POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Quantum Algorithms for Simulating 2D Flows through Porous Media using the Lattice Boltzmann Method

RIA ANN ZACHARIAH
Département de génie mécanique

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie mécanique

Août 2025

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Quantum Algorithms for Simulating 2D Flows through Porous Media using the Lattice Boltzmann Method

présenté par **Ria Ann ZACHARIAH**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

To my family and friends. . .

ACKNOWLEDGEMENTS

I would first like to express my sincere gratitude to the members of my jury: M. Sébastien Leclaire, M. David Vidal, M. François Bertrand, and M. Heng Li. Thank you for taking the time to evaluate my work and for your valuable feedback.
This research would not have been possible without the financial support of Calcul Québec and the Natural Sciences and Engineering Research Council of Canada (NSERC). Their contributions are gratefully acknowledged.
I would like to sincerely thank M. François Bertrand, my co-supervisor, for his guidance and support throughout this project. His experience and feedback were key in shaping the direction of this work, and I am grateful for the insight he shared along the way.
I am deeply grateful to M. David Vidal, my supervisor, for his mentorship, unwavering support, and for guiding me through every step of this thesis. His approach to research, anchored in curiosity and consistency, has left a lasting impression. I will always be grateful for the invaluable lessons I have learned under his supervision.
My sincere thanks also go to M. Heng Li, for introducing me to the fundamentals of quantum computing and for his continued support in answering questions and clarifying key concepts throughout the project.
I would also like to thank all the members of my research group for their constant support, technical help, and engaging discussions, which enriched my experience at every stage of this work. To my friends, thank you for always being there and for helping in every way you could.
And finally, to my family, thank you for being the constant behind every milestone and the strength behind every challenge.

ABSTRACT

Transport phenomena in porous media, such as fluid flow, heat transfer, and mass diffusion are essential in both natural and engineered systems. Modeling these processes in complex or multiscale geometries requires advanced computational methods. The Lattice Boltzmann Method (LBM), with its mesoscopic framework and discrete treatment of space, time, and velocities, captures macroscopic behavior through local particle interactions. As speed and accuracy demands grow, interest has shifted toward quantum computing, which exploits quantum mechanic properties to offer parallelism beyond classical capabilities.

This thesis presents a quantum formulation of the Lattice Boltzmann Method (QLBM) for simulating two-dimensional incompressible Stokes flows in porous domains. A velocity–pressure based framework was developed and implemented using quantum circuits, with complete support for D2Q9 lattices including diagonal population propagation to enable full stress tensor recovery. Quantum subroutines were designed for computing zero- and first-order moments, allowing direct extraction of macroscopic quantities such as density and velocity from the quantum states.

The framework incorporates no-slip and periodic boundary conditions, embedded within the quantum logic to operate independently at each lattice node. Thorough verification and validation against classical LBM solvers showed excellent agreement, with relative mean absolute errors (RMAE) below 0.001% for density and below 0.1% for velocity fields across grid resolutions. All subroutines were verified and integrated into a modular pipeline, enabling simulation of complex porous media with varying obstacle size and density.

A detailed quantum resource analysis was performed to study the scaling behavior with domain size and geometric complexity, quantifying the impact on circuit depth, qubit count, and gate operations. Simulations were carried out using ideal statevector backends due to current hardware limitations such as decoherence, gate noise, and restricted qubit availability. Nevertheless, the algorithm is designed to harness quantum parallelism and can be adapted for execution on NISQ devices as hardware advances. This work establishes a functional and extensible QLBM solver that bridges classical numerical fluid dynamics and quantum computation.

RÉSUMÉ

Les phénomènes de transport dans les milieux poreux, tels que l'écoulement des fluides, le transfert de chaleur et la diffusion de masse, sont essentiels dans les systèmes naturels et techniques. La modélisation de ces processus dans des géométries complexes ou multi-échelles nécessite des méthodes de calcul avancées. La méthode de Boltzmann sur réseau (LBM), avec son cadre mésoscopique et son traitement discret de l'espace, du temps et des vitesses, capture le comportement macroscopique par le biais d'interactions locales entre les particules. À mesure que les exigences en matière de vitesse et de précision augmentent, l'intérêt s'est porté sur l'informatique quantique, qui exploite les propriétés de la mécanique quantique pour offrir un parallélisme dépassant les capacités classiques.
Cette thèse présente une formulation quantique de la méthode Lattice Boltzmann (QLBM) pour simuler des écoulements de Stokes incompressibles bidimensionnels dans des domaines poreux. Une méthode basée sur la vitesse et la pression a été développée et mise en œuvre en utilisant des circuits quantiques. Cette méthode propose un support complet pour les lattices de type D2Q9, incluant la propagation diagonale de la population pour permettre la récupération complète du tenseur de contrainte. Des sous-programmes quantiques ont également été conçus pour calculer les moments d'ordre zéro et un, ce qui permet d'extraire directement des quantités macroscopiques telles que la densité et la vitesse à partir des états quantiques.
Cette méthode incorpore des conditions limites périodiques sans glissement intégrées dans la logique quantique pour fonctionner indépendamment à chaque nœud de la lattice. Une vérification et une validation approfondies par rapport aux solveurs LBM classiques ont montré un excellent accord entre nos résultats quantiques et ceux obtenus par un solveur classique, avec des erreurs absolues moyennes relatives (RMAE) inférieures à 0,001 % pour la densité et inférieures à 0,1 % pour les champs de vitesse, et ce sur l'ensemble des résolutions de grille considérées. Tous les sous-programmes ont été vérifiés et intégrés dans une pipeline modulaire, ce qui permet de simuler des milieux poreux complexes dont la taille et la densité des obstacles varient.
Une analyse détaillée des ressources quantiques a été réalisée pour étudier le comportement de mise à l'échelle en fonction de la taille du domaine et de la complexité géométrique, en quantifiant l'impact sur la profondeur du circuit, le nombre de qubits et les opérations de porte. Les simulations ont été effectuées en utilisant des vecteurs d'état idéaux (sans bruit) en raison des limitations matérielles actuelles telles que la décohérence, le bruit de grille et la disponibilité restreinte des qubits. Néanmoins, l'algorithme est conçu pour exploiter le

parallélisme quantique et peut être adapté pour être exécuté sur des dispositifs NISQ au fur et à mesure des progrès du matériel. Ce travail établit un solveur QLBM fonctionnel et extensible qui fait le lien entre la dynamique des fluides numérique classique et l'informatique quantique.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

PEM	Polymer Electrolyte Membrane
CFD	Computational Fluid Dynamics
FVM	Finite Volume Method
FEM	Finite Element Method
FDM	Finite Difference Method
LBM	Lattice Boltzmann Method
LGA	Lattice Gas Automata
QC	Quantum Computer
QLBM	Quantum Lattice Boltzman Method
PDE	Partial Differnetial Equation
NSE	Navier-Stokes Equations
MRT	Multiple Relaxation Time
HHL	Harrow-Hassidim-Lloyd Algorithm
LCU	Linear Combination of Unitaries

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

Porous media are materials made of a solid matrix that contains a network of pores. These pores allow transport phenomena such as fluid flow, heat transfer, mass diffusion, and, in some cases, the propagation of electrical or acoustic signals. These processes often take place simultaneously and are influenced by the complex geometry, connectivity, and heterogeneity of the pore structure. As a result, pore characteristics directly affect transport efficiency (how effectively substances move), retention times (how long substances remain), and reaction rates (how quickly reactions occur) within the material. [3]

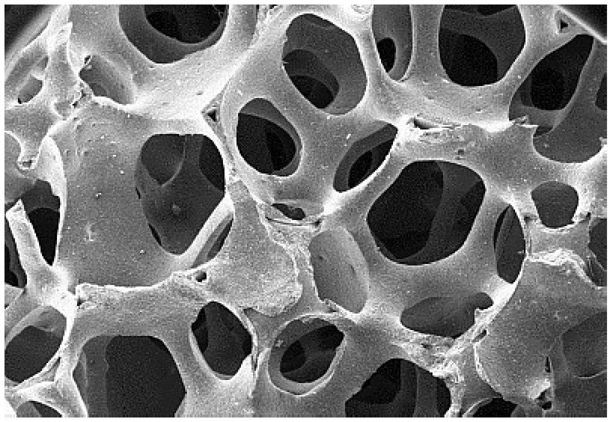


Figure 1.1 Open cell ceramic foam [1]

Porous materials appear in a variety of contexts: in hydrogeology, where soil and fractured rock regulate the movement of groundwater [4]; in biomedical engineering, where porous scaffolds support nutrient exchange and tissue growth [5]; and in chemical engineering, where open-cell ceramic foams serve as catalyst supports in multiphase reactors, enhancing gas-liquid interactions in systems like bubble columns and trickle bed reactors [6](Fig.1.1), among other applications. A prominent example in today's wave of clean energy innovation is found in polymer electrolyte membrane (PEM) fuel cells [7], where the gas diffusion layer must simultaneously manage the transport of reactant gases, heat, and liquid water across multiple scales. Effectively capturing the link between microscopic pore dynamics and macroscopic outcomes is key to both accurate simulations and the development of advanced technologies.

Given the critical role of transport phenomena in porous media, fluid flow is especially important because it directly governs many real-world processes. For example, it controls groundwater movement in aquifers, oil extraction in petroleum reservoirs, and blood circulation through biological tissues. While fluid flow in porous media is well-studied, accurately simulating it becomes increasingly challenging when pore sizes vary across multiple length scales.

Simulating such complex flow behavior requires advanced computational techniques. Computational Fluid Dynamics (CFD) is a core tool in fluid mechanics that uses numerical methods such as finite volume (FVM), finite element (FEM), or finite difference methods (FDM) to simulate and analyze fluid behavior. Despite its power and versatility, conventional CFD techniques face significant challenges when applied to porous media, especially with multiscale and heterogeneous pore networks. Accurately capturing flow within these complex geometries requires extremely fine computational grids and dense meshes to resolve the smallest pores, drastically increasing computational cost and runtime. Furthermore, ensuring numerical stability and achieving convergence across the wide range of length scales inherent to porous materials adds further complexity to these simulations.

In response to these challenges, the Lattice Boltzmann Method (LBM) has emerged as a powerful and faster alternative to conventional CFD approaches for simulating fluid dynamics, particularly in porous media [8]. Originating from lattice gas automata (LGA), LBM simulates fluid flow by modeling the collective motion of fluid particles on a discrete grid. Instead of directly solving the Navier–Stokes equations like traditional CFD methods, LBM is based on a simplified form of the Boltzmann equation from kinetic gas theory, where space, time, and particle velocities are discretized. This mesoscopic approach captures macroscopic fluid behavior through local microscopic particle collisions and streaming operations.

LBM features a simple, explicit algorithm, making it easy to implement and well-suited for parallel computing. Its local update rules allow for efficient execution on both CPU and GPU architectures, supporting large-scale simulations involving billions of lattice cells. It typically achieves second-order convergence in both space and time. Additionally, LBM employs structured Cartesian grids, which simplify meshing and allow fluid-solid interfaces to be represented using simple Boolean encoding. Moreover, LBM is not restricted to Navier–Stokes-based flow regimes and is well-suited for simulating complex transport phenomena, including multiphase flows, heat and mass transfer, and non-Newtonian fluids. These advantages have made LBM an increasingly popular tool across a broad range of scientific and engineering domains.

However, the need for speed, alongside increasing demands for accuracy and scale, continues to push the boundaries of classical computing. Even efficient methods like LBM become computationally expensive when resolving multiscale dynamics in large or complex domains. As a result, attention is shifting toward Quantum Computing (QC), a fundamentally new approach that may redefine what is computationally feasible (examples of QCs Fig.1.2).

(a) System One - IBMQ

(b) MonarQ - Anyon Systems

Figure 1.2 Examples of quantum computers

The growing interest in quantum approaches to computational physics is reflected in the increasing volume of research focused on quantum-based solutions to partial differential equations (PDEs). As illustrated in Fig.1.3[1], there has been a significant surge in activity over the past few years. This trend highlights a rising interest in the potential of quantum computing to address challenges traditionally tackled by classical methods, particularly in fields requiring high computational efficiency and scalability. Notably, the figure also shows that prior to 2020, research was primarily aimed at using LGA and LBM formulations to simulate fluid flow phenomena. While recent years have seen a broadening of focus toward a wider range of PDEs, interest in LGA and LBM, both rooted in PDEs, continues to grow in parallel, maintaining their sustained relevance within the quantum computing research landscape.

Unlike classical machines that process information using binary bits (0 or 1), quantum computers use quantum bits, or qubits, that leverage quantum mechanical principles to explore multiple computational paths simultaneously. This enables quantum computers to offer exponential speed-ups in certain tasks. While still in its early stages, quantum computing is being actively explored for a range of scientific problems, including fluid dynamics, where classical approaches face steep limitations, particularly when near real-time predictions are required.

In light of these challenges and opportunities, the objective of this thesis is **to develop an efficient quantum algorithm based on the Lattice Boltzmann Method for sim-**

[1]Web of Science prompt: "quantum computing" AND ("PDE" OR "LBM" OR "lattice boltzmann" OR "CFD" OR "heat transfer" OR "advection-diffusion" OR "mass transfer" OR "lattice gas")

ulating two-dimensional fluid flows through porous media in the Stokes regime. The purpose of this work is to explore a quantum-based formulation as a foundational step, laying the groundwork for future research and assessing the potential of quantum computing in fluid dynamics simulations as systems scale in complexity.

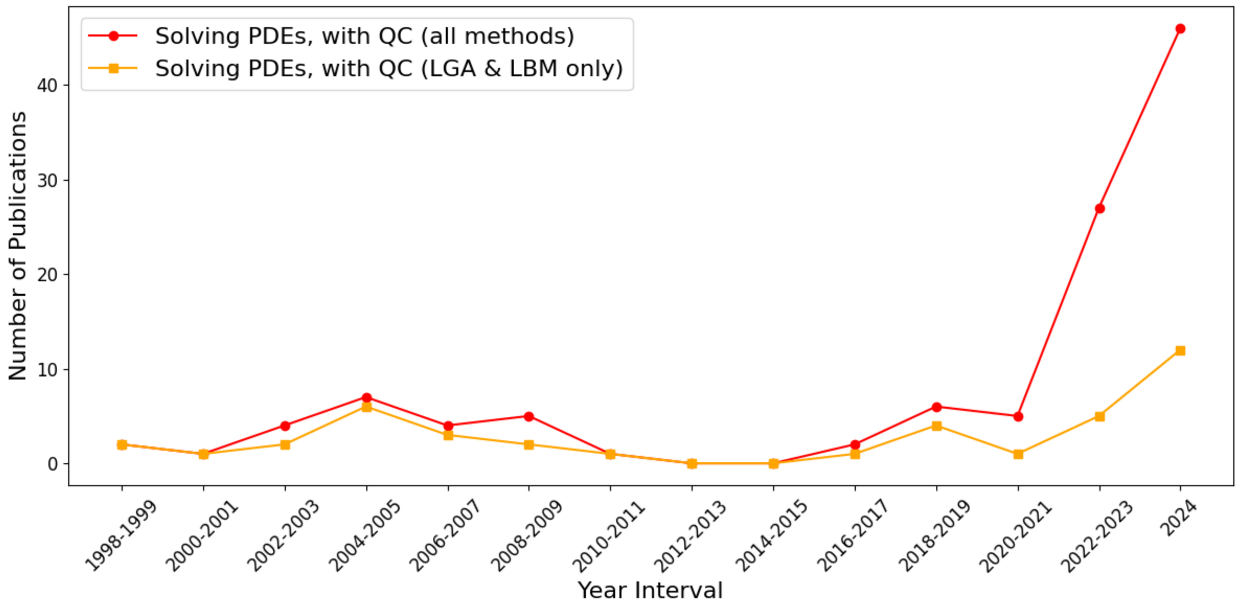


Figure 1.3 Publication trends in quantum computing for PDEs (*Web of Science*).
The red line shows all quantum PDE methods, while the orange line highlights lattice-based approaches within quantum PDEs (LGA and LBM). Both show rising interest, with LGA/LBM maintaining a steady share of overall research.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews the key concepts, recent advances, and research approaches that inform and inspire this thesis. It begins with the foundational principles of quantum computing, covering essential quantum phenomena, current hardware capabilities, and widely used simulators and libraries. The focus then shifts to LBM, summarizing its origins, mathematical formulation, and applications in fluid dynamics, along with its core strengths and limitations. The chapter concludes by exploring the intersection of these two domains through Quantum Lattice Boltzmann Methods (QLBM), highlighting efforts to harness quantum computing for enhanced fluid simulations. This review establishes the academic context and identifies the key research gaps addressed in this thesis.

2.1 Quantum Computing: Key Concepts and Background

In recent decades, the steady rise of classical computing power, famously predicted by Moore's Law, has enabled major technological progress. Yet as we near the limits of this trend, some problems have become so complex that even today's top supercomputers struggle to solve them. Quantum computing, a new paradigm rooted in quantum mechanics, offers the potential to fundamentally reshape how we process information. To understand this potential, we begin with its fundamental unit: **the qubit**.

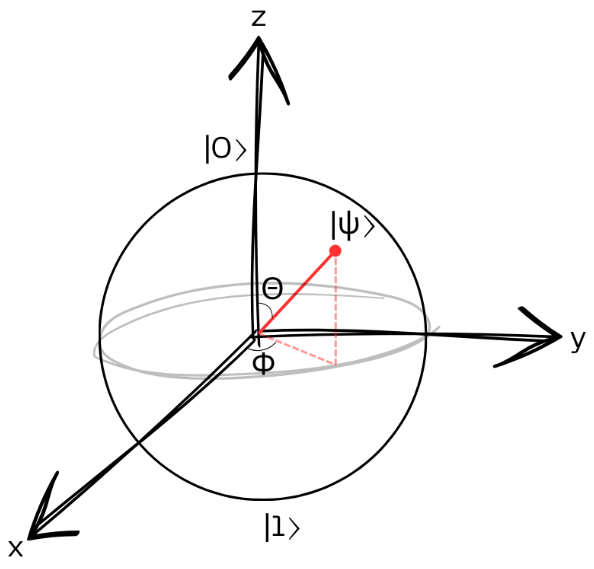


Figure 2.1 Geometric representation of a qubit state on the Bloch sphere

Unlike classical bits, which are either 0 or 1, qubits operate on fundamentally different principles. A qubit can exist in a superposition of the basis states $|0\rangle$ and $|1\rangle$ *(see Appendix A1 for more detailed description of basis states)*, enabling new forms of computation. This can be visualized using the Bloch sphere (Fig.2.1): if the bottom represents $|1\rangle$ and the top $|0\rangle$, then any point on the surface corresponds to a unique quantum state, a combination of both. Each state is defined by complex-valued amplitudes that determine the qubit's behavior and measurement probabilities. As the state evolves through quantum gates, it remains constrained by the principles of normalization and unitarity [9].

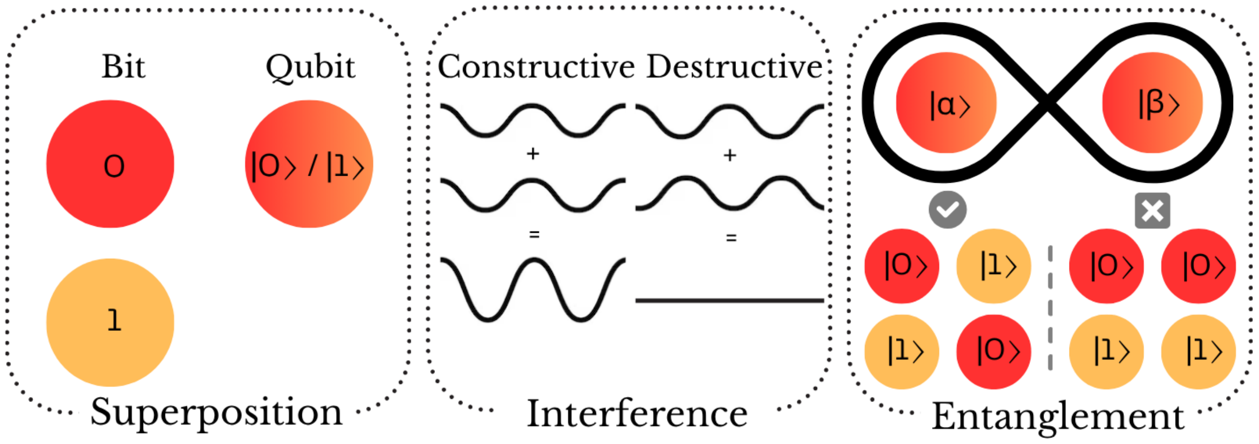


Figure 2.2 Quantum phenomena

What gives quantum computing its power are three uniquely quantum phenomena (Fig.2.2): superposition, entanglement, and interference.

1. **Superposition** allows qubits to exist in a combination of $|0\rangle$ and $|1\rangle$, enabling quantum systems to explore many possibilities simultaneously.

2. **Interference** enables quantum algorithms to reinforce correct computational paths while canceling out incorrect ones, guiding the system toward optimal results.

3. **Entanglement** links qubits so that the state of one instantly affects the other, regardless of distance, creating correlations that classical systems cannot replicate.

Together, these principles allow quantum algorithms to solve certain problems exponentially faster than classical methods.

2.1.1 Types of quantum computing paradigms and development tools

Having covered the fundamentals of quantum computation, we now turn to how these principles are implemented. Several quantum computing paradigms have emerged, each leveraging quantum mechanics in distinct ways [10]:

1. **Gate-based quantum computing** is the most widely used model, it relies on quantum circuits composed of gates that manipulate qubits to perform computations.

2. **Measurement-based quantum computing** performs computation using a highly entangled cluster states. Logic is performed through a sequence of targeted measurements, with outcomes determining the subsequent steps.

3. **Topological quantum computing** is a theoretical approach that encodes information in topologically protected states, offering resilience to errors through exotic particles like anyons.

4. **Quantum annealing / Adiabatic computing** is tailored for optimization, this model evolves a quantum system slowly towards its lowest-energy state, ideally representing the optimal solution.

Several open-source libraries have been developed to support gate-based quantum computing, including Qiskit (IBM), Cirq (Google), PennyLane (Xanadu), and Braket (AWS). These frameworks are primarily designed around quantum circuits, which are graph-based representations of operations on qubits and allow users to design, simulate, and execute quantum algorithms on simulators or real quantum hardware. While these libraries are tailored to the circuit model, they also offer some flexibility: for instance, measurement-based quantum computing can be emulated through circuit-level tools, though it is not their native paradigm. On the other hand, quantum annealing and topological quantum computing operate using fundamentally different principles that are not based on circuit-based models and generally require specialized platforms.

2.1.2 Quantum circuits: Building blocks of quantum algorithms

Quantum circuits provide a structured way to apply operations to qubits, much like classical circuits apply logic gates to bits. Each horizontal line or wire (Fig.2.3) represents a single qubit. A quantum register is a collection of such qubits, treated as a group for multi-qubit operations. Gates are applied sequentially from left to right, defining the circuit's depth, a key factor in both scalability and hardware performance.

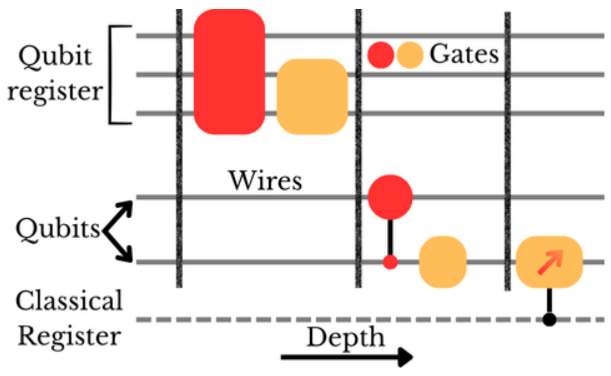


Figure 2.3 Quantum circuit building blocks

Before execution on actual quantum hardware, a circuit must be transpiled into native gates

compatible with the target device. This step may increase the circuit's depth due to hardware-specific constraints, such as limited qubit connectivity or native gate sets.
Quantum circuits also involve classical registers, which store the outcomes of quantum measurements. These registers hold definite binary values (0 or 1) after the qubits collapse from superposition upon measurement.
Qubits are typically labeled from top to bottom as `q[0]`, `q[1]`, ..., `q[n-1]`. In Dirac notation, however, the order is reversed, the leftmost qubit is the most significant, and the quantum state is written as $|q_n \dots q_0\rangle$, a convention followed throughout this thesis.

Encoding classical information into quantum circuits

Before computation begins, classical data must be encoded into quantum states. The choice of encoding directly affects circuit efficiency, depth, fidelity, and hardware requirements [9]. Common strategies include:

1. **Basis encoding** maps each bit string to a computational basis state (e.g., `101` $\rightarrow |101\rangle$). Simple and intuitive, requiring $\lceil \log_2(N) \rceil$ qubits for N values.
2. **Amplitude encoding** encodes data into quantum state amplitudes (e.g., the vector $[x_0, x_1, x_2, x_3]$ using 2 qubits becomes $|\psi\rangle = x_0 |00\rangle + x_1 |01\rangle + x_2 |10\rangle + x_3 |11\rangle$, where $|\psi\rangle$ is the amplitude encoded quantum state). Compact, but often requires complex state preparation.
3. **Angle encoding** maps features to gate rotation angles (e.g., $R_y(\theta)$). Common in variational algorithms due to its simplicity and flexibility.
4. **QSample encoding** encodes a probability distribution into the squared amplitudes of a quantum state, useful for probabilistic or sampling-based tasks.

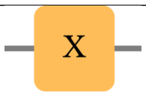
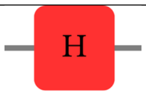
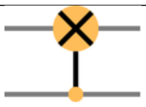
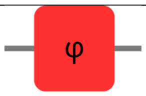
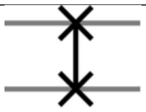
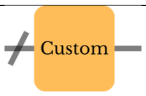
Quantum gates: Manipulating qubit states

After classical data is encoded into quantum states, quantum gates are applied to transform those states. These gates, analogous to logic gates in classical circuits, manipulate qubit states in controlled ways. Each quantum gate is represented by a unitary matrix U, satisfying [9]:

$$U^\dagger U = U U^\dagger = I \tag{2.1}$$

where $U^\dagger$ is the conjugate transpose of U, and I is the identity matrix. This ensures that quantum gates preserve normalization (total probability remains 1) and are reversible. Common quantum gates are shown in Table 2.1. Additional gates and their matrix forms are listed in *Appendix A1*.

Table 2.1 Common quantum gates and their functionalities

Gate	**Functionality**	**Circuit Representation**
Pauli-X (X)	Flips $\lvert 0\rangle \leftrightarrow \lvert 1\rangle$	
Hadamard (H)	Creates superpositions: $\lvert 0\rangle \to \frac{1}{\sqrt{2}}(\lvert 0\rangle + \lvert 1\rangle)$	
Controlled-NOT (CX)	Multi-qubit gate: flips target if control is $\lvert 1\rangle$; enables entanglement	
Phase (ϕ)	Applies a phase shift to $\lvert 1\rangle$: $P(\phi)\lvert 1\rangle = e^{i\phi}\lvert 1\rangle$	
SWAP	Exchanges the states of two qubits.	
Custom Gates	Any user-defined gate that is unitary and reversible	

Measurement and statevector representation

After operations are applied to a register of qubits (i.e., a group of qubits treated as a unit), the final step is typically measurement. Measurement collapses the quantum state into a classical outcome, destroying the superposition and phase information. Since this process is destructive, repeating the computation or extracting more information requires re-running the circuit from the beginning [9].

To analyze quantum behavior *before* measurement, researchers often use the statevector, a classical data structure that lists the complex amplitudes of each basis state in the system. For a 2-qubit register, for instance, the basis states are $\lvert 00\rangle$, $\lvert 01\rangle$, $\lvert 10\rangle$, and $\lvert 11\rangle$, and the quantum state is a linear combination of these. Although real quantum hardware cannot access the statevector during execution (as that would collapse the state), simulators can. This makes statevectors valuable for verifying circuit behavior and developing algorithms that rely on intermediate quantum information which is inherently not possible with actual qubits. While statevectors are powerful tools in simulation, translating these circuits to real hardware introduces practical challenges.

2.1.3 Practical limitations of quantum hardware circuits

Despite their potential, quantum hardware face several practical limitations:

- **Noise** occurs because qubits are extremely sensitive to environmental disturbances, which cause random errors.
- **Decoherence** happens when qubits lose coherence over time due to environmental noise that disrupts their quantum states, limiting computation duration.
- **Gate and measurement errors** arise from imperfections in operations and measurements, and these errors accumulate as circuits become deeper.
- **Circuit depth** affects reliability since longer sequences of gates increase the chance of errors, making shallow circuits more dependable.
- **Connectivity and crosstalk** refer to the limited interactions between qubits and the interference between nearby qubits, both of which add operational overhead.

These challenges necessitate circuit optimization, noise mitigation, and quantum error correction for scalable quantum computing.

2.1.4 Quantum circuit simulation

Current quantum hardware remains fragile and error-prone, making direct development and testing of algorithms on physical devices highly challenging. To overcome these limitations, researchers rely on quantum circuit simulators, classical tools that replicate the behavior of quantum systems in a controlled environment [11]. Simulators typically fall into three categories:

1. **Ideal simulators** assume perfect, noise-free operation to verify algorithm correctness.
2. **Noisy simulators** incorporate realistic errors such as decoherence and gate imperfections to assess robustness.
3. **Device-specific simulators** emulate specific hardware constraints, including native gate sets and qubit connectivity.

These tools are essential for refining algorithms, identifying performance bottlenecks, and optimizing designs before implementation on real quantum platforms.

2.1.5 Quantum algorithms and speedups

Quantum algorithms have demonstrated remarkable potential to outperform their classical counterparts in certain problem domains. Shor's algorithm [12], for example, can factor large integers exponentially faster than classical methods, posing a foundational challenge to modern cryptography. Grover's algorithm [13] offers a quadratic speedup for unstructured search problems, making it valuable for optimization and database search tasks. The Harrow-Hassidim-Lloyd (HHL) algorithm [14] allows exponential acceleration in solving specific linear systems, with broad applications in physics and machine learning.
Beyond these foundational examples, quantum speedups have been proposed for Monte Carlo simulations [15], combinatorial optimization [16], quantum chemistry and finance [17]. These developments highlight the growing relevance of quantum algorithms across diverse scientific and industrial domains.

2.1.6 Current state of quantum paradigm: NISQ era and outlook

Modern quantum hardware is situated in the Noisy Intermediate-Scale Quantum (NISQ) era, characterized by devices with tens to a few hundred qubits that are prone to noise, decoherence, and operational errors. Although not yet capable of fault-tolerant computation, NISQ systems provide a valuable testbed for developing quantum algorithms, error mitigation techniques, and hybrid classical-quantum approaches.
To harness the capabilities of these early-stage devices, quantum simulation has emerged as a practical and impactful application. Simulation strategies generally fall into three categories [18]:

1. **Type I: Classical simulation of quantum systems** – Numerical approaches on classical hardware to model quantum behavior (e.g., solving the Schrödinger equation).
2. **Type II: Quantum simulation of quantum systems** – Use of quantum hardware to simulate intrinsically quantum phenomena that are classically intractable.
3. **Type III: Quantum simulation of classical systems** – Application of quantum algorithms to simulate classical domains like fluid dynamics, heat transfer, or transport problems.

A notable effort under Type III is the quantum adaptation of LBM, a mesoscopic approach rooted in statistical mechanics. Originating from the Boltzmann equation, LBM discretizes particle distributions on a lattice to efficiently model complex fluid flows. Its inherent par-

allelism and local update rules make it a natural candidate for quantum translation: quantum circuits could harness superposition and entanglement to accelerate LBM's collision and streaming operations, exploiting quantum parallelism beyond classical capabilities. This synergy could enable high-resolution simulations that challenge traditional computational limits, particularly for problems where classical LBM faces memory or scalability bottlenecks.
While these approaches are still in the research phase, Type III simulations such as quantum LBM offer near-term opportunities to explore practical applications, forming a bridge between today's NISQ-era hardware and the long-term vision of scalable, fault-tolerant quantum computing.

2.2 The Lattice Boltzmann Method: Fundamentals and Applications

2.2.1 Classical CFD foundations

Classical CFD relies on the continuum description of fluids, governed by the Navier–Stokes equations (NSE), which represent conservation of momentum and mass. These equations are typically solved using numerical methods such as FDM, FVM, or FEM, as shown on the left branch of the schematic in Fig.2.4 [19].
In two dimensions, for dimensionless velocity components $u(x, y, t)$ and $v(x, y, t)$, the dimensionless NSE are:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{2.2}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{2.3}$$

accompanied by the continuity equation:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{2.4}$$

here, Re is the Reynolds number is given by :

$$Re = \frac{\rho U L}{\mu} = \frac{UL}{\nu} \tag{2.5}$$

where $\nu = \mu/\rho$ is the kinematic viscosity[1]. Although these methods are known for their accuracy, they are computationally intensive, particularly due to the complex coupling between velocity and pressure fields in incompressible flows. The nonlinear nature of the equations and the requirement for iterative solvers make simulations of turbulent or multiphase flows challenging. These limitations have led researchers to consider discrete, particle-based models such as LGA, which eventually evolved into LBM.

In particular flow regimes, such as those characterized by low Reynolds numbers, the inertial forces (convective term) of the Navier–Stokes equations become negligible compared to viscous forces. This leads to the simplification known as time-dependent *Stokes equations*, which lead to creeping flow, which is often encountered in porous media, microfluidics, and slow-moving suspensions. In the Stokes flow limit, the dimensionless Navier–Stokes equations reduce to:

$$\frac{\partial u}{\partial t} = -\frac{\partial p}{\partial x} + \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{2.6}$$

$$\frac{\partial v}{\partial t} = -\frac{\partial p}{\partial y} + \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{2.7}$$

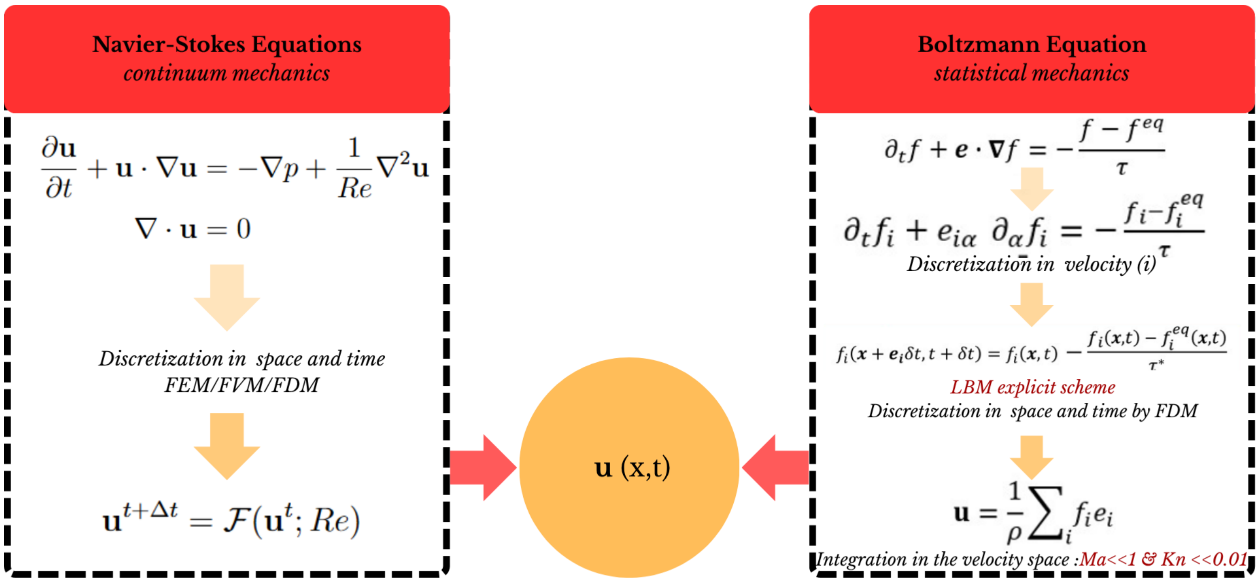

Navier-Stokes Equations
continuum mechanics
$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u} = -\nabla p + \frac{1}{Re}\nabla^2\mathbf{u}$
$\nabla\cdot\mathbf{u} = 0$
Discretization in space and time
FEM/FVM/FDM
$\mathbf{u}^{t+\Delta t} = \mathcal{F}(\mathbf{u}^t; Re)$
u (x,t)
Boltzmann Equation
statistical mechanics
$\partial_t f + \boldsymbol{e}\cdot\nabla f = -\frac{f - f^{eq}}{\tau}$
$\partial_t f_i + e_{i\alpha}\,\partial_\alpha f_i = -\frac{f_i - f_i^{eq}}{\tau}$
Discretization in velocity (i)
$f_i(\boldsymbol{x} + \boldsymbol{e}_i\delta t, t + \delta t) = f_i(\boldsymbol{x}, t) - \frac{f_i(\boldsymbol{x},t) - f_i^{eq}(\boldsymbol{x},t)}{\tau^*}$
LBM explicit scheme
Discretization in space and time by FDM
$\mathbf{u} = \frac{1}{\rho}\sum_i f_i e_i$
Integration in the velocity space : Ma<<1 & Kn <<0.01

Figure 2.4 Comparison between classical CFD and LBM. Here, $\mathbf{u} = (u, v)$ is the 2D velocity field with components in the x- and y-directions

[1]In porous media, the Reynolds number is defined as $Re_p = \frac{\rho U D_p}{\mu(1-\varepsilon)}$ where ε is the porosity and D_p is the characteristic particle or pore diameter. Flow is typically considered creeping when $Re_p < 8$ [20].

with the continuity equation unchanged:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0. \tag{2.8}$$

The absence of convective terms simplifies the mathematical treatment and highlights the dominance of viscous dissipation in such flows, making Stokes flow a useful approximation in the study of slow, viscous-dominated regimes.

2.2.2 Lattice gas automata

Lattice Gas Automata (LGA), developed in the 1980s, is the precursor to the Lattice Boltzmann Method (LBM). LGA simulates fluid flow at a microscopic level by modeling particles that move and collide on a discrete lattice. Each particle's presence or absence at a lattice site and direction is represented by Boolean variables. Particle collisions follow simple, rule-based interactions that locally conserve mass and momentum, allowing macroscopic fluid behavior to emerge from these microscopic dynamics.

However, to recover the correct continuum-scale behavior described by the Navier–Stokes equations, LGA simulations must be repeated a large number of times, with ensemble averaging applied to extract statistically meaningful results. This requirement arises due to the inherent stochasticity of LGA's collision rules, which introduce statistical noise and cause variability between individual runs. Additionally, representing particles as discrete binary states leads to high variance, further complicating the modeling of continuum fluids without extensive post-processing.

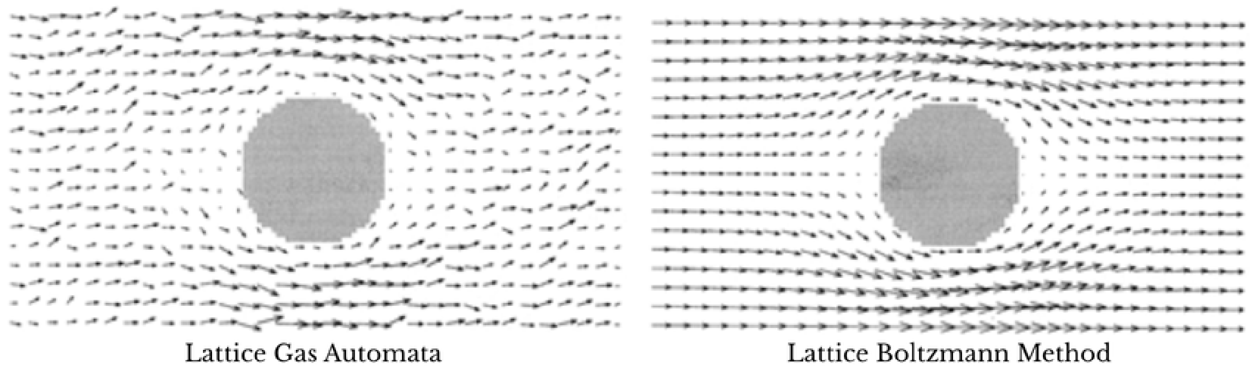


Figure 2.5 Comparison of flow around a cylinder using LGA and LBM [2]

2.2.3 The Lattice Boltzmann perspective

To overcome these limitations, the Lattice Boltzmann Method was developed. LBM replaces Boolean particles with continuous distribution functions representing expected particle populations per lattice direction. This deterministic, mesoscopic formulation reduces noise and improves numerical stability while retaining the local, particle-based structure of LGA. As a result, LBM enables efficient and accurate simulation of complex fluid flows. A comparative schematic of LGA and LBM velocity vector field, is shown in Fig.2.5. Both LGA and LBM operate on a regular discrete lattice, where particles (or probability distribution functions, called populations for short) propagate and interact at discrete spatial positions and time steps. The lattice defines the allowable directions in which populations can move between nodes and is critical for accurately capturing the fluid dynamics. A common notation used to describe the lattice structure is **D**n**Q**m, where n denotes the spatial dimension (e.g., 2 for two-dimensional, 3 for three-dimensional), and m represents the number of discrete propagation/streaming velocity directions per node.

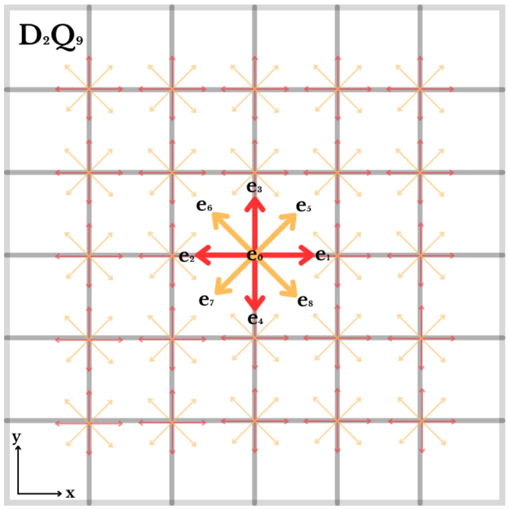


Figure 2.6 D2Q9 lattice structure

For example, the **D2Q9** lattice (Fig.2.6) widely used in 2D simulations, comprises nine velocity directions: one rest state, four axial directions (left, right, up, down), and four diagonal directions. It is important to note that this labeling scheme is not universal; the specific numbering and velocity vectors may vary across implementations or software packages, though the lattice connectivity remains conceptually consistent.

LBM offers an alternative approach rooted in statistical mechanics. It models fluid behavior at a mesoscopic scale by evolving discrete particle distribution functions over a lattice. Rather than directly solving the NSE, LBM simulates the evolution of particle populations moving in specific directions, with macroscopic fluid properties recovered via statistical averaging. At the heart of LBM lies the Boltzmann transport equation [8]:

$$\frac{\partial f}{\partial t} + \mathbf{e} \cdot \nabla f = -\frac{f - f^{\text{eq}}}{\tau} \tag{2.9}$$

where $f(\mathbf{x}, \mathbf{e}, t)$ is the single-particle distribution function representing the probability of finding a particle at position $\mathbf{x}$ with velocity $\mathbf{e}$ at time t, f^{eq} is the local equilibrium distribution, and τ is the relaxation time.

To derive the LBM formulation, the propagation velocity space is discretized into a finite set of directions $\mathbf{e}_\alpha$, and Equation 2.9 is discretized in velocity space using explicit finite-difference approximations. This leads to the LBM update rule:

$$f_\alpha(\mathbf{r} + \mathbf{e}_\alpha \Delta t, t + \Delta t) = (1 - \epsilon) f_\alpha(\mathbf{r}, t) + \epsilon f_\alpha^{\mathrm{eq}} + \Delta t w_\alpha S, \tag{2.10}$$

where, f_α denotes the particle population distribution function in the direction α, while $\mathbf{e}_\alpha$ represents the corresponding discrete velocity direction. w_α is the weight associated with each discrete velocity direction $\mathbf{e}_\alpha$ in the lattice, S is the source term, which can represent the effects of body forces(F_α), such as gravity or pressure gradients, as well as other external influences acting on the fluid and the term f_α^{eq} indicates the equilibrium distribution function. The variable Δt refers to the time step, and the dimensionless relaxation coefficient is given by $\epsilon = \frac{\Delta t}{\tau}$.

This formulation supports three processes: (i) **Streaming**: particles move to neighboring sites, (ii) **Collision**: distributions relax toward equilibrium, (iii) **Source**: inclusion of external influences. As illustrated on the right branch of the schematic in Fig.2.4, the continuous Boltzmann equation is discretized and simplified, yielding a numerically efficient and local evolution rule well suited to parallel computing.

Propagation and collision in the lattice Boltzmann method

The collision step in LBM involves a local relaxation process in which each distribution function f_α relaxes toward an equilibrium distribution f_α^{eq}, which depends on macroscopic quantities such as the local density ρ and velocity $\mathbf{u}$. Under the widely used Bhatnagar–Gross–Krook (BGK) approximation, this relaxation is described by:

$$f_\alpha^*(\mathbf{r}, t) = f_\alpha(\mathbf{r}, t) - \frac{\Delta t}{\tau} \left(f_\alpha(\mathbf{r}, t) - f_\alpha^{\mathrm{eq}}(\mathbf{r}, t) \right) \tag{2.11}$$

where $f_\alpha^*(\mathbf{r}, t)$ denotes the post-collision distribution function, and $f_\alpha(\mathbf{r}, t)$ denotes the pre-collision particle distribution function in the direction α at position $\mathbf{r}$ and time t, and $f_\alpha^{\mathrm{eq}}(\mathbf{r}, t)$ is the corresponding local equilibrium distribution function. The term τ represents the single relaxation time that governs the rate at which the system relaxes toward equilibrium, and Δt is the time step. Alternatively, introducing the relaxation parameter $\epsilon = \frac{\Delta t}{\tau}$, this can be rewritten as:

$$f_\alpha^*(\mathbf{r}, t) = (1 - \epsilon) f_\alpha(\mathbf{r}, t) + \epsilon f_\alpha^{\mathrm{eq}}(\mathbf{r}, t) \tag{2.12}$$

To simplify the implementation, we set $\tau = 1$ and $\Delta t = 1$, which gives $\varepsilon = 1$. Substituting into the equation yields:

$$f_\alpha^*(\mathbf{r}, t) = f_\alpha^{\mathrm{eq}}(\mathbf{r}, t),$$

indicating that the post-collision distribution is simply the equilibrium distribution. Here, f_α^* represents the post-collision distribution at site $\mathbf{r}$. This collision step is fully local and involves only information at the same lattice node, with no inter-node communication required. This locality is a major advantage of LBM, enabling highly parallel and computationally efficient implementations. Moreover, this property helps alleviate LBM's main drawback of relatively high memory requirements, by enabling streamlined memory access patterns and reduced communication overhead in distributed systems. This simplification is often used in theoretical studies or idealized simulations to isolate specific behaviors or reduce computational complexity [21]. This BGK model simplifies the collision process by encapsulating the interaction dynamics into a single relaxation time τ, allowing for an efficient implementation. While it is the most commonly employed model, more sophisticated alternatives, such as the Multi-Relaxation-Time (MRT), can offer improved stability and accuracy in challenging flow regimes.

During the streaming step, particle distribution functions f_α are advected along their discrete velocity directions $\mathbf{e}_\alpha$ to adjacent lattice nodes (Fig.2.6). This step models the streaming of particle distributions between lattice nodes and is expressed as:

$$f_\alpha(\mathbf{r} + \mathbf{e}_\alpha \Delta t, t + \Delta t) \leftarrow f_\alpha^*(\mathbf{r}, t) \tag{2.13}$$

The streaming operation transfers particle distribution functions from one lattice site to its immediate neighbors along discrete velocity directions. As it involves only first-neighbor connectivity, it is inherently quasi-local.

Connection to viscosity and physical interpretation

The relaxation time τ governs the rate at which the distribution functions return to equilibrium and also determines the kinematic viscosity ν of the simulated fluid. In the commonly used D2Q9 lattice configuration, the relation between τ and ν is given by:

$$\nu = c_s^2 \left(\tau - \frac{\Delta t}{2} \right) \tag{2.14}$$

where $c_s = \frac{1}{\sqrt{3}} \frac{\Delta x}{\Delta t}$ is the lattice speed of sound, where Δx and Δt are the lattice spacing and time step, respectively. This relationship provides a critical link between the mesoscopic formulation of LBM and the macroscopic properties of the fluid. Physically, a smaller τ implies a faster return to equilibrium, modeling a fluid with greater resistance to deformation. Conversely, a larger τ allows the fluid to retain non-equilibrium states longer, representing lower viscosity. Thus, Eq.2.14 form an essential bridge between the numerical behavior of the LBM algorithm and the physical characteristics of the modeled fluid.

The equilibrium distribution function

In LBM, the equilibrium distribution function f_α^{eq} represents the target state toward which the particle distributions relax during collisions. It is defined as:

$$f_\alpha^{\mathrm{eq}} = w_\alpha \, \phi \left[1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{u}}{c_s^2} + \frac{(\mathbf{e}_\alpha \cdot \mathbf{u})^2}{2c_s^4} - \frac{\mathbf{u} \cdot \mathbf{u}}{2c_s^2} \right] \tag{2.15}$$

For creeping flows (low Reynolds number) encountered in most porous media problems , inertial effects are negligible. Under the assumption $Re \ll 1$, nonlinear terms can be neglected and f_α^{eq}:

$$f_\alpha^{\mathrm{eq}} \approx w_\alpha \, \phi \left[1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{u}}{c_s^2} \right] \tag{2.16}$$

Incorporation of body forces in the lattice Boltzmann method

To simulate external forces like gravity or pressure gradients, LBM scheme can include a body force as a source term in the evolution equation. Eq.2.17 is adopted from [22] for the incorporation of body force in the system. There exist multiple expressions that have been implemented over the years to add a body force with various degrees of accuracy. The simplest and oldest approach is the LGA scheme where:

$$F_\alpha = w_\alpha \, \Delta t \left(\frac{\mathbf{F} \cdot \mathbf{e}_\alpha}{c_s^2} \right) \tag{2.17}$$

where F_α is the force term added to the distribution function in direction α, $\mathbf{F}$ is the external body force vector. This body force formulation is commonly used in LGA schemes and is chosen because it introduces error terms in the mass and momentum balances that vanish when the body force F_α is constant in space and time [22]. By considering F_α as spatially and temporally uniform, this approach avoids additional discretization errors in the macroscopic equations, effectively making the scheme accurate under these conditions. While other forcing schemes may offer improved accuracy for spatially or temporally varying forces, this formulation is particularly well-suited for creeping flows with constant body forces, where viscous effects dominate and inertial effects are negligible.

2.2.4 Boundary conditions in LBM

The implementation of appropriate boundary conditions is essential to ensure the physical accuracy and stability of LBM simulations. These conditions are usually applied at the population level rather than directly on the macroscopic quantities. Common types of boundary conditions include (Fig.2.7):

- **Bounce-back boundary conditions** enforce the no-slip condition at solid walls by reflecting particle distributions. In the full-way scheme, reflections occur at the fluid node adjacent to the wall, resulting in first-order accuracy. The halfway bounce-back scheme improves spatial accuracy to second order by placing the reflection point midway between the fluid and solid nodes.
- **Periodic boundaries** apply to domains with repeating spatial patterns, allowing fluid populations that exit one boundary to re-enter seamlessly from the opposite side.
- **Inflow and outflow boundaries** define the behavior of fluid at the domain's entrances and exits, indirectly prescribing velocity or pressure conditions to control how fluid enters or leaves the simulation domain.
- **Moving boundaries** simulate walls or interfaces in motion by updating the distribution functions at boundary nodes to reflect the relative velocity between the fluid and the boundary. In LBM, this is typically achieved by modifying the bounce-back scheme with a velocity correction, ensuring accurate enforcement of the no-slip condition in the moving frame.

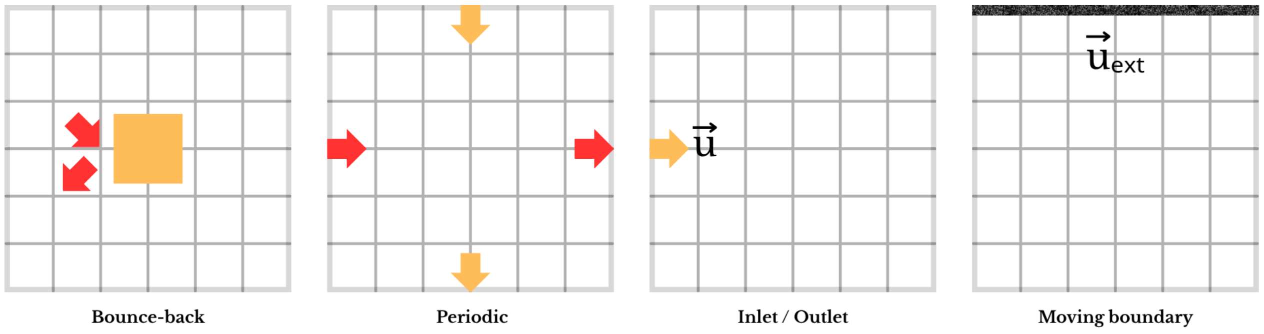


Figure 2.7 Common types of boundary conditions

The following are some of the most commonly used boundary conditions in LBM, each tailored to enforce specific physical constraints.

Halfway bounce-back boundary condition: The halfway bounce-back method improves upon the classical bounce-back by placing the solid boundary midway between fluid and solid lattice nodes. During the streaming step, any particle distribution streaming into a solid node is reflected back to its origin node along the opposite direction. Mathematically, for a distribution f_α streaming towards a solid boundary, the reflected distribution f_α (where $\bar{\alpha}$ indicates the direction opposite to α) is set as:

$$f_\alpha(\mathbf{r}, t + \Delta t) = f_{\bar{\alpha}}(\mathbf{r}, t), \tag{2.18}$$

effectively enforcing a no-slip velocity condition at the wall. This method balances computational efficiency with accuracy and is particularly suitable for complex geometries where solid boundaries do not align neatly with lattice nodes, however, it shows second-order accuracy only for straight walls.

Periodic boundary condition: Periodic boundaries assume the flow field repeats spatially, creating a seamless looping domain. Fluid leaving the domain through one boundary immediately re-enters from the opposite boundary with identical distribution functions. This is implemented by copying distribution functions at one edge of the computational domain to the opposite edge:

$$f_\alpha(x = 0, y, t) = f_\alpha(x = L_x, y, t), \tag{2.19}$$

and similarly for other directions as needed, where L_x is the domain length in the x-direction and α is the incoming direction to the domain.
Periodic boundary conditions are commonly used in simulations of infinite or repeating porous structures, channel flows with symmetry, and turbulence studies. They significantly reduce computational cost by minimizing artificial boundary effects.

Recovering Navier–Stokes behavior

LBM can be shown to recover the Navier–Stokes equations in the limit of low Mach number ($Ma \ll 1$) and low Knudsen number ($Kn \ll 0.01$) through the Chapman–Enskog multiscale expansion [8]. The macroscopic quantities, such as density is obtained as the zeroth moment of the distribution (2.20), and velocity is obtained as the first moment of the distribution function (2.21).

$$\rho = \sum_{\alpha} f_{\alpha} \tag{2.20}$$

$$\rho \mathbf{u} = \sum_{\alpha} f_{\alpha} \mathbf{e}_{\alpha} \tag{2.21}$$

By carefully selecting the equilibrium distribution function and lattice structure (e.g., D2Q9), LBM captures the essential hydrodynamic behavior consistent with the Navier–Stokes equations without explicitly solving the coupled pressure-velocity fields (Poisson equation) typical of classical computational fluid dynamics (CFD).

2.3 Quantum Approaches to Classical Fluid Dynamics

Having outlined the quantum advantages in recent years (Sec.2.1.5), particularly for Type III problems (Sec.2.1.6), a core challenge emerges: classical fluid dynamics problems are inherently nonlinear and non-unitary, whereas quantum computing requires operations to be unitary and reversible. This fundamental mismatch between dissipative classical systems and reversible quantum evolution presents a significant theoretical barrier to direct simulation.

2.3.1 Solving PDEs with quantum computing

Efforts to bridge this gap date back to 1926, when the Madelung transformation reformulated the Schrodinger equation in hydrodynamic terms [23], revealing a link to fluid dynamics. After early developments, the field stagnated for nearly a decade until the introduction of the Harrow–Hassidim–Lloyd (HHL) algorithm [14], a key quantum method for solving linear systems. Though initially theoretical, HHL has recently gained practical relevance after 2020 when adapted for fluid dynamics problems governed by PDEs. This revival aligned with IBM's launch of the IBM Q System One, sparking renewed interest in quantum computing for scientific applications.

Several NSE problems were approached using HHL. For instance, Lapworth integrated HHL with the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) method [24]. Other approaches incorporated Quantum Singular Value Transformation (QSVT) algorithms to handle sparse matrix structures which are common in fluid dynamics simulations. While QSVT enables efficient matrix inversion, convergence rates in hybrid solvers are sensitive to precision loss in eigenvalue estimation, revealing vulnerabilities in practical implementations [25]. State preparation techniques were also improved using sparsity to generate circuits in polynomial time; however, these methods still scale poorly with general states and require careful encoding to remain efficient [26]. While HHL demonstrated quantum advantage, its practical utility was hindered by the challenges of state preparation and readout. With n qubits encoding 2^n amplitudes, extraction of information becomes exponentially difficult. Although several encoding techniques have been developed, most suffer from exponential complexity. Despite this, the new schemes are optimized for sparse matrices, offering improved circuit depth and operational efficiency [26].

In 2020, Gaitan developed a quantum algorithm for solving the compressible Navier–Stokes equations, demonstrating quadratic speedup and accurate shockwave resolution, though the method was restricted to steady-state inviscid flows [27]. In 2021, Oz tackled nonlinear PDEs such as Burgers' equation using quantum circuits, showing classical-level accuracy but remaining limited by scalability and hardware constraints for 3D turbulent flows [28]. That same year, Kyriienko proposed Quantum Neural Networks combined with Carleman linearization[2] to simulate nonlinear dynamics, extending quantum benefits beyond dissipative regimes, although broader generalization across PDEs remains open [29]. In 2022, hybrid quantum-classical reservoir computing models emerged, demonstrating that moderately entangled qubits can emulate chaotic Lorenz-type dynamics with performance comparable to large classical reservoirs; however, the approach suffers under decoherence and shallow entanglement [30]. By 2023, new studies explored the Liouville equation, describing evolution in phase space as a quantum-suitable formulation. In 2024, Schalkers and Moller proposed a scalable quantum transport method, with reduced CNOT gate complexity and efficient velocity encoding, though constrained to structured grids and theoretical implementations [31]. Concurrently, efforts to parallelize quantum fluid simulations gained momentum. Broadbent and Kashefi proposed distributed quantum computing strategies that reduce circuit depth from polynomial to logarithmic scaling, enabling subprocess offloading across quantum circuits, though still awaiting hardware maturity [32]. Matteo and Mosca introduced Synthetiq, a simulated annealing-based circuit synthesis tool achieving significant gate depth reductions across arbitrary gate sets, yet its real-time application in fluid solvers remains to be demonstrated [33].

2.3.2 Solving LGA and LBM with quantum computing

In parallel, the Madelung transformation [23] connection was also later advanced by Succi and Benzi [34] in 1993, who showed how LBM could model pseudo-relativistic quantum systems, laying a foundation for quantum analogs of classical fluid behavior. From 1998 to 2002,

[2]Carleman linearization embeds nonlinear dynamics into a higher-dimensional linear systems.

there were significant strides using Lattice Gas Automata (LGA) [35] [36] [37] [38] [39], successfully implementing the collision step on quantum circuits. However, the streaming step remained classical, and the quantum circuits required a linearly growing number of qubits, which limited scalability.

In 2020, Steijl [40] implemented a collisionless Boltzmann equation model for highly rarefied gas flows, focusing solely on free-molecular flow with specular reflection. However, complex particle interactions were omitted. In 2024, Schalkers and Moller [41] advanced collisionless Boltzmann modeling by implementing accurate reflection behavior under all boundary conditions and successfully identifying grid points within obstacle walls of the domain. Their approach achieved a polynomial speedup, notably through a novel velocity vector encoding that enabled velocity flipping using a single-qubit operation rather than multi-qubit transformations. The work also proposed a new streaming approach that significantly reduced the CNOT gate count compared to prior quantum streaming techniques. Additionally, Schalkers and Moller presented an object wall encoding strategy whose complexity became independent of wall size, and a velocity encoding that allowed for linear-speed reflections irrespective of the number of velocities. This fail-safe, resource-efficient approach leads to physically correct flow behavior across a variety of configurations, though it remains constrained to Cartesian-aligned domains.

A key strategy for adapting LBM to quantum computing involves linearizing its nonlinear structure. Methods like Carleman linearization and Koopman–von Neumann embedding have been explored. Carleman linearization has proven more effective and quantum-compatible than Koopman-based techniques. Succi et al. applied Carleman linearization to the collision step of LBM, enabling unitary evolution of both collision and streaming components, and extending this to simulations of Kolmogorov-like flow characterized by chaotic vortical structures in turbulence. Here, a single quantum time-step was shown to mitigate the circuit depth issue [42], [43], [44]. Carleman linearization holds particular promise for quantum simulation (Type II problems that use quantum hardware to simulate classically intractable and intrinsically quantum phenomena) as it facilitates unitary transformation while preserving nonlinear system behavior.

Parallel developments were made by Budinski, who introduced a QLBM scheme for the advection-diffusion problem [45]. Several custom gates developed in this work, such as those for collision and propagation, have proven essential for subsequent research efforts. Since the resultant matrix was not inherently Hermitian, only time discretization was applied. Budinski later expanded this to a streamfunction–vorticity formulation of the NSE, eliminating the pressure term and reducing the problem to an advection-diffusion equation and a Poisson equation [46]. This formulation is developed by taking the curl of the momentum equations, and the pressure term is eliminated, yielding:

$$\nabla^2\psi = -\omega, \tag{2.22}$$

$$\frac{\partial\omega}{\partial t} + u\frac{\partial\omega}{\partial x} + v\frac{\partial\omega}{\partial y} = \frac{1}{Re}\left(\frac{\partial^2\omega}{\partial x^2} + \frac{\partial^2\omega}{\partial y^2}\right), \tag{2.23}$$

where ψ is the streamfunction, ω the scalar vorticity, and the velocity components relate to the streamfunction as:

$$u = \frac{\partial\psi}{\partial y}, \quad v = -\frac{\partial\psi}{\partial x}. \tag{2.24}$$

In this formulation, the LBM scheme remains fundamentally the same, with collision, streaming, and macroscopic retrieval stages adapted to evolve the two distribution functions representing the vorticity f_ω and streamfunction g_ψ fields. A D2Q5 lattice is employed, providing a minimal yet sufficient set of discrete velocities to capture the required dynamics. The equilibrium distributions are defined similarly, with the source term accounting for vorticity in the Poisson equation. Boundary conditions are straightforward to impose. For instance, on a domain boundary $\partial\Omega$, the streamfunction ψ is typically held constant, often set to zero. The corresponding equilibrium distributions satisfy:

$$\psi = \sum_{\alpha=0}^{N_l-1} g_\alpha^{eq} = g_0 + g_1 + g_2 + g_3 + g_4, \tag{2.25}$$

where ψ is the macroscopic streamfunction, g_α^{eq} is the equilibrium distribution function in direction α, and N_l denotes the number of discrete velocity directions in the D2Q5 lattice.

This leads to the boundary condition for the distributions:

$$g_{\partial\Omega} = -\sum_{\alpha|\alpha\neq\partial\Omega} g_\alpha, \tag{2.26}$$

where $g_{\partial\Omega}$ is the unknown incoming population at the boundary $\partial\Omega$, and the summation is taken over all known outgoing directions α not pointing into the wall.

The vorticity at the boundary is approximated by:

$$\omega_{i,N} = -2\left(\frac{\psi}{\Delta y^2} + \frac{U}{\Delta y}\right), \tag{2.27}$$

where $\omega_{i,N}$ is the vorticity at the top boundary node located at horizontal index i, Δy is the vertical lattice spacing, and U is the lid velocity in the lid-driven cavity case.

The wall equilibrium distribution in the wall direction is then given by:

$$g(x,y)_{\partial\Omega} = -\sum_{\alpha \neq \partial\Omega} g_\alpha - 2\left(-\frac{\psi}{\Delta y^2} + \frac{U}{\Delta y}\right), \tag{2.28}$$

where $g(x,y)_{\partial\Omega}$ denotes the population at a wall node located at position (x,y), and the second term accounts for the vorticity correction induced by the moving lid.

This was implemented within a single quantum circuit using the Linear Combination of Unitaries (LCU) method, a technique for expressing non-unitary operations as weighted sums of unitary operations. Additionally, this approach requires only the zeroth moment of the distribution functions to recover velocity, simplifying computations compared to classical LBM which requires first moments. However, this method is limited to two-dimensional flows because in three dimensions, vorticity becomes a vector quantity, and the scalar streamfunction no longer suffices.

More recently, Budinski has continued refining his approach through alternative formulations of lattice-based fluid solvers. In one work, he proposed an adaptive integer lattice gas algorithm (ALGA) capable of recovering the LBM equilibrium in the low-velocity regime, and further mapped it to a quantum algorithm using a linear collision operator [47], similar to his previous work. This highlights a possible direction for simplifying quantum implementations. However, his follow-up study introducing a fluctuation-free Float Lattice Gas Automata (FLGA) model [48, 49] revealed limitations: while the model simplifies computation and performs well in classical tests, the quantum version remains restricted to a single timestep due to the need for reinitialization after measurement. These developments indicate ongoing exploration, but further analysis is needed to assess whether LGA- or LBM-based methods provide a more feasible path for quantum computational fluid dynamics.

Building on these efforts, Lee et al., inspired by Budinski's work, implemented a parallelized version of the streamfunction–vorticity formulation with two populations in LBM, distributing tasks across multiple circuits to solve the NSE more efficiently [50]. This was done with a marked reduction in CNOT gates compared to existing QLBM circuits. This strategy has the additional benefit of the circuits being able to run concurrently, further halving the overall gate depth.

2.4 Identified gaps in literature

Despite notable progress, several critical scientific gaps remain in the potential application of QLBM for simulating fluid flow in porous media in 2D, with the possibility of extension to 3D:

- **Absence of velocity–pressure formulations:** Most quantum fluid models rely on stream-function–vorticity formulations, which are limited to 2-D.
- **Limited implementation of boundary conditions:** Essential boundary conditions such as bounce-back (for solid walls) and periodic boundaries (for infinite or repeating domains) are rarely incorporated, especially in velocity–pressure-based models, severely restricting the applicability of quantum algorithms to realistic fluid simulations.
- **No body-force implementations:** Body forces (e.g. gravity or pressure gradients) are critical for driving porous-media and channel flows, yet they have not been integrated into existing quantum fluid-dynamics models.
- **Lattice stencil used is limited to D2Q5:** Current quantum-LBM studies primarily restrict particle propagation to orthogonal directions, omitting diagonal links present in more complete lattices such as D2Q9 or D3Q19. Consequently, in simplified lattices like D2Q5 commonly used in prior work, the stress tensor is not fully recovered, leading to incomplete momentum exchange and reduced isotropy. To bypass this, most existing quantum-LBM efforts adopt the streamfunction–vorticity formulation, which eliminates the need for explicit stress tensor recovery and is well-suited for 2D incompressible flows, restricting scalability.
- **Limited work on 3-D flows:** The majority of existing quantum algorithms are confined to 2-D; extensions to 3-D are sparse because representing vector fields and complex geometries on quantum hardware is considerably more challenging.
- **Scalability issues with encoding and read-out:** Efficient state preparation and measurement remain significant bottlenecks, as most encoding schemes scale poorly and therefore diminish any potential quantum advantage.

CHAPTER 3 OBJECTIVE

This chapter presents the objectives of the research, beginning with the general objective that frames the overall aim. It then outlines specific objectives that define the key steps taken to address the research problem. The chapter concludes with a brief overview of the thesis structure.

3.1 General Objective

As mentioned in Chapter 1, the general objective of this thesis is **to develop an efficient quantum algorithm based on the Lattice Boltzmann Method for simulating two-dimensional fluid flows through porous media in the Stokes regime, that can be readily extended to 3D.**

3.2 Specific Objectives

The specific objectives of this thesis are as follows:

- **Develop a velocity–pressure formulation of QLBM:** transition from traditional streamfunction–vorticity schemes to a formulation based on velocity and pressure fields, enabling three-dimensional flow simulations. Although LBM is inherently a time-dependent algorithm, in this work focuses on steady-state solutions, treating the temporal evolution as an iterative approach toward convergence.

- **Design a flexible QLBM model that can reconstruct correctly the stress tensor:** extend population movement to include diagonal directions, allowing for proper stress tensor recovery and enabling analysis in D2Q9 lattice models.

- **Implement body force handling within the quantum algorithm:** incorporate source terms such as pressure gradients or gravitational forces, which are essential for driving flow in porous and channel environments.

- **Incorporate boundary conditions into the quantum framework:** integrate halfway bounce-back for solid walls and obstacles (i.e., in the form of grains) and periodic boundary conditions to simulate realistic and repeating domains.

- **Develop quantum subroutines for first-order moment calculation:** implement circuits that compute first-order velocity moments directly from the quantum state, facilitating observable readout of macroscopic quantities, as current quantum algorithms are limited to zero-order moment calculation.
- **Maximize quantum-side computation to reduce classical post-processing:** minimize reliance on classical post-processing by executing most simulation steps directly within the quantum framework, and aim to validate the approach using quantum simulators that approximate the constraints of actual quantum hardware.

3.3 Thesis Organization

Following the establishment of research objectives, the thesis proceeds to the core contributions beginning in Chapter 4. This chapter details the quantum implementation, starting with the main circuit for propagating microscopic densities, followed by modular subcircuits for summation, division, and velocity moment extraction. A hybrid quantum–classical strategy is also introduced to address computational limitations of current quantum hardware. Chapter 5 presents convergence and validation of the developed solver, followed by simulation results and their analysis. Chapter 6 concludes the thesis by summarizing the key contributions and findings, reflecting on the motivations and objectives, reviewing both classical and quantum LBM results, and discussing broader implications. It also addresses the main limitations encountered, particularly those related to quantum circuit complexity and simulation constraints and proposes potential future research directions.

CHAPTER 4 METHODOLOGY

This chapter outlines the methodology adopted to simulate fluid flows in porous media using quantum-enhanced techniques. The approach integrates classical fluid dynamics concepts with quantum circuit design, leveraging statevector simulation and modular quantum operations. Due to hardware constraints and the no-cloning principle in quantum computing, the overall computational pipeline is decomposed into smaller, independent quantum circuits. Each circuit handles a specific aspect of the simulation, allowing efficient management of quantum resources while enabling parallelization of operations where possible.

4.1 Simulation Overview

The simulation focuses on modeling a two-dimensional fluid flow through a porous medium on an $M \times M$ grid using the velocity-pressure formulation. The computational domain applies periodic boundary conditions on all four edges to mimic a periodic porous structure, while halfway bounce-back boundary conditions are implemented on internal obstacle nodes to simulate no-slip solid walls at the surface of the grains (obstacles).

The input to the simulation consists of macroscopic density values and zero velocity at each node, which are encoded into quantum states as the initial condition for the circuits. The target output is the velocity components u and v at each node, representing the resolved fluid flow field.

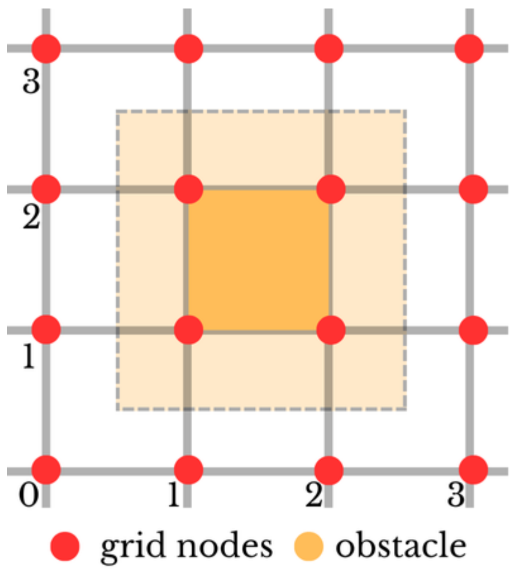


Figure 4.1 Reduced grid model (4x4)

Since quantum circuits lose all state information upon measurement or reset, the simulation is modularized into multiple quantum circuits, each assigned a distinct task such as applying boundary conditions or computing first-order moments required for velocity calculation. This modularity not only accommodates the constraints of quantum hardware but also enables parallelization of computations, making the method scalable and adaptable for future quantum architectures.

To facilitate clear presentation and interpretation of the methodology, a reduced model is adopted for pedagogical purposes. Specifically, a 4×4 lattice with a central 2×2 obstacle representing a porous grain is used, as shown in Fig.4.1. Although this small structure is not suitable for visualizing realistic fluid flow, it provides the simplest framework to understand the circuit structure and functionality. Additionally, some of the operations demonstrated in the quantum circuits are inspired by previous works [46], [50] and [51]. For readers seeking further details on the origin and theoretical basis of these operations, please refer to the corresponding references.

4.2 Quantum Implementation Strategy

Two primary implementation strategies were evaluated: a *fully quantum* modular workflow and a *hybrid* modular workflow in which quantum circuits act as coprocessors for classical routines. Since quantum circuits cannot store information after execution, the simulation was divided into six separate circuits. Each circuit is responsible for a specific part of the computation and is run either one after the other or in parallel, depending on how the data flows between them. Detailed circuit-level designs, including gate schematics, qubit register mappings, and formal input-output specifications, are provided in the following sections.

1. **Main circuit:** This circuit successfully implements the LBM collision and propagation steps, and enforces the periodic boundary condition at the domain boundary and halfway bounce-back boundary condition at the obstacle boundary. The output from this circuit are the *population* f_α at every lattice node.

2. **Zero-order moment (density) calculation circuit:** The sole purpose of this circuit is to calculate the *macroscopic* density at each node, which is the zeroth-order moment of the populations.

$$\rho(x,y,t) = \sum_{\alpha=0}^{N} f_\alpha(x,y,t) \tag{4.1}$$

   where $f_\alpha(x,y,t)$ is the population (i.e., probability distribution funciton) at each node and N is the number of lattice directions.

3. **First-order moment (momentum) calculation circuits:** Series of circuits 3(a)-3(d) are used to calculate the first-order moment of the LBM scheme as shown in Eq.4.2, where $\mathbf{e}_{ix}, \mathbf{e}_{iy}$ are the x and y components of $\mathbf{e}_\alpha = (\mathbf{e}_{\alpha x}, \mathbf{e}_{\alpha x})$.

$$u(x,y,t) = \frac{1}{\rho(x,y,t)} \sum_{\alpha=0}^{N} f_\alpha(x,y,t)\,\mathbf{e}_{ix}, \quad v(x,y,t) = \frac{1}{\rho(x,y,t)} \sum_{\alpha=0}^{N} f_\alpha(x,y,t)\,\mathbf{e}_{iy} \tag{4.2}$$

(a) **Division circuit:** The division circuit helps perform an element-wise division of each population component by its associated macroscopic density (Eq.4.3).

$$k_\alpha(x,y,t) = \frac{f_\alpha(x,y,t)}{\rho(x,y,t)} \tag{4.3}$$

where $k\alpha(x,y,t)$ is the intermediate calculation required to obtain the first-order moment.

(b) **Discrete velocity circuits:** The discrete velocity circuit is executed separately for the x and y components. In each case, the corresponding component of the discrete velocity vector is applied to *all* $k_\alpha(x,y,t)$ values encoded in the statevector as shown in Eq.4.4.

$$k_{dx}(x,y,t) = \frac{f_\alpha(x,y,t)}{\rho(x,y,t)}e_{ix}, \quad k_{dy}(x,y,t) = \frac{f_\alpha(x,y,t)}{\rho(x,y,t)}e_{iy} \tag{4.4}$$

(c) **SWAP circuits:** These circuits, distinct from the quantum SWAP gate, are applied separately to the x and y components. Their purpose is to rearrange relevant data values to simplify the final computation step, as described in Eq.4.5. The name "SWAP" reflects this logical reordering of quantities within the simulation framework.

$$k^*_{dx}(x,y) = \frac{f_\alpha(x,y,t)}{\rho(x,y,t)}e_{ix}, \quad k^*_{dy}(x,y,t) = \frac{f_\alpha(x,y,t)}{\rho(x,y,t)}e_{iy} \tag{4.5}$$

(d) **Selection summation circuit:** This step completes the LBM computation. The swapped state $k^*_\alpha(x,y,t)$ and the original state $k_\alpha(x,y,t)$ are selectively combined based on their directional components (i.e. terms corresponding to the x- and y-coordinates are summed separately). The final results are the velocity components in both the x and y directions at each node, as shown in Eq.4.2.

All circuits are executed at each time step, with each circuit simultaneously processing all nodes across the grid. Due to fundamental hardware limitations, the simulation is modularized, as quantum circuits cannot preserve intermediate computed values across executions unlike classical systems. This modular structure, however, offers key advantages: after the *division circuit*, circuits 4, 5, and 6 which are responsible for computing all the components of velocity can be executed in parallel, since these directional components are independent. This architecture is inherently scalable; for instance, extending the model to three dimensions would not increase the computational time per step, as the additional component could be

computed concurrently.
The discussion begins with the encoding strategy, followed by the implementation details for both the fully quantum modular approach and the hybrid quantum-classical modular workflow. A breif schematic of the entire hybrid and quantum process is shown in Fig.4.2.

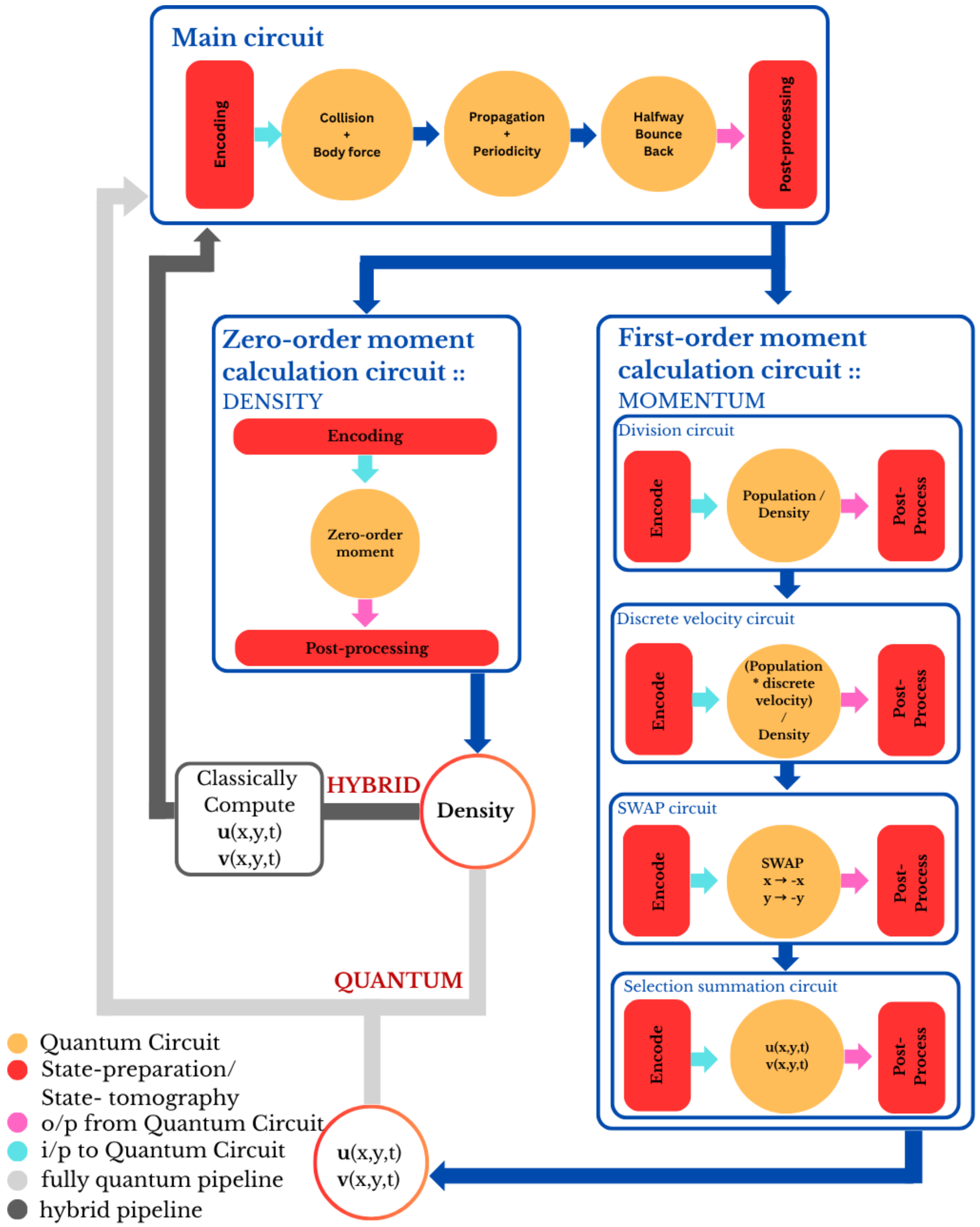


Figure 4.2 Overall schematic of the quantum and hybrid circuitry

4.3 Quantum System Architecture

4.3.1 Input encoding

As discussed in Section 2.1.2, encoding refers to the process of mapping classical data onto quantum states in a form suitable for quantum computation. For this architecture, *amplitude encoding* was adopted, as demonstrated in [52]. In amplitude encoding, a normalized[1] classical vector is embedded directly into the amplitudes of a quantum state:

$$|\psi\rangle = \sum_{i=0}^{2^n-1} a_i |i\rangle, \quad \text{where} \quad \sum_{i=0}^{2^n-1} |a_i|^2 = 1. \tag{4.6}$$

where n is the total number of qubits required. This technique was chosen for compatibility with foundational circuits from which this work draws inspiration, as several sub-circuits rely on amplitude-based logic. Additionally, amplitude encoding scales well with input dimension, encoding 2^n features using only n qubits.

Amplitude encoding is included in the Qiskit SDK [53] and is also widely recognized as a generic method in quantum computing literature. However, its practical use can be limited by the high circuit depth needed for arbitrary state preparation, especially as the input dimension grows. In this architecture, three amplitude-encoded quantum states are required: (i) the macroscopic density statevector $|\rho\rangle$, (ii) the population statevector $|f_\alpha\rangle$, and (iii) the unitary statevector $|U\rangle$ (all amplitudes in this statevector are 1). These states serve as the initial quantum representations for subsequent computations:

1. **Macroscopic density statevector $|\rho\rangle$:** This state encodes the macroscopic density values for all nodes. The density value for each node is repeated seventeen ($2\times(N-1)+1$, where N is the total number of discrete velocities) times in the statevector, structured to match the directional layout used in boundary condition computations. In the D2Q9 model, these directions correspond to: *(right, left, right, left, up, down, up, down, top right, bottom left, top right, bottom left, top left, bottom right, top left, bottom left, rest).* This repetition of directions is done to facilitate halfway bounce-back boundary condition. A final zero padding is added to align the encoded statevector with the n-element Hilbert space [2]. This encoding is only used in the main circuit. The rationale for the specific ordering and structure will be further elaborated in Section 4.3.2, which

[1]This is required by quantum mechanics to ensure the state vector has unit norm (total probability equals one). L^2 normalization was chosen here for its numerical stability and ability to preserve vector direction, maintaining relative magnitudes among features.

[2]The Hilbert space is the complex vector space in which quantum states reside, with each qubit doubling its dimensionality (i.e., n qubits span a 2^n-dimensional space)

discusses qubit register mappings and the indexing scheme.

2. **Population statevector $|f_\alpha\rangle$:** This state encodes the computed population values for all nodes in the D2Q9 model. The values are arranged in the directional order: *(rest, right, left, up, down, top right, top left, bottom left, bottom right)*, repeated across all lattice sites. This structure is designed to support operations in circuit types 2, 3, 4, and 5. As with the previous encoding, zero-padding is added to ensure alignment with the full Hilbert space.

3. **Unitary statevector $|U\rangle$:** This state encodes a value of 1 into a basis state that aligns with the Hilbert space dimensionality required for the corresponding circuit. It is exclusively used in Circuit 6 to define a unitary base state upon which further quantum operations are performed.

4.3.2 Main circuit

The main quantum circuit performs all core LBM operations for one time step. It takes the macroscopic density statevector as input and applies the collision operator, body force, propagation, and both periodic and halfway bounce-back boundary conditions. The circuit schematic is shown in Fig.4.3. The total number of qubits in the circuit is divided into four registers. The r_x and r_y registers represent spatial coordinates of all nodes along the x and y axes, requiring $2 \times \lceil \log_2 M \rceil$ qubits, where $M \times M$ is the size of the lattice. The d register encodes the discrete velocity directions associated with the populations f_α, requiring $\lceil \log_2 N \rceil$ qubits, where N is the number of velocity directions in the model. Finally, the ancilla register a consists of a single qubit used for intermediate calculations during circuit operations.

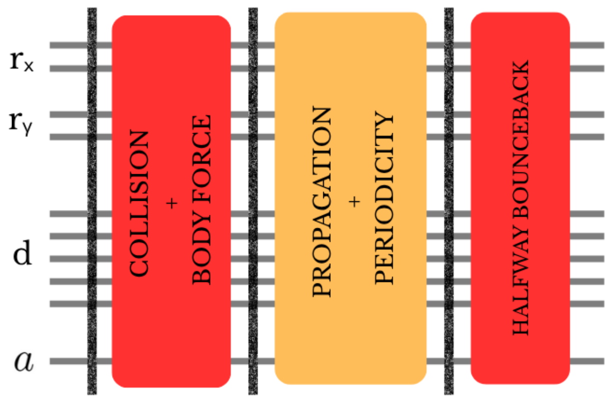


Figure 4.3 Main circuit schematic

To better understand this structure, consider the reduced 4×4 grid model shown in Fig.4.1. In this setup, the initial macroscopic density is distributed uniformly across all fluid nodes. Each discrete velocity direction (rest, left, right, up, down, and the four diagonals) is initially assigned the same probability density value, meaning all populations start with a uniform distribution. Although the physical model redistributes these population values according to

LBM equilibrium weights during the collision step, the input macroscopic density statevector is initialized uniformly for simplicity. The directional weighting is later applied by the collision operator. Thus, the initial quantum state $|\rho\rangle$ can be set as:

$$\begin{aligned} |\rho\rangle = |&\text{right}, \text{left}, \text{right}, \text{left}, \text{up}, \text{down}, \text{up}, \text{down}, \\ &\text{top right}, \text{bottom left}, \text{top right}, \text{bottom left}, \\ &\text{top left}, \text{bottom right}, \text{top left}, \text{bottom right}, \\ &\text{rest}, \text{zero padding}\rangle \end{aligned} \tag{4.7}$$

The repetition and ordering of directions are intentional and will be clarified in upcoming sections on boundary conditions and propagation mechanisms (Sec. 4.3.2).
Suppose we assign a macroscopic density of $1\,\text{kg/m}^3$ to the system. As mentioned earlier, initializing the circuit involves distributing this density equally across all discrete population density functions. To better understand this setup, we focus on the point $(x = 2,\ y = 0)$, specifically examining the rest population. Our goal is to determine the corresponding binary basis state so that this configuration can be directly indexed in the macroscopic density statevector.
Given the grid size $M = 4$, the number of qubits required for each coordinate is $\lceil \log_2 4 \rceil = 2$. Each 2-qubit register $(r_x,\ r_y)$ can represent the binary combinations 00, 01, 10, and 11, corresponding to spatial coordinates 0 through 3 [3].
As discussed in Section 2.1.2, the basis states are organized as $|q_a q_d q_y q_x\rangle$, where:

- q_a: ancilla qubit used for intermediate calculations,
- q_d: direction register representing the discrete velocity,
- q_y: 2-qubit register for the y-coordinate (e.g., 00 for $y = 0$),
- q_x: 2-qubit register for the x-coordinate (e.g., 10 for $x = 2$).

Given this configuration, the intermediate basis state becomes: $|q_a q_d 00\,10\rangle$.
The direction register q_d requires $\lceil \log_2 N \rceil$ qubits, where Q is the number of discrete velocity directions. Although the D2Q9 model inherently includes nine directions (rest, left, right, up, down, top right, top left, botom left, bottom right), precise and accurate implementation of boundary conditions required duplicating the non-rest directions. This prevents incorrect values from being picked up during boundary updates. As a result, the model uses $Q = 17$ discrete directions: two copies each of left, right, up, down, top right, top left, bottom left

[3]Each qubit can be in the state $|0\rangle$ or $|1\rangle$, resulting in 2^n possible outcomes, where n = number of qubits

and bottom right plus a single rest direction. To accommodate these, the required number of qubits is: $\lceil \log_2 17 \rceil = 5$. These 5 qubits span binary strings from `00000` to `11111`. Returning to the example of $(x = 2, y = 0)$ in the rest direction, the rest velocity is encoded as binary string `10000`. Hence, the complete basis state becomes: $|q_a\, 10000\, 00\, 10\rangle$.
The ancilla register q_a is a single qubit used during intermediate computations such as the collision operation. It can take values $|0\rangle$ or $|1\rangle$, depending on the logic of the circuit. However, in the initial statevector, before any gate operations, this qubit is always initialized to $|0\rangle$. Hence, the complete basis state for a fluid particle located at $x = 2, y = 0$, with rest velocity is: $|0100000010\rangle$.
This basis state serves as the index in the quantum statevector where a macroscopic density of 1 kg/m^3 is assigned. The quantum circuit corresponding to this reduced 4×4 model is illustrated in Fig.4.4. The L.C.U and the propagation (right, left, up, down) gates in the figure, are inspired by Budinski's work [46].

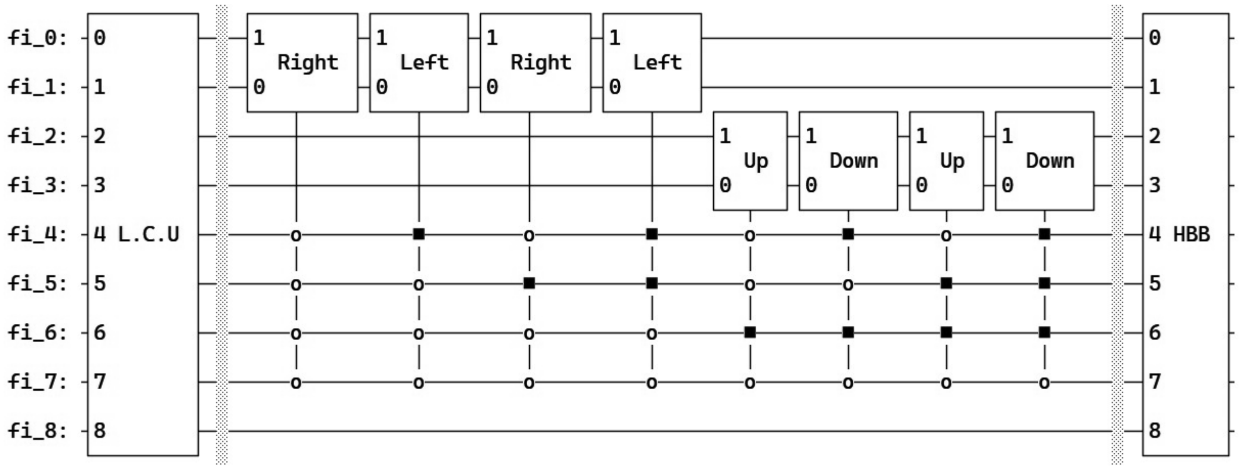


Figure 4.4 Main circuit (4x4 grid) schematic

When correlated to Fig.4.4, L.C.U is the collision + body force segment, right, left, up, down are the propagation + periodicity segment and HBB is the halfway bounceback segment.
The population density function $\rho(\alpha, \mathbf{r})$ is structured as a linear array of density values across all directions and positions, represented as:

$$\rho(\alpha, \mathbf{r}) = [\rho(0,0),\ \rho(0,1),\ \rho(0,2),\ \ldots,\ \rho(N, K{-}2),\ \rho(N, K{-}1),\ \rho(N, K)], \tag{4.8}$$

where $\alpha = 0, 1, \ldots, N$ indexes the discrete velocity directions, and M denotes the total number of spatial nodes in the lattice (i.e., $K = M \times N$). Each term $\rho(\alpha, \mathbf{r})$ specifies the population at a given position $\mathbf{r} = (x, y)$ and direction α. Now that the density function has been defined, the initial quantum statevector, prior to the application of any circuit operations, can be expressed as:

$$|\psi_0\rangle = |0\rangle_a \otimes \frac{1}{\|\rho\|} \sum_{i=0}^{K-1} \rho(\alpha, \mathbf{r})_i \, |i\rangle, \tag{4.9}$$

where $\rho(\alpha, \mathbf{r})$ are the population density functions encoded as amplitude coefficients, and $|i\rangle$ denotes the computational basis states as previously described. The index i iterates over all basis states, and the full statevector is normalized by the L^2-norm $\|\phi\|$ to ensure unit probability.

Collision operator and body force

In LBM, the collision operator [46] redistributes the population density functions toward their equilibrium values based on local macroscopic properties such as velocity and density. When a body force $\mathbf{F}_\alpha$ is present, the equilibrium distribution f^{eq} is modified to account for this external influence:

$$f^{\text{eq}}(\alpha, \mathbf{r}, \mathbf{t}) = w_\alpha \rho(\mathbf{r}, \mathbf{t}) \left(1 + \frac{\mathbf{e}_\alpha \cdot \mathbf{u}(\mathbf{r}, \mathbf{t})}{c_s^2} + \mathbf{F}_\alpha\right), \tag{4.10}$$

In a quantum context, the population distribution function $\rho(\alpha, \mathbf{r}, t)$ is encoded in a quantum statevector $|\psi_0\rangle$. To perform the collision step, the quantum state must be transformed according to the modified equilibrium distribution f^{eq}. This is captured by applying a diagonal matrix A constructed from the equilibrium values f_α^{eq}, the decomposition becomes:

$$A = \begin{bmatrix} f_1^{\text{eq}} I_n & 0 & \cdots & 0 \\ 0 & f_2^{\text{eq}} I_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f_K^{\text{eq}} I_n \end{bmatrix}, \tag{4.11}$$

where, I_n is the identity matrix. The post-collision state becomes:

$$|\psi_1\rangle = A|\psi_0\rangle. \tag{4.12}$$

However, since A is not generally unitary, it cannot be implemented directly on a quantum computer. To overcome this, the LCU framework is employed [54], which allows a non-unitary Hermitian[4] matrix A with spectral norm $\|A\| \leq 1$ to be decomposed as:

[4]A Hermitian matrix satisfies $A = A^\dagger$ and has real eigenvalues, making it suitable for quantum operations.

$$A = \frac{1}{2}(C_1 + C_2), \quad \text{where} \quad C_1 = A + i\sqrt{I - A^2}, \quad C_2 = A - i\sqrt{I - A^2}, \tag{4.13}$$

and C_1, C_2 are unitary matrices, with the corresponding LCU unitary components:

$$C_{1,2} = \begin{bmatrix} \exp\left(\pm i \arccos(f_1^{\text{eq}})\right) I_n & 0 & \cdots & 0 \\ 0 & \exp\left(\pm i \arccos(f_2^{\text{eq}})\right) I_n & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \exp\left(\pm i \arccos(f_K^{\text{eq}})\right) I_n \end{bmatrix}. \tag{4.14}$$

These unitaries are embedded into a larger unitary matrix U using block encoding techniques [55, 56] (Fig.4.5):

$$U = \frac{1}{2}\begin{bmatrix} C_1 + C_2 & C_1 - C_2 \\ C_1 - C_2 & C_1 + C_2 \end{bmatrix}, \tag{4.15}$$

such that the top-left sub-block of U effectively applies the matrix $A = \frac{1}{2}(C_1 + C_2)$ to the quantum state. The final result is extracted through post-selection on an ancilla qubit in the $|0\rangle$ state, discarding the unwanted component $|1\rangle$. The resulting post-collision quantum state is:

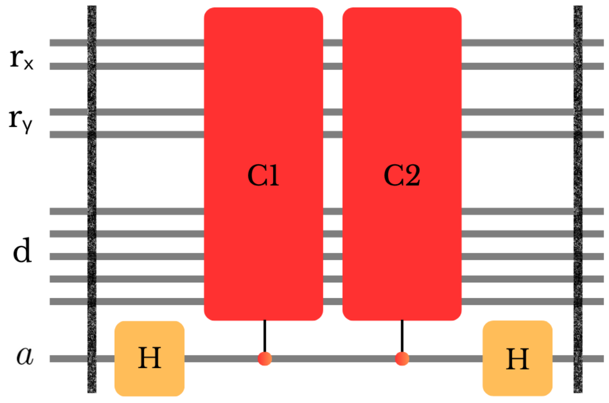


Figure 4.5 Block encoding of $B = (C_1 + C_2)/2$. This correspomds to the L.C.U segment in Fig.4.3.

$$|\psi_1\rangle = |0_a\rangle \otimes \frac{1}{\|\rho\|} \sum_{i=0}^{K-1} f_i^{\text{eq}} \rho(\alpha, \mathbf{r})_i \, |i\rangle \,, \tag{4.16}$$

where $\rho(\alpha, \mathbf{r})_i$ denotes the amplitude associated with velocity direction α at the i^{th} grid point, with $i = 0, 1, \ldots, K-1$. This approach provides a physically consistent and resource-efficient mechanism for implementing the collision operator in quantum LBM, as successfully shown by [46, 50].

Propagation and periodic boundary condition

The propagation step is responsible for advancing the particle distributions along their respective velocity directions to neighboring lattice sites. This step is governed by the rule shown in Eq. 2.13:

$$f_\alpha(\mathbf{r} + \mathbf{e}_\alpha \Delta t, t + \Delta t) \leftarrow f_\alpha^*(\mathbf{r}, t), \tag{4.17}$$

where f_α^* denotes the post-collision distribution and $\mathbf{e}_\alpha$ is the lattice velocity vector corresponding to direction α. This operation represents a deterministic translation along the lattice in direction α, and it constitutes the essential mechanism for transferring information across the grid.

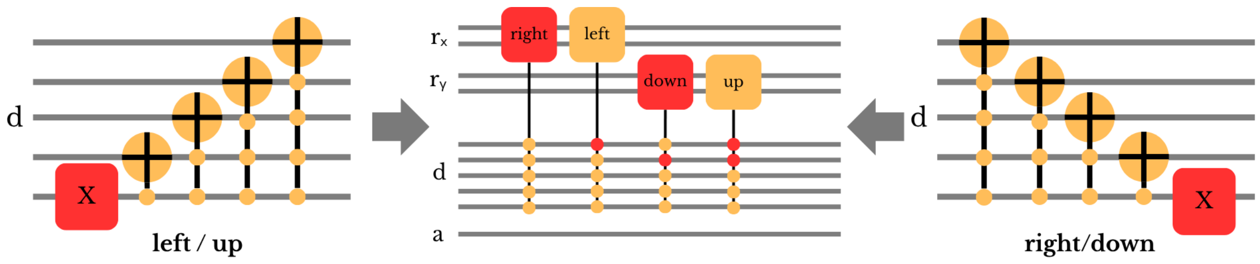


Figure 4.6 Propagation gates consisting of the left, right, up, and down operators are decomposed into CNOT and X gates, as shown in the left and right panels. Red control dots indicate $|0\rangle$, while yellow control dots indicate $|1\rangle$

In the classical LBM, propagation can be viewed as a discrete walk on a structured grid, where each population f_α shifts to a neighboring lattice site in the direction specified by α. In the quantum formulation, this step is implemented through a quantum walk. Controlled shift operators act on the position registers q_x and q_y, conditioned on the directional register q_d, which encodes the discrete velocity direction α. This approach is inspired by one-dimensional quantum walk demonstrations [57,58]. The shift operators, denoted R_n and L_n, are unitary permutation matrices that cyclically permute the entries of the quantum state. Specifically, R_n shifts amplitudes to the right (incrementing position), while L_n shifts to the left (decrementing position). Analogous operations are performed along q_y for vertical motion. Diagonal movement is achieved by applying the corresponding x- and y- shifts in sequence. Due to R_n and L_n being permutation matrices, the corresponding circuits require only CNOT and X gates and are inherently unitary. Moreover, the cyclic nature of these permutations naturally enforces periodic boundary conditions.

$$R_M = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}, \quad L_M = \begin{bmatrix} 0 & 1 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{4.18}$$

The resulting statevector $|\psi_2\rangle$ represents the distribution after propagation, shifted according to the encoded velocity direction. Figure 4.6 illustrates how the left and right shift gates are applied conditionally based on the value of q_d.

Halfway bounce-back boundary condition

The halfway bounce-back boundary condition is a widely used technique to enforce no-slip walls. In this approach, incoming particle populations at the fluid–solid interface are reflected back along their incoming directions, effectively reversing their velocities and preventing fluid penetration into the solid region. Crucially, the bounce-back operation is not performed on the obstacle node itself but rather on its neighboring fluid node. This ensures that the reflection takes place halfway between the obstacle and the fluid node, thereby preserving second-order accuracy. The discrete form of this operation is given by Eq.(4.19):

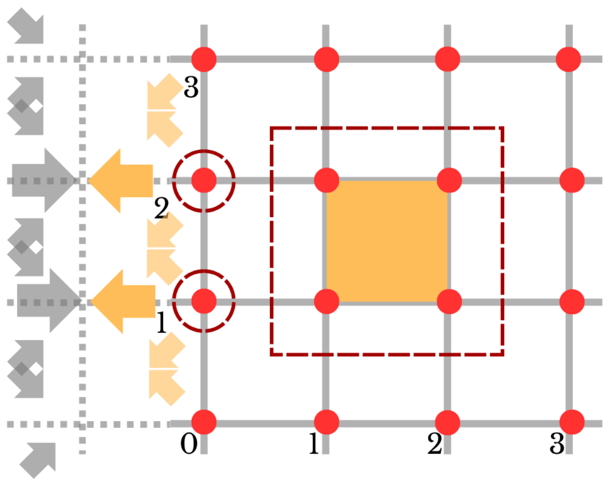


Figure 4.7 Illustration of the halfway bounce-back mechanism at the left wall

$$f_\alpha(\mathbf{r}, t + \Delta t) = f_\alpha(\mathbf{r}, t), \tag{4.19}$$

where f_α represents the reflected population traveling in the opposite direction to α, and $\mathbf{r}$ denotes the spatial position of the fluid node adjacent to the obstacle.
In the quantum implementation, the binary encodings of all neighboring fluid nodes adjacent to obstacle boundaries are first identified and stored during the state preparation phase. These binary strings, which represent the spatial coordinates in the quantum statevector basis, are later concatenated with the appropriate population direction to form the full quantum state labels.

For example, consider the *left wall* of the obstacle in Fig.4.1. The bounce-back operation will act on the neighboring fluid nodes located to the left of the wall, namely at positions $(1, 0)$ and $(2, 0)$, assuming a (Y, X) axis convention. This operation should swap the *right-moving populations* (incoming towards the wall) with their *left-moving* counterparts (reflected away from the wall), as illustrated in Fig.4.7.

From the statevector layout defined in Eq.4.7, the binary strings corresponding to the spatial coordinates $(1,0)$ and $(2,0)$ are: $(1,0) \rightarrow 0100, \quad (2,0) \rightarrow 1000$. To clarify the rationale behind the specific layout of the statevector in Eq.(4.7), the four most significant qubits encode the population direction. These directions are mapped as shown in Table 4.1:

Table 4.1 Bitstring assignments for each direction

Direction	**Bitstring**	**Direction**	**Bitstring**
Right	00000	Top right (2nd)	01010
Left	00001	Bottom left (2nd)	01011
Right (2nd)	00010	Top left	01100
Left (2nd)	00011	Bottom right	01101
Up	00100	Top left (2nd)	01110
Down	00101	Bottom right (2nd)	01111
Up (2nd)	00110	Rest	10000
Down (2nd)	00111		
Top right	01000		
Bottom left	01001		

To implement the bounce-back on the left wall, the known population states, specifically, the *right-moving populations*, are:

- For coordinate $(1,0)$: $|00000\ 01\ 00\rangle$, $|00010\ 01\ 00\rangle$
- For coordinate $(2,0)$: $|00000\ 10\ 00\rangle$, $|00010\ 10\ 00\rangle$

the corresponding *left-moving populations* are:

- For coordinate $(1,0)$: $|00001\ 01\ 00\rangle$, $|00011\ 01\ 00\rangle$
- For coordinate $(2,0)$: $|00001\ 10\ 00\rangle$, $|00011\ 10\ 00\rangle$

the corresponding *top right-moving populations* are:

- For coordinate $(1,0)$: $|01000\ 01\ 00\rangle$, $|01010\ 01\ 00\rangle$
- For coordinate $(2,0)$: $|01000\ 10\ 00\rangle$, $|01010\ 10\ 00\rangle$

the corresponding *top left-moving populations* are:

- For coordinate $(1, 0)$: $|01100\ 01\ 00\rangle$, $|01110\ 01\ 00\rangle$
- For coordinate $(2, 0)$: $|01100\ 10\ 00\rangle$, $|01110\ 10\ 00\rangle$

the corresponding *bottom left-moving populations* are:

- For coordinate $(1, 0)$: $|01001\ 01\ 00\rangle$, $|01011\ 01\ 00\rangle$
- For coordinate $(2, 0)$: $|01001\ 10\ 00\rangle$, $|01011\ 10\ 00\rangle$

and the corresponding *bottom right-moving populations* are:

- For coordinate $(1, 0)$: $|01101\ 01\ 00\rangle$, $|01111\ 01\ 00\rangle$
- For coordinate $(2, 0)$: $|01101\ 10\ 00\rangle$, $|01111\ 10\ 00\rangle$

Careful observation reveals that only a single qubit differs between the opposite-moving counterparts. This design choice is intentional and fundamental to the layout of the statevector. In the quantum circuit, all qubits other than the differing one serve as control qubits, while a CNOT gate is applied to the differing qubit. This controlled operation effectively swaps the opposite-moving populations, thereby simulating the bounce-back phenomenon. The corresponding quantum circuit structure is illustrated in Fig.4.8. While this implementation remains scalable and flexible, applicable to any number of obstacles by targeting specific coordinates, the principal limitation lies in circuit depth. As the number of obstacle boundaries increases, the number of controlled operations grows accordingly, leading to a longer circuit.

Once the halfway bounce-back criterion has been successfully enforced, the final statevector $|\psi_3\rangle$ now reflects this boundary condition.

Finally, during the post-processing phase, only the relevant data, corresponding to physically accurate population values is retained, while all extraneous amplitudes are discarded. This filtered data is then restructured into the population density function statevector $|f_\alpha\rangle$, which serves as the input for subsequent quantum circuit iterations. A key observation to note would be that, instead of merging the bounceback operation with propagation, this gate is applied *after* propagation. This setup, forces propagation of population to occur twice, which may degrade the order of accuracy of the solver.

Similar work has been done by Schalkers and Moller [26, 51], however, an entirely different encoding process was developed and was only demonstrated for orthogonal bounce back conditions.

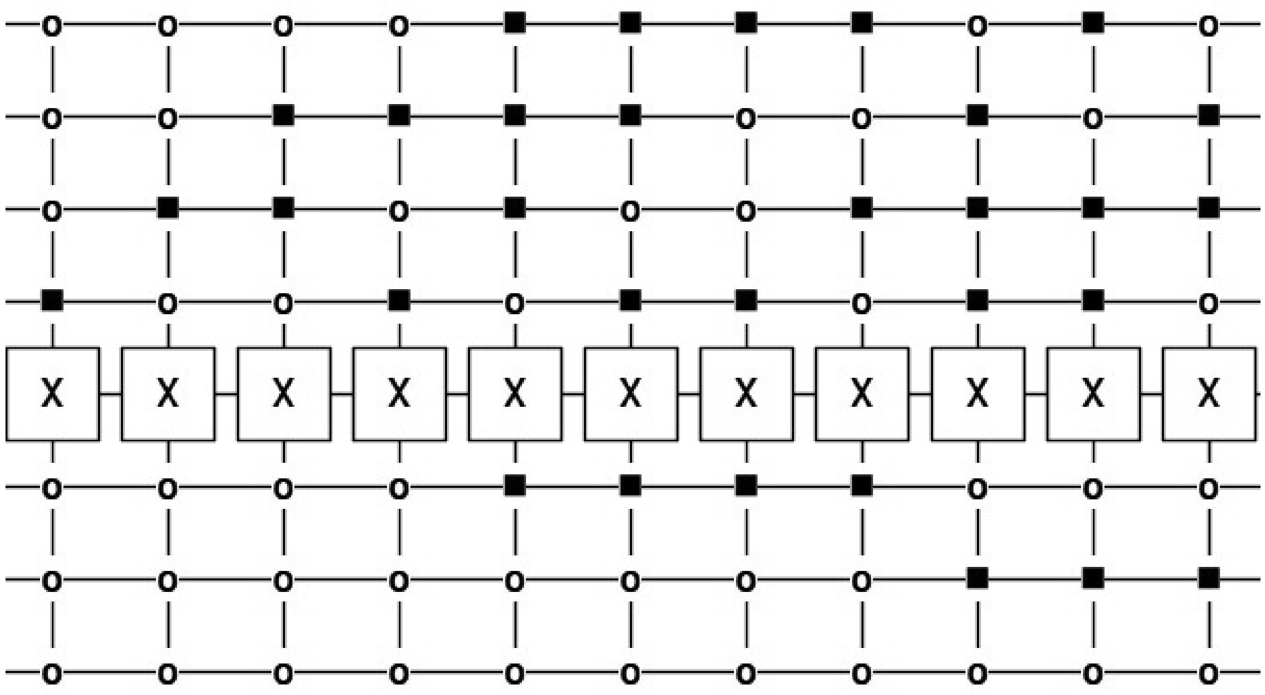


Figure 4.8 Halfway bounce-back schematic where each CNOT gate represents one bounceback node. This corresponds to the HBB segment in Fig.4.3.

4.3.3 Zero-order moment (density) calculation circuit

This circuit computes the zeroth-order moment Eq.(2.20), i.e., the macroscopic density field, in the LBM framework. It takes as input the microscopic density statevector produced by the main quantum evolution circuit. No ancilla qubits are required. The layout of the statevector is defined as:

$$|\rho\rangle = |rest, right, left, top, bottom, topright, topleft, bottomleft, bottomright, \\ zeropadding\rangle \tag{4.20}$$

The summation is achieved by applying Hadamard gates to the three qubits of the directional register. The Hadamard gate H transforms a single-qubit state $|\psi\rangle = a|0\rangle + b|1\rangle$ as:

$$H|\psi\rangle = \frac{a+b}{\sqrt{2}}|0\rangle + \frac{a-b}{\sqrt{2}}|1\rangle \tag{4.21}$$

This operation creates a superposition of directional states, encoding the sum of amplitudes in the $|0\rangle$ component and the difference in the $|1\rangle$ component. To retrieve the total density at each lattice node, only the $|0\rangle$ outcomes of the directional qubits are considered. The contribution of each Hadamard gate introduces a scaling factor of $1/\sqrt{2}$, which is corrected in post-processing by multiplying the final amplitudes by $\sqrt{2}^{\log_2(N_(q_d))}$, where $N_(q_d)$ is the total number of qubits having the Hadamard gate. For example, applying Hadamard gates to the directional register of the basis state $|0000\,00\,00\rangle$, where the last four qubits encode direction, yields:

$$(H|0\rangle \otimes H|0\rangle \otimes H|0\rangle \otimes H|0\rangle \otimes |00\,00\rangle) = \left(\frac{5\sqrt{2}}{2}\sum_{i=0}^{2\times(N-1)+1}|i\rangle\right)\otimes|00\,00\rangle = \frac{5\sqrt{2}}{2}\sum_{i=0}^{2\times(N-1)+1}|i\rangle\otimes|00\,00\rangle \tag{4.22}$$

This generates an equal-weight superposition of all eight directional states associated with the node at $(0,0)$. The amplitudes of the resulting statevector reflect the total population densities summed over directions, effectively encoding the macroscopic density at each site. At each timestep, statevector tomography[5] is used to extract the output of the circuit. The first K components of the statevector, where K is the number of lattice nodes, represent the zeroth-order moment. All other amplitudes, associated with directional and padding states, are discarded.

4.3.4 Division circuit

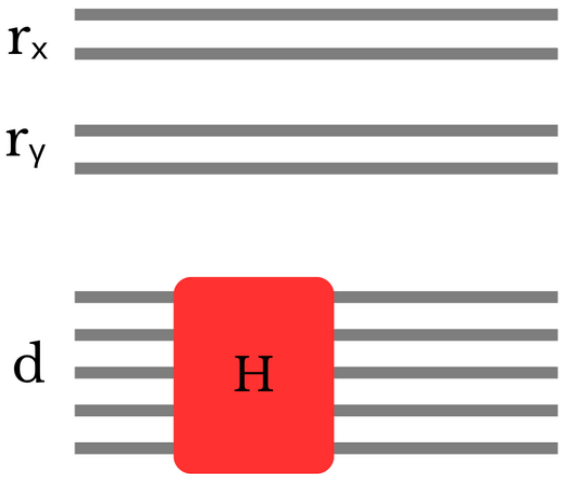
This circuit (Fig.4.9b) performs element-wise division between the population densities from the complete summation circuit and the microscopic densities from the main quantum circuit. As native division is not supported in quantum computation, this operation is implemented by embedding externally computed inverse values into custom quantum gates.

These inverses, derived from the complete summation circuit output, are used to construct a new quantum circuit, the *division circuit.* The microscopic density statevectors serve as input, while the circuit logic follows the same LCU framework as the collision step, including the use of an ancilla qubit for controlled operations.

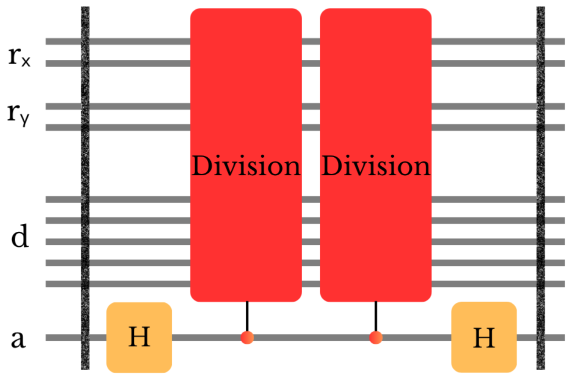
The circuit effectively multiplies the microscopic amplitudes by the inverse macroscopic densities, producing an output state corresponding to Eq.(4.3). During post-processing, irrelevant amplitudes are discarded to isolate the desired result.

Notably, this approach supports spatial parallelization, as each node's calculation is localized. This allows independent processing of all discrete directions, enhancing modularity and scalability.

[5]Statevector tomography is a quantum measurement technique used to reconstruct the complete quantum state (amplitudes and phases) of a system by performing a series of measurements in different bases.



(a)



(b)

Figure 4.9 (a) Quantum circuit for computing the zeroth-order moment (macroscopic density) in the LBM scheme
(b) Division circuit schematic (LCU construction)

4.3.5 Discrete velocity circuit

The discrete velocity circuit computes the directional contributions e_α required for momentum calculations, as defined in Eq.(4.4). The circuit layout (Fig.4.10) remains identical for all direction population density calculations, with the input being the population density statevector obtained from the main quantum circuit. No ancilla qubits are required in this circuit, and the statevector structure follows the same format as that used in the complete summation and density circuits.

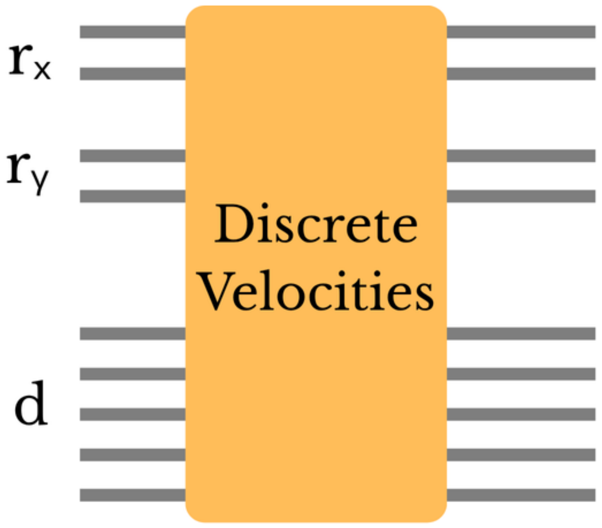


Figure 4.10 Discrete velocity circuit schematic

The only custom operation introduced is a diagonal unitary gate, which applies a phase factor of -1 to amplitude components corresponding to negative directions along the respective axis. This effectively encodes the discrete velocity components e_α for each direction (-1 is specific to D2Q9 conditions only, LCU method should be adopted for other configurations with non-unitary values). Separate circuits are constructed for the x and y components, as each requires a statevector tailored to its respective directional basis.

4.3.6 SWAP circuit

The SWAP circuit is designed to interchange directional components in preparation for the final circuit. Specifically, it swaps the opposing directions in their corresponding circuits. This operation does not modify amplitudes or perform any computation beyond reordering; its sole purpose is to facilitate compatibility and alignment in the subsequent circuit stages. Due to the differing binary basis states that represent directional components, the SWAP gate configurations must be tailored to each coordinate direction, with a combination of either x gate and SWAP gates. For example, distinct gate layouts are used for the x- and y-coordinate SWAP circuits, as illustrated in Fig.4.11.

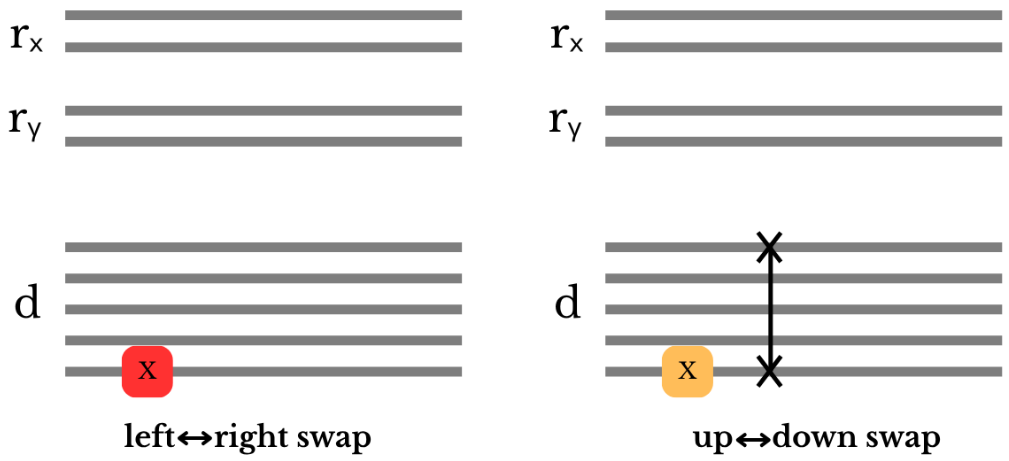


Figure 4.11 SWAP circuit examples

4.3.7 Selection summation circuit

This circuit (Fig.4.12) is a straightforward implementation of the LCU framework, similar to those used in the collision and division operations. However, unlike previous applications where LCU was employed to perform element-wise multiplication, here it is used to perform element-wise addition.

Traditionally, multiplication in LCU circuits involves the combination of two diagonal operators followed by normalization, as described in Eq.(4.15). In this case, the input to the circuit is the unitary statevector, while the two diagonal operators are constructed from the outputs of the discrete velocity and SWAP circuits. This construction is done separately for all the coordinate directions to isolate directional contributions. Thanks to the prior SWAP operation, the summation in this circuit can cleanly aggregate directional components within each coordinate independently.

In the post-processing stage, only the relevant output amplitudes are retained corresponding to the first-order moment values while all others are discarded. This results in the recovery of the macroscopic velocity fields u and v at each lattice node, which are essential for initiating the next time step in the simulation.

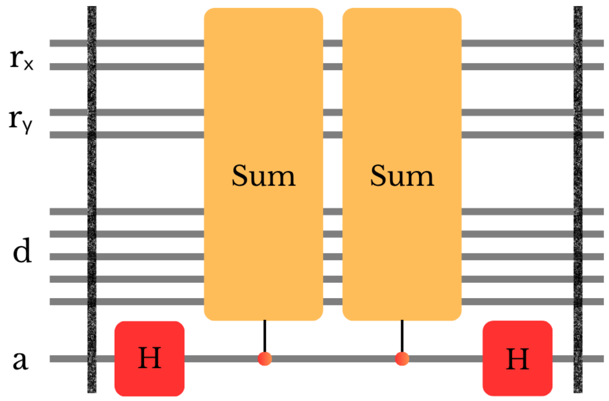


Figure 4.12 First moment circuit schematic

4.4 Quantum–Classical Hybrid Implementation Strategy

While the QLBM framework includes multiple quantum circuits, it is important to highlight that the majority of the LBM scheme, specifically, the propagation, collision and boundary condition steps was implemented within the main quantum circuit. The remaining circuits serve auxiliary purposes and are designed to execute four specific operations: complete summation (zeroth moment), element-wise division, element-wise multiplication, and element-wise addition (first moment). These auxiliary circuits were developed for arithmetic operations that are not natively supported by quantum hardware. Despite their utility, they were primarily used to demonstrate the feasibility of fully quantum implementations.

To assess the potential quantum advantage, a hybrid simulation strategy was also explored. In this variant, only the main circuit was deployed on quantum hardware, effectively using quantum computation as a co-processor. The post-collision zeroth and first moment calculations were instead carried out using classical routines based on analytical expressions. This allowed for a comparative evaluation of performance, modularity, and resource demands between full quantum and hybrid quantum–classical workflows.

CHAPTER 5 RESULTS

This chapter evaluates the classical and quantum LBM solvers through a structured framework of verification, validation, and performance analysis. It begins with code verification using Poiseuille flow simulations to confirm correct implementation of the numerical method. This is followed by solution verification via grid convergence analysis, ensuring that the solver exhibits the expected rate of accuracy improvement with mesh refinement. Following this, code-to-code verification is carried out through simulations of flow through porous media, focusing on local field variations and obstacle-induced effects to assess the solver's ability to reproduce relevant physical behavior. Finally, quantum-specific performance metrics such as gate counts, resource requirements, and runtime trade-offs are examined. This is followed with a scaling analysis to evaluate computational performance and explore the potential for quantum advantage in larger, more complex domains. The chapter concludes with a summary to the results analysed through this chapter. All the simulations in the chapter have been done on a 2 × Intel Gold 6148 Skylake @ 2.4 GHz CPU.

5.1 Code Verification: Poiseuille Flow Test

To verify the correct implementation of the classical and quantum LBM solvers, Poiseuille flow is simulated as a benchmark problem. This flow scenario, characterized by a parabolic velocity profile between two stationary plates, offers a well-defined analytical solution derived from NSE. By comparing the velocity profiles produced by both solvers to the analytical solution, we assess whether the numerical implementations correctly reproduce the expected physical behavior. Due to its simplicity and tractability, it serves as a standard test case for code verification in computational fluid dynamics.
Poiseuille flow models steady, incompressible, and laminar motion between two parallel plates, driven by a constant pressure gradient or an equivalent body force. In the absence of inertial effects, the analytical solution for the horizontal velocity component $u(y)$ in a channel of height H, subject to a uniform body force F, is given by:

$$u(y) = \frac{F}{2\nu}\, y(H - y),$$

where ν is the kinematic viscosity and $y \in [0, H]$ denotes the vertical coordinate across the channel height. This yields a symmetric parabolic profile with a maximum at the channel center and zero velocity at the walls, satisfying the no-slip boundary condition.

To verify correctness, velocity profiles obtained from both classical and quantum LBM solvers are compared against the analytical solution. Simulations are conducted on 4×4, 8×8 and 16×16 grids using kinematic viscosity $\nu = 1/6$, and a body force[1] selected to ensure a Reynolds number below 1. This low-Reynolds regime guarantees Stokes flow, making it ideal for isolating numerical correctness. The quantum solver's outputs are benchmarked against those from the classical implementation. To further assess accuracy, the classical solver, upon which the quantum solver is based is also run on increasingly refined grids to demonstrate convergence toward the analytical solution.

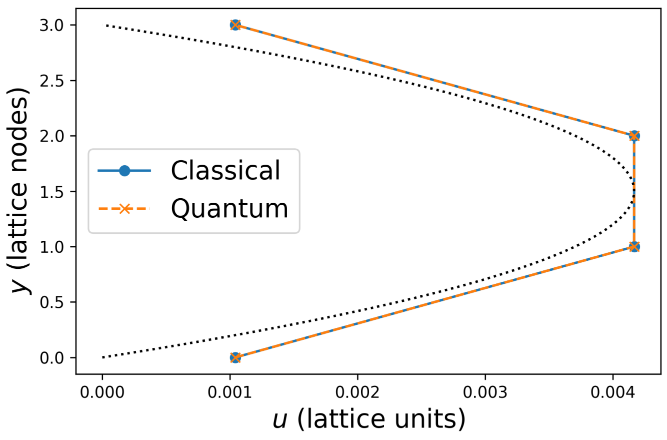


Figure 5.1 Classical LBM and Quantum LBM - 4x4 grid

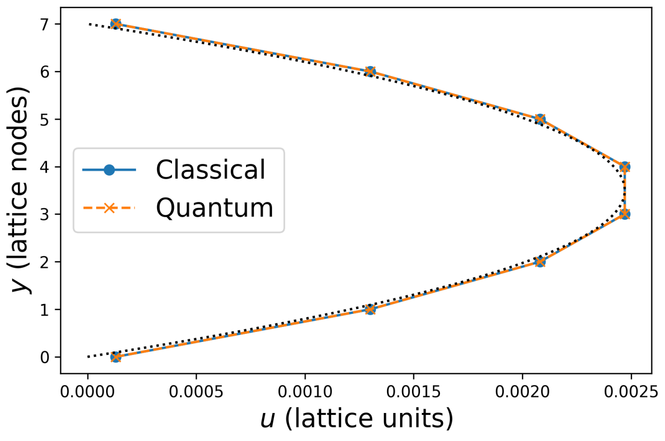


Figure 5.2 Classical LBM and Quantum LBM - 8x8 grid

[1]In LBM simulations, physical quantities are expressed in dimensionless *lattice units*, where characteristic length, time, and mass scales are normalized (e.g., lattice spacing $\Delta x = 1$, time step $\Delta t = 1$). This simplifies the numerical implementation while preserving the essential physics.

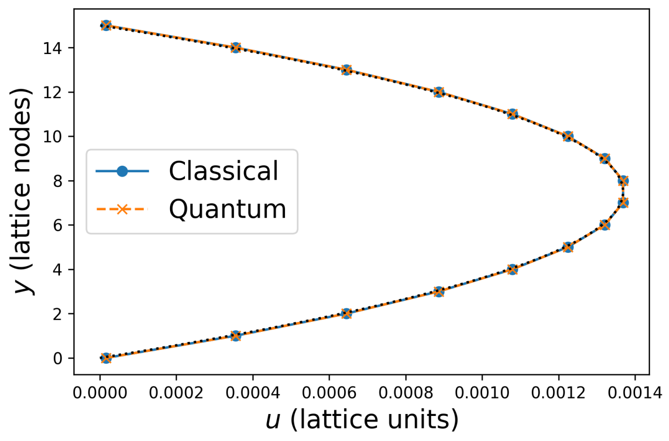


Figure 5.3 Classical LBM and Quantum LBM - 16x16 grid

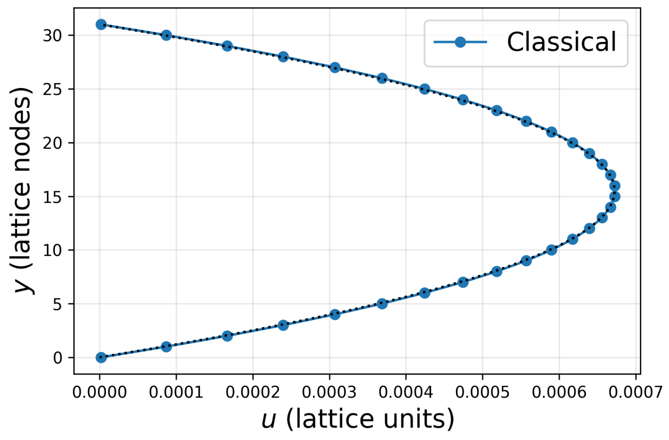


Figure 5.4 Classical LBM benchmarking - 32x32 grid

From Fig.5.1, 5.2, and 5.3, it can be observed that both the classical and quantum LBM solvers successfully reproduce the expected parabolic velocity profile of Poiseuille flow. The velocity profiles produced by both solvers are nearly identical and overlap closely in all cases, indicating strong agreement with the analytical solution. This strong agreement with the analytical solution on coarse grids confirms the correct implementation of the quantum solver. However, increasing the spatial resolution significantly increases the complexity of the quantum circuit, particularly due to the need for additional gates to encode bounce-back boundary conditions at higher fidelity. This leads to an exponential rise in gate count and circuit depth, limiting quantum simulations to relatively small grids in this study.

In contrast, the classical solver on which the quantum implementation is based, can be executed on much finer grids. Results for 32×32 grids (Fig.5.4) continue to show excellent agreement with the analytical solution, reinforcing the accuracy of the underlying numerical scheme. A grid convergence analysis is carried out to further evaluate the solver's numerical accuracy, specifically by examining how the error scales with grid refinement.

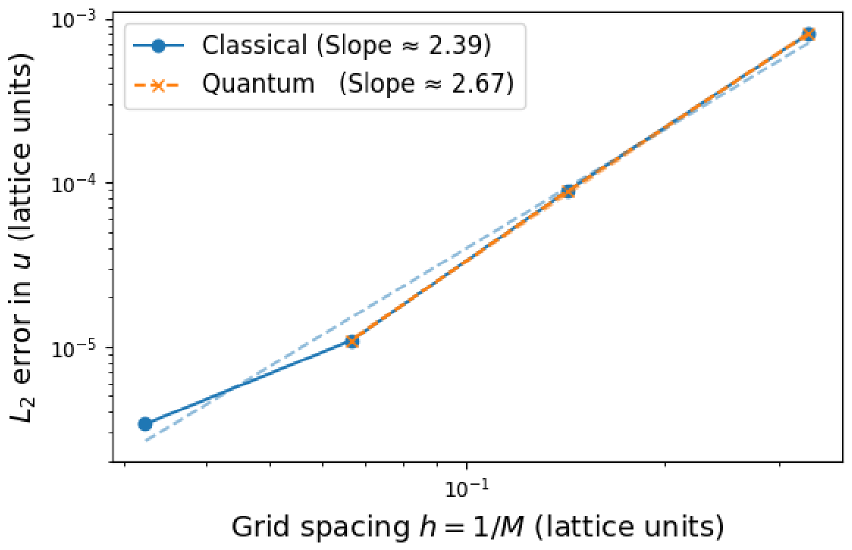


Figure 5.5 Log-log plot of relative L_2 discretization error in velocity profile vs. grid resolution for classical (4, 8, 16, 32) and quantum (4, 8, 16) LBM solvers.

5.2 Solution Verification: Grid Convergence Analysis

To evaluate the numerical accuracy of the solvers, a grid convergence analysis is performed. For LBM methods, second-order convergence is typically expected for flows bounded by straight walls, where the discretization error scales quadratically with the grid spacing. Fig.5.5 presents a log-log plot of the L_2 error versus grid spacing for both classical and quantum solvers.

We observe a convergence rate of 2.39 from the classical solver and 2.60 from the quantum solver, which is higher than the expected second-order behavior. This can be explained by the relationship between the relaxation parameter τ and the viscosity, as described in Eq. (2.14). To simplify the quantum circuit and eliminate nonlinear dependencies, the formulation adopted in this work, following prior literature, fixes $\tau = \Delta x = \Delta t = 1$. Under this constraint, the viscosity remains permanently defined as $\nu = 1/6$, preventing independent variation of the grid spacing to study convergence in a conventional manner. Consequently, maintaining a predefined Reynolds number alters the underlying physics of the flow, introducing a different compressibility error. It is suspected that this effect contributes to the observed convergence rates being higher than expected as the grid is refined.

In very recent work, Xiao et al. [59] investigated the quantum lattice-kinetic scheme and demonstrated that introducing a correction term in the equilibrium distribution enables independent control of viscosity and mesh size while preserving accuracy. Such modifications offer a potential solution to address the limitations of fixed-parameter formulations and may provide a more consistent framework for evaluating convergence in QLBM solvers.

5.3 Code-to-Code Verification: Flow Through Porous Media

5.3.1 Density field analysis

The macroscopic density field, derived as the zeroth moment of the distribution functions, is analyzed across multiple spatial resolutions: 4×4, 8×8, 16×16, and 32×32. Each simulation incorporates internal solid obstacles of varying geometries to simulate porous media conditions. Flow is driven through a periodic domain, with periodic boundary conditions applied along the horizontal direction -x, mimicking the repeating pore structures typical of such materials. Obstacles (grains) are treated as impermeable solids, and the no-slip condition at fluid–solid interfaces is enforced using the halfway bounce-back scheme. Results are presented for both classical and quantum LBM to assess consistency in reproducing key physical trends.

Fig.5.6 through 5.10 show the computed density fields for both solvers alongside their corresponding relative error heatmaps. As expected, the fluid density increases near obstacle surfaces due to local stagnation effects and decreases in regions of unobstructed flow. These spatial patterns are preserved across all grid resolutions, indicating consistent physical behavior with mesh refinement.

Sharp gradients observed around obstacle perimeters reflect the enforcement of no-slip conditions, while the seamless wrapping of flow across domain boundaries illustrates the periodicity in the horizontal direction. These results confirm that the boundary conditions are functioning as intended.

The quantum solver replicates the classical density distributions with high fidelity. This agreement is quantitatively supported by the error heatmaps, which show that relative mean absolute errors (RMAEs) remain below 0.001% across all test cases. As expected, the RMAE values decrease with increasing spatial resolution, indicating that finer grids yield a more accurate representation of the underlying physics. These results validate the accuracy and consistency of the quantum approach in capturing bulk flow behavior, even in the presence of geometric complexity. The dashed square represents the actual location of the obstacle (grains) in the following figures.

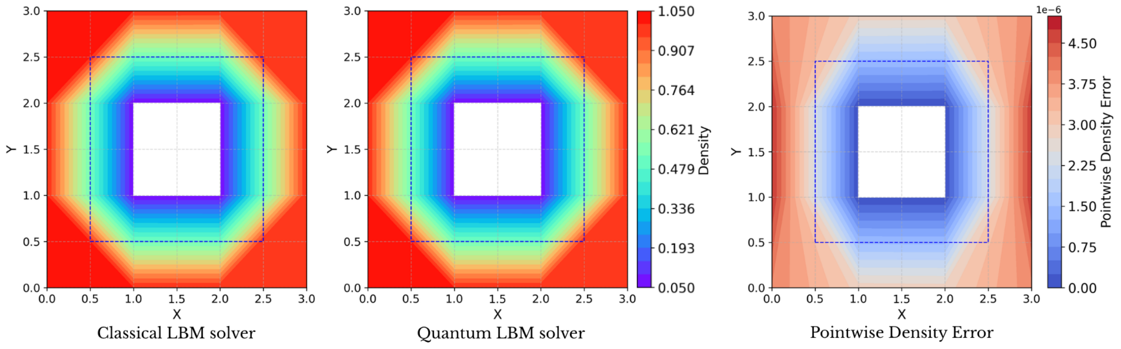


Figure 5.6 Comparison of classical and quantum steady-state density fields for a 4×4 grid with square obstacles (RMAE $= 0.0002891\%$).

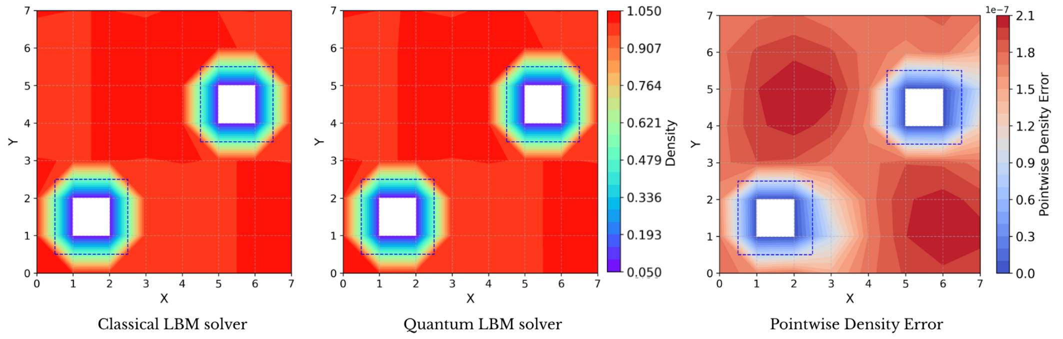


Figure 5.7 Comparison of classical and quantum steady-state density fields for an 8×8 grid with square obstacles (RMAE $= 0.0000164\%$).

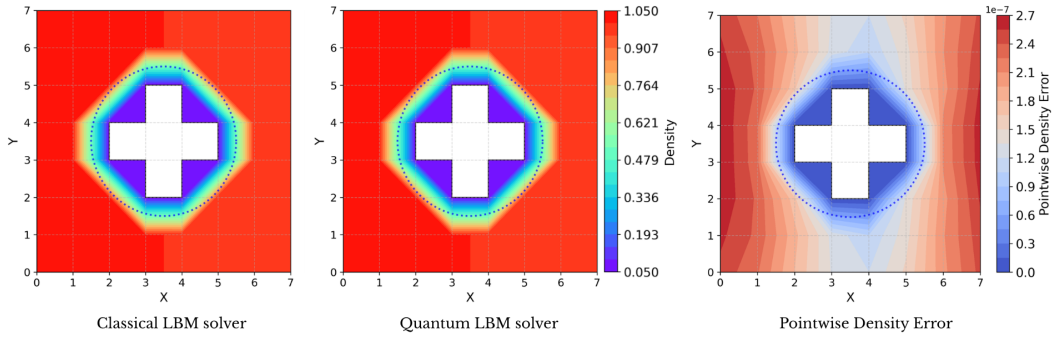


Figure 5.8 Comparison of classical and quantum steady-state density fields for an 8×8 grid with circular obstacles (RMAE $= 0.0000154\%$).

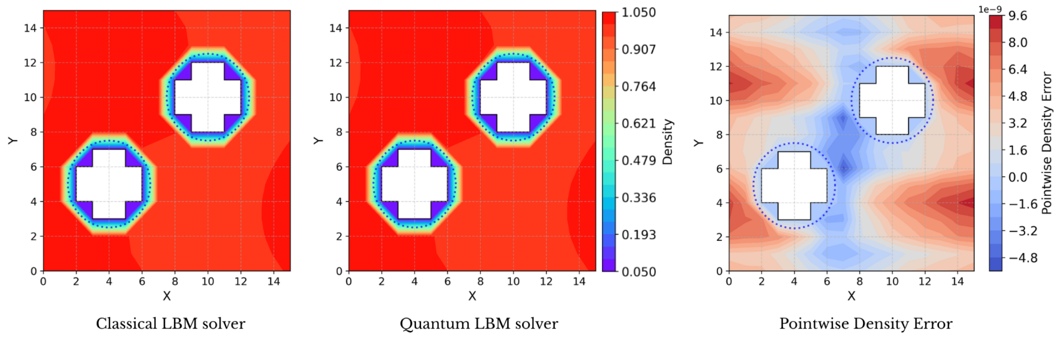


Figure 5.9 Comparison of classical and quantum steady-state density fields for a 16×16 grid with circular obstacles (RMAE = 0.0000003%).

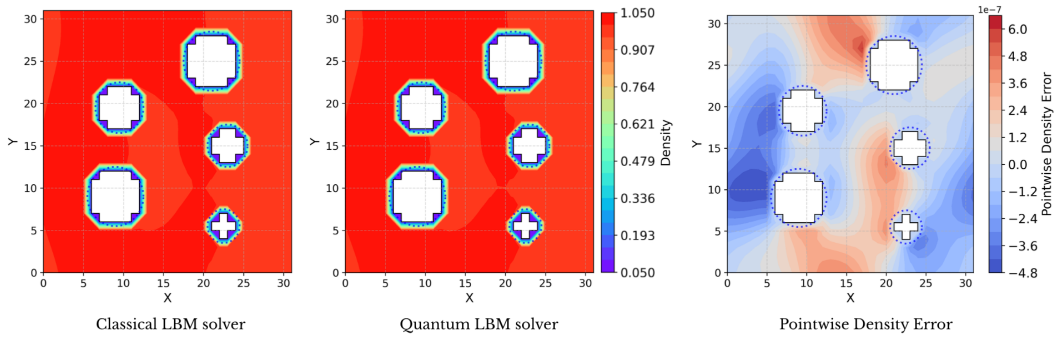


Figure 5.10 Comparison of classical and quantum density fields for a 32×32 grid with circular obstacles (up to timestep 30, RMAE = 0.0000145%).

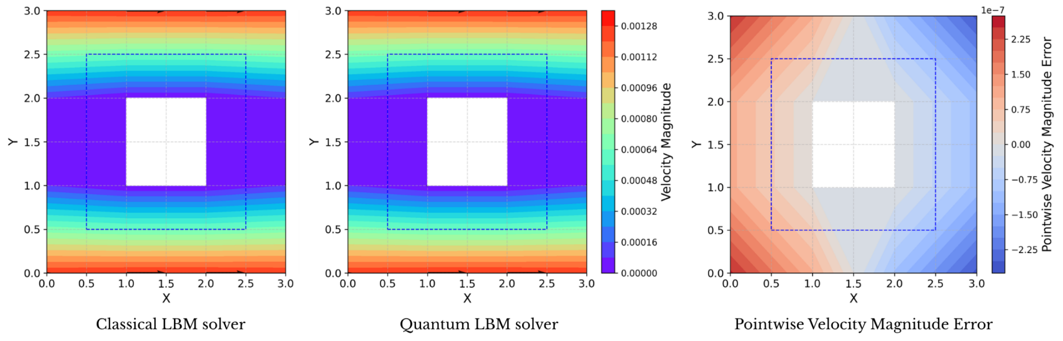


Figure 5.11 Comparison of classical and quantum steady-state velocity magnitude fields for an 4×4 grid with square obstacles (RMAE = 0.0416083%)

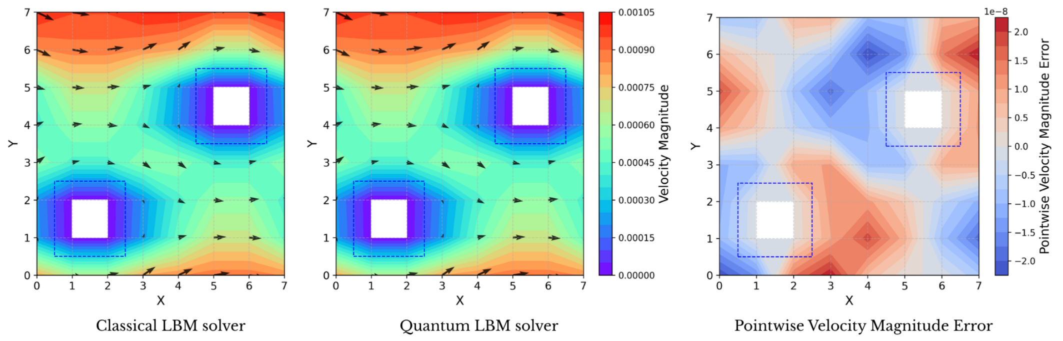


Figure 5.12 Comparison of classical and quantum steady-state velocity magnitude fields for an 8×8 grid with square obstacles (RMAE = 0.0018494%)

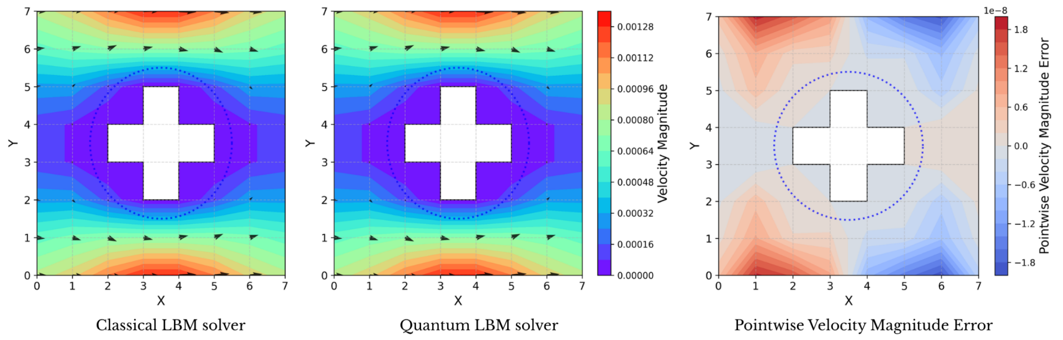


Figure 5.13 Comparison of classical and quantum steady-state velocity magnitude fields for an 8×8 grid with circular obstacles (RMAE = 0.0007081%)

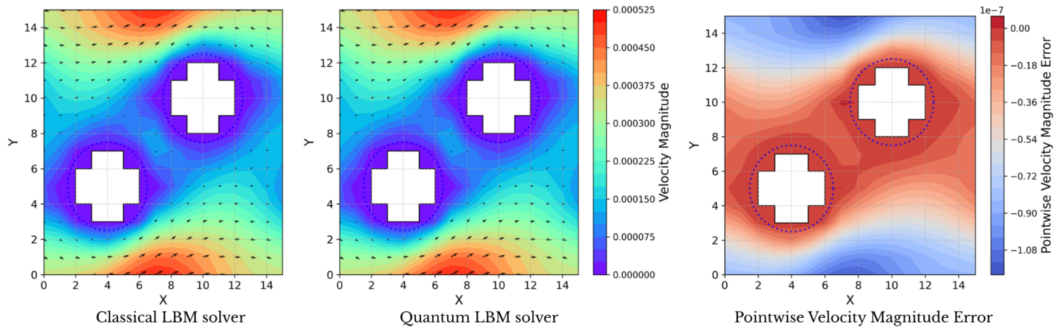


Figure 5.14 Comparison of classical and quantum steady-state velocity magnitude fields for an 16×16 grid with circular obstacles (RMAE = 0.0152197%)

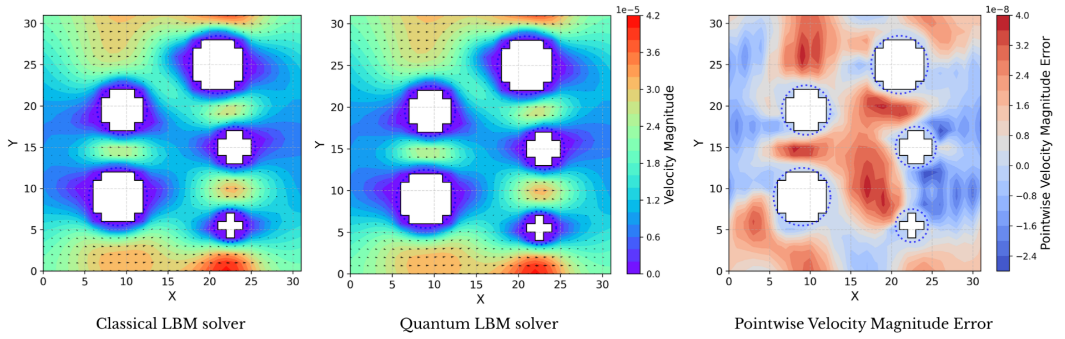


Figure 5.15 Comparison of classical and quantum velocity magnitude fields for an 32 × 32 grid with circular obstacles (upto timestep 30 RMAE = 0.1098909%)

5.3.2 Velocity field analysis and streamline visualization

The velocity components $\mathbf{u}(x, y, t)$, obtained as the first-order moments of the distribution functions, are analyzed across grid resolutions of 4×4, 8×8, 16×16, and 32×32. Each simulation incorporates internal solid obstacles of varying geometries to model porous media conditions. The resulting velocity fields from the classical and quantum LBM solvers are directly compared to assess consistency in capturing flow behavior across resolutions. The simulations follow the same boundary conditions and setup described in Sec. 5.3.1.

Fig.5.11 to 5.15 show the classical and quantum velocity magnitude fields for each grid, alongside corresponding error heatmaps. The velocity fields exhibit expected trends such as flow deceleration near obstacles and acceleration through narrow channels. These features are consistently captured by both solvers across all resolutions. Streamlines overlaid on the velocity magnitude plots visualize the influence of obstacle geometry on the direction of flow. The demonstrated agreement between classical and quantum results indicates that the quantum solver effectively captures localized hydrodynamic effects. In the 8×8 case (Fig.5.12), two square obstacles were placed deliberately to emphasize flow deflection around sharp-edged inclusions. In higher-resolution simulations, circular obstacles were introduced to better approximate curved geometries. The quantum solver reproduces the classical velocity fields with high fidelity. Error heatmaps support this agreement, with relative mean absolute errors (RMAEs) remaining below 0.1% in all cases. As expected, the RMAE values tend to decrease with increasing spatial resolution, reflecting improved numerical accuracy on finer grids. This trend is consistent with the observations in the density field analysis and further supports the consistency of the quantum solver in capturing essential flow characteristics. However, in Fig. 5.13, a slight increase in error is observed relative to adjacent resolutions. This case features two obstacles placed in close proximity (one lattice unit apart) to test the sensitivity of the model to tightly confined geometries. The resulting narrow channel appears to amplify local interactions and may increase the sensitivity of the quantum solver to bounce-back boundary conditions. Nonetheless, the error remains small, and overall solver fidelity is maintained. The dashed square represents the actual location of the obstacle (grains) in the following figures.

5.4 Scaling Behavior and Quantum Advantage Potential

The *qubit count*, *gate depth*, and *total gate operations* were systematically recorded for LBM implementation across grid sizes ranging from 4×4 to 32×32. Both fully quantum and hybrid quantum-classical schemes were evaluated to characterize their scaling behavior, quantify trade-offs between computational accuracy and quantum resource demands, and identify bottlenecks limiting near-term deployability on physical quantum hardware.

As shown in Table 5.1, the number of quantum gates required for a single timestep iteration is different for a fully-quantum to a hybrid model, with the hybrid model with reduced number of gate. The additional gates in the fully quantum scheme arise primarily from the computation of first-order moments, which in the hybrid approach (see Sec.4.4) are delegated to classical computation. This offloading significantly reduces the number of quantum operations required.

For reference, near-term quantum hardware such as MonarQ [60] supports approximately ~350 single-qubit gates (e.g., R_x, H, Z) and ~115 two-qubit gates (e.g., CNOT, CZ) per circuit execution before fidelity drops below acceptable thresholds. Given these limitations, the gate requirements of both the full and hybrid schemes exceed current feasible execution limits, confirming that running the full quantum LBM solver on real quantum hardware is not yet viable. Nonetheless, this analysis enables a concrete estimate of how close current algorithms are to practical execution and highlights areas for future optimization.

Table 5.1 Gate count for hybrid and quantum solvers

Solver Type	**Gate Count**
Quantum	23788
Hybrid	18189

The number of required qubits increases with grid resolution, as it is directly related to the number of lattice points and discrete velocity directions used in the LBM model (Table 5.2). For reference, the MonarQ QPU [60] supports up to 24 qubits, indicating that the encoded grid sizes explored in this study fall within the device's qubit capacity, however without room for ancillary operations or error correction.

Table 5.2 Number of qubits for hybrid and quantum solvers with regards to grid size

Grid size (M)	4x4	8x8	16x16	32x32
Number of qubits	9	11	13	15

Fig.5.16 shows the runtime required to complete one iteration of the LBM solver across increasing grid sizes. The classical solver exhibits near-linear scaling behavior with respect to grid size, which is expected due to the inherently local nature of LBM. Each grid cell updates based on a fixed layout of neighbors, resulting in a per-timestep computational cost that scales linearly with the number of lattice points. While further performance gains are possible through low-level memory optimizations and cache-aware programming, such enhancements were not the focus in this implementation.

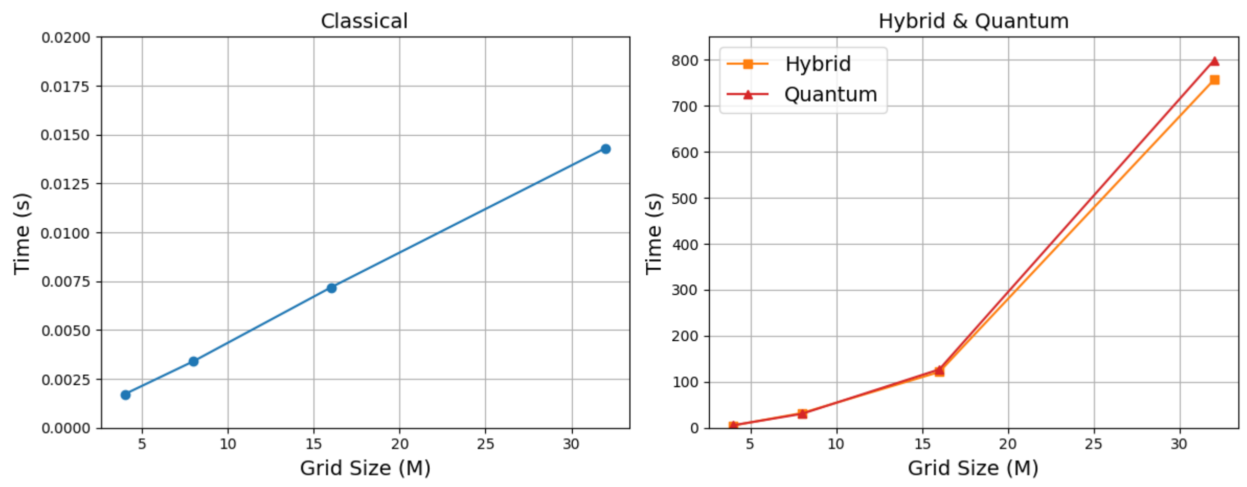


Figure 5.16 Time-to-solution vs. grid size for classical, hybrid, and full quantum solvers.

In contrast, the runtime of both the full quantum and hybrid solvers increases at a higher-than-linear rate. This growth is approximately quadratic, indicative of polynomial scaling, stemming from the increasing number of qubits required to represent the lattice and velocity space. As the number of qubits increases, the cost of statevector simulation in classical emulators grows exponentially, contributing to the observed runtime behavior. It is important to note, however, that the quantum *gate depth* remains constant across grid sizes in these experiments, as no additional interactions or obstacles were introduced. This implies that while the quantum register size increases with grid resolution (due to the need to represent more lattice sites and velocities), the circuit depth per timestep remains constant. This reflects a desirable property in the logical circuit design, where the computational workload grows spatially but not temporally, assuming no additional interactions such as obstacles are introduced.

To assess the sensitivity of each solver to geometric complexity, simulations were conducted on a fixed 8×8 grid while varying the number of embedded obstacles. Three configurations were tested, containing 1, 2, and 3 square obstacles of size 2×2. The corresponding runtime results for each solver are presented in Table 5.3.

Table 5.3 Runtime comparison across varying obstacle densities for classical, hybrid, and quantum solvers on an 8×8 grid

Number of Obstacles	**Classical Time (s)**	**Hybrid Time (s)**	**Quantum Time (s)**
1	0.004	50.863	49.000
2	0.003	50.087	51.915
3	0.008	49.828	51.989

Contrary to initial expectations, the classical solver does not exhibit a consistent increase in runtime with increasing obstacle count. This behavior arises from the unoptimized implementation of the bounce-back scheme. Instead of fusing propagation and bounce-back into a single step, common in high-performance classical LBM implementations, this approach performs them as separate operations, mirroring the structure required for quantum compatibility. Consequently, solid nodes trigger double propagation steps, which degrade overall performance. In an optimized classical solver, increasing the number of solid cells would typically reduce runtime due to fewer fluid updates; however, this efficiency is not realized here due to the conservative design constraints used for compatibility with the quantum back-end.

On the quantum side, both the fully quantum and hybrid solvers exhibit relatively stable runtimes across obstacle counts. Although each obstacle cell undergoing bounce-back introduces additional quantum gate, the total increase in runtime remains close to unchanged. The slight growth can be attributed to the growing conditional logic depth, especially in the fully quantum case. However, hybrid solvers manage this overhead more effectively by delegating such conditional operations to classical control paths. As a result, hybrid implementations demonstrate greater resilience to increases in geometric complexity, maintaining near-constant runtimes even with higher obstacle densities.

These findings suggest that while quantum and hybrid solvers currently lack runtime perfor-

mance advantages over classical solvers, they offer improved robustness when applied to more geometrically complex domains, such as porous media or structured materials—particularly valuable in the context of NISQ applications.
It is important to emphasize, however, that the results above are based on quantum algorithms executed on classical simulators. Such an approach is essential for verifying correctness but does not reflect the runtime benefits expected from real quantum hardware. To address this, estimates provided by IBMQ for the execution times of standard gates can be used to approximate the physical runtime of the same circuits. For example, IBMQ Eagle processor computes single-qubit gates such as Hadamard $\approx$ 60 ns, Pauli-X $\approx$ 100–120 ns, and the two-qubit CX gates $\approx$ 600–700 ns. By decomposing the QLBM circuit into this universal gate set and applying these hardware-level timings, a realistic runtime estimate can be obtained as shown in Table 5.4.

Table 5.4 Runtime comparison across varying obstacle densities for classical solver and quantum solver with estimated hardware runtime approximations on an 8×8 grid

Number of Obstacles	**Classical Time (ms)**	**Quantum Time (ms)**
1	4	65.32–71.47
2	3	66.00–72.15
3	8	67.38–73.46

From the data shown, although classical solvers retain an edge in the current regime, the quantum approach demonstrates the favorable scalability necessary for tackling larger and more computationally demanding fluid dynamics problems.

5.5 Summary of Results

- **Poiseuille flow verification:** both classical and quantum solvers accurately reproduce Poiseuille flow profiles, even on coarse grids. Quantum solver's correctness is validated by its close agreement with classical results.

- **Grid convergence analysis:** classical solver shows near-second-order convergence with a rate of 2.39, despite the use of double propagation. Quantum solver achieves a comparable convergence rate of 2.60, confirming numerical consistency.

- **Porous media validation:** quantum solver replicates classical density fields with a relative mean absolute error (RMAE) of less than 0.001%. Velocity fields are reproduced with RMAE below 0.1%, effectively capturing obstacle-induced effects across resolutions, confirming the physical fidelity of the quantum simulation.

- **Quantum resource analysis:** gate counts per timestep reach 23,788 for the quantum solver and drop to 18,189 for the hybrid solver due to classical offloading. Qubit demands scale with grid size, up to 15 for a 32×32 domain. However, full circuit execution remains infeasible on current hardware due to gate depth and fidelity constraints.

- **Runtime scaling:** classical solver runtime scales nearly linearly with grid size due to the locality of LBM operations, while quantum and hybrid solvers exhibit quadratic scaling from solver overheads. Notably, circuit depth remains constant with grid size, reflecting efficient logical scaling.

- **Obstacle handling:** the classical solver's runtime increases linearly with the number of solid nodes due to unoptimized bounce-back handling, whereas quantum and hybrid solvers experience only minor slowdowns. The hybrid solver, in particular, maintains more stable performance as obstacle density grows.

CHAPTER 6 CONCLUSION

This final chapter presents a summary of the key contributions and findings of the thesis. It revisits the motivations and objectives set out at the beginning, reviews the results obtained through classical and quantum LBM solvers, and discusses the broader implications of this work. The chapter also outlines the main limitations encountered during the research, particularly those related to quantum circuit complexity and simulation constraints. Finally, it proposes potential directions for future research aimed at overcoming current challenges and extending the applicability of quantum solvers in CFD.

6.1 Summary of Work

The general objective of this thesis was to develop an efficient quantum algorithm based on LBM for simulating two-dimensional fluid flows through porous media in the Stokes regime, that can be readily extended to 3D. This objective was successfully attained through the design, implementation, and verification of a QLBM framework that reproduces key algorithm features of classical LBM fluid solvers. The specific objectives outlined in Chapter 3 were addressed as follows:

1. **Velocity–pressure formulation of QLBM** was developed, replacing the traditional streamfunction–vorticity approach. This choice lays the groundwork for extending QLBM to three-dimensional domains in future studies.
2. **Stress tensor recovery in D2Q9 lattices:** The QLBM framework was successfully extended to support diagonal population propagation, enabling full reconstruction of the stress tensor. By activating all nine discrete velocity directions of the D2Q9 lattice, including diagonals, the model captures both normal and shear stress components. This enhancement ensures accurate representation of key physical effects and fulfills the objective of designing a stress-resolving quantum LBM. Moreover, the approach is readily extensible to higher-order lattice models (e.g., D2Q13, D3Q27), offering a scalable pathway for simulating more complex flow physics.
3. **Body force integration:** source terms, such as uniform body forces, were incorporated into the quantum algorithm. This enabled the modeling of pressure-driven flows relevant to porous media flows.

4. **Boundary condition enforcement and physical fidelity:** halfway bounce-back schemes were implemented to enforce no-slip conditions at solid walls, and periodic boundaries were used to simulate repeating domains. These were embedded directly into the quantum circuit logic, maintaining their spatial locality. The quantum solver accurately replicates classical results, with relative mean absolute errors (RMAE) below 0.001% for density fields and below 0.1% for velocity fields. Accurate reproduction of obstacle-induced effects across grid resolutions confirms the solver's physical fidelity and robustness.

5. **Quantum subroutine for first-order moment calculation:** Dedicated quantum subroutines were successfully developed to compute first-order velocity moments from quantum states.

6. **Minimizing classical dependence:** The LBM pipeline was designed to maximize quantum-side computation, reducing classical post-processing to only essential steps. This hybrid formulation shifts more workload onto the quantum circuit, bringing the approach closer to feasible deployment on NISQ-era simulators and hardware.

Collectively, these contributions establish a flexible and modular quantum LBM solver capable of simulating two-dimensional viscous flows, verifying its outputs against classical results, and laying the foundation for hybrid classical-quantum fluid dynamics solvers.

6.2 Limitations

Despite the demonstrated accuracy and functionality of the developed quantum LBM framework, several limitations persist:

1. **Encoding overhead:** while amplitude encoding was adopted for its compact qubit requirement, it remains computationally intensive. The preparation of arbitrary quantum states, involves complex unitary transformations, resulting in deeper circuits and increased compilation time.

2. **Lack of quantum state persistence:** quantum circuits do not allow intermediate state storage or persistent memory across timesteps. As a result, after each iteration, the quantum state must be measured (collapsed), the required macroscopic observables (e.g., velocity, pressure) extracted, and the circuit fully reinitialized for the next timestep. This disrupts temporal coherence and necessitates frequent classical–quantum interfacing, increasing overhead and limiting simulation continuity.

3. **Subroutine overhead, normalization, and noise sensitivity:** to compute multiple macroscopic quantities (e.g., density, velocity components), distinct quantum subroutines are invoked at each timestep. While this modular structure supports parallelism in theory, it introduces normalization overheads that can accumulate numerical errors and inflate circuit complexity. Furthermore, repeated normalization and measurement steps reduce efficiency and hinder scalability. While current results are based on ideal statevector simulations (noise-free), execution on real or noisy quantum backends would exacerbate these issues, as gate errors and decoherence would degrade accuracy across successive subroutine calls.

4. **Bounce-back gate cost:** every node undergoing bounce-back requires one to two additional quantum gates. As the number of obstacle nodes increases, the gate count and depth grow rapidly, significantly impacting circuit execution time and fidelity.

5. **Diagonal propagation complexity:** in D2Q9 models, diagonal propagation is implemented using combinations of quantum walk operators. These compound operations effectively double the gate count compared to orthogonal (D2Q5) movements, contributing further to circuit depth.

6. **Redundant propagation for bounce-back nodes:** In the current implementation, bounce-back is applied after the propagation step, causing obstacle nodes to undergo two propagation operations per timestep,one during propagation itself, and once in reverse during bounce-back. While this structure preserves compatibility with the quantum formulation, it introduces algorithmic redundancy. However, results show that this does not significantly affect numerical accuracy; in fact, the observed convergence rate exceeds second-order. Nonetheless, with a more optimized bounce-back implementation, the solver could achieve closer-to-ideal convergence rates and reduced runtimes, particularly for geometries with higher obstacle density.

7. **Hardware constraints and noise limitations:** due to current hardware limitations, including restricted qubit availability, circuit depth constraints, and significant noise-induced decoherence, it was not possible to run the quantum LBM model on noisy simulators or real quantum devices. As a result, all results were obtained using noiseless statevector simulations. Running on actual quantum hardware in its current state would likely produce unreliable or meaningless outputs.

6.3 Future Work

Building on the foundation established in this thesis, several avenues exist for improving and extending the current quantum LBM framework:

1. **Optimized encoding schemes:** Exploring alternative quantum state encoding methods such as angle-based encoding offers a promising pathway to reduce gate count and circuit depth. These approaches map physical quantities like velocity or pressure directly to quantum rotation angles, potentially simplifying quantum subroutine construction. While such schemes have been studied for scalar quantities or basic fluid properties, no prior work has successfully applied them to velocity-pressure formulations that include diagonal propagation or full stress tensor recovery in D2Q9 models. This remains an open direction for future research, particularly in the context of quantum fluid solvers.

2. **Extension to 3D geometries:** The current quantum LBM framework is designed to be extensible and can be adapted to three-dimensional models such as D3Q15, D2Q19 or D3Q27. Expanding to 3D would enable simulation of more realistic fluid systems, though it would also increase qubit requirements and circuit complexity. Future work should focus on efficient quantum encoding of 3D lattice directions and integration of appropriate boundary conditions.
   To illustrate the qubit scaling, a comparison can be made between a D2Q9 lattice on a 4×4 grid and a D3Q27 lattice on a $4 \times 4 \times 4$ grid. In the D2Q9 case, as presented in Section 4.3.2, the total number of qubits required is 10. Following the same methodology, for the D3Q27 lattice on a $4 \times 4 \times 4$ grid, r_x, r_y and r_z each require $\lceil \log_2 4 \rceil = 2$ qubits. For d, despite the total number of discrete velocities being 27, due to the structure of the bounce back segment, the total discrete velocities now are $(2 \times 27) - 1 = 53$. Hence the total number of qubits for this register are $\lceil \log_2 53 \rceil = 6$. Finally, including the singular ancilla qubit used for overall calculation, the total number of qubtis for D3Q27 are 13. Additionally, the transition from D2Q9 to D3Q27 does not require any modification to the gate structures themselves; only the register sizes change. A schematic representation of the proposed D3Q27 register architecture corresponding to the above calculation is shown in Fig. 6.1.

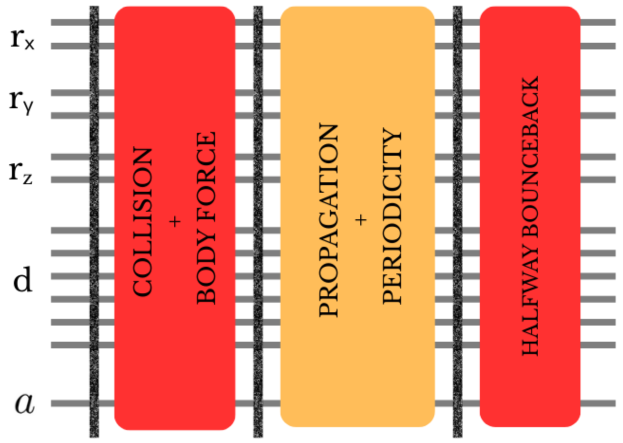


Figure 6.1 D3Q27 circuit schematic

3. **Efficient diagonal propagation:** developing specialized gates or streamlined walk operators to implement diagonal movements more efficiently would significantly reduce gate overhead in D2Q9 and higher-order lattices.

4. **Improved bounce-back implementation:** in classical LBM implementations, an optimized bounce-back scheme can be achieved by restricting propagation to nodes within the fluid domain, excluding obstacle-wall nodes that satisfy the no-slip condition. This selective propagation eliminates unnecessary updates to solid nodes and avoids redundant computations. While this approach is straightforward in the classical context, its direct translation to the quantum framework is non-trivial. In the quantum case, additional gates are required to first identify fluid-domain nodes, followed by conditional propagation applied only to these nodes. The subsequent bounce-back step still requires identifying obstacle-wall nodes and performing the reflection operation. Although this method resolves the *double propagation* issue, it introduces a significant increase in total gate count, leading to higher circuit depth and runtime. Therefore, refining the bounce-back implementation requires a new technique that effectively mitigates double propagation without imposing substantial quantum gate costs.

5. **Noise and decoherence analysis:** running the quantum LBM algorithm on noisy simulators or actual quantum hardware would allow for evaluation under realistic conditions. Understanding decoherence effects is essential to assessing the practicality of such quantum solvers on NISQ-era devices.

The development of quantum algorithms is still in its infancy, yet this study demonstrates the vast potential that quantum computing holds. The ability to model physically meaningful phenomena using quantum subroutines highlights a promising direction for computational science. Although the domains analyzed in this work are relatively small, limiting observable quantum advantages compared to classical approaches, the true benefits of quantum computation are expected to emerge when simulating much larger domains on actual quantum hardware. There, the inherent parallelism and entanglement of quantum processors can be fully leveraged to tackle problems that are challenging for classical machines. Current hardware limitations, such as qubit count, gate fidelity, and noise resilience, pose significant barriers to practical deployment, but progress at the algorithmic level remains encouraging. Importantly, while the long-term goal is to realize fully quantum algorithms, employing quantum processors as co-processors for targeted computations offers a pragmatic and effective strategy for benchmarking quantum advantage over classical solvers. This hybrid approach not only maximizes current hardware capabilities but also provides valuable insight into performance scaling and resource demands. As quantum processors evolve, the algorithms developed and validated today will be well-positioned to utilize future advancements, potentially transforming the landscape of scientific computing in the years to come.

REFERENCES

[1] H.-Z. Dresden-Rossendorf, “Helmholtz-zentrum dresden-rossendorf,” 2014. [Online]. Available: https://www.hzdr.de/db/Cms?pOid=42535&pNid=3367

[2] D. H. Rothman and S. Zaleski, *Lattice-Gas Cellular Automata: Simple Models of Complex Hydrodynamics.* Cambridge: Cambridge University Press, 1997.

[3] J. Bear, *Dynamics of Fluids in Porous Media.* New York: American Elsevier Publishing Company, 1972.

[4] I. Berre, F. Doster, and E. Keilegavlen, “Flow in fractured porous media: A review of conceptual models and discretization approaches,” *arXiv preprint arXiv:1805.05701*, 2018.

[5] T. Knight *et al.*, “Fabrication of a multi-layer three-dimensional scaffold with controlled porous micro-architecture for application in small intestine tissue engineering,” *Cell Adhesion & Migration*, pp. 267–274, 2013.

[6] M. Haase and J. Banhart, “60 years of open-celled ceramics based on replica technique: Applications, obstacles, and opportunities,” *Advanced Engineering Materials*, 2024.

[7] N. Zamel, X. Li, and E. Kjeang, “Water management in polymer electrolyte membrane fuel cells: Diffusion layers, channel flow, and porous media,” *Applied Energy*, pp. 1027–1043, 2015.

[8] T. Krüger *et al.*, *The Lattice Boltzmann Method: Principles and Practice.* Cham: Springer, 2017.

[9] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge: Cambridge University Press, 2010.

[10] M. Ivezic. (2023, Nov.) Taxonomy of quantum computing: Modalities & architectures. [Online]. Available: https://postquantum.com/quantum-modalities/taxonomy-modalities/?utm_source=chatgpt.com

[11] *Qiskit: An Open-source Framework for Quantum Computing*, Qiskit Development Team, 2024. [Online]. Available: https://qiskit.org/documentation/

[12] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” *SIAM Journal on Computing*, p. 1484–1509, 1997.

[13] L. K. Grover, “A fast quantum mechanical algorithm for database search,” in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC ’96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[14] A. W. Harrow, A. Hassidim, and S. Lloyd, “Quantum algorithm for linear systems of equations,” *Phys. Rev. Lett.*, p. 150502, 2009.

[15] A. Montanaro, “Quantum speedup of monte carlo methods,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, p. 20150301, 2015.

[16] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” 2014.

[17] D. Herman *et al.*, “Quantum computing for finance,” *Nature Reviews Physics*, p. 450–465, 2023.

[18] P. Saguat, “Classical lattice-boltzmann methods for fluid dynamics.” [Online]. Available: https://www.vki.ac.be/index.php/events-ls

[19] G. K. Batchelor, *An Introduction to Fluid Dynamics.* Cambridge University Press, 1967.

[20] R. B. Bird, W. E. Stewart, and E. N. Lightfoot, *Transport Phenomena*, 2nd ed. New York: John Wiley & Sons, 2002.

[21] D. Vidal, R. Roy, and F. Bertrand, “A parallel workload balanced and memory efficient lattice-boltzmann algorithm with single unit bgk relaxation time for laminar newtonian flows,” *Computers Fluids*, pp. 1411–1423, 2010.

[22] Z. Guo and C. Shu, *Lattice Boltzmann Method and Its Applications in Engineering.* World Scientific, 2013.

[23] E. Madelung, “Eine anschauliche deutung der gleichung von schrödinger,” *Naturwissenschaften*, pp. 1004–1004, 1926.

[24] L. Lapworth, “Implicit hybrid quantum-classical cfd calculations using the hhl algorithm,” 09 2022.

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

[26] M. Schalkers and M. Möller, "On the importance of data encoding in quantum boltzmann methods," *Quantum Information Processing*, 2024.

[27] F. Gaitan, "Finding flows of a navier–stokes fluid through quantum computing," *npj Quantum Information*, 2020.

[28] F. Oz *et al.*, "Solving burgers equation with quantum computing," *Quantum Information Processing*, 2021.

[29] O. Kyriienko, A. E. Paine, and V. E. Elfving, "Solving nonlinear differential equations with differentiable quantum circuits," *Physical Review A*, 2021.

[30] P. Pfeffer, F. Heyder, and J. Schumacher, "Quantum reservoir computing of thermal convection flow," *Physical Review Research*, 2022.

[31] S. Succi *et al.*, "Ensemble fluid simulations on quantum computers," 2023.

[32] A. Broadbent and E. Kashefi, "Parallelizing quantum circuits," *Theoretical Computer Science*, pp. 2489–2510, 2009.

[33] O. D. Matteo and M. Mosca, "Parallelizing quantum circuit synthesis," *Quantum Science and Technology*, 2016.

[34] S. Succi and R. Benzi, "Lattice boltzmann equation for quantum mechanics," *Physica D: Nonlinear Phenomena*, pp. 327–332, 1993.

[35] J. Yepez and B. Boghosian, "An efficient and accurate quantum lattice-gas model for the many-body schrödinger wave equation," *Computer Physics Communications*, pp. 280–294, 2001.

[36] J. Yepez, "Quantum computation of fluid dynamics," in *Quantum Computing and Quantum Communications*, ser. Lecture Notes in Computer Science, 1998. [Online]. Available: https://www.phys.hawaii.edu/~yepez/papers/publications/pdf/1999LectNotesCompSciVol1509Pg35.pdf

[37] ——, "Quantum lattice-gas model for computational fluid dynamics," *Physical Review E*, 2001.

[38] ——, "Quantum lattice-gas model for the burgers equation," *Journal of Statistical Physics*, pp. 203–224, 2002.

[39] M. A. Pravia *et al.*, "Experimental demonstration of quantum lattice gas computation," *Quantum Information Processing*, pp. 381–403, 2003.

[40] B. N. Todorova and R. Steijl, "Quantum algorithm for the collisionless boltzmann equation," *Journal of Computational Physics*, 2020.

[41] M. Schalkers and M. Möller, "Efficient and fail-safe quantum algorithm for the transport equation," *Journal of Computational Physics*, 2024.

[42] W. Itani and S. Succi, "Analysis of carleman linearization of lattice boltzmann," *Fluids*, 2022.

[43] C. Sanavio and S. Succi, "Quantum lattice boltzmann–carleman algorithm," 2023.

[44] Y. Ito, Y. Tanaka, and K. Fujii, "How to map linear differential equations to schrödinger equations via carleman and koopman-von neumann embeddings for quantum algorithms," 2023.

[45] L. Budinski, "Quantum algorithm for the advection–diffusion equation simulated with the lattice boltzmann method," *Quantum Information Processing*, 2021.

[46] ——, "Quantum algorithm for the navier–stokes equations," 2021.

[47] N. Fonio *et al.*, "Adaptive lattice gas algorithm: Classical and quantum implementations," 2025.

[48] A. D. B. Zamora *et al.*, "Efficient quantum lattice gas automata," 2024.

[49] ——, "Float lattice gas automata: A connection between molecular dynamics and lattice boltzmann method for quantum computers," 2025.

[50] M. Lee *et al.*, "A multiple-circuit approach to quantum resource reduction with application to the quantum lattice boltzmann method," 2024.

[51] M. Schalkers and M. Möller, "Momentum exchange method for quantum boltzmann methods," *Computers Fluids*, 10 2024.

[52] V. Shende, S. Bullock, and I. Markov, "Synthesis of quantum-logic circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1000–1010, 2006.

[53] A. Javadi-Abhari *et al.*, "Quantum computing with qiskit," *arXiv preprint arXiv:2405.08810*, 2024.

[54] A. M. Childs and N. Wiebe, "Hamiltonian simulation using linear combinations of unitary operations," *Quantum Information & Computation*, pp. 901–924, 2012.

[55] G. H. Low and I. L. Chuang, “Hamiltonian simulation by qubitization,” *Quantum*, 2019.

[56] S. Chakraborty, A. Gilyén, and S. Jeffery, “The power of block-encoded matrix powers: Improved regression techniques via faster hamiltonian simulation,” *arXiv preprint arXiv:1804.01973*, 2019.

[57] A. Ambainis *et al.*, “One-dimensional quantum walks,” *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pp. 37–49, 2001.

[58] S. Succi, *The Lattice Boltzmann Equation: For Complex States of Flowing Matter.* Oxford University Press, 2018.

[59] Y. Xiao *et al.*, “Quantum lattice kinetic scheme for solving two-dimensional and three-dimensional incompressible flows,” *arXiv preprint arXiv:2505.10883*, 2025. [Online]. Available: https://arxiv.org/abs/2505.10883

[60] Digital Research Alliance of Canada, “MonarQ Quantum Processor - Documentation,” https://docs.alliancecan.ca/wiki/MonarQ/en, 2024.

APPENDIX A APPENDICES

Dirac Notation and Basis States

Quantum mechanics and quantum computing use a compact mathematical language called **Dirac notation (bra-ket notation)** to describe quantum states.

Qubit and Computational Basis States

A **qubit** is a quantum version of a classical bit. It can be in state $|0\rangle$, $|1\rangle$, or a superposition of both:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{where } \alpha, \beta \in \mathbb{C},\ |\alpha|^2 + |\beta|^2 = 1$$

The states $|0\rangle$ and $|1\rangle$ form the standard **computational basis**:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

In this notation: - $|\cdot\rangle$ is called a **ket** (column vector) - $\langle\cdot|$ is called a **bra** (row vector; Hermitian conjugate)

For example:

$$\langle 0| = \begin{bmatrix} 1 & 0 \end{bmatrix}, \quad \langle 1| = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Multi-Qubit Basis States

Multi-qubit systems are described using the tensor product of single-qubit basis states. For two qubits, the computational basis includes states like:

$$|00\rangle = |0\rangle \otimes |0\rangle = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad |01\rangle = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \quad \text{etc.}$$

Superposition and Measurement

A qubit in a superposition exists in both $|0\rangle$ and $|1\rangle$ until measured. Upon measurement, it collapses to $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$.

Native Quantum Gate Matrices

Quantum gates are unitary operations (i.e., $U^\dagger U = I$) that evolve qubits. Some common **native gates** (i.e., implemented directly on hardware) and their matrix forms are:

Table A.1 Common native quantum gates

Gate	**Matrix Representation**	**Description**
I	$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$	Identity gate
X	$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$	Bit-flip (NOT gate)
Y	$\begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}$	Bit-and-phase-flip
Z	$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$	Phase-flip
H	$\frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$	Hadamard (creates superpositions)
S	$\begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$	Phase gate ($Z^{1/2}$)
T	$\begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}$	$\pi/8$ gate
$R_x(\theta)$	$\begin{bmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$	Rotation around X
$R_y(\theta)$	$\begin{bmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{bmatrix}$	Rotation around Y
$R_z(\theta)$	$\begin{bmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{bmatrix}$	Rotation around Z
CNOT	$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$	2-qubit gate: flips target if control is $|1\rangle$

These gates are the fundamental building blocks of quantum algorithms and can be combined to build more complex unitary operations.