TECHNICAL UNIVERSITY OF MUNICH

School of Computation, Information and Technology
Informatics

Master's Thesis in Informatics

Quantum Gate Simulation Using Tree Tensor Networks

Juxhin Allaisufi

TECHNICAL UNIVERSITY OF MUNICH

School of Computation, Information and Technology
Informatics

Master's Thesis in Informatics

Quantum Gate Simulation Using Tree Tensor Networks

Simulation von Quantengatteroperationen mittels Tree Tensor Networks

Author:	Juxhin Allaisufi
Supervisor:	Prof. Dr. Christian B. Mendl
Advisors:	M.Sc. Richard Milbradt
Submission Date:	31.03.2025

Disclaimer

I confirm that this Master's thesis is my own work and I have documented all sources and material used.

(Date) (Juxhin Allaisufi)

Acknowledgements

First of all, I would like to thank Prof. Dr. Christian B. Mendl for trusting me in completing this thesis, and M.Sc. Richard Milbradt for his guidance and input throughout the entire thesis.

I would like to continue with thanking my dear friends, Sara, Relisa, Zajmena and Moreno, without whose continuous and genuine support I would not have been to able finish this chapter of my life.

In addition, I would like to thank all the members of my family, my mother, sister, nephews, two cousins, my uncle and his wife, and my beautiful grandparents for their unconditional loving.

Dhe së fundmi dua të falënderoj tim atë, shpresoj të jesh krenar, babë!

Abstract

This thesis explores the simulation of quantum circuits using tree tensor networks (TTNs), a structured representation well-suited for modeling quantum systems with limited entanglement. The project focuses on simulating quantum gate behavior, visualizing qubit state evolution, and testing small-scale quantum protocols. As a case study, the encoding process of the Shor error-correcting code is implemented and extended to span 18 qubits, demonstrating the framework's scalability.

While some advanced operations are omitted due to current limitations, such as multi-controlled gates, the results confirm the correctness of the implemented logic and provide a foundation for future extensions. Planned improvements include expanding the supported gate set, validating additional gate types, and developing a testing infrastructure to ensure accuracy and stability. This work contributes to the broader effort of building efficient and extensible tools for quantum circuit simulation.

Zusammenfassung

Diese Arbeit untersucht die Simulation von Quantenschaltungen mithilfe von Tree Tensor Networks (TTNs), einer strukturierten Darstellung, die sich besonders gut für die Modellierung von Quantensystemen mit begrenzter Verschränkung eignet. Der Fokus liegt auf der Simulation des Verhaltens von Quantengattern, der Visualisierung der Zustandsentwicklung von Qubits sowie dem Testen kleiner Quantenprotokolle. Als Fallstudie wird der Kodierungsprozess des Shor-Fehlerkorrekturcodes implementiert und auf 18 Qubits erweitert, um die Skalierbarkeit des Frameworks zu demonstrieren.

Einige fortgeschrittene Operationen, wie beispielsweise mehrfach-kontrollierte Gatter, werden aufgrund aktueller Einschränkungen noch nicht unterstützt. Dennoch bestätigen die Ergebnisse die Korrektheit der implementierten Logik und bilden eine solide Grundlage für zukünftige Erweiterungen. Geplante Verbesserungen beinhalten die Erweiterung des Gattersatzes, die Validierung zusätzlicher Gattertypen sowie die Entwicklung einer Testinfrastruktur zur Sicherstellung von Genauigkeit und Stabilität. Diese Arbeit leistet damit einen Beitrag zum Aufbau effizienter und erweiterbarer Werkzeuge für die Simulation von Quantenschaltungen.

Contents

1 Introduction

By utilizing concepts from quantum mechanics including superposition, entanglement, and unitary evolution, quantum computing provides a radically novel method of computation. In the lack of large-scale, fault-tolerant quantum hardware, simulation tools are essential to the development, testing, and comprehension of quantum algorithms as the field develops. Tree tensor networks (TTNs), a family of tensor-based models renowned for their capacity to effectively describe specific entangled quantum states, are used in this thesis to investigate the modeling of quantum gates and small circuits.

The core objective of this work is to implement and simulate the behavior of both single- and multi-qubit quantum gates within a TTN framework. An abstract class structure is introduced, defining a common interface for all gates through the apply_gate, plot, and compare_results methods. This design enables flexibility and extensibility, and serves as a foundation for concrete implementations of key quantum operations, including the Pauli gates (X, Y, Z), the Hadamard gate, phase shift gates, and entangling gates such as CNOT, SWAP, and controlled phase operations. Each gate is defined not only by its action on qubits but also by its role in the broader context of quantum circuit simulation.

Visualization plays a central role in validating the simulation, with plots illustrating how the expectation values of local observables evolve over time. For instance, the application of the X gate clearly demonstrates the expected transition from the ground to the excited state in the Z basis, while the Z gate produces no change in Z-basis expectation when acting on a qubit in the $|+\rangle$ state, highlighting its phase-only effect. Multi-qubit gates such as CNOT are also analyzed in depth, showing how entanglement is generated and how the state of one qubit can conditionally affect another. These results not only verify the correctness of the implementation but also offer insight into the physical meaning of the gate operations within the TTN formalism.

As a practical application, this framework is used to simulate the encoding process of the Shor code, a foundational quantum error-correcting code designed to protect against arbitrary single-qubit errors. While the full correction logic is not implemented due to the current lack of multi-controlled gates such as CCNOT, the encoding steps are faithfully realized and repeated for two logical qubits across 18 physical qubits. This partial implementation demonstrates the structure of encoded states and lays the groundwork for future expansion toward complete quantum error correction procedures.

Controlled shift and phase shift gates are also implemented as part of this effort, though further validation is planned to confirm their correctness in full circuits.

Finally, the thesis outlines clear directions for future work. These include extending the gate set to incorporate CCNOT and other multi-controlled gates, validating more complex gate operations, and introducing an automated testing framework to ensure the stability and correctness of the simulation. Together, these contributions provide a flexible and scalable platform for simulating quantum gates and circuits using tensor networks, and offer a meaningful step toward high-fidelity quantum software tools.

2 Background

It is crucial to have a basic understanding of quantum computing and its guiding principles before diving into the backdrop of this thesis. A number of reputable books offer thorough introductions to this subject, shedding light on important ideas including quantum mechanics, qubits, quantum gates, and quantum algorithms.

Notable among them is Quantum Computation and Quantum Information by Michael A. Nielsen and Isaac L. Chuang [1], which remains one of the most influential references in the field. Additionally, Principles of Quantum Computation and Information by Giuliano Benenti, Giulio Casati, and Giuliano Strini [2] provides a structured approach to quantum information theory.

For a more structured introduction to the mathematical and physical foundations of quantum computing, An Introduction to Quantum Computing by Phillip Kaye, Raymond Laflamme, and Michele Mosca [3] is an excellent resource. More recent insights can be found in Introduction to Classical and Quantum Computing by T.G. Wong [4], which offers an accessible yet rigorous introduction to the field.

These works collectively form a solid basis for understanding the theoretical aspects of quantum computing, which is crucial for the discussions in this thesis.

2.1 Tensor Networks

Tensor networks are a powerful mathematical framework used to efficiently represent and manipulate high-dimensional data structures. They provide a structured way to decompose large tensors into smaller, interconnected components, significantly reducing the computational complexity associated with many-body systems, optimization problems, and machine learning models. The fundamental idea behind tensor networks is to exploit the entanglement properties of quantum systems, allowing for efficient simulation and computation in contexts where direct manipulation of large tensors would be infeasible [5]. Tensor networks have found extensive applications across multiple domains, including chemistry [6], open quantum systems [7], [8], [9], and condensed matter physics [10], [11]. Their utility extends further into holography, quantum state preparation, and large-scale quantum simulations [12], [13], [14], [15]. In machine learning, tensor-based methods play a crucial role in feature extraction, deep learning architectures, and efficient model training [16], [17], [18], [19].

Tensor networks have been widely applied in physics, particularly in the study of quantum many-body systems. Methods such as Matrix Product States(MPS)

and TTNs [20] enable the efficient representation of quantum states and aid in solving complex optimization problems. These techniques have been instrumental in advancing research in condensed matter physics and quantum simulations [21]. Beyond physics, tensor networks are increasingly being used in machine learning, where they offer an alternative to traditional deep learning architectures. By leveraging tensor decompositions, researchers have improved model interpretability and reduced memory requirements [22].

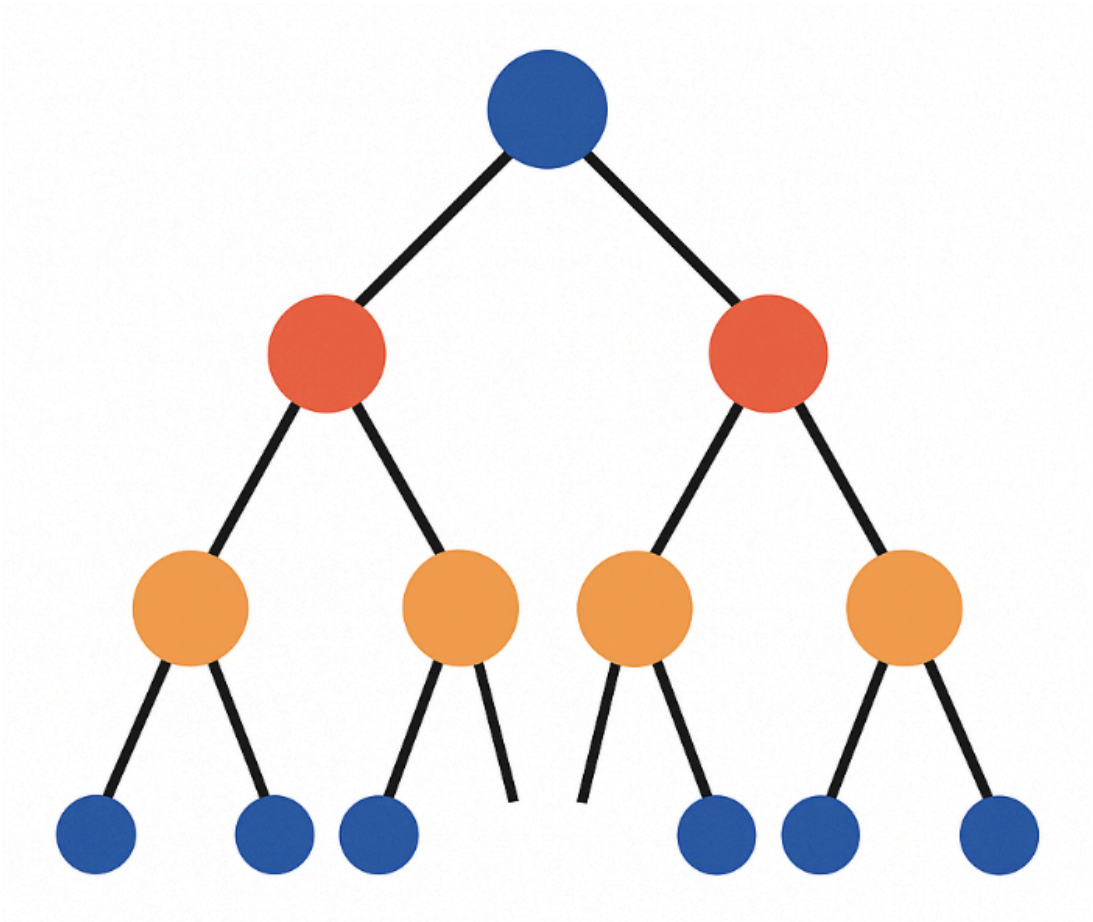

Figure 1: Tensor Network

Another promising direction is the intersection of tensor networks and probabilistic graphical models, which has led to new techniques for supervised learning. Recent research has demonstrated that tensor network-based models can outperform traditional machine learning methods in tasks such as image and speech classification [23]. Additionally, the development of open-source software, such as TensorNetwork, has facilitated the implementation of tensor network algorithms, making them more accessible to researchers working in both physics and artificial intelligence [24].

As tensor network methods continue to evolve, their impact is expected to grow across multiple disciplines, from quantum computing to large-scale optimization. Their ability to compress and efficiently process large datasets makes them a critical tool in the development of future computational models and simulations.

2.2 Tree Tensor Networs

TTNs are a specialized form of tensor networks that efficiently represent and process high-dimensional data by organizing tensors in a hierarchical, tree-like structure. This decomposition reduces the number of tensor contractions needed for complex calculations, making them particularly useful for systems with strong locality constraints and structured correlations. Unlike MPS, which arrange tensors in a linear sequence, they branch out into multiple layers, making them effective for problems involving hierarchical dependencies [25].

A major advantage of TTNs is their ability to encode quantum states with entanglement structures that naturally align with tree-like topologies. This feature is particularly valuable in quantum physics and computing, where they approximate wave functions and study many-body quantum systems. By reducing computational complexity, they facilitate the analysis of ground states, excited states, and quantum phase transitions, making them an essential tool in condensed matter physics. They have also been successfully used to optimize network structures dynamically, improving their accuracy for quantum simulations [26].

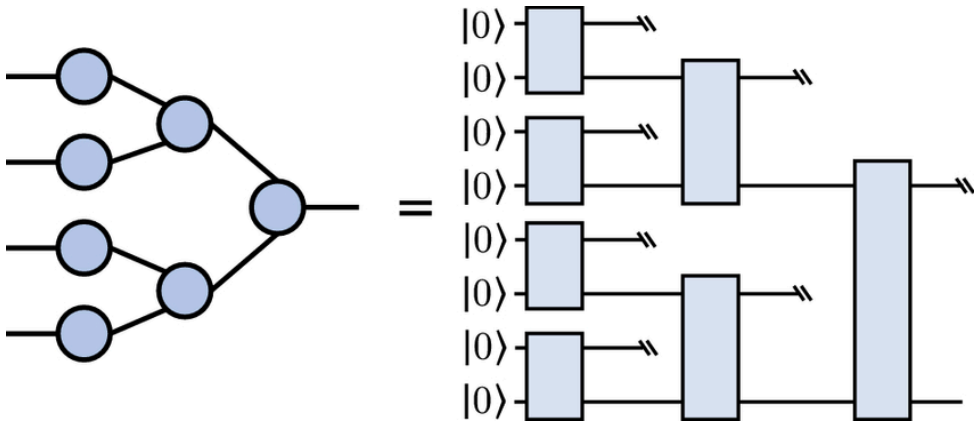

Figure 2: An example of the Tree Tensor Network (TTN) architecture. Source: Zhou & Du, Tensor Networks for Simulating Quantum Systems, 2023.

Beyond physics, their hierarchical structure has made them increasingly useful in machine learning and artificial intelligence. They efficiently process structured data,

making them well-suited for feature extraction, classification, and pattern recognition. In deep learning, TTNs offer an alternative to traditional neural networks by significantly reducing the number of parameters, making them ideal for applications requiring efficient data compression, such as image and speech recognition. Their ability to scale efficiently has been instrumental in improving computational performance in a range of AI applications [27].

Another important application is quantum circuit simulation, where they provide a scalable framework for representing quantum gates. Unlike traditional tensor networks, which become computationally expensive in large quantum systems, TTNs optimize the simulation of complex quantum algorithms. This makes them crucial for quantum information processing, including error correction, cryptography, and optimization problems. The hierarchical structure of TTNs allows for efficient state representations, enabling scalable classical simulations of quantum circuits [28].

In probabilistic modeling, TTNs help approximate high-dimensional probability distributions, aiding in Bayesian inference and graphical models. Their hierarchical structure enables scalable probabilistic reasoning, making them useful for modeling biological networks, financial systems, and social interactions.

As computational demands grow, TTNs are expected to become increasingly significant across multiple disciplines. Their ability to decompose high-dimensional problems into manageable components makes them invaluable for quantum simulations, machine learning, and probabilistic modeling. With continued advancements in tensor network algorithms and hardware acceleration, they are set to further drive innovation in computational science.

2.3 Quantum Bits

Classical information theory is built upon the concept of a bit, which exists in one of two distinct states: 0 or 1. This can be likened to a switch, where “on” corresponds to 1 and “off” to 0. In contrast, quantum information theory introduces the qubit, the quantum counterpart of the classical bit. Unlike classical bits, which are restricted to a single state at any given time, a qubit can exist in a superposition of both basis states, represented as $|0\rangle$ and $|1\rangle$. Mathematically, these states are expressed as $|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ and $|1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ within a Hilbert space. The notation used to describe these quantum states, known as Dirac notation or ket notation, represents them as column vectors.

In quantum mechanics, larger systems can be built by combining smaller ones. This method is used to create systems with multiple qubits. If the qubits are independent, their combined state can be described in the following way:

$$|\psi_1\psi_2\cdots\psi_n\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots |\psi_n\rangle \tag{1}$$

where $\psi_1, \psi_2, ..., \psi_n$ are the states of the individual qubits. The combined state is a *tensor product* of the individual states. The tensor product ($\otimes$) is a mathematical operation that combines two or more vectors to create a new vector. Assuming we have 2 qubits: $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ and $|\psi_2\rangle = \alpha_2|0\rangle + \beta_2|1\rangle$ with both qubits in $\mathbb{C}^2$, an arbitrary product state $|\psi\rangle$ in $\mathbb{C}^4$ from these qubits can be composed as:

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2}$$

with $|\alpha_{00}|^2 + |\alpha_{01}|^2 + |\alpha_{10}|^2 + |\alpha_{11}|^2 = 1$ and $|00\rangle, |01\rangle, |10\rangle, |11\rangle$ are the *computational basis states.* which are the tensor product of the individual basis states, e.g., $|00\rangle = |0\rangle \otimes |0\rangle = \binom{1}{0} \otimes \binom{1}{0}$.

2.4 Single Qubit Gates

In quantum computing, single-qubit gates operate on one single qubit. The most common single-qubit gates are the *Pauli gates*, the *Hadamard gate*, and the *Phase Shift gate*, which are represented by the matrices:

Pauli-X	Pauli-Y	Pauli-Z	Hadamard	Phase Shift
$\sigma_1 = \sigma_X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$	$\sigma_2 = \sigma_Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$	$\sigma_3 = \sigma_Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$	$H = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$	$S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$

Figure 3: Pauli, Hadamard, and Phase Shift matrices.

The Pauli-X gate is the quantum analogue of the NOT-gate. It flips the $|0\rangle$ to $|1\rangle$ and vice versa. The Pauli-Z flips the sign of the $|1\rangle$ state and leaves the $|0\rangle$ state unchanged. The Pauli-Y acts like a combination of the Pauli-X and Pauli-Z gates. It rotates the qubit states by π around the y-axis on the Bloch sphere. The X, Y, and Z gates are also called the *Pauli matrices*, and $\vec{\sigma} = (\sigma_1, \sigma_2, \sigma_3)$ is called a *Pauli vector* and is a vector of 2x2 matrices.

The Hadamard gate is one of the most used gates in quantum computing, and it changes the basis of the matrix from $\{|0\rangle, |1\rangle\}$ to $\{|+\rangle, |-\rangle\}$ and vice versa with $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

The Phase Shift Gate introduces a relative phase of $\pi/2$ to the $|1\rangle$ state while leaving $|0\rangle$ unchanged. This gate is commonly used in quantum algorithms to introduce controlled phase differences.

We can express all of the above in circuit notation:

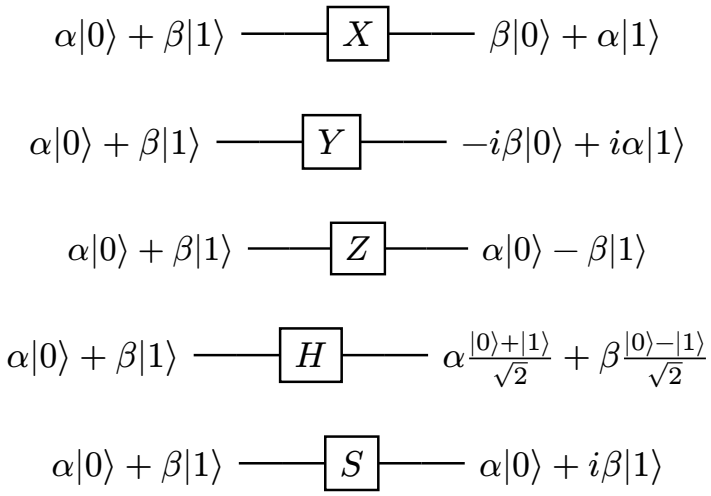

Figure 4: Most common single-qubit gates, including the Phase Shift Gate.

2.5 Multiple Qubit gates

Since single-qubit gates all have some rotation with a specific angle around the x, y, or z-axis, they can be represented in the Bloch sphere. However, multi-qubit gates are too complex for such representation. Instead, their behavior is described only using matrices.

The most common multi-qubit gate is the CNOT. It is a gate with two inputs: one *control* qubit and one *target* qubit. The CNOT gate flips the target qubit if the control qubit is in state $|1\rangle$ and does nothing if the control qubit is in state $|0\rangle$. In circuit notation:

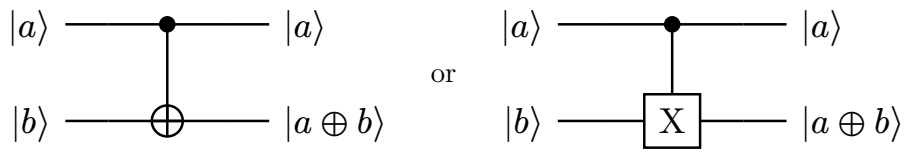

Figure 5: The CNOT gate.

With a, b $\in \{0,1\}$ and $\oplus$ means addition modulo 2 or just a simple logical XOR operator. The effect of the CNOT gate as a matrix can be expressed as:

$$\text{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3}$$

The effect of the CNOT gate on all the basis states is as follows:

$$|00\rangle \longmapsto |00\rangle,\ |01\rangle \longmapsto |01\rangle,\ |10\rangle \longmapsto |11\rangle,\ |11\rangle \longmapsto |10\rangle \tag{4}$$

Another important two-qubit gate is the SWAP gate, which interchanges the states of two qubits without modifying their individual values. Unlike the CNOT gate, which conditionally flips the target qubit based on the control qubit, the SWAP gate simply exchanges the quantum states of its two inputs. This operation is useful in quantum algorithms that require rearranging qubits in a register or when simulating physical systems where particles naturally exchange positions.

In circuit notation, the SWAP gate is represented as:

Figure 6: The SWAP gate.

The effect of the SWAP gate can be described using the following matrix representation:

$$\text{SWAP} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5}$$

The transformation can be explicitly seen as:

$$|00\rangle \longmapsto |00\rangle,\ |01\rangle \longmapsto |10\rangle,\ |10\rangle \longmapsto |01\rangle,\ |11\rangle \longmapsto |11\rangle \tag{6}$$

If one qubit is initially in state $|a\rangle$ and the other in $|b\rangle$, applying the SWAP gate exchanges their positions within the quantum register. Since this operation solely interchanges the qubits without modifying their individual states, it is both deterministic and reversible. This makes the SWAP gate particularly useful in quantum circuits that require rearranging qubits while preserving their quantum information.

The Controlled Phase Shift Gate (CPhase) is a two-qubit gate that applies a phase shift to the target qubit only when the control qubit is in state $|1\rangle$. Unlike the CNOT gate, which flips the target qubit, the CPhase gate modifies its phase without altering probability amplitudes. If the control qubit is $|0\rangle$, the target remains unchanged, but if it is $|1\rangle$, the target acquires a phase factor $e^{\{it\theta\}}$. Special cases include the Controlled-Z (CZ) gate when $t\theta = pi$, the Controlled-S (CS) gate when $t\theta = p\frac{i}{2}$, and the Controlled-

T (CT) gate when $t\theta = p\frac{i}{4}$. The CPhase gate is crucial in quantum algorithms like the Quantum Fourier Transform (QFT), quantum error correction, and entanglement generation.

2.6 Quantum Circuits

Quantum circuits are composed of a structured sequence of quantum gates that act on qubits to perform computations. The circuit model follows a layered architecture where qubits are initialized, manipulated using unitary transformations, and then measured to extract classical information. Unlike classical circuits, where logic gates deterministically map inputs to outputs, quantum circuits exploit superposition, entanglement, and interference to process information in fundamentally different ways.

The structure of a quantum circuit begins with qubit initialization, where each qubit is set to a defined state, typically $\mid 0 \rangle$ $|0\rangle$. The next stage involves quantum gate operations, where unitary transformations modify the qubit states. These gates can be classified into single-qubit and multi-qubit gates. Single-qubit gates, such as the Hadamard (H), Pauli (X, Y, Z), and phase gates (S, T), manipulate the state of individual qubits. Multi-qubit gates, such as the CNOT and Toffoli gates, introduce interactions between qubits, enabling the creation of entanglement and the implementation of complex quantum operations.

A key characteristic of quantum circuit structure is its depth, which refers to the number of sequential layers of gates that must be executed. A deeper circuit requires more operations and is susceptible to noise and decoherence, making depth optimization a critical concern in quantum computing. Parallelism within quantum circuits can help mitigate these issues by allowing multiple gates to operate simultaneously on different qubits, thereby reducing overall execution time.

Quantum circuits are often represented using quantum circuit diagrams, where time progresses from left to right. Wires correspond to qubits, and quantum gates are applied along these wires to modify their states. Measurement, typically performed at the end of the circuit, collapses qubit states into classical binary outcomes, making the results accessible for classical post-processing.

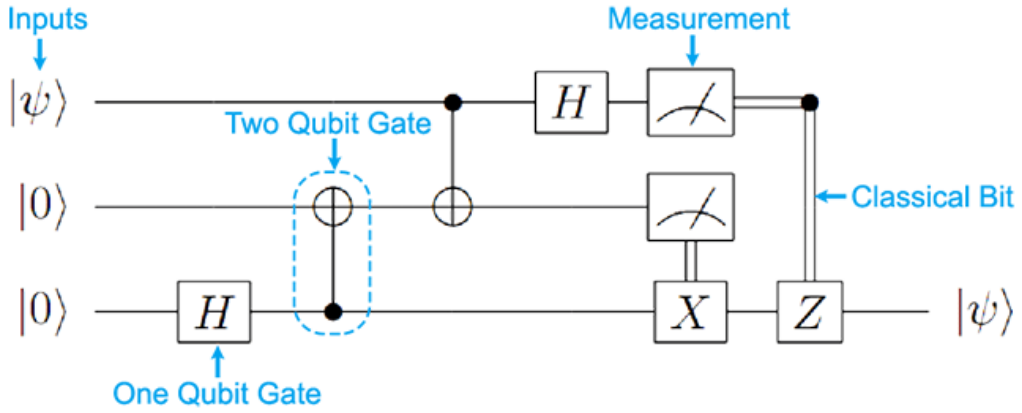

Figure 7: Visual explanation of quantum circuits.
Source: Siddhartha Rao, Quantum Computation Primer – Part 2, 2020.

Scalability is a major challenge in quantum circuit design. As the number of qubits grows, circuit complexity increases, making it difficult to maintain coherence and fidelity. Techniques such as circuit compilation, gate decomposition, and tensor network methods [29], [30] are crucial for optimizing quantum circuits. These methods help reduce gate count, minimize circuit depth, and enhance fault tolerance, making large-scale quantum computations more feasible.

As quantum technology advances, designing structured and efficient quantum circuits will be essential for unlocking the full potential of quantum computing. Well-optimized circuit architectures will enable the implementation of powerful quantum algorithms, bringing quantum advantage closer to practical realization.

2.7 Tree Tensor Networks in PyTreeNet

In PyTreeNet, we use tensor networks to simulate quantum circuits by encoding quantum gates as tensors and evolving quantum states through tensor contractions. Each gate in the circuit is represented by its corresponding unitary matrix which is applied to the quantum state vector. To perform these operations, we define the Hamiltonians of the gates, which describe their time evolution under the Schrödinger equation, allowing us to construct operators for continuous transformations if needed.

When simulating a quantum circuit, we initialize a tensor network representation of the quantum state, typically starting with the computational basis state Each quantum gate (Pauli gates, Hadamard, CNOT, Phase Shift, etc.) is applied by contracting its corresponding tensor with the state tensor at the appropriate qubit positions. Multi-

qubit gates like CNOT require entangling operations where tensors are contracted across multiple sites in the network.

For time evolution or parameterized gates we exponentiate the Hamiltonian using matrix exponentiation techniques to compute the gate operator:

$$U = e^{-iHt} \tag{7}$$

where H is the Hamiltonian of the gate and t is an evolution parameter. This formulation allows us to simulate continuous transformations, such as rotational gates or phase-shifting operations.

For circuit execution, we follow a sequential tensor contraction strategy where each applied gate updates the state representation iteratively. This approach ensures that we maintain an efficient representation of the quantum state while leveraging the hierarchical structure of the tree tensor network (TTN to optimize contractions. The final measurement is simulated by computing probability distributions from the state tensor, extracting observables like expectation values of Pauli operators.

By leveraging PyTreeNet's automatic differentiation and tensor contraction capabilities we ensure efficient simulation of quantum circuits, allowing us to scale beyond brute-force matrix multiplication methods typically used in quantum simulation

3 Implementing Quantum Gate Operations

The implementation of quantum gate simulations in this work is based on TTNs, integrated as an extension to the PyTreeNet library. This extension introduces a framework for applying and analyzing quantum gates within a tensor network representation, enabling efficient quantum simulations with reduced computational overhead. The developed framework is structured around a core abstract class, QuantumGate, which serves as a template for implementing specific quantum gate operations. This design ensures modularity, allowing seamless integration and expansion with additional gates in the future.

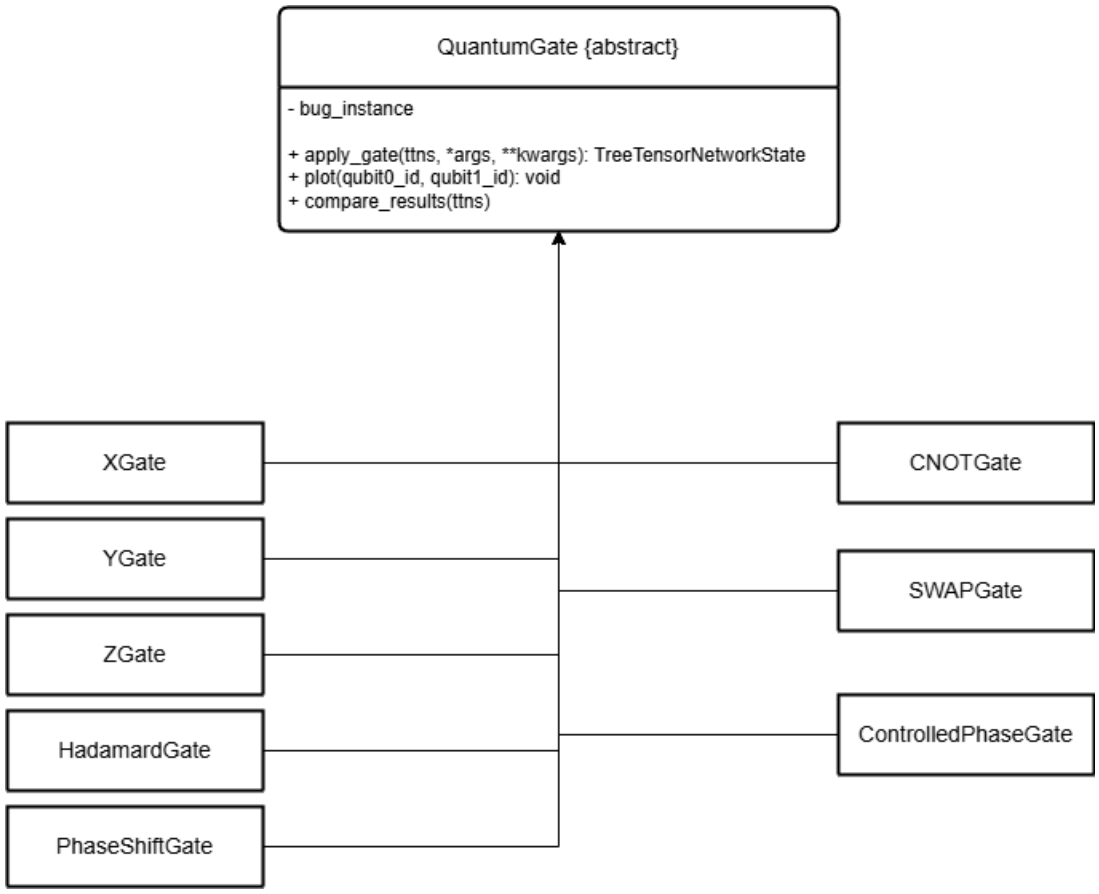

Figure 8: UML Diagram of Quantum Gates

The QuantumGate class defines three essential methods: apply_gate, which applies a quantum transformation to a qubit or set of qubits within the Tree Tensor Network State (TTNS); plot, which visualizes the local magnetization of the system after applying the gate; and compare_results, which facilitates benchmarking by comparing the simulated results against different final times for each gate. These methods provide

a structured and flexible approach to handling quantum operations within the tensor network framework.

Building upon this base class, specific quantum gates such as Pauli gates (X, Y, Z), the Hadamard gate, the CNOT gate, the SWAP gate, and phase shift gates have been implemented as subclasses. Each of these gates overrides the apply_gate method, ensuring that the appropriate unitary transformation is applied to the quantum state encoded in the tree tensor network. The implementation relies on tensor product representations of quantum operators, ensuring compatibility with the PyTreeNet infrastructure.

For instance, in the case of the Pauli-X gate, the apply_gate method constructs a tensor product term representing the Pauli-X matrix applied to the target qubit. This representation is then converted into a Hamiltonian, from which a Tree Tensor Network Operator (TTNO) is derived. Using the BUG (Basis-Update and Galerkin) method from PyTreeNet, the time evolution of the system is computed, allowing the quantum gate to be effectively simulated. The final state is updated based on this evolution, ensuring that the transformation is accurately represented in the TTN structure.

Similarly, the Hadamard gate follows a comparable approach but utilizes the well-known Hadamard matrix to generate quantum superpositions. This gate is essential for quantum algorithms as it enables the transformation of basis states into equal superpositions, a crucial property for many quantum computational tasks.

The CNOT gate, in contrast, introduces entanglement between two qubits by conditioning the application of an X operation on the control qubit's state. Its implementation involves defining a two-qubit tensor product that captures the conditional nature of the operation and using PyTreeNet's Hamiltonian-based time evolution to update the network state. The CNOT gate can be won from the Hermitian operator:

$$\mathrm{CNOT} = e^{-i\frac{\pi}{4}(I_1 - Z_1)(I_2 - X_2)}$$

Figure 9: Hermitian Operator of CNOT gate

And from this Hermitian operator, we generate the Hamiltonian of the CNOT gate. As seen form the operator, the Pauli-Z gate is applied to the control qubit, and the X-Pauli gate is applied to the target qubit.

The SWAP gate enables the exchange of quantum states between two qubits. Its implementation in the tensor network framework involves representing the three fundamental swap interactions (XX, YY, ZZ) within the Hamiltonian. By using tensor product decompositions, the correct unitary transformation is encoded in the network, ensuring that the state exchange is accurately simulated. Below is the formula of how to generate the SWAP gate from time evolution.

$$\text{SWAP} = e^{-it\frac{\pi}{4}(X_1X_2+Y_1Y_2+Z_1Z_2)}$$

Figure 10: Time evolution for the SWAP gate

A more complex operation included in this framework is the phase shift gate and its controlled variant, the Controlled Phase Gate. These gates introduce phase rotations to specific qubits and are particularly useful in quantum algorithms where controlled interference patterns play a role in computational speed-up. Their implementation involves constructing parameterized unitary matrices representing phase rotations and integrating them into the tensor network via Hamiltonian-based transformations.

The framework allows for plotting these expectation values over time, enabling visualization of the gate effects on the quantum state. These expectation values, particularly those of the Z operator, provide insight into the system's dynamics under various quantum transformations.

By incorporating quantum gate simulations into the tree tensor network architecture, the PyTreeNet library gains a new feature that improves its capacity to effectively simulate quantum systems. For systems with a lot of qubits, this technique offers a scalable method of modeling quantum circuits by utilizing unitary time evolution and tensor decompositions. The framework's modular design guarantees flexibility and for future additions like the addition of increasingly intricate multi-qubit gates and error correction systems.

By showing how tree tensor networks can be used efficiently for quantum gate operations, this work advances the field of quantum simulations. The potential of TTNs as a potent tool for quantum computing research is highlighted by their capacity to computationally efficiently and systematically imitate entanglement, superposition, and controlled operations. This approach opens the door for further optimizations and

practical uses in quantum information processing by enabling the study of quantum algorithms in a tensor network environment.

3.1 Time Evolution

This section describes the mechanism for applying quantum gates within a TTN framework, enabling the simulation of quantum circuits in a structured and scalable manner. The core functionality is implemented through the apply_gate function, which executes individual gate operations, and discrete_time_evolution, which allows for the sequential application of a set of quantum gates to evolve the quantum state over time.

A tree tensor network state and a series of quantum gates are intended inputs for the discrete_time_evolution function. Every gate in this series is represented as a tuple, with the kind of gate being indicated by the first element and the qubits participating in the operation being specified by the remaining items. Using the apply_gate function, the function applies the appropriate gate and its associated qubits to the network after iterating through this sequence. To guarantee that the entire quantum circuit is simulated in the right sequence, the quantum state is updated at each stage to represent the change that the gate imposes. The final quantum state at the end of the procedure captures the combined impact of all the gates that were used.

The primary execution unit for applying quantum gates is the apply_gate function. The function retrieves all relevant gate information from a predefined dictionary, GATE_CONFIGS, to guarantee flexibility and efficiency. The corresponding gate class, which provides the actual implementation, the default time step size, which establishes the discretization for time evolution, the default final time, which regulates the duration of the system's evolution under the gate operation, and any other configuration settings necessary for the simulation are all stored in this dictionary. Adaptable execution is made possible by overriding the defaults contained in GATE_CONFIGS with user-provided custom values for the time step size, final time, or configuration parameters.

The function establishes if the gate being applied is a single-qubit or two-qubit operation after the required parameters have been set. An X, Y, Z, Hadamard, or Phase Shift gate, for example, applies the gate to the designated qubit with the appropriate parameters if it functions on a single qubit. To guarantee that the proper transformation is executed, an extra phase angle parameter is supplied explicitly to the P_phi (phase shift) gate.

The function takes the control and target qubit identifiers and applies the appropriate transformation for two-qubit operations, including the CNOT and SWAP

gates, to guarantee that both qubits interact as intended. Similar to the P_phi gate, the Controlled Phase Shift (CP_phi) gate applies a phase shift parameter to two qubits rather than just one. The function makes sure that every transformation is carried out accurately and that the quantum state changes in accordance with the specified quantum circuit by differentiating between single-qubit and multi-qubit operations.

This approach offers a scalable and adaptable framework for applying quantum gates in a tree tensor network environment. The system may effectively imitate quantum circuits while preserving the benefits of tensor network-based representations by organizing the execution using the apply_gate function and managing sequential transformations with discrete_time_evolution. This method is ideal for investigating intricate quantum algorithms because it guarantees that quantum state evolution will continue to be computationally possible even as the number of qubits rises.

4 Results

In the following sections, we present the results of our quantum gate simulations using tree tensor networks. The goal is to provide a clear and intuitive view of how quantum gates behave when applied within a TTN-based simulation environment. We include visual representations of the gates and illustrate how the quantum states of the qubits evolve after each application. Additionally, we demonstrate the successful integration and performance of the discrete_time_evolution algorithm in simulating gate dynamics. These results collectively validate the correctness and effectiveness of our overall implementation.

4.1 Pauli-X Gate

The first result we present corresponds to the implementation of the `apply_gate` method for the Pauli-X gate, represented by the XGate class. The accompanying plot displays the expectation value of the Z-operator over time, effectively visualizing the local magnetization change induced by the application of the gate. Initially, the expectation value of the Z-operator is $+1$, indicating that the qubit is in the ground state $|0\rangle$, which is an eigenstate of the Z operator. As time evolves, the expectation value decreases smoothly and continuously, ultimately reaching -1. This behavior is consistent with the action of the Pauli-X gate, which performs a bit-flip operation by mapping $|0\rangle$ to $|1\rangle$. The observed evolution reflects a cosine-like transition, characteristic of coherent quantum dynamics under unitary evolution governed by a Hamiltonian containing the Pauli-X operator. This result confirms the correct implementation of the `apply_gate` method within the tree tensor network framework. Moreover, the successful visualization using the plot method demonstrates the ability of our simulation to track changes in local observables over time. The smooth and complete transition from $+1$ to -1 in the expectation value indicates that the simulation correctly captures the essential physics of the Pauli-X gate, validating both the gate logic and the underlying discrete_time_evolution algorithm used in our model.

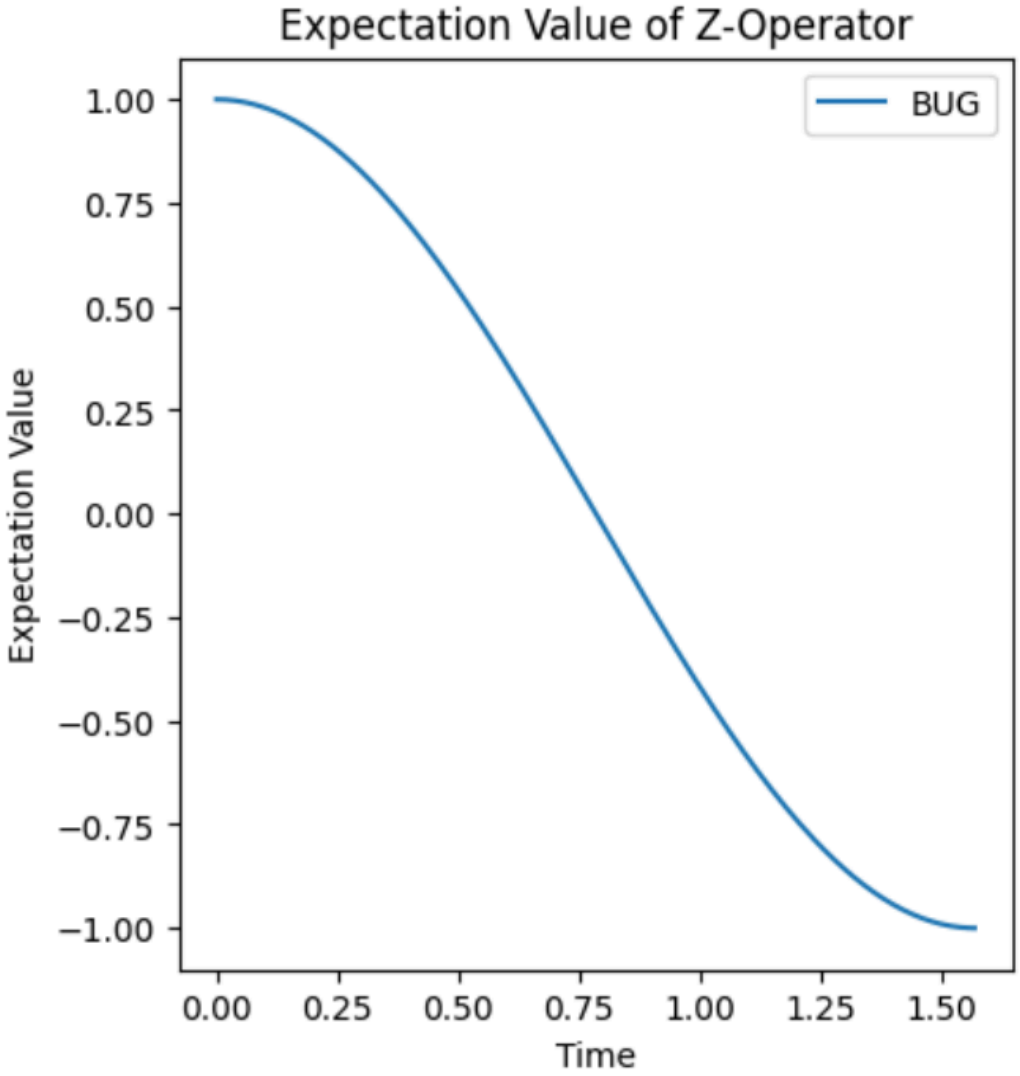

Figure 11: Plot of local magnetization change over time for the Pauli-X gate.

4.2 Pauli-Z Gate

We now present the result of the `apply_gate` method for the Pauli-Z gate, as implemented in the ZGate class. The corresponding graph shows the expectation value of the Z-operator over time. At first glance, the result may appear unusual, as the expectation value remains constant at zero throughout the entire evolution. However, this outcome is consistent with the physical properties of the Pauli-Z operation when analyzed more closely. In our simulations, local magnetization is computed using the expectation value of the Z-operator. Since the Pauli-Z gate acts by flipping the phase of the $|1\rangle$ component of the qubit without altering its population probabilities, its application does not result in a measurable change in the Z-basis expectation value when starting from a balanced or symmetric state, such as $|+\rangle$ or a superposition. In such cases, the contributions

from the $|0\rangle$ and $|1\rangle$ components cancel out, resulting in a net expectation value of zero. This result confirms the correct implementation of the gate and the consistency of our simulation framework with quantum mechanical principles. The behavior observed also reflects the inherent limitations of using Z-operator-based measurements to detect purely phase-based transformations, which is precisely the nature of the Pauli-Z operation.

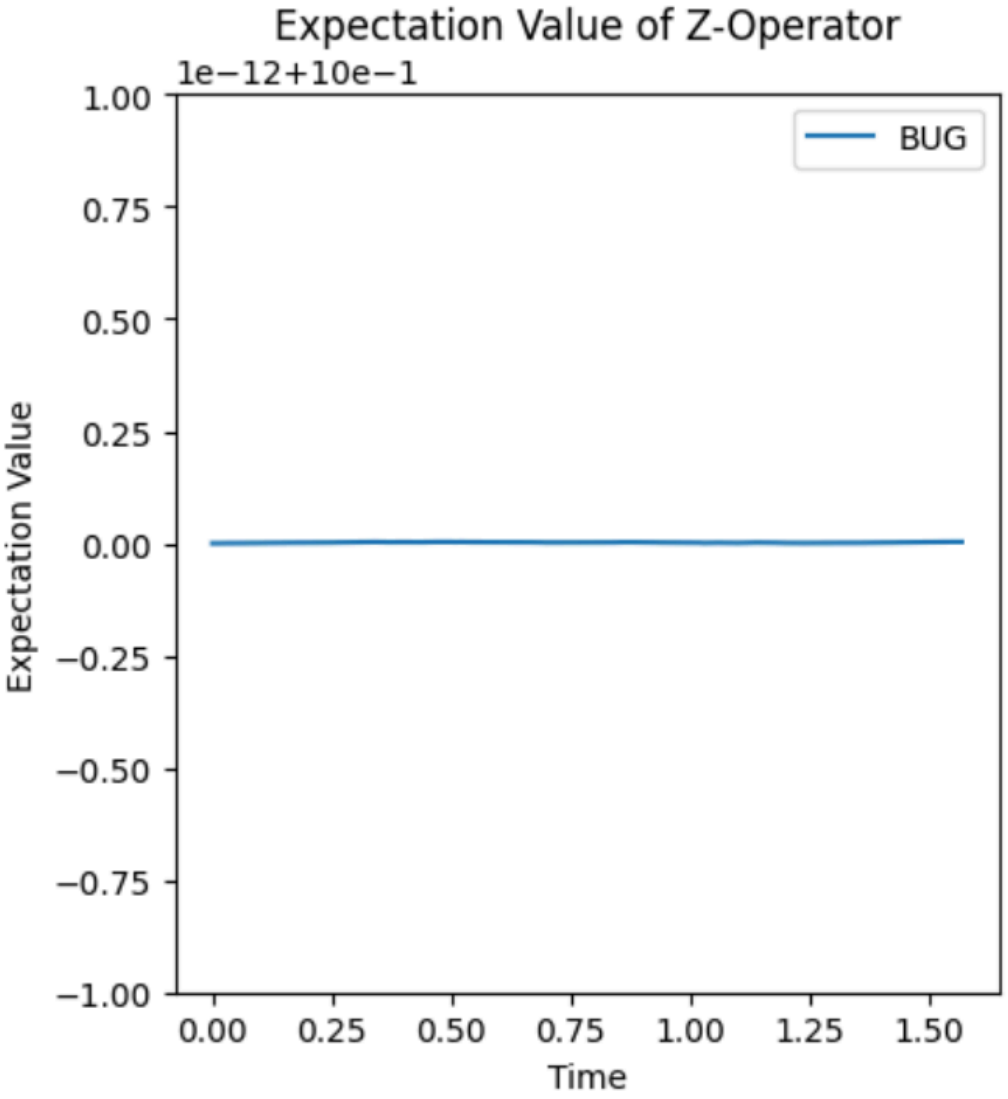

Figure 12: Plot of local magnetization change over time for the Pauli-Z gate.

4.3 CNOT Gate

For the CNOT gate simulation, the initial quantum state is prepared with qubit 0 being in a superposition state and qubit 1 being in the computational basis state $|0\rangle$. Specifically, qubit 0 is initialized to:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{8}$$

, which corresponds to the equal superposition state often denoted as $|+\rangle$, while qubit 1 is initialized to the pure state $|0\rangle$. The combined two-qubit system is therefore in the product state:

$$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \tag{9}$$

.

After applying the CNOT gate, where qubit 0 is the control and qubit 1 is the target, the quantum system evolves into the entangled state:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{10}$$

. This transformation implies that whenever the control qubit is in state $|1\rangle$, the target qubit is flipped from $|0\rangle$ to $|1\rangle$, as per the CNOT gate definition.

The results presented in the plot show the time evolution of the expectation value of the Z-operator on both qubits. For qubit 0, the expectation value remains essentially zero throughout the simulation, with only negligible numerical fluctuations on the order of 10^{-15} due to floating-point precision. This is consistent with the qubit being in the $|+\rangle$ state, which is an equal superposition of $|0\rangle$ and $|1\rangle$ and yields a Z expectation value of zero. For qubit 1, the expectation value of the Z-operator decreases smoothly from +1 to approximately 0, reflecting the evolution from a definite $|0\rangle$ state toward a superposition or entangled state as the system approaches the final Bell state. This behavior aligns perfectly with theoretical expectations and confirms the correct implementation of both the `apply_gate` method for the CNOT gate and the underlying discrete_time_evolution algorithm.

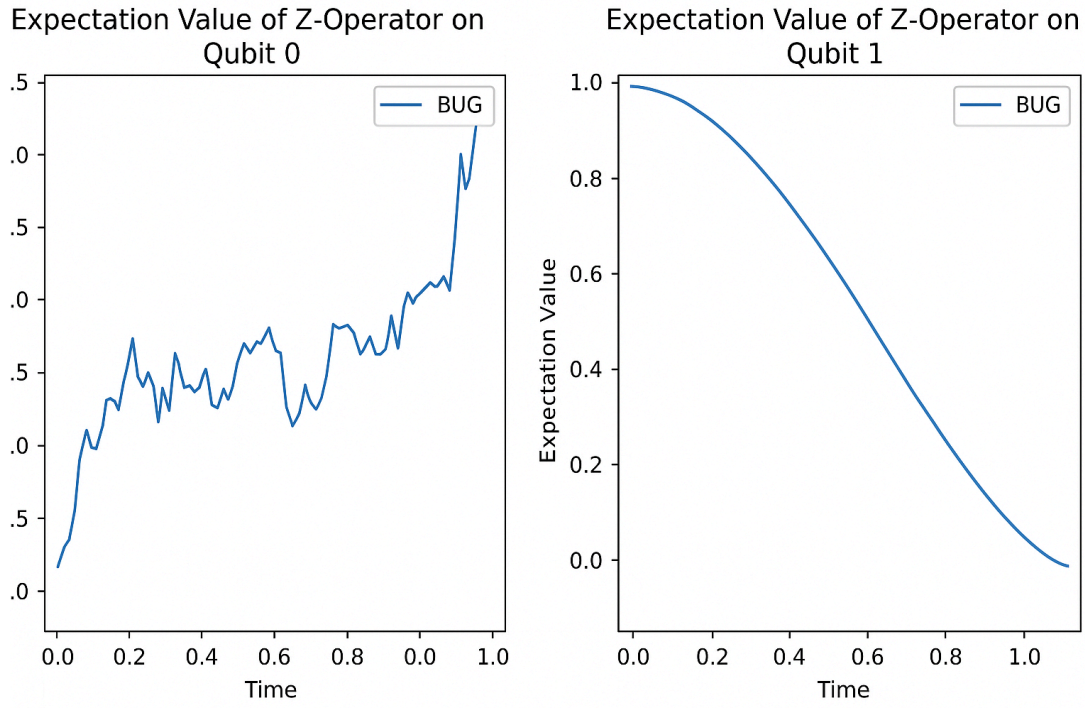

Figure 13: Plot of local magnetization change over time for the CNOT gate.

4.4 Bell State Circuit

In this section, we analyze the result of simulating a Bell state preparation circuit using our tree tensor network framework. The goal of this circuit is to create an entangled quantum state starting from a simple, separable input. We begin with both qubits initialized in the computational basis state $|00\rangle$. The first operation applied to the system is a Hadamard gate on the first qubit. This gate transforms the basis state $|0\rangle$ into a superposition of $|0\rangle$ and $|1\rangle$, specifically into:

$$\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \tag{11}$$

As a result, the total two-qubit system after this operation becomes:

$$\frac{1}{\sqrt{2}}(|00\rangle + |10\rangle) \tag{12}$$

At this point, the system is in a product state, qubit 0 is in a superposition, while qubit 1 remains in the $|0\rangle$ state.

The second step of the circuit involves the application of a CNOT gate, where qubit 0 serves as the control and qubit 1 as the target. The action of the CNOT gate is to flip

the state of the target qubit if and only if the control qubit is in the state $|1\rangle$. Applying this operation to the current state results in the transformation:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{13}$$

This new state is one of the four maximally entangled Bell states and demonstrates perfect quantum correlations between the two qubits. It signifies that a measurement of the first qubit will immediately determine the outcome of a measurement on the second qubit, despite no local information being stored in either qubit individually.

To validate this result, we extract and examine the final statevector produced by our TTN-based simulation. The numerical output is given by the following complex amplitude array:

$$\begin{pmatrix} 7.96326711e-04 - 7.07106557e-01j \\ 1.57198917e-15 - 1.24215148e-15j \\ 6.50761016e-03 - 5.98956085e-05j \\ -6.50761016e-03 - 7.07046661e-01j \end{pmatrix} \tag{14}$$

This array represents the coefficients α_{00}, α_{01}, α_{10}, and α_{11} in the general quantum state $|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle$. Upon close inspection, we observe that the dominant non-zero entries are those corresponding to the $|00\rangle$ and $|11\rangle$ components. The amplitude for $|00\rangle$ is approximately -0.7071, and similarly, the amplitude for $|11\rangle$ is approximately -0.7071, both of which are consistent with the expected values of $\frac{1}{\sqrt{2}}$ for a standard Bell state, up to a global phase. The remaining components, those for $|01\rangle$ and $|10\rangle$, have amplitudes that are several orders of magnitude smaller (on the order of 10^{-3} to 10^{-15}), which we attribute to numerical artifacts arising from floating-point precision and tensor contractions in the simulation.

This agreement between the theoretical Bell state and the simulated state confirms the correctness of the gate implementations, the state evolution logic, and the internal tensor network structure. It also illustrates the simulator's ability to capture entanglement, a fundamental aspect of quantum computation with high fidelity. The structure of the output vector demonstrates that entanglement was successfully generated and preserved, and the negligible amplitudes of unwanted basis states provide additional evidence of the precision of the simulation.

Overall, this result provides a concrete benchmark for both the correctness and the stability of our tree tensor network framework when applied to small entangling circuits

such as the Bell state preparation. It also showcases how even basic circuits can serve as meaningful validation tools for more complex quantum simulations.

4.5 Comparing Results

The `compare_results` method is used to compare how the `apply_gate` operates under different final time for a gate. The simulation was run for several different final times: $p\frac{i}{8}$, $p\frac{i}{4}$, $p\frac{i}{2}$, and pi. The X gate induces a rotation around the X-axis of the Bloch sphere, and each final time corresponds to a different angle of this rotation. The resulting quantum state for each case is as follows:

- Final time $p\frac{i}{8}$: $[0.92106099 + 0i, 0 - 0.38941834i]$
- Final time $p\frac{i}{4}$: $[0.70384532 + 0i, 0 - 0.71035327i]$
- Final time $p\frac{i}{2}$: $[0.00079633 + 0i, 0 - 0.99999968i]$
- Final time pi: $[-0.99996466 + 0i, 0 + 0.00840725i]$

At $p\frac{i}{8}$, the qubit has only slightly evolved from $|0\rangle$, with a small imaginary component appearing in the amplitude of $|1\rangle$, indicating the beginning of the rotation.

At $p\frac{i}{4}$, the state is closer to an equal superposition of $|0\rangle$ and $|1\rangle$, reflecting a rotation toward the equator of the Bloch sphere.

At $p\frac{i}{2}$, the qubit has rotated approximately halfway, and the state is now nearly entirely $|1\rangle$, with the amplitude of $|0\rangle$ approaching zero.

At pi, the state has undergone a full half-rotation, returning close to $|0\rangle$ but with a global phase of -1. This is evident from the negative real part of the $|0\rangle$ amplitude and the near-zero amplitude of $|1\rangle$.

These results align with the expected unitary evolution generated by the X Hamiltonian, and confirm that the simulation correctly models continuous quantum state transformation under time evolution.

4.6 Shor Code

One of the first and most basic quantum error-correcting codes is the Shor code. By encoding a single logical qubit into a state of nine physical qubits, it is intended to guard against both bit-flip and phase-flip problems. The Shor code is constructed using two levels of encoding: a regular three-qubit bit-flip code is applied to each of the generated qubits in the second layer, while a three-qubit repetition code in the Hadamard-transformed basis is used in the first layer to guard against phase-flip mistakes. This nested structure enables the code to correct any arbitrary single-qubit error.

A Hadamard gate is applied to the logical qubit first in the encoding procedure, and then two controlled-NOT gates are applied to divide the state among the three qubits. A second set of controlled-NOT gates is then used to further encode each of these three qubits, creating a total of nine qubits that together form a single logical qubit. During the error correction phase, some multi-qubit gates, such as Toffoli (CCNOT) gates, are commonly used to identify and fix faults.

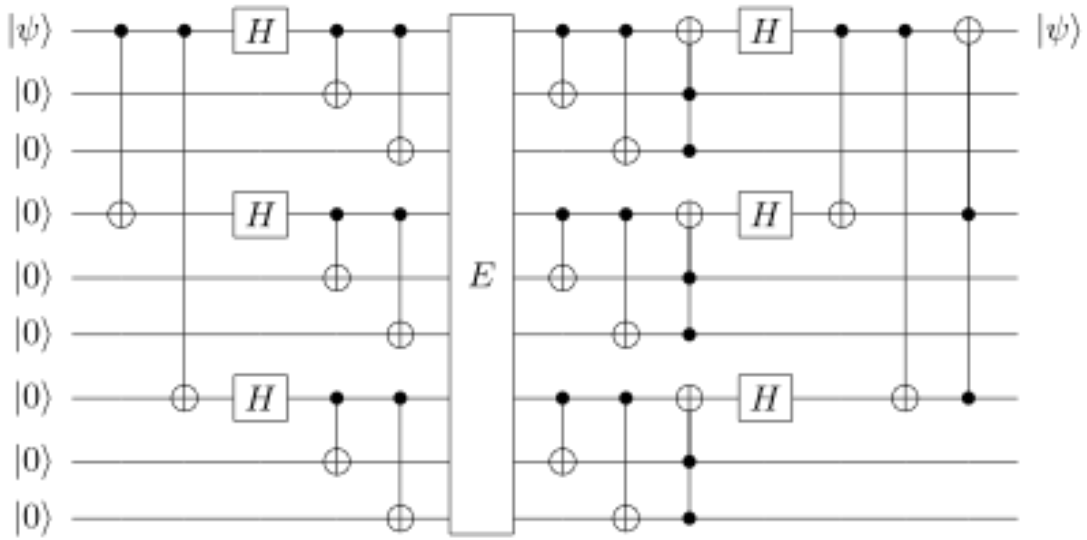

Figure 14: Shor Code. Source: Wikipedia

In our simulation, we implement the encoding and partial correction logic of the Shor code using our tree tensor network (TTN) framework. However, we omit the application of the CCNOT (Toffoli) gates, which are traditionally used for syndrome measurement and conditional correction, because multi-controlled gates are not currently implemented in our framework. Despite this limitation, the encoded state structure and bit-flip protection layers are fully constructed and preserved, allowing us to analyze the entanglement structure and simulate partial error correction behavior. To demonstrate scalability, we apply this procedure twice, resulting in the encoding of two logical qubits across a total of 18 physical qubits. This implementation serves as a foundation for future extensions of the TTN framework to support more advanced quantum error correction protocols.

```
gate_sequence = [
    # First Shor block (qubits 0-8)
    ("H", "qubit0", {}),
    ("CNOT", "qubit0", "qubit1", {}),
    ("CNOT", "qubit0", "qubit2", {}),
    ("H", "qubit0", {}),
```

```
    ("H", "qubit1", {}),
    ("H", "qubit2", {}),
    ("CNOT", "qubit0", "qubit3", {}),
    ("CNOT", "qubit1", "qubit4", {}),
    ("CNOT", "qubit2", "qubit5", {}),
    ("CNOT", "qubit3", "qubit6", {}),
    ("CNOT", "qubit4", "qubit7", {}),
    ("CNOT", "qubit5", "qubit8", {}),
    ("H", "qubit3", {}),
    ("H", "qubit4", {}),
    ("H", "qubit5", {}),
    # Second Shor block (qubits 9-17)
    ("H", "qubit9", {}),
    ("CNOT", "qubit9", "qubit10", {}),
    ("CNOT", "qubit9", "qubit11", {}),
    ("H", "qubit9", {}),
    ("H", "qubit10", {}),
    ("H", "qubit11", {}),
    ("CNOT", "qubit9", "qubit12", {}),
    ("CNOT", "qubit10", "qubit13", {}),
    ("CNOT", "qubit11", "qubit14", {}),
    ("CNOT", "qubit12", "qubit15", {}),
    ("CNOT", "qubit13", "qubit16", {}),
    ("CNOT", "qubit14", "qubit17", {}),
    ("H", "qubit12", {}),
    ("H", "qubit13", {}),
    ("H", "qubit14", {}),
]
```

The gate_sequence shown above is the structured input provided to our simulation framework. It is defined as a Python array, where each element is a tuple representing one quantum gate operation. A dictionary of optional configuration parameters, the qubit or qubits the gate applies to (for single- or two-qubit gates), and the gate's name as a string (e.g., "H", "CNOT") make up each tuple. Values like `time_step_size`, `final_time`, and a configuration object that regulates the application of the gate can all be found in this dictionary. If the user does not provide a configuration, the simulation automatically applies a predefined set of default parameters hardcoded within the framework.

When necessary, this style allows for precise control over gate application while providing a flexible and legible means of defining quantum circuits at a high level. Two iterations of the Shor encoding block, each working on nine qubits for a total of eighteen qubits, are present in the given sequence. Using Hadamard gates, CNOT gates to entangle qubits, and additional Hadamard gates at particular locations to get ready for error detection, the sequence's structure adheres to the logic of the Shor code.

As the algorithm iterates through the gate_sequence, the tree tensor network state (TTNS) is updated incrementally. Each gate in the sequence is applied to the TTNS using the `apply_gate` method of the corresponding gate class. For single-qubit gates like “H”, the method updates the tensor of the targeted node directly. For two-qubit gates like “CNOT” or “SWAP”, the TTNS is modified by performing the appropriate contraction and recompression steps, updating the tensors of both involved nodes and any connecting edges. This dynamic update process ensures that after each gate application, the TTNS reflects the new quantum state, while maintaining an efficient representation through the tree structure.

By the end of the sequence, the TTNS contains a full representation of the evolved quantum state after all specified gates have been applied in order. This approach provides a scalable way to simulate larger systems, observe intermediate states if needed, and extend the circuit logic by simply appending new elements to the `gate_sequence`.

5 Future Work

While the current implementation demonstrates the successful simulation of quantum gates and circuits using a tree tensor network framework, there remain several directions for future development and enhancement. A primary objective is the extension of the gate set supported by the framework. In particular, the implementation of multi-controlled gates such as the CCNOT (Toffoli) gate is essential for completing the full logic of quantum error correction protocols like the Shor code. Currently, the absence of this gate limits our ability to perform syndrome-based error correction, although the encoding steps have been fully realized.

In addition to expanding the gate set, further validation of existing implementations is required. Specifically, we have already developed controlled shift and phase shift gates, but their integration into complex circuits has yet to be thoroughly tested. Ensuring their correctness is essential, especially for simulating algorithms and protocols that rely heavily on controlled operations and precise phase manipulation.

Another critical area for future work is the development of a comprehensive testing framework. While the apply_gate method is the core interface for modifying quantum states, we currently lack automated verification of its correctness across various gates and configurations. Implementing unit tests that evaluate the output of this method against known analytical results or alternative simulation backends would provide confidence in the simulator's stability and accuracy. Such tests can include gate-specific checks, invariance properties, and expected state transitions for standard circuits.

Incorporating these tests into a continuous integration pipeline will allow us to detect regressions and inconsistencies early in the development cycle. Moreover, a robust testing suite will be indispensable for future contributions and feature expansions, as it ensures that new functionalities do not compromise existing components. Altogether, these extensions and verifications will strengthen the simulator's reliability and prepare it for simulating larger and more complex quantum systems.

6 Conclusion

This thesis presented the design and implementation of a quantum gate simulation framework based on tree tensor networks (TTNs), with the goal of efficiently modeling quantum circuits and exploring gate behavior through structured, low-complexity representations of quantum states. By defining a modular architecture centered around an abstract QuantumGate class and its concrete implementations, we were able to simulate a wide variety of single- and two-qubit gates, including Pauli gates, the Hadamard gate, phase shifts, CNOT, SWAP, and controlled-phase gates.

The correctness of these implementations was verified through dynamic visualizations of local observables, such as the Z-operator expectation values. These results not only aligned with theoretical predictions but also offered intuitive insight into how quantum states evolve under gate applications in a TTN framework. The simulation of key entangling operations, including the CNOT gate and the preparation of Bell states, further confirmed the ability of our system to accurately model entangled dynamics.

A notable application of the framework was the partial implementation of the Shor code, a foundational quantum error correction protocol. While the full syndrome-based correction process could not be executed due to the absence of the CCNOT gate in the current system, the encoding steps were fully constructed and repeated to cover 18 physical qubits, demonstrating the scalability and structure of logical encoding within the TTN model.

In addition, controlled shift and phase shift gates were implemented, laying the groundwork for further algorithmic extensions, though they require formal validation in complete circuits. The thesis also recognized the need for a robust testing infrastructure to ensure correctness and regression safety across future updates, particularly in the apply_gate method which forms the core of the simulation engine.

In summary, this work provides a flexible and extensible platform for simulating quantum gates and small circuits using tree tensor networks. It bridges theoretical models with practical implementation and paves the way for future developments in quantum simulation, error correction, and quantum software tooling. With further extensions, such as multi-controlled gate support and comprehensive testing, the framework will be well-positioned to model more complex quantum systems and contribute to the advancement of quantum computing research.

Bibliography

[1] Nielsen, M. A., and Chuang, I. L., 2010, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, Cambridge University Press.

[2] Benenti, G., Casati, G., and Strini, G., 2004, *Principles of Quantum Computation and Information*, World Scientific. [Online]. Available: https://books.google.de/books?id=5gmH-DLuIlMC.

[3] Kaye, P., Laflamme, R., and Mosca, M., 2007, *An Introduction to Quantum Computing*, Oxford University Press, Inc., USA.

[4] Wong, T., 2022, *Introduction to Classical and Quantum Computing*, Rooted Grove. [Online]. Available: https://books.google.de/books?id=M3jqzgEACAAJ.

[5] Biamonte, J., and Bergholm, V., 2017, "Tensor Networks in a Nutshell," arXiv. [Online]. Available: https://arxiv.org/pdf/1708.00006.

[6] Szalay, S., Pfeffer, M., Murg, V., Barcza, G., Verstraete, F., Schneider, R., and Legeza, Ö., 2015, "Tensor Product Methods and Entanglement Optimization for Ab Initio Quantum Chemistry," International Journal of Quantum Chemistry, **115**(19), pp. 1342–1391. https://doi.org/10.1002/qua.24898.

[7] Wood, C. J., Biamonte, J. D., and Cory, D. G., 2015, "Tensor Networks and Graphical Calculus for Open Quantum Systems," arXiv. https://doi.org/10.48550/arXiv.1111.6950.

[8] Jaschke, D., Montangero, S., and Carr, L. D., 2018, "One-Dimensional Many-Body Entangled Open Quantum Systems with Tensor Network Methods," Quantum Science and Technology, **4**(1). https://doi.org/10.1088/2058-9565/aae724.

[9] Strathearn, A., Kirton, P., Kilda, D., Keeling, J., and Lovett, B. W., 2018, "Efficient Non-Markovian Quantum Dynamics Using Time-Evolving Matrix Product Operators," Nature Communications, **9**(1). https://doi.org/10.1038/s41467-018-05617-3.

[10] Schollwöck, U., 2011, "The Density-Matrix Renormalization Group in the Age of Matrix Product States," Annals of Physics, **326**(1), pp. 96–192. https://doi.org/10.1016/j.aop.2010.09.012.

[11] Bañuls, M. C., 2023, "Tensor Network Algorithms: A Route Map," Annual Review of Condensed Matter Physics, **14**(1). https://doi.org/10.1146/annurev-conmatphys-040721-022705.

[12] Jahn, A., and Eisert, J., 2021, "Holographic Tensor Network Models and Quantum Error Correction: A Topical Review," Quantum Science and Technology, **6**(3). https://doi.org/10.1088/2058-9565/ac0293.

[13] Melnikov, A. A., Termanova, A. A., Dolgov, S. V., Neukart, F., and Perelshtein, M. R., 2023, "Quantum State Preparation Using Tensor Networks," Quantum Science and Technology, **8**(3). https://doi.org/10.1088/2058-9565/acd9e7.

[14] Patra, S., Jahromi, S. S., Singh, S., and Orús, R., 2024, "Efficient Tensor Network Simulation of Ibm's Largest Quantum Processors," Physical Review Research, **6**(1). https://doi.org/10.1103/PhysRevResearch.6.013326.

[15] Rieser, H.-M., Köster, F., and Raulf, A. P., 2023, "Tensor Networks for Quantum Machine Learning," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, **479**(2275). https://doi.org/10.1098/rspa.2023.0218.

[16] Ji, Y., Wang, Q., Li, X., and Liu, J., 2019, "A Survey on Tensor Techniques and Applications in Machine Learning," IEEE Access, **7**, pp. 162950–162990. https://doi.org/10.1109/ACCESS.2019.2949814.

[17] Panagakis, Y., Kossaifi, J., Chrysos, G. G., Oldfield, J., Nicolaou, M. A., Anandkumar, A., and Zafeiriou, S., 2021, "Tensor Methods in Computer Vision and Deep Learning," Proceedings of the IEEE, **109**(5), pp. 863–890. https://doi.org/10.1109/JPROC.2021.3074329.

[18] Sengupta, R., Adhikary, S., Oseledets, I., and Biamonte, J., 2022, "Tensor Networks in Machine Learning," Journal of the European Mathematical Society (JEMS), **126**. https://doi.org/10.4171/mag/101.

[19] Stoudenmire, E., and Schwab, D. J., 2016, "Supervised Learning with Tensor Networks," *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, eds.. [Online]. Available: https://papers.nips.cc/paper_files/paper/2016/file/53c04118df112c13a8c34b38343b9c10-Paper.pdf.

[20] Silvi, P., Tschirsich, F., Gerster, M., Jünemann, J., Jaschke, D., Rizzi, M., and Montangero, S., 2019, "The Tensor Networks Anthology: Simulation Techniques for Many-Body Quantum Lattice Systems," SciPost Phys. Lect. Notes, p. 8. https://doi.org/10.21468/SciPostPhysLectNotes.8.

[21] Bañuls, M. C., 2020, "Tensor Network Algorithms and Applications," Max Planck Institute. [Online]. Available: https://www.mpq.mpg.de/6944607/tensor-networks.

[22] Cichocki, A., Phan, A.-H., and Zhao, Q., 2017, "Tensor Networks for Dimensionality Reduction and Large-Scale Optimizations. Part 2 Applications and Future Perspectives," arXiv. [Online]. Available: https://arxiv.org/abs/1708.09165.

[23] Glasser, I., Pancotti, N., and Cirac, J. I., 2018, "From Probabilistic Graphical Models to Generalized Tensor Networks for Supervised Learning," arXiv. [Online]. Available: https://arxiv.org/abs/1806.05964.

[24] Roberts, C., Milsted, A., Ganahl, M., and others, 2019, "Tensornetwork: A Library for Physics and Machine Learning," arXiv. [Online]. Available: https://arxiv.org/abs/1905.01330.

[25] Xiang, T., 2013, "Tree Tensor Network States," *Density Matrix and Tensor Network Renormalization*, Cambridge University Press, p. 21. [Online]. Available: https://www.cambridge.org/core/books/density-matrix-and-tensor-network-renormalization/tree-tensor-network-states/6CB6537972F91F2C23497731868DDDAC.

[26] Nomura, Y., and Imada, M., 2023, "Automatic Structural Optimization of Tree Tensor Networks," Physical Review Research, **5**, p. 13031. https://doi.org/10.1103/PhysRevResearch.5.013031.

[27] Su, Y., Yuan, X., and Zeng, B., 2021, *Tensor Network Techniques for Quantum Computation*, Open Access Publishing in the Quantum Sciences. [Online]. Available: https://library.oapen.org/handle/20.500.12657/96026.

[28] Seitz, P., Medina, I., Cruz, E., Huang, Q., and Mendl, C. B., 2023, "Simulating Quantum Circuits Using Tree Tensor Networks," Quantum, **7**, p. 964. https://doi.org/10.22331/q-2023-03-30-964.

[29] Muñoz-Coreas, E., and Thapliyal, H., 2022, "Everything You Always Wanted to Know About Quantum Circuits," arXiv preprint. [Online]. Available: https://arxiv.org/abs/2208.11725.

[30] Fisher, M. P. A., Khemani, V., Nahum, A., and Vijay, S., 2022, "Random Quantum Circuits," arXiv preprint. [Online]. Available: https://arxiv.org/abs/2207.14280.

List of Acronyms

List of Figures