

# Density-Matrix Renormalization Group Algorithm for Simulating Quantum Circuits with a Finite Fidelity


Thomas Ayrál<sup>1</sup>,<sup>2</sup> Thibaud Louvet,<sup>2</sup> Yiqing Zhou<sup>3</sup>,<sup>2</sup> Cyprien Lambert,<sup>1</sup> E. Miles Stoudenmire,<sup>4</sup> and Xavier Waintal<sup>2,\*</sup>

<sup>1</sup>Atos Quantum Laboratory, Les Clayes-sous-Bois, France

<sup>2</sup>PHELIQS, Université Grenoble Alpes, CEA, Grenoble INP, IRIG, Grenoble 38000, France

<sup>3</sup>Department of Physics, Cornell University, Ithaca, New York 14853, USA

<sup>4</sup>Center for Computational Quantum Physics, Flatiron Institute, New York, New York 10010, USA

 (Received 12 July 2022; revised 15 December 2022; accepted 13 February 2023; published 10 April 2023)

We develop a density-matrix renormalization group (DMRG) algorithm for the simulation of quantum circuits. This algorithm can be seen as the extension of the time-dependent DMRG from the usual situation of Hermitian Hamiltonian matrices to quantum circuits defined by unitary matrices. For small circuit depths, the technique is exact and equivalent to other matrix product state–based techniques. For larger depths, it becomes approximate in exchange for an exponential speed up in computational time. Like an actual quantum computer, the quality of the DMRG results is characterized by a finite fidelity. However, unlike a quantum computer, the fidelity depends strongly on the quantum circuit considered. For the most difficult possible circuit for this technique, the so-called “quantum supremacy” benchmark of Google LLC [Arute *et al.*, Nature 574, 505 (2019)], we find that the DMRG algorithm can generate bit strings of the same quality as the seminal Google experiment on a single computing core. For a more structured circuit used for combinatorial optimization (quantum approximate optimization algorithm), we find a drastic improvement of the DMRG results with error rates dropping by a factor of 100 compared with random quantum circuits. Our results suggest that the current bottleneck of quantum computers is their fidelities rather than the number of qubits.

DOI: [10.1103/PRXQuantum.4.020304](https://doi.org/10.1103/PRXQuantum.4.020304)

## I. INTRODUCTION

Quantum computers and the quantum many-body problem are intimately connected. On the one hand, a quantum computer is essentially an instance of the quantum many-body problem on which one has a high level of control. On the other hand, the most promising applications that are foreseen for quantum computers correspond to solving other instances of the quantum many-body problem such as calculating the properties of new materials [1], of new molecules for medicine, or of new catalysts for important chemical reactions [2].

A common misconception of the field of quantum computing is that all quantum many-body problems are exponentially difficult to solve by classical computers because the size of the Hilbert space grows exponentially as  $2^N$

when the system size  $N$  increases. This supposedly dooms many-body simulations on classical computers to failure, and therefore calls for computers with quantum physics inside. This “large Hilbert space fallacy” is however contradicted by the success of classical many-body methods for tackling many of these hard problems. At heart, these methods use the fact that physical problems are *structured*. Physicists take advantage of the separation of time, energy, or length scales of statistical (mean-field) behavior, or of symmetries, to design methods to solve seemingly exponentially hard problems. Even for genuine strongly correlated systems, there exist very powerful many-body techniques that can solve them in a variety of situations, taking advantage of an underlying feature. Without these features—namely, had the physical world been a random point in the Hilbert space—there would indeed be nothing to understand and the problem would be exponentially difficult. However, since the physical world actually *makes sense*, one can argue that simulating a many-body physical problem is not as hopeless as one could naively think.

The subject of this article is to discuss the problem of simulability in the context of quantum computers that use quantum circuits, that is, discrete sequences of quantum

\*xavier.waintal@cea.fr

Published by the American Physical Society under the terms of the [Creative Commons Attribution 4.0 International](https://creativecommons.org/licenses/by/4.0/) license. Further distribution of this work must maintain attribution to the author(s) and the published article’s title, journal citation, and DOI.

gates. This questioning is at the core of the possibility for a “quantum advantage,” for if a quantum computer can be easily simulated, one might as well use the (classical) simulator instead of developing a genuine quantum computer. The quantum circuit model of quantum computing ignores many structures of the underlying many-body problem. For instance, it does not contain the basic concepts of space, time, or energy. Nor does it contain the concept of fermionic or bosonic statistics nor the representation of symmetries (spin, relativity). As a result, it might seem that such a quantum computer must be much harder to simulate than a physics-based many-body problem. An extreme version of a quantum circuit designed to be as featureless as possible is the seminal “quantum supremacy experiment” [3] of Google LLC. Google initially claimed that simulating their supremacy experiment would require 10 000 years on the largest existing supercomputer. Subsequent studies (that we review in Sec. II) [4–6] showed that this initial surmise was exaggerated and that the task could be executed in a few hundred seconds. This progress in classical simulations could be obtained through a precise analysis of the structure of the quantum circuit. Even though the initial claim by Google has been largely deflated, the Sycamore experiment remains an important benchmark for classical simulations. Besides, the computational cost of previous simulations that challenged Google remains exponential in the number  $N$  of qubits, leaving room for the experimentalists to regain the upper hand by adding a few qubits or gates.

In this article, we show that the Sycamore experiment can be simulated with the same level of fidelity as Google, up to a hundred qubits or more depending on the geometry. Our results are based on a different class of algorithm, borrowed from quantum many-body theory, whose complexity increases only as a power of  $N$ . This exponential gain in computational complexity is obtained in exchange for a *quantum state compression* that implies a finite fidelity of the calculation. In this sense, these algorithms share some characteristics with actual quantum computers, which also suffer from a finite fidelity. A first step in that direction was taken in Ref. [7], where some of us developed a quantum circuit version of the time-evolving bond-decimation [8–10] (TEBD) algorithm. It was found that surprisingly good fidelities could be obtained at a relatively low computational cost. Here, we develop a generalization of the density-matrix renormalization group [11,12] (DMRG) algorithm to quantum circuits. Although technically more complex, the DMRG algorithm allows one to improve on the TEBD algorithm in a systematic way.

We benchmark our quantum circuit DMRG algorithm on three different quantum circuits: the “quantum supremacy” sequence of Ref. [3]—optimized to be as difficult to simulate as possible—another slightly easier random circuit, and a “useful” circuit used in the

quantum approximate optimization algorithm (QAOA) [13] for combinatorial optimization. We find that even with the most difficult “quantum supremacy” sequence, the DMRG algorithm can produce bitstrings that have the same quality as that demonstrated in Ref. [3] on a single computing core. More importantly, we find that, for the QAOA sequence, we reach a fidelity per gate much higher than that found in the Sycamore processor. Our numerical experiments provide important benchmarks of the fidelities that can be reached on a classical computer, and therefore of what the quantum hardware must fulfill to claim genuine quantum supremacy or advantage. Since our DMRG algorithm scales only polynomially with  $N$ , it implies that quantum computers must improve their fidelities to access regimes that cannot be simulated.

This article is organized as follows. In Sec. II, we review the current status of quantum supremacy and of quantum circuit simulation techniques. Section III contains a summary of the main findings of this article. The DMRG technique is developed in Sec. IV. Section V showcases how the DMRG technique works in practice and discusses some implementation details. Section VI discusses in which regime the DMRG algorithm provides an optimum solution. Section VII shows the relation between the fidelity and the cross-entropy benchmarking obtained in our simulations. Finally, we conclude in Sec. VIII. This article relies heavily on tensor network techniques. Appendix A contains a short self-contained introduction to tensor networks for readers unfamiliar with these techniques.

## II. A CRITICAL REVIEW OF QUANTUM SUPREMACY

Almost three years ago, the announcement by Google of having reached the milestone of “quantum supremacy” dazzled both the academic community and the general public [3]. Google managed to control  $N = 53$  transmon qubits and to perform a circuit comprising 430 two-qubit gates. They obtained a quantum state that had a small (approximately 0.002), yet measurable overlap with the ideal state that they were supposed to get. While this state did not permit any useful computation, Google surmised that producing something similar using classical simulations would be prohibitive (10 000 years on the largest supercomputer) and hence claimed to have reached quantum supremacy.

Since Ref. [3], another group has produced an almost identical experiment using a very similar technology [14]. There has also been other claims of quantum supremacy, most notably using “boson sampling” [15]. Here, we will not discuss these more recent claims for two reasons. First, the hardness of these tasks is strongly debated [16–18]. Second, and more importantly, while the Google experiment represents a milestone on the path towards building a

quantum computer, these most recent claims correspond to very specific tasks and the devices are not programmable.

Here, we review the status of the classical simulation challenges to these “quantum supremacy” claims, namely, we review the various works that have attempted to simulate the experiments in a reasonable classical computing time [4–6,19]. In particular, we try to explain in simple terms why the claim of “10 000 years” was initially challenged to be only two days and why it has eventually been shown that a simulation of quantum supremacy could be performed in a few hundred seconds.

### A. An exponentially difficult experiment

A quantum computer with  $N$  qubits has an internal state that can be written as

$$|\Psi\rangle = \sum_{i_1 i_2 i_3 \dots i_N} \Psi_{i_1 i_2 i_3 \dots i_N} |i_1 i_2 i_3 \dots i_N\rangle, \quad (1)$$

where  $i_1 \in \{0, 1\}$ ,  $i_2 \in \{0, 1\}, \dots, i_N \in \{0, 1\}$  correspond to the different qubits. The vector  $\Psi$ , whose components are the complex numbers  $\Psi_{i_1 i_2 i_3 \dots i_N}$ , can be considered as a large vector of dimension  $2^N$ . One initializes the state in  $|\Psi(0)\rangle$  (typically all qubits in state 0, i.e.,  $\Psi_{i_1 i_2 i_3 \dots i_N} = \prod_p \delta_{i_p, 0}$ ) and then applies a sequence of unitary gates. Formally, these gates transform the state of the quantum computer into

$$|\Psi(D)\rangle = U^{(D)} U^{(D-1)} \dots U^{(2)} U^{(1)} |\Psi(0)\rangle, \quad (2)$$

where the  $U^{(p)}$  are unitary matrices (two-qubit gates or a combination thereof). A direct simulation of Eq. (2) by a series of (sparse) matrix-vector multiplications is referred to as a “Schrödinger approach.” In an experiment, however, one does not have access to the many-qubit wave function. Instead, one measures the different qubits and obtains a bitstring  $x = i_1 i_2 \dots i_N$  with probability  $Q(x)$ .

Arute *et al.* [3] used  $N = 53$  qubits and a highly unstructured set of  $N_{2g} = 430$  two-qubit gates spread over  $D = 20$  layers. The experiment was repeated  $N_{\#} \approx 10^6$  times to produce a sequence of bitstrings  $x_1, \dots, x_{N_{\#}}$ . Since the quantum sequence was highly unstructured, the distribution  $Q(x)$  was expected to be fairly chaotic, so that all bitstrings  $x$  would have a probability to be obtained of the order  $\propto 1/2^N$ , i.e., the experiment essentially outputs random bitstrings. Ref. [3] is primarily a global system validation experiment. The authors performed exact numerical simulations of Eq. (2) to obtain the exact distribution  $P(x) = |\langle x | \Psi(D) \rangle|^2 = |\Psi_{i_1 i_2 i_3 \dots i_N}^{(D)}|^2$  that should have been obtained from the experiment. By estimating how the perfect distribution  $P(x)$  correlates with the distribution  $Q(x)$  obtained experimentally, one is able to assert to which degree the quantum computer has performed the requested task. The metric used to analyze this correlation is the

“cross-entropy benchmarking”

$$\mathcal{F}_B \equiv 2^N \sum_x P(x) Q(x) - 1, \quad (3)$$

which can be estimated experimentally as

$$\mathcal{F}_B \approx \frac{2^N}{N_{\#}} \sum_{\alpha=1}^{N_{\#}} P(x_{\alpha}) - 1, \quad (4)$$

where  $N_{\#}$  is the total number of repetitions of the experiment and  $x_{\alpha}$  the measured bitstring for repetition  $\alpha$ . It is expected theoretically, and observed experimentally, that, due to decoherence and imprecisions in the gates and measurements, the cross-entropy benchmarking decays exponentially:

$$\mathcal{F}_B \propto e^{-\epsilon_B N D / 2} \quad (5)$$

with an error rate  $\epsilon_B$ . Arute *et al.* [3] were able to verify Eq. (5) with an error  $\epsilon_B \approx 1\%$ . For the largest depth  $D = 20$  where the exact distribution  $P(x)$  was too computationally costly to be calculated, they extrapolated that  $\mathcal{F}_B \approx 0.2\%$ . The “quantum supremacy” claim was that obtaining a set of  $N_{\#}$  bitstrings with the same fidelity  $\mathcal{F}_B \approx 0.2\%$  by simulations would require 10 000 years on the largest supercomputer. We emphasize that here classical simulations are used for two very different purposes: the first is to provide the value  $P(x_{\alpha})$  without which the cross-entropy benchmarking cannot be evaluated in the experiment and the second is to provide bitstrings (approximately) generated according to  $P(x)$  in order to spoof the experiment.

It should be noted that this experiment is exponentially difficult. Indeed, in order for the set of bitstrings to be distinguishable from just plain random bitstrings distributed uniformly, one needs the statistical error in the estimation of Eq. (4) to be smaller than what is estimated, i.e.,  $\mathcal{F}_B$ . Since the statistical error decreases with the number of repetitions as  $1/\sqrt{N_{\#}}$ , it implies an exponentially large number of samples,

$$N_{\#} \propto e^{\epsilon_B N D}. \quad (6)$$

Arute *et al.* [3] pushed the quantum chip to the extreme limit where there remained just enough fidelity for the signal to be measured. For instance, going to  $D = 39$  would have implied an increase of the measurement time by a factor of  $10^6$ . The authors also had to give up a little on the universality or programmability of the chip to maintain a low enough error rate  $\epsilon_B$ : they chose, for each pair of qubit, the two-qubit gate that had the best fidelity by optimizing the microwave pulses. Subsequent experiments that used the same chip but focused on “useful” quantum circuits used only 10–20 qubits to retain accurate enough results [20].

## B. Exchanging a smaller memory footprint for an exponential increase in the computational time

Around the time of Google’s supremacy claim, a team from IBM proposed an algorithm that, according to their estimation, would require only 2.5 days to complete the supremacy task instead of 10 000 years [21]. Such a speed up (a factor of  $10^5$ ) begs for an explanation. A direct naive “Schrödinger” evaluation of Eq. (2) would require of the order of  $N_{2g}2^N$  floating point operations by holding the vector  $\Psi$  in memory and applying the two-qubit gates one by one. Such an algorithm would require  $10^{18}$  operations. Hence, since large supercomputers can perform more than  $10^{17}$  floating point operations per second, the supremacy task could, according to this naive estimation, be performed in at most a few minutes, not thousands of years. This however requires one to hold a vector of size  $2^N$  in memory, i.e.,  $10^5$  terabytes of random access memory (RAM), which is more than what supercomputers have (typically by more than a factor of 10). To get around this difficulty, one designs algorithms that require exponentially more operations in exchange for a smaller memory footprint.

To illustrate how the trade-off between memory footprint and computational time can be implemented in practice, imagine that we group the qubits into two groups of  $N_1$  and  $N_2$  qubits ( $N_1 + N_2 = N$ ). A first index  $\alpha$  labels the first group  $i_1 i_2 \cdots i_{N_1}$  and a second index  $\beta$  labels the second group  $i_{N_1+1} \cdots i_N$ . An arbitrary gate  $U^{(p)}$  has matrix elements  $U_{\alpha\beta;\alpha'\beta'}^{(p)}$ . Such a tensor can be considered as a matrix where the two indices ( $\alpha, \alpha'$ ) are considered as a metaindex that indexes the rows and the two other indices ( $\beta, \beta'$ ) index the columns. Using singular value decomposition (SVD), such a matrix can be factorized into a sum of  $\chi_p$  terms of the form

$$U_{\alpha\beta;\alpha'\beta'}^{(p)} = \sum_{a=1}^{\chi_p} V_{\alpha\alpha'}^{(p)} W_{a\beta\beta'}^{(p)}, \quad (7)$$

where  $V_a^{(p)}$  and  $W_a^{(p)}$  act separately on the first and second groups of qubits, respectively (see Appendix A for details on the SVD operation). Since the wave function initially factorizes,  $\Psi_{\alpha\beta}^{(0)} = \Psi_{1\alpha}^{(0)} \Psi_{2\beta}^{(0)}$ , one can rewrite Eq. (2) as

$$\Psi_{\alpha\beta}^{(D)} = \sum_{a_1, \dots, a_D} \Psi_{1\alpha}^{(D)} \Psi_{2\beta}^{(D)} \quad (8)$$

with

$$\Psi_1^{(D)} = V_{a_D}^{(D)} V_{a_{D-1}}^{(D-1)} \cdots V_{a_1}^{(1)} \Psi_1^{(0)}, \quad (9)$$

$$\Psi_2^{(D)} = W_{a_D}^{(D)} W_{a_{D-1}}^{(D-1)} \cdots W_{a_1}^{(1)} \Psi_2^{(0)}. \quad (10)$$

Now, we only need to perform matrix vector products of much smaller sizes,  $2^{N_1} \ll 2^N$  and  $2^{N_2} \ll 2^N$ . In return

for this much smaller memory footprint,  $\Psi_1^{(D)}$  and  $\Psi_2^{(D)}$  depend explicitly on the SVD indices  $a_1, \dots, a_D$ . One has to repeat the calculation for each  $a_1, \dots, a_D$  to perform the sum, which has an exponential computational cost  $\propto \prod_p \chi_p$ . Simulations that use Eqs. (8)–(10) are known as “Schrödinger-Feynman” simulations. In a Schrödinger-Feynman simulation, the only problematic gates are the two-qubit gates that couple the two groups. These gates have  $\chi_p = 2$  (control-NOT or control-Z gates) or at most  $\chi_p = 4$  (arbitrary two-qubit gates). For all the gates that do not couple the two groups of qubits,  $\chi_p = 1$  and there is no increase in the computational time. Another aspect is that one can calculate the amplitude  $\Psi_{\alpha,\beta}$  for as many configurations  $\alpha, \beta$  as needed with no additional cost except for that of storing these amplitudes in memory. The initial statement of Google of 10 000 years was associated with an estimation of the computational cost of a Schrödinger-Feynman simulation. We see that this computational estimation is strongly tied to the available memory. More memory would allow one to perform an optimized splitting of the qubits into two groups or no splitting at all, resulting in a much smaller computational cost. The IBM proposal [21], which was not implemented, was to take advantage of the hard drives of the supercomputer as temporary storage so that the full  $N$ -qubit wave function could be held in memory, thereby considerably reducing the computational time. The drawback of this approach, besides the obvious difficulty of performing an actual implementation, is the fact that adding just one extra qubit would require a doubling of the memory footprint, hence making the simulation out of reach.

## C. The hierarchy of “open” versus “closed” versus “weak” simulations

The final blow on the supremacy claim came from a combination of works [4–6, 19], in which the authors found a route to perform the simulation of the “quantum supremacy” experiment in a few hundred seconds and demonstrated that the solution could be implemented in an actual very large supercomputer. This series of works essentially closed the gap between the simulations and the experiments. This corresponds to a drop by a factor of  $10^9$  with respect to the initial estimate of 10 000 years. This new gain comes from the combination of two new ingredients.

The first important point is that there are several simulation modes of decreasing power. The Schrödinger (Schrödinger-Feynman) simulation provides the full  $N$ -qubit wave function (as many amplitudes as one can store). We refer to this simulation mode as “open” in the sense that they do not target a specific bitstring  $x$ . Open simulations produce much more information than what the experiment outputs. Another simulation mode, which we refer to as a “closed” simulation, targets a single bitstring



$x$  and computes the amplitude

$$\Psi_x^{(D)} = \langle x | \Psi(D) \rangle = \langle x | U^{(D)} \dots U^{(2)} U^{(1)} | 0 \rangle. \quad (11)$$

Closed simulations are generically much easier than open ones. A final type of simulation, a “weak” simulation, produces the same output as an actual quantum computer, namely, random bitstrings distributed according to the probability  $|\Psi_x^{(D)}|^2$ . There exists a clear hierarchy between these different simulation modes: an open simulation provides more information than a closed one, which in turn provides more information than a weak simulation (or an actual quantum computer). One of the key steps in speeding up our simulations is to go from the open mode to the closed one.

The fact that closed simulations are superior to weak ones is not totally obvious. It follows from a simple algorithm recently proposed in Ref. [22] that allows one to sample  $|\Psi_x^{(D)}|^2$  from the calculation of a polynomial number of individual amplitudes  $\Psi_x^{(D')}$  at smaller depth  $D' \leq D$ . The algorithm of Ref. [22] constructs a bitstring  $x^D$  iteratively, starting from the initial bitstring  $x^0 = 00 \dots 0$  and taking into account the two-qubit gates one by one. Here  $x^{D'+1}$  is identical to  $x^{D'}$  except for the two qubits that are affected by the two-qubit gate. There are only four such bitstrings,  $x_1, x_2, x_3, x_4$ . One computes the four probabilities  $p_i = |\Psi_{x_i}^{D'+1}|^2 / \sum_j |\Psi_{x_j}^{D'+1}|^2$  and samples from this conditional distribution, i.e.,  $x^{D'+1} = x_i$  with probability  $p_i$ . It is straightforward to verify that this scheme indeed provides a bitstring distributed according to  $|\Psi_x^{(D)}|^2$ . Note that in the context of the supremacy experiment where all  $\Psi_x^{(D)}$  have similar orders of magnitude, this algorithm is not necessary and a simple Metropolis-Hastings sampling could be used instead (see the discussion in Appendix C).

#### D. Optimized contraction strategies of tensor networks

In the closed simulation mode, the challenge of the calculation lies in the summation over all the internal indices of the tensors (like the  $a_1, \dots, a_D$  indices introduced previously) in Eq. (11). Finding the optimum order for these summations is a hard (NP complete) problem, but there exist good heuristic algorithms for finding close to optimum contraction strategies [4]. These optimum orders are in general very different from a simple summation from right to left, i.e., from the contraction done in a Schrödinger or Schrödinger-Feynman simulation. The memory versus CPU trade-off is implemented using a “slicing” approach: one carefully selects a few indices that are frozen in order to lower the cost of contracting the tensor network. The different values taken by these indices (the equivalent of the  $a_1, \dots, a_D$  in the Schrödinger-Feynman simulations) are distributed over different computing nodes or GPU cards as these tasks are completely independent (embarrassingly parallel). For the reader not familiar with the concept of

tensor contractions and slicing, a small introduction is given in Appendix A.

Using these techniques (i.e., closed simulations and good contraction strategies), Gray and Kourtis [4] estimated that the time to compute a single amplitude on a graphics card (GPU) could be reduced down to 3088 years with perfect fidelity. The same authors estimated that it would take 197 days to match the supremacy experiment, i.e., produce one million samples with the 0.2% cross-entropy benchmarking fidelity. To arrive at this estimate, they took advantage of (i) the computing resources of the large supercomputer “Summit,” (ii) the fact that the computing time to compute a few amplitudes that differ only by the value of a few qubits is not significantly higher than computing a single amplitude, and (iii) the fact that a fidelity of 0.2% can be obtained by mixing a few (2000) high-amplitude (large  $|\Psi_x|^2$ ) samples with 998 000 bitstrings sampled from a uniform random distribution.

A few months later, the estimated time to sample one million bitstrings with 0.2% fidelity was further reduced to 19.3 days [23], based on similar ideas and refinements by a team at Alibaba. These authors estimated the time to compute one perfect sample on Summit to 833 seconds. Like the previously mentioned study [4], they proposed to sample Sycamore by computing batches of 64 amplitudes at no significant cost increase in a partly “closed,” partly “open” mode.

The approach was further optimized by Pan and Zhang [5] in the so-called “big-batch method” that optimized the choice of the qubits left in the open mode. They managed to compute two million bitstrings with a large 73.9% cross-entropy benchmarking in five days on a small cluster of 60 GPUs [5]. However, these bitstrings had many qubits in common and hence were strongly correlated. In a second study [6], they proposed a new “sparse state method” that lowers the contraction cost of the supremacy circuit tensor network by cleverly introducing a few errors at specific locations. They produced  $2^{20}$  independent batches of 64 correlated bitstrings in 15 h on a cluster of 512 GPUs, and via importance sampling finally obtained one million uncorrelated samples with 0.37% fidelity. In the same work, they estimated that the sampling time for Sycamore could be reduced to a few dozen seconds on a large supercomputer.

Liu *et al.* [19] produced two million correlated bitstrings from Sycamore with 0.2% cross-entropy benchmarking fidelity in 304 seconds. This calculation was performed on the Sunway TaihuLight supercomputer with 42 million effective cores, with an algorithm inspired by the work of Pan and Zhang [5], and taking advantage of a new heuristic for slicing and contraction path optimization. It demonstrated that the parallelization of these algorithms could be effectively implemented on a very large supercomputer. Together, Pan *et al.* [6] and Liu *et al.* [19] convincingly showed that a supercomputer can match the Sycamore

chip, even for a task whose only interest lies in having been optimized to be difficult to simulate.

Hence, the initial claim of “quantum supremacy” has been, to a large extent, deflated. However, the classical simulations reviewed above required colossal resources to achieve this goal. Since the problem is exponentially hard, a marginal improvement of the qubit fidelity (which would allow the experiment to go to larger depth) would have made the problem inaccessible to simulations. A second aspect is that problems that are impossible to simulate are easy to find, and the quantum supremacy experiment is to a large extent an artificial problem constructed for the sole purpose of being difficult to simulate. The question remains of what the quantum supremacy experiment taught us in terms of where one stands in the route to building genuine quantum computing capabilities for useful problems. In this article, we also use tensor network simulations to benchmark the performance of quantum computers. However, our focus is very different from the above high-performance computing calculations. While previous simulations are essentially exact with a small linear speed up coming from the finite targeted fidelity, we borrow quantum state compression techniques from many-body theory that exchange a finite fidelity for an exponential gain in computing time. As we will see, these complementary techniques provide strong insights in the influence of a finite fidelity on computing capabilities and what it would take for a quantum computer to reach a regime that is both interesting and out of reach of simulations.

### III. SUMMARY OF THE MAIN RESULTS

In this article, we develop an approximate DMRG algorithm for the simulation of quantum circuits. This algorithm is based on several controlled compression steps, where the compressed state is optimized variationally to maximize the fidelity. In exchange for the loss of information in the compression, the computational cost is considerably reduced. This technique is very different from existing tensor network simulation techniques for quantum circuits as they do not use compression. However, in principle, these other approaches could be combined with our compression technique. Before going into the mathematical details of how the technique works, we report on the results of our simulations for a few relevant circuits. Denoting by  $|\Psi_P\rangle$  the perfect state that one should obtain and by  $|\Psi_Q\rangle$  the actual approximate state obtained in the simulation, the main quantity of interest in this article is the fidelity  $\mathcal{F}$  of the simulation, defined as

$$\mathcal{F} = |\langle \Psi_Q | \Psi_P \rangle|^2. \quad (12)$$

Since  $\mathcal{F}$  decreases exponentially with the number  $N_{2g}$  of two-qubit gates [ $\mathcal{F} \approx \exp(-\epsilon N_{2g})$ ] in these simulations,

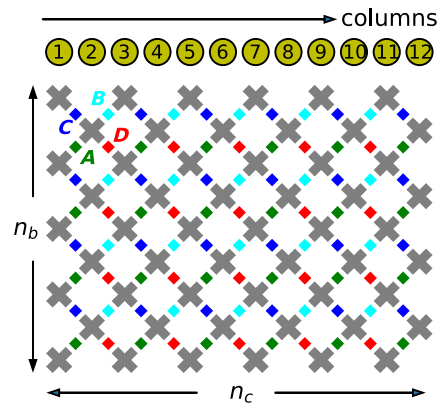


FIG. 1. Topology of the “quantum processor” simulated in this work. The system has  $n_c$  columns containing alternatively  $n_b$  and  $n_b - 1$  qubits. Here  $n_c = 12$  with  $n_b = 5$  corresponds to a 54-qubit planar processor that has the same topology as the Sycamore chip. The qubits have nearest-neighbor connectivity. In the quantum circuits corresponding to sequences I and II (see the text), the circuit is split into layers. In each layer, one applies a two-qubit gate between all the pairs of qubits coupled by a green rectangle ( $A$  layers), a light blue rectangle ( $B$  layers), a dark blue rectangle ( $C$  layers), or a red rectangle ( $D$  layers).

we define the error rate per two-qubit gate  $\epsilon$  as

$$\epsilon = 1 - \mathcal{F}^{1/N_{2g}} \approx -\frac{1}{N_{2g}} \log \mathcal{F}. \quad (13)$$

This quantity can be directly compared to experiments. For instance, assuming that  $\mathcal{F}_B = \mathcal{F}$ , then the  $\mathcal{F}_B = 0.2\%$  of the quantum supremacy experiment translates into an error  $\epsilon = 1.4\%$  per two-qubit gate, for each of the  $N_{2g} = 430$  two-qubit gates. This effective value accounts for the actual two-qubit gate errors (around 1%), the one-qubit gate errors (around 0.1%), and the measurement errors (around 3%).

We perform most of our simulations on a ( $N = 54$ )-qubit system very close to the 53-qubit Sycamore chip of Ref. [3]; see Fig. 1. All the simulations presented here have been performed with limited computational resources: one to few computing processes (fewer than 12) that have lasted at most a few hours. Simulations ran on desktop PCs or on clusters with large memory for larger simulations, on a few computing cores in both cases. We consider three different quantum circuits.

*Sequence I* is essentially the quantum supremacy sequence of Google. Here  $D = 20$  layers are applied. For each layer, one applies a random one-qubit gate on each qubit, followed by the so-called fsm gate on all pairs of qubits according to the pattern  $ABCD\text{-}CDBA\text{-}ABCD\text{-}\dots$  (see Fig. 1). Sequence I has been designed to entangle the qubits as quickly as possible and therefore be as hard to simulate as possible.

Sequence II is identical to sequence I but with a pattern rotated by  $90^\circ$ , i.e., the sequence is *CDBA-BACD-CDBA*...

Sequence III is, in contrast to sequences I and II, designed to perform a supposedly useful task. It implements the QAOA. The QAOA is attracting a lot of attraction as a candidate algorithm to solve combinatorial optimization problems on a (noisy) quantum computer [24]. It can be viewed as a discrete

variational version of the adiabatic quantum computing paradigm.

Our main findings are summarized in the next three subsections.

### A. Simulating the supremacy sequence

Figure 2(a) shows the error  $\epsilon$  obtained in our simulations for the quantum supremacy sequence I with a system

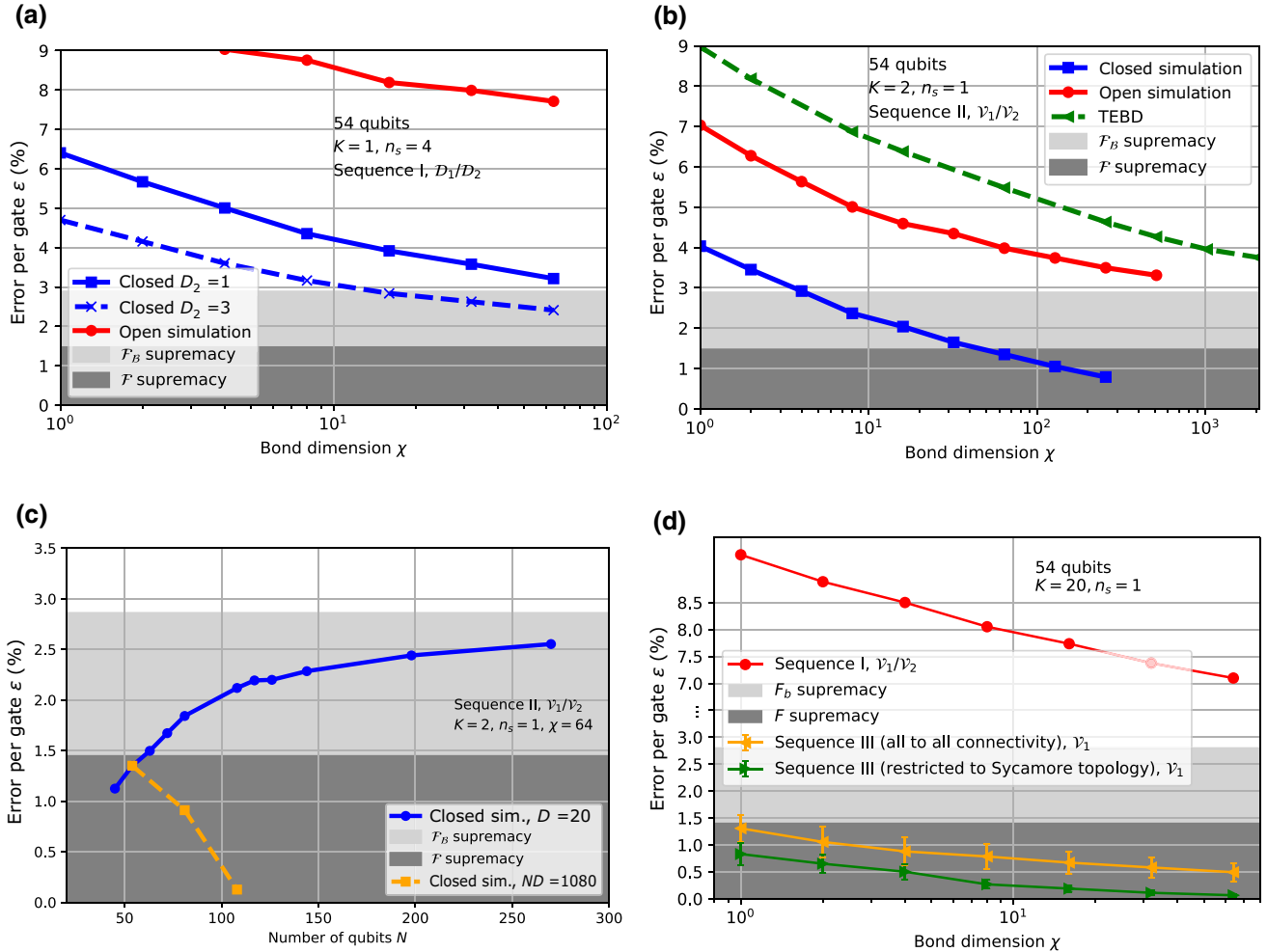


FIG. 2. Error rates achieved in our simulation. (a) Error rate per gate  $\epsilon$  as a function of bond dimension  $\chi$  for the supremacy sequence I. In the gray regions the quality of the output is as good or better than that produced in Ref. [3]; see the text. Light gray denotes  $\epsilon \leq 2.8\%$ ; the simulation fidelity matches the experiment even assuming that  $\mathcal{F}_B = \sqrt{\mathcal{F}}$ . Dark gray denotes  $\epsilon \leq 1.4\%$ ; the simulation fidelity matches the experiment even assuming that  $\mathcal{F}_B = \mathcal{F}$ . We denote by  $D_2$  the number of layers treated exactly at the end of the calculation; see Sec. IV C. The labels  $\mathcal{D}_1, \mathcal{D}_2, \nu_1, \nu_2$  indicate the grouping used in the construction of the matrix product state (MPS); see the discussion around Fig. 7. We denote by  $n_s$  the number of sweeps used in the DMRG optimization;  $K$  is the number of circuit layers. (b) Same as (a) for sequence II. (c) Error rate per gate  $\epsilon$  as a function of the number of qubits  $N$  for sequence II. Here  $n_c$  is increased at fixed  $n_b = 5$  and  $\chi = 64$ . Blue curve denotes the fixed depth  $D = 20$ . Orange curve denotes the fixed number of two-qubit gates, i.e.,  $ND = 1080$ . (d) Comparison between the error rates of sequences I and III in open simulations. Each point in the sequence III curves is averaged over ten graphs, with error bars corresponding to one standard deviation. Orange curve denotes the QAOA circuit assuming perfect topology. Green curve denotes the QAOA circuit using only the nearest-neighbor connectivity of the Sycamore chip.

of  $N = 54$  qubits. The  $x$  axis is the bond dimension  $\chi$  that controls the level of compression of the quantum state and hence the accuracy. The computational cost of the simulation scales polynomially as  $\propto \chi^2$  with a memory footprint  $\propto \chi^2$ . An exact simulation would correspond, at large depth, to an exponentially large  $\chi = 2^{N/2}$ , but here we are very far from this regime.

The red curve shows the results in the “open” mode. We find that the error rate for the largest  $\chi = 64$  studied is fairly high, around 8%, much higher than in state-of-the-art experiments. Going to the “closed” mode provides an important gain in error rate, typically by more than a factor of 2 with our technique, enabling one to reach  $\epsilon \approx 3\%$ . By optimizing the closed mode (curve  $D_2 = 3$ ; the details of the closed mode will be explained later), one reaches  $\epsilon \approx 2.5\%$  at a computational cost that is still moderate.

To compare this error rate with the experimental one, we need to know the error rate associated with the experimentally measured cross-entropy benchmarking, i.e.,  $\epsilon_B = 1 - \mathcal{F}_B^{1/N_{2g}} \approx -(\log \mathcal{F}_B)/N_{2g}$ , not  $\epsilon$ . Arute *et al.* [3] have argued that, for their experiment,  $\mathcal{F}_B \approx \mathcal{F}$  for large enough depths. While this statement can be proven for certain classes of noise, it is not universal. For instance, it cannot hold at small depth since  $\mathcal{F}(D=0) = 1$  while  $\mathcal{F}_B(D=0) = 2^N - 1$ . Nor does it hold for systems consisting of disjoint pieces, for which fidelity composes multiplicatively while  $\mathcal{F}_B$  composes additively when small [25].

For our simulation technique, a very different relation holds:

$$\mathcal{F}_B \approx \sqrt{\mathcal{F}}. \quad (14)$$

We provide strong evidence, both numerical and analytical, to support Eq. (14). It follows that the  $\epsilon = 2.5\%$  obtained in our simulations corresponds to  $\epsilon_B = \epsilon/2 = 1.25\% \leq 1.4\%$  (light gray zone in Fig. 2). Hence, the bitstrings provided by these simulations have a higher cross-entropy benchmarking fidelity than those provided by Arute *et al.* [3] and in that sense our technique can be considered as another breach in the claim of quantum supremacy. Note that, since the relation between  $\mathcal{F}$  and  $\mathcal{F}_B$  is highly nonuniversal, the fact that Eq. (14) holds in our simulation does not imply a similar relation in the experiments.

## B. Scaling with the number $N$ of qubits

A very important difference between the present work and previous attempts at bridging the quantum supremacy gap is the scaling of the simulations with the number of qubits. Indeed, in our simulations, the exponential difficulty lies in increasing the fidelity, not the number of qubits.

In our present implementation, the computational cost of a simulation scales as  $e^{\beta n_b} n_c D \chi^2$ , where  $n_b$  is the number

of qubits in the first column and  $n_c$  the number of columns (see Fig. 1). The memory required scales as  $e^{\beta n_b} n_c \chi^2$ . The parameter  $\beta \geq \log 2$  depends on the precise mode of calculation. We note that, while the present work uses one-dimensional tensor networks (MPSs), other networks such as the projected entangled pair states (PEPSs; two-dimensional networks) should provide a more favorable computational cost linear in both  $n_b$  and  $n_c$  [26,27].

The scaling with  $N$  is illustrated in Fig. 2(c), where we show a calculation as a function of  $N$  (by varying  $n_c$ ) at fixed  $\chi$  for sequence II. We perform simulations with more than 250 qubits, while the error  $\epsilon$  shows a limited increase before saturating (see the discussion in Ref. [7]). Such simulations would be totally out of the scope of any other existing simulation approach that does not use compression, since they all scale exponentially with  $N$ . We emphasize again that the experiment corresponding to the blue curve in Fig. 2(c) would be exponentially difficult: one cannot increase  $N$  at fixed  $D$  experimentally (even assuming that so many qubits would be available) because the cross-entropy benchmarking would become too small to be measurable in a reasonable time. Working at fixed experimental measurement time, i.e., fixed  $\mathcal{F}_B$  or, equivalently, keeping the product  $ND$  constant, corresponds to the orange curve in Fig. 2(c). We see here a first difference in behavior between our compression algorithms and actual experiments: our error rate  $\epsilon$  actually drastically drops with  $N$  in the orange curve, as our algorithm becomes essentially exact at small depth. This indicates that in order to beat compression algorithms, quantum hardware must improve in *fidelity* and/or *connectivity*: a mere increase in the number of qubit is not sufficient.

## C. Influence of the quantum circuit on the fidelity

A second important difference between actual experiments and our compression algorithm appears upon considering different quantum circuits. One of the chief results of Ref. [3] is that the fidelity of the experiment depends only on the number and type of gates applied and should to a large extent be agnostic to the type of circuit ran. In practice, however, random circuits such as sequence I or II are experimentally easier than more structured circuits. This is due to several factors: (i) these random circuits are optimally parallel without any idle time that could lead to further decoherence; (ii) there is a compensation of errors due the random choice of gates; and (iii) in the case of Ref. [3], a pair-by-pair optimization of the fidelity of the two-qubit gates that could not have been performed had these gates corresponded to the prescription of an algorithm. This increased difficulty—for experimental hardware—of running structured circuits is well known in the field of quantum benchmarking; see, e.g., Ref. [28].



In our compression algorithm, we find, in sharp contrast with the above observations, that more structured quantum circuits are much easier to compress than random ones. While this result is not surprising on a qualitative level, the magnitude of the improvement in error rate that we observe is very high, with an error rate for open simulation dropping by a factor of 100, from 8% for random circuits down to 0.07% for QAOA circuits with the same  $N = 54$  and  $N_{2g} = 430$ .

Figure 2(b) shows a result for sequence II, which is only a slight modification of sequence I. We observe that this slight modification of the sequence provides a twofold gain in  $\epsilon$ , bringing the open simulation almost down to the gray region and the closed simulations deep into the dark gray region. Figure 2(b) also shows the result of the TEBD algorithm of Ref. [7] (green curve), showing that the DMRG algorithm presented in this article is a clear improvement over the TEBD algorithm.

Figure 2(d) contrasts the results between sequence I and the QAOA sequence III. The results for the QAOA sequence correspond to the same number of qubits  $N = 54$  and the same two-qubit gate count  $N_{2g} = 430$ , i.e., such that experimentally one would expect a fidelity *lower* than the  $\mathcal{F}_B = 0.2\%$  observed for sequence I, for the reasons mentioned above. In contrast, the results of the compression algorithm are drastically better for sequence III than for sequence I. Two plots are presented in Fig. 2(d). The orange curve shows the simulation results of the QAOA sequence supposing that the  $N = 54$  chip had perfect connectivity (a two-qubit gate can be applied between any pair of qubits). We find that the error rate *in an open simulation* is reduced by a factor of 14 compared to sequence I, with  $\epsilon = 0.5\%$  at  $\chi = 64$ . The green curve corresponds to a circuit that respects the nearest-neighbor topology of the Sycamore chip. This topology puts additional constraints on the graphs that can be optimized with a given “budget”  $N_{2g}$  of two-qubit gates. It results in simpler graphs being simulated, and a further drop in the error rate down to 0.07% for  $\chi = 64$  in the hardest *open* simulation mode. We note that two recent works [29,30] considered a related problem (performance of a TEBD approach similar to Ref. [7] for a QAOA optimization) and arrived at conclusions that are, at least qualitatively, consistent with ours. The simulation of QAOA circuits with up to 54 qubits was also tackled in a recent work [31] using an alternative representation of the quantum state based on neural networks, the restricted Boltzmann machine [32], with similar conclusions.

While it is difficult to draw general conclusions from specific experiments, we conjecture that “useful” quantum circuits, structured by nature, are generically *much* easier to simulate than random ones. It follows that in order for a quantum computer to show a genuine quantum advantage (i.e., quantum supremacy but for a useful task), far better fidelities will need to be demonstrated by the hardware.

#### IV. A DENSITY-MATRIX RENORMALIZATION GROUP ALGORITHM FOR SIMULATING QUANTUM CIRCUITS

We now describe the quantum state compression algorithm used in this article. This algorithm is inspired by the DMRG algorithm that has been highly instrumental for solving one-dimensional (1D) quantum many-body problems [12,33]. Our algorithm can be considered as a “unitary” version of the original “Hermitian” DMRG algorithm.

Our method combines a Schrödinger type of simulation with compression steps where we approximate the quantum state with a matrix product state ansatz. This compression is performed every few layers of gates and requires one to find optimum contraction strategies for small tensor networks. The main new ingredient of this algorithm is the compression step. We emphasize that, although we introduce it in the context of a Schrödinger type of simulation, this step is in fact very general and could in principle be combined with other tensor network simulation approaches.

In a previous study [7] some of us had already considered compressed MPS states and designed a TEBD-like algorithm. The TEBD algorithm required the compression to be performed after each intertensor gate. Also, only gates that coupled one tensor to its nearest neighbours could be considered. In the DMRG algorithm presented here, these two limitations are lifted. We find the optimal compressed state after applying several layers of gates by iteratively solving a variational problem for the optimization of each tensor composing the MPS. This new approach allows us to reach much better fidelities. Most importantly, it is also more systematic: it can be generalized to any kind of quantum circuit and combined with any other tensor network technique.

Since this article relies heavily on the tensor networks naturally associated with quantum circuits, the reader not familiar with these concepts may read the short introduction in Appendix A.

##### A. The matrix product state ansatz

An arbitrary state  $|\Psi\rangle$  with  $N$  qubits

$$|\Psi\rangle = \sum_{i_1 i_2 i_3 \dots i_N} \Psi_{i_1 i_2 i_3 \dots i_N} |i_1 i_2 i_3 \dots i_N\rangle \quad (15)$$

is described by a very large tensor  $\Psi_{i_1 i_2 i_3 \dots i_N}$ . A MPS factorizes and compresses this tensor by writing it as a product of  $m$  tensors contracted in a chainlike geometry:

$$\begin{aligned} \Psi_{i_1 i_2 i_3 \dots i_N} &= \sum_{\alpha_1, \dots, \alpha_{m-1}} M_{i_1 i_2 \dots i_{r_1}, \alpha_1}^{(1)} M_{\alpha_1, i_{r_1+1} \dots i_{r_1+r_2}, \alpha_2}^{(2)} \\ &\times M_{\alpha_{m-1}, i_{N-r_{m+1}} \dots i_{n_d-1} i_N}^{(m)}. \end{aligned} \quad (16)$$

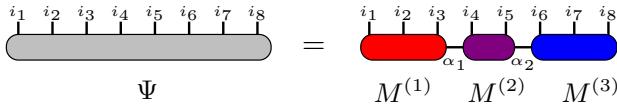


FIG. 3. Decomposition of an eight-qubit state as a three-tensor MPS.

See Fig. 3 for a schematic. Each tensor  $M^{(\tau)}$ ,  $\tau \in \{1, 2, \dots, m\}$ , contains information on the  $r_\tau$  qubits, i.e., we group the qubits into  $m$  groups and each group contains  $r_\tau$  qubits. Entanglement within one group is treated exactly, while only a limited amount of entanglement is allowed between different groups.

The “virtual” indices  $\alpha_\tau$  take at most  $\chi$  values, where  $\chi$  is known as the bond dimension. If  $\chi$  is exponentially large then a MPS can in fact describe any quantum state. Here, however, we restrict ourselves to rather small values of  $\chi$ , in which case the MPS can only be an approximation of the entangled state that one aims at describing. There exists an important literature on many-body problems that can be successfully addressed by MPS variational ansatz [12]. The unentangled initial product state is naturally a MPS with  $\chi = 1$ .

Note that, in contrast to the approach of Ref. [7], grouping the qubits is not, in principle, necessary: one could use instead the conventional  $\forall \tau, r_\tau = 1$  grouping and larger values of  $\chi$  to obtain a variational ansatz as expressive as that used with our nontrivial grouping  $r_\tau \geq 1$ . We find however that, in practice, some groupings can be advantageous for some circuits. For example, if a subset of qubits is highly entangled by the circuit, it should be assigned to the same group. Since the first and last tensors only have one virtual index instead of two, it is computationally advantageous to have more qubits in the corresponding groups.

### B. The main building block of the algorithm: the compression step

The central part of the algorithm performs the following task: one starts from a MPS  $|\Psi_Q(D)\rangle$  at depth  $D$ , supposedly a good approximation of the exact state  $|\Psi_P(D)\rangle$ , i.e., with a high fidelity  $\mathcal{F}(D)$ . The problem is to find the best MPS  $|\Psi_Q(D+K)\rangle$  that approximates the state  $U^{(D+K)} \dots U^{(D+1)} |\Psi_Q(D)\rangle$  after one has applied  $K$  layers of gates of the circuit, as illustrated on Fig. 4(a). Namely, we want to determine

$$|\Psi_Q(D+K)\rangle \equiv \operatorname{argmax}_{|\Psi\rangle, \langle\Psi|\Psi\rangle=1} |\langle\Psi|U^{(D+K)} \dots U^{(D+1)}|\Psi_Q(D)\rangle|^2. \quad (17)$$

To perform this optimization, we optimize one given tensor  $M^{(\tau)}$  of  $|\Psi\rangle$  at a time, while the remaining  $m-1$

tensors are kept fixed. This optimization can be performed exactly using the simple formula (20) derived below. It amounts to the contraction of a small tensor network. To obtain the global optimum, we sweep over the choice of the tensor  $\tau$  as in the single-site DMRG algorithm. Typically, a small number  $n_s$  of sweeps is needed to obtain convergence towards  $|\Psi_Q(D+K)\rangle$ . Note that we have also tried variants of this algorithm analogous to the original two-site DMRG algorithm (where two consecutive tensors are optimized simultaneously), but did not observe any significant improvement with respect to the simpler single-site version.

#### 1. Optimization of a single tensor

Once we fix all tensors  $M^{(\tau')}$  of MPS  $|\Psi\rangle$  except for tensor  $M^{(\tau)}$ , the scalar product to be optimized takes the form

$$\langle\Psi|U^{(D+K)} \dots U^{(D+1)}|\Psi_Q(D)\rangle = \operatorname{Tr} F^{(\tau)} M^{(\tau)*}, \quad (18)$$

where the trace means summation over all indices. Very importantly, this scalar product is a *linear* function of  $M^{(\tau)}$ . The tensor network for the left-hand side of Eq. (18) is shown in Fig. 4(b). It follows that  $F^{(\tau)}$  is defined by the contraction of the tensor network shown in Fig. 4(c). In other words,  $F^{(\tau)}$  simply corresponds to the tensor network for the full scalar product from which the  $M^{(\tau)}$  tensor has been removed. Note that in Fig. 4(c) the two tensors on the right (corresponding to  $\langle\Psi|$ ) are complex conjugated.

Before doing the optimization, we need to enforce the fact that MPS  $|\Psi\rangle$  is a normalized state, i.e.,  $\langle\Psi|\Psi\rangle = 1$ . This is best done by performing a series of QR factorizations on tensors  $M^{(\tau')}$  for  $\tau' \neq \tau$  to bring the MPS in the so-called “orthogonal form”; see Ref. [12]. In this form, the norm of the MPS is simply given by

$$\langle\Psi|\Psi\rangle = \operatorname{Tr} M^{(\tau)} M^{(\tau)*}. \quad (19)$$

Introducing the Lagrange multiplier  $\lambda$ , the optimization over  $M^{(\tau)}$  with the constraint  $\langle\Psi|\Psi\rangle = 1$  boils down to maximizing the function

$$|\operatorname{Tr} F^{(\tau)} M^{(\tau)*} - \lambda(1 - \operatorname{Tr} M^{(\tau)} M^{(\tau)*})|^2.$$

That is, we are maximizing a simple quadratic form. The optimum tensor  $M_{\max}^{(\tau)}$  is easily found, and is related to the “fitting” approach used in the MPS literature [34,35]. It reads

$$M_{\max}^{(\tau)} = \frac{1}{\sqrt{f_\tau}} F^{(\tau)} \quad (20)$$

with  $f_\tau = \operatorname{Tr} F^{(\tau)} F^{(\tau)*}$ . Equation (20) is the central equation, around which all this article is constructed. In

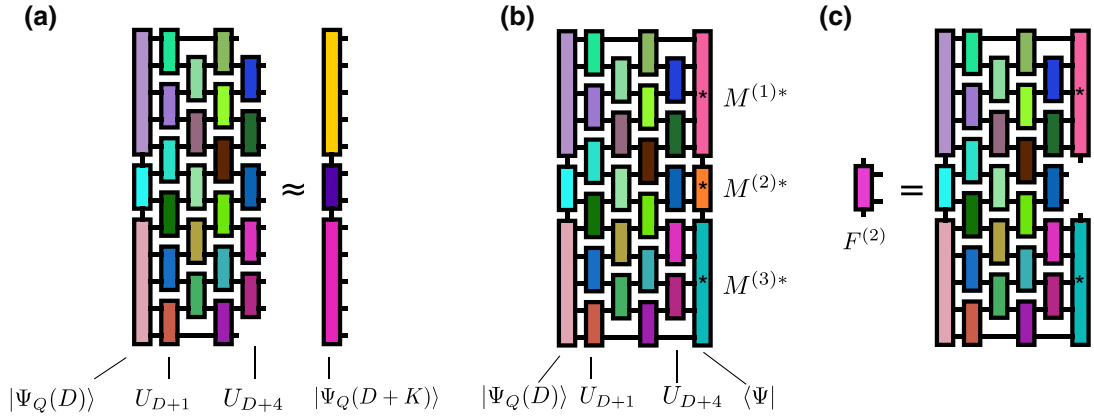


FIG. 4. Compression step in the DMRG algorithm. (a) General schematic of the compression step: one adds  $K$  layers of the quantum circuit, then approximates the resulting state with a MPS. (b) Tensor network representation of the scalar product to be optimized [Eq. (18)]. The asterisk indicates the use of the complex conjugate of the tensor. (c) The central part of the calculation is the computation of the  $F^{(\tau)}$  tensor.

addition, using Eqs. (18) and (20), one finds that the scalar quantity  $f_\tau$  is also the partial fidelity of the calculation

$$f_\tau = |\langle \Psi | U^{(D+K)} \dots U^{(D+1)} |\Psi_Q(D)\rangle|^2, \quad (21)$$

which allows one to keep track of the progress of the optimization inside a sweep over  $\tau$  or over different sweeps.

For each calculation of  $F^{(\tau)}$ , one obtains the local maximum of the partial fidelity with respect to  $M^{(\tau)}$ ; hence,  $f_\tau$  can only increase as we sweep over the different tensors  $\tau = 1 \dots m$ . We can then repeat the sweep over all tensors several times, yielding monotonically increasing fidelities  $f_1^{(1)}, \dots, f_m^{(1)}, f_1^{(2)}, \dots, f_m^{(2)}, \dots, f_1^{(n_s)}, \dots, f_m^{(n_s)}$ . The final value  $f_m^{(n_s)}$  that we obtain after several sweeps over the different tensors is the partial fidelity  $f_\delta$  that will enter our estimate of the fidelity  $\tilde{F}$  [see Eq. (27)], where  $\delta$  indexes the number of compression steps.

## 2. Contraction strategy for the tensor networks

To complete the single tensor optimization, we need to perform the actual computation of tensor  $F^{(\tau)}$ , i.e., we need a strategy for contracting the tensor network of Fig. 4(c). This calculation is performed exactly.

The order in which the contractions are performed has a large impact on the final computation cost. In the case of a deep circuit with few qubits, a ‘‘horizontal’’ contraction order (e.g., from left to right) will save computation cost. The horizontal contraction order corresponds to what is done in Schrödinger-like simulations. Its cost is prohibitive for a large number of qubits due to its exponential memory footprint  $2^N$ . Here, however, we consider a shallow circuit of only a few  $K$  layers at a time, so it is advantageous to perform the contraction in ‘‘vertical’’ order (e.g., from top to bottom) since the exponential cost is with respect to  $K$

instead of  $N$ . This contraction algorithm is a direct adaptation of the well-known algorithm for calculating the scalar product between two MPSs [12].

An example of a contraction path for  $K = 4$  is shown in Fig. 5. We first perform trivial contractions such as contracting one-qubit gates with nearby two-qubit gates. Then, we contract the first top line of tensors and move down until we have reached the (missing) tensor  $\tau$  that is being optimized. We repeat the same procedure from the bottom of the network upwards up to the missing  $\tau$ . Last, we merge the bottom part with the top part. Throughout the tensor network contraction, the largest tensors have  $K$  physical indices (corresponding to the horizontal edges) and two virtual indices (corresponding to the vertical edges). Each physical index represents  $n_b$  qubits, and thus has dimension  $2^{n_b}$ ; each virtual index has dimension  $\chi$ . The typical maximum memory footprint for large  $\chi$  thus scales as  $\chi^2 2^{n_b K}$ . This is much smaller than the  $2^N$  scaling that one would be facing with a naive horizontal contraction path.

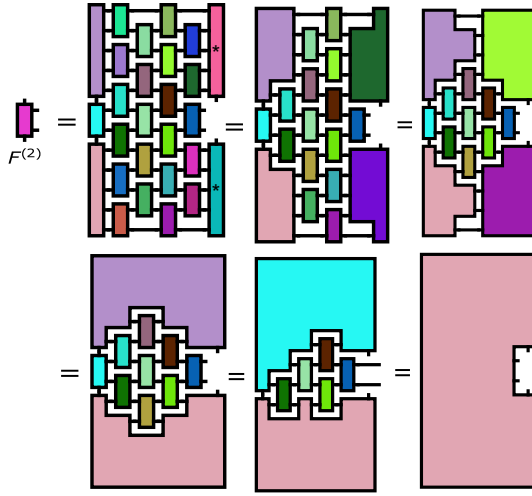
Note that all the tensor network techniques discussed in Sec. II (heuristics for contraction paths, slicing, etc.) could be used here to optimize and/or parallelize the calculation of  $F^{(\tau)}$ .

## C. Open versus closed simulation mode

The algorithm can be used in two modes, open or closed, as discussed in Sec. II. The open, ‘‘Schrödinger-like’’ mode provides the full quantum state after the  $D$  layers of the circuit. One simply adds  $K$  layers at a time using the compression step until one has added all the  $D$  layers.

In the closed simulation mode, we seek to calculate an amplitude

$$\Psi_x = \langle x | U_D U_{D-1} \dots U_3 U_2 U_1 | 0 \rangle \quad (22)$$


 FIG. 5. Contraction path to compute the  $F^{(\tau)}$  tensor.

for a fixed output bitstring  $x$ . A closed simulation calculation corresponds to the overlap of two MPSs, namely,  $U_{D/2+1} \cdots U_{D-1} U_D |x\rangle$  and  $U_{D/2} \cdots U_1 |0\rangle$ , which can be calculated with two separate open calculations whose circuit depths are halved compared to the open simulation mode. Since calculations at small depths give much better fidelities with our technique (there is less entanglement at small depth), the overall error rate  $\epsilon$  is much lower.

In practice, we first partition the circuit into three sub-circuits with respectively  $D_1$ ,  $D_2$ , and  $D_3$  layers,

$$D = D_1 + D_2 + D_3. \quad (23)$$

Then, we perform an open simulation with the first  $D_1$  layers of the circuit (the forward part),

$$|\Psi_Q(D_1)\rangle \approx \prod_{d=1}^{D_1} U_d |0\rangle. \quad (24)$$

Then, we perform a second open simulation starting from the  $|x\rangle$  product state with the last  $D_3$  layers of the circuit (the backward part):

$$|\Psi'_Q(D_3)\rangle \approx \prod_{d=D}^{D-D_3+1} U_d^\dagger |x\rangle. \quad (25)$$

Last, we add the remaining  $D_2$  layers and compute the remaining scalar product without approximation, using a contraction strategy analogous to that used for the calculation of the  $F^{(\tau)}$ :

$$\Psi_x \approx \langle \Psi'_Q(D_3) | \prod_{d=D_1+1}^{D_1+D_2} U_d | \Psi_Q(D_1) \rangle. \quad (26)$$

The last calculation is performed exactly at the same cost as a compression step for  $D_2 = K$ . It may be advantageous

to use  $D_1 > D_3$  and/or the corresponding bond dimensions  $\chi_1 > \chi_3$  if one wishes to calculate many different amplitudes  $\Psi_x$ . Indeed, the forward calculation needs to be done only once, while the backward and final calculations must be repeated for each bitstring  $x$ . On the other hand, if one seeks the best possible fidelity, one should increase  $D_2$  as much as possible in order to reduce the depth of the approximate parts of the calculation.

## V. DETAILS ON THE NUMERICAL EXPERIMENTS

### A. Three quantum circuits

In this article, we performed numerical experiments with three different quantum circuits, labeled sequences I, II, and III, as discussed in Sec. III. A schematic of the three sequences is shown in Figs. 6(a)–6(c), respectively.

Sequence I corresponds to the circuit of the quantum supremacy experiment, as shown in Fig. 6(a). It is designed to create a state as entangled as possible in as few steps as possible given the available nearest-neighbor connectivity. Each of the  $D$  layers (where  $D$  denotes the depth of the circuit;  $D = 20$  in the experiment) alternates between a one-qubit gate applied on all qubits (orange squares, drawn randomly from the  $\sqrt{X}$ ,  $\sqrt{Y}$ , and  $\sqrt{W}$  gates) and a two-qubit gate applied on a set of pairs of qubits, with four possible sets denoted by the letters  $A$ ,  $B$ ,  $C$ , and  $D$ , as shown in Fig. 1 ( $A$  is denoted with green,  $B$  with light blue,  $C$  with dark blue and  $D$  with red rectangles). The two-qubit gate is the  $\text{fsm}(\theta, \phi)$  gate defined in Ref. [3]. In the actual experiment, the values of  $\theta$  and  $\phi$  have been optimized for each pair of qubits to reach the best fidelity. Here we choose the constant values  $\theta = 1$ ,  $\phi = \pi/2$  that are close to the average experimental values, and deep in the difficult regime where the  $\text{fsm}$  gate has four different singular values. Our numerical results do not depend on this special choice. We also use the same sequence  $ABCD\text{-}CDAB\text{-}ABCD\text{-}\dots$  as Ref. [3] in the supremacy regime.

The alternative sequence II is a variation on the quantum supremacy sequence where we have changed the order of the gates applied. Sequence II is rotated by  $90^\circ$  compared to sequence I and reads  $CDDBA\text{-}BACD\text{-}CDDBA\text{-}\dots$ . This alternative sequence is just as useless but slightly “less random” than the supremacy sequence, since we have not designed it to be optimally random. This slight modification of the ordering has a strong impact of the fidelity found in the simulations.

Finally, we benchmarked our DMRG algorithm on a useful task, sequence III. Sequence III is actually not a specific sequence of gates but a protocol for generating circuits implementing the QAOA [13] for solving Max-Cut problems of combinatorial optimization. We generated many such problems for the  $N = 54$  “Sycamore” chip and selected instances where the associated QAOA circuit had



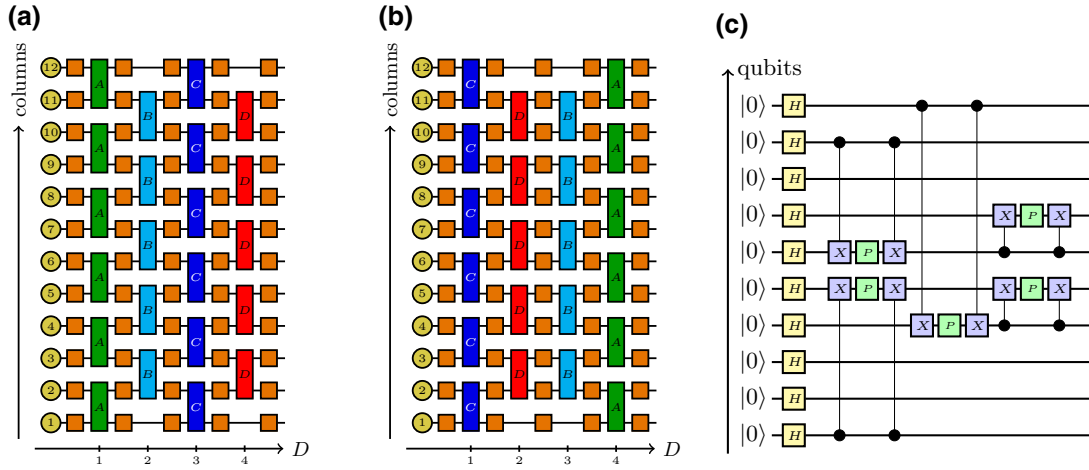


FIG. 6. The three sequences. Random quantum circuits defined by (a) sequence I and (b) sequence II. The circles represent columns of alternatively five and four qubits from the Sycamore chip, indexed from left to right. Orange squares correspond to applying a random one-qubit gate on each qubit (hence each column). Green, light blue, dark blue, and red rectangles correspond to applying two-qubit gates between all qubits coupled by the associated coupler, respectively  $A$ ,  $B$ ,  $C$ , and  $D$ . (c) Beginning of a ten-qubit circuit implementing the QAOA for a MaxCut problem (where  $P$  gates are parametrized phase shift gates).

a two-qubit gate count  $N_{2g} \approx 430$  similar to the gate count of sequences I and II. Figure 6(c) shows an example of such a circuit for a small number of qubits ( $N = 10$ ).

More specifically, we solve the MaxCut problem on Erdős-Rényi graphs  $\mathcal{G}(N, \mathcal{P})$ , a particular class of random graphs with  $N$  vertices and a probability  $\mathcal{P}$  for creating an edge between two vertices. The QAOA ansatz circuit [13] is of the form  $U = \prod_{k=1}^p U_B U_C$ , where  $U_B = \prod_{m=1, \dots, N} e^{-i\beta X_m}$  and  $U_C = \prod_{m,n \in E} e^{-i\gamma Z_m Z_n}$  with  $E$  the set of edges of the graph. Here, we are not interested in the result of the optimization itself. Hence, we set the variational parameters  $\beta$  and  $\gamma$  to random values. For the same reason, we set the number of QAOA layers  $p$  to 1. The edge density  $\mathcal{P}$  in the Erdős-Rényi graphs is adjusted so that the final two-qubit gate count is close to 430. We consider two different cases: without and with compilation. In the absence of compilation, the QAOA circuits do not necessarily comply with the grid connectivity of Sycamore (Fig. 1). To get about 430 gates, we pick  $\mathcal{P} = 32\%$ . In the second case, we compile the QAOA circuit to comply with the connectivity of Sycamore. The compilation uses SWAP insertion methods [36] to create a circuit that uses only the nearest-neighbor two-qubit gates available in Sycamore. As this procedure increases the depth of the circuit, we lower the edge probability  $\mathcal{P}$  down to 5% to keep the final two-qubit gate count to 430.

## B. Estimating the fidelity of a DMRG simulation

A very interesting feature of the DMRG algorithm is that the fidelity  $\mathcal{F} = |\langle \Psi_P | \Psi_Q \rangle|^2$  of the calculation can be easily estimated, even though we do not necessarily have access to the actual perfect state  $|\Psi_P\rangle$ . Inside a simulation,

we estimate the fidelity with

$$\tilde{\mathcal{F}} = \prod_{\delta} f_{\delta}, \quad (27)$$

where the partial fidelities  $f_{\delta}$  are the final fidelities of compression step  $\delta$ ,

$$f_{\delta} = |\langle \Psi_Q(\delta K + K) | U^{(D+K)} \dots U^{(D+1)} | \Psi_Q(D = \delta K) \rangle|^2, \quad (28)$$

at the end of the different optimization sweeps. The partial fidelity  $f_{\delta}$  is simply given by the norm of the last  $F^{(\tau)}$  tensor calculated during the compression step. It follows that our estimate of the error rate reads

$$\tilde{\epsilon} = 1 - \tilde{\mathcal{F}}^{1/N_{2g}}. \quad (29)$$

It was shown in Ref. [7] through a combination of analytical and numerical arguments that to very good approximation one has

$$\tilde{\mathcal{F}} \approx \mathcal{F}, \quad (30)$$

and the arguments and numerics given there remain valid for this article. However, since we have used a different compression algorithm here, we have performed an additional extensive numerical study of the validity of the multiplicative law [Eq. (30)] for up to a maximum of 36 qubits for which we can obtain the exact state  $|\Psi_P\rangle$ , and hence the exact fidelity  $\mathcal{F}$ . The multiplicative law (27) has a very important role, as it allows us to perform estimations of the fidelities in regimes where the exact calculation

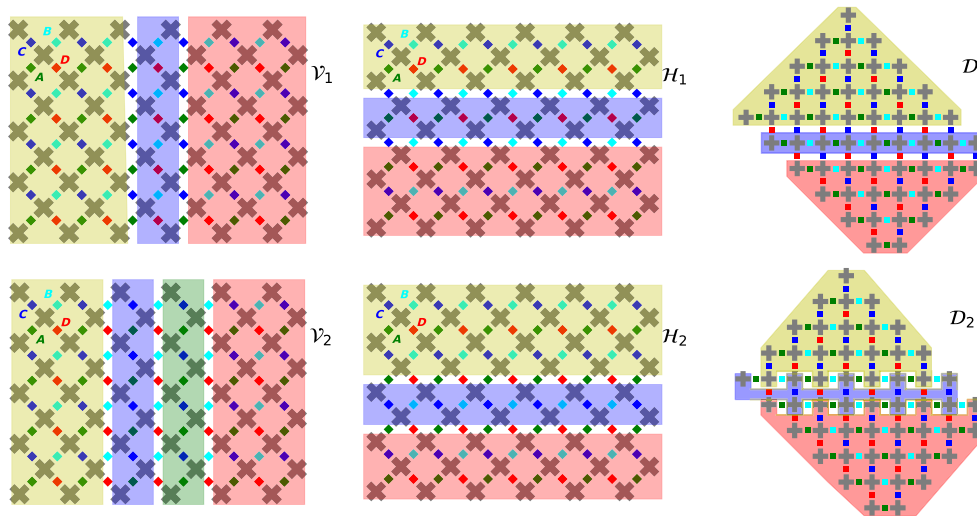


FIG. 7. Different qubit groupings used in our simulation. All qubits shaded with the same color (yellow, blue, green, or red) belong to the same tensor  $M^{(\tau)}$ .

is out of reach and only  $\tilde{\mathcal{F}}$  can be obtained. We find that Eq. (30) indeed holds in all regimes of interest for random circuits. In QAOA circuits,  $\tilde{F}$  becomes a lower bound of  $F$  (see Fig. 11 below), which means that the error rates we give are pessimistic estimates for the actual error rates we achieve.

Figure 10 below shows the comparison between the exact fidelity  $\mathcal{F}(D)$  (blue line) and our estimate  $\tilde{\mathcal{F}}(D)$  (dashed blue line) for a large choice of values of  $n_b$  and  $n_c$  up to  $N = 36$  qubits (beyond that, we do not have access to the exact state  $|\Psi_P\rangle$ ). We find a perfect match between the two curves in the relevant  $1 \geq \mathcal{F} \geq 1/2^N$  regime. For very large depth,  $\mathcal{F}$  saturates at an exponentially small value  $1/2^N$  while  $\tilde{\mathcal{F}}$  continues to decrease exponentially. The exponentially small asymptotic value  $1/2^N$  corresponds to the overlap between two independent Porter-Thomas chaotic vectors; see the derivation around Eq. (38). It is essentially the lowest fidelity that can be reached in a random circuit.

Interestingly, Eq. (27) provides an estimate of the error  $\epsilon_x$  even in the closed mode where we calculate a single amplitude  $\Psi_x$ . To check this assertion, we have computed the histogram of the error rates per gate in the closed mode for small systems where we could calculate all the amplitudes  $\Psi_x$  exactly. Typical results are shown in Fig. 8(b) below for three different configurations. We find that our estimated fidelity closely matches the actual value with a precision of a few percent (typically less than 5%) for *all the points in the histogram*. Note that in all regimes we have  $\tilde{\mathcal{F}} \leq \mathcal{F}$ , so that  $\tilde{\mathcal{F}}$  underestimates the actual fidelity.

### C. Different groupings of the qubits

Our DMRG algorithm gives us the freedom to define how we group the different qubits that correspond to each

tensor  $M^{(\tau)}$ ,  $1 \leq \tau \leq m$ . The different groupings that we have used are shown in Fig. 7, where each color corresponds to one tensor  $M^{(\tau)}$ . Note that using groupings is equivalent to using a maximum bond dimension that varies from tensor to tensor. The optimum choice depends strongly on the particular problem considered.

Different groupings may have some advantage depending on the actual circuit ran. The vertical groupings  $\mathcal{V}_1$  and  $\mathcal{V}_2$  group the qubits by columns. For instance,  $\mathcal{V}_1$  contains three tensors with respectively 5 (23), 2 (9), and 5 (22) columns (qubits). Layers  $B$  and  $D$  are “trivial” for grouping  $\mathcal{V}_1$ , i.e., they are internal to one tensor and hence have a perfect fidelity  $f_\delta = 1$  for any value of the bond dimension  $\chi$ . Likewise, layers  $A$  and  $C$  are trivial for grouping  $\mathcal{V}_2$ . For the larger systems of Fig. 2(c) with more than 54 qubits, we have used an extension of  $\mathcal{V}_1$  and  $\mathcal{V}_2$  by adding additional tensors of two columns to obtain  $N = 23 + 9(m - 2) + 22$  qubits with  $m$  tensors having respectively 23, 9, 9,  $\dots$ , 9 and 22 qubits. We find that the vertical groupings are optimum for sequence II.

Another possibility is to group the qubits horizontally in rows as in  $\mathcal{H}_1$  (for which  $A$  and  $D$  are trivial), or  $\mathcal{H}_2$  (for which  $B$  and  $C$  are trivial). Last, we may group the qubits diagonally as in  $\mathcal{D}_1$  and  $\mathcal{D}_2$ . In some calculations, we may try and alternate between two groupings, e.g.,  $\mathcal{D}_1$  and  $\mathcal{D}_2$  to optimize the number of trivial gates.

### D. Benchmark of the algorithm

Let us now see how the algorithm performs in practice. All the simulations are carried out on a Sycamore-like architecture with  $n_c$  columns where each column has  $n_b$  (for odd columns) or  $n_b - 1$  (for even columns) qubits, as shown in Fig. 1. The real Sycamore chip corresponds to 53 qubits arranged in  $n_c = 12$  columns with  $n_b = 5$  qubits in

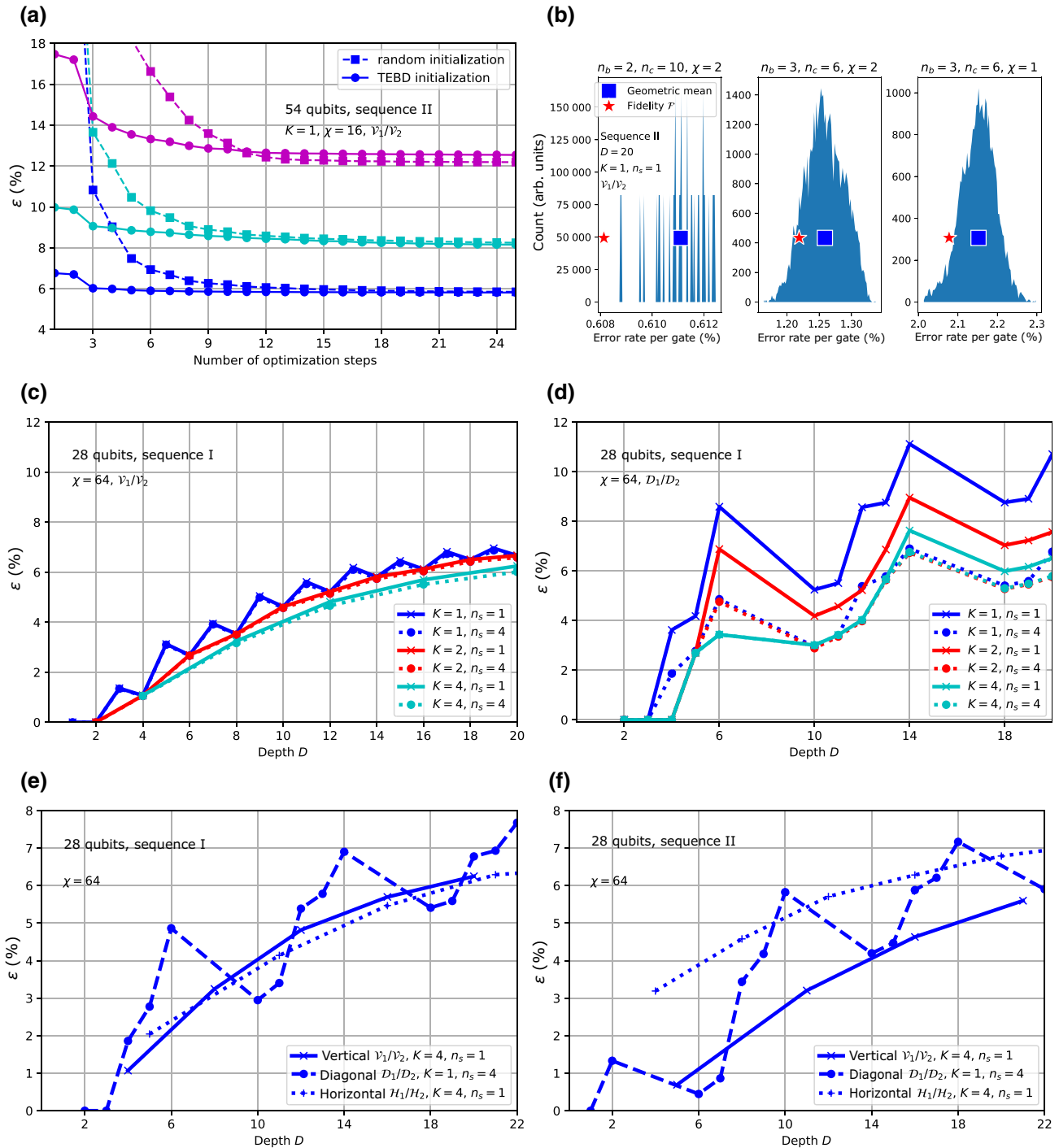


FIG. 8. Convergence of the DMRG algorithm. (a) Evolution of the error per gate versus the total number of local optimizations. Three steps correspond to a full sweep. Different colors correspond to different compression steps for a random (squares and dashed lines) and a TEBD (disks and solid lines) initialization. The different curves correspond to different compression steps in an open simulation [respectively the 4th (blue), 10th (cyan), and 12th (magenta) steps, with corresponding depths being  $D = 5, 13,$  and  $16$ ]. (b) Histogram of the error rates per gate  $\epsilon_x$  of the amplitudes of individual bitstrings  $x$  computed using closed simulation for various  $n_b, n_c,$  and  $\chi$ . Blue squares denote the geometric mean  $(\prod_x \epsilon_x)^{1/2^N}$ . Red stars denote the exact error per gate  $1 - \mathcal{F}^{2/(ND)}$ . (c),(d) Role of  $K$  and the number of sweeps  $n_s$ . Evolution of the error rate per gate  $\epsilon$  versus depth  $D$  for various numbers of layers  $K$  and numbers of sweeps. Parameters are  $\chi = 64, n_b = 4,$  and  $n_c = 8$ . (c) Vertical ordering: groupings  $\mathcal{V}_1, \mathcal{V}_2$ . (d) Diagonal ordering: groupings  $\mathcal{D}_1, \mathcal{D}_2$ . (e),(f) Role of the choice of qubit groupings and of the sequence of gates applied. Evolution of the error rate per gate as a function of current depth. (e) Sequence I, used in the Google supremacy experiment. (f) Sequence II, a variation on the supremacy sequence.

the first column. We also performed simulations on smaller systems where we could obtain the exact state (up to  $N = 35$  qubits) using Atos Quantum Learning Machine’s Schrödinger-style “qat-linalg” simulator.

### 1. Convergence of the DMRG compression step

Figure 8(a) shows the convergence of the optimized MPS during the DMRG sweeps. Each tensor is optimized separately one after the other. When all tensors have been optimized (one sweep), one repeats the procedure until convergence. Figure 8(a) shows how the error decreases as a function of the total number of tensors that have been optimized. More precisely, we plot the error per gate  $\epsilon = 1 - (f_\tau^{(k)})^{1/N_{2g}^{(k)}}$  as a function of the number of optimization steps, i.e., the number of evaluations of Eq. (20). Here  $N_{2g}^{(K)}$  is the number of two-qubit gates in the  $K$  newly added layers and  $f_\tau^{(k)}$  is the fidelity obtained upon optimizing tensor  $\tau$  in sweep  $k$ . Since the corresponding MPS contains  $m = 3$  tensors, three optimization steps correspond to one full sweep. Since each step provides a full optimization over one tensor, the error rates decrease monotonically, as expected.

Figure 8(a) shows two types of initialization of MPS  $|\Psi\rangle$  that is being optimized: either an arbitrary random MPS (squares and dashed lines) or an already partially optimized MPS obtained from the TEBD algorithm of Ref. [7] (disks and solid lines). By construction, the DMRG error can only be lower than the TEBD error. Unsurprisingly, we find that the convergence is much faster when starting from the TEBD initialization (typically 1–3 sweeps) than when starting from a random guess (typically 4–6 sweeps). However, the final error found shows weak dependence on the initial starting point [in Fig. 8(a) we show one case (pink curves) where there is a visible difference between the TEBD initialization and the random ones, but it is seldom observed]. We have also repeated the simulation with different random initializations and the error always converges to the same value. Since we optimize only one tensor at a time, we cannot dismiss the possibility of being trapped in a local minimum, but these observations indicate that the DMRG algorithm gets at least very close to the global optimum MPS. Note that this is the global optimum MPS for a given compression step. In Sec. VI, we discuss how the algorithm manages to track the global optimum for the full circuit after several compression steps.

### 2. Role of the number of layers per step $K$ and number of sweeps $n_s$

In most of the numerical experiments shown in this article, we have used the vertical ordering with  $K = 2$ ,  $n_s = 1$  or the diagonal ordering with  $K = 1$ ,  $n_s = 4$ . In the  $\mathcal{V}_1$  grouping, “even” layers of types  $B$  and  $D$  (see Fig. 7) can be absorbed trivially into the MPS without increasing the

bond dimension. It follows that the fidelity for  $K = 1$  and  $K = 2$  is actually the same. We have performed a few simulations in the  $\mathcal{V}_1$  grouping with  $K = 4$ . They present a small, typically 0.5% gain with respect to  $K = 2$  in return for a 16-fold increase in computing time. Note that this gain reflects our ability to *find* the MPS closest to  $|\Psi_P\rangle$ , not the ability of the MPS to capture the exact state. Indeed, the  $K = 2$  and  $K = 4$  calculations share the same bond dimension, and hence the same maximum level of possible entanglement.

Figure 8(c) shows the error rate  $\epsilon$  as a function of the depth  $D$  for different values of the number of layers  $K$  added inside the compression step, for the vertical grouping  $\mathcal{V}_1$ . The error rate for depth  $D \leq 2$  is zero as our MPS has a bond dimension large enough to accommodate the corresponding entanglement exactly. As one increases  $D$  further, one starts to see that the approximation and the error rate increase. As mentioned above, the error rates for  $K = 1$  and  $K = 2$  are identical with additional oscillations for the intermediate points for  $K = 1$ . This increase in the error rate for intermediate points for  $K = 1$  corresponds to the fact that these depths do not benefit from an upcoming trivial layer; hence, the average error rate is higher. For  $K = 4$ , we observe a small gain at large depths, but since the corresponding computational time increases significantly, we have not used  $K = 4$  in practice. Similar calculations for the diagonal grouping  $\mathcal{D}_1$  are shown in Fig. 8(d). The error rate shows oscillations due to the fact that, in this configuration, the  $D$  gate is very costly. Overall, we find that, for a large enough number of sweeps  $n_s$ , the  $K$  dependence of the error rate is small. This is already a strong indication that the algorithm provides a MPS not far from optimum, even though we are carrying out multiple compression steps. Increasing the number  $K$  of layers provides a small gain of  $< 1\%$  in the error rate at  $D = 20$ . However, the computational cost increases exponentially with  $K$ . Calculations with 54 qubits and  $K = 2$  are beyond the scope of the present article for the diagonal grouping.

### 3. Role of the qubit grouping

Figures 8(e) and 8(f) show the error rate versus depth for three different groupings: vertical, diagonal, and horizontal. We observe important variations of the error rate with the grouping as well as with the circuit (sequence I versus sequence II). Note that, for this small system of 28 qubits ( $n_b = 4$ ,  $n_c = 8$ ), sequence II is only marginally easier than sequence I because the system is almost like a square (as opposed to a rectangle for the Sycamore case  $n_b = 5$ ,  $n_c = 12$ ). This difference between different groupings is in itself not surprising: different circuits tend to entangle some qubits more than others. Since the choice of the grouping amounts to choosing the position of the “entanglement bottleneck,” there must be an optimum grouping for each circuit.



## VI. DOES THE DMRG ALGORITHM PROVIDE THE OPTIMAL MPS?

In this section, we discuss whether the MPS obtained by the DMRG algorithm corresponds to the best possible MPS. Indeed, two possible causes may prevent the final MPS obtained from being optimal: the fact that the optimization is broken into different compression steps, and the fact that within a compression step the optimization is performed tensor by tensor, not globally. We analyze this problem in the context of the random circuit of sequence II.

To assess this point, Fig. 9 shows three different errors  $\epsilon$  versus depth  $D$  curves.

- The blue squares (continuous line) show the error  $\tilde{\epsilon}$  obtained within the DMRG algorithm.
- The blue stars (dashed line) show the error  $\epsilon_{\text{SVD}}$  corresponding to best possible approximation of the exact state with a MPS, as explained below. This reference curve can only be obtained for a small number of qubits and is not available in general.
- The black dash-dot line shows the error  $\epsilon_{\text{opt}}$ , the best possible approximation of a purely chaotic state with a MPS, as given by

$$\epsilon_{\text{opt}} = \frac{1}{D} \left( \log 2 - \frac{\log 4\chi}{2N} \right) \quad (31)$$

(see the analytical derivation in Appendix B). We refer to Eq. (31) as the ‘‘chaotic optimum error.’’ The fact that the chaotic optimum error decreases with  $D$  stems from the simple fact that the best *fidelity* that one may obtain when approximating a chaotic state with a MPS is a finite number  $\mathcal{F}_{\text{opt}} = 4\chi/2^{N/2}$ , so that the error per gate must decrease.

### A. Best possible MPS calculation

To obtain the blue star (dashed line) ‘‘best possible MPS’’ curve of Fig. 9, we perform an exact ‘‘Schrödinger-like’’ simulation of the small 28-qubit system and obtain the exact state  $|\Psi_P(D)\rangle$ .

In a second step, we split the qubits into two groups  $A$  and  $B$  of equal size, so that  $|\Psi_P(D)\rangle$  reads

$$|\Psi_P(D)\rangle = \sum_{\alpha\beta} \Psi_{\alpha\beta}(D) |\alpha\rangle_A |\beta\rangle_B, \quad (32)$$

where state  $|\alpha\rangle_A$  ( $|\beta\rangle_B$ ) forms an orthonormal basis of  $A$  ( $B$ ). We perform a singular value decomposition  $\Psi = USV^\dagger$  of the  $2^{N/2} \times 2^{N/2}$  matrix,  $\Psi_{ab}$ , from which we get

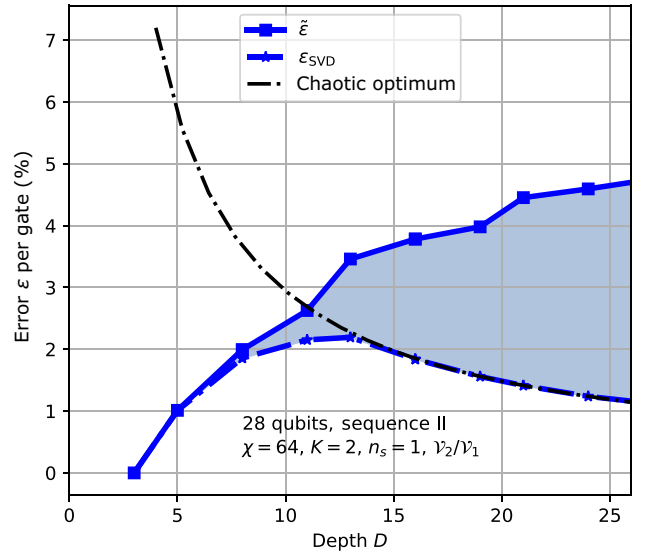


FIG. 9. Comparison of the error per gate to the SVD error as a function of depth  $D$ . Vertical qubit grouping  $\mathcal{V}_2$ . The chaotic optimum is Eq. (31).

the Schmidt decomposition of  $|\Psi\rangle$ :

$$|\Psi\rangle = \sum_{\mu} S_{\mu} |\mu\rangle_A |\mu\rangle_B \quad (33)$$

with  $|\mu\rangle_A = \sum_{\alpha} U_{\alpha\mu} |\alpha\rangle_A$  (and a similar expression for  $|\mu\rangle_B$ ). Sorting the singular values  $S_{\mu}$  in decreasing order, we can obtain the best approximate MPS by truncating the above expression and keeping the  $\chi$  largest singular values.

### B. Analytical calculation of the chaotic optimum (31)

The computation of the chaotic optimum (31) follows the same procedure as for the ‘‘best possible MPS’’ discussed above with a small modification: instead of starting with the exact state  $|\Psi_P(D)\rangle$ , we start with a fully chaotic state  $|\Psi\rangle$  distributed according to the Porter-Thomas distribution, i.e.,

$$|\Psi\rangle = \sum_{x=0}^{\mathcal{N}-1} \Psi_x |x\rangle, \quad (34)$$

where  $\mathcal{N} = 2^N$  and the Porter-Thomas vector  $\Psi_x$  corresponds to one column of a unitary matrix distributed according to the Haar (uniform) measure of  $U(\mathcal{N})$ .

The derivation of the ‘‘chaotic optimum’’ error formula follows from the properties of the singular values of random Gaussian matrices. It is performed in Appendix B.

### C. Numerical results

We find in Fig. 9 that the best possible error  $\epsilon_{\text{SVD}}$  first increases as the system gets more and more entangled,

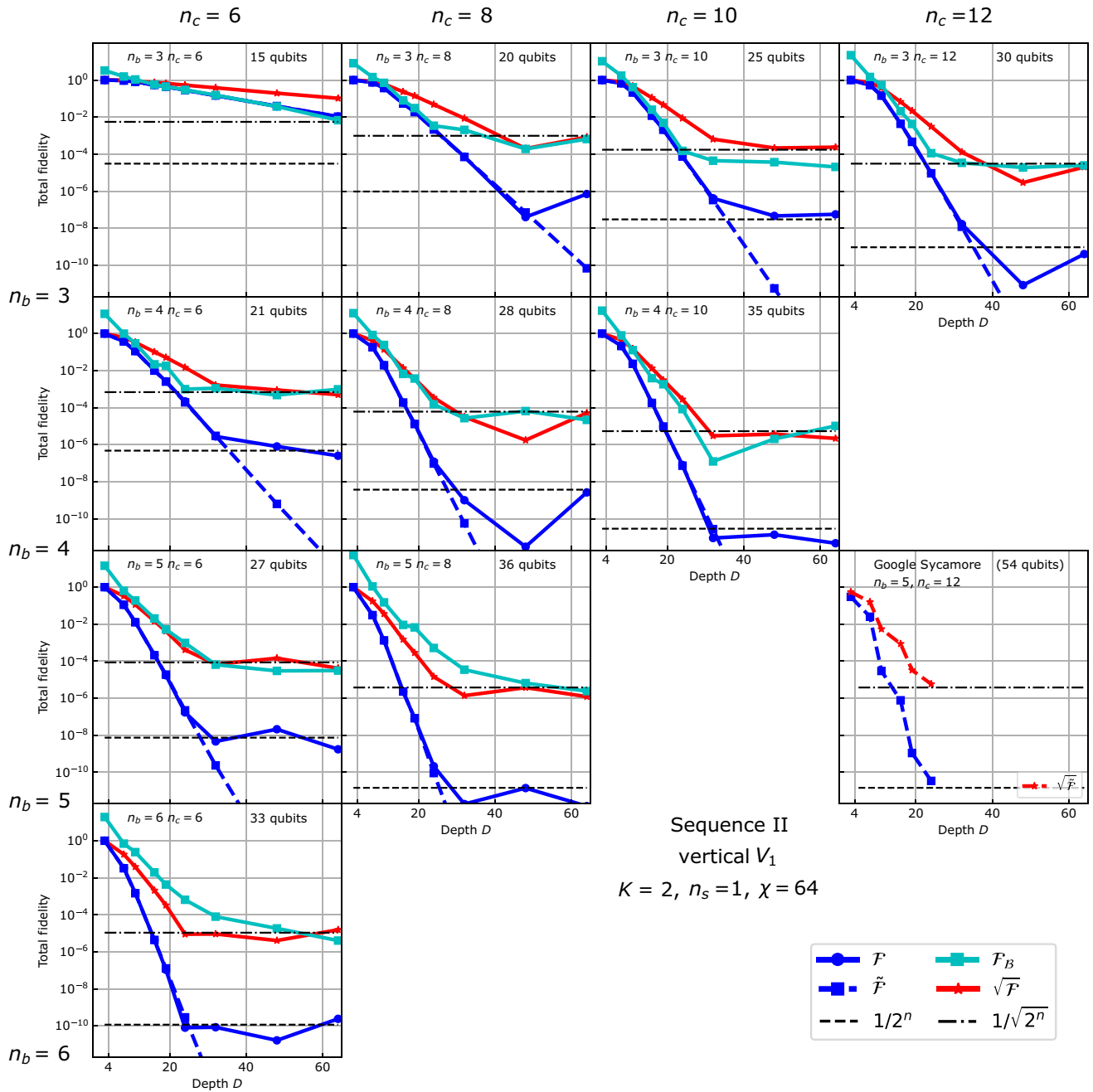


FIG. 10. Comparison of various fidelities. Fidelities as a function of total circuit depth  $D$ , at fixed bond dimension  $\chi = 64$ . Exact fidelity  $\mathcal{F}$  [Eq. (12)], cross-entropy benchmarking fidelity  $\mathcal{F}_B$  [Eq. (3)], product fidelity  $\tilde{\mathcal{F}}$  [Eq. (27)]. Open simulation mode and  $K = 2$  layers.

reaches a maximum at  $D \approx 12$ , and then starts to decrease following closely the chaotic optimum. The maximum error at  $D = 12$  hence corresponds to the depth beyond which the state of the system is chaotic.

In contrast, in our DMRG simulations the error  $\tilde{\epsilon}$  can, by construction, only increase. It eventually saturates to a finite value when the MPS tensors become random (see an in-depth discussion of this last point in Ref. [7]). Hence, it must deviate from the best possible approximation  $\epsilon_{\text{SVD}}$

at some point. Figure 9 shows that this deviation appears around  $D = 12$  when the system becomes chaotic. At small depths, the DMRG results are indeed very close to the best possible approximation. From the Fig. 9 data, we conjecture that the intersection between the DMRG error and the chaotic optimum can be used to estimate when the quantum state becomes chaotic. Before one reaches the chaotic regime, the DMRG algorithm is close to optimum. Conversely, Fig. 9 can be seen as a strong indication that

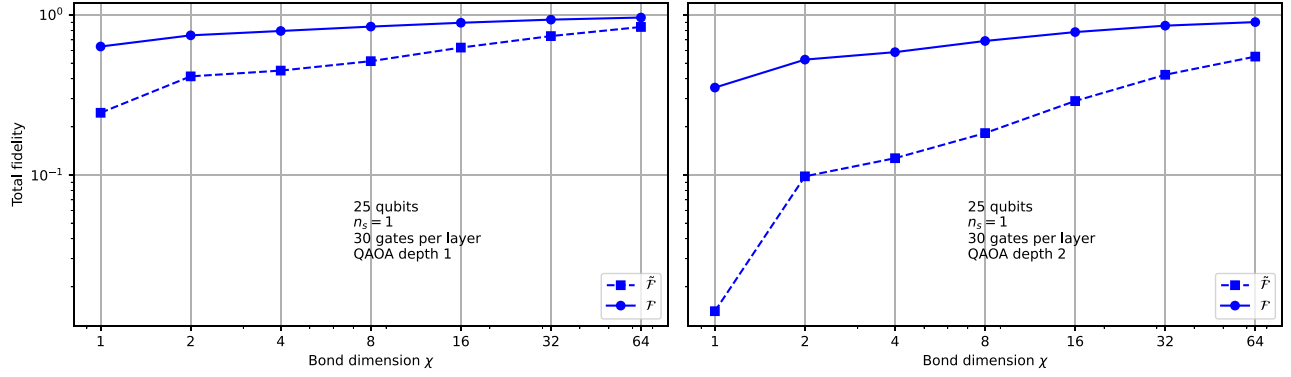


FIG. 11. Comparison of various fidelities for the QAOA circuit. Fidelities as a function of bond dimension  $\chi$ , at fixed circuit depth [ $p = 1$  (left) and  $p = 2$  (right)]. Exact fidelity  $\mathcal{F}$  [Eq. (12)] and product fidelity  $\tilde{\mathcal{F}}$  [Eq. (27)]. Open simulation mode.

the DMRG algorithm will perform significantly better for nonchaotic states than for chaotic ones.

## VII. RELATION BETWEEN FIDELITY AND CROSS-ENTROPY BENCHMARKING

The relation between the actual fidelity  $\mathcal{F}$  and the cross-entropy benchmarking  $\mathcal{F}_B$  is far from trivial. Arute *et al.* [3] have argued that *in their experiment* and *in the large-depth chaotic regime* both quantities are equivalent,  $\mathcal{F}_B \approx \mathcal{F}$ ; see also Ref. [7] for a discussion.

Here we show that *in our numerics* we have a very different relation

$$\mathcal{F}_B \approx \sqrt{\mathcal{F}}, \quad (35)$$

which implies that  $\mathcal{F}_B \gg \mathcal{F}$  (since  $\mathcal{F} \ll 1$ ). We give two bodies of evidence for Eq. (35).

- (a) A large body of numerical calculations for systems up to  $N = 36$  qubits where we can simulate the exact state  $|\Psi_P\rangle$ , and hence calculate both the left- and right-hand sides of Eq. (35)
- (b) An analytical calculation in the very large  $D$  limit. In this limit,  $|\Psi_P\rangle$  and  $|\Psi_Q\rangle$  converge to two independent random chaotic states and one can calculate the two fidelities exactly and show that Eq. (35) holds. It follows that the type of errors present in an experiment plays an important role in knowing which relation holds: assuming that one only makes precision errors experimentally (such as over rotations during a gate), one would retain a pure state at large depth and Eq. (35) would hold.

These two bodies of evidence are presented in the rest of this section.

### A. Numerical evidence for $\mathcal{F}_B \approx \sqrt{\mathcal{F}}$

Figure 10 shows  $\mathcal{F}$  (blue) and  $\mathcal{F}_B$  (green) for several values of  $n_b$  and  $n_c$  for which  $N \leq 36$ , so that the exact

state could be obtained using the large RAM of the Atos Quantum Learning Machine. Also shown is  $\sqrt{\mathcal{F}}$  (red), which is close to the green curve as well as the two exact asymptotic values  $1/2^N$  for  $\mathcal{F}$  and  $1/2^{N/2}$  for  $\mathcal{F}_B$ . As one gets closer to the Sycamore chip regime ( $n_b = 5$ ,  $n_c = 12$ ), for which no exact statement can be made, we find that the relation  $\mathcal{F}_B \approx \sqrt{\mathcal{F}}$  becomes increasingly valid.

In addition to these checks for random circuits, we also check, in Fig. 11, that the use of  $\tilde{\mathcal{F}}$  as a proxy for  $\mathcal{F}$  is justified. In random circuits, some of us proved that  $\mathcal{F} \approx \tilde{\mathcal{F}}$  [7] (this can be checked in Fig. 10). In QAOA circuits, we observe that  $\tilde{\mathcal{F}}$  is a lower bound for  $\mathcal{F}$ :  $\mathcal{F} \geq \tilde{\mathcal{F}}$ . Thus, the error rates we obtain (which are based on  $\tilde{\mathcal{F}}$ ) are pessimistic estimates for the actual error rates we achieve (which would correspond to  $\mathcal{F}$ , which however cannot be estimated efficiently).

### B. Fidelity and cross-entropy benchmarking in the chaotic limit

In this subsection, we calculate  $\mathcal{F}$  and  $\mathcal{F}_B$  analytically in the  $D \rightarrow \infty$  limit. In this limit, the two states  $|\Psi_P\rangle$  and  $|\Psi_Q\rangle$  become essentially independent and distributed according to a Porter-Thomas distribution. Let us define  $\mathcal{N} = 2^N$  and denote by  $U$  and  $V$  two  $\mathcal{N} \times \mathcal{N}$  matrices distributed according to the Haar (uniform) measure of the  $U(\mathcal{N})$  group. With this notation, the two states are, for very large depths, the first column of matrices  $U$  and  $V$ :  $\langle x|\Psi_P\rangle \equiv U_{x1}$  and  $\langle x|\Psi_Q\rangle \equiv V_{x1}$ . We further denote by  $\langle X \rangle = \int X dU dV$  the average over these ensembles. We make use of two integrals that can be found in Ref. [37]: for any matrices  $A, B, C$ , and  $D$ , one has

$$\int dU \text{Tr} A U B U^\dagger = \frac{1}{\mathcal{N}} \text{Tr} A \text{Tr} B \quad (36)$$

and

$$\begin{aligned} & \int dU \text{Tr} AUBUCU^\dagger DU^\dagger \\ &= \frac{1}{\mathcal{N}^2 - 1} \left[ \text{Tr} A \text{Tr} BD \text{Tr} C + \text{Tr} ADCB \right. \\ & \quad \left. - \frac{1}{\mathcal{N}} \text{Tr} A \text{Tr} BDC - \frac{1}{\mathcal{N}} \text{Tr} ADB \text{Tr} C \right]. \end{aligned} \quad (37)$$

Using the first of these two integrals, we obtain

$$\langle \mathcal{F} \rangle = \int dU dV \sum_{x, x'} U_{x1} U_{1x'}^\dagger V_{x1} V_{1x'}^\dagger = \frac{1}{\mathcal{N}} \quad (38)$$

and

$$\langle \mathcal{F}_B \rangle = \mathcal{N} \int dU dV \sum_x U_{x1} U_{1x}^\dagger V_{x1} V_{1x}^\dagger - 1 = 0. \quad (39)$$

Since  $\langle \mathcal{F}_B \rangle = 0$  vanishes in average, we need to calculate its variance to estimate its typical value. We get

$$\begin{aligned} \langle (\mathcal{F}_B)^2 \rangle &= \mathcal{N}^2 \int dU dV \sum_{xx'} \\ & \quad U_{x1} U_{x'1} U_{1x}^\dagger U_{1x'}^\dagger V_{x1} V_{x'1} V_{1x}^\dagger V_{1x'}^\dagger - 1. \end{aligned} \quad (40)$$

After some straightforward algebra, we get

$$\langle (\mathcal{F}_B)^2 \rangle = \frac{\mathcal{N} - 1}{(\mathcal{N} + 1)^2}. \quad (41)$$

It follows that at very large depths,  $\mathcal{F}_B \approx 1/\sqrt{\mathcal{N}} \approx \sqrt{\mathcal{F}}$ . Figure 12 shows numerical calculations of  $\mathcal{F}$  and  $\mathcal{F}_B$  for two independent Porter-Thomas vectors together with the analytical expressions derived above.

## VIII. CONCLUSIONS

We have presented an algorithm to efficiently simulate quantum circuits with finite fidelity. This algorithm extends Ref. [7], where some of us adapted the TEBD technique from many-body physics to the context of quantum circuits. Here we have introduced a generalization of the DMRG algorithm to quantum circuits. This new algorithm also has a simulation cost that scales polynomially with the number of qubits  $N$  and the depth of the circuit  $D$ . In addition, it is more general and more efficient than the previous TEBD-like algorithm. From the simulation point of view, we emphasize that the main building block of our DMRG algorithm, the ‘‘compression step,’’ is completely general and could be used in other contexts. In particular, it may be combined with other tensor network techniques such as slicing or contraction heuristics as in Refs. [4,5]. Another

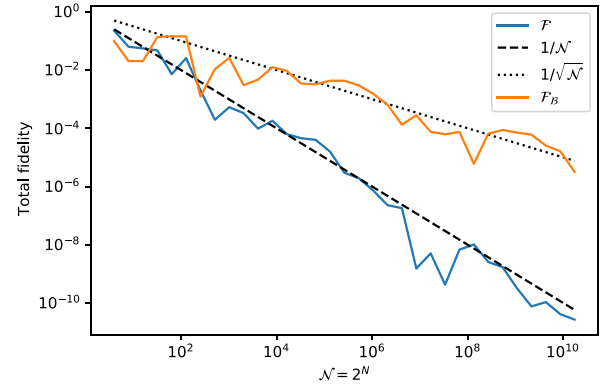


FIG. 12. Various fidelities of random states. Fidelity  $\mathcal{F} = |\langle \Psi_P | \Psi_Q \rangle|^2$  and cross-entropy benchmarking  $\mathcal{F}_B = \mathcal{N} \sum_x |\langle x | \Psi_P \rangle|^2 |\langle x | \Psi_Q \rangle|^2 - 1$  for two independent random vectors  $|\Psi_P\rangle$  and  $|\Psi_Q\rangle$  distributed according to the Porter-Thomas distribution.

direction would be to implement the DMRG algorithm for tensor networks different from a simple MPS such as a tree network or a two-dimensional lattice of tensors (PEPSs). These generalizations have not been attempted yet.

We have benchmarked our algorithm on the supremacy sequence designed by Google and found that we can produce amplitudes (hence bitstrings) of quality as good as in the quantum supremacy experiment [3]. More importantly, for QAOA sequences, representative of useful applications of quantum computers, we obtain much better fidelities than the supremacy threshold set by Google. Our results provide strong evidence that quantum advantage (the ability for a quantum computer to perform a useful task better than a classical computer) might be much harder than reaching quantum supremacy (the ability for a quantum computer to perform a given, not necessarily useful, task that no classical computer can perform), despite what the words seem to indicate. In particular, since our algorithm scales polynomially with the number of qubits, an improvement in the experimental fidelity is needed in order for the experiments to reach better results than the DMRG algorithm for useful tasks.

Our work emphasizes the need for benchmarks of quantum computers that test the actual usefulness of quantum processors, rather than their ability to perform a relatively contrived task, in order to incentivize hardware and software efforts towards concrete applications. Such benchmarks should strive to be application centric, hardware agnostic, and scalable. Some of us recently proposed a protocol fulfilling these criteria [38].

## ACKNOWLEDGMENTS

We thank Giuseppe Carleo, Lei Wang, and Pan Zhang for helpful feedback about prior work in simulating quantum circuits and for pointing out several errors in the



literature discussion. X.W. acknowledges funding from the French ANR QCONTROL as well as the Plan France 2030 ANR-22-PETQ-0007. T.L., X.W., and T.A. acknowledge funding from ANR QPEG. X.W. acknowledges funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No. 862683 (UltraFastNano). E.M.S. acknowledges helpful discussions with Soonwon Choi on the topic of cross-entropy benchmarking and fidelity. The Flatiron Institute is a division of the Simons Foundation. The computations were performed on the Atos Quantum Learning Machine.

## APPENDIX A: A SHORT INTRODUCTION TO TENSOR NETWORKS FOR SIMULATING QUANTUM CIRCUITS

In this appendix, we briefly review the main aspects of tensor networks in the context of quantum circuit simulation.

### 1. Basic definitions and actions: contracting and splitting

A tensor is simply an array of complex numbers that generalizes the concepts of vectors (1D array) and matrices (2D arrays) to an arbitrary number of indices. A tensor  $V_i$  with one index  $i$  (that takes a finite number of values) is simply a vector; a tensor with two indices  $A_{ij}$  is a matrix; a tensor  $M_{ijk}$  ( $P_{ijkl}$ ) is a 3D (4D) array of numbers. A tensor is represented graphically by a box (here a rectangle or a circle) with as many legs (outgoing lines) as there are indices; see the examples in Fig. 13.

There are two basic operations that one can do with tensors: contracting and splitting. Contractions of two tensors is the generalization of matrix-matrix multiplication. An example is shown in Fig. 13(a) for the contraction of  $M_{abj}$  with  $G_{ij}$ . The resulting tensor  $M'_{abi}$  is simply given by

$$M'_{abi} = \sum_j G_{ij} M_{abj}. \quad (\text{A1})$$

In other words, one performs a summation over the index  $j$  that links the two tensors.

The second operation, splitting of, e.g., a tensor  $U_{\alpha\beta\alpha'\beta'}$ , is illustrated in Fig. 13(b). It consists of three steps. First, one constructs two metaindices  $i$  and  $j$  that group several indices together. For instance, one may choose  $i = \alpha + N_\alpha\alpha'$  and  $j = \beta + N_\beta\beta'$ , where  $N_\alpha$  and  $N_\beta$  are the numbers of different values that the indices  $\alpha$  and  $\beta$  take, respectively. This allows us to define a one-to-one mapping between the tensor  $U_{\alpha\beta\alpha'\beta'}$  and a *matrix*  $\hat{U}$  defined as

$$\hat{U}_{ij} \equiv U_{\alpha(i)\beta(j)\alpha'(i)\beta'(j)}. \quad (\text{A2})$$

Second, we may use any result known from linear algebra on matrix  $\hat{U}$ , for instance, a QR decomposition, a SVD

decomposition, or any other decomposition. Let us suppose that we use a QR decomposition and write  $\hat{U} = \hat{Q}\hat{R}$ . Third, we use mapping (A2) backward to go back to the original indices and obtain

$$U_{\alpha\beta\alpha'\beta'} = \sum_a Q_{\alpha\alpha'a} R_{a\beta\beta'}, \quad (\text{A3})$$

i.e., we have split the tensor  $U_{\alpha\beta\alpha'\beta'}$  into terms of the contraction of two tensors  $Q_{\alpha\alpha'a}$  and  $R_{a\beta\beta'}$ .

### 2. Tensor networks for quantum circuits

There is a natural correspondence between the usual representation for quantum circuits and tensor networks. The left-hand side of Fig. 13(c) shows a small quantum circuit for four qubits that uses the standard Hadamard gate  $H$ , the controlled-NOT gate  $C^X$ , and the controlled-Z gate  $C^Z$ . The system wave function  $\Psi_{i_1 i_2 i_3 i_4}$  is a tensor whose explicit form is given by

$$\Psi_{i_1 i_2 i_3 i_4} = \sum_{i'_1 i'_2 i'_3 i'_4 i''_1 i''_2 i''_3 i''_4} C_{i_1 i_2 i'_1 i'_2}^X C_{i_3 i_4 i'_3 i'_4}^X C_{i'_2 i'_3 i''_2 i''_3}^Z H_{i'_1 0} H_{i'_2 0} H_{i'_3 0} H_{i'_4 0}, \quad (\text{A4})$$

i.e., it corresponds to the contraction of the tensor network shown on the right-hand side of Fig. 13(c). The problem of computing the wave function is reduced to the problem of performing the summation over the internal indices, i.e., the contraction of the tensor network [39]. Finding the best order to perform the contraction is in general a difficult (NP hard) problem for which there nevertheless exist good heuristics.

In order to perform the contraction of such a tensor network on a large computer that contains many cores, a common strategy is known as “slicing.” One selects a few indices and “freezes” them. In the above example, one could, for instance, decide to slice on indexes  $i'_2$  and  $i'_3$ . In practice, this means that each computing core is given a different value of ( $i'_2, i'_3$ ) and we compute the sum over all the other (nonfrozen) indices. At the end, the results from the different computing cores are simply added together. The advantage of the slicing approach is that it is embarrassingly parallel (the tasks given to the different cores are independent; they do not need to communicate); hence, one obtains an optimum speed up. The second advantage is that it lowers the memory footprint per core exponentially if the frozen indices are chosen correctly.

### 3. Schrödinger versus Schrödinger-Feynman-like simulations

There are several possible different strategies to contract the tensor network associated with a quantum circuit. Figure 14 shows two examples for the Schrödinger and the Schrödinger-Feynman approaches discussed in Sec. II. In

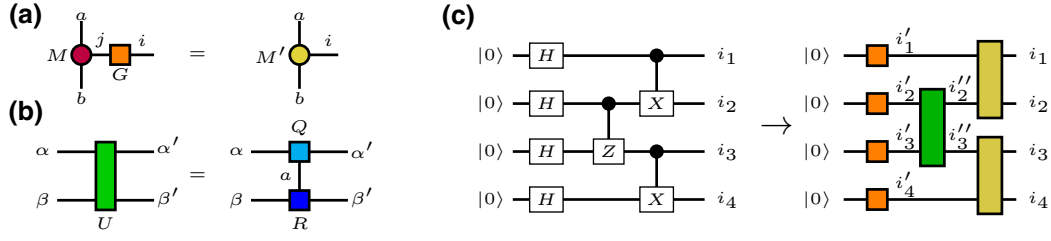


FIG. 13. Tensor networks and quantum circuits. (a) Contraction of a three-index tensor  $M$  with a one-qubit gate  $G$ . (b) QR decomposition of a two-qubit gate  $U$ . (c) From a quantum circuit to its tensor network representation.

the Schrödinger approach, the contraction of the network is performed from left to right, as shown in Fig. 14(b). In the Schrödinger-Feynman approach, shown in Fig. 14(c), one divides the qubits into two groups and splits the two-qubit gates that connect the two groups. For given values  $\alpha, \beta, \gamma$  of the indices cut by the dotted line, one may propagate the two substates  $\Psi_1^{(\alpha, \beta, \gamma)}$  and  $\Psi_2^{(\alpha, \beta, \gamma)}$  independently; see Eqs. (9) and (10) in the main text. Thus, instead of one complex simulation of the whole circuit, we perform  $\chi_p$  easier simulations, where  $\chi_p$  is the number of values taken by the extra bond indices.

#### 4. Pointers to the literature

The introduction above barely scratches the surface of the vast literature devoted to tensor networks and quantum state compression. Readers interested in the topic may find the following references useful. Eisert *et al.* [40] explained why, in many situations of physical interest, the entanglement is expected to be small; hence, the corresponding state is amenable to compression. Ref. [12] contains a pedagogical discussion of Matrix Product States (MPS) and their operator equivalent matrix product operators. Another classical introduction is Ref. [41]. A review of MPS-based time evolution methods can be found in Ref. [42]. The present work can be considered as an extension of these time-dependent techniques to discrete quantum circuits.

#### APPENDIX B: DERIVATION OF THE CHAOTIC OPTIMUM ERROR [EQ. (31)]

In this appendix, we prove that, for a chaotic state  $|\Psi\rangle$  distributed according to the Porter-Thomas distribution, the best possible fidelity that one may obtain by approximating it with a  $m = 2$  MPS is

$$\mathcal{F}_{\text{opt}} = \frac{4\chi}{2^{N/2}}. \quad (\text{B1})$$

To establish this result, the wave function  $\Psi_x$  is considered as a matrix  $\Psi_{\alpha\beta}$ , where index  $\alpha$  labels half of the qubits and  $\beta$  labels the other half. Performing a singular value decomposition of the  $2^{N/2} \times 2^{N/2}$  matrix  $\Psi_{\alpha\beta}$  to obtain its

singular values  $S_\mu$ , the fidelity for a bond dimension  $\chi$  is given by the largest  $\chi$  singular values

$$\mathcal{F}_{\text{opt}} = \sum_{\mu=1}^{\chi} S_\mu^2. \quad (\text{B2})$$

The proof consists of two parts:

- (a) establish that the matrix  $\Psi_{\alpha\beta}$  is, in the large- $(\mathcal{N} = 2^N)$  limit, a complex Gaussian random matrix;
- (b) use known results from random matrix theory to obtain the distribution of singular values from which one can obtain Eq. (B1) after a little algebra.

We perform these tasks below.

#### 1. Construction of a Porter-Thomas state from random Gaussian variables

We want to establish that a Porter-Thomas state can be constructed from random Gaussian variables. We recall that the sum  $S$  of the squares of  $k$  random normal variables  $X_i \sim \mathcal{N}(0, \sigma^2)$  follows a (generalized)  $\chi^2$  distribution with mean and variance

$$\mathbb{E}(S) = k/\sigma^2, \quad (\text{B3})$$

$$\text{Var}(S) = \frac{2k}{\sigma^4}. \quad (\text{B4})$$

Its probability density function is

$$P(s) = \frac{1}{2^{k/2} \Gamma(k/2) \sigma^2} \left( \frac{s}{\sigma^2} \right)^{k/2-1} e^{-s/(2\sigma^2)}. \quad (\text{B5})$$

Let us construct a wave function  $\Psi$  with  $2^N$  complex amplitudes

$$\Psi_x = \psi'_x + i\psi''_x, \quad (\text{B6})$$

and choose the real and imaginary components to be normally distributed:

$$\psi'_x \sim \mathcal{N}(0, 1/(2 \cdot 2^N)), \quad (\text{B7})$$

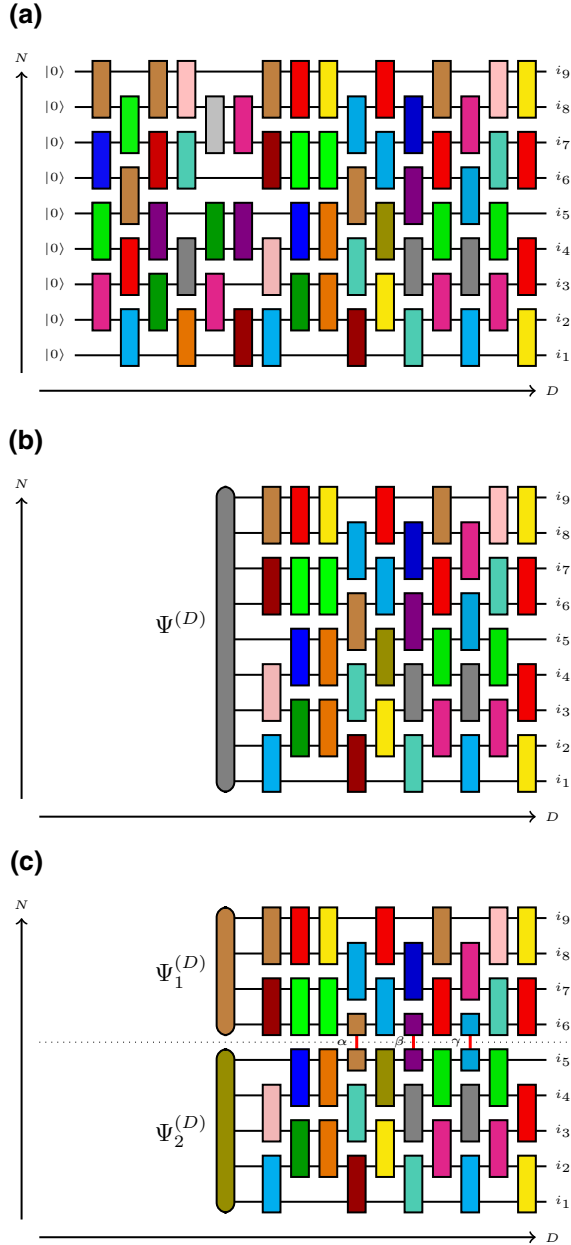


FIG. 14. Schematic of two contraction strategies for simulating a quantum circuit. (a) Quantum circuit to contract. (b) Schrödinger-like simulation. (c) Schrödinger-Feynman-like simulation.

$$\psi_x'' \sim \mathcal{N}(0, 1/(2 \cdot 2^N)). \quad (\text{B8})$$

Let us first consider the probability

$$p_x = |\Psi_x|^2 = (\psi_x')^2 + (\psi_x'')^2. \quad (\text{B9})$$

This random variable is a sum of normal variables. Applying Eq. (B5) with  $k = 2$  and  $\sigma^2 = 1/(2 \cdot 2^N)$ , we find

that

$$P(p_x) = 2^N e^{-2^N p_x}, \quad (\text{B10})$$

i.e., the Porter-Thomas distribution, as expected.

Let us now check that  $\Psi$  is normalized in the large- $N$  limit. Let us consider its norm

$$\|\Psi\|^2 = \sum_x |\Psi_x|^2 = \sum_x (\psi_x')^2 + (\psi_x'')^2. \quad (\text{B11})$$

This random variable is also a sum of normal variables. We can apply Eqs. (B3) and (B4) with  $k = 2 \cdot 2^N$  and  $\sigma^2 = 1/(2 \cdot 2^N)$ . We find that

$$\mathbb{E}(\|\Psi\|^2) = 1, \quad (\text{B12})$$

$$\text{Var}(\|\Psi\|^2) = \frac{1}{2^N}. \quad (\text{B13})$$

We have constructed a wave function whose norm is 1 on average, with a deviation to the average that vanishes exponentially fast as the number of qubits  $n$  increases.

As a result, matrix  $\Psi_{\alpha\beta}$  is a random complex Gaussian matrix, up to exponentially small corrections. Let us note that the Gaussian probability distribution that we have used,

$$P(\Psi) \propto \exp\left(-2^N \sum_x |\Psi_x|^2\right), \quad (\text{B14})$$

obviously respects the Haar invariance

$$P(\Psi) \prod_x d\Psi_x' \Psi_x'' = P(\tilde{\Psi}) \prod_x d\tilde{\Psi}_x' \tilde{\Psi}_x'' \quad (\text{B15})$$

for any unitary rotation  $\tilde{\Psi} = U\Psi$  with  $UU^\dagger = 1$  and for all  $N$ . It is only the constraint  $\|\Psi\| = 1$  that is enforced only in average with an exponentially small variance.

## 2. Scaling law for the singular values of a Porter-Thomas state

In this subsection, our goal is to understand how the Schmidt coefficients  $S_\mu$  of the Porter-Thomas state constructed in the previous subsection decrease as a function of the index  $\mu$ : a fast decrease will be synonymous of a high MPS quality.

For this purpose, we make use of known results from random matrix theory. More specifically, we use the fact that, in the large- $N$  limit, the average density  $\rho(S) \equiv 1/2^{N/2} \sum_{\mu=1}^{2^{N/2}} \delta(S - S_\mu)$  of the singular values  $S_\mu$  of a random complex  $2^{N/2} \times 2^{N/2}$  Gaussian matrix [with matrix elements  $\Psi_{\alpha,\beta} \sim \mathcal{N}(0, \sigma^2) + j\mathcal{N}(0, \sigma^2)$ ,  $\sigma^2 = 1/(2 \cdot 2^N)$ ,

as discussed in the previous subsection] follows a quadrant law (see, e.g., Ref. [43]):

$$\lim_{N \rightarrow \infty} 1/2^{N/2} \rho(s/2^{N/2}) = \frac{1}{\pi} \sqrt{4 - s^2}, \quad s \in [0, 2]. \quad (\text{B16})$$

In the large- $N$  limit, the number  $\mu$  of singular values above a given threshold  $S_0$  is given by

$$\mu(S_0) = 2^{N/2} \int_{S_0}^{2 \times 2^{-N/4}} \rho(S) dS. \quad (\text{B17})$$

Inverting the above function  $\mu(S_0)$  provides the sought-after scaling of the singular values,  $S_\mu$ . Introducing the rescaled singular value  $s = 2^{N/4} S$ , we get

$$\mu(s_0) = 2^{N/2} C(s_0) \quad (\text{B18})$$

with

$$C(s_0) = \frac{1}{\pi} \int_{s_0}^2 ds \sqrt{4 - s^2}. \quad (\text{B19})$$

It follows that  $S_\mu$  follows a scaling law

$$S_\mu = 2^{-N/4} g(\mu/2^{N/2}), \quad (\text{B20})$$

where the function  $g(x) = C^{-1}(x)$  is the inverse of  $C(s_0)$ , as illustrated in the lower panel of Fig. 15. The function  $C(s_0)$  corresponds to the area of a portion of a (distorted) circle and can be computed using a simple geometrical argument. Introducing the angle  $\theta$  (see the inset of Fig. 15), one obtains

$$C(s_0) = \frac{1}{\pi} \mathcal{A} \left( 2 \arccos \left( \frac{s_0}{2} \right) \right) \quad (\text{B21})$$

with  $\mathcal{A}(\theta) = \theta - \sin(\theta)$ . We therefore obtain

$$g(x) = 2 \cos \left( \frac{1}{2} \mathcal{A}^{-1}(\pi x) \right). \quad (\text{B22})$$

In particular, one has  $g(0) = 2$  and  $g(1) = 0$ .

This scaling law can be used to derive the behavior of the error rate in the chaotic limit, Eq. (31). Truncating the original wave vector  $|\Psi\rangle$  [Eq. (33)] to its first  $\chi$  eigenvalues ( $|\tilde{\Psi}\rangle$ ) yields the fidelity  $\mathcal{F}(\chi) = |\langle \tilde{\Psi} | \Psi \rangle|^2 =$

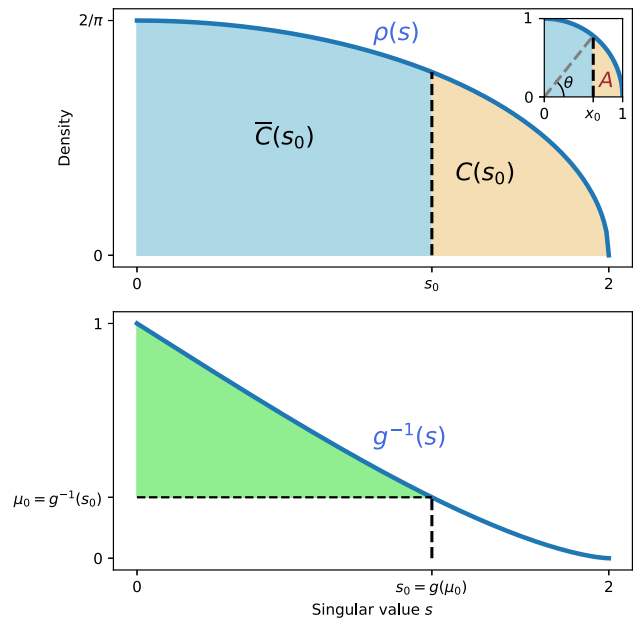


FIG. 15. Quadrant law (upper panel) and dispersion of the singular values (lower panel).

$\sum_{\mu=0}^{\chi-1} S_\mu^2$ . Thus, using Eq. (B20),

$$\mathcal{F}(\chi) = \int_0^{\chi/2^{N/2}} g^2(x) dx, \quad (\text{B23})$$

and in the  $\chi \ll 2^{N/2}$  limit we obtain the advertised chaotic limit

$$\mathcal{F}(\chi) = \frac{4\chi}{2^{N/2}} + O\left(\frac{\chi}{2^{N/2}}\right)^2 \quad (\text{B24})$$

and the associated value for the error rate  $\epsilon = 1 - f = 1 - \mathcal{F}^{2/(ND)}$ . Interestingly,  $\epsilon$  plateaus at  $\log(2)/D$  for large  $N$ .

We note that if all the singular values were equal ( $S_\mu = 1/2^{N/4}$ ) then we would have  $g(x) = 1$  and hence  $\mathcal{F} = \chi/2^{N/2}$  [following Eq. (B23)], namely, one fourth of the leading term of Eq. (B24). In other terms, the fidelity in the chaotic limit is only 4 times larger than in the worst possible situation where all the singular values are of equal importance.

We check in Fig. 16 that this scaling law of the singular values can indeed be observed. We perform a SVD decomposition of the random vector obtained by the procedure described in the previous subsection, and plot the corresponding function  $S_\mu$  properly rescaled. The result is almost indistinguishable from the analytical function  $g(x)$  calculated above.



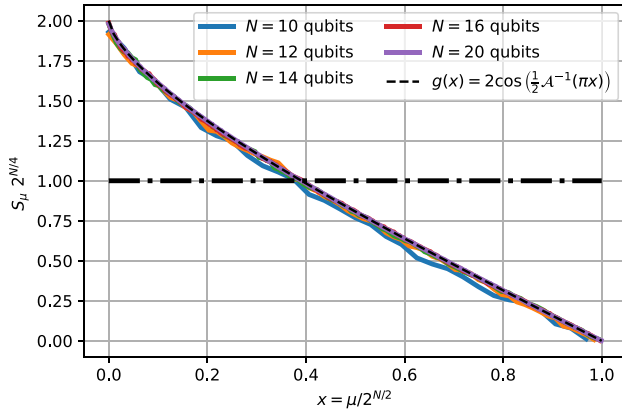


FIG. 16. Scaling behavior of the Schmidt coefficients (singular values)  $S_\mu$  of a Porter-Thomas wave vector for various numbers of qubits  $N$ . Black dashed line denotes the scaling function  $g(x)$ . Black dash-dot line denotes the constant singular values case ( $S_\mu = 1/2^{N/4}$ ).

### APPENDIX C: METROPOLIS SAMPLING FOR CLOSED SIMULATIONS

To produce the same output as a quantum computer within the framework of closed simulations, one must be able to sample from the distribution  $Q(x) = |\Psi_x|^2$ . This means not only computing amplitudes for a given bitstring but producing bitstrings distributed following  $Q(x)$ . A simple general algorithm for this task has been proposed in Ref. [22] (see also the discussion in Sec. II). The algorithm of Ref. [22] requires the calculations of  $\propto N_{2g}$  amplitudes per bitstring. In the case of random circuits such as sequences I and II, this is far from optimum. A possible strategy is to use the Metropolis algorithm to construct a Markov chain of bitstrings  $x_i$ : one picks  $x_{i+1}$  at random and accepts the proposed value with probability  $p_{\text{acc}} = \min(|\Psi_{x_{i+1}}/\Psi_{x_i}|^2, 1)$  (acceptance ratio). If the proposed move is refused then  $x_{i+1} \equiv x_i$ . In the random circuit of quantum supremacy, one quickly reaches a Porter-Thomas distribution, i.e., the amplitudes  $\Psi_x$  are themselves distributed according to an exponential law  $P(|\Psi_x|^2 = p) = 2^N \exp(-2^N p)$ . It follows that the average acceptance ratio is fairly large:

$$\langle p_{\text{acc}} \rangle = \int_0^\infty dx \int_0^\infty dy \min(x/y, 1) e^{-x} e^{-y} \approx 70\%. \quad (\text{C1})$$

That is, on average,  $1/0.7 \approx 1.5$  bitstrings must be calculated in order to get a new accepted bitstring.

If one wanted to pretend that the bitstrings came from an actual quantum computer, one would want to be almost certain that a given bitstring could not appear twice consecutively. In that case, one would want to keep only one bitstring every  $L$  Metropolis updates, hence lowering the

probability of repetition to about  $(1 - 0.7)^L = (0.3)^L$ . For  $L = 10$ , this would give a very low repetition probability of  $6 \times 10^{-4}\%$ . However, this precaution can probably be skipped as the cross-entropy benchmarking fidelity can be easily spoofed: simply ignoring the repeated values would create a bias in the distribution that would probably be very hard if not impossible to detect. Hence, one or two amplitudes per bitstring should be enough in practice.

- [1] Bela Bauer, Sergey Bravyi, Mario Motta, and Garnet Kin-Lic Chan, Quantum algorithms for quantum chemistry and quantum materials science, *Chem. Rev.* **120**, 12685 (2020).
- [2] Markus Reiher, Nathan Wiebe, Krysta M Svore, Dave Wecker, and Matthias Troyer, Elucidating reaction mechanisms on quantum computers, *Proc. Nat. Acad. Sci.* **114**, 7555 (2017).
- [3] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando G. S. L. Brandao, and David A. Buell, *et al.*, Quantum supremacy using a programmable superconducting processor, *Nature* **574**, 505 (2019).
- [4] Johnnie Gray and Stefanos Kourtis, Hyper-optimized tensor network contraction, *Quantum* **5**, 410 (2021).
- [5] Feng Pan and Pan Zhang, Simulating the Sycamore quantum supremacy circuits, (2021), [ArXiv:2103.03074](https://arxiv.org/abs/2103.03074).
- [6] Feng Pan, Keyang Chen, and Pan Zhang, Solving the Sampling Problem of the Sycamore Quantum Circuits, *Phys. Rev. Lett.* **129**, 090502 (2022).
- [7] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal, What Limits the Simulation of Quantum Computers?, *Phys. Rev. X* **10**, 041038 (2020).
- [8] A. J. Daley, C. Kollath, U. Schollwöck, and G. Vidal, Time-dependent density-matrix renormalization-group using adaptive effective Hilbert spaces, *J. Stat. Mech.: Theory Exp.* **2004**, P04005 (2004).
- [9] Guifré Vidal, Efficient Simulation of One-Dimensional Quantum Many-Body Systems, *Phys. Rev. Lett.* **93**, 040502 (2004).
- [10] Steven R. White and Adrian E. Feiguin, Real-Time Evolution Using the Density Matrix Renormalization Group, *Phys. Rev. Lett.* **93**, 076401 (2004).
- [11] Steven R. White, Density-matrix algorithms for quantum renormalization groups, *Phys. Rev. B* **48**, 10345 (1993).
- [12] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, *Ann. Phys. (N. Y.)* **326**, 96 (2011).
- [13] Farhi Edward, Goldstone Jeffrey, and Gutmann Sam, A quantum approximate optimization algorithm, (2014), [ArXiv:1411.4028](https://arxiv.org/abs/1411.4028).
- [14] Yulin Wu, *et al.*, Strong Quantum Computational Advantage Using a Superconducting Quantum Processor, *Phys. Rev. Lett.* **127**, 180501 (2021).
- [15] Han-Sen Zhong, *et al.*, Quantum computational advantage using photons, *Science* **370**, 1460 (2020).
- [16] A. S. Popova and A. N. Rubtsov, Cracking the quantum advantage threshold for Gaussian boson sampling, (2021), [ArXiv:2106.01445](https://arxiv.org/abs/2106.01445).

- [17] Benjamin Villalonga, Murphy Yuezheng Niu, Li Li, Hartmut Neven, John C. Platt, Vadim N. Smelyanskiy, and Sergio Boixo, Efficient approximation of experimental Gaussian boson sampling, (2021), [Arxiv:2109.11525](#).
- [18] Changhun Oh, Youngrong Lim, Bill Fefferman, and Liang Jiang, Classical Simulation of Boson Sampling Based on Graph Structure, *Phys. Rev. Lett.* **128**, 190501 (2022).
- [19] Yong Liu, Xin Liu, Fang Li, Haohuan Fu, Yuling Yang, Jiawei Song, Pengpeng Zhao, Zhen Wang, Dajia Peng, Huarong Chen, Chu Guo, Heliang Huang, Wenzhao Wu, and Dexun Chen, Closing the “quantum supremacy” gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer, [Arxiv:2110.14502v2](#).
- [20] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Sergio Boixo, Michael Broughton, Bob B. Buckley, David A. Buell *et al.*, Hartree-Fock on a superconducting qubit quantum computer, *Science* **369**, 1084 (2020).
- [21] Edwin Pednault, John A. Gunnels, Giacomo Nannicini, Lior Horesh, and Robert Wisnieff, Leveraging secondary storage to simulate deep 54-qubit Sycamore circuits, (2019), [Arxiv:1910.09534](#).
- [22] Sergey Bravyi, David Gosset, and Yincheng Liu, How to Simulate Quantum Measurement without Computing Marginals, *Phys. Rev. Lett.* **128**, 220503 (2022).
- [23] Cupjin Huang, Fang Zhang, Michael Newman, Junjie Cai, Xun Gao, Zhengxiong Tian, Junyin Wu, Haihong Xu, Huanjun Yu, Bo Yuan, Mario Szegedy, Yaoyun Shi, and Jianxin Chen, Classical simulation of quantum supremacy circuits, (2020), [Arxiv:2005.06787v1](#).
- [24] Matthew P. Harrigan, *et al.*, Quantum approximate optimization of non-planar graph problems on a planar superconducting processor, *Nat. Phys.* **17**, 332 (2021).
- [25] Xun Gao, Marcin Kalinowski, Chi-Ning Chou, Mikhail D. Lukin, Boaz Barak, and Soonwon Choi, Limitations of linear cross-entropy as a measure for quantum advantage, (2021), [Arxiv:2112.01657](#).
- [26] Yuchen Pang, Tianyi Hao, Annika Dugad, Yiqing Zhou, and Edgar Solomonik, in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis* (2020), p. 1.
- [27] Sheng-Hsuan Lin, Michael P. Zaletel, and Frank Pollmann, Efficient simulation of dynamics in two-dimensional quantum spin systems with isometric tensor networks, *Phys. Rev. B* **106**, 245102 (2022).
- [28] Timothy Proctor, Kenneth Rudinger, Kevin Young, Erik Nielsen, and Robin Blume-Kohout, Measuring the capabilities of quantum computers, *Nat. Phys.* **18**, 75 (2022).
- [29] Maxime Dupont, Nicolas Didier, Mark J. Hodson, Joel E. Moore, and Matthew J. Reagor, Calibrating the Classical Hardness of the Quantum Approximate Optimization Algorithm, *PRX Quantum* **3**, 040339 (2022).
- [30] Rishi Sreedhar, Pontus Vikstål, Marika Svensson, Andreas Ask, Göran Johansson, and Laura García-Álvarez, The quantum approximate optimization algorithm performance with low entanglement and high circuit depth, (2022), [Arxiv:2207.0340](#).
- [31] Matija Medvidović and Giuseppe Carleo, Classical variational simulation of the quantum approximate optimization algorithm, *npj Quantum Inf.* **7**, 101 (2021).
- [32] Bjarni Jónsson, Bela Bauer, and Giuseppe Carleo, Neural-network states for the classical simulation of quantum computing, (2018), [Arxiv:1808.05232](#).
- [33] Steven R. White, Density Matrix Formulation for Quantum Renormalization Groups, *Phys. Rev. Lett.* **69**, 2863 (1992).
- [34] Hamed Saberi, Andreas Weichselbaum, Lucas Lamata, David Pérez-García, Jan von Delft, and Enrique Solano, Constrained optimization of sequentially generated entangled multiqubit states, *Phys. Rev. A* **80**, 022334 (2009).
- [35] E. M. Stoudenmire and Steven R. White, Minimally entangled typical thermal state algorithms, *New J. Phys.* **12**, 055026 (2010).
- [36] Yuichi Hirata, Masaki Nakanishi, Shigeru Yamashita, and Yasuhiko Nakashima, in *2009 Third International Conference on Quantum, Nano and Micro Technologies* (IEEE, 2009), p. 26.
- [37] P. W. Brouwer and C. W. J. Beenakker, Diagrammatic method of integration over the unitary group, with applications to quantum transport in mesoscopic systems, *J. Math. Phys.* **37**, 4904 (1996).
- [38] S. Martiel, T. Ayrál, and C. Allouche, Benchmarking quantum coprocessors in an application-centric, hardware-agnostic and scalable way, *IEEE Trans. Quantum Eng.* **2**, (2021).
- [39] Igor L. Markov and Yaoyun Shi, Simulating quantum computation by contracting tensor networks, *SIAM J. Comput.* **38**, 963 (2008).
- [40] J. Eisert, M. Cramer, and M. B. Plenio, Colloquium: Area laws for the entanglement entropy, *Rev. Mod. Phys.* **82**, 277 (2010).
- [41] F. Verstraete, V. Murg, and J. I. Cirac, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, *Adv. Phys.* **57**, 143 (2008).
- [42] Sebastian Paeckel, Thomas Köhler, Andreas Swoboda, Salvatore R. Manmana, Ulrich Schollwöck, and Claudius Hubig, Time-evolution methods for matrix-product states, *Ann. Phys. (N. Y.)* **411**, 167998 (2019).
- [43] Jianhong Shen, On the singular values of Gaussian random matrices, *Linear Algebra Appl.* **326**, 1 (2001).