Fakultät für Informatik
Technische Universität München

Technische Universität München
Lehrstuhl für Wissenschaftliches Rechnen

Tensor networks and machine learning for approximating and optimizing functions in quantum physics

Moritz M. August

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:
Prof. Dr. Hans Michael Gerndt

Prüfende der Dissertation:
1. Prof. Dr. Thomas Huckle
2. Prof. Dr. Steffen J. Glaser
3. Prof. José Miguel Hernández-Lobato, Ph. D.
   Universität Cambridge, Vereinigtes Königreich

Die Dissertation wurde am 24.05.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 08.10.2018 angenommen.

Abstract

In this dissertation we explore the intersection of computer science and mathematics to address challenging problems in numerical quantum physics. We introduce, analyze and evaluate novel methods for the approximation of physical quantities of interest as well as the optimization of performance criteria in quantum control. These methods are based on techniques from the fields of tensor networks, numerical analysis and machine learning. Furthermore, we present work on the relation between machine learning and tensor network methods for the representation of quantum states.

We introduce a general algorithm which for the first time allows to approximate global functions $\mathsf{Tr} f(A)$ of matrix product operators A which represent Hermitian matrices of very high dimensionality. Following this, we present an analytical analysis of the partial results computed by the procedure. This analysis leads us to the discovery of a more efficient variant of the algorithm and we subsequently show that it can be applied to a large class of spin Hamiltonians in quantum physics. We finally demonstrate how our method yields a novel strategy to approximate properties of thermal equilibrium states, some of which were so far inaccessible for numerical methods.

In the second part, we present a novel and broadly applicable method for solving quantum control scenarios. The method employs a particular class of recurrent neural networks, the long short-term memory network, to probabilistically model control sequences and optimize these models with tools from supervised and reinforcement learning. In a first version, we use an optimization procedure inspired by evolutionary algorithms to train the networks. We demonstrate in a quantum memory setting that the method can produce better results than certain analytical solutions. We then improve on these results by introducing a different optimization strategy based on insights from reinforcement learning known as policy gradient algorithms. The combination of long short-term memory networks and policy gradient optimization schemes allows us to tackle a wide variety of control problems, which we demonstrate numerically.

Finally, we show results on the relation between tensor networks and a particular class of machine learning models, the restricted Boltzmann machine. We find that restricted Boltzmann machines can be generalized in the tensor network framework and gain insight about their efficiency in representing states of many-body quantum systems.

Zusammenfassung

In dieser Dissertation erkunden wir die Schnittstelle von Informatik und Mathematik, um herausfordernde Probleme in der numerischen Quantenphysik zu adressieren. Wir präsentieren, analysieren und evaluieren neuartige Methoden zur Approximation physikalischer Eigenschaften sowie der Optimierung von Optimalitätskriterien in der Quantenkontrolle. Diese Methoden basieren dabei auf Techniken aus den Gebieten der Tensornetzwerke, der numerischen Analysis und des maschinellen Lernens. Darüber hinaus präsentieren wir Arbeit zur Verbindung zwischen Methoden des maschinellen Lernens und der Tensornetzwerke im Rahmen der Approximation von Quantenzuständen.

Wir stellen einen allgemeinen Algorithmus vor, der zum ersten Mal die Approximation von globalen Funktionen der Form $\mathsf{Tr} f(A)$ ermöglicht. Dabei ist A hier ein Matrixproduktoperator, der hermitische Matrizen sehr hoher Dimensionalität repräsentiert. Daran anknüpfend präsentieren wir eine analytische Untersuchung der vom Algorithmus berechneten Teilergebnisse. Diese Analyse führt uns zu der Entdeckung einer effizienteren Variante des Algorithmus und wir zeigen im Folgenden, dass diese auf eine große Klasse quantenmechanischer Systeme anwendbar ist. Schlussendlich demonstrieren wir, wie unsere Methode eine neue Strategie zur Approximation von physikalischen Eigenschaften thermaler Gleichgewichtszustände zulässt. Einige dieser Eigenschaften waren dabei zuvor unzugänglich für numerische Methoden.

Im zweiten Teil stellen wir eine neue und vielfältig anwendbare Methode zur Lösung von Quantenkontrollszenarien vor. Diese Methode nutzt eine bestimmte Klasse von rekurrenten neuronalen Netzwerken, den Langes-Kurzzeitgedächtnis-Netzwerken, um Kontrollsequenzen probabilistisch zu modellieren. Die Modelle werden dabei mit Werkzeugen aus dem überwachten und bestärkenden maschinellen Lernen optimiert. In einer ersten Version nutzen wir eine von evolutionären Algorithmen inspirierte Optimierungsprozedur, um diese Modelle zu trainieren. Wir demonstrieren in einem Quantenspeicherszenario, dass die Methode bessere Ergebnisse als bestimmte analytische Lösungen erreichen kann. Daraufhin verbessern wir die Methode durch die Einführung einer anderen Optimierungsstrategie, welche auf als Vorschriftsgradientenmethoden bekannten Erkenntnissen des bestärkenden Lernens basiert. Die Kombination von Langes-Kurzzeitgedächtnis-Netzwerken und Vorschriftsgradienten-Optimierungsmethoden erlaubt es uns, eine hohe Bandbreite von Kontrollproblemen zu adressieren. Dies demonstrieren wir numerisch.

Zum Schluss präsentieren wir Ergebnisse zu der Beziehung zwischen Tensornetzwerken und einer bestimmten Klasse von Modellen des maschinellen Lernens, den beschränkten Boltzmann-Maschinen. Wir finden heraus, dass beschränkte Boltzmann-Maschinen im Tensornetzwerkrahmenwerk generalisiert werden können und gewinnen Einsichten bezüglich ihrer Effizienz in der Repräsentation der Zustände von Vielteilchen-Quantensystemen.

List of Publications

This thesis is based on the following publications of which 1. and 4. constitute the formal basis of the publication-based form.

1. M. August, M.C. Bañuls and T. Huckle. On the approximation of functionals of very large Hermitian matrices represented as matrix product operators. *Electronic Transactions on Numerical Analysis*, 46:215-232, 2017

2. M. August and T. Huckle. Towards a better understanding of the matrix product function approximation algorithm in application to quantum physics. *arXiv:1709.06847v2 [cs.NA]*, 2018

3. M. August and M.C. Bañuls. Efficient approximation for global functions of matrix product operators. *arXiv:1804.03613v2 [quant-ph]*, 2018

4. M. August and X. Ni. Using recurrent neural networks to optimize dynamical decoupling for quantum memory. *Physical Review A*, 95(1):012335, 2017

5. M. August and J.M. Hernández-Lobato. Taking gradients through experiments: LSTMs and memory proximal policy optimization for black-box quantum control. *arXiv:1802.04063v2 [cs.LG]*, 2018

6. I. Glasser, N. Pancotti, M. August, I. D. Rodriguez and J. I. Cirac. Neural-network quantum states, string-bond states and chiral topological states. *Physical Review X*, 8(1):011006, 2018

Contents

List of Figures

1 Introduction

But they are useless! They can only give you answers.

- Pablo Picasso on computers in 1964

Although the second part of the above statement remains correct in essence even more than 50 years after it was originally made, this has not stopped computers from deeply affecting almost every aspect of human society and inspiring many profound questions since then. Their ability to 'answer questions' has indeed been put to use with great success in areas ranging from industrial manufacturing over communication, medicine and transportation to politics, to only name a few. Even art and computers have started to engage in a symbiotic relationship. But most importantly, the computational paradigm has permeated science, especially the natural and formal sciences, and proved to be an indispensable enabler of insight in the face of ever more complex and numerous data generated by experiments and predictions made by theory. It is thus considered a fact that scientific computing and data science have established themselves as the third pillar of science, next to the traditional theory and experiment.

Among the various fields of science, quantum machanics (QM) can safely be regarded as one of the most challenging and fascinating disciplines as its underlying phenomena are both utterly alien to the human mind, evolved to understand Newtonian mechanics, and at the same time among the most accurately demonstrated in all of physics. Even more so, quantum physics poses a great challenge for computational scientists. The main reason for this is that the amount of information needed to be stored and processed to describe a quantum system grows exponentially with the number of particles in the system. The required storage capacity and processing power thus exceeds the capabilities of even the most powerful computers available today by orders of magnitude already for systems of moderate size. To counter this quantum instantiation of the *curse of dimensionality*, computational methods thus need to represent and process quantum physical systems in a compressed form or be able to make accurate predictions about them based only on a fraction of the potentially observable information.

The approach of compressing the required information has in recent years been implemented very successfully by tensor network (TN) methods. TNs are based on the insight that it is possible to decompose the matrices and vectors describing quantum systems into sets of smaller tensors which, combined in a certain way, allow for the retrieval of the original information. These tensors generally each represent the information of an individual particle in the system. Their dimensionality in an exact representation is hereby determined by the inter-particle correlations in that system. Reducing the dimensionality of a tensor then corresponds to ignoring certain information for the benefit of a more efficient representation and constitutes a lossy compression. It has been found that TNs allow the accurate description of physically interesting classes of quantum states and some of their properties and dynamics with a number of parameters only *polynomial* in the number of particles. TNs thus provide a way to break or at least mitigate the curse of dimensionality for physically relevant problems, allowing them to be treated computationally. Hence, at the time of this writing TNs pose the most successful approach to numerically simulating many-body quantum systems.

Machine learning (ML) methods on the other hand have become an important numerical tool for problems of automated statistical inference, pattern recognition, optimal control, clustering and more but have so far not been used extensively in numerical QM. An ML problem can be

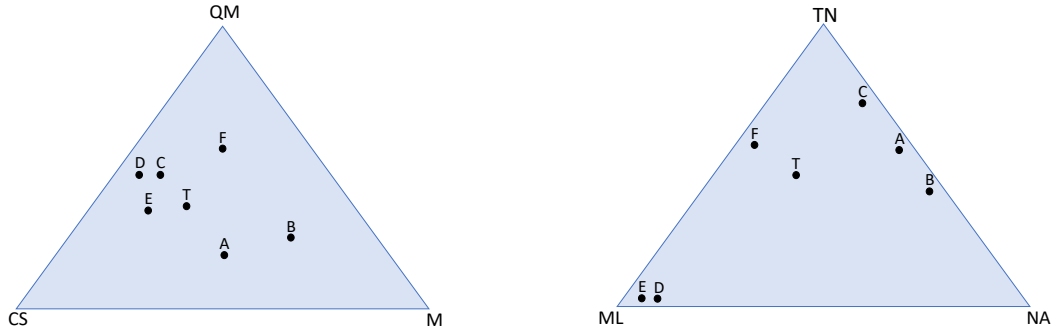
generally described by specifying an error function and a class of solution functions over which an optimization is performed based on available data. This training data is assumed to be obtained from a data-generating process or probability distribution. At the core of the solution to the ML problem then lies optimizing the function parameters given the training data such that the optimized parameters perform well for all potentially observable data generated by the same process. This property is commonly referred to as *generalization* and discriminates ML from pure optimization.

As we have seen above, an algorithm for quantum simulation must in general be able to compute accurate results from an at most polynomial subset of an exponential space of parameters or data in at most polynomial time. A similar problem can also be found in the related task of numerical quantum control. Here, the problem of simulating the time evolution of a physical system often representing an experiment is combined with an optimization over a space of sequences of parameters controlling the time evolution. This sequence space is at least exponential in the number of discrete time steps, independently of the size of the physical system whose simulation is challenging in itself for the above reasons. Despite this, the algorithm should be of at most polynomial complexity in the number of time steps and number of particles to be generally applicable. These considerations illustrate that a certain form of generalization is required for computationally feasible quantum simulation and control to perform well.

There is thus an alignment of the challenges in numerical QM with the problems of interest in ML which suggests that it is worthwhile to investigate ML as a promising candidate for tackling computational quantum problems. At the same time, the particular nature of quantum computation could potentially lead to the development of novel and possibly superior quantum ML (QML) algorithms. This mutually beneficial relationship has been recognized by a growing number of physicists and computer scientists in recent years, leading to the birth of the field of *quantum machine learning*. At the time of this writing the field is still in its infancy but shows great promise from our perspective.

As we have seen, obtaining a computationally feasible representation of a system or its state poses a major challenge in computational QM. However, such a representation usually is only the first step as one is often interested in approximating certain properties of a given system or state. One prominent example is the approximation of the lowest eigenvalue or ground state energy of a system which can be cast as a variational optimization problem. Many other properties of interest however take the form of a function of the system or its state that generally can not successfully be treated as an optimization problem. These include (properties of) thermal states in equilibrium or time evolved states and correlation functions as well as norms, distances and entropies. Although in certain cases symmetries can be exploited to project a system onto a space that is small enough to directly compute any function, this is not the case in general as not all physical systems exhibit such symmetries. For some of these functions TN approximation algorithms have been developed but many of them so far remain infeasible to approximate. This holds especially true for functions which are global in the sense that they take into account information about the entire system and not, for instance, only pairs of particles.

In addition to the approximation of functions of quantum states and systems, we have seen that in quantum control one needs to optimize certain parameters of a system's time evolution. This optimization over control sequences can for instance be implemented as the gradient-based optimization of the individual control parameters when an exact mathematical model of the quantum system is available and it is feasible to compute the related time evolution operators. While this approach works well for certain problem domains, it can not always be assumed that an exact analytical expression of the system is possible to obtain, especially for complex experimental setups. If in this case a simplified analytical model is employed for the



(a) Fields **(b)** Methods

Figure 1.1: Figure (a) illustrates the position of the individual publications (A-F, following the order of the appendix) contained in this thesis and the thesis (T) itself in the triangle between computer science (CS), math (M) and quantum mechanics (QM). In Figure (b), we show their location between machine learning (ML), numerical analysis (NA) and tensor networks (TN).

optimization or an analytical solution of such a model is used directly, the obtained solutions then likely are sub-optimal. In addition to this, the exact computation of the time evolution operators based on an analytical description of a system can only be done for small system sizes and so far it thus remains an open problem to perform numerical quantum control of many-body systems.

In this thesis, we aim to advance the state of the art in computational QM by introducing, analyzing and evaluating novel methods for the approximation and optimization of functions in quantum simulation and control. More concretely, in a first line of work we introduce the first TN method for the approximation of a general class of functions commonly encountered in QM. We provide an analytical analysis of it and discover a more efficient version of the method for particular inputs which we demonstrate to occur in QM. Finally, we demonstrate how the method can be applied to approximate several physically interesting quantities of particular systems to capture signals of thermal phase transitions and analyze inter-particle correlations. In a second line of work we explore the frontier of numerical QML. In this context we introduce a method for ML-based quantum control which relies on the probabilistic modelling of control sequences with an established class of recurrent neural networks. We present two variants of the method based on different training algorithms for these networks. We show how they can be used to solve relevant control problems and demonstrate that in certain cases they achieve better results than analytically derived solutions based on simplified assumptions. Finally, we present work on the connection between certain ML models and TNs which can both be used to approximate states of many-body systems. For our methods to be able to address the issues pointed out above, they rely on a minimized set of assumptions about the particular problems. As we demonstrate, the general methods can be easily combined with domain knowledge to tackle relevant problems which in some cases hitherto have not been possible to adress numerically at all.

We would like to point out that the work presented here is highly interdisciplinary as it combines methods from the areas of TNs, numerical analysis (NA) and ML to address relevant problems in numerical QM. From a different perspective, it is positioned between the fields of computer science, mathematics and quantum physics. In Figure 1.1(a) and (b), we illustrate the thematic orientation of this thesis and the individual works as shown in the appendix for the reader's information.

The scientific contribution of this thesis is three-fold. Firstly, by introducing novel numerical methods it broadens the range of problems that can be tackled numerically in QM. It also shows how to solve certain problems so far treated with tailored solutions by more general methods. Secondly, it contributes to establishing the field of QML by providing evidence that the automated optimization of quantum control parameters can be successfully tackled with ML techniques. Thirdly, this thesis permeates the walls of ignorance unfortunately often found even between related scientific disciplines by introducing concepts from computer science and numerical mathematics to the numerical QM community. It additionally tries to promote QM as an interesting numerical application.

The rest of this work is structured as follows: in Chapter 2, we will briefly introduce quantum physics and provide more rigorous explanations for the concepts more casually used above. This is followed by a short introduction to the relevant aspects of TNs in Chapter 3 and to Krylov algorithms and their connection to Gauss quadrature in Chapter 4. Chapter 5 introduces the relevant aspects of ML. We conclude with a discussion of the main results of this thesis against the background of related work in Chapter 6. The Appendices A-F finally show the publications this dissertation is based on.

2 Quantum physics

In this chapter we will provide a brief introduction to some of the core concepts of quantum physics. As befits the numerical nature of this thesis, we will introduce these concepts from a linear algebra perspective and assume systems to be of finite size. For a more thorough introduction to quantum physics we refer the reader to the standard literature [1, 2].

To be notationally compatible with QM literature, we will in the following make use of the Dirac notation. In our case then $|\psi\rangle$ denotes a complex column vector ψ, $\langle\psi|$ stands for its conjugate transpose and $\langle\psi|\psi'\rangle$ is the inner product of two vectors $|\psi\rangle$ and $|\psi'\rangle$. The outer product is consequentially denoted as $|\psi\rangle\langle\psi'|$.

2.1 Quantum states

We begin by defining the *state* or *wave function* $|\psi\rangle$ of a d-level particle s. In this thesis d will be referred to as the physical dimension. Now, $|\psi\rangle$ is an element of the d-dimensional complex Hilbert space $\mathcal{H}_d$ that is normalized such that it holds

$$\langle\psi|\psi\rangle = \|\psi\|^2 = 1. \tag{2.1}$$

This state can naturally also be written as a linear combination of *basis states*

$$|\psi\rangle = \sum_s \psi(s)\,|s\rangle \tag{2.2}$$

where the sum runs over all d basis states s can be in. The elements $\psi(s)$ of $|\psi\rangle$ are commonly referred to as the *amplitudes* of the state. Unless explicitly stated otherwise, we will in this thesis always assume $d = 2$ and s to be a spin-1/2 particle. Such 2-level particles also play an important role in quantum computation where they are called quantum bits or *qubits*. While most of the results of this thesis are independet of d, some results of the works in Appendices B and F assume $d = 2$.

Now, the next logical step is to define the state $|\psi\rangle$ of a system of L particles $s_1, s_2, \cdots, s_L$. As before, $|\psi\rangle$ is a normalized element of complex Hilbert space but now we have that

$$|\psi\rangle \in \mathcal{H}_1 \otimes \mathcal{H}_2 \otimes \cdots \otimes \mathcal{H}_L = \mathcal{H}^{\otimes L} \tag{2.3}$$

where the $\mathcal{H}_i$ are the local Hilbert spaces of the spin particles s_i, $\otimes$ denotes the Kronecker product and we have omitted the subscript d for the sake of clarity. This state can again be expressed in terms of its basis and the bases of the local Hilbert spaces as

$$\sum_{s_1,s_2,\cdots,s_L} \psi(s_1, s_2, \cdots, s_L)\,|s_1, s_2, \cdots, s_L\rangle \tag{2.4}$$

where $|s_1, s_2, \cdots, s_L\rangle = |s_1\rangle \otimes |s_2\rangle \otimes \cdots \otimes |s_L\rangle$ and the sum runs over all spin particles s_i. The above formulation allows us to briefly introduce three important concepts in quantum physics.

Firstly, we see that the state of a system with L spin particles is formally described by a complex vector of dimension 2^L. The *exponential growth of the Hilbert space* in the number of particles poses the main challenge for computational QM as even simply storing a quantum state

for systems with $L \gg 20$ is computationally infeasible. At the same time, many interesting phenomena in quantum physics can only be studied for relatively large systems as they in principle only occur in the thermodynamic limit $L \to \infty$. This necessitates the study of approximative computational methods in numerical quantum physics.

Secondly, it was shown above that the Hilbert space of a system consisting of L particles corresponds to the Kronecker product of the L local Hilbert spaces. As can be easily verified, it is however not true that all states in $\mathcal{H}^{\otimes L}$ have the form

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_L\rangle \tag{2.5}$$

of a Kronecker product over local states $|\psi_i\rangle$. Such states are fully defined by the states of the individual particles and are due to their form called *product states*. This in turn implies that those states in the Hilbert space that do not take this form must also contain information about inter-particle correlations. These states are called *entangled states* and play a central role in quantum physics and quantum technology. The term *entanglement* hereby refers to the inter-particle correlations existing in such states.

Thirdly, the requirement for states in quantum physics to be normalized corresponds to the fact that the squared amplitudes $\|\psi(s_1, s_2, \cdots, s_L)\|^2$ are usually interpreted as probabilities for the system to be in the respective basis state. The squared complex amplitudes thus constitute a discrete probability distribution over the set of basis states. If this distribution is not trivial, i.e. $\psi(s_1, s_2, \cdots, s_L) < 1 \forall s_1, s_2, \cdots, s_L$, then the system is said to be in a *superposition* of states.

While the above definition of quantum states already allowed us to briefly elaborate on some fundamental concepts in QM, we now need to introduce a generalization of states relevant for this thesis, called *density operators*. A density operator or mixed state is defined as

$$\rho = \sum_j p_j \, |\psi_j\rangle \, \langle\psi_j| \tag{2.6}$$

where it holds that $p_j \in \mathbb{R}, p_j \geq 0$ and $\sum_j p_j = 1$. The states $|\psi_j\rangle$ are in this context called *pure states* and correspond to states of one or more particles as defined above. By writing ρ in an orthonormal basis one obtains the corresponding *density matrix*. Density operators can thus be perceived as defining a probability distribution over pure states. For a measurement, in QM represented by a Hermitian operator O, the expected value of its outcome is then given by

$$\langle O \rangle = \sum_j p_j \, \langle\psi_j| \, O \, |\psi_j\rangle = \mathsf{Tr}\rho O. \tag{2.7}$$

Because of their probabilistic nature, mixed states described by density operators can be analyzed with analytical tools also known in classical information theory such as for instance entropy measures. An important property of both pure and mixed states is the scaling of the entanglement between subsystems. If a system is split into two parts and the entanglement between these two parts grows at most proportionally with the size of the boundary between these parts, it is said to exhibit an *area law* scaling [3, 4]. If it however scales with the volume of the boundary, it is said to obey a *volume law*. This notion can be made more precise in terms of the von Neumann entanglement entropy, which however exceeds the scope of this chapter. From this description it should only become clear intuitively that a state following an area law scaling requires less information to be described accurately than one obeying volume law scaling. Indeed, TN techniques implicitly rely on an area law scaling for an efficient and accurate representation of a state to be possible [5, 6].

2.2 Quantum systems

Having described states of quantum systems, we now turn to the description of the systems themselves. A system in quantum physics is described by a matrix, called the *Hamiltonian* matrix or simply Hamiltonian, which for an L-particle spin system takes the form of a Hermitian matrix $H \in \mathbb{C}^{2^L \times 2^L}$. The Hamiltonian describes the energy configuration of the given system and determines the interaction between its particles. One well-known example of a Hamiltonian describing a string of spin particles with nearest-neighbour interactions is the *Ising Hamiltonian* given by

$$H = J \sum_{i=1}^{L-1} I^{\otimes i-1} \otimes \sigma_x \otimes \sigma_x \otimes I^{\otimes L-i-1} + g \sum_{i=1}^{L} I^{\otimes i-1} \otimes \sigma_z \otimes I^{\otimes L-i} \tag{2.8}$$

$$+ h \sum_{i=1}^{L} I^{\otimes i-1} \otimes \sigma_x \otimes I^{\otimes L-i} \tag{2.9}$$

where J, g and h are real scaling constants or *field strengths*, I is the identity and $\sigma_{x,y,z}$ denote the Pauli matrices

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Note that in the physical literature, the identity matrices are typically omitted for convenience. In this model, the first term of the sum corresponds to the interaction between two neighbouring particles on the string whereas the last two terms only affect the individual particles. Although the Ising Hamiltonian is only one particular case, it suffices to illustrate the structure of sums of Kronecker products of local Hamiltonians that is common to Hamiltonians in QM. This structure reflects the tensor product structure of the underlying Hilbert space and in some cases allows for the efficient and exact expression of Hamiltonians in TN form [6].

As we have stated above, a Hamiltonian describes the energy configuration of a system. This is made explicit by the famous *time-independent Schrödinger equation*

$$H |\psi\rangle = E |\psi\rangle \tag{2.10}$$

where E is the total energy of the system. From a mathematical perspective, this equation is a standard eigenvalue problem. It reflects the fact that H as a Hermitian matrix admits a spectral decomposition

$$H = \sum_i E_i |\psi_i\rangle \langle\psi_i| \tag{2.11}$$

in terms of real eigenvalues or energies E_i and eigenstates $|\psi_i\rangle$. Then, the time-independent Schrödinger equation states that the action of the Hamiltonian on one of its eigenstates is equal to the scaling of that state by its energy. The eigenstate corresponding to the lowest energy E_0 is of particular interest in quantum mechanics and is referred to as the *ground state*. An important property of Hamiltonians thereby is, whether, roughly speaking, there exists a finite difference Δ between the ground state energy E_0 and the energy of the first excited state E_1, independently of L. Such Hamiltonians are referred to as *gapped*. A common way of approximating the ground state and its energy is to make use of the relation

$$\frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle} \geq E_0 \tag{2.12}$$

and minimize over $|\psi\rangle$. This is known as the *variational method* of quantum physics and poses a major application of TNs [7, 8].

Until now we have discussed several static properties of quantum systems and states but left open the question of how quantum systems evolve in time. The answer to this question is provided by the *time-dependent Schrödinger equation*

$$ih\frac{\delta}{\delta t}\left|\psi(t)\right\rangle = H\left|\psi(t)\right\rangle$$

where h is Planck's constant and t denotes the time. Solving the equation for $|\psi(t)\rangle$ yields the quantum time evolution

$$\left|\psi(t)\right\rangle = U(t)\left|\psi(0)\right\rangle$$

with the unitary time evolution operator $U(t) = e^{-itH/h}$. In the rest of this thesis, we will for convenience set $h = 1$. The equation obviously implies that the evolution of a quantum system in time is reversible, which for instance has severe effects on the way quantum computation must be implemented.

2.3 Quantum control

We will now use this insight to give a short introduction to quantum control as it is relevant for the work presented in appendices D and E. Here, we assume time to be discretized with time steps Δt and total time T. In quantum control, one then assumes to be able to control the time evolution by application of in our case time-independent *control Hamiltonians* $H_1, H_2, \cdots, H_C$, which yields the controlled time evolution

$$\left|\psi(T)\right\rangle = U(\Delta t, c_N)\cdots U(\Delta t, c_2)U(\Delta t, c_1)\left|\psi(0)\right\rangle$$

where $N = T/\Delta t$, $U(\Delta t, c_k) = e^{-i\Delta t\sum_{j=1}^{C} c_{kj}H_j}$ and the c_{kj} are time-dependent scaling constants for $H_1, H_2, \cdots, H_C$. It will however in general not be the case that we have full control over the system, for instance because of noise effects or because we expose the system to a constant magnetic field. Such effects can be modelled by introducing a noise or *drift Hamiltonian* H_0. In this thesis, we assume the drift Hamiltonian to be time independent which leads us to the final formulation

$$\left|\psi(T)\right\rangle = U(\Delta t, c_N, H_0)\cdots U(\Delta t, c_2, H_0)U(\Delta t, c_1, H_0)\left|\psi(0)\right\rangle$$

of the controlled time evolution with $U(\Delta t, c_k, H_0) = e^{-i\Delta t(H_0+\sum_{j=1}^{C} c_{kj}H_j)}$. Note that it is also possible to derive a controlled time evolution for density operators [9], which we however omit here for the sake of brevity.

Having obtained a precise formulation of the controlled time evolution, we need to formalize the actual control problem. It is natural to assume that we start with an *initial state* $|\psi(0)\rangle$ or the respective density operator $\rho(0)$ and would like to steer it towards a *target state* $|\psi^*\rangle$ or ρ^*. This requires us to define some distance function or measure of similarity between the intial and target state which can be optimized over the control parameter space. A natural candidate for this is the overlap as given by the inner product between the states, which is defined as

$$S(\psi^*, \psi(T)) = \langle\psi^*|\psi(T)\rangle \text{ or } S(\rho^*, \rho(T)) = \mathsf{Tr}\rho^{*\dagger}\rho(T)$$

respectively for Hermitian operators where † denotes the conjugate transpose. The states $|\psi(T)\rangle$ and $\rho(T)$ denote the result of the controlled time evolution. For a non-Hermitian

density operator, one can use only the real part given by $\mathrm{Re}(S(\rho^*, \rho(T)))$. Using this *figure of merit* or error function, a formal definition of the control problem is then given by

$$\max_{\{c_{kj}\}} S(\psi^*, \psi(T, \{c_{kj}\}) \text{ or } \max_{\{c_{kj}\}} S(\rho^*, \rho(T, \{c_{kj}\})$$

where we emphasize the resulting state's dependence on the control parameters. This problem statement is general enough to cover all control tasks considered in this dissertation.

3 Tensor networks

This chapter will provide a short introduction to some concepts from the field of tensor networks. Since in this thesis we only deal with particular classes of TNs called matrix product states (MPS) and matrix product operators (MPO), and certain generalizations of them, we will mainly focus our explanation on these. For a general introduction, we refer the interested reader to the overview articles in Refs. [10, 11, 12, 13]. In this chapter, we will take a numerical perspective and hence follow standard mathematical notation.

3.1 Basic concepts

Before we introduce MPS and MPO however, we will provide some brief background on tensor networks in general. To build up an intuition for TNs, it is instructive to familiarize oneself with the graphical notation used in the field to describe basic linear algebra operations on scalars, vectors, matrices and tensors of higher order. Figure 3.1 shows the notation for these linear algebra objects. The important aspect hereby is that the indices of a given tensor are depicted as lines going out from the square or circle representing the object itself. These lines are colloquially referred to as the *legs* of a tensor. Based on this notation for tensors, it is easy to represent the summation over a common index. This summation is in TN terminology referred to as the *contraction* of an index shared between two tensors. Figure 3.2 illustrates the standard matrix-vector and matrix-matrix products as well as the contraction of multiple indices between multiple tensors. The last contraction also is an example of a small tensor network which represents a tensor D_{mik} by a network of three tensors A_{nij}, B_{ljk} and C_{mnl} such that

$$D_{mik} = \sum_{j,l,n} A_{nij} B_{ljk} C_{mnl}. \tag{3.1}$$

Looking at the graphical representation of this sum, it then also becomes clear why it can be called a tensor network as the contractions can be identified with a graph. In this graph, the set of vertices V then corresponds to the set of tensors in the network and the set of edges E contains all contracted indices. In the above example, this would yield $V = \{A, B, C\}$ and $E = \{j, l, n\}$. The indices not contained in E are often called the *open indices* and constitute the indices of the resulting tensor D.

A natural question arising from these explanations is the question of how to find the representation, i.e. a *decomposition*, of a given tensor. While there exist several algorithms to find a decomposition for different classes of TNs most of these rely on two core routines, the *matricization* of a tensor and the *singular value decomposition* (SVD). Because of their central role for computing and understanding tensor decompositions we will now briefly describe both concepts. For a tensor $A_{i_1,i_2,\cdots,i_L} \in \mathbb{C}^{n_1 \times n_2 \times \cdots \times n_L}$, the *mode-k matricization* $\mathrm{Mat}(A_{i_1,i_2,\cdots,i_L}, k) \in \mathbb{C}^{n_k \times n_1 n_2 \cdots n_{k-1} n_{k+1} \cdots n_L}$ is given by

$$\mathrm{Mat}(A_{i_1,i_2,\cdots,i_L}, k) = A_{i_k,(i_1,\cdots,i_{k-1},i_{k+1},\cdots,i_L)} = \sum_{i_1,i_2,\cdots,i_L} a_{i_1,i_2,\cdots,i_L} e_{i_k} e^T_{i_1,\cdots,i_{k-1},i_{k+1},\cdots,i_L} \tag{3.2}$$

where $e_{i_1,\cdots,i_{k-1},i_{k+1},\cdots,i_L} = e_{i_1} \otimes \cdots \otimes e_{i_{k-1}} \otimes e_{i_{k+1}} \otimes \cdots \otimes e_{i_L}$ and e_i are the columns of the identity matrix. The matricization thus provides a tool to express a tensor in terms of matrices, which

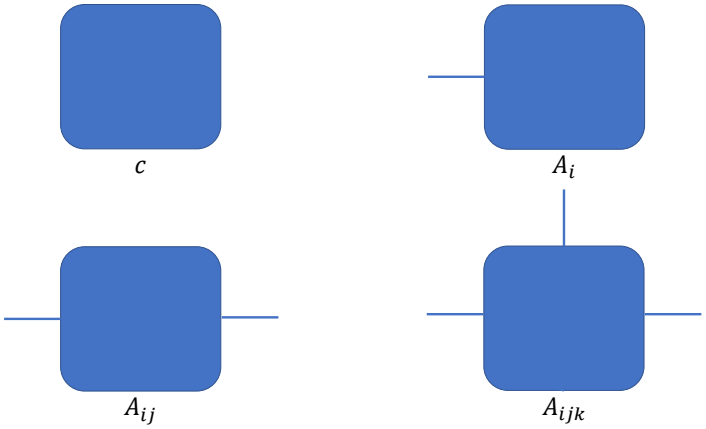


Figure 3.1: The graphical notations for tensors with zero, one, two or three indices from the top left to the bottom right. Higher numbers of indices are depicted following the same scheme.

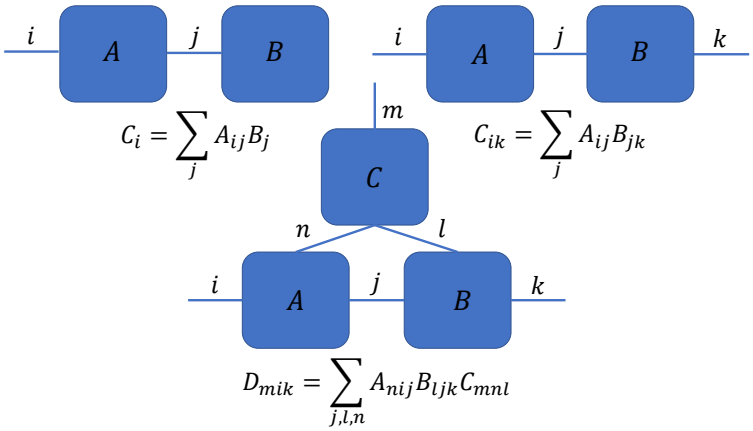


Figure 3.2: Examples of the graphical notation for contractions. The top left contraction corresponds to the standard matrix-vector product in linear algebra, whereas in the top right the matrix-matrix product is shown. In the bottom row, the contraction of a small tensor network is depicted.

for instance allows to perceive the contraction over an index i_k as the multiplication of mode-k matricized tensors. This multiplication is referred to as the *k-mode product*. The matricization also gives rise to the concept of the *multilinear rank* $(r_1, r_2, \cdots, r_L)$ of a tensor $A_{i_1,i_2,\cdots,i_L}$, which is simply defined in terms of the matrix rank as $r_k := rank(Mat(A,k))$, where we have omitted the indices for clarity. While there also exists a well-defined concept for the rank of a tensor [14], the concept of the multilinear rank is of high importance for the computation of tensor decompositions.

Being able to express tensors in matrix form leads us to the SVD. The SVD of a matrix $A \in \mathbb{C}^{m \times n}$ is defined as

$$A = U \Sigma V^* \tag{3.3}$$

with $U \in \mathbb{C}^{m \times m}$ and $V \in \mathbb{C}^{n \times n}$ being unitary matrices and $\Sigma \in \mathbb{R}^{m \times n}$ being a rectangular diagonal matrix with $\min(m,n)$ non-negative *singular values* σ_i on its diagonal. While the SVD is of great importance in many numerical fields, in the context of tensor networks it poses an elemental building block for the decomposition of a matrix or a matricized tensor. This is due to the fact that a matrix A can be easily decomposed into two matrices B and C by

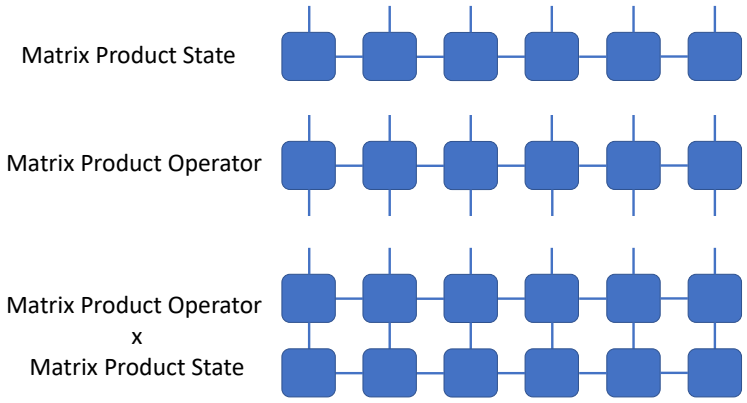


Figure 3.3: Graphical representations of an MPS, an MPO and MPO-MPS multiplication corresponding to matrix-vector multiplication. The top row shows the MPS, the middle row depicts the MPO and in the bottom row the multiplication is shown.

computing its SVD and defining, e.g., $B = U$ and $C = \Sigma V^*$. By restricting the decomposition to only consider the r largest singular values with the respective parts of U and V, one then obtains a truncated expression of rank r that is optimal with respect to the Frobenius norm. This property is known as the *Eckart-Young-Mirsky Theorem* [15]. The error of the truncation is determined by the magnitudes of the ignored singular values. This fact can be used to dynamically choose r when computing tensor decompositions. There exist generalizations of the SVD to tensors of higher order as most importantly the *higher-order SVD* [16] (HOSVD) which is based on the *Tucker decomposition* [17]. For the HOSVD, it is however the case that diagonality of Σ can not be guaranteed and only a weaker form of the Eckart-Young-Mirsky Theorem holds. As even the computation of the HOSVD relies on the repeated application of it, the SVD thus forms the backbone of many tensor decomposition schemes. In the context of QM it can furthermore be shown that the SVD is equivalent to the *Schmidt decomposition*, a measure for the entanglement of two systems. This equivalence provides justification for the application of the SVD in quantum physical tensor networks.

3.2 Matrix product states and operators

Following this more general introduction, it is now time to present MPS and MPO with their most important properties. As the name suggests, matrix product states and matrix product operators constitute a class of TNs designed to represent and approximate high-dimensional vectors and matrices. At the time of this writing, MPS and MPO approaches provide the foundation for some of the most successful simulation algorithms in numerical quantum many body physics. They are also used with success as analytical tools for the study of such systems. In quantum physics, MPS and MPO were first recognized to be useful numerical tools in the context of the *density matrix renormalization group* [7, 8], the most successful algorithm for the approximation of low-energy states today. Apart from the prime example of approximating ground states we already briefly mentioned in Chapter 2, e.g., also thermal states in equilibrium can be successfully tackled [18, 19] with MPS/MPO methods. However, an equivalent concept has also been independently developed in numerical mathematics and is there referred to as the tensor train (TT) decomposition [20]. Although we take a primarily numerical perspective in this chapter, we will mostly make use of the terminology used in numerical many-body quantum physics for the sake of compatibility with QM literature.

Now, an MPS describes the decomposition of a vector $v \in \mathbb{C}^N$ such that

$$v_i = v_{i_1 \dots i_L} = \mathsf{Tr} A_1^{i_1} A_2^{i_2} \cdots A_L^{i_L} \tag{3.4}$$

where we have split up the index i into L sub-indices $i_1, \dots, i_L$ of dimension d. The tensors $A_1, \dots, A_L \in \mathbb{C}^{d \times D \times D}$ are called the core tensors of the matrix product state. While the A_i in this generic formulation do not necessarily hold any interpretable information, in quantum physics each core tensor represents an individual particle on a ring or string. In fact, the dimension d then is precisely the physical dimension of the particles we have discussed in the last chapter. From the physical perspective, D controls the possible amount of information shared between neighbouring particles and is correspondingly called the *bond dimension*. The physical dimension is accessed by the indices i_j of the A_j. In numerical mathematics or computer science, the approach of splitting up the index i is a common way to obtain the MPS representation of a vector.

Above we have introduced the A_i to be of equal dimension. This naturally must not be the case as, e.g., in a quantum system the amount of entanglement must not be equal among all pairs of particles. Likewise, not all particles must be of equal type such that in principle also d can vary within an MPS. For the sake of simplicity and without loss of generality, we however assume $D = \max_i D_i$ for the potentially differing bond dimensions D_i of the A_i. We furthermore assume all core tensors to have the same physical dimension, which is also justified by the fact that in this thesis we only consider systems of spin-1/2 particles. Note that we have also assumed here that $N = d^L$, which, as we have seen in Chapter 2, occurs naturally in quantum physics. This relation provides the basis of the ability of MPS to represent very high-dimensional vectors.

One can also define a slightly different version of matrix product states such that

$$v_i = v_{i_1 \dots i_L} = A_1^{i_1} A_2^{i_2} \cdots A_L^{i_L} \tag{3.5}$$

where $A_1 \in \mathbb{C}^{d \times 1 \times D}$ and $A_L \in \mathbb{C}^{d \times D \times 1}$. From a physical perspective, this slightly simpler representation corresponds to a system with oben boundary-conditions (OBC), i.e. a string of particles. The form shown in equation 3.4 consequentially describes a system with closed or periodic boundary-conditions (CBC or PBC), hence forming a ring. Comparing both expressions, it also becomes clear that OBC poses a special case of PBC. If not explicitly stated otherwise, we will in this work always assume open boundaries. Our results are in general unaffected by this assumption. It holds for both kinds of MPS that a given element of the vector v is represented by a sequence of matrix multiplications. This explains the names of *matrix product states* and *tensor trains*.

Using equation 3.5, we can write the entire vector v as

$$v = \sum_{i_1, \dots, i_L}^{d} \left(A_1^{i_1} A_2^{i_2} \cdots A_L^{i_L}\right) \left(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L}\right) \tag{3.6}$$

$$= \sum_{k_2, \dots, k_{L-1}}^{D} \left(\sum_{i_1}^{d} A_{1,k_2}^{i_1} e_{i_1}\right) \otimes \left(\sum_{i_2}^{d} A_{2,k_2 k_3}^{i_2} e_{i_2}\right) \otimes \cdots \otimes \left(\sum_{i_L}^{d} A_{L,k_{L-1}}^{i_L} e_{i_L}\right) \tag{3.7}$$

$$= \sum_{k_2, \dots, k_{L-1}}^{D} u_{1,k_2} \otimes u_{2,k_2 k_3} \otimes \cdots \otimes u_{L,k_{L-1}} \tag{3.8}$$

where the e_j again refer to the jth columns of the identity matrix and the subscripts k_j and k_{j+1} correspond the row and column indices of the matrices $A_j^{i_j}$. From this expression, it

becomes clear that MPS do in fact exhibit the same tensor product structure we already found in quantum mechanics. For instance, it is easy to see that an MPS with a bond dimension of $D = 1$ is a product state as defined in Chapter 2. This also illustrates the connection between the bond dimension of an MPS and the entanglement of the represented state. Furthermore, this representation of MPS allows for a more easy comparison with other tensor decompositions in numerical computer science and mathematics, such as the *canonical polyadic decomposition* (CPD) [14].

We now move on to the representation of matrices as matrix product operators. To represent a matrix $B \in \mathbb{C}^{N\times N}$ as an MPO, the OBC MPS decomposition introduced above can be adapted in a straight-forward fashion such that

$$B_{ij} = B_{i_1\dots i_L j_1\dots j_L} = A_1^{i_1 j_1} A_2^{i_2 j_2} \cdots A_L^{i_L j_L} \tag{3.9}$$

where the two indices i and j have again been split up and we thus have that $A_1, \dots, A_L \in \mathbb{C}^{d\times d\times D\times D}$. In a direct generalization of the vector-case, one can now also express the entire matrix as

$$B = \sum_{i_1,\dots,i_L,j_1,\dots,j_L}^{d} (A_1^{i_1 j_1} A_2^{i_2 j_2} \cdots A_L^{i_L j_L}) \cdot (e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L})(e_{j_1}^T \otimes e_{j_2}^T \otimes \cdots \otimes e_{j_L}^T) \tag{3.10}$$

$$= \sum_{i_1,\dots,i_L,j_1,\dots,j_L}^{d} \sum_{k_2,\dots,k_{L-1}}^{D} (A_{1,k_2}^{i_1 j_1} A_{2,k_2 k_3}^{i_2 j_2} \cdots A_{L,k_{L-1}}^{i_L j_L}) \cdot (e_{i_1} e_{j_1}^T) \otimes (e_{i_2} e_{j_2}^T) \otimes \cdots \otimes (e_{i_L} e_{j_L}^T) \tag{3.11}$$

$$= \sum_{k_2,\dots,k_{L-1}}^{D} U_{1,k_2} \otimes U_{2,k_2 k_3} \otimes \cdots \otimes U_{L,k_{L-1}} \tag{3.12}$$

with e_j again being jth column of the identity matrix. However, the same expressions can also be derived for other product bases as, e.g., the Pauli basis. In fact, several spin Hamiltonians can be expressed *exactly* in MPO form with a small bond dimension of for instance only $D = 2$ or $D = 3$ [21, 6].

3.3 Properties of matrix product states and operators

Based on the above equations one can relatively easily derive expressions for basic linear algebra operations like the multiplication with a scalar, addition, the inner product or the matrix-vector product. In Figure 3.3 we depict an MPS and MPO in the graphical notation introduced before and show the tensor network corresponding to the multiplication of the MPO onto the MPS. Performing an addition or the multiplication of an MPO onto an MPS exactly however increases the bond dimension D of the representation. Adding two MPS or MPO with bond dimensions D and D' results in an MPS or MPO with bond dimension $D'' \leq D + D'$. In case of the multiplication it holds $D'' \leq D \cdot D'$ [10]. This is a direct consequence of the expressions for the respective operations.

From the above discussion it has also become clear that the bond dimension D is the critical factor determining the expressive power of a given MPS or MPO. While it is in principle always possible to represent a vector or matrix exactly by an MPS or MPO, the bond dimension required for this might in the worst case be exponential in L and go up to $d^{\lfloor L/2 \rfloor}$ for MPS [22]. This implies that for a limited maximal value of D, not all vectors and operators can be expressed exactly anymore. For these vectors and operators, the representation as MPS or MPO hence gives rise to approximation errors. This case of an approximative represention is then referred to as the representation with a *truncated* bond dimension. When appropriate,

we will indicate that a vector v or matrix B is approximated using a bond dimension D by writing $v[D]$ and $B[D]$ respectively. Still, for many physically interesting states and operators it was discovered that matrix product states and operators yield good approximations for $D \in \mathcal{O}(poly(L))$ [5, 18, 19]. Such D leads to a total number of parameters $LdD^2 \in \mathcal{O}(poly(L))$ for MPS and $Ld^2D^2 \in \mathcal{O}(poly(L))$ for MPO in contrast to d^L or d^{2L} for the entire respective vector or matrix. In this context, properties such as the area law discussed in the last chapter then become important.

Because of their ability to efficiently approximate relevant physical states to a good degree, MPS and MPO constitute the most successful TN approaches in numerical QM. Note that a polynomial complexity would not necessarily be considered efficient in computer science. But since in this case the actual complexity is exponential, we find it justified to make use of the term in this thesis. We would also like to point out that there is active research in the generalization of the above approaches to two or three dimensions. These approaches however face the major problem that already the contractions necessary to approximate states are infeasible to perform exactly [23]. We will thus and in the interest of brevity not comment further on this topic.

Because of their importance, several algorithms have been introduced in both the numerical and physical community that try to find optimal and canonical MPS or MPO representations for a given D. A general scheme that most successful methods follow hereby is to perform the optimization of the given MPS or MPO with respect to a certain error function in terms of *local updates*. Starting with the left- or right-most core tensor A_i, these methods update the current tensor (or two tensors in some cases) with all other core tensors being treated as constants. Then, they move on to the next respective tensor, perform the update and continue in this way until they reach the other end of the chain. The complete iteration over all core tensors is called a *sweep* and typically an algorithm sweeps back and forth until some convergence criterion is met. Because all except the current one or two tensors are assumed fixed, these algorithms typically allow for an efficient implementation in a dynamic programming style. Providing a more thorough introduction to the various algorithms unfortunately lies outside the scope of this chapter and thus we again refer the interested reader to the overview articles [10, 11, 13, 12].

4 Krylov methods

As they play an important role in this thesis, we will in this chapter explain the most relevant aspects of the Lanczos and global Lanczos algorithms as examples of Krylov methods. We will again take here a numerical perspective, underlined by use of mathematical notation.

4.1 Lanczos algorithms

Krylov algorithms constitute a well-established method in several numerical domains including numerical quantum physics and are used among other things to solve linear systems [24, 25, 26], find eigenvectors [27, 28, 29, 30] or approximate the time evolution of a quantum state [31, 32]. The general scheme of these algorithms is that for a given matrix $A \in \mathbb{C}^{N\times N}$ and an *initial vector* $u \in \mathbb{C}^N$, they iteratively build up an orthonormal basis $U_K = [u_1, u_2, \ldots, u_K] \in \mathbb{C}^{N\times K}$ of the Krylov subspace $\mathcal{KR}_K = \{u, Au, A^2u, \ldots, A^{K-1}u\}$. Here, K denotes the dimension of that space. Then, the projection of A on $\mathcal{KR}$ can be used to approximate various properties of A. We will now make this notion more precise by introducing the Lanczos algorithm and then later generalize our findings for the global Lanczos algorithm.

The Lanczos algorithm constitutes a special type of the Krylov algorithm in that it assumes A to be Hermitian and employs the Gram-Schmidt orthogonalization algorithm to construct U_K. While constructing the basis U_K, the algorithm simultaneously builds up a matrix $T_K \in \mathbb{R}^{K\times K}$ given by

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix}, \tag{4.1}$$

with α_i and β_i being defined by lines 4 and 10 in Algorithm 1. With these definitions, it then directly follows from the orthonormality of U_K and the hermiticity of A that

$$U_K^* A U_K = T_K \tag{4.2}$$

where U_K^* denotes the conjugate transpose of U_K. From this relation we then obtain that A is similar to T_N, where T_N is the tridiagonal matrix of full dimension N. This is an important insight, as it shows that the Lanczos algorithm can be perceived as computing a projection of A onto a lower-dimensional space. It also provides some intuition for the fact that the eigenvalues, also called *spectrum* in this thesis, of A are approximated by the eigenvalues of T_K. These eigenvalues are also referred to as the *Ritz values* of A [33]. We would also like to point out here that $T_K \in \mathbb{R}^{K\times K}$ must hold since the β_i are norms and the α_i on the diagonal must also be real-valued since A is Hermitian and similar to T_N.

Algorithm 1: Lanczos Algorithm

Input : Matrix $A \in \mathbb{C}^{N\times N}$, Starting Vector $u \in \mathbb{C}^N$, Number of Dimensions K

$u_0 \leftarrow 0$;
$v_0 \leftarrow u$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \|v_{i-1}\|$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $u_i \leftarrow v_{i-1}/\beta_i$;
  $v_i \leftarrow Au_i - \beta_i u_{i-1}$;
  $\alpha_i \leftarrow u_i^* v_i$;
  $v_i \leftarrow v_i - \alpha_i u_i$;
end

Output: Orthonormal Basis $U_K \in \mathbb{C}^{N\times K}$, Tridiagonal Matrix $T_K \in \mathbb{R}^{K\times K}$

To gain further insight, we now define the extended version $\widehat{T}_K$ of T_K as

$$\widehat{T}_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \\ 0 & \dots & 0 & \beta_{K+1} \end{bmatrix} \tag{4.3}$$

and obtain the partial Lanczos decomposition

$$AU_K = U_{K+1}\widehat{T}_K = U_K T_K + \beta_{K+1} u_{K+1} e_K^T \tag{4.4}$$

which connects U_K to U_{K+1} for $K < N$. The decomposition illustrates that the change between two iterations of the algorithm is proportional to β_{K+1}. This implies that the algorithm stops for $K < N$ when $\beta_{K+1} = 0$. In this case U_K then spans the eigenspace of A. To see why this decomposition holds, it is instructive to consider a three-term recurrence relation that follows directly from Algorithm 1. By construction, we have

$$Au_i = \beta_{i+1} u_{i+1} + \beta_i u_{i-1} + \alpha_i u_i \tag{4.5}$$

which can also be expressed as

$$\beta_{i+1} u_{i+1} = (A - \alpha_i I_N) u_i - \beta_i u_{i-1} \tag{4.6}$$

with I_N being the identity matrix of dimension N. We will return to this equality later as it plays an important role in this thesis.

From the discussion above, it is clear that for every vector $v \in \operatorname{span}(U_K)$ it holds that

$$v = \sum_{i=0}^{K-1} c_i (A^i u) = \left(\sum_{i=0}^{K-1} c_i A^i \right) u = p_v(A) u \tag{4.7}$$

with $p_v(A)$ being a polynomial in A of degree $K-1$. For a given analytic function $f : \mathbb{C}^{N\times N} \to \mathbb{C}^{N\times N}$ it then follows that there exists a $v_a \in \operatorname{span}(U_K)$ for which $p_{v_a}(A)$ corresponds to a power-series approximation around 0 of degree $K-1$ of $f(A)$. This perspective provides some

Algorithm 2: Global Lanczos Algorithm

Input : Matrix $A \in \mathbb{C}^{N\times N}$, Starting Matrix $U \in \mathbb{C}^{N\times M}$, Number of Dimensions K

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
for $i \leftarrow 1; i \leq K$ **do**
    $\beta_i \leftarrow \|V_{i-1}\|_F$;
    if $\beta_i = 0$ **then**
        break ;
    end
    $U_i \leftarrow V_{i-1}/\beta_i$;
    $V_i \leftarrow AU_i - \beta_i U_{i-1}$;
    $\alpha_i \leftarrow \langle U_i, V_i\rangle_F$;
    $V_i \leftarrow V_i - \alpha_i U_i$;
end

Output: Orthonormal Basis $\mathbf{U}_K \in \mathbb{C}^{N\times KM}$, Tridiagonal Matrix $T_K \in \mathbb{R}^{KM\times KM}$

initial insight into the connection between Krylov algorithms and function approximation. It also makes it clear that K plays a critical role for the accuracy of such an approximation.

Having discussed some important aspects of the original Lanczos algorithm, we now turn to a particular generalization of it, namely *block Lanczos* algorithms. Block Lanczos algorithms mainly differ from the original algorithm by assuming to be provided with an initial *block vector*, i.e. a block of column vectors. These algorithms have been developed to, e.g., solve linear systems with multiple right-hand sides. Hereby, they rely on efficient implementations of the matrix product to achieve better complexity than the repeated application of the original Lanczos algorithm. While there exist several block versions of the Lanczos algorithm [34, 35, 36, 37, 38], we will here only discuss the *global Lanczos* algorithm [39] since it is most relevant for this thesis. Correspondingly, we will in the following assume to have an *initial matrix* $U \in \mathbb{C}^{N\times M}$ with $M \leq N$ and denote by $\mathbf{U}_K \in \mathbb{C}^{N\times KM}$ the basis consisting of K basis matrices U_i.

To generalize the Lanczos algorithm to matrices, we must first define an inner product with respect to which the basis matrices U_i are orthogonalized. For the global Lanczos algorithm it is chosen as the *Frobenius inner product*

$$\langle U_i, U_j\rangle_F = \mathsf{Tr} U_i^* U_j \tag{4.8}$$

where $U_i, U_j \in \mathbb{C}^{N\times M}$. This choice of the inner product then induces the *Frobenius norm*

$$\|U_i\|_F = \sqrt{\langle U_i, U_i\rangle}. \tag{4.9}$$

Note that in quantum physics it is more common to refer to these functions as the *Hilbert-Schmidt* inner product and norm, which generalize the above definitions to operators in possibly infinite spaces. It is easy to see that the Frobenius inner product is in fact a direct generalization of the vector inner product employed in the standard Lanczos algorithm. In the context of this work, it is of crucial importance that the above inner product and norm are not defined in terms of individual columns of the U_i as in other block algorithms, but act on the entire matrices.

Having chosen the required inner product and norm, we find that Algorithm 2 indeed is a direct generalization of the standard Lanczos algorithm to matrices. Thus, we again obtain

that the algorithm constructs a projection T_K of A, given by

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix}, \tag{4.10}$$

resulting in the *partial global Lanczos* decomposition

$$A\mathbf{U}_K = \mathbf{U}_K \tilde{T}_K + \beta_{K+1} U_{K+1} E_K^T \tag{4.11}$$

with $\tilde{T}_K = T_K \otimes I_M \in \mathbb{R}^{KM \times KM}$ and $E_K^T = [\mathbf{0}, \cdots, \mathbf{0}, I_M] \in \mathbb{R}^{M \times KM}$. In analogy to the results we showed for the original Lanczos, it then also holds that

$$\beta_{K+1} U_{K+1} = (A - \alpha_K I_N) U_K - \beta_K U_{K-1} \tag{4.12}$$

as well as

$$\mathbf{U}_K^* A \mathbf{U}_K = T_K. \tag{4.13}$$

In fact, all properties of the original Lanczos algorithm stated above also hold for the global Lanczos algorithm.

4.2 Connection to Gauss quadrature

We have already briefly touched upon the fact that Krylov methods can be used to approximate functions of the input matrix A. There does however exist a more rigorous explanation, involving the *Gauss quadrature*, we will now elaborate on. In this explanation we focus on the global Lanczos algorithm as its connection to Gauss quadrature is most important for the work presented in this dissertation.

To connect the global Lanczos algorithm and Gauss quadrature, we begin by realizing that

$$u^* f(A) u = u^* V_A f(\Lambda_A) V_A^* u = \sum_{i=1}^{N} f(\lambda_i) \mu_i^2 = \int_a^b f(\lambda) \mathrm{d}\mu(\lambda) \tag{4.14}$$

where $V_A \Lambda_A V_A^*$ is the spectral decomposition of A, $\mu_i = e_i^T V_A^* u$ and

$$\mu(\lambda) = \begin{cases} 0 & \text{if } \lambda < \lambda_1 = a \\ \sum_{i=1}^{j} \mu_i^2 & \text{if } \lambda_j \leq \lambda < \lambda_{j+1} \\ \sum_{i=1}^{N} \mu_i^2 & \text{if } b = \lambda_N \leq \lambda \end{cases} \tag{4.15}$$

is a nondecreasing and piecewise-constant distribution function. As is customary, we assume that the eigenvalues of A are ordered ascendingly. From this equality, we then obtain

$$\mathcal{I} f \coloneqq \mathsf{Tr}(U^* f(A) U) = \sum_{i=1}^{N} e_i^* U^* V_A f(\Lambda_A) V_A^* U e_i = \sum_{i=1}^{N} \int_a^b f(\lambda) \mathrm{d}\mu_i(\lambda) = \int_a^b f(\lambda) \mathrm{d}\mu(\lambda) \tag{4.16}$$

where U can for instance be the starting matrix of the global Lanczos algorithm. Here, $\mu_i(\lambda)$ is defined analogously to equation 4.15 and $\mu(\lambda) \coloneqq \sum_{i=1}^{N} \mu_i(\lambda)$.

Hence the above integral is a Riemann-Stieltjes integral which can be approximated by Gauss-type quadratures. In the most general case, these take the form

$$\mathcal{G}f \coloneqq \sum_{k=1}^{K} \omega_k f(\theta_k) + \sum_{m=1}^{M} \nu_m f(\tau_m), \tag{4.17}$$

where θ_k and τ_m are referred to as the nodes and ω_k and ν_m as the weights of the quadrature. The remainder of a Gauss-type quadrature approximation is

$$\begin{aligned} \mathcal{R}f &\coloneqq \int_a^b f(\lambda)\mathrm{d}\mu(\lambda) - \mathcal{G}f \\ &= \frac{f^{2K+M}(\eta)}{(2K+M)!} \int_a^b \prod_{i=1}^{M}(\lambda - \tau_i) \left(\prod_{j=1}^{K}(\lambda - \theta_j) \right)^2 \mathrm{d}\mu(\lambda) \end{aligned} \tag{4.18}$$

where $\lambda_1 < \eta < \lambda_N$. By letting $M = 0$ one obtains the Gauss quadrature, whereas using prescribed nodes τ_i results for instance in the Gauss-Lobatto and Gauss-Radau quadratures [40, 39]. In case of the Gauss quadrature, the sign of the approximation error then corresponds to the sign of the $2K$-th derivative of the function f. This provides a relatively easy way of knowing if the Gauss quadrature poses a lower or upper bound to the true value for a given function. It also follows that $\mathcal{G}_K f$ is exact for all polynomials of degree smaller or equal to $2K - 1$ [40].

To perform a Gauss quadrature approximation, one must thus be able to determine optimal weights and nodes. To find such ω_k and θ_k, a possible way is to construct a sequence of polynomials $\{p_0, \cdots, p_K\}$ that are orthonormal in the sense that

$$\int_a^b p_i(\lambda) p_j(\lambda) \mathrm{d}\mu(\lambda) = \delta_{ij} \tag{4.19}$$

and follow the recurrence relation

$$\beta_i p_i(\lambda) = (\lambda - \alpha_{i-1}) p_{i-1}(\lambda) - \beta_{i-1} p_{i-2}(\lambda), \tag{4.20}$$

with $p_{-1}(\lambda) \equiv 0$ and $p_0(\lambda) \equiv 1$. The roots of the polynomial p_K have been shown to be optimal θ_k [40, 41]. From the above recurrence relation one can obtain a recurrence matrix T_K given by

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix}, \tag{4.21}$$

whose eigenvalues constitute the roots of $p_K(\lambda)$ and thus are optimal θ_k of $\mathcal{G}f$ [41]. The weights ω_k are determined by the squared first elements of the normalized eigenvectors of T_K such that

$$\mathcal{G}f = e_1^T f(T_K) e_1 = e_1^T V_T f(\Lambda_T) V_T^* e_1, \tag{4.22}$$

with $V_T \Lambda_T V_T^*$ being the spectral decomposition of T_K.

The above recurrence relation and tridiagonal matrix bare strong resemblence to what we saw in the introduction of the Lanczos algorithms. In fact, the basis matrices U_i as computed by the global Lanczos algorithm can be perceived as containing polynomials in A

$$U_i = p_{i-1}(A) U \tag{4.23}$$

where p_{i-1} is a polynomial of degree $i-1$. From the orthonormality of the U_i, we then obtain

$$\langle p_{i-1}(A)U, p_{j-1}(A)U\rangle = \langle U_i, U_j\rangle = \delta_{ij} \tag{4.24}$$

and making use of equation 4.16 we also find

$$\langle p_{i-1}(A)U, p_{j-1}(A)U\rangle = \mathsf{Tr}(U^* p_{i-1}(A)^* p_{j-1}(A)U) = \int_a^b p_{i-1}(\lambda)p_{j-1}(\lambda)\mathrm{d}\mu(\lambda). \tag{4.25}$$

This shows that the global Lanczos algorithm constructs a sequence of orthonormal polynomials which follow the recurrence relation stated in equation 4.20. The tridiagonal matrix T_K produced by the global Lanczos algorithm hence poses a recurrence matrix as it is required for the Gauss quadrature.

As a final remark we note that choosing $U \in \mathbb{C}^{N\times N}$ to be square and unitary results in

$$\mathsf{Tr}f(A) = \mathsf{Tr}(U^* f(A)U) = \int_a^b f(\lambda)\mathrm{d}\mu(\lambda) \approx e_1^T f(T_K)e_1. \tag{4.26}$$

For U with fewer columns than A and non-unitary $U \in \mathbb{C}^{N\times N}$, it however only holds $\mathsf{Tr}f(A) \approx \mathsf{Tr}(U^* f(A)U)$ and thus the Gauss quadrature does not approximate the actual function $\mathsf{Tr}f(A)$. While it is generally not assumed to be feasible to choose a U of size equal to A, we will introduce a way to achieve this (and unitarity) by means of tensor network representations in this dissertation.

5 Machine learning

As we have now reviewed in brief quantum physics, tensor networks and Krylov algorithms, what is left is to provide some background information on machine learning. This chapter will thus give a short introduction to machine learning in general and then present two classes of machine learning models which are relevant for this thesis. For a general introduction to the field and a more in-depth discussion of the topics presented below, we recommend the standard literature [42, 43, 44, 45].

5.1 Basic concepts

Machine learning (ML) can from an abstract perspective be defined as the study of algorithms that improve their performance P on some task T with experience E, such that P is not only improved for E itself but in general for all instances of T. This property is called *generalization* and as we have already hinted at in the introduction, it lies at the heart of machine learning. Without the generalization property, the above definition could also be interpreted as a pure optimization problem. Since it can be assumed that generalization is only possible if the structure of T is captured at least to a certain degree by the algorithm, the term *learning* is employed here. ML can thus also be perceived as tackling the problem of performing an optimization of P for some task T via solving the surrogate problem of optimizing P over a finite set of instances E of T. The crucial questions then are to determine until which point the optimization of the surrogate problem is equivalent to optimizing the actual problem, i.e. how long the learned structure generalizes, and how to maximize this overlap. An important concept in this context is that of *overfitting*, which refers to the misinterpretation of randomness in E as structure of T up to the point of simple memorization of the experienced instances. Opposed to this, the term *underfitting* is used to refer to the situation when relevant information in the data is ignored. Both phenomena adversly affect the generalization of a learning method.

While ML today is a wide field consisting of many separate lines of research, it can be roughly subdivided into three main branches: supervised learning, unsupervised learning and reinforcement learning. In the context of this thesis, especially supervised and reinforcement learning are relevant, but for the sake of completeness, we will also briefly introduce unsupervised learning.

5.1.1 Supervised learning

In *supervised learning*, we assume E to take the form of a *training data* set $D = \{(x_i, y_i)\}_{i=1}^{N}$ of N tuples of input-output pairs x_i and y_i. These tuples have been generated by a function $f : X \to Y$ such that $x_i \in X, y_i = f(x_i) \in Y$ where X, Y can in principle be any set. The term *supervised* now derives from the fact that the learning algorithm is given a set of samples x_i together with correct answers or *targets* y_i. Additionally, we have a set H of functions $h : X \to Y$, often called *hypothesis space* with the functions correspondingly called *hypotheses*. In this thesis, the hypothesis space will mostly correspond to certain classes of artificial neural networks (ANN) with a set of free parameters Θ that must be optimized. We will often refer to a given instance of an ANN as a *model*.

The task T now is to predict y from x based on H. The performance P for this task is measured by a *loss function* $L(y, h(x))$ with $L : Y \times Y \to \mathbb{R}$ and the optimization problem then

can be written as

$$\arg\min_{h \in H} L_T(h) = \arg\min_{h \in H} \sum_{i=1}^{N} L(y_i, h(x_i)) \tag{5.1}$$

to minimize the total loss by optimizing over the hypothesis space. If instead of the total loss the average loss, also called *risk*, $(1/N)L_T(h)$ is minimized, one obtains *empirical risk minimization* as defined in *statistical learning theory* (SLT) [46, 47]. The process of performing the above optimization is commonly referred to as *training*.

This formulation of the learning problem only refers to the given data but does so far not reflect the fact that we would like to choose $h \in H$ such that it performs well on the entire domain X. To address this issue, one commonly adds a *regularization* term $L_R(h)$ to $L_T(h)$ that penalizes strong adaption to the training data. Optimizing the regularized average loss then can be perceived as performing *structural risk minimization* from the SLT perspective. The reasoning behind regularization is that adapting too strongly to the training data likely results in overfitting. On the other hand, regularizing too strongly might prevent learning of characteristics that generalize. Thus, choosing the right kind of regularization is an important problem not only in supervised learning but in ML in general.

The regularization problem leads us to another tightly connected and important tradeoff in ML, namely the *bias-variance* tradeoff. The higher the expressive power or *capacity* of a set of hypotheses, the better it can fit or even overfit structure in the training data and the more volatile the results for training it on different training sets will be when no suitable regularization is applied. Such high volatility and overfitting are a case of high variance. If the capacity is too low however, it is likely that the relevant structure in the data can not be captured well enough by the hypotheses, they underfit, and the loss can not be minimized to a satisfying degree. This then is a case of high bias. The structure which is found in the data on the other hand then is unlikely to be simply random noise and will thus likely generalize. Since low capacity induces high bias and high capacity induces high variance, both sources of error unfortunately can not be minimized simultaneously. The tradeoff hence consists of choosing the capacity such that the induced variance does not result in overfitting but the bias is also not too high to learn the structure of the task. While the notions above can be made more precise by employing tools from SLT, we will in the interest of briefness not further dive into this.

As a last remark, we would like to point out that while the above introduction took a more traditional and *frequentist* perspective, supervised learning can also be conducted in the *Bayesian probability* framework. It is often possible to derive loss functions of the above form by taking the *log-likelihood* $\ln \prod_i p_h(y_i|x_i) = \sum_i \ln p_h(y_i|x_i)$ of the training set with respect to a defined probability distribution $p_h(y_i|x_i)$ which is parameterized by the hypotheses h. Optimizing the corresponding loss then constitutes performing a *maximum likelihood* estimation of h. If one furthermore assumes a prior distribution over hypothesis space to apply some kind of regularization, the optimization corresponds to an *a posteriori* estimation. As a more thorough review of *Bayesian machine learning* is out of scope for this chapter, we refer the interested reader to the standard literature [42, 43].

Supervised learning is most commonly applied to solve *classification* or *regression* problems.

5.1.2 Unsupervised learning

In contrast to supervised learning, *unsupervised learning* assumes to be only given a training set $D = \{x_i\}_{i=1}^N$ of samples $x_i \in X$ from some data-generating process or distribution over the domain X. The term *unsupervised* is hence used to reflect the absence of any provided

target values for the x_i. Thus, the task is not directly to learn a mapping from X to some image space Y but to find structure in the data itself, i.e. learn something about the data-generating process or distribution. One particular case of unsupervised learning for instance is the problem of *clustering*. Hereby, the domain X is assumed to be divided into several clusters whose members are similar according to some distance measure. These clusters are to be determined from D. A different variant of unsupervised learning makes the assumption that X can be mapped to a *latent space* Y of lower complexity than X which however still captures all or at least the major factors of variation. In this case, the learning problem is often formulated such that two functions $e : X \to Y$ and $d : Y \to X$ have to be learned such that $L(x_i, d(e(x_i)))$ is minimized where $L : X \times X \to \mathbb{R}$ is some distance measure. By solving this learning problem, one then hopes to gain a deeper understanding of the data or find a compact representation from which also new valid instances of the underlying process or distribution can be generated.

5.1.3 Reinforcement learning

Reinforcement learning (RL) again takes a different perspective that is more akin to how learning is thought to happen in biological systems. The learning process here is thought of as an action-feedback loop between a learning agent and an environment. The agent performs an action according to some *policy*, affecting the environment, and in turn receives a reward for the action as well as the new state of the environment after the action. Based on this information, the agent adapts its behavior to maximize the reward and the interaction continues.

This learning model is formally based on *Markov decision processes* (MDP), which are defined as a 5-tuple (S, A, P, R, γ) with S being a set of states of the environment and A denoting a set of actions the agent can perform. The probability of transitioning to a state $s' \in S$ after performing action $a \in A$ in state $s \in S$ is given by the function $P : S \times S \times A \to [0, 1]$ with $P(s, s', a) = p(s_{t+1} = s' | s_t = s, a_t = a)$. Here, p is the corresponding Markovian probability distribution and the subscripts t and $t + 1$ refer to consecutive points in time. The reward of an action a conducted in state s and resulting in state s' is given by the reward function $R : S \times S \times A \to \mathbb{R}$, $R(s, s', a)$. The scalar γ is called the *discount factor* and is used to quantify the preference for short-term over long-term rewards.

While MDPs pose the formal basis of RL scenarios, it is important to note that RL methods explicitly assume not to have access to their mathematical description. Especially P or R are assumed to be unknown to the learning agent. RL methods can thus be divided into two classes, depending on whether they try to infer knowledge about the MDP or not. The former approaches are referred to as *model-based* while the latter are consequentially called *model-free*. We will here only consider model-free methods.

Given an MDP, the learning problem in our case then consists in finding an optimal policy $\pi : S \times A \to [0, 1]$ where $\pi(s, a) = p_\pi(a_t = a | s_t = s)$ again corresponds to a conditional probability distribution. For such a probabilistic policy and deterministic state transitions, as we assume in this thesis, the learning problem can be formalized as the maximization of the expected discounted total reward

$$\arg\max_\pi \mathbb{E}_\pi[R(\tau)] = \arg\max_\pi \int \pi(\tau) R(\tau) d\tau \tag{5.2}$$

$$= \arg\max_\pi \int \prod_{t=0}^{T} \pi(s_{\tau,t}, a_{\tau,t}) \sum_{t=0}^{T} \gamma^t R(s_{\tau,t}, s_{\tau,t+1}, a_{\tau,t}) d\tau \tag{5.3}$$

where $\tau = s_{\tau,0}, a_{\tau,0}, s_{\tau,1}, a_{\tau,1}, \cdots, s_{\tau,T}, a_{\tau,T}, s_{\tau,T+1}$ is a trajectory or rollout of the MDP with policy π. As shown above, we define $\pi(\tau)$ and $R(\tau)$ in terms of $\pi(s, a)$ and $R(s, s', a)$ at the

individual time steps. Note that π need not necessarily be Markovian in the sense that a_t only depends on s_t, although this is known to suffice in general. While in principle T can be infinitely large, in the context of this thesis we will assume it to be finite.

A policy that optimizes the above criterion thus yields the maximal expected discounted reward $\max_\pi \mathbb{E}_\pi[R(\tau)]$. If a certain initial state s_0 is fixed, this expression can be understood as a probabilistic generalization of the *value function*

$$V(s_0) = \max_{\{a_t\}} \sum_{t=0}^{T} \gamma^t R(s_t, s_{t+1}, a_t) \tag{5.4}$$

as defined in the context of the *Bellman equations* for dynamic programming [48, 49]. Indeed, finite MDPs of small to medium size can be solved for deterministic policies in a dynamic programming fashion. However, there also exists a plethora of RL methods which can deal with large or infinite sets of states and actions as well as probabilistic policies in absence of explicit knowledge of P or R. In the interest of brevity, we will however only elaborate on the method relevant for this thesis.

In principle it would seem like a natural idea to directly optimize the reward with respect to π. This would be especially convenient if π depended on a set of continuous parameters Θ which could be optimized with gradient-descend methods, using modern numerical frameworks. Unfortunately however, we can not assume the reward function to be given in a closed-form equation that can be differentiated with respect to Θ. In fact, the reward is generally treated as a black-box function in RL scenarios. Employing a probabilistic policy π_Θ however provides a remedy for this situation as in this case it holds

$$\nabla_\Theta \mathbb{E}_{\pi_\Theta}[R(\tau)] = \int \nabla_\Theta \pi_\Theta(\tau) R(\tau) d\tau = \int \pi_\Theta(\tau) \nabla_\Theta \ln \pi_\Theta(\tau) R(\tau) d\tau = \mathbb{E}_{\pi_\Theta}[\nabla \ln \pi_\Theta(\tau) R(\tau)]. \tag{5.5}$$

By sampling from π_Θ, this expectation value and thus the gradient of the expected reward with respect to Θ can be approximated. This identity is known as the *REINFORCE* trick [50] and the class of RL techniques making use of it are called *policy gradient* algorithms. Policy gradient methods are conceptually relatively close to supervised learning as they allow for standard gradient-based optimization of policies for instance parameterized by ANNs. For a more in-depth introduction to RL, we refer the interested reader to the standard work by Sutton et. al. [51].

5.2 Long short-term memory networks

We will now give an introduction to recurrent neural network (RNN) models. To this end, we begin by introducing the basic variant and then show how it can be extended to the long short-term memory (LSTM) networks used in this thesis. RNNs constitute a class of ANN architectures for the modelling of discrete time-series data of often non-Markovian nature where an observation $x_t \in \mathbb{R}^N$ at some time t is influenced by previous observations $x_{t-1}, \ldots, x_1$. Examples for successful applications can for instance be found in language-related tasks such as speech recognition, natural language processing and automated translation [52, 53, 54, 55].

A standard RNN consists of two non-linear functions $h_t : \mathbb{R}^N \to \mathbb{R}^H$ and $o_t : \mathbb{R}^H \to \mathbb{R}^O$ given by

$$h_t = tanh(Ux_t + Vh_{t-1} + b^h) \tag{5.6}$$

$$o_t = f^o(Wh_t + b^o), \tag{5.7}$$

where $U \in \mathbb{R}^{H\times N}$, $V \in \mathbb{R}^{H\times H}$, $W \in \mathbb{R}^{O\times H}$, $b^h \in \mathbb{R}^{1\times H}$, $b^o \in \mathbb{R}^{1\times O}$ and thus the set of parameters is $\Theta = \{U, V, W, b^h, b^o\}$. Note that the parameters do not depend on t. The non-linearity f^o is chosen depending on whether the output of the RNN poses the input to another layer in a larger network or is the final output. In the latter case, f^o must yield values in the domain expected by the loss function, which could, for instance, be formulated in terms of probabilities. More intuitively, an RNN maintains a hidden *state* h which encodes information about all previous time steps. At a given time t, this information is combined with the new observation x_t to yield the new state h_t. This combination is determined by the weight matrices U and V and the bias-vector b^h. Based on h_t, the output o_t for the current time step is then computed as determined by W, b^o and f^o. The dimensionality of the state clearly plays a central role in determining the expressive power or capacity of the RNN.

Training an RNN on time-series data is commonly done by taking tuples (x_t, x_{t+1}) from the data and letting the model predict x_{t+1}, given x_t. The distance between the prediction $\hat{x}_{t+1}$ and the target value x_{t+1} is in these cases normally given by an error function which is differentiable with respect to Θ, allowing for gradient-based optimization.

In theory an RNN is capable of keeping relevant information about all previous time steps in h_t and combining this information with the newly obtained x_t to generate the output o_t. In fact, certain RNNs have been shown to be Turing-complete [56]. In practice however, training the standard RNN model breaks down already for relatively small total times T. This is due to the fact that according to the product rule of calculus, the gradient of $\delta h_t/\delta V$ includes a product over all previous time steps. When the norms of the individual gradients are not very close to one, the overall gradient will quickly take on values of very large or very small magnitude. This is being referred to as the *exploding* or *vanishing gradient* problem.

To increase the stability of the gradients, Hochreiter et. al. introduced LSTM [57] networks, which can be understood as defining the state h_t in a slightly more involved way. An LSTM cell is given by the equations

$$i_t = sigm(U^i x_t + V^i h_{t-1} + b^i) \tag{5.8}$$
$$f_t = sigm(U^f x_t + V^f h_{t-1} + b^f) \tag{5.9}$$
$$g_t = sigm(U^g x_t + V^g h_{t-1} + b^g) \tag{5.10}$$
$$\tilde{c}_t = tanh(U^{\tilde{c}} x_t + V^{\tilde{c}} h_{t-1} + b^{\tilde{c}}) \tag{5.11}$$
$$c_t = c_{t-1} * f_t + \tilde{c}_t * i_t \tag{5.12}$$
$$h_t = tanh(c_t) * g_t \tag{5.13}$$

with x_t again being the input at time t, h_{t-1} being the previous state and c_t being referred to as the *cell state*. In relation to this, $\tilde{c}_t$ is often called the *candidate state*. Hereby, $U^i, U^f, U^g, U^{\tilde{c}} \in \mathbb{R}^{H\times N}$, whereas $V^i, V^f, V^g, V^{\tilde{c}} \in \mathbb{R}^{H\times H}$, $b^i, b^f, b^g, b^{\tilde{c}} \in \mathbb{R}^{1\times H}$ and $*$ denotes the Hadamard product. The sigmoid function is defined to be $sigm(x) = 1/(1 + e^{-x})$. As becomes clear by considering the above equations, the core idea of an LSTM is to equip the model with a better means of control over the information that is propagated through time. This control is implemented by the *gates* i_t, f_t and g_t which take values in $[0, 1]$ and, from an intuitive perspective, control the flow of information from and to the cell state c_t and the state h_t. As we will see below, the cell state c_t is of crucial importance for the ability of LSTMs to learn long-term dependencies. We will in the following refer to i_t as the *input gate*, to f_t as the *forget gate* and to g_t as the *output gate* and now briefly describe their function from an intuitive perspective.

- At time t, x_t and the previous state h_{t-1} are combined in the LSTM cell as in the standard RNN. The result is squashed to the interval $[-1, 1]$ via the tangens hyperbolicus to yield candidate values for the next cell state $\tilde{c}_t$.

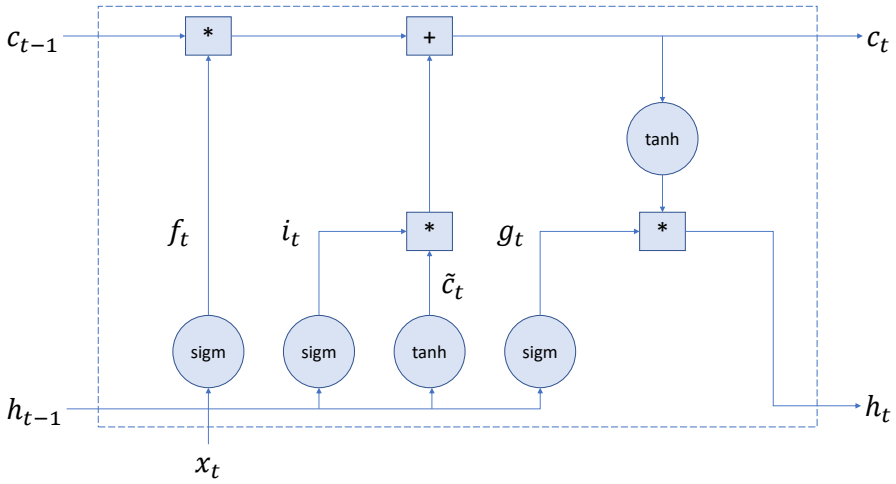


Figure 5.1: A graphical illustration of the long short-term memory cell.

- The input gate i_t now controls how much of the information from the candidate cell state enters the final cell state c_t. In the same way the forget gate f_t controls the information flow from the old cell state to c_t. The gated previous cell and candidate states are then added to yield the current cell state c_t.

- The cell state then again is brought to the interval $[-1, 1]$ by application of the *tanh* nonlinearity. The output gate g_t finally determines how much information of the squashed cell state is allowed to flow into the new state h_t of the LSTM unit.

We would finally like to point out two important consequences of this design. Firstly, by setting the forget gate f_t to zero and, the input gate i_t and output gate g_t to one, one almost obtains the state of the standard RNN. The only difference here is, that *tanh* is in fact applied twice. Secondly, when f_t is set to one but i_t is set to zero it holds that $c_t = c_{t-1}$ and thus the current input and previous state are ignored. This allows the LSTM to propagate a constant signal through time which in turn results in significantly more stable gradients. As a consequence, this specific configuration of the gates facilitates the training on long-term dependencies in the training data. It is called the *constant error carousel* and since the gate parameters are also subject to optimization, an LSTM cell is in principle able to learn when to function quasi like a standard RNN and when to ignore any new information and simply pass on an old cell state. The computations carried out within an LSTM cell are illustrated in Figure 5.1.

5.3 Restricted Boltzmann machines

As the second machine learning model of relevance for this thesis, we will now discuss the most important aspects of restricted Boltzmann machines (RBM). RBMs draw heavily from concepts of statistical physics [58] and have so far mainly been used as unsupervised learning models for the approximation of discrete probability distributions $P(v, h)$. Here, v is a configuration of *visible units* $v_1, v_2, \cdots, v_V$ and h is a configuration of the *hidden units* $h_1, h_2, \cdots, h_H$. Very importantly, both the v_i and h_i are hereby assumed to be binary valued variables, such that $v_i, h_i \in \{0, 1\}$. A particular configuration of visible and hidden units is then assigned an *energy*

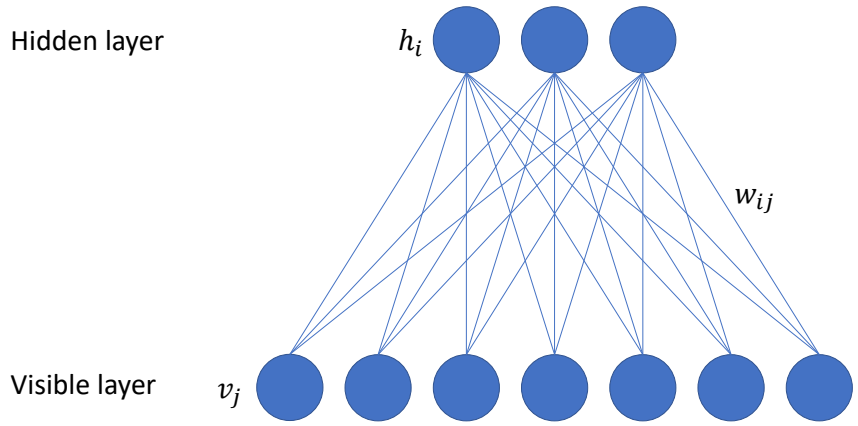


Figure 5.2: An illustration of a restricted Boltzmann machine.

given by

$$E(v,h) = -\sum_j a_j v_j - \sum_i b_i h_i - \sum_{i,j} w_{ij} h_i v_j \tag{5.14}$$

where $a \in \mathbb{R}^V, b \in \mathbb{R}^H$ and $W \in \mathbb{R}^{H \times V}$. This shows that RBMs can be direcly understood as classical spin systems. The energy $E(v,h)$ hereby is defined such that only interactions between visible and hidden units v_i and h_i are modelled but neither between visible or hidden units themselves. An RBM thus corresponds to an undirected bi-partite graph with the group of vertices v_i being called the *visible layer* and the group of vertices h_i being referred to as the *hidden layer*. This structure is illustrated in Figure 5.2.

An RBM thus is, as the name suggests, a restricted version of the more general *Boltzmann machine* which also allows for intra-layer connections. For given configurations of visible and hidden units, the probability $P(v,h)$ is then defined to be

$$P(v,h) = \frac{e^{-E(v,h)}}{Z} \tag{5.15}$$

where $Z = \sum_{v,h} e^{-E(v,h)}$ is the *partition function*. Note here that the partition function is generally intractable as it is an exponential sum. This is a fact that optimization and prediction algorithms have to account for. From this formulation it becomes apparent why this class of models is called restricted *Boltzmann* machines as they consitute a Boltzmann distribution. From a physical perspective, Equation 5.15 corresponds to a thermal state in equilibrium with unit temperature.

From the machine learning perspective, an RBM is now supposed to approximate the distribution a given training set $\{x_i\}$ was sampled from. In this context, the h_i model the correlations between different visible units v_i which correspond to the observed data x_i. To obtain the distribution over configurations v alone, we must consequentially marginalize over the hidden units such that we have

$$P(v) = \sum_h \frac{e^{-E(v,h)}}{Z}. \tag{5.16}$$

The particular structure of the interactions in the RBM then allows us to write $P(v)$ as

$$P(v) \propto \sum_h e^{-F(v,h)} \tag{5.17}$$

$$\propto \sum_h e^{\sum_j a_j v_j + \sum_i b_i h_i + \sum_{i,j} w_{ij} h_i v_j} \tag{5.18}$$

$$\propto e^{\sum_j a_j v_j} \sum_{h_1} e^{b_1 h_1 + \sum_j w_{1j} h_1 v_j} \sum_{h_2} e^{b_2 h_2 + \sum_j w_{2j} h_2 v_j} \dots \sum_{h_H} e^{b_H h_H + \sum_j w_{Hj} h_H v_j} \tag{5.19}$$

$$\propto e^{\sum_j a_j v_j} \prod_i (1 + e^{b_i + \sum_j w_{ij} v_j}) \tag{5.20}$$

$$\propto e^{\sum_j a_j v_j} \prod_i e^{softplus(\sum_j w_{ij} v_j + b_i)} \tag{5.21}$$

with the softplus-function being given by $softplus(x) = \log(1 + e^x)$. In the case of assuming $v_i, h_i \in \{-1, 1\}$, we can in the same way derive the corresponding expression for $P(v)$ to be

$$P(v) \propto e^{\sum_j a_j v_j} \prod_i cosh(\sum_j w_{ij} v_j + b_i). \tag{5.22}$$

This variant of the RBM is of importance for the results shown in Appendix F. In this context we also note that the probability is expressed as a product over the hidden units h_i.

The independence of the h_i and v_j also allows us to write $P(v|h) = \prod_j P(v_j|h)$ and $P(h|v) = \prod_i P(h_i|v)$. Having efficient expressions for the conditional distributions is important for the training of RBMs as the exact gradient is intractable but can be approximated via a *Gibbs-sampling* algorithm commonly known as *contrastive divergence* [59]. In a similar way, trained RBMs can for instance be used to complete partial data or sample new instances from the approximated distribution by performing Gibbs sampling. ML models which allow to generate new data following a learned distribution are referred to as *generative models*. This is in contrast to purely *discriminative models* that are for instance often used in classification and simply produce a target value for a given input.

6 Discussion

In this concluding chapter, we will discuss the main results of this thesis against the background of related work. The chapter is structured based on the employed numerical methods, such that we will discuss the results of Appendices A to C in a first part focussing on our work in the context of tensor networks. In a second part, we will then elaborate on the machine learning oriented work shown in Appendices D and E. In a short final section, we discuss the work shown in Appendix F as it combines methods from ML and TNs.

6.1 Tensor networks

For quantum states represented by density matrices ρ, as introduced in Chapter 2, many quantities of interest naturally take the form $\mathsf{Tr} f(\rho)$ with $f : \mathbb{C}^{2^L \times 2^L} \to \mathbb{C}^{2^L \times 2^L}$. An important example for instance is the von Neumann entropy

$$S(\rho) = -\mathsf{Tr}\rho \log \rho \tag{6.1}$$

which is a direct generalization of the entropy as defined in information theory and classical statistical mechanics to the quantum regime. In this particular case then, $f_S(\rho) = -\rho \log \rho$ and thus $S(\rho) = \mathsf{Tr} f_S(\rho)$. Another interesting example is the trace norm or, more formally, Schatten 1-norm of a state ρ given by

$$\|\rho\|_1 = \mathsf{Tr}\sqrt{\rho^\dagger \rho} \tag{6.2}$$

where consequentially $f_1(\rho) = \sqrt{\rho^\dagger \rho}$ and we hence have that $\|\rho\|_1 = \mathsf{Tr} f_1(\rho)$. The trace norm can for instance be used as a distance measure between two states ρ_A and ρ_B. This class of functions has the defining characteristic that they depend on the entire spectrum of ρ in a non-trivial way, like the logarithm or square root in the examples above. This is opposed to, e.g., the Frobenius norm $\|\rho\|_F = \sqrt{\mathsf{Tr}\rho^\dagger \rho}$, in which the trace is simply a sum over the squared singular values. We also use the term *global* to refer to properties which are a function of the entire spectrum of a given matrix or more generally to signal that the entire mathematical object is considered instead of individual parts in a decomposition.

As we have seen in Chapter 3, TNs rely on the local decomposition of quantum systems rather than a global description. Still, we have argued that basic linear algebra operations can be performed in the TN framework and thus also global quantities like the Frobenius norm based solely on these operations can be approximated. However, it is not directly possible to, e.g., compute the logarithm of a matrix product operator

$$\log\left(\sum_{i_1,\ldots,i_L,j_1,\ldots,j_L}^{d} (A_1^{i_1 j_1} A_2^{i_2 j_2} \cdots A_L^{i_L j_L}) \cdot (e_{i_1} e_{j_1}^T) \otimes (e_{i_2} e_{j_2}^T) \otimes \cdots \otimes (e_{i_L} e_{j_L}^T)\right) \tag{6.3}$$

or any other non-linear function for that matter. Even if it were possible to obtain the exact spectral decomposition of a given MPO, computing a function of it would still be intractable as the number of eigenvalues is in general exponential in L. Thus, even though functions of the type $\mathsf{Tr} f(\rho)$, or more generally of an MPO A, are of high relevance in quantum physics,

there existed no method to approximate such functions for many-body systems represented as MPOs.

In the work shown in Appendix A, we presented the first method to efficiently approximate functions $\mathsf{Tr}f(A)$ of Hermitian MPOs A for analytic functions f. The presented method combines the global Lanczos algorithm as introduced in Chapter 4 with the TN framework to perform a Gauss quadrature of the Riemann-Stieltjes integral representing $\mathsf{Tr}f(A)$. Hermiticity hereby is a requirement that is naturally fulfilled by operators in quantum physics but does also arise in other areas of application such as ML or network analysis. The requirement for f to be analytic ensures its approximability by quadrature algorithms.

While the Lanczos algorithm has already been used in the TN formalism to for instance approximate ground states [60] and time evolution [31, 61, 62, 32, 63], block variants were so far not employed to the best of our knowledge. An important reason for this is the fact that most block Krylov algorithms define the orthogonality criterion of their basis blocks in terms of the orthogonality of the individual columns of these blocks, see for instance Refs. [36, 35]. These formulations are thus susceptible to the curse of dimensionality in its quantum variant, meaning that these algorithms are in principle intractable for block sizes scaling more than polynomially with the size of the quantum system.

As we discussed in Chapter 4, the global block Lanczos [39] algorithm poses an exception to this rule as it makes use of an inner product which can be efficiently expressed in the TN format even for *block sizes exponential in the system size*. This algorithm was however previously unknown in the numerical QM community. In contrast to the standard matrix version of the global Lanczos algorithm, our TN algorithm now allows us to use *unitary starting blocks of size equal to that of A*. Indeed, the natural choice of the identity matrix has an *exact* MPO representation with minimal bond dimension $D = 1$ and so we have in our method that

$$\mathsf{Tr}(U^* f(A) U) = \mathsf{Tr} I f(A) I = \mathsf{Tr} f(A) \tag{6.4}$$

where $U = I$ is the starting matrix of our method and we denote the conjugate transpose by $*$ to stress the generality of the expression. While possible theoretically, such a choice of U is not considered in block Algorithms for full matrices as their underlying assumption is that it is only computationally feasible to perform matrix-vector multiplication or matrix-matrix multiplication for blocks with a significantly lower number of columns than that of A. In these cases it thus only holds $\mathsf{Tr}U^* f(A)U \approx \mathsf{Tr}f(A)$ and the integral approximated by the Gauss quadrature merely poses an *approximation* to the true integral.

We again note here that there exists work in numerical mathematics on approximating $\mathsf{Tr}f(A)$ for large full matrices A. These methods generally work by sampling starting vectors u to approximate $u^* f(A)u$ with Krylov methods and essentially computing Monte Carlo estimates of $\mathsf{Tr}f(A)$. The field is based on the results of Golub et. al. [41] we also made use of in Chapter 4. Apart from the work on the global Lanczos algorithm, other interesting related work was for instance presented in Ref. [64]. Before discovering the method presented here, we investigated such sampling-based approaches and were able to reformulate some of them in the TN formalism. However, we generally found the number of samples required for good approximations infeasible already for relatively small system sizes.

The combination of the global Lanczos algorithm with the TN framework hence allowed us to firstly address problems of size intractable before and secondly opened up the possibility to approximate the true integral. It furthermore allows for the incorporation of new results both from the side of TNs as well as from the side of numerical analysis, as for instance from Ref. [65]. However, it is clear that the method still only consitutes an approximation. This has two main reasons. Firstly, the number of Krylov dimensions K, which directly corresponds to the number of nodes used in the Gauss quadrature, must be limited to a feasible amount. This

amount is likely lower than the number of nodes required for exactness. Secondly, although we start with an exactly represented initial matrix, the sequentially constructed basis matrices can in general only be approximated by an MPO, since we need to define an upper bound D_{max} for the bond dimension used in a run of the algorithm. This in turn implies that the accuracy of the method can be controlled by adjusting K and D_{max}.

Indeed, we showed that the method converges to the exact result in absence of approximation errors and assuming a sufficently large K. One could in fact argue that D_{max} is the most important parameter of the method as the approximation error in the basis matrices influences the degree to which the U_i are orthogonal, which in turn determines how well the basis of the Krylov space is approximated.

It is clear that the values of K and D_{max} required for a specific accuracy depend on the MPO A and the function f. As is also indicated by the numerical results we presented in the respective work, if f for instance is such that only a few eigenvalues of A are relevant to determine $f(A)$ with high accuracy, already relatively small errors in the nodes of the quadrature might cause large deviations. Additionally, more quadrature nodes might be required to accurately represent the relevant part of the spectrum. On the other hand, the bond dimensions of A and its structure determine how well its powers computed during a run of the algorithm can be approximated with a given bond dimension.

To gain a better understanding of how the structure of A influences the partial results computed by the method, we subsequently conducted an analysis of their relation for the case of exact representations. The obtained results were presented in the work shown in Appendix B. One of the main findings of this analysis is that the basis matrices U_i as computed by the method indeed exhibit several properties derived from A, such as for instance symmetry/hermiticity, centro-symmetry/centro-hermiticity or certain commutation relations. This is relevant for the computation with MPO approximations insofar as certain symmetries and other properties can be checked for efficiently to monitor approximation errors and in some cases can be enforced. Furthermore, it is well-known [66, 32] that certain symmetries can be exploited to reduce the required bond dimension and thus obtain more efficient and accurate representations.

As another major result of the analysis, we found that for input MPOs A with a spectrum that is symmetric around zero, the tridiagonal matrix T_K takes the form

$$T_K = \begin{bmatrix} 0 & \beta_2 & & \mathbf{0} \\ \beta_2 & 0 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & 0 \end{bmatrix} \tag{6.5}$$

as it holds in this case that all $\alpha_i = 0$. This implies that for such an input, basis matrices U_i and U_{i-1} are orthogonal by construction. While this property can clearly be monitored at runtime, it also yields a more efficient version of our method as in this case U_i in principle must not be orthogonalized against U_{i-1}. We note here that these results also hold for the global Lanczos [39] algorithm with full matrices. As our underlying assumptions about the size of the U_i are however conflicting with the assumptions for the full matrix case, our particular setting was to the best of our knowledge not considered so far.

Having obtained the above insight, we then continued to investigate which kinds of Hamiltonians as possible inputs for our method exhibit the required spectral property. Here, we did in fact find a large class of spin Hamiltonians with finite-length neighbour interactions to have a spectrum symmetric around zero. A prominent example of this class is, e.g., the transverse-field Ising Hamiltonian, a special case of the Ising Hamiltonian shown in Chapter 2. This part of our analysis can be considered an extension of the one presented by Kressner et. al. in Ref. [67].

Our interest in Hamiltonians stemmed from the fact that, although we introduced the method in terms of the approximation of functions of states ρ, we did in fact find it to be possible and very advantageous to directly consider Hamiltonians as inputs in certain cases. More concretely, during the search for a first proper application of our method in QM, we realized that it could be beneficially applied to the approximation of properties of thermal equilibrium states. The thermal equilibrium or Gibbs state of a Hamiltonian H is given by

$$\rho(\beta, H) = \frac{e^{-\beta H}}{Z} \tag{6.6}$$

where β corresponds to the inverse temperature and $Z = \mathsf{Tr} e^{-\beta H}$ is the partition function. This is precisely the form of probability distribution we have already encountered in Chapter 5 for RBMs. These states are normally approximated by an algorithm commonly referred to as *imaginary time evolution* [31, 18, 19] which yields a TN representation of ρ for a given bond dimension.

While this TN approximation of ρ could in its MPO form be used as input of our method, it has two main problems. Firstly, the input only consitutes an approximation to the exact state we would like to approximate a property of. This naturally reduces the accuracy of the approximation. Second, it likely has a bond dimension which is already relatively high, slowing down the computation significantly. Making use of the generality of our algorithm, we realized that it is of course possible to express properties of Gibbs states as properties of the generating Hamiltonians. As an example, we can write the von Neumann entropy of a thermal equilibrium state $\rho(\beta, H)$ as

$$S(\rho(\beta, H)) = -\mathsf{Tr}\frac{e^{-\beta H}}{\mathsf{Tr} e^{-\beta H}} \ln \frac{e^{-\beta H}}{\mathsf{Tr} e^{-\beta H}} = \beta \frac{\mathsf{Tr} e^{-\beta H} H}{\mathsf{Tr} e^{-\beta H}} + \ln \mathsf{Tr} e^{-\beta H} = \beta \frac{F}{Z} + \ln Z \tag{6.7}$$

where $F = \mathsf{Tr} e^{-\beta H} H$ and Z again denotes the partition function. We can thus approximate the von Neumann entropy of the state directly from H by means of approximating F and Z and combining the approximations as shown above. As F and Z are functions of the same Hamiltonian, we can furthermore approximate both in a single run of the algorithm since they only differ in the function that is applied to the projected matrix T_K. Even more so, we can in principle approximate the entropy for a whole range of temperatures within one run of the method, as β is simply an argument to the function of H. This parallelization strategy to some extend makes up for the fact that the algorithm itself is clearly of sequential nature and parallelization can thus only be employed in the core linear algebra routines.

The main advantage of computing the approximations directly from the Hamiltonian however is that many Hamiltonians have an *exact* expression as MPOs with often very low bond dimension of $D = 2$ or $D = 3$, as already hinted at in Chapter 2. We thus can start the approximation from an exact input and have significantly faster computations. While it is true that in this case we generally have to add multiple terms with an approximation error, we found the gain in accuracy resulting from the exact input to outweigh this potential disadvantage. Additionally, a lower bond dimension of the input allows us to use higher values of D_{max} in the same runtime. In this context, we also found that our method can be used to approximate expectation values $\mathsf{Tr} O f(A)$ of certain positive operators O. For positive operators, it is possible to write $\mathsf{Tr} O f(A) = \mathsf{Tr}\sqrt{O}^* f(A)\sqrt{O}$. Due to the positivity of O, this expression again yields a Riemann-Stieltjes integral that can in principle be tackled via Gauss-quadrature. If O admits an efficient MPO representation of $\sqrt{O}$, we can then simply use $\sqrt{O}$ as starting matrix of the algorithm to approximate the expectation value.

We used these findings in Appendix C to demonstrate how our method can be employed to detect signals of a thermal phase transition for the Lipkin-Meshkov-Glick [68, 69, 70] (LMG)

Hamiltonian and approximate two-particle correlation functions for the transverse Ising Hamiltonian. The LMG Hamiltonian can be successfully studied analytically [71, 72, 73] which allowed us a comparison to reference results. On the other hand, due to the Hamiltonian exhibiting long-range interactions, its Gibbs states can not be approximated efficiently via imaginary time evolution. Hence, it was out of reach for TN methods thus far. Since our method does not require an MPO representation of ρ, it is unaffected by this property of the LMG or other Hamiltonians. It thus poses the first TN method to efficiently approximate properties of thermal equilibrium states with long-range interactions.

Having an MPO representation of ρ when it is feasible however is helpful, e.g., for the computation of two-particle correlation functions. This is due to the fact that the correlation between each pair of particles is given by a different operator. In this case, we were however still able to show that both the imaginary time evolution approach and our method produced very similar results for the transverse Ising Hamiltonian.

Although we have at the time of this writing introduced, analyzed and successfully demonstrated our method, more can still be done. Obviously, it would be interesting to identify and engage further promising areas of application in quantum many-body physics. However, from an algorithmical perspective it would also be worthwhile to see how far we can exploit the symmetry properties of the basis MPOs U_i to obtain more efficient and accurate results. Another interesting line of work would be to combine our algorithm with established methods to approximate extremal eigenvalues of MPOs. These approximations would likely yield more accurate (extremal) nodes for the Gauss quadrature performed in our method. This might help especially for Gibbs states ρ close to the ground state as in these cases only a few eigenvalues or energies still play a role. Finally, it would certainly also be interesting to explore what other insights from numerical analysis can be incorporated into our method.

6.2 Machine learning

Building an automated scientist has been a dream of some researches [74, 75] since the dawn of the computer era. With artificial intelligence (AI) having made drastic progress in recent years and now excelling in problems previously believed out of reach for the foreseeable future [76, 77, 78, 79, 80], this dream might today seem much closer to reality. But one may well ask the question if scientific discoveries made by machines can be understood by humans in the same way as if they had walked the path to these discoveries themselves. And if this is not the case, what is the value of a discovery not fully understood by humans? Are we willing to deligate scientific discovery to machines to speed up progress while sacrificing understanding, control and responsibility? These questions certainly are far-fetched given today's state of the art in machine learning, as the most successful branch of AI. At the time of this writing, it does so far lack any serious semantic reasoning capabilities as would certainly be required to do science. We would however like to point out that it is our opinion that science should always remain a human endeavor as it is, or should be, done to satisfy human curiosity and improve human and other animal's lives. The work presented here is thus not meant to replace humans in the research pipeline.

Having made our position on this issue clear, we are however convinced that ML can play an important role in supporting humans in many routine tasks of scientific and engineering work. Furthermore, in some cases it might even be a necessary requirement for progress to perform, e.g., a post-processing of scientific data via ML techniques as can for instance be seen in experiments with the Large Hadron Collider (LHC) [81, 82, 83]. Related to this example, one task in physics that lends itself especially well to treatment with ML approaches is the calibration or optimization of experiments.

While the outcome of experiments is of scientific relevance, obtaining the desired results often involves careful calibration of parameters, as for instance strengths of magnetic fields, whose precise values are not of high interest. Furthermore, especially for more complex experiments with many interacting parts, there likely exists no accurate analytical model which could be used to obtain optimal parameters. The proper calibration of an experiment thus can be a tedious task of only secondary interest to the experimental physicist. In this case, finding an optimal configuration of parameters is more important than fully understanding the reasons for its optimality. This is why the optimization of such experimental control variables poses a prime use-case for automated methods aiming to support rather than replace scientists.

Indeed, this has already been realized by experimental physicists and examples for automated optimization of experimental parameters range from gradient-based optimization [9, 84] over evolutionary algorithms [85] to, more recently, ML approaches [86, 87, 88, 89, 90]. From the computer scientific perspective, the existing methods however, while certainly successful in achieving their respective goals, leave room for improvement. Firstly, the ML-based methods introduced so far often make use of proprietary algorithms and models tailored to a specific application and effectively ignore the recent developments in machine learning. Partly as a consequence of this, these approaches then in some cases only consider simplified versions of the learning problems by for instance discretizing continuous parameters. Secondly, pure gradient-based optimization methods require an analytical model of the given experiment. As we have already argued above, such a model will likely only be available for comparatively simple experimental setups. An exact analytical model can additionally only be used for relatively small system sizes if the dimension of the Hilbert space can not be reduced by exploiting symmetries. Furthermore, gradient-based optimization methods typically only yield at most a small set of solutions which are in many cases not guaranteed to be optimal. To obtain new solutions, the optimization process often has to be restarted from scratch. Thirdly, evolutionary or hill-climbing algorithms typically converge relatively slowly.

In the works presented in Appendices D and E, we took a step towards solving these problems by introducing a novel general method to find optimal parameters in experiments. More precisely, we showed how to approximate the probability distribution of optimal or near-optimal sequences of control parameters by LSTM networks and introduced two algorithms to train these representations. As a concrete and important area of application in quantum physics, we hereby focussed on the problem of quantum control as introduced in Chapter 2. Our method thus constitutes an automated way of optimizing sequences of quantum control parameters by employing state-of-the-art machine learning models. It is designed such that it could also be used in conjunction with, e.g., gradient-based optimization methods by integrating their results into the training process of the LSTMs.

We based the development of our method on four main assumptions.

1. Optimal or near-optimal sequences of parameters for a given control problem share common structure. This is a necessary requirement for a ML model to be able to learn and justified by the high amount of structure often found in physical phenomena.

2. An analytical model of the control problem can not be assumed to be available or feasible to approximate. This reflects our goal of the methods to be generally applicable, also for more complex settings. In an ML sense, our methods are thus *model-free*. This black-box assumption also allows our method to find solutions tailored to a given hardware setup in a real experiment and not to an idealized model of it.

3. The figure of merit or reward function R can only be evaluated for entire sequences. By the laws of quantum physics, a measurement makes certain information of a quantum state permanently inaccessible. As evaluating an error function in general involves some

measurement, this can hence only be done after the controlled quantum time evolution is complete. From an RL perspective, this assumption results in a very sparse reward signal. Our method can however easily incorporate any kind of intermediate reward signal if available.

4. Repeated execution of a given control scenario is feasible such that sampling of results for given inputs is possible. In quantum control terminology, this corresponds to a *closed-loop* scenario.

Based on these assumptions it becomes clear that the ML model employed in our method must be able to capture structured time-series data as which the sequences of control parameters can be perceived. From an ML perspective, this quite naturally leads to the choice of long short-term memory networks as discussed in Chapter 5. The inherent property of ANN architectures to be stacked also potentially allows an LSTM network to consider structure at different time scales. While there do also exist similar RNN architectures [91], LSTMs are arguably the most established and so far no other superior architecture has been introduced [92]. Despite LSTMs being a natural fit, they were to the best of our knowledge previously not discussed in the numerical physics literature. Our work was thus the first to employ these models in numerical quantum physics. Note that LSTMs can also be used to simply model dependencies between variables, such that even experimental parameters which do not constitute a time series but can be thought of as sequentially dependent on each other can potentially be treated with this model.

Apart from the choice of the machine learning model, another fundamental and related decision is the modelling of the learning problem. In our work, we decided to treat the problem probabilistically in that the ML task was defined to be the approximation of the probability distribution of optimal and near-optimal sequences. This decision was made to reflect the fact that due to assumptions 2 and 3 there must be uncertainty about the optimal value of a control parameter at a given time step. Additionally, learning a distribution over the good sequences parameterized by an LSTM allows us to efficiently sample from it and examine the distribution's properties. This in turn facilitates a deeper analysis of the structural properties of the good solutions if desired. Furthermore, a trained LSTM representing the distribution of good solutions for a given control problem can be used as starting point for related control problems which likely yield similar solutions.

In contrast to pure gradient-based optimization methods, our ML method thus does not necessarily need to start from scratch for each new problem and can in principle even cope with changing conditions in an experiment. In this context it is noteworthy that the flexibility of LSTMs and ANN architectures in general allows them to parameterize any probability distribution that is sufficiently characterized by a finite number of parameters. This means that for discrete distributions such as for instance the Bernoulli or Categorical distributions, the networks can simply output an array of discrete probabilities $p_i \in [0, 1]$ with $\sum_i p_i = 1$ in the latter case. For continuous distributions, the network on the other hand can simply predict the required parameters like, e.g., the mean μ and variance σ^2 for the normal distribution. Note that with an LSTM we can approximate the full distribution over the time-series data $p(x_1, x_2, \cdots, x_T) = \prod_t p(x_t|x_1, x_2, \cdots, x_{t-1})$ as they are not limited to the Markovian case.

The final important question then was how to actually train the LSTMs given the starting condition that we assumed to neither have access to a gradient of the error function, nor to be in possession of a data set from which the distribution could be learned.

In the first work shown in Appendix D, we provided a possible answer at the example of dynamical decoupling for quantum memory. In this setting, a qubit or a system of qubits which is prepared in a certain state is coupled to an environment of other particles. The simple objective then is to preserve the qubit system's state over time. This is not a problem in

classical computers but in quantum computation, the interaction between the system and its environment causes detrimental effects such that information is eventually lost. One possibility to counter this effect is known as *dynamical decoupling* [93, 94] (DD) where the system is actively manipulated over time such that it is in the best case completely separated or decoupled from the environment. Under certain ideal assumptions such sequences are known to be optimal. The decoupling is often achieved by applying sequences of the Pauli matrices, including the identity, which can be perceived as rotations of the system's state. It had previously been shown that good DD sequences can be found via simple evolutionary algorithms [85]. Inspired by this result, we introduced an evolutionary-style algorithm to train the LSTMs, which in essence follows a nested optimization loop consisting of two steps:

1. Sample data set D_{new} from $p_\Theta(c)$, evaluate it with respect to R and combine it with the old data D_{old} to create a new training set D from $D_{new} \cup D_{old}$.

2. Maximize $\sum_{c \in D} \ln p_\Theta(c)$.

Hereby $p_\Theta(c)$ denotes the probability distribution over the sequences c which is parameterized by the LSTM weights Θ. The new training data D is obtained by sorting the set $D_{new} \cup D_{old}$ according to the sequences' quality as measured by the reward $R(c)$ and taking the best N sequences. Thus, the first step optimizes $R(c)$ with respect to D while the second step optimizes the likelihood of D with respect to the LSTM parameters Θ. Note that of course also regularization can be applied here. From the perspective of evolutionary algorithms, our training algorithm can thus be perceived as replacing the common mutation and crossover operations by sampling from a probability distribution which depends on the data of the previous iteration. A similar approach can for instance be found in Ref. [95] but to the best of our knowledge, LSTMs had previously not been used in this way. We would also like to point out here that solutions obtained from other control methods can be easily integrated into the training process by augmenting the sampled data with them.

In numerical experiments, we then evaluated our method for a particular quantum memory scenario. This scenario was chosen such that it violated some idealistic assumptions, needed to show optimality of the DD sequences, in favor of a more realistic setting. Our interest here naturally was to investigate if our method was able to compete with existing analytically optimal solutions. Another important aspect of these experiments however was to examine if the deviation from the ideal setting would allow our automated method to improve upon the analytical solutions. We indeed found our method to beat all considered classes of DD sequences for the examined control scenario. Furthermore, we found that replacing the LSTM by a simple n-gram model, which just approximates the conditional distribution for the last n time steps, resulted in worse outcomes. These findings thus showed that the generality of our method allowed it not only to perform well on the given problem but also to improve over solutions based on idealized assumptions. They also demonstrated the usefulness of LSTMs when compared to simpler probabilistic models. The fact that the LSTM networks were so successful in learning also implies the correctness of assumption 1 at least for the considered quantum memory scenario. However, while the work provided some justification for our ideas on ML-based quantum control, the optimization method itself was still rather simple and not as data efficient as would be desirable. Additionally, we had in this work dealt with sequences of Pauli matrices which resulted in a learning problem with discrete control parameters. We thus had so far not considered control problems with continuous parameters. In the subsequent work shown in Appendix E, we improved on this situation by employing tools from reinforcement learning.

After finishing the work described above, we realized that policy gradient methods as briefly descriped in Chapter 5 can be employed to approximate the gradient of R with respect to the

LSTM parameters of a given distribution. In fact they quite naturally fit to our previously stated assumptions 2 and 4. Due to their more sophisticated exploitation of the sampled data, they were also likely to be more efficient in training the LSTM networks. While first RL approaches were already discussed in the literature [86, 87], it had so far not been realized that the policy gradient approach could be used. We reviewed the literature on policy gradients and chose to base our improved method on the proximal policy optimization (PPO) algorithm introduced by Schulman et.al. [96]. One major contribution of the article was to introduce a new loss function for the policy gradient update that significantly increased its robustness while being easily applicable. Optimizing the loss was shown to lead to new reference results for various benchmark RL tasks.

To introduce our finding to a broader audience, we discussed and analyzed our approach to the quantum control problem from a reinforcement learning perspective. Based on this, we then introduced an improved version of the PPO algorithm. The main improvement hereby consisted of employing the agent with a memory of the best observed sequences. The memory is used in conjunction with newly sampled data to compute an update of Θ. We argued this to improve the convergence behavior of our method and obviously prevent good solutions from being lost. Furthermore, we showed how to use the memory to dynamically scale variance parameters of the continuous distributions employed in this work. When sampling from a probabilistic model of the sequences it is clear that a smaller variance results in the sampled sequences being closer to the predicted mean sequence, while a larger variance causes the opposite behavior. The variance of the employed distributions is thus a way to control the *greediness* of the learning agent and hence of great importance for the result of the optimization.

Employing the policy gradient approach and introducing our specific improvements then resulted in the following new abstract optimization loop:

1. Sample new data D from $p_\Theta(c)$, evaluate it with respect to R and update memory M.
2. Optimize $1/|D \cup M| \sum_{c \in D \cup M} L_{MPPO}(c, \Theta)$ for a fixed number of iterations.
3. Adapt variance parameters based on M.

Here, $L_{MPPO}(c, \Theta)$ denotes the new loss function

$$L_{MPPO}(c, \Theta) = \min(r(\Theta, c)A(c), \mathrm{clip}(r(\Theta, c), 1 - \epsilon, 1 + \epsilon)A(c)) \tag{6.8}$$

with $r(\Theta, c) = p_\Theta(c)/p_{\Theta_{old}}(c)$ and Θ_{old} denoting the weights of the previous iteration of the loop. The function $\mathrm{clip}(x, a, b)$ simply ensures $a \leq x \leq b$ and $A(c) = R(c) - R(c_{opt})$ with c_{opt} being the best sequence in the memory. The parameter ϵ controls the magnitude of $r(\Theta)$ and optimizing $L_{MPPO}(c, \Theta)$ thus results in a bounded conservative update of Θ. Note that in the second step of the loop, the loss is typically only optimized for a small number of iterations.

To evaluate the validity of the improved method, we presented results for the quantum memory scenario which we had already tackled in the previous work. We also considered two generalizations of it to continuous control parameters. We found the improved method to converge to good solutions relatively fast for all considered learning problems and were able to reproduce the results of the previous work in the discrete case. Furthermore, we provided a comparison of our method to another RL algorithm recently introduced in the context of controlling the transition of quantum systems between ground states [86]. Here we were also able to reproduce the reference results, although unfortunately the respective method can only tackle discrete problems. We also showed for this case how generalizations of the control problem to the continuous domain can be solved by our method.

For all the considered learning tasks, we explicitly demonstrated how our general method can be combined with insights about the control problems at hand. We found doing so very

advantageous for the learning behavior, especially for the continuous control tasks. While we explicitly made use of assumption 3 in this work, we also demonstrated how our method can be easily adapted to the optimization not over entire sequences but for instance parameters of individual time steps. For the considered learning tasks we did however find this to lead to worse results than optimizing over the complete sequences. Finally, we provided some further evidence for the correctness of assumption 1 by demonstrating the common structure of the best performing sequences for several of the considered control settings.

In conclusion, we thus found that combining LSTMs with our memory proximal policy optimization (MPPO) algorithm provides a versatile method to solve both discrete and continuous quantum control tasks. As we have seen, this even holds true in the challenging situation when no analytical model of the given problem is available and information about the quality of a control sequence can only be obtained for entire sequences. Indeed, shortly after the publication of our work, another article appeared that showed how to solve a different quantum control problem by a combination of LSTMs and standard policy gradients [97]. Not surprisingly, the article shares many of our ideas and arguments for the use of LSTMs and policy gradients in quantum control, especially for real experiments. This article was followed quickly by work [98] on using bi-directional LSTMs for yet another quantum control problem. These developments in combination with our own results suggest that the ideas developed in the context of this thesis do in fact pose a valid and promising approach to automated quantum control as an example of the ML-based optimization of experiments. As we employed numerical simulations of the control settings in absence of experimental implementations, our work can furthermore be seen as a successful example of the combination of classical simulation and machine learning in science.

There are however still many more possibilities for future work. Firstly, it would be very interesting to apply our method to an experimental implementation of a control problem. Secondly, there are still many ML methods which at the time of this writing have not been explored in this context but might well prove highly useful. Thirdly, it would be useful to establish a set of benchmark problems to properly evaluate and compare existing and new methods. Finally, there exist rigorous results on the controllability of quantum systems. This knowledge should not be ignored in real applications and it should be investigated how to best combine it with machine learning approaches. Generally, as the entire field of quantum machine learning is still in a very early phase, there is much room for exploration.

6.3 Tensor networks and machine learning

While tensor network methods such as MPS and MPO have been very successful in quantum many-body physics, the field naturally seeks to improve upon existing methods. One interesting attempt was recently made by Carleo et. al. [99], who showed how to approximate quantum states of spin systems with RBMs. In this picture, the state of a system of spins $s_1, s_2, \cdots, s_L$ is expressed by

$$|\psi\rangle = \sum_{s_1,s_2,\cdots,s_L} e^{\sum_j a_j s_j} \prod_{i=1}^{H} 2cosh(\sum_{j=1}^{L} w_{ij} s_j + b_i)/Z \, |s_1, s_2, \cdots, s_L\rangle \tag{6.9}$$

where the amplitudes of the wave function are given by the spin-1/2 formulation of the RBM as shown in Chapter 5. Here, H denotes the number of hidden units. It was shown how this parameterization can for instance be used to approximate ground states by employing variational Monte Carlo (VMC) techniques [100, 101]. Using an RBM as parametrization of the wave function has the advantage that the hidden units h_i are connected to all spins. The

RBM thus yields a more global description of a system as, e.g., compared to MPS which are inherently local as we have seen. These results quickly motivated further work on the subject [102, 103].

In the work of Appendix F, we discovered an unexpected relationship between certain kinds of tensor networks and RBM states. At the core of this discovery lies that fact one can rewrite the above expression as

$$|\psi\rangle_{s_1,s_2,\cdots,s_L} = e^{\sum_j a_j s_j} \prod_{i=1}^{H} 2cosh(\sum_{j=1}^{L} w_{ij} s_j + b_i)/Z \tag{6.10}$$

$$= 2/Z e^{\sum_j a_j s_j} \prod_{i=1}^{H} \mathsf{Tr} \begin{pmatrix} e^{\sum_{j=1}^{L} w_{ij} s_j + b_i} & 0 \\ 0 & e^{-\sum_{j=1}^{L} w_{ij} s_j - b_i} \end{pmatrix} \tag{6.11}$$

$$= 2/Z e^{\sum_j a_j s_j} \prod_{i=1}^{H} \mathsf{Tr} \left[\prod_{j=1}^{L} \begin{pmatrix} e^{w_{ij} s_j + b_j/L} & 0 \\ 0 & e^{-w_{1j} s_j - b_j/L} \end{pmatrix} \right] \tag{6.12}$$

$$= 2/Z e^{\sum_j a_j s_j} \prod_{i=1}^{H} \mathsf{Tr} A_{i,1}^{s_1} A_{i,2}^{s_2} \cdots A_{i,L}^{s_L} \tag{6.13}$$

where $s_j \in \{-1, 1\}$ as discussed before. The physical index s_j of the core tensors simply denotes whether $s_j = 1$ or $s_j = -1$ in the respective 2×2 matrix. This expression is proportional to a product of MPS and in fact corresponds to a special case of string bond states (SBS) [104, 105]. Furthermore, the fact that the matrices in the core tensors are diagonal shows that by allowing for non-diagonal matrices, a generalization of RBM states can be obtained. These findings motivated a further analysis and indeed we found several connections between RBMs and SBS as well as other state representations.

We were additionally able to provide numerical evidence for that fact that RBM states and their generalizations can approximate ground states of chiral Hamiltonians [106, 107, 108], as an especially challenging numerical problem, better than previously possible with TN methods. We however have to point out that the optimization of RBM states via VMC is very demanding from a numerical perspective and much less efficient than optimizations in the TN framework.

Our results have subsequently been referred to in several works, see for instance Refs. [109, 110, 111], and indicate that RBMs will certainly become more relevant for the simulation of many-body quantum systems in the future. The connection between RBMs and TNs we found might also yield a new direction for the application of TN techniques in machine learning. Finally, our findings provide an illustrative example of the closeness of many seemingly independent concepts employed in machine learning, tensor networks and quantum physics.

Bibliography

[1] C. Cohen, B. D. Tannoudji, and F. Laloë. *Quantum mechanics*. Hermann and Wiley, 1977.

[2] J. J. Sakurai and J. Napolitano. *Modern quantum mechanics*. Cambridge University Press, 2017.

[3] J. Eisert, M. Cramer, and M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Reviews of Modern Physics*, 82(1):277, 2010.

[4] M. B. Hastings. An area law for one-dimensional quantum systems. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(08):P08024, 2007.

[5] F. Verstraete and J. I. Cirac. Matrix product states represent ground states faithfully. *Physical Review B*, 73(9):094423, 2006.

[6] M. B. Hastings. Solving gapped hamiltonians locally. *Physical Review B*, 73(8):085115, 2006.

[7] S. R. White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863, 1992.

[8] S. Rommer and S. Östlund. Class of ansatz wave functions for one-dimensional spin systems and their relation to the density matrix renormalization group. *Physical Review B*, 55(4):2164, 1997.

[9] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser. Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms. *Journal of Magnetic Resonance*, 172(2):296–305, 2005.

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

[11] R. Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

[12] L. Grasedyck, D. Kressner, and C. Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

[13] M. Bachmayr, R. Schneider, and A. Uschmajew. Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations. *Foundations of Computational Mathematics*, 16(6):1423–1472, 2016.

[14] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products. *Studies in Applied Mathematics*, 6(1-4):164–189, 1927.

[15] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[16] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[17] L. R. Tucker. Implications of factor analysis of three-way matrices for measurement of change. *Problems in Measuring Change*, 15:122–137, 1963.

[18] F. Verstraete, J. J. Garcia-Ripoll, and J. I. Cirac. Matrix product density operators: Simulation of finite-temperature and dissipative systems. *Physical Review Letters*, 93(20):207204, 2004.

[19] M. Zwolak and G. Vidal. Mixed-state dynamics in one-dimensional quantum lattice systems: a time-dependent superoperator renormalization algorithm. *Physical Review Letters*, 93(20):207205, 2004.

[20] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[21] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(2):025012, 2010.

[22] F. Verstraete, D. Porras, and J. I. Cirac. Density matrix renormalization group and periodic boundary conditions: A quantum information perspective. *Physical Review Letters*, 93(22):227205, 2004.

[23] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Computational complexity of projected entangled pair states. *Physical Review Letters*, 98(14):140506, 2007.

[24] Y. Saad and M. H. Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7(3):856–869, 1986.

[25] I. C. Ipsen and C. D. Meyer. The idea behind krylov methods. *American Mathematical Monthly*, pages 889–899, 1998.

[26] D. Kressner and C. Tobler. Low-rank tensor krylov subspace methods for parametrized linear systems. *SIAM Journal on Matrix Analysis and Applications*, 32(4):1288–1316, 2011.

[27] D. Calvetti, L. Reichel, and D. C. Sorensen. An implicitly restarted lanczos method for large symmetric eigenvalue problems. *Electronic Transactions on Numerical Analysis*, 2(1):21, 1994.

[28] W. E. Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9(1):17–29, 1951.

[29] C. Lanczos. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45(4):255–282, 1950.

[30] A. Krylov. On the numerical solution of the equation by which in technical questions frequencies of small oscillations of material systems are determined. *Izvestija AN SSSR (News of Academy of Sciences of the USSR)*, 7(4):491–539, 1931.

[31] J. J. García-Ripoll. Time evolution of matrix product states. *New Journal of Physics*, 8(12):305, 2006.

[32] C. Hubig. *Symmetry-protected tensor networks.* PhD thesis, Ludwig-Maximilians-Universität München, 2017.

[33] Y. Saad. *Numerical methods for large eigenvalue problems.* Manchester University Press, 1992.

[34] L. Elbouyahyaoui, A. Messaoudi, and H. Sadok. Algebraic properties of the block gmres and block arnoldi methods. *Electronic Transactions on Numerical Analysis*, 33(4):207–220, 2009.

[35] R. G. Grimes, J. G. Lewis, and H. D. Simon. A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 15(1):228–272, 1994.

[36] P. L. Montgomery. A block lanczos algorithm for finding dependencies over gf (2). In *International Conference on the Theory and Applications of Cryptographic Techniques*, pages 106–120. Springer, 1995.

[37] G. H. Golub, F. T. Luk, and M. L. Overton. A block lanczos method for computing the singular values and corresponding singular vectors of a matrix. *ACM Transactions on Mathematical Software*, 7(2):149–169, 1981.

[38] J. Cullum and W. E. Donath. A block lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices. In *IEEE Conference on Decision and Control*, volume 13, pages 505–509. IEEE, 1974.

[39] M. Bellalij, L. Reichel, G. Rodriguez, and H. Sadok. Bounding matrix functionals via partial global block lanczos decomposition. *Applied Numerical Mathematics*, 94:127–139, 2015.

[40] P. J. Davis and P. Rabinowitz. *Methods of numerical integration.* Courier Corporation, 2007.

[41] G. H. Golub and G. Meurant. *Matrices, moments and quadrature with applications.* Princeton University Press, 2009.

[42] K. P. Murphy. *Machine learning: a probabilistic perspective.* MIT Press Cambridge, 2012.

[43] D. Barber. *Bayesian reasoning and machine learning.* Cambridge University Press, 2012.

[44] C. M. Bishop. *Pattern recognition and machine learning.* Springer, 2006.

[45] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning.* MIT Press Cambridge, 2016.

[46] V. N. Vapnik. *The nature of statistical learning theory.* Springer Science & Business Media, 2013.

[47] V. N. Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[48] R. E. Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences*, 38(8):716–719, 1952.

[49] R. E. Bellman. *Dynamic Programming.* Dover Publications, 2003.

[50] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning*, pages 5–32. Springer, 1992.

[51] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press Cambridge, 1998.

[52] M. Sundermeyer, R. Schlüter, and H. Ney. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[53] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko. Translating videos to natural language using deep recurrent neural networks. *arXiv:1412.4729 [cs.CV]*, 2014.

[54] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6645–6649. IEEE, 2013.

[55] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv:1609.08144 [cs.CL]*, 2016.

[56] H. T. Siegelmann and E. D. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50(1):132–150, 1995.

[57] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[58] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. In *Readings in Computer Vision*, pages 522–533. Elsevier, 1987.

[59] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[60] T. Huckle and K. Waldherr. Subspace iteration methods in terms of matrix product states. *PAMM*, 12(1):641–642, 2012.

[61] P. Schmitteckert. Nonequilibrium electron transport using the density matrix renormalization group method. *Physical Review B*, 70(12):121302, 2004.

[62] M. L. Wall and L. D. Carr. Emergent timescales in entangled quantum dynamics of ultracold molecules in optical lattices. *New Journal of Physics*, 11(5):055027, 2009.

[63] T. Keilmann and J. J. García-Ripoll. Dynamical creation of bosonic cooper-like pairs. *Physical Review Letters*, 100(11):110406, 2008.

[64] I. Han, D. Malioutov, H. Avron, and J. Shin. Approximating spectral sums of large-scale matrices using stochastic chebyshev approximations. *SIAM Journal on Scientific Computing*, 39(4):A1558–A1585, 2017.

[65] L. Reichel, M. M. Spalević, and T. Tang. Generalized averaged gauss quadrature rules for the approximation of matrix functionals. *BIT Numerical Mathematics*, 56(3):1045–1067, 2016.

[66] T. K. Huckle, K. Waldherr, and T. Schulte-Herbrüggen. Exploiting matrix symmetries and physical symmetries in matrix product states and tensor trains. *Linear and Multilinear Algebra*, 61(1):91–122, 2013.

[67] H. Fassbender and D. Kressner. Structured eigenvalue problems. *GAMM-Mitteilungen*, 29(2):297–318, 2006.

[68] H. J. Lipkin, N. Meshkov, and A. Glick. Validity of many-body approximation methods for a solvable model:(i). exact solutions and perturbation theory. *Nuclear Physics*, 62(2):188–198, 1965.

[69] N. Meshkov, A. Glick, and H. Lipkin. Validity of many-body approximation methods for a solvable model:(ii). linearization procedures. *Nuclear Physics*, 62(2):199–210, 1965.

[70] A. Glick, H. Lipkin, and N. Meshkov. Validity of many-body approximation methods for a solvable model:(iii). diagram summations. *Nuclear Physics*, 62(2):211–224, 1965.

[71] H. Quan and F. Cucchietti. Quantum fidelity and thermal phase transitions. *Physical Review E*, 79(3):031101, 2009.

[72] J. Wilms, J. Vidal, F. Verstraete, and S. Dusuel. Finite-temperature mutual information in a simple phase transition. *Journal of Statistical Mechanics: Theory and Experiment*, 2012(01):P01023, 2012.

[73] R. Botet, R. Jullien, and P. Pfeuty. Size scaling for infinitely coordinated systems. *Physical Review Letters*, 49(7):478, 1982.

[74] R. D. King, J. Rowland, S. G. Oliver, M. Young, W. Aubrey, E. Byrne, M. Liakata, M. Markham, P. Pir, L. N. Soldatova, et al. The automation of science. *Science*, 324(5923):85–89, 2009.

[75] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009.

[76] A. van den Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. van den Driessche, E. Lockhart, L. C. Cobo, F. Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. *arXiv:1711.10433 [cs.LG]*, 2017.

[77] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[78] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[79] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv:1712.01815 [cs.LG]*, 2017.

[80] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[81] G. Aad, T. Abajyan, B. Abbott, J. Abdallah, S. A. Khalek, A. Abdelalim, O. Abdinov, R. Aben, B. Abi, M. Abolins, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

[82] G. Aad, B. Abbott, J. Abdallah, O. Abdinov, R. Aben, M. Abolins, O. AbouZeid, H. Abramowicz, H. Abreu, R. Abreu, et al. Combined measurement of the higgs boson mass in p p collisions at s= 7 and 8 tev with the atlas and cms experiments. *Physical Review Letters*, 114(19):191803, 2015.

[83] G. Aad, B. Abbott, J. Abdallah, O. Abdinov, B. Abeloos, R. Aben, O. AbouZeid, N. Abraham, H. Abramowicz, et al. Measurements of the higgs boson production and decay rates and constraints on its couplings from a combined atlas and cms analysis of the lhc pp collision data at s= 7 s= 7 and 8 tev. *Journal of High Energy Physics*, 2016:1–113, 2016.

[84] T. Caneva, T. Calarco, and S. Montangero. Chopped random-basis quantum optimization. *Physical Review A*, 84(2):022326, 2011.

[85] G. Quiroz and D. A. Lidar. Optimized dynamical decoupling via genetic algorithms. *Physical Review A*, 88(5):052306, 2013.

[86] M. Bukov, A. G. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta. Machine learning meets quantum state preparation. the phase diagram of quantum control. *arXiv:1705.00565 [quant-ph]*, 2017.

[87] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko, M. Tiersch, A. Zeilinger, and H. J. Briegel. Active learning machine learns to create new quantum experiments. *Proceedings of the National Academy of Sciences*, page 201714936, 2018.

[88] A. Hentschel and B. C. Sanders. Machine learning for precise quantum measurement. *Physical Review Letters*, 104(6):063603, 2010.

[89] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders. Learning in quantum control: high-dimensional global optimization for noisy quantum dynamics. *Neurocomputing*, 268:116–126, 2017.

[90] P. B. Wigley, P. J. Everitt, A. van den Hengel, J. Bastian, M. A. Sooriyabandara, G. D. McDonald, K. S. Hardman, C. Quinlivan, P. Manju, C. C. Kuhn, et al. Fast machine-learning online optimization of ultra-cold-atom experiments. *Scientific Reports*, 6:25890, 2016.

[91] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: encoder-decoder approaches. *arXiv:1409.1259 [cs.CL]*, 2014.

[92] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv:1412.3555 [cs.NE]*, 2014.

[93] L. Viola, E. Knill, and S. Lloyd. Dynamical decoupling of open quantum systems. *Physical Review Letters*, 82(12):2417, 1999.

[94] A. M. Souza, G. A. Álvarez, and D. Suter. Robust dynamical decoupling for quantum computing and quantum memory. *Physical Review Letters*, 106(24):240501, 2011.

[95] M. Pelikan, D. E. Goldberg, and F. G. Lobo. A survey of optimization by building and using probabilistic models. *Computational Optimization and Applications*, 21(1):5–20, 2002.

[96] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv:1707.06347 [cs.LG]*, 2017.

[97] T. Fösel, P. Tighineanu, T. Weiss, and F. Marquardt. Reinforcement learning with neural networks for quantum feedback. *arXiv:1802.05267 [quant-ph]*, 2018.

[98] M. Ostaszewski, J. Miszczak, and P. Sadowski. Geometrical versus time-series representation of data in learning quantum control. *arXiv:1803.05169 [quant-ph]*, 2018.

[99] G. Carleo and M. Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355(6325):602–606, 2017.

[100] W. L. McMillan. Ground state of liquid he 4. *Physical Review*, 138(2A):A442, 1965.

[101] S. Sorella, M. Casula, and D. Rocca. Weak binding between two aromatic rings: Feeling the van der waals attraction by quantum monte carlo methods. *The Journal of Chemical Physics*, 127(1):014105, 2007.

[102] X. Gao and L.-M. Duan. Efficient representation of quantum many-body states with deep neural networks. *Nature Communications*, 8(1):662, 2017.

[103] Y. Huang and J. E. Moore. Neural network representation of tensor network and chiral states. *arXiv:1701.06246 [cond-mat.dls-nn]*, 2017.

[104] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac. Simulation of quantum many-body systems with strings of operators and monte carlo tensor contractions. *Physical Review Letters*, 100(4):040501, 2008.

[105] A. Sfondrini, J. Cerrillo, N. Schuch, and J. I. Cirac. Simulating two-and three-dimensional frustrated quantum systems with string-bond states. *Physical Review B*, 81(21):214426, 2010.

[106] S.-S. Gong, W. Zhu, and D. Sheng. Emergent chiral spin liquid: Fractional quantum hall effect in a kagome heisenberg model. *Scientific Reports*, 4:6317, 2014.

[107] T. Wahl, H.-H. Tu, N. Schuch, and J. Cirac. Projected entangled-pair states can describe chiral topological states. *Physical Review Letters*, 111(23):236805, 2013.

[108] D. Poilblanc. Investigation of the chiral antiferromagnetic heisenberg model using projected entangled pair states. *Physical Review B*, 96(12):121118, 2017.

[109] H. Saito and M. Kato. Machine learning technique to find quantum many-body ground states of bosons on a lattice. *Journal of the Physical Society of Japan*, 87(1):014001, 2017.

[110] S. R. Clark. Unifying neural-network quantum states and correlator product states via tensor networks. *Journal of Physics A: Mathematical and Theoretical*, 51(13):135301, 2018.

[111] G. Carleo, Y. Nomura, and M. Imada. Constructing exact representations of quantum many-body systems with deep neural networks. *arXiv:1802.09558 [cond-mat.dls-nn]*, 2018.

Acknowledgements

I would like to sincerely thank my family, friends and colleagues, who have in many ways contributed to making my PhD the great experience that it was. In particular, I would like to thank

- Thomas Huckle for accepting me as his student and supporting me during the entirety of my work. The level of support I had the luck of receiving from him is one that most PhD students can only dream of.
- Hans Bungartz for allowing me to join his chair and establishing a working culture based on own initiative and responsibility, common sense, encouragement, liberty and flexibility.
- Mari Carmen Bañuls for being open to a collaboration from which I have learned so much. Without her support, this dissertation would not have been possible in its present form.
- Miguel Hernández-Lobato for inviting me to a fun and fruitful visit in Cambridge and reviewing this dissertation.
- Steffen Glaser for showing interest in my work, providing an opportunity to work with real experimental data and reviewing this dissertation.
- Ignacio Cirac for letting me collaborate so much with his group and providing valuable insight in discussions.
- Xiaotong Ni for providing the first idea on where to apply machine learning in quantum physics and a great collaboration.
- Ivan Glasser, Nicola Pancotti, Ivan Rodriguez and Ignacio Cirac for another inspiring and fun collaboration.
- Anna Hashagen, David Leiner, Jakob Wierzbowski, Nicola Pancotti and the rest of the ExQM crew for a fun and inspiring PhD program with many thought-provoking discussions and recreational extracurricular activities. I am also grateful to Anna Hashagen for convincing me to try to make my PhD life easier.
- Thomas Schulte-Herbrüggen and Steffen Glaser for running ExQM and organizing a fantastic workshop. I am very happy to have been a part of ExQM.
- Tobi Neckel for fun and mostly relaxed cooperative work on the Ferienakademie. I can only hope to have such nice and understanding colleagues in the future.
- Emily Mo-Hellenbrand and Florian Künzner for being pleasant office mates I always enjoyed chatting with (sometimes maybe even a bit too much from a productivity-oriented perspective).
- Alfredo Parra, Nikola Tchipev and all my other colleagues for making the chair the fun, relaxed and simultaneously ambitious institution it is.

- Marie Eidtmann for supporting me in and beautifying every aspect of my life as well as finding all the typoos in this thesis.
- My family for their love and support and for having contributed in many ways to the success of my PhD.
- Rudolf Erttmann and his team without whom I would likely have never been able to pursue a PhD at all.
- all the nice folks I have forgotten. Sorry! You rule.

Appendix

A On the approximation of functionals of very large Hermitian matrices represented as matrix product operators

Authors Moritz August, Dr. Dr. Mari Carmen Bañuls and Prof. Dr. Thomas Huckle

Citation Electronic Transactions on Numerical Analysis, 46:215-232, 2017

Summary In this work, we introduced a method to approximate functions $\mathsf{Tr}f(A)$ of Hermitian matrices A represented by matrix product operators. The method builds on an expression of the global block Lanczos algorithm in the framework of tensor networks. As a first step, we discussed the connection between the global Lanczos algorithm and Gauss quadrature. Following this, we demonstrated how the global Lanczos algorithm can be expressed in the tensor network framework to allow for the approximation of $\mathsf{Tr}f(A)$ for very high-dimensional inputs and stated the main properties of our method. A major consequence of this reformulation in the tensor network format is that it allows us to use blocks of size equal to A. This makes it possible to employ unitary starting matrices U which in contrast to non-unitary matrices do not adversly affect the approximation of the function. Based on this finding, we then proved that our method converges to the exact result in absence of truncation errors in the matrix product operators and given a sufficiently large Krylov dimension. Furthermore, we provided an analysis of the complexity of our method and found it to be of polynomial complexity in the number of particles of a physical system. Finally, we provided numerical evidence that our method yields good approximations for the challenging example of functions of thermal quantum states in equilibrium of the transverse Ising Hamiltonian. We showed how the von Neumann entropy of such a Gibbs state ρ can be approximated by our method for different temperatures and discussed the different results. Additionally, we compared the performance of our implementation to the runtime of an established numerical framework for the approximation of $\mathsf{Tr}exp(A)$.

Contribution The author's contribution to this work lies in the development of the algorithm, its implementation and finally its evaluation for the numerical problems stated above. More precisely, the author experimented with a larger number of possible approximation methods based on Krylov algorithms before finding the presented method. In the course of this development phase, Prof. Huckle kindly supported the author by discussing ideas and pointing to possibly relevant works. The implementation of all developed ideas including the presented method was conducted by the author based on a `C++` tensor network library providing the core routines. This library was kindly made available by Dr. Dr. Bañuls who also supported the author by discussing implementational and physical questions. The author evaluated the implemented method using computational resources of the chair of scientific computing and the Leibniz supercomputing center. All figures and tables shown in the work were prepared by the author as well as the text itself. In writing the article, Prof. Huckle and Dr. Dr. Bañuls again supported the author.

Electronic Transactions on Numerical Analysis.
Volume 46, pp. 215–232, 2017.

ISSN 1068–9613.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

ON THE APPROXIMATION OF FUNCTIONALS OF VERY LARGE HERMITIAN MATRICES REPRESENTED AS MATRIX PRODUCT OPERATORS*

MORITZ AUGUST†, MARI CARMEN BAÑULS‡, AND THOMAS HUCKLE†

Abstract. We present a method to approximate functionals $\mathsf{Tr} f(A)$ of very high-dimensional Hermitian matrices A represented as Matrix Product Operators (MPOs). Our method is based on a reformulation of a block Lanczos algorithm in tensor network format. We state main properties of the method and show how to adapt the basic Lanczos algorithm to the tensor network formalism to allow for high-dimensional computations. Additionally, we give an analysis of the complexity of our method and provide numerical evidence that it yields good approximations of the entropy of density matrices represented by MPOs while being robust against truncations.

Key words. tensor decompositions, numerical analysis, Lanczos method, Gauss quadrature, quantum physics

AMS subject classifications. 65F60, 65D15, 65D30, 65F15, 46N50, 15A69

1. Introduction.

Approximating functionals of very large matrices is an important problem in many fields of science, such as network analysis [3, 9, 11, 26] or quantum mechanics [33, 37]. In many cases, the respective matrices are Hermitian due to either the underlying physical properties of the systems they describe or the way they are constructed from, e.g., a graph. Naturally, as the dimensionality of the matrices becomes very high, i.e., several tens or hundreds of thousands and above, explicit methods of function evaluation, like exact diagonalization, break down and approximations must be made.

One paradigm for the approximation of high-dimensional matrices that has gained a lot of attention especially in the quantum information, condensed matter, and numerical linear algebra communities are tensor network representations [2, 15, 18, 27, 33, 37]. Among the class of tensor networks, matrix product states (MPS) and matrix product operators (MPO) count among the best established methods. These representations approximate large tensors by contractions of multiple low-rank tensors in a row and have been shown to yield efficient parametrizations of many relevant states of quantum many-body systems [17, 28, 35].

In this work, we introduce a novel method to approximate functionals of the form $\mathsf{Tr} f(A)$ where we assume $f : \mathbb{C}^N \times \mathbb{C}^N \to \mathbb{C}^N \times \mathbb{C}^N$ to be smooth and defined on the spectrum of A as well as A to be Hermitian and given in MPO format. For our method, we have reformulated a block version of the Lanczos algorithm in MPO/MPS-representation. This particular block Lanczos algorithm will be referred to as global Lanczos algorithm in the following and has already been used to approximate functionals of the given form for explicitly stored matrices [4, 8]. Rewriting it for the tensor network formalism, however, allows us to consider block vectors of size identical to A, which was previously prohibitive. Our method is thus able to approximate $\mathsf{Tr} f(A)$ for certain $f(A)$ requiring only one carefully selected starting block vector. This means that we get rid of the approximation error induced by the need to combine the results obtained for multiple different starting block vectors. At the same time, we find the numerical error induced by the MPS/MPO representation to be comparably small. Our method can be applied whenever A is efficiently approximated by an MPO. We will in the following, however, focus on the case where A has directly been defined as an MPO.

The rest of this work is structured as follows: after a basic introduction to matrix product states and operators in Section 2, we will introduce the block Lanczos method we employ in

*Received December 1, 2016. Accepted May 5, 2017. Published online on July 7, 2017. Recommended by G. Rodriguez.
†Department of Informatics, Technical University of Munich, 85748 Garching, Germany ({august, huckle}@in.tum.de).
‡Max Planck Institute for Quantum Optics, 85748 Garching, Germany (mari.banuls@mpq.mpg.de).

ETNA
Kent State University and
Johann Radon Institute (RICAM)

this work and explain its connection to Gauss quadrature in Section 3. Following this, we will then state our method in Section 4, show how we have reformulated the global Lanczos method in the tensor network formalism, and discuss its properties as well as give an analysis of its complexity. Finally, in Section 5 we will provide numerical evidence for the fast and robust convergence of our method for the case of the trace-norm and von-Neumann entropy of quantum mechanical density matrices. We conclude with a discussion of the results in Section 6.

2. Matrix product states and operators. In the area of tensor networks, MPS and MPOs form a well-established class of tensor decompositions that allow for efficient and stable approximation of very high-dimensional vectors and matrices, respectively. While they are commonly used in theoretical and numerical quantum many-body physics to model, e.g., ground and thermal states [10, 29, 33, 35, 36, 37, 38, 39, 41], they also have been independently introduced in the numerical mathematics community under the name of Tensor Trains (TT) [27] as a general approximation tool. Since our work is mainly motivated by applications in quantum physics, we will adapt the respective terminology in the following.

A matrix product state is a decomposition of a vector $v \in \mathbb{C}^N$ such that

$$v_i = v_{i_1 \dots i_L} = \mathsf{Tr} C_1^{i_1} C_2^{i_2} \cdots C_L^{i_L},$$

where the index i is split up into L sub-indices of dimensionality d, called physical indices, $i_1, \dots, i_L$. We will refer to $C_1, \dots, C_L \in \mathbb{C}^{d \times D \times D}$ as the sites or core tensors of the MPS and D is called the bond dimension. The superscripts i_j of the C_j correspond to the physical indices. The concept of splitting up the index i is the standard way to represent vectors and matrices in TT/MPS form and is also used for so-called quantized tensor trains (QTT) [21] in the numerical community. While in principle every site may have its own bond dimensions, as long as they allow for contraction with neighbouring sites, for simplicity and without loss of generality, we will assume all sites to have identical bond dimension D. The physical dimension d is likewise assumed to be identical for all sites. It is important to note that $N = L^d$, as this relation forms the basis for the ability of MPS/MPOs to represent vectors and matrices of very high dimensionality.

A slightly different representation can be chosen, where

$$v_i = v_{i_1 \dots i_L} = C_1^{i_1} C_2^{i_2} \cdots C_L^{i_L},$$

with $C_1 \in \mathbb{C}^{d \times 1 \times D}$ and $C_L \in \mathbb{C}^{d \times D \times 1}$. In physical terms, the former representation corresponds to systems with closed boundary conditions (CBC) whereas the latter assumes open boundary conditions (OBC). It is clear that OBC is a special case of CBC. For the remainder of this work, we assume open boundary conditions for the sake of simplicity, but the algorithm can be applied to the CBC case, albeit with a modified computational cost. Following this decomposition, a particular element of v is described by a chain of matrix multiplications, explaining the name of the representation.

Now, the whole vector v can be written as

$$\begin{aligned} v &= \sum_{i_1, \dots, i_L}^{d} \left(C_1^{i_1} C_2^{i_2} \cdots C_L^{i_L}\right) \left(e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L}\right) \\ &= \sum_{k_2, \dots, k_{L-1}}^{D} \left(\sum_{i_1}^{d} C_{1,k_2}^{i_1} e_{i_1}\right) \otimes \left(\sum_{i_2}^{d} C_{2,k_2 k_3}^{i_2} e_{i_2}\right) \\ &\qquad \otimes \cdots \otimes \left(\sum_{i_L}^{d} C_{L,k_{L-1}}^{i_L} e_{i_L}\right) \end{aligned}$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

$$= \sum_{k_2,\ldots,k_{L-1}}^{D} u_{1,k_2} \otimes u_{2,k_2k_3} \otimes \cdots \otimes u_{L,k_{L-1}} ,$$

where e_j denotes the jth column of the identity matrix and the subscripts k_j and k_{j+1} denote the row and column indices of the matrices $C_j^{i_j}$, respectively. This expression sheds some light on the underlying tensor product structure of MPS and facilitates comparisons with other tensor decomposition schemes.

We now turn our attention to the representation of operators and matrices. Abstractly, one can define an MPO as an operator with an MPS representation in a direct product basis of the operator linear space. More concretely, for the representation of a matrix $A \in \mathbb{C}^{N\times N}$ as an MPO, the approach presented above can easily be adapted to yield

$$A_{ij} = A_{i_1\ldots i_L j_1\ldots j_L} = C_1^{i_1 j_1} C_2^{i_2 j_2} \cdots C_L^{i_L j_L},$$

where i, j have been split up as before, resulting in two superscripts this time, and with the matrices $C_1, \ldots, C_L \in \mathbb{C}^{d\times d\times D\times D}$. In analogy to the case for a vector, we can write the whole matrix as

$$\begin{aligned} A = &\sum_{i_1,\ldots,i_L}^{d} \sum_{j_1,\ldots,j_L}^{d} (C_1^{i_1 j_1} C_2^{i_2 j_2} \cdots C_L^{i_L j_L}) \\ &\qquad \cdot (e_{i_1} \otimes e_{i_2} \otimes \cdots \otimes e_{i_L})(e_{j_1}^T \otimes e_{j_2}^T \otimes \cdots \otimes e_{j_L}^T) \\ = &\sum_{i_1,\ldots,i_L}^{d} \sum_{j_1,\ldots,j_L}^{d} \sum_{k_2,\ldots,k_{L-1}}^{D} (C_{1,1k_2}^{i_1 j_1} C_{2,k_2k_3}^{i_2 j_2} \cdots C_{L,k_{L-1}1}^{i_L j_L}) \\ &\qquad \cdot (e_{i_1} e_{j_1}^T) \otimes (e_{i_2} e_{j_2}^T) \otimes \cdots \otimes (e_{i_L} e_{j_L}^T) \\ = &\sum_{k_2,\ldots,k_{L-1}}^{D} U_{1,k_2} \otimes U_{2,k_2k_3} \otimes \cdots \otimes U_{L,k_{L-1}}, \end{aligned}$$

where e_j is again the jth column of the identity and e_j^T is its transpose. Note that this also holds true for other product bases, like for instance the Pauli basis. Making use of these formulations, it is easy to show that basic operations such as scalar multiplication, addition, and inner product as well as the multiplication of an MPS by an MPO or of two MPOs can be performed in the formalism. The addition and non-scalar multiplication, however, lead to an increase in the bond dimension D. For the addition of two MPS/MPOs with bond dimensions D and D', the new bond dimension is $D'' \leq D + D'$, and for the multiplication, $D'' \leq D \cdot D'$ [33]. This can again easily be verified.

It is obvious from the above explanation that the bond dimension is the decisive factor for the expressive power of the formalism. An exact representation of a vector (operator) as an MPS (MPO) is always possible if we allow the bond dimension D to be big enough, which may mean exponentially large in L, up to $d^{\lfloor N/2\rfloor}$ [38]. When the maximum value of D is limited to some fixed value (*truncated*) smaller than the one required for exactness, not all vectors or operators can be represented, which may give rise to approximation errors. We will in the following denote that some vector v or matrix A is approximated with bond dimension D by writing $v[D]$ and $A[D]$, respectively. Nevertheless, it has been found that for many physically relevant states and operators, MPS/MPO yield good approximations for $D \in \mathcal{O}(poly(L))$ [35, 36, 41] leading to the total number of parameters $LdD^2 \in \mathcal{O}(poly(L))$ as opposed to d^L or d^{2L} for the whole vector or matrix, respectively. This constitutes another main reason for their usefulness as an efficient approximation scheme.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 1: Global Lanczos Algorithm.

Input : Matrix $A \in \mathbb{C}^{N\times N}$, Starting Matrix $U \in \mathbb{C}^{N\times M}$, Number of Dimensions K

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
for $i \leftarrow 1; i \leq K$ **do**
    $\beta_i \leftarrow \|V_{i-1}\|_F$;
    if $\beta_i = 0$ **then**
        break ;
    end
    $U_i \leftarrow V_{i-1}/\beta_i$;
    $V_i \leftarrow AU_i - \beta_i U_{i-1}$;
    $\alpha_i \leftarrow \langle U_i, V_i\rangle$;
    $V_i \leftarrow V_i - \alpha_i U_i$;
end

Output : Orthonormal Basis $\mathbf{U}_K \in \mathbb{C}^{N\times KM}$, Tridiagonal Matrix $T_K \in \mathbb{R}^{KM\times KM}$

Naturally, many methods have been developed to find optimal and canonical representations for a given D both in the numerical and the quantum physics community. The most important algorithms for optimizing a given MPS/MPO with respect to some error function and bond dimension thereby rely on local updates of the individual C_i with all other sites being treated as constants, rather than considering all parameters simultaneously. These algorithms, starting with the left- or right-most site, generally sweep back and forth over the chain of sites updating one site per step until convergence. As all sites not considered in a given step are treated as fixed, this sweeping scheme allows for reusage of previously computed values in a dynamical programming fashion. As explaining the details and the complexity of these algorithms exceeds the scope of this section, we refer the interested reader to the overview articles [2, 15, 33, 37].

3. The global Lanczos algorithm and Gauss quadrature. The idea of employing variants of Krylov methods to solve various types of problems, for instance, solving linear systems [19, 22, 31], finding eigenvectors [1, 5, 23, 24] or approximating the action of an exponential operator onto a vector [12], is already well-established. To solve, e.g., linear systems with multiple right-hand sides and for reasons of efficiency, block versions of the originally vector-based Krylov algorithms have been developed. While there exist several block versions of the Lanczos algorithm [6, 8, 13, 16, 25], we will only consider the one presented in [4] as it does not require the columns of the basis blocks to be orthogonal, which would be prohibitive for very large matrices.

Starting from an initial matrix $U \in \mathbb{C}^{N\times M}$, the algorithm will build up a basis of matrices $\mathbf{U}_i = [U_1, \cdots, U_i]$ with $\mathbf{U}_i \in \mathbb{C}^{N\times iM}$. Now, we first need to state the inner product with respect to which the individual U_i must be orthonormal and define it to be

$$\langle U_i, U_j\rangle = \mathsf{Tr} U_i^* U_j,$$

where $U_i, U_j \in \mathbb{C}^{N\times M}$. This induces the well known Frobenius norm

$$\|U_i\|_F = \langle U_i, U_i\rangle^{1/2},$$

and hence this definition of the inner product is a straightforward generalization of the one used in the standard Lanczos algorithm. Naturally, one may also choose different inner products [8].

ETNA
Kent State University and
Johann Radon Institute (RICAM)

For this work, we do, however, choose the Frobenius norm as it can be efficiently computed for MPOs. Equipped with this definition, we can see that Algorithm 1 is in fact a direct generalization of the standard Lanczos algorithm to the matrix-case. As such we find that after i steps, the method has produced the reduction T_i of A given by

$$T_i = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_i \\ \mathbf{0} & & \beta_i & \alpha_i \end{bmatrix}$$

and yields the partial global Lanczos decomposition

$$A\mathbf{U}_i = \mathbf{U}_i \tilde{T}_i + \beta_{i+1} U_{i+1} E_i^T,$$

where we define $\tilde{T}_i = T_i \otimes I_M \in \mathbb{R}^{iM \times iM}$ and $E_i^T = [\mathbf{0}, \cdots, \mathbf{0}, I_M] \in \mathbb{R}^{M \times iM}$. Furthermore, it holds that

$$\beta_{i+1} U_{i+1} = (A - \alpha_i I_N) U_i - \beta_i U_{i-1}$$

and

$$\mathbf{U}_i^* A \mathbf{U}_i = T_i,$$

if we apply the inner product defined previously. From now on, we will implicitly make use of this inner product whenever appropriate. Then, all other results obtained for the original Lanczos method carry over to the global Lanczos case.

To establish the link between the global Lanczos method and Gauss quadrature, we start by observing that

$$u^* f(A) u = u^* V_A f(\Lambda_A) V_A^* u = \sum_{i=1}^{N} f(\lambda_i) \mu_i^2 = \int_a^b f(\lambda) \mathrm{d}\mu(\lambda),$$

with $V_A \Lambda_A V_A^*$ being the spectral decomposition of A, $\mu_i = e_i^T V_A^* u$ and

$$\mu(\lambda) = \begin{cases} 0 & \text{if } \lambda < \lambda_1 = a \\ \sum_{i=1}^{j} \mu_i^2 & \text{if } \lambda_j \leq \lambda < \lambda_{j+1} \\ \sum_{i=1}^{N} \mu_i^2 & \text{if } b = \lambda_N \leq \lambda \end{cases}$$

being a piecewise constant and nondecreasing distribution function. Here we assume the eigenvalues of A to be ordered ascendingly. We can use this result to obtain

$$\begin{aligned} \mathcal{I} f &\coloneqq \mathsf{Tr}(U^* f(A) U) = \sum_{i=1}^{N} e_i^* U^* V_A f(\Lambda_A) V_A^* U e_i = \sum_{i=1}^{N} \int_a^b f(\lambda) \mathrm{d}\mu_i(\lambda) \\ &= \int_a^b f(\lambda) \mathrm{d} \sum_{i=1}^{N} \mu_i(\lambda) = \int_a^b f(\lambda) \mathrm{d}\mu(\lambda) \end{aligned}$$

for a matrix U like the initial matrix of the global Lanczos method, where we define $\mu_i(\lambda)$ analogously to the case above and $\mu(\lambda) \coloneqq \sum_{i=1}^{N} \mu_i(\lambda)$. This Riemann-Stieltjes integral can now be tackled via Gauss-type quadrature, the most general formulation of which is given by

$$\mathcal{G} f \coloneqq \sum_{k=1}^{K} \omega_k f(\theta_k) + \sum_{m=1}^{M} \nu_m f(\tau_m),$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where θ_k and τ_m are called the nodes and ω_k and ν_m the weights of the quadrature. In this work we only consider the case where $M = 0$.

It is well known that in order to determine the ω_k and θ_k that satisfy this property, one can construct a sequence of polynomials $\{p_0, \cdots, p_K\}$ that are orthonormal in the sense that

$$\int_a^b p_i(\lambda)p_j(\lambda)\mathrm{d}\mu(\lambda) = \delta_{ij}$$

and that satisfy a recurrence relation given by

$$\beta_i p_i(\lambda) = (\lambda - \alpha_{i-1})p_{i-1}(\lambda) - \beta_{i-1}p_{i-2}(\lambda), \tag{3.1}$$

where $p_{-1}(\lambda) \equiv 0$ and $p_0(\lambda) \equiv 1$. Then, the roots of p_K can be shown to be the optimal θ_k [7, 14]. Now, the recurrence relation yields a recurrence matrix T_K defined by

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix},$$

whose eigenvalues are the zeros of $p_K(\lambda)$ and consequentially the θ_k of $\mathcal{G}f$ [14]. The ω_k are given by the squared first elements of the normalized eigenvectors of T_K and so,

$$\mathcal{G}f = e_1^T f(T_K)e_1 = e_1^T V_T f(\Lambda_T)V_T^* e_1,$$

where $V_T\Lambda_T V_T^*$ is the spectral decomposition of T_K.

Now, the U_i from the global Lanczos method can be expressed by

$$U_i = p_{i-1}(A)U,$$

with p_{i-1} being some polynomial of degree $i-1$. Then, it is clear that

$$\langle p_{i-1}(A)U, p_{j-1}(A)U\rangle = \langle U_i, U_j\rangle = \delta_{ij}$$

and taking into account the above derivations

$$\begin{aligned}\langle p_{i-1}(A)U, p_{j-1}(A)U\rangle &= \mathsf{Tr}(U^* p_{i-1}(A)^* p_{j-1}(A)U) \\ &= \int_a^b p_{i-1}(\lambda)p_{j-1}(\lambda)\mathrm{d}\mu(\lambda).\end{aligned}$$

Hence, the global Lanczos method produces orthonormal polynomials [24] that in addition satisfy the recurrence relation stated in equation (3.1) by construction as we have seen above. The T_i obtained by the global Lanczos algorithm is thus the recurrence matrix needed to perform a Gauss quadrature with i nodes. If we choose $U \in \mathbb{C}^{N\times N}$ to be unitary, it follows that

$$\mathsf{Tr}f(A) = \mathsf{Tr}(U^* f(A)U) = \int_a^b f(\lambda)\mathrm{d}\mu(\lambda) \approx e_1^T f(T_i)e_1.$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

Algorithm 2: Approximation Algorithm.

Input : MPO $A[D_A] \in \mathbb{C}^{N\times N}$, Starting orthogonal MPO $U[D_{init}] \in \mathbb{C}^{N\times N}$, Number of Dimensions K, Maximal Bond-Dimension D_{max}, Stopping Criteria $\mathcal{S}$

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
$D \leftarrow D_{init}$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \sqrt{\texttt{contract}(V_{i-1}, V_{i-1})}$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $U_i \leftarrow \texttt{multiplyScalar}(1/\beta_i, V_{i-1})$;
  $D \leftarrow \min(D_{max}, D \cdot D_A)$;
  $V_i \leftarrow \texttt{multiplyAndOptimize}(A, U_i, D)$;
  $D \leftarrow \min(D_{max}, D + D_{U_{i-1}})$;
  $V_i \leftarrow \texttt{sumAndOptimize}(V_i, -\beta_i U_{i-1}, D)$;
  $\alpha_i \leftarrow \texttt{contract}(U_i, V_i)$;
  $D \leftarrow \min(D_{max}, D + D_{U_i})$;
  $V_i \leftarrow \texttt{sumAndOptimize}(V_i, -\alpha_i U_i, D)$;
  $V_T \Lambda_T V_T^* \leftarrow \texttt{spectralDecomposition}(T_i)$;
  $\mathcal{G}f \leftarrow \beta_1^2 e_1^T V_T f(\Lambda_T) V_T^* e_1$;
  if $\texttt{checkStop}(\mathcal{G}f, \Lambda_T, \mathcal{S})$ **then**
    break ;
  end
end

Output : Approximation $\mathcal{G}f$ of $\mathsf{Tr}f(A)$

4. Assembling the parts. Now that we have reviewed the relevant theoretical aspects, we will proceed by showing how we put together the pieces to obtain our algorithm. The whole algorithm is presented in Algorithm 2.

Since the global Lanczos method is based on matrix-matrix multiplications, additions of matrices, and multiplications of matrices by scalars, these operations have to be formulated for the MPO case. As the bond dimension of the basis-MPOs grows with the number of multiplications and additions, we need to keep track of the bond dimensions and perform projections onto lower bond dimensions whenever necessary. Thus, the input parameters of our method are the MPO $A[D_A] \in \mathcal{C}^{N\times N}$, an orthogonal starting MPO $U[D_{init}] \in \mathcal{C}^{N\times N}$, the maximal Krylov-dimension K, a set of stopping criteria $\mathcal{S}$, and finally the maximal bond dimension D_{max} of the U_i.

It should be stressed that we assume $U[D_{init}]$ to be unitary and of the same dimension as $A[D_A]$. This allows us to replace the approximation that had to be made previously by combining the estimations for several starting matrices by the exact computation since now it holds that $\mathsf{Tr}f(A) = \mathsf{Tr}U^* f(A)U$ if we assume U to be normalized without loss of generality. This is only possible due to the fact that we translate the Lanczos method to the MPO formalism.

Besides the case of very large matrices that can be explicitly stored but are too large for multiplications with equally sized matrices, this is especially important for the case where the respective matrices are only given in MPO format and N is of the order of several millions, as

ETNA
Kent State University and
Johann Radon Institute (RICAM)

in the case of quantum many-body systems. While we introduce some approximation error by using MPOs, we will show in Section 5 that these errors can be comparably small already for low bond dimensions in cases of practical interest. In the following, we will omit the declaration of the bond dimension of an MPO whenever it increases clarity.

While in principle every orthogonal MPO can serve as a starting point, in this work we choose $U[D_{init}]$ to be the identity matrix because it has a minimal MPO formulation of $C_i^{jk} = \delta_{jk}$. This allows us to start from the minimal bond dimension $D_{init} = 1$ and thus maximizes the amount of relevant information that we can store for a given D_{max}. In certain cases it might however be possible to choose a better starting MPO, e.g., when A is very close to being diagonal. Note that starting with the full identity matrix does not imply convergence in one step as the identity is not a basis of the space implied by the Frobenius inner product. For the implementation of the inner product and norm used in the global Lanczos algorithm, we observe that

$$\langle U_i, U_j \rangle = \sum_{k=1}^{N} \sum_{l=1}^{N} U_{i,kl} U_{j,lk} = U_{i,vec}^{*} U_{j,vec} \,,$$

where $U_{i,vec}$ and $U_{j,vec}$ are the vectorized versions of U_i and U_j, respectively. This allows us to make use of an efficient, exact way of computing the inner product of MPS [33] by rewriting the $C_i^{j_i k_i}$ as $C_i^{j_i'}$ with $\dim j_i' = \dim j_i \cdot \dim k_i$ and hence vectorizing the MPO. This functionality is implemented in `contract()`. The implementation of the scalar multiplication `multiplyScalar()` is straightforward as it corresponds to multiplying an arbitrary C_i—we choose C_1 for simplicity—of the respective MPO by the scalar at hand.

A bit more care has to be taken when implementing the functions for the multiplication and summation of MPOs, `multiplyAndOptimize()` and `sumAndOptimize()`, respectively. One possibility would be to first perform the respective operation exactly, i.e., use the bond dimension required for the exact result, and to project the resulting MPO onto the current D via the singular value decomposition (SVD) of its C_i in a second step. It has however been found that performing the projection simultaneously to the multiplication or summation at the level of the individual C_i yields superiour results; see [33, 37, 40]. In case of the multiplication, we implement this strategy by solving the optimization problem

$$\min_{\tilde{C}_i} \| A U_j[D_{old}] - \tilde{U}_j[D_{new}] \|_F^2 \,,$$

where D_{old} is the bond dimension used previous to the multiplication and D_{new} is the bond dimension used for the optimization. $\tilde{U}_j[D_{new}]$ denotes the result of the multiplication of A on U_j and the $\tilde{C}_i$ are its tensors. The implementation hence performs the multiplication of the MPOs at tensor level and directly optimizes the resulting tensors for the chosen bond dimension by employing the sweeping scheme sketched in Section 2. In order to apply this algorithm to the case of MPO-MPO multiplication, we rewrite AU_j as

$$(I \otimes A) U_{j,vec} = \begin{bmatrix} A & 0 & & \mathbf{0} \\ 0 & A & \ddots & \\ & \ddots & \ddots & 0 \\ \mathbf{0} & & 0 & A \end{bmatrix} \begin{bmatrix} U_{j,1} \\ U_{j,2} \\ \vdots \\ U_{j,N} \end{bmatrix} ,$$

with $U_{j,k}$ being the kth column vector of U_j. Due to technical reasons, we do in fact use $U_j \otimes I$ and A_{vec}. For the summation, we apply the same strategy and solve

$$\min_{\tilde{C}_i} \| \left(U_j[D_{old}] + \sum_k \gamma_k U_k[D_{old}^k] \right) - \tilde{U}_j[D_{new}] \|_F^2 \,,$$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

where D_{old} is the bond dimension used before the addition, D_{old}^k are some other previously used bond dimensions, D_{new} is the bond dimension to be used for the optimization, and $\gamma_k \in \mathbb{C}$ are some scalars. $\tilde{U}_j[D_{new}]$, similarly as before, represents the outcome of the summation and the $\tilde{C}_i$ are its tensors.

As it can be seen in Algorithm 2, we allow for exact multiplication and summation as long as the resulting bond dimension does not grow beyond D_{max}. This, however, happens quickly since D can grow exponentially with the number of iterations, and so most of the U_i will be represented based on D_{max}. This underlines the importance of D_{max} for the accuracy of the approximation.

After the algorithm has completed one iteration of the global Krylov method, the spectral decomposition of T_i is performed and the current approximation is computed. Based on the approximation and the eigenvalues of T_i, the algorithm then determines if it should be stopped in `checkStop`. Here we have to account for several factors.

Firstly, we know that $\mathcal{G}f$ converges to the correct value in absence of approximation errors. So, the algorithm can terminate when the distance between the previous and current $\mathcal{G}f$ becomes smaller than some ϵ.

Secondly, it is clear that the projection of the generated MPOs down to D_{max} introduces an approximation error. While it is possible to obtain the error of the optimization problems described above, it is not clear how the accumulated error influences $\mathcal{G}f$ precisely. However, a possible way of detecting when the approximation error has become too large is to check for the violation of some theoretical constraints. For instance, in case of a positive A, we know that the Ritz-values of A must be positive as well. If T_i starts to develop significant negative eigenvalues relative to the allowed numerical precision, we thus know that the total approximation error has reached a level that leads to unreasonable results. The same reasoning could be applied for other intervals in which one knows the spectrum of A to be in.

It is well known that, depending on the sign of the derivative of f in (λ_1, λ_N), $\mathcal{G}f$ can yield an upper/lower bound and that it converges to the true value. Based on this it is possible to show that $\mathcal{G}_M < \mathcal{G}_{M+1}$ for the lower-bound case and $\mathcal{G}_M > \mathcal{G}_{M+1}$ for the case of an upper bound [14]. This provides another stopping-criterion.

As the accumulation of truncation errors can lead to unreasonable results even before the violation of the above property, we propose to keep a moving average of the last k approximations and employ the 3σ-rule to detect improbable results. To dismiss unlikely results, the 3σ-rule makes use of Chebyshev's inequality, according to which the probability for a sample from a probability distribution with finite expected value and variance to be farther away from the expected value than three times the variance is roughly 11%. This heuristic is justified by the guaranteed convergence in the absence of numerical errors.

After having explained the algorithm, a few remarks are in order:

(i) In this version of the algorithm, we only consider the Gauss quadrature. This is mainly due to the fact that obtaining good lower or upper bounds on the spectrum of A is in general not possible because of the size of its dimensions. Analogously to [4], our algorithm can nevertheless be adapted to perform Gauss-Radau or Gauss-Lobatto approximations.

(ii) To improve numerical stability and prevent 'ghost' eigenvalues from occurring, it could be beneficial to perform reorthogonalization. Due to the MPO representation, this would, however, be very costly and not necessarily result in a large improvement. Thus, we do not consider this extension. It can, however, be easily added to the algorithm.

(iii) In the presented algorithm, we stick to the canonical way of orthogonalizing the new matrix against the old matrices individually. In the case of exactly stored

ETNA
Kent State University and
Johann Radon Institute (RICAM)

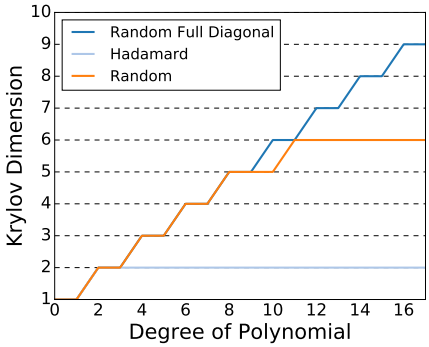


FIG. 4.1. *The number of Krylov dimensions needed to converge to the correct value of* $\mathrm{Tr}\sum_{i=0}^{g}(A)$ *over the degree of the polynomial for the exact version of the algorithm with* $L = \log N = 10$.

matrices/vectors, this scheme increases the numerical stability. Since we now employ approximations of the exact matrices, it might, however, be worth considering to compute α_i first and then optimize the sum containing both U_i and U_{i-1}. The advantage of being able to optimize the whole expression at once might outweigh the disadvantage of orthogonalizing against both matrices simultaneously. On the other hand, computing α_i first might lead to different and possibly worse results.

(iv) As we have stated above, it is possible to obtain approximation errors from both `multiplyAndOptimize` and `sumAndOptimize`. But these errors naturally only refer to the current optimization and do not allow for strict bounds on the overall error. One could of course try to increase the bond dimension for each individual optimization until its error converges to make sure the partial result is close to exact. The problem here is that due to the possibly exponential growth of the bond dimension needed for exactness, D_{max} is typically reached within very few iterations. From this point on, it is not possible to increase D any more and so, the information about the error provides little useful information. This is why we have resorted to the approach of checking for the violation of theoretically guaranteed behaviour.

(v) From the above explanations it is clear that K and D_{max} are the parameters that control the accuracy of the approximation. For the algorithm to be of use for very high-dimensional matrices, we must impose the restriction that $K, D_{max} \in \mathcal{O}(poly(L))$. This property is particularly relevant for quantum mechanical simulations where N grows exponentially with the number of particles.

While it is very difficult to rigorously analyze the convergence behaviour of our method when facing truncation errors introduced by the MPO/MPS representation, we are able to make a statement for the case of exact arithmetic without truncations.

THEOREM 4.1. *For* $f : \mathbb{C}^N \times \mathbb{C}^N \to \mathbb{C}^N \times \mathbb{C}^N$ *being a polynomial of degree* g *or a smooth function that is arbitrarily well approximated by its power series expansion up to* g *and exact arithmetic without truncations, Algorithm 2 converges to the exact value in* $\min\{g^*, \lfloor g/2 \rfloor + 1\}$ *steps, where* g^* *is the degree of the minimal polynomial of* A.

Proof. 1.) It follows from the fact that we perform a Gauss quadrature, which for i nodes is exact for all polynomials up to degree $2i - 1$, and employ the full identity matrix as starting matrix for our algorithm that our method requires maximally $\lfloor g/2 \rfloor + 1$ steps to converge to the exact value. 2.) Furthermore, it follows from the underlying Lanczos algorithm that our method will converge in maximally g^* steps, where g^* is the degree of the minimal polynomial of A. From 1.) and 2.), it directly follows that our method will converge in $\min\{g^*, \lfloor g/2 \rfloor + 1\}$

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 4.1
A listing of the complexity of the subfunctions of Algorithm 2. $L = \log N$ is the number of tensors of the MPOs, d is the physical dimension. For simplicity, all U_i are assumed to have bond dimension D_{max} and T_i is assumed to be in $\mathbb{R}^{K\times K}$.

Function	Complexity
`contract`	$\mathcal{O}(LD_{max}^3 d^2)$
`multiplyAndOptimize`	$\mathcal{O}(LD_{max}^3 D_A d^2)$
`sumAndOptimize`	$\mathcal{O}(LD_{max}^3 d^2)$
`multiplyScalar`	$\mathcal{O}(D_{max}^2 d^2)$
`spectralDecomposition`	$\mathcal{O}(K^3)$
`checkStop`	$\mathcal{O}(1)$

steps in the exact case. □

It is worth noting that this result would also hold for the global Lanczos method as introduced in [4] if the authors would not explicitly restrict themselves to starting matrices of significantly smaller dimension than that of A, as that restriction prevents the Gauss quadrature from converging to the exact value. While truncation errors introduced by the tensor network representation will deteriorate the convergence behaviour and the statement above will therefore not be directly applicable to practical applications in general, it is still instructive to understand the behaviour of the algorithm in the ideal case.

To illustrate the result, Figure 4.1 depicts the number of steps needed by the algorithm without truncations to converge to the correct result with a relative error of 10^{-12} for increasing degrees of the polynomials of the form $f(A) = \sum_{i=0}^{g} A^i$ and $N = 1024$. For a full diagonal matrix with entries randomly sampled from the uniform distribution over $[-1, 1]$, the algorithm in fact needs as many steps as required by the Gauss quadrature to reach exactness. On the other hand, for the Hadamard matrix, which only has two distinct eigenvalues, the method always converges in two steps. A full symmetric matrix with entries uniformly sampled from $[0, 1]$ typically only has a few dominant eigenvalues, which corresponds to the number of steps needed to converge being six in this case.

We will conclude this section with an analysis of the complexity of our algorithm. The complexities of the subfunctions of Algorithm 2 are listed in Table 4.1. For the analysis of `multiplyAndOptimize`, we have assumed D_A to be smaller or of the same order as D_{max}. If it were significantly larger, the complexity would change to $\mathcal{O}(LD_{max}^2 D_A^2 d^2)$. Note that this analysis does not extend to the number of sweeps necessary for the optimizations to converge. For the spectral decomposition, we have for simplicity assumed all T_i to be of size $K \times K$. Combining all the different results, we thus find that the overall complexity of Algorithm 2 is $\mathcal{O}(KLD_{max}^3 D_A d^4)$ with $L = \log N$ and since we require $K, D_{max} \in \mathcal{O}(\log N)$, this is equivalent to $\mathcal{O}(poly(L))$.

5. Numerical results. In this section we present numerical results obtained for a challenging problem of relevance in quantum many-body physics. Our goal thereby is to study the convergence of the results with increasing K and D_{max}. The problem we consider is the approximation of the von Neumann entropy. For a quantum state ρ[1], the von Neumann entropy is given by $S = -\mathsf{Tr}\rho \log \rho$. In the following, we will focus on the case of states of the form $\rho = e^{-\beta H} / \mathsf{Tr} e^{-\beta H}$, i.e., thermal equilibrium states for a Hamiltonian H at a certain

[1] ρ is a positive operator with unit trace, representing an ensemble of pure states.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

inverse temperature β[2]. Here, we assume H to be the Ising Hamiltonian with open boundary conditions that is given by

$$H = J \sum_{i=1}^{L-1} \sigma_i^x \sigma_{i+1}^x + g \sum_{i=1}^{L} \sigma_i^z + h \sum_{i=1}^{L} \sigma_i^x,$$

where $\sigma^{\{x,y,z\}}$ are the Pauli matrices

$$\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma^y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

Note, however, that here by $\sigma_i^{\{x,y,z\}}$ we actually denote the tensor product $I_1 \cdots \otimes I_{i-1} \otimes \sigma_i^{\{x,y,z\}} \otimes I_{i+1} \otimes \cdots \otimes I_L$ and analogously for $\sigma_i^{\{x,y,z\}} \sigma_{i+1}^{\{x,y,z\}}$ for simplicity of notation. The Hamiltonian describes a chain of spin particles with nearest neighbour interactions and two magnetic fields acting only on the individual particles. This choice of H has the advantage that it is exactly solvable for $h = 0$, a case commonly known as 'transverse Ising', and thus opens the possibility to obtain a reference solution for system sizes for which it could otherwise not be obtained [20, 32, 34]. Hence, in the following we will assume $h = 0$.

It is possible to find an MPO approximation to the thermal equilibrium state ρ by means of standard MPS-techniques [12, 36, 41]. It is customary to use a *purification* ansatz for this purpose, where $\rho(\beta/2)$ is approximated by standard imaginary time evolution, and the whole state is then written as $\rho \propto \rho(\beta/2)^* \rho(\beta/2)$. In the context of our algorithm, nevertheless, applying exactly this ρ involves a larger cost and worse numerical condition. Instead, we apply the method as described above to $\rho(\beta/2)$ and absorb the necessary squaring into the function that is to be approximated. In our case this means that instead of computing $f(\lambda_i) = \lambda_i \log \lambda_i$ for each Ritz-value, we can compute $f'(\lambda_i) = \lambda_i^2 \log \lambda_i^2$ for λ_i corresponding to $\rho(\beta/2)$. This allows us to apply the algorithm to a possibly much more benign input at the cost of an only slightly more complicated function. Due to truncation errors, the operator $\rho(\beta/2)$ may not be exactly Hermitian. This can be easily accounted for by taking its Hermitian part, $\frac{1}{2}[\rho(\beta/2)^* + \rho(\beta/2)]$, which is an MPO with at most twice the original bond dimension. In our experiments we, however, did not find this to be necessary.

Apart from the entropy, another interesting function to examine would have been the trace norm given by $||\rho||_1 = \mathsf{Tr}\sqrt{\rho^*\rho}$, i.e., the sum of the singular values. But as we only consider positive matrices in this scenario, this sum is equal to the trace which we know to be equal to 1 due to the normalization of the thermal state. Directly related to this, we find that α_1 as computed by our algorithm is given by

$$\alpha_1 = \mathsf{Tr} U_1^* V_1 = \mathsf{Tr} U_1^* \rho U_1 = \frac{1}{\beta_1^2} \mathsf{Tr} U^* \rho U = \frac{1}{\beta_1^2} \mathsf{Tr}\rho.$$

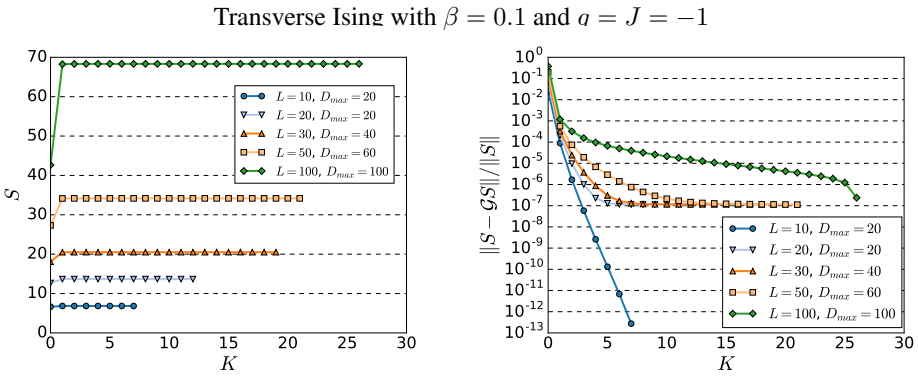
So, our algorithm computes the trace of the input MPO A in one step. We verified this result numerically and found it to hold for all considered cases. This means that the algorithm also computes the trace norm of $\rho(\beta)$ in one step in this case.

It is well known that if its sign is constant over the considered interval, then the $2K$th derivative of f determines whether $\mathcal{G}f$ poses a lower or upper bound of the true value. In our case and for $K > 1$, it is given by

$$\frac{d^{2K} - \lambda_i^2 \ln \lambda_i^2}{d^{2K} \lambda_i} = \frac{4(2K-3)!}{\lambda_i^{2K-2}},$$

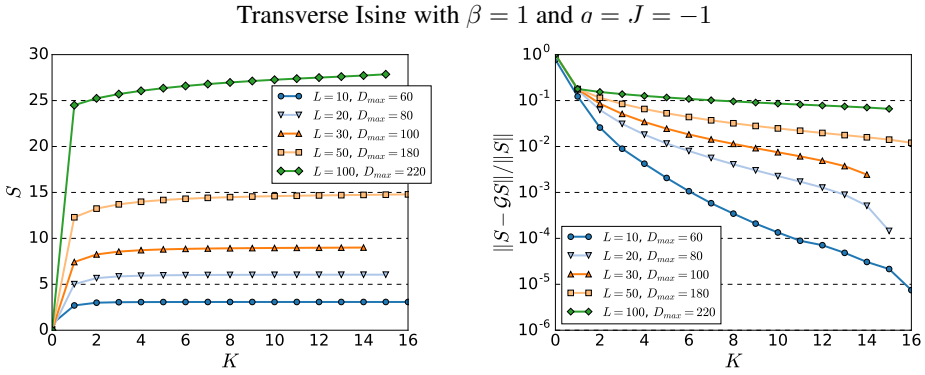
[2]Note that β is not related to the β_i computed by our algorithm.

ETNA
Kent State University and
Johann Radon Institute (RICAM)



(a) Entropy over K. (b) Relative error in the entropy over K.

FIG. 5.1. *Convergence behavior of the algorithm for $L \in \{10, 20, 30, 50, 100\}$, $\beta = 0.1$ and varying Krylov-dimension K. In (a), the convergence of the approximation is depicted. In (b), the convergence of the relative error is shown.*



(a) Entropy over K. (b) Relative error in the entropy over K.

FIG. 5.2. *Convergence behavior of the algorithm for $L \in \{10, 20, 30, 50, 100\}$, $\beta = 1$, and varying Krylov-dimension K. In (a), the convergence of the approximation is depicted. In (b), the convergence of the relative error is shown.*

with λ_i being the ith eigenvalue of ρ. Hence, we can expect our algorithm to provide increasingly tight lower bounds for the correct value. We use the violation of this property as a stopping criterion to account for the situation when truncation errors become too large. Additionally, we keep the average of the last three or four—depending on β—approximations and employ the aforementioned 3σ-rule. In case these stopping criteria are not met, we terminate the algorithm when the absolute difference between successive approximations is below 10^{-10}.

In our experiments we considered systems of size $L \in \{10, 20, 30, 50, 100\}$, Hamiltonian parameters $J = g = 1$, and inverse temperatures $\beta \in \{0.1, 1.0\}$. The bond dimension used to obtain $\rho(\frac{\beta}{2})$ was set to 20 for all cases. The convergence of the approximation as well as the relative error for $\beta = 0.1$ and $\beta = 1$ are shown in Figure 5.1 and Figure 5.2, respectively. Note that in order to compute the relative error, we used numerical diagonalization for $L = 10$ and the analytical solution [32] for $L > 10$. In all cases except for $L = 10$ and $\beta = 0.1$, where the algorithm was stopped when the distance between successive approximations reached the threshold, the algorithm was stopped when meeting the 3σ stopping criterion. Table 5.1

ETNA
Kent State University and
Johann Radon Institute (RICAM)

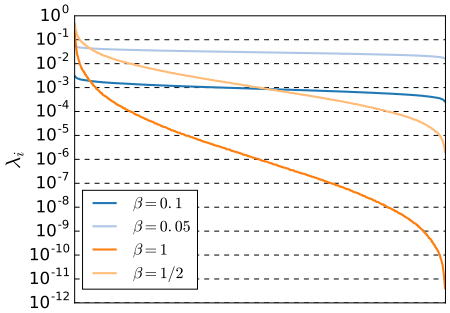


FIG. 5.3. *The spectra of $\rho(\beta)$ and $\rho(\frac{\beta}{2})$ of the transverse Ising Hamiltonian with $J = g = -1$ for $L = 10$ and $\beta \in \{0.1, 1\}$.*

shows the change of the relative error of the final approximations with growing D_{max} for $L \in \{50, 100\}$ and $\beta \in \{0.1, 1\}$.

For the case of $\beta = 0.1$ we observe fast convergence to good approximations in K and D_{max} as shown in Figure 5.1. The maximal bond dimension required for good convergence only grows mildly with L allowing our method to scale very well with the size of the input. The plots in Figure 5.1b show a plateau in the relative error at 10^{-7}. This corresponds to the non-vanishing difference between the exact solution and the numerical MPO used as input. Note that for $L = 10$, where the input is exact, the method is able to achieve a smaller error.

The results for the larger inverse temperature $\beta = 1$ paint a slightly different picture. While the overall behavior of our method remains the same and Figure 5.2a depicts good convergence especially for $L < 100$, Figure 5.2b shows that the relative error achieved is noticeably worse than for the case of $\beta = 0.1$. It also seems that larger values of D_{max} are required to achieve reasonable results. This phenomenon naturally becomes more pronounced with larger L.

We conjecture that the difference in the performance observed for the two considered values of β has two main reasons. Firstly, the bond dimension required for a good approximation of ρ grows with larger β. This might in turn increase the value of D_{max} required for good accuracy, and, correspondingly, increase the approximation error incurred by ρ, so that the computed function will be farther from the analytical solution. Secondly, the spectral properties of the obtained MPOs for the two considered cases are significantly different. In Figure 5.3, we show the spectra of ρ for both values of β and $\beta/2$ for $L = 10$, respectively. It is clearly visible that $\beta = 0.1$ poses a much more benign case. This is underlined by the condition numbers, which are roughly 11.9, $5.68 \cdot 10^{10}$, 3.5 and $2.38 \cdot 10^{5}$ for $\beta = 0.1$, $\beta = 1$, $\beta = 0.05$ and $\beta = 0.5$, respectively. They show that $\beta = 1$ in fact yields highly ill-conditioned MPOs, functions of which are hard to approximate. These considerations also make it clear that by absorbing the necessary squaring of the eigenvalues into the function, we obtain much more well-conditioned input MPOs of lower bond dimension. Hence, we can conclude that our method, while being influenced by both aforementioned factors, is relatively robust and even works reasonably well for very difficult cases.

Table 5.1 illustrates that while our method achieves a low error for $\beta = 0.1$ and a moderate error for $\beta = 1$ even for a small D_{max}, it profits from an increase of the maximal bond dimension. This effect is more pronounced for the lower β which is likely due to the reasons mentioned above. For instance, for $L = 100$ and $\beta = 0.1$, the error decreases by two orders of magnitude from 10^{-5} to about 10^{-7} when D_{max} is raised from 20 to 180,

ETNA
Kent State University and
Johann Radon Institute (RICAM)

TABLE 5.1
Relative error in the entropy of the transverse Ising Hamiltonian with $J = -1$ for $L \in \{100, 50\}$, $g \in \{1, 0.1\}$ and increasing values of the maximal bond dimension D_{max}.

D_{max}	$L = 100, \beta = 0.1$	$L = 100, \beta = 1$	$L = 50, \beta = 0.1$	$L = 50, \beta = 1$
20	$1.14 \cdot 10^{-05}$	$9.75 \cdot 10^{-02}$	$2.73 \cdot 10^{-07}$	-
40	$3.96 \cdot 10^{-06}$	$9.14 \cdot 10^{-02}$	$8.04 \cdot 10^{-08}$	$2.75 \cdot 10^{-02}$
60	$1.67 \cdot 10^{-06}$	$9.32 \cdot 10^{-02}$	$1.12 \cdot 10^{-07}$	$1.84 \cdot 10^{-02}$
80	$1.41 \cdot 10^{-06}$	$7.95 \cdot 10^{-02}$	$1.15 \cdot 10^{-07}$	$1.89 \cdot 10^{-02}$
100	$2.38 \cdot 10^{-07}$	$7.81 \cdot 10^{-02}$	-	$1.26 \cdot 10^{-02}$
120	$4.01 \cdot 10^{-07}$	$7.37 \cdot 10^{-02}$	-	$1.11 \cdot 10^{-02}$
140	$2.77 \cdot 10^{-07}$	$7.04 \cdot 10^{-02}$	-	$1.05 \cdot 10^{-02}$
160	$2.29 \cdot 10^{-07}$	$7.11 \cdot 10^{-02}$	-	$9.56 \cdot 10^{-03}$
180	$6.43 \cdot 10^{-08}$	$7.03 \cdot 10^{-02}$	-	$9.06 \cdot 10^{-03}$

TABLE 5.2
Comparison of the runtime in seconds between SciPy's `expm` function and Algorithm 2 for $\mathsf{Tr}\, exp(A)$ and increasing values of L. Lower runtimes are printed bold.

L	`scipy.expm`	`Algorithm 2`
10	**1**	23
11	**3**	23
12	**20**	31
13	129	**43**
14	994	**50**

which still constitutes a strong truncation. It is conceivable that a further increase of the maximal bond dimension would improve the accuracy. For $L = 50$ and $\beta = 0.1$, the error still decreases when D_{max} is raised from 20 to 40, but a further increase shows now effect due to the aforementioned small error already introduced by $\rho(\beta)$. We also found that D_{max} limits the number of basis MPOs that can be successfully orthogonalized and therefore effectively controls the maximally reachable K. Hence, D_{max} can be regarded as the decisive parameter of our method.

We do not provide a proper comparison to other methods at this point because of the simple reason that to the best of the authors knowledge there is no other algorithm that can solve the considered kind of problem for $L \gg 20$, but for $L \leq 20$ we expect the existing highly optimized methods to outperform our method in terms of runtime. However, from our analysis of the complexity we know that our method scales as $\mathcal{O}(\log N)$ with the matrix dimension N when all other parameters are kept constant. This is in contrast to methods based on full diagonalization, which scales as $\mathcal{O}(N^3)$. One can therefore expect our method to eventually outperform such approaches as well as approaches with super-logarithmic scaling in general for growing N when all other parameters are kept fixed. In Table 5.2, we provide a small comparison of the runtime of our algorithm to that of the `expm` function from the SciPy package for Python which implements a squaring-and-scaling approach. We approximated $\mathsf{Tr}\exp(A)$ where A is the transverse Ising Hamiltonian with $L \in \{10, 11, 12, 13, 14\}$. We set $D_{max} = 30$ which yielded good approximations and $K = 100$ to prevent the method from reaching the maximal Krylov dimension before convergence. We then let the algorithm run until the relative error between successive approximations was smaller than 10^{-8}. The results were obtained on an Intel i7-4790 CPU with 32GB RAM. While for smaller matrices up to $L = 12$ our method is significantly slower, it does not suffer from the same drastic increase in runtime with growing matrix size and hence outperforms `expm` for $L > 12$. We

ETNA
Kent State University and
Johann Radon Institute (RICAM)

found the `expm` function to break down due to memory requirements for $L > 14$. The results show that for the given parameters, the runtime of our method remains small for all considered matrix sizes. However, depending on D_{max} the runtime can significantly increase to several hours. Note though that there exist several ways to speed up computations in the MPO/MPS representation via exploitation of symmetries which we did not consider in our implementation.

6. Discussion. In this work, we have introduced a method to approximate functionals of the form $\mathsf{Tr} f(A)$ for matrices of dimension much larger than 2^{20}. We started by giving an overview of the mathematical and algorithmic ideas behind the method. Following this, a detailed description of the algorithm together with an analysis of its complexity was provided. We then presented numerical results for a challenging problem in quantum many-body physics. These results indicate that our method is able to produce good approximations for a number of Krylov steps and a maximal bond dimension logarithmic in the size of the matrix as long as the matrix exhibits some structure that can be expressed well in the MPO/MPS-formalism and is moderately well-conditioned. It was also shown that the maximal allowed bond dimension is the decisive parameter of the algorithm.

There are several ways to build upon this work. Firstly, an investigation of preconditioning methods suitable for our method could be fruitful. Secondly, a more thorough analysis of the effect of the approximation error introduced by the tensor network formalism on the approximation error of the Gauss quadrature would be an interesting addition. Thirdly, the connection of the approximability of a matrix by an MPO to the convergence behavior of our method could provide deeper understanding. Fourthly, it could be investigated which of the many improvements over the normal Gauss quadrature, as for instance [30], can be incorporated into our algorithm to make better use of the expensive information obtained in the Krylov iteration. Finally, the method naturally could be applied to solve practical problems of interest.

While our method was tested for a quantum mechanical problem, it is of course general in nature and can be applied to any case where the matrix in question can be formulated as an MPO or well approximated by one. Especially for matrices of dimension larger than 2^{10} that however can still be explicitly stored, it might be interesting to consider computing the desired function for the MPO representation.

Acknowledgements. This work was partly funded by the *Elite Network of Bavaria* (ENB) via the doctoral programme *Exploring Quantum Matter*.

REFERENCES

[1] W. E. ARNOLDI, *The principle of minimized iteration in the solution of the matrix eigenvalue problem*, Quart. Appl. Math., 9 (1951), pp. 17–29.
[2] M. BACHMAYR, R. SCHNEIDER, AND A. USCHMAJEW, *Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations*, Found. Comput. Math., 16 (2016), pp. 1423–1472.
[3] J. BAGLAMA, C. FENU, L. REICHEL, AND G. RODRIGUEZ, *Analysis of directed networks via partial singular value decomposition and Gauss quadrature*, Linear Algebra Appl., 456 (2014), pp. 93–121.
[4] M. BELLALIJ, L. REICHEL, G. RODRIGUEZ, AND H. SADOK, *Bounding matrix functionals via partial global block Lanczos decomposition*, Appl. Numer. Math., 94 (2015), pp. 127–139.
[5] D. CALVETTI, L. REICHEL, AND D. C. SORENSEN, *An implicitly restarted Lanczos method for large symmetric eigenvalue problems*, Electron. Trans. Numer. Anal., 2 (1994), pp. 1–21. http://etna.ricam.oeaw.ac.at/vol.2.1994/pp1-21.dir/pp1-21.pdf
[6] J. CULLUM AND W. DONATH, *A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices*, in 1974 IEEE Conference on Decision and Control, IEEE, New York 1974, pp. 505–509.
[7] P. J. DAVIS AND P. RABINOWITZ, *Methods of Numerical Integration*, 2nd ed., Dover, Mineola, 2007.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[8] L. ELBOUYAHYAOUI, A. MESSAOUDI, AND H. SADOK, *Algebraic properties of the block gmres and block Arnoldi methods*, Electron. Trans. Numer. Anal., 33 (2009), pp. 207–220. http://etna.ricam.oeaw.ac.at/vol.33.2008-2009/pp207-220.dir/pp207-220.pdf

[9] E. ESTRADA AND D. J. HIGHAM, *Network properties revealed through matrix functions*, SIAM Rev., 52 (2010), pp. 696–714.

[10] M. FANNES, B. NACHTERGAELE, AND R. F. WERNER, *Finitely correlated states on quantum spin chains*, Comm. Math. Phys., 144 (1992), pp. 443–490.

[11] C. FENU, D. MARTIN, L. REICHEL, AND G. RODRIGUEZ, *Block Gauss and anti-Gauss quadrature with application to networks*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1655–1684.

[12] J. J. GARCÍA-RIPOLL, *Time evolution of matrix product states*, New J. Phys., 8 (2006), Art. no. 305, 22 pages.

[13] G. H. GOLUB, F. T. LUK, AND M. L. OVERTON, *A block Lanczos method for computing the singular values of corresponding singular vectors of a matrix*, ACM Trans. Math. Software, 7 (1981), pp. 149–169.

[14] G. H. GOLUB AND G. MEURANT, *Matrices, Moments and Quadrature with Applications*, Princeton University Press, Princeton, 2010.

[15] L. GRASEDYCK, D. KRESSNER, AND C. TOBLER, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitt., 36 (2013), pp. 53–78.

[16] R. G. GRIMES, J. G. LEWIS, AND H. D. SIMON, *A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 228–272.

[17] M. B. HASTINGS, *An area law for one-dimensional quantum systems*, J. Stat. Mech. Theory Exp., (2007), pp. P08024, 14.

[18] T. HUCKLE, K. WALDHERR, AND T. SCHULTE-HERBRÜGGEN, *Computations in quantum tensor networks*, Linear Algebra Appl., 438 (2013), pp. 750–781.

[19] I. C. F. IPSEN AND C. D. MEYER, *The idea behind Krylov methods*, Amer. Math. Monthly, 105 (1998), pp. 889–899.

[20] D. KAREVSKI, *Ising quantum chains*, Preprint on arXiv, 2006. https://arxiv.org/abs/cond-mat/0611327

[21] B. KHOROMSKIJ, *O(d log n)-quantics approximation of $N - d$ tensors in high-dimensional numerical modeling*, Constr. Approx., 34 (2011), pp. 257–280.

[22] D. KRESSNER AND C. TOBLER, *Low-rank tensor Krylov subspace methods for parametrized linear systems*, SIAM J. Matrix Anal. Appl., 32 (2011), pp. 1288–1316.

[23] A. KRYLOV, *On the numerical solution of the equation by which the frequency of small oscillations is determined in technical problems*, Izv. Akad. Nauk SSSR Ser. Fiz.-Mat, 4 (1931), pp. 491–539.

[24] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, J. Research Nat. Bur. Standards, 45 (1950), pp. 255–282.

[25] P. L. MONTGOMERY, *A block Lanczos algorithm for finding dependencies over* $\mathrm{GF}(2)$, in Advances in Cryptology — EUROCRYPT '95, L. C. Guillou and J.-J. Quisquater, eds., vol. 921 of Lecture Notes in Comput. Sci., Springer, Berlin, 1995, pp. 106–120.

[26] M. E. J. NEWMAN, *Networks: An Introduction*, Oxford University Press, Oxford, 2010.

[27] I. OSELEDETS, *Tensor-train decomposition*, SIAM J. Sci. Comput., 33 (2011), pp. 2295–2317.

[28] D. PEREZ-GARCIA, F. VERSTRAETE, M. M. WOLF, AND J. I. CIRAC, *Matrix product state representations*, Quantum Inf. Comput., 7 (2007), pp. 401–430.

[29] B. PIRVU, V. MURG, J. I. CIRAC, AND F. VERSTRAETE, *Matrix product operator representations*, New J. Phys., 12 (2010), pp. 025012, 13.

[30] L. REICHEL, M. M. SPALEVIĆ, AND T. TANG, *Generalized averaged Gauss quadrature rules for the approximation of matrix functionals*, BIT, 56 (2016), pp. 1045–1067.

[31] Y. SAAD AND M. H. SCHULTZ, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.

[32] S. SACHDEV, *Quantum Phase Transitions*, Wiley Online Library, New York, 2007.

[33] U. SCHOLLWÖCK, *The density-matrix renormalization group in the age of matrix product states*, Ann. Physics, 326 (2011), pp. 96–192.

[34] S. SUZUKI, J.-I. INOUE, AND B. K. CHAKRABARTI, *Quantum Ising Phases and Transitions in Transverse Ising Models*, 2nd ed., vol. 862 of Lecture Notes in Physics, Springer, Heidelberg, 2013.

[35] F. VERSTRAETE AND I. CIRAC, *Matrix product states represent ground states faithfully*, Phys. Rev. B, 73 (2006), Art. no., 094423, 8 pages.

[36] F. VERSTRAETE, J. GARCÍA-RIPOLL, AND I. CIRAC, *Matrix product density operators: simulation of finite-temperature and dissipative systems*, Phys. Rev. Lett., 93 (2004), Art. no. 207204, 4 pages.

[37] F. VERSTRAETE, V. MURG, AND I. CIRAC, *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems*, Adv. in Phys, 57 (2008), pp. 143–224.

[38] F. VERSTRAETE, D. PORRAS, AND I. CIRAC, *Density matrix renormalization group and periodic boundary conditions: a quantum information perspective*, Phys. Rev. Lett., 93 (2004), Art. no. 227205, 4 pages.

ETNA
Kent State University and
Johann Radon Institute (RICAM)

[39] G. VIDAL, *Efficient classical simulation of slightly entangled quantum computations*, Phys. Rev. Lett.,, 91 (2003), Art. no., 147902, 4 pages.
[40] K. WALDHERR, *Numerical Linear and Multilinear Algebra in Quantum Control and Quantum Tensor Networks*, Verlag Dr. Hut, Munich, 2014.
[41] M. ZWOLAK AND G. VIDAL, *Mixed-state dynamics in one-dimensional quantum lattice systems: a time-dependent superoperator renormalization algorithm*, Phys. Rev. Lett., 93 (2004), Art. no. 207205, 4 pages.

B Towards a better understanding of the matrix product function approximation algorithm in application to quantum physics

Authors Moritz August and Prof. Dr. Thomas Huckle

Citation arXiv:1709.06847v2 [cs.NA], 2018

Summary This article poses a direct continuation of the work shown in Appendix A. After we had introduced our method to approximate functions $\mathsf{Tr}f(A)$, this work aimed at the extension of our theoretical understanding of the method. To achieve this, we consequentially analyzed the partial results produced by the underlying algorithm and showed that the basis matrices U_i inherit several symmetries and other properties from the input A. More concretely, we found that $\mathsf{Tr}U_i = 0$ if $\mathsf{Tr}A = 0$, all U_i commute with A and that the U_i inherit symmetry, persymmetry and centro symmetry from A (or Hermiticity in the complex case). We argued that these properties of our method can be leveraged to detect and correct approximation errors in the U_i. Furthermore, we proved the existence of a more computationally efficient variant of our method for inputs A exhibiting a spectrum that is point-symmetric around zero. For such inputs, we found the projected matrix T_K to take the form

$$T_K = \begin{bmatrix} 0 & \beta_2 & & \mathbf{0} \\ \beta_2 & 0 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & 0 \end{bmatrix} \tag{B.1}$$

where all $\alpha_i = 0$. This poses an analytical guarantee for orthogonality between subsequent basis matrices U_{i-1} and U_i. We illustrated the relevance of this finding by proving that a large class of spin Hamiltonians with finite-length neighbour interactions do in fact exhibit such a spectrum. Finally, we provided numerical results to support our analytical findings. We compared the runtime behavior of the specialized version of our method with the more general one and indeed found a speedup to exist. Furthermore, the numerical findings supported the results of our complexity analysis shown in Appendix A.

Contribution For this work, the contributions of the author again range from the derivation of the analytical findings, over the implementation and evaluation of the newly discovered variant of the algorithm to the writing of the article. In the derivation of the analytical results, Prof. Huckle kindly provided support by discussing ideas. Especially for the results concerning symmetric spectra of Hamiltonians, Prof. Huckle supported the author in discussing approaches to formalize and prove this property. The implementation end evaluation of the adapted algorithm was again based on the tensor network library provided previously by Dr. Dr. Bañuls and conducted by the author. All figures shown in the article were prepared by the author and also the text itself was written by author. Here, Prof. Huckle again kindly provided support by proof-reading the article.

TOWARDS A BETTER UNDERSTANDING OF THE MATRIX PRODUCT FUNCTION APPROXIMATION ALGORITHM IN APPLICATION TO QUANTUM PHYSICS

MORITZ AUGUST* AND THOMAS HUCKLE†

Abstract. We recently introduced a method to approximate functions of Hermitian Matrix Product Operators or Tensor Trains that are of the form $\mathsf{Tr}f(A)$. Functions of this type occur in several applications, most notably in quantum physics. In this work we aim at extending the theoretical understanding of our method by showing several properties of our algorithm that can be used to detect and correct errors in its results. Most importantly, we show that there exists a more computationally efficient version of our algorithm for certain inputs. To illustrate the usefulness of our finding, we prove that several classes of spin Hamiltonians in quantum physics fall into this input category. We finally support our findings with numerical results obtained for an example from quantum physics.

Key words. tensor decompositions, matrix product states, tensor trains, numerical analysics, Lanczos method, Gauss quadrature, quantum physics

AMS subject classifications. 65F60, 65D15, 65D30, 65F15, 46N50, 15A69

1. Introduction. Approximating functions of the form $\mathsf{Tr}f(A)$ where $f : \mathbb{C}^{N\times N} \to \mathbb{C}^{N\times N}$ is analytic and smooth for large Hermitian matrices $A \in \mathbb{C}^{N\times N}$ is a problem of interest in areas such as computational chemistry, graph theory or quantum physics. In quantum physics, fundamental properties of states of many particle systems such as the entanglement entropy, the trace norm, heat capacity or expectation values are defined as functions of this form [48].

While computing $\mathsf{Tr}f(A)$ is not challenging for small to medium size matrices, it becomes significantly harder for larger matrices of size 2^L with $L \gg 20$ where numerical diagonalization becomes computationally infeasible. We have recently addressed this issue by presenting the first algorithm [2] that is able to approximate such functions even for matrices of very high dimensionality via a combination of the global Krylov method with its connection to Gauss-type quadrature and the matrix product state (or tensor train) tensor decomposition scheme. Our method constructs a basis $[U_1, \dots, U_K]$ of $\mathrm{span}\{A^0, A^1, \dots, A^{K-1}\}$ where $U_i \in \mathbb{C}^{N\times N}$ and yields the projection T_K of A onto that space, which is used to approximate the desired function.

We have shown that our algorithm converges to the exact result or an arbitrarily good approximation thereof in the case of exact arithmetics and exact representation of the U_i. While this result is instructive to understand the theoretical capability of the method, in practice the tensor decomposition is used to approximate A and the U_i and hence introduces an approximation error into the calculations. Unfortunately, it is very difficult to analyze the propagation of such approximation errors over the course of a complete run of the algorithm and their influence on the final function approximation. It is therefore important to gain a deeper understanding of theoretical properties of partial results of the computation, namely the U_i and T_i, in order to be able to detect and possibly correct unwanted artifacts caused by the approximation errors. In addition to that, we would of course like to avoid unnecessary operations that might introduce approximation errors and waste runtime whenever possible. Thus, in this work we present several results regarding analytical properties of the U_i and T_i in the exact case and also show how these results can be used to obtain a more efficient version of our algorithm for a certain case of input matrices A.

*Department of Informatics, Technical University of Munich, 85748 Garching, Germany (august@in.tum.de)
†Department of Informatics, Technical University of Munich, 85748 Garching, Germany (huckle@in.tum.de)

Algorithm 1: Approximation Algorithm

Input : MPO $A[D_A] \in \mathbb{C}^{N\times N}$, Starting orthogonal MPO $U[D_{init}] \in \mathbb{C}^{N\times N}$, Number of Dimensions K, Maximal Bond-Dimension D_{max}, Stopping Criteria $\mathcal{S}$

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
$D \leftarrow D_{init}$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \sqrt{\texttt{innerProduct}(V_{i-1}, V_{i-1})}$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $U_i \leftarrow \texttt{multiply}(1/\beta_i, V_{i-1})$;
  $D \leftarrow \min(D_{max}, D \cdot D_A)$;
  $V_i \leftarrow \texttt{multiply}(A, U_i, D)$;
  $D \leftarrow \min(D_{max}, D + D_{U_{i-1}})$;
  $V_i \leftarrow \texttt{sum}(V_i, -\beta_i U_{i-1}, D)$;
  $\alpha_i \leftarrow \texttt{innerProduct}(U_i, V_i)$;
  $D \leftarrow \min(D_{max}, D + D_{U_i})$;
  $V_i \leftarrow \texttt{sum}(V_i, -\alpha_i U_i, D)$;
  $V_T \Lambda_T V_T^* \leftarrow \texttt{spectralDecomposition}(T_i)$;
  $\mathcal{G}f \leftarrow \beta_1^2 e_1^T V_T f(\Lambda_T) V_T^* e_1$;
  if $\texttt{checkStop}(\mathcal{G}f, \Lambda_T, \mathcal{S})$ **then**
    break ;
  end
end

Output: Approximation $\mathcal{G}f$ of $\mathsf{T}\mathsf{r}f(A)$

While our algorithm is of general nature, as was hinted at above an important field of application can be found in numerical quantum physics. Here, tensor networks have already been applied with great success for some time [19, 49, 57, 27, 55, 60, 59, 56, 43] but so far a method to approximate functions of the type considered here was lacking. Because of this, we will additionally present an analysis of possible use cases of our newly discovered algorithmic improvement for the application of spin Hamiltonians, an important problem in numerical quantum physics.

The rest of this work is structured as follows: in Section 2, we briefly introduce our method. Equipped with this knowledge, we present our analytical findings in Section 3. In Section 4, we then present our analysis of possible applications of the previously introduced results in quantum physics. Following this, we proceed to provide numerical evidence of the correctness of our claims regarding the existence of an improved version of our algorithm in Section 5. Finally, we conclude this work in Section 6.

2. The Algorithm. As we have stated above, our goal is to approximate functions of the form $\mathsf{T}\mathsf{r}f(A)$. For smaller to medium sized matrices, there already exists a well-established method to achieve this in performing a Gauss-type quadrature via the projection of A onto a Krylov space starting with a carefully chosen initial vector [7, 5, 21, 53, 44].

For symmetric or Hermitian matrices, the global Lanczos method recently was introduced as a formulation of the classical Lanczos method in terms of basis *matrices* with the

Frobenius inner product defined as

$$\langle U_i, U_j \rangle = \mathsf{Tr} U_i^* U_j$$

and $U_i, U_j \in \mathbb{C}^{N \times M}$ [7]. Note that this inner product acts on entire matrices, meaning that the global Lanczos method differs from other block Krylov methods in that it only orthogonalizes whole matrices in contrast to the individual columns therein. The algorithm iteratively builds up a basis $\mathbf{U}_i = [U_1, U_2, \ldots, U_i]$ of the Krylov space and yields the partial global Lanczos decomposition

$$A\mathbf{U}_i = \mathbf{U}_i \tilde{T}_i + \beta_{i+1} U_{i+1} E_i^T$$

where $\tilde{T}_i = T_i \otimes I_M \in \mathbb{R}^{iM \times iM}$ and $E_i^T = [\mathbf{0}, \cdots, \mathbf{0}, I_M] \in \mathbb{R}^{M \times iM}$. It was shown that the connection between the Lanczos algorithm and Gauss quadrature extends to the global Lanczos method. Hence, for an initial matrix $U \in \mathbb{C}^{N \times M}$ it in general holds

$$\mathsf{Tr} U^* f(A) U = \int f(\lambda) d\mu(\lambda) \approx \mathsf{Tr} f(A)$$

with μ being the distribution in the Riemann-Stieltjes integral generated by U and $\mathsf{Tr} U^* f(A) U$ yields a Gauss quadrature of $\mathsf{Tr} f(A)$. However, for the algorithm to remain computationally efficient or at least more efficient than computing the eigenvalue decomposition of A directly, it is required that $M \ll N$ and thus U can not be orthogonal/unitary. This implies that $U^* U \neq I$ and consequentially the method does in general not converge to the exact result so that sampling over multiple starting matrices is required if the approximation error is to be minimized. Additionally, for very large matrices even the computation of the aforementioned inner product becomes infeasible.

To allow for approximations of larger matrices, we reformulated the global Lanczos algorithm in terms of matrix product operators (MPO) which support all basic linear algebra operations. A matrix product operator decomposes a matrix $A \in \mathbb{C}^{N \times N}$ such that

$$A_{ij} = A_{i_1 \ldots i_L j_1 \ldots j_L} = \mathsf{Tr} C_1^{i_1 j_1} C_2^{i_2 j_2} \cdots C_L^{i_L j_L}$$

where the indices i, j are split up into $i_1, \ldots, i_L$ and $j_1, \ldots, j_L$ respectively and are called the physical indices. We here assume all physical indices to be of equal dimension $d = 2$ which corresponds to the assumption that $N = d^L$ but our results also carry over to the case of varying d_k. The $C_k \in \mathbb{C}^{D_k \times D_k \times d \times d}$ are called the core tensors where D_k is referred to as the bond dimension or auxiliary index. We additionally define the bond dimension of the MPO $D = \max_k D_k$ to be the maximal bond dimension over all core tensors. It follows that $C_k^{i_k, j_k}$ is a matrix of size at most $D \times D$ and the right-hand side of the above equation yields a scalar. The matrix product operator representation of a matrix requires Ld^2D^2 parameters which, depending on the choice of D, either poses an approximation or suffices for an exact representation. While naturally the accuracy of the approximation increases with growing D, it is commonly chosen such that $Ld^2D^2 \in \mathcal{O}(poly(L))$ and thus yields an efficient,i. e. polynomial in contrast to exponential in L, representation. It has been found that such choices of D often suffice for a good approximation. Note that especially in numerical quantum physics, it is possible and common to formulate matrices of interest, such as Hamiltonians, directly as matrix product operators and perform computations on them so that an explicitly stored matrix is at no point required. While explaining the decomposition in more detail exceeds the scope of this section, we refer the interested reader to the overview articles [49, 42, 24].

Our algorithm can thus be perceived as the global Lanczos algorithm reformulated for matrix product operators. However, the differences between the methods extend beyond the different possible sizes of the input matrices. In our method, we choose the identity matrix written as a matrix product operator as initial matrix U. This can be done exactly with a minimal bond dimension of $D = 1$. Thus, we are theoretically and practically able to start our computation with an orthogonal/unitary matrix of the same dimensionality as A which is not feasible in the original method as discussed above. Hence, instead of the previous equation

$$\mathsf{Tr} U^* f(A) U \approx \mathsf{Tr} f(A)$$

with $U \in \mathbb{C}^{N \times M}$ and $M \ll N$, in our method it holds

$$\mathsf{Tr} U^* f(A) U = \mathsf{Tr} f(A)$$

and $U \in \mathbb{C}^{N \times N}$ is orthogonal/unitary. Our algorithm consequentially computes a Krylov basis of $\mathrm{span}\{A^0, A^1, A^2, \cdots, A^{K-1}\}$ and the approximation error in $\mathsf{Tr} f(A)$ is controlled by the maximal Krylov dimension K and the maximally allowed bond dimension D_{max}. This implies that our algorithm produces an approximation of the exact result which can in principle be made arbitrarily accurate by increasing K and most importantly D_{max}. The method is shown in Algorithm 1. Note that the subfunctions `multiply` and `sum` involve solving an optimization problem to find a good representation of the result for a given bond dimension D. The respective optimization algorithms employ the sweeping scheme typical for tensor network optimizations where the individual core tensors are optimized sequentially in a dynamic programming fashion. In conclusion, our algorithm poses the first method to approximate functions of the form $\mathsf{Tr} f(A)$ of matrices of size significantly larger then 2^{20} and has no analytical lower bound on the approximation error.

3. Analytical Results. As in practice we must impose $D \in \mathcal{O}(poly(L))$ to remain computationally feasible, it is important to be able to detect when the approximations made lead to unreasonbly large errors in the computed basis matrices U_i and the projections T_i of A. This is especially important as it is clear that since the basis matrices are computed iteratively and depending on the previously computed ones, any error introduced in a given iteration will be propagted and influence all following iterations. Additionally, since the basic arithmetic operations are comparibly costly in the matrix product operator domain and can infuse errors into the computation, we would like to reduce their number whenever possible. In this section we will thus present some analytical results on properties of the basis matrices U_i and the projection T_i that can be checked for during a run of Algorithm 1 and that finally give rise to a more efficient version of our method for a special class of inputs. All following proofs and corollaries assume the matrix $A \in \mathbb{C}^{N \times N}$ to be Hermitian and the initial matrix U to be of the same dimensions as A. Note that we additionally assume exact arithmetics and exact representation of the U_i as our aim is to derive insight about the algorithm's ideal behaviour to be able to detect deviations from it. As we will make use of these equations in all following proofs of this section, it is worthwhile to explicitly state the update rules

$$U_{n+1} = \left(AU_n - \alpha_{n+1} U_n - \beta_n U_{n-1}\right) / \beta_{n+1}$$

and

$$\alpha_{n+1} = \mathsf{Tr} U_n^* U_{n+1}$$

as implied by Algoritm 1. Note also that our analysis is based on the Frobenius inner product that was, to the best of the author's knowledge, for the first time employed in the original global Lanczos algorithm [14, 7] and our consecutive work [2]. Combined with the fact that we assume A and U to be of equal dimension this shows that our results are complementary to other work on structured matrices in Krylov type algorithms [36, 17, 9, 8, 50, 37, 61]. Given these assumptions it is also clear that existing analyses of standard Lanczos type algorithms working on column vectors can not cover the following results.

We begin by stating a result about the inheritance of tracelessness of the basis matrices from the input A. Since it is possible to efficiently compute the trace of a given MPO, this property of the U_i can be efficiently checked for and, if desired, enforced during a run of the algorithm.

THEOREM 1. *If $A \in \mathbb{C}^{N\times N}$ is traceless and $U_0 \in \mathbb{C}^{N\times N} = I_N/\beta_0$, all basis matrices $U_i \in \mathbb{C}^{N\times N}, i \in \{1, \cdots, K\}$ as constructed by the algorithm are traceless.*

Proof. We prove the statement by induction over the iteration number n of the algorithm. For $n = 1$, it is easy to see that $\mathsf{Tr}U_1 = \mathsf{Tr}A/(\beta_1\beta_0) = 0$ as $\alpha_1 = \mathsf{Tr}A/\beta_0^2 = 0$. We now obtain for $n = 2$ that $\mathsf{Tr}U_2 = (\beta_1\beta_0\mathsf{Tr}U_1^*U_1 - \alpha_2/(\beta_1\beta_0)\mathsf{Tr}A - \beta_1/\beta_0\mathsf{Tr}I)/\beta_2 = 0$. This establishes the inductive basis.

In the inductive step for $n \geq 2$ we then have

$$\begin{aligned}\mathsf{Tr}U_{n+1} =&(\mathsf{Tr}AU_n - \alpha_{n+1}\mathsf{Tr}U_n - \beta_n\mathsf{Tr}U_{n-1})/\beta_{n+1}\\ =&(\mathsf{Tr}AU_n - \alpha_{n+1}0 - \beta_n 0)/\beta_{n+1}\\ =&0\end{aligned}$$

□

Next, we present a result about the commutation relation of A and the U_i that will also become useful for proving subsequent statements. As for the previous result, this property can be efficiently checked for during an execution of the algorithm assuming MPO representation of the U_i by computing the Frobenius norm, which is efficiently computable for MPOs, of the distance between AU_i and U_iA.

THEOREM 2. *If $A \in \mathbb{C}^{N\times N}$ commutes with $U_0 \in \mathbb{C}^{N\times N}$, A commutes with all basis matrices $U_i \in \mathbb{C}^{N\times N}, i \in \{1, \cdots, K\}$ as constructed by the algorithm.*

Proof. We again prove the statement by induction over the iteration number n. To start, we note that $[U_1, A] = ((AAU_0 - \alpha_1 AU_0) - (AAU_0 - \alpha_1 AU_0))/\beta_1 = 0$.

In the inductive step for $n \geq 1$, it is now straight forward to see that

$$\begin{aligned}[U_{n+1}, A] =& \left[(AU_nA - \alpha_{n+1}U_nA - \beta_nU_{n-1}A) - (AAU_n - \alpha_{n+1}AU_n - \beta_nAU_{n-1})\right]/\beta_{n+1}\\ =& \left[(AAU_n - \alpha_{n+1}AU_n - \beta_nAU_{n-1}) - (AAU_n - \alpha_{n+1}AU_n - \beta_nAU_{n-1})\right]/\beta_{n+1}\\ =&0\end{aligned}$$

□

COROLLARY 1. *We note that any $A \in \mathbb{C}^{N\times N}$ commutes with I_N. Thus it follows that the above statement holds for Algorithm 1.*

The following finding addresses the symmetry properties of the basis matrices in relation to the input A and the initial basis matrix U_0. These symmetry properties can as well be tested efficiently in a manner similar to the way the commutation relation can be checked since the permutation matrix J, like I, permits a formulation in MPO format with minimal bond dimension. Additionally, symmetries could be leveraged to obtain more efficient representations of the U_i by reflecting them in the structure of the MPOs and thus obtaining more

efficient and stable expressions.

THEOREM 3. *If $A \in \mathbb{R}^{N\times N}$ is symmetric, persymmetric or centrosymmetric and $U_0 \in \mathbb{R}^{N\times N}$ is symmetric, persymmetric or centrosymmetric and commutes with A, all basis matrices $U_i \in \mathbb{R}^{N\times N}, i \in \{1, \cdots, K\}$ as constructed by the algorithm are symmetric, persymmetric or centrosymmetric.*

Proof. As for the above statements, we prove this statement by induction over the iteration number n. To establish the inducte basis, we observe that for the case of symmetry $U_1^T = ((AU_0)^T - \alpha_1 U_0^T)/\beta_1 = (AU_0 - \alpha_1 U_0)/\beta_1 = U_1$. Likewise, we find that $U_1 J = (AU_0 - \alpha_1 U_0)/\beta_1 J = J((AU_0)^T - \alpha_1 U_0^T)/\beta_1 = JU_1^T$ for persymmetry and finally $JU_1 = J(AU_0 - \alpha_1 U_0)/\beta_1 = (AU_0 - \alpha_1 U_0)/\beta_1 J = U_1 J$ for centrosymmetry.

For $n \geq 1$, we can now make the inductive step by

$$\begin{aligned} U_{n+1}^T =& ((AU_n)^T - \alpha_{n+1} U_n^T - \beta_n U_{n-1}^T)/\beta_{n+1} \\ =& (U_n A - \alpha_{n+1} U_n - \beta_n U_{n-1})/\beta_{n+1} \\ =& U_{n+1} \end{aligned}$$

for symmetry,

$$\begin{aligned} U_{n+1} J =& (AU_n - \alpha_{n+1} U_n - \beta_n U_{n-1})/\beta_{n+1} J \\ =& (JA^T U_n^T - \alpha_{n+1} JU_n^T - \beta_n JU_{n-1}^T)/\beta_{n+1} \\ =& JU_{n+1}^T \end{aligned}$$

for persymmetry and

$$\begin{aligned} JU_{n+1} =& J(AU_n - \alpha_{n+1} U_n - \beta_n U_{n-1})/\beta_{n+1} \\ =& (AU_n J - \alpha_{n+1} U_n J - \beta_n U_{n-1} J)/\beta_{n+1} \\ =& U_{n+1} J \end{aligned}$$

for centrosymmetry. □

COROLLARY 2. *As can be easily verified based on its proof, the above statement extends to the case of hermiticity, perhermiticity and centrohermiticity when $A, U_i \in \mathbb{C}^{N\times N}, i \in \{0, \cdots, K\}$.*

COROLLARY 3. *We note that the matrix I_N is symmetric, persymmetric and centrosymmetric as well as hermitian, perhermitian and centrohermitian. Hence the above statements hold for Algorithm 1.*

We now turn our attention to a description of the U_i in terms of polynomials as might seem natural given the underlying Lanczos algorithm. However, we restrict our analysis to the particular case where all $\alpha_i = 0$ to obtain a result that will become important in the proof of the subsequent statement.

THEOREM 4. *If for $A \in \mathbb{C}^{N\times N}$ all $\alpha_i = 0, i \in \{1, \cdots, K\}$ as computed by the algorithm and $U_0 \in \mathbb{C}^{N\times N} = I_N/\beta_0$, then all $U_i \in \mathbb{C}^{N\times N}, i \in \{1, \ldots, K\}$ are polynomials of the form $\sum_{j\in 2\mathbb{N}_0 \leq i} c_j A^j$ if i is even and $\sum_{j\in 2\mathbb{N}-1 \leq i} c_j A^j$ if i is odd.*

Proof. We again prove the statement by induction over the iteration number n. We establish the inductive basis by observing that $U_0 = A^0/\beta_0, U_1 = A/(\beta_1\beta_0), U_2 = A^2/(\beta_2\beta_1\beta_0) - \beta_1/(\beta_2\beta_0)A^0$ and $U_3 = A^3/(\beta_3\beta_2\beta_1\beta_0) - (\beta_2^2 + \beta_1^2)/(\beta_3\beta_2\beta_1\beta_0)A$ are all polynomials of the types specified above.

In the inductive step, we then find for even n that

$$\begin{aligned}
U_{n+1} =&(AU_n - \beta_n U_{n-1})/\beta_{n+1} \\
=&\left(A \sum_{j\in 2\mathbb{N}_0 \leq n} c_j A^j - \beta_n \sum_{j\in 2\mathbb{N}-1 \leq n-1} d_j A^j\right)/\beta_{n+1} \\
=&\left(\sum_{j\in 2\mathbb{N}-1 \leq n+1} c_j A^j - \beta_n \sum_{j\in 2\mathbb{N}-1 \leq n-1} d_j A^j\right)/\beta_{n+1} \\
=&\sum_{j\in 2\mathbb{N}-1 \leq n+1} (c_j - \beta_n d_j)/\beta_{n+1} A^j
\end{aligned}$$

and analogously for odd n

$$\begin{aligned}
U_{n+1} =&(AU_n - \beta_n U_{n-1})/\beta_{n+1} \\
=&\left(A \sum_{j\in 2\mathbb{N}-1 \leq n} c_j A^j - \beta_n \sum_{j\in 2\mathbb{N}_0 \leq n-1} d_j A^j\right)/\beta_{n+1} \\
=&\left(\sum_{j\in 2\mathbb{N}_0 \leq n+1} c_j A^j - \beta_n \sum_{j\in 2\mathbb{N}_0 \leq n-1} d_j A^j\right)/\beta_{n+1} \\
=&\sum_{j\in 2\mathbb{N}_0 \leq n+1} (c_j - \beta_n d_j)/\beta_{n+1} A^j
\end{aligned}$$

where we defined $d_{n+1} \coloneqq 0$. □

We can now use this result to obtain a more profound insight.

THEOREM 5. *If $A \in \mathbb{C}^{N\times N}$ has a spectrum that is point-wise symmetric around zero and $U_0 \in \mathbb{C}^{N\times N} = I_N/\beta_0$, all the $\alpha_i, i \in \{1, \cdots, K\}$ as computed by the algorithm are zero.*

Proof. As done previously, we prove this statement by induction over the iteration number n. We start by observing that by assumption $\mathsf{Tr}A = 0$ and hence $\alpha_1 = \mathsf{Tr}A/\beta_0^2 = 0$. Consequentially, we have that $\alpha_2 = \mathsf{Tr}(AU_1)^* U_1 = 1/(\beta_1\beta_0)^2 \mathsf{Tr}A^3 = 0$ which is our inductive basis.

Now for the inductive step, we begin by noting that it follows from the inductive hypothesis that all $U_i, i \in \{1, \cdots, n\}$ are polynomials of the form defined in the previous statement. Then for odd n, it follows that

$$\begin{aligned}
\alpha_{n+1} =&\mathsf{Tr}U_n U_n^* A \\
=&\mathsf{Tr}\left(\sum_{j\in 2\mathbb{N}-1 \leq n} c_j A^j\right)\left(\sum_{j\in 2\mathbb{N}-1 \leq n} c_j A^{*^j}\right) A \\
=&\sum_{j\in 2\mathbb{N}+1 \leq 2n+1} c_j \mathsf{Tr}A^j \\
=&0.
\end{aligned}$$

Analogously, it follows for even n that

$$
\begin{aligned}
\alpha_{n+1} =& \mathsf{Tr} U_n U_n^* A \\
=& \mathsf{Tr} \left(\sum_{j \in 2\mathbb{N}_0 \leq n} c_j A^j \right) \left(\sum_{j \in 2\mathbb{N}_0 \leq n} c_j A^{*^j} \right) A \\
=& \sum_{j \in 2\mathbb{N}-1 \leq 2n+1} c_j \mathsf{Tr} A^j \\
=& 0.
\end{aligned}
$$

□

From this finding, we finally obtain the following corollary.

COROLLARY 4. *For $A \in \mathbb{C}^{N \times N}$ having a spectrum point-wise symmetric around zero, Algorithm 1 produces a bidiagonal matrix*

$$
T_K = \begin{bmatrix} 0 & \beta_1 & & \mathbf{0} \\ \beta_1 & 0 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & 0 \end{bmatrix}.
$$

This insight yields on one hand a more efficient version of the algorithm as each U_i is in theory guaranteed to be orthogonal to U_{i-1} and hence only one orthogonalization has to be performed in each iteration of the algorithm. Since orthogonalizations of MPOs require solving an optimization problem and are hence significantly more computationally demanding than the orthogonalization of full matrices, avoiding them results in a measurable reduction of the runtime as we will illustrate later. On the other hand, we obtain yet another means of checking for the effect of truncation errors by monitoring the magnitude of the α_i when it is known they must be zero. It is of course also possible to still orthogonalize against the previous two basis MPOs but always set $\alpha_i = 0$ to increase the approximations accuracy. However, it is worth noting that the condition of the α_i being equal to zero is necessary but not sufficient for the overall approximation of $\mathsf{Tr} f(A)$ to be accurate. The deviation from zero of the α_i does not allow us to draw strong conclusions about the accuracy of the approximation of $\mathsf{Tr} f(A)$. To illustrate this point, we add a few remarks.

- Although it seems reasonable to assume that when the $\|\alpha_i\|$ remain small the approximated values of β_i are also close to their true values, we have no way of inferring the error in the β_i from the error in the α_i. This is mainly the case because we do not have access to the true values of the β_i and we believe it not to be possible to establish an analytical practically relevant connection between both errors in our algorithm, especially when truncations come into play.
- If however we find some α_i to be significantly larger in magnitude than zero, we know that the respective two basis MPOs are not orthogonal as they should be which usually leads to the reocurrence of previously observed approximated eigenvalues. Although we know that the accuracy of the overall approximation will suffer from this, it is unfortunately not possible to make a more precise statement as we cannot tell in detail how a deviation from zero relates to amount and magnitude of such 'ghost' eigenvalues and again we do not know the error in the β_i.
- Although in principle one could counter growing magnitudes of the α_i by increasing D of the basis MPOs and (re)orthogonalizing, in practice the bond dimension of the basis MPOs reaches D_{max} after already a few iterations and by definition we

cannot exceed this value. Still, it would be possible to either restart a failed run with a larger maximal bond dimension or increase it dynamically until the α_i become small enough. The latter approach however could be argued to defeat the purpose of the D_{max} parameter.

4. Spectra of Hamiltonians. The results obtained in the previous section naturally raise the question what kinds of matrices exhibit a spectrum point wise symmetric around zero and how many cases of relevance there are. While we cannot give a general answer to this question, we can provide a partial answer for a specific application, namely spin systems in quantum physics. These systems are often studied analytically and numerically because they exhibit interesting physical phenomena while still allowing for the derivation of mathematically rigorous results and comparably efficient simulations by tensor network approaches.

Spin systems are described by their corresponding Hamiltonians which for open boundaries and interactions between direct neighbours take the form

$$H_{OBC} = \sum_{(i,\alpha)\in\mathcal{I}} \sum_{j=0}^{L-i} h_{ij\alpha} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j}$$

where $L \in \mathbb{N}$ is the number of spin particles and $\mathcal{I}$ is a set of tuples $(i, \alpha) \in \mathbb{N}_L \times \{x, y, z\}$ denoting the number of consecutive applications of σ_α. In this case, $\sigma_{x,y,z}$ denote the Pauli matrices

$$\sigma_x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

and the $h_{ij\alpha} \in \mathbb{R}$ simply are scaling constants. Similarly, the case of closed or periodic boundaries is expressed as

$$H_{PBC} = \sum_{(i,\alpha)\in\mathcal{I}} \sum_{j=0}^{L-i} h_{ij\alpha} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{(i,\alpha)\in\mathcal{I}} \sum_{k=1}^{i-1} h_{ik\alpha} \sigma_\alpha^{\otimes k} \otimes I^{\otimes L-i} \otimes \sigma_\alpha^{\otimes i-k}.$$

We will in the following denote the individual terms in the sums of the Hamiltonians as interaction terms and refer to the products of multiple Pauli matrices inside these interaction terms as blocks. This terminology is derived from the fact that each term describes the interaction between the particles at whose position there is a Pauli operator in the product. While the formulations introduced above naturally do not describe all possible Hamiltonians, they cover many interesting cases which are furthermore treatable via tensor network methods. This typically gets much more difficult for cases of arbitrary and long-range interaction patterns, which are not covered by the above expressions.

Now, one sufficient condition for the existence of a point-wise symmetric spectrum around zero looks as follows: for a given $H \in \mathbb{C}^{2^L \times 2^L}$, there exists a unitary and Hermitian matrix R of equal size, such that

$$RH = -HR$$

and consequentially by the standard eigenvalue formulation $Hv = \lambda v$ it holds that

$$H(Rv) = -\lambda(Rv).$$

In the following, we will make statements about the existence of such an R for several classes of spin Hamiltonians. Although absence of such an R does not imply that the Hamiltonian

in question does not exhibit a point symmetric spectrum around zero, the above formulation captures a large class of possible symmetries and thus poses a relevant albeit not final characterization of point symmetric spectra. Note that in quantum physics it is already known that one can make use of the rotation transformation properties of spin operators to change the sign of particular terms in a Hamiltonian [48]. However, here we tackle the problem of changing all terms, from now on also called interaction terms, in a Hamiltonian to relate different eigenvalues/-states to each other and our focus lies on formally defining classes of spin Hamiltonians for which such an R exists and which hence are valid inputs for our improved algorithm. A related mathematical discussion for general square matrices was presented by Fassbender et. al. [17].

Before we start, we remind ourselves that the Pauli matrices are Hermitian and unitary and that each pair of Pauli matrices anticommutes such that for $\alpha, \beta \in \{x, y, z\}$ it holds

$$\{\sigma_\alpha, \sigma_\beta\} = 2\delta_{\alpha,\beta} I.$$

Furthermore, we note that the Kronecker product of Hermitian and unitary matrices is again Hermitian and unitary. These properties will be used in all following proofs.

We start by considering Hamiltonians with open boundaries and neighbour interactions of arbitrary length for a single Pauli operator.

THEOREM 6. *For every spin Hamiltonian with open boundaries of the form*

$$H_{OBC,\alpha,i} = \sum_{j=0}^{L-i} h_j I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j}$$

where $\alpha \in \{x, y, z\}$ and $i, L \in \mathbb{N}$ and $i \leq L$, there exists a unitary $R \in \mathbb{C}^{2^L \times 2^L}$ such that $RH_{OBC,\alpha,i} = -H_{OBC,\alpha,i}R$.

Proof. We can construct $R = \left(I^{\otimes i-1} \otimes \sigma_{\alpha'}\right)^{\otimes L/i} \otimes I^{\otimes L\%i}$ with $\alpha' \neq \alpha$ as we need only apply one $\sigma_{\alpha'}$ for each block $\sigma_\alpha^{\otimes i}$ to change the sign in every term of the sum in $H_{\alpha,i}$ and thereby ultimately the sign of $H_{\alpha,i}$ itself. Hereby, $\otimes^{L/i}$ denotes the repition of the given expression for L/i times whereas $I^{\otimes L\%i}$ simply refers to a 'padding' of R to the required length L. □

While for the case of open boundaries and a single Pauli matrix the statement is quite universal, the additional structure introduced by periodic boundaries forces us to restrict the statement to odd interactions lengths.

THEOREM 7. *For every spin Hamiltonian with periodic boundaries of the form*

$$H_{PBC,\alpha,i} = \sum_{j=0}^{L-i} h_j I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{k=1}^{i-1} h_k \sigma_\alpha^{\otimes k} \otimes I^{\otimes L-i} \otimes \sigma_\alpha^{i-k}$$

where $\alpha \in \{x, y, z\}$ and $i \in 2\mathbb{N} - 1$, $L \in \mathbb{N}$ and $i \leq L$, there exists a unitary $R \in \mathbb{C}^{2^L \times 2^L}$ such that $RH_{PBC,\alpha,i} = -H_{PBC,\alpha,i}R$.

Proof. By defining $R = \sigma_{\alpha'}^{\otimes L}$ with $\alpha' \neq \alpha$ we obtain an odd number of sign changes in every term of the sum in $H_{PBC,\alpha,i}$ inducing a sign change of $H_{PBC,\alpha,i}$. □

These two statements together show that a large subset of spin Hamiltonians with interaction terms involving only one particular Pauli matrix exhibit a point symmetric spectrum around zero. Not surprisingly, the situation becomes more involved when considering Hamiltonians with up to two different Pauli operators and differing interaction lengths. We first

examine the case of interaction terms involving two differing Pauli operators.

THEOREM 8. *For every spin Hamiltonian with open boundaries of the form*

$$H_{OBC,\alpha,\beta,i,k} = \sum_{j=0}^{L-i} h_{\alpha j} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{l=0}^{L-k} h_{\beta l} I^{\otimes l} \otimes \sigma_\beta^{\otimes k} \otimes I^{\otimes L-k-l}$$

where $\alpha, \beta \in \{x, y, z\}$, $\alpha \neq \beta$, $i, k, L \in \mathbb{N}$ *and* $i, k \leq L$, *there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{OBC,\alpha,\beta,i,k} = -H_{OBC,\alpha,\beta,i,k}R$.

Proof. We have to distinguish two cases regarding the relation of i and k.

Case $i = k$

In this case there is a $\gamma \in \{x, y, z\} \setminus \{\alpha, \beta\}$ such that $\sigma_\alpha \sigma_\gamma = -\sigma_\gamma \sigma_\alpha$ and $\sigma_\beta \sigma_\gamma = -\sigma_\gamma \sigma_\beta$. Hence, we can again define $R = \left(I^{\otimes i-1} \otimes \sigma_\gamma\right)^{\otimes L/i} \otimes I^{\otimes L\%i}$ to obtain a unitary that induces a sign change in every block $\sigma_\alpha^{\otimes i}$ and $\sigma_\beta^{\otimes k}$ and hence changes the sign of $H_{OBC,\alpha,\beta,i,k}$.

Case $i \neq k$

Let w.l.o.g. $i > k$. Then we can construct R as the Kronecker product of L matrices such that at every k-th position we apply σ_α and at every i-th position we apply σ_β. In the case where multiples of i and k coincide, we again choose $\gamma \in \{x, y, z\} \setminus \{\alpha, \beta\}$ and apply it in these positions. The remaining free factors are again chosen to be the identity. It is evident from the construction of R that it induces exactly one sign change in every block $\sigma_\alpha^{\otimes i}$ and $\sigma_\beta^{\otimes k}$ respectively.

□

A different situtation presents itself when we again restrict the interaction terms in the Hamiltonian to involve only one Pauli operator but allow two different interaction lengths.

THEOREM 9. *For every spin Hamiltonian with open boundaries of the form*

$$H_{OBC,\alpha,i,k} = \sum_{j=0}^{L-i} h_{ij} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{l=0}^{L-k} h_{kl} I^{\otimes l} \otimes \sigma_\alpha^{\otimes k} \otimes I^{\otimes L-k-l}$$

where $\alpha \in \{x, y, z\}$, $i, k \in 2\mathbb{N} - 1$, $L \in \mathbb{N}$, *and* $i, k \leq L$, *there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{OBC,\alpha,i,k} = -H_{OBC,\alpha,i,k}R$.

Proof. We again have to distinguish two cases regarding the relation of i and k.

Case $i = k$

In this case, the Hamiltonian is a member of the class considered in Theorem 6.

Case $i \neq k$

In this case, we can again choose an $\alpha' \neq \alpha$ and define $R = \sigma_{\alpha'}^{\otimes L}$. R then induces an odd number of sign changes in every term of $H_{OBC,\alpha,i,k}$ and consequentially a sign change in the whole Hamiltonian.

□

This result can now easily be generalized to the case of more than two interaction lengths for a fixed Pauli operator.

COROLLARY 5. *By a straight forward generalization of the above proof we obtain that for all Hamiltonians with open boundaries and one Pauli operator of the form*

$$H_{PBC,\alpha} = \sum_{i \in \mathcal{I}} \sum_{j=0}^{L-i} h_{ij} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j}$$

where $\alpha \in \{x, y, z\}$ *and* $\mathcal{I} \subset 2\mathbb{N} - 1$, *there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{PBC,\alpha} = -H_{PBC,\alpha}R$.

While we restricted the interaction lengths to be odd for the statements above, we find that there exists another case for arbitrary interaction lengths with a certain relation between them.

THEOREM 10. *For every spin Hamiltonian with open boundaries of the form*

$$H_{OBC,\alpha,i,k} = \sum_{j=0}^{L-i} h_{ij} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{l=0}^{L-k} h_{kl} I^{\otimes l} \otimes \sigma_\alpha^{\otimes k} \otimes I^{\otimes L-k-l}$$

where $\alpha \in \{x, y, z\}$,$i, k \in \mathbb{N}, i/k \in 2\mathbb{N}-1, L \in \mathbb{N}$, *and* $i, k \leq L$, *there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{OBC,\alpha,i,k} = -H_{OBC,\alpha,i,k}R$.

Proof. Also here, we have to distinguish two cases regarding the relation of i and k.

Case $i = k$

In this case, the Hamiltonian is a member of the class considered in Theorem 6.

Case $i \neq k$

We can construct $R = \left(I^{\otimes k-1} \otimes \sigma_{\alpha'}\right)^{\otimes L/k} \otimes I^{\otimes L\%k}$ with $\alpha' \neq \alpha$. Since $i/k \in 2\mathbb{N}-1$ we find that R induces an odd number of sign changes in all terms of $H_{OBC,\alpha,i,k}$ and thus in the overall Hamiltonian. □

What is now left to discuss for Hamiltonians involving up to two different Pauli operators is the case of periodic boundaries, which again introduces more constraints. Hence we find that we can only make a positive statement about odd interaction lengths as follows.

THEOREM 11. *For every spin Hamiltonian with periodic boundaries of the form*

$$\begin{aligned} H_{PBC,\alpha,\beta,i,l} = &\sum_{j=0}^{L-i} h_{\alpha j} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{k=1}^{i-1} h_{\alpha k} \sigma_\alpha^{\otimes k} \otimes I^{\otimes L-i} \otimes \sigma_\alpha^{i-k} \\ &+ \sum_{m=0}^{L-l} h_{\beta m} I^{\otimes m} \otimes \sigma_\beta^{\otimes l} \otimes I^{\otimes L-l-m} + \sum_{n=1}^{l-1} h_{\beta n} \sigma_\beta^{\otimes n} \otimes I^{\otimes L-l} \otimes \sigma_\beta^{l-n} \end{aligned}$$

where $\alpha, \beta \in \{x, y, z\}$, $i, l \in 2\mathbb{N}-1$, $L \in \mathbb{N}$ *and* $i, l \leq L$, *there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{PBC,\alpha,i} = -H_{PBC,\alpha,i}R$.

Proof. As before we can choose $\gamma \in \{x, y, z\} \setminus \{\alpha, \beta\}$ and define $R = \sigma_\gamma^{\otimes L}$. Since $i, l \in 2\mathbb{N}-1$, R induces and odd number of sign changes in ever term of and consequentially in $H_{PBC,\alpha,\beta,i,l}$. □

This statement can again be readily generalized to multiple interaction lengths and hence more complex Hamiltonians.

COROLLARY 6. *By a straight forward generalization of the above proof we obtain that for all Hamiltonians with periodic boundaries and at most two different Pauli operators of the form*

$$\begin{aligned} H_{PBC,\alpha,\beta} = &\sum_{i\in\mathcal{I}} \sum_{j=0}^{L-i} h_{ij\alpha} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{i\in\mathcal{I}} \sum_{k=1}^{i-1} h_{ik\alpha} \sigma_\alpha^{\otimes k} \otimes I^{\otimes L-i} \otimes \sigma_\alpha^{\otimes i-k} \\ &+ \sum_{l\in\mathcal{J}} \sum_{j=0}^{L-l} h_{lj\beta} I^{\otimes j} \otimes \sigma_\beta^{\otimes l} \otimes I^{\otimes L-l-j} + \sum_{l\in\mathcal{J}} \sum_{k=1}^{l-1} h_{lk\beta} \sigma_\beta^{\otimes k} \otimes I^{\otimes L-l} \otimes \sigma_\beta^{\otimes l-k}. \end{aligned}$$

where $\alpha, \beta \in \{x, y, z\}$ *and* $\mathcal{I}, \mathcal{J} \subset 2\mathbb{N}-1$, *there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{PBC,\alpha,\beta} = -H_{PBC,\alpha,\beta}R$.

Now, we finally come to the case of Hamiltonians consisting of interaction terms generated by up to three Pauli operators which is clearly the most complicated setting. We begin by inspecting Hamiltonians with open boundaries involving all three Pauli matrices.

THEOREM 12. *For every spin Hamiltonian with open boundaries of the form*

$$H_{OBC,\alpha,\beta,\gamma,i,k,m} = \sum_{j=0}^{L-i} h_{\alpha j} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{l=0}^{L-k} h_{\beta l} I^{\otimes l} \otimes \sigma_\beta^{\otimes k} \otimes I^{\otimes L-k-l} + \sum_{n=0}^{L-m} h_{\gamma n} I^{\otimes n} \otimes \sigma_\gamma^{\otimes m} \otimes I^{\otimes L-m-n}$$

where $\alpha, \beta, \gamma \in \{x, y, z\}, \alpha \neq \beta \neq \gamma \neq \alpha, k, L \in \mathbb{N}, i, m \in 2\mathbb{N} - 1, i < k$ *and* $i, k, m \leq L$*, there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{OBC,\alpha,\beta,\gamma,i,k,m} = -H_{OBC,\alpha,\beta,\gamma,i,k,m}R$*.*

Proof. We define $R = \left(\sigma_\beta^{k-1} \otimes \sigma_\alpha\right)^{\otimes L/k} \otimes \sigma_\beta^{\otimes L\%k}$. It is clear that R induces an odd number of sign changes in all blocks $\sigma_\alpha^{\otimes i}$ since $i \leq k-1$ is odd. Similarly, it is obvious that R causes exactly one sign change in every block $\sigma_\beta^{\otimes k}$ through the single σ_α in the product. As both σ_α and σ_β cause sign changes in the blocks $\sigma_\gamma^{\otimes m}$ and m is odd, it is evident that R also induces and odd number of sign changes in this case. Hence it holds that $RH_{OBC,\alpha,\beta,\gamma,i,k,m} = -H_{OBC,\alpha,\beta,\gamma,i,k,m}R$.
□

Finally, we examine the case of two Pauli matrices and three different interaction lengths for open boundary conditions.

THEOREM 13. *For every spin Hamiltonian with open boundaries of the form*

$$H_{OBC,\alpha,\beta,i,k,m} = \sum_{j=0}^{L-i} h_{\alpha j} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{l=0}^{L-k} h_{\beta kl} I^{\otimes l} \otimes \sigma_\beta^{\otimes k} \otimes I^{\otimes L-k-l} + \sum_{n=0}^{L-m} h_{\beta mn} I^{\otimes n} \otimes \sigma_\beta^{\otimes m} \otimes I^{\otimes L-m-n}$$

where $\alpha, \beta \in \{x, y, z\}, \alpha \neq \beta, L \in \mathbb{N}, i, k, m \in 2\mathbb{N} - 1, i < k$ *and* $i, k, m \leq L$*, there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{OBC,\alpha,\beta,i,k,m} = -H_{OBC,\alpha,\beta,i,k,m}R$*.*

Proof. As before we can choose $\gamma \in \{x, y, z\} \setminus \{\alpha, \beta\}$ and define $R = \sigma_\gamma^{\otimes L}$. Since $i, k, l \in 2\mathbb{N} - 1$, R induces and odd number of sign changes in ever term of and consequentially in $H_{PBC,\alpha,\beta,i,l}$.
□

This statement can now again be generalized to multiple interaction terms.

COROLLARY 7. *Again by a straight forward generalization of the above proof we find that for all Hamiltonians with open boundaries and two Pauli operators of the form*

$$H_{PBC,\alpha,\beta} = \sum_{i \in \mathcal{I}} \sum_{j=0}^{L-i} h_{ij\alpha} I^{\otimes j} \otimes \sigma_\alpha^{\otimes i} \otimes I^{\otimes L-i-j} + \sum_{k \in \mathcal{J}} \sum_{j=0}^{L-k} h_{kj\beta} I^{\otimes j} \otimes \sigma_\beta^{\otimes k} \otimes I^{\otimes L-k-j}$$

where $\alpha, \beta \in \{x, y, z\}$ *and* $\mathcal{I} \subset 2\mathbb{N} - 1$*, there exists a unitary* $R \in \mathbb{C}^{2^L \times 2^L}$ *such that* $RH_{PBC,\alpha} = -H_{PBC,\alpha}R$*.*

To the best of our knowledge, we cannot make a positive statement for periodic boundaries and interaction terms involving all three Pauli operators. As a remark, we would like to point out that in addition to the Hamiltonians treated in this section, positive statements about

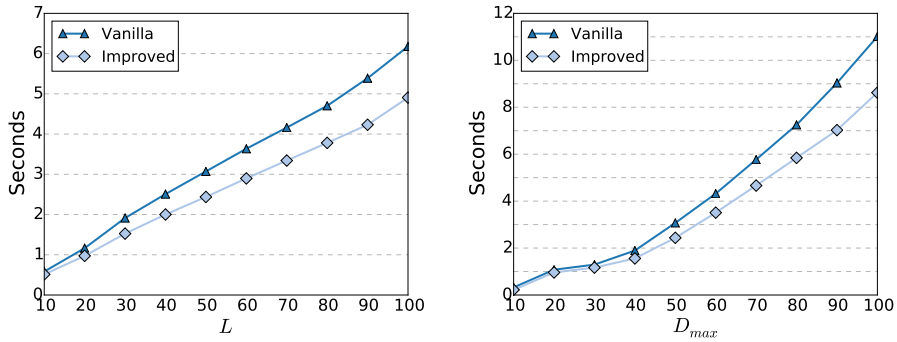


FIG. 5.1. *Comparisons of the runtime in seconds between the improved and the vanilla version of the algorithm. Left: Comparison of average runtime of one iteration over L with $D_{max} = 50$. Right: Comparison of average runtime of one iteration over D_{max} with $L = 50$.*

the existence of an R as considered here should be easy to proof in a very similar way for arbitrary interactions, i.e. interactions not between nearest neightbours but arbitrary particles, and odd numbers of particles affected by the interaction terms. Furthermore, as a special case of the Hamiltonian matrices discussed by Fassbender et. al. [17] symmetric two-by-two block matrices of the form

$$\begin{bmatrix} B & C \\ C & -B \end{bmatrix}$$

with $B, C \in \mathbb{R}^{N \times N}$ and B, C symmetric generally exhibit a spectrum symmetric around zero and can thus be considered valid inputs to the presented variant of our method. This of course is subject to the condition that they yield a sufficiently accurate and small MPO representation. As a final remark, we note that positive definitive Bethe-Salpeter Hamiltonian matrices in principle also pose a valid input to the algorithm [8, 51].

We have shown in this section that a significant subset of all spin Hamiltonians exhibits a point symmetric spectrum around zero according to the introduced characterization and that consequentially there exists a strong use case of our improvement of Algorithm 1 in quantum mechanical simulations. In the next section, we will now use a well known Hamiltonian belonging to this subset to numerically illustrate the advantage of the improved algorithm in this case.

5. Numerical Evidence. To provide numerical evidence of the correctness of our statements in Sections 3 and 4, we will now state results obtained by conducting some numerical experiments for the well known Ising Hamiltonian with a transverse field. The Hamiltonian is given by

$$H = J \sum_{i=1}^{L-1} I^{\otimes i-1} \otimes \sigma_x \otimes \sigma_x \otimes I^{\otimes L-(i+1)} + g \sum_{i=1}^{L} I^{\otimes i-1} \otimes \sigma_z \otimes I^{\otimes L-i}$$

where $\sigma_{x,z}$ are again the Pauli matrices. As we have seen in Section 4, the transverse field Ising Hamiltonian clearly has the spectral property required to apply the improved version of the algorithm. It also has the additional advantage that it can be diagonalized analytically to obtain reference results. Given a Hamiltonian, its thermal equilibrium, or Gibbs, state is

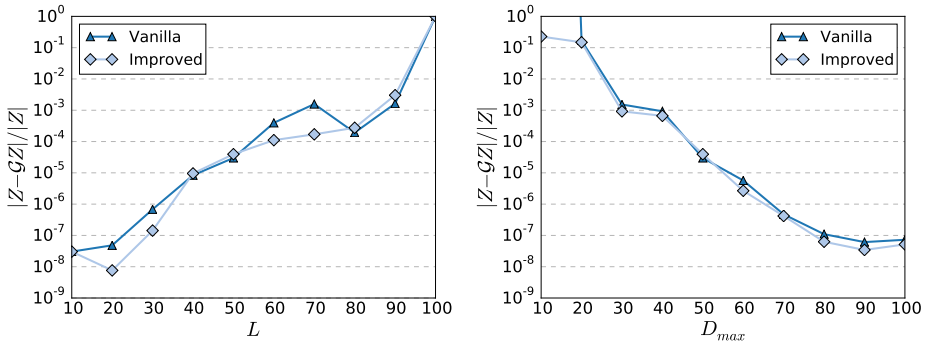


FIG. 5.2. *Comparisons of the relative error in Z between the improved and the vanilla version of the algorithm. Left: Comparison of the relative error over L with $D_{max} = 50$. Right: Comparison over the relative error over D_{max} and $L = 50$.*

described by

$$\rho(\beta) = \frac{e^{-\beta H}}{Z}$$

where β is the inverse temperature and

$$Z = \mathsf{Tr} e^{-\beta H}$$

is the so called partition function or simply the normalization constant of the distribution. As our goal in this section is to compare both versions of the algorithm and not to provide physically relevant results, we will simply approximate Z by chosing

$$f(H) = e^{-\beta H}$$

where we set the scaling coefficients of the Hamiltonian and the inverse temperature to $J = g = \beta = 1$.

To provide a thorough comparison between the vanilla, i.e. standard, and the improved version of our algorithm in terms of runtime and accuracy, we have conducted two sets of experiments. Firstly, we fixed the maximal bond dimension to be $D_{max} = 50$ and computed the average runtime for one iteration of the algorithm over a run of 50 iterations for L, i.e., the system size, increasing from 10 to 100. Secondly, we set $L = 50$ and increased D_{max} from 10 to 100 and again computed the average runtime of one iteration over a run of 50 iterations. The comparison of the runtimes is depicted in Figure 5.1.

For both of these settings, we also evaluated the approximation accuracy as the relative error in Z when we let the algorithm run until the relative difference between approximations results became smaller than 10^{-6}. These results are illustrated in Figure 5.2. All results reported here were obtained for a C++ implementation of our algorithm on an Intel i5-5200U mobile CPU.

The results in Figure 5.1 clearly show an advantage in runtime for the improved version of the algorith for all considered settings. On average over all conducted experiments this advantage is around 20%, which seems like only a modest improvement but can easily amount to several hours of runtime less for large systems and large values of D_{max}. The results additionally illustrate the linear complexity in L and cubic dependence on D_{max} we have claimed in [2] and which is not affected by the improvement introduced in this work.

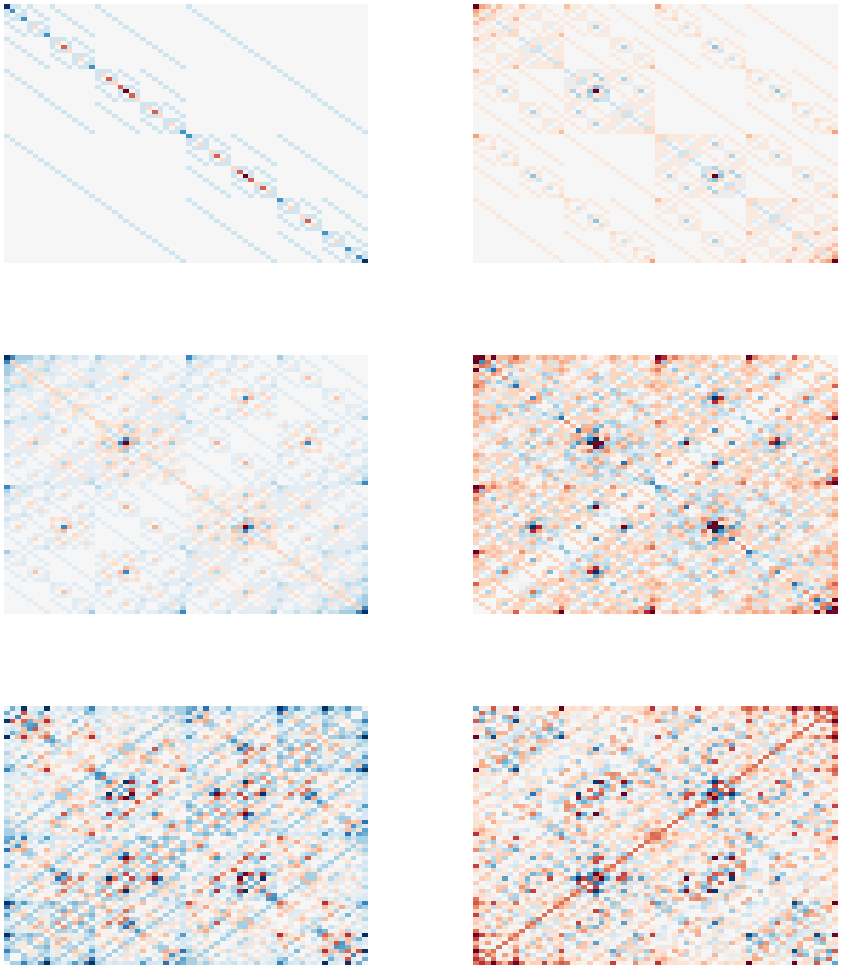


FIG. 5.3. *Heatmaps of the first six basis matrices as computed by the algorithm without approximations for the transverse Ising Hamiltonian with* $L = 10$.

In Figure 5.2 we can furthermore observe that for the case of an input that exhibits the required spectral symmetry the accuracy of both versions of the algorithm is similar with slight advantages for the improved variant. This might be due to the fact that the unnecessarily computed partial results in the vanilla version of the algorithm are not exactly zero and hence introduce a small amount of additional error into the approximation.

Finally, in Figure 5.3 we show heatmaps of the first six computed basis matrices in a run of the algorithm without approximations for $L = 10$. While the first basis matrix is simply the scaled transverse field Ising Hamiltonian, the following matrices represent its orthogonalized powers. Although this naturally does not constitute a rigorous argument, we can find by simple visual inspection that the basis matrices inherit the symmetric properties of the Hamiltonian, providing some intuition for the statements made in Section 3.

6. Conclusion. In this work we have tried to shed some more light on the analytic properties of the matrix product function approximation algorithm by analyzing the characteristics of the partial results computed during a full run. As a result, we have found that the basis matrices as computed by the algorithm inherit a range of properties from the input matrix. We have also seen that these properties then yield a more efficient version of the algorithm for a particular kind of input class, namely the class of matrices with point symmetric spectrum around zero. We then went on to show for the application of quantum physics that a variety of spin Hamiltonians exhibits this spectral symmetry property and that hence in this field of application the discovered improvement can be successfully applied in many cases. Finally, we demonstrated and verified our findings in numerical experiments conducted for the example of the Ising Hamiltonian with a transverse magnetic field.

While we were able to improve our understanding of the algorithm, more remains to be done, especially with respect to our understanding of the influence of the introduced truncation errors on the overall approximation accuracy. In addition to his, it would be interesting to see further applications of the algorithm outside of numerical quantum physics. Another possible route of further research would be the exploration of possible combinations of our algorithm with other methods that approximate single extremal eigenvalues. The approximations of extremal eigenvalues could be used to improve the accuracy of the approximation of the entire spectrum as computed by our algorithm.

7. Acknowledgements. We would like to thank the Elite Networks of Bavaria for partly funding this work via the doctoral programme *Exploring Quantum Matter (ExQM)*. We would also like to thank Mari Carmen Bañuls for being available for helpful discussions.

REFERENCES

[1] W. ARNOLDI, *The principle of minimized iterations in the solution of the matrix eigenvalue problem*, Quarterly of applied mathematics, 9 (1951), pp. 17–29.

[2] M. AUGUST, M. C. BAÑULS, AND T. HUCKLE, *On the approximation of functionals of very large hermitian matrices represented as matrix product operators*, Electronic Transactions on Numerical Analysis, 46 (2017), pp. 215–232.

[3] M. BACHMAYR, R. SCHNEIDER, AND A. USCHMAJEW, *Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations*, Foundations of Computational Mathematics, (2016), pp. 1–50.

[4] J. BAGLAMA, C. FENU, L. REICHEL, AND G. RODRIGUEZ, *Analysis of directed networks via partial singular value decomposition and gauss quadrature*, Linear Algebra and its Applications, 456 (2014), pp. 93–121.

[5] Z. BAI, G. FAHEY, AND G. GOLUB, *Some large-scale matrix computation problems*, Journal of Computational and Applied Mathematics, 74 (1996), pp. 71–89.

[6] C. BEKAS, E. KOKIOPOULOU, AND Y. SAAD, *An estimator for the diagonal of a matrix*, Applied numerical mathematics, 57 (2007), pp. 1214–1229.

[7] M. BELLALIJ, L. REICHEL, G. RODRIGUEZ, AND H. SADOK, *Bounding matrix functionals via partial global block lanczos decomposition*, Applied Numerical Mathematics, 94 (2015), pp. 127–139.

[8] P. BENNER, S. DOLGOV, V. KHOROMSKAIA, AND B. N. KHOROMSKIJ, *Fast iterative solution of the bethe–salpeter eigenvalue problem using low-rank and qtt tensor approximation*, Journal of Computational Physics, 334 (2017), pp. 221–239.

[9] K. BROWNE, S. QIAO, AND Y. WEI, *A lanczos bidiagonalization algorithm for hankel matrices*, Linear Algebra and its Applications, 430 (2009), pp. 1531–1543.

[10] D. CALVETTI, L. REICHEL, AND D. SORENSEN, *An implicitly restarted lanczos method for large symmetric eigenvalue problems*, Electronic Transactions on Numerical Analysis, 2 (1994), p. 21.

[11] J. CULLUM AND W. DONATH, *A block lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices*, in Decision and Control including the 13th Symposium on Adaptive Processes, 1974 IEEE Conference on, IEEE, 1974, pp. 505–509.

[12] P. DAVIS AND P. RABINOWITZ, *Methods of numerical integration*, Courier Corporation, 2007.

[13] V. DRUSKIN, *On monotonicity of the lanczos approximation to the matrix exponential*, Linear Algebra and its Applications, 429 (2008), pp. 1679–1683.
[14] L. ELBOUYAHYAOUI, A. MESSAOUDI, AND H. SADOK, *Algebraic properties of the block gmres and block arnoldi methods*, Electronic Transactions on Numerical Analysis, 33 (2009), p. 4.
[15] E. ESTRADA AND D. HIGHAM, *Network properties revealed through matrix functions*, SIAM review, 52 (2010), pp. 696–714.
[16] M. FANNES, B. NACHTERGAELE, AND R. F. WERNER, *Finitely correlated states on quantum spin chains*, Communications in mathematical physics, 144 (1992), pp. 443–490.
[17] H. FASSBENDER AND D. KRESSNER, *Structured eigenvalue problems*, GAMM-Mitteilungen, 29 (2006), pp. 297–318.
[18] C. FENU, D. MARTIN, L. REICHEL, AND G. RODRIGUEZ, *Block gauss and anti-gauss quadrature with application to networks*, SIAM Journal on Matrix Analysis and Applications, 34 (2013), pp. 1655–1684.
[19] J. J. GARCÍA-RIPOLL, *Time evolution of matrix product states*, New Journal of Physics, 8 (2006), p. 305.
[20] G. GOLUB, F. LUK, AND M. OVERTON, *A block lanczos method for computing the singular values and corresponding singular vectors of a matrix*, ACM Transactions on Mathematical Software (TOMS), 7 (1981), pp. 149–169.
[21] G. GOLUB AND G. MEURANT, *Matrices, moments and quadrature*, Pitman Research Notes in Mathmatics Series, (1994), pp. 105–105.
[22] ———, *Matrices, moments and quadrature with applications*, Princeton University Press, 2009.
[23] S. GOREINOV, I. OSELEDETS, AND D. SAVOSTYANOV, *Wedderburn rank reduction and krylov subspace method for tensor approximation. part 1: Tucker case*, SIAM Journal on Scientific Computing, 34 (2012), pp. A1–A27.
[24] L. GRASEDYCK, D. KRESSNER, AND C. TOBLER, *A literature survey of low-rank tensor approximation techniques*, GAMM-Mitteilungen, 36 (2013), pp. 53–78.
[25] R. GRIMES, J. LEWIS, AND H. SIMON, *A shifted block lanczos algorithm for solving sparse symmetric generalized eigenproblems*, SIAM Journal on Matrix Analysis and Applications, 15 (1994), pp. 228–272.
[26] M. HASTINGS, *An area law for one-dimensional quantum systems*, Journal of Statistical Mechanics: Theory and Experiment, 2007 (2007), p. P08024.
[27] T. HUCKLE, K. WALDHERR, AND T. SCHULTE-HERBRÜGGEN, *Computations in quantum tensor networks*, Linear Algebra and its Applications, 438 (2013), pp. 750–781.
[28] M. HUTCHINSON, *A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines*, Communications in Statistics-Simulation and Computation, 19 (1990), pp. 433–450.
[29] K. JBILOU, A. MESSAOUDI, AND H. SADOK, *Global fom and gmres algorithms for matrix equations*, Applied Numerical Mathematics, 31 (1999), pp. 49–63.
[30] D. KAREVSKI, *Ising quantum chains*, arXiv preprint cond-mat/0611327, (2006).
[31] B. KHOROMSKIJ, *O (dlog n)-quantics approximation of nd tensors in high-dimensional numerical modeling*, Constructive Approximation, 34 (2011), pp. 257–280.
[32] D. KRESSNER AND C. TOBLER, *Low-rank tensor krylov subspace methods for parametrized linear systems*, SIAM Journal on Matrix Analysis and Applications, 32 (2011), pp. 1288–1316.
[33] A. KRYLOV, *On the numerical solution of the equation by which the frequency of small oscillations is determined in technical problems*, Izv. Akad. Nauk SSSR Ser. Fiz.-Mat, 4 (1931), pp. 491–539.
[34] C. LANCZOS, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*, United States Governm. Press Office Los Angeles, CA, 1950.
[35] D. MARTIN, *Quadrature approximation of matrix functions, with applications*, PhD thesis, Kent State University, 2012.
[36] N. MASTRONARDI, M. SCHUERMANS, M. VAN BAREL, R. VANDEBRIL, AND S. VAN HUFFEL, *A lanczos–like reduction of symmetric structured matrices to semiseparable form*, Calcolo, 42 (2005), pp. 227–241.
[37] V. MEHRMANN AND D. WATKINS, *Structure-preserving methods for computing eigenpairs of large sparse skew-hamiltonian/hamiltonian pencils*, SIAM Journal on Scientific Computing, 22 (2001), pp. 1905–1925.
[38] C. MEYER, *The idea behind krylov methods*, The American mathematical monthly, 105 (1998), pp. 889–899.
[39] P. MONTGOMERY, *A block lanczos algorithm for finding dependencies over gf (2)*, in International Conference on the Theory and Applications of Cryptographic Techniques, Springer, 1995, pp. 106–120.
[40] M. NEWMAN, *Networks: an introduction*, Oxford university press, 2010.
[41] I. OSELEDETS, *Tensor-train decomposition*, SIAM Journal on Scientific Computing, 33 (2011), pp. 2295–2317.
[42] D. PEREZ-GARCIA, F. VERSTRAETE, M. WOLF, AND I. CIRAC, *Matrix product state representations*, arXiv preprint quant-ph/0608197, (2006).
[43] B. PIRVU, V. MURG, I. CIRAC, AND F. VERSTRAETE, *Matrix product operator representations*, New Journal of Physics, 12 (2010), p. 025012.
[44] L. REICHEL, M. SPALEVIĆ, AND T. TANG, *Generalized averaged gauss quadrature rules for the approxi-*

mation of matrix functionals, BIT Numerical Mathematics, (2015), pp. 1–23.
[45] Y. SAAD, *Numerical methods for large eigenvalue problems*, vol. 158, SIAM, 1992.
[46] Y. SAAD AND M. SCHULTZ, *Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM Journal on scientific and statistical computing, 7 (1986), pp. 856–869.
[47] S. SACHDEV, *Quantum phase transitions*, Wiley Online Library, 2007.
[48] J. J. SAKURAI AND E. D. COMMINS, *Modern quantum mechanics, revised edition*, AAPT, 1995.
[49] U. SCHOLLWÖCK, *The density-matrix renormalization group in the age of matrix product states*, Annals of Physics, 326 (2011), pp. 96–192.
[50] M. SHAO, F. H. DA JORNADA, L. LIN, C. YANG, J. DESLIPPE, AND S. G. LOUIE, *A structure preserving lanczos algorithm for computing the optical absorption spectrum*, arXiv preprint arXiv:1611.02348, (2016).
[51] M. SHAO AND C. YANG, *Properties of definite bethe–salpeter eigenvalue problems*, in International Workshop on Eigenvalue Problems: Algorithms, Software and Applications in Petascale Computing, Springer, 2015, pp. 91–105.
[52] S. SUZUKI, J. INOUE, AND B. CHAKRABARTI, *Quantum Ising phases and transitions in transverse Ising models*, vol. 862, Springer, 2012.
[53] J. TANG AND Y. SAAD, *A probing method for computing the diagonal of a matrix inverse*, Numerical Linear Algebra with Applications, 19 (2012), pp. 485–501.
[54] L. TREFETHEN AND D. BAU III, *Numerical linear algebra*, vol. 50, Siam, 1997.
[55] F. VERSTRAETE AND I. CIRAC, *Matrix product states represent ground states faithfully*, Physical Review B, 73 (2006), p. 094423.
[56] F. VERSTRAETE, J. GARCIA-RIPOLL, AND I. CIRAC, *Matrix product density operators: simulation of finite-temperature and dissipative systems*, Physical review letters, 93 (2004), p. 207204.
[57] F. VERSTRAETE, V. MURG, AND I. CIRAC, *Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems*, Advances in Physics, 57 (2008), pp. 143–224.
[58] F. VERSTRAETE, D. PORRAS, AND I. CIRAC, *Density matrix renormalization group and periodic boundary conditions: a quantum information perspective*, Physical review letters, 93 (2004), p. 227205.
[59] G. VIDAL, *Efficient classical simulation of slightly entangled quantum computations*, Physical Review Letters, 91 (2003), p. 147902.
[60] ———, *Efficient simulation of one-dimensional quantum many-body systems*, Physical review letters, 93 (2004), p. 040502.
[61] H. VOSS, *A symmetry exploiting lanczos method for symmetric toeplitz matrices*, Numerical Algorithms, 25 (2000), pp. 377–385.
[62] K. WALDHERR, *Numerical Linear and Multilinear Algebra in Quantum Control and Quantum Tensor Networks*, Verlag Dr. Hut, 2014.
[63] M. ZWOLAK AND G. VIDAL, *Mixed-state dynamics in one-dimensional quantum lattice systems: a time-dependent superoperator renormalization algorithm*, Physical review letters, 93 (2004), p. 207205.

C Efficient approximation for global functions of matrix product operators

Authors Moritz August and Dr. Dr. Mari Carmen Bañuls

Citation arXiv:1804.03613v2 [quant-ph], 2018

Summary Based on the previous results of the works shown in Appendices A and B, we demonstrated in this article, from a physical perspective, how to approximate any operator function of the form $\mathsf{Tr} f(A)$ provided that the input A is expressed as a Hermitian matrix product operator. We argued that our method gives access to global quantities of interest that are challenging or impossible to access with established tensor network techniques. Based on this, we showed how our method yields a novel strategy for approximating thermal properties of both short- and long-range Hamiltonians. This is due to the fact that in our method thermal properties can be approximated directly as functions of the Hamiltonian as opposed to an MPO representing the thermal state. Our method thus avoids the necessity to obtain such an MPO expression of a Gibbs state by performing the costly imaginary time evolution. We demonstrated how global quantities such as the von Neumann entropy

$$S(\rho) = -\mathsf{Tr}\rho \log \rho, \tag{C.1}$$

the heat capacity, the trace norm or the thermal fidelity can be approximated with our method. Additionally, we showed how the method can also be employed to approximate expectation values for positive operators O. We then illustrated the performance of our method by providing numerical results for thermal equilibrium states of two Hamiltonians, the transverse-field Ising and the Lipkin-Meshkov-Glick Hamiltonians. For the former we showed how to approximate two-particle correlations. We compared our results to results obtained via computing the correlations from a matrix product operator representation of the thermal state and generally found a good agreement. For the latter Hamiltonian, the numerical results illustrate that our method is able to approximate thermodynamic quantities and distance functions even though it exhibits long-range interactions. Based on these quantities, we were able to capture signals of the termal phase transition of the Hamiltonian.

Contribution For this work, the author was the main contributor to the analytical derivations, implementational tasks, the evaluation of the implemented ideas and the preparation of the article. The author developed the idea that approximating properties of thermal states directly from the Hamiltonians would both be possible and beneficial and showed how this can practically be done for the above quantities. Furthermore, the author realized that expectation values of positive operators can be expressed in the framework of the developed method. In finding an interesting physical application for these ideas, Dr. Dr. Bañuls kindly provided valuable insights. She also supported the author in discussing implementational questions and provided much valuable feedback on aspects of the numerical evaluation as well as the text and figures in the article.

Efficient approximation for global functions of matrix product operators

Moritz August[1] and Mari Carmen Bañuls[2]

[1] *Department of Informatics, Technical University of Munich, 85748 Garching, Germany*[*]
[2] *Max-Planck Institute for Quantum Optics, 85748 Garching, Germany*[†]

Building on a previously introduced block Lanczos method, we demonstrate how to approximate any operator function of the form $\mathsf{Tr} f(A)$ when the argument A is given as a Hermitian matrix product operator. This gives access to quantities that, depending on the full spectrum, are difficult to access for standard tensor network techniques, such as the von Neumann entropy and the trace norm of an MPO. We present a modified, more efficient strategy for computing thermal properties of short- or long-range Hamiltonians, and illustrate the performance of the method with numerical results for the thermal equilibrium states of the Lipkin-Meshkov-Glick and Ising Hamiltonians.

I. INTRODUCTION

Tensor networks (TN) have proved to be adequate ansätze for the description of ground states, low energy excitations and thermal equilibrium states of quantum many body systems [1–3]. Within some limitations, they can also be used to simulate real time dynamics [4–6]. In particular, the matrix product state (MPS) ansatz constitutes the best understood and most used TN family, and it underlies the success of the celebrated density matrix renormalization group algorithm (DMRG) [7].

The matrix product operator (MPO) generalizes MPS to operators, and provides a variational ansatz for mixed states [8–10]. It can also be used to efficiently describe many Hamiltonians of physical interest, as well as approximate evolution operators, among others, and has become a most useful tool in the application and understanding of TN algorithms in one and two dimensions. An operator given in MPO form can be efficiently applied to a MPS using standard TN techniques, and this operation constitutes a building block for algorithms that search for the ground state or simulate time evolution of a MPS.

The MPO form allows also an efficient calculation of some operator properties (e.g. the trace of the operator or of a few of its integer powers, which gives access to some integer α-Renyi entropies with $\alpha \geq 2$ [11]). However, already for moderate system sizes, it is in general not possible to access the full spectrum of the MPO, and hence there are physical properties that are difficult to estimate. This includes for instance the von Neumann entropy of a mixed state, or the trace distance between two MPOs. These quantities have in common that they can be written as functions of Hermitian matrices $A \in \mathbb{C}^{N\times N}$, such as Hamiltonians or density operators, that for finite dimensional systems take the form $\mathsf{Tr} f(A)$, or more generally sums and products of such terms.

We have recently introduced a numerical method [12] that based on a particular version of the Lanczos algorithm reformulated for MPOs approximates such functions for arbitrary Hermitian inputs. The algorithm implicitly performs a Gauss quadrature approximation and we have shown that it converges to the exact value in the absence of approximation errors.

In this paper we demonstrate that the algorithm can be used to compute physical quantities that are difficult to access in standard MPO calculations. The method reveals to be particularly useful in the case of thermal equilibrium states, since many thermal properties can be written as functions of the Hamiltonian. If the latter has a MPO expression, this can be used, for instance, to detect thermal phase transitions.

The rest of this work is structured as follows: in section II we briefly introduce our approximation method. Section III explicitly shows how to use the method to approximate observables in thermal equilibrium, as well as distances between Gibbs states. This strategy is illustrated with numerical results in section IV. Finally we summarize our conclusions in section V.

II. THE APPROXIMATION METHOD

Krylov type methods have already been used with success in combination with matrix product states (MPS) for the approximation of extremal eigenstates [13] or time evolution [4, 14–17] and dynamical correlation functions [18] as well as spectral functions [19].

In general these methods, that rely on a solid mathematical theory [20, 21], construct a basis of the Krylov subspace for an input matrix A and an initial vector b, $\mathcal{K}(A,b) := \mathrm{span}\{A^0 b, A^1 b, A^2 b, \cdots, A^{K-1} b\}$, and a projection of A onto the subspace, T_K. Their implementation only requires basic linear algebra operations, such as scalar multiplication, addition and inner products of vectors, which allows an easy reformulation in the tensor network framework. In particular the Lanczos algorithm, one of the best known Krylov subspace methods, finds the most significant eigenvalues and eigenvectors of a Hermitian matrix.

The basic vector-based Lanczos algorithm can be generalized to matrices by considering them to be block vectors comprised of several individual column vectors, each of them corresponding to an individual Krylov subspace.

[*] august@in.tum.de
[†] banulsm@mpq.mpg.de

These block Lanczos methods can then construct a basis of block vectors starting from a usually not square initial block that plays the role of the starting vector b used in the standard Lanczos algorithm, see e.g. [22, 23]. The benefit of these algorithms thereby lies in their ability to approximate several extremal eigenvalues or the solution of linear systems with multiple solution vectors simultaneously from a possibly larger search space while often yielding implementations with a better runtime compared to the repeated application of the original Lanczos algorithm.

While most block Lanczos algorithms are based on inner products and norms for the individual column vectors constituting the block vectors, a variant, called global block Lanczos algorithms, has been developed [24–26] which is based on inner products and corresponding norms defined for matrices. Based on these works, in [12] we recently presented a global block Lanczos method, from now on also simply referred to as block Lanczos method, for MPO operators that makes use of TN techniques. In particular we use the Hilbert-Schmidt inner product $\langle U, V\rangle = \mathsf{Tr}\left(U^\dagger V\right)$ and the induced Frobenius norm, and choose the identity matrix as the starting point. The algorithm then constructs a basis for the (operator) subspace $\mathcal{K}(A) = \mathrm{span}\{A^0, A^1, A^2, \cdots, A^{K-1}\}$.

Lanczos methods can be used to approximate functions of the form $b^\dagger f(A) b$ when numerical diagonalization is infeasible [27], a result which relies on a rigorous connection to Gauss quadrature. In the case of the global block method, this connection can be exploited to approximate functions of the form $\mathsf{Tr}\left(f(A)\right)$. More precisely, using any unitary complex initial matrix U, the trace can be written as a Riemann-Stieltjes integral, $\mathsf{Tr}\left[f(A)\right] = \mathsf{Tr}\left[U^\dagger f(A) U\right] = \int f(\lambda) d\mu(\lambda)$, where the integral is over the spectrum of A, and the measure $\mu(\lambda)$ is a piecewise constant distribution depending on the choice U, see e.g. [12] for details. It can be shown [26] that the eigenvalues of the projection T_K of A onto the Krylov subspace, produced by the global block Lanczos method for the initial matrix U, correspond precisely to the nodes of the K-node Gaussian quadrature approximation of the integral above. The corresponding weights are contained in the respective eigenvectors. When the function f is applied to the projected matrix, the first element of the diagonal of $f(T_K)$, scaled by the squared norm of the starting matrix b, evaluates the Gauss quadrature and thus provides an approximation to the trace. However, using a fully unitary starting matrix has so far been prohibitive in practice since in this case the complexity of the algorithm is no better than that of exact diagonalization.

The algorithm proposed in [12] and further analyzed in [28] applies the above property to approximate global functions of an operator A given in MPO form. We show the method schematically as pseudocode in Algorithm 1. As explained above, it proceeds by applying a global block Lanczos method with the identity as initial matrix. The identity is chosen here simply because it yields an exact MPO representation with bond dimension $D = 1$.

Algorithm 1: MPO Function Approximation

Input : MPO $A[D_A] \in \mathbb{C}^{N\times N}$, Starting unitary MPO $U[D_{init}] \in \mathbb{C}^{N\times N}$, Number of Dimensions K, Maximal Bond-Dimension D_{max}, Stopping Criteria $\mathcal{S}$

$U_0 \leftarrow 0$;
$V_0 \leftarrow U$;
for $i \leftarrow 1; i \leq K$ **do**
  $\beta_i \leftarrow \sqrt{\langle V_{i-1}|V_{i-1}\rangle}$;
  if $\beta_i = 0$ **then**
    break ;
  end
  $U_i \leftarrow \texttt{multiply}(1/\beta_i, V_{i-1})$;
  $V_i \leftarrow \texttt{multiply}(A, U_i, D_{max})$;
  $V_i \leftarrow \texttt{sum}(V_i, -\beta_i U_{i-1}, D_{max})$;
  $\alpha_i \leftarrow \langle U_i|V_i\rangle$;
  $V_i \leftarrow \texttt{sum}(V_i, -\alpha_i U_i, D_{max})$;
  $V_T \Lambda_T V_T^* \leftarrow \texttt{spectralDecomposition}(T_i)$;
  $\mathcal{G}f \leftarrow \beta_1^2 e_1^T V_T f(\Lambda_T) V_T^* e_1$;
  if $\texttt{checkStop}(\mathcal{G}f, \Lambda_T, \mathcal{S})$ **then**
    break ;
  end
end
Output: Approximation $\mathcal{G}f$ of $\mathsf{Tr}f(A)$

In principle however, any unitary MPO could be used, a fact we will take advantage of in Section III. It is important to note here that only due to the use of the TN framework we are able to use full unitary starting matrices. The method then constructs the orthogonal basis of the Krylov subspace in the usual way, i.e. successively applying the operator A and orthogonalizing, but restricting all the basis matrices to be of the MPO form with a maximum bond dimension of D_{max}. Iteratively, the method thus constructs the projection matrix

$$T_K = \begin{bmatrix} \alpha_1 & \beta_2 & & \mathbf{0} \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_K \\ \mathbf{0} & & \beta_K & \alpha_K \end{bmatrix}$$

where the α_i and β_i are computed as shown in Algorithm 1. The spectrum of T_K then in essence constitutes an approximation to the spectrum of the input MPO A.

The multiplication and the sum of MPOs, denoted as the subfunctions `sum` and `multiply` in the pseudocode, involve an optimization over MPOs for a given target bond dimension. In our algorithm we use the bond dimension required for an exact representation until it exceeds the maximal bond dimension D_{max}. All the operations involved can be carried out efficiently for MPO operators, so that the method allows the approximation of functions of the A which would be otherwise inaccessible, as in principle they would require to compute the whole spectrum.

Representing the basis elements as MPS, or vectorized MPOs, with limited bond dimension up to $D_{\max}$ introduces a truncation error. Together with the error intro-

duced by the limited dimension of the Krylov subspace, the truncation error determines the accuracy of the approximation. Since in absence of these errors the method is known to converge to the exact result, by adjusting these parameters, the numerical errors can be controlled.

Further details about the theoretical background and the algorithm itself can be found in [12, 26, 27]. In the next sections we show how this method can be applied to compute physical quantities in quantum many-body systems.

III. FUNCTIONS OF GIBBS STATES

For a quantum system governed by a Hamiltonian H, the operator

$$\rho(H,\beta) = \frac{e^{-\beta H}}{Z} \tag{1}$$

describes the thermal equilibrium state at inverse temperature β and is called Gibbs ensemble. The denominator in the expression is the partition function, $Z = \mathsf{Tr} e^{-\beta H}$.

It has been shown that MPOs can approximate Gibbs states of local Hamiltonians accurately [29, 30]. From a numerical perspective, MPO approximations to thermal states can be found with efficient imaginary time evolution algorithms [4, 8, 9].

In principle, we could then apply the block Lanczos method to the MPO approximation found with one such approximation algorithm, in order to evaluate a given function $f(\rho)$. However, the particular form of the Gibbs state allows us to write quantities directly as a function of the Hamiltonian, so that instead of approximating $f(\rho)$ as a function of ρ, we can approximate $f[\rho(H,\beta)]$ as a function of H.

This strategy offers multiple advantages. Firstly, for many interesting physical problems, for instance for local interactions, H has an exact MPO representation. In that case the method does not suffer from initial approximation errors, in contrast to the case where an approximation of $\rho(H,\beta)$ is used as input. In the case of a long-range Hamiltonian, often a good MPO approximation of H exists with reduced bond dimension [10]. In contrast, approximating the Gibbs state as an MPO may be prohibitive in terms of bond dimension, such that this strategy allows access to thermal observables which would be otherwise difficult or infeasible to approximate numerically. A second advantage is that the (exact or approximate) MPO representation of H usually has a much lower bond dimension than what is required for a good approximation of $\rho(H,\beta)$, significantly reducing the overall execution time of the algorithm. Thirdly, since the Krylov subspace constructed by our algorithm only depends on H, we can in principle use it to approximate multiple different functions of H and a broad range of different temperatures in a single run. As we illustrate in the next paragraphs, this strategy can be applied to compute different physical quantities which are not easy to estimate for a general MPO, including the von Neumann entropy, or the trace norm distance between two thermal states at different temperatures. The latter quantity, as the equally accessible thermal fidelity, can be used to detect thermal phase transitions [31, 32]. Other quantities that can be approximated by this method include the heat capacity [33] and correlation functions.

1. *Thermodynamic quantities*

The von Neumann entropy is defined as $S(\rho) = -\mathsf{Tr}(\rho \ln \rho)$. For a thermal state it can be written as a function of H,

$$S[\rho(H,\beta)] = \beta \frac{F}{Z} + \ln Z. \tag{2}$$

where $F = \mathsf{Tr}\left(e^{-\beta H} H\right)$ is the free energy. Thus S can be expressed in terms of two trace functions, F and Z, that depend on the same input operator H. Therefore, as discussed above, we can approximate and then combine F and Z to yield S in a single run of the algorithm. On the other hand, β is just a parameter of the function, so that we can also approximate S for arbitrarily many values of β simultaneously in one run of the method. In the following, we will consider the von Neumann entropy per particle denoted as $s = S/L$ where L is the system size.

We can also estimate other thermodynamic quantities that derive from the partition function, i. e. any quantity that is expressible in terms of derivatives of Z or $\ln Z$. Next to the von Neumann entropy, one other such quantity, which is especially relevant in the context of probing thermal phase transitions, is the specific heat capacity (i.e. heat capacity per site)

$$c = \frac{\beta^2}{L} \frac{\partial^2 \ln Z}{\partial \beta^2} \tag{3}$$

which can be written as

$$c = \frac{\beta^2}{L} \left[\frac{G}{Z} - \left(\frac{F}{Z} \right)^2 \right] \tag{4}$$

where $G = \mathsf{Tr}\left(H^2 e^{-\beta H}\right)$. We can obtain c by simultaneously approximating F, G and Z, as in the case of the von Neumann entropy.

2. *Distance measures*

The trace norm of an operator A, defined $\|A\|_1 = \mathsf{Tr}\sqrt{A^\dagger A}$, is difficult to compute for MPOs, as in general no efficient MPO description exists for the square root. However, it can be easily computed using the block Lanczos method, as we discussed in [12]. If the operator

is a function of the Hamiltonian, the strategy presented here will in general improve the approximation as compared to the case where an MPO representation of $A^\dagger A$ is used as input.

This strategy gives easy access to the trace distance between thermal states at different temperatures. More concretely, for two inverse temperatures β_0 and β_1, the trace distance

$$D_T(\beta_0,\beta_1) = \left\| \frac{e^{-\beta_0 H}}{Z(\beta_0)} - \frac{e^{-\beta_1 H}}{Z(\beta_1)} \right\|_1 \tag{5}$$

is a function of the Hamiltonian H. To approximate it with our method we need an estimate for the partition functions at both temperatures $Z(\beta_0)$ and $Z(\beta_1)$. As before, the latter can be approximated independently, and their values can then be used to evaluate the trace distance. Notice that the distance between two arbitrary MPO operators can be approximated by straightforward application of the method in [12] to the MPO that represents their difference. Using this procedure we can compute also trace distances between thermal equilibrium states at the same temperature, but varying Hamiltonian parameters, whenever a MPO approximation can be found to the Gibbs states. This however may be difficult, for instance if the Hamiltonian has long-range interactions.

Another distance measure between quantum states is the Uhlmann fidelity, which for a pair of states ρ and σ is defined as $F(\rho,\sigma) = \mathsf{Tr}\sqrt{\rho^{1/2}\sigma\rho^{1/2}}$. As was shown in [31] and further studied in [32], the *thermal fidelity*, i.e. the fidelity between two thermal equilibrium states for the same Hamiltonian at different temperatures, can detect a thermal phase transition. For two inverse temperatures β_0 and β_1, the thermal fidelity can be expressed in terms of three partition functions,

$$F_T(\beta_0,\beta_1) = \frac{Z\left(\frac{\beta_0+\beta_1}{2}\right)}{\sqrt{Z(\beta_0)Z(\beta_1)}}, \tag{6}$$

which implies that it can be approximated with the strategy outlined in the previous section.

3. Other expectation values

Interestingly, the same method can also be used to approximate the expectation values of some observables. Generically, expectation values take the form $\mathsf{Tr}\,[f(A)O]$ which in principle cannot be directly approximated. However, if O is positive, we can write

$$\mathsf{Tr}\,[f(A)O] = \mathsf{Tr}\left[\sqrt{O} f(A) \sqrt{O}^\dagger\right] = \int f(\lambda) d\mu_O(\lambda), \tag{7}$$

with a function μ_O specific for the operator O. The positivity of O is required to ensure that μ_O is a distribution function. If $\sqrt{O}$ admits an efficient representation as MPO, we can compute the expression above by simply using $\sqrt{O}^\dagger$ as the starting point of our algorithm, instead of the identity. Even if the operator O is not positive, it can always be written as linear combination of positive terms, e.g. splitting it in positive and negative part, although other decompositions with more terms may be more convenient. If each of these positive terms has an efficient MPO representation of its square root, it will be possible to approximate the expectation value by computing each contribution independently. Operators that allow this treatment, are, for instance, few-body spin correlators, as we detail in the next section.

IV. NUMERICAL RESULTS

To illustrate the performance of the method we present numerical results for the exploration of a thermal phase transition in a long-range model, and the extraction of two-point spin correlation functions.

A. Phase Transition of the LMG model

The Lipkin-Meshkov-Glick (LMG) Hamiltonian [34–36] for a system of L spin-1/2 particles is given by

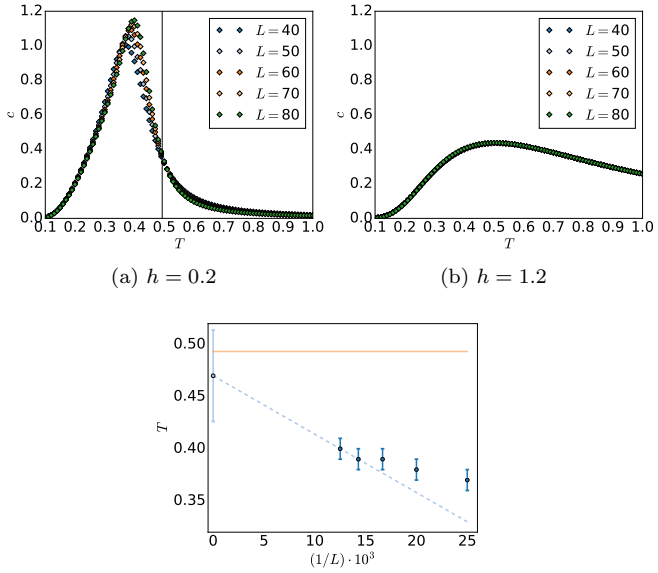
$$H = -\frac{S_x^2}{L} - 2hS_z, \tag{8}$$

where $S_\alpha = \sum_i^L \sigma_i^\alpha/2$ for $\alpha \in \{x,y,z\}$ are the collective spin operators, and each σ_i^α stands for the corresponding Pauli operator at site i. The model exhibits a quantum phase transition for $h = 1$ [37] and a thermal phase transition for $h < 1$ at a critical temperature

$$T_c(h) = \frac{h}{2\tanh^{-1}(h)}. \tag{9}$$

The model has been largely studied in the literature, at both zero and finite temperature. Here we focus on the properties of the thermal equilibrium states, so specially related to our study are some recent works that explored the thermal phase transition from a quantum information point of view. The first study of the finite temperature phase diagram of the model using the thermal fidelity was performed in [32]. In Wilms et. al. [33], using both analytical and numerical results, it was demonstrated that the mutual information, which measures quantum and classical correlations, was sensitive to the phase transition. The fidelity metric was also used by Scherer et. al. [38] to explore analytically the phase transition of the isotropic LMG model.

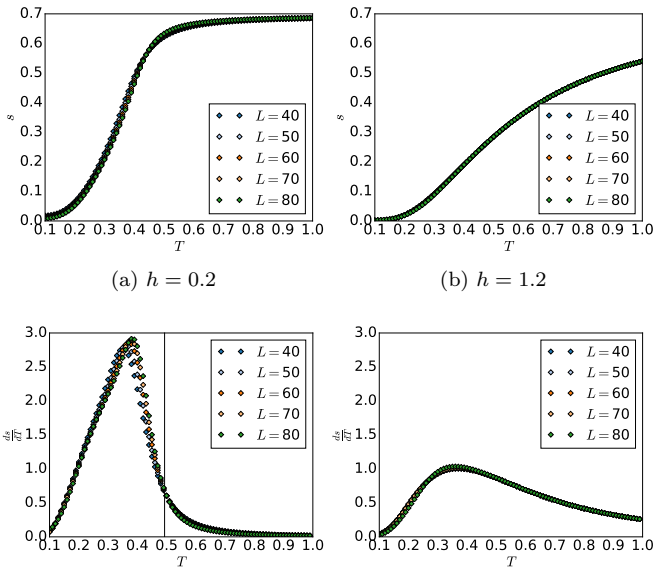
Besides the thermal fidelity, other quantities can be analyzed to locate the critical temperature from numerical studies of finite systems. Here we use the block Lanczos algorithm described in the previous sections to compute the specific heat capacity and the entropy density, as well as the thermal fidelity and trace distance, and we show



(a) $h = 0.2$ (b) $h = 1.2$

(c) Extrapolation of T_c

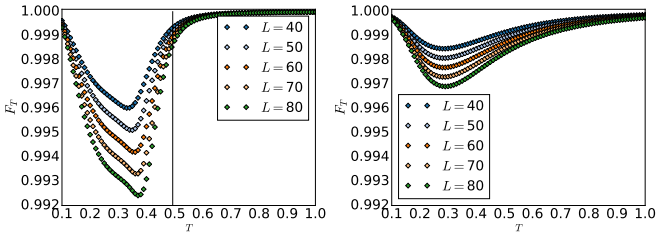
FIG. 1. Heat capacity per particle in the thermal equilibrium state of the LMG model, for different system sizes $40 \leq L \leq 80$ and $h = 0.2$ (a) and $h = 1.2$ (b) as a function of temperature. The vertical scale is the same in both plots for the sake of comparison. Our results are in good agreement with those presented in [33]. The black vertical line in Figure (a) indicates the critical temperature T_c. Figure (c) shows the scaling of the peak position with the system size. The error bars of the data points reflect the imprecision in the location of each peak. Extrapolating the results of the largest systems ($L = 70$, 80) we estimate a critical temperature $T_c \approx 0.47$.



(a) $h = 0.2$ (b) $h = 1.2$

(c) $h = 0.2$ (d) $h = 1.2$

FIG. 2. Top row: von Neumann entropy per site s of the Gibbs state for the LMG Hamiltonian for $h = 0.2$ (a) and $h = 1.2$ (b) over the temperature T for several system sizes. Bottom row: the discrete temperature derivative of s over T for $h = 0.2$ (c) and $h = 1.2$ (d).



(a) $h = 0.2$ (b) $h = 1.2$

FIG. 3. Thermal fidelity F_T in the Gibbs state for the LMG Hamiltonian, for $h = 0.2$ (a) and $h = 1.2$ (b) and several system sizes. The black vertical line in Figure (a) indicates the critical temperature T_c.

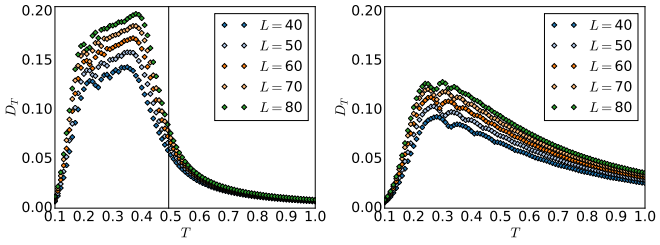
how the results are sensitive to the presence of a phase transition.

The Hamiltonian (8) can be exactly expressed as an MPO with constant bond dimension $D = 3$. In contrast, approximating the Gibbs state as an MPO would be difficult, due to the long-range interactions. Although specific algorithms exist that can approximate the required imaginary time evolution for long-range Hamiltonians [39–41], the bond dimension required for an accurate description can get large, up to the point of not being computable in practice. Thus, expressing the thermal observables directly as functions of the Hamiltonian and applying the block Lanczos method to the latter is a more reasonable strategy for this kind of models.

We consider here finite systems of sizes $L \in \{40, 50, 60, 70, 80\}$ and use our method to probe the behavior of the quantities of interest as a function of temperature in the interval $T \in [0.1, 1]$, for $h \in \{0.2, 1.2\}$, (only for $h < 1$ the system exhibits a phase transition). The maximal size of the Krylov basis is set to 70, and the maximal bond dimension of the basis MPOs varies from $D = 150$ for $L = 40$ to $D = 250$ for $L = 80$. For the sake of brevity, we will in the following refer to the thermal fidelity at a given temperature T, for states that differ in δT, as $F_T(T) := F_T(\frac{1}{T}, \frac{1}{T+\delta T})$ and define the trace distance $D_T(T)$ analogously.

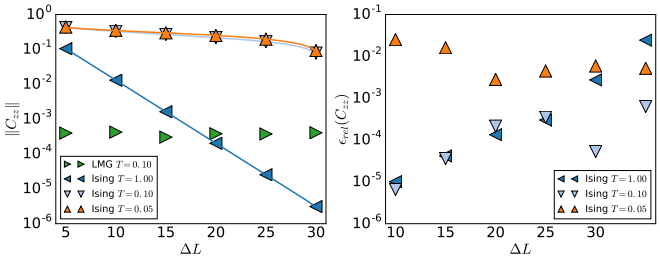
Note that for a given system size L and field strength h, the algorithm can approximate all the required functions for all values of T in a single run.

Our results are shown in Figures 1 (heat capacity per site c), 2 (entropy per site s), 3 (thermal fidelity F_T) and 4 (trace distance D_T). Comparing the plots for both values of h studied, specially for heat capacity, thermal fidelity and trace distance, we see a clear signal of the presence of the phase transition for $h = 0.2$, despite the system sizes considered here being far from the thermodynamic limit. In particular, the location of the maximum (minimum in the case of F_T) can be used to estimate the critical



(a) $h = 0.2$ (b) $h = 1.2$

FIG. 4. Trace distance D_T between thermal equilibrium states of the LMG model as a function of the temperature for different system sizes L and $h = 0.2$ (a) and $h = 1.2$ (b). The black vertical line in Figure (a) indicates the critical temperature T_c in Eq. (9).



(a) Absolute values (b) Relative difference

FIG. 5. Two-site correlation function $C_{zz}(L/2, L/2 + \Delta L)$ in the thermal state of Ising and LMG Hamiltonians, as a function of the distance ΔL, for a system size $L = 60$. The discrete data points shown as markers correspond to the results computed with our algorithm, while for the Ising case, quasi-exact results from the MPO approximation are shown as lines, for reference. In (a), the absolute values are shown for both Hamiltonians while (b) depicts the relative error for the Ising model.

temperature. Finite size effects are noticeable, with the location of the extremes of all quantities moving closer to the exact T_c (9) for larger system sizes. We can estimate the location of the critical temperature by a finite size extrapolation, as shown in Figure 1(c). Using the data of the heat capacity for $L = 70, 80$, provides already a relatively good estimation of the critical temperature $T_c \approx 0.47 \pm 0.04$, where the error corresponds to the difference with respect to the estimator obtained from including also $L = 60$ in the fit. The extrapolated value is already close to the exact solution, and compatible with it within the error bars. To obtain a more precise estimate of the critical temperature, more data for larger systems sizes and possibly a smaller temperature step could be computed with our method.

The von Neumann entropy per site does not show a sharp peak, but it also exhibits a qualitatively different behavior for both values of h. For $h = 0.2$, where a phase transition exists, the value of the entropy per site of the thermal equilibrium state develops a rapid change from $s = 0$ at low temperatures, to $s \approx 0.7$ after the critical temperature, with the increase becoming faster with growing system sizes. In contrast, in the case $h = 1.2$ the entropy density increases smoothly and does not show a clear finite size scaling. This different behavior is made more apparent by looking at the (discrete) derivative of the entropy with respect to the temperature, shown in Figures 2(c) and 2(d). In these plots, similar to the behavior of the heat capacity, we appreciate a sharp peak signaling the phase transition only in the case $h = 0.2$.

Overall the results show that already with moderate computational effort, our method can be used to reveal interesting physical phenomena that would otherwise be hard to access numerically. As the method is theoretically guaranteed to converge to the exact solution in absence of numerical and approximation errors, more accurate results can in principle be obtained by increasing the maximal bond dimension and size of the Krylov basis. Indeed, due to the permutation symmetry of the Hamiltonian (8), in this particular model large systems can be explored with exact diagonalization, as was done in [32, 33]. The goal of the calculations presented above was thus not to compete with those results, but to use them as reference and to probe the performance of the algorithm. We expect that the strategy presented here will be most useful for cases where the dimension of the problem is genuinely exponential, and exact diagonalization cannot be applied.

B. Two-site correlations

An interesting property of thermal equilibrium states may be the two-site correlations at a certain distance. If the Gibbs state admits a good MPO approximation which can be efficiently found, for instance via standard imaginary time evolution algorithms, then such correlations can be accurately computed. Our method will be most useful when no such MPO representation is available, but for the purpose of benchmarking we choose a model where a good MPO approximation of the thermal state is easy to find. In particular, we consider the Ising model in a transverse field on finite open chains,

$$H_I = J \sum_{i=1}^{L-1} \sigma_i^x \sigma_{i+1}^x + g \sum_{i=1}^{L} \sigma_i^z, \tag{10}$$

and use the standard approximation of the Gibbs ensemble as an MPO as a quasi-exact reference to benchmark the estimations obtained with our method. We will also illustrate the results obtained for the thermal states of the LMG Hamiltonian discussed above, although in that case no MPO approximation is available for comparison.

We define a two-point correlation function $C_{zz}(i, j) = \langle \sigma_i^z \sigma_j^z \rangle - \langle \sigma_i^z \rangle \langle \sigma_j^z \rangle$. In the case of a mixed state ρ,

$$C_{zz}(i, j) = \mathsf{Tr}\left(\rho\, \sigma_i^z \sigma_j^z\right) - \mathsf{Tr}\left(\rho\, \sigma_i^z\right) \mathsf{Tr}\left(\rho\, \sigma_j^z\right), \tag{11}$$

which yields three expectation values that need to be computed or approximated. We have seen in Section III that our method can approximate expectation values of positive operators O, if there is an efficient MPO representation for $\sqrt{O}$. Although σ_i^z and $\sigma_i^z\sigma_j^z$ are not positive, they can be decomposed using $\sigma_i^z = \Pi_0^i - \Pi_1^i$ and

$$\sigma_i^z\sigma_j^z = \left(\Pi_0^i\Pi_0^j + \Pi_1^i\Pi_1^j\right) - \left(\Pi_1^i\Pi_0^j + \Pi_0^i\Pi_1^j\right), \quad (12)$$

where $\Pi_s^i := |s\rangle\langle s|$ for $s = 0, 1$ projects the i-th site of the chain onto state $|s\rangle$ and acts as the identity on the remaining sites. For simplicity we do not explicitly show the identity operators. Each of the terms within brackets is a projector, and therefore positive, and has a square root (itself) that admits an MPO-representation with bond dimension $D = 1$ for σ_i^z or $D = 2$ for $\sigma_i^z\sigma_j^z$. This formulation thus provides us with a means of approximating two-site correlations with the block Lanczos method.

In the particular case of the Ising model, the spin-flip symmetry of the Hamiltonian (10) allows a further simplification, since we can write

$$\mathsf{Tr}\left[\rho(H_I,\beta)\sigma_i^z\sigma_j^z)\right] = 2\left\{\mathsf{Tr}\left(\rho(H_I,\beta)\Pi_0^i\Pi_0^j\right)\right. \quad (13)$$

$$\left. -\mathsf{Tr}\left(\rho(H_I,\beta)\Pi_1^i\Pi_0^j\right)\right\}, \quad (14)$$

halving the number of expectation values that need to be approximated and consequentially improving accuracy and runtime.

In Figure 5, we show the results for the correlation $C_{zz}(L/2, L/2 + \Delta L)$, between the middle site and the right half of the chain, in the Gibbs state at temperatures $T \in \{0.05, 0.1, 1\}$, for a system size $L = 60$, $J = 1$ and transverse field $g = 1$. The bond dimension was set to $D = 250$ and the maximal Krylov dimension to 100. We compare the results to those from the MPO obtained by imaginary time evolution with the purification ansatz, which were obtained using bond dimension $D = 120$ for the purification and Trotter step $\delta = 0.01$ and checked to be sufficiently converged for this comparison.

Figure 5(a) shows that the results obtained for the Ising Hamiltonian by the two different methods are in good agreement for all the temperature values considered. The correlations in the thermal equilibrium state decay exponentially with distance, with the correlation length becoming larger for lower temperatures, as the quantum critical point at $T = 0$ is approached. The relative errors, shown in Figure 5(b), in general increase with the distance, which we attribute to the fact that the absolute values are indeed smaller, since we observe an approximately constant absolute error for all distances. Notice that the results for the LMG case, shown also in Figure 5(a), show that the method can capture the correlation of the model despite its long-range interactions. We observe that for the smallest temperature, $T = 0.05$, the error is considerably larger. We attribute this comparatively larger deviation mainly to an error in the approximation of the partition function as it enters all approximated expectation values. The error reflects the fact that a larger bond dimension and possibly a larger Krylov dimension is required for $T = 0.05$ to obtain more accurate estimates.

V. CONCLUSION

In [12] we introduced a block Lanczos method for approximating functions of the form $\mathsf{Tr}\left[f(A)\right]$ of any Hermitian operator A given as an MPO. The method gives access to global functions of the operator that depend on the full spectrum and are usually not accessible with standard tensor network tools. In this work we have discussed how to use the method for physically interesting quantities, such as the von Neumann entropy or the trace norm of an MPO, and in particular how to most efficiently approximate thermal properties. As we have argued, by expressing the quantities of interest in the thermal equilibrium state as functions of the Hamiltonian, and applying the Lanczos method directly to the latter, we can explore the thermal properties in an efficient way. The strategy allows the evaluation of a variety of properties without the need of a prior approximation of the thermal state as an MPO, and can be applied for long-range interactions. We have discussed how a single run of the algorithm is enough to evaluate different functions over the whole range of possible temperatures.

We have then shown how to approximate several physical quantities for Gibbs states, namely the heat capacity, thermal fidelity, trace distance, von Neumann entropy and some expectation values. To illustrate the performance of the method we have presented results for the thermal states of the Lipkin-Meshkov-Glick and Ising Hamiltonians. In the LMG model, we have shown how the method can estimate multiple quantities that detect the presence of a thermal phase transition. The case of the Ising model has been used to benchmark the approximation of correlation functions.

This algorithm provides a new tool to extend the capabilities of the tensor network toolbox, by giving access to global functions whose calculation would otherwise be unfeasible. The calculations presented in this paper aim at benchmarking the method with known results, but we expect the technique to be most useful in cases when an MPO approximation of the Gibbs state is not available, but the Hamiltonian has an MPO description, as can be the case for long-range interactions.

Apart from algorithmic improvements such as leveraging symmetries in the input to reduce the required bond dimension and runtime, an interesting and promising direction of future research would be to identify additional physical applications of the algorithm that so far were out-of-reach for tensor network algorithms. The algorithm's generality and capability of approximating global quantities make this seem like a worthwhile endeavor.

ACKNOWLEDGMENTS

This work was partly funded by the Elite Network of Bavaria via the doctoral programme *Exploring Quantum Matter (ExQM)*.

[1] U. Schollwöck, Annals of Physics **326**, 96 (2011).
[2] F. Verstraete, V. Murg, and J. I. Cirac, Advances in Physics **57**, 143 (2008).
[3] R. Orús, Ann. Phys. **349**, 117 (2014).
[4] J. J. García-Ripoll, New Journal of Physics **8**, 305 (2006).
[5] T. Barthel, U. Schollwöck, and S. R. White, Physical Review B **79**, 245101 (2009).
[6] A. Müller-Hermes, J. I. Cirac, and M. C. Bañuls, New Journal of Physics **14**, 075003 (2012).
[7] S. R. White, Phys. Rev. Lett. **69**, 2863 (1992).
[8] F. Verstraete, J. Garcia-Ripoll, and I. Cirac, Physical review letters **93**, 207204 (2004).
[9] M. Zwolak and G. Vidal, Physical review letters **93**, 207205 (2004).
[10] B. Pirvu, V. Murg, I. Cirac, and F. Verstraete, New Journal of Physics **12**, 025012 (2010).
[11] A. Rényi, *On measures of entropy and information*, Tech. Rep. (HUNGARIAN ACADEMY OF SCIENCES Budapest Hungary, 1961).
[12] M. August, M. C. Bañuls, and T. Huckle, Electronic Transactions on Numerical Analysis **46**, 215 (2017).
[13] T. Huckle and K. Waldherr, PAMM **12**, 641 (2012).
[14] P. Schmitteckert, Phys. Rev. B **70**, 121302 (2004).
[15] M. L. Wall and L. D. Carr, New Journal of Physics **14** (2012).
[16] C. Hubig, *Symmetry-protected tensor networks*, Ph.D. thesis, Ludwig Maximilian University of Munich (2017).
[17] T. Keilmann and J. J. García-Ripoll, Physical review letters **100**, 110406 (2008).
[18] P. E. Dargel, A. Wöllert, A. Honecker, I. P. McCulloch, U. Schollwöck, and T. Pruschke, Phys. Rev. B **85**, 205119 (2012).
[19] P. E. Dargel, A. Honecker, R. Peters, R. M. Noack, and T. Pruschke, Phys. Rev. B **83**, 161104 (2011).
[20] A. Krylov, Izv. Akad. Nauk SSSR Ser. Fiz.-Mat **4**, 491 (1931).
[21] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators* (United States Governm. Press Office Los Angeles, CA, 1950).
[22] G. H. Golub and R. Underwood, in *Mathematical software* (Elsevier, 1977) pp. 361–377.
[23] G. H. Golub, F. T. Luk, and M. L. Overton, ACM Transactions on Mathematical Software (TOMS) **7**, 149 (1981).
[24] K. Jbilou, A. Messaoudi, and H. Sadok, Applied Numerical Mathematics **31**, 49 (1999).
[25] L. Elbouyahyaoui, A. Messaoudi, and H. Sadok, Electronic Transactions on Numerical Analysis **33**, 4 (2009).
[26] M. Bellalij, L. Reichel, G. Rodriguez, and H. Sadok, Applied Numerical Mathematics **94**, 127 (2015).
[27] G. Golub and G. Meurant, Pitman Research Notes in Mathmatics Series , 105 (1994).
[28] M. August and T. Huckle, arXiv preprint arXiv:1709.06847 (2017).
[29] M. B. Hastings, Phys. Rev. B **73**, 085115 (2006).
[30] A. Molnar, N. Schuch, F. Verstraete, and J. I. Cirac, Phys. Rev. B **91**, 045138 (2015), 1406.2973.
[31] P. Zanardi, H. Quan, X. Wang, and C. Sun, Physical Review A **75**, 032109 (2007).
[32] H. Quan and F. Cucchietti, Physical Review E **79**, 031101 (2009).
[33] J. Wilms, J. Vidal, F. Verstraete, and S. Dusuel, Journal of Statistical Mechanics: Theory and Experiment **2012**, P01023 (2012).
[34] H. J. Lipkin, N. Meshkov, and A. Glick, Nuclear Physics **62**, 188 (1965).
[35] N. Meshkov, A. Glick, and H. Lipkin, Nuclear Physics **62**, 199 (1965).
[36] A. Glick, H. Lipkin, and N. Meshkov, Nuclear Physics **62**, 211 (1965).
[37] R. Botet, R. Jullien, and P. Pfeuty, Phys. Rev. Lett. **49**, 478 (1982).
[38] D. D. Scherer, C. A. Müller, and M. Kastner, Journal of Physics A: Mathematical and Theoretical **42**, 465304 (2009).
[39] J. Haegeman, J. I. Cirac, T. J. Osborne, I. Pižorn, H. Verschelde, and F. Verstraete, Physical review letters **107**, 070601 (2011).
[40] M. P. Zaletel, R. S. Mong, C. Karrasch, J. E. Moore, and F. Pollmann, Physical Review B **91**, 165112 (2015).
[41] B.-B. Chen, Y.-J. Liu, Z. Chen, and W. Li, Phys. Rev. B **95**, 161104 (2017).

D Using recurrent neural networks to optimize dynamical decoupling for quantum memory

Authors Moritz August and Dr. Xiaotong Ni

Citation Physical Review A, 95(1):012335, 2017

DOI 10.1103/PhysRevA.95.012335

Summary In this work we demonstrated how recurrent neural networks can be employed to optimize dynamical decoupling in quantum memory. We introduced LSTM networks and argued that they are well suited for probabilisticaly modelling control sequences for dynamical decoupling. Dynamical decoupling hereby poses a comparably easy method to suppress errors in time-evolving quantum systems under certain assumptions. We presented an optimization algorithm which is inspired by evolutionary algorithms. In each iteration of a nested optimization loop, it first constructs a data set of (increasingly good) control sequences by sampling from the LSTMs and then trains them on it. This process is repeated such that finally LSTMs approximating the probability distribution over optimal or near-optimal sequences are obtained. Furthermore, the presented method can be easily employed in real experiments to find sequences tailored to the particular hardware setup, since it treats the error function as a black box. We evaluated our method for a quantum memory setting with a reasonable set of physical assumptions. Here, we found the LSTM networks to improve over time and finally generate sequences with performance superior to certain analytical solutions. These results were thereby obtained with minimum use of domain knowledge and starting from uniformly random sequences. Additionally, we compared the results obtained by modelling the sequences with LSTM networks to the case where n-gram models were used instead. Here we found the LSTM to perform better, suggesting that their ability to capture long-range correlations is indeed useful. We also showed that the best dynamial decoupling sequences we found did in fact exhibit strong structure, providing further justification for our modelling approach.

Contribution In this work, the author contributed to the development of the method itself, its implementation and the subsequent evaluation as well as the preparation of the article. More precisely, Dr. Ni provided the motivation and physical domain knowledge for the quantum memory setting. Based on this background information, the author then evaluated several learning methods for the problem and eventually found the presented method. During the development of the method, Dr. Ni was often available for helpful discussion. The implementation of the method was conducted by the author, while Dr. Ni contributed to the implementation of the simulation of the quantum time evolution. The numerical experiments were mainly conducted by the author. The figures shown in the article were prepared together by Dr. Ni and the author. The more physical sections of the article and the discussion on optimization methods were mostly written by Dr. Ni, while the author provided the rest of the text and was generally responsible for the final form of the article.

PHYSICAL REVIEW A **95**, 012335 (2017)

Using recurrent neural networks to optimize dynamical decoupling for quantum memory

Moritz August*
Department of Informatics, Technical University of Munich, 85748 Garching, Germany

Xiaotong Ni†
Max-Planck Institute for Quantum Optics, 85748 Garching, Germany
(Received 23 September 2016; published 27 January 2017)

We utilize machine learning models that are based on recurrent neural networks to optimize dynamical decoupling (DD) sequences. Dynamical decoupling is a relatively simple technique for suppressing the errors in quantum memory for certain noise models. In numerical simulations, we show that with minimum use of prior knowledge and starting from random sequences, the models are able to improve over time and eventually output DD sequences with performance better than that of the well known DD families. Furthermore, our algorithm is easy to implement in experiments to find solutions tailored to the specific hardware, as it treats the figure of merit as a black box.

DOI: 10.1103/PhysRevA.95.012335

I. INTRODUCTION

A major challenge of quantum information processing (e.g., quantum computation and communication) is to preserve the coherence of quantum states. While in principle we can build a fault-tolerant quantum memory or universal quantum computer once the error rate of the device is below a certain threshold, it is still beyond nowadays experimental capacity to build a decent-size quantum computer. One less explored area is the optimization of implementing a fault-tolerant protocol on a concrete experimental setting. This is often a tedious problem, due to the amount of details in the real devices and the fact that the architectures of both experimental devices and theoretical protocols are still rapidly changing. Thus, an attractive approach is to automatize this optimization task. Apart from convenience, it is conceivable that with less human intuition imposed, the upper bound of the performance will be higher. This has previously been proven to be true in fields such as computer vision where artificial neural network (ANN) models that try to solve tasks without using handcrafted representations of data have overtaken approaches based on human insight in tasks such as image classification and object recognition [1]. Another interesting recent example is the ability of ANNs to learn how to play games on a human or even superhuman level without any or just little prior knowledge about the respective games [2,3].

Automatically optimizing parameters in real (or numerical simulations of) experiments is not a new idea. For example, it has been applied to optimizing the pulse shape of a laser, the parameters of Hamiltonians to achieve certain unitary operations, or the parameters of dynamical decoupling and cold-atom experiments. Most works that attempt to obtain optimal parameters use genetic algorithms [4–8] or (to some degree [9]) local searches such as gradient descent [7,10–14] and the Nelder-Mead simplex method [15–17]. It is argued that by using these optimization methods directly on the experiments, we can avoid the difficulty of modeling the imperfect control and the system-environment interaction. However, one possible weakness of these optimization methods is that they generate new trials only by looking at a fixed number of previous ones and often they need to restart once they reach a local minimum. Thus, in the long run, they do not fully utilize all the data generated by the experiments.

In this work we propose an orthogonal approach, where we try to mimic the structure of good parameters by building a model that approximates the probability distribution of these parameters. After an initial optimization, this model can then be used to efficiently generate new possible trials and can be continuously updated based on new data. In particular, based on the problem we attempt to solve, we choose this model to be a variant of the recurrent neural network (RNN), which makes our approach very similar to the way in which natural languages or handwriting are currently modeled. This ansatz enables us to exploit the models and insights developed by the machine learning community and possibly translate further progress there into the field of quantum control. It is worth pointing out that the machine learning part of this work is purely classical; only the (classical) data are related to quantum time evolution. Among the previous works, the approach by Wigley *et al.* in [18] is the most similar to ours, as they attempt to build a model from the data and utilize the model to perform optimization. Classical machine learning is also used in [19–21] to characterize the error models in quantum error correction and to react accordingly.

To demonstrate the feasibility of using our method to help optimize quantum memory, we consider the problem of automatically learning and optimizing dynamical decoupling sequences (almost) without using any prior knowledge. Dynamical decoupling (DD) [22] is a technique that combats certain noise by applying a sequence of unitary operations on the system (see [23,24] for a review). It has a less stringent requirement compared to general error correction protocols, which allows it to be demonstrated in experiments [15,23,25] in contrast to other methods. Moreover, known classes of good DD sequences have a relatively simple and well-defined structure. Based on the assumption that this holds true also for yet unknown and possibly better classes of sequences, it is

*august@in.tum.de
†xiaotong.ni@mpq.mpg.de

conceivable that a learning algorithm could eventually sample them without the need of using heavy mathematics.

To clarify, we do *not* attempt to solve the following two questions.

First, what do RNNs try to learn? It is known that RNNs can incorporate both short- and long-range correlation, which is desirable in our case, but it is unclear which one the gradient training method prioritizes. Indeed, it is an ongoing study to understand the behavior of RNNs [26]. Nevertheless, we choose to use RNNs since there are heuristic arguments on the advantage of them compared to similar models [27] and they benefit greatly from modern machine learning libraries and hardware.

Second, what is the optimal machine learning algorithm to find the best DD sequences? It is clear that we cannot claim our algorithm is the best one as there is not much theoretical understanding on RNNs. Indeed, the present authors believe there is much room for improvement, possibly by using better heuristics or taking into account more prior knowledge of DD. However, our work demonstrates that with a general model and a small amount of human effort, we can already achieve nontrivial results for certain problems.

II. BACKGROUND

A. Dynamical decoupling

The majority of dynamical decoupling schemes are designed for error models where the system-environment interaction can be described by a Hamiltonian. We will use $\mathcal{H}_S$ and $\mathcal{H}_B$ to denote the Hilbert space of the system and environment (often called bath), respectively. The difference between system and environment is that the former represents the part of the Hilbert space we can apply the Hamiltonian on and in which we store quantum information. The total noise Hamiltonian is

$$H_0 = H_S \otimes I_B + I_S \otimes H_B + H_{SB}.$$

Without intervention, in general H_0 would eventually destroy the quantum states we store in $\mathcal{H}_S$. To suppress this noise, we could apply a time-dependent Hamiltonian $H_C(t)$ to the system, which makes the total Hamiltonian $H(t) = H_0 + H_C(t)$. In the ideal case, we can control $H_C(t)$ perfectly and reach very high strength (i.e., norm of the Hamiltonian), which allows the ideal pulse

$$V(t) = O\delta(t - t_0).$$

It applies a unitary operator e^{-iO} to the system for an infinitely small duration (we set $\hbar = 1$ in this work). A very simple DD scheme for a qubit (a two-level system S) is the XY_4 sequence: It applies pulses of the Pauli matrices X and Y alternatingly with equal time interval τ_d in between. A complete cycle consists of four pulses $XYXY$, thus the total time period of a cycle is $T_c = 4\tau_d$. In the limit of $\tau_d \to 0$, the qubit can be stored for an arbitrarily long time. The intuition behind DD sequences is the average Hamiltonian theory. Let $U_C(t) = \mathcal{T}\exp\{-i\int_0^t dt' H_C(t')\}$ be the total unitary applied by $H_C(t')$ up to time t. In the interaction picture defined by $U_C(t)$, the dynamics is governed by the Hamiltonian $\tilde{H}(t) = U_C^\dagger(t) H_0 U_C(t)$. If the time interval τ_d between pulses is much smaller than the time scale defined by the norm of $\|H_0\|$, it is reasonable to consider the average of $\tilde{H}(t)$ within a cycle. The zeroth-order average Hamiltonian in T_c (with respect to τ_d) is

$$\bar{H}^{(0)} = \frac{1}{T_c}\int_0^{T_c} dt' U_C^\dagger(t) H_0 U_C(t).$$

For the XY_4 sequences introduced above, it is easy to compute $\bar{H}^{(0)} = \frac{1}{4}\sum_{\sigma\in\{I,X,Y,Z\}} \sigma H_0 \sigma$. Since the mapping $O \to \sum_{\sigma\in\{I,X,Y,Z\}} \sigma O \sigma$ maps any 2×2 matrix to 0, by linearity we know $\bar{H}^{(0)} = 0$.

Here we are going to list several classes of DD sequences. We will first explain how to concatenate two sequences, as most long DD sequences are constructed in this manner. Given two DD sequences $A = P_1 \cdots P_m$ and $B = Q_1 \cdots Q_n$, the concatenated DD sequence $A[B]$ is

$$A[B] = (P_1 Q_1) Q_2 \cdots Q_n (P_2 Q_1) Q_2 \cdots Q_n \cdots (P_m Q_1) Q_2 \cdots Q_n.$$

As an example, when we concatenate the length-2 and length-4 sequences XX and $XYXY$, we obtain $IYXYIYXY$. For convenience, we will use CDD to denote these sequences. Note that originally CDD is used to denote sequences generated solely from recursively concatenating $XYXY$ with itself.

We will use P_i to represent any Pauli matrix X, Y, or Z and for $i \neq j$, $P_i \neq P_j$. The families of DD sequences can then be listed as follows: DD4, length-4 sequences $P_1P_2P_1P_2$; DD8, length-8 sequences $IP_2P_1P_2IP_2P_1P_2$; EDD8, length-8 sequences $P_1P_2P_1P_2P_2P_1P_2P_1$; CDD16, length-16 concatenated sequences; $DD4[DD4]$; CDD32, length-32 concatenated sequences $DD4[DD8]$ and $DD8[DD4]$; and CDD64, length-64 concatenated sequences $DD4[CDD16]$ and $DD8[DD8]$. Longer DD sequences can again be obtained by the concatenation of the ones listed above and in the ideal situation they provide better and better protection against the noise. However, with realistic experimental capability, the performance usually saturates at a certain concatenation level. Since at this moment we are only optimizing short DD sequences, the listed ones are sufficient to provide a baseline for our purpose. One important family we did not include here is the Knill DD (KDD) [28], because it requires the use of non-Pauli gates.

However, we cannot expect these requirements to be met in all real-world experiments. The two major imperfections that are often studied are the flip-angle errors and the finite duration of the pulses. Flip-angle errors arise from not being able to control the strength and time duration of $H_C(t)$ perfectly, thus the intended pulse $V(t) = O\delta(t)$ becomes $V(t) = (1 \pm \epsilon)O\delta(t)$. Also, since zero-width pulses $O\delta(t)$ are experimentally impossible, we must consider finite-width pulses that approximate the ideal ones. In this paper we will only consider the imperfection of finite-width pulses. However, it is straightforward to apply our algorithm to pulses with flip-angle errors.

B. Measure of performance

There are multiple ways to quantify the performance of DD sequences. In practice, we choose different measures to

suit the intended applications. Here we use the same measure as in [24,29], which has the advantage of being (initially) state independent and having a closed formula for numerical simulation:

$$D(U,I) = \sqrt{1 - \frac{1}{d_S d_B}\|\mathrm{Tr}_S(U)\|_{\mathrm{Tr}}},$$

where U represents the full evolution operator generated by $H(t)$, d_S and d_B are the dimensions of the system and environment Hilbert space $\mathcal{H}_S$ and $\mathcal{H}_B$, respectively, $\|X\|_{\mathrm{Tr}} = \mathrm{Tr}(\sqrt{X^\dagger X})$ is the trace norm, and $\mathrm{Tr}_S(\cdot)$ is the partial trace over $\mathcal{H}_S$. The smaller $D(U,I)$ is, the better the system preserved its quantum state after the time evolution. For example, the ideal evolution $U = I_S \otimes U_B$ has the corresponding $D(U,I) = 0$.

In experiments, it is very hard to evaluate $D(U,I)$, as we often do not have access to the bath's degree of freedom. Instead, the performance of DD sequences is often gauged by doing process tomography for the whole time duration where DD is applied [25,30]. Although it is a different measure compared to our choice above, the optimization procedure can still be applied as it does not rely on the concrete form of the measure. Moreover, for solid state implementations such as superconducting qubits or quantum dots, a typical run of initialization, applying DD sequences and measurements, can be done on the time scale of 1 ms or much faster. Thus, it is realistic that on the time scale of days we can gather a large data set of DD sequences and their performance, which is needed for our algorithm.

C. Recurrent neural networks

Sequential models are widely used in machine learning for problems with a natural sequential structure, e.g., speech and handwriting recognition, protein secondary structure prediction, etc. For dynamical decoupling, not only do we apply the gates sequentially in the time domain, but also the longer DD sequences are often formed by repetition or concatenation of the short ones. Moreover, once the quantum information of the system is completely mixed into the environment, it is hard to retrieve it again by DD. Thus, an educated guess is that the performance of a DD sequence largely depends on the short subsequences of it, which can be modeled well by the sequential models.

Since our goal is not simply to approximate the distribution of good dynamical decoupling sequences by learning their structure but to sample from the learned distribution to efficiently generate new good sequences, we will further restrict ourselves to the class of generative sequential models. Overall, these models try to solve the following problem: Given $\{x_i\}_{i<t}$, approximate the conditional probability $p(x_t|x_{t-1},\ldots,x_1)$. As a simple example, we can estimate the conditional probability $p(x_t|x_{t-1})$ from a certain data set and use it to generate new sequences.[1] For more sophisticated problems (e.g., natural language or handwriting), it is not enough to only consider the nearest-neighbor correlations as simple models like Markov chains of order one do.

The long short-term memory (LSTM) network, a variation of the RNN, is a state-of-the-art technique for modeling longer correlations [32] and is comparably easy to train. The core idea of RNNs is that the network maintains an internal state in which it encodes information from previous time steps. This allows the model to, at least theoretically, incorporate all previous time steps into the output for a given time. Some RNNs have even been shown to be Turing complete [33]. In practice, however, RNNs often can only model relatively short sequences correctly due to an inherently unstable optimization process. This is where LSTMs improve over normal RNNs, as they allow for training of much longer sequences in a stable manner. Furthermore, LSTMs, like all ANNs, are based on matrix multiplication and the elementwise application of simple nonlinear functions. This makes them especially efficient to evaluate.

Algorithm 1. Optimization algorithm.

Input: Number of initial models to train: n
Number of models to keep: k
Percentage of data to keep: p
Set of possible topologies: $\mathcal{M}$
Size of data: d
$D \leftarrow$ `generateRandomData` (d)
$D, \langle \varsigma_s \rangle \leftarrow$ `keepBestData` (D,p)
$M \leftarrow$ `trainRandomModels` $(n,D,\mathcal{M})$
$M \leftarrow$ `keepBestKModels` (M,k)
while $\langle \varsigma_s \rangle$ *not converged* do
  $M \leftarrow$ `trainBestModels` (D)
  $D \leftarrow$ `generateDataFromModels` (M,d)
  $D, \langle \varsigma_s \rangle \leftarrow$ `keepBestData` (D,p)
end
Output: $\langle \varsigma_s \rangle, D, M$

From the machine learning perspective, we treat the problem at hand as a supervised learning problem where we provide the model with examples that it is to reproduce according to some error measure. It is also possible to formulate our problem in the framework of reinforcement learning. However, since we only compute the performance of a whole DD sequence, there is no immediate reward when choosing a gate in the middle of the sequence. Given the length of the sequences we are optimizing, it is likely a reinforcement learning algorithm will need help from certain (un)supervised learning, similar to the way in [3]. A short introduction to machine learning, LSTMs, and their terminology can be found in the Appendixes. More exhaustive discussions can be found in [34–36].

III. ALGORITHM

The algorithm presented in this section is designed with the goal in mind to encode little prior knowledge about the problem into it, in order to make it generally applicable to different imperfections in the experiment. Following this idea, the method is agnostic towards the nature of the considered gates, the noise model, and the measure of performance. To implement this, the algorithm assumes that the individual

[1]This idea can be at least dated back to Shannon [31], where this model generated "English sentences" like "On ie antsoutinys are t inctore st be s deamy achin d ilonasive tucoowe at ...."

gates are represented by a unique integer number such that every sequence $s \in \mathcal{G}^{\otimes L_s}$, with $\mathcal{G}$ denoting the set of unique identifiers and L_s being the length of s, and it is provided with a function $f(s)$ to compute the score ς_s of a given sequence s, taking into account the noise model. The optimization problem we want to solve is

$$\min_s f(s) = \min_s \varsigma_s.$$

By assumption, we have no information about f but can efficiently evaluate it. We furthermore assume the set of good sequences to exhibit common structural properties that can be learned well by a machine learning model. So we propose to solve it indirectly by training a generative model $m \in \mathcal{M}$ to approximate the distribution of good sequences, $\mathcal{M}$ being the set of possible models. That means we assume $s_t \sim p_m(s_{t-1}, \ldots, s_1)$, with s_t being the gate at time t and p_m denoting the distribution learned by m. Then we want to find an optimal m that ideally learns a meaningful representation of the structure of good sequences. In this work we choose the type of model to be the LSTM. We now tackle this surrogate problem by alternatingly solving

$$\max_{m \in \mathcal{M}} \mathcal{L}(m|T),$$

where $\mathcal{L}$ denotes the likelihood and T the training data, and then sampling sequences from the model m to generate a new T consisting of better solutions. The algorithm hence consists of two nested optimization loops, where the inner loop fits a number of LSTMs to the current data while the outer loop uses the output of the inner loop to generate new training data. This scheme of alternatingly fixing the data to optimize the models and consecutively fixing the models to optimize the data resembles the probabilistic model building genetic algorithm [37] and to some extent the expectation-maximization algorithm [38]. The method is shown in Algorithm 1. Partial justification of this heuristic algorithm is given in Appendix C. However, it is easy to see that the algorithm will not always find the global optimum. For example, it is conceivable that for certain problems the second to the 100th best solutions share no common structure with the first one. In that case, it would be unlikely for the machine learning approach to find the optimal one. There is however likely no universal method to bypass this obstruction, as unless we know the best sequences already, it is impossible to verify that they exhibit some structure similar to the training sets. This obstruction seems natural since many optimization problems are believed to be computationally hard. Thus, we should not assume to be able to solve them by the above routine.

We will now explain the most important aspects of the algorithm in more detail.

(a) Choice of LSTMs. The data we want to generate in our application are of sequential nature. This makes employing LSTMs an obvious choice as they pose one of the most powerful models available today for sequential data. Furthermore, the known well-performing families of DD sequences are constructed by nested concatenations of shorter sequences and hence show strong local correlations as well as global structure. Long short-term memories and especially models consisting of multiple layers of LSTMs are known to perform very well on such data and should therefore be able to learn and reproduce this multiscale structure better than simpler and shallow models.

(b) Generation of the initial training data. The size d and the quality, i.e., the percentage p of the initial data to be kept, are the parameters that we can specify. The data are then generated by sampling a gate from the uniform distribution over all gates for each time step. The average score of the initial data can then be used as a baseline to compare against in case no other reference value is available. We would like to point out that in the application considered in this work, an alternative way to generate the initial data might be to use the models trained on shorter sequences. This approach could lead to an initial data set with much higher average score, but at the price of introducing the bias from the previously trained RNNs.

(c) Training of the LSTMs. To reduce the chance of ending up in a bad local optimum, for each training set several different architectures of LSTMs are trained (see Appendix D 2 for detailed description of LSTMs). These models are independently sampled $\mathcal{M}$. More precisely, for the first generation of models, we sample a larger set of n models from $\mathcal{M}$ and train them. We then select the best k models and reuse them for all following generations. While it might introduce some bias to the optimization, this measure drastically reduces the number of models that need to be trained in total. The training problem is defined by assuming a multinoulli distribution over the gates of each time step and minimizing the corresponding negative log-likelihood $-\sum_t \delta_{s_t,i} \log_2 p_{m,i}(s_{t-1}, \ldots, s_1)$, where i is the index of the correct next gate, $p_{m,i}$ is its predicted probability computed by the LSTM m, and $\delta_{s_t,i} = 1$ if and only if $s_t = i$. This error measure is also known as the cross entropy. To avoid overfitting, we use a version of early stopping where we monitor the average score $\langle \varsigma_s \rangle_{p_m}$ of sequences generated by m and stop training when $\langle \varsigma_s \rangle_{p_m}$ stops improving. We employ the optimizer Adam [39] for robust stochastic optimization.

(d) Selecting the best models. As we employ early stopping based on the average score $\langle \varsigma_s \rangle_{p_m}$, we also rank every trained model m according to this measure. One could argue that ranking the models with respect to their best scores would be a more natural choice. This however might favor models that actually produce bad sequences but have generated a few good sequences only by chance. Using $\langle \varsigma_s \rangle_{p_m}$ is hence a more robust criterion. It would of course be possible to also consider other modes of the p_m, such as the variance or the skewness. These properties could be used to assess the ability of a model to generate diverse and good sequences. We find, however, that the models in our experiments are able to generate new and diverse sequences, thus we only use the average score as a benchmark for selecting models.

(e) Generation of the new training data. The selected models are used to generate d new training data by sampling from p_m. This is done by sampling s_t from $p_i(s_{t-1}, \ldots, s_1)$ beginning with a random initialization for $t = 1$ and then using s_{t-1} as input for time step t. We combine the generated sequences with the previous training sets, remove any duplicates, and order the sequences by their scores. We then choose the best p percent for the next iteration of the optimization. This procedure ensures a monotonic improvement of the training data. Note that all selected models contribute equally many data to strengthen the diversity of the new training data. A

possible extension would be to apply weighting of the models according to some properties of their learned distributions. Note though that ordering the generated sequences by their score is already a form of implicit weighting of the models.

IV. NUMERICAL RESULTS

A. Noise model and the control Hamiltonian

Throughout the paper we will use the same noise model as in [24]. We consider a 1-qubit system and a 4-qubit bath, namely, $\dim(\mathcal{H}_S) = 2$ and $\dim(\mathcal{H}_B) = 16$. The small dimension of the bath is for faster numerical simulation and there is no reason for us to think that our algorithm would only work for a small bath as the size of the bath enters the algorithm only via the score-computation function. The total noise Hamiltonian consists of (at most) three-body interactions between the system and bath qubits with random strength

$$H_0 = \sum_{\mu\in\{I,X,Y,Z\}} \sigma^\mu \otimes B_\mu, \tag{1}$$

where σ^μ is summed over Pauli matrices on the system qubit and B_μ is given by

$$B_\mu = \sum_{i\neq j}\sum_{\alpha,\beta} c^\mu_{\alpha\beta}\left(\sigma_i^\alpha \otimes \sigma_j^\beta\right),$$

where i,j is summed over indices of the bath qubits and $\sigma_i^{\alpha\ (\beta)}$ is the Pauli matrix on qubit i of the bath. We consider the scenario where the system-bath interaction is much stronger than the pure bath terms. More precisely, we set $c^\mu_{\alpha\beta} \approx 1000 c^I_{\alpha\beta}$ for $\mu \in \{X,Y,Z\}$. Apart from this constraint, the absolute values $|c^\mu_{\alpha\beta}|$ are chosen randomly from a range $[a,b]$, where we set $b \approx 3a$ to avoid too many terms vanishing in (1). The result Hamiltonian has a 2-norm $\|H_0\| = 20.4$.

For the control Hamiltonian, we consider the less explored scenario where the pulse shape have finite width but no switch time between them (100% duty cycle). In other words, the control Hamiltonian is piecewise constant

$$H_C(t) = H_k \quad \text{for} \quad k\tau_d \leqslant t < (k+1)\tau_d,$$

where τ_d is a small time period with respect to the norm of H_0 and $e^{-iH_k\tau_d} \in \{I,X,Y,Z\}$. This is a good toy model for experimental settings whose DD performance is mainly limited by the strength of the control Hamiltonian, but not the speed of shifting between Hamiltonians. Since this regime is less explored in theoretical studies, it is an interesting scenario to explore via machine learning. Another restriction we put on $H_C(t)$ is

$$H_C(t) = -H_C(T-t),$$

where T is the total evolution time. This condition ensures $U_C(T) = \mathcal{T}\exp\{-i\int_0^T dt' H_C(t')\} = I$ and it allows us to apply the same code on the setting where the system has more than one qubit. It is known that this family of symmetric Hamiltonians can remove the first-order terms of τ_d in the average Hamiltonian [22,40]. So, strictly speaking, this should be counted as prior knowledge. However, when we compare the known DD sequences with the numerically found ones, we also use the symmetric version of the known DD sequences. Thus, we perform the comparison on equal terms.

B. Numerical experiments

In the following we present the results of a number of experiments we have conducted to evaluate the performance of our method. We consider sequences consisting of 32, 64, and 128 gates for varying values of τ_d. This translates to having to optimize the distribution of the first 16, 32, and 64 gates, respectively. To compute ς_s, we use the figure of merit D as defined in Sec. II A. Thus, a lower score is better. For $\mathcal{M}$, we consider models with two or three stacked LSTM layers followed by a final softmax layer. The layers comprise 20 to 200 units where layers closer to the input have a higher number of units. We allow for peephole connections and linear projections of the output of every LSTM layer to a lower number dimension [35]. The optimization parameters are also randomly sampled from sets of reasonable values. We choose the step rate to be in $\{10^{-1},10^{-2}\}$ and the batch size to take values in $\{200,500,1000\}$. The parameters specific to the Adam optimizer β_1, β_2, and ϵ, we sample from $\{0.2,0.7,0.9\}$, $\{0.9,0.99,0.999\}$, and $\{10^{-8},10^{-5}\}$, respectively. We perform a truncation of the gradients to 32 time steps in order to counter instabilities in the optimization (see Appendix D 3). As we have stated above, we also employ early stopping in the sense that, for every optimization of a model, we keep the parameters that generate the sequences with the best average score. The algorithm was run until either the best known score was beat or the scores converged, depending on the goal of the respective experiment. We will now briefly list the concrete experiment settings and discuss the results.

(i) Experiment E1: Length 32. In this first experiment, we considered sequences of 32 gates with $\tau_d = 0.002$. We let the algorithm train $n = 30$ models initially and set the number of models to be kept k to 5. We combined the data generated by the LSTMs with the previous training set after each generation and chose the best 10% as the new training data, consisting of 10 000 sequences for each generation. We let every model train for 100 epochs.

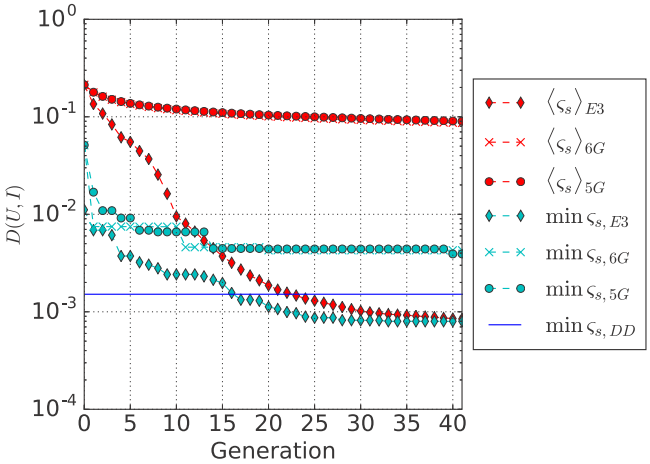
(ii) Experiment E2: Length 64. In our next experiment, we tackled a more difficult scenario with 64 gates and a larger $\tau_d = 0.004$. We set $n = 50$ and $k = 5$. Again, we used the best 10% of both generated and previous data as new training data, which consist of a total 10 000 sequences for each training set.

(iii) Experiment E3: Length 128. In the third experiment we tried our method on even longer sequences of 128 gates with τ_d again being 0.004. Due to the very large sequence space, we set the size of the training sets to 20 000, again using the best 10% of sequences generated by the selected models and the previous training set. The number of epochs was increased to 200. We set $n = 30$ and $k = 5$. Here we let the algorithm run until both the average and the best score converged to examine its behavior in long runs.

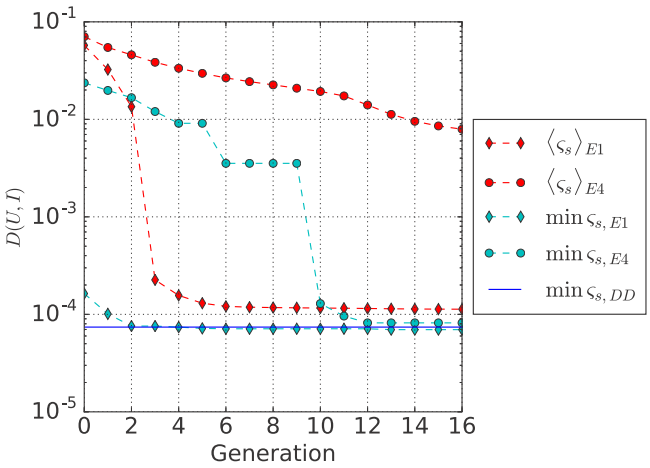
(iv) Experiment E4: Length 32 with random gates. Finally, we tested the performance of Algorithm 1 in the case where we replaced the Pauli gates $\{I,X,Y,Z\}$ with ten randomly chosen gates. More precisely, we chose each gate g_j to be a randomly generated single two-dimensional unitary operator with eigenvalues 1 and -1, i.e., $g_j = U_j^\dagger X U_j$, where U_j is a random unitary. All other parameters were kept as in experiment $E1$.

TABLE I. Comparison of the results obtained in experiments $E1$, $E2$, $E3$, and $E4$ to the best theoretically derived DD families. For each experiment, the average and best score of the last training data and the average score of the best model of the last generation are shown. They are compared to random sequences and the two DD classes that yield the best average and overall best score, respectively. The best results are printed bold.

Sequences	$\langle \varsigma_s \rangle$	$\min \varsigma_s$
	Experiment $E2$	
EDD8	0.002398	0.002112
CDD32	0.053250	0.000803
last training set $E2$	**0.000712**	**0.000381**
best model $E2$	0.016692	
random	0.341667	
	Experiment $E3$	
EDD8	0.004793	0.004222
CDD64	0.031547	0.001514
last training set $E3$	**0.000827**	**0.000798**
best model $E3$	0.029341	
random	0.44918	
	Experiments $E1$ and $E4$	
EDD8	0.000151	0.000133
CDD16	0.010699	0.000074
last training set $E1$	**0.000112**	**0.000070**
last training set $E4$	0.007178	0.000082
best model $E1$	0.003089	
random	0.125371	



(a) Experiment E3



(b) Experiments E1 and E4

FIG. 1. Convergence of the algorithm in (a) $E3$ compared to the case where LSTMs are replaced by 5/6-gram models and (b) $E1$ compared to $E4$ as both consider the same problem setting. In (a) it is clearly visible that LSTMs outperform the n-gram models, while (b) reflects the physical knowledge that the Pauli unitaries are a better choice than random gates. Note that the red (dark) crosses in (a) are almost covered by the red (dark) circles. As a reference, we show the score of the best DD sequence obtained from the known DD classes.

In Table I we compare the last training data and the best model of the last generation of $E1$–$E4$ against the two DD families that achieve the best average and minimal scores for the given experiment, respectively. We also plot the convergence of the training data of $E3$ and $E1$ with $E4$ in the Figs. 1(a) and 1(b), respectively. In general, the results for $E1$, $E2$, and $E3$ clearly show that our method outperforms DD, achieving a better minimal score of the generated data in a moderate number of iterations and with a relatively small set of models. The results of $E4$ will be discussed below. These findings indicate that our method converges to good local optima and that the models are able to learn a meaningful internal representation of the sequences that allows for efficient sampling of good sequences. There is however a noticeable gap between the scores of the training data and the models. A possible remedy for this could be an increase of the training data size or an adjustment of the model parameters in later stages of the optimization to account for the change in the structure of the data.

To assess the importance of LSTMs for the performance of our algorithm, in experiment $E3$, we also ran a different version of our method where we replaced the LSTMs by simple 5/6-gram models, which only model and generate sequences based on local correlations (see Appendix A 2 for the definition). The convergence plots in Fig. 1(a) show that LSTMs are indeed superior to the simpler models. They are able to improve the average and best scores faster and ultimately let the algorithm converge to a better local optimum. This advantage most likely results from the fact that the LSTM models are able to leverage information about longer-range correlations in the data. These results hence justify our choice of LSTMs as a machine learning model to optimize DD sequences.

We also compared the results of experiments $E1$ and $E4$ to examine the importance of using the Pauli group as the gate set. Figure 1(b) shows that while for $E1$ the average score quickly becomes very good and the best score exceeds the best known result after a few generations, in $E4$ the average score of the data improves much slower and remains significantly worse than that of $E1$. Although the best score exhibits a much stronger improvement, it eventually converges to a value slightly worse than that of the best theoretical DD sequence and the one found in $E1$. This is expected since with the Pauli group we can achieve first-order decoupling with DD sequences of length 4, which is the shortest. On the other hand, with random unitaries, in general it will take much longer sequences to have

approximate first-order decoupling, during which the system and environment can become fairly entangled.

Another interesting aspect to note is the rather strong improvement of the average scores occurring in $E3$ and $E1$ between generations 8 to 10 and 2 to 3, respectively. These jumps can be explained by the known existence of several strictly separate regimes in sequence space that differ strongly in their performance. The results indicate that our algorithm is able to iteratively improve the learned distributions to eventually capture the regime of very good sequences.

In order to verify that sampling the initial training data from the distributions learned for shorter sequences is a viable alternative to uniform sampling, we let the best model obtained in $E2$ generate an initial data set for the problem setting of $E3$. The obtained data were found to have an average score of 0.037 175, which is about one order of magnitude better than the average of the initial training data generated by uniform sampling.

V. CONCLUSION

We have introduced a method for optimizing dynamical decoupling sequences that differs from previous work by the ability to utilize much larger data sets generated during the optimization. Its ability to efficiently generate large sets of good sequences could be used along with other optimization methods to cover their weaknesses or to perform statistical analysis of these sequences. We showed that for certain imperfect control Hamiltonians, our method is able to outperform (almost all) known DD sequences. The little prior knowledge about DD we use is (i) choosing Pauli operators as pulses in the sequences (see experiment $E4$ and its discussion), (ii) choosing specific lengths for the DD sequences, and (iii) enforcing the reversal symmetry, as discussed in Sec. IV A. However, we do not need to initialize the data set in a specific way as in Appendix C 5 a of [24], which actually contains a certain amount of prior knowledge of DD. Also, our method does not fundamentally rely on the prior knowledge stated above. It is conceivable that the use of this prior knowledge can be lifted, at the price of a possibly much slower optimization procedure. For example, the KDD scheme helps to further increase the performance of CDD sequences in some experiments [23]. Thus, an interesting question is when given the freedom of applying non-Pauli gates and choosing variable lengths of the sequences, whether our algorithm could discover a similar strategy. Thus, a possible direction of future research is to see how we can minimize the slowdown when not incorporating any prior knowledge and whether we can obtain good DD sequences with non-Pauli pulses.

While we have applied the algorithm to the case of quantum memory and compared it to dynamical decoupling, it is of general nature. It can in principle be applied to every problem where the optimization of a sequence of gates with respect to some well-defined figure of merit is desired and where it is feasible to evaluate this performance measure for larger numbers of sequences. However, due to the nature of the underlying machine learning model, good results will likely only be obtained for problems whose solution depends strongly on local correlations in the sequences.

ACKNOWLEDGMENTS

We would like to thank Geza Giedke for helpful discussions and comments on the draft. The idea of this paper partially stems from a discussion between Courtney Brell and X.N. about using genetic algorithms to optimize quantum memory. We would also like to thank Peter Wittek for helpful comments. M.A. acknowledges funding by the Elite Network of Bavaria via the doctoral programme "Exploring Quantum Matter."

M.A. and X.N. are contributed equally to this work.

APPENDIX A: ANALYSIS

1. Local correlations of DD sequences

As we suggested earlier, the reason we use RNNs as the probabilistic model is that the performance of dynamical decoupling sequences heavily depends on their local correlations. To illustrate this fact, we can count the frequency of length-2 (see Table II) (or length-3) subsequences from the training set of the 30th generation in experiment 3. We can then compare these statistics to the ones of the sequences generated by the LSTM, which is trained based on the training set. We can see indeed that the percentages match very well. To get more detail about local correlations, we could also count the frequency of length-3 subsequences (see Table III). Note that since the table is based on the data sets in the late stage of the optimization, the distribution of the subsequences are already very polarized. However, we observe the same behavior (the percentages matches well) in other experiments at different stages of the optimization as well. However, RNNs do not only take into account local correlations, as we show in Fig. 1 that they perform better compared to the n-gram models, which we will introduce in the next section.

2. The n-gram models

n-grams are the simplest sequential models that treat the sequences as stationary Markov chains with order $n-1$.

TABLE II. Frequency of length-2 subsequences, from the training set and the set generated by the trained LSTM (given in parentheses) at generation 30 of experiment 3. The total number of subsequences is around 1.2×10^6.

Previous gate \ Next gate	I	X	Y	Z
I	0.00% (0.00%)	0.04% (0.08%)	0.15% (0.68%)	0.02% (0.08%)
X	0.05% (0.22%)	5.38% (5.04%)	30.53% (30.47%)	1.39% (1.26%)
Y	0.07% (0.20%)	30.17% (30.47%)	18.40% (18.61%)	5.84% (5.50%)
Z	0.01% (0.02%)	1.90% (1.68%)	5.75% (5.42%)	0.30% (0.27%)

TABLE III. Frequency of length-3 subsequences started with gate X, from the training set and the set generated by the trained LSTM (given in parentheses) at generation 30 of experiment 3. The total number of subsequences started with X is around 450 000.

Second gate \ Last gate	I	X	Y	Z
I	0.00% (0.00%)	0.02% (0.05%)	0.12% (0.55%)	0.00% (0.01%)
X	0.00% (0.00%)	1.40% (1.22%)	11.99% (11.52%)	0.32% (0.32%)
Y	0.15% (0.47%)	44.79% (45.09%)	33.39% (33.54%)	4.11% (3.85%)
Z	0.01% (0.01%)	2.38% (2.14%)	1.05% (0.98%)	0.28% (0.26%)

Operationally, given a set of sequences, we first estimate the conditional probability distribution

$$p_{x_n,x_{n-1}\cdots x_1} = \Pr(X_t = x_n | X_{t-1} = x_{n-1}, \ldots X_{t-n+1} = x_1).$$

Note that we assume the conditional probability is independent of t (hence stationary Markov chain). The estimation is done by counting over the whole set of sequences. The generation of new sequences based on the conditional probability $p_{x_n,x_{n-1}\cdots x_1}$ is straightforward, as we can repeatedly sample from it based on the previous $n-1$ items. This behavior is different compared to that of the RNNs, which have memory units that can store information for an arbitrarily long time in theory.

3. Optimization without reusing data from previous training sets

During the optimization processes in the main text, we always reuse the data from previous training sets, in the sense that we first add the new sequences generated by the models to the training sets and then delete the worst sequences. An interesting question is what will happen if we generate new training sets completely from the trained models. In Fig. 2 we plot the counterpart of Fig. 1(a) with this modification (as well as not deleting duplicated sequences from the training set). We can see that for the LSTMs experiment, the final minimum score gets slightly worse, which is 0.000 874. However, the 5/6-gram experiments actually perform better when not reusing data. While it seems counterintuitive, this can be possibly explained by the fact that in the case of reused data with unique sequences the higher diversity of the data might make it harder for the models to find local correlations, which then in turn slows down the optimization. There is other interesting information contained in the plot. For example, we can see the minimum scores almost always decrease, which implies that the LSTMs are able to learn new information about good sequences in most generations.

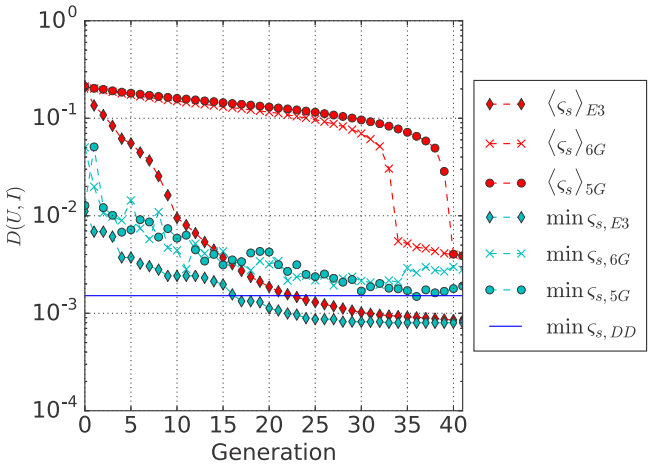


FIG. 2. The 3- and 5/6-gram experiments without data reusage. Otherwise, the experiments are done in the same way as in Fig. 1(a).

4. Performance of the obtained sequences with a larger heat bath

In the main text, all the numerical simulations are done on a randomly generated noise Hamiltonian with the dimension of the bath being $\dim(\mathcal{H}_B) = 16$. The small dimension of the bath is used in order to have a fast simulation. Here we test the performance of some obtained sequences from experiment 2, in the presence of a larger bath with $\dim(\mathcal{H}_B) = 128$. Apart from the change of dimension, the Hamiltonian H_0 is again randomly generated according to the description in Sec. IV A, which has a 2-norm $\|H_0\| = 24.0$. We then computed the scores of the top 500 DD sequences in the last generation of experiment 2. The results are shown in Table IV. While the best score of the obtained sequences is worse than the best score of CDD32, it is clear that, on average, the obtained sequences still work fairly well. This also suggests that our algorithm is potentially capable of adapting to the particular noise Hamiltonian, as the learned sequences outperform known DD families in experiment 2.

APPENDIX B: BEST SEQUENCES

We list here the best sequences we found in experiments 1–3 from the numerical results section. We denote the identity by I and X, Y, Z refer to the respective Pauli matrices. Note that we show only the first half of the complete sequence as the second one is just the first half reversed. In experiment 1 we found $X,Y,X,Z,X,Y,X,Z,Z,X,Y,X,Z,X,Y,X$; in experiment 2, $Z,Z,X,Z,Z,Z,X,Z,Z,X,Z,X,X,X,Z,X,X,X,Z,X,X,Z,X$,

TABLE IV. Comparison between the scores of the top 500 DD sequences in the last generation of experiment 2 and some DD families for the larger bath $\dim(\mathcal{H}_B) = 128$. The best score of the 500 sequences is worse than the best score of CDD32. However, it is clear that, on average, the obtained sequences still work fairly well.

Sequences	$\langle \varsigma \rangle$	$\min \varsigma$
EDD8	0.002781	0.002203
CDD32	0.053753	0.000432
top 500	0.001081	0.000626

X,X,Z,X,Z,Z,X,Z,Z; and in experiment 3, $Z,X,Z,Z,Y,X,Y,Z,Y,X,Y,X,Y,Y,X,Y,Y,Y,X,Y,Y,Y,X,Y,Y,X,Y,X,Y,X,Y,Y,Z,X,Z,Y,Z,X,Z,Y,X,Y,X,X,Y,X,Y,X,Y,X,Y,Y,X,Y,Y,Y,X,Y,X,X,Y,X,X$.

APPENDIX C: COMPARISON OF OPTIMIZATION ALGORITHMS

In this appendix we will give a comparison between several optimization algorithms applied to black-box problems. In other words, the algorithm needs to optimize (minimize) the objective function f only by looking at the values of $f(x)$ (without knowing the concrete formula of it). We are going to look at the following types of algorithms: gradient-based algorithms (when we can access the gradient of f), e.g., Newton's method, variants of gradient descent; Metropolis-Hasting algorithms and its variants, e.g., simulated annealing; and genetic algorithm and its variants, e.g., a probabilistic model building genetic algorithm (PMBGA). The performance of an optimization algorithm depends heavily on the class of the problems it is applied to. (This fact is remotely related to the no free lunch theorem for optimization). Thus, in the following, we will use different objective functions to illustrate the strong and weak points of those algorithms.

1. Gradient-based algorithms

To understand the idea of these algorithms, it is enough to consider $f:\mathbb{R}\to\mathbb{R}$ defined on a single variable. The simplest gradient descent for finding the minimum of f is the following iterative algorithm: starting from a random number x_0 and successively computing $x_{n+1}=x_n-\alpha f'(x_n)$. Gradient-based algorithms perform well on functions with nonvanishing gradients almost everywhere and very few local minima and likely have a poor performance otherwise. For example, the above algorithm would perform very well on a simple function $f(x)=x^2$, but much worse on the fast oscillating function

$$f(x)=\sin(8x)+0.5\sin(4x)+0.3\sin(2x)+0.1\sin(x). \tag{C1}$$

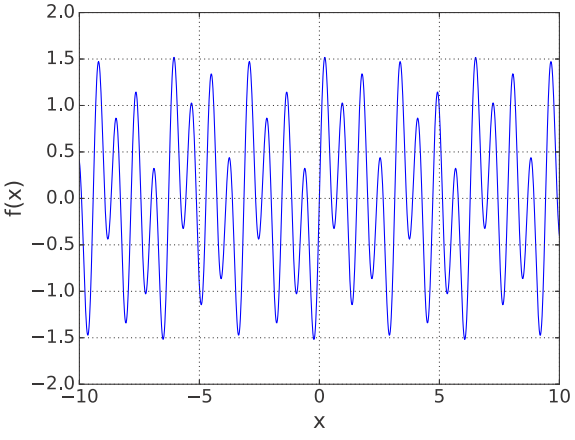


FIG. 3. Plot of the function (C1).

We plot the above function in Fig. 3. It is easy to see that we can construct $f(x)=\sum_{i=1}^{N}a_i\sin(2^i x)$ such that the chance of finding the global minimum is arbitrarily small.

2. Simulated annealing

Simulated annealing (SA) and its variants stem from the Metropolis-Hastings algorithm. The main idea is constructing a family of probability distribution $p(x,T)$ based on the values of the objective function $f(x)$, with the requirement $p(x,0)>0$ only when x is a global minimum of f. Then we repeatedly sample from $p(x,T)$ while slowly decreasing T. In practice, simulated annealing is also an iterative algorithm, i.e., it chooses x_{n+1} based on x_n. Since SA uses the Metropolis-Hastings algorithm as a subroutine, there is a nonzero chance to choose x_{n+1} such that $f(x_{n+1})>f(x_n)$. So, in principle, SA could escape from local minima, which is an advantage compared to gradient descent. Simulated annealing also works for functions with discrete variables. As a trade-off, it is likely to be slower compared to gradient descent when f has very few local minima. Moreover, while SA has the mechanism to escape from local minima, in practice it could work poorly on functions with many local minima and high barriers between them, e.g., the function (C1).

3. Genetic algorithms and beyond

In this section we will assume that f has the form $f:\mathbb{R}^N\to\mathbb{R}$. A common feature in all versions of genetic algorithms is that they maintain a population of solutions $\{\vec{x}_i,1\leqslant i\leqslant M\}$, where $\vec{x}_i=(x_{i1},\ldots,x_{iN})$. For the first generation, a number of $M'>M$ solutions is randomly generated, then we pick the $\vec{x}_i$ with the M smallest $f(\vec{x}_i)$ as the population. To generate new potential solutions for new generations, several different operations are introduced. In the original genetic algorithm, the two such operations are crossover and mutation. The effect of the mutation operation on a solution $\vec{x}$ is

$$(x_1,\ldots,x_j,\ldots,x_N)\to(x_1,\ldots,x'_j,\ldots,x_N),$$

where x'_j is a random number. The crossover operation acts on two solutions $\vec{x}$ and $\vec{y}$,

$$(\vec{x},\vec{y})\to(x_1,\ldots,x_j,y_{j+1},\ldots,y_N),$$

where the position j is picked randomly. Then we can use these two operations to generate M'' new test solutions from the first generation, combine them with the M old solutions, and pick the top M solutions as the population of the second generation. Later generations can be obtained by repeating these steps.

To illustrate the advantage of the (original) genetic algorithm, we can consider the objective function f,

$$f(\vec{x})=\sum_j f_j(x_j).$$

In this case, if $f(\vec{x})$ is (relatively) small, then either $\sum_{j=1}^{k}f_j(x_j)$ or $\sum_{j=k+1}^{N}f_j(x_j)$ is (relatively) small. Thus the crossover operations serve as nonlocal jumps, while the mutation operations help to find local minimum. However, in general, it is not clear for what kind of function f the inclusion of the crossover operations could provide an advantage.

It is easy to construct counterexamples such that the crossover operations deteriorate the performance, such as

$$f(\vec{x}) = f(\vec{x}_a,\vec{x}_b) = \|\vec{x}_a - \vec{x}_b\|,$$

where $\vec{x}_a$ and $\vec{x}_b$ have equal dimension and $\|\cdot\|$ is the Euclidean norm. Clearly, in most cases, the crossover of two good solutions will only produce inferior new solutions.

It turns out that the most important feature of genetic algorithms is the use of a population. In comparison, other optimization methods we mentioned previously only keep track of the last test solution. If we are willing to believe that good solutions of the function f have a certain structure (thus partially dropping the black-box requirement of f), it is possible that we can identify this structure from the solutions in the population and then generate new test solutions. This idea has led to the so-called probabilistic model building genetic algorithm and its variants [37,41]. The optimization algorithm we introduced in the main text is also closely related to this idea.

Instead of going through the details of these algorithms, we will explain the idea using a simple example, as illustrated in Fig. 4. Suppose that we want to minimize a function $f(x,y)$ with two variables defined on a finite region of $\mathbf{R}^2$ and prior knowledge of f allows us to make the hypothesis h that all points $\{(x,y)\}$ with values $f(x,y) < M$ exist in a certain region A [e.g., the square in Fig. 4(a)]. By sampling random points from the domain of the function, we can verify or refute the hypothesis h. For simplicity, we assume that h is satisfied for all sampled points and N of them is inside the region; then the opposite hypothesis of an α fraction of points $\{(x,y)\}$ with values $f(x,y) < M$ existing outside the region A will give the observed data a likelihood of $(1-\alpha)^N$. Thus, we can just optimize f over the region A by ignoring a very small fraction of the good solutions. It is easy to see that we can iterate this process, as long as we can formulate a small number of hypotheses such that one of them will describe the good solutions correctly. Our algorithm in the main text resembles this toy example. However, for functions in high dimension and sophisticated generative models such as RNNs, it is hard to give a mathematical justification like in the above example.

It is natural to concatenate the above process [see Fig. 4(b)]. Let S_0 be the domain of f, and S_1 be the points in region A. By sampling enough points from S_1, we might be able to build a model and sample from a even smaller set S_2 with the good solutions (e.g., find a region $B \subset A$). This way we will introduce a series of sets $\{S_i\}_{i\leqslant K}$ that we can sample from. Assuming that the order of these subsets satisfies $|S_{i+1}| < \frac{1}{2}|S_i|$, then in the ideal scenario the above iterative algorithm would provide an exponential speed-up with respect to K. However, it is worth pointing out that automatically building a model from a data set is, in general, a difficult task (if possible at all).

As another concrete example we can consider the objective function (C1) and a routine that looks for the periodicity of the data and then generates new test solutions accordingly. After we go through multiple generations, it is likely that the population would converge to the correct periodic subset that has the minimum $f(x)$.

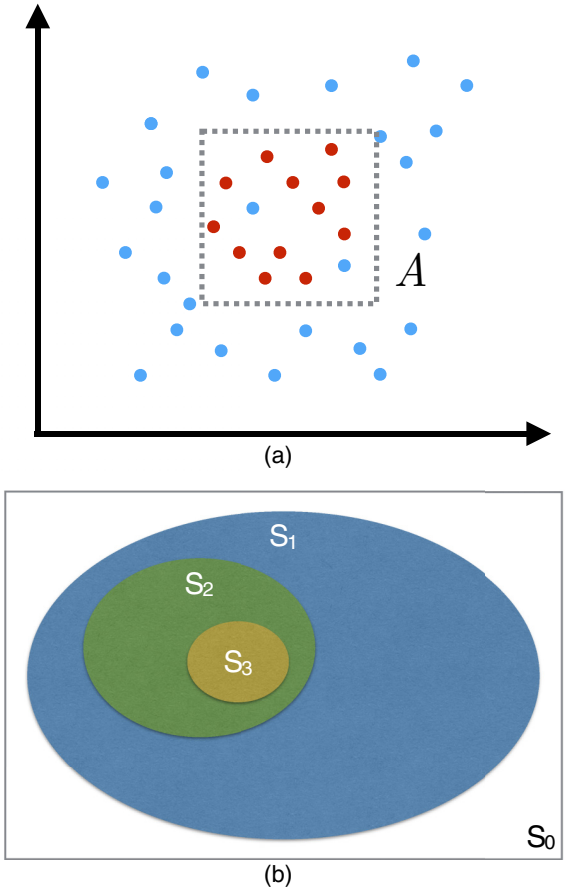


FIG. 4. Outline of our algorithm: (a) demonstrates that if we can model the distribution correctly, then we will be able to sample from good solutions more efficiently [red (darker) points correspond to smaller $f(x,y)$] and (b) illustrates the idea of concatenating the step performed in (a) in order to achieve an exponential speed-up compared to random search.

4. Summary

As seen in the discussion above, each of these optimization methods has its strong and weak points. Thus different methods are chosen depending on the prior knowledge we have on the concrete problems. It should be emphasized that we should not consider these methods as in a pure competition; instead, they can be used in complement with each other. For example, stochastic gradient Langevin dynamics (SGLD) [42] can be viewed as a combination of gradient descent and annealing, and in [43] it is mentioned that inclusion of the deterministic hill climber (discrete version of gradient descent) can lead to a substantial speed-up in the PMBGA.

APPENDIX D: MACHINE LEARNING

This section will give a brief overview over the subfield of machine learning known as supervised learning and introduce a model for time-series data, known as recurrent neural networks. Furthermore, some aspects of the optimization of this class of models will be elaborated on.

1. Supervised learning

The field of machine learning can be divided into three main subfields: supervised learning, unsupervised learning, and reinforcement learning. These branches differ from each other by the way in which the respective models obtain information about the utility of their generated outputs.

In the case of supervised learning, it is assumed that for every input that a model will be trained on, a "supervisor" provides a target, corresponding to the desired output of the model for the given input. These pairs of inputs and desired outputs are then used to make the model learn the general mapping between input and output.

More formally and from a Bayesian perspective, one assumes to have a data set D of size N, consisting of several tuples of independent and identically distributed observations $x \in \mathbf{R}^l$ and corresponding targets $y \in \mathbf{R}^k$ such that

$$D = \{(x_i, y_i)|_{i=1}^N\},$$

where x_i and y_i are instances of two random variables X and Y, respectively. These random variables are assumed to be distributed according to some unknown probability distribution p_{gen}, the so-called data-generating distribution

$$X, Y \sim p_{\text{gen}}(X, Y).$$

The goal of any supervised learning method now is to approximate the conditional distribution $p_{\text{gen}}(Y|X)$ in a way that allows for evaluation in some new observation $x_* \notin \{x_i\}|_{i=1}^N$. Since p_{gen} is not available, one resorts to fitting the empirical distribution p_{emp} given by D as a surrogate problem.

A typical way of deriving a concrete optimization problem from this is to make an assumption regarding the form of p_{gen} and treating the model at hand as a distribution $p_M(Y|X, \Theta)$ of this kind, parametrized by the parameters of the model Θ that are also often called the weights of the model. Now the fitting of the model can be perceived as a maximum-likelihood problem and hence the supervised learning problem can be formulated as

$$\max_{\Theta} \mathcal{L}(\Theta|D) = \max_{\Theta} \prod_i p_M(y_i|x_i, \Theta),$$

making use of the independent and identically distributed assumption. A commonly employed trick to obtain a more benign optimization problem is to instead optimize the negative log-likelihood. As the logarithm is a monotonic function, this transformation does not change the location of the optimum in the error landscape, but turns the product of probabilities into a sum over the tuples in D. This step then yields a minimization problem, given by

$$\min_{\Theta} -\frac{1}{N} \sum_i \log_2 p_M(y_i|x_i, \Theta),$$

which is also called empirical risk minimization (ERM). These statements of the problem can now be tackled with the optimization methods appropriate for the given model. In the case of the RNN, gradient-based optimization is the state-of-the-art approach and will be explained in Sec. V.

While it is obvious that fitting a model with respect to p_{emp} is identical to fitting it to p_{gen} as long as every tuple in D is only considered once, this is not necessarily true anymore when considering each tuple multiple times. This however is needed by many models in order to fit their parameters to a satisfying degree. In order to prevent the model from learning characteristics of the empirical distribution that are not present in the data-generating distribution, a phenomenon commonly known as overfitting, often some form of regularization, is applied. This may be done by punishing too large parameter values, stopping the training after performance starts to decrease on some holdout data set or by averaging over multiple models. Note that in the Bayesian picture some penalty terms can be perceived as the logarithm of a prior distribution over Θ, hence turning the optimization problem into finding the maximum *a posteriori* parameters.

2. Recurrent neural networks

In this section the recurrent neural network model will be discussed. We will start with an introduction of the standard version of the model and based upon this explain the advanced version of the model employed in this work in a second step.

a. Standard RNN model

In many areas of application, the data can be perceived as, often non-Markovian, discrete time-series data, such that an observation $x_t \in \mathbb{R}^l$ at some time t depends on the previous observations $x_{t-1}, \ldots, x_1$ or with respect to the framework introduced above,

$$X_t \sim p(X_t|X_{t-1}, \ldots, X_1).$$

While Markov chains have been the state-of-the-art approach for this kind of data in recent decades, with the recent rise of artificial neural networks, RNNs [44,45] have also gained momentum and are now generally considered to be the most potent method.

An RNN is defined by the two nonlinear maps $s_t : \mathbb{R}^l \to \mathbb{R}^h$ and $o_t : \mathbb{R}^h \to \mathbb{R}^o$ given by

$$s_t = f_s(Ux_t + Ws_{t-1} + b_s),$$
$$o_t = f_o(Vs_t + b_o),$$

where $U \in \mathbb{R}^{h\times l}$, $W \in \mathbb{R}^{h\times h}$, $V \in \mathbb{R}^{o\times h}$, $b_s \in \mathbb{R}^{1\times h}$, $b_o \in \mathbb{R}^{1\times o}$, and the trainable parameters of the models are constituted by $\Theta = \{U, V, W, b_s, b_o\}$. The nonlinear function f_s is often chosen to be tanh, the rectifier function given by

$$\text{rect}(x) = \max(0, x),$$

or the sigmoid function given by

$$\text{sigm}(x) = \frac{1}{1 + e^{-x}}.$$

The function f_o must be chosen according to the distribution that is to be approximated by the model. For the case of a multinoulli distribution as assumed in this work, the corresponding function would be the softmax, defined as

$$\text{softmax}(x)_j = \frac{e^{x_j}}{\sum_k e^{x_k}},$$

the superscripts in this case denoting the single elements of the vector x.

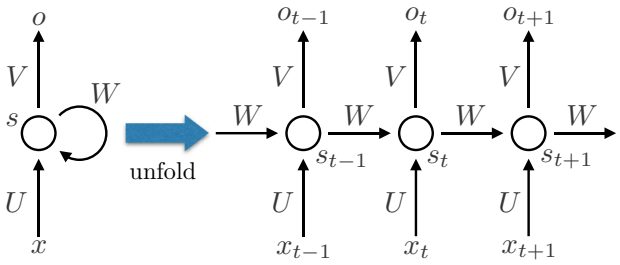


FIG. 5. Standard model of a recurrent neural network shown for three time steps.

The intuition behind this simple model is that it combines its information about the input at a given time step with a memory of the previous inputs, referred to as the state of the network. The precise nature of this combination and the state depends on the weight matrices U and V and the bias vector b_s. The combined information is then used as input of the chosen nonlinear function f_h to generate the next state. From this state, the output o_t is then computed as defined by W, b_o, and f_o. The effect of an RNN acting on the sequence $\{x_t\}$ is illustrated in Fig. 5.

From the above explanation, it is clear that the power of the model depends strongly on the size of the hidden state h. It should however also be noted that another effective way of increasing the expressive power of an RNN is to construct a composition of multiple functions of the form of s_t (see Fig. 6). In the machine learning terminology, the respective functions are called the layers of an artificial neural network and the number of composed functions is referred to as the depth of a network. The layers between the input and the output are referred to as hidden layers. The common intuitive reasoning behind stacking multiple layers is that it will allow the network to learn a hierarchy of concepts, called features, from the initial input data. Thereby, the features are assumed to be of increasing complexity with every layer, as they are based on a linear combination of the features learned by the layer below. Apart from this intuitive reasoning, also more rigorous work on the benefits of using at least one hidden layer between input and output can be found in the literature [46–48]. This ansatz of increasing the power of neural network models via deepening their architecture is publicly known as deep learning and has led to a drastic increase in success of machine learning methods during the past decade. However, having a composition of many state-computing functions of similar size can slow down the optimization process. This is why, when forming such a composition, each pair of functions is often connected via a simple linear projection from the space of the state of the earlier function onto some lower-dimensional space that is then used by the following function. Note that while all the above claims seem natural and lead to a good enough performance for our paper, more benchmarking is needed to really confirm them.

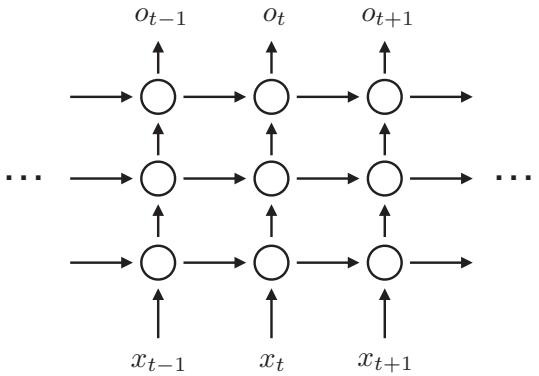


FIG. 6. Illustration of an RNN with three hidden layers.

Now, in the case of supervised learning, one assumes to be in possession of a set of time series $x_1, \ldots, x_n$ that will be used to let the RNN learn to predict series of this kind. The natural way of doing this is to define the pairs $(x_i, y_i) := (x_t, x_{t+1})$. While in principle the model is capable of taking into account all previous time steps, in practice it shows that optimization is only feasible for a relatively short number of steps. This is mainly due to the fact that the gradients that are needed to optimize the parameters of an RNN tend to grow to infinity or zero for higher numbers of steps. This will be discussed more in depth below.

b. Long short-term memory networks

In order to improve upon the standard RNN, Hochreiter and Schmidhuber introduced the long short-term memory network [49], which provides a different way of computing the state of an RNN. Hence the following set of equations can be perceived as a replacement for s_t from the previous section. The main advantage of the approach is that it drastically mitigates the problem of unstable gradients by construction. It is defined by the following set of equations:

$$\begin{aligned} i_t &= \mathrm{sigm}(U^i x_t + W^i s_{t-1} + b^i), \\ f_t &= \mathrm{sigm}(U^f x_t + W^f s_{t-1} + b^f), \\ o_t &= \mathrm{sigm}(U^o x_t + W^o s_{t-1} + b^o), \\ \tilde{c}_t &= \tanh(U^{\tilde{c}} x_t + W^{\tilde{c}} s_{t-1} + b^{\tilde{c}}), \\ c_t &= c_{t-1} * f_t + \tilde{c}_t * i_t, \\ s_t &= \tanh(c_t) * o_t, \end{aligned} \tag{D1}$$

where again x_t is the input at time step t, s_{t-1} is the previous state of the network, and c_t is the state of the cell. In addition, $U^i, U^f, U^o, U^{\tilde{c}} \in \mathbb{R}^{h \times l}$, while $W^i, W^f, W^o, W^{\tilde{c}} \in \mathbb{R}^{h \times h}$, $b^i, b^f, b^o, b^{\tilde{c}} \in \mathbb{R}^{1 \times h}$, and $*$ denotes the elementwise multiplication.

As it can be seen from the equations, the way in which an LSTM computes the state is a bit more involved. If needed, it may however just be treated as a black box and can be stacked just in the same manner as it was described for the plain RNN model. The general idea of an LSTM is to give the model a higher degree of control over the information that is propagated from one time step to the next. This is achieved by making use of so-called gates that control the information flow to and from the network and cell state. These gates, by taking into account the previous state and the new input, output vectors of values in $[0, 1]$ that determine how much information they let through. In the equations given above, i_t is called the input gate, f_t is

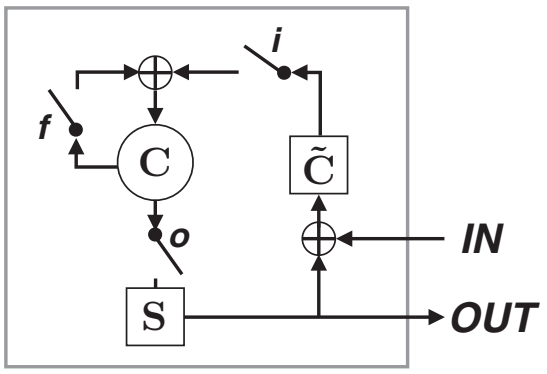


FIG. 7. Long short-term memory model illustrated in a schematic way. In addition to the diagram, the input gate i, the forget gate f, and the output gate o all depend on the current input x_t and previous state of the network s_{t-1}, as described in (D1).

referred to as the forget gate, and o_t denotes the output gate. Now, the mechanism works as follows.

(i) For a given time step t, the new input and previous network state are processed by $\tilde{c}_t$ like for the standard RNN and the output values are squashed to the interval $[-1,1]$ to yield candidate values for the next cell state.

(ii) The input gate i_t determines how to manipulate the information flow from the candidate cell state. Likewise, the forget gate f_t determines how to affect the information flow from the old cell state. The gated previous cell state and the gated input are then added to form the new cell state c_t.

(iii) Finally, the output gate o_t determines what to output from the new cell state. The new cell state is then also projected onto the interval $[-1,1]$ and put through the output gate to become the network state.

The whole process is shown in Fig. 7.

Naturally, there exists a plethora of possibilities to adapt the normal LSTM as explained above. One important enhancement is commonly referred to as peepholes, which allows the gates to incorporate the cell state via an extra term in the sum, in addition to the input and the network state. One other popular possibility introduced in [50] is the use of projection layers between different time steps of LSTM. In this case, we replace s_{t-1} by r_{t-1} in the equations for i_t, f_t, o_t, and $\tilde{c}_t$ and add the simple equation

$$r_t = W^p s_t,$$

where $W^p \in \mathcal{R}^{k\times h}$ is the projection matrix. In this work we have made use of both of these extensions of the normal LSTM. For an exhaustive overview over the known variants of the LSTM, we refer the interested reader to [35].

3. Optimization of RNNs

As the optimization problem described in the beginning of this section cannot be solved analytically for the models considered in this work, gradient-based approaches have established themselves as the state of the art. However, in the case of fitting the parameters of neural network models, three main restrictions need to be accounted for.

(a) The number of parameters for neural network models easily exceeds 100 000 and can for larger architectures go up to several tens or even hundreds of millions. Hence, computing the Hessian (or its inverse) explicitly is not tractable and so one is limited to first-order or approximative second-order methods.

(b) As the error function that is minimized is only a surrogate error function, its global optimum is not necessarily the optimum of the error function one actually wants to minimize.

(c) For many real-world data sets, computing the gradient of the complete sum of the error function over all samples is not feasible. Hence, the sum is normally split up into smaller parts called mini batches and these batches are looped over. A complete loop over D is then called an epoch.

These restrictions have led to the rise of an own subfield of machine learning that is concerned with the parallelization of gradient computations in the mini batch case, the approximation of second-order information, and the formal justification for the splitting up of the error function. All of the currently available methods are nevertheless extensions of the simplest method for gradient-based optimization known as steepest gradient descent: At iteration i in the loop over the batches, the parameters Θ are updated according to

$$\Theta_{i+1} = \Theta_i - \gamma \frac{\partial \mathcal{E}}{\partial \Theta_i},$$

where $\mathcal{E}(D,\Theta)$ is the respective error function and γ is called the step rate. The most straightforward natural adaption is to make γ depend on the iteration and slowly decrease it over time, following the intuition that smaller steps are beneficial the closer one gets to the respective optimum. In addition to that, many methods employ some kind of momentum term [51] or try to approximate second-order information and scale the gradient accordingly [39].

Besides this, the size of the batches also has an influence on the performance of the respective optimization method. In the extreme case where each batch only consists of one sample, the gradient descent method is known to converge almost surely to an optimum under certain constraints [52]. As picking individual samples for optimization can be perceived as sampling from the empirical distribution to approximate the overall gradient, this method is called stochastic gradient descent. Using single data points however is computationally inefficient and by definition leads to heavily oscillating optimization, so it is common practice to resort to larger batches. Following the ERM interpretation, batches B consisting of S_B samples are often used to compute an approximation of the mean gradient over D given by

$$\left\langle \frac{\partial \mathcal{E}}{\partial \Theta_i} \right\rangle_D \approx \left\langle \frac{\partial \mathcal{E}}{\partial \Theta_i} \right\rangle_B = \frac{1}{S_B} \sum_{(x,y)\in B} \frac{\partial \mathcal{E}_{(x,y)}}{\partial \Theta_i},$$

where obviously

$$\lim_{|B|\to|D|} \left\langle \frac{\partial \mathcal{E}}{\partial \Theta_i} \right\rangle_B = \left\langle \frac{\partial \mathcal{E}}{\partial \Theta_i} \right\rangle_D.$$

This interpretation is used, e.g., by the recently proposed algorithm Adam, which has been shown to yield very good local optima while being very robust with respect to noisy gradients and needing comparatively little adjustment of its parameters. We have employed Adam for fitting the models used in this work.

While the approach to optimizing artificial neural networks is well established, this does not change the fact that the optimization problems posed by them are inherently difficult. It is well known that the error landscape becomes less smooth the more layers one adds to a network. This results in error surfaces with large planes where $\frac{\partial \mathcal{E}}{\partial \Theta} \approx 0$ that are followed by short but very steep cliffs. If the step rate is not adapted correctly, the optimization procedure is very likely to get stuck in one these planes or saddle points and to jump away from an optimum in the vicinity of Θ if evaluated on one of the cliffs. The phenomena of the frequent occurrence of very large or very small gradients are referred to in the literature as the exploding gradient or vanishing gradient problem, respectively. To get a better understanding of why these problems exist, it is instructive to examine how the gradients for a given model are obtained.

As has been explained above, multilayer neural network models are a composition of nonlinear functions $\mathbb{R}^{i_k} \to \mathbb{R}^{o_k}$: $x_{k+1} = f_k(W_k x_k + b_k)$, where W_k is the weight matrix, b_k the bias vector, x_0 the input data, and x_K the final output of the network. From this definition it is clear that $o_k = i_{k+1}$. For convenience, we define $y_k \equiv W_k x_k + b_k$. In order to obtain the gradient for a specific W_k or b_k one must obviously make use of the chain rule such that

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial W_k} &= \frac{\partial \mathcal{E}}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial y_k} \frac{\partial y_k}{\partial W_k} \\ &= \frac{\partial \mathcal{E}}{\partial x_K} \left(\prod_{j=k+1}^{K-1} \frac{\partial x_{j+1}}{\partial x_j} \right) \frac{\partial x_{k+1}}{\partial y_k} \frac{\partial y_k}{\partial W_k}\end{aligned}$$

and

$$\begin{aligned}\frac{\partial \mathcal{E}}{\partial b_k} &= \frac{\partial \mathcal{E}}{\partial x_{k+1}} \frac{\partial x_{k+1}}{\partial y_k} \frac{\partial y_k}{\partial b_k} \\ &= \frac{\partial \mathcal{E}}{\partial x_K} \left(\prod_{j=k+1}^{K-1} \frac{\partial x_{j+1}}{\partial x_j} \right) \frac{\partial x_{k+1}}{\partial y_k} \frac{\partial y_k}{\partial b_k},\end{aligned}$$

where $\frac{\partial}{\partial W_k}$ is the shortcut of doing the derivative elementwise:

$$\left[\frac{\partial}{\partial W_k} \right]_{ab} = \frac{\partial}{\partial [W_k]_{ab}}.$$

The same convention applies to $\frac{\partial}{\partial b_k}$. As $\frac{\partial}{\partial W_k}$ and $\frac{\partial}{\partial b_k}$ depend on all the gradients of the later layers, this formulation yields an efficient method of computing the gradients for all layers by starting with the uppermost layer and then descending in the network, always reusing the gradients already computed. Together with the fact that many of the commonly used nonlinearities have an easy closed-form expression of the first derivative, this allows for fully automatic computation of the gradients as it is done in every major deep learning framework. This dynamic programming method of computing the gradients is known in the literature as backpropagation. The vanishing (exploding) gradient problem arises because of the product $\prod_{j=k+1}^{K-1} \frac{\partial x_{j+1}}{\partial x_j}$ in the above equations. For example, if one of the $\frac{\partial x_{j+1}}{\partial x_j} \approx 0$ in the product, then likely we have $\frac{\partial \mathcal{E}}{\partial W_k} \approx 0$, which leads to an ineffective gradient descent. Similarly, if many of the terms $\frac{\partial x_{j+1}}{\partial x_j}$ have large norms, then there is a possibility that $\frac{\partial \mathcal{E}}{\partial W_k}$ becomes too large, which often causes the optimization method to jump out of a local optimum.

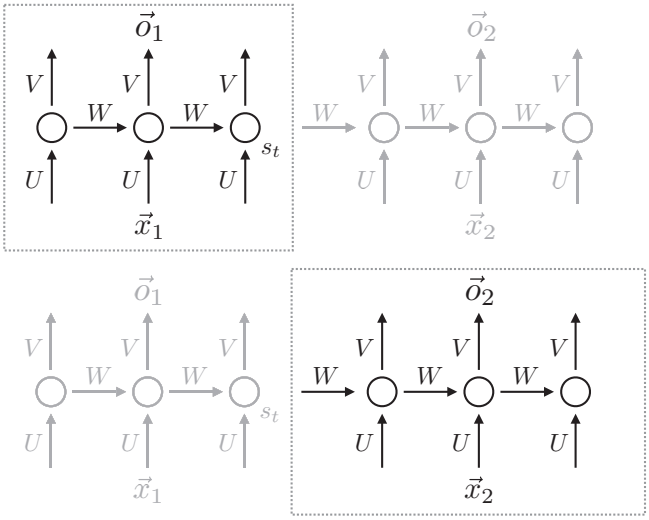


FIG. 8. Illustration of how we truncate the gradient computation for long sequences. Here we divide the sequences into two halves. As the first step, we compute the gradient of the error function $\mathcal{E}(\vec{x}_1, \vec{o}_1)$ with respect to the parameters U, V, W, while ignoring the other half of the network. In the second step, we compute the gradient of $\mathcal{E}(\vec{x}_2, \vec{o}_2)$, while treating the final state of the network s_t of the first half as a constant. The final gradients are approximated by the sums of these two constituents. Thus, we are able to avoid the instability of computing gradients, but still capture the correlation between two halves, since we feed the final network state s_t into the second half.

In the case of an RNN as defined in Sec. V, the above generic equations for the derivative become a little more involved, as in addition to the term for possibly multiple stacked layers, a term accounting for states of previous times has to be added. Nevertheless, at the heart of the problem, it is still about computing derivatives of composite functions. This slightly more involved backpropagation method is known as backpropagation through time and can also be fully automatized. Similar to the multilayer neural network models mentioned above, the gradient computation of RNNs also has these instability issues. As can be seen from Fig. 5, the same matrix W is used in all time steps of an RNN. Thus, a tiny change of W could affect the output o_t drastically when the time step t gets big. In other words, the derivative of the error function $\mathcal{E}$ with respect to W could again become very large or very small in certain situations. To deal with this issue, we could truncate the number of time steps during the computation, as described in Fig. 8. More discussion on this topic can be found in Sec. 3.2 of [32].

APPENDIX E: TECHNICAL ASPECTS

For the implementation of this work, we have made use of PYTHON with the numerical libraries NumPy, SciPy, and TensorFlow [53–55]. All experiments were run on single workstations with up to eight threads and a GeForce Titan X. The runtime of the experiments varied, depending on the optimization parameters, from a few hours to days.

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, in *Proceedings of the Conference on Advances in Neural Information Processing Systems 25*, edited by F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (MIT Press, Cambridge, MA, 2012), pp. 1097–1105.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, arXiv:1312.5602.

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, Nature (London) **529**, 484 (2016).

[4] R. S. Judson and H. Rabitz, Phys. Rev. Lett. **68**, 1500 (1992).

[5] U. Las Heras, U. Alvarez-Rodriguez, E. Solano, and M. Sanz, Phys. Rev. Lett. **116**, 230504 (2016).

[6] I. Geisel, K. Cordes, J. Mahnke, S. Jöllenbeck, J. Ostermann, J. Arlt, W. Ertmer, and C. Klempt, Appl. Phys. Lett. **102**, 214105 (2013).

[7] Ł. Pawela and P. Sadowski, Quantum Inf. Process. **15**, 1937 (2016).

[8] M. Grace, C. Brif, H. Rabitz, I. A. Walmsley, R. L. Kosut, and D. A. Lidar, J. Phys. B **40**, S103 (2007).

[9] M. Krenn, M. Malik, R. Fickler, R. Lapkiewicz, and A. Zeilinger, Phys. Rev. Lett. **116**, 090405 (2016).

[10] C. Chen, L.-C. Wang, and Y. Wang, Sci. World J. **2013**, 869285 (2013).

[11] D. Dong, C. Chen, B. Qi, I. R. Petersen, and F. Nori, Sci. Rep. **5**, 7873 (2015).

[12] L. Banchi, N. Pancotti, and S. Bose, npj Quantum Inf. **2**, 16019 (2016).

[13] S. Machnes, U. Sander, S. J. Glaser, P. de Fouquières, A. Gruslys, S. Schirmer, and T. Schulte-Herbrüggen, Phys. Rev. A **84**, 022305 (2011).

[14] D. J. Egger and F. K. Wilhelm, Phys. Rev. Lett. **112**, 240503 (2014).

[15] M. J. Biercuk, H. Uys, A. P. VanDevender, N. Shiga, W. M. Itano, and J. J. Bollinger, Nature (London) **458**, 996 (2009).

[16] P. Doria, T. Calarco, and S. Montangero, Phys. Rev. Lett. **106**, 190501 (2011).

[17] J. Kelly *et al.*, Phys. Rev. Lett. **112**, 240504 (2014).

[18] P. B. Wigley, P. J. Everitt, A. van den Hengel, J. W. Bastian, M. A. Sooriyabandara, G. D. McDonald, K. S. Hardman, C. D. Quinlivan, P. Manju, C. C. N. Kuhn, I. R. Petersen, A. N. Luiten, J. J. Hope, N. P. Robins, and M. R. Hish, Sci. Rep. **6**, 25890 (2016).

[19] J. Combes, C. Ferrie, C. Cesare, M. Tiersch, G. J. Milburn, H. J. Briegel, and C. M. Caves, arXiv:1405.5656.

[20] D. Orsucci, M. Tiersch, and H. J. Briegel, Phys. Rev. A **93**, 042303 (2016).

[21] M. Tiersch, E. J. Ganahl, and H. J. Briegel, Sci. Rep. **5**, 12874 (2015).

[22] L. Viola, E. Knill, and S. Lloyd, Phys. Rev. Lett. **82**, 2417 (1999).

[23] A. M. Souza, G. A. Alvarez, and D. Suter, Phys. Rev. Lett. **106**, 240501 (2011).

[24] G. Quiroz and D. A. Lidar, Phys. Rev. A **88**, 052306 (2013).

[25] G. de Lange, Z. H. Wang, D. Ristè, V. V. Dobrovitski, and R. Hanson, Science **330**, 60 (2010).

[26] A. Karpathy, J. Johnson, and L. Fei-Fei, arXiv:1506.02078.

[27] Z. C. Lipton, J. Berkowitz, and C. Elkan, arXiv:1506.00019.

[28] C. A. Ryan, J. S. Hodges, and D. G. Cory, Phys. Rev. Lett. **105**, 200402 (2010).

[29] M. D. Grace, J. Dominy, R. L. Kosut, C. Brif, and H. Rabitz, New J. Phys. **12**, 015001 (2010).

[30] A. M. Souza, G. A. Álvarez, and D. Suter, Philos. Trans. R. Soc. A **370**, 4748 (2012).

[31] C. Shannon, Bell Syst. Tech. J. **27**, 379 (1948).

[32] A. Graves, arXiv:1308.0850.

[33] J. B. Pollack, On connectionist models of natural language processing, Ph.D. thesis, New Mexico State University, 1987.

[34] M. A. Nielsen, *Neural Network and Deep Learning* (Determination Press, 2015).

[35] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, arXiv:1503.04069.

[36] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016).

[37] M. Pelikan, D. E. Goldberg, and F. G. Lobo, Comput. Optim. Appl. **21**, 5 (2002).

[38] A. P. Dempster, N. M. Laird, and D. B. Rubin, J. R. Stat. Soc. Ser. B **39**, 1 (1977).

[39] D. Kingma and J. Ba, arXiv:1412.6980.

[40] A. M. Souza, G. A. Álvarez, and D. Suter, Phys. Rev. A **85**, 032306 (2012).

[41] M. Pelikan, in *Hierarchical Bayesian Optimization Algorithm* (Springer, Berlin, 2005), pp. 31–48.

[42] M. Welling and Y. W. Teh, in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (ICML, Bellevue, 2011), pp. 681–688.

[43] M. Pelikan and A. K. Hartmann, in *Scalable Optimization via Probabilistic Modeling* (Springer, Berlin, 2006), pp. 333–349.

[44] R. J. Williams and D. Zipser, Neural Comput. **1**, 270 (1989).

[45] P. J. Werbos, Proc. IEEE **78**, 1550 (1990).

[46] Y. Bengio and Y. LeCun, in *Large-Scale Kernel Machines*, edited by L. Bottou, O. Chapelle, D. DeCoste, and J. Weston (MIT Press, Cambridge, MA, 2007), Chap. 14.

[47] K. Hornik, M. Stinchcombe, and H. White, Neural Networks **2**, 359 (1989).

[48] K. Hornik, Neural Networks **4**, 251 (1991).

[49] S. Hochreiter and J. Schmidhuber, Neural Comput. **9**, 1735 (1997).

[50] H. Sak, A. Senior, and F. Beaufays, arXiv:1402.1128.

[51] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (ICML, Bellevue, 2013), pp. 1139–1147.

[52] D. Saad, *On-line Learning in Neural Networks* (Cambridge University Press, Cambridge, 2009), Vol. 17.

[53] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, Comput. Sci. Eng. **13**, 22 (2011).

[54] E. Jones, T. Oliphant, and P. Peterson (unpublished).

[55] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, software available from TensorFlow.org.

E Taking gradients through experiments: LSTMs and memory proximal policy optimization for black-box quantum control

Authors Moritz August and Prof. Dr. José Miguel Hernández-Lobato

Citation arXiv:1802.04063v2 [cs.LG], 2018

Summary Following the article shown in Appendix D, in this work we presented an improved method to solve black-box quantum control with LSTM networks. Since here we made use of concepts from reinforcement learning, we tried to use the opportunity to introduce the application of quantum control as an interesting reinforcement learning problem to the ML community. To this end, we provided an analysis of the structure of the reinforcement learning problems arising in our black-box quantum control scenario. We argued that learning agents parameterized by LSTM networks trained via policy gradient algorithms constitute a versatile method to solving such problems. Based on it as a recent and successful policy gradient method, we then introduced a variant of the proximal policy optimization (PPO) algorithm we called memory proximal policy optimization (MPPO). To demonstrate the versatility of our general method, we subsequently applied it to the previously tackled scenario of dynamical decoupling for quantum memory as well as the problem of transitioning between ground states. The latter setting had also recently been discussed in another work on reinforcement learning for quantum control. For these two settings, we introduced several learning problems of varying difficulty in the discrete and continuous domain. We demonstrated for each case how our method can easily incorporate physical domain knowledge. Then, we evaluated our method's performance for all introduced learning problems. We found it to be able to reproduce reference results as well as successfully tackle problems out of reach for the reference methods. Additionally, we evaluated the improvements introduced in the MPPO algorithm and found the results to favor our method. We furthermore found our formulation of the learning problem in terms of entire control sequences to yield better results than a formulation based on individual control parameters. Finally, we were again able to provide evidence for our assumption that optimal or near-optimal control sequences share common structure.

Contribution The author's contribution ranges from the development of the method over its implementation and evaluation to the preparation of the article. The author realized the good fit of policy gradients to the black-box quantum control setting. The author then discovered the PPO algorithm in the literature as the at the time of the writing of the article best performing policy gradient algorithm. Based on the reinforcement learning structure of the considered control problems, the author then developed the MPPO algorithm and devised the learning problems it would be evaluated on. The implementation and evaluation were also performed by the author as well as the preparation of the article itself. During all phases of this work, Prof. Hernández-Lobato provided valuable feedback and was kindly available for discussion concerning all questions and ideas.

Taking gradients through experiments: LSTMs and memory proximal policy optimization for black-box quantum control

Moritz August
Department of Informatics, Technical University of Munich, 85748 Garching, Germany (august@in.tum.de)

José Miguel Hernández-Lobato
Computational and Biological Learning Lab, University of Cambridge, CB2 1PZ Cambridge, United Kingdom (jmh233@cam.ac.uk)

In this work we introduce the application of black-box quantum control as an interesting reinforcement learning problem to the machine learning community. We analyze the structure of the reinforcement learning problems arising in quantum physics and argue that agents parameterized by long short-term memory (LSTM) networks trained via stochastic policy gradients yield a general method to solving them. In this context we introduce a variant of the proximal policy optimization (PPO) algorithm called the memory proximal policy optimization (MPPO) which is based on this analysis. We then show how it can be applied to specific learning tasks and present results of numerical experiments showing that our method achieves state-of-the-art results for several learning tasks in quantum control with discrete and continuous control parameters.

I. INTRODUCTION

As a result of collaborative efforts by academia and industry, machine learning (ML) has in recent years led to advancements in several fields of application ranging from natural language and image processing over chemistry to medicine. In addition to this, reinforcement learning (RL) has recently made great progress in solving challenging problems like Go or Chess [1, 2] with only small amounts of prior knowledge which was widely believed to be out of reach for the near future. Consequentially, RL is nowadays thought to hold promise for applications such as robotics or molecular drug design. This success naturally raises the question of what other areas of application might benefit from the application of machine learning.

Quantum mechanics and especially quantum computing is of special interest to the machine learning community as it can not only profit from applications of state-of-the-art ML methods but is also likely to have an impact on the way ML is done in the future [3]. This bidirectional influence sets it apart from most other applications and is a strong incentive to investigate possible uses of machine learning in the field despite the comparably steep learning curve.

One challenging and important task in the context of quantum physics is the control of quantum systems over time to implement the transition between an initial and a defined target physical state by finding good settings for a set of control parameters [4]. This problem lies at the heart of quantum computation as performing any kind of operation on quantum bits (qubits) amounts to implementing a controlled time evolution with high accuracy in the face of noise effects induced by the environment. Apart from the relevance to quantum computation, the analysis and understanding of the properties of quantum control problems also is an interesting research problem in its own right. However, for a given physical system as implemented in a real experiment it is in general not possible to express all influence factors and dependencies of particles in mathematical form to perform an analytical analysis or gradient-based optimization of the control variables. Thus, physicists have for some time been proposing automated solutions for these problems [5–9] that are able to find good control parameter settings while being as agnostic as possible about the details of the problem in question. Unfortunately though, these approaches are in general based on tailored solutions that do not necessarily generalize to other problems as they, e.g. only consider discrete variables when the underlying problem is actually continuous and are not always very sample efficient.

In this work we improve over the status quo by introducing a control method based on recurrent neural networks (RNNs) and policy gradient reinforcement learning that is generic enough to tackle every kind of quantum control problem while simultaneously allowing for the incorporation of physical domain knowledge. More precisely, we present an improved version of the recently introduced proximal policy optimization (PPO) algorithm [10] and use it to train Long Short-Term Memory (LSTM) [11] networks to approximate the probability distribution of good sequences of control parameters. We furthermore show how physical domain knowledge can be incorporated to obtain state-of-the-art results for two recently addressed control problems [8, 9]. While our method is based on an analysis of the reinforcement problem underlying quantum control, it can also be applied to other RL problems yielding the same structure. Our contribution hence is threefold in that we firstly introduce the general method, secondly demonstrate how to successfully apply it to quantum control problems and thirdly, by doing so, try to stimulate a more intense exchange of ideas between quantum physics to the broader machine learning community to facilitate mutual benefit.

The rest of this work is structured as follows: in Sec-

tion II, we provide a very brief introduction to quantum control, followed by a discussion and analysis of the reinforcement learning problem posed by quantum control in Section III. Building on the analysis, we present the method in Section IV and subsequently introduce two concrete quantum control problems in Sections V A and V B respectively. We then present numerical results obtained by our method for these problems and compare them to those of existing solutions in Section VI. Finally, we conclude with a discussion of the work in Section VII.

II. QUANTUM CONTROL

The time evolution of a physical system in quantum mechanics is described by the Schrödinger equation

$$ih\frac{\delta}{\delta t}\left|\psi(t)\right\rangle = H\left|\psi(t)\right\rangle \tag{1}$$

where H is the Hamiltonian, a complex Hermitian Matrix describing the energy of the physical system, and h is Planck's constant [12]. Hereby, $|\psi\rangle$ is the Dirac notation for a physical state which for finite dimensional systems as we treat here corresponds to a complex column vector of the same dimensionality as the Hamiltonian's. The conjugate transpose of a vector $|\psi\rangle$ then is denoted as $\langle\psi|$ such that $\langle\psi,\psi\rangle$ denotes the inner and $|\psi\rangle\langle\psi|$ the outer product. The Schrödinger equation yields the unitary quantum time evolution

$$\left|\psi(t)\right\rangle = e^{-itH/h}\left|\psi(0)\right\rangle. \tag{2}$$

In a discretized time setting with time steps Δt the evolution for a total time T can thus be written as

$$\left|\psi(T)\right\rangle = e^{-i\Delta tH/h^{L}}\left|\psi(0)\right\rangle \tag{3}$$

where we define $L = T/\Delta t$. In quantum control we now assume to be able to control the time evolution by application of so-called control Hamiltonians $H_1, \cdots, H_C$, which yields the controlled time evolution

$$\left|\psi(T)\right\rangle = e^{-i\Delta t\sum_{i=1}^{C} c_{iL}H_i/h}\ldots \tag{4}$$

$$e^{-i\Delta t\sum_{i=1}^{C} c_{i1}H_i/h}\left|\psi(0)\right\rangle \tag{5}$$

where the c_{it} are time-dependent scaling constants for the control Hamiltonians. This formulation however assumes that we have full control over the system which due to various kinds of noise or environmental effects will not be the case. Hence we introduce a noise or drift Hamiltonian H_0, which we here assume to be time independent and of constant strength, and obtain the final formulation

$$\left|\psi(T)\right\rangle = e^{-i\Delta t(H_0+\sum_{i=1}^{C} c_{iL}H_i)}\ldots \tag{6}$$

$$e^{-i\Delta t(H_0+\sum_{i=1}^{C} c_{i1}H_i)}\left|\psi(0)\right\rangle \tag{7}$$

where we set $h = 1$ for convenience.

Now that we have a well-defined notion of our control problem, we need to state the actual goal that we aim to achieve. Generally, starting from an initial state $|\psi(0)\rangle$ or the corresponding density operator $\rho(0) = |\psi(0)\rangle\langle\psi(0)|$ we would like to obtain an evolution to target state $|\psi^*\rangle$ or $\rho^* = |\psi^*\rangle\langle\psi^*|$. Hence we need to define some similarity measure between the state we actually obtain after evolving for time T and our ideal result. The easiest way of doing this is simply to compute the overlap between these states by

$$S(\psi^*, \psi(T)) = \langle\psi^*, \psi(T)\rangle \tag{8}$$

or

$$S(\rho^*, \rho(T)) = \mathsf{Tr}\rho^{*\dagger}\rho(T) \tag{9}$$

respectively for Hermitian operators and correspondingly only using the real part $\mathrm{Re}(S(\rho^*, \rho(T)))$ for non-Hermitian ones [13].

Equipped with this metric, we can formally define the problem we would like to solve as

$$\max_{\{c_{it}\}} S(\rho^*, \rho(T, \{c_{it}\}). \tag{10}$$

This formulation is broad enough to capture every problem from synthesizing certain quantum gates over evolving from one eigenstate of a Hamiltonian to another to storing the initial state in a quantum memory setting.

III. REINFORCEMENT LEARNING: WHY AND WHAT?

As we have seen above, solving quantum control problems amounts to determining an optimal or at least good sequence of principly continuous variables that describe the influence we exert on the system at each discrete time step. If a rigorous mathematical description of the evolution dynamics is available, there exist methods like GRAPE [13] or CRAB [14, 15] to obtain good solutions. However, the gap between theory and experiment also does not close in quantum mechanics and hence it is reasonable to assume that the actual dynamics of a real experiment will slightly differ from the mathematical model due to various noise effects induced by the environment. As can for instance also be observed in robotics, these slight differences between theory/simulation and real world implementation might still have a significant impact on the optimization problem to be solved. Additionally, it is clear that in general it is neither an interesting nor feasible task to derive a proper mathematical model for the effect of every influence factor in a real experiment [8].

This shows that it is worthwhile to investigate ways of optimizing such a control problem from a black box perspective in the sense that we are agnostic about the actual time evolution dynamics of the system and can only observe the final results obtained by a chosen set

of parameters. In fact, in the absence of a mathematical model it is the only possible option to obtain information after the end of an experiment as in quantum mechanics a measurement during the experiment would in general cause the wave function to collapse and hence destroy the experiment without any way of determining what the final outcome would have been. Hence the task we would like to solve is to find a controller or at least find a good sequence of control parameters based on the outcomes of trial runs of a given experiment, which in quantum control terminology corresponds to a closed-loop setting.

While one viable route to solving this problem would be to use classical evolutionary or hill-climbing algorithms or more advanced black-box methods such as Bayesian optimization, another interesting option is to fit a generative probabilistic model from which we can efficiently sample good sequences. This approach has two advantages. Firstly, we can iteratively update the model by fitting it to additional data we might acquire after the initial fitting phase. Doing so allows it to improve over previous results or make it adapt to changing conditions, e.g. a change of the noise Hamiltonian after some time. This is in contrast to pure optimization methods which would have to start from scratch for every problem. Secondly, by examining the distribution over the sequence space the model has learned and inspecting the best sampled control sequences, it might be possible to gain a better understanding of the underlying dynamics of a system.

It is clear that the sequences of control parameters in a quantum control problem should not be treated as i.i.d. as a given choice of parameters c_t at time t potentially depends on all previous choices $c_1, \cdots, c_{t-1}$ and thus we have a conditional distribution $p(c_t|c_1, \cdots, c_{t-1})$. This kind of distribution can successfully be learned by modern RNN variants, such as LSTM or Gated Recurrent Unit (GRU) networks. This can for instance be seen in natural language processing (NLP) problems, which feature similar structure and where RNNs have led to breakthrough results in recent years. Note that, with this modelling decision, we still capture the full multivariate distribution $p(c_1, \cdots, c_T)$ as by the factorization rule of probabilities it holds that

$$p(c_1, \cdots, c_T) = \prod_{t=1}^{T} p(c_t|c_1, \cdots, c_{t-1}). \tag{11}$$

Having decided on the class of models to employ, we are left with the question of how to fit them. This is non-trivial as we obviously can not hope to be able to obtain gradients of real-world experiments and also can not assume to have any a priori data available. Hence, we must 'query' the experiment to obtain tuples of sequences and results. Thereby we would naturally like to be as sample efficient as possible and hence have to find an intelligent way to draw samples from the experiment and learn from them.

In a recent attempt to address this problem, an evolutionary-style algorithm for training LSTMs was introduced [9] that iteratively generates better data and fits the models to that data, then uses sampling from these models instead of the usual mutation operations to generate new sequences. While the algorithm was able to find better sequences than known in theory for the considered control problem of quantum memory, it was only demonstrated for a discretized version of the problem and there is room for improvement with respect to the efficient use of sampleded sequences.

A more direct solution to this black-box optimization problem would however be if we were able to simply approximate the gradient of the error function with respect to the parameters of our model from the sampled data. Being able to obtain an approximate gradient would allow us to optimize our model in a gradient descent fashion and thus to leverage existing optimization methods mainly used in superivsed learning. Indeed, this is a typical RL scenario which is commonly referred to as *policy gradient* learning. In the following, we will thus show how to solve the optimization task at hand by perceiving the problem of black-box quantum control as an RL problem and tackling it with a state-of-the-art policy gradient algorithm. To this end, we start by analyzing the particular reinforcement learning problem posed by black-box quantum control.

As we only receive a result or measurement, from now on also referred to as reward, after having chosen a complete sequence of control parameters, we can perceive the sequence $c = (c_1, \cdots, c_T)$ as a single action of the RL agent for which it receives a reward $R(c)$. This approach most clearly reflects the envisioned closed-loop control scenario explained above. Modelling the sequences and their respective results in this way then implies that our Markov decision process (MDP) takes the form of a bipartite graph consisting of a single initial state s_0 on the left and multiple final states s_c on the right that are reached deterministically after exactly one action c. The set of states S of this MDP is thus given by $S = s_0 \cup \{s_c\}$ while the set of actions A corresponds to $A = \{c\}$ and the transition probabilities are defined as $P_c(s_0, s_c) = 1$. The reward $R(c)$ of an action c is determined by the associated value of the error function as defined in Section II. We assume here that two different sequences always lead to different final states of the system, which is the most challenging conceivable case as equivalence classes in the sequence space would effectively reduce the size of the search space. This particular structure then implies that the value function simplifies to

$$V(s_0) = \max_c R(c) = R(c^{opt}) \tag{12}$$

where c^{opt} is the optimal sequence and the Q-function

$$Q(s_0, c) = R(c) \tag{13}$$

is in fact independent of the state and equal to the reward function $R(c)$ as each sequence c is associated with exactly one final state. Additionally, the number of actions

$|\{c\}|$ and hence final states x_c is *at least* exponential in the number of possible values of control parameters per time step t and generally infinite. This learning setting can be perceived as a *multi-armed bandit* [25] problem but constitutes a special case as firstly we assume to be only able to perform one action, i. e. generate one sequence, before receiving the total reward and secondly the actions are not atomic but rather exhibit a structure we exploit for learning.

While it is true that one could derive a different formulation of the problem by considering the c_t to be individual actions and using the discounted rewards of complete sequences, this approach puts more emphasis on optimal local behvaior of the agent when our goal clearly is to optimize the global performance, i.e. to generate the best possible sequences of control parameters. However, for this RL problem to be solvable the compositional structure of the actions c is in fact of critical importance as we will discuss now.

In principle, the RL problem amounts to learning to choose the best out of up to infinitely many possible actions which in general clearly is unsolvable for every algorithm. So, why can we hope to achieve something with an algorithm learning from trials in the introduced problem setting? The main reason for this is in fact that we know that the actions the agent takes are not atomic but concatenations of multiple sub-actions which have a physical meaning. Nature as we perceive it seems to be governed by simple underlying rules (or complex rules that are at least approximated very well by simple ones) which allows us to capture them with mathematical expressions. This in turn implies that there is much structure to be found in Nature and hence it is reasonable to assume that likewise the desirable actions in our learning problem share certain patterns which can be discovered. More precisely, we conjecture that solving the particular problems we are tackling in this work requires less abstract conceptual inference, which would still be out of reach for todays machine learning models, and more recognition of patterns in large sets of trials, i.e. control sequences, and hence in fact lends itself to treatment via machine learning and especially contemporary RNN models. Some empirical evidence for the validity of this conjecture has recently been provided for the problem of quantum memory [9] and for a problem related to quantum control, the design of quantum experiments [6].

IV. THE LEARNING ALGORITHM

Having discussed the modelling of the control sequences and the RL problem, we will now introduce the actual learning algorithm we employ. As we have seen above, we can not perform direct optimization of $R(c)$ as we cannot access $\nabla R(c)$. However, it has long been known that it is possible to approximate $\nabla_\Theta \mathbf{E}_c[R(c)]$ since

$$\nabla_\Theta \mathbf{E}_c[R(c)] = \mathbf{E}_c[\nabla \ln p_\Theta(c) R(c)] \tag{14}$$

where $\mathbf{E}_c$ is the expectation over the sequence space and $p_\Theta(c)$ is the stochastic policy of the agent parameterized by the weight vector Θ, which in this work corresponds to an RNN. This insight is known as the likelihood ratio or REINFORCE [16] trick and constitutes the basis of the policy gradient approach to reinforcement learning. From the physics point of view, the trick allows us to take the gradient of the average outcome of a given experiment with respect to the parameters of our stochastic controller and perform gradient-based optimization while being agnostic about the mechanisms behind the experiment, i.e. model-free. In a sense we thus have a way of taking a gradient through an experiment without the necessity to mathematically model every variable of influence and their interplay. From a different perspective, this approach simply corresponds to maximizing the likelihood of sequences that are weighted by their results, such that the agent has a higher incentive to maximize the likelihood of good sequences. The approach can be refined by replacing the weighting by the pure reward $R(c)$ with an approximation of the advantage $A(s,c) = Q(s,c) - V(s)$. This has been shown to improve the convergence significantly and especially for continuous control problems, policy gradient methods outperform Q-learning algorithms [10].

Despite such improvements, policy gradient approaches still suffer from slow convergence or catastrophicly large updates, which has led to the development of improvements such as trust region policy optimization [17] (TRPO). These methods however make use of second-order information such as inverses of the Hessian or Fisher information matrix and hence are very difficult to apply in large parameter spaces which are common in the deep learning regime. The underlying idea of such improvements thereby is limiting the magnitude of updates to Θ by imposing constraints on the difference between p_Θ and $p_{\Theta_{new}}$ in order to prevent catastrophic jumps out of optima and achieve a better convergence behvaior.

In an effort to strike a balance between ease of application and leveraging the insights behind TRPO, recently a novel policy gradient scheme called proximal policy optimization [10] (PPO) was introduced. One main novelty hereby lies in the introduced loss, which is for a general RL scenario given by

$$L^{CLIP}(\Theta) = \mathbf{E}_t[\min(r_t(\Theta)A_t, \tag{15}$$

$$\text{clip}(r_t(\Theta), 1-\epsilon, 1+\epsilon)A_t)] \tag{16}$$

where $\mathbf{E}_t$ and A_t are the expectation over time steps and the advantage at time t respectively, which both need to be approximated. The term r_t is defined as the ratio of likelihoods

$$r_t(\Theta) = \frac{p_\Theta(c_t|s_t)}{p_{\Theta_{old}}(c_t|s_t)} \tag{17}$$

of actions c_t in states s_t in our notation and we define $\text{clip}(a, b, c) = \min(\max(a, b), c)$. The distribution $p_\Theta(c_t|s_t)$ is a stochastic policy depending on parameters Θ. Note that this generic formulation assumes multiple actions c_t per episode and thus does not yet apply to the learning scenario discussed here.

The objective function poses a lower bound on the improvement induced by an update and hence establishes a trust region around Θ_{old}. The hyperparameter ϵ controls the maximal improvement and thus the size of the trust region.

Now, the basic algorithm is defined as follows:

1. Obtain new set of trajectories, i.e. sequences, C, by sequentially sampling from $p_\Theta(c_t|s_t)$.
2. Optimize L^{CLIP} over C for K iterations.
3. Set $\Theta_{old} = \Theta$.
4. Repeat until convergence.

Note that there exists a straight-forward generalization to the case of multiple agents but as we can not reasonably assume in our application to have access to multiple identical experiments, we only consider the case of one agent here. The algorithm was shown to achieve state-of-the-art performance for several discrete and contiouous control tasks, which makes it ideally suited for the problems tackled in this work. However, we will now introduce a few improvements tailored to our specific reinforcement learning problem as defined in the previous section which we will for the sake of brevity from now on refer to as memory proximal policy optimization (MPPO).

Since in our problem we only consider episodes consisting of one action c, the objective becomes

$$L_1^{CLIP}(\Theta) = \mathbf{E}_c[\min(r(\Theta)A, \tag{18}$$

$$\text{clip}(r(\Theta), 1-\epsilon, 1+\epsilon)A)] \tag{19}$$

with

$$r(\Theta, c) = \frac{p_\Theta(c)}{p_{\Theta_{old}}(c)} \tag{20}$$

and $p_\Theta(c)$ being parameterized by an LSTM, as discussed above. A again denotes the advantage function. We have omitted the dependence on c in L_1 for the sake of clarity. Since we know that in our problem setting it holds that $Q(c, s) = R(c)$, the advantage function becomes

$$A(c) = R(c) - V(c). \tag{21}$$

It is worth noting that this implies that in our scenario there is no need to approximate the Q-function as we can access it directly. In fact approximating the Q-function and hence $R(c)$ would be equivalent to solving the optimization problem as we could use the approximator to optimize over its input space to find good sequences. The quality of the approximation of $A(c)$ consequentially only depends on the approximation of $V(c)$. While there exist many sophisticated ways of approximating the value function [10, 18] in our case the optimal approximation is given by

$$\hat{V}(c) = R(c^*) \tag{22}$$

where c^* is the best sequence we have encountered so far. Since we do not know the best sequence and its corresponding reward (at best we know an upper bound), the reward of the best sequence found so far is the closest approximation we can make. The optimal approximation of the advantage $A(c)$ hence is given by

$$\hat{A}(c) = R(c) - R(c^*). \tag{23}$$

Since we need to store c^* to compute the advantage approximation and are generally interested in keeping the best solution, it is a natural idea to equipping the agent with a memory M of the best sequences found so far. We can then formulate a memory-enhanced version of the PPO algorithm:

1. Obtain new set of trajectories, i.e. sequences, C, by sampling from $p_\Theta(c)$.
2. Update the memory of best sequences M
3. Optimize L_1^{CLIP} over $C \cup M$ for K iterations.
4. Set $\Theta_{old} = \Theta$.
5. Repeat until convergence

The memory sequences are treated as newly sampled sequences such that their weighting always is performed with respect to the current values of Θ_{old} and Θ. This ensures compatibility with the policy gradient framework while the access to the best actions discovered so far leads to a better convergence behavior as we will see later. Note that, under the previously introduced assumption, the best sequences share common structural properties. Maximizing the expected reward over all sequences $\mathbf{E}_c[R(c)]$ is thus equivalent to maximizing the expected reward over the sequences in the memory $\mathbf{E}_{c\in M}[R(c)]$ which ensures relevance and stability of the updates computed over M. This memory scheme furthermore is different from experience replay in Q-learning [19] as only the best sequences are kept and reintroduced to the agent. The relation between $|C|$ and $|M|$ thereby is a new hyperparameter of the algorithm affecting the exploration-exploitation dynamics of the learning process.

Another factor that has a significant impact on the exploration behavior is the value of the scaling or variance parameter of the probability distributions employed in continuous control tasks, such as for instance the standard deviation σ of the univariate normal distribution or the covariance matrix Σ in the multivariate case. It is clear that a large variance induces more exploration while

a small variance corresponds to a more exploitation-oriented behavior. Over the course of training an agent to find a good policy it is hence reasonable to start with a larger variance and reduce it during the optimization until it reaches a defined minimal value. However, while the agent usually learns to predict the mean of the given distribution, the variance parameter is currently often treated as fixed or follows a predefined decay schedule which does not account for the randomness in the training process. Utilizing the sequence memory, we propose an improvement by introducing a dynamical adaptation scheme for the variance parameters depending on the improvement of the memory M. More concretely, we propose to maintain a window W_i of the relative improvements of the average rewards in memory

$$W_i = \left[\frac{\overline{R(M_{i-l+1})} - \overline{R(M_{i-l})}}{\overline{R(M_{i-l})}}, \cdots, \frac{\overline{(M_i)} - \overline{R(M_{i-1})}}{\overline{R(M_{i-1})}} \right] \tag{24}$$

where $\overline{R(M_i)}$ denotes the average reward over the memory in iteration i of the optimization and l is the window length. At every l-th step in the optimization, we then compute a change parameter

$$\alpha_t = 1 + \frac{\overline{W_{t-l}} - \overline{W_t}}{\overline{W_{t-l}}} \tag{25}$$

with $\overline{W_t}$ being the window average and multiply (possibly clipped) the variance parameters by it. Note that we assume here monotonic improvement of M and $R \in [0, 1]$. This scheme thus poses a dynamic adaptation of the variance parameters based on second-order information of the improvement of the average reward of M. It follows the intuition that if the improvement slows down, a decrease of the variance gives the agent more control over the sampled actions and allows for a more exploitation-oriented behavior. On the other side, when the improvement accelerates, it appears reasonable to prevent too greedy a behavior by increasing the uncertainty in the predicted actions. The same scheme can furthermore also be applied to parameters such as ϵ, which plays a similar role to the variance.

In conclusion, extending the PPO training with a memory of the best perceived actions prevents good solutions of the control problem to be lost, gives the agent access to the best available advantage estimate, improves convergence and allows to dynamically scale the variance parameters of respective distributions from which actions are sampled. While we introduce this variant of the PPO algorithm for our specific application, we believe that it would generalize to other applications of reinforcement learning.

V. APPLYING THE METHOD

In this section, we will now introduce two quantum control scenarios that were recently explored via machine learning [8, 9]. We show how one can apply our method to tackle some interesting learning tasks arising in these control settings by leveraging physical domain knowledge.

A. Quantum Memory

One particular instance of a quantum control problem is the problem of storing the state of a qubit, i.e. a two-level system used in quantum computation. This is, next to quantum error correction, a very relevant problem in quantum computation. Here we assume that our qubit is embedded in some environment, called the bath, such that the complete system lives in the Hilbert space

$$\mathcal{H} = \mathcal{H}_S \otimes \mathcal{H}_B$$

with the subsripts S and B denoting the space of the system and bath respectively. If we let this system evolve freely, decoherence effects will over time destroy the state of the qubit. Hence the question is how we can intervene to prevent the loss of the state in the presence of the environment or, for computer scientific purposes, the noise where we assume to have control over the qubit only. From a quantum computing perspective, we would like to implement a gate that performs the identity function over a finite time interval.

Qubit states are commonly represented as points on the Bloch sphere [4] and the effect of the environment on the qubit can in this picture be perceived as some rotation that drives the qubit away from its original position. To counter this problem we must hence determine a good rotation at each time step such that we negate the effect of the environment. So, our goal is to dynamically decouple the qubit from its bath by performing these rotations. The rotation of a qubit is defined as

$$R_n(\alpha) = e^{-i\frac{\alpha}{2} n\sigma}$$

with n being a unit vector specifying the rotation axis, α denoting the rotation angle and σ the 'vector' of the stacked Pauli matrices $\sigma_{\{x,y,z\}}$ [20]. Thus our controlled time evolution operator per time step t becomes

$$U(n_t, \alpha_t) = e^{-i\Delta t(H_0 + \frac{\alpha_t}{2\Delta t} n_t \sigma \otimes I_B)},$$

expressing that we only apply the rotation to the qubit, but not the bath. The noise Hamiltonian H_0 here reflects the effect of the bath on the qubit and I_B simply denotes the identity of size of the dimensionality of $\mathcal{H}_B$ such that the Kronecker product yields a matrix of equal size to H_0.

One possible metric to quantify how well we were able to preserve the qubit's state is

$$D(U, I) = \sqrt{1 - \frac{1}{d_S d_B} \|\mathsf{Tr}_S(U)\|_{\mathsf{Tr}}}$$

with U denoting the total evolution operator, I the identity and Tr_S is the partial trace over the system [21]. $\|U\|_{\mathsf{Tr}} = \mathsf{Tr}\sqrt{U^\dagger U}$ is the trace or nuclear norm. This distance measure is minimized by the ideal case $U = I_S \otimes U_B$ with an arbitrary unitary U_B acting on the bath. Thus, the problem we would like to solve is a special instance of quantum control and can be formulated as

$$\min_{\{(n_t,\alpha_t)\}} D(U(\{(n_t,\alpha_t)\}, I).$$

Having introduced the quantum memory scenario, we now turn to a description of possible reinforcement learning tasks in this context. We present three different formulations of the setting which we will in the following refer to as the discrete, semi-continuous and continuous case. These formulations differ in the parametrization of the rotation $R_n(\alpha)$ that is to be performed at each time step.

Discrete case: It is known from analytical derivations that the Pauli matrices $\sigma_{\{0,x,y,z\}}$ give rise to optimal sequences under certain ideal conditions [22, 23], where at each time step exactly one of the rotations $R_{\{0,x,y,z\}} = e^{-i\frac{\pi}{2}\sigma_{\{0,x,y,z\}}}$ is performed. σ_0 hereby denotes the identity. Hence, in the simplest formulation we can define the problem as choosing one of the four Pauli matrices at each time step. This formulation then leads to a sequence space S of size $|S| = 4^T$ being exponential in the sequence length T. This is the formulation which was also used in recent work on quantum memory [9].

Semi-continuous case: While the class of sequences introduced above is provably ideal under certain conditions, one might be interested in allowing the agent more freedom to facilitate its adaption to more adverse conditions. This can in a first step be achieved by allowing the agent full control over the rotation angle while keeping the discrete formulation for the rotation axis. That means that at each time step, the agent will have to choose a rotation axis from $\sigma_{\{0,x,y,z\}}$ as before, but now must also predict the rotation angle $\alpha \in [0, 2\pi]$. As α can take infinitely many values, this formulation of the problem now yields a sequence space S of inifinite size, making it much harder from a reinforcement learning perspective. To lighten this burden we can make use of the fact that we know that in principle a rotation around π is ideal. Thus, we will interpret the output of the agent as the deviation from π $\Delta\alpha \in [-\pi, \pi]$. This should facilitate learning progress even in the early training phase.

Continuous case: Finally, we can of course also allow the agent full control over both the rotation angle and axis. This formulation of the problem requires the agent to predict a unit vector $n \in \mathbb{R}^3$ and a corresponding rotation angle α for each time step. It is clear that without any prior knowledge it will be very difficult for the agent to identify the 'right corner' of this infinite sequence space. We hence propose to again leverage the knowledge about Pauli rotations being a good standard choice by having the agent predict a Pauli rotation together with the deviation in n and α. While for α we have already seen how this can be easily achieved, n requires slightly more insight. As is customary in quantum physics, every state of a two-dimensional particle $|\psi\rangle$ can be represented by choosing two angles $\theta \in [0, \pi]$ and $\phi \in [0, 2\pi]$, yielding the three-dimensional real unit Bloch vector

$$b = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}.$$

We can hence use this formulation to parameterize n by θ and ϕ. It is easy to see that the Pauli rotations correspond to the unit vectors that equal a one-hot encoding of the Pauli matrices such that we obtain the following identities

$$\theta_x = \theta_y = \phi_y = \frac{\pi}{2} \text{ and}$$
$$\phi_x = \phi_z = \theta_z = 0$$

with periodicity in π. We can now leverage this knowledge by translating the Pauli rotation axis chosen by the agent into its Bloch expression and requring it to predict the deviations $\Delta\theta$ and $\Delta\phi$. In this way the agent has access to the full axis space. As with the rotation angle, this formulation has the effect that the agent starts learning from a reasonable baseline.

B. Ground state transitions

Another scenario that was recently addressed in an anlysis of the characteristics of the optimization problem behind controlling systems out of equilibrium [8] is the transition between ground states of different Hamiltonians. The considered class of Hamiltonians was thereby defined to be the class of Ising Hamiltonians given by

$$\begin{aligned} H(J,g,h) = J\sum_{i=1}^{L-1} I^{\otimes i-1} \otimes \sigma_x \otimes \sigma_x \otimes I^{\otimes L-(i+1)} \\ + g\sum_{i=1}^{L} I^{\otimes i-1} \otimes \sigma_z \otimes I^{\otimes L-i} \\ + h\sum_{i=1}^{L} I^{\otimes i-1} \otimes \sigma_x \otimes I^{\otimes L-i} \end{aligned}$$

where the $\sigma_{\{x,y,z\}}$ again denote the Pauli matrices and L specifies the number of particles. In this setting we furthermore set $J = g = -1$, leaving h as the only free parameter specifing the strength of the magnetic field

represented by σ_x. From a mathematical perspective, the ground state $|E_{min}(h)\rangle$ of a given Hamiltonian $H(h)$ is then defined as the eigenvector of $H(h)$ corresponding to its lowest eigenvalue.

In the considered scenario we now choose the initial and target states to be $|\psi_i\rangle = |E_{min}(h_i)\rangle$ and $|\psi^*\rangle = |E_{min}(h^*)\rangle$ respectively where $h_i \neq h^*$ are particular choices of h. The controlled time evolution operator is then simply defined to be the one generated by $H(h)$ as given by

$$U(h_t) = e^{-i\Delta t/\hbar H(h_t)}$$

where we assume h_t to be time dependent. The closeness between the state resulting from the controlled time evolution $|\psi(T)\rangle$ and the target state $|\psi^*\rangle$ is measured by their squared overlap

$$S_2(\psi^*, \psi(T)) = |\langle\psi^*, \psi(T)\rangle|^2,$$

similar to what was shown in Section II. We thus obtain the optimization problem formulation

$$\max_{\{h_t\}} S_2(\psi^*, \psi(\{h_t\}))$$

representing the quantum control optimization problem.

Next, we will introduce some RL tasks arising in this control scenario. Similarly to the the taxonomy introduced above, we will thereby distinguish between a discrete, a continuous and a constrained case. These cases correspond to different domains of possible values for the time dependent field strengths h_t. All of them however have in common that we assume a maximal magnitude h_{max} of the field strength such that $h_t \in [-h_{max}, h_{max}]$ holds. This is simply done to reflect the fact that in real experiments infinite field strengths are impossible to achieve.

Discrete case: Knowing that the potentially continuous domain of our control parameter h_t is upper and lower bounded by $\pm h_{max}$, we can apply Pontryagin's principle to limit ourselves to actions $s_t \in \{-h_{max}, h_{max}\}$. We thus obtain a reinforcement learning problem where at each point in time the agent has to make a binary decision. While this is arguably the easiest conceivable scenario, the sequence space still is of size $|S| = 2^T$.

Continuous case: Although we know from theory that optimal sequences will comprise only extremal values of the control parameter h_t, it is still interesting to examine if the agent is able to discover this rule by itself. In this case we hence allow the agent to freely choose $h_t \in [-h_{max}, h_{max}]$ which again presents us with a sequence space of infinite size. Following our reasoning from the continuous quantum memory case, we cast the problem as learning the deviation $\Delta h \in [0, h_{max}]$ from $\pm h_{max}$. Hence, for each time t the agent must predict the deviation Δh and decide to which of the two extremal values the deviation should be applied. This formulation clearly allows the agent to predict any value in $[-h_{max}, h_{max}]$.

TABLE I. The best values of $D(U, I)$ found by or method for the discrete, semi-continuous and continuous quantum memory learning tasks together with baseline results. The reference values were taken from [9] and computed with the corresponding algorithm for $T = 0.512$ and $\Delta t = 0.002$. Lower values are better.

	$\Delta t = 0.002$		$\Delta t = 0.004$	
$T =$	0.064	0.512	0.256	0.512
Ref.	$7 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$8 \cdot 10^{-4}$
Disc.	$7 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$8 \cdot 10^{-4}$
Semi-Cont.	$6 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$8 \cdot 10^{-4}$
Cont.	$6 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$4 \cdot 10^{-4}$	$7 \cdot 10^{-4}$

Constrained case: In the continuous case as defined above, we know that the agent should ideally learn to predict deviations of 0 to achieve sequences with extremal values of h_t. We can thus try to make the problem more challenging by imposing an upper bound $B < T|h_{max}|$ on $\sum_t |h_t|$, representing an upper limit of the total field strength. Imposing such a bound is not an artificial problem as it could for instance be used to model energy constraints in real experiments. This constraint can easily be realized by defining the reward of a sequence s to be

$$R(s) = \begin{cases} S_2(\psi^*, \psi(s)) \text{ if } \sum_t |h_t| \leq B \\ 0 \text{ else.} \end{cases}$$

This constraint requires the agent to learn how to distribute a global budget over a given sequence where it can maximally allocate an absolute field strength of $|h_{max}|$ to each action s_t. As it is not clear which values are optimal in principle for a given bound B, instead of a deviation we here let the agent directly predict the field strength h_t.

VI. RESULTS

In this section we will now present numerical results for the two application scenarios presented above to illustrate the validity of our method and the usefulness of the MPPO algorithm. As we did not have at our disposal real physical experiments implementing these scenarios, the results presented in the following are based numerical simulations.

FIG. 1. The best 10 sequences found for the discrete learning problem with varying parameters of T and Δt. It is clearly visible how the best sequences for each setting share common structural properties and also exhibit recurring patterns making them amenable to machine learning models.

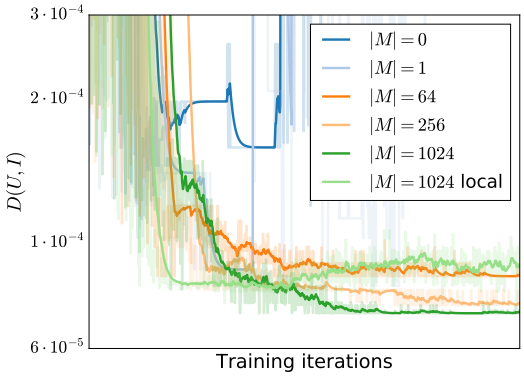


FIG. 2. A comparison of the convergence behavior of the best results sampled per iteration for different sizes of the memory, no memory and a memory with the L^{CLIP} loss applied to the invididual c_t for $T = 0.064$ and $\Delta t = 0.002$. The convergence becomes more stable with larger memory and updates based on the entire sequences lead to convergence to better sequences.

A. Quantum Memory

For the quantum memory scenario, we investigate the performance of our algorithm for different lengths of the discrete time step, total evolution times and across the three formulations of the problem described above. More concretely, we explore the method's behavior for a discrete time evolution with $\Delta t \in \{0.002, 0.004\}$, $T \in \{0.064, 0.256, 0.512, 1.024\}$ and a physical system consisting of one memory qubit coupled to a bath of four qubits with up to three-body interactions to allow for a comparison with the baseline results [9]. We refer the interested reader to this article for a more precise description of the physical setup. While we ultimately would like to optimize $D(U(c), I)$ as defined above, we used $1 - D(U(c), I)$ as a reward signal to obtain an $R(c) \in [0, 1]$. We furthermore shifted the reward such that a uniformly random policy obtains zero reward on average.

As the three learning tasks introduced for this scenario differ in their action domains, we need to use a different probabilistic modelling for each setting. For the discrete case, we simply model each element c_t of a sequence c by a categorical distribution such that we have

$$p(c) = \prod_t Cat(c_t \in \{I, X, Y, Z\} | \{p_{I,t}, p_{X,t}, p_{Y,t}, p_{Z,t}\})$$

for a complete sequence c. In the semi-continuous case we employ a mixture-of-Gaussians distribution which yields

$$p(c) = \prod_t \sum_{i \in \{I,X,Y,Z\}} p_{i,t} \mathcal{N}(c_t = \Delta\alpha | \mu_{i,t}, \sigma_t).$$

This can easily be generalized to the continuous case via a multivariate mixture-of-Gaussians distribution with diagonal covariance matrix such that we obtain

$$p(c) = \prod_t \sum_{i \in \{I,X,Y,Z\}} p_{i,t} \mathcal{N}(c_t = \{\Delta\alpha, \Delta\theta, \Delta\phi\} | \boldsymbol{\mu}_{i,t}, \sigma_t I).$$

Note that we have omitted here the dependence on the weights Θ for the sake of brevity. As discussed in Section III, we use an LSTM to parameterize these probability densities. More concretely, we use a two-layer LSTM and use its output as input to a softmax layer to predict the $p_{i,t}$. From this output state and the relevant parts of the output from the previous time step we also predict the μ_i for $\Delta\alpha$ in the semi-continuous case and analogously for $\Delta\theta$ and $\Delta\phi$ in the continuous case. For every deviation output we train an individual output unit for each discrete rotation. For the semi-continuous and continuous tasks, we scale the standard deviation σ_t and PPO parameter ϵ over the course of the optimization using our introduced adaption scheme with a window size of 10 and optimize the loss function with the Adam optimizer [24].

The scores $D(U(c), I)$ of the best sequences found in our numerical experiments are listed in Table I. They clearly show that our method is able to achieve the same or slightly better results as the baseline algorithm from [9] for all considered settings and learning tasks. For the semi-continuous case, we observe that for the setting involving the shortest sequences slightly better sequences than in the discrete case can be found. For longer sequences the performance is on par with the discrete sequences. The same in principle holds for the continuous case with the exception of the results for $T = 0.512$ and $\Delta t = 0.004$ being slightly better then for the other two cases. Overall we can conclude that our method finds sequences several orders of magnitude better than those a random policy generates, which are generally in the interval $[0.1, 0.5]$, showing that in all cases LSTMs trained by the MPPO algorithm seem to perform quite well. We can also see that the discrete sequences pose a strong baseline that is hard to beat even with a fully continuous approach and in fact we observed the predicted deviations

TABLE II. The best values of S_2 obtained by our method for the discrete, continuous and constrained ground state transition learning problems with reference values taken from [8]. Higher values are better.

	$T=0.5$	$T=1$	$T=3$
Ref. $(L=1)$	0.331	0.576	1
Disc. $(L=1)$	0.331	0.576	1
Cont. $(L=1)$	0.331	0.576	1
Disc. $(L=5)$	0.57	0.767	1
Cont. $(L=5)$	0.57	0.768	1
Const. $(B=20)$	0.313	–	–
Const. $(B=30)$	0.322	–	–
Const. $(B=40)$	–	0.572	–
Const. $(B=50)$	–	0.577	–
Const. $(B=60)$	–	0.577	–
Const. $(B=120)$	–	–	1
Const. $(B=140)$	–	–	1
Const. $(B=160)$	–	–	1

to converge to very small values. The results furthermore support the conjecture that good sequences share common structure and local patterns that can be learned which is also illustrated in Figure 1. Here, the best 10 sequences found during the training process in the discrete case for three different settings are shown, illustrating the high degree of structure that the best sequences exhibit. The structural similarities become more apparent with growing sequence length. Interestingly, in all cases the best sequences only make use of two of the four Pauli rotations and less surprisingly never use the identity 'rotation'. In Figure 2 we show the effect of different sizes of the memory M on the convergence of the best sequences in the discrete case for otherwise constant optimization parameters. As can be seen, when not using a memory or only storing the best sequence, the optimization diverges. For larger sizes of the memory, the algorithm converges to better and better sequences, arriving at the best sequence found for this setting with a memory of 1024 sequences. We also compared the performance of our algorithm to updates computed not over complete sequences but over the single control parameters c_t as done in the PPO algorithm for $|M| = 1024$. While also the latter performs well, only the former converges to the best sequence.

B. Ground state transition

In the ground state transition setting, we evaluate our method for times $T \in \{0.5, 1, 3\}$ with $\Delta t = 0.05$ and an initial $h_i = -2$, target $h^* = 2$ as well as $|h_{max} = 4|$ to achieve comparability with the baseline results [8]. For the discrete and continuous case,

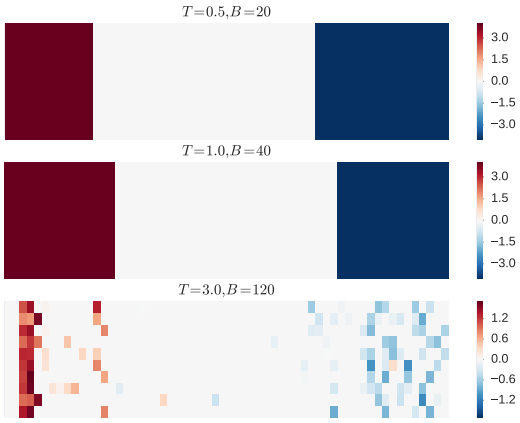


FIG. 3. The 10 best sequences found for different values of T and a maximal field strength B amounting to half of the maximally possible. While the best sequences for $T = 0.5$ and $T = 1.0$ are very similar und use the maximal possible absolute field strength, the best sequences for $T = 3.0$ use much smaller pulses.

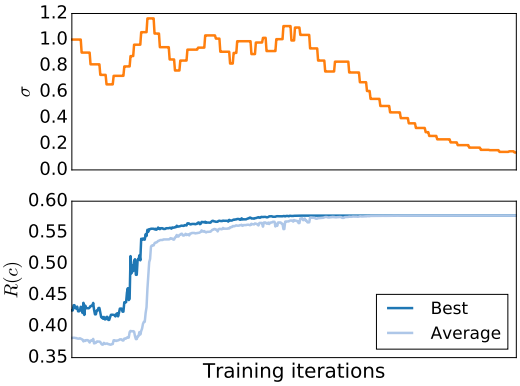


FIG. 4. The convergence the best and average reward per iteration together with the dynamically adapted σ for the constrained scenario with $T = 1.0$ and $B = 60$.

we consider systems of size $L = 1$ and $L = 5$ and $B \in \{20, 30, 40, 50, 60, 100, 120, 140\}$ with $L = 1$ for the constrained case. Since the overlap S_2 as defined above already lies in the interval $[0, 1]$, we used it directly as reward function, again shifting it such that a uniformly random policy achieved zero reward.

The probabilistic modelling of the sequences is similar to the quantum memory case in that we use a categorical distribution for the discrete case and a mixture-of-Gaussians for both the continuous and constrained tasks. Thereby, we model the probability density of the deviations Δh_t in the continuous case and the predicted absolute value of h_t in the constrained case. The distributions are parameterized in the same way as above, namely by a two layer LSTM form whose output state both the discrete probabilities and the means for both discrete cases as predicted. The optimization is conducted as in the

quantum memory scenario.

The results of our numerical experiments are listed in Table II. As shown, our method was able to replicate the baseline results from [8] both for the discrete and the continuous formulation of the problem for a system size of $L = 1$ and also performs well for larger systems of $L = 5$ with both versions yielding generally the same results. We indeed found the continuous version to converge to predicting zero deviation as it was expected to. For the constrained case we can see that our method converges to sequences whose performance is surprisingly close to the baseline results even when allowed to use only half of the maximal absolute field strength. For $T = 3.0$ the imposed constraints in fact seem to have no negative effect as apparently already sequences with a very small total field strength suffice to achieve perfect overlap. This is also illustraed by Figure 3 which shows the best 10 sequences found during the training process for $T \in \{0.5, 1.0, 3.0\}$ and B set to half the maximal total field strength. While for the smaller two total times the sequences are very similar and always make use of the maximal field strength or apply no pulse at all, for $T = 3.0$ only the general scheme of applying positive pulses first, then doing nothing and finally applying negativ pulses persists. The individual pulses that are applied are very weak and and entire sequence typically only amounts to a total absolute strength of ~ 6. This phenomenon is likely caused by the fact that the optimization problem in this case becomes significantly easier for longer times [8]. In Figure 4 we display the convergence of the best and average results sampled per iteration together with the dynamic schedule for sigma during the optimization. It can be seen that σ is dynamically increased when the convergence slows down, decreased when it speeds up and finally converges to a stable value as the optimization converges as well. In other scenarios we also observed our adaption scheme to perform similarly to a decayed annealing schedule.

VII. CONCLUSION AND FUTURE WORK

In this work we have tried to introduce quantum physics and especially problems in (black-box or model-free) quantum information and quantum control to a broader audience in the machine learning community and showed how they can be successfully tackled with state-of-the-art reinforcement learning methods. To this end, we have given a brief introduction to quantum control and discussed different aspects of the application of reinforcement learning to it. We have argued that LSTMs are a good choice to model the sequences of control parameters arising in quantum control and shown how black-box quantum control gives rise to a particular reinforcement learning problem for whose optimization policy gradient methods are a natural choice. As a recent and successful variant of policy gradient algorithms, we have adapted the PPO scheme for our application and introduced the MPPO algorithm. We then went on to show how our general method for treating black-box quantum control can be easily combined with physical prior knowledge for two example scenarios and presented numerical results for a range of learning tasks arising in this context. These results showed that our method is able to achieve state-of-the-art performance in different tasks while being able to address problems of discrete and continuous control alike and provided evidence for the hypotheses that machine learning is a good choice for the automated optimization of parameters in experiments.

This work can also be understood to some extent as a contribution to the debate about how much prior knowledge is necessary for machine learning algorithms to perform well in real-world tasks. During the course of this work, we have found it a necessary precondition for the addressed problems in continuous domains to be solvable to incorporate physical domain knowledge such as known good rotation axes and angles. Without this information a reinforcement learning agent would be required to at least implicitly learn about certain laws of physics to not be lost in the infinite action space of which only a negligibly small part results in good solutions. This clearly is out of scope for current models and algorithms without symbolic reasoning capacity and might remain so for some time especially when the data collected by the agent is very small compared to the search space.

Finally, interesting directions of future work would be to apply the method to a real experiment and evaluate its performance there as well as to develop a set of benchmark problems in quantum control to compare the different already existing algorithms on neutral grounds. It would also be interesting to investigate which other problems of relevance yield reinforcement learning problems similarly structured to the formulation presented in this work.

[1] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, Nature **550**, 354 (2017).

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, arXiv preprint arXiv:1712.01815 (2017).

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Nature **549**, 195 (2017).

[4] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," (2002).

[5] P. B. Wigley, P. J. Everitt, A. van den Hengel, J. Bastian, M. A. Sooriyabandara, G. D. McDonald, K. S. Hardman, C. Quinlivan, P. Manju, C. C. Kuhn, *et al.*, Scientific reports **6** (2016).

[6] A. A. Melnikov, H. P. Nautrup, M. Krenn, V. Dunjko,

M. Tiersch, A. Zeilinger, and H. J. Briegel, Proceedings of the National Academy of Sciences , 201714936 (2018).
[7] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, Neurocomputing (2017).
[8] M. Bukov, A. G. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, arXiv preprint arXiv:1705.00565 (2017).
[9] M. August and X. Ni, Physical Review A **95**, 012335 (2017).
[10] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, arXiv preprint arXiv:1707.06347 (2017).
[11] S. Hochreiter and J. Schmidhuber, Neural computation **9**, 1735 (1997).
[12] C. Cohen, B. D. Tannoudji, and F. Laloë, Hermann and Wiley (1977).
[13] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, Journal of magnetic resonance **172**, 296 (2005).
[14] P. Doria, T. Calarco, and S. Montangero, Phys. Rev. Lett. **106**, 190501 (2011).
[15] T. Caneva, T. Calarco, and S. Montangero, Physical Review A **84**, 022326 (2011).
[16] R. J. Williams, in *Reinforcement Learning* (Springer, 1992) pp. 5–32.
[17] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, in *International Conference on Machine Learning* (2015) pp. 1889–1897.
[18] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, in *International Conference on Machine Learning* (2016) pp. 1928–1937.
[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, Nature **518**, 529 (2015).
[20] J. J. Sakurai and E. D. Commins, *Modern quantum mechanics, revised edition* (AAPT, 1995).
[21] G. Quiroz and D. A. Lidar, Phys. Rev. A **88**, 052306 (2013).
[22] L. Viola, E. Knill, and S. Lloyd, Phys. Rev. Lett. **82**, 2417 (1999).
[23] A. M. Souza, G. A. Álvarez, and D. Suter, Phys. Rev. Lett. **106**, 240501 (2011).
[24] D. P. Kingma and J. Ba, arXiv preprint arXiv:1412.6980 (2014).
[25] H. Robbins, in *Herbert Robbins Selected Papers* (Springer, 1985) pp. 169–177.

F Neural-network quantum states, string-bond states and chiral topological states

Authors Ivan Glasser, Nicola Pancotti, Moritz August, Dr. Ivan D. Rodriguez and Prof. Dr. J. Ignacio Cirac

Citation Physical Review X, 8(1):011006, 2018

DOI 10.1103/PhysRevX.8.011006

Summary Recently before the appearance of the article shown here, quantum states based on RBMs had been introduced as an approach to describing the wave function of quantum many-body systems. Based on these findings, we demonstrated in this work that there exist strong connections between particular classes of tensor-network states and neural-network states as given by RBMs. More precisely, we showed that RBMs are string-bond states with non-local geometry and small bond dimension while short-range RBMs are entangled plaquette states. These findings provide some insight about the underlying structure of restricted Boltzmann machines and their capability of efficiently representing many-body quantum states. Based on these results, we found that string-bond states furthermore pose a general approach to increasing the representational power of RBM states and constitute a natural generalization to systems with larger local Hilbert space. We then compared the benefits and disadvantages of the different parameterizations of quantum states and introduced a way of combining them. Doing so makes it possible to leverage both the efficiency of neural-network quantum states and the entanglement structure of tensor networks. This combination thus poses a new representational approach able to approximate the wave function of strongly correlated systems. We demonstrated the advantages of our approach by exactly representing certain states with chiral topological order, which poses a challenge for previous tensor network approaches. We additionally presented numerical evidence that quantum states based on restricted Boltzmann machines can approximate a chiral spin liquid with higher accuracy than local string-bond states and entangled plaquette states. Our findings show the efficiency of restricted Boltzmann machines in describing complex quantum wave functions and additionally might indicate that string-bond states can be a useful tool in more established machine learning areas.

Contribution In this work, the author contributed to the discussion of the relation between machine learning models such as RBMs and TNs. Furthermore, the author contributed to the implementation of the learning algorithm. More precisely, the author analyzed the relation between string-bond states and RBMs, especiallly with respect to the question if deep RBMs, i. e. RBMs consisting of multiple layers, could be expressed as such TNs. The author found this not to be the case. The author also contributed to obtaining a more stable optimization procedure and evaluated numerical frameworks for the implementation. Finally, the author provided feedback in the preparation of the article and was generally involved in the ongoing discussion during all stages of the work.

Neural-Network Quantum States, String-Bond States, and Chiral Topological States

Ivan Glasser,[1] Nicola Pancotti,[1] Moritz August,[2] Ivan D. Rodriguez,[1] and J. Ignacio Cirac[1]
[1]*Max-Planck-Institut für Quantenoptik, Hans-Kopfermann-Straße 1, D-85748 Garching, Germany*
[2]*Department of Informatics, Technical University of Munich, Boltzmannstraße 3, D-85748 Garching, Germany*

(Received 17 October 2017; revised manuscript received 8 December 2017; published 11 January 2018)

Neural-network quantum states have recently been introduced as an Ansatz for describing the wave function of quantum many-body systems. We show that there are strong connections between neural-network quantum states in the form of restricted Boltzmann machines and some classes of tensor-network states in arbitrary dimensions. In particular, we demonstrate that short-range restricted Boltzmann machines are entangled plaquette states, while fully connected restricted Boltzmann machines are string-bond states with a nonlocal geometry and low bond dimension. These results shed light on the underlying architecture of restricted Boltzmann machines and their efficiency at representing many-body quantum states. String-bond states also provide a generic way of enhancing the power of neural-network quantum states and a natural generalization to systems with larger local Hilbert space. We compare the advantages and drawbacks of these different classes of states and present a method to combine them together. This allows us to benefit from both the entanglement structure of tensor networks and the efficiency of neural-network quantum states into a single Ansatz capable of targeting the wave function of strongly correlated systems. While it remains a challenge to describe states with chiral topological order using traditional tensor networks, we show that, because of their nonlocal geometry, neural-network quantum states and their string-bond-state extension can describe a lattice fractional quantum Hall state exactly. In addition, we provide numerical evidence that neural-network quantum states can approximate a chiral spin liquid with better accuracy than entangled plaquette states and local string-bond states. Our results demonstrate the efficiency of neural networks to describe complex quantum wave functions and pave the way towards the use of string-bond states as a tool in more traditional machine-learning applications.

DOI: 10.1103/PhysRevX.8.011006

Subject Areas: Computational Physics, Condensed Matter Physics, Quantum Physics

I. INTRODUCTION

Recognizing complex patterns is a central problem that pervades all fields of science. The increased computational power of modern computers has allowed the application of advanced methods to the extraction of such patterns from humongous amounts of data, and we are witnessing an ever-increasing effort to find novel applications in numerous disciplines. This led to a line of research now called quantum machine learning [1], which is divided into two main branches. The first tries to develop quantum algorithms capable of learning, i.e., to exploit speed-ups from quantum computers to make machines learn faster and better. The second branch, which we consider in this work, uses classical machine-learning algorithms to extract insightful information about quantum systems.

The versatility of machine learning has allowed scientists to employ it in a number of problems, which span from quantum control [2–4] and error correcting codes [5] to tomography [6]. In the last few years, we have also been experiencing interesting developments for some central problems in condensed matter, such as quantum phase classification or recognition [7–10], improvement of dynamical mean field theory [11], enhancement of quantum Monte Carlo methods [12,13], or approximations of thermodynamic observables in statistical systems [14].

An idea that received a lot of attention from the scientific community consists in using neural networks as variational wave functions to approximate ground states of many-body quantum systems [15]. These networks are trained or optimized by the standard variational Monte Carlo (VMC) method, and while a few different neural-network architectures have been tested [15–17], the most promising results so far have been achieved with Boltzmann machines [18]. In particular, state-of-the-art numerical results have

been obtained on popular models with restricted Boltzmann machines (RBM), and recent effort has demonstrated the power of deep Boltzmann machines to represent ground states of many-body Hamiltonians with polynomial-size gap and quantum states generated by any polynomial size quantum circuits [19,20].

Other seemingly unrelated classes of states that are widely used in condensed-matter physics are tensor-network states. In 1D, matrix product states (MPS) can approximate ground states of physical Hamiltonians efficiently [21,22], and their structure has led to both analytical insights over the entanglement properties of physical systems and efficient variational algorithms for approximating them [23–25]. The natural extension of MPS to larger-dimensional systems are projected entangled pair states (PEPS) [26]; however, their exact contraction is $\#P$ hard [27], and algorithms for optimizing them need to rely on approximations. Another approach to define higher-dimensional tensor networks consists in first dividing the lattice into overlapping clusters of spins. The wave function of the spins in each cluster is then described by a simple tensor network. The global wave function is finally taken to be the product of these tensor networks, which introduces correlations among the different clusters. This construction for local clusters parametrized by a full tensor gives rise to entangled plaquette states (EPS) [28–30], while taking one-dimensional clusters of spins each described by a MPS leads to a string-bond states (SBS) Ansatz [31,32]. These states can be variationally optimized using the VMC method [31,33] and have been applied to 2D and 3D systems.

All these variational wave functions have been successful in describing strongly correlated, quantum many-body systems, including topologically ordered states. The toric code [34] is a prototypical example, which can be written exactly as a PEPS [35], an EPS [30], a SBS [31], or a short-range RBM [36]. This shows that, in some cases, tensor-network and neural-network quantum states can be related. Indeed, it was recently shown that local tensor networks can be represented efficiently by deep Boltzmann machines [19,20,37]. However, not every topological state can be easily represented by local tensor networks. A class of states for which this is challenging are chiral topological states breaking time-reversal symmetry. Such states were first realized in the context of the fractional quantum Hall (FQH) effect [38], and significant progress has since been made towards the construction of lattice models displaying the same physics, either in Hamiltonians realizing fractional Chern insulators [39–44] or in quantum antiferromagnets on several lattices [45–48]. One approach to describe the wave function of these antiferromagnets is to use parton-constructed wave functions [49–52]. It has also been suggested to construct chiral lattice wave functions from the FQH continuum wave functions, the paradigmatic example being the Kalmeyer-Laughlin wave function [53]. Efforts to construct chiral topological states with PEPS have been undertaken recently [54–58], but the resulting states are critical. In the noninteracting case, it has moreover been proven that the local parent Hamiltonian of a chiral fermionic Gaussian PEPS has to be gapless [55].

In this work, we show that there is a strong relation between restricted Boltzmann machines and tensor-network states in arbitrary dimensions. We demonstrate that short-range RBM are a special subclass of EPS, while fully connected RBM are a subclass of SBS with a flexible nonlocal geometry and low bond dimension. This relation provides additional insights over the geometric structure of RBM and their efficiency. We discuss the advantages and drawbacks of RBM and SBS and provide a way to combine them together. This generalization in the form of nonlocal string-bond states takes advantage of both the entanglement structure of tensor networks and the efficiency of RBM. It allows for the description of states with larger local Hilbert space and has a flexible geometry. Moreover, it can be combined with more traditional Ansatz wave functions that serve as an initial approximation of the ground state.

We then apply these methods to the challenging problem of approximating chiral topological states. We prove that any Jastrow wave function, and thus the Kalmeyer-Laughlin wave function, can be written exactly as a RBM. Moreover, we show that a remarkable accuracy can be achieved numerically with much less parameters than is required for an exact construction. We numerically evaluate the power of EPS, SBS, and RBM to approximate the ground state of a chiral spin liquid for which the Laughlin state is already a good approximation [45], and we find that RBM and nonlocal SBS are able to achieve lower energy than the Laughlin wave function. By combining these classes of states with the Laughlin wave function, we are able to reach even lower energies and to characterize the properties of the ground state of the model.

The paper is organized as follows: In Sec. II, we introduce the variational Monte Carlo method and show how it can be used to optimize both tensor-network and neural-network states. In Sec. III, the mapping between RBM, EPS, and SBS is derived, and its geometric implications are discussed. Finally, we apply these techniques to the approximation of chiral topological states in Sec. IV.

II. VARIATIONAL MONTE CARLO WITH TENSOR-NETWORK AND NEURAL-NETWORK STATES

A. Variational Monte Carlo method

Given a general Hamiltonian H, one of the main challenges of quantum many-body physics is to find its ground state $|\psi_0\rangle$ satisfying the Schrödinger equation $H|\psi_0\rangle = E_0|\psi_0\rangle$. This eigenvalue problem can be mapped to an optimization problem through the variational principle, stating that the energy of any quantum state is higher than the energy of the ground state. A general pure quantum state on a lattice with N spins can be expressed in the basis

spanned by $|s_1, \ldots, s_N\rangle$, where s_i are the projections of the spins on the z axis, as

$$|\psi\rangle = \sum_{s_1,\ldots,s_N} \psi(s_1, \ldots, s_N)|s_1, \ldots, s_N\rangle. \tag{1}$$

Finding the ground state amounts to finding the exponentially many parameters $\psi(s_1, \ldots, s_N)$ minimizing the energy, which can only be done exactly for small sizes. Instead of searching for the ground state in the full Hilbert space, one may restrict the search to an Ansatz class specified by a particular form for the function $\psi_w(s_1, \ldots, s_N)$ depending on polynomially many variational parameters w. The VMC method [59,60] provides a general algorithm for optimizing the energy of such a wave function. One can compute the energy by expressing it as

$$E_w = \frac{\langle\psi|H|\psi\rangle}{\langle\psi|\psi\rangle} = \sum_{\mathbf{s}} p(\mathbf{s}) E_{\mathrm{loc}}(\mathbf{s}), \tag{2}$$

where $\mathbf{s} = s_1, \ldots, s_N$ is a spin configuration, $p(\mathbf{s}) = [|\psi_w(\mathbf{s})|^2/(\sum_{\mathbf{s}}|\psi_w(\mathbf{s})|^2)]$ is a classical probability distribution, and the local energy $E_{\mathrm{loc}}(\mathbf{s}) = \sum_{\mathbf{s}'}\langle\mathbf{s}|H|\mathbf{s}'\rangle\{\psi_w(\mathbf{s}')/[\psi_w(\mathbf{s})]\}$ can be evaluated efficiently for Hamiltonians involving few-body interactions. The energy is therefore an expectation value with respect to a probability distribution p that can be evaluated using Markov chain Monte Carlo sampling techniques such as the Metropolis-Hastings algorithm [61,62]. The second ingredient required to minimize the energy with respect to the parameters w is the gradient of the energy, which can be expressed in a similar form since

$$\frac{\partial E_w}{\partial w_i} = 2\sum_{\mathbf{s}} p(\mathbf{s})\Delta_{w_i}(\mathbf{s})^*(E_{\mathrm{loc}}(\mathbf{s}) - E_w), \tag{3}$$

where we have defined $\Delta_{w_i}(\mathbf{s}) = \{1/[\psi_w(\mathbf{s})]\}\{[\partial\psi_w(\mathbf{s})]/\partial w_i\}$ as the log-derivative of the wave function with respect to some parameter w_i. This is also an expectation value with respect to the same probability distribution p and can therefore be sampled at the same time, which allows for the use of gradient-based optimization methods. At each iteration, the energy and its gradient are computed with Monte Carlo, the parameters w are updated by small steps in the direction of the negative energy derivative ($w_i \leftarrow w_i - \alpha[\partial E_w/(\partial w_i)]$), and the process is repeated until convergence of the energy. The VMC method, in its simplest form, only requires the efficient computation of $\{[\psi_w(\mathbf{s}')]/\psi_w(\mathbf{s})\}$ for two spin configurations s and s', as well as the log-derivative of the wave function $\Delta_w(\mathbf{s})$. More efficient optimization methods can be used, such as conjugate-gradient descent, stochastic reconfiguration [63,64], the Newton method [65], or the linear method [66–68].

At this point, one has to choose a special form for the wave function ψ_w. One of the traditional variational wave functions for a many-body quantum system is a Jastrow wave function [59,69], which consists, in its most general form, of a product of wave functions for all pairs of spins:

$$\psi_w(\mathbf{s}) = \prod_{i<j} f_{ij}(s_i, s_j), \tag{4}$$

where each f_{ij} is fully specified by its four values $f_{ij}(s_i, s_j)$, $s_i,\ s_j \in \{-1, 1\}$. Such an Ansatz does not presuppose a particular local geometry of the many-body quantum state: In general, this Ansatz can be nonlocal because of the correlations between all pairs of spins [Fig. 1(a)]. A local structure can be introduced by choosing a form for f_{ij} that decays with the distance between positions i and j.

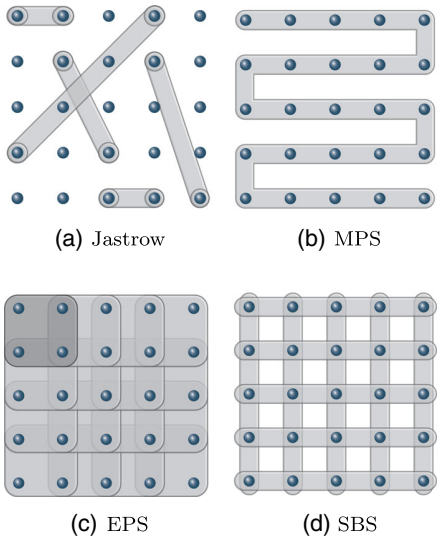


FIG. 1. Geometry of Ansatz wave functions: (a) Jastrow wave functions include correlations within all pairs of spins. (b) MPS in 2D cover the lattice with one snake. (c) EPS include all spin correlations within each plaquette (2×2 on the figure) and mediate correlations between distant spins through overlapping plaquettes. (d) SBS cover the lattice with many 1D strings on which the interactions within spins are captured by a MPS.

B. Variational Monte Carlo method with tensor networks

In condensed-matter physics, important assets to simplify the problem are the geometric structure and locality of physical Hamiltonians. In 1D, it has been proven that ground states of gapped local Hamiltonians have an entanglement entropy of a subsystem that grows only like the boundary of the subsystem [21]. States satisfying such an area law can be efficiently approximated by MPS [22]. Matrix product states are one-dimensional tensor-network states whose wave function for a spin configuration reads

$$\psi_w(\mathbf{s}) = \mathrm{Tr}\left(\prod_{j=1}^{N} A_j^{s_j}\right). \tag{5}$$

For each spin and lattice site, the matrix $A_i^{s_i}$ of dimension $D \times D$, where D is called the bond dimension, contains the variational parameters. Matrix product states can be efficiently optimized using the density matrix renormalization group (DMRG) [70], but the previously described VMC method can also be applied [31,33] by observing that the ratio of two configurations is straightforward to compute and that the log-derivative with respect to some matrix $A_k^{s'_k}$ is given by

$$\Delta_{A_k^{s_{k'}}}(\mathbf{s}) = \frac{\delta_{s_k,s_{k'}}(A_{k+1}^{s_k}\ldots A_N^{s_N}A_1^{s_1}A_{k-1}^{s_{k-1}})^\top}{\mathrm{Tr}(A_1^{s_1}\ldots A_N^{s_N})}. \tag{6}$$

In some cases, this method is less likely to be trapped in a local minimum than DMRG since all coefficients can be updated at once. In addition, the cost only scales as $O(D^3)$ in the bond dimension for periodic boundary conditions.

In higher dimensions, matrix product states can be defined by mapping the system to a line [Fig. 1(b)]. The problem of this construction is evident from Fig. 1(b). Spins that sit close to each other might be separated by a long distance on the line; the Ansatz thus fails to reproduce the local structure of the state, which leads to an exponential scaling of the computing resources needed with the system size [71]. The natural extensions of MPS to 2D systems are projected entangled pair states (PEPS) [26], for which the wave function can be written as a contraction of local tensors on the 2D lattice. While PEPS have been successful in describing strongly correlated quantum many-body systems, their exact contraction is $\#P$ hard [27], and their optimization cannot rely on the standard VMC method without approximations. In the following, we instead consider other classes of tensor-network states in more than one dimension for which the exact computation of the wave function is efficient, which allows for the direct use of the VMC method.

One approach consists in cutting a lattice in P small clusters of n_p spins, or plaquettes, and constructing the wave function exactly on each plaquette. The wave function of the full quantum system is then taken to be the product of the wave functions in each plaquette, in a mean-field fashion. Choosing overlapping plaquettes allows one to go beyond the mean field and include correlations between different plaquettes [Fig. 1(c)]. The wave function of such an EPS (also called a correlated product state) is written as [28–30]

$$\psi_w(\mathbf{s}) = \prod_{p=1}^{P} C_p^{\mathbf{s}_p}, \tag{7}$$

where a coefficient $C_p^{\mathbf{s}_p}$ is assigned to each of the 2^{n_p} (for spin-1/2 particles) configurations $\mathbf{s}_p = s_{a_1},\ldots,s_{a_{n_p}}$ of the spins on the plaquette p. Each C_p can be seen as the most general function on the Hilbert space corresponding to the spins in plaquette p. The accuracy can be improved by enlarging the size of the plaquettes, and the Ansatz is exact once the size of the plaquettes reaches the size of the lattice (which can only be achieved on small lattices). Moreover, once the spin configuration $\mathbf{s}_p$ is fixed, the log-derivative of the wave function with respect to the variational parameters is simply

$$\Delta_{C_p^{s_p}}(\mathbf{s}) = \frac{1}{C_p^{\mathbf{s}_p}}, \tag{8}$$

which is efficient to compute.

EPS are limited to small plaquettes since, for each plaquette, the number of coefficients scales exponentially with the size of the plaquette. However, one can generalize this Ansatz by describing the state of clusters of spins by a MPS, avoiding the exponentially many coefficients needed. The lattice is now cut in overlapping 1D strings, which can mediate correlations on longer distances compared to local plaquettes [Fig. 1(d)]. The resulting Ansatz is a SBS [31] defined by a set of strings $i \in S$ (each string i is an ordered subset of the set of spins) and a MPS for each string:

$$\psi_w(\mathbf{s}) = \prod_i \mathrm{Tr}\left(\prod_{j\in i} A_{i,j}^{s_j}\right). \tag{9}$$

The descriptive power of this Ansatz is highly dependent on the choice of strings: For example, by using small strings covering small plaquettes and a large bond dimension, it includes EPS, whereas a single long string in a snake pattern includes MPS in 2D. In 3D, it has been used by choosing strings parallel to the axes of the lattice [32]. Since the form of the wave function is a product of MPS, the log-derivative with respect to some elements present in one of the MPS is simply the log-derivative for the corresponding MPS [Eq. (6)]. The VMC procedures for optimizing SBS and MPS thus have the same cost. In addition, the ratio of two configurations that differ only by a few spins can be computed by considering only the strings including these spins, which speeds up the computation considerably. Let us note that a SBS can be mapped analytically to a MPS but that the resulting MPS would have a bond dimension exponential in the number of strings.

C. Variational Monte Carlo method with neural networks

Recently, it was realized that the VMC method can be viewed as a form of learning, which motivated the use of another class of seemingly unrelated states for describing the ground state of many-body quantum states: Neural-network quantum states [15] are quantum states for which the wave function has the structure of an artificial neural network. While a few different networks have been investigated [6,15–17], the most promising results so far have been obtained with Boltzmann machines [18]. Boltzmann

machines are types of generative stochastic artificial neural networks that can learn a distribution over the set of their inputs. In quantum many-body physics, the inputs are spin configurations, and the wave function is interpreted as a (complex) probability distribution that the networks try to approximate. Boltzmann machines consist of two sets of binary units (classical spins): the visible units v_i, $i \in \{1, \ldots, N\}$, corresponding to the configurations of the original spins in a chosen basis, and hidden units h_j, $j \in \{1, \ldots, M\}$, which introduce correlations between the visible units. The whole system interacts through an Ising interaction, which defines a joint probability distribution over the visible and hidden units as the Boltzmann weight of this Hamiltonian:

$$P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{\mathcal{H}(\mathbf{v},\mathbf{h})}, \tag{10}$$

where the Hamiltonian $\mathcal{H}$ is defined as

$$\mathcal{H} = \sum_j a_j v_j + \sum_i b_i h_i + \sum_{i<j} c_{ij} v_i v_j + \sum_{i,j} w_{ij} h_i v_j + \sum_{i<j} d_{ij} h_i h_j,$$

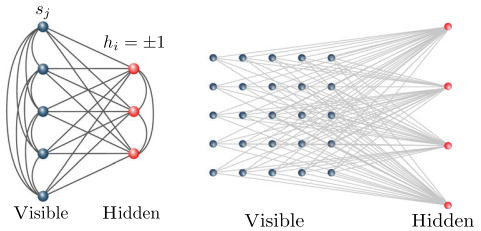
and Z is the partition function. The marginal probability of a visible configuration is then given by summing over all possible hidden configurations:

$$P(\mathbf{v}) = \sum_{\mathbf{h}} \frac{1}{Z} e^{\mathcal{H}(\mathbf{v},\mathbf{h})}, \tag{11}$$

and we take this quantity as the Ansatz for the wave function: $\psi_w(\mathbf{s}) = P(\mathbf{s})$. The variational parameters are the complex parameters of the Ising Hamiltonian. In the case where there are interactions between the hidden units [Fig. 2(a)], the Boltzmann machine is called a deep Boltzmann machine. It has been shown that deep Boltzmann machines can efficiently represent ground states of many-body Hamiltonians with polynomial-size gaps, local tensor-network states, and quantum states generated by any polynomial-size quantum circuits [19,20,37]. On the other hand, computing the wave function $\psi_w(\mathbf{s})$ of such a deep Boltzmann machine in the general case is intractable because of the exponential sum over the hidden variables, so the VMC method cannot be applied to deep Boltzmann machines without approximations. We therefore turn to the investigation of restricted Boltzmann machines (RBM), which only include interactions between the visible and hidden units (as well as the one-body interaction terms that correspond to biases). In this case, the sum over the hidden units can be performed analytically, and the resulting wave function can be written as (here we take the hidden units to have values ± 1):

$$\psi_w(\mathbf{s}) = e^{\sum_j a_j s_j} \prod_i \cosh\left(b_i + \sum_j w_{ij} s_j\right). \tag{12}$$

RBM can represent many quantum states of interest, such as the toric code [36], any graph state, cluster states, and coherent thermal states [19]; however, the possibility of efficiently computing $\psi_w(\mathbf{s})$ prevents it from approximating all PEPS and ground states of local Hamiltonians [19]. On the other hand, since computing $\psi_w(\mathbf{s})$ and its derivative is very efficient, RBM can be optimized numerically via the VMC method.



(a) Deep Boltzmann machine (b) Restricted Boltzmann machine in 2D

FIG. 2. (a) Boltzmann machines approximate a probability distribution by the Boltzmann weights of an Ising Hamiltonian on a graph including visible units (corresponding to the spins s_j) and hidden units h_i, which are summed over. (b) Restricted Boltzmann machines (here in 2D) only include interactions between the visible and the hidden units.

III. RELATIONSHIP BETWEEN TENSOR-NETWORK AND NEURAL-NETWORK STATES

While the machine-learning perspective that leads to the application of Boltzmann machines to quantum many-body systems seems quite different from the information-theoretic approach to the structure of tensor-network states, we see that they are in fact intimately related. It was recently shown that, while fully connected RBM can exhibit volume-law entanglement, contrary to local tensor networks, all short-range RBM satisfy an area law [72]. Moreover, short-range and sufficiently sparse RBM can be written as a MPS [37], but doing so for a fully connected RBM would require an exponential scaling of the bond dimension with the size of the system. In this section, we show that there is a tighter connection between RBM and the previously introduced tensor networks in arbitrary dimensions.

A. Jastrow wave functions, RBM, and the Majumdar-Gosh model

Before turning to tensor networks, let us first consider the simple case of the Jastrow wave function [Eq. (4)]. Boltzmann machines that include only interactions between the visible units lead to a wave function

$$\psi_w(\mathbf{s}) = \prod_k e^{a_k s_k} \prod_{i<j} e^{c_{ij} s_i s_j}, \tag{13}$$

which has the form of a product between functions of pairs of spins and is thus a Jastrow wave function. More generally, semirestricted Boltzmann machines, including interactions between visible units as well as between hidden and visible units, are a product of a RBM and a Jastrow factor.

Nevertheless, one may ask whether a RBM alone is enough to describe a Jastrow factor. We first rewrite the RBM as

$$\psi_w(\mathbf{s}) = \prod_j A_j^{s_j} \prod_i \left(B_i \prod_j W_{ij}^{s_j} + \frac{1}{B_i \prod_j W_{ij}^{s_j}} \right), \tag{14}$$

where we have redefined the parameters with uppercase letters as the exponential of the original parameters, thus removing the exponentials in the hyperbolic cosine. This form will be convenient for the numerical simulations presented later. Since Jastrow wave functions are a product of functions of all pairs of spins, let us show that a RBM with one hidden unit can represent any function of two spins. It then follows that a RBM with $M = N(N-1)/2$ hidden units, each representing a function of one pair of spins, can represent a Jastrow wave function with polynomial resources. We thus have to solve for a system of four nonlinear equations with $s_1,\ s_2 \in \{-1, 1\}$ and f the most general function of two spins: $\psi_w(s_1, s_2) = f(s_1, s_2)$. This system is solved in Appendix A, which provides an analytical solution for the parameters of the RBM to represent the Jastrow wave function exactly, or to arbitrary precision if $f(s_1, s_2) = 0$ for some spins.

As an application, we use this result to write the ground state of the Majumdar-Gosh model [73] exactly as a RBM. The Majumdar-Ghosh model is defined by the following spin-1/2 Hamiltonian:

$$H = J \sum_{i=1}^{N-1} \mathbf{S}_i \cdot \mathbf{S}_{i+1} + \frac{J}{2} \sum_{i=1}^{N-2} \mathbf{S}_i \cdot \mathbf{S}_{i+2}. \tag{15}$$

The ground-state wave function is a product of singlets formed by neighboring pairs of spins:

$$|\psi\rangle \propto \prod_{n=1}^{N/2} |\uparrow_{2n-1}\rangle|\downarrow_{2n}\rangle - |\downarrow_{2n-1}\rangle|\uparrow_{2n}\rangle. \tag{16}$$

This wave function can also be expanded in the computational basis as

$$\psi(s_1, \ldots, s_N) \propto \prod_{n=1}^{N/2} (-1)^{(s_{2n-1}+3)/2} \delta_{s_{2n-1} \neq s_{2n}}, \tag{17}$$

$$\propto \prod_{n=1}^{N/2} f(s_{2n-1}, s_{2n}). \tag{18}$$

Using the previous result, each function of two spins f can be written as a RBM using one hidden unit, which leads to a RBM representation of the ground states with $M = N/2$ hidden units. We also find numerically on small systems that a RBM using less than $M = N/2$ has higher energy than the ground state, which suggests that $M = N/2$ could be optimal.

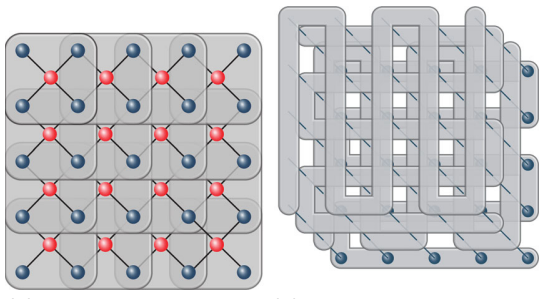
B. Short-range RBM are EPS

Let us now turn to the specific case of RBM with short-range connections (sRBM). This encompasses all quantum states that have previously been written exactly as a RBM, such as the toric code or the 1D symmetry-protected topological cluster state [36]. Such states have weight connections between visible hidden units that are local. Each hidden unit is connected to a local region with, at most, d neighboring spins. If we divide the lattice into M subsets $p_i,\ i \in \{1, \ldots, M\}$, the wave function can be rewritten as (we omit here the biases a_j, which are local one-body terms)

$$\psi_w(\mathbf{s}) = \prod_{i=1}^{M} \cosh\left(b_i + \sum_{j \in p_i} w_{ij} s_j \right) \tag{19}$$

$$= \prod_{i=1}^{M} C_i^{\mathbf{s}_i}, \tag{20}$$

where $\mathbf{s}_i$ is the spin configuration in the subset p_i. This is the form [Eq. (7)] of an EPS [Fig. 3(a)]. For translational-invariant systems, the short-range RBM becomes a convolutional RBM, which corresponds to a translational-invariant EPS. The main difference between a short-range RBM and an EPS is that the RBM considers a very specific function among all possible functions of the spins inside a plaquette; hence, EPS are more general than short-range RBM. This also directly implies that the



(a) Local RBM as an EPS (b) RBM as a nonlocal SBS

FIG. 3. (a) A locally connected RBM is an EPS where each plaquette encodes the local connections to a hidden unit. (b) Once expressed as a SBS, a fully connected RBM can be represented by many strings on top of each other. Enlarging the RBM by using noncommuting matrices to nonlocal SBS induces a geometry in each string.

entanglement of short-range RBM follows an area law. The main advantage of short-range RBM over EPS is that, because of the exponential scaling of EPS with the size of the plaquettes, larger plaquettes can be used in short-range RBM than in EPS. Since, in practice, for finite systems it is possible to work directly with fully connected RBM, we argue that EPS or fully connected RBM should be preferred to short-range RBM for numerical purposes.

C. Fully connected RBM are SBS

Fully connected RBM, on the other hand, do not always satisfy an area law [72] and hence cannot always be approximated by local tensor networks. Nevertheless, one can express the RBM wave function as (here, we also omit the bias a_j)

$$\psi_w(\mathbf{s}) = \prod_i \cosh\left(b_i + \sum_j w_{ij}s_j\right) \tag{21}$$

$$\propto \prod_i (e^{b_i+\sum_j w_{ij}s_j} + e^{-b_i-\sum_j w_{ij}s_j}) \tag{22}$$

$$\propto \prod_i \mathrm{Tr}\begin{pmatrix} e^{b_i+\sum_j w_{ij}s_j} & 0 \\ 0 & e^{-b_i-\sum_j w_{ij}s_j} \end{pmatrix} \tag{23}$$

$$\propto \prod_i \mathrm{Tr}\left(\prod_{j\in i} A_{i,j}^{s_j}\right), \tag{24}$$

where

$$A_{i,j}^{s_j} = \begin{pmatrix} e^{b_i/N+w_{ij}s_j} & 0 \\ 0 & e^{-b_i/N-w_{ij}s_j} \end{pmatrix} \tag{25}$$

are diagonal matrices of bond dimension 2. This shows that RBM are string-bond states, as the wave function can be written as a product of MPS over strings, where each hidden unit corresponds to one string. The only difference between the SBS as depicted in Fig. 1(d) and the RBM is the geometry of the strings. In a fully connected RBM, each string goes over the full lattice, while SBS have traditionally been used with smaller strings and with, at most, a few strings overlapping at each lattice site.

D. Generalizing RBM to nonlocal SBS

In the SBS language, RBM consists of many strings overlapping on the full lattice. The matrices in each string in the RBM are diagonal and hence commute, so they can be moved in the string up to a reordering of the spins. This means that each string does not have a fixed geometry and can adapt to stronger correlations in different parts of the lattice, even over long distances. This motivates us to generalize RBM to SBS with diagonal matrices in which each string covers the full lattice [Fig. 3(b)]. In the following, we denote these states as nonlocal dSBS. This amounts to relaxing the constraints on the RBM parameters to the most general diagonal matrix and enlarging the bond dimension of the matrices. For example, taking the matrices

$$A_{i,j}^{s_j} = \begin{pmatrix} a_{i,j}^{s_j} & 0 & 0 \\ 0 & b_{i,j}^{s_j} & 0 \\ 0 & 0 & c_{i,j}^{s_j} \end{pmatrix}, \tag{26}$$

with different parameters $a_{i,j}^{s_j}$ for each string, lattice site, and spin direction, leads to the wave function (here, $D = 3$)

$$\psi_w(\mathbf{s}) = \prod_i \left(\prod_j a_{i,j}^{s_j} + \prod_j b_{i,j}^{s_j} + \prod_j c_{i,j}^{s_j}\right). \tag{27}$$

Note that even for 2×2 matrices, the nonlocal dSBS is more general than a RBM since the coefficients in each of the two matrices corresponding to one spin are independent from each other, which is not the case in the RBM.

Generalizing such a wave function to larger spins than spin-1/2 is straightforward since the spin s_i is just indexing the parameters. This provides a way of defining a natural generalization of RBM that can handle systems with larger physical dimension. For instance, this can be applied to spin-1 systems, while a naive construction for a RBM with spin-1 visible and hidden units leads to additional constraints, as well as to approximate bosonic systems by truncating the local Hilbert space of the bosons.

A further way to extend this class of states is to include noncommuting matrices. This fixes the geometry of each string by defining an order and also enables us to represent more complicated interactions. In the following, we refer to SBS in such a geometry as nonlocal SBS. The advantage of this approach is that it can capture more complex correlations within each string while introducing additional geometric information about the problem at hand. However, it comes at a greater numerical cost than nonlocal dSBS or RBM because of the additional number of parameters. In practice, one can use an already-optimized RBM or dSBS as a way of initializing a nonlocal SBS.

In some cases, the SBS representation is more compact than the RBM/dSBS representation. Let us consider again the ground state of the Majumdar-Gosh Hamiltonian, which we previously wrote as a RBM with $M = N/2$ hidden units. The ground state of the Majumdar-Gosh Hamiltonian can also be written as a simple MPS with bond dimension 3 and periodic boundary conditions, with matrices [24]

$$A_n^{s_n=-1} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 0 & 0 \end{pmatrix}, \qquad A_n^{s_n=1} = \begin{pmatrix} 0 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \tag{28}$$

or for open boundary conditions with

$$A_{2n}^{s=-1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \qquad A_{2n}^{s=1} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \tag{29}$$

$$A_{2n-1}^{s=-1} = \begin{pmatrix} 1 & 0 \end{pmatrix}, \qquad A_{2n-1}^{s=1} = \begin{pmatrix} 0 & 1 \end{pmatrix}. \tag{30}$$

Since this state is a MPS, it is also a SBS with one string. The RBM representation of the same state requires $N/2$ strings. In practice, the number of nonzero coefficients is comparable since, in both cases, the representation is sparse; however, for numerical purposes, a fully connected RBM needs of the order $O(N^2)$ parameters before finding the exact ground state, while a MPS or SBS with one string will need $O(N)$ parameters for both open and periodic boundary conditions.

Another example is the AKLT model [74] defined by the following spin-1 Hamiltonian in periodic boundary conditions:

$$H = \sum_{i=1}^{N} \left[\frac{1}{2} \mathbf{S}_i \cdot \mathbf{S}_{i+1} + \frac{1}{6} (\mathbf{S}_i \cdot \mathbf{S}_{i+1})^2 + \frac{1}{3} \right]. \tag{31}$$

Its ground state has a simple form as a MPS of bond dimension 2. It can also be written as an exact RBM by mapping the system to a spin-$1/2$ chain, but the number of hidden units needed for an exact representation scales as $O(N^2)$ in the system size [75]. We have numerically optimized the spin-1 extension of a RBM with the form Eq. (27) (see Appendix B for the details of the numerical optimization) and found that, already for small sizes of the chain, a much higher number of parameters is required to approach the ground-state energy as compared to a SBS with noncommuting matrices, which is exact with one string of bond dimension 2 (Fig. 4). We also show in Sec. IV that, in some other cases, the RBM needs less parameters than a SBS to obtain a similar energy. This demonstrates that both RBM and SBS have advantages and that their efficiency depends on the particular model that is investigated. It remains an open question whether there exist MPS or SBS that can provably not be efficiently approximated by a RBM (for which the RBM would need exponentially many parameters).

To be able to use both the advantages of RBM (efficient to compute, few parameters) and of SBS (complex representation, geometric interpretation), one can use the flexibility of SBS by including some strings that have a full MPS over the whole lattice, some strings that include only local connections and that will ensure that the locality of the system is preserved, and some strings that have the form of a RBM and that can easily capture large entanglement and long-range correlations. In many cases of interest, an initial approximation of the ground state can be obtained, either by optimizing simpler wave functions or by first applying DMRG to optimize a MPS. This initial approximation can then be used in conjunction with the previous Ansatz classes by multiplying an Ansatz wave function by the initial approximation. For the resulting wave function,

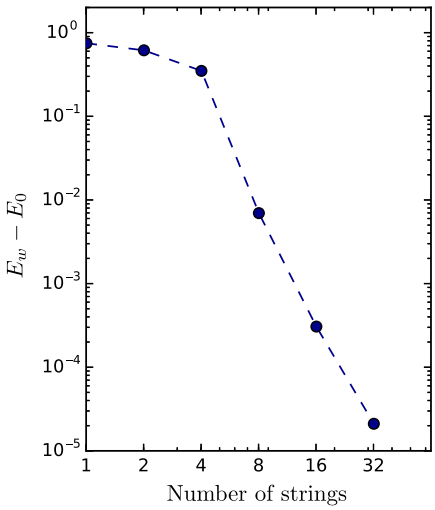


FIG. 4. Energy difference with the exact ground-state energy of a spin-1 extension of a RBM [Eq. (27)] with $D = 2$ and different number of strings for the AKLT model on a spin-1 chain with eight spins. A nonlocal SBS with noncommuting matrices and one string is exact within numerical accuracy.

$$\psi_w(\mathbf{s}) = \psi_w^{\text{init}}(\mathbf{s}) \psi_w^{\text{SBS}}(\mathbf{s}), \tag{32}$$

the ratio of the wave function on two configurations, as well as the log-derivatives, depends only on the respective ratio and log-derivatives of each separate wave function, making the application of the VMC method straightforward. This procedure has the advantage of reducing the number of parameters necessary for obtaining a good approximation to the ground state and making the optimization procedure more stable since the initial state is not a completely random state. Such a procedure provides a generic way to enhance the power of more specific Ansatz wave functions tailored to particular problems, as we demonstrate in the next section. A similar technique has been used to construct tensor-product projected states with tensor networks in Ref. [76], and more generally, it can be used to project the wave function of an initial reference state in a Fock space and is thus also suitable to describe fermionic systems.

IV. APPLICATION TO CHIRAL TOPOLOGICAL STATES

In this section, we turn to a practical application on a challenging problem for traditional tensor-network methods, namely, the approximation of a state with chiral topological order. While chiral topological PEPS have been constructed, the resulting states are critical. Moreover, the local parent Hamiltonian of a chiral fermionic Gaussian PEPS has to be gapless [55]. In the

following, we investigate if this obstruction carries on to the tensor-network and neural-network states that we have introduced previously.

A. RBM can describe a Laughlin state exactly

Let us consider a lattice version of the Laughlin wave function at filling factor $1/2$ defined for a spin-$1/2$ system as

$$\psi_{\text{Laughlin}}(\mathbf{s}) = \delta_{\mathbf{s}} \prod_k \chi_k^{s_k} \prod_{i<j} (z_i - z_j)^{\frac{1}{2} s_i s_j}, \tag{33}$$

where $\delta_{\mathbf{s}}$ fixes the total spin to 0, the z_i are the complex coordinates of the positions of the lattice sites, and the phase factor is defined as $\chi_k^{s_k} = e^{i\pi(k-1)(s_k+1)/2}$, ensuring that the state is a singlet. This wave function is equivalent to the Kalmeyer-Laughlin wave function in the thermodynamic limit and has been shown to describe a lattice state sharing the topological properties of the continuum Laughlin states on several lattices [77–79]. In addition, it can be written as a correlator from conformal fields, which has enabled the exact derivation of parent Hamiltonians for this state on any finite lattice [80].

The Laughlin wave function has the structure of a Jastrow wave function, and we have shown in Sec. III A that any Jastrow wave function can be written as a RBM with $M = N(N-1)/2$ hidden units. It follows that RBM and nonlocal SBS can represent a gapped chiral topological state exactly. This is in sharp contrast to local tensor-network states for which there is no exact description known for a (noncritical) chiral topological state. This difference is due to the nonlocal connections in the RBM and Jastrow wave function, which allow them to easily describe a Laughlin state. We note that a chiral p-wave superconductor is another example of a gapped chiral topological state that has been recently written as a (fermionic) quasilocal Boltzmann machine [20].

However, the previous construction is not satisfactory in the sense that the RBM requires a number of hidden units scaling as $O(N^2)$, which is too high for numerical purposes on lattices that are not extremely small. We thus turn to the approximate representation of the Laughlin wave function using a RBM.

B. Numerical approximation of a Laughlin state

The lattice Laughlin wave function we consider has an exact parent Hamiltonian on a finite lattice [80] defined as

$$H_{\text{parent}} = \frac{2}{3} \sum_{i \neq j} |w_{ij}|^2 \mathbf{S}_i \cdot \mathbf{S}_j + \frac{2}{3} \sum_{i \neq j \neq k} \bar{w}_{ij} w_{ik} \mathbf{S}_j \cdot \mathbf{S}_k - \frac{2i}{3} \sum_{i \neq j \neq k} \bar{w}_{ij} w_{ik} \mathbf{S}_i \cdot (\mathbf{S}_j \times \mathbf{S}_k), \tag{34}$$

where $w_{ij} = [(z_i + z_j)/z_i - z_j]$ and $\mathbf{S}_j = (S_j^x, S_j^y, S_j^z)$ is the spin operator at site j. We specialize to the square lattice with open boundary conditions and minimize the energy of different wave functions with respect to this Hamiltonian by applying the VMC method presented in Sec. II B with a stochastic reconfiguration optimization, which is equivalent to the natural gradient descent [63,81,82] (details of the numerical optimization can be found in Appendix B). Results are presented in Table I.

TABLE I. Energy per site difference with the ground-state energy and overlap with the Laughlin state of different Ansatz wave functions optimized with respect to the Hamiltonian H_{parent} on a 6×6 square lattice with open boundary conditions. Note that sRBM have M' hidden units connected to all spins in each plaquette of size 3×3, while RBM have M hidden units connected to all spins of the lattice.

Ansatz	$(E_w - E_0)/N$	$\lvert\langle\psi_w \lvert \psi_{\text{Laughlin}}\rangle\rvert$
EPS 2×2	4.3×10^{-2}	46.10%
EPS 3×3	2.2×10^{-2}	75.79%
sRBM $M' = 1$	8.3×10^{-2}	0.01%
sRBM $M' = 2$	3.1×10^{-2}	46.32%
sRBM $M' = 4$	2.5×10^{-2}	59.07%
RBM $M = N$	5.8×10^{-4}	99.7%
RBM $M = 2N$	1.1×10^{-5}	99.99%

We find that EPS with plaquettes of size up to 3×3 have an energy difference with the Laughlin state of the order 10^{-2}, which is better than a short-range RBM (denoted sRBM) on 3×3 plaquettes and up to $M' = 4$ hidden units per plaquette, while the energy of a fully connected RBM with $M = 2N$ hidden units is within 10^{-5} of the energy of the ground state. The resulting RBM uses much less hidden units than would be required for it to be exact, yet it reaches an overlap of 99.99% with the Laughlin wave function. This result shows that the fully connected structure of the RBM is an advantage to describe this state and that EPS can be used instead of short-range RBM. Moreover, we have found that EPS are easier to optimize numerically than a short-range RBM: They are more stable since each coefficient is considered separately, no exponentials or products that lead to unstable behavior are present, and the derivatives have a very simple form [Eq. (8)].

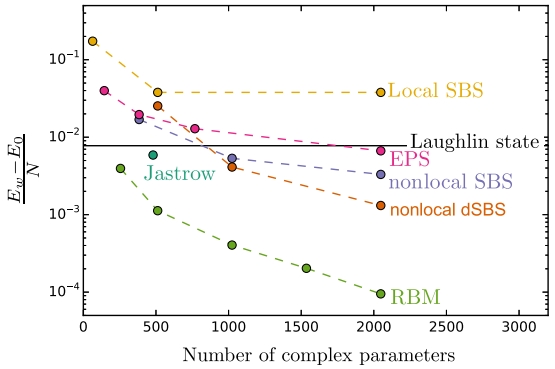
C. Numerical approximation of a chiral spin liquid

The previous results indicate that RBM might be useful for approximating chiral topological states numerically, but they are limited to relatively small sizes because of the nonlocal nature of the parent Hamiltonian, which includes interactions between all triplets of spins on the lattice. In Ref. [45], a local Hamiltonian stabilizing a state in the same class as the Laughlin state was obtained by restricting H_{parent} to local terms and setting the long-range interactions to zero. This leads to the Hamiltonian

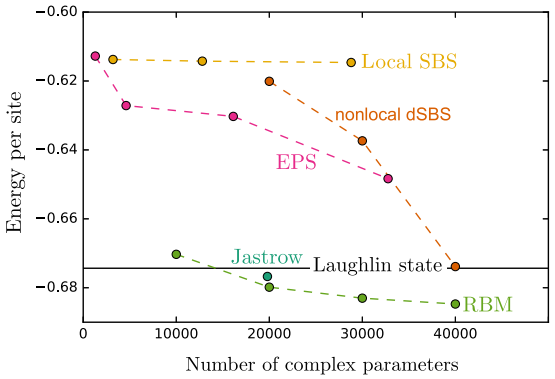
$$H_l = J \sum_{\langle i,j \rangle} \mathbf{S}_i \cdot \mathbf{S}_j + J_\chi \sum_{\langle i,j,k \rangle_\circlearrowleft} \mathbf{S}_i \cdot (\mathbf{S}_j \times \mathbf{S}_k), \tag{35}$$

where $\langle i,j\rangle$ indicates indices of nearest neighbors on the lattice and $\langle i,j,k\rangle_{\circlearrowleft}$ indicates indices of all triangles of neighboring spins, with vertices labeled in the counterclockwise direction. We focus on the case $J=1, J_\chi=1$ for which the ground state of H_I has above 98% overlap with the Laughlin wave function [Eq. (33)] on a 4×4 lattice. We minimize the energy of different classes of states on 4×4 and 10×10 square lattices with open boundary conditions. For optimizing wave functions with tens of thousands of parameters, we use a batch version of stochastic reconfiguration, which optimizes a random subset of the parameters at each iteration (see Appendix B). We consider several Ansatz wave functions, including EPS with plaquettes of size 2×2, 3×2, 4×2, and 3×3; local SBS covering the lattice with horizontal, vertical, and diagonal strings and increasing bond dimension; RBM with an increasing number of hidden units; nonlocal SBS with diagonal matrices (denoted dSBS) or with noncommuting matrices of bond dimension 2, and different numbers of strings covering the full lattice. We observe that, while the optimization of EPS and SBS is particularly stable, the optimization of RBM can lead to numerical instabilities that are resolved by writing the RBM in the form presented in Eq. (14). Since we use the same optimization procedure for all Ansatz wave functions and since the required time (and memory) to perform the optimization is mainly a function of the number of parameters and of the accuracy, we can compare the Ansatz classes by comparing how many parameters are needed to obtain a similar energy.

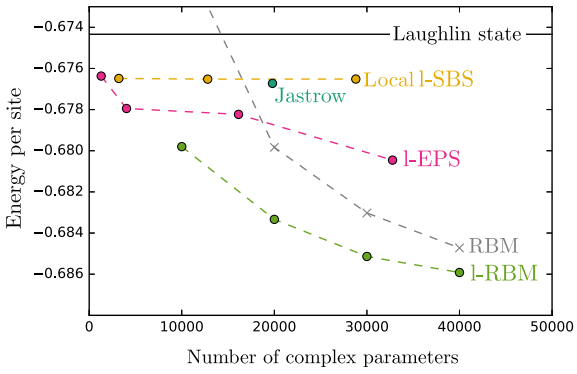
We first focus [Fig. 5(a)] on a 4×4 lattice for which the exact ground state can be obtained using exact diagonalization. Local SBS have an energy higher than the Laughlin state, and the energy is saturated with increasing bond dimension, which means that the pattern of horizontal, vertical, and diagonal strings is not enough to capture all correlations in the ground state. While a large 4×4 plaquette would make EPS exact on this small lattice, this would require 2^{16} parameters. The energy of the Laughlin state is already reached for 3×3 plaquettes. RBM with a number of hidden units larger than N and nonlocal SBS with a corresponding number of strings have lower energy than the Laughlin state or the Jastrow wave function. When the number of strings grows, the energy decreases even further. On a larger 10×10 lattice [Fig. 5(b)], the exact ground-state energy is unknown, but we can compare the energy of the different Ansatz wave functions and observe similar results. Only the Jastrow wave function, nonlocal SBS, and RBM have an energy comparable to the Laughlin state. Notice that nonlocal SBS have a constant factor more parameters than a RBM for the same number of strings. On the one side, this allows SBS to achieve better energy than RBM with the same number of strings. On the other side, this comes with the drawback that we can only optimize fewer strings, and on the large lattice, we are numerically limited to nonlocal dSBS with up to N strings. We can conclude that RBM are particularly efficient in this



(a) 4x4 lattice



(b) 10x10 lattice



(c) 10x10 lattice

FIG. 5. Energy of H_I per site for different optimized Ansatz wave functions on a square lattice. The number of parameters (N_p) is modified by increasing the bond dimension D (local SBS, $N_p\propto D^2$), the size of the plaquettes (EPS, $N_p\propto M_P 2^P$, where M_P is the number of plaquettes and P is the number of spins in one plaquette), the number of strings M_S (nonlocal SBS and dSBS, $N_p\propto M_S$), or the number of hidden units M_h (RBM, $N_p\propto M_h$). (a) The 4×4 lattice for which the energy difference with the exact ground-state energy is plotted. (b) The 10×10 lattice for which the exact ground-state energy is unknown and the reference energy of the Laughlin state is indicated as a black line. (c) Optimization of wave functions that have been multiplied by the Laughlin wave function on a 10×10 lattice. The original RBM results are indicated for reference as grey crosses.

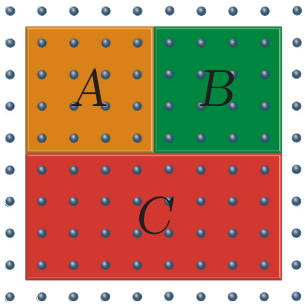


FIG. 6. Partition of the lattice used to compute the topological entanglement entropy.

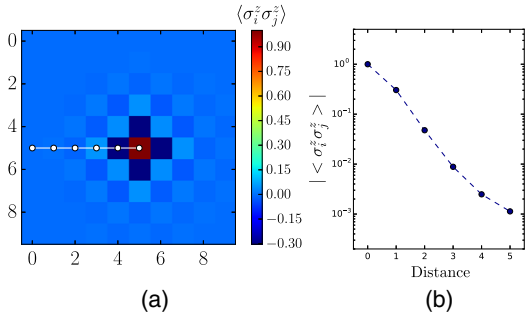


FIG. 7. (a) The spin-spin correlation function between one lattice site (in red) and all other spins on the lattice measured on the optimized l-RBM with lowest energy reveals the antiferromagnetic behavior of the correlations. (b) Decay of the correlations with the distance across the direction indicated in (a) as a white solid line. The error bars are within dot size, and finite-size effects can already be seen for the last point.

example since they require significantly less parameters than SBS for attaining the same energy. This has to be contrasted with the previous examples of the Majumdar-Gosh and AKLT models where the opposite was true. Therefore, each class of states has advantages and drawbacks depending on the model we are looking at. We note, in addition, that a nonlocal SBS can be initialized with the results of a previous optimization with a RBM, which could provide a way of minimizing the difficulties of optimizing a large number of parameters.

As we have previously noticed, we can also use an initial approximation of the ground state in combination with the previous Ansatz classes. In the case of the Hamiltonian H_l, the analytical Laughlin wave function can be used as our initial approximation in Eq. (32). We denote l-EPS (resp. l-SBS, l-RBM) a wave function that consists of a product of the Laughlin wave function and an EPS (resp. SBS, RBM) and minimize the energy of the resulting states. This allows us to obtain lower energies for each Ansatz class [Fig. 5(c)]. Once the wave functions are optimized, their properties can be computed using Monte Carlo sampling. To check that the ground state is indeed in the same class as the Laughlin state, we compute the topological entropy of some of the optimized states by dividing the lattice into four regions (Fig. 6) and computing the Renyi entropy $S_A^{(2)} = -\ln \mathrm{Tr}\rho_A^2$ of each subregion using the Metropolis-Hastings Monte Carlo algorithm with two independent spin chains [83,84]. The topological entanglement entropy is then defined as [85,86]

$$S_{\mathrm{topo}} = S_A^{(2)} + S_B^{(2)} + S_C^{(2)} - S_{AB}^{(2)} - S_{AC}^{(2)} - S_{BC}^{(2)} + S_{ABC}^{(2)}, \tag{36}$$

TABLE II. Topological entanglement entropy (TEE) of the analytical Laughlin state and optimized l-EPS, RBM, and l-RBM.

Ansatz	TEE
Laughlin	−0.339(3)
l-EPS 3×3	−0.36(1)
RBM $M = 4N$	−0.34(1)
l-RBM $M = 4N$	−0.34(1)

and it is expected to be equal to $-\ln 2 \approx -0.347$ for the Laughlin state [87]. The results we obtain are presented in Table II and provide additional evidence that the ground state of H_l has the same topological properties as the Laughlin state. The Hamiltonian H_l was recently investigated on an infinite lattice using infinite PEPS [88], and further evidence was provided that the ground state is chiral. The PEPS results suggest the presence of long-range algebraically decaying correlations that may be a feature of the model or a restriction of PEPS to study chiral systems. The correlations on short distances agree with the correlations that we can compute on our finite system [Fig. 7(a)], but our method does not allow us to make claims about the long-distance behavior of the correlation function. We also observe that fully connected RBM cannot be defined directly in the thermodynamic limit without a truncation of the distance of the interaction between visible and hidden units, thus transforming the RBM into a short-range RBM (albeit of larger range than an EPS). In Ref. [72], it was observed that the entanglement entropy of some specific short-range RBM can be computed analytically from the weights of the RBM. The method we use here works in the general case and also for a fully connected RBM, but it requires Monte Carlo sampling of the wave function. The optimized RBM weights encode all of the information about the wave function; thus, it would be interesting to understand more precisely which quantities can be extracted directly from them. Whether direct information about the phase of the system can be obtained in this way without requiring Monte Carlo sampling remains an interesting open problem for future work.

V. CONCLUSION

We have shown that there is a strong connection between neural-network quantum states in the form of Boltzmann machines and some tensor-network states that can be optimized using the variational Monte Carlo method:

While short-range restricted Boltzmann machines are a subclass of entangled plaquette states, fully connected restricted Boltzmann machines are a subclass of string-bond states. However, these string-bond states are different from traditional string-bond states because of their nonlocal structure, which connects every spin on the lattice to every string. This enabled us to generalize restricted Boltzmann machines by introducing nonlocal (diagonal or noncommuting) string-bond states, which can be defined for larger local Hilbert space and with additional geometric flexibility. We compared the power of these different classes of states and showed that, while there are cases where string-bond states require less parameters than fully connected restricted Boltzmann machines to describe the ground state of a many-body Hamiltonian, there are also cases where the additional parameters in each string make string-bond states less efficient to optimize numerically. We applied these methods to the challenging problem of describing states with chiral topological order, which is hard for traditional tensor networks. We showed that every Jastrow wave function, and thus a Laughlin wave function, can be written as an exact restricted Boltzmann machine. In addition, we gave numerical evidence that a restricted Boltzmann machine with a much smaller number of hidden units can still give a good approximation to the Laughlin state. Finally, we turned to the approximation of the ground state of a chiral spin liquid and showed that restricted Boltzmann machines achieve a lower energy than the Laughlin state and the same topological entanglement entropy. We argued that combining different classes of states allows us to take advantage of the initial knowledge of the model and of the particularities of each class. This was demonstrated by combining a Jastrow wave function to tensor networks and restricted Boltzmann machines, which allowed us to get lower energies than the initial states and to characterize the ground state.

Our work sheds some light on the representative power of restricted Boltzmann machines and establishes a bridge between their optimization and the optimization of tensor-network states. On the one hand, the methods developed in this work can be used to target the ground state of other Hamiltonians, and it would be interesting to know whether similar results can be achieved, for example, for non-Abelian chiral spin liquids [89,90] or generalized to fermionic systems of electrons in the continuum displaying the fractional quantum Hall effect. On the other hand, we also showed that some tools used in machine learning can be rephrased in tensor-network language, thus providing additional physical insights about the systems they describe. Matrix product states have already been used as a tool for supervised learning [91,92], and our work opens up the possibility of using not only restricted Boltzmann machines but also string-bond states to represent a probability distribution over some data while encoding additional information about its geometric structure.

ACKNOWLEDGMENTS

We would like to thank Martin Ganahl, Xun Gao, and Giuseppe Carleo for discussions. This work was supported by the ERC grant QUENOCOBA, ERC-2016-ADG (Grant No. 742102). N. P. and M. A. acknowledge financial support from the Exploring Quantum Matter program.

Note added.—Recently, related independent work came to our attention. Nomura *et al.* [93] combine RBM with pair-product wave functions and apply them to the Heisenberg and Hubbard models. Clark [94] constructs a mapping between RBM and EPS/correlator product states. Kaubruegger *et al.* [95] give further analytical and numerical evidence supporting the application of RBM to chiral topological states such as the Laughlin state.

APPENDIX A: JASTROW WAVE FUNCTIONS ARE RESTRICTED BOLTZMANN MACHINES

Let us show that a RBM with one hidden unit can represent any function f of two spins. It then follows that a RBM with $M = N(N-1)/2$ hidden units, each representing a function of one pair of spins, can represent a Jastrow wave function. We parametrize f by its four values on two spins $s_1,\ s_2 \in \{-1, 1\}$ and solve for a system of four nonlinear equations:

$$F_{11} = A_1 A_2 \left(W_1 W_2 + \frac{1}{W_1 W_2} \right), \tag{A1}$$

$$F_{-1-1} = \frac{1}{A_1 A_2} \left(W_1 W_2 + \frac{1}{W_1 W_2} \right), \tag{A2}$$

$$F_{1-1} = \frac{A_1}{A_2} \left(\frac{W_1}{W_2} + \frac{W_2}{W_1} \right), \tag{A3}$$

$$F_{-11} = \frac{A_2}{A_1} \left(\frac{W_2}{W_1} + \frac{W_1}{W_2} \right), \tag{A4}$$

where we have set $B_1 = B_2 = 1$. The RBM is well defined when all parameters are nonzero, and we change variables by defining $X = W_1 W_2$, $Y = (W_1/W_2)$, $A = A_1 A_2$, $B = (A_1/A_2)$, obtaining a new set of equations:

$$F_{-1-1} A^2 = F_{11}, \tag{A5}$$

$$F_{-11} B^2 = F_{1-1}, \tag{A6}$$

$$X^2 - \frac{1}{A} X + 1 = 0, \tag{A7}$$

$$Y^2 - \frac{1}{B} Y + 1 = 0. \tag{A8}$$

We first suppose that the values $F_{s_i s_j}$ are nonzero. These quadratic equations all have nonzero analytical solutions

in the complex plane, which we denote A_0, B_0, X_0, and Y_0. The original parameters are then the solutions of

$$W_1^2 = X_0 Y_0, \tag{A9}$$

$$W_2^2 = X_0 / Y_0, \tag{A10}$$

$$A_1^2 = A_0 B_0, \tag{A11}$$

$$A_2^2 = A_0 / B_0, \tag{A12}$$

which is again a set of quadratic equations with nonzero analytical solutions. If $F_{11} = F_{-1-1} = 0$ (resp. $F_{1-1} = F_{-11} = 0$), the exact solution is given directly by $A_0 = 1$, $X_0 = i$ (resp. $B_0 = 1$, $Y = i$). In the remaining cases where some $F_{s_i s_j}$ are zeros, the equations do not always have an exact solution, but the function can still be approximated to arbitrary precision. This case corresponds to strong restrictions on the part of the Hilbert space, which is used to write the wave function, and these constraints can also be imposed on the states directly by adding a delta function to the wave function, which is equal to 1 only when the constraints on the spins are satisfied. Having a Markov chain Monte Carlo sampling that does not visit these states then allows for a more efficient sampling.

APPENDIX B: OPTIMIZATION PROCEDURE

The goal is to minimize the energy $E_{\mathbf{w}}$ depending on some vector of parameters $\mathbf{w}$. We define $\mathbf{f}$ to be the energy gradient vector at $\mathbf{w}$. Expanding the energy to first order around $\mathbf{w}$ leads to the steepest gradient descent, where the variational parameters are updated at each iteration according to $\mathbf{w}' = \mathbf{w} + \boldsymbol{\gamma}$, with a change of parameters given by $\boldsymbol{\gamma} = -\alpha \mathbf{f}$. Here, α is a small step size. Expanding the energy to second order instead would result in the Newton method with a change of parameters given by

$$\boldsymbol{\gamma} = -\alpha \mathbf{H}^{-1} \mathbf{f}, \tag{B1}$$

where $\mathbf{H}$ is the Hessian of the energy. Small changes of the variational parameters may, however, lead to big changes in the wave function, especially in the case of compact nonlocal representations like RBM in which each parameter affects each part of the wave function. Taking into account the metric of changes of the wave function leads to the stochastic reconfiguration [63] method, which is equivalent to the natural gradient descent [82] and replaces the Hessian in Eq. (B1) by the covariance matrix of the derivatives of the wave function, avoiding the computation of the second-order derivatives of the energy.

The stochastic reconfiguration method can also be viewed as an approximate imaginary-time evolution in the tangent space of the wave function. Consider the normalized wave function $|\bar{\psi}_0\rangle$ and its derivatives

$$|\bar{\psi}_0\rangle = \frac{|\psi_0\rangle}{\sqrt{\langle \psi_0 | \psi_0 \rangle}}, \tag{B2}$$

$$|\bar{\psi}_i\rangle = \frac{|\psi_i\rangle}{\sqrt{\langle \psi_0 | \psi_0 \rangle}} - \frac{\langle \psi_0 | \psi_i \rangle}{\langle \psi_0 | \psi_0 \rangle} \frac{|\psi_0\rangle}{\sqrt{\langle \psi_0 | \psi_0 \rangle}}, \tag{B3}$$

defining a nonorthogonal basis set Ω. Expanding the wave function to linear order around some parameters $\mathbf{w}$ leads to

$$|\bar{\psi}(\mathbf{w} + \boldsymbol{\gamma})\rangle = \sum_{i=0}^{N_w} \gamma_i |\bar{\psi}_i\rangle. \tag{B4}$$

To minimize the energy, one can apply the imaginary-time evolution operator $e^{-\alpha H}$, which, expanded to first order for small α, is $1 - \alpha H$. The change of coefficients $\boldsymbol{\gamma}$ is found by applying this operator to $|\bar{\psi}(\mathbf{w} + \boldsymbol{\gamma})\rangle$ and projecting in the set Ω, which leads to the equation

$$-\alpha \langle \bar{\psi}_i | H | \bar{\psi}_0 \rangle = \sum_{j=1}^{M} \langle \bar{\psi}_i | \bar{\psi}_j \rangle \gamma_j, \tag{B5}$$

which can be rewritten as

$$\boldsymbol{\gamma} = -\alpha \mathbf{S}^{-1} \mathbf{f}, \tag{B6}$$

where $S_{ij} = \langle \bar{\psi}_i | \bar{\psi}_j \rangle$ and $f_i = \langle \bar{\psi}_i | H | \bar{\psi}_0 \rangle$. If we expand these expressions as expectation values over the probability distribution $p(\mathbf{s}) = \{[|\psi_w(\mathbf{s})|^2] / \sum_{\mathbf{s}} |\psi_w(\mathbf{s})|^2\}$, we obtain

$$f_i = \langle \Delta_i^* E_{\mathrm{loc}} \rangle - \langle \Delta_i^* \rangle \langle E_{\mathrm{loc}} \rangle, \tag{B7}$$

$$S_{ij} = \langle \Delta_i^* \Delta_j \rangle - \langle \Delta_i^* \rangle \langle \Delta_j \rangle, \tag{B8}$$

where the local energy is defined as $E_{\mathrm{loc}}(\mathbf{s}) = \sum_{\mathbf{s}'} \langle \mathbf{s} | H | \mathbf{s}' \rangle \{[\psi_w(\mathbf{s}')] / \psi_w(\mathbf{s})\}$ and the log-derivative of the wave function as $\Delta_w(\mathbf{s}) = \{1/[\psi_w(\mathbf{s})]\}\{[\partial \psi_w(\mathbf{s})] / \partial w\}$. Finally, the complete algorithm is as follows:

(1) Using a Metropolis-Hastings algorithm, generate samples of the probability p and compute stochastic estimates for the expectation values $\langle \Delta_j \rangle$, $\langle E_{\mathrm{loc}} \rangle$, $\langle \Delta_i^* E_{\mathrm{loc}} \rangle$, $\langle \Delta_i^* \Delta_j \rangle$.
(2) Construct the vector $\mathbf{f}$ and matrix $\mathbf{S}$.
(3) Update the parameters according to $\mathbf{w} \leftarrow \mathbf{w} - \alpha \mathbf{S}^{-1} \mathbf{f}$.
(4) Repeat the full procedure until convergence of the energy.

In practice, we repeat the full procedure 1000 to 20 000 times until the energy is converged. To optimize a large number of parameters, we randomly select a subset of the parameters of size up to 10 000 at each iteration of the algorithm and update only these parameters. This reduces the computational cost associated with the operations dealing with $\mathbf{f}$ and $\mathbf{S}$. Moreover, we can avoid forming the full matrix $\mathbf{S}$ by instead solving Eq. (B6) with a

conjugate-gradient solver [81]. Numerical stability can be achieved by adding a small constant ϵ to the diagonal elements of the matrix $\mathbf{S}$, rotating the direction of change towards the steeped descent direction. We find that a step size α of the order $1/\sqrt{i}$, where i is the iteration step, works well in conjunction with a large stabilization at the beginning, while a fixed step size can also be chosen in conjunction with a small stabilization of the order $10^{-4} - 10^{-8}$ by performing several optimizations. At the later stages of the optimization, the step size is lowered to ensure that the energy is converged. Further improvements are achieved by projecting the wave functions in a subset of fixed total spin when it is conserved by the Hamiltonian we consider [96]. The spin-flip symmetry can be enforced in a RBM by choosing the bias $b_i = 0$.

[1] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, *Quantum Machine Learning*, Nature (London) **549**, 195 (2017).

[2] E. Zahedinejad, J. Ghosh, and B. C. Sanders, *Designing High-Fidelity Single-Shot Three-Qubit Gates: A Machine-Learning Approach*, Phys. Rev. Applied **6**, 054005 (2016).

[3] M. August and X. Ni, *Using Recurrent Neural Networks to Optimize Dynamical Decoupling for Quantum Memory*, Phys. Rev. A **95**, 012335 (2017).

[4] L. Banchi, N. Pancotti, and S. Bose, *Quantum Gate Learning in Qubit Networks: Toffoli Gate without Time-Dependent Control*, npj Quantum Inf. **2**, 16019 (2016).

[5] G. Torlai and R. G. Melko, *Neural Decoder for Topological Codes*, Phys. Rev. Lett. **119**, 030501 (2017).

[6] G. Torlai, G. Mazzola, J. Carrasquilla, M. Troyer, R. Melko, and G. Carleo, *Many-Body Quantum State Tomography with Neural Networks*, arXiv:1703.05334.

[7] E. P. L. van Nieuwenburg, Y.-H. Liu, and S. D. Huber, *Learning Phase Transitions by Confusion*, Nat. Phys. **13**, 435 (2017).

[8] J. Carrasquilla and R. G. Melko, *Machine Learning Phases of Matter*, Nat. Phys. **13**, 431 (2017).

[9] P. Broecker, J. Carrasquilla, R. G. Melko, and S. Trebst, *Machine Learning Quantum Phases of Matter Beyond the Fermion Sign Problem*, Sci. Rep. **7**, 8823 (2017).

[10] L. Wang, *Discovering Phase Transitions with Unsupervised Learning*, Phys. Rev. B **94**, 195105 (2016).

[11] L.-F. Arsenault, A. Lopez-Bezanilla, O. A. von Lilienfeld, and A. J. Millis, *Machine Learning for Many-Body Physics: The Case of the Anderson Impurity Model*, Phys. Rev. B **90**, 155136 (2014).

[12] J. Liu, Y. Qi, Z. Y. Meng, and L. Fu, *Self-Learning Monte Carlo Method*, Phys. Rev. B **95**, 041101 (2017).

[13] L. Huang and L. Wang, *Accelerated Monte Carlo Simulations with Restricted Boltzmann Machines*, Phys. Rev. B **95**, 035105 (2017).

[14] G. Torlai and R. G. Melko, *Learning Thermodynamics with Boltzmann Machines*, Phys. Rev. B **94**, 165134 (2016).

[15] G. Carleo and M. Troyer, *Solving the Quantum Many-Body Problem with Artificial Neural Networks*, Science **355**, 602 (2017).

[16] Z. Cai, *Approximating Quantum Many-Body Wave-Functions Using Artificial Neural Networks*, arXiv:1704.05148.

[17] H. Saito, *Solving the Bose-Hubbard Model with Machine Learning*, J. Phys. Soc. Jpn. **86**, 093001 (2017).

[18] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, *A Learning Algorithm for Boltzmann Machines*, Cogn. Sci. **9**, 147 (1985).

[19] X. Gao and L.-M. Duan, *Efficient Representation of Quantum Many-Body States with Deep Neural Networks*, Nat. Commun. **8**, 662 (2017).

[20] Y. Huang and J. E. Moore, *Neural Network Representation of Tensor Network and Chiral States*, arXiv:1701.06246.

[21] M. B. Hastings, *An Area Law for One-Dimensional Quantum Systems*, J. Stat. Mech. (2007) P08024.

[22] F. Verstraete and J. I. Cirac, *Matrix Product States Represent Ground States Faithfully*, Phys. Rev. B **73**, 094423 (2006).

[23] F. Verstraete, V. Murg, and J. Cirac, *Matrix Product States, Projected Entangled Pair States, and Variational Renormalization Group Methods for Quantum Spin Systems*, Adv. Phys. **57**, 143 (2008).

[24] U. Schollwöck, *The Density-Matrix Renormalization Group in the Age of Matrix Product States*, Ann. Phys. (Amsterdam) **326**, 96 (2011).

[25] S. R. White, *Density-Matrix Algorithms for Quantum Renormalization Groups*, Phys. Rev. B **48**, 10345 (1993).

[26] F. Verstraete and J. I. Cirac, *Renormalization Algorithms for Quantum-Many Body Systems in Two and Higher Dimensions*, arXiv:0407066.

[27] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, *Computational Complexity of Projected Entangled Pair States*, Phys. Rev. Lett. **98**, 140506 (2007).

[28] A. Gendiar and T. Nishino, *Latent Heat Calculation of the Three-Dimensional $q = 3$, 4, and 5 Potts Models by the Tensor Product Variational Approach*, Phys. Rev. E **65**, 046702 (2002).

[29] F. Mezzacapo, N. Schuch, M. Boninsegni, and J. I. Cirac, *Ground-State Properties of Quantum Many-Body Systems: Entangled-Plaquette States and Variational Monte Carlo*, New J. Phys. **11**, 083026 (2009).

[30] H. J. Changlani, J. M. Kinder, C. J. Umrigar, and G. K.-L. Chan, *Approximating Strongly Correlated Wave Functions with Correlator Product States*, Phys. Rev. B **80**, 245116 (2009).

[31] N. Schuch, M. M. Wolf, F. Verstraete, and J. I. Cirac, *Simulation of Quantum Many-Body Systems with Strings of Operators and Monte Carlo Tensor Contractions*, Phys. Rev. Lett. **100**, 040501 (2008).

[32] A. Sfondrini, J. Cerrillo, N. Schuch, and J. I. Cirac, *Simulating Two- and Three-Dimensional Frustrated Quantum Systems with String-Bond States*, Phys. Rev. B **81**, 214426 (2010).

[33] A. W. Sandvik and G. Vidal, *Variational Quantum Monte Carlo Simulations with Tensor-Network States*, Phys. Rev. Lett. **99**, 220602 (2007).

[34] A. Kitaev, *Fault-Tolerant Quantum Computation by Anyons*, Ann. Phys. (Amsterdam) **303**, 2 (2003).

[35] F. Verstraete, M. M. Wolf, D. Perez-Garcia, and J. I. Cirac, *Criticality, the Area Law, and the Computational Power of*

Projected Entangled Pair States, Phys. Rev. Lett. **96**, 220601 (2006).
[36] D.-L. Deng, X. Li, and S. Das Sarma, *Exact Machine Learning Topological States*, Phys. Rev. B **96**, 195145 (2017).
[37] J. Chen, S. Cheng, H. Xie, L. Wang, and T. Xiang, *On the Equivalence of Restricted Boltzmann Machines and Tensor Network States*, arXiv:1701.04831.
[38] D. C. Tsui, H. L. Stormer, and A. C. Gossard, *Two-Dimensional Magnetotransport in the Extreme Quantum Limit*, Phys. Rev. Lett. **48**, 1559 (1982).
[39] M. Levin and A. Stern, *Fractional Topological Insulators*, Phys. Rev. Lett. **103**, 196803 (2009).
[40] D. Sheng, Z.-C. Gu, K. Sun, and L. Sheng, *Fractional Quantum Hall Effect in the Absence of Landau Levels*, Nat. Commun. **2**, 389 (2011).
[41] T. Neupert, L. Santos, C. Chamon, and C. Mudry, *Fractional Quantum Hall States at Zero Magnetic Field*, Phys. Rev. Lett. **106**, 236804 (2011).
[42] Y.-F. Wang, Z.-C. Gu, C.-D. Gong, and D. N. Sheng, *Fractional Quantum Hall Effect of Hard-Core Bosons in Topological Flat Bands*, Phys. Rev. Lett. **107**, 146803 (2011).
[43] K. Sun, Z. Gu, H. Katsura, and S. Das Sarma, *Nearly Flatbands with Nontrivial Topology*, Phys. Rev. Lett. **106**, 236803 (2011).
[44] N. Regnault and B. A. Bernevig, *Fractional Chern Insulator*, Phys. Rev. X **1**, 021014 (2011).
[45] A. E. B. Nielsen, G. Sierra, and J. I. Cirac, *Local Models of Fractional Quantum Hall States in Lattices and Physical Implementation*, Nat. Commun. **4**, 2864 (2013).
[46] B. Bauer, L. Cincio, B. P. Keller, M. Dolfi, G. Vidal, S. Trebst, and A. W. W. Ludwig, *Chiral Spin Liquid and Emergent Anyons in a Kagome Lattice Mott Insulator*, Nat. Commun. **5**, 5137 (2014).
[47] Y.-C. He, D. N. Sheng, and Y. Chen, *Chiral Spin Liquid in a Frustrated Anisotropic Kagome Heisenberg Model*, Phys. Rev. Lett. **112**, 137202 (2014).
[48] S.-S. Gong, W. Zhu, and D. N. Sheng, *Emergent Chiral Spin Liquid: Fractional Quantum Hall Effect in a Kagome Heisenberg Model*, Sci. Rep. **4**, 6317 (2014).
[49] G. Baskaran, Z. Zou, and P. Anderson, *The Resonating Valence Bond State and High-T_c Superconductivity—A Mean Field Theory*, Solid State Commun. **63**, 973 (1987).
[50] I. Affleck, Z. Zou, T. Hsu, and P. W. Anderson, *$SU(2)$ Gauge Symmetry of the Large-U Limit of the Hubbard Model*, Phys. Rev. B **38**, 745 (1988).
[51] X.-G. Wen, *Projective Construction of Non-Abelian Quantum Hall Liquids*, Phys. Rev. B **60**, 8827 (1999).
[52] W.-J. Hu, W. Zhu, Y. Zhang, S. Gong, F. Becca, and D. N. Sheng, *Variational Monte Carlo Study of a Chiral Spin Liquid in the Extended Heisenberg Model on the Kagome Lattice*, Phys. Rev. B **91**, 041124 (2015).
[53] V. Kalmeyer and R. B. Laughlin, *Equivalence of the Resonating-Valence-Bond and Fractional Quantum Hall States*, Phys. Rev. Lett. **59**, 2095 (1987).
[54] T. B. Wahl, H.-H. Tu, N. Schuch, and J. I. Cirac, *Projected Entangled-Pair States Can Describe Chiral Topological States*, Phys. Rev. Lett. **111**, 236805 (2013).
[55] J. Dubail and N. Read, *Tensor Network Trial States for Chiral Topological Phases in Two Dimensions and a No-Go Theorem in Any Dimension*, Phys. Rev. B **92**, 205307 (2015).
[56] S. Yang, T. B. Wahl, H.-H. Tu, N. Schuch, and J. I. Cirac, *Chiral Projected Entangled-Pair State with Topological Order*, Phys. Rev. Lett. **114**, 106803 (2015).
[57] D. Poilblanc, J. I. Cirac, and N. Schuch, *Chiral Topological Spin Liquids with Projected Entangled Pair States*, Phys. Rev. B **91**, 224431 (2015).
[58] S. Yang, T. B. Wahl, H.-H. Tu, N. Schuch, and J. I. Cirac, *Chiral Projected Entangled-Pair State with Topological Order*, Phys. Rev. Lett. **114**, 106803 (2015).
[59] W. L. McMillan, *Ground State of Liquid* He^4, Phys. Rev. **138**, A442 (1965).
[60] W. M. C. Foulkes, L. Mitas, R. J. Needs, and G. Rajagopal, *Quantum Monte Carlo Simulations of Solids*, Rev. Mod. Phys. **73**, 33 (2001).
[61] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, *Equation of State Calculations by Fast Computing Machines*, J. Chem. Phys. **21**, 1087 (1953).
[62] W. K. Hastings, *Monte Carlo Sampling Methods Using Markov Chains and Their Applications*, Biometrika **57**, 97 (1970).
[63] S. Sorella, *Generalized Lanczos Algorithm for Variational Quantum Monte Carlo*, Phys. Rev. B **64**, 024512 (2001).
[64] S. Sorella, *Wave Function Optimization in the Variational Monte Carlo Method*, Phys. Rev. B **71**, 241103 (2005).
[65] C. J. Umrigar and C. Filippi, *Energy and Variance Optimization of Many-Body Wave Functions*, Phys. Rev. Lett. **94**, 150201 (2005).
[66] M. P. Nightingale and V. Melik-Alaverdian, *Optimization of Ground- and Excited-State Wave Functions and van der Waals Clusters*, Phys. Rev. Lett. **87**, 043401 (2001).
[67] J. Toulouse and C. J. Umrigar, *Optimization of Quantum Monte Carlo Wave Functions by Energy Minimization*, J. Chem. Phys. **126**, 084102 (2007).
[68] C. J. Umrigar, J. Toulouse, C. Filippi, S. Sorella, and R. G. Hennig, *Alleviation of the Fermion-Sign Problem by Optimization of Many-Body Wave Functions*, Phys. Rev. Lett. **98**, 110201 (2007).
[69] R. Jastrow, *Many-Body Problem with Strong Forces*, Phys. Rev. **98**, 1479 (1955).
[70] S. R. White, *Density Matrix Formulation for Quantum Renormalization Groups*, Phys. Rev. Lett. **69**, 2863 (1992).
[71] S. Liang and H. Pang, *Approximate Diagonalization Using the Density Matrix Renormalization-Group Method: A Two-Dimensional-Systems Perspective*, Phys. Rev. B **49**, 9214 (1994).
[72] D.-L. Deng, X. Li, and S. Das Sarma, *Quantum Entanglement in Neural Network States*, Phys. Rev. X **7**, 021021 (2017).
[73] C. K. Majumdar and D. K. Ghosh, *On Next Nearest Neighbor Interaction in Linear Chain. I*, J. Math. Phys. (N.Y.) **10**, 1388 (1969).
[74] I. Affleck, T. Kennedy, E. H. Lieb, and H. Tasaki, *Rigorous Results on Valence-Bond Ground States in Antiferromagnets*, Phys. Rev. Lett. **59**, 799 (1987).
[75] X. Gao (private communication).

[76] O. Sikora, H.-W. Chang, C.-P. Chou, F. Pollmann, and Y.-J. Kao, *Variational Monte Carlo Simulations Using Tensor-Product Projected States*, Phys. Rev. B **91**, 165113 (2015).
[77] A. E. B. Nielsen, J. I. Cirac, and G. Sierra, *Laughlin Spin-Liquid States on Lattices Obtained from Conformal Field Theory*, Phys. Rev. Lett. **108**, 257206 (2012).
[78] H.-H. Tu, A. E. B. Nielsen, J. I. Cirac, and G. Sierra, *Lattice Laughlin States of Bosons and Fermions at Filling Fractions* $1/q$, New J. Phys. **16**, 033025 (2014).
[79] I. Glasser, J. I. Cirac, G. Sierra, and A. E. B. Nielsen, *Lattice Effects on Laughlin Wave Functions and Parent Hamiltonians*, Phys. Rev. B **94**, 245104 (2016).
[80] A. E. B. Nielsen, J. I. Cirac, and G. Sierra, *Quantum Spin Hamiltonians for the* $SU(2)_k$ *WZW Model*, J. Stat. Mech. (2011) P11014.
[81] E. Neuscamman, C. J. Umrigar, and G. K.-L. Chan, *Optimizing Large Parameter Sets in Variational Quantum Monte Carlo*, Phys. Rev. B **85**, 045103 (2012).
[82] S. Amari, *Natural Gradient Works Efficiently in Learning*, Neural Comput. **10**, 251 (1998).
[83] M. B. Hastings, I. González, A. B. Kallin, and R. G. Melko, *Measuring Renyi Entanglement Entropy in Quantum Monte Carlo Simulations*, Phys. Rev. Lett. **104**, 157201 (2010).
[84] J. Wildeboer and N. E. Bonesteel, *Spin Correlations and Topological Entanglement Entropy in a Non-Abelian Spin-One Spin Liquid*, Phys. Rev. B **94**, 045125 (2016).
[85] A. Kitaev and J. Preskill, *Topological Entanglement Entropy*, Phys. Rev. Lett. **96**, 110404 (2006).
[86] M. Levin and X.-G. Wen, *Detecting Topological Order in a Ground State Wave Function*, Phys. Rev. Lett. **96**, 110405 (2006).
[87] O. S. Zozulya, M. Haque, K. Schoutens, and E. H. Rezayi, *Bipartite Entanglement Entropy in Fractional Quantum Hall States*, Phys. Rev. B **76**, 125310 (2007).
[88] D. Poilblanc, *Investigation of the Chiral Antiferromagnetic Heisenberg Model Using PEPs*, Phys. Rev. B **96**, 121118 (2017).
[89] M. Greiter and R. Thomale, *Non-Abelian Statistics in a Quantum Antiferromagnet*, Phys. Rev. Lett. **102**, 207203 (2009).
[90] I. Glasser, J. I. Cirac, G. Sierra, and A. E. B. Nielsen, *Exact Parent Hamiltonians of Bosonic and Fermionic Moore-Read States on Lattices and Local Models*, New J. Phys. **17**, 082001 (2015).
[91] A. Novikov, M. Trofimov, and I. Oseledets, *Exponential Machines*, arXiv:1605.03795.
[92] E. M. Stoudenmire and D. J. Schwab, *Supervised Learning with Quantum-Inspired Tensor Networks*, Adv. Neural Inf. Process. Syst. **29**, 4799 (2016).
[93] Y. Nomura, A. Darmawan, Y. Yamaji, and M. Imada, *Restricted-Boltzmann-Machine Learning for Solving Strongly Correlated Quantum Systems*, Phys. Rev. B **96**, 205152 (2017).
[94] S. R. Clark, *Unifying Neural-Network Quantum States and Correlator Product States via Tensor Networks*, arXiv:1710.03545.
[95] R. Kaubruegger, L. Pastori, and J. C. Budich, *Chiral Topological Phases from Artificial Neural Networks*, arXiv:1710.04713.
[96] D. Tahara and M. Imada, *Variational Monte Carlo Method Combined with Quantum-Number Projection and Multi-Variable Optimization*, J. Phys. Soc. Jpn. **77**, 114701 (2008).