Journal Pre-proof

A multiple-circuit approach to quantum resource reduction with application to the quantum lattice Boltzmann method

Melody Lee, Zhixin Song, Sriharsha Kocherla, Austin Adams, Alexander Alexeev, Spencer H. Bryngelson

PII: S0167-739X(25)00270-5
DOI: https://doi.org/10.1016/j.future.2025.107975
Reference: FUTURE 107975

To appear in: *Future Generation Computer Systems*

Received date : 2 January 2025
Revised date : 19 April 2025
Accepted date : 11 June 2025

Please cite this article as: M. Lee, Z. Song, S. Kocherla et al., A multiple-circuit approach to quantum resource reduction with application to the quantum lattice Boltzmann method, *Future Generation Computer Systems* (2025), doi: https://doi.org/10.1016/j.future.2025.107975.

A multiple-circuit approach to quantum resource reduction with application to the quantum lattice Boltzmann method

Melody Lee[a], Zhixin Song[b], Sriharsha Kocherla[a], Austin Adams[c], Alexander Alexeev[d], Spencer H. Bryngelson[a,d,e]

[a]School of Computational Science & Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[b]School of Physics, Georgia Institute of Technology, Atlanta, GA 30332, USA
[c]School of Computer Science, Georgia Institute of Technology, Atlanta, GA 30332, USA
[d]George W. Woodruff School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA
[e]Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology, Atlanta, GA 30332, USA

Abstract

This work proposes a multi-circuit quantum lattice Boltzmann method (QLBM) algorithm that leverages parallel quantum computing to reduce quantum resource requirements. Computational fluid dynamics (CFD) simulations often entail a large computational burden on classical computers. At present, these simulations can require up to trillions of grid points and millions of time steps. To reduce costs, novel architectures like quantum computers may be intrinsically more efficient for these computations. Current quantum algorithms for solving CFD problems are based on a single quantum circuit and, in many cases, use lattice-based methods. Current quantum devices are adorned with sufficient noise and make large and deep circuits untenable. We introduce a multiple-circuit algorithm for a quantum lattice Boltzmann method (QLBM) solution of the incompressible Navier–Stokes equations. The method, called QLBM-frugal, aims to create more practical quantum circuits and strategies for differential equation-based problems. The presented method is validated and demonstrated for 2D lid-driven cavity flow. The two-circuit algorithm exhibits a notable reduction in CX gates, which account for the majority of the runtime on quantum devices. Compared to the baseline QLBM technique, a two-circuit strategy shows increasingly large improvements in gate counts as the qubit size, or problem size, increases. For 64 lattice sites, the CX count was reduced by 35%, and the gate depth decreased by 16%. This strategy also enables concurrent circuit execution, further halving the seen gate depth.

Keywords: Quantum algorithms; lattice Boltzmann method; parallel quantum computation; quantum resource reduction

1. Introduction

1.1. Motivation and application of quantum computation for scientific computing

Computationally intensive algorithms are obstacles to the scientific computing and engineering communities. Quantum computers can achieve complexity improvements by leveraging the properties of quantum mechanics, making them of interest. A prominent example of this so-called *quantum speedup* is demonstrated by Shor's algorithm [1]. The algorithm finds the prime factorization of integers in logarithmic time, an exponential speedup over the best-known classical algorithm. Other quantum algorithms include linear systems solvers [2, 3], Monte Carlo methods [4], and machine learning algorithms [5, 6]. With the recent progress in quantum hardware, these algorithms have been applied in industries such as optimization [7], quantum chemistry [8], and finance [9]. The potential for such large speedups makes partial differential equation (PDE) solvers attractive. However, classical problems in this realm tend to be nonlinear and non-unitary, presenting hurdles for quantum computing [10]. Attention has been turned toward addressing this challenge.

1.2. Quantum computation for computational fluid dynamics

Computational fluid dynamics (CFD) problems often require some of the world's largest classical supercomputers to solve. Focus has been directed toward alternative quantum algorithms for these problems [11–26]. The connection between quantum mechanics and fluid dynamics emerged early on through the Madelung equations, a reformulation of the Schrödinger equations [12, 27]. This relationship between quantum computing and CFD was further explored in 1993, when Succi and Benzi [28] used the lattice Boltzmann equation to describe non-relativistic quantum mechanics.

Over a decade later, the Harrow–Hassidim–Lloyd (HHL) algorithm, a sparse linear solver with exponential quantum speedup, was used to solve the incompressible Navier–Stokes equations [2]. Many quantum CFD solvers are also based on the quantum singular value transformation algorithm [29]. Hybrid quantum and classical algorithms have also been presented for CFD problems [30–32]. One reduces the CFD problem to a form that a quantum device can solve, and the latter solves the resulting linear system. However, the advantage posed by the HHL

Email address: shb@gatech.edu (Spencer H. Bryngelson)
Code available at: https://github.com/comp-physics/QLBM-frugal

algorithm does not account for the state preparation and readout requirements [33]. Implementing efficient input and output methods for quantum computers is in itself a challenge: loading an exponentially large state space representative of n qubits requires 2^n amplitudes to be encoded on a quantum computer. Various approaches have been taken, but present generalized encoding methods are upper-bounded by an exponential complexity [34–36], with slightly improved efficiency and depth on encoding sparse matrices [37].

1.3. Quantum computing for mesoscale CFD methods

This paper extends the body of work focused on mesoscale methods for fluid simulations. The dissipative particle dynamics and Boltzmann equation-based methods are commonly used in classical computation. Boltzmann-based equations characterize the statistical behavior of a system of particles, and their implementation in a quantum setting is the focus of this work. The Boltzmann methodology is primarily solved using lattice gas automata (LGA) [38, 39] and the lattice Boltzmann method (LBM) [40, 41]. LBM is the more advantageous of the two because it enables noise resilience and flexibility in complex multiphysics simulations. Prior studies have presented adaptations of both methods. For example, previous bodies of work proposed quantum algorithms for the lattice gas model [42–44]. More recently, a quantum lattice gas model was presented by Zamora, Budinski, Niemimäki, and Lahtinen [45] with improved efficiency. In 2020, Todorova and Steijl [46] described a quantum algorithm for the collisionless Boltzmann equation. This method accounts for the case of nearly free molecular flows while excluding complex non-unitary particle collisions.

Linearizing the problem is the first step in mapping the lattice Boltzmann method to a form suitable for quantum applications. This struggle with nonlinear terms is due to the linear framework on which quantum computing is defined. Budinski [47] introduced the first complete quantum lattice Boltzmann method (QLBM) for the advection–diffusion equation. The proposed method simulates particles across a grid of lattice sites with the truncated linear collision operator and extracts macroscopic quantities from mesoscopic simulations. Later, Budinski [48] extended the same approach to solve the 2D lid-driven cavity flow using the stream function–vorticity formulation of the Navier–Stokes equations. This adaptation removes the pressure term, reducing the problem to an advection–diffusion equation and a Poisson problem, solved on a single and dense quantum circuit. The work presented in this paper is a direct improvement on [48] by separating the computation of stream function and vorticity into two circuits and hence achieve circuit complexity improvements. Meanwhile, Wawrzyniak et al. [49] further generalized QLBM to 3D and arbitrary velocity set.

On the other side, the Carleman and Koopman–von Neumann embeddings have been used extensively to linearize equations for quantum computing [50, 51]. The Carleman linearization method embeds the variables of nonlinear equations directly into a quantum state up to certain truncation level. This approach is advantageous over the Koopman linearization, which instead embeds the variables through orthogonal polynomials. Itani and Succi [52] demonstrated a Carleman linearization for the collision term and later extended the work to demonstrate unitary evolution for both the collision and streaming operators [10]. Subsequently, Sanavio and Succi [53] introduced a QLBM algorithm that simulated Kolmogorov-like flows using the Carleman linearization method. They demonstrated that their implementation of the single-time-step operator would have a fixed circuit depth.

1.4. Limitations and recent improvements

Many of the proposed algorithms are presumed to operate on fault-tolerant quantum computers [17, 48, 53, 54]. Although the theory is promising, noise and decoherence in present quantum devices result in high error rates. Consequently, foundational algorithms, such as HHL, cannot be supported on modern quantum hardware [30]. The coherence time of a device is the range of time during which the quantum device reliably maintains its state such that the results obtained are within reason [55, 56]. Circuits whose runtime exceeds this coherence time output solutions that contain excessive noise, making them unusable. The sensitivity to the environment makes it challenging to scale quantum algorithms, as they require a large number of qubits. As a result, the aforementioned quantum CFD algorithms must be simulated.

The limitations in current quantum hardware are denoted as Noisy Intermediate-Scale Quantum (NISQ). These limitations have driven research towards hybrid variational quantum algorithms (VQAs) for both adiabatic and gate-based quantum computing [57]. The adiabatic configuration models the system as an energy minimization problem cast as an Ising Hamiltonian [58]. However, this approach is unable to adapt to allow control over precision and other parameters [59].

Gate-based VQAs are proposed as near-term replacements for HHL without the quantum advantage. Demirdjian et al. [60] solved the 1D advection–diffusion equation via Carleman linearization with reasonable accuracy. This approach uses a variational quantum linear solver [61] to compute the solution. Lubasch et al. [62] demonstrated that VQAs may account for nonlinear effects. However, beyond a certain circuit depth in noisy devices, VQAs are unlikely to outperform optimal classical counterparts [63]. Thus, methods to reduce the circuit depth and number of quantum gates applied are beneficial.

Several proposed modifications to the QLBM algorithm, as presented by Budinski [47, 48], aim to address the issue of requiring extensive quantum resources, including circuit depth and entangling two-qubit gate counts. For instance, the initial implementation employs a basic (canonical) shift algorithm to realize particle streaming in the model. Budinski et al. [64] reduced the two-qubit gate counts of the canonical shift algorithm by a constant factor using parallel state shift in both 1D left and right lattice directions. Meanwhile, new improvements in particle collision steps have been explored [65–68] through novel encoding schemes. Among them, Wang et al. [68] presents an interesting ensemble description that combines LBM and LGA. Xu et al. [67] also improves the collision term by introducing an

ancilla-free implementation with a small reduction of $O(N_{\text{qubit}})$ two-qubit gates. Last, Tiwari et al. [69] demonstrated the first 2D QLBM solve of the advection–diffusion equation on IonQ's quantum hardware. They evolve an initial Gaussian distribution using 10 time steps and 19-qubits with careful tensor network initialization, improved streaming step, and an error detection scheme.

1.5. Objectives

This manuscript presents a two-circuit 2D QLBM algorithm, referred to as QLBM-frugal here, to solve the incompressible Navier–Stokes equations. The major motivation of this work lies in the realization that the dense single-circuit scheme of Budinski [48] can be decomposed into two independent solves. These two solves can hence be optimized using classical parallelization on the same quantum processor without sophisticated quantum superposition, but come at the cost of tomographic techniques to read out solutions and pass them to each other. The method is validated against the classical LBM solution to the 2D lid-driven cavity flow problem. The quantum resources required for QLBM-frugal are compared to other approaches, and the parallelized implementation is discussed. Section 2 describes the classical LBM theory for simulating the advection–diffusion equation. Section 3 shows a single-circuit quantum implementation of the LBM for solving the advection–diffusion equation. In section 4, the two-circuit quantum implementation for solving the 2D Navier–Stokes equations is provided. Section 5 presents verification against the classically-solved LBM simulation and quantum resource estimations. Section 6 discusses the current limitations of the work. Section 7 summarizes the contributions of the presented method and its potential impact.

2. Mathematical formulation

2.1. Advection–diffusion equation

The advection–diffusion equation describes flow in a way that treats advection and diffusion as concurrent processes. The governing advection–diffusion PDE is

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = D\frac{\partial^2 \phi}{\partial x^2}, \tag{1}$$

where advection is defined by $c(\partial\phi/\partial x)$ and diffusion is defined by $D(\partial^2\phi/\partial x^2)$. Here, c and D are the advection and diffusion coefficients, and $\phi(t, x)$ is a scalar concentration field. We use the advection–diffusion algorithm devised by Budinski [47] as the foundation for the presented multi-circuit approach.

2.2. General lattice Boltzmann formulation

The LBM model for fluid flow simulates the evolution of particle flow over time, given by [70, 71] as

$$f_\alpha(\boldsymbol{r} + e_\alpha \Delta t, t + \Delta t) = (1 - \epsilon) f_\alpha(\boldsymbol{r}, t) + \epsilon f_\alpha^{(\text{eq})} + \Delta w_\alpha S, \tag{2}$$

where f_α indicates the particle distribution along each link α. e_α is the velocity of a particle in α, t is time, δt is the time step, $\boldsymbol{r}$ is the cell position, S is the source term, w_α is the proportion of particles streaming in link α, and $\epsilon = \Delta t/\tau$ where τ is the relaxation time. We more generally define $\boldsymbol{r}$ using Cartesian coordinates, whose notation depends on the dimension of the vector space. For example, in a one-dimensional space, we express $\boldsymbol{r} = (x)$, while in a two-dimensional space, we express $\boldsymbol{r} = (x, y)$. Figure 1 shows how the particle distribution may be visualized along a grid of cells. Each time step represents particle movement into neighboring cells, as determined by the above equation.

To solve (2) in the one- and two-dimensional planes, we investigate three configurations of the spatial lattice model. For a DnQm configuration, we consider an n-dimensional lattice grid with $m = |\alpha|$ links, where each link is a direction in which particles are propagated. This paper considers advection–diffusion results for the D1Q2, D1Q3, and D2Q5 configurations.

2.3. Formulation for one-dimensional scheme

Initially, we devise a one-dimensional model for the D1Q2 configuration. Two velocity vectors are considered such that

$$e_1 = -e_2 = \Delta x/\Delta t \tag{3}$$

for corresponding distribution functions f_1 and f_2. The symmetric boundary condition of the LBM imposes this equality. Our model resolves around the local equilibrium distribution function [72], formulated as

$$f_\alpha^{(\text{eq})}(\boldsymbol{r}, t) = w_\alpha \phi(\boldsymbol{r}, t)\left(1 + \frac{\boldsymbol{e_\alpha} \cdot \boldsymbol{c}}{c_s^2}\right), \tag{4}$$

where $\phi(\boldsymbol{r}, t)$ corresponds to the lattice site at lattice position $\boldsymbol{r}$ $= (x_i)$, $\boldsymbol{c}$ is the advection velocity vector, and c_s is the speed of sound. We set the weights $w_{1,2} = 0.5$ and $c_s = 1$, in accordance with the approach by Budinski [47]. Likewise, the D1Q3 configuration is considered with three velocity vectors, where $e_0 = 0$ and $e_1 = -e_2 = 1$. Let the speed of sound $c_s = 1/\sqrt{3}$ following Servan-Camas and Tsai [73].

2.4. Formulation for two-dimensional scheme

We now consider the two-dimensional D2Q5 scheme. Budinski [48] shows that these equations are reduced to resemble the advection–diffusion formulation, referencing his earlier work [47] in setting the parameters and governing formulas. We adapt this advection–diffusion formulation to a multiple-circuit method for advection and diffusion. This formulation is expressed as

$$\frac{\partial \phi}{\partial t} + c\frac{\partial \phi}{\partial x} = D\frac{\partial^2 \phi}{\partial x^2}, \tag{5}$$

where advection is defined by $c(\partial\phi/\partial x)$ and diffusion is defined by $D(\partial^2\phi/\partial x^2)$. Here, c and D are the advection and diffusion coefficients and $\phi(t, x)$ is a scalar concentration field.

The equilibrium distribution function is solved using (4), where $\boldsymbol{r} = (x_i, y_i)$. Let $c_s = 1/\sqrt{3}$ [73] and the weights be defined such that $w_0 = 2/6$ and $w_{2,3,4} = 1/6$ respective to the directions illustrated in fig. 1 [74].

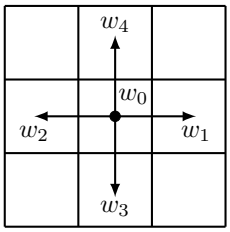


Figure 1: Illustration of a D2Q5 LBM lattice with a streaming particle and link weights w_α.

Following standard practice [40], the relaxation time τ relates to the diffusion constant D with

$$D = c_s^2\left(\tau - \frac{\Delta t}{2}\right). \tag{6}$$

Herein we use $D = 1/6$ and $\tau = \Delta t$. Recall we have set $c_s = 1/\sqrt{3}$. This gives $\varepsilon = \tau/\Delta t = 1$ and

$$f_\alpha(\boldsymbol{r} + \boldsymbol{e_\alpha}\Delta t, t + \Delta t) = f_\alpha^{(\mathrm{eq})} \tag{7}$$

simplifies as

$$f_\alpha(\boldsymbol{r} + \boldsymbol{e_\alpha}\Delta t, t + \Delta t) = w_\alpha \phi(\boldsymbol{r}, t)\left(1 + \frac{\boldsymbol{e_\alpha}\cdot\boldsymbol{c}}{c_s^2}\right), \tag{8}$$

which includes the collision (4) and the streaming steps (7). The concentration field is thus

$$\phi(\boldsymbol{r}, t) = \sum_{\alpha=0}^{N-1} f_\alpha(\boldsymbol{r}, t), \tag{9}$$

summing the particle distributions across all link directions α.

3. Quantum Lattice Boltzmann method

3.1. Quantum Lattice Boltzmann method for the advection–diffusion equation

We now discuss the premise of the body of work devised by Budinski [48]. Consider, first, the advection–diffusion equations. The quantum circuit performs the collision operator, followed by the streaming of particles and recalculation of macroscopic variables. Boundary conditions are applied at the end of each time step. No special care is needed when boundary conditions are periodic; the left and right shift gates L and R automatically propagate boundary conditions to the designated lattice site for each link α. Figure 2 illustrates how the quantum circuit's qubits are organized into 4 registers, or groupings of qubits, to solve the advection–diffusion equation using a D2Q5 scheme. Registers r_0 and r_1 contain $N_{\mathrm{qubit}} = \log_2(M)$ qubits, where M is the number of lattice sites in each dimension. Register d has $\lceil \log_2(m) \rceil$ qubits, where m is the number of LBM links α. Register a holds a single ancilla qubit necessary for applying the non-unitary collision operator using linear combination of unitaries (LCU).

3.1.1. Encoding input

For the current study, the amplitude encoding technique described in the work of Shende et al. [75] is employed. Amplitude encoding is part of the Qiskit toolkit [76], although it is also a generic encoding algorithm with an expensive gate count. We further elaborate on this in section 6.

At the start of each time step, we are given a distribution $\phi(\boldsymbol{r}, t)$ We define $\phi(\alpha, \boldsymbol{r})$ to be the value of ϕ at time t and position $\boldsymbol{r}$ at link direction α. Given a discretized concentration

$$\begin{aligned}\phi(\alpha, \boldsymbol{r}) = [&\phi(0,0), \phi(0,1), \phi(0,2), \ldots, \\ &\phi(4, M-2), \phi(4, M-1), \phi(4, M)],\end{aligned} \tag{10}$$

the initial statevector $|\psi_0\rangle$ is

$$|\psi_0\rangle = |0\rangle_a \otimes \frac{1}{\|\phi\|}\sum_{i=0}^{2M-1} \phi(\alpha, \boldsymbol{r})\,|i\rangle\,. \tag{11}$$

This formulation normalizes $\phi(\alpha, \boldsymbol{r})$, and the initial qubit states follow from amplitude encoding. In fig. 2, register r_0 stores the data at each link direction α. This data can be retrieved by specifying the link α in register d.

Applying the collision operator to the initial statevector $|\psi_0\rangle$ is equivalent to multiplying the $\phi(\alpha, \boldsymbol{r})$ with weight coefficients

$$w_\alpha\left(1 + \frac{\boldsymbol{e_\alpha}\cdot\boldsymbol{c}}{c_s^2}\right), \tag{12}$$

which follow from (4). This strategy is discussed further in the next subsection.

3.1.2. Collision operator

The collision step (fig. 2 (a)) computes the equilibrium distribution function $f_a^{(\mathrm{eq})}$, which requires computing the proportion of the distribution ϕ in each link α. The collision operator entails applying the coefficient matrix A to the current statevector $|\psi_0\rangle$.

The coefficient matrix A is not unitary, so it cannot be directly translated into a quantum gate. To mitigate this, an important strategy, devised by Childs and Wiebe [78], is employed by Budinski [48]. Suppose we split A into a linear combination of unitary matrices, C_1 and C_2, related to the original matrix as

$$C_{1,2} = A \pm i\sqrt{I - A^2}. \tag{13}$$

As $A = (C_1 + C_2)/2$, an operation with A is computed via block encoding [79, 80], where C_1 and C_2 are unitary, but A is, in general, not.

The circuit in fig. 3 evolves an input statevector $|\psi\rangle$ according to the large unitary

$$U = \frac{1}{2}\begin{bmatrix} V + W & V - W \\ V - W & V + W \end{bmatrix}, \tag{14}$$

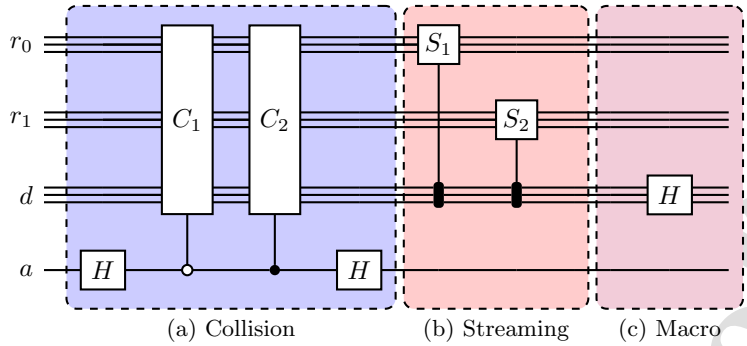


Figure 2: QLBM circuit for an advection–diffusion time step as devised by Budinski [47]. The collision operator $A = (C_1 + C_2)/2$, and shift operators $S_{1,2} \in \{L, R\}$ propagate particles in each link direction α. In panel (c), a Hadamard gate [77] is applied to the qubits in register d.

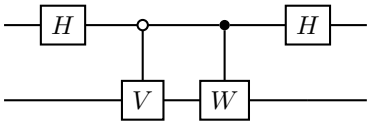


Figure 3: A block encoding of $B = (V + W)/2$. Here, V and W are unitary matrices.

where $B = (V + W)/2$. Thus, B is a subblock of the block matrix U.

Post-selection selects quantum states for specific measurement outcomes. Here, we use post-selection to measure the result after the collision operator for a 0 ancilla (after the block encoding).

The collision matrix is

$$A = \begin{bmatrix} k_1 I_m & 0 \\ 0 & k_2 I_m \end{bmatrix}, \quad \text{where} \quad k_\alpha = w_\alpha \left(1 + \frac{\boldsymbol{e}_\alpha \cdot \boldsymbol{c}}{c_s^2}\right) \tag{15}$$

are the link coefficients described by (12). Panel (a) of fig. 2 uses a linear combination of unitaries to apply matrix A to the input. The Hadamard gates are used in the block encoding process, along with C_1 and C_2 operations, which are derived from the unitary matrices in (13). The coefficients representing the proportion of particles in each link α are k_1 and k_2. The collision matrices are

$$C_{1,2} = \begin{bmatrix} \exp(\pm i \arccos(k_1)) I_m & 0 \\ 0 & \exp(\pm i \arccos(k_2)) I_m \end{bmatrix}. \tag{16}$$

The collision operator A transforms statevector $|\psi_0\rangle$ via a linear combination of C_1 and C_2, but requires an ancilla qubit a, which stores orthogonal data $(C_1 - C_2)/2$ when the ancilla is $|1\rangle$.

The orthogonal data was ignored through post-selection. The result of this linear combination is

$$|\psi_1\rangle = \frac{1}{||\phi||} \sum_i a_{i,i} \phi_{i,i} |i\rangle, \tag{17}$$

which encodes the post-collision values for each link direction α, for which the ancilla is $|0\rangle$.

The diagonal nature of the collision matrix A makes further optimizations of interest. The C_1 and C_2 collision operators derived from this A are diagonal and can be expressed in terms of pure rotation gates and CX (or two-qubit gate) operations [81]. There exist approaches to optimize the implementation of diagonal unitary gates [82, 83], although some rely on an efficient black-box oracle to compute mappings. The difficulty within QLBM comes from applying C_1 and C_2 as controlled unitary gates, which introduce more costly CX gates to the circuit decomposition after transpilation. Thus, work has been done to improve the general collision term. Xu et al. [67] design a collision operator that removes the role of the ancilla register using R_y and controlled-R_y gates, achieving a linear reduction on the magnitude of $O(\log_2(M))$ in CX gate count, given an $M \times M$ lattice. This improvement, among others, to the collision term could further reduce the resource usage and computational runtime of the proposed stream function and vorticity functions.

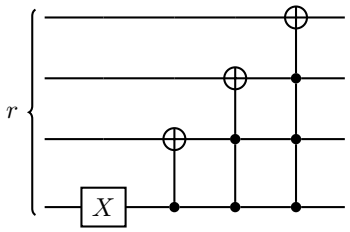
3.2. *Particle streaming*

The streaming step propagates particles in each link α to the neighboring site. Figure 5 shows the shift operators R and L, which are controlled on registers d with $\lceil \log_2(m) \rceil$ qubits and stream particles to neighboring lattice sites. The shift matrices are permutation matrices as

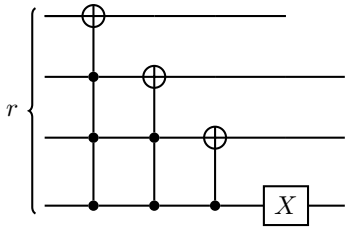
$$R = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \quad \text{and} \quad L = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}_{N_{\text{qubit}} \times N_{\text{qubit}}}, \tag{18}$$

and are both unitary matrices.

The resulting statevector $|\psi_2\rangle$ has a distribution shifted to a neighboring lattice site, determined by its link value α_i. Figure 4 shows how the right and left shift gates R and L are controlled on the link qubits d to act on the part of the statevector for distribution α.



(a) Left shift operator L.



(b) Right shift operator R.

Figure 4: Circuit decomposition for (a) left shift and (b) right shift operators.

Figure 5 shows the up and down operators used in a 2D lattice, which are implemented through the L and R operators applied via r_1. The shift algorithm presented here is a canonical version originating in quantum random walk [84], which is composed of cascading multi-controlled X gates. This quantum subroutine is used to propagate the weights only in the streaming function, as illustrated in figs. 5 and 6. Many recent studies on QLBM [49, 67] also utilize this canonical shift algorithm due to its simplicity and universality.

Proposals for more efficient shift algorithms aim to achieve a simultaneous shift towards multiple lattice directions. Compared to the canonical shift algorithm, a parallel shift algorithm introduced in [64] achieves state shift in both 1D left and right lattice directions by dividing the quantum state into even and odd basis components. Although both of them exhibit a linear scaling for the 2-qubit CX gate counts $O(N_{\text{qubit}})$, the parallel version largely improves the prefactor of this linear gate complexity by a factor of ~ 3. This strategy represents a meaningful, though constant, improvement, particularly for large problem sizes. When considering the entire circuit of streaming step, the shift algorithm is controlled by $\lceil \log_2(m) \rceil$ ancilla qubits and this eventually leads to $O(N_{\text{qubit}}^2)$ CX gates in total, where N_{qubit} is the total number of qubits in the circuit [67]. More recently, Tiwari et al. [69] proposed to use the one-hot encoding with $m-1$ ancilla qubits for the streaming step. By trading off between ancilla qubits and gate counts, this encoding would lead to a significant reduction of CX gates. These improvements are specific to the streaming step and can be additive to this work, but do not impact our conclusions regarding the separation of streaming and vorticity.

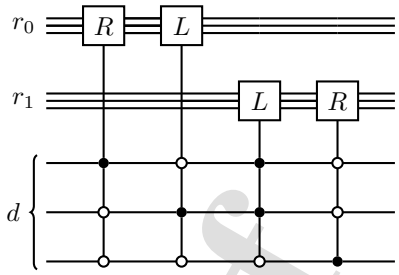


Figure 5: Streaming for a two-dimensional lattice grid. The streaming step uses right and left shift operators to shift distributions in their respective link directions α_1 and α_2 by controlling the gates on qubits in each link register d. These gates are applied to qubits in r_0 for left (right) shifts and qubits in r_1 for up(down) shifts.

3.3. Macroscopic variable retrieval

We retrieve the distribution $\phi(\boldsymbol{r}, t)$ by summing $f_\alpha^{(\text{eq})}$ over the link directions α. Figure 2 shows how this is accomplished by applying Hadamard gates to each of the qubits in link registers d, as shown in fig. 2 (c) [77]. A Hadamard gate H applied to a statevector $|\psi\rangle$, resulting in

$$H|\psi\rangle = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}\begin{bmatrix}\psi_a\\ \psi_b\end{bmatrix} = \frac{1}{\sqrt{2}}\begin{bmatrix}\psi_a+\psi_b\\ \psi_a-\psi_b\end{bmatrix} = \frac{\psi_a+\psi_b}{\sqrt{2}}|0\rangle + \frac{\psi_a-\psi_b}{\sqrt{2}}|1\rangle. \tag{19}$$

When the Hadamard gate is applied to a link qubit in d, it stores the sum of the two amplitudes as the $|0\rangle$ amplitude and the difference as the $|1\rangle$ amplitude. Thus, the sum can be post-selected by ignoring $|1\rangle$ measurements. With this post-selection, the Hadamard gates sum the distributions but introduce a factor of $1/\sqrt{2}$ per gate. This pre-factor is post-processed out of the computation by multiplication of a factor of $\sqrt{2}^{\log_2 m}$ where $m = |\alpha|$ is the number of link distributions. At each time step, we retrieve the circuit result via state tomography, which must be used to extract the first M lattice site elements.

4. Quantum Lattice Boltzmann method for the Navier–Stokes equations

4.1. Stream function–vorticity formulation

The incompressible 2D Navier–Stokes equations in Cartesian coordinates are

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{\text{Re}}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{20}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{\text{Re}}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{21}$$

where u and v are the velocity components in the x and y coordinate directions, p is the pressure, and Re is the Reynolds number, which is the ratio of inertial to viscous effects [85].

Taking the curl of the above Navier–Stokes equations recasts them in the so-called vorticity–stream function formulation, re-

moving the pressure term p and yielding

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = -\omega, \tag{22}$$

$$\frac{\partial\omega}{\partial t} + u\frac{\partial\omega}{\partial x} + v\frac{\partial\omega}{\partial y} = \frac{1}{\mathrm{Re}}\left(\frac{\partial^2\omega}{\partial x^2} + \frac{\partial^2\omega}{\partial y^2}\right). \tag{23}$$

In this formulation, (22) and (23) use vorticity ω and stream function ψ instead of directional speeds u and v. The velocity vector is thus $\boldsymbol{u} = \{u, v\}$. The stream function relates to the directional velocities as

$$\frac{\partial\psi}{\partial x} = u \quad \text{and} \quad \frac{\partial\psi}{\partial y} = -v. \tag{24}$$

Thus, (22) is a Poisson equation in stream function ψ and (23) is an advection–diffusion equation in vorticity ω.

4.1.1. Lattice-based representation

With the stream function–vorticity formulation, the collision, streaming, and macro lattice stages follow as

$$f_\alpha^{(\mathrm{eq})}(\boldsymbol{r}, t) = w_\alpha\omega(\boldsymbol{r}, t)\left(1 + \frac{\boldsymbol{e}_\alpha \cdot \boldsymbol{u}}{c_s^2}\right), \tag{25}$$

$$f_\alpha(\boldsymbol{r} + \boldsymbol{e}_\alpha\Delta t, t + \Delta t) = f_\alpha^{(\mathrm{eq})}, \tag{26}$$

$$\omega(\boldsymbol{r}, t) = \sum_\alpha f_\alpha(\boldsymbol{r}, t). \tag{27}$$

These stages, buttressed via the circuits of section 3.1, enable the vorticity ω computation.

The equilibrium distribution function for the Poisson equation ($\nabla^2\psi = -\omega$) is $g_\alpha^{(\mathrm{eq})}(\boldsymbol{r}, t) = w_\alpha\psi(\boldsymbol{r}, t)$. The streaming and macro steps match those of sections 3.2 and 3.3, but the source term $S = -\omega$ is added during the collision step (a). Thus, the relaxation operator is

$$g_\alpha(\boldsymbol{r} + \boldsymbol{e}_\alpha\Delta t, t + \Delta t) = g_\alpha^{(\mathrm{eq})} + \Delta w_\alpha S, \tag{28}$$

and macro retrieval equation

$$\psi(\boldsymbol{r}, t) = \sum_\alpha g_\alpha(\boldsymbol{r}, t). \tag{29}$$

4.1.2. Boundary conditions

The validation problem for the proposed method is a 2D lid-driven cavity flow. The spatial domain is $\Omega \in x, y$ with lengths L_x and L_y and boundary $\partial\Omega$. The stream function ψ is constant along the boundaries, with $\psi = 0$ used here. For the lattice Boltzmann method,

$$\psi = \sum_{\alpha=0}^{N_{\mathrm{links}}-1} g_\alpha^{(\mathrm{eq})} = g_0 + g_1 + g_2 + g_3 + g_4, \quad \text{so,} \quad g_{\partial\Omega} = -\sum_{\alpha|\alpha\neq\partial\Omega} g_\alpha. \tag{30}$$

Defining the vorticity expression (22) in terms of the stream function and expanding it in its Taylor series gives

$$\omega_{i,N_{\mathrm{qubit}}} = -2\left(\frac{\psi}{\Delta y^2} + \frac{U}{\Delta y}\right), \tag{31}$$

along the boundaries $\partial\Omega$, where U is wall-parallel velocity of the top wall. For a stationary wall, $U = 0$. The wall equilibrium distribution in the direction of the wall is

$$g(x, y)_{\partial\Omega} = -\sum_{\alpha|\alpha\neq\partial\Omega} g_\alpha - 2\left(\frac{-\psi}{\Delta y^2} + \frac{U}{\Delta y}\right). \tag{32}$$

To implement (32), the matrix

$$B = \begin{bmatrix} 0 & \cdots & 0 \\ \vdots & I_{N_{\mathrm{qubit}}-2} & \vdots \\ 0 & \cdots & 0 \end{bmatrix}, \tag{33}$$

is applied to the statevector $|\phi\rangle$, where B is of size $N_{\mathrm{qubit}} \times N_{\mathrm{qubit}}$, $N_{\mathrm{qubit}} = \log_2(M)$ in each dimension, and I_n denotes the size n identity matrix. Setting $g_{\partial\Omega} = 0$, while retaining other distribution values, enforces the boundary condition for the stream function circuit $|\psi_{\mathrm{b.c.}}\rangle = |0\rangle^{\otimes N_{\mathrm{qubit}}}$.

The vorticity circuit boundary conditions are applied via a linear combination of B to account for its nonlinear nature. This linear unitary combination is

$$D_{1,2} = B \pm i\sqrt{I_{N_{\mathrm{qubit}}} - B^2}, \tag{34}$$

following section 3.1.2.

4.2. Two-circuit model

The deviation from Budinski [48]'s original algorithm lies in the two-circuit approach, which defines distinct circuits for computing the stream function and vorticity. To devise this, we adopt the advection–diffusion circuits, as discussed in [47], and apply different bounds specific to the lid-driven cavity case. We discuss each circuit in more detail below.

4.2.1. Vorticity circuit

The vorticity circuit computes the vorticity ω for the current time step. The vorticity circuit in fig. 6 and advection–diffusion circuit in fig. 2 are nearly identical, save for the boundary conditions. This matching occurs because the vorticity equation in (23) follows the same form as the diffusion equation. The Navier–Stokes algorithm's vorticity circuit is the same as the advection–diffusion one if the boundary conditions are computed classically.

If boundary conditions are included, an extra qubit b stores them. Boundary conditions require additional computation, as enforcing such conditions is not a unitary operation. For this, the linear combination of unitaries is used [86], described further in section 3.1.2. The input to this circuit for the no-boundary version is the vorticity from the previous time step, ω_{t-1}. When using boundary conditions, the circuit input includes pre-computed

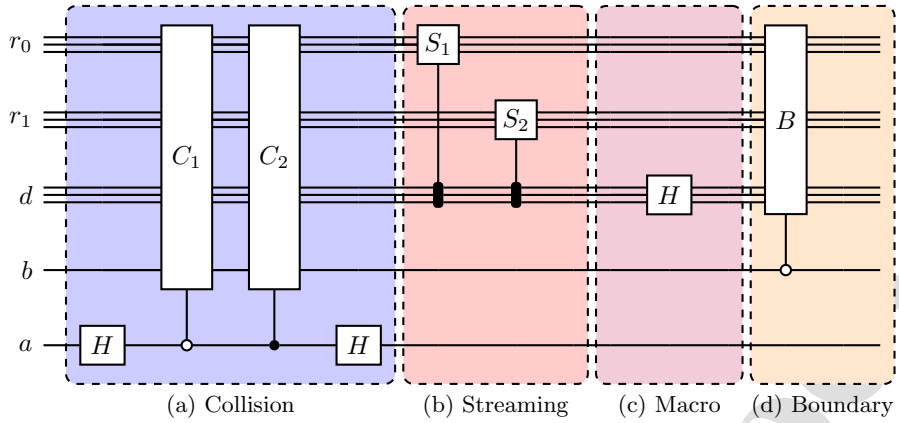


Figure 6: The original D2Q5 vorticity circuit proposed by Budinski [47], modified from the generalized circuit in fig. 2 for boundary conditions. The modified circuit includes an additional qubit, b, to store the boundary conditions and perform computations. We propose an algorithm that splits this circuit into two distinct parts.

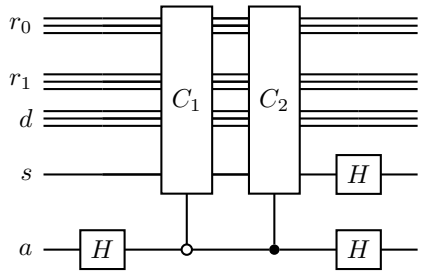


Figure 7: The collision portion of the stream function circuit. An additional qubit s stores the source term, and an additional Hadamard gate on qubit s after the block encoding adds the source term to the link distributions α_i before streaming.

boundary conditions in the boundary qubit b, computed following the descriptions in section 4.1.2.

4.2.2. *Stream function circuit*

The stream function circuit is the other quantum circuit used in this two-circuit model. The D2Q5 stream function circuit shown in fig. 7 is referred to for additional context. This circuit is similar to the advection–diffusion circuit with classical boundary conditions but includes an additional source term. Figure 7 shows the principle difference between these circuits; the additional qubit s stores the source term, $S = -\omega$.

The C_1 and C_2 gates also operate on the source term, and the Hadamard gate on qubit s adds this source to the stream function's equilibrium distribution function $g_\alpha^{(\mathrm{eq})}$. The same qubit-addition process from Section 4.2.1 applies. Boundary conditions require an additional b qubit store and a boundary gate B, which is not unitary and again is implemented via a linear combination of matrices following (14). The previous stream function, ψ_{t-1}, and source term, $S = -\omega_{t-1}$, serve as the input to the stream function circuit. The boundary conditions are computed according to section 4.1.2 for our quantum boundary condition variant.

5. Simulations and results

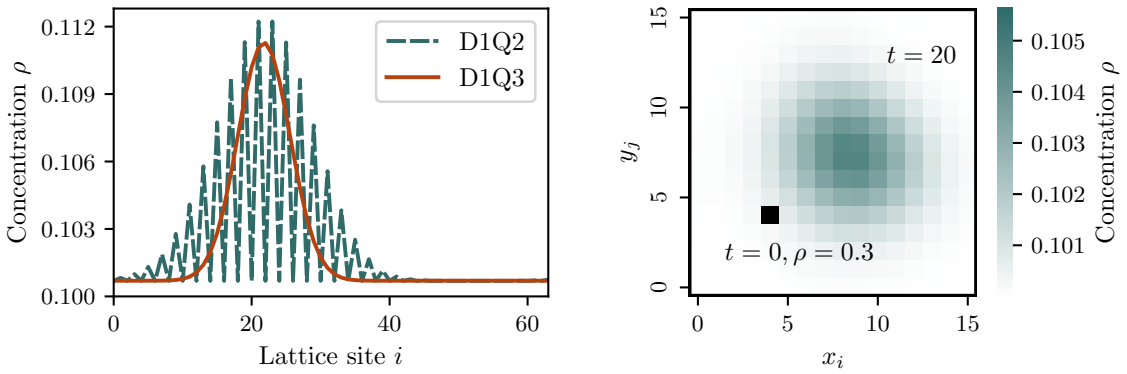
The QLBM algorithm of Budinski [48] uses a single circuit to solve the Navier–Stokes equations. This work builds upon corresponding research on advection–diffusion algorithms [47]. QLBM-frugal separates the computational process into distinct stream function and vorticity circuits. We verify the algorithm against that of Budinski [47]. We consider two cases for the advection–diffusion algorithm, the D1Q2 and D1Q3 lattice schemes, and validate the implementation accuracy before separating the stream function and vorticity circuits. We also consider a lid-driven cavity problem to verify the two-circuit approach. With this, we can compare how the resource use of the algorithm scales under increasing lattice site count against the work of Budinski [47].

5.1. *Advection–diffusion equation*

We apply the QLBM circuit to the advection–diffusion equation to obtain results for two example configurations. We verify these results against the expected outcomes returned by the classical LBM. In the 1D case, D1Q2 and D1Q3 lattice schemes simulate a dense concentration of $\rho = 0.2$ at source $x_i = 10$, undergoing advection and diffusion with uniform advection velocity of $c = 1/5$ and diffusion coefficient $D = 1/6$. The validation problem for the 2D case follows a diffusing concentration $\rho = 0.3$ at source $(x_i, y_j) = (4, 4)$ and 0.1 elsewhere, solved via a D2Q5 scheme.

Figure 12 shows the results of statevector simulations over 50 timesteps. A checkboard pattern arises in the D1Q2 case because the distribution moves wholly into neighboring areas, resulting in half of the lattice sites having zero particles. The algorithmic deficiency is remedied via the D1Q3 lattice scheme by setting the weight of vector w_0, specified in fig. 1, to 2/3. Figure 12 (a) shows that D1Q3 resolves the checkerboarding problem of D1Q2.

Figure 12 (b) shows the results of the 2D test problem solved with the D2Q5 scheme after 20 timesteps. There exists an initial



(a) D1Q2 and D1Q3

(b) D2Q5

Figure 8: Quantum LBM (a) D1Q2, D1Q3, and (b) D2Q5 results for the advection–diffusion equation. The initial condition in (a) is a point source of $\rho = 0.2$ at $x = 10$ and $\rho = 0.1$ otherwise. The initial condition in (b) is a point source of $\rho = 0.3$ at $(x, y) = (4, 4)$ and $\rho = 0.1$ elsewhere.

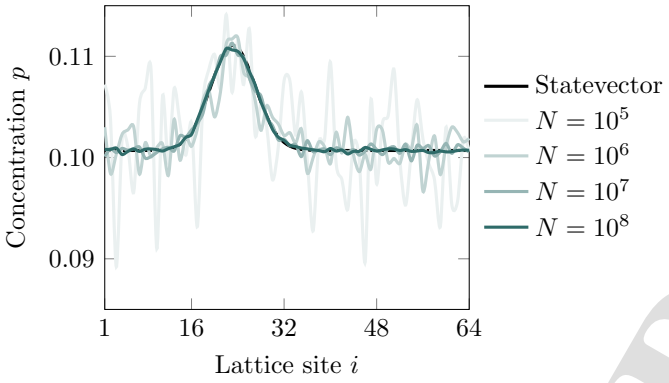


Figure 9: The results of the D1Q3 algorithm at iteration 50 are obtained via finite sampling of the quantum circuit at various sample sizes $N_{\text{samp.}}$.

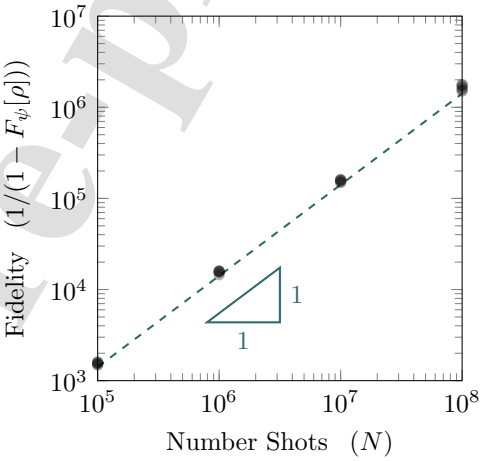


Figure 10: Verification via linear convergence with shot number for the advection–diffusion D1Q3 algorithm.

source at $(x_i = 4, y_j = 4)$, with an advection velocity in the positive x and y directions. The source advects in the direction of the velocity and diffuses outwards. The solution behaves as expected and agrees with classical LBM results.

The results are further validated via finite sampling. Figure 9 displays the results of finite sampling with various shot sample sizes $N_{\text{samp.}}$. We verify the behavior of the algorithm by computing the state fidelity of results obtained via finite sampling to the statevector of the ideal solution. The quantum state fidelity, $F_\psi(\rho)$, of a (probabilistic) mixed quantum state ρ with respect to a pure quantum state ψ is expressed as

$$F_\psi(\rho) = \langle\psi|\rho|\psi\rangle = \text{Tr}(p|\psi\rangle\langle\psi|). \tag{35}$$

Here, ψ is the expected solution obtained via the QLBM algorithm, and ρ is the probabilistic outcome obtained via finite sampling methods. From the equation defined in (35), we conclude that if ρ perfectly resembles the ideal solution, then we have $F_\psi(\rho) = 1$. The fidelity is expected to be expressed as a function of the number of shots $N_{\text{samp.}}$ used in the experiment, where in accordance to Yu et al. [87]

$$N_{\text{samp.}} \propto \frac{1}{1 - F_\psi(\rho)} \tag{36}$$

Figure 10 shows the results of this verification. The slope of the fit line in fig. 10 is 1.01, a 1% difference from the expected proportion.

5.2. *Navier–Stokes equations*

Figure 11 shows isocontours of the stream function for a lid-driven cavity problem. The problem serves to verify the two-circuit QLBM against a classical implementation of the lattice Boltzmann method.

To define relative errors between the quantum and classical lattice algorithms, we denote

$$\psi_{i,j} = \psi_{(x_i,y_j)} \quad \text{and} \quad \omega_{i,j} = \omega_{(x_i,y_j)}, \tag{37}$$

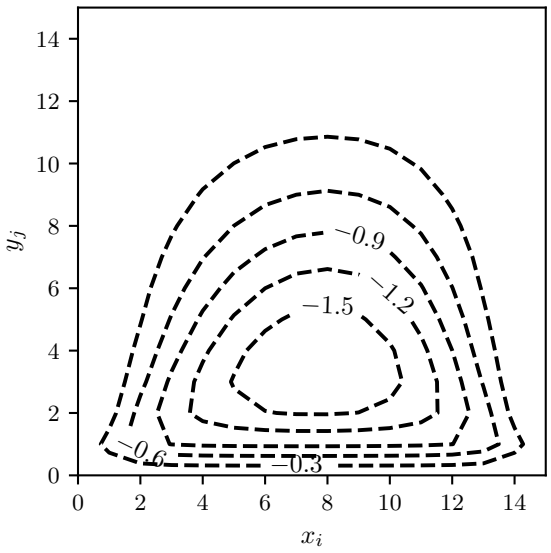


Figure 11: A 2D lid-driven cavity flow. Steady-state stream function isocontours are shown as labeled. The initial conditions are $\psi = 0$, $w = 0$, and $U = 1$, done over 80 timesteps.

	CX Gates	Circuit Depth
Single-circuit QLBM	25	58
Stream function	4.3	9.4
Vorticity	12	39
Stream function without boundaries	4.2	15

Table 1: Quantum resource estimation (all counts are in units of 10^4) for a D2Q5 algorithm with lattice size 64×64.

where $x_i = i\Delta x$ and $y_j = j\Delta y$. The local L_1 relative error between the classical and two-circuit QLBM Navier–Stokes solver is, thus,

$$\varepsilon_{\psi;i,j} = \frac{\psi_{i,j}^{\text{classic.}} - \psi_{i,j}^{\text{quant.}}}{\psi_{i,j}^{\text{classic.}}} \quad \text{and} \quad \varepsilon_{\omega;i,j} = \frac{\omega_{i,j}^{\text{classic.}} - \omega_{i,j}^{\text{quant.}}}{\omega_{i,j}^{\text{classic.}}}. \tag{38}$$

The relative errors are shown for the cavity problem in fig. 13. Figure 13 shows that the two-circuit QLBM agrees with the classical LBM when simulated with the statevector simulator in Qiskit's SDK [76]. The statevector simulator perfects exact tomography but is generally limited to small simulations due to exponentially increasing memory requirements with qubit numbers.

5.2.1. Quantum resource estimation and improvement

Implementing the two-circuit method for solving the Navier–Stokes equations using quantum lattice-based algorithms shows quantum resource advantages over the single-circuit method. The advantages are achieved in two areas: two-qubit gate count and runtime. Two-qubit gates like CX are slower and more error-prone than single-qubit gates, which has prompted bodies of work to reduce counts of these gates in quantum algorithms [88]. The projected runtime is calculated by summing the implementation time for each gate in the circuit. As such, the runtime is approximately proportional to the circuit depth.

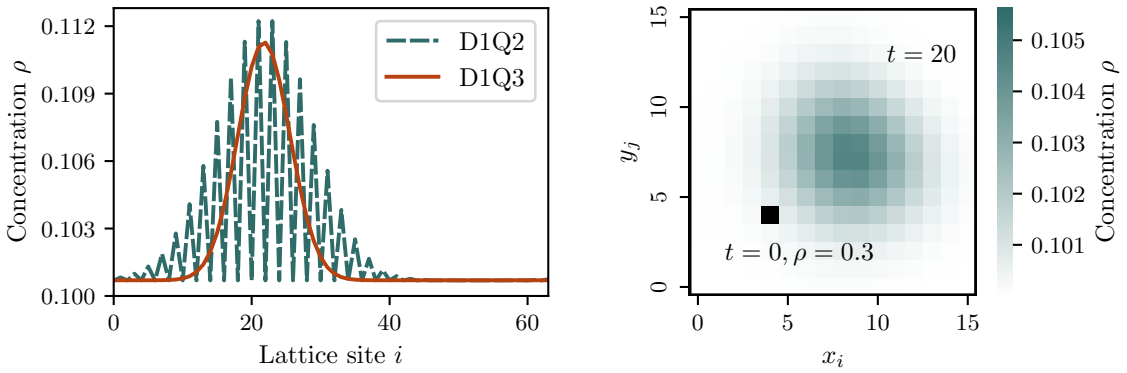
We show the reduction in gate counts by converting the circuit into a series of equivalent one- and two-qubit gates via the Qiskit transpiler. This procedure is conducted on a 64×64 lattice, with no prior optimizations or conversions to backend-specific gate sets. Table 1 shows that pre-computing the boundary conditions using classical methods reduces the gate count by at least 35%. By running the QLBM-frugal stream function and vorticity circuits concurrently, circuit depth is reduced to the maximum depth of its constituent circuits. Table 1 shows that a 33% reduction in circuit depth when compared against the single-circuit algorithm by Budinski [48].

We similarly transpile the results relative to a select backend. During compilation, we set the optimization level to be 3, as defined in Javadi-Abhari et al. [76] documentation. The optimization levels in Qiskit range from 0 to 3, corresponding to more aggressive optimization. The most aggressive optimizations include noise-adaptive qubit mapping and gate cancellation. This compilation reduces the circuit to one with fewer gates yet retains the same functionality. Still, such optimization only modestly reduces gate counts in the cases discussed herein.

The IBM Brisbane device is the backend of the transpilation process. Brisbane supports 127 qubits and is sufficient for resource estimation. The prior case of table 1 transpiled the circuit into a generalized set of gates. With the Brisbane backend, the transpiler identifies the device-specific gate set, which includes single-qubit X, RZ, and SX gates, as well as the two-qubit Echoed Cross-Resonance (ECR) gate. ECR gates and a series of single-qubit rotations are applied to implement a CX gate. Thus, reducing the ECR gate count is commensurate with a reduction in the CX count. The transpilation, optimization, and resource estimation processes are both performed on a local classical device before simulation.

Table 2 illustrates the resource estimation for a 16×16 lattice size case. All algorithms shown in the table, including the QLBM algorithm by Budinski [48], are optimized via the Qiskit transpiler before resource estimation. Table 2 shows that, after optimization, we see a 33% reduction in ECR gates. Of the total ECR gates applied, table 2 shows that the boundary conditions make up 44% of the ECR gate counts in the QLBM-frugal algorithm. It remains to be shown whether the reduction in two-qubit gates offsets the cost of computing these bounds on a classical device. Doing so involves accounting for the lattice size, hardware capabilities, and error rates of the gates. By computing the results of the stream function and vorticity circuits concurrently, the circuit depth is reduced by 41%.

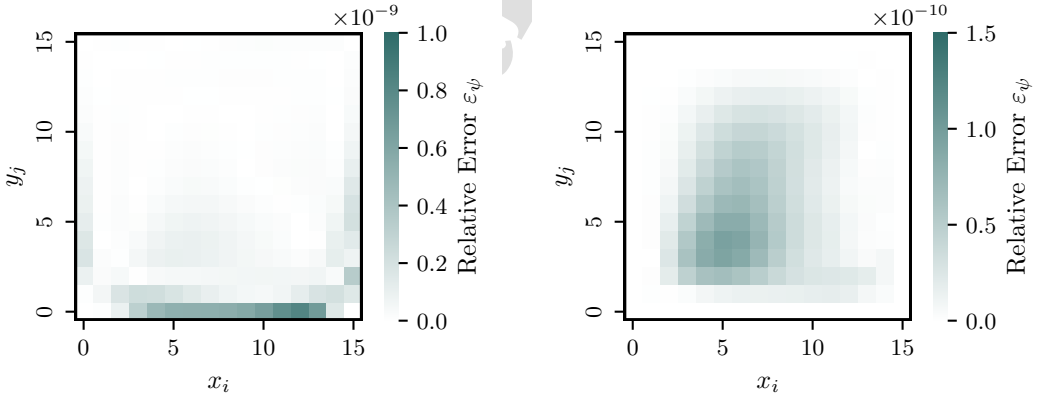
We now consider how the resource estimation changes with respect to the lattice size. First, the growth in ECR gate counts is observed in fig. 14 as the algorithm scales from a 2×2 to 32×32 lattice size. Figure 14 shows that the number of two-qubit gates required by Budinski [48]'s QLBM algorithm increases more



(a) D1Q2 and D1Q3

(b) D2Q5

Figure 12: Quantum LBM (a) D1Q2, D1Q3, and (b) D2Q5 results for the advection–diffusion equation. The initial condition in (a) is a point source of $\rho = 0.2$ at $x = 10$ and $\rho = 0.1$ otherwise. The initial condition in (b) is a point source of $\rho = 0.3$ at $(x, y) = (4, 4)$ and $\rho = 0.1$ elsewhere.



(a) Relative error ε_ω for vorticity circuit

(b) Relative error ε_ψ for stream function circuit

Figure 13: Relative error between the classical LBM and two-circuit quantum methods for (a) vorticity and (b) stream function.

	ECR Gates	Circuit Depth
Single-circuit QLBM	43	135
Stream function	7.6	29
Vorticity	21	80
Stream function without boundaries	7.5	28
Vorticity function without boundaries	5.0	17

Table 2: Quantum resource estimation, in units of 10^4, for a D2Q5 algorithm with lattice size 16×16 following transpiling with optimization level 3 on the IBM Brisbane backend.

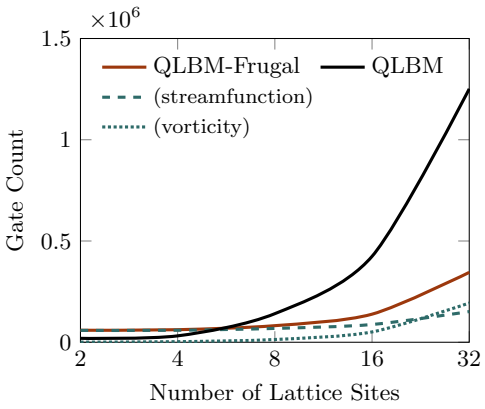


Figure 14: Comparison of costly two-qubit gate counts on the D2Q5 algorithm for the proposed two-circuit approach against Budinski [47]'s algorithm. Transpiled on the IBM Brisbane backend. The horizontal axis "Number of Lattice Sites" scales logarithmically.

rapidly than that of the proposed QLBM-frugal algorithm. The number of ECR gates is on the order of one million, indicating a notable difference in cumulative error between the initial and proposed algorithms. We obtained a runtime estimate for each transpiled circuit corresponding to these lattice sizes via the Qiskit scheduler. These are calculated based on the execution time of the gates on the specified backend. These estimates do not account for the encoding and readout costs, which are discussed in section 6.

Figure 15 shows the runtime of the respective algorithms. The stream function and vorticity circuits are presumed to run concurrently, indicating that the QLBM algorithm's runtime scales with the fastest-growing circuit. Note that the runtime of all of the circuits considered exceeds the coherence time of the IBM Brisbane hardware, as with most available quantum hardware at present. These coherence times are on the order of several hundred microseconds. Thus, results can not be reliably computed on a real device for even the smallest number of lattice sites. Furthermore, any classical simulations that attempt to replicate the error rates of available hardware will yield only noise. Nonetheless, fig. 15 demonstrates that running QLBM-frugal in parallel achieves a reduction in runtime when compared to the work by Budinski [47].

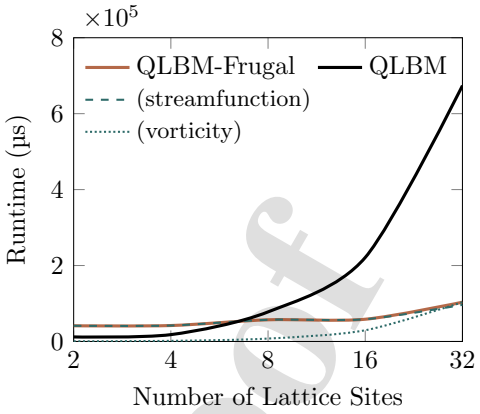


Figure 15: Runtime estimation for the proposed D2Q5 QLBM-frugal algorithm without quantum bounds, compared against the QLBM algorithm proposed by Budinski [47]. The circuit is transpiled with optimization level 3 on the IBM Brisbane backend.

6. Limitations of current work

6.1. Hardware limitations

Current quantum devices have high noise floors and low qubit counts. LBM problems are often large, requiring many qubits, and so are beyond the capabilities of current quantum devices. The results presented here are derived from quantum simulation. Concurrently executing the stream function and vorticity circuits doubles the number of required qubits if they are "parallelized" on the same quantum device. Instead, the quantum work can be distributed across multiple quantum processors; however, we limit the present work to a single-device analysis.

While the qubit count is doubled, the depth of each circuit is reduced from the initial algorithm proposed by Budinski [47]. This strategy reduces the overall runtime and two-qubit gate count. However, the estimated runtime exceeds the coherence time of available hardware by several orders of magnitude. The presented method still depends on quantum simulators, even for small problems.

6.2. Encoding and readout costs

The algorithm of Budinski [47] and the presented two-circuit approach begin with an arbitrary state. This approach assumes that the state has been encoded in the quantum RAM or an equivalent set of qubits. Unless the amplitudes are roughly uniform, the encoding process is costly. For some quantum algorithms, this could reduce the expected quantum advantage over their classical counterparts to a polynomial one [33]. In the Qiskit implementation of state preparation, the resource cost scales exponentially with respect to the number of qubits [89]. Thus, appreciating the impacts of encoding can inform the application of QLBM algorithms.

The final readout process depends on the final state of the solution. Quantum state tomography measures (or reads out) the state of the quantum device. One requires $O(3^{N_{\text{qubit}}})$ measurements to determine the quantum state of a $2^{N_{\text{qubit}}}$-dimensional

Hilbert space [90]. This exponential readout cost can meaningfully impact the runtime and required resources. This cost motivates an implementation that bypasses exponential readout costs or admits a heuristic-equipped final state for tomography. More efficient quantum state tomography methods exist in special cases, and the topic remains an active research problem [90–94].

7. Conclusion

This work presents improvements on the quantum lattice Boltzmann method for solving the two-dimensional Navier–Stokes equations. We present a QLBM algorithm that utilizes concurrent-circuit computation of the stream function and vorticity. These modifications introduce the possibility of solving the Navier–Stokes equations via parallelization or distributed computing. We first verified the solution returned by the proposed QLBM-frugal algorithm against classical LBM methods. We selected the lid-driven cavity problem for algorithm verification, implementing both QLBM and QLBM-frugal using Qiskit. The error between the results is negligible on ideal quantum devices, indicating the algorithm is practical on fault-tolerant devices.

Moreover, we demonstrate that QLBM-frugal achieves a reduction in the two-qubit gate count compared to the previous single-circuit implementation of an otherwise similar algorithm [48]. Quantum devices are bottlenecked, in part, by limitations due to errors and environmental interference. Reducing the number of two-qubit gates will improve the algorithm's accuracy. We demonstrate a 33% reduction in the two-qubit ECR gate count following optimization and transpilation on the IBM Brisbane device for the 16×16 lattice and a 35% reduction in CX gate count when the algorithm is transpiled relative to a general gate set on the larger 64×64 lattice, corresponding to a reduction in the total circuit depth. Concurrent computation of the stream function and vorticity circuits reduces the depth on the 16×16 lattice size case by 41%. This reduction is significant: even for small problems, nearly one million individual gates are required to implement the circuit on quantum hardware. This reduction eliminates approximately $O(10^5)$ qubit rotations and gates from the computation for each iteration, thereby reducing the accumulated error associated with continued gate application. Future works should keep exploring more efficient parallel shift algorithms [64] to higher dimensions and further reduce the gate counts.

The reduction in ECR gates and depth extends to cases beyond the 16×16 lattice size. We show that the QLBM-frugal resource requirements for lattice sizes ranging from 2×2 to 32×32 grow at a slower rate than that of the QLBM algorithm. The expected number of qubits scales logarithmically with the lattice size, $O(\log M)$, slower than the linear growth demonstrated in Yepez [44]. The runtime of the present work, QLBM-frugal, scales more slowly than the traditional QLBM case. While we are far from being capable of implementing such a large circuit on current NISQ hardware, the changes make the circuit more feasible and less prone to errors on an ideal device relative to its predecessors.

The encoding and readout costs remain a bottleneck for the algorithm. The processes must be performed at each iteration to input the prior state of the system and extract the information following each computation. This process is costly, and its complexity scales exponentially with respect to the number of qubits. The current work does not address read-in or out costs, although a comprehensive QLBM strategy for NISQ devices requires attention to this aspect.

Acknowledgments

We thank Professors J. Yepez and L. Budinski for fruitful discussions of this work. This work was supported in part by DARPA grant number HR0011-23-3-0006, the Georgia Tech Quantum Alliance, and a Georgia Tech Seed Grant. This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.

References

[1] P. W. Shor, Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer, SIAM Journal on Computing **26** (1997) 1484–1509.
[2] A. W. Harrow, A. Hassidim, S. Lloyd, Quantum algorithm for solving linear systems of equations, Physical Review Letters **103** (2009) 150502.
[3] B. D. Clader, B. C. Jacobs, C. R. Sprouse, Preconditioned quantum linear system algorithm, Physical Review Letters **110** (2013) 250504.
[4] A. Montanaro, Quantum speedup of Monte Carlo methods, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **471** (2015) 20150301.
[5] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, J. M. Gambetta, Supervised learning with quantum-enhanced feature spaces, Nature **567** (2019) 209–212.
[6] Y. Liu, S. Arunachalam, K. Temme, A rigorous and robust quantum speed-up in supervised machine learning, Nature Physics **17** (2021) 1013–1017.
[7] E. Farhi, J. Goldstone, S. Gutmann, A quantum approximate optimization algorithm, arXiv preprint arXiv:1411.4028 (2014).
[8] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. D. Sawaya, S. Sim, L. Veis, A. Aspuru-Guzik, Quantum chemistry in the age of quantum computing, Chemical Reviews **119** (2019) 10856–10915.
[9] D. Herman, C. Googin, X. Liu, Y. Sun, A. Galda, I. Safro, M. Pistoia, Y. Alexeev, Quantum computing for finance, Nature Reviews Physics **5** (2023) 450–465.
[10] W. Itani, K. R. Sreenivasan, S. Succi, Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term, Physics of Fluids **36** (2024).
[11] N. Gourianov, M. Lubasch, S. Dolgov, Q. Y. van den Berg, H. Babaee, P. Givi, M. Kiffner, D. Jaksch, A quantum-inspired approach to exploit turbulence structures, Nat. Comput. Sci. **2** (2022) 30–37.
[12] S. Succi, W. Itani, K. Sreenivasan, R. Steijl, Quantum computing for fluids: Where do we stand?, Europhysics Letters **144** (2023) 10001.
[13] S. Succi, W. Itani, C. Sanavio, K. R. Sreenivasan, R. Steijl, Ensemble fluid simulations on quantum computers, Computers & Fluids **270** (2024) 106148.
[14] R. Steijl, Quantum algorithms for fluid simulations, Advances in Quantum Communication and Information (2019) 31.
[15] Y. Moawad, W. Vanderbauwhede, R. Steijl, Investigating hardware acceleration for simulation of CFD quantum circuits, Frontiers in Mechanical Engineering **8** (2022) 925637.
[16] R. Steijl, G. N. Barakos, Parallel evaluation of quantum algorithms for computational fluid dynamics, Computers & Fluids **173** (2018) 22–28.
[17] F. Gaitan, Finding flows of a Navier–Stokes fluid through quantum computing, npj Quantum Information **6** (2020) 61.

[18] S. S. Bharadwaj, K. R. Sreenivasan, Quantum computation of fluid dynamics, arXiv preprint arXiv:2007.09147 **15** (2020) 1–20.

[19] S. S. Bharadwaj, K. R. Sreenivasan, Hybrid quantum algorithms for flow problems, Proceedings of the National Academy of Sciences **120** (2023) e2311014120.

[20] X. Li, X. Yin, N. Wiebe, J. Chun, G. K. Schenter, M. S. Cheung, J. Mülmenstädt, Potential quantum advantage for simulation of fluid dynamics, Physical Review Research **7** (2025) 013036.

[21] F. Oz, R. K. Vuppala, K. Kara, F. Gaitan, Solving Burgers' equation with quantum computing, Quantum Information Processing **21** (2022) 1–13.

[22] D. Jaksch, P. Givi, A. J. Daley, T. Rung, Variational quantum algorithms for computational fluid dynamics, AIAA Journal **61** (2023) 1885–1894.

[23] P. Givi, A. J. Daley, D. Mavriplis, M. Malik, Quantum speedup for aeroscience and engineering, AIAA Journal **58** (2020) 3715–3727.

[24] G. Xu, A. J. Daley, P. Givi, R. D. Somma, Turbulent mixing simulation via a quantum algorithm, AIAA Journal **56** (2018) 687–699.

[25] G. Xu, A. J. Daley, P. Givi, R. D. Somma, Quantum algorithm for the computation of the reactant conversion rate in homogeneous turbulence, Combustion Theory and Modelling **23** (2019) 1090–1104.

[26] S. Sammak, A. Nouri, N. Ansari, P. Givi, Quantum computing and its potential for turbulence simulations, in: Mathematical Modeling of Technological Processes: 8th International Conference, CITech 2015, Almaty, Kazakhstan, September 24-27, 2015, Proceedings 8, Springer, 2015, pp. 124–132.

[27] E. Madelung, Eine anschauliche deutung der Gleichung von Schrodinger, Die Naturwissenschaften **14** (1926) 1004–1004.

[28] S. Succi, R. Benzi, Lattice Boltzmann equation for quantum mechanics, Physica D: Nonlinear Phenomena **69** (1993) 327–332.

[29] A. Gilyén, Y. Su, G. H. Low, N. Wiebe, Quantum singular value transformation and beyond: Exponential improvements for quantum matrix arithmetics, Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing (2019).

[30] L. Lapworth, A hybrid quantum-classical CFD methodology with benchmark HHL solutions, arXiv preprint arXiv:2206.00419 (2022).

[31] S. S. Bharadwaj, K. R. Sreenivasan, Hybrid quantum algorithms for flow problems, Proceedings of the National Academy of Sciences **120** (2023).

[32] Z. Song, R. Deaton, B. Gard, S. H. Bryngelson, Incompressible Navier–Stokes solve on noisy quantum hardware via a hybrid quantum–classical scheme, Computers & Fluids **288** (2025) 106507.

[33] S. Aaronson, Read the fine print, Nature **11** (2015) 291–293.

[34] P. Niemann, R. Datta, R. Wille, Logic synthesis for quantum state generation, 2016 IEEE 46th International Symposium on Multiple-Valued Logic (ISMVL) (2016).

[35] F. Mozafari, M. Soeken, H. Riener, G. De Micheli, Automatic uniform quantum state preparation using decision diagrams, 2020 IEEE 50th International Symposium on Multiple-Valued Logic (ISMVL) (2020) 170–175.

[36] I. F. Araujo, D. K. Park, F. Petruccione, A. J. da Silva, A divide-and-conquer algorithm for quantum state preparation, Scientific Reports **11** (2021).

[37] N. Gleinig, T. Hoefler, An efficient algorithm for sparse quantum state preparation, 2021 58th ACM/IEEE Design Automation Conference (DAC) (2021).

[38] U. Frisch, D. d'Humieres, B. Hasslacher, P. Lallemand, Y. Pomeau, J. P. Rivet, Lattice gas hydrodynamics in two and three dimensions, in: Modern Approaches to Large Nonlinear Physical Systems Workshop, 1986.

[39] U. Frisch, B. Hasslacher, Y. Pomeau, Lattice-gas automata for the Navier–Stokes equation, Physical Review Letters **56** (1986) 1505–1508.

[40] T. Krüger, H. Kusumaatmaja, A. Kuzmin, O. Shardt, G. Silva, E. Viggen, The Lattice Boltzmann Method: Principles and Practice, Graduate Texts in Physics, Springer International Publishing, 2016.

[41] S. Chen, G. D. Doolen, Lattice Boltzmann method for fluid flows, Annual Review of Fluid Mechanics **30** (1998) 329–364.

[42] J. Yepez, Lattice-gas quantum computation, International Journal of Modern Physics C **09** (1998) 1587–1596.

[43] J. Yepez, Quantum lattice-gas model for computational fluid dynamics, Physical Review E **63** (2001) 046702.

[44] J. Yepez, Quantum lattice-gas model for the Burgers' equation, Journal of Statistical Physics **107** (2002) 203–224.

[45] A. D. B. Zamora, L. Budinski, O. Niemimäki, V. Lahtinen, Efficient quantum lattice gas automata, Computers & Fluids **286** (2025) 106476.

[46] B. N. Todorova, R. Steijl, Quantum algorithm for the collisionless Boltzmann equation, Journal of Computational Physics **409** (2020) 109347.

[47] L. Budinski, Quantum algorithm for the advection–diffusion equation simulated with the lattice Boltzmann method, Quantum Information Processing **20** (2021) 57.

[48] L. Budinski, Quantum algorithm for the Navier–Stokes equations by using the streamfunction-vorticity formulation and the lattice Boltzmann method, International Journal of Quantum Information **20** (2022) 2150039.

[49] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indinger, C. F. Janßen, U. Schramm, N. A. Adams, A quantum algorithm for the lattice-Boltzmann method advection-diffusion equation, Computer Physics Communications **306** (2025) 109373.

[50] J.-P. Liu, H. Ø. Kolden, H. K. Krovi, N. F. Loureiro, K. Trivisa, A. M. Childs, Efficient quantum algorithm for dissipative nonlinear differential equations, Proceedings of the National Academy of Sciences **118** (2021) e2026805118.

[51] Y. Ito, Y. Tanaka, K. Fujii, How to map linear differential equations to schrödinger equations via Carleman and Koopman-von Neumann embeddings for quantum algorithms, arXiv preprint arXiv:2311.15628 (2023).

[52] W. Itani, S. Succi, Analysis of Carleman linearization of lattice Boltzmann, Fluids **7** (2022).

[53] C. Sanavio, S. Succi, Lattice Boltzmann–Carleman quantum algorithm and circuit for fluid flows at moderate Reynolds number, AVS Quantum Science **6** (2024).

[54] P. C. Costa, S. Jordan, A. Ostrander, Quantum algorithm for simulating the wave equation, Physical Review A **99** (2019).

[55] D. A. Lidar, I. L. Chuang, K. B. Whaley, Decoherence-free subspaces for quantum computation, Physical Review Letters **81** (1998) 2594–2597.

[56] T. D. Ladd, F. Jelezko, R. Laflamme, Y. Nakamura, C. Monroe, J. L. O'Brien, Quantum computers, Nature **464** (2010) 45–53.

[57] D. Jaksch, P. Givi, A. J. Daley, T. Rung, Variational quantum algorithms for computational fluid dynamics, AIAA Journal **61** (2023) 1885–1894.

[58] S. Srivastava, V. Sundararaghavan, Box algorithm for the solution of differential equations on a quantum annealer, Physical Review A **99** (2019).

[59] N. Ray, T. Banerjee, B. Nadiga, S. Karra, Towards solving the Navier–Stokes equation on quantum computers, arXiv preprint arXiv:1904.09033 (2019).

[60] R. Demirdjian, D. Gunlycke, C. A. Reynolds, J. D. Doyle, S. Tafur, Variational quantum solutions to the advection–diffusion equation for applications in fluid dynamics, Quantum Information Processing **21** (2022) 322.

[61] C. Bravo-Prieto, R. LaRose, M. Cerezo, Y. Subasi, L. Cincio, P. J. Coles, Variational quantum linear solver, Quantum **7** (2023) 1188.

[62] M. Lubasch, J. Joo, P. Moinier, M. Kiffner, D. Jaksch, Variational quantum algorithms for nonlinear problems, Physical Review A **101** (2020) 010301.

[63] G. De Palma, M. Marvian, C. Rouzé, D. S. França, Limitations of variational quantum algorithms: A quantum optimal transport approach, PRX Quantum **4** (2023).

[64] L. Budinski, O. Niemimäki, R. Zamora-Zamora, V. Lahtinen, Efficient parallelization of quantum basis state shift, Quantum Science and Technology **8** (2023) 045031.

[65] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indiniger, C. F. Janßen, U. Schramm, N. A. Adams, Dynamic circuits for the quantum lattice-Boltzmann method, arXiv preprint arXiv:2502.02131 (2025).

[66] D. Wawrzyniak, J. Winter, S. Schmidt, T. Indinger, U. Schramm, C. Janßen, N. A. Adams, Unitary quantum algorithm for the lattice-boltzmann method, arXiv preprint arXiv:2405.13391 (2024).

[67] L. Xu, M. Li, L. Zhang, H. Sun, J. Yao, Improved quantum lattice Boltzmann method for advection-diffusion equations with a linear collision model, Physical Review E **111** (2025) 045305.

[68] B. Wang, Z. Meng, Y. Zhao, Y. Yang, Quantum lattice boltzmann method for simulating nonlinear fluid dynamics, arXiv preprint arXiv:2502.16568 (2025).

[69] A. Tiwari, J. Iaconis, J. Jojo, S. Ray, M. Roetteler, C. Hill, J. Pathak, Algorithmic advances towards a realizable quantum lattice boltzmann method, arXiv preprint arXiv:2504.10870 (2025).

[70] D. H. Rothman, S. Zaleski, Lattice-gas cellular automata: Simple models of complex hydrodynamics, Cambridge University Press, 1997.

[71] J.-P. Rivet, J. P. Boon, Lattice Gas Hydrodynamics, Cambridge University Press, 2001.

[72] A. Mohamad, Lattice Boltzmann Method, volume 70, Springer, 2011.
[73] B. Servan-Camas, F. T.-C. Tsai, Non-negativity and stability analyses of lattice Boltzmann method for advection–diffusion equation, Journal of Computational Physics **228** (2009) 236–256.
[74] I. Ginzburg, Equilibrium-type and link-type lattice Boltzmann models for generic advection and anisotropic-dispersion equation, Advances in Water Resources **28** (2005) 1171–1195.
[75] V. Shende, S. Bullock, I. Markov, Synthesis of quantum-logic circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25** (2006) 1000–1010.
[76] 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, J. M. Gambetta, Quantum computing with Qiskit, arXiv preprint arXiv:2405.08810 (2024).
[77] M. A. Nielsen, I. L. Chuang, Quantum Computation and Quantum Information, Cambridge University Press, 2000.
[78] A. M. Childs, N. Wiebe, Hamiltonian simulation using linear combinations of unitary operations, Quantum Information & Computation **12** (2012) 901–924.
[79] S. Chakraborty, A. Gilyén, S. Jeffery, The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation, 46th International Colloquium on Automata, Languages, and Programming (ICALP 2019) (2019).
[80] G. H. Low, I. L. Chuang, Hamiltonian simulation by qubitization, Quantum **3** (2019) 163.
[81] S. S. Bullock, I. L. Markov, Asymptotically optimal circuits for arbitrary n-qubit diagonal computations, arXiv preprint quant-ph/0303039 (2008).
[82] S. Zhang, K. Huang, L. Li, Depth-optimized quantum circuit synthesis for diagonal unitary operators with asymptotically optimal gate count, Physical Review A **109** (2024) 042601.
[83] J. Zylberman, U. Nzongani, A. Simonetto, F. Debbasch, Efficient quantum circuits for non-unitary and unitary diagonal operators with space-time-accuracy trade-offs, ACM Transactions on Quantum Computing (2024).
[84] B. L. Douglas, J. Wang, Efficient quantum circuit implementation of quantum walks, Physical Review A—Atomic, Molecular, and Optical Physics **79** (2009) 052335.
[85] G. K. Batchelor, An introduction to fluid dynamics, Cambridge University Press, 1967.
[86] D. A. Meyer, Quantum mechanics of lattice gas automata: One-particle plane waves and potentials, Physical Review E **55** (1997) 5261.
[87] X. Yu, J. Shang, O. Gühne, Statistical methods for quantum state verification and fidelity estimation, Advanced Quantum Technologies **5** (2022).
[88] B. Park, D. Ahn, Reducing CNOT count in quantum Fourier transform for the linear nearest-neighbor architecture, Scientific Reports **13** (2023).
[89] V. Shende, S. Bullock, I. Markov, Synthesis of quantum-logic circuits, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **25** (2006) 1000–1010.
[90] V. T. Hai, L. B. Ho, Universal compilation for quantum state tomography, Scientific Reports **13** (2023).
[91] Y. Quek, S. Fort, H. K. Ng, Adaptive quantum state tomography with neural networks, npj Quantum Information **7** (2021).
[92] A. Rocchetto, S. Aaronson, S. Severini, G. Carvacho, D. Poderini, I. Agresti, M. Bentivegna, F. Sciarrino, Experimental learning of quantum states, Quantum Information and Measurement (QIM) V: Quantum Technologies (2019).
[93] M. Cramer, M. B. Plenio, S. T. Flammia, R. Somma, D. Gross, S. D. Bartlett, O. Landon-Cardinal, D. Poulin, Y.-K. Liu, Efficient quantum state tomography, Nature Communications **1** (2010).
[94] S. Aaronson, The learnability of quantum states, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences **463** (2007) 3089–3114.

Appendix A. List of Abbreviations

CFD	computational fluid dynamics
HHL	Harrow-Hassidim-Lloyd algorithm
LBM	lattice Boltzmann method
LGA	lattice gas automata
QLBM	quantum lattice Boltzmann method
NISQ	noisy intermediate-scale quantum
VQA	variational quantum algorithm
CX	controlled-X gate (two-qubit gate)
ECR	echoed cross-resonance gate (two-qubit gate)

Appendix B. List of Variables

c	advection coefficient
D	diffusion coefficient
$\phi(t, x)$	scalar concentration field
α	link index
f_α	particle distribution along α
e_α	velocity of particle in α
t	time
r	cell position
S	source term
w_α	proportion of particles streaming in link α
τ	relaxation time
n	dimension of lattice grid
m	number of directions of particle propagation, or links, given by $\|\alpha\|$
f_α^{eq}	equilibrium distribution
$\vec{c}$	advection velocity vector
c_s	speed of sound
L	left shift quantum gate
R	right shift quantum gate
r_1, r_2	quantum registers with $\log_2(M)$ qubits
M	number of lattice sites
d	quantum register with $\lceil \log_2(m)$ qubits, corresponding to the links
a	quantum register with ancilla
A	collision coefficient operator
C_1, C_2	unitary collision operators
N_{qubit}	total number qubits
H	Hadamard gate
ω	vorticity
ψ	stream function
x, y	Cartesian coordinate directions
$g_\alpha^{(eq)}$	equilibrium distribution function for the Poisson equation in link α
g_α	particle distribution function for the Poisson equation in link α
ω	spatial domain in x, y space
L_x	x-length of spatial domain ω
L_y	y-length of spatial domain ω
U	top lid value, i.e. boundary condition
ρ	concentration at given site
$N_{\text{samp.}}$	number sample sizes
F_ψ	quantum state fidelity

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: