frontiers | Frontiers in Mechanical Engineering

TYPE Original Research
PUBLISHED 16 January 2026
DOI 10.3389/fmech.2025.1717775

OPEN ACCESS

EDITED BY
Zhenhua Chai,
Huazhong University of Science and Technology, China

REVIEWED BY
Arash Shams Taleghani,
Aerospace Research Institute, Ministry of Science, Research and Technology, Tehran, Iran
Xiaodong Niu,
Shantou University, China
Ming Li,
China University of Petroleum (East China), China

*CORRESPONDENCE
Zhengliang Liu,
zhengliang.liu@stfc.ac.uk

RECEIVED 02 October 2025
REVISED 04 November 2025
ACCEPTED 05 November 2025
PUBLISHED 16 January 2026

CITATION
Liu Z, John B and Emerson DR (2026) Quantum Lattice Boltzmann Method based on linear equilibrium distribution functions.
Front. Mech. Eng. 11:1717775.
doi: 10.3389/fmech.2025.1717775

Quantum Lattice Boltzmann Method based on linear equilibrium distribution functions

Zhengliang Liu*, Benzi John and David R. Emerson

Scientific Computing Department, STFC Daresbury Laboratory, Warrington, United Kingdom

In this paper, we propose a complete formulation of the Lattice Boltzmann Method adapted for quantum computing. The classical collision, based on linear equilibrium distribution functions and streaming steps, are reformulated as linear algebraic operations. The inherently non-unitary collision operator is decomposed using Singular Value Decomposition and the Linear Combination of Unitaries technique. Bounce-back boundary conditions are incorporated directly into the collision matrix, while the streaming step is realized through conditional unitary shift operations on spatial registers, controlled by lattice velocity indices encoded in the distribution function register. This formulation ensures that the streaming step remains purely unitary. The resulting quantum circuit is implemented using Qiskit and validated against Couette flow and Poiseuille flow benchmarks. The simulation accurately reproduces the expected velocity profile, with relative errors below 10^{-4}. This work establishes a foundational framework for quantum fluid solvers and provides a pathway toward quantum computational fluid dynamics.

KEYWORDS

quantum computation, CFD, Lattice Boltzmann (LB) Method, boundary condition, QISKit

1 Introduction

Quantum computing offers fundamentally new computational capabilities by exploiting the principles of quantum mechanics, such as superposition, interference, and entanglement. These properties enable quantum systems to process and store information in ways that are inaccessible to classical computers, achieving potential speedups ranging from polynomial to exponential in specific applications, owing to the ability to operate on exponentially large state spaces in superposition (Bharadwaj, 2024). Quantum algorithms are expected to perform better than their classical counterparts in tasks such as optimization, search, and simulating physical systems. However, the quantum devices that are currently available remain limited by decoherence and circuit depth, making the practical implementation of complex algorithms challenging. Nonetheless, advances in quantum hardware and control makes possible the near-term feasibility of quantum algorithms for meaningful scientific applications. This motivates the development of quantum algorithms that not only offer theoretical advantages but also operate efficiently.

Computational Fluid Dynamics (CFD) has long been dominated by classical numerical methods for solving the Navier–Stokes and related partial differential equations. Turbulent flow simulations, in particular, demand significant computational resources due to the wide range of temporal and spatial scales involved. Quantum computing promises to alleviate some of these challenges, such

as high Reynolds number flow, by exploiting its parallelism and high-dimensional state representation. Initial quantum research has focused primarily on solving linear systems, such as the Poisson and advection-diffusion equations (Esmaeilifar et al., 2024). However, real-world fluid dynamics is fundamentally nonlinear and dissipative, presenting conceptual challenges for quantum algorithms. Quantum mechanics prohibits straightforward computation of quadratic terms like u^2 by storing temporary copies (no-cloning theorem), and imposes unitary (i.e., reversible) dynamics, whereas viscous dissipation in fluids is inherently irreversible. When dealing with nonlinear problems, some studies have attempted to design quantum algorithms based on linearization techniques, such as Carleman linearization (Liu et al., 2021), and the Fokker-Planck equation (Tennie and Magri, 2024). However, these methods require mapping the original problem to a higher-dimensional space, and quantum algorithms can only achieve effective speedup when the dimension is a polynomial function of the original problem's dimension.

One promising route to quantum CFD is the Quantum Lattice Boltzmann Method (QLBM). Classical Lattice Boltzmann Methods (LBM) simulate fluid dynamics in terms of discrete particle distributions via collision and streaming steps on discrete lattices, thus avoiding the explicit discretization of the Navier–Stokes equations (Liu et al., 2017). The inherently linear structure of the LBM propagation operation makes it aligned with quantum computational models. Quantum LBM studies have begun to address both the encoding of particle distributions and the execution of collision/streaming steps using a quantum algorithm.

Ljubomir (Budinski, 2022) pioneered the use of standard-form encoding for the collision and boundary-condition operations, while employing quantum-walk protocols to realize the streaming step. Wawrzyniak et al., (2025) extended this by introducing general quantum building blocks for initialization, collision, and streaming that are adaptable to arbitrary lattice-velocity sets in one, two, and three dimensions, and demonstrated their algorithm on non-uniform advection problems. In that work, both collision and streaming were expressed as linear algebraic transformations, and the velocity field was externally prescribed. Xu et al. (2025) proposed an ancilla-free QLBM formulation for advection-diffusion equations, using local unitary operations to reduce circuit depth and eliminate the need for quantum tomography in each simulation loop. Kocherla et al. (2024) proposed a two-circuit QLBM algorithm for solving the 2D Navier–Stokes equations in a stream function–vorticity form. By separating the stream function and vorticity into distinct quantum circuits, the method achieves reductions in both gate count and circuit depth. Kumar and Frankel (Kumar and Frankel, 2025) incorporated boundary conditions within the streaming step, which required treating both collision and streaming as non-unitary processes. This necessitated separate Singular Value Decomposition (SVD) and Linear Combination of Unitaries (LCU) decompositions for each operation, leading to increased circuit depth and reduced computational efficiency.

To address nonlinearity in the collision step, Steijl (Steijl, 2023) focused on encoding nonlinear equilibrium distributions using quantum floating-point arithmetic, analyzing trade-offs between circuit width and numerical precision. Wang et al. (2025) introduced a node-level ensemble formulation by transforming low-dimensional nonlinear fluid systems into medium-dimensional linear lattice gas representations. To address deviations from physical constraints during ensemble transformations, they introduce an auxiliary H-step to maintain near-equilibrium velocity distributions. In addition, Carleman linearization techniques have been effectively applied to nonlinear collisions by Itani et al. (2024), and further developed by Claudio and Sauro (Sanavio and Succi, 2024; Sanavio et al., 2025), allowing quantum-compatible treatment of nonlinear LBM dynamics. Zeng et al. (2025) developed a hybrid quantum lattice Boltzmann method that uses a linearized non-equilibrium collision operator to preserve unitarity while maintaining accuracy and enable adjustable relaxation parameters for different flow regimes.

Software and hardware developments have further promoted the QLBM framework. Georgescu et al. (2025) introduced a Python-based framework that integrates quantum circuit generation, simulation, and performance analysis into a unified platform for facilitating rapid prototyping and deployment of QLBM algorithms. Tiwari et al. (2025) demonstrated the first realization of QLBM on actual quantum hardware. Their implementation successfully simulated the time evolution of a 2D Gaussian density distribution under advection-diffusion, and was extended to 3D flow fields with non-uniform advection, representing a significant milestone in the path toward practical quantum CFD.

Based on the foundations established in prior studies, this paper presents a novel Quantum Lattice Boltzmann Method (QLBM) framework built on linear equilibrium distribution functions. We reformulate both the classical collision and streaming steps into quantum operations. The collision operator is expressed in a linear algebraic form, making it suitable for decomposition using Singular Value Decomposition (SVD) and implementation using the Linear Combination of Unitaries (LCU) method. Bounce-back boundary conditions are directly integrated into the collision operator, while imposed wall velocities are encoded within the distribution function register. This approach ensures that the streaming step remains fully unitary, as it is implemented through conditional cyclic shift operations controlled by lattice velocity indices. The quantum circuits are implemented using Qiskit and validated against benchmark Couette flow and Poiseuille flow cases, where analytical velocity profiles are available for direct comparison. The aim of this research is to establish a theoretically sound and efficient QLBM framework, with validation against classical benchmark cases. The primary objective is to demonstrate algorithmic feasibility and physical accuracy while achieving reduced circuit depth compared to prior QLBM formulations. From a computational perspective, maintaining unitary evolution not only ensures physical reversibility but also minimizes circuit depth, thereby improving algorithmic scalability on near-term and fault-tolerant quantum architectures. The remainder of the paper is organized as follows: Section 2 introduces the fundamentals of the classical LBM. Section 3 details the quantum implementation of LBM, including encoding strategies (Section 3.2), construction of the collision operator (Section 3.3), treatment of boundary conditions (Section 3.4), and the streaming operation (Section 3.6). Numerical validation results are presented in Section 4. Finally, conclusions and directions for future research are discussed.

2 The lattice Boltzmann methods

The Lattice Boltzmann Method is a mesoscopic numerical approach for simulating fluid dynamics, particularly effective for complex boundaries and multiphysics problems. It models fluid flow by tracking the evolution of particle distribution functions on a discrete lattice. The fundamental equation with the Bhatnagar–Gross–Krook (BGK) collision operator and without external forcing, is written as:

$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) - f_i(\mathbf{x},t) = -\frac{1}{\tau}\left[f_i(\mathbf{x},t) - f_i^{\mathrm{eq}}(\mathbf{x},t)\right], \quad (1)$$

where $f_i(\mathbf{x},t)$ is the distribution function in the ith lattice direction at position $\mathbf{x}$ and time t, and $\mathbf{c}_i$ are the discrete lattice velocities. The right-hand side models relaxation toward an equilibrium distribution f_i^{eq} over a characteristic relaxation time τ, which is related to the kinematic viscosity ν and the lattice speed of sound c_s as:

$$\tau = \frac{\nu}{c_s^2} + \frac{1}{2}.$$

Equation 1 is typically solved in two consecutive steps (Liu et al., 2022): the collision step and the streaming step, expressed respectively as:

$$f_i^{\star}(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{1}{\tau}\left[f_i(\mathbf{x},t) - f_i^{\mathrm{eq}}(\mathbf{x},t)\right],$$
$$f_i(\mathbf{x}+\mathbf{c}_i\Delta t, t+\Delta t) = f_i^{\star}(\mathbf{x},t).$$

Here, $f_i^{\star}(\mathbf{x},t)$ represents the post-collision distribution function. The equilibrium distribution function f_i^{eq} plays a central role in linking the microscopic particle dynamics to macroscopic fluid behavior.

When the viscous effects dominate over advection, the equilibrium distribution can be approximated using a linearized distribution:

$$f_i^{\mathrm{eq}}(\mathbf{x},t) = w_i\left(\rho + \rho_0\frac{\mathbf{c}_i\mathbf{u}}{c_s^2}\right), \quad (2)$$

where w_i are lattice weights specific to the lattice model used (e.g., D2Q9), ρ is the macroscopic fluid density, ρ_0 is a reference density, and $\mathbf{u}$ is the macroscopic velocity.

After each time step, the macroscopic fluid properties are recovered by taking moments of the distribution functions:

$$\rho(\mathbf{x},t) = \sum_i f_i(\mathbf{x},t),$$
$$\mathbf{u}(\mathbf{x},t) = \frac{1}{\rho_0}\sum_i f_i(\mathbf{x},t)\mathbf{c}_i. \quad (3)$$

In this study, the two-dimensional nine-velocity model (D2Q9) is employed, which consists of one rest particle, four particles moving along the coordinate axes, and four particles moving diagonally.

To impose no-slip boundary conditions at solid walls, the bounce-back scheme is utilized. Specifically, on the top and bottom boundaries, a prescribed velocity $\mathbf{u}_w$ is imposed at the midpoints between boundary and fluid nodes using the bounce-back with velocity correction. The boundary condition is formulated as:

$$f_{\bar{i}}(\mathbf{x}_b,t) = f_i^{\star}(\mathbf{x}_b,t) - 2w_i\rho_w\frac{\mathbf{c}_i\mathbf{u}_w}{c_s^2}, \quad (4)$$

where $\mathbf{x}_b$ denotes a boundary node location, $\bar{i}$ indicates the direction opposite to i (i.e., $\mathbf{c}_{\bar{i}} = -\mathbf{c}_i$), and ρ_w is the fluid density at the wall. This corrected bounce-back approach allows enforcement of moving or stationary wall conditions with second-order accuracy.

3 Quantum lattice Boltzmann method implementation

The Quantum Lattice Boltzmann Method maps the classical Lattice Boltzmann dynamics onto a quantum computer to simulate fluid dynamics in a potentially more efficient manner compared to classical LBM. In particular, QLBM exploits amplitude encoding to represent the high-dimensional distribution functions within the amplitudes of a quantum state and performing the collision and streaming steps via unitary (and, when necessary, ancilla-assisted non-unitary) operations. In this section, we describe the encoding strategy, the quantum implementation of the collision operator via singular value decomposition (SVD) and the linear combination of unitaries (LCU) method, the treatment of bounce-back boundary conditions, and the streaming step via conditional bit-shifts.

3.1 Quantum LBM workflow

The quantum lattice Boltzmann method simulation is implemented using a state vector-based quantum circuit model. The overall procedure is outlined below in pseudocode:

```
Initialize quantum LBM circuit
Compute initial distribution functions
Compute collision matrix
Set bounce-back boundary conditions in collision matrix
for t = 0 to n_step - 1 do
  Encode initial field into quantum state: scale_f ← InitialEncoding(f_ini)
  Apply collision operator: scale_C ← MatrixMultiplier
  Apply streaming operator: Stream
  Compile and simulate circuit: result ← simulator.run
  Extract state vector: ψ ← result.get_statevector()
  Update field: f_ini ← ReconstructField(ψ, scale_f, scale_C)
end for
```

Algorithm 1. Quantum LBM Simulation

- InitialEncoding: Maps the distribution functions into a quantum state using amplitude encoding.
- MatrixMultiplier: Applies the collision operator using singular value decomposition (SVD) and linear combination of unitaries (LCU). This involves:
  - Decomposing the collision matrix into unitary components U, Σ, and $V^{\dagger}$.
  - Constructing ancilla-controlled phase rotations based on normalized singular values.
  - Applying extended unitary matrices to the quantum register.
- Stream: Implements conditional shift operations that mimic particle propagation in the lattice. These are realized using multi-controlled unitary gates conditioned on velocity directions.

3.2 Encoding of distribution functions

In this implementation, four quantum registers are to used to encode distribution functions of a D2Q9 model. The fluid domain is discretized into a mesh of $n_x \times n_y$ nodes, with each node holding 9 discrete velocity distribution functions (as in the D2Q9 model). These distributions are encoded into quantum registers as follows:

- q_x: Position register in the x-direction, requiring $\log_2(n_x)$ qubits.
- q_y: Position register in the y-direction, requiring $\log_2(n_y)$ qubits.
- q_f: Distribution function register, requiring $\log_2(9) = 4$ qubits to encode the 9 discrete velocity directions.
- q_a: Ancilla qubit for implementing non-unitary operations via linear combinations of unitaries.

The total number of qubits required is $\log_2(n_x n_y) + 5$. Before encoding, the classical distribution vector $\mathbf{f} \in \mathbb{R}^{9n_x n_y}$ is normalized to form a valid quantum state:

$$\hat{\mathbf{f}} = \frac{\mathbf{f}}{\|\mathbf{f}\|_2}.$$

The quantum state after amplitude encoding becomes:

$$|\phi\rangle = \sum_{k=0}^{n_x n_y - 1} \sum_{i_q=0}^{8} \hat{f}_{i_q,k} |0\rangle_a |i_q\rangle_f |k\rangle_{x,y},$$

where $|k\rangle_{x,y}$ denotes the spatial position encoded by registers q_x and q_y, and $\hat{f}_{i_q,k}$ is the normalized distribution function for direction i_q at node k.

3.3 Collision

In the classical LBM, the collision step updates the distributions based on local equilibrium. By substituting the macroscopic quantities ρ and $\mathbf{u}$ using the moment relations in Equation 3, the equilibrium distribution in Equation 2 can be written in terms of the distribution functions:

$$f_i^{\mathrm{eq}} = \sum_j w_i \left(1 + \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{c_s^2}\right) f_j.$$

The post-collision distribution function is then given by:

$$f_i^{\star} = \left(1 - \frac{1}{\tau}\right) f_i + \frac{1}{\tau} \sum_j w_i \left(1 + \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{c_s^2}\right) f_j = \sum_j C_{ij} f_j,$$

where the collision matrix C_{ij} is defined as:

$$C_{ij} = \delta_{ij}\left(1 - \frac{1}{\tau}\right) + \frac{w_i}{\tau}\left(1 + \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{c_s^2}\right),$$

and δ_{ij} is the Kronecker delta. For the D2Q9 lattice model, the discrete velocity set are explicitly defined as: $\mathbf{c}_x = [0, 1, 0, -1, 0, 1, -1, -1, 1]$ and $\mathbf{c}_y = [0, 0, 1, 0, -1, 1, 1, -1, -1]$, with corresponding lattice weights $w = [4/9,\ 1/9,\ 1/9,\ 1/9,\ 1/9,\ 1/36,\ 1/36,\ 1/36,\ 1/36]$. The lattice sound speed is $c_s = 1/\sqrt{3}$. The full matrix representation of the collision operation $\mathbf{Af}$ across all velocity directions is given by:

$$\mathbf{Af} = \begin{bmatrix} C_{00} & \cdots & C_{0n_q} & 0 & \cdots & 0 & & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & & \vdots & \ddots & \vdots \\ C_{n_q 0} & \cdots & C_{n_q n_q} & 0 & \cdots & 0 & & 0 & \cdots & 0 \\ 0 & \cdots & 0 & & & & & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \ddots & & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & & & & & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & \cdots & 0 & C_{00} & \cdots & C_{0n_q} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & C_{n_q 0} & \cdots & C_{n_q n_q} \end{bmatrix} \begin{bmatrix} f_0^0 \\ \vdots \\ f_{nq}^0 \\ f_0^k \\ \vdots \\ f_{nq}^k \\ f_0^n \\ \vdots \\ f_{nq}^n \end{bmatrix}$$

The collision operator $\mathbf{A}$ is non-unitary and cannot be directly implemented on a quantum computer. To address this, we employ the Singular Value Decomposition and the Linear Combination of Unitaries technique (Kumar and Frankel, 2025).

First, we perform SVD on the collision matrix:

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\dagger},$$

where $\mathbf{U}$ and $\mathbf{V}$ are unitary matrices, $\mathbf{V}^{\dagger}$ is the conjugate transpose of $\mathbf{V}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix with non-negative real singular values. The unitary matrices $\mathbf{U}$ and $\mathbf{V}$ can be implemented directly on a quantum computer.

To implement $\boldsymbol{\Sigma}$, we use the LCU method (Childs and Wiebe, 2012), which decomposes it as:

$$\boldsymbol{\Sigma} = \frac{1}{2}(\mathbf{B}_1 + \mathbf{B}_2),$$

where:

$$\mathbf{B}_1 = \boldsymbol{\Sigma} + i\sqrt{\mathbf{I} - \boldsymbol{\Sigma}^2}, \quad \mathbf{B}_2 = \boldsymbol{\Sigma} - i\sqrt{\mathbf{I} - \boldsymbol{\Sigma}^2},$$

and both $\mathbf{B}_1$ and $\mathbf{B}_2$ are unitary matrices. The LCU is implemented using a Hadamard gate on the ancilla qubit, followed by controlled-unitary gates conditioned on the ancilla state, and a final Hadamard gate to complete the linear combination. The full quantum circuit for the collision step is illustrated in Figure 1. During the decomposition of the collision matrix using SVD and the construction of the LCU components, numerical noise can arise due to floating-point precision and small singular values. To address this, we normalize the singular values, and extend unitary matrices with identity padding to preserve unitarity. These measures ensure the reliability of the quantum circuit under state vector simulation.

3.4 Boundary condition

Unlike previous QLBM implementations where the bounce-back boundary condition was incorporated into the streaming step, leading to a non-unitary streaming operator, we incorporate the bounce-back boundary condition, given by Equation 4, into the collision matrix, $\mathbf{A}$. This ensures that the streaming step remains purely unitary and is handled separately. The bounce-back method replaces incoming distribution functions at the boundary with their oppositely directed outgoing post-collision counterparts.

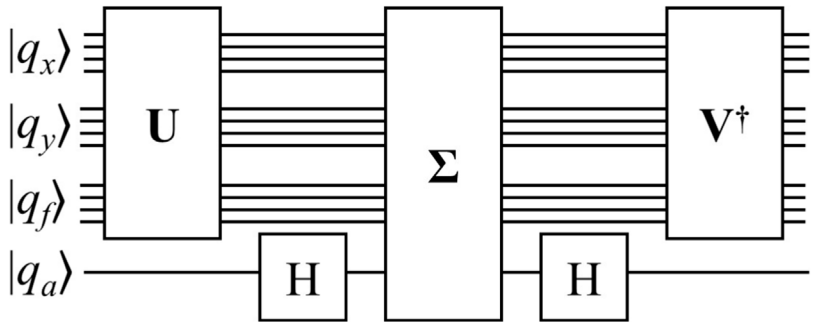

FIGURE 1
Quantum circuit for the collision operation.

To impose this within a quantum circuit, we modify the collision matrix coefficients at the corresponding mirrored boundary. For instance, to enforce bounce-back on the top boundary, we alter the matrix coefficients for the bottom boundary accordingly. This avoids additional gates during the quantum streaming operation.

To implement a constant wall velocity, $\mathbf{u}_w$, its components are encoded into unused amplitude states in the q_f register. For the D2Q9 model, 4 qubits (16 states) are available, of which 7 states ($|1001\rangle$ to $|1111\rangle$) are unused. These can be exploited to embed $\mathbf{u}_w$. For example, consider a grid with $n_x = 1$, $n_y = 3$, and a top boundary velocity $\mathbf{u}_w = [u_w, 0]$. The complete matrix for the D2Q9 model becomes:

$$\mathbf{Af} = \begin{bmatrix}
C_{00} & \cdots & C_{08} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
C_{30} & \cdots & C_{38} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\
0 & \cdots & 0 & 0 & \cdots & 0 & C_{20} & \cdots & C_{28} & -w_2\rho_w\frac{c_{x2}}{c_s^2} \\
C_{50} & \cdots & C_{58} & \vdots & \ddots & \vdots & 0 & \cdots & 0 & \vdots \\
C_{60} & \cdots & C_{68} & 0 & \cdots & 0 & 0 & \cdots & 0 & 0 \\
0 & \cdots & 0 & 0 & \cdots & 0 & C_{50} & \cdots & C_{58} & -w_5\rho_w\frac{c_{x5}}{c_s^2} \\
0 & \cdots & 0 & 0 & \cdots & 0 & C_{60} & \cdots & C_{68} & -w_6\rho_w\frac{c_{x6}}{c_s^2} \\
0 & \cdots & 0 & C_{00} & \cdots & C_{08} & 0 & \cdots & 0 & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & C_{80} & \cdots & C_{88} & 0 & \cdots & 0 & 0 \\
0 & \cdots & 0 & 0 & \cdots & 0 & C_{00} & \cdots & C_{08} & 0 \\
\vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\
0 & \cdots & 0 & 0 & \cdots & 0 & C_{80} & \cdots & C_{88} & 0 \\
0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & 1
\end{bmatrix}
\begin{bmatrix}
f_0^{0,0} \\ \vdots \\ f_3^{0,0} \\ f_4^{0,0} \\ f_5^{0,0} \\ f_6^{0,0} \\ f_7^{0,0} \\ f_8^{0,0} \\ f_0^{0,1} \\ \vdots \\ f_8^{0,1} \\ f_0^{0,2} \\ \vdots \\ f_8^{0,2} \\ u_w
\end{bmatrix}$$

3.5 Forces

For steady, incompressible flows, the forcing term expanded to first order in velocity space can be expressed as

$$F_i = w_i\frac{\mathbf{c}_i}{c_s^2}\mathbf{F},$$

where $\mathbf{F}$ denotes the force density. With trapezoidal discretization, the collision step becomes

$$f_i^\star(\mathbf{x},t) = f_i(\mathbf{x},t) - \frac{1}{\tau}\left[f_i(\mathbf{x},t) - f_i^{\mathrm{eq}}(\mathbf{x},t)\right] + \left(1 - \frac{1}{2\tau}\right)F_i\Delta t,$$

and the macroscopic velocity is updated as

$$\mathbf{u}(\mathbf{x},t) = \frac{1}{\rho_0}\sum_i f_i(\mathbf{x},t)\mathbf{c}_i + \frac{\Delta t}{2\rho_0}\mathbf{F}.$$

Following the same procedure as in Section 3.3, the post-collision distribution function can be expressed in matrix form as

$$f_i^\star = \sum_j C_{ij}f_j + w_i\frac{\Delta t}{c_s^2}\mathbf{c}_i\mathbf{F},$$

where C_{ij} is the collision matrix defined previously in Section 3.3. In the quantum setting, the force density $\mathbf{F}$ is encoded into the unused amplitude states of the q_f register, in a manner analogous to the wall-velocity encoding described above.

3.6 Stream

The streaming step corresponds to particle propagation along discrete lattice directions. In the quantum setting, this is implemented using conditional shift operations controlled by the distribution index q_f.

The positive (**P**, see Figure 2a) and negative (**N**, see Figure 2b) cyclic shift operators under periodic boundary conditions are defined as (Budinski, 2021):

$$\mathbf{P} = \sum_{i=0}^{2^M-1} |(i+1) \bmod 2^M\rangle\langle i|,$$
$$\mathbf{N} = \sum_{i=0}^{2^M-1} |i\rangle\langle(i+1) \bmod 2^M|.$$

The actual shifts are applied only on registers q_x or q_y, conditioned on the direction encoded in q_f. For example, distribution f_6 with velocity $\mathbf{c}_8 = [1,-1]$ requires a positive shift on q_x and a negative shift on q_y. This is implemented using multi-

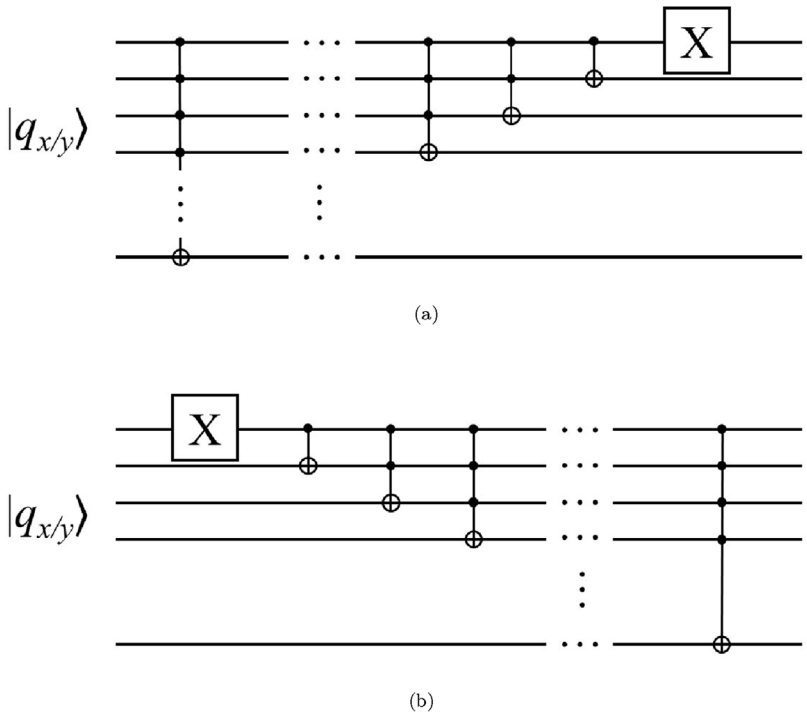

FIGURE 2
Quantum circuit for cyclic positive and negative shift operations: **(a)** Positive shift operator (P). **(b)** Negative shift operator (N).

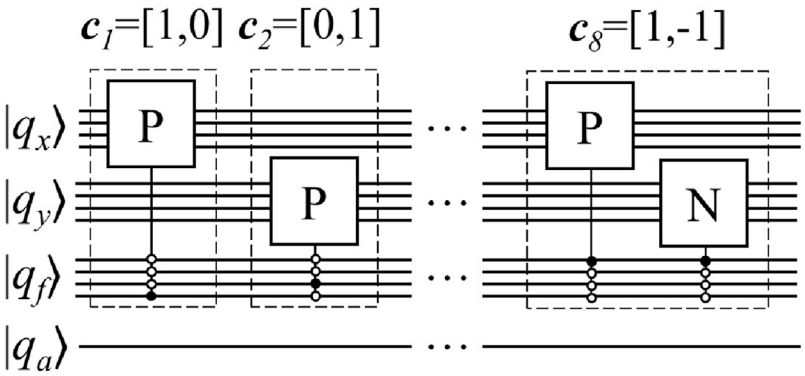

FIGURE 3
Quantum circuit for cyclic positive and negative shift operations.

controlled gates, targeting the basis state $|1000\rangle_f$. The full quantum circuit for the streaming step is shown in Figure 3.

4 Validation

The quantum LBM is implemented using the Qiskit toolbox (version 1.4.2) developed by IBM (Javadi-Abhari et al., 2024) with the StatevectorSimulator, which deterministically evolves the quantum state. The real part of the state vectors obtained from the deterministic state vector simulator is extracted and used to reinitialize the quantum state at each subsequent time step.

To validate the implementation, we consider the classical plane Couette flow problem. This choice is particularly suitable since the linear equilibrium distribution function assumption neglects nonlinear terms, and therefore does not recover the full Navier–Stokes equations with convection. The computational domain consists of $n_x \times n_y = 8 \times 8$ lattice nodes, with periodic boundary conditions along the

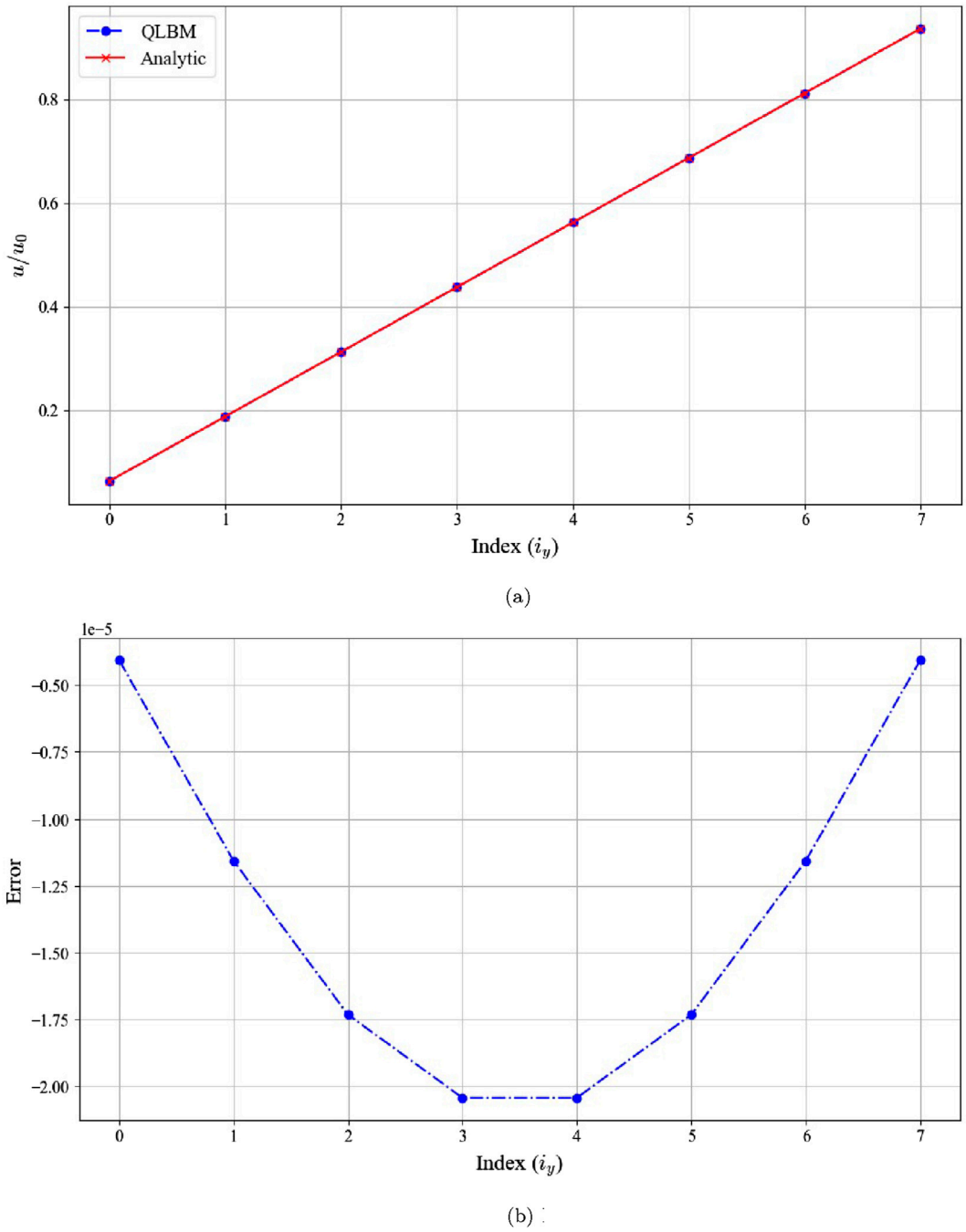

FIGURE 4
Normalized velocity and relative error along the y-direction for Couette flow: **(a)** Normalized velocity. **(b)** Relative error.

x-direction. The top and bottom boundaries are modeled as no-slip walls, with velocities $u_w = 0.02$ (top) and 0 (bottom), respectively. The simulation parameters are as follows: relaxation time $\tau = 0.9$, grid spacing $\Delta x = 1$, time step $\Delta t = 1$, and a total of 500 time steps. The distribution function is initialized as $f_i = w_i$, corresponding to a static flow field with zero velocity and unit density.

Due to the bounce-back boundary condition, which imposes the wall velocity at the mid-point of the boundary node, the analytical velocity profile for Couette flow is given by:

$$u_x^a(i_y) = u_w\left(\frac{i_y}{n_y} + 0.5\right),$$

where i_y is the index in the y-direction. As shown in Figure 4, the velocity profile predicted by the QLBM closely matches the analytical solution. The relative difference $(u_x - u_x^a)/u_w$ remains below 10^{-4} across the domain, confirming the accuracy of the quantum LBM implementation.

We further validate the quantum LBM implementation against plane Poiseuille flow, where forcing terms are included. The setup is

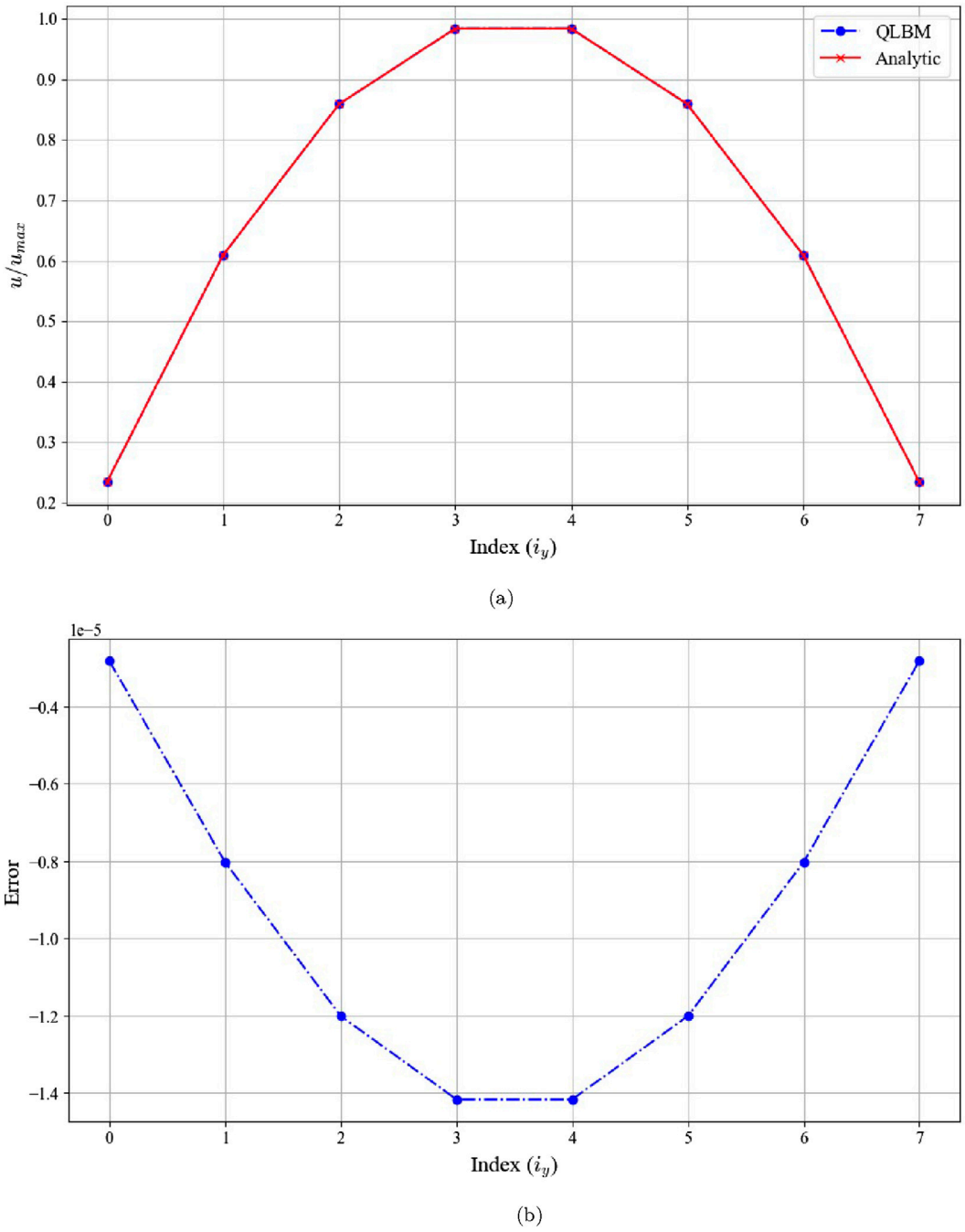

FIGURE 5
Normalized velocity and relative error along the y-direction for Poiseuille flow: **(a)** Normalized velocity. **(b)** Relative error.

identical to the Couette case, except that both the top and bottom boundaries are stationary, and the relaxation time is set to $\tau = \sqrt{3/16} + 0.5$.

The analytical velocity profile for Poiseuille flow is given by

$$u_x^a(i_y) = -\frac{4u_{\max}}{n_y^2}(i_y + 0.5)(i_y + 0.5 - n_y),$$

where $u_{\max} = 0.1$. As shown in Figure 5, the QLBM accurately reproduces the parabolic velocity profile, with the predicted solution matching the analytical curve. The normalized error $(u_x - u_x^a)/u_{\max}$ remains below 10^{-4} across the domain, demonstrating that the quantum LBM formulation accurately reproduces the parabolic velocity profile of Poiseuille flow.

To assess scalability, we compare the quantum circuit characteristics for Couette and Poiseuille flows at two grid resolutions: $n_x = n_y = 8$ and $n_x = n_y = 16$. For the smaller grid, the quantum circuit has a depth of 5164, 11 qubits, and circuit size of 14,416. For the larger grid, the depth increases to 20,534, with 13 qubits and circuit size of 65,628. This growth in circuit size and depth reflects the increased computational cost associated with

encoding more distribution functions. The number of qubits grows logarithmically with the number of nodes, while the depth and size scale with the number of operations required for collision and streaming steps.

5 Conclusion

In this work, we presented a quantum lattice Boltzmann method framework by reformulating the classical collision and streaming steps for implementation on quantum hardware. The collision step was recast as a pure matrix multiplication applied to all distribution functions across the computational domain. To handle the non-unitary nature of the collision operator, we decomposed the collision matrix using Singular Value Decomposition and implemented it via the Linear Combination of Unitaries method, enabling compatibility with quantum circuits.

The bounce-back boundary condition was incorporated directly into the collision matrix, and the imposed wall velocity was encoded within the distribution function register. This approach allows the streaming step to remain unitary, in contrast to the method of Kumar and Frankel (Kumar and Frankel, 2025), which embeds the wall boundary in the streaming step and consequently requires solving another non-unitary operator. In our formulation, the streaming step was implemented as conditional cyclic shift operations on the spatial registers, controlled by the lattice velocity indices encoded in the distribution function register.

The complete quantum circuit was implemented using Qiskit and simulated using a deterministic state vector simulator to perform iterative time evolution. Validation against the plane Couette flow and Poiseuille flow demonstrated that the quantum LBM accurately reproduces expected velocity profiles, with relative errors below 10^{-4}. The Couette flow benchmark confirms the correct handling of linear velocity profiles, while the Poiseuille flow demonstrates that the method accurately incorporates forcing terms. These results collectively confirm the correctness of the proposed quantum LBM framework under simplified flow conditions.

The current formulation is limited to single-phase and linearized flow regimes. Extending quantum LBM to multiphase or interface-driven problems remains an open challenge. These phenomena typically require nonlinear collision operators and interface tracking mechanisms (e.g., color-gradient methods) (Noori et al., 2021; Sheikholeslam Noori et al., 2019), which are difficult to implement on quantum hardware due to the inherent linearity of quantum mechanics. Future work may explore hybrid quantum–classical approaches to approximate nonlinear behavior, potentially enabling quantum simulations of more complex fluid systems.

Additionally, while current results are limited to running simulations of the quantum algorithms (via Qiskit) on a classical electronic computer, rather than execution on actual quantum hardware, the proposed formulation provides a foundational structure for quantum LBM algorithms. The current implementation requires full reinitialization using state vector extraction at each time step. However, this process could potentially be replaced by mid-circuit measurement followed by post-selection: retaining the state when the ancilla qubit is measured in the $|0\rangle$ state and discarding outcomes where it is $|1\rangle$, thereby removing garbage terms introduced by the LCU method. One limitation of this approach is the decreasing success probability over time, as the number of successful post-selections diminishes during successive iterations. In such scenarios, amplitude amplification techniques could be employed to improve the likelihood of retaining valid states.

Even though a direct complexity comparison with classical LBM is not straightforward due to the fundamentally different encoding schemes, quantum algorithms may offer asymptotic advantages in memory efficiency and parallelism, particularly for high-dimensional fluid systems. The simulation is performed on classical hardware without modeling quantum noise or error correction. The linearized equilibrium model restricts applicability to low Reynolds number flows. Furthermore, the encoding of complex boundary conditions and generalized forcing schemes remains a challenge. These aspects represent important directions for future research. Extensions to nonlinear flow regimes, more efficient state encoding schemes, and hybrid quantum–classical solvers represent promising directions for further development and scaling of the quantum LBM framework. The implementation code is available at: https://github.com/dugudyoudi/quantum-lbm/tree/main/linear.

Data availability statement

The datasets presented in this study can be found in online repositories. The names of the repository/repositories and accession number(s) can be found below: https://github.com/dugudyoudi/quantum-lbm/tree/main/linear%20equilibrium.

Author contributions

ZL: Conceptualization, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review and editing. BJ: Funding acquisition, Project administration, Writing – review and editing. DE: Funding acquisition, Supervision, Writing – review and editing.

Funding

The authors declare that financial support was received for the research and/or publication of this article. The authors gratefully acknowledge financial support from the Computational Science Centre for Research Communities (CoSeC) and the Collaborative Computational Project in Quantum Computing (CCP-QC), which funded this work.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The authors declare that Generative AI was used in the creation of this manuscript. Generative AI was used solely for language refinement and proofreading purposes. All scientific content, analysis, and conclusions are the original work of the authors.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers. Any product that may be evaluated in this article, or claim that may be made by its manufacturer, is not guaranteed or endorsed by the publisher.

References

Bharadwaj, S. S. (2024). QFlowS: Quantum simulator for fluid flows. *Phys. Fluids* 36 (10), 107112. doi:10.1063/5.0226074

Budinski, L. (2021). Quantum algorithm for the advection–diffusion equation simulated with the lattice Boltzmann method. *Quantum Inf. Process.* 20 (2), 57. doi:10.1007/s11128-021-02996-3

Budinski, L. (2022). Quantum algorithm for the Navier–Stokes equations by using the streamfunction-vorticity formulation and the lattice boltzmann method. *Int. J. Quantum Inf.* 20 (02), 2150039. doi:10.1142/s0219749921500398

Childs, A. M., and Wiebe, N. (2012). Hamiltonian simulation using linear combinations of unitary operations. *Quantum Inf. comput.* 12, 901–924. doi:10.26421/qic12.11-12-1

Esmaeilifar, E., Ahn, D., and Myong, R. S. (2024). Quantum algorithm for nonlinear Burgers' equation for high-speed compressible flows. *Phys. Fluids* 36 (10), 106110. doi:10.1063/5.0231994

Georgescu, C. A., Schalkers, M. A., and Möller, M. (2025). qlbm–a quantum lattice Boltzmann software framework. *Comput. Phys. Commun.* 315, 109699. doi:10.1016/j.cpc.2025.109699

Itani, W., Sreenivasan, K. R., and Succi, S. (2024). Quantum algorithm for lattice Boltzmann (QALB) simulation of incompressible fluids with a nonlinear collision term. *Phys. Fluids* 36 (1), 017112. doi:10.1063/5.0176569

Javadi-Abhari, A., Treinish, M., Krsulich, K., Wood, C. J., Lishman, J., Gacon, J., et al. (2024). Quantum computing with qiskit.

Kocherla, S., Adams, A., Song, Z., Alexeev, A., and Bryngelson, S. H. (2024). A two-circuit approach to reducing quantum resources for the quantum lattice boltzmann method.

Kumar, E. D., and Frankel, S. H. (2025). Quantum unitary matrix representation of the lattice Boltzmann model for low Reynolds fluid flow simulation. *AVS Quantum Sci.* 7 (1), 013802. doi:10.1116/5.0245082

Liu, Z., Tian, F.-B., Young, J., and Lai, J. C. S. (2017). Flapping foil power generator performance enhanced with a spring-connected tail. *Phys. Fluids* 29 (12), 123601. doi:10.1063/1.4998202

Liu, J.-P., Kolden, H. Ø., Krovi, H. K., Loureiro, N. F., Trivisa, K., and Childs, A. M. (2021). Efficient quantum algorithm for dissipative nonlinear differential equations. *Proc. Natl. Acad. Sci.* 118 (35), e2026805118. doi:10.1073/pnas.2026805118

Liu, Z., Tian, F.-B., and Feng, X. (2022). An efficient geometry-adaptive mesh refinement framework and its application in the immersed boundary lattice boltzmann method. *Comput. Methods Appl. Mech. Eng.* 392, 114662. doi:10.1016/j.cma.2022.114662

Noori, M. S., Taleghani, A. S., and Rahni, M. T. (2021). Surface acoustic waves as control actuator for drop removal from solid surface. *Fluid Dyn. Res.* 53 (4), 045503. doi:10.1088/1873-7005/ac12af

Sanavio, C., and Succi, S. (2024). Lattice Boltzmann–Carleman quantum algorithm and circuit for fluid flows at moderate reynolds number. *AVS Quantum Sci.* 6 (2), 023802. doi:10.1116/5.0195549

Sanavio, C., Simon, W. A., Ralli, A., Love, P., and Succi, S. (2025). Carleman-lattice-boltzmann quantum circuit with matrix access oracles. *Phys. Fluids* 37 (3), 037123. doi:10.1063/5.0254588

Sheikholeslam Noori, S., Taeibi Rahni, M., and Shams Taleghani, S. (2019). Multiple-relaxation time color-gradient lattice boltzmann model for simulating contact angle in two-phase flows with high density ratio. *Eur. Phys. J. Plus* 134 (8), 399. doi:10.1140/epjp/i2019-12759-x

Steijl, R. (2023). Quantum circuit implementation of multi-dimensional non-linear lattice models. *Appl. Sci.* 13 (1), 529. doi:10.3390/app13010529

Tennie, F., and Magri, L. (2024). "Solving nonlinear differential equations on quantum computers: a fokker-planck approach," in *arXiv preprint arXiv:2401.13500*.

Tiwari, A., Iaconis, J., Jojo, J., Ray, S., Roetteler, M., Hill, C., et al. (2025). Algorithmic advances towards a realizable quantum lattice boltzmann method.

Wang, B., Meng, Z., Zhao, Y., and Yang, Y. (2025). "Quantum lattice Boltzmann method for simulating nonlinear. *fluid Dyn*. arXiv preprint arXiv:2502.

Wawrzyniak, D., Winter, J., Schmidt, S., Indinger, T., Janßen, C. F., Schramm, U., et al. (2025). A quantum algorithm for the lattice-Boltzmann method advection-diffusion equation. *Comput. Phys. Commun.* 306, 109373. doi:10.1016/j.cpc.2024.109373

Xu, L., Li, M., Zhang, L., Sun, H., and Yao, J. (2025). Improved quantum lattice Boltzmann method for advection-diffusion equations with a linear collision model. *Phys. Rev. E* 111, 045305. doi:10.1103/PhysRevE.111.045305

Zeng, K.-Y., Niu, X.-D., Khan, A., Li, D.-C., and Yamaguchi, H. (2025). A quantum computing-based lattice boltzmann method with a linearized non-equilibrium collision operator and modular circuit for practical flow simulation. *Phys. Fluids* 37 (8), 081701. doi:10.1063/5.0278054