

AppQSim: Application-oriented benchmarks for Hamiltonian simulation on a quantum computer

Etienne Granet¹* and Henrik Dreyer¹*Quantinuum, Leopoldstrasse 180, 80804 Munich, Germany*

(Received 2 April 2025; accepted 25 September 2025; published 10 November 2025)

We introduce AppQSim, a benchmarking suite for quantum computers focused on applications of Hamiltonian simulation. We consider five different settings for which we define a precise task and score: condensed matter and material simulation (dynamic and static properties), nuclear magnetic resonance simulation, chemistry ground-state preparation, and classical optimization. These five different benchmark tasks display different resource requirements and scalability properties. We introduce a metric to evaluate the quality of the output of a tested quantum hardware, called distinguishability cost, defined as the minimal number of gates that a perfect quantum computer would have to run to certify that the output of the benchmarked hardware is incorrect.

DOI: [10.1103/qt55-7d6r](https://doi.org/10.1103/qt55-7d6r)

I. INTRODUCTION

Quantum computing hardware has recently witnessed rapid and impressive improvements [1–5]. It has become clear that the difficulty of classically simulating quantum computers greatly depends on the circuits to run. While certain specific circuits are already impossible to simulate classically on the best hardware [2–4], many circuits that accomplish a useful, application-centered task can still be simulated as of today. For this reason, even though these difficult-to-simulate circuits give a certain measure of the overall power of a given hardware, they cannot be used to accurately evaluate their ability to solve concrete tasks. Now that the technology is moving from an “abstract” quantum advantage era to a “practical” quantum advantage era, the need for application-oriented benchmarks becomes more pressing.

The purpose of this paper is to introduce an application-oriented benchmarking suite for quantum computers focused on Hamiltonian simulation, called AppQSim. It has been partially incorporated into a more general application-oriented benchmarking suite called BenchQC [6]. We study different settings considered to be some of the promising applications of quantum computing, namely, material simulation, quantum chemistry, Nuclear Magnetic Resonance (NMR) simulation, and classical optimization. For example, the benchmarks we define cover applications such as the simulation of neutron-scattering experiments, the computation of spectrum generated by NMR experiments, or finding the maximal cut on a graph.

Focusing a benchmark metric on applications is somehow at odds with benchmark scalability, since benchmarking supposes to know the expected result, whereas relevant

applications of quantum computing are those beyond reach of classical computers. To deal with this, we proposed benchmarking settings with varied scalability properties and closeness to applications. We defined a number of criteria that an ideal application-oriented benchmark protocol should satisfy, and used these criteria as a guide for defining the benchmark settings. The characteristics of the five different benchmark settings we defined are summarized in Table I. In the following, we present briefly each of these protocols.

Section III describes the “flagship” benchmark of AppQSim, which is the computation of dynamic properties in conducting materials. We define a simulation setup protocol similar (but not identical) to the simulation of the Hubbard model, whose exact results can be classically computed in polynomial time. This guarantees the benchmark to be scalable. The quantities computed are those required to simulate neutron-scattering experiments, yielding an almost end-to-end application-oriented benchmark. We introduce a score called “distinguishability cost” to measure the quality of the benchmarked hardware, which is the minimal number of gates to run on a perfect quantum computer to be able to affirm that the output of the benchmarked hardware is incorrect. Stated differently, this measures the number of computations that the benchmarked hardware can do while staying indistinguishable from a perfect hardware. This score is a physical and meaningful number that does not require context to be interpreted, and directly informs the end user of how noisy a given hardware is for a given application.

In Sec. IV, we introduce another material-simulation benchmark focused on equilibrium state preparation. We use Hamiltonian simulation to prepare adiabatically a low-energy equilibrium state of the Heisenberg model on a kagome lattice. This kind of adiabatic preparation of low-temperature state is known to display lower sensitivity to hardware noise [7–9], probing different capacities of the hardware. Despite the exact result being exponentially costly to compute classically, the difficulty of the preparation of the ground state in this highly quantum model ensures that the benchmark will remain relevant for years to come. Moreover, even beyond the classically simulable

*Contact author: etienne.granet@quantinuum.com

TABLE I. Summary of the characteristics of the benchmarks in the AppQSim suite. *Benchmark scalability* means the classical resources required to assign a score to the hardware output, as a function of system size N . *Minimal hardware requirement* is the minimal number of qubits and two-qubit gate error rate, assuming all-to-all connectivity, to run the benchmark with nontrivial output score. *Ideal connectivity* indicates the hardware connectivity that is most suited to the benchmark. *Circuit geometry* indicates the aspect ratio of the circuits involved. *Random circuit* indicates whether several different random circuits have to be generated to run the benchmark. *Resource requirements* indicate the resources to run the benchmark, in terms of number of qubits, number of gates, and number of shots. Scalable means the number can be varied from low to high in the benchmark. Controllable means the number is left to be set by the user, depending on the characteristics of the machine, with the range of freedom indicated in parenthesis. Low means that the benchmark should be implementable on most available hardware, at the time of this paper. High means that the benchmark is currently too costly or at the limits of the capacities of all currently available hardware. *Comparison possible beyond exact result* indicates whether the benchmark can be used to compare different hardware, even in the regime where no exact solution can be computed classically.

	Material simulation (dynamic)	Material simulation (static)	Nuclear Magnetic Resonance	Quantum chemistry	Classical optimization
Benchmark scalability	Polynomial	Constant	Exponential, $N \lesssim 20$	Constant	Exponential, $N \lesssim 1000$
Minimal hardware requirement	6 qubits, any error rate	3 qubits, any error rate	7 qubits, error rate $< 10^{-3}$	4 qubits, any error rate	4 qubits, any error rate
Ideal connectivity	2D	2D	All-to-all	All-to-all	All-to-all
Circuit geometry	Square	Rectangle	Small width, large depth	Rectangle, controlled	Square
Random circuit	No	No	Yes	Yes	No
Resource requirements	Qubits: scalable Gates: scalable Shots: controllable (intermediate/high)	Qubits: scalable Gates: scalable Shots: low	Qubits: low Gates: high Shots: high	Qubits: scalable Gates: scalable Shots: controllable (intermediate/high)	Qubits: scalable Gates: scalable Shots: controllable (low/high)
Comparison possible beyond exact result	No	Yes	No	Yes	Yes

regime, the output of two different hardware can still be compared.

In Sec. V, we present a benchmark of NMR experiment simulation. This is a fully end-to-end application-oriented benchmark, with the score being the average precision that one can obtain on the couplings between the nuclear spins of a benzene molecule when comparing to an NMR experiment. The system sizes cannot be scaled arbitrarily, but the large circuit depth required guarantees again the benchmark to remain relevant for several years.

In Sec. VI, we then move on to the ground-state preparation of molecular systems. To bypass the prohibitive cost of energy measurement in these systems to a precision that cannot be obtained with classical computers, we adopt a mirror-circuit-like approach to define a score. This allows the end user to run the benchmark for arbitrary system sizes. This comes at the cost of a more abstract score not directly related to a quantity to measure in a concrete application. The benchmark also tests the ability of the hardware to generate and run random circuits.

Finally, in Sec. VII we present a benchmark for Hamiltonian simulation applied to classical optimization. The benchmark is not a variational algorithm (as is often implemented), but instead a deterministic heuristic protocol to solve Max-Cut that has been observed to work to at least around 100 qubits. The score directly measures the ability of the quantum computer to find the exact optimal value, and is thus directly application-oriented. Specific classical optimization

algorithms can solve the problem up to the order of 1000 qubits, which guarantees the relevance of the benchmark for a long time, at least up to the time where practical quantum advantage would be observed for that application. Even beyond the classical simulability, the output of different hardware can still be compared.

Before detailing the precise protocols in these benchmarks, we present in Sec. II a brief overview of existing benchmarks, as well as the five criteria that we defined for an ideal application-oriented benchmark.

II. PREVIOUS WORKS AND GOALS

A. Overview of previous works

There exist three main approaches to evaluate the quality of quantum hardware. The first approach is a *low-level benchmark*, where one directly measures the quality of basic hardware components or operations such as gate fidelity or state preparation and measurement errors. Well-established approaches are randomized benchmarking [10,11], gate set tomography [12], or cycle benchmarking [13]. While these metrics provide a detailed quality assessment of the basic components of the hardware, the overall performance of an algorithm results from a complex interaction of all these error sources. These interactions can further depend on the structure of the circuit implemented and on higher-level hardware characteristics such as connectivity or speed. There can be very

significant differences in performance for different tasks with same hardware resources.

A second approach to hardware quality assessment is *circuit benchmarks*, where an entire circuit is run on the hardware, instead of individual operations on isolated qubits. Well-known examples are quantum volume [14], generation of random bit strings [15,16], and protocols based on “mirror circuits” [17]. These circuit benchmarks capture different characteristics of the hardware in a holistic way and give a better idea of its overall capacities. However, they do not capture how much of a certain noise feature a given application can tolerate.

The third approach to hardware benchmarking is *application-oriented benchmarks*. These benchmarks directly evaluate the ability of the hardware to solve a given real application. There already exist several application-oriented scores and benchmark suites. Benchmarks focused on simulation of physical systems include, for example, preparing the ground state of the one-dimensional Fermi-Hubbard model using Variational Quantum Eigensolver (VQE) [18,19] or the ground state of small molecules using VQE [20]. Certain benchmarks propose implementation of Hamiltonian simulation for specific systems [21,22]. Benchmarks on classical optimization applications include solving a Max-Cut problem with Quantum Approximate Optimization Algorithm [21,23], Max-Clique problems [24], some industry-relevant problems like the robot path and vehicle optimization problems [25], as well as other benchmarking suites containing multiple problem instances [26,27], or machine-learning problems [22]. Finally, some benchmarks include linear algebra routines such as Quantum Fourier Transform, quantum matrix inversion [28,29], linear equation solving [22], and quantum singular value transformation for Hamiltonian simulation [30]. There exist works specifically proposing benchmarking libraries for Hamiltonian simulation, but without specifying a particular task [31].

B. Goal of the present work

Most of these application-oriented benchmarks rely on VQE-like algorithms. These typically involve shallow circuits with limited number of gates, but require several different circuits and sometimes a large number of measurements to optimize the VQE parameters. The actual scalability and usefulness of these variational approaches have been put in question, with serious obstacles such as the hostile optimization landscape or the effect of noise [32]. It could render these benchmarks obsolete if they become impossible to implement on near-term devices.

In contrast, algorithms based on Hamiltonian simulation appear to be underrepresented in these benchmarks. Hamiltonian simulation consists in applying a time evolution operator e^{itH} on the qubit register, where t is some simulation time and H a Hamiltonian. It is proven to be implementable in polynomial time on a quantum computer with very simple routines like a Trotter decomposition. It appears in many algorithms, such as Quantum Phase Estimation and adiabatic state preparation, with applications ranging from material and molecular simulation to classical optimization. Despite the

high likelihood that Hamiltonian simulation will play a prominent role in the Noisy Intermediate-Scale Quantum (NISQ) era and beyond, there seems to be no application-oriented benchmark specifically devoted to it. The purpose of the AppQSim benchmarking suite that we introduce in this paper is to fill this gap.

Before entering the precise definition of the benchmarking tasks, we are going to define a number of criteria that we will follow to define AppQSim. We argue that an ideal benchmark protocol should have following characteristics:

(1) *Application-oriented*: The benchmark should evaluate as closely as possible the ability of a quantum computer to solve a concrete use case that end users would actually implement.

(2) *Scalable*: Given hardware results, the classical runtime for computing the score should be only polynomial in system parameters. This would ensure that the benchmark can be run even beyond the classically simulable regime.

(3) *Relatively cheap*: Benchmarking tends to always be resource-consuming. Yet, in practice, it does not take so many gates or so many shots for a user of quantum computers to get an idea of the limits of the hardware. An ideal benchmark protocol should thus not require unreasonably large resources.

(4) *Difficult to spoof*: If a benchmark protocol becomes widely adopted, users will try to optimize the score for their specific machine in any possible way within the benchmark protocol. The outcome of the benchmark protocol should thus not be easy to replicate with workarounds or by chance.

(5) *Hardware-agnostic*: Different quantum computing platforms can vastly differ in their specifications and abilities, such as number of qubits, native entangling operation, connectivity, coherence time, and speed. The benchmark should not favor any of these platforms beyond the quality of the output. If a same output quality can be obtained by large number of shots or high two-qubit gate fidelity, both approaches should lead to the same score.

As explained in the Introduction, these different properties for an ideal benchmark tend to be at odds with each other. For example, scalability often comes with less application-oriented benchmarks. If the output of a benchmarking task can be computed classically, it cannot be a direct application of quantum computers as an end user would just use this classical computation. Generic scalable benchmarking methods such as mirror circuits give in the end no information about the ability of the quantum hardware to compute specific application-relevant quantities. Similarly, ensuring that a benchmark is difficult to spoof tends to make it more expensive. As the output of a quantum computer is probabilistic by nature, ensuring that a good-quality outcome is not the result of randomness requires several repetitions, which increases the cost of implementing a benchmark.

In this paper, we are going to follow these five criteria to define benchmarking tasks for Hamiltonian simulation. Only two of the different applications presented will fulfill all of the five criteria. These are the free fermion benchmark presented in Sec. III, which is the “flagship” benchmark of this AppQSim suite, and the low-energy state preparation benchmark of Sec. IV. The three other benchmarks only partly fulfill this list of criteria.

III. APPLICATION: SIMULATION OF CONDUCTING MATERIALS

A. Context and motivation

Electrons in materials can be modeled by spinful fermions hopping from one atomic orbital to another. One of the most famous models for electrons in solids is the so-called Hubbard model. This model (or variants thereof) is believed to be able to describe high-temperature superconductivity of the cuprates whose pairing mechanism still has not been fully understood. For this reason, its solution has been the study of countless academic and industry work, and its utility has been estimated in the billions of dollars [33]. Since Hamiltonian simulation is one of the simplest tasks that a quantum computer is likely to be able to perform exponentially faster than a classical computer, it puts the simulation of the Hubbard model at the forefront of near-term applications of quantum hardware, in the NISQ era and beyond.

Mathematically, the Hamiltonian of the Hubbard model is

$$H_H = -t \sum_{\langle i,j \rangle, \sigma} (c_{i,\sigma}^\dagger c_{j,\sigma} + c_{j,\sigma}^\dagger c_{i,\sigma}) + V \sum_i \left(n_{i,\uparrow} n_{i,\downarrow} - \frac{1}{4} \right), \quad (1)$$

where $c_{i,\sigma}$ denotes the fermion annihilation operator at site $i = 1, \dots, N$ and spin $\sigma \in \{\uparrow, \downarrow\}$, which satisfy canonical anticommutation relations, where $n_{i,\sigma} = c_{i,\sigma}^\dagger c_{i,\sigma}$ is the mode occupation number, t and V some parameters, and $\langle i, j \rangle$ means that the two sites i, j are neighbors on the lattice considered.

There is no known classical algorithm to simulate the time evolution of a system described by the Hubbard model, except for small system sizes (with state vector simulations) or for short times (with tensor network or neural network techniques). From a benchmark perspective, this is of course problematic as the result of the quantum hardware cannot be compared to the exact result beyond these cases. Even in the NISQ era, quantum computers are able to reach settings that can become challenging for classical computers [3,34], and a benchmark specifically focused on classically simulatable regimes would be too restrictive. There is, however, a simple way of modifying the Hamiltonian, without much modifying the circuits run on the hardware, to make the simulation classically easier. In absence of interaction between the spins, i.e., when $V = 0$, the system describes free fermions, which is exactly solvable, and the computation time to classically simulate the system scales polynomially with system size and simulation time.

B. The benchmark

The benchmark that we propose is the implementation of the compact encoding of Ref. [35] for this free fermion system.

We consider a square lattice with widths L_x, L_y , thus containing $L = L_x L_y$ sites, and impose periodic boundary conditions. We will restrict to only L_x and L_y even. We add to this lattice $L/2$ ancillas positioned in every other face of the square lattice, in a checker-board pattern as illustrated in Fig. 1. The total system thus possesses $N = 3L/2$ sites. On this system,

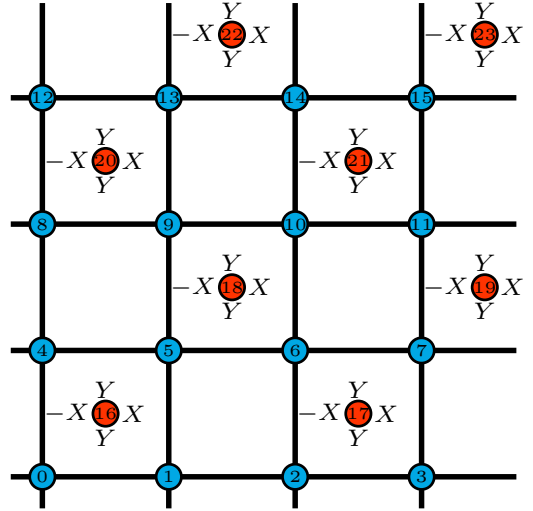


FIG. 1. Square lattice after the compact fermion encoding. Blue circles indicate sites of the original lattice and red circles indicate ancillas.

we define the following Hamiltonian:

$$H = \frac{1}{2} \sum_{\langle i,j \rangle} (X_i X_j + Y_i Y_j) P_a, \quad (2)$$

where the sum runs over all the edges $\langle i, j \rangle$ of the square lattice that links neighboring sites i, j , and where a refers to the ancilla that is contained in the face adjacent to edge $\langle i, j \rangle$. P_a is the Pauli matrix acting on the ancilla a , equal to $P_a = Y_a$ if $\langle i, j \rangle$ is a horizontal edge and equal to $P_a = X_a$ (resp. $-X_a$) if $\langle i, j \rangle$ is a vertical edge on the right (resp. left) of the ancilla. These three different possibilities are sketched in Fig. 1.

The initial state $|\psi\rangle$ that we consider is defined as follows. We initialize the L lattice sites in a product state in the Z basis, with a predefined value $n_j \in \{0, 1\}$ for each site j . We fix this function to be

$$n_j = \begin{cases} 1, & \text{if } j_y < L_y/2, \\ 0, & \text{if } j_y \geq L_y/2, \end{cases} \quad (3)$$

with $j_y = 0, \dots, L_y - 1$ denoting the vertical component of the site. The ancillas are initialized in a ground state of the toric code, as required by this fermionic encoding [35]. This is done as follows. For an ancilla a , we denote by a^1, a^2, a^3 the two ancillas on top left and top right of ancilla a , and the ancilla two lines on top of a in the same column, applying the boundary conditions in both vertical and horizontal directions. For example, in Fig. 1, if $a = 18$, then $a^1, a^2, a^3 = 20, 21, 22$. Then, we define the unitary operator

$$V_a = CX_{a,a^3} CX_{a,a^2} CX_{a,a^1} H_a, \quad (4)$$

with H_a denoting the Hadamard gate on ancilla a , and $CX_{a,b}$ the CNOT gate with control a and target b , with the first operator applied being the rightmost. Similarly, we define

$$\tilde{V}_a = CX_{a,a^2} CX_{a,a^4} CX_{a,a^5} H_a, \quad (5)$$

with a^4 being the ancilla on bottom right of ancilla a , and a^5 the ancilla two columns to the right of ancilla a on the same row, applying periodic boundary conditions in both directions.

For example, in Fig. 1, if $a = 16$, this is $a^4 = 22$ and $a^5 = 17$. We then apply the operators $V_a, \tilde{V}_a$ on some ancillas in a specific order, as prescribed and illustrated in Appendix A, in order to prepare the ground state of the toric code on the ancillas. In the particular case of system size 4×4 , applying the ordering of Appendix A, we would apply V_a on ancillas $a = 18, 19$ and then $\tilde{V}_a$ on $a = 16$. This operation defines the state

$$|\tilde{\psi}\rangle = \prod_a \tilde{V}_a \prod_a V_a \prod_{j=0, \dots, L-1} X_j^{n_j} |0\rangle, \quad (6)$$

where the product of $V_a, \tilde{V}_a$ is as specified in Appendix A. Finally, we define the initial state of the quantum computer as $|\psi\rangle$, where

$$|\psi\rangle = \prod_{a=1}^{L/2} Q_a |\tilde{\psi}\rangle, \quad (7)$$

where Q_a acts only on ancilla $a + L$, with $Q = HS^\dagger$ if the ancilla is on an odd row and $Q = SHS$ if the ancilla is on an even row, with H denoting here the Hadamard gate and S the usual S gate.

We note that this state preparation protocol also holds when one of the lengths L_x or L_y is equal to 2, applying strictly the periodic boundary conditions in the operator V_a . For example, in size $L_x = 2, L_y = 4$, the operator V_9 applies a CNOT from site 9 to site 10, 10 again, and then 11, which means a single CNOT from site 9 to site 11.

To entirely describe our protocol, we now define the precise Trotterization to use in the benchmark. For a given Trotter step size δt , each Trotter step operator U is decomposed as

$$U = U_{|,2} U_{|,1} U_{-,2} U_{-,1}. \quad (8)$$

Here, we defined

$$U_{-,1} = \exp\left(\frac{i\delta t}{2} \sum_{\substack{(i,j) \\ \text{even row}}} Y_i Y_j P_a\right) \exp\left(\frac{i\delta t}{2} \sum_{\substack{(i,j) \\ \text{odd row}}} X_i X_j P_a\right), \quad (9)$$

as well as $U_{|,1}$ identically but with columns instead of rows, and $U_{-,2}$ identically but swapping $X_i X_j$ and $Y_i Y_j$. For definiteness, we will fix the Trotter step to $\delta t = 0.2$.

Finally, we measure the lattice sites in the Z basis. We form the operator

$$\mathcal{O} = \sum_{j=1}^L f_j Z_j, \quad (10)$$

for a given function f_j . We fix the following function:

$$f_j = \begin{cases} -1, & \text{if } j_y < L_y/2, \\ 1, & \text{if } j_y \geq L_y/2. \end{cases} \quad (11)$$

With this definition, $\mathcal{O}$ measures the imbalance of fermions between the lower and upper parts of the lattice.

The exact outcome of the quantum circuit obtained after n applications of the Trotter operator U can be computed (see Appendix B). The result is expressed as

$$\begin{aligned} \langle \mathcal{O}(n) \rangle_{\text{exact}} = & \sum_{\varphi_x=0,1/2} \sum_{\varphi_y=0,1/2} \left(\sum_{k,q \in K_{\varphi_x,\varphi_y}} \hat{f}(k-q) (\alpha_n^*(k) \alpha_n(q) \right. \\ & - \beta_n^*(-k) \beta_n(-q)) \hat{n}(k-q) \\ & \left. + \hat{f}(0) \sum_{k \in K_{\varphi_x,\varphi_y}} |\beta_n(k)|^2 \right), \end{aligned} \quad (12)$$

where K_{φ_x,φ_y} denotes the set of pairs

$$\begin{aligned} K_{\varphi_x,\varphi_y} = & \left\{ \left(\frac{2\pi(k_x + \varphi_x)}{L_x}, \frac{2\pi(k_y + \varphi_y)}{L_y} \right), \right. \\ & \left. k_{x,y} = 0, \dots, L_{x,y} - 1 \right\}. \end{aligned} \quad (13)$$

The coefficients $\alpha_n(k), \beta_n(k)$ are given by

$$\begin{aligned} \alpha_n(k) = & e^{-i n \epsilon_k} + i(-\sin(2\delta t)(\cos k_x + \cos k_y) + 2 \sin(2\delta t) \sin^2(\delta t) \cos k_x \cos k_y (\cos k_x + \cos k_y) + \sin \epsilon_k) \frac{\sin(n \epsilon_k)}{\sin \epsilon_k}, \\ \beta_n(k) = & (i \sin^2(\delta t)(\sin(2k_x) + \sin(2k_y)) - 2i \sin^4(\delta t)(\cos^2(k_x) \sin(2k_y) + \cos^2(k_y) \sin(2k_x)) \\ & + \sin^2(\delta t) \sin(2\delta t)(\cos(k_x) \sin(2k_y) + \cos(k_y) \sin(2k_x))) \frac{\sin(n \epsilon_k)}{\sin \epsilon_k}, \end{aligned} \quad (14)$$

with

$$\begin{aligned} \epsilon_k = & \text{sgn}(\cos k_x + \cos k_y) \arccos[1 - 2 \sin^2(\delta t) \\ & \times (\cos k_x + \cos k_y)^2 + 4 \sin^4(\delta t) \cos k_x \cos k_y \\ & \times (1 + \cos(k_x + k_y))]. \end{aligned} \quad (15)$$

This function $\langle \mathcal{O}(n) \rangle$ follows a nontrivial trajectory, while still being computable in a time that is polynomial in the system size. For example, we depict in the right panel of

Fig. 2 this observable as a function of the number of Trotter steps, for systems of different sizes up to $32 \times 32 = 1024$ way beyond the regime that is accessible to general-purpose classical methods. For better comparison of different system sizes, since the timescale is proportional to the linear system size L_x in the protocol that we defined, the observable is plotted as a function of number of Trotter steps divided by L_x in the right panel of Fig. 2. In the left panel, we show the observable obtained for a larger number of Trotter steps in a simulable system size $L = 4 \times 4$. We compare this exact value

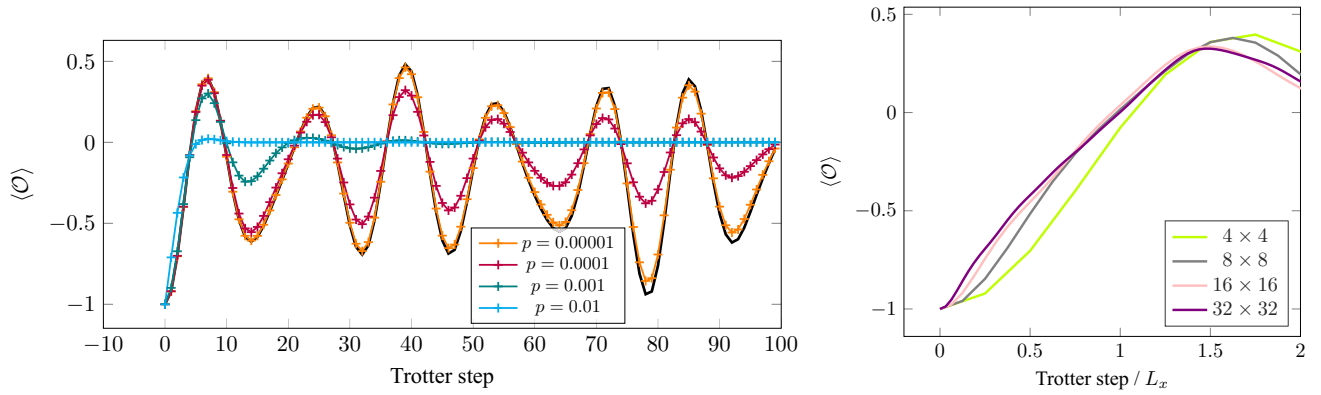


FIG. 2. Left: Curve $\langle \mathcal{O} \rangle$ as a function of Trotter step number, on a system of size $L = 4 \times 4$, for different depolarizing noise levels p per two-qubit gate. The curves are averaged over 20 different realizations of noise where single-qubit gate rotations with a small random angle are inserted after every two-qubit gate, as described in Ref. [36]. The black continuous curve is the exact value. Right: Exact noiseless curve $\langle \mathcal{O} \rangle$ as a function of number of Trotter steps divided by L_x , for different system sizes L .

to numerical simulations of noisy circuits for different error rates per two-qubit gate p , displaying a generic attenuation of the signal with circuit depth.

C. The score

We now would like to assign a score to a given output of a hardware to benchmark. Unlike classical computers, quantum computers can only output “shots” over which one has to average in order to obtain an expectation value of an operator $\langle \mathcal{O}(n) \rangle$. The precision achieved on the quantum computer is thus directly related to the time spent on the computation. If because of hardware imperfections the quantum computer has a bias in the expectation value $\langle \mathcal{O}(n) \rangle$, this bias will not be *detectable* if after averaging over a finite number of shots the error bars are larger than the bias. Hence, the noisier a hardware, the faster the imperfections can be detected as it will require averaging over fewer shots. Conversely, a given hardware with low noise will be statistically indistinguishable from noiseless, before a certain amount of resources are spent to reach the precision where the bias due to imperfections becomes visible. This suggests a physical and intuitive way of measuring the accuracy of a given quantum hardware, by answering the following question: How many gates does a perfect quantum computer have to implement (or similarly, how much time does it need), running the same circuit as the benchmarked hardware, to certify that the output of the benchmarked hardware is incorrect? We will call this quantity *distinguishability cost*.

In our case, we fix the following computational task: computing the expectation values $\langle \mathcal{O}(n) \rangle$ after $n = 1, \dots, T$ Trotter steps, with final time fixed to $T = 2L_x$, in a square lattice $L_x = L_y$. Let us denote by m_n the estimates obtained for these expectation values on a benchmarked hardware, and consider that we run the same circuit on a perfect hardware, obtaining estimates s_n with standard deviations σ_n . For the moment, we will not take into account the error bars on the estimates m_n obtained from the benchmarked hardware. The output of the benchmarked hardware can be certified to be incorrect if the outcomes $m_1, \dots, m_T$ are statistically incompatible with the unbiased estimates s_n with standard

deviations σ_n . This statistical compatibility can be inferred from a chi-square test with T degrees of freedom. We will say that the output of the benchmarked hardware is certified to be incorrect if it fails the chi-square test by 3σ , namely, if

$$\chi_2^{(T)} \left(\sum_{n=1}^T \frac{(s_n - m_n)^2}{\sigma_n^2} \right) > 0.997, \quad (16)$$

where $\chi_2^{(T)}$ denotes a chi-square cumulative distribution function with T degrees of freedom.

Given outputs m_n , it is a nontrivial problem to find the best strategy to follow on the (gedanken) perfect hardware to certify that these outputs are incorrect as quickly as possible. One would ideally run on the perfect hardware only the noisiest time point, but that time point cannot be known in advance without running other time points on the perfect hardware. While we could implement numerically an efficient strategy for this, we prefer instead to compute the *minimal* resources required to certify incorrectness of the outputs, even in the case where the user would know which points are the noisiest. This definition has the advantage of being simpler, more canonical, and not sensitive to precise details of the implementation of the strategy followed.

Let us now determine this minimal cost. The cost of running a shot for time point n is proportional to n , because the number of gates is proportional to n (neglecting for simplicity the gates appearing in the state preparation, before applying the first Trotter step). The variance σ_n^2 associated with the measurement of $\mathcal{O}$ after n Trotter steps is

$$\sigma_n^2 = \langle \mathcal{O}(n)^2 \rangle - \langle \mathcal{O}(n) \rangle^2. \quad (17)$$

With S_n shots, the estimated $\langle \mathcal{O}(n) \rangle_{\text{est}}$ on the perfect hardware for the quantity $\langle \mathcal{O}(n) \rangle$ is thus

$$\langle \mathcal{O}(n) \rangle_{\text{est}} = \langle \mathcal{O}(n) \rangle + \frac{\sigma_n}{\sqrt{S_n}} \xi, \quad (18)$$

where ξ is a random Gaussian variable with mean 0 and variance 1. This variance per shot σ_n involves correlations of the type $Z_i Z_j$, which can also be computed exactly in polynomial time in this free fermion benchmark. We refer the reader to Appendix B 4 for a detailed calculation method.

Hence, denoting t_n the exact expectation value of $\mathcal{O}$ after n Trotter steps, the cheapest way of certifying incorrectness of the outputs is to only run the time point n_* that maximizes $(t_n - m_n)^2 / (n\sigma_n^2)$. In that case, the number of shots to run is S_{n_*} that satisfies

$$\chi_2^{(T)} \left(\frac{(t_{n_*} - m_{n_*})^2}{\sigma_{n_*}^2} S_{n_*} \right) = 0.997. \quad (19)$$

The total number of gates is then equal to $12LS_{n_*}n_*$, namely, $12Ln_*$ two-qubit gates per circuit, repeated S_{n_*} times (we again did not take into account the gates used in the state preparation). Any strategy has to run at least that many gates, and even more so if one does not have access to the exact value t_n beforehand. This will thus be the definition of our score

$$S(\{m_n\}) = 12LS_{n_*}n_*, \quad (20)$$

where S_{n_*} is the unique solution to Eq. (19), and where n_* maximizes $(t_n - m_n)^2 / n$. The interpretation of S is the smallest number of two-qubit gates that a perfect quantum computer would have to implement, running the same circuit as the benchmarked hardware, to certify that the output of the benchmarked hardware is incorrect. This number of two-qubit gates does not refer to the number of gates per circuit, but to the total number of gates run across different circuits and shots. We note that by “two-qubit gate” we mean *logical* two-qubit gate, namely, the operation that acts on the qubits that host the quantum information (whether encoded with quantum error correction or not—in the latter case, it is the physical two-qubit gate). The gate count also should *not* take into account auxiliary gates such as SWAP gates in case the hardware does not support the implementation of a two-qubit gate between arbitrary qubits. The definition of the score (20) is thus imposed to be the same for any platform, architecture, or compilation scheme.

We now would like to take into account the effect of error bars on estimates m_n obtained on the benchmarked hardware. Namely, we consider that the benchmarked hardware outputs a mean value m_n with standard deviation τ_n because of shot noise. This shot noise τ_n is *not* related to the shot noise on the ideal perfect hardware σ_n that is used to compute the score. This is an imprecision that comes from the fact that a finite number of shots has been run on the hardware to be tested. Since this imprecision τ_n always deteriorates the accuracy and the confidence that we can have on the hardware output m_n , a finite τ_n should come with two requirements: (1) It should never improve the score compared to $\tau_n = 0$, and (2) it should fix a maximal possible score, independently of m_n , because any hardware defect with impact lower than τ_n cannot be detected. A simple way to take these two requirements into account is to apply the previous framework where we neglected shot noise, but with the distance between exact and measured $|t_n - m_n|$ replaced by

$$|t_n - m_n| \longrightarrow \max(|t_n - m_n|, \tau_n), \quad (21)$$

the maximal value between the distance to exact and the standard deviation. This minimal change is basically implementing the idea that with an error bar $\tau_n > 0$, we should not be able to get a precision better than τ_n on the exact value. Hence, we modify the protocol as follows. We define n_* as

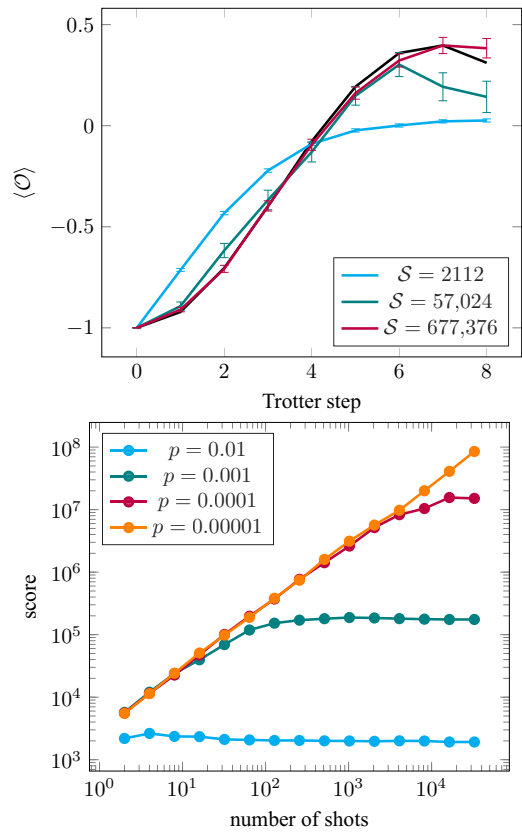


FIG. 3. Top: Value of the score (in the legend) obtained in size $L = 4 \times 4$ for different noise levels p and for different number of shots N_S (cyan: $p = 0.01$ and $N_S = 10^3$; teal: $p = 0.001$ and $N_S = 50$; purple: $p = 0.0001$ and $N_S = 200$). The score indicates the minimal number of gates that a perfect quantum computer would have to run across different circuits to certify that the output curve is incorrect. Higher score means better hardware. Bottom: Score obtained in size $L = 4 \times 4$ as a function of number of shots per time point, for different noise levels.

the time point n that maximizes

$$\frac{\max((t_n - m_n)^2, \tau_n^2)}{n\sigma_n^2}. \quad (22)$$

Then, the number of shots to run is S_{n_*} that satisfies

$$\chi_2^{(T)} \left(\frac{\max((t_n - m_n)^2, \tau_n^2)}{\sigma_n^2} S_{n_*} \right) = 0.997, \quad (23)$$

and the score is set again to

$$S(\{m_n, \tau_n\}) = 12LS_{n_*}n_*. \quad (24)$$

In the top panel of Fig. 3, we present different curves obtained in size $L = 4 \times 4$ using different depolarizing noise levels and number of shots, and compute their score. In the bottom panel of Fig. 3, we show the score obtained as a function of the number of shots per time point, for different noise levels. The general behavior of the curves is to first be proportional to the number of shots (which is expected when the number of shots is the limiting factor of the precision), and then saturate at some finite value (when the limiting factor

is hardware noise). We also observe that the score is almost always an increasing function of the number of shots.

D. Summary

The benchmark protocol evaluates the ability of the hardware to simulate the dynamics of electrons in conducting materials. The system considered is a nearest-neighbor hopping Hamiltonian on a square lattice, similar to a Hubbard model but without electron-electron interaction. Hamiltonian simulation is implemented with a Trotter decomposition. The scalability of the benchmark relies on the free fermion structure of the Hamiltonian considered, which allows for computing the exact mean position of the electrons in a time that is polynomial in the system size. The score is defined as the “distinguishability cost,” which is the minimal number of gates that an ideal perfect hardware would need to implement to certify that the output of the tested hardware is incorrect.

E. Comparison with criteria

Let us now justify the definition of the benchmark protocols, in the light of the criteria defined in Sec. II B. As said above, the objective of this benchmark protocol was to evaluate the ability of a quantum computer to simulate the dynamics of quantum particles, such as electrons in a conducting material. We focused the benchmark on the Hubbard model, as it is the most studied electronic model. Researchers and engineers would likely implement this system on quantum computers once they prove more powerful than classical computers for this task.

Before addressing the question of the scalability of a benchmark, let us make the following simple comment. Contrary to other applications like classical optimization problems where one always seeks to minimize or maximize the output of the algorithm, benchmarking the simulation of physical systems requires to compare the output of the hardware with an exact known value: A measured particle density x_0 can be interpreted and evaluated only in the light of the knowledge of the exact value x . This poses thus the obvious problem of the scalability of the benchmark, as defined in Sec. II B. Since the classical simulation of the system is exponential in either the number of qubits or the simulation time, a direct comparison of the measured observable with the exact observable of the Hubbard model is not scalable. There exist methods such as mirror circuit benchmarks that allow to estimate the amount of noise in any circuit, simulable or not, that we could apply to the simulation of the Hubbard model as well. However, it is known that in Hamiltonian simulation, *local* observables like particle density are much less sensitive to noise than global observables like a return probability [8,9]. A mirror circuit benchmark would thus vastly overestimate the noise in local observables, making the benchmark less application-oriented and more theoretical.

The free fermionic system that we use for the benchmark solves this problem. Simulating the Hubbard model is equivalent to simulating two species of noninteracting fermions (up spin and down spin), with additional density-density interactions. These density-density interactions are simple ZZ rotations that are much less costly to implement than the

fermion hopping, which require a fermionic encoding and more gates. For example, in our implementation on two copies of noninteracting fermions on L sites, the density-density interaction costs L two-qubit gates, while a Trotter step of the kinetic terms would cost much more, in total $24L$ two-qubit gates. Hence, in terms of gate count, circuit structure, observables measured, and lattice geometry, simulating free fermions on a quantum computer is very similar to simulating the Hubbard model. The same local observables that one would measure in the Hubbard model can be measured on hardware and directly compared to the exact value for this noninteracting fermion benchmark, which can be computed in polynomial time in system size and simulation time due to the absence of interactions. The free fermion benchmark that we have defined is thus both scalable and application-oriented.

This benchmark also satisfies the other criteria of Sec. II B. It is relatively cheap, in the sense that it is not more costly than the Hubbard model simulation that the end user would actually implement. The number of shots is a free parameter that the user can choose optimally to reduce costs: namely, they can choose it so that shot noise is just slightly below hardware noise, without degrading the score. This is a direct property of the distinguishability cost that we introduced. To make the benchmark difficult to spoof, we imposed to simulate several different time points. The score that we defined imposes the hardware to perform well for all the time points, preventing to get good scores from just accidental agreement on one time point. Moreover, the specific initial state chosen evolves on a timescale of order L , and remains of order 1 for all these time points, avoiding constant signal or very small signal, which could have led to easier spoofing as well. Finally, we chose a two-dimensional (2D) square lattice geometry both to stay as close as possible to relevant use cases (since the Hubbard model has been mostly studied on 2D square lattices) and also to keep the benchmark easy to implement on any architectures, whether it is all-to-all coupled or gridlike architectures. The benchmark is thus also hardware-agnostic.

This free fermion benchmark thus satisfies all the criteria that we defined in Sec. II B.

IV. APPLICATION: STATIC OBSERVABLES AT LOW TEMPERATURE

A. Context and motivation

Materials often display exotic properties as their temperature is lowered, with phases requiring quantum physics in order to be described accurately, such as superconducting phases or Fermi liquids. The computation of static, equilibrium expectation values at low temperature in these many-body physics Hamiltonians can become difficult or unreliable to perform with classical computers for intermediate-size to large systems.

On a quantum computer, the adiabatic algorithm is a generic way of preparing the ground state of a Hamiltonian. It can be formulated as follows. Given an initial Hamiltonian H_I whose ground state can be prepared efficiently on a quantum computer and a final Hamiltonian H_F whose ground state is the target state, we define the time-dependent Hamiltonian

$$H(s) = \varphi(1-s)H_I + \varphi(s)H_F, \quad (25)$$

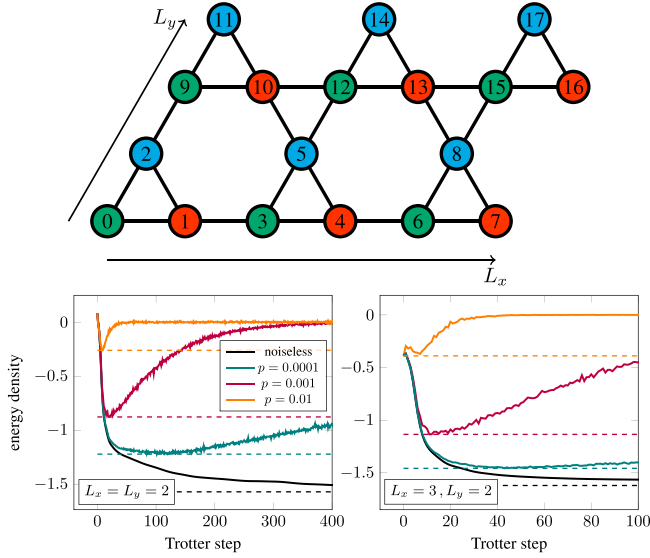


FIG. 4. Top: Kagome lattice with $L_x = 3$ and $L_y = 2$. Bottom: Energy density as a function of number of Trotter steps in the benchmark setup, for different system sizes and different noise levels. Dashed lines indicate the minimum reached by the curve with the same color, and correspond to the benchmark score.

with $\varphi(s)$ for $0 \leq s \leq 1$ a scheduling function that is continuous and satisfies $\varphi(0) = 0$, $\varphi(1) = 1$. For a given parameter $T > 0$ called adiabatic time, we then define the state $|\psi_T(t)\rangle$ by the fact that $|\psi_T(t=0)\rangle$ is the ground state of H_I , and is evolved under the time-dependent Schrödinger equation

$$i\partial_t |\psi_T(t)\rangle = H(t/T) |\psi_T(t)\rangle, \quad (26)$$

for times $0 \leq t \leq T$. The adiabatic theorem of quantum mechanics says that if $H(s)$ is gapped for all $0 \leq s \leq 1$, then $|\psi_T(t=T)\rangle$ gets closer to the ground state of H_F as T grows larger, and becomes the ground state of H_F when $T \rightarrow \infty$. All the scalability aspect of the adiabatic algorithm depends on how large T has to be to reach a certain precision on the ground-state energy.

B. The benchmark

As a benchmark, we consider the Heisenberg antiferromagnet model on a kagome lattice. This model describes the material $\text{YCu}_3[\text{OH}(\text{D})]_{6.5}\text{Br}_{2.5}$ [37] and the precise properties of its ground state are still debated [38,39]. The Hamiltonian of this system is given by

$$H = \sum_{\langle i,j \rangle} X_i X_j + Y_i Y_j + Z_i Z_j, \quad (27)$$

where $\langle i, j \rangle$ means that sites i, j are neighbors on the kagome lattice. We parametrize this lattice by two integers L_x, L_y that count the number of small disjoint triangles in the vertical and horizontal directions, with $N = 3L_x L_y$ sites in total, and impose open boundary conditions. The sites are enumerated within triangles first, then along the x direction, and then along the y direction. An example of this kagome lattice with site numbering and bonds between sites is represented in Fig. 4. To define an adiabatic path to prepare the ground state of this

model, we set the initial Hamiltonian to

$$H_I = \sum_{j=1}^N (-1)^j X_j. \quad (28)$$

The ground state of such Hamiltonian is simply given by a Neel state in the X basis, $|+-+ - \dots\rangle$. We then fix the Trotter step dt as a function of s , the scheduling time, as

$$dt(s) = 0.2\sqrt{1-s}. \quad (29)$$

This time-dependent Trotter step size is justified as follows. Along the path, the Trotter step does not have to be taken smaller and smaller to implement the adiabatic evolution, because the Trotterization simply implements an adiabatic evolution with respect to the Floquet Hamiltonian, namely, the effective Hamiltonian equivalent to implementing one Trotter step, including Trotter error. However, at the end of the path, this Trotter step size must be sent to zero in order to prepare the ground state of H , and not a Floquet version of it. This is why we impose $dt(s) \rightarrow 0$ when $s \rightarrow 1$. The precise functional form of $dt(s)$ has been then chosen from numerical simulations, ensuring a relatively fast decay of energy at the beginning, while still converging to the ground state in the limit of large number of steps, in order to make the benchmark implementable on noisy hardware where a relatively small number of gates can be implemented. This numerics has been done in size $L_x = L_y = 2$ and $L_x = 3, L_y = 2$. As for the scheduling function $\varphi(s)$ entering Eq. (25), we choose the following form:

$$\varphi(s) = \frac{1 + \tanh(\tan(s\pi - \pi/2))}{2}, \quad (30)$$

which is simply chosen so as to smoothly interpolate between $\varphi(0) = 0$ and $\varphi(1) = 1$ while having all derivatives vanishing at $s = 0$ and $s = 1$. Finally, the ordering of the terms in the Trotter decomposition is taken to be first applying the single-qubit rotations corresponding to H_I in Eq. (28), then the XX terms corresponding to H in Eq. (27), then all the YY terms, and then all the ZZ terms.

We note that there exist initial states that are simple to prepare and that have much lower energy with respect to H than this Neel state in the X basis. For example, one can consider a perfect matching of the kagome graph, i.e., a subset of edges such that all vertices touch exactly once an edge in the subset, and initialize a singlet state on all these edges. This state has much lower energy with respect to H than the Neel state we start from, and is the ground state of the Heisenberg model defined only on the edges included in the perfect matching. We could thus do an adiabatic evolution between these two Hamiltonians. However, we observed that the convergence to the ground state of H with this adiabatic path is still very slow, despite the initial energy being much lower. Because of the noise introduced by gates, it would make it difficult for any current hardware to actually lower the energy compared to this perfect matching initial state. For the sake of defining a practical benchmark, we thus preferred to start from a much higher-energy state, giving more amplitude to a noisy hardware to reduce the energy and thus yielding more meaningful benchmarking scores.

C. The score

In this protocol, the only degree of freedom remaining is M , the number of Trotter steps performed. Only in the limit $M \rightarrow \infty$ is the exact adiabatic evolution implemented and the energy of the Hamiltonian $H_F = H$ minimized. On actual hardware, however, noise precludes running arbitrarily deep circuits and effectively heats up the system, which competes with the cooling of the adiabatic process. At small M , heating due to imperfect adiabatic evolution dominates, and at large M , heating due to hardware noise dominates. There is thus a nontrivial optimal number of Trotter steps M_* at which the energy is minimized. Given a mean energy E_M obtained with M Trotter steps, and with δE_M the standard deviation, we take $E_M + 2\delta E_M$ as the result energy, in order to avoid overshooting due to shot noise. We then define the benchmark score of a benchmarked hardware as

$$\mathcal{S}_{\text{KH}} = \min_{M \geq 1} (E_M + 2\delta E_M). \quad (31)$$

We plot in the bottom panel of Fig. 4 the energy density obtained as a function of the number of Trotter steps, for different noise levels, together with the exact ground state. We see that for nonzero noise level p , the energy typically displays the expected behavior, with an initial decrease and then an increase at large number of Trotter steps. The exact ground-state energy can be obtained up to around $N \sim 30$ with classical computers, depending on the resources allocated. The comparison with the exact result is thus not scalable. However, even for system sizes beyond the classically simulable regime, the performance of the same algorithm run on different hardware can be compared, by directly comparing the energy density attained, the smaller being the best.

D. Summary

The benchmark protocol evaluates the ability of the hardware to prepare low-energy state of materials with the adiabatic algorithm. The system considered is the Heisenberg model on a kagome lattice, which combines standard and relatively simple models and lattices, while being difficult to tackle with classical computers. Hamiltonian simulation is implemented with a Trotter decomposition. The score is directly the minimal energy obtained. The benchmark is scalable as the score does not require the knowledge of the exact ground-state energy.

E. Comparison with criteria

We now review the criteria defined in Sec. IIB for this benchmark. The benchmark is application-oriented as it directly evaluates the ability of the hardware to prepare the low-energy state of a system that is difficult to obtain classically. We chose the particular case of the Heisenberg model on a kagome lattice as the properties of its ground state are still debated and spark significant academic interest, guaranteeing the usefulness of implementing this model on a quantum computer. Moreover, the benchmark is hardware-agnostic as it does not require all-to-all connectivity. Since the score is defined as the lowest energy obtained, the benchmark can be run for any system size without requiring the knowledge of the exact value. Different hardware can always be compared

to each other and the benchmark is scalable. It is also difficult to spoof, since it is unlikely that hardware defects would lead to preparing the ground state of this complicated system by chance. Finally, the benchmark is also relatively cheap to implement, since the end user is free to choose how many Trotter steps and how many shots they perform. They will thus naturally choose a number of Trotter steps that will give the best score, avoiding significant resource waste.

V. APPLICATION: NUCLEAR MAGNETIC RESONANCE

A. Context and motivation

NMR experiments are a key tool for material and molecular structure elucidation. They consist in polarizing all the nuclear spins of a sample material in a specific direction with a high magnetic field, and then measuring the relaxation of the magnetic field generated by the nuclear spins. The NMR spectrum of the sample material obtained by Fourier transforming the signal measured is then a signature of the bonds between the atoms supporting the nuclear spins. The classical simulation of NMR experiments can be done efficiently with dedicated softwares at high external magnetic field [40]. However, the simulation is more difficult in case of low external magnetic field, which is cheaper to implement experimentally. This low-field simulation of NMR experiments is one of the promising near-term applications of quantum computers [41–44], although the precise settings where quantum computers would bring a practical advantage are still debated. The purpose of this present work is not to enter this debate, but instead to define a benchmark setup based on the performance of a quantum computer to infer couplings between nuclear spins in a molecule through NMR simulation.

These NMR experiments at low field are modeled as follows [45]. The signal measured in an NMR experiment, called free induction decay (FID), can be written as

$$\text{FID}(t) = \text{tr} [\Pi^\dagger S_z(t) \Pi S_z(0)]. \quad (32)$$

Here, the total magnetization S_z of the molecule is

$$S_z = \sum_{j=1}^N \gamma_j Z_j, \quad (33)$$

where each qubit corresponds to each of the N nuclear spins $1/2$ contained in the molecule, and with γ_j the gyromagnetic factor of nuclear spin j . The unitary operator Π represents the initial pulse

$$\Pi = e^{i\tau \sum_{j=1}^N \gamma_j X_j}, \quad (34)$$

with τ the pulse duration. The time-evolved spin $S_z(t)$ is given by

$$S_z(t) = e^{iHt} S_z e^{-iHt}, \quad (35)$$

with H the Hamiltonian describing the interactions between the N spins. In the absence of external magnetic field and for spin $1/2$ nuclei, this Hamiltonian can be written as

$$H = \frac{1}{4} \sum_{i < j} J_{ij} (X_i X_j + Y_i Y_j + Z_i Z_j), \quad (36)$$

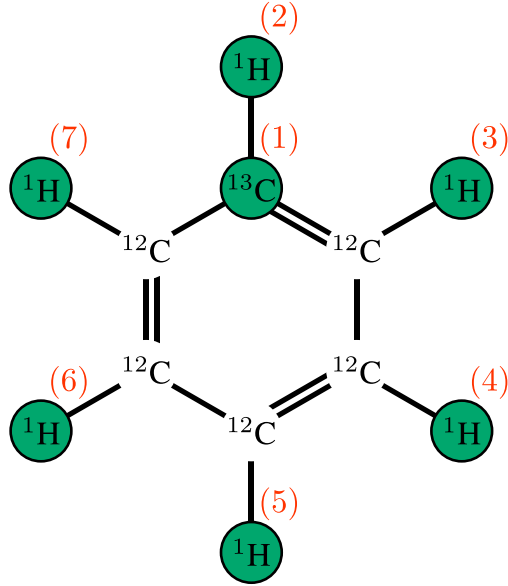


FIG. 5. Depiction of the benzene molecule. The spinful atoms are indicated with a green circle. The numbering of the different spinful nuclei is given in red.

with J_{ij} the so-called J coupling between nuclear spins i and j , which is an effective spin-spin interaction resulting from the electron bondings in the molecule.

From the measurement of the FID, one then computes the spectrum

$$S(\omega) = \int_0^\infty e^{i\omega t} \text{FID}(t) dt. \quad (37)$$

This amplitude $S(\omega)$ is the signal that the NMR end user is interested in. In an actual NMR experiment, the FID that is measured is the sum of all the tiny magnetic fields generated by the nuclei of all the molecules in the sample. Because of small perturbations, these slowly desynchronize with time, which results in an exponential decay in the FID. For liquid NMR, this exponential decay is very often modeled by an apodization term e^{-t/T_2} multiplying $\text{FID}(t)$, with T_2 a certain relaxation time.

B. The benchmark

The benchmark we propose is the benzene- $^{13}\text{C}_1$ molecule depicted in Fig. 5. It contains seven nuclear spins, six hosted by the hydrogen atoms and one by the carbon-13 atom. The gyromagnetic factors are $\gamma_1 = 67.2828$ for the ^{13}C nucleus and $\gamma_j = 267.522$ for the ^1H nuclei, $j = 2, \dots, 7$. The J couplings obtained from experiments are listed in Table II. The pulse time is taken to be $\tau = \frac{\pi}{2\gamma_1}$. The maximal simulation time is taken to be $T = 50$, and we fix an arbitrary but realistic relaxation time $T_2 = 10$. Given the exact $\text{FID}(t)$, we define the spectrum to which the hardware is to be compared as

$$S_{\text{exact}}(\omega) = \int_0^T e^{i\omega t} e^{-t/T_2} \text{FID}(t) dt. \quad (38)$$

Since the model is defined on only seven qubits, this quantity can be quickly computed classically with arbitrary precision.

TABLE II. Coefficients J_{ij} of the benzene- $^{13}\text{C}_1$ molecule, from Ref. [46].

i/j	2	3	4	5	6	7
1	158.354	1.133	7.607	-1.296	7.607	1.133
2		7.540	1.380	0.661	1.380	7.540
3			7.543	1.377	0.658	1.373
4				7.535	1.382	0.658
5					7.535	1.377
6						7.543

We impose that the time evolution is implemented using a Trotter evolution, with Trotter step

$$U = \prod_{i < j} e^{i \frac{\delta t}{4} X_i X_j} e^{i \frac{\delta t}{4} Y_i Y_j} e^{i \frac{\delta t}{4} Z_i Z_j}, \quad (39)$$

where δt is a given Trotter step size. We fix the ordering of the couplings to be given by applying the gates in the following order: (1, 2), (3, 4), (5, 6), (1, 7), (2, 3), (4, 5), (6, 7), (1, 3), (4, 6), (2, 7), (3, 5), (1, 6), (2, 4), (5, 7), (1, 4), (1, 5), (2, 5), (2, 6), (3, 6), (3, 7), and (4, 7). The benchmark user is free to choose the Trotter step size δt , but is fixed to be the same for all time points.

C. The score

1. Overview

We propose to evaluate the outcomes $\text{FID}(n\delta t)$ of the quantum computer in a most application-oriented way. NMR experiments are performed to elucidate the structure of a given molecule. In our simple use case of the benzene molecule, this would mean computing the J couplings J_{ij} between every spinful nuclei. Given an NMR spectrum obtained from experiment, we would perform simulation with some trial couplings $\tilde{J}_{ij}$, and then take as an estimate of the actual couplings J_{ij}^{est} the trial couplings corresponding to the spectrum that matches the experiment the most closely. A natural score is then the mean error between estimated coefficients J_{ij}^{est} and actual coefficients J_{ij} .

The computation of the score assigned to values $\text{FID}(n\delta t)$ measured on the hardware for $n = 0, \dots, T/\delta t$ is done in multiple stages.

2. Compatibility measure

First, given a candidate spectrum $S_{\text{can}}(\omega)$, we would like to evaluate the compatibility with our measured time series $\text{FID}(n\delta t)$ from the hardware. We call the output of that stage “compatibility measure.” From the time series, we compute the spectrum as

$$S_{\text{hard}}(\omega) = \delta t \sum_{n=0}^{T/\delta t} e^{i\omega n\delta t} e^{-n\delta t/\tilde{T}_2} \delta_n \text{FID}(n\delta t), \quad (40)$$

where $\tilde{T}_2$ is a parameter, and with $\delta_n = 1/2$ if $n = 0$ or $n = T/\delta t$, and $\delta_n = 1$ otherwise. This δ_n term removes potential baseline offset [45]. The spectrum $S_{\text{hard}}(\omega)$ is computed at the values ω where $S_{\text{can}}(\omega)$ is available. The benchmark user is free to choose $\tilde{T}_2$ (even to take it negative) to optimize the

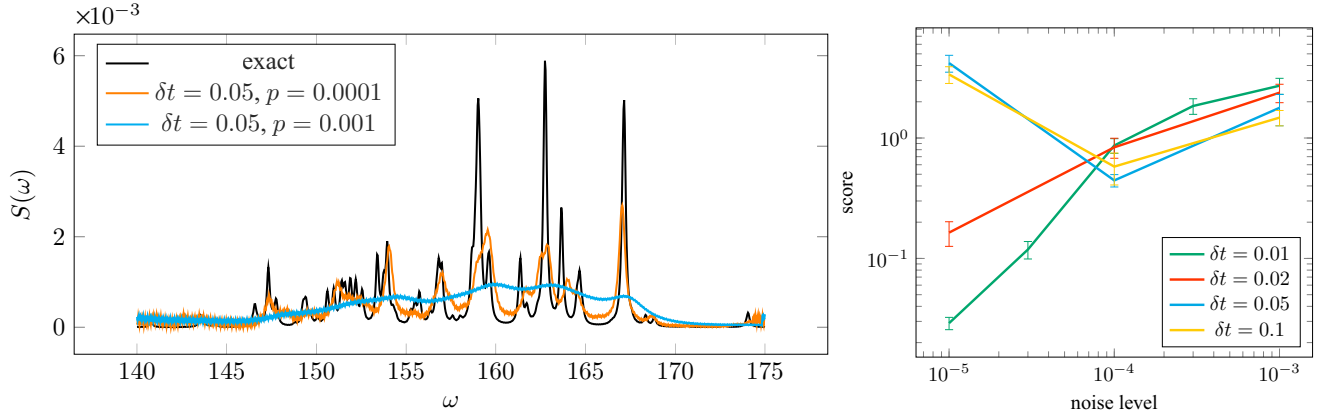


FIG. 6. Left: Exact spectrum $S(\omega)$ obtained with $\delta t = 0.01$ and $T_2 = 10$ (black), and noisy spectra obtained with $\delta t = 0.05$ and noise levels $p = 0.0001$, $p = 0.001$ (cyan and orange) after optimizing $\tilde{T}_2$ and $\delta\omega$ as described in Sec. VC2. Right: Root mean square ΔJ on the estimated J couplings, as a function of the noise level, for different Trotter steps.

agreement with $S_{\text{exact}}(\omega)$. Hardware results are indeed going to come with noise that will already induce an exponential decay on the data: When comparing with an actual NMR experiment, such an additional exponential decay $\tilde{T}_2$ can always be incorporated to match the exponential decay observed in the NMR experiment. The benchmark user is also free to set $\text{FID}(n\delta t) = 0$ for time points that they decide not to compute. Moreover, we also allow the user to apply a shift in the frequencies, namely, to redefine

$$S'_{\text{hard}}(\omega) = S_{\text{hard}}(\omega + \delta\omega), \quad (41)$$

with an arbitrary parameter $\delta\omega$ so as to optimize agreement with $S_{\text{can}}(\omega)$. We indeed observed that Trotter errors coming from the finite Trotter step size tend to globally slightly shift the frequencies. While impacting significantly point-by-point agreement between $S_{\text{can}}(\omega)$ and $S_{\text{hard}}(\omega)$, this effect does not prevent identification of the spectrum, and so we decide to mitigate it with the above freedom to shift the frequencies. For ease of implementation and to avoid having to introduce an arbitrary scale, we impose that the shift in Eq. (41) is applied periodically on the range of ω 's. The agreement between $S'_{\text{hard}}(\omega)$ and $S_{\text{can}}(\omega)$ is evaluated by maximizing the inner product $F(S'_{\text{hard}}, S_{\text{can}})$ with

$$F(A, B) = \frac{\sum_{\omega} A(\omega)B(\omega)}{\sqrt{\sum_{\omega} A(\omega)^2 \sum_{\omega} B(\omega)^2}}. \quad (42)$$

Namely, the final compatibility measure between the measured time series and the candidate spectrum is the maximal value of $F(S'_{\text{hard}}, S_{\text{can}})$ obtained when optimizing $\tilde{T}_2$ and $\delta\omega$. In the left panel of Fig. 6, we present simulated spectra $S'_{\text{hard}}(\omega)$ obtained after this optimization, for different noise levels, and comparison to the exact spectrum computed with $\delta t = 0.01$.

3. Identification within a database

We now would like to use the compatibility measure defined in the previous subsection to identify, among a database of molecular spectra, the spectrum that is the most compatible with our measured time series. We define these databases of molecular spectra as being composed of 100 spectra of simulated benzene molecules, but with different J couplings.

One of the 100 spectra is computed with the exact J couplings given in Table II. The 99 other spectra are computed with perturbed J couplings $\tilde{J}_{ij}$ randomly generated as follows:

$$\tilde{J}_{ij} = J_{ij} + 0.01 \times 2^{0.1m}\xi, \quad (43)$$

for $m = 0, \dots, 98$, and with ξ a random Gaussian variable with mean 0 and variance 1 (randomly drawn for every couple (i, j) and sample in the database). This way, the database contains spectra that are very close to the exact spectrum of benzene, for the lowest values of m . However, it also contains spectra that are very different from it, for the largest values of m . For every sample in the database, one computes the exact $\text{FID}_{\text{can}}(n\delta t)$ for $n = 0, \dots, T/\delta t$ with noiseless numerical simulation, with $\delta t = 0.01$ and $T = 50$. Then, one computes the spectrum associated with this m th sample:

$$S_{\text{can}}^{(m)}(\omega) = \delta t \sum_{n=0}^{T/\delta t} e^{i\omega n\delta t} e^{-n\delta t/T_2} \delta_n \text{FID}_{\text{can}}(n\delta t), \quad (44)$$

with $T_2 = 10$. We will be only interested in the frequency region $140 \leq \omega \leq 175$, which contains most of the interesting features of this molecule. For definiteness, we will compute the spectrum at 1000 equally spaced values of ω between 140 and 175. The compatibility measure defined in the previous subsection will thus depend only on the frequencies within this range. Once the whole database is generated, we look for the sample that has the highest compatibility measure with our time series measured on the hardware. The estimated J couplings J_{ij}^{est} are then set to be the J couplings of this most compatible sample within the database.

4. The score

Given estimated J couplings J_{ij}^{est} , we define the quality of the estimate as the mean square error

$$\Delta J = \sqrt{\frac{1}{21} \sum_{i < j} (J_{ij} - J_{ij}^{\text{est}})^2}, \quad (45)$$

where 21 is the number of J couplings in our particular case of benzene. Given a time series measured on the hardware,

this ΔJ is a random variable, since it depends on the random database generated for comparing the spectra. To define a score that is not a random variable, we then define the average

$$S_{\text{NMR}} = \mathbb{E}[\Delta J], \quad (46)$$

where the statistical average $\mathbb{E}$ is over different random databases. If only a small number of databases are generated, one can provide an error bar on top of this average. This score has a very simple application-oriented meaning: It is the precision that the user can expect to obtain on the J couplings, if the hardware was used to compare the spectrum of benzene with simulations.

In the right panel of Fig. 6, we plot the root-mean-square error ΔJ obtained by running the benchmark on noisy simulated circuits, for different error probability per two-qubit gate and different Trotter steps δt . At low noise level, we observe that small Trotter steps are more able to recover the true values of the J couplings. This is expected as at low error rate, Trotter errors dominate. For these small Trotter steps, increasing the error rate blurs the NMR signal and decreases the precision. At larger error rate, larger Trotter steps perform better because in this regime, noise dominates over Trotter error, and circuits with large Trotter steps have fewer gates.

Some comments on the generality of this benchmark are in order. Contrary to the previous benchmarks presented in Secs. III and IV, the size of the benchmark system we propose cannot be scaled arbitrarily. The computation of the score that we defined requires exact knowledge of the spectrum, which can be done classically with state-vector simulation only up to $\lesssim 20$ spinful nuclei. This is justified by the fact that, first, there are actual potential use cases beyond classical simulability that do not require much more qubits, less than 100 [43,44], and second, the phenomenon of dilution of error ensures that the gate fidelity required to accurately simulate NMR experiments does not scale with system size [8,44]. Hence, instead, this benchmark is meant to evaluate the ability of a hardware (in the future, potentially with quantum error correction) to simulate long time evolution with deep circuits, through a concrete application use case.

D. Summary

The benchmark protocol evaluates the ability of the hardware to simulate NMR experiments. The system considered is a small benzene molecule. Hamiltonian simulation is implemented with a Trotter decomposition. The protocol is directly aimed at estimating the precision that one can obtain on the J couplings of nuclei spins by comparing the NMR experiment to simulations. The score is this average precision obtained.

E. Comparison with criteria

Let us now study how well the criteria defined in Sec. II B fit with this benchmark. This benchmark involves the simulation of a short-range spin Hamiltonian on a three-dimensional graph. The ability of the quantum computer to simulate such Hamiltonians is already tested in the first and second benchmarks in Secs. III and IV. We thus decided to focus this benchmark on the actual application targeted (NMR experiment simulation) rather than to fulfill all the criteria of

Sec. II B. The criterion that is the best fulfilled is thus the application orientedness, since the score directly measures the precision that one can obtain on the J couplings of a benzene molecule, by comparing the quantum computer simulation to the NMR experiment. This comes at the cost of absence of scalability, since the benchmark requires to know the exact solution to compute the score. The benchmark is also rather costly, to remove shot noise and compute enough time points to obtain good precision. On the other hand, the benchmark is difficult to spoof, especially for high scores, since this requires the simulation of a very precise spectrum, and the benchmark is also hardware-agnostic, since it involves only a few qubits with only short-range couplings.

VI. APPLICATION: GROUND-STATE ENERGY OF MOLECULES

A. Context and motivation

One of the main tasks of quantum chemistry is the determination of chemical reaction rates. This requires the knowledge of the ground-state energy of molecules as a function of their geometry with high precision. For intermediate to large numbers of orbitals, reaching this high precision becomes a difficult or impossible task with classical computers.

Mathematically, the Hamiltonian of a molecule decomposed onto N orbitals (i.e., qubits) can be written as

$$H = \sum_{i,j} h_{ij} c_i^\dagger c_j + \sum_{i,j,k,l} h_{ijkl} c_i^\dagger c_j^\dagger c_k c_l, \quad (47)$$

with h_{ij} , h_{ijkl} some coefficients. This Hamiltonian is then usually expressed in terms of Pauli matrices through a Jordan-Wigner transformation. Although bearing many similarities with condensed matter systems, these chemical problems have two important specificities: The number of terms in the Hamiltonian is large and the precision required on the ground-state energy is high, called chemical accuracy, on the order of 1 millihartree. This necessitates using different techniques than the Trotter algorithm to implement the time evolution, because Trotter error incurs a heating that precludes reaching the ground state with the precision required. This has been studied in a previous paper [47] and it was found that using another algorithm for the time evolution that is free of any discretization error would save a very significant number of gates to obtain chemical accuracy with an adiabatic algorithm for a number of small molecules.

B. The benchmark

We define the benchmarking system to be a linear chain of L hydrogen atoms, each separated by a distance $d = 0.74$ nm, decomposed in the STO-3G basis set. This system is sketched in Fig. 7. They are defined on $N = 2L$ qubits. The fermionic basis is optimized using restricted Hartree-Fock. The resulting coefficients of the second-quantized Hamiltonian in Eq. (47) can be computed with standard libraries such as PySCF [48–50]. In case of an odd number L of hydrogen atoms, we remove one electron in order to keep an even number of electrons and be able to run the restricted Hartree-Fock optimization. We then decompose the Hamiltonian into Pauli strings using a Jordan-Wigner transformation. Next, we use

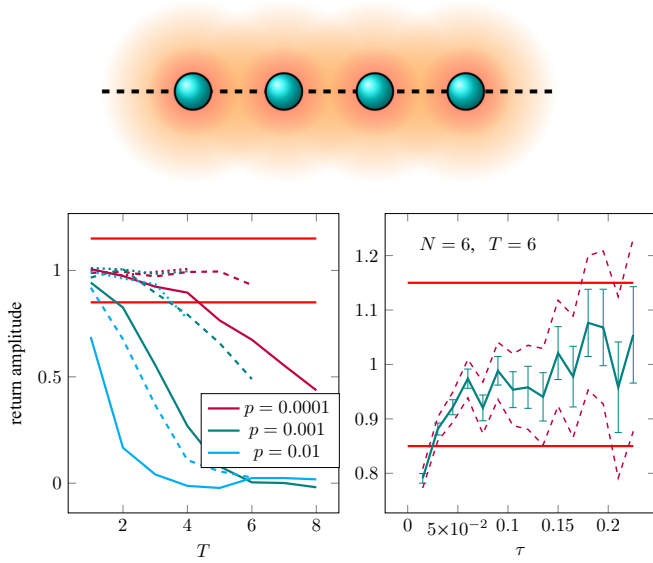


FIG. 7. Top: Depiction of the system benchmark, a hydrogen chain. Bottom left: Return amplitude measured as a function of T , for different error rates, at optimal gate angle τ , in size $N = 4$ (dotted), $N = 6$ (dashed), and $N = 8$ (solid). The red lines indicate the thresholds for the size to be validated at $T = N$. Bottom right: Return amplitude with error bars as a function of the gate angle τ for a fixed number of shots, with error rate $p = 0.001$, size $N = 6$, and time $T = N$. The purple dashed line indicates the expectation value plus or minus two error bars, which is used as the criterion for passing the test in the benchmark.

particle number conservation to add to the Hamiltonian the quantity $-\alpha(\sum_{j=1}^N Z_j)^2$ without changing its eigenstates. The coefficient α is taken to be the median of the coefficients in front of terms $Z_j Z_k$ in the Pauli string decomposition of the Hamiltonian, as this allows one to minimize the 1-norm of the Hamiltonian, i.e., the sum of the absolute values of the coefficients. Indeed, the gate cost of the algorithm that we are going to use depends on simulation time t and 1-norm as $\mathcal{O}(t^2 \mu^2)$, so reducing the 1-norm significantly spares resources. In this way, we obtain a decomposition of the Hamiltonian

$$H_F = \sum_n c_n P_n, \quad (48)$$

with c_n some coefficients and P_n Pauli strings on N qubits. Up to changing P_n into $-P_n$, we will assume $c_n > 0$. These decompositions are given in a Zenodo repository for several chain sizes [51].

We then define the time-dependent Hamiltonian

$$H(t) = \left(1 - \frac{t}{T}\right) H_I + \frac{t}{T} H_F, \quad (49)$$

where $0 \leq t \leq T$ with T a total adiabatic time, and with H_I containing only the $c_n P_n$ terms of H_F where the Pauli string P_n is a single Z term (located at any site). This time-dependent Hamiltonian has been studied in Ref. [47]. It implements an adiabatic evolution from a diagonal Hamiltonian H_I , whose ground state is the Hartree-Fock state $|\text{HF}\rangle = |1 \cdots 10 \cdots 0\rangle$, to the target Hamiltonian H_F , whose ground-state energy is the sought quantity. It has been shown numerically up to

$N = 20$ that an adiabatic time $T = N$ is sufficient to prepare a state whose energy is within chemical accuracy of the ground-state energy, i.e., such that the energy difference is smaller than 10^{-3} . For this benchmark, we propose the implementation of this adiabatic state preparation:

$$|\psi_f\rangle = U(T)|\text{HF}\rangle, \quad (50)$$

where $U(T)$ implements the time-dependent Hamiltonian evolution [Eq. (49)] up to time $t = T = N$. To implement this time evolution, we propose the randomized algorithm of Ref. [52]. This algorithm allows for an exact implementation of the Hamiltonian dynamics, without any Trotter error, while still displaying a finite average number of gates in each circuit. We briefly review here how this algorithm works. We define the time-dependent coefficients $c_n(t) = c_n$ if P_n is a single Pauli $\pm Z$, and $c_n(t) = \frac{t}{T} c_n$ otherwise. The algorithm consists in generating random unitary operators U that depend on a chosen gate angle $0 < \tau < \pi/2$, with the property that their average is proportional to the evolution operator. Namely, we have

$$\mathbb{E}[U] = \lambda \mathcal{T} e^{i \int_0^T ds H(s)}, \quad (51)$$

where $\mathbb{E}$ denotes the statistical average over the random unitary U , λ denotes an attenuation factor equal to

$$\lambda = \exp \left(-\tan(\tau/2) \sum_n \int_0^T c_n(t) dt \right), \quad (52)$$

and $\mathcal{T} e^{i \int_0^T ds H(s)}$ denotes the time-ordered exponential implementing an evolution according to the time-dependent Hamiltonian $H(s)$. The unitaries U are generated as a product of Pauli string rotations:

$$U = e^{i\tau P_{n_1}} \cdots e^{i\tau P_{n_M}}, \quad (53)$$

where the application of each rotation $e^{i\tau P_n}$ is a random event following a Poisson process with time-dependent rate $c_n(t)/\sin \tau$ between times 0 and T . Namely, the probability that the rotation $e^{i\tau P_n}$ is applied between time t and $t + dt$ is given by $c_n(t) dt / \sin \tau$. This is described precisely in Refs. [47, 52].

We thus introduce an ancilla and initialize the total state on $N + 1$ qubits in

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle \otimes |\text{HF}\rangle + |1\rangle \otimes |\text{HF}\rangle). \quad (54)$$

We then generate a random unitary U according to the previous random process, and apply it on the system qubits, conditioned on the ancilla being 1.

Denoting ρ_i the mixed state obtained after running this random time evolution, we have

$$\begin{aligned} \rho_i &= \frac{1}{2} |0\rangle\langle 0| \otimes |\text{HF}\rangle\langle \text{HF}| + \frac{1}{2} |1\rangle\langle 1| \otimes \rho_{\text{unkn}} \\ &\quad + \frac{\lambda}{2} |0\rangle\langle 1| \otimes |\text{HF}\rangle\langle \psi_f| + \frac{\lambda}{2} |1\rangle\langle 0| \otimes |\psi_f\rangle\langle \text{HF}|, \end{aligned} \quad (55)$$

with $|\psi_f\rangle$ the target state in Eq. (50), with ρ_{unkn} some unknown density matrix, and with λ given in Eq. (52).

Repeating the same process on this output density matrix, but conditioning the ancilla to be 0 instead of 1 (and of course,

generating a different random Poisson process), we obtain the density matrix

$$\rho_f = \frac{1}{2}|0\rangle\langle 0| \otimes |\text{HF}\rangle\langle \text{HF}| + \frac{1}{2}|1\rangle\langle 1| \otimes \rho'_{\text{unkn}} + \frac{\lambda^2}{2}|0\rangle\langle 1| \otimes |\psi_f\rangle\langle \psi_f| + \frac{\lambda^2}{2}|1\rangle\langle 0| \otimes |\psi_f\rangle\langle \psi_f|, \quad (56)$$

with ρ'_{unkn} another unknown density matrix. By taking expectation value of X on the ancilla, one gets access to the exact state $|\psi_f\rangle$

$$\text{tr}[X_a \rho_f] = \lambda^2 |\psi_f\rangle\langle \psi_f|. \quad (57)$$

Namely, the expectation value of any observable $\mathcal{O}$ within $|\psi_f\rangle$ can be obtained as

$$\langle \psi_f | \mathcal{O} | \psi_f \rangle = \lambda^{-2} \text{tr}[(X \otimes \mathcal{O}) \rho_f]. \quad (58)$$

By setting $\mathcal{O} = I$, the left-hand side is equal to 1, and so $\text{tr}[X_a \rho_f]$ must be equal to λ^2 . The agreement of the benchmarked hardware with that theoretical expectation value gives a way of evaluating the quality of the computation.

In this randomized algorithm, the end user can choose the gate angle τ without influence on the result [Eq. (58)]. Changing the gate angle τ , however, modifies the average number of gates in the circuit and the value of λ . The number of gates in the circuit is proportional to $1/\sin \tau$, and the attenuation factor is given in Eq. (52). The number of shots to perform to obtain a given precision on $\langle \psi_f | \mathcal{O} | \psi_f \rangle$ in Eq. (58) scales as λ^{-4} . Hence, changing τ allows for balancing the number of gates in the circuit (and hence the noise) and the number of shots to perform. Increasing τ decreases linearly the number of gates in the circuit, but increases exponentially the number of shots to perform. Which τ to choose depends on the hardware: Fast architectures where large numbers of shots can be done prefer larger values of τ ; slower but more precise architectures prefer smallest values of τ . There is a choice of τ that minimizes the total number of gates to implement to reach a certain precision on a noiseless perfect hardware, approximately equal to $\tau = 1/(\int_0^T c_n(t) dt)$ [52]. However, in the presence of noise, larger values of τ might be more efficient. We therefore leave to the end user the freedom to choose the gate angle τ . This benchmark setup thus automatically balances gate fidelity and clock speed.

C. The score

We assign the following score to the benchmark. We say that the hardware passes the test in size N if the return amplitude plus or minus two error bars at time $T = N$ is contained around 1 plus or minus a threshold value Θ that we set to $\Theta = 0.15$. Namely, let us denote by E the expectation value obtained for the quantity $\lambda^{-2} \text{tr}[(X \otimes I) \rho_f]$ (i.e., λ^{-2} times the expectation value of X on the ancilla), and δE one standard deviation on the estimate, when run at time $T = N$. Then, we say that the hardware passes the test at size N if

$$0.85 = 1 - \Theta \leq E \pm 2\delta E \leq 1 + \Theta = 1.15. \quad (59)$$

Then, the score S_{QC} assigned to this quantum chemistry benchmark is the largest system size N for which the hardware passes the test.

In Fig. 7, we show a run of this benchmark for small system sizes $N = 4, 6, 8$. In the left panel, we show the return amplitude measured as a function of T for different system sizes and error rates, when choosing the gate angle τ to be the optimal value. Here, all system sizes fail the test for error rate $p = 0.01$. For $p = 0.001$, only $N = 4$ passes the test. For $p = 0.0001$, $N = 4$ and $N = 6$ pass the test, but not $N = 8$. In the right panel, we show the effect of gate angle in the case $N = 6$ and $T = 6$ at error rate $p = 0.001$. At optimal gate angle, the test fails. However, one sees that by increasing gate angle one can decrease the effect of noise so as to obtain a return amplitude above the threshold. However, if one increases the gate angle too much, error bars grow and the test fails again. This shows that this benchmark allows the user to take advantage of a high clock speed that allows for running a high number of shots, and so increasing the gate angle τ to mitigate the effect of hardware imperfections.

D. Summary

The benchmark protocol evaluates the ability of the hardware to implement an algorithm for preparing the ground state of small molecules. The systems considered are hydrogen chains in an STO-3G basis. The algorithm used is an adiabatic evolution implemented with a stochastic algorithm that simulates the exact Hamiltonian evolution without any discretization error. The score is based on a consistency check similar to a mirror circuit, and is scalable.

E. Comparison with criteria

Let us now study the definition of the benchmark protocol in the light of the criteria defined in Sec. II B. The main strength of the benchmark is its scalability, since the expectation value of X on the ancilla can be computed classically for arbitrary system sizes. Since this value is nontrivial, and since the circuits run are random circuits, the benchmark is difficult to spoof. One other strength of the benchmark is that it allows for balancing number of shots and number of gates through varying the gate angle of the stochastic algorithm used, and gives the possibility to a fast but less precise hardware to do a lot of shots to compensate for a low gate fidelity. The benchmark protocol is thus particularly hardware-agnostic. However, the value measured does not bring information on the chemical system considered, and is similar to the information obtained from a mirror circuit (although in our case, the observable measured is local). This benchmark protocol is thus the least application-oriented of the five benchmark protocols we defined. Regarding the cost of the benchmark, it is quite expensive to implement, but this is also a property of the chemical system studied, their dense Hamiltonian, and the high precision required on the ground-state energy. It is thus an unavoidable drawback of benchmark protocols defined from chemistry systems.

VII. APPLICATION: CLASSICAL OPTIMIZATION

A. Context and motivation

Classical optimization problems consist in finding the minimum of a cost function over a (usually) discrete set of

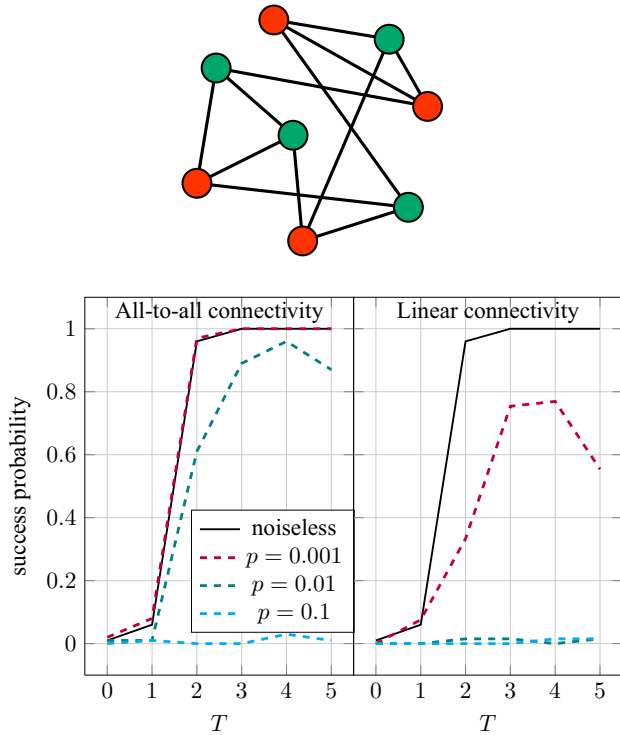


FIG. 8. Top: Example of a 3-regular graph with an optimal coloring maximizing the number of edges between green and red nodes. Bottom: Probability of finding the optimal cut in a graph of size $N = 20$ in 10^3 shots, as a function of T in the benchmark setting, for different noise levels p , comparing all-to-all connectivity and linear connectivity. This probability is estimated from 100 runs of the protocol, so the curves plotted display a slight randomness $< 1\%$. The all-to-all connected simulated hardware passes the benchmark for $p = 0.001$ and $p = 0.01$, while the linear-connected simulated hardware passes the benchmark only for $p = 0.001$.

configurations, such as the traveling salesman problem or the knapsack problem. What makes these problems attractive to quantum computing is, first, the (quasi)guarantee that these problems cannot be solved classically in polynomial time (otherwise $P = NP$), ensuring that they will always become impossible to solve classically provided the system size is large enough, and second, the wide relevance of these problems to several sectors of the industry. In quantum computing, they can be formulated as finding the ground state of a *classical* Hamiltonian H , namely, that contains only Z Pauli matrices. The simplest optimization problem in this formulation is the so-called Max-Cut problem, whose Hamiltonian is

$$H = \sum_{\langle i, j \rangle} Z_i Z_j, \quad (60)$$

where $\langle i, j \rangle$ means that sites i, j are neighbors on a given graph. The ground state of H is a product state in the Z basis whose values 0, 1 partition the graph into two subgraphs such that the number of edges connecting one subgraph to the other is maximal. We show in Fig. 8 an example of a graph with such a maximal partition.

One way of finding the ground state of H on a quantum computer is to use the adiabatic algorithm. Given a Trotter

step δt and a number of steps T , we implement the unitary operator

$$U(T) = W_1(\delta t) \cdots W_{3\delta t/T}(\delta t) W_{2\delta t/T}(\delta t) W_{\delta t/T}(\delta t), \quad (61)$$

with

$$W_s(\delta t) = \prod_{\langle j, k \rangle} e^{is\delta t Z_j Z_k} \prod_j e^{-i(1-s)\delta t X_j}. \quad (62)$$

Preparing initially the quantum computer in the state $|+\cdots+\rangle$, provided T is large enough and δt small enough, the final state obtained

$$|\psi\rangle = U|+\cdots+\rangle \quad (63)$$

should have large overlap with the ground state of H . By measuring the qubits in the Z basis, one obtains a list of bits that should have a non-negligible probability to provide a solution to the Max-Cut problem.

B. The benchmark

For this benchmark, we will fix the graphs to be 3-regular graphs, i.e., graphs in which every vertex has exactly three neighbors. We fix moreover the Trotter step δt to be equal to $\delta t = 0.25$. This setup has been extensively tested in Ref. [53] and it has been observed that taking $T = \mathcal{O}(L)$ is enough to be able to find the ground state, for systems up to size ≈ 100 . While the ground state of these systems cannot be found classically for arbitrary system sizes, there exist classical approximate solvers that are very likely to be able to find the exact ground state in reasonable runtime up to sizes $\sim 10^3$ [53,54]. This ensures that the benchmark can be implemented on hardware for probably several years to come. Even beyond the classically simulatable regime, different hardware can still be compared to each other.

We note that, beyond the classically simulatable system sizes and approximate evolutions for larger system sizes, we do not have any guarantee that the algorithm still succeeds in finding the optimal cuts for arbitrary system sizes. However, this problem is common to other application-oriented benchmarks of NP-hard problems [23]. In the absence of proof of efficiency of an algorithm, we can only rely on simulatable system sizes or heuristics to hope for future efficiency.

C. The score

We say that a given hardware is able to solve a graph G if there is experimental evidence for the existence of a value of adiabatic time T and of a number of shots N_S , such that by measuring Eq. (63) in the Z basis N_S times, the optimal solution is obtained with probability larger than $1/2$. The end user is free to choose an appropriate value of adiabatic time T and of number of shots N_S . To be able to claim that a given hardware solves the graph, we require that the user runs a minimum of ten groups of N_S shots, and that counting 1 for each group of shots containing the optimal solution, and 0 otherwise, the mean value of this random variable is larger than 0.5 by two standard deviations.

Then, we say that a given hardware passes the benchmark in size N if there exists at least one connected 3-regular graph on N sites that the given hardware is able to solve. We define

the score $S_{\text{Max-Cut}}$ to be the largest system size N for which the hardware passes the test. A refinement of the score can be made by giving, for that value of N and T , the average time to solution defined as N_S times the average run-time of one shot.

We note that by increasing the number of shots N_S , we can always obtain a non-negligible probability of measuring the optimal solution by just random guess. This feature is not a loophole of the benchmark. While this strategy can be implemented for small system sizes, it would require scaling N_S exponentially with N for larger N and quickly becomes impractical. From an application point of view, a hardware that is able to run a large number of shots quickly should indeed be considered more powerful than a slow hardware, all other things being equal. Imposing a number of shots N_S would set an intrinsic timescale that could be detrimental to certain hardware or become obsolete in the future, if machines become faster or instead slower due to, e.g., error correction.

In Fig. 8, we present some numerical noisy simulations of this benchmark, showing the probability of finding the optimal cut as a function of T in a given graph of size $N = 20$, for a number of shots $N_S = 10^3$. We compare two simulated hardware architectures, one where all qubits are connected to each other, and another one where gates can be applied only between neighboring qubits on a line, requiring the implementation of additional SWAP gates to connect arbitrary qubits. With all-to-all connectivity, the simulated hardware would pass the benchmark for two-qubit error rate $p = 0.001$ and $p = 0.01$, but would fail for $p = 0.1$, because no value of T leads to a success probability larger than $1/2$. With linear connectivity, only for $p = 0.001$ would it pass the benchmark.

D. Summary

The benchmark protocol evaluates the ability of the hardware to solve a Max-Cut problem using an adiabatic algorithm. The systems considered are random 3-regular graphs. Hamiltonian simulation is implemented with a Trotter decomposition. The score is given by the maximal system size at which there exists one such graph for which the hardware is able to find the maximal cut with non-negligible probability.

E. Comparison with criteria

Let us now justify the definition of the benchmark protocols, in the light of the criteria defined in Sec. II B. First, the benchmark is application-oriented, as it measures the size of the largest graph for which the quantum computer can find the optimal cut. Second, it is relatively cheap. The circuits involved are Trotter decomposition of a local Hamiltonian with a number of gates scaling only linearly with system size. The user is free to choose the number of shots. Since the cut of the graph can be computed on every single shot, no averaging over shots is required, so a relatively modest number of shots can suffice to run the benchmark. Third, although the benchmark can be spoofed for small system sizes by doing a large number of shots and finding the optimal solution by chance, this technique is exponentially costly in system size and cannot be reasonably implemented for larger graphs with say $N > 20$.

Let us consider now the two remaining criteria, i.e., whether the benchmark is scalable and hardware-agnostic. The fact that the number of shots is a free parameter of the benchmark helps the benchmark be hardware-agnostic, as it can be implemented cheaply on both slow and fast devices. However, the 3-regular graphs generated for the benchmark involve many long-distance couplings between the vertices, which translate into two-qubit gates applied on qubits far apart. The benchmark is thus certainly more easily implemented on all-to-all coupled quantum computers than on gridlike architectures. On the other hand, this long-range connectivity in the graph is required for the Max-Cut problem to not be efficiently solvable by classical means. It is thus imposed by the nature of the application we aim at solving. Considering only graphs that can be easily implemented on gridlike architectures would make the benchmark less application-oriented. We can thus consider that the benchmark is as hardware-agnostic as it can be, for this precise application of Max-Cut on 3-regular graphs.

Finally, the benchmark as defined is strictly speaking not scalable, as it requires the knowledge of the exact maximal cut. It also relies on the actual efficiency of the Floquet adiabatic algorithm for this application, which has been tested with simulations only up to around 100 qubits. However, for this Max-Cut problem on 3-regular graphs, the exact solution can be computed for much larger system sizes than for regular Hamiltonian simulation of physical systems. In practice, the benchmark can be run up to system sizes of order 10^3 , which guarantees it to be relevant for several years to come. Thus, although the benchmark cannot be run for arbitrary system sizes, it can be run for a number of qubits way beyond the classical simulable regime of the quantum circuit.

VIII. CONCLUSION

We have introduced an application-oriented benchmarking suite for quantum computers that is focused on Hamiltonian simulation. We have defined five different benchmark settings that correspond to some of the most prominent potential applications of quantum computing, namely, material and condensed matter physics simulation (dynamic problems and static problems), Nuclear Magnetic Resonance, quantum chemistry, and classical optimization. Specifically, we presented explicit benchmark settings for (1) computing the dynamics of electronic systems, (2) computing the values of static observables of condensed matter physics at low temperature, (3) computing the spectrum generated by nuclear magnetic resonance experiments, (4) preparing the ground state of a hydrogen chain in quantum chemistry, and (5) solving the Max-Cut problem on 3-regular graphs.

A scalable application-oriented benchmark can be sometimes contradictory, as benchmarking supposes to know the exact result, whereas the best applications of quantum computing are those beyond reach of classical computers. We tried to slalom between these contradictions and defined different settings that, although not all scalable and not all implementing an end-to-end quantum computing application, address a variety of circuit geometries, application practicality, qubit connectivities, and scalability properties that altogether should draw an accurate overview of the ability of a

given quantum computing hardware to solve some real-world applications.

Besides these benchmarks, we introduced a metric to measure the capabilities of a quantum computing hardware at a given task that involves computing the expectation value of an observable. The metric is based on the idea that, since a certain minimal number of shots have to be performed on the quantum computer to reach a given precision on the expectation value, a systematic bias coming from noise might not be detectable before a certain number of shots have been performed. Stated differently, given a certain gate budget, a noisy quantum computing hardware can be in practice indistinguishable from a perfect quantum computer at a given task, if the effect of hardware imperfections is below the shot noise. We thus introduced the notion of *distinguishability cost* to measure the quality of a quantum computing hardware at a given task, as the minimal number of gates that a perfect quantum computer has to run to certify that the output of the benchmarked hardware is incorrect. The appeal of this score is that it is universally applicable to any problem involving expectation values, and outputs a number with direct physical and practical meaning.

ACKNOWLEDGMENTS

We thank Yi Hsiang Chen, Daniel Mills, and Kushal Seetharam for comments on the draft. The project was funded by the Bavarian Ministry of Economic Affairs, Regional Development and Energy (StMWi) under project Bench-QC (Grant No. DIK0425/01).

DATA AVAILABILITY

The data that support the findings of this article are openly available [55].

APPENDIX A: TORIC CODE STATE PREPARATION

In Fig. 9, we graphically represent the toric code state preparation used in Sec. III B, for the case $L_x = L_y = 8$. The toric code state preparation for other even dimensions L_x, L_y is readily deduced from Fig. 9.

APPENDIX B: FREE FERMION CALCULATIONS

1. Generalities

We consider the free fermion Hamiltonian

$$H = \sum_{\langle i,j \rangle} c_i^\dagger c_j + c_j^\dagger c_i, \quad (\text{B1})$$

where the sum runs over the edges of a $L_x \times L_y$ square lattice with periodic boundary conditions. After a Jordan-Wigner

transformation, the term $c_i^\dagger c_j + c_j^\dagger c_i$ becomes

$$c_i^\dagger c_j + c_j^\dagger c_i = \frac{1}{2}(X_i X_j + Y_i Y_j) \prod_{i \ll k \ll j} Z_k, \quad (\text{B2})$$

and we have

$$c_i c_j - c_i^\dagger c_j^\dagger = \frac{1}{2}(X_i X_j - Y_i Y_j) \prod_{i \ll k \ll j} Z_k, \quad (\text{B3})$$

where $i \ll k \ll j$ means that site k is comprised between sites i and j in a given ordering of all the sites. We decompose

$$H = H_{\uparrow,X} + H_{\uparrow,Y} + H_{\rightarrow,X} + H_{\rightarrow,Y}, \quad (\text{B4})$$

with

$$H_{\uparrow,X} = \frac{1}{2} \sum_{\substack{\langle i,j \rangle \\ \text{vertical}}} X_i X_j \prod_{i \ll k \ll j} Z_k, \quad (\text{B5})$$

and identically for $\rightarrow$ meaning that $\langle i, j \rangle$ is a horizontal bond, and with $H_{\uparrow,Y}$ being the same but with $Y_i Y_j$ instead of $X_i X_j$. In terms of fermions, we have

$$H_{\rightarrow,X/Y} = \frac{1}{2} \sum_{\substack{\langle i,j \rangle \\ \text{horizontal}}} c_i^\dagger c_j + c_j^\dagger c_i \pm (c_i c_j - c_i^\dagger c_j^\dagger). \quad (\text{B6})$$

2. Time evolution of Fourier modes

We rewrite this Hamiltonian with the Fourier transform

$$c_j = \frac{1}{\sqrt{L}} \sum_{k \in K} c(k) e^{ijk}, \quad (\text{B7})$$

where $K = \{\frac{2\pi(k_x, k_y)}{L_x}, k_{x,y} = 0, \dots, L_{x,y} - 1\}$. In this expression, we see site j as a couple $j = (j_x, j_y)$ with $j_{x,y} = 0, \dots, L_{x,y} - 1$ and the scalar product defined as $jk = j_x k_x + j_y k_y$. This yields

$$H_{\rightarrow,X/Y} = \sum_{k \in K} c^\dagger(k) c(k) \cos k_x \pm \frac{i}{2} \sin k_x (c^\dagger(k) c^\dagger(-k) - c(k) c(-k)). \quad (\text{B8})$$

We have

$$\begin{aligned} [H_{\rightarrow,X/Y}, c(k)] &= -\cos k_x c(k) \mp i \sin k_x c^\dagger(-k), \\ [H_{\rightarrow,X/Y}, c^\dagger(-k)] &= \cos k_x c^\dagger(-k) \pm i \sin k_x c(k). \end{aligned} \quad (\text{B9})$$

Hence, we have the evolution equation under $H_{\rightarrow,X/Y}$:

$$\partial_t \begin{pmatrix} c(k) \\ c^\dagger(-k) \end{pmatrix} = \begin{pmatrix} -i \cos k_x & \mp \sin k_x \\ \pm \sin k_x & i \cos k_x \end{pmatrix} \begin{pmatrix} c(k) \\ c^\dagger(-k) \end{pmatrix}. \quad (\text{B10})$$

Let us perform this evolution for a time δt for $H_{\rightarrow,X}$, and then for a time δt for $H_{\rightarrow,Y}$. Using a symbolic software, we find that the vector after this evolution is

$$U_k \begin{pmatrix} c(k) \\ c^\dagger(-k) \end{pmatrix}, \quad (\text{B11})$$

with

$$U_k = \begin{pmatrix} 1 - 2 \sin^2(\delta t) \cos^2 k - i \sin(2\delta t) \cos k & i \sin^2(\delta t) \sin(2k) \\ i \sin^2(\delta t) \sin(2k) & 1 - 2 \sin^2(\delta t) \cos^2 k + i \sin(2\delta t) \cos k \end{pmatrix}. \quad (\text{B12})$$

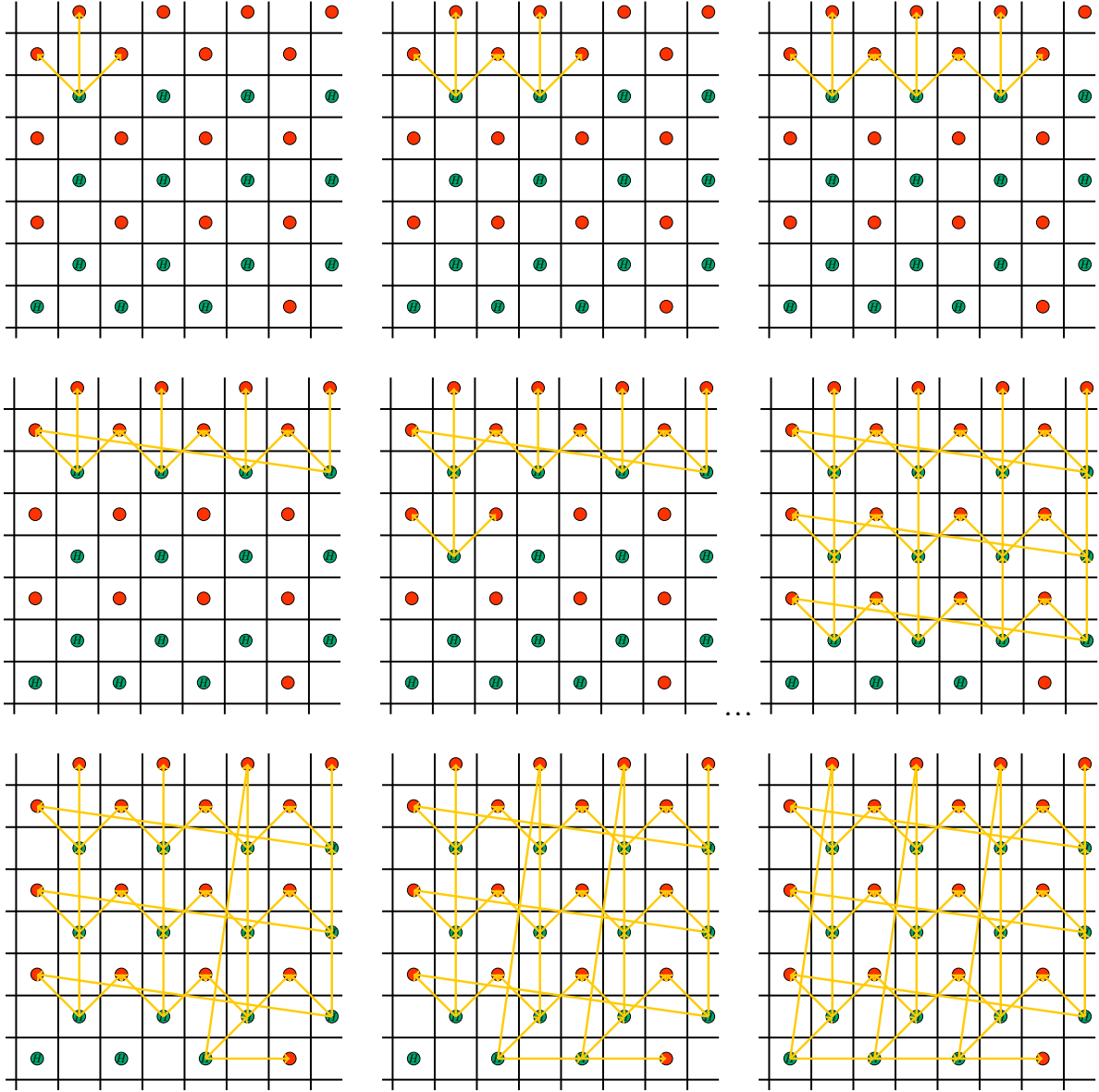


FIG. 9. The different steps in the toric code state preparation in size 8×8 , ordered in reading direction. The circles represent the ancillas, with green circles representing those where a Hadamard gate is applied at the beginning. Then, the yellow arrows indicate the application of CNOTs, with the arrow pointing toward the target qubit. The system qubits represented by blue circles in Fig. 1 are not shown in this picture.

The same equations hold true for $H_{\uparrow,X/Y}$, with k_x replaced by k_y . It follows that after a full Trotter step, the operators $c(k)$, $c^\dagger(-k)$ are mapped to

$$\begin{pmatrix} c(k) \\ c^\dagger(-k) \end{pmatrix} \mapsto U_{k_y} U_{k_x} \begin{pmatrix} c(k) \\ c^\dagger(-k) \end{pmatrix}. \quad (\text{B13})$$

3. Observable with two fermions

Let us now consider an observable of the form

$$\mathcal{O} = \sum_j f_j c_j^\dagger c_j, \quad (\text{B14})$$

with f_j some function of the site j . We have

$$\mathcal{O} = \sum_{k,q \in K} \hat{f}(k-q) c^\dagger(k) c(q), \quad (\text{B15})$$

with

$$\hat{f}(k) = \frac{1}{L} \sum_j e^{ijk} f_j. \quad (\text{B16})$$

After application of n Trotter steps, let us write the decomposition

$$(U_{k_y} U_{k_x})^n = \begin{pmatrix} \alpha_n(k) & \beta_n(k) \\ -\beta_n^*(k) & \alpha_n^*(k) \end{pmatrix}, \quad (\text{B17})$$

with $\alpha_n(k)$, $\beta_n(k)$ coefficients. Writing explicitly the unitary matrix $U_{k_x} U_{k_y}$, one finds that its eigenvalues are $e^{\pm i\epsilon_k}$ with

$$\epsilon_k = \text{sgn}(\cos k) \arccos[1 - 2 \sin^2(\delta t)(\cos k + \cos q)^2 + 4 \sin^4(\delta t) \cos k \cos q (1 + \cos(k+q))]. \quad (\text{B18})$$

One then knows that the coefficients $\alpha_n(k)$, $\beta_n(k)$ are a linear combination of $e^{in\epsilon_k}$ and $e^{-in\epsilon_k}$. From the cases $n = 0$, $n = 1$, one then finds

$$\begin{aligned}\alpha_n(k) &= e^{-in\epsilon_k} + i(-\sin(2\delta t)(\cos k_x + \cos k_y) + 2\sin(2\delta t)\sin^2(\delta t)\cos k_x \cos k_y(\cos k_x + \cos k_y) + \sin \epsilon_k) \frac{\sin(n\epsilon_k)}{\sin \epsilon_k}, \\ \beta_n(k) &= (i\sin^2(\delta t)(\sin(2k_x) + \sin(2k_y)) - 2i\sin^4(\delta t)(\cos^2(k_x)\sin(2k_y) + \cos^2(k_y)\sin(2k_x)) \\ &\quad + \sin^2(\delta t)\sin(2\delta t)(\cos(k_x)\sin(2k_y) + \cos(k_y)\sin(2k_x))) \frac{\sin(n\epsilon_k)}{\sin \epsilon_k}.\end{aligned}\quad (\text{B19})$$

We thus have after n Trotter steps

$$\begin{aligned}\mathcal{O}(n\delta t) &= \sum_{k,q \in K} \hat{f}(k-q)[\alpha_n^*(k)\alpha_n(q)c^\dagger(k)c(q) \\ &\quad + \beta_n^*(k)\alpha_n(q)c(-k)c(q) + \alpha_n^*(k)\beta_n(q)c^\dagger(k)c^\dagger(-q) \\ &\quad + \beta_n^*(k)\beta_n(q)c(-k)c^\dagger(-q)].\end{aligned}\quad (\text{B20})$$

Let us evaluate it in a product state with mode occupation n_j on site j . Introducing

$$\hat{n}(k) = \frac{1}{L} \sum_j e^{ijk} n_j, \quad (\text{B21})$$

we get that $\langle c^\dagger(k)c(q) \rangle = \hat{n}(k-q)$, $\langle c(-k)c(q) \rangle = 0$, $\langle c^\dagger(k)c^\dagger(-q) \rangle = 0$, and $\langle c(-k)c^\dagger(-q) \rangle = \delta_{k,q} - \hat{n}(k-q)$. Hence,

$$\begin{aligned}\langle \mathcal{O}(n\delta t) \rangle &= \sum_{k,q \in K} \hat{f}(k-q)(\alpha_n^*(k)\alpha_n(q) \\ &\quad - \beta_n^*(-k)\beta_n(-q)\hat{n}(k-q) + \hat{f}(0) \sum_{k \in K} |\beta_n(k)|^2).\end{aligned}\quad (\text{B22})$$

4. Higher-weight observables

We now consider the higher-weight observables $\mathcal{O}_{[w]}$ defined as

$$\mathcal{O}_{[w]} = \sum_{i_1 < \dots < i_w} f_{i_1} \dots f_{i_w} Z_{i_1} \dots Z_{i_w}, \quad (\text{B23})$$

for some f_i . The variance per shot of the observable $\mathcal{O}$ as defined in Eq. (10) involves $\mathcal{O}_{[w]}$ with $w = 2$. The observable $\mathcal{O}_{[w]}$ is exactly the coefficient in front of ϵ^w in the Taylor expansion of

$$F(\epsilon) = (1 + \epsilon f_1 Z_1)(1 + \epsilon f_2 Z_2) \dots (1 + \epsilon f_L Z_L). \quad (\text{B24})$$

Let us write

$$\begin{aligned}F(\epsilon) &= \exp \left(\sum_{j=1}^L \ln(1 + \epsilon f_j Z_j) \right) \\ &= \exp \left(\frac{1}{2} \sum_{j=1}^L \ln(1 - \epsilon^2 f_j^2) + \ln \frac{1 + \epsilon f_j}{1 - \epsilon f_j} Z_j \right).\end{aligned}\quad (\text{B25})$$

Introducing

$$S_{2w} = \sum_{j=1}^L f_j^{2w}, \quad S_{2w+1} = \sum_{j=1}^L f_j^{2w+1} Z_j, \quad (\text{B26})$$

we have

$$F(\epsilon) = \exp \left(- \sum_{w \geq 1} S_w \frac{(-\epsilon)^w}{w} \right). \quad (\text{B27})$$

From this expression, the observable $\mathcal{O}_{[w]}$ [Eq. ((B23) for all w can be expressed in terms of S_w]. For example, the first few terms are

$$\begin{aligned}\mathcal{O}_{[1]} &= S_1, \\ \mathcal{O}_{[2]} &= \frac{S_1^2 - S_2}{2}, \\ \mathcal{O}_{[3]} &= \frac{S_1^3 - 3S_1 S_2 + 2S_3}{6}, \\ \mathcal{O}_{[4]} &= \frac{S_1^4 - 6S_1^2 S_2 + 3S_2^2 + 8S_1 S_3 - 6S_4}{24}.\end{aligned}\quad (\text{B28})$$

The problem of computing $\mathcal{O}_{[w]}$ is thus reduced to that of computing the powers S_w^p . When w is even, this is only a scalar. When w is odd, in terms of the fermions, this can be written as

$$S_{2w+1}^p = \left(\sum_{j=1}^L (1 - 2c_j^\dagger c_j) f_j^{2w+1} \right)^p. \quad (\text{B29})$$

We expand it as

$$S_{2w+1}^p = \sum_{q=0}^p \binom{p}{q} (-2)^q \tilde{S}_{2w+1}^q \Sigma_{2w+1}^{p-q}, \quad (\text{B30})$$

with

$$\tilde{S}_{2w+1} = \sum_{j=1}^L f_j^{2w+1} c_j^\dagger c_j, \quad \Sigma_{2w+1} = \sum_{j=1}^L f_j^{2w+1}. \quad (\text{B31})$$

After n Trotter steps, using Eq. (B20) with $\hat{f}$ replaced by the Fourier transform of f_j^{2w+1} , the powers $\tilde{S}_{2w+1}^p$ are expressed as sums of terms of the type

$$c^\dagger(k_{i_1})c(k_{j_1}) \dots c^\dagger(k_{i_p})c(k_{j_p}), \quad (\text{B32})$$

as well as with any c replaced by $c^\dagger$, and vice versa. The expectation values of these expressions are computed using Wick's theorem. Namely, we have the recursive formula for any m :

$$\begin{aligned}\langle \bar{c}(k_1) \dots \bar{c}(k_{2m}) \rangle &= \sum_{i=2}^{2m} (-1)^{i-1} \langle \bar{c}(k_1) \bar{c}(k_i) \rangle \\ &\quad \times \langle \bar{c}(k_2) \dots \bar{c}(k_{i-1}) \bar{c}(k_{i+1}) \dots \bar{c}(k_{2m}) \rangle,\end{aligned}\quad (\text{B33})$$

where by $\bar{c}$ we mean any of c or $c^\dagger$.

- [1] M. DeCross, R. Haghshenas, M. Liu, E. Rinaldi, J. Gray, Y. Alexeev, C. H. Baldwin, J. P. Bartolotta, M. Bohn, E. Chertkov *et al.*, Computational power of random quantum circuits in arbitrary geometries, *Phys. Rev. X* **15**, 021052 (2025).
- [2] D. Bluvstein, S. J. Evered, A. A. Geim, S. H. Li, H. Zhou, T. Manovitz, S. Ebadi, M. Cain, M. Kalinowski, D. Hangleiter *et al.*, Logical quantum processor based on reconfigurable atom arrays, *Nature (London)* **626**, 58 (2024).
- [3] Y. Kim, A. Eddins, S. Anand, K. X. Wei, E. Van Den Berg, S. Rosenblatt, H. Nayfeh, Y. Wu, M. Zaletel, K. Temme *et al.*, Evidence for the utility of quantum computing before fault tolerance, *Nature (London)* **618**, 500 (2023).
- [4] M. Foss-Feig, A. Tikku, T.-C. Lu, K. Mayer, M. Iqbal, T. M. Gatterman, D. Gresh, A. Hankin, N. Hewitt, C. V. Horst *et al.*, Experimental demonstration of the advantage of adaptive quantum circuits, [arXiv:2302.03029](https://arxiv.org/abs/2302.03029).
- [5] S. A. Moses, C. H. Baldwin, M. S. Allman, R. Ancona, L. Ascarrunz, C. Barnes, J. Bartolotta, B. Bjork, P. Blanchard, M. Bohn *et al.*, A race-track trapped-ion quantum processor, *Phys. Rev. X* **13**, 041052 (2023).
- [6] F. Geissler, E. Stopfer, C. Ufrecht, N. Meyer, D. D. Scherer, F. Wagner, J. M. Oberreuter, Z. Chen, A. Farace, D. Gutina *et al.*, BenchQC—Scalable and modular benchmarking of industrial quantum computing applications, [arXiv:2504.11204](https://arxiv.org/abs/2504.11204).
- [7] B. F. Schiffer, A. F. Rubio, R. Trivedi, and J. I. Cirac, The quantum adiabatic algorithm suppresses the proliferation of errors, [arXiv:2404.15397](https://arxiv.org/abs/2404.15397).
- [8] E. Granet and H. Dreyer, Dilution of error in digital Hamiltonian simulation, *PRX Quantum* **6**, 010333 (2025).
- [9] E. Chertkov, Y.-H. Chen, M. Lubasch, D. Hayes, and M. Foss-Feig, Robustness of near-thermal dynamics on digital quantum computers, [arXiv:2410.10794](https://arxiv.org/abs/2410.10794).
- [10] J. Emerson, R. Alicki, and K. Życzkowski, Scalable noise estimation with random unitary operators, *J. Opt. B: Quantum Semiclassical Opt.* **7**, S347 (2005).
- [11] E. Knill, D. Leibfried, R. Reichle, J. Britton, R. B. Blakestad, J. D. Jost, C. Langer, R. Ozeri, S. Seidelin, and D. J. Wineland, Randomized benchmarking of quantum gates, *Phys. Rev. A* **77**, 012307 (2008).
- [12] R. Blume-Kohout, J. K. Gamble, E. Nielsen, K. Rudinger, J. Mizrahi, K. Fortier, and P. Maunz, Demonstration of qubit operations below a rigorous fault tolerance threshold with gate set tomography, *Nat. Commun.* **8**, 14485 (2017).
- [13] A. Erhard, J. J. Wallman, L. Postler, M. Meth, R. Stricker, E. A. Martinez, P. Schindler, T. Monz, J. Emerson, and R. Blatt, Characterizing large-scale quantum computers via cycle benchmarking, *Nat. Commun.* **10**, 5347 (2019).
- [14] A. W. Cross, L. S. Bishop, S. Sheldon, P. D. Nation, and J. M. Gambetta, Validating quantum computers using randomized model circuits, *Phys. Rev. A* **100**, 032328 (2019).
- [15] 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).
- [16] C. Neill, P. Roushan, K. Kechedzhi, S. Boixo, S. V. Isakov, V. Smelyanskiy, A. Megrant, B. Chiaro, A. Dunsworth, K. Arya *et al.*, A blueprint for demonstrating quantum supremacy with superconducting qubits, *Science* **360**, 195 (2018).
- [17] T. Proctor, K. Rudinger, K. Young, E. Nielsen, and R. Blume-Kohout, Measuring the capabilities of quantum computers, *Nat. Phys.* **18**, 75 (2022).
- [18] P.-L. Dallaire-Demers, M. Stęchły, J. F. Gonthier, N. T. Bashige, J. Romero, and Y. Cao, An application benchmark for fermionic quantum simulations, [arXiv:2003.01862](https://arxiv.org/abs/2003.01862).
- [19] B. T. Gard and A. M. Meier, Classically efficient quantum scalable Fermi-Hubbard benchmark, *Phys. Rev. A* **105**, 042602 (2022).
- [20] A. J. McCaskey, Z. P. Parks, J. Jakowski, S. V. Moore, T. D. Morris, T. S. Humble, and R. C. Pooser, Quantum chemistry as a benchmark for near-term quantum computers, *npj Quantum Inf.* **5**, 99 (2019).
- [21] T. Tomesh, P. Gokhale, V. Omole, G. S. Ravi, K. N. Smith, J. Vizslai, X.-C. Wu, N. Hardavellas, M. R. Martonosi, and F. T. Chong, SupermarQ: A scalable quantum benchmark suite, in *2022 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE, Seoul, South Korea, 2022), pp. 587–603.
- [22] T. Lubinski, J. J. Goings, K. Mayer, S. Johri, N. Reddy, A. Mehta, N. Bhatia, S. Rappaport, D. Mills, C. H. Baldwin *et al.*, Quantum algorithm exploration using application-oriented performance benchmarks, [arXiv:2402.08985](https://arxiv.org/abs/2402.08985).
- [23] S. Martiel, T. Ayril, and C. Allouche, Benchmarking quantum coprocessors in an application-centric, hardware-agnostic, and scalable way, *IEEE Trans. Quantum Eng.* **2**, 1 (2021).
- [24] W. van der Schoot, R. Wezeman, N. Neumann, F. Phillipson, and R. Kooij, Extending the q -score to an application-level quantum metric framework, in *2024 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Montreal, Canada, 2024), Vol. 1, pp. 941–951.
- [25] J. R. Finžgar, P. Ross, L. Hölscher, J. Klepsch, and A. Luckow, Quark: A framework for quantum computing application benchmarking, in *2022 IEEE International Conference on Quantum Computing and Engineering (QCE)* (IEEE, Broomfield, CO, USA, 2022), pp. 226–237.
- [26] K. Mesman, Z. Al-Ars, and M. Möller, QPack: Quantum approximate optimization algorithms as universal benchmark for quantum computers, [arXiv:2103.17193](https://arxiv.org/abs/2103.17193).
- [27] F. Barbaresco, L. Rioux, C. Labreuche, M. Nowak, N. Olivier, D. Nicolazic, O. Hess, A.-L. Guilmin, R. Wang, T. Sassolas *et al.*, BACQ—Application-oriented benchmarks for quantum computing, [arXiv:2403.12205](https://arxiv.org/abs/2403.12205).
- [28] Y. Dong and L. Lin, Random circuit block-encoded matrix and a proposal of quantum LINPACK benchmark, *Phys. Rev. A* **103**, 062412 (2021).
- [29] A. Cornelissen, J. Bausch, and A. Gilyén, Scalable benchmarks for gate-based quantum computers, [arXiv:2104.10698](https://arxiv.org/abs/2104.10698).
- [30] Y. Dong, K. B. Whaley, and L. Lin, A quantum Hamiltonian simulation benchmark, *npj Quantum Inf.* **8**, 131 (2022).
- [31] N. P. Sawaya, D. Marti-Dafcik, Y. Ho, D. P. Tabor, D. E. B. Neira, A. B. Magann, S. Premaratne, P. Dubey, A. Matsuura, N. Bishop *et al.*, HamLib: A library of Hamiltonians for benchmarking quantum algorithms and hardware, *Quantum* **8**, 1559 (2024).
- [32] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles, Noise-induced barren plateaus in variational quantum algorithms, *Nat. Commun.* **12**, 6961 (2021).
- [33] A. A. Agrawal, J. Job, T. L. Wilson, S. Saadatmand, M. J. Hodson, J. Y. Mutus, A. Caesura, P. D. Johnson, J. E. Elenewski,

- K. J. Morrell *et al.*, Quantifying fault tolerant simulation of strongly correlated systems using the Fermi-Hubbard model, [arXiv:2406.06511](#).
- [34] R. Nigmatullin, K. Hemery, K. Ghanem, S. Moses, D. Gresh, P. Siegfried, M. Mills, T. Gatterman, N. Hewitt, E. Granet *et al.*, Experimental demonstration of breakeven for a compact fermionic encoding, *Nat. Phys.* **21**, 1319 (2025).
- [35] C. Derby, J. Klassen, J. Bausch, and T. Cubitt, Compact fermion to qubit mappings, *Phys. Rev. B* **104**, 035118 (2021).
- [36] E. Granet, K. Hémerly, and H. Dreyer, Analog simulation of noisy quantum circuits, *Phys. Rev. Res.* **7**, 013213 (2025).
- [37] W.-Y. Liu, S.-S. Gong, Y.-B. Li, D. Poilblanc, W.-Q. Chen, and Z.-C. Gu, Gapless quantum spin liquid and global phase diagram of the spin-1/2 j_1 - j_2 square antiferromagnetic Heisenberg model, *Sci. Bull.* **67**, 1034 (2022).
- [38] J.-W. Mei, J.-Y. Chen, H. He, and X.-G. Wen, Gapped spin liquid with z_2 topological order for the kagome Heisenberg model, *Phys. Rev. B* **95**, 235107 (2017).
- [39] A. M. Läuchli, J. Sudan, and R. Moessner, $S = \frac{1}{2}$ kagome Heisenberg antiferromagnet revisited, *Phys. Rev. B* **100**, 155142 (2019).
- [40] H. J. Hogben, M. Krzystyniak, G. T. Charnock, P. J. Hore, and I. Kuprov, *Spinach*—A software library for simulation of spin dynamics in large spin systems, *J. Magn. Reson.* **208**, 179 (2011).
- [41] D. Sels, H. Dashti, S. Mora, O. Demler, and E. Demler, Quantum approximate Bayesian computation for NMR model inference, *Nat. Mach. Intell.* **2**, 396 (2020).
- [42] K. Seetharam, D. Biswas, C. Noel, A. Risinger, D. Zhu, O. Katz, S. Chattopadhyay, M. Cetina, C. Monroe, E. Demler *et al.*, Digital quantum simulation of NMR experiments, *Sci. Adv.* **9**, eadh2594 (2023).
- [43] J. E. Elenewski, C. M. Camara, and A. Kalev, Prospects for NMR spectral prediction on fault-tolerant quantum computers, [arXiv:2406.09340](#).
- [44] A. Khedri, P. Stadler, K. Bark, M. Lodi, R. Reiner, N. Vogt, M. Marthaler, and J. Leppäkangas, The impact of noise on the simulation of NMR spectroscopy on NISQ devices, [arXiv:2404.18903](#).
- [45] Q. Stern and K. Sheberstov, Simulation of NMR spectra at zero and ultralow fields from A to Z—A tribute to prof. Konstantin L'vovich Ivanov, *Magn. Reson.* **4**, 87 (2023).
- [46] A. Wilzewski, S. Afach, J. W. Blanchard, and D. Budker, A method for measurement of spin-spin couplings with sub-mHz precision using zero- to ultralow-field nuclear magnetic resonance, *J. Magn. Reson.* **284**, 66 (2017).
- [47] E. Granet, K. Ghanem, and H. Dreyer, Practicality of quantum adiabatic algorithm for chemistry applications, *Phys. Rev. A* **111**, 022428 (2025).
- [48] Q. Sun, X. Zhang, S. Banerjee, P. Bao, M. Barbry, N. S. Blunt, N. A. Bogdanov, G. H. Booth, J. Chen, Z.-H. Cui *et al.*, Recent developments in the PySCF program package, *J. Chem. Phys.* **153**, 024109 (2020).
- [49] Q. Sun, Libcint: An efficient general integral library for Gaussian basis functions, *J. Comput. Chem.* **36**, 1664 (2015).
- [50] Q. Sun, T. C. Berkelbach, N. S. Blunt, G. H. Booth, S. Guo, Z. Li, J. Liu, J. D. McClain, E. R. Sayfutyarova, S. Sharma *et al.*, PySCF: The python-based simulations of chemistry framework, *Wiley Interdiscip. Rev.: Comput. Mol. Sci.* **8**, e1340 (2018).
- [51] E. Granet and H. Dreyer, Hydrogen chain Hamiltonians, Zenodo (2025), <https://doi.org/10.5281/zenodo.16614244>.
- [52] E. Granet and H. Dreyer, Hamiltonian dynamics on digital quantum computers without discretization error, *npj Quantum Inf.* **10**, 82 (2024).
- [53] E. Granet and H. Dreyer, Benchmarking a heuristic Floquet adiabatic algorithm for the Max-Cut problem, *Sci. Rep.* **15**, 31983 (2025).
- [54] I. Dunning, S. Gupta, and J. Silberholz, What works best when? A systematic evaluation of heuristics for Max-Cut and QUBO, *INFORMS J. Comput.* **30**, 608 (2018).
- [55] E. Granet and H. Dreyer, Numerical data for AppQSim, Zenodo (2025), <https://doi.org/10.5281/zenodo.17224040>.