Technical University of Munich
School of Natural Sciences

Bachelorarbeit

Effiziente Matrixproduktoperator basierte Zeitentwicklung

Bachelor's Thesis

Fast Matrix Product Operator Based Time Evolution

Author: Martin Hefel

Supervisors: M.Sc. Jakob Unfried
Prof. Dr. Frank Pollmann
Theoretical Solid-State Physics

Second Assessor: Prof. Dr. Michael Knap
Collective Quantum Dynamics

Submission Date: December 18, 2023

Kurzfassung

Wir stellen einen auf dem Matrixproduktoperator (MPO) basierenden Zeitentwicklungsalgorithmus vor, der anstelle einer Singulärwertzerlegung (SVD) ein auf der QR-Zerlegung basierende Variationsmethode verwendet. Wir zeigen, dass sich dadurch die Skalierung des Zeitentwicklungsalgorithmus von d^3 auf d^2 verbessert, wobei d die Dimension des lokale Hilbertraums ist. Darüber hinaus, läuft der vorgeschlagene Algorithmus, im Gegensatz zu der etablierten SVD basierten Trunkierungsmethode, effizient auf GPUs. Dies führt zu einem zusätzlichen, hardwareabhängigen Speedup.

Abstract

We propose and benchmark a matrix product operator (MPO) based time evolution which uses a variational QR decomposition based truncation scheme instead of a singular value decomposition (SVD). This improves the scaling of the time evolution algorithm with respect to the local Hilbert space dimension d from d^3 to d^2. Additionally, we demonstrate that the proposed algorithm runs efficiently on GPUs in contrast to the established SVD based truncation method. This results in an additional, hardware-dependent speedup.

Table of Contents

1 Introduction . . . 1

2 Theoretical Background . . . 2
2.1 Quantum Many-Body Systems . . . 2
2.2 Matrix Decomposition . . . 3
2.2.1 Singular Value Decomposition (SVD) . . . 3
2.2.2 QR Decomposition . . . 4
2.3 Entanglement in Quantum Many-Body Systems . . . 4
2.3.1 Density Matrix . . . 4
2.3.2 Schmidt Decomposition . . . 5
2.3.3 Entanglement Entropy . . . 5
2.3.4 Area Law . . . 6
2.4 Finite Systems in One Dimension . . . 7
2.4.1 Tensor Network Notation . . . 7
2.4.2 Matrix Product States (MPS) . . . 9
2.4.3 Canonical Form . . . 9
2.4.4 Compression of a MPS . . . 11
2.4.5 Matrix Product Operators (MPO) . . . 11

3 MPO Based Time-Evolution . . . 12
3.1 Time Evolution Algorithm . . . 12
3.2 Truncation Schemes . . . 16
3.2.1 SVD Based Method . . . 16
3.2.2 QR Decomposition Based Method . . . 16
3.2.3 QR Decomposition Based Method with CBE . . . 18
3.3 Computation Cost . . . 19

4 Benchmark . . . 20
4.1 Model Implementation . . . 20
4.1.1 Quantum Clock Model . . . 20
4.1.2 Number of Sweeps . . . 22
4.2 Result Benchmark . . . 23
4.3 Timing Benchmark . . . 24

5 Conclusion . . . 27

List of Figures . . . 28

Abbreviations . . . 29

Terms . . . 30

References . . . 33

Appendix . . . 34
A Optaining the MPS of a generic pure state . . . I
B Information about the developed Python library . . . II

1. Introduction

Simulating quantum many-body systems in and out of equilibrium is essential for understanding the physics of microscopic systems. The success of the density matrix renormalization group (DMRG) [1] in finding the ground state properties of one-dimensional systems in terms of matrix product states (MPS) led to the development of related techniques to describe the dynamics of quantum many-body systems [2–4]. These methods allow access to experimentally relevant observables, such as structure factors or other response functions, which can be compared, among others, with data from neutron scattering [5]. By now, the MPS formalism is an established tool to simulate one-dimensional systems in condensed matter theory.

It has been shown [6–10] that there is potential to accelerate MPS based algorithms by running linear algebra operations on graphics processing units (GPUs) or tensor processing units (TPUs). However, many MPS based algorithms rely on the singular value decomposition (SVD), which does not run efficiently in the GPU implementations known to us. A modified version of the time evolving block decimation (TEBD) algorithm [11] showed that using a truncation scheme based on the QR decomposition instead of the SVD runs efficiently on GPUs and additionally improves the scaling of the algorithm with respect to the local Hilbert space dimension d from d^3 to d^2. While the TEBD [2] is highly successful in simulating time evolutions of nearest neighbor interacting Hamiltonians, it is not applicable to long-ranged Hamiltonians. In contrast to that, the matrix product operator (MPO) based time evolution allows such time evolutions by successively applying the time evolution MPO to the MPS. Yet, time evolution MPOs were long impractical since naive time-steppers, like an Euler step, resulted in an error per site which diverges with the system size. However, it recently has been demonstrated [12] that a modified Euler step allows compact MPO representations with a constant error per site and hence practical MPO based time evolutions.

In this work, we expand the success of the QR decomposition based truncation method for the TEBD [11] to the MPO based time evolution [12] for one-dimensional systems. This is achieved analogously to Ref. [11] by replacing the SVD based truncation method of the MPO based time evolution algorithm with a variational QR decomposition based truncation scheme. This improves the scaling of the time evolution algorithm from d^3 to d^2 and leads to a significant hardware-dependent speedup when computed on GPUs.

We start in chapter 2 with the theoretical background, reaching from the definition of the SVD and QR decomposition up to the MPS and MPO formalism. In chapter 3, we introduce the MPO based time evolution algorithm and implement the QR decomposition based truncation method. In chapter 4, we benchmark the result of a global quench in the quantum clock model, followed by a runtime benchmark of the time evolution algorithm and the truncation schemes on CPU and GPU. Finally, we conclude our findings in chapter 5.

2. Theoretical Background

In this chapter, we briefly introduce the theoretical background that is used to perform a MPO based time evolution. We start by facing the computational challenges of simulating time evolutions of generic quantum many-body states. Followed by the introduction of the SVD and the QR decomposition, which play a central role in this work. Equipped with these tools, we proceed with the phenomenon of entanglement in quantum many-body systems. This leads to the definition of the area law, which forms the basis of the efficiency of the subsequently introduced MPS and MPO formalism.

2.1. Quantum Many-Body Systems

A generic state $|\psi\rangle$ of a quantum many-body system on a lattice with N sites lives in the Hilbert space $\mathcal{H}$, which is formed by the tensor product of the local Hilbert spaces $\mathcal{H}_n$, meaning $\mathcal{H} = \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_N$. The dimension d_n of the local Hilbert space $\mathcal{H}_n$ is defined by the number of local basis states $|j_n\rangle$. Assuming that each local Hilbert space $\mathcal{H}_\mathrm{n}$ has the same dimension $d_\mathrm{n} = d$, we can determine the dimension of the many-body Hilbert space $\mathcal{H}$ to d^N. The wavefunction of a generic state can be written as

$$|\psi\rangle = \sum_{j_1, j_2, \ldots, j_N} \psi_{j_1 j_2 \ldots j_N} |j_1, j_2, \ldots, j_N\rangle \tag{2.1}$$

where $|j_1, j_2, \ldots, j_N\rangle$ represents a basis state of the many-body Hilbert space and $\psi_{j_1 j_2 \ldots j_N}$ the probability amplitude with the normalization condition $\sum_{j_1, j_2, \ldots, j_N} |\psi_{j_1 j_2 \ldots j_N}|^2 = 1$. The time evolution $|\psi(t)\rangle$ of a such generic state $|\psi(t_0)\rangle$ needs to comply with the Schrödinger equation

$$\mathrm{i}\hbar \frac{\partial}{\partial t} |\psi(t)\rangle = H |\psi(t)\rangle \tag{2.2}$$

where H is the Hamiltonian of the System. If we consider H to be time-independent

$$|\psi(t)\rangle = U(t, t_0) |\psi(t_0)\rangle \qquad \text{with} \qquad U(t, t_0) = e^{-\mathrm{i}(t - t_0)H/\hbar} \tag{2.3}$$

complies with equation (2.2) where $U(t, t_0)$ is the time evolution operator. The time evolution operator U is unitary, which follows from the probability conservation. Moreover, U needs to fulfill the composition property $U(t_2, t_0) = U(t_2, t_1)U(t_1, t_0)$ for $t_2 > t_1 > t_0$ and it must reduce to the identity matrix as $\mathrm{d}t$ goes to zero, meaning $\lim_{\mathrm{d}t \to 0} U(t_0 + \mathrm{d}t, t_0) = \mathbb{1}$, where $\mathbb{1}$ is the identity matrix. (see [13] sec. 3 and [14] sec. 2.1.1, sec. 2.1.2)

Considering the scaling d^N of the dimension of the many-body Hilbert space $\mathcal{H}$, we can calculate the storage space needed to save all coefficients $\psi_{j_1 j_2 \ldots j_N}$ of a generic state $|\psi\rangle$ to $\sim d^N$ bytes. Computing the inner product of two generic states is equivalent to performing $\sim d^N$ double-precision float operations. As a reference, current supercomputers [15] can

handle up to $\sim 10^{17}$ double-precision float operations per second. While this sounds like a lot, it is nothing compared to the computational power needed to handle realistic systems with $N \sim 10^{23}$ particles. Consequently, it is impossible to simulate large systems by using pure brute force. However, we have assumed that the states actually live in the d^N Hilbert space without any limitations. In practice, this is not the case since the vast majority of the Hilbert space is unreachable for large systems [16]. If we focus only on a certain region of the Hilbert space, where the states obey the so-called area law, the MPS formalism allows us to efficiently represent and simulate time evolutions of quantum many-body systems.

2.2. Matrix Decomposition

2.2.1. Singular Value Decomposition (SVD)

Let $M \in \mathbb{C}^{m \times n}$ then M can be written in the SVD

$$M = USV^\dagger \tag{2.4}$$

where U and V are $m \times m$ and $n \times n$ unitary matrix, where the columns of U and V are eigenvectors of the matrices $MM^\dagger$ and $M^\dagger M$. S is a rectangular diagonal $m \times n$ Matrix with non-negative entries, the so-called singular values of M. The number of non-zero singular values is the rank k of the Matrix M. Note that if M has full rank, then $k = \min\{m, n\}$. The SVD can be written in a reduced form where U and V are semi-unitary $m \times k$ and $n \times k$ matrices U_{red} and V_{red} and S is a $k \times k$ square diagonal matrix S_{red}. This reduced form is achieved by removing all zero-row- and zero-column-vectors of S, including the ones where the singular values are zero, together with the respective columns of U and V. These removed columns and vectors do not contribute to the representation of M. Hence there is no "loss of information", meaning $||M - M_{\text{red}}|| = 0$. Moving forward, we assume that the SVD is expressed in this reduced form. Moreover, we assume that the singular values and the corresponding columns of U and V are sorted in descending order. We can write eq. (2.4) as

$$M \approx \sum_{i=1}^{\eta \leq k} u_i S_i v_i^\dagger \tag{2.5}$$

where u_i and v_i are the orthonormal column vectors of U and V. The matrix M is represented exactly for $\eta = k$. However, to save memory when storing the matrix M, we can instead save the matrices U, V and S with $\eta < k$. This means we discard some orthonormal column vectors u_i and v_i, which is equivalent to truncating the rows of U and V. This results in a truncation error, which can be calculated to $\sum_{i=\eta+1}^{k} S_i$. Hence, the truncation error is dependent on the nature of the singular values S_i and thus M. The fact that this is the optimal truncation of the matrix M not only makes the SVD a good tool for image compressions [17], but also allows us to represent some quantum many-body states more efficiently, as we will see in sec. 2.3.2.

As it will be important later, a remark on the uniqueness of the SVD: The singular values S_i of S are uniquely defined by M. Hence, S is uniquely defined by M. However, U and V are not uniquely defined by M. First, if there are multiple equal singular values, meaning S is degenerate, the corresponding columns of U and V can be swapped while they still yield the SVD. Secondly, any unitary diagonal matrix ϕ can be supplemented into eq. (2.4) to retrieve a new set of semi-unitary matrices $\tilde{U}$ and $\tilde{V}$:

$$USV^\dagger = U\phi\phi^\dagger SV^\dagger = U\phi S\phi^\dagger V^\dagger = U\phi S(V\phi)^\dagger = \tilde{U}S\tilde{V}^\dagger \tag{2.6}$$

where we used the very definition of unitary matrices and that two diagonal matrices commute. These gauge transformations ϕ are of the form

$$\phi = \begin{pmatrix} e^{\mathrm{i}\varphi_1} & & & \\ & e^{\mathrm{i}\varphi_2} & & \\ & & \ddots & \\ & & & e^{\mathrm{i}\varphi_k} \end{pmatrix} \quad \text{with} \quad \varphi_i \in [0, 2\pi). \tag{2.7}$$

(see [18] thm. 2.26, [19] thm. 7.46 and [20] sec. 4.1.1)

2.2.2. QR Decomposition

Let $M \in \mathbb{C}^{m\times n}$ with $m \geq n$ then M can be written in the QR decomposition

$$M = QR \tag{2.8}$$

where Q is a $m \times n$ semi-unitary matrix and R is a upper-triangular $n \times n$ matrix. Q is unitary if $m = n$. For $m < n$ we define the QR decomposition as in NumPy [21], thus Q is a $m \times m$ unitary matrix and R is a $m \times n$ matrix with $R_{ij} = 0$ if $m \geq i > j$. For completeness, we mention that we define the LQ decomposition $M = LQ$ analogously to the QR decomposition, where L is a lower-triangular matrix. (see [18] thm. 2.17 and [20] sec. 4.1.2)

2.3. Entanglement in Quantum Many-Body Systems

2.3.1. Density Matrix

The density matrix can be used to define mixed and pure ensembles (or states), which is a property needed in the discussion of entanglement. The density operator ρ of an ensemble of states $|\psi_k\rangle$ is defined by

$$\rho = \sum_k p_k \, |\psi_k\rangle \, \langle\psi_k| \tag{2.9}$$

where p_k is the probability weight of the state $|\psi_k\rangle$ with the normalisation condition $\sum_k p_k = 1$. The density matrix is obtained from the density operator by choosing a base in the respective Hilbert space. The density matrix ρ is hermitian and satisfies the normalization condition $\mathrm{tr}(\rho) = 1$. An ensemble (or state) is called *pure* when there is only one probability weight

$p_n = 1$ for one state $|\psi_n\rangle$ and $p_k = 0$ for all other states $|\psi_k\rangle$ with $k \neq n$. Consequently the density matrix can be written as $\rho = |\psi_n\rangle \langle\psi_n|$ and thus $\rho^2 = \rho$ and $\mathrm{tr}\left(\rho^2\right) = 1$. Otherwise, the ensemble (or state) is called *mixed*. (see [14] sec. 3.4.2)

2.3.2. Schmidt Decomposition

Let us consider the bipartition of the m dimensional Hilbert space $\mathcal{H}$ in two subsystems A and B with their $m_{\mathrm{A(B)}}$ dimensional Hilbert space $\mathcal{H}_{\mathrm{A(B)}}$. The tensor product of the Hilbert spaces $\mathcal{H}_{\mathrm{A}}$ and $\mathcal{H}_{\mathrm{B}}$ composes the Hilbert space $\mathcal{H}$, meaning

$$\mathcal{H} = \mathcal{H}_{\mathrm{A}} \otimes \mathcal{H}_{\mathrm{B}} \tag{2.10}$$

with $m = m_{\mathrm{A}} m_{\mathrm{B}}$. Let $\{|i\rangle_{\mathrm{A}}\}$ and $\{|j\rangle_{\mathrm{B}}\}$ be orthonormal bases of the respective Hilbert spaces $\mathcal{H}_{\mathrm{A}}$ and $\mathcal{H}_{\mathrm{B}}$. Hence $\{|i\rangle_{\mathrm{A}} \otimes |j\rangle_{\mathrm{B}}\}$ is an orthonormal base of the Hilbert space $\mathcal{H}$ and any pure state $|\psi\rangle \in \mathcal{H}$ can be written as

$$|\psi\rangle = \sum_{i,j} a_{ij} \, |i\rangle_{\mathrm{A}} \otimes |j\rangle_{\mathrm{B}} = \sum_{i,j} a_{ij} \, |i_{\mathrm{A}} j_{\mathrm{B}}\rangle \tag{2.11}$$

where $a_{ij} \in \mathbb{C}$ is an amplitude which satisfies the normalisation condition $\sum_{i,j} |a_{ij}|^2 = 1$ if the state $|\psi\rangle$ is normalized. To represent the state $|\psi\rangle$ in a more convenient way we write all amplitudes a_{ij} in a $m_{\mathrm{A}} \times m_{\mathrm{B}}$ matrix ψ, with $a_{ij} = \psi_{ij}$, perform a SVD of ψ and obtain the matrices U, $V^\dagger$ and Λ (S in sec. 2.2.1). This allows us to write a_{ij} as $\sum_{\alpha=1}^{k} U_{i\alpha} \Lambda_\alpha V^\dagger_{\alpha j}$, where k is the rank of the matrix ψ. Hence we can write eq. (2.11) as

$$|\psi\rangle = \sum_{i,j} \sum_{\alpha=1}^{k} U_{i\alpha} \Lambda_\alpha V^\dagger_{\alpha j} \, |i\rangle_{\mathrm{A}} \otimes |j\rangle_{\mathrm{B}} = \sum_{\alpha=1}^{k} \Lambda_\alpha \sum_i U_{i\alpha} \, |i\rangle_{\mathrm{A}} \otimes \sum_j V^\dagger_{\alpha j} \, |j\rangle_{\mathrm{B}} = \sum_{\alpha=1}^{k} \Lambda_\alpha \, |\alpha\rangle_{\mathrm{A}} \otimes |\alpha\rangle_{\mathrm{B}} \tag{2.12}$$

where we used U and $V^\dagger$ to map $\{|i\rangle_{\mathrm{A}}\}$ and $\{|j\rangle_{\mathrm{B}}\}$ into a new base $\{|\alpha\rangle_{\mathrm{A}}\}$ and $\{|\alpha\rangle_{\mathrm{B}}\}$. This decomposition is called the *Schmidt decomposition*. The orthonormal base kets $|\alpha\rangle_{\mathrm{A(B)}} \in \mathcal{H}_{A(B)}$ are also called Schmidt states and the unique amplitude $\Lambda_\alpha \in \mathbb{R}_{\geq 0}$ is also called Schmidt coefficient or Schmidt value, which satisfies the normalization condition $\sum_\alpha \Lambda_\alpha^2 = 1$ if the state $|\psi\rangle$ is normalized. As with the SVD, the Schmidt base is unique up to degeneracies and a gauge transformation ϕ (see eq. (2.7)). Note that the Schmidt decomposition has a maximum of $k = \min(m_{\mathrm{A}}, m_{\mathrm{B}})$ terms since there exist only $m_{\mathrm{A(B)}}$ orthonormal base kets $|\alpha\rangle_{\mathrm{A(B)}}$ which span the Hilbert space $\mathcal{H}_{A(B)}$. As we will see in the next section, the Schmidt decomposition gives us direct insight into the bipartite entanglement of the state. (see [22] sec. 10.2.1, lem. 10.1.2., [20] sec. 4.1.1 and [13] sec. 2)

2.3.3. Entanglement Entropy

The *entanglement entropy* of the bipartite system in sec. 2.3.2 is defined as the *von Neuman entropy* of the reduced density matrix ρ_{A} or ρ_{B}

$$S_{\mathrm{vN}} = S_{\mathrm{vN}}(\rho_{\mathrm{B}}) = S_{\mathrm{vN}}(\rho_{\mathrm{A}}) = -\mathrm{tr}(\rho_{\mathrm{A}} \log \rho_{\mathrm{A}}) \tag{2.13}$$

where the reduced density matrix ρ_{A} of a pure state is defined as the partial trace of the density matrix $\rho = |\psi\rangle\langle\psi|$, thus $\rho_{\mathrm{A}} = \mathrm{tr}_{\mathrm{B}}(|\psi\rangle\langle\psi|)$. The eigenvalues and eigenstates of ρ_{A} are the squared Schmidt values Λ_α and Schmidt states $|\alpha\rangle_{\mathrm{A}}$ of the schmidt decomposition of $|\psi\rangle$ (see eq. (2.12)). Meaning the reduced density matrix can be written in the spectral-decomposition $\rho_{\mathrm{A}} = \sum_\alpha \Lambda_\alpha^2 |\alpha\rangle_{\mathrm{A}} \langle\alpha|_{\mathrm{A}}$ (equivalent for ρ_{B}). Consequently eq. (2.13) can be expressed as

$$S_{\mathrm{vN}} = -\sum_\alpha \Lambda_\alpha^2 \log \Lambda_\alpha^2. \tag{2.14}$$

The larger the entanglement entropy S_{vN} is, the more entangled the subsystem A and B. If there is no entanglement between subsystem A and B, the entanglement entropy S_{vN} is zero, meaning there is only one Schmidt value with $\Lambda_1 = 1$, while all others are zero. In other words, a pure state $|\psi\rangle \in \mathcal{H} = \mathcal{H}_{\mathrm{A}} \otimes \mathcal{H}_{\mathrm{B}}$ is not entangled if it can be written as a product state $|\psi\rangle = |\phi\rangle_{\mathrm{A}} \otimes |\phi\rangle_{\mathrm{B}}$, where $|\phi\rangle_{\mathrm{A(B)}} \in \mathcal{H}_{\mathrm{A(B)}}$. Moreover, if subsystem A and B are entangled, the reduced density matrix $\rho_{\mathrm{A(B)}}$ represents a mixed state on the respective subsystem $\mathrm{A(B)}$. (see [22] sec. 7.1, sec. 10.3.1 and [13] sec. 2)

2.3.4. Area Law

Let us consider a one-dimensional quantum chain with N sites and d local basis states, which is "cut" (bipartite) in half in two subsystems L and R, where $\mathrm{L(R)}$ represents the left (right) part of the system. The entanglement entropy of a randomly chosen state $|\psi_{\mathrm{rand}}\rangle \in \mathcal{H} = \mathcal{H}_{\mathrm{L}} \otimes \mathcal{H}_{\mathrm{R}}$ is given by $S_{\mathrm{vN}} \approx \frac{N}{2}\log d - \frac{1}{2}$ [23, 24]. This means that the entanglement entropy grows proportional to the size $N/2$ of the subsystem. More generally speaking, it grows proportional to the volume of the of the smaller subsystem. This property is called *volume law*. In contrast to that, the entanglement entropy of a ground state $|\psi_0\rangle \in \mathcal{H} = \mathcal{H}_{\mathrm{L}} \otimes \mathcal{H}_{\mathrm{R}}$ of a gapped and local Hamiltonian H is independent of the subsystem size, hence constant [25, 26]. More generally speaking, it grows proportional to the area of the cut, which is zero for a one-dimensional system. This property is called *area law*. Even though the area law states form only a small manifold of the whole Hilbert space $\mathcal{H}$ they contain the low-energy states of gapped and local Hamiltonians. If a state $|\psi\rangle \in \mathcal{H} = \mathcal{H}_{\mathrm{L}} \otimes \mathcal{H}_{\mathrm{R}}$ is an area law state, only a small amount of Schmidt values Λ_α significantly contribute to the Schmidt decomposition. This allows us to efficiently compress quantum states by truncating the Schmidt decomposition. In this context, efficiently means that we find a finite η, which is independent of the system size, for all $\varepsilon > 0$ such that

$$|| \, |\psi\rangle - |\psi_{\mathrm{trunc}}\rangle \, || = \left|\left| |\psi\rangle - \sum_{\alpha=1}^{\eta} \Lambda_\alpha |\alpha\rangle_{\mathrm{A}} \otimes |\alpha\rangle_{\mathrm{B}} \right|\right| = \sqrt{\sum_{\alpha=\eta+1}^{k} \Lambda_\alpha^2} < \varepsilon. \tag{2.15}$$

This property shows that we do not need to consider the whole Hilbert space to describe area law states and allows us to efficiently describe area law states with MPS, as we will see in the next section. (see [13] sec. 2.1 and [27] sec. 3.4)

2.4. Finite Systems in One Dimension

In this work, we limit ourselves to one-dimensional, finite systems. Hence, moving forward, we consider a one-dimensional quantum chain with N sites. The local Hilbert space of site n is spanned by d local basis states $|j_n\rangle$. In the following, we will give an overview of the diagrammatic notation of tensor networks followed by the introduction of the MPS and MPO formalism, which is mainly based on Ref. [13] and [20].

2.4.1. Tensor Network Notation

While the diagrammatic notation of tensor networks is widely used in the field, there are some minor differences when it comes to the notation and visual representation of tensors. We represent a tensor $T_{\gamma_1\gamma_2\ldots\gamma_N}$ with N indicies γ_n by a symbol with N legs, like in Fig. 1(a). A tensor with two indices is a matrix M and a tensor with one index is a vector v. Moreover, a tensor with zero indices represents a scalar, which is, for example, the result of a full contraction of two tensors. Connecting two tensors means contracting the corresponding indices (legs) by summing over them, thus performing a generalization of the matrix multiplication. For example the contraction of two legs γ_1 and β_2 of two tensors $T^{[1]}_{\gamma_1\gamma_2}$ and $T^{[2]}_{\beta_1\beta_2\beta_3}$, as illustrated in Fig. 1(b), is given by

$$\sum_k T^{[1]}_{k\gamma_2} T^{[2]}_{\beta_1 k\beta_3} = \theta_{\gamma_2\beta_1\beta_3} \tag{2.16}$$

where $\theta_{\gamma_2\beta_1\beta_3}$ is the contracted tensor. Later, we will use "physical" and "virtual" indices where we write the former as superscript and the latter as subscript. Virtual indices are indices belonging to matrices. For example, if we contract multiple tensors of the form $T^{[n]\gamma_n}_{\alpha_n\alpha_{n+1}}$, which are collections of $\dim(\gamma_\mathrm{n})$ matrices, in a row as shown in Fig. 1(c), we express the contraction as

$$\sum_{\alpha_2,\alpha_3} T^{[1]\gamma_1}_{\alpha_1\alpha_2} T^{[2]\gamma_2}_{\alpha_2\alpha_3} T^{[3]\gamma_3}_{\alpha_3\alpha_4} = T^{[1]\gamma_1} T^{[2]\gamma_2} T^{[3]\gamma_3} = \theta^{\gamma_1\gamma_2\gamma_3}_{\alpha_1\alpha_4} \tag{2.17}$$

where we recognized the sums over α_2 and α_3 as matrix multiplication and remove the virtual indices from the tensors $T^{[n]\gamma_n}_{\alpha_n\alpha_{n+1}}$. Another tensor operation is the reshaping of tensors, where we change the way how we organize the variables in tensors to use matrix decompositions, such as the SVD or QR decomposition, on multi-leg tensors. A simple example is reshaping a $d_1 \times d_2 \times d_3$ tensor $T^{\gamma_1}_{\alpha_1\alpha_2}$ into a $(d_1 d_2) \times d_3$ tensor $T_{(\gamma_1\alpha_1)\alpha_2}$. In the diagrammatic notation, we visualize a reshaped tensor by merging the respective legs of the tensor, as shown in Fig. 1(d). Although there are many more details to tensor networks (see [28] sec. 2., sec. 3. and [29] sec. 5.2), we will stop here and develop the necessary properties and notations on the fly.

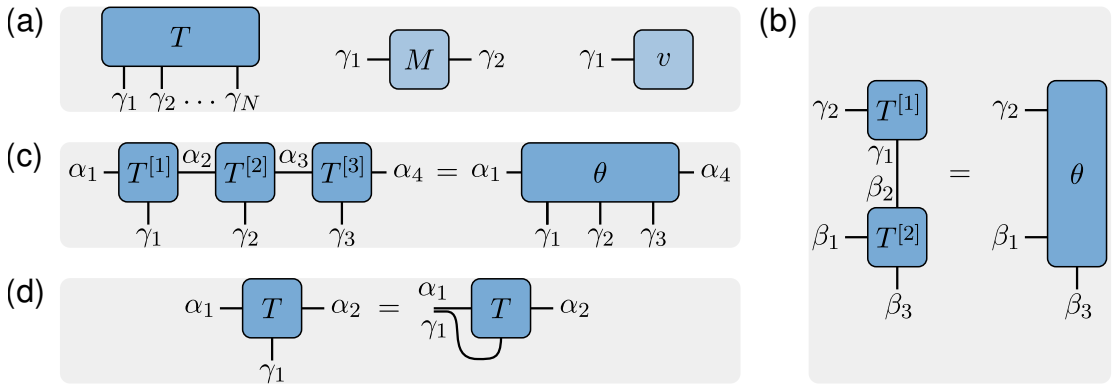


Figure 1: (a) Diagrametic notation of a tensor $T_{\gamma_1\gamma_2\ldots\gamma_N}$ with N indices (legs) γ_n, a matrix M and a vector v. (b) Contraction of two tensors $T^{[1]}_{\gamma_1\gamma_2}$ and $T^{[2]}_{\beta_1\beta_2\beta_3}$, yielding the contracted tensor $\theta_{\gamma_2\beta_1\beta_3}$ as written in eq. (2.16). (c) Contraction of three tensors of the same form $T^{[n]\gamma_n}_{\alpha_n\alpha_{n+1}}$ yielding the contracted tensor $\theta^{\gamma_1\gamma_2\gamma_3}_{\alpha_1\alpha_4}$, which can be written as a product of matrices as in eq. (2.17). (d) Reshaping a three-leg tensor $T^{\gamma_1}_{\alpha_1\alpha_2}$ into a two-leg tensor $T_{(\gamma_1\alpha_1)\alpha_2}$.

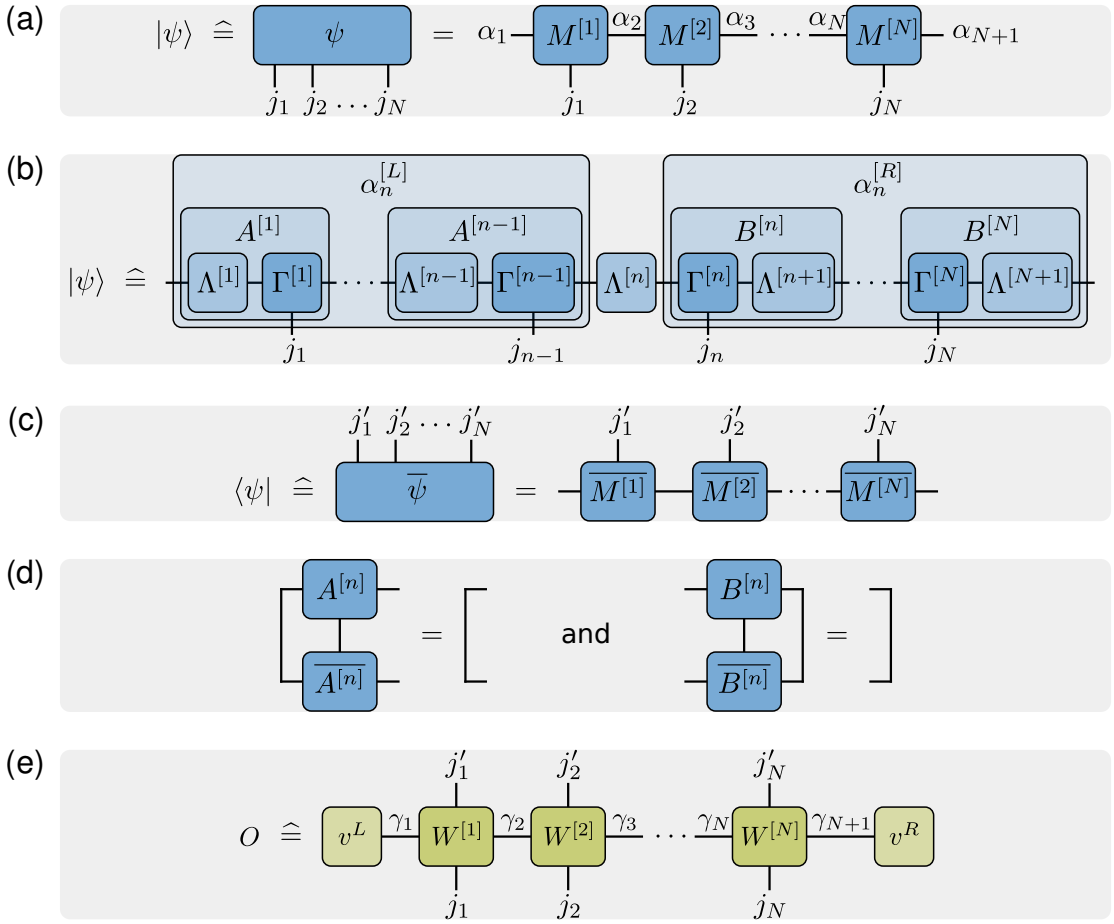


Figure 2: (a) Diagrammatic notation of a state $|\psi\rangle$ as MPS as in eq. (2.18) (b) A state $|\psi\rangle$ in the canonical form as in eq. (2.25). Also shown is the relation to the mixed canonical form with eq. (2.26) and the Schmidt decomposition of the bipartite system $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$, where $\alpha_n^{[L(R)]}$ is the MPS of the state $|\alpha_n\rangle_{L(R)}$. (c) Diagrammatic notation of the hermitian conjugate of a state $|\psi\rangle$: $\langle\psi| = |\psi\rangle^\dagger$. (d) Orthogonality condition of the left and right semi-unitary form A and B as in eq. (2.27). (e) Diagrammatic notation of an operator O as MPO as in eq. (2.28).

2.4.2. Matrix Product States (MPS)

A generic pure state $|\psi\rangle$ (see eq. (2.1)) can be written as a MPS [30–32]

$$\begin{aligned} |\psi\rangle &= \sum_{j_1,j_2,\ldots,j_N} \sum_{\alpha_2,\ldots,\alpha_N} M^{[1]j_1}_{\alpha_1\alpha_2} M^{[2]j_2}_{\alpha_2\alpha_3} \ldots M^{[N]j_N}_{\alpha_N\alpha_{N+1}} |j_1, j_2, \ldots, j_N\rangle \\ &= \sum_{j_1,j_2,\ldots,j_N} M^{[1]j_1} M^{[2]j_2} \ldots M^{[N]j_N} |j_1, j_2, \ldots, j_N\rangle \end{aligned} \tag{2.18}$$

where $M^{[n]j_n}_{\alpha_n\alpha_{n+1}}$ is a tensor with the virtual indices α_n and α_{n+1} and the physical index j_n. To ensure that the product of all $\chi_n \times \chi_{n+1}$ matrices $M^{[n]j_n}$ forms a scalar (to be precise, a 1×1 matrix), namely $\psi_{j_1 j_2 \ldots j_N}$, the bond dimension χ of the boundary legs α_1 and α_{N+1} is one, hence $\chi_1 = \chi_{N+1} = 1$. This means the first and last matrices are essentially vectors. However, defining them as matrices leads to a consistent layout of the MPS. Every site n has d local basis states $|j_n\rangle, j_n \in \{1, 2, \ldots, d\}$, which result, for a generic state $|\psi\rangle$, in d different matrices $M^{[n]j_n}$. The collection of all d matrices $M^{[n]j_n}$ is represented by the already introduced tensor $M^{[n]j_n}_{\alpha_n\alpha_{n+1}}$. This allows us to represent $|\psi\rangle$ as a tensor $\psi^{j_1 j_2 \ldots j_N}$ in MPS form, as shown in Fig. 2(a). In general, any pure state can be represented exactly by a MPS. However, it is evident from app. A, which shows how to obtain the MPS of any pure state $|\psi\rangle$, that the bond dimension χ grows exponentially with each site up to $\chi \sim d^{N/2}$ in the middle of the chain. Nevertheless, the canonical form introduced in the next section links the MPS with the Schmidt decomposition and allows us to efficiently compress the MPS of area law states. ([20] sec. 4.1.4 and [13] sec. 3.1)

2.4.3. Canonical Form

The MPS representation in eq. (2.18) is not unique, for example the MPS is invariant under the gauge transformation

$$M^{[n]j_n} \to \tilde{M}^{[n]j_n} := M^{[n]j_n} X^{-1} \quad , \quad M^{[n+1]j_{n+1}} \to \tilde{M}^{[n+1]j_{n+1}} := X M^{[n+1]j_{n+1}} \tag{2.19}$$

where X is a invertible $\chi_{n+1} \times \chi_{n+1}$ matrix. The gauge may be chosen such that the MPS is in a more convenient manner, in the so-called "canonical form". The left canonical form of a MPS can be obtained by proceeding as instructed by Ref. [20] sec. 4.4. We start by reshaping the tensor $M^{[n]j_n}_{\alpha_n\alpha_{n+1}}$ to $M^{[n]}_{(j_n\alpha_n)\alpha_{n+1}}$ and perform a SVD

$$M^{[n]}_{(j_n\alpha_n)\alpha_{n+1}} = \sum_{\beta_{n+1}} A^{[n]}_{(j_n\alpha_n)\beta_{n+1}} X_{\beta_{n+1}\alpha_{n+1}} \quad \text{with} \quad X_{\beta_{n+1}\alpha_{n+1}} = \Lambda^{[n+1]}_{\beta_{n+1}\beta_{n+1}} (V^\dagger)_{\beta_{n+1}\alpha_{n+1}} \tag{2.20}$$

where $A^{[n]}_{(j_n\alpha_n)\beta_{n+1}}$ and $(V^\dagger)_{\beta_{n+1}\alpha_{n+1}}$ are semi-unitary matrices and $\Lambda^{[n+1]}_{\beta_{n+1}\beta_{n+1}}$ is a diagonal matrix containing the singular values of $M^{[n]}_{(j_n\alpha_n)\alpha_{n+1}}$. We reshape the matrix $A^{[n]}_{(j_n\alpha_n)\beta_{n+1}}$ to a tensor $A^{[n]j_n}_{\alpha_n\beta_{n+1}}$ and treat $X_{\beta_{n+1}\alpha_{n+1}}$ as a transformation matrix for $M^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$ to obtain

$\tilde{M}^{[n+1]j_{n+1}}_{\beta_{n+1}\alpha_{n+2}}$. We use eq. (2.20) to replace $M^{[1]j_1}_{\alpha_1\alpha_2}$ in eq. (2.18):

$$\begin{aligned}|\psi\rangle &= \sum_{j_1,j_2\ldots,j_N}\sum_{\alpha_2,\ldots,\alpha_N} M^{[1]}_{(j_1\alpha_1)\alpha_2} M^{[2]j_2}_{\alpha_2\alpha_3}\ldots M^{[N]j_N}_{\alpha_N\alpha_{N+1}}\,|j_1,j_2,\ldots,j_N\rangle \\ &= \sum_{j_1,j_2,\ldots,j_N}\sum_{\alpha_2,\ldots,\alpha_N}\sum_{\beta_2} A^{[1]}_{(j_1\alpha_1)\beta_2} X_{\beta_2\alpha_2} M^{[2]j_2}_{\alpha_2\alpha_3}\ldots M^{[N]j_N}_{\alpha_N\alpha_{N+1}}\,|j_1,j_2,\ldots,j_N\rangle \\ &= \sum_{j_1,j_2,\ldots,j_N}\sum_{\alpha_3,\ldots,\alpha_N}\sum_{\beta_2} A^{[1]j_1}_{\alpha_1\beta_2}\tilde{M}^{[2]j_2}_{\beta_2\alpha_3}\ldots M^{[N]j_N}_{\alpha_N\alpha_{N+1}}\,|j_1,j_2,\ldots,j_N\rangle\,.\end{aligned} \tag{2.21}$$

We proceed analogously with the next sites hence for $n=2$ we get:

$$\tilde{M}^{[2]j_2}_{\beta_2\alpha_3} = \tilde{M}^{[2]}_{(j_2\beta_2)\alpha_3} = \sum_{\beta_3} A^{[2]}_{j_2\beta_2\beta_3} X_{\beta_3\alpha_3} \Rightarrow A^{[2]j_2}_{\beta_2\beta_3} \text{ and } \tilde{M}^{[3]j_3}_{\beta_3\alpha_4}. \tag{2.22}$$

We iterate over every site in this manner until we reach the last site, where we obtain a 1×1 transformation matrix $X_{\beta_{N+1}\alpha_{N+1}}$ containing the norm of the state $|\psi\rangle$, which complies with the boundary condition of the MPS. We rename the index α_1 to β_1 and get

$$|\psi\rangle = \sum_{j_1,j_2,\ldots,j_N}\sum_{\beta_2,\ldots,\beta_N} A^{[1]j_1}_{\beta_1\beta_2} A^{[2]j_2}_{\beta_2\beta_3}\ldots A^{[N]j_N}_{\beta_N\beta_{N+1}}\,|j_1,j_2,\ldots,j_N\rangle \tag{2.23}$$

which is a MPS in left canonical form. The right canonical form can be obtained analogously by starting the above algorithm at the last site and iterating to the left with

$$M^{[n]}_{\alpha_n(j_n\alpha_{n+1})} = \sum_{\beta_n} X_{\alpha_n\beta_n} B^{[n]}_{\beta_n(j_n\alpha_{n+1})} \quad \text{with} \quad X_{\alpha_n\beta_n} = U_{\alpha_n\beta_n}\Lambda^{[n]}_{\beta_n\beta_n}. \tag{2.24}$$

The left and right canonical form is related to the canonical form [33, 34]

$$|\psi\rangle = \sum_{j_1,j_2,\ldots,j_N} \Lambda^{[1]}\Gamma^{[1]j_1}\Lambda^{[2]}\Gamma^{[2]j_2}\Lambda^{[3]}\ldots\Lambda^{[N]}\Gamma^{[N]j_N}\Lambda^{[N+1]}\,|j_1j_2\ldots j_N\rangle \tag{2.25}$$

where $\Lambda^{[1]} = \Lambda^{[N+1]} = (1)$ are 1×1 boundary matrices, with

$$A^{[n]j_n} = \Lambda^{[n]}\Gamma^{[n]j_n} \qquad \text{and} \qquad B^{[n]j_n} = \Gamma^{[n]j_n}\Lambda^{[n+1]}. \tag{2.26}$$

As it turns out, the canonical form allows us to read off the Schmidt decomposition of the bipartite system $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$, where L represents the left side, containing sites 1 to $n-1$, and R the right side, containing sites n to N. The Schmidt states $|\alpha_n\rangle_{\mathrm{L}}$ and $|\alpha_n\rangle_{\mathrm{R}}$ can be obtained by contracting all tensors left and right to the diagonal matrix $\Lambda^{[n]}$ containing the Schmidt values of the Schmidt decomposition, as visualized in Fig. 2(b). The Schmidt decomposition can be obtained in a similar manner when the MPS of the state is in a mixed canonical form, meaning the MPS is formed by A and B tensors, as visualized in Fig. 2(b) as well.

Note that the QR and LQ decomposition can be used to obtain general semi-unitary matrices A and B. We replace the SVD in eq. (2.20) (eq. (2.24)) with the QR (LQ) decomposition,

where the semi-unitary matrix Q replaces A (B) and R (L) replace the transformation matrix X. However, we are not obtaining the Schmidt values and do not map into the Schmidt base as with the SVD. Hence, we are not yielding the canonical form. We call this form the left and right semi-unitary form. Although the semi-unitary form is weaker than the canonical form, it allows us to make use of the orthogonality conditions

$$\left(A^{[n]}_{(j_n\alpha_n)\beta_n}\right)^\dagger A^{[n]}_{(j_n\alpha_n)\beta_n} = \mathbb{1} \quad \text{and} \quad B^{[n]}_{(j_n\alpha_n)\beta_n}\left(B^{[n]}_{(j_n\alpha_n)\beta_n}\right)^\dagger = \mathbb{1}. \tag{2.27}$$

This relation holds for the canonical form and the semi-unitary form since, in both cases, $A^{[n]}_{(j_n\alpha_n)\beta_n}$ and $B^{[n]}_{(j_n\alpha_n)\beta_n}$ are semi-unitary matrices. By convention, we draw the physical legs of a bra, which is the hermitian conjugate of the ket MPS, hence $\langle\psi| = |\psi\rangle^\dagger$, as pointing upwards and denote them as j'_n, as demonstrated in Fig. 2(c). Hence eq. (2.27) transfers to the relation shown in Fig. 2(d). ([13] sec. 3.2)

2.4.4. Compression of a MPS

To compress a MPS of a state $|\psi\rangle$, we truncate the bond dimensions between the sites. To truncate the bond dimension χ_n of a bond α_n between site $n-1$ and n of a MPS in canonical form, as in 2(b), we keep only the $\eta \leq \chi_n$ most relevant Schmidt values $\Lambda^{[n]}_\alpha$ and truncate the corresponding rows of $A^{[n-1]j_{n-1}}$ and columns of $B^{[n]j_n}$. Mathematically, the truncation is described by a $\chi_n \times \eta$ (or $\eta \times \chi_n$) projection matrix P_η to truncate rows (or columns), which is applied from the right (or left). To keep the state normalized, we renormalize the remaining Schmidt values. As already mentioned in sec. 2.4.2 the MPS can only be compressed efficiently, meaning according to eq. (2.15), if the state obeys the area law. This allows us to limit the required storage space from $\sim d^N$ bytes (see sec. 2.1) to $\sim Nd\eta^2$ bytes for an area law state, assuming a truncated bond dimension η for each bond of the system. ([20] sec. 4.1.3 and 4.5, [13] sec. 3.2)

2.4.5. Matrix Product Operators (MPO)

Similar to the MPS an operator O can be written as MPO

$$O = \sum_{j_1,j_2,\ldots,j_N} \sum_{j'_1,j'_2,\ldots,j'_N} v^L W^{[1]j_1j'_1} W^{[2]j_2j'_2} \ldots W^{[N]j_Nj'_N} v^R \left|j_1, j_2, \ldots, j_N\right\rangle \left\langle j'_1, j'_2, \ldots, j'_N\right| \tag{2.28}$$

where $W^{[n]j_nj'_n}$ is a $D \times D$ matrix with the physical indices j_n and j'_n. v^L and v^R are the left and right boundary vectors of the MPO. The matrix $W^{[n]j_n,j'_n}$ contains D^2 $d \times d$ matrices and forms the tensor $W^{[n]j_nj'_n}_{\gamma_n\gamma_{n+1}}$. The diagrammatic notation of a MPO can be seen in Fig. 2(e). In sec. 4.1.1 we show how the MPO of a Hamiltonian, and its corresponding time evolution operator is obtained. ([20] sec. 5., [13] sec. 3.4)

3. MPO Based Time-Evolution

In this chapter, we propose, analog to the QR decomposition based TEBD [11], a MPO based time evolution [12] which relies on the QR decomposition instead of the SVD. We start by showing how to apply a time evolution MPO to a state represented by a MPS to obtain the MPS of the evolved state. This procedure is referred to as the time evolution algorithm. A central part of this algorithm is the decomposition and truncation of a two-site wavefunction. The algorithm performing this decomposition and truncation is referred to as truncation scheme, truncation method, or just truncator. These truncation methods usually rely on a SVD. This is where the variational QR decomposition based truncation comes into play, which we introduce with and without controlled bond expansion (CBE). Additionally, we implement the established SVD based truncator as a reference. These three truncation schemes are referred to as QR+CBE, QR and SVD. Information about the developed object-based Python library, where the respective algorithms have been implemented, can be found in app. B. We conclude this chapter by calculating the computational cost for the time evolution algorithm and the three truncation schemes.

3.1. Time Evolution Algorithm

As a recap, to compute the evolved state $|\psi_{t+\Delta t}\rangle = |\psi(t_0 + \Delta t)\rangle$ of a state $|\psi_t\rangle = |\psi(t_0)\rangle$, we need to apply the time evolution operator U to the state $|\psi_t\rangle$ (see eq. (2.3)). The evolved state $|\psi_{t+\Delta t}\rangle$ can be determined exactly, but the bond dimension χ would grow exponentially with the time t [35]. To efficiently represent the MPS we want to limit the maximum allowed bond dimension by $\chi_{\max}$. Consequently we only approximate the exact evolved state $|\psi_{t+\Delta t}\rangle$ by an approximation $|\phi_{t+\Delta t}\rangle$. Thus, we want to find an approximated evolved state $|\phi_{t+\Delta t}\rangle$ which has a minimum distance to the exact evolved state $|\psi_{t+\Delta t}\rangle = U|\psi_t\rangle$, hence minimize the distance

$$\Delta^2 = || \, |\phi_{t+\Delta t}\rangle - U|\psi_t\rangle \, ||^2 = \langle\phi_{t+\Delta t}|\phi_{t+\Delta t}\rangle + \langle\psi_t|U^\dagger U|\psi_t\rangle - 2\,\mathrm{Re}\,\langle\phi_{t+\Delta t}|U|\psi_t\rangle\,. \quad (3.1)$$

The time evolution operator U is unitary and $|\psi_t\rangle$ assumed to be normalized, thus $\langle\psi_t|U^\dagger U|\psi_t\rangle$ can be determined to 1. The approach to minimize the distance Δ is very similar to the DMRG algorithm [1], we variationally optimize a two-site wavefunction, represented by

$$(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}} = \sum_{\beta_n, \alpha_{n+1}} (\Lambda_{t+\Delta t})^{[n]}_{\alpha_n \beta_n} (B_{t+\Delta t})^{[n]j_n}_{\beta_n \alpha_{n+1}} (B_{t+\Delta t})^{[n+1]j_{n+1}}_{\alpha_{n+1} \alpha_{n+2}} \quad (3.2)$$

of the MPS, while keeping all other tensors fixed. The diagrammatic notation of $\langle\phi_{t+\Delta t}|\phi_{t+\Delta t}\rangle$ and $\langle\phi_{t+\Delta t}|U|\psi_t\rangle$ is shown in Fig. 3(a), where we introduce the left and right environment $L^{[n]}$ and $R^{[n+1]}$ containing the contraction of the tensors left and right to the two-site tensor $(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$. For $\langle\phi_{t+\Delta t}|\phi_{t+\Delta t}\rangle$ we use the orthonormality condition for the left and right semi-unitary form (see eq. (2.27)) to reduce the term to a local contraction of $(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$

and $\overline{(\theta_{t+\Delta t})}^{[n]j'_n j'_{n+1}}_{\alpha_n \alpha_{n+2}}$. We optimize Δ^2 in eq. (3.1) with respect to $(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$, hence $\partial \Delta^2 / \partial \overline{(\theta_{t+\Delta t})}^{[n]j'_n j'_{n+1}}_{\alpha_n \alpha_{n+2}} = 0$ to obtain a update rule for $(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ as is ilustrated in the diagrammatic notation in Fig. 3(b). Note that we used that the derivative of a tensor network, which is linearly dependent on a tensor, with respect to that tensor, is the tensor network where the respective tensor has been removed (see [29] sec. 5.2.3). As shown in Fig. 3(c), we can determine the approximated evolved state $|\phi_{t+\Delta t}\rangle$ with the following steps:

(i) First, we calculate the left environment $L^{[1]}$ and all right environments $R^{[N]}, R^{[N-1]}, \ldots, R^{[2]}$ with an initial guess for the approximated evolved state $|\phi_{t+\Delta t}\rangle$, which we assume to be in the right semi-unitary form (the developed library uses $|\psi_t\rangle$ as initial guess for $|\phi_{t+\Delta t}\rangle$).

(ii) We compute the two-site tensor $(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ and use a truncation method (for now, we treat them as a given, we will define them in sec. 3.2) to get the tensors $(A_{t+\Delta t})^{[n]j_n}_{\alpha_n \alpha_{n+1}}$ and $(C_{t+\Delta t})^{[n+1]j_{n+1}}_{\alpha_{n+1} \alpha_{n+2}}$ for site n and $n+1$, where we only need the former.

(iii) The tensor $\overline{(A_{t+\Delta t})}^{[n]j'_n}_{\alpha_n \alpha_{n+1}}$ is contracted to the left environment $L^{[n]}$, together with $(A_t)^{[n]j_n}_{\alpha_n \alpha_{n+1}}$ and $W^{[n]j_n j'_n}_{\gamma_n \gamma_{n+1}}$, to retrive the left environment $L^{[n+1]}$. It is convenient to store the left environments $L^{[n]}$ for each site n before contracting them to retrieve $L^{[n+1]}$, as they are needed in the next step.

We perform step (ii) and (iii) for $n = 1, 2, \ldots, N-2$ and collect the left environments $L^{[1]}, L^{[2]}, \ldots, L^{[N-1]}$. Then, we change the direction and go from right to left, starting at site $N-1$:

(iv) We compute the two-site tensor $(\theta_{t+\Delta t})^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ and use a truncation method to get the tensors $(C_{t+\Delta t})^{[n]j_n}_{\alpha_n \alpha_{n+1}}$ and $(B_{t+\Delta t})^{[n+1]j_{n+1}}_{\alpha_{n+1} \alpha_{n+2}}$ for site n and $n+1$, where we only need the latter. Depending on the truncation method, we receive the Schmidt values $(\Lambda_{t+\Delta t})^{[n+1]}$ for site $n+1$ as well.

(v) Besides using the obtained tensors $(B_{t+\Delta t})^{[n+1]j_{n+1}}_{\alpha_{n+1} \alpha_{n+2}}$ to get the right environment $R^{[n]}$ for the next step, we save the tensor together with the Schmidt values $(\Lambda_{t+\Delta t})^{[n+1]}$, if available, to update the initial guess of the approximated evolved state $|\phi_{t+\Delta t}\rangle$.

We perform step (iv) and (v) for $n = N-1, N-2, \ldots, 1$ and collect the updated right environments $R^{[N]}, R^{[N-1]}, \ldots, R^{[2]}$. Once we reach the first site again, one "sweep" is completed, and we have an updated approximated evolved state $|\phi_{t+\Delta t}\rangle$ in the right semi-unitary form. If we are not satisfied with the distance Δ between the approximated evolved state $|\phi_{t+\Delta t}\rangle$ and the exact evolved state $|\psi_{t+\Delta t}\rangle$ we can, to some extent, improve the approximation with additional sweeps. We will investigate the improvement of the distance Δ in dependence on the number of sweeps on a practical model in sec. 4.1.2. Once satisfied with the approximation one time step form $|\psi_t\rangle$ to $|\phi_{t+\Delta t}\rangle \approx |\psi_{t+\Delta t}\rangle$ is completed.

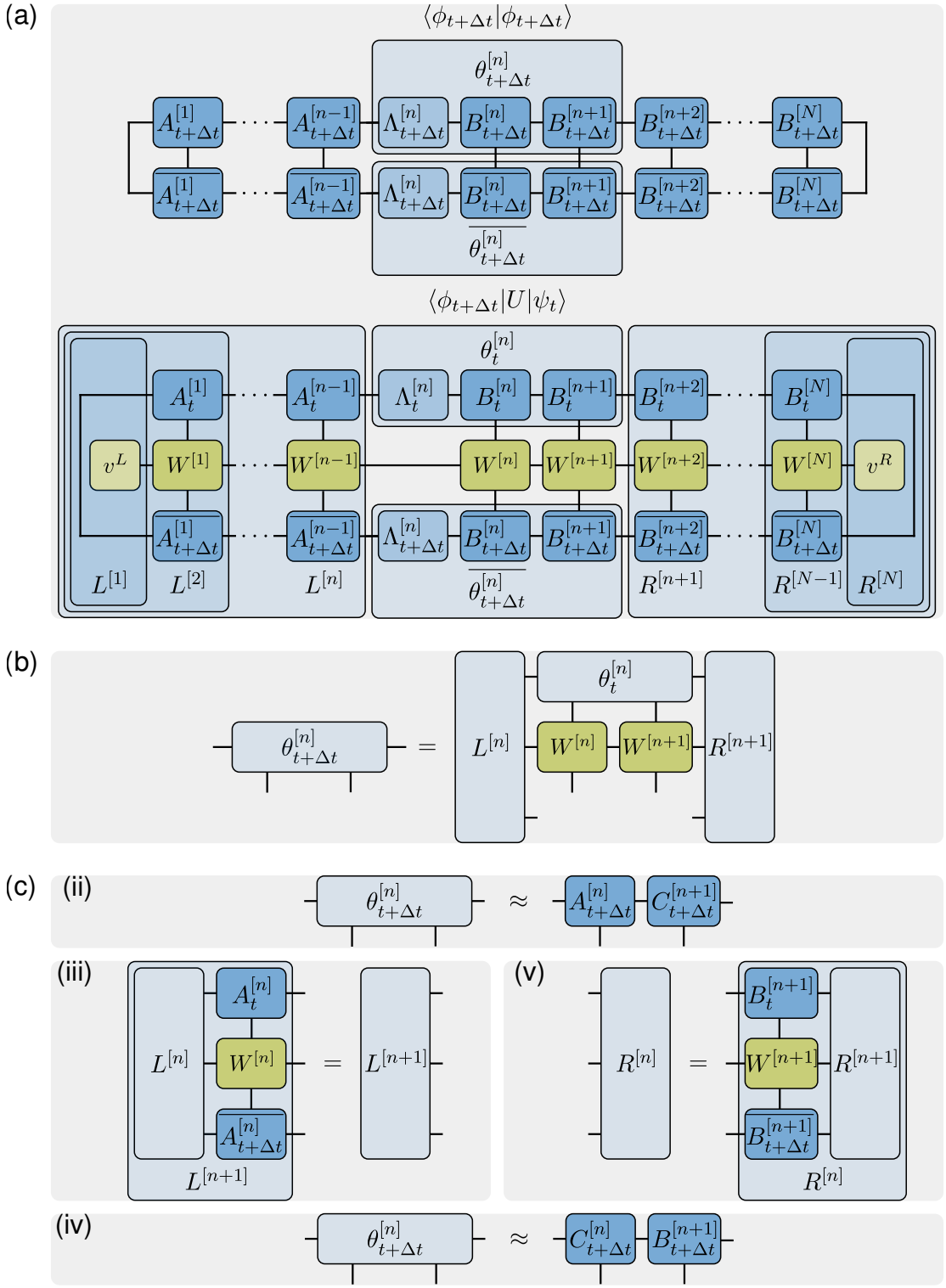


Figure 3: (a) The squared norm $\langle\phi_{t+\Delta t}|\phi_{t+\Delta t}\rangle$ of the approximated evolved state $|\phi_{t+\Delta t}\rangle$ in mixed canonical form and its overlap $\langle\phi_{t+\Delta t}|U|\psi_t\rangle$ with the exact evolved state $|\psi_{t+\Delta t}\rangle = U|\psi_t\rangle$, where the time evolution operator U is given by an MPO. The left and right environments $L^{[n]}$ and $R^{[n]}$ contain the contractions left and right to site n and $n+1$. (b) The update rule for the two-site wavefunction represented by the two-site tensor $\theta^{[n]}_{t+\Delta t}$ as a result of the minimization of Δ^2 in eq. (3.1). (c) Steps for one sweep: (ii)/(iv) Calculate and truncate the two-site tensor $\theta^{[n]}_{t+\Delta t}$ to retrieve the left/right semi-unitary form $A^{[n]}_{t+\Delta t}/B^{[n+1]}_{t+\Delta t}$ for site $n/n+1$. (iii)/(v) Use the obtained tensor $A^{[n]}_{t+\Delta t}/B^{[n+1]}_{t+\Delta t}$ to update the left/right environment L/R.

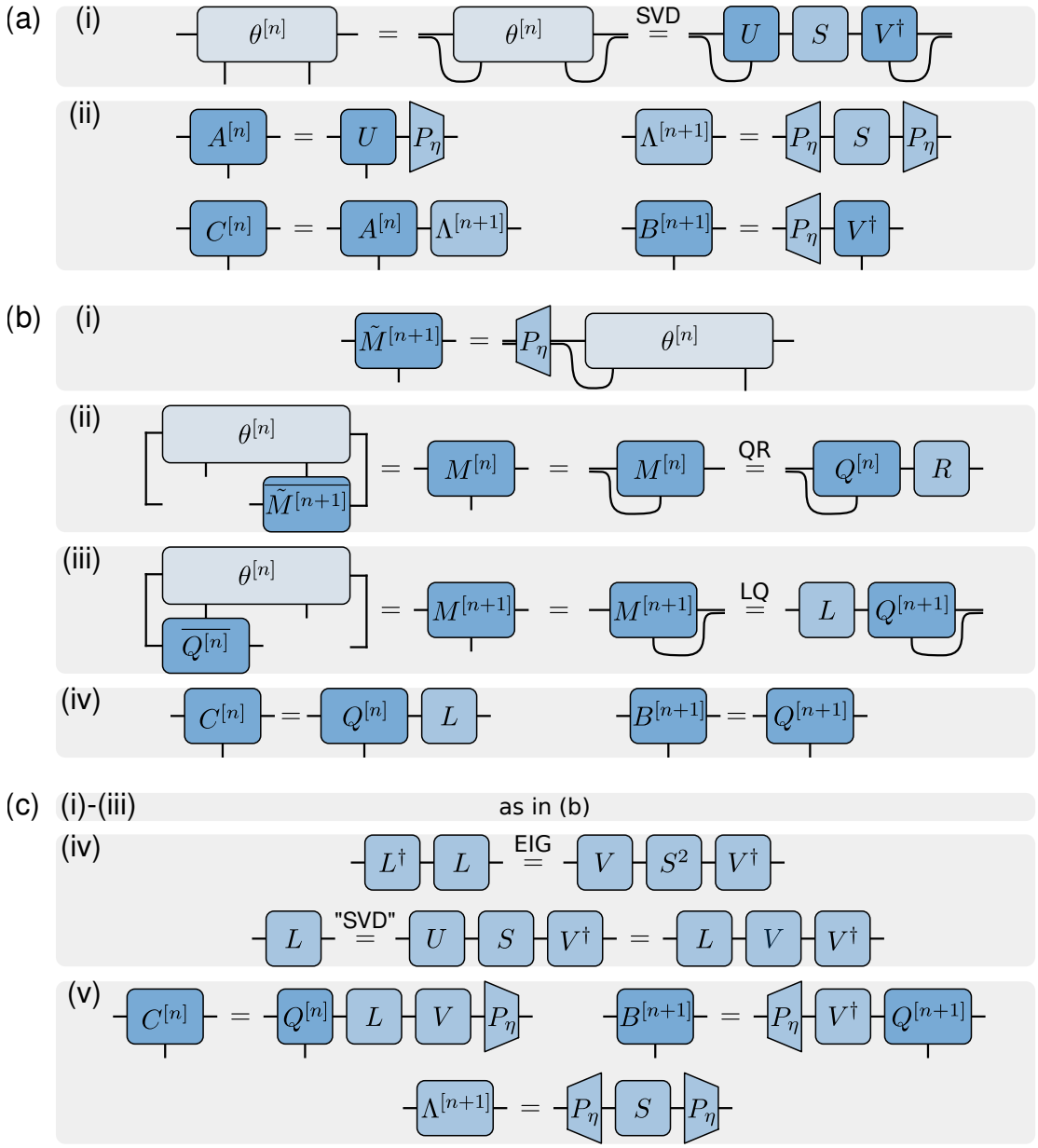


Figure 4: Diagramatic notation on how to obtain the right semi-unitary form $B^{[n+1]}$ for site $n+1$ of the different truncation schemes: (a) Algorithm for the SVD based truncation method: (i) Perform a SVD of the reshaped two-site tensor $\theta^{[n]}$. (ii) Truncate the tensors and assign them to $A^{[n]}$, $B^{[n+1]}$ and $\Lambda^{[n+1]}$. Compute $C^{[n]}$ to have a uniform layout. (b) Algorithm for the QR decomposition based truncation method: (i) Use the truncated two-site tensor $\theta^{[n]}$ as an initial guess for the tensor $\tilde{M}^{[n+1]}$ at site $n+1$. (ii) Contract the two-site tensor $\theta^{[n]}$ with $\overline{\tilde{M}}^{[n+1]}$ to get the tensor $M^{[n]}$ and perform a QR decomposition of $M^{[n]}$. (iii) Contract the two-site tensor $\theta^{[n]}$ with $\overline{Q}^{[n]}$ to get the tensor $M^{[n+1]}$ and perform a LQ decomposition of $M^{[n+1]}$. Repeat step (ii) and (iii) with $\tilde{M}^{[n+1]} = Q^{[n+1]}$ to improve the decomposition if desired. (iv) Assign the tensor $Q^{[n+1]}$ to $B^{[n+1]}$ and compute $C^{[n]}$. (c) Algorithm for the QR decomposition based method with CBE: Steps (i)-(iii) are the same as in (b). (iv) Perform a spectral-decomposition (EIG) of $L^\dagger L$ to obtain the matrix V and S, yielding parts of the SVD of L. Instead of calculating the "left" side U of the SVD we write L as $LVV^\dagger$. (v) Use $V^\dagger$ to map $Q^{[n+1]}$ into the Schmidt base. Contract and truncate the tensors and assign them to $C^{[n]}$, $B^{[n+1]}$ and $\Lambda^{[n+1]}$.

3.2. Truncation Schemes

In this section, we define the three truncation schemes (SVD, QR and QR+CBE) to decompose and truncate a two-site tensor into either $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}} = \sum_{\alpha_{n+1}} A^{[n]j_n}_{\alpha_n \alpha_{n+1}} C^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$ if the left semi-unitary form of site n is of interest or $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}} = \sum_{\alpha_{n+1}} C^{[n]j_n}_{\alpha_n \alpha_{n+1}} B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$ if the right semi-unitary form of site $n+1$ is of interest. In contrast to the QR truncator, we obtain $A^{[n]j_n}_{\alpha_n\alpha_{n+1}}$ or $B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$ in the left or right canonical form including the Schmidt values $(\Lambda_{t+\Delta t})^{[n+1]}$ for site $n+1$ for the SVD and QR+CBE based truncation method. In the following, we explain the truncation methods as if the right semi-unitary form of site $n+1$ is of interest. The left semi-unitary form of site n can be obtained analogously. Moreover, for simplicity we refer to $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ as $d \times d \times \chi \times \chi$ tensor, which is in general not the case. All three methods are designed to handle any $d \times d \times \chi_1 \times \chi_2$ tensor.

3.2.1. SVD Based Method

Out of the three methods, the SVD based truncation method is the most straightforward one. We essentially perform a Schmidt decomposition of the bipartite system $\mathcal{H} = \mathcal{H}_n \otimes \mathcal{H}_{n+1}$. We proceed as illustrated in Fig. 4(a):

(i) We start by reshaping the tensor $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ into the $d\chi \times \chi d$ matrix $\theta^{[n]}_{(j_n\alpha_n)(\alpha_{n+2}j_{n+1})}$ and perform a SVD of this matrix: $\theta^{[n]}_{(j_n\alpha_n)(\alpha_{n+2}j_{n+1})} = \sum_\beta U^{[n]}_{(j_n\alpha_n)\beta} S^{[n+1]}_{\beta\beta} (V^\dagger)^{[n+1]}_{\beta(\alpha_{n+2}j_{n+1})}$. The obtained semi-unitary matrices are reshaped into the tensors $U^{[n]j_n}_{\alpha_n\beta}$ and $(V^\dagger)^{[n+1]j_{n+1}}_{\beta\alpha_{n+2}}$.

(ii) Next, we truncate the tensors and assigning them to the sites: $A^{[n]j_n}_{\alpha_n\gamma} = \sum_\beta U^{[n]j_n}_{\alpha_n\beta}(P_\eta)_{\beta\gamma}$, $\Lambda^{[n+1]}_{\gamma\alpha_{n+1}} = \sum_\beta (P_\eta)_{\gamma\beta} S^{[n+1]}_{\beta\beta} (P_\eta)_{\beta\alpha_{n+1}}$ and $B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}} = \sum_\beta (P_\eta)_{\alpha_{n+1}\beta} (V^\dagger)^{[n+1]j_{n+1}}_{\beta\alpha_{n+2}}$. Finally, we renormalize $\Lambda^{[n+1]}_{\gamma\alpha_{n+1}}$ and compute $C^{[n]j_n}_{\alpha_n\alpha_{n+1}} = \sum_\gamma A^{[n]j_n}_{\alpha_n\gamma} \Lambda^{[n+1]}_{\gamma\alpha_{n+1}}$ to get a uniform layout of the returned tensors of the truncators.

The projection matrix P_η (as introduced in sec. 2.4.4) truncates the bond dimension χ to a desired value η by keeping the η largest singular values. We dynamically adjusted $\eta \leq \chi_{\max}$ to the number of Schmidt values, which are bigger than a chosen threshold. Moreover, $\chi_{\max}$ is a defined limit for η to avoid a blow-up in the bond dimension. The SVD based truncation delivers the best possible approximation of the two-site tensor $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$, as it exactly yields the Schmidt decomposition. The truncation error can be determined analog to eq. (2.15), by

$$\varepsilon_{\text{trunc}} = \sqrt{\sum_{\alpha=\eta+1}^{\chi d} (\Lambda^{[n+1]})^2_\alpha}. \tag{3.3}$$

3.2.2. QR Decomposition Based Method

While the QR decomposition gives us a semi-unitary matrix, we do not obtain the Schmidt values or map into the Schmidt base as with the SVD. Meaning we can not directly perform a QR decomposition of the two-site matrix $\theta^{[n]}_{(j_n\alpha_n)(\alpha_{n+2}j_{n+1})}$ since we would have no measure

on how to truncate the matrices afterward. Instead, we perform a local minimization as introduced in Ref. [11]. We proceed analogously to sec. 3.1; We want to find a state $|\theta_{\mathrm{app}}\rangle$ which has a minimum distance to the exact state $|\theta\rangle$. Hence, we want to minimize the distance

$$\Delta^2 = ||\,|\theta\rangle - |\theta_{\mathrm{app}}\rangle\,||^2 = \langle\theta|\theta\rangle + \langle\theta_{\mathrm{app}}|\theta_{\mathrm{app}}\rangle - 2\,\mathrm{Re}\,\langle\theta|\theta_{\mathrm{app}}\rangle\,. \tag{3.4}$$

We proceed as illustrated in Fig. 4(b):

(i) We start by reshaping the tensor $\theta^{[n]j_nj_{n+1}}_{\alpha_n\alpha_{n+2}}$ into the $d\chi \times d \times \chi$ tensor $\theta^{[n]j_{n+1}}_{(j_n\alpha_n)\alpha_{n+2}}$. Then we truncate the leg $(j_n\alpha_n)$ of the tensor and use it as our initial guess for the tensor for site $n+1$: $\tilde{M}^{[n+1]j_{n+1}}_{\beta\alpha_{n+2}} = \sum_{(j_n\alpha_n)}(P_\eta)_{\beta(j_n\alpha_n)}\theta^{[n]j_{n+1}}_{(j_n\alpha_n)\alpha_{n+2}}$. Note that we start with the initial guess at site n instead of $n+1$ when the left semi-unitary form of site n is of interest.

(ii) We define the approximated state as $|\theta_{\mathrm{app}}\rangle = \sum_{j_n,j_{n+1}}\sum_\beta M^{[n]j_n}_{\alpha_n\beta}\tilde{M}^{[n+1]j_{n+1}}_{\beta\alpha_{n+2}}\,|j_n, j_{n+1}\rangle$, where $M^{[n]j_n}_{\alpha_n\beta}$ is the tensor at site n. We update $M^{[n]j_n}_{\alpha_n\beta}$ by minimizing eq. (3.4) with respect to $M^{[n]j_n}_{\alpha_n\beta}$ and using the orthonormality condition to get the update rule: $M^{[n]j_n}_{\alpha_n\beta} = \sum_{j_{n+1},\alpha_{n+2}}\theta^{[n]j_nj_{n+1}}_{\alpha_n\alpha_{n+2}}\overline{\tilde{M}}^{[n+1]j_{n+1}}_{\beta\alpha_{n+2}}$. We reshape the updated tensor $M^{[n]j_n}_{\alpha_n\beta}$ into the $\chi d \times \eta$ matrix $M^{[n]}_{(j_n\alpha_n)\beta}$ and perform a QR decomposition to obtain the left semi-unitary form for site n: $M^{[n]}_{(j_n\alpha_n)\beta} = \sum_\gamma Q^{[n]}_{(j_n\alpha_n)\gamma}R_{\gamma\beta}$. By reshaping the tensor, we obtain $Q^{[n]j_n}_{\alpha_n\gamma}$, which is our new guess for site n.

(iii) Similar to step (ii) we get the update rule for the tensor on site $n+1$: $M^{[n+1]j_{n+1}}_{\gamma\alpha_{n+2}} = \sum_{j_n,\alpha_n}\overline{Q}^{[n]j_n}_{\alpha_n\gamma}\theta^{[n]j_nj_{n+1}}_{\alpha_n\alpha_{n+2}}$. Next, we reshape $M^{[n+1]j_{n+1}}_{\gamma\alpha_{n+2}}$ into the $\eta \times \chi d$ matrix $M^{[n+1]}_{\gamma(\alpha_{n+2}j_{n+1})}$ and perform a LQ decomposition to obtain the right semi-unitary form for site $n+1$: $M^{[n+1]}_{\gamma(\alpha_{n+2}j_{n+1})} = \sum_{\alpha_{n+1}}L_{\gamma\alpha_{n+1}}Q^{[n+1]}_{\alpha_{n+1}(\alpha_{n+2}j_{n+1})}$. By reshaping the tensor, we obtain $Q^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$, which is our new guess for site $n+1$.

This completes one iteration. The approximation $|\theta_{\mathrm{app}}\rangle$ can be improved by either fixing $\eta = \chi_{\max}$ and repeating step (ii) and (iii) N_{it} times with an updated initial guess $\tilde{M}^{[n+1]j_{n+1}}_{\beta\alpha_{n+2}} = Q^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$, which scales with $N_{\mathrm{it}}\chi^3d^2$, or by increasing η, which scales with $\chi^2\eta d^2$. The stated scaling factors originate in eq. (3.8) in the upcoming section about the computational costs. The approximation $|\theta_{\mathrm{app}}\rangle$ can be quantified by the truncation error $\varepsilon_{\mathrm{trunc}}$, as introduced in eq. (3.5). Once satisfied with the approximation, we can proceed with the final step:

(iv) Finally we renormalize $L_{\gamma\alpha_{n+1}}$, compute $C^{[n]j_n}_{\alpha_n\alpha_{n+1}} = \sum_\gamma Q^{[n]j_n}_{\alpha_n\gamma}L_{\gamma\alpha_{n+1}}$ and asign $B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}} = Q^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$.

The default setting of the developed library is to not improve $|\theta_{\mathrm{app}}\rangle$, neither with more iterations nor with an adjusted η. We will see how this affects the time evolution in sec. 4.2 by monitoring the truncation error, which we define as

$$\varepsilon_{\mathrm{trunc}} = ||\theta - \theta_{\mathrm{app}}||. \tag{3.5}$$

3.2.3. QR Decomposition Based Method with CBE

To improve the QR decomposition based method, we determine the Schmidt values $\Lambda^{[n+1]}$ and map $Q^{[n+1]j_{n+1}}$ into the Schmidt base by performing a spectral-decomposition of $L^\dagger L$. This allows us to implement a CBE for the QR decomposition based method. To do so we expand steps (i)-(iii) and replace step (iv) from sec. 3.2.2 by the following steps (as illustrated in Fig. 4(c)):

(iv) To obtain the spectral-decomposition (EIG in Fig. 4(c)), we calculate the eigenvalues and eigenvectors of $L^\dagger L$, arrange the eigenvalues in descending order in a diagonal matrix D and the corresponding eigenvectors in a matrix V to get: $L^\dagger L = VDV^\dagger$. We know from sec. 2.2.1, that performing the spectral-decomposition of $L^\dagger L$ yields the singular values $S = \sqrt{D}$ and a unitary matrix V of the SVD of L: $L = USV^\dagger$. Consequently we can replace the matrix $L_{\gamma\alpha_{n+1}}$ from step (iii) in sec. 3.2.2 by $USV^\dagger$. However, we do not have U, but as it turns out, we do not need it to properly represent $\theta^{[n]j_nj_{n+1}}$, since we can write L as $LVV^\dagger$. This way, we can map $Q^{[n+1]j_{n+1}}$ into the Schmidt base and obtain the Schmidt values S without computing the whole SVD. Note that if the left canonical form of site n is of interest, we perform a spectral-decomposition of $RR^\dagger$ to obtain the unitary matrix U and write R as $UU^\dagger R$.

(v) According to step (ii) and (iii) from sec. 3.2.2 the two-site tensor can be written as $\theta^{[n]j_nj_{n+1}}_{\alpha_n\alpha_{n+2}} = \sum_{\gamma,\alpha_{n+1}} Q^{[n]j_n}_{\alpha_n\gamma} L_{\gamma\alpha_{n+1}} Q^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$. Now we replace $L_{\gamma\alpha_{n+1}}$ with the expression derived in the last step and get $\theta^{[n]j_nj_{n+1}}_{\alpha_n\alpha_{n+2}} = \sum_{\gamma,\delta,\zeta,\kappa} Q^{[n]j_n}_{\alpha_n\gamma} L_{\gamma\delta} V_{\delta\zeta} (V^\dagger)_{\zeta\kappa} Q^{[n+1]j_{n+1}}_{\kappa\alpha_{n+2}}$. We compute and truncate the tensors $C^{[n]j_n}_{\alpha_n\alpha_{n+1}} = \sum_{\gamma,\delta,\zeta} Q^{[n]j_n}_{\alpha_n\gamma} L_{\gamma\delta} V_{\delta\zeta} (P_\eta)_{\zeta\alpha_{n+1}}$ and $B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}} = \sum_{\zeta,\kappa} (P_\eta)_{\alpha_{n+1}\zeta} (V^\dagger)_{\zeta\kappa} Q^{[n+1]j_{n+1}}_{\kappa\alpha_{n+2}}$. For the Schmidt values we get $\Lambda^{[n+1]}_{\gamma\alpha_{n+1}} = \sum_\beta (P_\eta)_{\gamma\beta} S_{\beta\beta} (P_\eta)_{\beta\alpha_{n+1}}$. We conclude the step by renormalizing $C^{[n]j_n}_{\alpha_n\alpha_{n+1}}$ and $\Lambda^{[n+1]}_{\gamma\alpha_{n+1}}$.

A few remarks to steps (iv) and (v): The matrix U of the SVD can be obtained by performing a spectral-decomposition of $LL^\dagger$. However, the obtained matrix U and the matrix V from step (iv) would, in general, not be part of "the same" SVD since the matrices U and V are not unique (see sec. 2.2.1). We can still obtain U with V, since we know that $USV^\dagger = LVV^\dagger$ and thus $U = LVS^{-1}$. This would allow us to map $Q^{[n]j_n}$ into the Schmidt base as well, and we would be able to yield the whole Schmidt decomposition as in the SVD based method in sec. 3.2.1. But, the inverse S^{-1} of the Schmidt values is numerically unstable due to the fact that the Schmidt values can get very small. Nonetheless, it shows why we can truncate the tensor $C^{[n]j_n}_{\alpha_n\alpha_{n+1}}$. Even though $C^{[n]j_n}_{\alpha_n\alpha_{n+1}}$ is not the Schmidt base by applying $LV = US$ to $Q^{[n]j_n}$ we receive the Schmidt base scaled by the Schmidt values. Consequently, we can truncate $C^{[n]j_n}_{\alpha_n\alpha_{n+1}}$.

With the CBE based truncation method, we can dynamically adjust $\eta \leq \chi_{\max}$ to the number of Schmidt values, which are bigger than a chosen threshold and limit η by $\chi_{\max}$. If $\eta = \chi d$, the obtained tensor $B^{[n+1]j_{n+1}}$ from the QR decomposition based method is equivalent to the one obtained from the SVD based method. In general, we do not use $\eta = \chi d$, since we would

lose our speedup in the scaling, as evident from eq. (3.8). This means the truncation error $\varepsilon_{\mathrm{trunc}}$ is, in general, larger than the one from the SVD based truncation. The truncation error $\varepsilon_{\mathrm{trunc}}$ can be determined as in eq. (3.5).

3.3. Computation Cost

The scaling of the computational cost of the time evolution algorithm, as presented in sec. 3.1 is

$$\mathrm{O}_{\mathrm{Alg}} = (N-2)(\mathrm{O}_{(\mathrm{i})} + 2N_{\mathrm{sw}}(\mathrm{O}_{(\mathrm{ii})/(\mathrm{iv})} + \mathrm{O}_{(\mathrm{iii})/(\mathrm{v})})) \tag{3.6}$$

where N_{sw} are the number of sweeps and $\mathrm{O}_{(\xi)}$ denotes the scaling of the operations in step (ξ), reaching from (i) to (v) (see sec. 3.1). They are determined to

$$\begin{aligned} \mathrm{O}_{(\mathrm{i})} &= 2\chi^3 Dd + \chi^2 D^2 d^2 \\ \mathrm{O}_{(\mathrm{ii})/(\mathrm{iv})} &= \chi^3 d^2 D + 2\chi^3 Dd + 2\chi^2 D^2 d^2 + \mathrm{T}_{\mathrm{trunc}} \\ \mathrm{O}_{(\mathrm{iii})/(\mathrm{v})} &= \chi^3 Dd \end{aligned} \tag{3.7}$$

where $\mathrm{T}_{\mathrm{trunc}}$ is the scaling of the truncation shemes in sec. 3.2 which are determined to

$$\begin{aligned} \mathrm{T}_{\mathrm{SVD}} &= \mathrm{SVD}_{\chi d \times \chi d} + \chi^3 d = \chi^3 d^3 + \chi^3 d \\ \mathrm{T}_{\mathrm{QR}} &= 2N_{\mathrm{it}}\mathrm{QR}_{\chi d \times \eta} + 2N_{\mathrm{it}}\chi^2 \eta d^2 + (\chi\eta^2 d) = 4N_{\mathrm{it}}\chi^2\eta d^2 + (\chi\eta^2 d) \\ \mathrm{T}_{\mathrm{QR+CBE}} &= \mathrm{T}_{\mathrm{QR}} + \eta^3 + \mathrm{EIG}_{\eta\times\eta} + \chi\eta^2 d + (\chi\eta^2 d) = 4N_{\mathrm{it}}\chi^2\eta d^2 + \chi\eta^2 d + 2\eta^3 + (2\chi\eta^2 d). \end{aligned} \tag{3.8}$$

The expressions in brackets are the additional cost for computing the tensor C, which is only necessary if we are interested in the truncation error $\varepsilon_{\mathrm{trunc}}$. For the cost calculation, we only consider the tensor contractions and the decompositions as they consume the majority of the computation time. For the decompositions we assumed the follwoing: the SVD and QR decomposition scale with $m^2 n$ for a matrix $M \in \mathbb{C}^{m\times n}$ with $m \geq n$ and the spectral-decomposition (EIG in eq. (3.8)) of a matrix $M \in \mathbb{C}^{n\times n}$ scales with n^3. The LQ decomposition scales analogously and is listed as QR decomposition in eq. (3.8). Concluding that the algorithm scales with $\mathrm{O}_{\mathrm{Alg}} = \mathcal{O}(NN_{\mathrm{sw}}\chi^3 d^2 D) + \mathrm{T}_{\mathrm{trunc}}$, where $\mathrm{T}_{\mathrm{SVD}} = \mathcal{O}(\chi^3 d^3)$ and $\mathrm{T}_{\mathrm{QR}} = \mathcal{O}(\chi^3 d^2)$ as well as $\mathrm{T}_{\mathrm{QR+CBE}} = \mathcal{O}(\chi^3 d^2)$ for $D, d < \chi = \eta$.

4. Benchmark

In this chapter, we analyze the time evolution algorithm and the truncation schemes with the quantum clock model, as it allows a general scaling of the local Hilbert space dimension d. Before using the model, we examine how many sweeps are needed to obtain a reasonable result. After that, we proceed analogously to the benchmark of the QR decomposition based truncation method for the TEBD [11]: We perform a global quench of the $d = 5$ quantum clock model to check the agreement between the results of the truncation schemes and proceed with a timing benchmark of the time evolution algorithm and the truncation methods. The used hardware for the benchmark is an Intel® Core™ i7-11850H (32 GB RAM) running on 14 threads and a NVIDIA A100 (80 GB RAM). The CPU calculations run on the NumPy backend while the GPU calculations run on PyTorch. All simulations are performed in double precision, meaning complex128.

4.1. Model Implementation

4.1.1. Quantum Clock Model

The Hamiltonian of the quantum clock model for nearest- and next-nearest-neighbor interactions reads

$$H = -J_1 \sum_{\mathrm{n}} (Z_{\mathrm{n}} Z_{\mathrm{n}+1}^{\dagger} + \mathrm{h.c.}) - J_2 \sum_{\mathrm{n}} (Z_{\mathrm{n}} Z_{\mathrm{n}+2}^{\dagger} + \mathrm{h.c.}) - g \sum_{\mathrm{n}} (X_{\mathrm{n}} + \mathrm{h.c.}) \tag{4.1}$$

with the unitary operators

$$Z = \begin{pmatrix} 1 & & & & \\ & \omega & & & \\ & & \omega^2 & & \\ & & & \ddots & \\ & & & & \omega^{d-1} \end{pmatrix} \quad \text{and} \quad X = \begin{pmatrix} 0 & 1 & & & \\ & 0 & 1 & & \\ & & 0 & \ddots & \\ & & & \ddots & 1 \\ 1 & & & & 0 \end{pmatrix} \tag{4.2}$$

where $\omega = e^{2\pi i/d}$. The operators X and Z are generalizations of the Pauli matrices, meaning $X = \sigma^x$ and $Z = \sigma^z$ for $d = 2$. Note that X and Z are dimensionless quantities. Consequently, the coupling constants J_1, J_2, and g have the unit J. While there is research [36] on the phase transition of the quantum clock model for nearest-neighbor interaction for different Hilbert space dimensions d, there seems to be no information of the phase transitions for nearest- and next-nearest-neighbor interactions.

In the following, we derive the MPO for the Hamiltonian H and its time evolution operator U by viewing the MPO as a finite-state machine as introduced in Ref. [12]. We start by expressing

the Hamiltonian in eq. (4.1) with tensor products as

$$\begin{aligned} H = &- J_1(Z \otimes Z^\dagger \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \mathbb{1} \otimes Z \otimes Z^\dagger \otimes \cdots \otimes \mathbb{1} + \ldots \\ &\quad + Z^\dagger \otimes Z \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \mathbb{1} \otimes Z^\dagger \otimes Z \otimes \cdots \otimes \mathbb{1} + \ldots) \\ &- J_2(Z \otimes \mathbb{1} \otimes Z^\dagger \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes Z^\dagger \otimes \cdots \otimes \mathbb{1} + \ldots \\ &\quad + Z^\dagger \otimes \mathbb{1} \otimes Z \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \mathbb{1} \otimes Z^\dagger \otimes \mathbb{1} \otimes Z \otimes \cdots \otimes \mathbb{1} + \ldots) \\ &- g((X + X^\dagger) \otimes \mathbb{1} \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \mathbb{1} \otimes (X + X^\dagger) \otimes \mathbb{1} \otimes \cdots \otimes \mathbb{1} + \ldots) \end{aligned} \tag{4.3}$$

which can be represented as a $D = 6$ MPO with

$$W_H^{[n]} = \begin{array}{c} \begin{matrix} \mathbb{1}_L & a_1 & a_2 & a_3 & a_4 & I_R \end{matrix} \\ \begin{pmatrix} \mathbb{1} & Z & Z^\dagger & 0 & 0 & -g(X + X^\dagger) \\ & & & \mathbb{1} & 0 & -J_1 Z^\dagger \\ & & & & \mathbb{1} & -J_1 Z \\ & & & & & -J_2 Z^\dagger \\ & & & & & -J_2 Z \\ & & & & & \mathbb{1} \end{pmatrix} \end{array} \begin{matrix} I_L \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ \mathbb{1}_R \end{matrix} \quad \text{and} \quad \begin{aligned} v_H^L &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \\ v_H^R &= \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}^T \end{aligned} \tag{4.4}$$

where each variable a_i leads to one interaction term in eq. (4.3) and $I_{L(R)}$ places an identity matrix $\mathbb{1}$ on the left (right) side of the expression. To obtain the time evolution operator we need to compute $U(\Delta t) = e^{-\mathrm{i}\Delta t H/\hbar} = e^{\tau H}$ with $\tau = -\mathrm{i}\Delta t/\hbar$ (see eq. (2.3)). Note that we set $\hbar = 1$ and thus Δt has the unit $\frac{1}{\mathrm{J}}$. In general, $U(\Delta t)$ can only be approximated. Ref. [12] gives instructions on deriving such approximations with a modified euler step. The main idea is that a Hamiltonian expressed as a sum of terms admits a local version of a Runge-Kutta step, which allows to improve the Euler step. This leads to a compact MPO representation with a constant error per site that can be extended to higher-order approximations in $\mathcal{O}(\Delta t^p)$. We use the approximation $U^{\mathrm{I}}(\Delta t)$ with an total error of $\mathcal{O}(N\Delta t^2)$. In accordance with [12] eq. 8 we construct the $D = 5$ time evolution MPO with the entries $\hat{A}, \hat{B}, \hat{C}, \hat{D}$ of $W_H^{[n]}$ to obtain

$$W_U^{\mathrm{I}[n]} = \begin{pmatrix} \mathbb{1} - \tau g(X + X^\dagger) & \sqrt{\tau} Z & \sqrt{\tau} Z^\dagger & 0 & 0 \\ -\sqrt{\tau} J_1 Z^\dagger & & & \mathbb{1} & 0 \\ -\sqrt{\tau} J_1 Z & & & & \mathbb{1} \\ -\sqrt{\tau} J_2 Z^\dagger & & & & \\ -\sqrt{\tau} J_2 Z & & & & \end{pmatrix} \quad \text{and} \quad \begin{aligned} v_U^L &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\ v_U^R &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}^T \end{aligned} \tag{4.5}$$

The fact that the MPO given by the $W_U^{\mathrm{I}[n]}$ is only an approximation to the exact time evolution operator means that we must choose small time steps Δt to obtain a good approximation of the evolved state. Moreover, we mentioned in sec. 2.1 that the time evolution operator needs to be unitary, which is not the case for our approximation. This means that, in general, a MPS is no longer normalized after applying the MPO.

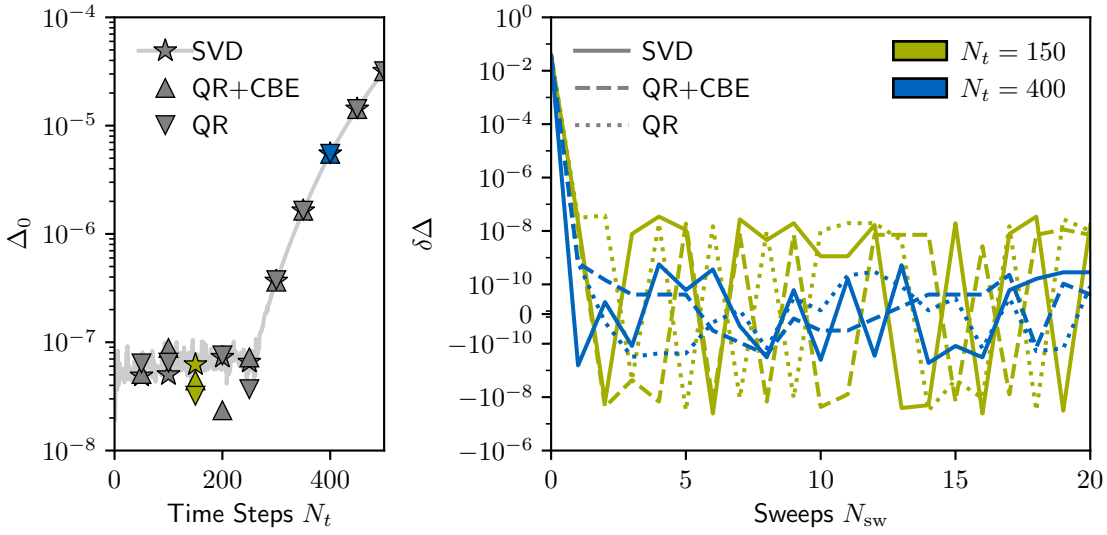


Figure 5: Distance $\Delta = \Delta_0 + \delta\Delta$ of a time evolution with $\Delta t = 0.001$ for the $d = 5$ quantum clock model with the parameters $J_1 = J_2 = 1$ and $g = 3$ of a system with $N = 10$ sites for different truncation schemes (marker shape on the left, linestyle on the right). We start with the $Z = 1$ product state at $t = 0$. On the left every 50th datapoint of $\Delta_0 = \lim_{N_{\mathrm{sw}}\to\infty} \Delta$ is plotted against the time steps N_t. In addition, all data points of the SVD based truncation method for Δ_0 are shown as a solid grey line. On the right, the deviation $\delta\Delta$ to Δ_0 after N_{sw} sweeps is shown for two selected time steps N_t (marker color on the left, line color on the right). For every time step N_t, a total of $N_{\mathrm{sw}} = 20$ sweeps are performed. The bond dimension for all truncation methods is limited by $\chi_{\max} = 62$. For the SVD and QR+CBE method, we discard all Schmidt values smaller than 10^{-14}.

4.1.2. Number of Sweeps

To determine the required number of sweeps to obtain a reasonable result, we analyze the distance Δ between the approximated evolved state $|\phi_{t+\Delta t}\rangle$ and the exact evolved state $|\psi_{t+\Delta t}\rangle = U|\psi_t\rangle$ as introduced in eq. (3.1). To account for the not norm preserving time evolution operator U we normalize the states and get

$$\Delta^2 = \left|\left| \frac{|\phi_{t+\Delta t}\rangle}{||\,|\phi_{t+\Delta t}\rangle\,||} - \frac{U|\psi_t\rangle}{||U|\psi_t\rangle||} \right|\right|^2 = 2 - 2\frac{\mathrm{Re}\,\langle\phi_{t+\Delta t}|U|\psi_t\rangle}{\sqrt{\langle\phi_{t+\Delta t}|\phi_{t+\Delta t}\rangle}\sqrt{\langle\psi_t|U^\dagger U|\psi_t\rangle}}. \tag{4.6}$$

where $\langle\phi_{t+\Delta t}|\phi_{t+\Delta t}\rangle = 1$, since the approximated evolved state $|\phi_{t+\Delta t}\rangle$ is normalized due to the nature of the truncation schemes (see sec. 3.2). The distance Δ can be separated into two parts. The first one is a factor which can be characterized as $\Delta_0 = \lim_{N_{\mathrm{sw}}\to\infty} \Delta$ and thus can not be optimized by the number of sweeps N_{sw}. In contrast to that, the deviation $\delta\Delta$ to this factor Δ_0 can be optimized by the number of sweeps N_{sw}. Hence we express the distance as $\Delta = \Delta_0 + \delta\Delta$.

Fig. 5 shows the behavior of Δ_0 and $\delta\Delta$ of a time evolution, based on the SVD, QR and QR+CBE truncation scheme, of a system with $N = 10$ sites in the $d = 5$ quantum clock model. We observe that Δ_0 is continously $\lesssim 10^{-7}$ for $N_t \lesssim 250$. This represents the regime where the bond dimension $\chi_{\max}$ is sufficient to represent the exact evolved state $|\psi_{t+\Delta t}\rangle$.

However, after $N_t \approx 250$, we reach a point where we discard too many Schmidt states that carry significant weight in the state $|\psi_{t+\Delta t}\rangle$. Consequently, we lose precision, and the distance to the exact evolved state $|\psi_{t+\Delta t}\rangle$ increases. By analyzing the deviation $\delta\Delta$ to Δ_0 for two selected time steps N_t in dependence of the number of sweeps N_{sw}, we see that one sweep reduces $|\delta\Delta|$ from $10^{-1} \sim 10^{-2}$ to $\lesssim 10^{-7}$ for both time steps N_t. More than one sweep does not demonstrate to further reduce $\delta\Delta$.

For all truncation methods the parameter Δ_0 fluctuates with a standard deviation $\Delta_{0,\mathrm{SD}}$ around a mean value $\Delta_{0,\mathrm{mean}}$ for $N_t \lesssim 250$. The mean values $\Delta_{0,\mathrm{mean}}$ of the truncation methods match up to $\sim 10^{-8}$, while the standard deviation $\Delta_{0,\mathrm{SD}}$ of the QR+CBE and QR method are almost twice as large as the one from the SVD. At first sight, it might appear as a contradiction that the QR decomposition based truncation methods partially have a lower distance Δ to the exact evolved state than the SVD based truncation method since we mentioned in sec. 3.2.1 that the SVD based truncation delivers the best possible approximation of the two-site wavefunction, which should result in a better approximation of the evolved state $|\phi_{t+\Delta t}\rangle$. However, the distance is only computed to the previous time step, which was calculated with the respective method and not an objectively exact evolved state. Consequently, the QR decomposition based truncation methods can have a lower distance Δ to their respective previous state than the SVD based method to theirs.

4.2. Result Benchmark

In Fig. 6, we perform a time evolution, based on the SVD, QR and QR+CBE truncation scheme, of a system with $N = 10$ sites in the $d = 5$ quantum clock model, hence the same parameters as in sec. 4.1.2. To have a reference for the time evolution algorithm, we perform the same simulation with TeNPy [13]. The local Z and X expectation value and the half chain entanglement entropy S_{vN} extracted from the simulation, based on the three different truncations methods, match with the ones computed by TeNPy up to a relative deviation of $10^{-10} \sim 10^{-11}$ in the regime of acceptable distances $\Delta \lesssim 10^{-5}$, that is until $t \approx 0.3$ for $\chi_{\max} = 62$. Moreover the truncation error $\varepsilon_{\mathrm{trunc}}$ matches up to $\sim 10^{-8}$ in the same regime. Note that there is no data for the entanglement entropy S_{vN} for the QR truncator since we do not obtain the Schmidt values for this method. In addition, we run the SVD and QR+CBE based simulation for multiple bond dimensions $\chi_{\max}$ beyond times where the approximation of the evolved state breaks down, as quantified by the large distance Δ. The dynamic bond truncation of the SVD and QR+CBE method work as expected, verified by the progressive increase of the maximum bond dimension χ in the system until $\chi_{\max}$ is reached. Also, the increase in the entanglement entropy S_{vN} for higher maximum bond dimensions $\chi_{\max}$ is an expected behavior. Allowing a higher bond dimension means being able to hold more Schmidt values and thus reaching higher entanglement entropies (see eq. (2.14)). Additionally, we observe a shift of the truncation error $\varepsilon_{\mathrm{trunc}}$ curve and deviation Δ curve since we can well-approximate the exact evolved state for a longer period of time t. Besides that, the total wall time of the simulations has the same order of magnitude as the TeNPy [13] simula-

tion, indicating that no significant computational mistakes were made, for example, inefficient order of tensor contractions. Concluding that the time evolution algorithm works properly and that all truncation methods lead to a result with similar accuracy within a reasonable wall time.

4.3. Timing Benchmark

In Fig. 7, we perform a timing benchmark of a single update of the MPS from $|\psi_t\rangle$ to $|\phi_{t+\Delta t}\rangle \approx |\psi_{t+\Delta t}\rangle$, based on the SVD, QR and QR+CBE truncation scheme, of a system with $N = 30$ sites in the quantum clock model, where $2/3$ of the matrices of the MPS have full bond dimension χ. Furthermore, we perform a single truncation of an extracted two-site tensor $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ with size $d \times d \times \chi \times \chi$ to obtain the tensor $B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$ without computing the tensor $C^{[n]j_n}_{\alpha_n \alpha_{n+1}}$. As predicted in sec. 3.3 we observe a quadratic scaling in the local Hilbert space dimension d for the QR and QR+CBE based truncation method instead of a cubic scaling as for the SVD based truncation method. Also, as expected, we observe a cubic scaling in the bond dimension χ for all truncation methods. Furthermore, we can clearly observe an additional speedup of the QR and QR+CBE based method by performing the simulation on the GPU. In contrast to that, the SVD based truncation method does not experience a significant speedup on the GPU. Besides that, we can see that the additional computational cost of the QR+CBE compared to the QR method is higher on GPU than on CPU. This suggests that the used spectral-decomposition does not efficiently run on GPU. We see that this effect weakens for large d, as indicated by the converging data points for the benchmark for d. This is expected since the scaling $\chi^3 d^2$ of the efficient QR decomposition at some point dominates the scaling χ^3 of the less efficient spectral-decomposition (see eq. (3.8)). We conclude that the QR decomposition based trunation methods scale as expected with $\chi^3 d^2$ instead of $\chi^3 d^3$ as for the SVD based truncation method and that the scaling of the time evolution algorithm is indeed limited by the truncation schemes. Moreover, the QR decomposition based truncation methods run efficiently on the GPU.

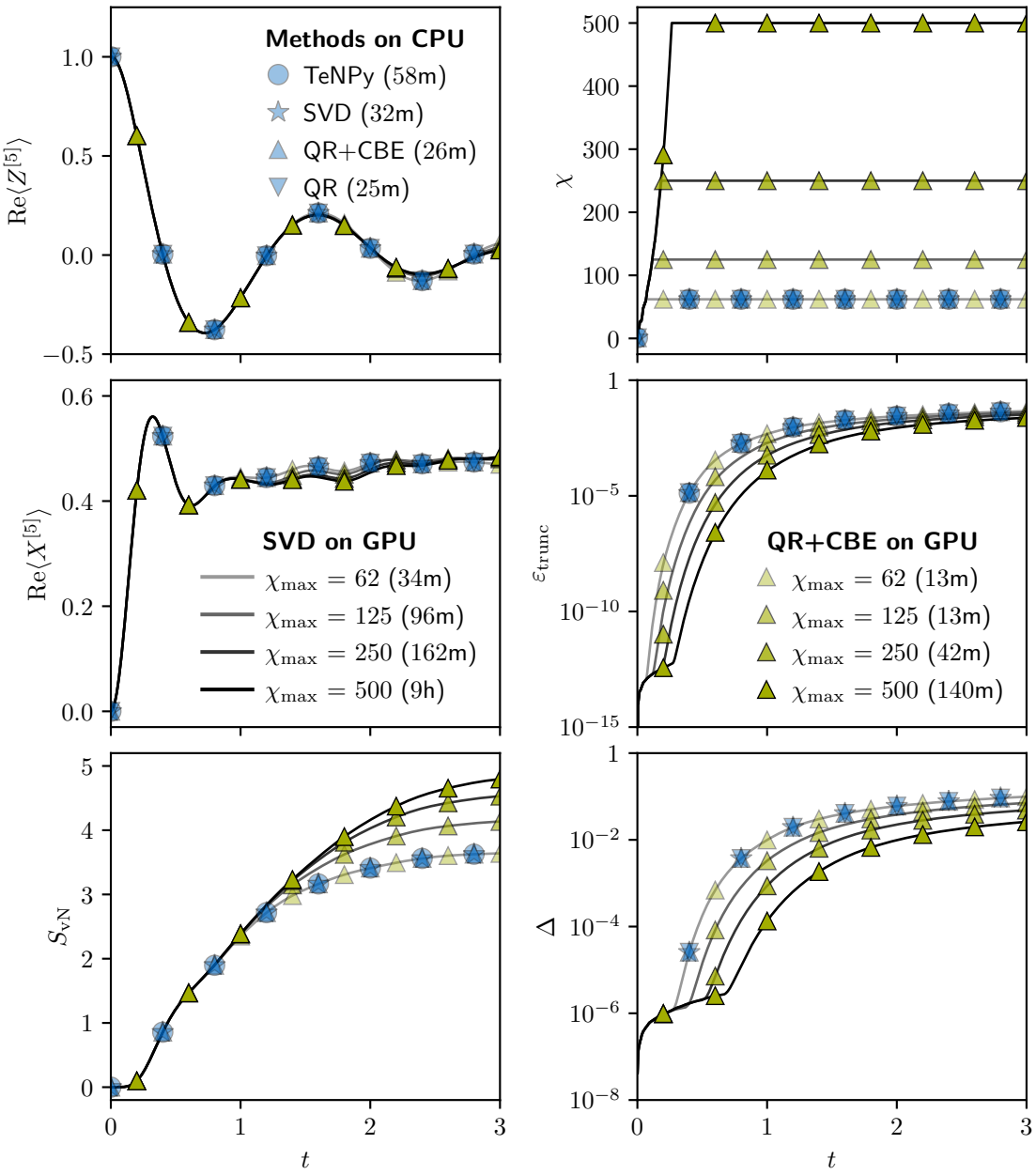


Figure 6: Time evolution with $\Delta t = 0.001$ of a global quench in the $d = 5$ quantum clock model with the parameters $J_1 = J_2 = 1$ and $g = 3$ of a system with $N = 10$ sites. We start with the $Z = 1$ product state at $t = 0$. Shown are the local Z and X expectation values (top and middle left), the half chain entanglement entropy S_{vN} (bottom left), the maximum bond dimension in the system χ (top right), the propagated largest truncation error $\varepsilon_{\mathrm{trunc}}$ (middle right) and the propagated distance Δ (bottom right). Every 400th datapoint is plotted. We perform $N_{\mathrm{sw}} = 1$ sweep for all simulations and limit the bond dimension by $\chi_{\max}$. For the TenPy simulation, the SVD and QR+CBE method, we discard all Schmidt values smaller than 10^{-14}. The time in the legend denotes the total wall time for each method, which includes computing the shown data.

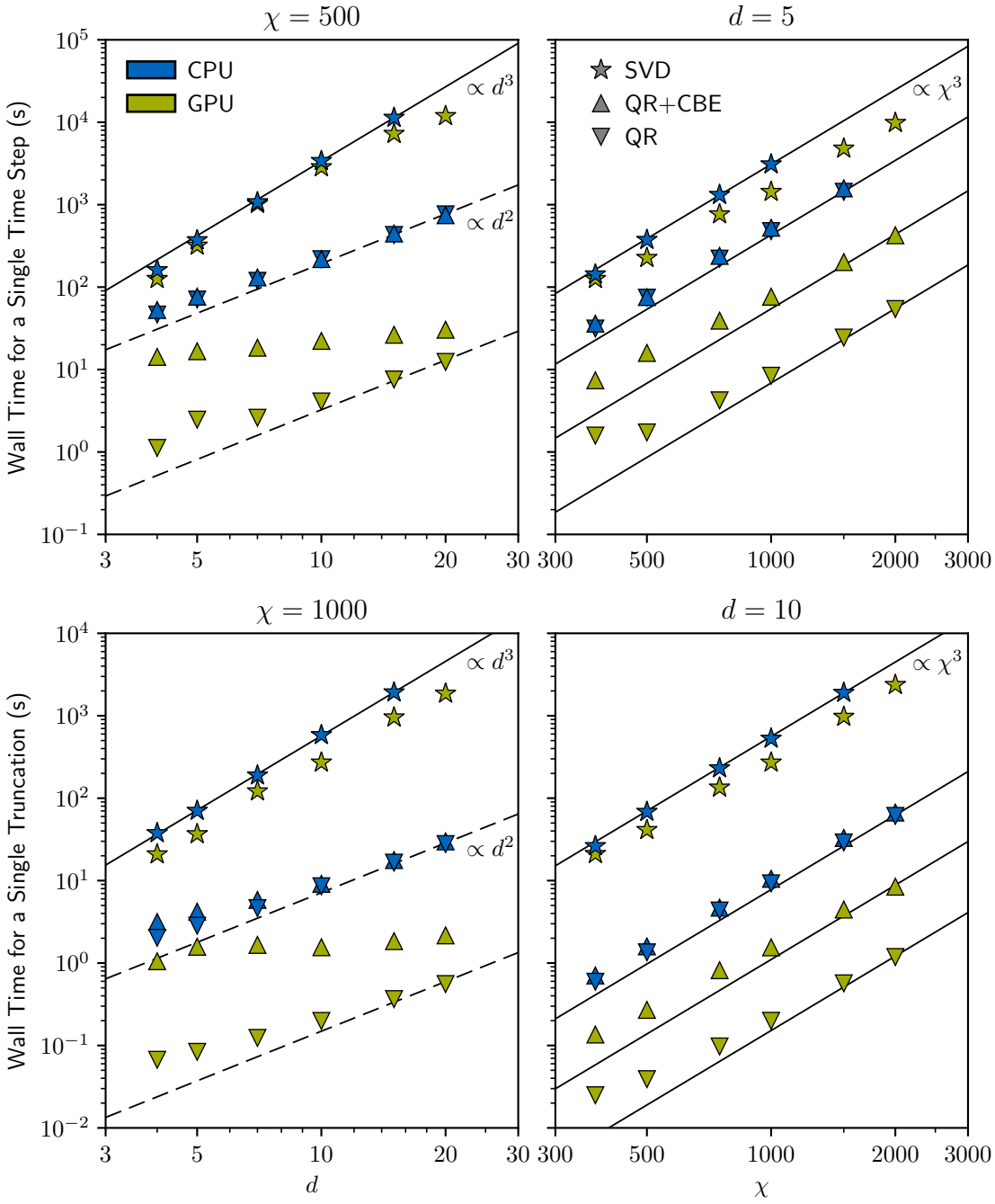


Figure 7: Timing benchmark for a single update of the MPS, as described in sec. 3.1, from $|\psi_t\rangle$ to $|\phi_{t+\Delta t}\rangle \approx |\psi_{t+\Delta t}\rangle$ (top) and for the truncation of a single two-site tensor $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$, as described in sec. 3.2, to obtain the tensor $B^{[n+1]j_{n+1}}_{\alpha_{n+1}\alpha_{n+2}}$ (bottom) for different hardware (marker color) and truncation schemes (marker shape). We perform the benchmark for the local Hilbert space dimension d (left) and the bond dimension χ (right). For the single update of the MPS, we compute a single time step with $N_{\mathrm{sw}} = 1$ sweep of a system with $N = 30$ sites where $2/3$ of the bond dimensions χ_n in the system have a bond dimension of $\chi_n = \chi$ for $d \geq 4$ and $\chi \leq 1024$ or $d \geq 5$ and $\chi \leq 3125$. For the single truncation, we perform an isolated truncation of an extracted two-site tensor $\theta^{[n]j_n j_{n+1}}_{\alpha_n \alpha_{n+2}}$ with size $d \times d \times \chi \times \chi$. The presented time is the total wall for a single time step or a single truncation where only the necessary operations are performed. This means no additional data like the truncation error $\varepsilon_{\mathrm{trunc}}$ or the distance Δ is computed. The missing data points of the CPU were not obtainable due to hardware limitations.

5. Conclusion

We applied the QR decomposition based truncation method for the TEBD [11] to the MPO based time evolution [12] for one-dimensional systems. This was achieved by replacing the SVD based truncation method of the MPO based time evolution algorithm with a variational QR decomposition based truncation scheme. We demonstrated that we are able to extract the same data with similar accuracy but with a scaling of $\chi^3 d^2$ instead of $\chi^3 d^3$ as for the SVD based truncation method. Additionally, we showed that in contrast to the SVD based truncation method, the QR decomposition based truncation schemes run efficiently on GPU, which resulted in an additional, hardware-dependent speedup.

The introduced algorithm could be modified by updating a single-site wavefunction instead of a two-site wavefunction. The update rule for the single-site wavefunction would be analog to the one shown in Fig. 3(b). A sweep could be performed by updating the tensor at the respective site with the semi-unitary tensor obtained by the QR decomposition of the single-site wavefunction. However, the bond dimension of the system would not be able to grow this way. One would need to perform a bond expansion of the single-site wavefunction prior to the QR decomposition. This might be doable with a subspace expansion similar to the one introduced in the single-site DMRG [37]. While one single-site sweep is probably cheaper than one two-site sweep, obtaining a well-approximated evolved state might cost more. It would need to be investigated if the additional cost of getting a better approximation of the single-site update outweighs the cost of the more expensive, faster converging two-site update.

Code availability: The object-based library is available on GitHub [38].

List of Figures

1 Diagrammatic notation of tensors networks and tensor network operations. . . 8

2 Diagrammatic notation of the MPS and MPO formalism. 8

3 Description of the time evolution algorithm in the diagrammatic notation. . . . 14

4 Description of the truncation methods in the diagrammatic notation. 15

5 Distance $\Delta = \Delta_0 + \delta\Delta$ of a time evolution of a global quench in the $d = 5$ quantum clock model. 22

6 Result benchmark of a time evolution of a global quench in the $d = 5$ quantum clock model. 25

7 Timing benchmark of the time evolution algorithm and the truncation methods. 26

Abbreviations

CBE Controlled Bond Expansion
CPU Central Processing Unit
DMRG Density Matrix Renormalization Group
GPU Graphics Processing Unit
MPO Matrix-Product Operator
MPS Matrix-Product State
SVD Singular Value Decomposition
TEBD Time Evolving Block Decimation
TPU Tensor Processing Unit

Terms

hermitian $M \in \mathbb{C}^{m\times n}$ is hermitian if $m = n$ and $M = M^\dagger$. Every eigenvalue of a hermitian operator is real. (see [39] 7.11 and 7.13)

invertible $M \in \mathbb{C}^{m\times m}$ is invertible if M has full rank. (see [39] 3.69, 3.117 and 3.20)

rank The rank of a $M \in \mathbb{C}^{m\times n}$ matrix is the dimension of the span of the rows (columns) of M in $\mathbb{C}^n$ ($\mathbb{C}^m$). (see [39] 3.115 and 3.118)

semi-unitary We call $M \in \mathbb{C}^{m\times n}$ semi-unitary if $m > n$ and $M^\dagger M = \mathbb{1}_n$ or if $m < n$ and $MM^\dagger = \mathbb{1}_m$.

singular value The singular values of $M \in \mathbb{C}^{m\times n}$ are the nonnegative square roots of the eigenvalues of $M^\dagger M$. (see [39] 7.52 and [19] def. 7.43)

span The span of $v_1, \ldots, v_m$ vectors in $\mathbb{C}^n$ is defined as $\mathrm{span}(v_1, \ldots, v_m) = \{a_1 v_1 + \cdots + a_m v_m \,|\, a_1, \ldots, a_m \in \mathbb{C}\}$. (see [39] 2.5)

spectral-decomposition A hermitian matrix $M \in \mathbb{C}^{n\times n}$ has the spectral decomposition $M = USU^\dagger$, where $U \in \mathbb{C}^{n\times n}$ is a unitary matrix and $S \in \mathbb{C}^{n\times n}$ is a diagonal matrix containing the eigenvalues of M. (see [18] cor. 2.20)

unitary $M \in \mathbb{C}^{m\times n}$ is unitary if $m = n$ and $MM^\dagger = M^\dagger M = \mathbb{1}$ and thus $M^\dagger = M^{-1}$. An equivalent condition is that M preserves norms. (see [39] 7.37 and 7.42)

References

[1] Steven R. White. “Density matrix formulation for quantum renormalization groups”. In: *Phys. Rev. Lett.* 69 (19 Nov. 1992), pp. 2863–2866. DOI: 10.1103/PhysRevLett.69.2863.

[2] Guifré Vidal. “Efficient Classical Simulation of Slightly Entangled Quantum Computations”. In: *Phys. Rev. Lett.* 91 (14 Oct. 2003), p. 147902. DOI: 10.1103/PhysRevLett.91.147902.

[3] Steven R. White and Adrian E. Feiguin. “Real-Time Evolution Using the Density Matrix Renormalization Group”. In: *Phys. Rev. Lett.* 93 (7 Aug. 2004), p. 076401. DOI: 10.1103/PhysRevLett.93.076401.

[4] Andrew John Daley et al. “Time-dependent density-matrix renormalization-group using adaptive effective Hilbert spaces”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2004.04 (2004), P04005. DOI: 10.1088/1742-5468/2004/04/P04005.

[5] Matthias Gohlke et al. “Dynamics of the Kitaev-Heisenberg Model”. In: *Phys. Rev. Lett.* 119 (15 Oct. 2017), p. 157203. DOI: 10.1103/PhysRevLett.119.157203.

[6] Weitang Li, Jiajun Ren, and Zhigang Shuai. “Numerical assessment for accuracy and GPU acceleration of TD-DMRG time evolution schemes”. In: *The Journal of Chemical Physics* 152.2 (Jan. 2020), p. 024127. ISSN: 0021-9606. DOI: 10.1063/1.5135363.

[7] Feng Pan and Pan Zhang. “Simulation of Quantum Circuits Using the Big-Batch Tensor Network Method”. In: *Phys. Rev. Lett.* 128 (3 Jan. 2022), p. 030501. DOI: 10.1103/PhysRevLett.128.030501.

[8] Markus Hauru et al. *Simulation of quantum physics with Tensor Processing Units: brute-force computation of ground states and time evolution*. 2021. arXiv: 2111.10466 [quant-ph].

[9] Alan Morningstar et al. “Simulation of Quantum Many-Body Dynamics with Tensor Processing Units: Floquet Prethermalization”. In: *PRX Quantum* 3 (2 May 2022), p. 020331. DOI: 10.1103/PRXQuantum.3.020331.

[10] Martin Ganahl et al. “Density Matrix Renormalization Group with Tensor Processing Units”. In: *PRX Quantum* 4.1 (Feb. 2023). ISSN: 2691-3399. DOI: 10.1103/prxquantum.4.010317.

[11] Jakob Unfried, Johannes Hauschild, and Frank Pollmann. “Fast time evolution of matrix product states using the QR decomposition”. In: *Phys. Rev. B* 107 (15 Apr. 2023), p. 155133. DOI: 10.1103/PhysRevB.107.155133.

[12] Michael P. Zaletel et al. “Time-evolving a matrix product state with long-ranged interactions”. In: *Physical Review B* 91.16 (Apr. 2015). DOI: 10.1103/physrevb.91.165112.

[13] Johannes Hauschild and Frank Pollmann. “Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy)”. In: *SciPost Phys. Lect. Notes* (2018). Code available from https://github.com/tenpy/tenpy, p. 5. DOI: 10.21468/SciPostPhysLectNotes.5.

[14] J. J. Sakurai and Jim Napolitano. *Modern Quantum Mechanics*. 3rd ed. Cambridge University Press, 2020. DOI: 10.1017/9781108587280.

[15] IBM. *What is supercomputing?* https://www.ibm.com/topics/supercomputing. [Online; accessed 27-October-2023].

[16] David Poulin et al. “Quantum Simulation of Time-Dependent Hamiltonians and the Convenient Illusion of Hilbert Space”. In: *Phys. Rev. Lett.* 106 (17 Apr. 2011), p. 170501. DOI: 10.1103/PhysRevLett.106.170501.

[17] Tim Baumann. *Image Compression with Singular Value Decomposition*. http://timbaumann.info/svd-image-compression-demo/. [Online; accessed 09-December-2023].

[18] Zhong-Zhi Bai and Jian-Yu Pan. “Chapter 2: Matrix Decompositions”. In: *Matrix Analysis and Computations*. SIAM, 2021, pp. 45–75. DOI: 10.1137/1.9781611976632.ch2.

[19] M. Thamban Nair and Arindama Singh. *Linear Algebra*. 1st ed. Springer, 2018. DOI: 10.1007/978-981-13-0926-7.

[20] Ulrich Schollwöck. “The density-matrix renormalization group in the age of matrix product states”. In: *Annals of Physics* 326.1 (Jan. 2011), pp. 96–192. DOI: 10.1016/j.aop.2010.09.012.

[21] Charles R. Harris et al. “Array programming with NumPy”. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[22] Mou-Hsiung Chang. *Theory of Quantum Information with Memory*. Berlin, Boston: De Gruyter, 2022. ISBN: 9783110788105. DOI: 10.1515/9783110788105.

[23] Don N. Page. “Average entropy of a subsystem”. In: *Physical Review Letters* 71.9 (Aug. 1993), pp. 1291–1294. DOI: 10.1103/physrevlett.71.1291.

[24] Siddhartha Sen. “Average Entropy of a Quantum Subsystem”. In: *Physical Review Letters* 77.1 (July 1996), pp. 1–3. DOI: 10.1103/physrevlett.77.1.

[25] Jens Eisert, Marcus Cramer, and Martin B Plenio. “Colloquium: Area laws for the entanglement entropy”. In: *Reviews of modern physics* 82.1 (2010), p. 277. DOI: 10.1103/RevModPhys.82.277.

[26] M B Hastings. “An area law for one-dimensional quantum systems”. In: *Journal of Statistical Mechanics: Theory and Experiment* 2007.08 (Aug. 2007), P08024–P08024. DOI: 10.1088/1742-5468/2007/08/p08024.

[27] Román Orús. “A practical introduction to tensor networks: Matrix product states and projected entangled pair states”. In: *Annals of Physics* 349 (2014), pp. 117–158. ISSN: 0003-4916. DOI: https://doi.org/10.1016/j.aop.2014.06.013.

[28] Jacob Biamonte and Ville Bergholm. “Tensor Networks in a Nutshell”. In: (2017). DOI: 10.48550/arXiv.1708.00006.

[29] Simone Montangero. *Introduction to Tensor Network Methods*. Springer Nature Switzerland AG, 2018. DOI: 10.1007/978-3-030-01409-4.

[30] Mark Fannes, Bruno Nachtergaele, and Reinhard F Werner. “Finitely correlated states on quantum spin chains”. In: *Communications in mathematical physics* 144 (1992), pp. 443–490. DOI: 10.1007/BF02099178.

[31] Stellan Östlund and Stefan Rommer. “Thermodynamic Limit of Density Matrix Renormalization”. In: *Physical Review Letters* 75.19 (Nov. 1995), pp. 3537–3540. DOI: 10.1103/physrevlett.75.3537.

[32] Stefan Rommer and Stellan Östlund. “Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group”. In: *Phys. Rev. B* 55 (4 Jan. 1997), pp. 2164–2181. DOI: 10.1103/PhysRevB.55.2164.

[33] G. Vidal et al. “Entanglement in Quantum Critical Phenomena”. In: *Phys. Rev. Lett.* 90 (22 June 2003), p. 227902. DOI: 10.1103/PhysRevLett.90.227902.

[34] G. Vidal. “Classical Simulation of Infinite-Size Quantum Lattice Systems in One Spatial Dimension”. In: *Phys. Rev. Lett.* 98 (7 Feb. 2007), p. 070201. DOI: 10.1103/PhysRevLett.98.070201.

[35] N Schuch et al. “On entropy growth and the hardness of simulating time evolution”. In: *New Journal of Physics* 10.3 (Mar. 2008), p. 033032. DOI: 10.1088/1367-2630/10/3/033032.

[36] G. Sun et al. “Phase transitions in the $\mathbb{Z}_p$ and U(1) clock models”. In: *Phys. Rev. B* 100 (9 Sept. 2019), p. 094428. DOI: 10.1103/PhysRevB.100.094428.

[37] C. Hubig et al. “Strictly single-site DMRG algorithm with subspace expansion”. In: *Phys. Rev. B* 91 (15 Apr. 2015), p. 155115. DOI: 10.1103/PhysRevB.91.155115. URL: https://link.aps.org/doi/10.1103/PhysRevB.91.155115.

[38] Martin Hefel. “Fast-Matrix-Product-Operator-Based-Time-Evolution”. In: *GitHub* (Dec. 2023). URL: https://github.com/MartinHefel/Fast-Matrix-Product-Operator-Based-Time-Evolution.

[39] Sheldon Axler. *Linear algebra done right*. 3rd ed. Springer, 2015. DOI: 10.1007/978-3-319-11080-6.

[40] Adam Paszke et al. “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Appendix

The appendix can be found on the following pages.

A. Optaining the MPS of a generic pure state

To obtain the MPS of a generic pure state $|\psi\rangle$ with d^N amplitudes $\psi_{j_1 j_2 \ldots j_N}$ (see eq. 2.1), we proceed as instruceted by Ref. [20] sec. 4.1.3.: We start by writing all d^N amplitudes in a $1 \times d^N$ matrix and reshape it into a $d \times d^{N-1}$ matrix ψ, where the coefficients map as: $\psi_{j_1(j_2 \ldots j_N)} = \psi_{j_1 j_2 \ldots j_N}$. This bipartites our system into two subsystems. We acquire the Schmidt decomposition by performing a SVD of $\psi_{j_1(j_2 \ldots j_N)}$:

$$\psi_{j_1(j_2 \ldots j_N)} = \sum_{\alpha_2}^{\chi_2} U_{j_1 \alpha_2} \Lambda_{\alpha_2 \alpha_2} (V^\dagger)_{\alpha_2(j_2, \ldots j_N)} = \sum_{\alpha_2}^{\chi_2} U_{j_1 \alpha_2} \psi_{\alpha_2(j_2 \ldots j_N)} \tag{1}$$

where $U_{j_1 \alpha_2}$, $\Lambda_{\alpha_2 \alpha_2}$ and $(V^\dagger)_{\alpha_2(j_2 \ldots j_N)}$ are the matrices obtained by the SVD as introduced in sec. 2.2.1 and $\chi_2 \leq d$ is the rank of the matrix $\psi_{j_1(j_2 \ldots j_N)}$. Moreover we define the matrix $\psi_{\alpha_2(j_2 \ldots j_N)} = \Lambda_{\alpha_2 \alpha_2} (V^\dagger)_{\alpha_2(j_2 \ldots j_N)}$. We proceed by writing the d column vectors of $U_{j_1 \alpha_2}$ with dimension χ_2 in a collection of d vectors $A^{[1] j_1}_{\alpha_2}$ and reshape $\psi_{\alpha_2(j_2 \ldots j_N)}$ in a $\chi_2 d \times d^{N-2}$ matrix $\psi_{(\alpha_2 j_2)(j_3 \ldots j_N)}$ and thus

$$\psi_{j_1 j_2 \ldots j_N} = \psi_{j_1(j_2 \ldots j_N)} = \sum_{\alpha_2}^{\chi_2} A^{[1] j_1}_{\alpha_2} \psi_{(\alpha_2 j_2)(j_3 \ldots j_N)}. \tag{2}$$

Analogously to eq. 1 we perform a SVD of $\psi_{(\alpha_2 j_2)(j_3 \ldots j_N)}$:

$$\begin{aligned} \psi_{(\alpha_2 j_2)(j_3 \ldots j_N)} &= \sum_{\alpha_2}^{\chi_2} \sum_{\alpha_3}^{\chi_3} A^{[1] j_1}_{\alpha_2} U_{(j_1 \alpha_2)\alpha_3} \Lambda_{\alpha_3 \alpha_3} (V^\dagger)_{\alpha_3(j_3 \ldots j_N)} \\ &= \sum_{\alpha_2}^{\chi_2} \sum_{\alpha_3}^{\chi_3} A^{[1] j_1}_{\alpha_2} A^{[2] j_2}_{\alpha_2 \alpha_3} \psi_{(\alpha_3 j_3)(j_4 \ldots j_N)} \end{aligned} \tag{3}$$

where $A^{[2] j_2}_{\alpha_2 \alpha_3}$ is a collection of d matrices with the dimention $\chi_2 \times \chi_3$, where $\chi_3 \leq \chi_2 d \leq d^2$. By proceeding in this manner until we reach the last site, we obtain

$$\psi_{j_1 j_2 \ldots j_N} = \sum_{\alpha_2, \ldots, \alpha_N} A^{[1] j_1}_{\alpha_2} A^{[2] j_2}_{\alpha_2 \alpha_3} \ldots A^{[N-1] j_{N-1}}_{\alpha_{N-1} \alpha_N} A^{[N] j_N}_{\alpha_N}. \tag{4}$$

To be consistent with the representation in eq. 2.18 we transform the χ_2 dimensional vector $A^{[1] j_1}_{\alpha_1}$ into a $1 \times \chi_2$ matrix $A^{[1] j_1}_{\alpha_1 \alpha_2}$ (equivalent for $A^{[N] j_N}_{\alpha_N}$) and obtain the MPS of the state as

$$\begin{aligned} |\psi\rangle &= \sum_{j_1, j_2, \ldots, j_N} \sum_{\alpha_2, \ldots, \alpha_N} A^{[1] j_1}_{\alpha_1 \alpha_2} A^{[2] j_2}_{\alpha_2 \alpha_3} \ldots A^{[N] j_N}_{\alpha_N \alpha_{N+1}} |j_1, j_2, \ldots, j_N\rangle \\ &= \sum_{j_1, j_2, \ldots, j_N} A^{[1] j_1} A^{[2] j_2} \ldots A^{[N] j_N} |j_1, j_2, \ldots, j_N\rangle \end{aligned} \tag{5}$$

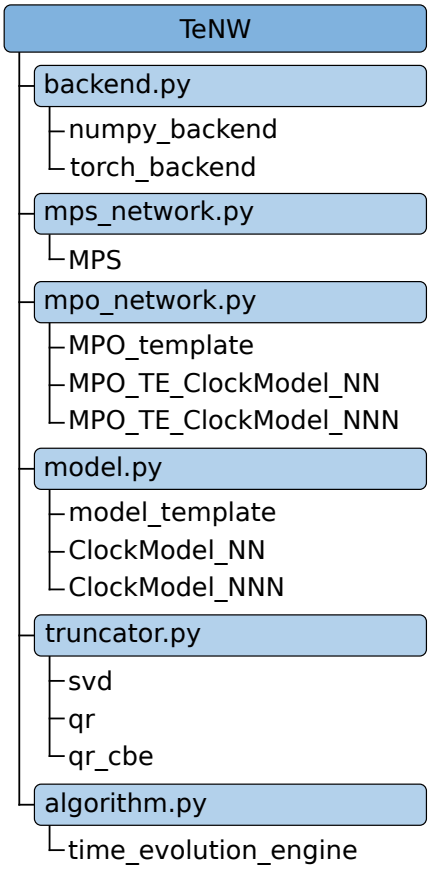
B. Information about the developed Python library

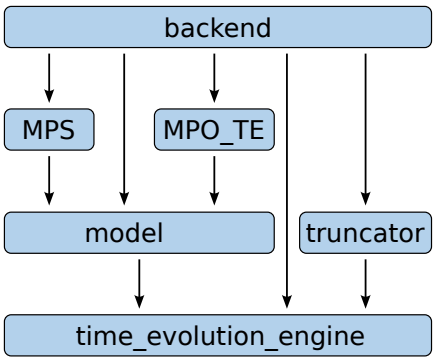
The developed object-based Python library can perform a MPO based time evolution of a MPS with three different truncation methods (SVD, QR and QR+CBE) on two different backends (NumPy [21] and PyTorch [40]). The library and a script showing how to perform a time evolution for a certain set of parameters are available on GitHub [38].

The used assumptions of the library are the following: The system is one-dimensional and finite with N sites. There are no periodic boundary conditions between the last and the first site. The local Hilbert space dimension d as well as the MPO dimension D are identical for each site.

In general, models that compile with the assumptions above can be implemented by defining the MPO in *mpo_network.py* and referencing it in the newly created corresponding model in *model.py*. Object templates show how to define a time evolution MPO in *mpo_network.py* and a corresponding model in *model.py*. The implemented models in *model.py* are the quantum clock model for nearest-neighbor interactions (*ClockModel_NN*) and nearest- and next-nearest-neighbor interactions (*ClockModel_NNN*).

The Image below illustrates the object structure of the library (left) and the relation between the objects (right). Every object can be used and tested separately, which is why every object relies on the backend. The backends, defined in *backend.py*, are a “dictionary” that directs a generalized function call, like “perform the QR decomposition of matrix M” to the respective library (NumPy or PyTorch) in the proper data format. The backend used to run calculations on the GPU is PyTorch, which relies on toolkits like cuda or ROCm. Consequently, Nvidia or AMD graphics cards are needed to run the backend on GPU. The MPS and MPO networkork are defined in *mps_network.py* and *mpo_network.py*. The MPS class features many MPS related functions, for example, calculating the entanglement entropy. The file *mpo_network.py* contains a collection of MPOs which are needed for other objects, for example, the models defined in *model.py*. The three truncation methods (*svd*, *qr* and *qr_cbe*) are defined in *truncator.py*. To perform a time evolution, one needs to use the *time_evolution_engine* in *algorithm.py*, which needs a *truncator*, a *backend* to run on and a *model* to evolve.


TeNW
backend.py
numpy_backend
torch_backend
mps_network.py
MPS
mpo_network.py
MPO_template
MPO_TE_ClockModel_NN
MPO_TE_ClockModel_NNN
model.py
model_template
ClockModel_NN
ClockModel_NNN
truncator.py
svd
qr
qr_cbe
algorithm.py
time_evolution_engine


backend
MPS
MPO_TE
model
truncator
time_evolution_engine