University of Amsterdam

Master Thesis 60 EC

Investigating many-body systems on quantum computers

Author:
Bjarne Bouwer

Supervisors:
Prof. dr. Michael Walter
Dr. Jonas Helsen

Second assessor:
Dr. Philippe Corboz

Abstract

Noisy intermediate-scale quantum computers are limited in their performance due to the low number of qubits and high error rate. A quantum circuit is limited in its circuit depth so that errors do not build up to unmanageable levels. However, a theoretical proposal, DMERA, based on the MERA tensor network by Kim and Swingle provides us with a method to construct a quantum circuit that is inherently noise resilient and does not require many qubits [1]. The ansatz can reliably prepare a desired quantum state by iteratively applying a quantum channel with one stable fixed point. In a recent paper, Sewell and Jordan have numerically studied DMERA by performing experiments on a quantum computer [2].

In this thesis, we will investigate DMERA in a formal and experimental manner. We reproduce and verify the results from Sewell and Jordan via experiments on a quantum computer. Furthermore, we introduce a new quantum algorithm that can extract the scaling dimensions of a critical model from measurement data using the eigenvalues of DMERA.

Acknowledgments

First, I would like to thank Michael Walter and Jonas Helsen for the great guidance and discussions, I definitely learned a lot from them. Even though this thesis was done during a weird time period where we had to meet in hybrid or completely digital fashion, they still made me feel at home. Any time I had a question they were happy to answer me.

I want to thank the QuSoft quantum information group for letting me be part of the weekly meetings and being so welcoming. In particular, I would like to thank Freek Witteveen for the help in the beginning of the project and sharing his knowledge on tensor networks.

Thanks to Amazon we were able to perform experiments on a real quantum computer. Through their Amazon Web Services we were given access to a wide selection of quantum computers.

For their continued support during my studies, I want to thank my parents, sister, and brother, for taking care of me and always showing interest in what I was working on. Lastly, I am grateful that I got to finish my master's degree with my dear friends who were a constant factor in the time I was working on my thesis and who were always there in the library with me.

Contents

Chapter 1

Introduction

In recent years, there has been a rapid development in the world of quantum computing. Many companies and organizations, such as Amazon [3], IBM [4], and Google [5], have shown interest in building a quantum computer that can outperform even the fastest and largest classical computer. The reason for this interest is the long list of potential applications, some of which can revolutionize the digital world. Among these future applications are usages in medicine development [6], financial modeling [7], and optimization problems [8]. Another notable entry is the ability to break internet encryption. When Peter Shor proposed an algorithm that allows one to relatively easily solve the math problems that guard everyone's money, personal details and more, people began to take interest in quantum computing and realize that this technology can be world changing, for the better or worse.

Unfortunately, the current generation of quantum computers are not advanced enough to realize many of these potential applications. The machines do not have enough computing power (qubits) yet and any result obtained is not reliable due to the many mistakes that occur during computation. The tasks that are run on quantum computers are limited in their size, as errors would otherwise keep building up. John Preskill coined the term Noisy Intermediate-Scale Quantum (NISQ) era for this generation of quantum computers [9]. He argues that even with 100 qubits we should not expect too many great results yet. It would take orders of magnitude more qubits to be able to efficiently run many of the quantum algorithms.

Quantum networks on quantum computers

If we have NISQ era quantum computers and they are supposedly poor in quality, what *can* we use them for? Kim and Swingle proposed an ansatz called DMERA (Figure 1.1), based on a known tensor network (MERA), that can in theory function well on a NISQ era computer [1]. They suggest that the ansatz is robust against noise and does not require many qubits to function properly. DMERA is used to prepare ground states (lowest energy quantum states) of a physical model, which can then be used to compute physically interesting things such as the energy or other quantities.

DMERA is based on tensor networks which are conventionally implemented on classical computers. Tensor networks are used to simulate quantum mechanical systems, such as a chain or lattice of strongly coupled spin particles. Ultimately, the goal is to prepare ground states of such quantum systems and compute values which are of interest. These networks have been used successfully in many studies and outperform other classical methods [10, 11, 12].

A commonly used tensor network is the matrix product state (MPS) which is very efficient for computations of one-dimensional systems. However, there is one particular scenario where it is not efficient: the critical domain. A material in a phase transition or highly entangled

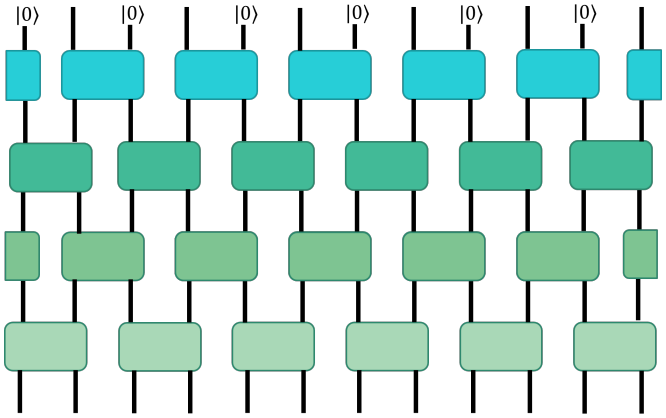


Figure 1.1: An example of DMERA.

systems are both examples of a critical system. MPS does not exhibit the same properties that these critical systems have, for example scale invariance or entanglement entropy scaling. There is however one type of tensor network that has the same properties as many critical systems and thus should be able to perform well in simulating criticality.

The multiscale entanglement renormalization ansatz (MERA) is built specifically to preserve entanglement and is designed with critical systems in mind [13, 14]. It also has the properties that follow the critical systems' properties, such as correct entanglement entropy scaling and decay of correlation functions [15].

Kim and Swingle combined both MERA and quantum computing to obtain DMERA. By making adjustments to tailor MERA for quantum computers, they developed a method that can be used efficiently and accurately on NISQ era quantum computers.

One big hurdle to deal with is the fact that quantum computers make many mistakes. Luckily, DMERA is intrinsically noise resilient due to an inner layer structure. At each layer, new pure qubits are introduced and old used qubits which contain noise are removed. As a result, the accumulating noise during the computation stays limited and the approximation of the ground state becomes more accurate. We will discuss the bound on the noise and a corresponding mathematical proof in Section 3.2.3.

Furthermore, DMERA can be computed quickly and efficiently on a quantum computer. With a small enough gate time, a quantum computation of DMERA would only take a few seconds [1]. If we were to simulate DMERA on a classical computer instead, we would need to store the information in memory and use a lot of time to compute the gates in DMERA. In comparison, for a DMERA circuit depth D a quantum computer spends $\mathcal{O}(D \log N)$ time to compute a circuit with $\mathcal{O}(D^d)$ qubits where N is the system size and d the number of spatial dimensions whereas a classical computer would have to spend time that scales *exponentially* in $\mathcal{O}(D^d)$.

Although a classical computer cannot efficiently simulate DMERA for many qubits, it is nonetheless convenient to use a classical simulation for verification of the results from a real quantum computer. A quantum computer will always be prone to errors and noise, while a classical computer can implement DMERA without any faults. A classical simulation can be used to predict what *should* happen and also what *will* happen on a quantum computer. We can simulate a noisy environment by choosing a noise model and applying it to the classical simulation to replicate the errors that typically occur on a quantum computer.

With a classically simulated and real quantum computer in hand, we can test the perfor-

mance of DMERA by preparing quantum states and measuring the energy of the states. The prepared states should approximate the ground state better and better the more DMERA layers we use, which we can test this by measuring their energy. We should find a convergence to the true ground state energy as a function of the number of applied layers. In Section 4.2 we discuss the method and results of this experiment.

The same experiment has been performed by Sewell and Jordan on a different quantum computer [2]. In their results they observed a convergence to a value close to the true ground state energy and they were able to verify the theoretical proposal of Kim and Swingle. In this thesis, we have used the methods of Sewell and Jordan to verify and build upon their results.

Contributions

In their paper they briefly discuss the concept of scaling dimensions which are a property of a critical model, specifically of one that is described by a conformal field theory. Scaling dimensions can be of physical interest, so finding a way to obtain these quantities from the measurement data would be useful. The scaling dimensions of a critical model can be corresponded with the scaling dimensions of DMERA, so instead we will try to extract the scaling dimensions of DMERA using a new quantum algorithm.

The DMERA scaling dimensions can in turn be computed from its eigenvalues. As briefly mentioned before, DMERA is constructed from several layers and each layer is an application of some function Φ such that the DMERA can be described as repeated iterations of Φ. The function Φ and its eigenvalues dictate the convergence to the ground state and this can easily be observed in the convergence of the expectation value of an observable O (Figure 1.2). For n iterations of Φ, the expectation value of the observable $\langle O \rangle$ shows an exponential decay to some final value. The decay contains information on the eigenvalues and thus the scaling dimensions.

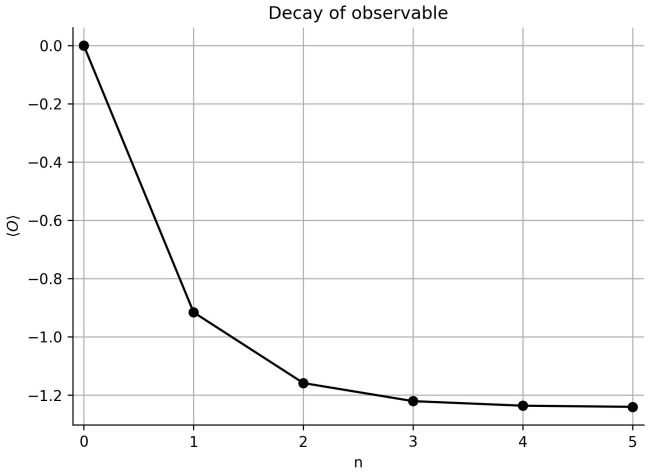


Figure 1.2: Decay of the expectation value of an observable $\langle O \rangle$ as a function of the number of DMERA layers n.

We propose a new quantum algorithm to extract the eigenvalues from the decay. This algorithm uses a sub-algorithm called ESPRIT [16] which is the key part of the new algorithm. The ESPRIT algorithm can estimate the eigenvalues that could be present in a decay curve, such as the one displayed in Figure 1.2. The new algorithm combines ESPRIT and a statistical test to determine how many and which eigenvalues are most likely present in a decay curve.

From Figure 1.2 we could find a few eigenvalues of DMERA, but to find more eigenvalues and map out the eigenvalue structure we would need more decay curves to draw information from. By picking many observables and starting states, we can cover the whole basis space of DMERA and theoretically extract every eigenvalue of DMERA. Using the new algorithm, we are able to extract several eigenvalues from the data and correspond them to the scaling dimensions of DMERA.

To summarize, the current generation of quantum computers make many mistakes in their computations, so Kim and Swingle proposed an ansatz for a quantum circuit which is robust against noise and can prepare states that are useful for research of critical physical models. Using the methods described by Sewell and Jordan, we have computed the energies of the prepared states and verified both their experimental results and the theoretical proposal of Kim and Swingle. Finally, we propose a new quantum algorithm that can extract scaling dimensions from measurement data by finding eigenvalues of DMERA.

In this thesis we research the quantum Ising chain model with transverse field as our critical model. This is a relatively simple model and easily verifiable as this model has been examined on both MERA by Evenbly and White [17] and DMERA by Sewell and Jordan.

Organization

In Chapter 2 we introduce basis concepts from quantum computing and information theory which appear frequently in the thesis. In Chapter 3 we discuss the MERA and DMERA tensor networks by first introducing MERA and its properties and then making the step to its variant DMERA which has similar properties as MERA. In Chapter 4 we examine several concepts that are used for experiments on a quantum computer and its simulation thereof before performing an experiment to measure the energies of prepared states by the DMERA circuit. Finally, in Chapter 5, after examining the eigenvalue structure of DMERA, we introduce a new quantum algorithm that can extract the eigenvalues of the DMERA channel using data from measurements and we test its performance on simulated data.

Chapter 2

Quantum computing

Quantum computers work inherently different from classical computers, however many parallels can be drawn between both types of computers. Both use a form of bits, the building blocks of the logical computer, and logic gates to control the bits. The gates can be combined into a circuit which outputs a desired result. In the case of classical computing, this result is always determined by the input (save for any incidental errors). However, for quantum computers, the result is dependent on the probabilities of the possible states as prepared by the circuit. The prepared state is in a superposition of all possible outcomes so one trial of the circuit can return a different output from another trial. The concept of superposition is unique to quantum computers and when used properly it can significantly speed up computations compared to classical computers. Superposition is used in many quantum algorithms, such as in Shor's [18] and Grover's [19] algorithm, for a great decrease in computation time over classical algorithms.

This section serves as a brief introduction to the basics of quantum computing. For further and more thorough discussion on quantum computing and information, see Nielsen and Chuang [20] or Watrous [21].

In this chapter, we will introduce the quantum bit and its properties in Section 2.1, and quantum gates to manipulate the quantum bits in Section 2.2. We then introduce quantum channels which generalizes the idea of sending or manipulating quantum information in Section 2.3. Lastly, we discuss entanglement entropy which can be used to quantify entanglement between quantum (sub)systems in Section 2.4.

2.1 Quantum bits

A quantum bit (qubit) is the fundamental building block of quantum computing and is the quantum counterpart of the classical bit. Formally, we describe it as a quantum state or object that exists in a two-dimensional Hilbert space $\mathcal{H} = \mathbb{C}^2$. A qubit can be described by two basis vectors for which we will choose the *computational basis* $\{|0\rangle, |1\rangle\}$, where

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{2.1}$$

In general, a qubit, represented by the vector $|\psi\rangle \in \mathcal{H}$, will be in a superposition of $|0\rangle$ and $|1\rangle$, that is $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle$. Due to normalization, we require that $|\alpha|^2 + |\beta|^2 = 1$.

We can combine qubits to represent quantum systems larger than one qubit by taking the tensor product. To describe a two-qubit system, we join together two single qubits, that is

$$|\omega\rangle = |\psi\rangle \otimes |\phi\rangle. \tag{2.2}$$

In computational basis, we often leave out the tensor product sign $\otimes$ to get $|0\rangle \otimes |1\rangle \rightarrow |01\rangle$. This concept can be trivially extended to an n-qubit system.

Density matrices

Commonly the density matrix, or operator, $\rho \in L(\mathcal{H})$ is used to describe a quantum state instead of $|\psi\rangle$. In general,

$$\rho = \sum_i p_i \rho_i = \sum_i p_i \, |\psi_i\rangle\langle\psi_i| \,, \tag{2.3}$$

where $p_i \geq 0$ are the probabilities for ρ_i. In addition, the trace of ρ must be one, i.e. $\mathrm{Tr}[\rho] = 1$. Formally, a density matrix ρ is a positive semidefinite operator as it is Hermitian and its eigenvalues are non-negative. We denote this as $\rho \in PSD(\mathcal{H})$.

Both vector and density matrix formulations are equivalent to each other, however in some situations one may be more convenient to use than the other. The vector formulation $|\psi\rangle$ is useful when the quantum system is in a pure state, such as an eigenstate of the system. The density matrix formulation can represent both pure states and *mixed states*. Mixed states are used to describe entanglement between two subsystems, which is otherwise not possible to do with the vector notation.

2.2 Quantum circuits

The goal of quantum computing is to manipulate qubits in such a way that a measurement returns the answer to a problem. In order to manipulate the qubits, a set of instructions should be constructed where the instructions are *unitary transformations*. In quantum computing, unitary transformations are represented as quantum (logic) gates. Quantum circuits can be built from these quantum gates by placing them in series and parallel for n qubits.

Operating with a quantum gate $U \in L(\mathcal{H})$ on a qubit $|\psi\rangle$ is defined as $U\,|\psi\rangle$. For density matrices, a gate U acts both on the left and right of ρ,

$$U[\rho] = U\rho U^\dagger.$$

Furthermore, a unitary U has the property that $U^\dagger U = I$ and it follows then that the inner product is conserved after a unitary transformation: $\langle U\psi, U\phi\rangle = \langle\psi|\, U^\dagger U \,|\phi\rangle = \langle\psi|\phi\rangle = \langle\psi,\phi\rangle$. Additionally, by the cyclic property of trace we have that $\mathrm{Tr}[U[\rho]] = \mathrm{Tr}[\rho] = 1$ which implies that unitaries map quantum states to quantum states.

Commonly used quantum gates are

- Pauli spin operators $X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, $Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, $Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
- Hadamard gate $H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
- Phase gate $P(\phi) = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix}$
- Controlled NOT gate $\mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

Quantum circuits

Quantum circuits are the bread of quantum computing and quantum states are the butter. Circuits act on n qubits and are a (tensor) product of quantum gates. For example, to prepare a so-called Bell state $|\Phi^+\rangle$ we can construct a small quantum circuit which uses only two gates. In equation form this reads

$$\begin{aligned}\mathrm{CNOT}(H \otimes I)\,|00\rangle &= \mathrm{CNOT}\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \\ &= \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle),\end{aligned}$$

where H is the Hadamard gate which brings a $|0\rangle$-state in an equal superposition of $|0\rangle$ and $|1\rangle$ and CNOT performs a spin flip on the second qubit if the first qubit is in the $|1\rangle$-state. However, we can also represent this operation with a diagram. In Figure 2.1 we visualize the qubits as wires running from left to right passing through the gates. Measuring the prepared state in computational basis will result in the $|00\rangle$-state in 50% of the results and the $|11\rangle$-state in the other 50%. A single measurement will yield only one of the results, therefore it is crucial to perform many trials and measurements of the circuit to get a realistic view of the actual prepared state.

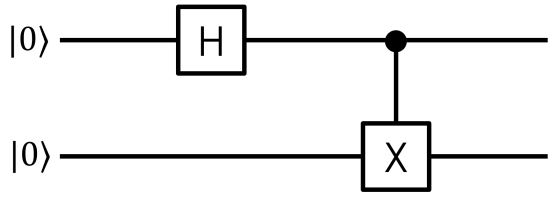


Figure 2.1: A quantum circuit that prepares the Bell state $|\psi\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$ from the starting state $|00\rangle$. The H-gate denotes the Hadamard gate, which brings the first qubit in an equal superposition of $|0\rangle$ and $|1\rangle$. The X-gate together with the dot denotes the CNOT gate, where the first qubit is the control bit. The X-gate is performed only if the first qubit is in the $|1\rangle$-state.

2.3 Quantum channels

Quantum circuits are a great tool to effectively implement a set of instructions, but we will want to use a more general notion. For this, we will consider *quantum channels* [20]. A quantum channel is a way to transmit quantum and classical information. Usually, this information is in the form of qubits.

Formally, a quantum channel Φ is a linear map that is completely positive and trace preserving [21]. Complete positivity means that $\Phi \otimes I_R$ sends any positive semidefinite operator to another positive semidefinite operator for all $\mathcal{H}_R$. Trace preserving simply implies that the trace of an operator is preserved after an application of the channel, i.e. $\mathrm{Tr}[\Phi[X]] = \mathrm{Tr}[X]$ for $X \in L(\mathcal{H})$. These two definitions ensure that any quantum state $\rho \in PSD(\mathcal{H}_A)$ gets sent to another quantum state $\sigma \in PSD(\mathcal{H}_B)$.

A quantum circuit follows the definition of a quantum channel, however the opposite is not always true. In Section 2.2 it was clear that quantum circuits are a combination of quantum gates which are by definition unitary, and a quantum circuit is then by extension also unitary. A

quantum channel, however, does need not be unitary. Unitarity implies reversibility and there exist some channels that cannot be reversed. Consider a quantum channel Φ that measures a state and prepares the state $\rho_0 = |0\rangle\langle 0|$. Then for any ρ,

$$\Phi[\rho] = \rho_0 = |0\rangle\langle 0|\,,$$

which is a valid quantum channel but not unitary.

Quantum channels can act consecutively on a state ρ. If first channel Φ is applied to ρ and then the channel Ψ, we write this as $\Psi[\Phi[\rho]]$ or $\Psi \circ \Phi[\rho]$. In the case that $\Phi = \Psi$ we can alternatively write this as $\Phi^2[\rho]$.

2.4 Entanglement entropy and area law

Quantum systems can have either classical or quantum correlations between each other. These two types of correlations can be distinguished from each other by writing out the joint quantum state in terms of the individual qubit registers. If a state has quantum correlations then we speak of entanglement.

Separable states can have classical correlations and are of the form

$$\rho_{AB} = \sum_i p_i\, \rho_{A,i} \otimes \rho_{B,i}, \tag{2.4}$$

where ρ_{AB} is the joint state of two qubits A and B [21]. An example of a separable state would be $\rho = \frac{1}{2}\left(|00\rangle\langle 00| + |11\rangle\langle 11|\right)$, which is analogous to two coins that always land on the same side together.

Entangled states are states that *cannot* be written as a separable state. This implies that there is some correlation between the subsystems that is not classical. Commonly used entangled states are the *Bell states*,

$$\rho_{Bell} = \left|\Phi^+\right\rangle\!\left\langle\Phi^+\right|; \qquad \left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right). \tag{2.5}$$

This state cannot be written in the form of Eq. 2.4 so it is an entangled state.

The "amount" of entanglement between subsystems can be measured with the von Neumann entropy, or entanglement entropy, which is defined as

$$S(\rho) = -\operatorname{Tr}[\rho \log \rho] = -\sum_i p_i \log p_i. \tag{2.6}$$

Consider the Bell state from Eq. 2.5. The entanglement entropy of register A is found by computing $S(\rho_A) = -\operatorname{Tr}[\rho_A \log \rho_A]$, with $\rho_A = \operatorname{Tr}_B[\rho_{Bell}]$, to get $S(\rho_A) = \log 2$ which turns out to be the largest value for a two-dimensional system that one can get. For this reason the Bell state is sometimes called the maximally entangled state.

Area law

If we were to pick a random state ρ from a Hilbert space $\mathcal{H}$, there will generally be some entanglement between the subsystems of ρ which we can quantify with the von Neumann entropy. Suppose we pick a random state $\rho_{rdm} \in \mathcal{H}$ for L sites with dimensionality d, then $S(\rho_{rdm}) \approx L^d$, i.e. the entropy scales as the volume of the system. Most of the p_i in ρ_{rdm} are non-vanishing so the entanglement entropy becomes large.

Now, a ground state ρ_{gs} of a non-critical Hamiltonian H will generally have only a few relevant p_i so instead the entropy scales as the *boundary* of the system, $S(\rho_{gs}) \approx L^{d-1}$. This is known as the area law of entanglement entropy [22]. Additionally, ground states of *critical* Hamiltonians acquire a logarithmic correction, so $S(\rho_{gs,crit}) \approx L^{d-1} \log L$.

Chapter 3

Tensor networks

Quantum systems can have exponentially large associated Hilbert spaces, which can be difficult to deal with for a classical computer. Let us consider a joint system of 3 spins. The associated Hilbert space of this system is $\mathcal{H}_{012} = \mathcal{H}_0 \otimes \mathcal{H}_1 \otimes \mathcal{H}_2$ which has a dimensionality of $2 \cdot 2 \cdot 2 = 2^3 = 8$. The dimension of the Hilbert space for a n-spin system would be 2^n, exponential in n. An exponentially growing Hilbert space dimension is concerning if one wants to classically simulate a quantum system. Classical computers that are used for quantum physical calculations will have to store that information in memory which makes it difficult to handle systems with numerous particles as the memory requirement quickly grows.

Fortunately, tensor networks, a type of computational method for classical computers, are designed to tackle many-body quantum systems differently that avoids the exponential growth problem. By only considering a select part of the full Hilbert space, tensor networks do not need to consider an exponentially growing space. In addition, tensor networks offer up accuracy in trade for speed and efficiency by limiting the size of its tensors. This trade is allowed to work since tensor networks are generally used to approximate ground states. Ground states have special properties compared to a general state, such as the entanglement entropy scaling which we discussed in Section 2.4, which is why tensor networks can approximate ground states in a computationally favorable way.

One of the most commonly used tensor networks is the so-called matrix product state (MPS) [23, 24]. In short, MPS works by decomposing a large tensor, e.g. a state which describes many spin particles, into a chain of smaller tensors whose size depends on the number of particles. By restricting the tensors' size up to a certain value χ we decrease the amount of information in our computations, which makes it easier for a classical computer to deal with the calculations. One might think that we then lose too much information to accurately approximate the quantum system, but by keeping the most relevant states during the size restriction, MPS is still able to make an accurate approximation if the value χ is chosen appropriately. For more discussion on MPS, see the review of Schöllwock [24].

In this chapter, we will introduce and discuss the multiscale entanglement renormalization ansatz (MERA), a different type of tensor network. Later, we will introduce the deep MERA (DMERA), a variant on MERA which is suited to be used on a quantum computer.

3.1 MERA

The matrix product state is not efficient dealing with critical systems, as their properties (e.g. entanglement entropy scaling, lack of scale invariance, decay of correlation function) do not

align with the properties of critical systems. MPS can still approximate the ground state of a critical system, but cannot capture all the behaviors of such a system.

The multiscale entanglement renormalization ansatz (MERA) is a tensor network that is based on the concepts of renormalization group and entanglement, and aims to handle (one-dimensional) critical Hamiltonians in a more efficient manner than MPS [14, 25, 26, 27]. MERA has several properties which coincide with the properties that critical systems also have, which we will discuss in detail in the following section.

In this section, we will first formally define a translation and scale invariant one-dimensional binary MERA, and discuss several aspects such as bond dimension and causal cone. Then we will look at scale invariance in more detail and define the ascending/descending superoperators in Section 3.1.5. Lastly, we discuss the evaluation of expectation values of observables in Section 3.1.6.

3.1.1 Formal definition

A translation and scale invariant one-dimensional binary MERA (Figure A.1) can be seen as a transformation from an initial state $|\psi_0\rangle$ that lives on a lattice $\mathcal{L}_0$ to a final state $|\psi_T\rangle$ on a lattice $\mathcal{L}_T$. A lattice $\mathcal{L}_i$ containing n_i sites with local dimension d has an attached Hilbert space $\mathcal{H}_{\mathcal{L}_i} = \mathbb{C}_0^d \otimes \mathbb{C}_1^d \otimes ... \otimes \mathbb{C}_{n_i-1}^d$, for now we assume n_i to be an arbitrary integer. The transformation takes place in T steps; the number of steps depends on how the MERA is structured. Let U describe the MERA, then

$$U = U_{T-1}\, U_{T-2} \cdots U_0$$

with $U_i : \mathcal{H}_{\mathcal{L}_i} \mapsto \mathcal{H}_{\mathcal{L}_{i+1}}$, which are referred to as (MERA) layers. Then the state transforms with each layer,

$$|\psi_0\rangle \to |\psi_1\rangle \to ... \to |\psi_T\rangle\,,$$

where $|\psi_{i+1}\rangle = U_i\,|\psi_i\rangle$. Each U_i is described by two tensors u and w. The u-tensors, also called *disentanglers*, are described by

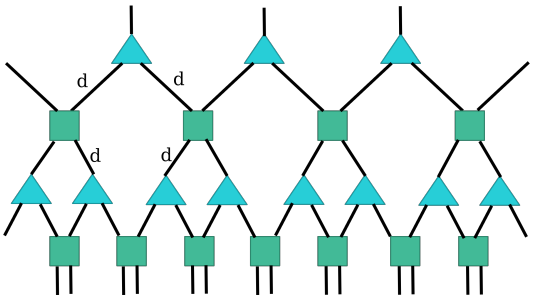
$$u : (\mathbb{C}^d)^{\otimes 2} \mapsto (\mathbb{C}^d)^{\otimes 2}, \qquad u^\dagger u = u\, u^\dagger = I_{d^2}, \tag{3.1}$$

and the w-tensors, called *isometries*, are described by

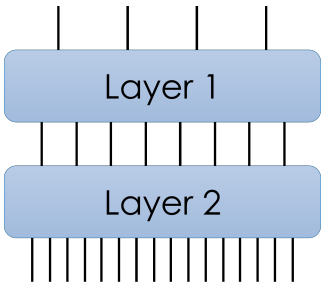
$$w : \mathbb{C}^d \mapsto (\mathbb{C}^d)^{\otimes 2}, \qquad w^\dagger w = I_d. \tag{3.2}$$

As MERA is a tensor network, we can represent the network using the diagrammatic tensor notation, see Appendix A.1. A graphical representation is shown in Figure A.1. The lattice points in $\mathcal{L}_i$ are represented as wires and the tensors as (x_{in}, x_{out})-tensors, where x_{in} is the number of wires going into the tensor and x_{out} the number of wires leaving the tensor. We know that each isometry w is the same by translational invariance, which implies that all ingoing wires must have the same dimension d and vector space $\mathbb{C}^d$. By the definitions in Eq. 3.1 and 3.2 we can see that the w-tensors are of the type (1,2) and the u-tensors are of the type (2,2). The tensors are laid out in a brick wall structure where the u-tensors entangle neighboring wires.

Here we have explicitly defined a specific MERA scheme, a binary MERA, by choosing the isometries to map from $\mathbb{C}^d$ to $(\mathbb{C}^d)^{\otimes 2}$. One can also define a ternary MERA by letting the co-domain of the isometry be $(\mathbb{C}^d)^{\otimes 3}$ [28]. Different schemes also exist such as a modified binary MERA and branching MERA [26, 27], or even generalizations to higher dimensions [14]. These schemes differ in contraction cost scaling and efficiency of accuracy. In this thesis we will assume that every MERA is binary for simplicity.



(a) Two MERA layers. The isometries are represented by the blue tensors and disentanglers by the green tensors. The dimension of the tensors is determined by the bond dimension d.



(b) Consecutive MERA layers which show the fine-graining effect of MERA transformations. Each layer introduces new sites and entangles them with the existing sites. The inner structure of the two layers is explicitly shown in (a).

Figure 3.1: The effects of MERA transformations on a small and large-scale level.

Criticality properties

In general, we want MERA to have similar properties to the systems we study, namely critical systems. Two of those properties are translation and scale invariance. Earlier, we asserted that all isometries and disentanglers are the same in U_i. This choice ensures that the structure is invariant under shifting positions of the tensors. One can also let all tensors be different and freely adjust each tensor to optimize the network. For scale invariance all U_i consist of the same tensors $\{w, u\}$. Each layer U_i is then independent of the scale. By requiring these invariance properties, for a given local dimension d and number of MERA layers N, we lower the number of parameters to describe the tensors from $\mathcal{O}(d^4 N)$ to $\mathcal{O}(d^4 \log_2 N)$ by translational invariance, and even further to $\mathcal{O}(d^4)$ by scale invariance. We will return to scale invariance of MERA in Section 3.1.5.

At this point, we should look at the similarities between quantum circuits and the given definition of MERA. Quantum circuits implement a transformation from an initial n-qubit state $|\psi_0\rangle$, by convention $|0\rangle^{\otimes n}$, to a final state $|\psi_f\rangle$ by applying various quantum gates to the qubits. Similarly, MERA transforms an initial state $|\psi_0\rangle$ to a final state $|\psi_T\rangle$ by repeatedly applying quantum circuits. If we think of MERA as a particular quantum circuit, then we can interpret the lattice points as qubits and the $\{w, u\}$ unitaries as two-qubit quantum gates. Precisely this interpretation is what makes MERA a particularly good and natural candidate for a tensor network that can be implemented on a quantum computer.

3.1.2 Bond dimension

From the current definition of MERA it is not apparent how to adjust the size of the tensors which can impact the accuracy and efficiency of the ansatz. As it is currently defined, the size of the tensors depends only on the dimensionality d of the lattice sites, e.g. $d = 2$ for spins. However, we can change d into a parameter which we can adjust by grouping sites together.

The performance of MPS is often indicated with a quantity χ_{MPS} which serves to put a limit on the bond dimension, the dimension between the tensors. The larger the bond dimension is allowed to be, the more accurate MPS will be. This however comes with a (literal) cost of computational time. We can use a similar notion to quantify the accuracy of MERA. Earlier we saw that the number of parameters needed to describe the tensors that make up MERA is $\mathcal{O}(d^4)$, where d is the dimension of the legs that point in- or outwards of the tensor. If we can

increase d, then the number of parameters increases, and therefore the accuracy of MERA.

Increasing bond dimension

Given a lattice $\mathcal{L}_0$ with n_0 sites that have dimension d, let us group s sites together into bigger sites with dimension d^s (Figure 3.2). The MERA that would act on this now has to contain tensors with larger dimensions, namely $d^{2s} \times d^{2s}$ for disentanglers and $d^{2s} \times d^s$ for isometries. These bigger tensors contain more elements and thus more number of parameters, and we theorized that the accuracy of MERA should therefore increase. In Figure 3.3 the relative error rate of the computed ground state energy and the exact ground state energy as a function of the bond dimension for both MERA and MPS is shown [25]. As expected, MPS becomes more accurate for larger values of χ_{MPS}. Similarly, MERA is more accurate for increasing bond dimensions χ_{MERA}, what we have been calling the local dimension d thus far. At this point it seems appropriate to promote the local dimension d to the bond dimension χ_{MERA} as we have shown that it fulfills a similar role to χ_{MPS}; it is a parameter that signifies the accuracy of the corresponding tensor network.

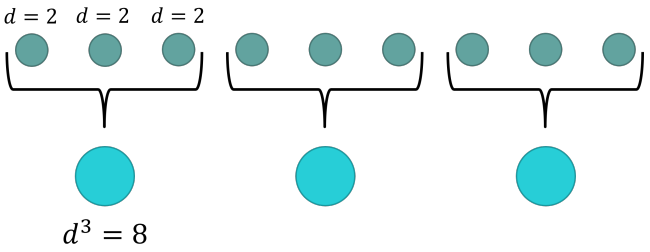


Figure 3.2: Grouping of sites to increase the dimensionality of a "site".

Note that grouping sites together is purely for computational purposes and, in most cases, has no physical meaning. However, it does not mean that we *lose* physical meaning. Any observable that we may be interested in can be transformed to this grouping ansatz. For example, a d-dimensional observable h can be appropriately embedded by $s-1$ identities such that it becomes a d^s-dimensional observable, that is

$$h \mapsto h' = I \otimes ... \otimes I \otimes h \otimes I \otimes ... \otimes I,$$

where I are $d \times d$ identities.

The grouping ansatz can be generalized by transitional layers, which are placed before the scale/translational invariant MERA. The transitional layers are allowed to have arbitrary isometries and disentanglers, with arbitrary dimensions, that generally differ from the ones used in the MERA. As such they can be freely adjusted in a method called *tensor optimization* where by an iterative process each individual tensor is optimized in such a way that the following MERA better approximates the ground state energy. We can consider a transitional layer that contains isometries (or identities) that simply group two sites together, that is $w : \mathbb{C}^d \otimes \mathbb{C}^d \mapsto \mathbb{C}^{2d}$, which can be extended to allow for an arbitrary number of sites to be grouped together. Besides this specific application of transitional layers, we will not make further use of them. For more discussion on transitional layers and tensor optimization, see [26].

3.1.3 Causal cone

Up until this point we did not assume anything about the width of the initial lattice $\mathcal{L}_0$ because that does not matter yet until we contract the network. As it turns out, if we place an observable

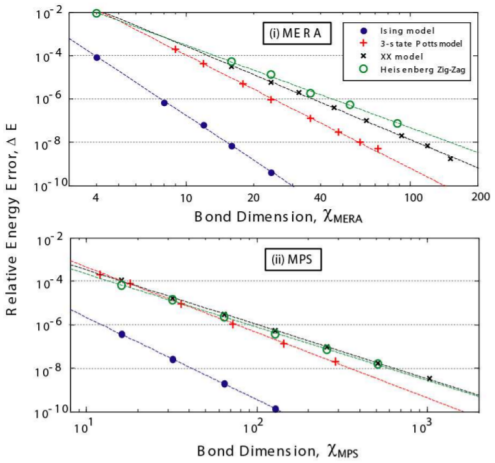


Figure 3.3: Relative error in ground state energy of a select few critical spin models, for both MERA and MPS. The energy error scales polynomially with the bond dimension for both tensor networks. Figure from [26].

at the bottom of the network and compute the expectation value of that observable (which we will do later in Section 3.1.6), only certain tensors can effect the state that ends up at the observable. These tensors are part of the *causal cone*, as shown in Figure 3.4. The notion of a cone of causality is not unique to MERA; it is similar to the light cone that can be found in the theory of relativity. Events (tensors) that happen outside the cone cannot influence the present (observable).

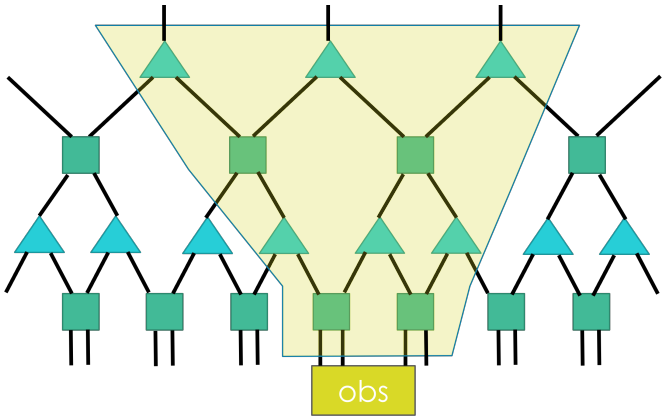


Figure 3.4: The causal cone of a three-site observable (obs) in MERA. The cone has a fixed width for this operator.

The MERA contains tensors that are not causally connected to an observable O (i.e. not in the causal cone), so there will be situations where u directly contracts with $u^\dagger$ or w with $w^\dagger$, see Figure 3.6. As defined earlier, we know that $u^\dagger u = u\,u^\dagger = I$ and $w^\dagger w = I$, so these contractions contribute nothing to the computation and we can safely ignore them. We can say that the tensors that do not annihilate with each other are elements of the causal cone.

Fixed widths

A practically useful property of the causal cone of a MERA is that the width, in terms of the number of sites, will converge to a fixed value which depends on the support of the observable. Let us consider an three-site observable O (Figure 3.6). After the contraction of all the tensors that do not effect O, there are only three sites left that did not contract to identity, so the causal cone of a three-site observable also has a width of three sites. This is one of the few fixed values but we can write down a general formula to find the other fixed values. We will investigate this by seeing what the effect of one $U_i^\dagger$ transformation is on n sites, or rather, how the width of the causal cone of n sites changes. We will separate the cases of odd and even number of sites.

For an odd number of sites, n sites will connect to $\lceil \frac{n}{2} \rceil$ disentanglers ($u^\dagger$) which in turn connect to $\lceil \frac{n}{2} \rceil + 1$ isometries ($w^\dagger$). The width of a causal cone with n sites as its base is $\lceil \frac{n}{2} \rceil + 1$ sites after going up one layer,

$$n \to \left\lceil \frac{n}{2} \right\rceil + 1. \tag{3.3}$$

As an example for $n = 1$, the single site must connect to a single disentangler which connects to 2 isometries, so the causal cone now has a width of 2 sites. Similarly, $n = 3$ has a cone width of 3 sites, $n = 5$ a cone width of 4 sites and $n = 7$ a cone width of 5 sites. An interesting observation we can already make, is that a causal cone of 3 sites has a fixed width.

The counting works differently for an even number of sites. Instead, n sites can connect to either $\frac{n}{2}$ or $\frac{n}{2} + 1$ disentanglers depending on which n sites are being considered. Let us call an observable *aligned* if it follows the former case and *disaligned* for the latter case, see Figure 3.5. An aligned observable will then connect to $\frac{n}{2} + 1$ isometries and the latter case to $\frac{n}{2} + 2$ isometries. The causal cone of n sites has a width of either $\frac{n}{2} + 1$ or $\frac{n}{2} + 2$ after going up one layer,

$$n \to \frac{n}{2} + 1 \quad \text{(aligned)} \tag{3.4}$$

$$n \to \frac{n}{2} + 2 \quad \text{(disaligned)}. \tag{3.5}$$

This implies that the cone of $n = 2$ has a width of 2 or 3 sites, $n = 4$ a cone width of 3 or 4 sites and $n = 6$ a cone width of 4 or 5 sites. Again, we observe that 2 and 4 sites can have a causal cone with a fixed width, depending on the exact positions of those sites in relation to the MERA.

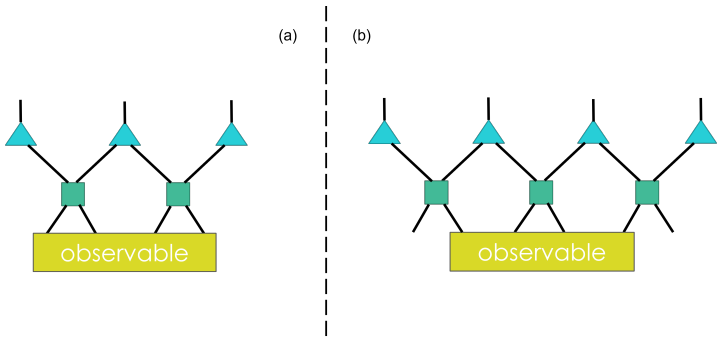


Figure 3.5: A four-site operator that is (a) aligned and connected to two disentanglers; or (b) disaligned and connected to three disentanglers.

We have seen that 2 and 4 sites can have a causal cone with a fixed width in some situations,

whereas 3 sites are guaranteed to have a constant width for its causal cone. Given that there are n_i sites on each lattice $\mathcal{L}_i$, we would have had to compute $\mathcal{O}(n_i^2)$ tensors for each MERA layer. Now, in the worst case, we only have to consider up to 7 tensors per layer that contribute to the expectation value of an observable. This significantly reduces the amount of tensors that take part in computations, and consequently reduces the computation time.

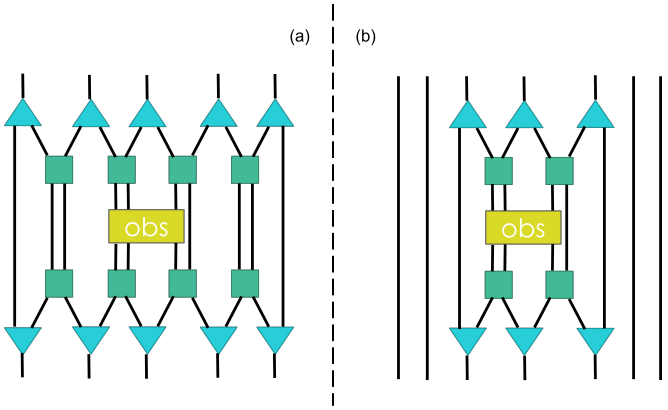


Figure 3.6: Contraction to identity of tensors that are not causally connected to the three-site operator O.

Superoperators

To make the concept of a causal cone more precise, we turn to *ascending superoperators*. Up until this point we have imagined the MERA to transform an initial state to a final state, $|\psi_f\rangle = U|\psi_0\rangle$. If we computed the expectation value of an observable O, this would be $\langle\psi_f|O|\psi_f\rangle$. However, from a mathematical perspective it is equivalent to transform the observable O by absorbing U and $U^\dagger$ to obtain the transformed observable O', that is

$$\langle\psi_f|O|\psi_f\rangle = \langle\psi_0|U^\dagger O U|\psi_0\rangle = \langle\psi_0|O'|\psi_0\rangle\,. \tag{3.6}$$

However as we know, not all tensors in U are involved with O. Precisely which tensors are involved is dependent on the support of O, more so which exact sites support O. As proof by example, a two-site operator that is directly connected to two disentanglers will then connect to three isometries, whereas a two-site operator that is connected to just one disentangler will connect to two isometries. In the former case, O' now has a larger support than O; ideally we want the transformed operators to retain the same support.

For now, let us consider three-site operators, although this argument can be extended to any odd number of sites. A local operator $O_{i+1}^{[r-1,\,r,\,r+1]}$ on lattice $\mathcal{L}_{i+1}$ that is supported on adjacent sites $[r-1,\,r,\,r+1]$ can be transformed to the lattice $\mathcal{L}_i$ by the ascending superoperator $\mathcal{S}$ such that

$$O_i^{[r'-1,\,r',\,r'+1]} = \mathcal{S}[O_{i+1}^{[r-1,\,r,\,r+1]}], \tag{3.7}$$

where $[r'-1,\,r',\,r'+1]$ are adjacent sites on lattice $\mathcal{L}_i$. It can be ambiguous how exactly O is transformed with respect to left and right side of disentanglers, so instead we will average over both possibilities such that $\mathcal{S} = \frac{1}{2}(\mathcal{S}_L + \mathcal{S}_R)$. Eq. 3.7 can also be expressed in diagrammatic notation, see Figure 3.7. We can take this figure to be the definition of how $\mathcal{S}$ acts on three-site operators.

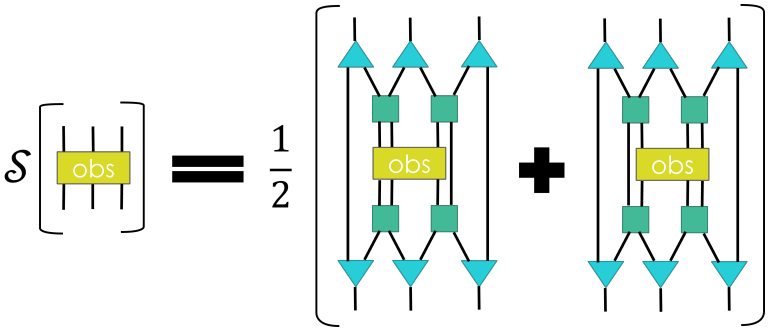


Figure 3.7: The ascending superoperator $\mathcal{S}$ acting on a three-site operator. Both possibilities $\mathcal{S}_L$ and $\mathcal{S}_R$ need to be averaged over.

Similarly, we can define a descending superoperator $\mathcal{D}$ given a density operator ρ on lattice $\mathcal{L}_i$,

$$\rho_{i+1}^{[r'-1,\,r',\,r'+1]} = \mathcal{D}[\rho_i^{[r-1,\,r,\,r+1]}], \tag{3.8}$$

where again we average such that $\mathcal{D} = \frac{1}{2}\left(\mathcal{D}_L + \mathcal{D}_R\right)$. The ascending and descending superoperators are each others dual, $\mathcal{D} = \mathcal{S}^*$, as we have the relation

$$\mathrm{Tr}(O_{i+1}\,\mathcal{D}[\rho_i]) = \mathrm{Tr}(\mathcal{S}[O_{i+1}]\,\rho_i), \tag{3.9}$$

which is equivalent to Eq. 3.6.

Both superoperators can be related to the Schrödinger and Heisenberg picture. The descending superoperator transforms the wavefunction which is precisely the Schrödinger picture, whereas evolving the operator corresponds to the Heisenberg picture. Both interpretations lead to the same result, however in certain situations it may be insightful to choose one over the other.

3.1.4 Entanglement entropy scaling

Ground states of critical one-dimensional Hamiltonians have a logarithmic scaling in its entropy, see Section 2.4. This scaling cannot be represented well by MPS, but MERA does succeed at that. The following derivation is based on [29] for $D = 1$.

An important property of the entanglement entropy is that it is upper bounded by $S(\rho) \leq \log_2 d$, where d is the dimension of ρ. Another property is the triangle inequality, which leads to $S(\rho_A) \leq S(\rho_{AB}) + S(\rho_B)$. Combining both properties we get

$$S(\rho_A) \leq S(\rho_{AB}) + \log_2 d. \tag{3.10}$$

In the process of preparing the fixed state we have the sequence $\rho_0 \to \rho_1 \to \ldots \to \rho_T$ by applying $\mathcal{D}$ T times. As we go through this sequence, we descend through the causal cone and trace out sites. The traced sites add entropy recursively, that is

$$S(\rho_z) \leq S(\rho_{z-1}) + \log_2(d)\, n_z^{tr}, \tag{3.11}$$

where $0 < z \leq T$ and n_z^{tr} is the number of traced sites. At any point the entropy in the sequence is upper bounded by $S(\rho_z) \leq l_z \log_2 d$, where l_z is the number of sites at scale z. Iterating Eq. 2.6 until we get to ρ_T gives us

$$S(\rho_T) \leq \log_2(d)\left(l_T + N_T^{tr}\right), \tag{3.12}$$

where $N_T^{tr} = \sum_{z=0}^{T-1} n_z^{tr}$.

From the causal cone discussion in Section 3.1.3, we know that the fixed width of the causal cone of 3-site observables is 3 sites. N_T^{tr} is proportional to the number of scales that are *not* at the fixed width yet. Starting with L sites, it takes roughly $\log_2 L$ iterations of $\mathcal{D}$ to reduce the cone width to 3 sites. To be more precise, we take $N_T^{tr} = 2\log_2(L-2)$. Finally, we obtain the entropy of ρ_T,

$$S(\rho_T) \leq \log_2(d)(3 + 2\log_2(L-2)). \tag{3.13}$$

The entropy scaling in Eq. 3.13 coincides with the entropy scaling for critical ground states. This implies that MERA is suited for approximating these ground states and capturing the critical behaviors.

3.1.5 Scaling superoperators

In Section 3.1.1 we assumed a scale invariant MERA, which implies that all scales have the same set of tensors as their building blocks. As such, the layers can be interpreted as repeatedly applying the same function. To be precise, we can refer to the ascending superoperator $\mathcal{S}$ that was introduced in the previous section now as the *scaling superoperator*. We assume that $\mathcal{S}$ is diagonalizable, then this superoperator has the property that

$$\mathcal{S}[\phi_a] = \lambda_a \phi_a, \qquad \langle \hat{\phi}_a, \phi_b \rangle = \delta_{ab}, \tag{3.14}$$

for eigenvalues λ_a and right eigenoperators ϕ_a [15]. $\hat{\phi}_a$ are the left eigenoperators of $\mathcal{S}$ and by duality the right eigenoperators of $\mathcal{D}$, the dual of $\mathcal{S}$. We identify ϕ_a as operators/observables that sit inside the MERA and $\hat{\phi}_a$ as "starting states" of MERA. The associated eigenvalues λ_a are the same for both ϕ_a and $\hat{\phi}_a$. In general, $\hat{\phi}_a$ need not be proper quantum states. In fact, only the first left eigenoperator is a quantum state.

$\mathcal{S}$ scales its eigenoperators by a factor λ_a with each application. Sometimes we will talk about the scaling dimension $\Delta_a = -\log_2 \lambda_a$. We can decompose any operator O in terms of ϕ_a,

$$O = \sum_a \langle \phi_a, O \rangle \phi_a = \sum_a \mathrm{Tr}\left[\hat{\phi}_a O\right] \phi_a, \tag{3.15}$$

where $\langle \cdot, \cdot \rangle$ is the Hilbert-Schmidt inner product. This then allows us to write the action of $\mathcal{S}$ on an arbitrary operator O as

$$\mathcal{S}[O] = \sum_a \lambda_a \,\mathrm{Tr}\left[\hat{\phi}_a O\right] \phi_a. \tag{3.16}$$

Due to the way we have defined MERA, it follows that $\mathcal{S}$ is unital. Let us again look at Figure 3.7, but now consider operator O to be the three-site identity operator $I \otimes I \otimes I$. Consequently, the tensors can contract with each other to identity and we are left with three straight wires, so $\mathcal{S}[I] = I$. This implies that there is at least one λ_a such that $\lambda_I = 1$. For the other eigenvalues λ_a we will assume that they are strictly less than 1.

There must then also be a fixed point of $\mathcal{D}$ that has the same eigenvalue $\lambda_I = 1$ due to the dual nature of $\mathcal{S}$ and $\mathcal{D}$, shown in Eq. 3.9. There exists a density matrix ρ^* such that $\mathcal{D}[\rho^*] = \rho^*$, so repeated applications of $\mathcal{D}$ converges an arbitrary density matrix ρ to the fixed point ρ^*,

$$\lim_{T\to\infty} \underbrace{\mathcal{D} \circ ... \circ \mathcal{D}}_{T} [\rho] = \rho^*. \tag{3.17}$$

The fixed point ρ^* depends on the MERA and its tensors w, u. By choosing the right construction for the tensors we can have ρ^* be the ground state of a (critical) Hamiltonian.

3.1.6 Evaluation of observables and correlators

The goal of MERA is to prepare an approximate ground state of a Hamiltonian and find the associated ground state energy. So far we have laid out the tools and properties of MERA necessary to do exactly that. Let us know discuss how to evaluate observables and two-point correlation functions

Observables

We will keep the assumption that the observable O is a three-site operator. From Section 3.1.5 we know that $\mathcal{S}[O] = \sum_a \lambda_a \operatorname{Tr}\left[\hat{\phi}_a O\right]\phi_a$, so then $\mathcal{S}^k[O] = \sum_a \lambda_a^k \operatorname{Tr}\left[\hat{\phi}_a O\right]\phi_a$. The expectation value of an observable O after k layers is $\langle O\rangle_{\rho_k} = \operatorname{Tr}[O\rho_k] = \operatorname{Tr}\left[\mathcal{S}^k[O]\rho_0\right]$. Combining both expressions for $\mathcal{S}_k[O]$ and $\langle O\rangle_{\rho_k}$ we get

$$\begin{aligned}\langle O\rangle_{\rho_k} &= \operatorname{Tr}\left[\sum_a \lambda_a^k \operatorname{Tr}\left[\hat{\phi}_a O\right]\phi_a \rho_0\right] \\ &= \sum_a \lambda_a^k \operatorname{Tr}\left[\hat{\phi}_a O\right] \operatorname{Tr}[\phi_a \rho_0].\end{aligned} \tag{3.18}$$

In the limit of k goes to infinity, Eq. 3.18 simplifies to

$$\langle O\rangle_{\rho^*} = \operatorname{Tr}[\rho^* O], \tag{3.19}$$

where ρ^* is the fixed point state of the descending superoperator $\mathcal{D}$.

There are three ways to prepare Eq. 3.19. The first one is to prepare the state by repeatedly applying the descending superoperator on the top layer until the bottom layer has been reached, where the operator O sits, to obtain the state ρ^*. The second approach is to apply the ascending superoperator to the observable O many times until the top layer has been reached. Lastly, in a general manner, one can use both ascending and descending superoperators to reach a layer τ, where $0 \leq \tau \leq T$. All three methods are computationally equivalent as they involve $\mathcal{O}(\log n)$ uses of the ascending/descending superoperator and ultimately lead to a cost of $\mathcal{O}(d^9 \log n)$ [25].

Two-point correlation functions

Now that we know how to evaluate observables we can extend this notion to two-point correlation functions of the form $\langle \phi_\alpha(x)\phi_\beta(y)\rangle$, where ϕ_α are the right eigenoperators of $\mathcal{S}$ and x and y indicate the position of the operators, for $x \neq y$. Even though no general closed expression exists for binary MERA, we can take an intuitive approach to this problem. Note that $\langle \phi_\alpha(x)\phi_\beta(y)\rangle \neq \langle \phi_\alpha(x)\rangle\langle \phi_\beta(y)\rangle$ as eventually their causal cones will connect after a certain number of iterations of $\mathcal{S}$ and the observables are not independent anymore.

Suppose x and y are sufficiently apart, that is they are not nearest neighbors, then their causal cones are not yet connected. After a number of layers their causal cones will be connected and at this point we can simply evaluate $\langle \phi_\alpha(x)\phi_\beta(y)\rangle$ like Eq. 3.19. The question now is how many iterations of $\mathcal{S}$ are necessary to achieve this.

To gain intuition for this question, let us look at the same situation but for a ternary MERA [15]. "Ternary" implies that isometries map one site to three sites. In this case $\phi_\alpha(x)$ and $\phi_\beta(y)$ can be placed at specific sites such that $x - y = 3^q$ for $q = 1, 2, \ldots$. Then, after $q = \log_3 |x - y|$ iterations of $\mathcal{S}$ transformations the operators $\phi_\alpha(x)$ and $\phi_\beta(y)$ are nearest neighbors and we have

$$\langle \phi_\alpha(x)\phi_\beta(y)\rangle = \frac{\operatorname{Tr}[(\phi_\alpha(x) \otimes \phi_\beta(y))\rho]}{|x - y|^{\Delta_\alpha + \Delta_\beta}}, \tag{3.20}$$

where $\Delta_\alpha = -\log_3(\lambda_\alpha)$ is the scaling dimension of ϕ_α.

Let us now return to a binary MERA. This is no longer an ideal case, but we can make an educated guess that we will need roughly $\log_2|x-y|$ iterations of $\mathcal{S}$ to have $\phi_\alpha(x)$ and $\phi_\beta(y)$ be nearest neighbors. The evaluation of the two operators will then look similar, namely

$$\langle\phi_\alpha(x)\phi_\beta(y)\rangle \approx \frac{1}{(\lambda_\alpha\lambda_\beta)^c}\frac{\mathrm{Tr}[(\phi_\alpha(x)\otimes\phi_\beta(y))\rho]}{|x-y|^{\Delta_\alpha+\Delta_\beta}}, \tag{3.21}$$

where now $\Delta_\alpha = \log_2(\lambda_\alpha)$. There might be an additional factor $(\lambda_\alpha\lambda_\beta)^{-c}$ for $c = 0, 1, 2, ...$ due to the exact placement of the operators and overestimation of the number of iterations of $\mathcal{S}$.

The most important feature of this expression is the polynomial decay of the correlator with respect to the scaling dimensions. This polynomial behavior of the correlations is also present for systems at the critical point and this regime is precisely what we are interested in.

To summarize, due to the way MERA is structured, we can generalize the tensor network to a repeated application of superoperators, which stems from the fact that the causal cone of an operator can have a constant width. These superoperators come with a set of eigenoperators and eigenvalues which describe the behavior of a general operator or state. Only one of these eigenoperators has eigenvalue $\lambda = 1$, which corresponds to the fixed point of MERA. Additionally, we discovered that MERA has similar properties to systems at the critical point, namely for the entanglement entropy scaling and decay of the correlator. Based on the properties, MERA looks like a better candidate for studying critical systems than MPS, which does not have the same properties.

3.2 DMERA

Now that we have established MERA we can move to DMERA. As hinted in the formal introduction of MERA (Section 3.1.1), we can interpret the network in a quantum manner by replacing the tensors with quantum gates and the legs with qubits. In this section, we formalize this concept and discuss several unique features of DMERA.

DMERA has been proposed by Kim and Swingle in 2017 as a method to study strongly interacting quantum many-body states at criticality on a noisy quantum computer [1]. They suggest that DMERA has several strong qualities – of which we will discuss a few in this section – namely fast contraction on a QC, approximation of a wide variety of physical ground states, resilience to noise and good approximation of the ground state energy in presence of noise.

In this section we will formally introduce DMERA and its differences with MERA, then we will discuss how the fixed width of the causal cone changes with larger depth D and finally we prove that DMERA with noisy gates has an upper bound on the cumulative error.

3.2.1 Formal definition

The deep MERA (DMERA) (Figure 3.8) is a quantum circuit that transforms an initial n-qubit state $|\psi_0\rangle = |0^n\rangle \in \mathcal{H}_0$ to a final state $|\psi_f\rangle \in \mathcal{H}_f$ in T steps as

$$|\psi_{i+1}\rangle = U_i\left(|\psi_i\rangle \otimes \left|0^{2^i}\right\rangle\right), \tag{3.22}$$

where U_i is a depth D quantum circuit (layer) and $\left|0^{2^i}\right\rangle$ contains 2^i qubits in $|0\rangle$-state which are placed alternating with the qubits from $|\psi_i\rangle$ [1]. Alternatively, we can merge U_i with $\left|0^{2^i}\right\rangle$,

$$U_i' = U_i \otimes [I \otimes |0\rangle \otimes I \otimes ... \otimes |0\rangle], \tag{3.23}$$

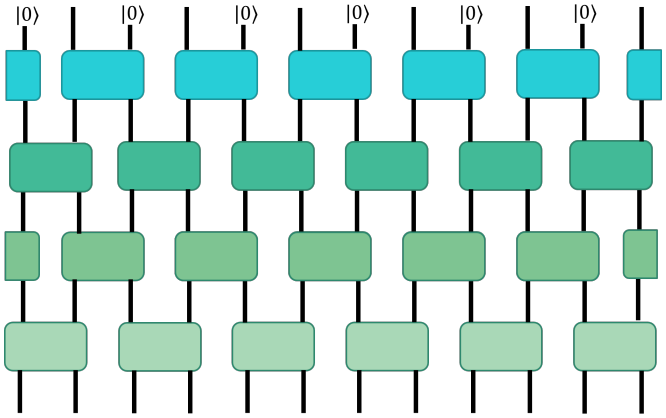


Figure 3.8: One DMERA layer with depth $D = 4$. The wires are qubits and the tensors are two-qubit quantum gates. At each isometry (blue) a new qubit in the $|0\rangle$-state is inserted.

such that $U_i' : \mathcal{H}_i \mapsto \mathcal{H}_{i+1} = \mathcal{H}_i^{\otimes 2}$ to get

$$|\psi_{i+1}\rangle = U_i' |\psi_i\rangle . \tag{3.24}$$

All layers U_i are constructed from the same set of two-qubit unitaries $\{w, u_1, u_2, ..., u_{D-1}\}$, where the w-unitaries are applied first to ψ_i (interspersed with $|0\rangle$), then the u_1-unitaries and so on, in a brick wall structure, see Figure 3.8.

So far, we have been more or less repeating the definition of MERA from Section 3.1.1. Many properties and tools that apply to DMERA can be copied directly from MERA, such as translation and scale invariance or evaluation of observables. However, there are some subtle differences. For example, we consider DMERA to contain unitaries instead of tensors because we exclusively use two-qubit unitary gates. Another difference is that we explicitly introduce a qubit in the $|0\rangle$-state at every isometry, instead of letting the isometry be general.

Depth

However, there is one major difference: DMERA is allowed to have more than two unitary layers in each U_i. This is necessary as the DMERA works with qubits, which have a fixed local dimension $d = 2$. We cannot group the qubits together to increase the bond dimension as was the case for MERA, since DMERA needs to be a valid quantum circuit to work on a quantum computer. Nonetheless, the extra unitary layers in U_i allows us to increase the bond dimension and accuracy of DMERA.

We will now show that any DMERA can be formulated as a MERA, and vice versa. Consider a DMERA with n qubits and a depth $D = 2$. This simply corresponds to a MERA with $d = 2$. Now consider a DMERA with depth $D = 3$. To relate this to a MERA, let us group the unitaries $\{w, u_1, u_2\}$ together such that we get new effective unitaries $\{w^*, u^*\}$, see Figure 3.9 [17]. These unitaries still follow the MERA definition from Eq. 3.1 and 3.2,

$$w^* : \mathbb{C}^4 \mapsto (\mathbb{C}^4)^{\otimes 2}, \qquad u^* : (\mathbb{C}^4)^{\otimes 2} \mapsto (\mathbb{C}^4)^{\otimes 2},$$

so they are a valid set of tensors. This now allows us to write down a MERA with $d = 4$. Similarly, we can do this for any D to obtain a MERA with $d = 2^{D-1}$ by continuing this "triangular" grouping. By going the opposite way a MERA with $d = 2^{D-1}$ can be formulated as a DMERA with depth D.

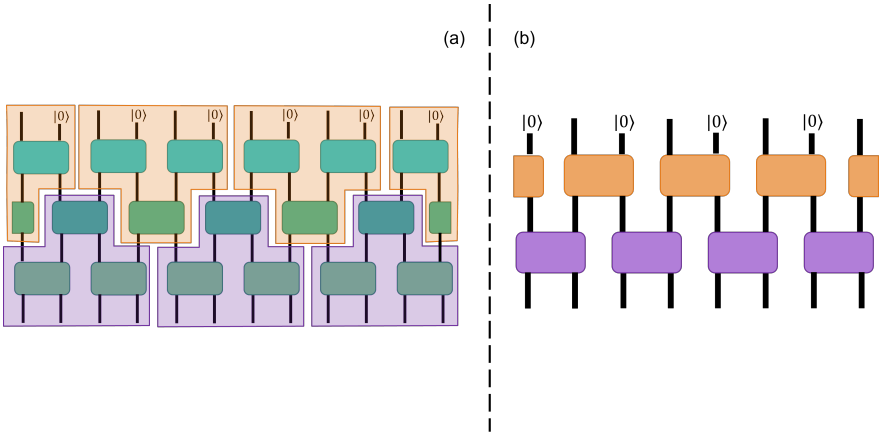


Figure 3.9: Structure with 3 layers transformed to an ordinary MERA with increased local dimension

3.2.2 Causal cone

We have discussed the causal cone for MERA in Section 3.1.3 where we have seen that the causal cone of an operator O can have a fixed width. The addition of more unitary layers means that the fixed cone widths we derived for MERA generally do not apply to DMERA. Instead, we will also need to take the depth D into account. Similarly in the earlier derivation, we will look at the effect of one $U^\dagger$ transformation on an n-qubit operator with odd n and later we will generalize to even n. Note that the case of $D = 2$ is trivial because it is the same for MERA.

For odd n, an operator O connects to $\lceil \frac{n}{2} \rceil$ $u_2^\dagger$-unitaries, then to $(\lceil \frac{n}{2} \rceil + 1)$ $u_1^\dagger$-unitaries and finally to $(\lceil \frac{n}{2} \rceil + 2)$ $w^\dagger$-unitaries. At the end of the $U^\dagger$ transformation we have mapped an n-qubit operator to an $(\lceil \frac{n}{2} \rceil + 2)$-qubit operator, disregarding any unitaries that would contract to identity like before. For $D = 3$, we can see that a 1-qubit operator has a causal cone width of 3 qubits, a 3-qubit operator a cone width of 4 qubits, a 5-qubit operator a cone width of 5 qubits and a 7-qubit operator a cone width of 6 qubits. The new fixed width of a causal cone is now 5 qubits which differs from the previously found values for MERA.

At this point we can already generalize the result by considering the effect of the extra layer in the $D = 3$ example. The extra layer adds one extra qubit to the cone width because q entanglers always connect to $q + 1$ unitaries from below due to the brick wall structure. For a given D and odd n the width of the causal cone is transformed as

$$n \to \left\lceil \frac{n}{2} \right\rceil + D - 1, \tag{3.25}$$

and for even n,

$$n \to \frac{n}{2} + D - 1 \quad \text{(aligned)} \tag{3.26}$$

$$n \to \frac{n}{2} + D \quad \text{(disaligned)}, \tag{3.27}$$

where observables can be aligned or disaligned, c.f. Section 3.1.3.

We are particularly interested in the fixed widths since this allows us to formulate the DMERA transformation as a quantum channel which transforms an n-qubit operator to an

n-qubit operator. Recall that

$$\langle\psi_{i+1}|\, O\, |\psi_{i+1}\rangle = (\langle\psi_i| \otimes \langle 0^{2^i}|)U^\dagger O\, U(|\psi_i\rangle \otimes |0^{2^i}\rangle), \tag{3.28}$$

then we can identify a quantum channel Φ_i such that

$$(\langle\psi_i| \otimes \langle 0^{2^i}|)U^\dagger O\, U(|\psi_i\rangle \otimes |0^{2^i}\rangle) = \langle\psi_i|\, \Phi_i[O]\, |\psi_i\rangle\,, \tag{3.29}$$

where

$$\Phi_i[O] = \langle 0^{2^i}|\, U^\dagger O\, U\, |0^{2^i}\rangle\,, \tag{3.30}$$

for any operator O.

We see that Φ_i has the same function as the scaling operator $\mathcal{S}$ from Section 3.1.5. Similarly, we will only consider cases where Φ_i acts on operators O whose support is equal to the fixed width of the causal cone of Φ_i. In other words, $\Phi_i[O]$ and O must have support on the same number of qubits. Therefore, we can drop the index i for simplicity.

3.2.3 Noise resilience

In this section we will investigate the noise resilience of DMERA in two ways; first we will prove that the noise generated by a noisy DMERA channel $\tilde{\Phi}$ after n layers is upper bounded by ν, such that $\mu < \nu < 1$ where μ is the second-largest eigenvalue, and the depth D, that is

$$\left\|\tilde{\Phi}^n - \Phi^n\right\|_\infty \leq \frac{\mathcal{O}(\epsilon D^2)}{1-\nu}, \tag{3.31}$$

and secondly we will investigate experimentally how an early error dissipates with consecutive noiseless layers. The first proof is based on the derivation from Kim-Swingle, supported by results from Wolf [1, 30].

Mathematical argument

We assume that the DMERA channel Φ has only one eigenvalue λ such that $|\lambda| = 1$. Furthermore, we assume that the noisy channel $\tilde{\Phi}$ is unital but can have more eigenvalues with modulus 1.

In this section we will make use of the *induced operator norm* which is given by

$$\|\Phi\|_{\infty\to\infty} = \max_{\|O\|_\infty=1} \|\Phi[O]\|_\infty\,; \qquad \|O\|_\infty = \sup_{\psi,\phi;\, \langle\psi|\phi\rangle=1} |\,\langle\psi|\, O\, |\phi\rangle\,|, \tag{3.32}$$

for any quantum channel Φ. For the rest of this section we will only use the induced operator norm, so for simplicity we will write $\|\Phi\|_{\infty\to\infty}$ as $\|\Phi\|_\infty$.

To formally analyze the noise resilience, we will first need to introduce the noisy version of Φ, namely $\tilde{\Phi}$. The noisy counterpart is constructed with noisy gates which have some error given by

$$\|\Gamma - \tilde{\Gamma}\|_\infty \leq \epsilon, \tag{3.33}$$

where Γ is a two-qubit gate and ϵ is the error rate.

We would like to express the error between Φ and $\tilde{\Phi}$. We do so by replacing a noiseless gate with a noisy gate and observe the effect on the error. Intuitively, for an N-gate quantum circuit we might assume that we need to apply Eq. 3.33 for each gate we replace. As we will see, this intuitive idea is correct.

First, let us examine a simple circuit with 2 consecutive gates, namely $\|\Gamma_2\Gamma_1 - \tilde{\Gamma}_2\tilde{\Gamma}_1\|$, and express the error rate as

$$\begin{aligned}\|\Gamma_2\Gamma_1 - \tilde{\Gamma}_2\tilde{\Gamma}_1\| &= \|\Gamma_2\Gamma_1 - \tilde{\Gamma}_2\Gamma_1 - (\tilde{\Gamma}_2\tilde{\Gamma}_1 - \tilde{\Gamma}_2\Gamma_1)\| \\ &= \|(\Gamma_2 - \tilde{\Gamma}_2)\Gamma_1 - \tilde{\Gamma}_2(\tilde{\Gamma}_1 - \Gamma_1)\| \\ &\leq \|\Gamma_2 - \tilde{\Gamma}_2\| \cdot \|\Gamma_1\| + \|\tilde{\Gamma}_2\| \cdot \|\Gamma_1 - \tilde{\Gamma}_1\| \\ &\leq 2\epsilon, \end{aligned} \tag{3.34}$$

where in the first line we effectively added 0, from the second to the third line we used the triangle inequality and finally we used Eq. 3.33. Similar arguments can be used to show that $\|\Gamma_2 \otimes \Gamma_1 - \tilde{\Gamma}_2 \otimes \tilde{\Gamma}_1\| \leq 2\epsilon$. For a circuit with N consecutive gates we can write the error rate as

$$\begin{aligned}\|\Gamma_N \cdots \Gamma_1 - \tilde{\Gamma}_N \cdots \tilde{\Gamma}_1\| &= \|\Gamma_N \cdots \Gamma_1 - \tilde{\Gamma}_N\Gamma_{N-1} \cdots \Gamma_1 - (\tilde{\Gamma}_N \cdots \tilde{\Gamma}_1 - \tilde{\Gamma}_N\Gamma_{N-1} \cdots \Gamma_1)\| \\ &= \|(\Gamma_N - \tilde{\Gamma}_N)\Gamma_{N-1} \cdots \Gamma_1 - \tilde{\Gamma}_N(\tilde{\Gamma}_{N-1} \cdots \tilde{\Gamma}_1 - \Gamma_{N-1} \cdots \Gamma_1)\| \\ &\leq \epsilon + \|\Gamma_{N-1} \cdots \Gamma_1 - \tilde{\Gamma}_{N-1} \cdots \tilde{\Gamma}_1\| \\ &\leq \epsilon + \epsilon(N-1) \\ &= \epsilon N. \end{aligned} \tag{3.35}$$

Then, by induction it holds that

$$||\,\Phi - \tilde{\Phi}\,||_\infty \leq \mathcal{O}(\epsilon D^2), \tag{3.36}$$

as the DMERA channel Φ contains D layers and $\mathcal{O}(D)$ gates within each layer, so in total Φ holds $\mathcal{O}(D^2)$ gates.

As we are particularly interested in the decaying property of Φ, we are free to eliminate the stationary part of operator O, i.e. by removing the identity operator which corresponds to the largest eigenvalue of 1. Let us define the traceless operator $\bar{O} = O - \frac{\mathrm{Tr}(O)}{d} I$. Using this operator, we can write

$$\tilde{\Phi}^n[\bar{O}] - \Phi^n[\bar{O}] = \tilde{\Phi}^n[O] - \Phi^n[O]. \tag{3.37}$$

Let us introduce $\Phi_{(1)}$ which is defined as

$$\Phi_{(1)} = \sum_{k:|\lambda_k|=1} \lambda_k P_k, \tag{3.38}$$

where P_k is the projection for λ_k (Eq. 6.13 from [30]). Given that Φ has only one eigenvalue with modulus 1, then there is only one $P_k = cI$ and $\Phi_{(1)}$ projects any operator onto the identity operator, i.e. $\Phi_{(1)}[O] = cI$. Using this projector and the traceless operator $\bar{O}$ we can write

$$\Phi^n[\bar{O}] = \Phi^n[O] - \Phi^n_{(1)}[O], \tag{3.39}$$

which has a spectral radius of $\mu = \max_{\lambda \in \mathrm{spec}(\Phi)}\{|\lambda| < 1\}$. In other words, μ is the second-largest eigenvalue of Φ.

Now that we have Eq. 3.39 we can give an upper bound on $||\Psi^n||_\infty$. Using Theorem 8.23 from [30] we know that for any norm

$$||\Phi^n - \Phi^n_{(1)}|| \leq C\mu^n n^{d_\mu - 1}, \tag{3.40}$$

where C is a constant independent of n and μ, and d_μ is the size of Jordan block corresponding to μ. In general, we do not know the size of the Jordan block of an arbitrary eigenvalue of our

channel, and especially not the Jordan block corresponding to μ. However, for any $\mu < \nu < 1$ there is an n_0 such that $\nu^n \geq \mu^n n^{d-1}$ for $n \geq n_0$. Indeed,

$$\begin{aligned} & \nu^n \geq \mu^n n^{d-1} \\ \iff & n \log \nu \geq n \log \mu + (d-1) \log n \\ \iff & n \log \frac{\nu}{\mu} \geq (d-1) \log n \\ \iff & \frac{n}{\log n} \geq \frac{d-1}{\log \frac{\nu}{\mu}}. \end{aligned} \tag{3.41}$$

Therefore, for $\mu < \nu < 1$ we can write

$$||\Phi^n - \Phi^n_{(1)}|| = \mathcal{O}(\nu^n). \tag{3.42}$$

Using the telescopic decomposition, we can write $\tilde{\Phi}^n - \Phi^n$ as

$$\tilde{\Phi}^n - \Phi^n = \sum_{i=1}^{n} \tilde{\Phi}^{i-1} \circ (\tilde{\Phi} - \Phi) \circ \Phi^{n-i}. \tag{3.43}$$

For the proof and discussion of the telescopic decomposition, see Appendix A.2.

We can find the upper bound of the error after n layers as follows,

$$\begin{aligned} \left\| \tilde{\Phi}^n - \Phi^n \right\|_\infty &= \left\| \sum_{k=0}^{n} \tilde{\Phi}^{k-1} \circ (\tilde{\Phi} - \Phi) \circ \Phi^{n-k} \right\|_\infty \\ &\leq \sum_{k=0}^{n} ||\tilde{\Phi}^{k-1}||_\infty \cdot ||\tilde{\Phi} - \Phi||_\infty \cdot ||\Phi^{n-k} - \Phi^{n-k}_{(1)}||_\infty \\ &\leq \sum_{k=0}^{n} ||\tilde{\Phi} - \Phi||_\infty \cdot \nu^{n-k} \\ &\leq \mathcal{O}(\epsilon D^2) \sum_{k=0}^{n} \nu^{n-k} \\ &\leq \frac{\mathcal{O}(\epsilon D^2)}{1-\nu}. \end{aligned} \tag{3.44}$$

From the second to the third line, we used that $\tilde{\Phi}^{k-1}$ is norm-nonincreasing. Next, we used Eq. 3.36 and by using Eq. 3.39 we can apply the upper bound from Eq. 3.42. Finally, we notice that the sum is upper bounded by the geometric series of ν.

The result from Eq. 3.44 tells us that the upper bound of error on observables is independent of n, the number of DMERA layers. Additionally, we expect the error to decrease with additional DMERA layers, since we discard qubits that have errors in them and introduce new qubits at each layer. Therefore, we should expect that increasing the number of layers is beneficial to estimating the ground state energy, based on the argument of both the scaling dimensions analysis and the previously found upper bound. Another interesting consequence of this result is that if there exists a noiseless DMERA that can approximate the ground state energy then there exists a noisy DMERA that approximates the ground state energy up to an error of $\frac{\mathcal{O}(\epsilon D^2)}{1-\nu}$.

Experimental argument

Now that we have proven an upper bound on the noise, let us consider a different argument for the noise resilience of DMERA. With each DMERA layer, or channel iteration, we introduce pure qubits and throw away qubits that potentially contain errors. As a result, the errors cannot grow beyond a certain size, which is exactly the result derived in Eq. 3.44. Instead of considering many (noisy) layers, let us see what happens with only one noisy layer followed by several noiseless layers. In Figure 3.10 we can see the effect of the error from a single noisy channel iteration on the ground state energy after several iterations of the noiseless channel. The difference between the measured ground state energy $\tilde{E}$ and the exact ground state energy E_0 converges exponentially for all noise levels. From this we can see that noise tends to spread out and get eliminated in further layers. The two-qubit gate infidelity noise model, described later in Section 4.1.2, was used with various levels of noise.

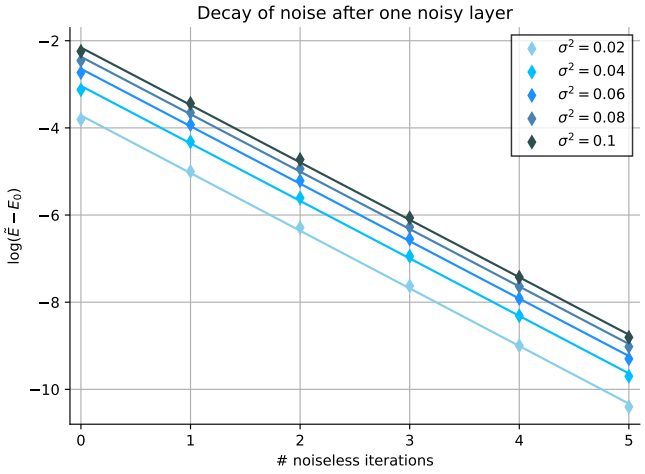


Figure 3.10: Decay of noise after an initial noisy layer followed by noiseless layers. This experiment was performed for several levels of noise indicated by the noise parameter σ^2.

Chapter 4

Implementing DMERA

We have laid the groundwork for implementing DMERA on a quantum computer in the previous section. What is left is to discuss the practical side of implementing DMERA by for example discussing the specific quantum gates that will be used. To test the implementation, we verify the performance by computing the energy of the prepared states and comparing the results to those of Sewell and Jordan [2].

For this thesis, we have chosen the trapped ion QC developed by IonQ which currently contains 11 qubits [31]. The access to this QC is facilitated by the Amazon Braket service. This particular QC has the advantage that for all qubits the fidelity is relatively high compared to its competitors. In addition, the QC has all-to-all connectivity so any two qubits can interact with each other without the use of SWAP gates.

In a recent study in 2021, Sewell and Jordan successfully managed to implement DMERA to prepare the ground state of the quantum Ising chain model on the Honeywell HØ trapped ion QC [2]. The machine has six qubits that are also all connected. The advantage of this computer over the IonQ machine is that it allows for mid-circuit qubit resets. Any qubit that has been used earlier in the quantum circuit can be reset and used again in the computation which is extremely useful for DMERA implementations as old qubits can be reset and inserted at a new layer. Mid-circuit resets thus allow for an arbitrary number of layers. In this thesis we will adapt their DMERA implementation and compare our obtained results with their results.

In this chapter, we will discuss the practicalities of running DMERA on both a simulated and real quantum computer, such as compiling a DMERA circuit in terms of native gates before performing the circuit on a quantum computer. We then measure the energy of the states that were prepared by circuits with various number of layers to test the performance of DMERA.

4.1 Practicalities

4.1.1 Native gates

Quantum computers have a preference to work with so-called native gates. These are the basic operations that a quantum computer uses, depending on its type of design. Any unitary that is given to a quantum computer will first be compiled to a series of native gates. To avoid any unnecessary errors, we will want to present a DMERA circuit to the computer that is already comprised of native gates. Our circuit only consists of the w and u unitaries, so we will want to decompose them in terms of native gates (for now we do not consider the observable as part of the circuit). The two native gates we will use for this is the Mølmer-Sørensen XX gate (MS)

and the Z-axis rotation gate (R_z) as they are used on the IonQ quantum computer [31].

The R_z gate is defined as

$$\begin{aligned} R_z(\theta) &= e^{-i\frac{\theta}{2}Z} \\ &= \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \end{aligned} \tag{4.1}$$

where Z is the Pauli Z spin operator. R_z rotates a single qubit around the z-axis. On a quantum computer this operation is particularly useful since it does not actually have to be performed. Instead, it can be performed virtually by updating the angles of the following gates, which is equivalent to rotating the qubit [32]. For this reason the R_z gate adds practically no runtime or errors.

The MS gate is a two-qubit gate that can be implemented particularly well on an ion-trap quantum computer and is shown to allow for efficient construction of quantum circuits [33]. The XX MS gate, also known as the Ising XX coupling gate, can be defined as

$$\begin{aligned} XX_{mn}(\theta) &= e^{-i\frac{\theta}{2}X_mX_n} \\ &= \begin{pmatrix} \cos\left(\frac{\theta}{2}\right) & 0 & 0 & -i\sin\left(\frac{\theta}{2}\right) \\ 0 & \cos\left(\frac{\theta}{2}\right) & -i\sin\left(\frac{\theta}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\theta}{2}\right) & \cos\left(\frac{\theta}{2}\right) & 0 \\ -i\sin\left(\frac{\theta}{2}\right) & 0 & 0 & \cos\left(\frac{\theta}{2}\right) \end{pmatrix} \end{aligned} \tag{4.2}$$

where X_m is the Pauli X spin operator on the m-th qubit. X_m and X_n commute with each other, so XX_{mn} is invariant under interchange of qubits m and n.

With these two native gates, we can construct a native gate decomposition for the DMERA gates w and u, as described in [2]:

$$\begin{aligned} w(\theta) &= XX(\pi/2) \cdot R_{z,1}(\theta) \cdot R_{z,2}(\theta - \pi/2) \cdot XX(-\pi/2) \\ u(\theta) &= XX(\pi/2) \cdot R_{z,1}(\theta) \cdot R_{z,2}(\theta) \cdot XX(-\pi/2), \end{aligned} \tag{4.3}$$

where $R_{z,n}$ acts on the n-th qubit. Note that the angle dependence resides only in the rotation gate which is implemented as a virtual operation.

Sewell and Jordan researched two different native gate decompositions, namely Eq. 4.3 and another with angle dependence in the XX gate [2]. The latter one was found to be more susceptible to noise than the former one, hence Eq. 4.3 has been chosen for this thesis.

4.1.2 Noise models

Before we discuss how to simulate DMERA on a classical computer, we must first consider the types of noise that are usually present in quantum computations. In this thesis we consider two sources of noise: two-qubit gate infidelity and spontaneous depolarization. In this section, we will discuss what these sources are and how they can be modeled.

Two-qubit gate infidelity noise

Two-qubit gate infidelity is the imprecise implementation of a two-qubit gate on the quantum computer. Generally, this error occurs due to imprecise angles of the gate. For this reason, Sewell and Jordan modeled gate infidelity in their experiment as imprecise rotation angles of the XX gates given by a Gaussian distribution with standard deviation σ [2]. We use σ as a parameter to indicate the level of noise for two-qubit gate infidelity.

The noisy version of the XX gate is denoted as $\widetilde{XX}$ and the Kraus operators, which will be given meaning in Section 4.1.3, are given by

$$K_0 = \sqrt{\frac{1+e^{-\sigma^2/2}}{2}} XX(\theta) \qquad K_1 = \sqrt{\frac{1-e^{-\sigma^2/2}}{2}} XX(\theta-\pi). \tag{4.4}$$

The $\widetilde{XX}$ gate is equivalent to applying an ideal XX gate with a probability of $(1-e^{-\sigma^2/2})/2$ to flip both qubits afterwards. The gate infidelity noise is correlated between the two qubits.

Depolarizing noise

Another source of error we have to deal with is spontaneous errors due to the environment (e.g. cosmic rays, external electromagnetic fields, internal particle interactions). When a qubit interacts with the environment in these manners, the qubit may become depolarized with probability p. We can model spontaneous noise by applying the depolarizing channel $F_p[\rho]$, given by

$$\begin{aligned} F_p[\rho] &= (1-p)\rho + \frac{p}{3}(X\rho X + Y\rho Y + Z\rho Z) \\ &= \left(1 - \frac{4p}{3}\right)\rho + \frac{2p}{3} I, \end{aligned} \tag{4.5}$$

to each qubit before and after applying an XX gate. In this way, the noise is uncorrelated between the qubits. Note that it is not necessary to apply the depolarizing channel before and after a phase rotation gate as this is implemented virtually. The parameter p will be used to indicate the level of noise for depolarizing error.

4.1.3 Classical simulation of DMERA

Classical simulation can be used to predict what should happen on a quantum computer and also to predict what will realistically happen. Simulating DMERA without any added noise gives us a prediction how well the method can approach the desired ground state in an ideal scenario. However, on a NISQ-era quantum computer we will always have to deal with noise, so ideally we will also want to simulate noisy DMERA circuits. In the previous section (Section 4.1.2) we introduced two noise models which we can now use to simulate noise on a classical computer.

The advantages of using classical simulations is that one has access to more information that is normally not available on a quantum computer, such as the matrix structure of the DMERA channel or mid-circuit probing of the qubits. This information is useful to accurately implement DMERA in experiments. Another benefit is that the classical simulation used in this thesis is deterministic and thus the end results do not change with successive trials due to randomness as is the case for quantum computers. All these benefits make it so that classical simulations have been an essential part of this research.

In this section we will discuss the procedure of simulating DMERA on a classical computer using Kraus operators and how this method can be used to introduce noise in the simulation. Lastly, we introduce an alternative method to Kraus operators by using an efficient tensor contraction method that can be used to evaluate noiseless DMERAs.

Kraus operators

In general, we will perform a DMERA layer by doing each unitary layer separately. Any completely positive trace-preserving map can be written in terms of its Kraus operators X_i

such that

$$\Omega[M] = \sum_{i}^{r} X_i M X_i^\dagger, \tag{4.6}$$

for all $M \in L(\mathcal{H})$ [21]. For a 3-qubit state ρ, we can then compute the isometries layer as

$$w \otimes w \otimes w[\rho] = \sum_{i,j,k} (X_i \otimes X_j \otimes X_k)\rho(X_i \otimes X_j \otimes X_k)^\dagger, \tag{4.7}$$

where X are the Kraus operators of w. We can generalize this statement to N qubits by allowing N isometries.

For the simplest case, namely the noiseless case with only 1 Kraus operator for w and u, we use the matrices

$$w(\theta) = \begin{pmatrix} \cos\left(\theta - \frac{\pi}{4}\right) & 0 & 0 & \sin\left(\theta - \frac{\pi}{4}\right) \\ 0 & \frac{1}{2}\sqrt{2} & -\frac{1}{2}\sqrt{2} & 0 \\ 0 & \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} & 0 \\ -\sin\left(\theta - \frac{\pi}{4}\right) & 0 & 0 & \cos\left(\theta - \frac{\pi}{4}\right) \end{pmatrix} \qquad u(\theta) = \begin{pmatrix} \cos(\theta) & 0 & 0 & \sin(\theta) \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin(\theta) & 0 & 0 & \cos(\theta) \end{pmatrix}, \tag{4.8}$$

given in [17]. For $D = 2$ there are only two unitary layers formed by $w(\theta_1)$ and $u(\theta_2)$ inside each DMERA layer where $\theta_1 = \frac{\pi}{12}$ and $\theta_2 = -\frac{\pi}{6}$. For $D = 4$ we instead have four unitary layers consisting of $w(\theta_1), u(\theta_2), u(\theta_3)$ and $u(\theta_4)$. The angles $\theta_1, \theta_2, \theta_3, \theta_4$ are derived in the paper by Evenbly and White [17].

To add two-qubit gate infidelity from Section 4.1.2 we use the Kraus operators and the native gate decomposition given by Sewell and Jordan [2]. The gate XX has two Kraus operators K_0 and K_1 so from Eq. 4.3 we know that each w and u has four Kraus operators, namely two for each appearance of XX. In total, we have four Kraus operators per w and u gate and Eq. 4.7 still applies.

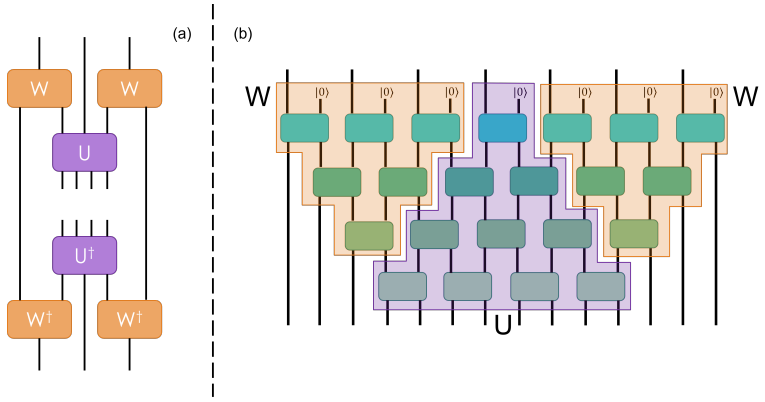


Figure 4.1: (a) The general contraction scheme for any depth DMERA. Note that not all legs need have the same dimensions. This scheme can be used for both left and right alignment by contracting the appropriate leg between U and $U^\dagger$. (b) Example for $D = 4$ of construction blocks used in the scheme.

The Kraus operator method is computationally intensive as there are a lot of matrix multiplications involved. Specifically for the noiseless channel there is an alternative numerical method that we can use which is more optimized and much faster to compute. Due to the

fact that we are dealing with tensors, we can use e.g. the `einsum()` function from the *numpy* Python package [34]. This specific function is highly optimized to work with tensors and can efficiently contract them.

Additionally, we can construct a general contraction scheme that works for any depth D. For a DMERA with any depth $D > 1$ we can group the tensors in such a way that we can form blocks that fit the scheme shown in Figure 4.1.

The classical simulation code we created and used for this thesis can be found on GitHub [35].

4.1.4 Algorithmic cooling

The quantum computer that we have used in this project is the IonQ quantum computer which has 11 qubits (at the time of writing). For the simplest case of DMERA, namely depth 2, we need three new qubits for each layer to insert at the isometries. As we also need three qubits to start with, we need $3(L+1)$ qubits if we want to perform L DMERA layers. With 11 qubits we are limited to two layers, however if we had one more qubit we would be able to do one additional layer for a total of three layers.

A class of methods exist called *algorithmic cooling* which allows one to cool one qubit by heating up other qubits [36, 37]. We can understand cooling as making the qubit more likely to be in one state than in the other state. Heating up is then the opposite, that is making the state more mixed. Alternatively, we can view algorithmic cooling as a transfer of entropy where the entropy flows from one qubit to other qubits such that the targeted qubit has a lower entropy in the end.

For algorithmic cooling to work we assume that all qubits involved in the cooling process start with a bias towards $|0\rangle$, i.e. $P(0) = \frac{1+\epsilon}{2}$ and $P(1) = \frac{1-\epsilon}{2}$, where ϵ is the polarization bias. If a qubit has a starting bias towards $|1\rangle$ then we can simply flip the spin so that the bias is now towards $|0\rangle$. In general the bias ϵ will not be the same for all qubits, so instead we can use that qubit A has bias ϵ_A, qubit B has bias ϵ_B and so on.

Note that a quantum computer is a closed system of qubits so we cannot transfer heat to the outside. This eliminates several choices of algorithmic cooling which make use of a heat bath. Using a heat bath would allow us to cool a qubit to an arbitrary temperature, but a closed system with a fixed amount of qubits naturally defines a limit of how much a qubit can be cooled. In an ideal scenario we approach the cooling limit, but in practice this turns out to be difficult to achieve.

3B-Comp

As we have no access to a heat bath, we have to find a different method. What we can do is swap probabilities of states around such that the targeted qubit becomes colder. A method we can use for 3 qubits is called 3B-Comp [37]. This method performs the swap $|011\rangle \leftrightarrow |100\rangle$ and leaves the other states unchanged. The swap $|011\rangle \leftrightarrow |100\rangle$ is the only swap that would increase the bias ϵ of the first qubit, as any other swap would either reduce or not change the bias, given that all three qubits have a bias towards $|0\rangle$. We can implement the swap with the small circuit shown in Figure 4.2 which involves the use of a CNOT, X and CSWAP gate. The CSWAP gate swaps two target qubits if the control bit is 1, else it does nothing.

The 3B-Comp method works for only three qubits, however we can use it for more than three qubits by repeatedly applying the swap for different sets of three qubits. Suppose we have five qubits A, B, C, D and E with equal bias and we want to cooldown qubit A. We can perform the first swap with qubits A, B and C and the second swap with qubits A, D and E. Assuming that all qubits have a bias towards $|0\rangle$ qubit A should be cooled down further due to the extra iteration of 3B-Comp than with a single iteration of 3B-Comp.

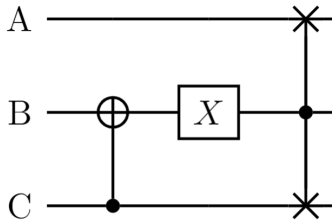


Figure 4.2: The 3B-Comp circuit. This effectively swaps the states $|011\rangle$ and $|100\rangle$ and leaves the other states intact. In this configuration qubit A is cooled down and thus its bias towards $|0\rangle$ is increased while qubits B and C are heated up.

Now that we have introduced a simple method for three qubits, let us put this to practice. After we have done two DMERA layers, we have used 9 from the available 11 qubits of the IonQ QC. 3 of the used qubits are kept in the computation and 3 additional qubits need to be inserted into the circuit as pure $|0\rangle$. As we only have 2 qubits left over, we need to cool one previously used qubit back to $|0\rangle$ for an additional layer. By using 3B-Comp we can attempt to cooldown one single qubit by using the other qubits that are left over after the other layers.

Timing of cooling

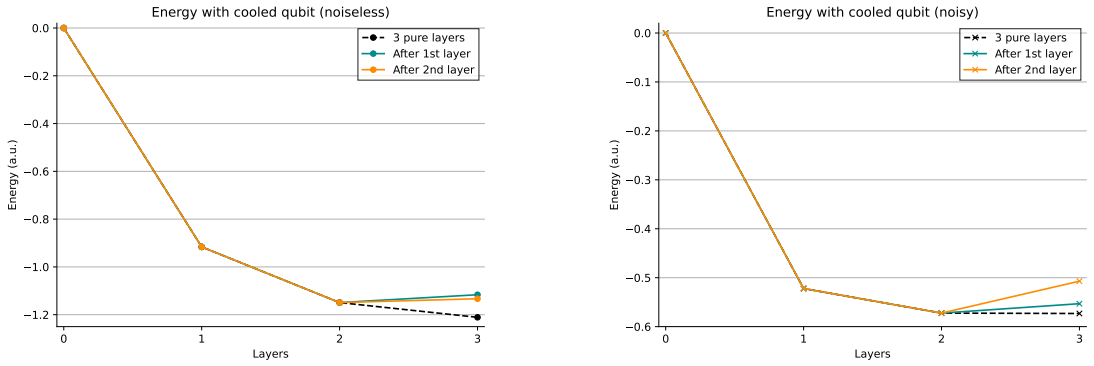
The first choice to consider, is whether the cooling should take place after the first or the second layer. After the first layer means that there are only three qubits available for algorithmic cooling which limits the number of times we can iterate 3B-Comp. However, it would allow us to insert the cooled qubit back in already at the second layer. Inserting an impure qubit at the new layer introduces noise, however this may be balanced out by the noise resilience of DMERA. Additionally, using only three qubits also uses fewer gates meaning there is less error due to gate infidelity.

On the other hand, cooling after the second layer means there are more qubits available and thus we can construct a larger circuit that can cool down one qubit further than what is possible with only three qubits. In an ideal scenario, a larger circuit would be able to cool more efficiently, but in a noisy environment it might work reversed; a large circuit could generate more noise and in the end the qubit would be less cool in comparison.

To investigate this issue of cooling timing, we will test several scenarios. First, we cool one qubit after the first layer for both the noiseless and noisy case. Then, we do the same but with the qubit cooled after the second layer. In both noiseless and noisy cases, we only have to use the impure qubit for a DMERA circuit with three layers. For DMERA with less than three layers we have enough pure qubits to use, so the difference in cooling timing will only be apparent for three layers. We compare the results with three layers that all have pure qubits.

The results are shown in Figure 4.3. In general, the energies perfectly overlap for the first two layers as no impure qubits are used yet. When preparing the circuit with three layers we introduce an impure qubit halfway through the circuit depending on the scenario and measure the energy only after the third layer. For the noiseless case, in Figure 4.3a we see that it is energetically favorable to cool the qubit after two layers. We argued before that this would be the case as there are more qubits available to cool with. In a noisy environment, the opposite is true, as can be seen in Figure 4.3b. Running a large cooling circuit is prone to more noise than a smaller cooling circuit, thus it is better to cool after one layer.

All scenarios in Figure 4.3 show a common theme. The energy of the third layer is never lower than the energy of the second layer, regardless of the timing of the cooling algorithm.



(a) Cooling with no noise.

(b) Cooling with depolarizing noise $p = 0.05$.

Figure 4.3: The energy after three layers with a qubit that is cooled after the first or second layer.

We expect an exponential decay in the energy with pure qubits, so for algorithmic cooling to be of use we need to see the same behavior when using cooled qubits. If the energy does not lower after cooling, we are better off not doing the cooling at all, as it would interfere with the decaying behavior of DMERA. As this simple experiment shows, it is not beneficial for us to cool a qubit to perform an extra layer.

There are several reasons why algorithmic cooling does not give the desired results. Firstly, we are very limited in the number of qubits available for the cooling process. This means we are also limited in the number of times we can do 3B-Comp or exchange probabilities in any other manner. Eventually, the qubits besides the targeted qubit are too hot and we hit a natural limit of cooling. Additionally, we do not have access to a heat bath. This means that our circuit, or quantum computer, is a closed system and we cannot exchange heat with the outside. If we did have a heat bath, there would be more possibilities for cooling algorithms.

It is also possible that we did not find the optimal cooling circuit or algorithm for the specific situation. For the cooling algorithm, we have limited ourselves to using only 3B-Comp as this can be used in a general situation, which might also not be the optimal choice. We could implement a specific circuit that cools the targeted qubit down further, however we assumed that we do not know anything about the qubits except that they have a bias towards $|0\rangle$. This limits our approach to an algorithmic method, which we have used to no success.

4.2 Energies

One way to benchmark the implementation of DMERA is to compute the energy of the prepared state and compare the results with the true ground state energy. When the measured energy converges to the exact energy and follows the expected behavior, i.e. an exponential decay, we can be confident that the DMERA works as expected. In this section we will first discuss two general points on the Hamiltonian and left-right alignment of the circuit, then we will describe the method to find the ground state energy for both classical simulation and quantum computers and lastly we showcase the results of both methods.

Hamiltonian of model

The critical system of choice is the Ising quantum critical chain, which is a conformal field theory. Conformal field theories and its applications are outside the scope of this thesis, but the important property we will use is that such a theory is by definition invariant under angle-preserving transformations. We have already implicitly used this fact in Section 3.1.5.

The associated Hamiltonian for this critical Ising model is given by

$$H = -\sum_r X_r X_{r+1} + Z_r. \tag{4.9}$$

However, we will use the transformed Hamiltonian for a fully scale-invariant (D)MERA

$$H = \sum_r (X_r Z_{r+1} X_{r+2} - X_r X_{r+1}). \tag{4.10}$$

The detailed derivation and discussion of Eq. 4.10 is given by Evenbly and White [17]. Effectively, we only prepare three qubits (for depth 2) with DMERA to the ground state so the Hamiltonian reduces to

$$H = XZX - \frac{1}{2}(IXX + XXI), \tag{4.11}$$

where the factor $\frac{1}{2}$ accounts for the average of both XX terms. We will explicitly use this expression as the three-qubit observable to measure the energy of the prepared state.

Left and right centering

In Section 3.1.3 we briefly discussed the issue of left and right centered qubits where we had to average over both options to get the correct superoperator, that is $\mathcal{S} = \frac{1}{2}(\mathcal{S}_L + \mathcal{S}_R)$. This issue is also present for DMERA, so similarly we will need to average over both options in both the classical simulation and quantum computation by constructing a left and right channel, i.e. $\Phi = \frac{1}{2}(\Phi_L + \Phi_R)$, and process the data accordingly in further analysis.

4.2.1 Classical method

Using the methods described in Section 4.1.3 we can compute the expectation value of H to find the energy of the prepared state. Once a state ρ has been prepared by the DMERA circuit, the energy can be computed as

$$E = \langle H \rangle_\rho = \mathrm{Tr}[H\rho].$$

To account for the left and right centering of the channel, we compute both centerings of the channel. After each layer we take the average over the three remaining qubits for both centering which will then serve as the input for the next layer. Repeating this for all layers gives us the average over all possible configurations for DMERA.

The convergence of the ground state energy is shown in Figure 4.4 and the converged values in Table 4.1. After roughly five layers the starting state has almost converged to the fixed state in all cases and thus the energy will also not change much further. The exponentially decaying convergence occurs because, when viewed in the Heisenberg picture, the decay is characterized by the eigenoperators ϕ_a of Φ that are present in the Hamiltonian. All eigenoperators have an associated eigenvalue λ_a which is strictly less than 1 except for the fixed operator which has eigenvalue 1. When a layer is applied, all components gain a factor λ_a and after only a few layers the non-fixed eigenoperators vanish. This point has been discussed in more detail in Section 3.1.5.

The noise models show convergence after several layers, despite the potential concern that error builds up in large quantum circuits. This result indicates that DMERA is robust to noise and supports the findings from Section 3.2.3.

We can also see that a larger depth gives a closer approximation to the true ground state energy than a small depth. This result is not unexpected, as we have seen before that increasing the bond dimension of an MPS or MERA better approximates the ground state energy. Increasing the depth is an alternative way of increasing the bond dimension.

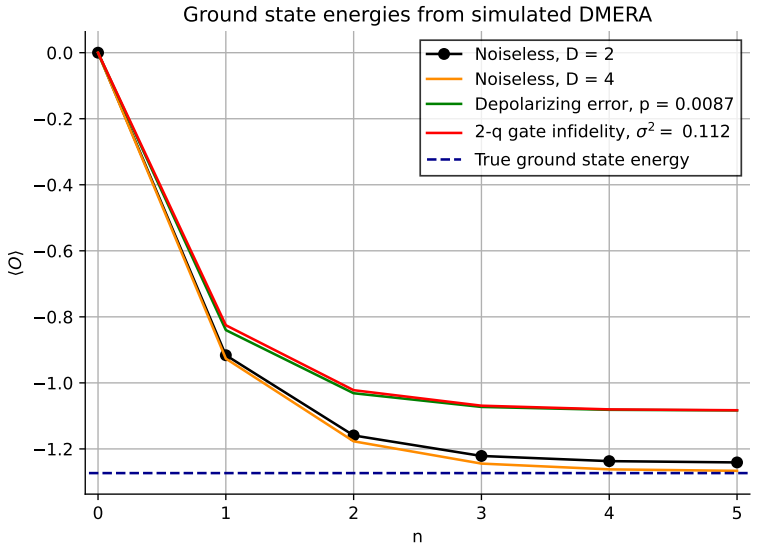


Figure 4.4: The decay of the ground state energies to the converged values. As more layers are applied to a starting state, the energy converges to a final value. This is shown for the noiseless case with $D = 2$ and $D = 4$ and with added depolarizing and gate infidelity noise.

	Exact	Noiseless (D = 2)	Noiseless (D = 4)	Depolarizing	Gate infidelity
E	-1.27324	-1.24223	-1.26784	-1.08375	-1.08360

Table 4.1: The converged energy values compared with the exact value $E_0 = -\frac{4}{\pi}$. The values have been computed after applying 20 DMERA layers to the starting state $\rho = |000\rangle\langle 000|$. The code used to compute and generate this figure can be found in [35].

4.2.2 Quantum method and experiment

The quantum computation is limited in its capabilities compared to the classical simulation. In actuality, the classical simulation can do non-physical things like measuring multiple observables one after each other. Nonetheless, the method to extract energies from the quantum computer is more involved than it is for classical simulation, so we will first need to describe this method.

Method

Each experiment run on the quantum computer consists of a number of shots. One shot is a single preparation and measurement of a state after going through the quantum circuit.

We can see this prepared state as a probability distribution for each basis state. Each shot might give a different value than the previous shot due to this distribution. To get an accurate estimation of the expectation value of an observable we want to do experiments with many shots such that the sample size is large enough.

We cannot implement the Hamiltonian from Eq. 4.11 as it is stated. Instead, we need to separate the Hamiltonian in the three observables XZX, IXX and XXI and measure each observable individually. The energy can then be calculated in classical post-processing by adding up the expectation values of each observable.

It is not possible to do the averaging over the left and right centering of the three qubits after a layer, but instead the quantum circuit needs to be constructed explicitly for either left or right centering. As there are two choices for each layer, we need to construct 2^N different circuits for a DMERA with N layers. In total, we need to perform 3×2^N different experiments and bring the results together in post-processing to find the expectation value of the full Hamiltonian.

The error margin can also be found indirectly from the experiment. First, we obtain the marginal probabilities of the three qubits that are measured from the data. We then artificially reproduce the experiment on a classical computer by randomly picking an outcome according to the obtained marginal probability distribution. All outcomes are stored to form a large set of "measurements" from which we can compute the standard deviation. This standard deviation is then the error margin for our results.

Experiment

The experiment was performed with the IonQ 11-qubit trapped ion quantum computer [31] which can be accessed through the Amazon Braket service [38]. In total, we have run 21 tasks with 1000 shots each: 3 tasks for 0 layers, 6 tasks for 1 layers and 12 tasks for 2 layers. The results from the tasks have been summed up to find the energy and averaged according to the number of possible left/right configurations.

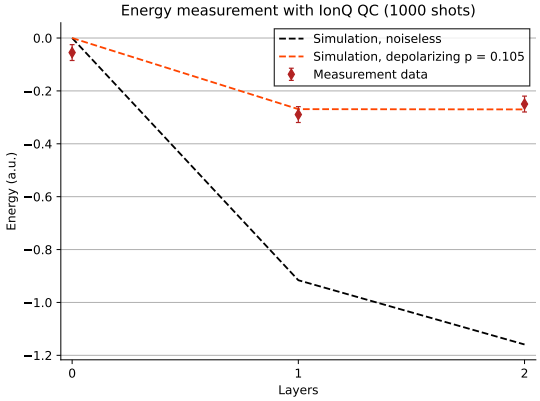


Figure 4.5: Energy results for two layers with standard deviation $\sigma = 0.03$. Data is from IonQ quantum computer with curves from simulations (noiseless and noisy). The code used to send the quantum tasks to the QC and analyze the data can be found in [35].

The results of the experiment are shown in Figure 4.5. The standard deviation is comparable for each data point and has a value of 0.03. Additionally, the energies as found by classical simulations for the noiseless case and the depolarizing noise case ($p = 0.105$) are shown. It is immediately clear that the results from the quantum computer differ significantly from the results for the noiseless case. The energy did not lower between 1 layer and 2 layers which does

not line up with our expectation of an exponential decay. However, we may not have performed enough shots or experiments to get precise measurement values.

The simulation needs a large amount of noise to fit the QC data. We have chosen depolarizing noise as the noise model as it is a commonly occurring error and the noise parameter that fits well to the data is $p = 0.105$. This value is much larger than the value that Sewell and Jordan used in their paper and that we have used in the classical simulations [2]. The choice of noise model is however not as relevant. Regardless of the chosen noise model, one would need to add a high amount of noise to fit the QC data.

As we are only able to perform experiments that are limited in terms of channel iterations and qubit usage, we refer to Sewell and Jordan for a more thorough and detailed experiment [2]. Their results show that DMERA works well on a real quantum computer and converges after only five layers (Figure 9 in their paper). They found an approximation of the ground state energy of $E = -1.08$ which is 15% higher than the true ground state energy.

Chapter 5

Extracting scaling dimensions

Throughout this thesis we have seen enough reasons to believe that DMERA is robust against noise, from a mathematical proof in Section 3.2.3 to a convergence of the ground state energy in Section 4.2. We can conclude that DMERA is an ansatz that can reliably prepare a ground state when implemented on a noisy quantum computer. Now that we have computed energies, we look to find and investigate other physical quantities. An interesting quantity is the so-called scaling dimensions, which are briefly discussed in both Kim-Swingle and Sewell-Jordan [1, 2].

At the foundation of preparing and converging to a ground state lie the eigenvalues of the DMERA channel Φ. The eigenvalues govern the decaying behavior to the ground state, which corresponds to the largest eigenvalue. These eigenvalues are interesting for investigation into DMERA, but they can additionally be linked with another physical quantity, namely the scaling dimensions. This quantity is of interest in the study of conformal field theories, which can describe critical models such as the quantum Ising chain that we have been using. A scaling dimension describes how the corresponding operator transforms under a scaling transformation, i.e. $x \to \alpha x$ for some scale factor α [39].

Each layer of DMERA in fact implements a scaling transformation, namely with a scaling factor $\alpha = 2$ as we double the number of qubits at each layer. This hints at a presence of scaling dimensions in DMERA which we can then find from the eigenvalues. With a simple conversion, $\Delta = -\log_2 \lambda$, we can obtain the scaling dimension Δ from the eigenvalue λ. As the specific DMERA channel investigates a critical model, we can correspond the DMERA scaling dimensions to the scaling dimensions of the critical model.

We know that DMERA is noise resilient and that shows in the energy computations as well. We speculate that the scaling dimensions are in turn also resilient against noise. To investigate this hypothesis, we will have to compute the eigenvalues of the DMERA channel. In a classical simulation this is a simple task as we have direct access to the channel which is diagonalizable. However, on a quantum computer this is not possible. Instead, we can extract the eigenvalues directly from the measurement data as the eigenvalues are directly involved in the convergence of the expectation value of an observable. For the ground state energy we already observed that there is a convergence to a fixed value. This behavior is not exclusive to the Hamiltonian, but can be seen for all observables. By using a new quantum algorithm, we can extract the eigenvalues from the measurement data and compute the scaling dimensions for DMERA.

The code used in this section for both the classical and quantum algorithm can be found in [35].

In this section, we will first investigate the robustness of the scaling dimensions in a classical simulation in Section 5.1 by diagonalizing the DMERA channel Φ. Then in Section 5.2 we

extract the scaling dimensions on a (simulated) quantum computer by first introducing a new quantum algorithm and then testing the algorithm on simulated QC data to find the scaling dimensions.

5.1 Classical algorithm

Finding the eigenvalues for the classical simulator is a matter of diagonalizing the DMERA channel Φ. This can be done explicitly since we have direct access to Φ and can easily compute the corresponding matrix. For the method using tensor contraction we can simply contract the tensors and form a new tensor which can be interpreted as a matrix at which point the eigenvalues are easily evaluated.

For eigenvalues in a noisy channel we turn to Kraus operators once again, c.f. Section 4.1.3. As we are modeling quantum gates, we do not have an explicit matrix representation so we will have to prepare one in order to diagonalize the DMERA channel. Superoperators are linear maps from operators to operators so we can construct a matrix representation by picking a basis for the space of operators and computing each matrix element [40].

We will use that Φ maps the space of n-qubit states to itself. For this space, let us pick the basis $|i\rangle\langle j|$ for $i, j = 0, 1, \ldots, 2^n - 1$, where $|i\rangle$ indicates the vector whose i-th element is 1 and all other elements are 0. Each element of the matrix representation M of Φ is then given by

$$\begin{aligned} M_{\{(kl),(ij)\}} &= \langle |k\rangle\langle l| \,, \Phi[|i\rangle\langle j|]\rangle \\ &= \mathrm{Tr}\big[(|k\rangle\langle l|)^\dagger \Phi[|i\rangle\langle j|]\big] \\ &= \langle k| \, \Phi[|i\rangle\langle j|] \, |l\rangle \,, \end{aligned} \tag{5.1}$$

where $\langle \cdot, \cdot \rangle$ is the Hilbert-Schmidt inner product, for $k, l = 0, 1, \ldots, 2^n - 1$. Note that (kl), and likewise (ij), constitutes one index and that there is freedom in choosing how the index is formed from k and l. For example, one can pick the row-ordering to be $(00), (01), \ldots, (0d), (10), \ldots, (dd)$, where $d = 2^n$, and similarly for the column-ordering. The particular choice for the ordering does not affect further results.

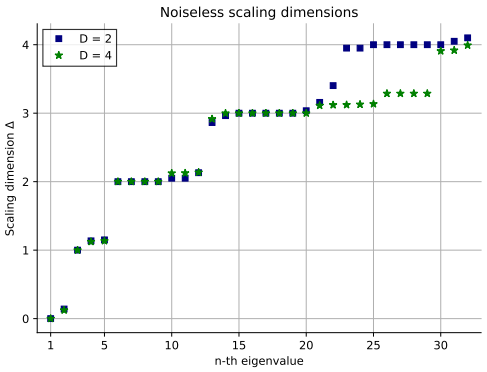
Using Eq. 5.1 for depth $D = 2$ we obtain the matrix representation M of Φ by computing all its elements for each permutation of i, j, k, l. Concretely, this involves $(2^3)^4 = 4096$ calculations as Φ is a map from 3-qubit states to 3-qubit states. Once the matrix M is determined, we can diagonalize M and compute its eigenvalues and eigenoperators. Currently, we are interested in the scaling dimensions Δ_a which can be found from the eigenvalues via $\Delta_a = 2^{-\lambda_a}$.

Results

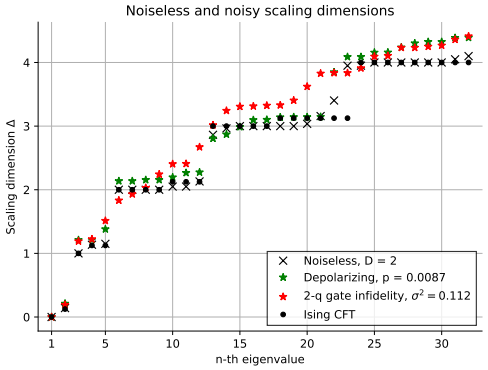
The 32 smallest scaling dimensions for both the noiseless and noisy case are plotted in Figure 5.1. Figure 5.1a contains the scaling dimensions found for two different DMERA depths in the exact case. Figure 5.1b includes the exact and noisy scaling dimensions for two different noise models. The latter figure also appears in Sewell-Jordan and in this thesis we managed to reproduce the figure using the same noise models and parameters [2].

In the exact case, both depths agree on scaling dimensions until after the 20^{th} eigenvalue. The reason for this discrepancy could be due to the larger space of operators for $D = 4$. The causal cone for $D = 4$ is extended to include 7 qubits as opposed to 3 qubits for $D = 2$. The dimensions of the space of operators for $D = 4$ is then $2^{14} \times 2^{14}$ and much larger than the space for $D = 2$. Consequently, there are many more eigenvalues in the space for $D = 4$. The distribution of eigenvalues may differ between both depths which could explain the deviation.

For the noise models, the noisy scaling dimensions are remarkably close to the exact scaling dimensions. The decaying behavior is dictated by the smallest scaling dimensions, which in



(a) The 32 smallest noiseless scaling dimensions as obtained from the classical simulation of DMERA. Two different depths for DMERA layers are shown



(b) The 32 smallest scaling dimensions with and without noise. The two noise models used in this simulation are described in Section 4.1.2 with depolarizing noise $p = 0.0087$ and two-qubit gate infidelity $\sigma = 0.112$.

Figure 5.1: Scaling dimensions for different scenarios. Code used to produce the figures from [35].

particular are close across the three cases. We know from the energy calculations in Section 4.2 and the bound on the error of the noisy channel in Section 3.2.3 that DMERA is stable in presence of noise and such it is not surprising that the scaling dimensions are also stable in a noisy environment. This result gives us the confidence that DMERA can yield reliable results when performed on a real quantum computer.

5.2 Quantum algorithm

In the previous section we obtained the eigenvalues of the DMERA channel by diagonalization. This method is possible only if one has access to the matrix representation of the channel. For a quantum computer, this is unfortunately not the case. The amount of information that we can access via a quantum computer is severely limited compared to a classical simulation. Nonetheless, it is possible to extract the eigenvalues out of the data returned by the quantum computer. In this section, we will discuss a new algorithm that estimates the eigenvalues of a DMERA channel Φ given the expectation values of an observable for up to k layers.

5.2.1 ESPRIT algorithm

The mechanism that enables us to obtain the eigenvalues is the "Estimation of Signal Parameters via Rotational Invariance Techniques" (ESPRIT) algorithm [16]. What this allows one to do, in short, is to estimate the λ_a from a series X with elements of the form

$$X_n = \sum_{a=1} x_a \lambda_a^n \tag{5.2}$$

for $n = 0, 1, \ldots, k$. Conventionally, this algorithm is used in signal processing for direction of arrival estimation, where one estimates real frequencies ω_a by using $\lambda_a = e^{-2\pi i \omega_a}$ [41, 42, 43, 44].

Let us first define our input that we will use for the algorithm. For any 3-qubit observable O we can obtain $\eta \in \mathbb{R}^{k+1}$ for up to k layers whose elements are given by

$$\eta_n = \langle O \rangle_{\rho_n} = \mathrm{Tr}[\Phi^n[O]\rho_0]. \tag{5.3}$$

The magnitude of $\langle O \rangle_{\rho_n}$ is determined by the eigenvalues of the DMERA channel Φ. As its eigenvalues have modulus strictly less than one, except for the largest which has modulus one, $\langle O \rangle_{\rho_n}$ will be solely determined by the largest eigenvalue for large n. For this reason, the elements of η typically show a decaying behavior, hence we will call η the *observable decay curve*. If O is the Hamiltonian H then η gives the energies for up to k layers, which we extensively discussed in Section 4.2.

Using Eq. 3.18 we can write η in a more suggestive manner so that we can justify using ESPRIT,

$$\eta_n = \sum_a x_a \lambda_a^n = \sum_a x_a 2^{-\Delta_a}, \tag{5.4}$$

where $x_a = \mathrm{Tr}\left[\hat{\phi}_a O\right] \mathrm{Tr}[\phi_a \rho_0]$ is a factor independent of n. Each element is then a sum of exponentials, similar to X, and we see that η is an appropriate input for ESPRIT. Note that we will have *complex* frequencies as opposed to real frequencies. In general this will not be an issue as the ESPRIT algorithm works regardless of the nature of its input.

Hankel matrix

The next ingredient we will need for ESPRIT is the Hankel matrix $\mathcal{H}(X)$. The defining property of a Hankel matrix is that each skew diagonal contains the same entries, that is $h_{i,j} = h_{i-1,j+1}$. By specifying the first column and the last row, combined with this property, the Hankel matrix is completely determined. This approach works not only for square Hankel matrices, but for also for non-square Hankel matrices. A general $\mathcal{H}(X) \in \mathbb{C}^{m \times n}$ for $X =$

$(x_1, x_2, \ldots, x_{m+n-1})$ is then defined as

$$\mathcal{H}(X) = \begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_2 & x_3 & \cdots & x_{n+1} \\ \vdots & \vdots & \ddots & \vdots \\ x_m & x_{m+1} & \cdots & x_{m+n-1} \end{pmatrix} \tag{5.5}$$

As an example, for the array $X = (1, 2, 3, 4, 5, 6)$, the Hankel matrix $\mathcal{H}(X) \in \mathbb{C}^{3\times 4}$ is given by

$$\mathcal{H}(X) = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \end{pmatrix}. \tag{5.6}$$

Specifically for ESPRIT where $X \in \mathbb{C}^{k+1}$, in general we will construct $\mathcal{H}(X) \in \mathbb{C}^{(L+1)\times(k-L+1)}$ by choosing $L = \lfloor \frac{k+1}{2} \rfloor$.

ESPRIT algorithm

Now we can describe the sequence of the ESPRIT algorithm. The following description is based on [45].

Algorithm 1 ESPRIT algorithm

Input: $X \in \mathbb{C}^{k+1}$, desired number of eigenvalues q
Output: $\Lambda_q \in \mathbb{C}^q$

1. Choose $L = \lfloor \frac{k+1}{2} \rfloor$
2. Compute the Hankel matrix $\mathcal{H}(X) \in \mathbb{C}^{(L+1)\times(k-L+1)}$
3. Perform the singular value decomposition on $\mathcal{H}(X)$,

   $$\mathcal{H}(X) = [U\, U_\perp]\, \Sigma\, [V\, V_\perp],$$

   where

   - $U \in \mathbb{C}^{(L+1)\times q}$
   - $U_\perp \in \mathbb{C}^{(L+1)\times(L+1-q)}$
   - $\Sigma \in \mathbb{C}^{q\times q}$
   - $V \in \mathbb{C}^{(k-L+1)\times q}$
   - $V_\perp \in \mathbb{C}^{(k-L+1)\times(k-L+1-q)}$
4. Find U_0 by taking the first L rows of U and U_1 by taking the last L rows of U
5. Compute $\Psi = U_0^+ U_1$, where U_0^+ is the Moore-Penrose inverse of U_0
6. Compute the eigenvalues of Ψ, which is $\Lambda_q = \{\lambda_a\}_{j=1}^q$
7. Return the eigenvalues Λ_q

If input X is in the form of $X = \sum_{a=1}^q x_a \lambda_a$ then the eigenvalues of Ψ are precisely the same λ_a as in X. However, in the case that we have a noisy input $\tilde{X} = X + \eta$, where η represents a noise vector, we should still expect to recover approximately λ_a [45].

5.2.2 F-test for nested models

Running the ESPRIT algorithm on η in the form of Eq. 5.4 gives us back q eigenvalues of the DMERA channel Φ, depending on the restriction q. In general, we do not know what value to choose for restriction q so we need a statistical test to determine what value for q is optimal. The F-test is suited for this problem. We will use this test to compare two models with q and $q+1$ eigenvalues and seeing which one of the two fits "best" to η.

Specifically, we consider the F-test for nested models [46]. Consider the sequences $y_1 = a + bx_1$ and $y_2 = a + bx_1 + cx_2$, then y_1 is nested inside y_2 as it has fewer parameters than y_2 does. If we were to fit both y_1 and y_2 to a set of points Y then we expect y_2 to have a better fit than y_1. In general, a sequence with more parameters would fit better to a set of points than a sequence with fewer parameters. However, by allowing more parameters, we increase the number of degrees of freedom which we are trying to restrict. The F-test serves to give the balancing point when adding more parameters is 'too much'.

For two models M_1 and M_2, where M_1 is nested inside M_2, and a data set D the F-value (or F-statistic) is given by

$$F = \frac{\left(\frac{\mathrm{SS}_1 - \mathrm{SS}_2}{p_2 - p_1}\right)}{\left(\frac{\mathrm{SS}_2}{n - p_2}\right)}, \tag{5.7}$$

where SS_i is the residual sum-of-squares of model M_i, p_i the number of parameters in model M_i and n the number of data points in D. The F-value is then compared to the value of the F-distribution for the values of degrees of freedom $p_2 - p_1$ and $n - p_1$ and $\alpha = 0.05$. We will call this value from the F-distribution the critical value F_{crit}. If the calculated F-value is larger than the critical value, then the model with more parameters is a better fit. If it is instead smaller than the critical value, then the simpler model is correct.

We can apply the F-test for two models with eigenvalues obtained from the ESPRIT algorithm with restriction q and $q+1$. We take a model $M_q(k)$ to be

$$M_q(k) = \sum_{a=1}^{q} w_a \lambda_a^k, \tag{5.8}$$

where w_a is the weight for each λ_a and k is the number of DMERA layers. Starting with $M_1(k)$ and $M_2(k)$, we compute the F-value between these models and compare the value to the F_{crit}. If the model with more parameters is a better fit, then we repeat the same process, now for $M_2(k)$ and $M_3(k)$. This procedure is continued until we find an F-value which is smaller than the critical value. Once this is determined, we can be certain that only the q eigenvalues given by ESPRIT are statistically likely to be eigenvalues of the DMERA channel.

The weights w_a in $M_q(k)$ can be determined by fitting the model to η using specialized coding functions, e.g. `curve_fit()` from the *scipy* Python package [47]. These functions will find an optimized set of weights that minimizes the sum of squares. This method is called (non-linear) least squares. The optimized weights can then be used in the model $M_q(k)$ for the F-test.

5.2.3 Extraction algorithm

Now that we have discussed all the necessary parts, we can assemble the extraction algorithm. In this following section we will introduce the extraction algorithm and discuss how to use in order to find the eigenvalues of the DMERA channel.

Using the ESPRIT algorithm (Algorithm 1) we are able to estimate eigenvalues λ_a from X given it is of the form of $X = \sum_{a=1} x_a \lambda_a^k$. In general, we do not know precisely how many

λ_a are present in X, so the input restriction q is not yet determined. For that reason, we try different restrictions q and see which model M_q constructed from the q values returned by ESPRIT has the "best fit" to X, as described in Section 5.2.2. Algorithm 2 provides guidelines to find a set of eigenvalues that are likely to be in X.

Algorithm 2 Eigenvalue extraction algorithm

Input: $X \in \mathbb{C}^{k+1}$
Output: $\Lambda_q \in \mathbb{C}^q$

1. Set $q = 1$
2. Input X into ESPRIT (Algorithm 1) with restriction q to obtain Λ_q. Repeat for restriction $q + 1$
3. Construct models M_q, M_{q+1} with the values from Λ_q and Λ_{q+1}
4. Find weights $w_a^{(q)}$ and $w_a^{(q+1)}$ by fitting M_q and M_{q+1} to X
5. Compute the F-value for M_q and M_{q+1}
   - If F-value $\geq F_{\text{crit}}$,
     then go back to step 2 and increase q by 1
   - If F-value $< F_{\text{crit}}$,
     then return Λ_q as computed in step 2

If we use η as an input for Algorithm 2 then we are able to find q likely eigenvalues from the DMERA channel. The only information required from the (simulated) quantum computer is measurements for a given observable for up to k layers. The algorithm and computation of the eigenvalues is done classically and this ensures that the algorithm is as reliable as possible.

Limitations

A single pair of a starting state and an observable cannot not yield all eigenvalues from the channel. The main reason is that a starting state ρ_0 and an observable O cannot have overlap with all left and right eigenoperators from the channel. Recall from Section 3.1.5 that O has overlap with the right eigenoperator ϕ_a of Φ if $\langle \phi_a, O \rangle \neq 0$. Similarly, we can say that ρ_0 has overlap with the left eigenoperator $\hat{\phi}_a$ of Φ if $\langle \hat{\phi}_a, \rho_0 \rangle \neq 0$. If either $\langle \phi_a, O \rangle$ or $\langle \hat{\phi}_a, \rho_0 \rangle$ vanishes for a, then there will be no contribution from λ_a in η and thus it cannot be found by the extraction algorithm.

Another problem inherent to DMERA is that there is a bias in η towards larger eigenvalues. Large eigenvalues decay much slower than small eigenvalues and are the dominant part in the signal after many layers. Smaller eigenvalues, however, decay quickly and due to the discrete nature of layers they will be negligible after only a few layers. As we have access only to the data from the observable decay, it is difficult to distinguish between the contribution of small eigenvalues and noise. Both the ESPRIT and the extraction algorithm will then likely classify these smaller eigenvalues as noise and filter these out.

The first problem can be solved by collecting data from many pairs of different starting states and observables. Different pairs will have overlap with different eigenoperators and as a result we can retrieve more eigenvalues of the channel. We will discuss the data collection in Section 5.2.4 and the results in Section 5.2.5. Another benefit of collecting more data is that

we will find more of the same eigenvalues, so the certainty that a found eigenvalue is correct increases.

5.2.4 Data collection

As discussed in Section 5.2.3, a single pair of starting state and observable cannot find all eigenvalues of the DMERA channel Φ. To find as many eigenvalues as possible, we need to consider many pairs of starting states and observables. In addition, the starting states and observables must be simple so they can easily be prepared on a quantum computer. For these reasons, we consider the stabilizer states as the starting states and Pauli gates as the observables. In this section, we will explore the reasoning for these choices and discuss how the data is collected and analyzed.

Stabilizer states and observables

Our goal is to choose a set of starting states which forms a basis for 3-qubit states so that all left eigenoperators of Φ are contained in this set. Trivially, this can be done by taking the set of left eigenoperators and transforming them to states, i.e. $\rho = \frac{I}{d} + \hat{\phi}_a$ for all but the fixed point state. This set would yield every eigenvalue when analyzed with the eigenvalue extraction algorithm. However, we assume that we do not know the eigenoperators and in addition they would be difficult to prepare on a quantum computer. Instead, we will need to find a different set of states which are easy to prepare and that form a basis of quantum states.

Let us consider stabilizer states. We say that $|\psi\rangle$ is stabilized by a unitary U if $U|\psi\rangle = +1|\psi\rangle$. For example, $|0\rangle$ is stabilized by Z and $|1\rangle$ is stabilized by $-Z$. Stabilizer states can be generated by a circuit of stabilizer gates (CNOT, Hadamard H and phase gate P) starting from $|0\dots0\rangle\langle0\dots0|$ [48]. Normally, the states are used in quantum error-correction, but they also have uses outside this area.

Starting from $|0\rangle\langle0|$, we can create the single qubit stabilizer states

$$|0\rangle\langle0| = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}, \qquad |+\rangle\langle+| = \frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}, \qquad |i\rangle\langle i| = \frac{1}{2}\begin{pmatrix}1 & -i\\ i & 1\end{pmatrix},$$

$$|1\rangle\langle1| = \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}, \qquad |-\rangle\langle-| = \frac{1}{2}\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix}, \qquad |-i\rangle\langle-i| = \frac{1}{2}\begin{pmatrix}1 & i\\ -i & 1\end{pmatrix}.$$

One can check that the given states are stabilized by either $\pm X$, $\pm Y$ or $\pm Z$, and trivially by I.

One of our requirements is that this set of single qubit stabilizer states forms a basis for 3-qubit density matrices. First, let us see if this set forms a basis for 1-qubit density matrices. In fact, these six states form an overcomplete basis as two states can be written as a linear combination of other states in this set, that is

$$|-i\rangle\langle-i| = |0\rangle\langle0| + |1\rangle\langle1| - |i\rangle\langle i|$$

and

$$|-\rangle\langle-| = |0\rangle\langle0| + |1\rangle\langle1| - |+\rangle\langle+|,$$

so we can eliminate $|-i\rangle\langle-i|$ and $|-\rangle\langle-|$ from the set. We can alternatively write the remaining states as

$$|0\rangle\langle0| = \frac{1}{2}(I+Z), \qquad |+\rangle\langle+| = \frac{1}{2}(I+X),$$

$$|1\rangle\langle1| = \frac{1}{2}(I-Z), \qquad |i\rangle\langle i| = \frac{1}{2}(I+Y).$$

Then for a general state ρ we have

$$\begin{aligned}\rho &= \alpha\,|+\rangle\langle+| + \beta\,|i\rangle\langle i| + \gamma\,|0\rangle\langle 0| + \delta\,|1\rangle\langle 1| \\ &= \frac{\alpha}{2}(I+X) + \frac{\beta}{2}(I+Y) + \frac{\gamma}{2}(I+Z) + \frac{\delta}{2}(I-Z) \\ &= \frac{1}{2}\left[(\alpha+\beta+\gamma+\delta)I + \alpha X + \beta Y + (\gamma-\delta)Z\right], \end{aligned} \tag{5.9}$$

where we recognize this expression as the Bloch sphere picture if we require $\alpha+\beta+\gamma+\delta = 1$. As the Pauli matrices form a basis for 1-qubit states, then the set $\{|0\rangle\langle 0|, |1\rangle\langle 1|, |+\rangle\langle +|, |i\rangle\langle i|\}$ does as well. Given this set then by extension we can construct a basis for 3 qubits, i.e. $\{|0\rangle\langle 0|, |1\rangle\langle 1|, |+\rangle\langle +|, |i\rangle\langle i|\}^{\otimes 3}$.

Now that we have shown that this set of stabilizer states forms a basis for 3-qubit states, we also note that stabilizer states are particularly easy to prepare on a quantum computer. They are generated from a circuit consisting of stabilizer gates, which itself are basic operations that a quantum computer can easily perform. In particular, the phase gate P does not happen physically but is rather performed virtually, thus any usage of P cannot introduce error. Starting from $|0\rangle$ we can construct the stabilizer states as

$$\begin{aligned} |0\rangle\langle 0| &= I[|0\rangle\langle 0|] \\ |1\rangle\langle 1| &= X[|0\rangle\langle 0|] \\ |+\rangle\langle +| &= H[|0\rangle\langle 0|] \\ |i\rangle\langle i| &= PH[|0\rangle\langle 0|]. \end{aligned}$$

For the observables we will pick the set consisting of Pauli gates $\{X, Y, Z\}$. These observables are the natural choice as quantum computers already use these measurement gates and any arbitrary observable is converted in terms of the Pauli gates by the quantum computer. For example, the Hamiltonian that we used in Section 4.2 can be decomposed in terms of Pauli gates as $H = XZX - \frac{1}{2}(IXX + XXI)$.

Data collection

From the basis states set we can construct $4^3 = 64$ starting states and from the Pauli gates set $3^3 = 27$ observables, then in total we have 1728 pairs for which we can compute the observable decay curve η. However, most pairs have no overlap at all and return an empty η, i.e. $\eta_n = 0$ for all n. This occurs in roughly between 50 and 75% of the pairs which reduces the number of different η to be between 400 and 900.

In general, we will want to collect data points until the curve η has converged. The convergence rate differs from state-observable pair to another, so we need a method that utilizes relative changes. We say that the curve is converged when the mean of the relative changes of the last three data points is less than 0.02. As the curve may sometimes decay too slowly for this to occur, we also include a forced stop at 30 layers. The value of 0.02 is obtained through trial-and-error by looking at a state-observable pair decay curve at different cut-off values, determining when it looks converged and cross-checking with other state-observable pairs. The ESPRIT algorithm performs best on decay curves that are just converged.

Combining decay curves

Algorithm 2 is best used with a combined sample of multiple η, instead of individual η. Combining multiple decay curves decreases the susceptibility to errors and deviations, and emphasizes the decay from the eigenvalues. Consider two decay curves $\eta^{(1)}, \eta^{(2)} \in \mathbb{R}^{k+1}$ which

contain the same eigenvalues λ_a. Then in general an element from $\eta^{(1)}$, and similarly for $\eta^{(2)}$, will be of the form

$$\eta_m^{(1)} = \sum_a x_a^{(1)} \lambda_a + \epsilon^{(1)}(m),$$

where $\epsilon^{(1)}(m)$ is a noise vector. Adding both curves $\eta^{(1)}, \eta^{(2)}$ gives

$$\eta_m^{(1)} + \eta_m^{(2)} = \sum_a (x_a^{(1)} + x_a^{(2)})\lambda_a + (\epsilon^{(1)}(m) + \epsilon^{(2)}(m)).$$

Assuming the curves are similar enough, that is $x_a^{(1)}$ and $x_a^{(2)}$ have the same sign and similar magnitude, and that the noise is uncorrelated, then the linear sum will give us a better estimation of λ_a than the individual curves would. Extending this method to summing more than two curves will improve the results even further.

For this procedure, we assumed that both $\eta^{(1)}$ and $\eta^{(2)}$ share the same eigenvalues. In general, it is unlikely that this assumption holds for any two curves, and if there are more than two curves involved then it becomes increasingly unlikely. However, the linear combination method still works if we assume that the curves share *some* eigenvalues, especially those in which we are interested. If a set of curves share the same larger eigenvalues, then that is a good candidate set to use to find those eigenvalues. In practice, we will combine the curves from the state-observable pairs which share the same observable. From observations by using Algorithm 2 on all decay curves, we found that pairs which share an observable have a similar decay curve and yield similar eigenvalues.

The decay curves $\eta^{(1)}$ and $\eta^{(2)}$ need not have the same length, i.e. $k_1 \neq k_2$, as they may have converged at different lengths. In this case we cannot directly sum both curves, so to do this we extend one of the curves until their lengths are equal. For example, if $\eta^{(1)}$ is shorter than $\eta^{(2)}$, so $k_1 < k_2$, then we add $k_2 - k_1$ elements onto $\eta^{(1)}$ whose values are equal to the value of the previously last element of $\eta^{(1)}$. We call this procedure *padding*.

5.2.5 Results

The scaling dimensions found using Algorithm 2 are shown in Table 4.1, compared with values from the exact Ising CFT and matrix diagonalization from classical simulation. The values found with the algorithm are more likely to correspond with the values from the diagonalization method, as both methods aim to find the scaling dimensions of the same function, i.e. the DMERA channel. We can see in the results that this is indeed the case.

	Ising CFT	Diagonalization ($D = 2$)	Algorithm (error 1σ)
Δ_0	0	0	0.013 ± 0.009
Δ_1	0.125	0.140	0.140 ± 0.001
Δ_2	1	1	1.033 ± 0.011
Δ_3	1.125	1.136	1.135 ± 0.086
Δ_4	2	2	1.942 ± 0.047

Table 5.1: Scaling dimensions obtained with Algorithm 2. The error size is one standard deviation. The code used to generate the data and perform Algorithm 2 can be found in [35].

Although some eigenvalues are abundant in the data, such as λ_1 (with corresponding $\Delta_1 = 0.140 \pm 0.001$), other eigenvalues do not occur often. The eigenvalue λ_2 (w. corresp. $\Delta_2 = 1.033 \pm 0.011$) was only encountered in the data of a low number of state-observable pairs, while the eigenvalues λ_0 (w. corresp. $\Delta_0 = 0.013 \pm 0.009$) and λ_1 were present in the data

of almost all state-observable pairs. Consequently, the Δ_2 from the algorithm is 3σ removed from the diagonalization (and Ising CFT) value while both Δ_0 and Δ_1 are within 1.5σ of the diagonalization value. This rare occurrence of some eigenvalues makes it difficult to accurately determine if the found eigenvalues are real or only noise.

Padding curves allows us to sum curves that do not have the same length, i.e. not the same convergence rate. However, it can happen that the length of the linear combination is too great. This makes it difficult for the ESPRIT algorithm to find good approximations of eigenvalues that are in the data. Normally, we do not want the decay curve η to be unnecessarily long, hence why we only collect data points until η has converged. However, when we pad an η so it can be used in the linear combination, we extend the η past its convergence, which only worsens the performance of ESPRIT. To make sure we get the best performance from ESPRIT, we remove several last elements from the linear combination before using it as input for the extraction algorithm.

5.2.6 Discussion and outlook

To conclude, based on the results shown in Table 5.1, the extraction algorithm seems to perform well as we were able to recover most of the larger DMERA scaling dimensions within one or two σ. The results show a bias towards larger eigenvalues, as small eigenvalues decay after only a few channel iterations, which makes it difficult to be recovered by the extraction algorithm. In this thesis this bias was not an issue as we were not focused on recovering specific eigenvalues, however the algorithm might not be suited for research that is interested in many smaller eigenvalues.

Outlook

For future outlook, there are several things that can be considered. Firstly, the performance is directly related to the input data, so with more data (decay curves) the algorithm might be able to more accurately extract eigenvalues/scaling dimensions. One method that could be deployed is *shadow estimation* [49]. By using many random unitaries we would be able to accurately determine the expectation value of an observable O which in turn results in accurate decay curves.

Secondly, in this thesis we have used a DMERA circuit that is derived analytically and is thus both scale and translation invariant [17]. However, this approach might not be optimal for approaching the ground state energy. By starting from the analytic circuit and using a hybrid quantum-classical method of optimization, described in Kim-Swingle [1], we can minimize the energy further.

Another potential next step for DMERA would be to explore different critical models. The quantum Ising chain is a relatively simple model such that DMERA does not yield a large benefit over a classical method. An one-dimensional model that is proposed by Kim and Swingle is the so-called D1-D5 system, which is constructed in string theory [50]. Furthermore, DMERA can currently only investigate 1D systems, however an extension to higher dimensions would be an interesting point of research. There are many 2D systems that are non-trivial to solve, such as the Hubbard model [51].

Finally, a recent study proposed a generalized MERA (gMERA) network that is also suited for a quantum computer [52]. Although gMERA is a different approach to preparing MERA on a quantum computer, its results are consistent with results obtained from DMERA. A comparison between DMERA and gMERA would be useful to determine which method has a better performance and higher accuracy for a given situation.

Appendix A

Appendices

A.1 Tensor contraction

Tensors are a generalization of scalars, vectors, matrices and higher multidimensional arrays, or simply speaking a mathematical object with indices. A tensor T is characterized by the way its components transform under a change of basis, for example

$$T^{\rho}_{\sigma} = \frac{\partial x^{\mu}}{\partial x^{\sigma}} \frac{\partial x^{\rho}}{\partial x^{\nu}} T^{\nu}_{\mu}. \tag{A.1}$$

We can identify a tensor in terms of its *rank*, or type, which describes how many lower and upper indices a tensor has. For example, the tensor T in Eq. A.1 is a (1,1)-rank tensor. A general (m, n)-rank tensor has m upper indices and n lower indices.

A new tensor can be formed by combining tensors and summing over the connected indices, which we call *tensor contraction*. Using the Einstein summation convention, an example of tensor contraction is

$$v_l = A^k_l w_k, \tag{A.2}$$

where we have summed over the index k and contracted A^k_l and w_k to obtain v_l.

Alternatively, there is a graphical notation for tensors, sometimes called the Penrose graphical notation [53, 54]. A tensor is represented as a shape with a number of legs which is equal to the number of indices, see Figure A.1a. Tensor contraction can then be visualized as connecting the legs of the tensors that are to be contracted. Eq. A.2 and Figure A.1b imply the same thing, but differ in notation.

(a) Different tensors (b) Tensor contraction

Figure A.1: Tensors

A.2 Telescopic decomposition

In this appendix we will prove that we can express $A^m - B^m$ as

$$A^m - B^m = \sum_{i=1}^{m} A^{i-1}(A_i - B_i)B^{m-i}, \tag{A.3}$$

which we call the telescopic decomposition.

Suppose A^m is a product of A_i,

$$A^m = \prod_{i=1}^{m} A_i = A_1 \cdots A_m, \tag{A.4}$$

where A_i come with an associative product, i.e. $(A_1A_2)A_3 = A_1(A_2A_3)$, and similarly for B^m.

We will prove Eq. A.3 by induction. First for the base case $m = 1$,

$$A^1 - B^1 = I(A_1 - B_1)I = A_1 - B_1$$

and then for the induction step,

$$\begin{aligned} A^{m+1} - B^{m+1} &= A^m A_{m+1} - A^m B_{m+1} + A^m B_{m+1} - B^m B_{m+1} \\ &= A^m(A_{m+1} - B_{m+1}) + (A^m - B^m)B_{m+1} \\ &= A^m(A_{m+1} - B_{m+1}) + \left(\sum_{i=1}^{m} A^{i-1}(A_i - B_i)B^{m-i}\right) B_{m+1} \\ &= \sum_{i=1}^{m+1} A^{i-1}(A_i - B_i)B^{m+1-i}, \end{aligned}$$

which concludes our proof by induction.

Bibliography

[1] Isaac H. Kim and Brian Swingle. *Robust entanglement renormalization on a noisy quantum computer*. 2017. arXiv: 1711.07500.

[2] Troy J. Sewell and Stephen P. Jordan. *Preparing Renormalization Group Fixed Points on NISQ Hardware*. 2021. arXiv: 2109.09787.

[3] Amazon. *Quantum Computing Service*. URL: https://aws.amazon.com/braket.

[4] IBM. *IBM Quantum Computing*. URL: https://www.ibm.com/quantum.

[5] Google. *Google Quantum AI*. URL: https://quantumai.google.

[6] Matthias Evers, Anna Heid, and Ivan Ostoji. *Pharma's digital Rx: Quantum computing in drug research and development*. 2021. URL: https://www.mckinsey.com/industries/life-sciences/our-insights/pharmas-digital-rx-quantum-computing-in-drug-research-and-development.

[7] Dylan Herman et al. *A Survey of Quantum Computing for Finance*. 2022.

[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014.

[9] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* 2 (Aug. 2018), p. 79.

[10] Xiongjie Yu, David Pekker, and Bryan K. Clark. "Finding Matrix Product State Representations of Highly Excited Eigenstates of Many-Body Localized Hamiltonians". In: *Phys. Rev. Lett.* 118 (1 Jan. 2017), p. 017201.

[11] Michele Dolfi et al. "Matrix product state applications for the ALPS project". In: *Computer Physics Communications* 185.12 (Dec. 2014), pp. 3430–3440.

[12] Philippe Corboz et al. "Simulation of strongly correlated fermions in two spatial dimensions with fermionic projected entangled-pair states". In: *Physical Review B* 81.16 (Apr. 2010).

[13] G. Vidal. "Entanglement Renormalization". In: *Physical Review Letters* 99.22 (Nov. 2007).

[14] G. Vidal. "Class of Quantum Many-Body States That Can Be Efficiently Simulated". In: *Physical Review Letters* 101.11 (Sept. 2008). ISSN: 1079-7114.

[15] Robert N. C. Pfeifer, Glen Evenbly, and Guifré Vidal. "Entanglement renormalization, scale invariance, and quantum criticality". In: *Physical Review A* 79.4 (Apr. 2009). ISSN: 1094-1622.

[16] Richard H. Roy, Arogyaswami Paulraj, and Thomas Kailath. "Estimation of Signal Parameters via Rotational Invariance Techniques - ESPRIT". In: *MILCOM 1986 - IEEE Military Communications Conference: Communications-Computers: Teamed for the 90's* 3 (1986), pp. 41.6.1–41.6.5.

[17] Glen Evenbly and Steven R. White. "Entanglement Renormalization and Wavelets". In: *Physical Review Letters* 116.14 (Apr. 2016). ISSN: 1079-7114.

[18] P.W. Shor. "Algorithms for quantum computation: discrete logarithms and factoring". In: *Proceedings 35th Annual Symposium on Foundations of Computer Science* (1994), pp. 124–134.

[19] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855.

[20] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. 10th. Cambridge University Press, 2000.

[21] John Watrous. *The Theory of Quantum Information*. Cambridge University Press, 2018.

[22] J. Eisert, M. Cramer, and M. B. Plenio. "Area laws for the entanglement entropy". In: *Reviews of Modern Physics* 82.1 (Feb. 2010), pp. 277–306.

[23] J. Ignacio Cirac et al. "Matrix product states and projected entangled pair states: Concepts, symmetries, theorems". In: *Rev. Mod. Phys.* 93 (4 Dec. 2021), p. 045003.

[24] Ulrich Schollwöck. "The density-matrix renormalization group in the age of matrix product states". In: *Annals of physics* 326.1 (2011), pp. 96–192.

[25] G. Evenbly and G. Vidal. "Algorithms for entanglement renormalization". In: *Physical Review B* 79.14 (Apr. 2009). ISSN: 1550-235X.

[26] G. Evenbly and G. Vidal. *Quantum Criticality with the Multi-scale Entanglement Renormalization Ansatz*. Springer, Berlin, Heidelberg, 2013.

[27] G. Evenbly and G. Vidal. "Class of Highly Entangled Many-Body States that can be Efficiently Simulated". In: *Physical Review Letters* 112.24 (June 2014). ISSN: 1079-7114.

[28] Ya-Lin Lo et al. "Quantum impurity in a Luttinger liquid: Universal conductance with entanglement renormalization". In: *Physical Review B* 90 (Dec. 2014). DOI: 10.1103/PhysRevB.90.235124.

[29] G. Evenbly and G. Vidal. "Scaling of entanglement entropy in the (branching) multiscale entanglement renormalization ansatz". In: *Physical Review B* 89.23 (June 2014).

[30] Michael M. Wolf. *Quantum Channels and Operators: Guided Tour*. 2012. URL: https://www-m5.ma.tum.de/foswiki/pub/M5/Allgemeines/MichaelWolf/QChannelLecture.pdf.

[31] K. Wright et al. "Benchmarking an 11-qubit quantum computer". In: *Nature Communications* 10.1 (Nov. 2019).

[32] David C. McKay et al. "Efficient Z-gates for quantum computing". In: *Physical Review A* 96.2 (Aug. 2017).

[33] Dmitri Maslov and Yunseong Nam. "Use of global interactions in efficient quantum circuit constructions". In: *New Journal of Physics* 20.3 (Mar. 2018), p. 033018. ISSN: 1367-2630.

[34] Charles R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362.

[35] Bjarne Bouwer. *Python scripts for project*. URL: `https://github.com/amsqi/msc-bjarne`.

[36] Yuval Elias, Tal Mor, and Yossi Weinstein. "Semioptimal practicable algorithmic cooling". In: *Physical Review A* 83.4 (Apr. 2011).

[37] José M. Fernandez et al. "Algorithmic Cooling of Spins: A Practicable Method for Increasing Polarization". In: *International Journal of Quantum Information* 02 (2004), pp. 461–477.

[38] Amazon Web Services. *Amazon Braket*. 2020. URL: `https://aws.amazon.com/braket/`.

[39] John Cardy. *Scaling and Renormalization in Statistical Physics*. Cambridge University Press, 1996, pp. 52–54.

[40] Walter Rudin. *Principles of mathematical analysis*. McGraw-Hill Book Company, Inc., New York-Toronto-London, 1953, p. 210.

[41] Feifei Gao and Alex B Gershman. "A generalized ESPRIT approach to direction-of-arrival estimation". In: *IEEE signal processing letters* 12.3 (2005), pp. 254–257.

[42] Tukaram Baburao Lavate, VK Kokate, and AM Sapkal. "Performance analysis of MUSIC and ESPRIT DOA estimation algorithms for adaptive array smart antenna in mobile communication". In: *2010 Second International Conference on Computer and Network Technology*. IEEE. 2010, pp. 308–311.

[43] Bjorn Ottersten and Thomas Kailath. "Direction-of-arrival estimation for wide-band signals using the ESPRIT algorithm". In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 38.2 (1990), pp. 317–327.

[44] Shahram Shahbazpanahi, Shahrokh Valaee, and Mohammad Hasan Bastani. "Distributed source localization using ESPRIT algorithm". In: *IEEE Transactions on Signal Processing* 49.10 (2001), pp. 2169–2178.

[45] Weilin Li, Wenjing Liao, and Albert Fannjiang. *Super-resolution limit of the ESPRIT algorithm*. 2019. URL: `https://arxiv.org/abs/1905.03782`.

[46] Andrew Siegel. *Practical Business Statistics*. 7th ed. Academic Press, 2016, pp. 367–369.

[47] Pauli Virtanen et al. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python". In: *Nature Methods* 17 (2020), pp. 261–272.

[48] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Physical Review A* 70.5 (Nov. 2004).

[49] Hsin-Yuan Huang, Richard Kueng, and John Preskill. "Predicting many properties of a quantum system from very few measurements". In: *Nature Physics* 16.10 (June 2020), pp. 1050–1057.

[50] Gautam Mandal. *A review of the D1/D5 system and five dimensional black hole from supergravity and brane viewpoint*. 2000.

[51] Daniel P. Arovas et al. "The Hubbard Model". In: *Annual Review of Condensed Matter Physics* 13:239–74 (2022).

[52] Sajant Anand et al. *Holographic quantum simulation of entanglement renormalization circuits*. 2022.

[53] Roger Penrose. "Applications of negative dimensional tensors". In: *Combinatorial mathematics and its applications* 1 (1971), pp. 221–244.

[54] Jacob C. Bridgeman and Christopher T. Chubb. "Hand-waving and interpretive dance: an introductory course on tensor networks". In: *Journal of Physics A: Mathematical and Theoretical* 50 (2016).