dimension of this index is the product of the separate dimensions. Step 3 decomposes the matrix into a product of 2 matrices. The last step transforms the indices back to separate legs. The rank of the original matrix is kept the same if

$$
\dim v = \min(\dim u, \dim w) \cdot d^2.
\tag{2.20}
$$

Here, d is the physical dimension. Many different matrix decompositions exist. Some useful examples here are Singular Value Decomposition (SVD) decomposition, eigenvalue decomposition, QR, $\cdots$.

2.3.1.4   Truncation

The above procedure also gives a way to truncate the bond dimension: first take together 2 neighbouring sites, then perform an SVD Decomposition: $T = U\Sigma V^\dagger$. Split as

$$
T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^\dagger,
\tag{2.21}
$$

where Σ_1 contains the n largest singular values. Then $\hat{T} = U_1 \Sigma_1 V_1^\dagger$ is the best rank n approximation to the original tensor.

$$
min_{rank(A) \leq n} \|T - A\| = \|T - \hat{T}\|
\tag{2.22}
$$

2.3.1.5 Inversion

Suppose we want to find a MPO O for given tensors A and B such that the following holds

$$= u \qquad (2.23)$$

The indices can be taken together in the following way: $\alpha = (ui_1j_1i_2j_2)$ and $\beta = (i_3j_3v)$. The problem is now rewritten as

$$A_{\alpha\gamma}O_{\gamma\beta} = B_{\alpha\beta}. \qquad (2.24)$$

This is a standard matrix equation and can hence be solved with linear algebra packages. Note that it is not necessary to calculate the full inverse of A^{-1} to obtain the solution, because linear solver are generally much faster. As this is one of the core problems to solve both in 1D and 2D, this will be discussed in detail in section 5.2.

2.3.1.6 Contraction order

TN diagrams determine unambiguously the result of a contraction. This does not mean all contraction orders are equivalent. The number of operations needed to contract 2 vectors is D. Each additional leg multiplies the number of operations by the bond dimension. Two contraction schemes with different time complexity are shown in fig. 2.3.
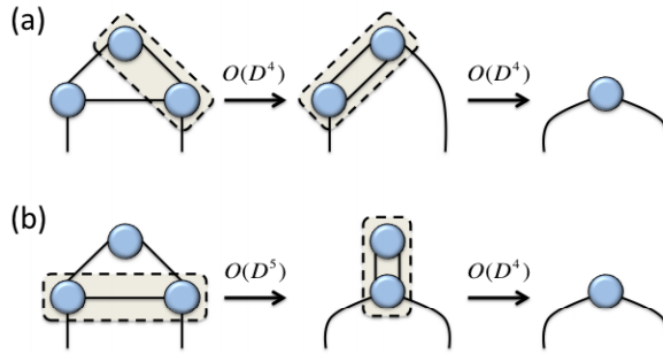
Figure 2.3: (a) Contraction of 3 tensors in $O(D^4)$ time (b) Contraction of same tensors in $O(D^5)$ time. Figure taken from [9].

2.4   MPS algorithms

This section and the following one, will introduce some different TN algorithms. The goal is to provide an intuitive explanation on how these algorithms work. For rigorous derivations and mathematical details, see [12] .

2.4.1   Canonical form

A translation invariant MPS is of the form eq. (2.6). It can easily be seen that inserting XX^{-1} on each bond doesn't change the contracted tensor. This is called a gauge transformation. Defining

$$A_l = \quad\rhd\quad = X\!-\!\!\bigcirc\!-\!X^{-1}$$

$$A_r = \quad\lhd\quad = Y\!-\!\!\bigcirc\!-\!Y^{-1} \tag{2.25}$$

and

$$\diamondsuit = XY^{-1} = C, \tag{2.26}$$

eq. (2.6) becomes

$$\cdots\ \rhd\ \rhd\ \diamondsuit\ \lhd\ \cdots\ . \tag{2.27}$$

Introducing one more tensor

$$A_c = \ \square\ = \ \rhd\ \diamondsuit\ = \ \diamondsuit\ \lhd\ . \tag{2.28}$$

At the moment the matrices X and Y are not yet defined. To bring an MPS A in its canonical form, the following choice is made

$$\rhd\!\!\!\rhd\ = \ \ \ = I \tag{2.29a}$$

$$\lhd\!\!\!\lhd\ = \ \ \ = I. \tag{2.29b}$$

The tensors below are the conjugated version of the tensors above. There is still some freedom: a unitary gauge transformation can still be applied. With this choice, it is possible to bring C in diagonal form. [12]

2.4.2 Entropy
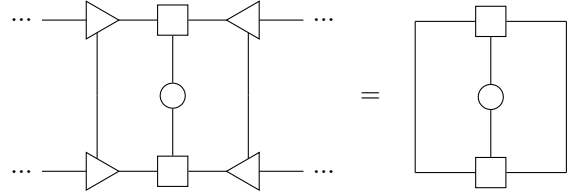
The canonical form above is very convenient to calculate the entropy, which is given by

$$S = -\sum_i \alpha_i^2 \log(\alpha_i^2) \tag{2.30}$$

where α_i are the singular values of C.

2.4.3 Expectation value

One property of MPS is that expectation values can be calculated exactly. Suppose we have an operator $\hat{O}$ acting on a single site of the MPS: $\left\langle \Psi \left| \hat{O} \right| \Psi \right\rangle$

$$\tag{2.31}$$

where we made use of eq. (2.29). Similar techniques allow for the calculation of correlation functions, energies, ...

2.5 TN contraction

Suppose we have an MPO $\hat{O}$ that represents a probability of finding encountering a given state: $\hat{O} |\Psi_i\rangle = p_i |\Psi_i\rangle$. The average value of a local measurement X is given by $\sum_i \left\langle \Psi_i | \hat{X}\hat{O} |\Psi_i \middle| \Psi_i | \hat{X}\hat{O} |\Psi_i \right\rangle = \mathrm{Tr}\left(\hat{X}\hat{O}\right)$. This will be needed in section 3.4.1. This section explains how to do this with tensor networks for infinite system sizes.

2.5.1   1D problem

The normalised probability is given by

$$\langle X \rangle = \frac{}{} \qquad (2.32)$$

In the thermodynamic limit there are an infinite number of A to the left and the right. This can be solved by taking the left and right fixed points Gl and Gr of the traced MPO A corresponding to the largest eigenvector λ.

$$\qquad (2.33)$$

Equation eq. (2.32) can now be easily calculated

$$\langle X \rangle = \frac{}{\lambda^n \quad Gr - Gr}. \qquad (2.34)$$

2.5.2   2D problem

PEPS contraction concerns a similar problem, but in 2D. Here, instead of already applying operator X, the (one site) reduced density matrix $\rho_{i,j}$ is com-

puted:

$$\rho_{i,j} =$$ (2.35)

This problem is much harder than in 1D. For instance, the related problem of calculating the expectation value of an operator, similar to section 2.4.3 in 1D, is #P-Hard [9].

2.5.3 Overview methods

As 2D contraction is a difficult task, and often the bottleneck in simulations, many algorithms exist to perform this. The classification presented here is taken from [13].

2.5.3.1 Real-space renormalisation-group methods

The general idea behind these methods is to course grain the TN. The first algorithm in this group of methods was Tensor renormalisation group (TRG) [14], shown in fig. 2.4. In the first step i, the tensor are split with an SVD in 2 different ways, depending on the position on the lattice. Step ii recombines 4 halves of the decomposition into 1 new tensor. The result is a rotated grid, with side length $\sqrt{2}$. The bond dimension is truncated during the SVD step.

Many other variants exist, such as Tensor Network Renormalisation (TNR), which will not be discussed here.

2.5.3.2 Corner transfer matrix methods

Another method goes by the name Corner transfer matrix renormalisation group (CTMRG), as described in [15]. The full network is approximated by 4 corner matrices C and 4 half row transfer matrices T as shown in fig. 2.5a. The idea is to find matrices such that inserting a row and truncating it again results in the same tensors. This is shown in fig. 2.5b. First, a new row is inserted. From fig. 2.5a it can be seen that this should not change the environment. Some tensors are taken together in step 2. As a final step, the tensors are suitably
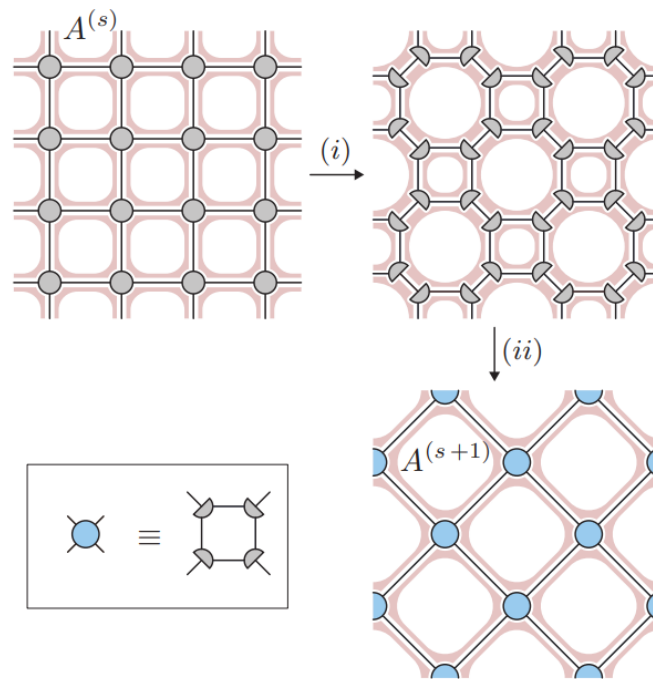
Figure 2.4: Steps in TRG procedure. Figure taken from [14].
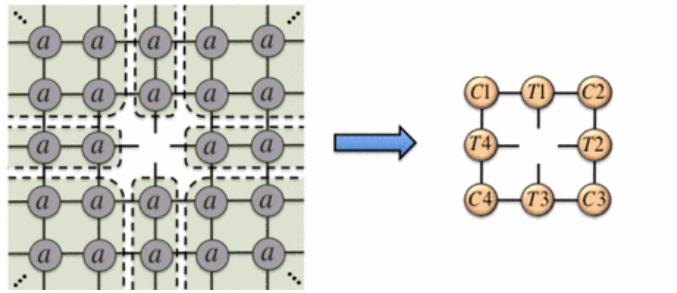
(a) Definition of environment
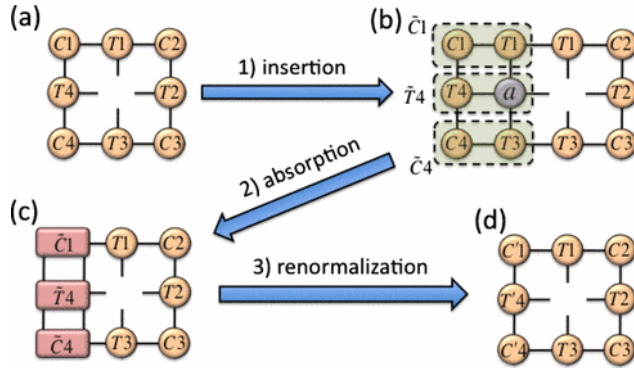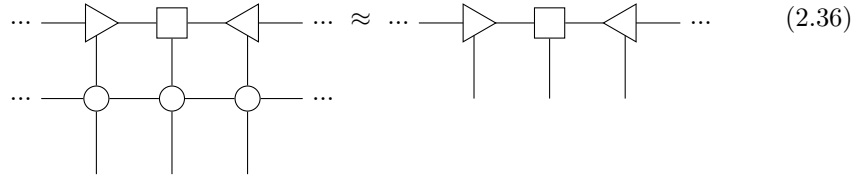
(b) 3 steps of CTMRG

Figure 2.5: Figures adapted from [15]

truncated. Once this cycle converges, the environment is known. Note that many important details are not written down here, and this is merely to give some intuition about TN contraction algorithms.

2.5.3.3   Boundary methods

The goal of these methods is to find an MPS fixed point for the infinite lattice:

$$\tag{2.36}$$
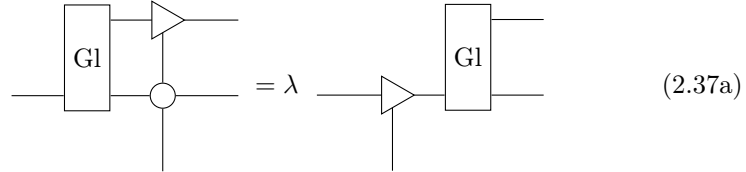
2.5.3.3.1   Time-evolving block decimation

Time-evolving block decimation was introduced as a method to calculate thermal states (this will also be encountered later on in section 3.4.3). It can also be used as a power method for contraction. The boundary MPS is applied to the MPO, and afterwards truncated the enlarged MPO is truncated. This is done by performing a truncation between all the even and odd sites, and afterwards between all odd and even sites. When this procedure is applied enough times, a fixed point is reached.

2.5.4   VUMPS
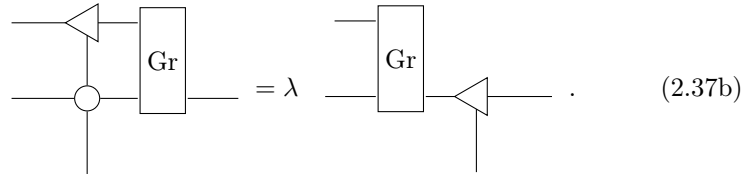
The purpose of this section is to give some intuition on the variational uniform Matrix Product State (VUMPS) algorithm, which will be used later on in this thesis. The goal is to find an MPS layer for the MPO (see eq. (2.33)). The expression holds approximately, because the MPS on the left-hand side has a larger bond dimension than on the right-hand side.

2.5.4.1   The equations

The method looks for tensors Gl and Gr which fulfil the conditions

$$= \lambda \qquad\qquad\qquad\qquad \tag{2.37a}$$

$$= \lambda \qquad\qquad\qquad\qquad . \tag{2.37b}$$

These eigentensor equations are solved in practice in a slightly different manner

$$ \text{(diagram)} = \lambda \ \text{(diagram)} \ , \tag{2.38} $$

where the Ar tensor was connected from below on both sides and eq. (2.29) was used on the right-hand side. The blocks in eq. (2.37) form a zipper from the left and right respectively. Each application brings down one of the MPS tensors in the following way

$$ \text{(diagram)} \tag{2.39} $$

The left and right zipper can now move towards each other, until they meet at the center. One more condition is needed to fullfil eq. (2.36):

$$ \text{(diagram)} = \lambda_{AC} \ \text{(diagram)} \ . \tag{2.40} $$

This completely determines the problem. One more equation is used in practice to solve the problem. Combining one of the equations in 2.37, the definition of A_c eq. (2.40) and eq. (2.29) gives C

$$ \text{(diagram)} = \lambda_C \ \text{(diagram)} \ . \tag{2.41} $$

Then eq. (2.36) is solved

$$
\tag{2.42}
$$

Contracting a 2D TN is thus reduced to solving the eq. (2.28), eq. (2.37), eq. (2.40) and eq. (2.41) simultaneously. Inspection of the equations shows that the following cycle needs to be solved:
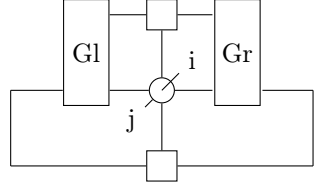
- $A_c, C \rightarrow A_l, A_r$ eq. (2.28)

- $A_l, A_r \rightarrow Gl, Gr$ eq. (2.37)

- $Gl, Gr \rightarrow A_c, C$ eq. (2.40) and eq. (2.41)

The calculated environment (i.e the tensor GLm Gr and MPS A) can now be used to solve the original problem. The same MPS [1]. Equation (2.35) now becomes

$$
\tag{2.43}
$$

---

[1]this will not be always the case, see section 2.5.4.4

As always, the identity (2.29) can be used to simplify the equation to

$$\begin{array}{c} \text{Gl} \quad \boxed{} \quad i \quad \text{Gr} \\ \otimes \\ j \end{array} \qquad (2.44)$$

2.5.4.2   The derivation
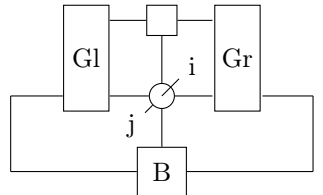
While the above derivation is reasonable, it is not very rigorous. The algorithm finds its origins in tangent space methods, as explained in [12]. Not every state in the many body Hilbert space can be represented by an MPS. By carefully constructing the tangent space and making use of the available gauge freedom, a compact expression can be found for the tangent space projector $\mathcal{P}_A$. This projects a state from the many body Hilbert space onto the tangent space of an MPS A. For an optimal MPS approximation A, the error made in the approximation should be orthogonal to the tangent space. For eq. (2.36) this means that the application of the projection of the error on the tangent space should be zero, i.e. the MPS is at a variational minimum [13]. Applying the projector $\mathcal{P}_A$ to eq. (2.36) exactly gives rise to the equations stated earlier.

2.5.4.3   Multisite

In [13], a version of VUMPS is proposed that can compute the environment of an m by n unit cell with only a linear increase in computation cost. Larger unit cells might be needed for ground states with a non-trivial unit cell.

2.5.4.4   Calculating the MPS from below

One question, which does not seem to be answered in literature, is which tensor B to use to complete the contraction from below for a complete general MPO

$$\begin{array}{c} \text{Gl} \quad \boxed{} \quad i \quad \text{Gr} \\ \otimes \\ j \\ \boxed{B} \end{array} \qquad (2.45)$$

We will get back to this question in section 6.4.5.2. A method suggested to me, and used throughout this thesis, is finding the largest eigenvalue from below

(similar to eq. (2.40)):

$$
\begin{array}{c}
\boxed{\text{Gl}} \;\; \ominus \;\; \boxed{\text{Gr}} \\
\boxed{\text{B}}
\end{array}
= \lambda_B \quad \boxed{\text{B}} \tag{2.46}
$$

This has some problems: B can not be brought into a canonical form according to eq. (2.28) (in combination with the analogue of eq. (2.41)), and hence does not represent a fixed point MPS as was the case from above. This also doesn't work in the multisite setting. This finding is not completely surprising, suppose we have a fixed point MPS B (denoted by bold tensors) from below (for instance obtained by performing the VUMPS algorithm on an MPO rotated by 180 degrees), then contracting the upper half plane and the lower half plane amounts to:

$$
\cdots \;\; \triangleright \;\; \boxed{\text{Gl}} \;\; \underset{j}{\overset{i}{\oslash}} \;\; \boxed{\text{Gr}} \;\; \triangleleft \;\; \cdots \tag{2.47}
$$

To solve this equation, the left and the right fixed points need to be calculated

$$
\boxed{\text{Fl}} \;\; \triangleright \;\; \triangleright \; = \lambda_f \;\; \boxed{\text{Fl}} \tag{2.48}
$$

Then (with Fr the analogous fixed point from the right side) the problem becomes:

$$
\boxed{\text{Fl}} \;\; \boxed{\text{Gl}} \;\; \underset{j}{\overset{i}{\oslash}} \;\; \boxed{\text{Gr}} \;\; \boxed{\text{Fr}} \tag{2.49}
$$

Both methods could be equal, if the following holds (up to a constant factor)

$$
\boxed{\text{B}} \;\; = \;\; \boxed{\text{Fl}} \;\; \blacksquare \;\; \boxed{\text{Fr}} \;\; . \tag{2.50}
$$

Some numerical investigation is needed to test whether or not this method gives a correct result.

2.6 Conclusion

TNs were introduced from a practical point of view: what different kinds exist, how can you manipulate a TN, etc. Some useful algorithms are discussed. One of the reasons MPSs are the standard to simulate strongly correlated systems is its successful canonical form as discussed. Density-matrix renormalization group (DMRG), first introuduced by White in 1992, is one of the most successful methods to study 1D lattices. This algorithm can be reformulated to fit in the MPS-formalism. [16] The difficulty of contracting an 2D TN was touched upon, and some powerful algorithms are introduced, in particular VUMPS. The engineering approach was taken, mainly focussing on the diagrams and not on the mathematical rigour behind them. For VUMPS, the question is raised what the best way is to contract the lower half plane. An untested solution is proposed.

Chapter 3

Strongly correlated matter

> All models are wrong, but some
> are useful.
>
> ———————————————
>
> George Box

3.1 Introduction

This chapter talks about strongly correlated matter. First, the focus will be on
the different phases of matter, and phase transitions between them. There are
a lot of interesting aspects related to phase transitions, not in the least univer-
sality. This principle says that the physics of many models at criticality can be
captured by a limited numbers of classes, each charactered by a set of critical
exponents. From a simulation point of view, the finite-size scaling method is
introduced to capture these properties while using a relative small grid. In the
second section, some models are introduced, together with some known proper-
ties of these models. These models will be used to test the cluster expansions
introduced later. In a last section, an overview of operator exponentials is given.
These are used both in statistical mechanics as in time evolution of a quantum
state. The current methods for approximating these exponentials within the
TN framework are discussed. The cluster expansions from this thesis, will also
be a TN method to simulate the operator exponentials.

3.2 Phases and Criticality

3.2.1 Phases of matter

An important area of research is the study of the different phases of (quantum)
matter. A phase is a state of matter in which the macroscopic physical proper-
ties of the substance are uniform on a macroscopic length scale. These phases

can be measured by the thermodynamic function, i.e. by a function of a few macroscopic parameters. [17]. More precisely, for a given phase the properties vary as an analytic function of the macroscopic variables. Interesting physics happens at the boundary between 2 or more distinct phases. The phase transitions were classified by Ehrenfest [18], who looked at the free energy across the phase boundary. If the free energy shows a discontinuity, it is called first order (or discontinuous) phase transition. Similarly, if the derivative shows a discontinuity, it is called second order (or continuous). Higher order phase transitions are possible, and there are even examples of infinite order transitions, such as the Berezinskii-kosterlitz-thouless (BKT) transition. In 2016, kosterlitz and thouless were awarded the nobel prize in Physics for their discovery [19].

3.2.2   Symmetry breaking

Sometimes, but not always, a phase transition is related to spontaneous symmetry breaking. A state $|\Psi\rangle$ is said to be symmetric under a unitary transformation U if the state only changes by a phase factor: $\hat{U}|\Psi\rangle = e^{i\phi}|\Psi\rangle$. A Hamiltonian possesses a symmetry if it commutes with U: $[H, U] = 0$ [20]. A remarkable fact is that many ground states are not invariant under a symmetry U of the Hamiltonian. For phase transitions associated with a broken symmetry, one can define an order parameter. This parameter evaluates to 0 for the symmetric phase, but not for the spontaneous broken phase. In continuous or second-order phase transitions the order parameter increases continuously from zero as the critical temperature is traversed. The entropy also changes continuously. On the other hand, the correlation length and related energy scales diverge at the critical temperature. In fact, at the critical temperature of a second-order phase transition, the system becomes scale-invariant, in the sense that physical properties no longer depend on the length (or energy) scale at which they are probed. Many symmetry-breaking phase transitions are second-order, with the onsets of superfluidity, (anti)ferromagnetism and many phases of liquid crystals as famous examples[20].

3.2.3   Universality

Universality looks at the behaviour of the system near a continuous phase transition. These can be described well by so-called power laws. For classical phase transitions (driven by temperature) near the critical temperature T_c, observables a_i depend in the following way on the reduced temperature $t = \frac{T-T_c}{T_c}$: $a_i(t) \sim t^{\alpha_i}$ (see section 3.2.6). One would expect that the set of critical exponents α_i depends on the precise form of the Hamiltonian of the system, but it turns out these exponents can be captured by a limited number of universality classes. This means that the physics near criticality is completely understood once it is understood for one member of the class.

Table 3.1: Definition of parameters for Ising model

Symbol	name
m	magnetisation
ξ	correlation length
g	external field
t	reduced temperature
τ	relaxation time

Table 3.2: Definition of parameters for Ising model. Values from [21].

Symbol	relation	2D value
β	$m \sim t^\beta t$	1/8
ν	$\xi \sim t^{-\nu}$	1
z	$\tau = \xi^z$	1

3.2.4 Critical exponents for spin systems

Table 3.1 defines some parameters for the Ising system (see also section 3.3.1) and table 3.2 its relevant critical exponents. The 2-point correlation function is defined as $f(x,y) = \langle m(x)m(y)\rangle - \langle m(x)\rangle \langle m(y)\rangle$. At larger distances this decays exponentially fast $f(x,y) = e^{-\frac{|x-y|}{\xi}}$ (see section 3.2.6), where ξ is the correlation length. For the ordered phase, the following relations hold: $m \sim |t|^\beta$, $\xi(t) \approx |t|^{-\nu}$ [21]. At the critical temperature near a quantum phase transition for 2D ising, the following relation holds: $t \approx |g - g_c|^{z\nu_{3D}}$ [22].

3.2.5 Finite-size scaling

Phase transitions only occur for systems with an infinite number of degrees of freedom. [23] This poses a problem, as in for instance Monte Carlo simulations only finite grids can be simulated. One computational expensive way to extract the properties in the thermodynamic limit is by making the grid increasingly bigger until the properties have converged. Fisher's insight was that the properties could also be extrapolated from different finite-size calculations [24] by making the following assumption: near a critical point, every thermodynamic property scales as a universal function of L/ξ, with L the size of the system and ξ the correlation length. Define $t = \frac{T-T_c}{T_c}$. The mathematical formulation is [25]

$$A(T, L) = L^{\kappa/\nu} f_A(tL^{1/\nu}). \tag{3.1}$$

This holds for t small (near critical point) and L sufficient large compared to the lattice spacing. The exponents can be fitted by plotting $A(T, L)L^{-\kappa/\nu}$ as a function of $tL^{1/\nu}$ for different sizes L and temperature t. For the correct critical exponents and critical temperature, all the points should collapse to one single graph. From the ansatz eq. (3.1), other ways can be derived to determine certain coefficients.

3.2.5.1 Finite-size scaling for MPS

The finite-size scaling for MPS is somewhat different. In [26], it is argued that δ can take the place of $1/L$. Suppose λ_i are the eigenvalues of eq. (2.38) ordered from the largest real part to smallest. Then

$$\epsilon_i = -\log(|\lambda_i|) \tag{3.2}$$

and

$$\delta = -\sum_i c_i \epsilon_i \quad \sum_i c_i = 0 \tag{3.3}$$

The intuition behind this is as follows: for an MPS approximation, the gaps in the transfer spectrum always go to zero for sufficiently large bond dimension. The distance between these gaps is thus a measure for the system size.

3.2.5.2 Subleading corrections

In [25], it is argued that the form proposed in eq. (3.1) does also not fully take into account the finite-size effects. Subleading corrections could be introduced as follows:

$$A(T, L) = L^{\kappa/\nu}(1 + cL^{-\omega})f_A(tL^{1/\nu} - dL^{-\phi/\nu}) \tag{3.4}$$

Indeed, this reduces to eq. (3.1) for sufficiently large L. Because there is always data needed to perform the collapse in a region around the critical point, the original procedure could be biased and result in wrong parameters. On the other hand, introducing extra parameters can lead to overfitting, again not improving the result.

3.2.6 CFT

One of the tools to derive some properties of phase transitions is Conformal Field Theory (CFT). CFT is a quantum field theory, which obeys an additional rule: the physics remains invariant under a conformal transformation. The exact form of these transformations is

$$g'_{\mu\nu}(x') = \Lambda(x)g_{\mu\nu}(x). \tag{3.5}$$

In 2D, the shape of the correlation functions can be determined exactly, and indeed correspond to the form from the previous section. One of the properties characterising a conformal field theory is the central charge c, which can only take a discrete number of values. In the case of unitary systems with $c \leq 1$, this has turned out to give a complete classification of possible two-dimensional critical behaviour [27]. For the 2D Ising model, the central charge is $c = 1/2$. A scaling relation used in the results chapter is [28]

$$Le^{6S(T,L)/c} \sim \xi. \tag{3.6}$$

3.2.7 Quantum phase transitions

A traditional second order phase transition is driven by a change in temperature. Quantum phase transitions on the other hand happen at zero temperature under influence of another parameter g of the model. At finite temperature, 2 things can happen (see fig. 3.1): either there is a line connecting a classical second order phase transition to the quantum phase transition, or the phase transition disappears at finite temperatures [29].
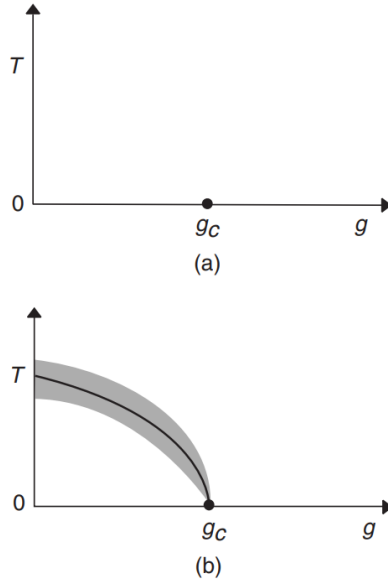
Figure 3.1: Two possible phase diagrams of a system near a quantum phase transition. In both cases there is a quantum critical point at $g = g_c$ and $T = 0$. In (b), there is a line of $T > 0$ second-order phase transitions terminating at the quantum critical point. The theory of phase transitions in classical systems driven by thermal fluctuations can be applied within the shaded region of (b). Figure and caption taken from [29].

3.3 Models

Models are a simplified mathematical description that captures some relevant physics of more complicated systems. This section introduces some specific models, their relevance and some properties. These models will be used later to benchmark the developed TN cluster expansions.

3.3.1 Ising model

The prototypical example of a model in the field of strongly correlated matter is the Ising model. It was first introduced 1925 by Ernest Ising, as a model to capture ferromagnetism. He proved that for a linear chain, there is no phase transition at finite temperatures. He wrongly concluded that this would also be the case in higher dimensions, but it turned out to be one of the deepest and far-reaching problems in the 20th century [30].

The Ising model, in essence, assigns an energy contribution to neighbouring spins. These spins sit on a fixed position on a chain (1D) or lattice (2D/3D/...). In classical Ising, the operators in the Hamiltonian all commute with each other. An energy is assigned between neighbouring spins. There is also a contribution for the alignment with an external magnetic field in the same direction. In TFI model, a transversal field is added. The operator measuring the transversal field no longer commutes with the operator measuring the alignment, and hence this is a quantum model. Often, the particles on the grid are spin 1/2 particles, but of course other particles are possible. Many generalizations exist for the Ising model, which will not be discussed her.

3.3.1.1 Classical Ising

The classical Ising model is given by the Hamiltonian

$$H = -J \left(\sum_{\langle ij \rangle} \sigma_i \sigma_j + h \sum_i \sigma_i \right), \tag{3.7}$$

where $\langle ij \rangle$ runs over all neighbouring lattice sites. The possible values of σ depends on the spin dimension. For spin 1/2 lattices $\sigma \in -1, +1$. g encodes the interaction strength of the external magnetic field. The sign J determines the low temperature ground state. A positive J will tend to align all neighbouring spins at low temperature. This is often called ferromagnetic, because all the aligned spins cause a macroscopic magnetisation. On the other hand, a negative J causes neighbouring spins to have an opposite sign. Depending on the sign of the longitudinal field h, the spins tend to align or antialign with this external field. This lifts the degeneracy of the ground state.

3.3.1.1.1 1D The classical 1D model was solved analytically by Lens. It can be solved analytically and shows no phase transitions at finite temperature.

3.3.1.1.2 2D In 2D, it becomes important to define the lattice. Here, and in the simulations, we will consider a square lattice. This model was famously solved by Lars Onsager in 1944, by using the transfer matrix method. In 2 dimensions, the Ising model has a continuous phase transition at finite temperature. The critical temperature is $T_c = \frac{2J}{T ln(\sqrt{2}+1)}$. Only the $h = 0$ case is solved analytically. For higher dimensions, no analytical solution is known. On

different lattices, interesting things can happen. For instance, the ground state of an antiferromagnet on a triangular lattice is not obvious to determine. The spins tend to antialign, but at least 2 of 3 spins on the corner of a triangle have to align. This is called frustration.

3.3.1.2 Transverse Field Ising

As we all know, the real world behaves, certainly at small length and time scales, quantum mechanically. Therefore, it is important to understand how the Transverse Field Ising (TFI) model differs from the classical model. In the TFI model, the operators no longer commute with each other. An example is the TFI model given by the Hamiltonian

$$\hat{H} = -J \left(\sum_{\langle ij \rangle} \sigma_i^x \sigma_j^x + g \sum_i \sigma_i^z \right). \tag{3.8}$$

In the case that $g = 0$, this is the classical Ising model (in the $h = 0$ case).

3.3.1.2.1 1D Different to the classical case, the 1D model contains a quantum phase transition at $g = 1$. For smaller fields, the ground state has a macroscopic magnetisation, for higher g not. The transition is of type Figure 3.1 (a). The critical value can be calculated exactly using the Kramers-Wannier duality [31]. The fact that this model has a phase transition can also be understood with section 3.3.4.

3.3.1.2.2 2D The 2D phase diagram of the TFI model is shown in fig. 3.2. The notation $\Gamma = g$ is taken in this figure. The classical phase transition is visible at $g = 0$. The red dot is the quantum critical point, situated at $G \approx 3.04$. A part of this phase diagram will be reproduced in section 5.4.3.

3.3.2 Heisenberg

The Heisenberg model is given by

$$\hat{H} = -\left(\sum_{\langle ij \rangle} J_x \sigma_i^z \sigma_j^z + J_y \sigma_i^y \sigma_j^y + J_z \sigma_i^z \sigma_j^z + h \sum_i \sigma_i^z \right). \tag{3.9}$$

This is a whole class of models, and they have different names depending on the values of J_α with $\alpha = x, y, z$. E.g. $J_x = J_y \neq J_z = \Delta$ is called the XXZ model, because the weight for σ^z is different. The isotropic version or simply Heisenberg model in the remainder of this thesis, which will be simulated in chapter 6. The model is

$$\hat{H} = -J \left(\sum_{\langle ij \rangle} \vec{\sigma} \cdot \vec{\sigma} \right). \tag{3.10}$$
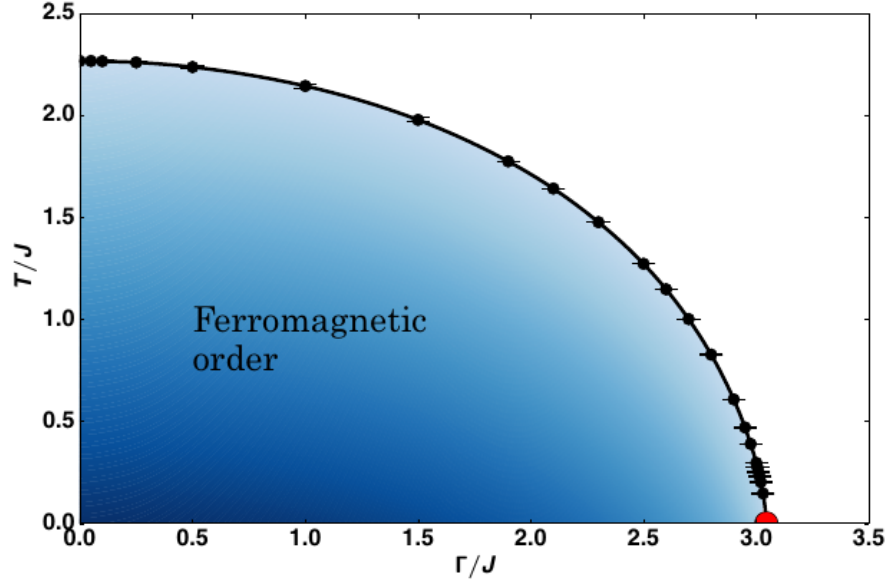
Figure 3.2: Phase diagram for 2D TFI model. Figure taken from [22].

3.3.3 Random

It's also possible to construct random Hamiltonians. This is a useful, because it ensures that no special simple structure is present. The single site (which can be thought of as an external field as before) and nearest neighbour Hamiltonians are generated by making Hermitian matrices with random real and complex numbers between -1 and 1. In order to make them comparable, the energy scale is set such that the norm of the Hamiltonian evaluated on 2 sites is 1.

3.3.4 Quantum to classical mapping

In a certain sense, a quantum model in d dimensions can be mapped to a classical model in d+1 dimension. Due to the Lie product formula

$$e^{A+B} = \lim_{M \to \infty} (e^{A/M} e^{B/M})^M \tag{3.11}$$

we can rewrite the partition function as

$$
\begin{aligned}
Z &= \sum_n e^{-\beta E_n} \\
&= \sum_n \left\langle n \left| e^{-\beta \hat{H}} \right| n \right\rangle \\
&= \sum_{n_1 \cdots n_M} \left\langle n_1 \left| e^{-\beta \hat{H}/M} \right| n_2 \right\rangle \left\langle n_2 \left| e^{-\beta \hat{H}/M} \right| n_3 \right\rangle \cdots \left\langle n_M \left| e^{-\beta \hat{H}/M} \right| n_1 \right\rangle \\
&= \sum_{n_1 \cdots n_M} \left\langle n_1 \left| e^{-\beta \hat{H}/M} \right| n_2 \right\rangle \left\langle n_2 \left| e^{-\beta \hat{H}/M} \right| n_3 \right\rangle \cdots \left\langle n_M \left| e^{-\beta \hat{H}/M} \right| n_1 \right\rangle
\end{aligned}
\tag{3.12}
$$

where completeness relations $I = \sum_{n_i} |n_i\rangle \langle n_i|$ inserted M times in line 3. This is similar to a path integral. The quantum imaginary time has become a spatial dimension. The quantum model of d dimensions now has the structure of a classical model in d+1 dimensions. In this way, the 2D TFI model can be mapped to the 3D classical Ising model [32].

3.4   Operator exponentials

While it is often possible to find exact MPO representation to represent a wide class of Hamiltonians (see section 2.2.3.2), it is much harder to do the same for exponentiated operators. These operators play an important role: they act as time evolution operators for quantum systems $|\Psi(t)\rangle = \exp\left(-i\hat{H}t\right)|\Psi(0)\rangle$. A very similar operator governs the partition function in statistical mechanics: the probability of finding a system at inverse temperature $\beta = \frac{1}{T}$ in a microstate i is given by $p_i = \exp\left(-\beta \hat{H}_i\right)$. This is often called "imaginary" time, due to the substitution $\beta = it$. The ability to calculate these operators is essential for understanding the dynamics of a given quantum model, and making contact with real world observations of these systems at finite temperature.

3.4.1   Statistical mechanics

The physics of a system in thermodynamic equilibrium can be derived from its partition function Z. The classical formula generalises to a density matrix ρ as follows:

$$
\begin{aligned}
Z &= \sum e^{-\beta E_n} \\
&= \sum_n \left\langle n \left| e^{-\beta \hat{H}} \right| n \right\rangle \\
&= \mathrm{Tr}\left(e^{-\beta \hat{H}}\right)
\end{aligned}
\tag{3.13}
$$

The first line is the partition function for classical discrete systems. The index n runs over all possible microstates. It is known that the probability to find the

system in a given microstate is given by

$$p_i = \frac{\sum e^{-\beta E_i}}{Z}.$$

(3.14)

A useful quantity is the density matrix ρ:

$$\begin{aligned} \rho &= \sum_j p_i \left|\Psi_j\right\rangle \left\langle\Psi_j\right| \\ &= \sum_j \frac{e^{-\beta\hat{H}}}{Z} \left|\Psi_j\right\rangle \left\langle\Psi_j\right|. \end{aligned}$$

(3.15)

With this notation, the partition function Z and ensemble average of an operator $\hat{X}$ are given by:

$$\begin{aligned} Z &= \text{Tr}(\rho) \\ \langle X \rangle &= \text{Tr}\left(\rho\hat{X}\right). \end{aligned}$$

(3.16)

3.4.2 Applications

3.4.2.1 Temporal correlation functions

The dynamical behaviour of a system can be captured by its dynamic correlation function:

$$\begin{aligned} C(r,t) &= \left\langle \hat{X}(0,0)\hat{X}(r,t) \right\rangle \\ &= \left\langle \hat{X}(0)e^{i\hat{H}t}\hat{X}(r)e^{-i\hat{H}t} \right\rangle \end{aligned}$$

(3.17)

This requires time evolution operators $e^{-i\hat{H}t}$.

3.4.2.2 Ground state

One practical way of finding the ground state $|E_0\rangle$ is to cool down a given random state $|\Psi(0)\rangle$ to very small T (large β) [9]:

$$|E_0\rangle = \lim_{\beta\to\infty} \frac{e^{-\beta\hat{H}}|\Psi(0)\rangle}{\langle\Psi(\beta)\,|\,\Psi(\beta)\rangle}, \quad |\Psi(\beta)\rangle = e^{-\beta\hat{H}}|\Psi(0)\rangle.$$

(3.18)

3.4.3 TN methods

In the following section I will give a very short review of the current TN methods to simulate real or imaginary time evolution. This overview is mainly based on the review paper [33].

3.4.3.1 Approximations to $\hat{U}(\delta)$

The goal is to approximately make a MPO for a small timestep δ which gives a new MPS at time $t + \delta$.

3.4.3.1.1 Time Evolving Block Decimation (TEBD) Time-evolving block decimation (TEBD) uses the Trotter-Suzuki decomposition. Suppose the chain is split in even and odd sites.

$$\hat{H} = \hat{H}_{\text{even}} + \hat{H}_{\text{even}} \tag{3.19}$$

$$\hat{U} = e^{-i\delta\hat{H}_{\text{even}}} e^{-i\delta\hat{H}_{\text{even}}} e^{-i\delta\left[\hat{H}_{\text{even}}, \hat{H}_{\text{odd}}\right]}$$
$$\approx e^{-i\delta\hat{H}_{\text{even}}} e^{-i\delta\hat{H}_{\text{even}}} \tag{3.20}$$

This is now easy to solve: first apply $e^{-i\delta\hat{H}_{\text{even}}}$ for every even bond and afterwards $e^{-i\delta\hat{H}_{\text{odd}}}$. The error is $O(\delta^2)$ and the number of steps to reach temperature β is β/δ. The error can be made arbitrarily small. This can be generalised to higher order schemes.

3.4.3.1.2 MPO $W^{I,II}$ These methods directly use the MPO representation of a certain Hamiltonian This is a more recent method (2015) to construct an MPO first detailed in [34]. The idea is to generalise

$$1 + \delta \sum_x H_x \to \prod_x (1 + \delta H_x) \tag{3.21}$$

The error is formally still $O(\delta^2)$, but includes many more terms. The advantages lay in the fact that the form above has an efficient representation as an MPO. MPO W^I and W^{II} are capable of dealing with long-ranged interaction terms which makes it suitable to simulate 2D systems [33].

3.4.3.2 Global Krylov method

Krylov methods are widely used in linear algebra to calculate eigenvectors. An example is the Lanczos algorithm. These methods are applied to MPSs, but do not fully make use of its structure. For this method, only a MPO representation is needed.

3.4.3.3 MPS-local methods

3.4.3.3.1 Local Krylov The Krylov methods from the previous paragraph can be adapted to work on a reduced basis.

3.4.3.3.2 TDVP Time-dependent variational principle (TDVP) can be seen as a further development of the local Krylov method. Its also been formulated in as a tangent space algorithm, similar to the VUMPS derivation (section 2.5.4.2). The Schrödinger equation becomes

$$i\frac{\partial \left|\Psi(A(t))\right\rangle}{\partial t} = \mathcal{P}_{A(t)} H \left|\Psi(A(t))\right\rangle. \tag{3.22}$$

Where the right-hand side is projected on the tangent space, because the left-hand side is also a tangent vector. (See [12]).

3.5   Conclusion

An introduction to phases of matter, and their surprising structure was discussed. 2 models, namely Heisenberg model and the Ising model for different dimensions are discussed. The phases for different dimensions are listed. The last section explained the importance of operator exponentials to understand the physics of these models. Competing TN methods to approximate them numerically were very briefly listed.

Chapter 4

Construction Cluster expansion

4.1 Introduction

This is the key chapter of the whole thesis. Here, the novel TN method to construct the operator $e^{-\beta \hat{H}}$ is explained in detail. The basis of the method was first introduced in [35]. There are many variations on the same idea. Some of the most notable examples in 1D will be discussed. At this point, no simulation results will be given. These can be found in chapter 6. The section mentions some objective info about the construction, such as the bond dimension. The 2D construction will generalise the best result from 1D. First, a construction analogous to 1D will be presented. As can be expected, some new ideas are needed to capture the rich physics of the models in 2D. The question of how to construct these cluster expansions and other implementation details are reported in chapter 5.

4.1.1 Notation

First, some extra clarification on the notation is needed in order to avoid confusion. In the following, the external legs and virtual level 0 will be omitted. Also, all the physical indices will not be shown. This should not be confused with the diagram earlier.

$$O^{00} = \quad \underset{j}{\overset{i}{\vline}} \begin{array}{c} 0 \quad 0 \end{array} = \bigcirc \tag{4.1}$$

Each virtual level has its own bond dimension. The bond dimension of level 0 is 0. Two neighbouring sites connected through virtual level 1 are similarly denoted by

$$
O^{01}O^{10} = \quad \underset{j_1 \quad\quad j_2}{\overset{i_1 \quad\quad i_2}{\underset{0}{\bigcirc}\overset{1}{}\underset{0}{\bigcirc}}} \quad = \bigcirc \overset{1}{} \bigcirc .
\tag{4.2}
$$

As a reminder, unlabeled incdices such as in

$$
\overset{i_1 \quad\quad i_2 \quad\quad i_3}{\underset{j_1 \quad\quad j_2 \quad\quad j_3}{\overset{0}{\bigcirc}\bigcirc\overset{0}{\bigcirc}}} \quad = \bigcirc\!-\!\bigcirc\!-\!\bigcirc
\tag{4.3}
$$

implies a summation over all possible virtual levels. A combination is valid if all the separate tensors were defined previously. The goal is capture the exponential of a Hamiltonian operator $\hat{H}$ with local interactions

$$
\hat{H} = \left(\sum_{<ij>} H_2^i H_2^j + \sum_i H_1^i \right).
\tag{4.4}
$$

This Hamiltonian consists of 1 and 2 site operators. More general Hamiltonians can be used. The notation for the contraction of the TN will also be used to denote the Hamiltonian evaluated on the given geometry

$$
\begin{aligned}
H\left(\bigcirc\!-\!\bigcirc\!-\!\bigcirc\right) = & H_1 \otimes 1 \quad \otimes 1 \\
+ & 1 \quad \otimes H_1 \otimes 1 \\
+ & 1 \quad \otimes 1 \quad \otimes H_1 \\
+ & H_2 \otimes H_2 \otimes 1 \\
+ & 1 \quad \otimes H_2 \otimes H_2 \\
& \qquad .
\end{aligned}
\tag{4.5}
$$

4.1.2   Idea

This chapter shows the main construction of this dissertation. A cluster expansion is used to approximate $\exp\left\{\hat{H}\right\}$ for every possible geometry. The goal is to make a MPO/PEPO which captures the tensor exponential in the thermodynamic limit. The main idea is to make an extensive expansion by adding blocks which solve the model exactly on a local patch. Crucially, the expansion is not in the inverse temperature β but in the size of the patches. The local patches

are separated by a virtual level 0 bond. To make this somewhat more precise, the first steps of the expansion are shown here. The smallest patch, i.e. 1 site, encodes the exponential of that Hamiltonian

$$\bigcirc = \exp\left(-\beta H(\bigcirc)\right). \tag{4.6}$$

If there were no 2 site interactions, this already captures the full diagonalisation. Of course, such a model wouldn't be useful. The next step is to introduce 2 site interactions, where the one site interactions are subtracted from the diagonalised Hamiltonian.

$$\bigcirc \!\!-\!\!\overset{1}{}\!\!-\!\!\bigcirc = \exp -\beta H(\bigcirc \!\!-\!\!-\!\!\bigcirc)$$
$$-\bigcirc \!\!-\!\!\overset{0}{}\!\!-\!\!\bigcirc \tag{4.7}$$

The 2 site patch includes all orders in β, and not just second order terms. This is somewhat similar to the MPO W methods explained in section 3.4.3. Contraction of larger network lead to many terms, such as

$$\bigcirc \!\overset{1}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{1}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{1}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc. \tag{4.8}$$

The beauty of this lays in the fact that disconnected regions (regions separated by level 0) combine in exactly the right way to capture the terms appearing in the series expansion of the exact tensor exponential. Only the terms of the exponential which acts on 3 or more neighbouring sites at once, are not accounted for. Notice that in eq. (4.7), 2 new blocks are introduced

$$\overset{i}{\underset{j}{\overset{0}{-}\!\bigcirc\!\overset{1}{-}}} \quad \text{and} \quad \overset{i}{\underset{j}{\overset{1}{-}\!\bigcirc\!\overset{0}{-}}} \quad .$$ As can be seen, the dimension of virtual

level 1 needs to be d^2 (section 2.3.1.3), with d the dimension of physical level. Although different possible constructions already differ in the next step, one more step is added to make te construction and notation clear.

$$\bigcirc \!\overset{1}{-}\! \bigcirc \!\overset{1}{-}\! \bigcirc = \exp -\beta H(\bigcirc \!\!-\!\!-\!\!\bigcirc \!\!-\!\!-\!\!\bigcirc)$$
$$-\bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc$$
$$-\bigcirc \!\overset{1}{-}\! \bigcirc \!\overset{0}{-}\! \bigcirc$$
$$-\bigcirc \!\overset{0}{-}\! \bigcirc \!\overset{1}{-}\! \bigcirc \tag{4.9}$$
$$= \exp -\beta H(\bigcirc \!\!-\!\!-\!\!\bigcirc \!\!-\!\!-\!\!\bigcirc)$$
$$-\bigcirc \!\!-\!\!\bigcirc \!\!-\!\!\bigcirc$$

This is a cluster expansion of order $p = 3$, because the (longest) chain has 3 connected sites solved exactly. The error is of $O\left(\beta^{p-1}\right)$. The right-hand side of eq. (4.9) can be ommited, as it is just evaluating the exponentiated Hamiltonian on the same geometry as the left hand side and substructing all possible contractions of the blocks which were added previously. This very compact notation will be able to capture the essence of the different constructions. Because it is important for the remainder of the chapter, it is stressed that for an equation similar to

$$\boxed{\circ\!\!-\!\!\overset{1}{}\!\!-\!\!\circ\!\!-\!\!\overset{1}{}\!\!-\!\!\circ}, \tag{4.10}$$

the right-hand side of eq. (4.9) is implied. In the following section, different types will be discussed. For every chain lenght, a new block is defined. This could be done in numerous ways. The different types list some ways to do this.

4.2   Construction MPO

4.2.1   Type A

This type was originally proposed in [35]. The first few blocks in the expansion are

$$\tag{4.11}$$

The following types of blocks appear in the cluster expansion:

$$\tag{4.12}$$

with $n \in \mathbb{N}_0$ and $m = n - 1$. The O^{nn} block is in defined for a chain with an odd number of sites. The contraction of O^{nm} and O^{mn} is defined by a chain with even order. The decomposition is defined up to a gauge transformation.

4.2.1.1   Dimension

In this scheme, virtual level n has dimension d^{2n}. This dimension can be truncated if some error is allowed for the longest chain.

4.2.1.2 Discussion

Type A can form long chain, e.g.

$$
\text{O} \overset{1}{-} \text{O} \overset{1}{-} \text{O} \overset{1}{-} \text{O} \overset{1}{-} \text{O} \overset{1}{-} \text{O} \overset{0}{-} \text{O} \overset{1}{-} \text{O} \overset{1}{-} \text{O} \overset{1}{-} \text{O}. \tag{4.13}
$$

The block O^{11} is only defined to solve the matrix exponential exactly for 3 neighbouring sites. Therefore, the question arises whether these long chains will improve the error or make it worse for cyclic systems.

4.2.2 Type B

Type B only contains blocks of the following form; O^{mn} and O^{n0}. The first few blocks are

$$
\tag{4.14}
$$

The following split is made: $O^{mn} \cong U^n$ and $O^{n0} \cong \Sigma V^\dagger$. In this way the left inverse exists and doesn't need any calculation: $O^{mn} = U^\dagger$.

$$
\tag{4.15}
$$

4.2.2.1 Dimension

From the construction the bond dimension grows from the left to the right. All steps add d^2 to the dimension. For the last step, there are only d^2 non-zero

singular values. Introducing

$$
\begin{array}{c}
\underset{\text{m}}{\rule{0pt}{0pt}}\ O\ \underset{\text{n}}{\rule{0pt}{0pt}} \quad = A^m_{(\alpha ij)\beta}
\end{array}
$$

$$
\begin{array}{c}
\underset{\text{n}}{\rule{0pt}{0pt}}\ O\ \underset{0}{\rule{0pt}{0pt}} \quad = B^n_{(\alpha ij)\beta}\ .
\end{array}
\tag{4.16}
$$

Then the MPO doesn't change if there are matrices A'^n, A'^{n+1} and B'^n such that

$$
\begin{aligned}
S &= A^n A^{n+1} = A'^n A'^{n+1}\\
T &= A^n B^{n+1} = A'^n B'^{n+1}.
\end{aligned}
\tag{4.17}
$$

This can be used to decrease the bond dimension of the MPO. Matrices with optimal bond dimension can be calculated with generalised SVD. Generalised SVD decomposes 2 matrices as

$$
\begin{aligned}
S^\dagger &= (U\Sigma_1)Q^\dagger\\
T^\dagger &= (V\Sigma_2)Q^\dagger.
\end{aligned}
\tag{4.18}
$$

Identification gives $A'^m = Q$ The new bond dimension is the $\dim n' = d^2 \cdot \min(\dim n - 1, \dim(n+1))$. This is still higher than the dimension of type A.

4.2.2.2   Discussion

The bond dimension is larger than type A, but the long chains from type A are absent. The left inverse is always well-defined and doesn't need any computation, because Hermitian matrix U can be inverted easily. One major drawback is that for long chains, the virtual bonds are very large before they can be shrunk with the Generalised SVD procedure.

4.2.3   Type C

This type implements the same strict type as Type B, but in a different way. No calculation is involved, except the calculation of the exponentiated Hamiltonian to certain order. The following kind of MPO strings are allowed:

$$
\tag{4.19}
$$

and so forth. The primed indices have no special meaning, and could also be numbered starting from the largest index of the previous block +1. All but one MPO elements are chosen to be the identity matrix. The middle one is the exponentiated Hamiltonian with reshaped legs.

4.2.3.1 Discussion

As can be expected from the construction, the bond dimension grows very fast. This type is just as precise as Type B.

4.2.4 Type D

This type uses a different setup which tries to capture the best of both Type A and B. Type A can handle long range correlation better because of the introduction of O^{nn}, but the inverse was not necessarily well-defined. Type B had well conditioned inverses, but performs worse. The blocks appearing in type D are as follows:

$$
\overset{m}{-}\!\!\bigcirc_{\!O}\!\!\overset{n}{-}\!\!\bigcirc_{\!D_n}\!\!\overset{n}{-}\!\!\bigcirc_{\!O}\!\!\overset{m}{-} \quad and \quad \overset{n}{-}\!\!\bigcirc_{\!O}\!\!\overset{n}{-} \quad ,
\tag{4.20}
$$

which is similar to type A. The idea is to keep O^{mn} unitary up to a constant factor.

$$
\overset{m}{-}\!\!\bigcirc_{\!O}\!\!\overset{n}{-}\!\!\bigcirc_{\!D_n}\!\!\overset{n}{-}\!\!\bigcirc_{\!O}\!\!\overset{m}{-} \quad = U\Sigma V^\dagger
\tag{4.21}
$$

Matrix D_n is the singular value diagonal matrix divided by a normalisation factor ϕ. Both U and V are multiplied by $\sqrt{\phi}$.

4.2.4.1   Discussion

It's not completely clear what the values of ϕ should be. If ϕ is too large, long chains are not suppressed. If phi is too small, the O^{nn} blocks will become large and hence the chain will diverge again. A reasonable value is the sum of the singular values. Other combinations could be tried.

4.2.4.2   Matrisation

The cost of this type lies in the fact that it has no compact way of casting it to a matrix. The following works, but has quite a large dimension:

$$
\left[
\begin{array}{ccc|cc|cc|cc}
O_{00} & O_{01} & & -2O_{01} & & O_{01} & & O_{01}D_1^{1/2} & \\
O_{10} & & O_{12} & & -2O_{12} & & O_{12} & & \\
 & O_{21} & & & & & & & \\
\hline
O_{10} & & & & & & & & \\
 & O_{21} & & & & & & & \\
\hline
O_{10} & & & & & O_{11} & & & \\
 & O_{21} & & & & & O_{22} & & \\
\hline
D_1^{1/2}O_{10} & & & & & & & & D_1^{-1/2}O_{12}D_2^{1/2} \\
 & & & & & & D_2^{1/2}O_{21}D_1^{-1/2} & &
\end{array}
\right]
$$

4.2.5   Type E

Again, this is a strict variant which needs exactly twice the bond dimension of type A. The idea is to split every chain in a left and a right part. For the left part, the numbers increase while the right part they decrease. This construction carries over well to higher dimensions. The first few blocks are:

$$(4.22)$$

The construction is very similar to type A.

4.2.6 Type F

The idea behind this type is very similar to type D. The blocks look as follows:

$$\tag{4.23}$$

The blocks $O^{n-1,n}$ and $O^{n,n-1}$ are unitary matrices from the SVD decomposition, scaled by the largest singular value. The blocks $O^{n-1,n'}$ and $O^{n',n-1}$ are then used to actually solve the problem for the even length chains. In the next step, $O^{n,n}$ is added as usual. The idea here is once again to keep the inverses well-defined.

4.2.7 Conclusion

Some of the 1D constructions tried are discussed here. They can be roughly divided into 2 groups. B, C and E are strict variants, meaning only the explicitly constructed blocks will appear in the final expansion. As a consequence, they have exactly the same predictive power. While B has some advantageous properties such as its final bond dimension and well-defined inverses, type E will be used in the results chapter due to its simplicity and scalability. It also generalises well to 2D, in contrast to type B. The second category are the unstrict types. Type A has the lowest possible bond dimension to exactly represent a chain of a given length. One hurdle to overcome is the badly conditioned inverses, when implemented naively. Types D and F try to remedy this. D scales very badly with the maximum number of sites, and has a construction which doesn't fit in with the simple diagrams. The construction was only implemented in 1D code due to this. This type won't be reported, because it has a similar performance to type F. In short, type A, E and F will be reported in section 6.2.

4.3 Construction PEPO

While there were some interesting choices in the 1D construction, the number of possibilities in 2D is virtually limitless. The focus will mainly be to generalise type A to 2D. As can be expected, the construction starts off quite similar. The following blocks are called 'Linear', due to the way they will be solved.

4.3.1 Linear blocks

$$O^{0000} = \bigcirc = \quad \overset{0}{\underset{j}{\underset{0}{\bigotimes}}}{}^{i}_{0} \tag{4.24}$$

This block now contains 2 physical indices and 4 virtual legs. Similar to 1D construction, the 0 legs and physical indices are hidden. The next order 2 blocks are

$$\bigcirc \overset{1}{\quad} \bigcirc \quad \overset{\bigcirc}{\underset{1}{\big|}} \tag{4.25}$$

From 3 blocks onwards, the number of extra blocks start to increase:

$$\tag{4.26}$$

Of course, besides linear blocks also $\perp$ and $+$ shapes need to be constructed to fully capture the model.

$$\tag{4.27}$$

$$\tag{4.28}$$

Here, and in the following, care has to be taken in which order the blocks are added. In general, every block which fits in the given block needs to be constructed earlier. The construction continues by introducing a virtual level 2 as before:

$$\bigcirc \overset{1}{\quad} \bigcirc \overset{2}{\quad} \bigcirc \overset{1}{\quad} \bigcirc \tag{4.29}$$
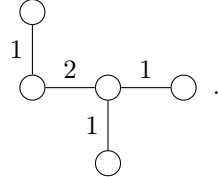
Again, 2 blocks are introduced. For other variations of the linear chain, only one block needs to be solved

$$\text{(4.30)}$$
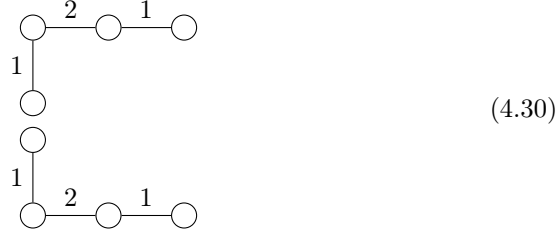
Due to the way they are constructed, the error for every linear chain of a given length will be the same as in the 1D case. Once again, all possible $\perp$ and $+$ blocks are created. Some of these blocks are shown here:

$$\text{(4.31)}$$

$$\text{(4.32)}$$
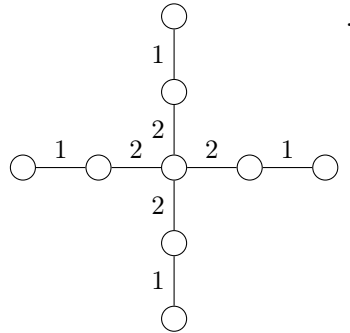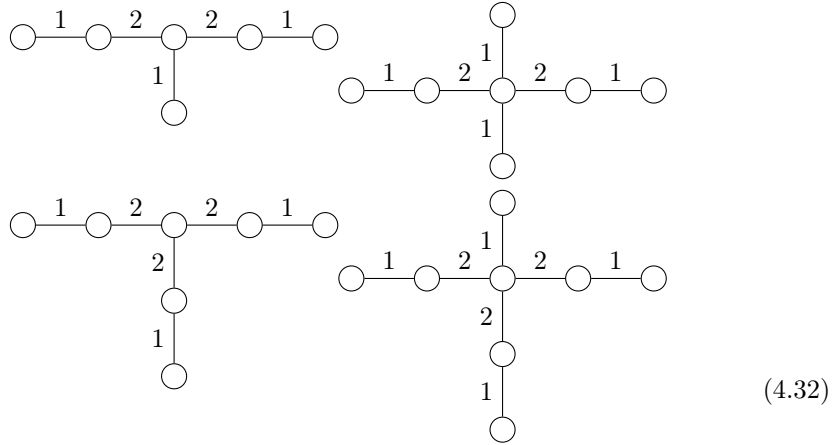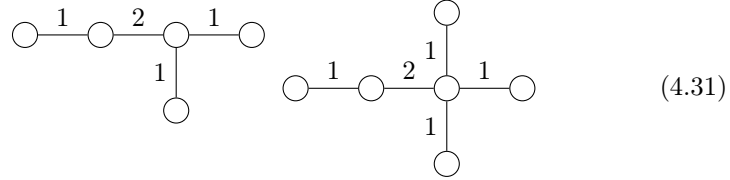
For each block shown, there are still multiple permutations of the legs possible. Due to the way the virtual level 1 blocks were added, all variations are captured, such as

$$\tag{4.33}$$

It is clear that a completely automated solver is needed to construct all these different blocks. From here on, the construction generalises easily to higher block numbers, and to higher dimensions. The difference between eq. (4.31) and the blocks in eq. (4.31) is the longest chain that fit in the geometry. For eq. (4.31), only chains of order 4 are present, while eq. (4.32) has chains of length 5. It seems that when a virtual level is present, it is most advantageous to create both the even and odd order chains, but this will not be the case when a virtual level is truncated.

4.3.2 Loops

While the blocks above certainly encode many finite-size patches, there are still quite some patches that need to be encoded. The simplest case is a 1 square loop.

$$\tag{4.34}$$

It is clear that this problem cannot be solved with the techniques from the previous section. The square loop needs a new virtual level α. In general, all the loop levels will be named with Greek letters for convenience. The simplest choice for the loop is:

$$\tag{4.35}$$

At this point all blocks of order 4 are solved.

4.3.2.1 Single extensions

The loops need to be connected to the linear blocks. One way to do this is as follows:

$$\tag{4.36}$$
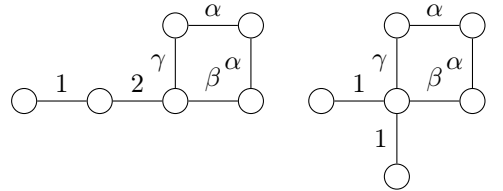
With the given blocks, the following combination is also possible

$$\tag{4.37}$$

This results in very large errors, and it would require many more blocks to be added in order to counteract this. Luckily, there are many possibilities in 2D. One example is
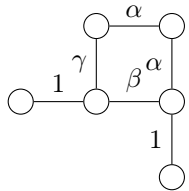
$$\tag{4.38}$$

Different to the α-cornerpiece, only a loop with on one corner an extension is possible. Of course, there is no need to stop here, the following blocks can now be constructed easily

$$\tag{4.39}$$

4.3.2.2 Double extensions

It seems as if one of the corner pieces can be used as follows

$$\tag{4.40}$$

But in order to make a meaningful change in the residual error, the bond dimension of both α and β needs to be enlarged significantly. It is more advantageous to introduce yet another level δ, which forms the link between the 2 parts. As both corner tensors can be optimised at once, the total bond dimension is lower than for the previous suggestion, but still larger than the dimensions of the other loop levels.

$$\begin{array}{c}\text{(diagram)}\end{array} \tag{4.41}$$

4.3.2.3   Larger loops

One question which comes to mind is where the focus should be for the construction. Making blocks for all possible single loop extensions comes at an increasing cost in total bond dimension. From a physics point of view, the model will be better approximated when the smallest non-solved patch is solved by introducing new blocks. On the other hand, the already included blocks may cause an error which was not present before the blocks were introduced. One example is a linear chain which closes upon itself, in for instance a 2 by 3 rectangle. Another example are + blocks which connect upon themselves in the following shapes

$$\begin{array}{c}\text{(diagram)}\end{array}. \tag{4.42}$$
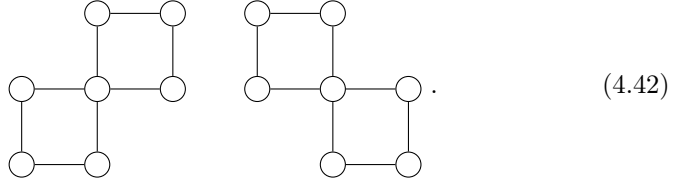
One way to solve this, without breaking the rotation pattern of β and γ, is

$$\begin{array}{c}\text{(diagram)}\end{array} \tag{4.43}$$
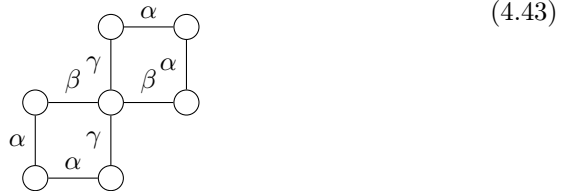
Many more blocks were tried, such as blocks to solve

$$\begin{array}{c}\text{(diagram)}\end{array}. \tag{4.44}$$

Many expansions were tried and failed, including the double extensions and larger loops discussed here. In section 6.4.5.2, it is explained why the results of these additions were not very reliable.

4.3.2.4 Bond dimension

While for the linear blocks it is clear what the bond dimension should be, this is not the case for the graphs with loops. This problem of the ranks cannot be directly solved for tensor ring decomposition [36], but needs to be deduced during the construction. In practice, a bond dimension of 6 is enough to fully solve eq. (4.35) (for physical dimension 2). For eq. (4.38), at least bond dimension 8 is required for β and γ level. This is also sufficient to solve longer extensions and multiple extensions from one corner.

4.4 Symmetry

When the linear blocks are constructed without symmetry considerations, quite some blocks are required. For any one of the 4 legs, the number can range from 1 to M, with M the maximum virtual bond dimension. This results in $(M + 1)^4$ blocks. With the solver presented in the next chapter, this can be done. Another possibility is to restrict eq. (4.25) further by imposing rotation symmetry of the PEPO legs:

$$\underset{1}{\bigcirc\!\!-\!\!-} \;=\; \underset{1}{-\!\!-\!\!\bigcirc} \;=\; \overset{\bigcirc}{\underset{1}{\big|}} \;=\; \overset{1}{\underset{\bigcirc}{\big|}} \tag{4.45}$$

In this way, only the blocks unique up to a permutation of the legs need to be solved. More generally, the following rotation symmetry on the PEPO level is imposed

$$
\begin{array}{c}
\overset{b}{\underset{j\,d}{\underset{a}{\diagup\!\!\!\bigcirc\!\!\diagup}}}{}^{i}_{\;c}
\end{array}
\;=\;
\begin{array}{c}
\overset{c}{\underset{j\,a}{\underset{b}{\diagup\!\!\!\bigcirc\!\!\diagup}}}{}^{i}_{\;d}
\end{array}
\;. \tag{4.46}
$$

4.5 Conclusion

The cluster expansions are introduced here for both a 1D and 2D setting. This can be represented with some very compact notation, as explained in eq. (4.9). The best 1D type, namely A, was used as an inspiration for 2D cluster expansion. For 2D, mainly the linear blocks, loops and single extension will be important in the results.

Chapter 5

Framework implementation

> Experience is the name everyone
> gives to their mistakes.
>
> ———————————————
>
> Oscar Wilde
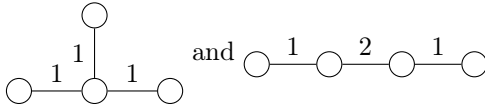
5.1 Introduction

The construction was explained in the previous chapter, but there is a lot of work that needs to be done to go from an idea to a working implementation in MATLAB. The framework to do these calculations was programmed starting from scratch during the course of this thesis. This chapter aims to show some work involved. First, the solvers, used to numerically extract the new blocks introduced for each equation in the previous chapter, are explained. There are 3 different solvers: a linear solver based on matrix inversion, a nonlinear solver based on MATLAB's fsolve and a sequential linear solver, which iteratively solves each occurring tensor with the linear solvers takes a step in that direction. The solver need to be as fast as possible, numerically stable and accurate. The optimisation section gives more details about the framework in general. A section is dedicated to the code for generating the phase diagrams reported in the results chapter. The source code is freely available, and section 5.5 shows very briefly how the code is structured. Finally, some limitations and possibilities relating to the framework are discussed.

5.2 Solvers

Solving the blocks introduced in the previous sections need 2 different approaches. The graphs of the maps can be split in 2 groups. The first one, considers problems where there are no loops and at most one node with more

than 2 legs Examples include: and (see section 4.3.1). These will be reduced to a standard matrix problem, and solve with matrix (pseudo-) inversion. The other group, of course, constitutes the nonlinear problems. This includes every problem where a block (or rotated version) occurs more than once, problems which include loops, ...
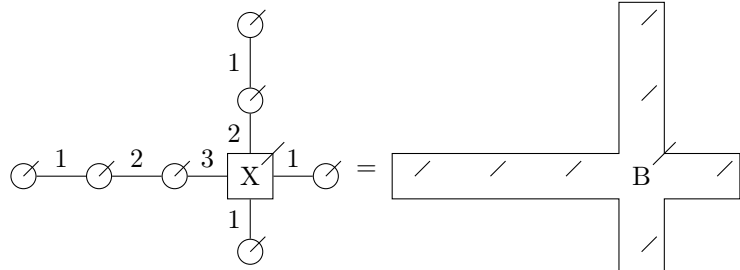
5.2.1   Linear solver

The linear solver is a general purpose block solver which reduces the problem to a set of linear matrix equations. The linear have a tree structure, where the new block is the root of the tree, and all the branches need to be inverted. Let $I^m = (i_1^1 i_2^1 \cdots i_{n_1}^1)$ be all the physical indices of one leg m and α^m the index between leg m and unknown matrix X. Then the problem can in general, after some tedious tensor leg bookkeeping, be rewritten in the following form

$$A^1_{I^1 \alpha^1} A^2_{I^2 \alpha^2} \cdots A^m_{I^m \alpha^m} X_{\alpha^1 \alpha^2 \ldots \alpha^m j} = B_{I_1 I_2 \cdots I_m j} \qquad (5.1)$$

Here i_N^M has the following meaning: M numbers the different legs or branches of the tree, N number of sites of the leg and i numbers the bra and ket states and has dimension d^2. Hence, the bond dimension of $I_n = d^{2n_m}$.

An example of this procedure is

(5.2)

(5.3)

$$
\begin{array}{c}
\includegraphics \\
\end{array}
\quad X - j \;=\; B - j \; . \tag{5.4}
$$

On the first line, the original problem is shown. Block X is the newly introduced block. The right-hand side B is the residual error, i.e. the exponentiated Hamiltonian minus all possible contractions in the network. The second line reorders and groups the physical indices in legs, and transforms the right-hand side accordingly. The third line shows the tensors in the form of eq. (5.1), including the matching labels.

5.2.1.1   Full inverse

One way of solving this equation is by performing by contracting the grouping the matrices A^i to one matrix $A = A^1 \otimes A^2 \cdots \otimes A^m$. The equation to solve is

$$
\begin{aligned}
&A_{I^1 I^2 \dots I^n \alpha^1 \alpha^2 \dots \alpha^m} X_{\alpha^1 \alpha^2 \dots \alpha^m j} \\
&= B_{I^1 I^2 \dots I^m j}.
\end{aligned} \tag{5.5}
$$

The fastest way to solve this system is by using a linear solver. This results in some numerical problems. This is a result of the potentially ill conditioned inverses of legs $A^i i$, inherent to the construction. A pseudoinverse of the full matrix can be easily obtained and resolves this issue (see for numerical example section 6.2.2). The pseudoinverse A^+ of matrix A is calculated as follows:

$$
A = U \Sigma V^\dagger \tag{5.6}
$$

$$
A^+ = V \Sigma^+ U^\dagger, \tag{5.7}
$$

where for Σ^+ all the non-zero diagonal elements are replaced by their inverse. In the numerical pseudoinverse, every singular value below a given threshold e.g. $\sigma_0 = 10^{-12}$ is set to zero. Linear solvers that perform a pseudoinverse also exist. The problem with this full inverse is that the bond dimension increases very fast: matrix A has dimension $d^{2 \sum_m n_m} \times d^{2 \sum_m n_m}$. Although using a linear solver instead of full inversion is considerably faster, this method still becomes computationally infeasible for large systems.

5.2.1.2   Sequential inverse

A second method consist of solving the following sequence of linear problems
one leg at a time

$$A^1_{I^1\alpha^1} X^1_{\alpha^1 I^2 \dots I^m j} = B_{I_1 I_2 \dots I_m j}$$
$$A^2_{I^2\alpha^2} X^2_{\alpha^1 \alpha^2 I^3 \dots I^m j} = B^1_{\alpha^1 I_2 \dots I_m j}$$
$$\vdots$$
$$A^m_{I^m\alpha^m} X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B^{m-1}_{\alpha^1 \alpha^2 \dots \alpha^{m-1} I_m j} \ .$$

$$(5.8)$$

Here, X^1 is the contraction of tensor X with legs A^2 till A^m. In the next step,
leg A^2 is inverted. This continues till the last leg is inverted. While this method
is very quick and scales well, in practice it results in an unstable scheme. Solving
sequentially, the errors of the pseudoinverses (or worse full inverse) accumulate.
If there are 4 legs, the threshold needs to be set at $\sigma_0 = \sqrt[4]{10^{-12}}$. The inverse
now becomes a bad approximation of the problem, rendering the results useless
(but stable).

5.2.1.3   Sparse full inverse

Luckily the problem can be resolved by first performing an SVD decomposition
of $A^m_{I^m\alpha^m} = U^m_{I_m\beta^m} S^m_{\beta^m\gamma^m} V^{m\dagger}_{\gamma^m\alpha^m}$ matrices, with S diagonal and U and V unitary.
All the U^m matrices can be inverted by applying the Hermitian transpose to
the corresponding leg m of B. The tensor $S = S^1 \otimes S^2 \cdots \otimes S^m$ is very sparse
and can be (pseudo)-inverted at once. For a full rank construction, S is already
diagonal. For truncated constructions (or inverses involving loops), this is no
longer the case. The last step consist of applying all the matrices V^m to the
right-hand side. This is shown in this equation

$$A^1_{I^1\alpha^1} A^2_{I^2\alpha^2} \cdots A^m_{I^m\alpha^m} X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B_{I_1 I_2 \dots I_m j} \qquad (5.9)$$
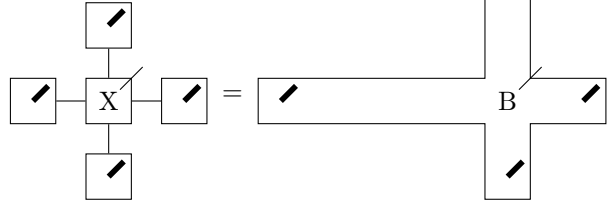$$U^1_{I_1\beta^1} S^1_{\beta^1\gamma^1} V^{1\dagger}_{\gamma^1\alpha^1}$$
$$U^2_{I_2\beta^2} S^2_{\beta^2\gamma^2} V^{2\dagger}_{\gamma^2\alpha^2} \cdots$$
$$U^m_{I_m\beta^m} S^m_{\beta^m\gamma^m} V^{m\dagger}_{\gamma^m\alpha^m}$$
$$X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B_{I_1 I_2 \dots I_m j} \qquad (5.10)$$
$$S_{(\beta^1\beta^2\dots\beta^m)(\gamma^1\gamma^2\dots\gamma^m)}$$
$$V^{1\dagger}_{\gamma^1\alpha^1} V^{2\dagger}_{\gamma^2\alpha^2} \cdots V^{m\dagger}_{\gamma^m\alpha^m}$$
$$X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B'_{(\beta_1\beta_2\dots\beta_m)j}. \qquad (5.11)$$
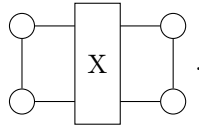
The complexity is determined by the SVD decomposition of the individual legs.
Due to its sparsity, S does not take much space to construct and is quite fast
to pseudoinvert. Doing the pseudoinverse at once means that it has the same
precision as the full pseudoinverse, as desired. It is also possible to take a
pseudoinverse of a matrix with a QR-decomposition, which is faster [37]. The
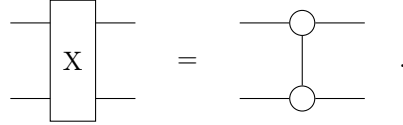
matrices Q^i could be inverted directly, and the $R = R^1 \otimes R^2 \cdots \otimes R^m$ matrix is still upper triangular, allowing for back-substitution. This method is not used because of the memory requirements to store this matrix are larger, and the triangular matrix R is not in the correct form for the linear pseudoinversion solver.

5.2.1.4 Extension

The linear solver is made for linear problems. Nevertheless, it can solve every local patch appearing in a map [1], such as 2 neighbouring sites. These sites are split using an SVD decomposition. A problem could e.g. be defined as

$$. \tag{5.12}$$

The solver treats this as a linear problem with 2 legs. After that block X is solved, it is decomposed as

$$. \tag{5.13}$$

Another example is the following corner block $O^{1\gamma\beta0}$. It may not seem like a linear problem, but it can be solved with the linear solver without any problem

$$. \tag{5.14}$$

The algorithm will split this problem in $m = 2$ legs, treating the loop as one.

5.2.2 Nonlinear solver

In some cases, the above solver does not return the best possible solution to a given problem. The reason is that it is not able to incorporate symmetries or solve problems where the new blocks appear more than once. A new solver is needed which does not rely on methods from linear algebra, but on more general nonlinear least squares solvers.

In essence, the nonlinear least squares solver needs as input a vector with the error values $\vec{f}(\vec{x})$, and if possible also the Jacobian, i.e. $J_{I,J} = \frac{\partial f_I}{\partial x_J}$. This info is used to choose a direction and a step size, minimising the error. An improved point x is chosen by the algorithm, until some convergence criterium is reached. The implementation uses MATLAB's fsolve routine, which uses the Levenberg-Marquardt algorithm under the hood.

---

[1]Gives tensor X with

5.2.2.1   Automatic differentiation

With some care, the Jacobian can be calculated for a general TN in an automated way. It amounts to contracting the network with the tensor x_J removed, and treating the non-contracted indices as external ones. This becomes clearer upon inspection of eq. (5.4). $\frac{\partial B_{I^1 I^2 \ldots I^m j}}{\partial X_{\alpha_1 \alpha_2 \cdots \alpha_m j}} = A_{I^1 I^2 \ldots I^m \alpha_1 \alpha_2 \cdots \alpha_m}$. If a tensor appears in multiple places, the sum rule for derivatives has to be used.

5.2.2.2   Symmetry

The nonlinear solver can handle rotated and permuted blocks. For instance, a simple loop (square) can be solved by rotating one tensor $O^j_{\alpha\alpha 00}$ 4 times, once for every corner. The Jacobian is now calculated by applying the chain rule. Another example where this is useful is given in eq. (4.45).

5.2.2.3   Combining problems

As only solver, the nonlinear solver can solve multiple (non-neighbouring) tensors at once, and also do this for multiple geometries at once. This does however result in slow solving times.

5.2.3   Sequential linear solver

While from the previous section it seems that all nonlinear problems need to be solved with the nonlinear solver, this is in fact not the case. This solver takes as input multiple new tensors, and solves them one by one. As this is not truly a linear system, the error will not be zero after one pass. But solving the tensors repeatedly lowers the error at each step, giving an iterative procedure. This procedure can be sped up by reusing some parts of the calculations involved in the linear solver. For example, the exponentiated Hamiltonian and contraction of all virtual levels that do not involve the optimised blocks only needs to be performed once.

   The step is chosen as follows: suppose X is the current tensor and X' the newly computed one. Then the tensor is updated as follows: $X \leftarrow X + \alpha(X' - X)$. If the error has increased, the step is made smaller. The algorithm stops after a number of steps or when a certain threshold is reached.

5.2.4   Conclusion

The framework is equipped with 3 different solvers, designed to solve different problems. The code below shows how they are called from within the code:

```
[obj, ln_prefact, err] = solve_lin_and_assign(obj, map, {pattern},
                ln_prefact, struct);
[obj, ln_prefact, err] = solve_sequential_lin_and_assign(obj, map, {pattern},
                ln_prefact, struct, {rot_90});
[obj, ln_prefact, err] = solve_non_lin_and_assign(obj, {map}, {pattern},
```

```
ln_prefact, struct, {rot_90});
```

They only need a map, which is the geometry of the problem, and a pattern, i.e. the new block to add. `rot_90` is an optional argument, listing all the permutations (such as rotation symmetry over 90 degrees). For completeness, `ln_prefact` is the normalisation factor as will be discussed in the next section 5.3.3.

Whenever the linear solver can be used, it is the solver of choice. The sparse full inverse procedure is fast and handles the ill-conditioned inverses very well. The sequential linear solver builds on this solver to handle the introduction of multiple new tensors, possibly related to each other through a permutation. The nonlinear solver is at the moment only the fastest for small highly nonlinear problems, such as solving eq. (4.35) in a rotation invariant manner. The nonlinear solver can optimise multiple problems at once, and could be extended to fully use internal symmetries of the model.

5.3 Optimisation

5.3.1 Bookkeeping

One important aspect of programming these general solvers is to devise a scheme that keeps track of all the involved tensors and transforms to problem to the form described above. In the code, the geometric info is represented by a map. This keeps track of the neighbours for each site, the numeric indices of the internal and external legs and a list to perform the contractions. The solvers make only use of this info. The advantage is that for other geometries such as cyclic maps, or even other lattices, only the code to generate the maps need to be extended. These maps are used throughout the framework: to calculate the exponentiated Hamiltonians, contract the tensor networks,...

5.3.2 Fast contraction

One particular task is to determine all the possible combinations of virtual levels for a given geometry. Simply looping over all possible combinations scales as n^m, with n the number of virtual levels and m the number of internal legs. This quickly becomes a bottleneck. This problem can be restated as a PEPS contraction in the following way: for each site make a tensor $T^i_{\alpha\beta\gamma\delta}$ where i encodes all the non-empty combinations of legs ($\alpha\beta\gamma\delta$). On each site, the right boundary conditions need to be applied to get the right geometry. After setting the boundary conditions, the sparse PEPS network can be contracted and the resulting tensor gives, after decoding, all the possible contractions. Due to its sparsity, this performs quite fast. As an added bonus, removing a tensor from T gives all contractions without this tensor. As both results are sorted, the subset of contractions containing a given tensor can also be found fast.

5.3.3   Normalisation

For many of the end results, the PEPO cells can be divided by a normalisation factor. Normalising the calculations is important, because $\exp\left(\hat{H}\right)$ scales exponentially in the number of sites. Luckily, the exponential can be calculated directly in normalised form. Suppose H is the matrisation of the Hamiltonian evaluated for a certain geometry. This is a Hermitian matrix and can be diagonalised $H = QDQ^\dagger$ with Q unitary. Then

$$exp(H_{d^N} - N\alpha I) = Q exp(D - N\log(\alpha)I)Q^\dagger \tag{5.15}$$

$$= Q \begin{bmatrix} exp(D_{11} - N\log(\alpha)) & & \\ & \ddots & \\ & & exp(D_{d^N d^N} - N\log(\alpha)) \end{bmatrix} Q^\dagger \tag{5.16}$$

$$= \frac{exp(H_{d^N})}{\alpha^N}. \tag{5.17}$$

With I the unit matrix. Next to a global normalisation factor, every block calculation calculates a specific normalisation factor such that the largest eigenvalue of $exp(H)$ is of the order 1. The cut-off for pseudoinversion σ_0 is applied on the normalised problem.

5.3.4   Internal representation

Two main internal representations are used to construct the given MPO/ Projected Entangled Pairs Operator (PEPO). Either, it is stored as a cell of tensor, one per combination of virtual levels, or as one big tensor where the blocks are added to during the construction. For sparse types, a multidimensional sparse tensor can be chosen as format. Given that MATLAB doesn't support multidimensional matrices by default, this[38] library is used.

5.3.5   Even faster inverses

While the inversion procedure above states how to make use of pseudoinverses, it was not yet clear in the 1D case it was needed. The 1D implementation uses a trick to get all the inverses for free from the SVD decomposition. Take the MPO which corresponds to a unitary matrix:

$$\cong U^n_{\alpha(ij\beta)} \tag{5.18}$$

Then the inverse MPO can be calculated by taking its Hermitian conjugate and reshaping.

$$
\beta \underset{j}{\overset{i}{\left(O_n^{-1} \right)}} \gamma \quad \cong U^{n\dagger}_{(ij\beta)\gamma} \tag{5.19}
$$

The inverse of the chain can be computed with a tensor contraction

$$
\alpha \left(O_n^{-1} \right) \left(O_m^{-1} \right) - \cdots - \left(O_1^{-1} \right) 0 \quad . \tag{5.20}
$$

The physical indices need to be contracted with the corresponding indices of the tensor to apply the inverse to.

5.3.6 Buffering Results

Some calculations, such as calculating the matrix exponential, take some time. In 1D code, the same calculations were performed over and over again, and hence a buffer mechanism was written to store these results. In the 2D framework, this not necessary as the solvers only calculate the matrix exponential once and return the blocks together with the made error.

5.3.7 Profiling

To get a sense of the speed, constructing a 2D PEPO up till order 6 with level 3 truncated at bond dimension 20 with loop extensions takes about 25 seconds on my PC. The most time intensive processes involve performing the contractions. For larger systems, the time limiting factor is calculating the exponential of the Hamiltonian.

5.3.8 Calculating the error

Every solver returns the residual error for the new block. This comes at almost no cost, because all the calculations are already done during the solving procedure.

5.4 Calculating phase diagrams

This section details how the phase diagrams are calculated, stored and the critical parameters fitted. The results are discussed in section 6.4.

5.4.1   Points sampling

This concerns the problem which temperatures to select to calculate the phase diagram. As the transition between 2 phases is sharp, a uniform sampling in T is not the best option. A very fine grid is needed to capture the transition well, requiring high computation times. The sampling starts by calculating the magnetisation for N uniformly distributed sample points between 2 temperatures. These calculations are performed in parallel on a multicore server. When they are all finished (or have run for a maximum amount of iterations), all the arch lengths are calculated, and a new T point is repeatedly inserted in the largest interval until N new points are selected. The desired arch length $\Delta S^2 = \Delta m^2 + r\Delta T^2$ between points in the m-T plane can be set, and calculations continue until the goal is reached.

5.4.2   Storing the information

Each run has a template with all the common model info. For each point 2 files are stored. One file contains the info and results, such as temperature, magnetisation, correlation length, etc. The other file is much larger and contains the PEPO tensor, the calculated VUMPS environments, ... The files of the first kind are used in other calculations, such as the fitting procedure. Reassembling the files into one structure happens in a central function. Another function is able to reprocess already calculated points. The sampling can be continued from where it was last stopped.

5.4.3   Fitting

The code performs a finite-size scaling as explained in section 3.2.5. The fitting procedure works as follows: a function f_X defined by a limited number of parameters is made for every observable $X \in \{m, \xi, S\}$. The parametrisation is chosen such that it has the right scaling behaviour, and the analytical derivative is known. The code performs a nonlinear optimisation, where the error is either the vertical distance to f_X or the orthogonal distance. The fitted function and parameters are determined simultaneously. The optimisation runs for a limited number of cycles. Afterwards, a random displacement is made to the parameters of the current best fit. This is repeated until convergence. The code to perform this collapse was originally written by Bram Vanhecke. The adapted version includes the possibility to fit the subleading corrections and c_i to calculate δ.

5.5   How to use

All the code needed to generate all the results from this dissertation is available on my GitHub page `https://github.com/DavidDevoogdt/Thesis_Tensor_Networks`. The starting points to explore the code are in the readme file. The most important folder is `src_2D`, as the 1D code is on all levels inferior. This is mainly due to the lack of pseudoinversion.

5.5.1 Source code structure 1D

The implementation of the different MPO types can be found under `src_1D/generateMPO.m`. It bundles some helper functions such as contracting a chain or cycle of MPOs or construction of an exponentiated Hamiltonian for the given input Hamiltonian. Another example is constructing the inverse by sequential inverse MPO contraction, ...

`src_1D/test_old.m` contains the code to create the plots to compare different types and orders.

5.5.2 Source code structure 2D

To use the code, go to the root folder and execute `doPath.m` The construction of the PEPO's happen in `src_2D/PEPO_constructions/`. The names coincide with the names in the pdf. The results from section 6.2 and section 6.3 are made with `test.m` and `test_2D.m`. The plotting in 2D happens with `proces_test_2D.m` Generating a phase diagram can be done with `Ising2D_par`. E.g.the $g = 2.5$ transition reported in section 6.4 can be generated with

```
Ising2D_par(8, 2.5, 'g', struct('testing',0,'unit_cell',1,
                                'par',1,'order',5,'do_loops',1));
```

and visualised with `proces_Ising2D.m` Finally, fitting the curves happens with `dofit.m`

5.5.3 VUMPS code

One external package is not checked in on git: MatlabTrack. This is used in `src_2D/process_PEPO/PEPO_vumps.m` to calculate the VUMPS environment. This proprietary code was written by the QuantumGroup@UGent. To obtain the code, go to `https://quantumghent.github.io/software/`.

5.6 Limitations and outlook

In short, all components are in place in the framework to generate easily and efficiently all the blocks.

5.6.1 Implementation

By far the largest amount of time invested in this thesis was creating and testing this framework. Everything (except the fitting code) was written from scratch. Implementation of the maps and solvers in its full generality is a very error-prone problem. In hindsight, the 1D framework may seem unneeded, given that the 2D framework is even better. This is not the case. Constructing the 2D framework was only possible due to the lessons learned (and mistakes made) in the 1D code.

Code quality

The first and most important goal of writing numerical code is of course that it compiles and that the results are as correct. But this is only the first step. The 2D framework neatly orders the different task is functions to avoid as much code duplication as possible. This improves readability and decreases the number of errors as every component is used in many ways.

Size limitation

The main bottleneck is, as expected, calculating the matrix exponential for large systems ($N>14$). For large maps, contracting the PEPO network is an equally expensive operation. The other components are efficient enough to not cause any troubles. In particular, the solvers aren't a limiting factor at this point to go further.

Lattices

The models studied were all on a square lattice. It would be beneficial to be able to simulate other lattices, but also higher dimensions could be included. In essence all the information of the lattice is contained in the maps generated for each calculation, such as all the connected sites, how to contract them, etc. Although undoubtedly many details will need to be changed to use it in practice, the solvers can stay almost the same.

Symmetries

At the moment rotation and permutation symmetries can be included in the construction of the blocks. Internal symmetries are not yet included at the moment. This could push computational boundaries further.

Chapter 6

Results

> With four parameters I can fit an elephant, and with five I can make him wiggle his trunk
>
> _John von Neumann_

6.1 Introduction

This section explains the accuracy and performance of the given TN cluster expansions. The first section compares the different constructions in 1D based on the error relative to the exact solution. The best algorithm will form the basis for the 2D results. First, similar to 1D, the results will be checked based on the error relative to the exact solution. Then, the expansion is used to calculate the phase diagram of the 2D TFI model.

6.2 Results 1D

6.2.1 Exact tensor matrix exponential

The performance of the MPO construction can be compared with the exact diagonalisation of the Hamiltonian for a given number of sites. To obtain a faithful result, the number of sites should be as high as possible. In practice, diagonalisation of large matrices becomes slow and memory consuming. The size grows exponentially in the number of sites: $d^n \times d^n$. A double takes 8 bytes of memory. A rough estimation of the amount of RAM R needed to store this complex array is

$$R = d^{2n} \times 16 \text{ bytes} \tag{6.1}$$

which means a 14 site chain already takes up more than 4 GB of RAM. The complexity to calculate a matrix exponential scales as $O(n^3)$ [39]. In practice

this means that, without any tricks, the matrix exponential can be calculated for 12 sites. State-of-the-art algorithm for exact diagonalisation, which include all symmetries and are optimised for parallelisation, can calculate up to 50 sites. [40]

6.2.1.1   Norms

The Schatten 2 norm is used in the following analysis, denoted by $\| \cdot \|_2$. In the figures the relative error ϵ is reported.

$$\epsilon = \frac{\left\| \boxed{e^{-\beta \hat{H}_n}} - \,^{0}\!-\!\!\bigcirc\!\!-\!\!\bigcirc\!\!- \cdots -\!\!\bigcirc\!\!-\!^{0} \right\|_2}{\left\| \boxed{e^{-\beta \hat{H}_n}} \right\|_2}$$

(6.2)

This norm can only be calculated for a finite number of sites. The influence of the number of sites for a linear and cyclic chain are shown in fig. 6.1. As expected, the cyclic norm represents large systems better for the same number of sites. The linear norm keeps increasing with every added site. Calculating the cyclic norm comes at the extra cost. In this chapter, the cyclic norm will be given for M=11 sites.

6.2.2   Inversion procedure

6.2.2.1   Full pseudoinversion

The procedure for inversion of the linear blocks was detailed in section 5.2.1. To demonstrate the need for a pseudoinverse, the error for TFI models is shown in fig. 6.2. The pseudoinversion has 1 parameter σ_0, the cut-off below which the singular values are set to zero. A value of 10^{-12} seems optimal in the sense that it doesn't introduce large fluctuations, but still is able to produce good inverses.

6.2.2.1.1   Truncation   The original 1D code does not yet have this full inversion procedure. Instead, an optimality criterium was devised to truncate the series. Needless to say, even with this criterium, the results were a lot worse at low beta.

6.2.3   Models

With these basic definitions and findings out of the way, the different cluster expansions can be tested against different physical models.

(a) Cyclic error

(b) Linear error
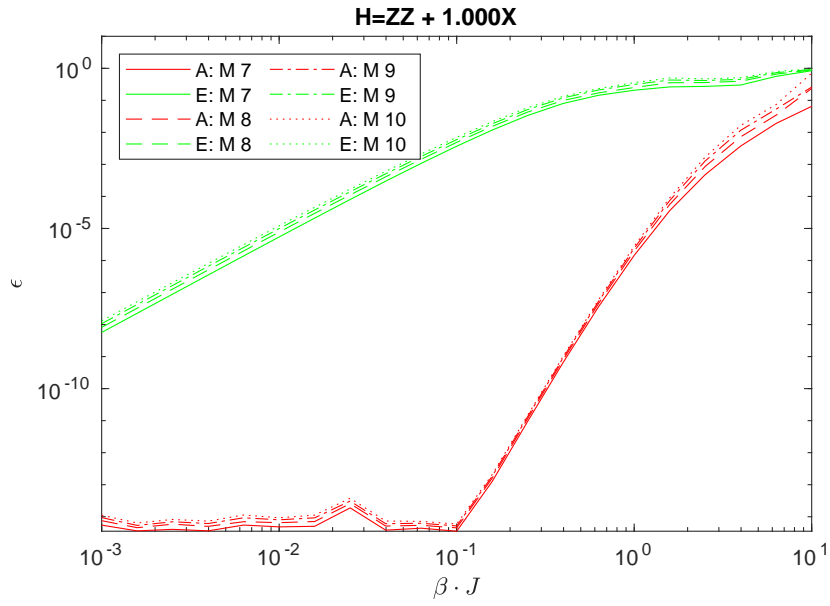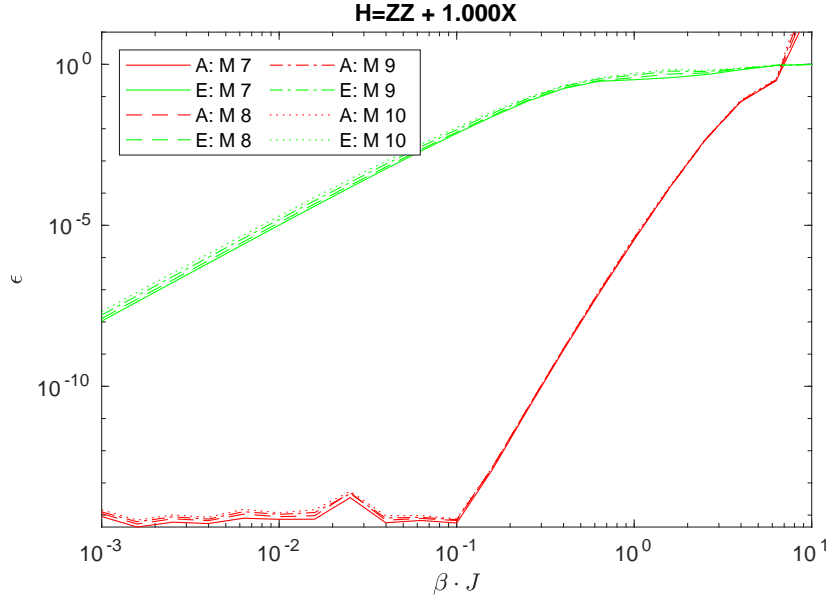
Figure 6.1: Different error measures for 1D TFI model
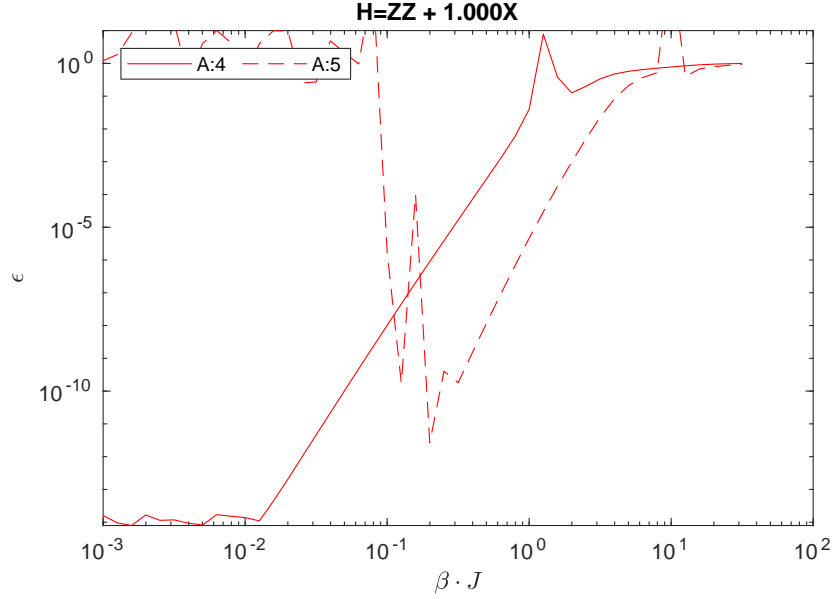
Figure 6.2: Error for TFI, computed with full inversion

6.2.3.1   Ising

The results for the different types are bundled in fig. 6.3. The vertical axis shows the logarithm of the relative error ϵ, horizontal axis the normalised inverse $\beta = \frac{J}{T}$. The most surprising finding is the fact that the strict type E performs worse than the others. For low β, the difference is more than 5 orders of magnitude. This will generalise: the strict variants are not the best choice. Now lets focus on the 2 other variants. Type A clearly outperforms type F for all $\beta < 2$ by quite some margin. Only for large β, type F has the upper hand. While not completely clear in the picture, type A has quite a large error for some orders at $\beta \in (2, 10)$, but higher orders seem to solve the problem. The construction of type F requires twice the bond dimension, and hence type A performs clearly better overall.

6.2.3.2   Heisenberg

Now we focus on the spin 1/2 Heisenberg model on a chain. The results are again displayed in fig. 6.4. The exact type seems to perform a lot better here, but still has consistently the largest error of the 3 methods for a given order. For low β, type A again performs better than type F. At $\beta \approx 1$ and larger, F starts to improve upon the result of A, and doesn't have a divergent error.
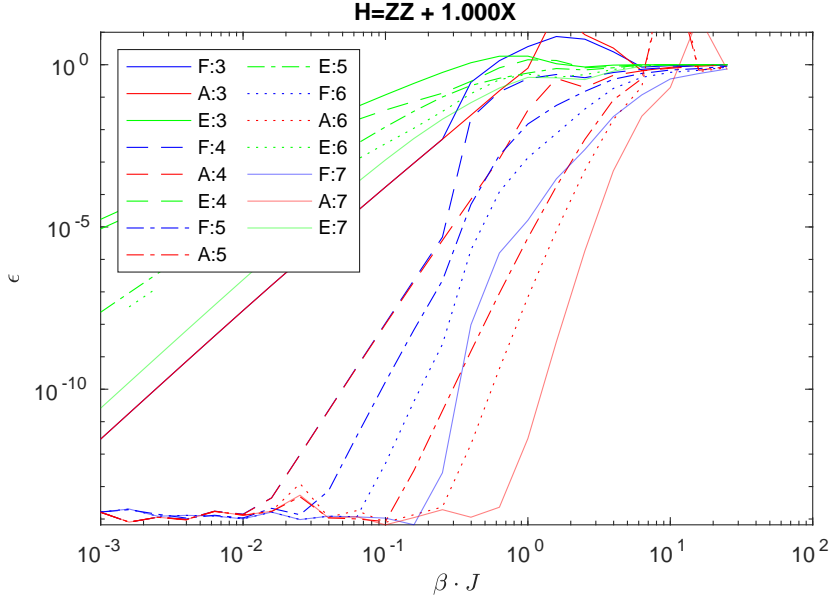
Figure 6.3: Comparison type A, E and F for TFI model.
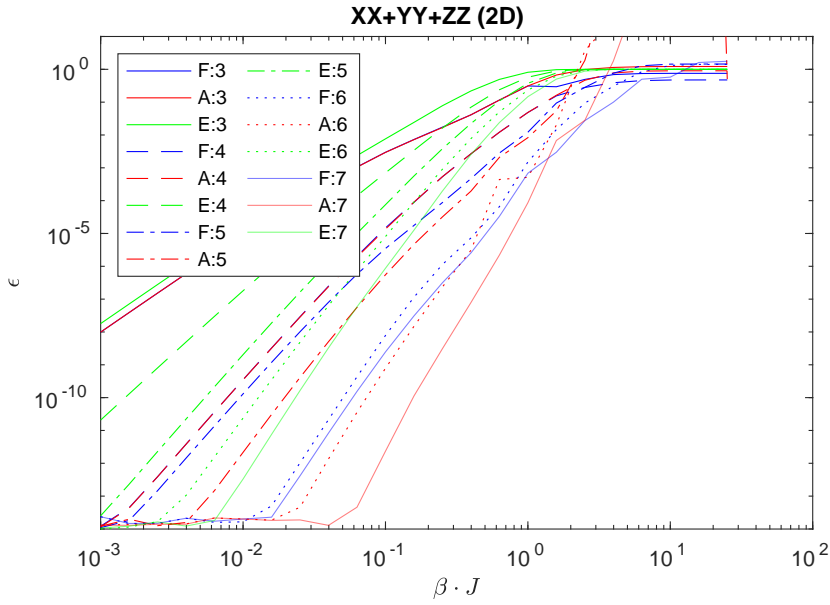
Figure 6.4: Comparison type A, E and F for Heisenberg model.

(a)

(b)
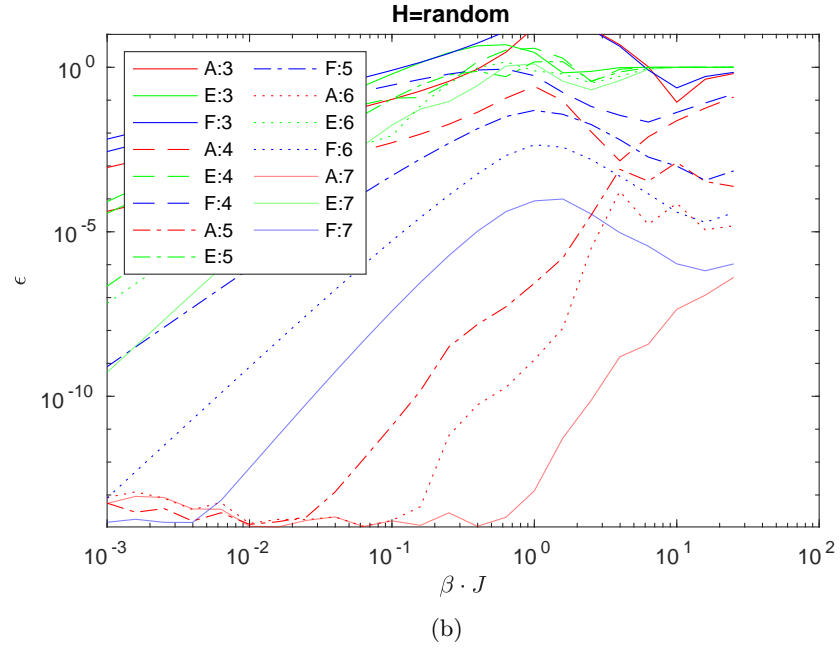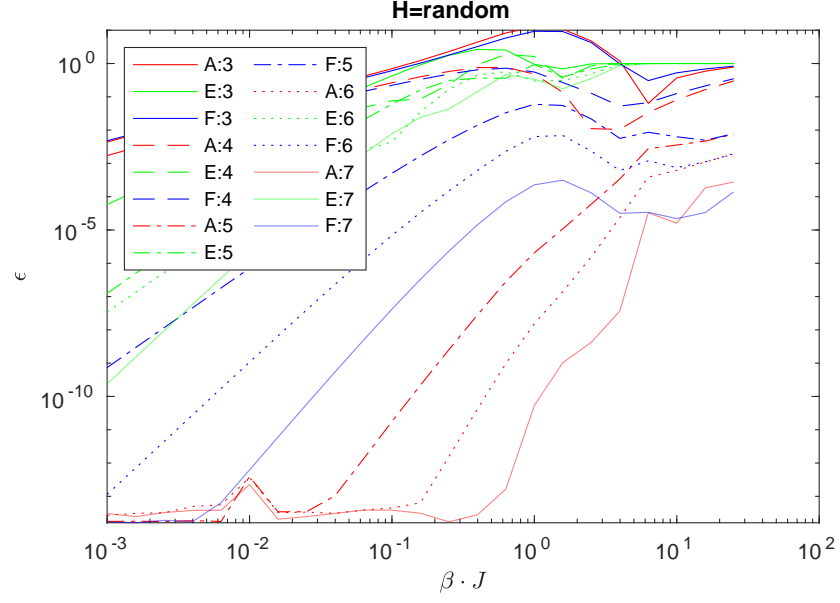
Figure 6.5: Comparison type A, E and F for 2 random generated Hamiltonians.

6.2.3.3 Random

To give a representative overview for random Hamiltonians, 2 simulations were run. The construction for the random Hamiltonians was explained in section 3.3.3. Figure 6.5 shows the results. The results show once again that the strict variant performs worse than the other 2. In comparison to the previous models, type F performs very well, especially at high β. These Hamiltonians were included to ensure that the chosen models are not too 'simple'.

6.2.4 Real time evolution
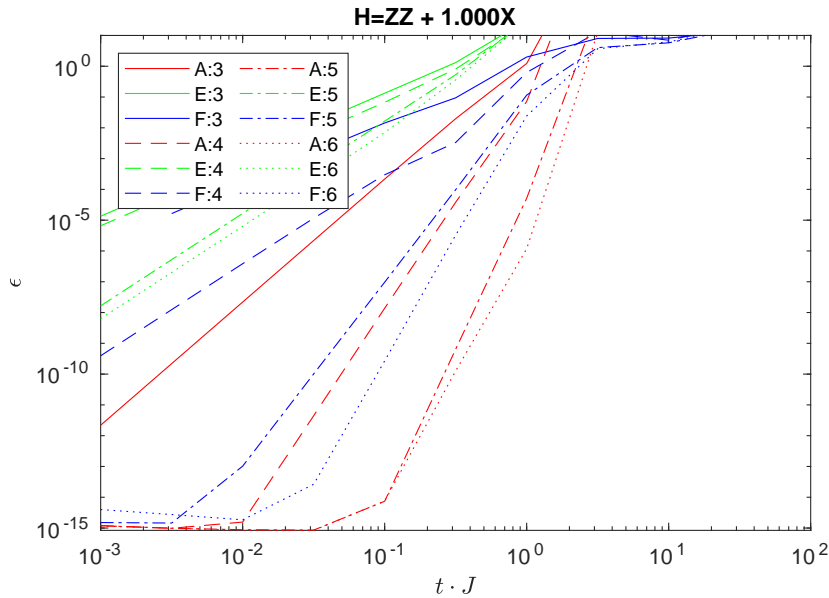
Figure 6.6: Comparison type A, E and F for TFI model. The horizontal axis represents the time step, not the inverse temperature. Virtual level 3 is truncated to $\chi = 20$.

The method can also be used to construct real time evolution. Figure 6.6 shows the error of the matrix exponential in function of the time step $t = -i\beta$. Also here, the results are promising. The construction handles complex matrices well. Similar to the previous conclusion, type A performs better than type F, which performs better than type E. The error of type A becomes quite large for high temperatures (not visible on figure), while type F keeps the errors low. This is not by any means a complete analysis of the real time evolution operator, but just a numeric example.

6.2.5   Conclusion

The biggest lesson, which took some time to figure out, was the correct way to implement the inversion procedure. It is essential that the pseudoinverse is taken for all the legs at once. The different types were tested against each other, and it is clear that type A is better in almost all situations. The strict variants score worst. Type F keeps the inverses well-conditioned, and this shows at large β. Also, real time evolution works well. The truncation procedure works as expected. A longer chain should not be constructed when a shorter chain was not solved fully (e.g. due to a truncation of the previous level). Constructions with explicit rotation symmetry (see section 4.4) perform exactly as good as the ones without.

6.3   Results 2D

The results in 1D are very promising, but the real open challenges are situated in 2 (or higher) dimensions. Therefore, it is an interesting and relevant question whether the results generalise to 2D. On the one hand, a similar error measure as in 1D is determined. For the error measure, we will need to settle for what can be computed within a reasonable amount of time. As a second measure, and from physical point of view the most interesting one, some phase transitions of the 2D TFI model are determined, and compared against values in literature.

6.3.1   Norm

Once again, a suitable norm has to be derived. The situation is more difficult than in 1D, because contracting a PEPO TN has a much larger computational complexity than in 1D. Ideally, the norm would be calculated on an n by n cyclical grid (2D grid on a torus). Here n has to be at least 1 larger than the largest order. In practice, this is not achievable in a reasonable amount of time. The limitations are twofold: the maximum number of sites to calculate the matrix exponential is still around 14, and the contraction of the TN on a torus is limited to 3 by 3. This does not capture the long chains. The limitations on the PEPO contraction can be somewhat relaxed by not computing the full network, but computing the reduced density matrix. Here, most of the sites have their physical indices traced out, except for one site (see eq. (2.35)). The

final error measures are defined by

$$\rho_{i,j}^1 = \qquad (6.3)$$

and

$$\rho_{i,j}^2 = \qquad . \qquad (6.4)$$

For the first one, all the blocks in the expansion can be fitted into the geometry. The norm is cyclical in both x and y, given less biased results as in 1D. The second reduced density matrix is not cyclic, but includes more loop type contributions. Both relative errors will be used, defined by

$$\epsilon^\alpha = \frac{\left\|\rho_{exact,i,j}^\alpha - \rho_{i,j}^\alpha\right\|_2}{\left\|\rho_{exact,i,j}^\alpha\right\|_2} \quad \alpha \in [1,2]. \qquad (6.5)$$

For the given error measures, cluster expansions up till order 5 can be tested. The second norm takes the most time to compute, due to the larger complexity of contracting the TN.

6.3.2 Models

6.3.2.1 Ising

For this test, the cluster expansion is of order 5, and the transversal field is of magnitude $g = 2.5$. The results can be seen in fig. 6.7. There are 3 different constructions: without loops, only plaquette term (eq. (4.35)) and with loop extensions from one corner (eq. (4.39)). This is tested for both norms (see
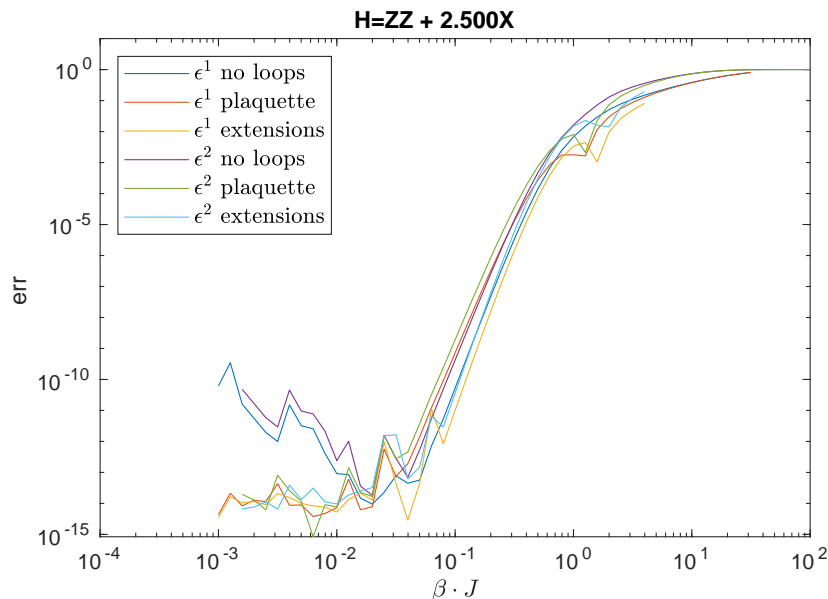
Figure 6.7: The errors ϵ^i for 2D TFI model.

eq. (6.5)). The plaquette term (eq. (4.35)) is mainly important at low β. It also improves the error at $\beta \approx 1$ considerably. The extensions improve the results slightly, but at large cost in bond dimension. Both norms show the same trend, with norm ϵ^2 being somewhat more strict. Of course, the real norm for an infinite lattice is even more restrictive than both norms calculated.

6.3.2.2   Heisenberg

The results for the harder to simulate Heisenberg model are shown in fig. 6.8. The conclusion is once again that the error $\epsilon^2 > \epsilon^1$ most of the time. The loops are absolutely a big improvement for the results at low β. The loop extension also help to lower the error considerably.

6.3.3   Conclusion

It is interesting to compare the no loops errors from 2D with (fig. 6.3 A:5) and (fig. 6.4 A:5), the equivalent 1D errors. They roughly match up, as can be expected. A similar observation can be made for lower order constructions. This places the additional improvement of the error due to single extensions into context: they make an improvement, but one can expect the higher order expansion with same bond dimension to be just as successful. In general, the results above show that the cluster expansions are also promising in 2D setting. The fact that the version with plaquette extensions works well also opens up the
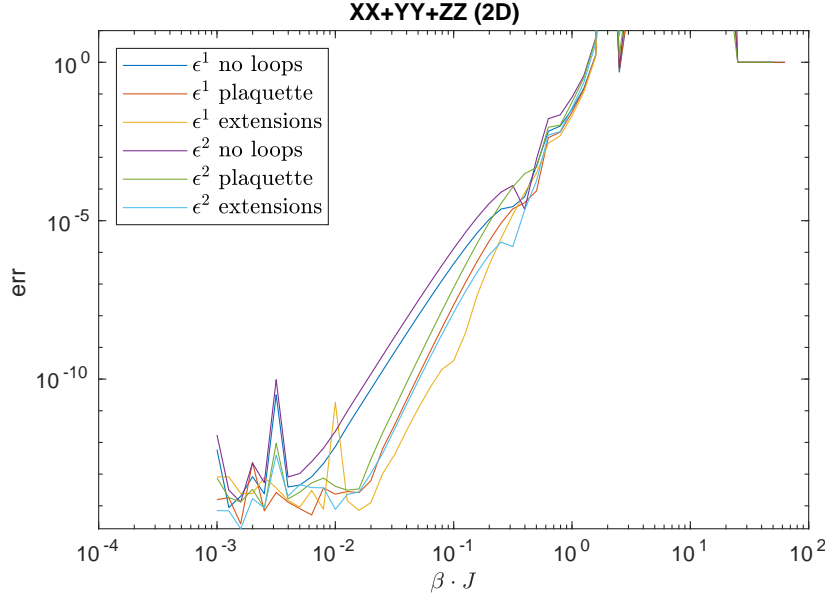
Figure 6.8: The errors ϵ^i for 2D Heisenberg model.

pathway for simulations in higher dimension. The computation of the 'linear' blocks scale with the number of legs. The generalisation from 2x2 plaquette term to 2x2x2 cube should still be feasible, as the solvers are cable of handling systems with 8 sites.

6.4   Phase diagram 2D Transverse Field Ising model

Now we have an idea how accurate the cluster expansions is, we can use it to calculate the thermodynamic properties of the TFI model. The sampling of points was explained in section 5.4. As a reference, fig. 3.2 is shown once more in fig. 6.9 At the relevant temperature, an order 5 series expansion without loops is sufficient. This keeps the bond dimension small and hence the environment can be computed faster with VUMPS.

6.4.1   Classical Ising phase transition

The classical Ising model on a square lattice has an exact solution. Onsager calculated the critical temperature to be $T_c = \frac{2J}{k \ln(1+\sqrt{2})} \approx 2.69185 J/k$. The units are normalised with $J = k = 1$ in the remainder of this chapter. A test for the construction is to simulate this phase transition (see section 5.4.3) and compare the fitted results. The simulated data points are shown in fig. 6.10. Each of the figures has 4 different subplots. The left upper corner shows m vs
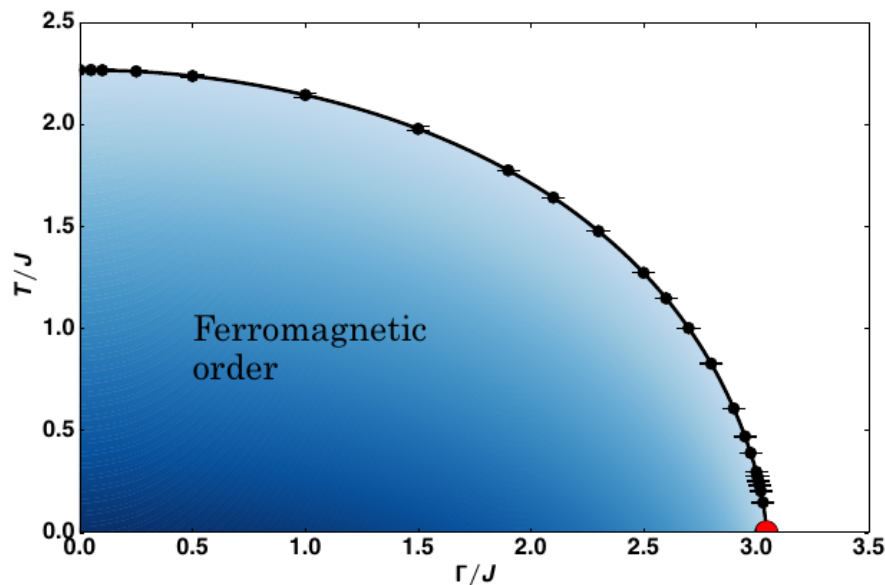
Figure 6.9: Phase diagram for 2D TFI model. Figure taken from [22].

T. For low T, there is clearly a non-zero macroscopic magnetisation, while high T has a zero expectation value as expected. The other three plots show the data collapse of the finite-size scaling of the entanglement entropy S, the correlation length ξ and the magnetisation m. δ is the Marek gap, a measure for the system size as explained in section 3.2.5. Near criticality, they collapse well, but away from the critical point there is quite some systematic variation. The reason is shown in fig. 3.1. Only in some limited range around the critical region, the universal scaling holds. A more zoomed in version is shown in fig. 6.11. Clearly, the data collapses a lot better as expected.

In fact, the data collapses so well that the critical exponents and the temperature could be determined with the given data. The numerical fit (with starting point away from critical temperature), results in a numerical value of $T_c = 2.691(9)$.

6.4.2   g=2.5 phase transition

The previous example simulated the classical 2D Ising model. The question is whether this result will carry on into the quantum regime. The results for the full phase diagram are shown in fig. 6.12 and points in the direct neighbourhood of the phase transition are shown in fig. 6.13, similar to the classical case. Of course, quantitative details are different: the phase transition happens at $T \approx 1.274$ and the maximum magnetisation is lower than 1, in accordance to fig. 3.2. The variation of m around the critical temperature for different bond
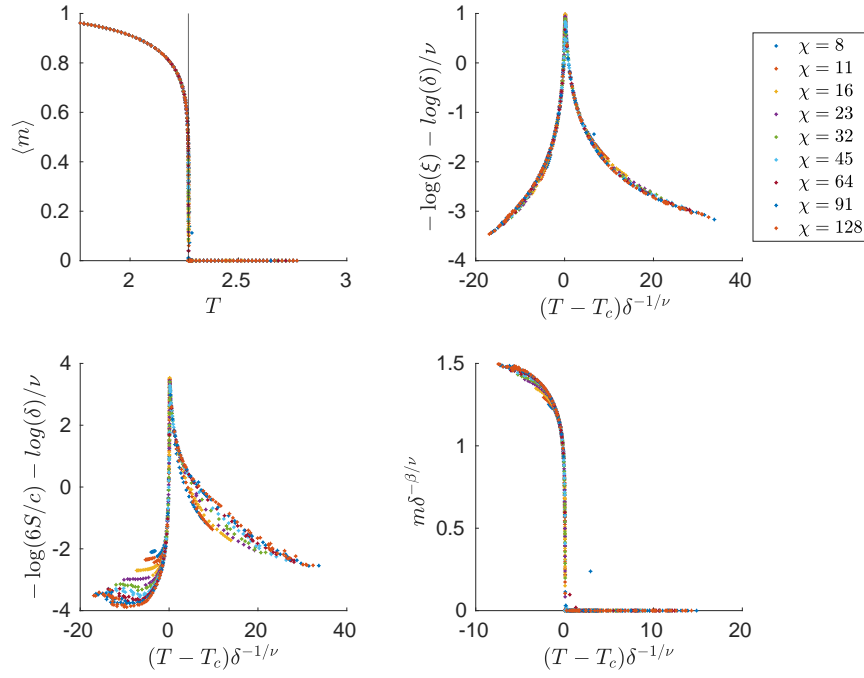
Figure 6.10: Data collapse for $g = 0$ phase transition of TFI Model.
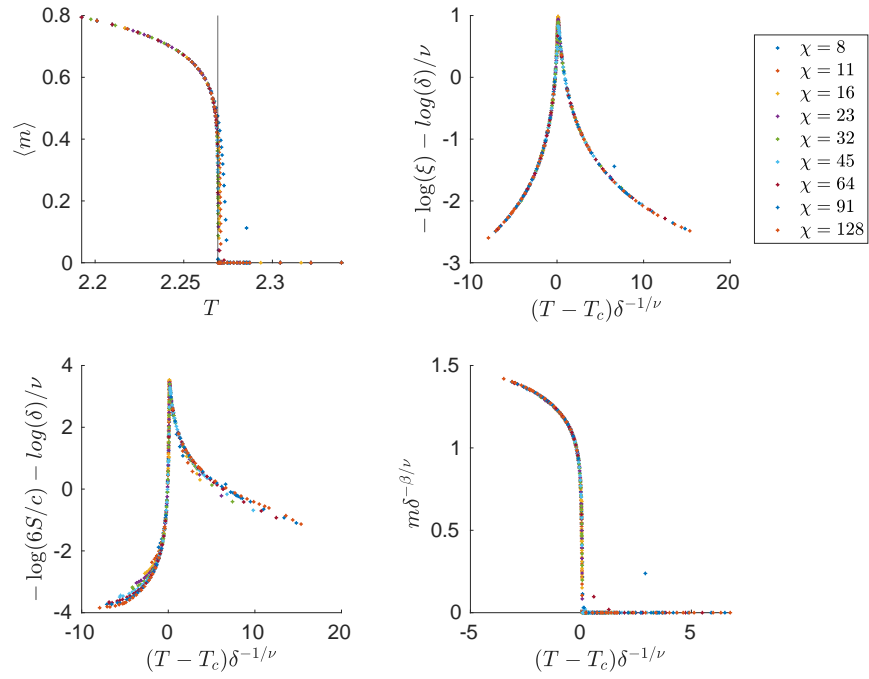
Figure 6.11: Data collapse for $g = 0$ phase transition of TFI Model. Points are taken from $T \in [T_c - 0.08, T_c + 0.08]$
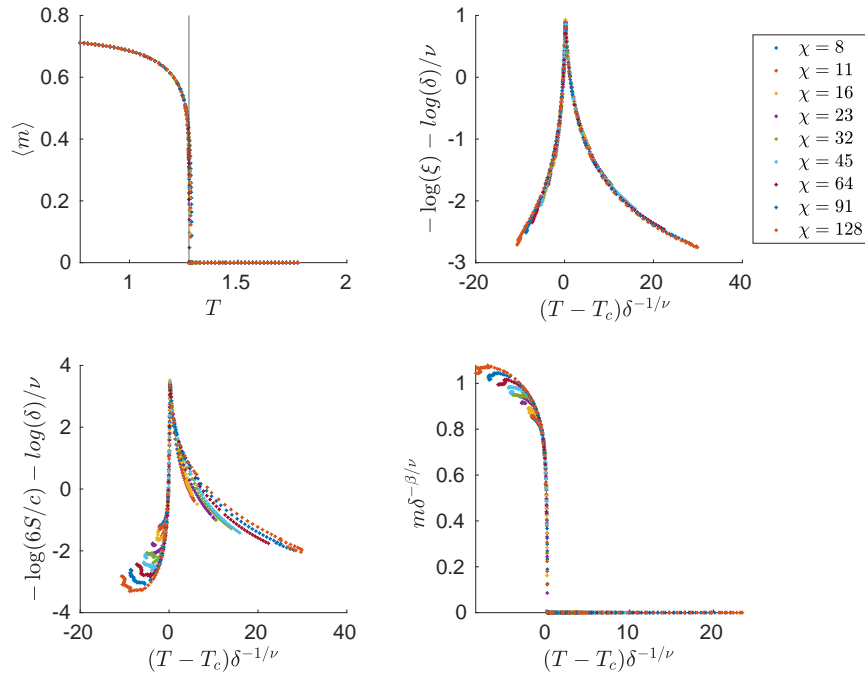
Figure 6.12: Data collapse for $g = 2.5$ phase transition of TFI Model.
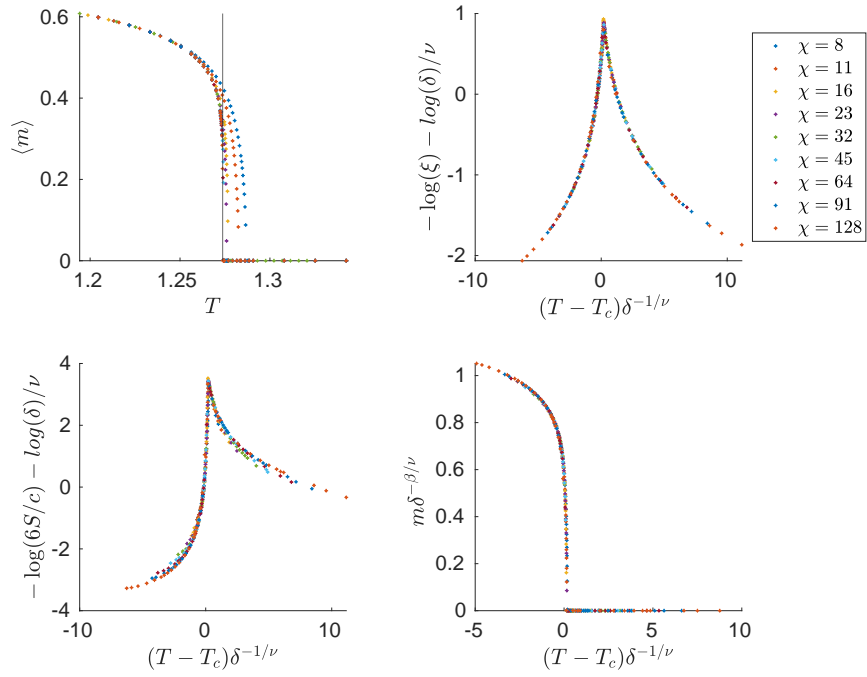
Figure 6.13: Data collapse for $g = 2.5$ phase transition of TFI Model. Data points are taken from $T \in [T_c - 0.08, T_c + 0.08]$.

dimensions is much higher. This is not necessarily a problem, as this gives better data for the finite-size scaling. There is no known analytical expression for the critical temperature. With quantum Monte Carlo techniques, a value of $T_c = 1.2737(6)$ is obtained, while state-of-the-art TN techniques provide a value of $T_c = 1.2737(2)$ [41]. With the series expansion from this paper and VUMPS, $T_c = 1.2736(6)$, indicating that this faithfully captures the physics. To put his into context: the directly calculated error ϵ^2 at $T = 1.2$ was around 0.006. This result was achieved for a PEPO with bond dimension 21 (order 5, no loops).

6.4.3 T=0.7 quantum phase transition
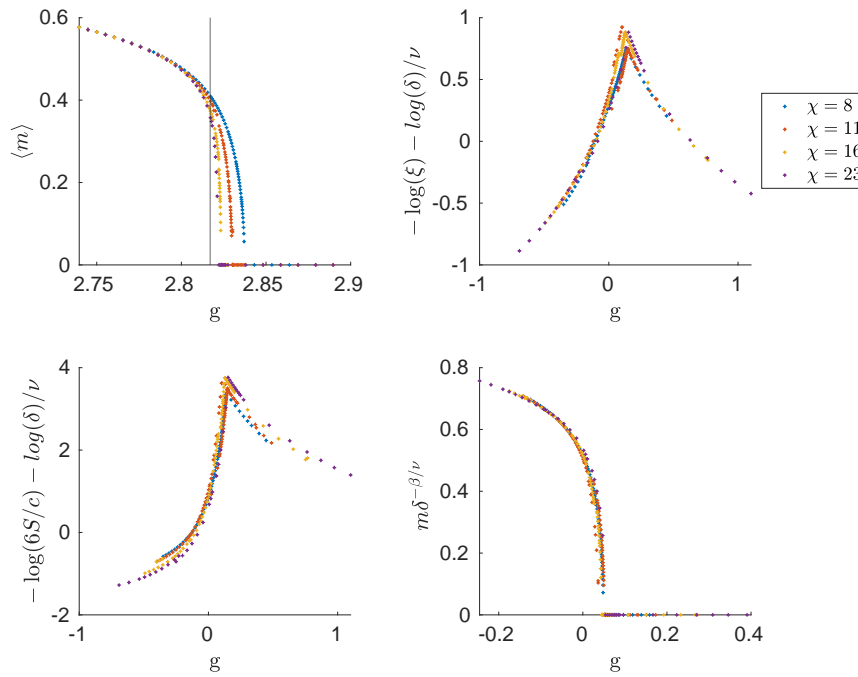
Figure 6.14: Results for $T = 0.7$ phase transition. The green points are a truncated order 6 construction, the others order 5.

Up until now, the phase diagram in fig. 6.9 was explored at constant g in function of T, but of course there is also a transition at constant T and varying g. T = 0.7 is chosen. The results are shown in fig. 6.14. Clearly, the results do not collapse as well as for the other models. The reason is that the order of the expansion is not high enough. A more accurate version is shown in fig. 6.15. This expansion is of order 6, where virtual level 3 is truncated to dimension 20. The others are order 5. Both include loop contributions, as opposed to
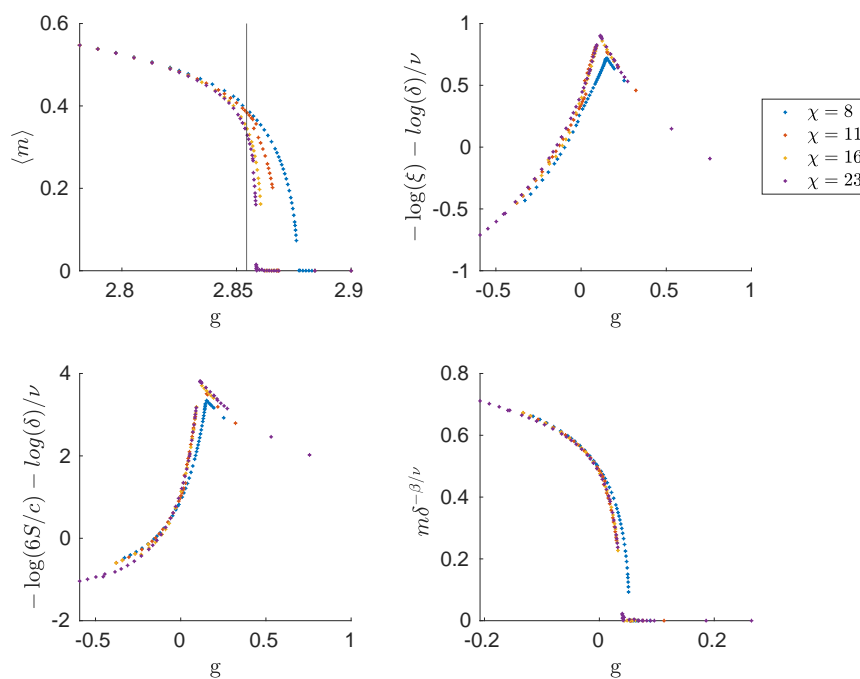
Figure 6.15: Results for $T = 0.7$ phase transition. The cluster expansion is of order 6.

the previous results. The fitted critical temperature moves from $T_c = 2.817$ to $T_c = 2.854$. The total bond dimension is $(D = 1 + 4 + 16 + 20 + 6)$ instead of $(D = 1 + 4 + 16 + 6)$. The results, are both fitted without $\chi = 8$ due to obvious finite-size effects.

6.4.4 Tricritical point

Now that we know the critical transversal field can be determined, all the tools are present to extrapolate the tricritical point, indicated by a red dot in fig. 6.9. To achieve this, the following scaling relation can be used

$$T_c = |g_c - g_{c,q}|^{z\nu_{3D}} . \tag{6.6}$$

Near the critical point, the critical temperature T_c and the critical transversal field g_c are related to each other by the value of the quantum critical point $g_{c,q}$ and critical exponent $z = 1$ and $\nu_{3D} \approx 0.62998$ [22]. This could perfectly be done, given enough computation time to calculate for many temperatures the critical transversal field with a finite-size scaling, similar to section 6.4.3. Possibly, also a scaling in truncation dimension D for the cluster expansion is needed.

6.4.5 Going beyond

With all the built machinery to construct cluster expansions, it is logical to calculate the phase diagram with increased precision.

6.4.5.1 Higher order

Going to higher order (i.e. longer linear chains) and adding the single loop contribution works well. The bond dimension of the largest virtual level (3 in this case) can be truncated in the construction. The linear and nonlinear solver find the least squares solution to the problems. Care has to be taken in order to not violate one of the conclusions in 1D: never construct a longer chain than the previously fully solved one. For instance if the bond is truncated to 10, the linear chains can be constructed up till order 4. This means eq. (4.31) can be added but not eq. (4.32), because the longest chains are order 5. Level 5 can still be constructed and contracted easily, but virtual level 3 has a bond dimension of 64. Compared to the previous bond dimensions (1,4 and 16) this is quite large and needs to be truncated.

6.4.5.2 Loops and extensions

Adding loops decreases the direct error. But there is also a surprising result: all possible loop extensions result in a higher error when the environment is calculated in the thermodynamic limit. The fluctuation of the magnetisation increases drastically. With increasing bond dimension χ, this is somewhat better but still not good enough. The question is whether this is a result of a failing

cluster expansion, or the inability of VUMPS to calculate the correct environment. During my thesis, my focus has largely been on the first case. After all, the framework was completely built from scratch and errors happen, and there is no guarantee that the series even converges. But introduction of very strict variants, such as the generalization to type E, where left/upper extension have level a and the right/lower extensions are of type a'

$$\qquad\qquad (6.7)$$

also introduce these fluctuations in the calculated magnetisation. The VUMPS procedure comes with 2 implicit assumptions. Firstly, in section 2.5.4.4 the procedure for finding B from below was explained. In the case tested here, the PEPO is rotationally invariant and real. Solving the same equations but with complex tensors, results in wrong magnetisation. For models like Heisenberg, complex tensors are needed to solve the rotationally invariant equations. Switching to complex tensors has no effect on the direct error measures. This suggests the used method for finding B was incorrect. Implementing the suggested solution in section 2.5.4.4 would come at no extra cost: the tensor is exactly rotation invariant and hence the fixed point form below is automatically known in this case. A second assumption made during this thesis is that the wave function can be represented by a 1 by 1 unit cell. Despite efforts made, the multisite version [13] doesn't seem to produce sensible results, even for PEPO's where the 1 by 1 unit cell does give the right results. It's clear that more research is needed here. One way to locate the problem would be to use another algorithm to contract the network, such as corner transfer matrix renormalization group (CTMRG) as used in [41], or even using the single site VUMPS algorithm combined with blocking:

$$\qquad\qquad (6.8)$$

6.4.5.3   Better extrapolation

The spirit behind finite-size scaling is to calculate more with the data available. In section 3.2.5, 2 additional variations were suggested. One is to account for the subleading finite-size corrections, the other to change the way of calculating δ.

6.4.5.3.1 Subleading corrections Introducing subleading corrections requires 4 parameters for each fitted universal function: ω, ϕ, c and d from eq. (3.4). Compared to the one parameter T_c or g_c, this is a lot and can be used to make many graphs collapse into a single graph. The analysis in [42] carefully estimates the error made with the fitting procedure. A similar amount of rigour would be required to fully trust the results. Nevertheless, fitting with subleading corrections results in critical temperatures close to the ones fitted without, and the subleading exponents are close to 1, as expected from the subleading series expansion.

6.4.5.3.2 Choice δ Different choices of c_i in eq. (3.3) are possible to construct δ. Variational optimisation was done as suggested in [13]. Both the x- and y-axis should be normalised, because depending on δ the scale changes. If the scale is based on the outer points, there is a flaw in the fitting procedure. The variational minimum goes to a point where at least for one point δ is extremely close to 0, and hence everything on the x-axis except that point is pinched together, resulting in a very small relative error. Therefore, it is better to normalise according to the points at e.g. the 25th percentile and 75th percentile of the range on the axis. This improves the collapse, but is not clear how this carries over to the predicted T_c or g_c.

6.4.6 Conclusion

The conclusion from the direct results in section 6.3 seems to carry over to a lattice in the thermodynamic limit very well. Comparison with the critical temperatures from literature confirmed that this method is able to approximate the operator $\exp\left(-\beta\hat{H}\right)$ accurately, even for a cluster expansion of order 5 without loops. This has only a bond dimension of 21, enabling the use of larger VUMPS bond dimensions χ. The path for determining a value for the tricritical, an important test to compare this method to the literature, is clear and mostly requires more precise data. With this data, and careful analysis of the end results, the techniques of section 6.4.5.3 could be used to improve the results further. The results also indicate that the current procedure to determine the magnetisation with VUMPS, and in particular the question of how to close the VUMPS environment from below as introduced in section 2.5.4.4, deserves some closer attention in the future.

6.5 Conclusion

This chapter tested the proposed cluster expansion in a number of different ways. In section 6.2, the accuracy of the constructions introduced in section 4.2 were measured against the exact exponentiation of the Hamiltonian on a cyclic chain. It was shown that taking the pseudoinverse as explained in section 5.2.1 is absolutely necessary to obtain a converging cluster expansion. A surprising result is that the strict types, which only add the explicitly calculated blocks to

the expansion but not longer chains, result in a less performant cluster expansion. Independent of the Hamiltonian, type A results in the smallest errors and the smallest bond dimension. The largest virtual level can be truncated to any desired bond dimension. Constructing an explicit rotationally invariant MPO does not result in a lower error. Overall, the 1D results show that an order 7 construction (bond dimension 86) can represent $e^{\beta \hat{H}}$ up to machine precision for larger imaginary time steps. For the transverse Ising model, $\beta_{exact} = 0.6$ while for Heisenberg it amounts to $\beta_{exact} = 0.05$. The construction from 1D was generalised to 2D in section 4.3. section 6.3 presented the 2D results in a manner very similar to the 1D results. It was found that the 1D construction generalises well to 2D. Beside the equivalent blocks from 1D, also the loop contribution are important to keep the error low. Higher extensions improve upon the result, but they also require a large bond dimension. The results are somewhat less reliable than the 1D counterpart, because even with reduced density matrices, it is hard to contract a TN large enough to capture all the details of the model. The fact that linear blocks and only one single loop are able to capture the essence of the exponential opens up the possibility to generalise the method to 3D setting. With the results from these sections, it is clear that this method holds some real potential. In section 6.4, some phase transitions are calculated with the constructed tensor exponentials in combination with VUMPS. The $g = 0$ and $g = 2.5$ critical temperatures T_c match very well with the values found in literature. Calculation of the $T = 0.7$ phase transition shows that also closer to the quantum critical point, (quantum) critical behaviour is found. The path is open to calculate the tricritical point of the 2D TFI model with the current methods.

Chapter 7

Conclusion and lookout

7.1 Conclusion

In chapter 1 the quantum many-body problem was introduced as one of the challenges faced by modern physics. The problem is not related to the theory, but to the practical difficulty of calculating and simulating the properties of strongly correlated matter. A macroscopic overview of one class of techniques, namely TNs, was given. In chapter 2, a brief introduction to TNs was given. The focus was mainly to provide some insight in the algorithms using the graphical TN notation. In particular, the VUMPS algorithm was explained. Also, a technique was suggested to close the environment from below, because the current procedure only seems to work only in specific cases. Chapter 3 explains some concepts related to phases and phase transitions. The critical exponents associated with continuous phase transitions are discussed, together with a practical way, called finite-size scaling, to obtain them from simulations. This chapter also gives an overview of 2 important models in the field: the Heisenberg model and the TFI model in different dimensions. Finally, the need for operator exponentials and some TN methods to simulate time evolution are discussed. Chapter 4 is the center point of this dissertation: it explains how the cluster expansions are made. This is written down in a very compact way, making abstraction of all implementation details and results. Chapter 5 details the working of the 3 different solvers: a linear solver, a nonlinear solver and a sequential linear solver. The linear solver must be implemented with great care in order to handle the ill-conditioned inverses. This is done by calculating the pseudoinverse instead of a full inverse. This can be done at the cost of an SVD decomposition per leg, while maintaining the accuracy of inverting all the legs at once. Also, an algorithm for determining all possible virtual indices of a given map using PEPS contraction is given. A starting point for exploring the source code is given. Chapter 6 finally quantifies the quality of the cluster expansions. This is done with respect to the exact solution in 1D and 2D. As a second test, the thermal phase diagram of the TFI model is calculated. The

numerical temperatures of the phase transition at a constant transversal field correspond well to the values in literature. A clear path is set out to calculate the tricritical point.

7.2   Outlook

The cluster expansions have a lot of potential. In the course of one Master's dissertation, it is not possible to look into every possibility. Here are some interesting prospects:

Real time evolution

Some first results in 1D (section 6.2.4) show that the method also works for real time evolution. Section 3.4.3 introduced some methods to calculate time evolutions with TNs. Crucially, all these methods come with a detailed error analysis. For instance, some methods have a proven error per step of $O\left((t/N)^2\right)$, resulting in an error of $O\left(t^2/N\right)$ at time t. This can be made arbitrarily small. A similar analysis of this method is not yet performed.

Quantum critical point

As stated in section 6.4.4, it should be possible to calculate the quantum critical point of the TFI model using the current methods. This would allow to make a better comparison of this method better with other methods in literature.

Lattices

During this thesis, a square lattice was used. This is definitely not the only choice. Other lattices could potentially result in even lower errors. Due to the generality of the solvers, changing the lattice should be within reach given more time.

Higher dimensions

A promising conclusion from section 6.4 is that a combination of linear blocks and simple loops seems sufficient to construct a good TN. This opens up the path to create a 3D version of the operator $e^{\beta \hat{H}}$.

Symmetries

The limitations of simulation could be pushed beyond what is possible in the current framework by incorporating all symmetries available (both spatial and internal). This already exists for diagonalisation (e.g. see [40]). This could be embedded in the framework.

Bibliography

[1] R. Sinatra, P. Deville, M. Szell, D. Wang, A. L. Barabási, A century of physics (oct 2015). `arXiv:1612.00079, doi:10.1038/nphys3494`.

[2] S. Hansson, Johan (Division of Physics, Luleå University of Technology, SE-971 87 Luleå, The 10 Biggest Unsolved Problems in Physics, International Journal of Modern Physics and Applications 1 (1) (2015) 12–16.
URL `http://www.publicscienceframework.org/journal/ijmpa`

[3] Z. E. Musielak, B. Quarles, The three-body problem (aug 2014). `arXiv:1508.02312, doi:10.1088/0034-4885/77/6/065901`.
URL `http://arxiv.org/abs/1508.02312http://dx.doi.org/10.1088/0034-4885/77/6/065901`

[4] M. T. Mora-Aznar, Condensed-Matter and Materials Physics: Basic Research for Tomorrow's Technology, European Journal of Physics 21 (2) (2000) 197–202. `doi:10.1088/0143-0807/21/2/707`.

[5] E. G. Lewars, Computational chemistry: Introduction to the theory and applications of molecular and quantum mechanics, Springer Netherlands, 2011. `doi:10.1007/978-90-481-3862-3`.

[6] A. Alexandradinata, N. P. Armitage, A. Baydin, W. Bi, Y. Cao, H. J. Changlani, E. Chertkov, E. H. d. S. Neto, L. Delacretaz, I. E. Baggari, G. M. Ferguson, W. J. Gannon, S. A. A. Ghorashi, B. H. Goodge, O. Goulko, G. Grissonnanche, A. Hallas, I. M. Hayes, Y. He, E. W. Huang, A. Kogar, D. Kumah, J. Y. Lee, A. Legros, F. Mahmood, Y. Maximenko, N. Pellatz, H. Polshyn, T. Sarkar, A. Scheie, K. L. Seyler, Z. Shi, B. Skinner, L. Steinke, K. Thirunavukkuarasu, T. V. Trevisan, M. Vogl, P. A. Volkov, Y. Wang, Y. Wang, D. Wei, K. Wei, S. Yang, X. Zhang, Y.-H. Zhang, L. Zhao, A. Zong, The Future of the Correlated Electron Problem (oct 2020). `arXiv:2010.00584`.
URL `http://arxiv.org/abs/2010.00584`

[7] P. Corboz, Lecture 1: tensor network states, Tech. rep.

[8] I. Cirac, D. Perez-Garcia, N. Schuch, F. Verstraete, Matrix Product States and Projected Entangled Pair States: Concepts, Symmetries, and Theo-

rems (2020). `arXiv:2011.12127`.
URL `http://arxiv.org/abs/2011.12127`

[9] R. Orús, A practical introduction to tensor networks: Matrix product states and projected entangled pair states (oct 2014). `arXiv:1306.2164, doi: 10.1016/j.aop.2014.06.013`.

[10] V. Zauner-Stauber, L. Vanderstraeten, M. T. Fishman, F. Verstraete, J. Haegeman, Variational optimization algorithms for uniform matrix product states, Physical Review B 97 (2018) 45145. `doi:10.1103/ PhysRevB.97.045145`.

[11] S.-J. Ran, Tensor Network Contractions :Methods and Applications to Quantum Many-Body Systems, 2020.
URL `http://www.springer.com/series/5304`

[12] L. Vanderstraeten, J. Haegeman, F. Verstraete, Tangent-space methods for uniform matrix product states, SciPost Phys. Lect. Notes 7 (2019). `doi:10.21468/SciPostPhysLectNotes.7`.

[13] A. Nietner, B. Vanhecke, F. Verstraete, J. Eisert, L. Vanderstraeten, Efficient variational contraction of two-dimensional tensor networks with a non-trivial unit cell, Quantum 4 (2020). `arXiv:2003.01142, doi:10.22331/ Q-2020-09-21-328`.

[14] M. Hauru, C. Delcamp, S. Mizera, Renormalization of tensor networks using graph independent local truncations, Tech. rep. `arXiv:1709.07460v2`.

[15] R. Orús, G. Vidal, Simulation of two-dimensional quantum systems on an infinite lattice revisited: Corner transfer matrix for tensor contraction`doi: 10.1103/PhysRevB.80.094403`.

[16] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Annals of Physics 326 (1) (2011) 96–192. `arXiv:1008.3477, doi:10.1016/j.aop.2010.09.012`.
URL `http://arxiv.org/abs/1008.3477http://dx.doi.org/10.1016/ j.aop.2010.09.012`

[17] H. Nishimori, G. Ortiz, Elements of Phase Transitions and Critical Phenomena, Vol. 9780199577, Oxford University Press, 2011. `doi:10.1093/ acprof:oso/9780199577224.001.0001`.

[18] G. Jaeger, The ehrenfest classification of phase transitions: Introduction and evolution, Archive for History of Exact Sciences 53 (1) (1998) 51–81. `doi:10.1007/s004070050021`.

[19] W. Bletenholz, U. Gerber, Berezinskii-kosterlitz-thouless transition and the Haldane conjecture: Highlights of the Physics Nobel Prize 2016, Revista Cubana de Fisica 33 (2) (2016) 156–168. `arXiv:1612.06132`.

[20] A. J. Beekman, L. Rademaker, J. van Wezel, An Introduction to Spontaneous Symmetry Breaking (sep 2019). `arXiv:1909.01820, doi:10.21468/scipostphyslectnotes.11`.

[21] G. Ódor, Universality classes in nonequilibrium lattice systems (jul 2004). `arXiv:0205644, doi:10.1103/RevModPhys.76.663`.

[22] S. Hesselmann, S. Wessel, Thermal Ising transitions in the vicinity of two-dimensional quantum critical points, PHYSICAL REVIEW B 93 (2016) 155157. `doi:10.1103/PhysRevB.93.155157`.

[23] L. P. Kadanoff, Theories of Matter: Infinities and Renormalization (2010). `arXiv:1002.2985`.
URL `http://arxiv.org/abs/1002.2985`

[24] M. E. Fisher, The theory of equilibrium critical phenomena (jul 1967). `doi:10.1088/0034-4885/30/2/306`.
URL `https://iopscience.iop.org/article/10.1088/0034-4885/30/2/306https://iopscience.iop.org/article/10.1088/0034-4885/30/2/306/meta`

[25] K. S. D. Beach, L. Wang, A. W. Sandvik, Data collapse in the critical region using finite-size scaling with subleading corrections (may 2005). `arXiv:0505194`.
URL `http://arxiv.org/abs/cond-mat/0505194`

[26] B. Vanhecke, J. Haegeman, K. Van Acoleyen, L. Vanderstraeten, F. Verstraete, Scaling Hypothesis for Matrix Product States, Physical Review Letters 123 (25) (2019). `arXiv:1907.08603, doi:10.1103/PhysRevLett.123.250604`.

[27] P. Ginsparg, E. Brézin, J. Zinn-Justin, Applied Conformal Field Theory, Tech. rep. (1988).

[28] P. Calabrese, J. Cardy, Entanglement entropy and quantum field theory, Journal of Statistical Mechanics: Theory and Experiment (6) (2004). `arXiv:0405152, doi:10.1088/1742-5468/2004/06/P06002`.

[29] S. Sachdev, Quantum phase transitions, Physics World 12 (4) (1999) 33–38. `doi:10.1088/2058-7058/12/4/23`.
URL `https://iopscience.iop.org/article/10.1088/2058-7058/12/4/23https://iopscience.iop.org/article/10.1088/2058-7058/12/4/23/meta`

[30] A. Taroni, Statistical physics: 90 years of the Ising model (dec 2015). `doi:10.1038/nphys3595`.
URL `www.nature.com/naturephysics`

[31] D. Radicevic, Spin Structures and Exact Dualities in Low Dimensions (2018). `arXiv:1809.07757`.
URL `http://arxiv.org/abs/1809.07757`

[32] T. H. Hsieh, From d-dimensional Quantum to d + 1-dimensional Classical Systems, Student review (2) (2015) 1–4.

[33] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, C. Hubig, Time-evolution methods for matrix-product states, Annals of Physics 411 (2019). `arXiv:1901.05824, doi:10.1016/j.aop.2019.167998`.

[34] M. P. Zaletel, R. S. K. Mong, C. Karrasch, J. E. Moore, F. Pollmann, Time-evolving a matrix product state with long-ranged interactions, PHYSICAL REVIEW B 91 (2015) 165112. `doi:10.1103/PhysRevB.91.165112`.

[35] B. Vanhecke, L. Vanderstraeten, F. Verstraete, Symmetric cluster expansions with tensor networks, Physical Review A 103 (2) (2021). `arXiv:1912.10512, doi:10.1103/PhysRevA.103.L020402`.

[36] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor Ring Decomposition (2016). `arXiv:1606.05535`.
URL `http://arxiv.org/abs/1606.05535`

[37] P. Moylan, Qr factorisation and pseudoinverse of rank- deficient matrices, ftp://pmoylan.org/papers/90.QR

[38] J. Matt, N-dimensional sparse arrays.
URL   `https://www.mathworks.com/matlabcentral/fileexchange/29832-n-dimensional-sparse-arrays`

[39] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later (2003). `doi:10.1137/S00361445024180`.

[40] A. Wietek, A. M. Läuchli, Sublattice coding algorithm and distributed memory parallelization for large-scale exact diagonalizations of quantum many-body systems, Physical Review E 98 (3) (2018). `arXiv:1804.05028, doi:10.1103/PhysRevE.98.033309`.

[41] P. Czarnik, P. Corboz, Finite correlation length scaling with infinite projected entangled pair states at finite temperature, Physical Review B 99 (2019) 245107. `doi:10.1103/PhysRevB.99.245107`.

[42] L. Wang, K. S. Beach, A. W. Sandvik, High-precision finite-size scaling analysis of the quantum-critical point of S=1 2 Heisenberg antiferromagnetic bilayers, Physical Review B - Condensed Matter and Materials Physics 73 (1) (2006). `arXiv:0509747, doi:10.1103/PhysRevB.73.014431`.