Alma Mater Studiorum · University of Bologna

Department of Physics and Astronomy
Master Degree in Physics

Utilizing Global Subspace Expansion for Time Dependent Variational Principle on Spanning Tree Tensor Networks

Supervisor:

Richard Milbradt, M.Sc.

Co-supervisors:

Prof. Christian B. Mendl
Prof. Elisa Ercolessi

Presented by:

Pouriya Haji Ghadimi

Academic Year 2023/2024

"Programmed by quanta, physics gave rise first to chemistry and then to life; programmed by mutations and recombination, life gave rise to Shakespeare; programmed by experience and imagination, Shakespeare gave rise to Hamlet."

— Seth Lloyd

Abstract

We adapted the *Global Subspace Expansion* (GSE) algorithm on tree tensor networks (TTN), integrated it into the *Time Dependent Variational Principle algorithm* (GSE-TDVP), to mitigate the 1-site TDVP projection error, while achieving the accuracy of 2-site TDVP. We utilized an adaptive adjustment of a truncation parameter within the GSE algorithm, to achieve stable bond growth under controlled conditions during the evolution. Furthermore, we investigated the representation of wave-function of 2D systems with spanning tree as an intermediate approach between *Matrix Product States* (MPS) and *Projected Entangled-Pair Sates* (PEPS). We conducted a methodological error analysis of the GSE-TDVP algorithm across various spanning tree structures, and proposed a scoring metric based on TDVP performance and the network connectivity. A significant bottleneck in the algorithm—matrix exponential approximation—was addressed by incorporating an enhanced *Krylov subspace projection* method with adaptive time-stepping. Finally, we enhanced the *Tensor Jump Method* (TJM) by utilizing GSE-TDVP on spanning tree and benchmarked a simple dissipative dynamics on a 2D lattice.

Contents

Introduction

The study of quantum many-body systems lies at the heart of modern theoretical physics, offering insights into fundamental questions about the nature of quantum entanglement with practical challenges in simulating complex quantum phenomena ranging from superconductivity to quantum phase transitions. These systems, characterized by a vast number of interacting particles, present a formidable challenge: their complexity grows exponentially with size, making exact solutions computationally infeasible beyond small scales. Over the past few decades, researchers have turned to innovative numerical tools to represent quantum states efficiently by breaking them into manageable pieces—tensors—connected in a network, allowing scientists to simulate systems that were once out of reach. Researchers have developed various network shapes—some like chains, others like grids or trees—to simulate quantum system dynamics over time or understand the state of system at the lowest possible energy. This thesis enters this landscape with a focus on loop-free tree networks, by addressing their shortcomings and tailoring them to practical quantum simulation.

Tensor Network (TN) Time-evolution algorithms face several difficulties. Traditional approaches often suffer from large truncation errors, poor scalability with long-range interactions, or rapid entanglement growth. A promising alternative is the *Time-Dependent Variational Principle* (TDVP) [1], which projects the *Schrödinger* equation onto the tangent space of a fixed-rank tensor network. TDVP not only preserves unitarity more reliably over long times but can also incorporate adaptive bond-dimension growth to manage rising entanglement. However, a fundamental trade-off emerges: 1-site TDVP is computationally cheaper but may undershoot the true entanglement growth, whereas 2-site TDVP explores a larger variational space by increasing bond dimensions—albeit at a substantially greater computational cost. The main focus of this thesis is to address this challenge by constructing a global correction space, informed by the entire network, and incorporating it into the TDVP algorithm within .

In chapter 1, we introduce the fundamental concepts of tensor networks along with the essential tools needed for implementing *Tree Tensor Networks* (TTN). Chapter 2 presents TDVP algorithm and details the practical steps for its adaptation to the TTN framework. At the end of this chapter, we investigate the representation of 2D Many-Body Wave-functions on spanning tree structures. Building on this, in chapter 3, we introduce *Global Subspace Expansion* (GSE) [2] algorithm, providing practical implementation guidelines within the TTN framework. Next, by utilizing GSE, we propose a stable strategy for enlarging the 1-site tangent space during TDVP evolution (GSE-TDVP) and address computational bottlenecks in matrix exponentiation through enhanced *Krylov* strategy, thereby improving the overall scalability of the method. Finally, Chapter 4 integrates the developed GSE-TDVP with the *Tensor Jump Method* (TJM) [3] and extends the simulation to two-dimensional systems via a spanning tree structure. This integration enhances TJM's robustness for simulating dissipative dynamics in larger lattices, as demonstrated by benchmarks on a *2D Transverse-field Ising model.*

By focusing on three key areas—enhancing the TDVP with *Global Subspace Expansion* (GSE), exploring spanning tree structures, and improving the *Tensor Jump Method* (TJM)—this work contributes to the ongoing effort to make TN simulations more accurate and practical.

1. Tree Tensor Networks (TTN)

Tensor network (TN) methods are taking a central role in modern computational quantum physics [4–6]. They can provide an efficient approximation to certain classes of quantum states, with graphical representation that can simplifies complex calculations by visually encoding tensor contractions, making it easier to understand and manipulate quantum circuits, channels, protocols, open systems, and various transformations, evolutions, and interactions that quantum states undergo.

1.1 Tensor Network (TN) Methods

Solving the *Schrödinger* equation for many-body Hamiltonians in the presence of interactions poses a fundamental and complex challenge due to the presence of an exponentially large number of degrees of freedom. This growth leads to both increasing memory requirements to store exponentially many complex values and computational times to perform the operations needed to simulate their dynamics. To get some perspective about the degree of complexity, we can recall that the exact numerical solution for a tiny nanoscale piece of material would already require a computer with a hard disk containing more bits than atoms present in the entire universe.

Early solutions mostly relied on approximative and effective theories where the properties of the interacting system can be adequately described in a single-particle picture with interactions only entering at a perturbative level such as the *Hartree-Fock* approximation (which employed a mean-field approach) [7], many-body perturbation theory (which treated interactions as small deviations from a solvable reference system) [8], the *Configuration Interaction* (CI) method (which expanded the wavefunction in a basis of many-particle configurations) [9], and *Kohn-Sham Density Functional Theory* (DFT) (which recast the problem into an effective single-particle framework) [10]. Although their approximative nature only accounts for weak quantum correlations, strong interactions that give rise to these intriguing quantum effects and phases of matter including Mott insulators [11], high-Tc superconductivity [12], frustrated quantum magnetism [13], the fractional quantum Hall effect [14], or quantum critical phases of matter [15] and so on, lead to a breakdown of any perturbative ansatz and as a result, the full many-body wave-function has to be taken into account.

The most straightforward way of tackling the many-body wave-function numerically is to generate the full many-body Hamiltonian of the system and use *Exact Diagonalization* (ED) to obtain all or a subset of its eigenvectors [16]. The complexity to perform the full diagonalization scales as $O(d^{3N})$, where N represents the number of sites in the lattice system and d is the local dimension, limiting its applicability especially in the context of many-body systems with long correlation lengths [16]. Another set of approaches are series expansion techniques, which expand a specific quantity in terms of a power series of one or more parameters [17]. The main limitation of these techniques is achieving high enough expansion orders to get the convergence of quantity of interest. This is particularly difficult at or below a thermal phase transition and for strongly correlated ground-state phases, where long-ranged correlations dominantly determine the physical properties of a system [18]. *Quantum Monte-Carlo* (QMC) techniques [19] are widely applied to study strongly correlated systems stochastically, by sampling the partition function of the full interacting system. For a vast number of systems, QMC can be implemented to treat very large system sizes of hundreds of sites or even directly working in the thermodynamic limit with only polynomial cost scaling. However, for most frustrated magnets and models of itinerant fermions, the computation time of the algorithms grows exponentially with the system size due to the so-called sign problem [20].

TN approaches, on the other hand, where the wave-function of a quantum many-body system is parametrized in terms of a set of interconnected tensors, are at the forefront of classical tools for scalable quantum simulation. Although these ansatz by construction only work efficiently (i.e., with polynomial cost scaling) for a specific set of lowly-entangled states in the *Hilbert* space, they turn out to be excellent representations of many low-energy wave-functions of strongly correlated systems [21]. Therefore, the main motivation of using TN include the absence of a sign problem, their non-perturbative character, and their ability to treat large system sizes. TN methods have been extensively explored and opened up various new research directions towards understanding non-equilibrium [22], finite temperature [23] and topological properties [24] of low-dimensional quantum systems and offer promising approaches in the context of two-dimensional correlated systems such as single- and multi-band Hubbard and t-J models [25].

Beyond condensed matter physics, TN methods keep generating impact, particularly in high-energy physics and quantum chemistry. In high-energy physics, TN have been instrumental in studying lattice gauge theories, providing an alternative to traditional *Monte Carlo* methods [26]. Additionally, TN have been applied in holography and quantum gravity, offering insights into the *AdS/CFT* correspondence through the study of entanglement structures in many-body systems [27]. In quantum chemistry, TN techniques have been leveraged to efficiently approximate wave-functions of molecular electronic structures, enabling high-precision computations of strongly correlated electrons in complex molecular systems [28].

Beyond physics, TN have emerged as powerful tools in machine learning, where they enable efficient parameterization of deep learning models and reduce computational complexity. In supervised learning, tensor-based models have achieved high classification accuracy with reduced computational cost [29]. In natural language processing, TN have been employed in probabilistic sequence modeling, providing innovative approaches for structured text generation and inference [30]. In medical imaging, TN have demonstrated competitive performance in image classification tasks while requiring fewer computational resources than deep learning models [31]. Furthermore, TN have found applications in cognitive science and decision-making, where they have been used in planning and active inference to model probabilistic decision-making [32].

1.2 Tree Tensor Networks (TTN)

In the simulation of quantum many-body systems, an effective variational ansatz must satisfy two primary conditions: it should yield a highly accurate representation of the imaginary and real time-evolved target state, and it must permit the efficient computation of key physical quantities, including local observables and their corresponding correlation functions.

At the foundation of TN methods, *Matrix Product States* (MPS) were first successfully applied in the Density Matrix Renormalization Group (DMRG) to efficiently capture the ground states of one-dimensional quantum systems with high accuracy [33]. Over time, MPS have been extended beyond static ground-state properties to finite-temperature physics [34], quantum dynamics [35], and open quantum systems [36, 37]. And recent advancements leverage MPS in hybrid quantum-classical simulations [38] and adaptive quantum circuits [39].

While MPS is suited to deal with non-critical, short-range, one-dimensional Hamiltonians, their applicability is limited in higher-dimensional systems and critical states, where entanglement grows beyond what a low-bond-dimension MPS can efficiently capture [40, 41].

The *Projected Entangled Pair States* (PEPS) was later introduced as a natural generalization of MPS to two- and higher-dimensional lattice systems [41]. PEPS represents the wave-function as a network of tensors arranged on a lattice, where each site is connected to its neighbors via virtual bonds, allowing for an efficient encoding of quantum correlations and entanglement structure of higher-dimensional ground states, making them a powerful tool for studying two-dimensional quantum spin systems, the Hubbard model, and frustrated magnets [42]. However, the computational cost of contracting PEPS grows exponentially with system size, typically scaling with bond dimension as $O(\chi^{10})$ or higher, making their numerical implementation significantly more demanding than MPS [43]. Recent advances in tensor contraction techniques and variational optimization have improved the efficiency of PEPS methods, enabling studies of larger systems and more complex quantum phases [44, 45].

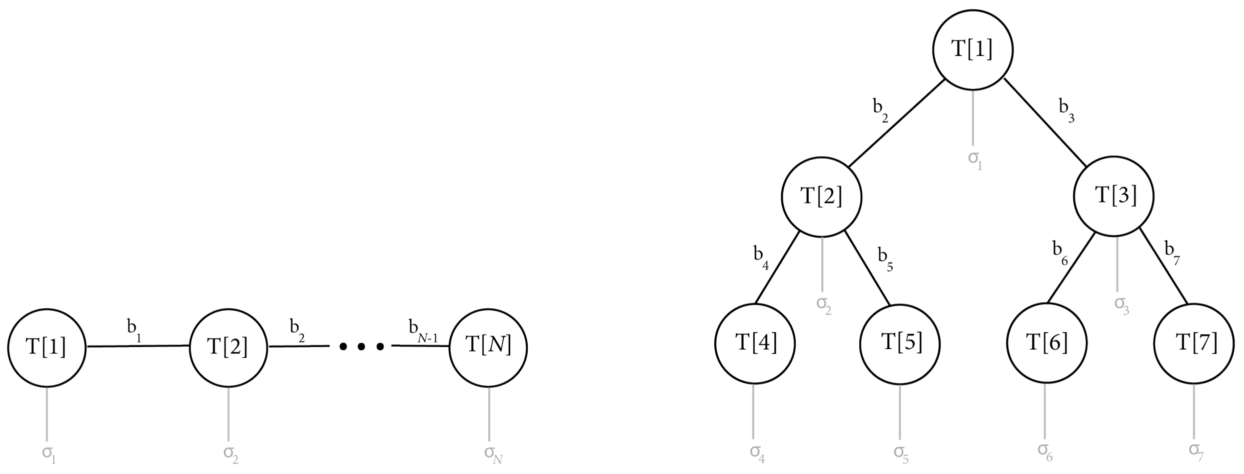
The *Multi-scale Entanglement Renormalization Ansatz* (MERA) was introduced as an alternative TN structure specifically designed to efficiently describe critical systems and scale-invariant quantum states [46]. Additionally, MERA has been instrumental in studying conformal field theories and holographic dualities by providing a TN realization of the *AdS/CFT* correspondence [47].

1.2.1 Tree Tensor Network State (TTNS) Ansatz

Tree Tensor Networks (TTN) provide an alternative TN architecture designed to efficiently represent quantum many-body states by arranging tensors in a hierarchical tree-like structure [48]. Unlike MPS, which follows a linear connectivity pattern, TTN employs a branching structure that enables a more flexible representation of quantum correlations, making it well-suited for capturing long-range entanglement in higher-dimensional systems [49]. In quantum chemistry, TTN provides an efficient description of molecular systems with complex entanglement structures [50]. TTN has been shown to outperform MPS in DMRG calculations for complex molecular systems in [51]. A substantial advantage of TTNS is that at a finite bond dimension, it is able to capture algebraically decaying correlation functions. This in contrast to MPS which is only able to represent exponentially decaying correlations [52]. Tree-like TNS with uniform binary-tree structures naturally exhibit power-law two-body correlations [53], making TTN effective in addressing challenges posed by critical long-range correlations and interactions [54].

Consider a quantum system with N degrees of freedom, each associated with a physical basis $|\sigma_i\rangle$ with local physical dimension d. The wave-function can be expressed as a high-order tensor $\Psi^{\sigma_1\sigma_2\cdots\sigma_N}$, belonging to the exponentially large *Hilbert* space $\mathcal{H} = (\mathbb{C}^d)^{\otimes N}$, which makes direct computations with Ψ impractical. The *Tree Tensor Network State* (TTNS) ansatz addresses this challenge by approximating Ψ, i.e., the full wave-function is factorized into N multi-dimensional arrays (tensors). Therefore, the wave-function is expressed as a contraction of N low-rank tensors connected with a set of virtual bonds B_i for each local tensor $T[i]$ at site i, where $B_i = \{b_j : b_j \text{ is connected to node } i\}$ represents all virtual bonds connecting tensor $T[i]$ to its neighbors in the tree structure. This is given by

$$|\Psi\rangle = \sum_{\{b\},\{\sigma\}} T[1]^{\sigma_1}_{B_1} T[2]^{\sigma_2}_{B_2} \cdots T[N]^{\sigma_N}_{B_N} |\sigma_1\sigma_2\cdots\sigma_N\rangle\,, \tag{1.1}$$



(a) Matrix Product State (MPS). (b) Perfect Binary Tree.

Figure 1.1: Examples of Tensor Network Structures.

where $\{b\} = \{b_1 \cdots b_L\}$ denotes the set of all virtual indices and $\{\sigma\}$ represents the set of physical indices. The contraction between local tensors $T[i]$ is organized according to a tree structure through the B_i collective indices. Denoting the virtual bond dimension of b_i by χ^i and the physical dimension of σ_i by d^i, each local tensor $T[i]$ has a shape determined by its set of virtual indices B_i and its physical index. In general, if $B_i = \{b_{j_1}, b_{j_2}, \dots, b_{j_k}\}$, then the tensor $T[i]$ has the shape $\left(\chi^{j_1},\, \chi^{j_2},\, \dots,\, \chi^{j_k},\, d^i\right)$, where k is the number of its neighbors (parent and children).

The TTN structure can be naturally encoded with a graph $G = (V, E)$, where each tensor corresponds to a vertex $i \in V$, and each virtual bond between tensors corresponds to an edge $(i, j) \in E$.

Example 1: MPS. MPS could be considered as a maximal unbalanced binary tree where the structure is a simple chain of N tensors, as illustrated in Figure 1.1(a), consisting of :

- Root node $T[1]$: With no parent index, a single child index b_1 (connecting to $T[2]$), and one physical index σ_1. One could also choose a tensor in the middle as the root. This does however, make implementation unnecessarily complicated.
- Intermediate nodes $T[2], \dots, T[N-1]$: Each has two virtual (bond) indices, the parent index b_{i-1} (shared with the preceding tensor $T[i-1]$) and the child index b_i (connecting to $T[i+1]$). Each tensor also has one physical index σ_i.
- Leaf nodes $T[N]$: Has only a parent index b_{N-1}, carries the physical index σ_N, and has no child index, as it is the end of the chain.

The full MPS wave-function reads:

$$|\Psi\rangle \;=\; \sum_{b_1,\dots,b_{N-1}} T[1]^{\,\sigma_1}_{\,b_1}\, T[2]^{\,\sigma_2}_{\,b_1,b_2}\, \dots\, T[N]^{\,\sigma_N}_{\,b_{N-1}}\, |\sigma_1\, \sigma_2\, \cdots\, \sigma_N\rangle. \tag{1.2}$$

Example 2: Perfect Binary Tree. A perfect binary tree of depth M is labeled by $i = 1, \dots, 2^M - 1$ tensors consisting of:

- Root node $T[1]$: Has no parent index and exactly two child indices b_2 and b_3, connecting it to $T[2]$ and $T[3]$.

- Internal nodes $2 \leq i \leq 2^{M-1} - 1$: Each has one parent index (shared with node $\lfloor i/2 \rfloor$) and two child indices, namely b_{2i} and b_{2i+1}, connecting it to $T[2i]$ and $T[2i+1]$.

- Leaf nodes $i > 2^{M-1} - 1$: These have no children, so each leaf node $T[i]$ has just one parent index $b_{\lfloor i/2 \rfloor}$.

When all the virtual (bond) indices b_i are summed (contracted) over, the full TTNS wavefunction is:

$$\begin{aligned} |\Psi\rangle = &\sum_{\{\sigma\}} \sum_{b_2, b_3, \ldots, b_{2^M-1}} T[1]^{\sigma_1}_{b_2, b_3} \\ &\times \prod_{i=2}^{2^{M-1}-1} T[i]^{\sigma_i}_{b_{\lfloor i/2 \rfloor}, b_{2i}, b_{2i+1}} \prod_{j=2^{M-1}}^{2^M-1} T[j]^{\sigma_j}_{b_{\lfloor j/2 \rfloor}} |\sigma_1 \, \sigma_2 \, \ldots \, \sigma_{2^M-1}\rangle. \end{aligned} \tag{1.3}$$

A perfect binary tree with depth $M = 3$, as illustrated in Figure 1.1(b), has $2^3 - 1 = 7$ nodes and takes the form:

$$\begin{aligned} |\Psi\rangle = \sum_{\sigma_1, \ldots, \sigma_7} \sum_{b_2, b_3, b_4, b_5, b_6, b_7} & T[1]^{\sigma_1}_{b_2, b_3} T[2]^{\sigma_2}_{b_2, b_4, b_5} T[3]^{\sigma_3}_{b_3, b_6, b_7} \\ & \times T[4]^{\sigma_4}_{b_4} T[5]^{\sigma_5}_{b_5} T[6]^{\sigma_6}_{b_6} T[7]^{\sigma_7}_{b_7} \left|\sigma_1 \sigma_2 \ldots \sigma_7\right\rangle. \end{aligned} \tag{1.4}$$

1.2.2 Tree Tensor Network Operators (TTNO)

If one thinks of TTN as parameterizing a large "vector" in a high-dimensional space, then TTNO is the generalization to the case of a "matrix" acting in the same space. The construction of the TTNO representation of the Hamiltonian and observables is an essential starting point of the most TN-based quantum simulations such as *Density Matrix Renormalization Group* (DMRG) [55], *Time-Dependent Variational Principle* (TDVP) [1], and *Basis Update and Galerkin Integrator* (BUGI) [56]. For sufficiently small systems, we can construct the complete matrix presentation of a Hamiltonian and, from there, obtain a TTNO by using singular value decompositions (SVD). The resulting TTNO will then have the minimal possible bond dimension due to the truncation of zero-valued singular values. However, carrying out full SVDs becomes infeasible for larger systems, motivating the development of more scalable construction schemes.

In the context of *Matrix Product Operator* (MPO) several methods have been explored for constructing tensor network operator representations. A widely used approach is based on finite state automata [57], where Hamiltonians with structured interactions can be systematically converted into MPO representations. Other approaches involve bipartite graph theory [58], compression methods [59] and canonical orthogonalization [60] also have been utilized.

Adapting such ideas to tree topologies has recently led to TTNO construction algorithms founded on state diagrams, which yields a TTNO that often exhibits near-optimal bond dimensions when benchmarked against the fully SVD-compressed version. These techniques have been applied to nearest-neighbor, long-range, and open-system Hamiltonians—such as spin chains coupled to bosonic bath sites—where a tree connectivity can substantially lower the overall bond dimension of the Hamiltonian compared to linear MPO representations [61]. More recent work [62] has advanced this approach by integrating bipartite graph theory with *Symbolic Gaussian elimination* (SGE) preprocessing step, which preserves the symbolic nature of the Hamiltonian while enabling the algorithm to handle cases with uniform coefficients—scenarios where the original method was suboptimal. Demonstrated through experiments on random Hamiltonians, lattice models, and a realistic cavity-molecule system, SGE-enhanced approach consistently achieves optimal bond dimensions. In this thesis, we employ this method as a key tool for constructing TTNO representations of quantum Hamiltonians.

Just as the TTNS expresses a quantum state as a contraction of low-rank tensors, a quantum operator $\hat{O}$ can be represented as:

$$\hat{O} = \sum_{\{b\},\{\sigma\},\{\sigma'\}} W[1]_{B_1}^{\sigma_1',\sigma_1} W[2]_{B_2}^{\sigma_2',\sigma_2} \cdots W[N]_{B_N}^{\sigma_N',\sigma_N} \Big|\sigma_1' \, \sigma_2' \, \cdots \, \sigma_N'\Big\rangle\Big\langle\sigma_1 \, \sigma_2 \, \cdots \, \sigma_N\Big|, \tag{1.5}$$

where each tensor $W[i]$ is expanded in the basis $|\sigma_i\rangle$ and carries two physical indices σ_i' (output) and σ_i (input). For each operator tensor $W[i]$, let $B_i = \{b_{j_1}, b_{j_2}, \ldots, b_{j_k}\}$ denote the set of virtual bonds attached to node i, with corresponding bond dimensions $\chi^{j_1}, \chi^{j_2}, \ldots, \chi^{j_k}$. Then the tensor $W[i]$ has the shape $\left(\chi^{j_1}, \, \chi^{j_2}, \, \ldots, \, \chi^{j_k}, \, d^i_{\text{out}}, \, d^i_{\text{in}}\right)$. The physical indices $(d^i_{\text{out}}, d^i_{\text{in}})$ contract according to the standard quantum mechanical rules for operators.

1.3 TTN Operations

1.3.1 Operator-State Contraction

For a single site i, when the operator tensor $W[i]$ acts on the state tensor $T[i]$, the input physical index of $W[i]$ (with dimension d^i_{in}) contracts with the physical index of $T[i]$.

$$\left(W[i] \, T[i]\right)_{B_i''}^{\sigma_i'} = \sum_{\sigma_i} W[i]_{B_i'}^{\sigma_i',\sigma_i} \, T[i]_{B_i}^{\sigma_i}, \tag{1.6}$$

Assuming $T[i]$'s shape: $\left(\chi^{j_1}, \, \chi^{j_3}, \, \ldots, \, \chi^{j_k}, \, d^i\right)$ and $W[i]$'s shape: $\left(\chi'^{j_1}, \, \chi'^{j_3}, \, \ldots, \, \chi'^{j_k}, \, d^i_{\text{out}}, \, d^i_{\text{in}}\right)$, the shape of the result $W[i] \cdot T[i]$ is: $\left(\chi'^{j_1} \cdot \chi^{j_1}, \, \chi'^{j_3} \cdot \chi^{j_3}, \, \ldots, \, \chi'^{j_k} \cdot \chi^{j_k}, \, d^i_{\text{out}}\right)$.

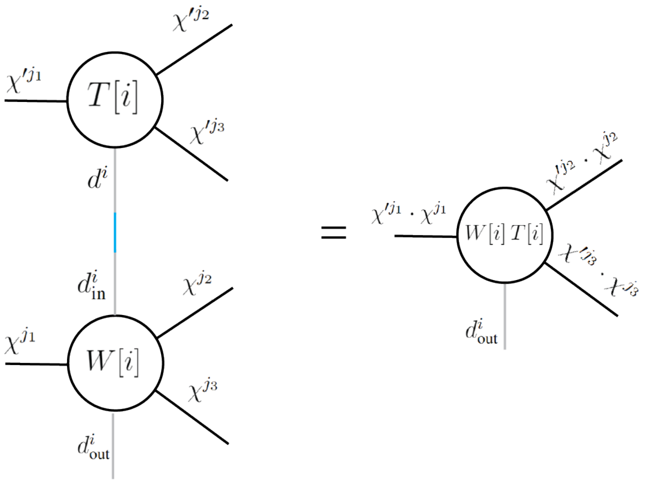


Figure 1.2: Local Operator-State Contraction.

If the TTNO structure have the same tree topology as the TTNS, i.e., corresponds to the same graph G, the TTNO can be applied to the TTNS by contracting all local tensors. So that the new state is given by

$$|\Psi_{\text{new}}\rangle \; = \; \hat{O}\,|\Psi\rangle \; = \; \sum_{\{b''\},\{\sigma'\}} \left(\prod_{i=1}^{N} \sum_{\sigma_i} W[i]_{B_i'}^{\sigma_i',\sigma_i} \, T[i]_{B_i}^{\sigma_i} \right)_{B_i''}^{\sigma_i'} |\sigma_1' \, \sigma_2' \, \cdots \, \sigma_N'\rangle \, . \tag{1.7}$$

In the MPS framework, naively applying an MPO (with same bond dimensions ω for all bonds) to an MPS (with same bond dimension χ for all bonds) without any intermediate compression, all bonds increase from χ to $\chi\omega$, and then restoring them back to $\leq \chi$ at each site by SVD costs $\mathcal{O}(N(\chi\omega d)^3)$. By contrast, the Density Matrix Compression algorithm [63] treats $\hat{O}|\Psi\rangle$ as a large TN and builds partial-trace density matrices, using iterative methods (e.g., Lanczos) that only require applying ρ to vectors without storing or processing a full chain of dimension $(\chi\omega)$ everywhere at once. Sweeping bond by bond in this fashion yields an optimal MPS approximation of $\hat{O}|\Psi\rangle$ with leads to costs more akin to DMRG sweeps. This DMRG-like approach could be adapted to TTN, to increase efficiency and scalability of this operation. However, in our study, the only situation that we need to perform this contraction is in constructing few *Krylov* basis, where the cost of contraction is managed by an approximation strategy.

1.3.2 TTNO Summation

To construct the addition of two TTNOs with the same network topology, let us say TTNO$_3$ = TTNO$_1$ + TTNO$_2$, each local tensor of TTNO$_3$ is formed by contracting the corresponding physical indices of $W_1[i]$ and $W_2[i]$ with two rank-3 Kronecker delta tensors :

$$W_3[i]^{\sigma'_{3i},\sigma_{3i}}_{B_{3i}} = W_1[i]^{\sigma'_{1i},\sigma_{1i}}_{B_{1i}} \; W_2[i]^{\sigma'_{2i},\sigma_{2i}}_{B_{2i}} \; \delta^{(3)}_{\sigma'_{1i},\sigma'_{2i},\sigma'_{3i}} \; \delta^{(3)}_{\sigma_{1i},\sigma_{2i},\sigma_{3i}}. \tag{1.8}$$

The first Kronecker delta, $\delta^{(3)}_{\sigma'_{1i},\sigma'_{2i},\sigma'_{3i}}$, connects the output physical indices of $W_1[i]$, $W_2[i]$, while the second Kronecker delta, $\delta^{(3)}_{\sigma_{1i},\sigma_{2i},\sigma_{3i}}$, connects the input physical indices of $W_1[i]$, $W_2[i]$.

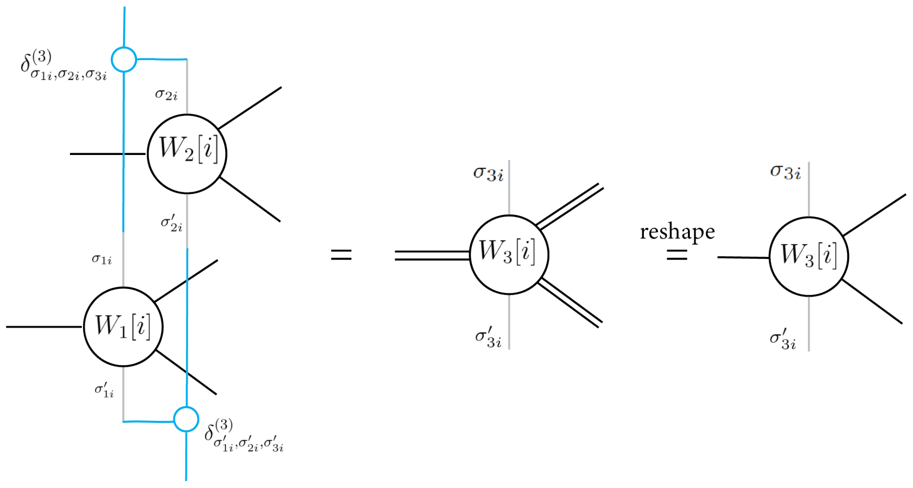


Figure 1.3: Local TTNO Summation.

Each local tensor of TTNO$_3$ is constructed by combining the corresponding local tensors of TTNO$_1$ and TTNO$_2$ through four contractions (two for the bra indices and two for the ket indices).
Assuming $W_1[i]$'s shape: $\left(\chi^{j_1},\, \chi^{j_3},\, \ldots,\, \chi^{j_k},\, d^i_{out}, d^i_{in}\right)$ and $W_2[i]$'s shape: $\left(\chi'^{j_1},\, \chi'^{j_3},\, \ldots,\, \chi'^{j_k},\, d^i_{out}, d^i_{in}\right)$, the shape of the result $W_3[i]$ is $\left(\chi'^{j_1} + \chi^{j_1},\, \chi'^{j_3} + \chi^{j_3},\, \ldots,\, \chi'^{j_k} + \chi^{j_k},\, d^i_{out}, d^i_{in}\right)$.

1.3.3 Expectation Value

The evaluation of expectation value for a 2D lattice wave-function represented by PEPS requires summing over an exponentially large configuration space, making the exact contraction a #P-hard problem. The intermediate tensor dimensions grow exponentially with the bond dimension

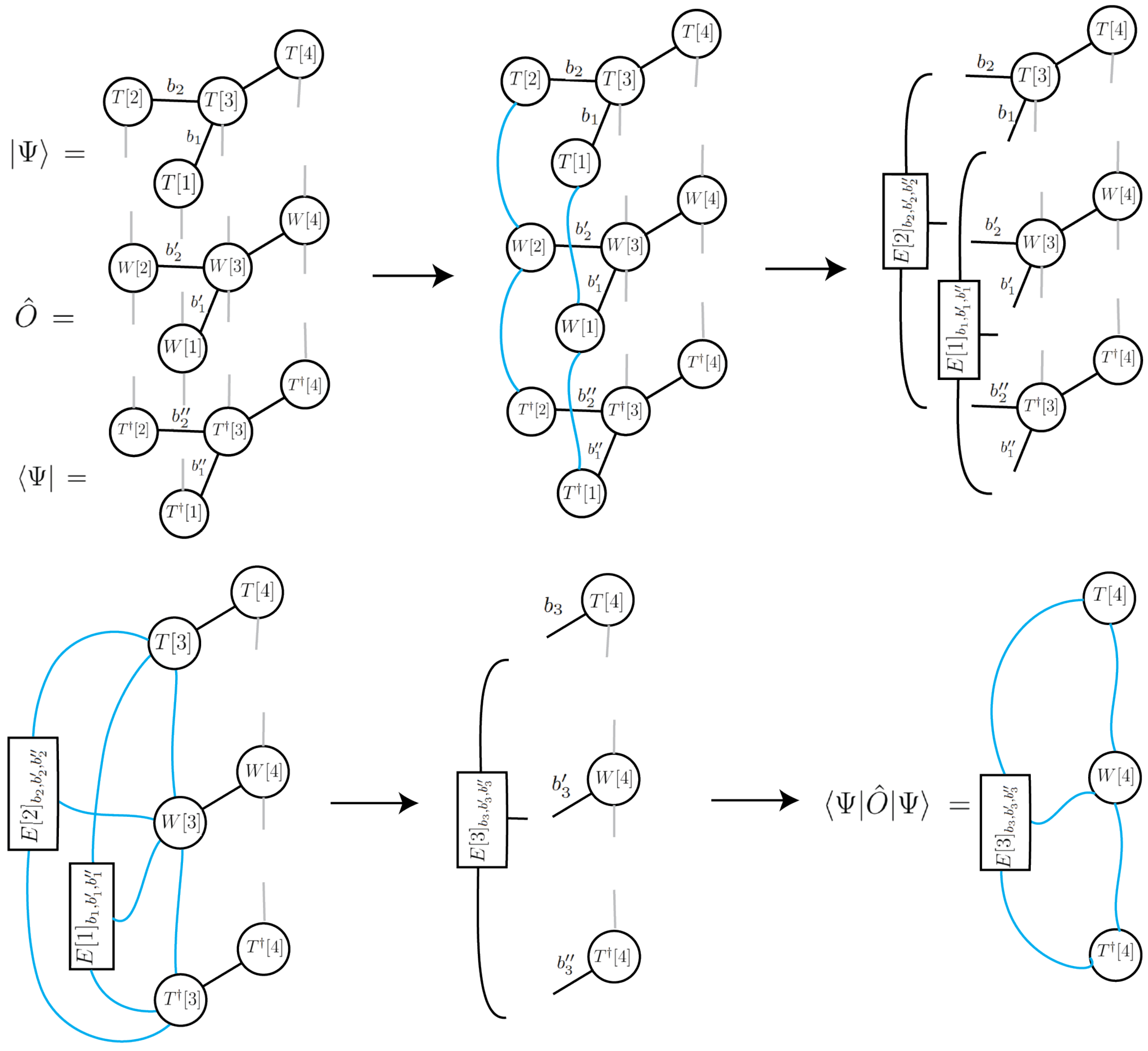

$|\Psi\rangle =$
$\hat{O} =$
$\langle\Psi| =$
$T[1]$ $T[2]$ $T[3]$ $T[4]$ b_1 b_2
$W[1]$ $W[2]$ $W[3]$ $W[4]$ b'_1 b'_2
$T^\dagger[1]$ $T^\dagger[2]$ $T^\dagger[3]$ $T^\dagger[4]$ b''_1 b''_2
$E[2]_{b_2,b'_2,b''_2}$
$E[1]_{b_1,b'_1,b''_1}$
$E[3]_{b_3,b'_3,b''_3}$
b_3 b'_3 b''_3
$\langle\Psi|\hat{O}|\Psi\rangle =$

Figure 1.4: Expectation value contraction scheme for simple TTN with 4 Sites, where site 1 and 2 are leaf nodes, sites 3 is an internal node, and site 4 is the root..

and system size, necessitating the use of approximate methods such as boundary-MPS or corner transfer matrix techniques [64]. In contrast, loop-free topology enable TTNS to be contracted sequentially, which prevents exponential growth in intermediate dimensions, allowing for exact and efficient computation of expectation values in polynomial time.

Given a TTNS $|\Psi\rangle$ and a TTNO $\hat{O}$ (both sharing the same topology), the expectation value is computed as

$$\langle\Psi|\hat{O}|\Psi\rangle = \sum_{\{b\},\{b'\},\{b''\}} \prod_{i=1}^{N} \left(\sum_{\sigma_i,\sigma'_i} T^\dagger[i]^{\sigma'_i}_{B_i} \, W[i]^{\sigma'_i,\sigma_i}_{B'_i} \, T[i]^{\sigma_i}_{B''_i} \right), \tag{1.9}$$

where the summation runs over all virtual bond indices.

As illustrated in a simple example of a TTN with 4 sites in Figure 1.4, the contraction is performed successively from the leaf nodes to the root. The contraction scheme can be summarized as follows:

1. At a Leaf Node l:

$$E[l]_{b_l,b'_l,b''_l} = \sum_{\sigma_l,\sigma'_l} T^\dagger[l]^{\sigma'_l}_{b_l} \, W[l]^{\sigma'_l,\sigma_l}_{b'_l} \, T[l]^{\sigma_l}_{b''_l}, \tag{1.10}$$

where node l connects to its parent with virtual bonds b_l .

2. At an Internal Node i: Let the child nodes of i be denoted by $c \in \text{children}(i)$ and $B_{c\to p} \equiv \{b \in B_c \mid b \text{ is the bond connecting node } c \text{ to its parent}\}$.

$$E[i]_{B_i,B'_i,B''_i} = \sum_{\substack{B_{c\to p}\\ B'_{c\to p}\\ B''_{c\to p}}} \left(\sum_{\sigma_i,\sigma'_i} T^\dagger[i]^{\sigma'_i}_{B_i}\, W[i]^{\sigma'_i,\sigma_i}_{B'_i}\, T[i]^{\sigma_i}_{B''_i} \prod_{c\in\text{children}(i)} E[c]_{B_c,B'_c,B''_c} \right), \tag{1.11}$$

3. At the Root Node r: The final contraction yields the scalar expectation value:

$$\langle\Psi|\hat{O}|\Psi\rangle = \sum_{\substack{B_{c\to p}\\ B'_{c\to p}\\ B''_{c\to p}}} \left(\sum_{\sigma_r,\sigma'_r} T^\dagger[r]^{\sigma'_r}_{B_r}\, W[r]^{\sigma'_r,\sigma_r}_{B'_r}\, T[r]^{\sigma_r}_{B''_r} \prod_{c\in\text{children}(r)} E[c]_{B_c,B'_c,B''_c} \right) \tag{1.12}$$

Remark. Given two TTNS, $|\Psi_1\rangle$ and $|\Psi_2\rangle$, with the same tree structure, their inner product :

$$\langle\Psi_2|\Psi_1\rangle = \sum_{\{b\},\{b'\}} \prod_{i=1}^{N} \left(\sum_{\sigma_i,\sigma'_i} (T[i]^\dagger)^{\sigma'_i}_{B_i}\, T[i]^{\sigma_i}_{B'_i} \right) \tag{1.13}$$

can be computed by following the same scheme without the operator tensors $W[i]$.

1.4 Canonical Form

The representation of a quantum state as a Tensor Network State (TNS) is inherently non-unique, i,e., each virtual bond b between two tensors $T[i]$ and $T[j]$ in the network can be multiplied by an arbitrary invertible matrix G on and its inverse G^{-1}, leaving the global contraction unchanged [65]. Formally,

$$T[i] \longrightarrow T[i]\cdot G \qquad T[j] \longrightarrow G^{-1}\cdot T[j] \tag{1.14}$$

where G is $\chi^d \times \chi^d$ (the dimension of bond d). Such transformations, known as gauge transformations, reflect the many equivalent ways of factorizing the same global tensor $\Psi^{\sigma_1\cdots\sigma_N}$.

In practice, one fixes the gauge frequently through QR or *Singular Value Decomposition* (SVD) by imposing orthogonality condition on local tensors. Such transformations facilitate the computation of linear equations that arise in variational algorithms by eliminating many degrees of freedom through distribution of singular values, thus stabilizing iterative algorithms [66]. Moreover, gauging TNS into a canonical form speeds up tensor contractions, minimizes bond dimensions via systematic truncations, and provides direct access to *Schmidt decompositions* across sub-tree bipartitions [67].

The canonical form representation of a TN state relies on the ability to split a tensor via a *Schmidt decomposition*, which is not straightforward in PEPS due to the presence of loops. Cutting a single bond in PEPS does not divide the network into two independent pieces, making it non-trivial to define orthonormal basis states simultaneously for all bond indices [68]. On the other hand, TTNS are loop-free, allowing the tensors to be transformed into a canonical form, where the tensors satisfy orthogonality conditions.

1.4.1 1-Site Canonical Form

Consider a TTN state

$$|\Psi\rangle = \sum_{\{\sigma\},\{b\}} \mathrm{Tr}(\prod_i T[i]^{\sigma_i}_{B_i})|\{\sigma\}\rangle \tag{1.15}$$

Let B_i denote the set of bond indices connecting node i to its neighbors. If we single out one bond α_i as the orthogonality direction and collect all other bonds (together with the physical index σ_i) into $\Gamma_i = \{\sigma_i\} \cup \Big(B_i \setminus \{\alpha_i\}\Big)$, then the local tensor $T[i]$ is regarded as a linear map

$$T[i] : \bigotimes_{x\in\Gamma_i} V_x \longrightarrow V_{\alpha_i}, \tag{1.16}$$

where V_x is the vector space associated with each index $x \in \Gamma_i$ and V_{α_i} is the vector space corresponding to the orthogonality direction α_i. We say that $T[i]$ is orthogonalized toward α_i if it satisfies the following condition:

$$\sum_{\Gamma_i} T[i]^*_{\alpha_i,\Gamma_i}\, T[i]_{\Gamma_i\,\alpha_i'} \;=\; \delta_{\alpha_i,\alpha_i'}, \tag{1.17}$$

Equivalently, when $T[i]$ is reshaped as a matrix with row indices corresponding to Γ_i and column index α_i, we have $T[i]^\dagger\, T[i] \;=\; I_{V_{\alpha_i}}$, ensuring that the columns of $T[i]$ form an orthonormal set in V_{α_i}.

Graphical Notation : The orthogonalized tensors are depicted by triangle where the line connecting from vertex shows the orthogonality direction. In this example, the T tensor is orthogonalized, as the contraction of the tensor with its conjugate transpose yields the identity matrix.

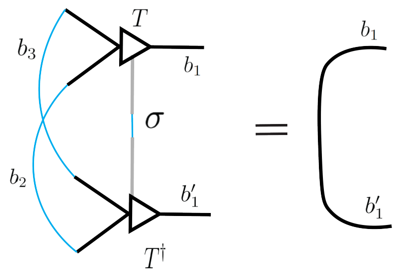


Figure 1.5: Graphical Notation of Orthogonalized Tensor.

We now describe how to transform a local tensor $T[i]$ into an orthogonalized form along a chosen linking bond $\alpha_{i\leftrightarrow j}$ connecting node i to a neighboring node i. In what follows, let B_i be the set of all virtual bond indices at node i and define $\Gamma_i = \{\sigma_i\} \cup \Big(B_i \setminus \{\alpha_{i\to j}\}\Big)$,

The procedure, as illustrated in Figure 1.6, consists of the following steps:

1. Reshape the tensor $T[i]$ into a matrix $M[i]$ with row indices corresponding to Γ_i and column index corresponding to $\alpha_{i\to j}$.
2. Perform a QR decomposition on $M[i]$ and unfold $Q[i]$ back into its higher-order tensor form with new index r that runs over the intermediate dimension:

$$T[i]_{\Gamma_i,\alpha_{i\to j}} = Q[i]_{\Gamma_i,r}\, R[i]_{r,\alpha_{i\to j}} \tag{1.18}$$

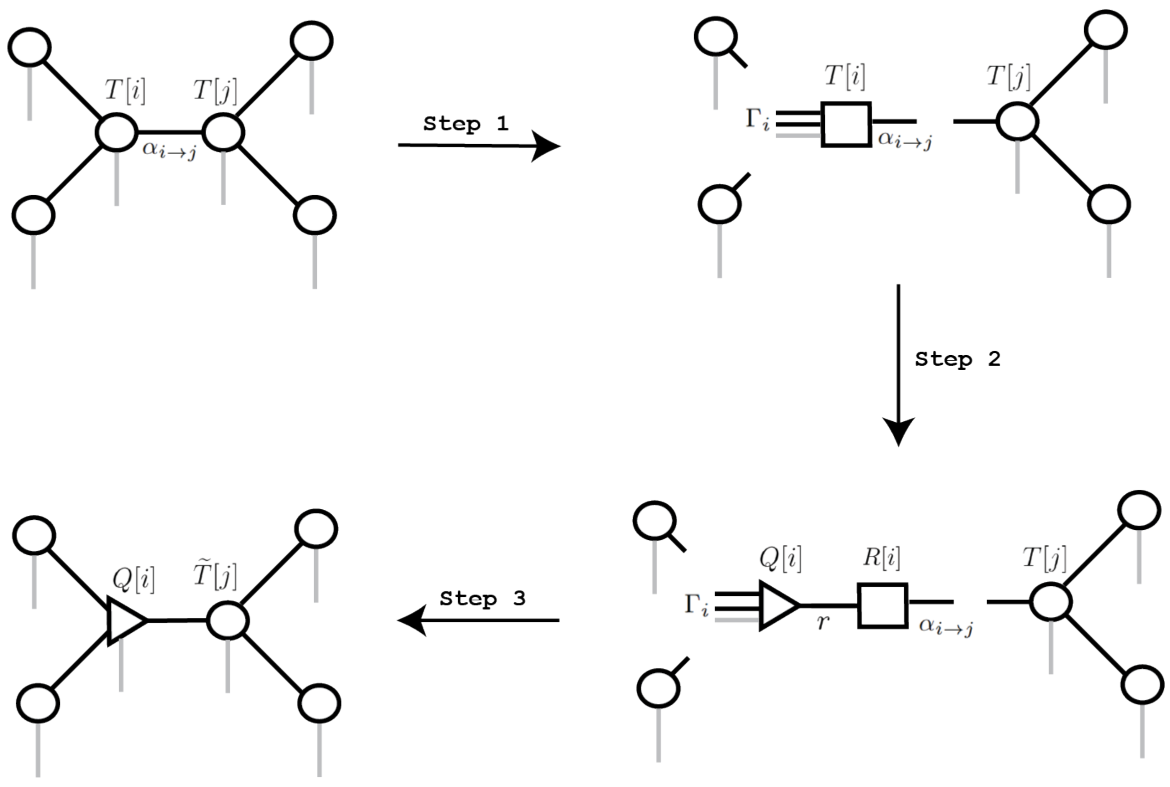


Figure 1.6: Imposing Orthogonality Condition Steps

Then, by construction, $Q[i]$ satisfies the orthogonality condition:

$$\sum_{\Gamma_i} Q[i]^*_{r,\Gamma_i}\, Q[i]_{\Gamma_i,r'} \;=\; \delta_{r,r'}. \tag{1.19}$$

Thus, update the tensor $T[i]$ with $Q[i]$.

3. Contract the matrix $R[i]$ with the neighboring tensor $T[j]$ along the linking bond:

$$\widetilde{T}[j] \;=\; \sum_{b_{i\leftrightarrow j}} R[i]_{r,\,b_{j\to i}} T[j]_{b_{j\to i},\Gamma_j}, \tag{1.20}$$

where $\Gamma_j = \{\sigma_j\} \cup \Big(B_j \setminus \{\alpha_{j\to i}\}\Big)$.
Finally, update $T[j]$ with $\widetilde{T}[j]$.

Note that, choosing a particular bond as the orthogonality direction determines the specific partitioning of the tensor's indices and hence the direction in which orthogonality is enforced.

Definition 1.4.1. A TTN is in *1-site canonical form* [69] with *orthogonalization center* at node c if all other tensors $T[i]$ satisfy the orthogonality constraint toward the node c.

Example. To clarify the concept, let us consider the norm of the state that is preserved after transformation to canonical form and can be simply computed as the norm of the orthogonality center tensor $T[c]$.

$$\langle\Psi|\Psi\rangle = \mathrm{Tr}(T[c]^\dagger \cdot T[c]) \tag{1.21}$$

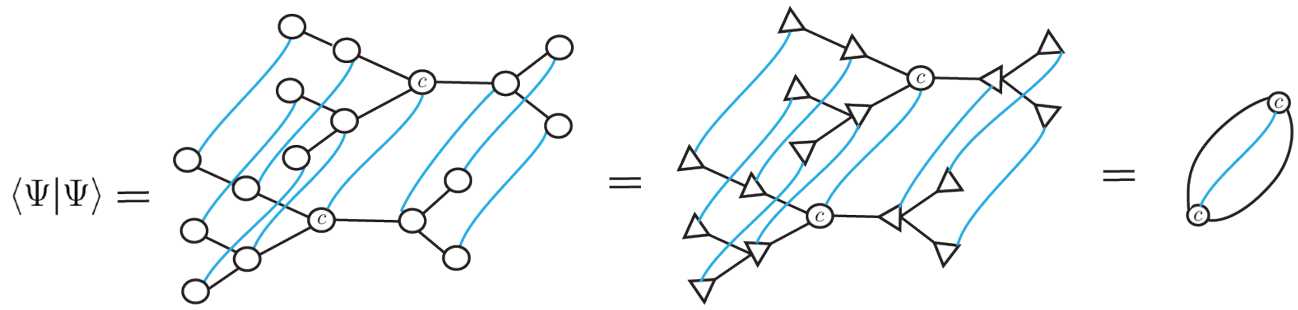


Figure 1.7: TTN Inner Product in *1-Site Canonical Form.*

1.4.2 QR Canonical Transformation

The canonical transformation exploits gauge freedom, by imposing Orthogonality constraint through a so-called pulling-through approach [70] which proceeds with sequential QR decompositions through the network, i.e, moving from the leaves toward a chosen *orthogonalization center* c. The following algorithm describes this procedure :

Algorithm 1: Transform TTN to *1-Site Canonical Form*

```
Input: TTN with tensors $T[i]$ for nodes $i = 1, \dots, N$, orthogonality center $c$
Output: 1-Site Canonical Form with orthogonality center $c$
1  distance ← Compute distances from all nodes i to c such that distance[i] = distance from node i to c
2  Maximum distance in the tree = $D \leftarrow \max_i$ distance$[i]$
3  for $d \leftarrow D$ to 1 do
4      foreach node $i$ with distance$[i] = d$ do
5          Neighbor with smallest distance = $j \leftarrow \arg\min_{k\in\mathrm{neighbors}(i)}$ distance$[k]$
6          Let $\Gamma_i = \{\sigma_i\} \cup$ (All virtual bond indices of $T[i] \setminus \{b_{i\to j}\}$)
7          $M[i] \leftarrow T[i].reshape(\prod_{x\in\Gamma_i} x, b_{i\to j})$
8          $Q[i]_{\Gamma_i, r}, R[i]_{r, b_{i\to j}} \leftarrow$ QR(M[i]) and unfold Q to T's original shape
9          Update $T[i] \leftarrow Q[i]_{\Gamma_i, r}$
10         Let $\Gamma_j = \{\sigma_j\} \cup$ (All virtual bond indices of $T[j] \setminus \{b_{j\to i}\}$)
11         Update $T[j] \leftarrow \sum_{b_{i\leftrightarrow j}} R[i]_{r, b_{j\to i}} T[j]_{b_{j\to i}, \Gamma_j}$ // new bond index is now $r$
12     end
13 end
14 The TTN is now in 1-site canonical form with orthogonality center $c$.
```

Definition 1.4.2. The *distance* of two nodes in a tree is the number of edges in the unique simple path between the nodes.

Example. Here we illustrate the above steps for a simple TTN.

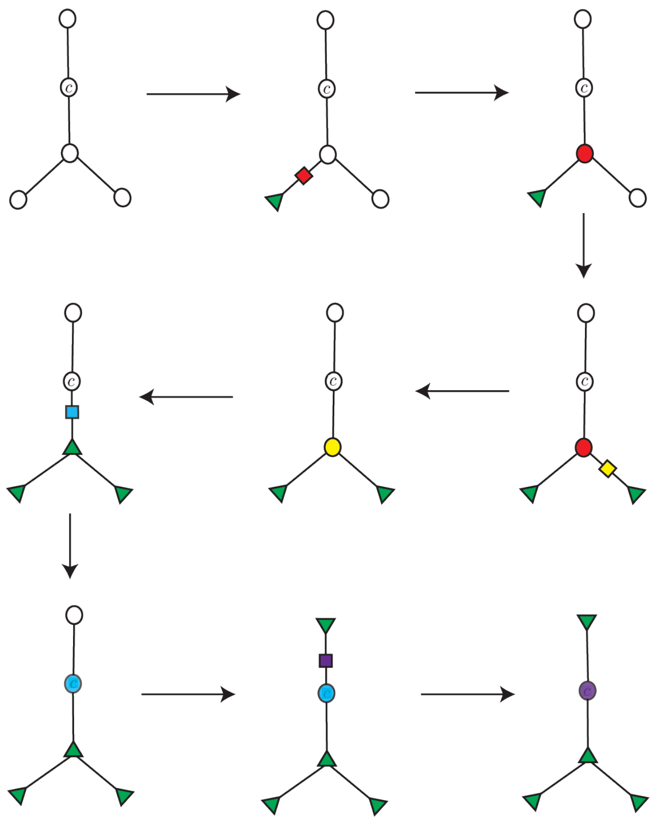


Figure 1.8: *1-Site Canonical Tran*sformation Steps

Starting from a TTN in *1-Site Canonical Tran*, let us say at node i, which has several neighboring nodes $\{j_1, j_2, \ldots, j_k\}$, if we impose orthogonality condition on the tensor $T[i]$ along a bond $b_{i\leftrightarrow j_l}$, then the TTN would be still in *1-site canonical form* with new orthogonalization at site j_k.

1.4.3 SVD Canonical Transformation

The QR decomposition can be replaced with SVD, to facilitate singular values truncation, which is essential for variational algorithms and efficient computations. Given a TTN with neighboring tensors $T[i]$ and $T[j]$ connected by a bond $\alpha_{i\leftrightarrow j}$, we can impose orthogonality condition on tensor $T[i]$ very similar to QR-based approach. We first decompose the reshaped tensor $M[i]$ with row indices corresponding to Γ_i and column index corresponding to $\alpha_{i\to j}$.

$$M[i] = U[i]_{\Gamma_i, r} S[i]_{r,r'} V^\dagger[i]_{r', \alpha_{i\to j}} \tag{1.22}$$

where $U[i]$ and $V[i]$ are unitary, and $S[i]$ is a diagonal matrix of singular values. Next, we set $T[i]$ to $U[i]$ after reshaping back to the original tensor form with r as the new bond index. Then Contract $S[i] \cdot V^\dagger[i]$ with the neighboring tensor $T[j]$:

$$\widetilde{T}[j]_{r,\Gamma_j} = \sum_{\alpha_{i\leftrightarrow j}, r'} S[i]_{r,r'} V^\dagger[i]_{r', \alpha_{i\to j}} T[j]_{\alpha_{j\to i}, \Gamma_j} \tag{1.23}$$

Therefore, by implementing these steps in algorithm 1, the TTN can be similarly transformed to *1-site canonical form* with orthogonality center at c.

1.4.4 TTN Virtual Bond Truncation

Bond dimension is not merely a technical detail, but it is fundamentally linked to the physical properties of the system being studied. For instance, in PEPS, the bond dimension must be sufficiently large to capture the area law governing entanglement entropy in quantum systems with short-range interactions [71]. Conversely, in certain scenarios, such as systems that approach the critical point, the bond dimension must scale superpolynomially with system size to maintain an accurate representation [72].

Variations in bond dimension can lead to transitions from area-law to logarithmic scaling of entanglement entropy. This phenomenon is crucial for accurately capturing the physical characteristics of these systems, emphasizing the importance of bond optimization in theoretical models. The mean-field theory proposed by Lopez-Piqueres [73] indicates that entanglement phase transitions can occur in random tree tensor networks, where the bond dimension acts as a control parameter. As the bond dimension is varied, a clear transition is observed between area-law and logarithmic scaling of entanglement entropy. Similarly, Yang et al.'s work [74] on random stabilizer tensor networks verifies this finding, demonstrating varying the bond dimensions we can observe a transition between an area and volume phase with a logarithmic critical point.

In addition to varying bond dimension, Liu et al.[75] discuss the concept of critical bond dimension as a pivotal factor in understanding the entanglement transitions within quantum circuits. Specifically, when the bond dimension q is greater than critical bond dimension q_c, the boundary state exhibits volume-law entanglement, indicating a high degree of entanglement across the system. Conversely, when q falls below q_c, the system transitions to an area-law entangled state, characterized by reduced entanglement. This transition is closely related to the suppression of entanglement due to the measurement of bulk qubits, which alters the entanglement structure of the remaining boundary qubits. The authors emphasize that the critical bond dimension serves as a control parameter that dictates the entanglement phase of the system.

The arrangement of virtual bonds, directly influences the efficiency of quantum circuit simulations. Seitz et al.[76] illustrate how rooted tree tensor networks can be employed to simulate quantum circuits effectively, where the virtual bonds play a crucial role in threading the entanglement through the network. Two-qubit gates are decomposed via SVD, and the resulting virtual bond is threaded through the tree, increasing the edge dimension between the relevant nodes. The method ensures that bond dimensions remain within a predefined limit, which constrains the types of circuits that can be simulated. This work underscores the pivotal role of bond dimension optimization in enhancing the efficiency and capability of TTN-based quantum circuit simulations.

If one applies truncation to the singular values in step (1.22), then all virtual bonds of TTN could be efficiently truncated in the process of transforming a TTN to canonical form.
We use three `svd_truncation_parameters` to control the truncation process: $(\chi_{\max},\ \epsilon_{\text{rel}},\ \epsilon_{\text{total}})$ where $\chi_{\max}$ is the maximum bond dimension, ϵ_{rel} is the relative truncation tolerance, and ϵ_{total} is the absolute (total) truncation tolerance.

Suppose $S = \operatorname{diag}(s_1, s_2, \ldots, s_R)$ with $s_1 \geq s_2 \geq \cdots \geq s_R \geq 0$. Then, the truncation procedure is as follows:

1. Compute the cutoff threshold : $\tau = \max\Big(\epsilon_{\text{rel}}\, s_1,\ \epsilon_{\text{total}}\Big)$
2. Keep only the singular values satisfying $s_k > \tau$. Let the number of such values be D. Then:

- If $D > \chi_{\max}$, keep only the top $\chi_{\max}$ singular values.
- If $D = 0$, then all singular values would have been truncated. As a fallback, keep only s_1, the largest singular value.
- Otherwise, keep D singular values.

3. Optionally renormalize the kept singular values so that their sum matches the sum of the original sum. Specifically, the truncated singular values are rescaled via:

$$s_k \longleftarrow s_k \cdot \frac{\sum_{j=1}^{R} s_j}{\sum_{j\in\text{kept}} s_j}, \quad \forall s_k \in \mathbf{s}_{\text{kept}}. \tag{1.24}$$

Truncation Error : Consider the SVD of tensor T. Let $\chi_{\text{eff}} < R$ be the number of kept singular values. Then the truncated tensor T_{trunc} discards $\sum_{\alpha=\chi_{\text{eff}}+1}^{R} s_\alpha^2$ of the squared norm, so

$$\|T - T_{\text{trunc}}\|_F^2 = \sum_{\alpha=\chi_{\text{eff}}+1}^{R} s_\alpha^2.$$

with the upper bound

$$\|T - T_{\text{trunc}}\|_F^2 \le \min\Big(\epsilon_{\text{total}}^2,\, \epsilon_{\text{rel}}^2\, s_1^2\Big) \cdot \Big(R - (\chi_{\text{eff}} + 1)\Big),$$

which provides the worst-case estimate of the truncation error.

1.4.5 Bond Canonical Form

Definition 1.4.3. The SVD approach naturally leads to the concept of the *bond canonical form*, where the singular values S are retained as an intermediate tensor on a chosen bond, and all other tensors in the TTN are orthogonalized toward that bond.

This form is particularly advantageous for studying entanglement properties, as the singular values directly correspond to the *Schmidt* coefficients across the bond.

Consider a TTN with N nodes and a specific bond h (connecting nodes i and j) selected as the canonical bond. To achieve bond canonical, first orthogonalize all tensors, except those adjacent to h (let us say $T[i]$ and $T[j]$), toward the bond h, using the pulling-through approach (via QR or SVD).

$$|\Psi\rangle = \sum_{\{\sigma\},\{b\}} \widetilde{T}[1]^{\sigma_1}_{B_1} \cdots T[i]^{\sigma_i}_{B_i} T[j]^{\sigma}_{B_j} \cdots \widetilde{T}[N]^{\sigma_N}_{B_N} |\sigma_1\sigma_2\cdots\sigma_N\rangle, \tag{1.25}$$

Next, let $T[i]^{\sigma_i}_{B_i}$ and $T[j]^{\sigma_j}_{B_j}$ be the tensors connected by $h \in B_i \cap B_j$, with $B_i = \{h, b_{i_1}, \ldots, b_{i_{m_i}}\}$ and $B_j = \{h, b_{j_1}, \ldots, b_{j_{m_j}}\}$. Define $\Gamma_i = \{\sigma_i\} \cup \Big(B_i \setminus \{b_k\}\Big)$ and $\Gamma_j = \{\sigma_j\} \cup \Big(B_j \setminus \{b_k\}\Big)$, and reshape the contraction $T[i] \cdot T[j]$ over h into a matrix M with row index corresponding to Γ_i and column index corresponding to Γ_j.

$$M_{\Gamma_i,\Gamma_j} = \sum_h T[i]_{\Gamma_i,h} T[j]_{h,\Gamma_j}. \tag{1.26}$$

Perform SVD on M, and unfold the resulting matrices U and $V^\dagger$ back into original form.

$$M_{\Gamma_i,\Gamma_j} = U[i]_{\Gamma_i,r} S_{r,r'} V^\dagger[j]_{r',\Gamma_j}, \tag{1.27}$$

where $U[i]$ and $V^\dagger[j]$ are unitary, and S is diagonal with non-negative singular values.
The TTN is then restructured as:

1. Impose orthogonality towards site j : $\widetilde{T}[i]_{\Gamma_i,r} = U[i]_{\Gamma_i,r}$,
$$\sum_{\Gamma_i} \widetilde{T}[i]^*_{r,\Gamma_i} \widetilde{T}[i]_{\Gamma_i,r'} = \delta_{r,r'}, \tag{1.28}$$
2. Create diagonal matrix $S_{r,r'}$ on the linking bond,
3. Impose orthogonality towards site i : $\widetilde{T}[j]_{r',\Gamma_j} = V^\dagger[j]_{r',\Gamma_j}$,
$$\sum_{\Gamma_j} \widetilde{T}[j]_{r',\Gamma_j} \widetilde{T}[j]^*_{\Gamma_j,r} = \delta_{r',r}. \tag{1.29}$$

The full state representation in *Bond Canonical form* is:

$$|\Psi\rangle = \sum_{\{\sigma\},\{b\},r,r'} \widetilde{T}[1]^{\sigma_1}_{B_1} \cdots \widetilde{T}[i]^{\sigma_i}_{\Gamma_i \setminus \sigma_i, r} S_{r,r'} \widetilde{T}[j]^{\sigma_j}_{r',\Gamma_j \setminus \sigma_j} \cdots \widetilde{T}[N]^{\sigma_N}_{B_N} |\sigma_1 \sigma_2 \cdots \sigma_N\rangle, \tag{1.30}$$

where all tensors except S are orthogonal toward h, and the contraction over r and r' links the sub-trees through S.

Example. In this form, the norm is simply computed as

$$\langle\Psi|\Psi\rangle = \mathrm{Tr}(S^\dagger S) = \sum_r S^2_{r,r}, \tag{1.31}$$

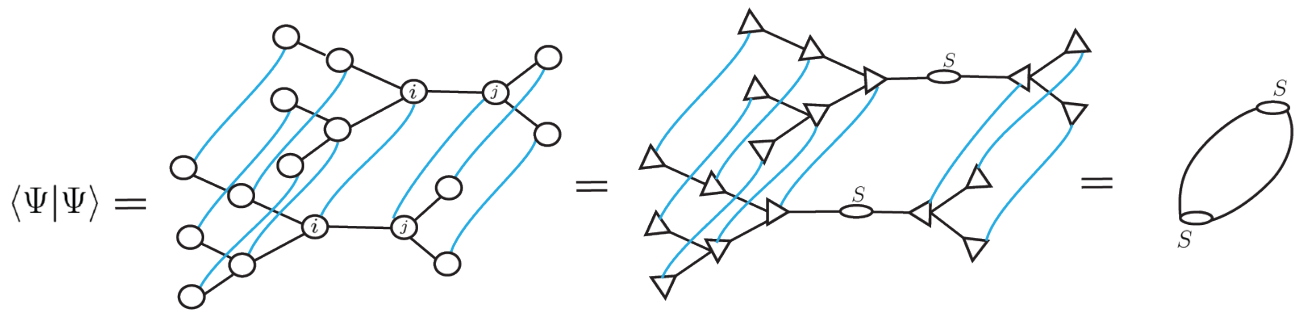


Figure 1.9: TTN Inner Product in *Bond Canonical Form.*

It is worth noting that during this work, we will not need to explicitly transform the TTN to *bond canonical form*, but the concept will manifest in the Time-Dependent Variational Principle (TDVP) algorithm.

2. Time Dependent Variational Principle (TDVP)

This chapter explores the adaptation of TDVP to TTN, presenting a powerful approach to simulate the dynamics of quantum many-body systems within a variational framework. At its core, TDVP operates by evolving quantum states on a TTN manifold, utilizing the tangent space to project the *Schrödinger* equation into a low-dimensional subspace defined by 1-site and 2-site projectors, ensuring globally optimized truncation and preservation of unitarity. We introduce the foundational formalism, detailing how the tangent space captures infinitesimal state deformations. The chapter provides practical tools for implementation, including the TDVP algorithm with its sweeping mechanism, environment caching to enhance computational efficiency, and an *orthogonalization path* strategy to manage gauge invariance across the tree structure. Building on this foundation, we propose using spanning trees for lattice systems, and try to identify optimal TTN configurations that balance computational cost and physical accuracy.

2.1 Introduction

For the simulation of quantum many-body systems, the *Time-Evolving Block Decimation* (TEBD) [77], *Time-Dependent Density Matrix Renormalization Group* (tDMRG) [78] and *Krylov*-based approaches [79] have been extensively utilized.

TEBD relies on the *Trotter-Suzuki* decomposition of the global evolution operator e^{-iHt} into a product of local gates, which are then applied to the TNS. The bond dimensions are then truncated based on a local *Schmidt decomposition* to minimize the *Frobenius norm* error locally: $|| \Psi_{updated}\rangle - |\Psi_{truncated}\rangle \|_F$. Implementing trotterization for Hamiltonians with long-range interactions, requires numerous time-consuming swap gate operations and high-order *Trotter decomposition.* One approach to treat long-range couplings is to construct MPO approximation of the time-evolution operator [40]. TEBD also Struggles with long-time evolution, due to entanglement growth, and energy drift is expected due to *Trotter* errors accumulation.

A more advanced approach known as tDMRG incorporates the TEBD simulation algorithm in the DMRG framework; It was originally focused on real-time dynamics in strongly correlated systems and later extended to finite-temperature dynamics via purification techniques. This method opens the possibility for different time evolution strategies (e.g. *Lanczos*) and more sophisticated variational optimization methods (e.g. functional energy minimization $\min_\psi \left|\psi - e^{-iHt}|\psi_0\rangle\right|^2$) instead of predetermined truncation parameters to ensures that the truncation error is minimized globally within the variational space.

And the *global Krylov method* simply translates the *Lanczos* formalism [80] for unitary time evolution to TNS, i.e. one can approximate the action of the time evolution operator on the initial state

$$e^{-iH\delta}|\psi_0(t)\rangle = |\psi_0(t+\delta)\rangle \approx \sum_{l=0}^{k-1} \frac{(-i\Delta t)^l}{l!} \hat{H}^l |\psi(t)\rangle \tag{2.1}$$

in a truncated manner. Specifically, The full Hamiltonian could be mapped onto *Krylov* space resulting in a tridiagonal effective Hamiltonian, which can be easily diagonalized and it can be shown that the extremal eigenvalues are good approximate to the extremal eigenvalues of the original system [81]. A typical problem of *Krylov* subspace methods [82] is the need to represent potentially highly entangled *Krylov* vectors (typically much more entangled than the actual time-evolved state) as TNS. If many *Krylov* vectors are desired, truncation errors affecting the orthogonality of the basis vectors do not simply add to the overall error, but may quickly degrade the overall quality of the *Krylov* space, leading to a poor result. Also, If one uses a simple *Gram-Schmidt* procedure to orthogonalize vectors by successive additions of TNS, new truncation errors are introduced during this procedure, which will quite often entail the same problem. When orthogonality is lost, *Krylov* vectors begin to overlap significantly, which increases the entanglement encoded in each vector. The accumulation of entanglement affects not only individual *Krylov* vectors but also the process of constructing the time-evolved state. Since the time evolution involves a linear combination of the *Krylov* vectors, the entanglement from each non-orthogonal vector contributes to the overall entanglement in the resulting state. This makes it increasingly difficult to represent the time-evolved state efficiently within the MPS framework.

A detailed comparison of various time evolution algorithms in Ref.[82] concluded that, despite each method having its own strengths and weaknesses, TDVP stands out as one of the most reliable approaches for time evolution.

2.2 Tangent Space Approach

The tangent space approach was first formalized in 2011 by Haegeman and collaborators [1], who introduced the mathematical framework for time evolution within the tangent space. A major breakthrough happened in 2015 when Lubich et al. proposed the explicit construction of tangent space projectors [83]. Then in 2016, Haegeman, Lubich et al.[84] showed how one could solve the projected *Schrödinger* equation, and since then, the tangent space approach has been widely used in numerous applications. The first motivation for the development of TDVP for TTN was in the study of *Dynamical Mean-Field Theory* (DMFT), where the *Fork Tensor Product States* (FTPS) was used to represent multi-orbital *Anderson Impurity Model* (AIM), as earlier TEBD method struggles with systems that exhibit off-diagonal hybridizations [85].

Let the state Ψ be represented on a tree topology $T = (V, E)$ and let the set of bond dimensions $D = \{D_e \mid e \in E\}$ define a manifold $\mathcal{M}[T, D]$ of TTNS. Consider the time-dependent *Schrödinger* equation

$$i\frac{\partial}{\partial t}\Psi^{\mathcal{M}[T,D]}(t) = \hat{H}\Psi^{\mathcal{M}[T,D]}(t). \tag{2.2}$$

Numerical solutions typically propagate $\Psi^{\mathcal{M}[T,D]}(t)$ in small time steps Δt:

$$\Psi^{\mathcal{M}[T,D]}(t+\Delta t) = \Psi^{\mathcal{M}[T,D]}(t) + \Delta t \cdot \left(-i\hat{H}\Psi^{\mathcal{M}[T,D]}(t)\right) + \mathcal{O}(\Delta t^2). \tag{2.3}$$

This iterative process requires repeated evaluations of $\hat{H}\Psi^{\mathcal{M}[T,D]}(t)$, which can increase the computational cost by producing states that deviate from the initial TTN subspace with fixed a bond dimensions. Tangent space methods address this by recognizing that for small Δt, the change $\Delta\Psi = \Psi^{\mathcal{M}[T,D]}(t+\Delta t) - \Psi^{\mathcal{M}[T,D]}(t)$ lies in a low-dimensional tangent space associated with the manifold $\mathcal{M}[T, D]$. While the entire state changes globally, the tangent space is defined as the linearized space of all infinitesimal deformations that arise from varying one or two tensors at a time, capturing the localized contributions that collectively approximate the full change.

TDVP mitigates these issues by evolving the quantum state within the manifold of tensor network states. In the 1-site TDVP, the bond dimensions remain fixed during evolution, making it computationally efficient but less capable of accommodating rapid entanglement growth. In the 2-site TDVP, the bond dimensions are dynamically expanded in a controlled, optimal way to accommodate entanglement growth. More specifically, the environments surrounding the updated tensors explicitly influence the truncation process. If the entanglement grows and the environment dictates that the current bond dimension is insufficient, TDVP dynamically increases the bond dimension by retaining more singular values. This ensures that the updated tensors remain consistent with the global environment and the truncation errors is minimized.

In TDVP, the evolution is performed variationally by projecting onto the tangent space of the TNS manifold before performing the SVD truncation. This finds the best possible approximation of the evolved state of TNS with fixed bond dimension that respects the structure of the state manifold, and ensures the truncation error is minimized globally within the variational space. As result of evolution being confined to the variational subspace, unitarity is preserved better over long times. However, in TEBD and tDMRG truncation is performed after each gate application, which does not guarantee that the new state remains the best possible variational state within the MPS manifold of fixed bond dimension, and as a result leads to loss of unitarity.

Unlike TEBD, which relies on a *Trotter decomposition* of the Hamiltonian, TDVP directly evolves the state using the full Hamiltonian. This makes TDVP particularly efficient for systems with long-range interactions, where decomposing the Hamiltonian into local gates becomes impractical.

2.2.1 Projection onto the Tangent Space

A critical step in the tangent space method is the construction of a global projector $\hat{\mathcal{P}}$ onto the 1-site/2-site tangent spaces $\mathbb{V}^{\mathrm{1s/2s}}_{|\Psi^{\mathcal{M}[T,D]}\rangle}$:

$$\hat{\mathcal{P}}^{\mathrm{1s/2s}} : \mathcal{H} \to \mathbb{V}^{\mathrm{1s/2s}}_{|\Psi^{\mathcal{M}[T,D]}\rangle} \tag{2.4}$$

The key insight of this approach, as depicted in Figure 2.1, is that the projection onto the tangent space should be applied immediately after the Hamiltonian $\hat{H}$ acts on the state $|\Psi^{\mathcal{M}[T,D]}\rangle$ rather than performing the full time evolution first and then projecting.

$$|\delta\Psi^{\mathcal{M}[T,D]}\rangle = \hat{\mathcal{P}}^{\mathrm{1s/2s}}\hat{H}|\Psi^{\mathcal{M}[T,D]}\rangle \subseteq \mathbb{V}^{\mathrm{1s/2s}}_{|\Psi^{\mathcal{M}[T,D]}\rangle}. \tag{2.5}$$

This ensures that the time evolution runs entirely within the tangent space of the manifold $\mathcal{M}[T, D]$ at every step. The time-dependent *Schrödinger* equation is then solved within the tangent space.

$$i\frac{\partial}{\partial t}|\Psi^{\mathcal{M}[T,D]}(t)\rangle = \hat{\mathcal{P}}\hat{H}|\Psi^{\mathcal{M}[T,D]}(t)\rangle. \tag{2.6}$$

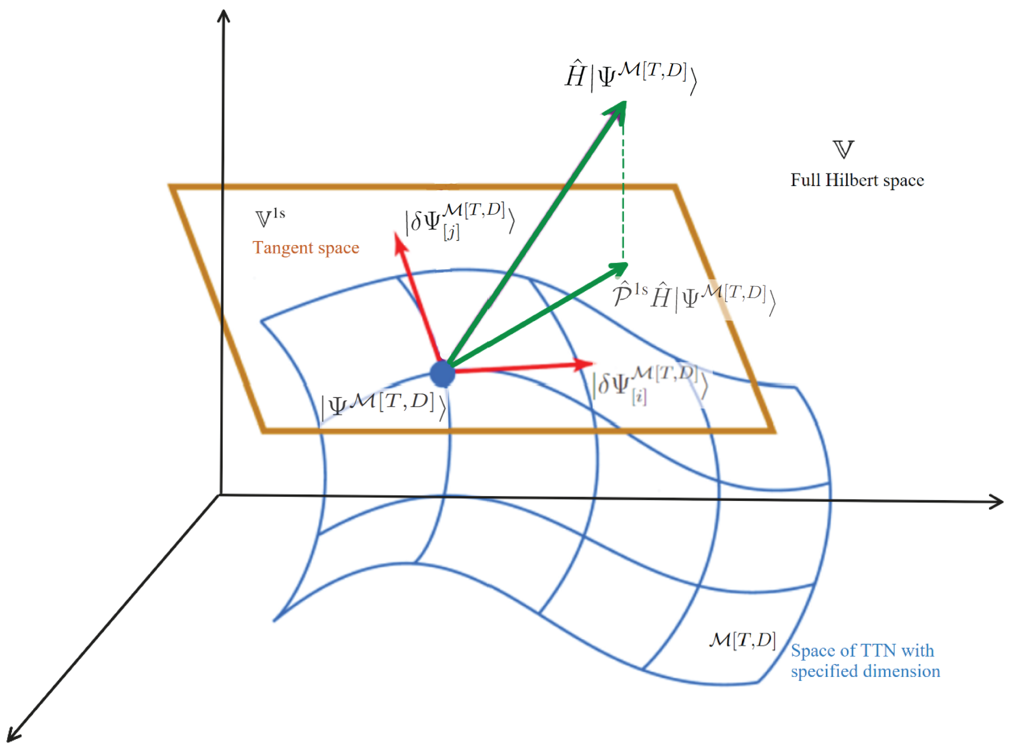

$\hat{H}|\Psi^{\mathcal{M}[T,D]}\rangle$
$\mathbb{V}$
Full Hilbert space
$\mathbb{V}^{1s}$
Tangent space
$|\delta\Psi_{[j]}^{\mathcal{M}[T,D]}\rangle$
$\hat{\mathcal{P}}^{1s}\hat{H}|\Psi^{\mathcal{M}[T,D]}\rangle$
$|\Psi^{\mathcal{M}[T,D]}\rangle$
$|\delta\Psi_{[i]}^{\mathcal{M}[T,D]}\rangle$
$\mathcal{M}[T,D]$
Space of TTN with specified dimension

Figure 2.1: Projection onto the Tangent Space

The 1-site tangent space of state $\Psi^{\mathcal{M}[T,D]}$ is the space of all infinitesimal deformations of the state arising from varying a single tensor $T[i]$ at a time, while keeping all other tensors fixed:

$$|\delta\Psi_{[i]}{}^{\mathcal{M}[T,D]}\rangle = \frac{\partial|\Psi^{\mathcal{M}[T,D]}\rangle}{\partial T[i]} \cdot \delta T[i] = \sum_{\{\sigma\}} \Big(T[1]^{\sigma_1} \cdots \delta T[i]^{\sigma_i} \cdots T[N]^{\sigma_N}\Big)|\sigma_1\sigma_2\cdots\sigma_N\rangle. \tag{2.7}$$

The full 1-site tangent space is then the span of these localized tangent vectors:

$$\mathbb{V}^{1s}_{|\Psi^{\mathcal{M}[T,D]}\rangle} = \text{span}\left\{|\delta\Psi_{[i]}{}^{\mathcal{M}[T,D]}\rangle \ : \ i = 1, 2, \ldots, N\right\}. \tag{2.8}$$

The formal definition of the 1-site tangent space, using local projectors, is:

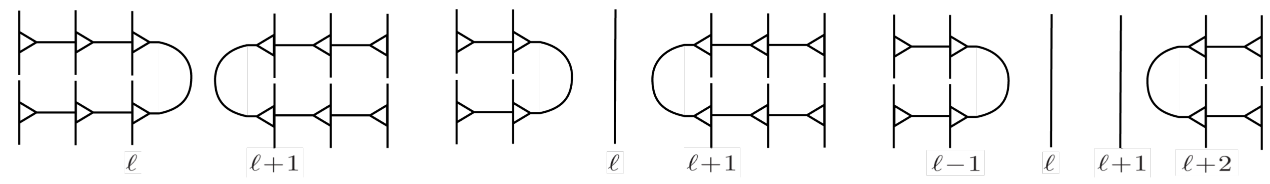
$$\mathbb{V}^{1s} = \text{span}\left\{\text{Im}(\hat{\mathcal{P}}_i^{1s}) \ : \ i = 1, 2, \ldots, N\right\}. \tag{2.9}$$

Here, $\text{Im}(\hat{\mathcal{P}}_i^{1s})$ denotes the image of the projector $\hat{\mathcal{P}}_i^{1s}$, defined as the subspace of the *Hilbert* space onto which $\hat{\mathcal{P}}_i^{1s}$ maps any state.

$$\text{Im}(\hat{\mathcal{P}}_i^{1s}) = \left\{|\phi\rangle \in \mathbb{V} \ : \ |\phi\rangle = \hat{\mathcal{P}}_i^{1s}|\psi\rangle \text{ for some } |\psi\rangle \in \mathbb{V}\right\}. \tag{2.10}$$

where $\mathbb{V}$ is the full *Hilbert* space of the system.

The projectors $\hat{\mathcal{P}}_i^{1s}$ are constructed such that the tangent space captures all deformations at site i by projecting onto the subspace spanned by the derivative $\frac{\partial|\Psi\rangle}{\partial A[i]}$, and preserve gauge invariance, meaning that the tangent space remains well-defined under canonical transformations.



(a) Local Bond Projector (b) 1-Site Local Projector (c) 2-Site Local Projector

Figure 2.2: Local Projectors.

The 2-site tangent space extends this concept by simultaneously allowing variations of two neighboring tensors $T[i]$ and $T[i+1]$, while keeping the rest fixed:

$$|\delta\Psi^{\mathcal{M}[T,D]}_{[i,i+1]}\rangle = \sum_{\{\sigma\}} \Big(T[1]^{\sigma_1}\cdots(\delta T[i]^{\sigma_i}T[i+1]^{\sigma_{i+1}} + T[i]^{\sigma_i}\delta T[i+1]^{\sigma_{i+1}})\cdots T[N]^{\sigma_N}\Big)|\sigma_1\sigma_2\cdots\sigma_N\rangle, \tag{2.11}$$

The full 2-site tangent space is:

$$\mathbb{V}^{2\text{s}}_{|\Psi^{\mathcal{M}[T,D]}\rangle} = \text{span}\left\{|\delta\Psi^{\mathcal{M}[T,D]}_{[i,i+1]}\rangle \;:\; i = 1,2,\ldots,N-1\right\}. \tag{2.12}$$

The formal definition of the 2-site tangent space, using local projectors, is:

$$\mathbb{V}^{2\text{s}} = \text{span}\left\{\text{Im}(\hat{\mathcal{P}}^{2s}_i) \;:\; i = 1,2,\ldots,N-1\right\}, \tag{2.13}$$

These projectors map the state $|\Psi\rangle$ to the space of all possible 2-site variations at sites i and $i+1$.

2.2.2 Local and Global Projectors

The local n_s-site local projector $\hat{\mathcal{P}}^{n_s}_i$ targets variations across n contiguous sites starting at site i. These projectors form a nested hierarchy:

$$\hat{\mathcal{P}}^{\text{b}}_\ell \subset \hat{\mathcal{P}}^{1\text{s}}_\ell \subset \hat{\mathcal{P}}^{2\text{s}}_\ell, \tag{2.14}$$

This reflects the increasing ability of these projectors to capture larger-scale variations in the TNS. Smaller subspaces are fully contained in larger ones which enables refinement of approximations by progressively including more degrees of freedom. Extending to higher site projectors improves the resolution of energy variance and local excitations at the cost of higher computational resources.

The bond projector $\hat{\mathcal{P}}^{\text{b}}_\ell = \hat{\mathcal{P}}^{0\text{s}}_\ell$, illustrated in Figure 2.2a, is defined on bond ℓ (the virtual space between sites ℓ and $\ell+1$).

$$\hat{\mathcal{P}}^{\text{b}}_\ell = \hat{\mathcal{P}}_\ell \hat{Q}_{\ell+1}. \tag{2.15}$$

Where $\hat{\mathcal{P}}_\ell$ is the projector onto the left space $\mathbb{V}_\ell$, and $\hat{Q}_{\ell+1}$ is the projector onto the right space $\mathbb{W}_{\ell+1}$. Thus, $\hat{\mathcal{P}}^{\text{b}}_\ell$ maps the full Hil*Hilbert* bert space $\mathbb{V}$ into:

$$\text{Im}(\hat{\mathcal{P}}^{\text{b}}_\ell) = \mathbb{V}_\ell \otimes \mathbb{W}_{\ell+1}. \tag{2.16}$$

The 1-site projector $\hat{\mathcal{P}}^{1\text{s}}_\ell$, illustrated in Figure 2.2b, projects the full *Hilbert* space $\mathbb{V}$ into the

1-site space at site ℓ. It is defined as:

$$\hat{\mathcal{P}}_{\ell}^{1\text{s}} = \hat{\mathcal{P}}_{\ell-1} \otimes \mathbb{I}_d \otimes \hat{Q}_{\ell+1}. \tag{2.17}$$

Here, $\hat{\mathcal{P}}_{\ell-1}$ is the projector onto the left space at site $\ell - 1$, $\mathbb{I}_d$ is the identity operator acting on the local *Hilbert* space v_ℓ, and $\hat{Q}_{\ell+1}$ is the projector onto the right space at site $\ell + 1$. Thus, $\hat{\mathcal{P}}_{\ell}^{1\text{s}}$ maps the full *Hilbert* space $\mathbb{V}$ into:

$$\text{Im}(\hat{\mathcal{P}}_{\ell}^{1\text{s}}) = \mathbb{V}_{\ell-1} \otimes v_\ell \otimes \mathbb{W}_{\ell+1}. \tag{2.18}$$

The 2-site projector $\hat{\mathcal{P}}_{\ell}^{2\text{s}}$, illustrated in Figure 2.2c, projects the full *Hilbert* space $\mathbb{V}$ into the 2-site space spanning sites ℓ and $\ell + 1$. It is defined as:

$$\hat{\mathcal{P}}_{\ell}^{2\text{s}} = \hat{\mathcal{P}}_{\ell-1} \otimes \mathbb{I}_d \otimes \mathbb{I}_d \otimes \hat{Q}_{\ell+2}. \tag{2.19}$$

Here, $\hat{\mathcal{P}}_{\ell-1}$ is the projector onto the left space at site $\ell - 1$, $\mathbb{I}_d \otimes \mathbb{I}_d$ acts as the identity operator on the 2-site local *Hilbert* space $v_\ell \otimes v_{\ell+1}$, and $\hat{Q}_{\ell+2}$ is the projector onto the right space at site $\ell + 2$. Thus, $\hat{\mathcal{P}}_{\ell}^{2\text{s}}$ maps the full *Hilbert* space $\mathbb{V}$ into:

$$\text{Im}(\hat{\mathcal{P}}_{\ell}^{2\text{s}}) = \mathbb{V}_{\ell-1} \otimes v_\ell \otimes v_{\ell+1} \otimes \mathbb{W}_{\ell+2}. \tag{2.20}$$

The projectors $\hat{\mathcal{P}}_{\ell}^{\text{b}}, \hat{\mathcal{P}}_{\ell}^{1\text{s}}, \hat{\mathcal{P}}_{\ell}^{2\text{s}}$ satisfy the following properties:

- Idempotence:

$$(\hat{\mathcal{P}}_{\ell}^{\text{ns}})^2 = \hat{\mathcal{P}}_{\ell}^{\text{ns}} \tag{2.21}$$

- Mutual Commutativity:

$$[\hat{\mathcal{P}}_{\ell}^{\text{b}}, \hat{\mathcal{P}}_{\ell}^{1\text{s}}] = 0, \quad [\hat{\mathcal{P}}_{\ell}^{\text{b}}, \hat{\mathcal{P}}_{\ell}^{2\text{s}}] = 0, \quad [\hat{\mathcal{P}}_{\ell}^{1\text{s}}, \hat{\mathcal{P}}_{\ell}^{2\text{s}}] = 0. \tag{2.22}$$

  Projectors targeting different subspaces do not interfere with each other's action, enabling simultaneous optimization or decomposition across different sites or bonds without inconsistencies.

The representation of projection of a state with the bond, 1-site, and 2-site local projectors is shown in the following Table 2.1. The projected states are brought into a locally canonical form via isometries enforcing local orthonormality, and preserving physical properties while spanning distinct variational subspaces.

The table 2.2 represent the Local projection of Hamiltonian $\hat{H}$ into the bond, 1-site, and 2-site subspaces, with their finite matrix representation achieved by projecting the Hamiltonian onto the basis states of the corresponding subspaces.

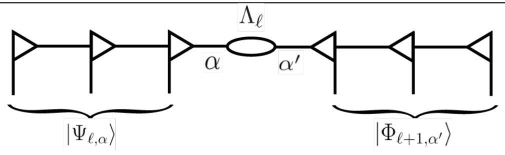
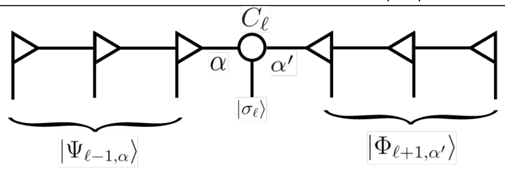
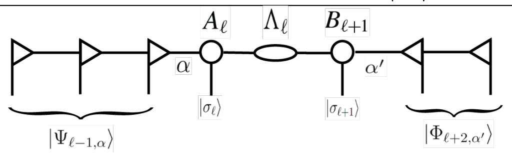
 Λ_ℓ α α' $\lvert\Psi_{\ell,\alpha}\rangle$ $\lvert\Phi_{\ell+1,\alpha'}\rangle$ Bond canonical form of $\lvert\Psi\rangle$	$\lvert\Psi\rangle = \sum_{\alpha,\alpha'} \lvert\Psi_{\ell,\alpha}\rangle \psi^{\mathrm{b}}_{\ell,\alpha\alpha'} \lvert\Phi_{\ell+1,\alpha'}\rangle,$ $\psi^{\mathrm{b}}_\ell = \Lambda_\ell$
 C_ℓ α α' $\lvert\sigma_\ell\rangle$ $\lvert\Psi_{\ell-1,\alpha}\rangle$ $\lvert\Phi_{\ell+1,\alpha'}\rangle$ 1-site canonical form of $\lvert\Psi\rangle$	$\lvert\Psi\rangle = \sum_{\alpha,\alpha',\sigma_\ell} \lvert\Psi_{\ell-1,\alpha}\rangle \psi^{1\mathrm{s}}_{\ell,\alpha\alpha'} \lvert\sigma_\ell\rangle \lvert\Phi_{\ell+1,\alpha'}\rangle,$ $\psi^{1s}_\ell = C_\ell$
 A_ℓ Λ_ℓ $B_{\ell+1}$ α α' $\lvert\sigma_\ell\rangle$ $\lvert\sigma_{\ell+1}\rangle$ $\lvert\Psi_{\ell-1,\alpha}\rangle$ $\lvert\Phi_{\ell+2,\alpha'}\rangle$ 2-site canonical form of $\lvert\Psi\rangle$	$\lvert\Psi\rangle = \sum_{\alpha,\alpha',\sigma_\ell,\sigma_{\ell+1}} \lvert\Psi_{\ell-1,\alpha}\rangle \psi^{2\mathrm{s}}_{\ell,\alpha\alpha'} \lvert\sigma_\ell\rangle \lvert\sigma_{\ell+1}\rangle \lvert\Phi_{\ell+2,\alpha'}\rangle,$ $\psi^{2s}_\ell = A_\ell \Lambda_\ell B_{\ell+1}$

Table 2.1: State Local Projections.

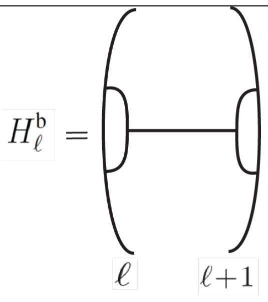
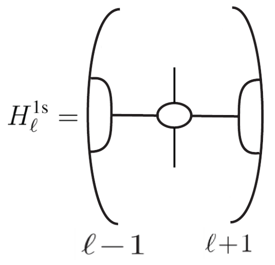
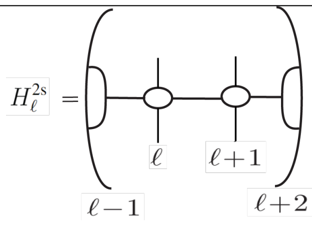
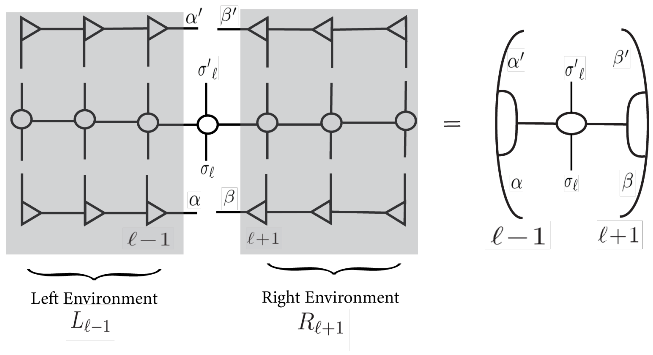
 Effective link Hamiltonian	$\langle\Phi_{\ell+1}\lvert\langle\Psi_\ell\rvert\hat{H}\lvert\Psi_\ell\rangle\rvert\Phi_{\ell+1}\rangle$ $= H^{\mathrm{b}}_\ell = L_\ell R_{\ell+1}$
 Effective 1-site Hamiltonian	$\langle\Phi_{\ell+1}\lvert\langle\sigma_\ell\rvert\langle\Psi_{\ell-1}\rvert\hat{H}\lvert\Psi_{\ell-1}\rangle\lvert\sigma_\ell\rangle\rvert\Phi_{\ell+1}\rangle$ $= H^{1\mathrm{s}}_\ell = L_{\ell-1} W[\ell] R_{\ell+1}$
 Effective 2-site Hamiltonian	$\langle\Phi_{\ell+2}\lvert\langle\sigma_{\ell+1}\rvert\langle\sigma_\ell\rvert\langle\Psi_{\ell-1}\rvert\hat{H}\lvert\Psi_{\ell-1}\rangle\lvert\sigma_\ell\rangle\sigma_{\ell+1}\rangle\rvert\Phi_{\ell+2}\rangle$ $= H^{2\mathrm{s}}_\ell = L_{\ell-1} W[\ell] W[\ell+1] R_{\ell+2}$

Table 2.2: Hamiltonian Local Projections.

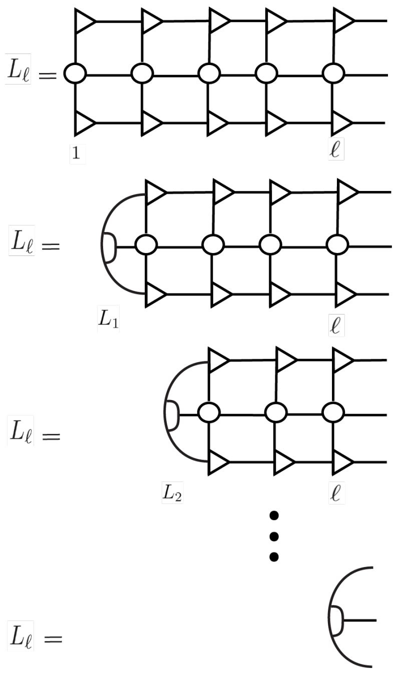
Now, Let us consider the 1-site Hamiltonian before and after the environemts being contracted:



with matrix elements:

$$[H_\ell^{1s}]^{\sigma_\ell \sigma'_\ell}_{\alpha\alpha',\beta\beta'} = [L_{\ell-1}]_{\alpha\alpha'} \, [W_\ell]^{\sigma_\ell,\sigma'_\ell} \, [R_{\ell+1}]_{\beta\beta'} \tag{2.23}$$

The environments are recursively constructed toward the center, and each single step they are saved in the cached to avoid recalculation.



The global n_s-site projector $\hat{\mathcal{P}}_{n_s}$ serves to capture variations in the TNS over n_s contiguous sites across the entire lattice. The naive approach would be defining the global n_s-site projector as $\sum_i \hat{\mathcal{P}}_i^{ns}$. However, Contributions from local projectors $\hat{\mathcal{P}}_i^{ns}$ overlap with those of neighboring sites, leading to double-counting. To resolve this redundancy, the contributions from different local projectors must be orthogonalized. Therefore, addressing this issue, involves systematically combining local projectors to subtract the overlaps. We skip the details of this mathematical derivation, which could be found in the work of Gleis, Li, and von Delft, particularly Sections II and III [86].

The corrected form of the global n_s-site projector, which removes overlaps between local projectors, is

$$\hat{\mathcal{P}}^{\mathrm{ns}} = \sum_{\ell=1}^{L-n+1} \hat{\mathcal{P}}_\ell^{\mathrm{ns}} - \sum_{\ell=1}^{L-n} \hat{\mathcal{P}}_{\ell+1}^{\mathrm{(n\text{-}1)s}}, \tag{2.24}$$

And special cases of this construction are:

- 1-Site Global Projector:

$$\mathcal{P}^{1s} = \sum_{\ell=1}^{L} \mathcal{P}_\ell^{1s} - \sum_{\ell=1}^{L-1} \mathcal{P}_\ell^{b}, \tag{2.25}$$

- 2-Site Global Projector:

$$\mathcal{P}^{2s} = \sum_{\ell=1}^{L-1} \mathcal{P}_\ell^{2s} - \sum_{\ell=1}^{L-2} \mathcal{P}_{\ell+1}^{1s}, \tag{2.26}$$

2.3 TDVP algorithm

The *Schrödinger* equation in 1(2)-site TDVP is then solved within the tangent space by substituting this 1(2)-site global projector (2.25) and (2.26) into (2.6), leading to serious of coupled local *Schrödinger* equations that can be solved in a sweeping fashion, i.e., updating one bond or tensor at a time.

1-Site TDVP (1TDVP)

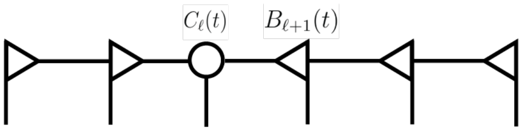
In order to evolve the state one time-step τ, we should successively perform these forward and backward local updates. At site ℓ the 1TDVP local equations are:

$$\begin{aligned} i\dot{C}_l(t) &= H_l^{1s} C_l(t) \\ C_l(t+\tau) &= e^{-iH_l^{1s}\tau} C_l(t) \quad \text{Forward time-step} \end{aligned} \tag{2.27}$$

$$\begin{aligned} i\dot{\Lambda}_l(t) &= -H_l^{b} \Lambda_l(t) \\ \Lambda_l(t-\tau) &= e^{iH_l^{b}\tau} \Lambda_l(t) \quad \text{Backward time-step} \end{aligned} \tag{2.28}$$

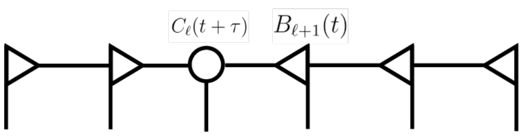
Bellow we demonstrate the steps in one local update, which should be done successively for all sites during sweeps. Starting with state in 1-site canonical form, updating the site ℓ involves the following steps:

Consider two neighboring sites $C_\ell(t)$ and $B_{\ell+1}(t)$



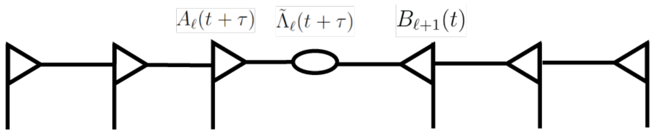
1. Evolve $C_\ell(t)$ forward in time:

$$C_\ell(t+\tau) = e^{-iH_\ell \tau} C_\ell(t)\, B_{\ell+1}(t) \tag{2.29}$$



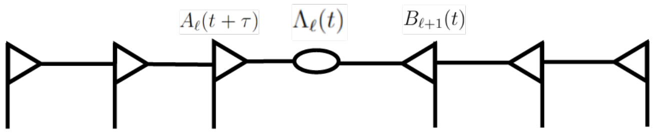
2. Decompose $C_\ell(t+\tau)$:

$$C_\ell(t+\tau) = A_\ell(t+\tau)\Lambda_\ell(t+\tau)\, B_{\ell+1}(t) \tag{2.30}$$



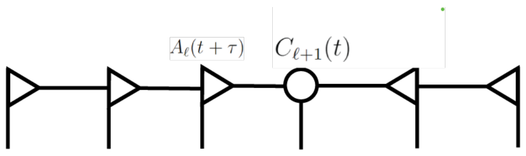
3. Evolve $\Lambda_\ell(t+\tau)$ backward in time:

$$\Lambda_\ell(t) = e^{iH_\ell^b \tau}\Lambda_\ell(t+\tau) \tag{2.31}$$



4. Contract $\Lambda_\ell(t)$ with $B_{\ell+1}(t)$ to shift the orthogonality center to site $\ell+1$:

$$C_{\ell+1}(t) = \Lambda_\ell(t)\, B_{\ell+1}(t) \tag{2.32}$$



5. Update the left environment E_ℓ with $A_\ell(t+\tau)$.

2-Site TDVP (2TDVP)

At site ℓ the 2TDVP equations are:

$$\begin{aligned} i\dot{\psi}_l^{2s}(t) &= H_l^{1s}\psi_l^{2s}(t) \\ \psi_l^{2s}(t+\tau) &= e^{-iH_l^{2s}\tau}\psi_l^{2s}(t) \quad \text{forward time-step} \end{aligned} \tag{2.33}$$

$$\begin{aligned} i\dot{\psi}_{l+1}^{1s}(t) &= -H_{l+1}^{1s}C_{l+1}^{1s}(t) \\ \psi_{l+1}^{1s}(t-\tau) &= e^{iH_{l+1}^{1s}\tau}C_{l+1}^{1s}(t) \quad \text{Backward time-step} \end{aligned} \tag{2.34}$$

2.4 From MPS to TTN

The detailed adaptation of TDVP to general loop-free and finite-size tensor networks with arbitrary tree topologies is presented in the work of Bauernfeind et al. (2020) [85]. They demonstrated how tree-like architecture is utilized to efficiently project the *Schrödinger* equation's evolution onto the TTN manifold. In the following section, we delve into the details of TDVP algorithm on TTN, and investigate strategies to minimize the overal computational cost. In the following, we assume a direct mapping of the physical degrees of freedom of individual sites to the nodes of the TTNS. In contrasts with hierarchical representations that introduce auxiliary tensors for coarse-grained representations, this method encodes entanglement solely through direct connections between physical site. Note that all sites have physical index which would not be depicted in the following figures for simplicity.

In the MPS framework, by maintaining and systematically updating left and right environments

at each sweeping step, the overall computational cost of variational algorithms like DMRG and TDVP can be reduced, by a factor of total number of sites, if the algorithm reuses previously computed contractions. For instance, when moving, from site i to site $i+1$, to the right, only a single environment $L_i \to L_{i+1}$ is needed to be computed and stored, while all left and right environemts toward i and $i+1$ are already available from the previous sweeping steps. This rolling update scheme eliminates redundant calculations by storing intermediate results, which act as partial contractions of the network to the surrounding sites of the target optimization site. The efficiency gain is particularly pronounced where multiple sweeps are required, as the environment tensors effectively serve as a computational cache that preserves the results of expensive tensor contractions across iterations.
In the TTN framework, however, the environment caching becomes more complicated, due to the branching structure of TTNS. Updating a target node j with neighbors $\{n_1, ..., n_k\}$, requires all environments $\{E_{(n_1,j)}, \cdots, E_{(n_k,j)}\}$ toward site j. More precisely, obtaining 1-site effective Hamiltonian at site j requires contractions of these environments:

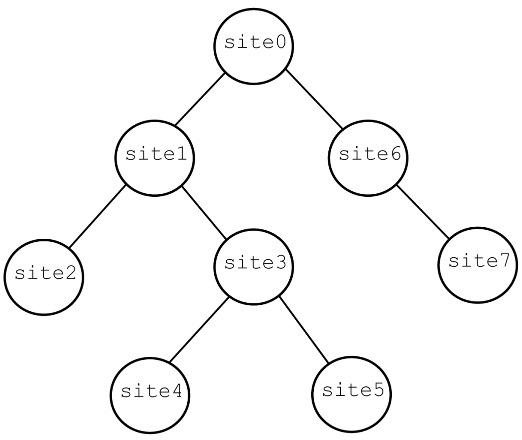
$$H_{\text{eff}}^{[j]} = \sum_{n \in \text{neighbors}(j)} E_{(n,j)} \cdot W[j] \tag{2.35}$$

Similarly, the 2-site effective Hamiltonian between sites j and j', is given by

$$H_{\text{eff}}^{[j,j']} = \sum_{\substack{n \in \text{neighbors}(j) \\ n \neq j' \\ n' \in \text{neighbors}(j') \\ n' \neq j}} E_{(n,j)} \cdot W[j] \cdot W[j'] \cdot E_{(n',j')} \tag{2.36}$$

In addition, the Forward and Backward sweeping path in MPS does not directly translate to TTN, as we face multiple paths to move on at each branching node, and the gauge choice requirement at each local update, adds to this complexity. If TDVP redundantly revisits nodes or subtrees during traversal, it will need to perform QR decompositions multiple times on the same nodes which increase the computational cost. The number of orthogonalisations could be minimized by carefully structuring the order in which nodes are updated, For this purpose, an *update path* $P = [p_1, p_2, \ldots, p_N]$, is constructed to guide the traversal of the tree.

We will use this tree structure as a running example to demonstrate how we manage this challenge in the TDVP algorithm.



This tree consists of:

- Root node: `site0`.
- Leaf nodes: `site2`, `site4`, `site5`, `site7`.

- Internal nodes: `site1`, `site3`, `site6`.

2.4.1 General Rules for *update path* Construction

1. **Main Paths:**
   The key idea is to prioritize processing along a *main path* before handling branches.

   - Construct the *main path* M_{up}^1 from furthest leaf p_1 to the root r:

   $$p_{far} = \arg\max_{\ell \in L} d(r, \ell),$$

   $$M_{up}^1 = [p_{far}, p_2, p_3, \ldots, p_r].$$

     where $d(r, \ell)$ denotes the shortest path length from the root r to the leaf ℓ and L denotes all leaf nodes in the tree T. For an MPS this would be the only path.
   - If the root had $n \geq 3$ children, construct the corresponding $n-1$ other paths $M_{up}^{i>1}$, like M_{up}^1 from other furthest leaf to the root.
   - Construct second *main path* M_{down} from the root r to the furthest leaf p'_{far}:

   $$p'_{far} = \arg\max_{\ell' \in L'} d(r, \ell'),$$

   $$M_{down} = [p_r, p'_2, p'_3, \ldots, p'_{far}].$$

     where L' denotes all leaf nodes that are in the constructed M_{up}^i paths.

   Example: The leaf nodes distances from root `site0` are:

Leaf Node	Distance
`site2`	2
`site4`	3
`site5`	3
`site7`	2

   The starting node is `site4`, as it is the furthest from the root. Starting from `site4`, the path to the root `site0` is:

   $$M_{up}^1 = [\texttt{site4}, \texttt{site3}, \texttt{site1}, \texttt{site0}].$$

   The second *main path* is constructed from the root to the furthest leaf `site7`:

   $$M_{down} = [\texttt{site0}, \texttt{site6}, \texttt{site7}].$$

   $$M_{up}^{i>2} = \varnothing$$

2. **Recursive Branch Traversal:**
   By treating the *main path* as the priority and handling side branches only after the *main path* is resolved, the algorithm effectively separates the tree traversal into manageable chunks. Subtrees (branches) are processed recursively, starting from the leaves of the subtree and working upward, following this logic:

   - Start and add nodes along the path M_{up}, until reaching a branching node.
   - Dive into each branch:

(a) Start at the child nodes of the branching node.
(b) Process all descendants of each child recursively, moving deeper until reaching the leaf nodes.
(c) After processing all subtrees a child, completing the branch in a bottom-to-top order.
(d) continue to the next node in the *main path*.

- The M_{up}^{i} paths for i > 2 should be process likewise before going down with M_{down} path.

Example:

- Add `site4`.
- At `site3`: go up from `site5`.
- At `site1`: go up from `site2`.
- At `site0`: go down to `site6` and then `site7`.

3. **Path Finalization:**

- Combine all traversed subbranches and the *main path* M:

Example:

$$P = [\texttt{site4}, \texttt{site5}, \texttt{site3}, \texttt{site2}, \texttt{site1}, \texttt{site0}, \texttt{site6}, \texttt{site7}].$$

2.4.2 Orthogonalization Path in TDVP

We learned how the TDVP on TTN can systematically runs over a *main path* and subsequently process the branches, however, there is another challenge.

It is important to observe that at each local site update in TDVP algorithm 2.3:

- In step 4 : The *orthogonalization center* is automatically moved to the *next site*.
- In step 5 : The evolved environment $E(\textit{target site}, \textit{next site})$ tensors are computed and stored in the environment cache dictionary.

Hoewever, when the *update path* traverse to the bottom sites, the effective hamiltoninan contsruction at the new *target site* requires:

1. The tensor at the *target site* to be the *orthogonalization center*.
2. All environments towards the *target site* to be computed and stored in the cache dictionary.

The above conditions are managed by associating an *orthogonalization path* to each site in the *update path*.

$$\textit{Orthogonalization Path} = [O_1^{\text{orth}}, O_2^{\text{orth}}, \dots]$$

The *orthogonalization path* has two purposes:

1. Determines the sweep direction.
2. Ensures that the *orthogonalization center* is at the *target site*.

The *orthogonalization path* is constructed as follows:

1. **Initialization:** Start with the *update path*:

$$P = [p_1, p_2, \dots, p_N],$$

where p_1 is the first site to be updated, and p_N is the last.

2. **Compute Intermediate Paths:** For each consecutive pair of sites (p_i, p_{i+1}) in P:

   - Find the shortest path between them: $\text{path}(p_i, p_{i+1})$.
   - Exclude the starting site p_i to avoid redundancy: $p_i^{\text{orth}} = \text{path}(p_i, p_{i+1})[1:]$.

3. **Combine All Intermediate Paths:**
Collect the intermediate paths p_i^{orth} for $i \in \{1, \cdots, N-1\}$.

Example:
Given *update path* $P = [\texttt{site4}, \texttt{site5}, \texttt{site3}, \texttt{site2}, \texttt{site1}, \texttt{site0}, \texttt{site6}, \texttt{site7}]$,

From → To	Full Path	*orthogonalization path*
`site4` → `site5`	[`site4`, `site3`, `site5`]	[`site3`, `site5`]
`site5` → `site3`	[`site5`, `site3`]	[`site3`]
`site3` → `site2`	[`site3`, `site1`, `site2`]	[`site1`, `site2`]
`site2` → `site1`	[`site2`, `site1`]	[`site1`]
`site1` → `site0`	[`site1`, `site0`]	[`site0`]
`site0` → `site6`	[`site0`, `site6`]	[`site6`]
`site6` → `site7`	[`site6`, `site7`]	[`site7`]

The complete *orthogonalization path* for the example tree is:

$$\begin{aligned} \textit{Orthogonalization Path} = &[[\texttt{site3}, \texttt{site5}], [\texttt{site3}], \\ &[\texttt{site1}, \texttt{site2}], [\texttt{site1}], \\ &[\texttt{site0}], [\texttt{site6}], [\texttt{site7}]] \end{aligned}$$

To better understand the TDVP process, we will now illustrate a complete sequence of operations in a single time-step of the first-order 1TDVP algorithm.

For each site p_i in the *update path* P :

- The *next site* would be *Orthogonalization Path*$[i][0]$. In step 5 of 2.3, the Λ_i is then contracted with *next site*. Subsequently, an environment $E(\textit{target site}, \textit{next site})$ is computed and stored in the cache dictionary.

- The *orthogonalization center* is moved along the *Orthogonalization Path*$[i-1][0]$ and the environment tensor is computed accordingly right before local update. This ensures the the existence of environemt needed for the effective Hamiltonian construction in the next local update.

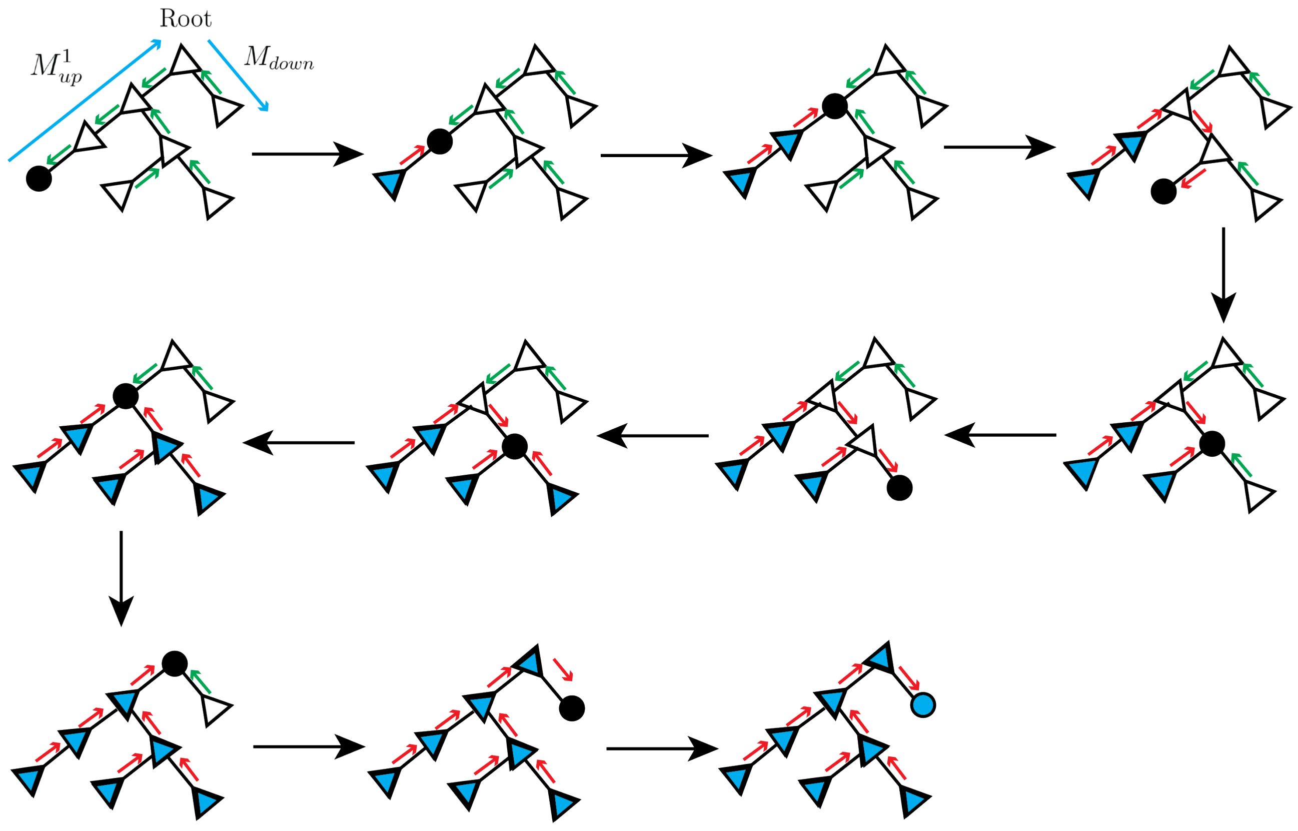


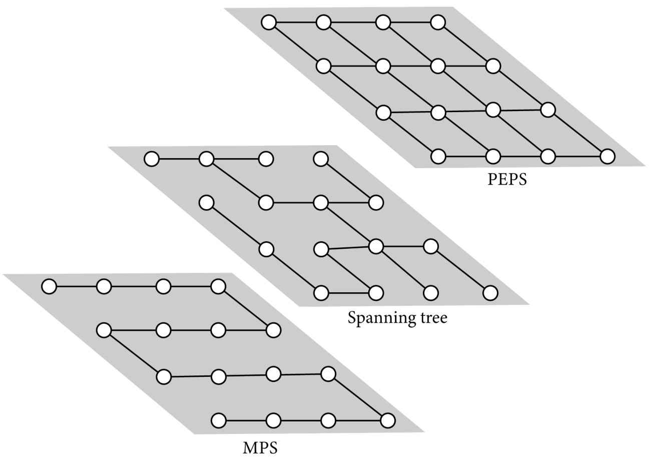
Figure 2.3: Illustrative Example of a Single Time-Step Evolution with First-Order 1-site TDVP. The symbols and arrows in the figure have the following meanings:

- $\longrightarrow$: Direction of Initial Environments (calculated before and at the beginning of each time-step)
- $\longrightarrow$: Direction of new Environments (calculated after each local update)
- ● : The orthogonality center before being updated
- ▷ : Isometric tensor before being updated
- ▶ : Isometric tensor after being updated
- ● : Last tensor in the update-path after being updated

Remark. The second-order TDVP reduces error by employing a symmetric integration scheme that performs forward with $\delta t/2$ and backward sweeps with $\delta t/2$ in each time-step. Consequently, the error scaling improves from $\mathcal{O}(\delta t^2)$ in first-order TDVP to $\mathcal{O}(\delta t^3)$. This effectively cancels higher-order error terms that arise in the integration of the *Schrödinger* equation without requiring smaller time steps, making it particularly suitable for long-time evolution while preserving energy conservation and fidelity. Moreover, in second-order TDVP, during the backward sweep, the intial environemts, that are needed to start the next time-step, are automatically calculated during the backward sweep, which we would had to be accounted for separately in first-order TDVP anyway. In backward sweep, the *update path* is reversed and a new *orthogonalization path* should be, similarly, constructed with the new *update path*.

2.5 Representation of Many-Body Wavefunctions on a Lattice

Tree Tensor Network State (TTNS) can approximates the full many-body wavefunction $|\psi\rangle$ of a lattice system by hierarchically decomposing it into a network of tensors organized along the structure of a spanning tree. A spanning tree is a subgraph of a connected graph that includes all vertices of the graph, is acyclic, and has exactly $|V| - 1$ edges, where $|V|$ is the number of vertices. Formally, if the lattice is represented as a graph $G = (V, E)$, a spanning tree $T = (V, E_T)$ satisfies $E_T \subseteq E$, $|E_T| = |V| - 1$, and T is connected and acyclic. This construction ensures that the connectivity of the original lattice is retained whithout any loops. This could be interpreted as an intermediate approach between snake-like mapping with MPS and PEPS.



To optimize the computational performance of TDVP on a Lattice, we aim to identify configurations that lead to the most efficient execution of algorithm steps in terms of CPU runtime and memory usage. Furthermore, we hypothesize that configurations with higher connectivity can better approximate the quantum state, leading to a more accurate time evolution. Our approach involves generating multiple TTNS with random spanning trees and analyzing the performance of the algorithm under these varying configurations. This allows us to identify bottlenecks and determine the configurations that minimize computational overhead.

In hierarchical representation, significant efforts have been made to identify an optimal spatial tree structure for better numerical efficiency and accuracy. Notably, Hikihara et al. [87] proposed an automatic structural optimization algorithm for TTNs, which refines the tree network by dynamically reconfiguring isometries to minimize bipartite entanglement entropy along each bond. Their methodology involves iteratively sweeping through the tensor network and selecting the optimal local reconnections to reduce truncation errors, much like the DMRG sweeps. By applying this technique to inhomogeneous antiferromagnetic Heisenberg chains, they demonstrated that the optimized TTN naturally adapts to the entanglement structure of the quantum state. Their follow-up work [88] applied the algorithm to the *Rainbow-chain* model, a system where the ground state is approximately represented by singlet pairs spanning various distances. Their study revealed that the optimization process does more than just reduce computational overhead—it also visualizes the intrinsic entanglement geometry of the system. The algorithm successfully reorganized the TTN structure to align with the natural singlet-pair distribution, confirming its capability to extract the optimal spatial layout of the tensor network for a given quantum state.

2.5.1 Random Spanning Tree

To introduce variability in the TTN configuration, we generate spanning trees randomly using *Wilson*'s algorithm. This method ensures that the spanning trees are sampled uniformly from the set of all possible spanning trees for a given lattice.

Wilson's algorithm [89] generates unbiased random spanning trees through *Loop-Erased Random Walks* (LERW). The process involves starting from an arbitrary root node and iteratively connecting unvisited nodes to the tree via random walks. Loops encountered during the walks are erased to maintain the acyclic property of the tree. The following code implements *Wilson*'s algorithm to generate a random spanning tree for a two-dimensional lattice.

Algorithm 2: *Wilson*'s algorithm for Random Spanning Tree

```
Input: $lattice\_nodes$: A set of lattice nodes.
Output: $tree\_dict$: A dictionary representing parent child relations of nodes
Choose a random root node $root \in lattice\_nodes$
Mark $root$ as visited: $visited \leftarrow \{root\}$
Initialize $unvisited \leftarrow lattice\_nodes \setminus \{root\}$
while $unvisited \neq \emptyset$ do
    Select a random starting node $start \in unvisited$
    Initialize $path \leftarrow [start]$
    while $path[-1] \notin visited$ do
        $current \leftarrow path[-1]$
        Determine neighbors of $current$:
        $neighbors \leftarrow \{(current[0] + dx, current[1] + dy) \mid dx, dy \in \{-1, 0, 1\},$
                    $(current[0] + dx, current[1] + dy) \in lattice\_nodes\}$
        Choose a random neighbor $next \in neighbors$
        if $next \in path$ then
            Erase the loop in $path$: $path \leftarrow path[: \mathrm{index}(next) + 1]$
        end
        else
            Append $next$ to $path$: $path \leftarrow path + [next]$
        end
    end
    for $i = 0$ to $|path| - 2$ do
        $parent \leftarrow path[i + 1]$, $child \leftarrow path[i]$
        Update tree_dict
        $tree\_dict$: $tree\_dict[child]["parent"] \leftarrow parent$
        $tree\_dict[parent]["children"] \leftarrow tree\_dict[parent]["children"] + [child]$
        Mark $child$ as visited: $visited \leftarrow visited \cup \{child\}$
        Remove $child$ from $unvisited$: $unvisited \leftarrow unvisited \setminus \{child\}$
    end
end
Return $tree\_dict$
```

2.5.2 Measure TTNS Connectivity

The *Average Path Length* (APL) measures the average distance between all pairs of nodes in the tree. A smaller APL indicates that nodes in the tree are closer to each other on average, suggesting higher overall connectivity. Conversely, a larger APL indicates less connectivity because nodes are, on average, farther apart.

$$\mathrm{APL(TTN)} = \frac{1}{n(n-1)} \sum_{u,v \in V} d(u, v), \tag{2.37}$$

where, n is the number of nodes, V is the set of nodes, and $d(u, v)$ is the distance between nodes u and v.

While APL is a useful metric for assessing overall connectivity, it does not provide insights into how connectivity is distributed among nodes. Specifically, it cannot differentiate between configurations where connectivity is evenly spread across the network and those where it is concentrated in specific regions. Therefore, we introduce *Weighted Path Length Index* (WPLI) to integrate both the APL and its variability.

$$\text{WPLI} = \frac{1}{\text{std\_dev\_length (APL)} \cdot \text{mean\_length (APL)}}, \tag{2.38}$$

where, mean_length (APL) represents the APL's mean value, quantifying the overall connectivity of the network, and std_dev_length (APL) represents the standard deviation of path lengths, capturing the spread or unevenness of connectivity across the network.

WPLI can be computed using the following algorithm:

Algorithm 3: Compute WPLI

```
Input: TTN
Output: WPLI
1  Initialize path_lengths ← empty list
2  foreach node1 ∈ list of all nodes do
3      foreach node2 ∈ list of all nodes do
4          if node1 ≠ node2 then
5              path ← path_from node1 to node2
6              path_lengths.append(len(path) - 1)
7          end
8      end
9  end
10 mean_length ← mean of path_lengths
11 std_dev_length ← standard deviation of path_lengths
12 return $\frac{1}{\text{std\_dev\_length} \cdot \text{mean\_length}}$
```

Smaller mean_length reflects shorter paths, indicative of higher connectivity, and Smaller std_dev_length indicates that path lengths are more evenly distributed, reflecting balanced connectivity across the network. So, WPLI is maximized when both mean_length and std_dev_length are small, representing a configuration with high connectivity and even distribution.

2.5.3 Profiling TTNS Configurations and Bottleneck Analysis

To understand how computational costs scale with different configurations of the underlying tree structure, the following profiling is employed to identify bottlenecks.

The function that updates the environment during sweeps (*update_tree_cache*) has a relatively significant cost among those functions that their number of calls varries across various configurations. However, it turns out that not only the number of calls to this function but also deeper underlying functions, such as *numpy.tensordot* calls or even the total function calls, do not have a strong correlation with the total CPU *runtime.* Nonetheless, the number of *update_tree_cache* calls remains a good indicator of the trade-off between network's complexity and its capacity to support increased bond dimension, as snake-like mappings involve the minimum number of environment constructions, while allow faster bond growth with less effect on CPU *runtime* in comparison with configurations with larger *update_tree_cache_ncalls.* This lead us to find configuration with higher WPLI and lower *update_tree_cache_ncalls*, while tracking *runtime* itself can helps us to differentiate tree structures that have a similar number of *update_tree_cache_ncalls* but with closely comparable WPLI.

To achieve this, we generate a set of random tree structures using *Wilson*'s algorithm and execute a single time-step δt evolution of the second-order 1-site TDVP algorithm with a same random Hamiltonians and initial product states for all configurations. To quantitatively evaluate and prioritizes configurations, we defined a scoring metric with adjustable weighting parameters as follows:

1. Compute the Z-scores for each configuration:

$$Z_i^{\text{runtime}} = \frac{r_i - \bar{r}}{\sigma_r}, \quad Z_i^{\text{n\_calls}} = \frac{v_i - \bar{v}}{\sigma_v}, \quad Z_i^{WPLI} = \frac{w_i - \bar{w}}{\sigma_w} \tag{2.39}$$

   where r_i, v_i, and w_i are the *runtime*, *update_tree_cache_ncalls*, and *WPLI* values for configuration i, $\bar{r}$, $\bar{v}$, and $\bar{w}$ are the means, and σ_r, σ_v, and σ_w are their respective standard deviations.

2. Invert the Z-score of *runtime* and *n_calls* (short for *update_tree_cache_ncalls*) since lower values are preferable.

$$Z_i^{\text{runtime}} \to -Z_i^{\text{runtime}}, \quad Z_i^{\text{n\_calls}} \to -Z_i^{\text{n\_calls}}, \tag{2.40}$$

3. Compute the composite score S_i for each configuration i using adjustable weights α, β γ.

$$S_i = \alpha Z_i^{\text{runtime}} + \beta Z_i^{\text{n\_calls}} + \gamma Z_i^{WPLI}, \tag{2.41}$$

4. Normalize the scores S_i across the subset to a range of [0, 1].

$$S_i^{\text{norm}} = \frac{S_i - S_{\text{min}}}{S_{\text{max}} - S_{\text{min}}}, \tag{2.42}$$

   where S_{min} and S_{max} are the minimum and maximum scores in the subset.

2.5.4 Profiling on Random 6×6 Lattice

To demonstrate the methodology, we conduct profiling on 200 spanning tree configurations of a 6×6 lattice, generated using *Wilson*'s algorithm. Both the Hamiltonians and states were initialized with random entries, and the states were prepared with a uniform virtual bond dimension of 6. A single time-step evolution was performed for each trial using the second-order 1-site TDVP algorithm. In the following, we present the results for 4 different weighted scoring metrics, and visualize the highest and lowest scored spanning trees.

To facilitate an intuitive comparison, configurations are visualized on parallel coordinate plots, with each axis representing a normalized metric. A heatmap is applied to color-code configurations to differentiate configurations based on normalized *runtime.*

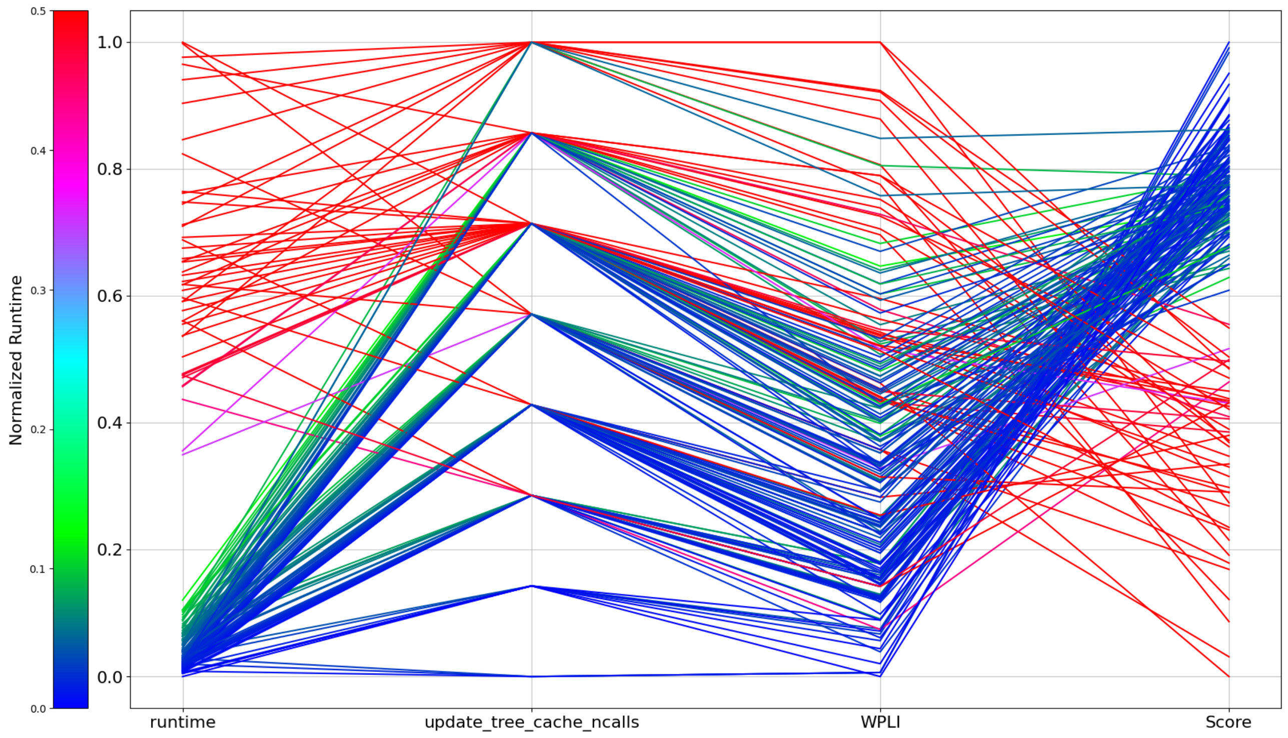
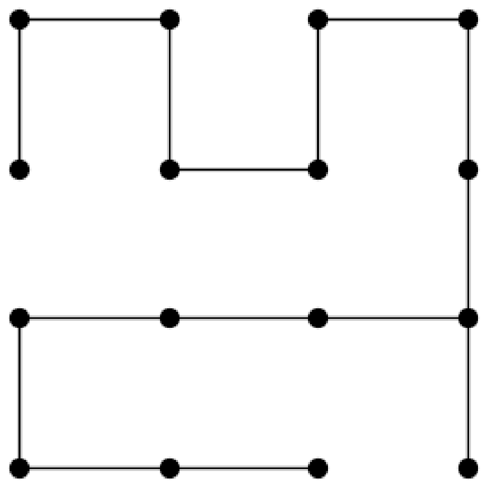
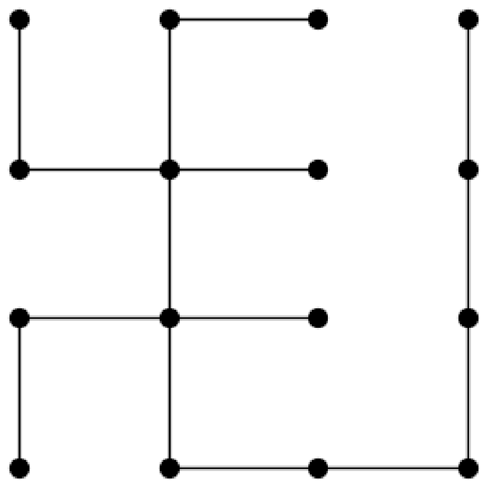


Figure 2.4: profiling results for $\alpha = 1, \beta = 1, \gamma = 1$.



Score = 0.91

runtime: 0.75
update_tree_cache_ncalls: 32
WPLI: 5.77



Score = 0

runtime: 19
update_tree_cache_ncalls: 42
WPLI: 12.4

Remark. Highest score in this case corresponds to snake-like mapping, which we should exclude.

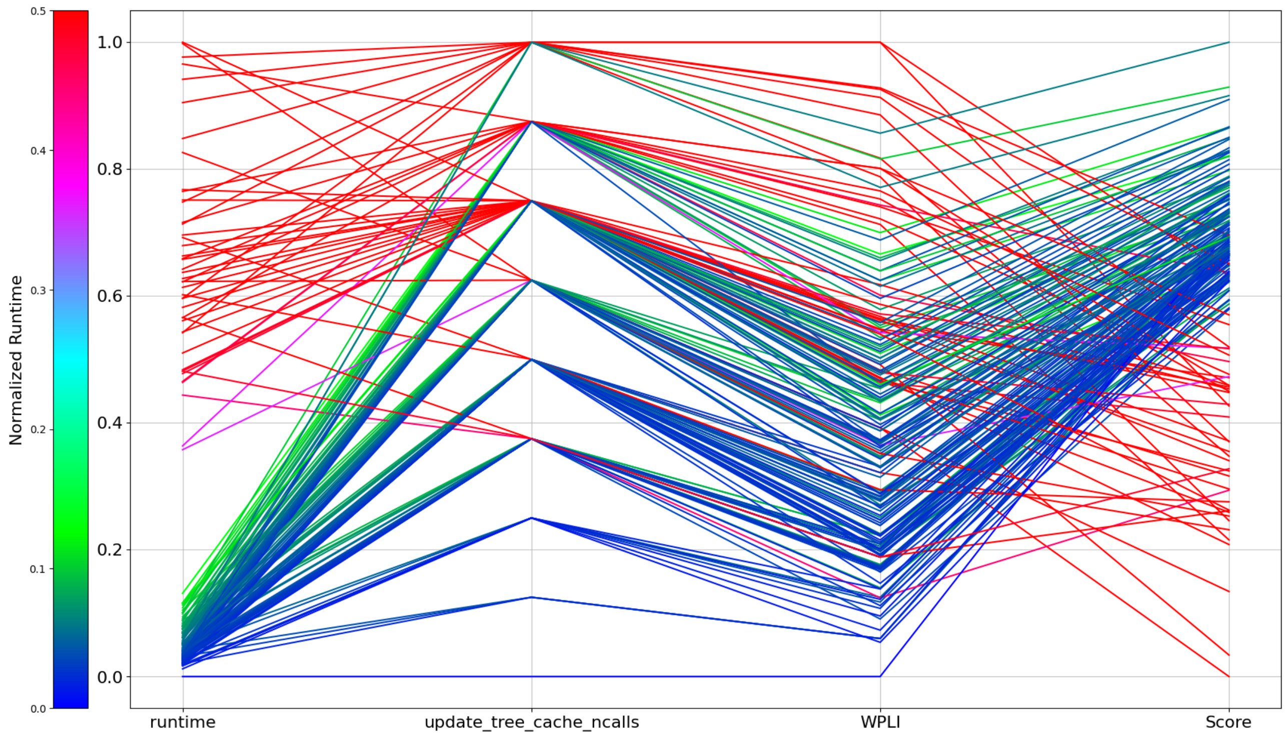
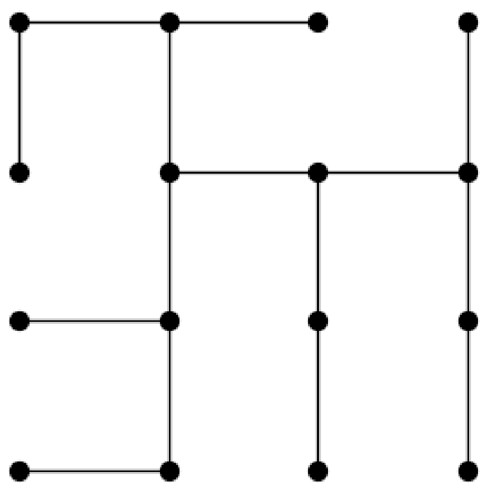
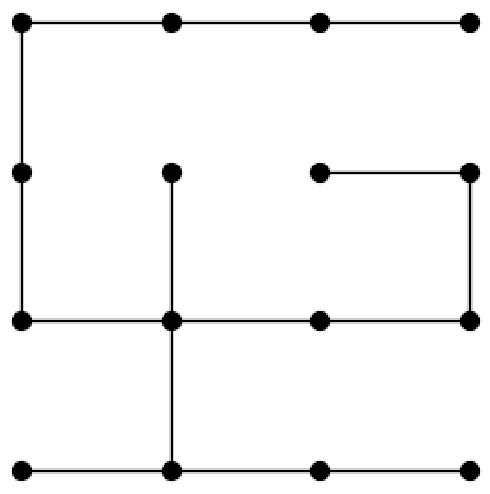


Figure 2.6: profiling results for $\alpha = 2, \beta = 1, \gamma = 2$.





Score = 1

runtime: 1.3
update_tree_cache_ncalls: 46
WPLI: 18

Score = 0

runtime: 18.9
update_tree_cache_ncalls: 40
WPLI: 10.9

2.5.5 Benchmark : Bose-Hubbard on 4x4 Lattice

In this experiment we aim to examine the performance of the second-order 1TDVP for different spanning tree configurations.

Hamiltonian. We consider the two-dimensional (2D) *Bose–Hubbard* Hamiltonian on an $L \times L$ square lattice with periodic boundary conditions along both directions.

$$\hat{H} = -J \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \Big(\hat{b}^{\dagger}_{(i,j)} \hat{b}_{(i+1 \mod L, j)} + \text{H.c.} \Big) \tag{2.43}$$

$$-J \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \Big(\hat{b}^{\dagger}_{(i,j)} \hat{b}_{(i, j+1 \mod L)} + \text{H.c.} \Big) \tag{2.44}$$

$$+\frac{U}{2} \sum_{i=0}^{L-1} \sum_{j=0}^{L-1} \hat{n}_{(i,j)} \Big(\hat{n}_{(i,j)} - 1 \Big). \tag{2.45}$$

Where $\hat{b}^{\dagger}_{(i,j)}$ and $\hat{b}_{(i,j)}$ are the bosonic creation and annihilation operators at site (i, j), and the parameters J and U respectively determine the hopping amplitude and on-site interaction strength. Here, the sums over i and j run from 0 to $L - 1$, as the site labels start from $(i, j) = (0, 0)$:

For the exact error analysis, the exact solution could be achieved using the *QuTiP* library. However, the largest (2D) *Bose–Hubbard* Hamiltonian that *QuTiP* can handle is maximum 16 particles on 4×4 lattics. Therefore, we set lattice size $L = 4$ with maximum 1 particle allowed per site, which makes the third terms in the Hamiltonian vanish.

Parameters and Simulation Time. We use natural units with $\hbar = 1$, taking J as the reference energy scale and setting $J = 2$. The interaction strength is $U = 0.1\,J = 0.2$. This places us in the $J \gg U$ regime, where kinetic energy dominates and particles exhibit significant mobility. The evolution final time is set to $t = 2$, corresponding to $2/J = 1$ in physical time units. All simulations are performed using a time-step of $\delta t = 0.01$.

Initial States. For our numerical experiment, we first run the profiling introduced in Section 2.5.3 for 200 iterations. Then we initialize each tree in a checkerboard configuration of zero and one boson per site, which provides a non-trivial starting point particularly suitable for testing our algorithm capability to study far-from-equilibrium bosonic dynamics in lattice systems.

$$|\psi_0\rangle = \bigotimes_{i=0}^{L-1} \bigotimes_{j=0}^{L-1} \begin{cases} |1\rangle_{(i,j)}\,, & \text{if } (i+j) \bmod 2 = 0, \\ |0\rangle_{(i,j)}\,, & \text{if } (i+j) \bmod 2 = 1. \end{cases} \tag{2.46}$$

Then we pad all virtual bonds with zeros up to different values for diffenet configurations to achieve comparable running times (between 20 to 25 minumtes) as trees with higher connecttivity could handle smaller bond dimensions. All 200 configurations falls into $\{30, 32, 34, 36, 38, 40, 42, 44, 46, 46, 50\}$ *update_tree_cache_ncalls* values.

Observable. To investigate correlations, we monitor the sum over all density–density correlator at distances = 3,

$$\hat{O}^{(3)}_{nn} = \sum_{\substack{(i,j),\,(i',j') \\ \text{dist}((i,j),(i',j'))=3}} \hat{n}_{(i,j)}\, \hat{n}_{(i',j')}, \tag{2.47}$$

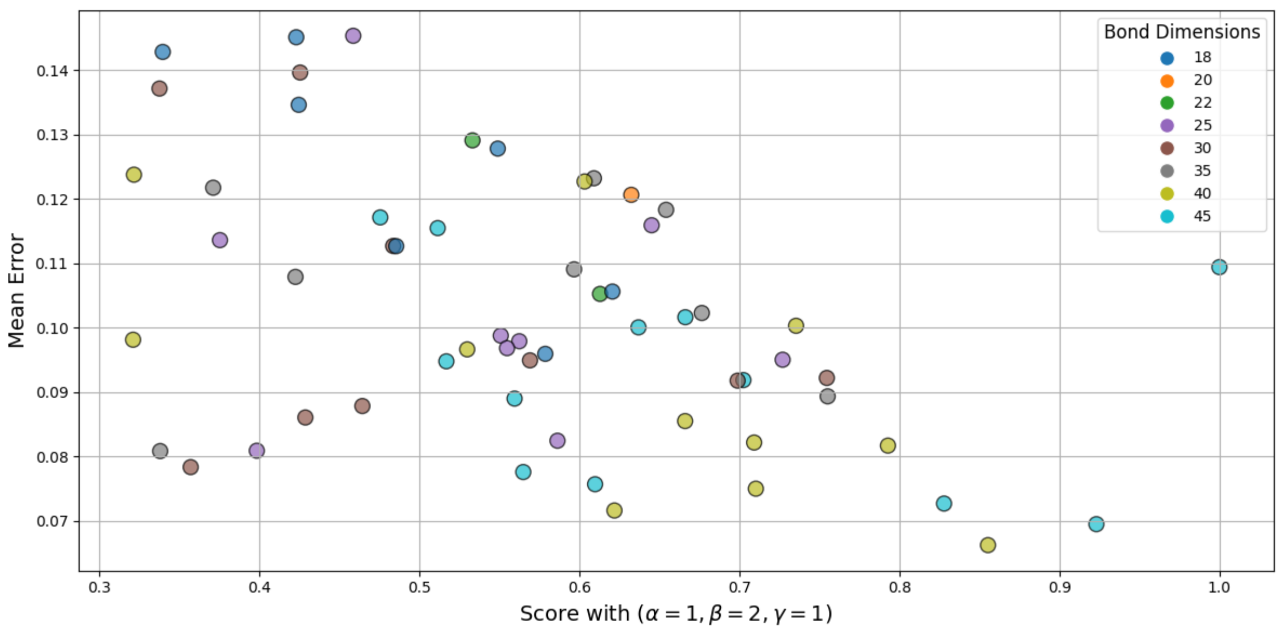
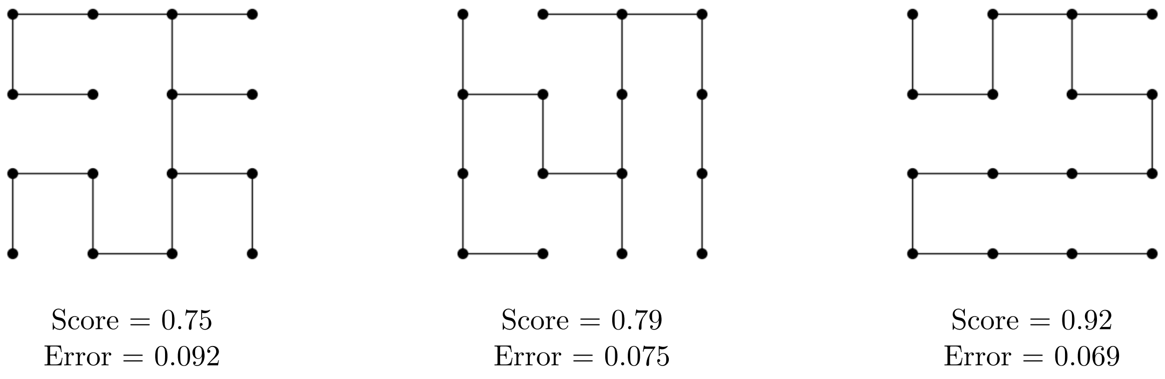


Figure 2.8: Mean error of different configurations against their scores that were computed beforehand



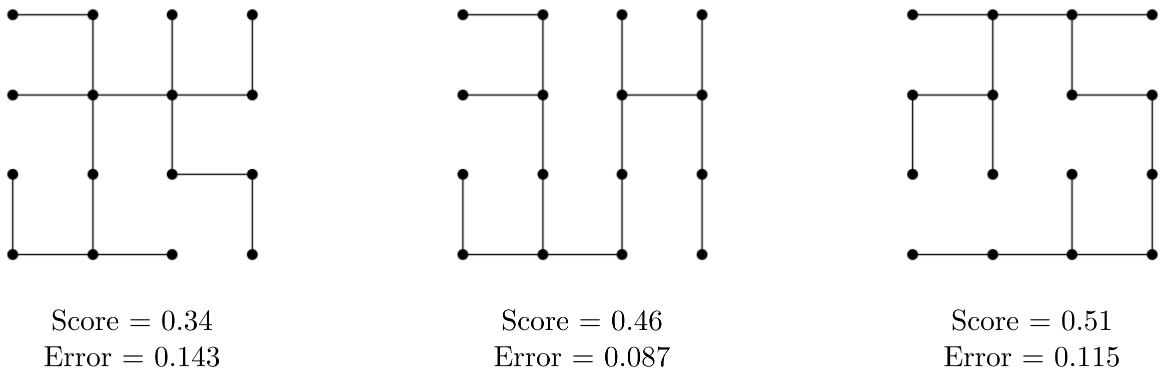


Figure 2.9: Example of different TTNS with corresponding scores and errors

Figure 2.8 illustrates the mean absolute error of the observable $\widehat{O}_{nn}^{(3)}$ evaluated across all $200 = \frac{2}{0.01}$ time-steps.

$$\text{mean error} = \frac{1}{200} \sum_{k=1}^{200} \left| \widehat{O}_{\text{nn,sim}}^{(3)}(k \cdot 0.01) - \widehat{O}_{\text{nn,exact}}^{(3)}(k \cdot 0.01) \right| \tag{2.48}$$

providing a precise measure of the average error over the entire simulation duration.

The outcome confirms the effectiveness of the scoring metric in identifying configurations that lead to lower errors. TTNS with scores higher than 0.7 exhibit errors bellow 10^{-1}. The Score = 1 configuration corresponds to the MPS snake-like mapping, has the minimum *update_tree_cache_ncalls*. Moreover, for this model, we found the optimal weight for the scoring metric to be $\alpha = 1, \beta = 2, \gamma = 1$.

Figure 2.9, shows 6 examples that were used in the benchmark, with thier corresponding scores and errors.

3. Global Subspace Expansion

The introduction of the density matrix renormalization group [90, 91] and *Matrix Product States* algorithms to simulate ground states of one-dimensional quantum many-body systems [92, 93] have given strong evidence for the fact that physically interesting states are confined to reside in a small submanifold of the full *Hilbert* space which could be achieved via truncation of virtual bonds. This observation was later proven in the context of quantum information theory in terms of entanglement properties of ground states. More specifically, it was shown that ground states of one-dimensional systems whose Hamiltonian is gapped are only weakly entangled (they obey the area law) and can as such be faithfully and efficiently simulated in terms of matrix product states [94].

The virtual bond dimensions of a TN determines the number of variational parameters, associated storage and computational costs and the capacity of the network to represent quantum states. A TNS ($|\Psi\rangle \in V^{\otimes L}$) with higher bond dimensions captures more complex entanglement structures and provide more accurate representation within the allowed symmetry sectors. However, a TTN with restricted bond dimension χ provides an efficient representation of specific many-body wave functions (subset $S_\chi \subset V^{\otimes L}$ of $|\Psi\rangle$) which requires a number of coefficients that grows polynomially (often linearly) with system size L, rather than exponentially.

3.1 Local Subspace Expansion

The idea of subspace expansion was originally introduced by Hubig et al. (2015) [95] as a strategy in 1DMRG, designed to enrich the variational space by appending the dominant singular vectors of the residual error $|Z\rangle = H|\psi\rangle - E|\psi\rangle$ to the MPS tensor basis. This allows the algorithm to escape local minima and enhance convergence.
In their algorithm the expansion term P_i is constructed by multiplying mixing factor α, that controls the magnitude of the subspace expansion, with residual tensor at site i that is approximated by neglecting the right environment R_{i+1}.

$$P_i = \alpha L_{i-1} \cdot M_i \cdot W_i, \tag{3.1}$$

where, L_{i-1} is the left contraction environment up to site $i-1$, M_i is the target MPS tensor and W_i is the local MPO (Hamiltonian) tensor.

Given the initial dimensions of the tensors:

- $M_i \in \mathbb{C}^{d \times m_{i-1} \times m_i}$,
  where: d: Physical dimension, m_{i-1}: Left bond dimension, m_i: Right bond dimension.
- $P_i \in \mathbb{C}^{d \times m_{i-1} \times m_{P_i}}$,
  where $m_{P_i} = w \cdot m_i$, representing the additional bond space introduced by the subspace expansion.

- Neighboring tensor $B_{i+1} \in \mathbb{C}^{d\times m_i \times m_{i+1}}$.

The expansion term P_i is added to M_i and zero is padded to neighbor site B_{i+1} for compatibility, resulting in an expanded tensor:

$$\tilde{M}_i = \begin{bmatrix} M_i & P_i \end{bmatrix} \in \mathbb{C}^{d\times m_{i-1}\times(m_i+m_{P_i})}, \tag{3.2}$$

$$\tilde{B}_{i+1} = \begin{bmatrix} B_{i+1} \\ 0 \end{bmatrix} \in \mathbb{C}^{d\times(m_i+m_{P_i})\times m_{i+1}}, \tag{3.3}$$

Next, SVD is performed on $\tilde{M}_i$ to retain the most significant singular vectors:

$$\tilde{M}_i = U\Sigma V^\dagger, \tag{3.4}$$

where: $U \in \mathbb{C}^{d\times m_{i-1}\times m_{\max}}$, $\Sigma \in \mathbb{C}^{m_{\max}\times m_{\max}}$, $V^\dagger \in \mathbb{C}^{m_{\max}\times(m_i+m_{P_i})}$
The truncation process retains the top $m_{\max}$ singular values:

$$M_i' = U[:,:,:m_{\max}] \in \mathbb{C}^{d\times m_{i-1}\times m_{\max}}, \tag{3.5}$$

$$B_{i+1}' = (\Sigma V^\dagger)\cdot \tilde{B}_{i+1} \in \mathbb{C}^{d\times m_{\max}\times m_{i+1}}. \tag{3.6}$$

In this process the bond dimension of M_i at m_i expands to $m_{\max}$, while the physical properties are preserved.

3.2 Global Subspace Expansion (GSE)

Motivation : TDVP1 vs TDVP2 error analysis

In 1-site TDVP, the bond dimensions of the TNS is constant throughout the evolution, whereas, in 2-site TDVP, in each 2-site local update (2.33), the bond dimensions, due to contraction before update and SVD decomposition after update of ψ_l^{2s}, is dynamically adjusted with parameters introduce in 1.4, which leads to lower projection error at the cost of introducing of truncation error. The projection error arises from the fact that the true time evolution vector $\hat{H}|\Psi^{\mathcal{M}[T,D]}(t)\rangle$ differs from the projected one $\hat{\mathcal{P}}\hat{H}|\Psi^{\mathcal{M}[T,D]}(t)\rangle$. The evaluation of projection error have greater computational cost than a 1-site TDVP sweep.

The error analysis of two exactly solvable models, one evolved with 1-site TDVP with varying bond dimensions, and the other evolved with 2-site TDVP with varying truncation parameters was carried out in [96]. In 2-site TDVP with suitable truncation parameters, bond dimensions stay at lower dimensions early on and expands gradually when necessary. 1-site TDVP, on the other hand, requires bond dimensions to be set higher than the initial bond dimensions in 2-site TDVP case, otherwise we would face significant projection error. However, the initial overestimated bond dimension leads to unnecessary computation at the early stages of the evolution, as a result the runtime scales badly with the system size. Notably, the outperformance of 1-site TDVP compared to 2-site TDVP was observed When setting the bond dimensions, by padding with zero, up to the optimal value, which could be achieved by calculating the error at final the time of 2-site TDVP simulation (assuming the exact solution is known). This is mainly because 1-site updates require only a local QR decomposition on a single site, whereas 2-site TDVP involves a larger 2-site tensor and requires an SVD, which is generally more expensive. However, in a real case scenario, we don't have the exact solution to calculate the error at the final time, and the optimal bond dimension is not known beforehand. Therefore, bond dimension are inevitably either over or underestimated.

Motivated by the mentioned drawback of 1-site TDVP and higher computational cost of 2-site TDVP, we are intrigued to investigate strategies to dynamically adjust the bond dimensions during the 1-site TDVP evolution, to achieve the same accuracy of 2-site TDVP while avoiding its higher computational cost. It is worth mentioning that simply padding the bond with zeros is not an effective strategy to reduce the projection error [97].

3.3 GSE-TDVP Algorithm

We learned how MPS manifold could be enlarged directly by appending a local subspace to the MPS tensor at each site. Inspired by the subspace expansion algorithm in DMRG, Yang and White [2] have shown how one can expand the MPS manifold with another manifold globally, thereby enlarging the tangent space before a TDVP time-step, to make it directed to the true equation of motion, with the intermediate numerical costs between 1TDVP and 2TDVP. In this method, *Krylov* vectors serve as ancillary MPSs to enrich the basis of the time-evolving MPS through the gauge degree of freedom, thus avoiding the problems of loss of orthogonality and production of unnecessarily highly entangled states. In the following, we will describe the GSE-TDVP algorithm adapted to TTN in detail.

Algorithm 4: GSE-1TDVP Algorithm

```
Input  : $|\psi(t=0)\rangle$, $H$, time_step = $\delta t$, final_time = $T$, expansion_step
Output: $|\psi(T)\rangle$
1 $N_{\text{steps}} \leftarrow T/\delta t$
2 for $i = 0$ to $N_{steps} - 1$ do
3     $|\psi((i+1)\delta t)\rangle \leftarrow$ Run one time_step 1TDVP on $|\psi(i\delta t)\rangle$
4     if $i$ = expansion_step then
5         $|\psi((i+1)\delta t)\rangle \leftarrow$ Run GSE on $|\psi((i+1)\delta t)\rangle$
6     end
7 end
8 return $|\psi(T)\rangle$
```

Definition 3.3.1. The concept of *Krylov subspaces* arises from the iterative methods that generate a sequence of vectors from a given initial vector and a matrix, allowing for the approximation of solutions to linear systems, defined as :

$$\mathcal{K}_m(H, |\psi_0\rangle) = \text{span}\left\{|\psi_0\rangle, H|\psi_0\rangle, H^2|\psi_0\rangle, \ldots, H^{m-1}|\psi_0\rangle\right\} \tag{3.7}$$

In GSE, instead of directly constructing the *Krylov* subspace, we could achieve good accuracy with modest computational resources by an approximate expansion that is constructed by successive application of first-order expansion of the time evolution operator $(\mathbb{I} - i\tau H)$ to capture low-order corrections by slightly perturbs the state, leading to controlled entanglement growth. In this process, the Hamiltonian TTNO is multiplied element-wise by $-i\tau$, where τ represents a small time-step, and subsequently summed with the identity TTNO as we discussed in 1.3.2. For numerical stability, after each application of $(\mathbb{I} - i\tau H)$, all bonds are truncated with adjustable `svd_truncation_parameters` by canonical form transformation using SVD as discussed in Section 1.4.4, and the resulting TTN is normalized. The initial TTN $|\psi_0\rangle$ is enlarged with the next *Krylov* basis element and the resulting would be enlarged with the next *Krylov* basis element, and the iteration continues until all *Krylov* basis are incorporated.

The general structure of GSE algorithm involves these main steps:

Algorithm 5: Global Subspace Expansion (GSE)

```
   Input: $|\psi_0(t)\rangle$, $H$, m, $\tau$, svd_truncation_parameter
   Output: $|\psi^{expanded}\rangle$
1  Construct m Krylov_basis ($|\psi_0(t)\rangle$ , $H$, $\tau$, svd_truncation_parameter):
2  Krylov_basis ← $\{|\psi_0(t)\rangle, (1 - i\tau H)|\psi_0(t)\rangle, (1 - i\tau H)^2|\psi_0(t)\rangle, \ldots, (1 - i\tau H)^{k-1}|\psi_0(t)\rangle\}$
3  Initialize: $|\psi^{expanded}\rangle$ ← Krylov_basis[0]
4  for $i = 0$ to m − 1 do
5      $|\psi^{expanded}\rangle$ ← Expand $|\psi^{expanded}\rangle$ by Krylov_basis[i]
6      $|\psi^{expanded}\rangle$ ← Normalize $|\psi^{expanded}\rangle$
7  end
8  return $|\psi^{expanded}\rangle$
```

Remark. The normalized TTN (line 6) is simply achieved by dividing the TTN local tensor at *orthogonalization center* by its norm.

To illustrate steps, we will focus on first iteration (i = 0) in line 4, were $|\psi^{expanded}\rangle$ is constructed by enlarging $|\psi_1\rangle = |\psi_0\rangle$ subspace with $|\psi_2\rangle = (1 - i\tau H)|\psi_0(t)\rangle$.

The algorithm sweeps through the TTN, expanding one virtual bond at a time. The optimal *update path* follows the principles discussed in Section 2.4.1. A crucial challenge in this procedure is in the calculation of the reduced density matrix for each local update, as the direction of partitioning must be determined appropriately, at branching sites. We will refer to this reduced density matrix as the *Partial Reduced Density Matrix* (PRDM).

More precisely, as the algorithm proceed along the *update path*, it requires the following conditions to be satisfied:

- The *target node* must be at the *orthogonalization center.*
- The *next node* to be updated must be chosen such that the branch node can be partitioned in a way that the unvisited sites could be traced out in a way that give the correct PRDM.

This can be effectively managed by employing the *orthogonalization path* as described in Section 2.4.2.

Algorithm 6: Sweeping Through the TTN and Expanding Virtual Bonds

```
   Require: update path, orthogonalization path
1  Initialize $\psi_1$ and $\psi_2$ to be site canonical at update path[0]
2  for i, target node ∈ enumerate(update path[:-1]) do
3      target node ← update path[i]
4      next node ← orthogonalization path[i][0]
5      Construct the $\psi^{expanded}$ tensor at target node
6      Update $\psi_1$ and $\psi_2$ tensors at next node
7      if orthogonalization path[i] > 1 then
8          $\psi_1$ ← Move orthogonalization center of $\psi_1$ to orthogonalization path[i][−1]
9          $\psi_2$ ← Move orthogonalization center of $\psi_2$ to orthogonalization path[i][−1]
10     end
11 end
12 Construct the $\psi^{expanded}$ tensor at target node[−1]
```

Line 5-10 of Algorithm 6

Notation remark: The bonds are labeled by their corresponding node and the node that they are connected to. Thus, the bond b_{AB} and b_{BA} have the same dimension, and should be contracted with each other. And *target node* and *next node* are denoted by T and N, respectively.

Step 1. Compute PRDM at Site *target node* toward *next node*

$$\rho^{\mathrm{N}}_{1,T} = \sum_{b_{TN}} C_T \left(C_T\right)^{\dagger}, \tag{3.8}$$

$$\rho^{\mathrm{N}}_{2,T} = \sum_{b_{TN}} \tilde{C}_T \left(\tilde{C}_T\right)^{\dagger}. \tag{3.9}$$

Combine the reduced density matrices:

$$\rho^{\mathrm{N}}_{T} = \rho^{\mathrm{N}}_{1,T} + \rho^{\mathrm{N}}_{2,T} \in \mathbb{C}^{d^n \times d^n}, \tag{3.10}$$

Step 2. Diagonalize PRDM

Diagonalize the reduced density matrix:

$$\rho^{\mathrm{N}}_{T} = U\Sigma^2 U^{\dagger}, \quad U \in \mathbb{C}^{d^n \times d^n}, \quad \Sigma^2 \in \mathbb{C}^{d^n \times d^n} \tag{3.11}$$

where d is the physical dimension and $n \in \mathbb{N}$.

Step 3. Construct the Expanded Tensor of $|\psi^{expanded}\rangle$ at *target node*

The tensor of $|\psi^{\text{expanded}}\rangle$ at site *target node* $= T$ is updated with $U^{\dagger}$ which is orthogonalized toward the *next node* .

Step 4. Update *next node* tensors of ψ_1 and ψ_2 with U

Step 4.1. move the *orthogonalization center* to the *next node*
Perform a QR decomposition on the C_T, and contract R to the *next node* tensor.

$$C_T = Q_T, C_N = R_T \cdot C_N \tag{3.12}$$

similarly for $\tilde{C}_T$:

$$\tilde{C}_T = Q_T, \tilde{C}_N = R_T \cdot \tilde{C}_N \tag{3.13}$$

construct $(UC)_T$ and $(U\tilde{C})_T$ matrix by contracting C_T and $\tilde{C}_T$ with U

$$(UC)_T = \text{Contract}\left(C_T \cdot U\right) \in \mathbb{C}^{d_{b_{TN}} \times d^n} \tag{3.14}$$

$$(U\tilde{C})_T = \text{Contract}\left(\tilde{C}_T \cdot U\right) \in \mathbb{C}^{\tilde{d}_{b_{TN}} \times d^n} \tag{3.15}$$

In this contraction, all legs of C_T and $\tilde{C}_T$ except the leg toward the *next node* i.e. b_{TN} are contracted with the corresponding legs of U except the leg toward the Singular value matrix Σ.

Step 4.2. Multiply $(UC)_T$ and $(U\tilde{C})_T$ matrices with *next node* tensor T_N and $\tilde{T}_N$

$$C_N = \text{Contract}\left(T_N \cdot (UC)_T\right), \tag{3.16}$$

$$\tilde{C}_N = \text{Contract}\left(\tilde{T}_N \cdot (U\tilde{C})_T\right) \tag{3.17}$$

These two tensors will be used to bulid the PRDM in the next iteration. They belong to $\mathbb{C}^{\prod_{i=1} d_{b_i} \times d^m \times d}$ where the product is over all neighbors except the *target node* and m is the number of PRDM that has been allready contracted in the subtree toward that specific leg.

Step 5. Move *orthogonalization center* of ψ_1 and ψ_1 at branch nodes

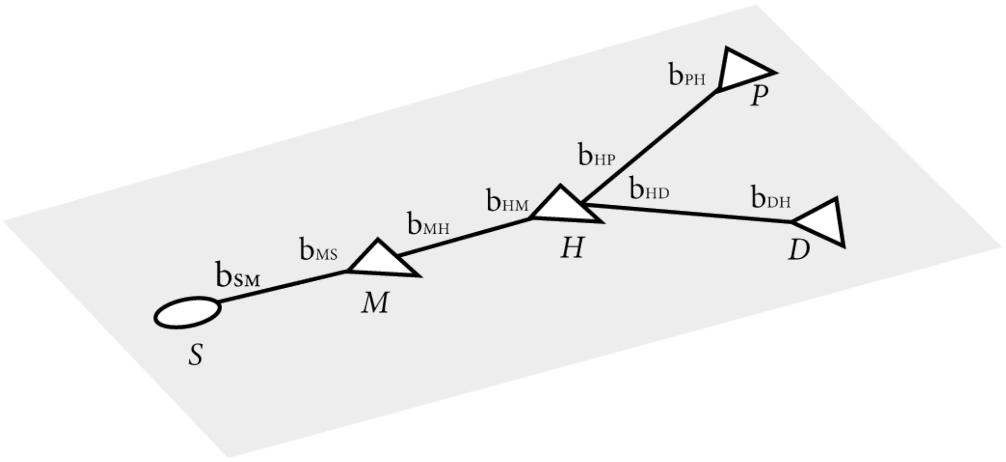
By moving the *orthogonalization center* of ψ_1 and ψ_2 to the new site, we make sure that the next iteration will start with the *orthogonalization center* at the *target node.*

Line 12 of Algorithm 6

Final step. Construct the last tensor of $|\psi^{expanded}\rangle$

Given *update path[-1] = L* , C_L and $\tilde{C}_L$ were updated in the previous iteration, where L was the *next node.* The last tensor of $|\psi^{expanded}\rangle$ is updated with the C_L and $\tilde{C}_L$ is discarded.

Example. In the following illustrative example, we will go through the steps of algorithm 6 for a simple TTN with 5 nodes.



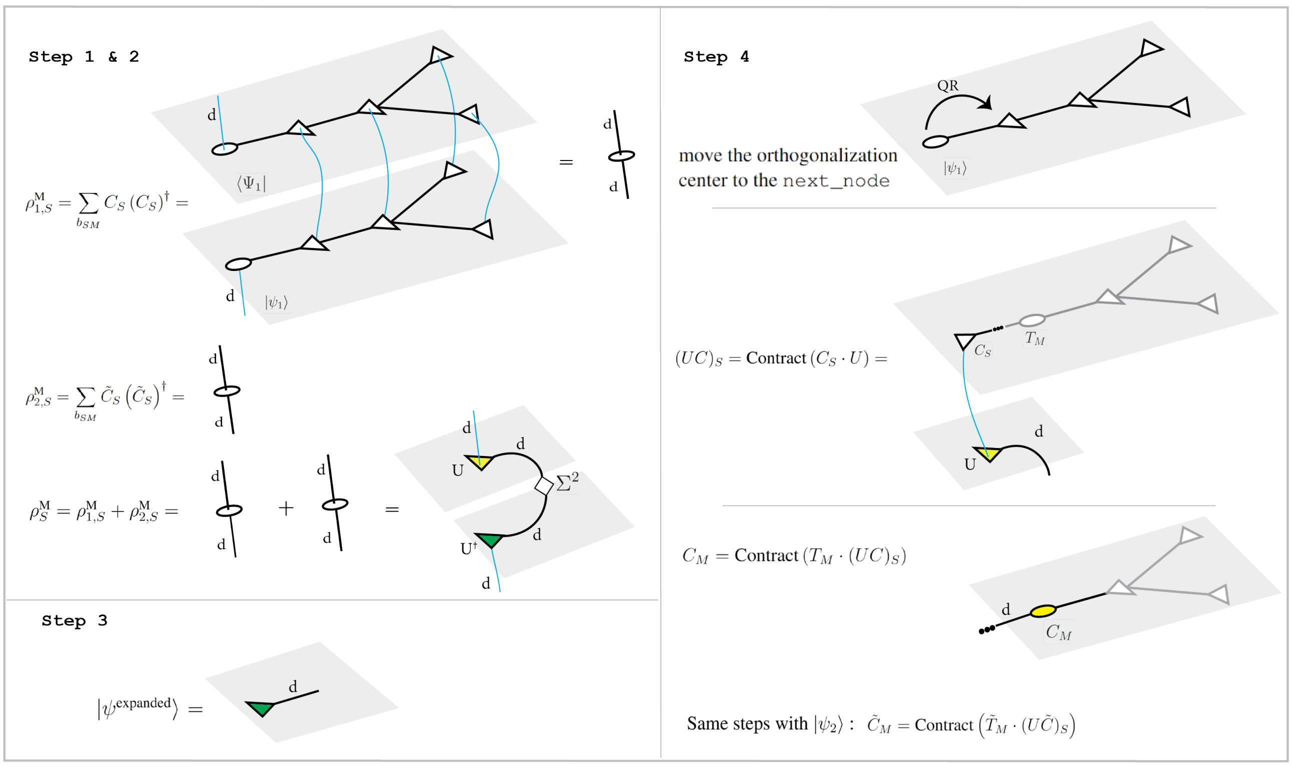
Both TTNSs are in site-canonical form with *orthogonalization center* at leaf node *update path*$[0] = S$ with tensors $C_S \in \mathbb{C}^{d_{b_{SM}} \times d}$ and $\tilde{C}_S \in \mathbb{C}^{\tilde{d}_{b_{SM}} \times d}$ respectively.

$$|\psi_1\rangle \;= \sum_{\{b\},\{\sigma\}} C_S^{\sigma_S}{}_{b_{SM}} T_M^{\sigma_M}{}_{b_{MS},b_{MH}} T_H^{\sigma_H}{}_{b_{HM},b_{HD},b_{HP}} T_D^{\sigma_D}{}_{b_{DH}} T_P^{\sigma_P}{}_{b_{PH}} \, |\sigma_S\sigma_M\sigma_H\sigma_D\sigma_P\rangle$$

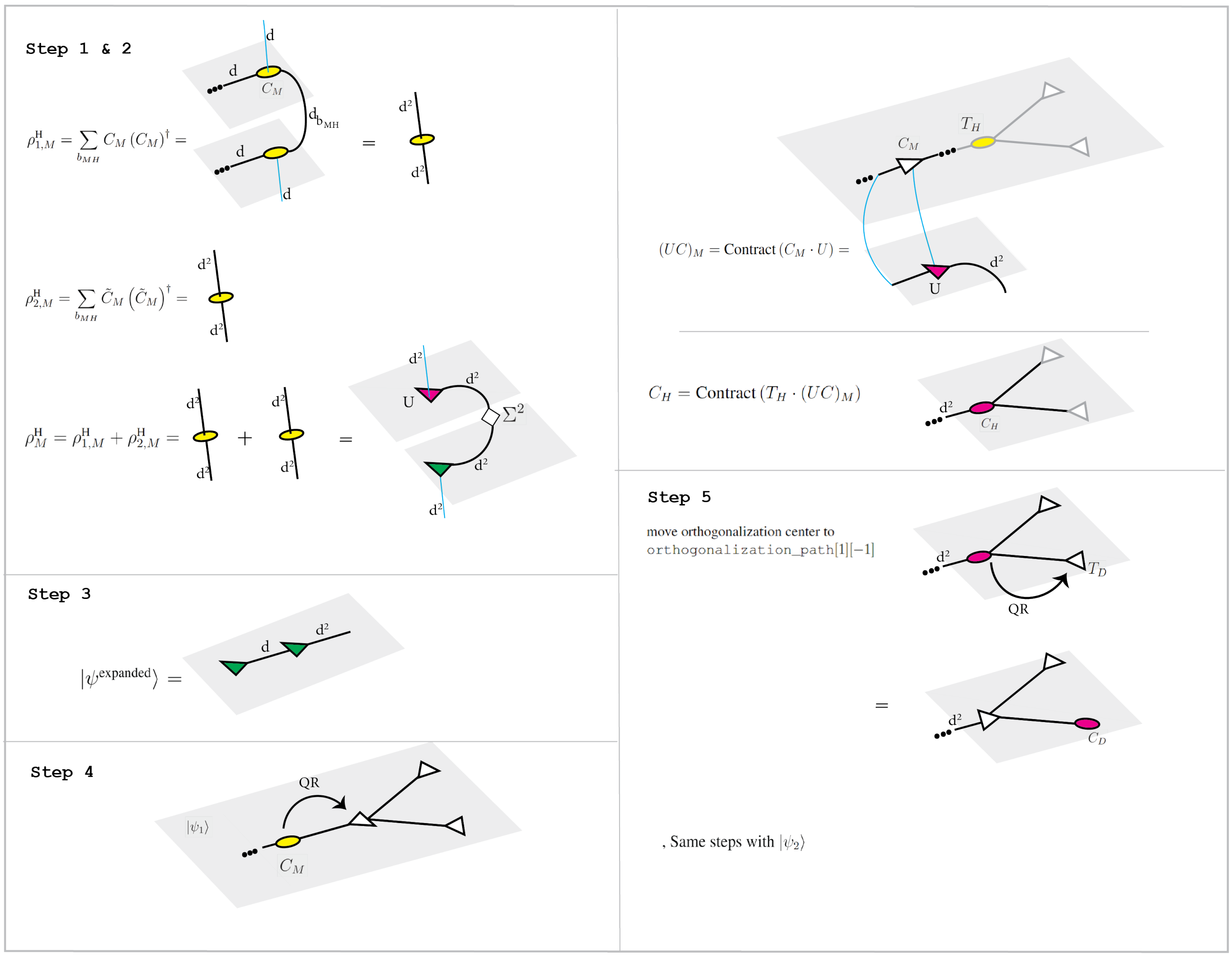
$$|\psi_2\rangle = \sum_{\{b\},\{\sigma\}} \tilde{C}_S^{\sigma_S}{}_{b_{SM}} \, \tilde{T}_M^{\sigma_M}{}_{b_{MS},b_{MH}} \, \tilde{T}_H^{\sigma_H}{}_{b_{HM},b_{HD},b_{HP}} \, \tilde{T}_D^{\sigma_D}{}_{b_{DH}} \, \tilde{T}_P^{\sigma_P}{}_{b_{PH}} \, |\sigma_S\sigma_M\sigma_H\sigma_D\sigma_P\rangle$$

update path $= [S, M, D, H, P]$
orthogonalization path $= [[M], [H, D], [H], [P]]$

$$\mathbf{i = 0 :}\ \textit{target node} = S\ ,\ \textit{next node}\ = M$$


Step 1 & 2
$\rho^{M}_{1,S} = \sum_{b_{SM}} C_S\,(C_S)^\dagger =$
$\langle\Psi_1|$
$|\psi_1\rangle$
d
=
$\rho^{M}_{2,S} = \sum_{b_{SM}} \tilde{C}_S\,(\tilde{C}_S)^\dagger =$
$\rho^{M}_{S} = \rho^{M}_{1,S} + \rho^{M}_{2,S} =$
+
U
$U^\dagger$
Σ^2
Step 3
$|\psi^{\text{expanded}}\rangle =$
Step 4
QR
move the orthogonalization center to the next_node
$(UC)_S = \text{Contract}\,(C_S \cdot U) =$
C_S
T_M
$C_M = \text{Contract}\,(T_M \cdot (UC)_S)$
C_M
Same steps with $|\psi_2\rangle$: $\tilde{C}_M = \text{Contract}\left(\tilde{T}_M \cdot (U\tilde{C})_S\right)$

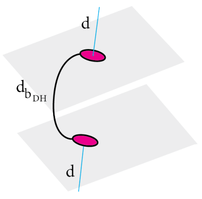
$$\mathbf{i = 1 :}\ \textit{target node} = M\ ,\ \textit{next node}\ = H$$


Step 1 & 2
$\rho^{H}_{1,M} = \sum_{b_{MH}} C_M\,(C_M)^\dagger =$
C_M
$d_{b_{MH}}$
d
d^2
=
$\rho^{H}_{2,M} = \sum_{b_{MH}} \tilde{C}_M\,(\tilde{C}_M)^\dagger =$
$\rho^{H}_{M} = \rho^{H}_{1,M} + \rho^{H}_{2,M} =$
+
U
Σ^2
Step 3
$|\psi^{\text{expanded}}\rangle =$
Step 4
QR
$|\psi_1\rangle$
T_H
$(UC)_M = \text{Contract}\,(C_M \cdot U) =$
$C_H = \text{Contract}\,(T_H \cdot (UC)_M)$
C_H
Step 5
move orthogonalization center to orthogonalization_path[1][−1]
T_D
C_D
, Same steps with $|\psi_2\rangle$

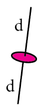
$\mathbf{i} = \mathbf{2}:$ *target node* $= D$, *next node* $= H$

Step 1 & 2

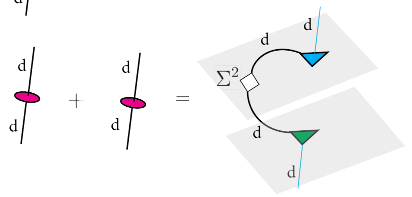
$\rho^{\mathrm{H}}_{1,D} = \sum_{b_{DH}} C_D \, (C_D)^\dagger =$



$\rho^{\mathrm{H}}_{2,D} = \sum_{b_{DH}} \tilde{C}_D \left(\tilde{C}_D\right)^\dagger =$

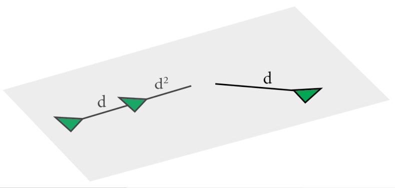


$\rho^{\mathrm{H}}_{D} = \rho^{\mathrm{H}}_{1,D} + \rho^{\mathrm{H}}_{2,D} =$

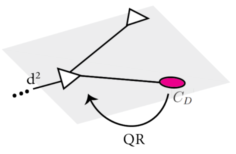


Step 3

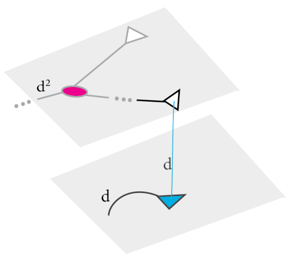
$|\psi^{\text{expanded}}\rangle =$



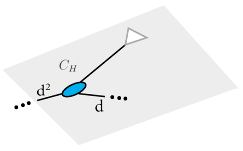
Step 4



$(UC)_D = \text{Contract}\,(C_D \cdot U) =$



$C_H = \text{Contract}\,(C_H \cdot (UC)_D)$

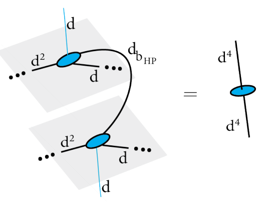


Same steps with $|\psi_2\rangle$

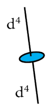
$\mathbf{i} = \mathbf{3}$ **and final step** $:$ *target node* $= H$, *next node* $= P$

Step 1 & 2

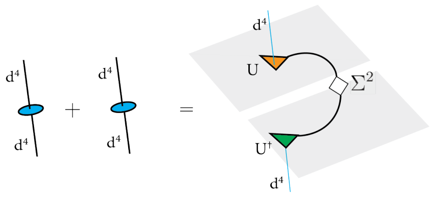
$\rho^{\mathrm{P}}_{1,H} = \sum_{b_{HP}} C_H \, (C_H)^\dagger =$



$\rho^{\mathrm{P}}_{2,H} = \sum_{b_{HP}} \tilde{C}_H \left(\tilde{C}_H\right)^\dagger =$

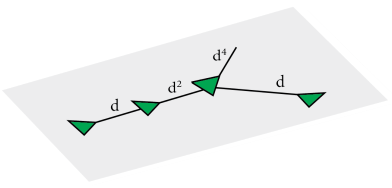


$\rho^{\mathrm{P}}_{H} = \rho^{\mathrm{P}}_{1,H} + \rho^{\mathrm{P}}_{2,H} =$

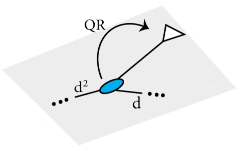


Step 3

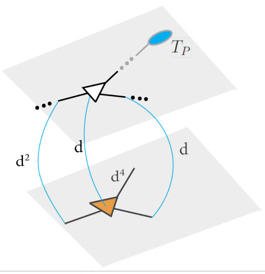
$|\psi^{\text{expanded}}\rangle =$



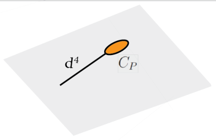
Step 4



$(UC)_H = \text{Contract}\,(C_H \cdot U) =$

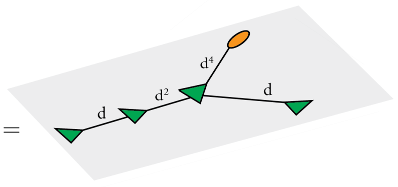


$C_P = \text{Contract}\,(T_P \cdot (UC)_H)$



Final step

$|\psi^{\text{expanded}}\rangle =$



3.3.1 GSE-TDVP Optimizations

1. Path Dependence: The first issue with this algorithm is that the growing dimension on each bond depends on the *update path* we initially chose. For instance, in the previously discussed example, no matter how many times GSE is performed, the bond between nodes S and M will not exceed a dimension of d.

Solution : We sacrifice the efficiency of initializing the *main path* M^i_{up} from the furthest leaf and initialize path separately for different GSE during the evolution, i.e., M^i_{up} are constructed by randomly selecting a leaf node, and the remainder follows the steps outlined in Section 2.4.1. This trade-off is acceptable, as GSE is executed significantly fewer times during evolution compared to TDVP.

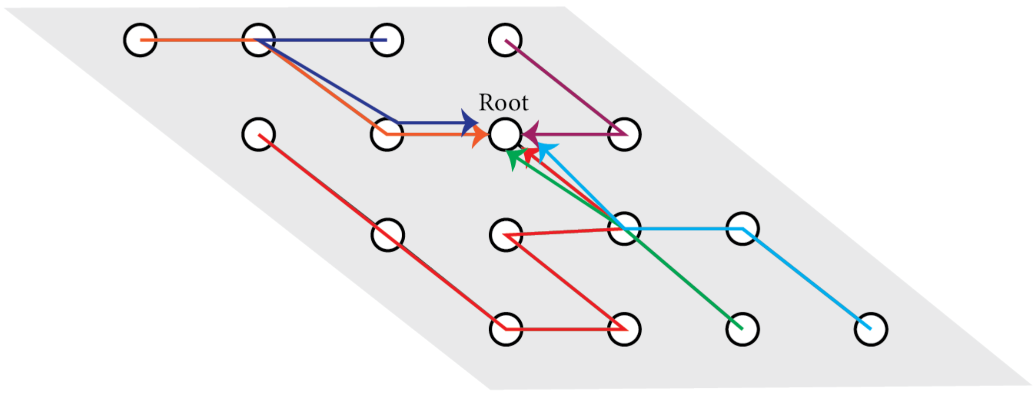


Figure 3.1: M^i_{up} is selected randomly from these options for different GSEs

2. Exponential Growth: The other challenge with the current algorithm is that, even with the application of aggressive truncation in line 1 of algorithm 5, the bond dimension of the *Krylov* basis grows exponentially $(d^1, d^2, \dots, d^n)$ with the system size.

Solution: To manage computational resources effectively, truncation should be applied directly to the PRDM after diagonalization (3.11) to control the dimension growth on bonds. Specifically, a relative truncation parameter, similar to ϵ_{rel} introduced in Section 1.4.4, is employed to retain significant components by discarding those below a fraction of the largest eigenvalue. However, a fixed ϵ_{rel} proves to be inefficient, as the precision of truncation must be adjusted dynamically as the state evolves. For instance, beginning with a very large ϵ_{rel} can lead to rapid bond growth initially, which may be unnecessary. To ensure stable bond growth, ϵ_{rel} is dynamically adapted based on feedback from the results of the expansion.

First, a fixed acceptable range `R` for the total expanded bond dimension is defined. If the difference between total bonds dimension after and before expansion falls within this range, we proceed to the next TDVP time-step. If not, the relative truncation parameter is increased or decreased by predefined factors (`increase_factor` or `decrease_factor`) depending on whether the total expanded bond dimension exceeds or falls below `R`. A new GSE computation is performed with the updated parameter, and this procedure is repeated for a maximum of `num_trials` iterations within the GSE_increase(_decrease) function until the desired *expanded_dim* is identified. The updated relative truncation parameter ϵ_{rel} after each GSE execution is used as the initial parameter for the subsequent GSE.

Since over-expansion is more likely during the evolution, it is more efficient to set `increase_factor` larger than `decrease_factor` to minimize the number of GSE executions, i.e. If under-expansion occurs during the execution of the **GSE__increase** function, the process switches to **GSE__decrease**, and start and growing expanded bond from lower bound of `R`. After achieving `max_total_bond` the TDVP runs without GSE until the `final_time` $= T$.

With this approach, stable and gradual bond growth can be achieved under controlled conditions.

Algorithm 7: GSE with Dynamical Truncation Parameter

```
Input: $|\psi\rangle$, $\epsilon_{\text{rel}}$, R, max_trials, increase_factor, decrease_factor, max_total_bond
Output: $|\psi^{\text{ex}}\rangle$, $\epsilon'_{\text{rel}}$
1  Initial Expansion Attempt: Run GSE on $|\psi\rangle$ with $\epsilon_{\text{rel}}$ to produce $|\psi^{\text{ex}}\rangle$
2  Compute:
       $expanded\_dim \leftarrow \sum_{\text{bond}\in|\psi^{\text{ex}}\rangle} \text{dimension(bond)} - \sum_{\text{bond}\in|\psi\rangle} \text{dimension(bond)}$
3  Check Expansion Dimension:
4  if $expanded\_dim \in R$ then
5      Acceptable expansion found in initial attempt.
6      $\epsilon'_{\text{rel}} \leftarrow \epsilon_{\text{rel}}$
7  else
8      if $expanded\_dim >$ max(R) then
9          Handle Over-expansion: Increase $\epsilon_{\text{rel}}$
10
               $|\psi^{\text{ex}}\rangle, \epsilon'_{\text{rel}}, switch\_to\_decrease \leftarrow$ GSE__increase($|\psi\rangle, \epsilon_{\text{rel}}$, max_trials, R, increase_factor)
11         if $switch\_to\_decrease$ then
12             Run GSE in decrease mode if expanded_dim falls below acceptable range
13
                   $|\psi^{\text{ex}}\rangle, \epsilon'_{\text{rel}} \leftarrow$ GSE__decrease($|\psi\rangle, \epsilon_{\text{rel}}$, max_trials, R, decrease_factor)
14         end
15     else if $expanded\_dim <$ min(R) then
16         Handle Under-expansion: Decrease $\epsilon_{\text{rel}}$
17
               $|\psi^{\text{ex}}\rangle, \epsilon'_{\text{rel}} \leftarrow$ GSE__decrease($|\psi\rangle, \epsilon_{\text{rel}}$, max_trials, R, decrease_factor)
18     end
19 end
20 check overgrown bond dimensions:
21 compute new $expanded\_dim$
22 if $expanded\_dim >$ max_total_bond then
23     $|\psi^{\text{ex}}\rangle \leftarrow |\psi\rangle$
24     Terminate GSE for the rest of the evolution.
25 return $|\psi^{ex}\rangle$, $\epsilon'_{rel}$
```

Algorithm 8: GSE_increase

```
Input: $|\psi\rangle$, $\epsilon_{\text{rel}}$, max_trials, R, increase_factor
Output: $|\psi^{\text{ex}}\rangle$, $\epsilon'_{\text{rel}}$, switch_to_decrease
1  $num\_trials \leftarrow 0$
2  while $num\_trials < max\_trials$ do
3      $num\_trials \leftarrow num\_trials + 1$
4      Increase $\epsilon'_{\text{rel}}$ :  $\epsilon'_{\text{rel}} \leftarrow \epsilon'_{\text{rel}} +$ increase_factor
5      $|\psi^{\text{ex}}\rangle \leftarrow$ Run GSE on $|\psi\rangle$ with $\epsilon'_{\text{rel}}$
6      Compute :  $expanded\_dim \leftarrow \sum_{\text{bond}\in|\psi^{\text{ex}}\rangle} \text{dimension(bond)} - \sum_{\text{bond}\in|\psi\rangle} \text{dimension(bond)}$
7      Check expanded_dim :
8      if $expanded\_dim \in R$ then
9          return $|\psi^{\text{ex}}\rangle$, $\epsilon'_{\text{rel}}$, False
10     else
11         if $expanded\_dim < \min(R)$ then
12             Switch to decrease mode :
13             if $expanded\_dim \leq 0$ then
14                 $|\psi^{\text{ex}}\rangle \leftarrow |\psi\rangle$
15             end
16             return $|\psi^{\text{ex}}\rangle$, $\epsilon'_{\text{rel}}$, True
17     end
18 end
19 Exceeded Trials Without Success
20 $\epsilon'_{\text{rel}} \leftarrow \epsilon'_{\text{rel}} +$ increase_factor
21 return $|\psi\rangle$, $\epsilon'_{\text{rel}}$, False
```

Algorithm 9: GSE_decrease

```
Input: $|\psi\rangle$, $\epsilon_{\text{rel}}$, max_trials, R, decrease_factor
Output: $|\psi^{\text{ex}}\rangle$, $\epsilon'_{\text{rel}}$
1  $num\_trials \leftarrow 0$
2  while $num\_trials < max\_trials$ do
3      $num\_trials \leftarrow num\_trials + 1$
4      Decrease $\epsilon'_{\text{rel}}$ :  $\epsilon'_{\text{rel}} \leftarrow \epsilon'_{\text{rel}} +$ decrease_factor
5      $|\psi^{\text{ex}}\rangle \leftarrow$ Run GSE on $|\psi\rangle$ with $\epsilon'_{\text{rel}}$
6      Compute :  $expanded\_dim \leftarrow \sum_{\text{bond}\in|\psi^{\text{ex}}\rangle} \text{dimension(bond)} - \sum_{\text{bond}\in|\psi\rangle} \text{dimension(bond)}$
7      Check expanded_dim :
8      if $expanded\_dim \in R$ then
9          return $|\psi^{\text{ex}}\rangle$, $\epsilon'_{\text{rel}}$
10     else
11         if $expanded\_dim > \max(R)$ then
12             return $|\psi\rangle$, $\epsilon'_{\text{rel}}$
13     end
14 end
15 Exceeded Trials Without Success
16 $\epsilon'_{\text{rel}} \leftarrow \epsilon'_{\text{rel}} +$ decrease_factor
17 return $|\psi\rangle$, $\epsilon'_{\text{rel}}$
```

3. Bottleneck in local updates: As we go to network structures with higher WPLI, we face a bottleneck in the action of the local effective Hamiltonian ((2.35) with $n > 2$) acting on tensors at branching sites (2.27). This creates a challenge for efficiently expanding bonds in TTN.

Solution : The representational power of TTNS with higher WPLI justifies using approximation techniques to compute each single `time_step` $= \delta t$ evolution of site tensors C_j or link tensors Λ_j

$$e^{-i\delta t H_j^{1\mathrm{s}}} C_j \quad , \quad e^{i\delta t H_j^{b}\tau} \Lambda_l(t)$$

with relatively aggressive parameters when the dimension of the effective Hamiltonian is larger than `size_threshold`.

We implemented the *Krylov subspace projection* method introduced in *Expokit: a software package for computing matrix exponentials* [98] to circumvent computing the full matrix exponential. While the core of the algorithm is based on a standard *Krylov subspace projection* approach, these enhancements yields both performance gains and accuracy guarantees:

1. **Adaptive time-stepping with sub-intervals.** The algorithm approximates a time-step δt evolution by internally dividing this interval into a sequence of smaller intervals $\tau_1, \tau_2, \ldots$, such that $\sum_k \tau_k = \delta t$. These sub-steps τ are chosen adaptively based on multiple error indicators during each sub-step and constrained by safety factors to limit the risk of overshooting. This approach ensures that the approximation stays within a user-defined `error_tol`, while avoiding unnecessary computations or inaccuracies that might occur with a fixed step size.

2. **Early termination of the Arnoldi process.** In large-scale numerical exponentiation, the Arnoldi iteration can often capture the most significant part of the exponential's action with fewer vectors. Once the residual norm drops below a threshold, the iteration terminates early. This avoids building an unnecessarily large *Krylov* basis when the main part of the exponential map is already well-approximated.

These features make the algorithm more accurate and efficient compared to a basic *Krylov* projection method, *Chebyshev* [99] and *pade approximatio*n [100].

Here we briefly discuss the core functionality of adaptive *Krylov* exponential propagation algorithm:

Aim : approximate $w(t) = e^{tA}v$:

Algorithm Inputs :

- `t` $= \pm i\delta t$, `A` $= H_j^{1\mathrm{s}}$ or H_j^b , `v` $= C_j$ or Λ_j
- `error_tol`: The desired accuracy threshold for controlling approximation errors
- `krylov_dim`: The *Krylov* subspace dimension used for projection

Algorithm constants :

- safety factors : $\gamma = 0.9, \quad \delta = 1.2,$
- Tolerance for Arnoldi process termination : $\mathrm{btol} = 1 \times 10^{-7}$
- maximum number of time-step refinements : $\mathrm{max\_iter} = 10$

The algorithm starts with an initial τ_0, constructs a *Krylov* subspace for the current τ_k, adaptively adjusts τ_k based on error estimates, and approximates the action of the matrix exponential on the vector for the accepted τ_k. The algorithm repeats the adaptive process over successive sub-intervals until the accumulated time t_k reaches the desired final time t. The final approximation is then given by $w(t) \approx e^{tA}v$.

The algorithm computes a sequence of vectors: $w_0, w_1, w_2, \ldots, w_K$

where $w_0 = v$ is the initial vector, and each w_k represents the approximation after the k-th sub-interval τ_k. At each sub-interval, the approximation is updated as: $w_k \approx e^{\tau_k A} w_{k-1}$, where $t_k = \sum_{i=1}^{k} \tau_i$ is the accumulated time after k subintervals. The iteration starts from $k = 1$ and continues until t_k reaches or exceeds the total time t. If t_k is about to exceed t, the algorithm dynamically adjusts the current τ_k to match the remaining time, ensuring that $t_k = t$ exactly after the final step.

***Krylov* Subspace Construction:**
Define the norm of the initial vector: $\beta = \|w_{k-1}\|$.
Start with the normalized vector: $v_1 = \frac{w_{k-1}}{\beta}$.
Using the Arnoldi process, construct an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ for the *Krylov* subspace $\mathcal{K}_m(A, w_{k-1})$.
The Arnoldi steps are as follows:
For $j = 1, 2, \ldots, m$:

1. Compute a temporary vector $p = Av_j$.
2. For $i = 1, \ldots, j$, set $h_{i,j} = \langle p, v_i \rangle$ and update $p \leftarrow p - h_{i,j} v_i$.
3. Set $h_{j+1,j} = \|p\|$.
4. Break the loop if $h_{j+1,j} <$ btol.
5. Normalize $v_{j+1} = \frac{p}{h_{j+1,j}}$.

After m steps (or earlier if a breakdown occurs), define the $m \times m$ principal sub-matrix H_m as the upper-left $m \times m$ portion of the larger Hessenberg matrix formed during the process.

Local Error Estimation: To estimate the local error of the approximation, the algorithm applies the matrix exponential of the extended Hessenberg matrix $H_{m+1,m}$ to the first basis vector:

$$f = e^{\tau_k H_{m+1,m}} e_1 \in \mathbb{C}^{m+1},$$

where $H_{m+1,m}$ is an $(m+1) \times m$ Hessenberg matrix. For the computation, $H_{m+1,m}$ is conceptually augmented to a square $(m+1) \times (m+1)$ matrix by adding zeros to unused columns, enabling the evaluation of the matrix exponential. The vector f is decomposed as $f = \begin{pmatrix} f^{(m)} \\ f_{m+1} \end{pmatrix}$. Here, $f^{(m)} \in \mathbb{C}^m$ corresponds to the components within the original *Krylov* subspace $\mathcal{K}_m(A, w_{k-1})$, while $f_{m+1} \in \mathbb{C}$ represents the contribution from the additional basis vector, which lies outside the subspace. The component f_{m+1} serves as an indicator of how well the current *Krylov* subspace captures the action of $e^{\tau_k A}$ on w_{k-1}. Specifically, $\text{err}_1 = |\beta f_{m+1}|$ directly measures the contribution of the extra *Krylov* vector. And $\text{err}_2 = |\beta f_{m+1} \|Av_{m+1}\||$ measures how strongly A influences the additional *Krylov* vector outside the subspace.

1. If err_1 is significantly larger than err_2, it suggests that the subspace adequately represents the action of A on w_{k-1}, and thus the local error can be estimated primarily by err_2. Parameter r is set to $\frac{1}{m}$.
2. If err_1 is moderately larger than err_2, the local error is computed as $\text{err}_{\text{loc}} = \frac{\text{err}_1 \cdot \text{err}_2}{\text{err}_1 - \text{err}_2}$. Here, r is set to $\frac{1}{m}$
3. Otherwise, if err_1 is not larger than err_2, the local error is simply err_1. In this case, r is set to $\frac{1}{m-1}$.

These estimates ensure that the approximation remains within the user-defined `error_tol`, guiding the adaptive time-stepping mechanism to adjust τ_k appropriately.

Adaptive Time-Stepping: At each time-step, compare err_{loc} with the tolerance `error_tol`:

- If $\text{err}_{\text{loc}} \leq \delta\, \tau_k \left(\frac{\tau_k\, \texttt{error\_tol}}{\text{err}_{\text{loc}}} \right)^r$:
  - Accept the current τ_k and advance the solution by time τ_k : $t_k \leftarrow t_k + \tau_k$,
  - Approximate $e^{\tau_k A} w_{k-1} \approx \beta V_m e^{\tau_k H_m} e_1$.
  - Update the next initial vector : $w_k \leftarrow e^{\tau_k A} w_{k-1}$,
- If $\text{err}_{\text{loc}} > \texttt{error\_tol}$: Adjust the step size: $\tau_k \leftarrow \gamma \tau_k \left(\frac{\tau_k \cdot \texttt{error\_tol}}{\text{err}_{\text{loc}}} \right)^r$
- Repeat the error estimation and step size adjustment for up to max_iter times. If the error remains above tolerance after max_iter refinements, raise an exception.

4. Benchmark

4.1 Tensor Jump Method (TJM)

4.1.1 Overview

We utilize GSE-TDVP to enhance the *Tensor Jump Method* (TJM) [3], which is an extended version of the *Monte Carlo Wave-Function* (MCWF) for TNS. This method is designed to recover the *Markovian quantum process* described by the master equation in *Lindblad* form.

$$\frac{d\rho}{dt} = -i\,[H_0, \rho] + \sum_{m=1}^{k} \gamma_m \left(L_m\, \rho\, L_m^\dagger - \frac{1}{2}\{L_m^\dagger L_m,\, \rho\} \right), \tag{4.1}$$

where H_0 is the system Hamiltonian and L_m are the *Lindblad* jump operators with coupling factors γ_m.

The MCWF method decomposes the evolution of an open quantum system into $\mathtt{N}$ independent trajectories, each representing a non-unitary evolution of a pure state followed by stochastic quantum jumps. Averaging over many trajectories reconstructs the full non-unitary dynamics without directly solving the *Lindblad* equation which scales as $\mathcal{O}(n\, d^{6L})$ in superoperator formalism [101], while the MCWF method, with the same setting, requires $\mathcal{O}\Big(\mathtt{N}\, n\, d^{3L}\Big)$.
This convergence is guaranteed by the statistical properties of *Monte Carlo* sampling. Defining the reconstructed density matrix as

$$\rho_{\mathtt{N}}(t) = \frac{1}{\mathtt{N}} \sum_{i=1}^{\mathtt{N}} |\Psi_i(t)\rangle \langle \Psi_i(t)|\,, \tag{4.2}$$

The deviation of the *Monte Carlo* estimate $\rho_{\mathtt{N}}(t)$ from the true density matrix $\rho(t)$ scales as

$$\hat{\sigma}\left[\rho_{\mathtt{N}}(t)\right] = \sqrt{V\left[\rho_{\mathtt{N}}(t)\right]} = \sqrt{\mathbb{E}\left[\|\rho_{\mathtt{N}}(t) - \rho(t)\|^2\right]} = O\left(\frac{1}{\sqrt{\mathtt{N}}}\right), \tag{4.3}$$

independent of the system size. This property, combined with the efficient handling of unitary evolution via TDVP, yields a scalable simulation method for open quantum systems. It can achieve comparable accuracy with substantially reduced computation time compared to state-of-the-art MPDO *Lindbladian* approaches (e.g., via the *LindbladMPO* package [102]).

4.1.2 Single Trajectory Evolution Steps

Each trajectory in the TJM evolves under the non-Hermitian effective Hamiltonian $H = H_0 + H_D$, where H_0 represents the unperturbed part of the system, while $H_D = -\frac{i}{2} \sum_{m=1}^{k} \gamma_m\, L_m^\dagger\, L_m$ encodes dissipation.

To achieve second-order accuracy, we can approximates the non-unitary time-evolution operator

$U(\delta t) = e^{-i(H_0+H_D)\delta t}$ via Strang splitting :

$$U^{(i)}(\delta t) = e^{-iH_D\frac{\delta t}{2}} e^{-iH_0\delta t} e^{-iH_D\frac{\delta t}{2}} + \mathcal{O}(\delta t^3), \tag{4.4}$$

reducing the time-step error from $\mathcal{O}(\delta t^2)$ to $\mathcal{O}(\delta t^3)$, which allows for larger time steps or fewer computational resources for a given precision. By adapting this splitting, TJM is then incorporate stochastic jumps, resulting in the piecewise function $F_j[\delta t]$

$$F_j[\delta t] = \begin{cases} \mathcal{J}_e[\delta t]\, D\left[\frac{\delta t}{2}\right] & \text{if } j = 0, \\ \mathcal{J}_e[\delta t]\, D[\delta t]\, U[\delta t] & \text{if } 0 < j < n, \\ \mathcal{J}_e[\delta t]\, D\left[\frac{\delta t}{2}\right]\, U[\delta t] & \text{if } j = n. \end{cases} \tag{4.5}$$

Which defines the sequence of operations applied to the state at each time-step j, where $j = 0, 1, \dots, n$, and $n = \frac{T}{\delta t}$ is the total number of time steps for a simulation duration T. Here, the unitary evolution operator $U[\delta t] = e^{-iH_0\delta t}$ evolves the state with TDVP algorithm, while the dissipative operator $D[\delta t] = e^{-iH_D\delta t}$ accounts for the non-unitary effects of the environment over time-step δt, which reduces the state's norm due to dissipation. Similarly, $D\left[\frac{\delta t}{2}\right]$ applies the dissipation over half the time-step at the simulation boundaries to maintain the symmetry of the Strang splitting. And $\mathcal{J}_e[\delta t]$ represents the stochastic jump process, which determines whether a quantum jump occurs based on a random variable $\epsilon \in [0, 1]$. The probability of a jump is computed as

$$\delta p = \sum_{m=1}^{k} \delta t \gamma_m \langle \Psi(t)| L_m^\dagger L_m |\Psi(t)\rangle. \tag{4.6}$$

If $\epsilon < \delta p$, a jump operator L_m is applied with probability

$$\Pi_m = \frac{\delta t \gamma_m \langle \Psi(t)| L_m^\dagger L_m |\Psi(t)\rangle}{\delta p}, \tag{4.7}$$

followed by normalization. Otherwise, the state is simply normalized to account for the dissipative reduction in norm.

Note that the Strang splitting introduces a half-time-step lag between the unitary and dissipative evolutions, which is only fully corrected when the final operator $F_n[\delta t]$ is applied. To address this and enable accurate sampling at each time-step, TJM employs a sampling ($|\Phi\rangle$), initialized as $|\Phi(0)\rangle = F_0[\delta t]|\Psi(0)\rangle$ and evolved iteratively via $|\Phi((j+1)\delta t)\rangle = F_j[\delta t]|\Phi(j\delta t)\rangle$. The quantum state $|\Psi(j\delta t)\rangle$ is then retrieved by applying $F_n[\delta t]$ to the sampling MPS, ensuring precise sampling while maintaining second-order accuracy.

For more detailed explanations about the implementation, please refer to the original article, particularly sections III.A and III.B [3].

4.1.3 TJM Optimization

TJM utilizes TDVP for the unitary part of the evolution, while incorporating dissipation and stochastic jump procedures to account for the non-unitary dynamics. The original paper presents a *Hybrid strategy* that begins with 2TDVP until reaching a certain maximum bond dimension, then switches to 1TDVP. In contrast, the GSE-TDVP strategy can enrich the state with *Krylov* basis states throughout the simulation. This approach allows virtual bond dimensions to grow to desired values at any stage of the evolution, eliminating the need for 2-site updates and state truncation. Consequently, it enables more stable bond growth, better distribution of computational resources, exploration of larger bond dimensions, and ultimately higher accuracy. Moreover, we extend TJM to the TTN framework. First, we validate that

TTNS can encode 2D states and operators, better than MPS, by benchmarking a solvable 3×3 lattice model. We then take this approach further by performing a large-scale simulation on an 8×8 lattice, demonstrating the scalability of the method on consumer-grade CPUs.

4.2 2D Transverse-Field Ising Dissipative Simulation

In this simulation, we explore the *2D transverse-field Ising model's* coherent dynamics—driven by its Hamiltonian— supplemented by non-unitary evolution due to environmental interactions, leading to dissipative processes, which causes its dynamics to deviate from purely unitary (coherent) evolution.

Hamiltonian. We consider the two-dimensional (2D) transverse-field Ising model on an $M \times M$ square lattice:

$$\hat{H} \;=\; -J \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \Big(\hat{\sigma}^z_{(i,j)} \hat{\sigma}^z_{(i+1,j)} \;+\; \hat{\sigma}^z_{(i,j)} \hat{\sigma}^z_{(i,j+1)} \Big) \;-\; g \sum_{i=0}^{M-1} \sum_{j=0}^{M-1} \hat{\sigma}^x_{(i,j)}. \tag{4.8}$$

The parameter J sets the strength of the nearest-neighbor coupling while g determines the transverse field amplitude. Here, the indices i and j run from 0 to $M-1$ with the site labels starting from $(i,j) = (0,0)$.

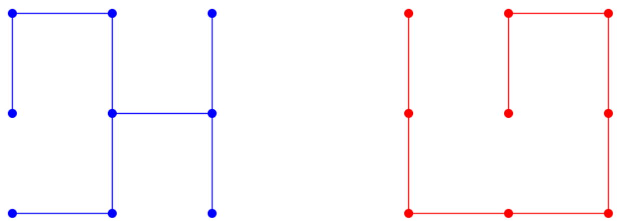
Noise model. Here, the dissipative processes are introduced through set of single-site *Lindblad* jump operators acting on every site (i,j) with uniform coupling factors.

- Relaxation Operator $\hat{\sigma}^-_{(i,j)}$ with Coupling Strength γ^-:
  The operator $\hat{\sigma}^-_{(i,j)}$ drives each site from its excited state ($|1\rangle$) to its ground state ($|0\rangle$). This is analogous to spontaneous emission in atomic systems or spin-lattice relaxation in magnetic materials. Physically, this process suppress quantum fluctuations (induced by the transverse field) and pushing the system toward a more classical, ordered state.
- Dephasing Operator: $\hat{\sigma}^z_{(i,j)}$ with Coupling Strength γ^z:
  The operator $\hat{\sigma}^z_{(i,j)}$ introduces phase randomization, disrupting the system's quantum coherence without changing the populations of the computational basis states. This process eliminates off-diagonal elements of the density matrix, reducing quantum superpositions and localizing the system in the Z-basis.

The interplay between these dissipative mechanisms and the system's coherent dynamics strikes a balance where coherent quantum effects are partially preserved but steadily diminished by environmental noise. This setup allows us to study how dissipation influences the emergence of classical behavior and the modification of quantum critical points in a higher-dimensional quantum system.

4.2.1 Experiment 1

TTN structures. In first experiment we compare the absolute error of snake-like and TTN mappings on 3×3 lattice (M=3) which is the largest model we could obtain the exact result with *qutip.mesolve* [103] package.



Parameters. The simulation parameters are summarized in Table 4.1.

Table 4.1: Simulation Parameters

Second-order 1TDVP Parameters	
Coupling factors	$J = 1, \quad g = 0.5$
Initial state	$\lvert\psi_0\rangle = \bigotimes_{i=0}^{M-1} \bigotimes_{j=0}^{M-1} \lvert 1\rangle_{(i,j)}$
`time_step`	$\delta t = 0.1$
`final_time`	$T = 10$
TJM Parameters	
Coupling factors	$\gamma^{-} = 0.1, \quad \gamma^{z} = 0.1$
Number of trajectories `N`	200
Matrix Exponential Parameters	
`size_threshold`	500
`error_tol`	1×10^{-5}
`krylov_dim`	6
GSE Parameters	
Krylov space dimension `m`	6
Coefficient τ	0.1
`svd_truncation_parameter`	$\chi_{\max} = +\infty, \ \epsilon_{\text{rel}} = 0.001, \ \epsilon_{\text{total}} = -\infty,$
ϵ_{rel}	0.2
Acceptable range `R`	$(1, 10)$
`expansion_step`	10
`max_trials`	10
`increase_factor`	0.08
`decrease_factor`	0.02
`max_total_bond`	TTN : 40 , snake-like : 50

Results

CPU time. It takes 22 minumtes to run the both simulation.

Energy Error.
Defining discrete times are defined as

$$t_m = m\,\delta t, \quad m = 0, 1, \ldots, n, \quad \text{with final time } T = n\,\delta t.$$

Let $\langle \hat{H} \rangle_{\text{approx}}(t_m)$ and $\langle \hat{H} \rangle_{\text{exact}}(t_m)$ denote the approximate and exact energy expectation values at the discrete times. We observe :

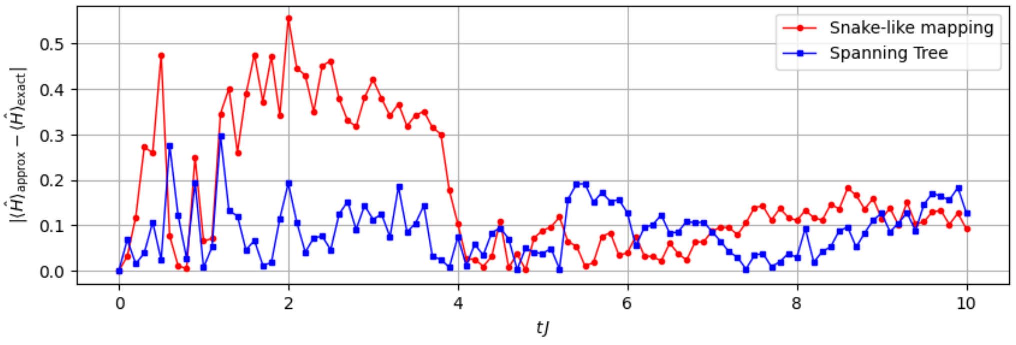


Figure 4.1: Absolute error of the energy expectation value for the snake-like and TTN mappings.

Computing *Accumulated Energy Error (AEE)* over the time interval $[0, T]$,

$$\text{AEE}(T) = \sum_{m=0}^{n} \left| \langle E \rangle_{\text{approx}}(t_m) - \langle E \rangle_{\text{exact}}(t_m) \right|. \tag{4.9}$$

we get TEE(10) = 12.0 for the snake-like mapping and TEE(10) = 8.9 for the TTN mapping, which confirms the overall accuracy of Spanning tree approach.

Local Errors. Let $\langle \hat{\sigma}^x_{(i,j)} \rangle_{\text{approx}}(t_m)$ and $\langle \hat{\sigma}^x_{(i,j)} \rangle_{\text{exact}}(t_m)$ be the approximate and exact expectation values of the Pauli X operator at site (i, j) at time t_m. Then the *Mean Accumulated Error (MAE)* up to time T

$$\text{MAE}(T) = \frac{1}{9} \sum_{m=0}^{n} \sum_{i=0}^{2} \sum_{j=0}^{2} \left| \langle \hat{\sigma}^x_{(i,j)} \rangle_{\text{approx}}(t_m) - \langle \hat{\sigma}^x_{(i,j)} \rangle_{\text{exact}}(t_m) \right|, \tag{4.10}$$

captures the per-site absolute error summed over all sites at each time-step and then cumulatively over time.

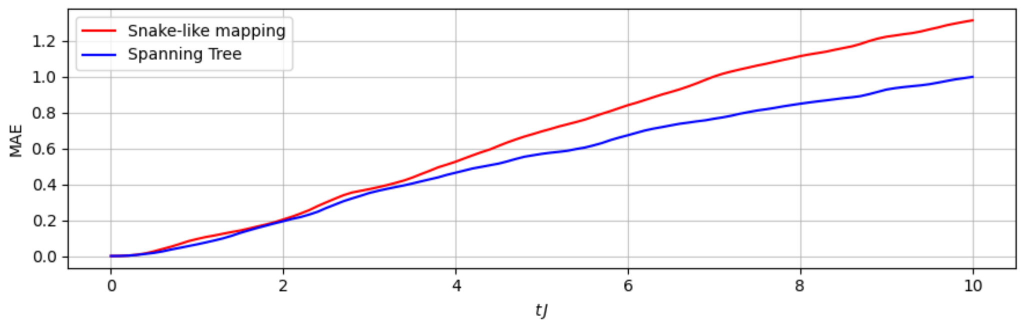


Figure 4.2: Mean Accumulated Error of the Pauli X operator for the snake-like and TTN mappings.

Bond Growth. For each trajectory $r \in \{1, \ldots, \mathtt{N}\}$, let $\chi^{(r)}(t_m)$ be the total bond dimension at time t_m.
Then the mean bond dimension at time t_m is

$$\bar{\chi}(t_m) = \frac{1}{N} \sum_{r=1}^{N} \chi^{(r)}(t_m), \tag{4.11}$$

with standard deviation

$$\hat{\sigma}_{\chi}(t_m) = \sqrt{\frac{1}{N} \sum_{r=1}^{N} \left(\chi^{(r)}(t_m) - \bar{\chi}(t_m) \right)^2}. \tag{4.12}$$

This plot displays the evolution of $\bar{\chi}(t)$ over time along with a shaded region spanning from $\bar{\chi}(t) - \hat{\sigma}_{\chi}(t)$ to $\bar{\chi}(t) + \hat{\sigma}_{\chi}(t)$, thereby capturing both the overall trend and the variability across trajectories.

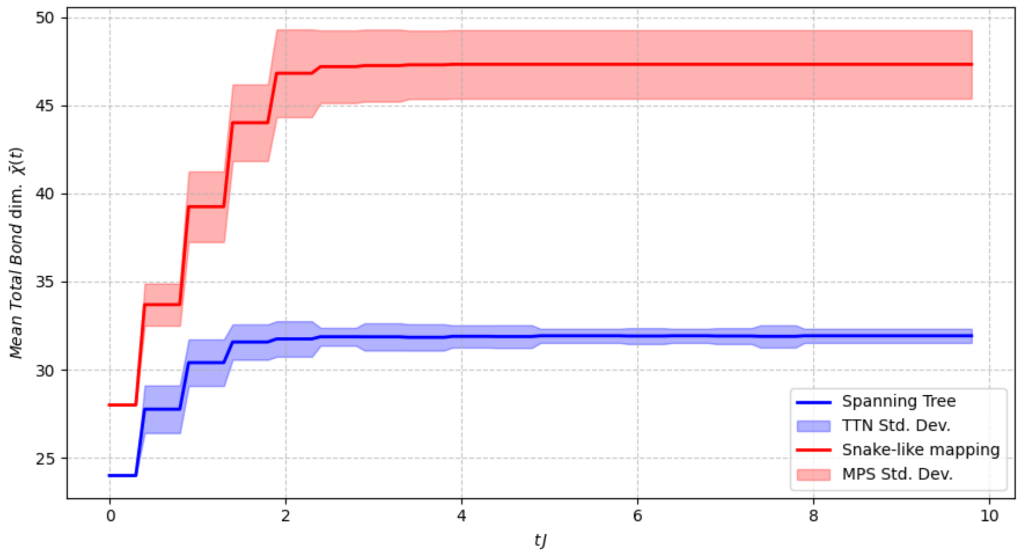
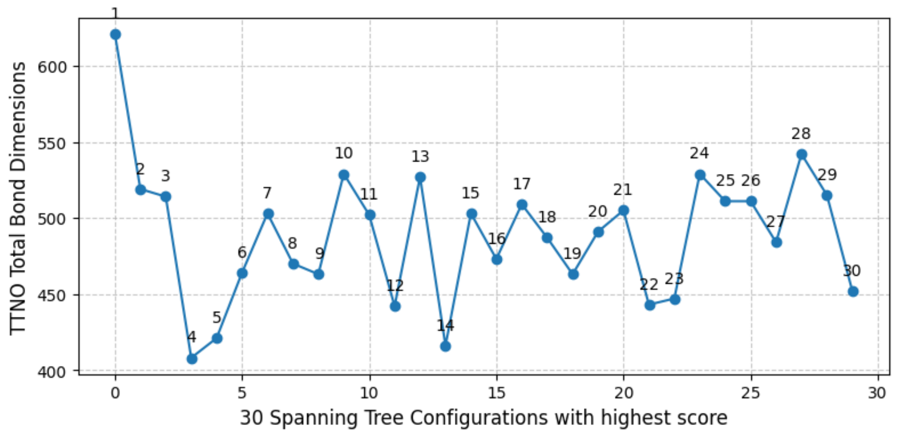
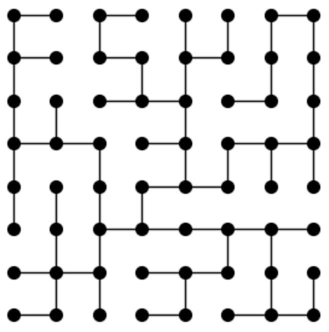


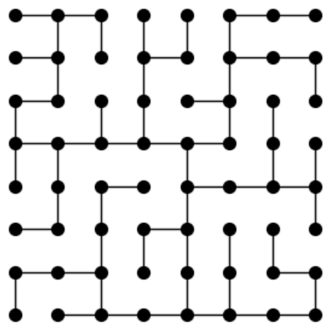
Figure 4.3: Mean total bond dimension over all trajectories for the snake-like and TTN mappins.

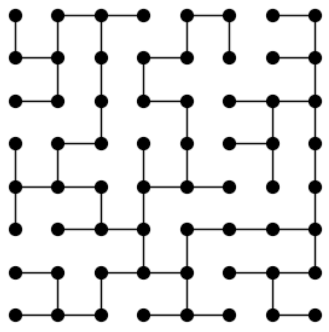
4.2.2 Experiment 2

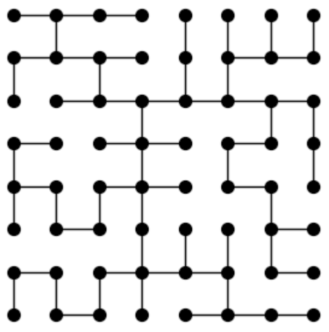
TTN structures. In the second experiment, we simulate the same model on a larger 8×8 lattice ($M = 8$). In the first step, we perform the profiling described in Section 2.5.3 on 300 randomly generated spanning trees to identify the most efficient structures. Since, in TJM, individual trajectories evolve independently, to better capture the true dynamics, we can add another layer of randomness to the algorithm by letting the algorithm to randomly select spanning tree for each trajectory. Therefore, we select the top 30 structures with the highest scores (using weights $\alpha = 1$, $\beta = 1$, $\gamma = 1$), construct their corresponding TTNOs, and, to minimize computational cost, discard those with a total bond dimension exceeding 450.

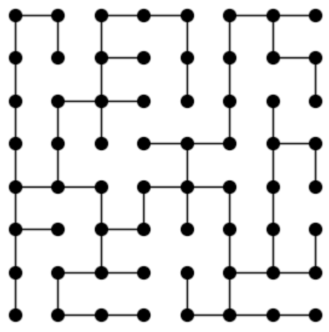



(a) structure #4


(b) structure #5


(c) structure #12


(d) structure #14


(e) structure #22

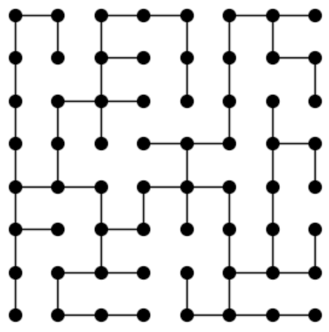

(f) structure #23

Figure 4.4: Selected structures with the highest scores with corresponding TTNO bond dimension bellow 450.

Parameters. The simulation parameters are summarized in the following Table:

Table 4.2: Simulation Parameters

Second-order 1TDVP Parameters	
Coupling factors	$J = 1, \quad g = 0.5$
Inittial state	$\|\psi_0\rangle = \left(\bigotimes_{i=0}^{\frac{M}{2}-1} \bigotimes_{j=0}^{M-1} \|1\rangle_{(i,j)} \right) \otimes \left(\bigotimes_{i=\frac{M}{2}}^{M-1} \bigotimes_{j=0}^{M-1} \|0\rangle_{(i,j)} \right)$
`time_step`	$\delta t = 0.1$
`final_time`	$T = 10$
TJM Parameters	
Coupling factors	$\gamma^{-} = 0.1, \quad \gamma^{z} = 0.1$
Number of trajectories `N`	300
Matrix Exponential Parameters	
`size_threshold`	500
`error_tol`	1×10^{-5}
`krylov_dim`	6
GSE Parameters	
Krylov space dimension `m`	6
Coefficient τ	0.1
`svd_truncation_parameter`	$\chi_{\max} = +\infty,\ \epsilon_{\text{rel}} = 0.001,\ \epsilon_{\text{total}} = -\infty,$
ϵ_{total}	10^{-15}
Acceptable range `R`	$(10, 50)$
`expansion_step`	12
`max_trials`	10
`increase_factor`	10
`decrease_factor`	20
`max_total_bond`	340

Remark. For larger lattices, working with total truncation ϵ_{total} instead of ϵ_{rel} in algorithm 7 proves more flexible, as it allows for expanding bond dimensions with finer resolution. Consequently, the cutoff threshold introduced in Section 1.4.4, which is applied to diagonalized PRDM singular values in Equation (3.11), is solely determined by ϵ_{total}. With this modification, the GSE_increase algorithm 8 and GSE_decrease algorithm 9 require ϵ_{total} to be multiplied (instead of summation) by increase_factor or divided by decrease_factor (instead of subtraction), respectively.

Results

CPU time. With 16 GB of available RAM, a maximum of 3 concurrent processes can be used to parallelize this computation, and it required approximately 800 minutes (≈ 2.6 minutes per trajectory) to complete on a system equipped with an *Intel Core i7-1065G7* processor operating at a base frequency of 1.30 GHz.

Local spin expectation values. We illustrate the local $\hat{\sigma}^z$ expectation values for each site (i,j) at 5 differet time-step.

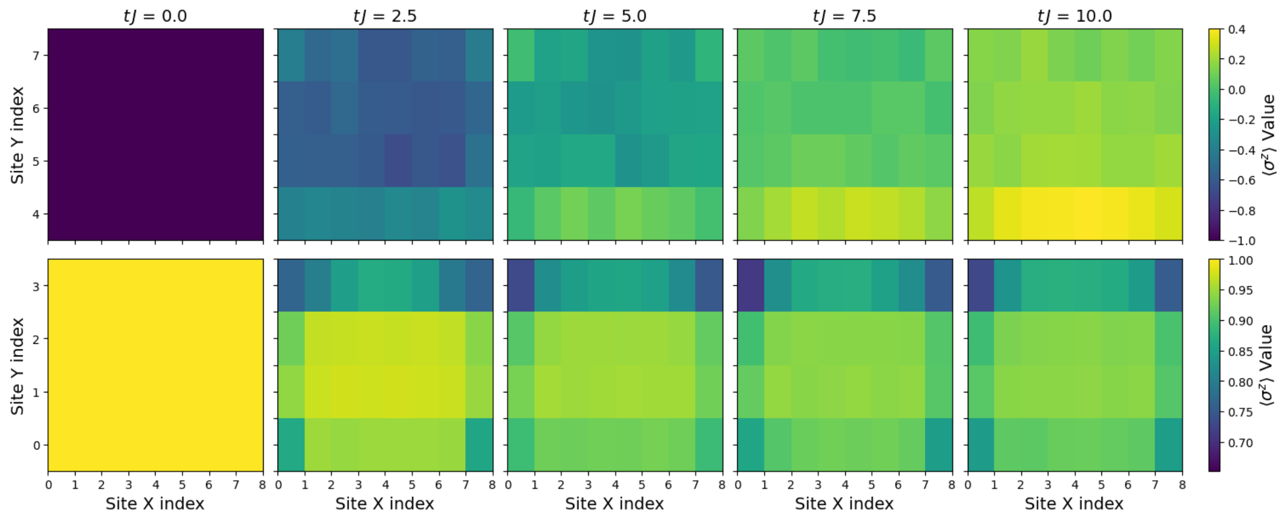


Figure 4.5: Local spin $\langle \hat{\sigma}^z_{(i,j)}(t_m)\rangle$ evolution for all sites, at $m = \{0, 25, 50, 75, 100\}$ time steps, and at two sides of domain wall separately.

Convergence. For each site (i,j) on the $M \times M$ lattice and each trajectory $r \in \{1, 2, \dots, N\}$, let $\hat{\sigma}^z_{(i,j)}(r, t_m)$ represent the local $\hat{\sigma}^z$ observable at time t_m, as measured on trajectory r. Then the final result is achieved by averaging over N trajectories.

$$\langle \hat{\sigma}^z_{(i,j)}\rangle_N(t_m) \;=\; \frac{1}{N}\sum_{r=1}^{N}\langle \hat{\sigma}^z_{(i,j)}\rangle(r, t_m). \tag{4.13}$$

Since we have used 300 trajectories, for any chosen $N \leq 300$, we compare $\langle \hat{\sigma}^z\rangle_N$ to the reference average $\langle \hat{\sigma}^z\rangle_{300}$. A convenient measure of their difference at time t_m is the mean absolute deviation over *all* sites:

$$D_N^{300}(t_m) \;=\; \frac{1}{M^2}\sum_{i=0}^{M-1}\sum_{j=0}^{M-1}\left|\langle \hat{\sigma}^z_{(i,j)}\rangle_N(t_m) \;-\; \langle \hat{\sigma}^z_{(i,j)}\rangle_{300}(t_m)\right|. \tag{4.14}$$

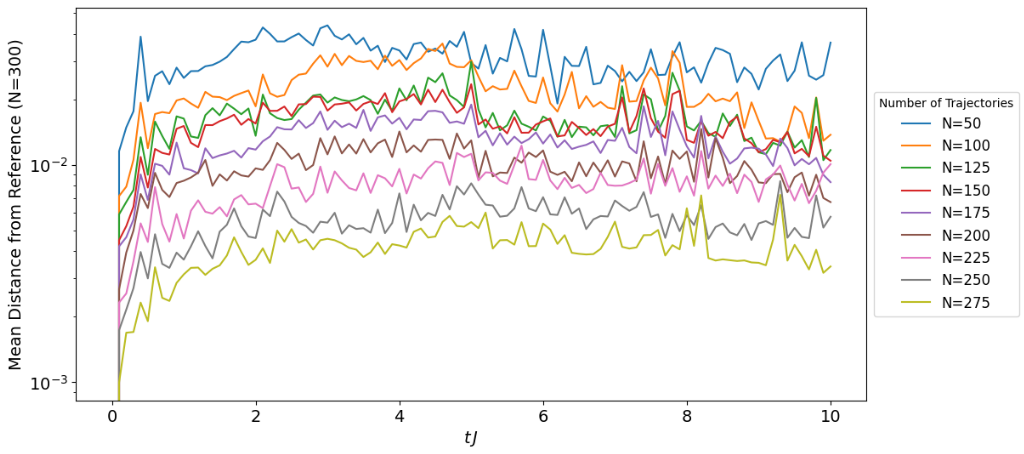
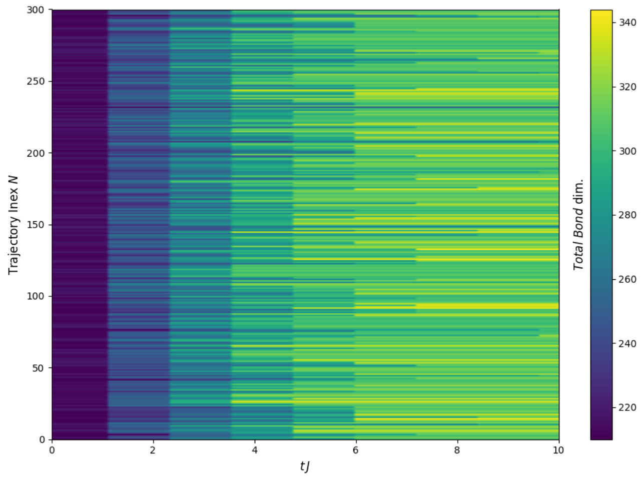


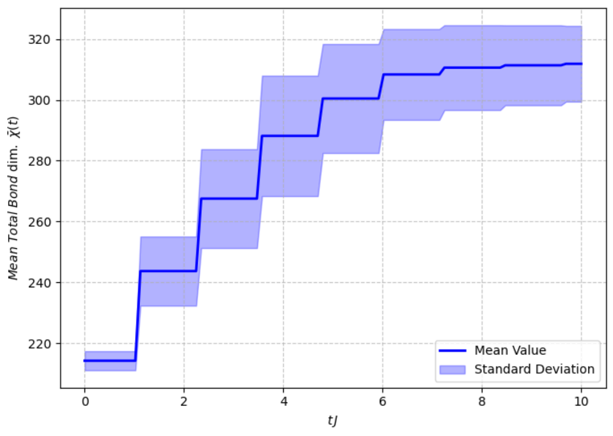
Figure 4.6: Distance $D_N^{300}(t_m)$ plotted against time steps $m = \{0, 1, \dots, 100\}$ for $N = 300$ trajectories.

As N increases, $D_N(t_m)$ approaches zero, demonstrating that the local $\langle\hat{\sigma}^z\rangle$ measurements converge to the reference solution at $N_{\text{ref}} = 300$.

Bond Growth. In the following plots, we display the evolution of total virtual bond dimensions.



(a) Total Bond Dimensions for each trajectory.



(b) Mean Total Bond Dimensions over all trajectories.

Figure 4.7: Total Virtual Bond Dimension growths as GSE is applied every `expansion_step` = 12, until the `max_total_bond` = 340 is reached.

We observe that the total bond dimensions grow steadily over time within a fixed acceptable range `R` = 50, until reaching a maximum value of 340. The mean total bond dimension over all trajectories shows a similar trend, with a slight increase in variability as we each trajectory randomly chose between 6 configurations.

Final Discussion and Outlook

As demonstrated in the final chapter, integrating the GSE algorithm into the TDVP, can contribute to the scalability of TN algorithms that require tracking the real time dynamic of system. This improvement could potentially extend to fermionic tensor networks and models with a broader variety of geometries like honeycomb or triangular lattices.

Several potential optimizations could be explored in the future. For instance, in algorithm 4, we execute GSE on every predefined intervals. This could be improved by making the interval adaptive. One approach I investigated involves evolving both the original and expanded states in parallel at each time step, then measuring the difference between a chosen observable of the two states. If the convergence falls below a threshold, the original state is retained, and the simulation proceeds to the next time step. Otherwise, the simulation steps back, expands the state with less aggressive truncation, and repeats the process until convergence is achieved. However, a fixed threshold turned out to not perform reliably across general cases. Alternative strategies worth exploring include using entanglement-based measures or monitoring energy conservation to determine the optimal point for bond expansion.

Another optimization to improve GSE efficiency involves leveraging the *Lanczos* method to compute a specific number of eigenvalues. We attempted to shift from using a dynamic cutoff threshold for truncating the full eigenvalue spectrum of the PRDM (3.11), to dynamically determining the number of eigenvalues computed via the *Lanczos* method. However, finding a dynamic adjustment for this number that in a way that ensures stable bond growth throughout the evolution remains challenging and requires further investigation.

Moreover, It is worth mentioning that, we investigated the performance of TDVP on TTN in *three-legged* TTN form, which was shown to be more efficient in DMRG calculation [104]. Motivated to overcome the bottleneck in local updates observed in 3.3.1, we assumed that new bonds in this form could carry information from branching nodes, potentially achieving the same accuracy with smaller matrix exponentials. However, further experiments revealed that, in general cases, this approach does not offer a favorable trade-off due to the additional computational cost of environment calculations introduced by adding new nodes.

Another approach we investigated, though unsuccessfully, was representing the density matrix on a tree and using TDVP to directly simulate the *Lindblad* equation in its superoperator formalism [105]. Although *Completely Positive and Trace-Preserving* (CPTP) condition was satisfied, we noticed that, due to non-trivial distances between density matrices, the TDVP projector becomes ineffective. Consequently, TDVP appears only suited for simulating pure states [106].

Acknowledgements

As I sit down to write these words, I find myself reflecting on the winding path that brought me to this moment—a path paved not just by my own efforts, but by the brilliance, generosity, and quiet dedication of others. It's a humbling thing, to stand on the shoulders of those who see farther than I ever could alone.

I would like to begin by expressing my admiration for Professor Christian B. Mendl for his visionary leadership in establishing and guiding the quantum computing group at the Technical University of Munich and for bringing together a community of experts who push the boundaries of research in this field. Although I did not have the privilege to work directly with him, his efforts in assembling such a vibrant group made it possible for me to explore this area in a supportive environment.

My deepest gratitude, though, belongs to Richard Milbradt. Richard, a PhD student in the group, was my guide, my collaborator, my anchor through this process. There's something profound about someone who not only masters a field but opens it up for others to walk through. Without his innovative work on PyTreeNet library—the very foundation of my work—which provides essential tools and insights needed to navigate the complexities of tensor networks, this achievement would not have been possible. Richard's patience, his insights, and his willingness to walk alongside me as I stumbled through this work made all the difference.

And then there's Professor Elisa Ercolessi, my supervisor back at the University of Bologna. Although she was not directly involved in the research itself, her feedbacks and assistance in organizing this project helped keep me on track and navigate the chaos of organizing this project as an exchange student far from home. I'm grateful for her trust and her support.

This thesis isn't just a stack of pages or a collection of code—it's a piece of me, shaped by the people who gave me their time, their expertise, and their faith. To Professor Mendl, Richard, and Professor Ercolessi: thank you. You've not only made this work possible, but you've reminded me why we do this—why we chase the unknown, why we build, why we learn. It's about the pursuit, yes, but it's also about the people we meet along the way.

Bibliography

[1] Jutho Haegeman, J. Ignacio Cirac, Tobias J. Osborne, Iztok Pižorn, Henri Verschelde, and Frank Verstraete. Time-dependent variational principle for quantum lattices. *Physical Review Letters*, 107(7), August 2011. ISSN 1079-7114. doi: 10.1103/physrevlett.107.070601. URL http://dx.doi.org/10.1103/PhysRevLett.107.070601.

[2] Mingru Yang and Steven R. White. Time-dependent variational principle with ancillary krylov subspace. *Physical Review B*, 102(9), September 2020. ISSN 2469-9969. doi: 10.1103/physrevb.102.094315. URL http://dx.doi.org/10.1103/PhysRevB.102.094315.

[3] Aaron Sander, Maximilian Fröhlich, Martin Eigel, Jens Eisert, Patrick Gelß, Michael Hintermüller, Richard M. Milbradt, Robert Wille, and Christian B. Mendl. Large-scale stochastic simulation of open quantum systems, 2025. URL https://arxiv.org/abs/2501.17913.

[4] Mari Carmen Bañuls. Tensor network algorithms: a route map. 2022. doi: 10.48550/ARXIV.2205.10345. URL https://arxiv.org/abs/2205.10345.

[5] S. Montangero. Tensor network methods. pages 49–77, 2018. doi: 10.1007/978-3-030-01409-4_5.

[6] Jacob Biamonte and Ville Bergholm. Tensor networks in a nutshell, 2017. URL https://arxiv.org/abs/1708.00006.

[7] J. C. Slater. A simplification of the hartree-fock method. *Physical Review*, 81(3):385–390, February 1951. ISSN 0031-899X. doi: 10.1103/physrev.81.385. URL http://dx.doi.org/10.1103/PhysRev.81.385.

[8] Erik Waltersson. Performance of many-body perturbation theory calculations on 2d quantum dots. 2007.

[9] Carlos F. Bunge. *Present Status of Selected Configuration Interaction With Truncation Energy Error*, pages 3–34. Elsevier, 2018. doi: 10.1016/bs.aiq.2017.05.001. URL http://dx.doi.org/10.1016/bs.aiq.2017.05.001.

[10] Nadia Salami and Aliasghar Shokri. *Electronic structure of solids and molecules*, pages 325–373. Elsevier, 2021. ISBN 9780128188064. doi: 10.1016/b978-0-12-818806-4.00002-4. URL http://dx.doi.org/10.1016/B978-0-12-818806-4.00002-4.

[11] Masatoshi Imada, Atsushi Fujimori, and Yoshinori Tokura. Metal-insulator transitions. *Reviews of Modern Physics*, 70(4):1039–1263, October 1998. ISSN 1539-0756. doi: 10.1103/revmodphys.70.1039. URL http://dx.doi.org/10.1103/RevModPhys.70.1039.

[12] Elbio Dagotto. Correlated electrons in high-temperature superconductors. *Reviews of Modern Physics*, 66(3):763–840, July 1994. ISSN 1539-0756. doi: 10.1103/revmodphys.66.763. URL http://dx.doi.org/10.1103/RevModPhys.66.763.

[13] Leon Balents. Spin liquids in frustrated magnets. *Nature*, 464(7286):199–208, March 2010. ISSN 1476-4687. doi: 10.1038/nature08917. URL http://dx.doi.org/10.1038/nature08917.

[14] Horst L. Stormer. Nobel lecture: The fractional quantum hall effect. *Reviews of Modern Physics*, 71(4):875–889, July 1999. ISSN 1539-0756. doi: 10.1103/revmodphys.71.875. URL http://dx.doi.org/10.1103/RevModPhys.71.875.

[15] Subir Sachdev. *Quantum Phase Transitions*. Cambridge University Press, January 2000. ISBN 9780511622540. doi: 10.1017/cbo9780511622540. URL http://dx.doi.org/10.1017/CBO9780511622540.

[16] Alexander Weiße and Holger Fehske. *Exact Diagonalization Techniques*, page 529–544. Springer Berlin Heidelberg. ISBN 9783540746850. doi: 10.1007/978-3-540-74686-7_18. URL http://dx.doi.org/10.1007/978-3-540-74686-7_18.

[17] Jaan Oitmaa, Chris Hamer, and Weihong Zheng. *Series Expansion Methods for Strongly Interacting Lattice Models*. Cambridge University Press, April 2006. ISBN 9780521143592. doi: 10.1017/cbo9780511584398. URL http://dx.doi.org/10.1017/cbo9780511584398.

[18] Benedikt Bruognolo. *Tensor network techniques for strongly correlated systems: Simulating the quantum many-body wavefunction in zero, one, and two dimensions*. PhD thesis, Dissertation, July 6 2017.

[19] Emanuel Gull, Andrew J. Millis, Alexander I. Lichtenstein, Alexey N. Rubtsov, Matthias Troyer, and Philipp Werner. Continuous-time monte carlo methods for quantum impurity models. *Reviews of Modern Physics*, 83(2):349–404, May 2011. ISSN 1539-0756. doi: 10.1103/revmodphys.83.349. URL http://dx.doi.org/10.1103/RevModPhys.83.349.

[20] Matthias Troyer and Uwe-Jens Wiese. Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Physical Review Letters*, 94 (17), May 2005. ISSN 1079-7114. doi: 10.1103/physrevlett.94.170201. URL http://dx.doi.org/10.1103/PhysRevLett.94.170201.

[21] J. Eisert. Entanglement and tensor network states. 2013. doi: 10.48550/ARXIV.1308.3318. URL https://arxiv.org/abs/1308.3318.

[22] Álvaro M. Alhambra and J. Ignacio Cirac. Locally accurate tensor networks for thermal states and time evolution. 2021. doi: 10.48550/ARXIV.2106.00710. URL https://arxiv.org/abs/2106.00710.

[23] Xiangjian Qian, Jiale Huang, and Mingpu Qin. Augmenting finite temperature tensor network with clifford circuits, 2024. URL https://arxiv.org/abs/2410.15709.

[24] Brian Swingle and Xiao-Gang Wen. Topological properties of tensor network states from their local gauge and local symmetry structures, 2010. URL https://arxiv.org/abs/1001.4517.

[25] J.P.F. LeBlanc, Andrey E. Antipov, Federico Becca, Ireneusz W. Bulik, Garnet Kin-Lic Chan, Chia-Min Chung, Youjin Deng, Michel Ferrero, Thomas M. Henderson, Carlos A. Jiménez-Hoyos, E. Kozik, Xuan-Wen Liu, Andrew J. Millis, N.V. Prokof'ev, Mingpu Qin, Gustavo E. Scuseria, Hao Shi, B.V. Svistunov, Luca F. Tocchio, I.S. Tupitsyn, Steven R. White, Shiwei Zhang, Bo-Xiao Zheng, Zhenyue Zhu, and Emanuel Gull. Solutions of the

two-dimensional hubbard model: Benchmarks and results from a wide range of numerical algorithms. *Physical Review X*, 5(4), December 2015. ISSN 2160-3308. doi: 10.1103/physrevx.5.041041. URL http://dx.doi.org/10.1103/PhysRevX.5.041041.

[26] Simone Montangero, Enrique Rico, and Pietro Silvi. Loop-free tensor networks for high-energy physics. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 380(2216), December 2021. ISSN 1471-2962. doi: 10.1098/rsta.2021.0065. URL http://dx.doi.org/10.1098/rsta.2021.0065.

[27] Ning Bao, Geoffrey Penington, Jonathan Sorce, and Aron C. Wall. Holographic tensor networks in full ads/cft, 2019. URL https://arxiv.org/abs/1902.10157.

[28] Román Orús. Tensor networks for complex quantum systems. *Nature Reviews Physics*, 1(9):538–550, August 2019. ISSN 2522-5820. doi: 10.1038/s42254-019-0086-7. URL http://dx.doi.org/10.1038/s42254-019-0086-7.

[29] E. Miles Stoudenmire and David J. Schwab. Supervised learning with quantum-inspired tensor networks. 2016. doi: 10.48550/ARXIV.1605.05775. URL https://arxiv.org/abs/1605.05775.

[30] Jinhui Wang, Chase Roberts, Guifre Vidal, and Stefan Leichenauer. Anomaly detection with tensor networks, 2020. URL https://arxiv.org/abs/2006.02516.

[31] Raghavendra Selvan and Erik B Dam. Tensor networks for medical image classification, 2020. URL https://arxiv.org/abs/2004.10076.

[32] Samuel T Wauthier, Tim Verbelen, Bart Dhoedt, and Bram Vanhecke. Planning with tensor networks based on active inference. *Machine Learning: Science and Technology*, 5(4):045012, October 2024. ISSN 2632-2153. doi: 10.1088/2632-2153/ad7571. URL http://dx.doi.org/10.1088/2632-2153/ad7571.

[33] Ulrich Schollwöck. Simulations with matrix product states. In *AIP Conference Proceedings*, pages 135–225. AIP, 2012. doi: 10.1063/1.4755823. URL http://dx.doi.org/10.1063/1.4755823.

[34] Atsushi Iwaki and Chisa Hotta. Thermal pure quantum matrix product states: a simple numerical protocol for finite temperature. *Journal of Physics: Conference Series*, 2207 (1):012031, March 2022. ISSN 1742-6596. doi: 10.1088/1742-6596/2207/1/012031. URL http://dx.doi.org/10.1088/1742-6596/2207/1/012031.

[35] Angus J. Dunnett and Alex W. Chin. Matrix product state simulations of non-equilibrium steady states and transient heat flows in the two-bath spin-boson model at finite temperatures. *Entropy*, 23(1):77, January 2021. ISSN 1099-4300. doi: 10.3390/e23010077. URL http://dx.doi.org/10.3390/e23010077.

[36] Baptiste Anselme Martin, Thomas Ayral, François Jamet, Marko J. Rančić, and Pascal Simon. Combining matrix product states and noisy quantum computers for quantum simulation. *Physical Review A*, 109(6), June 2024. ISSN 2469-9934. doi: 10.1103/physreva.109.062437. URL http://dx.doi.org/10.1103/physreva.109.062437.

[37] Regina Finsterhölzl, Manuel Katzer, Andreas Knorr, and Alexander Carmele. Using matrix-product states for open quantum many-body systems: Efficient algorithms for markovian and non-markovian time-evolution. *Entropy*, 22(9):984, September 2020. ISSN 1099-4300. doi: 10.3390/e22090984. URL http://dx.doi.org/10.3390/e22090984.

[38] Yi Fan, Jie Liu, Zhenyu Li, and Jinlong Yang. Quantum circuit matrix product state ansatz for large-scale simulations of molecules. *Journal of Chemical Theory and Computation*, 19(16):5407–5417, July 2023. ISSN 1549-9626. doi: 10.1021/acs.jctc.3c00068. URL http://dx.doi.org/10.1021/acs.jctc.3c00068.

[39] Kevin C. Smith, Abid Khan, Bryan K. Clark, S.M. Girvin, and Tzu-Chieh Wei. Constant-depth preparation of matrix product states with adaptive quantum circuits. *PRX Quantum*, 5(3), September 2024. ISSN 2691-3399. doi: 10.1103/prxquantum.5.030344. URL http://dx.doi.org/10.1103/prxquantum.5.030344.

[40] Michael P. Zaletel, Roger S. K. Mong, Christoph Karrasch, Joel E. Moore, and Frank Pollmann. Time-evolving a matrix product state with long-ranged interactions. *Physical Review B*, 91(16), April 2015. ISSN 1550-235X. doi: 10.1103/physrevb.91.165112. URL http://dx.doi.org/10.1103/PhysRevB.91.165112.

[41] F. Verstraete, V. Murg, and J.I. Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, March 2008. ISSN 1460-6976. doi: 10.1080/14789940801912366. URL http://dx.doi.org/10.1080/14789940801912366.

[42] Wen-Yuan Liu, Yi-Zhen Huang, Shou-Shu Gong, and Zheng-Cheng Gu. Accurate simulation for finite projected entangled pair states in two dimensions. *Physical Review B*, 103(23), June 2021. ISSN 2469-9969. doi: 10.1103/physrevb.103.235155. URL http://dx.doi.org/10.1103/PhysRevB.103.235155.

[43] Michael Lubasch, J Ignacio Cirac, and Mari-Carmen Bañuls. Unifying projected entangled pair state contractions. *New Journal of Physics*, 16(3):033014, March 2014. ISSN 1367-2630. doi: 10.1088/1367-2630/16/3/033014. URL http://dx.doi.org/10.1088/1367-2630/16/3/033014.

[44] Philippe Corboz. Variational optimization with infinite projected entangled-pair states. *Physical Review B*, 94(3), July 2016. ISSN 2469-9969. doi: 10.1103/physrevb.94.035133. URL http://dx.doi.org/10.1103/PhysRevB.94.035133.

[45] Xie-Hang Yu, J. Ignacio Cirac, Pavel Kos, and Georgios Styliaris. Dual-isometric projected entangled pair states. *Physical Review Letters*, 133(19), November 2024. ISSN 1079-7114. doi: 10.1103/physrevlett.133.190401. URL http://dx.doi.org/10.1103/PhysRevLett.133.190401.

[46] G. Vidal. Class of quantum many-body states that can be efficiently simulated. *Physical Review Letters*, 101(11), September 2008. ISSN 1079-7114. doi: 10.1103/physrevlett.101.110501. URL http://dx.doi.org/10.1103/PhysRevLett.101.110501.

[47] Alexander Jahn and Jens Eisert. Holographic tensor network models and quantum error correction: A topical review. 2021. doi: 10.48550/ARXIV.2102.02619. URL https://arxiv.org/abs/2102.02619.

[48] Y.-Y. Shi, L.-M. Duan, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Physical Review A*, 74(2), August 2006. ISSN 1094-1622. doi: 10.1103/physreva.74.022320. URL http://dx.doi.org/10.1103/PhysRevA.74.022320.

[49] V. Murg, F. Verstraete, Ö. Legeza, and R. M. Noack. Simulating strongly correlated quantum systems with tree tensor networks. *Physical Review B*, 82(20), November 2010. ISSN 1550-235X. doi: 10.1103/physrevb.82.205105. URL http://dx.doi.org/10.1103/PhysRevB.82.205105.

[50] Naoki Nakatani and Garnet Kin-Lic Chan. Efficient tree tensor network states (ttns) for quantum chemistry: Generalizations of the density matrix renormalization group algorithm. 2013. doi: 10.48550/ARXIV.1302.2298. URL https://arxiv.org/abs/1302.2298.

[51] Weitang Li, Jiajun Ren, Hengrui Yang, Haobin Wang, and Zhigang Shuai. Optimal tree tensor network operators for tensor network simulations: Applications to open quantum systems. *The Journal of Chemical Physics*, 161(5), August 2024. ISSN 1089-7690. doi: 10.1063/5.0218773. URL http://dx.doi.org/10.1063/5.0218773.

[52] V. Murg, F. Verstraete, R. Schneider, P. R. Nagy, and Ö. Legeza. Tree tensor network state with variable tensor order: An efficient multireference method for strongly correlated systems. *Journal of Chemical Theory and Computation*, 11(3):1027–1036, February 2015. ISSN 1549-9626. doi: 10.1021/ct501187j. URL http://dx.doi.org/10.1021/ct501187j.

[53] P. Silvi, V. Giovannetti, S. Montangero, M. Rizzi, J. I. Cirac, and R. Fazio. Homogeneous binary trees as ground states of quantum critical hamiltonians. *Physical Review A*, 81(6), June 2010. ISSN 1094-1622. doi: 10.1103/physreva.81.062335. URL http://dx.doi.org/10.1103/PhysRevA.81.062335.

[54] Alexander Kliesch and Robert König. Continuum limits of homogeneous binary trees and the thompson group. *Physical Review Letters*, 124(1), January 2020. ISSN 1079-7114. doi: 10.1103/physrevlett.124.010601. URL http://dx.doi.org/10.1103/PhysRevLett.124.010601.

[55] Steven R. White. Density-matrix algorithms for quantum renormalization groups. *Physical Review B*, 48(14):10345–10356, October 1993. ISSN 1095-3795. doi: 10.1103/physrevb.48.10345. URL http://dx.doi.org/10.1103/PhysRevB.48.10345.

[56] Gianluca Ceruti, Jonas Kusch, Christian Lubich, and Dominik Sulz. A parallel basis update and galerkin integrator for tree tensor networks, 2024. URL https://arxiv.org/abs/2412.00858.

[57] F. Fröwis, V. Nebendahl, and W. Dür. Tensor operators: Constructions and applications for long-range interaction systems. *Physical Review A*, 81(6), June 2010. ISSN 1094-1622. doi: 10.1103/physreva.81.062337. URL http://dx.doi.org/10.1103/PhysRevA.81.062337.

[58] Jiajun Ren, Weitang Li, Tong Jiang, and Zhigang Shuai. A general automatic method for optimal construction of matrix product operators using bipartite graph theory. *The Journal of Chemical Physics*, 153(8), August 2020. ISSN 1089-7690. doi: 10.1063/5.0018149. URL http://dx.doi.org/10.1063/5.0018149.

[59] C. Hubig, I. P. McCulloch, and U. Schollwöck. Generic construction of efficient matrix product operators. *Physical Review B*, 95(3), January 2017. ISSN 2469-9969. doi: 10.1103/physrevb.95.035129. URL http://dx.doi.org/10.1103/PhysRevB.95.035129.

[60] Garnet Kin-Lic Chan, Anna Keselman, Naoki Nakatani, Zhendong Li, and Steven R. White. Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms. *The Journal of Chemical Physics*, 145(1), July 2016. ISSN 1089-7690. doi: 10.1063/1.4955108. URL http://dx.doi.org/10.1063/1.4955108.

[61] Richard M. Milbradt, Qunsheng Huang, and Christian B. Mendl. State diagrams to determine tree tensor network operators. *SciPost Physics Core*, 7(2), June 2024. ISSN 2666-9366. doi: 10.21468/scipostphyscore.7.2.036. URL http://dx.doi.org/10.21468/SciPostPhysCore.7.2.036.

[62] Hazar undefinedakır, Richard M. Milbradt, and Christian B. Mendl. Optimal symbolic construction of matrix product operators and tree tensor network operators, 2025. URL https://arxiv.org/abs/2502.18630.

[63] Ian P McCulloch. From density-matrix renormalization group to matrix product states. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(10):P10014–P10014, October 2007. ISSN 1742-5468. doi: 10.1088/1742-5468/2007/10/p10014. URL http://dx.doi.org/10.1088/1742-5468/2007/10/P10014.

[64] Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, October 2014. ISSN 0003-4916. doi: 10.1016/j.aop.2014.06.013. URL http://dx.doi.org/10.1016/j.aop.2014.06.013.

[65] Arturo Acuaviva, Visu Makam, Harold Nieuwboer, David Pérez-García, Friedrich Sittner, Michael Walter, and Freek Witteveen. The minimal canonical form of a tensor network. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, page 328–362. IEEE, November 2023. doi: 10.1109/focs57990.2023.00027. URL http://dx.doi.org/10.1109/FOCS57990.2023.00027.

[66] Yifan Zhang and Edgar Solomonik. On stability of tensor networks and canonical forms, 2020. URL https://arxiv.org/abs/2001.01191.

[67] Peng-Fei Zhou, Ying Lu, Jia-Hao Wang, and Shi-Ju Ran. Tensor network efficiently representing schmidt decomposition of quantum many-body states. *Physical Review Letters*, 131(2), July 2023. ISSN 1079-7114. doi: 10.1103/physrevlett.131.020403. URL http://dx.doi.org/10.1103/PhysRevLett.131.020403.

[68] F. Verstraete, J. I. Cirac, and V. Murg. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. 2009. doi: 10.48550/ARXIV.0907.2796. URL https://arxiv.org/abs/0907.2796.

[69] G. Catarina and Bruno Murta. Density-matrix renormalization group: a pedagogical introduction. *The European Physical Journal B*, 96(8), August 2023. ISSN 1434-6036. doi: 10.1140/epjb/s10051-023-00575-2. URL http://dx.doi.org/10.1140/epjb/s10051-023-00575-2.

[70] Glen Evenbly. A practical guide to the numerical implementation of tensor networks i: Contractions, decompositions and gauge freedom, 2022. URL https://arxiv.org/abs/2202.02138.

[71] Mingpu Qin. Combination of tensor network states and green's function monte carlo. 2020. doi: 10.48550/ARXIV.2006.15608. URL https://arxiv.org/abs/2006.15608.

[72] Xuanmin Cao, Qijun Hu, and Fan Zhong. Scaling theory of entanglement entropy in confinements near quantum critical points. *Physical Review B*, 98(24), December 2018. ISSN 2469-9969. doi: 10.1103/physrevb.98.245124. URL http://dx.doi.org/10.1103/physrevb.98.245124.

[73] Javier Lopez-Piqueres, Brayden Ware, and Romain Vasseur. Mean-field entanglement transitions in random tree tensor networks. 2020. doi: 10.48550/ARXIV.2003.01138. URL https://arxiv.org/abs/2003.01138.

[74] Zhi-Cheng Yang, Yaodong Li, Matthew P. A. Fisher, and Xiao Chen. Entanglement phase transitions in random stabilizer tensor networks. 2021. doi: 10.48550/ARXIV.2107.12376. URL https://arxiv.org/abs/2107.12376.

[75] Hanchen Liu, Tianci Zhou, and Xiao Chen. Measurement induced entanglement transition in two dimensional shallow circuit. 2022. doi: 10.48550/ARXIV.2203.07510. URL https://arxiv.org/abs/2203.07510.

[76] Philipp Seitz, Ismael Medina, Esther Cruz, Qunsheng Huang, and Christian B. Mendl. Simulating quantum circuits using tree tensor networks. *Quantum*, 7:964, March 2023. ISSN 2521-327X. doi: 10.22331/q-2023-03-30-964. URL http://dx.doi.org/10.22331/q-2023-03-30-964.

[77] E. Jeckelmann, A. Cojuhovschi, M. Einhellinger, and M. Paech. Investigation of luttinger liquids with dmrg and tebd methods(new development of numerical simulations in low-dimensional quantum systems: From density matrix renormalization group to tensor network formulations). 95:618–618, 2011.

[78] Hai bo Ma, Zhen Luo, and Yao Yao. The time-dependent density matrix renormalisation group method. *Molecular Physics*, 116:854 – 868, 2018. doi: 10.1080/00268976.2017.1406165.

[79] P. E. Dargel, A. Wöllert, A. Honecker, I. P. McCulloch, U. Schollwöck, and T. Pruschke. Lanczos algorithm with matrix product states for dynamical correlation functions. *Physical Review B*, 85(20), May 2012. ISSN 1550-235X. doi: 10.1103/physrevb.85.205119. URL http://dx.doi.org/10.1103/PhysRevB.85.205119.

[80] 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(4):255, October 1950. ISSN 0091-0635. doi: 10.6028/jres.045.026. URL http://dx.doi.org/10.6028/jres.045.026.

[81] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst, editors. *Templates for the solution of algebraic eigenvalue problems*. Society for Industrial and Applied Mathematics, January 2000.

[82] Sebastian Paeckel, Thomas Köhler, Andreas Swoboda, Salvatore R. Manmana, Ulrich Schollwöck, and Claudius Hubig. Time-evolution methods for matrix-product states. *Annals of Physics*, 411:167998, December 2019. ISSN 0003-4916. doi: 10.1016/j.aop.2019.167998. URL http://dx.doi.org/10.1016/j.aop.2019.167998.

[83] Christian Lubich, Ivan V. Oseledets, and Bart Vandereycken. Time integration of tensor trains. *SIAM Journal on Numerical Analysis*, 53(2):917–941, January 2015. ISSN 1095-7170. doi: 10.1137/140976546. URL http://dx.doi.org/10.1137/140976546.

[84] Jutho Haegeman, Christian Lubich, Ivan Oseledets, Bart Vandereycken, and Frank Verstraete. Unifying time evolution and optimization with matrix product states. *Physical Review B*, 94(16), October 2016. ISSN 2469-9969. doi: 10.1103/physrevb.94.165116. URL http://dx.doi.org/10.1103/PhysRevB.94.165116.

[85] Daniel Bauernfeind and Markus Aichhorn. Time dependent variational principle for tree tensor networks. *SciPost Physics*, 8(2), February 2020. ISSN 2542-4653. doi: 10.21468/scipostphys.8.2.024. URL http://dx.doi.org/10.21468/SciPostPhys.8.2.024.

[86] Andreas Gleis, Jheng-Wei Li, and Jan von Delft. Projector formalism for kept and discarded spaces of matrix product states. *Physical Review B*, 106(19), November 2022. ISSN 2469-9969. doi: 10.1103/physrevb.106.195138. URL http://dx.doi.org/10.1103/PhysRevB.106.195138.

[87] Toshiya Hikihara, Hiroshi Ueda, Kouichi Okunishi, Kenji Harada, and Tomotoshi Nishino. Automatic structural optimization of tree tensor networks. 2022. doi: 10.48550/ARXIV.2209.03196. URL https://arxiv.org/abs/2209.03196.

[88] Toshiya Hikihara, Hiroshi Ueda, Kouichi Okunishi, Kenji Harada, and Tomotoshi Nishino. Visualization of entanglement geometry by structural optimization of tree tensor network, 2024. URL https://arxiv.org/abs/2401.16000.

[89] David Bruce Wilson. Generating random spanning trees more quickly than the cover time. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96*, STOC '96, page 296–303. ACM Press, 1996. doi: 10.1145/237814.237880. URL http://dx.doi.org/10.1145/237814.237880.

[90] Steven R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866, November 1992. ISSN 0031-9007. doi: 10.1103/physrevlett.69.2863. URL http://dx.doi.org/10.1103/PhysRevLett.69.2863.

[91] U. Schollwöck. The density-matrix renormalization group. *Reviews of Modern Physics*, 77(1):259–315, April 2005. ISSN 1539-0756. doi: 10.1103/revmodphys.77.259. URL http://dx.doi.org/10.1103/RevModPhys.77.259.

[92] Guifré Vidal. Efficient simulation of one-dimensional quantum many-body systems. *Physical Review Letters*, 93(4), July 2004. ISSN 1079-7114. doi: 10.1103/physrevlett.93.040502. URL http://dx.doi.org/10.1103/PhysRevLett.93.040502.

[93] F. Verstraete, D. Porras, and J. I. Cirac. Density matrix renormalization group and periodic boundary conditions: A quantum information perspective. *Physical Review Letters*, 93(22), November 2004. ISSN 1079-7114. doi: 10.1103/physrevlett.93.227205. URL http://dx.doi.org/10.1103/PhysRevLett.93.227205.

[94] F. Verstraete and J. I. Cirac. Matrix product states represent ground states faithfully. *Physical Review B*, 73(9), March 2006. ISSN 1550-235X. doi: 10.1103/physrevb.73.094423. URL http://dx.doi.org/10.1103/PhysRevB.73.094423.

[95] C. Hubig, I. P. McCulloch, U. Schollwöck, and F. A. Wolf. Strictly single-site dmrg algorithm with subspace expansion. *Physical Review B*, 91(15), April 2015. ISSN 1550-235X. doi: 10.1103/physrevb.91.155115. URL http://dx.doi.org/10.1103/PhysRevB.91.155115.

[96] Richard M. Milbradt, Qunsheng Huang, and Christian B. Mendl. Pytreenet: A python library for easy utilisation of tree tensor networks, 2024. URL https://arxiv.org/abs/2407.13249.

[97] Jutho Haegeman, Tobias J. Osborne, and Frank Verstraete. Post-matrix product state methods: To tangent space and beyond. *Physical Review B*, 88(7), August 2013. ISSN 1550-235X. doi: 10.1103/physrevb.88.075133. URL http://dx.doi.org/10.1103/PhysRevB.88.075133.

[98] Roger B. Sidje. Expokit: a software package for computing matrix exponentials. *ACM Transactions on Mathematical Software*, 24(1):130–156, March 1998. ISSN 1557-7295. doi: 10.1145/285861.285868. URL http://dx.doi.org/10.1145/285861.285868.

[99] Subhasis Ghora, Tarakanta Nayak, Soumen Pal, and Pooja Phogat. Chebyshev's method for exponential maps, 2024. URL https://arxiv.org/abs/2411.11290.

[100] M. Arioli, B. Codenotti, and C. Fassino. The padé method for computing the matrix exponential. *Linear Algebra and its Applications*, 240:111–130, June 1996. ISSN 0024-3795. doi: 10.1016/0024-3795(94)00190-1. URL http://dx.doi.org/10.1016/0024-3795(94)00190-1.

[101] Daniel Manzano. A short introduction to the lindblad master equation. *AIP Advances*, 10(2), February 2020. ISSN 2158-3226. doi: 10.1063/1.5115323. URL http://dx.doi.org/10.1063/1.5115323.

[102] Haggai Landa and Grégoire Misguich. Nonlocal correlations in noisy multiqubit systems simulated using matrix product operators. *SciPost Physics Core*, 6(2), May 2023. ISSN 2666-9366. doi: 10.21468/scipostphyscore.6.2.037. URL http://dx.doi.org/10.21468/SciPostPhysCore.6.2.037.

[103] QuTiP Developers. mesolve. https://qutip.org/docs/4.0.2/modules/qutip/mesolve.html. QuTiP 4.0.2 Documentation.

[104] Klaas Gunst, Frank Verstraete, Sebastian Wouters, Örs Legeza, and Dimitri Van Neck. T3ns: Three-legged tree tensor network states. *Journal of Chemical Theory and Computation*, 14(4):2026–2033, February 2018. ISSN 1549-9626. doi: 10.1021/acs.jctc.8b00098. URL http://dx.doi.org/10.1021/acs.jctc.8b00098.

[105] Morag Am-Shallem, Amikam Levy, Ido Schaefer, and Ronnie Kosloff. Three approaches for representing lindblad dynamics by a matrix-vector notation, 2015. URL https://arxiv.org/abs/1510.08634.

[106] Christina V. Kraus and Tobias J. Osborne. Time-dependent variational principle for dissipative dynamics. *Physical Review A*, 86(6), December 2012. ISSN 1094-1622. doi: 10.1103/physreva.86.062115. URL http://dx.doi.org/10.1103/PhysRevA.86.062115.