

Scalable simulation of quantum many-body dynamics with OR-represented quantum algebra

Lukas Broers^{1,*}, Rong-Yang Sun^{1,2} and Seiji Yunoki^{1,3,4,5}

¹*Computational Materials Science Research Team, RIKEN Center for Computational Science (R-CCS), Kobe, Hyogo, Japan*

²*RIKEN Interdisciplinary Theoretical and Mathematical Sciences Program (iTHEMS), Wako, Saitama, Japan*

³*Quantum Computational Science Research Team, RIKEN Center for Quantum Computing (RQC), Wako, Saitama, Japan*

⁴*Computational Condensed Matter Physics Laboratory, RIKEN Cluster for Pioneering Research (CPR), Wako, Saitama, Japan*

⁵*Computational Quantum Matter Research Team, RIKEN Center for Emergent Matter Science (CEMS), Wako, Saitama, Japan*



(Received 2 July 2025; revised 7 April 2026; accepted 14 July 2026; published 18 August 2026)

High-performance numerical methods are essential not only for advancing quantum many-body physics but also for enabling integration with emerging quantum computing platforms. We present a scalable, general-purpose parallel algorithm for quantum simulations based on OR-represented quantum algebra (ORQA). This framework applies to arbitrary spin systems and naturally integrates with quantum circuit simulation in the Heisenberg picture, making it particularly relevant to recent large-scale experiments on superconducting qubit processors [Y. Kim *et al.* *Nature (London)* **618**, 500 (2023)]. As a benchmark, we simulate the kicked Ising model on a 127-qubit heavy-hexagon lattice, tracking the time evolution of local magnetization using up to 1×10^{12} Pauli strings. Executed on the supercomputer Fugaku, our simulations exhibit strong scaling up to 2^{17} parallel processes with near-linear communication overhead. These results establish ORQA as a practical and high-performance tool for quantum many-body dynamics and highlight its potential for integration into hybrid quantum-classical computational frameworks, complementing recent advances in tensor network and surrogate simulation techniques.

DOI: 10.1103/y8ft-m61w

I. INTRODUCTION

Despite continuous progress in the development of quantum hardware, the milestone of fault-tolerant quantum computing (FTQC)—and with it, definitive quantum supremacy—has yet to be achieved [1–4]. With the steadily improving capabilities of noisy intermediate-scale quantum devices [5,6], the hybrid integration of quantum processors with high-performance computing (HPC) pipelines has emerged as a promising strategy for realizing quantum advantage before the FTQC era. Recent advances in quantum-centric supercomputing architectures exemplify this direction [7,8]. At the same time, pre-fault-tolerant quantum processors have demonstrated practical

utility through accurate expectation value measurements, enabled by sophisticated error mitigation techniques [3]. Remarkably, these results have been reproduced—and in some cases even surpassed—by state-of-the-art classical simulation methods [9–14]. In parallel, developments in quantum machine learning have highlighted the potential of hybrid quantum-classical optimization loops, suggesting further opportunities for synergy between quantum hardware and classical HPC infrastructures [15,16].

This class of hybrid quantum approaches relies equally on progress in classical high-performance computation and on the development of more capable quantum processors. Among the many computational techniques developed to address quantum many-body systems, tensor-network methods stand out as particularly powerful. In particular, matrix product states (MPS) [17–20] and projected entangled-pair states [21,22] have become foundational tools in computational quantum physics [23,24]. While these methods were not originally designed to interface with quantum hardware, they have proven indispensable for benchmarking, verifying, and interpreting

*Contact author: Lukas.broers@riken.jp

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

results from quantum simulation experiments. In addition, recent developments in surrogate simulation techniques, such as sparse Pauli dynamics and operator-based classical simulation frameworks [10,25,26], offer complementary strategies that scale favorably with certain circuit structures or observables. These approaches are helping to bridge the gap between classical and quantum simulations, providing practical tools for interpreting experimental results and guiding algorithmic design.

In this work, we demonstrate a large-scale, high-performance implementation of the parallelized OR-represented quantum algebra (ORQA) formalism [27], executed on the supercomputer Fugaku. ORQA enables efficient computation over large collections of Pauli strings and is naturally suited for quantum simulations involving operator dynamics, such as sparse Pauli evolution [14,26]. As a benchmark, we simulate the kicked Ising model on a two-dimensional heavy-hexagon geometry, following the layout used in the IBM *Eagle* processor [3]. Our implementation tracks the dynamics of over 1×10^{12} Pauli strings, achieving a wall-clock time of only a few seconds per quantum gate. This enables the simulation of circuits containing more than 10 000 gates on 127 qubits, with an overall run-time of just a few hours. Interestingly, we find that the relationship between simulation accuracy and the number of retained Pauli strings is highly sensitive to model-specific parameters. In particular, tuning a single physical parameter can significantly alter the complexity and effectiveness of the ORQA simulation.

Our implementation demonstrates both strong and weak scaling up to 2^{17} parallel computational processes. Beyond this point, linearly increasing communication overhead becomes the dominant bottleneck and can only be mitigated when the number of Pauli strings exceeds the memory capacity of the available hardware. Within these practical limitations, our method enables the efficient simulation of arbitrary quantum circuits. As such, the ORQA framework constitutes a powerful and general-purpose numerical tool with broad applicability, particularly in quantum-hybrid high-performance computing and the simulation of quantum many-body dynamics.

The remainder of this paper is organized as follows. In Sec. II, we briefly review the ORQA formalism, introduce its key data structures, and summarize the parallel implementation. Section III presents benchmark results for the kicked Ising model on the heavy-hexagon geometry and analyzes the performance characteristics of our method. We conclude in Sec. IV with a summary of our findings and an outlook for future work. Additional details regarding the ORQA algorithm and the parallel implementation are provided in Appendixes A–D. To further demonstrate the versatility of our method, we include a second, smaller-scale application to a different system in Appendix E.

Simulation accuracy is examined in Appendix F, and the specifications of the computing system used for benchmarking are detailed in Appendix G.

II. METHODS

The ORQA formalism [27] enables efficient manipulation of Pauli strings by reducing matrix multiplications to bitwise logical operations on encoded indices. A Pauli string on n qubits is expressed as

$$\sigma_I = \sigma_{I_1 | \dots | I_n} = \bigotimes_{i=1}^n \sigma_{I_i}, \quad (1)$$

where each I_i is a two-bit string encoding the i th local Pauli operator according to the mapping $\sigma_{00_2} = \text{id}$, $\sigma_{01_2} = \sigma_x$, $\sigma_{10_2} = \sigma_y$, and $\sigma_{11_2} = \sigma_z$. We refer to the full $2n$ -bit concatenation $I = I_1 | \dots | I_n$ as a multi-index. The product of two arbitrary Pauli strings obeys the algebraic rule

$$\sigma_I \sigma_J = i^{B(I,J)} \sigma_{I \underline{\vee} J}, \quad (2)$$

where $\underline{\vee}$ denotes the bitwise exclusive-OR (XOR) operation. The exponent $B(I, J)$ is determined by local contributions:

$$B(I, J) = \sum_{i=1}^n b(I_i, J_i), \quad (3)$$

where $I_i, J_i \in \{00_2, 01_2, 10_2, 11_2\}$, and the local phase $b(I_i, J_i)$ captures the structure constants of the $\mathfrak{su}(2)$ algebra. Specifically, $b(I_i, J_i) = 0$ whenever either $I_i = 00_2$ or $J_i = 00_2$ (i.e., an identity operator is present). Otherwise, the phase is given by $b(I_i, J_i) = \sum_{k=1}^3 \epsilon_{I_i J_i k}$, where ϵ_{ijk} denotes the Levi-Cevita symbol.

To make use of this formalism, we express a generic Hermitian operator as a linear combination of Pauli strings:

$$O = \sum_{I=0}^{4^n-1} O_I \sigma_I, \quad (4)$$

where $O_I \in \mathbb{R}$. In general, the summation runs over all 4^n possible Pauli strings on n qubits. However, in practice, a large fraction of the coefficients O_I may vanish or be sufficiently small to be neglected without introducing significant error. We denote by $|O|$ the number of Pauli strings retained in the representation of O and refer to this quantity as the effective complexity, as it directly determines the computational cost of the algorithm.

We consider simulations of quantum circuits in the Heisenberg picture, where the dynamics of observables O measured at the end of the circuit are tracked. Using

Eq. (2), the action of a unitary gate generated by a single Pauli string σ_J with rotation angle θ on the operator O defined in Eq. (4) can be expressed as

$$\frac{1}{2^n} \text{Tr} \left[\sigma_I e^{i\frac{\theta}{2}\sigma_J} O e^{-i\frac{\theta}{2}\sigma_J} \right] = \begin{cases} O_I, & \text{if } B(I, J) \text{ is even,} \\ \cos(\theta) O_I + (-1)^{\frac{B(I, J)+1}{2}} \sin(\theta) O_{I \underline{\vee} J}, & \text{if } B(I, J) \text{ is odd,} \end{cases} \quad (5)$$

for all multi-indices I . Here, the left-hand side denotes the coefficient of σ_I in the operator O after conjugation by the unitary $e^{-i\frac{\theta}{2}\sigma_J}$, corresponding to the applied gate. Note that this expression relies on the identity $(I \underline{\vee} J) \underline{\vee} J = I$ as well as $B(I, J) = -B(J, I)$. This transformation enables a straightforward generalization to arbitrary dynamics, as it determines the evolution of each coefficient O_I in an arbitrary operator O under the action of the unitary gate $e^{-i\frac{\theta}{2}\sigma_J}$. The full time evolution is then obtained by iteratively applying Eq. (5) to the operator O for each gate in the circuit. This update rule is analogous to those used in sparse Pauli dynamics [14,26,28,29] and low weight efficient simulation [10,25], which evolve sums of Pauli strings using stabilizer decompositions [30] or Pauli transfer matrices [31,32]. In contrast, our method is based on the ORQA formalism [27], without requiring circuit preprocessing into Clifford and non-Clifford components. Moreover, our implementation is optimized for execution on large-scale CPU-based supercomputers. Additional algorithmic details are provided in Appendix A.

We parallelize this calculation by distributing the representation of the operator O across N parallel processes, each of which independently computes the updates given in Eq. (5). The growth in the number of Pauli strings over time and their distribution across different processes are illustrated in Figs. 1(c) and 1(d). Details of the data structure used to represent O numerically are provided in Appendix B. Each process, labeled by index m , maintains a partial representation $O^{(m)}$ of the full operator, such that

$$O = \sum_{m=0}^{N-1} O^{(m)}. \quad (6)$$

These partial representations are fully disjoint in the sense that each possible multi-index I appears in exactly one of the $O^{(m)}$. To achieve and manage this efficiently, we introduce a distribution map

$$f: \{0, 1\}^{2n} \rightarrow \mathbb{N}_N, \quad (7)$$

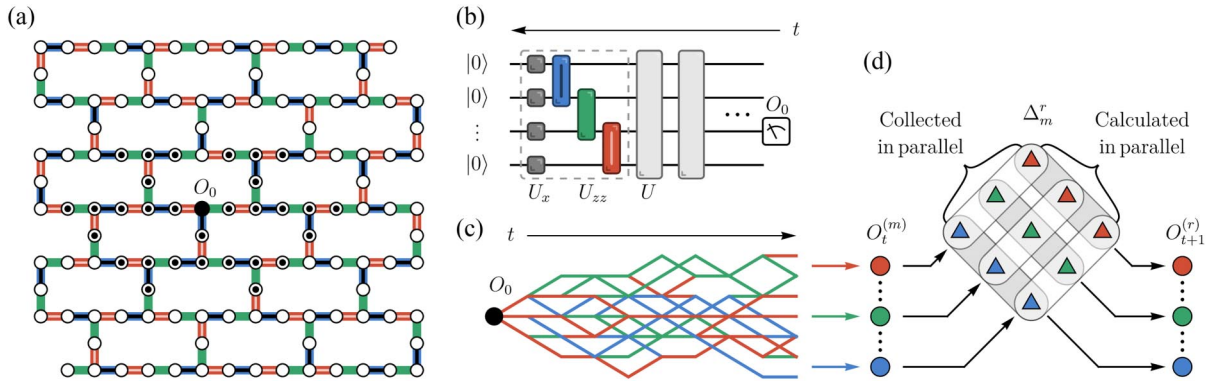


FIG. 1. Illustration of the model and simulation method. (a) Heavy-hexagon geometry of the 127-qubit IBM *Eagle* processor simulated in this work. Circles represent qubits, with connections indicating where two-qubit gates can be applied. The filled circle near the center marks the location of the initial operator $O_0 = \sigma_z^{62}$, which is evolved in time. Half-filled circles indicate the interior of the causal light cone after $t = 5$ time steps, originating from the initial operator. The two-qubit connectivity is partitioned into three disjoint sets, each enabling the parallel application of gates. (b) Quantum circuit implementing the Trotterized dynamics of the kicked Ising model. Each layer U , enclosed by a dashed box, consists of a sequence of single-qubit gates U_x and two-qubit entangling gates U_{zz} , forming a single time step. The U_{zz} gates are grouped into three color-coded sets, corresponding to the partitioning in (a), to facilitate parallel execution. (c) Time evolution of the initial operator O_0 , which begins as a single Pauli string and grows over time to include many additional Pauli strings, distributed across parallel processes labeled by m . In the Heisenberg picture, the operator evolution proceeds backward in time relative to the forward circuit direction shown in (b). (d) At each time step t , the partial representations $O_t^{(m)}$ generate update lists Δ_m^r , which are communicated in parallel to the target processes r . These updates are required to construct the next-step operator $O_{t+1}^{(r)}$. Note that the color coding in (a), (b) is unrelated to the color coding in (c), (d).

which assigns each possible Pauli string to a specific process $m \in \mathbb{N}_N = \{0, 1, \dots, N-1\}$. The exact form of f is nontrivial and has a significant impact on parallel performance; our approach and its rationale are discussed in Appendix C. While the specific assignment of a given multi-index I to a process m is not directly significant, it is essential that f distributes multi-indices $I \in \{0, 1\}^{2n}$ approximately uniformly across all processes. This ensures balanced memory usage and computational load, i.e., $|O^{(m)}| \approx |O|N^{-1}$, which is crucial for maintaining scalability. Since each process holds a disjoint subset $O^{(m)}$, it must also compute a corresponding subset of updates, based on Eq. (5), that are needed by other processes. Each process m is therefore responsible for generating a collection of update lists, denoted Δ_m^r , to be sent to processes $r \neq m$. Efficient computation and distribution of these updates across all processes is a central feature of our algorithm. Further implementation details are provided in Appendix A.

Within a circuit simulation, each quantum gate typically generates additional Pauli strings, leading to a steady—and potentially exponential—increase in the size of the operator representation O . To control this growth, we apply dynamic truncation to each partial representations $O^{(m)}$ by removing components $O_I^{(m)}$ that satisfy

$$|O_I^{(m)}| \leq \epsilon_0 \max_I |O_I|, \quad (8)$$

where $\epsilon_0 > 0$ is a parameter that controls the strictness of the truncation. Note that the maximum is taken over all coefficients in the full operator O , not just those in $O^{(m)}$, to ensure consistent truncation across all processes, regardless of how the distribution map f partitions the data. In principle, various truncation strategies can be employed. We discuss the specific truncation scheme used in this work in Appendix D and benchmark its accuracy through comparisons with exact numerical results in Appendix F.

III. RESULTS

To test our method, we consider the kicked Ising model, which has been studied experimentally [3] and whose dynamics have been semiquantitatively reproduced using various classical simulation techniques [9–14]. The unitary dynamics of this model are described by a Trotterized sequence of single- and two-qubit gates:

$$U = \underbrace{\prod_{\langle i,j \rangle} \exp\left(-i \frac{\theta_{zz}}{2} \sigma_z^i \sigma_z^j\right)}_{U_{zz}} \underbrace{\prod_{j=1}^n \exp\left(-i \frac{\theta_x}{2} \sigma_x^j\right)}_{U_x}, \quad (9)$$

where σ_x^j and σ_z^j are the Pauli- X and Pauli- Z operators acting on the j th qubit. The two-qubit interactions are

defined between neighboring qubits on a two-dimensional heavy-hexagon geometry with $n = 127$ qubits, as illustrated in Fig. 1(a). The notation $\langle i, j \rangle$ in Eq. (9) refers to a pair of adjacent qubits i and j on this lattice. The corresponding quantum circuit structure, consisting of layers of U_{zz} and U_x gates, is shown in Fig. 1(b). Throughout this work, we fix the interaction strength at $\theta_{zz} = -\frac{\pi}{2}$ and vary the transverse field parameter θ_x . Within this model, we study the local magnetization at qubit $j = 62$ after applying t layers of U , defined as

$$\langle M_t \rangle = \langle 0 | (U^\dagger)^t \sigma_z^{62} (U)^t | 0 \rangle, \quad (10)$$

where $|0\rangle = |0\rangle^{\otimes n}$ is the initial product state. We compute this quantity in the Heisenberg picture by evolving the initial observable $O_0 = \sigma_z^{62}$. We emphasize that, owing to the generality of our method, any linear combination of Pauli strings with arbitrary coefficients can be used as the initial operator O_0 . Likewise, the method is applicable to arbitrary quantum circuits U . Details of the computing system used for our simulations are provided in Appendix G.

We present the results of our simulations in Figs. 2 and 3. Figure 2(a) shows the time evolution of the magnetization $\langle M_t \rangle$ for $\theta_x = 0.1\pi$ and truncation thresholds $\epsilon_0 = 10^{-4}$ (green), 10^{-5} (blue), and 10^{-6} (red). The corresponding effective complexity $|O|$, defined as the number of retained Pauli strings, is shown in Fig. 2(b) for the same set of parameters. Figures 2(c) and 2(d) present the same quantities for $\theta_x = 0.25\pi$. The magnetization results exhibit a visible dependence on the truncation threshold ϵ_0 , reflecting the impact of the number of Pauli strings retained at each stage of the evolution. As ϵ_0 decreases, truncation becomes less aggressive, resulting in a systematic increase in $|O|$ and improved simulation accuracy. In all cases, the number of Pauli strings grows rapidly at early times, followed by a pronounced slow-down and eventual saturation due to truncation. This saturation behavior is qualitatively influenced by the choice of θ_x , with different values leading to markedly different asymptotic values of $|O|$ for the same ϵ_0 . Nonetheless, even in regimes where saturation occurs, truncation-induced errors can accumulate over long time evolutions and eventually degrade simulation accuracy. This effect is particularly evident in Fig. 2(a), where maintaining accuracy up to $t = 50$ requires setting ϵ_0 as low as 10^{-6} .

Figure 2(e) shows the magnetization $\langle M_t \rangle$ and the effective complexity $|O|$ as functions of θ_x at a fixed evolution time $t = 20$, for several values of the truncation threshold ϵ_0 . Error-mitigated experimental results from Ref. [3] are also shown as black solid lines with shaded confidence intervals. Notably, the parameter regime $\theta_x \in [\pi/8, \pi/4]$ is particularly sensitive to truncation, exhibiting the largest deviations among the different values

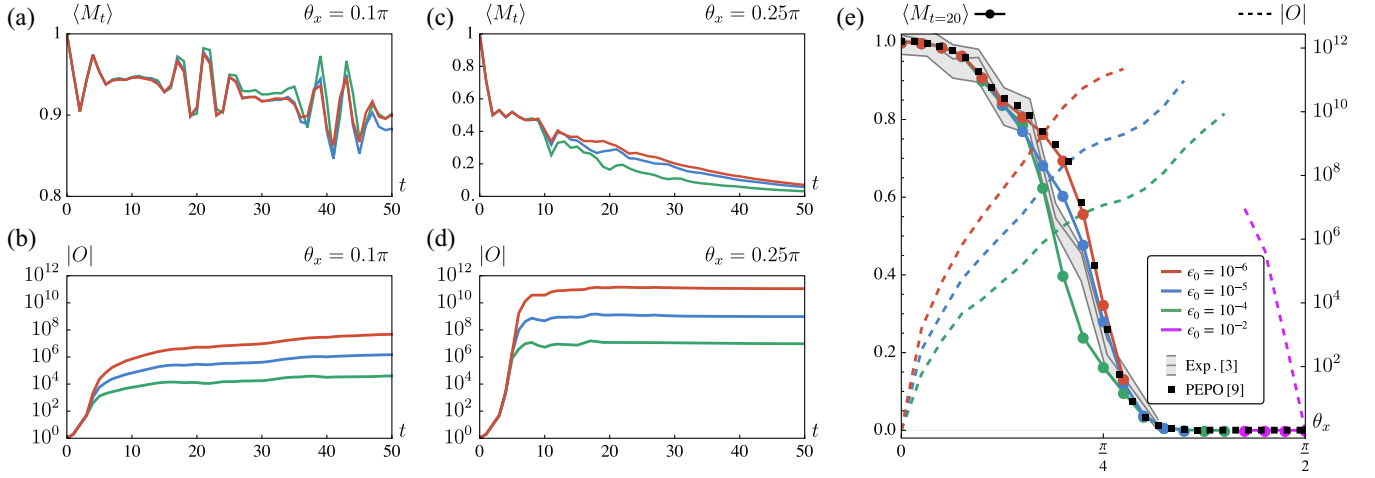


FIG. 2. Simulation of the kicked Ising model on the 127-qubit heavy-hexagon lattice. (a) Time evolution of the magnetization $\langle M_t \rangle$ for $\theta_x = 0.1\pi$ and truncation thresholds $\epsilon_0 = 10^{-4}$ (green), 10^{-5} (blue), and 10^{-6} (red). (b) Corresponding effective complexity $|O|$ for the same simulations as in (a). (c), (d) Same as (a), (b), respectively, but for $\theta_x = 0.25\pi$. (e) Magnetization $\langle M_t \rangle$ (solid circles) and effective complexity $|O|$ (dashed lines) at $t = 20$ as functions of θ_x for different values of ϵ_0 : 10^{-2} (purple), 10^{-4} (green), 10^{-5} (blue), and 10^{-6} (red). Error-mitigated experimental results from Ref. [3] are shown as black solid lines with shaded confidence intervals. For comparison, the results obtained using a tensor-network method based on projected entangled-pair operators, extrapolated to infinite bond dimension ($\chi \rightarrow \infty$), from Ref. [9] are shown as black squares. The underlying heavy-hexagon geometry and the location of the magnetization observable are shown in Fig. 1(a).

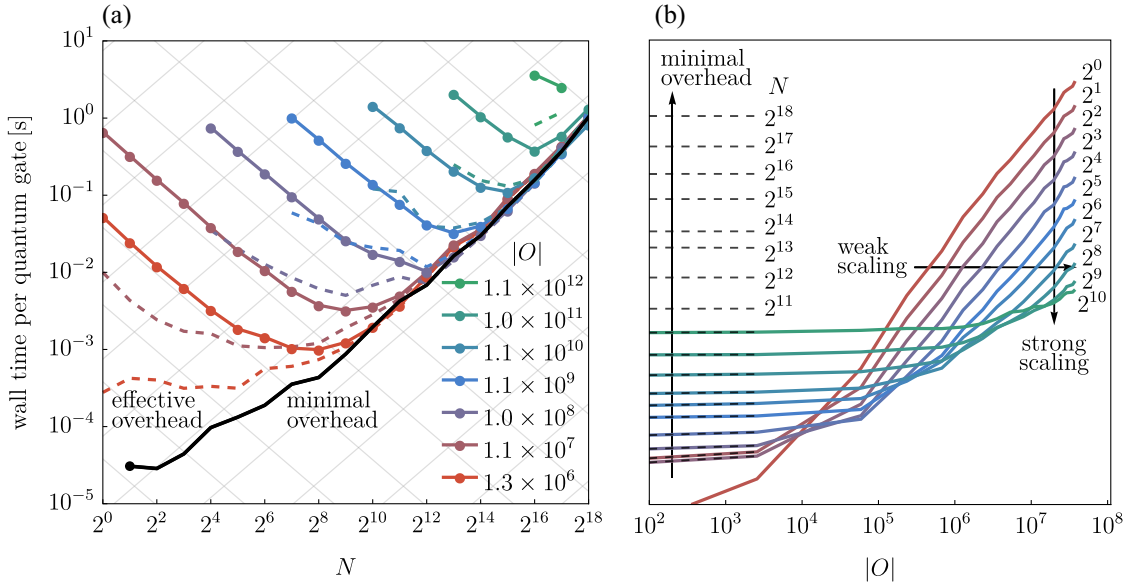


FIG. 3. Scaling behavior of quantum circuit simulations. (a) Wall time per quantum gate as a function of the number of parallel computational processes N . The solid black line denotes the minimal communication overhead that persists even in the limit $|O| \rightarrow 0$. Colored curves show the total wall time per gate (solid lines with symbols) and the effective communication overhead (dashed lines) for various effective complexities $|O|$, as labeled. Diagonal grid lines represent ideal N and N^{-1} scaling for visual reference. (b) Wall time per quantum gate as a function of effective complexity $|O|$ for different values of N , as indicated, using the same vertical-axis scale as in (a). For small $|O|$, performance is limited by the minimal communication overhead (dashed lines). As $|O|$ increases, the computational cost of simulating the operator dynamics becomes dominant, resulting in favorable scaling. Strong scaling is evident from the approximate N^{-1} scaling of wall time (vertical arrow), and the results also exhibit near-perfect weak scaling, with wall time remaining nearly constant when both $|O|$ and N are scaled proportionally (horizontal arrow).

of ϵ_0 . This challenging regime coincides with the region where other classical simulations also report substantial discrepancies, as discussed in recent studies (see, for example, Fig. 8 in Ref. [11]). We additionally include the numerical results from Ref. [9], shown as black squares. These data were obtained using a projected entangled-pair operator (PEPO) method, extrapolated to infinite bond dimension. The PEPO results are in excellent agreement with our simulation at $\epsilon_0 = 10^{-6}$, lending further support to the accuracy of our method in this regime.

While these methods are not straightforward to compare directly, we provide representative metrics from Ref. [9] for context. The memory footprint of PEPO is $64\chi^n$ bytes per site for double-precision complex numbers, where n denotes the local coordination number. On the 127-qubit heavy-hexagon lattice, this corresponds to approximately 20 GB for $\chi = 256$, and obtaining a single data point required about 7 h in their simulations. Scaling to larger bond dimensions is expected to be challenging, as the memory and time complexities scale as $\mathcal{O}(\chi^3)$ and $\mathcal{O}(\chi^5)$, respectively. Moreover, such tensor-network approaches are generally difficult to scale efficiently to massively parallel multiprocess architectures. For comparison, in our most accurate simulations with $\epsilon_0 = 10^{-6}$, we find $|O| \approx 10^{11}$ at $\theta_x = \pi/4$. This corresponds to a memory footprint of approximately 4 TB (about 40 bytes per Pauli string) and a simulation time of about 25 min using 12 288 computational nodes in parallel. Since both memory and run-time scale approximately linearly with $|O|$, many of the data points at $\epsilon_0 = 10^{-6}$ require substantially fewer computational resources. For example, at $\theta_x \approx 0.15\pi$, we find $|O| \approx 2 \times 10^8$, corresponding to a memory footprint comparable to 20 GB, while the simulation completes within a few seconds using 4096 nodes in parallel. In Appendix E, we present an additional benchmark comparing our method with MPS simulations, providing further insight into the differences between these approaches.

For $\theta_x = 0$ and $\theta_x = \pi/2$, the dynamics correspond to a fully classical regime and a dual-unitary circuit, respectively, both of which exhibit minimal effective complexity. As θ_x increases from zero, the effective complexity grows rapidly, reflecting the increasingly entangled and intricate nature of the system's dynamics. It is worthy to note, however, that a large number of Pauli strings does not necessarily indicate physically complex behavior. For example, when $\theta_x \lesssim \pi/2$, the system undergoes rapid thermalization, resulting in a fast decay of the magnetization. This qualitative feature is captured even under strong truncation, as shown for $\epsilon_0 = 10^{-2}$ in purple. In this regime, although the magnetization quickly decays and becomes featureless, the effective complexity still increases sharply as θ_x decreases from $\pi/2$, indicating that the growth in $|O|$ is not always directly linked to observable physical complexity.

Figure 3(a) shows the wall time required to apply a single quantum gate as a function of the number of parallel computational processes N during circuit simulation. This value is computed as the average wall time per circuit layer, obtained by dividing the total wall time for each layer by the number of gates that it contains (127 single-qubit X -rotation gates and 144 two-qubit ZZ -rotation gates). The total wall times per gate are shown as solid lines with symbols. Results are shown for various values of the effective complexity $|O|$ (i.e., the number of retained Pauli strings), corresponding to snapshots taken at different times during the time evolution of the operator O with varying parameters. The data exhibit clear strong-scaling behavior: the wall time approximately follows a N^{-1} power-law scaling until communication overhead becomes the dominant cost.

Dashed lines in Fig. 3(a) represent the measured communication overhead across processes. This includes both the actual data transfer time and the additional cost associated with initiating and finalizing MPI communication routines involving a large number of processes. We refer to this latter contribution as the minimal communication overhead, which is also indicated in isolation as a solid black line. Notably, when communication becomes the bottleneck, the actual data transfer time remains negligible, and nearly the entire overhead arises from this minimal initialization and finalization cost. As a result, we observe consistently favorable scaling across all tested values of $|O|$. The minimal communication overhead is reported as an average, as it fluctuates slightly between runs depending on the physical allocation of processes and the topology of the interconnect network on the computing system. At $N = 2^{18}$, the minimal communication overhead becomes sufficiently large such that maintaining strong-scaling efficiency requires simulating more than $|O| = 10^{12}$ Pauli strings. This value approaches the memory limits of our current hardware configuration, rendering further increases in N ineffective under the present setup.

Figure 3(b) shows the average wall time per quantum gate as a function of the effective complexity $|O|$. Each curve corresponds to the same circuit simulation executed with a different number of parallel computational processes N . At early stages of the time evolution, the number of Pauli strings is relatively small, and the wall time is dominated by the minimal communication overhead, which is largely independent of $|O|$ but increases with N . As the simulation progresses and the number of Pauli strings grows, the computational cost associated with updating the time-evolved operator O becomes the dominant factor. In this regime, the wall time decreases approximately linearly with N , confirming strong-scaling behavior, as indicated by the vertical arrow. Furthermore, the results demonstrate near-perfect weak scaling: the wall time remains nearly constant when both $|O|$ and N are

increased proportionally, as indicated by the horizontal arrow.

Finally, we emphasize that the scaling behavior shown in Fig. 3 is general and does not explicitly depend on physical details of the simulated system, such as the number of qubits. Instead, the overall performance is primarily dictated by the effective complexity $|O|$, which is governed by the chosen truncation strategy. By contrast, the relationship between $|O|$ and the accuracy of the simulation is determined by the physical characteristics of the system under study. Ultimately, it is these system-specific properties that determine the practical effectiveness of our method.

IV. CONCLUSION

We have presented a large-scale, parallelized, high-performance implementation of the ORQA formalism [27] for simulating unitary quantum dynamics in the Heisenberg picture, in the spirit of sparse Pauli dynamics [13]. As a benchmark application, we simulated the kicked Ising model on a 127-qubit heavy-hexagon geometry, involving over 10 000 quantum gates and up to 50 circuit layers. Our implementation tracked over 1×10^{12} Pauli strings across 2^{17} parallel processes, achieving wall times of only a few seconds per gate, resulting in total run times of just a few hours on the supercomputer Fugaku. This performance highlights the scalability and numerical efficiency of our approach. Moreover, the ORQA framework is well suited for development on GPU-based supercomputing architectures, which promises to further accelerate performance. It also has the potential to play a vital role in quantum-HPC hybrid workflows on emerging quantum-centric supercomputing platforms.

We have compared our simulation results for the magnetization $\langle M_z \rangle$ at $t = 20$ across various values of θ_x with error-mitigated experimental data obtained from the IBM *Eagle* processor [3]. Overall, the agreement is satisfactory, except in the region $\theta_x \approx 0.2\pi - 0.25\pi$, which coincides with the regime where different classical simulation techniques yield divergent results, as comprehensively summarized in Ref. [11]. It is worth noting, however, that our results obtained with the smallest truncation threshold, $\epsilon_0 = 10^{-6}$, exhibit excellent agreement across all values of θ_x with those produced by a tensor-network method based on PEPO in the Heisenberg picture, extrapolated to infinite bond dimension [9]. While we do not claim that our results are definitive, as systematic convergence checks with respect to ϵ_0 remain to be performed, these comparisons suggest that further experimental investigations, particularly in this challenging parameter regime, would be highly valuable, in parallel with continued advancements in classical simulation techniques.

Moreover, we have demonstrated both weak and strong scaling in our method, as well as a time complexity that

scales nearly linearly with the number of retained Pauli strings. Notably, this scaling behavior is independent of the specific physical system being simulated. The accuracy and tractability of simulations performed using our approach depend solely on the number of Pauli strings that can be safely discarded without incurring significant error. This trade-off is governed by the physical properties of the system and the specific truncation strategy employed. We anticipate that future efforts aimed at tailoring truncation schemes to the structure of particular physical problems will play a central role in further improving both the efficiency and the scope of applicability of our algorithm.

The method presented in this work is general and highly versatile: it can, in principle, be applied directly to arbitrary quantum circuits, observables, and states, irrespective of the system's geometry or number of qubits. Although not explored in the present study, our implementation can be readily extended to simulate dissipative dynamics and imaginary-time evolution. By design, it provides direct access to the coefficients of all retained Pauli strings contributing to a given expectation value, without requiring additional computational overhead. These features establish our implementation as a powerful and broadly applicable numerical tool for quantum many-body physics, with potential relevance to a wide range of problems in quantum simulation and quantum-HPC hybrid computation.

ACKNOWLEDGMENTS

We are grateful to Hai-Jun Liao and Tao Xiang for sharing their results from Ref. [9], and to the IBM Quantum team, as well as Tomonori Shirakawa, for valuable discussions. Numerical simulations were performed on the supercomputer Fugaku and the HOKUSAI supercomputer at RIKEN (Project ID RB240036). A portion of this work is based on results obtained from Project No. JPNP20017, supported by the New Energy and Industrial Technology Development Organization (NEDO). This study was also supported by JSPS KAKENHI Grants No. JP21H04446 and No. JP22K03520. Additional support was provided by JST COI-NEXT (Grant No. JPMJPF2221) and the Program for Promoting Research on the Supercomputer Fugaku (Grant No. MXP1020230411) by MEXT, Japan. We further acknowledge support from the UTokyo Quantum Initiative, the RIKEN TRIP initiative (RIKEN Quantum), and the COE research grant in computational science from Hyogo Prefecture and Kobe City through the Foundation for Computational Science.

DATA AVAILABILITY

The data that support the findings of this article are not publicly available upon publication because it is not

technically feasible and/or the cost of preparing, depositing, and hosting the data would be prohibitive within the terms of this research project. The data are available from the authors upon reasonable request.

APPENDIX A: ALGORITHMIC DETAILS

Our parallel numerical algorithm proceeds as follows. Each process m holds a partial representation $O^{(m)}$ of the full operator O , such that their union reconstructs the complete operator, as described in Eq. (6). Each $O^{(m)}$ is implemented as a dynamically sized list of tuples $(I, O_I^{(m)})$, where I is the multi-index labeling a Pauli string and $O_I^{(m)}$ is its corresponding coefficient. Each process sequentially applies the gates in the quantum circuit, where each gate is represented as a tuple (J, θ_J) corresponding to the unitary operator $e^{-i\frac{\theta_J}{2}\sigma_J}$. For each gate, the process loops over the elements $(I, O_I^{(m)})$ of its local operator $O^{(m)}$. If σ_I and σ_J do not commute, the process registers the corresponding updates using Eq. (5). Each process m maintains N dynamically allocated update lists, denoted Δ_r^m , one for each process $r < N$. The local update $O_I^{(m)} \leftarrow \cos(\theta_J)O_I^{(m)}$ is performed immediately. The associated nonlocal update is appended to the list $\Delta_{f(I \vee J)}^m$ as a tuple

$$\left(I \vee J, (-1)^{\frac{B(IJ)+1}{2}} \sin(\theta_J) O_I^{(m)} \right), \quad (\text{A1})$$

where f denotes the index-distribution function defined in Eq. (7). After all elements of $O^{(m)}$ have been processed, process m will have generated update lists Δ_r^m for each r , representing the contributions that process r must apply due to Pauli strings originally held by m . Each process then writes its update list Δ_r^m into a remote memory window associated with process r . Finally, each process m gathers and applies the updates addressed to it, i.e., the union $\cup_{r=0}^{N-1} \Delta_r^m$, to its local operator $O^{(m)}$. These procedures are schematically summarized in Figs. 1(c) and 1(d). This yields a globally consistent update to the operator O , with contributions uniformly distributed across all N processes.

In Algorithm 1, we present the pseudocode for our parallel method, describing the operations performed by process m during a single layer of quantum gates. The algorithm captures the essential structure, including the sequence of computational and communication steps, while omitting certain implementation-specific details. In particular, we do not show the specifics of the truncation strategy nor the procedure used to evaluate and broadcast the global maximum value of $|O_I|$ across all processes. We also omit technical aspects such as the additional communication required to allocate the appropriate buffer size for the remote memory access (RMA) window $\tilde{\Delta}$. The symbols $\Leftarrow$ and $\Rightarrow$ are used to denote the insertion and deletion of elements in a list or hash map, respectively.

ALGORITHM 1. Parallel ORQA Circuit Simulation.

Require: Partial operator representation $O^{(m)} = \{(I, O_I^{(m)}), \dots\}$
Require: Quantum circuit $U = \{(J, \theta_J), \dots\}$
Require: Distribution map $f: \{0, 1\}^{2n} \rightarrow \mathbb{N}_N$
Require: Truncation threshold ϵ

```

for  $(J, \theta_J)$  in  $U$  do                                ▷ Loop over quantum gates
  Initialize update buffers:  $\Delta_r^m \leftarrow \emptyset \ \forall r \in \mathbb{N}_N$ 
  for  $(I, O_I^{(m)})$  in  $O^{(m)}$  do                            ▷ Loop over Paulis trings
     $\phi \leftarrow B(I, J)$ 
    if  $\phi$  is odd then                                       ▷ Calculate updates
       $O_I^{(m)} \leftarrow \cos(\theta_J) O_I^{(m)}$ 
       $\Delta_{f(I \vee J)}^m \Leftarrow (I \vee J, (-1)^{\frac{\phi+1}{2}} \sin(\theta_J) O_I^{(m)})$ 
    end if
  end for
  Allocate RMA windows:  $\tilde{\Delta}_r^m \leftarrow \emptyset \ \forall r \in \mathbb{N}_N$ 
  for  $r$  in  $\mathbb{N}_N$  do                                         ▷ Distribute updates
    if  $\Delta_r^m \neq \emptyset$  then
      MPI_Put  $\Delta_r^m \rightarrow \tilde{\Delta}_r^m$ 
    end if
  end for
  for  $(I, \delta)$  in  $\cup_r \tilde{\Delta}_r^m$  do                               ▷ Apply received updates
    if  $I \notin O^{(m)}$  then
       $O^{(m)} \Leftarrow (I, \delta_I)$ 
    else
       $O_I^{(m)} \leftarrow O_I^{(m)} + \delta_I$ 
    end if
  end for
  for  $(I, O_I^{(m)})$  in  $O^{(m)}$  do                               ▷ Truncate
    if  $|O_I^{(m)}| \leq \epsilon$  then
       $O^{(m)} \Rightarrow (I, O_I^{(m)})$ 
    end if
  end for
end for

```

APPENDIX B: HASH MAP REPRESENTATION

An essential aspect of our implementation is the choice of data structure used to store the partial operator $O^{(m)}$. To this end, we employ a hash map. The suitability of a given hash map depends on several performance-critical criteria that vary by application. In our case, the data structure must satisfy the following three key requirements: First, it must support efficient read, write, and erase operations, even when managing more than 10^7 elements. Second, it must incur minimal memory overhead, as memory usage constitutes a primary bottleneck in large-scale simulations. Third, it must provide high iteration speed, since the algorithm loops over the entire contents of $O^{(m)}$ twice per quantum gate applied. While a custom-designed hash map tailored specifically to the ORQA formalism could, in principle, offer further performance gains, we found that the open-source library *emhash8* [33] meets all three of the above requirements. Among the various alternatives that we tested, *emhash8* demonstrated the best overall

performance in terms of memory efficiency, access speed, and iteration throughput.

APPENDIX C: DISTRIBUTION MAP

The distribution map f introduced in Eq. (7) deterministically assigns each Pauli string to one of the available computational processes, with the goal of achieving an approximately uniform distribution. A straightforward and viable choice for f would be a deterministic, noncryptographic hash function. However, a more subtle yet equally significant requirement is the sparsity of the update pattern Δ_r^m —that is, each process m should ideally communicate with only a small subset of other processes r , while still maintaining global load balance. This sparsity becomes increasingly crucial at large scales: as the number of processes grows, minimizing the total number of interprocess communication calls becomes more significant than reducing the size of individual messages. Standard hash functions typically yield a uniform distribution of message sizes across all destinations, but they also tend to produce a large number of small messages sent to many different targets. This pattern can congest the communication network and introduce significant latency due to message handling overhead. To mitigate this, we adopt a more structured, enumeration-based scheme for f in place of a generic hash function. Details of our mapping strategy are described below.

Given a multi-index I , we define the distribution map f as

$$f(I) = \left[\sum_{j=0}^{\lceil \frac{2n}{k} \rceil - 1} I^{(j,k)} \right] \bmod N, \quad (\text{C1})$$

where n is the number of qubits, N is the number of computational processes, and k is an integer block size parameter chosen such that 2^k is ideally close to N . Here, $I^{(j,k)}$ denotes the j th k -bit segment of the $2n$ -bit multi-index I , counted from the least significant bits (i.e., from the right in our convention), and interpreted as an integer (e.g., $00_2 = 0$, $01_2 = 1$, $10_2 = 2$, and $11_2 = 3$ when $k = 2$). This procedure partitions I into $\lceil 2n/k \rceil$ blocks of k bits, converts each block to an integer, sums the resulting values, and applies a modulo- N reduction. This mapping approximately distributes all possible Pauli strings uniformly across the N processes while promoting communication sparsity—that is, each process tends to interact with only a small subset of other processes.

To understand this, consider the set $P_m = \{I \in \{0,1\}^{2n} | f(I) = m\}$, which represents all possible Pauli strings assigned to process m . When a quantum gate generated by σ_j is applied, the multi-index is updated as $I \underline{\vee} J$, and the new target process becomes

$$f(I \underline{\vee} J) = m + \sum_{j=1}^{2|J|} \pm 2^{J_j \bmod k} \bmod N, \quad (\text{C2})$$

where $|J|$ is the number of nonidentity local Pauli operators in σ_J , and J_j denotes the position of the j th such operator in the multi-index J .

As an illustrative example, consider the four-qubit Pauli string $\sigma_z \sigma_0 \sigma_z \sigma_0$. Here, $|J| = 2$ since two nonidentity Pauli operators are present: $\sigma_z^{(1)}$ and $\sigma_z^{(3)}$, counting from the right. The full multi-index of this Pauli string is 11001100_2 so that possible bit flips occur only at positions $J_1 = 2$, $J_2 = 3$, $J_3 = 6$, and $J_4 = 7$, again counting from the right and starting from 0. In this case, the number of distinct target processes $f(I \underline{\vee} J)$ is bounded by $2^{2|J|} + 1 = 17$ (depending on N and k), which represents a substantial reduction compared with the N possible destinations that would arise when using a typical hash function. Thus, Eq. (C1) achieves sparse interprocess communication while maintaining an approximately uniform global distribution of Pauli strings.

To demonstrate that the distribution map f produces an overall uniform distribution of Pauli strings while maintaining communication sparsity, we present in Fig. 4 the ratio between the largest and smallest values of $|O^{(m)}|$ across all processes, plotted as a function of the minimum value $\min_m(|O^{(m)}|)$ observed in a representative simulation. When this ratio approaches unity, the distribution of Pauli strings is considered effectively uniform. For small-scale simulations involving on the order of 10^3 Pauli strings, this ratio can reach values as high as 4, or even higher for fewer Pauli strings, indicating some degree of load imbalance. However, such imbalance is relevant only in the regime of very small problem sizes and has negligible impact on large-scale performance. As the number of Pauli strings per process increases, the ratio

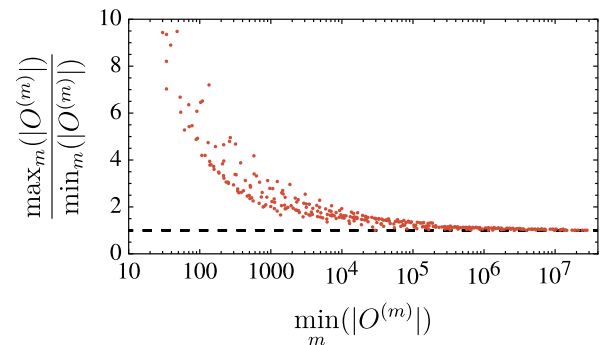


FIG. 4. Uniformity of Pauli string distribution across processes. Ratio between the maximum and minimum number of Pauli strings among all partial representations $O^{(m)}$, plotted as a function of the minimum value $\min_m(|O^{(m)}|)$. Values approaching unity indicate a more uniform distribution of Pauli strings across the parallel processes.

consistently converges toward 1, indicating improved load balancing. In particular, once the number of retained Pauli strings exceeds approximately 10^5 per process, the distribution becomes sufficiently uniform for practical purposes. In large-scale scenarios, such as simulations executed on the supercomputer Fugaku, it is advisable to target workloads involving between 10^6 and 10^8 Pauli strings per process. In this regime, the distribution achieved by the distribution map f is effectively uniform, and the associated communication overhead becomes negligible.

Although not employed in the present study, the uniformity of the distribution map can be further improved by modifying the baseline distribution map f to a perturbed version:

$$g_s(I) = f(I) + (h(I) \bmod s) \bmod N, \quad (\text{C3})$$

where $h(I)$ is a deterministic, noncryptographic hash function, and s is a small positive integer. This extension introduces mild stochasticity into the output of f , effectively randomizing the distribution while largely preserving the communication sparsity provided by the original map. The limiting behavior of g_s is well controlled: when $s = 1$, we recover the original mapping $g_1(I) = f(I)$; for $s = N$, the distribution approximates that of a conventional hash function. In this work, we adopt the unperturbed case $g_1(I) = f(I)$, which provides a favorable trade-off between communication sparsity and implementation simplicity. Nonetheless, choosing a small value of $s > 1$ may offer improved load balancing in other settings, and exploring this direction could be valuable for future applications.

APPENDIX D: TRUNCATION

For a given physical problem, it is generally difficult to predict in advance how many Pauli strings must be retained in representing the time-evolved operator to achieve a desired level of accuracy. It is therefore essential to employ a truncation heuristic that selectively discards coefficients deemed negligible. Although such truncation inevitably introduces numerical error, it is crucial for keeping the effective complexity of the operator within computationally tractable bounds. The overarching objective is to maximize accuracy while minimizing the number of retained Pauli strings. We consider the development of more advanced truncation strategies to be a promising direction for further improving both the efficiency and accuracy of our method. As a baseline, in this work, we analyze the simple amplitude-based criterion defined in Eq. (8).

However, it is *a priori* not clear how many Pauli strings will be retained for a given choice of the truncation threshold ϵ_0 . In practice, we begin with simulations at relatively large truncation thresholds, typically $\epsilon_0 \simeq 10^{-3}$,

where only a small number of Pauli strings is retained. These simulations are computationally inexpensive and can be efficiently repeated for progressively smaller values of ϵ_0 , typically decreasing in orders of magnitude. This procedure allows us to estimate the number of retained Pauli strings for a given circuit and choice of ϵ_0 and thereby determine the required computational resources. A more systematic approach to selecting ϵ_0 and quantifying its relation to the effective complexity depends on the specific truncation strategy, which is itself heuristic. A comprehensive study of truncation strategies and their impact on the effective complexity of our method is a substantial task that we leave for future work.

An interesting feature of the truncation procedure emerges in the regime where a large number of Pauli strings are retained: the distribution of their coefficients begins to exhibit stochastic characteristics. As the number of retained Pauli strings increases, the values of their coefficients tend to form a continuous distribution. To illustrate this behavior, we examine the normalized histogram of the coefficients O_I in the time-evolved operator O , obtained from a representative simulation after t circuit layers. We refer to this histogram as the Pauli string density, denoted $D_t(x)$, which characterizes the coefficient distribution in the presence of truncation. To account for the overall decay in the magnitude of O_I over time, we introduce a rescaled variable $x = O_I / \max_I |O_I|$. Under this normalization, the truncation threshold in Eq. (8) corresponds to a sharp cutoff at $|x| = \epsilon_0$.

In Fig. 5, we present the Pauli string density $D_t(x)$ as a function of time t for a representative quantum circuit simulation with $\theta_x = 0.1\pi$ and a truncation threshold of $\epsilon_0 = 10^{-6}$. At early times, the limited number of Pauli strings is evident from the appearance of discrete, high-density peaks in the histogram. As the simulation progresses, the number of retained Pauli strings increases, and

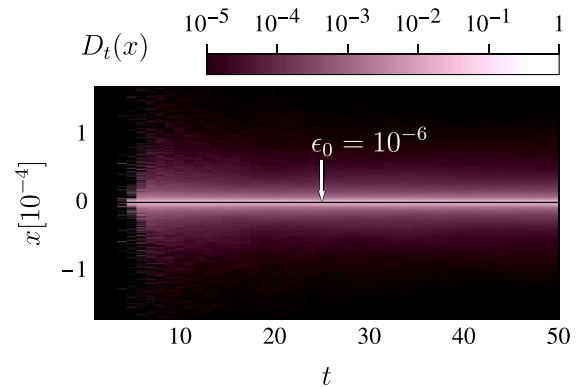


FIG. 5. Dynamics of the Pauli string density. Normalized histogram $D_t(x)$ of the rescaled Pauli string coefficients $x = O_I / \max_I |O_I|$ as a function of time t , computed for $\theta_x = 0.1\pi$ with truncation threshold $\epsilon_0 = 10^{-6}$. The sharp cutoff at $|x| \leq \epsilon_0$ reflects the applied truncation.

the histogram gradually evolves into a smoother, continuous distribution. A sharp cutoff appears at $|x| \leq \epsilon_0$, reflecting the imposed truncation threshold. The shape of the density distribution is governed by the dynamics of the underlying physical system and can vary significantly across different models. Consequently, the effectiveness of truncation is highly sensitive to the system-specific structure of the Pauli coefficient distribution. We anticipate that a deeper understanding of such distributions in various physical contexts will be essential for developing more effective and adaptive truncation strategies.

APPENDIX E: FURTHER DEMONSTRATION

For the purpose of further demonstrating the versatility of our algorithm, we present a second example on a smaller system size, comparing our results with MPS and state-vector simulations. We consider the isotropic Heisenberg model

$$H = J \sum_{\langle i,j \rangle} (\sigma_x^i \sigma_x^j + \sigma_y^i \sigma_y^j + \sigma_z^i \sigma_z^j) + h \sum_j \sigma_z^j, \quad (\text{E1})$$

for which we construct a digitized (Floquet) version, i.e., a kicked Heisenberg model, analogous to the kicked Ising model discussed in Sec. III. Here, σ_y^j denotes the Pauli- Y operator acting on the j th qubit. In this example, a single time step is generated by the unitary operator

$$U = \prod_{\langle i,j \rangle} e^{-iJ\sigma_x^i \sigma_x^j} e^{-iJ\sigma_y^i \sigma_y^j} e^{-iJ\sigma_z^i \sigma_z^j} \prod_j e^{-ih\sigma_z^j}, \quad (\text{E2})$$

which defines the Trotterized time evolution. In both Eqs. (E1) and (E2), $\langle i, j \rangle$ denotes pairs of nearest-neighbor qubits on a fragment of three loops of the heavy-hexagonal lattice consisting of $n = 28$ qubits, as shown in Fig. 6(a). We set the parameters to $J = -h$. We simulate $t = 30$ layers of U using our algorithm and compare the results with MPS simulations with bond dimensions up to $\chi = 4000$, as well as exact state-vector calculations. For this purpose, we consider the expectation value

$$\langle m_t \rangle = \langle \psi | (U^\dagger)^t \sigma_z^{16} (U)^t | \psi \rangle, \quad (\text{E3})$$

where $|\psi\rangle = \bigotimes_j |q_j\rangle$ is a product state of $|0\rangle$ and $|1\rangle$ forming a checkerboard pattern, as shown in Fig. 6(a), with $q_j = 0$ on white sites and $q_j = 1$ on black sites. The initial operator σ_z^{16} is located at the highlighted site at the center of the lattice.

Figure 6(b) shows the expectation value $\langle m_t \rangle$ up to $t = 30$ for $h = 0.48$ (blue), 0.44 (green), and 0.4 (red). The solid lines correspond to the ORQA results, while the dashed lines represent the MPS results. The brightness of the colors indicates the numerical complexity of the

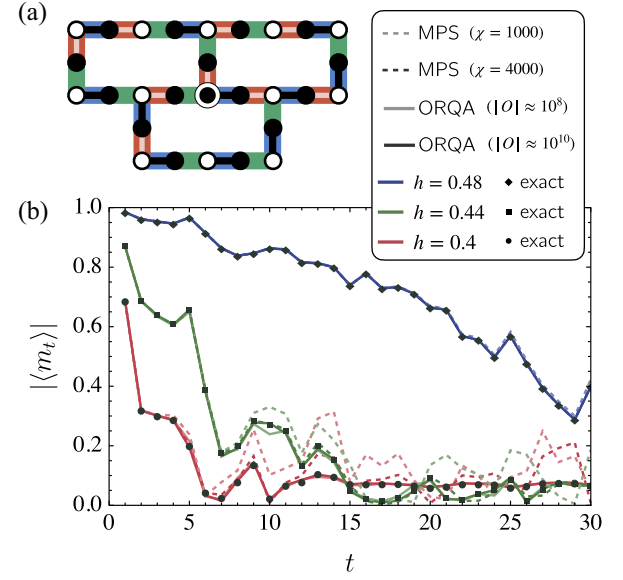


FIG. 6. Comparison of ORQA and MPS simulations of the kicked Heisenberg model defined by the Floquet unitary operator U in Eq. (E2). (a) Fragment of the heavy-hexagonal lattice with $n = 28$ qubits on which the Trotterized dynamics is simulated. White and black circles represent qubits initialized in the $|0\rangle$ and $|1\rangle$ states in $|\psi\rangle$, respectively. Two-qubit gates are applied in parallel between qubits connected by bonds of the same color. The highlighted central qubit is the site at which the initial local magnetization σ_z^{16} is defined. (b) Expectation value $\langle m_t \rangle$ obtained from the ORQA (solid lines) and MPS (dashed lines) simulations for three values of h up to $t = 30$. The brightness of the lines indicates the numerical complexity, corresponding to the number of retained Pauli strings $|O|$ in ORQA and the bond dimension χ in MPS. Exact state-vector results are shown as black dots.

simulations, corresponding to the number of retained Pauli strings $|O|$ in ORQA and the bond dimension χ in MPS. For reference, results obtained from exact state-vector simulations are shown as black dots. For $h = 0.48$, both ORQA and MPS perform very well, with only the MPS simulations at $\chi = 1000$ showing slight deviations at later times. In contrast, for $h = 0.44$ and $h = 0.4$, ORQA exhibits significantly improved convergence compared with MPS. In particular, for $h = 0.4$, MPS calculations with a maximum bond dimension of $\chi = 4000$ still show noticeable deviations, whereas even a smaller ORQA simulation with approximately $|O| \approx 10^8$ Pauli strings already achieves excellent agreement with the exact results. This demonstrates that ORQA can achieve higher accuracy than MPS at comparable computational cost in this regime.

We further present results of ORQA simulations of the staggered magnetization,

$$\langle A_t \rangle = \frac{1}{n} \langle \psi | (U^\dagger)^t \left[\sum_j (-1)^{q_j} \sigma_z^j \right] (U)^t | \psi \rangle, \quad (\text{E4})$$

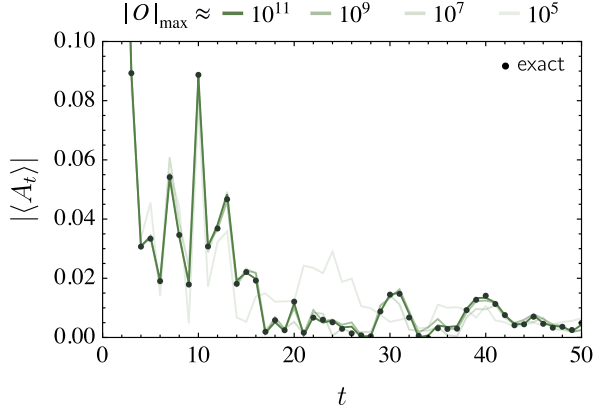


FIG. 7. ORQA simulations of the expectation value of the staggered magnetization $\langle A_t \rangle$ in Eq. (E4) for $h = 0.44$ on the heavy-hexagonal lattice with $n = 28$ qubits [see Fig. 6(a)]. The number of retained Pauli strings $|O|$ ranges from 10^5 to 10^{11} , as indicated by the brightness of the solid lines, corresponding to truncation thresholds from $\epsilon_0 = 10^{-3}$ down to $\epsilon_0 = 10^{-6}$. Exact state-vector results are shown as black dots.

to demonstrate the capability of our method to handle more involved initial operators. Figure 7 shows results up to $t = 50$, with the number of retained Pauli strings ranging from 10^5 to 10^{11} , as indicated by the brightness of the solid lines. For comparison, exact state-vector results are also shown as black dots. Although the staggered magnetization takes very small values at later times, the ORQA simulations show reliable convergence to the exact results. Notably, later times do not necessarily converge more slowly than earlier times, indicating that truncation errors do not simply accumulate over time but instead emerge at specific times depending on the dynamics and the truncation heuristic employed in ORQA. This further supports the robustness of ORQA in capturing nontrivial operator dynamics.

APPENDIX F: ERROR ANALYSIS

We validate the correctness of our algorithm by setting the truncation threshold to $\epsilon_0 = 0$ and comparing the results with those obtained from an exact state-vector

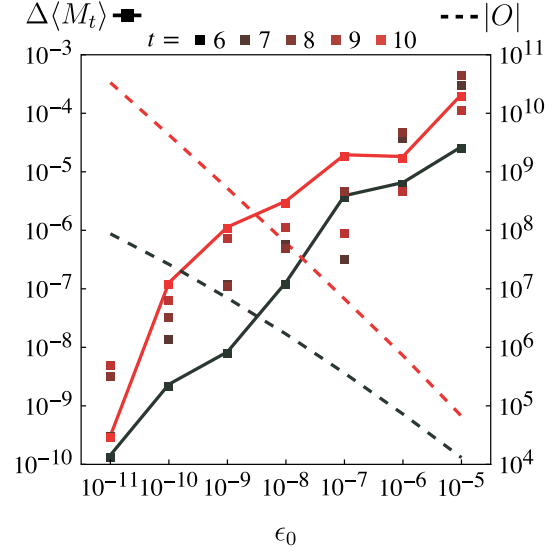


FIG. 8. Residual of the magnetization $\Delta \langle M_t \rangle$ after t time steps as a function of the truncation threshold ϵ_0 for the kicked Ising model with $\theta_x = 0.9\pi$ on a heavy-hexagonal lattice with $n = 127$ qubits. The residual is defined with respect to a reference simulation using $\epsilon_0 = 10^{-12}$. Results are shown for all time steps from $t = 6$ (darkest color) to $t = 10$ (brightest color). The data for $t = 6$ and $t = 10$ are connected by solid lines to guide the eye. For these cases, the corresponding effective complexity $|O|$ is also shown as dashed lines.

simulation for a tractable system size. Specifically, we simulate the first five time steps of the same system studied in Sec. III, using $\theta_x = 0.9\pi$. During this time evolution, the causal light cone reaches a maximum extent of 31 qubits, as illustrated in Fig. 1(a). Theoretically, our method becomes exact in the limit $\epsilon_0 \rightarrow 0$. Indeed, for $\epsilon_0 = 0$, we find that our results agree with the exact state-vector simulation within numerical double-precision accuracy (see Table I). Furthermore, even at a small but finite truncation threshold $\epsilon_0 = 10^{-5}$, the results remain in excellent agreement with the exact solution, as shown in Table I. This confirms that the truncation procedure is correctly implemented and does not introduce spurious errors for sufficiently small values of ϵ_0 .

TABLE I. Magnetization $\langle M_t \rangle$ for the kicked Ising model with $\theta_x = 0.9\pi$ up to $t = 5$ computed using an exact state-vector simulation and our ORQA method with truncation thresholds $\epsilon_0 = 0$ and $\epsilon_0 = 10^{-5}$. Excellent agreement is observed for $\epsilon_0 = 0$, and the results for $\epsilon_0 = 10^{-5}$ remain close to the exact values, validating the accuracy of our implementation.

t	State vector	$\epsilon_0 = 0$	$\epsilon_0 = 10^{-5}$
1	-0.95105651629515	-0.95105651629515	-0.95105651629515
2	0.90450849718747	0.90450849718747	0.90450849718747
3	-0.9423841865631	-0.9423841865631	-0.9423841865631
4	0.97436799430392	0.97436799430396	0.97438562148434
5	-0.95315824248959	-0.95315824248965	-0.95311531321393

We continue this analysis and further study the same setup up to $t = 10$, for which exact solutions are unavailable. Here, we compare the residual of the magnetization $\langle M_t \rangle$ obtained with a truncation threshold ϵ_0 to that of a reference simulation using $\epsilon_0 = 10^{-12}$, defined as

$$\Delta \langle M_t \rangle = |\langle M_t \rangle_{\epsilon_0} - \langle M_t \rangle_{\epsilon_0=10^{-12}}|. \quad (\text{F1})$$

Figure 8 shows this residual as a function of ϵ_0 on a log-log scale. The corresponding effective complexity $|O|$ at $t = 6$ and $t = 10$ is also shown as dashed lines. In these calculations, the residual exhibits systematic convergence as the truncation threshold ϵ_0 is decreased. This demonstrates that the truncation error is well controlled and can be systematically reduced.

APPENDIX G: COMPUTATIONAL ENVIRONMENT

The simulations presented in this work were performed on the supercomputer Fugaku, which comprises 414 racks, each containing 384 compute nodes, for a total of 158 976 nodes. Each node is equipped with 48 computational cores and 32 GiB of memory. Because memory usage constitutes a primary constraint in our implementation, we utilized between 8 and 22 computational processes per node, assigning one process per core. This allocation provides each process with approximately 1–4 GiB of memory, sufficient to store on the order of 10^7 Pauli strings, including all associated memory overhead. In our largest simulations, we employed up to 12 288 nodes, corresponding to a total of up to $2^{18} = 262,144$ parallel processes. This configuration allows us to efficiently simulate operator dynamics involving more than 10^{12} Pauli strings within practical wall-time limits. Remarkably, even at this scale, only a few percent of the total wall time is spent on interprocess communication, underscoring the scalability and efficiency of our implementation.

[1] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. Brandao, D. A. Buell *et al.*, Quantum supremacy using a programmable superconducting processor, *Nature (London)* **574**, 505 (2019).
 [2] Y. A. Liu, X. L. Liu, F. N. Li, H. Fu, Y. Yang, J. Song, P. Zhao, Z. Wang, D. Peng, H. Chen, C. Guo, H. Huang, W. Wu, and D. Chen, Closing the “quantum supremacy” gap: Achieving real-time simulation of a random quantum circuit using a new Sunway supercomputer, in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '21* (Association for Computing Machinery, New York, NY, USA, 2021).
 [3] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. van den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme, and

A. Kandala, Evidence for the utility of quantum computing before fault tolerance, *Nature (London)* **618**, 500 (2023).
 [4] R. Acharya, D. A. Abanin, L. Aghababaie-Beni, I. Aleiner, T. I. Andersen, M. Ansmann, F. Arute, K. Arya, A. Asfaw, N. Astrakhantsev *et al.*, Quantum error correction below the surface code threshold, *Nature (London)* **638**, 920 (2025).
 [5] J. Preskill, Quantum computing in the NISQ era and beyond, *Quantum* **2**, 79 (2018).
 [6] K. Bharti, A. Cervera-Lierta, T. H. Kyaw, T. Haug, S. Alperin-Lea, A. Anand, M. Degroote, H. Heimonen, J. S. Kottmann, T. Menke, W.-K. Mok, S. Sim, L.-C. Kwek, and A. Aspuru-Guzik, Noisy intermediate-scale quantum algorithms, *Rev. Mod. Phys.* **94**, 015004 (2022).
 [7] Y. Alexeev, M. Amsler, M. A. Barroca, S. Bassini, T. Battelle, D. Camps, D. Casanova, Y. J. Choi, F. T. Chong, C. Chung *et al.*, Quantum-centric supercomputing for materials science: A perspective on challenges and future directions, *Future Gener. Comput. Syst.* **160**, 666 (2024).
 [8] J. Robledo-Moreno, M. Motta, H. Haas, A. Javadi-Abhari, P. Jurcevic, W. Kirby, S. Martiel, K. Sharma, S. Sharma, T. Shirakawa, I. Sitdikov, R.-Y. Sun, K. J. Sung, M. Takita, M. C. Tran, S. Yunoki, and A. Mezzacapo, Chemistry beyond the scale of exact diagonalization on a quantum-centric supercomputer, *Sci. Adv.* **11**, eadu9991 (2025).
 [9] H.-J. Liao, K. Wang, Z.-S. Zhou, P. Zhang, and T. Xiang, Simulation of IBM’s kicked Ising experiment with projected entangled pair operator, *arXiv:2308.03082*.
 [10] M. S. Rudolph, E. Fontana, Z. Holmes, and L. Cincio, Classical surrogate simulation of quantum systems with LOWESA, *arXiv:2308.09109*.
 [11] J. Tindall, M. Fishman, E. M. Stoudenmire, and D. Sels, Efficient tensor network simulation of IBM’s eagle kicked ising experiment, *PRX Quantum* **5**, 010308 (2024).
 [12] S. Patra, S. S. Jahromi, S. Singh, and R. Orús, Efficient tensor network simulation of IBM’s largest quantum processors, *Phys. Rev. Res.* **6**, 013326 (2024).
 [13] T. Begošić, J. Gray, and G. K.-L. Chan, Fast and converged classical simulations of evidence for the utility of quantum computing before fault tolerance, *Sci. Adv.* **10**, eadk4321 (2024).
 [14] T. Begošić, J. Gray, and G. K.-L. Chan, Fast and converged classical simulations of evidence for the utility of quantum computing before fault tolerance, *Sci. Adv.* **10**, eadk4321 (2024).
 [15] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd, Quantum machine learning, *Nature (London)* **549**, 195 (2017).
 [16] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, Challenges and opportunities in quantum machine learning, *Nat. Comput. Sci.* **2**, 567 (2022).
 [17] S. R. White, Density matrix formulation for quantum renormalization groups, *Phys. Rev. Lett.* **69**, 2863 (1992).
 [18] S. Östlund and S. Rommer, Thermodynamic limit of density matrix renormalization, *Phys. Rev. Lett.* **75**, 3537 (1995).
 [19] U. Schollwöck, The density-matrix renormalization group in the age of matrix product states, *Ann. Phys. (Amsterdam)* **326**, 96 (2011).

- [20] S. Paeckel, T. Köhler, A. Swoboda, S. R. Manmana, U. Schollwöck, and C. Hubig, Time-evolution methods for matrix-product states, *Ann. Phys. (Amsterdam)* **411**, 167998 (2019).
- [21] F. Verstraete and J. I. Cirac, Renormalization algorithms for quantum-many body systems in two and higher dimensions, [arXiv:Cond-Mat/0407066](#).
- [22] F. Verstraete and J. I. Cirac, Valence-bond states for quantum computation, *Phys. Rev. A* **70**, 060302 (2004).
- [23] R. Orús, Tensor networks for complex quantum systems, *Nat. Rev. Phys.* **1**, 538 (2019).
- [24] J. I. Cirac, D. Pérez-García, N. Schuch, and F. Verstraete, Matrix product states and projected entangled pair states: Concepts, symmetries, theorems, *Rev. Mod. Phys.* **93**, 045003 (2021).
- [25] E. Fontana, M. S. Rudolph, R. Duncan, I. Rungger, and C. Cîrstoiu, Classical simulations of noisy variational quantum circuits, *npj Quantum Inf.* **11**, 84 (2025).
- [26] T. Begušić and G. K.-L. Chan, Real-time operator evolution in two and three dimensions via sparse Pauli dynamics, *PRX Quantum* **6**, 020302 (2025).
- [27] L. Broers and L. Mathey, Exclusive- or encoded algebraic structure for efficient quantum dynamics, *New J. Phys.* **28**, 044508 (2026).
- [28] N. Loizeau, J. C. Peacock, and D. Sels, Quantum many-body simulations with paulistrings.jl, *SciPost Phys. Codebases* **54** (2025).
- [29] M. S. Rudolph, T. Jones, Y. Teng, A. Angrisani, and Z. Holmes, Pauli propagation: A computational framework for simulating quantum systems, [arXiv:2505.21606](#).
- [30] S. Aaronson and D. Gottesman, Improved simulation of stabilizer circuits, *Phys. Rev. A* **70**, 052328 (2004).
- [31] J. M. Chow, J. M. Gambetta, A. D. Córcoles, S. T. Merkel, J. A. Smolin, C. Rigetti, S. Poletto, G. A. Keefe, M. B. Rothwell, J. R. Rozen, M. B. Ketchen, and M. Steffen, Universal quantum gate set approaching fault-tolerant thresholds with superconducting qubits, *Phys. Rev. Lett.* **109**, 060501 (2012).
- [32] P. Rall, D. Liang, J. Cook, and W. Kretschmer, Simulation of qubit quantum circuits via Pauli propagation, *Phys. Rev. A* **99**, 062337 (2019).
- [33] ktprime *et al.*, Emhash—fast and memory efficient open addressing C++ flat hash table/map (2025).