

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

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

Academic year 2020-2021

Todo list

date	vii
vind bron en voorbeelden	2
area law+picture	2
sign problem monte carlo,	2
write about tensor networks	2
MPO examples	6
bron MPS	10
explain this	15
bond dim op figuur	16
citation	20
reference	22
reference	23
line crossing	23
q potts,...	26
blabla	26
is this true??	26
citation	27
value	27
Add more	46
figure	53
reformulate	55
New figure with entropy formula fixed $6S/c$ instead of $cS/6$	72

Foreword

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.²⁰

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.

May 27, 2021

date

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

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

Academic year 2020-2021

Abstract

Simulating many-body quantum systems is one of the challenges faced by modern Physics. One particular problem is computing the operator $e^{-\beta\hat{H}}$, which is used to compute the partition function of a quantum model. This function is needed to make prediction about the macroscopic behaviour of quantum systems. Of particular interest are the phase transitions. The 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 of the phase transition of transversal field Ising model in 2D produces results that can challenge recent developments in literature.

Keywords

Tensor networks, Thermal States, Cluster Expansions, Phase transition, Transversal Ising model

¹Student 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 computing the operator $e^{-\beta\hat{H}}$, which is used to compute the partition function of a quantum model. This function is needed to make prediction about the macroscopic behaviour of quantum systems. Of particular interest are the phase transitions. The 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 of the phase transition of transversal field Ising model in 2D produces results that can challenge recent developments in literature.

1 Introduction

2 Tensor Networks


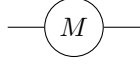
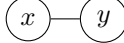
Matrix Product States (MPS) form an ansatz to describe the

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

The tensor C needs d^n numbers to describe the full wave function. In uniform MPS, the tensor $C^{i_1 i_2 \dots i_n}$ is subdivided into the product of n tensors C and a matrix M that contains the boundary conditions

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

Table 1: Caption

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

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

$$\begin{array}{c} i_1 \quad i_2 \quad \quad \quad i_n \\ | \quad | \quad \quad \quad | \\ \textcircled{C} - \textcircled{C} - \dots - \textcircled{C} - \textcircled{M} \\ \text{---} \end{array} \quad (3)$$

An MPS has 2 dimension, the physical dimension of the particles $d = \dim(|i_2\rangle)$ and the dimension χ of the bonds between the tensors. The cluster expansion will rely 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

2.1 MPO

A matrix product operator (MPO), is similar to an MPS but has 2 physical legs i and j . The following

compact notation is used in this paper

$$\begin{array}{c} i \\ | \\ 0 \text{---} \bigcirc \text{---} 0 \\ | \\ j \end{array} = \bigcirc. \quad (4)$$

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, and summation over virtual level 1 on 2 neighbouring sites is implied. Summation over shared virtual bond 1 on 2 neighbouring sites is denoted as

$$\bigcirc \text{---}^1 \text{---} \bigcirc. \quad (5)$$

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

$$\bigcirc \text{---} \bigcirc \text{---} \bigcirc. \quad (6)$$

3 Strongly correlated matter

3.1 Transversal Ising

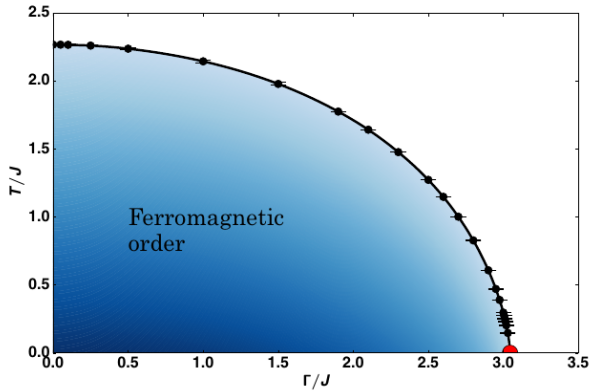


Figure 1: Phase diagram for 2D transversal Ising model. Figure taken from [1].

3.2 Criticality

3.3 operator exponentials

3.4 TN contraction

4 Cluster Expansion

The novel method to construct a PEPO $e^{-\beta\hat{H}}$ with cluster expansions. An example is given by eq. (6). This was first introduced in [2]. The goal is capture the exponential of the Hamiltonian operator \hat{H}

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

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

$$\begin{aligned} H(\bigcirc \text{---} \bigcirc \text{---} \bigcirc) &= H_1 \otimes 1 \otimes 1 \\ &+ 1 \otimes H_1 \otimes 1 \\ &+ 1 \otimes 1 \otimes H_1 \\ &+ H_2 \otimes H_2 \otimes 1 \\ &+ 1 \otimes H_2 \otimes H_2 \\ &\dots \end{aligned} \quad (8)$$

4.1 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

$$\bigcirc = \exp(-\beta H(\bigcirc)). \quad (9)$$

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.

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

Contraction of larger network lead to many terms, such as

$$\bigcirc \overset{1}{-} \bigcirc \overset{0}{-} \bigcirc \overset{0}{-} \bigcirc \overset{0}{-} \bigcirc \overset{1}{-} \bigcirc \overset{0}{-} \bigcirc \overset{1}{-} \bigcirc \overset{0}{-} \bigcirc \overset{0}{-} \bigcirc \quad (11)$$

The beauty of this lays in the fact that disconnected regions (regions separated by level 0) combine in exactly the right way to capture the terms appearing in the series expansion of the exact tensor exponential. [2] 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 different possible constructions, already differ in the next step, one more step is added to make the construction and notation clear.

$$\begin{aligned} \bigcirc \overset{1}{-} \bigcirc \overset{1}{-} \bigcirc &= \exp -\beta H(\bigcirc \overset{1}{-} \bigcirc \overset{1}{-} \bigcirc) \\ &\quad - \bigcirc \overset{0}{-} \bigcirc \overset{0}{-} \bigcirc \\ &\quad - \bigcirc \overset{1}{-} \bigcirc \overset{0}{-} \bigcirc \\ &\quad - \bigcirc \overset{0}{-} \bigcirc \overset{1}{-} \bigcirc \\ &= \exp -\beta H(\bigcirc \overset{1}{-} \bigcirc \overset{1}{-} \bigcirc) \\ &\quad - \bigcirc \overset{0}{-} \bigcirc \overset{0}{-} \bigcirc \end{aligned} \quad (12)$$

This is called an 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

$$\boxed{\bigcirc \overset{1}{-} \bigcirc \overset{1}{-} \bigcirc}, \quad (13)$$

the right-hand side of eq. (12) is implied. In the following section, different types will be discussed. For every chain length, a new block is defined. This could be done in numerous ways. The different types will be discussed in the next sections.

4.2 1D

4.2.1 Type A

$$\bigcirc \quad (14a)$$

$$\bigcirc \overset{1}{-} \bigcirc \quad (14b)$$

$$\bigcirc \overset{1}{-} \bigcirc \overset{1}{-} \bigcirc \quad (14c)$$

$$\bigcirc \overset{1}{-} \bigcirc \overset{2}{-} \bigcirc \overset{3}{-} \bigcirc \quad (14d)$$

$$\bigcirc \overset{1}{-} \bigcirc \overset{2}{-} \bigcirc \overset{3}{-} \bigcirc \overset{4}{-} \bigcirc. \quad (14e)$$

4.2.2 Type E

$$\bigcirc \quad (15a)$$

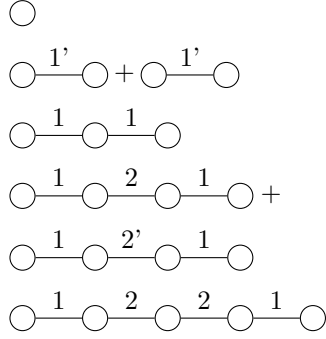
$$\bigcirc \overset{1}{-} \bigcirc \quad (15b)$$

$$\bigcirc \overset{1}{-} \bigcirc \overset{1'}{-} \bigcirc \quad (15c)$$

$$\bigcirc \overset{1}{-} \bigcirc \overset{2}{-} \bigcirc \overset{1'}{-} \bigcirc \quad (15d)$$

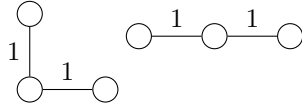
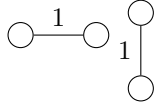
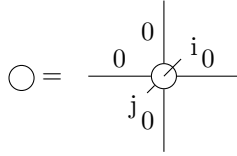
$$\bigcirc \overset{1}{-} \bigcirc \overset{2}{-} \bigcirc \overset{2'}{-} \bigcirc \overset{1'}{-} \bigcirc \quad (15e)$$

4.2.3 Type F

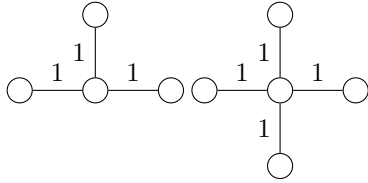


4.3 2D

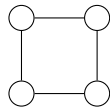
4.3.1 Linear



and rotations



4.3.2 Nonlinear



4.4 Solvers

An example of this procedure is

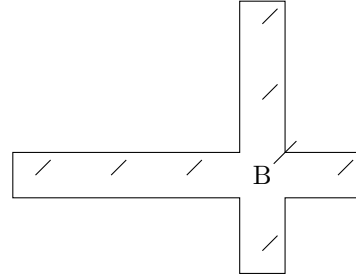
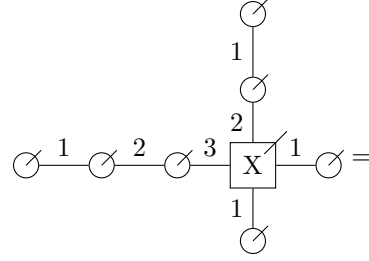
(16a)

(16b)

(16c)

(16d)

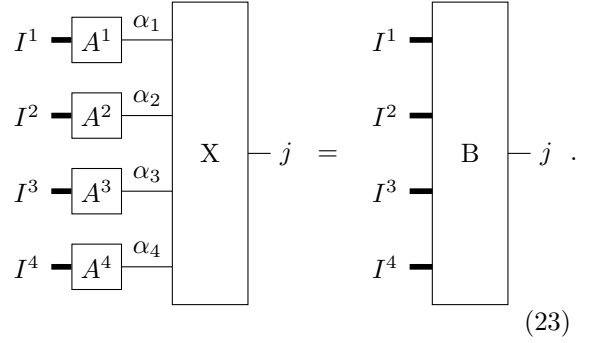
(16e)



(17)

(18)

(19)



(22)

(23)

4.4.1 Linear Solver

(20) Full inverse Sequential inverse Sparse full inverse

4.4.2 Nonlinear Solver

4.4.3 Sequential Solver

5 Results

5.1 Direct

(21)

5.2 2D Transversal Ising

- g=0 value

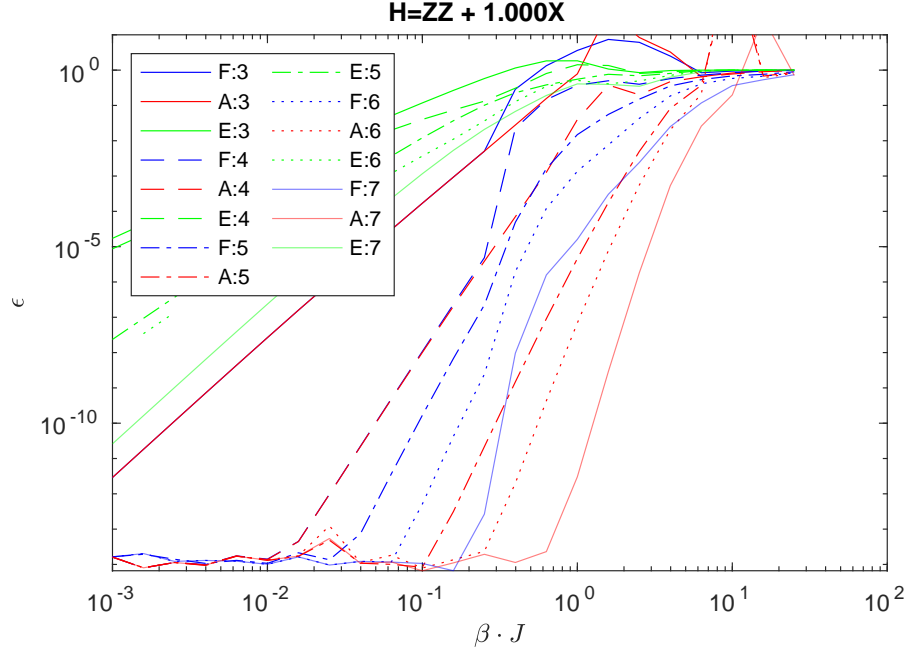


Figure 2: Comparison type A, E and F for Transversal Ising. 1D

- $g=2.5$ value
- tricrit

6 Outlook

References

- [1] S Hesselmann and S Wessel. Thermal Ising transitions in the vicinity of two-dimensional quantum critical points. *PHYSICAL REVIEW B*, 93:155157, 2016.
- [2] Bram Vanhecke, Laurens Vanderstraeten, and Frank Verstraete. Symmetric cluster expansions with tensor networks. *Physical Review A*, 103(2), 2021.

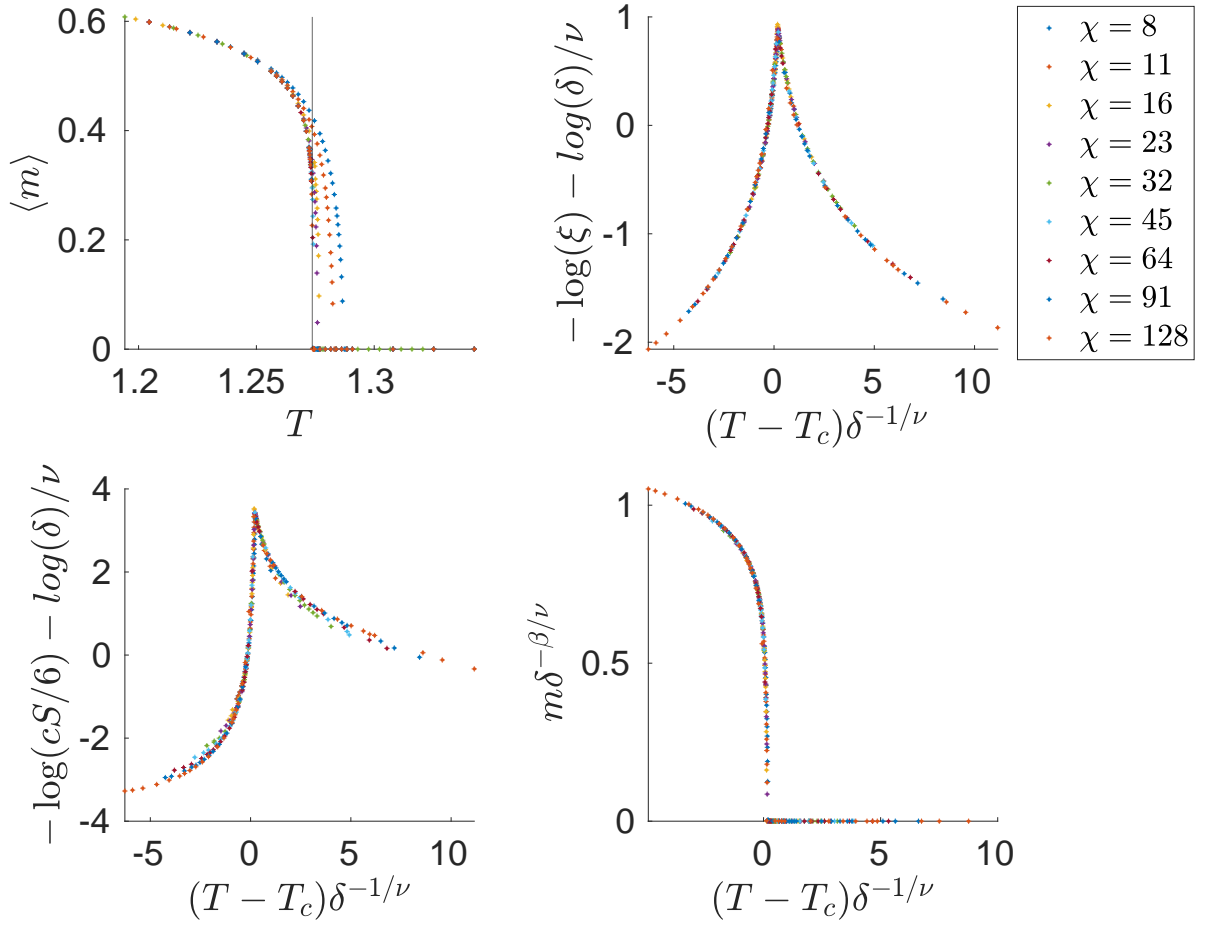


Figure 3: Data collapse for $g = 2.5$ phase transition of transversal Ising Model. Data points are taken from $T \in [T_c - 0.08, T_c + 0.08]$.

Contents

Foreword	vi
Abstract	viii
Extended abstract	ix
List of Acronyms	xx
1 Introduction	1
1.1 Introduction	1
1.2 Tensor networks	2
2 Tensor networks	3
2.1 Introduction	3
2.2 Tensor networks	3
2.2.1 Graphical notation	3
2.2.2 Representing a quantum state	3
2.2.3 Classification	5
2.2.3.1 MPS	5
2.2.3.2 MPO	5
2.2.3.3 PEPS	6
2.2.3.4 PEPO	6
2.2.3.5 Others	7
2.3 Tensor network manipulations	7
2.3.1 Basics	7
2.3.1.1 Grouping legs	7
2.3.1.2 Decomposition	8
2.3.1.3 Truncation	8
2.3.1.4 Virtual levels	9
2.3.1.5 Inversion	9
2.3.1.6 Contraction order	9
2.4 MPS algorithms	9
2.4.1 Canonical form	10
2.4.1.1 Entropy	11
2.4.1.2 Expectation value	11

2.5	Tensor network contraction	11
2.5.1	1D problem	12
2.5.2	2D problem	13
2.5.3	Overview methods	13
2.5.3.1	Real-space renormalisation-group methods . . .	13
2.5.3.2	Corner transfer matrix methods	13
2.5.3.3	Boundary methods	14
2.5.4	VUMPS	16
2.5.4.1	The equations	16
2.5.4.2	The derivation	19
2.5.4.3	Multisite	19
2.5.4.4	Calculating the MPS from below	19
2.6	Conclusion	20
3	Strongly correlated matter	21
3.1	Introduction	21
3.2	Phases and Criticality	21
3.2.1	Phases of matter	21
3.2.2	Symmetry breaking	22
3.2.3	Universality	22
3.2.4	Critical exponents for spin systems	22
3.2.5	Finite-size scaling	23
3.2.5.1	Finite-size scaling for MPS	23
3.2.5.2	Subleading corrections	24
3.2.6	CFT	24
3.2.7	Quantum phase transitions	24
3.3	Models	24
3.3.1	Ising model	25
3.3.1.1	Classical Ising	26
3.3.1.2	Quantum Ising	26
3.3.2	Heisenberg	28
3.3.3	Random	28
3.3.4	Quantum to classical mapping	28
3.4	Operator exponentials	29
3.4.1	Statistical mechanics	29
3.4.2	Applications	30
3.4.2.1	Temporal correlation functions	30
3.4.2.2	Ground state	30
3.4.3	Tensor network methods	30
3.4.3.1	Approximations to $\hat{U}(\delta)$	30
3.4.3.2	Global Krylov method	31
3.4.3.3	MPS-local methods	31
3.5	Conclusion	31

4	Construction Cluster expansion	33
4.1	Introduction	33
4.1.1	Notation	33
4.1.2	Idea	34
4.2	Construction MPO	36
4.2.1	Type A	36
4.2.1.1	Dimension	37
4.2.1.2	Discussion	37
4.2.2	Type B	37
4.2.2.1	Dimension	38
4.2.2.2	Discussion	38
4.2.3	Type C	38
4.2.3.1	Discussion	39
4.2.4	Type D	39
4.2.4.1	Discussion	40
4.2.4.2	Matrisation	40
4.2.5	Type E	40
4.2.6	Type F	41
4.2.7	Conclusion	41
4.3	Construction PEPO	41
4.3.1	Linear blocks	42
4.3.2	Loops	44
4.3.2.1	Single extensions	44
4.3.2.2	Double extensions	45
4.3.2.3	Larger loops	45
4.3.2.4	Other ideas	46
4.3.2.5	Bond dimension	46
4.4	Symmetry	46
4.5	Conclusion	47
5	Framework implementation	49
5.1	Introduction	49
5.2	Solvers	49
5.2.1	Linear solver	50
5.2.1.1	Full inverse	51
5.2.1.2	Sequential inverse	52
5.2.1.3	Sparse full inverse	52
5.2.2	Extension	53
5.2.3	Nonlinear solver	53
5.2.3.1	Automatic differentiation	54
5.2.3.2	Symmetry	54
5.2.3.3	Combining problems	54
5.2.4	Sequential linear solver	54
5.2.4.1	Conclusion	54
5.3	Optimisation	55
5.3.1	Bookkeeping	55

5.3.2	Fast contraction	55
5.3.3	Normalisation	56
5.3.4	Internal representation	56
5.3.5	Even faster inverses	56
5.3.6	Buffering Results	57
5.3.7	Profiling	57
5.3.8	Calculating the error	57
5.4	Calculating phase diagrams	57
5.4.1	Points sampling	58
5.4.2	Storing the information	58
5.4.3	Fitting	58
5.5	How to use	58
5.5.1	Source code structure 1D	59
5.5.2	Source code structure 2D	59
5.5.3	VUMPS code	59
5.6	Limitations and outlook	59
5.6.1	Implementation	59
5.6.2	Code quality	60
5.6.3	Size limitation	60
5.6.4	Lattices	60
5.6.5	Symmetries	60
6	Results	61
6.1	Introduction	61
6.2	Results 1D	61
6.2.1	Exact tensor matrix exponential	61
6.2.1.1	Norms	62
6.2.2	Inversion procedure	62
6.2.2.1	Full pseudoinversion	62
6.2.3	Models	64
6.2.3.1	Ising	64
6.2.3.2	Heisenberg	64
6.2.3.3	Random	65
6.2.4	Real time evolution	65
6.2.5	Conclusion	66
6.3	Results 2D	66
6.3.1	Norm	68
6.3.2	Models	69
6.3.2.1	Ising	69
6.3.2.2	Heisenberg	70
6.3.3	Conclusion	70
6.4	Phase diagram 2D transversal Ising model	71
6.4.1	Classical Ising phase transition	71
6.4.2	$g=2.5$ phase transition	72
6.4.3	$T=0.7$ quantum phase transition	73
6.4.4	Tricritical point	74

6.4.5	Going beyond	75
6.4.5.1	Higher order	75
6.4.5.2	Loops and extensions	76
6.4.5.3	Better extrapolation	79
6.4.6	Conclusion	79
6.5	Conclusion	80
7	Conclusion and lookout	81
7.1	Conclusion	81
7.2	Outlook	82

List of Acronyms

MPS Matrix Product State

MPO Matrix Product Operator

PEPS Projected Entangled Pairs State

PEPO Projected Entangled Pairs Operator

VUMPS variational uniform Matrix Product State algorithm

TEBD Time Evolving Block Decimation

SVD Singular Value Decomposition

CFT Conformal Field Theory

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 understood. 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.

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 historical importance is the 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.

In reality, the real challenge is to model the macroscopic properties of quantum many-body system with around 10^{23} particles. Needless to say, this is not an easy task at all. Finding good and computable approximations is of primary importance in the fields of quantum chemistry, condensed matter physics and materials science.

In computational chemistry, the many-body problem is tackled with methods which fall in one of the following categories: (post-)Hartree-Fock methods, density functional theory (DFT) and force-field methods. While they have many

vind bron en voorbeelden

applications, these methods are not fully able to capture all the properties of the so-called strongly correlated matter.

Examples of phase of strongly correlated matter which are not yet understood include high-T superconductors, topological ordered phases, quantum spin liquids [2]. There exist different methods to investigate these exciting materials. A very limited number of models is quantum integrable, meaning they can be solved in a non perturbative way. Also, some properties of models near criticality can be determined exactly with conformal field theory (CFT). But for some 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, 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. [3]

In this thesis, a technique is proposed that builds on the broad field of tensor networks.

1.2 Tensor networks

This is often referred to as the curse of dimensionality. The size of the Hilbert space of quantum states grows exponentially fast. This prevents an efficient description of all possible quantum states.

area law+picture

sign problem monte carlo,
...

write about tensor net-
works

Chapter 2

Tensor networks

2.1 Introduction

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

2.2 Tensor networks



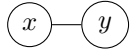
2.2.1 Graphical notation

Before explaining tensor networks, 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. Some examples are shown in table 2.1. Every leg which is connected to multiple tensors, is contracted.

2.2.2 Representing a quantum state

Tensor networks come in many shapes and forms. They are really used to represent a tensor with many legs. A general quantum state with N sites can

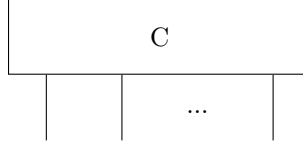
Table 2.1: Caption

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

be described in a given basis $|i\rangle$ in the following way:

$$|\Psi\rangle = \sum_{i_1 i_2 \dots i_n} C^{i_1 i_2 \dots i_n} |i_1\rangle \otimes |i_2\rangle \otimes \dots \otimes |i_n\rangle \quad (2.1)$$

Here the tensor C holds all the information of the quantum state. The graphical representation can be seen in fig. 2.1.

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

This requires an exponential number d^n 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:

$$C^{i_1 i_2 \dots i_n} = C_{\alpha_1}^{1 i_1} C_{\alpha_1 \alpha_2}^{2 i_2} \dots C_{\alpha_{n-1}}^{n i_n} \quad (2.2)$$

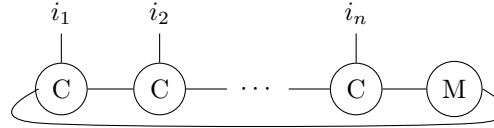
$$\begin{array}{c} i_1 \quad i_2 \quad \quad i_n \\ | \quad | \quad \quad | \\ \textcircled{C^1} - \textcircled{C^2} - \dots - \textcircled{C^n} \end{array} \quad (2.3)$$

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 over 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 $C_{\alpha\beta}^i$ 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:

$$|\Psi\rangle = \text{Tr}(C^{i_1} C^{i_2} \dots C^{i_n} M) |i_1\rangle \otimes |i_2\rangle \otimes \dots \otimes |i_n\rangle \quad (2.4)$$



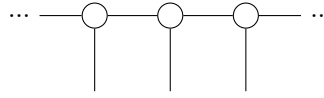
$$(2.5)$$

2.2.3 Classification

Tensor networks 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:

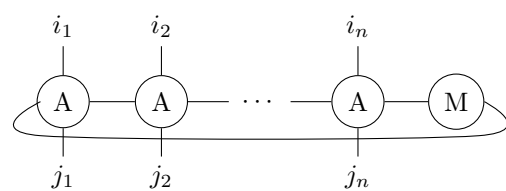


$$(2.6)$$

This will be the form we will work with in this thesis. Of course, larger unit cells are also possible. MPS 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 [2]. 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. MPS's exactly follow this area law.

2.2.3.2 MPO

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

$$\begin{aligned} \hat{O} = & \sum \text{Tr}(A^{i_1 j_1} A^{i_2 j_2} \dots A^{i_n j_n} M) \\ & \times |i_1\rangle \langle j_1| \otimes |i_2\rangle \langle j_2| \otimes \dots \otimes |i_n\rangle \langle j_n| \end{aligned} \quad (2.7)$$


The matrix M contains the boundary conditions of the operator. It's clear that this can again be made translationally invariant. Acting with an MPS on

a MPO $\hat{O}|\Psi\rangle$ again results in an MPS, as expected. todofiguur mps mpo 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.

MPO examples

2.2.3.2.1 Examples of MPO Hamiltonians

2.2.3.3 PEPS

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

(2.9)

2.2.3.4 PEPO

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

(2.10)

2.2.3.5 Others

To show there exist many more tensor networks 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. MERA (c) is also shown. These tensor networks are special because they can be contracted exactly, as opposed to for instance PEPS.

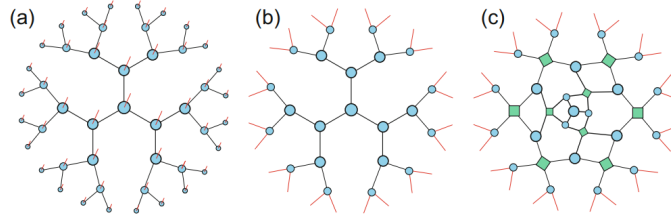


Figure 2.2: (a) and (b) two different TTNSs and (c) MERA. Figure taken from [4].

2.3 Tensor network manipulations

This section serves as an introduction of tensor network manipulations. The overview mainly focusses on MPS/MPO networks, but most of the operations translate to the 2D case. The MPS's 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:

$$\begin{aligned}
 T^{i_1 i_2 j_1 j_2} &= \begin{array}{c} i_1 \quad i_2 \\ \boxed{T} \\ j_1 \quad j_1 \end{array} \\
 &\cong (i_1 j_1) \boxed{T} (i_2 j_2) \\
 &= T^{(i_1 j_1)(i_2 j_2)}
 \end{aligned} \tag{2.11}$$

The dimension of the new leg is the product of the dimension of the individual legs. Contracting 2 merged legs with 2 merged legs is exactly the same as contracting them separately. 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 Decomposition

The grouping above can be applied to decompose a tensor into 2 tensors with matrix techniques. An example, which will be needed later on, is give here.

$$\begin{aligned}
 & \begin{array}{c} i_1 \quad i_2 \\ | \quad | \\ \text{u} - \boxed{O^{uv,vw}} - \text{w} \\ | \quad | \\ j_1 \quad j_1 \end{array} = O_{\alpha_u \gamma_w}^{i_1 i_2 j_1 j_2} \\
 & \cong O_{(\alpha_u i_1 j_1)(\gamma_w i_2 j_2)}^{uw} \\
 & = O_{(\alpha_u i_1 j_1)\alpha_v}^{uv} O_{\alpha_v(\alpha_w i_2 j_2)}^{vw} \\
 & \cong \begin{array}{c} i_1 \quad i_2 \\ | \quad | \\ \text{u} - \bigcirc - \text{v} - \bigcirc - \text{w} \\ | \quad | \\ j_1 \quad j_1 \end{array}
 \end{aligned} \tag{2.12}$$

The indices U, V and W represent block indices. Step 2 reshapes and groups the indices on to 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. For an exact representation, the bond dimension of virtual level v is the product of the dimensions of the outer legs

$$\dim v = \min(\dim u, \dim w) \cdot (\dim i)^2. \tag{2.13}$$

Many matrix decompositions exist. Some useful examples here are SVD decomposition, eigenvalue decomposition, QR, \dots .

2.3.1.3 Truncation

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

$$T = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix} \begin{bmatrix} V_1 & V_2 \end{bmatrix}^\dagger, \tag{2.14}$$

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.

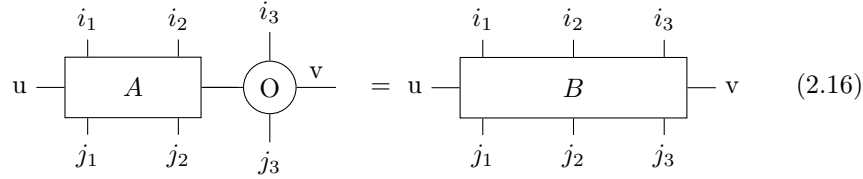
$$\min_{\text{rank}(A) \leq n} \|T - A\| = \|T - \hat{T}\| \quad (2.15)$$

2.3.1.4 Virtual levels

In the previous example, the levels were indicated with a block index or virtual level. The idea is to separate the contraction into blocks. This is completely analogous to matrix block multiplication. This will be a more natural form to represent the algorithm. Of course, one can easily switch between block representation and the full one.

2.3.1.5 Inversion

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



$$\begin{array}{c} i_1 \quad i_2 \quad i_3 \\ \text{u} - \boxed{A} - \text{v} - \bigcirc O - \text{v} \\ j_1 \quad j_2 \quad j_3 \end{array} = \begin{array}{c} i_1 \quad i_2 \quad i_3 \\ \text{u} - \boxed{B} - \text{v} \\ j_1 \quad j_2 \quad j_3 \end{array} \quad (2.16)$$

Again, the indices can be taken together in the following way: $\alpha = (ui_1j_1i_2j_2)$ and $\beta = (i_3j_3v)$:

$$A_{\alpha\gamma} O_{\gamma\beta} = B_{\alpha\beta} \quad (2.17)$$

This is a standard matrix equation and can hence be solved with linear algebra packages. Note that it is not necessary to calculate A^{-1} to obtain the solution, 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

Tensor network diagram 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 different contraction orders are shown in fig. 2.3.

2.4 MPS algorithms

This section and the following one, will introduce some different tensor network algorithms. The goal is to provide an intuitive explanation on how these algo-

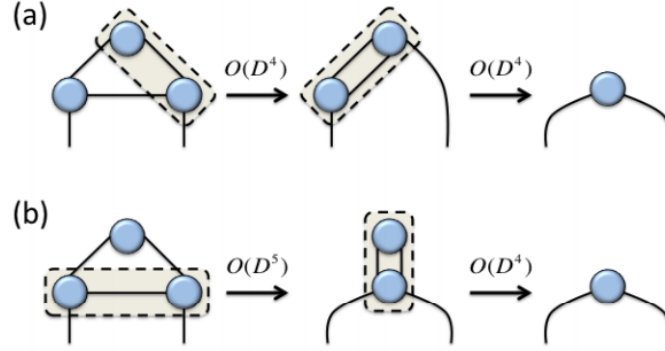


Figure 2.3: (a) Contraction of 3 tensors in $O(D^4)$ time (b) Contraction of same tensors in $O(D^5)$ time. Figure taken from [2].

bron MPS

gorithms work. For rigorous derivations and mathematical details, other sources can be read.

2.4.1 Canonical form

A translation invariant MPS has the form shown in 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

$$\begin{aligned} A_l &= \text{---} \triangleleft \text{---} = X \text{---} \bigcirc \text{---} X^{-1} \\ A_r &= \text{---} \triangleright \text{---} = Y \text{---} \bigcirc \text{---} Y^{-1} \end{aligned} \quad (2.18)$$

and

$$\text{---} \diamond \text{---} = XY^{-1} = C, \quad (2.19)$$

eq. (2.6) becomes

$$\dots \text{---} \triangleleft \triangleleft \diamond \triangleright \text{---} \dots \quad (2.20)$$

Introducing one more tensor:

$$A_c = \text{---} \square \text{---} = \text{---} \triangleleft \diamond \text{---} = \text{---} \diamond \triangleright \text{---} \quad (2.21)$$

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

$$\begin{array}{c} \text{---} \triangle \text{---} \\ | \\ \text{---} \triangle \text{---} \end{array} = \begin{array}{|c|} \hline \\ \hline \end{array} = I \quad (2.22a)$$

$$\begin{array}{c} \text{---} \triangleleft \text{---} \\ | \\ \text{---} \triangleleft \text{---} \end{array} = \begin{array}{|c|} \hline \\ \hline \end{array} = I. \quad (2.22b)$$

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.

2.4.1.1 Entropy

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

$$S = - \sum_i \alpha_i^2 \log(\alpha_i^2) \quad (2.23)$$

where α_i are the singular values of C .

2.4.1.2 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: $\langle \Psi | \hat{O} | \Psi \rangle$

$$\begin{array}{c} \dots \text{---} \triangle \text{---} \square \text{---} \triangleleft \text{---} \dots \\ | \\ \dots \text{---} \triangle \text{---} \square \text{---} \triangleleft \text{---} \dots \end{array} \quad \begin{array}{c} \square \\ | \\ \bigcirc \\ | \\ \square \end{array} = \begin{array}{|c|} \hline \square \\ | \\ \bigcirc \\ | \\ \square \\ \hline \end{array} \quad (2.24)$$

where we made use of eq. (2.22). Similar techniques allow for the calculation of correlation functions, energies, ...

2.5 Tensor network contraction

The contraction in this section is somewhat different. The MPO with the same $S_n = (i_1 i_2 \dots i_n) = (j_1 j_2 \dots j_n)$ represents the probability of finding the system in state S_n . This will be needed in section 3.4.1. On a patch, an operator X is applied. This could be a measurement.

2.5.1 1D problem

Suppose that there is an MPO representation A that gives the probability of a state, and an operator X which performs a measurement over a number of sites. The expectation value $\langle X \rangle$ is given by

$$\langle X \rangle = \frac{\text{Diagram 1}}{\text{Diagram 2}}. \quad (2.25)$$

Diagram 1 (Numerator): A horizontal chain of MPO tensors A with vertical indices connected to a chain of operator tensors X . The A tensors are arranged in a row, and the X tensors are arranged in a row above them, connected by vertical lines. Ellipses indicate the chain continues to the left and right.

Diagram 2 (Denominator): A horizontal chain of MPO tensors A with vertical indices connected to each other, forming a closed loop. Ellipses indicate the chain continues to the left and right.

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

$$\begin{aligned} Gl - A - &= \lambda Gl - \\ A - Gr &= \lambda - Gr \end{aligned} \quad (2.26)$$

The diagrams show the fixed point equations for the MPO A . The first equation shows Gl (a vector) connected to A (a tensor) from the left, resulting in λGl . The second equation shows A connected to Gr (a vector) from the right, resulting in λ (a scalar) connected to Gr .

Equation eq. (2.25) can now be easily calculated

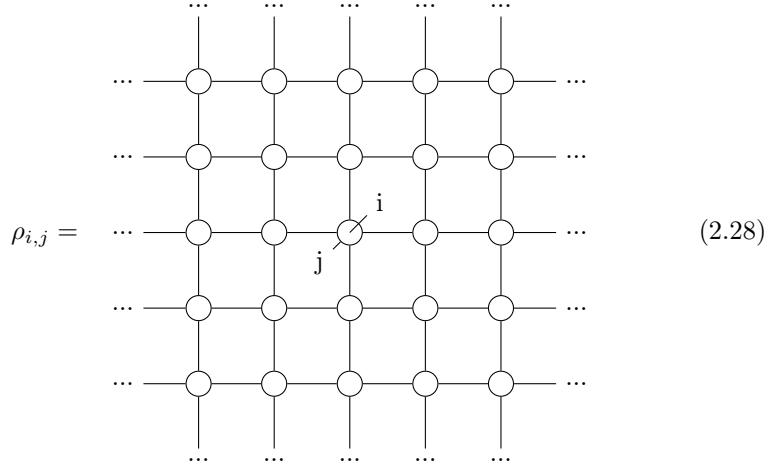
$$\langle X \rangle = \frac{\text{Diagram 3}}{\lambda^n Gr - Gr}. \quad (2.27)$$

Diagram 3 (Numerator): Similar to Diagram 1, but the chain of A tensors is terminated on the left by Gl and on the right by Gr .

Denominator: $\lambda^n Gr - Gr$, where λ^n is a scalar and Gr is a vector.

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 computed:



$$\rho_{i,j} = \quad (2.28)$$

This problem is much harder than in 1D. For instance, the related problem of calculating the expectation value of an operator, similar to section 2.4.1.2 in 1D, is #P-Hard [2].

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 is taken from [5].

2.5.3.1 Real-space renormalisation-group methods

The general idea is to coarse grain the tensor network. Of the many methods, historically the first one was Tensor renormalisation group (TRG) [6], 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 [7]. 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

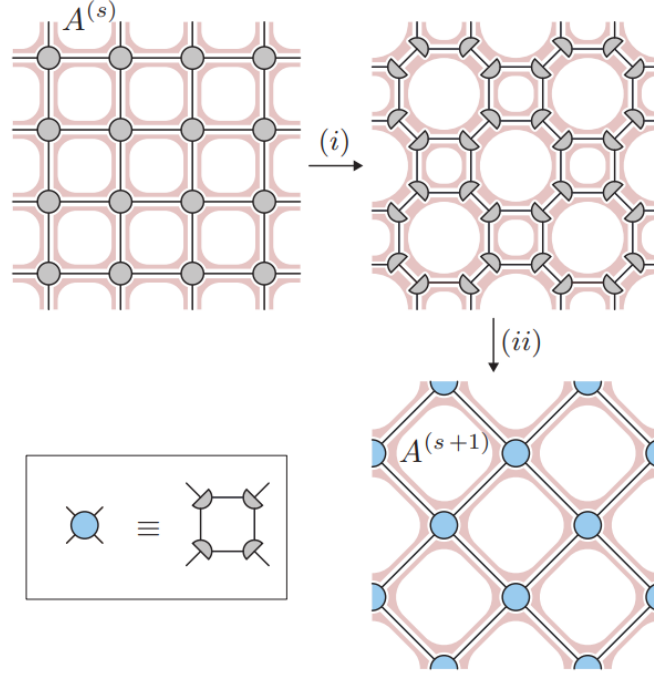


Figure 2.4: Steps in TRG procedure. Figure taken from [6].

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 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 tensor network contraction algorithms.

2.5.3.3 Boundary methods

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

$$\begin{array}{c}
 \dots \rightarrow \triangle \rightarrow \square \rightarrow \triangleleft \rightarrow \dots \\
 \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\
 \dots \rightarrow \bigcirc \rightarrow \bigcirc \rightarrow \bigcirc \rightarrow \dots
 \end{array}
 \approx
 \begin{array}{c}
 \dots \rightarrow \triangle \rightarrow \square \rightarrow \triangleleft \rightarrow \dots \\
 \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \\
 \dots \rightarrow \bigcirc \rightarrow \bigcirc \rightarrow \bigcirc \rightarrow \dots
 \end{array}
 \quad (2.29)$$

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

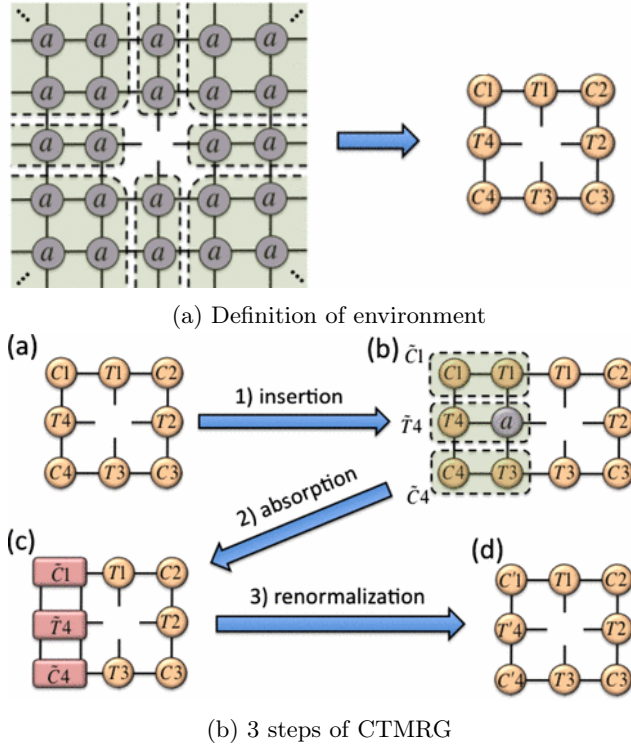


Figure 2.5: Figures adapted from [7]

encountered later on in section 3.4.3). It can also be used as a power method for contraction. The idea is shown in fig. 2.6. The boundary MPS is applied, and afterwards truncated. When this procedure is applied enough times, a fixed point is reached.

explain this

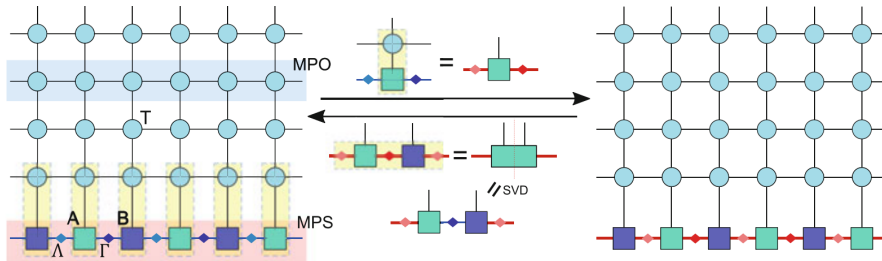


Figure 2.6: Figure taken from [4].

bond dim op figuur

1

2.5.4.1 The equations

Suppose there are tensors which fulfil the conditions stated below

(2.30a)

$$(2.30b)$$

.

$$(2.31)$$

Where the Ar tensor was connected from below on both sides and eq. (2.22) was used on the right-hand side. The blocks in eq. (2.30) form a zipper from the left and right respectively. Each application brings down one of the MPS

tensors in the following way.

(2.32)

The left and right zipper can now move towards each other, until they meet at the center. In order to become the same MPS as before the application to the MPO (eq. (2.29)), one more condition is needed:

$$\text{Gl} \text{---} \square \text{---} \text{Gr} \text{---} \bigcirc \text{---} \lambda_{AC} \text{---} \square \text{---} \quad (2.33)$$

This completely determines the problem, but one more equation is used to solve the problem. Combinationing one of the equations in 2.30, the definition of A_c eq. (2.33) and one of eq. (2.22) gives C

$$\text{---} \begin{array}{|c|} \hline \text{Gl} \\ \hline \end{array} \begin{array}{|c|} \hline \diamond \\ \hline \end{array} \begin{array}{|c|} \hline \text{Gr} \\ \hline \end{array} \text{---} = \lambda_C \text{---} \begin{array}{|c|} \hline \diamond \\ \hline \end{array} \text{---}. \quad (2.34)$$

Then eq. (2.29) is solved

(2.35)

Contracting a 2D tensor network is thus reduced to solving the eq. (2.21), eq. (2.30), eq. (2.33) and eq. (2.34) simultaneously. Inspection of the equations shows that the following cycle needs to be solved:

- $A_c, C \rightarrow A_l, A_r$ eq. (2.21)
- $A_l, A_r \rightarrow Gl, Gr$ eq. (2.30)
- $Gl, Gr \rightarrow A_c, C$ eq. (2.33) and eq. (2.34)

The calculated environment can now be used to solve the original problem. Due to symmetry, the same MPS can be applied from below. Equation (2.28) now becomes

(2.36)

As always, the identity (2.22) can be used to simplify the equation to

(2.37)

2.5.4.2 The derivation

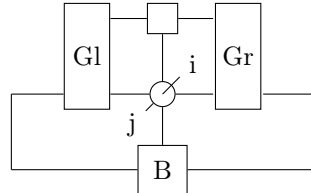
While the above derivation is reasonable, it is not very rigorous. The algorithm finds its origins in tangent space methods, such as explained in [8]. Not every state in the many body Hilbert space can be represented by an MPS. By carefully constructing the tangent space and making use of the available 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 to 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. (2.29) 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 [5]. Applying the projector \mathcal{P}_A to eq. (2.29) exactly gives rise to the equations stated earlier.

2.5.4.3 Multisite

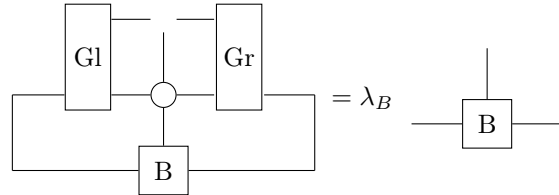
In [5], a version of VUMPS is proposed that can compute the environment of an m by n unit cell with only a linear increase in computation cost. Larger unit cells might be needed for ground states with a non-trivial unit cell.

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:

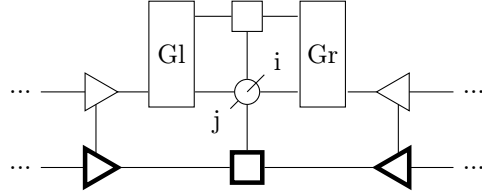

(2.38)

The reason for this doubt will be explained 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. (2.33)):

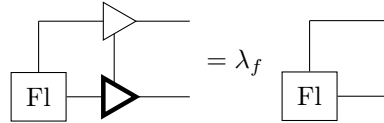

(2.39)

The idea is to use the same environments Gl and Gr , but find the fixed point from below. This has some problems: B can not be brought into a canonical form according to eq. (2.21) (in combination with the analogue of eq. (2.34)), and hence does not represent a fixed point MPS as was the case from above. This also doesn't work in the multisite setting.

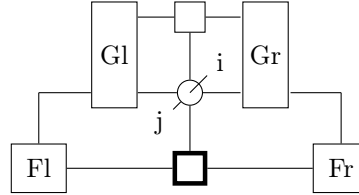
It is not completely surprising, suppose we have a fixed point MPS B (denoted in bold) from below (for instance obtained by performing the VUMPS algorithm on an MPO rotated by 180 degrees), then contracting the upper half plane and the lower half plane amounts to:


(2.40)

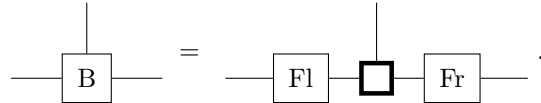
To solve this equation, the left and the right fixed points need to be calculated


(2.41)

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


(2.42)

Of course both methods could also be equal, if the following holds (up to a factor)


(2.43)

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

2.6 Conclusion

Tensor networks were introduced from a practical point of view: what different kinds exist, how to reason about and manipulate tensor networks. Some useful algorithms are discussed. One of the reasons MPS's are the standard to simulate strongly correlated systems is its successful canonical form as discussed. Another groundbreaking algorithm not discussed here is DMRG, introduced by [White](#). The difficulty of contracting an 2D tensor network 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

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. It turns out that there are a lot of interesting aspects related to phase transitions, such as universality. This principle says that the physics of many models at criticality can be captured by a limited numbers of classes, each characterized 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 properties 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 tensor network framework are discussed. The cluster expansions from this thesis, will also be a tensor network 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 properties 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. [9]. 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 [10], 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 BKT transition.

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 : $[H, U] = 0$ [11]. 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, scale invariance systems become 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[11].

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.

3.2.4 Critical exponents for spin systems

Table 3.1 defines some critical exponents for the Ising system. 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: $m \sim |t|^\beta$, $\xi(t) \approx |t|^{-\nu}$. At the critical temperature near a quantum phase transition $m \approx |g - g_c|^{\frac{1}{\delta}}$.

Table 3.1: Definition of critical exponents for Ising model

Symbol	name
m	magnetisation
ξ	correlation length
g	external field
t	reduced temperature
d	dimension

3.2.5 Finite-size scaling

Phase transitions only occur for systems with an infinite number of degrees of freedom. 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 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

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

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.

reference

line crossing

3.2.5.1 Finite-size scaling for MPS

The finite-size scaling for MPS is somewhat different. In [12], it is argued that δ can take the place of $1/L$. Suppose λ_i are the eigenvalues of eq. (2.31) ordered from the largest real part to smallest. Then

$$\epsilon_i = -\log(|\lambda_i|) \quad (3.2)$$

and

$$\delta = -\sum_i c_i \epsilon_i \quad \sum_i c_i = 0 \quad (3.3)$$

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 [13], 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:

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

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

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

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 [14]. For the 2D Ising model, the central charge is $c = 1/2$. A scaling relation used in the results chapter is [15]

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

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: 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 [16].

3.3 Models

The goal of numerical techniques is to simulate the physics of real world systems. These are, to some extent, captured by different models. Models are a simplified mathematical description that captures some relevant physics of more complicated systems. This section introduces some specific models, their

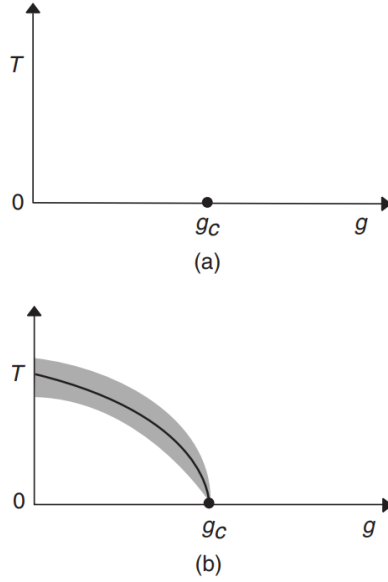


Figure 3.1: Two possible phase diagrams of a system near a quantum phase transition. In both cases there is a quantum critical point at $g = g_c$ and $T = 0$. In (b), there is a line of $T > 0$ second-order phase transitions terminating at the quantum critical point. The theory of phase transitions in classical systems driven by thermal fluctuations can be applied within the shaded region of (b). Figure and caption taken from [16].

relevance and some properties. These models will be used later to benchmark the developed tensor network expansion.

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 [17].

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 quantum Ising 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.

q potts,...

Many generalizations exist for the Ising model.

3.3.1.1 Classical Ising

The classical Ising model is given by the Hamiltonian

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

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.

blabla

3.3.1.1.1 1D The classical 1D model was solved analytically by Lens.

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 the is solved analytically. For higher dimensions, no analytical solution is known. For these cases, we need to use numerical techniques if we want to understand the behaviour of these models. 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. Remarkably, as will be explained in section 3.2, the physics at the phase transition does remain invariant when the

is this true??

lattice is changed.

3.3.1.2 Quantum 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 quantum Ising model differs from the classical model. In the quantum Ising model, the operators no longer commute with each other. An example is the transversal Ising model given by the following Hamiltonian:

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

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 self duality. The fact that this model has a phase transition can also be understood with section 3.3.4.

citation

3.3.1.2.2 2D The 2D phasediagram 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 A part of this phase diagram will be reproduced in section 5.4.3.

value

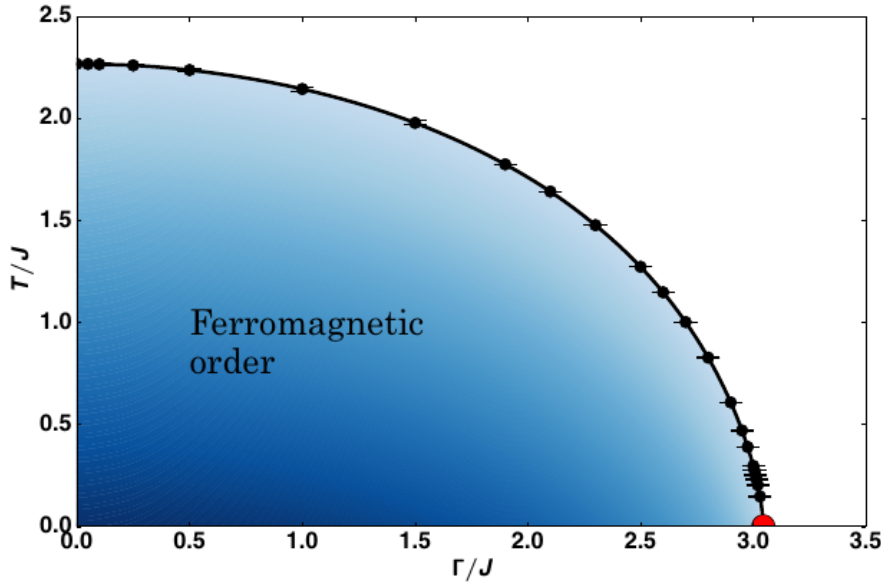


Figure 3.2: Phase diagram for 2D transversal Ising model. Figure taken from [18].

3.3.2 Heisenberg

The Heisenberg model is given by

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

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

$$\hat{H} = -J \left(\sum_{\langle ij \rangle} \vec{\sigma} \cdot \vec{\sigma} \right). \quad (3.10)$$

This model has an $SU(3)$ symmetry, meaning it stays the same when rotated in 3D space.

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

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

we can rewrite the partition function as

$$\begin{aligned} Z &= \sum_n e^{-\beta E_n} \\ &= \sum_n \langle n | e^{-\beta \hat{H}} | n \rangle \\ &= \sum_{n_1 \dots n_M} \langle n_1 | e^{-\beta \hat{H}/M} | n_2 \rangle \langle n_2 | e^{-\beta \hat{H}/M} | n_3 \rangle \dots \langle n_M | e^{-\beta \hat{H}/M} | n_1 \rangle \\ &= \sum_{n_1 \dots n_M} \langle n_1 | e^{-\beta \hat{H}/M} | n_2 \rangle \langle n_2 | e^{-\beta \hat{H}/M} | n_3 \rangle \dots \langle n_M | e^{-\beta \hat{H}/M} | n_1 \rangle \end{aligned} \quad (3.12)$$

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 transversal field Ising model can be mapped to the 3D classical Ising model [19].

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 $|\Psi(t)\rangle = \exp(-i\hat{H}t) |\Psi(0)\rangle$. 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 $p_i = \exp(-\beta\hat{H}_i)$. 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:

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

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

$$p_i = \frac{\sum e^{-\beta E_i}}{Z}. \tag{3.14}$$

An useful quantity is the density matrix ρ :

$$\begin{aligned} \rho &= \sum_j p_j |\Psi_j\rangle \langle \Psi_j| \\ &= \sum_j \frac{e^{-\beta \hat{H}}}{Z} |\Psi_j\rangle \langle \Psi_j|. \end{aligned} \tag{3.15}$$

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

$$\begin{aligned} Z &= \text{Tr}(\rho) \\ \langle X \rangle &= \text{Tr}(\rho \hat{X}). \end{aligned} \quad (3.16)$$

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:

$$\begin{aligned} C(r, t) &= \langle \hat{X}(0, 0) \hat{X}(r, t) \rangle \\ &= \langle \hat{X}(0) e^{i\hat{H}t} \hat{X}(r) e^{-i\hat{H}t} \rangle \end{aligned} \quad (3.17)$$

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) [2]:

$$|E_0\rangle = \lim_{\beta \rightarrow \infty} \frac{e^{-\beta \hat{H}} |\Psi(0)\rangle}{\langle \Psi(\beta) | \Psi(\beta) \rangle}, \quad |\Psi(\beta)\rangle = e^{-\beta \hat{H}} |\Psi(0)\rangle. \quad (3.18)$$

3.4.3 Tensor network methods

In the following section I will give a very short review of the current tensor network methods to simulate real or imaginary time evolution. This overview is mainly based on the review paper [20].

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

$$\hat{H} = \hat{H}_{\text{even}} + \hat{H}_{\text{odd}} \quad (3.19)$$

$$\begin{aligned} \hat{U} &= e^{-i\delta \hat{H}_{\text{even}}} e^{-i\delta \hat{H}_{\text{odd}}} e^{-i\delta [\hat{H}_{\text{even}}, \hat{H}_{\text{odd}}]} \\ &\approx e^{-i\delta \hat{H}_{\text{even}}} e^{-i\delta \hat{H}_{\text{odd}}} \end{aligned} \quad (3.20)$$

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 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 [21]. The idea is to generalise

$$1 + \delta \sum_x H_x \rightarrow \prod_x (1 + \delta H_x) \quad (3.21)$$

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 [20].

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 MPS's, 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

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

Where the right-hand side is projected on the tangent space, because the left-hand side is also a tangent vector. (See [8]).

3.5 Conclusion

An introduction to phases of matter, and their surprising structure was discussed. 2 models, namely Heisenberg model and the Ising 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 tensor network 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 method to construct the operator $e^{-\beta\hat{H}}$, is explained in detail. The basis of the method was first introduced in [22]. 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, an analogous construction as in 1D will be presented. As can be expected, also 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.

$$\begin{array}{c} i \\ | \\ 0 \text{---} \bigcirc \text{---} 0 \\ | \\ j \end{array} = \bigcirc \quad (4.1)$$

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

$$\begin{array}{c} i_1 \\ | \\ \text{---} \bigcirc \text{---} \bigcirc \text{---} \\ | \quad | \\ j_1 \quad j_2 \end{array} \quad \begin{array}{c} i_2 \\ | \\ \text{---} \bigcirc \text{---} \bigcirc \text{---} \\ | \quad | \\ j_2 \quad j_1 \end{array} = \bigcirc \text{---} \bigcirc. \quad (4.2)$$

As a reminder, unlabeled indices such as in

$$\begin{array}{c} i_1 \\ | \\ \text{---} \bigcirc \text{---} \bigcirc \text{---} \bigcirc \text{---} \\ | \quad | \quad | \\ j_1 \quad j_2 \quad j_3 \end{array} \quad \begin{array}{c} i_2 \\ | \\ \text{---} \bigcirc \text{---} \bigcirc \text{---} \bigcirc \text{---} \\ | \quad | \quad | \\ j_2 \quad j_1 \quad j_3 \end{array} \quad \begin{array}{c} i_3 \\ | \\ \text{---} \bigcirc \text{---} \bigcirc \text{---} \bigcirc \text{---} \\ | \quad | \quad | \\ j_3 \quad j_2 \quad j_1 \end{array} = \bigcirc \text{---} \bigcirc \text{---} \bigcirc \quad (4.3)$$

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 the Hamiltonian operator \hat{H}

$$\hat{H} = \left(\sum_{\langle ij \rangle} H_2^i H_2^j + \sum_i H_1^i \right) \quad (4.4)$$

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

$$\begin{aligned} H(\bigcirc \text{---} \bigcirc \text{---} \bigcirc) &= H_1 \otimes 1 \otimes 1 \\ &+ 1 \otimes H_1 \otimes 1 \\ &+ 1 \otimes 1 \otimes H_1 \\ &+ H_2 \otimes H_2 \otimes 1 \\ &+ 1 \otimes H_2 \otimes H_2 \\ &. \end{aligned} \quad (4.5)$$

4.1.2 Idea

This chapter shows the main construction of this dissertation. A cluster expansion is used to approximate $e^{\hat{H}}$ 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

$$\bigcirc = \exp(-\beta H(\bigcirc)). \quad (4.6)$$

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.

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

Contraction of larger network lead to many terms, such as

$$\bigcirc \overset{1}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc. \quad (4.8)$$

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: $\begin{array}{c} i \\ | \\ \bigcirc \text{---} 0 \text{---} 1 \\ | \\ j \end{array}$ and

of course also $\begin{array}{c} i \\ | \\ \bigcirc \text{---} 1 \text{---} 0 \\ | \\ j \end{array}$. As can be seen, the dimension of virtual level

1 needs to be d^2 , with d the dimension of physical level. Although different possible constructions already differ in the next step, one more step is added to

make the construction and notation clear.

$$\begin{aligned}
 \bigcirc \overset{1}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc &= \exp -\beta H(\bigcirc \text{---} \bigcirc \text{---} \bigcirc) \\
 &\quad - \bigcirc \overset{0}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \\
 &\quad - \bigcirc \overset{1}{\text{---}} \bigcirc \overset{0}{\text{---}} \bigcirc \\
 &\quad - \bigcirc \overset{0}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc \\
 &= \exp -\beta H(\bigcirc \text{---} \bigcirc \text{---} \bigcirc) \\
 &\quad - \bigcirc \text{---} \bigcirc \text{---} \bigcirc
 \end{aligned} \tag{4.9}$$

This is called a cluster expansion of order 3, because there are 3 connected sites solved exactly. The right-hand side of eq. (4.9) 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

$$\boxed{\bigcirc \overset{1}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc}, \tag{4.10}$$

the right-hand side of eq. (4.9) is implied. In the following section, different types will be discussed. For every chain length, 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 [22]. The first few blocks in the expansion are

$$\begin{aligned}
 &\bigcirc \\
 &\bigcirc \overset{1}{\text{---}} \bigcirc \\
 &\bigcirc \overset{1}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc \\
 &\bigcirc \overset{1}{\text{---}} \bigcirc \overset{2}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc \\
 &\bigcirc \overset{1}{\text{---}} \bigcirc \overset{2}{\text{---}} \bigcirc \overset{2}{\text{---}} \bigcirc \overset{1}{\text{---}} \bigcirc.
 \end{aligned} \tag{4.11}$$

The following types of blocks appear in the cluster expansion:

$$\begin{array}{c} | \\ \text{n} \text{---} \bigcirc \text{---} \text{m} \\ | \end{array}, \quad \begin{array}{c} | \\ \text{m} \text{---} \bigcirc \text{---} \text{n} \\ | \end{array} \quad \text{and} \quad \begin{array}{c} | \\ \text{n} \text{---} \bigcirc \text{---} \text{n} \\ | \end{array} \quad (4.12)$$

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^{2*n} . 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. $\bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^0 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc$. 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

$$\begin{array}{l} \bigcirc \\ \bigcirc \text{---}^1 \text{---} \bigcirc \\ \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^1 \text{---} \bigcirc \\ \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^2 \text{---} \bigcirc \text{---}^3 \text{---} \bigcirc \\ \bigcirc \text{---}^1 \text{---} \bigcirc \text{---}^2 \text{---} \bigcirc \text{---}^3 \text{---} \bigcirc \text{---}^4 \text{---} \bigcirc \end{array} \quad (4.13)$$

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: $O^{mn} = U^\dagger$.

$$\begin{array}{c} i_n \quad i_{n+1} \\ | \quad | \\ \text{m} \text{---} \bigcirc \text{---} \text{n} \text{---} \bigcirc \text{---} 0 \\ | \quad | \\ j_n \quad j_{n+1} \end{array} = U^n \Sigma V^\dagger \quad (4.14)$$

4.2.2.1 Dimension

From the construction the bond dimension grows from the left to the right. For the last step, there are only d^2 non-zero singular values. All steps add d^2 to the dimension. For the last step, only d^2 non-zero singular values need to be kept. With the following notation:

$$\begin{array}{c}
 i \\
 | \\
 \text{---} \text{m} \bigcirc \text{n \text{---}} \\
 | \\
 j
 \end{array} = A_{(\alpha ij)\beta}^n$$

$$\begin{array}{c}
 i \\
 | \\
 \text{---} \text{n} \bigcirc \text{0 \text{---}} \\
 | \\
 j
 \end{array} = B_{(\alpha ij)\beta}^n$$
(4.15)

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

$$\begin{aligned}
 S &= A^n A'^n \\
 T &= A'^n B'^{n+1}.
 \end{aligned}$$
(4.16)

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

$$\begin{aligned}
 S^\dagger &= (U \Sigma_1) Q^\dagger \\
 T^\dagger &= (V \Sigma_2) Q^\dagger.
 \end{aligned}$$
(4.17)

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

$$\begin{array}{c}
\bigcirc \\
\bigcirc \text{---}^1 \text{---} \bigcirc \\
\bigcirc \text{---}^{1'} \text{---} \bigcirc \text{---}^{1'} \text{---} \bigcirc \\
\bigcirc \text{---}^{1''} \text{---} \bigcirc \text{---}^{2''} \text{---} \bigcirc \text{---}^{3''} \text{---} \bigcirc \\
\bigcirc \text{---}^{1'''} \text{---} \bigcirc \text{---}^{2'''} \text{---} \bigcirc \text{---}^{3'''} \text{---} \bigcirc \text{---}^{4'''} \text{---} \bigcirc
\end{array} \tag{4.18}$$

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. Looking 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 performed in most of the cases worse. The blocks appearing in type D are as follows:

$$\begin{array}{c} | \\ \text{m} \text{---} \bigcirc \text{---}^n \text{---} D_n \text{---}^n \text{---} \bigcirc \text{---}^m \\ | \end{array} \text{ and } \begin{array}{c} | \\ \text{n} \text{---} \bigcirc \text{---}^n \\ | \end{array}, \tag{4.19}$$

which is similar to type A.

$$\begin{array}{c} | \\ \text{m} \text{---} \bigcirc \text{---}^n \text{---} D_n \text{---}^n \text{---} \bigcirc \text{---}^m \\ | \end{array} = \begin{array}{c} | \\ \text{n} \text{---} \boxed{L_n^{-1} M_{2n+2} R_n^{-1}} \text{---}^n \\ | \end{array} \tag{4.20}$$

$$= U \Sigma V^\dagger$$

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, large chains are not suppressed. If ϕ 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:

O_{00}	O_{01}	O_{12}	$-2O_{01}$	$-2O_{12}$	O_{01}	O_{12}	$O_{01}D_1^{1/2}$
O_{10}	O_{21}						
O_{10}	O_{21}				O_{11}	O_{22}	
$D_1^{1/2}O_{10}$							$D_1^{-1/2}O_{12}D_2^{1/2}$
							$D_2^{1/2}O_{21}D_1^{-1/2}$

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:

$$\begin{array}{c}
 \bigcirc \\
 \bigcirc \text{---} 1 \text{---} \bigcirc \\
 \bigcirc \text{---} 1 \text{---} \bigcirc \text{---} 1' \text{---} \bigcirc \\
 \bigcirc \text{---} 1 \text{---} \bigcirc \text{---} 2 \text{---} \bigcirc \text{---} 1' \text{---} \bigcirc \\
 \bigcirc \text{---} 1 \text{---} \bigcirc \text{---} 2 \text{---} \bigcirc \text{---} 2' \text{---} \bigcirc \text{---} 1' \text{---} \bigcirc
 \end{array} \tag{4.21}$$

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:

$$\begin{array}{c}
 \bigcirc \\
 \bigcirc \xrightarrow{1'} \bigcirc + \bigcirc \xrightarrow{1'} \bigcirc \\
 \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{1} \bigcirc \\
 \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{2} \bigcirc \xrightarrow{1} \bigcirc + \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{2'} \bigcirc \xrightarrow{1} \bigcirc \\
 \bigcirc \xrightarrow{1} \bigcirc \xrightarrow{2} \bigcirc \xrightarrow{2} \bigcirc \xrightarrow{1} \bigcirc
 \end{array} \tag{4.22}$$

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 given chain. In the next step, $O^{n,n}$ is added as usual. The idea here is once again to keep this block small, in order to not cause any divergences.

4.2.7 Conclusion

Many 1D constructions 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, and 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.

$$\bigcirc = \begin{array}{c} \text{0} \\ | \\ \text{0} \quad \text{i} \text{0} \\ \diagup \quad \diagdown \\ \text{j} \text{0} \\ | \\ \text{0} \end{array} \quad (4.23)$$
$$\begin{array}{c}
 \textcircled{} \xrightarrow{1} \textcircled{} \\
 | \\
 \textcircled{}
 \end{array}
 \quad (4.24)$$

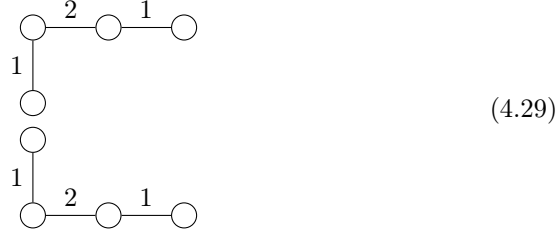
(4.25)

(4.26)

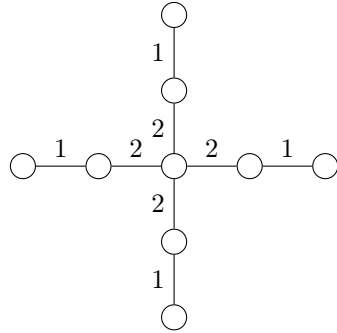
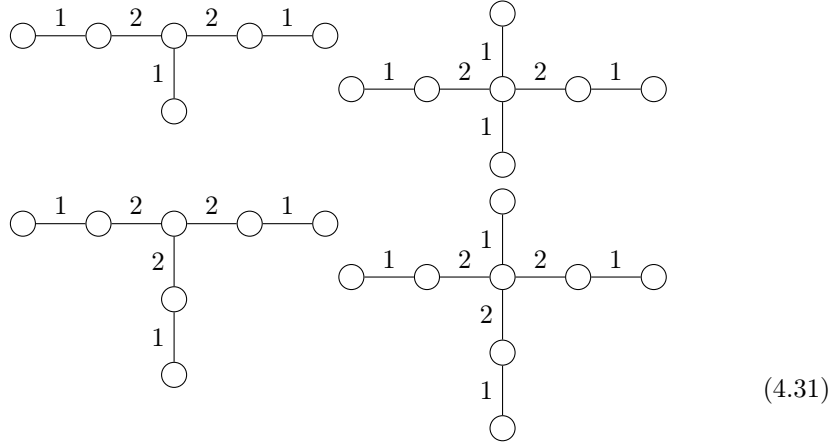
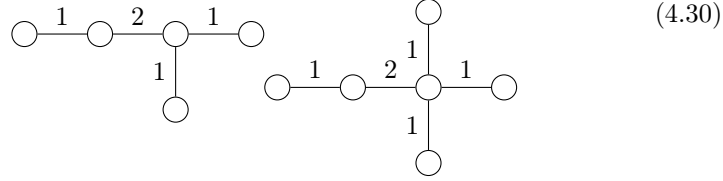
(4.27)

$$\begin{array}{c} 1 \quad 2 \quad 1 \\ \bigcirc - \bigcirc - \bigcirc - \bigcirc \end{array} \quad (4.28)$$

Again, 2 blocks are introduced. For other variations of the linear chain, only one block needs to be solved



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 T and + blocks are created. Some of these blocks are shown here:

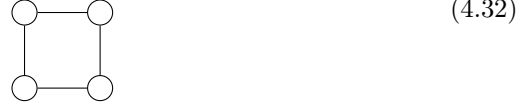


For each block show, there are still multiple permutations of the legs possible. It is clear that a completely automated solver is needed to construct all these different blocks. From here on, the construction generalises easily to higher block numbers, and to higher dimensions.

The difference between eq. (4.30) and the blocks in eq. (4.30) is in the larger chain. For eq. (4.30), only chains of order 4 are present, while eq. (4.31) has chains of length 5. It seems that when a virtual level is present, it is most advantageous to create both chains and all blocks, 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.



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:



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:



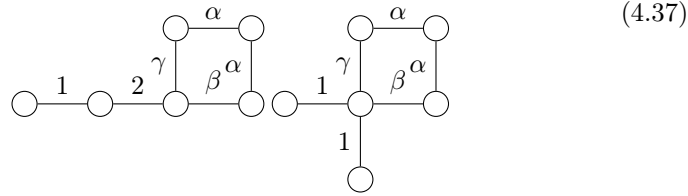
With the given blocks, the following combination is also possible



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 options in 2D. One example is:

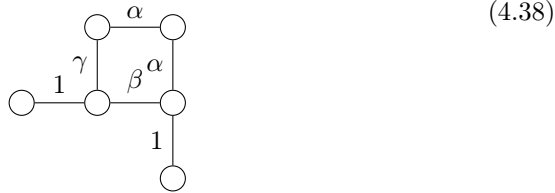


No other combinations are possible, except a loop with on one corner an extension. Of course, there is no need to stop here, the following blocks can now be constructed easily.



4.3.2.2 Double extensions

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



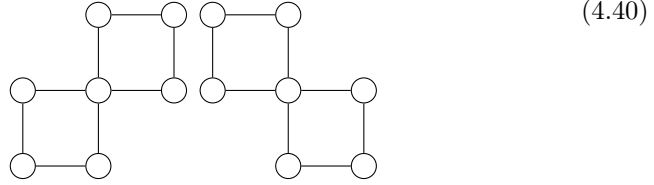
But in order to make a meaningful change in the residual error, the bond dimension of both α and β needs to be enlarged significantly. It is more advantageous to introduce yet another level δ , which forms the link between the 2 parts. As both corner tensors can be optimised at once, the total bond dimension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



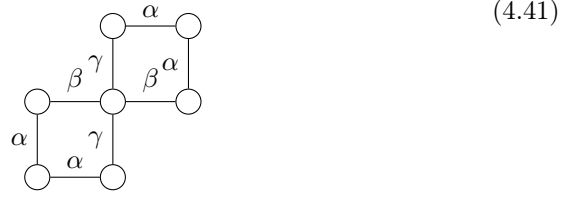
4.3.2.3 Larger loops

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 shape:



One way to solve this, without breaking the rotation pattern of β and γ , is



But this block introduces an infinite tiling, corrupting the expansion.

4.3.2.4 Other ideas

Add more

Here it starts to get really tricky.

4.3.2.5 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 [23], but needs to be deduced during the construction. In practice, a bond dimension of 6 is enough to fully solve eq. (4.33) (for physical dimension 2). For eq. (4.36), 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. (4.24) further by imposing rotation symmetry of the PEPO legs:

$$\bigcirc \text{---} 1 = \text{---} 1 \bigcirc = \begin{array}{c} \bigcirc \\ | \\ 1 \end{array} = \begin{array}{c} 1 \\ | \\ \bigcirc \end{array} \quad (4.42)$$

In this way, only the blocks unique up to a permutation of the legs need to be solved.

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. The reason for this will be discussed in section 6.4.5.2.

Chapter 5

Framework implementation

Experience is the name everyone
gives to their mistakes.

Oscar Wilde

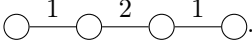
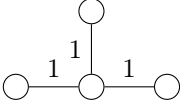
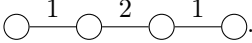
5.1 Introduction

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 `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

The construction can be written down quite compactly as done in the previous section, but actually implementing the code in full generality is somewhat more complex. In practice, the 1D and 2D implementation were performed separately. Where a part of the code but mainly the ideas and lessons learned from 1D were

taken to the 2D code. Of course, the 1D construction is just a subset of the 2D implementation. In fact, the 2D implementation even outperforms the 1D implementation for reasons with will be explained later. The main focus of this chapter will go to the 2D implementation. At the end some particular optimisations in 1D will be highlighted. 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
 
 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. Linear block consist of a tree structure, where the new block is the root of the tree, and all the branches need to be inverted. Let $I^m = (i_1^1 i_2^1 \dots i_{n_1}^1)$ 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

$$A_{I_1 \alpha^1}^1 A_{I_2 \alpha^2}^2 \dots A_{I_m \alpha^m}^m X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B_{I_1 I_2 \dots I_m j} \quad (5.1)$$

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

$$\text{Diagram (5.2)} \quad (5.2)$$

$$\text{Diagram (5.3)} \quad (5.3)$$

$$\text{Diagram (5.4)} \quad (5.4)$$

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 different matrices A_i to one matrix $A = A^1 \otimes A^2 \dots \otimes A^m$. The equation to solve is

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

The most 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 , 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:

$$A = U\Sigma V^\dagger \quad (5.6)$$

$$A^+ = V\Sigma^+U^\dagger, \quad (5.7)$$

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. The problem with this full inverse is that the bond dimension increases very fast: matrix A has dimension $d^{2\Sigma_m n_m} \times d^{2\Sigma_m n_m}$. 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:

$$\begin{aligned} A_{I^1\alpha^1}^1 X_{\alpha^1 I^2 \dots I^m j} &= B_{I^1 I^2 \dots I^m j} \\ A_{I^2\alpha^2}^2 X_{\alpha^1 \alpha^2 I^3 \dots I^m j} &= B_{\alpha^1 I^2 \dots I^m j} \\ &\vdots \\ A_{I^m\alpha^m}^m X_{\alpha^1 \alpha^2 \dots \alpha^m j} &= B_{\alpha^1 \alpha^2 \dots \alpha^{m-1} I^m j} \end{aligned} \quad (5.8)$$

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.

5.2.1.3 Sparse full inverse

Luckily the problem can be resolved by first performing an SVD decomposition of $A_{I^m\alpha^m}^m = U_{I^m\beta^m}^m S_{\beta^m\gamma^m}^m V_{\gamma^m\alpha^m}^{m\dagger}$ 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 $S = S^1 \otimes S^2 \dots \otimes S^m$ 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

$$A_{I^1 \alpha^1}^1 A_{I^2 \alpha^2}^2 \cdots A_{I^m \alpha^m}^m X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B_{I_1 I_2 \dots I_m j} \quad (5.9)$$

$$\begin{aligned} & U_{I_1 \beta^1}^1 S_{\beta^1 \gamma^1}^1 V_{\gamma^1 \alpha^1}^{1\dagger} \\ & U_{I_2 \beta^2}^2 S_{\beta^2 \gamma^2}^2 V_{\gamma^2 \alpha^2}^{2\dagger} \cdots \\ & U_{I_m \beta^m}^m S_{\beta^m \gamma^m}^m V_{\gamma^m \alpha^m}^{m\dagger} \\ & X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B_{I_1 I_2 \dots I_m j} \end{aligned} \quad (5.10)$$

$$\begin{aligned} & S_{(\beta^1 \beta^2 \dots \beta^m)(\gamma^1 \gamma^2 \dots \gamma^m)} \\ & V_{\gamma^1 \alpha^1}^{1\dagger} V_{\gamma^2 \alpha^2}^{2\dagger} \cdots V_{\gamma^m \alpha^m}^{m\dagger} \\ & X_{\alpha^1 \alpha^2 \dots \alpha^m j} = B'_{\beta_1 \beta_2 \dots \beta_m j}. \end{aligned} \quad (5.11)$$

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 [24]. The Q 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 pseudoinversion algorithm.

5.2.2 Extension

The linear solver is made for linear problems. Nevertheless, it can solve every local patch appearing in a map, such as 2 neighbouring sites. These sites are split using an SVD decomposition. Another example is the following corner block, which can perfectly be solved with the linear solver.

figure



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

5.2.3 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.

5.2.3.1 Automatic differentiation

With some care, the Jacobian can be calculated for a general tensor network 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). $\frac{\partial B_{I^1 I^2 \dots I^m j}}{\partial X_{\alpha_1 \alpha_2 \dots \alpha_m j}} = A_{I^1 I^2 \dots I^m \alpha_1 \alpha_2 \dots \alpha_m}$. If a tensor appears in multiple places, the sum rule for derivatives has to be used.

5.2.3.2 Symmetry

The nonlinear solver can handle rotated and permuted blocks. For instance, a simple loop (square) can be solved by rotating one tensor $O_{\alpha\alpha 00}^j$ 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. (4.42).

5.2.3.3 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.

5.2.4 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.

5.2.4.1 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. (4.33) 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 framework provides some tools to transform these maps into other maps, for instance by removing 1 site.

reformulate

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 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 $T_{\alpha\beta\gamma\delta}^i$ 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 $\exp(\hat{H})$ 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

$$\exp(H_{d^N} - N\alpha I) = Q \exp(D - N \log(\alpha) I) Q^\dagger \quad (5.13)$$

$$= Q \begin{bmatrix} \exp(D_{11} - N \log(\alpha)) & & \\ & \ddots & \\ & & \exp(D_{d^N d^N} - N \log(\alpha)) \end{bmatrix} Q^\dagger \quad (5.14)$$

$$= \frac{\exp(H_{d^N})}{\alpha^N}. \quad (5.15)$$

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.

5.3.4 Internal representation

Two main internal representations are used to construct the given MPO. Either, the MPO is stored as a cell of matrices, or as one big matrix where the blocks are added to during the construction. The output type can be chosen. For some types, sparse matrices are used during the construction. Given that MATLAB doesn't support multidimensional matrices by default, this[25] 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:

$$\begin{array}{c} i \\ | \\ \alpha \text{---} \bigcirc_{O_n} \text{---} \beta \\ | \\ j \end{array} \cong U_{\alpha(ij\beta)}^n \quad (5.16)$$

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

$$\begin{array}{c} i \\ | \\ \beta \text{---} \bigcirc \text{---} \gamma \\ | \\ j \end{array} O_n^{-1} \cong U_{(ij\beta)\gamma}^{n\dagger} \quad (5.17)$$

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

$$\begin{array}{c} \alpha \text{---} \bigcirc \text{---} \bigcirc \text{---} \dots \text{---} \bigcirc \text{---} 0 \\ | \quad | \quad \quad \quad | \end{array} O_n^{-1} \quad O_m^{-1} \quad \dots \quad O_1^{-1} \quad (5.18)$$

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 arch length in the m - T plan can be changed to require more points in the m direction than T direction.

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 superior solvers in 2D

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 MPO's 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 `pIsing2D_par`. 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 from the ground up (except the fitting code). 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.

5.6.2 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. For instance, there are functions generating and manipulating maps, which depends on the geometry of the problem. Other functions only use the bonds and contractions listed in that map object, and hence another lattice can be used without changing the other components too much.

5.6.3 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, which aren't a limiting factor.

5.6.4 Lattices

The models studied were all on a square lattice. The universal physics does not depend on the details of the model, such as the 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.

5.6.5 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. Including them could push the 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 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 transversal field Ising 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 results, 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

$$R = d^{2n} \times 16 \text{ bytes} \quad (6.1)$$

which means a 14 site chain already takes up more than 4 GB of RAM. The complexity to calculate a matrix exponential scales as $O(n^3)$ [26]. 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. [27]

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.

$$\epsilon = \frac{\left\| e^{-\beta \hat{H}_n} - \begin{array}{c} 0 \quad \bigcirc \quad \bigcirc \quad \dots \quad \bigcirc \quad 0 \\ \text{---} \end{array} \right\|_2}{\left\| e^{-\beta \hat{H}_n} \right\|_2} \quad (6.2)$$

This norm can only be calculated for a finite number of sites. The influence of the number of sites for a linear and cyclic 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.

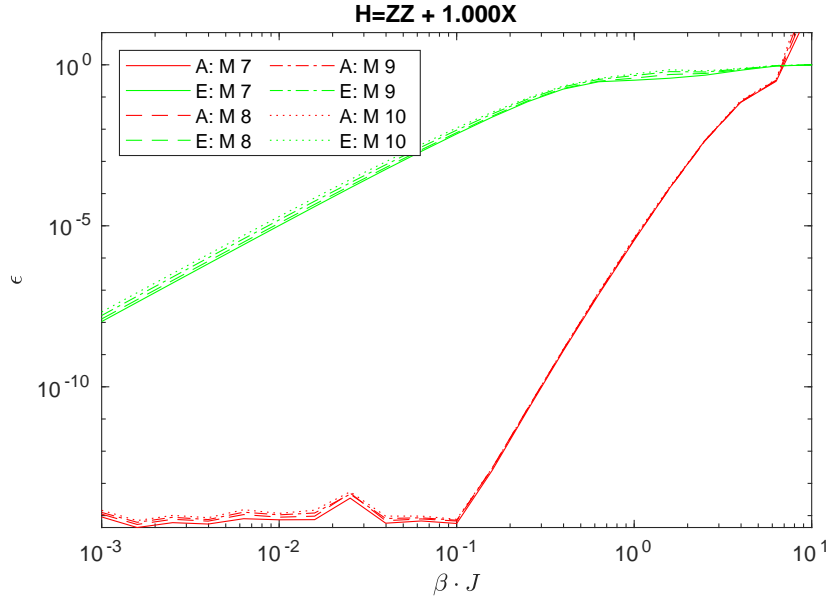
Calculating the cyclic norm comes at the extra cost of contracting a cyclic tensor network. In this chapter, the cyclic norm will be given for $M=11$ sites.

6.2.2 Inversion procedure

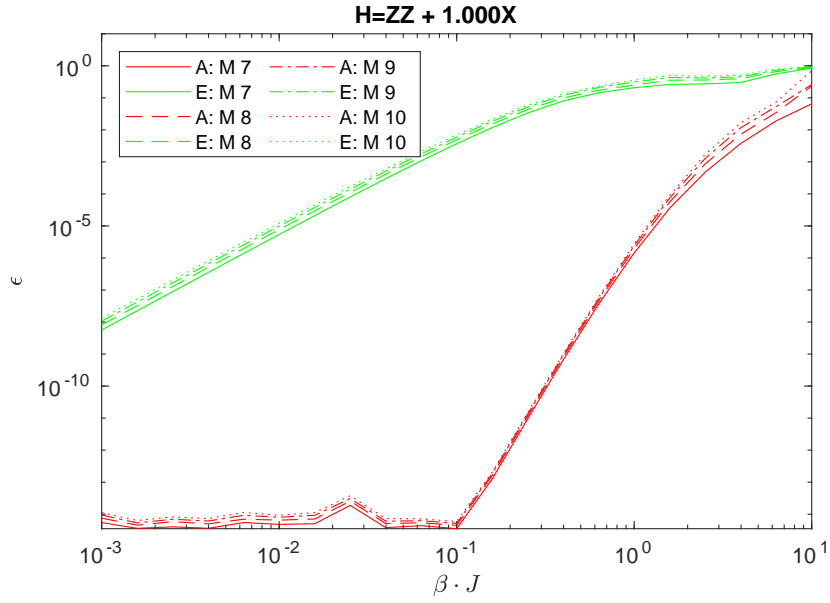
6.2.2.1 Full pseudoinversion

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 transversal Ising 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.



(a) Cyclic error



(b) Linear error

Figure 6.1: Different error measures for 1D transversal Ising model

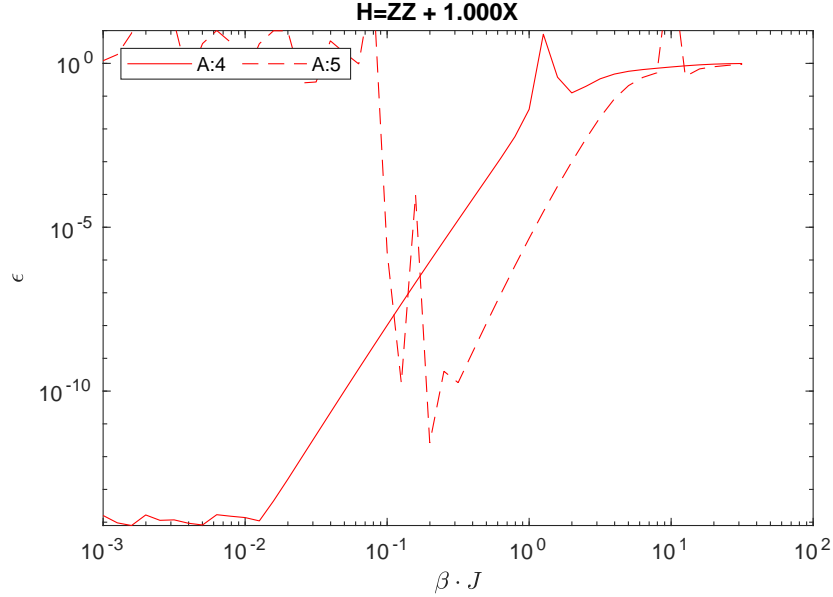


Figure 6.2: Error for transversal Ising, computed with full inversion

6.2.3 Models

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

6.2.3.1 Ising

The results for the different types are all 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. Taken together with its large bond dimension, this expansion and all other strict variants are seen to be of no use.

Now let's 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,

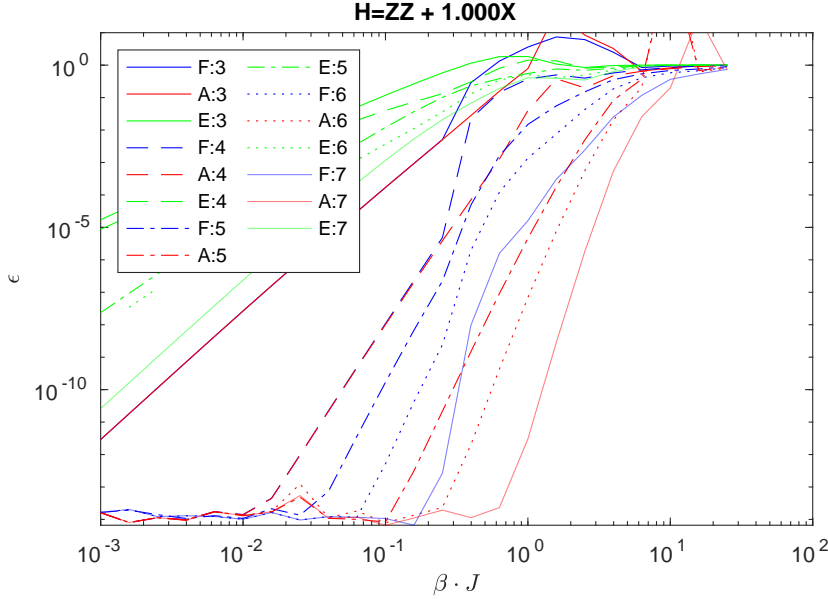


Figure 6.3: Comparison type A, E and F for Transversal Ising.

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.

6.2.3.3 Random

To give a representative overview for random Hamiltonians, several simulations were run. The construction for the random Hamiltonian was explained in section 3.3.3. Figure 6.5 shows the results for 2 random Hamiltonians. This test result is added to ensure that the chosen models are not too 'simple'. 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 .

6.2.4 Real time evolution

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

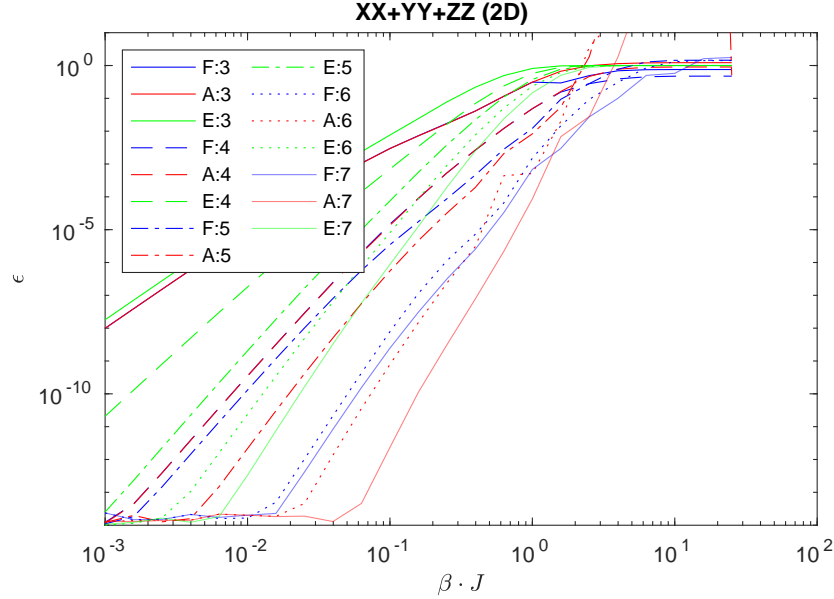


Figure 6.4: Comparison type A, E and F for Heisenberg model.

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 worse. Type F keeps the inverses well-defined, 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. These results are twofold. On the one hand, a similar measure as in 1D is calculated to measure how good a method works on a specific 2D grid. Here, we will need to settle for what can be computed. As a second measure, and from physical point of view the most interesting one,

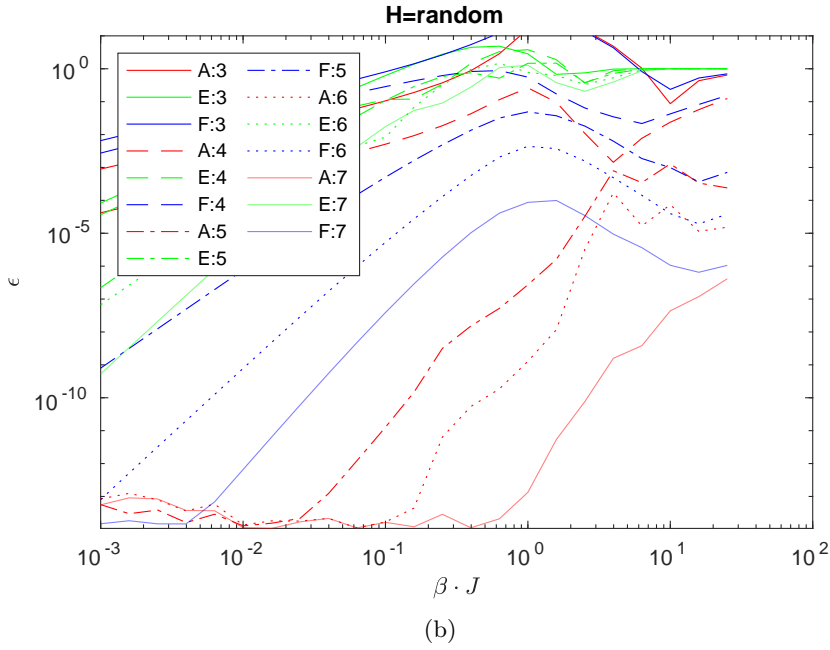
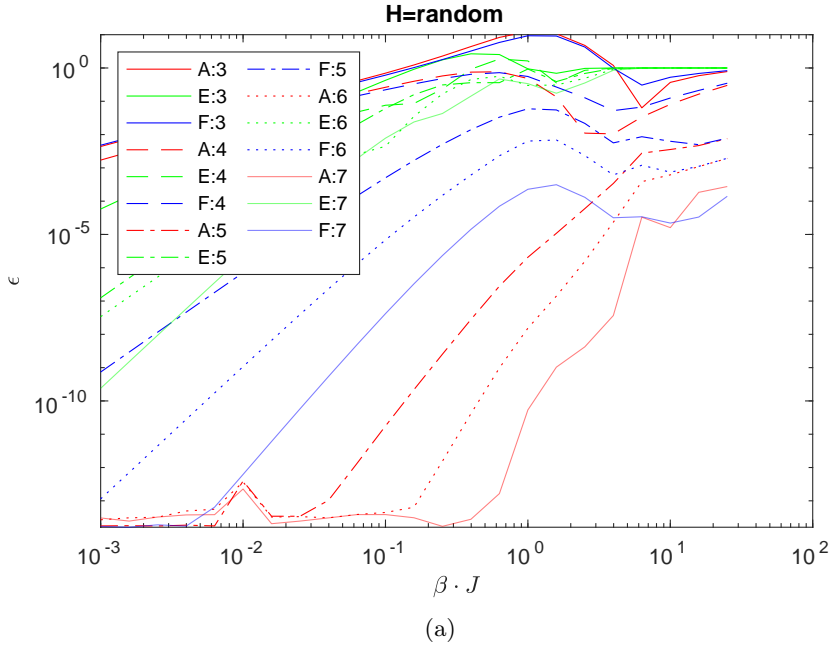


Figure 6.5: Comparison type A, E and F for 2 random generated Hamiltonians.

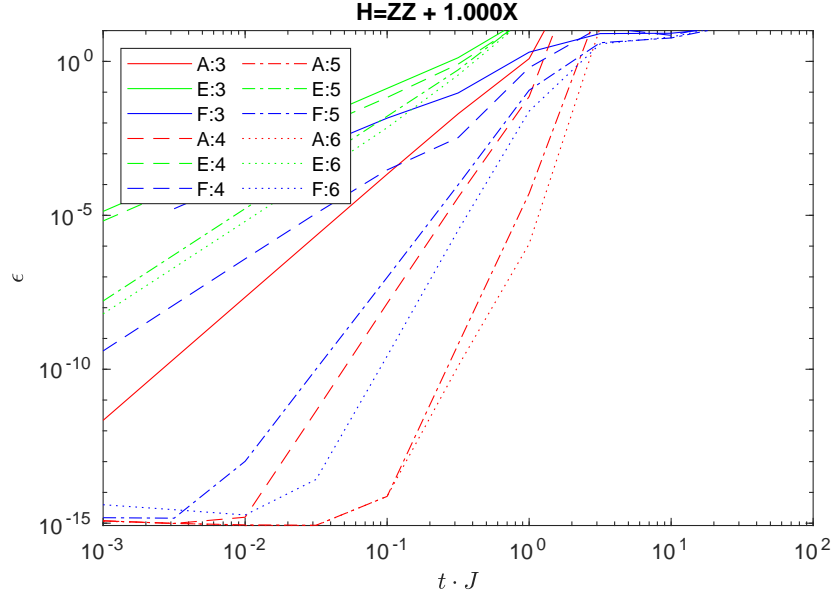


Figure 6.6: Comparison type A, E and F for transversal Ising model. The horizontal axis represents the time step, not the inverse temperature. Virtual level 3 is truncated to $\chi = 20$.

some phase transitions of the 2D transversal Ising 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 tensor network 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 explicitly constructed chain. 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 tensor network on a torus is limited to 3 by 3. This 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.

$$\rho_{i,j}^1 =$$

$$\rho_{i,j}^2 = \text{Diagram of a lattice structure with a central node labeled } i \text{ and } j. \quad (6.4)$$
$$\epsilon^\alpha = \frac{\|\rho_{exact,i,j}^\alpha - \rho_{i,j}^\alpha\|_2}{\|\rho_{exact,i,j}^\alpha\|_2} \quad \alpha \in [1, 2]. \quad (6.5)$$

There are 3 different constructions: without loops, only plaquette term (eq. (4.33)) and with loop extensions from one corner (eq. (4.37)). This is tested for both norms (see eq. (6.5)).

The plaquette term (eq. (4.33)) 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.

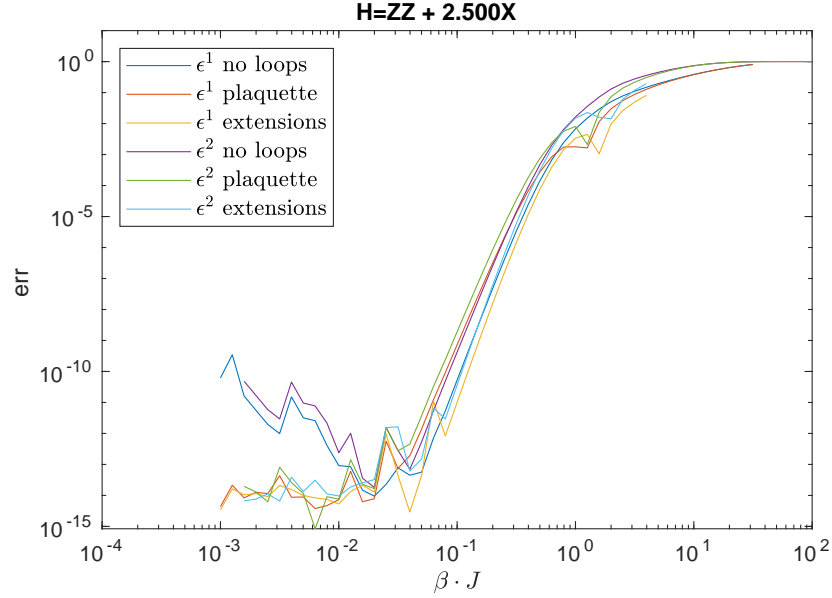


Figure 6.7: The errors ϵ^i for 2D transversal field Ising model.

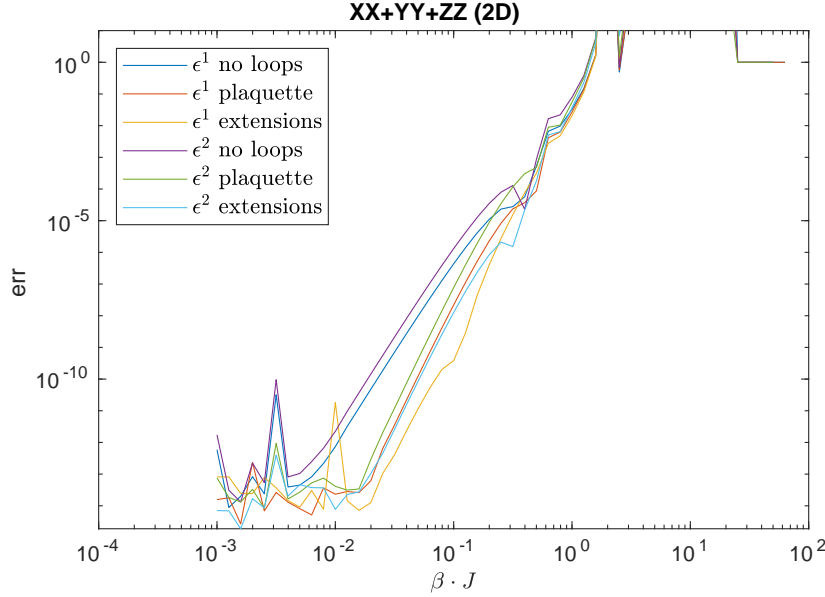
6.3.2.2 Heisenberg

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.

Figure 6.8: The errors ϵ^i for 2D Heisenberg model.

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 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 transversal Ising model

Now we have an idea how accurate the cluster expansions is, we can use it to calculate the thermodynamic properties of the transverse field Ising model. The sampling of points was explained in section 5.4.

As a reference, fig. 3.2 is shown once more

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 $T_c = \frac{2J}{k \ln(1+\sqrt{2})} \approx 2.69185J/k$. The units are normalised with $J = k = 1$. A test for the construction is to simulate

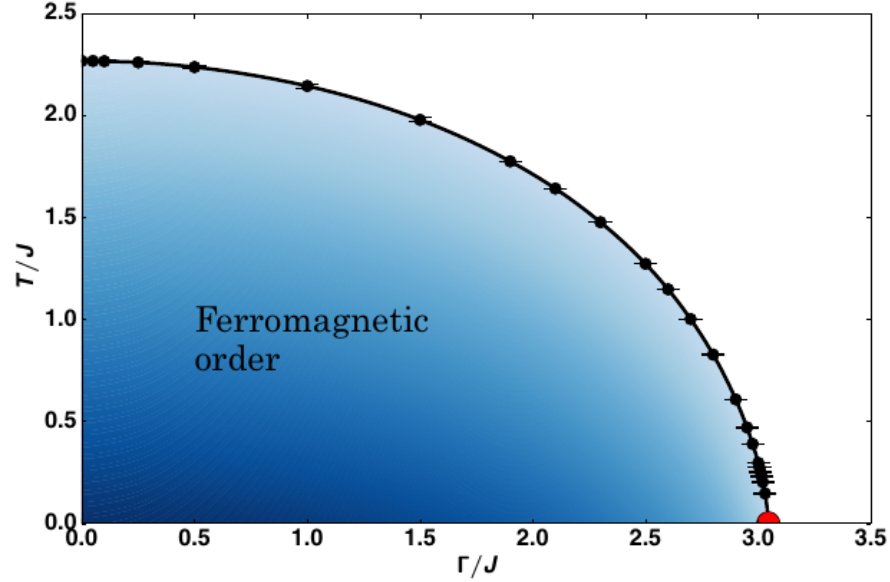


Figure 6.9: Phase diagram for 2D transversal Ising model. Figure taken from [18].

this and compare the fitted (see section 5.4.3) results.

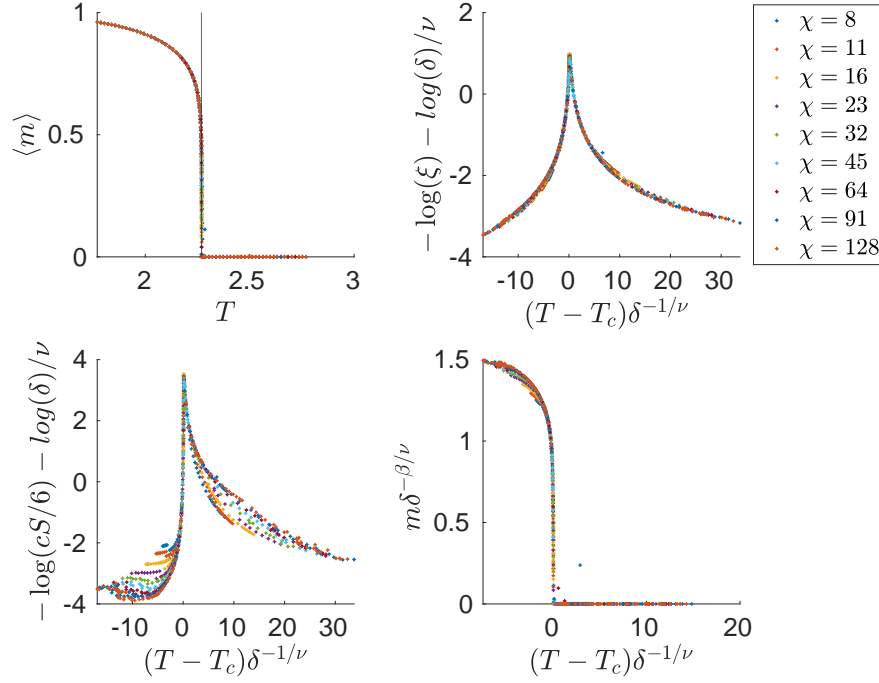
The results are shown in fig. 6.10. Each of the figures has 4 different subplots. The left upper corner shows m vs T . For low T , there is clearly a non-zero macroscopic magnetisation, while high T has a zero expectation value as expected. The other three plots show the data collapse of the finite-size 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

New figure with entropy formula fixed $6S/c$ instead of $cS/6$

Figure 6.10: Data collapse for $g = 0$ phase transition of transversal Ising Model.

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 dimensions is much higher. A much larger χ is needed for the fixed point MPS. 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 tensor network techniques provide a value of $T_c = 1.2737(2)$ [28]. With the series expansion from this paper and VUMPS, $T_c = 1.2736(6)$, indicating that this faithfully captures the physics. To put this 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

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

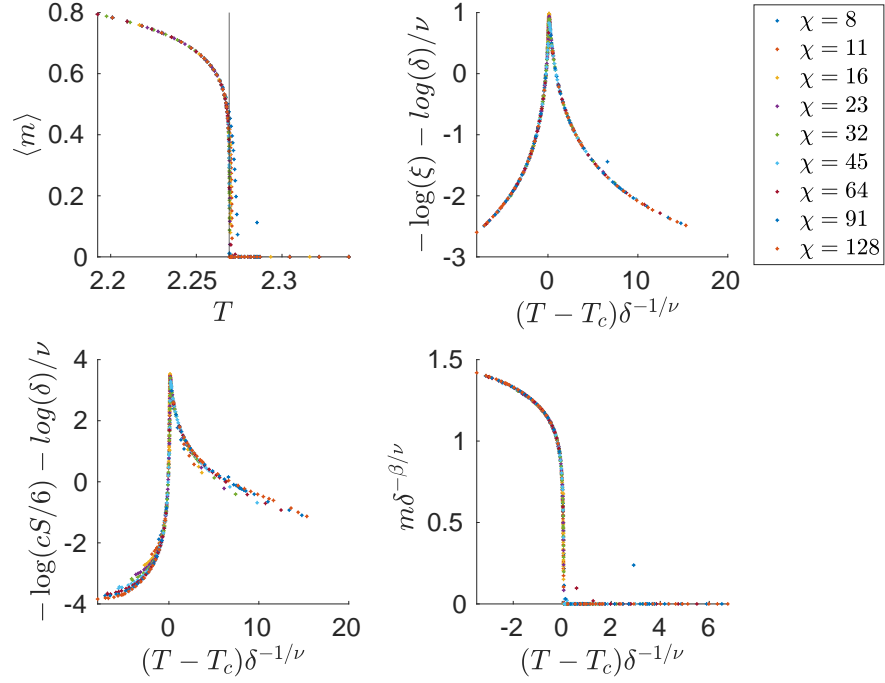


Figure 6.11: Data collapse for $g = 0$ phase transition of transversal Ising Model. Data points are taken from $T \in [T_c - 0.08, T_c + 0.08]$

is not high enough. The green curve is order 6, where virtual level is truncated to dimension 20. The others are order 5. Both include loop contributions, as opposed to the previous results. The green and blue curve both have MPS bond dimension $\chi = 8$, but predict quite different magnetisation for large g . This shows that for $g > 0.7$, order 5 is not sufficient any more.

A more accurate version is shown in fig. 6.15. Here, all the cluster expansions are of order 6 with a total bond dimension of $(D = 1 + 4 + 16 + 20)$ instead of $(D = 1 + 4 + 16)$. The results, fitted without $\chi = 8$ due to obvious finite-size effects, show a much better collapse.

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

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

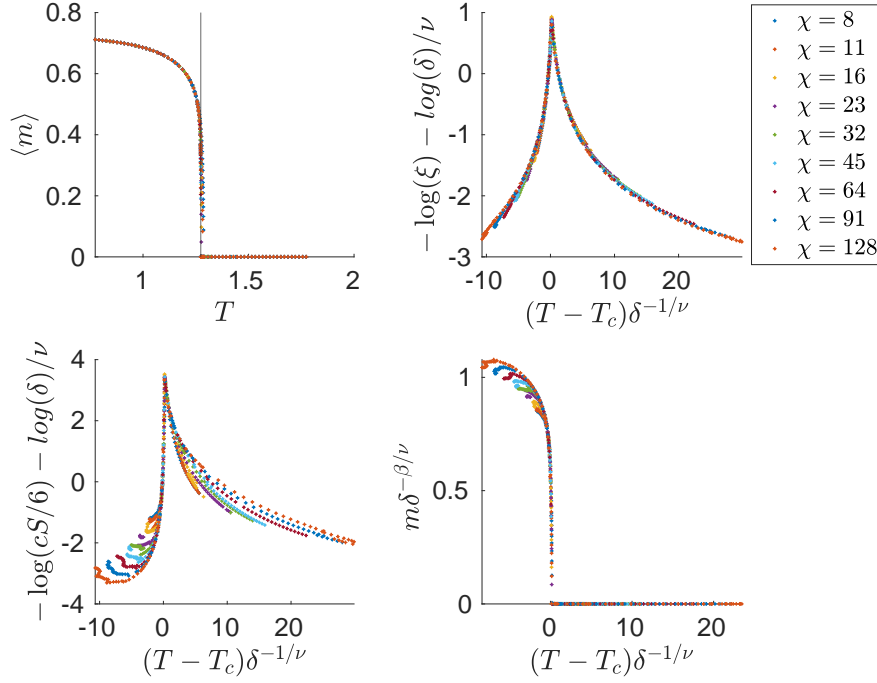


Figure 6.12: Data collapse for $g = 2.5$ phase transition of transversal Ising Model.

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 $\nu_{3D} \approx 0.62998$ [18]. 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.

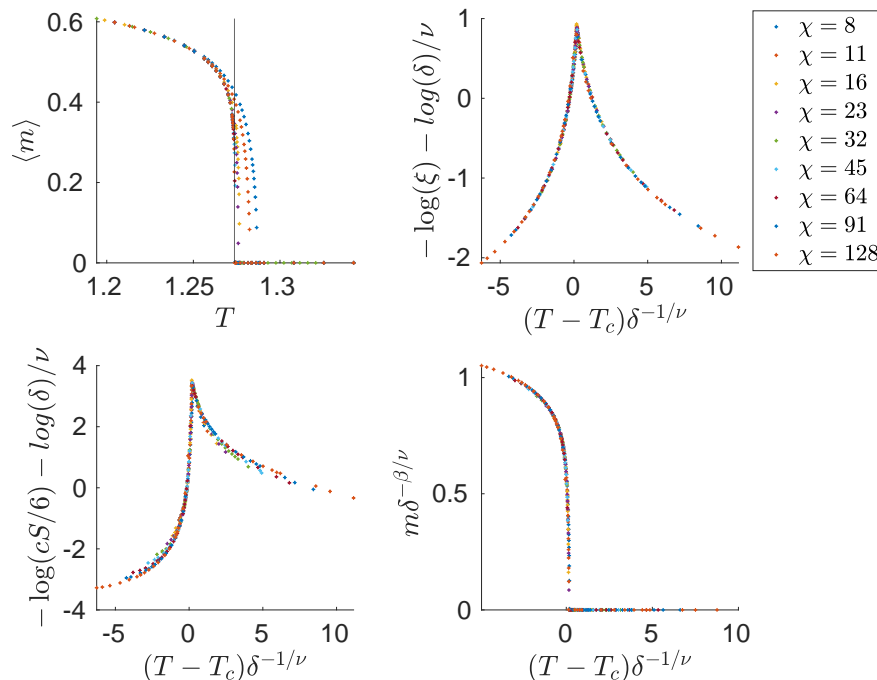


Figure 6.13: Data collapse for $g = 2.5$ phase transition of transversal Ising Model. Data points are taken from $T \in [T_c - 0.08, T_c + 0.08]$.

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. (4.30) can be added but not eq. (4.31), 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

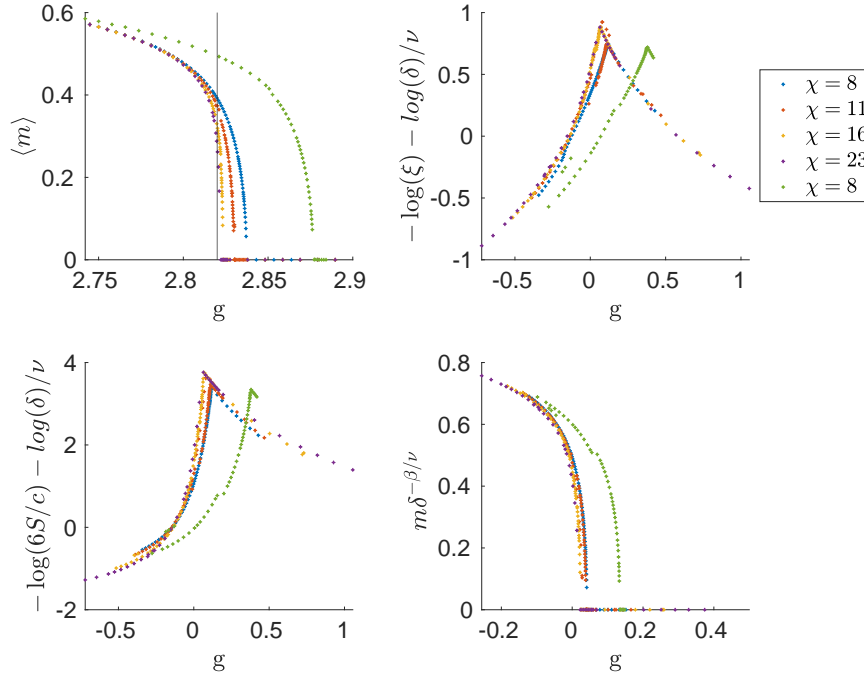


Figure 6.14: Results for $T = 0.7$ phase transition. The green points are a truncated order 6 construction, the others order 5.

Figure 1 consists of two Feynman diagrams, (a) and (b). Diagram (a) is a loop diagram with four vertices and four internal lines, all labeled α . An external line labeled a is attached to one of the vertices. Diagram (b) is a loop diagram with four vertices and four internal lines, all labeled α . Two external lines are attached to the vertices: one labeled α and the other labeled a' .

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 equation 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

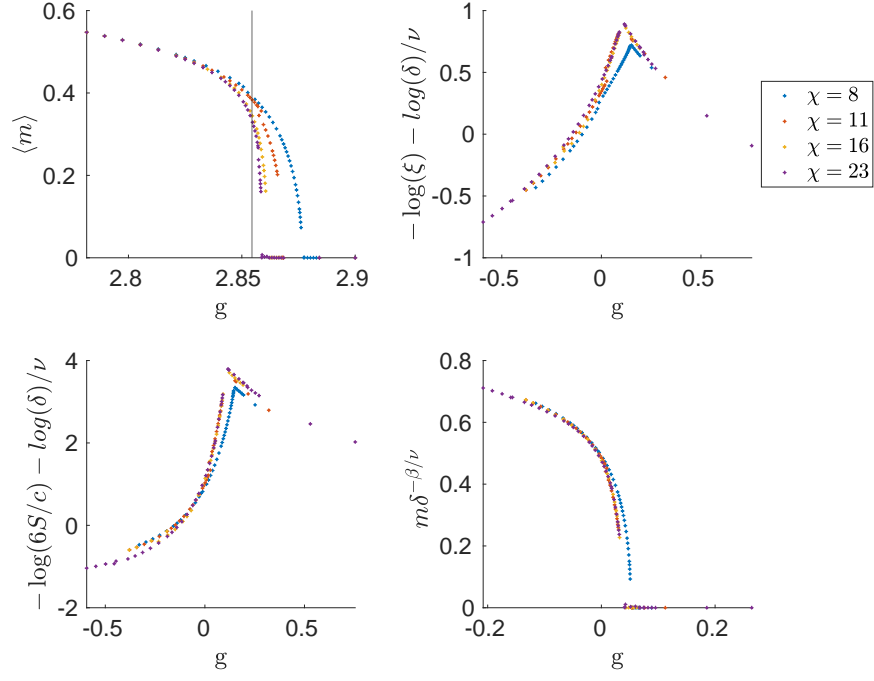


Figure 6.15: Results for $T = 0.7$ phase transition. The cluster expansion is of order 6.

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 [5] 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 [28], or even using the single site VUMPS algorithm combined with blocking:

(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 [29] 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 [5]. Both the x- and y-axis should be normalised, because depending on δ the scale changes. If scaled 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 much the prediction of T_c or g_c is improved.

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 $\exp(-\beta\hat{H})$ very well, 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 dimension. The path for determining a value for the tricritical, an important test for any method, is clear and mostly requires more precise data. With

more data points, and careful analysis of the end results, the techniques of section 6.4.5.3 could be used to calculate even more precise results. 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 in section 4.2 was 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 no 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 the $e^{\beta\hat{H}}$ 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 tensor network large enough to capture all the details of the model. The fact that linear blocks and 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 $g = 2.5$ phase transitions 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, critical behaviour is found. The path is open to calculate the quantum critical point of transversal Ising 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, tensor networks, was given. In chapter 2, a brief introduction to tensor networks was given. The focus was mainly to provide some insight in the algorithm using the graphical tensor network 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 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 Ising model in different dimensions. Finally, the need for operator exponentials and tensor network 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 transversal field Ising model is

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

7.2 Outlook

The cluster expansions have a lot of potential. In the course of one master dissertation, it is not possible to look into every possibility because a lot of groundwork need to be covered. 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 tensor networks. Crucially, all these methods come with a detailed error analysis. For instance, some methods have a proven error per step of $O\left(\frac{t^2}{N}\right)$, resulting in an error of $O\left(\frac{t^2}{N}\right)$ at time t . This can be made arbitrarily small.

Quantum critical point

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

Lattices

The whole 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, this 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 tensor network. 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 [27]). 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] 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](#).
- [3] P. Corboz, Lecture 1: tensor network states, Tech. rep.
- [4] S.-J. Ran, Tensor Network Contractions :Methods and Applications to Quantum Many-Body Systems, 2020.
URL <http://www.springer.com/series/5304>
- [5] 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](#).
- [6] M. Hauru, C. Delcamp, S. Mizera, Renormalization of tensor networks using graph independent local truncations, Tech. rep. [arXiv:1709.07460v2](#).
- [7] 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](#).
- [8] 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](#).
- [9] 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](#).
- [10] 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](#).

- [11] A. J. Beekman, L. Rademaker, J. van Wezel, An Introduction to Spontaneous Symmetry Breaking (sep 2019). [arXiv:1909.01820](#), [doi:10.21468/scipostphyslectnotes.11](#).
- [12] 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](#).
- [13] 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>
- [14] P. Ginsparg, E. Brézin, J. Zinn-Justin, *Applied Conformal Field Theory*, Tech. rep. (1988).
- [15] 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](#).
- [16] 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/10.1088/2058-7058/12/4/23>
<https://iopscience.iop.org/article/10.1088/2058-7058/12/4/23/meta>
- [17] A. Taroni, Statistical physics: 90 years of the Ising model (dec 2015). [doi:10.1038/nphys3595](#).
URL www.nature.com/naturephysics
- [18] 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](#).
- [19] T. H. Hsieh, From d-dimensional Quantum to d + 1-dimensional Classical Systems, *Student review* (2) (2015) 1–4.
- [20] 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](#).
- [21] 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](#).
- [22] 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](#).

- [23] Q. Zhao, G. Zhou, S. Xie, L. Zhang, A. Cichocki, Tensor Ring Decomposition (2016). [arXiv:1606.05535](#).
URL <http://arxiv.org/abs/1606.05535>
- [24] P. Moylan, Qr factorisation and pseudoinverse of rank- deficient matrices, <ftp://pmoylan.org/papers/90.QR>
- [25] J. Matt, N-dimensional sparse arrays.
URL <https://www.mathworks.com/matlabcentral/fileexchange/29832-n-dimensional-sparse-arrays>
- [26] C. Moler, C. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later (2003). [doi:10.1137/S00361445024180](#).
- [27] 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](#).
- [28] 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](#).
- [29] 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](#).

