Fully quantum algorithm for mesoscale fluid simulations with application to partial differential equations ⊘

Sriharsha Kocherla (ID) ; Zhixin Song (ID) ; Fatima Ezahra Chrit (ID) ; Bryan Gard (ID) ; Eugene F. Dumitrescu (ID) ; Alexander Alexeev (ID) ; Spencer H. Bryngelson ✉ (ID)

View Online

Export Citation

Fully quantum algorithm for mesoscale fluid simulations with application to partial differential equations

View Online   Export Citation   CrossMark

Sriharsha Kocherla,[1] (ID) Zhixin Song,[2] (ID) Fatima Ezahra Chrit,[1,3] (ID) Bryan Gard,[4] (ID) Eugene F. Dumitrescu,[5] (ID) Alexander Alexeev,[3] (ID) and Spencer H. Bryngelson[1,3,6,a)] (ID)

AFFILIATIONS

[1]School of Computational Science and Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
[2]School of Physics, Georgia Institute of Technology, Atlanta, Georgia 30332
[3]George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332
[4]CIPHER, Georgia Tech Research Institute, Atlanta, Georgia 30332
[5]Oak Ridge National Laboratory, Oak Ridge, Tennessee 37830
[6]Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, Georgia 30332

[a)]Author to whom correspondence should be addressed: shb@gatech.edu

ABSTRACT

Fluid flow simulations marshal our most powerful computational resources. In many cases, even this is not enough. Quantum computers provide an opportunity to speed up traditional algorithms for flow simulations. We show that lattice-based mesoscale numerical methods can be executed as efficient quantum algorithms due to their statistical features. This approach revises a quantum algorithm for lattice gas automata to reduce classical computations and state preparation at every time step. For this, the algorithm approximates the qubit relative phases and subtracts them at the end of each time step. Phases are evaluated using the iterative phase estimation algorithm and subtracted using single-qubit rotation phase gates. This method optimizes the quantum resource required and makes it more appropriate for near-term quantum hardware. We also demonstrate how the checkerboard deficiency that the D1Q2 scheme presents can be resolved using the D1Q3 scheme. The algorithm is validated by simulating two canonical partial differential equations: the diffusion and Burgers' equations on different quantum simulators. We find good agreement between quantum simulations and classical solutions for the presented algorithm.

Published under an exclusive license by AIP Publishing. https://doi.org/10.1116/5.0217675

I. INTRODUCTION

Solving partial differential equations (PDEs) is necessary for modeling most scientific problems. However, it is computationally expensive, especially for large domains and high-dimensional configuration spaces. For example, in computational fluid dynamics, the cost of performing a direct numerical simulation of turbulence increases with Reynolds number, representing the ratio of inertial to viscous effects in the flow, as $\mathcal{O}(\mathrm{Re}^3)$.[1] Such high computational expense motivates a broad category of work to achieve algorithmic speedups for PDE solvers.

Quantum computers are emerging as an increasingly viable tool for algorithm speedup enabled by the principles of quantum mechanics, such as superposition and entanglement. In some cases, exponential speedups over classical computer architectures (like that of von Neumann) are possible.[2,3] In the sciences, algorithmic complexity

improvements are seen in quantum chemistry,[4] machine learning,[5–7] finance,[8,9] and more.

Quantum algorithms for solving PDEs have also gained attention. Harrow *et al.*[10] demonstrated the first quantum algorithm, HHL, for solving linear systems of equations with exponential speedup. Such improvements can be marshaled in the PDE space for solving discretized systems of equations via, for example, the method of lines. Childs and Liu[11] proposed a quantum algorithm for linear ordinary differential equations (ODEs) based on spectral methods, providing a global approximation to the solution using linear combinations of basis functions. Childs *et al.*[12] improved the complexity of quantum algorithms for linear PDEs using adaptive-order finite difference and spectral methods. Other approaches to solving PDEs were proposed by Berry;[13] for example, reducing an ODE system via discretization and solving it via the appropriate quantum algorithms. Gaitan[14] used this

approach to solve the 1D Navier–Stokes equations for the flow through a convergent–divergent nozzle. However, these methods require deep circuits and many quantum gates to achieve results comparable to what one can solve on even a modern laptop. This limitation is prohibitive for most of the algorithms mentioned above but paints a bright future for larger, fault-tolerant quantum devices that can better reject gate noise.

In the near term, variational quantum algorithms are well-suited for current NISQ (noisy intermediate-scale quantum) devices. For example, hybrid quantum–classical procedures can evaluate the solution quality via a cost function using a quantum computer and optimize variational parameters using a classical computer. Thus, variation methods enable quantum algorithms with relatively shallow gate depths and qubit counts. Bravo-Prieto et al.[15] introduced the variational quantum linear solver (VQLS), which uses the Hadamard test to solve a linear system. Liu et al.[16] showed that a Poisson problem can be solved by first converting it to a linear system amenable to the quantum alternating operator ansatz (QAOA) algorithm. However, variation algorithms require a (sometimes) unclear ansatz and do not guarantee convergence or speedup. The continued tension between these quantum algorithm threads has contributed to improving each.

Because of the linearity of quantum mechanics, quantum algorithms aiming to solve nonlinear PDEs are immature compared to the linear case. Liu et al.[17] proposed the first quantum algorithm for dissipative nonlinear differential equations using the method of Carleman linearization, which maps the system of nonlinear equations to an infinite-dimensional system of linear differential equations, which are then solved using quantum linear system algorithms. Their algorithm was tested to solve the 1D Burgers' equation. Kyriienko et al.[18] used the variational method based on differentiable quantum circuits to solve nonlinear differential equations. Their method solved a quasi-1D approximation of Navier–Stokes equations. Lubasch et al.[19] used multiple copies of variational quantum states and tensor networks to solve nonlinear PDEs.

Other promising numerical methods include mesoscale strategies, which operate between molecular and continuum scales. Lattice methods are a common example and are well-suited for quantum computation because they are intrinsically statistical, resolving only samples or probabilities of the fictitious particles they comprise. Moreover, they are based on simple mathematical calculations and are suitable for parallel computation because interactions between lattice nodes are linear, and nonlinearity enters during a local collision step. The lattice Boltzmann method (LBM) is a common approach that solves the Boltzmann transport equation. Mezzacapo et al.[20] developed the first quantum simulator following a lattice kinetic formalism to solve fluid dynamics transport phenomena. Todorova and Steijl[21] solved the collisionless Boltzmann equation on a quantum computer, inspired by quantum algorithms solving the Dirac equation. Budinski[22] proposed a novel quantum algorithm that solves the advection–diffusion equation by the LBM and extended it to solve the two-dimensional (2D) Navier–Stokes equations using stream function–vorticity formulation.[23] This technique has a logarithmic scaling of qubits relative to lattice grid size. Itani and Succi[24] explored the Carleman linearization of the collision term of the lattice Boltzmann equation. Using this linearization technique, Itani et al.[25] proposed a quantum lattice Boltzmann algorithm where collision and streaming are achieved by unitary evolution. Schalkers and Möller[26] propose a space-time encoding method to

implement unitary collision and streaming operators without linearization.

The LBM originates from the lattice gas automata (LGA), a cellular automaton that can simulate fluid flows. A quantum computer can be viewed as a quantum cellular automaton (QCA), where each cell is a quantum system with a state depending on neighboring cells. Quantum LGA is a subclass of QCA that represents particles interacting under physical constraints and evolving on a lattice. The main drawback of the LGA is statistical noise, though this vanishes when crafted for mesoscopic scales. Yepez[27] introduced the first quantum LGA (QLGA) algorithm for simulating fluid flow at the mesoscopic scale using a discretized Boltzmann transport equation.

The original algorithm was designed for a specialized type-II quantum computer, proposed in the early 2000s.[28] Such computers consisted of several small quantum computers connected by classical communication channels, allowing qubit shifts during the algorithm's streaming step.[28] Type II architectures no longer exist. Instead, we present an algorithm more suitable for current architectures that avoids some assumed-efficient operations on previous Type II devices. For example, many QLGA/LBM algorithms require estimating the distribution functions needed to re-prepare the quantum states at each time step. This process can be prohibitively expensive on current, and likely future, quantum devices.

Herein, we present an approach that overcomes the limitations of previous lattice-based quantum algorithms shown by Yepez.[28,29] We focus on improving the efficiency and scalability of previous QLGA techniques. The cost of general quantum state preparation scales exponentially with respect to the qubit count.[30] This expensive step, assumed in previous quantum lattice-based methods using amplitude encoding, presents scaling issues for utility-scale simulations. Our work seeks to mitigate these expense and scalability issues. The proposed algorithm uses a quantum implementation of the streaming step, approximates the qubit relative phases, and subtracts them at the end of each time step, avoiding repeated encoding operations and delaying the need for measurement to the end of the computation.

We do not interrogate the practicality of the presented algorithmic strategies on actual quantum devices but rather on simulators. The algorithm presented herein is not appropriate for solving practical fluid dynamics problems, in full, on current quantum devices, for which there is perhaps no known algorithm at the time of writing. Instead, simulators help achieve validation and facilitate analysis. Specifically, we implement our algorithm in Qiskit[31] and XACC for the diffusion and Burgers' equations.

This manuscript continues as follows. Section II describes the classical lattice gas automata and lattice Boltzmann method. Section III introduces the quantum counterpart of the classical LGA. Section IV presents the QLGA variants we propose, including quantum streaming, the D1Q3 scheme, and a restless quantum lattice Boltzmann method (QLBM) algorithm. We validate these variants by testing them to solve the 1D diffusion and Burgers' equations. Last, Sec. V compares different quantum simulation strategies for the proposed algorithm using Qiskit and XACC.

II. CLASSICAL LATTICE METHODS

The lattice gas automata (LGA) method is a cellular automaton that simulates fluids.[32,33] The LGA state is described by *occupation numbers*: Boolean variables indicating whether or not a fictitious particle is present at a specific lattice node, moving in a certain lattice
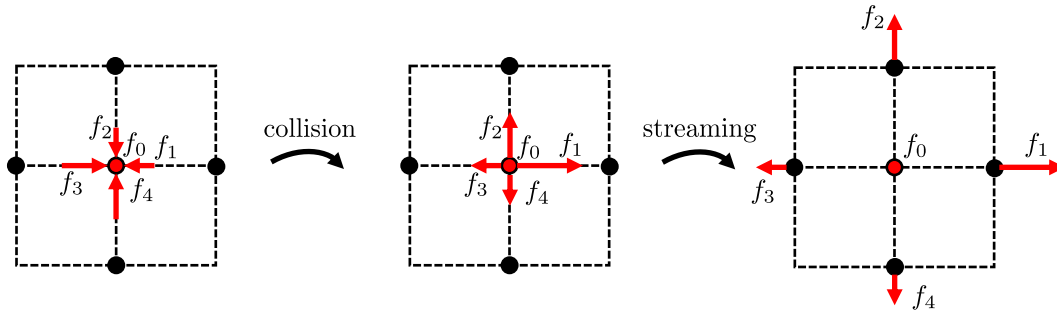
FIG. 1. Illustration of collision and streaming steps for a single time step of a D2Q5 scheme.

direction at a specific time. These fictitious particles obey local collision rules that conserve mass and momentum.

The main drawback of LGA is noise, which can be reduced via coarse-graining over large domains or time intervals but remains costly in computing time and memory. Instead, LGA can be directly modeled at the mesoscopic scale with the Boltzmann equation.[34,35] This approach is called the lattice Boltzmann method (LBM), which eliminates the LGA noise by replacing Boolean variables with continuous distributions, i.e., the mean occupation numbers, also called density distribution functions. We take this approach herein, which is thus a QLBM-type technique. The LBM approach replaces the collision rule with a continuous function called the collision operator. The distribution functions f_α ($0 \leq f_\alpha \leq 1$) evolve in time according to the kinetic equation,

$$f_\alpha(\boldsymbol{r} + \boldsymbol{c}_\alpha \Delta t, t + \Delta t) = f_\alpha(\boldsymbol{r}, t) + \Omega_\alpha[f_\alpha(\boldsymbol{r}, t)], \qquad (1)$$

where $\boldsymbol{r}$ is the position of a lattice site, $\boldsymbol{c}_\alpha$ is the lattice velocity, Δt is the time step, t is time, Ω_α is the collision operator, and α refers to the velocity discretization index. During one time step, as illustrated in Fig. 1, the computation evolves through two processes: collision, where particles meet at the same lattice site, and their distribution functions are evaluated according to the collision operator Ω_α, and streaming, where particles shift to the neighboring sites following their lattice directions. Collision causes the relaxation of all local distribution functions to an equilibrium where $\Omega_\alpha[f_\alpha = f_\alpha^{\mathrm{eq}}] = 0$.

Each discretization of the velocity space is described by its number of spatial dimensions a and velocity directions b, with scheme notation DaQb. For example, a D1Q2 scheme uses a 1D lattice with two velocity directions, i.e., a right-going particle and a left-going particle for each lattice site.

Macroscopic variables, like the fluid density or velocity, are determined from the moments of the distribution functions.[36] For example, the density at lattice site i corresponds to

$$\rho_i = \sum_{\alpha=1}^{b} f_{\alpha,i}. \qquad (2)$$

Herein, we divide 1D spatial domains into N_s uniformly spaced lattice sites $i = 0, \ldots, N_s - 1$. For example, domain $X \in [0, L]$ has site locations $x_i = i\Delta x$ where $\Delta x = L/N_s$ is the lattice spacing, usually set to 1.

III. QUANTUM LATTICE METHODS

The quantum LGA (QLGA) is a measurement-based algorithm that introduces a superposition of qubit states within a small spatial region and for a short period. It directly maps the bits used in a classical lattice gas to qubits and models the lattice gas at the mesoscopic scale, which can be described using the lattice Boltzmann equation. Figure 2 shows an example of the associated quantum circuit used to solve the 1D diffusion equation using the D1Q2 scheme defined in Sec. II. The algorithm has four operations: (re)initializing the qubit states via *encoding*, *collision* at each lattice site, measuring the qubit occupancy via *localization*, and *streaming* the qubits on the lattice appropriately.

Encoding and collision are performed in the quantum space, but the measurement step and classical streaming computation mean the quantum state must be re-encoded. The encoding step in QLGA[27] scales linearly in the number of qubits required for encoding with respect to the size of the simulation. There are QLBM methods[22,23] that scale logarithmically with respect to the size of the grid. However, these entail a prohibitively large number of quantum gates (in many cases more than 10^5 for even a small, simplified problem) a trade-off that may be unappealing.

The system wavefunction can be written as

$$|\Psi(x_0, x_1, \ldots, x_{N_s-1}, t)\rangle = \bigotimes_{i=0}^{N_s-1} |\Psi(x_i, t)\rangle. \qquad (3)$$

Following Dirac's bra-ket notation, the ket vector $|\Psi(x_i, t)\rangle$ represents the on-site state, i.e., at the lattice site of coordinate x_i, of dimension 2^b. In the D1Q2 scheme, each lattice site comprises two particles, one moving to the right and the other to the left. The first step of the QLGA algorithm is mapping each particle to one qubit by encoding the occupancy probability $f_\alpha(x_i, t)$ to αth qubit as

$$|q_\alpha(x_i, t)\rangle = \sqrt{1 - f_\alpha(x_i, t)}|0\rangle + \sqrt{f_\alpha(x_i, t)}|1\rangle$$
$$\text{for} \quad i = 0, \ldots, N_s - 1 \quad \text{and} \quad \alpha = 1, 2. \qquad (4)$$
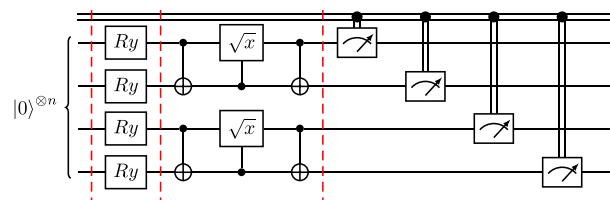

FIG. 2. QLGA with classical streaming (Ref. 27). R_y is a rotation gate by the angle $\theta_\alpha = 2\arccos\sqrt{1 - f_\alpha(x_i, t)}$ for the qubit of state $|q_\alpha(x_i, t)\rangle$. The controlled $\sqrt{x}$ gate can be decomposed efficiently on the latest IBM Heron and Eagle processors, which consists of a two-qubit controlled phase rotation $P(\pi/2)$ gate between two Hadamard gates on the target qubit.

This step is implemented via the $R_y(\theta_\alpha)$ rotation gate with angle $\theta_\alpha = 2\arccos\sqrt{1 - f_\alpha(x_i, t)}$ for the qubit of state $|q_\alpha(x_i, t)\rangle$. Then, the full quantum state at each lattice site x_i is the tensor product state of two qubits,

$$|\Psi(x_i, t)\rangle = |q_1(x_i, t)\rangle \otimes |q_2(x_i, t)\rangle \quad \text{for} \quad i = 0, \ldots, N_s - 1. \quad (5)$$

The relative phase of the qubits is set to zero in this step. Then, QLGA applies the collision locally to each lattice site, a unitary operation that relaxes the distribution functions to equilibrium. The postcollision state is

$$|\Psi'(x_i, t)\rangle = U|\Psi(x_i, t)\rangle, \quad (6)$$

where U is the collision operator that causes local quantum entanglement of the on-site qubits. It is implemented via a unitary gate with a block-diagonal entangling $U(2)$ matrix as follows:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & e^{i\phi}e^{i\xi}\cos\theta & e^{i\phi}e^{i\zeta}\sin\theta & 0 \\ 0 & -e^{i\phi}e^{-i\zeta}\sin\theta & e^{i\phi}e^{-i\xi}\cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad (7)$$

where ϕ, ξ, ζ, and θ are Euler angles chosen to satisfy the PDE of interest in the continuum limit. We choose $\phi = -\pi/4$, $\xi = 0$, $\zeta = \pi/2$, and $\theta = \pi/4$ to model the diffusion equation and $\phi = 0$, $\xi = \zeta$, and $\theta = \pi/4$ to model Burgers' equation.[29,37]

The next step is measuring all the $2N_s$ qubits associated with N_s lattice sites. Notice the local nature of these measurements. The algorithm starts with a product state. The encoding (re-preparation) and collision steps are local operations inside each lattice site and, hence, do not change the structure of the product state. Thus, the number of measurements required in each time step scales linearly with the number of lattice sites. Then, the postcollision occupancy probabilities $f'_\alpha(x_i, t)$ are computed from measurement outcomes. The nonunitary measurement operation destroys the quantum superposition and entanglement that the collision step may have caused. The postcollision occupancy probability is

$$f'_\alpha(x_i, t) = \langle \Psi'(x_i, t)|n_\alpha|\Psi'(x_i, t)\rangle, \quad (8)$$

where n_α are the number operators (observables that can be measured to count the number of particles) for $\alpha = 1, 2$.

Streaming is performed by shifting the qubits to their neighboring positions according to their right ($\alpha = 1$) or left ($\alpha = 2$) lattice directions. The poststreaming occupancy probabilities can be written as

$$f_\alpha(x_i, t + \Delta t) = f'_\alpha(x_i + e_\alpha \Delta x, t), \quad (9)$$

where $e_1 = -1$ and $e_2 = 1$ are the qubit directions, Δx is the spacing, and Δt is the time step.

These four operations can be encapsulated in the following equation:

$$f_\alpha(x_i + e_\alpha \Delta x, t + \Delta t) = f_\alpha(x_i, t) + \Omega_\alpha[f_\alpha(x_i, t)], \quad (10)$$

which is in the form of the classical lattice Boltzmann equation with the collision operator given by

$$\Omega_\alpha[f_\alpha(x_i, t)] = \langle \Psi(x_i, t)|(U^\dagger n_\alpha U - n_\alpha)|\Psi(x_i, t)\rangle. \quad (11)$$

Equilibrium distributions can be found by satisfying the equilibrium condition: $\Omega_\alpha[f_\alpha = f_\alpha^{eq}] = 0$. For the diffusion equation, we get equal equilibrium probabilities $f_\alpha^{eq} = \rho/2$ for $\alpha = 1, 2$.[37] For Burgers' equation, the equilibrium distribution corresponds to $f_\alpha^{eq} = \rho/2 + e_\alpha/2\left(\sqrt{2} - \sqrt{1 + (\rho - 1)^2}\right)$.[29] The equilibrium distributions for diffusion and Burgers' equations differ due to the inherent differences in the underlying physical processes. The equilibrium state of the diffusion equation is characterized by maximum entropy and minimum energy. This equation results in a smooth, uniform distribution. Meanwhile, the equilibrium state of Burgers' equation is characterized by the balance of convective and diffusive forces.

This streaming step was designed to run classically on type-II quantum computers.[28] One time step of QLGA consists of the four operations described above. Unlike most quantum gate models, where measurement is performed at the end of the computation, QLGA requires measurement at each time step to extract the distribution functions, which are then used to encode the next step. Next, we present a method for improving these unnecessary computations that is more appropriate for modern quantum hardware.

IV. A FULLY QUANTUM LATTICE BOLTZMANN ALGORITHM

This section presents a single-quantum computer version of QLGA that avoids the type-II quantum computers of previous studies. We formulate a fully quantum lattice Boltzmann method (QLBM) algorithm and verify it against known, classical solutions. Since our techniques solve the lattice Boltzmann equation at the mesoscopic scale using density distributions instead of Boolean operators, we denote our QLGA variants as QLBM.

A. Quantum streaming

The first revision to the QLGA algorithm performs the streaming step using quantum operations, not through classical postprocessing. It can be implemented before the measurement step via the permutation function that applies a sequence of SWAP gates to shift the qubits to their neighbors. The corresponding quantum circuit is shown in Fig. 3 for the diffusion and Burgers' equations.

We validate the proposed variant by solving the 1D diffusion and Burgers' equations using Gaussian and sinusoidal distributions as the initial condition, respectively. Periodic boundary conditions are used. We compare the two QLBM solutions, i.e., with classical streaming and quantum streaming, with the classical solution using the D1Q2 scheme and 26 simulated qubits. The simulations were performed using a Qiskit AerSimulator with the statevector simulation method, and the counts were sampled from the statevector using 10^5 shots. The spatial domain consists of $N_s = 13$ lattice sites corresponding to 26 qubits. Results are shown after ten time steps.

The results of Fig. 4 demonstrate good agreement with the classical solution. This variant removes the need for classical communication between lattice sites and does not cause global entanglement, as SWAP gates merely interchange the qubit quantum states. Since the quantum streaming does not change the local nature of product states, the total number of measurements required in each time step still scales linearly with the number of lattice sites.

In terms of complexity, we examine the relationship between the circuit depth and the system size by analyzing the different operations
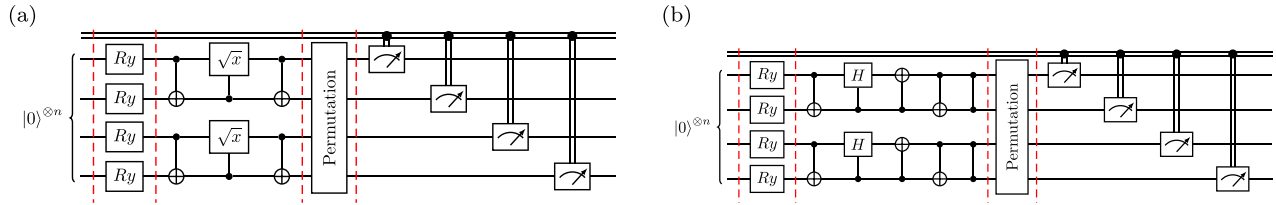
FIG. 3. QLBM with quantum streaming solving (a) the diffusion equation and (b) Burgers' equation. The permutation block encapsulates a sequence of SWAP gates that shift the qubits during the streaming step.

of the algorithm separately. The encoding step consists of applying one rotation gate simultaneously to each qubit. Thus, changing the grid resolution and the qubit number does not affect the circuit depth. The same unitary operator is applied to each pair of qubits for the collision step. Thus, the circuit depth scales as $\mathcal{O}(1)$ with the number of qubits n. For the streaming step implemented using the permutation function, the number of SWAP gates applied to one of the qubits increases linearly with the number of qubits. Therefore, the complexity of the algorithm is given by $\mathcal{O}(n)$. A D1Q3 scheme can also be implemented by using an extra qubit to create a smoother curve (Fig. 9), as shown in Appendix A. For a D1Q2 problem with a point source, particles do not remain in the original lattice site but propagate through the lattices over time. Therefore, a D1Q3 lattice scheme produces a smoother curve due to the nonzero distribution at one lattice site.

B. Restless QLBM

An issue with QLGA is that the quantum state collapses due to measurements performed during each time step, requiring re-encoding. However, we should allow the qubits to explore more complex states to achieve truly quantum computation. Furthermore, measurement on superconducting quantum computers can typically take an order of magnitude longer than gate times, adding significant time to the algorithm.[38] Thus, the goal is to eliminate repeated initialization of the quantum state and repeated measurements.

We attempted a unitary transformation using random rotations to reset the qubit relative phases after each time step. However, this

random phase kicking leads to dephasing, where the expected value of the off-diagonal elements of the density matrix decay to zero with time.[30] Therefore, a different approach is needed.

To this end, our algorithm adds a phase correction step, approximating the qubit relative phases and subtracting them at the end of each time step, delaying the need for measurement until the end of the computation. Phase correction is achieved via quantum phase estimation (QPE) as a sub-routine to estimate the eigen-phases of the encoding and collision unitaries.[30] These eigen-phases are then used to classically estimate the qubit relative phases, subtracted at the end of each time step. The accuracy of the QPE depends on the number of ancillary qubits it uses. Thus, we use the iterative phase estimation (IPE) algorithm, which only requires a single auxiliary qubit and evaluates the phase bit by bit through a repetitive process.[39] The accuracy of the IPE algorithm is determined by the number of iterations instead of the number of ancillary qubits. Therefore, they are of great importance for near-term quantum computing. The IPE algorithm still requires measurement to estimate the phase. However, only one qubit, i.e., the auxiliary qubit, is measured, and not all qubits are in the quantum register.

Although the phase estimation algorithm could help avoid measurements at the end of each time step and re-preparation in the next time step, it introduces extra $O(n)$ mid-circuit measurements in each time step. This overhead balances off the save for measurement. What is left is the resource reduction for state re-preparation with $O(T)$ circuit depth, where T is the number of time steps.

In the context of QLBM, we apply the IPE algorithm to every lattice site, i.e., to a two-qubit unitary for the D1Q2 scheme. We set the
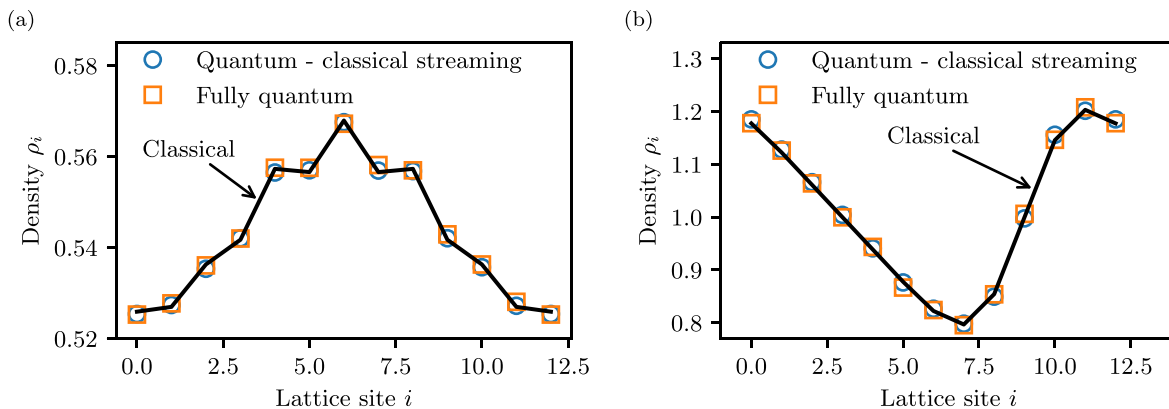

FIG. 4. QLBM solutions to (a) the diffusion equation and (b) Burgers' equation using the D1Q2 scheme. "Quantum—classical streaming" corresponds to the classical streaming algorithm discussed in previous work (Ref. 27), "Fully quantum" refers to the present algorithm using direct measurements, and "Classical" is the classical implementation of the lattice method. The small differences between Quantum—classical streaming and Fully quantum are due to finite sampling in quantum measurement.
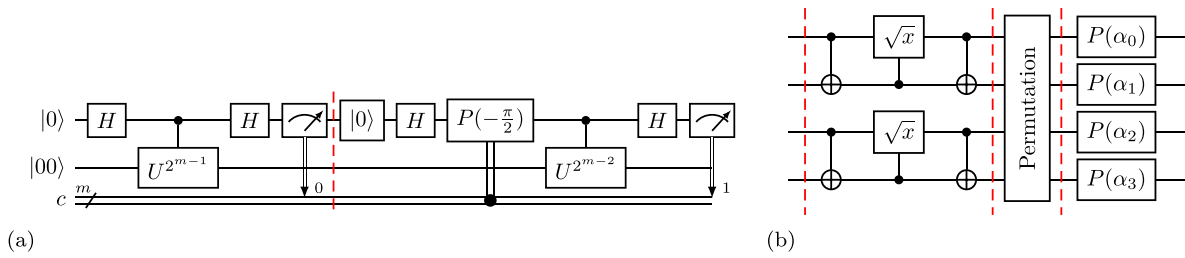
FIG. 5. (a) IPE circuit used to estimate the qubits relative phase α_i (first two iterations) and (b) restless QLBM circuit solving the diffusion equation.

unitary U to be the product of the encoding and the collision operations since the streaming step only swaps the qubits and does not change their phases. For a total number of iterations m, the IPE algorithm estimates the best m bit approximation to ϕ with $\varphi = 0.\varphi_1...\varphi_m = \sum_{k=1}^{m} \varphi_k/2^k$, where φ_k are the phase bits ($1 \leq k \leq m$) in binary expansion. At each iteration, k, the IPE algorithm applies the unitary U 2^{m-k} times controlled on the auxiliary qubit. Due to phase kickback, the relative phase of the auxiliary qubit becomes $\exp(i2\pi 0.\varphi_{m-k+1}\varphi_{m-k+2}...\varphi_m)$. Thus, to estimate the phase bit φ_{m-k+1}, a phase correction of $-2\pi(\varphi_{m-k+2}/2^2 + \cdots + \varphi_m/2^m)$ is performed, controlled on the measurement outcome of the previously estimated phase bits $\varphi_{m-k+2}, ..., \varphi_m$. The phase bit φ_{m-k+1} is then estimated by measuring the auxiliary qubit in the x-basis. The first two iterations of the IPE algorithm, which estimates the two least significant bits of the phase φ_m and φ_{m-1}, is illustrated in Fig. 5(a). One time step of the restless QLBM circuit is shown in the circuit of Fig. 5(b), which consists of collision, streaming, and phase correction operations.

Figure 6 presents the numerical simulation of our variant using the IPE algorithm to solve the 1D diffusion equation. The simulation uses the D1Q2 scheme with 26 qubits and runs on the statevector simulator. The spatial domain consists of $N_s = 13$ lattice sites, and results are shown after six time steps. The quantum solution and the classical one agree well, with the root-mean-square error between them below the finite sampling threshold. This variant improves the efficiency of the QLGA algorithm as the encoding is used only once at the beginning of the computation to prepare the quantum states, and measurement is only performed at the end of the computation.

The number of iterations m is determined by the desired accuracy, here $1/2^q$, with a probably of success of at least p, via

$$m = q + \left\lceil \log_2\left(2 + \frac{1}{2(1-p)}\right) \right\rceil. \tag{12}$$

The difference between the estimated phase φ and the actual phase ϕ satisfies $0 \leq \Delta\varphi \leq 1/2^q$, where $\Delta\varphi = \phi - \varphi$. The probability of success p quantifies the likelihood one measures the best m bit approximation φ.[30] Such a probability is lower bounded by $4/\pi^2 \approx 0.405$ but can be problematic in larger qubit cases. Since the IPE algorithm only operates on each lattice site, running locally on each site requires $b + 1$ qubits. Hence, the desired accuracy $1/2^q$ is a more important factor. We numerically verify that it suffices to choose $q = 12$ and $p = 0.75$, hence $m = 14$, to conduct a simulation comparable to classical solutions in Fig. 6(a), where the finite approximation error $N_s/2^q$ for phase estimation is smaller than the finite sampling threshold $1/\sqrt{N}$. We further compare how different numbers of iterations m impact the overall algorithmic error in Fig. 6(b). For a larger simulation, q can be determined as $q \geq \log_2(N_s\sqrt{N})$. One can readily generalize this algorithm to a DaQb scheme, where the size of the quantum circuit for the IPE algorithm increases to $b + 1$ qubits. This change increases circuit complexity, though not necessarily the number of bits.

Regarding complexity, having only one ancillary qubit comes with the cost of $O(2^{m-1})$ gates, where m is the number of IPE iterations related to the precision of the estimated phases. Given unbiased assumptions, the finite sampling error of the iterative phase estimation scales proportional to $1/\sqrt{N}$, where N is the shot number in quantum
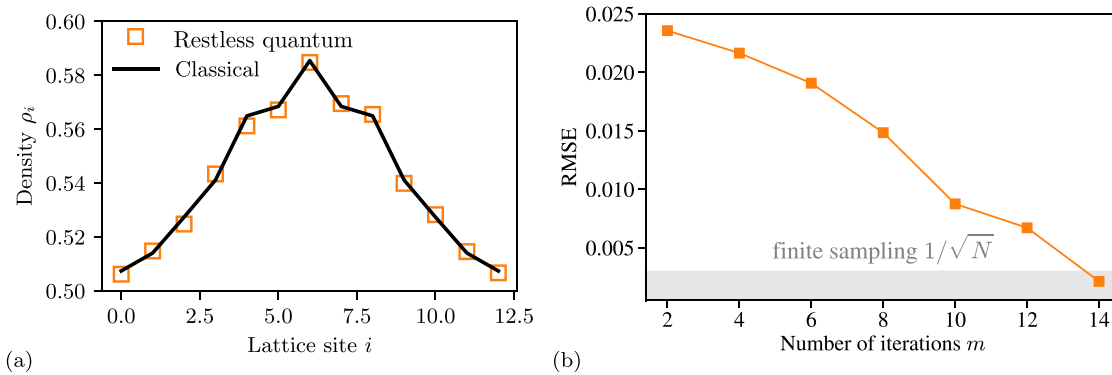

FIG. 6. (a) Restless QLBM (present work) solution to the diffusion equation using the IPE algorithm with $m = 14$ iterations, $N = 10^5$ shots, and its comparison to classical results for the same method and resolution. (b) Comparing different number of iterations m impact the overall algorithmic error.
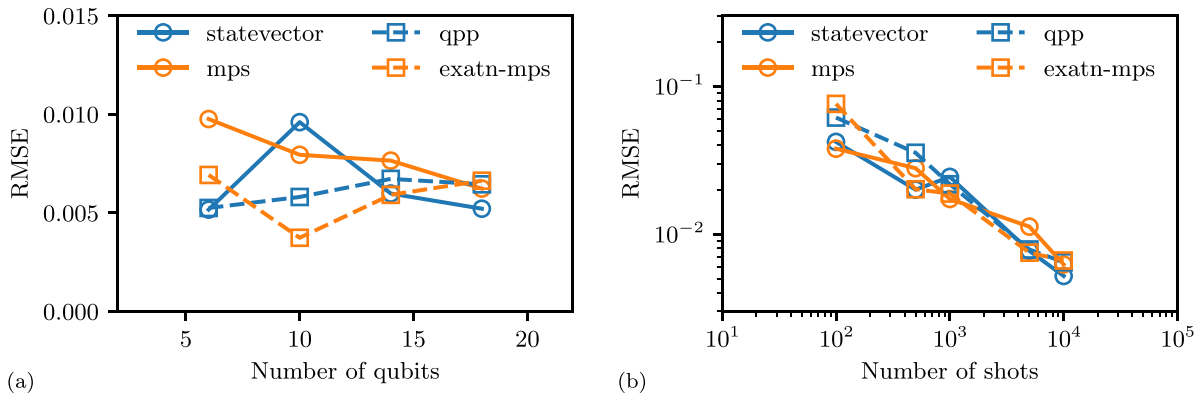
FIG. 7. RMSE of the quantum solution solving the diffusion equation compared to the classical one. (a) 10^4 shots and (b) 18 qubits. Qiskit (solid lines, ——) statevector and MPS simulators and XACC simulators (dashed lines, – – –) qpp and exatn-mps are shown. The colors correspond to (blue) statevector-like and (orange) matrix product state simulators. Notably, each lattice site contains two qubits for a D1Q2 scheme.

measurement. A qubit state is multiparametric, so subtracting the phase with a single estimation is more efficient.

V. RESULTS

We test different quantum simulator implementations of our QLBM algorithm and its circuit for verification. An in-depth discussion of the scaling of these simulators can be found in Appendix B. In this section, we take measurements from each simulator to show the general trends of RMSE with respect to the quantum hardware constraints.

We run the QLBM circuit of Fig. 3 on the four different simulators and compare their performance. Figure 7 shows and the root-mean-squared error (RMSE),

$$\text{RMSE} = \sqrt{\frac{\sum_{i=0}^{N_s-1} \left(\rho_i^{(\text{quant.})} - \rho_i^{(\text{classic.})} \right)^2}{N_s}}, \quad (13)$$

where N_s is the number of lattice sites of the quantum solution compared to the classical one as a function of the number of shots (samples from the underlying density functions) and qubits. The errors are reported from simulations as a function of the number of lattice sites in Fig. 7(a) and as a function of the number of shots in Fig. 7(b).

These figures demonstrate the scalability of this method. The RMSE for the algorithm follows the trend of decreasing quadratically with the number of shots, as shown in Fig. 7(b). An RMSE of 0.01 occurs at less than 10^4 shots. This accuracy level is proportional to $1/\sqrt{N}$ due to finite sampling of the quantum state, where N is the shot number. The number of qubits does not affect the RMSE greatly, as shown in Fig. 7(a), and the technique can scale well to more lattice sites with a linear increase in qubits.

The MPS simulator uses a local representation in a factorized form of tensor products.[40] MPS ensures that the overall structure remains small as long as the circuit has a low degree of entanglement, enabling more simulated qubits. To demonstrate this, we run the QLBM circuit of Fig. 3(a) on the Qiskit MPS simulator for a resolution of 51 lattice sites corresponding to 102 qubits using 10^6 shots. Results after ten time steps are shown in Fig. 8 and demonstrate good agreement with the classical solution.

VI. CONCLUSION AND DISCUSSION

In this work, we present a revised quantum lattice gas algorithm that can be used to simulate fluid flows. The algorithm of Yepez[41] was designed for type-II quantum computers that are no longer used. The presented algorithm uses a full quantum implementation of the streaming step to reduce classical communication between lattice sites. In addition, quantum streaming optimizes repetitive encoding operations by estimating the qubit relative phases and subtracting them at the end of each time step. State re-preparation and measurements can scale exponentially in modern computers, and our work on state preparation optimization is essential for efficiently adapting the QLGA to modern hardware. The algorithm is tested to solve canonical PDEs: the diffusion and Burgers' equations. We explored different quantum simulators available on Qiskit and XACC quantum computing frameworks and compared their performance and wall-clock time. Matrix product state simulators are most efficient for the presented quantum circuits, as they have low and local entanglement. These strategic algorithm improvements enable the execution of the quantum lattice gas algorithm on quantum hardware, a subject of future work.

An important limitation of the current algorithm is the size of the required quantum register, which scales linearly with the number of
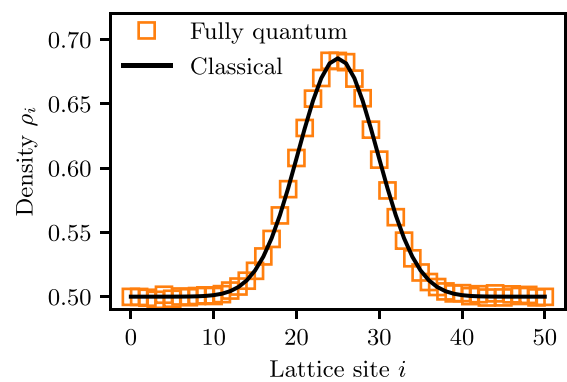

FIG. 8. QLBM solution using D1Q3 scheme to the diffusion equation using the Qiskit MPS simulator and its comparison to classical results using 102 qubits.

20 September 2024 15:49:28

lattice sites. For example, the number of required qubits in a two-dimensional $M \times M$ lattice is $5M^2$ using the D2Q5 scheme. This requirement constrains the ability to extend this algorithm to higher dimensions due to the limited number of qubits available on current quantum hardware. Thus, more appropriate encoding schemes, such as amplitude-based encoding, are needed to scale logarithmically with the number of lattice sites. For example, Budinski[22] implements the streaming step via a quantum random walk, similar to the collisionless Boltzmann equation solved on a quantum computer by Todorova and Steijl.[21] Indeed, in principle, one could design an algorithm that uses a quantum random walk for both the collision and streaming operations, which we will investigate in future work.

ACKNOWLEDGMENTS

AUTHOR DECLARATIONS
Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Sriharsha Kocherla: Conceptualization (equal); Data curation (equal); Methodology (equal); Software (equal); Validation (equal); Writing – original draft (equal). **Zhixin Song:** Conceptualization (equal); Data curation (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal); Validation (equal); Writing – original draft (equal). **Fatima Ezahra Chrit:** Conceptualization (equal); Formal analysis (equal); Investigation (equal); Methodology (equal); Software (equal). **Bryan Gard:** Formal analysis (equal); Investigation (equal); Methodology (equal); Resources (equal). **Eugene F. Dumitrescu:** Formal analysis (equal); Investigation (equal); Methodology (equal); Project administration (equal); Software (equal).

Alexander Alexeev: Conceptualization (equal); Funding acquisition (equal); Methodology (equal); Project administration (equal); Supervision (equal); Writing – original draft (equal). **Spencer H. Bryngelson:** Conceptualization (equal); Funding acquisition (equal); Investigation (equal); Methodology (equal); Supervision (equal); Validation (equal); Visualization (equal); Writing – original draft (equal).

DATA AVAILABILITY

The data that support the findings of this study are openly available in GitHub at https://github.com/comp-physics/fully-QLBM, Ref. 46.

APPENDIX A: D1Q3 IMPLEMENTATION

The implementations of the QLBM mentioned in Secs. II and III used the D1Q2 scheme, which uses two moving particles per lattice site. This scheme shows checkerboard pathologies when using a sharp function as the initial condition because it simulates two independent sublattices.[27] Yepez[37] presents a cure for this, allowing only one qubit to move while the other remains stationary. However, this requires two collision and streaming procedures per time step. Our results show that the quantum D1Q3 scheme, which adds a stationary qubit to each lattice site, solves this problem. A D1Q3 scheme has three particle distributions across three links: left, right, and stationary. So, some fictitious particles remain stationary in a time step, smoothing the curve. Figure 9 compares the quantum solution of the diffusion equation using the D1Q2 scheme with the D1Q3 scheme for a delta function as the initial condition. These simulations were performed using the Qiskit matrix product state simulator, 45 qubits for the D1Q3 scheme, and the sampling procedure used 10^6 shots. The spatial domain consists of $N_s = 15$ lattice sites, and results are shown after ten time steps. The result shows that the quantum D1Q3 solution agrees with the classical one and that the D1Q3 version remedies the checkerboard effect. We note that the D1Q3 scheme does not affect algorithmic complexity and can employ measurement avoidance analogous to the D1Q2 scheme presented in the main manuscript body.
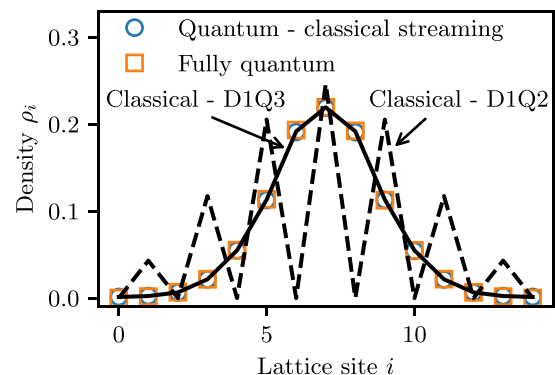

FIG. 9. QLBM solutions to the diffusion equation using the D1Q3 scheme. Quantum—classical streaming corresponds to the classical streaming algorithm discussed in previous work (Ref. 27), Fully quantum refers to the present algorithm, and Classical is the classical implementation of the lattice methods as labeled.
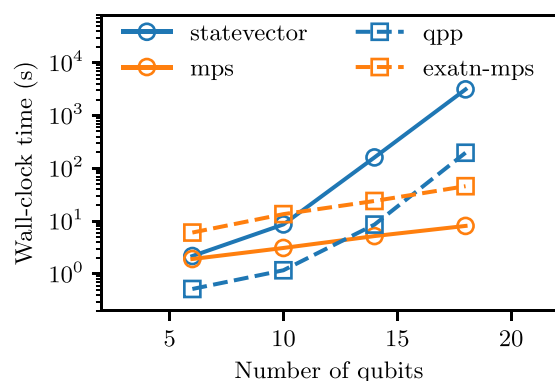
FIG. 10. Wall-clock time in seconds using 10^4 shots.

APPENDIX B: SIMULATOR RESULTS

In Fig. 10, we show the results for different quantum simulator implementations of our algorithm and its circuit. Simulators include statevector-based and tensor-network types that use the matrix product state (MPS) method.[42] Two different simulation frameworks are tested: Qiskit and XACC.[43] Statevector and MPS simulators are part of the Qiskit Aer backend. The XACC qpp (Quantum++) simulator is based on a C++ general-purpose quantum computing library.[44] The XACC exatn-mps simulator is a part of the TNQVM (tensor-network quantum virtual machine) simulation backend and uses a noiseless, matrix product state (MPS) wave function decomposition for the quantum circuit.[45] The wall-clock times of Fig. 7(c) were obtained via simulation on an AMD EPYC 7742 (64-core) processor.

We find that the results are shot-noise limited, and all quantum simulators obtain the same accuracy in terms of RMSE, which decreases from less than 10% for 10^2 shots to less than 1% for 10^4 shots. However, the Qiskit MPS simulator is at least one order of magnitude faster than the statevector ones. This behavior is expected. The Qiskit statevector and XACC (qpp) simulators generate the full state vector, which scales exponentially with the number of qubits.

REFERENCES

[1] S. B. Pope, *Turbulent Flows* (Cambridge University Press, 2000).
[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing* (ACM, 1996), pp. 212–219.
[3] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science* (IEEE, 1994), pp. 124–134.
[4] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre *et al.*, Chem. Rev. **119**, 10856 (2019).
[5] P. Wittek, *Quantum Machine Learning: What Quantum Computing Means to Data Mining* (Academic Press, 2014).
[6] M. Schuld, I. Sinayskiy, and F. Petruccione, Quantum Inf. Process. **13**, 2567 (2014).
[7] F. Marquardt, SciPost Phys. Lect. Notes **29**, 1–44 (2021).
[8] D. J. Egger, C. Gambella, J. Marecek, S. McFaddin, M. Mevissen, R. Raymond, A. Simonetto, S. Woerner, and E. Yndurain, IEEE Trans. Quantum Eng. **1**, 3101724 (2020).
[9] A. Bouland, W. van Dam, H. Joorati, I. Kerenidis, and A. Prakash, "Prospects and challenges of quantum finance," arXiv:2011.06492 (2020).
[10] A. W. Harrow, A. Hassidim, and S. Lloyd, Phys. Rev. Lett. **103**, 150502 (2009).
[11] A. M. Childs and J.-P. Liu, Commun. Math. Phys. **375**, 1427 (2020).
[12] A. M. Childs, J.-P. Liu, and A. Ostrander, "High-precision quantum algorithms for partial differential equations," arXiv:2002.07868 (2021).
[13] D. W. Berry, J. Phys. A **47**, 105301 (2014).
[14] F. Gaitan, npj Quantum Inf. **6**, 61 (2020).
[15] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, and P. J. Coles, "Variational quantum linear solver," arXiv:1909.05820 (2020).
[16] H.-L. Liu, Y.-S. Wu, L.-C. Wan, S.-J. Pan, S.-J. Qin, F. Gao, and Q.-Y. Wen, Phys. Rev. A **104**, 022418 (2021).
[17] J.-P. Liu, H. Ø. Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, and A. M. Childs, Proc. Natl. Acad. Sci. U. S. A. **118**, e2026805118 (2021).
[18] O. Kyriienko, A. E. Paine, and V. E. Elfving, Phys. Rev. A **103**, 052416 (2021).
[19] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, and D. Jaksch, Phys. Rev. A **101**, 010301 (2020).
[20] A. Mezzacapo, M. Sanz, L. Lamata, I. Egusquiza, S. Succi, and E. Solano, Sci. Rep. **5**, 13153 (2015).
[21] B. N. Todorova and R. Steijl, J. Comput. Phys. **409**, 109347 (2020).
[22] L. Budinski, Quantum Inf. Process. **20**, 57 (2021).
[23] L. Budinski, "Quantum algorithm for the Navier–Stokes equations," arXiv:2103.03804 (2021).
[24] W. Itani and S. Succi, Fluids **7**, 24 (2022).
[25] W. Itani, K. R. Sreenivasan, and S. Succi, "Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term," arXiv:2304.05915 (2023).
[26] M. A. Schalkers and M. Möller, Quantum Inf. Process. **23**, 20 (2024).
[27] J. Yepez, Phys. Rev. E **63**, 046702 (2001).
[28] J. Yepez, Int. J. Mod. Phys. C **12**, 1273 (2001).
[29] J. Yepez, J. Stat. Phys. **107**, 203 (2002).
[30] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information*, 10th ed. (Cambridge University Press, Cambridge, NY, 2010).
[31] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with Qiskit," arXiv:2405.08810 [quant-ph] (2024).
[32] U. Frisch, D. d'Humières, B. Hasslacher, P. Lallemand, Y. Pomeau, and J.-P. Rivet, "Lattice gas hydrodynamics in two and three dimensions," in *Lattice Gas Methods for Partial Differential Equations* (CRC Press, 2019), pp. 77–136.
[33] U. Frisch, B. Hasslacher, and Y. Pomeau, "Lattice-gas automata for the Navier–Stokes equation," in *Lattice Gas Methods for Partial Differential Equations* (CRC Press, 2019), pp. 11–18.
[34] G. R. McNamara and G. Zanetti, Phys. Rev. Lett. **61**, 2332 (1988).
[35] D. A. Wolf-Gladrow, *Lattice-Gas Cellular Automata and Lattice Boltzmann Models: An Introduction* (Springer, 2004).
[36] S. Chen and G. D. Doolen, Annu. Rev. Fluid Mech. **30**, 329 (1998).
[37] J. Yepez, Int. J. Mod. Phys. C **12**, 1285 (2001).
[38] Google Quantum AI, see https://quantumai.google/hardware/datasheet/weber.pdf for "Quantum computer datasheet" (2021).
[39] M. Dobšíček, G. Johansson, V. Shumeiko, and G. Wendin, Phys. Rev. A **76**, 030306 (2007).
[40] G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003).
[41] J. Yepez, "Quantum computation of fluid dynamics," Access No. ADA437298 (AFRL, 1998).
[42] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, "Matrix product state representations," arXiv:quant-ph/0608197 (2006).
[43] A. J. McCaskey, E. F. Dumitrescu, D. Liakh, M. Chen, W. Feng, and T. S. Humble, "Extreme-scale programming model for quantum acceleration within high performance computing," arXiv:1710.01794 (2017).
[44] V. Gheorghiu, PLoS One **13**, e0208073 (2018).
[45] A. McCaskey, E. Dumitrescu, M. Chen, D. Lyakh, and T. Humble, PLoS One **13**, e0206704 (2018).
[46] F. E. Chrit, S. Kocherla, B. Gard, E. F. Dumitrescu, A. Alexeev, and S. H. Bryngelson (2023). "Fully quantum algorithm for lattice Boltzmann methods with application to partial differential equations," GitHub. https://github.com/comp-physics/fully-QLBM