Density Matrix Renormalization Group with Tensor Processing Units

Martin Ganahl,[1,2,*] Jackson Beall,[1,2] Markus Hauru,[2,3] Adam G.M. Lewis,[1,2] Tomasz Wojno,[1,2]
Jae Hyeon Yoo,[2,4,5] Yijian Zou,[2,6] and Guifre Vidal[2,4,7]

[1] *SandboxAQ, Palo Alto, California, USA*
[2] *Sandbox@Alphabet, Mountain View, California 94043, USA*
[3] *The Alan Turing Institute, 96 Euston Road, London, England NW1 2DB, United Kingdom*
[4] *X, the Moonshot Factory, Mountain View, California 94043, USA*
[5] *Google Core, Mountain View, California 94043, USA*
[6] *Stanford Institute for Theoretical Physics, Stanford University, Palo Alto, California 94305, USA*
[7] *Google Quantum AI, Mountain View, California 94043, USA*

(Received 28 April 2022; revised 11 January 2023; accepted 26 January 2023; published 16 February 2023)

Google's tensor processing units (TPUs) are integrated circuits specifically built to accelerate and scale up machine learning workloads. They can perform fast distributed matrix multiplications and therefore be repurposed for other computationally intensive tasks. In this work we demonstrate the use of TPUs for accelerating and scaling up the density matrix renormalization group (DMRG), a powerful numerical approach to compute the ground state of a local quantum many-body Hamiltonian. The cost of DMRG scales with system size N as $O(ND^3)$, where the so-called bond dimension D regulates how expressive the underlying matrix product state (MPS) variational ansatz is. We consider lattice models in two spatial dimensions, with square lattices of size 10×10 (free fermions) and 20×20 (transverse field Ising model), for which the required MPS bond dimension is known to scale at least as $\exp(\sqrt{N})$. Using half of a TPU v3 pod (namely 1024 TPU v3 cores), we reach an unprecedentedly large bond dimension $D = 2^{16} = 65\,536$, for which optimizing a single MPS tensor takes about 2 min.

DOI: 10.1103/PRXQuantum.4.010317

I. INTRODUCTION

The density matrix renormalization group (DMRG) [1,2] algorithm is the gold standard method for computing ground states and low-lying excited states of one-dimensional (1D) local Hamiltonians [3–8]. Since its original formulation, the DMRG method and its descendants [9–15] have been applied to a wide variety of problems, both in one and higher dimensions, ranging from quantum chemistry [16–29] to material science [30–32], quantum computing [33–41] and machine learning [42–49] to solving partial differential equations [50,51]. Tensor network methods, of which DMRG is the most successful incarnation, hold the promise of revolutionizing these fields.

DMRG is an optimization method over a matrix product state (MPS) [52–56], which is a powerful variational ansatz in the form of a one-dimensional tensor network. For a system size N (where N denotes, e.g., the number of sites in a lattice model or the number of electronic orbitals in a molecule), the MPS is made of N tensors and the computational cost of DMRG scales as $O(ND^3)$. Here, the so-called bond dimension D determines how much quantum entanglement the MPS is capable of accounting for, and may depend on the system size, that is, $D = D(N)$. For instance, to accurately represent a generic wave function, the bond dimension must grow as $D = \exp(N)$, making the MPS optimization as expensive as a direct, brute-force ground-state computation. Fortunately, most ground states of local Hamiltonians in d spatial dimensions contain restricted amounts of entanglement according to the so-called *area law* (with possible logarithmic corrections). Both the incredible success of DMRG for one-dimensional systems and its challenges in higher dimensions can be understood to be a direct consequence of this area law.

Such applications of DMRG in $d > 1$ dimensions, while exceedingly difficult and computationally expensive, are also tremendously important in order to account for exotic quantum effects that other, less expensive methods fail to capture. Recent years have seen a growing effort to understand how modern computer architectures,

[*] martin.ganahl@gmail.com

in particular high performance computing clusters, can be used to speed up such computations [26,57–62]. In this work we show that Googles's tensor processing units (TPUs) [63,64], originally developed for machine learning workloads but more recently applied to other computational tasks [65–73], can be leveraged to perform, within hours, large scale DMRG calculations of 2D quantum systems that would otherwise take many months to years to finish on conventional shared memory hardware with up to a few dozen CPU cores. We demonstrate the approach in 2D square lattice models of sizes 10×10 and 20×20. To the best of our knowledge, the largest bond dimension employed, $D = 2^{16} = 65\,536$, sets a new record. (This is achieved using only 1024 TPU v3 cores, namely half of a TPU v3 pod, and without exploiting internal symmetries in the MPS representation, and therefore there is significant room for further increasing D; see Sec. VI). These results herald a new age of DMRG and, more generally, tensor network methods, with the potential to transform the computational landscape in all research areas where such techniques are applied, from condensed matter to quantum chemistry, materials science and machine learning.

The paper is organized as follows: in Sec. II we review some relevant aspects of the DMRG algorithm, the MPS ansatz, and the entanglement area law; in Sec. III we then briefly describe TPUs; in Sec. IV we introduce the strategy used to distribute DMRG on TPUs, including data distribution, a necessary out-of-core approach, and distributed tensor contractions; in Sec. V we present benchmark results using two models on a square lattice: free fermions and the transverse field Ising model; we conclude the paper with a summary and discussion in Sec. VI. We also include Appendices A–C with additional technical details of our DMRG implementation.

II. DENSITY MATRIX RENORMALIZATION GROUP

In this section we present a brief review of the DMRG algorithm [1] and the MPS ansatz [52], as well as other relevant background material.

We consider a lattice system made of N sites, with each site described by a vector space of finite dimension q and orthonormal basis $\{|i\rangle\}$, $i = 1, 2, \ldots, q$. For instance, with $q = 2$, each site is represented by a two-dimensional vector space with orthonormal basis $\{|1\rangle, |2\rangle\}$, corresponding, e.g., to empty or occupied fermionic states $\{|0\rangle, |1\rangle\}$ if each site represents a spinless fermionic degree of freedom, or to spin-up or spin-down states $\{|\uparrow\rangle, |\downarrow\rangle\}$ if each site represents a spin-$\frac{1}{2}$ quantum spin degree of freedom, as in the two examples used later in this paper. The many-body wave function of the lattice system then reads

$$|\psi\rangle = \sum_{i_1 i_2 \cdots i_N} \psi^{i_1 i_2 \cdots i_N} |i_1 i_2 \cdots i_N\rangle, \tag{1}$$

where $\psi^{i_1 i_2 \cdots i_N}$ denotes q^N (possibly complex) amplitudes and $|i_1 i_2 \cdots i_N\rangle$ stands for the product basis $|i_1\rangle \otimes |i_2\rangle \otimes \cdots \otimes |i_N\rangle$ for the q^N-dimensional vector space of the N sites. Similarly, the local many-body Hamiltonian expressed in the same basis reads

$$\mathcal{H} = \sum_{\{i\},\{j\}} \mathcal{H}^{i_1 j_1 i_2 j_2 \cdots i_N j_N} |i_1 i_2 \cdots i_N\rangle \langle j_1 j_2 \cdots j_N|, \tag{2}$$

although a more natural, efficient expression is as a sum of local terms. Our goal is to compute an accurate approximation to the ground state $|\psi_{\mathrm{GS}}\rangle$ of the lattice Hamiltonian $\mathcal{H}$, without explicitly storing the wave-function amplitudes in Eq. (1), which would incur a computational cost exponential in the system size N.

A. Matrix product decompositions

For that purpose, one can use the DMRG algorithm [1], which is a variational method in the space of MPSs [52]. The MPS ansatz consists of a collection of N order-3 tensors

$$\{M_1, M_2, \ldots, M_N\}. \tag{3}$$

Each tensor M_k has (possibly complex) components $[M_k]^{i_k}_{\alpha_{k-1}\alpha_k}$, where each index α_k takes D_k different values (that is, $\alpha_k = 1, 2, \ldots, D_k$). In other words, for each value of $i_k \in \{1, 2, \ldots, q\}$, tensor M_k defines a matrix $[M_k]^{i_k}$ of size $D_{k-1} \times D_k$ with matrix elements labeled by indices α_{k-1} and α_k. Following a common practice, we refer to the index i_k labeling the local basis of states as a *physical index*, and to the indices α_{k-1} and α_k as *bond indices*. The bond indices α_0 and α_N will be chosen to have dimensions $D_0 = D_N = 1$, so that, for fixed value of the physical indices i_1 and i_N, $[M_1]^{i_1}$ and $[M_N]^{i_N}$ are not matrices but vectors (of dimension D_1 and D_{N-1} and components $[M_1]^{i_1}_{1\alpha_1}$ and $[M_2]^{i_N}_{\alpha_{N-1}1}$, respectively). Given the above N tensors, the MPS ansatz assumes that the q^N wave-function amplitudes in Eq. (1) can be written as

$$\begin{aligned}\psi^{i_1 i_2 \cdots i_N} &= \sum_{\{\alpha\}} [M_1]^{i_1}_{1\alpha_1} [M_2]^{i_2}_{\alpha_1\alpha_2} \cdots [M_N]^{i_N}_{\alpha_{N-1}1} \\ &= [M_1]^{i_1} \cdot [M_2]^{i_2} \cdot \cdots \cdot [M_N]^{i_N},\end{aligned} \tag{4}$$

where in the first line we have explicitly written both the indices of each tensor and the sum over the matrix indices $\{\alpha\} = \alpha_1, \alpha_2, \ldots, \alpha_{N-1}$, whereas in the second line we regarded each $[M_k]^{i_k}$ as a matrix (except for $[M_1]^{i_1}$ and $[M_N]^{i_N}$, which are vectors) and used the matrix product symbol '$\cdot$' to represent matrix-matrix multiplication (respectively, matrix-vector multiplication). Note that the name 'matrix product state' of this variational ansatz comes from the fact that it expresses the wave-function amplitudes as the matrix product (4). We note in passing that any wave function $\psi^{i_1 i_2 \cdots i_N}$ can be cast in principle into a MPS form using a sequence of SV or QR decompositions. However, such an approach in general requires exponential computational resources.

Intuitively, the MPS ansatz assumes that state $|\psi\rangle$ has a restricted amount of entanglement. Indeed, the so-called bond dimension D_k limits how much entanglement the ansatz can represent between two parts of the system, namely between a part containing the first k sites (from site 1 to site k) and another part containing the rest of the sites (from site $k+1$ to site N). In particular, the larger the bond dimension D_k, the more entanglement the MPS can account for between these two parts, and thus the more capable the ansatz is to represent entangled many-body wave functions. On the other hand, tensor M_k contains $q \times D_{k-1} \times D_k$ variational parameters, so that the cost of storing the MPS grows with the bond dimensions. Quite often, the bond dimension D_k is chosen according to the rule

$$D_k = \begin{cases} q^k & \text{for } 1 \le k < k_1, \\ D & \text{for } k_1 \le k \le k_2, \\ q^{N-k} & \text{for } k_2 < k \le N-1, \end{cases} \tag{5}$$

namely such that it grows exponentially with k for small k until it reaches some maximum allowed bond dimension D, and similarly with $N-k$ for large $k < N$. Given a choice of maximum bond dimension D, k_1 above is simply the smallest site index such that $D < q^{k_1}$, whereas k_2 is the largest site index such that $D < q^{N-k_2}$. In many applications the above prescription implies that most of the MPS tensors have size $q \times D^2$. Then the memory space required to store the MPS scales with the bond dimension D and system size N as $O(ND^2)$.

The MPS ansatz comes with so-called gauge freedom [74–76], in that the amplitudes $\psi^{i_1 i_2 \cdots i_N}$ are invariant under the simultaneous change $[M_k]^{i_k} \to [M_k]^{i_k} \cdot Q_k$ and $[M_{k+1}]^{i_{k+1}} \to Q_k^{-1} \cdot [M_{k+1}]^{i_{k+1}}$, where Q_k is any invertible $D_k \times D_k$ matrix and Q_k^{-1} is its inverse. Indeed, the double replacement is easily seen to leave the matrix product $[M_k]^{i_k} \cdot [M_{k+1}]^{i_{k+1}}$ invariant in Eq. (4), so that the wave-function amplitudes are not changed. Using this gauge freedom, we can bring MPS (3) into the so-called *central* gauge [74,75] with respect to site n,

$$\{A_1, \ldots, A_{n-1}, C_n, B_{n+1}, \ldots, B_N\}, \tag{6}$$

where letters A and B are used to denote MPS tensors that satisfy the following orthogonality constraints:

$$\sum_{i_k} ([A_k]^{i_k})^\dagger \cdot [A_k]^{i_k} = \mathbb{1} \quad \text{for all } k < n, \tag{7}$$

$$\sum_{i_k} [B_k]^{i_k} \cdot ([B_k]^{i_k})^\dagger = \mathbb{1} \quad \text{for all } k > n. \tag{8}$$

The central tensor C_n above does not satisfy any of the two relations. The central form is important in order to both simplify, and provide numerical stability to, the MPS optimization procedure, as briefly reviewed below.

The system's Hamiltonian $\mathcal{H}$ in Eq. (2) can be similarly expressed in matrix product operator (MPO) [77–81] form, given in terms of a sequence of N order-4 tensors

$$\{H_1, H_2, \ldots, H_N\}, \tag{9}$$

where each tensor H_k has components $[H_k]^{i_k j_k}_{m_{k-1} m_k}$. For fixed values of the physical indices i_k and j_k, we can think of $[H_k]^{i_k j_k}$ as a $D'_{k-1} \times D'_k$ matrix. We again refer to i_k and j_k as *physical* indices and to m_{k-1} and m_k as MPO *bond* indices. The Hamiltonian coefficients in Eq. (2) can then be written as

$$\begin{aligned} \mathcal{H}^{i_1 j_1 i_2 j_2 \cdots i_N j_N} &= \sum_{\{m\}} [H_1]^{i_1 j_1}_{1 m_1} [H_2]^{i_2 j_2}_{m_1 m_2} \cdots [H_N]^{i_N j_N}_{m_{N-1} 1} \\ &= [H_1]^{i_1 j_1} \cdot [H_2]^{i_2 j_2} \cdot \cdots \cdot [H_N]^{i_N j_N}. \end{aligned} \tag{10}$$

Importantly, in this case the MPO representation is not used as a variational ansatz for the Hamiltonian $\mathcal{H}$, but as a convenient way of exactly representing it.

B. Variational energy optimization

The exact ground state $|\psi_{\text{GS}}\rangle$ of Hamiltonian $\mathcal{H}$ is state $|\psi\rangle$ that minimizes the expectation value of the energy, as given by $E(|\psi\rangle) \equiv \langle\psi|\mathcal{H}|\psi\rangle / \langle\psi|\psi\rangle$. Accordingly, a MPS approximation $|\psi^\star_{\text{MPS}}\rangle$ to the ground state $|\psi_{\text{GS}}\rangle$ is obtained by minimizing the energy

$$E(|\psi_{\text{MPS}}\rangle) = \frac{\langle\psi_{\text{MPS}}|\mathcal{H}|\psi_{\text{MPS}}\rangle}{\langle\psi_{\text{MPS}}|\psi_{\text{MPS}}\rangle} \tag{11}$$

over the set of states $|\psi_{\text{MPS}}\rangle$ that can be written as a MPS (for some fixed choice of bond dimensions $\{D_k\}$),

$$|\psi^\star_{\text{MPS}}\rangle = \arg\min_{|\psi_{\text{MPS}}\rangle} E(|\psi_{\text{MPS}}\rangle). \tag{12}$$

In the following we outline the main steps of the DMRG algorithm, which aims at obtaining $|\psi^\star_{\text{MPS}}\rangle$, and refer the reader to the literature [1,75] for more details.

Starting from some initial state $|\psi_{\text{MPS}}\rangle$ given by a set of MPS tensors $\{M_k\}$ in Eq. (3), the DMRG algorithm attempts to minimize the energy $E(|\psi_{\text{MPS}}\rangle)$ in Eq. (11) by iteratively optimizing one MPS tensor at a time. More concretely, we first optimize the tensor on site $n=1$, then the tensor on site $n=2$, etc., all the way to the tensor on site $n=N$, in what is known as a *forward sweep*. That is followed by a *backward sweep*, progressing from the last site to the first one. For a given site n, the optimization proceeds as follows. First, the MPS is written in the central canonical form for that site, namely as in Eq. (6).

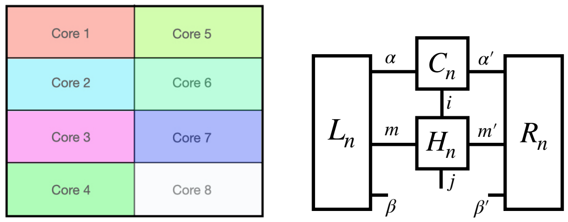

FIG. 1. Left: matrix distribution pattern on TPU cores. The figure shows the distribution pattern of a $D \times D$ matrix on eight available TPU cores, arranged on a 4×2 torus. Each colored panel has shape $D/4 \times D/2$. Right: tensor network diagram for a key tensor contraction in the DMRG algorithm, where the one-site MPS tensor (regarded as an effective vector wave function for a given site) is multiplied by three other tensors (corresponding to an effective Hamiltonian for that site).

Then the central tensor C_n is replaced with a new tensor C'_n that is chosen in a way as to optimally lower $E(|\psi_{\text{MPS}}\rangle)$ while keeping the rest of the MPS tensors constant. This is accomplished by diagonalizing a linear operator using a Krylov-space method such as the Lanczos method (see Fig. 1 and the text below for more details). The leading computational cost of the DMRG algorithm scales as $O(N(qD'D^3 + q^2D'^2D^2))$.

The DMRG algorithm proceeds with forward and backward sweeps until either the energy $E(|\psi_{\text{MPS}}\rangle)$ has converged to some desired accuracy, or a maximum sweep number has been reached. Although ideally one would like to obtain the optimal MPS ground-state approximation $|\psi^{\star}_{\text{MPS}}\rangle$ in Eq. (12), in practice, it is understood that one must settle for a reasonably converged MPS ground-state approximation, also denoted $|\psi^{\star}_{\text{MPS}}\rangle$ in the rest of this paper.

C. Computational cost and area law

For simplicity, from now on we assume a uniform MPS bond dimension D. Optimizing a central tensor $C_n \to C'_n$ (e.g., using the Lanczos method) as well as shifting the central canonical form (6) from site n to site $n+1$ (which can be accomplished using a polar, QR, or singular value decomposition [82]) both have a computational cost $O(D^3)$, leading to a total cost per optimization sweep that scales with bond dimension D and system size N as $O(ND^3)$. It is important to emphasize the *linear* scaling in N at fixed D, as opposed to the *exponential* scaling incurred in a brute-force ground-state computation using the q^N amplitudes in Eq. (1). However, in order to achieve some desired accuracy with a MPS computation, the bond dimension D may need to be adjusted as a function of the system size N, that is, in general $D = D(N)$, in which case the scaling of computational resources may no longer be linear in N.

In order to gain further insight into computational costs, a useful rule of thumb is to think that the bond dimension D must grow exponentially with the entanglement entropy S [83–85] of the target ground state $|\psi_{\text{GS}}\rangle$ for half of the system, that is, $D \sim \exp(S)$. For a generic (nonlocal) Hamiltonian $\mathcal{H}$, the half-system entropy of the ground state is expected to grow linearly in the system size, $S \approx N$, a scaling known as the entanglement *volume law*. In this case, the bond dimension must grow exponentially with system size, $D \sim \exp(N)$, and using a MPS representation is not computationally advantageous with respect to a brute-force computation. Luckily, ground states of local Hamiltonians often have a more forgiving scaling of entanglement entropy S with system size N, known as the entanglement *area law* [83,85–92] (sometimes with a logarithmic correction), which justifies the use of a MPS representation and the DMRG algorithm.

Specifically, in a d-dimensional cubic lattice made of $N = L^d$ sites, the half-system entropy of a state obeying the *area law* scales as

$$S \sim L^{d-1} = N^{(d-1)/d} \quad \text{(area law)} \tag{13}$$

or, in the presence of a logarithmic correction, as

$$S \sim N^{(d-1)/d} \times \log N \quad \text{(logarithmic correction).} \tag{14}$$

The above rule of thumb then indicates that the required MPS bond dimension should respectively scale as

$$D \sim \exp(N^{(d-1)/d}) \quad \text{(area law),} \tag{15}$$

$$D \sim \exp(N^{(d-1)/d}) \times \text{poly}(N) \quad \text{(logarithmic correction).} \tag{16}$$

In $d = 1$ dimensions, the ground state typically obeys an area law if the local Hamiltonian has a finite energy gap. In this case $S \sim N^0$ suggests that a constant bond dimension D, independent of the system size N, may suffice to accurately approximate the ground state $|\psi_{\text{GS}}\rangle$ with a MPS, resulting in an overall computational cost linear in N. For gapless Hamiltonians in $d = 1$ dimensions, the ground state often exhibits a logarithmic correction to the area law, $S \sim \log(N)$, resulting in a polynomial bond dimension $D \sim \text{poly}(N)$ and thus also a computational cost that grows as some power of the system size N. We conclude that DMRG is efficient for ground states of $d = 1$ systems.

In $d = 2, 3$ dimensions, ground states of gapped and gapless local Hamiltonians often obey the entanglement area law, with some gapless systems (e.g., systems with a Fermi surface of dimension $d-1$) also displaying logarithmic corrections. For such systems, the above rule of thumb indicates that DMRG has cost that scales at least as

$\exp(\sqrt{N})$ and $\exp(N^{2/3})$ in $d = 2, 3$ dimensions, respectively. Note that, in spite of this exponential scaling of computational costs (in a fractional power of N), DMRG still has significant advantage with respect to the $\exp(N)$ scaling of a brute-force computation.

While the area law (with possible logarithmic corrections in certain critical systems) has mostly been investigated in regularly structured lattice models, we expect it to also roughly apply to more generic systems, such as large molecules in $d = 3$ dimensions, where DMRG is used in the context of quantum chemistry [16–29].

In conclusion, DMRG is efficient in $d = 1$ dimensions, where it is firmly established as the method of choice to compute ground states. On the other hand, DMRG scales exponentially as $\exp(\sqrt{N})$ and $\exp(N^{2/3})$ in $d = 2, 3$ dimensions. In spite of this unfavorable scaling, DMRG is actively used in restricted $d > 1$ geometries such as thin two-dimensional strips and cylinders [93–107] and small three-dimensional molecules [29,108–115]. In such cases, the massive computational cost of running DMRG constitutes the main roadblock to studying larger systems. As we show in this work using tensor processing units, specialized hardware originally developed to accelerate and scale up machine learning workloads can be repurposed to also accelerate and scale up DMRG computations, significantly increasing the bond dimension D that can be afforded. This allows us to use DMRG to more accurately address larger systems in $d > 1$ dimensions.

III. TPU CORES, BOARDS, AND PODS

TPUs are application-specific integrated circuits developed by Google specifically for large-scale machine learning applications [63,64]. However, in recent times a growing number of papers have demonstrated their applicability to accelerating and scaling up other computationally intensive tasks, including large-scale dense linear algebra operations [67], the simulation of quantum circuits [70,71], brute-force ground-state computation and dynamics simulation in quantum many-body systems [65,66,69], and quantum chemistry electronic structure computations using density functional theory [68,72,73]. In this work we focus on TPUs of third generation, denoted v3 in the following. (After completion of our work, TPUs of fourth generation, with increased compute power, were made available. The results presented in this work can be straightforwardly generalized to TPU v4.) In the third generation, eight TPU v3 cores form a TPU board, and up to 256 TPU v3 boards can be connected into a TPU pod (with 2048 TPU v3 cores).

A single TPU v3 core is equipped with two matrix-multiplication units (MXUs) and 16 GB of on-chip, high-bandwidth memory (HBM). A MXU is a systolic array that can multiply matrices of size 128×128 natively, using multiplication of floating numbers in half precision (specifically, in *brain float* 16 format, or bf16) and accumulation in single precision (fp32). Using six passes through the MXU, a single TPU core can however also deliver over 10 TFLOPS of single-precision (fp32) matrix-matrix multiplication.

At the next level we find a TPU board, which is actually the smallest available configuration, with eight TPU v3 cores and one controlling host CPU machine. The eight cores are arranged on a 2D torus and, importantly, each core is connected to its neighbors through a fast intercore interconnect (ICI) communication link (with 656-GB/s bandwidth [64]). A TPU board has a total of 128 GB of HBM and can yield up to about 80 TFLOPS of single-precision matrix-matrix multiplication [67].

Finally, up to 256 TPU boards (that is, up to 2048 TPU cores) can be joined into a TPU v3 pod, where the cores are again arranged on a 2D torus and directly connected to nearest neighbors with ICI links, with a total of 32 TB of HBM and near 20 PFLOPS of single-precision matrix-matrix multiplication [67]. One can also use a slice of a pod containing an intermediate number of TPU cores. For instance, in Fig. 4 below we provide performance results and estimates for slices with 32, 128, 512, and 2048 cores. The largest TPU configuration we use in this work is half a pod (that is, 1024 cores).

TPUs can be programmed using XLA [116], an optimized graph compiler that translates from roughly C-like commands called HLOs to roughly assemblylike equivalents called LLOs. The HLOs themselves may be written directly, but are usually instead “traced” from any of several higher-level languages. For the DMRG work presented in this paper, we wrote the code with Jax [117], a NumPy-like interface to XLA, following the single-instruction multiple data (SIMD) paradigm.

IV. DMRG ON TPUS

The performance of our large-scale implementation of DMRG on multicore TPU configurations is based on three main points: (i) individual MPS tensors (and other auxiliary tensors) are distributed through the available TPU cores; (ii) an out-of-core approach is adopted in order to more efficiently use the 16 GB of high-bandwidth memory on each TPU core; (iii) tensor contractions are accelerated through parallelization.

A. Data distribution

The largest data objects in a DMRG simulation are (a) the N order-3 MPS tensors M_n that contain the variational parameters of the ansatz and (b) two sets of N auxiliary tensors L_n and R_n, called left and right environment Hamiltonian tensors [1,75], which, given a choice of central site n, represent a contraction of MPS and MPO tensors for all sites $k < n$ and all sites $k > n$, respectively. In components, we can write $[M_n]^i_{\alpha\beta}$, $[L_n]^m_{\alpha\beta}$, $[R_n]^m_{\alpha,\beta}$, where greek letters are used to denote "large" indices, such as the MPS bond indices, with size D that in our implementation could potentially be scaled up to $D \sim 10^5$, whereas roman letters are used to denote "small" indices, namely the physical index i taking q values for $q = 2$ in the examples below, and the MPO bond dimension D', which in those examples grows up to $D' \sim 100$.

Let T denote any of these order-3 tensors, with components $[T]^i_{\alpha\beta}$. In our implementation, we regard tensor T as a collection $\{T_{i=1}, T_{i=2}, \ldots\}$ of large matrices, where each matrix T_i has components $[T_i]_{\alpha\beta}$ given by $[T]^i_{\alpha\beta}$. Each matrix T_i is then distributed across all available TPU cores in a checkerboard fashion, as shown in Fig. 1 for the case of eight TPU cores. Each matrix panel is stored in the high-bandwidth memory of the corresponding TPU core. Since in SIMD code each matrix panel is expected to have the same size, matrix dimensions are chosen appropriately such that they can be evenly divided by the grid shape of the TPU cluster. The motivation to distribute data in this way will become clear below, where tensor contractions are reduced to sequences of distributed matrix-matrix multiplications.

B. Out-of-core approach

The memory required to store the MPS tensors and the left and right environment Hamiltonian tensors scales as $O(ND^2q)$ and $O(ND^2D')$, respectively, where N is the number of sites, D the MPS bond dimension, q the physical index dimension, and D' the MPO bond dimension. For large DMRG computations, these memory requirements can quickly become prohibitive, when compared to the total available HBM on TPUs. For example, for a system of two-level quantum degrees of freedom on each site (local dimension $q = 2$) on a square lattice made of a 10×10 grid (number of sites $N = 100$) with a local Hamiltonian consisting only of a nearest-neighbor fermionic hopping term (MPO bond dimension $D' = 22$) and with a MPS bond dimension $D = 2^{16} = 65\,536$, the minimally required memory using single precision (4 bytes per fp32) grows to about 32 TB, which therefore already exhausts the maximally available 32-TB HBM of an entire TPU v3 pod.

On the other hand, at any given time of a DMRG optimization sweep only a small number of such tensors are really required for processing on the TPUs. We have therefore adopted an out-of-core approach, where the bulk of the data is stored on hard drives (which are readily and cheaply available). For one optimization step of the DMRG algorithm, the necessary data are read from hard drive into the HBM of the TPUs, where the relevant optimization step is executed, then the result is written back to the hard drive. To minimize the idling time of the TPUs, we utilize three simultaneous threads to perform DMRG optimization on the TPUs, and reading and writing of data from and to the disk.

C. Distributed tensor contractions

To illustrate how tensor contractions are performed in a distributed way, let us consider the contraction of the tensor network shown in the right panel of Fig. 1, which is the bottleneck operation in the DMRG algorithm. Given a site n, the tensor network contains the central MPS tensor C_n for that site, as well as the left and right environment Hamiltonian tensors L_n and R_n and the MPO tensor H_n. Conceptually, we can think of the contraction of this tensor network as corresponding to a "vector-matrix" multiplication, if we regard the central MPS tensor C_n as representing a "vector" (an effective wave function on site n) and the remaining three tensors as representing a "matrix" (an effective Hamiltonian H_{eff} on site n). This effective "vector-matrix" multiplication needs to be performed several times as part of the Lanczos tridiagonalization (which aims to compute the ground state of the effective Hamiltonian H_{eff} on site n as part of a single DMRG optimization step).

In order to proceed, we first preprocess the MPO tensor H_n, with components $[H_n]^{ij}_{\alpha\beta}$ that often vanish (sparse MPO), into a list of nonzero components $v = \{v_1, v_2, \ldots\}$ and their corresponding multi-indices $p = \{(a_1, b_1, c_1, d_1), (a_2, b_2, c_2, d_2), \ldots\}$ such that $H^{a_k b_k}_{c_k d_k} = v_k$.

The tensor network contraction is then performed by looping over all elements in the lists; see Algorithm 1 below. Note that, for each nonzero value in v, we must perform two matrix-matrix multiplications involving matrices from tensors L_n, C_n, and R_n. As explained above, each of these matrices has been suitably distributed among the TPU cores. Then we multiply them using the *scalable universal matrix multiplication algorithm* (SUMMA) [118], following a TPU implementation discussed in Ref. [67]. Each iteration of the loop produces a different distributed matrix, which is weighted by the corresponding weight v_k and added to one of the q matrices that will constitute the final order-3 tensor with the result of the tensor network contraction.

This approach is particularly appealing for highly sparse MPO tensors, as one typically finds when the MPO is

```
function CONTRACT($(i, v, L, R, C)$)
    $M$ = zeros_like($(C)$)                    ▷ Container for storing the final contraction result
    for $n = 0 \ldots \mathrm{len}(i) - 1$ do
        $a, b, c, d = i[n]$
        $T$ = SUMMA(SUMMA($(L[c, \ldots], C[a, \ldots]), R[d, \ldots]$))
        $M[b, \ldots] += v[n]*T$                 ▷ Accumulate matrix multiplications
    end for
    return $M$
end function
```

Algorithm 1. Contraction algorithm

encoding a local Hamiltonian of a lattice model. For dense MPO matrices, as, e.g., appearing in some quantum chemistry applications of DMRG, a vectorized approach can be more efficient.

While the discussion above focuses on the computation of the energy, the computation of other observables (local or nonlocal) can be carried out in the same way using standard techniques.

Another important step in the DMRG algorithm is tensor orthogonalization [75], which is traditionally implemented using a QR decomposition or a singular value decomposition. In this work we choose to perform orthogonalization using instead a polar decomposition (which, in a so-called two-site DMRG approach can also be used for optimal tensor truncation). Further implementation details can be found in the Appendix A.

V. RESULTS

In order to benchmark our distributed implementation of DMRG on TPUs, we compute a MPS approximation $|\psi^{\star}_{\mathrm{MPS}}\rangle$ to the ground state $|\Psi_{\mathrm{GS}}\rangle$ of two different 2D square lattice models. The first one is a model of free spinless fermions on a lattice of size 10×10, which can also be solved efficiently using the free fermion formalism, so that we have the exact solution to compare against. Its ground state displays a logarithmic correction to the area law, making this model extremely challenging from a DMRG perspective. The second model is the transverse field Ising model on a lattice of size 20×20, for which we do not have an exact solution, but other techniques can be used. The ground state of this model obeys an area law. This makes it less computationally demanding for DMRG, allowing us to consider a larger lattice.

The two models analyzed in this section are already well understood. We have chosen them mostly for two reasons. On the one hand, they are challenging from a DMRG perspective and, as such, can be used to meaningfully illustrate the use of very large bond dimension, as made available by TPUs. On the other hand, such models are often also used to benchmark other methods, including quantum Monte Carlo [119] and numerical linked-cluster expansions [120] or other tensor network algorithms such as those based on a tree tensor network [121], the multiscale entanglement renormalization ansatz [122,123], and projected entangled pair states (PEPSs) [124]. Benchmarking DMRG on the same models (although for different system sizes) enables useful comparisons.

A. Free fermion model

We first consider, on a 10×10 square lattice with $N = 100$ sites, the nearest-neighbor Hamiltonian

$$\mathcal{H}_{\mathrm{SF}} = -\sum_{\langle i,j \rangle} \hat{c}_i^{\dagger} \hat{c}_j + \mu \sum_i \hat{c}_i^{\dagger} \hat{c}_i \tag{17}$$

with $\hat{c}_i$ (anticommuting) fermionic annihilation operators and μ the chemical potential. This model describes a system of noninteracting electrons that can hop from each site to its nearest-neighboring ones, where the value of μ can be tuned to determine the number of electrons in the ground state (e.g., $N/2 = 50$ particles for $\mu = 0$). Using the free fermion formalism, the quadratic Hamiltonian (17) can be numerically diagonalized with computational cost that scales just as $O(N^3)$, instead of the generic $O(\exp(N))$ of a brute-force diagonalization. This is in contrast with the interacting case (e.g., if we added quartic terms to the above Hamiltonian), where the free fermion formalism can no longer be used. Here, the ground-state energy from the $O(N^3)$ diagonalization will be used to assess the accuracy of the DMRG result.

It is important to emphasize that, despite the lack of interactions, computing the ground state of Hamiltonian (17) is still a formidable challenge from the perspective of the DMRG algorithm. Indeed, for a sufficiently small value of $|\mu|$, this Hamiltonian is seen to describe a system with a one-dimensional Fermi surface, which results in the presence of a large number of gapless excitations. As such, its ground state $|\psi_{\mathrm{GS}}\rangle$ displays a logarithmic correction to the area law [83,89], implying that an accurate MPS approximation $|\Psi^{\star}_{\mathrm{MPS}}\rangle$ requires a bond dimension D expected to scale faster than $O(\exp(\sqrt{N}))$; see Eqs. (14) and (16) for $d = 2$. This is the strongest scaling of ground-state entanglement (and bond dimension D) observed to naturally occur in condensed matter systems in $d = 2$ dimensions. Thus, as far as DMRG is concerned, this noninteracting lattice model is not easier than a strongly interacting lattice model.

Figure 2 presents the DMRG approximations $E(|\psi^{\star}_{\mathrm{MPS}}\rangle)$ for the ground-state energy $E(|\psi_{\mathrm{GS}}\rangle)$ for $\mu = 0$ (half filling). The Hamiltonian is encoded in a MPO with bond dimension $D' = 22$. The top panel shows the converged DMRG energy as a function of the bond dimension D. The red line denotes the numerically exact value obtained from a $O(N^3)$ diagonalization. The bottom panel in Fig. 2 shows the evolution of the relative error of this energy as a function of bond dimension D. At $D = 2^{16}$ the approximation achieves a relative accuracy of less than one part in a million. Note that simulations are carried out in single precision, limiting the maximum achievable accuracy to about 10^{-7}. To the best of our knowledge, these are the largest DMRG computations (in terms of bond dimension) to date. Results for $D = 2^{16} = 65\,536$ are obtained within roughly 15 h on a slice of a TPU v3 pod made of 512 cores. We use 10 sweeps and a Krylov dimension of four for the sparse diagonalization required for optimizing each tensor.

A remark regarding internal symmetries in the MPS representation is in order. Hamiltonian $\mathcal{H}_{\mathrm{SF}}$ commutes with the particle number operator $\mathcal{N} = \sum_i \hat{c}_i^{\dagger} \hat{c}_i$, indicating

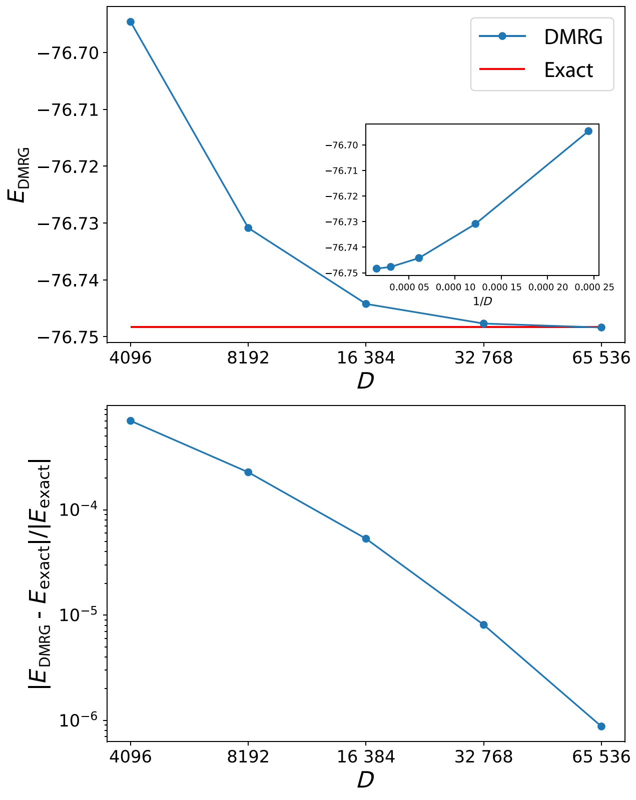

FIG. 2. Top: convergence of the DMRG ground-state energy towards the exact value as a function of bond dimension D for a 2D system of spinless free fermions on a 10×10 square lattice. Bottom: convergence of the relative error in energy for the same system.

particle number preservation, an internal U(1) symmetry generated by $\mathcal{N}$. Thus, its ground state $|\psi_{\mathrm{GS}}\rangle$ also has a well-defined particle number N_{GS}, $\mathcal{N}|\psi_{\mathrm{GS}}\rangle = N_{\mathrm{GS}}|\psi_{\mathrm{GS}}\rangle$. This can be exploited in DMRG [76,125–128]. Indeed, by specializing the MPS tensors M_k to be themselves invariant (covariant) under the Abelian U(1) symmetry group, we can ensure that the MPS representation is exactly symmetric with the correct number N_{GS} of particles, $\mathcal{N}|\psi_{\mathrm{MPS}}\rangle = N_{\mathrm{GS}}|\psi_{\mathrm{MPS}}\rangle$. In addition, this confers each MPS tensor a block-sparse structure that significantly reduces the number of variational parameters to be optimized, as well as the required computational cost.

Our current distributed implementation of DMRG on TPUs does not enforce or exploit the above model's internal U(1) symmetry. Our goal here is to benchmark the performance of DMRG in a way that the results are representative of a more general 2D lattice model, where such internal symmetry may not be present (see, e.g., our next example). We foresee nevertheless no obstruction to incorporating particle conservation in our current implementation.

B. Transverse field Ising model

As a second benchmark, we also consider the transverse field Ising model,

$$\mathcal{H}_{\mathrm{TFI}} = -\sum_{\langle i,j\rangle} \hat{\sigma}_i^z \hat{\sigma}_j^z + B\sum_i \hat{\sigma}_i^x \qquad (18)$$

on a 20×20 square lattice with $N = 400$ sites. Here $\hat{\sigma}_i^x$ and $\hat{\sigma}_i^z$ are Pauli matrices and B the magnetic field strength. This model represents a system of spin-$\frac{1}{2}$ quantum spin degrees of freedom with ferromagnetic interaction $\hat{\sigma}_i^z\hat{\sigma}_j^z$ between nearest-neighbor spins and subject to an external transverse magnetic field. This model is invariant under spin-flip transformations (internal Z_2 symmetry) generated by the unitary operator $U = \prod_i \hat{\sigma}_i^x$, which we again do not enforce or exploit in our DMRG implementation. The model has a quantum critical point for $B_c \approx 3.04$, and thus we expect the ground state $|\psi_{\mathrm{GS}}\rangle$ at or near this value of the magnetic field to be robustly entangled. Since there is no Fermi surface, the ground-state entanglement entropy scales as an area law without logarithmic corrections, as previously confirmed using other methods [119,120,122,123]. Accordingly, the bond dimension D required for an accurate approximation $|\psi^{\star}_{\mathrm{MPS}}\rangle$ scales as $O(\exp(\sqrt{N}))$; see Eqs. (13) and (15) for $d = 2$. This is still a very challenging computation for DMRG, but the milder scaling of the entanglement entropy (compared to the free fermion model above) allows us to consider a larger lattice.

Figure 3 shows the DMRG approximation $E(|\psi^{\star}_{\mathrm{MPS}}\rangle)$ for the (unknown) ground-state energy $E(|\psi_{\mathrm{GS}}\rangle)$ of the transverse field Ising Hamiltonian (18) on a 20×20 square lattice, at a near-critical magnetic field strength of $B = 3.0$ and exactly encoded in a MPO with bond dimension $D' = 22$. The plot shows how the converged DMRG energy per site appears to saturate to a constant as we increase the bond dimension D, in clear analogy with Fig. 2 for the spinless fermion model, where such saturation was to the correct value of the ground-state energy. While this model cannot be solved analytically, results from numerical studies using, e.g., Monte Carlo or tensor network methods [124] are available, albeit not at the exact same system size. At bond dimension $D = 2^{15} = 32\,768$ our simulations already reached the maximum level of achievable accuracy within single-precision arithmetic.

C. Scaling of runtimes

Finally, Fig. 4 shows measured and estimated runtimes, for fixed MPO bond dimension $D' = 22$ and as a function of the MPS bond dimension D, for the update of a single MPS tensor. These times include the time to perform the Lanczos tridiagonalization, the orthogonalization of the optimized tensor, and the update of one left or right block Hamiltonian.

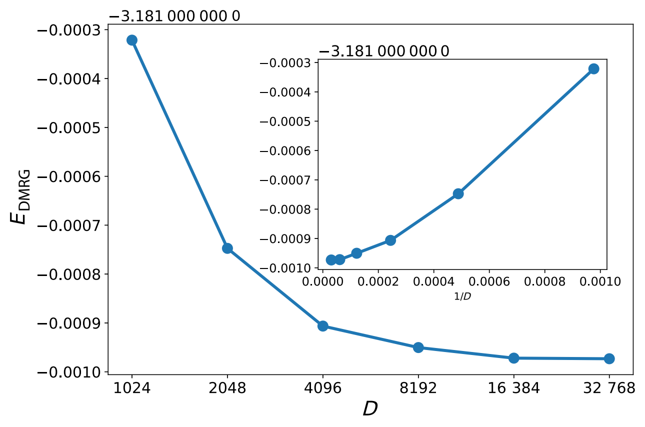

FIG. 3. Convergence of the DMRG ground-state energy per site as a function of bond dimension D for the critical transverse field Ising model on a 20×20 lattice, leading to an energy density of $-3.181\,97(2)$. For comparison, Lubasch *et al.* [124] obtained the values $-3.172\,10(1)$ and $-3.182\,43(1)$ for the energy density using a PEPS simulation on 11×11 and 21×21 lattices, respectively.

At fixed TPU configuration (fixed color in the plot), the runtimes are seen to scale with the MPS bond dimension D as D^3. On the other hand, if every time that we double the bond dimension we quadruple the number of TPU cores (see the data points for $D = 2^{15}, 2^{16}, 2^{17}$, and 2^{18} for 32, 128, 512, and 2048 cores, respectively), then the runtimes grow approximately only linearly in D. The type of scaling is key to reaching unprecedentedly large bond dimensions with affordable run times.

Figures 5(a) and 5(b) show strong and weak scaling behavior, respectively, for a DMRG-style matrix-vector operation (see Fig. 1, right panel). In the strong scaling case for $D = 16\,384$ we observe a runtime reduction by a

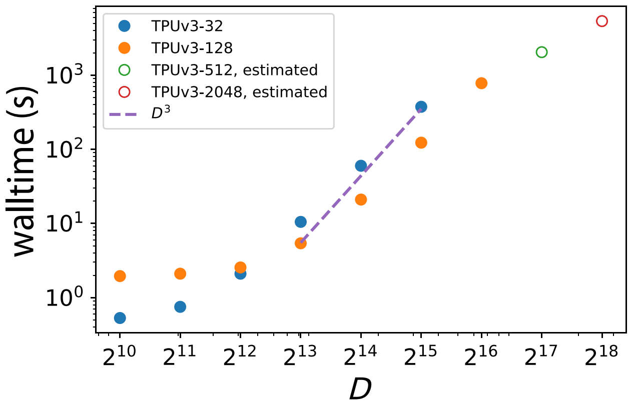

FIG. 4. Measured and estimated runtimes per DMRG optimization step (including the Lanczos update, tensor orthogonalization, and update of the effective environment Hamiltonian) on different TPU cluster sizes. The solid line is proportional to the expected D^3 scaling of the DMRG algorithm.

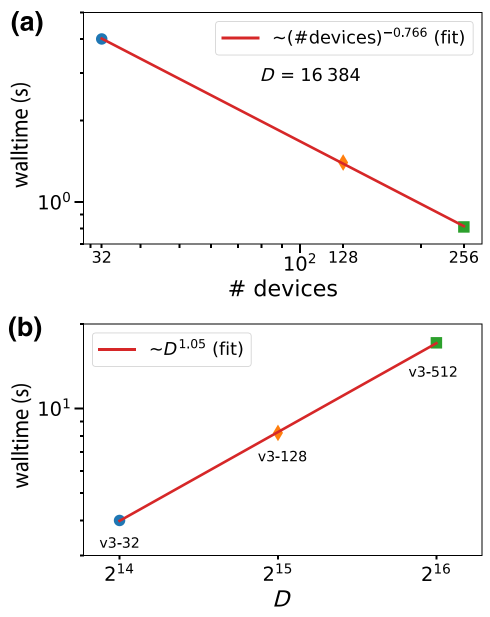

FIG. 5. (a) Strong scaling analysis for a DMRG-style matrix-vector operation for a bond dimension of $D = 16384$ on an increasingly larger number of TPU cores. Doubling the number of devices we observe a roughly ($2^{0.766} \approx 1.7$)-fold reduction in runtime. For ideal strong scaling, the slope of the line would be -1, compared to the measured -0.766. (b) Weak scaling analysis of the same DMRG-style matrix vector operation as in (a). Between subsequent data points from left to right, the bond dimension is doubled while the number of devices is quadrupled. Doubling the bond dimension increases runtimes by 2^3, while quadrupling the number of cores should (ideally) reduce runtimes by a factor of 4, resulting in an (ideal) increase of runtimes by a factor of 2. The slope of 1.05 in the figure shows that we essentially observe such ideal scaling (namely slope 1.0) for the largest problem sizes investigated.

factor of $2^{0.766} \approx 1.7$ upon doubling the resources, somewhat short of the ideal factor of 2. For the weak scaling case [shown in Fig. 5(b)] on the other hand, we observe an almost ideal scaling of runtimes (by a factor of 2) upon both doubling the bond dimension and quadrupling the amount of TPU cores.

VI. SUMMARY AND DISCUSSION

We have presented an implementation of the DMRG algorithm on Google's TPUs. Our implementation leverages the distributed accelerated hardware and high-bandwidth memory of a TPU cluster to perform DMRG simulations at unprecedented scale, speed, and accuracy. We benchmarked the implementation on two problems that are notoriously difficult from a DMRG perspective, namely a system of spinless fermions on a 10×10 square lattice known to display a logarithmic correction to the area law, and the (near-)critical transverse field

Ising model on a 20×20 lattice. We performed simulations with bond dimensions of up to $D = 2^{16}$, to the best of our knowledge the largest ever simulated bond dimension so far. We obtained converged results at these bond dimensions in just less than a day. We estimate that such simulations would take months to years on standard shared-memory hardware with up to a few dozen CPU cores, using highly efficient, compiled code. Our results show that compute clusters of hardware accelerators can be leveraged very efficiently for tensor network computations. For our demonstration, we used TPUs, but we would like to emphasize that similar results can be obtained with a cluster of tightly connected graphic processing units (GPUs).

There are several obvious ways to further improve the performance of our implementation of DMRG on TPUs. On the one hand, at an algorithmic level, we already mentioned that one can exploit the internal symmetries of a model [e.g., internal U(1) symmetry corresponding to particle number conservation in the 2D free fermion model (17)]. Incorporating the internal U(1) symmetry into our implementation of DMRG will lead to a substantial reduction of variational parameters and runtimes for the same (effective) bond dimension. At fixed TPU configuration, we expect to then be able to further increase the maximal bond dimension D by perhaps up to about 5–10 times. On the other hand, the largest TPU configuration used in this work was made of 1024 cores, or half a TPU v3 pod. Using a full TPU v3 pod with 2048 cores would result in a roughly 2 times faster computation at fixed bond dimension D. Alternatively, it would allow us to increase the largest D by a factor $\sqrt{2}$. While conducting our simulations, Google announced the fourth generation of TPUs, which are currently available. A TPU v4 pod (with 8192 TPU v4 cores) would allow for an additional 2 times increase of the maximal bond dimension D at comparable runtimes. On the other hand, a superpod of NVIDIA's DGX nodes (with each DGX node containing eight A100 or H100 GPUs) could be utilized in a similar way to reach even larger bond dimensions.

It is worth pointing out that MPS algorithms similar to DMRG also form the basis for more sophisticated tensor networks approaches like, e.g., projected entangled pair states for 2D quantum lattice systems [12], and the availability of fast, large-scale MPS algorithms hence directly impacts not only DMRG but the field of tensor network algorithms as a whole.

In conclusion, the large-scale implementation of DMRG with TPUs presented in this paper may have profound impacts in fields such as condensed matter physics [95, 107], quantum chemistry [16–29], and material science [30–32] to machine learning [42–49], where MPS and tensor network algorithms are either well established or rapidly gaining traction.

ACKNOWLEDGMENTS

The authors would like to thank Örs Legeza and Steve White for useful discussions and comments. The authors also want to thank the JAX team for their continuous support, in particular Skye Wanderman-Milne and Jake Bradbury. This research is supported with Cloud TPUs from Google's TPU Research Cloud (TRC). Sandbox@Alphabet was a team within the Alphabet family of companies, which includes Google, Verily, Waymo, X, and others. G.V. acknowledges support as a CIFAR fellow in the Quantum Information Science Program, a Distinguished Invited Professor at the Institute of Photonic Sciences (ICFO), and a Distinguished Visiting Research Chair at Perimeter Institute. Research at Perimeter Institute is supported by the Government of Canada through the Department of Innovation, Science and Economic Development and by the Province of Ontario through the Ministry of Research, Innovation and Science.

APPENDIX A: TENSOR ORTHOGONALIZATION

A crucial step in the DMRG method is the *orthogonalization* of an optimized MPS tensor M with components $M^i_{\alpha\beta}$. That refers to either one of the following decompositions:

$$[M]^i_{\alpha\beta} = \sum_\gamma [A]^i_{\alpha\gamma}[R^{(l)}]_{\gamma\beta} \quad \text{(left orthogonalization)}, \tag{A1}$$

$$[M]^i_{\alpha\beta} = \sum_\gamma [R^{(r)}]_{\alpha\gamma}[B]^i_{\gamma\beta} \quad \text{(right orthogonalization)}. \tag{A2}$$

Here tensors A and B satisfy the left and right isometric constraints in Eqs. (7) and (8), respectively, which we rewrite here in components as

$$\sum_{i,\gamma} \left([A]^i_{\gamma\alpha}\right)^* [A]^i_{\gamma\beta} = \delta_{\alpha\beta} \quad \text{(left isometry)}, \tag{A3}$$

$$\sum_{i,\gamma} [B]^i_{\alpha\gamma} \left([B]^i_{\beta\gamma}\right)^* = \delta_{\alpha\beta} \quad \text{(right isometry)}. \tag{A4}$$

In the following we focus on the left orthogonalization, Eqs. (A1) and (A3). The right orthogonalization in Eqs. (A2) and (A4) can be obtained similarly. For the rest of this appendix, we regard tensor M as a matrix, defined to have coefficients

$$[M]_{(i\alpha)\beta} = [M]^i_{\alpha\beta}, \tag{A5}$$

where we joined tensor indices i and α into a single matrix index $(i\alpha)$ (note that prior to joining i and α, index i was changed from an upper index to a lower index, which is a trivial change in our setting). This corresponds to a so-called *reshaping* operation, which turns tensor M into a matrix (also denoted M!) according to Eq. (A5). Note that matrix M is in general rectangular. For instance, at constant MPS bond dimension D, it has shape $(qD) \times D$, where q is the dimension of the vector space describing one site of the lattice, so that it has q times more rows than columns. Our goal is to obtain D orthonormal columns, in the sense of Eq. (A3).

Two popular approaches to orthonormalize M are the use of either a QR decomposition or a singular value decomposition (SVD) of M. A TPU distributed version of QR or SVD was not available to us at the time we implemented DMRG. We were able to implement instead a TPU distributed version of the polar decomposition, which requires mostly distributed matrix-matrix multiplications and additions. The polar decomposition of M reexpresses this matrix as a product of an isometric $(qD) \times D$ matrix U and a positive semidefinite Hermitian $D \times D$ matrix H, i.e.,

$$M = U \cdot H. \tag{A6}$$

We can then obtain tensor A and matrix $R^{(l)}$ in Eqs. (A1) and (A3) from U and H simply according to

$$[A]^{i}_{\alpha\beta} = [U]_{(i\alpha)\beta}, \qquad [R^{(l)}]_{\alpha\beta} = [H]_{\alpha\beta}. \tag{A7}$$

The polar decomposition can be obtained by first normalizing M into X_0 so that its largest singular value is upper bounded by 1, namely,

$$X_0 = M/z, \qquad z = ||M|| = \sqrt{\mathrm{tr}(M^{\dagger} \cdot M)}, \tag{A8}$$

and then converging the *Newton-Schultz* iteration:

$$X_{i+1} = X_i \cdot \left(\tfrac{3}{2}\mathbb{1} - \tfrac{1}{2}X_i^{\dagger} \cdot X_i\right). \tag{A9}$$

It is easily verified that each iteration step in Eq. (A9) applies the polynomial $P(x) = \frac{3}{2}x - \frac{1}{2}x^3$ to the (renormalized) singular values of M, while preserving its left and right singular vectors. Iterative application $\lambda_{i+1} = P(\lambda_i)$ maps initial real numbers λ_0 in the interval $(0, 1]$ to 1, in the limit $i \to \infty$ (when a singular value x is not too small, the iteration will turn it into 1 *quadratically*, that is, with a deviation from 1 that is suppressed quadratically in the number of iterations). Equation (A9) hence converges to the polar factor U of X_0 (and thus of M) [129]. We can then also obtain H from M and U simply using $H = U^{\dagger} \cdot M$.

It is instructive to relate the polar decomposition to the SVD of M, given by $M = W \cdot S \cdot V$, where S is a diagonal matrix with the singular values and W and V are

```
function POLAR_FACTOR($M$)
    $z = \|M\|$
    $M \leftarrow M/z$
    $q = M$.shape[0]
    converged = False
    while not converged do
        $T$= zeros_like($M[0,\ldots]$)
        for $i$=0...$q-1$ do
            $T$+=SUMMA($M[i,\ldots], M[i,\ldots]$,
                        herm_A=True,                    ▷ Hermitian transpose of $A$
                        herm_B=False)
        end for
        for $i = 0 \ldots q-1$ do
            $M[i,\ldots] \leftarrow \frac{3}{2}M[i,\ldots] - \frac{1}{2}$SUMMA($M[i,\ldots], T$)
        end for
        converged = CHECK_UNITARITY($M$)
    end while
    return $M$
end function
```

Algorithm 2. Newton-Schultz iteration for the polar factor of M

unitary (or isometric) matrices. The unitary (or isometric) and Hermitian factors U and H then read

$$U = W \cdot V, \qquad H = V^{\dagger} \cdot S \cdot V, \tag{A10}$$

where we have used the fact that

$$M = W \cdot S \cdot V = (W \cdot V) \cdot (V^{\dagger} \cdot S \cdot V) = U \cdot H.$$

We note that the case of singular values that are identically zero can be approximately addressed by adding to M a diagonal constant perturbation of magnitude equal to machine precision ϵ.

Iteration (A9) requires only matrix multiplications, transpositions, matrix addition, and complex conjugation as fundamental operations. In the distributed setting, we implement the first two using the well-known SUMMA algorithm [118] for distributed matrix multiplications (SUMMA can also handle the case of multiplication of transposed matrices). Matrix addition and matrix complex conjugation of distributed matrices is trivial in that it can be carried out locally on each core.

APPENDIX B: TENSOR TRUNCATION

Another operation of central importance in some implementations of DMRG (and, more generally, in many other tensor network algorithms) is *truncation* of a matrix, namely rank reduction by retaining only its largest singular values. In DMRG, this is needed in the context of a two-site update. The DMRG implementation described in this paper corresponds to a one-site update and does not require tensor truncations, but here we explain how to implement them for completeness.

Consider a matrix M and its SVD

$$M = W \cdot S \cdot V, \tag{B1}$$

where W and V are unitary (or isometric) matrices and S is a diagonal matrix with the singular values s_{α} of M in its diagonal, that is, $[S]_{\alpha\alpha} = s_\alpha$, organized in decreasing order $s_1 \geq s_2 \geq \cdots \geq s_m \geq 0$. In components, the SVD reads

$$[M]_{\alpha\beta} = \sum_{\gamma=1}^{m} [W]_{\alpha\gamma} s_\gamma [V]_{\gamma\beta}. \tag{B2}$$

Before proceeding further, we note here that in tensor network algorithms, matrix M to be truncated will often result from reshaping a higher-order tensor (e.g., an order-4 tensor assigned to two adjacent sites of a lattice in a two-site DMRG update). However, the specific origin of M is not important in our discussion below.

A δ-truncated singular value decomposition of M corresponds to the SVD of another matrix $\tilde{M}$ obtained by keeping only the singular values s_α of M that are larger than δ. Suppose that there are m' (with $m' \leq m$) such singular values. Then $\tilde{M}$ is defined through

$$[\tilde{M}]_{\alpha\beta} = \sum_{\gamma=1}^{m'} [\tilde{W}]_{\alpha\gamma} s_\gamma [\tilde{V}]_{\gamma\beta}, \tag{B3}$$

where $\tilde{W}$ and $\tilde{V}$ are obtained from W and V by keeping only their first m' columns and rows, respectively. That is,

$$[\tilde{W}]_{\alpha\gamma} = [W]_{\alpha\gamma} \quad \text{for all } \alpha,\ 1 \leq \gamma \leq m', \tag{B4}$$

$$[\tilde{V}]_{\gamma\beta} = [V]_{\gamma\beta} \quad \text{for all } \beta,\ 1 \leq \gamma \leq m'. \tag{B5}$$

We can similarly define a truncated singular value matrix $\tilde{S}$, of size $m' \times m'$, as the diagonal matrix that contains the m' singular values s_α organized in decreasing order, such that

$$[\tilde{S}]_{\gamma\gamma} = [S]_{\gamma\gamma} = s_\gamma, \qquad 1 \leq \gamma \leq m'. \tag{B6}$$

Matrix $\tilde{M} = \tilde{W} \cdot \tilde{S} \cdot \tilde{V}$ can then be seen to be the rank-m' best approximation to M, in that the difference matrix $\Delta \equiv M - \tilde{M}$ has the smallest possible norm $||\Delta|| = \sqrt{\mathrm{tr}(\Delta \cdot \Delta^\dagger)}$.

Our goal is to produce two matrices F and G, with m' columns and rows, respectively, such that their product equates to $\tilde{M}$, that is,

$$\tilde{M} = F \cdot G. \tag{B7}$$

In a tensor network algorithm, the pair F, G corresponds to adjacent tensors where the bond index connecting them has been truncated (e.g., two adjacent MPS tensors during a two-site update in DMRG). An obvious way to obtain F and G is from the SVD of M, by choosing, e.g., $F = \tilde{W}$ and $G = \tilde{S} \cdot \tilde{V}$. However, here we are interested in obtaining F and G without resorting to a SVD of matrix M.

Remarkably, the above task can be achieved with the polar decomposition that, as described in the previous appendix, can be implemented using a small set of simple matrix operations: matrix-matrix multiplications and additions, as well as matrix transposition and complex conjugation. Next we describe how.

As a first step, we use the polar decomposition to obtain the isometric and positive semidefinite factors U and H of matrix M in Eq. (A6). By construction, H has the singular values s_α of M as its eigenvalues. As a second step, we compute the polar decomposition of $H - \mathbb{1}\delta$. Let U' and H' be the resulting unitary and positive semidefinite factors,

$$H - \mathbb{1}\delta = U' \cdot H'. \tag{B8}$$

In general, the polar decomposition $Z = X \cdot |Z|$ of a Hermitian matrix Z with (real) eigenvalues z_α is given in terms of a unitary matrix X and a positive semidefinite matrix $|Z|$ with very simple structure: both X and $|Z|$ have the same eigenvectors as Z; moreover, for the αth common eigenvector, $|Z|$ has as eigenvalue the absolute value $|z_\alpha|$ of the corresponding eigenvalues z_α of Z, whereas X has as eigenvalue $\sigma_\alpha = \pm 1$, where the sign is such that $z_\alpha = \sigma_\alpha |z_\alpha|$. In other words, the unitary factor U' in Eq. (B8) must have m' eigenvalues $+1$ (for the m' eigenvectors of $H - \mathbb{1}\delta$ with positive eigenvalues $s_\alpha - \delta > 0$) and the rest of the eigenvalues must be -1 (for the eigenvectors of $H - \mathbb{1}\delta$ with negative eigenvalues $s_\alpha - \delta < 0$). In particular, we can use U' to define two projectors P_+ and P_- onto the positive and negative subspaces of $H - \mathbb{1}\delta$ (equivalently, the subspaces of H with $s_\alpha > \delta$ and with $s_\alpha < \delta$) by

$$P_\pm = \frac{\mathbb{1} \pm U'}{2}, \tag{B9}$$

and use them in turn to define projections $H_{>\delta}$ and $H_{<\delta}$ of matrix H onto its $s_\alpha > \delta$ subspace and $s_\alpha < \delta$ subspace,

$$H_{>\delta} \equiv P_+ \cdot H \cdot P_+, \qquad H_{<\delta} \equiv P_- \cdot H \cdot P_-, \tag{B10}$$

such that $H = H_{>\delta} + H_{<\delta}$. We can thus write

$$M = U \cdot H = U \cdot H_{>\delta} + U \cdot H_{<\delta}, \tag{B11}$$

where the first term $U \cdot H_{>\delta} = U \cdot P_+ \cdot H$ corresponds to the largest m' singular values s_α of M. In other words, we have obtained the best rank-m' approximation $\tilde{M}$ to M:

$$\tilde{M} = U \cdot P_+ \cdot H. \tag{B12}$$

However, we have not yet reduced the number of columns of U. For that purpose, we must find an isometry C_+ with

```
1: function SUBSPACE($P$)
2:     $m' = \mathrm{Tr}P$
3:     $C = \mathrm{RANDOM}\ (m, m')$          ▷ Initial guess of shape $m \times m'$
4:     $X = P@C$          ▷ Use, e.g., SUMMA in the distributed setting
5:     $C = \mathrm{POLAR\_FACTOR}\ (X)$          ▷ Polar decomposition instead of a QR decomposition
6:     return $C$
7: end function
```

Algorithm 3. Subspace iteration for an $m \times m$ projector matrix P with rank m'

m' columns such that

$$P_+ = C_+ C_+^\dagger. \tag{B13}$$

That is, we need to find an orthonormal basis for the m'-dimensional column space of the rank-m' projector P_+. We achieve this using a slight modification of the standard, QR-based subspace-iteration method to avoid the use of a QR decomposition (see Appendix C). Then we have

$$\tilde{M} = U \cdot P_+ \cdot H = U \cdot C_+ \cdot C_+^\dagger \cdot H = \tilde{U} \cdot \tilde{H}, \tag{B14}$$

where matrices

$$\tilde{U} \equiv U \cdot C_+, \qquad \tilde{H} \equiv C_+^\dagger \cdot H, \tag{B15}$$

have m' columns and rows, respectively, and therefore qualify as matrices F and G in the truncated decomposition (B7).

A similar approach based on the McWeeny iteration can be used to truncate to a fixed number of singular values, instead of truncating singular values below a certain threshold δ [130].

APPENDIX C: SUBSPACE ITERATION

Consider a $m \times m$ Hermitian matrix P that is a rank-m' projector, namely, such that

$$P \cdot P = P, \qquad \mathrm{tr}(P) = m', \tag{C1}$$

where we also assume that P is rank deficient, meaning that $m' < m$. Our goal is to find an isometric matrix C of shape $m \times m'$ such that we can write P as the product

$$P = C \cdot C^\dagger. \tag{C2}$$

For that purpose, we can use Algorithm 3. It is a specialization (for a rank-deficient projector P) of the subspace iteration method that can more generally be used to compute the first n dominant eigenvectors of a matrix. Specifically, we modify the standard subspace iteration method in two ways: (1) since $P^2 = P$, a single iteration is sufficient (so we skip looping over steps 4 and 5); (2) we use a polar decomposition (easier to implement on a distributed TPU setting) instead of the usual QR decomposition.

[1] S. R. White, Density Matrix Formulation for Quantum Renormalization Groups, Phys. Rev. Lett. **69**, 2863 (1992).
[2] S. R. White, Density-matrix algorithms for quantum renormalization groups, Phys. Rev. B **48**, 10345 (1993).
[3] K. A. Hallberg, Density-matrix algorithm for the calculation of dynamical properties of low-dimensional systems, Phys. Rev. B **52**, R9827 (1995).
[4] T. D. Kühner and S. R. White, Dynamical correlation functions using the density matrix renormalization group, Phys. Rev. B **60**, 335 (1999).
[5] E. Jeckelmann, Dynamical density-matrix renormalization-group method, Phys. Rev. B **66**, 045114 (2002).
[6] T. Barthel, U. Schollwöck, and S. R. White, Spectral functions in one-dimensional quantum systems at finite temperature using the density matrix renormalization group, Phys. Rev. B **79**, 245101 (2009).
[7] B. Pirvu, J. Haegeman, and F. Verstraete, A matrix product state based algorithm for determining dispersion relations of quantum spin chains with periodic boundary conditions, Phys. Rev. B **85**, 035130 (2012).
[8] J. Haegeman, B. Pirvu, D. J. Weir, J. I. Cirac, T. J. Osborne, H. Verschelde, and F. Verstraete, Variational matrix product ansatz for dispersion relations, Phys. Rev. B **85**, 100408(R) (2012).
[9] T. Nishino, Density matrix renormalization group method for 2D classical models, J. Phys. Soc. Jpn. **64**, 3598 (1995).
[10] T. Nishino and K. Okunishi, Corner transfer matrix renormalization group method, J. Phys. Soc. Jpn. **65**, 891 (1996). publisher: The Physical Society of Japan,.
[11] G. Vidal, Efficient Classical Simulation of Slightly Entangled Quantum Computations, Phys. Rev. Lett. **91**, 147902 (2003).
[12] F. Verstraete and J. I. Cirac, Renormalization algorithms for quantum-many body systems in two and higher dimensions, (2004), ArXiv:cond-mat/0407066.
[13] G. Vidal, Entanglement Renormalization, Phys. Rev. Lett. **99**, 220405 (2007).
[14] G. Vidal, A Class of Quantum Many-Body States That Can Be Efficiently Simulated, Phys. Rev. Lett. **101**, 110501 (2008).
[15] I. P. McCulloch, Infinite size density matrix renormalization group, revisited, (2008), ArXiv:0804.2509.
[16] S. R. White and R. L. Martin, Ab initio quantum chemistry using the density matrix renormalization group, J. Chem. Phys. **110**, 4127 (1999).
[17] G. K.-L. Chan and S. Sharma, The density matrix renormalization group in quantum chemistry, Annu. Rev. Phys. Chem. **62**, 465 (2011).
[18] S. Wouters and D. Van Neck, The density matrix renormalization group for ab initio quantum chemistry, Eur. Phys. J. D **68**, 272 (2014).

[19] S. Szalay, M. Pfeffer, V. Murg, G. Barcza, F. Verstraete, R. Schneider, and O. Legeza, Tensor product methods and entanglement optimization for ab initio quantum chemistry, Int. J. Quantum Chem. **115**, 1342 (2015).

[20] M. Reiher, DMRG in quantum chemistry: from its relation to traditional methods to n-orbital density matrices and beyond, Online presentation available at https://confs.physics.ox.ac.uk/pauli2016/files/slides/Reiher.pdf.

[21] R. Olivares-Amaya, W. Hu, N. Nakatani, S. Sharma, J. Yang, and G. K.-L. Chan, The ab-initio density matrix renormalization group in practice, J. Chem. Phys. **142**, 034102 (2015).

[22] T. Yanai, Y. Kurashige, W. Mizukami, J. Chalupský, T. N. Lan, and M. Saitow, Density matrix renormalization group for ab initio calculations and associated dynamic correlation methods: A review of theory and applications, Int. J. Quantum Chem. **115**, 283 (2015).

[23] G. K.-L. Chan, A. Keselman, N. Nakatani, Z. Li, and S. R. White, Matrix product operators, matrix product states, and ab initio density matrix renormalization group algorithms, (2016), ArXiv:1605.02611.

[24] S. R. White and E. M. Stoudenmire, Multi-sliced Gausslet basis sets for electronic structure, Phys. Rev. B **99**, 081110(R) (2019).

[25] A. Baiardi and M. Reiher, The density matrix renormalization group in chemistry and molecular physics: Recent developments and new challenges, J. Chem. Phys. **152**, 040903 (2020).

[26] J. Brabec, J. Brandejs, K. Kowalski, S. Xantheas, O. Legeza, and L. Veis, Massively parallel quantum chemical density matrix renormalization group method, (2020), ArXiv:2001.04890.

[27] G. Barcza, V. Ivády, T. Szilvási, M. Vörös, L. Veis, A. Gali, and O. Legeza, DMRG on top of plane-wave Kohn-Sham orbitals: Case study of defected boron nitride, (2020), ArXiv:2006.04557.

[28] Y. Qiu and S. R. White, Hybrid Gausslet/Gaussian basis sets, (2021), ArXiv:2103.02734.

[29] J. J. Goings, A. White, J. Lee, C. S. Tautermann, M. Degroote, C. Gidney, T. Shiozaki, R. Babbush, and N. C. Rubin, Reliably assessing the electronic structure of cytochrome P450 on today's classical computers and tomorrow's quantum computers, (2022), ArXiv:2202.01244.

[30] M. Ganahl, M. Aichhorn, H. G. Evertz, P. Thunström, K. Held, and F. Verstraete, Efficient DMFT impurity solver using real-time dynamics with matrix product states, Phys. Rev. B **92**, 155132 (2015).

[31] M. Ganahl, P. Thunström, F. Verstraete, K. Held, and H. G. Evertz, Chebyshev expansion for impurity models using matrix product states, Phys. Rev. B **90**, 045144 (2014).

[32] D. Bauernfeind, M. Zingl, R. Triebl, M. Aichhorn, and H. G. Evertz, Fork Tensor-Product States: Efficient Multiorbital Real-Time DMFT Solver, Phys. Rev. X **7**, 031013 (2017).

[33] M. B. Hastings, An area law for one-dimensional quantum systems, J. Stat. Mech.: Theory Exp. **2007**, P08024 (2007).

[34] 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).

[35] C. Huang, F. Zhang, M. Newman, X. Ni, D. Ding, J. Cai, X. Gao, T. Wang, F. Wu, G. Zhang, H.-S. Ku, Z. Tian, J. Wu, H. Xu, H. Yu, B. Yuan, M. Szegedy, Y. Shi, H.-H. Zhao, C. Deng, and J. Chen, Efficient parallelization of tensor network contraction for simulating quantum computation, Nat. Comput. Sci. **1**, 578 (2021).

[36] S. Bravyi, M. Suchara, and A. Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[37] A. J. Ferris and D. Poulin, Tensor Networks and Quantum Error Correction, Phys. Rev. Lett. **113**, 030501 (2014).

[38] C. T. Chubb and S. T. Flammia, Statistical mechanical models for quantum codes with correlated noise, Ann. de l'Institut Henri Poincaré D **8**, 269 (2021).

[39] A. S. Darmawan and D. Poulin, Linear-time general decoding algorithm for the surface code, Phys. Rev. E **97**, 051302(R) (2018).

[40] I. L. Markov and Y. Shi, Simulating quantum computation by contracting tensor networks, SIAM J. Comput. **38**, 963 (2008).

[41] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven, Simulation of low-depth quantum circuits as complex undirected graphical models, (2018), ArXiv:1712.05384.

[42] X. Gao, Z.-Y. Zhang, and L.-M. Duan, A quantum machine learning algorithm based on generative models, Sci. Adv. **4**, eaat9004 (2018).

[43] S. Cheng, L. Wang, T. Xiang, and P. Zhang, Tree tensor networks for generative modeling, Phys. Rev. B **99**, 155131 (2019).

[44] E. M. Stoudenmire and D. J. Schwab, Supervised learning with quantum-inspired tensor networks, (2017), ArXiv:1605.05775.

[45] E. M. Stoudenmire, Learning relevant features of data with multi-scale tensor networks, Quantum Sci. Technol. **3**, 034003 (2018).

[46] I. Glasser, N. Pancotti, and J. I. Cirac, From probabilistic graphical models to generalized tensor networks for supervised learning, (2019), ArXiv:1806.05964.

[47] T. Vieijra, L. Vanderstraeten, and F. Verstraete, Generative modeling with projected entangled-pair states, (2022), ArXiv:2202.08177.

[48] C. Roberts, A. Milsted, M. Ganahl, A. Zalcman, B. Fontaine, Y. Zou, J. Hidary, G. Vidal, and S. Leichenauer, TensorNetwork: A library for physics and machine learning, (2019), ArXiv:1905.01330.

[49] Z.-Y. Han, J. Wang, H. Fan, L. Wang, and P. Zhang, Unsupervised Generative Modeling Using Matrix Product States, Phys. Rev. X **8**, 031012 (2018).

[50] M. Lubasch, P. Moinier, and D. Jaksch, Multigrid renormalization, J. Comput. Phys. **372**, 587 (2018).

[51] N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner, and D. Jaksch, A quantum-inspired approach to exploit turbulence structures, Nat. Comput. Sci. **2**, 30 (2022).

[52] M. Fannes, B. Nachtergaele, and R. F. Werner, Finitely correlated states on quantum spin chains, Commun. Math. Phys. **144**, 443 (1992).

[53] 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).

[54] S. Östlund and S. Rommer, Thermodynamic Limit of Density Matrix Renormalization, Phys. Rev. Lett. **75**, 3537 (1995).

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

[56] J. Dukelsky, M. A. Martín-Delgado, T. Nishino, and G. Sierra, Equivalence of the variational matrix product method and the density matrix renormalization group applied to spin chains, Europhys. Lett. **43**, 457 (1998). publisher: IOP Publishing,.

[57] H. Zhai and G. K.-L. Chan, Low communication high performance ab initio density matrix renormalization group algorithms, J. Chem. Phys. **154**, 224116 (2021).

[58] Y. Motoyama, T. Okubo, K. Yoshimi, S. Morita, T. Kato, and N. Kawashima, TeNeS: Tensor network solver for quantum lattice systems, (2021), ArXiv:2112.13184.

[59] R. Levy, E. Solomonik, and B. K. Clark, Distributed-memory DMRG via sparse and dense parallel tensor contractions, SC20: Int. Conf. High Perform. Comput., Netw., Storage Anal., 1 (2020).

[60] H. Ueda, K. Okunishi, S. Yunoki, and T. Nishino, Corner transfer matrix renormalization group analysis of the two-dimensional dodecahedron model, Phys. Rev. E **102**, 032130 (2020).

[61] C. Nemes, G. Barcza, Z. Nagy, O. Legeza, and P. Szolgay, The density matrix renormalization group algorithm on kilo-processor architectures: Implementation and trade-offs, Comput. Phys. Commun. **185**, 1570 (2014).

[62] P. Secular, N. Gourianov, M. Lubasch, S. Dolgov, S. R. Clark, and D. Jaksch, Parallel time-dependent variational principle algorithm for matrix product states, Phys. Rev. B **101**, 235123 (2020).

[63] N. Jouppi, D. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson, A domain-specific supercomputer for training deep neural networks, Commun. ACM **63**, 67 (2020).

[64] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, and A. Borchers, *et al.*, in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ISCA '17 (Association for Computing Machinery, New York, NY, USA, 2017), p. 1.

[65] A. Morningstar, M. Hauru, J. Beall, M. Ganahl, A. G. M. Lewis, V. Khemani, and G. Vidal, Simulation of Quantum Many-Body Dynamics with Tensor Processing Units: Floquet Prethermalization, PRX Quantum **3**, 020331 (2022).

[66] M. Hauru, A. Morningstar, J. Beall, M. Ganahl, A. Lewis, and G. Vidal, Simulation of quantum physics with tensor processing units: Brute-force computation of ground states and time evolution (2021), ArXiv:2111.10466.

[67] A. G. M. Lewis, J. Beall, M. Ganahl, M. Hauru, S. B. Mallick, and G. Vidal, Large-scale distributed linear algebra with tensor processing units, PNAS **119**, e2122762119 (2022).

[68] R. Pederson, J. Kozlowski, R. Song, J. Beall, M. Ganahl, M. Hauru, A. G. M. Lewis, Y. Yao, S. B. Mallick, V. Blum, and G. Vidal, Large scale quantum chemistry with tensor processing units, J. Chem. Theory Comput. **19**, 25 (2023).

[69] R. Shillito, A. Petrescu, J. Cohen, J. Beall, M. Hauru, M. Ganahl, A. G. M. Lewis, G. Vidal, and A. Blais, Dynamics of transmon ionization (2022), ArXiv:2203.11235.

[70] E. Gustafson, B. Holzman, J. Kowalkowski, H. Lamm, A. C. Y. Li, G. Perdue, S. Boixo, S. Isakov, O. Martin, and R. Thomson, *et al.*, Large scale multi-node simulations of $\mathbb{Z}_2$ gauge theory quantum circuits using Google Cloud platform (2021), ArXiv:2110.07482.

[71] Martin Ganahl, *et al.*, Tensor processing units for simulating quantum circuits, *in preparation*.

[72] Ruyi Song, *et al.*, Accelerated quantum chemistry calculations with tensor processing units: from biology to materials science, *in preparation*.

[73] John Kozlowski, *et al.*, Full protein density functional theory with tensor processing units, *in preparation*.

[74] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, Matrix product states and projected entangled pair states: Concepts, symmetries, theorems, Rev. Mod. Phys. **93**, 045003 (2021).

[75] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, Ann. Phys. (N. Y) **326**, 96 (2011).

[76] I. P. McCulloch, From density-matrix renormalization group to matrix product states, J. Stat. Mech.: Theory Exp. **2007**, P10014 (2007).

[77] M. Zwolak and G. Vidal, Mixed-State Dynamics in One-Dimensional Quantum Lattice Systems: A Time-Dependent Superoperator Renormalization Algorithm, Phys. Rev. Lett. **93**, 207205 (2004).

[78] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac, Matrix Product Density Operators: Simulation of Finite-Temperature and Dissipative Systems, Phys. Rev. Lett. **93**, 207204 (2004).

[79] B. Pirvu, V. Murg, J. I. Cirac, and F. Verstraete, Matrix product operator representations, New J. Phys. **12**, 025012 (2010).

[80] L. Michel and I. P. McCulloch, Schur forms of matrix product operators in the infinite limit, (2010), ArXiv:1008.4667.

[81] C. Hubig, I. P. McCulloch, and U. Schollwöck, Generic construction of efficient matrix product operators, Phys. Rev. B **95**, 035129 (2017).

[82] G. H. Golub and C. Reinsch, Singular value decomposition and least squares solutions, Numer. Math. **14**, 403 (1970).

[83] G. Vidal, J. I. Latorre, E. Rico, and A. Kitaev, Entanglement in Quantum Critical Phenomena, Phys. Rev. Lett. **90**, 227902 (2003).

[84] I. Peschel, Calculation of reduced density matrices from correlation functions, J. Phys. A: Math. General **36**, (2003).

[85] J. I. Latorre, E. Rico, and G. Vidal, Ground state entanglement in quantum spin chains, Quantum Inf. Comput. **4**, 48 (2004).
[86] L. Bombelli, R. K. Koul, J. Lee, and R. D. Sorkin, Quantum source of entropy for black holes, Phys. Rev. D **34**, 373 (1986).
[87] M. Srednicki, Entropy and Area, Phys. Rev. Lett. **71**, 666 (1993).
[88] L. Amico, R. Fazio, A. Osterloh, and V. Vedral, Entanglement in many-body systems, Rev. Mod. Phys. **80**, 517 (2008).
[89] M. M. Wolf, Violation of the Entropic Area Law for Fermions, Phys. Rev. Lett. **96**, 010404 (2006).
[90] D. Gioev and I. Klich, Entanglement Entropy of Fermions in Any Dimension and the Widom Conjecture, Phys. Rev. Lett. **96**, 100503 (2006). publisher: American Physical Society,.
[91] I. Peschel and V. Eisler, Reduced density matrices and entanglement entropy in free lattice models, J. Phys. A: Math. Theor. **42**, 504003 (2009). publisher: IOP Publishing,.
[92] J. Eisert, M. Cramer, and M. B. Plenio, Colloquium: Area laws for the entanglement entropy, Rev. Mod. Phys. **82**, 277 (2010).
[93] Y.-F. Jiang, J. Zaanen, T. P. Devereaux, and H.-C. Jiang, Ground state phase diagram of the doped Hubbard model on the four-leg cylinder, Phys. Rev. Res. **2**, 033073 (2020). publisher: American Physical Society,.
[94] H.-C. Jiang and T. P. Devereaux, Superconductivity in the doped Hubbard model and its interplay with next-nearest hopping t', Science **365**, 1424 (2019). publisher: American Association for the Advancement of Science,.
[95] S. Yan, D. A. Huse, and S. R. White, Spin-liquid ground state of the S = 1/2 kagome Heisenberg antiferromagnet, Science **332**, 1173 (2011).
[96] E. Stoudenmire and S. R. White, Studying two-dimensional systems with the density matrix renormalization group, Annu. Rev. Condens. Matter Phys. **3**, 111 (2012).
[97] S. R. White and D. J. Scalapino, Density Matrix Renormalization Group Study of the Striped Phase in the 2D t-J Model, Phys. Rev. Lett. **80**, 1272 (1998).
[98] S. R. White and D. J. Scalapino, Hole and pair structures in the t-J model, Phys. Rev. B **55**, 6504 (1997).
[99] S. R. White and D. J. Scalapino, Energetics of Domain Walls in the 2D t-J Model, Phys. Rev. Lett. **81**, 3227 (1998).
[100] T. Tohyama, C. Gazza, C. T. Shih, Y. C. Chen, T. K. Lee, S. Maekawa, and E. Dagotto, Stripe stability in the extended t-J model on planes and four-leg ladders, Phys. Rev. B **59**, R11649(R) (1999).
[101] S. R. White and D. J. Scalapino, Competition between stripes and pairing in a t-J model, Phys. Rev. B **60**, R753 (1999).
[102] A. P. Kampf, D. J. Scalapino, and S. R. White, Stripe orientation in an anisotropic t-J model, Phys. Rev. B **64**, 052509 (2001). publisher: American Physical Society.
[103] S. R. White and D. J. Scalapino, Phase separation and stripe formation in the two-dimensional t-J model: A comparison of numerical results, Phys. Rev. B **61**, 6320 (2000).
[104] M. Q. Weng, D. N. Sheng, Z. Y. Weng, and R. J. Bursill, Spin-liquid phase in an anisotropic triangular-lattice Heisenberg model: Exact diagonalization and density-matrix renormalization group calculations, Phys. Rev. B **74**, 012407 (2006).
[105] L. Cincio and G. Vidal, Characterizing Topological Order by Studying the Ground States of an Infinite Cylinder, Phys. Rev. Lett. **110**, 067208 (2013).
[106] 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).
[107] Simons Collaboration on the Many-Electron Problem, J. P. F. LeBlanc, A. E. Antipov, F. Becca, I. W. Bulik, G. K.-L. Chan, C.-M. Chung, Y. Deng, M. Ferrero, T. M. Henderson, C. A. Jiménez-Hoyos, E. Kozik, X.-W. Liu, A. J. Millis, N. V. Prokofev, M. Qin, G. E. Scuseria, H. Shi, B. V. Svistunov, L. F. Tocchio, I. S. Tupitsyn, S. R. White, S. Zhang, B.-X. Zheng, Z. Zhu, and E. Gull: Solutions of the Two-Dimensional Hubbard Model: Benchmarks and Results from a Wide Range of Numerical Algorithms, Physical Review X **5**, 0410412015.
[108] G. K.-L. Chan and M. Head-Gordon, Highly correlated calculations with a polynomial cost algorithm: A study of the density matrix renormalization group, J. Chem. Phys. **116**, 4462 (2002).
[109] G. K.-L. Chan and M. Head-Gordon, Exact solution (within a triple-zeta, double polarization basis set) of the electronic Schrödinger equation for water, J. Chem. Phys. **118**, 8551 (2003).
[110] O. Legeza, J. Röder, and B. A. Hess, QC-DMRG study of the ionic-neutral curve crossing of LiF, Mol. Phys. **101**, 2019 (2003).
[111] O. Legeza, J. Röder, and B. A. Hess, Controlling the accuracy of the density-matrix renormalization-group method: The dynamical block state selection approach, Phys. Rev. B **67**, 125114 (2003).
[112] G. K.-L. Chan, M. Kállay, and J. Gauss, State-of-the-art density matrix renormalization group and coupled cluster theory studies of the nitrogen binding curve, J. Chem. Phys. **121**, 6110 (2004).
[113] G. Moritz, B. A. Hess, and M. Reiher, Convergence behavior of the density-matrix renormalization group algorithm for optimized orbital orderings, J. Chem. Phys. **122**, 024107 (2005).
[114] G. Moritz, A. Wolf, and M. Reiher, Relativistic DMRG calculations on the curve crossing of cesium hydride, J. Chem. Phys. **123**, 184105 (2005).
[115] O. Weser, L. Freitag, K. Guther, A. Alavi, and G. L. Manni, Chemical insights into the electronic structure of Fe(II) porphyrin using FCIQMC, DMRG, and generalized active spaces, Int. J. Quantum. Chem. **121**, e26454 (2021).
[116] https://tensorflow.org/xla, accessed: 2021-10-01.
[117] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, JAX: Composable transformations of Python+NumPy programs, (2018).

[118] R. A. Van De Geijn and J. Watts, SUMMA: Scalable universal matrix multiplication algorithm, Concurr.: Pract. Experience **9**, 255 (1997).

[119] S. Inglis and R. G. Melko, Entanglement at a two-dimensional quantum critical point: a T=0 projector quantum Monte Carlo study, New J. Phys. **15**, 073048 (2013).

[120] A. B. Kallin, K. Hyatt, R. R. P. Singh, and R. G. Melko, Entanglement at a Two-Dimensional Quantum Critical Point: A Numerical Linked-Cluster Expansion Study, Phys. Rev. Lett. **110**, 135702 (2013).

[121] A. Milsted, M. Ganahl, S. Leichenauer, J. Hidary, and G. Vidal, TensorNetwork on TensorFlow: A spin chain application using tree tensor networks, (2019), ArXiv:1905.01331.

[122] L. Cincio, J. Dziarmaga, and M. M. Rams, Multiscale Entanglement Renormalization Ansatz in Two Dimensions: Quantum Ising Model, Phys. Rev. Lett. **100**, 240603 (2008).

[123] G. Evenbly and G. Vidal, Entanglement Renormalization in Two Spatial Dimensions, Phys. Rev. Lett. **102**, 180406 (2009).

[124] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, Algorithms for finite projected entangled pair states, Phys. Rev. B **90**, 064425 (2014).

[125] S. Singh, H.-Q. Zhou, and G. Vidal, Simulation of one-dimensional quantum systems with a global SU(2) symmetry, New J. Phys. **12**, (2010).

[126] S. Singh, R. N. C. Pfeifer, and G. Vidal, Tensor network decompositions in the presence of a global symmetry, Phys. Rev. A **82**, 050301(R) (2010).

[127] S. Singh and G. Vidal, Tensor network states and algorithms in the presence of a global SU(2) symmetry, Phys. Rev. B **86**, 195114 (2012).

[128] P. Schmoll, S. Singh, M. Rizzi, and R. Orús, A programming guide for tensor networks with global SU(2) symmetry, Ann. Phys. (N. Y) **419**, 168232 (2020).

[129] Y. Nakatsukasa and N. J. Higham, Backward stability of iterations for computing the polar decomposition, SIAM J. Matrix Anal. Appl. **33**, 460 (2012).

[130] A. H. R. Palser and D. E. Manolopoulos, Canonical purification of the density matrix in electronic-structure theory, Phys. Rev. B **58**, 12704 (1998).