University of Split
Faculty of Science

Tensor network algorithm for solving quantum physics on high-performance computing clusters

Master thesis

Rocco Barač

Split, October 2023

Temeljna dokumentacijska kartica

Sveučilište u Splitu
Prirodoslovno – matematički fakultet
Odjel za fiziku
Ruđera Boškovića 33, 21000 Split, Hrvatska

Diplomski rad

Algoritam tenzorskih mreža za rješavanje problema u kvantnoj fizici na računalnim klasterima visokih performansi

Rocco Barač

Sveučilišni diplomski studij Fizika; smjer: Astrofizika i fizika elementarnih čestica

Sažetak:

U sklopu druge kvantne revolucije koja je trenutačno u tijeku, sposobnost preciznog predviđanja ponašanja višečestičnih kvantnih sustava putem računalnih simulacija ostaje od najveće važnosti. Tenzorske mreže pružaju jedinstven način za klasičnu simulaciju takvih sustava. Ovaj rad pokazuje kako se serijski algoritam stablastih tenzorskih mreža može prilagoditi za paralelni rad na više procesorskih jezgara, omogućavajući brže manipuliranje stablastih tenzorskih mreža na računalnim klasterima visokih performansi. Pokazujemo kako ideja algoritma radi na primjeru s 2 qubita pronalazeći energiju osnovnog stanja u 1D kvantnom Isingovom modelu. Buduća poboljšanja trebala bi omogućiti manipuliranje stablastim tenzorskim mrežama temeljenim na algoritmima ovog rada te točno rješavati probleme višečestične kvantne fizike, kako u statici tako i u dinamici.

Ključne riječi: tenzori, TN, TTN, HPC, Ising, algoritam, paralelni

Rad sadrži: 43 stranice, 24 slike, 0 tablica, 41 literaturni navod. Izvornik je na engleskom jeziku.

Mentor: prof. dr. sc. Leandra Vranješ Markić

Neposredni voditelj: prof. dr. sc. Simone Montangero

Ocjenjivači: prof. dr. sc. Leandra Vranješ Markić,
izv. prof. dr. sc. Petar Stipanović,
Josipa Šćurla, mag. phys.

Rad prihvaćen: 25. 10. 2023.

Basic documentation card

University of Split
Faculty of Science
Department of Physics
Ruđera Boškovića 33, 21000 Split, Croatia

Master thesis

Tensor network algorithm for solving quantum physics on high-performance computing clusters

Rocco Barač

University graduate study programme Physics, specialization in Astrophysics and Elementary Particle Physics

Abstract:
With the second quantum revolution currently ongoing, the ability to precisely predict the behavior of many-body quantum systems via computer simulations remains of utmost importance. Tensor networks provide a unique way to simulate these systems classically. This thesis shows how a serial tree tensor network algorithm can be adapted to operate across multiple computer threads, making it possible to manipulate these networks faster on high-performance computing clusters. We show how the idea of the algorithm works on a 2-qubit example by finding the ground state energy in a 1D quantum Ising model lattice. Future improvements should make it possible to manipulate large tree tensor networks based on the algorithms of this thesis and accurately solve many-body quantum physics problems, both in statics and dynamics.

Keywords:	tensors, TN, TTN, HPC, Ising, algorithm, parallel
Thesis consists of:	43 pages, 24 figures, 0 tables, 41 references. Original language: English.
Supervisor:	Prof. Dr. Leandra Vranješ Markić
Leader:	Prof. Dr. Simone Montangero
Reviewers:	Prof. Dr. Leandra Vranješ Markić, Asoc. Prof. Dr. Petar Stipanović, Josipa Šćurla, M. Phys.
Thesis accepted:	October 25th, 2023

Thesis is deposited in the library of the Faculty of Science, University of Split.

Contents

1 Introduction **1**

2 Quantum States and Entanglement **3**

2.1 Pure and Mixed States 3

2.1.1 Density Matrix Formalism 3

2.1.2 Pure States 4

2.1.3 Mixed States 5

2.1.4 Purity 6

2.2 Multipartite Systems 6

2.2.1 Reduced Density Matrix 7

2.3 Entanglement 8

2.3.1 Separable States 8

2.3.2 Entanglement in Bipartite Systems 9

2.3.3 Entropy of Entanglement 10

2.4 Schmidt Decomposition 11

2.4.1 Low-Entanglement Approximations 11

2.5 1D Quantum Ising Model 12

3 Tensor networks **14**

3.1 Tensors and tensor networks 14

3.2 Basics of Tensor Network Notation 15

3.3 Tensor Network Links 16

3.4 Computational Complexity in Tensor Network Algorithms 17

3.5 Singular Value Decomposition 19

3.6 Matrix Product States 21

3.7 Matrix Product Operators 21

3.8 Tree Tensor Networks 22

4 Algorithm **23**

4.1 Lanczos Algorithm 24

4.2 Serial Algorithm 25

4.2.1 Isometry Center 25

4.2.2 Effective Operators 27

4.2.3 Single-Tensor Update 27

4.2.4 Tree Tensor Network Sweep 30

4.3 Parallel Algorithm 30

4.3.1 Splitting the Isometry Center to All Tensors 31

4.3.2 Effective Operators 31

4.3.3 Parallel TTN Update 33
4.4 2-Qubit Example 34

5 Summary and outlook 39

Bibliography 40

1 Introduction

With the second quantum revolution unravelling [1], we are seeing the primary focus of quantum physics research being shifted from understanding the basic principles of quantum mechanics to solving complex problems using modern quantum-based technology, and controlling quantum systems to develop new technologies which will further improve both our understanding of the world and open the door for many new advancements in industry [2]. When the first quantum revolution started at the beginning of the 20th century, our understanding of the basic concepts of fundamental physics changed drastically. We suddenly went from looking at physics through the lens of classical mechanics on all scales to an entirely new set of rules on the microscopic scale governed by the laws of the emerging theory of quantum mechanics. This shift changed our understanding of the universe from the strictly deterministic nature of classical mechanics, where we could determine the entire past and the future of a system of particles just by knowing the current state of the system, to the probabilistic nature of quantum mechanics, where there are sets of possible past and future states of a system with assigned probabilities.

Although the fundamental equations of quantum mechanics may have a simple form, many problems arise when we look at systems of particles. When trying to find the possible energy levels of a system by looking at the stationary solutions of time-independent Schrödinger equation, we soon realize that for the vast majority of Hamiltonians, there are no analytical solutions. These problems are extensively studied in quantum many-body (QMB) physics, and sometimes analytical workarounds can be found by simplifying the problem. However, that is often impossible, and we must resort to numerical methods.

When solving problems in QMB physics numerically, the main obstacle is the exponential growth of the total Hilbert space dimension with the number of elementary constituents of the system , such as quantum particles or lattice sites [3]. This makes finding the solutions for larger system sizes impossible on a classical computer. For example, in a system of N spin-1/2 particles the dimension of the Hilbert space grows as 2^N [4], thus the memory requirements to store a single quantum state, with double precision complex coefficients, would exceed 10 terabytes of memory at just N$\approx$40, which means that exact diagonalization techniques [5] are restricted only to much smaller system sizes.

While many numerical QMB methods exist [6–8], a convenient way to make many-body quantum problems solvable in a reasonable amount is based on the renormalization group (RG) paradigm, which is the idea of truncating the exponential increasing Hilbert space based on energy considerations [4]. The assumption is that a system's low-energy physics is primarily influenced by the low-energy sections of its individual components [9]. Conventional RG techniques iteratively expand the system size, discarding the high-energy sections during each step. This concept has seen further refinement in condensed matter systems through the introduction of the density matrix renormalization group (DMRG) [10]. A computationally manageable variational ansatz capable of capable of capturing the many-body properties of a system, while being as unbiased as possible, are tensor network (TN) states [3, 11]. The computational complexity of TN states is directly controlled by their entanglement, and they are particularly useful in low-entanglement systems [3]. Fortunately, there are many systems with short-range interacting Hamiltonians where the ground state entanglement is low, as they obey the area laws of entanglement, where the entanglement entropy scales with the boundary of a region rather than its volume [12].

The importance of tensor networks in the second quantum revolution lies in classically simulating low-entanglement systems which are used to build quantum hardware, e.g, based on superconducting qubits [13], trapped ions [14] or neutral atoms [15]. This thesis expands on the already existing library Quantum TEA [16], which uses tensor networks to simulate quantum systems and solve machine learning tasks. The type of tensor networks used in this thesis are tree tensor networks (TTNs) [17]. TTNs can be used in various QMB problems, which includes both dynamics and statics, so developing better algorithms is extremely important. My thesis explains the process of expanding the existing functions and methods from the Quantum TEA library to work in parallel on multiple processor threads. These new algorithms will allow for significantly faster run times when working on high-performance computing (HPC) clusters where many threads can be utilized. Modifications of serial DMRG algorithms to parallel have already been previously described and proven to work [18], so we expect that the new algorithm described in this thesis will greatly improve the Quantum TEA library, once it can be deployed on more than 2 qubits.

In this thesis, we first cover the quantum mechanics background needed to understand how low-entanglement TN approximations work, we then introduce the basics of tensor networks, followed by describing the original serial algorithm and its modification to the parallel algorithm, which is then shown to work on a 2-qubit example of the 1D quantum Ising model [19], motivating us to continue developing the algorithm.

2 Quantum States and Entanglement

The state of a quantum system is most often described by its state vector, usually represented as $|\psi\rangle$. However, representing quantum states as state vectors is only possible if we are dealing with pure states, and there is another different category of states, called mixed states, which do not share this property [20]. In this chapter's first part, we define pure and mixed states and explain their differences. To do that, we will first introduce the concept of density matrices, with which we can describe both pure and mixed states [21].

In the second subchapter, we introduce multipartite systems [22], i.e., systems composed of multiple subsystems, where we explain how their states are constructed in general and how we can view individual subsystems using the so-called reduced density matrix [22].

Arguably the most important subchapter is the one about entanglement [23], where we introduce the basics of entanglement and show some interesting properties in the case of a 2-qubit system. We also show how entropy can be quantified through the Von Neumann entanglement entropy [24] and introduce the area law of entanglement entropy [12].

What follows next is a subchapter about the Schmidt decomposition [25], which is particularly useful in the context of making accurate numerical approximations. It is closely linked to the singular value decomposition (SVD) [3], which is discussed in chapter 3.5.

The last part of the chapter explains the basics of the 1D quantum Ising model [19]. This simple model captures the quantum effects in condensed matter lattices where neighboring spin interactions occur. We also introduce its Hamiltonian, on which the ground state search of our algorithm is tested in the last chapter.

2.1 Pure and Mixed States

One way to split quantum states into two distinct categories is to split them into pure and mixed states [26]. To do that, we will first need to introduce the concept of a density matrix [4], the importance of which will become apparent by the end of this subchapter.

2.1.1 Density Matrix Formalism

A different way of describing quantum states is through the use of density matrices. In a finite-dimensional Hilbert space of dimension d, we can write any state vector $|\psi\rangle$ in the form:

$$|\psi\rangle = \sum_{i=1}^{d} \alpha_i |\Phi_i\rangle , \tag{2.1}$$

where $|\Phi_i\rangle$ are the basis vector of our Hilbert space, and α_i are their corresponding complex coefficients. To simplify matters, from now on, we assume that all states are normalized.

The density matrix operator of the state $|\psi\rangle$ is defined as the outer product of $|\psi\rangle$ and its corresponding dual space vector $\langle\psi|$, which is the definition of the projection operator to the state $|\psi\rangle$:

$$\hat{\rho} = |\psi\rangle \langle\psi| . \tag{2.2}$$

If we write each $|\Phi_i\rangle$ as a column vector where only the i-th element is equal to 1, and the rest are 0, the density matrix operator can now be expressed as a dxd matrix of the form:

$$\hat{\rho} = \begin{pmatrix} |\alpha_1|^2 & \alpha_1\alpha_2^* & \alpha_1\alpha_3^* & \dots & \alpha_1\alpha_d^* \\ \alpha_2\alpha_1^* & |\alpha_2|^2 & \alpha_2\alpha_3^* & \dots & \alpha_2\alpha_d^* \\ \alpha_3\alpha_1^* & \alpha_3\alpha_2^* & |\alpha_3|^2 & \dots & \alpha_3\alpha_d^* \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \alpha_d\alpha_1^* & \alpha_d\alpha_2^* & \alpha_d\alpha_3^* & \dots & |\alpha_d|^2 \end{pmatrix} . \tag{2.3}$$

Examining the terms of the density matrix, we see an interesting pattern. The amplitudes squared of the coefficients α_i, which correspond to the probabilities of measuring $|\psi\rangle$ to be in the state $|\Phi_i\rangle$, are on the main diagonal of our density matrix. This may seem unimportant now, but it will turn out to be one of the main motivations for using the density matrix formalism.

2.1.2 Pure States

Pure states are the first thing everyone thinks of when thinking of quantum states. They are simply states which can be described by state vectors. In finite-dimensional Hilbert spaces, we can always describe them in the form shown in Eq. (2.1). It is also immediately apparent that if we take multiple vector states from the same Hilbert space that their linear combination will also be a valid state vector from the same Hilbert space. Pure states usually appear in systems without hidden information, such as interactions with the environment.

We can now express fundamental quantum mechanical quantities, such as the norm of a state and the expectation value of an operator, using the density matrix representation:

$$|\langle\psi|\psi\rangle|^2 = \langle\psi| \, |\psi\rangle \langle\psi| \, |\psi\rangle = \mathrm{Tr}(\hat{\rho}) = 1, \tag{2.4}$$

$$\langle\hat{O}\rangle = \langle\psi|\hat{O}|\psi\rangle = \langle\psi|\psi\rangle \langle\psi| \, \hat{O} \, |\psi\rangle = \langle\psi| \, \hat{\rho}\hat{O} \, |\psi\rangle = \mathrm{Tr}(\hat{\rho}\hat{O}), \tag{2.5}$$

where Tr is the trace operator, and $\hat{O}$ is any quantum operator.

2.1.3 Mixed States

Mixed states, often referred to as statistical or classical mixtures, are states that can not be described by state vectors. They usually occur due to our incomplete knowledge of the quantum system [26]. To realize how that is possible, we must remember that the quantum state vector is not a measurable quantity. In some situations, we can only know the probability p_i of finding the system in the state $|\Phi_i\rangle$, where $|\Phi_i\rangle$ would be a state from the Hilbert space of the entire system. This level of knowledge is not enough to define the original state vector of the system, such as the one shown in Eq. (2.1). To do that, we would need to know the exact values of the complex coefficients α_i, which are unique (up to a global phase factor). The probabilities p_i correspond to $|\alpha_i|^2$, so we would know the value of each α_i only up to its own phase factor, and quantum state vectors are not invariant under relative phase changes. For a quick check you can see that inner product of the original normalized state vector and the one where a relative phase factor is introduced is not equal to 1. The systems we describe as mixed states can be in some pure state. However, due to our inability to determine which state that is or our inability to account for all degrees of freedom (e.g., because of environmental entanglement), we describe them as classical mixtures of pure states with assigned probabilities.

This is where the density matrix formalism finally shows its advantages. We can generalize the density matrix operator as the sum of the projection operator to each pure state in the mixture, $|\psi_i\rangle\langle\psi_i|$, weighted by their probability, p_i, to get:

$$\hat{\rho} = \sum_{i=1}^{N} p_i |\psi_i\rangle \langle\psi_i| , \tag{2.6}$$

where N is the number of states in the mixture. Since the sum $\sum_{j=1}^{N} p_j$ is, by definition, equal to 1, we can get back to the pure state case of $|\psi_i\rangle$, from Eq. (2.3), by setting the condition $p_i = \delta_{ij}$ for every p_i. If we now write the mixed state density operator in its matrix form, we get:

$$\hat{\rho} = \begin{pmatrix} p_1 & 0 & 0 & \dots & 0 \\ 0 & p_2 & 0 & \dots & 0 \\ 0 & 0 & p_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & p_N \end{pmatrix} . \tag{2.7}$$

We can immediately notice that the probabilities of the states are on the main diagonal, but now, all the off-diagonal elements are equal to 0. This leads us to conclude that Tr($\hat{\rho}$)=1 just as in Eq. (2.4). Similarly, the operator expectation value in this case can be written as $\langle\hat{O}\rangle = \sum_{i=1}^{N} p_i \langle\psi_i|\hat{O}|\psi_i\rangle$, which by using the density matrix operator definition from Eq. (2.6) would again lead us to $\langle\hat{O}\rangle = \mathrm{Tr}(\hat{\rho}\hat{O})$, just like in Eq. (2.5). We have to be careful with the shape of the mixed state density matrix. If we change the basis of our system, the matrix will stop being diagonal. However, the trace operator is independent to this change of basis, even though the individual components of the sum will be different. In conclusion, in the basis where the probabilities are on the diagonal, the state is mixed is all non-diagonal elements are equal to zero.

2.1.4 Purity

A simple way of measuring how close a system is to a pure state is by introducing the purity concept of purity [26]:

$$\gamma \equiv \text{Tr}(\hat{\rho}^2). \tag{2.8}$$

If we look at Eq. (2.2), we immediately see that, since $\hat{\rho}$ is a projection, we get:

$$\gamma = \text{Tr}(\hat{\rho}^2) = \text{Tr}(\hat{\rho}) = 1. \tag{2.9}$$

The closer the system's purity is to 1, the more pure it is, i.e., we have more information about the said system [26]. Suppose we wish to find the other extreme, the so-called maximally mixed state [27]. In that case, we can minimize the purity $\gamma = \sum_{i=1}^{N} p_i^2$ subject to the constraint $\sum_{i=1}^{N} p_i = 1$, using the method of Lagrange multipliers [28], to get the condition result:

$$p_i = \frac{1}{N}, \forall i \in \{1, 2, 3, \ldots, N\}. \tag{2.10}$$

To summarize, for the purity of a mixture of N states, the following condition holds:

$$\frac{1}{N} \leq \gamma \leq 1. \tag{2.11}$$

2.2 Multipartite Systems

In quantum mechanics, a composite or multipartite system is a system that consists of several smaller subsystems [22]. These smaller subsystems can be systems of particles, lattice sites, individual particles, individual degrees of freedom (like spin and angular momentum, for example), and many other things.

Each subsystem can be associated with its own Hilbert space. Let us consider N subsystems. Let $\mathcal{H}_n$ be the Hilbert space for the n-th subsystem, where $n \in \{1, 2, 3, \ldots, N\}$. The entire Hilbert space is the tensor product of all the individual Hilbert spaces:

$$\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes ... \otimes \mathcal{H}_N. \tag{2.12}$$

The total dimension d of the total Hilbert space is the product of the dimensions of all the subsystem Hilbert spaces, i.e., $d = d_1 \cdot d_2 \cdot ... \cdot d_N$, and the total wavefunction can be written as:

$$|\Psi\rangle = \sum_{\alpha_1,\alpha_2,...\alpha_N} C_{\alpha_1,\alpha_2,...\alpha_N} |\alpha_1\alpha_2...\alpha_N\rangle, \tag{2.13}$$

where $C_{\alpha_1,\alpha_2,...\alpha_N}$ are complex coefficients, each index α_n takes values of $1, 2, ..., d_n$, the state vector $|\alpha_1\alpha_2...\alpha_N\rangle$ is shorthand notation for $|\alpha_1\rangle \otimes |\alpha_2\rangle \otimes ... \otimes |\alpha_N\rangle$, and each set $|\alpha_n\rangle$ spans the entire local Hilbert space $\mathcal{H}_n$.

In many cases, the system will be composed of identical subsystems with the same dimension, in which case the total dimension of the Hilbert space is d^N, where we have assumed that d is the dimension of each local Hamiltonian. We can immediately see that numerically solving systems like these becomes increasingly more difficult as we add more particles due to the exponential growth of the total Hilbert space.

2.2.1 Reduced Density Matrix

The definition of the density matrix of a multipartite quantum system is the same as in the previous subchapter. However, we often want to look at the properties of only one of the subsystems independently of the rest of the system. For this purpose, we introduce the concept of the reduced density matrix [22]. To get the reduced density matrix for the n-th subsystem in a system of N particles, we take the partial trace over all other degrees of freedom, as shown below:

$$\begin{aligned} \hat{\rho_n} &= \text{Tr}_{m\neq n}(\hat{\rho}) \\ &= \langle \alpha_1\alpha_2...\alpha_{n-1}\alpha_{n+1}...\alpha_N | \, \hat{\rho} \, | \alpha_1\alpha_2...\alpha_{n-1}\alpha_{n+1}...\alpha_N \rangle \, . \end{aligned} \tag{2.14}$$

The reduced density matrix encapsulates all the information we can obtain about a subsystem through measurements, even if we cannot access the other subsystems. What we are actually doing when performing a partial trace is averaging out all the other degrees of freedom [22].

Moreover, the form of the reduced density matrix can give insights into the nature of correlations between the subsystems, which is covered in the next chapter.

2.3 Entanglement

Entanglement is one of the most profound and distinct concepts in quantum mechanics because of how counterintuitive it was at the time. In its simplest form, entanglement refers to quantum states of multiple subsystems where the state of one subsystem cannot be described independently of the state of the other subsystems. This inherent connection of quantum states poses both theoretical challenges, but it also offers practical advantages, particularly in the field of quantum information.

In this subchapter, we first introduce entanglement through separable states and show how the density matrix formalism can be used to determine whether a bipartite system is entangled. We then briefly cover the concept of entanglement entropy and the area law of entanglement, which is crucial for studying low entanglement systems using tensor networks.

2.3.1 Separable States

To introduce the concept of separable states [29], we first start with product states [29]. A product state $|\psi\rangle$ is a multipartite state which can be written as a tensor product of each of the N subsystem states $|\psi_i\rangle$:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes ... \otimes |\psi_N\rangle . \tag{2.15}$$

Since this is a pure state, we already know how to construct the corresponding density matrix $\hat{\rho}$ from Eq. (2.2), which, when written out in terms of subsystem density matrices $\hat{\rho}_i$, takes the following form:

$$\hat{\rho} = \hat{\rho}_1 \otimes \hat{\rho}_2 \otimes ... \otimes \hat{\rho_N}. \tag{2.16}$$

We are interested in product states because when we make measurements on one of the subsystems, we do not influence the state of any of the other subsystems, meaning that the entire system is fully non-entangled. Using product states, we can define a more general form of non-entangled states called separable states. Separable states are probabilistic mixtures (or, more accurately, convex combinations) of product states, in the same way mixed states are a convex combination of pure states. We can write a general separable state density matrix $\hat{\rho}$ as :

$$\hat{\rho} = \sum_{i=1}^{M} p_i \cdot \hat{\rho}_i, \tag{2.17}$$

where M is the number of product states in the mixture, each $\hat{\rho}_i \equiv |\psi_i\rangle \langle\psi_i|$ is the density matrix of its corresponding product state and $|\psi_i\rangle$ are now pure states. Additionally, the set of p_i must satisfy the condition $\sum_{i=1}^{M} p_i = 1$.

We must remember that there is no one-to-one correspondence between pure/mixed states and separable/non-separable states. It is only true that separable states are pure if and only if they are product states, and in general, finding out if a state is separable for many-body systems is a computationally challenging problem.

2.3.2 Entanglement in Bipartite Systems

The connection between pure and mixed states with separable and non-separable states can easily be seen in the case of a bipartite system.

Let us now first examine the case where the system is fully separable. We first define two 2-state normalized state vectors of subsystems A and B as follows:

$$\begin{aligned} |\psi\rangle_A &= \alpha\,|0\rangle_A + \beta\,|1\rangle_A \\ |\psi\rangle_B &= \gamma\,|0\rangle_B + \delta\,|1\rangle_B\,. \end{aligned} \tag{2.18}$$

When taking the tensor product, we get the product state:

$$\begin{aligned} |\psi\rangle &= |\psi\rangle_A \otimes |\psi\rangle_B \\ &= \alpha\gamma\,|00\rangle + \alpha\delta\,|01\rangle + \beta\gamma\,|10\rangle + \beta\delta\,|11\rangle\,. \end{aligned} \tag{2.19}$$

Let us now consider the reduced density matrix of subsystem A:

$$\begin{aligned} \hat{\rho}_A &= \mathrm{Tr}(|\psi\rangle\langle\psi|)_B \\ &= \langle 0|_B\,|\psi\rangle\langle\psi|\,|0\rangle_B + \langle 1|_B\,|\psi\rangle\langle\psi|\,|1\rangle_B \\ &= (\alpha\gamma\,|0\rangle + \beta\gamma\,|1\rangle)(\alpha^*\gamma^*\,\langle 0| + \beta^*\gamma^*\,\langle 1|) \\ &+ (\alpha\delta\,|0\rangle + \beta\delta\,|1\rangle)(\alpha^*\delta^*\,\langle 0| + \beta^*\delta^*\,\langle 1|) \\ &= \left(|\alpha\gamma|^2 + |\alpha\delta|^2\right)|0\rangle\langle 0| + (\alpha\gamma\beta^*\gamma^* + \alpha\delta\beta^*\delta^*)\,|0\rangle\langle 1| \\ &+ (\alpha^*\gamma^*\beta\gamma + \alpha^*\delta^*\beta\delta)\,|1\rangle\langle 0| + \left(|\beta\gamma|^2 + |\beta\delta|^2\right)|1\rangle\langle 1| \end{aligned} \quad . \tag{2.20}$$

We can now notice that $\hat{\rho}_A$ has the same form as a density matrix for pure states from Eq. (2.3), which is not a coincidence.

Let us now look at a completely different example, where the measurement of subsystem A completely determines the subsequent state of subsystem B. We can write the general state vector of one of these possible systems as:

$$|\psi\rangle = \alpha\,|00\rangle + \beta\,|11\rangle\,. \tag{2.21}$$

In this scenario, if we measure the first qubit (2-state system) to be in the state $|0\rangle$ or $|1\rangle$, we would find the second qubit in the same state, and vice versa.

If we calculate the reduced density matrix of subsystem A in the same way as in the previous example, we get:

$$\hat{\rho}_A = |\alpha|^2 |0\rangle \langle 0| + |\beta|^2 |1\rangle \langle 1| , \tag{2.22}$$

which has the shape of a mixed state density matrix, as in Eq. (2.7). If $|\alpha|^2 = |\beta|^2$=1/2 in (2.21), we would get a so-called maximally entangled state, and if we then compare Eq. (2.22) to Eq. (2.7) we would see that the reduced density matrix $\hat{\rho}_A$ would correspond to a maximally mixed state density matrix.

These results were expected. If we start with a non-entangled pure quantum system, we should expect that if we average out the effects of one subsystem to another, we should get a pure quantum state. On the other hand, if we start with an entangled system where the measurement of one subsystem uniquely determines the state of the other, averaging over one subsystem will just give us the probability of finding the second system in each state (if the basis is such to give us a diagonal reduced density matrix). This was only useful as an example because things become very complicated in large systems, which is not a convenient way to measure entanglement.

In general, the off-diagonal elements of a density matrix in a multipartite system are called quantum coherences, and they tell us how the state of a subsystem is connected to the rest of the system. They play an essential role in studying multipartite systems, but the details are not important for understanding this thesis.

2.3.3 Entropy of Entanglement

One way to measure the entanglement between two subsystems is through the introduction of Von Neumann entropy [24], which is defined as $S(\rho) = -\operatorname{Tr}(\rho \log_2 \rho)$. For two uncorrelated subsystems:

$$S(\rho_A) = -\operatorname{Tr}(\rho_A \log_2 \rho_A) = -\operatorname{Tr}(\rho_B \log_2 \rho_B) = S(\rho_B), \tag{2.23}$$

where the two subsystems have been labeled as A and B. The value of S is 1 for maximally entangled systems and 0 for non-entangled systems. We can use the Von Neumann entanglement on larger systems by grouping all of the subsystems into two subsystems and measuring the entanglement between them. Doing this to different bipartitions of a system can give us some insights about the entanglement in the entire system.

Another essential concept is the so-called area law of entanglement [12], which states that entanglement entropy scales with the size of the first region, A. In many quantum systems, particularly in lower dimensions or at low temperatures, the entanglement entropy does not scale with the volume of the region A (as one might at first expect) but rather with the area of the boundary between the two subsystems, A and B, due to the interactions with the closest neighbors being dominant. Hence, the name "area law."

More formally, for a d-dimensional system:

$$S(\rho_A) \propto L^{d-1}, \tag{2.24}$$

where L is the linear size of the region A.

2.4 Schmidt Decomposition

The Schmidt decomposition of a system divided into two subsystems, A and B, is defined as:

$$|\psi\rangle = \sum_k \lambda_k |u_k\rangle_A \otimes |v_k\rangle_B, \tag{2.25}$$

where $|u_k\rangle_A$ are orthonormal states in subsystem A, $|v_k\rangle_B$ are orthonormal states in subsystem B, and λ_k are Schmidt coefficients, which are non-negative and usually normalized in a way that $\sum_k \lambda_k^2 = 1$. The number of positive Schmidt coefficients is called the Schmidt rank, and the maximum number of the coefficients is equal to $\min(d_A, d_B)$ where d_A and d_B are the dimensions of subsystems A and B.

2.4.1 Low-Entanglement Approximations

The Schmidt decomposition is closely related to the singular value decomposition (SVD) [3], which is explained in Ch. 3.5. The reason why it is so useful lies in the fact that it quantifies how important each term in the Schmidt decomposition is by the value of their Schmidt coefficient. In tensor network algorithms, we discard the values with the small Schmidt coefficients, vastly simplifying the calculations by reducing the dimension of the total Hilbert space while keeping the results accurate. If we have a low-entangled system, many of the coefficients will be very small, making TN methods extremely powerful in those scenarios.

The extreme case of these low-entanglement approximations is the mean-field (MF) approximation [4], where we successively divide the systems into smaller subsystems while keeping only the term with the highest Schmidt decomposition. In MF approximations, the total wavefunction of a multipartite system composed of N subsystems takes the form of a product state:

$$|\Psi\rangle_{MF} = |\psi\rangle_1 \otimes |\psi\rangle_2 \otimes \dots \otimes |\psi\rangle_N\,, \tag{2.26}$$

where $|\psi\rangle_i \in \mathcal{H}_i$. The memory requirements to store the state, which depend on the number of indices needed to describe the total wavefunction, are now defined as $d = d_1 + d_2 + ... + d_N$. Assuming that all d_i are equal, we immediately see that the total memory requierements now grow linearly instead of exponentially. This makes the MF approximation a very powerful tool in numerically solving QMB problems with large numbers of particles on classical computers. Unfortunately, such an approximation is not justified in many cases, as the quantum correlations between the subsystems can not always be ignored.

2.5 1D Quantum Ising Model

Because the algorithm this thesis describes is used for performing a ground state search, we should first introduce the concept of local Hamiltonians [30] in general. A local Hamiltonian is an operator that describes the energy of a quantum system, where the interactions between the system's constituents are "local", i.e., particles (or other degrees of freedom, such as lattice sites) interact primarily with their immediate neighbors. In the context of many-body systems, a common example is a spin lattice where each spin interacts with its neighbors via some potential.

Mathematically, for a system of N particles, a local Hamiltonian H can be written as a sum of terms H_i that each act non-trivially only on a small number of particles:

$$H = \sum_{i=1}^{N} H_i \tag{2.27}$$

Each H_i might only act on particle i and its neighbors but is the identity operator on other distant particles.

One of the most popular examples is the 1D quantum Ising model [19], which describes a chain of spins in one dimension, where each spin can either point up or down. The spins are viewed as projections along the z-axis, and an external transverse magnetic field is applied in the x-axis, which creates a bias for one x-axis spin direction compared to the other. This system's Hamiltonian incorporates spin-spin interactions and an external magnetic field. It is defined as:

$$H = -J \sum_i \sigma_i^z \sigma_{i+1}^z - g \sum_i \sigma_i^x, \tag{2.28}$$

where J is the coupling strength between adjacent spins, σ_i^z and σ_i^x are Pauli matrices, g is the coupling coefficient between the spins and the external magnetic field, and the summation is done over all lattice sites.

The 1D Quantum Ising model is very useful for studying magnetic phase transitions. As one varies the parameter g/J, the quantum Ising model undergoes a phase transition. When g/J<1, the system is in a ferromagnetic phase where spins prefer to align. On the other hand, when g/J>1, the system is in a paramagnetic phase dominated by the transverse field. At the critical point, g/J=1, the system is highly correlated over long distances. In the ferromagnetic and paramagnetic phases, the entanglement entropy is constant, regardless of how we partition the system, which is in line with Eq. (2.24). However, at the critical point, the area law does not hold.

3 Tensor networks

In this chapter, we explain everything about tensor networks (TNs) needed for understanding the algorithm of this thesis. We start with defining tensors and basic tensor diagram notation. Moving from there, we define the basic tensor index operations and explain computational complexity in the context of tensor networks, particularly in the case of tensor contractions. We also describe the tensor decomposition used throughout the algorithm, called the singular value decomposition (SVD). We also go through how SVDs can be applied to tensors of arbitrary dimensions and how it allows us to simplify our QMB physics algorithms by neglecting low-entanglement terms.

Later, we introduce common types of tensor networks, namely the matrix product state (MPS) and its corresponding operator TN, the matrix product operator (MPO). We end the chapter with defining tree tensor networks (TTNs), the tensor network the algorithm of this thesis is based on.

3.1 Tensors and tensor networks

Tensors are mathematical objects with an arbitrary number of indices used to store information in a convenient way. A tensor with N indices is called an N-rank tensor, which, in general, has the form:

$$T_{\alpha_1,\alpha_2,\ldots,\alpha_N}, \tag{3.1}$$

where $\alpha_n \in \{1, 2, ..., d_n\}$. The total tensor can be interpreted as an object that stores $d = d_1 \cdot d_2 \cdot \ldots \cdot d_N$ scalars in an ordered way.

A tensor network is a network composed of many different tensors connected through index contractions [4]. Some of the most common examples of tensor contractions are matrix-vector multiplication and matrix-matrix multiplication:

$$\begin{aligned} \mathbf{A} \cdot \vec{v} &= \sum_j A_{ij} v_j, \\ (\mathbf{AB})_{ij} &= \sum_k A_{ik} B_{kj}. \end{aligned} \tag{3.2}$$

A general QMB wavefunction has the form as shown in Eq. (2.13). TNs can express the state amplitude $C_{\alpha_1,\alpha_2,\ldots\alpha_N}$ as a contraction over smaller tensors. The contracted indices are usually called auxiliary or virtual indices, while the other indices are called physical indices.

3.2 Basics of Tensor Network Notation

One of the main advantages of tensor networks is our ability to draw tensor network diagrams and manipulate the tensors using diagram notation. Tensors are depicted as nodes, i.e., some shape, which is usually a circle for a simple tensor and a square if it is an operator. It is common to use different colors or labels to distinguish between different tensors when it is not obvious. The lines coming out of the tensors are called links, and the number of lines represents the number of tensor indices. Some basic TN diagrams are shown in Fig. 1, namely a vector, a matrix, a 3-rank tensor, and an N-rank tensor.

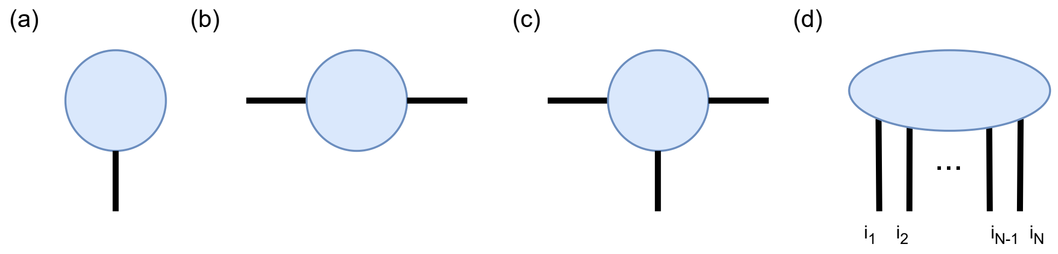

Figure 1: *Basic TN diagrams. (a) TN diagram of a vector (rank-1 tensor). (a) TN diagram of a matrix (rank-2 tensor). (a) TN diagram of a rank-3 tensor. (a) TN diagram of a rank-N tensor.*

When two tensors are connected via links, that corresponds to the contraction over the index the link represents. Examples of matrix-vector multiplication, matrix-matrix multiplication, and an example of the trace operator are shown in Fig. 2.

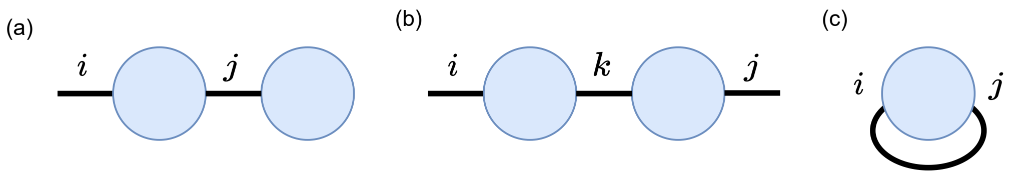

Figure 2: *TN diagrams for: (a) matrix-vector multiplication $\sum_j A_{ij}v_j$; (b) matrix-matrix multiplication $\sum_k A_{ik}B_{kj}$; (c) trace operator $\sum_{ij}\delta_{ij}A_{ij} = \sum_i A_{ii}$*

We can even show outer products by drawing two tensors next to each other, without connecting them, as shown in Fig. 3a, and dual space vectors as a ball with the link in the opposite direction like in Fig. 3b.

We can now clearly see how TN diagrams are a fantastic way of visualizing tensors and index contractions. Many operations can be visualized with TN diagrams. In the following chapters, we will see various link manipulations, tensor splitting and decompositions, but even operations like partial derivatives and many others can be represented via TN diagrams [3, 4].

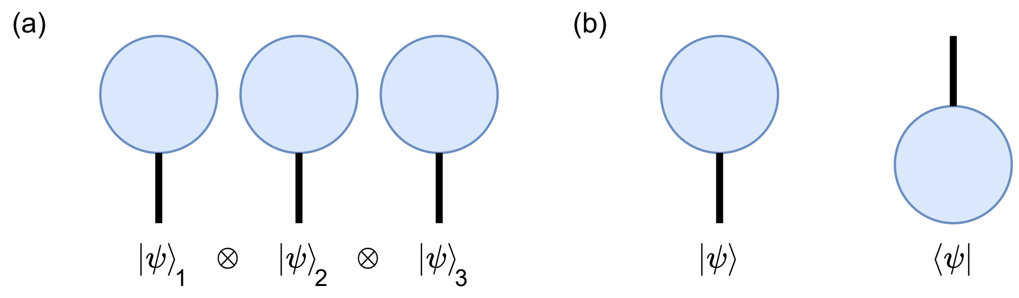
(a)
$|\psi\rangle_1 \otimes |\psi\rangle_2 \otimes |\psi\rangle_3$
(b)
$|\psi\rangle$
$\langle\psi|$

Figure 3: *TN diagrams for: (a) outer product of 3 state vectors ; (b) a ket vector with its corresponding dual/bra vector*

3.3 Tensor Network Links

The role of links is crucial in tensor networks, as they define how the tensors within are connected. Manipulating these links is crucial in understanding how to work with tensor networks, so we cover some basic link operations in this chapter.

The most basic link operation is link permutation, corresponding to reordering the tensor indices. One possible permutation of the tensor T_{abcd} would be the tensor T_{cbda}. The simplest form of link permutation is in the case of a rank-2 tensor, where performing a link permutation is called transposing a matrix: $M^T_{ab} = M_{ba}$.

Next, we have link fusion. For an arbitrary tensor $T_{\alpha_1,\alpha_2,...,\alpha_N}$, we can construct another tensor, T', by fusing the links in the following manner:

$$T'_{\alpha_1,\alpha_2,...,\alpha_i,\beta,\alpha_j,...,\alpha_N} = T_{\alpha_1,\alpha_2,...,\alpha_i,(\alpha_{i+1},...,\alpha_{j-1}),\alpha_j,...,\alpha_N}, \tag{3.3}$$

where we have contracted the indices $(\alpha_{i+1}, ..., \alpha_{j-1})$ into β. The index β now takes values in the Cartesian product of all fused links, i.e., it must cover all the combinations of all the different values each contracted indices may take. Since each index α_n can have d_n different values, β can have $d_{i+1} \cdot d_{i+2} \cdot ... \cdot d_{j-2} \cdot d_{j-1}$, which preserves the overall dimension of the tensor in terms of the number of scalars it can store. A simple example of link fusion would be converting a matrix (a tensor with two links) into a vector (a tensor with one link), where the total number of elements remains the same:

$$\begin{pmatrix} a_1 & a_2 \\ a_3 & a_4 \end{pmatrix} \rightarrow \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix}. \tag{3.4}$$

The inverse operation of link fusion is link splitting. When splitting links, we split the desired index α_n into multiple other indices, which we can label as $\beta_1, \beta_2, ..., \beta_m$. The same rules apply again, the dimension of α_n, i.e., the number of values α_n can take, has to equal the product of the dimensions of all β_i. Since the link dimension is a natural number, this limits our options with link splitting because we cannot write any natural number as a product of as many other natural numbers we want. In practice, we often do link fusing and splitting together when we want to do some operation in between, in which case we can just split the links in a way to get the tensor of the original shape. We can also sometimes add zeroes if necessary. For example, if we want to reshape a 3x1 column vector into a matrix, we could do it like this:

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ 0 \end{pmatrix} \rightarrow \begin{pmatrix} a_1 & a_2 \\ a_3 & 0 \end{pmatrix}. \tag{3.5}$$

However, this is not recommended practice and should be avoided if possible, as increasing the number of elements we work with usually increases the computational cost of running our code.

The last link operation we cover is link compression. Link compression is reducing the dimension of a tensor link by discarding a subset of the values the corresponding index can take. By reducing the dimension of a link, we also reduce the total dimension of the entire tensor, making it possible to speed up calculations. Link compression is the generalization of reducing the dimension of matrices when they are in a block matrix form, as can be seen below:

$$\begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ 0 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \tag{3.6}$$

where both the first and the second link have been reduced from dimension 3 to dimension 2.

3.4 Computational Complexity in Tensor Network Algorithms

Computational complexity refers to the amount of resources (usually time or memory) required to execute an algorithm as a function of the size of its input. In simple terms, it shows how an algorithm's runtime (or space requirements) grows when the input data grows.

There are different ways to represent computational complexity, the most common being the big O notation. It describes the upper limit of time or space required as the input size grows. For example:

- $\mathcal{O}(1)$: Constant complexity. No matter how large the input, the algorithm takes a constant amount of time (or space).
- $\mathcal{O}(m)$: Linear complexity. The time (or space) taken grows linearly with the input size.

- $\mathcal{O}(m^2)$: Quadratic complexity. The time (or space) taken grows quadratically with the input size.
- $\mathcal{O}(2^m)$: Exponential complexity. The time (or space) taken grows exponentially with the input size.

Additionally, if we do a fixed number of operations of varying complexity in a row, the total complexity is equal to that of the operation with the highest complexity (i.e. that of the function which grows the fastest as $m \to \infty$, for smaller m-s, lower order terms might still dominate). On the other hand, if we do m^N operations of complexity $\mathcal{O}(m^M)$, then the overall complexity is $\mathcal{O}(m^N m^M)$.

Since computers often deal with large inputs (in the number of bits), classical computers have difficulty solving problems with exponential complexity, often making it impossible to solve such problems. These types of problems are actually the main reason why we want to develop quantum computers. In chapter 2.2, we saw that the size of a multi-particle Hilbert space grows exponentially with the number of particles. Quantum computers use entangled particles to store and process information, making the amount of data they can store rise exponentially with the number of particles (without taking many other effects, such as noise [31], into consideration). However, we are using TNs to make classical simulations of quantum systems in this thesis, so we need to make approximations of the system where the complexity does not grow exponentially with the number of particles.

To understand how we can reduce the complexity of TN algorithms, we first need to introduce the concept of bond dimension. Bond dimension (denoted m) is one of the key parameters that dictate the computational cost of TN states. When two tensors are connected via a link, the range of the shared index represented by the links (previously also referred to as the link or index dimension), is called the bond dimension. When describing QMB systems with TNs, the bond dimension determines the amount of entanglement between the subsystems we consider. The relationship between bond dimension and entanglement will become more apparent in the following two chapters.

As an example, we will now show the computational complexity of some basic tensor contractions, as tensor contractions are often the most complex tasks when running a TN algorithm. To simplify matters, we assume all links are of dimension m. When calculating a scalar product, we have m multiplications and m-1 additions since $\vec{u} \cdot \vec{v} = \sum_{i=1}^{m} u_i v_i$, which corresponds to complexity $\mathcal{O}(m)$ [4]. When performing a matrix-vector multiplication, $(\mathbf{A} \cdot \vec{v})_i = \sum_{j=1}^{m} A_{ij} v_j$, we get the i-th element of the new vector by performing a scalar product of the i-th row of the matrix with the original vector. Thus, the number of scalar product operations of complexity $\mathcal{O}(m)$ scales with the number of matrix rows, which is again m, gives us the overall complexity of $\mathcal{O}(m^2)$. We can now clearly see that the complexity scales with the number of elements of the resulting tensor and the number of rank-1 tensor contractions. Thus, we can clearly see that the overall complexity of tensor contractions scales with the product of the dimensions of all links, where we count the shared links only once. Some simple tensor contractions and their complexities are shown in Fig. 4. In practice, some

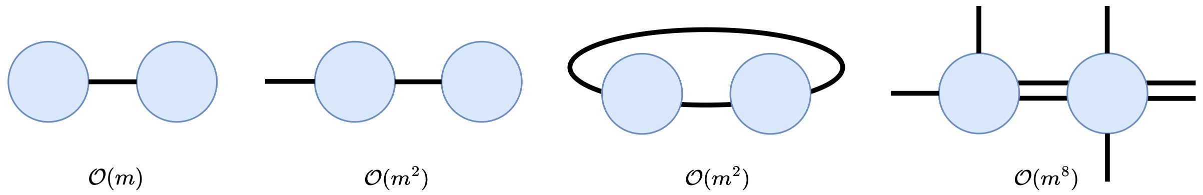

Figure 4: *Computational complexities of various tensor contractions, with all links being of dimension m. The total complexity scales with the total number of links the contracted tensors have, with the shared links only being counted only once.*

algorithms can reduce the complexity of some contractions. For example: the contraction of square matrices can be reduced from $\mathcal{O}(m^3)$ to $\mathcal{O}(m^{2.376})$ if we use optimized methods [32].

When dealing with TN algorithms, we often contract entire groups of tensors into one. The order of those contractions matters, as seen in the example in Fig. 5. There is no general rule in what order we should perform the contractions. What we usually do instead is avoid the high-complexity contractions as much as possible, as one high-complexity contraction has a much higher impact on the overall runtime of our program than multiple low-complexity contractions. We replace these high-complexity contractions with multiple lower-complexity ones wherever we can, and we do that iteratively many times to reduce the overall complexity as much as possible. There are also many other ways we could optimize a TN algorithm in addition to optimizing just the tensor contractions [3], but that is beyond the scope of this thesis.

Additionally, we could parallelize some tasks to multiple processing threads. For example: we could perform a scalar product by assigning each term in the sum to a different thread, and add them all up on a single thread in the end. Alternatively, we could do a matrix-vector multiplication by assigning a single scalar product to a single thread (each scalar product representing an element of the resulting vector), and let each thread calculate one element before recreating the final vector on a single thread in the end. In the case of an entire TN, we could also contract some tensors on one thread and others on another. When extending this logic to a large number of tensors and threads, with the order of contractions also being extremely important, we can see that there are many possible TN algorithms we could create. Thus, finding optimized TN algorithms is a challenging task, and creating just one such algorithm was enough to motivate this entire thesis while still leaving room for improvement.

3.5 Singular Value Decomposition

The singular value decomposition (SVD) is a matrix factorization method that decomposes a general mxn matrix M into three matrices, often labeled as U, S and $V^\dagger$, where $M = USV^\dagger$.

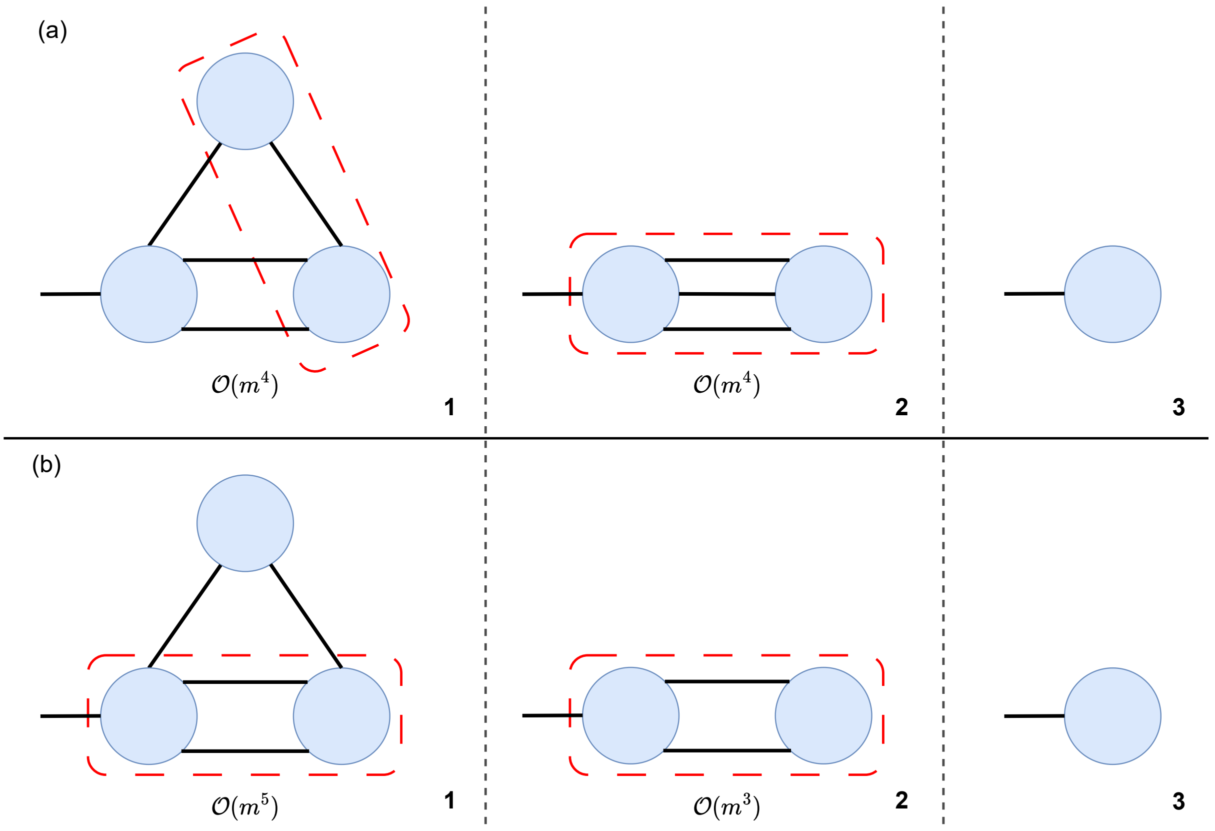

Figure 5: *Example of how the total computational complexity of contracting a group of tensors is determined by the order of contractions. All links are assumed to be of dimension m, the contracted tensors are circled with red dashed lines, and the corresponding contraction complexities are shown below the diagrams. In (a) the total complexity is $\mathcal{O}(m^4)$, while in (b) the total complexity is $\mathcal{O}(m^5)$.*

U and V are semi-unitary matrices, meaning that they satisfy the conditions $U^\dagger U = \mathbf{I_m}$ and $V^\dagger V = \mathbf{I_n}$. S is a diagonal matrix, and its diagonal elements form a set of non-negative, real-valued scalars $\{\lambda_k\}$, called singular values.

In the context of tensor networks, we perform a singular value of an arbitrary tensor by performing link operations in the following way [3]:

$$T_{(\alpha_1\ldots\alpha_i)(\alpha_{i+1}\ldots\alpha_N)} \overset{\text{fuse}}{=} T_{ab} \overset{\text{SVD}}{=} USV^\dagger = \sum_k U_{ak}\lambda_k V^\dagger_{kb} \overset{\text{split}}{=} \sum_k U_{(\alpha_1\ldots\alpha_i)k}\lambda_k V^\dagger_{k(\alpha_{i+1}\ldots\alpha_N)}. \quad (3.7)$$

Even though performing an SVD has a computational cost, it makes up for it by allowing us to perform an appropriate link compression, which is done by compressing the indices of the S matrix. The singular values are set up in a descending order on the main diagonal. Some of them might be zero, already leaving us with a block-diagonal matrix. We can go further by rounding all singular values smaller than some ϵ to zero. This truncation creates an even smaller block-diagonal matrix, which is then compressed by removing all rows and columns with only zeroes left, just as in the Eq. (3.6). Since the links of S are shared with U and V, we do the same link compression on all of these tensors, reducing the computational cost of subsequent

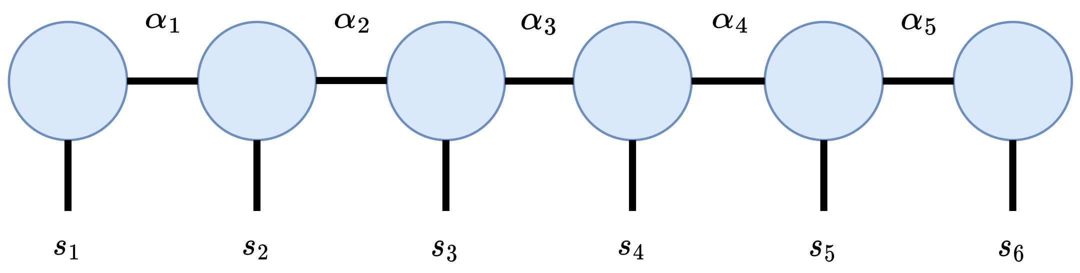

Figure 6: *Example of a 6-site MPS tensor network diagram. Auxiliary indices are labelled as α_i, and physical indices as s_i*

contractions of any of these tensors.

In the context of QMB, the singular values capture the entanglement properties of the system [3]. The physical justification for discarding the smallest singular values is that by doing so, we are removing the low-entanglement terms from our tensor. We can now also see why tensor networks are so good at describing low-entanglement systems. In low-entanglement systems, a significant number of the singular values will be very small, which allows us to decrease the computational cost of performing further operations and, in turn, allows us to simulate larger systems.

3.6 Matrix Product States

The matrix product state (MPS) is one of the most successful tensor networks. It is often used to study one-dimensional quantum systems, where it has shown remarkable success. The MPS tensor for N particles is defined as:

$$T_{s_1 s_2 \dots s_N} = \sum_{\{\alpha\}} A^{\alpha_1}_{s_1} A^{\alpha_2}_{\alpha_1 s_2} A^{\alpha_3}_{\alpha_2 s_3} \dots A^{\alpha_{N-1}}_{\alpha_{N-2} s_{N-1}} A_{\alpha_{N-1} s_N}, \tag{3.8}$$

where the physical indices are labeled with s_i, and the auxiliary indices with α_i. An example of an MPS TN diagram is shown in Fig. 6. The MPS is a generalization of the MF approximation. In the MPS case, neighboring tensors are connected via links, enabling us to capture entanglement between neighboring particles. The MF approximation is the limiting case of the MPS approximation, with all bond dimensions going to one.

3.7 Matrix Product Operators

The straightforward generalization of the MPS ansatz for operators is the matrix product operator (MPO) ansatz [33]. Because we already know how the MPS ket and bra states can be represented by looking at figures 3b and 6, we can construct an appropriate MPO. When

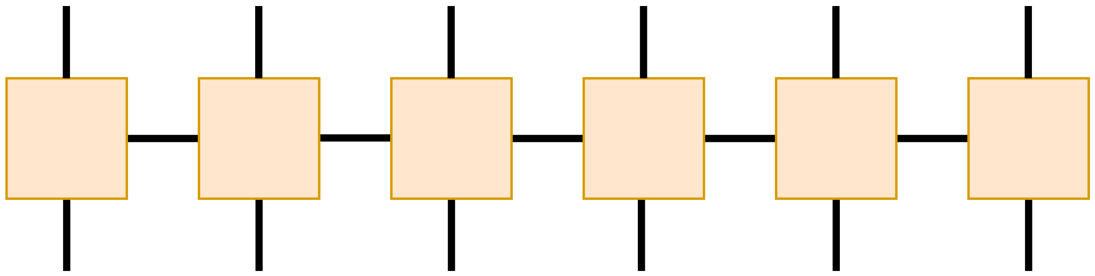

Figure 7: *Example of a 6-site MPO tensor network diagram.*

calculating the expectation value $\langle\psi|\,\hat{O}\,|\psi\rangle$, we should get a scalar. This means that the MPO ansatz should have the same shape as that of the MPS but with twice as many physical links. The MPO tensor ansatz is thus:

$$T^{s'_1 s'_2 \ldots s'_N}_{s_1 s_2 \ldots s_N} = \sum_{\{\alpha\}} A^{\alpha_1 s'_1}_{s_1} A^{\alpha_2 s'_2}_{\alpha_1 s_2} A^{\alpha_3 s'_3}_{\alpha_2 s_3} \ldots A^{\alpha_{N-1} s'_{N-1}}_{\alpha_{N-2} s_{N-1}} A^{s'_N}_{\alpha_{N-1} s_N}, \tag{3.9}$$

and the corresponding TN diagram in the case of a 6-site MPO is shown in Fig. 7, where we have used the convention of drawing operators as squares instead of circles.

3.8 Tree Tensor Networks

This subchapter introduces a type of tensor network called a binary tree tensor network, which we call a tree tensor network (TTN) throughout this thesis. This type of network is built on top of a one-dimensional lattice of physical links but has an advantage over the MPS when dealing with periodic boundary conditions [3, 34]. TTNs are composed of tensors with three links. They are arranged in a hierarchical structure where each tensor is connected to one tensor above (the parent tensor) and two tensors below (children tensors). The top two tensors are connected to each other with their parent link, and all the tensors in the bottom layer have free/physical links. When working with TTN algorithms, it is typical to label the TTN tensors by their layer and position in the layer. The first layer is usually the top layer, while the first tensor is the left-most tensor. An example of a 3-layer TTN diagram is shown in Fig. 8.

TTNs can be constructed in many different ways. We can always start from a more straightforward TN for our QMB system tensor, such as the MPS, and then through a series of tensor contractions and decompositions, link splitting and fusing, we could get a TTN shape.

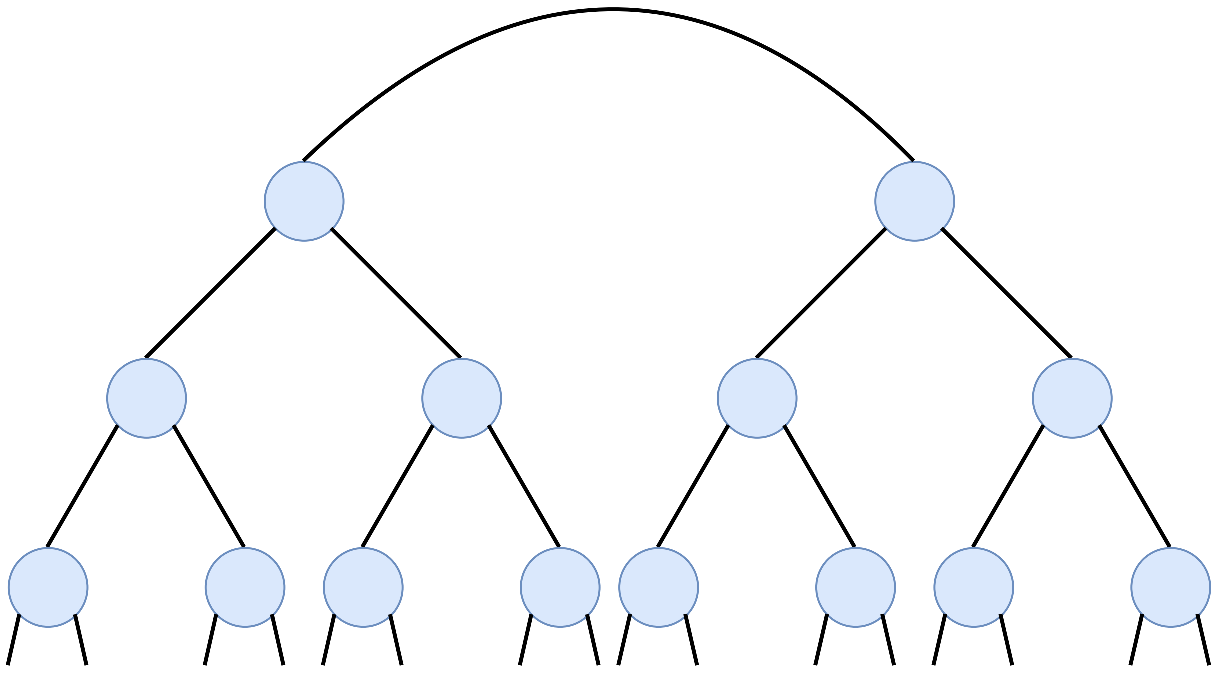

Figure 8: *Example of a 3-layer TTN diagram.*

4 Algorithm

This section is the main part of the thesis, as the algorithm is described here. We start with explaining the Lanczos algorithm, an iterative algorithm used to find the most extreme eigenvalues and their corresponding eigenvectors in the case of Hermitian operators. This algorithm is an extremely useful iterative method, and it has already shown great success in other TN algorithms used in QMB, such as the density matrix renormalization group (DMRG) [18].

We then continue describing the initial serial algorithm from the Quantum TEA [16] library. Here we introduce and describe the concept of isometry centers and effective operators, and how they are used in the context of this algorithm. The algorithm starts with a random TTN state, after which a series of single-tensor updates are performed by applying the Lanczos algorithm. These single-tensor updates aim to minimize the expectation value of the Hamiltonian, making this a variational approach (as is typical of TN algorithms).

The parallel algorithm is introduced as a modification of the serial algorithm. For the algorithm to work properly, it is necessary to somehow make every tensor in the TTN an isometry center, and we describe exactly how to accomplish it in Ch. 4.3.1. The effective operators also need to be introduced, and once that is done, we can perform single-tensor updates on every tensor simultaneously in parallel by giving each tensor its own processing thread. The usual size of TTNs makes this algorithm most suitable for high performance computing (HPC) clusters. However, even HPC clusters have a limited number of threads, and further improvements of the algorithm are possible. This can be done by grouping tensors together and applying two-tensor updates in addition to single-tensor updates.

In Ch. 4.4 we show how the principles of the algorithm work on a 2-qubit Ising model system. The tensors are updated simultaneously and they give the expected results, both for the phase diagram and the energy.

4.1 Lanczos Algorithm

The Lanczos algorithm is an iterative algorithm used to find the extreme eigenvalues and their corresponding eigenvectors of an nxn Hermitian matrix. It is particularly useful in QMB physics as we often aim to find the extreme eigenvalues and eigenvectors, and most quantum mechanics operators are Hermitian. In the context of this thesis, we use it for finding the lowest eigenvalue and the corresponding eigenvector of the Hamiltonian, giving us the ground-state energy and the ground-state vector, respectively.

The Lanczos algorithm is the simplification of the more general Arnoldi iteration, which works on non-Hermitian matrices as well. To explain the Arnoldi iteration, we first start with explaining the basic power iteration. Given an arbitrary nxn matrix A, and a random vector v, applying the matrix A to the vector v iteratively, and normalizing after every step, will result in getting the eigenvector corresponding to the largest eigenvalue [35]. To get the lowest eigenvector, we could simply multiply the matrix by -1. The Arnoldi iteration is a modification which uses the modified Gram-Schmidt procedure, i.e., with every new iteration vector, the components of the vector which are projections along the previous attempts are subtracted from the vector. Let v_k be vector we get from the k-th iteration, with v_0 being the initial vector, and A be the matrix whose maximum eigenvalue we wish to find. The procedure can be summed up in the following pseudocode:

```
$v_0$ = random vector
$k$ = 1

Repeat until $v_k$ converges:
    $v_k = Av_{k-1}$
    for $j$ from 1 to $k-1$:
        $h_{j,k-1} = v_j^* v_k$
        $v_k = v_k - h_{j,k-1} v_j$
    $h_{k,k-1} = ||v_k||$
    $v_k = v_k / h_{k,k-1}$
    $k = k + 1$
```

If A is Hermitian like in the Lanczos algorithm, things get simplified a lot. We can conclude that most $h_{j,k-1}$ are 0 because $h_{j,k-1} = v_j^* v_k = v_j^* A v_{k-1} = v_j^* A^* v_{k-1} = (Av_j)^* v_{k-1}$. Since all the vectors v_k are orthogonal by construction, $h_{j,k-1}$ is not equal to 0 only if j=$k-2$, greatly simplifying the algorithm.

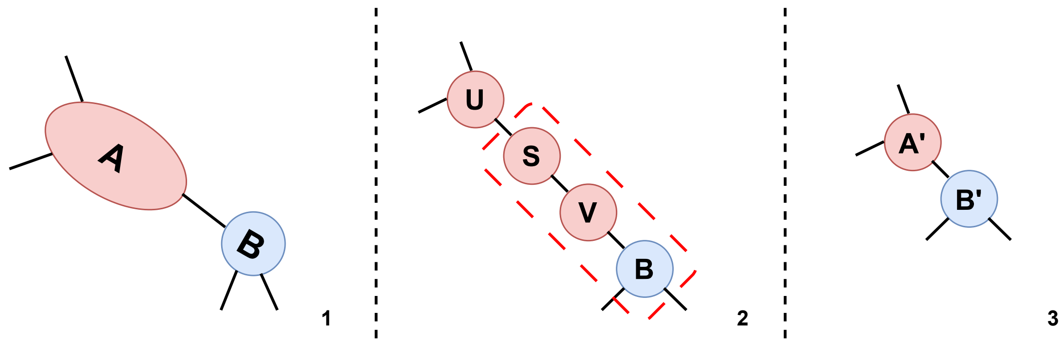

Figure 9: *Moving the isometry center from A to B via SVD. A is decomposed into U, S and V. U becomes the new A', while the contraction of S, V and B becomes the new B'.*

4.2 Serial Algorithm

To describe the parallel algorithm, we first need to describe the initial serial algorithm from the Quantum TEA [16] library. The part of the library this code is from is called Quantum TEA Leaves, and it consists of just Python code, a part of which has many functions used to manipulate TTNs. In this section, we describe the basics of the serial algorithm by covering the most important concepts needed to perform a ground-state search. The serial algorithm starts from a random state TTN and converges it to the ground-state TTN.

4.2.1 Isometry Center

To start, we first need to introduce the concept of the isometry center [36]. The isometry center of the tensor network is the tensor where most of the information about the state of the system is stored, as it is the only non-unitary tensor in the entire network. Moving the isometry center is done through a series of SVD (or QR [4]) decompositions, as shown in the example of SVDs in Fig. 9. This process is repeated for every tensor by going along the shortest path from said tensor towards the desired isometry center tensor. All the tensors along the path are also turned into unitary tensors (actually semi-unitary, but the contractions are always in the correct direction, so we just refer to them as unitary in this chapter). Once the tensor network is set up in a state with one isometry center and the other tensors are unitary, the TN is said to be isometrized. In the algorithm, the position of the tensor is labeled by its layer and the position in said layer, with both of these starting from 0. The TTN in Fig. 10 is said to be isometrized at position (1,2). The isometry center is often used in various TN algorithms (such as DMRG [37]). An isometrized tensor network has unique properties that can be exploited to reduce the program's computational costs. As an example, we can see how easy it is to normalize an isometrized TTN by taking the scalar product of the TTN with its corresponding dual space TTN, as is demonstrated in Fig. 11. Here, all the unitary tensors become identities, reducing the problem to just contracting the isometry center with itself.

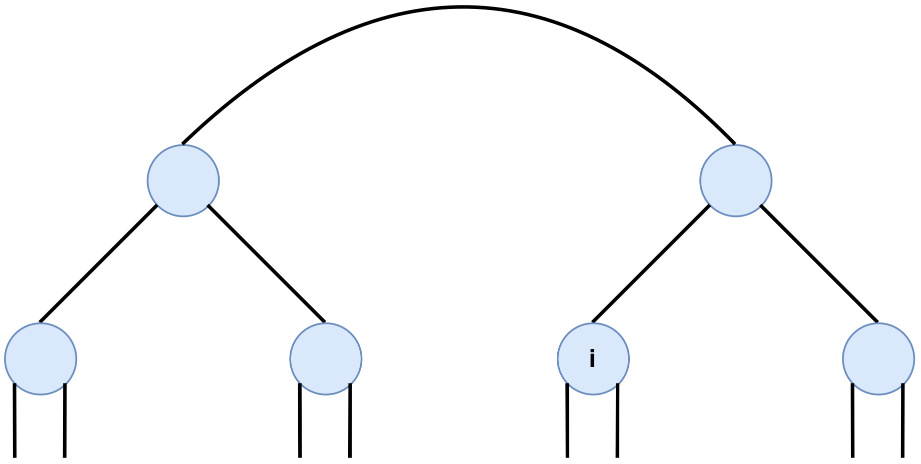

Figure 10: *An example of a 2-layer TTN with the isometry center labeled with "i" at the position (1,2)*

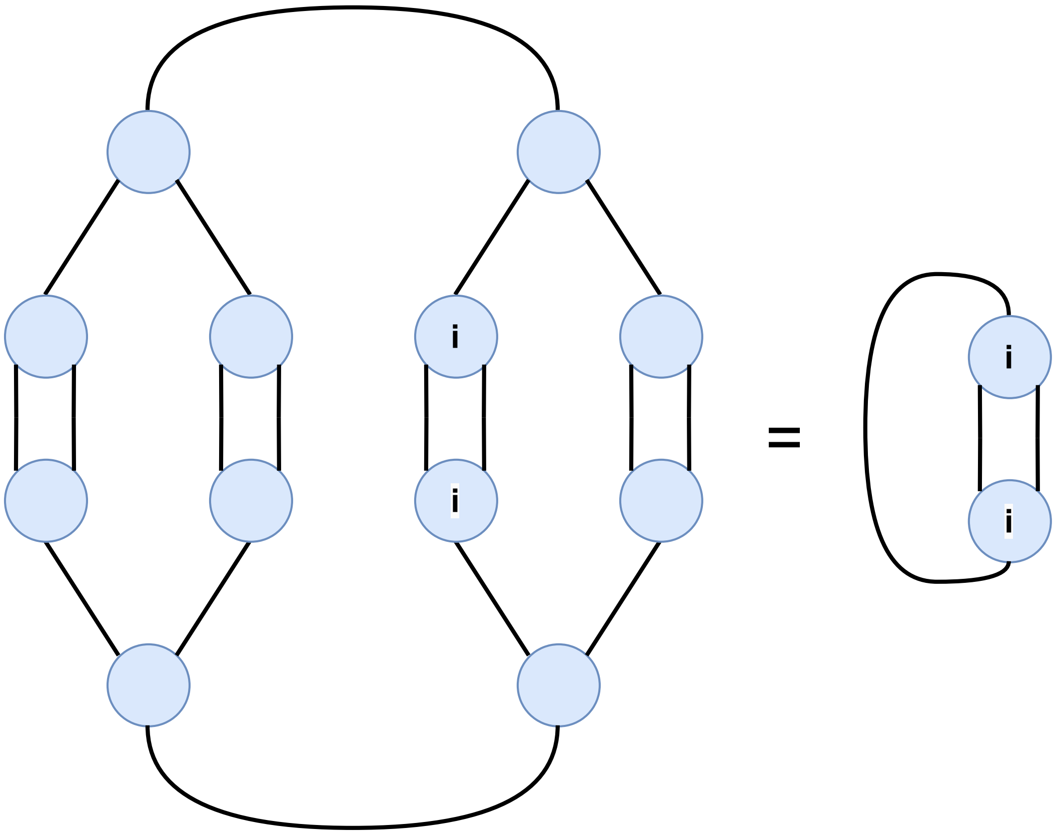

Figure 11: *Taking the norm of an isometrized TTN reduced to just taking the norm of the isometry center, here at the position (1,3).*

4.2.2 Effective Operators

Since TN algorithms are variational algorithms, we perform a ground state search by first defining the expectation value of the Hamiltonian in terms of TNs. In this specific algorithm, the Hamiltonian for the 1D quantum Ising model is written as an MPO, introduced in 3.6. The expectation value of the Hamiltonian, which is to be minimized, is shown in an example in Fig. 12.

Throughout the algorithm, large parts of the TTN will be contracted. To avoid contracting the same tensors multiple times, effective operators are introduced. This enables us to greatly reduce the overall computational complexity of the algorithm. Before constructing the effective operators, the TTN is isometrized. Effective operators are contractions of the entire Hamiltonian expectation value TN towards the isometry center, and they are placed on the links of the TTN. An example of some of the effective operators where the TTN is isometrized towards (0,0), here labeled as the tensor 1, can be seen in Fig. 13.

The algorithm always isometrizes the TTN towards (0,0) at the beginning. This enables us to construct the effective operators from the bottom up, using the effective operators from the layer below. We iterate over tensors in each layer, constructing the effective operator on the parent link by contracting the tensor with the effective operators on its children links and with its dual tensor, as can be seen in Fig. 14. The bottom effective operators are trivial as they represent only the Hamiltonian sites. From this point on, whenever the isometry center is moved, the effective operators are adjusted accordingly. This is done by updating the effective operator on the link connecting the original and the new isometry center. The update is done in the same way the operator was originally created, by using the updated tensor and contracting it with the children effective operators as in Fig. 14.

4.2.3 Single-Tensor Update

The algorithm modifies the entire TTN one tensor at a time. The process starts by first moving the isometry center to the tensor that is about to be updated, updating the effective operators appropriately. When performing a ground-state search, it is the eigenvalues of the Hamiltonian we wish to find. Since the TTN is isometrized, all the other tensors just change the basis of the Hamiltonian, which does not impact the eigenvalues. This means that that we can treat the effect of the effective operators on the isometry center as an effective Hamiltonian acting on a vector, with the effective Hamiltonian having the same eigenvalues as the Hamiltonian. This is exactly the situation where the Lanczos algorithm from Ch. 4.1 can be used, as we are dealing with a hermitian operator and searching for just the lowest eigenvalue. We apply one iteration of the Lanczos algorithm by contracting the effective Hamiltonian with the isometry center. Since the common Python functions which perform the Lanczos algorithm have to take a vector as

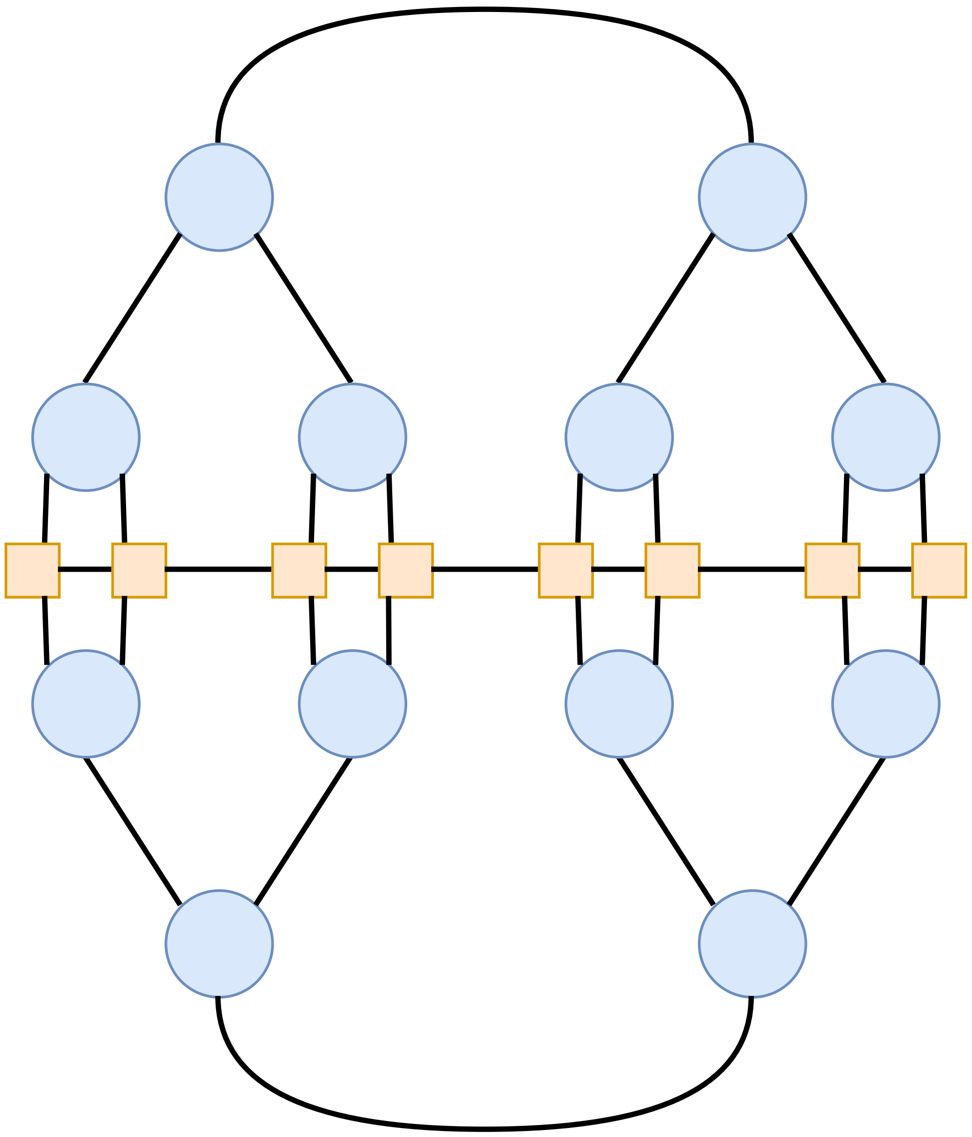

Figure 12: *Example of taking the expectation value of an operator in the MPO form, with respect to a state vector vector represented as a TTN. The MPO tensors are represented as yellow squares with links, and it represents an operator. The TTNs are TN structure with tensors represented as blue circles with links. The top TTN represents a ket vector, while the bottom TTN represents the corresponding bra vector.*

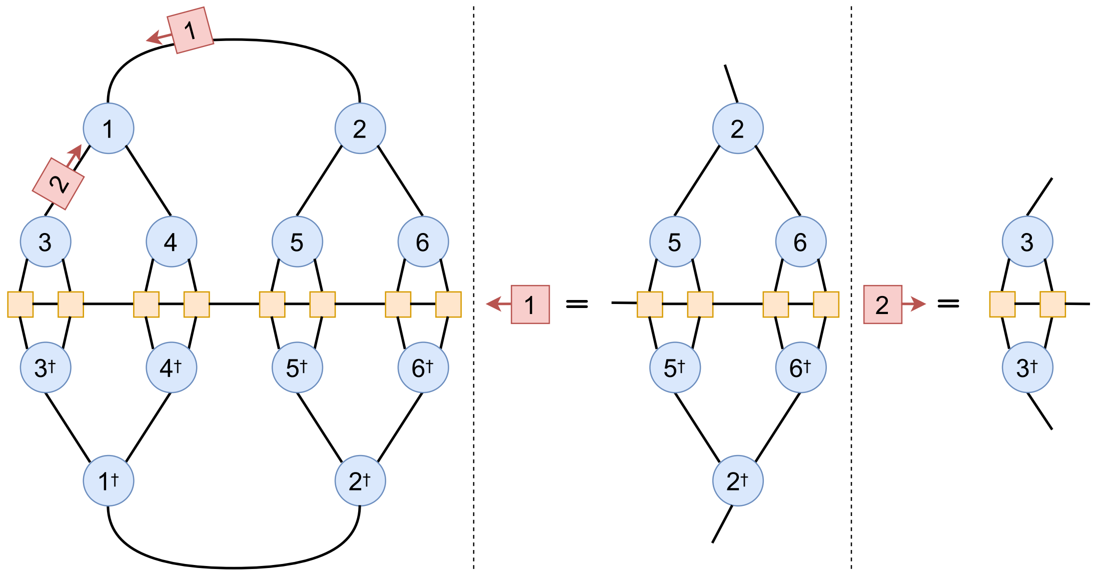

Figure 13: *The left image shows an example of effective operators going towards (0,0), which is always in the direction of the parent link with the exception of the (0,0) tensor itself. They represent contractions of the entire operator expectation value TN up to the same tensor on the adjunct TTN. The middle image shows what the effective operator 1 represents, and the right image shows what the effective operator 2 represents.*

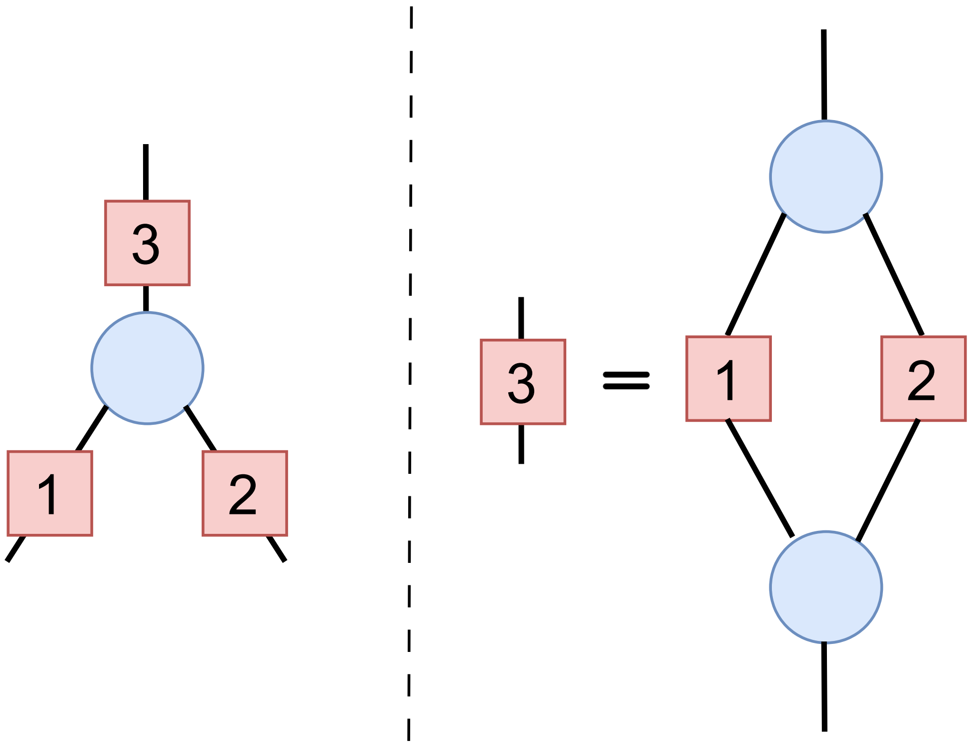

Figure 14: *Example of building an effective operator on the parent link by using the effective operators from the children links. The effective operator on the parent link is the contraction of the tensor, its adjunct, and the effective operators on the children links between them. The effective operators may have additional links, depending on the structure of the operator whose expectation value we are calculating.*

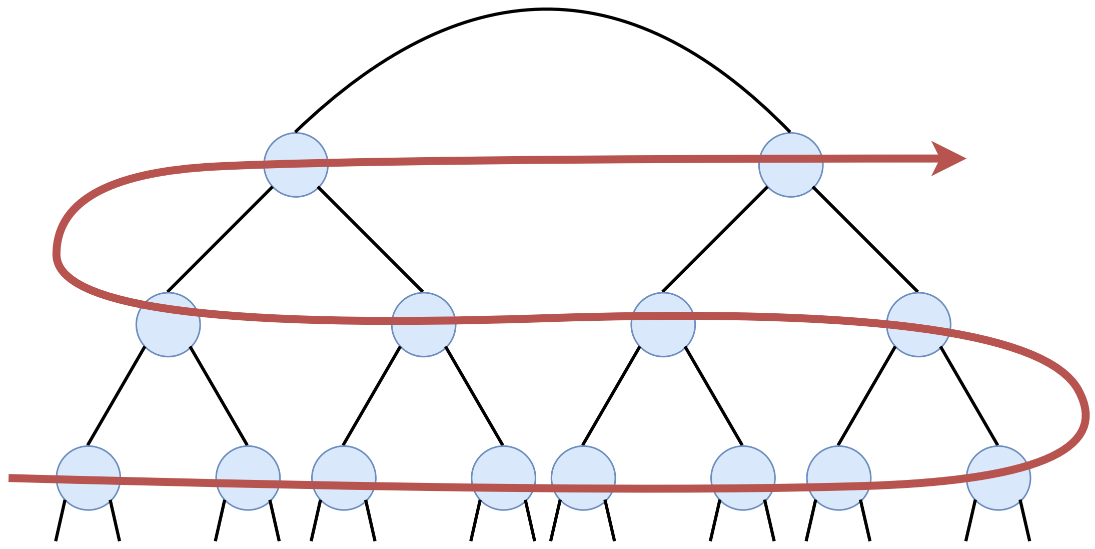

Figure 15: *Example of a serial algorithm TTN sweep performed on a 3-layer TTN. The sweep arrow tells us the order in which the tensors are optimized.*

an input and give a vector as output, we reshape the isometry center into a vector through link compression. We already know how the effective Hamiltonian acts on the isometry center in tensor format when they are contracted, so we can just reshape the resulting tensor into a vector in the same way as before. The Lanczos algorithm does not need the effective Hamiltonian in matrix form, it just needs to know how that operator acts on any vector to be able to continue the iteration.

4.2.4 Tree Tensor Network Sweep

The single-tensor update we introduced works by minimizing one tensor while keeping the rest of the TTN constant. To minimize the entire TTN, we have optimize all tensors. This is done by optimizing each tensor individually through a TTN sweep, as shown on Fig. 15. We start the sweep by optimizing the tensors in the bottom layer by going from left to right, moving the isometry center along the way. When moving to the next layer (one above), we optimize the tensors in the opposite direction, thus reducing the path for moving the isometry center, and increasing the algorithm's efficiency. When finishing a sweep across the entire TTN, we have to remember that the tensors from the beginning of the sweep were optimized with an operator that was constructed from non-optimized tensors. For this reason, the sweep is performed iteratively, until our result for the ground-state energy converges, finishing the ground-state search.

4.3 Parallel Algorithm

Now we move on to the main subject of this thesis, the parallel algorithm. The main idea is to be able to apply a single-tensor update across the entire TTN at the same time, instead of doing a sweep. To do this, various parts of the algorithm need to be changed, which is described in the following chapters.

4.3.1 Splitting the Isometry Center to All Tensors

Since we want to use the same single-tensor update as before, we need to somehow split the isometry center across all tensors. We start the same way as before, by isometrizing the TTN. Now, instead of moving the isometry center, we have to split it! We perform an SVD decomposition (Eq. (3.7)) on the isometry center as before. This time, we insert an identity matrix between the S and $V^\dagger$ matrices, and we write it as $S^{-1}S$. We then make the original isometry center a contraction of U and S matrices, and the new isometry is now a contraction of $V^\dagger$ and the original tensor, while an extra S^{-1} is left on the link between the two isometry centers. The splitting procedure is shown in Fig. 16. To make things simpler, it is best to start with the TTN isometrized at one of the top tensors, and we choose the top left tensor, at position (0,0). From here we first split the isometry center into the top right tensor. After we have our two top isometry centers, we continue splitting the isometry center into both of their respective children tensors. The process is repeated for all tensors in the layer of new isometry centers, until we reach the bottom of the TTN. If we had started with the isometry center at some random tensor in the TTN, it would still take the same amount of splitting to get the isometry center everywhere, but with this technique (going layer by layer), we can easily parallelize the isometry center splitting as well. By giving each tensor in a layer its own thread, it modifies just itself and the children tensor, and after each layer, the information on the modified tensors is exchanged across all threads.

4.3.2 Effective Operators

Having each tensor be the isometry center means that every one of them should have appropriate effective operators. If we just use the effective operators from the serial algorithm, some of them will be useless for other tensors, as they would be contractions of the TN in the opposite direction. Instead of making effective operators for each individual tensor, it is more efficient to just have two effective operators on every link in the TN, representing contractions in opposite directions. One set of effective operators represents contractions toward the (0,0) tensor, while the other set represents contractions away from the (0,0) tensor. An example with a few of these operators is shown in Fig. 17.

The operators are constructed by first constructing the effective operators towards (0,0) like in the serial algorithm. The other half is constructed when splitting the isometry center, which is easier since we can already use the first set of effective operators. The inverses of the singular value matrix are included into the effective operators. When creating the effective operators,

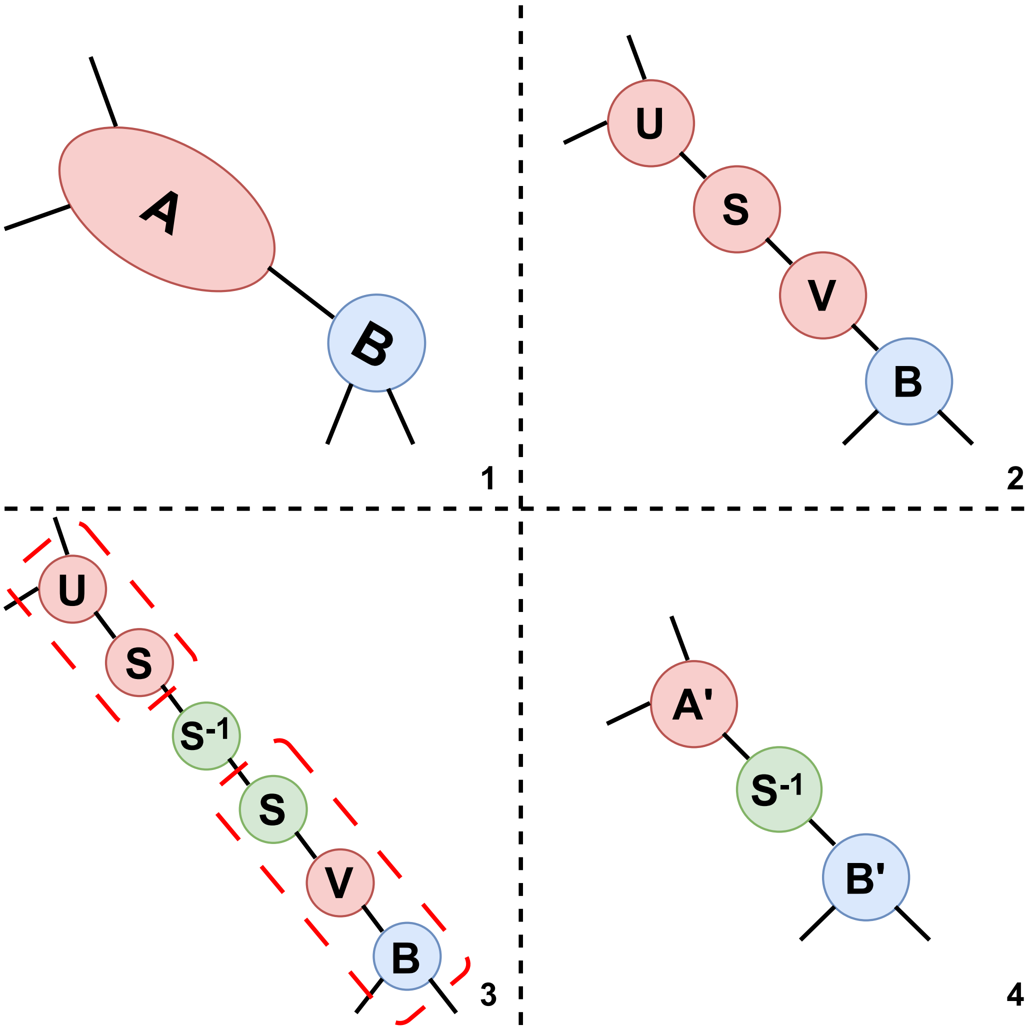

Figure 16: *Splitting the isometry center from tensor A to tensor B. After performing the SVD decomposition, an identity is inserted between S and V, and written as $S^{-1}S$. The tensors U and S are contracted into the new tensor A', while the tensors S, V and B are contracted to form the new tensor B'. An additional tensor, S^{-1}, is left on the link between the two isometry centers.*

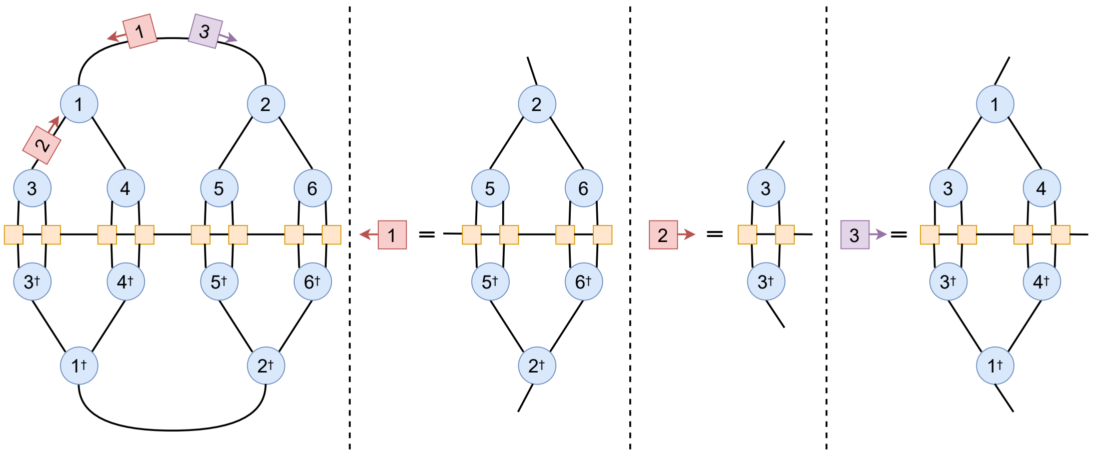

Figure 17: *Example of effective operators going towards and away from (0,0). They represent contractions of the entire TN up to the same tensor on the adjunct TTN.*

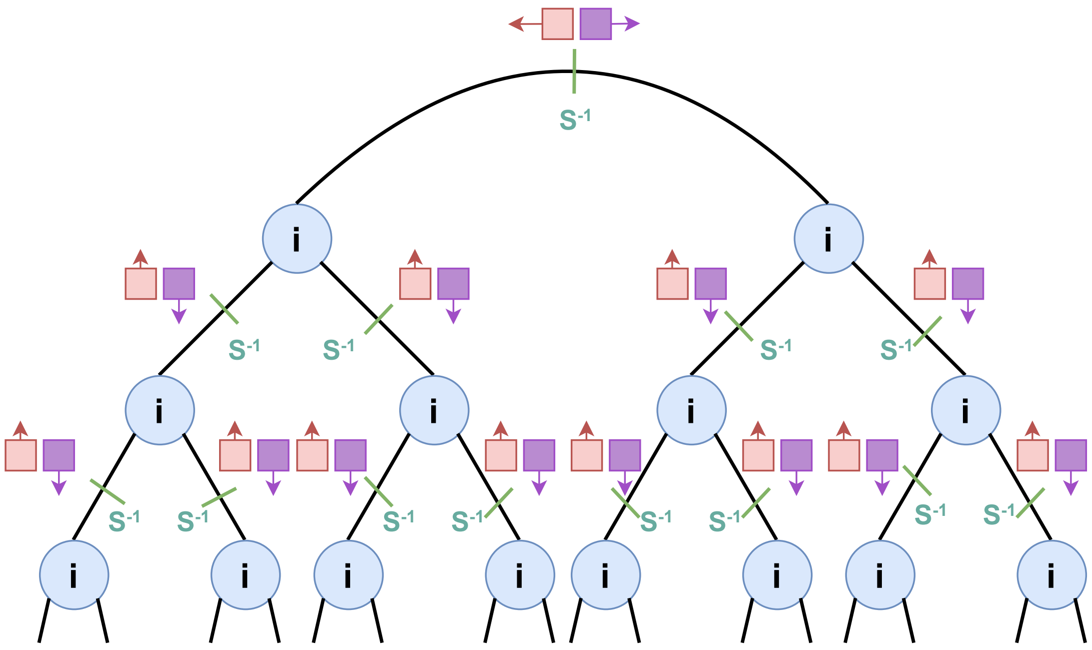

Figure 18: *Parallel TTN ready for optimization. The operators going toward (0,0) are red squares, while the ones going away are purple squares. The isometry center is split to all tensors.*

contracting S^{-1} with the operator on the desired end of the link cancels out the S matrix part of them, making them unitary again. Without this, the operator we get for the Lanczos iteration would not represent an effective Hamiltonian, making it useless for the purpose of performing a ground-state search.

4.3.3 Parallel TTN Update

By splitting the isometry center to all tensors as described in Ch. 4.3.1, along with the effective operators as described in Ch. 4.3.2, we finally get the TTN in the form from Fig. 18. When in this form, every tensor can be updated simultaneously using the single-tensor optimization from Ch. 4.2.3, each on its own processing thread. Taking the correct effective operators is easy with the way they are constructed. Every tensor takes the "incoming" (going towards (0,0)) effective operators from their children links, and the "outgoing" (going away from (0,0)) effective operator from their parent link. The exception is the (0,0) tensor, which takes the incoming effective operators from all links. The entire TTN is thus optimized by optimizing all tensors simultanously. After each iteration, the inverse singular values on the link no longer contract with their respective tensors to form a unitary tensor, so they must be updated as well. This is done by letting every tensor, except the (0,0) tensor, contract with the inverse singular values from their parent tensor. The TTN is now again isometrized at (0,0), just as in the beginning. This entire process is then iteratively repeated until the ground ground-state energies from each single-tensor update converge to the same result. Since the optimization process is the most computationally complex, parallelizing it can significantly reduce the computation

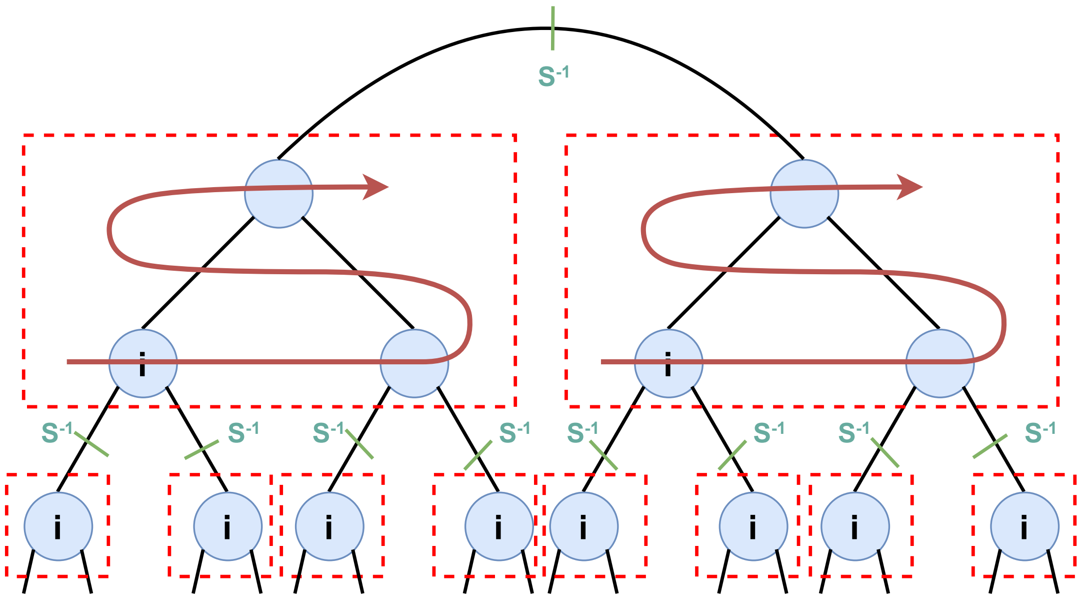

Figure 19: *An example of an algorithm which groups tensors to perform both sweeps on groups of tensors on some threads, while doing single-tensor updates on others. The dashed lines represent one processing thread.*

time. This is especially useful when running the algorithm on high-performance computing (HPC) clusters, as is almost always the case. HPC clusters have a large number of processing threads, making it able to run a large number of single-tensor optimizations in parallel, which would not be possible on personal computers.

Although HPC clusters can have a very large number of threads, we are often interested in simulating large TTNs, so we expect to run into situations where the TTN has more tensors than the HPC cluster has available threads. We could, of course, perform the algorithm by waiting for some single-tensor updates to finish, so others can take their place. However, this is not ideal and the algorithm is not intented to be used in this way. A better approach would be upgrading the algorithm in a way that the tensors are grouped together and sent to the same processing thread. One of the possibilities is to mix up the serial update of a group of tensors using sweeps in addition to applying single-tensor updates to other tensors if we have enough processing threads. An example of this is in Fig. 19. However, this is also not optimal, as there are algorithms which can perform certain two-tensor updates faster than performing two single-tensor updates in a row, as well as many other possibilities to consider.

4.4 2-Qubit Example

We now show to find the ground-state energy of a 2-qubit quantum Ising model system by using the logic of the algorithm described in the last chapter. We describe the system with an MPS TN due to it being only a 2-site system. The Hamiltonian for this system, as stated by Eq. (2.28),

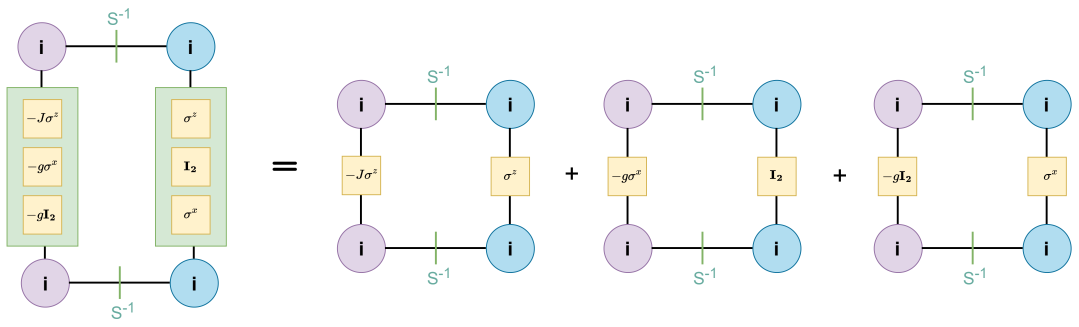

Figure 20: *The expectation value of a 2-qubit MPO Ising model Hamiltonian. The rectangles on the left TN are a shorthand notation for term by term summation. The first and the second qubit are of different colors for clarity.*

is:

$$H = -J\sigma_A^z \otimes \sigma_B^z - g\sigma_A^x \otimes \mathbf{I_2} - g\mathbf{I_2} \otimes \sigma_B^x, \tag{4.1}$$

where the first and second qubit are labelled A and B, respectively. The Hamiltonian is represented by a 4x4 matrix, making it easy to solve the problem analytically, by diagonalizing the Hamiltonian. This enables us to check whether the algorithm works.

Since state vectors of a 2-qubit system are 4x1 vectors, we start the algorithm with generating a random state vector. We want to have a 2-site MPS, and two tensors to optimize simultaneously, so we reshape the 4x1 vector into a 2x2 matrix through link fusion, as demonstrated in Eq. (3.4). An SVD is performed and the isometry center is split by making US qubit A, and SV qubit B, while having an extra S^{-1} on the link connecting them. The parts of the Hamiltonian acting on qubit A connect to its physical link, and the same goes for qubit B. Since the Hamiltonian is a sum of three terms, we use an arbitrary shorthand notation for the expectation value shown on Fig. 20. To get the usual MPO shape, we would need to increase the dimensions of all tensors through outer products with identity tensors, and contract those extra dimensions by inserting a link between the two parts of the Hamiltonian. If we contract two 3-term tensors (like the two Hamiltonian parts for each qubit), we do it by contracting them term by term, i.e., the first term of the first tensor is contracted with the first term of the second tensor etc.

There is no need for effective operators in this case example. In this example, the effective Hamiltonian operators act on each qubit as the contraction of the entire TN into the qubit of interest, with the exception of contracting its dual-space counterpart qubit. In one iteration of the algorithm, the qubits are optimized separately via Lanczos, making it possible to parallelize the process onto two threads if we want. They are then contracted into one 2x2 matrix, normalized, and the 2-qubit MPS TN is recontstructed via SVD, making it ready for the next iteration. One cycle of updating the qubits in this way is shown in Fig. 21. In each iteration, the two qubits are first updated without exchaning any information about the changes in the TN, which is part of the process that can be parallelized. Those two qubits, together with the inverse singular

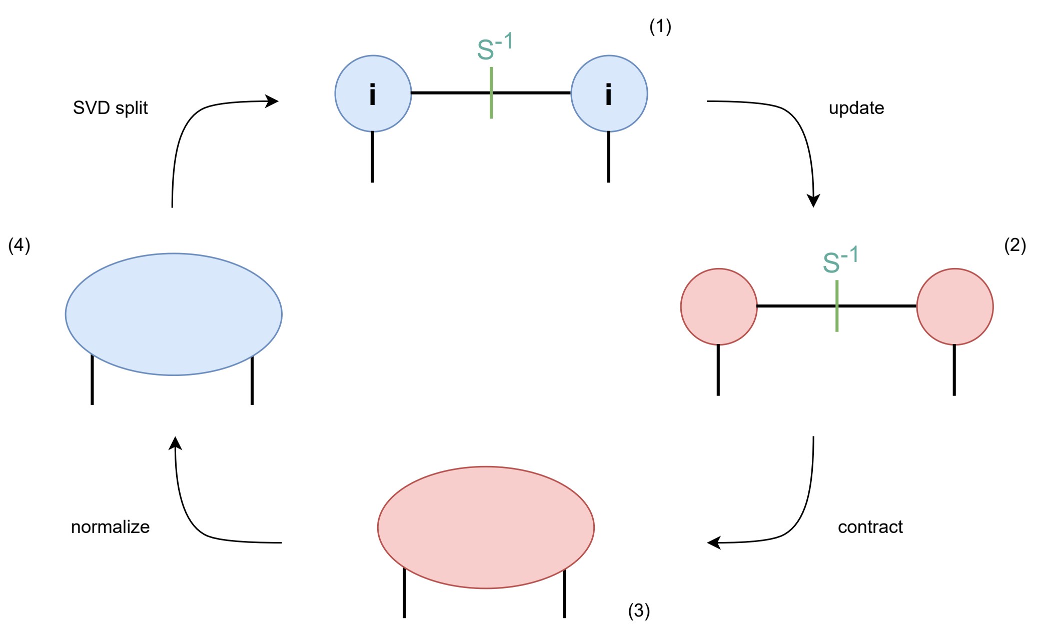

Figure 21: *One optimization iteration cycle from the 2-qubit example. (1) We start off with a 2-site MPS with the isometry center split, (2) simultaneously optimize both tensors, (3) contract the TN back into a single tensor, (4) normalize the contracted tensor, and finally get back to (1) using an SVD and splitting the isometry center.*

values, are contracted into a 2x2 matrix again. The matrix is normalized element-wise, where the sum of the squared norms of the elements must equal to 1, as they represent the probability amplitudes of a state vector. This matrix is again decomposed via SVD, and the isometry center is split, making the TN ready for the next iteration.

The results almost completely converge even after the first iteration, and after a few more, the error compared to the analytical result is almost non-existent. Plotting error graphs in this situation would not make much sense, since it would just fluctuate depending on inherent numerical errors and the initial random wavefunction. What we do instead is plot the average magnetization in the x direction as a function of g, while keeping J constant, and see if it matches our theoretical predictions from Ch. 2.5. We do the same thing for the ground-state energy. All variables are dimensionless for simplicity.

We start with the x-axis average magnetization diagram. The average magnetization is defined as the arithmetic mean of the magnetization of the two individual qubits. In regular quantum mechanics notation, the average magnetization in is written as follows:

$$\langle M \rangle = \frac{1}{2}(\langle\psi| \sigma^x \otimes \mathbf{I_2} |\psi\rangle + \langle\psi| \mathbf{I_2} \otimes \sigma^x |\psi\rangle), \tag{4.2}$$

where $|\psi\rangle$ represents the state of the total system, which is the ground state in our case. We can see the this calculation in TN diagram notation in Fig. 22.

We first look at the magnetization diagram from Fig. 23, where J=1 and g is varied from 0 to

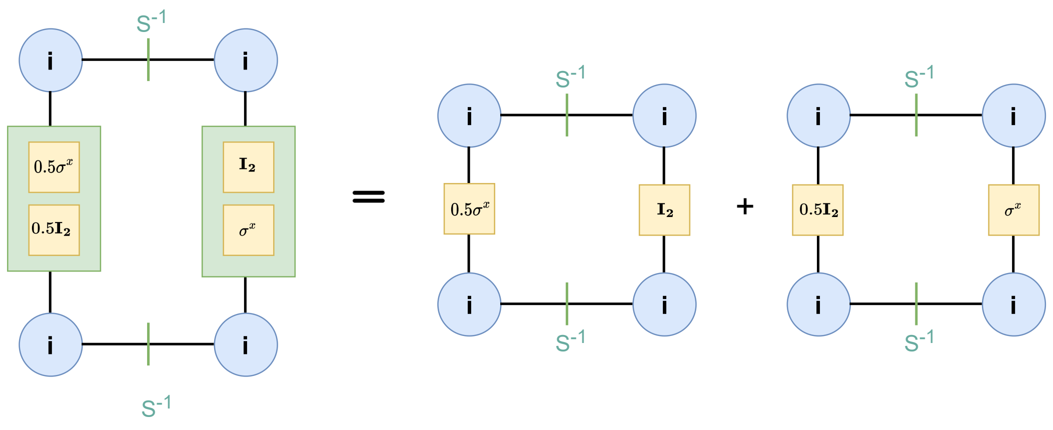

Figure 22: *Average x-direction magnetization from the 2-qubit example in TN form.*

4. The results are just as expected! The average magnetization in the x direction is 0 when g is 0. This is because there is no bias introduced by the external magnetic field to the directions of the spins. Around g=1=J, we can see a transition from the ferromagnetic phase to the paramagnetic phase as the spins align with the external magnetic field. At g=4, the magnetization is already pretty close to 1, meaning that the spins have aligned with the external magnetic field.

The ground-state energy diagram from Fig. 24 also makes sense. The spin coupling without the external magnetic field is enough to cause a bound state, resulting in the negative energy. With the increasing magnetic field strength, it becomes harder and harder to dislodge the spins, resulting in an even lower energy.

With this example, we conclude our algorithm chapter. The idea of the algorithm worked just as expected in the 2-qubit case, motivating us to continue its development and incorporate everything the Quantum TEA [16] library.

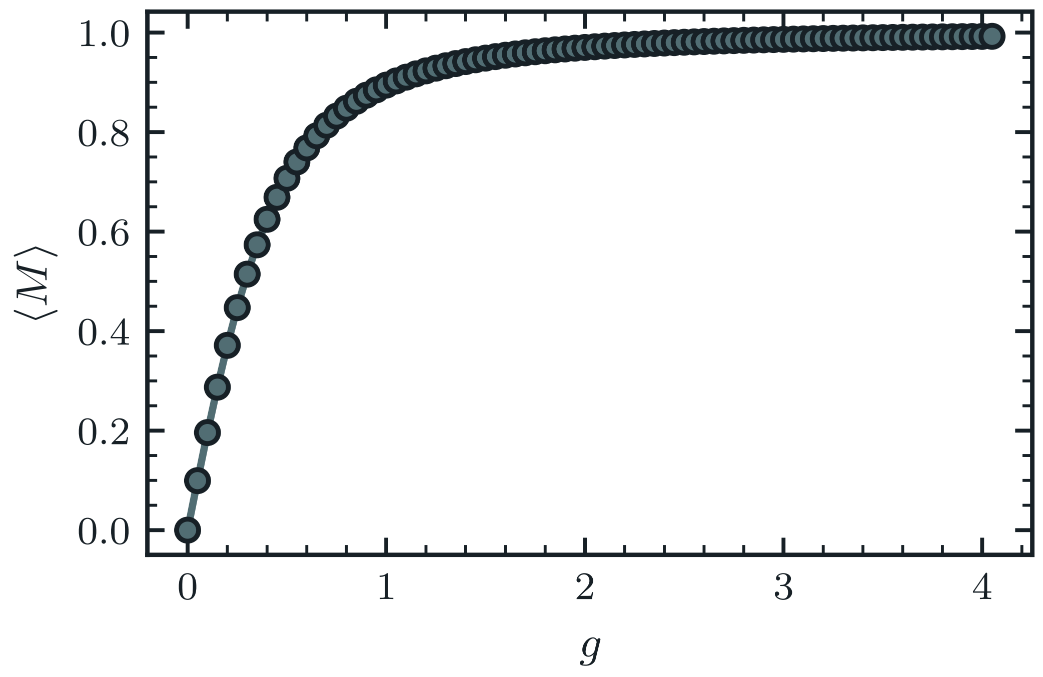

Figure 23: *Average magnetization diagram of a 2-qubit Ising model system. The magnetization in the direction of the transverse magnetic field increases as the coupling constant g increases, while J is constant and equals 1.*

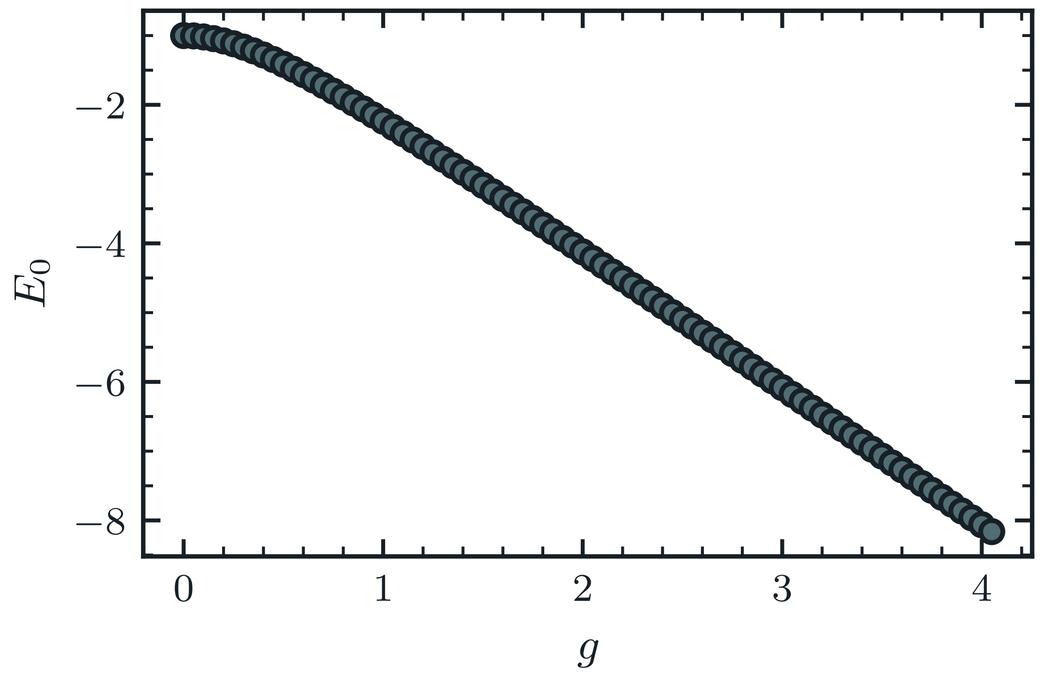

Figure 24: *Ground state energy of a 2-qubit Ising model system. The ground state energy decreases E_0 as the coupling constant g increases, while J is constant and equals 1.*

5 Summary and outlook

Throughout this thesis, we have detailed the adaptation of a serial TTN algorithm to function in parallel, aiming to decrease the computation time of QMB simulations on HPC clusters. This was done by splitting the isometry center to all tensors in the TTN, making it possible to do single-tensor optimizations on multiple processing threads simultaneously. The methods developed in this thesis work on an arbitrarily large TTN, and a successful ground state search with this kind of algorithm was shown to work in chapter 4.4, where we have used the algorithm on a 2-qubit system in a 1D quantum Ising model lattice.

We plan on further improving the algorithm and expanding it so that it can work on a limited number of computer threads, which is always the case when working with real-world computer clusters. Every improvement will be incorporated into the Quantum TEA [16] library, allowing everyone to use our algorithms in their research and further improve our understanding of QMB systems. Tensor networks have now been around for more than two decades, but are still an exciting and developing field. Their role in the second quantum revolution [38] will undoubtedly continue to be significant. With quantum simulations still being limited by the small number of qubits [39] and the always-present quantum noise [31] in current quantum hardware [40], tensor networks provide a unique way to simulate low-entanglement quantum systems on classical computers [3]. Developing new algorithms, such as the one described in this thesis, will bring us closer to large-scale quantum computers with a significant quantum advantage, which will have many uses in various fields, both in research and industry [2], such as optimization, cryptography, chemistry and drug development, machine learning [41] and many more [42].

Bibliography

[1] I. H. Deutsch. *Harnessing the Power of the Second Quantum Revolution*. In: PRX Quantum 1 (2 2020), p. 020101. doi: 10.1103/PRXQuantum.1.020101.

[2] J. P. Dowling and G. J. Milburn. *Quantum Technology: The Second Quantum Revolution*. In: Philosophical Transactions: Mathematical, Physical and Engineering Sciences 361.1809 (2003), pp. 1655–1674. ISSN: 1364503X. URL: http://www.jstor.org/stable/3559215 (visited on 10/07/2023).

[3] P. Silvi, F. Tschirsich, M. Gerster, J. Jünemann, D. Jaschke, M. Rizzi, and S. Montangero. *The Tensor Networks Anthology: Simulation techniques for many-body quantum lattice systems*. In: SciPost Phys. Lect. Notes (2019), p. 8. doi: 10.21468/SciPostPhysLectNotes.8.

[4] S. Montangero. *Introduction to Tensor Network Methods*. Springer International publishing, 2018. doi: 10.1007/978-3-030-01409-4.

[5] J. M. Zhang and R. X. Dong. *Exact diagonalization: the Bose–Hubbard model as an example*. In: European Journal of Physics 31.3 (2010), p. 591. doi: 10.1088/0143-0807/31/3/016.

[6] C. Froese Fischer. *General Hartree-Fock program*. In: Computer Physics Communications 43.3 (1987), pp. 355–365. ISSN: 0010-4655. doi: https://doi.org/10.1016/0010-4655(87)90053-1.

[7] “Index”. In: Monte Carlo Methods. John Wiley Sons, Ltd, 2008, pp. 199–203. ISBN: 9783527626212. doi: https://doi.org/10.1002/9783527626212.index. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9783527626212.index.

[8] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal. *Quantum Monte Carlo simulations of solids*. In: Rev. Mod. Phys. 73 (1 2001), pp. 33–83. doi: 10.1103/RevModPhys.73.33.

[9] K. G. Wilson. *The renormalization group: Critical phenomena and the Kondo problem*. In: Rev. Mod. Phys. 47 (4 1975), pp. 773–840. doi: 10.1103/RevModPhys.47.773.

[10] S. R. White. *Density matrix formulation for quantum renormalization groups*. In: Phys. Rev. Lett. 69 (19 1992), pp. 2863–2866. doi: 10.1103/PhysRevLett.69.2863.

[11] R. Orús. *A practical introduction to tensor networks: Matrix product states and projected entangled pair states*. In: Annals of Physics 349 (2014), pp. 117–158. ISSN: 0003-4916. doi: 10.1016/j.aop.2014.06.013.

[12] J. Eisert, M. Cramer, and M. B. Plenio. *Colloquium: Area laws for the entanglement entropy*. In: Rev. Mod. Phys. 82 (1 2010), pp. 277–306. doi: 10.1103/RevModPhys.82.277.

[13] S. Bravyi, O. Dial, J. M. Gambetta, D. Gil, and Z. Nazario. *The future of quantum computing with superconducting qubits*. In: Journal of Applied Physics 132.16 (Oct. 2022), p. 160902. ISSN: 0021-8979. doi: 10.1063/5.0082975. eprint: https://pubs.aip.org/aip/jap/article-pdf/doi/10.1063/5.0082975/16515734/160902_1_online.pdf.

[14] K. R. Brown, J. Kim, and C. Monroe. *Co-designing a scalable quantum computer with trapped atomic ions*. In: npj Quantum Information 2.1 (Nov. 2016), p. 16034. doi: 10.1038/npjqi.2016.34.

[15] K. Wintersperger, F. Dommert, T. Ehmer, A. Hoursanov, J. Klepsch, W. Mauerer, G. Reuber, T. Strohm, M. Yin, and S. Luber. *Neutral atom quantum computing hardware: performance and end-user perspective*. In: EPJ Quantum Technology 10.1 (Aug. 2023), p. 32. doi: 10.1140/epjqt/s40507-023-00190-1.

[16] Quantum TEA. https://www.quantumtea.it/. Accessed: 08/10/2023. Git download page: https://baltig.infn.it/quantum_tea/quantum_tea.

[17] Y.-Y. Shi, L.-M. Duan, and G. Vidal. *Classical simulation of quantum many-body systems with a tree tensor network*. In: Phys. Rev. A 74 (2 2006), p. 022320. doi: 10.1103/PhysRevA.74.022320.

[18] E. M. Stoudenmire and S. R. White. *Real-space parallel density matrix renormalization group*. In: Phys. Rev. B 87 (15 2013), p. 155137. doi: 10.1103/PhysRevB.87.155137.

[19] G. B. Mbeng, A. Russomanno, and G. E. Santoro. The quantum Ising chain for beginners. 2020. arXiv: 2009.09208 `[quant-ph]`.

[20] F. Rioux. *QUANTUM TUTORIALS*. Chapter: State Vectors and State Operators-Superpositions, Mixed States, and Entanglement; Accessed: 11/10/2023. LibreTexts^{TM}. URL: https://chem.libretexts.org/Bookshelves/Physical_and_Theoretical_Chemistry_Textbook_Maps/Quantum_Tutorials_(Rioux)/01%3A_Quantum_Fundamentals/1.109%3A_State_Vectors_and_State_Operators-_Superpositions_Mixed_States_and_Entanglement.

[21] A. S. Kholevo. *Statistical Structure of Quantum Theory*. Springer Berlin, 2001. ISBN: 3540420827; 9783540420828. URL: https://worldcat.org/title/318268606.

[22] I. Bengtsson and K. Zyczkowski. A brief introduction to multipartite entanglement. 2016. arXiv: 1612.07747 `[quant-ph]`.

[23] S. Perdrix. “Quantum Entanglement Analysis Based on Abstract Interpretation”. In: Static Analysis. Ed. by M. Alpuente and G. Vidal. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 270–282. ISBN: 978-3-540-69166-2. doi: 10.1007/978-3-540-69166-2_18.

[24] T Mendes-Santos, G Giudici, R Fazio, and M Dalmonte. *Measuring von Neumann entanglement entropies without wave functions*. In: New Journal of Physics 22.1 (2020), p. 013044. doi: 10.1088/1367-2630/ab6875.

[25] S. Sciara, R. L. Franco, and G. Compagno. *Universality of Schmidt Decomposition and Particle Identity*. In: Scientific Reports 7.1 (2017), p. 44675. ISSN: 2045-2322. doi: 10.1038/srep44675.

[26] J. C. A. Barata, M. Brum, V. Chabu, and R. C. da Silva. *Pure and Mixed States*. In: Brazilian Journal of Physics 51.2 (2020), pp. 244–262. doi: 10.1007/s13538-020-00808-0.

[27] K. Jeong and Y. Lim. *Purification of Gaussian maximally mixed states*. In: Physics Letters A 380.43 (2016), pp. 3607–3611. ISSN: 0375-9601. doi: https://doi.org/10.1016/j.physleta.2016.09.001.

[28] T. Shifrin. *Multivariable Mathematics: Linear Algebra, Multivariable Calculus, and Manifolds*. John Wiley Sons, 2004. ISBN: 978-0-471-52638-4.

[29] S. Halder and U. Sen. *Separability and entanglement in superpositions of quantum states*. In: Physical Review A 107.2 (2023). doi: 10.1103/physreva.107.022413.

[30] D. Nagaj. Local Hamiltonians in Quantum Computation. 2008. arXiv: 0808.2117 `[quant-ph]`.

[31] S. Resch and U. Karpuzcu. *Benchmarking Quantum Computers and the Impact of Quantum Noise*. In: ACM Computing Surveys 54 (July 2021), pp. 1–35. doi: 10.1145/3464420.

[32] D. Coppersmith and S. Winograd. *Matrix multiplication via arithmetic progressions*. In: Journal of Symbolic Computation 9.3 (1990). Computational algebraic complexity editorial, pp. 251–280. ISSN: 0747-7171. doi: https://doi.org/10.1016/S0747-7171(08)80013-2.

[33] B Pirvu, V Murg, J. I. Cirac, and F Verstraete. *Matrix product operator representations*. In: New Journal of Physics 12.2 (2010), p. 025012. doi: 10.1088/1367-2630/12/2/025012.

[34] M. Gerster, P. Silvi, M. Rizzi, R. Fazio, T. Calarco, and S. Montangero. *Unconstrained tree tensor network: An adaptive gauge picture for enhanced performance*. In: Phys. Rev. B 90 (12 2014), p. 125154. doi: 10.1103/PhysRevB.90.125154.

[35] C. Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. In: Journal of Research of the National Bureau of Standards 45.4 (1950), pp. 255–282. doi: 10.6028/jres.045.026.

[36] M. Gerster, P. Silvi, M. Rizzi, R. Fazio, T. Calarco, and S. Montangero. *Unconstrained Tree Tensor Network: An adaptive gauge picture for enhanced performance*. In: Physical Review B 90 (June 2014). doi: 10.1103/PhysRevB.90.125154.

[37] S.-H. Lin, M. P. Zaletel, and F. Pollmann. *Efficient simulation of dynamics in two-dimensional quantum spin systems with isometric tensor networks*. In: Phys. Rev. B 106 (24 2022), p. 245102. doi: 10.1103/PhysRevB.106.245102.

[38] I. H. Deutsch. *Harnessing the Power of the Second Quantum Revolution*. In: PRX Quantum 1 (2 2020), p. 020101. doi: 10.1103/PRXQuantum.1.020101.

[39] S. Patra, S. S. Jahromi, S. Singh, and R. Orus. Efficient tensor network simulation of IBM's largest quantum processors. 2023. arXiv: 2309.15642 `[quant-ph]`.

[40] H. Ma, M. Govoni, and G. Galli. *Quantum simulations of materials on near-term quantum computers*. In: npj Computational Materials 6.1 (2020), p. 85. doi: 10.1038/s41524-020-00353-z.

[41] T. Felser, M. Trenti, L. Sestini, A. Gianelle, D. Zuliani, D. Lucchesi, and S. Montangero. Quantum-inspired Machine Learning on high-energy physics data. 2021. arXiv: 2004.13747 `[stat.ML]`.

[42] V. Hassija, V. Chamola, A. Goyal, S. Kanhere, and N. Guizani. *Forthcoming Applications of Quantum Computing: Peeking into the Future*. In: IET Quantum Communication 1 (Nov. 2020), p. 1. doi: 10.1049/iet-qtc.2020.0026.