Ulm University
Institute of Theoretical Physics
Controlled Quantum Dynamics group

Quantum state tomography using maximum likelihood estimation on purified matrix product states

Bassem Tarek Safieldeen

Monday, 15.8.2016

Bachelor thesis

Supervisor: Milan Holzäpfel, M.Sc.
Advisor: Prof. Dr. Martin B. Plenio

Declaration

I, hereby, declare that the work in this thesis is performed by me without external resources other than what is acknowledged and cited.

Bassem Tarek Safieldeen

Ulm, August, 2016

Abstract

We compare the performance of the maximum likelihood estimation algorithm for quantum state tomography when used with the matrix product operator and the purified matrix product state forms of the state of a one dimensional quantum system. The comparison is based on reconstruction quality (how close we get to the state we are trying to reconstruct) and reconstruction speed (how many iterations are needed to reach that state). Moreover, we examine the scalability of the two algorithms for physical systems with large number of sites. Finally, we discuss a possible approach to make the maximum likelihood estimation algorithm faster.

Contents

Acknowledgments

Although this is only a bachelor thesis, I think a few acknowledgments are in order.

Besides writing a respectable Bachelor thesis, my aim from this exchange semester was to get to know what kinds of efforts are being made and what kinds of questions are being asked in order to exploit the wildness of Quantum Mechanics for creating revolutionary devices, with my focus being on quantum computers. This endeavor, I believe, will eventually have the double benefit of bettering our understanding of Quantum Mechanics, which is exciting in its own right. Between the papers I have read on the different topics in the field, the many talks I attended at the institute, the research visit to KIT, the discussions I had with my colleagues, and my own work on the thesis, I think I achieved that aim to some extent, and for that I would like to express my gratitude to a few people. First, I would like to express my gratitude to Prof. Barton Zwiebach, Prof. Wolfgang Ketterle, and whoever else is involved in making the online MIT lectures. It is partially through these lectures that I first learned about the field of Quantum Computation and Quantum Information. I am grateful to Prof. Martin Plenio for giving me the opportunity to work on my thesis in his group, and for offering his help when I mentioned I was planning to visit other research groups where research on quantum computers was being carried out. I would also like to express my gratitude to my supervisor, Milan Holzäpfel, for his guidance in and outside the thesis work. I would like to thank him and my officemate Thomas Theurer for taking the time to meticulously proofread my thesis (multiple times). Furthermore, I would like to thank Dr. Martin Weides for hosting me in his institute for a week and for giving me the freedom to wander around, and I would like to thank his students for allowing me to pester them for that week. The visit was absolutely one of the highlights of my stay. Last but not least, I would like to express my gratitude to Prof. Gerd Baumann for helping me apply to this project, for showing interest in my progress throughout my stay, and for taking the time to meet with me whenever he happened to be in Ulm.

Chapter 1

Introduction

Suppose we have a quantum many-body system with an unknown state $\hat{\rho}$. To find out the state of the system, we must perform measurements on the system. One measurement will not be enough since all it will do is collapse the state to one of the eigenstates of the observable corresponding to the measurement we made. Instead, we must perform many, ideally infinite, measurements on identically prepared copies of the state using a set of observables that are informationally complete, i.e. a set of observables that we can use to gain all possible information about the state. This process of reconstructing an unknown quantum state by doing many measurements on identical copies of it is called Quantum State Tomography (QST). QST is frequently used in labs to verify the state of quantum many-body system before or/and after an operation has been done on the system (see [7, 23, 13, 22, 11, 21] for example), making it a powerful tool to have when building a quantum information processing device.

1.1 The Curse of Dimensionality

As the technology of building quantum processing devices matures and the number of subsystems (e.g. qubits) in our system increases, so will the dimension of the Hilbert space in which the system's state lives, and quickly standard QST algorithms fail due the exponential growth of the size of the density matrix of the state and the amount of computational resources needed to store it and do operations on it. This exponential growth of the Hilbert space first manifests itself in the number of measurements (and consequently the time of measurement) that must be done in the lab to obtain the information necessary to reconstruct the state of the system (e.g. [11]).

However, with the advent of Matrix Product Operators (MPO) and Matrix Product States (MPS) it was found that, for many physical states and approximations thereof (to be discussed in later sections), this exponential scaling of the information included in a state with the number of sites does not occur [28]. The MPO representation of these states reduces the number of parameters needed to capture the information contained in the state significantly and allows for efficient application of QST on these states, a process which would otherwise be extremely difficult for large systems. In [6], the authors showed that arming the Maximum Likelihood Estimation (MLE) algorithm, a widely used QST algorithm [15], with the MPO representation makes QST (on states that are accurately representable by the MPO form) truely scalable to large systems.

1.2 This Thesis

Although it is an indisputable improvement over traditional QST approaches, applying the MLE algorithm using the MPO form of the state is not flawless. For a variety of reasons to be discussed in this thesis, reconstruction algorithms, including those involving the MPO form, often fail to produce a positive semi-definite estimate of the state. For the reconstruction algorithm we are going to focus on in this thesis, the MLE algorithm, this failure is usually accompanied by an instability of the algorithm, which can lead to the complete failure of the reconstruction process (chapter 4). An alternative to using the MPO form for doing QST is to use the locally purified form of the state, or the Purified Matrix Product State (PMPS) form. The main purpose of this thesis is to explore the performance of the PMPS-MLE in terms of reconstruction quality and speed, and compare it with the MPO-MLE.

This thesis is organized as follows: In chapter 2, we introduce the concept of QST and we talk about the different algorithms that could be used to reconstruct the state, including the MLE algorithm. Then in chapter 3, we talk about all the details of the Matrix Product form of the state, and how manipulation of and operations on the quantum state are done in this form. In chapter 4, we show how the MLE algorithm is merged with the MPO form of the state, then we discuss an improvement over the original MPO-MLE algorithm introduced in [6], and finally, we discuss the several issues with the MPO-MLE that motivate the search for a better algorithm. In chapter 5, we introduce the PMPS form of a quantum state, and we use it to introduce the PMPS-MLE algorithm and explain how it can be a viable alternative to the MPO-MLE algorithm. After that, we present a comparison of the PMPS-MLE and the MPO-MLE that addresses the issues in the MPO-MLE. We conclude chapter 5 by examining the scalability of the PMPS-MLE and MPO-MLE algorithms to large system sizes. In chapter 6, we discuss a modified MLE algorithm and provide examples to show that it is faster than the standard MLE algorithm.

Chapter 2

Quantum State Tomography

In this chapter we discuss two approaches to Quantum State Tomography (QST), one of which is the Maximum Likelihood Estimation (MLE) algorithm, which we shall be using throughout this thesis.

2.1 Quantum State Tomography

One of the most popular platforms for research on quantum information devices is the linear ion trap, which consists of a string of ions trapped in a Paul trap [7]. For the linear ion trap quantum processor, a qubit's two levels are encoded in a pair of electronic stable and metastable levels of the ion. Manipulation of the quantum state of each ion is done using a laser tuned to the energy between these two energy levels of the ion, and two-qubit operations, like the C-NOT gate, are done by exploiting the coulomb force between the ions [7].

Now, suppose we are given such a system and we want to determine its quantum state, i.e. we want to do QST. The (yet unknown) state of the system can be written as a weighted sum of the elements of an orthonormal operator basis. Using the Pauli spin basis, we can write the state as

$$\hat{\rho} = \sum_i \mathrm{tr}(\hat{P}_i^\dagger \hat{\rho}) \hat{P}_i,$$

where $\hat{P}_i = \hat{P}_1^{i_1} \otimes \cdots \otimes \hat{P}_N^{i_N}$, and N is the number of ions in the trap; $\hat{P}_k^1 = \mathbb{I}/\sqrt{2}$, $\hat{P}_k^2 = \hat{\sigma}_x/\sqrt{2}$, $\hat{P}_k^3 = \hat{\sigma}_y/\sqrt{2}$, and $\hat{P}_k^4 = \hat{\sigma}_z/\sqrt{2}$ are the Pauli spin matrices up to a normalizing constant. Thus, in order to determine the state, we only need to know what the expectation values $\mathrm{tr}(\hat{P}_i^\dagger \hat{\rho})$ are. These values can be completely determined by doing measurements whose corresponding observables can be decomposed into projectors onto the eigenvectors of the observable [18]. That is,

$$\hat{\sigma}_x = |x_{\mathrm{up}}\rangle \langle x_{\mathrm{up}}| + |x_{\mathrm{down}}\rangle \langle x_{\mathrm{down}}|$$

$$\hat{\sigma}_y = |y_{\mathrm{up}}\rangle \langle y_{\mathrm{up}}| + |y_{\mathrm{down}}\rangle \langle y_{\mathrm{down}}|$$

$$\hat{\sigma}_z = |z_{\mathrm{up}}\rangle \langle z_{\mathrm{up}}| + |z_{\mathrm{down}}\rangle \langle z_{\mathrm{down}}|,$$

where

$$|x_{\text{up}}\rangle\langle x_{\text{up}}| = \frac{1}{2}\begin{pmatrix}1 & 1\\ 1 & 1\end{pmatrix}$$

$$|x_{\text{down}}\rangle\langle x_{\text{down}}| = \frac{1}{2}\begin{pmatrix}1 & -1\\ -1 & 1\end{pmatrix}$$

$$|y_{\text{up}}\rangle\langle y_{\text{up}}| = \frac{1}{2}\begin{pmatrix}1 & -i\\ i & 1\end{pmatrix}$$

$$|y_{\text{down}}\rangle\langle y_{\text{down}}| = \frac{1}{2}\begin{pmatrix}1 & i\\ -i & 1\end{pmatrix}$$

$$|z_{\text{up}}\rangle\langle z_{\text{up}}| = \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix}$$

$$|z_{\text{down}}\rangle\langle z_{\text{down}}| = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix}.$$

For each Pauli setting, the set of the corresponding projectors form what is called a POVM set. POVM is short for Positive Operator Valued Measurements; and a POVM set is defined as a set whose elements are hermitian, positive semi-definite, and complete, i.e. all elements sum to the identity matrix [5]. The projectors for all elements in our Pauli basis are then combined into one large POVM set (which still needs to be normalized) such that the POVM measurement set for one qubit consists of the elements $\hat{\Pi}_i^j$, where $\hat{\Pi}_1^1 = |x_{\text{up}}\rangle\langle x_{\text{up}}|$, $\hat{\Pi}_1^2 = |x_{\text{down}}\rangle\langle x_{\text{down}}|$, $\hat{\Pi}_2^1 = |y_{\text{up}}\rangle\langle y_{\text{up}}|$, and so on until $\hat{\Pi}_3^2 = |z_{\text{down}}\rangle\langle z_{\text{down}}|$. Note that the large POVM set has to be normalized to makes sure its elements sum to the identity. For a system consisting of more than one qubit, the projectors consist of all possible tensor products of the projectors in the single-qubit POVM. So for a system of two qubits, the POVM contains the elements $\hat{\Pi}_1^1 = |x_{\text{up}}\rangle\langle x_{\text{up}}| \otimes |x_{\text{up}}\rangle\langle x_{\text{up}}|$, $\hat{\Pi}_1^2 = |x_{\text{up}}\rangle\langle x_{\text{up}}| \otimes |x_{\text{down}}\rangle\langle x_{\text{down}}|$, and so on until $\hat{\Pi}_6^4 = |z_{\text{down}}\rangle\langle z_{\text{down}}| \otimes |z_{\text{down}}\rangle\langle z_{\text{down}}|$. For a system of N qubits, the POVM will contain $3^N \times 2^N = 6^N$ elements.

Each of the elements in a POVM set corresponds to a measurement outcome. The probability of getting the outcome corresponding to POVM element $\hat{\Pi}_i^j$ is

$$p_i^j = p(\hat{\Pi}_i^j|\hat{\rho}) = \text{tr}(\hat{\Pi}_i^j\hat{\rho}). \tag{2.1}$$

If we have all p_i^j probabilities then we can solve equation (2.1) for the unknown state $\hat{\rho}$.

Finding out those probabilities in the lab corresponds to going to our linear trap, preparing our (unknown) state by firing a sequence of laser pulses, then, for each ion, doing a measurement by firing a sequence of laser pulses to rotate the state to a particular Pauli basis setting, and finally recording which of the $\hat{\Pi}_i^j$ outcomes occurred. We then prepare the same state again by firing the same sequence of laser pulses, and we again do a measurement in the same Pauli basis setting as before. After doing many measurements in one basis setting, we move on the next and repeat the process. After finishing all the experiments for any one basis setting, we will know the frequency of occurrences of the outcome $\hat{\Pi}_i^j$, $f_i^j = \frac{n_i^j}{M}$, where n_i^j is the number of times that outcome occurred and M is total number of times we repeated the measurement in that particular setting. As M approaches infinity, the frequencies approach the exact probabilities, i.e. $f_i^j \to p_i^j$. Since we can never do infinite measurements in in the lab, we use the frequencies instead of the probabilities to solve equation (2.1) for the unknown state. Equation (2.1) then becomes

$$f_i^j = \mathrm{tr}(\hat{\Pi}_i^j \hat{\rho}). \tag{2.2}$$

This equation is in fact a linear system of 6^N equations, with one equation for each of the 6^N outcomes $\hat{\Pi}_i^j$.

2.2 Linear Inversion

The Linear Inversion method is the most straightforward method for "inverting" the frequencies f_i^j to get the state $\hat{\rho}$.

First we rewrite some equations in a more useful form: looking at equations (2.2) (one equation for each of the $\hat{\Pi}_i^j$ outcomes), we can write

$$f_i^j = \mathrm{tr}(\hat{\Pi}_i^j \hat{\rho}) = \sum_{x,y} (\hat{\Pi}_i^j)_{x,y} \hat{\rho}_{y,x} = \sum_{x,y} (\hat{\Pi}_i^j)_{x,y} (\hat{\rho}^T)_{x,y} = \sum_e \mathrm{vec}(\hat{\Pi}_i^j)_e \, \mathrm{vec}(\hat{\rho}^T)_e,$$

and recognizing the last part as a vector multiplications, we can write

$$f_i^j = \mathrm{vec}(\hat{\Pi}_i^j)^T \, \mathrm{vec}(\hat{\rho}^T).$$

We can then collect all 6^N equations into one by stacking all the $\mathrm{vec}(\hat{\Pi}_i^j)$ into a matrix $\boldsymbol{A}$ and stacking all the f_i^j into a column vector $\boldsymbol{f}$, i.e.,

$$\begin{bmatrix} f_1^1 \\ f_1^2 \\ \vdots \\ f_{3^N}^{2^N} \end{bmatrix} = \begin{bmatrix} \mathrm{vec}(\hat{\Pi}_1^1)^T \\ \mathrm{vec}(\hat{\Pi}_1^2)^T \\ \vdots \\ \mathrm{vec}(\hat{\Pi}_{3^N}^{2^N})^T \end{bmatrix} \mathrm{vec}(\hat{\rho}^T),$$

or

$$\boldsymbol{f} = \boldsymbol{A}\boldsymbol{\rho}, \tag{2.3}$$

where $\boldsymbol{\rho} = \mathrm{vec}(\hat{\rho}^T)$. Getting $\hat{\rho}$ can now be in principle be done by using the Moore-Penrose pseudoinverse of the matrix $\boldsymbol{A}$ under the requirement that the POVM set is informationally complete [14].

There are two main problems with using this method. The first is that, because the system of equations (2.3) is overcomplete, any noise in the measurements might cause the system to have no solution. The second problem is that even if a solution could be found despite of the noise in the measurements, e.g. by introducing a loss function [5], the solution is not guaranteed to be positive semi-definite.

Another reconstruction algorithm that can be used for QST is the Maximum Likelihood Estimation (MLE) algorithm [15].

2.3 Maximum Likelihood Estimation

The likelihood function of a quantum state is defined as the probability of getting n_i^j occurrences of the measurement outcomes represented by $\hat{\Pi}_i^j$, i.e.

$$\mathcal{L}(\hat{\rho}) = \prod_{i,j} (p_i^j)^{n_i^j} = \prod_{i,j} \mathrm{tr}(\hat{\Pi}_i^j \hat{\rho})^{n_i^j},$$

where the multinomial factor was dropped since it has no effect on the results; the $\hat{\Pi}_i^j$ form our POVM set. The likelihood function is maximized when the state $\hat{\rho}$ is the most likely positive semi-definite state that if we did the measurements on would return the outcome frequencies n_i^j. To find our lab state, we can also maximize the log of the likelihood function since the log is a strictly increasing function. The log-likelihood function is

$$log(\mathcal{L}(\hat{\rho})) = \sum_{i,j} n_i^j log(\mathrm{tr}(\hat{\Pi}_i^j \hat{\rho})).$$

The state maximizing the log-likelihood function is known to satisfy the extremal equation [15]

$$\hat{\rho} = \hat{R}(\hat{\rho})\hat{\rho},$$

where the operator $\hat{R}$ is defined as

$$\hat{R} = \frac{1}{M} \sum_{i,j} \frac{n_i^j}{\mathrm{tr}(\hat{\Pi}_i^j \hat{\rho})} \hat{\Pi}_i^j$$

and $M = \sum_{i,j} n_i^j$. The extremal equation can then be transformed to the fixed-point algorithm [15]

$$\hat{\rho}_{k+1} = \mathcal{N}[\hat{R}(\hat{\rho}_k)\hat{\rho}_k\hat{R}(\hat{\rho}_k)], \tag{2.4}$$

where $\mathcal{N}$ represents normalizing the state after each iteration. Note that the set we are optimizing over, the set positive semi-definite states with trace one, is a convex set.

The fixed-point algorithm can be restricted to modified to produce pure state estimates by writing

$$|\psi_{k+1}\rangle = \mathcal{N}[\hat{R}(|\psi_k\rangle)\,|\psi_k\rangle].$$

This, however, makes the state space non-convex, which introduces the possibility of the algorithm getting stuck in local maxima. In [5], however, it is shown that this modified algorithm can produce satisfactory results.

2.3.1 Diluted Maximum Likelihood Estimation

It is worth mentioning that although the MLE algorithm in equation (2.4) cannot be analytically proven to converge to the global maximum [17], the algorithm is generally observed to be convergent [5, 10, 15]. However, a generalized form of the algorithm can be proven to be convergent. More precisely, it can be proven that diluting the $\hat{R}(\hat{\rho})$ operator to $\frac{\mathbb{I}+\epsilon\hat{R}(\hat{\rho})}{1+\epsilon}$ guarantees that the algorithm converges when $\epsilon \ll 1$, at the expense of the rate of convergence. The proof is as follows [17].

The original MLE algorithm is

$$\hat{\rho}_{k+1} = \mathcal{N}[\hat{R}_k\hat{\rho}_k\hat{R}_k].$$

After diluting $\hat{R}$, the algorithm becomes

$$\hat{\rho}_{k+1} = \mathcal{N}[\frac{\mathbb{I}+\epsilon\hat{R}_k}{1+\epsilon}\hat{\rho}_k\frac{\mathbb{I}+\epsilon\hat{R}_k}{1+\epsilon}] \tag{2.5}$$

$$= \frac{1}{(1+\epsilon)^2}[\hat{\rho}_k + \epsilon\hat{\rho}_k\hat{R}_k + \epsilon\hat{R}_k\hat{\rho}_k + \epsilon^2\hat{R}_k\hat{\rho}_k\hat{R}_k],$$

where ϵ is a positive number. Approximating $(1+\epsilon)^{-2} \approx 1-2\epsilon$ and neglecting higher order terms in ϵ, we get

$$\hat{\rho}_{k+1} = \hat{\rho}_k + \Delta\hat{\rho}_k, \tag{2.6}$$

where $\Delta\hat{\rho}_k = \epsilon(\hat{\rho}_k\hat{R}_k + \hat{R}_k\hat{\rho}_k - 2\hat{\rho}_k)$. The log-likelihood function is then

$$log(\mathcal{L}(\hat{\rho}_{k+1})) = \sum_{i,j} n_i^j log(\mathrm{tr}(\hat{\Pi}_i^j\hat{\rho}_{k+1}))$$

$$= \sum_{i,j} n_i^j log(\mathrm{tr}(\hat{\Pi}_i^j\hat{\rho}_k)) + \sum_{i,j} n_i^j log(1 + \frac{1}{\mathrm{tr}(\hat{\Pi}_i^j\hat{\rho}_k)}\mathrm{tr}(\hat{\Pi}_i^j\Delta\hat{\rho}_k)),$$

and using the approximation $log(1+x) \approx x$ for $x \ll 1$, we can write

$$log(\mathcal{L}(\hat{\rho}_{k+1})) = log(\mathcal{L}(\hat{\rho}_k)) + \sum_{i,j}\frac{n_i^j}{\mathrm{tr}(\hat{\Pi}_i^j\hat{\rho}_k)}\mathrm{tr}(\hat{\Pi}_i^j\Delta\hat{\rho}_k)$$

$$= log(\mathcal{L}(\hat{\rho}_k)) + \mathrm{tr}(\hat{R}_k\Delta\hat{\rho}_k)$$

$$= log(\mathcal{L}(\hat{\rho}_k)) + \epsilon\,\mathrm{tr}(\hat{R}_k(\hat{\rho}_k\hat{R}_k + \hat{R}_k\hat{\rho}_k - 2\hat{\rho}_k)).$$

Then, using the cyclic property of the trace and the relation $\mathrm{tr}(\hat{R}\hat{\rho}) = 1$, we can write

$$log(\mathcal{L}(\hat{\rho}_{k+1})) = log(\mathcal{L}(\hat{\rho}_k)) + 2\epsilon(\mathrm{tr}(\hat{R}_k\hat{\rho}_k\hat{R}_k) - 1).$$

All that is left now to prove that the log-likelihood increases in each iteration (until it reaches the maximum) is to prove that $\mathrm{tr}(\hat{R}_k\hat{\rho}_k\hat{R}_k) \geq 1$. Any positive semi-definite matrix $\hat{\rho}$ has a unique positive square root $\rho^{1/2}$. We can use that fact to write

$$\mathrm{tr}(\hat{R}_k\hat{\rho}_k\hat{R}_k) = \mathrm{tr}(\hat{R}_k\hat{\rho}_k\hat{R}_k)\,\mathrm{tr}(\hat{\rho}_k)$$

$$= (\hat{R}_k\hat{\rho}_k^{1/2}, \hat{R}_k\hat{\rho}_k^{1/2})(\hat{\rho}_k^{1/2}, \hat{\rho}_k^{1/2}), \tag{2.7}$$

where the Hilbert-Schmidt scalar product $(\hat{A}, \hat{B})$ is defined as $(\hat{A}, \hat{B}) = \sum_{i,j} A_{i,j}^* B_{j,i} = \mathrm{tr}(\hat{A}^\dagger\hat{B})$. Furthermore, we can say

$$\mathrm{tr}^2(\hat{R}_k\hat{\rho}_k) = |(\hat{R}_k\hat{\rho}_k^{1/2}, \hat{\rho}_k^{1/2})|^2 = 1. \tag{2.8}$$

Relating equations (2.7) and (2.8) using the Cauchy-Schwarz inequality,

$$\mathrm{tr}(\hat{R}_k\hat{\rho}_k\hat{R}_k) = (\hat{R}_k\hat{\rho}_k^{1/2}, \hat{R}_k\hat{\rho}_k^{1/2})(\hat{\rho}_k^{1/2}, \hat{\rho}_k^{1/2}) \geq |(\hat{R}_k\hat{\rho}_k^{1/2}, \hat{\rho}_k^{1/2})|^2 = \mathrm{tr}^2(\hat{R}_k\hat{\rho}_k) = 1, \tag{2.9}$$

completes the proof.

2.4 Local Measurements

The POVM operators we consider are of the form $\hat{\Pi}_i^j = \hat{\pi}_{i_1}^{j_1} \otimes \hat{\pi}_{i_2}^{j_2} \otimes \cdots \otimes \hat{\pi}_{i_N}^{j_N}$, where for the Pauli measurement scheme (the one we are using) $j_k \in \{1, 2\}$, $i_k \in \{1, 2, 3\}$, and

$$\hat{\pi}_{i_k}^{j_k} \in \{|x_{up}\rangle \langle x_{up}| , |x_{down}\rangle \langle x_{down}| , \ldots , |z_{up}\rangle \langle z_{up}| , |z_{down}\rangle \langle z_{down}|\}.$$

If we have N qubits, one can see that our POVM set will contain 6^N elements. Furthermore, in order to obtain the frequencies n_i^j we will have do $3^N m$ measurements in the lab, where m is the number of times we measure each observable. It has been demonstrated, however, that one can get a good estimate of the state by only doing the measurements on blocks of contiguous sites instead of on all sites at once [6]. In fact, for pure states, it has been shown that an MPS can be completely reconstructed using local measurements if it belongs to the class of "injective" MPS (injectivity shall be explained in more details in later sections) [5]. This helps ameliorate the exponential measurement effort and computational cost by allowing us to make our measurement settings such that we measure only a block of r qubits, instead of all N, in each measurement. That is, we make our POVM elements $\hat{\Pi}_i^j = \mathbb{I} \otimes \cdots \otimes \mathbb{I} \otimes \hat{\pi}_{i_k}^{j_k} \otimes \hat{\pi}_{i_{k+1}}^{j_{k+1}} \otimes \cdots \otimes \hat{\pi}_{i_{k+r-1}}^{j_{k+r-1}} \otimes \mathbb{I} \otimes \cdots \otimes \mathbb{I}$. One can see that this way reduces the $3^N m$ measurements to $3^r m(N - r + 1)$ measurements.

Chapter 3

The Matrix Product Form

As mentioned before, Quantum State Tomography becomes exponentially expensive as the size of the system grows, because the state vector (matrix) grows exponentially and becomes harder to store and to do computations with. For a wide range of physically relevant states (section 3.6), however, the exponential growth of the state matrix can be avoided if the state is written in another form, namely, the Matrix Product form. In this chapter we derive the Matrix Product form of the state from its vector form (pure states) and matrix form (mixed state) (sections 3.1 and 3.2). Then in sections 3.3 and 3.4, we explain how one manipulate the Matrix Product form in calculations and do operations on it (addition, multiplication, expectation values, etc). In section 3.5, we introduce the PMPS form of a mixed state, laying the ground work for the comparison between the MPO-MLE and the PMPS-MLE to be carried out in chapter 5. Finally, in section 3.6, we give an overview of the types of states that can be written efficiently in their Matrix Product form. The derivations in this chapter are based on the derivations in [24].

3.1 Derivation

A pure quantum state of a system consisting of N sites can be written with a product basis as

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \Psi_{\sigma_1\sigma_2\ldots\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle ,$$

where Ψ is a column vector that contains the coefficients of $|\psi\rangle$ in the orthonormal product basis $|\sigma_1\sigma_2\ldots\sigma_N\rangle$. It is exactly this form that becomes cumbersome to handle during computations on large systems. Another form which is more efficient is called the Matrix Product State (MPS) form and is written as

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} A^{\sigma_1}A^{\sigma_2}\ldots A^{\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle ,$$

where the A^{σ_k} are sets of complex matrices except for A^{σ_1} and A^{σ_N}, which are sets of complex row and column vectors respectively, i.e. $A^{\sigma_k} \in \mathbb{C}^{D_k\times D_{k+1}}$, where $D_1 = D_N = 1$. Note that the A^{σ_k} are in general different, i.e. $A^{\sigma_k} \neq A^{\sigma_j}$ for $k \neq j$.

The MPS form of a pure state is derived from the standard form by sweeping over the chain of sites and doing Singular Value Decomposition (SVD) to the vector $\Psi_{\sigma_1\sigma_2\ldots\sigma_N}$ at each bi-partition. The procedure is as follows:

1. We reshape the vector $\Psi_{\sigma_1\sigma_2\ldots\sigma_N}$ into the $d \times d^{N-1}$ matrix $\Psi_{\sigma_1,\sigma_2\ldots\sigma_N}$.

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \Psi_{\sigma_1,\sigma_2\ldots\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \tag{3.1}$$

2. We do an SVD on that matrix:

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x} U_{\sigma_1,x_1} S_{x_1,x_1} V^{\dagger}_{x_1,\sigma_2\ldots\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,.$$

   S is a diagonal matrix that contains the singular values of our Ψ matrix. The rank of S is equivalent to the Schmidt rank of that particular bi-partition, and depending on the state, the S matrix can contain small singular values that can be removed without affecting the state. Indeed, the inefficiency of the regular state representation stems from the fact that those small singular values in the S matrices are not removed and, so, are taking up unnecessary room.

3. After tailoring S to contain only elements that are larger than a certain threshold on its diagonal, we have

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x'_1} U_{\sigma_1,x'_1} S_{x'_1,x'_1} V^{\dagger}_{x'_1,\sigma_2\ldots\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,.$$

   Then we reshape the U matrix into the 3-dimensional tensor$A^{\sigma_1}_{1,x'_1}$, and we multiply the S matrix into the $V^{\dagger}$ matrix to get

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x'_1} A^{\sigma_1}_{1,x'_1} \Psi_{x'_1,\sigma_2\ldots\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle$$

4. We then reshape $\Psi_{x'_1,\sigma_2\ldots\sigma_N}$ into $\Psi_{x'_1\sigma_2,\sigma_3\ldots\sigma_N}$ and we repeat the steps above until we finally reach the MPS form

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x'_1,\ldots,x'_{N-1}} A^{\sigma_1}_{1,x'_1} A^{\sigma_2}_{x'_1,x'_2} \ldots A^{\sigma_N}_{x'_{N-1},1} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,,$$

   or in short,

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} A^{\sigma_1} A^{\sigma_2} \ldots A^{\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,,$$

   where the summations over the bond indices (the x indices) have been recognized as matrix multiplications.

Recalling that $A^{\sigma_k} \in \mathbb{C}^{D_k \times D_{k+1}}$, the Bond Dimension of an MPS is defined as $\max_{\mathrm{k}} D_k$, or the maximum Schmidt rank across all possible bi-partitions of the system. Given its relation to the Schmidt rank, one can see how the bond dimension of an MPS can be a quantifier for how much entanglement exists in the system. Defined as $S = -\sum_i |\alpha_i|^2 log|\alpha_i|^2$, where α_i are the Schmidt coefficients, the von Neumann entropy is a well-known measure for entanglement of pure states. From this, one can conclude that as the entanglement between the state's bi-partitions increases, so does the bond dimension of the MPS used represent it.

The derivation of the Matrix Product Operator (MPO) form of the density matrix of a mixed state proceeds exactly as the derivation of the MPS form. A mixed state in its standard form is written as

$$\hat{\rho} = \sum_{\sigma_1\sigma'_1\sigma_2\sigma'_2\ldots\sigma_N\sigma'_N} \rho_{\sigma_1\sigma'_1\sigma_2\sigma'_2\ldots\sigma_N\sigma'_N} \left|\sigma_1\sigma_2\ldots\sigma_N\right\rangle \left\langle\sigma'_1\sigma'_2\ldots\sigma'_N\right| .$$

Like in the pure state case, this is usually not the most efficient form to write the density matrix, and a more efficient form is the MPO form. Briefly, the steps of deriving the MPO form are as follows:

1. Reshape the vector $\rho_{\sigma_1\sigma'_1\sigma_2\sigma'_2\ldots\sigma_N\sigma'_N}$ into the $d^2 \times d^{2(N-1)}$ matrix $\rho_{\sigma_1\sigma'_1,\sigma_2\sigma'_2\ldots\sigma_N\sigma'_N}$
2. Apply SVD to $\rho_{\sigma_1\sigma'_1,\sigma_2\sigma'_2\ldots\sigma_N\sigma'_N}$.
3. Tailor the S matrix to contain only singular values that are larger than a certain threshold, then multiply the S and the $V^\dagger$ matrices to get the matrix $\rho_{x',\sigma_2\sigma'_2\ldots\sigma_N\sigma'_N}$, and reshape the $U_{\sigma_1\sigma'_1,x'_1}$ matrix into the tensor $W^{\sigma_1,\sigma'_1}_{1,x'_1}$.
4. Repeat the above steps until the last site to get the MPO form

$$\hat{\rho} = \sum_{\sigma_1\sigma'_1\ldots\sigma_N\sigma'_N} \sum_{x'_1x'_2\ldots x'_{N-1}} W^{\sigma_1,\sigma'_1}_{1,x'_1} W^{\sigma_2,\sigma'_2}_{x'_2,x'_2} \ldots W^{\sigma_N,\sigma'_N}_{x'_{N-1},1} \left|\sigma_1\sigma_2\ldots\sigma_N\right\rangle \left\langle\sigma'_1\sigma'_2\ldots\sigma'_N\right| ,$$

or in short,

$$\hat{\rho} = \sum_{\sigma_1\sigma'_1\ldots\sigma_N\sigma'_N} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2} \ldots W^{\sigma_N,\sigma'_N} \left|\sigma_1\sigma_2\ldots\sigma_N\right\rangle \left\langle\sigma'_1\sigma'_2\ldots\sigma'_N\right| ,$$

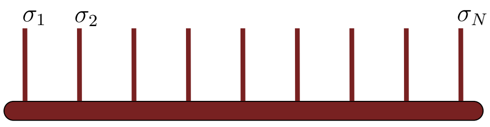
where the summations over the bond indices (the x indices) have been recognized as matrix multiplications.

The derivation outlined above can be represented using the convenient graphical representation that has become inseparable from the Matrix Product representation as the form was developed (see figure 3.1). As shall be demonstrated in the coming sections, the graphical representation can be a very useful tool for representing the basic operations that can be done using the Matrix Product form.

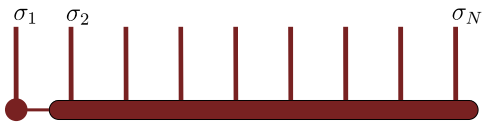
3.2 Normal Forms

The MPS derived in section 3.1 is called a left-normal MPS. An MPS is said to be left-normal if, for all l, $\sum_{\sigma_l} A^{\dagger\sigma_l} A^{\sigma_l} = \mathbb{I}$. This property follows directly from the fact that, for each SVD step we do in the derivation, the property $U^\dagger U = \mathbb{I}$ holds. Apart from the left-normal form, there is the right-normal form and the k-normal form. Each normal form can be used to make certain calculations easier. A demonstration of that can be seen in section 3.4.

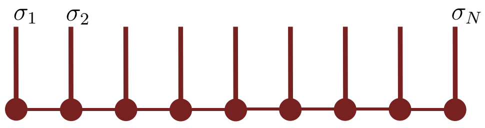
The difference between each normal form comes form the different derivation procedures. An MPS in its right-normal form is derived by doing the SVD starting from



(a) The state vector



(b) Matrix A^{σ_1} formed using SVD.



(c) All matrices A^{σ_i} formed. The state is now in its MPS form.

Figure 3.1: Graphical representation of the state-to-MPS conversion process. The vertical legs are referred to as "physical legs", and they represent the "physical" indices (σ indices) of each tensor $A^{\sigma_i}_{x_{i-1},x_i}$. The horizontal legs are called "bond legs" and they represent the "bond" indices (x indices). Connecting two legs of two tensors together represents taking the trace over the index corresponding to those legs.

the end of the chain: the vector $\Psi_{\sigma_1\sigma_2\ldots\sigma_N}$ is first reshaped into the matrix $\Psi_{\sigma_1\ldots\sigma_{N-1},\sigma_N}$ (compare that with equation (3.1)), and then after the first SVD the state becomes

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x_1} U_{\sigma_1\ldots\sigma_{N-1},x_1} S_{x_1,x_1} V^\dagger_{x_1,\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,,$$

then, like before, the singular value matrix S_{x_1,x_1} is truncated to retain only large singular values, and the columns of $U_{\sigma_1\ldots\sigma_{N-1},x_1}$ and the rows of $V^\dagger_{x_1,\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle$ are truncated accordingly, which changes the state to

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x'_1} U_{\sigma_1\ldots\sigma_{N-1},x'_1} S_{x'_1,x'_1} V^\dagger_{x'_1,\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,.$$

The new S and U are then multiplied together into the matrix $\Psi_{\sigma_1\ldots\sigma_{N-1},x'_1}$ and the matrix $V^\dagger_{x'_1,\sigma_N}$ is reshaped into the set of vectors $B^{\sigma_N}_{x'_1,1}$. The state then becomes

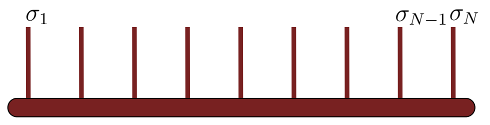
$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} \sum_{x'_1} \Psi_{\sigma_1\ldots\sigma_{N-1},x'_1} B^{\sigma_N}_{x'_1,1} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,.$$

The process is then continued for the rest of the chain until we finally get

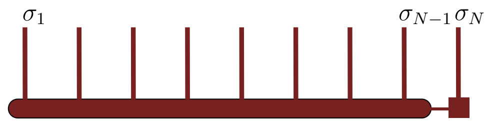
$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} B^{\sigma_1}B^{\sigma_2}\ldots B^{\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle \,,$$

where again it must pointed out that the sets B^{σ_k} are not the same for all k. This form is called the right-normal form because, for all l, $\sum_{\sigma_l} B^{\sigma_l}B^{\dagger\sigma_l} = \mathbb{I}$, which follows from $V^\dagger V = \mathbb{I}$.

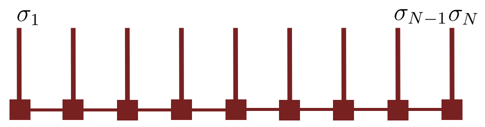
The right-normal and left-normal forms are both special cases of the more general k-normal form. The k-normal form is derived by doing SVD from both the right and left ends of the chain until the k-th site is reached, such that all matrices corresponding to



(a) The state vector



(b) Matrix B^{σ_N} formed using SVD.



(c) All matrices B^{σ_i} formed. The state is now in its MPS form.

Figure 3.2: Graphical representation of the process of converting a state vector to its right-normal MPS form.

sites after site k satisfy $\sum_{\sigma_l} B^{\sigma_l} B^{\dagger\sigma_l} = \mathbb{I}$ and all matrices corresponding to sites before site k satisfy $\sum_{\sigma_l} A^{\sigma_l} A^{\dagger\sigma_l} = \mathbb{I}$ (the matrix on site k can satisfy either conditions). An MPS in its k-normal form is written as

$$|\psi\rangle = \sum_{\sigma_1\sigma_2\ldots\sigma_N} A^{\sigma_1}\ldots A^{\sigma_{k-1}} A^{\sigma_k} B^{\sigma_{k+1}}\ldots B^{\sigma_N} |\sigma_1\sigma_2\ldots\sigma_N\rangle .$$

Note that the right normal-form is equivalent to the 0-normal form, and that the left-normal form is equivalent to the N-normal form.

All what has been explained in this section also applies directly to MPOs.

3.3 Operations with the Matrix Product Form

Adding two states, acting on a state with an operator, taking the inner product of two states, etc. can all be done directly on the MPS (MPO) form of the state, and are very conveniently represented using the graphical representation [24]. In this section we explain how the Matrix Product form can be used to do basic calculation, like a adding two quantum states, operating with a operator on a state, etc.

3.3.1 Addition

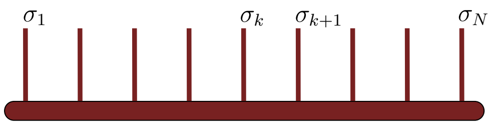
The addition of two MPO is an operation that we use in frequently in our reconstruction algorithm, as will be show in a later chapter.

Consider the states to be added

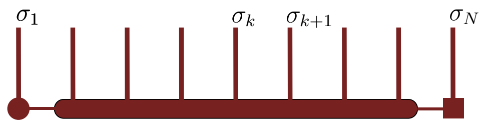
$$\hat{\rho}_1 = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2}\ldots W^{\sigma_N,\sigma'_N} |\sigma\rangle\langle\sigma'|$$

and

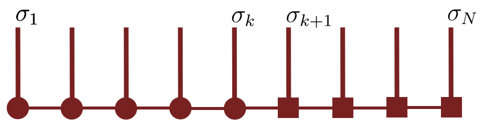
$$\hat{\rho}_2 = \sum_{\sigma,\sigma'} M^{\sigma_1,\sigma'_1} M^{\sigma_2,\sigma'_2}\ldots M^{\sigma_N,\sigma'_N} |\sigma\rangle\langle\sigma'| .$$



(a) The state vector



(b) Matrices A^{σ_1} and B^{σ_N} formed using SVD.



(c) All matrices A^{σ_i} formed. The state is now in its MPS form.

Figure 3.3: Graphical representation of the process of converting a state vector to its k-normal MPS form.

The addition happens as follows:

$$\hat{\rho}_3 = \hat{\rho}_1 + \hat{\rho}_2 = \sum_{\sigma,\sigma'} X^{\sigma_1,\sigma'_1} X^{\sigma_2,\sigma'_2} \dots X^{\sigma_N,\sigma'_N} \left|\sigma\right\rangle \left\langle\sigma'\right| ,$$

where $X^{\sigma_k} = W^{\sigma_k} \oplus M^{\sigma_k}$ for all sites but the first and the last. For those we form X by joining W and M in a row and column vector, respectively: $X^{\sigma_1} = [W^{\sigma_1} M^{\sigma_1}]$ and $X^{\sigma_N} = [W^{\sigma_N} M^{\sigma_1}]^T$. For the sites in the middle, the direct sum correspond to forming a diagonal matrix X^{σ_k} whose diagonal entries are the W and M matrices, i.e

$$X^{\sigma_k} = W^{\sigma_k} \oplus M^{\sigma_k} = \begin{bmatrix} W^{\sigma_k} & 0 \\ 0 & M^{\sigma_k} \end{bmatrix} .$$

As one can see, the bond dimension of the new matrix X^{σ_k} is the sum of the bond dimensions of the matrices W^{σ_k} and M^{σ_k}. And for the first and the last sites, the outermost bond dimensions remain 1 as they should but the middle bond dimensions add.

An important point to keep in mind is that, while the bond dimension of the resultant MPO is the sum of the bond dimensions of the summed MPOs, the new bond dimension might not be the optimal one and so one can compress the new MPO without changing the state (compression is explained in a later section). One can see how this is true by considering the case $\hat{\rho}_2 = \hat{\rho}_1 + \hat{\rho}_1 = 2\hat{\rho}_1$. Note that $\hat{\rho}_2$ is the same as $\hat{\rho}_1$ except for a factor of 2, and since scalar multiplications do not affect the bond dimensions, the bond dimension of $\hat{\rho}_2$, contrary to what the addition procedure explained above produced, cannot be more than the bond dimension of $\hat{\rho}_1$.

3.3.2 Operator Product

Suppose we have the mixed state $\hat{\rho} = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2} \dots W^{\sigma_N,\sigma'_N} \left|\sigma\right\rangle \left\langle\sigma'\right|$, and the operator $\hat{O} = \sum_{\gamma,\gamma'} X^{\gamma_1,\gamma'_1} X^{\gamma_2,\gamma'_2} \dots X^{\gamma_N,\gamma'_N} \left|\gamma\right\rangle \left\langle\gamma'\right|$, and we want to act with the operator on the state. This happens as follows:

$$\begin{aligned}
\hat{O}\hat{\rho} &= \sum_{\sigma,\sigma',\gamma,\gamma'} X^{\gamma_1,\gamma'_1}\ldots X^{\gamma_N,\gamma'_N} W^{\sigma_1,\sigma'_1}\ldots W^{\sigma_N,\sigma'_N} \left|\gamma\right\rangle \cancelto{\delta_{\gamma',\sigma}}{\left\langle\gamma'\middle|\sigma\right\rangle} \left\langle\sigma'\right| \\
&= \sum_{\gamma,\sigma',\gamma'} \sum_{y,x} X^{\gamma_1,\gamma'_1}_{1,y_1}\ldots X^{\gamma_N,\gamma'_N}_{y_{N-1},1} W^{\gamma'_1,\sigma'_1}_{1,x_1}\ldots W^{\gamma'_N,\sigma'_N}_{x_{N-1},1} \left|\gamma\right\rangle \left\langle\sigma'\right| \\
&= \sum_{\gamma,\sigma',\gamma'} \sum_{y,x} X^{\gamma_1,\gamma'_1}_{1,y_1} W^{\gamma'_1,\sigma'_1}_{1,x_1}\ldots X^{\gamma_N,\gamma'_N}_{y_{N-1},1} W^{\gamma'_N,\sigma'_N}_{x_{N-1},1} \left|\gamma\right\rangle \left\langle\sigma'\right| \\
&= \sum_{\gamma,\sigma'} P^{\gamma_1,\sigma'_1}\ldots P^{\gamma_N,\sigma'_N} \left|\gamma\right\rangle \left\langle\sigma'\right|, \qquad (3.2)
\end{aligned}$$

where $P^{\sigma_k,\gamma'_k}_{a_{k-1},a_k} = \sum_{\gamma'} X^{\gamma_k,\gamma'_k}_{y_{k-1},y_k} W^{\gamma'_k,\sigma'_k}_{x_{k-1},x_k}$, and $a_{k-1} = (x_{k-1}, y_{k-1})$ and $a_k = (x_k, y_k)$, i.e. multiplying two MPOs results in an MPO whose bond dimension is the product of the bond dimension of the multiplied MPOs.

Acting with an operator $\hat{O}$ on a pure state $|\psi\rangle$ follows in a similar fashion. As before, let the operator

$$\hat{O} = \sum_{\gamma,\gamma'} X^{\gamma_1,\gamma'_1} X^{\gamma_2,\gamma'_2}\ldots X^{\gamma_N,\gamma'_N} \left|\gamma\right\rangle \left\langle\gamma'\right|,$$

and let the state

$$|\psi\rangle = \sum_{\sigma} A^{\sigma_1} A^{\sigma_2}\ldots A^{\sigma_N} \left|\sigma\right\rangle.$$

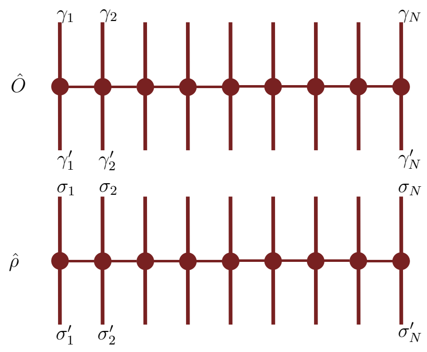
Then

$$\begin{aligned}
\hat{O}\left|\psi\right\rangle &= \sum_{\gamma,\gamma',\sigma} X^{\gamma_1,\gamma'_1}\ldots X^{\gamma_N,\gamma'_N} A^{\sigma_1}\ldots A^{\sigma_N} \left|\gamma\right\rangle \cancelto{\delta_{\gamma',\sigma}}{\left\langle\gamma'\middle|\sigma\right\rangle} \qquad (3.3) \\
&= \sum_{\gamma,\gamma'} \sum_{y,x} X^{\gamma_1,\gamma'_1}_{1,y_1} A^{\gamma'_1}_{1,x_1}\ldots X^{\gamma_N,\gamma'_N}_{y_{N-1},1} A^{\gamma'_N}_{x_{N-1},1} \left|\gamma\right\rangle \\
&= \sum_{\gamma} B^{\gamma_1}\ldots B^{\gamma_N} \left|\gamma\right\rangle,
\end{aligned}$$

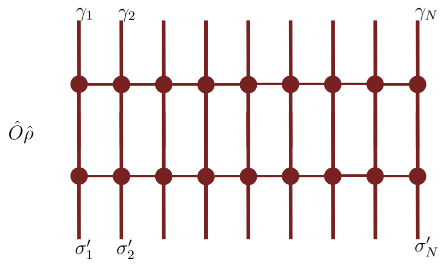
where

$$B^{\gamma_k}_{a_{k-1},a_k} = \sum_{\gamma'} X^{\gamma_k,\gamma'_k}_{y_{k-1},y_k} A^{\gamma'_k}_{x_{k-1},x_k},$$

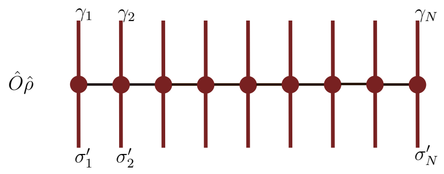
and $a_{k-1} = (x_{k-1}, y_{k-1})$ and $a_k = (x_k, y_k)$, i.e. the bond dimensions multiply. The graphical equivalent of the MPO-MPS and the MPO-MPO operations can be seen in figures 3.5 and 3.4.



(a) The top MPO represents the operator and the bottom MPO represents the state.



(b) Acting with the operator on the state: summing over physical indices is represented by connecting the corresponding physical legs together.



(c) The resultant MPO: the darker color of the bond legs (horizontal legs) represents the larger bond dimensions of the resultant MPO.

Figure 3.4: Operation of an operator on a mixed state. The resultant MPO has a bond dimension that is the product of the bond dimensions of the initial MPOs.

3.3.3 Inner Products, Expectation values, and Partial Traces

In this subsection we explain the details of some more Matrix Product manipulations that are used frequently in the the MLE algorithm.

(i) Inner Products. Suppose we have the two pure states

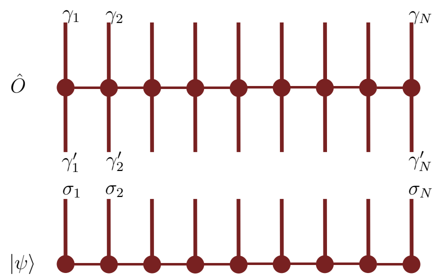
$$|\psi\rangle = \sum_{\sigma} A^{\sigma_1} A^{\sigma_2} \dots A^{\sigma_N} |\sigma\rangle$$

and

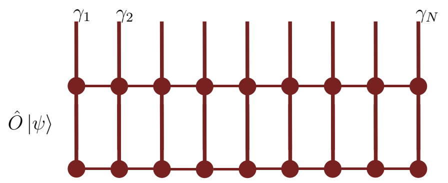
$$|\phi\rangle = \sum_{\sigma} \tilde{A}^{\sigma_1} \tilde{A}^{\sigma_2} \dots \tilde{A}^{\sigma_N} |\sigma\rangle \, ,$$

and we need to calculate their inner product $\langle\phi|\psi\rangle$. The calculation starts as

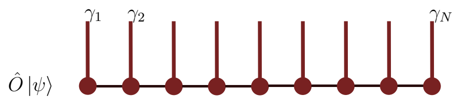
$$\langle\phi|\psi\rangle = \sum_{\sigma,\sigma'} (\tilde{A}^{\sigma'_1} \tilde{A}^{\sigma'_2} \dots \tilde{A}^{\sigma'_N})^\dagger A^{\sigma_1} A^{\sigma_2} \dots A^{\sigma_N} \overset{\delta_{\sigma,\sigma'}}{\cancel{\langle\sigma'|\sigma\rangle}}$$



(a) The top MPO is the operator and the bottom MPS is the pure state to be operated on.



(b) Acting with Operator on the state



(c) The resultant MPS

Figure 3.5: Operation of an MPO on an MPS. The resultant MPS has bond dimension $D_3 = D_1 \times D_2$, where D_1 is the bond dimension of the MPO and D_2 is the bond dimension of the MPS.

$$= \sum_{\sigma} (\tilde{A}^{\sigma_1}\tilde{A}^{\sigma_2}\dots\tilde{A}^{\sigma_N})^{\dagger} A^{\sigma_1}A^{\sigma_2}\dots A^{\sigma_N}$$

$$= \sum_{\sigma} \tilde{A}^{\sigma_N\dagger}\dots\tilde{A}^{\sigma_2\dagger}\tilde{A}^{\sigma_1\dagger}A^{\sigma_1}A^{\sigma_2}\dots A^{\sigma_N}. \tag{3.4}$$

How we approach the next step is crucial to the efficiency of any algorithm calculating the overlap between two MPSs. Assume for simplicity that all the A matrices are square matrices. If we compute the expression in equation (3.4) by first multiplying all the matrices and then summing over the physical indices, we will have done a total of d^N matrix multiplications and $d^N - 1$ matrix additions, which is exponentially expensive. If, on the other hand, we reorder the summing operations such that equation (3.4) becomes

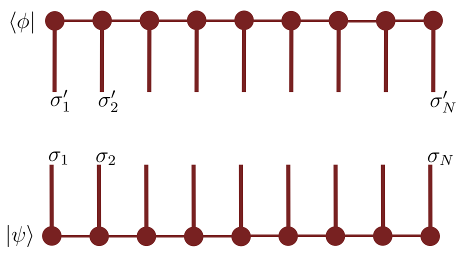
$$\langle\phi|\psi\rangle = \sum_{\sigma_N} \tilde{A}^{\sigma_N\dagger}\dots\Big(\sum_{\sigma_2}\tilde{A}^{\sigma_2\dagger}\Big(\sum_{\sigma_1}\tilde{A}^{\sigma_1\dagger}A^{\sigma_1}\Big)A^{\sigma_2}\Big)\dots A^{\sigma_N}. \tag{3.5}$$

Unlike with equation (3.4), computing equation (3.5) requires doing only Nd matrix multiplications and N matrix additions. Both approaches can be represented pictorially as in figure 3.6.

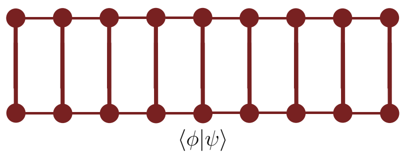
(ii) Expectation Values. Calculating expectation values of observables with respect to states written in their Matrix Product form is straightforward. Let us consider the pure case first. Suppose we have the pure state

$$|\psi\rangle = \sum_{\sigma} A^{\sigma_1}A^{\sigma_2}\dots A^{\sigma_N}\,|\sigma\rangle\,,$$

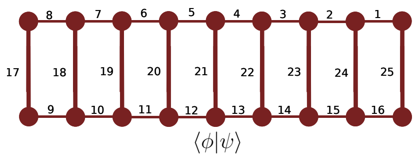
and the operator



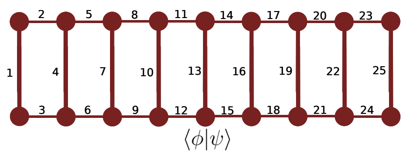
(a) Pure states: a bra is represented by an upside-down MPS.



(b) Inner product the two states



(c) Inefficient leg contraction



(d) Efficient leg contraction

Figure 3.6: Calculating the inner product of two states, there are two ways we can sum over the bond and physical indices. As explained in the text, the first way (equation (3.4)) is exponentially expensive, whereas the second way (equation (3.5)) is only polynomially expensive. Thus, the order in which we contract the indices is crucial to the efficiency of algorithms that use the Matrix Product form. Connected legs represent summing over indices.

$$\hat{O} = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2} \dots W^{\sigma_N,\sigma'_N} |\sigma\rangle \langle\sigma'| \, .$$

The expectation value of $\hat{O}$ with respect to $|\psi\rangle$ is then

$$\langle\hat{O}\rangle_\psi = \langle\psi| \hat{O} |\psi\rangle \, .$$

We then use equation (3.3) to calculate the action of the operator on the ket: $\hat{O}|\psi\rangle = |\phi\rangle$. We then get

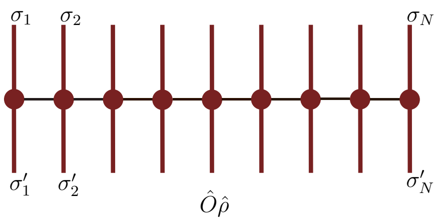
$$\langle\hat{O}\rangle_\psi = \langle\psi|\phi\rangle \, .$$

Finally, we use equation (3.5) to calculate the inner product.

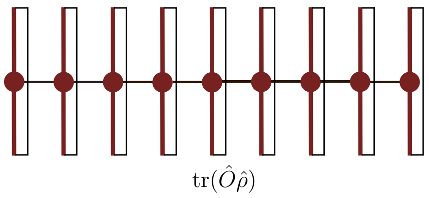
For the case of mixed states, the expectation value of an operator $\hat{O}$ with respect to a mixed state $\hat{\rho}$ is $\langle\hat{O}\rangle_{\hat{\rho}} = \mathrm{tr}(\hat{O}\hat{\rho})$. We can use equation (3.2) to calculate the action of $\hat{O}$ on $\hat{\rho}$ to get

$$\mathrm{tr}(\hat{O}\hat{\rho}) = \mathrm{tr}\Big(\sum_{\sigma,\gamma'} P^{\sigma_1,\gamma'_1} P^{\sigma_2,\gamma'_2} \dots P^{\sigma_N,\gamma'_N} |\sigma\rangle \langle\gamma'|\Big)$$

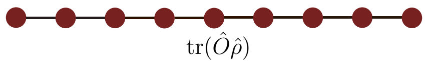
$$= \sum_{\sigma'} \sum_{\sigma,\gamma'} P^{\sigma_1,\gamma'_1} P^{\sigma_2,\gamma'_2} \dots P^{\sigma_N,\gamma'_N} \cancel{\langle\sigma'|\sigma\rangle} \nearrow \delta_{\sigma',\sigma} \quad \cancel{\langle\gamma'|\sigma'\rangle} \nearrow \delta_{\sigma',\gamma'}$$



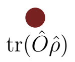
(a) The MPO that is to be traced



(b) taking the trace



(c) The result is a chain of matrices (no physical legs)



(d) Multiplying the matrices together gives a matrix with dimensions 1×1, i.e., a number.

Figure 3.7: Calculating the expectation value of an operator with respect to a mixed state. Tracing over the physical indices is represented connecting the physical legs of each site together.

$$= \sum_{\sigma} P^{\sigma'_1,\sigma'_1} P^{\sigma'_2,\sigma'_2} \dots P^{\sigma'_N,\sigma'_N}.$$

As before, we can reorder the sums to make the computation of the previous expression much more efficient:

$$\mathrm{tr}(\hat{O}\hat{\rho}) = \sum_{\sigma_1} P^{\sigma'_1,\sigma'_1} \sum_{\sigma_2} P^{\sigma'_2,\sigma'_2} \dots \sum_{\sigma_N} P^{\sigma'_N,\sigma'_N}.$$

This can be represented pictorially as in figure 3.7.

(iii) Partial Trace. Taking the partial trace of an MPO is an operation frequently used to reduce the state of the entire system to the state of a subsystem. Suppose we have the MPO

$$\hat{\rho} = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2} \dots W^{\sigma_N,\sigma'_N} \left|\sigma\right\rangle \left\langle\sigma'\right|,$$

and we want to trace out all sites except the second and third, i.e., we want to do $\hat{\rho}_{2,3} = \mathrm{tr}_{sites\neq 2,3}(\hat{\rho})$. To do that we write

$$\hat{\rho}_{2,3} = \mathrm{tr}_{sites\neq 2,3}(\hat{\rho}) = \mathrm{tr}_{sites\neq 2,3}(\sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2} \dots W^{\sigma_N,\sigma'_N} \left|\sigma\right\rangle \left\langle\sigma'\right|)$$

$$\hat{\rho}_{2,3} = \mathrm{tr}_{sites\neq 2,3}(\hat{\rho}) = \mathrm{tr}_{sites\neq 2,3}(\sum_{\sigma_1\sigma_1'\sigma_2\sigma_2'\dots\sigma_N\sigma_N'} W^{\sigma_1,\sigma_1'}W^{\sigma_2,\sigma_2'}\dots W^{\sigma_N,\sigma_N'}\left|\sigma_1\sigma_2\dots\sigma_N\right\rangle\left\langle\sigma_1'\sigma_2'\dots\sigma_N'\right|)$$

$$= \sum_{\gamma_1\gamma_4\cdots\gamma_N}\sum_{\sigma,\sigma'} W^{\sigma_1,\sigma_1'}W^{\sigma_2,\sigma_2'}\dots W^{\sigma_N,\sigma_N'}\left\langle\gamma_1\gamma_4\dots\gamma_N|\sigma_1\sigma_4\dots\sigma_N\right\rangle\left|\sigma_2\sigma_3\right\rangle\left\langle\sigma_2'\sigma_3'\right|\left\langle\sigma_1'\sigma_4'\dots\sigma_N'|\gamma_1\gamma_4\dots\gamma_N\right\rangle$$

$$= \sum_{\gamma_1,\gamma_4,\dots,\gamma_N}\sum_{\sigma_2\sigma_2'\sigma_3\sigma_3'} W^{\gamma_1,\gamma_1}W^{\sigma_2,\sigma_2'}W^{\sigma_3,\sigma_3'}W^{\gamma_4,\gamma_4}\dots W^{\gamma_N,\gamma_N}\left|\sigma_2\sigma_3\right\rangle\left\langle\sigma_2'\sigma_3'\right|$$

$$= \sum_{\gamma_1} W^{\gamma_1,\gamma_1}\sum_{\sigma_2\sigma_2'\sigma_3\sigma_3'}(W^{\sigma_2,\sigma_2'}W^{\sigma_3,\sigma_3'}\left|\sigma_2\sigma_3\right\rangle\left\langle\sigma_2'\sigma_3'\right|)\sum_{\gamma_4}W^{\gamma_4,\gamma_4}\dots\sum_{\gamma_N}W^{\gamma_N,\gamma_N}.$$

Figure 3.8 shows the pictorial representation of the partial trace.

3.4 Compression

As was just shown, doing any operation with a MPS/MPO increases the bond dimension of the MPS/MPO, making it an essential step in any algorithm utilizing these forms to compress the MPS/MPO back to a manageable bond dimension. There are two main methods of compressing an MPS or an MPO, singular value decomposition (SVD) compression and variational compression. In our simulations we use variational compression, which is the one explained in this section.

Suppose we have the state

$$\hat{\rho} = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma_1'}W^{\sigma_2,\sigma_2'}\dots W^{\sigma_N,\sigma_N'}\left|\sigma_1\sigma_2\dots\sigma_N\right\rangle\left\langle\sigma_1'\sigma_2'\dots\sigma_N'\right|$$

which has a large bond dimension D and we need to compress it to the smaller bond dimension D' using the variational compression method. To do that we assume there is a state $\hat{\rho}'$ whose bond dimension is $D' < D$, with

$$\hat{\rho}' = \sum_{\sigma,\sigma'} V^{\sigma_1,\sigma_1'}V^{\sigma_2,\sigma_2'}\dots V^{\sigma_N,\sigma_N'}\left|\sigma_1\sigma_2\dots\sigma_N\right\rangle\left\langle\sigma_1'\sigma_2'\dots\sigma_N'\right|.$$

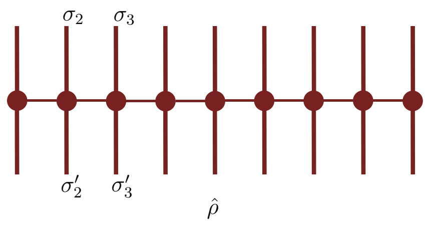
By minimizing the distance between $\hat{\rho}$ and $\hat{\rho}'$, i.e. minimizing

$$\|\hat{\rho}-\hat{\rho}'\|_F^2 = \mathrm{tr}(\hat{\rho}^\dagger\hat{\rho}) - \mathrm{tr}(\hat{\rho}\hat{\rho}') - \mathrm{tr}(\hat{\rho}'^\dagger\hat{\rho}) + \mathrm{tr}(\hat{\rho}'^\dagger\hat{\rho}'),$$

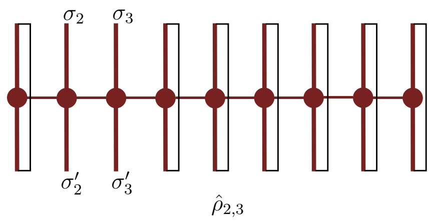
we find the state $\hat{\rho}'$ with bond dimension $D' < D$ which is closest to our original state $\hat{\rho}$. The minimization is done as follows:

$$\frac{\partial}{\partial V^{*\sigma_k,\sigma_k'}}\|\hat{\rho}-\hat{\rho}'\|_F^2 = \frac{\partial}{\partial V^{*\sigma_k,\sigma_k'}}(-\mathrm{tr}(\hat{\rho}'^\dagger\hat{\rho}) + \mathrm{tr}(\hat{\rho}'^\dagger\hat{\rho}')) = 0,$$

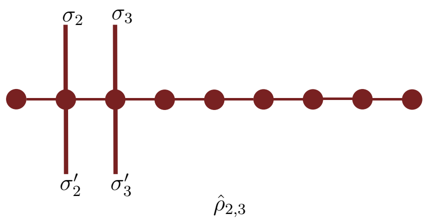
where the first two terms vanish because they don't contain the term $V^{*\sigma_k,\sigma_k'}$. Writing $\hat{\rho}'$ in its k-normal form (see section 3.2) greatly simplify the computation of the two derivatives. After writing $\hat{\rho}'$ in its k-normal form, the results of the derivatives are [5]



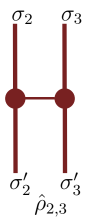
(a) The MPO that is to be traced



(b) taking the partial trace



(c) The trace results in matrices (no physical legs).



(d) The matrices are then multiplied into the tensors by taking the sums over the bond indices.

Figure 3.8: Reducing a mixed state to the state of a subsystem.

$$\frac{\partial}{\partial V^{*\sigma_k,\sigma'_k}} \operatorname{tr}(\hat{\rho}'^{\dagger}\hat{\rho}') = V^{\sigma_k,\sigma'_k}$$

and

$$\frac{\partial}{\partial V^{*\sigma_k,\sigma'_k}} (-\operatorname{tr}(\hat{\rho}'^{\dagger}\hat{\rho})) = -LW^{\sigma,\sigma'}R,$$

where

$$L = \sum_{\sigma,\sigma'} V^{\dagger\sigma_{k-1},\sigma'_{k-1}} \dots V^{\dagger\sigma_1,\sigma'_1} W^{\sigma_1,\sigma'_1} \dots W^{\sigma_{k-1},\sigma'_{k-1}}$$

and

$$R = \sum_{\sigma,\sigma'} W^{\sigma_{k+1},\sigma'_{k+1}} \dots W^{\sigma_N,\sigma'_N} V^{\dagger\sigma_N,\sigma'_N} \dots V^{\dagger\sigma_{k+1},\sigma'_{k+1}}.$$

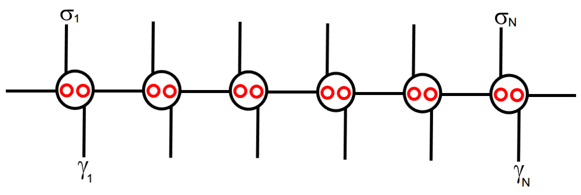


Figure 3.9: Each matrix A of the PMPS form represents a real and ancillary site.

Then

$$V^{\sigma_k,\sigma'_k} = LW^{\sigma_k,\sigma'_k}R.$$

If we sweep through the chain and replace each of the MPO matrices in every step, i.e. sweeping through all k, we get the compressed the MPO in the end. Practically, one sweep is enough to compress the MPO with out too much compression error. The same compression procedure is used to compress MPSs.

Since we do not use the SVD compression procedure, for details about it the reader is referred to [24]. The reason we opted for variational compression is mainly that the SVD produces a less accurate compressed state than that that the variational compression produces. Moreover, although variational compression is an iterative process and takes time to reach the optimal compressed state if an appropriate initial state is not provided, SVD can be shown to be more expensive than variational compression for large bond dimension [24].

3.5 Purified Matrix Product States

In this section we explain another form in which a mixed state can be written, namely its locally Purified Matrix Product State (PMPS) form.

A mixed state $\hat{\rho}$ for an N-sites systems can be written in its MPO form as

$$\hat{\rho}_{\text{lab}} = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1}W^{\sigma_2,\sigma'_2}\dots W^{\sigma_N,\sigma'_N} \left|\sigma_1\sigma_2\dots\sigma_N\right\rangle \left\langle\sigma'_1\sigma'_2\dots\sigma'_N\right| .$$

If we purify the system by adding ancillary sites, the pure state of the combined system can be written in its PMPS form as [28, 20]

$$\left|\psi_{\text{lab}}\right\rangle = \sum_{\sigma,\gamma} A^{\sigma_1,\gamma_1}A^{\sigma_2,\gamma_2}\dots A^{\sigma_N,\gamma_N} \left|\sigma_1\gamma_1\sigma_2\gamma_2\dots\sigma_N\gamma_N\right\rangle .$$

Each matrix A^{σ_l,γ_l} represents the l^{th} real site, whose state vector is $|\sigma_l\rangle$, and the l^{th} ancillary site, whose state vector is $|\gamma_l\rangle$. The arrangement of the sites in the purified system is such that each of the real sites has next to it an ancillary site (figure 3.9).

To obtain the mixed state of the real system, $\hat{\rho}_{\text{lab}}$, from the pure state of the purified system, $|\psi_{\text{lab}}\rangle$, we simply trace out the ancillary systems:

$$\hat{\rho}_{\text{lab}} = \text{tr}_{\text{ancilla}}(\left|\psi_{\text{lab}}\right\rangle \left\langle\psi_{\text{lab}}\right|)$$

$$= \text{tr}_\gamma(\sum_{\sigma,\gamma,\sigma',\gamma'} \sum_{x,x'} A^{\sigma_1,\gamma_1}_{1,x_1}\dots A^{\sigma_N,\gamma_N}_{x_{N-1},1} A^{\dagger\sigma'_N,\gamma'_N}_{x_{N-1}',1}\dots A^{\dagger\sigma'_1,\gamma'_1}_{1,x'_1} \left|\sigma,\gamma\right\rangle \left\langle\sigma',\gamma'\right|)$$

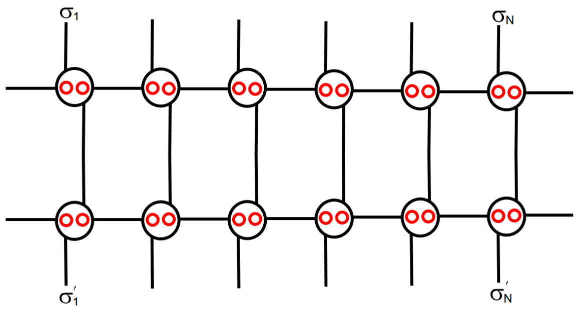


Figure 3.10: Reducing a PMPS to its corresponding MPO: connected physical legs represent taking the sum over the corresponding physical indices.

$$= \mathrm{tr}_\gamma\Big(\sum_{\sigma,\gamma,\sigma',\gamma'} \sum_{x,x'} A_{1,x_1}^{\sigma_1,\gamma_1} A_{1,x'_1}^{\dagger\sigma'_1,\gamma'_1} \dots A_{x_{N-1},1}^{\sigma_N,\gamma_N} A_{x_{N-1'},1}^{\dagger\sigma'_N,\gamma'_N} |\sigma,\gamma\rangle \langle\sigma',\gamma'|\Big),$$

and after computing the trace we get

$$\hat{\rho}_{\mathrm{lab}} = \sum_{\sigma,\sigma'} \sum_{y,y'} W_{1,y_1}^{\sigma_1,\sigma'_1} W_{y_1,y_2}^{\sigma_2,\sigma'_2} \dots W_{y_{N-1},1}^{\sigma_N,\sigma'_N} |\sigma\rangle \langle\sigma'| \,,$$

where

$$W_{y_{l-1},y_l}^{\sigma_l,\sigma'_l} = \sum_{\gamma_l} A_{x_{l-1},x_l}^{\sigma_l,\gamma_l} A_{x'_{l-1},x'_l}^{\dagger\sigma'_l,\gamma_l}, \tag{3.6}$$

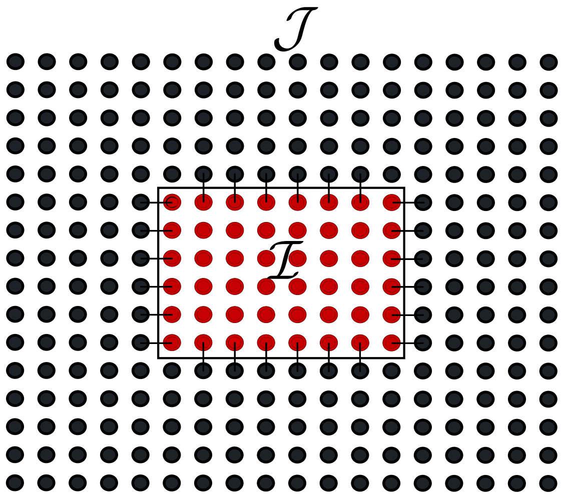
and $y_{l-1} = (x_{l-1}, x'_{l-1})$ and $y_l = (x_l, x'_l)$. Recognizing the summations over the bond indices as matrix multiplication, we can write the final result as

$$\hat{\rho}_{\mathrm{lab}} = \sum_{\sigma,\sigma'} W^{\sigma_1,\sigma'_1} W^{\sigma_2,\sigma'_2} \dots W^{\sigma_N,\sigma'_N} |\sigma\rangle \langle\sigma'| \,.$$

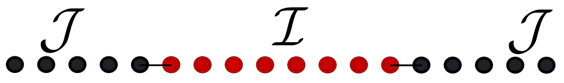
Note that state matrices that are obtained this way are, due the step in equation (3.6), positive semi-definite. Because this step is done at the end of the PMPS-MLE algorithm, the PMPS-MLE algorithm, unlike the MPO-MLE algorithm, always produces a positive semi-definite state.

3.6 Approximability by the Matrix Product Form

Using the derivation steps in section 3.1, we can write any state in its Matrix Product form. For a generic state, however, the number of parameters of the Matrix Product form of the state grows exponentially with the size of the system, and so the Matrix Product form in that case is at best equally efficient as the vector (or matrix) form ([16, 8]). Only if a state satisfies certain criteria can it be written efficiently in its Matrix Product form, i.e. as a Matrix Product state (or operator) with a small bond dimension. Although such criteria are not fully understood [26], some criteria have been shown to be reliable in making a conclusion about whether a state can be efficiently written as Matrix Product states. These criteria are related to the amount of correlation in the system. As can be seen from the steps in section 3.1, the smaller the Schmidt rank for any bi-partition of



(a) Area law for a 2D system



(b) Area law for a 1D system

Figure 3.11: For states that obey an area law, only the sites on the boundary of a region are entangled with the systems outside the region. In the upper figure, the inner rectangle represents a small region of the system (large rectangle). Only the elements on the boundary of the inner rectangle are entangled to the elements outside, so the entanglement entropy scales with the perimeter of the inner rectangle rather than with its area. Similarly, in a 1D system only neighboring sites are entangled to each other, so the entanglement entropy is constant rather than scaling with the lengths of the two regions.

the system is, the smaller the bond dimension of the MPS form of the state will be. By using the bond dimension to upper bound the von Neumann entropy of the system, also referred to as the entanglement entropy, the Schmidt rank for pure states is a measure that can be used to quantify the amount of correlation for a pure state [5]. Pure states that can be written as low bond dimension MPS have been shown to have Rényi entropies whose scaling with the size of the system obeys what is known as an "area law" [5].

3.6.1 Area Law

Suppose you have a many-body system which consists of the subsystems $\mathcal{I}$ and $\mathcal{J}$. The state of the system is said to obey an area law, and hence have a small amount of correlations, if only the sites at the boundary between subsystems $\mathcal{I}$ and $\mathcal{J}$ are strongly correlated. A consequence of that is that if it is a 3D system we are talking about, with subsystem $\mathcal{I}$ constituting a certain volume of the system and subsystem $\mathcal{J}$ constituting the rest of the system, then the correlations between the subsystem $\mathcal{I}$ and subsystem $\mathcal{J}$ scale not with the volume occupied by system $\mathcal{I}$, but with the surface area of that volume of sites, hence the name "area law". Similarly, if the system we are talking about is a 2D arrangement of sites, then the quantity of correlation between subsystem $\mathcal{I}$, which constitutes a certain area of the system, and subsystem $\mathcal{J}$, which constitutes the rest of the system, scales with the perimeter of the area occupied by subsystem $\mathcal{J}$. Since we are interested in linear arrangements of sites in this thesis (MPS and MPO are representations for 1D systems), the quantity of correlations between two subsystems, i.e. the Rényi entropies (for pure states), in a 1D systems are constant with the subsystem size. That is, if the state on the 1D system perfectly satisfies an area law then the bond dimension of the MPS will be independent of the size of the subsystems.

The relation between the bond dimension of a state and how well it follows an area law suggests that ground states of Hamiltonians that have a short range of interaction (local Hamiltonians) can be represented efficiently in the Matrix Product form. Indeed, it is now well known that pure states that are unique ground states of gapped R-local Hamiltonians (e.g. the Ising Hamiltonian) have an efficient MPS representation [8, 5]. Furthermore, thermal states of such Hamiltonians have been observed to have an efficient MPO representation (e.g. [28, 9]).

Other examples of states that have efficient matrix product representations are states like GHZ states, W states, and cluster states [12].

Chapter 4

Quantum State Tomography using Matrix Product Operators

The MLE algorithm explained in section 2.3 can be combined with the MPO representation in chapter 3, leading to a much more efficient QST algorithm due to the advantages of the Matrix Product form of the state [5]. We refer to this version of the MLE algorithm as the MPO-MLE algorithm, and in this chapter we detail how the algorithm works (section 4.1). Moreover, we discuss several issues with the MPO-MLE algorithm (section 4.2).

4.1 Maximum Likelihood Estimation using Matrix Product Operators

The MLE algorithm outlined in section 2.3 is given by the iterations prescription

$$|\psi_{k+1}\rangle = \mathcal{N}[\hat{R}(|\psi_k\rangle)\,|\psi_k\rangle] \tag{4.1}$$

for pure states, and

$$\hat{\rho}_{k+1} = \mathcal{N}[\hat{R}(\hat{\rho}_k)\hat{\rho}_k\hat{R}(\hat{\rho}_k)] \tag{4.2}$$

for mixed states. Under the assumption that the pure (mixed) state we are trying to reconstruct can be accurately approximated by its MPS (MPO) representation (see section 3.6) – strictly speaking, we are also assuming that that all states which the MLE could land on can be approximated by their MPS (MPO) representations –, equations 4.1 and 4.2 can be made to be much less computationally demanding by writing the operator $\hat{R}$ in its MPO form, the pure state vector $|\psi_k\rangle$ in its MPS form, and the mixed state matrix $\hat{\rho}_k$ in its MPO form. Without this modification, the MLE tomography scheme becomes exponentially hard to do as the size of the physical system grows due to the exponential growth of the number of parameters in the state vector (matrix) and the amount of computational resources needed to store them and do computations with them. From now on we will refer to the modified MLE algorithm on pure states (equation 4.1) as MPS-MLE and the modified MLE algorithm on mixed states (equation 4.2) as MPO-MLE.

4.1.1 The MPO-MLE Algorithm

The steps for doing MPO-MLE are as follows:

1. Do the measurement experiments and collect the outcome frequencies f_i^j corresponding to the POVM elements $\hat{\Pi}_i^j$.

2. choose the completely mixed state as the initial point for the MLE.

3. Do the iteration: $\hat{\rho}_{k+1} = \mathcal{N}[\hat{R}(\hat{\rho}_k)\hat{\rho}_k\hat{R}(\hat{\rho}_k)]$, where

$$\hat{R}(\hat{\rho}_\mathrm{k}) = \sum_{i,j} \frac{f_i^j}{\langle \hat{\Pi}_i^j \rangle_{\rho_\mathrm{k}}} \hat{\Pi}_i^j. \tag{4.3}$$

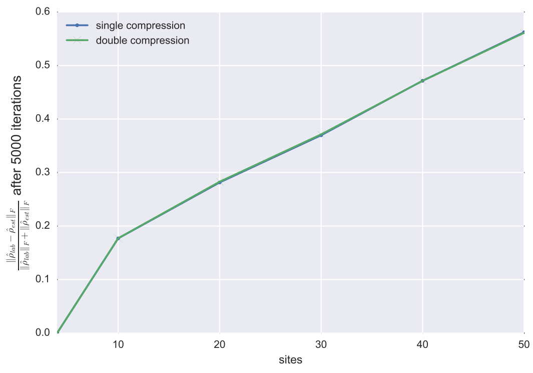
4. Compress the estimate state:

$$\hat{\rho}_{k+1} \rightarrow \mathcal{C}[\hat{\rho}_{k+1}],$$

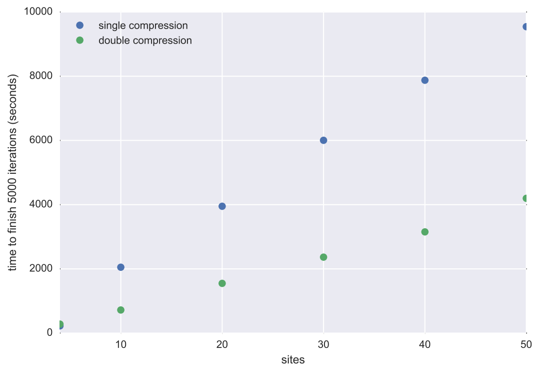
where $\mathcal{C}[.]$ represents doing compression to what is in the brackets.

The operator $\hat{R}$, as can be seen from equation (4.3), can be formed by adding all $6^r(N-r+1)$ weighted POVM elements together. Since the POVM elements are local operators and are of the tensor product form, they each have bond dimension 1, and since adding MPOs adds the bond dimensions, this suggests that the R operator has bond dimension $6^r(N-r+1)$, which scales polynomially in system size. In practice, we observe that $\hat{R}$ can be usually compressed to an even smaller bond dimension, which leads us to safely conclude that $\hat{R}$ can be represented efficiently as an MPO.

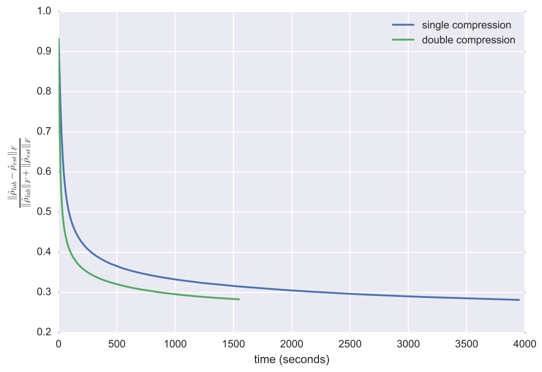
As in the regular MLE algorithm, the next step after collecting the outcome frequencies from the measurement experiments is to start the iterations with a completely mixed state, to be as unbiased as possible. In the regular MLE, the $\hat{R}$ operator and the matrix $\hat{\rho}_k$ have the same dimensions, and multiplying them results in a matrix of the same dimensions. In the MPO-MLE, on the other hand, multiplying $\hat{R}$ (now an MPO with bond dimension D_1) by the state matrix $\hat{\rho}$ (now an MPO with bond dimension D_2) results in an MPO with bond dimension $D_3 = D_1 \times D_2$. Compressing the new MPO to a small bond dimension is an essential process, because without compression the bond dimension quickly grows with each iteration and the MPO-MLE becomes as inefficient as the regular MLE. Compression is also essential in MPS-MLE since operating with $\hat{R}$ (MPO with bond dimension D_1) on $|\psi\rangle$ (MPS with bond dimension D_2) results in an MPS with bond dimension $D_3 = D_1 \times D_2$. Compression is especially crucial in MPO-MLE, however, because in each iteration we perform two MPO-MPO multiplications. In the regular MLE, the result of each iteration is a vector with length 2^N (in case we are dealing with a pure state), or a matrix with dimensions $2^N \times 2^N$ (in case we are dealing with a mixed state), where N is the number of qubits in the system; we know beforehand what the sizes of the matrices are. In the MPS-MLE or the MPO-MLE algorithms, on the other hand, the situation is not so simple since the bond dimension (which is practically the size of the matrices of the MPO or MPS form) of the state that comes out of each iterations is determined by the bond dimension to which we do our compression. Adding that to the fact that, in general, we do not know what the bond dimension of the lab state is and, as shall be shown in subsection 4.2.3, compressing to a too small bond dimension will lead to an estimate state that is not positive semi-definite, choosing an appropriate compression bond dimension is a step that has to be done with care. Apart from a few families of states (e.g. GHZ states), we never know exactly what the bond dimension of the lab state is, and, at best, we can estimate the lab state's bond dimension based on



(a)



(b)



(c) 20 Sites

Figure 4.1: The speed advantage of the double compression scheme over the single compression scheme becomes increasingly significant as the system size increases, since doing the additional compression saves a lot of computational cost in the second MPO-MPO multiplication. At the same time, the double compression approach converges to the same state as the no-middle-compression scheme. The block width is 4; the lab state MPO was generated by multiplying a random PMPS of bond dimension 3 with its adjoint. Compression in each iteration is done to the bond dimension of the lab state.

the number of sites in the system and the type of Hamiltonian under which the state was evolved.

In the next subsection we explore a modification of the compression scheme used in [6]. Namely, in the MPO-MLE we do two compression steps in each iteration instead of one. We show that this make the MPO-MLE significantly faster.

4.1.2 Compression

In the original paper where MPO-MLE was first introduced [6], compressing the estimate state $\hat{\rho}_{k+1}$ was done once at the end of the iteration, i.e. $\hat{\rho}_{k+1} = \mathcal{C}[\hat{R}\hat{\rho}_k\hat{R}]$, where $\mathcal{C}[.]$ represents doing a compression to what is in the square brackets. However, the algorithm can be made to be significantly faster if an additional compression is done after the multiplication of the first $\hat{R}$; i.e. $\hat{\rho}_{k+1} = \mathcal{C}[\mathcal{C}[\hat{R}\hat{\rho}_k]\hat{R}]$. A comparison of our double compression approach and the single compression approach from [6] is shown in figure 4.1. In addition to converging to the same state, the double compression approach is clearly faster than the single compression approach. In the following we use the double compression scheme.

4.2 Reasons for Convergence Failure

The goal of the MLE algorithm is to obtain the state which maximizes the log likelihood function for a given set of measurement outcomes. In each iteration, the MLE algorithm jumps from the current estimate state to another one in the state space until, ideally, it reaches our lab state. Given a set of measurements, the state that the MLE algorithm finally returns might or might not be the labs state, depending on the several factors we are going to discuss in this section.

4.2.1 Local Maxima

One of the reasons the MPO-MLE algorithm could fail to produce a good estimate of the lab state is the possibility of getting stuck in local maxima. The MLE algorithm functions by finding the global maximum of the log-likelihood function (chapter 2). Since the log function is a concave function and is strictly increasing, and the state space in the regular MLE (MLE without MPO) is convex, the algorithm can never get stuck in local maxima since none can exist [15]. In the MPO-MLE on the other hand, the situation requires more caution. If the bond dimension of the state in its MPO form is not maximum, i.e. not $d^{2\lfloor \frac{N}{2} \rfloor}$, the state space is not guaranteed to be convex anymore, which means that the MPO-MLE algorithm might get stuck in local maxima and fail to reach the global maximum.

To see why this is true, consider the following case. Suppose one has the two states $\hat{\rho}_1$ and $\hat{\rho}_2$, both of which have bond dimension D which is less than the maximum allowable bond dimension $d^{2\lfloor \frac{N}{2} \rfloor}$ that the states could have for a given number of sites N; i.e $\hat{\rho}_1, \hat{\rho}_2 \in \mathcal{M}_D$, where $\mathcal{M}_D$ is the space of MPOs with bond dimension D. Because it is possible that a state $\hat{\rho}_3$ on the line $(1-\lambda)\hat{\rho}_1 + \lambda\hat{\rho}_2$, $\lambda \in (0,1)$, will have a bond dimension of $2 \times D$ (see section 3.3.1) and so $\hat{\rho}_3 \notin \mathcal{M}_D$, the space $\mathcal{M}_D$ is not convex. A simple example of that is the mixed state $\hat{\rho} = \frac{1}{2}(|00\rangle\langle 00| + |11\rangle\langle 11|)$. Although this state is formed from two states with bond dimension 1, it has bond dimension 2.

Thus, anytime the MPO-MLE algorithm is working on a state space with less than the maximum bond dimension, it is at risk of getting stuck in a local maximum.

4.2.2 Small Measurement Block Size

As has been explained in subsection 2.4, we can make the MLE algorithm much faster by choosing that our measurements are done on a small block of contiguous sites rather than on the whole system. By choosing that our measurements are done on a small block, we change the scaling of the number of measurements we have to make from scaling exponentially with the system size to scaling polynomially with the system size. The speedup follows from the fact that the $\hat{R}$ operator becomes faster to calculate and has a smaller bond dimension than before (because less summands are used to form it). This speedup, however, comes at a price. Namely, a state's reductions to a certain block size do not, in general, uniquely determine the state, and in turn, neither do the expectation values or outcome frequencies of local operators or measurements done on these blocks. The practical result of this fact is that, even if we perform infinite measurements, we can never reconstruct the lab state if we are doing the measurements on a block that is too small. Although the MLE might converge to some final state, if the block size is smaller than a certain minimum, that final state will likely not be the lab state we are trying

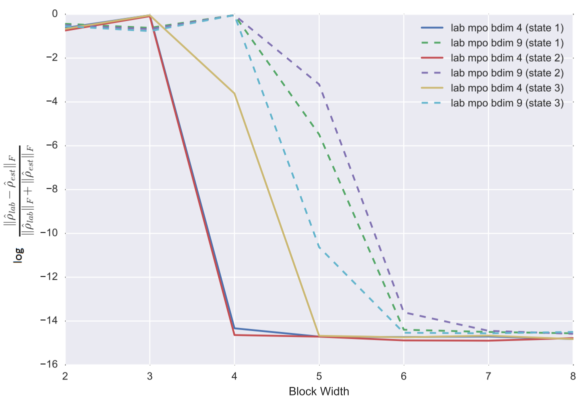


Figure 4.2: The larger the bond dimension of the lab state becomes, the larger the measurement block size has to be in order to reconstruct the state perfectly. The system simulated in this figure consists of 8 qubits; the lab state MPOs were generated by multiplying random PMPSs of bond dimensions 2 and 3 with their adjoints; each data point is acquired after doing 100000 iterations.

to reconstruct. What block size is big enough to determine the state uniquely by all its reductions? In case we are talking about pure states (MPS): It can be shown that if the state is the unique ground state of an r-local Hamiltonian we can uniquely determine the state by doing measurements on blocks of size r, with a necessary condition on r that $d^r \geq D^2$, where $d = 2$ if we are dealing with qubits and D is the bond dimension ([5, 25]). Such states are called “injective” states. Note from the previous condition that the block width is bound from below by the bond dimension of the MPS. For MPOs, although it has also been shown in [5] that we can use local information to reconstruct a state, we cannot guarantee that such state will be unique, i.e. it might be that many states can have the same local information. An analogous proof to the MPS case is not possible since it is not yet known whether MPOs have analogous objects to the parent Hamiltonians in the MPS case. Still, from figure 4.2 we can see that it remains true that a state with a larger bond dimension will require local information on larger blocks to be successfully estimated.

Furthermore, we conclude from figure 4.3 that in case the block width we are using is smaller than the minimum suggested by figure 4.2 (block width of 6 for a lab state with bond dimension 9), the MLE for larger systems will converge to a final state that is farther away from the lab state than the state that the MLE on a smaller system would converge to. This implies that, for a fixed block width, increasing the number of sites will decrease the reconstruction quality, until at some point the quality becomes too low for the MLE to be usable at that block width; for example, we can not use a block of width 3 for a system with size 10 and a system with size 200, because for the 200-qubits system the reconstructed state will probably not be anywhere near the lab state (the relative error is already more than 50 percent for a 30-qubits system, figure 4.3). It then follows that this approach of using r-local blocks for the measurement experiments does not make the MLE as scalable as one might think. The scalability of the MPO-MLE to larger systems is examined in detail in section 5.3, where it will also be shown that even if the block width is more than the required minimum that is suggested by figure 4.2, the MLE for larger systems returns a state that has a larger error than the MLE for smaller systems.

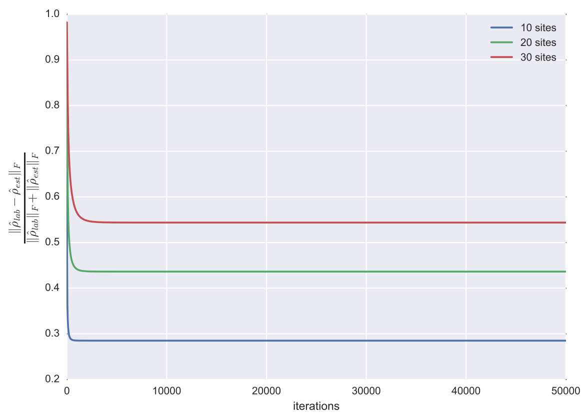


Figure 4.3: In the regime where the block size is smaller than a certain minimum, the terminal state for larger systems will be farther from the lab state. The block width is 3; the lab state MPO was generated by multiplying a random PMPS of bond dimension 3 with its adjoint.

4.2.3 Compression Error and Negative Eigenvalues

In this subsection we discuss a problem that is observed for MPO-MLE, so in this subsection we will not be mentioning MPS-MLE. This does not make the observations any less important since in a real tomography experiment, the state of the system is almost always a mixed state.

Assuming that the measurement block size is large enough, we are lucky enough to not get stuck in any local maxima, and we have done infinite measurements (which is practically never the case), we are still at risk of running into a serious problem. While the MLE algorithm preserves the positivity of the estimate, it does not enforce positivity. That is, if for some reason the estimate acquires a negative eigenvalue at some point during the iterations, the estimate state after the next iteration will most likely not be positive semi-definite. Since the initial state we start the iterations with is positive semi-definite, the only possible way negative eigenvalues can be introduced to the estimate state is during compression. As we are compressing from a large bond dimension to a small bond dimension, we essentially discard some of the singular values of the state we are compressing. The larger the singular values we drop, the larger the compression error is going to be, i.e. the farther the compressed state will be from the uncompressed state. More often than not, compression error introduces small negative eigenvalues in the estimate state that grow in magnitude as we do further compression in subsequent iterations. An example of that is shown in figure 4.4. The reason that compression error happens in the first place is that the MPO produced in each iteration does not, in general, have the bond dimension that we compress to. That is, even if a lab state has bond dimension 4, the estimate state in one of the iterations might have bond dimension 10. From figure 4.4 we see that this is true since if we compress that estimate state back to bond dimension 4, the resultant state will likely have at least one negative eigenvalue. The figure shows us that as we increase the compression bond dimension, causing the compression error to become less severe, the negative eigenvalues are not as large as when we compress to smaller bond dimension. From figure 4.4, we also see that the introduction of negative eigenvalues into the iterations often has the effect of destabilizing the MLE algorithm in the sense that the MLE starts to exhibit divergent behavior. When we compare the MPO-MLE and the PMPS-MLE in section 5.2 we will see that such instabilities can become so extreme that the estimate state completely diverges away from the lab state. Note that even for the diluted MLE algorithm (equation (2.5)) the introduction of negative eigenvalues in the estimate state breaks the convergence guarantee, since a key requirement in the proof in subsection 2.3.1 is that the state $\hat{\rho}_k$ is positive semi-definite.

Since compression is essential in making the MPO-MLE efficient, it seems inevitable that the MPO-MLE will produce an estimate with a negative eigenvalue. Despite of that, we can still hope to use the MPO-MLE to get an approximately positive state estimate if we stop the iterations before the negative eigenvalues grow in magnitude too much. But since no general efficient algorithms for checking the positivity of the state in its MPO form can exist ([19, 20]), we have to rely solely on the behavior of the MLE to recognize when negative eigenvalues start to be a problem (the eigenvalue plots in figure 4.4 are possible since the size of the system is small). From the plots in figure 4.4, it is clear that when the magnitude of the most negative eigenvalue becomes comparable (by a factor of about 10^2) to the magnitude of the largest positive eigenvalue, the relative norm distance curve stops decreasing and might even exhibit sharp increases like the ones shown in the plot, i.e. the estimate state moves away from the lab state. This suggests a technique that we can use to ameliorate the negativity of the state at the end of the MLE algorithm: we can monitor the distance between the current estimate state and the lab state and stop the iterations once the curve starts increasing. We cannot use the relative Frobenius norm distance between the estimate state and the lab state to do this monitoring since in a real QST experiment we do not know the lab state to begin with, so we have to find another measure that does not require knowledge of the lab state but exhibits similar behavior as the relative Frobenius norm distance when negative eigenvalues become a problem. One measure that we can use is the sum of the differences between the POVM outcome frequencies for the lab state and the estimate state, i.e. $\sum_{i,j} |f^j_{i,\text{lab}} - p^j_{i,\text{est}}|$. We already know $f^j_{i,\text{lab}}$ from the measurement experiments we did on our system, and the values of $p^j_{i,\text{est}}$ can be efficiently calculated for the MPO form of the estimate state using $p^j_{i,\text{est}} = \text{tr}(\hat{\Pi}^j_i \hat{\rho}_\text{est})$, where $\hat{\Pi}^j_i$ are the POVM elements for our Pauli measurement scheme. If we assume that we did a large enough number of measurements and the outcome frequencies we acquired from the measurement experiments are close enough to the true probability, i.e. $f^j_{i,\text{lab}} \approx \text{tr}(\hat{\Pi}^j_i \hat{\rho}_\text{lab})$, then

$$\sum_{i,j} |f^j_{i,\text{lab}} - p^j_{i,\text{est}}| \tag{4.4}$$

$$\approx \sum_{i,j} |\,\text{tr}(\hat{\Pi}^j_i \hat{\rho}_\text{lab}) - \text{tr}(\hat{\Pi}^j_i \hat{\rho}_\text{est})| = \sum_{i,j} |\,\text{tr}(\hat{\Pi}^j_i (\hat{\rho}_\text{lab} - \hat{\rho}_\text{est}))|.$$

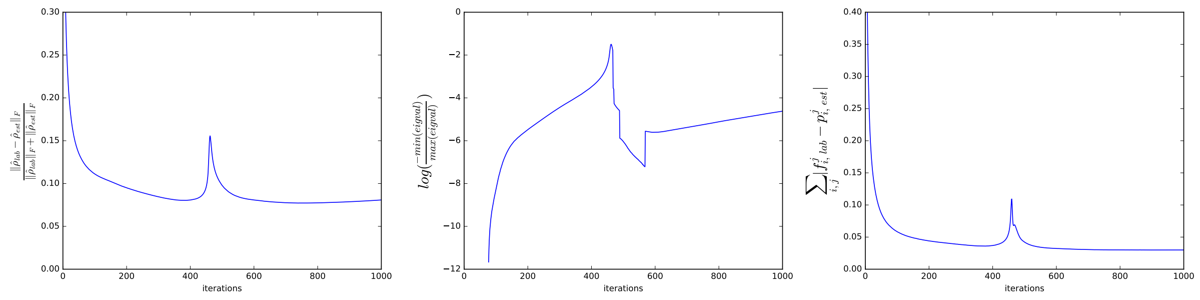
Looking at figure 4.4, we can see that the previous expression indeed behaves similarly to the relative Frobenius norm distance between the lab state and the estimate state. Then, as soon as an increase in the curve is detected, we can stop the iterations and, to ensure that the negative eigenvalues of the estimate are small enough to say the state is atleast approximately positive semi-definite, we take our estimate state to be the state that lies a few tens of iterations before where we stopped.

4.2.4 Noisy Measurements

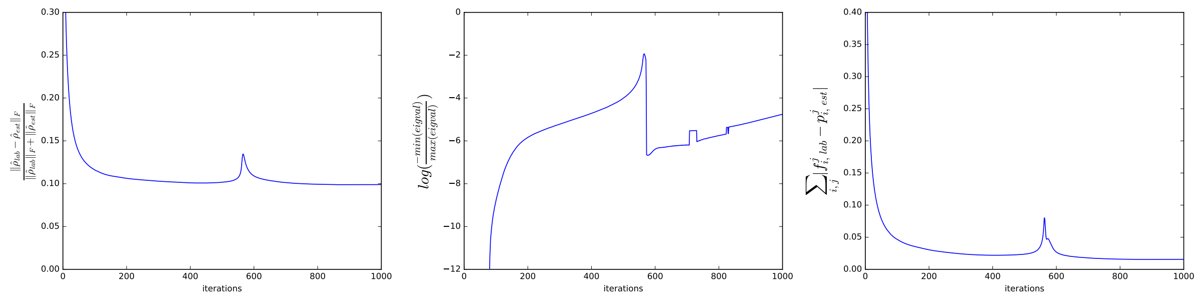
In a real tomography experiment we never have the exact outcome probabilities p^j_i corresponding to the POVM elements $\hat{\Pi}^j_i$. We refer to the measurements in this case as "noisy measurements". The noise in our measurements decreases as we increase the number of measurement experiments we do for each Pauli basis setting. Remember that, for one qubit, each pair of elements in the POVM corresponds to one Pauli setting, i.e. the projector pair $\hat{\Pi}_1^1 = \left|\sigma^x_{\text{up}}\right\rangle \left\langle\sigma^x_{\text{up}}\right|$ and $\hat{\Pi}_1^2 = \left|\sigma^x_{\text{down}}\right\rangle \left\langle\sigma^x_{\text{down}}\right|$ are possible outcomes of the measurement setting σ^x. Then for r qubits, if we measure each Pauli setting m times we will have performed a total of $3^r m(N-r+1)$ measurements. The noise in our measurements vanishes as $m \to \infty$. Thus, for less than infinite measurements, the MLE algorithm will not converge to the state in the lab, but might converge to a good estimate of it.

An important thing to keep in mind is that, for a given set of noisy measurements, the state that maximizes the log-likelihood function will not be the lab state, and such state might be less approximable by an MPO with the bond dimension we are compressing to than if the measurements were noiseless. That is, as we change the number of samples we do per Pauli settings, the states that maximize the log-likelihood function will be different and will require different (possibly larger) bond dimensions to be accurately approximated by the MPO form, and this has the effect of changing the compression error in the MLE iterations. Since the compression error introduces negative eigenvalues into the estimate state (subsection 4.2.3), which cause the MPO-MLE to become unstable, we can conclude that noise contributes to making the MPO-MLE unstable. Figure 4.5 confirms this conclusion and shows what happens as we go from 10 to 100 and 1000 MPO-MLE iterations, allowing the noise-caused negative eigenvalues to grow. Notice from plots (a) and (b) in figure 4.5 that the negative eigenvalues grow with the number of iterations as the compression error accumulates.

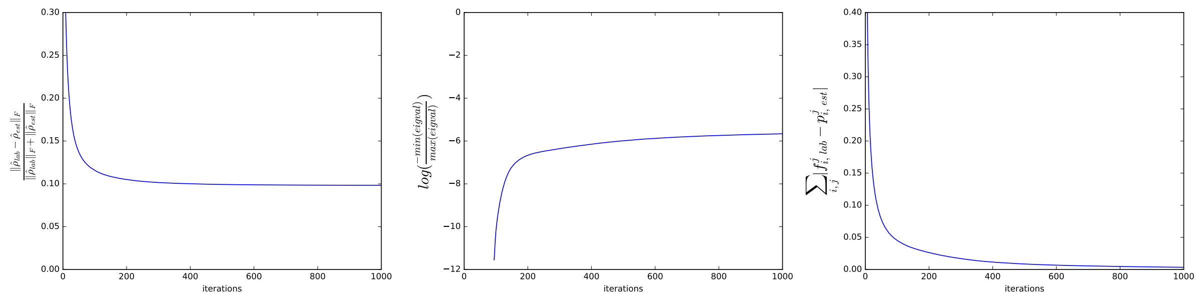
The results in this section show the effect of noise on a small system and the measurement block size used was equal to the size of the system. It is left for future work to explore the effect of noise on large systems, where the measurement block size is much smaller than the size of the system.



(a) Compressing to bond dimension 4

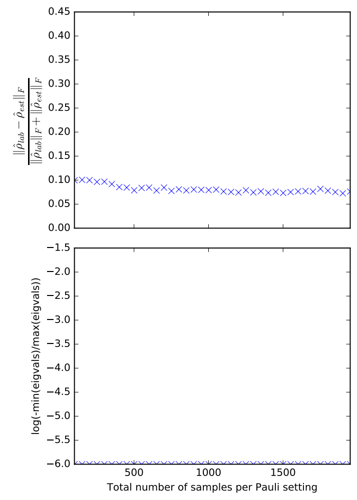


(b) Compressing to bond dimension 6

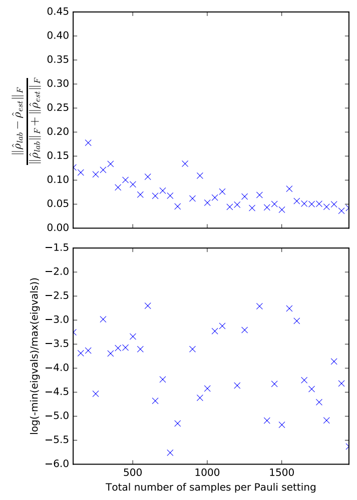


(c) Compressing to bond dimension 8

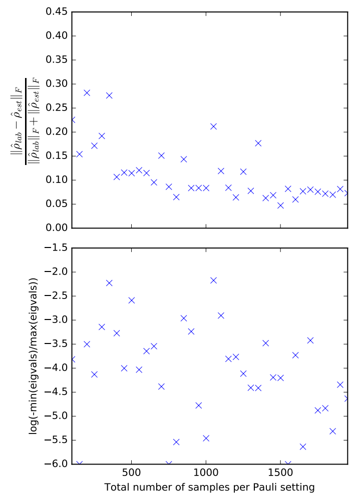
Figure 4.4: By comparing the plots, one can see how compression error introduces negative eigenvalues and causes instabilities in the algorithm. The example in these plots also suggests that the $\sum_{i,j} |f^j_{i,\mathrm{lab}} - p^j_{i,\mathrm{est}}|$ measure behaves like the Frobenius norm distance, and so can be used to detect MLE instabilities in a real tomography experiment, where we do not know the lab state. The system simulated in this figure consists of 6 qubits; The block width is 3; the lab state MPOs were generated by multiplying random PMPSs of bond dimension 2 with their adjoints.



(a) 10 iterations



(b) 100 iterations



(c) 1000 iterations

Figure 4.5: The norm distance curve becomes more erratic as the negative eigenvalues become comparable (by a factor of about 10^2) to the positive eigenvalues. Note that the erratic behavior is less severe as we increase the number of measurements we make per Pauli setting. This shall be confirmed also for large systems in the next chapter. Sites = 4 ; Block Width = 4 ; the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint ; the outcome frequencies for the POVM elements of each Pauli basis setting are simulated by drawing from the exact probabilities according to a multinomial distribution.

Chapter 5

Quantum State Tomography using Purified Matrix Product States

The MPO-MLE algorithm sometimes fails to produce a positive semi-definite estimate of the lab state, mainly due to fact that compression error introduces negative eigenvalues into the estimate state, an effect which is contributed to by the presence of noise in our measurements (see subsections 4.2.3 and 4.2.4). Adding to that problem is the fact that we cannot efficiently recognize when the estimate state (in its MPO form) has gained a negative eigenvalue during the iterations (e.g. [19] and [20]). As a workaround, it was suggested (e.g. [5]) that we can guarantee that the MLE produces a positive semi-definite state if we modify it such that it iterates over purifications of the system rather than the system itself, and after the iterations have finished we trace out the ancillary systems, which leaves us with a positive semi-definite state by construction. Furthermore, in addition to always producing a positive semi-definite state, each iteration in the PMPS-MLE takes less time to be computed than each iteration of the MPO-MLE due to the structure of the algorithm. We refer to this version of the MLE algorithm as PMPS-MLE. As elegant as it seems, PMPS-MLE in general could be less useful than the MPO-MLE due to the fact that the PMPS and MPO forms are inequivalent [20]. That is, although the PMPS-MLE will always produce a positive semi-definite state, it is possible that we might have to choose an arbitrarily large bond dimension for our estimate state, i.e. a large compression bond dimension, to be able to reconstruct the lab state. With that said, states that are physically relevant, e.g. thermal state of local Hamiltonians, are yet to be explicitly shown to have an inefficient PMPS bond dimension. In this chapter, we assume that the states we are dealing with have an efficient PMPS bond dimension.

In this chapter we introduce the PMPS-MLE algorithm (section 5.1), then in section 5.2 we use it to compare the PMPS-MLE and the MPO-MLE in a way that addresses the issues in the MPO-MLE algorithm discussed in section 4.2, especially the measurement noise issue and the negative eigenvalue issue. Finally, we examine the scalability of the PMPS-MLE and the MPO-MLE to large systems (section 5.3).

5.1 The PMPS-MLE Algorithm

Simply put, the steps of doing QST using the PMPS-MLE algorithm are as follows:

1. Do the measurement experiments and collect the outcome frequencies f_i^j corresponding to the POVM elements $\hat{\Pi}_i^j$.

2. Choose a purification $|\psi_{\text{init}}\rangle$ of the completely mixed state as the initial point for the MLE.
3. Do the iteration: $|\psi_{\text{k+1}}\rangle = \mathcal{N}[\hat{R}(|\psi_{\text{k}}\rangle)\,|\psi_{\text{k}}\rangle]$, where $\hat{R}(|\psi_{\text{k}}\rangle) = \sum_{i,j} \frac{f_i^j}{\langle \hat{\Pi}_i^j \rangle_{\psi_{\text{k}}}} \hat{\Pi}_i^j$.
4. Compress the estimate state:
$$|\psi_{k+1}\rangle \to \mathcal{C}[|\psi_{k+1}\rangle].$$
5. Trace out the ancillary sites after the last MLE iteration to get the estimate state $\hat{\rho}_{\text{end}}$.

One difference between the MPO-MLE and the PMPS-MLE is in steps 3 and 4. Where in the MPO-MLE the next state estimate $\hat{\rho}_{\text{k+1}}$ is acquired from the current state estimate $\hat{\rho}_{\text{k}}$ by operating on it by $\hat{R}$ from both sides (equation (4.2)), in the PMPS-MLE, similar to the MPS-MLE, only one operation with $\hat{R}$ is required (equation (4.1)). This causes each iteration of the PMPS-MLE to be faster than each iteration of the MPO-MLE, as shall be demonstrated in section 5.2.1.

Other than the difference in speed due to the different number of matrix multiplications, one would expect the MPO-MLE and the PMPS-MLE to behave in exactly the same way, i.e approach the lab state in the same number of iterations, since the MPO-MLE iteration equation can be written as

$$\hat{\rho}_{k+1} = \hat{R}\hat{\rho}_k\hat{R}$$

$$= \hat{R}\ \text{tr}_{\text{ancilla}}(|\psi_k\rangle\langle\psi_k|)\,\hat{R} = \text{tr}_{\text{ancilla}}(\hat{R}\,|\psi_k\rangle\langle\psi_k|\,\hat{R})$$

$$= \text{tr}_{\text{ancilla}}(|\psi_{k+1}\rangle\langle\psi_{k+1}|). \tag{5.1}$$

That, however, is not the case, due to reasons that will be explained in the next section.

An interesting thing to note is that removing the last step of the PMPS-MLE algorithm, namely tracing out the ancillary sites, practically gives us a purification algorithm in the sense that we give to the algorithm outcome frequencies as input and get the purification of that state as output. Since compression must be done after each iteration to keep the bond dimension under control, the output state will not be an exact purification of the lab state, but might be a good approximation of it. It would be interesting to compare the purification that we get out of the PMPS-MLE with what we get from the purification methods presented in [20], which produce an exact purification.

5.2 MPO-MLE vs PMPS-MLE

In this section we compare the performance of the MPO-MLE and the PMPS-MLE in terms of convergence speed and reconstruction quality.

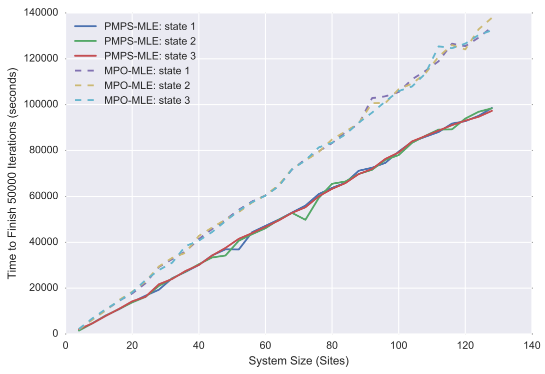


Figure 5.1: Comparison between the time the two MLE algorithms need to complete 50000 iterations: it is clear from the plot that each PMPS-MLE iteration is faster than each MPO-MLE iteration. In this plot, the double compression scheme was used for the MPO-MLE (see section 4.1.2). The single compression scheme used in [6] would have made the MPO-MLE to be even slower than in this plot. Block Width = 4 ; the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint.

5.2.1 Convergence Speed and Reconstruction Quality

Because the PMPS is a pure state, the PMPS-iteration equation is identical to the MPS-MLE iterations equation, i.e.

$$|\psi_{k+1}\rangle = \mathcal{N}[\hat{R}(|\psi_k\rangle)\,|\psi_k\rangle].$$

The fact that each PMPS-MLE iteration involves only one MPO-PMPS multiplication, in contrast to the two MPO-MPO multiplication done in each MPO-MLE iteration, suggests that a computer running PMPS-MLE will be faster in computing a certain number of iterations than it would be if it were running the MPO-MLE, as can be seen in figure 5.1.

On the other hand, from the plots in figure 5.2 we conclude the following: although the PMPS-MLE is faster than the MPO-MLE in terms of the time taken to finish a certain number of iterations, the MPO-MLE is usually faster at converging to the terminal state than the PMPS-MLE. That is, the MPO-MLE needs less iterations to reach its terminal state than the PMPS-MLE. There are two possible reasons behind this. The first is that compression has a different effect on the state estimate in the two MLE algorithms. If we do not do any compression at all, the two algorithms behave in exactly the same way, as equation (5.1) shows. If we add compression to the scene, however, we see that the two algorithms are not the same. Taking compression into consideration, the MPO-MLE becomes

$$\hat{\rho}_{k+1} = \mathcal{C}\Big[\mathcal{C}\big[\hat{R}\hat{\rho}_k\big]\hat{R}\Big]$$

$$= \mathcal{C}\Big[\mathcal{C}\big[\hat{R}\ \mathrm{tr}_{\mathrm{ancilla}}(|\psi_k\rangle\langle\psi_k|)\big]\hat{R}\Big] = \mathcal{C}\Big[\mathcal{C}\big[\mathrm{tr}_{\mathrm{ancilla}}(\hat{R}\,|\psi_k\rangle\langle\psi_k|)\big]\hat{R}\Big],$$

and the PMPS-MLE becomes

$$\hat{\rho}_{k+1} = \mathrm{tr}_{\mathrm{ancilla}}(|\psi_{k+1}\rangle\langle\psi_{k+1}|)$$
$$= \mathrm{tr}_{\mathrm{ancilla}}(\mathcal{C}\big[\hat{R}\,|\psi_k\rangle\big]\mathcal{C}\big[\langle\psi_k|\,\hat{R}\big]).$$

Comparing the above equations with equation (5.1), we can see that compression alone makes both algorithms different. Compression does not merely reduce the bond dimension of a state, but it can also bring the estimate state closer to the lab state by removing small singular values that are contributing to the Frobenius relative norm distance between the two. The second possible reason that the two algorithms behave differently is that, if our lab state has a PMPS representation with bond dimension D_{PMPS}, then, unlike the PMPS-MLE, the MPO-MLE iterations are allowed to land on states that have a PMPS bond dimension $D'_{\text{PMPS}} > D_{\text{PMPS}}$, which might provide a "shortcut" to the lab state. This is due to the fact that the MPO and PMPS representations are not equivalent [20].

In summary, both MPO-MLE and PMPS-MLE have their advantages and disadvantages. From the simulations we have run (figure 5.2), we have seen that the MPO-MLE converges faster than the PMPS-MLE in terms of iterations, although it might happen that it becomes unstable very early in the iterations, in which case the PMPS-MLE would be more useful. Indeed, in the next subsection we will see that when we introduce noise into the measurements, the instability of the MPO-MLE becomes severe enough for the performance of the PMPS-MLE to come close to that of the MPO-MLE. Moreover, while the PMPS-MLE can need more iterations to converge to a certain error as the measurement noise decreases, it always produces a positive semi-definite estimate.

5.2.2 Noisy Measurements Revisited

As was demonstrated in section 4.2.4, noisy measurements participate in introducing negative eigenvalues into the estimate states of the MPO-MLE because different sets of noisy measurements correspond to different states (with different bond dimensions) that maximize the log-likelihood function, leading to the possibility of large compression errors, and consequently large negative eigenvalues in the estimate states. But since the PMPS-MLE produces states that are, by construction, positive semi-definite, noisy measurements can never cause negative eigenvalues to appear in the state estimate. The worst that noisy measurements can do is cause the PMPS-MLE to produce a state estimate that does not overlap with high accuracy with the lab state we are trying to reconstruct. Figure 5.3 shows the same plot from section 4.2.4, with the PMPS-MLE also plotted for comparison. It can be clearly seen from the figure that PMPS-MLE does not become erratic as the MPO-MLE when the measurements are noisy. Since in a real QST experiment the measurements will always be noisy (we can never make infinite measurements), figure 5.3 tells us that the PMPS-MLE might perform, on average, better than the MPO-MLE as the noise in the measurements increases.

This conclusion is confirmed for many system sizes by figure 5.5, where the relation between the number of samples, reconstruction error, and number of sites is plotted. From the figure we can see that the noisier the measurements are, the larger the reconstruction error will be for both the MPO-MLE and the PMPS-MLE algorithm. But more importantly, we can clearly see how as the measurement noise increases the speed advantage that the MPO-MLE had over the PMPS-MLE seems to vanish. Figure 5.4 gives us a better view of how the instability of the MPO-MLE increases as the we go from infinite samples per Pauli setting to 1000 samples per Pauli setting.

5.3 Examining Scalability

With the merging of the Matrix Product representation and the MLE algorithm in [6], the problem of the exponential growth of the Hilbert space with the number of constituents of the system, often called the "curse of dimensionality", seems to be solved for a wide

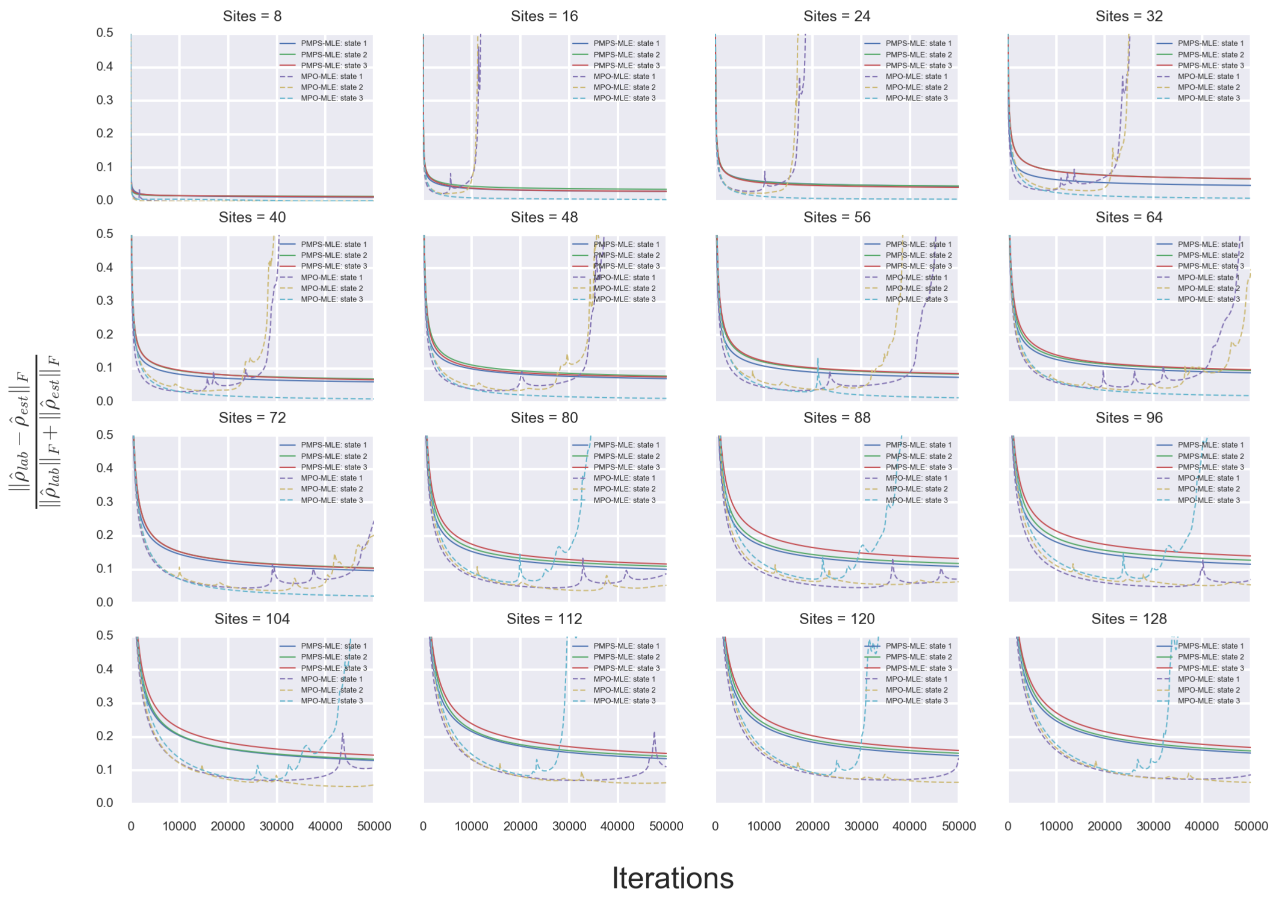
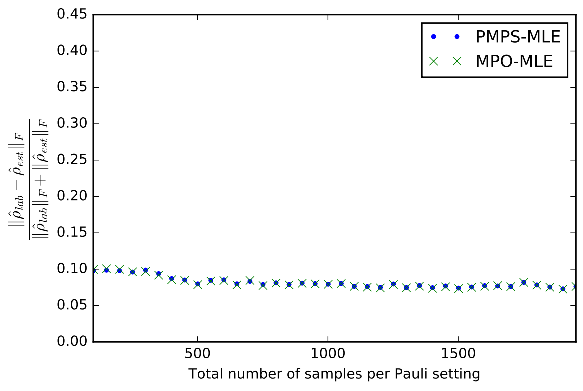


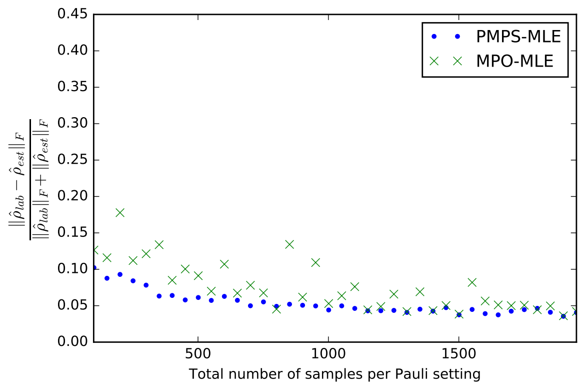
Figure 5.2: Two things can be noticed from these plots: the first is that the MPO-MLE needs less iterations to get within a certain distance from the lab state; the second is that, unlike the MPO-MLE, the PMPS-MLE does not exhibit the kind of instabilities that the MPO-MLE exhibits because the state estimate produced by the PMPS-MLE is, by construction, positive semi-definite. Note that it appears to be completely random where the negative eigenvalues start appearing. Block Width = 4 ; the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint.

spectrum of states that are physically relevant in quantum information science (see section 3.6). The number of sites in the system can increase, but the number of parameters needed to represent the state in its MPS (MPO) form remains manageable. Moreover, the strategy of doing the measurement experiments on blocks of r consecutive sites instead of on the whole system reduces the scaling of the number measurement experiments needed from exponential with the number of sites N in the system to a linear scaling. While this suggests that the MPO-MLE and the PMPS-MLE are scalable to large system sizes, it is not yet clear how scalable they truly are; it might be that a larger system needs significantly more MLE iterations than a smaller system to converge. In this section we examine the scalability of the PMPS-MLE and the MPO-MLE algorithms.

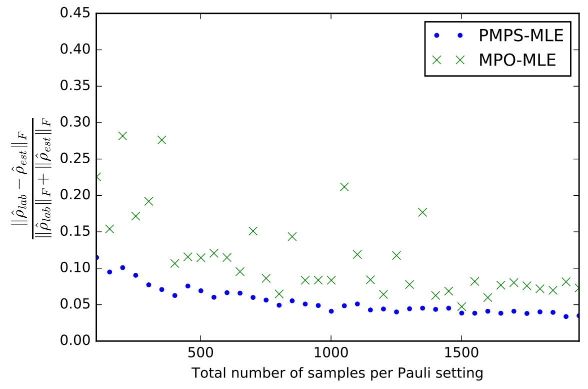
From a practical point of view, the PMPS-MLE or the MPO-MLE are successful if we can use them to estimate the lab state to within a certain threshold error in a reasonable number of iterations. From figure 5.6 we can already see that larger systems will require more iterations to reach a certain relative norm distance. It is also clear that, for a



(a) 10 iterations



(b) 100 iterations



(c) 1000 iterations

Figure 5.3: As the number of iterations increase, the negative eigenvalues forming in the MPO-MLE estimate states cause the estimate states to move away from the lab state. In comparison, the PMPS-MLE is unaffected. Sites = 4 ; Block Width = 4 ; the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint ; the outcome frequencies for the POVM elements of each Pauli basis setting are simulated by drawing from the exact probabilities according to a multinomial distribution.

fixed target relative norm distance, the number of iterations does not grow linearly with the systems size. Looking at figure 5.6, one can notice that if the required threshold error is lowered too much, the number of iterations needed to achieve that error for a large system becomes impractically large, so in this section we base our conclusions on the requirement that the number of iterations does not exceed 50000 iterations (50000 iterations take about 36 hours to simulate using the computational setup we are using; see chapter 8). From this requirement it immediately follows that, beyond a certain system size, MPO-MLE and PMPS-MLE will simply fail to reconstruct the lab state to within any reasonably low target error that we set. It is worth mentioning that increasing the measurement block size decreases the number of iterations needed to achieve a certain threshold relative norm distance. But, since the number of measurements experiments one has to make increases exponentially with the block size, the maximum block size that can be used will not be significantly larger than the block size we are using in this section, especially as the size of the considered physical system increases.

In figure 5.7 (a) we plot the number of iterations needed to reach a certain target relative norm distance against the size of the system for three different states. Plotting

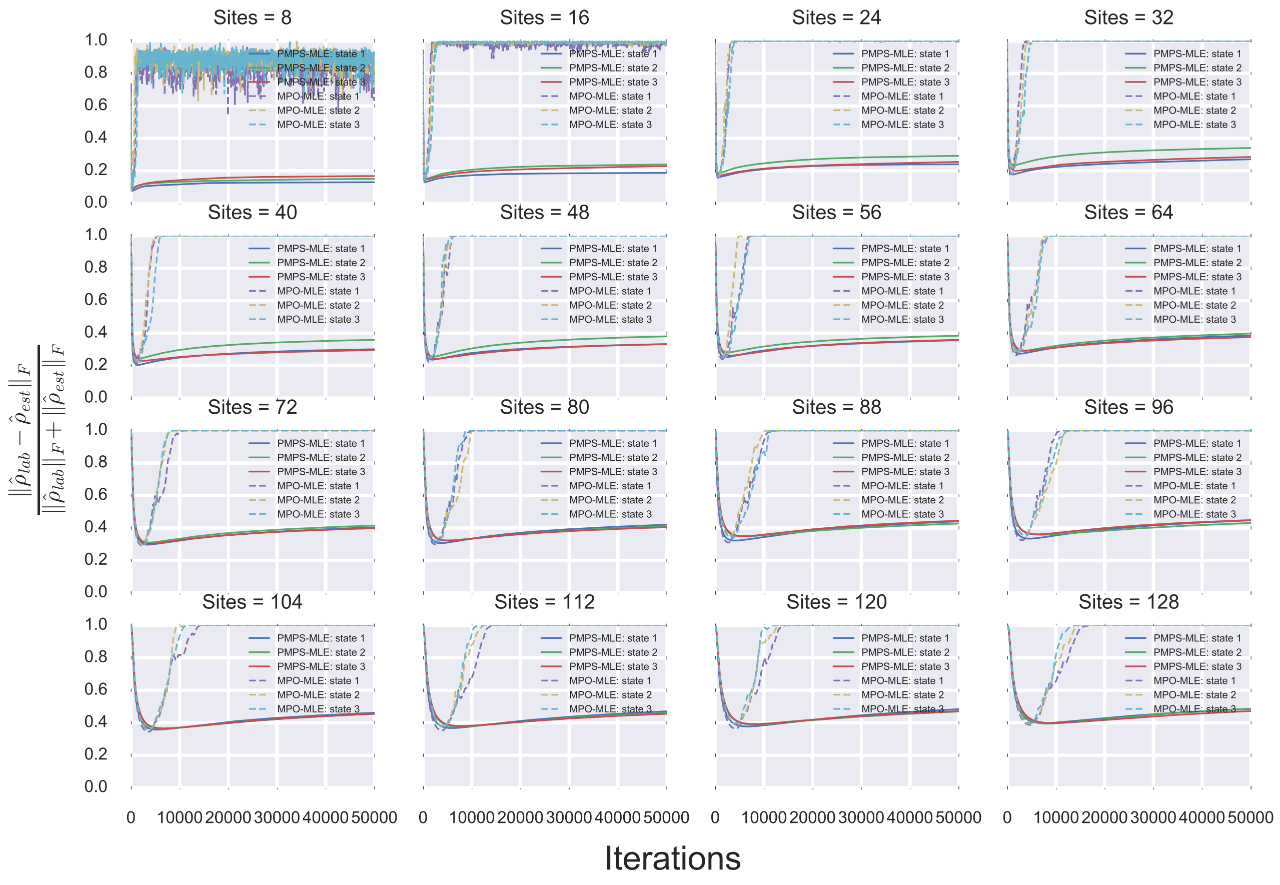
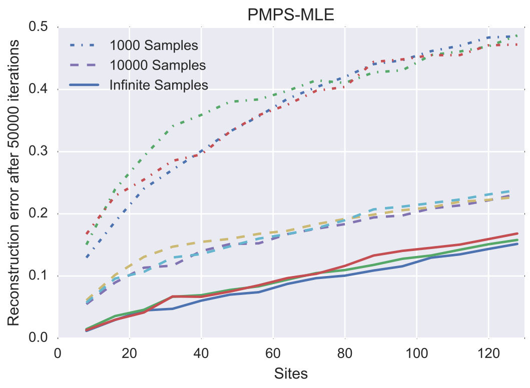


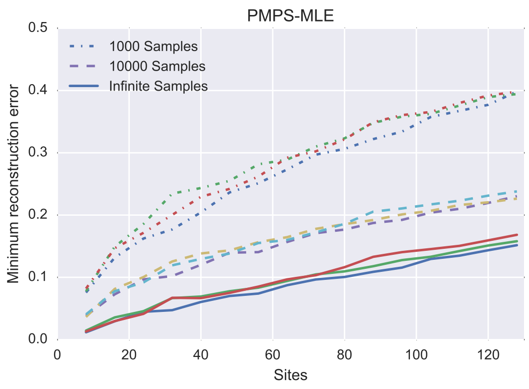
Figure 5.4: This plot is the same as plot 5.2 but it is done for 1000 samples per Pauli setting. One thing to notice from this plot is that, compared to the plot in figure 5.2, the MPO-MLE becomes unstable earlier in the iterations, and the PMPS-MLE exhibits a rise in the Frobenius distance.

the same plot on a log-log scale (plot (b)), we can see that, as far as we can see (in terms of iterations), it is approximately a straight line, suggesting that the scaling of the number of iterations with the target error is polynomial. We can then vary the target error and observe the change of the slope of the line in figure 5.7 (b), and thus the exponent of the polynomial scaling. As expected, we find that the exponent depends on the required target error, i.e $I \sim N^{f(\text{error})}$, where N is the number of qubits in the system and I the number of iterations needed to reach the target relative norm distance (5.8). From figure 5.8 we can see that the smaller we set our target error the greater the exponent will be.

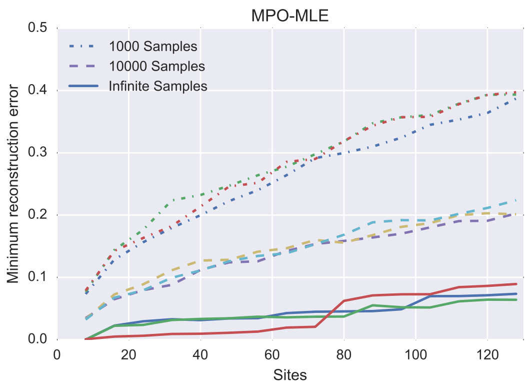
Note that plots 5.6, 5.7, and 5.8 are for the PMPS-MLE. The same plots can be done for the MPO-MLE by considering the portion of the norm distance curves where the MPO-MLE is still stable, but these plots will not be reliable in making any conclusions about the scaling of the iterations with the system sizes due to the fact that the instabilities seems to start randomly in the iterations. However, from figure 5.9 we can see that roughly the scaling should not be much different, since from the stable portion of the figure we can see that the number of iterations needed to reach a certain error scale polynomially with the system size.



(a)



(b)



(c)

Figure 5.5: Due to the compression error, which is contributed to by the noise in the measurements, the MPO-MLE diverges after a certain number of iterations. Thus, to plot the MPO-MLE curves (plot (c)), we must use the minimum error reached in all the iterations. At 1000 samples per Pauli setting, the PMPS also exhibits a rise (not divergence) in the Frobenius distance (see figure 5.4), and thus to calculate plot (b) we use the minimum of the curve. Comparing between plots (a) and (b), one can see that at 10000 and infinite samples the curves in the two plots are approximately the same. The minimum could be found in a real tomography experiment (where we do not know the lab state) using the distance between outcome frequencies of the estimate state and the probabilities of the lab state (equation (4.4)); the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint ; the outcome frequencies for the POVM elements of each Pauli basis setting are simulated by drawing from the exact probabilities according to a multinomial distribution.

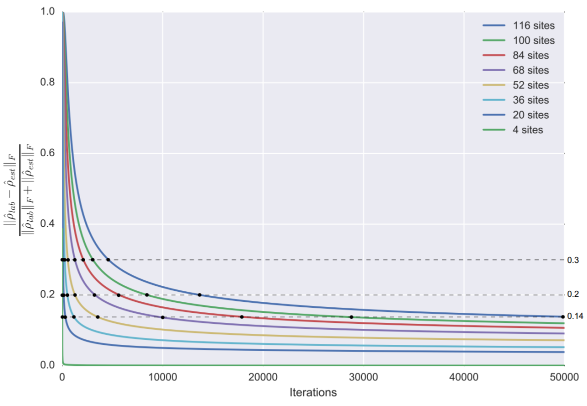
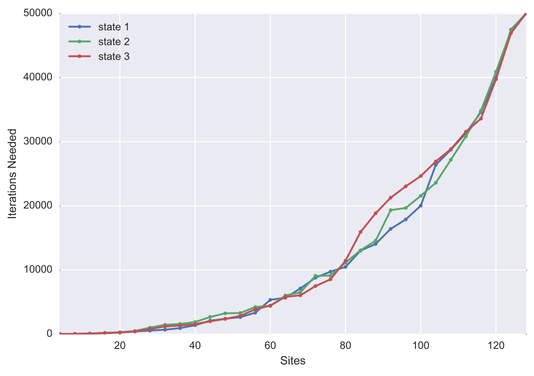
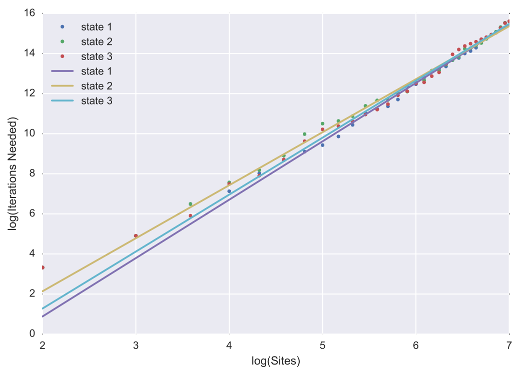


Figure 5.6: PMPS-MLE: For the range of system sizes that we have checked, more iterations are needed to reach a certain target error as the system size increases. Block Width = 4 ; the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint.

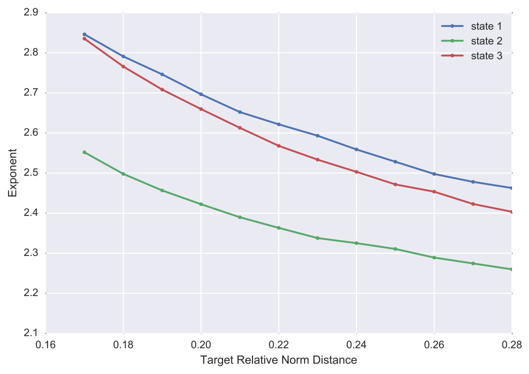


(a)



(b)

Figure 5.7: PMPS-MLE: Each data point in plot (a) is acquired for the three states when $\frac{\|\hat{\rho}_{\text{lab}} - \hat{\rho}_{\text{est}}\|_F}{\|\hat{\rho}_{\text{lab}}\|_F + \|\hat{\rho}_{\text{est}}\|_F}$ is less than 0.15 for state 1, 0.16 for state 2, and 0.17 for state 3 (these are the minimum values reached for each of the states after 50000 iterations). From the linear fits in plot (b), we acquire the exponent of the polynomial in plot (a). The exponent for these particular target error is 3. Block Width = 4 ; the lab state MPO was generated by multiplying a random PMPS of bond dimensions 2 with its adjoint.



(a) Exponent vs Target Error

Figure 5.8: The exponent is calculated by calculating the slopes of the lines in plot (a) for a range of target errors. As the target error is made smaller the exponent of the curve increases.

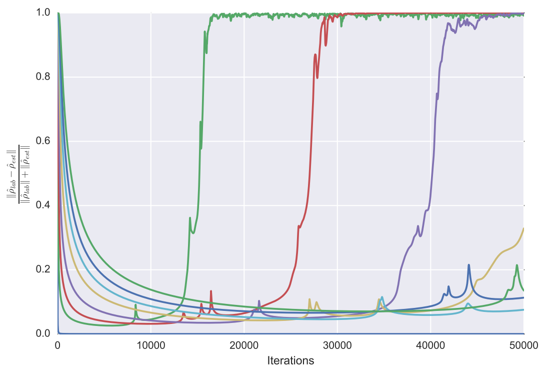


Figure 5.9: MPO-MLE: relative norm distance is plotted against iterations. The different curves correspond to different system sizes, with the topmost curve belonging to the largest system (116 qubits) and the bottommost curve belonging to the smallest system (4 qubits).

Chapter 6

Speeding Up the MLE Algorithm

In the convergence proof in subsection 2.3.1, a step size ϵ is guaranteed to exist such that the likelihood function is increased in each iteration until the maximum is reached. The increase in the likelihood function can be small or large for different values of ϵ. Finding the optimum step size ϵ_{opt} for which the increase in the likelihood function is maximum, a procedure referred to as an "exact line search" procedure, is an expensive optimization problem if we plan to do it in each iteration [10]. In [10], the authors demonstrate an "inexact line search" MLE algorithm where the criteria for choosing a step size ϵ is not that it is the optimum one, but that it produces a certain minimum increase in the likelihood function. In [27], an improvement over the algorithm in [10] is demonstrated where the value of ϵ is more optimally chosen using an interpolation scheme whose support values are many different values of ϵ that increase the likelihood function. Both algorithms can be considered Gradient Ascent algorithms. In this chapter we discuss a different approach to increase the speed of convergence of the MLE algorithm, and we attempt to apply it to the MPO-MLE and PMPS-MLE algorithms as well.

6.1 Speeding Up the Regular MLE

Although, like the algorithms in [10] and [27], our algorithm has the same form as equation (2.6), the motivation behind the choice of the step size is different. Inspired by the monotonic increase of the log-likelihood of the MLE in each iteration, one can think of speeding up the convergence by somehow making an educated guess about what the estimate state will be after a few iterations instead of actually doing the iterations; i.e. extrapolating a future estimate state from the previous estimate states. To understand the behavior of the estimate state matrix during the application of the MLE, it is instructive to monitor the entries of the matrix in each iteration. Figure 6.1 shows us that each entry of the estimate state matrix $\hat{\rho}_{\text{est}}$ converges to the corresponding entry in the lab state matrix $\hat{\rho}_{\text{lab}}$ in an approximately monotonic fashion. By "approximately monotonic" we mean that the convergence is monotonic over many iterations, but is not globally monotonic. Since the number iterations over which the converge is monotonic is usually large, this approximate monotonicity can be used to speed up the convergence of the MLE.

Thus, to make a guess about what $\hat{\rho}_{\text{est}}$ will be after a few iterations, we can make a guess for each of its entries. This can be simply done using linear extrapolation, i.e.

$$\rho_{k+1}^{00} = \rho_k^{00} + t(\rho_k^{00} - \rho_{k-p}^{00}),$$

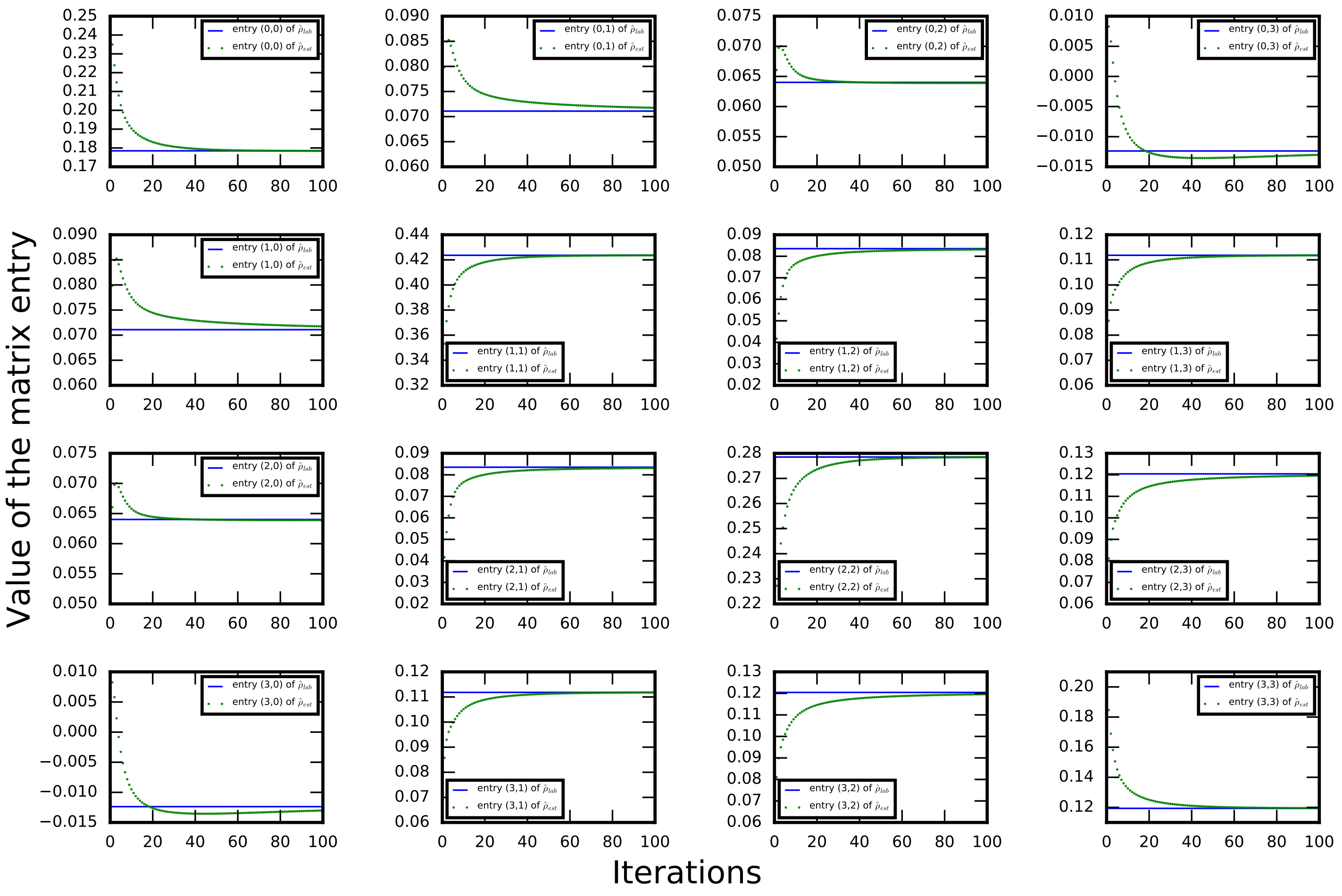


Figure 6.1: The plot shows the convergence behavior of the entries of the estimate state with each iteration for the MLE algorithm on 2 qubits. Note that while for some of the entries the convergence is not globally monotonic, it is monotonic over large numbers of iterations.

where ρ^{00}_{k-p} is the top-leftmost entry of the estimate state matrix before p iterations, and t is the step size. If we do this for every entry of $\hat{\rho}_{\text{est}}$, it might be possible to accelerate the convergence of the MLE algorithm. Doing the extrapolation for each entry individually is equivalent to doing it on the whole matrix once, i.e.

$$\hat{\rho}_{k+1} = \hat{\rho}_k + t(\hat{\rho}_k - \hat{\rho}_{k-p}). \tag{6.1}$$

Notice that, without additional constraints, equation (6.1) is not guaranteed to produce a positive semi-definite state due to the fact that the difference of two positive semi-definite matrices (the second term on the right hand side of equation (6.1)) is not necessarily positive semi-definite. A difference of positive semi-definite matrices also exists in equation (2.6), but it has been shown in [10] that the difference is positive semi-definite.

Besides giving us a negative matrix, if $\hat{\rho}_k - \hat{\rho}_{k-p}$ happens to be negative – which we have observed to be very likely – this can lead to instabilities in the MLE iterations and can cause the iterations to not converge (see subsection 4.2.3). Heuristically, we have observed that as long as the negative eigenvalues spawned in equation (6.1) are small, the MLE iterations that are done after the guess can compensate for them and the instability does not occur. The algorithm we use is as follows. First we do a certain number of MLE iterations like in the regular algorithm. Then we use the current state estimate, $\hat{\rho}_{\text{k}}$, and

Algorithm 1: Speeding Up MLE

```
Data: The inputs to the algorithm are $p$, $f$, and basic iterations, which is the
      number of MLE iterations we do before we make our guess.
1  initialization;
2  step = $f/p$ ;
3  for itr < Iterations: itr++ do
4      Do MLE iterations normally;
5      if count == basic iterations then
6          $\hat{\rho}_{\text{k-p}} = \hat{\rho}_{\text{k}}$;
7      end
8      if count == basic iterations + p then
9          $\hat{\rho}_{\text{k}} = \hat{\rho}_{\text{k}} + \text{step} * (\hat{\rho}_{\text{k}} - \hat{\rho}_{\text{k-p}})$;
10         Normalize $\hat{\rho}_{\text{k}}$ ;
11         count = 0;
12     end
13     count++;
14 end
```

the state estimate before p iterations, $\hat{\rho}_{\text{k-p}}$, to extrapolate the next state estimate after f iterations, $\hat{\rho}_{\text{k+f}}$. We then again run the MLE iterations as usual. See algorithm 1. The purpose of the MLE iterations we run before $\hat{\rho}_{\text{k-p}}$ is to attempt to compensate for any negative eigenvalues resulting from the difference term in equation (6.1). Ideally, these iterations should be few. However, if the negative eigenvalues cause the algorithm to diverge, we increase them until the algorithm is stable again.

Figures 6.2 and 6.3 demonstrate the improvement of the new algorithm over the regular one for the MLE on a 2 qubit system. It is important to point out that as the estimate state approaches the lab state, the slope of the curve becomes smaller, which causes the extrapolation to not be effective at large iterations. One method to circumvent this obstacle is to increase the parameter f in algorithm 1 (use a farther guess) as the iterations increase. This indeed becomes necessary as we apply the algorithm to large system sizes, since the slope of the curve decreases relatively early before the estimate state approaches the lab state.

6.2 Speeding Up MPO-MLE

The simple form of equation (6.1) prompts trying the new algorithm with the MPO-MLE and the PMPS-MLE algorithms. The same intuition we used to justify equation (6.1), namely the approximate monotonic convergence of the individual entries of $\hat{\rho}_{\text{est}}$, can not be used in the case of MPO-MLE and PMPS-MLE due to the action of compression in each iteration. Specifically, compressing the MPO or the PMPS forms of the state changes the entries of the state matrix, and so the convergence of the entries will not be as smooth as it is in the regular MLE algorithm (figure 6.1). Remarkably, however, the improved algorithm seems to be as usable in the MPO-MLE and PMPS-MLE as it is in the regular MLE. This is especially interesting since it will speed up the convergence of the MPO-MLE and PMPS-MLE, allowing us to apply them for larger systems, which, in a sense, increases their scalability. Figure 6.4 demonstrates that despite the action of

Algorithm 2: Speeding Up MLE: adaptive extrapolation parameters selection.

```
Data: The inputs to the algorithm are $p$, $f$, and basic iterations, which is the
      number of MLE iterations we do before we make our guess.
initialization;
step = $f/p$ ;
for itr < Iterations: itr++ do
    Do MLE iterations normally;
    if count == basic iterations then
        $\hat{\rho}_{\text{k-p}} = \hat{\rho}_{\text{k}}$;
    end
    if count == basic iterations + p then
        $\hat{\rho}_{\text{k}} = \hat{\rho}_{\text{k}} + \text{step} * (\hat{\rho}_{\text{k}} - \hat{\rho}_{\text{k-p}})$;
        Normalize $\hat{\rho}_{\text{k}}$ ;
        $p = \text{ceil}(itr/15)$;
        basic iterations $= \max(5, \text{ceil}(itr/5))$;
        $f = \max(25, \text{ceil}(itr))$;
        step = $f/p$ ;
        count = 0;
    end
    count++;
end
```

compression, the extrapolation scheme works well with the MPO-MLE algorithm.

A further improvement of the algorithm can be achieved by adjusting it such that the step size is increased as the iterations advance. This is done by increasing f, as shown in Algorithm 2. It is important to point out that we have observed that increasing f alone without increasing *basic iterations* might cause the algorithm to destabilize. The resulting improvement of the adaptive algorithm for a system of 8 qubits is shown in figure 6.5.

6.3 Speeding Up PMPS-MLE

Since each PMPS-MLE iteration updates the state of the purification of the system rather than the state system itself, the intuition behind equation (6.1) will be different in that it only assumes that the entries of the pure state of the purified system will converge monotonically. For the PMPS-MLE, equation (6.1) becomes

$$|\psi_{k+1}\rangle = |\psi_k\rangle + t(|\psi_k\rangle - |\psi_{k-p}\rangle). \tag{6.2}$$

Figure 6.6 shows the performance of the improved PMPS-MLE and the improved PMPS-MLE with an adaptive extrapolation parameter choice. Note that the improvement in the PMPS-MLE is not as significant as the improvement in the MPO-MLE. In particular, notice how it seems that after a certain number of iterations the regular PMPS-MLE catches up with the improved PMPS-MLE. After trying to vary the extrapolation parameters in algorithms 1 and 2 with no enhanced performance, we suspect that the relatively poor performance of the improved PMPS-MLE compared to the MPO-MLE is related to the fact that, in the PMPS-MLE, the estimate mixed state at each iteration is a quadratic function of the pure state $|\psi_{k+1}\rangle$ that was extrapolated. That is,

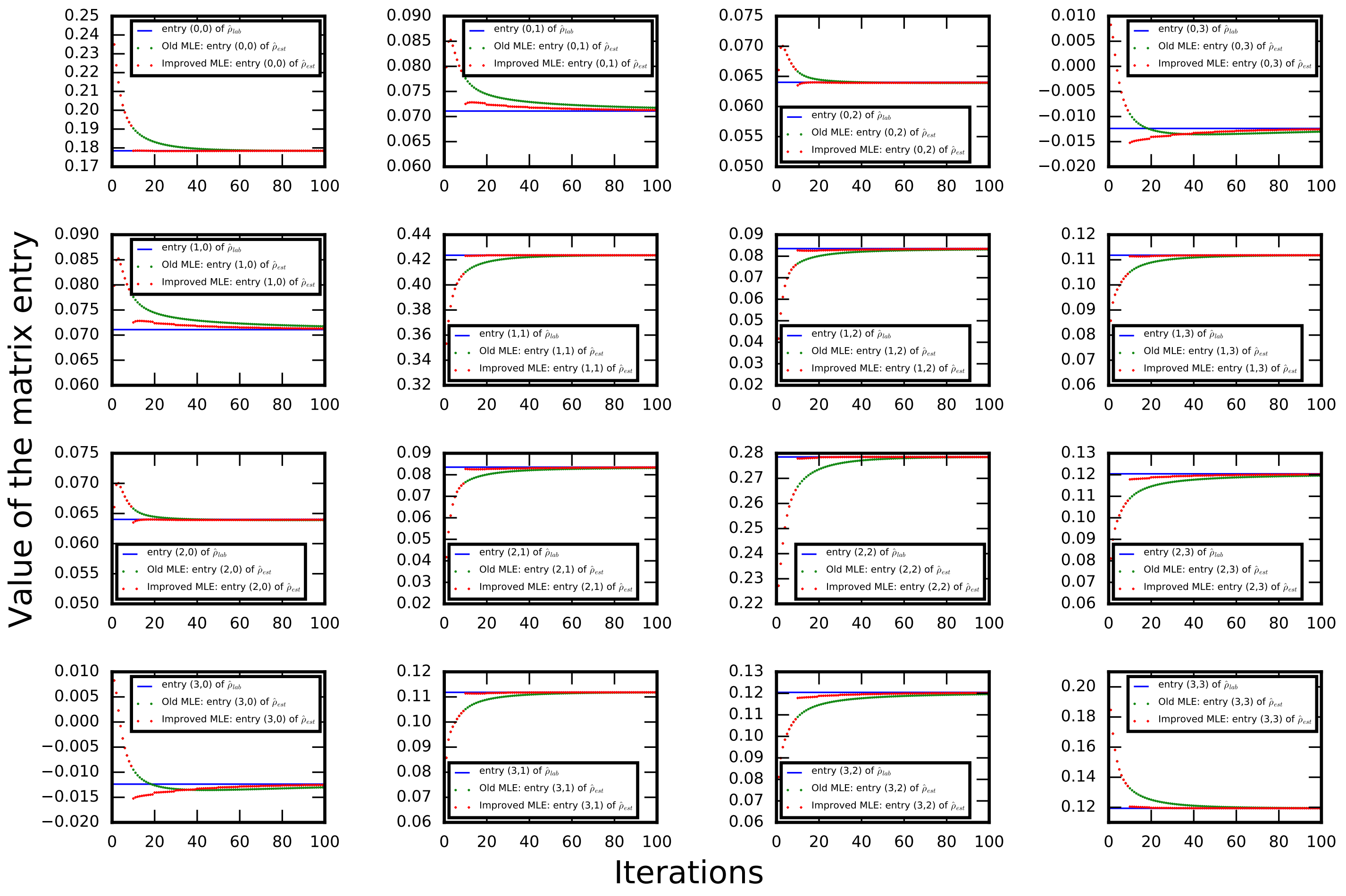
$$\hat{\rho}_{k+1} = \mathrm{tr}_{\text{ancilla}} |\psi_{k+1}\rangle \langle\psi_{k+1}|$$

$$= \mathrm{tr}_{\text{ancilla}} \Big(|\psi_k\rangle + t(|\psi_k\rangle - |\psi_{k-p}\rangle)\Big)\Big(\langle\psi_k| + t(\langle\psi_k| - \langle\psi_{k-p}|)\Big)$$

$$\begin{gathered} = (1+2t+t^2)\,\mathrm{tr}_{\text{ancilla}} |\psi_k\rangle \langle\psi_k| - (t+t^2)\,\mathrm{tr}_{\text{ancilla}} |\psi_k\rangle \langle\psi_{k-p}| \\ -(t+t^2)\,\mathrm{tr}_{\text{ancilla}} |\psi_{k-p}\rangle \langle\psi_k| + t^2\,\mathrm{tr}_{\text{ancilla}} |\psi_{k-p}\rangle \langle\psi_{k-p}| \end{gathered}$$

$$= (1+2t+t^2)\hat{\rho}_k + t^2\hat{\rho}_{k-p} - (t+t^2)\,\mathrm{tr}_{\text{ancilla}}(|\psi_k\rangle \langle\psi_{k-p}| + |\psi_{k-p}\rangle \langle\psi_k|).$$

Note that, in the substitution $\mathrm{tr}_{\text{ancilla}} |\psi_k\rangle \langle\psi_k| = \hat{\rho}_k$, $\hat{\rho}_k$ is only the same as that produced by the MPO-MLE under the assumption that we are not doing any compression.



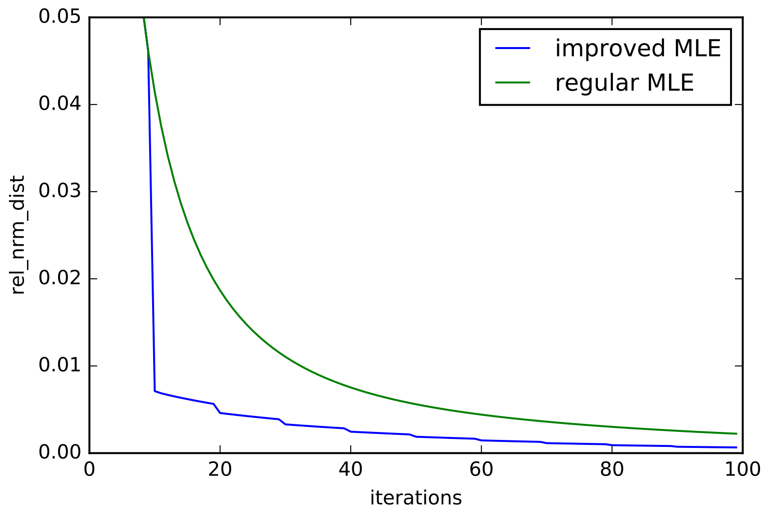
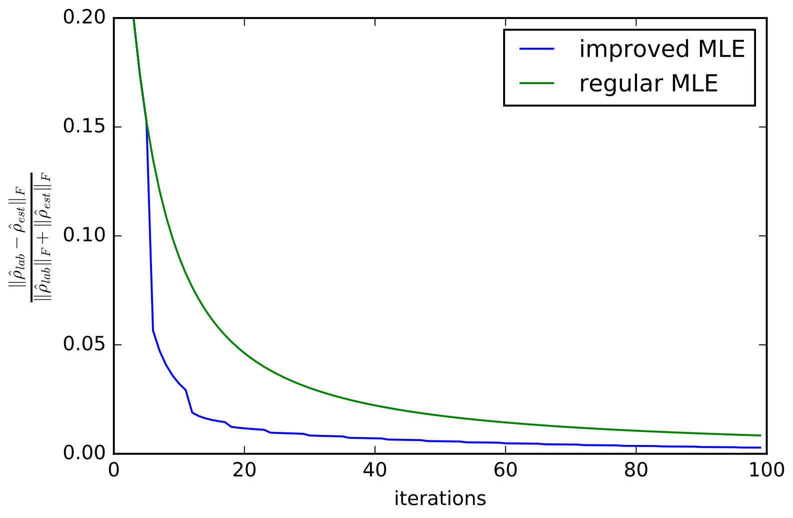
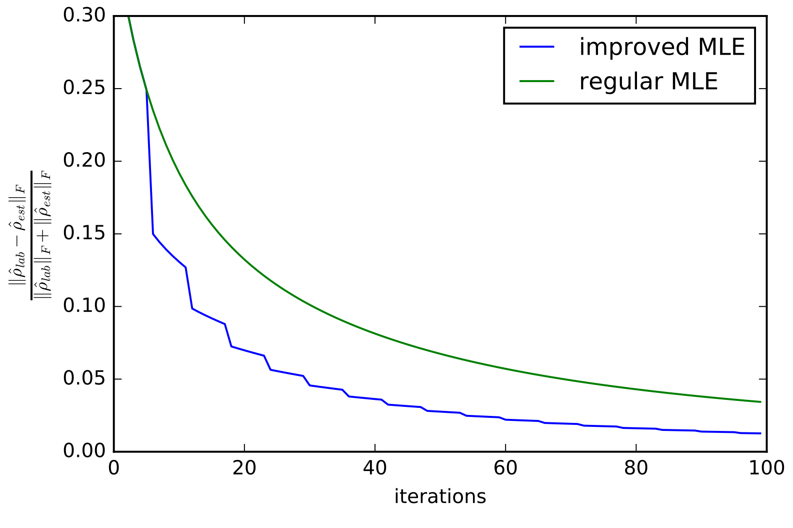


Figure 6.2: Note that although the convergence of some of the entries in the regular algorithm is not globally monotonic, the extrapolation scheme still produces a significant speedup over the regular algorithm. The extrapolation scheme in this plot uses $f = 15$, $p = 2$, *basic iterations* $= 8$. The MLE in this plot is on 2 sites.

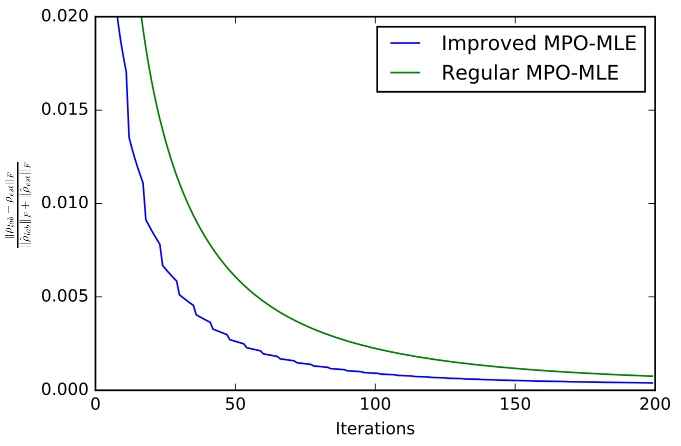


(a)

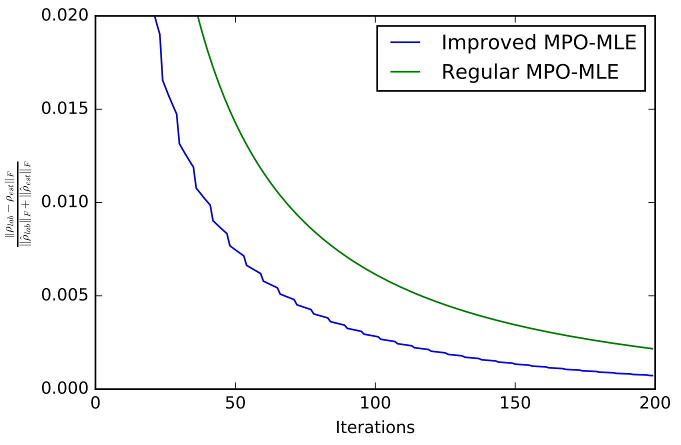


(b)

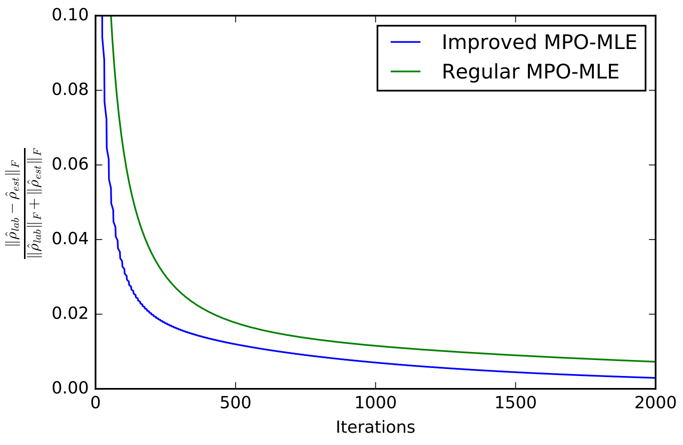
Figure 6.3: The plot shows a comparison between the regular and improved MLE for a system of 3 sites (plot (a)) and 4 sites (plot (b)). The extrapolation scheme in this plot uses $f = 15$, $p = 2$, *basic iterations* $= 4$. The values of these parameters are chosen such that extrapolation does not destabilize the convergence (e.g. by overshooting the true values too much).



(a)

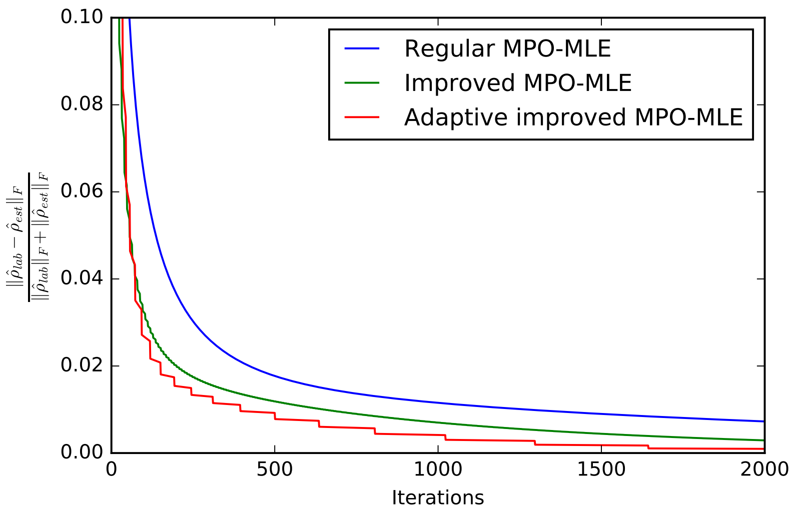


(b)



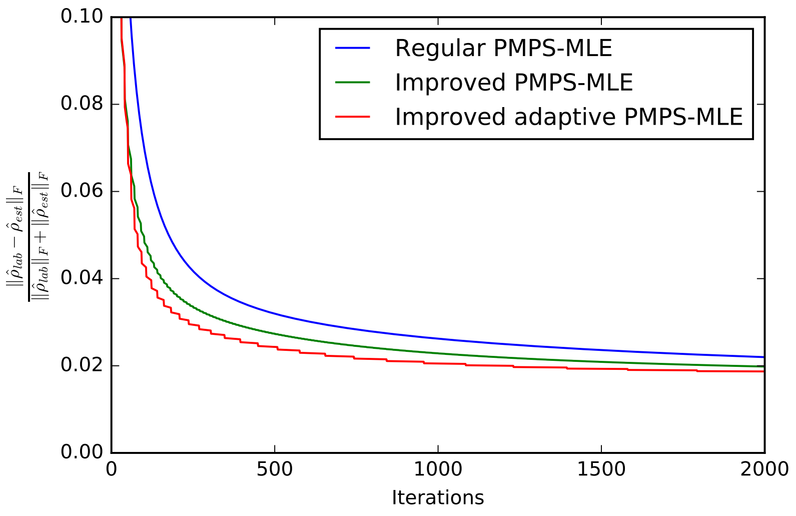
(c)

Figure 6.4: The plot shows a comparison between the regular and improved MPO-MLE for a system of 3 sites (plot (a)), 4 sites (plot (b)), and 8 sites (plot(c)). The first two plots use perfect information, while the third plot uses a measurement block width of 4. The extrapolation scheme in the first two plots uses $f = 8$, $p = 2$, basic iterations $= 4$, while in the third plot the parameters are $f = 24$, $p = 2$, basic iterations $= 6$. The values of these parameters are chosen such that extrapolation does not destabilize the convergence (e.g. by overshooting the true values too much).



(a)

Figure 6.5: The plot shows a comparison between the regular, improved MPO-MLE, and improved MPO-MLE with adaptive extrapolation parameters selection for a system of 8 sites. The measurement block width is 4. The extrapolation scheme in this plot start with the extrapolation parameters $f = 25$, $p = 2$, *basic iterations* $= 6$. Then, the parameters are updated as in Algorithm 2.



(a)

Figure 6.6: The plot shows a comparison between the regular, improved PMPS-MLE, and improved PMPS-MLE with adaptive extrapolation parameters selection for a system of 8 sites. The measurement block width is 4.

Chapter 7

Conclusion

In [6], the authors merged the MLE algorithm with the MPO form of the state, leading to a quantum state tomography (QST) scheme that is reasonably scalable to large system sizes for many families of states. The matrix product operator maximum likelihood estimation algorithm (MPO-MLE), however, often fails to produce a positive semi-definite state due to the introduction of negative eigenvalues in the compression step of the algorithm (section 4.2). The compression error responsible for the negative eigenvalues can be amplified by a multitude of factors, most prominently the presence of noise in the measurements. Since measurement noise is inevitable, MPO-MLE will often produce a state that is not positive semi-definite. A possible solution for that problem is to do the MLE using the purification of the state, i.e. using its purified matrix product state (PMPS) form, since, by construction, the PMPS-MLE produces a positive semi-definite estimate. Our results suggest that the PMPS-MLE can be more useful than the MPO-MLE in the sense that, because it never produces states that are not positive semi-definite, it is immune to the instability problem of the MPO-MLE, in which the MPO-MLE iterations exhibit divergent behavior in terms of the Frobenius relative norm distance (see chapter 5). Moreover, due to the structure of the PMPS-MLE algorithm (4.1) the PMPS-MLE is more computationally efficient than the MPO-MLE, although the MPO-MLE is sometimes faster (in terms of needed iterations) to get closer to the lab state (susbsection 5.2.1). The MPO-MLE algorithm's speed is the result of the different effect of compression in both algorithms.

It is important to mention that in [20] the authors show that the MPO and PMPS representations of a quantum state are not equivalent. That is, a quantum state can have an MPO representation with a small bond dimension but a PMPS representation with an arbitrarily large bond dimension, making PMPS-MLE for these cases impractical. With that said, it is not yet known how practical PMPS-MLE will be for states that are of physical interest, like thermal states of local Hamiltonians, since in [28] the authors report observing thermal states with an efficient PMPS representation. Combined with the results from [28], the results in this thesis provide grounds for further exploration of the feasibility of the PMPS-MLE algorithm for doing QST.

Furthermore, in section 5.3 we explored how truly scalable the PMPS-MLE and the MPO-MLE are for large system sizes. The conclusion that we can make about the scalability is bipartite. The first part is that, since we can only do a limited number of iterations in a reasonable amount of time and since the MLE on larger systems requires more iterations to reach the target reconstruction error that we set, a maximum system size will be reached for which we will not be able to reconstruct the state to within the set error. One possible solution that one might think of is to increase the width of the block on which we do our measurements, but since the number of measurements scale exponentially with the block width, we are limited by the measurement block width we can use. In that sense, the scalability of the PMPS-MLE and the MPO-MLE algorithms to large system sizes is limited by the number of iterations we can do, the target error we require, and the block width we are using. The second part of the conclusion is that if we fix the maximum number of iterations we can do (we fix it to 50000 iterations in section 5.3) and the block width, the number of iterations needed to reach a certain target error for the PMPS-MLE and the MPO-MLE algorithms scales approximately polynomially with the system size, with an exponent that increases as we tighten the target error. For the data that we have, the exponent of the polynomial was small. Our limited data does not tell us whether the increase in the exponent as we tighten the error (and allow for more iterations) will be tolerable.

As an extension to subsections 4.2.4 and 5.2.2, we also examined the performance of the PMPS-MLE and the MPO-MLE algorithms for large systems in the case of noisy measurements. What we found is that the instability in the MPO-MLE becomes more severe as the noise in the measurements increases, as predicted by figure 4.5. Furthermore, for both the MPO-MLE and PMPS-MLE, the reconstruction error increases as the noise in the measurements increases.

Finally, in chapter 6, we propose modifications to the MLE, MPO-MLE, and PMPS-MLE algorithm to speed up their convergence. The scheme relies on the monotonic convergence of the entries of the state matrix to extrapolate the next state estimate. After testing the improved algorithm on systems of 3, 4, and 8 sites, it was found that the extrapolation scheme significantly speeds up the convergence of the MLE and MPO-MLE algorithms, and only slightly improves the PMPS-MLE. We suspect the small improvement of the PMPS-MLE to be due to the fact that the extrapolation schemes extrapolate the pure state $|\psi\rangle_{k+1}$, from which the state of our system $\hat{\rho}_{k+1}$ is acquired by tracing out the ancillary sites.

7.1 Future Work

For future work, it would be interesting to examine the scaling of the iterations needed for convergence with systems size when noise is introduced in the measurements. It would also be interesting to explore the scaling of the MPO-MLE and PMPS-MLE convergence speed with the width of the measurement block. This question will be especially interesting to answer because, although the MPO-MLE and the PMPS-MLE will need less iterations to converge to a certain estimate state as the size of the measurement block increases, they will need more time to do so since the number of measurements (and hence the number of elements in out POVM set) scales exponentially with the size of the measurement block.

Moreover, it would be interesting to compare the improved MLE algorithms in [10] and [27] with the algorithm we propose in chapter 6, especially for larger system sizes. We also leave it for future work to explore how the improved MPO-MLE algorithm performs for cases where the MPO-MLE becomes unstable due to compression error. Recall that the compression error occurs because, during the MLE iterations, the estimate state might have a larger bond dimension than the bond dimension we are compressing to (4.2.3). Because the improved MPO-MLE "skips" some of these states by extrapolating an estimate state which is closer to the lab state (and so has a closer bond dimension to that of the lab state), we speculate that the improved MPO-MLE will be less prone to the instabilities caused by compression error.

Chapter 8

The Codes and Computational Setup

8.1 The Computational Setup

The plots in figures 4.1, 4.2, 4.3, 5.1, 5.2, 5.6, 5.7, 5.8, and 5.9 were calculated on the JUSTUS supercomputer [1] in Ulm University. The cluster consists of 444 processing nodes, with each node consisting of two Intel Xeon E5-2630v3 (Haswell, 8-core, 2.4 GHz) processors. The Numdb library was used for submitting and collecting jobs from the JUSTUS supercomputer [2].

8.2 Code

The MPnum library [3] and the MPO-tomography-python library [4] were used for doing all MLE calculations. The functions in the library were designed based on the Matrix Product notation explained in chapter 3 and in [24]. The codes used for all plots in this thesis consist of the same core.

8.2.1 MPO-MLE

First, we generate a random positive semi-definite lab state by generating a random MPO and then multiplying it by its adjoint. Multiplication by the adjoint is done using the `mpnum.mpsmpo.pmps_to_mpo()` function, which produces a positive semi-definite MPO due to the same reason in equation (3.6):

```
rng = np.random.RandomState(seed=1)
pmps_lab = factory.random_mpo(sites=SITES, ldim=2, bdim=sqrt(BDIM), randstate=rng)
rho_lab = mpnum.mpsmpo.pmps_to_mpo(pmps_lab)
```

Next, the initial state of the MPO-MLE or the PMPS-MLE is initialized as pure state using the `factory.eye()` function:

```
rho_init = factory.eye(SITES , 2) / 2**SITES
```

Then, the outcome probabilities of the POVM elements are calculated by doing

(MWIDTH is the block size):

```
mypovm = mpnum.povm.MPPovm.from_local_povm(mpnum.povm.pauli_povm(2), MWIDTH)
expctvals = [obs.to_array().real for obs in mypovm.expectations(rho_lab)]
```

If we want to simulate noisy measurements, we specify the number of measurements for the POVM elements of each Pauli basis rotations, and then we pass the draw from a multinomial distribute based on the exact probabilities:

```
def get_freqs(expctvals, MWIDTH, samples_per_pauli_setting):
       #seed positions
        positions = [[] for l in range(3**MWIDTH)]
        X = list(it.product([0,2,4], repeat=MWIDTH))
        for i in range(3**MWIDTH):
            positions[i].append(X[i])

        #all possible positions
        import operator
        X = list(it.product([0,1], repeat=MWIDTH))
        X = X[1:]
        for i in range(3**MWIDTH):
            for a in X:
                positions[i].append(tuple(map(operator.add, positions[i][0], a)))

        #fetch probabilities
        import copy
        expctvalsflat = np.array(expctvals).reshape(-1)
        probspersetting = copy.deepcopy(positions)
        for i in range(3**MWIDTH):
            for j in range(2**MWIDTH):
                probspersetting[i][j] = expctvalsflat[int(
                  ''.join(map(str,positions[i][j])), 6)]

        #sample
        frqs = copy.deepcopy(probspersetting)
        for i in range(3**MWIDTH):
                frqs[i] = list(rng.multinomial(samples_per_pauli_setting,
                 probspersetting[i] / sum(probspersetting[i]), 1)[0])

        #put freqs back in place in the flattened expctvals array
        frqsnew = expctvalsflat
        for i in range(3**MWIDTH):
            for j in range(2**MWIDTH):
                frqsnew[int(''.join(map(str,positions[i][j])), 6)] = frqs[i][j]

         #reshape
        frequencies = frqsnew.reshape(np.array(expctvals).shape)


        return frequencies
```

Then, we create an MLE generator which will generate the estimate state in each iteration. In this step the algorithm is fed the compression bond dimension for the estimate state (we are using the double compression scheme so we have to feed the algorithm two compression bond dimension, `MPO_COMP_BDIM` and `MPO_COMP_BDIM_MIDDLE`) and the $\hat{R}$ propagator in equation (4.2). In this step we also specify the compression method. We use variational compression through out this thesis (see section 3.4).

```
mle = ml.MlEstimator(SITES, 2, mypovm, compr_args = {'method': 'var',
 'bdim': R_COMP_BDIM})

rho_gen = mle._iterative_solution(expctvals, rho_init, compr_args = {'method': 'var',
'bdim': MPO_COMP_BDIM}, compr_args_interm = {'method': 'var',
 'bdim': MPO_COMP_BDIM_MIDDLE})
```

The final step is to carry on with the iterations:

```
 for i in range(ITERATIONS):
     rho_est, _ = next(rho_gen)
```

8.2.2 PMPS-MLE

Until the creation of the MLE generator, the code for the PMPS-MLE is the same as in the MPO-MLE. For the PMPS-MLE, that step is as follows:

```
mle = ml.MlEstimator(SITES, 2, mypovm, compr_args = {'method': 'var',
 'bdim': R_COMP_BDIM})

pmps_gen = mle._iterative_solution(expctvals, pmps_init, compr_args = {'method': 'var',
'bdim': MPO_COMP_BDIM})
```

The `pmps_gen` object generates the iterations of the equation (4.1). The iterations are then done as

```
 for i in range(ITERATIONS):
     pmps_est, _ = next(pmps_gen)
```

To get the estimate state from `pmps_est`, we do the following:

```
 rho_est = mpnum.mpsmpo.pmps_to_mpo(pmps_est)
```

List of Figures

Bibliography

[1] `https://www.bwhpc-c5.de/wiki/index.php/Category:BwForCluster_Chemistry`. The authors acknowledge support by the state of Baden-Wrttemberg through bwHPC and the Germany Research Foundation (DFG) through grant no INST 40/467-1 FUGG.

[2] Unpublished library due to Milan Holzäpfel.

[3] `http://mpnum.readthedocs.io/en/latest/`.

[4] Unpublished library due to Milan Holzäpfel and Daniell Suess.

[5] T. Baumgratz. Efficient system identification and characterization for quantum many-body systems. *PhD thesis*, 2014.

[6] T. Baumgratz, A. Nüßeler, M. Cramer, and M. B. Plenio. A scalable maximum likelihood method for quantum state tomography. *New Journal of Physics*, 15(12):125004, 2013.

[7] R. Blatt, H. Häffner, C. Roos, C. Becher, and F. Schmidt-Kaler. Ion trap quantum computing with ca+ ions. In *Experimental Aspects of Quantum Computing*, pages 61–73. Springer, 2005.

[8] J. Eisert. Entanglement and tensor network states. In *Modeling and Simulation*, page 520. 2013.

[9] J. Eisert, M. Cramer, and M. B. Plenio. *Colloquium* : Area laws for the entanglement entropy. *Rev. Mod. Phys.*, 82:277–306, Feb 2010.

[10] Doglas S. Gonçalves, Márcia A. Gomes-Ruggiero, and Carlile Lavor. Global convergence of diluted iterations in maximum-likelihood quantum tomography. *Quantum Info. Comput.*, 14(11-12):966–980, September 2014.

[11] H. Häffner, W. Hänsel, C. Roos, J. Benhelm, M. Chwalla, T. Körber, U. D. Rapol, M. Riebe, P. O. Schmidt, C. Becher, O. Gühne, W. Dür, and R. Blatt. Scalable multiparticle entanglement of trapped ions. *Nature*, 438(7068):643–646, 2005.

[12] M. Holzäpfel, T. Baumgratz, M. Cramer, and M. B. Plenio. Scalable reconstruction of unitary processes and hamiltonians. *Phys. Rev. A*, 91:042129, Apr 2015.

[13] J. P. Home, M. J. McDonnell, D. M. Lucas, G. Imreh, B. C. Keitch, D. J. Szwer, N. R. Thomas, S. C. Webster, D. N. Stacey, and A. M. Steane. Deterministic entanglement and tomography of ionspin qubits. *New Journal of Physics*, 8(9):188, 2006.

[14] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, second edition, 2012. Cambridge Books Online.

[15] Z. Hradil, J. Řeháček, J. Fiurášek, and M. Ježek. 3 maximum-likelihood methodsin quantum mechanics. In *Quantum state estimation*, pages 59–112. Springer Berlin Heidelberg, 2004.

[16] Y. Huang. Classical simulation of quantum many-body systems. 2007.

[17] Jaroslav Řeháček, Zden ěk Hradil, E. Knill, and A. I. Lvovsky. Diluted maximum-likelihood algorithm for quantum tomography. *Phys. Rev. A*, 75:042108, Apr 2007.

[18] D. F. V. James, P. G. Kwiat, W. J. Munro, and A. G. White. Measurement of qubits. *Phys. Rev. A*, 64:052312, Oct 2001.

[19] M. Kliesch, D. Gross, and J. Eisert. Matrix-product operators and states: Np-hardness and undecidability. *Phys. Rev. Lett.*, 113:160503, Oct 2014.

[20] G. De las Cuevas, N. Schuch, D. Prez-Garca, and J. I. Cirac. Purifications of multipartite states: limitations and constructive methods. *New Journal of Physics*, 15(12):123021, 2013.

[21] J. L. O'Brien, G. J. Pryde, A. G. White, T. C. Ralph, and D. Branning. Demonstration of an all-optical quantum controlled-not gate. *Nature*, 426(6964):264–267, 2003.

[22] K. J. Resch, P. Walther, and A. Zeilinger. Full characterization of a three-photon greenberger-horne-zeilinger state using quantum state tomography. *Phys. Rev. Lett.*, 94:070402, Feb 2005.

[23] L. Rippe, M. Nilsson, S. Kröll, R. Klieber, and D. Suter. Experimental demonstration of efficient and selective population transfer and qubit distillation in a rare-earth-metal-ion-doped crystal. *Phys. Rev. A*, 71:062328, Jun 2005.

[24] U. Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, 2011.

[25] N. Schuch. Condensed matter applications of entanglement theory. In *Quantum Information Processing. Lecture Notes of the 44th IFF Spring School 2013*. 2013.

[26] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Entropy scaling and simulability by matrix product states. *Phys. Rev. Lett.*, 100:030504, Jan 2008.

[27] Yong Siah Teo. Numerical estimation schemes for quantum tomography. *arXiv preprint arXiv:1302.3399*, 2013.

[28] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac. Matrix product density operators: Simulation of finite-temperature and dissipative systems. *Phys. Rev. Lett.*, 93:207204, Nov 2004.