DEPARTMENT OF PHYSICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Quantum Science and Technology

A Stochastic Tensor Network Method for Simulating Open Quantum Systems

Aaron Sander

DEPARTMENT OF PHYSICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Quantum Science and Technology

A Stochastic Tensor Network Method for Simulating Open Quantum Systems

Eine stochastische Tensornetzwerkmethode zur Simulation von offenen Quantensystemen

Author:	Aaron Sander
First Referee:	Prof. Dr. Christian Mendl
Second Referee:	Prof. Dr. Stefan Filipp
Submission Date:	31. October 2022

Erklärung

Ich versichere hiermit, dass ich die von mir eingereichte Abschlussarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

München, 31.10.2022, Aaron Sander

Acknowledgments

I would like to first thank my supervisor Prof. Dr. Christian Mendl for his support and guidance throughout this thesis. I would also like to thank Richard Milbradt, Dr. Kévin Hémery, and Federico Roy for their help during tricky parts of this work as well as their experience and opinions on properly representing the results. Finally, I would like to thank Emiliano Godínez, Emily Haworth, Filippo Romano, and Varun Seshradi for their exceptional proofreading skills.

Code and Figures

All code in this work was written independently using standard Python libraries and does not include any outside physics or tensor network packages. All figures were created independently for this work in Inkscape.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Classical simulations are currently one of the most useful tools for comparing quantum theory with experimental results. This allows us to gain a deeper understanding of quantum systems as well as build and scale more stable, reliable quantum computers. Quantum systems are notoriously difficult to simulate classically due to the exponential growth of values needed to represent larger systems. This growth leads to both increasing memory requirements to store exponentially many complex values and longer computational time in order to perform operations on these systems such as time evolution. Tensor networks, particularly Matrix Product States (MPS), are famously at the forefront of quantum simulation as they allow us to reduce the necessary memory usage and computational time. Rather than represent quantum systems i.e. quantum states, as vectors that grow exponentially with system size, MPSs allow us to split it into a tensor train with each tensor representing a local system. This allows us to perform local operations rather than global operations as well as store the quantum system with fewer values. The caveat is that the bond dimension grows between these tensors as the system gets larger. However, the ability to truncate this to much lower dimensions has been proven to be a successful approximation method that enables MPSs to be reliable as a simulation method.

A particularly interesting and useful process to simulate is open quantum systems in which a quantum system interacts with its environment and gives us a more realistic description of how it evolves. The environment can cause effects such as relaxation, dephasing, or thermal excitations in the quantum system that can make the quantum computing architectures and calculations unreliable. In order to build better systems and algorithms, we need to understand these noise processes in order to mitigate the error that they cause. The first step in doing so is to classically simulate these noise models.

In this thesis, the goal is to combine the power of tensor networks with the usefulness of open quantum systems in order to extend the simulation capabilities. It focuses primarily on a stochastic method known as the quantum jump method which already allows us to represent noise processes as a vector rather than a full density matrix. The main work in this thesis has been to convert this quantum jump method into a tensor network algorithm, benchmark its capabilities, and apply it to some toy model applications. This algorithm is shown to be able to extend the simulation capabilities for open quantum systems to larger system sizes than are currently possible. Doing so gives us a fuller understanding of the effect of noise on quantum systems such that we can mitigate error in the building and scaling of quantum technologies.

2 Fundamentals

2.1 Open Quantum Systems

In order to realistically understand quantum systems, various environmental effects must be taken into account that cause deviation between theoretical models presented by closed systems and experimental results. The effects can be represented by considering not only a quantum system in the calculations, but also an environmental system containing it. The interactions or noise processes between these two systems lead to a noisy version of the original quantum system which can align better with experimental results. This in turn gives us a starting point for improving the design of the systems and the mitigation of error caused by environmental noise.

2.1.1 Noise Processes

Noise is a primary source of error in quantum computing calculations due to the instability of both the qubits and the gate application during algorithms. The ability to properly simulate these processes would open the door to finding ways to mitigate them or utilize them in order to create more stable, reliable quantum hardware and quantum algorithms. There are several basic noise processes such as relaxation, dephasing, and thermal excitations. These are primarily the ones considered in this thesis as they are generally relevant for all quantum computing architectures. Specific platforms can have additional noise processes such as charge noise and flux noise in superconducting qubits [9], but they are not directly taken into account in this thesis. Most of the analysis is not concerned with the specific noise process itself, but rather its strength relative to other parameters of the system.

Each physical noise process is mathematically represented by a jump operator as well as an associated coupling factor related to how often it occurs or how strong its effect is on the system. These coupling factors are typically determined by experimentally measuring certain timespans in various systems, such as relaxation or dephasing time, so there can be a variety of possible values. The strength and calculation of the coupling factors could be different based on the application. They are typically proportional to the inverse of the timescale in which they occur for example $\gamma_{\text{relaxation}} = \frac{1}{T_1}$. This section will introduce what these noise processes look like mathematically and the implementation of them in a simulation will be shown later.

A quantum jump can be defined as a sudden transition of a system between one state and another. This is a drastic change in the system which happens instantaneously, distinguishing it from a classical process. As with most fundamental quantum physics, the physicality and philosophy behind this occurring is up for debate, however it gives relevant results in practice.

Possible noise processes are represented by the jump operators and potential coupling factors found in Table 2.1.

Relaxation is represented by a de-excitation operator performing a jump from a higher state to a lower state

$$\begin{aligned}\sigma_- |0\rangle &= 0 \\ \sigma_- |1\rangle &= |0\rangle\end{aligned}$$

such that in matrix form $\sigma_- = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$ or generalized to a d-level system

$$\begin{aligned}\sigma_- |0\rangle &= 0 \\ \sigma_- |k\rangle &= |k-1\rangle\end{aligned}$$

where $\{k \in \mathbb{Z} : 0 \leq k \leq d-1\}$ with matrix elements

$$\sigma_-^{jk} = \begin{cases} 1 & k - j = 1 \\ 0 & \text{else} \end{cases}$$

The coupling factor is related to the longitudinal decay time T_1.

Likewise, thermal excitations are represented by an excitation operator jumping from a lower to a higher state i.e.

$$\begin{aligned}\sigma_- |0\rangle &= |1\rangle \\ \sigma_- |1\rangle &= 0\end{aligned}$$

such that in matrix form $\sigma_- = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}$ or generalized to a d-level system

$$\begin{aligned}\sigma_- |d\rangle &= 0 \\ \sigma_- |k\rangle &= |k+1\rangle\end{aligned}$$

where $\{k \in \mathbb{Z} : 0 \leq k \leq d-1\}$ with matrix elements

$$\sigma_-^{jk} = \begin{cases} 1 & j - k = 1 \\ 0 & \text{else} \end{cases}$$

with coupling factor determined by the temperature T of the environment.

Dephasing is described by a rotation around the z-axis with the Pauli matrix σ_z. This was then generalized to d-levels using the generalized spin matrices of the special unitary group $SU(n)$ i.e. a symmetric diagonal with descending odd integers. The coupling factor is calculated from the effective transversal decay time T_2^*. One could also consider idle cross-talk which applies dephasing to multiple qudits simultaneously, potentially with an exponentially decaying coupling factor based on distance.

Noise Process	Jump Operator	Coupling Factor
Relaxation	σ_-	$\gamma_{\text{relaxation}}$
Dephasing	σ_z	$\gamma_{\text{dephasing}}$
Thermal	σ_+	γ_{thermal}
Cross-Talk	$\sigma_z^{[i]}\sigma_z^{[j]}$	γ_{ij}

Table 2.1: Potenial noise processes and their corresponding jump operator

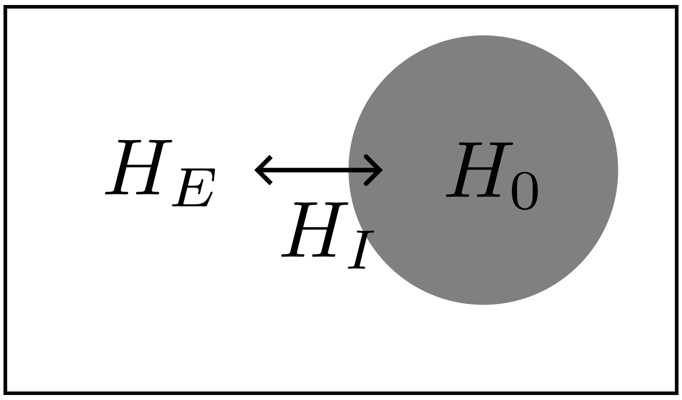


Figure 2.1: Visualization of a quantum system embedded in its environment with interaction

2.1.2 Derivation of the Lindblad Master Equation

The dynamics of open quantum systems i.e. the interactions between a quantum system and its environment are often described by the Lindblad master equation. This describes the application of non-unitary, irreversible dynamic processes to create a Markov chain describing the motion of the system. This means that the evolution of a system is based purely on its state at that specific time and thus has no memory of its previous states. The Lindblad master equation is thus a key part of understanding noise processes such as relaxation and dephasing which are utilized later in this thesis [2] [13].

This section is used as background in order to understand the basis of solving the dynamics of open quantum systems and what assumptions and considerations are taken into account when forming the master equation. In order to derive this, consider a total Hilbert space $\mathcal{H}_T = \mathcal{H} \otimes \mathcal{H}_E$ which contains the Hilbert space of the quantum system $\mathcal{H}$ as well as the Hilbert space of the system's environment $\mathcal{H}_E$. The evolution of this total system is given by the von Neumann equation which is the quantum equivalent of the classical Liouville equation

$$\dot{\rho}_T = -\frac{i}{\hbar}[H_T, \rho_T(t)] \tag{2.1}$$

where the density matrix of the quantum system is given by tracing out the environment such that $\rho(t) = \mathrm{Tr}_E[\rho_T]$. It is then possible to define a total Hamiltonian describing the energy of the total system

$$H = H_0 \otimes \mathbb{1} + \mathbb{1} \otimes H_E + \alpha H_I \tag{2.2}$$

with H_0 describing the quantum system, H_E describing the environment, and H_I describing the interactions between the system and its environment with strength $\alpha \ll 1$. The interaction term can further be decomposed into operators acting on the system $S_i \in B(\mathcal{H})$ and operators acting on the environment $E_i \in B(\mathcal{H}_E)$ such that

$$H_I = \sum_i S_i \otimes E_i \tag{2.3}$$

where $B(\mathcal{H})$ represents the space of bounded operators on the Hilbert space $B : \mathcal{H} \to \mathcal{H}$ and $B(\mathcal{H}_E)$ on the environment $B : \mathcal{H}_E \to \mathcal{H}_E$.

The dynamics can then be analyzed in the interaction picture wherein density matrices evolve with time due to the interaction Hamiltonian and operators evolve with the system and environment Hamiltonians. An arbitrary operator $\hat{O} \in B(\mathcal{H}_T)$ is time-evolved according to

$$\hat{O}(t) = e^{i(H_0+H_E)t}\hat{O}e^{-i(H_0+H_E)t} \tag{2.4}$$

while the time-evolution of the density matrix is given by

$$\dot{\rho}_T = -\frac{i}{\hbar}\alpha[H_I(t), \rho_T(t)] \tag{2.5}$$

Integrating this to solve for $\rho_T(t)$,

$$\rho_T(t) = \rho_T(0) - \frac{i}{\hbar}\alpha \int_0^t d\tau [H_I(\tau), \rho_T(\tau)] \tag{2.6}$$

This equation is difficult to work with because it depends on the density matrix at all previous times τ but this can be avoided by substituting it into Eq. (2.5)

$$\dot{\rho}_T = -\frac{i}{\hbar}\alpha[H_I(t), \rho_T(0)] - \frac{\alpha^2}{\hbar^2}\int_0^t d\tau \Big[H_I(t), [H_I(\tau), \rho_T(\tau)]\Big] \tag{2.7}$$

Repeating the process once more for the $\rho_T(\tau)$ such that the equation is no longer dependent on previous times τ

$$\dot{\rho}_T = -\frac{i}{\hbar}\alpha[H_I(t), \rho_T(0)] - \frac{\alpha^2}{\hbar^2}\int_0^t d\tau \Big[H_I(t), [H_I(\tau), \rho_T(t)]\Big] + \mathcal{O}(\alpha^3) \tag{2.8}$$

The equation of motion needs to be in terms of the system density matrix ρ rather than the total density matrix ρ_T such that tracing over the environment results in

$$\dot{\rho} = \mathrm{Tr}_E[\dot{\rho}_T] = -\frac{i}{\hbar}\alpha \mathrm{Tr}_E[H_I(t), \rho_T(0)] - \frac{\alpha^2}{\hbar^2}\int_0^t d\tau \mathrm{Tr}_E\Big[H_I(t), [H_I(\tau), \rho_T(t)]\Big] \tag{2.9}$$

This has not yet removed the dependency on the total density matrix $\rho_T(t)$ and therefore $\rho(t)$ is also not yet directly solvable. Therefore, the following assumptions need to be made. The initial state at $t = 0$ is assumed to be a separable state in which there are no correlations between the system and the environment i.e. $\rho_T(0) = \rho(0) \otimes \rho_E(0)$. This means no previous interactions have occurred or if they have, they have been very short-lived and negligible. The initial state of the environment is also assumed to be a thermal state described by

$$\rho_E(0) = \frac{e^{\frac{-H_E}{k_B T}}}{\mathrm{Tr}\left[e^{\frac{-H_E}{k_B T}}\right]} \tag{2.10}$$

with Boltzmann constant k_B and temperature T. Eq. (2.9) can now be simplified by applying these assumptions and the definition of H_I to the first commutator term

$$\begin{aligned} \mathrm{Tr}_E[H_I(t), \rho_T(0)] &= \mathrm{Tr}_E\Big[\sum_i S_i(t) \otimes E_i(t), \rho_T(0)\Big] \\ &= \mathrm{Tr}_E\Big[\sum_i S_i(t) \otimes E_i(t)\rho_T(0) - \rho_T(0) S_i(t) \otimes E_i(t)\Big] \\ &= \sum_i \Big(S_i(t)\rho(0)\mathrm{Tr}_E\Big[E_i(t)\rho_E(0)\Big] - \rho(0)S_i(t)\mathrm{Tr}_E\Big[\rho_E(0)E_i(t)\Big]\Big) \end{aligned} \tag{2.11}$$

If no energy has yet been lost to the environment at $t = 0$, then $\langle E_i \rangle = \mathrm{Tr}[E_i \rho_E(0)] = 0 \ \forall i$. This simplifies the equation of motion to

$$\dot{\rho} = -\frac{\alpha^2}{\hbar^2} \int_0^t d\tau \mathrm{Tr}_E\Big[H_I(t), [H_I(\tau), \rho_T(t)]\Big] \tag{2.12}$$

Due to the fact that this derivation is in the weak-coupling regime ($\alpha \ll 1$), the system and environment remain non-correlated during the entire time evolution. This assumes that the timescales of correlations and relaxation of the environment are much smaller than the typical timescale of the system. This results in $\rho_T(t) = \rho(t) \otimes \rho_E(0)$ which when substituted into the equation of motion

$$\dot{\rho} = -\frac{\alpha^2}{\hbar^2} \int_0^t d\tau \mathrm{Tr}_E\Big[H_I(t), [H_I(\tau), \rho(t) \otimes \rho_E(0)]\Big] \tag{2.13}$$

This depends on the initial state and is therefore non-Markovian. In order to obtain a Markovian equation that is independent of previous states, the integration limit $t \to \infty$ is extended and a change of variables is performed where $\tau \to t - \tau$

$$\dot{\rho} = -\frac{\alpha^2}{\hbar^2} \int_0^\infty d\tau \mathrm{Tr}_E\Big[H_I(t), [H_I(t-\tau), \rho(t) \otimes \rho_E(0)]\Big] \tag{2.14}$$

Lastly, the commutators are expanded

$$\begin{aligned} \dot{\rho} = -\frac{\alpha^2}{\hbar^2}\mathrm{Tr}\Big[&\int_0^\infty d\tau H_I(t)H_I(t-\tau)\rho(t) \otimes \rho_E(0) - \int_0^\infty d\tau H_I(t)\rho(t) \otimes \rho_E(0)H_I(t-\tau) \\ &- \int_0^\infty d\tau H_I(t-\tau)\rho(t) \otimes \rho_E(0)H_I(t) + \int_0^\infty d\tau \rho(t) \otimes \rho_E(0)H_I(t-\tau)H_I(t)\Big] \end{aligned} \tag{2.15}$$

Next consider the spectrum of the superoperator $\tilde{H}A = [H, A] = HA - AH$, $\forall A \in B(\mathcal{H})$. The eigenvectors of the superoperator form a complete basis of the space of bounded operators on the Hilbert space $B(\mathcal{H})$ such that the interaction operators can be expanded in this basis

$$S_i = \sum_{\omega} S_i(\omega) \tag{2.16}$$

where ω are the eigenvalues of the superoperator $\tilde{H}$ and the operators in this basis fulfill the commutator relation $[H, S_i(\omega)] = -\omega S_i(\omega)$ or likewise for the conjugate $[H, S_i^\dagger(\omega)] = \omega S_i^\dagger(\omega)$. The evolution of these operators in the Schrödinger picture is used to transform the interaction Hamiltonian

$$\tilde{H}_I(t) = \sum_{k,\omega} e^{-i\omega t} S_k(\omega) \otimes \tilde{E}_k(t) \tag{2.17}$$

Applying the eigenvalue decomposition such that $H_I(t-\tau)$ is in terms of $S_k(\omega)$ and $H_I(t)$ in terms of $S_k^\dagger(\omega)$ results in

$$\begin{aligned}\dot{\rho}(t) = \sum_{k,\ell,\omega,\omega'} \Big(& e^{i(\omega'-\omega)t}\Gamma_{k\ell}(\omega)\Big[S_\ell(\omega)\rho(t), S_k^\dagger(\omega')\Big] \\ & + e^{-i(\omega'-\omega)t}\Gamma_{k\ell}^*(\omega')\Big[S_\ell(\omega), \rho(t)S_k^\dagger(\omega')\Big]\Big)\end{aligned} \tag{2.18}$$

where the effects of the environment are represented by the factors

$$\Gamma_{k\ell}(\omega) = \int_0^\infty d\tau e^{i\omega\tau}\mathrm{Tr}_E\Big[\tilde{E}_k^\dagger(t)\tilde{E}_\ell(t-\tau)\rho_E(0)\Big] \tag{2.19}$$

As Eq. 2.15 is able to return non-positive density matrices, the rotating wave approximation (RWA) is applied in order to ensure positivity. This is applied by excluding fast oscillating terms $|\omega - \omega'| \gg \alpha^2$ and in the low-coupling regime where $\alpha \to 0$, only resonant terms $\omega = \omega'$ remain. This reduces the equation of motion to

$$\dot{\rho}(t) = \sum_{k,\ell,\omega}\Big(\Gamma_{k\ell}(\omega)\Big[S_\ell(\omega)\rho(t), S_k^\dagger(\omega)\Big] + \Gamma_{k\ell}^*(\omega')\Big[S_\ell(\omega), \rho(t)S_k^\dagger(\omega)\Big]\Big) \tag{2.20}$$

The dynamics are then divided into Hermitian and non-Hermitian parts such that $\Gamma_{kl}(\omega) = \frac{1}{2}\gamma_{kl}(\omega) + i\pi_{kl}$ where

$$\gamma_{k\ell}(\omega) = \Gamma_{k\ell}(\omega) + \Gamma_{k\ell}^*(\omega) = \int_{-\infty}^{\infty} d\tau e^{i\omega\tau}\mathrm{Tr}\Big[E_k^\dagger(\tau)E_\ell\rho_E(0)\Big] \tag{2.21}$$

$$\pi_{k\ell}(\omega) = -\frac{i}{2}(\Gamma_{k\ell}(\omega) + \Gamma_{k\ell}^*(\omega)) \tag{2.22}$$

Returning to the Schrödinger equation, the Hermitian and non-Hermitian parts of the equation of motion can be separated resulting in

$$\begin{aligned}\dot{\rho}(t) = & -\frac{i}{\hbar}[H_0 + H_{LS}, \rho(t)] \\ & + \sum_{k,\ell,\omega}\gamma_{kl}(\omega)\Big(S_\ell(\omega)\rho(t)S_k^\dagger(\omega) - \frac{1}{2}\Big\{S_k^\dagger(\omega)S_\ell(\omega), \rho(t)\Big\}\Big)\end{aligned} \tag{2.23}$$

This leads to a term representing the Lamb Shift which renormalizes the system energy levels due to interactions with the environment such that $H_{LS} = \sum_{k,l,\omega} \pi_{kl}(\omega) S_k^\dagger(\omega) S_\ell(\omega)$.

Some final adjustments can be made to obtain the Lindblad master equation. Since $\gamma_{kl}(\omega)$ is defined as the Fourier transform of a positive function (due to the trace), the summation is diagnonalized. Finally, the Lamb Shift is assumed to be 0. This results in the overall Lindblad equation which will be considered in this thesis with jump operators L_m and coupling factors γ_m

$$\dot{\rho}(t) = -\frac{i}{\hbar}[H_0, \rho(t)] + \sum_m \gamma_m \left(L_m \rho(t) L_m^\dagger - \frac{1}{2}\left\{ L_m^\dagger L_m, \rho(t) \right\} \right) \tag{2.24}$$

2.1.3 Exact Solution - Superoperator Formalism

For a qudit d-level system with L sites, the density operator is the size $d^L \times d^L$ i.e. $\rho \in \mathbb{C}^{d^L \times d^L}$. For systems with more levels or more sites, this becomes computationally expensive and impossible to calculate and forms the basis of the need for simulation methods and approximations. For benchmarking the methods created later in this thesis, it is necessary to solve the master equation exactly and in order to do so, this equation is converted into a superoperator formalism. This formalism is a straight-forward way of performing the calculation and the eigenvalue spectrum of the superoperator itself can contain useful information such as whether the system will decay (i.e. it has negative eigenvalues).

In order to do this, the master equation is brought into a form such that only a simple matrix multiplication is required to solve for the time evolution. Just as for some operator acting on a ket $|\psi'\rangle = \hat{O}|\psi\rangle$ (or matrix acting on a vector), a superoperator similarly acts on a vectorized operator. This means the Lindbladian superoperator is of the dimensions $\hat{\mathcal{L}} \in \mathbb{C}^{d^{2L} \times d^{2L}}$ such that

$$|\dot{\rho}(t)\rangle\rangle = \mathcal{L}\,|\rho\rangle\rangle \tag{2.25}$$

where the double ket denotes a vectorized density operator $|\rho\rangle\rangle \in \mathbb{C}^{d^{2L}}$ of the form

$$\begin{aligned} |\cdot\rangle\rangle : \mathbb{C}^{d^L \times d^L} &\to \mathbb{C}^{d^{2L}} \\ \sum_{i,j} \rho_{ij} |i\rangle\langle j| &\mapsto \sum_{i,j} \rho_{ij} |i\rangle \otimes |j\rangle \end{aligned} \tag{2.26}$$

or explicitly

$$\rho = \begin{pmatrix} \rho_{00} & \cdots & \rho_{0,d^L} \\ \vdots & & \vdots \\ \rho_{d^L,0} & \cdots & \rho_{d^L d^L} \end{pmatrix} \to |\rho\rangle\rangle = \begin{pmatrix} \rho_{00} \\ \rho_{01} \\ \vdots \\ \rho_{d^L,0} \\ \vdots \\ \rho_{d^L,d^L} \end{pmatrix} \tag{2.27}$$

The Liouvillian term of the master equation is $\dot{\rho}(t) = -i[H_0, \rho(t)] = -i(H_0\rho - \rho H_0)$ where the notation of time dependence is dropped for brevity. For the superoperator, all terms need to act on the operator from the left such that it can be converted into matrix multiplication. In order to do this, the terms are rewritten as a left $\mathcal{L}[H_0]$ (not to be confused with the superoperator $\hat{\mathcal{L}}$ itself since this is just for the derivation) and right application $\mathcal{R}[H_0]$ where

$$\mathcal{L}(H_0)\rho = \sum_{i,j} \rho_{ij} H_0 \left|i\right\rangle \left\langle j\right| \tag{2.28}$$

$$\mathcal{R}(H_0)\rho = \sum_{i,j} \rho_{ij} \left|i\right\rangle \left\langle j\right| H_0 \tag{2.29}$$

In order to preserve this form, the vectorized form must continue acting on its row or column element (i.e. $|i\rangle$ and $\langle j|$) where

$$|\mathcal{L}(H_0)\rho\rangle\rangle = \sum_{i,j} \rho_{ij} (H_0 \otimes \mathbb{1}) \left|i\right\rangle \otimes \left|j\right\rangle \tag{2.30}$$

$$|\mathcal{R}(H_0)\rho\rangle\rangle = \sum_{i,j} \rho_{ij} (\mathbb{1} \otimes H_0^T) \left|i\right\rangle \otimes \left|j\right\rangle \tag{2.31}$$

This results in the vectorized Liouvillian term

$$|\dot{\rho}(t)\rangle\rangle = -i[(H_0 \otimes \mathbb{1}) - (\mathbb{1} \otimes H_0^T)] \left|\rho\right\rangle\rangle =: \mathcal{H} \left|\rho\right\rangle\rangle \tag{2.32}$$

Likewise, the terms containing each jump operator need to be vectorized. For a single jump operator L_m this has the following form by expanding the anticommutator

$$\gamma_m \left(L_m \rho L_m^\dagger - \frac{1}{2}\left\{ L_m^\dagger L_m, \rho \right\} \right) = \gamma_m \left(L_m \rho L_m^\dagger - \frac{1}{2}[L_m^\dagger L_m \rho + \rho L_m^\dagger L_m] \right) \tag{2.33}$$

Using the same logic as for the Liouvillian term, this can be expressed as

$$\begin{aligned} |\mathcal{D}_m\rho\rangle\rangle &= \gamma_m \left(L_m \otimes L_m^\dagger \left|\rho\right\rangle\rangle - \frac{1}{2}\left[(L_m^\dagger L_m \otimes \mathbb{1}) \left|\rho\right\rangle\rangle + (\mathbb{1} \otimes L_m^T L_m^*) \left|\rho\right\rangle\rangle \right] \right) \\ &= \frac{\gamma_m}{2} \left(2L_m \otimes L_m^\dagger - L_m^\dagger L_m \otimes \mathbb{1} - \mathbb{1} \otimes L_m^T L_m^* \right) \left|\rho\right\rangle\rangle \end{aligned} \tag{2.34}$$

where $\mathcal{D}_m$ stands for the dissipative term for a single jump operator L_m. Putting these together and summing over all possible jump operators, the Lindbladian superoperator has the following form

$$\begin{aligned} \hat{\mathcal{L}} &= \mathcal{H} + \sum_m \mathcal{D}_m \\ &= -i[(H \otimes \mathbb{1}) - (\mathbb{1} \otimes H)] + \sum_m \frac{\gamma_m}{2} \left(2L_m \otimes L_m^\dagger - L_m^\dagger L_m \otimes \mathbb{1} - \mathbb{1} \otimes L_m^T L_m^* \right) \end{aligned} \tag{2.35}$$

This superoperator can then be used on the vectorized density operator to solve Eq. (2.25).

2.2 Tensor Networks

Knowledge of tensor networks is essential for understanding the methods used and created in this thesis. This section looks at the basic components used in this thesis, first at a very intuitive level and then introduces some of the mathematics required for a deep dive into the algorithms used. For more extensive explanation, a good starting point is "Hand-waving and Interpretive Dance: An Introductory Course on Tensor Networks" [3]. Tensor networks is very diagrammatic which makes it very intuitive physically, however everyone tends to have different preferences for symbols. In this work, a square represents a tensor with no specific form, a right-pointing triangle is the left-canonical form, and a left-pointing triangle is the right-canonical form which will be discussed in depth later.

2.2.1 Matrix Product States and Matrix Product Operators

First it is necessary to introduce the basic elements of tensor networks that will be used extensively throughout this thesis. Matrix Product States (MPS) and Matrix Product Operators (MPO) allow us to better represent many-body systems. Without tensor networks, the state of an L site many-body system is described by a vector with dimension d^L where d is the physical dimension. Operators acting on this are of the size $d^L \times d^L$. This quickly becomes too large to store in memory or to perform calculations on. All operators also need to effectively act globally on the entire vector rather than a single site or locally. However, a tensor network representation of this solves both problems in that the state can be represented as a product of single site, local tensors and operators as a chain of local operations. Tensors themselves are generalizations of vectors (order-1 tensors) and matrices (order-2 tensors) and can be thought of as multi-dimensional arrays. Any tensor order can be reshaped into another, moving down an order by combining dimensions (flattening a matrix into a vector) or moving up an order by adding dummy dimensions (a vector of length 3 becomes a matrix (3×1) or a order 3 tensor $(3 \times 1 \times 1)$.

The most common operation in tensor networks is a contraction in which a common dimension of neighboring tensors is summed over. This is a generalization of matrix-vector or matrix-matrix multiplication in that we contract across one dimension of the first and down a dimension of the second. When contracting dimensions of larger and larger systems, the most computationally efficient contraction order is not obvious and is an active area of research [16]. This is represented by connecting two tensors in a diagram as seen in Fig. 2.2 and described in Algorithm 1.

An MPS is simply an extension of this concept where each tensor represents a single site. Each tensor has a physical dimension which points outward into which local operations can be contracted into the site. Typically, downwards facing physical dimensions represent a ket and upwards facing are the bra or the conjugate. Between each tensor is a bond dimension which represents entanglement in the system and grows exponentially towards the middle of the chain of sites. If these bond dimensions are full, it is an

Algorithm 1 Tensor Contractions

```
procedure CONTRACT(A, B, [A dims to contract], [B dims to contract])
    Sum along given dimensions of A and B
    Example: $\big(A = R^{a_{\ell-2},a_{\ell}-1}\ B = M^{a_{\ell-1},a_\ell,\sigma_\ell}$ [right], [left] $\big)$
    $C^{a_{\ell-2},\sigma_\ell} = \sum_{a_{\ell-1}} R^{a_{\ell-2},a_\ell-1} M^{a_{\ell-1},a_\ell,\sigma_\ell}$
end procedure
```

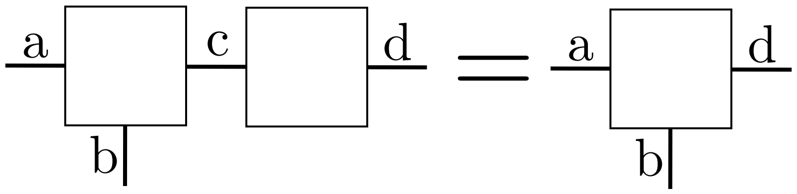


Figure 2.2: Example of an order 3 tensor with dimensions (a, b, c) and order 2 tensor with (c, d) contracting along the c dimension

exact representation of the quantum state. However, the exponential growth is problematic for scaling tensor networks. It is known that truncating these bond dimensions results in a good approximation of a state and so one can and should reduce it as much as possible through singular value truncation or compression [17]. MPOs are similarly represented, but have two physical dimensions which are used to perform operations on states to transform them into other states. A Hamiltonian applied to a state is shown as a tensor network in Fig. 2.3.

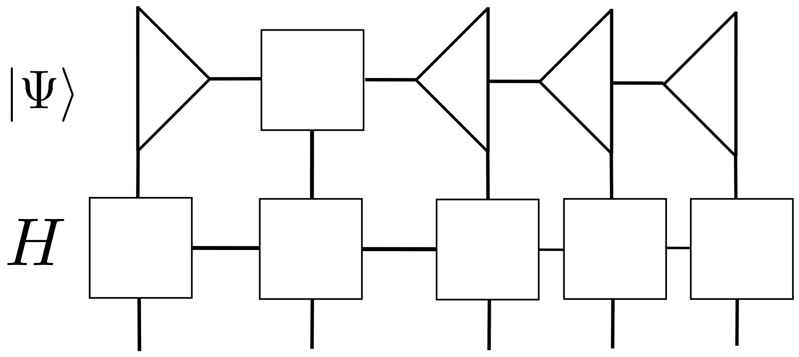
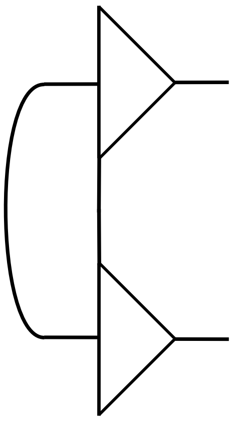


Figure 2.3: Diagram of the operation $H\,|\Psi\rangle$ where the MPS is in site canonical form at site 2

2.2.2 Canonical Forms

The tensors of an MPS can be put into canonical form in which the contraction along specific dimensions results in the identity. This speeds up calculations as a tensor of this form can be entirely ignored if it contracts with itself such as in the case of a scalar product or local expectation value. A tensor can be in left canonical or right canonical form which relates to which contractions lead to the identity represented by a single line in tensor network notation. This is a generalization of the matrix properties $MM^\dagger = \mathbb{1}$ and $M^\dagger M = \mathbb{1}$ respectively. This can be seen in Fig. 2.4.


(a) Left-Canonical Form

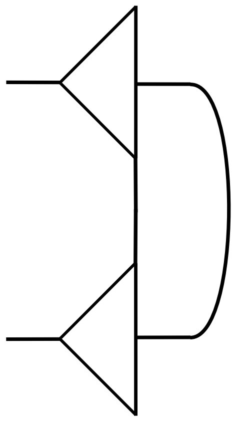

(b) Right-Canonical Form

Figure 2.4: Canonical forms which simplify to the identity

2.2.3 Decompositions and Normalization

It is possible to convert between these forms using the QR decomposition or Singular Value Decomposition (SVD). Since these operations are only defined for matrices, a basic explanation of how to reshape tensors in order to decompose and update them can be found in Algorithm 2.

Algorithm 2 MPS Tensor Decompositions

```
function MATRICIZE($M$)
    $M^{(a_{\ell-1}\sigma_\ell),a_\ell} \leftarrow M^{a_{\ell-1},a_\ell,\sigma_\ell}$    ▷ Combine left bond $a_{\ell-1}$ and physical dimension $\sigma_\ell$
    return $M^{\text{mat}} := M^{(a_{\ell-1}\sigma_\ell),a_\ell}$
end function
function DECOMPOSE($M$, DecompositionType)
    $M^{\text{mat}}$ = Matricize(M)
    if DecompositionType = SVD then
        $U, S, V = \text{SVD}(M^{\text{mat}})$
        return U, S, V
    else if DecompositionType = QR then
        $Q, R = \text{QR}(M^{\text{mat}})$
        return Q, R
    end if
end function
```

These can also be combined such that the MPS is in site canonical or mixed canonical form. This means there is a site selected with no specific form such that everything left of it is left-canonical and everything right of it is right-canonical. Many tensor network algorithms such as the Density Matrix Renormalization Group (DMRG) work by sweeping across the chain of sites. Keeping this in a site canonical form allows operations to focus purely on that site known as the orthogonality center and ignore the others. This sweeping algorithm is used often and it is a crucial component of this thesis. Given an MPS in site canonical form, the orthogonality center can be shifted to the right by performing a decomposition, setting the eigenvector matrix of the QR or SVD to the original tensor after reshaping, and contracting the other matrices into the next site. This can be seen in Algorithm 3 and Fig. 2.5.

In this thesis, states that lose their norm due to noise are used, so it is important to understand an MPS is normalized. This is done by continuously shifting the orthogonality center through decompositions like above all the way across the chain. Once the end is reached, the extra matrices such as the R in the QR decomposition or the $S, V^\dagger$ in the SVD hold a singular value which is the previous norm. When normalizing the state, this can effectively just be dropped after the final decomposition.

Algorithm 3 Shifting Orthogonality Center Right

```
procedure ShiftOrthogonalityCenterRight($|\Psi\rangle$, $\ell$)
    $M_\ell = |\Psi^{[\ell]}\rangle$                                   ▷ Current center at $\ell$th site Matrix of MPS $|\Psi\rangle$
    $Q_\ell, R_\ell$ = Decompose($M_\ell$, DecompositionType=QR)                 ▷ SVD also valid
    $Q_\ell^{a_{\ell-1},a_\ell,\sigma_\ell} \leftarrow Q_\ell^{(a_{\ell-1}\sigma_\ell),a_\ell}$                        ▷ Re-open combined dimensions
    $|\Psi^{[\ell]}\rangle \leftarrow Q_\ell$                               ▷ Update site to be left canonical
    if $\ell \neq L$ then                                        ▷ $L$ is length of MPS
        $M_{\ell+1} = |\Psi^{[\ell+1]}\rangle$
        $M'_{\ell+1}$ = Contract($R_\ell, M_{\ell+1}$ [right], [left] )          ▷ Contract R into next site
        $|\Psi^{[\ell+1]}\rangle \leftarrow M'_{\ell+1}$
    else if $\ell = L$ then
        $R_\ell$ is the norm, ignore it/throw it away if normalizing
    end if
end procedure
```

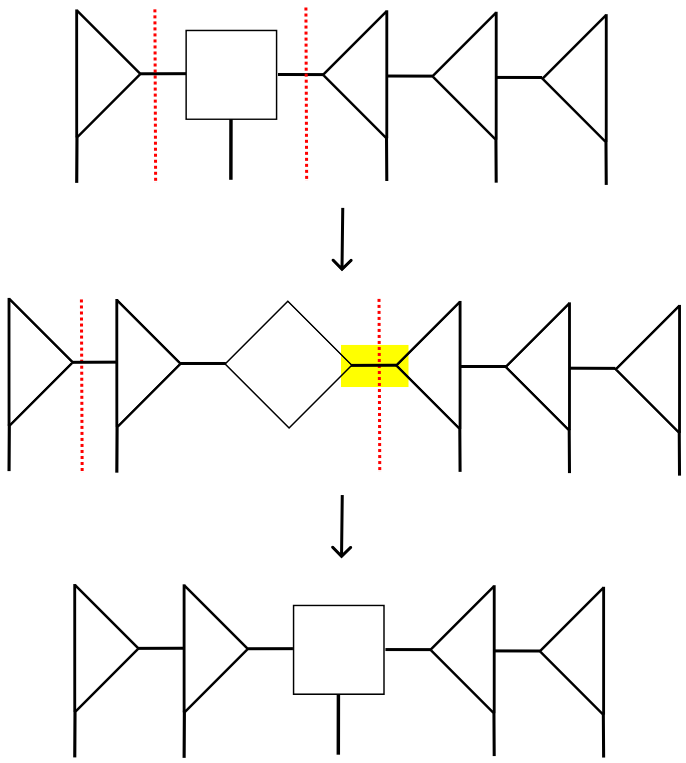


Figure 2.5: QR decomposition on the current site to move the orthogonality center to the right by contracting the R into the next site

2.3 Mathematics of Tensor Networks

In order to properly understand the time evolution algorithm introduced in the next section, it is important to understand some of the deeper mathematical properties of tensor networks. This section will re-introduce some concepts already mentioned, but at a more rigorous level. Understanding these concepts is very useful for further developing tensor network algorithms.

2.3.1 Geometry of Matrix Product States

First some mathematical properties of MPSs and their ability to represent quantum states in a Hilbert space need to be introduced. It can be shown that an MPS, being a set of parameters i.e. elements in tensors, can represent a manifold within the Hilbert space. The properties and geometry of both the parameter space within which the MPS lies and its corresponding manifold in Hilbert space will later be used to define the steps necessary for performing a time-evolution of the quantum state as a tensor network according to the Dirac-Frenkel Time-Dependent Variational Principle (TDVP).

Gauge Invariance and Canonical Forms

A parameter space $\mathbb{M} \subseteq \mathbb{R}^{\eta}$ is defined where each point $M \in \mathbb{M}$ corresponds to an ordered set of tensors containing parameters that define a Matrix Product State (MPS) of L lattice sites. The quantities $\chi_{n-1}, \chi_n, d_n \in \mathbb{Z}$ denote the local left and right bond dimensions and local physical dimension, respectively, at each site of the lattice. The MPS can then be defined as a set of tensors $M := \{M_n \in \mathbb{C}^{\chi_{n-1} \times \chi_n \times d_n} : 1 \leq n \leq L\}$. Each element of this set is a single tensor M_n corresponding to site $n \in \mathbb{Z}$ containing parameters indexed by $M_n^{a_{n-1}, a_n, \sigma_n} \in M_n$. Each site has virtual bond indices $\{a_n \in \mathbb{Z} : 1 \leq a_n \leq \chi_n\}$ and physical index $\{\sigma_n \in \mathbb{Z} : 1 \leq \sigma_n \leq d_n\}$ with $a_0 = a_L = 1$. This results in the dimension of the parameter space $\eta = \prod_{n=1}^{L} \chi_{n-1}\chi_n d_n$.

For a Hilbert space $\mathcal{H} := \mathbb{C}^{\prod_{n=1}^{L} d_n}$, there exists a map Ψ from the parameter space $\mathbb{M}$ to a smooth submanifold $\mathcal{M} \subseteq \mathcal{H}$ such that all points on the manifold $|\Psi(M)\rangle \in \mathcal{M}$ correspond to a quantum state described by the parameters of the MPS i.e.

$$\begin{aligned} \Psi : \mathbb{M} &\to \mathcal{M} \subseteq \mathcal{H} \\ M &\mapsto |\Psi(M)\rangle \in \mathcal{M} \end{aligned} \tag{2.36}$$

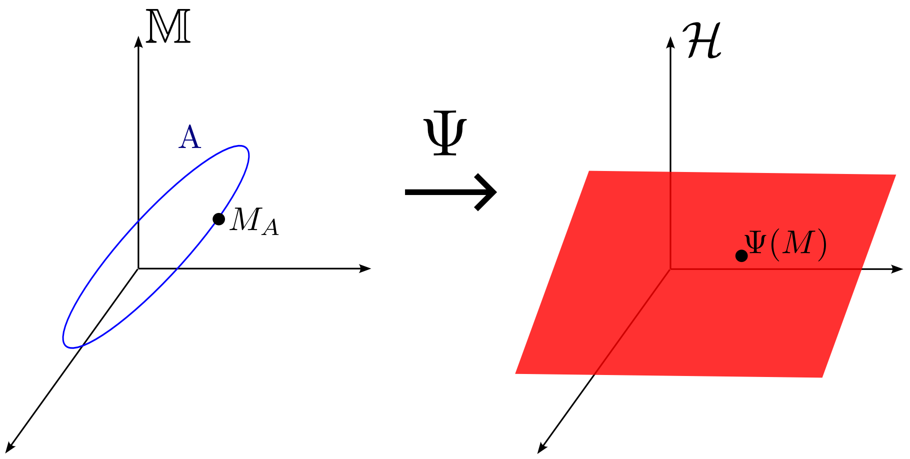


Figure 2.6: Wavefunction represented as a map from a parameter space containing MPSs and a manifold in Hilbert space

The map Ψ is injective such that there exists an orbit $A \in \mathbb{M}$ where $\forall M_A \in A$ then $|\Psi(M_A)\rangle = |\Psi(M)\rangle$. Proof that A is an orbit rather than disconnected points can be found in [8]. From this, it follows that MPSs are gauge-invariant and non-unique for describing quantum states [5].

Given a group of local gauge transformations on the MPS defined as the direct product of linear groups i.e. isomorphisms to the group of invertible matrices

$$\begin{aligned} G_{\text{MPS}} &:= \prod_{n=1}^{L} \text{GL}(\chi_n; \mathbb{C}) \\ &= \text{GL}(\chi_1, \mathbb{C}) \times \text{GL}(\chi_2, \mathbb{C}) \times \dots \end{aligned} \tag{2.37}$$

A right group action is defined such that

$$\begin{aligned} \Gamma : \mathbb{M} \times G_{\text{MPS}} &\to \mathbb{M} \\ (M, G) &\mapsto M^{[G]} \end{aligned} \tag{2.38}$$

where $(M, G) = ((M_n)_{n=1}^{L}, (G_n)_{n=1}^{L})$, $M^{[G]} = (M_n^{[G]})_{n=1}^{L}$, $M_n^{[G]} = G_{n-1}^{-1} M_n G_n$, and $G_0 = G_L$. The cancellation of the matrices $G_{n-1}^{-1} G_n$ leads to the gauge invariance of the MPS such that

$$\forall G \in G_{\text{MPS}}, \forall M_A \in A : |\Psi^{[G]}(M_A)\rangle = |\Psi(M)\rangle \tag{2.39}$$

Due to the gauge invariance, there is some choice in the structure of the tensors, so it is possible to specify certain conditions i.e. fix the gauge of the tensors in order for them to be more useful in further calculations. Left- and right-gauge fixing conditions can be applied such that for each $M_n \in M$, the tensors must satisfy conditions which reduce them to the identity [5].

Definition 2.3.1 (Left Canonical Form (Left-Gauge Fixing)). *An MPS tensor is said to be in left canonical form if the contraction of the left bond dimension and physical dimension of a tensor M_n and its conjugate $\overline{M}_n$ reduces to the identity*

$$\sum_{\sigma_n, a_{n-1}}^{d_n, \chi_{n-1}} M_n^{a_{n-1}, a_n, \sigma_n} \overline{M}_n^{a_{n-1}, a_n, \sigma_n} = \mathbb{1}_{\chi_n \times \chi_n} \tag{2.40}$$

Definition 2.3.2 (Right Canonical Form (Right-Gauge Fixing)). *An MPS tensor is said to be in right canonical form if the contraction of the right bond dimension and physical dimension of a tensor M_n and its conjugate $\overline{M}_n$ reduces to the identity*

$$\sum_{\sigma_n, a_n}^{d_n, \chi_n} \overline{M}_n^{a_{n-1}, a_n, \sigma_n} M_n^{a_{n-1}, a_n, \sigma_n} = \mathbb{1}_{\chi_{n-1} \times \chi_{n-1}} \tag{2.41}$$

Definition 2.3.3 (Mixed Canonical Form). *An orthogonality center is defined at some site $\{j \in \mathbb{Z} : 1 \leq j \leq L\}$. This means that all sites $n < j$ are left canonical, the site j itself has no extra condition applied to it, and all sites $n > j$ are right canonical i.e.*

$$\begin{aligned} \sum_{\sigma_n, a_{n-1}}^{d_n, \chi_{n-1}} A_n^{a_{n-1}, a_n, \sigma_n} \overline{A}_n^{a_{n-1}, a_n, \sigma_n} &= \mathbb{1}_{\chi_n \times \chi_n} : n < j \\ \sum_{\sigma_n, a_n}^{d_n, \chi_n} \overline{B}_n^{a_{n-1}, a_n, \sigma_n} B_n^{a_{n-1}, a_n, \sigma_n} &= \mathbb{1}_{\chi_{n-1} \times \chi_{n-1}} : n > j \end{aligned} \tag{2.42}$$

Left- and right-canonical tensors are renamed in tensor network notation to $M_n \to A_n$ and $M_n \to B_n$ respectively. This results in an MPS in mixed canonical form described by the ordered list $M = (A_1, \ldots, A_{j-1}, M_j, B_{j+1}, \ldots, B_L)$. This naming convention will be used from here on.

Decompositions for Gauge Fixing

These gauge conditions can then be applied to an MPS tensor by using a QR decomposition or Singular Value Decomposition (SVD) such that

$$M_n = QR = A_n C_n \tag{2.43}$$

or

$$M_n = U_n S_n V_n^\dagger = A_n C_n = C_n B_n \tag{2.44}$$

then setting $A_n = U_n$, $B_n = V_n^\dagger$, and $C_n = S_n V_n^\dagger$ or $C_n = U_n S_n$ where C_n is will be be referred to later as a bond tensor. Detailed information on the decompositions in tensor networks can be found in [17].

2.3.2 Tangent Space

A key component for implementing the Dirac-Frenkel Time-Dependent Variational Principle (TDVP) algorithm is understanding the structure of the tangent space of this manifold [6].

For each point $|\Psi(M)\rangle \in \mathcal{M}$, corresponding tangent space can be defined that is spanned by all the directional derivatives of the state such that

$$\mathcal{T}_M\mathcal{M} := \text{span}\{|\partial_i\Psi\rangle : i = 1, \ldots, \eta\} \tag{2.45}$$

where η denotes the total number of parameters in the MPS. This directional derivative corresponds to each parameter where $i = (n, a_{n-1}, a_n, \sigma_n)$ is a collective index such that

$$|\partial_i\Psi\rangle := \frac{\partial}{\partial M_i}|\Psi(M)\rangle = \frac{\partial}{\partial M_n^{a_{n-1},a_n,\sigma_n}}|\Psi(M)\rangle \tag{2.46}$$

This forms the full tangent bundle for the manifold $\mathcal{TM} := \bigcup_M \mathcal{T}_M\mathcal{M}$. Each vector in this tangent space can be defined by the overcomplete basis of directional derivatives such that [18]

$$|\Phi(N;M)\rangle := \sum_i N^i \frac{\partial}{\partial M_i}|\Psi(M)\rangle = \sum_i N^i |\partial_i\Psi(M)\rangle \tag{2.47}$$

A bundle map is then defined from the tangent space of the orbit to the tangent space of the manifold where [4]

$$\begin{aligned} d\Psi : \mathcal{T}_A\mathcal{M} &\to \mathcal{T}_M\mathcal{M} \\ (N, M) &\mapsto |\Phi(N;M)\rangle \end{aligned} \tag{2.48}$$

This allows any operations on the parameters i.e. the MPS to be mapped and understood in the context of the Hilbert space manifold.

2.4 Time-Dependent Variational Principle (TDVP)

An essential part of this thesis is using the Dirac-Frenkel Time-Dependent Variational Principle (TDVP) as a tool for unitary time evolution. This was selected as it has the ability to consider longer-range interactions than nearest-neighbor and also does not naturally increase the bond dimension of the matrix product states. This could simulate for example a grid-like MPS where "neighboring" sites are not nearest neighbors in the chain. For specific applications with pure nearest-neighbor interactions, Time-Evolving Block Decimation (TEBD) would be a reasonable alternative.

2.4.1 Approximating the Schrödinger Equation

The time evolution of a state on this manifold can then be analyzed wherein the time-dependent Schrödinger equation is evolved according to a Hamiltonian which is a self-adjoint operator on the Hilbert space where $\hat{H} \in L(\mathcal{H})$ and $\hbar = 1$.

$$i\partial_t \left|\Psi(M(t))\right\rangle = \hat{H} \left|\Psi(M(t))\right\rangle \tag{2.49}$$

Notice that since the left hand side is a partial derivative, it lies within the tangent space and can be described by tangent vectors

$$i\partial_t \left|\Psi(M(t))\right\rangle = \left|\Phi(\dot{M}; M(t))\right\rangle \tag{2.50}$$

where the tangent vector Φ is related to the time-evolved parameters $\dot{M}$. Due to the definition of the tangent vector in Eq.(2.47), this converts the Schrödinger equation from a higher-dimensional Hilbert space to a set of non-linear, symplectic partial differential equations on the manifold which can be mapped to the parameter space. For brevity, the time-dependence notation is dropped such that the parameters $M := M(t)$.

The application of the Hamiltonian to the right hand side of Eq.(2.49) leads to an increase in the number of necessary parameters i.e. an increase in the bond dimensions such that $\chi_n \leq \tilde{\chi}_n$ and this means a higher-dimensional parameter space $\tilde{\mathbb{M}}$ is needed to describe the evolved quantum state. Due to this, the state evolves away from the manifold to a larger manifold such that $\hat{H}\left|\Psi(M)\right\rangle \in \tilde{\mathcal{M}}$ where $\mathcal{M} \subseteq \tilde{\mathcal{M}}$.

The time-evolved parameters are then constrained by minimizing the difference between the true evolution and the tangent vector. From this follows the minimization problem and the following proposition stated in [18]

$$\dot{M} = \underset{N}{\operatorname{argmin}} \| \hat{H}\left|\Psi(M)\right\rangle - \left|\Phi(N; M)\right\rangle \|_2^2 \tag{2.51}$$

Proposition 2.4.1. *The minimization problem in Eq.(2.51) is equivalent to projecting the time-evolved state $\hat{H}\left|\Psi(M)\right\rangle$ onto the tangent space of the initial manifold $\mathcal{T}_M\mathcal{M}$ thus minimizing the difference between a tangent vector and the states location on the manifold according to an action principle.*

$$i\partial_t \left|\Psi(M)\right\rangle = P_{\mathcal{T}_M\mathcal{M}} \hat{H} \left|\Psi(M)\right\rangle \tag{2.52}$$

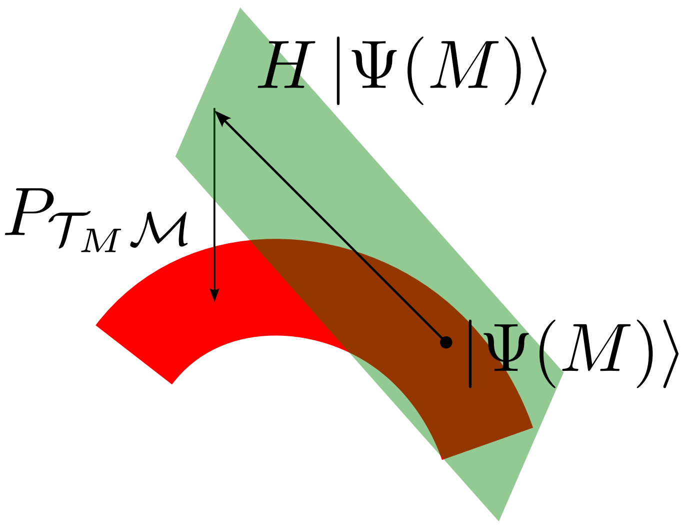


Figure 2.7: Time evolution of the MPS away from the manifold along its tangent space followed by a projection back to the manifold

Proposition 2.4.2. *The manifold $\mathcal{M}$ is a symplectic submanifold of $\mathcal{H}$.*

The previous proposition and following properties are stated in [12]. The inner product $\langle\cdot|\cdot\rangle$ of the Hilbert space which is antilinear in the first argument and linear in the second argument can be used to define the real bilinear form

$$\omega(|\Psi(M_1)\rangle\,,|\Psi(M_2)\rangle) = -2\mathrm{Im}\,\langle\Psi(M_1)|\Psi(M_2)\rangle \tag{2.53}$$

For every state $|\psi\rangle := |\Psi(M)\rangle \in \mathcal{H}$, there exists a unique projection $|w\rangle := P_{\mathcal{T}_M\mathcal{M}}\,|\Psi(M)\rangle$ where $|\Psi(M)\rangle \in \mathcal{T}_M\mathcal{M}$ such that $\omega(|v\rangle\,,|w\rangle) = \omega(|v\rangle\,,|\psi\rangle)\ \forall\,|v\rangle \in \mathcal{T}_M\mathcal{M}$. The non-degeneracy of the two-form ω means that both the manifold and projector are symplectic. In terms of the time-evolution, the two-form can be written as a function of the time-evolved state

$$\omega(v,\frac{du}{dt}) = 2\mathrm{Re}\,\langle v|Hu\rangle \tag{2.54}$$

From this, it is possible to analyze certain properties of the dynamics on this manifold.

Theorem 2.4.3 (Energy Conservation)**.** *The total energy $\langle H\rangle$ of a symplectic system is conserved.*

Proof: For some state $|u\rangle$ as defined above and Hamiltonian $\hat{H}$, it can be shown that

$$\frac{d}{dt}\,\langle u|\,\hat{H}\,|u\rangle = 2\mathrm{Re}\,\langle\dot{u}|Hu\rangle = \omega(|\dot{u}\rangle\,,|\dot{u}\rangle) = 0 \tag{2.55}$$

Theorem 2.4.4 (Invariants)**.** *Let $\hat{A}$ be a self-adjoint operator which commutes with the Hamiltonian $\hat{H}$ such that $[H,A] = 0$. If this operator acts on states in the manifold such that $\hat{A}\,|u\rangle \in \mathcal{T}_M\mathcal{M}\ \forall\,|u\rangle \in \mathcal{M}$ then its average $\langle\hat{A}\rangle$ is conserved along variational approximations $|u(t)\rangle$ such that $\langle u(t)|\,\hat{A}\,|u(t)\rangle = const.$*

Corollary 2.4.5 (Norm Conservation). *By setting $\hat{A} = \mathbb{1}$, the norm is conserved such that $\langle u(t)|u(t)\rangle = const.$*

Projection

During the optimization step, each tensor is optimized site-wise in sweeps similar to the Density Matrix Renormalization Group (DMRG) algorithm [17] since it is a commonly implemented, fundamental algorithm used in tensor networks and thus makes this TDVP implementation accessible. For the ordered set of tensors (M_n) this means moving left-right from site $n = (1, \dots, L-1)$ followed by right-left where $n = (L, \dots, 2)$ such that at each step a single site tensor M_n is optimized. It is important work with an MPS in mixed-canonical form such that $M = (A_1, \dots, A_{j-1}, M_j, B_{j+1}, \dots, B_L)$ for some orthogonality center j in order to greatly simplify the algorithm due to reduction of tensors to the identity.

Next a left and right bipartition of the Hilbert space around a site j is introduced such that

$$\mathcal{H} = \mathcal{H}_j^L \otimes \mathcal{H}_{j+1}^R \tag{2.56}$$

with $\mathcal{H}_j^L = \mathcal{H}_1 \otimes \cdots \otimes \mathcal{H}_j$ and $\mathcal{H}_{j+1}^R = \mathcal{H}_{j+1} \otimes \cdots \otimes \mathcal{H}_L$. Within this bipartition, left and right states are defined such that

$$|\Psi_j^L\rangle = \sum_{\{\sigma_n\},\{a_n\}} M_1 \dots M_j \, |\sigma_1, \dots, \sigma_j\rangle \tag{2.57}$$

$$|\Psi_j^R\rangle = \sum_{\{\sigma_n\},\{a_n\}} M_j \dots M_L \, |\sigma_j, \dots, \sigma_L\rangle \tag{2.58}$$

where σ_n are the physical dimensions and a_n are the bond dimensions associated with each tensor M_n. As full tensors are optimized at a time rather than individual elements, the indices are removed from the tensors for brevity such that $M_n^{a_{n-1}, a_n, \sigma_n} \to M_n$ as well as $|\Psi(M)\rangle \to |\Psi\rangle$. This is done often in literature and may be confusing due to Ψ being the map, the quantum state, and the MPS itself, however from now on it will refer to the MPS.

Left and right projectors can now be defined which project onto each of these subspaces.

$$\begin{aligned} P_j^L &: \mathcal{H} \to \mathcal{H}_j^L \otimes \mathcal{H}_{j+1}^R \\ &|\Psi\rangle \mapsto \left[|\Psi_j^L\rangle \langle \Psi_j^L| \otimes \mathbb{1}_{j+1}^R \right] |\Psi\rangle \\ &:= \left[\sum_{\{\sigma_n\}} \sum_{\{\overline{\sigma}_n\}} \sum_{\{a_n\}} M_1 \dots M_j \overline{M}_j \dots \overline{M}_1 \, |\sigma_1 \dots \sigma_j\rangle \langle \overline{\sigma}_1 \dots \overline{\sigma}_j| \otimes \mathbb{1}_{j+1}^R \right] |\Psi\rangle \end{aligned} \tag{2.59}$$

$$\begin{aligned} P_j^R &: \mathcal{H} \to \mathcal{H}_{j-1}^L \otimes \mathcal{H}_j^R \\ &|\Psi\rangle \mapsto (\mathbb{1}_{j-1}^L \otimes |\Psi_j^R\rangle \langle \Psi_j^R|) \, |\Psi\rangle \\ &:= \left[\mathbb{1}_{j-1}^L \otimes \sum_{\{\sigma_n\}} \sum_{\{\overline{\sigma}_n\}} \sum_{\{a_n\}} M_L \dots M_j \overline{M}_j \dots \overline{M}_L \, |\sigma_j \dots \sigma_L\rangle \langle \overline{\sigma}_j \dots \overline{\sigma}_L| \right] |\Psi\rangle \end{aligned} \tag{2.60}$$

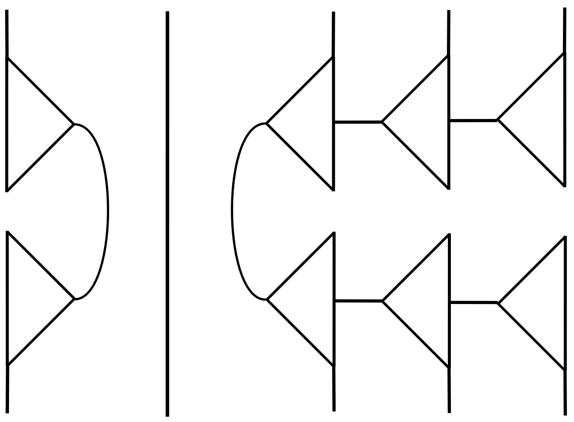


Figure 2.8: 5-Site local Krylov projector used to fix the orthogonality center at site 2

Note that if the state is in a canonical form, nearly all terms simplify for the left projector $M_i\overline{M_i} \to A_i\overline{A_i} = \mathbb{1}$ and right projector $\overline{M}_iM_i \to \overline{B}_iB_i = \mathbb{1}$.

The local Krylov projectors which are used to fix the orthogonality center to a site j are now defined as

$$\begin{aligned} \Pi_j^M &: \mathcal{H} \to \mathcal{T}_M\mathcal{M} \\ &|\Psi\rangle \mapsto (P_{j-1}^L \otimes \mathbb{1}_j \otimes P_{j+1}^R)\,|\Psi\rangle \end{aligned} \tag{2.61}$$

A filter which projects the state onto its bipartition is defined such that it can remove any states identical to the original $|\Psi\rangle$. This is used so that the algorithm does not project back to the original state or an equivalent representation in the parameter orbit.

$$\begin{aligned} F_j^M &: \mathcal{H} \to \mathcal{T}_M\mathcal{M} \\ &|\Psi\rangle \mapsto (P_j^L \otimes P_{j+1}^R)\,|\Psi\rangle \end{aligned} \tag{2.62}$$

This results in the tangent space projector $P_{\mathcal{T}_M\mathcal{M}}$ by using the above definitions [15]

$$\begin{aligned} P_{\mathcal{T}_M\mathcal{M}} &: \mathcal{H} \to \mathcal{T}_M\mathcal{M} \\ &|\Psi\rangle \mapsto (\sum_{j=1}^{L}\Pi_j^M - \sum_{j=1}^{L-1}F_j^M)\,|\Psi\rangle \end{aligned} \tag{2.63}$$

2.4.2 Implementing TDVP in Tensor Networks

When applying the tangent space projector to the Schrödinger equation, it can be split into L forward- and $L-1$ backward-evolving time differential equations which can be solved site-wise as discussed before with the analogy to DMRG. These correspond to the L sites and $L-1$ bonds of the MPS.

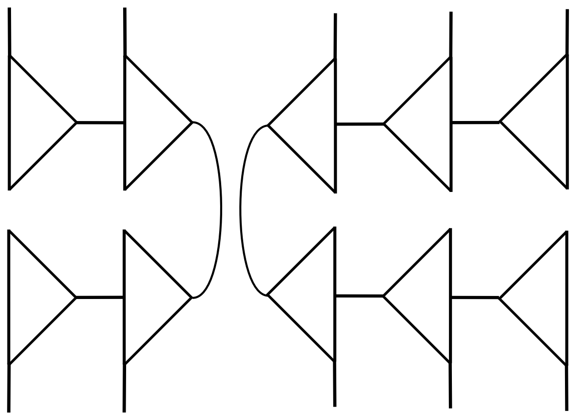


Figure 2.9: 5-Site local filter projector used to project onto a bipartition

$$\begin{aligned}\partial_t |\Psi(M)\rangle &= -iP_{\mathcal{T}_M\mathcal{M}} H |\Psi\rangle \\ &= -i(\sum_{j=1}^{L} \Pi_j^M - \sum_{j=1}^{L-1} F_j^M) H |\Psi\rangle \\ &= -i \sum_{j=1}^{L} \Pi_j^M H |\Psi\rangle + i \sum_{j=1}^{L-1} F_j^M H |\Psi\rangle\end{aligned} \tag{2.64}$$

From this, it is possible to define the following equations for solving time evolution at each site j

- Forward-Evolving Equations

$$\partial_t |\Psi\rangle = -i\Pi_j^M H |\Psi\rangle \tag{2.65}$$

- Backward-Evolving Equations

$$\partial_t |\Psi\rangle = +iF_j^M H |\Psi\rangle \tag{2.66}$$

These equations can be brought into a form which evolves each site tensor M_j and its corresponding bond tensor C_j. In order to do this, the MPS starts in a mixed canonical form at site j as prescribed in Eq.(2.42)

$$|\Psi\rangle = \sum_{\{\sigma_n\},\{a_n\}} A_1 \dots A_{j-1} M_j B_{j+1} \dots B_L |\sigma_1 \dots \sigma_L\rangle \tag{2.67}$$

Next a single site tensor is defined at site j

$$|\Psi_j\rangle = M_j |\sigma_j\rangle \tag{2.68}$$

as well as the site-wise derivative of the state as prescribed in Eq.(2.46)

$$|\partial_{M_j}\Psi\rangle = \sum_{\substack{\{\sigma_n\}\backslash\sigma_j \\ \{a_n\}\backslash\{a_{j-1},a_j\}}} A_1 \dots A_{j-1}B_{j+1}\dots B_L \,|\sigma_1,\dots\sigma_{j-1},\sigma_{j+1},\dots,\sigma_L\rangle \tag{2.69}$$

And equivalently, the definition of the left and right bipartition states is used as in Eq.(2.57) and (2.58), but in left- and right-canonical form around the site j

$$|\Psi^L_{j-1}\rangle = \sum_{\{\sigma_n\},\{a_n\}} A_1 \dots A_{j-1}\,|\sigma_1,\dots,\sigma_{j-1}\rangle \tag{2.70}$$

$$|\Psi^R_{j+1}\rangle = \sum_{\{\sigma_n\},\{a_n\}} B_{j+1} \dots B_L\,|\sigma_{j+1},\dots,\sigma_L\rangle \tag{2.71}$$

The tensor product of these can be further defined such that

$$|\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle = \sum_{\substack{\{\sigma_n\}\backslash\sigma_j \\ \{a_n\}\backslash\{a_{j-1},a_j\}}} A_1 \dots A_{j-1}B_{j+1}\dots B_L \,|\sigma_1,\dots\sigma_{j-1},\sigma_{j+1},\dots,\sigma_L\rangle \tag{2.72}$$

From these definitions, it is possible to derive several important equivalencies. The derivative and this bipartition are equivalent such that

$$|\partial_{M_j}\Psi\rangle = |\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle \tag{2.73}$$

The state can then be broken up into partitions of a left-environment, the site itself, and a right environment

$$|\Psi\rangle = |\Psi^L_{j-1}\rangle \otimes |\Psi_j\rangle \otimes |\Psi^R_{j+1}\rangle \tag{2.74}$$

Finally, the state is a product of its site-wise derivative and that site tensor

$$|\Psi\rangle = \partial_{M_j}|\Psi\rangle \otimes |\Psi_j\rangle \tag{2.75}$$

These definitions can now be used to bring the forward- and backward-evolving equations into the proper form [15].

Site Tensor Update

Beginning with the forward-evolving equation Eq.(2.65), $\langle\Psi^L_{j-1}| \otimes \langle\Psi^R_{j+1}|$ is applied to both sides starting with the LHS

$$\begin{aligned}
(\langle\Psi^L_{j-1}| \otimes \langle\Psi^R_{j+1}|)\partial_t|\Psi\rangle &= \partial_t[(\langle\Psi^L_{j-1}| \otimes \langle\Psi^R_{j+1}|)\,|\Psi\rangle] \\
&= \partial_t[(\langle\Psi^L_{j-1}| \otimes \langle\Psi^R_{j+1}|)\,|\Psi^L_{j-1}\rangle \otimes |\Psi_j\rangle \otimes |\Psi^R_{j+1}\rangle] \\
&= \partial_t|\Psi_j\rangle \\
&= \partial_t M_j\,|\sigma_j\rangle
\end{aligned} \tag{2.76}$$

where the ket in the final line would be dropped in tensor network notation.

Likewise, this is applied to the RHS

$$\begin{aligned}
(\langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}|)(-i\Pi^M_j H |\Psi\rangle) &= -i[\langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| \Pi^M_j H] |\Psi\rangle \\
&= -i[\langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| \Pi^M_j H] |\Psi^L_{j-1}\rangle \otimes |\Psi_j\rangle \otimes |\Psi^R_{j+1}\rangle \\
&= -iH^{\text{eff}}_j |\Psi_j\rangle \\
&= -iH^{\text{eff}}_j M_j |\sigma_j\rangle
\end{aligned} \tag{2.77}$$

with effective Hamiltonian $H^{\text{eff}}_j = \langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| \Pi^M_j H |\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle$.

Due to the mixed canonical form and applying the definition of the local Krylov projectors, this simplifies into

$$\begin{aligned}
H^{\text{eff}}_j &= \langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| \Pi^M_j H |\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle \\
&= \langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| \left[P^L_{j-1} \otimes \mathbb{1}_j \otimes P^R_{j+1}\right] H |\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle \\
&= \langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| \left[|\Psi^L_{j-1}\rangle \langle \Psi^L_{j-1}| \otimes \mathbb{1}_j \otimes |\Psi^R_{j+1}\rangle \langle \Psi^R_{j+1}|\right] H |\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle \\
&= \langle \Psi^L_{j-1}| \otimes \langle \Psi^R_{j+1}| H |\Psi^L_{j-1}\rangle \otimes |\Psi^R_{j+1}\rangle \\
&= \langle \partial_{M_j}\Psi| H |\partial_{M_j}\Psi\rangle
\end{aligned} \tag{2.78}$$

Overall, this results in a differential equation for updating each site tensor M_j

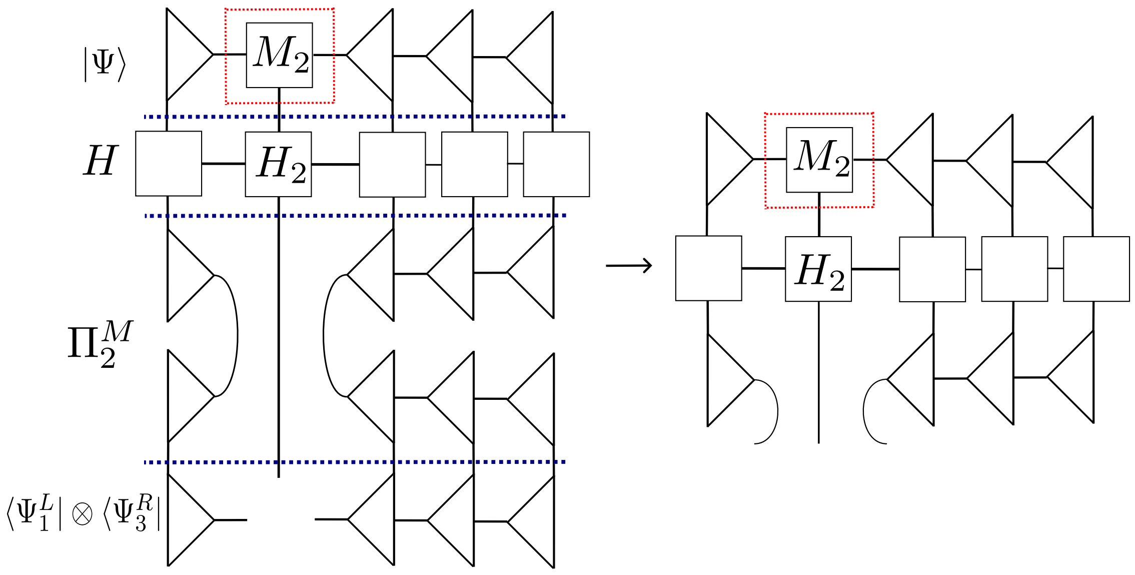
$$\partial_t M_j = -iH^{\text{eff}}_j M_j \tag{2.79}$$

$$M_j(t) = e^{-iH^{\text{eff}}_j t} M_j \tag{2.80}$$

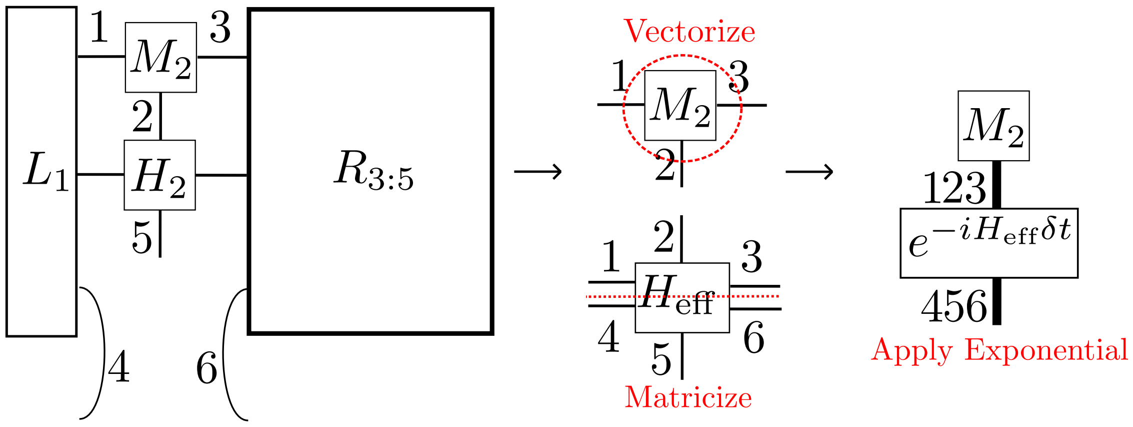
As the sweep is performed, this equation is solved for a site j before solving the backwards-evolving equation i.e. updating the associated bond tensor at bond j. Each one is then alternated between until they have been solved for all parameter tensors.

Algorithm 4 Forward Evolution

```
function ForwardEvolve($|\Psi\rangle, H, L, R, \delta t, \ell$)
    $M_\ell = |\Psi^{[\ell]}\rangle$
    $H_{\text{eff}} = \text{Contract}\left(L_{1:\ell-1}, M_\ell, H_\ell, R_{\ell:L}\right)$
    $H^{\text{mat}}_{\text{eff}} = \text{Matricize}\left(H_{\text{eff}}\right)$
    $M^{\text{vec}}_\ell = \text{Vectorize}\left(M_\ell\right)$
    $M^{\text{vec}'}_\ell = \exp\left(-iH^{\text{mat}}_{\text{eff}}\delta t\right) M^{\text{vec}}_\ell$        ▷ Applied with Lanczos method
    $M'_\ell = \text{Reshape}\left(M^{\text{vec}'}_\ell\right)$
    $|\Psi^{[\ell]}\rangle \leftarrow M'_\ell$
end function
```



(a) Simplification to forward evolution (site tensor update) to convert it to a DMRG-like sweep due to site canonical form



(b) Update of site tensor with environments. All right environments are calculated before starting a sweep (so environments from site L to 1, from L to 2, etc.) and then just replaced at each step.

Figure 2.10: Solving the forward- evolving equation for a single site

Bond Tensor Update

After solving at site j according to Eq.(2.80), the newly time-evolved tensor $M_j \rightarrow \tilde{M}_j$ is decomposed using either a QR decomposition or Singular Value Decomposition as described in Eq.(2.43) and Eq.(2.44) such that $\tilde{M}_j = \tilde{A}_j\tilde{C}_j$. This moves the focus from site j to the following bond.

Now $\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|$ can be applied to both sides of the backwards-evolving equation Eq.(2.66) as done with the forward-evolving equation, however, the updated left

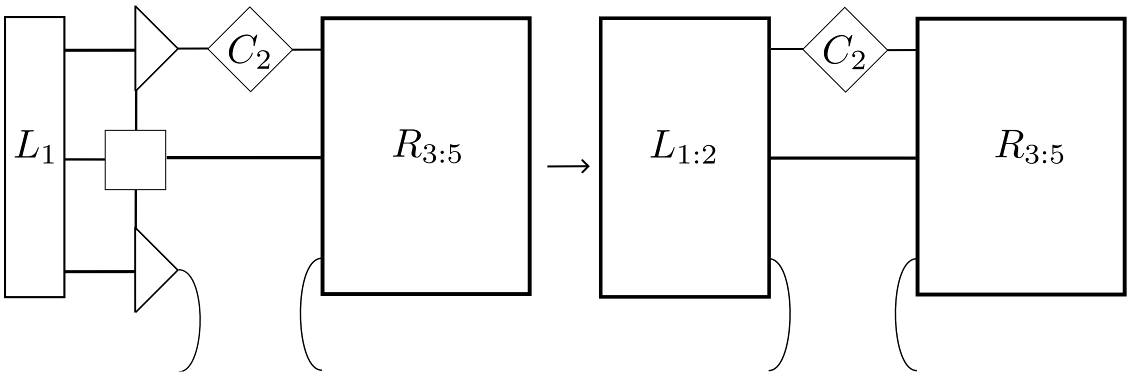


Figure 2.11: The updated M_j from the forward evolution is decomposed into $M_j = A_jC_j$ with a QR decomposition. Then, the site of the MPS is updated to A_j and the left environment is updated as seen above.

state $|\tilde{\Psi}_j^L(\tilde{A}_j)\rangle$ containing the updated canonical tensor $\tilde{A}_j$ is now applied. Terms with tilde contain the forward-evolved tensor $\tilde{M}_j$. This also sees a gauge transformed state such that $|\tilde{\Psi}_j(\tilde{M}_j)\rangle = C_j\,|\tilde{\Psi}_j(\tilde{A}_j)\rangle$.

$$\begin{aligned}
(\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|)\partial_t\,|\tilde{\Psi}\rangle &= \partial_t[(\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|)\,|\tilde{\Psi}\rangle] \\
&= \partial_t[(\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|)\,|\Psi_{j-1}^L\rangle \otimes |\tilde{\Psi}_j(\tilde{M}_j)\rangle \otimes |\Psi_{j+1}^R\rangle] \\
&= \langle\tilde{\Psi}_j(\tilde{A}_j)|\,\partial_t\,|\tilde{\Psi}_j(\tilde{M}_j)\rangle \\
&= \langle\tilde{\Psi}_j(\tilde{A}_j)|\,\partial_t C_j\,|\tilde{\Psi}_j(\tilde{A}_j)\rangle \\
&= \partial_t C_j
\end{aligned} \tag{2.81}$$

Likewise for the RHS

$$\begin{aligned}
(\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|)(+iF_j^M H\,|\Psi\rangle) &= +i[\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|\,F_j^M H]\,|\Psi\rangle \\
&= +i[\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|\,F_j^M H]\,|\Psi_{j-1}^L\rangle \\
&\quad \otimes |\tilde{\Psi}_j(\tilde{M}_j)\rangle \otimes |\Psi_{j+1}^R\rangle \\
&= +i[\langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|\,F_j^M H]\,|\Psi_{j-1}^L\rangle \\
&\quad \otimes C_j\,|\tilde{\Psi}_j(\tilde{A}_j)\rangle \otimes |\Psi_{j+1}^R\rangle \\
&= +i[\langle\tilde{\Psi}(\tilde{A}_j)|\,F_j^M H\,|\tilde{\Psi}(\tilde{A}_j)\rangle]C_j \\
&= +iH_j^{\text{eff}}C_j
\end{aligned} \tag{2.82}$$

Since the MPS is in mixed canonical form, an effective Hamiltonian is created for the bond j by applying the definition of the filter projection such that

$$\begin{aligned}
H_j^{\text{eff}} &= \langle\tilde{\Psi}(\tilde{A}_j)|\, F_j^M H\, |\tilde{\Psi}(\tilde{A}_j)\rangle \\
&= \langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R| \left[\tilde{P}_j^L \otimes P_{j+1}^R\right] H(|\tilde{\Psi}_j^L(\tilde{A}_j)\rangle \otimes |\Psi_{j+1}^R\rangle) \\
&= \langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R| \left[|\tilde{\Psi}_j^L(\tilde{A}_j)\rangle \langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes |\Psi_{j+1}^R\rangle \langle\Psi_{j+1}^R|\right] H\, |\tilde{\Psi}_j^L(\tilde{A}_j)\rangle \otimes |\Psi_{j+1}^R\rangle \\
&= \langle\tilde{\Psi}_j^L(\tilde{A}_j)| \otimes \langle\Psi_{j+1}^R|\, H\, |\tilde{\Psi}_j^L(\tilde{A}_j)\rangle \otimes |\Psi_{j+1}^R\rangle \\
&= \langle\tilde{\Psi}(\tilde{A}_j)|\, H\, |\tilde{\Psi}(\tilde{A}_j)\rangle
\end{aligned} \tag{2.83}$$

It is then possible to update the bond tensor according to

$$\partial_t C_j = +iH_j^{\text{eff}} C_j \tag{2.84}$$

$$C_j(t) = e^{+iH_j^{\text{eff}} t} C_j \tag{2.85}$$

Combining the site and bond tensor updates above into sweeps results in the full TDVP algorithm for time-evolving an MPS described in Algorithm 6.

Algorithm 5 Backward Evolution

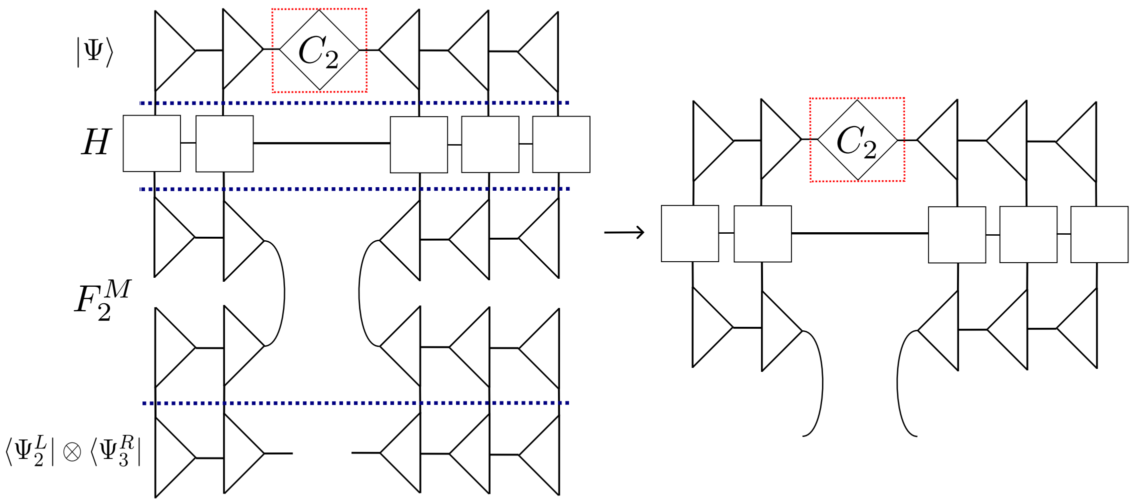
```
function BACKWARDEVOLVE((|Ψ'⟩, H, C_ℓ, L_{1:ℓ}, R_{ℓ:L}, δt/2, ℓ))
    $M_\ell = |\Psi^{[\ell]}\rangle$
    $B_{\ell+1} = |\Psi^{[\ell+1]}\rangle$                          ▷ Following site is canonical
    $H_{\text{eff}} = \text{Contract}\left(L_{1:\ell}, M_\ell, H_\ell, R_{\ell:L}\right)$
    $H_{\text{eff}}^{\text{mat}} = \text{Matricize}\left(H_{\text{eff}}\right)$
    $C_\ell^{\text{vec}} = \text{Vectorize}\left(C_\ell\right)$
    $C_\ell^{\text{vec}'} = \exp\left(+iH_{\text{eff}}^{\text{mat}}\delta t\right) C_\ell^{\text{vec}}$          ▷ Applied with Lanczos method
    $C'_\ell = \text{Reshape}\left(C_\ell^{\text{vec}'}\right)$
    $M'_{\ell+1} = \text{Contract}\left(C'_\ell, B_{\ell+1}\right)$
    $|\Psi^{[\ell+1]}\rangle \leftarrow M'_{\ell+1}$
end function
```

2.4.3 Error Analysis

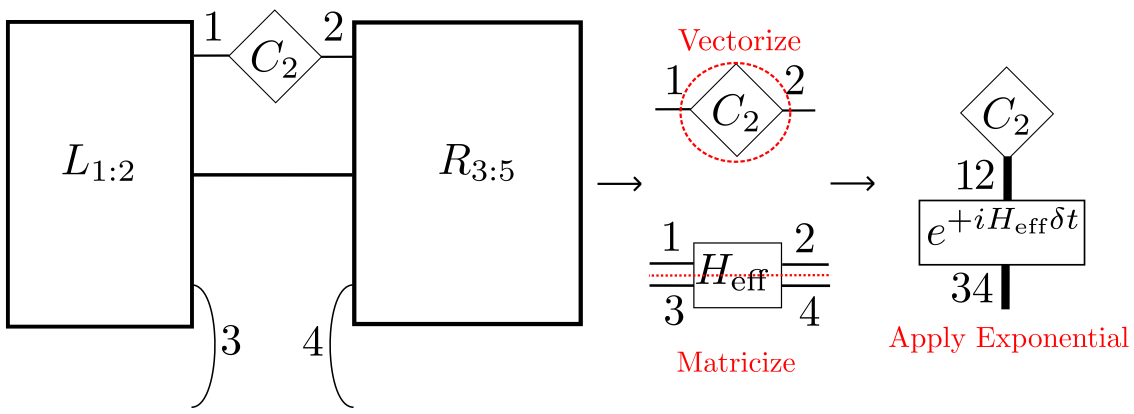
A general differential equation is defined as $\dot{y} = f(y)$ where $y = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a point in phase space and $f(y)$ is some vector-valued function describing the velocity of the solution $y(t)$ passing through the given point.

The phase space system has a fundamental property called the flow over time t which maps an initial value $y(0) = y_0$ to a phase space orbit $y(t)$ (also known as a trajectory, however not in the sense of a Monte Carlo trajectory). The mathematical definition of flow is as a group action where

$$\begin{aligned}
\psi &: X \times \mathbb{R} \to X \\
&(y_0, t) \mapsto \psi(y_0, t) =: y(t)
\end{aligned} \tag{2.86}$$



(a) Simplification to the backward evolution (bond tensor update) to convert it to a DMRG-like sweep due to site canonical form



(b) Update of bond tensor with environments. All right environments are calculated before starting a sweep (so environments from site L to 1, from L to 2, etc.) and then just replaced at each step.

Figure 2.12: Solving the backward-evolving equation for a single bond

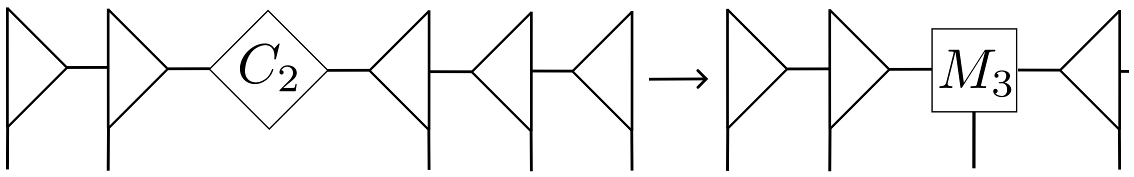


Figure 2.13: The updated C_j from the backward evolution is simply multiplied into the next site of the MPS in order to repeat the algorithm.

Algorithm 6 Single Timestep TDVP Algorithm

```
function InitializeEnvironments($|\Psi\rangle, H$)
    $L_1 = \mathbb{1}_{1\times1\times1}$
    $R_{L:L} = \mathbb{1}_{1\times1\times1}$    ▷ Environments for first and last site is an identity tensor
    for $\ell \in [L:1]$ do    ▷ Right to left sweep
        $M_\ell = |\Psi^{[\ell]}\rangle$
        $R_{\ell:L}$ = Contract $\left(R_{\ell+1:L}, M_\ell, H_\ell, \overline{M}_\ell\right)$    ▷ Contract previous R into right bonds
    end for
    return $L_1, \{R_{\ell:L}\}$
end function

function UpdateEnvironment($|\Psi\rangle, H, L_{[1:\ell-1]}$)
    $A_\ell, C_\ell$ = Decompose $\left(|\Psi'\rangle, \ell\right)$    ▷ QR decomposition
    $|\Psi'^{[\ell]}\rangle \leftarrow A_\ell$
    $L_{1:\ell}$ = Contract $\left(L_{1:\ell-1}, A_\ell, H_\ell, \overline{A}_\ell\right)$
    return $L_{1:\ell}$
end function

function TDVP($|\Psi\rangle, H, \delta t$)    ▷ L with no subscripts is length of MPS
    $L_1, \{R_{\ell:L}\}_{\ell=1}^{L}$ = InitializeEnvironments $\left(|\Psi\rangle, H\right)$
    for $\ell \in [1:L-1]$ do    ▷ Last site needs to forward evolve full timestep
        $|\Psi'\rangle$ = ForwardEvolve $\left(|\Psi\rangle, H, L_{1:\ell-1}, R_{\ell:L}, \frac{\delta t}{2}, \ell\right)$
        $L_{1:\ell}$ = UpdateEnvironment $\left(|\Psi'\rangle, H, L_{1:\ell-1}\right)$
        $|\Psi''\rangle$ = BackwardEvolve $\left(|\Psi'\rangle, H, C_\ell, L_{1:\ell}, R_{\ell:L}, \frac{\delta t}{2}, \ell\right)$
    end for
    $|\Psi'''\rangle$ = ForwardEvolve $\left(|\Psi''\rangle, H, L_{1:L-1}, R_L, \frac{\delta t}{2}, L\right)$
    for $\ell \in [L:1]$ do
        Right-left sweep as above but flipped
    end for
    return $|\Psi(t+\delta t)\rangle$
end function
```

An alternate definition of the flow which mathematically is actually known as a time-t map is defined by

$$\begin{aligned} \psi_t : \mathbb{R}^n &\to \mathbb{R}^n \\ y_0 &\mapsto \psi(y_0, t) =: y(t) \end{aligned} \tag{2.87}$$

From here on, this time-t map will define the exact flow.

Definition 2.4.6 (Time-Symmetry). *If the exact flow of a differential equation satisfies the property*

$$\psi_{-t}^{-1} = \psi_t \tag{2.88}$$

then it is time-symmetric.

The exact flow can be generalized to a discrete flow (also known as the numerical flow or one-step method) representing a mapping between timesteps

$$\begin{aligned} \phi_{\delta t} : \mathbb{R}^n &\to \mathbb{R}^n \\ y(t) &\mapsto \phi_{\delta t}(y(t)) =: y(t + \delta t) \end{aligned} \tag{2.89}$$

Definition 2.4.7 (Adjoint Method). *If a discrete method satisfies a property similar to the time-symmetry in Def. 2.4.6 such that*

$$\phi_{\delta t}^{*} = \phi_{-\delta t}^{-1} \tag{2.90}$$

then it is an adjoint method.

The definitions of the first-order Euler methods are considered where an explicit Euler method is an evaluation of the function at the next timestep based on an explicit evaluation of the function at the original timestep i.e.

$$y(t + \delta t) = y(t) + \delta t f(y(t)) \tag{2.91}$$

and likewise an implicit Euler method is based on an evaluation of the function at that timestep itself such that

$$y(t + \delta t) = y(t) + \delta t f(y(t + \delta t)) \tag{2.92}$$

The local integration error of the first-order Euler methods is given by $\mathcal{O}(\delta t^2)$. This section can be found with more detail in [7].

Structure Preservation and Numerical Error

It is important to use numerical methods that preserve geometric properties of the flow of the differential equations once discretized

$$M_j(t + \delta t) = e^{-iH_j^{\text{eff}} \delta t} M_j(t) \tag{2.93}$$

$$C_j(t+\delta t) = e^{+iH_j^{\text{eff}}\delta t} C_j(t) \tag{2.94}$$

This means choosing numerical integration methods such as symplectic integrators for Hamiltonian systems and structure-preserving methods on manifolds that preserve certain structures in the system. In this case, the symplecticity conserves not only the energy and operator phase space evolution $\dot{O} = i\{\hat{H}, \hat{O}\}$, but also the symmetries of single-site operators of the form

$$U = \prod_{\ell=1}^{L} u_{[\ell]} \tag{2.95}$$

which are especially suited for use in the MPS tensor network representation.

As mentioned in Section 2.4.2, the simplifications to the effective Hamiltonians due to the mixed canonical form lead to a DMRG-like integration sweep being possible. The site-wise left to right sweep wherein the discretized equations Eq.(2.93) and (2.94) are solved is an explicit Euler method and the following right to left sweep is an implicit Euler method. The combination of these methods, however, leads to an adjoint method and therefore a second-order symmetric integrator with $\mathcal{O}(\delta t^3)$.

Proposition 2.4.8. *The Time-Dependent Variational Principle has timestep error $\mathcal{O}(\delta t^3)$.*

Proof: TDVP is an adjoint method. This means that the error is calculated by comparing the results between an exact flow ψ_t and a discrete flow $\phi_{\delta t}$ of order p such that

$$\phi_{\delta t}(y_0) = \psi_{\delta t}(y_0) + C(y_0)\delta t^{p+1} + \mathcal{O}(\delta t^{p+2}) \tag{2.96}$$

where C is some first-order coefficient. Likewise, for the adjoint

$$\phi^*_{\delta t}(y_0) = \psi_{\delta t}(y_0) + (-1)^p C(y_0)\delta t^{p+1} + \mathcal{O}(\delta t^{p+2}) \tag{2.97}$$

the local error between the exact and discrete flow is thus given by

$$e = C(y_0)\delta t^{p+1} + \mathcal{O}(\delta t^{p+2}) \tag{2.98}$$

and

$$e^* = (-1)^p C(y_0)\delta t^{p+1} + \mathcal{O}(\delta t^{p+2}) \tag{2.99}$$

If $\phi_{\delta t}$ is adjoint, then $e = e^*$ and therefore from the RHS

$$C(y_0)\delta t^{p+1} + \mathcal{O}(\delta t^{p+2}) = (-1)^p C(y_0)\delta t^{p+1} + \mathcal{O}(\delta t^{p+2}) \tag{2.100}$$

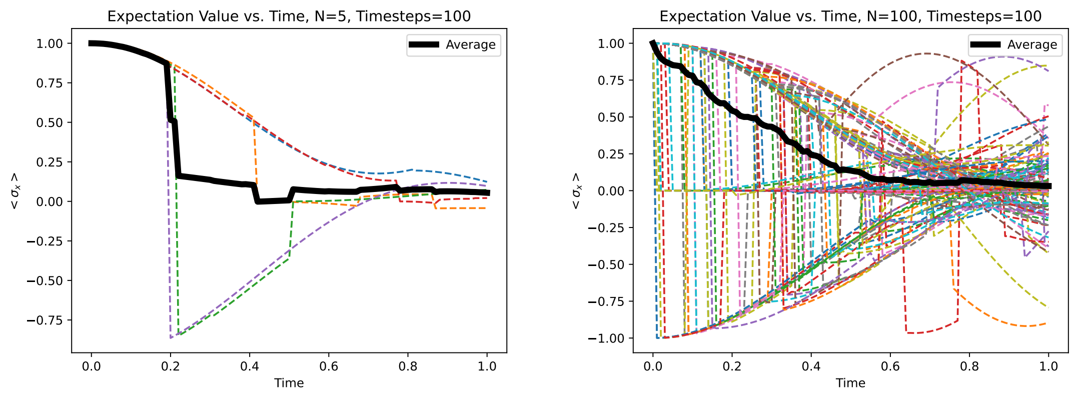
This means the order p must be even and therefore $C(y_0) = 0$ if p is odd. The maximum error is given by the minimal possible p such that $p = 1$ and $\mathcal{O}(\delta t^3)$. Taking into account that there are two sweeps, a half timestep $\delta t \to \frac{\delta t}{2}$ is used for practical reasons [7]. In order to fulfill the structure preservation as well as speed up the computation significantly, the Lanczos method as described in [11] and the Suzuki-Trotter expansion which is also a symplectic integrator and has an error of the order $\mathcal{O}(T\delta t^{2p})$ are used for an elapsed time simulation $T = n\delta t$ with n timesteps.

2.4.4 Practical Computational Considerations

TDVP can be a very computationally expensive algorithm due to all of the tensor contractions required. It is absolutely essential to initialize each sweep properly. First, each step must be in site canonical form or the above algorithms do not hold. Every site in the overall algorithm shown in Fig. 2.12 and 2.10 would otherwise be required to be contracted. Next, before a sweep occurs, one should calculate all the possible right environments. When starting at site 1, all the other sites from L to 2 should be contracted to create its right environment. This is then stored as well as all the other right environments for the next steps of the sweep i.e. L to 3 and all the way up to L to $L-1$. Then, the left environments are continuously updated in a similar fashion while dropping the right environments when they are no longer needed. Once a sweep left to right is completed, the new right environments are initialized using the previous left environments and the algorithm is simply reversed. In the code used in this thesis, right to left sweeps were avoided by simply flipping the tensor network each time and updating the algorithms accordingly.

3 Tensor Network Quantum Jump Method

The quantum jump method, also known as the Monte Carlo Wavefunction Method (MCWF) or stochastic Schrödinger equation, is a method for simulating open quantum systems, in particular, by approximating the Lindblad master equation through averaging quantum trajectories [14]. A quantum trajectory can be understood as the path a quantum system takes through Hilbert space with some stochastic processes added to it. By averaging many trajectories, it is possible to approximate various physical quantities, in this case, the state resulting from evolution according to the master equation. This process is created by stochastically applying jump operators representing noise processes. These jumps are described as sudden, severe changes to the system such as for example instantaneous relaxation from an excited state $|1\rangle$ to $|0\rangle$. The motivation for using this method is that it is not necessary to store the full density matrix describing the system. A system with local physical dimension $d \in \mathbb{Z}$ and number of sites $L \in \mathbb{Z}$ exists in a Hilbert space of the dimensions $\mathcal{H} = \mathbb{C}^{d^L}$. The density matrix thus has dimensions $\rho \in \mathbb{C}^{d^L \times d^L}$. The quantum jump method permits performing all operations on the state vector $\psi \in \mathbb{C}^{d^L}$ such that saving d^{2L} complex values is no longer needed, but rather only d^L. This results in an obvious advantage in memory storage but also an advantage in computation time for the master equation which will be explored later in Sec. 3.3. The goal of this method is to calculate the time evolution of the state vector up to some elapsed time T with timesteps δt with stochastic application of jump operators. After introducing what will now be referred to as the original quantum jump method or the vector jump method, the tensor network jump method created during this thesis will be presented and its capabilities benchmarked.



(a) Averaging over 5 trajectories (b) Averaging over 1000 trajectories

Figure 3.1: Example of averaging over trajectories for building intuition

3.1 Quantum Jump Method

3.1.1 Initialization

The Lindblad master equation as derived in Sec. 2.1.2 is again defined as

$$\dot{\rho} = -\frac{i}{\hbar}[H_0, \rho] - \sum_m \gamma_m (L_m \rho L_m^\dagger - \frac{1}{2}\{L_m^\dagger L_m, \rho\}) \tag{3.1}$$

where the Hamiltonian H_0 is a self-adjoint operator on the Hilbert space $H_0 \in L(\mathcal{H})$, $\{L_m\}_m \in L(\mathcal{H})$ is a set of jump operators corresponding to possible jumps in the system, $\{\gamma_m\}_m \in \mathbb{R}$ are coupling factors corresponding to the strength of the noise i.e. related to the likelihood of the jumps, and $\{\cdot, \cdot\}$ is the anti-commutator.

A non-Hermitian effective Hamiltonian is constructed using the set of jump operators and their coupling factors where

$$H = H_0 - \frac{i\hbar}{2}\sum_m \gamma_m L_m^\dagger L_m = H_0 + H_{nH} \tag{3.2}$$

with the initial Hermitian part H_0 and non-Hermitian part defined by

$$H_{nH} = -\frac{i\hbar}{2}\sum_m \gamma_m L_m^\dagger L_m \tag{3.3}$$

For clarity, these jump operators are all possible jumps for the system. For example, consider relaxation from the excited state to ground state described by the single site jump operator $\sigma_- = |0\rangle\langle 1| = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$. Assuming that all L sites are equivalent i.e. each site has the possibility of this jump occurring, then the set of jump operators would be

$\{\sigma_-^{[\ell]}\}_{\ell=1}^L$ where each operator corresponds to the relaxation of site l such that

$$\sigma_-^{[\ell]} = \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{\text{sites } [1,l-1]} \otimes \underbrace{\sigma_-}_{l^{\text{th}} \text{ site}} \otimes \underbrace{\mathbb{1} \otimes \cdots \otimes \mathbb{1}}_{\text{sites } [l+1,L]} \tag{3.4}$$

Likewise, for other noise processes such as excitation, dephasing, etc., the jump operators are constructed similarly for multi-partite systems.

3.1.2 Non-Unitary Time-Evolution

Once the effective Hamiltonian is initialized, the time-evolution of a state $|\psi(t)\rangle \in \mathcal{H}$ is performed by constructing the discretized time-evolution operator

$$U(\delta t) = e^{-\frac{i}{\hbar}H\delta t} = \mathbb{1} - \frac{i}{\hbar}H\delta t + \mathcal{O}(\delta t^2) \tag{3.5}$$

such that $U(\delta t)\,|\psi(t)\rangle = |\psi^{(i)}(t+\delta t)\rangle$ with the superscript (i) denoting this as the initial time-evolved state before adjusting it stochastically. Numerically, it is best to avoid the exponentiation of a matrix so the Lanczos method [11] is used in this algorithm. It is also important to note that this operator is not unitary since H is purposely non-Hermitian and therefore not norm-preserving, however, this de-normalization is a key part of calculating when to apply jumps.

3.1.3 Calculating Jump Probabilities

After each timestep, the de-normalization is calculated or in terms of the method, the probability that a jump has occurred in the system is calculated. Intuitively, this would mean that a large de-normalization due to the non-unitary time-evolution would lead to a larger probability that a sudden change has occurred during that timestep. The probability of each possible jump in the system occurring is calculated by looking at the de-normalization as follows

$$\begin{aligned}
\langle\psi^{(i)}(t+\delta t)|\psi^{(i)}(t+\delta t)\rangle &= \langle\psi(t)|\,U^\dagger(t)U(t)\,|\psi(t)\rangle \\
&\approx \langle\psi(t)|\,(\mathbb{1} + \frac{i}{\hbar}H^\dagger\delta t)(\mathbb{1} - \frac{i}{\hbar}H\delta t)\,|\psi(t)\rangle \\
&= \langle\psi(t)|\psi(t)\rangle - \delta t\frac{i}{\hbar}\,\langle\psi(t)|\,H - H^\dagger\,|\psi(t)\rangle \\
&= 1 - \delta t\frac{i}{\hbar}\,\langle\psi(t)|\,(H_0 - \frac{i\hbar}{2}\sum_m \gamma_m L_m^\dagger L_m) \\
&\qquad - (H_0^\dagger + \frac{i\hbar}{2}\sum_m \gamma_m L_m^\dagger L_m)\,|\psi(t)\rangle \\
&= 1 - \delta t\sum_m \gamma_m\,\langle\psi(t)|\,L_m^\dagger L_m\,|\psi(t)\rangle \\
&= 1 - \sum_m \delta p_m \\
&= 1 - \delta p
\end{aligned} \tag{3.6}$$

This leads to a stochastic factor corresponding to each possible jump

$$\delta p_m = \delta t \gamma_m \langle \psi(t) | L_m^\dagger L_m | \psi(t) \rangle \tag{3.7}$$

which is defined by the size of the timestep δt, the coupling factor strength for that jump γ_m, and the norm of the jumped state $L_m |\psi(t)\rangle$. The overall stochastic factor determining if a jump has occurred is given by the sum of these individual terms such that

$$\delta p = \sum_m \delta p_m \tag{3.8}$$

Of these terms making up the stochastic factors, only the timestep size can be controlled, so it is important to adjust it such that $\delta p \ll 1$ as jumps are supposed to be a rare event when averaging trajectories [14].

3.1.4 Stochastic Application of Jumps

Once the jump probabilities have been calculated, the stochastic element can be added to this method by splitting the next step into one of two regimes. First, a random number ϵ is selected such that $\{\epsilon \in \mathbb{R} : 0 \leq \epsilon \leq 1\}$. This is then compared to the δp in order to make a decision after each timestep.

If $\epsilon \geq \delta p$, no jump occurred during this timestep and the time-evolved state is simply normalized before moving onto the next iteration.

$$|\psi(t + \delta t)\rangle = \frac{|\psi^{(i)}(t + \delta t)\rangle}{\sqrt{1 - \delta p}} \tag{3.9}$$

This should occur most of the time if δp is sufficiently small.

If $\epsilon < \delta p$, a quantum jump occurred within the previous timestep. Once this has been determined, a probability distribution of the terms $\{\delta p_m\}$ needs to be created by normalizing it against the total δp

$$\Pi_m = \frac{\delta p_m}{\delta p} \tag{3.10}$$

such that $\sum_m \Pi_m = 1$. A jump is selected according to this probability distribution and normalize the jumped state such that

$$|\psi(t + \delta t)\rangle = \frac{L_m |\psi(t)\rangle}{\sqrt{\frac{\delta p}{\gamma_m \delta t}}} \tag{3.11}$$

Note that this jump is applied to the state pre-time evolution from $t \to t + \delta t$ since the jump is determined to have occurred within that timestep. This methodology is then repeated until the desired elapsed time T is reached, resulting in a single quantum trajectory and a final state $|\psi(T)\rangle$.

3.1.5 Equivalence to Master Equation

If the above algorithm is performed N times i.e. forming N total trajectories, this can be shown to be equivalent to the master equation in the limit of $N \to \infty$. First, it is necessary to justify that δp is a reasonable stochastic factor to use. Say for N trajectories, jumps occur in N' trajectories. Since ϵ is compared against against δp, for many trajectories this leads to $\frac{N'}{N} = \frac{\delta p}{\max(\epsilon) - \min(\epsilon)} = \frac{\delta p}{1} = \delta p$. This justifies that δp is in fact the probability that a jump occurs in the system [14].

To show the equivalence to the time-evolution of a density matrix according to the master equation, the density matrix of a single trajectory $\mu_i(t) = |\psi_i(t)\rangle \langle\psi_i(t)|$ is considered and the average over N trajectories defined as

$$\overline{\mu}(t) = \frac{1}{N} \sum_{j=1}^{N} |\psi_i(t)\rangle \langle\psi_i(t)| \tag{3.12}$$

For $N \to \infty$, a time-evolved state is described by combining the possibilities of a jump occuring and not such that

$$\begin{aligned}
\overline{\mu}(t+\delta t) &= (1-\delta p) \frac{|\psi^{(i)}(t+\delta t)\rangle}{\sqrt{1-\delta p}} \frac{\langle\psi^{(i)}(t+\delta t)|}{\sqrt{1-\delta p}} \\
&\quad + \delta p \sum_m \frac{L_m |\psi(t)\rangle}{\sqrt{\frac{\delta p}{\gamma_m \delta t}}} \frac{\langle\psi(t)| L_m^\dagger}{\sqrt{\frac{\delta p}{\gamma_m \delta t}}} \\
&= (\mathbb{1} - \frac{i}{\hbar} H \delta t) |\psi(t)\rangle \langle\psi(t)| (\mathbb{1} + \frac{i}{\hbar} H^\dagger \delta t) \\
&\quad + \delta t \sum_m \gamma_m L_m |\psi(t)\rangle \langle\psi(t)| L_m^\dagger \\
&\approx |\psi(t)\rangle \langle\psi(t)| - \frac{i}{\hbar} H \delta t |\psi(t)\rangle \langle\psi(t)| + |\psi(t)\rangle \langle\psi(t)| \frac{i}{\hbar} H^\dagger \delta t \\
&\quad + \delta t \sum_m \gamma_m L_m |\psi(t)\rangle \langle\psi(t)| L_m^\dagger \\
&= \overline{\mu}(t) - \frac{i}{\hbar} H \delta t \overline{\mu}(t) + \overline{\mu}(t) \frac{i}{\hbar} H^\dagger \delta t \\
&\quad + \delta t \sum_m \gamma_m L_m \overline{\mu}(t) L_m^\dagger
\end{aligned} \tag{3.13}$$

Taking the derivative such that the LHS of the master equation is created

$$\begin{aligned}
\frac{d}{dt}\overline{\mu}(t) &\approx \frac{\overline{\mu}(t+\delta t)-\overline{\mu}(t)}{\delta t} \\
&= -\frac{i}{\hbar}H\overline{\mu}(t)+\overline{\mu}(t)\frac{i}{\hbar}H^{\dagger} \\
&\quad+\sum_m \gamma_m L_m\overline{\mu}(t)L_m^{\dagger} \\
&= -\frac{i}{\hbar}\left[(H_0-\frac{i\hbar}{2}\sum_m\gamma_m L_m^{\dagger}L_m)\overline{\mu}(t)+\overline{\mu}(t)(H_0+\frac{i\hbar}{2}\sum_m\gamma_m L_m^{\dagger}L_m)\right] \\
&\quad+\sum_m \gamma_m L_m\overline{\mu}(t)L_m^{\dagger} \\
&= -\frac{i}{\hbar}[H_0,\overline{\mu}(t)]-\sum_m\gamma_m(L_m\overline{\mu}(t)L_m^{\dagger}-\frac{1}{2}\{L_m^{\dagger}L_m,\overline{\mu}(t)\})
\end{aligned} \tag{3.14}$$

which has the form of the master equation such that for sufficiently many trajectories [14]

$$\rho = \lim_{N\to\infty}\frac{1}{N}\sum_{j=1}^{N}|\psi_j(t)\rangle\langle\psi_j(t)| \tag{3.15}$$

3.2 Tensor Network Quantum Jump Method

In order to implement the original method in tensor networks, the original quantum jump method is decomposed into an algorithm based mainly on sweeping across tensor sites and applying the different steps site-wise. For each trajectory, rather than using a state vector of length d^L for L sites, it is represented as an MPS of L tensors as presented previously in Sec. 2.2 and 2.3. This is a natural step forward for the jump method as it is expected to allow more sites in a simulation as well as give a more intuitive application of local noise processes. Tensor networks is known to scale linearly with sites when truncating the bond dimensions and the ability to keep local $d \times d$ jump operators rather than global $d^L \times d^L$ can speed up the computation significantly.

3.2.1 Adapting the Time Evolution

The core difference from the original quantum jump method is that Strang splitting [12] is applied to the effective Hamiltonian described in Eq.(3.2) such that a non-Hermitian Hamiltonian is not used in the TDVP evolution due to its potential numerical instability. The non-Hermitian Hamiltonian is first split into a product of all site-wise components such that it can be represented as a sweep across the tensor network

$$\begin{aligned} H_{nH} &= -\frac{i\hbar}{2}\sum_m \gamma_m L_m^\dagger L_m \\ &= -\frac{i\hbar}{2}\sum_\ell \sum_m \gamma_m^{[\ell]} L_m^{\dagger[\ell]} L_m^{[\ell]} \\ &= \sum_\ell H_{nH}^{[\ell]} \end{aligned} \tag{3.16}$$

where $[\ell]$ indicates a local application to site l. For example, if only relaxation is considered then the set of jump operators is $\{\sigma_-^{[\ell]}\}$ as defined in Eq.(3.4) for all sites $\{\ell \in \mathbb{Z} : 1 \leq \ell \leq L\}$.

This Strang splitting results in the following time-evolution operator comprised of three sweeping sub-algorithms

$$\begin{aligned} U(\delta t) &= e^{-\frac{i}{\hbar}H\delta t} \\ &= e^{-\frac{i}{\hbar}(H_0+H_{nH})\delta t} \\ &= e^{-\frac{i}{\hbar}H_{nH}\frac{\delta t}{2}} e^{-\frac{i}{\hbar}H_0\delta t} e^{-\frac{i}{\hbar}H_{nH}\frac{\delta t}{2}} + \mathcal{O}(\delta t^3) \\ &\approx \big(\prod_\ell e^{-\frac{i}{\hbar}H_{nH}^{[\ell]}\frac{\delta t}{2}}\big)\big(e^{-\frac{i}{\hbar}H_0\delta t}\big)\big(\prod_\ell e^{-\frac{i}{\hbar}H_{nH}^{[\ell]}\frac{\delta t}{2}}\big) \\ &= \mathrm{D}\big[\frac{\delta t}{2}\big]\ \mathrm{TDVP}\big[\delta t\big]\ \mathrm{D}\big[\frac{\delta t}{2}\big] \end{aligned} \tag{3.17}$$

where D corresponds to a sweep with the non-Hermitian, dissipative part of the Hamiltonian and TDVP the unitary time evolution. To show that the dissipation is indeed a sweep across all sites, this can be further broken down into a product of all evolutions according to all possible jumps on that site by inserting its definition from Eq.(3.3)

$$\begin{aligned} e^{-\frac{i}{\hbar}H_{nH}^{[\ell]}\frac{\delta t}{2}} &= e^{-\frac{i}{\hbar}\left(-\frac{i\hbar}{2}\sum_m \gamma_m^{[\ell]} L_m^{\dagger[\ell]} L_m^{[\ell]}\right)\frac{\delta t}{2}} \\ &= \prod_m e^{-\frac{1}{2}\gamma_m^{[\ell]} L_m^{\dagger[\ell]} L_m^{[\ell]}\frac{\delta t}{2}} \\ &= \prod_m \mathrm{D}_m^{[\ell]}\big[\frac{\delta t}{2}\big] \end{aligned} \tag{3.18}$$

This puts the operator $U(\delta t)$ in the form of a half timestep dissipation D sweep across all sites followed by a full timestep Hermitian TDVP evolution of two sweeps and finally another half timestep dissipation sweep. Reminder here that TDVP is made of a left-right half-timestep sweep followed by a right-left half-timestep sweep as shown in Sec. 2.4.3.

The Strang splitting was selected as it benefits from a timestep error of $\mathcal{O}(\delta t^2)$ compared to the first-order Baker-Campbell-Hausdorff (BCH) splitting with $\mathcal{O}(\delta t)$ i.e. for the BCH formula [12]

$$e^{(A+B)\delta t} = e^{A\delta t}e^{B\delta t} + \mathcal{O}(\delta t) \tag{3.19}$$

Algorithm 7 Dissipative Sweep

```
function D($|\Psi\rangle$, $\delta t$, $\{L_m\}$, $\{\gamma_m\}$)
    for $\ell \in [1, L]$ do                                          ▷ Sweep sites
        $M_\ell = |\Psi^{[\ell]}\rangle$                                   ▷ $\ell$th site Matrix of MPS $|\Psi\rangle$
        for $L_m^{[\ell]} \in \{L_m^{[\ell]}\}$ do                          ▷ Loop over all jump operators at site
            $D_m^{[\ell]}\left[\frac{\delta t}{2}\right] = e^{-\frac{1}{2}\gamma_m^{[\ell]} L_m^{\dagger[\ell]} L_m^{[\ell]} \frac{\delta t}{2}}$      ▷ Lanczos for matrix exponential
            $M'_\ell = \text{Contract}\big(D_m^{[\ell]}\left[\frac{\delta t}{2}\right], M_\ell \text{ [physical], [physical] }\big)$   ▷ Apply dissipation
        end for
        ShiftOrthogonalityCenterRight$\big(|\Psi\rangle, \ell\big)$        ▷ Keeps MPS in canonical form
    end for
    return $|\Psi'\rangle$
end function
```

and the Strang splitting

$$e^{(A+B)\delta t} = e^{A\frac{\delta t}{2}} e^{B\delta t} e^{A\frac{\delta t}{2}} + \mathcal{O}(\delta t^2) \tag{3.20}$$

For n timesteps or an elapsed time $T = n\delta t$ however, this decreases the error at the expense of significantly increasing the computational time due to the extra term. For $A = H_{nH}$ describing a dissipative sweep and $B = H_0$ a TDVP unitary time evolution sweep, the ratio between sweeps in the first-order BCH and Strang splitting for n timesteps is given by

$$\frac{\text{Strang sweeps}}{\text{BCH sweeps}} = \frac{4n}{3n} = 1.333\ldots \tag{3.21}$$

or 33 % more sweeps. Note that this is not directly equivalent to 33 % more computational time as TDVP sweeps are more expensive than dissipative sweeps.

The form of the Strang splitting can be used to reduce this significantly and still gain the reduction in timestep error.

For intuition, extending the Strang splitting to two timesteps results in

$$\begin{aligned} e^{(A+B)2\delta t} &= (e^{A\frac{\delta t}{2}} e^{B\delta t} e^{A\frac{\delta t}{2}})(e^{A\frac{\delta t}{2}} e^{B\delta t} e^{A\frac{\delta t}{2}}) + \mathcal{O}(\delta t^2) \\ &= e^{A\frac{\delta t}{2}} e^{B\delta t} (e^{A\frac{\delta t}{2}} e^{A\frac{\delta t}{2}}) e^{B\delta t} e^{A\frac{\delta t}{2}} + \mathcal{O}(\delta t^2) \\ &= e^{A\frac{\delta t}{2}} e^{B\delta t} e^{A\delta t} e^{B\delta t} e^{A\frac{\delta t}{2}} + \mathcal{O}(\delta t^2) \end{aligned} \tag{3.22}$$

and further for n timesteps or a total elapsed time $T = n\delta t$

$$e^{(A+B)T} = e^{A\frac{\delta t}{2}} e^{B\delta t} \big(e^{A\delta t} e^{B\delta t}\big)^{n-1} e^{A\frac{\delta t}{2}} + \mathcal{O}(\delta t^2) \tag{3.23}$$

This forms results in the following ratio

$$\frac{\text{Strang sweeps}}{\text{BCH sweeps}} = \frac{4 + 3(n-1)}{3n} = \frac{3n+1}{3n} = 1 + \frac{1}{3n} \xrightarrow[n\to\infty]{} 1 \tag{3.24}$$

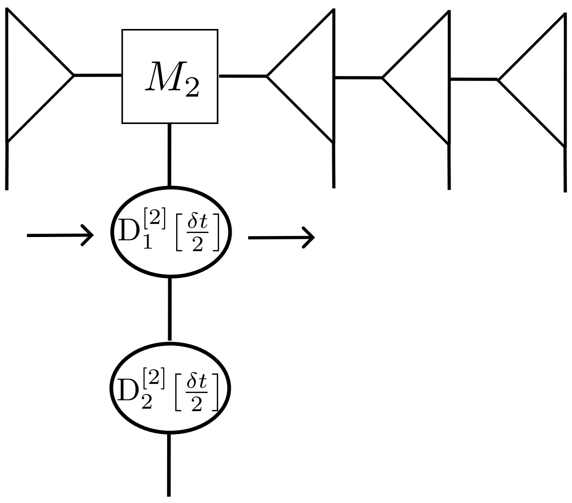

M_2
$D_1^{[2]}[\frac{\delta t}{2}]$
$D_2^{[2]}[\frac{\delta t}{2}]$

Figure 3.2: Dissipative sweep where all jump operators associated with a site are contracted into it using the Lanczos method. Site canonical form is used so that the state is prepared for TDVP once the sweep is finished. Described in Algorithm 7.

Therefore, there is effectively no increase in computational time from the first-order method except a single additional sweep.

Using the identity in Eq.(3.23) and setting $A = H_{nH}$ and $B = H_0$, there is the following chain for the time evolution where each term corresponds to a dissipative or TDVP sweep

$$U(T) = \mathrm{D}[\frac{\delta t}{2}]\ \mathrm{TDVP}[\delta t]\ \Big(\mathrm{D}[\delta t]\ \mathrm{TDVP}[\delta t]\Big)^{n-1}\ \mathrm{D}[\frac{\delta t}{2}] \tag{3.25}$$

In order to prepare the MPS for the TDVP sweep as well as the jump application sweep in the next section, the MPS is continuously kept in site canonical form. This means after applying a dissipative time evolution to a site, the orthogonality center is shifted to maintain the site canonical form. TDVP does not implicitly require a canonical form, but this significantly reduces the contractions needed as all others sites can be ignored. This is the method described previously in Algorithm 3 and Fig. 2.5.

3.2.2 Adapting the Jump Application

Following each timestep, the jump probabilities need to be calculated and a jump operator applied. This step is done at the following point in the evolution shown by the vertical arrows (note that the operator application is from right to left):

$$U(T) = \mathrm{D}[\frac{\delta t}{2}]\ \mathrm{TDVP}\ [\delta t]\ \Big(\uparrow \mathrm{D}[\delta t]\ \mathrm{TDVP}[\delta t]\ \Big)^{n-1}\ \uparrow \mathrm{D}[\frac{\delta t}{2}] \tag{3.26}$$

This means jumps are not applied directly on a timestep due to the first term. The stochastic nature of ths algorithm means there is no effect on the accuracy since $\delta t \approx$

$\frac{\delta t}{2}$ for sufficiently small timesteps. When applying this sequence of operators, it is necessary to simultaneously be able to continue applying jumps at the proper time while also sampling the result on the timestep rather than at a shifted timestep.

A solution to this is to use a stochastic MPS that is continuously iterated over, which is denoted Φ, and the actual resulting MPS Ψ sampled from the stochastic MPS which can be used for calculations such as the expectation value. Functionally what this means is that the final operators $\mathrm{D}\big[\frac{\delta t}{2}\big]\mathrm{TDVP}\ \big[\delta t\big]$ are only applied when sampling such that

$$|\Psi(n\delta t)\rangle = \mathrm{D}\Big[\frac{\delta t}{2}\Big]\ \mathrm{TDVP}\ \big[\delta t\big]\ |\Phi(n\delta t)\rangle \tag{3.27}$$

and

$$|\Phi(n\delta t)\rangle = \Big(\uparrow \mathrm{D}\big[\delta t\big]\ \mathrm{TDVP}\big[\delta t\big]\ \Big)^{n-1} \uparrow \mathrm{D}\Big[\frac{\delta t}{2}\Big]\ |\Phi(0)\rangle \tag{3.28}$$

This can be seen in Algorithm 9.

When calculating the jump probabilities, the terms are calculated by

$$\delta p_m = \delta t \gamma_m \langle\Phi(t)|\, L_m^\dagger L_m\, |\Phi(t)\rangle \tag{3.29}$$

in a sweep as done in Sec. 3.1. In order to do this, at each step, a single jump operator is applied at site l by contracting it into a copy of the MPS to create $L_m^{[\ell]}\,|\Phi(t)\rangle$. The scalar product of this state is then used to calculate its corresponding δp_m. This is then repeated for all operators on that site before moving to the next site in the sweep. Once the sweep is finished, all the probabilities have been calculated just as done in the vector jump method. Further details can be found in Algorithm 8.

As done previously, whether or not a jump occurs is stochastically selected. If it has not, the state is normalized. If it has, the jump is applied by contracting the jump operator L_m into that site and then normalizing. The normalization step in tensor networks is not as straightforward as simply dividing by a constant such as in the state vector, so one last sweep is performed in this timestep iteration as described in Sec. 2.3.1. Therefore, once this step is complete, one last sweep from right-left is done such that the MPS is in right-canonical form (or site-canonical form at site 1) and ready to perform the next timestep iteration.

Algorithm 8 Applying Stochastic Jumps

```
function CalculateStochasticTerms($|\Phi\rangle$, $\delta t$, $\{L_m\}$, $\{\gamma_m\}$)
    for $\ell \in [1, L]$ do                                         ▷ Sweep sites
        $M_\ell = |\Phi^{[\ell]}\rangle$                              ▷ $\ell$th site Matrix of MPS $|\Phi\rangle$
        for $L_m^{[\ell]} \in \{L_m^{[\ell]}\}$ do          ▷ Loop over all jump operators at site
            $M'_\ell = \text{Contract}\big(L_m^{[\ell]}, M_\ell\big)$           ▷ Create jumped state
            $\delta p_m = \delta t \gamma_m^{[\ell]} \text{Contract}\big(\overline{M}'_\ell, M'_\ell\big)$
            ShiftOrthogonalityCenterRight$\big(|\Phi\rangle, \ell\big)$    ▷ Keeps MPS in canonical form
        end for
    end for
    $\delta p = \sum_m \delta p_m$
    return $\{\delta p_m\}, \delta p$
end function                                   ▷ Do not jump MPS itself in this function

function CreateProbabilityDistribution($\{\delta p_m\}, \delta p$)
    for $\delta p_m \in \{\delta p_m\}$ do
        $\Pi_m = \frac{\delta p_m}{\delta p}$          ▷ Normalize to create probability distribution
    end for
    return $\{\Pi_m\}$
end function

function ApplyJumps($|\Phi(t)\rangle, |\Psi^{(i)}(t+\delta t)\rangle, \{L_m\}, \{\gamma_m\}, \delta t$)
    $\{\delta p_m\}, \delta p$ = CalculateStochasticTerms $\big(|\Phi(t)\rangle, \delta t, \{L_m\}, \{\gamma_m\}\big)$
    Select random $\epsilon$ where $\{\epsilon \in \mathbb{R} : 0 \leq \epsilon \leq 1\}$
    if $\epsilon \leq \delta p$ then
        $\{\Pi_m\} = \text{CreateProbabilityDistribution}\big(\{\delta p_m\}, \delta p\big)$
        Randomly select $m$ based on probability $\Pi_m$
        $m \to L_m =: L_m^{[\ell]}$                        ▷ Selected jump is related to some site
        $M_\ell = |\Phi^{[\ell]}(t)\rangle$                ▷ The MPS is jumped at the previous time
        $M'_\ell = \text{Contract}\big(L_m^{[\ell]}, M_\ell\big)$
        $|\Phi^{[\ell]}(t+\delta t)\rangle \leftarrow M'_\ell$  ▷ No need for canonical form since it is normalized next
    else
        $|\Phi(t+\delta t)\rangle \leftarrow |\Psi^{(i)}(t+\delta t)\rangle$          ▷ Time-evolved state without jump
    end if
    $|\Phi(t+\delta t)\rangle = \text{Normalize}\ \big(|\Phi(t+\delta t)\rangle\big)$
    return $|\Phi(t+\delta t)\rangle$
end function
```

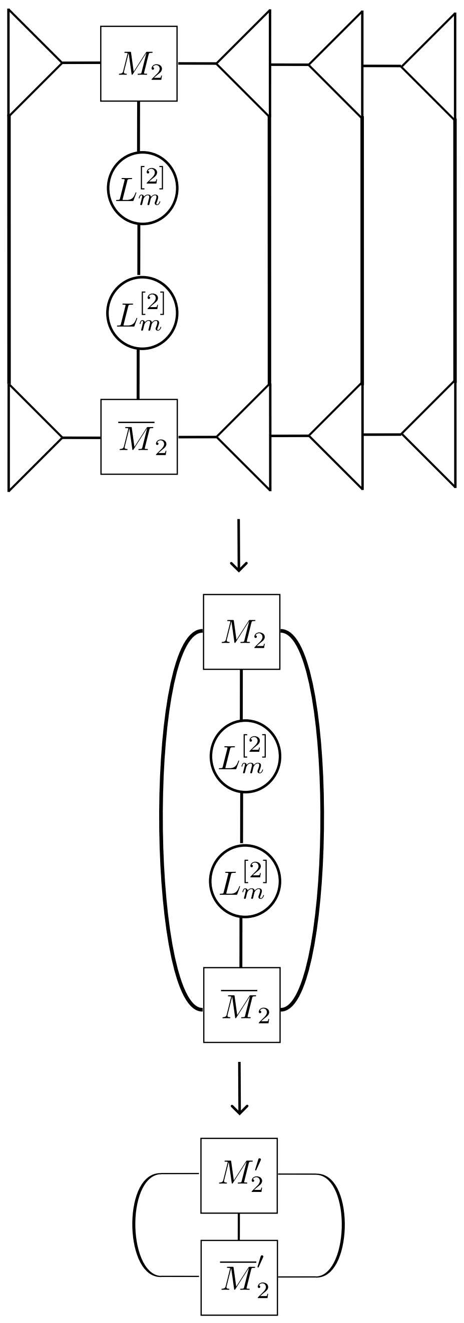


Figure 3.3: Sweep for calculating stochastic terms δp_m by contracting the jump operator into its given site in site canonical form. This is equivalent to a scalar product of the jumped state or an expectation value of $(L_m^{[\ell]})^2$. Described in Algorithm 8.

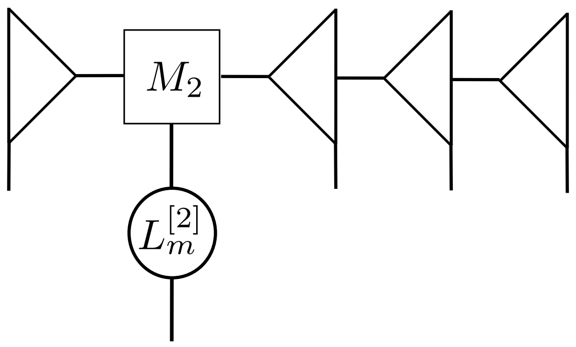


Figure 3.4: Application of selected jump to state. Described in Algorithm 8.

Algorithm 9 Core Time Evolution Algorithm

```
function CoreAlgorithm($|\Phi(t)\rangle$, $H$, $\delta t$, timestep, $n$, $\{L_m\}$, $\{\gamma_m\}$)
    if timestep = 1 or $n = 1$ then                              ▷ $i^{\text{th}}$ timestep, $n$ total timesteps
        $|\Psi^{(i)}(t+\delta t)\rangle = \mathrm{D}(|\Phi\rangle, \frac{\delta t}{2}, \{L_m\}, \{\gamma_m\})$      ▷ Non-unitary dissipative sweep
    else
        $|\Psi'\rangle = \mathrm{TDVP}\big(|\Phi(t)\rangle, H, \delta t\big)$
        $|\Psi^{(i)}(t+\delta t)\rangle = \mathrm{D}(|\Psi'\rangle, \frac{\delta t}{2}, \{L_m\}, \{\gamma_m\})$
    end if
    $|\Phi(t+\delta t)\rangle = \mathrm{ApplyJumps}\ \big(|\Phi(t)\rangle, |\Psi^{(i)}(t+\delta t)\rangle, \{L_m\}, \{\gamma_m\}, \delta t\big)$

    $|\Psi'\rangle = \mathrm{TDVP}\big(|\Phi(t+\delta t)\rangle, H, \delta t\big)$
    $|\Psi''\rangle = \mathrm{D}(|\Psi'\rangle, \frac{\delta t}{2}, \{L_m\}, \{\gamma_m\})$
    $|\Psi(t+\delta t)\rangle = \mathrm{Normalize}\ \big(|\Psi''\rangle\big)$

    return $|\Psi(t+\delta t)\rangle$, $|\Phi(t+\delta t)\rangle$                ▷ Actual MPS $\Psi$, Stochastic MPS $\Phi$
end function
```

Algorithm 10 Tensor Network Jump Method

```
procedure TN JUMP($|\Psi(0)\rangle$, $T$, $n$, $N$)
    $\delta t = \frac{T}{n}$                                              ▷ n total timesteps
    for trajectory $\in [1, N]$ do
        for timestep $\in [1, n]$ do
            $|\Psi(t+\delta t)\rangle$, $|\Phi(t+\delta t)\rangle$ = CoreAlgorithm $\big(\ |\Phi(t)\rangle$, $\delta t$, timestep, $n$,
                                                        $\{L_m\}$, $\{\gamma_m\}\ \big)$
            SiteCanonicalForm $\big(|\Psi(t+\delta t)\rangle, \ell\big)$                ▷ $\ell$ site where operator is
            $\langle\hat{O}(t+\delta t)\rangle = \langle\Psi(t+\delta t)|\,\hat{O}\,|\Psi(t+\delta t)\rangle$        ▷ Running average calculated
            Avg$\langle\hat{O}(t+\delta t)\rangle$ = (trajectory $-$ 1)$\langle\hat{O}(t+\delta t)\rangle + \frac{1}{\text{trajectory}}\langle\hat{O}(t+\delta t)\rangle$
            SiteCanonicalForm $\big(|\Psi(t+\delta t)\rangle, \ell = 0\big)$
        end for
    end for
end procedure
```

3.3 Benchmarking

There are several important benchmarks considered for this method in order to test its overall behavior and stability. The Ising model $H = -J\sum_j \sigma_j^z \sigma_{j+1}^z - g\sum_j \sigma_j^x$ is used with $J = 1$ and $g = 0.5$ for all tests in this section as well as a starting state $|+\cdots+\rangle$ corresponding to the x-basis. For the tests, two noise processes, relaxation and dephasing, are considered which are represented by the operators σ_- and σ_z respectively. Unless specified otherwise, this system is simulated up to a normalized time $T = 1$. Many of the tests in this section are benchmarking the ability to simulate various timescales relative to the strength of the noise i.e. magnitude of γ. For all tests, the coupling factors are set such that they occur on a similar timescale with $\gamma := \gamma_{\text{relaxation}} = \gamma_{\text{dephasing}}$.

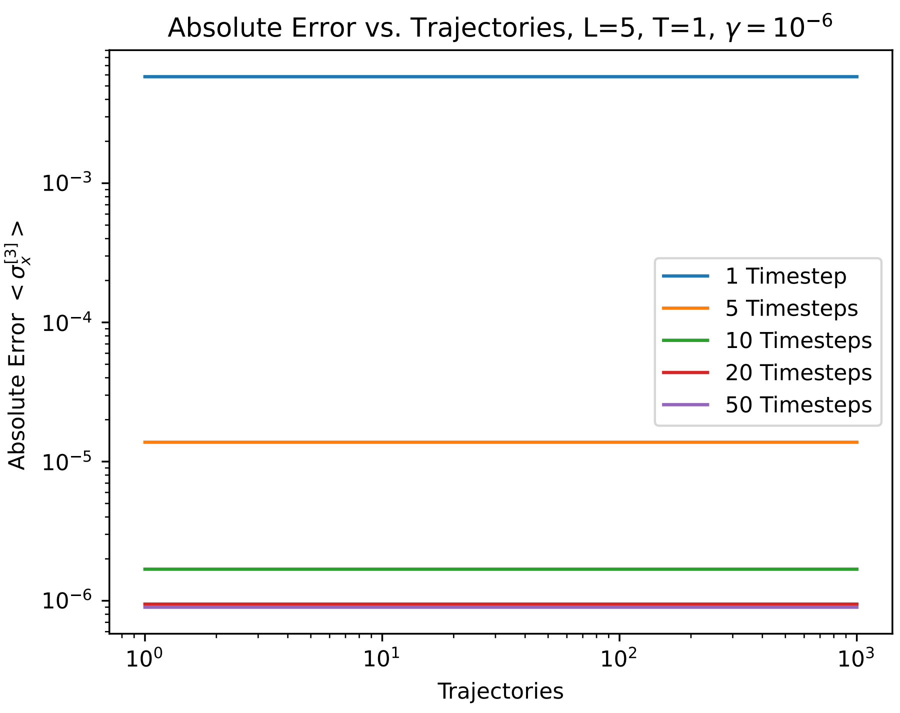
3.3.1 Trajectory Convergence (Noise Regimes)

The first test of the tensor network jump method is to analyze its convergence as a function of trajectories. For this test, it is compared to a small system with $L = 5$ such that it is still possible to calculate the exact solution using the superoperator. The error is then the absolute difference between this exact solution and the result of the tensor network jump method. This size system was chosen as it is the limit of what can be calculated in a reasonable timeframe using the superoperator. The MPS is at full-rank with a max bond dimension $\chi = 4$. The expectation value of an operator $\langle \sigma_x^{[3]} \rangle$ at the center of the chain at site $\ell = 3$ is calculated at each timestep.

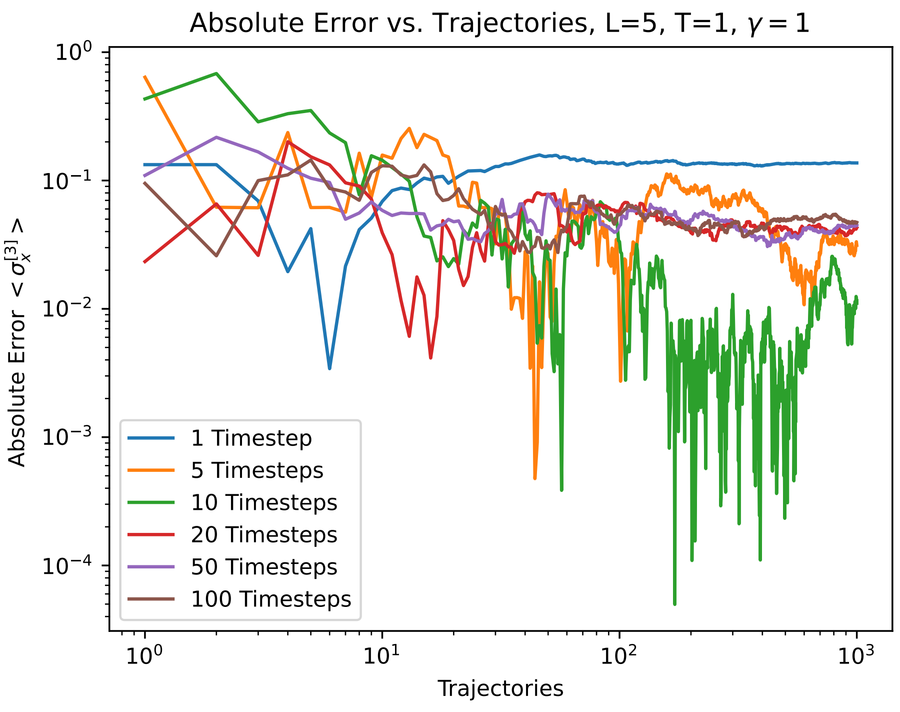
This test has been performed performed for three noise regimes, weak noise $\gamma \ll T^{-1}$, medium noise $\gamma \approx T^{-1}$, and strong noise $\gamma \gg T^{-1}$. The results of this can be seen in Fig. 3.5.

In the weak noise regime where the noise is much less than the inverse of the simulation time, there is convergence with a single trajectory regardless of the timestep size, however the overall error is reduced with more timesteps, converging around 20-50 timesteps at $\epsilon \approx 10^{-6}$. This regime approaches a noise-less simulation since $\gamma \to 0$.

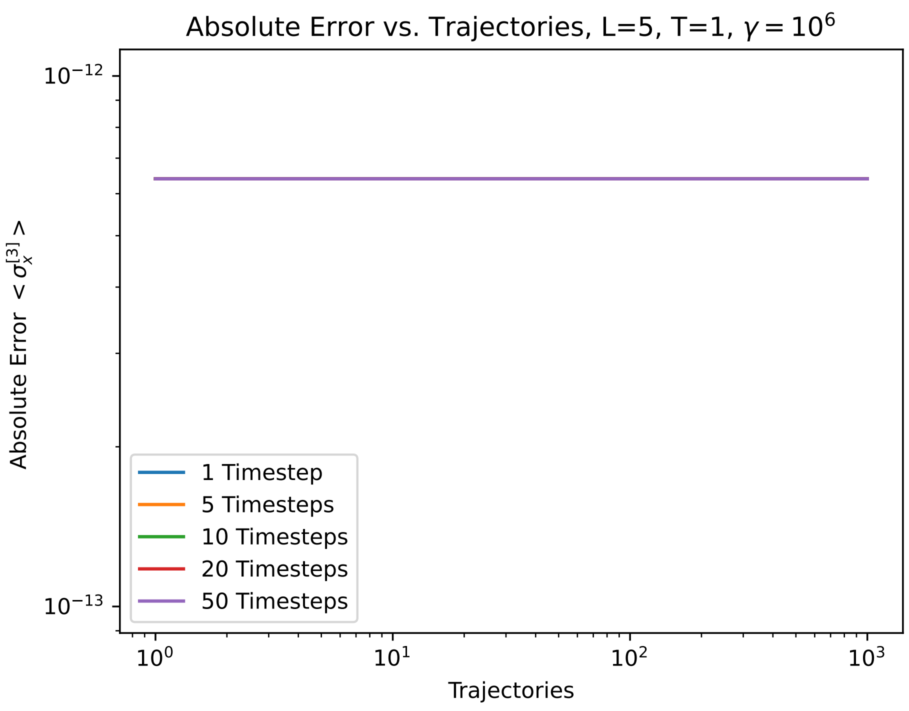
In the regime where the noise is approximately on par with the inverse of the simulation time, there is more traditional convergence for a stochastic simulation. As is standard with Monte Carlo methods, the error ϵ should converge with more trajectories according to $\epsilon \sim \frac{1}{\sqrt{N}}$ by the central limit theorem. All timesteps practically converge around 100 total trajectories according to this test. For 20 timesteps or more, the simulation converges to roughly the same error. For higher timesteps such as 100, it simply converges to this with less total trajectories. There is an anomaly in that 10 timesteps performs best and follows the linear slope that would be expected for $\epsilon \sim \frac{1}{\sqrt{N}}$ on a log-log plot. As there is generally the correct linear slope for less than 10 trajectories, it is possible that this shows an over-convergence or convergence to some base error rate. This should be investigated further. In the strong noise regime, the simulation immediately converges with very little error regardless of the number of timesteps. This can be interpreted to mean that the noise simply takes over before any real dynamics can occur.



(a) Weak noise $\gamma = 10^{-6} \ll T^{-1}$



(b) Medium Noise $\gamma = 1 \approx T^{-1}$



(c) Strong noise $\gamma = 10^6 \gg T^{-1}$

Figure 3.5: Error relative to number of trajectories for weak, medium, and strong noise regimes

These results indicate a need to analyze the error over a larger range of noise strengths with a fixed number of trajectories $N = 100$, but otherwise with the same parameters as in the previous test. The results of this are found in Fig. 3.6.

At the limits of this graph, the asymptotic regions plotted in the previous test can be seen. There is a low error, converging weak noise region from $\gamma T = 10^{-6}$ to $\gamma T = 10^{-1}$. Between $\gamma T = 10^{-1}$ and $\gamma T = 10^3$, there is the medium noise regime, or transition regime, where the algorithm generally sees convergence for all timesteps, but with greater error. As this test was done with 100 trajectories, there is oscillation in the results, but it is expected that this graph would smooth out for $N \to \infty$. Overall, more timesteps does indeed decrease the error. Finally, there is an abrupt anomaly in the cutoff between the transition regime and strong noise around $\gamma T = 10^3$. The low error, strong noise regime begins earlier for fewer timesteps. This may simply be an artifact of the method since the probability of jumps is directly proportional to the magnitude of the timestep such that a jump is more likely and thus causing it to converge quicker. All timesteps converge to the same error for higher noise as seen in the previous test. These regimes are summarized along with other results in Table 3.1.

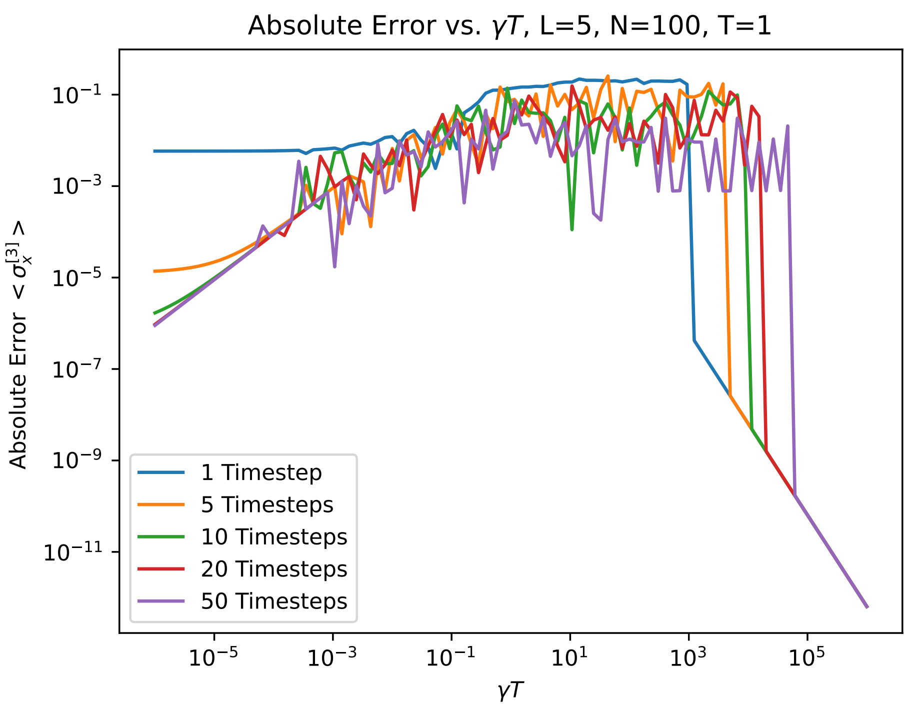


Figure 3.6: Error as a function of noise strength γ and simulation time T

3.3.2 Bond Dimension Convergence (Entanglement Regimes)

Now that it has been shown that the method converges with a sufficient number of timesteps and trajectories for $L = 5$ and varying γ, the system size can be extended past an exactly calculable system in order to take advantage of the usage of tensor networks. For the following tests, the same Ising model as presented previously is used, but extended to $L = 16$. Since the Lindblad equation requires a density matrix and/or superoperator to solve, this is not comparable to how many sites exact diagonalization without noise processes can calculate which can typically do further than 16. For this many sites, a max bond dimension $\chi_{\max} = 256$ is needed to exactly represent the state. It is necessary for calculation speed to reduce this as much as possible without losing too much information about the entanglement. In order to do this, the entanglement entropy around the center bond is considered. This is calculated by contracting the center sites at this bond and applying an SVD i.e.

$$\Theta^{a_{l-1},\sigma_l,a_{l+1},\sigma_{l+1}} = \sum_{a_\ell}^{\chi_\ell} M_\ell^{a_{\ell-1},a_\ell,\sigma_\ell} M_{\ell+1}^{a_\ell,a_{\ell+1},\sigma_{\ell+1}} \tag{3.30}$$

The corresponding physical dimension and bond are combined such that they can be decomposed as a matrix

$$\Theta^{a_{l-1},\sigma_l,a_{l+1},\sigma_{l+1}} \rightarrow \Theta^{(a_{l-1},\sigma_l),(a_{l+1},\sigma_{l+1})} \tag{3.31}$$

The SVD is then applied, but only the singular values $\{s_i\}$ are needed and are truncated against machine error. These are normalized such that $\sum_i s_i^2 = 1$ and the entanglement entropy is calculated with

$$S = -\sum_i s_i^2 \log(s_i^2) \tag{3.32}$$

The entanglement is calculated over a range of noise strengths as well as for different timesteps with 100 trajectories and a max bond dimension $\chi_{\text{max}} = 16$. More trajectories are used to stabilize any oscillations in the values. The results of this can be seen in Fig. 3.7.

The entanglement is shown to be reduced for more timesteps. This means that for more timesteps in the simulation, a lower bond dimension can be used. There are also three distinct entanglement regimes which will be referrred to as the strong entanglement regime, weak entanglement regime, and product state regime. These essentially correlate with the noise regimes seen previously.

There is an abrupt cut between the weak entanglement and product state regimes which signifies that a strong enough noise destroys the entanglement. However, this exact point differs with the number of timesteps in the simulation as it enters the product state regime at a lower noise strength with less timesteps. This may be a result of how often jumps are applied as the stochastic factor $\delta p \propto \delta t$, but further analysis is necessary. This coincides with the results seen in the trajectory test in Fig. 3.6 and is summarized later in Table 3.1.

It can also be observed that longer simulation times cause an increase in entanglement which requires a larger bond dimension to capture the dynamics. Although this is expected, these longer times also converge with noise. This means that with stronger noise, not only can the dynamics be simulated with a small bond dimension, but also for a longer period of time without needing to allow it to grow. This can be seen in Fig. 3.8.

All benchmarks up to this point have been for an elapsed time $T = 1$ which is equivalent to $T = \frac{1}{J}$ since the interaction parameter $J = 1$. Next, the convergence of the changing max bond dimension is analyzed in the three noise regimes presented earlier but up to a larger time $T = 5$ (or $T = \frac{5}{J}$). The expectation value is sampled for each timestep resulting in 100 total points over 100 trajectories. The results of this can be seen in Fig. 3.9.

It can be noted that in the weak noise regime, the bond dimension necessary to simulate the dynamics is roughly $\chi_{\text{max}} = 2^{tJ}$. A time $t = \frac{1}{J}$ requires $\chi_{\text{max}} = 2$, $t = \frac{2}{J}$ requires $\chi_{\text{max}} = 4$, etc.

The transition regime sees convergence for all bond dimensions greater than 1. In this region, it is possible to reasonably simulate the dynamics with $\chi_{\text{max}} = 2$. It is interpreted as the noise destroying enough entanglement to allow the system to be easily simulated. Luckily, most of the applications would also likely be in this range especially if simulating the dynamics during the noise process itself which would be a simulation time of roughly $\gamma T \approx 1$. Finally, the strong noise regime sees a convergence

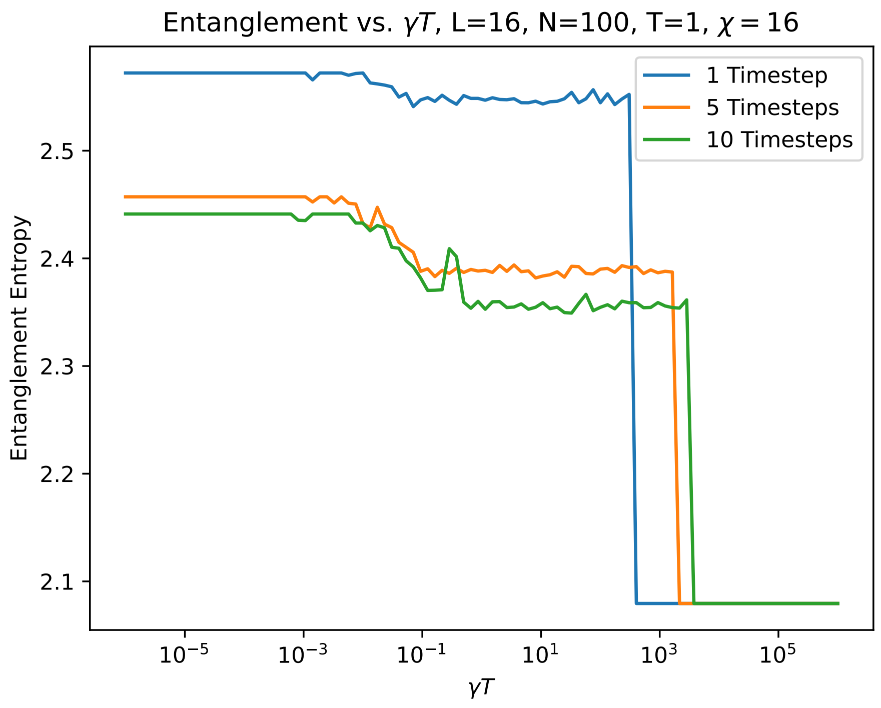


Figure 3.7: Entanglement as a function of noise strength γ and simulation time T for varying total timesteps

of all bond dimensions such that it continuously remains in a product state with $\chi_{\text{max}} = 1$.

Since the bond dimensions represent entanglement within the system and its growth, some observations can be made about the effect noise has on the entanglement. The weak noise region requires the largest bond dimension as this becomes closest to the noise-less simulation. Physically, this means weak noise does not inhibit entanglement growth in the system.

In the medium noise regime, the values start converging for $\chi_{\text{max}} = 2$. This means that there is entanglement, but the dynamics do not require a large bond dimension to be represented. In this regime, the noise starts to have an effect on the dynamics of the system such that it can inhibit entanglement growth, however it does not entirely destroy entanglement within the system. As an addendum, this does sometimes already see convergence with $\chi_{\text{max}} = 1$ if there are many timesteps (100-1000).

Lastly, for strong noise, it is possible to exactly represent the dynamics of the state for $\chi_{\text{max}} = 1$ showing that the system is always in a product state with no entanglement. This can be interpreted as the noise acting on the system so strongly that it destroys the entanglement.

The results of these benchmarks can be reasonably summarized into several simu-

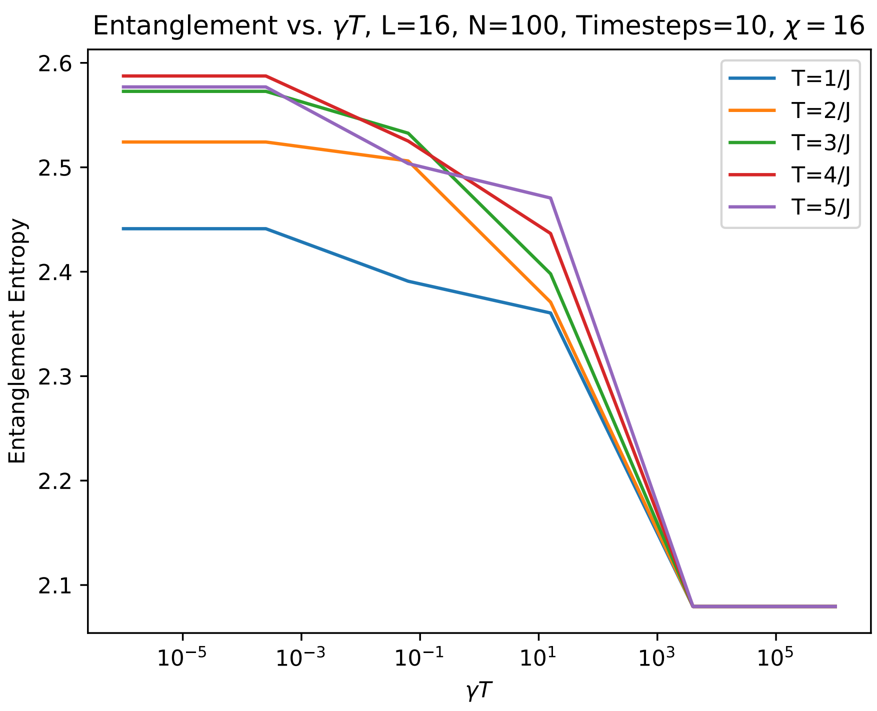
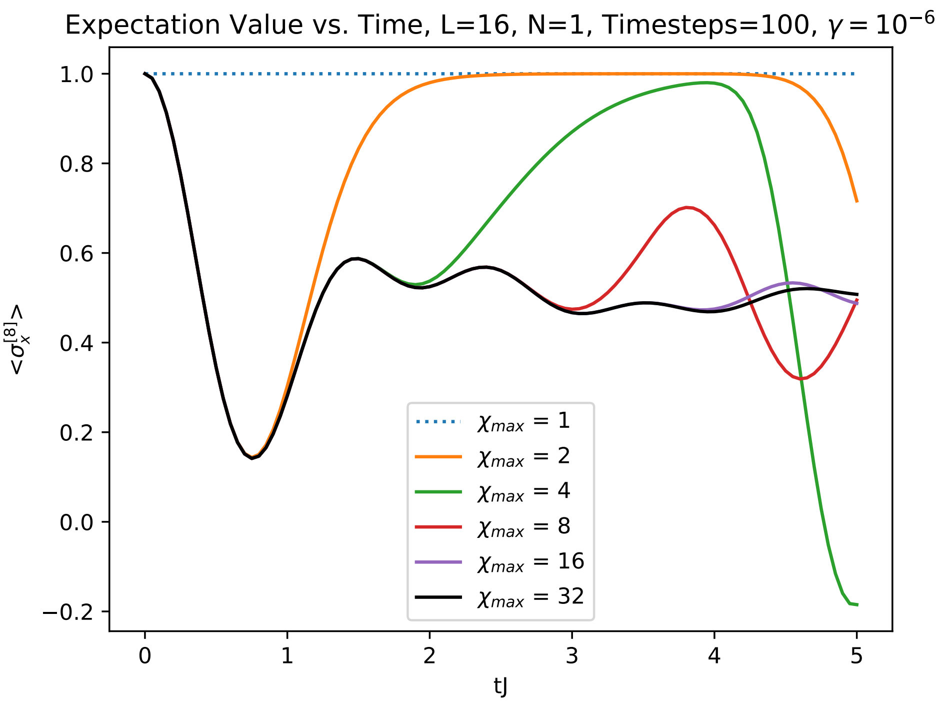


Figure 3.8: Entanglement as a function of noise strength γ and simulation time T for varying timescales

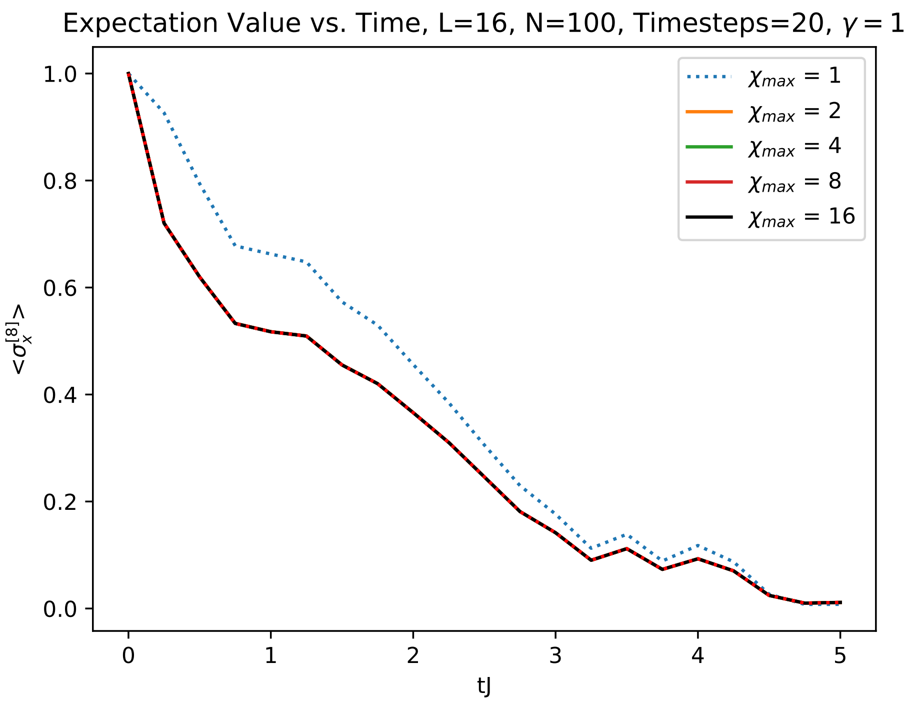
lation regimes based on the relation between noise strength and simulation time γT. Note that all tests are simulating up to $T = 1$ (or $T = \frac{1}{J}$), and to build intuition, it is important to recognize these graphs as a type of normalized noise process per unit time. For example, a noise process occurring every 1 microsecond, simulated over 1 second will be $\gamma T = 10^{-6}$. The dynamics of the noise process itself should be able to be seen around roughly $\gamma T \approx 1$. This is due to the noise strength being inversely proportional to its occurrence in time, that is, $\gamma \sim \frac{1}{T_\gamma}$ or $\gamma \sim \frac{1}{\sqrt{2T_\gamma}}$ in some cases.

Noise Regime	Entanglement Regime	γT	χ_{max} needed
Weak Noise	Strong Entanglement	$-\infty$ to 10^{-1}	2 to Full
Transition	Weak Entanglement	10^{-1} to 10^{3}	2
Strong Noise	Product State	10^{3} to ∞	1

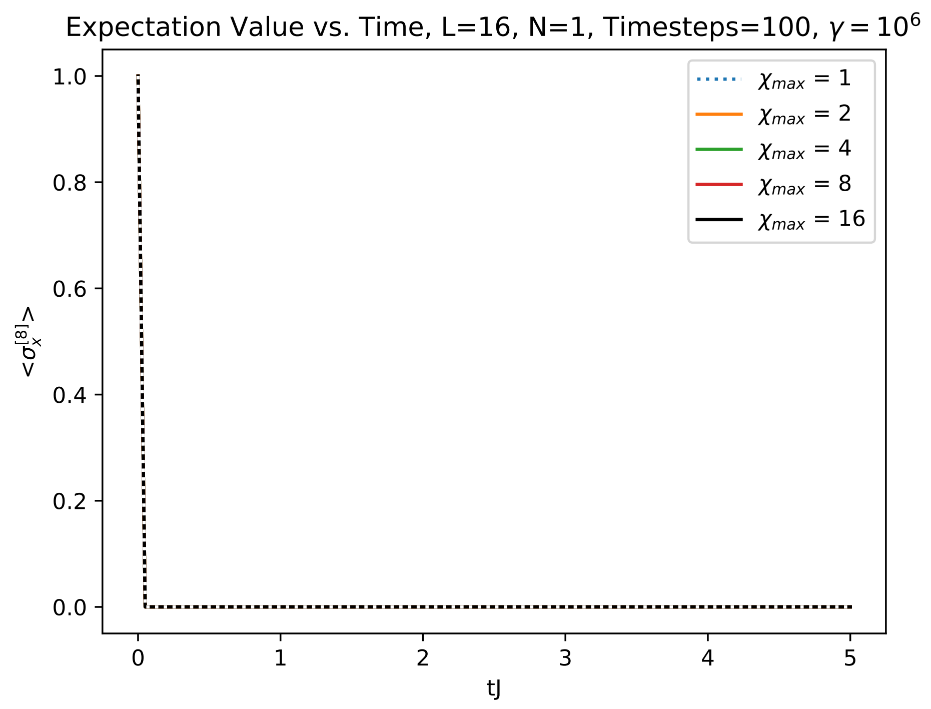
Table 3.1: Identifiable regimes corresponding noise strength, entanglement, and max bond dimension needed for the ratio between noise strength γ and simulation time T



(a) Weak noise $\gamma = 10^{-6} \ll T^{-1}$



(b) Medium Noise $\gamma = 1 \approx T^{-1}$



(c) Strong noise $\gamma = 10^6 \gg T^{-1}$

Figure 3.9: Expectation value calculated for various bond dimensions over a time period in different noise regimes

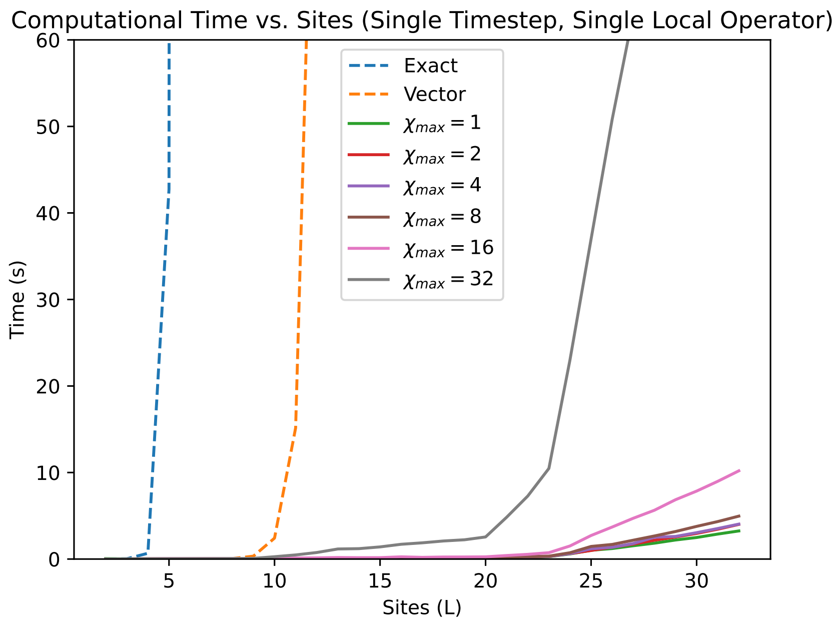
3.3.3 Computational Time

Finally, the computational time is compared between the superoperator method, the original vector jump method, and the TN jump method for a single timestep. This is plotted for several different bond dimensions so that it is possible to see how the time to simulate each of the previous regimes grows with system size. The following computational times are calculated with a cutoff at 60 seconds for a single timestep.

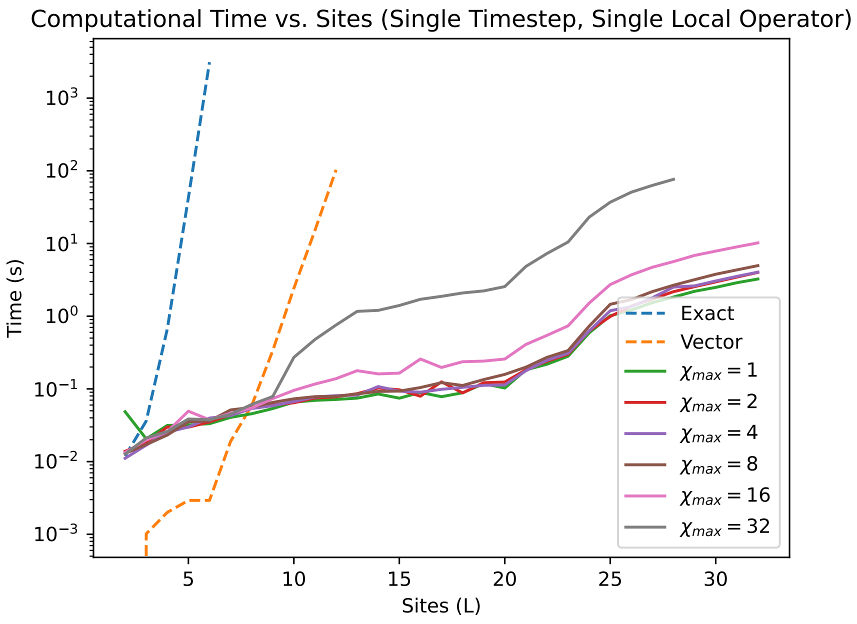
This is first shown as a comparison for a low number of sites such that the tensor network method can be compared to the exact calculation and vector method. This is plotted in a linear and log scale in Fig. 3.10. First, it is shown that the vector method is quicker for sites $L < 8$, but this graph does not capture the additional speedup from the TN method never needing to initialize more than a $d \times d$-level operator. The vector method sees a major slowdown in all cases due to having to initialize large $d^L \times d^L$ jump operators before starting the simulation which can be slow even when the operators are sparse. This took roughly an hour to generate on a personal computer for $L = 6$ (superoperator) and $L = 12$ (vector method) before even calculating anything. As expected, tensor networks grows linearly with "small" system sizes, with a higher slope for higher bond dimensions, eventually hitting a point of exponential growth. As shown before, these high bond dimensions can be avoided and thus also avoid unwanted growth in computational time

Now that this method has been compared with the exact calculation and the original vector method, it is possible to look at how far it could feasibly scale. For this, both a local operator as done previously as well as global operators have been considered. Calculating a global operator requires a full sweep calculating expectation values for each site rather than placing the MPS in site canonical form and calculating a single site. The results of the local operator calculation can be seen in Fig. 3.11 and the global operator in Fig. 3.12, both with linear and log scale.

The tensor network method is shown to scale significantly further than the original vector method. On a personal computer, it is possible to calculate a single timestep up to 3500 sites in less than a minute. There is also a different scaling factor with the local and global operator calculations due to the need to sweep for a global calculation. The global calculation is still well within the same limits, but will slowly scale worse as the site number is increased even further. In order to calculate a local operator, the site is shifted from left canonical into site canonical form before calculation, while for the global operator a sweep is performed. This explains some of the jaggedness or non-linearity in the local operator calculation, but could most likely also be optimized.

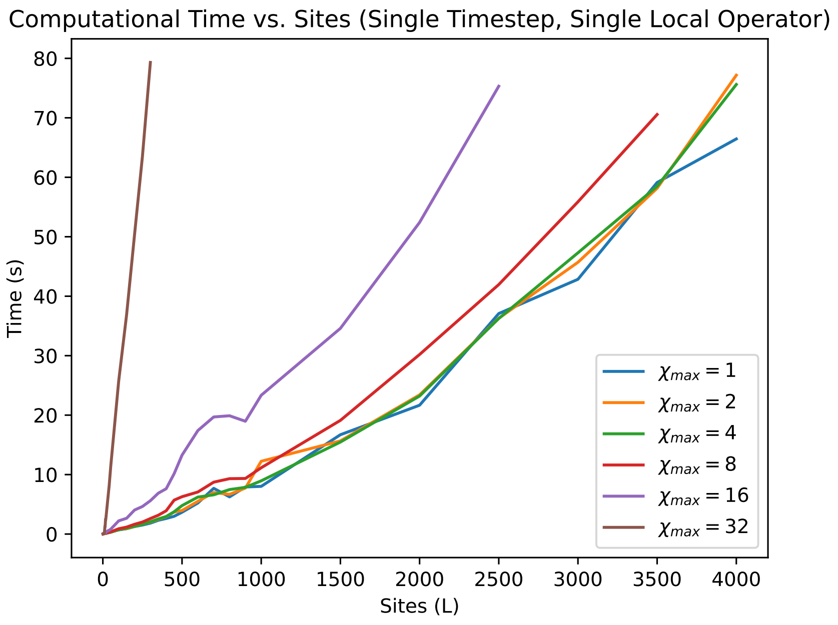


(a) Linear Computational Time

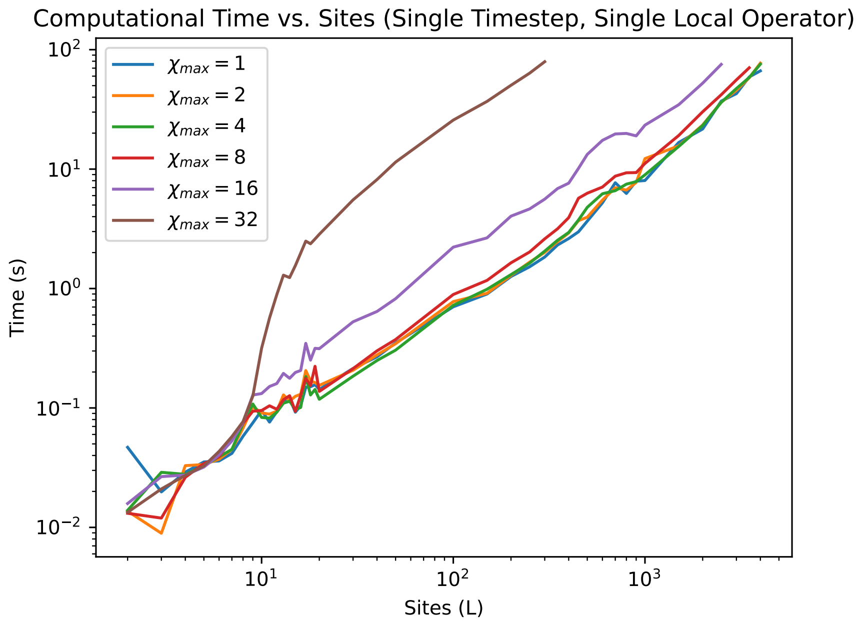


(b) Log Computational Time

Figure 3.10: Comparison between exact superoperator solution, the original vector method, and tensor networks for various bond dimensions for a single timestep

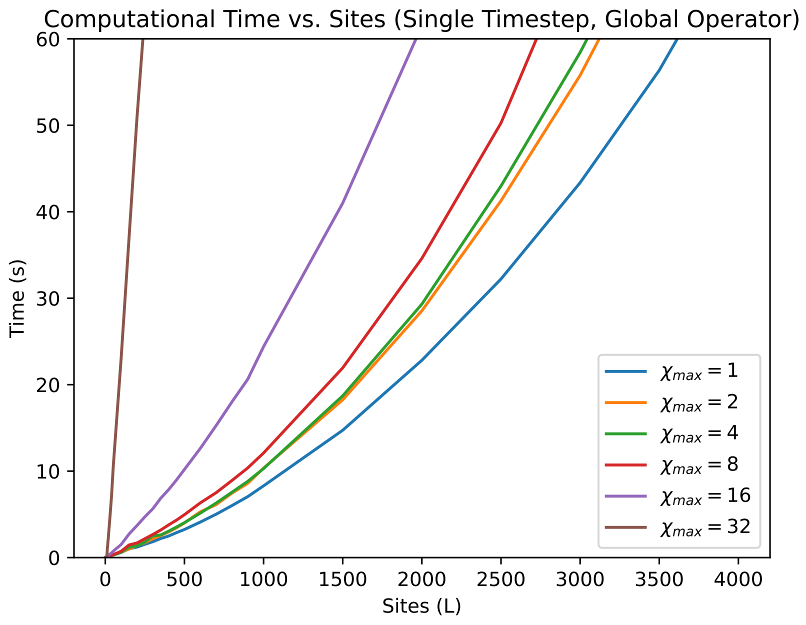


(a) Linear Computational Time

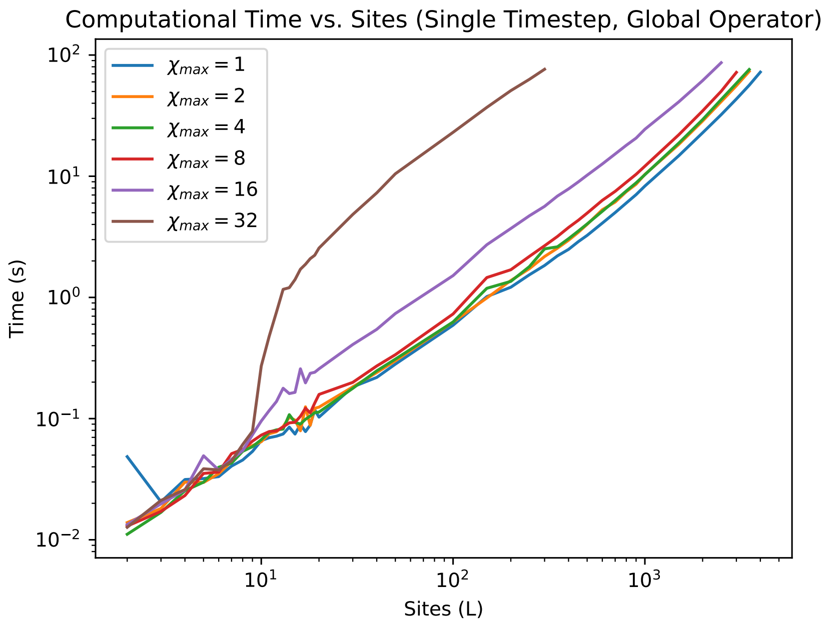


(b) Log Computational Time

Figure 3.11: Comparison between tensor networks of various bond dimensions for a single timestep calculating a local operator at the center site



(a) Linear Computational Time



(b) Log Computational Time

Figure 3.12: Comparison between tensor networks of various bond dimensions for a single timestep calculating a global operator

4 Applications

4.1 Coupled Transmons

As a first application, a small, realistic system is analyzed such that the capabilities of this method can be tested. Here a SWAP gate between two superconducting transmon qubits coupled through a resonator (effectively a fixed-frequency transmon) is simulated. For this, the following effective Hamiltonian found in [10] is used

$$\begin{aligned} H = {} & \omega_1(t) b_1^\dagger b_1 + \frac{\alpha_1}{2} b_1^\dagger b_1 (b_1^\dagger b_1 - \mathbb{1}) \\ & + \omega_2(t) b_2^\dagger b_2 + \frac{\alpha_2}{2} b_2^\dagger b_2 (b_2^\dagger b_2 - \mathbb{1}) \\ & + \omega_2^R a^\dagger a \\ & + g_1(t)(a + a^\dagger)(b_1^\dagger b_1) + g_2(t)(a + a^\dagger)(b_2^\dagger + b_2) \end{aligned} \tag{4.1}$$

The architecture that this represents is found in Fig. 4.1. The b_i operators correspond to the ladder operators on the i^{th} qubit and a operator corresponds to the resonator (and likewise for their conjugates). As superconducting qubits are bosonic systems, the number of levels in the qubits as well as the resonator can be controlled. This means the b_i operators are $d \times d$ matrices and the a operator is a $D \times D$ matrix where d is the number of qubit levels and D the levels of the resonator. In this use case, the levels are set the same such that $d = D$, however it is not necessary for the method to work. This Hamiltonian is essentially a sum over the individual Hamiltonians of the transmons, the resonator, as well as an interaction term.

The ω_i terms are the qubit frequencies, ω^R is the resonator frequency, α_i are the anharmonicities, and g_i are the coupling strengths between the qubits and the resonator. The values of the parameters used are found in Table 4.1.

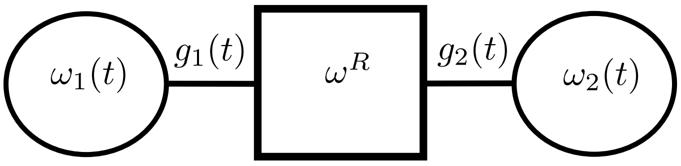


Figure 4.1: Architecture of two transmons coupled through a fixed-frequency resonator [10]

As this is a tensor network method, this Hamiltonian must be converted into a Matrix Product Operator (MPO). In order to do this, the system's layout needs to be defined.

$\omega_1/2\pi$	$\omega_2/2\pi$	$\omega^R/2\pi$	α_1	α_2	g_1	g_2
4.2	4.2	10	-0.32	-0.295	0.307	0.307

Table 4.1: Parameters used for simulating a SWAP gate on two transmons coupled through a resonator. All units are in GHz.

A 3-site MPS where the outer sites represent the qubits with physical dimensions d and the interior site is the resonator with physical dimension D is used for this. This creates a very intuitive representation of the system that mimics the architecture itself. Each site of the MPO then needs to contain the relevant terms for that site alone. Since the operators act on different sites, they can be viewed as local operators such that $b_1 = b \otimes \mathbb{1}_{D\times D} \otimes \mathbb{1}_{d\times d}$, $a = \mathbb{1}_{d\times d} \otimes a \otimes \mathbb{1}_{d\times d}$, and $b_2 = \mathbb{1}_{d\times d} \otimes \mathbb{1}_{D\times D} \otimes b$ (and likewise for the creation operators). Note that the identities as well as the zeros matrices in the MPO must maintain the correct dimension for its site. All together this results in the following MPO representation:

$$W_1 = \left(\omega_1(t)(b^\dagger b) + \tfrac{\alpha_1}{2}(b^\dagger b)(b^\dagger b + \mathbb{1}),\ \ \mathbb{1},\ \ g_0(b^\dagger + b),\ \ \mathbb{1}\right) \tag{4.2}$$

$$W_2 = \begin{pmatrix} \mathbb{1} & 0 & 0 & 0 \\ 0 & 0 & \omega^R a^\dagger a & 0 \\ a^\dagger + a & 0 & 0 & 0 \\ 0 & a^\dagger + a & 0 & \mathbb{1} \end{pmatrix} \tag{4.3}$$

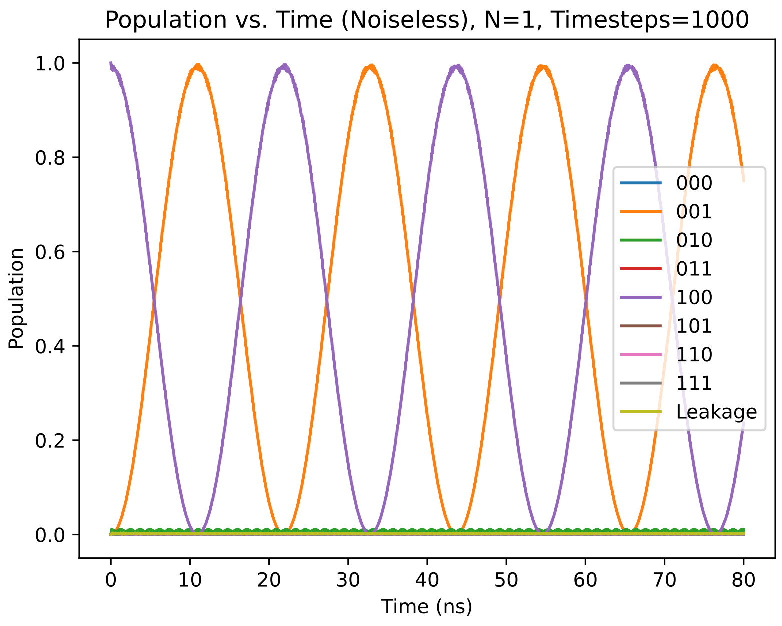
$$W_3 = \begin{pmatrix} \mathbb{1} \\ g_2(b^\dagger + b) \\ \mathbb{1} \\ \omega_2(t)(b^\dagger b) + \tfrac{\alpha_2}{2}(b^\dagger b)(b^\dagger b + \mathbb{1}) \end{pmatrix} \tag{4.4}$$

The MPS is then initialized in $|100\rangle$ such that one qubit is excited, the resonator contains 0 excitations, and the other qubit is in the ground state. Tensor networks has a nice advantage in that it is possible to significantly extend the number of levels very easily and represent a more realistic bosonic system with many levels. Since no analysis was done on the entanglement or truncating the bond dimension of d-level systems, the levels were set to a basic 3-level system for both qubits and the resonator such that leakage outside of the computational space i.e. to the $|2\rangle$ state can be calculated. It would be advantageous to raise this as much as possible to get the most realistic simulation then truncate the bond dimensions.

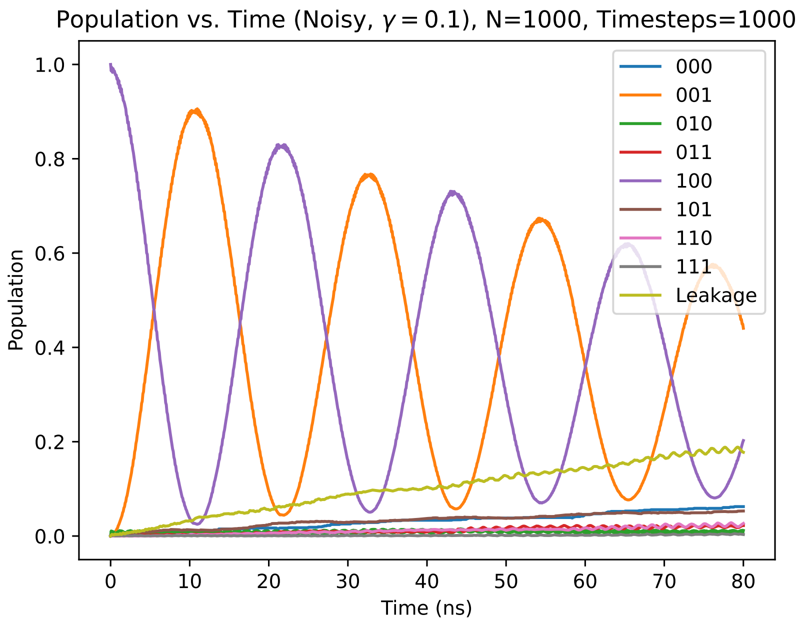
For this simulation, 1000 timesteps are used as it was seen that the dynamics are very timestep dependent relative to the previous simulations. At each timestep, the population of all the possible states is calculated. This is done by taking the expectation value of operators of the form $P_{ijk} = |i\rangle\langle i| \otimes |j\rangle\langle j| \otimes |k\rangle\langle k|$ for all bitstrings ijk. The leakage is then calculated by $L = 1 - \sum_{[ijk]} P_{ijk} - P_{i2k}$. This last term is due to the fact that the resonator is not part of the computational space and it does not matter if it fills up with excitations.

This has been simulated for both the noiseless, perfect simulation as well as a noisy with relaxation, dephasing, and excitations for $N = 1000$ trajectories over 80 ns. The noise strengths are all set equal to occur at a strength $\gamma = 0.00125$ ns^{-1} $= 1.25$ MHz ($\gamma T = 0.1$). This was set arbitrarily as a proof of concept based on the previous results such that noise disrupts the system but does not completely overwhelm it. Realistic γ's would need to be investigated for actual systems.

The results of this simulation are seen in Fig. 4.2. The bitstrings correspond to the system layout such that the first bit is the state of the first qubit, second is the resonator, and third is the second qubit. In the noiseless simulation, the expected behavior of a SWAP between the qubits can be seen. A slight hybridization with the resonator state 010 is visible as the system is not fully in the dispersive regime where the detuning with the resonator is much greater than the coupling strengths [1]. In the noisy simulation, a decrease in the amplitudes of the expected SWAP states is visible and an increase in leakage. There is also a slight increase in the population of 000 and 101 as well as some hybridization when an excitation is in the resonator.



(a) Noiseless Transmon



(b) Transmon with random relaxation, dephasing, and excitations with noise strength strength $\gamma = 0.00125\ \text{ns}^{-1} = 1.25$ MHz ($\gamma T = 0.1$)

Figure 4.2: Comparison between noiseless simulation of coupled transmons and a noisy simulation with $d = 3$ levels, relaxation, dephasing, and excitations

4.2 Scaling

In this section, the knowledge of the previous benchmarks is used to see how far the method can be pushed. The goal here is to capture the dynamics of a noise process itself evolving under the same Ising model presented previously with $J = 1$ and $g = 0.5$. Both relaxation and dephasing are still considered with $\gamma = 1$ and simulated up to a time where it is possible to see the dynamics of the relaxation and dephasing themselves. As seen previously for smaller systems, this is expected to be on a timescale with the same magnitude as the noise i.e. $T \approx T_\gamma = \frac{1}{\gamma}$. Since this is in the transition regime and weak entanglement regime, it is simulated with a bond dimension $\chi_{\max} = 2$.

Seen in Fig. 4.3, the expectation value of an operator at the center of a chain of 5, 16, 50, 100, and 1000 sites is simulated using 100 timesteps and 100 trajectories. More timesteps and trajectories would cause a smoother curve similar to the coupled transmon simulation in Fig. 4.1. There is a similar pattern for all system sizes with a decay into a minimum followed by a slight increase before remaining constant. More sites leads to this decay occurring in a longer timespan as well as finally ending up in a higher constant value. This could either be some form of stabilization from the increased number of interactions or the dynamics are simply not being captured properly. It is noted that for 100 and 1000 sites, there is pure decay and the dip is not visible. It is currently not possible to check if the dynamics shown are correct as it is not possible to calculate it exactly and the hardware to compare experimentally does not exist. As the original method was created with far less sites in mind, further modifications may need to be included for these larger system sizes.

This method was originally considered for small system sizes in which a jump will affect the entire system greatly and any single site is also highly likely to be jumped compared to larger systems. It could be possible that there are simply not enough jumps occurring for such large systems as the ones considered in this section. If there are 1000 sites, the number of possible trajectories the system could take is so vast that it is not possible to capture the true dynamics without significantly more trajectories. However, it is necessary to avoid this or it will lead to much longer computational time.

If the dynamics in Fig. 4.3 are correct, then nothing needs to be modified from the original method. If the dynamics are wrong, there is a clear improvement that could be made to introduce more variance into the trajectories.

Currently, maximum one jump per timestep is allowed. For say $L = 5$, that means 1 out of 5 i.e. 20 % of the sites experience noise for a single jump. For $L = 1000$, that is 0.1 %. It could make sense to select more than one jump at a time to introduce more variation in the trajectories.

With this in mind, a modified method was considered for 100 sites where if a jump is to occur i.e. $\delta p < \epsilon$, then 20 % of the sites will experience a jump. No site is able to experience more than one jump in a timestep meaning for example a site cannot experience relaxation and dephasing simultaneously. This means after applying the jump, all possible jump operators on that site are removed by setting them to 0 in the probability distribution followed by renormalizing it. The results of this can be seen in Fig. 4.4. This modification causes the dip to appear as seen in the smaller system sizes. Further analysis on whether this is the correct dynamics or not is necessary, but this result looks positive.

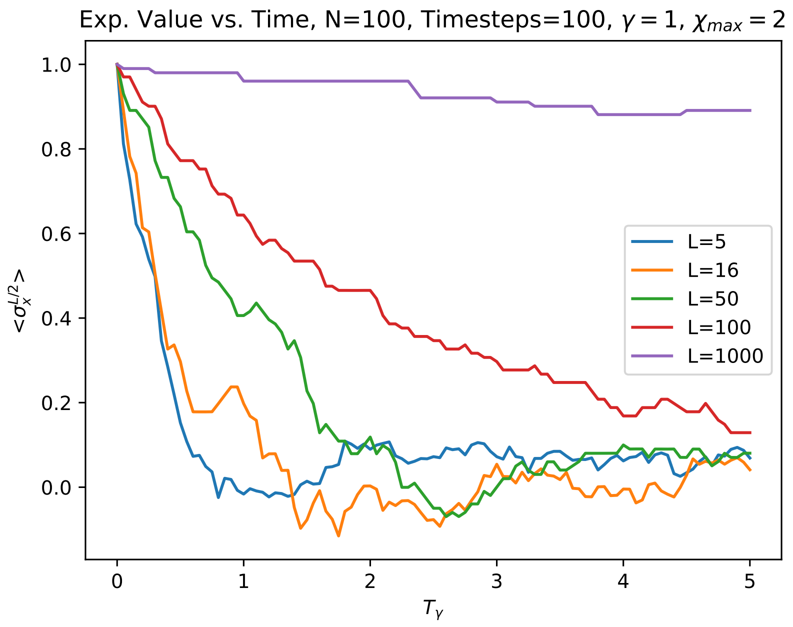


Figure 4.3: Calculation of the expectation value of a local operator at the center site to attempt to capture the dynamics of a noise process

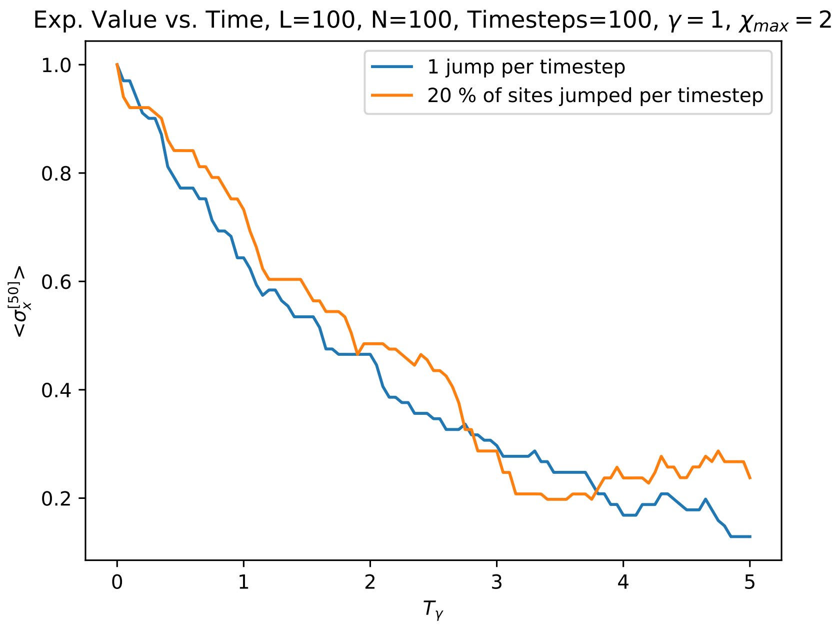


Figure 4.4: Modification to allow 20 % of sites to be jumped compared to single jump

5 Conclusion and Outlook

This work has introduced the tensor network quantum jump method and demonstrated its capabilities as a noise simulation method. As with any simulation method, reducing error and computational time is always desired. There are several possible improvements that can explored and potentially easily implemented in the tensor network quantum jump method. This includes implementing the two-site TDVP to reduce projection error as well as using a matrix-free exponential such as the Arnoldi method rather than the Lanczos method. The Arnoldi method could also potentially allow the dissipative sweep to be included in the TDVP sweep itself.

A clear next step would be to implement a time-dependent Hamiltonian using this method to analyze noise processes during gate applications to larger systems than currently possible. The use of TDVP in this method also opens the door for simulating 2D systems by allowing long-range interactions. This could be facilitated through using a snaking MPS model in which more than nearest neighbor interactions are included in the Hamiltonian.

Overall, the creation of this tensor network quantum jump method allows us to scale noise simulations further than previously possible. This method does not encounter the exponential growth for solving a Lindblad master equation that is encountered in an exact calculation or the original quantum jump method. The results of this work have shown that the tensor network quantum jump method can easily calculate the effect of noise processes on systems over 100 qubits on a personal computer as well as excellent scaling to even larger system sizes. This allows us to extend our simulation capabilities with the intent to be used as a state-of-the-art benchmarking and error mitigation tool as quantum computing platforms scale to more and more qubits.

Bibliography

[1] M. Boissonneault, J. M. Gambetta, and A. Blais. Dispersive regime of circuit qed: photon-dependent qubit dephasing and relaxation rates. 10 2008. doi: 10.1103/PhysRevA.79.013819. URL `http://dx.doi.org/10.1103/PhysRevA.79.013819`.

[2] C. A. Brasil, F. F. Fanchini, and R. de Jesus Napolitano. A simple derivation of the lindblad equation. 10 2011. doi: 10.1590/S1806-11172013000100003. URL `http://dx.doi.org/10.1590/S1806-11172013000100003`.

[3] J. C. Bridgeman and C. T. Chubb. Hand-waving and interpretive dance: An introductory course on tensor networks. 3 2016. doi: 10.1088/1751-8121/aa6dc3. URL `http://dx.doi.org/10.1088/1751-8121/aa6dc3`.

[4] J. Haegeman, J. I. Cirac, T. J. Osborne, I. Pizorn, H. Verschelde, and F. Verstraete. Time-dependent variational principle for quantum lattices. 3 2011. doi: 10.1103/PhysRevLett.107.070601. URL `http://dx.doi.org/10.1103/PhysRevLett.107.070601`.

[5] J. Haegeman, T. J. Osborne, and F. Verstraete. Post-matrix product state methods: To tangent space and beyond. 5 2013. doi: 10.1103/PhysRevB.88.075133. URL `http://dx.doi.org/10.1103/PhysRevB.88.075133`.

[6] J. Haegeman, M. Marien, T. J. Osborne, and F. Verstraete. Geometry of matrix product states: Metric, parallel transport, and curvature. *Journal of Mathematical Physics*, 55, 2 2014. ISSN 00222488. doi: 10.1063/1.4862851.

[7] E. Hairer, M. Hochbruck, A. Iserles, and C. Lubich. Geometric numerical integration. *Oberwolfach Reports*, 2009. doi: 10.4171/owr/2006/14.

[8] P. Kramer and M. Saraceno. *Geometry of the Time-Dependent Variational Principle in Quantum Mechanics*, volume 140, pages 112–121. 01 2007. ISBN 978-3-540-10271-7. doi: 10.1007/3-540-10271-X_317.

[9] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits. 4 2019. doi: 10.1063/1.5089550. URL `http://dx.doi.org/10.1063/1.5089550`.

[10] H. Lagemann, D. Willsch, M. Willsch, F. Jin, H. D. Raedt, and K. Michielsen. Numerical analysis of effective models for flux-tunable transmon systems. 1 2022. doi: 10.1103/PhysRevA.106.022615. URL `http://dx.doi.org/10.1103/PhysRevA.106.022615`.

[11] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45, 1950. ISSN 0091-0635. doi: 10.6028/jres.045.026.

[12] C. Lubich. *From quantum to classical molecular dynamics : reduced models and numerical analysis*. European Mathematical Society, 2008. ISBN 9783037190678.

[13] D. Manzano. A short introduction to the lindblad master equation. 6 2019. doi: 10.1063/1.5115323. URL `http://dx.doi.org/10.1063/1.5115323`.

[14] K. Mølmer, Y. Castin, and J. Dalibard. Monte carlo wave-function method in quantum optics. *J. Opt. Soc. Am. B*, 10:524–538, 3 1993. doi: 10.1364/JOSAB.10.000524. URL `http://opg.optica.org/josab/abstract.cfm?URI=josab-10-3-524`.

[15] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig. Time-evolution methods for matrix-product states. *Annals of Physics*, 411, 12 2019. ISSN 1096035X. doi: 10.1016/j.aop.2019.167998.

[16] F. Schindler and A. S. Jermyn. Algorithms for tensor network contraction ordering. 1 2020. doi: 10.1088/2632-2153/ab94c5. URL `http://dx.doi.org/10.1088/2632-2153/ab94c5`.

[17] U. Schollwoeck. The density-matrix renormalization group in the age of matrix product states. 8 2010. doi: 10.1016/j.aop.2010.09.012. URL `http://dx.doi.org/10.1016/j.aop.2010.09.012`.

[18] L. Vanderstraeten, J. Haegeman, and F. Verstraete. Tangent-space methods for uniform matrix product states. 10 2018. doi: 10.21468/SciPostPhysLectNotes.7. URL `http://dx.doi.org/10.21468/SciPostPhysLectNotes.7`.