Editors' Suggestion   Featured in Physics

Strategies for solving the Fermi-Hubbard model on near-term quantum computers

Chris Cade,[1,*] Lana Mineh,[1,2,3] Ashley Montanaro,[1,2] and Stasja Stanisic[1]
[1]*Phasecraft Ltd, Bristol BS1 5DD, United Kingdom*
[2]*School of Mathematics, University of Bristol, Bristol BS8 1UG, United Kingdom*
[3]*Quantum Engineering Centre for Doctoral Training, University of Bristol, Bristol BS8 1FD, United Kingdom*

(Received 21 February 2020; accepted 20 October 2020; published 10 December 2020)

The Fermi-Hubbard model is of fundamental importance in condensed-matter physics, yet is extremely challenging to solve numerically. Finding the ground state of the Hubbard model using variational methods has been predicted to be one of the first applications of near-term quantum computers. Here we carry out a detailed analysis and optimization of the complexity of variational quantum algorithms for finding the ground state of the Hubbard model, including costs associated with mapping to a real-world hardware platform. The depth complexities we find are substantially lower than previous work. We performed extensive numerical experiments for systems with up to 12 sites. The results suggest that the variational ansätze we used—an efficient variant of the Hamiltonian variational ansatz and a generalization thereof—will be able to find the ground state of the Hubbard model with high fidelity in relatively low quantum circuit depths. Our experiments include the effect of realistic measurements and depolarizing noise. If our numerical results on small lattice sizes are representative of the somewhat larger lattices accessible to near-term quantum hardware, they suggest that optimizing over quantum circuits with a gate depth less than a thousand could be sufficient to solve instances of the Hubbard model beyond the capacity of classical exact diagonalization.

DOI: 10.1103/PhysRevB.102.235122

Modeling quantum-mechanical systems is widely expected to be one of the most important applications of near-term quantum computing hardware [1–3]. Quantum computers could enable the solution of problems in the domains of many-body quantum physics and quantum chemistry that are intractable for today's best supercomputers.

Quantum algorithms have been proposed for both dynamic and static simulation of quantum systems. In the former case, one seeks to approximate time evolution according to a certain quantum Hamiltonian. In many physically relevant cases, such as Hamiltonians obeying a locality constraint on their interactions, this can be carried out efficiently, i.e., in time polynomial in the system size [4]; by contrast, even to write down a classical description of the quantum system would take exponential time. However, in cases where the performance of the quantum simulation algorithm has been calculated and optimized in detail, solving a large enough problem instance to be practically relevant is still beyond the capabilities of present-day quantum computing technology. For example, several recent works describing highly-optimized algorithms for time-dynamics simulation [5–7] determine complexities in the range of 10^5–10^8 quantum gates to simulate systems beyond classical capabilities. By comparison, the most complex quantum circuit executed in the recent demonstration by Google of a quantum computation outperforming a classical supercomputer contained 430 two-qubit gates [8].

In the case of static simulation, the canonical problem is to produce the ground state of a quantum Hamiltonian. Once this state is produced, measurements can be performed to determine its properties. Although this problem is expected to be computationally hard for quantum computers in the worst case [9], it is plausible that instances of practical importance could nevertheless be solved efficiently. A prominent class of methods for producing ground states are variational methods and, in particular, the variational quantum eigensolver [10,11] (VQE). The VQE framework can be seen as a hybrid quantum-classical approach to produce a ground state of a quantum Hamiltonian H. A classical optimizer is used to optimize over quantum circuits which produce states $|\psi\rangle$ that are intended to be the ground state of H. The cost function provided to the optimizer is an approximation of the energy $\langle\psi|H|\psi\rangle$, which is estimated using a quantum computer.

Here our focus is on variational algorithms for a specific task: constructing the ground state of the iconic 2D Fermi–Hubbard model [12,13]. This model is of particular interest for several reasons. First, despite its apparent simplicity, its theoretical properties are far from fully understood [13–15]. Second, it is believed to be relevant to physical phenomena of extreme practical importance, such as high-temperature superconductivity [16]. Third, its regular structure and relatively simple form suggest that it may be easier to implement on a near-term quantum computer than, for example, model systems occurring in quantum chemistry.

The Hubbard Hamiltonian is defined as

$$H = -t \sum_{\langle i,j\rangle,\sigma} (a_{i\sigma}^{\dagger} a_{j\sigma} + a_{j\sigma}^{\dagger} a_{i\sigma}) + U \sum_{i} n_{i\uparrow} n_{i\downarrow}, \tag{1}$$

[*]Present address: QuSoft and CWI, Amsterdam, Netherlands.

where $a_{i\sigma}^{\dagger}$, $a_{i\sigma}$ are fermionic creation and annihilation operators; $n_{i\uparrow} = a_{i\uparrow}^{\dagger}a_{i\uparrow}$ and similarly for $n_{i\downarrow}$; the notation $\langle i, j\rangle$ in the first sum associates sites that are adjacent in an $n_x \times n_y$ rectangular lattice (grid); and $\sigma \in \{\uparrow, \downarrow\}$. The first term in Eq. (1) is called the hopping term, with t being the tunneling amplitude, and the second term is called the interaction or onsite term where U is the Coulomb potential. We will usually fix $t = 1$, $U = 2$ (similarly to Ref. [17]); see Appendix D 1 for results suggesting that the complexity of approximately finding the ground state of H is not substantially different for other U not too large and sufficiently bounded away from 0. We sometimes also consider what we call the noninteracting version of the Hubbard model, which only contains the hopping term.

On an $n_x \times n_y$ grid, the Hubbard Hamiltonian can be represented as a sparse square matrix with $2^{2n_xn_y}$ rows. Although the size of this matrix can be reduced by restricting to a subspace corresponding to a given occupation number and taking advantage of translation and spin invariance, the worst-case growth of the size of these subspaces is still exponential in $N = n_xn_y$. This exponential growth severely limits the capability of classical exact solvers to address this model. For example, Yamada *et al.* [18] report an exact solution of the Hubbard model with 17 fermions on 22 sites requiring over 7 TB of memory and 13 TFlops on a 512-node supercomputer. By contrast, a Hubbard model instance with N sites can be represented using a quantum computer with $2N$ qubits (each site can contain at most one spin-up and at most one spin-down fermion, so two qubits are required per site). This suggests that a quantum computer with around 50 qubits could already simulate instances of the Hubbard model going beyond classical capabilities.

Approximate classical techniques such as the quantum Monte Carlo and density matrix renormalization group methods can address larger grids (up to thousands of sites) than near-term quantum computers, but experience difficulties in certain coupling regimes and away from half-filling, leading to substantial uncertainties in physical quantities [15]. The hope is that quantum computing, while addressing smaller system sizes, could evade the difficulties experienced by these methods (such as the "sign problem" in quantum Monte Carlo methods) and enable access to these regimes.

Another approach to understanding the Hubbard model via a quantum device is analog quantum simulation [2,19]: engineering a special-purpose quantum system that implements the Hubbard Hamiltonian directly [20–22]. Analog quantum simulators are easier to implement experimentally than universal quantum computers and enable access to much larger systems than will be possible using near-term quantum computers. However, they are inherently less flexible than digital quantum simulation in terms of the Hamiltonians that can be implemented and the measurements that can be performed, and experience difficulties with reaching sufficiently low temperatures to demonstrate phenomena such as superconductivity [19,21,23].

Prior work on variational methods for solving the Hubbard model [17,24–26] (discussed in Sec. I) has left a number of important questions open which must be answered to understand whether it is a realistic target for near-term quantum computers. These include: What is the precise complexity of implementing the variational ansatz? How well will the optimization routines used handle statistical noise, and noise in the quantum circuit? How complex is the procedure required to produce the initial state?

Here we address all these questions and develop detailed resource estimates and circuit optimizations, as well as extensive numerical experiments for grids with up to 12 sites (24 qubits) to estimate how well realistic near-term quantum computers will be able to solve the Hubbard model. Although the Hubbard model is easily solvable directly by a classical algorithm for systems of this size, these experiments give insight into the likely performance of VQE on instances that are beyond this regime. Unlike some previous work, our focus is on solving instances just beyond the capability of classical hardware (e.g., size 10×10 or smaller) using machines with few (e.g., at most 200) physical qubits. In this regime, it is essential to carry out precise complexity calculations to understand the feasibility of the VQE approach.

A key ingredient in the complexity calculations for our circuits will be their depths. To compute this, we assume that the quantum computer can implement arbitrary two-qubit gates, and that one-qubit gates can be implemented at zero cost. These assumptions are not too unrealistic. Almost all the two-qubit gates we will need are rotations of the form $e^{i(\theta(XX+YY)+\gamma ZZ)}$ (up to single-qubit unitaries), which can be implemented natively on some superconducting qubit platforms, and one-qubit gates can be implemented at substantially lower cost in some architectures [27].

When simulating a VQE experiment on a classical computer, one can consider three different levels of realism:

(1) The simplest but least realistic level is to assume that we can perform exact energy measurements to learn $\langle\psi|H|\psi\rangle$, which can be used directly as input to a classical optimizer.

(2) The next level of realism is to simulate the result of energy measurements as if they were performed on a quantum computer but to assume that the quantum computer is perfect, i.e., does not experience any noise.

(3) Finally, one can simulate the effect of noise during the quantum computation.

In this paper, we consider all these levels. The main results we obtain can be summarized as follows:

(1) The most efficient approach we found for encoding fermions as qubits, for the small-sized grids we considered (indeed, for grids such that $\min\{n_x, n_y\} \leqslant 8$), was the Jordan-Wigner transform, both in terms of space and (perhaps surprisingly) in terms of circuit depth. See Appendix A for details.

(2) We develop an approach to efficiently implement a variant of the so-called Hamiltonian variational (HV) ansatz [17], and generalizations of this ansatz, in the Jordan-Wigner transform (Sec. I C). The circuit depth is as low as $2n_x + 1$ per ansatz layer on a fully connected architecture, and $6n_x + 1$ per layer on an architecture such as Google Sycamore [8]. See Table I for some examples. This method can also be used to implement the fermionic Fourier transform (FFT) more efficiently than previous work for small grid sizes.

(3) We introduce an efficient method of measuring the energy of a trial state produced using this ansatz (Sec. I D),

TABLE I. Example circuit depths per layer of the efficient ansätze for various architectures (for n_x even/odd).

Architecture	Ansatz circuit depth per layer
Fully connected	$\mathbf{2n_x + 1/2n_x + 2}$
	$4 \times 4 / 4 \times 5 : 9$
	$5 \times 5 / 5 \times 6 : 12$
	$6 \times 6 : 13$
Nearest neighbor	$\mathbf{4n_x/4n_x + 1}$
	$4 \times 4 / 4 \times 5 : 16$
	$5 \times 5 / 5 \times 6 : 21$
	$6 \times 6 : 24$
Google Sycamore	$\mathbf{6n_x + 1/6n_x + 2}$
	$4 \times 4 / 4 \times 5 : 25$
	$5 \times 5 / 5 \times 6 : 32$
	$6 \times 6 : 37$

which requires only five computational basis measurements and allows for a simple notion of error detection.

(4) In numerical experiments with simulated exact energy measurements and using the Limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) optimizer, the error with the true ground state (measured either by fidelity or energy error) decreases exponentially with the circuit depth in layers (Fig. 8). This gives good evidence that the efficient HV ansatz is able to represent the ground state of the Hubbard model efficiently, at least for the small grid sizes accessible to near-term hardware.

(5) For all grids with at most 12 sites, 0.99 fidelity to the ground state (which is nondegenerate in all the cases we consider) can be achieved using an efficient HV ansatz circuit with at most 18 layers (Fig. 7). The results are consistent with a grid with N sites needing $O(N)$ layers; in all cases, we found that at most $1.5N$ layers were needed. We present a generalization of the HV ansatz called the number preserving (NP) ansatz (Sec. I B), which gives more freedom in the choice of gates. This generally performs better in terms of the depth required to achieve high fidelity with the ground state but requires more optimization steps.

(6) In numerical experiments with simulated realistic energy measurements on systems with up to nine sites, the coordinate descent (CD) [28–30] and simultaneous perturbation stochastic approximation (SPSA) [31–33] algorithms are both able to achieve high fidelity with the ground state (e.g., SPSA achieves fidelity >0.977 for a 3×3 grid; see Table III and Fig. 9) by making a number of measurements which would require a few hours[1] of execution time on a real quantum computer. On 2×2 and 2×3 grids, the two algorithms achieved similar final fidelities, while on 1×6 and 3×3 grids SPSA performed substantially better.

(7) In numerical experiments with simulated depolarizing noise in the quantum circuit for systems with up to six sites, error rates of up to 10^{-3} do not have a significant effect on the fidelity of the solution (Table IV). The use of error detection gives a small but noticeable improvement to the fidelity (Fig. 10).

We conclude that variational methods show significant promise for producing the ground state of the Hubbard model for grid sizes somewhat beyond what is accessible with classical computational methods. Highly optimized ansatz circuits can be designed; the depth required for these circuits to find the ground state seems to scale favorably with the size of grid; and the use of realistic measurements and noise in the circuit do not reduce final fidelities unreasonably.

Based on these results, it seems plausible that an instance of the Hubbard model larger than the capacity of classical exact diagonalization methods could be solved by optimizing over quantum circuits with depth 300–500 (on a fully connected architecture). This is substantially smaller than previous estimates for other proposed applications of near-term quantum computers, albeit beyond the capacity of leading hardware available today. Although exact diagonalization provides more information than producing the ground state on a quantum computer, physically important quantities (such as correlation functions) are nevertheless accessible. This suggests that variational quantum algorithms could become an important tool for the study of the Hubbard model.

I. THE VARIATIONAL METHOD

Our work fits within the standard VQE framework [10,11]. The field of variational quantum algorithms is already too vast to sensibly summarize here. The VQE algorithm has been implemented experimentally in a number of platforms including photonics [10], superconducting qubits [32–34], and trapped ions [35–38], while there have also been numerous theoretical developments [17,26,39–45].

A number of works have applied VQE to the Hubbard model specifically. Wecker *et al.* [17] developed the HV ansatz, which will be a key tool that we will use and expand upon (see Sec. I B). They tested it for the half-filled Hubbard model for systems of up to 12 sites—in the case of simulated exact energy measurements, they used ladders with dimensions $n_x \times 2$ for $n_x = 2, \dots, 6$; in the case of realistic energy measurements, they tested a system of size 4×2. Implementation of two layers of this ansatz for a 4×2 system would require 1000 gates, according to their estimate (we reduce this estimate substantially; see Sec. B). Dallaire-Demers *et al.* [26] also developed a low-depth circuit ansatz inspired by the unitary coupled cluster ansatz and applied it to the 2×2 Hubbard model.

Reiner *et al.* [24] recently studied how gate errors affect the HV ansatz. They considered a model where gates are subject to fixed unitary over-rotation errors and found that for small system sizes (grids of size 2×2, 3×2 and 3×3), reasonably small errors did not prevent the variational algorithm from finding a high-quality solution. Verdon *et al.* [25] developed an approach to optimizing VQE parameters using recurrent neural networks, and applied it to Hubbard model instances of size 2×2, 3×2, and 4×2. Wilson *et al.* [46] designed a somewhat related "meta-learning" approach to VQE which they tested on the *spinless* Hubbard model on three sites.

We also remark that several endeavours (e.g., Refs. [6,47–49]) have studied the complexity of quantum algorithms for

[1]~57M circuit evaluations; Google's Sycamore processor can perform 1M circuit evaluations in 200 s [8].

simulating time-evolution or thermodynamic properties of the Hubbard model.

The VQE framework requires a few different ingredients to be specified:

(1) The encoding used to represent fermions as qubits

(2) The properties of variational ansätze (circuit family, initial state, etc.)

(3) Implementation of energy measurements

(4) Selection of classical optimizer

Additionally, there are some important implementation details to be determined for the resulting quantum circuits to be executed in a real-world architecture. In the remainder of this section, we describe the approach we took to fill in all these details.

A. Fermionic encoding

We use the well-known Jordan-Wigner (JW) encoding of the fermionic Hamiltonian H as a qubit Hamiltonian. This encoding has no overhead in qubit count, as each site maps to two qubits. The downside is that some fermionic interactions map to long strings of Pauli operators, whose length increases with the grid size. We will need to implement time evolution according to the hopping terms in H; this also has complexity that increases with the grid size.

There are other encodings (such as the Bravyi-Kitaev superfast encoding [50] and Ball-Verstraete-Cirac encoding [51,52]) which produce local operators at the expense of using additional qubits. However, for small grid sizes, the complexity of the corresponding quantum circuits for time evolution seems to be higher than optimized methods that use fermionic swap networks to implement the required time-evolution operations under the Jordan-Wigner transform. See Appendix A for a discussion.

The Jordan-Wigner encoding associates each fermionic mode (corresponding to a site on a grid and a choice of spin) with a qubit. The encoding can be seen as assigning a position on a line to each fermionic mode. We use the so-called snake-shaped configuration shown in Fig. 1, which illustrates a setting where the qubits are laid out according to the Google Sycamore architecture[2] [8]. The advantage of using this configuration is that we can make use of fermionic swap networks for efficiently implementing the ansatz circuits (see Sec. I C) and carry out Hamiltonian measurements using the lowest number of circuit preparations (see Sec. I D).

Each hopping term between qubits i and j $(i < j)$ maps to a qubit operator via

$$a_i^\dagger a_j + a_j^\dagger a_i \mapsto \tfrac{1}{2}(X_i X_j + Y_i Y_j) Z_{i+1} \cdots Z_{j-1}.$$

For $j = i + 1$ (a hopping term between horizontally adjacent qubits), there is only the bare hopping term $\frac{1}{2}(X_i X_j + Y_i Y_j)$. For vertically adjacent qubits, the bare hopping term is accompanied by the string of Z operators $Z_{i+1} \cdots Z_{j-1}$. Each on-site term acting on qubits i and j maps to a qubit operator via

$$a_i^\dagger a_i a_j^\dagger a_j \mapsto \tfrac{1}{4}(I - Z_i)(I - Z_j),$$

[2]That is, a natural generalization of the qubit topology reported in Ref. [8] to larger system sizes.

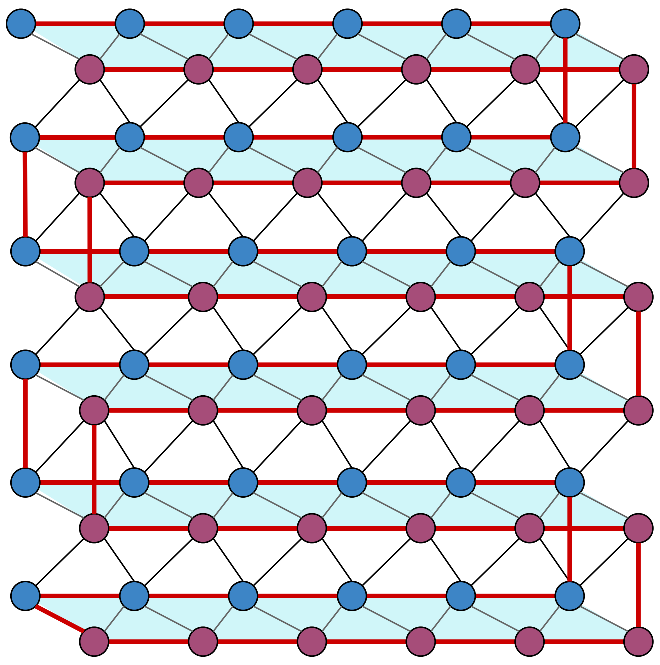

FIG. 1. An illustration of how fermionic modes can be mapped to physical qubits on a physical architecture such as Google's Sycamore device [8]. The fermionic modes (blue: spin-up; red: spin-down) on a 6×6 lattice are mapped to qubits in an array of size $2 \times 6 \times 6$. The red line represents the order associated with the JW encoding of the qubits, which moves from the top left toward the right. The blue panels are added to aid visualization. Note that the red line does not follow the true connectivity of the qubits (the thin black lines), and hence any local operator with respect to the JW encoding is not necessarily local with respect to the physical connectivity of the qubits, and vice versa.

whether or not qubits i and j are adjacent in the Jordan-Wigner encoding. Hence, as we will see, the vertical hopping terms are the most difficult of these three types of terms to implement efficiently.

B. Variational ansätze

Various variational ansätze have been proposed for use within the VQE framework, including the HV ansatz [17], hardware-efficient ansätze [32], unitary coupled clusters [10,36], and others.

The HV ansatz is based on intuition from the quantum adiabatic theorem, which states that one can evolve from the ground state of a Hamiltonian H_A to the ground state of another Hamiltonian H_B by applying a sequence of evolutions of the form e^{-itH_A}, e^{-itH_B} for sufficiently small t. In the case of the Hubbard model, we start in the ground state of the noninteracting Hubbard Hamiltonian $(U = 0)$ for a given occupation number, which can be prepared efficiently [53,54], and then evolve to the ground state of the full Hubbard model, including the on-site terms.

Rather than alternating evolutions according to the full hopping and on-site terms in the Hamiltonian H in Eq. (1), it is natural to split H into parts that consist of terms that are sums of commuting components, which could allow for more efficient time evolution. This also allows for these terms

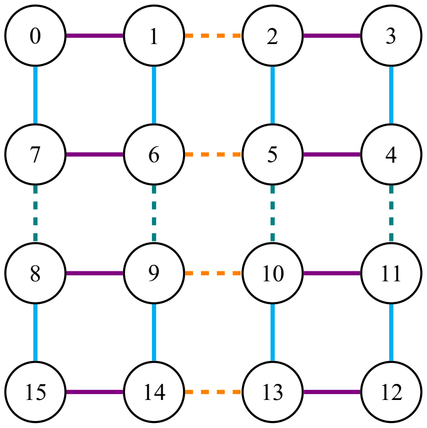

FIG. 2. The four sets of hopping terms (for a fixed spin). Hopping terms of the same color commute, and hence in principle can be implemented simultaneously. Purple corresponds to the horizontal terms H_1, dashed orange to H_2, blue to the vertical terms V_1, and dashed green to V_2.

to have different coefficients, while still respecting overall symmetries of the Hamiltonian. Then a layer of the HV ansatz is a unitary operator of the form

$$e^{it_{V_2}H_{V_2}}e^{it_{H_2}H_{H_2}}e^{it_{V_1}H_{V_1}}e^{it_{H_1}H_{H_1}}e^{it_{H_O}H_O}, \qquad (2)$$

where H_O is the on-site term, H_{V_1} and H_{V_2} are the vertical hopping terms, H_{H_1} and H_{H_2} are the horizontal hopping terms as shown in Fig. 2. Different layers can have different parameters. Note that there is some freedom in the order with which we can implement these terms, and also that some of them may not be needed depending on the grid dimensions. The vertical hopping terms are nontrivial to implement efficiently in the JW transform, given the potentially long strings of Z operators associated with each of them. We remark that a similar technique of decomposition into commuting parts is common in quantum Monte Carlo methods, where it is known as the checkerboard decomposition.

The HV ansatz has been shown to be effective for small Hubbard model instances [17,24] and involves a small number of variational parameters: at most five per layer. One disadvantage of this ansatz is that preparing the initial state is a nontrivial task. It can be produced using the (2D) FFT, for which efficient algorithms are known [53,54], or via a direct method based on the use of Givens rotations [54]. We calculated the complexity of an asymptotically fast algorithm for the FFT presented in Ref. [54] and also developed an alternative implementation strategy using fermionic swap networks, which may be of independent interest. We found that for grids of size up to 20×20, neither of these strategies was more efficient than direct preparation of the initial state using Givens rotations [54], which has circuit depth $n_x n_y - 1$ (assuming an arbitrary circuit topology). See Appendix E for the details.

To avoid this depth overhead for constructing the initial state, we also considered an ansatz which is a generalization of HV. This ansatz benefits from the same theoretical guarantees that arbitrary-length circuits can find the ground state of H while being more general and allowing for an initial state that is significantly more straightforward to generate. However, the trade-off is that it uses more parameters, making the optimization process more challenging.

The ansatz, which we call the NP ansatz, is derived from HV by replacing all hopping and on-site terms with a more general NP operator[3] parameterized by two angles θ and ϕ, and implemented by the two-qubit unitary

$$U_{\mathrm{NP}}(\theta,\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & i\sin\theta & 0 \\ 0 & i\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & e^{i\phi} \end{pmatrix}.$$

The noninteracting ground state can still be used as the initial state, although computational basis states (where the Hamming weight is equal to the fermonic occupation number of interest) can also be used with some success (see Appendix C 1). Then one layer of the ansatz consists of applying a $U_{\mathrm{NP}}(\theta,\phi)$ gate (with varying angles θ, ϕ) across each pair of qubits that correspond to fermionic modes that interact according to the Hubbard Hamiltonian H in Eq. (1). That is, we apply $U_{\mathrm{NP}}(\theta,\phi)$ gates for all pairs of modes (i,σ), (j,τ) such that either $i \sim j$ and $\sigma = \tau$ (hopping terms) or $i = j$ and $\sigma \neq \tau$ (onsite terms). As before, different layers can have different parameters.

For an $n_x \times n_y$ grid, one layer of the NP ansatz requires

$$2(2(n_x(n_y-1)+n_y(n_x-1))+n_x n_y) = 10n_x n_y - 4n_x - 4n_y$$

parameters. The HV ansatz is the special case of the NP ansatz that also preserves spin and where many parameters are fixed to be identical or 0.

C. Efficient implementation of HV and NP ansätze

Hopping terms between vertically adjacent qubits that are not local with respect to the JW encoding must be accompanied by a string of Z operators (see Sec. I A), which can be costly to implement. To reduce the overhead associated with these vertical hopping terms, we use a technique of Kivlichan *et al.* [55] based on networks of fermionic SWAP gates, though with some minor changes for efficiency. In particular, we remove some unnecessary vertical fermionic SWAP gates and instead only swap horizontally adjacent qubits. This means that, for an $n \times n$ grid, only n repetitions of a column-permuting subroutine (which itself has depth 2) are necessary to be able to implement all vertical hopping terms locally, in comparison to the $\frac{3}{\sqrt{2}}n$ iterations that are deemed to be necessary in Ref. [55]. We now describe this approach; Appendix E gives a comparison to the approach of Ref. [55], in the closely analogous context of implementing the FFT. In what follows, we write JW-adjacent to mean adjacent with respect to the JW encoding, and when we say that an operator is implemented locally, we mean that the two qubits that it acts on are JW adjacent.

[3]This is similar to the exchange-type entangling gates discussed in Refs. [33,44]; an alternative notion of number-preserving VQE ansatz was studied in Ref. [42].

We use fermionic SWAP (FSWAP) gates to move qubits that were originally not JW adjacent into JW-adjacent positions. The FSWAP gate acts as a SWAP gate for fermions, and corresponds to the unitary operator

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}.$$

This allows vertical hopping interactions to be implemented locally, whilst maintaining the correct parity on all qubits. That is, we repeatedly apply the operator $U_R U_L$, where U_L swaps odd-numbered columns with those to their right, and U_R swaps even-numbered columns with those to their right. After each application of $U_R U_L$, a new set of qubits that were previously not vertically JW-adjacent are made JW adjacent, meaning that the vertical hopping interaction between them can be implemented locally using a single NP operator, without Z strings. For an $n_x \times n_y$ grid, it suffices to apply $U_R U_L$ a total of n_x times to allow all vertical interactions to be implemented locally and return the qubits to their original positions.

Note that the vertical terms are implemented in a different order to the horizontal terms. If the columns begin in the order $1, 2, 3, 4 \ldots, n$ (assuming that n is even), then after a single application of $U_R U_L$, they are re-ordered to $2, 4, 1, 6, \ldots, n-3, n-1$. Each subsequent application of $U_R U_L$ will place a new even-numbered column to the far left, and a new odd-numbered column to the far right, until $n/2$ applications have seen every even-numbered column at the left, and every odd-numbered column at the right. Since it is at the far ends that vertical terms can be applied locally, then after $n/2$ applications of $U_R U_L$, all terms that can be applied locally at the left will have been applied for the even-numbered columns, and similarly for the odd-numbered columns. Applying another $n/2$ iterations of $U_R U_L$ will see all even-numbered columns move to the right and all odd-numbered columns to the left, which allows the remaining terms to be implemented locally. Figure 3 illustrates the order in which the vertical hopping terms are implemented for a 4×4 grid of fermions (ignoring spin).

If we assume that gates can be applied across arbitrary pairs of qubits and that both FSWAP and U_{NP}[4] can be implemented in depth 1, then the circuit used to implement all vertical hopping terms will have depth $2n_x$ for even n_x, and depth $2n_x + 1$ for odd n_x. This is because for even n_x the hopping terms can be implemented in parallel with U_R and for odd n_x some hopping terms can be implemented in parallel with U_L and others with U_R; one hopping term is left over in the latter case, leading to an overall overhead of 1. All horizontal hopping terms can be implemented in depth 2, and all on-site terms in depth 1. In fact, it is possible to perform a combined horizontal hopping term and FSWAP operation in depth 1.[5]

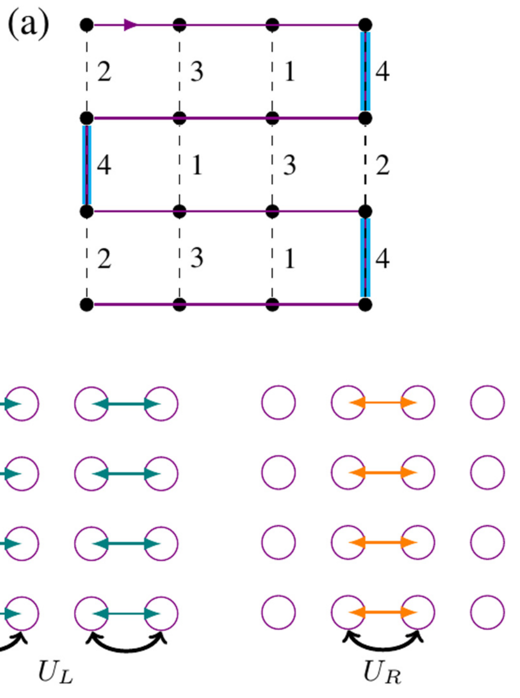

FIG. 3. (a) Vertical hopping term implementation for a 4×4 grid of fermions. The numbers i show which vertical term will be implemented after i applications of $U_R U_L$. The highlighted blue lines show the only places where the hopping terms can be implemented – at the JW-adjacent positions. (b) Action of U_L and U_R on the grid of qubits.

By replacing the first and last layers of FSWAP gates (the first corresponding to U_L, and the last corresponding to U_R) with such a combined operation, we effectively fold the horizontal hopping terms into the swap network operator $U_R U_L$. Hence, the final depth of the circuit that implements one layer of the ansatz is $2n_x + 1$ for even n_x, and $2n_x + 2$ for odd n_x. Figure 4 shows the circuit used to implement a single layer of the ansatz for a 4×4 grid, for just one of the spins (and therefore omitting the on-site interactions).

We stress that this efficient version of the HV ansatz is different from the standard HV ansatz, in that vertical hopping terms are implemented in a different order. We refer to it as the efficient HV (EHV) ansatz below.

It is worth comparing the complexity of the EHV ansatz to what we would obtain by implementing time evolution according to each term in the HV ansatz directly. Considering a $2 \times n_y$ grid (the first case where the two ansätze differ), using the snake ordering, horizontal hopping terms and on-site interactions can each be implemented in depth 1. Vertical interactions either can be implemented in depth 1, or require an operation of the form $e^{i\theta(XX+YY)ZZ}$ to be implemented. As discussed in Appendix A 1, this can be achieved with a circuit of depth 4, assuming that arbitrary 2-qubit gates are available. Therefore, the overall depth of the circuit for each layer is $2 + 2 \times (1 + 4) = 12$, which is more than twice as large. For grids where n_x is larger, the improvement will be even more pronounced. As another comparison, Reiner *et al.* [24] reported a circuit with 81 two-qubit gates per layer for a 3×3 grid, whereas the circuit here would

[4]The hopping terms in the HV ansatz $e^{i\theta(XX+YY)/2}$ are a special case of U_{NP} where $\phi = 0$.

[5]Up to single qubit gates: FSWAP $\cdot U_{\mathrm{NP}}(\theta, \phi) = (Z^{3/2} \otimes Z^{3/2}) \cdot U_{\mathrm{NP}}(\theta + \frac{\pi}{2}, \phi)$.

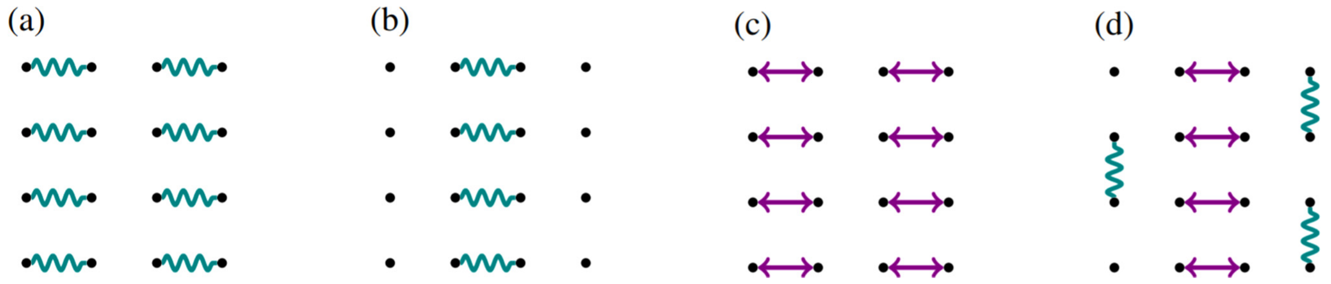

FIG. 4. Quantum circuit elements required to implement one layer of the EHV or NP ansatz for a single spin-type. Circuit layers go from (a) to (d), with (c) and (d) repeated thrice more to complete the swap network. Wavy green lines are the number-preserving unitaries U_{NP}. Purple arrows are FSWAP gates, with (c) representing U_L and (d) representing U_R implemented in parallel with vertical hopping terms. In our implementation, the (b) layer is moved to the end, allowing the horizontal hopping terms in (a) and (b) to be combined with the FSWAP gates in (c) and (d), respectively.

use at most $9 + 2 \times 3 \times (6 + 2) = 57$ two-qubit gates per layer.

Finally, we remark that all this discussion has assumed the use of open boundary conditions in the Hubbard model. Periodic boundary conditions in the horizontal direction can be implemented without any overhead but periodic boundaries in the vertical direction are significantly more challenging. However, smooth boundary conditions, which can be even more advantageous in terms of reducing finite-size effects [56], can also be implemented efficiently.

D. Measurement

At the end of each run of the circuit, we need to measure the energy of the state $|\psi\rangle$ produced with respect to H. (Note that, unlike quantum Monte Carlo methods, there is no issue with correlation between runs and each measurement is assumed to be independent.) The most naïve method to achieve this would involve measuring $\langle\psi|H_i|\psi\rangle$ for each term H_i in H. For an $n_x \times n_y$ grid, there are $4n_xn_y - 2n_x - 2n_y$ hopping terms and n_xn_y on-site terms, giving $5n_xn_y - 2n_x - 2n_y$ terms in total, which can be a significant overhead (e.g., 156 terms for $n_x = n_y = 6$). Even worse, these terms involve long-range interactions via the Jordan-Wigner transform, suggesting that energy measurement can be challenging.

However, it turns out that many of these terms can be measured in parallel by grouping them together into at most five commuting sets. There have been a number of recent works on general techniques for splitting the terms of a local Hamiltonian into commuting sets [57–61]; here we have a particularly efficient way to do this using the lattice structure of the Hamiltonian. The on-site terms can be measured all at once and the hopping terms can be broken into at most four sets—two horizontal and two vertical—as displayed in Fig. 2.

First, the on-site terms can simply be measured by carrying out a computational basis measurement on every qubit. In the Jordan-Wigner picture, the on-site terms map to a matrix of the form $\frac{1}{4}(I - Z_i)(I - Z_j) = |11\rangle\langle 11|_{ij}$. So, the energy for each term corresponding to a particular site is the probability that the two qubits corresponding to this site (spin up and spin down) are both measured to be in state 1.

Horizontal hopping terms take the form $\frac{1}{2}(X_iX_{i+1} + Y_iY_{i+1})$. These terms can be measured efficiently by first transforming into a basis in which this operator is diagonal. This can be done with the quantum circuit U shown in Fig. 5, which diagonalizes $\frac{1}{2}(XX + YY)$ as $D = |01\rangle\langle 01| - |10\rangle\langle 10|$ and so the expectation of $\frac{1}{2}(XX + YY)$ is equivalent to the probability of getting the outcome 01 minus the probability of getting 10. It is important to note that we cannot measure the hopping term on qubit pairs $(i - 1, i)$ and $(i, i + 1)$ simultaneously due to this transformation, and so if $n_x > 2$ we require two preparations of the ansatz circuit to measure the horizontal hopping terms.

The vertical terms can be measured in a similar way, but with the added complication of the Pauli-Z strings $\frac{1}{2}(X_iX_j + Y_iY_j)Z_{i+1}\cdots Z_{j-1}$. Qubits i and j are treated like the horizontal hopping terms and the Z strings are dealt with by multiplying the expectation by a parity term. Doing a computational basis measurement on qubits $i + 1$ to $j - 1$ and counting the number of times that 1 is measured gives the parity term. If there are an even number, the parity is 1, otherwise it is -1.

All the vertical hopping terms can also be measured with at most two executions of the ansatz circuit. For example, consider the 4×4 grid shown in Fig. 2. For the first set of vertical hopping terms (shown in blue), we can apply U to all eight pairs of qubits corresponding to these terms simultaneously (pairs $(0, 7), (1, 6), \ldots, (11, 12)$). Since U has the property that $U^\dagger(Z \otimes Z)U = Z \otimes Z$, we can then collect statistics for measuring D on each pair of qubits and all the required Z-strings (e.g., $Z_1 \ldots Z_6$) simultaneously. This is a consequence of the chosen Jordan-Wigner ordering—there are always an even number of Pauli-Z operators in between qubits i and j.

Note that, in our scheme, measurement is the one point in the circuit where quantum gates need to be applied across qubits that are not adjacent with respect to the JW encoding. We also remark that this approach allows a simple notion

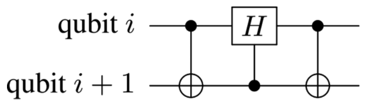

FIG. 5. Unitary U that transforms into the $\frac{1}{2}(XX + YY)$ basis.

of error detection by checking the Hamming weight of the returned measurement results (see Sec. I F).

Recently, Cai [62] described an alternative approach to obtaining the expectation value using five measurements, based on switching the Jordan-Wigner ordering around when measuring the vertical terms, making the vertical hopping terms the JW-adjacent ones and hence removing the Pauli-Z strings. The cost of implementing this approach would be similar to the approach proposed here in the case of square grids (or perhaps slightly more efficient). For nonsquare grids, the approach proposed here will be more efficient, as one can choose the orientation of the grid to minimise the length of Jordan-Wigner strings, whereas the approach of Ref. [62] needs to run the quantum circuit twice, one for each orientation.

E. Classical optimizer

The VQE algorithm makes many calls to the quantum computer to produce trial quantum states. First, we will lay out some of the terms that will be important in our analysis:

(1) Circuit evaluation = one run of the quantum computer

(2) Energy measurement = five circuit evaluations (see Sec. I D)

(3) Energy estimate = m energy measurements (also referred to as *function evaluation* in the context of optimization routines)

We can determine a rough budget for a reasonable number of calls as follows. We start by assuming that we can perform each two-qubit quantum gate in 100 ns and that measurements are instantaneous (to justify this, even faster gates than this have been demonstrated in superconducting qubit systems [27], and measurements have been demonstrated that are fast enough that their cost is negligible over the whole circuit [63]). Assume for simplicity that the depth of the whole circuit is 100 and that the cost of classical computation is negligible. Then 10^5 runs of the quantum computer can be executed per second. If we would like to ultimately estimate the energy up to an accuracy of $\sim 10^{-2}$, approximately 10^4 circuit evaluations are required to estimate each of the five terms (see Fig. 19 in the Appendix for numerical results to justify this assumption, where, for a particular instance, we found that m measurements achieved energy error $\approx 1.3/\sqrt{m}$). Thus approximately two energy estimates up to this precision can be obtained per second. So in 5×10^4 s, corresponding to approximately 14 h, we can produce approximately 10^5 energy estimates up to an accuracy of $\sim 10^{-2}$. This motivates us to use $\sim 10^5$ as the budget for the number of function evaluations used by the optimizer. (In fact, in our numerical experiments below, we found that substantially fewer evaluations were sufficient.)

We evaluated different optimization methods given in the NLopt C library for nonlinear optimization [64] and found that L-BFGS was usually a very effective algorithm to use when considering a perfect, noiseless, version of VQE with simulated exact energy measurements. Other algorithms required many more iterations or often found lower-quality local minima. To estimate the gradient, as required for L-BFGS, we used a simple finite difference approximation.

Including realistic measurements turns the optimization problem into a stochastic one. In this setting, we found that standard deterministic optimization methods provided by NLopt were ineffective (either failing completely or producing low-quality results). We therefore turned to stochastic optimization methods such as the simultaneous perturbation stochastic approximation (SPSA) algorithm [31], which has been successfully used in VQE experiments on superconducting hardware [32,33] and a CD algorithm [28–30] that has been shown to be effective for small VQE instances. We remark that, during preparation of this paper, alternative stochastic optimization techniques for VQE have been developed [25,46,65]; evaluating and improving such techniques in the context of the Hubbard model is an important direction for future work.

1. Simultaneous perturbation stochastic approximation

The SPSA algorithm [31] works in a similar way to the standard gradient descent algorithm, but rather than estimating the full gradient, instead picks a random direction to estimate the gradient along. This is intended to make SPSA robust against noise and to require fewer function evaluations. Many aspects of this algorithm can be tailored to the specific problem at hand, such as parameters that govern the rate of convergence, terminating tolerances and variables on which the tolerance is monitored, and the number of gradient evaluations to average the estimated gradient over.

Each gradient evaluation is estimated from two function evaluations (as compared with typically twice the number of parameters for finite difference methods) and is given by

$$g(\boldsymbol{\theta}_k) = \frac{f(\boldsymbol{\theta}_k + c_k\boldsymbol{\Delta}_k) - f(\boldsymbol{\theta}_k - c_k\boldsymbol{\Delta}_k)}{2c_k}\boldsymbol{\Delta}_k^{-1},$$

where $\boldsymbol{\theta}_k$ is the current parameter vector after k steps, c_k is an optimization parameter to be determined, and the parameters are perturbed with respect to a Bernoulli ± 1 distribution $\boldsymbol{\Delta}_k$ with probability $\frac{1}{2}$ for each outcome. The gradient step size is $c_k = c/(k+1)^{\gamma}$, where in our experiments $\gamma = 0.101$ was chosen to be the ideal theoretical value [66] and $c = 0.2$. The parameters are then updated via

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - a_k g(\boldsymbol{\theta}_k),$$

where $a_k = a/(k+1+A)^{\alpha}$ dictates the speed of convergence. Similarly to γ, $\alpha = 0.602$ is chosen as the ideal theoretical value [66], while we set the stability constant $A = 100$ and $a = 0.15$. The values of a and c were chosen by a joint parameter sweep. We found that the parameters generally had to be small to reduce the rate of convergence, which allowed us to reach a more accurate result but with more iterations.

The main modification we made to the standard SPSA algorithm is to perform multiple runs of the optimizer. We start with two coarse runs with a high level of statistical noise where we calculate the energy estimate using only 10^2 and then 10^3 energy measurements. This is followed by a finer run where SPSA is restarted using 10^4 energy measurements for the estimate and averaging over two gradient evaluations in random directions for $g(.)$. The number of steps in this three-stage optimization is determined by a ratio of 10 : 3 : 1. Figure 6 shows the beneficial effect of starting by making less accurate measurements, as described.

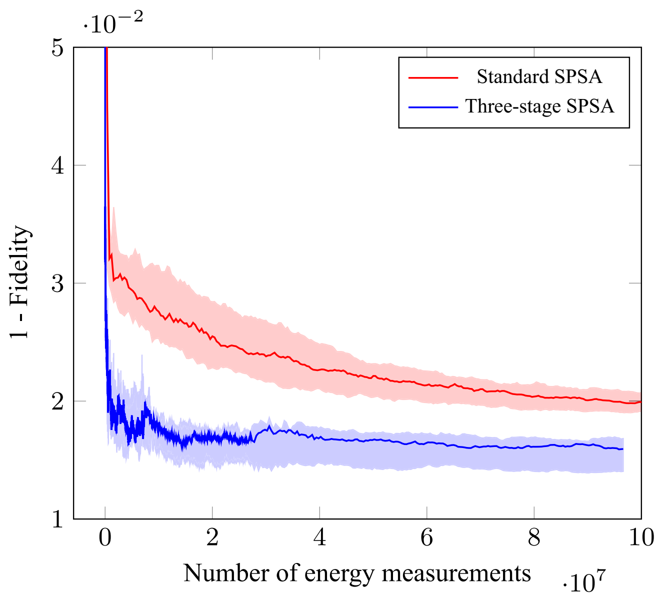

FIG. 6. Infidelity achieved over five runs of the standard SPSA algorithm (where each energy estimate is formed of 10^4 energy measurements and two gradient evaluations are taken in each iteration) and a modified three-stage SPSA algorithm which starts with less accurate measurements, as described in the text. Results are shown for a 1×6 grid, EHV ansatz, depth 5. The solid lines show the median of the runs and the limits of the shaded regions are the maximum and minimum values seen over the five runs.

2. *Coordinate descent algorithm*

We now describe an alternative algorithm based on an approach independently discovered by Refs. [28–30]. The basic algorithm presented in these works can be applied to variational ansätze where the gates are of the form $e^{i\theta H}$ for Hamiltonians H such that $H^2 = I$ (e.g., Pauli matrices). It is based on the nice observation that, for gates of this form, the energy of the corresponding output state is a simple trigonometric polynomial in θ (if all other variational parameters are fixed). This implies that it is sufficient to evaluate the energy at a small number (three) of choices for θ to *analytically* determine its minimum with respect to θ. The algorithm proceeds by choosing parameters in some order (e.g., a simple cyclic sequential order, or randomly) and minimizing with respect to each parameter in turn. It is shown in Refs. [28–30] that this approach can be very effective for small VQE instances.

We use a generalization of this approach which works for any Hamiltonian with integer eigenvalues. This enables us to apply the algorithm to the NP (and hence HV) ansatz, because each gate in the ansatz can be seen as combining the pair of gates $e^{i\theta(XX+YY)/2}$, $e^{i\phi|11\rangle\langle 11|}$. The corresponding Hamiltonians have eigenvalues $\{0, \pm 1\}$, $\{0, 1\}$, respectively. The generalization is effectively the same as the one presented in Refs. [28,30] to optimize over separate gates which share the same parameters. However, here we present the algorithm and its proof somewhat differently and include a full argument for how to compute the minimum with respect to θ, which is not included in Refs. [28,30].

The algorithmic approach has been given different names in the literature (sequential minimal optimization [28], Rotosolve [29], Jacobi diagonalization [30]). Here we prefer yet another name, coordinate descent [67] (CD), because this encompasses the approach we consider, whereas the above names technically refer to special cases of the approach which are not directly relevant to the algorithm we use.[6]

Let A be a Hermitian matrix with eigenvalues $\lambda_k \in \mathbb{Z}$, and assume that $e^{i\theta A}$ is one of the gates (parametrised by θ) in a variational ansatz. Then the energy of the output state with respect to H can be written as

$$\operatorname{tr}[HUe^{i\theta A}|\psi\rangle\langle\psi|e^{-i\theta A}U^\dagger]$$

for some state $|\psi\rangle$ and unitary operator U that do not depend on θ. Writing $A = \sum_k \lambda_k P_k$ for some orthogonal projectors P_k and using linearity of the trace, this expands to

$$\sum_{k,l} e^{i\theta(\lambda_k - \lambda_l)} \operatorname{tr}[HUP_k|\psi\rangle\langle\psi|P_l U^\dagger].$$

If Δ denotes the set of possible differences $\lambda_k - \lambda_l$, and $D = \max_{k,l} |\lambda_k - \lambda_l|$, this expression can be rewritten as

$$f(\theta) = \sum_{\delta\in\Delta} c_\delta e^{i\theta\delta}$$

for some coefficients $c_\delta \in \mathbb{C}$. This is a (complex) trigonometric polynomial in θ of degree D. So, it can be determined completely by evaluating it at $2D + 1$ points. A particularly elegant choice for these is $\theta \in \{2k\pi/(2D+1) : -D \leqslant k \leqslant D\}$. Then the coefficients c_k can be determined via the discrete Fourier transform:

$$c_k = \frac{1}{2D+1} \sum_{l=-D}^{D} e^{-2\pi i k l/(2D+1)} f(2\pi l/(2D+1)).$$

To minimize f, we start by computing the derivative

$$\frac{df}{d\theta} = i \sum_{k=-D}^{D} k c_k e^{ik\theta},$$

and finding the roots of this function. To find these roots, we consider the function $g(\theta) = e^{2iD\theta}\frac{df}{d\theta}$. Every root of $\frac{df}{d\theta}$ is a root of $g(\theta)$, and as $g(\theta)$ is a polynomial of degree $2D$ in $e^{i\theta}$, its roots can be determined efficiently (e.g., by computing the eigenvalues of the companion matrix of g).

Finally, we ignore all roots that do not have modulus 1 (i.e., consider only roots of the form $e^{i\theta}$) and choose the root $e^{i\theta_{\min}}$ at which $f(\theta_{\min})$ is smallest. Note that the only steps throughout this algorithm which require evaluation of $f(\theta)$ using the quantum hardware are the $2D + 1$ evaluations required for polynomial interpolation.

The above argument extends to the situation where we have m Hamiltonian evolution operations in the circuit that all depend on the same parameter θ; in this case, one obtains a trigonometric polynomial of degree mD (see Refs. [28,30] for a proof), which is determined by its values at $2mD + 1$

[6]Sometimes the term coordinate descent is used for algorithms that perform gradient descent in each coordinate; we stress that here we instead *exactly* minimize over each coordinate.

points. This enables us to apply this optimization algorithm to the (efficient) HV ansatz as well.

F. Handling noise

The VQE approach needs to contend with two different kinds of noise: statistical noise inherent to the quantum measurement process and errors in the circuit. Statistical noise can be mitigated by simply taking more measurements, while the ansätze we use allow for a simple notion of error-detection with no overhead in terms of number of qubits or execution time. The NP (and hence HV) ansatz corresponds to quantum circuits where every operation in the circuit preserves fermionic occupation number (equivalently, Hamming weight after the Jordan-Wigner transform). So, if the final state of the quantum algorithm contains support on computational basis states of different Hamming weight to the start of the algorithm, one can be confident that an error has occurred.

Further, the Hamming weight of the final state can be measured as part of the measurement procedure described in Sec. I D without any additional cost. On-site energy measurements simply correspond to measurements in the computational basis, while measurements corresponding to hopping terms split pairs of qubits according to the pair's total Hamming weight. So, Hamming weights of pairs (and hence the total Hamming weight) can be determined simultaneously with measuring according to hopping terms.

II. NUMERICAL VALIDATION

We developed a high-performance software tool in C++, based on the Quantum Exact Simulation Toolkit [68] (QuEST), which enabled the ansätze we used to be validated and compared. The tests were mainly carried out on the Google Cloud Platform. In the preliminary tests, we found that GPU-accelerated QuEST commonly outperformed QuEST running on CPU only (whether single-threaded, multithreaded, or distributed). For most of the results reported here, we found a speedup of 4–5x when compared with a 16 vCPU machine (n1-highcpu-16) available on Google Cloud, which is similar to the speed-up reported in Ref. [68]. The GPU-accelerated tests were carried out using a single vCPU machine (n1-standard-1) equipped with either NVIDIA Tesla P4 (nvidia-tesla-p4) or NVIDIA Tesla K80 (nvidia-tesla-k80). Some of the noisy experiments were carried out on a single vCPU instance (n1-standard-1), as for some of the smaller grid sizes it was found that a single CPU performs similarly to a GPU-accelerated version (for small grid sizes, the data transfer between CPU and GPU dominates the run time).

We carried out the following tests. First, we tested the expressivity of the HV, EHV, and NP ansätze by simulating the VQE algorithm using these ansätze with (unrealistic) exact energy measurements and increasing circuit depths and grid sizes. This builds confidence that the variational approach will be effective for grid sizes beyond those that can be simulated with classical hardware. Next, we tested the effect of realistic energy measurements; that is, we simulate the entire variational process, including measuring the energy via the procedure described in Sec. I D. Finally, we tested the effect of noise in the quantum circuit. By contrast with the coherent errors considered in Ref. [24], we used a depolarizing noise model.

TABLE II. Number of occupied orbitals corresponding to the lowest energy of the Hubbard Hamiltonian for each grid size tested.

Occupied orbitals	Grid sizes
2	$1\times 2,\ 1\times 3,\ 2\times 2$
3	1×4
4	$1\times 5,\ 1\times 6,\ 2\times 3$
6	$1\times 7,\ 1\times 8,\ 2\times 4,\ 3\times 3$
7	1×9
8	$1\times 10,\ 1\times 11,\ 2\times 5,\ 2\times 6$
9	$1\times 12,\ 3\times 4$

For realistic energy measurements, we obtained a significant speedup by storing the probability amplitudes of the final state produced by the circuit. Computational basis measurements on that state were then simulated by sampling from this distribution, hence avoiding the need to rerun the circuit. This optimization is not available with noisy circuits, so those tests are much more computationally intensive.

We now outline some implementation decisions that were made. First, unless specified otherwise, we started with the number of occupied orbitals that corresponds to the lowest energy of the Hamiltonian H defined in Eq. (1) (not, e.g., the half-filled case as in Ref. [17]). These occupation numbers are listed in Table II. The ansätze we use preserve the fermion number and so remain in this subspace throughout the optimization process.

For the HV ansatz, one needs to choose the ordering of Hamiltonian terms for time evolution [see Eq. (2)]. By contrast, for the two efficient ansätze, this ordering is largely predetermined, except that we have a choice of when to implement the on-site terms in the EHV ansatz; we chose to do so at the start of each layer. In the case of $1 \times n_y$ grids, we used a O, V_1, V_2 ordering. For $2 \times n_y$ grids, we used a O, H, V_1, V_2 ordering (except 2×2, where there is no V_2 term). For $3 \times n_y$ grids, we used an O, H_1, V_1, V_2, H_2 ordering.

For all ansätze, one needs to choose initial parameters. We used a simple deterministic choice of initial parameters, which (similarly to Ref. [24]) were all set to $1/L$, where L is the number of layers. We also experimented with choosing initial parameters at random, e.g., within the range $[0, 2\pi/100]$; this achieved similar performance, suggesting that the optimization does not experience significant difficulties with local minima. In all cases, the initial state was the ground state of the noninteracting model (see Appendix C for a discussion of the effect of starting in a computational basis state).

A. Ability to represent ground state of the Hubbard model

The circuit ansätze we consider are divided into layers and, as the number of layers increases, the representational power of the ansatz increases. An initial test of the power of the variational method for producing ground states of the Hubbard model is to determine the number of layers required to produce the ground state $|\psi_G\rangle$ to fidelity 0.99, where

$$\text{fidelity}(|\psi\rangle) = |\langle\psi_G|\psi\rangle|^2.$$

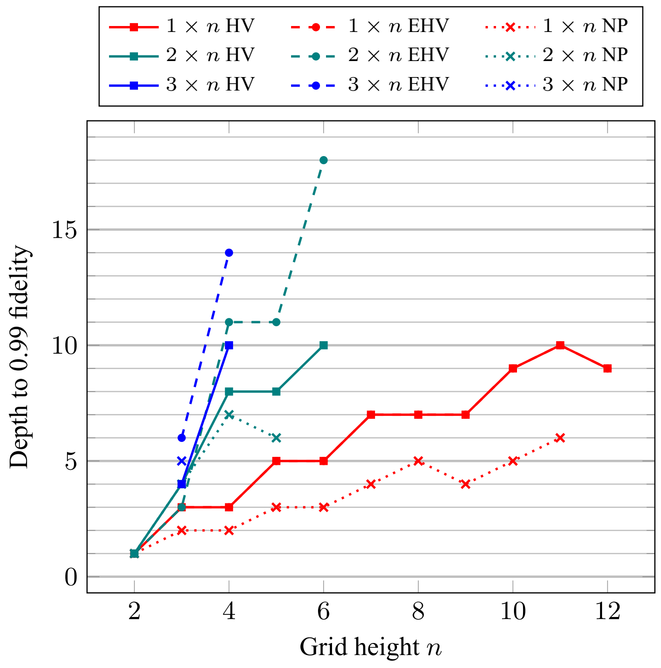

FIG. 7. Depths required (in terms of ansatz layers) to represent the ground state of the $n_x \times n_y$ Hubbard model for the HV, EHV, and NP ansätze. Each point corresponds to the minimal-depth circuit instance we found (using the L-BFGS optimizer) that produces a final state with fidelity at least 0.99 with the true Hubbard model ground state ($t = 1, U = 2$). Tests run for all grids of size $n_xn_y \leqslant 12$. For $1 \times n$ grids, HV and EHV are the same.

In Fig. 7, we show this for the HV, EHV, and NP ansätze. This illustrates that the EHV ansatz (which can be implemented efficiently) performs relatively well in comparison with the well-studied HV ansatz. In most cases (except the 2×3 grid), the HV ansatz requires a lower number of layers, but this is outweighed by the depth reduction per layer achieved by using the EHV ansatz. Note that in the case $n_x = 1$, the two ansätze are equivalent.

Figure 7 also illustrates that the NP ansatz generally requires lower depth than the other two ansätze to achieve high fidelity. This is expected, as it corresponds to optimizing over a larger set of circuits. However, it illustrates that the optimization procedure does not experience any significant difficulties with this larger set other than increased runtime, corresponding to the larger number of parameters. This increase can be significant; e.g., a 1×11 grid required approximately 10^5 function evaluations and a runtime of 16.5 h on a GPU-accelerated system to achieve fidelity 0.99 using the NP ansatz, whereas achieving the same fidelity using the EHV ansatz required fewer than 9000 function evaluations and a runtime of 1.5 h.

In Fig. 8, we illustrate how the fidelity improves with depth using the EHV ansatz, for the largest grid sizes we considered. In each case, the infidelity decreases exponentially with depth. Notably, 2×6 seems to be more challenging than 3×4.

B. Optimization with realistic measurements

We compared the ability of the SPSA and CD algorithms to find the ground state of Hubbard model instances for four representative grid sizes: 2×2, 1×6, 2×3, and 3×3. For CD, we fixed the number of approximate energy estimates to $\sim 1.2 \times 10^3$, where each estimate consists of 10^4 energy measurements. This translates to a limit of $\sim 6 \times 10^7$ circuit evaluations. For SPSA, on the other hand, the number of energy estimates was limited to $\sim 1.2 \times 10^4$, due to the number of measurements per estimate changing throughout the course of the optimization. As described in Sec. I E 1, we carry out a three-stage optimization routine and set the ratio of 10 : 3 : 1 for very coarse, coarse, and smooth function evaluations, respectively. By limiting to a total of $\sim 1.2 \times 10^4$ energy estimates, we allow for a similar total limit as that of CD, $\sim 1.2 \times 10^7$ energy measurements (or $\sim 6 \times 10^7$ circuit evaluations).

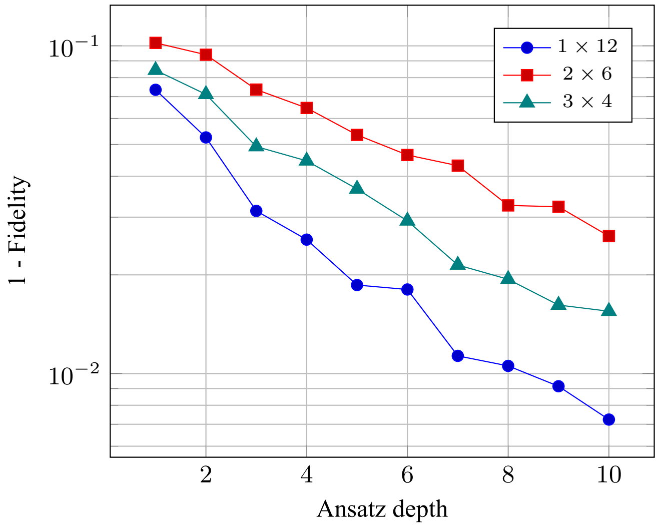

FIG. 8. Scaling of infidelity (1−fidelity) with number of layers of EHV ansatz for grids with 12 sites.

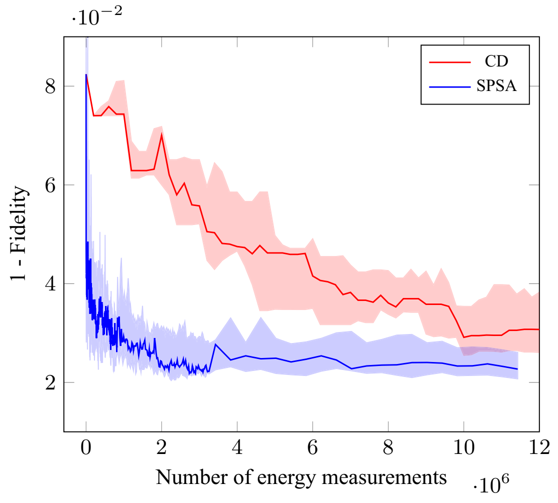

FIG. 9. Infidelity reached during the optimization process with CD and SPSA optimizers and realistic measurements. Results are shown for five runs of a 3×3 grid, EHV ansatz, depth 6. The solid lines show the median of the runs and the limits of the shaded regions are the maximum and minimum values seen over the five runs.

TABLE III. Final infidelity reached for CD and SPSA optimizers and realistic measurements, compared with the best infidelity achieved by the L-BFGS optimizer with exact measurements. EHV ansatz. CD, and SPSA results are median of five runs.

Grid	Depth	CD	SPSA	L-BFGS
2×2	1	0.0068	0.0066	0.0066
1×6	5	0.0293	0.0199	0.0098
2×3	3	0.0202	0.0199	0.0075
3×3	6	0.0307	0.0227	0.0068

For each grid size, we determined the final fidelity of the output of the VQE algorithm with the true ground state after the fixed number of measurements. For the circuit depth, we chose the minimal depth for which the ground state is achievable (via Fig. 7).

The results are shown in Fig. 9 and Table III. In all cases, both algorithms are able to achieve relatively high fidelity (considering that each energy measurement involves at most 10^4 circuit runs, suggesting an error of $\sim 10^{-2}$). However, in the case of 1×6 and 3×3 grids, SPSA achieves a noticeably higher fidelity. It is also interesting to note in Fig. 9 that SPSA uses substantially fewer energy measurements to achieve a high fidelity. One reason for this may be that each iteration of CD requires more energy measurements ($2n_xn_y + 1 = 19$ for a 3×3 grid, as compared with two energy measurements for SPSA).

C. Optimization with noisy quantum circuits

We next evaluated the effect of noise on the ability of the VQE algorithm to find the ground state of the Hubbard model. We considered a simple depolarizing noise model where, after each two-qubit gate, each qubit experiences noise with probability p (modeled as Pauli X, Y, Z operations occurring with equal probability). We examined noise rates $p \in \{10^{-3}, 10^{-4}, 10^{-6}\}$ and grid sizes 2×2, 1×6 and 2×3. These experiments are substantially more computationally costly than those with realistic measurements.

We tested the effect of the error-detection procedure described in Sec. I F. When an error is detected by the Hamming weight being incorrect, that run is ignored, and the measurement procedure continues until the intended number of valid energy measurements are produced for each type of term. Hence the total number of energy measurements is somewhat larger than the noiseless case.

We list the final fidelities achieved for different grid sizes, error rates, and optimization algorithms in Table IV. An illustrative set of runs for a 2×3 grid is shown in Fig. 10. The

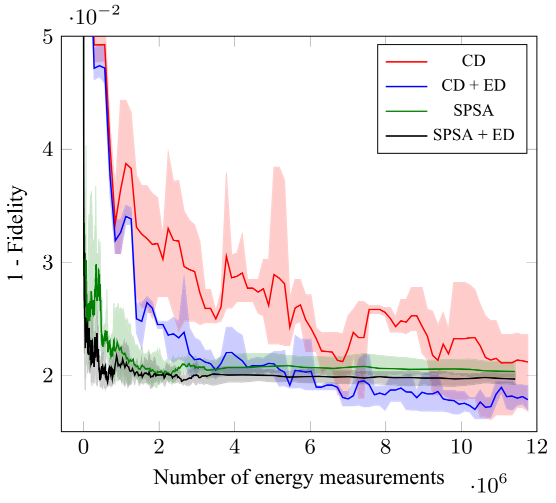

FIG. 10. Infidelity reached during the optimization process with CD and SPSA optimizers, with and without error detection (ED). 2×3 grid, 10^{-3} error rate, EHV ansatz, depth 3. The solid lines show the median of the runs and the limits of the shaded regions are the maximum and minimum values seen over the 3 runs.

overhead of error detection is not shown in this figure (that is, measurements where an error is detected are not counted). One can see that in all cases, errors do not make a significant difference to the final fidelities achieved, compared with the noiseless results in Table III. The use of error detection seems to usually lead to a small but noticeable improvement in the final fidelity achieved, as well as seeming to make the performance of the optimizer during a run less erratic. We note that error detection might have a more relevant role for bigger grid sizes, due to higher depths and longer circuit run times. However, more detailed experiments would be required to fully assess the benefit of error detection.

III. CONCLUDING REMARKS

We have carried out a detailed study of the complexity of variational quantum algorithms for finding the ground state of the Hubbard model. Our numerical results are consistent with the heuristic that the ground state of an instance on N sites could be approximately produced by a variational quantum circuit with $\sim N$ layers (and in all cases we considered, the number of layers required was at most $1.5N$).

If only around N layers are required, then the ground state of a 5×5 instance (larger than the largest instance solved classically via exact diagonalization [18]) could be found

TABLE IV. Infidelities at end of runs for varying grid sizes and noise rates, with error detection off/on. Median of three runs. EHV ansatz, depths 1, 5, 3, respectively.

	CD		SPSA			CD		SPSA			CD		SPSA	
2×2	No ED	ED	No ED	ED	1×6	No ED	ED	No ED	ED	2×3	No ED	ED	No ED	ED
10^{-3}	0.0066	0.0066	0.0066	0.0067	10^{-3}	0.0262	0.0297	0.0187	0.0176	10^{-3}	0.0231	0.0174	0.0201	0.0196
10^{-4}	0.0067	0.0068	0.0066	0.0066	10^{-4}	0.0250	0.0259	0.0188	0.0180	10^{-4}	0.0169	0.0179	0.0194	0.0195
10^{-6}	0.0065	0.0064	0.0066	0.0066	10^{-6}	0.0288	0.0257	0.0197	0.0185	10^{-6}	0.0174	0.0183	0.0199	0.0194

using a quantum circuit on 50 qubits with around 25 layers, corresponding to an approximate two-qubit gate depth of $24 + 25 \times (2 \times 5 + 2) + 1 = 325$ in a fully connected architecture, including the depth required to produce the initial state. This is significantly lower than the complexity for time-dynamics simulation reported in Refs. [5–7], but is still beyond the capabilities of today's quantum computing hardware. Although our work considered only relatively shallow quantum circuit depths, the ability of the NP ansatz to find ground states suggests that the classical optimization routines used could continue to work for these deeper circuits, as this ansatz used a much larger number of parameters, e.g., over 400 for the largest grids we considered.

While the Hubbard model is an important benchmark system in its own right, its simple structure facilitates an easier implementation of VQE than for typical electronic structure Hamiltonians. An important direction for future work is to carry out a similarly detailed analysis of the complexity of VQE for other practically relevant electronic systems.

Determining the optimal choice of classical optimizer remains an important challenge. It is plausible that the optimizers used here could be combined or modified to improve their performance, and other methods that have been studied contemporaneously with this work include adaptive optimization algorithms [65] and techniques based on machine learning or "meta-learning" [25,46]. Future work should evaluate such methods for larger-scale instances of the Hubbard model and other challenging problems in many-body physics.

Note added. Recently, we became aware of Ref. [62], which also determines theoretical resource estimates for applying the HV ansatz to solve the Hubbard model via VQE. The results obtained are qualitatively similar to ours; our circuit complexity bounds are lower, although the gate count estimates of Ref. [62] use a more restrictive gate set and topology targeted at efficient implementation on a specific hardware platform, so are not directly comparable. For example, if solving a 5×5 instance with a ten-layer HV ansatz, Ref. [62] would estimate a complexity of 11,300 two-qubit gates. By contrast, our estimate with unrestricted two-qubit gates and interaction topology [see Eq. (B2)] is fewer than 3351 two-qubit gates. The implementation strategy of Ref. [62] uses only nearest-neighbor interactions; the strategy discussed in Sec. B for a nearest-neighbor architecture is similar, but with some small differences.

Data are available at the University of Bristol data repository, data.bris [69].

ACKNOWLEDGMENTS

We would like to thank Toby Cubitt, John Morton, and the rest of the PhaseCraft team for helpful discussions and feedback, and Zhenyu Cai and Craig Gidney for comments on a previous version. L.M. received funding from the Bristol Quantum Engineering Centre for Doctoral Training, EPSRC Grant No. EP/L015730/1. Google Cloud credits were provided by Google via the EPSRC Prosperity Partnership in Quantum Software for Modeling and Simulation (EP/S005021/1).

APPENDIX A: ALTERNATIVE FERMION ENCODINGS

A well-known issue with the Jordan-Wigner transform is that it can produce qubit Hamiltonians which contain long strings of Z operators, leading to high-depth quantum circuits. This prompts us to consider alternative encodings of fermions as qubits which could reduce this depth. Here we evaluate two prominent encodings which produce qubit operators whose locality does not depend on the size of the grid. Although we have not proven that the time-evolution circuits we find are optimal, they provide an indication of the relative complexity of these encodings.

1. Ball-Verstraete-Cirac encoding

The encoding which (for arbitrary-sized grids) produces the lowest-weight qubit operators known is the Ball-Verstraete-Cirac or auxiliary fermion encoding [70], developed independently in Refs. [51,52].

The Ball-Verstraete-Cirac encoding can be seen as an optimized Jordan-Wigner encoding that avoids the need for long Z strings, at the expense of adding more qubits. Each fermionic mode (with the possible exception of two of the corners of the grid) is associated with an auxiliary mode, and vertical hopping terms use these modes. In this section, we change notation slightly and let operators of the form $X_{k,l}$ denote Pauli operators acting on the site k, l, while letting primed operators of the form $X'_{k,l}$ denote Pauli operators acting on the auxiliary mode associated with site k, l. Although there is some freedom in the encoding, the simplest mapping of the hopping terms presented in Ref. [52] is as follows. Each vertical hopping term $a^\dagger_{k,l}a_{k,l+1} + a^\dagger_{k,l+1}a_{k,l}$ maps to either

$$V_{k,l} := (-1)^{l+1}(X_{k,l}X_{k,l+1} + Y_{k,l}Y_{k,l+1})X'_{k,l}Y'_{k,l+1}$$

if k is odd, or

$$V'_{k,l} := (-1)^{l+1}(X_{k,l}X_{k,l+1} + Y_{k,l}Y_{k,l+1})Y'_{k,l}X'_{k,l+1}$$

if k is even. Each horizontal hopping term $a^\dagger_{k,l}a_{k+1,l} + a^\dagger_{k+1,l}a_{k,l}$ maps to

$$H_{k,l} := (X_{k,l}X_{k+1,l} + Y_{k,l}Y_{k+1,l})Z'_{k,l}.$$

The on-site terms remain the same as in the usual JW encoding. Using that $X \otimes X + Y \otimes Y$ can be mapped to $Z \otimes I - I \otimes Z$ by unitary conjugation, time evolution according to each horizontal term can be implemented with a circuit of two-qubit gates of depth 4 (which is more efficient than time evolving according to the XXZ terms and YYZ terms separately). For any term of the form $(X_1X_2 + Y_1Y_2)Z_3$, we first map the first two qubits to $Z_1 - Z_2$, then perform time evolution $e^{i\theta Z_1Z_3}$, $e^{-i\theta Z_2Z_3}$, then undo the first transformation. The vertical terms are similar, but somewhat more complicated. Now we want to evolve according to a term of the form $(X_1X_2 + Y_1Y_2)X_3Y_4$. We perform the same map on the first two qubits; then evolve according to $e^{i\theta Z_1X_3Y_4}$; then similarly for $-Z_2$; and then undo the first map. Now the intermediate time-evolution steps each can be implemented using a circuit of depth 3, because they correspond to computing parities of three bits each (and some additional one-qubit gates, which we do not count). However, the parity of qubits 3 and 4 does not need to be recomputed between these time-evolution steps, which saves depth 2; and the unitary operation diagonalizing $X_1X_2 + Y_1Y_2$ can be performed in parallel with computing this parity. These two optimizations reduce the overall depth complexity of time evolution according to each vertical term to 4, which is more efficient than implementing the $XXXY$ and $YYXY$ terms separately.

Therefore, the depth required to carry out all time-evolution steps for an arbitrary grid under the Ball-Verstraete-Cirac transformation is $2(4+4)+1=17$, assuming that an arbitrary two-qubit gate can be implemented in depth 1, and that there are no locality restrictions. This is higher than the cost of executing all time-evolution steps in one layer of the NP ansatz under the Jordan-Wigner transformation for all $n_x \times n_y$ grids such that $\min\{n_x, n_y\} \leqslant 8$. The Ball-Verstraete-Cirac encoding also comes with a significant increase in qubit count (from $2n_xn_y$ to $4(n_xn_y-1)$ [70]), as well as an additional cost for preparing the initial state, which we have not considered here.

2. Bravyi-Kitaev superfast encoding

Bravyi and Kitaev introduced another encoding of fermions as qubits [50] which produces $O(1)$-local operators, and which is now known as the Bravyi-Kitaev superfast transformation. In this encoding, one introduces a qubit for every hopping term in H (equivalently, a qubit for each edge in the lattices for each spin), giving an overall system size of $4n_xn_y - 2n_x - 2n_y$ qubits. Then, as described in Ref. [71], horizontal hopping terms $a_j^\dagger a_k + a_k^\dagger a_j$ map to terms of the form

$$\tfrac{1}{2}Y_j^{\rightarrow}\left(Z_j^{\downarrow}Z_k^{\uparrow} - Z_j^{\uparrow}Z_j^{\leftarrow}Z_k^{\rightarrow}Z_k^{\downarrow}\right),$$

where we follow the notation from Ref. [71] that arrow superscripts identify qubits in terms of their positions relative to sites k and j. Vertical hopping terms $a_j^\dagger a_k + a_k^\dagger a_j$ map to terms of the form

$$\tfrac{1}{2}Y_j^{\uparrow}\left(Z_k^{\leftarrow}Z_k^{\rightarrow}Z_k^{\uparrow}Z_j^{\leftarrow}Z_j^{\rightarrow}Z_j^{\downarrow} - I\right).$$

Finally, on-site interactions $n_{k\uparrow}n_{k\downarrow}$ map to terms

$$\tfrac{1}{4}\left(I - Z_k^{\leftarrow}Z_k^{\uparrow}Z_k^{\rightarrow}Z_k^{\downarrow}\right)\left(I - Z_{k'}^{\leftarrow}Z_{k'}^{\uparrow}Z_{k'}^{\rightarrow}Z_{k'}^{\downarrow}\right),$$

where k and k' correspond to sites in the spin-$\uparrow$ and spin-$\downarrow$ lattices, respectively.

In the horizontal hopping term, all Pauli matrices act on separate qubits with the exception of the $Y_j^{\rightarrow}$ component. Up to local unitary operations on the corresponding qubit, these terms (and the others) can be interpreted as performing rotations conditional on the parities of subsets of bits. The parity of five bits that needs to be computed for the $Y_j^{\rightarrow}Z_j^{\uparrow}Z_j^{\leftarrow}Z_k^{\rightarrow}Z_k^{\downarrow}$ part dominates the complexity of the whole evolution, as the part involving three qubits ($Y_j^{\rightarrow}Z_j^{\downarrow}Z_k^{\uparrow}$) can be executed in parallel with this. Then the depth required for time evolution for each hopping term is 6 two-qubit gates (the subroutine comprises a depth-3 circuit of CNOT gates to compute the parity of five bits; one one-qubit rotation gate; and another depth-3 circuit to uncompute). The vertical term is similar, but involves parities of seven bits, which can also be evaluated in depth 3, giving a depth-6 circuit in total (and noting that the identity term produces a single-qubit gate).

Finally, evolution according to each on-site term can be performed by first storing the parity of the four required bits in the lattice for each spin (which requires depth 2), then performing a two-qubit gate across the two lattices, and uncomputing the first step. The total two-qubit gate depth is 5.

Note that each of the horizontal and vertical hopping terms across sites j and k involves all qubits adjacent to j and k. This implies that, e.g., considering two horizontal terms across the pairs of sites (j_1, k_1) and (j_2, k_2), if j_2 is a neighbor of k_1 in a horizontal direction, or j_2 is a neighbor of j_1 in a vertical direction, there will be qubits that participate in the encoded hopping terms for both of these terms. To avoid these qubits overlapping, all qubits involved in different hopping terms should be distance 2 from each other. This would involve splitting the horizontal (and similarly vertical) hopping terms into six groups: by row (even versus odd), and by column (mod 3). A similar issue occurs with the onsite terms, which each involve all qubits neighboring a particular site. However, here all terms can be implemented using two groups.

In total, then, the depth to carry out all time-evolution steps under the Bravyi-Kitaev superfast encoding (under the same assumptions as the previous section) is $2(6\times 6) + 2\times 5 = 82$, which is substantially higher than the Ball-Verstraete-Cirac encoding. We have not attempted to optimize the circuits sketched above and it is possible that the large overhead from needing to split terms into groups that are implemented separately could be reduced or eliminated by implementing the required parity operations in a carefully chosen order. If it were possible to implement all groups of commuting horizontal, vertical, and on-site terms simultaneously (similar to the Ball-Verstraete-Cirac encoding) we would achieve a depth of $4\times 6+5=29$, which is still worse than the Ball-Verstraete-Cirac encoding.

APPENDIX B: IMPLEMENTATION ON HARDWARE

The description of the EHV ansatz from Sec. I B assumes that gates can be implemented across arbitrary pairs of qubits. Most quantum computing architectures have restrictions on their connectivity. These architectures will in general require additional swap operators to move pairs of qubits into positions in which they can interact, and then to move them back again. However, almost all of the gates that are applied in the ansatz take place along the 1D line of the JW ordering; the only other gates are on-site terms. This means that the EHV and NP ansätze can be implemented on a $2\times(n_xn_y)$ nearest-neighbor architecture with no depth overhead per circuit layer. This approach would require an overhead scaling with n_x to measure the vertical hopping terms, and would also require the qubit layout to be particularly "long and thin" (or a larger lattice of which this would be a subgraph). In this Appendix, we describe alternative approaches to implement the EHV/NP ansätze on realistic architectures whose shape is closer to the shape of the grid itself.

Once we have decided on a qubit layout, we can consider the cost of implementing the operator U_RU_L from Sec. I B and how it can be combined with the vertical hopping terms. Since vertical hopping terms are always applied in the same positions (those pairs of qubits that are vertically JW-adjacent), the same operator is used to apply all of them (one round at a

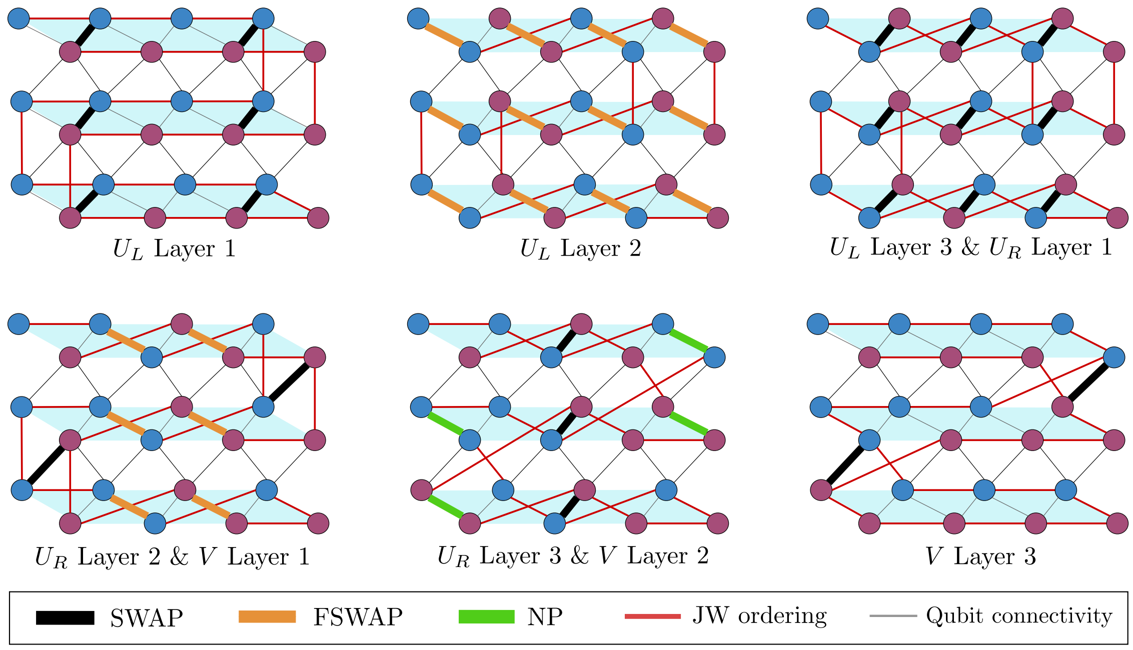
U_L Layer 1
U_L Layer 2
U_L Layer 3 & U_R Layer 1
U_R Layer 2 & V Layer 1
U_R Layer 3 & V Layer 2
V Layer 3
SWAP
FSWAP
NP
JW ordering
Qubit connectivity

FIG. 11. Implementation of the operator VU_RU_L (each split into three layers) on the Google Sycamore architecture for even n_x, shown here for a 4×3 grid. The VU_RU_L operator handles only the vertical hopping terms of the NP ansatz, and we remind the reader that the NP ansatz is a generalization of the HV ansatz. Here we assume that SWAP, FSWAP, and number-preserving (U_{NP}) gates can be implemented in depth 1. Once again the red lines represent the ordering of qubits due to the JW encoding. Observe that during the circuit, the red lines move—this represents the fact that qubits move physically while retaining the same JW ordering. However, applying an FSWAP gate between two JW-adjacent qubits has the effect of swapping the *ordering* of the two qubits, as well as their physical positions. Hence, FSWAP gates do not alter the relationship between the JW ordering and the physical layout of the qubits, whilst conventional SWAP gates do.

time)—we will call this V. The depth of the circuit required to implement one layer of the ansatz will then be determined by the depth of the circuit required to implement VU_RU_L, which is repeated n_x times, plus the depth of the circuits used to implement the horizontal hopping and onsite terms.

On a nearest-neighbor architecture, we could use a qubit layout similar to that described in Fig. 1 but where the lattice consists of alternating rows of spin-up and spin-down qubits. In this layout, horizontally JW-adjacent qubits are physically adjacent, but vertically JW-adjacent qubits are not. This means that the operators U_L and U_R, which swap horizontally JW-adjacent qubits, can be implemented directly in depth 1 each. The operator V requires that each pair of vertically JW-adjacent qubits are moved so that they become physically adjacent, and then moved back again, which can be achieved using two layers of SWAP gates. The first layer of SWAP gates can be implemented in parallel with the U_R operator (for even n_x,[7]) meaning that VU_RU_L can be implemented by a circuit of depth 4 (as was mentioned in Sec. I C). Also as discussed in Sec. I C, we can fold the horizontal hopping interactions into the swap network. Finally, all on-site interactions can be implemented in depth 1. This yields a final circuit depth of $4n_x + 1$ per layer.

[7]For odd n_x, some of the SWAP gates can be implemented in parallel with U_R, and others with U_L. In the end, this incurs an extra overhead of only depth 1, using an approach similar to that described in Fig. 12.

This approach is quite similar to the swap network used in Ref. [55]. There, spin-up and spin-down qubits are adjacent in the JW ordering (with an alternating up, down, down, up, up ... pattern), as opposed to the alternating rows used here. A depth upper bound of $3\sqrt{2}n_x$ per layer was stated in Ref. [55]; it was recently observed by Cai [62] that this can be improved to $4n_x$ using a modified swap network, similar to the one we use here. The depth of $4n_x + 1$ stated here could be decreased to $4n_x$ to match this by combining on-site interactions with SWAP operations, although this would change the ordering of the interactions performed in the ansatz. The alternating approach of Refs. [55,62] seems to need an additional swap gate at the end when measuring some of the horizontal hopping terms (those corresponding to pairs that are distance 3 in the JW ordering) but it should be possible to remove this by changing the JW ordering for runs that finish by measuring these terms.

The above interlaced approach would result in a physical lattice of shape $n_x \times (2n_y)$. However, instead of alternating rows of spin-up and spin-down, we can also place the spin-up lattice physically next to the spin-down lattice. This results in a lattice of shape $(2n_x) \times (n_y)$. The horizontally and vertically JW-adjacent terms are then adjacent on the physical lattice as well, and we can carry out these terms as described in Sec. I C. However, the qubits which we want to implement on-site terms across are distance n_x from each other. Using a swap network of depth $n_x - 1$, where the ith layer swaps i pairs of adjacent qubits starting from the middle of each row, we can bring the required qubits next to each other. We then perform the on-site gate and then use $n_x - 1$ more layers to swap to the original position. This approach then gives a final depth of $4n_x - 1$ for even n_x, and $4n_x$ for odd n_x which is a slight improvement on the interlaced approach. Also, the interlaced approach requires an additional layer of SWAP gates at the end of the algorithm to measure vertical hopping terms, which is not required for the separated approach.

As another example, we consider how to implement the above ansatz efficiently on Google's Sycamore architecture [8]. We use the qubit layout described in Sec. I A. Once again, we are concerned with the depth of the circuit required to implement VU_RU_L. In the Sycamore architecture, no JW-adjacent qubits are physically adjacent—they are all distance 1 away from each other—and so each of U_R, U_L, and V must be split into three layers each: one to swap qubits into physically adjacent positions, one to carry out the required interaction, and one more to swap the qubits back to their original positions. Many of these layers overlap and can be implemented in parallel. Figure 11 illustrates how to implement the operator VU_RU_L with a circuit of depth 6 for even values of n_x.

Once again, we can fold the horizontal hopping interactions into the swap network, and all on-site interactions can be implemented in depth 1. This yields a final circuit depth of $6n_x + 1$ per layer for even values of n_x.[8] For odd values of n_x, we lose the ability to implement the vertical hopping terms in parallel with the operator U_R, which increases the depth of the final circuit. In Fig. 12 we show how to implement the operator VU_RU_L in depth 7. Here (but not in the even n_x case), the first and last layers can be implemented in parallel, and so we obtain a final circuit depth for the ansatz of $6n_x + 2$ per layer, one more than in the even case.

We are now able to compare the effect of different qubit connectivities on circuit depth. These are shown in Table I in the introduction. An estimate of two-qubit gate complexity (as opposed to depth) for a complete run of the whole circuit for the efficient version of the HV or NP ansatz follows.

The cost of preparing the initial state is at most $2(N-1)\lfloor N/2\rfloor$ gates, where $N = n_xn_y$. Then the cost of the ansatz circuit itself is at most the depth per layer multiplied by the maximal number of two-qubit gates applied per step of the circuit (which is at most N), multiplied by the number of layers. Finally, there is a cost of at most N for the two-qubit gates required for performing the final measurement. For example, in the case of a fully connected architecture, the gate complexity for a circuit with L layers is at most

$$(N-1)N + (2n_x + 1)NL + N \tag{B1}$$

for even n_x, and

$$2(N-1)\lfloor N/2\rfloor + (2n_x + 2)NL + N \tag{B2}$$

for odd n_x. In the special case of a 2×4 system with two layers, and using a more careful calculation, we obtain a bound of at most 36 gates per layer, giving an upper bound of 136 gates in total. By contrast, the estimate for this case in Ref. [17] was 1000 gates, more than a factor of 7 higher.

[8]Note that there is no dependence on n_y. If $n_y < n_x$, we are free to rotate the grid (i.e., by choosing a snake-shaped ordering that travels along the y axis) so our new grid has n_y columns. Therefore, the circuit depth is more correctly stated as $6\ \min\{n_x, n_y\} + 1$ for even n_x, n_y.

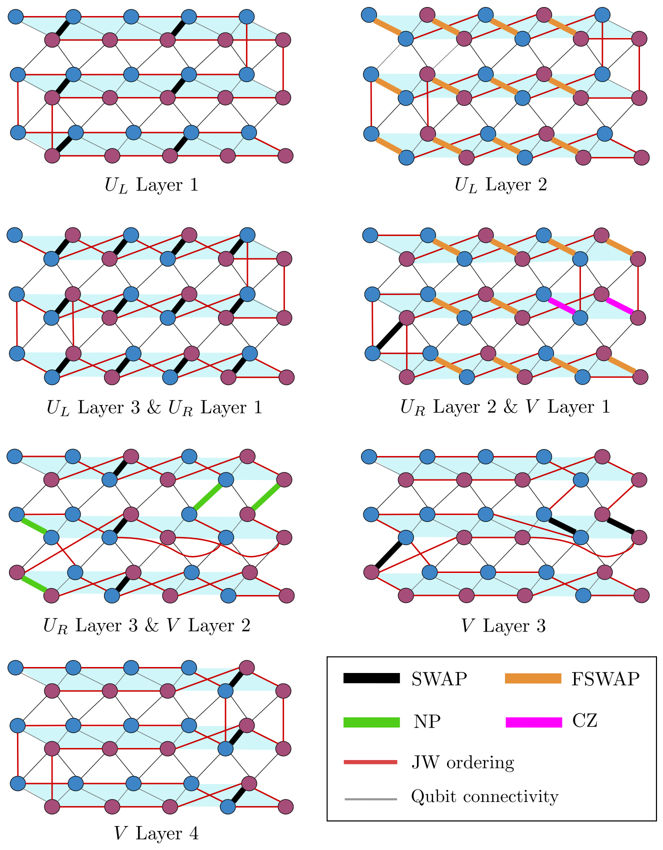

FIG. 12. Implementation of the operator VU_RU_L on the Google Sycamore architecture for odd n_x, shown here for a 5×3 grid. Note the CZ gates in the fourth layer of the circuit which is a combination of the FSWAP gate from layer 2 of U_R and the SWAP gate from layer 1 of V.

APPENDIX C: THE NUMBER-PRESERVING ANSTAZ

In this Appendix, we will go into details about the choices that can be made when implementing the NP ansatz. As with many ansätze, we must specify properties such as starting parameters and initial states.

1. Initial state

As well as the ground state of the noninteracting Hubbard model, the NP ansatz also allows a computational basis state with the correct fermionic occupation number as an initial state. All gates in the circuit are fermionic NP, so the VQE method will find the ground state of the Fermi-Hubbard Hamiltonian restricted to the chosen occupation number subspace. This allows a saving in initial complexity compared with starting in the ground state of the noninteracting model (although with an associated penalty in terms of the number of layers required to find the ground state).

The sites we choose to be occupied by fermions can make a significant difference to the complexity at a fixed depth. We ran a number of tests brute forcing all the possible starting states on selected small grid sizes. We found that in many

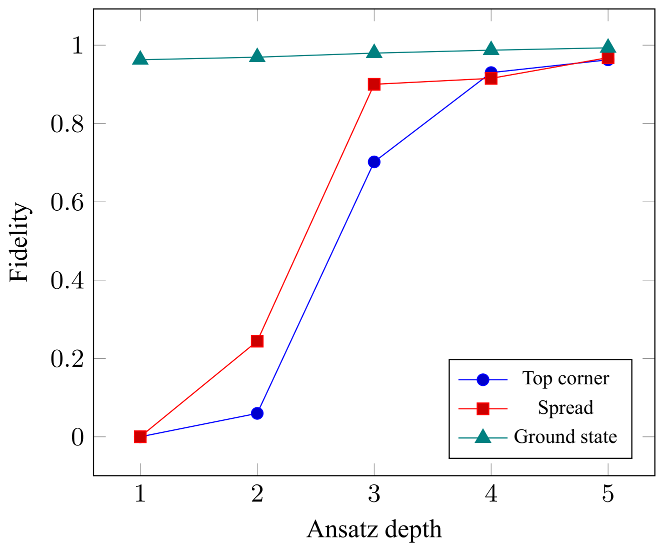

FIG. 13. Comparison of initial fermion placements against using the ground state as a starting state for a 3 × 3 grid occupied by six fermions. For the spread-out state, we occupied both spins for the three sites along the main diagonal of the grid. The spread-out placement generally performs better than the top corner placement that fills the first six orbitals, especially for lower depths. Only starting in the ground state achieves fidelity 0.99, while the others reach around 0.96 in depth 5.

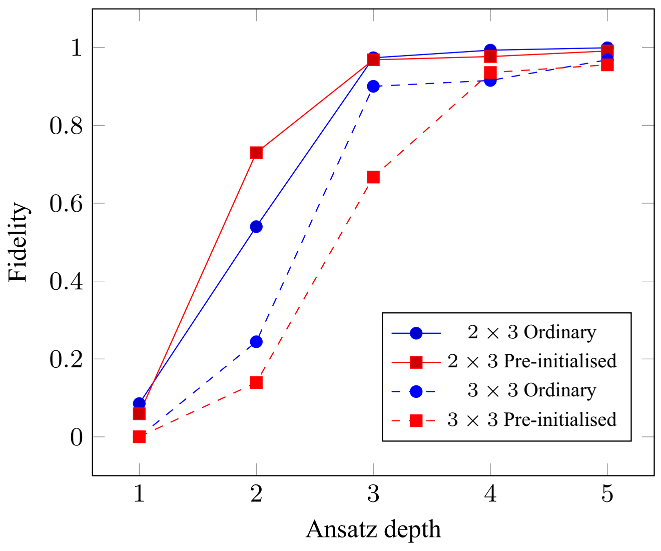

FIG. 14. Comparison of the preinitialized NP ansatz to the ordinary NP ansatz for 2 × 3 occupied by four fermions and 3 × 3 occupied by six fermions. The initial placement of the fermions is spread out (for 2 × 3 we fully occupy two sites at opposite corners of the grid, 3 × 3 is explained in Figure 13). Preinitialization improves the results for 2 × 3 depth 2, but makes it worse for 3 × 3 in all cases. The difference between the ordinary and preinitialized ansatz reduces as the depth increases; similar behavior was demonstrated in Fig. 13.

cases the best states reached errors several orders of magnitude better than the worst states, but given the small lattice sizes considered, the pattern for picking these good states remains unclear.

An intuitive approach would be to place fermions evenly across the grid, allowing them to quickly spread out. Then the ground state (if it does indeed correspond to a spread-out state) can be produced from the initial state using potentially fewer layers of the ansatz circuit. Empirically, we observed that the optimizer performed better with this layout than a naïve one where fermions are placed at the top left corner of the grid, although we note that other schemes might yield even better results. Figure 13 gives a demonstration for a 3 × 3 grid occupied by six fermions.

For a 3 × 3 grid, ground state of the noninteracting model can be prepared in depth 8 (assuming unrestricted qubit connectivity), whereas each NP ansatz layer requires depth 7. So, in this case, starting with a computational basis state does not seem to be advantageous. We further remark that the NP ansatz starting from a computational basis state cannot find the true ground state of the noninteracting Hubbard model in the case where the number of fermions with each spin is 1. This is because all computational basis states with Hamming weight 1 are in the null space of this model, and hopping terms preserve this subspace, as we show in Appendix C 3.

2. Preinitializing ansatz parameters

In the main paper, the initial state of the NP ansatz is the noninteracting Hubbard model ground state. However, starting with a computational basis state, the ansatz (and therefore the optimizer) has to do more work to produce something close to the ground state of the full model.

To reduce the work that the optimizer needs to do, we can first find an ansatz circuit that produces a state close to the ground state of the noninteracting model by classically emulating the VQE procedure. Because we only need to consider a single spin, the number of qubits in the emulation is halved. For small grid sizes feasible on near-term quantum devices, the noninteracting problem will be tractable on a classical computer. An advantage of classically emulating the procedure (rather than also running these smaller instances on a quantum computer) is that we can use simulated exact measurements.

Once we have performed the optimization classically, we can preinitialize the parameters of the full-model ansatz by using the final parameters from the noninteracting model. The intuition is that by allowing the optimization procedure to begin with a circuit that produces the ground state of the noninteracting model (which we know is a good choice from Fig. 7), it then "only" has to optimize this circuit to produce a ground state of the complete model, having already been pointed in the right direction.

However, it is not clear when this procedure is beneficial as for some grid sizes and depths it causes the ansatz to perform worse. Figure 14 demonstrates this for 2 × 3 and 3 × 3 grids where the initial placement of the fermions is spread out. We note that different placements change how effective the preinitialized ansatz is, and that this requires more investigation.

3. Occupation number 1

Here we show that the NP ansatz starting from a computational basis state cannot find the ground state of the noninteracting Hubbard Hamiltonian, when there is one occupied mode. All computational basis states with Hamming weight 1 are in the null space of the non-interacting Hubbard Hamiltonian. To show that the ground state cannot be found, it is sufficient to prove that time evolution according to hopping terms preserves this subspace.

In a system with N modes, any state which is a linear combination of occupation number 1 basis states can be written as $\sum_{k=1}^{N} \alpha_k |e_k\rangle$ for some coefficients α_k, where e_k is the vector with Hamming weight 1 whose kth entry is 1. Within this N-dimensional space, the hopping term $(X_iX_j + Y_iY_j)/2$ (where i and j are adjacent in the Jordan-Wigner ordering) acts as an X gate within the two-dimensional subspace span$\{|e_i\rangle, |e_j\rangle\}$. Write X_{ij} for this gate. A state with Hamming weight 1 is contained within the null space of the hopping term between modes i and j (assuming that i and j are adjacent in the Jordan-Wigner ordering) if

$$\begin{aligned} 0 &= \left(\sum_{k=1}^{N} \alpha_k^* \langle e_k|\right) X_{ij} \left(\sum_{l=1}^{N} \alpha_l |e_l\rangle\right) \\ &= \alpha_i^*\alpha_j + \alpha_j^*\alpha_i \\ &= 2\,\mathrm{Re}(\alpha_i^*\alpha_j). \end{aligned}$$

Consider an arbitrary three-dimensional subspace corresponding to adjacent modes i, j, k in the Jordan-Wigner ordering. Then

$$\begin{aligned} &e^{i\theta X_{ij}}(\alpha|e_i\rangle + \beta|e_j\rangle + \gamma|e_k\rangle) \\ &\quad= (\alpha\cos\theta + i\beta\sin\theta)|e_i\rangle \\ &\qquad + (i\alpha\sin\theta + \beta\cos\theta)|e_j\rangle + \gamma|e_k\rangle \\ &\quad=: \alpha'|e_i\rangle + \beta'|e_j\rangle + \gamma'|e_k\rangle. \end{aligned}$$

To show that this state is contained within the null space of all hopping terms, it is sufficient to show that $\mathrm{Re}((\alpha')^*\beta') = \mathrm{Re}((\gamma')^*\beta') = 0$.

The former claim is immediate as the initial state is in the null space of X_{ij}. For the latter claim, we have

$$\begin{aligned} \mathrm{Re}((\gamma')^*\beta') &= \mathrm{Re}(\gamma^*(i\alpha\sin\theta + \beta\cos\theta)) \\ &= \cos\theta\,\mathrm{Re}(\gamma^*\beta) - \sin\theta\,\mathrm{Im}(\gamma^*\alpha). \end{aligned}$$

We have $\mathrm{Re}(\gamma^*\beta) = 0$ as the initial state is in the null space of X_{jk}. To see that $\mathrm{Im}(\gamma^*\alpha) = 0$, write $\alpha = r_\alpha e^{is_\alpha}$, and similarly for β, γ. Then, as $\alpha^*\beta$ and $\beta^*\gamma$ are imaginary from the same null space constraint, we have that $s_\beta - s_\alpha$ and $s_\gamma - s_\beta$ are in the set $\{\pm\pi/2, \pm3\pi/2\}$. So, $s_\gamma - s_\alpha$ must be an integer multiple of π, implying that $\gamma^*\alpha$ is real.

APPENDIX D: SIMULATION CHOICES

This Appendix summarizes the reasoning behind some choices that were made in our tests and presents additional results for other regimes.

1. Effect of choice of U parameter

Throughout this paper, we fixed the weight U of the on-site term in the Hubbard Hamiltonian Eq. (1) to 2, as was also done in Ref. [17]. To justify this, we considered three grid sizes (2×2, 1×6 and 3×3) and evaluated the fidelity achieved for different choices of U by optimizing using L-BFGS with exact energy measurements within the EHV ansatz, at the same depth for which the $U = 2$ case achieves fidelity >0.99. This gives a measure of the difficulty of finding the ground state. The results are shown in Fig. 15. One can see that the fidelity decreases as U increases, as expected given that the ansatz begins in the ground state of the $U = 0$ model. However, the final fidelity achieved continues to be quite high for all $U \leqslant 4$.

Figure 16 demonstrates the minimal depth of the EHV ansatz required to reach 0.99 fidelity as U varies. In general, the depth required increases as U does, which is to be ex-

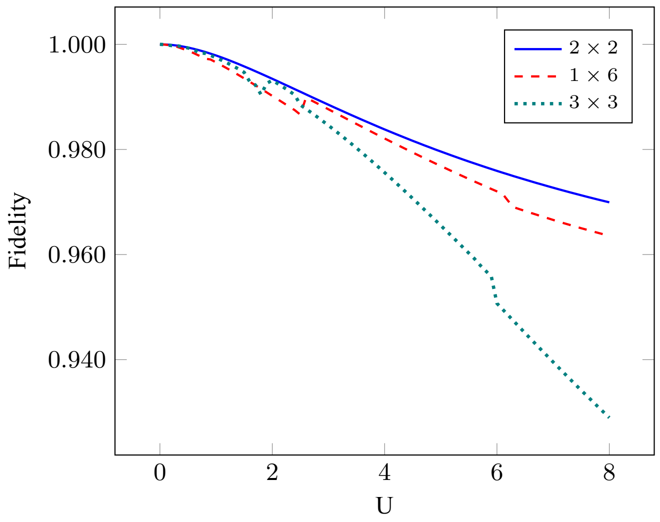

FIG. 15. The final fidelity achieved with varying U for a 2×2 grid (at depth 1), 1×6 (at depth 5), and 3×3 (at depth 6), using the EHV ansatz, simulated exact energy measurements, and the L-BFGS optimizer. U incremented in steps of size 0.1.

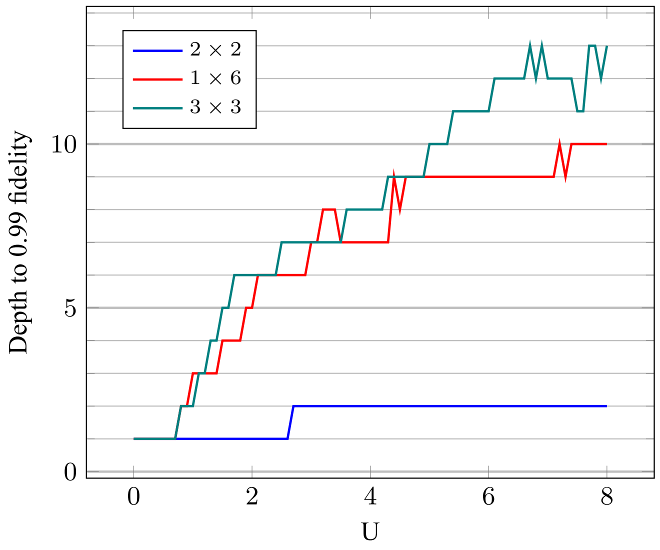

FIG. 16. Depth of the efficient Hamiltonian variational ansatz required to reach 0.99 fidelity with the ground state of the Hubbard model as a function of U with grid sizes 2×2, 1×6, and 3×3.

TABLE V. Comparison of FFT circuit depths and directly preparing the Slater determinant using Givens rotations for a variety of $n \times n$ grids.

Grid size	Asymptotically efficient (predicted)	Asymptotically efficient (actual)	Swap network	Swap network (modified)	Givens rotations
4×4	82	82	54	27	15
6×6	126	123	112	65	35
8×8	170	164	265	119	63
10×10	214	205	383	189	99
12×12	258	246	636	275	143
14×14	302	287	814	377	195
16×16	346	328	1167	495	255
18×18	390	369	1492	629	323

pected as we begin in the $U = 0$ ground state. As we can see from Fig. 16, to get to the physically interesting intermediate coupling regime $U = 4$, where classical methods experience significant uncertainties [[15], Table V], only requires one or two extra ansatz layers from $U = 2$. However, the more strongly correlated model $U = 8$ requires roughly double the ansatz layers.

We remark that, when optimizing with realistic measurements, the statistical uncertainty in the energy measurements is likely to increase linearly with U. This is because the energy is measured by summing several measurement results, some of which are scaled by a U factor. Thus going from $U = 2$ to $U = 8$ (for example) would likely require 16 times more measurements to achieve the same level of statistical uncertainty.

2. The half-filled regime

While we were mostly concerned with finding the ground state of the original Hamiltonian presented in Eq. (1), solutions of certain restricted cases can be of interest as well. A prominent restriction is that of half filling, where the number of fermions in the lattice is exactly half of the number of sites. This case is easier to solve classically due to the lack of a sign problem [15], enabling quantum Monte Carlo methods to succeed. However, the special physical and mathematical characteristics of the half-filled regime make it an important one in which to also benchmark VQE methods.

The performance of our algorithm in terms of depth to high-fidelity solution can be seen in Fig. 17; we can see that the depths required in the half-filling case are comparable to depths required to find the ground state of the full Hamiltonian for the same ansatz. In Fig. 18, we can see how the infidelity, absolute error with the actual ground state, and absolute error with the true double occupancy of the ground state changed with depth for an example grid size of 2x4, also at half filling. While the optimization is carried out by minimizing the energy, we can see that the infidelity and the error in the double occupancy follow a very similar trend to the error in the ground energy. This gives us reason to believe that energy is a good figure of merit to optimize on, even if a different property of the ground state (such as double occupancy) might

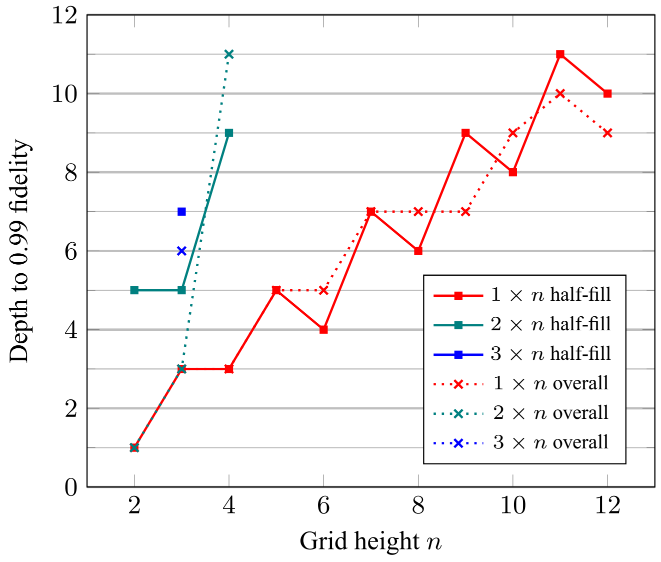

FIG. 17. Depth of the efficient Hamiltonian variational ansatz required to reach 0.99 fidelity with the ground state of the half-filled Hubbard model ($t = 1, U = 2$). Comparison with depth required to find the overall ground state (data reproduced from Fig. 7).

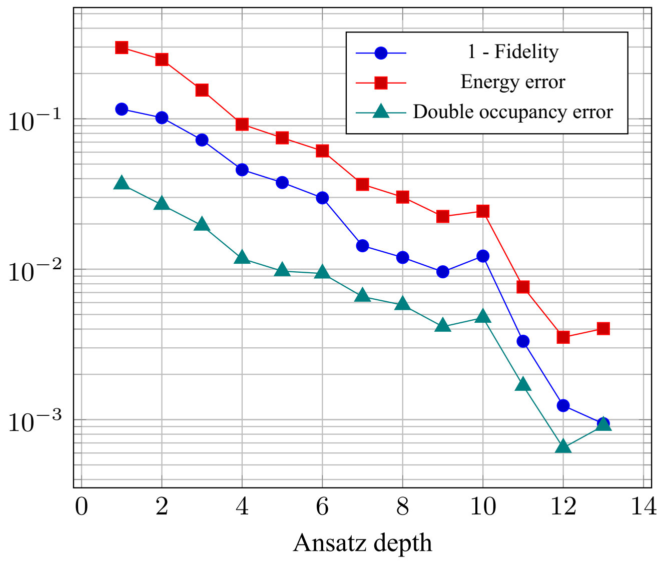

FIG. 18. Infidelity, absolute error with the actual ground state, and absolute error with the true double occupancy for various depths of the efficient Hamiltonian variational ansatz for the 2×4 half-filled Hubbard model.

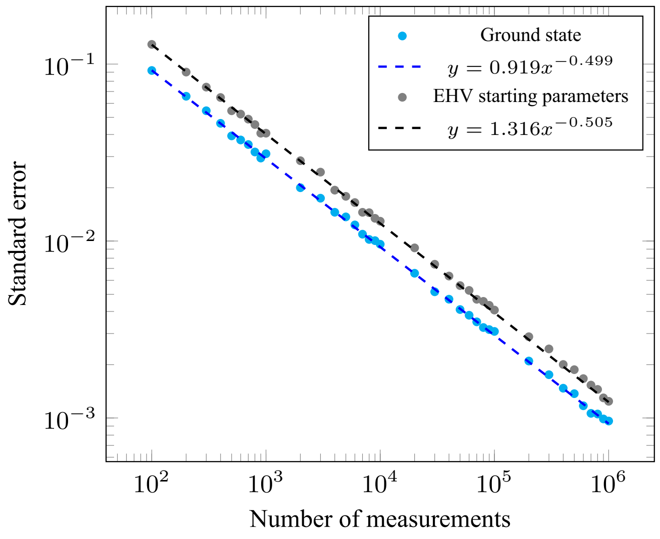

FIG. 19. Statistical error in approximating the energy with respect to the number of measurements made. Results are shown for a 2×2 grid and the starting parameters chosen for EHV are $1/d$ where d is the depth of the ansatz (six in this case). Each point on the graph is the standard deviation of 1000 samples, where each sample is the error in the estimated energy achieved using m measurements.

be the one we are interested in. The situation is similar away from half filling.

Another peculiarity about the half-filled case is that degeneracy in the ground states of the noninteracting Hamiltonian, which is the initial state for the EHV ansatz, is common. If the degeneracy is low enough (only a few states), then trying out each of the degenerate states as the initial state might be feasible. However, in some of the lattices with higher degeneracy we tried a few different solutions to arrive at a successful initial state. For the results presented in Fig. 17, the initial states were generated as follows: If there was no degeneracy, then the choice was the single ground state; for grid size 2×2, one of the hopping terms in the Hamiltonian (between sites top left two sites) was altered by $\epsilon = 0.0001$ allowing for a splitting between the degenerate states; for grid sizes 2×5, 3×3, a superposition over all the degenerate states was created and the weights of each of the states were added as parameters to be optimized over in the main optimization; for all other degenerate states, a manual selection of initial state was carried out by trial and error.

3. Characterizing statistical noise in the ansatz circuits

In Fig. 19, we present numerical results that justify performing 10^4 measurements on each term in the Hamiltonian to estimate the energy to an accuracy of $\sim 10^{-2}$. The statistical error on the state is larger when the circuit which produces it is not generating the ground-state (i.e., an eigenstate) of the Hamiltonian. Note that the lines of best fit in the figure show that the standard error goes down like $1/\sqrt{m}$, where m is the number of measurements, as expected.

APPENDIX E: PREPARING THE INITIAL STATE OF THE NON-INTERACTING HUBBARD HAMILTONIAN

This Appendix compares the complexities of different methods for preparing the ground state of the noninteracting Hubbard Hamiltonian [Eq. (1) with $U = 0$]: first, approaches to implement the 2D FFT on rectangular grids of size $n_x \times n_y$; and then an approach based on Givens rotations [54]. We will see that the latter is the most efficient for small grid sizes, while for large grid sizes an FFT algorithm of Ref. [54] is superior. The most efficient implementation of the full FFT for small grid sizes is the approach based on FSWAP networks.

1. Naïve approach to implementing the FFT

The naïve approach to implementing the FFT first separates it into horizontal and vertical components, $\mathcal{F}_x$ and $\mathcal{F}_y$. The terms $\mathcal{F}_x$ and $\mathcal{F}_y$ are products of commuting terms that involve qubits only in the same row and column, respectively. To implement $\mathcal{F}_x$, we apply the 1D Fourier transform on all rows in parallel. To implement $\mathcal{F}_y$, it is necessary to implement the 1D Fourier transform on all columns, but with the appropriate parity corrections (Z strings) attached to each Givens rotation performed. The parity corrections prevent us from implementing this part of the circuit on all columns in parallel, since the corrections span across multiple columns. Assuming that we don't implement any of the Z strings acting on the same row in parallel, and we use a simple 1D nearest-neighbor circuit for computing the parity corrections, then the depth of any FFT circuit implemented naïvely in this way will be

$$T_{\mathcal{F}}(n_x) + T_{\mathcal{F}}(n_y)\sum_{i=1}^{n_x} 2(n_x - i) + 1 = T_{\mathcal{F}}(n_x) + T_{\mathcal{F}}(n_y)n_x^2,$$

where $T_{\mathcal{F}}(n)$ is the depth of the circuit implementing the 1D FFT on n qubits. For an $n \times n$ lattice, the depth is thus $\Theta(n^3)$, assuming the depth of the 1D FFT is $O(n)$.[9]

2. Asymptotically efficient implementation of the FFT

Jiang *et al.* [54] described a method to implement the FFT on a 2D qubit array of size $n_x \times n_y =: N$ with $O(\sqrt{N})$ depth and $O(N^{3/2})$ gates. As in the previous section, the general approach is to factor the FFT into its horizontal and vertical components $\mathcal{F} = \mathcal{F}_x\mathcal{F}_y$.

Under the Jordan-Wigner transform, the horizontal part is straightforward to implement. Indeed, we can implement the 1D Fourier transform in parallel for all rows, without the need for parity corrections. However, the vertical component is much harder to implement because of the non-local parity operators required to correctly implement two-qubit interactions between neighboring qubits in a column. The approach developed in Ref. [54] is to decompose the vertical term as

$$\mathcal{F}_y = \Gamma^\dagger \mathcal{F}_y^b \Gamma,$$

where $\mathcal{F}_y^b$ is the vertical component without the parity operators (i.e., the 1D Fourier transform), and $\Gamma = \Gamma^\dagger$ is a diagonal

[9]In a fully connected architecture, parallel circuits could be used to implement the parity corrections; however, these would still not be competitive with the best complexity we find below using swap networks.

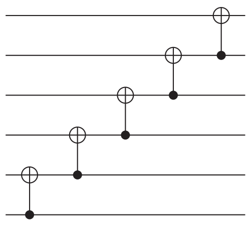
FIG. 20. Circuit to transform qubits in a column to the parity basis.

(in the computational basis) unitary that 're-attaches' the parity operators. $\mathcal{F}_y^b$ can be implemented using the same circuit as for $\mathcal{F}_x$, but applied to all columns in parallel. The circuit for Γ is more complicated.

The operator Γ can be implemented by attaching an additional qubit per row, and then using these to keep track of the parities of the qubits in their corresponding row. The general approach is roughly as follows (for details, see Ref. [54]):

(1) Convert each column to the parity basis via a sequence of CNOT gates.

(2) Move the ancilla qubits to the left while updating their parity, using a SWAP gate, followed by a CNOT between the ancilla and the system qubit now to its right. As the qubits move to the left, we apply a sequence of Controlled-Z (CZ) gates to update the phases of the system qubits.

(3) Once all qubits reach the left-hand side, undo the conversion to the parity basis by undoing the CNOT gates.

(4) Move the ancilla qubits rightward by applying the CNOT and SWAP gates in reverse. At each step, apply some more CZ gates to update the parities of the system qubits correctly.

Every step requires a circuit of depth $O(\sqrt{N})$ with $O(N)$ gates. Here we will attempt to calculate the constants associated with this asymptotic scaling.

(1) Step 1 is a transformation to the parity basis. This circuit requires $n_y - 1$ gates per column, and therefore $n_x(n_y - 1) = N - n_x$ gates in total, with a depth of $n_y - 1$ (see Fig. 20).[10]

(2) The circuit in step 2 is more complicated to implement efficiently. To implement these gates as they are described in Ref. [54] on a nearestneighbor architecture, we would need to perform a number of swap operations to bring the system and ancilla qubits together, requiring two swap operations per gate. Luckily, we can avoid paying double for these gates by reordering the SWAP and CNOT gates used to move the ancilla qubits to the left.

By using this reordering approach, the total number of gates required to implement the step is $\frac{3n_y}{2}n_x = \frac{3N}{2}$ and the total depth is $4n_x$.

This calculation ignores the fact that there are vertical CZ gates acting on nonadjacent rows. Here we have two options. One option is to move the qubits in all odd-numbered rows so they are all adjacent to each other before step 2, and then move them back afterward. Using the circuit in Fig. 21, this adds an additional overhead of $n_x(\frac{n_y}{2} - 2)(\frac{n_y}{2} - 1)$ gates with a depth of $n_y - 2$ (for both doing and undoing the circuit).

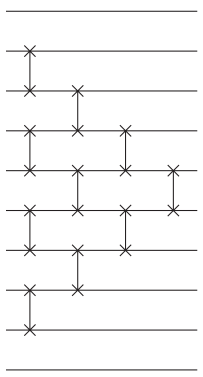
FIG. 21. Circuit to bring all odd-numbered rows and all even-numbered rows together.

Alternatively, we could just apply SWAP gates as and when we need them. This would involve applying two SWAP gates per vertical CZ operation in this step. We can't apply any of them in parallel with any of the CZ operations, and so we have a total overhead of $n_x n_y$ gates and an increased depth of n_x. Hence, we always save a constant depth of 2 by using the first approach and for $4 \leqslant n_y \leqslant 13$, it uses fewer gates.

Choosing the first approach for implementing CZ gates on non-adjacent rows, step 2 can be implemented using a circuit with depth $4n_x + n_y - 2$. If we are allowed to apply gates to arbitrary pairs of qubits, then the circuit depth reduces to just $4n_x$.

(3) Step 3 uses the same gates as step 1, except in reverse and with $n_y - 1$ additional CNOT gates for the ancilla column, and therefore requires a circuit of $(n_x + 1)(n_y - 1)$ gates with a depth of $n_y - 1$.

(4) Step 4 is similar to step 2, but somewhat simpler. We move the ancilla qubits to the right using CNOT and SWAP gates, whilst applying local CZ gates at each time step. The number of gates required to move the qubits to the right is $2n_x n_y$, and the number of CZ gates that need to be applied is $n_x n_y$, giving a total of $3n_x n_y$. We can apply all gates in parallel for each row, giving a total depth of $2n_x + 2n_x = 4n_x$.

Once again, using a nearest neighbor architecture and the first approach mentioned in Step 2 would increase this circuit depth to $4n_x + n_y - 2$.

Putting all these steps together gives us a total circuit depth to implement Γ of

$$n_y - 1 + n_y - 1 + 4n_x + 4n_x = 2n_y + 8n_x - 2$$

if we are allowed to apply gates to arbitrary pairs of qubits, and a depth of

$$\begin{aligned} &n_y - 1 + n_y - 1 + 4n_x + n_y - 2 + 4n_x + n_y - 2 \\ &\quad = 4n_y + 8n_x - 6 \end{aligned}$$

if we use a nearest-neighbor architecture. Suppose that the depth of the circuit used to implement the 1D FFT on n qubits is $T_{\mathcal{F}}(n)$. Then combining the three stages described above: $\mathcal{F}_x$, $\mathcal{F}_y^b$, and Γ and $\Gamma^\dagger$, we obtain a final circuit depth of

$$T_{\mathcal{F}}(n_x) + T_{\mathcal{F}}(n_y) + 2(8n_x + 2n_y - 2).$$

[10]This depth could be reduced to $O(\sqrt{n_y})$ on a nearest-neighbor architecture, or $O(\log n_y)$ on a fully connected architecture [72]. This would reduce the grid size slightly at which this approach starts to outperform its competitors.

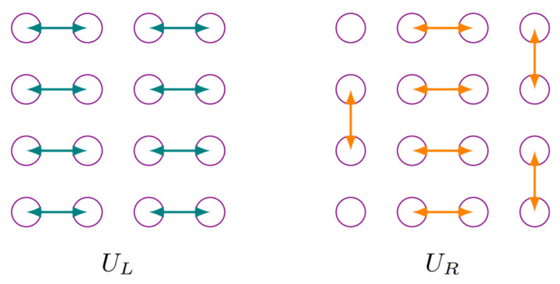

FIG. 22. Action of sets of fermionic swaps U_L and U_R on a 4×4 grid of qubits, using the swap network of [55]. The ordering of the qubits is the snake ordering in the main paper.

If we are restricted to an architecture that only allows interactions between neighboring qubits, the depth of the circuit required to implement the FFT is

$$T_{\mathcal{F}}(n_x) + T_{\mathcal{F}}(n_y) + 2(8n_x + 4n_y - 6).$$

However, when we combine all stages of the FFT circuit together, there are some overlaps that are not accounted for in the above analysis. This means that the actual (optimized) circuit depth will be slightly less than predicted. In Ref. [54], the authors show that the 1D Fourier transforms can be implemented with circuits of depth $T_{\mathcal{F}}(n_x) = n_x - 1$ and $T_{\mathcal{F}}(n_y) = n_y - 1$. Table V shows the actual versus predicted depths of the FFT circuit for a number of (square) grid sizes on an unrestricted architecture. From these numbers it appears that parallelization of the stages gives us a depth saving of $3(n/2 - 2)$ for an $n \times n$ grid.

3. Fermionic swap networks for the FFT

To avoid needing to implement the phase corrections from the previous section, we could instead use the notion of a FSWAP network [55]. Here, we use FSWAP operators to move qubits next to each other so they can interact without the need for parity corrections. Crucially, these swap operators correctly maintain the relative phases between qubits required by the JW ordering. The idea is to apply a number of layers of two-qubit FSWAP gates so by the time we are done, every qubit has been adjacent to every other qubit, enabling it to interact without needing to worry about phase corrections due to the JW encoding. This notion can be extended to a 2D grid of spin orbitals. Following Ref. [55], by using a total of $3\sqrt{N/2}$ FSWAP operators we can implement all vertical and horizontal gates in the FFT using gates that can be implemented by nearest-neighbor interactions. These swap operators remove the need to implement the Z strings required to correctly simulate the vertical hopping terms under the JW transform.

The approach of Ref. [55] is based on a repeated pattern of fermionic swaps denoted U_L and U_R, where (unlike the definition in the main text of the present work) these occur along the "snake" ordering in the JW encoding (see Fig. 22). Using these, one is able to bring spin orbitals from adjacent rows next to each other in the canonical ordering so that the hopping term may be applied locally. First, one applies U_L. This will enable application of the first vertical hopping term that could not previously be reached. Then, one should repeatedly apply $U_L U_R$. After each application of $U_L U_R$, new vertical hopping terms become available until one has applied $U_L U_R$ a total of $\sqrt{N/8} - 1$ times. At that point, one needs to reverse the series of swaps until the orbitals are back to their original locations in the canonical ordering. After this, applying $U_L U_R$ will cause the qubits to circulate in the other direction. This should be repeated a total of $\sqrt{N/8} - 1$ times to make sure that all neighboring orbitals are adjacent at least once. The total number of layers of fermionic swaps required for the whole procedure is $3\sqrt{N/2}$.

To see how this swap network works for the FFT, we need to consider the structure of the 1D FFT circuit. If we use the approach from Jiang *et al.* [54] to implement the 1D FFT, then in the example of the 4×4 grid, there are two stages to the circuit: first we apply Givens rotations between all vertically adjacent qubits in the second and third rows, and then we apply Givens rotations between all vertically adjacent qubits in the first and second, and third and fourth rows. Since we have to apply gates from the first stage before we can apply gates from the second stage (to the same column), we can't take advantage of many of the local interactions made available during a single iteration of the swap network: we have to wait for every vertically adjacent qubit from the second and third rows to move to the local interaction zone and have a Givens rotation applied to them before we can take advantage of any of the other local interactions made available. In short, we lose the ability to parallelise, but gain something from not needing to implement the Z strings for every vertical interaction.

This problem becomes even worse for larger grid sizes, and dramatically worsens the scaling of the algorithm. Table V provides the depths required to implement the FFT using the above swap network approach. Clearly the depth scales as $O(N)$, compared to the scaling of $O(\sqrt{N})$ for the ancilla-based approach described in the previous section. However, the depth is superior for small grid sizes.

a. Modified swap network

It is possible to modify the approach from Ref. [55] to reduce the complexity even further, using the same approach taken for the implementation of VQE layers described in the main text. In this section we will be using U_R and U_L from Fig. 3(b). The basic idea is to repeatedly swap entire columns using parallel FSWAP gates, which eventually allows all vertical interactions to be implemented locally (with respect to the JW encoding).

To analyze the complexity of this method for implementing the vertical component of the FFT on grids of size $n_x \times n_y$, we can view the swap network as acting on a line (since it swaps entire columns in parallel). Our approach is to apply iterations composed of $n_x/2$ rounds of FSWAP operations, where we alternate between swapping odd-numbered columns with the columns to their right (the operator U_L), and swapping even-numbered columns with those to their right (the operator U_R). In this way, following the first iteration (i.e., after n_x rounds of FSWAP gates), all even-numbered columns have reached (at some point) the left-hand side of the grid, and all odd-numbered columns have reached the right-hand side. This allows us to apply the first round of the FFT on the odd-numbered columns.

After the second iteration, all odd-numbered columns reach the left-hand side, and all even-numbered columns reach the right-hand side. This allows us to apply the first round of the FFT on the even-numbered columns, and the second round of the FFT on the odd-numbered columns (in parallel). We can continue 'bouncing' the odd and even columns from left to right in this way until we have been able to apply all $n_y - 1$ rounds of the FFT to both sets of columns. This will require $n_y - 1$ iterations in total. Since each iteration is composed of $2n_x$ layers of swap operations, the total depth (for the vertical component) will be $2n_x(n_y - 1)$ (assuming that the Givens rotations can be implemented in depth 1). Table V provides the actual circuit depths for implementing the *full* (i.e., both horizontal and vertical terms) FFT for a number of different grid sizes.

If we let $T_{\mathcal{F}}(n)$ be the depth of the circuit that implements the 1D FFT on n qubits, then the depth of the circuit required to implement the 2D FFT using this swap-network approach will be

$$T_{\mathcal{F}}(n_x) + 2n_x T_{\mathcal{F}}(n_y).$$

4. Summary of approaches to implement the FFT

In the previous sections, we computed the depths of circuits required to implement the FFT using four approaches: a naïve implementation, an asymptotically optimal implementation due to Jiang *et al.* [54], and two swap-network approaches, one of which is due to Kivlichan *et al.* [55] and the other of which is a modification thereof.

The naïve approach is immediately seen to be prohibitively costly (in terms of circuit depth) even for smaller grid sizes. The implementation from Ref. [54], although asymptotically better, requires relatively high depth circuits for small grid sizes. In addition, the approach requires a number of ancilla qubits, which makes it a less attractive option for implementing the FFT on near-term architectures with few qubits. Finally, we modified a swap-network based approach from Ref. [55] to obtain an implementation of the FFT with low circuit depths for smaller grid sizes. The circuit depths for two of the more promising approaches, expressed in terms of the complexity $T_{\mathcal{F}}(n)$ of the 1D FFT on n qubits, and assuming an arbitrarily connected architecture, are (1) asymptotically efficient implementation (from Ref. [54]): $T_{\mathcal{F}}(n_x) + T_{\mathcal{F}}(n_y) + 2(8n_x + 2n_y - 2)$ and (2) modified swap-network approach: $T_{\mathcal{F}}(n_x) + 2n_x T_{\mathcal{F}}(n_y)$.[11]

Hence, if $(2n_x - 1)T_{\mathcal{F}}(n_y) < 2(8n_x + 2n_y - 2)$, the swap-network approach will be more efficient. For square lattices of fermions, this condition becomes $T_{\mathcal{F}}(n) < \frac{20n-4}{2n-1}$. For small n, it seems likely that this condition will be satisfied, and therefore the swap-network based approach will be more efficient. Indeed, if $T_{\mathcal{F}}(n) = n - 1$ (from the algorithm of Ref. [54]), this condition is satisfied for $n \leqslant 11$.

5. Complexity of preparing Slater determinants directly

Reference [54] describes an approach for preparing Slater determinants on an $n_x \times n_y$ lattice using a sequence of Givens rotations applied to a computational basis state. This work uses a freedom in the representation of Slater determinants, which allows fewer Givens rotations to be applied if the occupation number is known ahead of time.

The circuit derived from this approach runs in depth $n_x n_y - 1$, so is always more efficient than all the approaches discussed above, apart from the efficient FFT circuit in [54]. Compared with the algorithm of Ref. [54], the Slater determinant approach will be more efficient for small lattice sizes. Table V also lists the depths of the circuits required to prepare Slater determinants on various $n \times n$ lattices.

[11]It should be possible to absorb the horizontal component of the FFT [with cost $T_{\mathcal{F}}(n_x)$] into the FSWAP gates applied during the sorting network, which would reduce the depth of this approach to $2n_x T_{\mathcal{F}}(n_y)$.

[1] J. I. Cirac and P. Zoller, Goals and opportunities in quantum simulation, Nat. Phys. **8**, 264 (2012).

[2] I. M. Georgescu, S. Ashhab, and F. Nori, Quantum simulation, Rev. Mod. Phys. **86**, 153 (2014).

[3] J. Preskill, Quantum computing in the NISQ era and beyond, Quantum **2**, 79 (2018).

[4] S. Lloyd, Universal quantum simulators, Science **273**, 1073 (1996).

[5] A. M. Childs, D. Maslov, Y. Nam, N. J. Ross, and Y. Su, Toward the first quantum simulation with quantum speedup, Proc. Natl. Acad. Sci. USA **115**, 9456 (2018).

[6] R. Babbush, C. Gidney, D. W. Berry, N. Wiebe, J. McClean, A. Paler, A. Fowler, and H. Neven, Encoding Electronic Spectra in Quantum Circuits with Linear T Complexity, Phys. Rev. X **8**, 041015 (2018).

[7] Y. Nam and D. Maslov, Low-cost quantum circuits for classically intractable instances of the Hamiltonian dynamics simulation problem, npj Quantum Inf. **5**, 44 (2019).

[8] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, R. Biswas, S. Boixo, F. G. S. L. Brandao, D. A. Buell, B. Burkett, Y. Chen, Z. Chen, B. Chiaro, R. Collins, W. Courtney, A. Dunsworth, E. Farhi, B. Foxen, A. Fowler, C. Gidney, M. Giustina, R. Graff, K. Guerin, S. Habegger, M. P. Harrigan, M. J. Hartmann, A. Ho, M. Hoffmann, T. Huang, T. S. Humble, S. V. Isakov, E. Jeffrey, Z. Jiang, D. Kafri, K. Kechedzhi, J. Kelly, P. V. Klimov, S. Knysh, A. Korotkov, F. Kostritsa, D. Landhuis, M. Lindmark, E. Lucero, D. Lyakh, S. Mandrà, J. R. McClean, M. McEwen, A. Megrant, X. Mi, K. Michielsen, M. Mohseni, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Y. Niu, E. Ostby, A. Petukhov, J. C. Platt, C. Quintana, E. G. Rieffel, P. Roushan, N. C. Rubin, D. Sank, K. J. Satzinger, V. Smelyanskiy, K. J. Sung, M. D. Trevithick, A. Vainsencher, B. Villalonga, T. White, Z. J. Yao, P. Yeh, A. Zalcman, H. Neven, and J. M. Martinis, Quantum supremacy using a programmable superconducting processor, Nature **574**, 505 (2019).

[9] S. Gharibian, Y. Huang, Z. Landau, and S. W. Shin, Quantum Hamiltonian complexity, Found. Trends Theor. Comput. Sci. **10**, 159 (2015).

[10] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'Brien, A

variational eigenvalue solver on a photonic quantum processor, Nat. Commun. **5**, 4213 (2014).

[11] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, The theory of variational hybrid quantum-classical algorithms, New J. Phys. **18**, 023023 (2016).

[12] J. Hubbard, Electron correlations in narrow energy bands, Proc. R. Soc. London, Ser. A **276**, 238 (1963).

[13] Editorial, The Hubbard model at half a century, Nat. Phys. **9**, 523 (2013).

[14] D. Scalapino, Numerical studies of the 2D Hubbard model, in *Handbook of High-Temperature Superconductivity* (Springer, New York, NY,2007), pp. 495–526.

[15] J. P. F. LeBlanc, A. E. Antipov, F. Becca, I. W. Bulik, G. K.-L. Chan, C.-M. Chung, Y. Deng, M. Ferrero, T. M. Henderson, C. A. Jiménez-Hoyos, E. Kozik, X.-W. Liu, A. J. Millis, N. V. Prokof'ev, M. Qin, G. E. Scuseria, H. Shi, B. V. Svistunov, L. F. Tocchio, I. S. Tupitsyn, S. R. White, S. Zhang, B.-X. Zheng, Z. Zhu, and E. Gull, Solutions of the Two-Dimensional Hubbard Model: Benchmarks and Results from a Wide Range of Numerical Algorithms, Phys. Rev. X **5**, 041041 (2015).

[16] E. Dagotto, Correlated electrons in high-temperature superconductors, Rev. Mod. Phys. **66**, 763 (1994).

[17] D. Wecker, M. B. Hastings, and M. Troyer, Progress towards practical quantum variational algorithms, Phys. Rev. A **92**, 042303 (2015).

[18] S. Yamada, T. Imamura, and M. Machida, 16.447 Tflops and 159-billion-dimensional exact-diagonalization for trapped fermion-Hubbard model on the earth simulator, in *ACM/IEEE SC 2005 Conference* (IEEE, Seattle, WA, 2005).

[19] E. Altman, K. R. Brown, G. Carleo, L. D. Carr, E. A. Demler, C. Chin, B. Demarco, S. E. Economou, M. Eriksson, K.-M. C. Fu, M. Greiner, K. R. A. Hazzard, R. G. Hulet, A. J. Koll'ar, B. L. Lev, M. D. Lukin, R. Ma, X. Mi, S. Misra, C. Monroe, K. W. Murch, Z. Nazario, K.-K. Ni, A. C. Potter, P. Roushan, M. Saffman, M. Schleier-Smith, I. Siddiqi, R. W. Simmonds, M. Singh, I. B. Spielman, K. Temme, D. S. Weiss, J. Vuckovic, V. Vuletić, J. Ye, and M. W. Zwierlein, Quantum simulators: Architectures and opportunities, arXiv:1912.06938.

[20] T. Hensgens, T. Fujita, L. Janssen, X. Li, C. J. V. Diepen, C. Reichl, W. Wegscheider, S. D. Sarma, and L. M. K. Vandersypen, Quantum simulation of a Fermi-Hubbard model using a semiconductor quantum dot array, Nature **548**, 70 (2017).

[21] L. Tarruell and L. Sanchez-Palencia, Quantum simulation of the Hubbard model with ultracold fermions in optical lattices, C. R. Phys. **19**, 365 (2018).

[22] C. Gross and I. Bloch, Quantum simulations with ultracold atoms in optical lattices, Science **357**, 995 (2017).

[23] T. Esslinger, Fermi-Hubbard physics with atoms in an optical lattice, Annu. Rev. Condens. Matter Phys. **1**, 129 (2010).

[24] J.-M. Reiner, F. Wilhelm-Mauch, G. Schön, and M. Marthaler, Finding the ground state of the Hubbard model by variational methods on a quantum computer with gate errors, Quantum Sci. Technol. **4**, 035005 (2019).

[25] G. Verdon, M. Broughton, J. R. McClean, K. J. Sung, R. Babbush, Z. Jiang, H. Neven, and M. Mohseni, Learning to learn with quantum neural networks via classical neural networks, arXiv:1907.05415.

[26] P.-L. Dallaire-Demers, J. Romero, L. Veis, S. Sim, and A. Aspuru-Guzik, Low-depth circuit ansatz for preparing correlated fermionic states on a quantum computer, arXiv:1801.01053.

[27] R. Barends, J. Kelly, A. Megrant, A. Veitia, D. Sank, E. Jeffrey, T. C. White, J. Mutus, A. G. Fowler, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, C. Neill, P. O'Malley, P. Roushan, A. Vainsencher, J. Wenner, A. N. Korotkov, A. N. Cleland, and J. M. Martinis, Superconducting quantum circuits at the surface code threshold for fault tolerance, Nature **508**, 500 (2014).

[28] K. M. Nakanishi, K. Fujii, and S. Todo, Sequential minimal optimization for quantum-classical hybrid algorithms, Phys. Rev. Research **2**, 043158 (2020).

[29] M. Ostaszewski, E. Grant, and M. Benedetti, Quantum circuit structure learning, arXiv:1905.09692.

[30] R. M. Parrish, J. T. Iosue, A. Ozaeta, and P. L. McMahon, A Jacobi Diagonalization and Anderson Acceleration algorithm for variational quantum algorithm parameter optimization, arXiv:1904.03206.

[31] J. C. Spall, An overview of the simultaneous perturbation method for efficient optimization, Johns Hopkins APL Tech. Dig. **19**, 482 (1998).

[32] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta, Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets, Nature **549**, 242 (2017).

[33] M. Ganzhorn, D. Egger, P. Barkoutsos, P. Ollitrault, G. Salis, N. Moll, M. Roth, A. Fuhrer, P. Mueller, S. Woerner, I. Tavernelli, and S. Filipp, Gate-Efficient Simulation of Molecular Eigenstates on a Quantum Computer, Phys. Rev. Appl. **11**, 044092 (2019).

[34] P. O'Malley, R. Babbush, I. Kivlichan, J. Romero, J. McClean, R. Barends, J. Kelly, P. Roushan, A. Tranter, N. Ding, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. G. Fowler, E. Jeffrey, E. Lucero, A. Megrant, J. Mutus, M. Neeley, C. Neill, C. Quintana, D. Sank, A. Vainsencher, J. Wenner, T. White, P. Coveney, P. Love, H. Neven, A. Aspuru-Guzik, and J. Martinis, Scalable Quantum Simulation of Molecular Energies, Phys. Rev. X **6**, 031007 (2016).

[35] C. Hempel, C. Maier, J. Romero, J. McClean, T. Monz, H. Shen, P. Jurcevic, B. P. Lanyon, P. Love, R. Babbush, A. Aspuru-Guzik, R. Blatt, and C. F. Roos, Quantum Chemistry Calculations on a Trapped-Ion Quantum Simulator, Phys. Rev. X **8**, 031022 (2018).

[36] Y. Shen, X. Zhang, S. Zhang, J.-N. Zhang, M.-H. Yung, and K. Kim, Quantum implementation of the unitary coupled cluster for simulating molecular electronic structure, Phys. Rev. A **95**, 020501 (2017).

[37] Y. Nam, J.-S. Chen, N. C. Pisenti, K. Wright, C. Delaney, D. Maslov, K. R. Brown, S. Allen, J. M. Amini, J. Apisdorf, K. M. Beck, A. Blinov, V. Chaplin, M. Chmielewski, C. Collins, S. Debnath, A. M. Ducore, K. M. Hudek, M. Keesan, S. M. Kreikemeier, J. Mizrahi, P. Solomon, M. Williams, J. D. Wong-Campos, C. Monroe, and J. Kim, Ground-state energy estimation of the water molecule on a trapped ion quantum computer, arXiv:1902.10171.

[38] O. Shehab, K. A. Landsman, Y. Nam, D. Zhu, N. M. Linke, M. J. Keesan, R. C. Pooser, and C. R. Monroe, Toward convergence of effective field theory simulations on digital quantum computers, Phys. Rev. A **100**, 062319 (2019).

[39] N. Moll, P. Barkoutsos, L. S. Bishop, J. M. Chow, A. Cross, D. J. Egger, S. Filipp, A. Fuhrer, J. M. Gambetta, M. Ganzhorn, A. Kandala, A. Mezzacapo, P. Müller, W. Riess, G. Salis, J. Smolin, I. Tavernelli, and K. Temme, Quantum optimization using variational algorithms on near-term quantum devices, Quantum Sci. Technol. **3**, 030503 (2018).

[40] D. Wecker, B. Bauer, B. K. Clark, M. B. Hastings, and M. Troyer, Gate-count estimates for performing quantum chemistry on small quantum computers, Phys. Rev. A **90**, 022305 (2014).

[41] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall, An adaptive variational algorithm for exact molecular simulations on a quantum computer, Nat. Commun. **10**, 3007 (2019).

[42] B. T. Gard, L. Zhu, G. S. Barron, N. J. Mayhall, S. E. Economou, and E. Barnes, Efficient symmetry-preserving state preparation circuits for the variational quantum eigensolver algorithm, npj Quantum Inf. **6**, 10 (2020).

[43] J. Lee, W. J. Huggins, M. Head-Gordon, and K. B. Whaley, Generalized unitary coupled cluster wave functions for quantum computation, J. Chem. Theory Comput. **15**, 311 (2018).

[44] P. K. Barkoutsos, J. F. Gonthier, I. Sokolov, N. Moll, G. Salis, A. Fuhrer, M. Ganzhorn, D. J. Egger, M. Troyer, A. Mezzacapo, S. Filipp, and I. Tavernelli, Quantum algorithms for electronic structure calculations: Particle-hole hamiltonian and optimized wave-function expansions, Phys. Rev. A, **98**, 022322 (2018).

[45] J. Romero, R. Babbush, J. R. McClean, C. Hempel, P. J. Love, and A. Aspuru-Guzik, Strategies for quantum computing molecular energies using the unitary coupled cluster ansatz, Quantum Sci. Technol. **4**, 014008 (2018).

[46] M. Wilson, S. Stromswold, F. Wudarski, S. Hadfield, N. M. Tubman, and E. Rieffel, Optimizing quantum heuristics with meta-learning, arXiv:1908.03185.

[47] P.-L. Dallaire-Demers and F. K. Wilhelm, Quantum gates and architecture for the quantum simulation of the Fermi-Hubbard model, Phys. Rev. A **94**, 062304 (2016).

[48] P.-L. Dallaire-Demers and F. K. Wilhelm, Method to efficiently simulate the thermodynamic properties of the Fermi-Hubbard model on a quantum computer, Phys. Rev. A **93**, 032303 (2016).

[49] J.-M. Reiner, S. Zanker, I. Schwenk, J. Leppäkangas, F. Wilhelm-Mauch, G. Schön, and M. Marthaler, Effects of gate errors in digital quantum simulations of fermionic systems, Quantum Sci. Technol. **3**, 045008 (2018).

[50] S. B. Bravyi and A. Y. Kitaev, Fermionic quantum computation, Ann. Phys. **298**, 210 (2002).

[51] R. C. Ball, Fermions without Fermion Fields, Phys. Rev. Lett. **95**, 176407 (2005).

[52] F. Verstraete and J. I. Cirac, Mapping local Hamiltonians of fermions to local Hamiltonians of spins, J. Stat. Mech. (2005) P09012.

[53] F. Verstraete, J. I. Cirac, and J. I. Latorre, Quantum circuits for strongly correlated quantum systems, Phys. Rev. A **79**, 032316 (2009).

[54] Z. Jiang, K. J. Sung, K. Kechedzhi, V. N. Smelyanskiy, and S. Boixo, Quantum Algorithms to Simulate Many-Body Physics of Correlated Fermions, Phys. Rev. Appl. **9**, 044036 (2018).

[55] I. D. Kivlichan, J. McClean, N. Wiebe, C. Gidney, A. Aspuru-Guzik, G. K.-L. Chan, and R. Babbush, Quantum Simulation of Electronic Structure with Linear Depth and Connectivity, Phys. Rev. Lett. **120**, 110501 (2018).

[56] M. Vekić and S. R. White, Hubbard model with smooth boundary conditions, Phys. Rev. B **53**, 14552 (1996).

[57] W. J. Huggins, J. McClean, N. Rubin, Z. Jiang, N. Wiebe, K. B. Whaley, and R. Babbush, Efficient and noise resilient measurements for quantum chemistry on near-term quantum computers, arXiv:1907.13117.

[58] O. Crawford, B. van Straaten, D. Wang, T. Parks, E. Campbell, and S. Brierley, Efficient quantum measurement of pauli operators, arXiv:1908.06942.

[59] P. Gokhale and F. T. Chong, $O(N^3)$, Measurement cost for variational quantum eigensolver on molecular Hamiltonians, arXiv:1908.11857.

[60] A. F. Izmaylov, T.-C. Yen, and I. G. Ryabinkin, Revising the measurement process in the variational quantum eigensolver: Is it possible to reduce the number of separately measured operators? Chem. Sci. **10**, 3746 (2019).

[61] P. Gokhale, O. Angiuli, Y. Ding, K. Gui, T. Tomesh, M. Suchara, M. Martonosi, and F. T. Chong, Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families, arXiv:1907.13623.

[62] Z. Cai, Resource estimation for quantum variational simulations of the Hubbard model: The advantage of multi-core NISQ processing, Phys. Rev. Appl. **14**, 014059 (2020).

[63] T. Walter, P. Kurpiers, S. Gasparinetti, P. Magnard, A. Potočnik, Y. Salathé, M. Pechal, M. Mondal, M. Oppliger, C. Eichler, and A. Wallraff, Rapid High-Fidelity Single-Shot Dispersive Readout of Superconducting Qubits, Phys. Rev. Appl. **7**, 054020 (2017).

[64] S. G. Johnson, The NLopt nonlinear-optimization package, http://github.com/stevengj/nlopt.

[65] J. Kübler, A. Arrasmith, L. Cincio, and P. Coles, An adaptive optimizer for measurement-frugal variational algorithms, Quantum **4**, 263 (2020).

[66] J. Spall, Implementation of the simultaneous perturbation algorithm for stochastic optimization, IEEE Trans. Aerosp. Electron. Syst. **34**, 817 (1998).

[67] P. Tseng, Convergence of a block coordinate descent method for nondifferentiable minimization, J. Optim. Theory Appl. **109**, 475 (2001).

[68] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, QuEST and high performance simulation of quantum computers, Sci. Rep. **9**, 10736 (2019).

[69] A. Montanaro, C. Cade, L. Mineh, and S. Stanisic, Data from "Strategies for solving the Fermi-Hubbard model on near-term quantum computers" (2020), https://doi.org/10.5523/bris.1873owc1bcmrw1y4raeeuygzuy.

[70] J. D. Whitfield, V. Havlíček, and M. Troyer, Local spin operators for fermion simulations, Phys. Rev. A **94**, 030301 (2016).

[71] V. Havlíček, M. Troyer, and J. D. Whitfield, Operator locality in the quantum simulation of fermionic models, Phys. Rev. A **95**, 032332 (2017).

[72] Craig Gidney (private communication, 2019).